IEICE TRANS. INF. & SYST., VOL.E103–D, NO.2 FEBRUARY 2020

INVITED PAPER *Special Section on Security, Privacy, Anonymity and Trust in Cyberspace Computing and Communications*

A Survey on Mobile Malware Detection Techniques

Vasileios KOULIARIDIS[†a)], **Konstantia BARMPATSALOU**[††b)], **Georgios KAMBOURAKIS**[†††c)], *and* **Shuhong CHEN**[††††d)], ***Nonmembers***

SUMMARY Modern mobile devices are equipped with a variety of tools and services, and handle increasing amounts of sensitive information. In the same trend, the number of vulnerabilities exploiting mobile devices are also augmented on a daily basis and, undoubtedly, popular mobile platforms, such as Android and iOS, represent an alluring target for malware writers. While researchers strive to find alternative detection approaches to fight against mobile malware, recent reports exhibit an alarming increase in mobile malware exploiting victims to create revenues, climbing towards a billion-dollar industry. Current approaches to mobile malware analysis and detection cannot always keep up with future malware sophistication [2], [4]. The aim of this work is to provide a structured and comprehensive overview of the latest research on mobile malware detection techniques and pinpoint their benefits and limitations.

key words: *smartphones, mobile malware, malware detection*

1. Introduction

Continuous improvement in mobile device hardware and mobile communication technologies has led to a highly interconnected world, but also a world grown highly vulnerable. Cybercriminals have proven significantly efficient in uncovering new vulnerabilities in popular mobile Operating Systems (OS) and the installed applications (apps). As a result, more and more mobile malware families are introduced every year [1]. Just in 2017, more than 20M malware samples have been detected [2], while other studies show that the chance for monetization is a key factor responsible for the rise of mobile malware [3].

The increasing amount of malware introduced each year is only a facet of this evolution. The variety of these malware and the vulnerabilities exposed call for new and improved detection methods. Furthermore, while researchers strive to find alternative detection schemes to counter mobile malware, recent reports show an alarming increase in mobile malware exploiting victims to create revenues [2].

Until now, mobile malware detection techniques have been surveyed by several works. Yan et al. [4] compare mobile malware detection methods based on several different evaluation criteria and metrics, but mainly focus on the Android OS. La Polla et al. [5] survey the evolution of mobile threats, vulnerabilities and intrusion detection systems over the period 2004–2011. While this is one of the most comprehensive works on the topic, by now it misses current developments. Gandotra et al. [6] examine techniques for analyzing and classifying mobile malware. Furthermore, they list several works for each detection technique. Nevertheless, they do not discuss the effectiveness of each work based on their evaluation results. Yan et al. [7] report on mobile malware categories, taxonomy and attack vectors. Furthermore, they provide a comparison of dynamic mobile malware detection methods and discuss future research trends.

This survey aims to provide state-of-the-art information on current mobile malware trends. Furthermore, it offers a comprehensive overview of the different approaches to mobile malware detection, in an effort to understand their detection method, discuss their evaluation results, and possibly categorize each contribution under a novel classification scheme.

The remainder of this paper is organized in the following manner: Section 2 focuses on current mobile malware and their effects on the end-users. Section 3 presents the different mobile malware detection techniques. The same section categorizes various mobile malware detection approaches based on the corresponding detection techniques. Section 4 provides a discussion on the findings. Finally, Sect. 5 concludes the paper.

2. Mobile Malware Analysis

While there are many conventional types of mobile malware, including Trojans, worms [8], botnets [9], spyware and ransomware, the latest ilks seem to be driven by a common factor, monetization [2], [3], [10]. The latest mobile malware trends can be summarized as follows:

- Mobile Banking Trojans: Trojans steal user's confidential information without the user's knowledge. They can usurp browsing history, messages, contacts and even banking credentials. According to the McAfee

Manuscript received March 10, 2019.
Manuscript revised August 8, 2019.
Manuscript publicized November 27, 2019.

[†]The author is with the Department of Information & Communication Systems Engineering, University of Aegean, Greece.
[††]The author is with the Centre for Informatics and Systems of the University of Coimbra, Coimbra, Portugal.
[†††]The author is with the European Commission, Joint Research Centre (JRC), Ispra, Italy.
[††††]The author is with the Guangzhou University, China.
a) E-mail: bkouliaridis@aegean.gr (Corresponding author)
b) E-mail: konstantia@dei.uc.pt
c) E-mail: georgios.kampourakis@ec.europa.eu
d) E-mail: shuhongchen@gzhu.edu.cn
DOI: 10.1587/transinf.2019INI0003

Mobile Threat Report [2], mobile banking Trojans, such as BankBot [11], increased by 60% in 2018. End-user devices get infected by fake updates, email and SMS phishing.

