IJIRCCE

e-ISSN: 2320-9801 | p-ISSN: 2320-9798

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

Volume 10, Issue 6, June 2022

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.165

International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.165 |

|| Volume 10, Issue 6, June 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1006056 |

A Reviev on Droid detector Using Android

Suprita Pujar, Mrs. Usha C

Master of Computer Applications , UBDTCE Davanagere, Karnataka , India

Faculty, Department of MCA, UBDTCE Davanagere, Karnataka, India

ABSTRACT: Smartphones and mobile tablets are rapidly becoming indispensable in daily life. Android has been the most popular mobile operating system since 2012. However, owing to the open nature of Android, countless malicious are hidden in a large number of benign apps in Android markets that seriously threaten Android security. Deep learning is a new area of machine learning research that has gained increasing attention in artificial intelligence. In this study, we propose to associate the features from the static analysis with features from dynamic analysis of Android apps and characterize malicious using deep learning techniques. We implement an online deep-learning-based Android malicious detection engine (Droid detector) that can automatically detect whether an app is a malicious or not. With thousands of Android apps, we thoroughly test Droid detector and perform an in depth analysis on the features that deep learning essentially exploits to characterize malicious. The results show that deep learning is suitable for characterizing Android malicious and especially effective with the availability of more training data. Droid detector can achieve 96.76% detection accuracy, which outperforms traditional machine learning techniques. An evaluation of ten popular anti-virus software's demonstrates the urgency of advancing our capabilities in Android malicious detection.

KEYWORDS: Android security, malicious detection, characterization, deep Learning, association rules mining.

I. INTRODUCTION

Android dramatically surpassed a billion shipments of its devices in 2014 and has remained the No.1 mobile operating system since 2013

Google Play Store and other third-party markets, play an important role in the popularity of Android devices, countless of malicious being hidden behind a large number of benign apps that seriously threatens users' security and privacy. Moreover, a report from McAfee Labs reveals that 3.73 million pieces of mobile malware were identified in 2013, increasing an astounding 197% from the end of 2012. Consequently, an urgent need arises to develop powerful solutions for Android malicious detection. Unfortunately, the Android market currently has no such solution. Today, the main countermeasure to defence against malicious on Android platforms is a risk communication mechanism that warns users about the permissions required before installing each app. This mechanism is rather ineffective, Droid detector: Android Malicious Characterization and Detection Using Deep Learning technique, thus requiring too much technical knowledge for a user to be able to differentiate malicious from benign apps. Note that both a benign and a malicious app may require the same permissions and are thus indistinguishable via this permission-based mechanism. In general, permission-based approaches are developed primarily for risk assessment rather than malicious detection. Clearly, a better characterization of Android malicious would achieve a better accuracy in their detection. Droid Ranger and Risk Ranker, two typical signature-based methods, try to characterize malicious using specific patterns in the bytecode and Application Program Interfaces (API) calls. However, these signature-based methods can be easily evaded by bytecode-level transformation attacks. It has been pointed out that signature-based methods, it cannot catch several types of code-loading-technique-based Android malicious. Previous research has revealed that Android malware is rapidly evolving to circumvent signature based characterizations and thus calls for the development of next-generation anti-mobile-malware solutions. Android malware evidently cannot be adequately characterized using only specific patterns (signatures). In view of this situation, machine learning-based methods are being proposed to characterize Android malicious that extract features by the static or dynamic analysis of Android apps and learn the distinctions between malicious and benign apps automatically. In particular, these machine-learning-based methods can avoid the need to manually craft and update detection

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.165 |

|| Volume 10, Issue 6, June 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1006056 |

rules, which is crucial for keep pace with the variety of Android malicious. We describe our development of a deep-learning-based Android malicious detection engine (Droid detector) that has been put online for user testing and can automatically detect whether an app is a malicious or not.

III. BACKGROUND

In this section, we introduce malicious detection methods and a common limitation of machine/deep learning-based Android malicious detection approaches, the mainstream of malicious detection approaches, that hinders practical uses of them.

DETECTING ANDROID MALICIOUS

Android malicious detection approaches can be categorized into two groups based on analysis methods (i.e., dynamic analysis and static analysis) used to collect features of malicious: (1) dynamic analysis-based malicious detection approaches and (2) static analysis-based ones.

