Int. j. inf. tecnol. (March 2017) 9(1):111–117
DOI 10.1007/s41870-017-0010-2

ORIGINAL RESEARCH

Android malware detection: state of the art

Sunil Kumar Muttoo[1] **· Shikha Badhani**[2]

Published online: 22 February 2017

Abstract Android malware is on the rise along with the popularity of Android OS. Malware writers are using novel techniques to create malicious Android applications which severely undermine the capability of traditional malware detectors which are incompetent towards detecting these unknown malicious applications. The features obtained from static and dynamic analysis of Android applications can be used to detect unknown Android malware by using machine learning techniques. This paper presents an analysis of various Android malware detection systems and compares them based on various parameters such as detection technique, analysis method, and features extracted. We were able to find research work in all the Android malware detection techniques which employ machine learning which also highlights the fact that machine learning algorithms are used frequently in this area for detecting Android malware in the wild.

Keywords Android malware · Anomaly detection · Dynamic analysis · Machine learning · Misuse detection · Static analysis

✉ Sunil Kumar Muttoo
skmuttoo@cs.du.ac.in

Shikha Badhani
sbadhani@maitreyi.du.ac.in

[1] Department of Computer Science, University of Delhi, Delhi, India

[2] Department of Computer Science, Maitreyi College, University of Delhi, Delhi, India

1 Introduction

The dominant market position of Android OS has also attracted the interest of malware authors exploring the vulnerabilities of Android OS. As per G DATA mobile malware report [1], their experts have discovered around 4900 fresh Android malware every day in the first quarter of 2015. Malware have been categorized based on the functionalities of their malicious payload as license escalation, remote control by communicating with C & C server, financial charge and collecting information [2]. The scale of damage caused by an Android malware depends on its functionality and sometimes the results of its activity are invisible for the users which are a cause of concern. AVG mobile security team [3] recently discovered an Android malware, PowerOffHijack, which hijacks the shutdown process and the device remains functional giving it the freedom to move around on the device and steal data even though it appears to be off. In 2014, Doctor Web [4] had issued a warning to Android users about a new Trojan (BankBot.21.origin) that can steal information about the credit cards they use for transactions on Google Play. With such sophistication and experimentation, Android malware has become a critical threat.

This has led to an increase in the research work in the area of Android malware detection where researchers have introduced novel techniques and improvised the current ones focusing on Android malware detection.

In this paper, we have presented various detection techniques which give a better understanding of these techniques and explore the fact that detecting known/unknown malware using machine learning techniques is on the rise. An updated image of the past and latest Android malware detection techniques is discussed in this paper, and illustrated with examples.

This paper is structured as follows. In Sect. 2, we discuss Android Architecture and Sect. 3 presents the components of an Android application. Section 4 provides an overview of the security features present in Android. Section 5 presents various Android application analysis methods and Sect. 6 presents the Android malware detection techniques. Section 7 presents various Android malware detection systems that have been developed. Emerging directions are discussed in Sect. 8 and finally, we conclude in Sect. 9.