- Cryptocurrency Mining: While not as sophisticated as their desktop counterparts, mobile malware related to Bitcoin mining has increased by 80% in 2018 [2]. According to Kaspersky Security Network [12], most malware of this type is hidden within popular apps, that were secretly mining cryptocurrency while showing soccer videos.
- Ransomware: This type of mobile malware prevents users from accessing the data on their devices by encrypting them, until a considerable ransom amount is paid. In the first half of 2017 mobile ransomware has increased by 60% [2]. While this growth was triggered by the "Ransom.AndroidOS.Congur" malware family [13], many other ransomware families still present an alarming threat to users who must choose to either pay the ransom or end up with possibly valuable encrypted data.
- Hybrid: This type of mobile malware is very common nowadays. For example, Android/LokiBot [2] combines the functionality of a banking trojan with crypto ransomware. It can encrypt files but it might also send fake notifications in an attempt to trick users into logging in to their bank account. Android/LokiBot has targeted more than 100 financial institutions and kit sales on the dark web generating a profit of up to 2$ million [2].

3. Mobile Malware Detection

Mobile malware detection methods serve as countermeasures for the existing malware. However, their functionality differs according to variables related to the focus of each method. In this section, we classify the existing research works, according to the detection techniques reported by the authors, and we review their functionality and effectiveness.

Malicious activity detection in mobile devices occurs in different patterns. Researchers have not yet agreed on a unified classification. One aspect claims that there exist two main types of malicious software analysis methods, namely *static* and *dynamic*. Other researchers however, use an inverse approach in malware detection classification, where static and dynamic detection serve as subcategories to signature and anomaly-based techniques [14]. Figure 1 depicts the malware detection classification used in this work.

For the survey part of this work, we have focused on research papers dated no more than 10 years ago. The considered works have been categorized in the following subsections in chronological order.

3.1 Signature-Based Detection

The main categorization vector in malware detection methods is related to the detection type. The two main detection types are signature-based and anomaly-based. Signature-based detection collects patterns and signatures from known malware and compares them against suspicious pieces of code in order to determine whether they are malicious or benign. Signature-based detection techniques are further classified to *Behavior* and *Static* signature-based subcategories. Static signature-based techniques are used by most of the commercial antivirus software solutions.

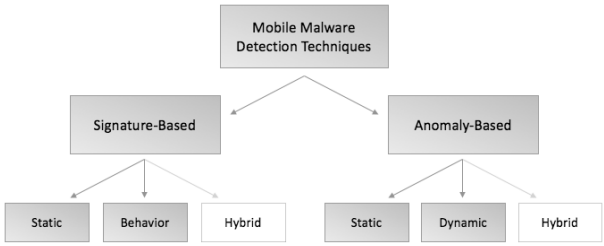

Fig. 1 Mobile malware detection classification

Static Signature-based Detection: This type uses a database containing entries of malware sample signatures and compares objects that reside either in the RAM or in the SD storage of the device for matching patterns. Enck et al. [15] proposed a security service for the Android Operating System (OS), named *Kirin*. *Kirin* certificates an app at install time, using a set of security rules, which are templates designed to match suspicious properties in apps' security configuration. More specifically, after the installer extracts security configuration from the package manifest, *Kirin* evaluates the configuration against a collection of predefined security rules.

Behavior Signature-based Detection: In static signature-based technique, the acquisition of signatures occurs during the decomposition and analysis of the malware source code. On the other hand, signatures in dynamic behavior-based techniques are acquired after the execution of the malicious code. More specifically, information is gathered during app execution to decide its maliciousness. This is done using preconfigured and predetermined attack patterns that are given beforehand by experts to build a signature database or a pattern set [4].

Chen et al. [16] proposed a detection approach which identifies threat patterns. It analyzes the function invocation, as well as the data flow to detect malicious behaviors in Android devices. More specifically, their scheme uses reverse engineering to recreate the source code and class files from each app and builds the corresponding API invocation and dependency graphs. Based on these two graphs, their system can detect threat patterns, which may reveal whether an app attempts to access confidential information or perform any illegal access. Their experiments show 91.6% detection rate over 252 malicious samples.

Hybrid Signature-based Detection: Hybrid signature-based detection includes both static and behavior signature-based detection. Papamartzivanos et al. [17] proposed a host and cloud-based system that operates under a crowdsourcing logic. Their system includes 3 main services, namely

privacy-flow tracking, crowdsourcing, and detection and reaction against privacy violations. The client communicates with the cloud services over a TLS connection so as to be relieved from resource demanding tasks. More specifically, the client consists of 3 modules, namely privacy inspection, response, and event sensor. The cloud side also comprises 3 modules, namely crowdsourcing, detection, and hook update.

3.2 Anomaly-Based Detection

Anomaly-based methods use a less strict approach. This is done by observing normal behavior of a device for a certain amount of time and using the metrics of that normal model as a comparison vector to deviant behavior. In regards to the analysis part, the static and dynamic methods are used. The static approach examines an app before installation by dissecting it, whereas the dynamic performs the analysis during the app execution, by gathering data such as system calls and events. Either in the dynamic or the static version, anomaly-based detection techniques comprise two parts, the training and detection phase. During the former, a non-infected system is operating normally and this procedure is observed and tracked. On the other hand, the detection phase serves as a testing period, when deviations from the training period model are considered anomalies.

Static Anomaly-based Detection: Static anomaly-based detection methods do not require the execution of the malicious payload. Their function is to check the code of the potentially malicious app for specific snippets of code, suspicious functionality, and other behavioral traits. It is not only capable of detecting unknown malware, but also of pointing out potential vulnerabilities in the source code. However, this method has its shortcomings as well. False positive ratios continue to be high and the task of code inspection can be costly in resources such as time and computational power.