Dynamic analysis-based malicious detection approaches have an advantage over static analysis-based approaches in analysing concrete behaviours of malicious. Also, they have another advantage of analysing malicious equipped with anti-analysis mechanisms such as obfuscation. However, typically the dynamic analysis method consumes a lot of resources and time because we actually need to execute applications.

On the other hand, static analysis-based malicious detection approaches identify features of malicious without executing them, and thus, the cost for analysing each application is much lower than dynamic analysis-based approaches in general. Because of the advantage of using less computing resources and high accuracy in static analysis-based malicious detection approaches, most malicious detection approaches employ the static analysis method for extracting malicious features.

Typical features used for static analysis-based malicious detection approaches

The first step to develop a malicious detection system is to decide features of malicious to distinguish them from benign applications. Typically, developer-written descriptions, user reviews, permissions, opcode and APIs are used as such features.

Developer-written descriptions:A couple of research work employed developer-written descriptions on applications as a key feature for detectingmalicious. However, detecting malicious based on developer-written descriptions is not reliable because inferring accurate execution behaviours of applications is unlikely possible.

User reviews: Among Android malicious detection approaches, there were attempts that employ user reviews as an important feature. However, similar to the malicious detection approaches that use developer-written descriptions, the accuracy is not high enough to be used in a practical manner because user reviews usually do not contain concrete explanations on applications that can be used for detecting malicious.

Opcode: Several previous work showed there are common patterns of opcode that can be used to classify malicious applications. They used common patterns of opcode such as *move* and *invoke* of bytecode in malicious applications.

Permissions: There have been many research work for detecting malicious based on permissions that applications require (e.g., a user's location, phone information, a mobile device's network status etc.). These approaches detect malware by using commonly used permissions such as network permission with users' location in malicious applications. However, Avdiienko et al. showed that similar tomalicious, most benign Android applications access sensitive information of users and use a lot of permissions that are also typically used inmalicious. Consequently, permission-based malicious detection approaches could incur a high false positive rate.

APIs: Many approaches attempted to classify malicious applications based on APIs used in them. By analysing APIs used in some applications, we can understand functionalities that the application provides to users. For example, if an application uses APIs such as android.telephony and android.telecom, we can know that the

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.165 |

|| Volume 10, Issue 6, June 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1006056 |

application would monitor a mobile phone's network status and manages phone calls. As such, Android APIs provides functional information about what an application does. Therefore, we can infer an application's behaviour by using APIs used in the application. However, if we only use APIs as a key feature for identifyingmalicious, we can have high false positives because analysing APIs does not provide an application's concrete behaviours and there are a lot of common APIs used in both benign and malicious applications.

IV. METHODOLOGY

in Fig. 1. Droid detector has been open online for user testing and can automatically detect whether a submitted app is a malicious or not. Once the. apk file of an app is submitted, Droid detector checks its integrity and determines whether it is a complete, correct, and legitimate Android application. Next, Droid detector executes a static analysis to obtain the permissions and sensitive APIs that are used by this app. Then, Droid detector executes a dynamic analysis by installing and running this app in Droid Box for a fixed period of time. In this way, Droiddetector identifies the dynamic behaviours that are being performed.

We have completely automated the static and dynamic analyses of Droid detector. Once the total 192 binary features described in Section 2 have been collected, they are input in the deep learning model for classification. The detection results, including detailed information from the integrity check and both analyses, are then reported to the users. Since the new types of apps are constantly emerging, we have designed two crawler modules. One is used for crawling benign apps from the Google Play Store and the other is used for crawling malware from well-known malware sources. Using this strategy, we expectDroid detector to keep pace with the evolution of Android malicious.