2 Android architecture

Android operating system is a software stack that contains the following six layers [5]: Linux Kernel, Hardware Abstraction Layer (HAL), Android Runtime, Native Libraries, Application Framework, and Applications (Fig. 1). At the base of the hierarchy is the Linux kernel which presents a level of abstraction between the device hardware and the upper layers of the software stack. It manages all the core functionalities of Android such as process management, memory management, security and networking. It also contains various device drivers related to camera, display, Wi-Fi, Bluetooth, audio, USB etc. On top of Linux kernel is the Hardware Abstraction Layer (HAL) which permits the Android application framework to correspond with the hardware specific device drivers by defining functional interfaces for each hardware component such as the camera module, Bluetooth module etc. The next layer is the Android Runtime which comprise a set of core Java libraries that provides most of the functionality for creating Android applications and the ART. Being implemented in Java, Android is executed by a Java Virtual Machine (JVM). Initially, Dalvik Virtual Machine (DVM) was used which is Android's implementation of JVM optimized for mobile devices. DVM executes files in Dalvik Executable (.dex) plan which is optimized for nominal memory footprint. For Android devices running Android version 5.0 or higher, a new runtime, DVM is replaced by its successor- the Android Runtime (ART) [6]. Advantages of ART over DVM are ahead-of-time (AOT) compilation, improved garbage collection and debugging support. AOT means applications are compiled into native machine code upon their installation and stored in device internal memory. Thus, any subsequent execution of the application would be faster since the translation has already taken place during installation. Parallel to the Android Runtime is the layer for native C/C++ libraries that are utilised by the operating system as well as various components of Android. Some of the major native libraries included are—C/C++ core libraries libc and SSL, OpenGL ES libraries for rendering 2D and 3D graphics, SQLite for native databases support, surface manager for display management, media library for playback of audio and video and so on. Android also provides Android Native Development Kit (NDK) to access these native libraries for implementing parts of an application. On top of the native libraries and Android Runtime lies the Java API Framework that provides the building blocks in the form of APIs written in Java language to create Android applications such as call management, location management, resource management, activity manager, user interface management via View system, data sharing via content providers, etc. The topmost layer is the System Application layer. This layer provides both—set of core built-in applications installed with every device (SMS messaging, dialer, web browser, contacts, calendar, and more) and an interface to any third-party application to access the key services by using the APIs provided by the API framework.

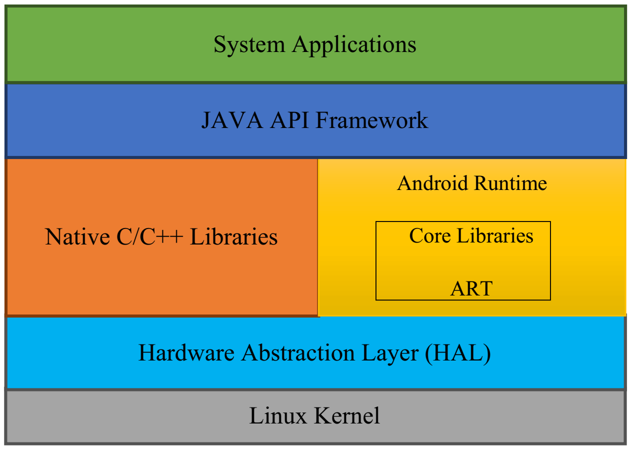

Fig. 1 Android architecture

3 Android application components

Android Studio [7]—the official Android SDK, provides an extensive set of application programming interface to developers for developing Android applications. Android applications are written in Java programming language and can also be written in native code. They are compiled and packaged in an APK (Android package) which is an archive file. Each application runs in a separate process and is composed of a mandatory XML descriptor file called AndroidManifest.xml. The Android Manifest file contains all the details required by the android system about the application. It allows defining the packages, APIs, libraries needed by an application, permissions enforced and requested by the application, descriptors of application components etc. Application components can be of four types: Activity, Service, Content Provider, and Broadcast

Receiver [8]. These components communicate through messages called Intents.

- An Activity represents a single screen with a user interface.
- A Service runs in the background to perform long-running tasks.
- A Content Provider is used to share and store structured and persistent data using a database interface.
- A Broadcast Receiver is responsible to receive and react to system-wide announcements.

Consider the caller application of an Android device. This application may include various components such as activities for viewing contacts and calling them. There may be a service for continuing a call in the background in case you want to use another application in the middle of a call for writing text. A broadcast receiver might be there for incoming calls. Lastly, content providers for sharing contacts on the device.

4 Android built-in security mechanisms

The Android platform being open-source makes it easily vulnerable to security attacks. Hence, security of Android applications becomes a major concern and a challenge. For that Android provides various built-in security mechanisms both at Linux kernel level and application framework level to develop secure applications. The Android security model consists of the following security features in order to protect user data, system resources and provide application isolation [9]:

- *Application sandbox*—Each application runs in its own process and address space. After installation, each application is assigned a unique Linux user ID which remains constant. Thus, applications run in sandboxes having limited access to system resources.
- *Secure IPC mechanism*—An application can interact with other applications as well as remote servers through secure Interprocess Communication (IPC) mechanisms.
- *Signatures*—Before installation, each application developed must be digitally signed with a certificate which identifies the developer of the application. This makes sure that that the future updates for an application are coming from the same developer and also establishes trust between applications. Applications having the same signature can share the same user ID and thus, become visible to each other and can also run in the same process.
- *Permissions*—If any application needs to access components of another application, it must have the required permissions to do so. Android's manifest file declares the permissions that an application requires and these permissions must be granted during installation. Conversely, applications can also declare their own permissions for other applications to use in the manifest file.