Wu et al. [18] implemented *DroidMat*, which provides malware detection through manifest and API call tracing. The authors extract app information from its manifest file and disassembly codes. More specifically, they collect information from the app's manifest file such as "intent", which is an abstract description of an operation to be performed, and Inter-Component Communications (ICC) and API calls related to permissions. The authors collected 238 Android malware and 1,500 benign apps to test *DroidMat*, and their results show an up to 97.87% accuracy rate in detecting mobile malware.

An approach which analyses an app's permissions to detect malware in Android *(PUMA)*, was presented by Sanz et al. [19]. The authors gathered 1,811 benign Android apps, as well as 4,301 malware samples. The authors state that they observed several differences in permissions usage by malware apps. More specifically, they noticed that malware often requires only one permission, while benign apps usually ask for 2 or 3 permissions. The authors used several machine learning techniques for malware detection, including SimpleLogistic, NaiveBayes, BayesNet, SMO, IBK, J48, RandomTree, and RandomForest. Finally, they performed analysis on the extracted permissions from mobile apps and observed a detection accuracy of 92%.

Peiravian et al. [20] proposed the combination of permissions and API calls and the use of machine learning methods to detect malicious Android apps. Their framework consists of 4 components. The first one decompresses the APK file of an app to extract the manifest and class files. The second characterizes apps based on the requested permissions and API calls. The third one carries out feature extraction on the permissions and API calls. The latter employs the training of the classification models from the collected data. The authors state that during the evaluation tests, the proposed method achieved a promising detection rate, while holding precision up to 94.9%.

In an attempt to address the issue of removing malicious apps from mobile app markets, Chakradeo et al. [21] proposed an approach for market-scale mobile malware analysis *(MAST)*. *MAST* analyzes attributes extracted from the app package and uses Multiple Correspondence Analysis (MCA) to measure the correlation between multiple categorical data. Furthermore, only easily obtained attributes are extracted to keep *MAST* less costly than meticulous analysis. These attributes are permissions included in the manifest file, intent filters and pre-agreed upon action strings (also included in the manifest file), native libraries inside the source code and malicious payloads hidden in zip files inside the app package. During the training phase, 15,000 apps from Google Play [25] and a dataset of 732 known-malicious apps were used to train *MAST*. According to the authors, *MAST* triage processes mobile app markets in less than a quarter of the time required to perform signature detection.

Liang et al. [22] proposed a permission combination-based scheme for Android mobile malware detection. The authors collected permission combinations declared in the app manifest file, which are requested frequently by mobile malware, but rarely by benign apps. More specifically, a tool called *k-map* was developed in order to find permission combinations extracted from the app's manifest file. Moreover, they calculated the permission request frequencies out of the permission combinations extracted. Their experiments showed that the system was able to detect malware with low false positive and negative rates, that is, malware detection rate up to 96%, and the benign app recognition rate was up to 88% [22].

Canfora et al. [23] proposed mobile malware detection using op-code frequency histograms. Their approach classifies malware by focusing on the number of occurrences of a specific group of op-codes. More specifically, the authors used a detection technique, which uses a vector of features obtained from 8 Dalvik op-codes. These op-codes are usually used to alter the app's control flow. After training the classifier, the authors tested their proposed method to conclude that these features are able to classify a mobile app as trusted or malicious with a precision rate of 93.9%.

Yusof et al. [24] proposed a mobile botnet classification based on permissions and API calls. During the training phase, 5,560 malware from 179 different mobile malware families were collected. The authors examined 50 Android botnet samples using static analysis and reverse engineering to extract the 16 most important permissions and 31 API Calls from the botnet samples. Finally, they chose 800 random apps from Google Play [25] to test their classification using Naive Bayes, K-nearest Neighbour, Random Forest, and Support Vector Machine algorithms. Their results achieved 99.4% detection rate and 16.1% false positive rate.

Li et al. [26] proposed *SIGPID*, a malware detection system based on permission usage analysis on the Android platform. To test their detection model, the authors collected 3 different datasets which contain 2,650, 5,494 and 54,694 malware apps respectively. Their detection model uses 22 out of 135 permissions to improve the runtime performance by 85.6%. Finally, they used machine learning algorithms to evaluate their results, including RandomForest, PART, FT, RotationForest, RandomCommittee, and SVM, and achieved a detection rate of 93.62%.

Tao et al. [27] proposed *MalPat*, an automated malware detection system which scans for malicious patterns in Android apps. During the training phase, the authors were able to acquire hidden patterns from malware and extract APIs that are widely used in Android malware. The authors collected 31,185 benign apps and 15,336 malware samples and extracted features from the source code of decompiled files. To evaluate *MalPat*, the authors followed a repeated process, in which they randomly selected a percentage of both malicious and benign datasets as the training set, and the remaining part is regarded as the testing set. The average of their results show that *MalPat* can detect malware with a 98.24% F1 score.

Shen et al. [28] proposed a malware detection approach based on information flow analysis. The authors proposed complex-flow as a new representation schema for information flows. According to the authors, complex-flow is a set of simple flows that share a common portion of code. For example, if an app is able to read contacts, store them and then send them over the Internet, then these two data flows would be (contact, storage) and (contact, network). The authors state that their approach can detect if an information flow is malicious or not based on the app's behavior along the flow. When a new app is installed, their system compares its behavior patterns (obtained from the complex-flows representation of the app) to decide whether it is more similar to benign or malicious apps from the training set using two-class SVM classification. During the evaluation process, the authors used 4 different data sets, totaling 8,598 apps, to test the precision of their detection approach.