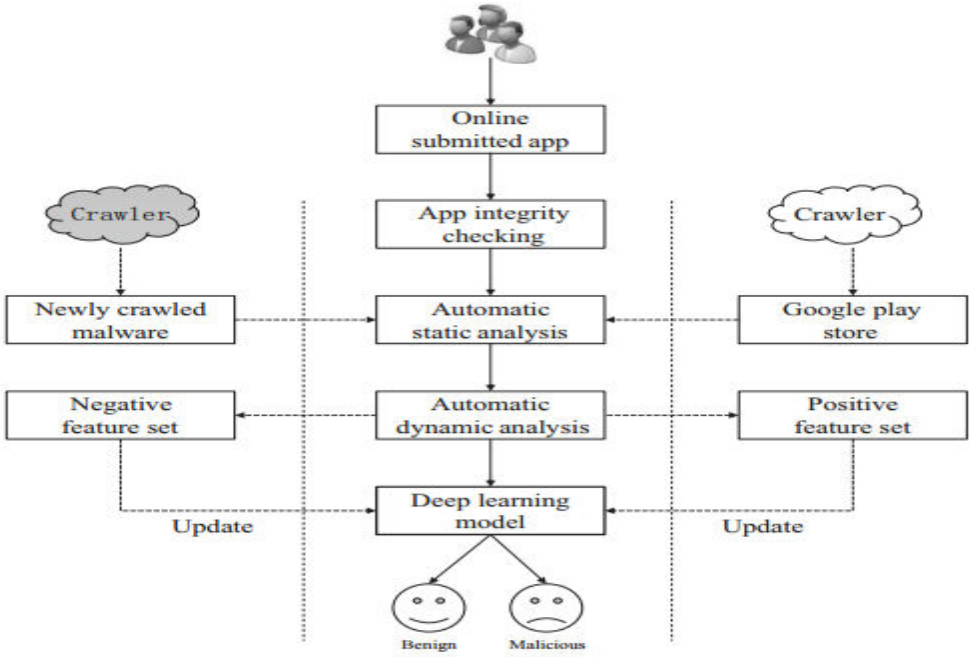

Fig 1 Framework of Droid detector

The schematic representation of the architecture of this work (Figure 1) shows the first phase starting with the mixture of malware and benign Android Application Package (APK) files taken from a dataset that is publicly avail-able to everyone in [15]. Then, we extract the static features by our own Python script written in Jupyter notebook environment [20]. The features are API-calls and permissions. These features are formatted and stored in Comma- Separated Values (CSV) file as a data frame that serves the training process. Finally, we test all classifiers using 10-folds cross-validation, in addition to calculating the appropriate metrics for the evaluation.

Fig. 1: Malware detection phases

3.1. Dataset

International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.165 |

|| Volume 10, Issue 6, June 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1006056 |

Indeed, there is a need for a comprehensive and reliable dataset to help deep and machine learning models test and validate the Android malware detection system. Therefore, this study has used part of the CICAndMal2017 dataset, which has been produced and published by Lashkari et al. [15], and is available on the Canadian Institute of Cybersecurity website [21]. It was gathered based on real experiments. Our dataset is a combination of 347 benign samples and 365 malware samples of Android applications, where the malwares consist of groups called families, namely; Adware, Ransomware, Scareware, and SMSware. Each family performs specific attacks that is related to its name, where the Adware and SMSware sends annoying adds and SMS during running the application, while Scareware and Ransomware ask the users for fees and ransoms so that attackers do not crash their system or stall their private data.

3.2. Feature Extraction and Preprocessing

In term of classification, it is essential to choose features that indicate which class the new record would belong to. From this standpoint, the permissions and API-calls are extracted from all Android applications, and both were included as features in the dataset. Androguard is a full package tool designed to interact with Android files and re-stricted only to python environments [22]. It can be used as a tool for reverse engineering single Android applications. The Androguard tool is used to analyze APK files by separately extracting the DEX file permissions for each APK file. Hence, we constructed a data frame containing features (columns) and applications (rows), where each column represents specific permission or API-call with binary values, while rows represent both malware and benign APK files.

3.3. The Classifier Framework

Traditional machine learning classifiers often show high performance in dealing with labeled data [23]. We use multiple machine learning classifiers, namely, Support Vector Machine (SVM), K-Nearest Neighbors (KNN), De-850 Omar N. Elayan et al. / Procedia Computer Science 184 (2021) 847–852 The schematic representation of the architecture of this work (Figure 1) shows the first phase starting with the mixture of malware and benign Android Application Package (APK) files taken from a dataset that is publicly avail- able to everyone in [15]. Then, we extract the static features by our own Python script written in Jupyter notebook environment [20]. The features are API-calls and permissions. These features are formatted and stored in Comma-Separated Values (CSV) file as a data frame that serves the training process. Finally, we test all classifiers using 10-folds cross-validation, in addition to calculating the appropriate metrics for the evaluation.

IV. RESULTS

We have conducted our experiments on two types of models, traditional machine learning classifiers and deeplearning. First, we train the classifiers using the dataset, and then we perform the testing and evaluation. In both experiments, the classifications are based on the features extracted from permissions and API-calls.

4.1. Traditional Machine Learning

Specifically, we have chosen SVM, KNN, DT, RF, and NB to build a useful model in detecting malicious applications, and can assist with both classification and regression tasks. Table 1 shows the classifiers performance in detecting harmful Android applications.