5 Android application analysis

Before proceeding towards Android malware detection techniques it is important to introduce how the Android applications are analyzed. There are three methods—Static, Dynamic and Hybrid. In the static method, the source code of the Android application is analyzed and this technique does not require to execute the application. A major problem with static analyzers is their incapability to detect the malicious behavior of an application at runtime. On the other hand, dynamic analysis involves extracting information from the traces obtained by executing an Android application in a sand-boxed environment but it involves the overhead of executing the applications and moreover it is not known when an application will show malicious behavior which makes dynamic analysis a slow process. The hybrid method combines the use of static and dynamic methods for analysis thus utilizing the pros of both methods for a better detection methodology.

6 Android malware detection techniques

A malware detector identifies and contains malware before it can reach a system or network [10]. This section introduces the commonly used Android malware detection techniques and their evaluation criteria. Before proceeding further, let us briefly discuss what Machine learning is and how it is categorized.

Machine learning [11] is the study of computer programs capable of learning their own previous experience to improve their performance of a task. Machine learning has been used widely in the field of Android malware detection to either learn malicious characteristics to detect malware based on the similarity (misuse detection) with them or learn from the benign characteristics to detect malware based on the deviation from them (anomaly detection) [12]. Machine learning algorithms can be segregated into two categories (Fig. 2)—supervised and unsupervised learning. Supervised learning is inferring a function from labeled data for classifying data for responses that have few known values or for performing regression for responses that are continuous and real-valued. Unsupervised learning is exploring the data to find some intrinsic structures in them such as clustering data to find similarity group called clusters.

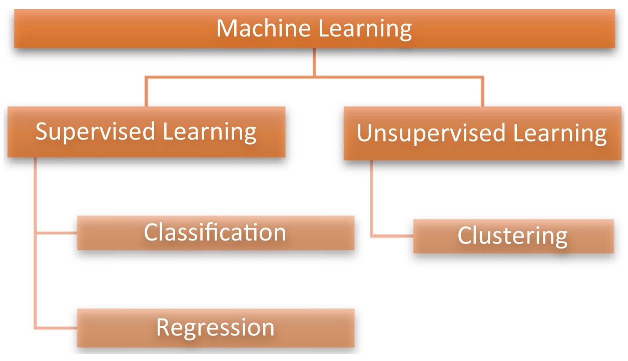

Fig. 2 Machine learning

The interested reader is referred to the overview of supervised [13] and unsupervised machine learning algorithms [14] for further study.

The Android malware detection techniques can be broadly classified into two categories [15, 16]—Misuse Detection (or Knowledge-based or Signature-based) and Anomaly Detection (or Behavior-based).

In this paper, we'll use the terms—misuse detection and anomaly detection for the above two categories which are discussed below.

6.1 Misuse detection

This technique detects malware by comparing them with the knowledge accumulated from known malware in the form of their signatures. Malware is detected based on the similarity with the known malware. It can accurately detect known malware but is quite ineffective against unknown attacks as no signatures are available. Also, variants of the previous malwares that deviates slightly are difficult to be detected by this technique. To overcome this restriction, machine learning has been used in misuse detection to automatically generate signatures for detecting unknown malwares [17]. Misuse detection requires continuous updation of the signature database with the signatures of the known malware. Another problem is that with such large increase in the number of malwares which have been detected [1], the size of the signature databases has also increased which makes it not feasible to be stored on the mobile phones being constrained by limited storage [15].