Dynamic Anomaly-based Detection: In dynamic anomaly-based detection, the training and detection phases happen during the execution of the app. Apart from the capability of detecting unknown malware, this trait also enables the detection of zero-day attacks. However, as already mentioned before, the false positive rate issues are rather intense, particularly in dynamic anomaly-based detection techniques. In order to soften this incident, accurate normal behavioral models have to be constructed during the training sessions.

Shabtai et al. [29] presented a system for detecting meaningful deviations in a mobile app's network behavior. The system monitors the running apps to create their "normal" network behavior. It is then able to detect deviations from the learned patterns. According to the authors, their main goal was "to learn user-specific network traffic patterns for each app and determine if meaningful changes occur". For this reason, semi-supervised machine learning methods were used to create the normal behavioral patterns and to detect deviations from the app's expected behavior.

Damopoulos et al. [30] proposed a tool which dynamically analyzes iOS apps in terms of method invocation. The authors designed and implemented an automated malware analyzer and detector for the iOS platform, namely *iDMA*. *iDMA* is able to generate exploitable results, which can be used to trace app's behavior to decide if it contains malicious code. Also, Damopoulos et al. [31] proposed an IDS framework that supports both host- and cloud-based protection mechanisms. Their framework employs diverse anomaly-based mechanisms. To evaluate their architecture, the authors developed a proof-of-concept implementation of the framework, equipped with 4 smartphone detection mechanisms. "The first two detection mechanisms, namely SMS Profiler and *iDMA*, aim to detect the illegitimate use of system services and identify unknown malware. The other two, coined iTL and Touchstroke, can provide (post) authentication to ensure the legitimacy of the current user" [31].

Jang et al. [32] presented Andro-AutoPsy, an anti-malware system based on similarity matching of malware information. During the training phase, the authors gathered malware-centric and malware creator-centric information from anti-virus technical reports, malware repositories, community sites and web crawling. The authors chose 5 footprints as features: "the serial number of a certificate, malicious API sequence, permission distribution (critical permission set, likelihood ratio), intent and the intersection of the usage of system commands and the existence of forged files" [32]. Andro-Autopsy consists of a client app running on the device and a remote server. The client app sends the app package file (.apk) to the remote server. The latter entity then analyzes the app and decides whether it is malicious or not, based on integrated footprints. The authors state that Andro-AutoPsy "successfully detected and classified malware samples into similar subgroups by exploiting the profiles extracted from integrated footprints" [32], while it is able to detect zero-day exploits at the same time. Furthermore, Andro-AutoPsy allows anti-virus vendors to conduct similarity matching on previously detected samples.

Chen et al. [33] aimed to combine network traffic analysis with machine learning methods to identify malicious network behavior in highly imbalanced traffic. The authors captured traffic from over 5,560 mobile malware samples. Furthermore, they designed a tool to convert mobile traffic

packets into traffic flows. According to the authors, the accuracy rate of the machine learning classifiers can reach up to 99.9%. However, the performance of the classifiers declines when the imbalanced problem gets worse.

Kouliaridis et al. [34] proposed *Mal-warehouse*, an open-source tool performing data collection-as-a-service for Android malware behavioral patterns. Specifically, the authors collected 14 malware samples to analyze their effects on the Android platform. The authors developed an open source tool called *"MIET"*, which extracts usage information, over a period of time, from the Android device for each malware installed on the device. Finally, *Mal-warehouse* is enhanced with a detection module, which the authors evaluated via the use of machine learning techniques.

Wang et al. [35] proposed a method which combines analysis of network traffic with the c4.5 machine learning algorithm which according to the authors is capable of identifying Android malware with high accuracy. During the evaluation process the authors tested their model with 8,312 benign apps and 5,560 malware samples. Furthermore, their results show that the proposed model performs better than state-of-the-art approaches. Finally, when combining two detection mechanisms, it achieves a detection rate of 97.89%.

Hybrid Anomaly-based Detection: Hybrid anomaly-based detection incorporates both static and dynamic anomaly-based detection. Hanlin et al. [36] presented a cloud-based Android malware analysis service called *ScanMe mobile*. The service allows users to scan app package files on their smartphone's SD memory card and perform dynamic analysis in a pre-configured sandbox environment prior installing them. The service also allows users to compile a comprehensive report, and share the report via a web interface. ScanMe mobile performs both static and dynamic analysis on app package files. The authors collected malicious and benign app samples to test the service on different Android devices. According to the authors, the system scored a detection rate of 85% when dynamic analysis was employed.

Alam et al. [37] proposed *DroidNative* for the detection of both bytecode and native code Android malware. According to the authors, *DroidNative* is the first scheme to build cross-platform (x86 and ARM) semantic-based signatures for Android and operates at the native code level. When apps are analyzed, bytecode components are passed to an Android Runtime (ART) [38] compiler to produce a native binary. The binary code is then disassembled and translated into Malware Analysis Intermediate Language (MAIL) code. After MAIL code is generated, *DroidNative* operates in two phases, training and testing. To evaluate *DroidNative*, the authors performed a series of tests with more than 5,490 Android apps. Their results demonstrated a detection rate of 93.57% with a false positive rate of 2.7%. Unfortunately, as with all static analysis detection techniques, *DroidNative* cannot detect compressed or encrypted malicious code.