Table 1: The performance of the tested algorithms in detecting Android malware.

Metrics	SVM	KNN	DT	RF	NB	GRU
Accuracy	96.2%	97.2%	96.6%	97.8%	93.9%	98.2%
Precision	98.1%	98.3%	96.2%	98.8%	94.3%	96.9%
Recall	94.4%	96.0%	97.3%	97.2%	93.5%	99.2%
F-measure	96.2%	97.1%	96.6%	97.9%	93.9%	98.0%

RF (Random Forest) classifier has achieved the highest level of accuracy with 97.8% score, and the highest score of correct predictions, where the F-score showed that it was able to predict 97.9% of the dataset correctly. Also, RF classifier was able to predict 97.2% of the malicious samples, correctly. Regardless to that, RF

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.165 |

|| Volume 10, Issue 6, June 2022 ||

| DOI: 10.15680/IJIRCCE.2022.1006056 |

classifier has achieved the highest level of precision, in which it was able to identify 98.8% of the whole dataset during testing process.

4.2. Deep Learning

Our deep learning approach have classified Android based on permissions and API calls. We stopped the model training at the 25th Epoch, since the loss rate in training and testing have become 0.07%, and the level of accuracy have reached 98.2%. Also, deep learning classifier was able to correctly predict 98.0% of the dataset. Furthermore, it has achieved high score of recall and precision, in which it was able to correctly detect 99.2% of the malicious samples. The results show that the model performance was enhanced using deep learning approaches, where the results ofdeep learning classifier was better than the results of the traditional machine learning classifiers. But, both experiments have achieved excellent result, thus the models that are based on permissions and API-calls considered to give promising results in term of Android malicious detection.

V. CONCLUSION

In this project, we presented DL-Droid, an automated dynamic analysis framework for Android malware detection. DL-Droid employs deep learning with a state-based input generation approach as the default method, although it has the capability to employ the state-of-the-practice popular Monkey tool (stateless method). We evaluated DL-Droid using Android applications, static and dynamic features, comparing its performance to traditional machine learning classifiers as well as existing DL-based frameworks.

- The presented results clearly demonstrate that DL-Droid achieved high accuracy performance reaching better figures than those presented in existing deep learning-based Android malicious detection frameworks.
- To the best of our knowledge, this is the first work to investigate deep learning using dynamic features extracted from apps using real phones.
- Our results also highlight the significance of enhancing input generation for dynamic analysis systems that are designed to detect Android malicious using machine learning.
- As future work, self-adaptation such as introduced and investigated recently for Intrusion Detection systems (Papamartzivanos et al., 2019) could be explored as a means of improving the performance of the deep learning based system for Android malware detection.

REFERENCES

[1] Gartner, Gartner says Android has surpassed a billion shipments of devices, http://www.gartner.com/newsroom/id/2954317, 2015.
[2] T. Vida's, D. Votipka, and N. Christin, all your droid is belonging to us:A survey of current Android attacks, in Proceedings of the 5th USENIX Workshop on Offensive Technologies (WOOT), 2011, pp. 81–90.
[3] A. P. Felt, M. Finifter, E. Chin, S. Hanna, and D. Wagner, A survey of mobile malware in the wild, in Proceedings of the 1st ACM Workshop on Securityand Privacy in Smartphones and Mobile Devices (SPSM), 2011, pp. 3–14.
[4] McAfee, McAfee labs threats report, http://www. mcafee.com/us/resources/reports/rp-quarterly-threat-q4-2013.pdf, 2015.
[5] A. Mylonas, A. Kastania, and D. Gritzalis, Delegate the smartphone user?Security awareness in smartphone platforms, Computers & Security, vol. 34,pp. 47–66, 2013.
[6] Z. Fang, W. Han, and Y. Li, Permission based Android security: Issues and countermeasures, Computers & Security, vol. 43, pp. 205–218, 2014.
[7] R. Pandita, X. Xiao, W. Yang, W. Enck, and T. Xie, Whyper: Towards automating risk assessment of mobile applications, in Proceedings of the 22nd USENIX Security Symposium (USENIX Security), 2013, pp. 527–542.

INNO SPACE
SJIF Scientific Journal Impact Factor

Impact Factor: 8.165

doi cross ref

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

9940 572 462 6381 907 438 ijircce@gmail.com

www.ijircce.com

Scan to save the contact details