6.2 Anomaly detection

It is considered complementary to misuse detection. It focuses on the normal behavior of an Android application rather than detecting the presence of malware signatures. Malware is detected based on the deviation from the normal behavior. Machine learning has also been used in anomaly detection which is characterized by two phases—training phase and a detection phase. In training phase, the normal behavior of Android applications is learned using machine learning and then during detection phase, the learned normal behavior is compared with the current behavior of an Android application being analyzed for classification as malicious or benign or in other words malware detection. Since this technique does not rely on the characteristics of known malware, it can detect novel ones since similar behavior patterns are shared by novel and existing malware because novel malwares are often created by adding new malicious behaviors to the existing malware [18]. However, any previously unobserved normal behavior can also be flagged as malicious by this technique resulting in false alarms. To reduce the false alarms, specification-based detection is performed in which specifications of Android applications are manually developed and then detection is performed based on the deviations from normal specifications of an Android application. The disadvantage of this technique is that developing the specifications of Android applications is a very time-consuming process [15].

For the evaluation of the efficiency of these techniques following parameters are highlighted in [16, 19] in the context of intrusion-detection systems:

- *Accuracy*—It's a measure of correctness. Inaccuracy occurs when a legitimate application is flagged as malicious i.e., false positive.
- *Performance*—It's the rate at which the applications are analyzed for malware detection by a malware detector. Low rate implies compromising real-time detection.
- *Completeness*—It's the capability of a detector to detect all the malicious applications. Incompleteness occurs when a malicious application is flagged as legitimate i.e., false negative.
- *Fault Tolerance*—The malware detector itself should be resilient to malicious attacks. Being vulnerable to such attacks would defeat the whole purpose of the malware detection.
- *Timeliness*—Timely detection of a malware will not only help in preventing the damage caused but also curb its propagation. ESET, an IT security firm [20] recently detected a malware Android/Spy.Feabme.A which steals facebook credentials. This malicious application was installed by over 500,000 Android users from Google play store and following its detection it was taken down. This fact highlights the timely detection of malware and prevention of its propagation.

7 Comparative analysis and discussion

There has been a lot of quality research work done in the area of Android malware detection where researchers have used different techniques and analyzed them. In this section, we have listed various Android malware detection systems and compared them based on various parameters such as analysis method (Fig. 3) and detection technique used (Fig. 4).

Android Application Sandbox (AA Sandbox) [21] detects malicious applications by using both static and dynamic analyses on Android applications. In static analysis, it scans for usage of Java Native Interface, reflection, services and IPC provision, permissions and creating native children processes etc. In dynamic analysis, it summarizes the system log file to create a system histogram which contains how many times a system call is used during application runtime.

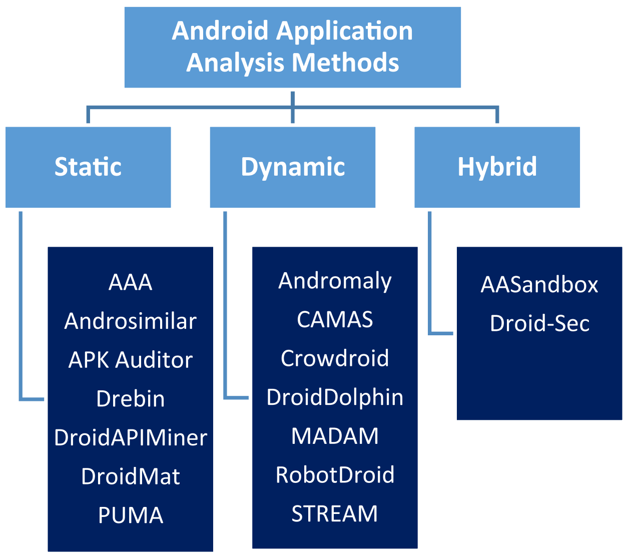

Fig. 3 Classification of android malware detection systems based on analysis methods

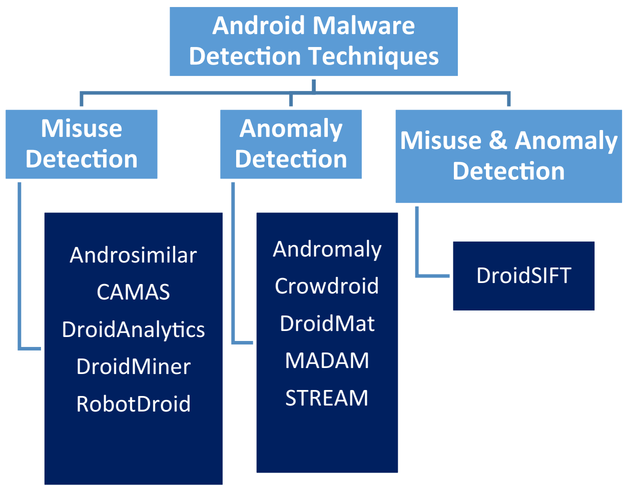

Fig. 4 Classification of android malware detection systems based on detection techniques