Fei et al. [39] proposed a hybrid approach for mobile malware detection. The authors collected information pertaining to runtime system calls over a set of known malware and benign apps using a dynamic approach. More specifically, they gathered system-calling data during runtime by modifying the Android OS source code. Furthermore, they processed and analyzed the collected information to create malicious patterns and normal patterns from both system calls and sequential system calls. That is, malicious and normal patterns are produced "by calculating the ratio of the average frequency of a sequential system call in the set of malware and the average frequency of the same sequential system calls in the set of benign apps" [39]. According to the authors, the accuracy rate of their detection approach exceeds 90%.

4. Discussion

This section presents a comprehensive comparison of the 22 mobile malware detection approaches surveyed in Sect. 3. Figure 2 illustrates the timeline of the research works included in this survey. As already mentioned, the surveyed works are dated between 2009 and 2018. Different kinds of geometrical shapes refer to detection classification (e.g., square to static signature-based, trapezium to behavior signature-based, parallelogram to hybrid signature-based, circle to static anomaly based, diamond to dynamic anomaly-based, and hexagon to hybrid anomaly-based). The various works are placed within the diagram in chronological order (top to bottom). Numbers inside them correspond to the matching reference. The letter on the left refers to OS type (A is for Android, I is for iOS), while the letter on the right refers to the detection method. The selection of letters is as close to the first letter of each detection method as possible. Solid lines between two shapes imply influence (of a given work vis-a-vis to another), while dashed ones imply compliance or reference to previous work.

As shown in Fig. 2, Enck et al. [15] and Wu et al. [18] had an important impact on the evolution of mobile malware detection. Furthermore, while there is a variation in detection methods used during the previous 8 years, latest contributions lean towards anomaly-based detection. More specifically:

- At least 9 out of 22 approaches depend on the app's manifest file for their detection process, including [15], [18]–[20], [22], [24], [26], [27], [36]. Permission analysis is a popular detection technique among these approaches and it is the most popular detection technique since 2014. According to evaluation results from these contributions, permission-based detection can produce results with high detection rate, but also in some cases high false positive rate (FPR).
- Schemes which utilize native code analysis, such as Alam et al. [37], can produce a high detection rate of up to 93.57% and 2.7% FPR. Unfortunately, this approach cannot detect compressed or encrypted code.
- Complex-flow analysis is a new type of information flow analysis proposed by Shen et al. [28], which ac-