Android Application Analyzer (AAA) [22] uses permissions requested by Android applications during installation as features. Both, unsupervised (k-Means clustering) and supervised (Naïve Bayes classifier) machine learning algorithms are used. They successfully detected the following known malwares—DogWar (Trojan), ICalender (Premium SMS) and SuperSolo (DroidDream). They created two malware samples and were able to detect them as malicious.

Andromaly [23] monitors Android devices and extracts features such as CPU consumption, number of packets sent, number of running processes, keyboard/touch-screen pressing, battery level etc. It applies three feature selection methods (Chi Square, Fisher Score and Information Gain) to remove redundant or irrelevant features and then evaluates various classifiers (k-Means, Logistic Regression, Histograms, Decision Tree, Bayesian Networks and Naïve Bayes).

Androsimilar [24] automatically generates statistically robust variable length signatures for applications using SDHash approach and detects malware based on a similarity score. They claim that it is effective against code obfuscation and repackaging.

APK Auditor [25] uses permissions, services and receivers of Android applications as features and uses a statistical scoring approach to detect malicious Android applications.

Classifying Android Malware Through Subgraph Mining (CAMAS) [26]—extracted a subset of frequent subgraphs of system calls executed by Android application and then applied machine learning techniques(Linear discriminant classifier (LDC), Quadratic discriminant classifier (QDC), k-NN, and Artificial neural networks (ANN)) for Android malware detection.

Crowdroid [27] obtained real traces of applications' behavior based on crowd sourcing and clustered them using k-means algorithm on a server for detection. It achieved 100% detection of self-written applications and also tested the system against few known malwares—Steamy Window (PJApps) and Monkey Jump 2 (HongTouTou).

Drebin [28] extract features such as hardware components, requested and used permissions, activities, Services, Content providers, Broadcast receivers, Filtered intents, Suspicious and restricted API calls, Network addresses etc. It embeds them in vector space and then performs classification using linear Support Vector machines (SVM). On five popular smartphones, Drebin required 10 s for an analysis on average.

DroidAPIMiner [29] uses the requested permissions, API calls, package and parameter level information as features and evaluated them by using three different classifiers (Linear SVM, kNN, Decision Trees -C4.5 and ID3).

DroidDolphin [30] uses dynamic features such as incoming/outgoing network data, file read and write operations, started services, loaded classes through DexClassLoader, information leaks via the network, files and SMS, logcat etc. SVM is then used to classify applications as benign or malicious by using these features.

DroidMat [31] ses permissions, intents and API calls as features. Initially it performs unsupervised machine learning on these features by using algorithms such as k-Means or EM clustering. Then, the clustered features are used to perform classification by using kNN or Naïve Bayes classifier.

DroidMiner [32] used a behavioral graph to abstract malicious behavior patterns (activities, services, broadcast receivers, content observers, permission-related API functions etc.) into a sequence of threat modalities, and then applied machine-learning techniques to identify Android malware.

Droid-Sec [33] uses a hybrid approach to extract both static (permissions and sensitive APIs) and dynamic (behaviors extracted from logs) features from Android applications. It then uses deep learning to perform classification.

DroidSIFT [34] classifies Android malware via API dependency graphs and uses graph similarity metrics to detect unknown malware by using Naïve Bayes classifier. It contains both, a signature detection system and anomaly detection.

Multi-level Anomaly Detector for Android Malware (MADAM) [35] globally monitors Android at the kernel-level and user-level extracting features such as system calls, running processes, free RAM, CPU usage, user activity/idleness, key-stroke, called numbers, sent/received SMS, and Bluetooth/Wi-Fi analysis to detect real malware to distinguish between standard behaviors and malicious ones. It uses various machine learning algorithms for performing the classification.

Permission Usage to detect Malware in Android (PUMA) [36] extract permissions from Android applications and evaluated by using different machine learning classifiers (Simple Logistic, Naïve Bayes, Bayes Net, SMO, IBK, J48, Random Tree, and Random Forest) using k-fold cross validation.

RobotDroid [37] extracts dynamic features such as Intent issued and system resources access by applications in Android-based smartphone operating systems. Then, it performs SVM active learning to detect malware.

System for Automatically Training and Evaluating Android Malware Classifiers (STREAM) [38] demonstrates the effectiveness of different machine learning classifiers (Random Forest, Naive Bayes, Multilayer Perceptron, Bayes net, Logistic and J48) in detecting Android malware. It focuses on profiling applications to obtain information (battery, binder, memory, network, and permissions) used in dynamic analysis.

From the above analysis, it is evident that the use of machine learning algorithms is dominant. It is being used irrespective of the analysis method used—static, dynamic or hybrid and detection technique applied—Misuse, Anomaly or Misuse and Anomaly. Another important fact is that a variety of machine learning algorithms are used in Android malware detection. Even the new areas of machine learning research such as deep learning are used in Android malware detection [33]. Thus, each time a new machine learning algorithm is introduced, it gives a new direction to researchers to implement the same in the area of Android malware detection which would not only help them to analyze the efficiency of the algorithm but also help in improving the Android malware detection efficiency.

8 Emerging directions

We have identified a number of directions in which Android malware detection is progressing. Android malware is growing and this problem needs to be addressed by creating more effective Android malware detection systems to not only improve the accuracy of detecting known malwares but also uncover zero-day malware attacks. Another desirable aspect is timely detection of malware to prevent it to cause further harm by spreading itself. Hence, we propose to create a new Android malware detection system by performing static analysis and extracting features such as permissions and APIs and then classifying them by using Extreme Learning Machine(ELM) [39]. ELM has been used for Android malware detection in [40] which extracts static features from Dalvik instructions. In our proposed work, not only different set of features are used but we also perform an empirical evaluation and compare the performance of ELM with other machine learning algorithms.

9 Conclusion

A lot of work has been done in the area of Android malware detection. Researchers are also using the combination of misuse and anomaly detection for improving the detection accuracy. This paper lists the features extracted for the purpose malware detection as well as various machine learning algorithms being used for the same. The presence of machine learning algorithms in all categories of malware detection techniques and analysis methods highlights the fact that machine learning algorithms are being used frequently in this area for detecting Android malware in the wild.