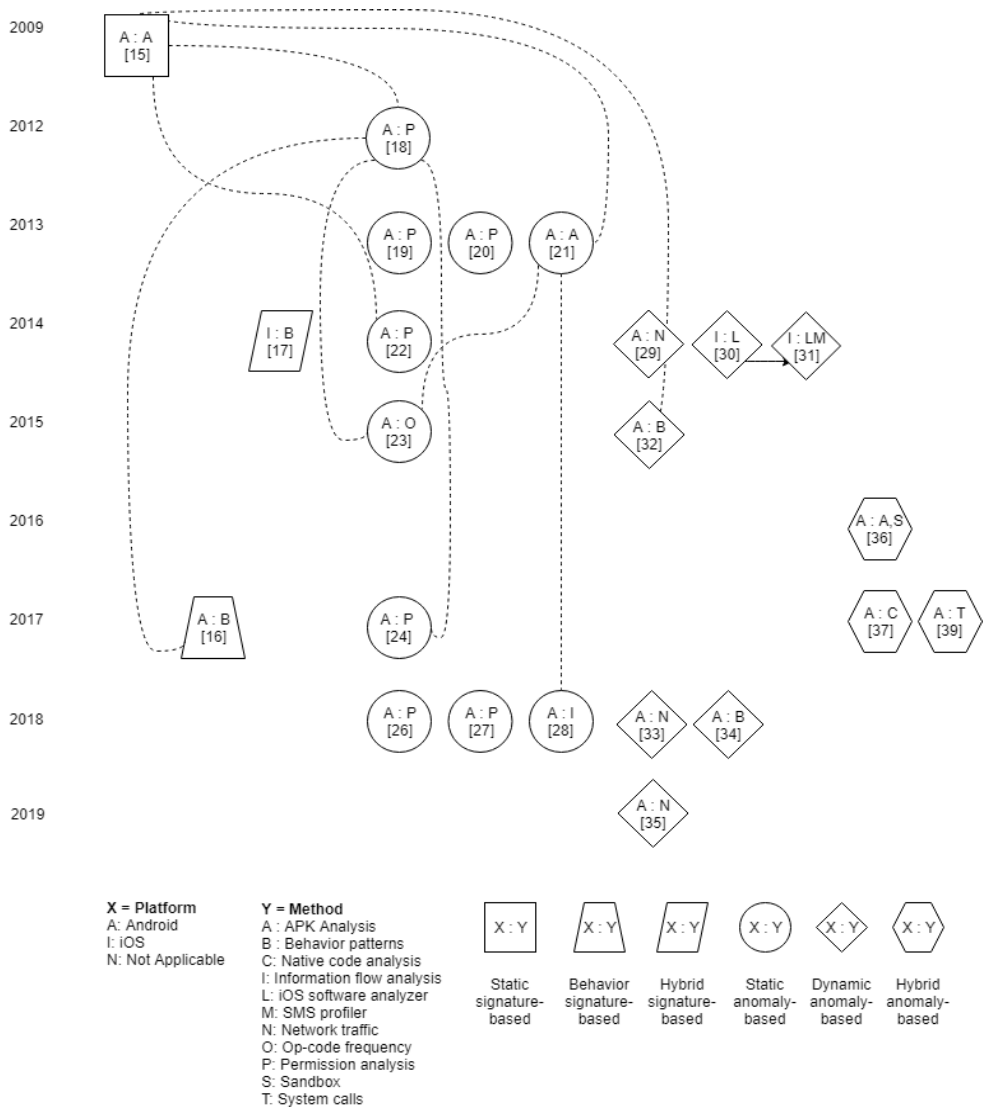

Fig. 2 Malware detection techniques in chronological order

cording to the authors, can produce 86.5% detection rate. Unfortunately, their method cannot detect malicious behavior that is present in native code, which is the case for some of the latest mobile malware.

- Chen et al [33] produced the highest accuracy rate among dynamic anomaly-based approaches. However, while this approach can be highly accurate, it can only detect a subset of malware samples, i.e. those that generate considerable network traffic.
- iOS Detection approaches, such as the work proposed by Damopoulos et al. [30], [31], produce high accuracy results, however these approaches require jailbreaking [40], which could put the device at risk and make the end-user reluctant to employ it.
- Hanlin et al. [36] use sandboxing to safely analyze malware behavior. Although this is a rather promising approach, previous research has shown that some mobile malware are able to detect emulators by looking into several device features [41].
- Some methods combine 2 detection categories into a hybrid solution so as to detect a wide range of malware types. Several of these hybrid solutions carry out mobile malware detection on both the host and cloud, including [17], [31], [36], [39]. While hybrid solutions could offer many benefits, the small amount of reported results from the works included in Sect. 3, as well as previous work [45] suggests that these benefits should be subject to careful examination.

Some approaches were rendered as inconclusive during this survey due to doubtful methodologies or metrics. These approaches are:

- Canfora et al. [23] showed a promising accuracy rate of up to 95% using OP-code frequency analysis, but their results are doubtful due to outdated app samples dated from 2012.
  Tao et al. [27] showed high F1 score, but the authors used an outdated Android OS version and malware

samples.

- Damopoulos et al. [30] proposed a promising approach for the iOS platform and reported zero FPR, but failed to report on essential data, such as the number of non-malware samples used.

Most of the techniques surveyed in Sect. 3 still lack in detecting zero-day malware, but this is somewhat expected. Furthermore, with the current sophistication of malware, it is difficult to detect it through traditional rule matching using existing technologies [42], [43]. This may be the main reason behind the large number of malicious apps still on the loose in official app stores. Therefore, future research efforts should concentrate on clarifying how to efficiently join detection techniques into hybrid solutions with the purpose of increasing the subset of malware which can be detected, as proposed in previous work [44], but also offer actual detection improvement [45].

5. Conclusions

This work provides a state-of-the-art survey on the timely topic of mobile malware detection techniques. To do so, we categorized and succinctly analyzed the various detection schemes as proposed in the literature during the last 8 years, i.e., from 2011 to 2018, based on their detection method. We also highlight on the benefits and limitations per category of techniques and per examined scheme where applicable, in an effort to offer a comprehensive overview of this challenging and fast evolving topic. As a side contribution, we elaborated on the existing interrelations between the examined works, which not only reveals the major influencers in this fast evolving research area, but also the chief challenges to be addressed in the near future.