References

1. G Data Mobile Malware Report (2015) https://public.gdatasoftware.com/Presse/Publikationen/Malware_Reports/G_DATA_MobileMWR_Q1_2015_US.pdf
2. Zhou Y, Jiang X (2012) Dissecting Android malware: characterization and evolution. Proc IEEE Symp Secur Priv 4:95–109
3. "AVG." http://now.avg.com/malware-is-still-spying-on-you-after-your-mobile-is-off/
4. "Dr. Web." https://news.drweb.com/show/?i=5860&lng=en
5. "Platform Architecture." https://developer.android.com/guide/platform/index.html
6. "ART and Dalvik." https://source.android.com/devices/tech/dalvik/
7. "Android Studio." https://developer.android.com/studio/index.html
8. "Application Fundamentals." https://developer.android.com/guide/components/fundamentals.html
9. Enck W, Ongtang M, McDaniel P (2009) Understanding Android Security. IEEE Secur Priv 7:50–57
10. Christodorescu M, Jha S (2004) Testing malware detectors. ACM SIGSOFT Softw Eng Notes 29:34
11. Mitchell TM (1997) Machine learning. McGraw-Hill, Inc., New York
12. Sammut C, Webb GI (2011) Encyclopedia of machine learning. Springer Sci. Bus. Media, vol. 33, pp. 439–447
13. Kotsiantis SB (2007) Supervised machine learning: a review of classification techniques. Mach Learn 31:249–268
14. Ghahramani Z (2004) Unsupervised learning. Mach. Learn. 2003 (Summer Sch., pp 72–112
15. Ning P (2003) Intrusion detection techniques. Internet Encycl
16. Debar H, Dacier M, Wespi A (1999) Towards a taxonomy of intrusion-detection systems. Comput Netw 31(8):805–822
17. Hassan D, Might M (2014) A similarity-based machine learning approach for detecting adversarial android malware
18. Christodorescu M, Jha S, Seshia SA, Song D, Bryant RE (2005) Semantics-aware malware detection. In: Proceedings of IEEE symposium security privacy, pp 32–46
19. Porras PA, Valdes A (1998) Live traffic analysis of TCP/IP Gateways. In: Proceedings of 1998 ISOC symposium network distributed system security NDSS98
20. "ESET."
21. Bläsing T, Batyuk L, Schmidt AD, Camtepe SA, Albayrak S (2010) An android application sandbox system for suspicious software detection. In: Proceedings of 5th IEEE international conference malicious unwanted software, Malware 2010, pp 55–62
22. Almin SB, Chatterjee M (2015) A novel approach to detect android malware. Procedia Comput Sci 45:407–417
23. Shabtai A, Kanonov U, Elovici Y, Glezer C, Weiss Y (2012) 'Andromaly': a behavioral malware detection framework for android devices. J Intell Inf Syst 38(1):161–190
24. Faruki P, Laxmi V, Bharmal A, Gaur MS, Ganmoor V (2015) AndroSimilar: robust signature for detecting variants of Android malware. J Inf Secur Appl 22:66–80
25. Talha KA, Alper DI, Aydin C (2015) APK auditor: permission-based android malware detection system. Digit Investig 13:1–14
26. Martinelli F, Saracino A, Sgandurra D (2014) Classifying android malware through subgraph mining. Lect Notes Comput Sci 8247:268–283
27. Burguera I, Zurutuza U, Nadjm-Tehrani S (2011) Crowdroid: behavior-based malware detection system for android. In: Proceedings of 1st ACM work security privacy smartphones mobile devices—SPSM'11, p. 15
28. Arp D, Spreitzenbarth M, Malte H, Gascon H, Rieck K (2014) Drebin: effective and explainable detection of android malware in your pocket. In: Symposium network distributed system security, pp 23–26
29. Aafer Y, Du W, Yin H (2013) DroidAPIMiner: mining API-level features for robust malware detection in android. Secur Priv Commun Netw 127:86–103
30. Wu W (2014) DroidDolphin: a dynamic android malware detection framework using big data and machine learning
31. Wu DJ, Mao CH, Wei TE, Lee HM, Wu KP (2012) DroidMat: android malware detection through manifest and API calls tracing. In: Proceedings of 2012 7th Asia joint conference information security AsiaJCIS 2012, pp 62–69
32. Yang C, Xu Z, Gu G, Yegneswaran V, Porras P (2014) Droid-Miner: automated mining and characterization of fine-grained malicious behaviors in android applications. Lect Notes Comput Sci 8712:163–182
33. Yuan Z, Lu Y, Wang Z, Xue Y (2014) Droid-Sec: deep learning in android malware detection. Sigcomm 2014:371–372
34. Zhang M, Duan Y, Yin H, Zhao Z (2014) Semantics-aware android malware classification using weighted contextual API dependency graphs. In: Proceedings of 2014 ACM SIGSAC conference computer communication security, pp 1105–1116
35. Dini G, Martinelli F, Saracino A, Sgandurra D (2012) MADAM: a multi-level anomaly detector for android malware. Lect Notes Comput Sci 7531:240–253
36. Sanz B, Santos I, Laorden C, Ugarte-Pedrero X, Bringas PG, Álvarez G (2013) PUMA: permission usage to detect malware in android. Advances intelligent systems computing, vol. 189 AISC, pp 289–298
37. Zhao M, Zhang T, Ge F, Yuan Z (2012) Robotdroid: a lightweight malware detection framework on smartphones. J Netw 7(4):715–722
38. Amos B, Turner H, White J (2013) Applying machine learning classifiers to dynamic android malware detection at scale. In: 9th International wireless communication mobile computer conference IWCMC 2013, pp. 1666–1671
39. Huang G-B, Zhu Q-Y, Siew C-K (2006) Extreme learning machine: theory and applications. Neurocomputing 70:489–501
40. Zhang W, Ren H, Jiang Q, Zhang K (2015) Exploring feature extraction and ELM in malware detection for android devices. Lect Notes Comput Sci 9377:489–498