References

[1] S. Peng, Min Wu, G. Wang, S. Yu, "Propagation Model of Smartphone Worms Based on Semi-Markov Process and Social Relationship Graph," Comput. Secur., vol.44, pp.92–103, 2014.
[2] McAfee, "Mobile Threat Report," https://www.mcafee.com/enterprise/en-us/assets/reports/rp-mobile-threat-report-2018.pdf, accessed May 10 2018.
[3] Symantec, "Motivations of Recent Android Malware," http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/motivations_of_recent_android_malware.pdf, accessed Dec. 10 2018.
[4] P. Yan, Z. Yan, "A survey on dynamic mobile malware detection," Softw. Qual. J., vol.26, no.3, pp.891–919, 2018.
[5] M. La Polla, F. Martinelli, and D. Sgandurra, "A Survey on Security for Mobile Devices," IEEE Commun. Surv. Tutorials, vol.15, no.1, pp.446–471, 2013.
[6] E. Gandotra, D. Bansal, S. Sofat, "Malware Analysis and Classification: A Survey," Journal of Information Security, vol.5, no.2, pp.56–64, 2014.
[7] P. Yan, Z. Yan, "A survey on dynamic mobile malware detection," Softw. Qual. J., vol.26, no.3, pp.891–919, 2018.
[8] S. Peng, Min Wu, G. Wang, S. Yu, "Modeling the dynamics of worm propagation using two-dimensional cellular automata in smartphones," J. Comput. Syst. Sci., vol.79, no.5, pp.586–595, 2013.
[9] M. Anagnostopoulos, G. Kambourakis, and S. Gritzalis, "New facets of mobile botnet: architecture and evaluation," Int. J. Inf. Secur., vol.15, no.5, pp.455–473, 2016.
[10] S. Yu, G. Wang, and W. Zhou, "Modeling malicious activities in cyber space," IEEE Netw., vol.29, no.6, pp.83–87, 2015.
[11] thehackernews.com, "BankBot Returns On Play Store – A Never Ending Android Malware Story," https://thehackernews.com/2017/11/bankbot-android-malware.html, accessed July 9 2018.
[12] Kaspersky, "Hidden miners on Google Play," https://www.kaspersky.com/blog/google-play-hidden-miners/21882/, accessed July 9 2018.
[13] securelist.com, "Mobile malware evolution 2017," https://securelist.com/mobile-malware-review-2017/84139/, accessed July 9 2018.
[14] A. Amamra, C. Talhi, and J.-M. Robert, "Smartphone malware detection: From a survey towards taxonomy," 2012 7th International Conference on Malicious and Unwanted Software, pp.79–86, 2012.
[15] W. Enck, M. Ongtang, P. McDaniel, "On lightweight mobile phone application certification," Proc. 16th ACM conference on Computer and communications security - CCS '09, pp.235–245, 2009.
[16] C.-M. Chen, G.-H. Lai, and J.-M. Lin, "Identifying Threat Patterns of Android Applications," AsiaJCIS, pp.69–74, 2017.
[17] D. Papamartzivanos, D. Damopoulos, and G. Kambourakis, "A Cloud-based Architecture to Crowdsource Mobile App Privacy Leaks," Proc. 18th Panhellenic Conference on Informatics - PCI '14, pp.1–6, 2014.
[18] D.-J. Wu, C.-H. Mao, T.-E. Wei, H.-M. Lee, and K.-P. Wu, "DroidMat: Android Malware Detection through Manifest and API Calls Tracing," AsiaJCIS, pp.62–69, 2012.
[19] B. Sanz, I. Santos, C. Laorden, X. Ugarte-Pedrero, P.G. Bringas, and G. Álvarez, "PUMA: Permission Usage to Detect Malware in Android," Advances in Intelligent Systems and Computing, vol.189, pp.289–298, Jan. 2013.
[20] N. Peiravian and X. Zhu, "Machine Learning for Android Malware Detection Using Permission and API Calls," 2013 IEEE 25th International Conference on Tools with Artificial Intelligence, pp.300–305, 2013.
[21] S. Chakradeo, B. Reaves, P. Traynor, and W. Enck, "MAST: Triage for market-scale mobile malware analysis," WiSec 2013 - Proc. 6th ACM Conference on Security and Privacy in Wireless and Mobile Networks, 2013.
[22] S. Liang and X. Du, "Permission-combination-based scheme for Android mobile malware detection," 2014 IEEE International Conference on Communications (ICC), pp.2301–2306, 2014.
[23] G. Canfora, F. Mercaldo, and C.A. Visaggio, "Mobile malware detection using op-code frequency histograms," Proc. 12th International Conference on Security and Cryptography, pp.27–38, 2015.
[24] M. Yusof, M.M. Saudi, and F. Ridzuan, "A new mobile botnet classification based on permission and API calls," 2017 Seventh International Conference on Emerging Security Technologies (EST), pp.122–127, 2017.
[25] Google, "Google play," https://play.google.com/store/apps, accessed June 6 2018.
[26] J. Li, L. Sun, Q. Yan, Z. Li, W. Srisa-an, and H. Ye, "Significant Permission Identification for Machine-Learning-Based Android Malware Detection," IEEE Trans. Ind. Inf., vol.14, no.7, pp.3216–3225, 2018.
[27] G. Tao, Z. Zheng, Z. Guo, and M.R. Lyu, "MalPat: Mining Patterns of Malicious and Benign Android Apps via Permission-Related APIs," IEEE Trans. Rel., vol.67, no.1, pp.355–369, Mar. 2018.
[28] F. Shen, J.D. Vecchio, A. Mohaisen, S.Y. Ko, and L. Ziarek, "Android Malware Detection using Complex-Flows," 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), pp.2430–2437, 2017.
[29] A. Shabtai, L. Tenenboim-Chekina, D. Mimran, L. Rokach, B. Shapira, and Y. Elovici, "Mobile Malware Detection through Analysis of Deviations in Application Network Behavior," Comput. Secur., vol.43, pp.1–18, 2014.

[30] D. Damopoulos, G. Kambourakis, S. Gritzalis, and S.O. Park, "Exposing mobile malware from the inside (or what is your mobile app really doing?)," Peer-to-Peer Netw. Appl., vol.7, no.4, pp.687–697, Dec. 2014.

[31] D. Damopoulos, G. Kambourakis, and G. Portokalidis, "The Best of Both Worlds: A Framework for the Synergistic Operation of Host and Cloud Anomaly-based IDS for Smartphones," Proc. 7th European Workshop on System Security - EuroSec '14, pp.1–6, 2014.

[32] J.-W. Jang, H. Kang, J. Woo, A. Mohaisen, and H.K. Kim, "Andro-AutoPsy: Anti-malware system based on similarity matching of malware and malware creator-centric information," Digital Investigation, vol.14, pp.17–35, Sept. 2015.

[33] Z. Chen, Q. Yan, H. Han, S. Wang, L. Peng, L. Wang, and B. Yang, "Machine learning based mobile malware detection using highly imbalanced network traffic," Inform. Sciences., vol.433-434, pp.346–364, 2018.

[34] V. Kouliaridis, K. Barmpatsalou, G. Kambourakis, and G. Wang, "Mal-Warehouse: A Data Collection-as-a-Service of Mobile Malware Behavioral Patterns," 2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), pp.1503–1508, 2018.

[35] S. Wang, Z. Chen, Q. Yan, B. Yang, L. Peng, Z. Jia, "A mobile malware detection method using behavior features in network traffic," J. Netw. Comput. Appl., vol.133, pp.15–25, 2019.

[36] H. Zhang, K.D. Pham, Y. Cole, L. Ge, S. Wei, W. Yu, C. Lu, G. Chen, D. Shen, and E. Blasch, "ScanMe mobile: a cloud-based Android malware analysis service," ACM SIGAPP Applied Computing Review, vol.16, no.1, pp.36–49, 2016.

[37] S. Alam, Z. Qu, R. Riley, Y. Chen, and V. Rastogi, "DroidNative: Automating and optimizing detection of Android native code malware variants," Comput. Secur., vol.65, pp.230–246, 2017.

[38] Android, "Android Runtime," https://source.android.com/devices/tech/dalvik, accessed Oct 18 2018.

[39] F. Tong and Z. Yan, "A Hybrid Approach of Mobile Malware Detection in Android," J. Parallel. Distr. Com., vol.103, pp.22–31, 2017.

[40] C. Miller, D. Blazakis, D. Daizovi, S. Esser, V. Lozzo, and R.-P. Weinmann, iOS Hackers Handbook, John Wiley & Sons, Indianapolis, 2012.

[41] T. Petsas, G. Voyatzis, E. Athanasopoulos, M. Polychronakis, and S. Ioannidis, "Rage Against the Virtual Machine: Hindering Dynamic Analysis of Android Malware," Proc. 7th European Workshop on System Security - EuroSec '14, pp.1–6, 2014.

[42] Q. Zhou, F. Feng, Z. Shen, R. Zhou, M.-Y. Hsieh, and K.-C. Li, "A novel approach for mobile malware classification and detection in Android systems," Multimed. Tools. Appl., vol.78, no.3, pp.3529–3552, 2019.

[43] S. Sharmeen, S. Huda, J.H. Abawajy, W.N. Ismail, and M.M. Hassan, "Malware Threats and Detection for Industrial Mobile-IoT Networks," IEEE Access, vol.6, pp.15941–15957, 2018.

[44] S. Arshad, M.A. Shah, A. Wahid, A. Mehmood, H. Song, and H. Yu, "SAMADroid: A Novel 3-Level Hybrid Malware Detection Model for Android Operating System," IEEE Access, vol.6, pp.4321–4339, 2018.

[45] A. Damodaran, F.D. Troia, C.A. Visaggio, T.H. Austin, and M. Stamp, "A comparison of static, dynamic, and hybrid analysis for malware detection," J. Computer Virology and Hacking Techniques, vol.13, no.1, pp.1–12, 2017.

Vasileios Kouliaridis holds a M.Sc. in Information and Communication Systems Security from the department of Information and Communication Systems Engineering of the University of the Aegean. He is currently a PhD candidate at the Department of Information and Communication Systems Engineering of the University of the Aegean. His research interests are in the fields of mobile security and mobile malware analysis.

Konstantia Barmpatsalou received the B.Sc. (Diploma) degree in information systems engineering and the M.Sc. degrees in information systems management and information security from the Department of Information and Communication Systems Engineering, University of the Aegean, Samos, Greece. She is currently pursuing the Ph.D. degree with the Department of Informatics Engineering, University of Coimbra. Her research interests include concepts such as digital forensics, information security, networks security, and intelligent systems.

Georgios Kambourakis received the Ph.D. degree in information and communication systems engineering from the Department of Information and Communications Systems Engineering, University of the Aegean, Greece, where he is currently a Full Professor. Since Oct. 2019, Georgios is on unpaid leave from the University, while he is working for the European Commission at the European Joint Research Centre, Ispra, VA, Italy. His research interests are in the fields of mobile and wireless networks security and privacy. He has over 120 refereed publications in the above areas. More info at: http://www.icsd.aegean.gr/gkamb

Shuhong Chen is an Associate Professor of Computer Science at Guangzhou University, China. She is now a visting scholar at University of Florida, USA. Her research interests include trust evaluation in mobile social networks, performance analysis, and computer networks. Dr. Shuhong Chen has published more than 30 Journal and Conference papers, including Information Sciences, Future Generation Computer Systems, etc. Her research is supported by National Natural Science Foundation of China and Hunan Provincial Natural Science Foundation of China. She has served as General Chair for Ubisafe 2017, 2018 and 2019, Organizing Chair for SpaCCS 2017, IEEE ISPA 2017, and IUCC 2017.