International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. 10, No.1 (Feb-2021)
http://dx.doi.org/10.12785/ijcds/100132

Static Analysis of Malware in Android-based Platforms: A Progress Study

Abdulhamid Ahmed Ali[1] and Antar Shaddad H. Abdul-Qawy[1]

[1]*Department of Science and Information Technology, Faculty of Science, SUMAIT University, Zanzibar, Tanzania*

Received 20 Mar. 2020, Revised 10 Sep. 2020, Accepted 25 Dec. 2020, Published 8 Feb. 2021

Abstract: Android-based platforms enable various applications to request and gain permissions when they need to access the resources of our mobile-phones. This keeps users' data and credentials on hazards and makes them vulnerable to attackers. Several research works have been conducted on this issue and numerous techniques have been developed for detecting malware. Some of these techniques focus on static analysis by inspecting the application package to discover any suspicious hidden code. Such static analysis based schemes are commonly utilized in anti-virus software, including the signature-based and the permission-based mechanisms. In this context, this paper provides a progress study for static analysis of malware in Android-based platforms. We initially investigate common types of malware and present the main categories of malware analysis methods. We, then, provides a literature review on some research works that have been introduced in the last few years on static analysis of Android malware, for both signature-based and the permission-based approaches. However, it is encouraged to provide novel ideas that can help in developing innovative solutions and intelligent systems for detecting various malware in our digital world.

Keywords: Malware Detection, Static Analysis, Android Applications, Signature-based, Permission-based.

1. INTRODUCTION

Nowadays, the rapid growth in various aspects of ICT (Information and Communication Technology) has led to major changes in the way of managing our life [1, 2]. The use of mobile phones, tablets, laptops, desktops, and various smart devices is being adapted recently by most of the individuals, workgroups, and enterprises, for accomplishing different tasks of our daily activities [3]. The open-source platforms, such as Android OS, have been widely utilized in millions of mobile devices over the world. Users are allowed to freely download, install, and run different third-party applications. This has led to many advantages and disadvantages. One of the disadvantages is the invasion of Malwares [4]. Attackers take the opportunity of such activities to reach their targets and achieve their aims on Android-based devices. This can be realized by injecting/writing a malicious code within the applications they develop so that the targeted information (e.g., valuable, private, and sensitive data) is transferred to them once the user runs such applications, with which the malicious code starts running in the background [5]. This happens without the user awareness of what is occurring while he/she is enjoying other services provided by such harmful applications [5, 6, 7].

E-mail: hamid@sumait.ac.tz, antarabdulqawy@sumait.ac.tz

Therefore, malware detection, classification, analysis, and treatment are considered as key research concerns presently [8]. Developing robust mechanisms to ensure trustable and reliable third-party applications for such rapidly growing technology is highly needed. Innovative solutions can be efficiently used in reducing the expert efforts and manual work required in malware analysis [9]. The aim is to allow data and algorithms to drive decisions that require finding-out correlations between large numbers of samples turns out to yield many accurate results than that obtained when humans do the job [10]. This helps in avoiding unauthorized access to sensitive data and private information, as well as preventing suspicious codes for being running hiddenly on our daily used smart devices. This paper provides a valuable study of static analysis approaches introduced for detecting malwares on Android-based smartphones. We start by overviewing common types of malware software in Section 2. We, then, discuss different methods utilized in detecting and preventing malware on Android OS platforms in Section 3, and focus, in particular, on static analysis methods in Section 4. Finally, in Section 5, we introduce a review study for several methods, models, and mechanisms proposed in literature for static malware detection and analysis in Android operating systems. We conclude our paper in Section 6.

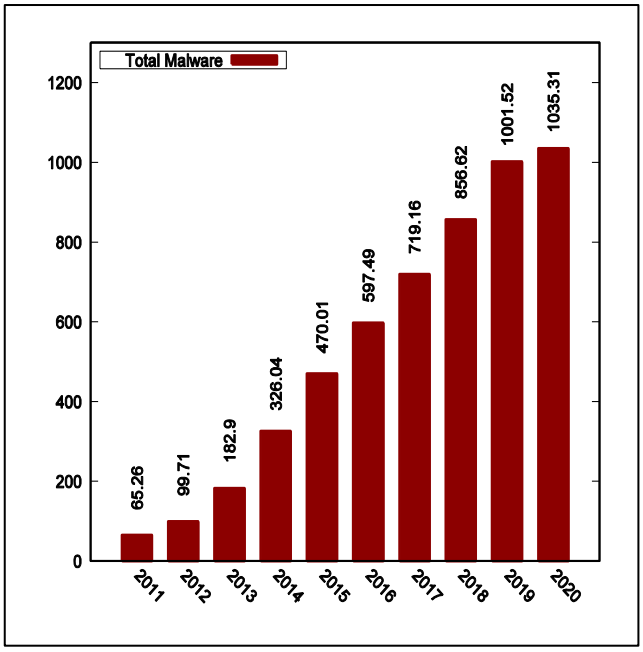

Figure 1. New malware increase in last 10 years (data is adapted from:[14])

2. MALWARE'S OVERVIEW

A malware is defined as application software that includes deleterious code to hurt or steal information of users once it is being run on the targeted devices [11]. Malware has been widely used as a weapon to launch cyber-attacks to target individual devices as well as IT-based systems. Such attacks may compromise the system internals (e.g., memory, files, communication ports) and externals (e.g. of cyber-physical systems), steal sensitive and very important information, deteriorate system performance, and carry-out legal tasks once they gain unauthorized access [12]. Open-source platforms and system software, on which developers heavily rely, are more prone to malware software created by attackers intentionally. The attackers use this chance as an opportunity to get things they want by injecting/writing some malicious code in the system/user applications. When users run such applications, the malicious process starts without users being aware of what is happening in the background of their devices and systems [13]. However, malware shows a noticeable continuing in their trends of evolution. Figure 1 shows how the new malware software rapidly increase over the last ten years [14], while Figure 2 depicts a breakdown of malware software as reported in [15] based on analysis study for the two last years. Below, we discuss some of these well-known malware programs, which have been classified into a number of categories based on their behaviors.

A. Viruses

Viruses are known to be a small piece of malicious code intending to harm the systems, steal data, create botnets, erase files, or render advertisements. A virus is capable to replicate its code and propagate to files of other machines by inserting itself to common programs, and, hence, whenever these host programs are executed by users, the code of the virus also executes [16]. This replication into the other existing programs is the main characteristic in defining virus. Thus, the system can be infected initially through programs which need to be run only once. In addition, the virus can also infect various executables, script files, and vulnerable web applications [17]. Viruses can be distributed to other systems with the help of network connections and portable storage media (such as CD, USB, etc.), which also may be corrupted by the viruses they transfer [18].

B. Worms

Computer worms are self-contained programs which have the ability to replicate functional copies of themselves and spread through network connections [19]. Worms are considered amongst the common and earlier malware software [20], where they, typically, run in the system background without the user's knowledge. They exploit OS vulnerabilities, so they can cause a dangerous deterioration to the system and network performance by consuming a huge amount of bandwidth and overloading web servers. This happens by the means of encrypting and deleting files as well as sending junk emails (such as Melissa and My doom). Also, this type of malware can use a harmful payload to damage host devices beyond disseminating the worms [21]. However, the key difference between worms and common computer viruses

is their ability to replicate and spread themselves without relying on end-user activities or any other program. Examples of malicious worms include Iloveyou, SQL Slammer, and MS Blaster worms [22].

C. Trojans

Trojans (also called Trojan horse) are the dominant type of malware that imitate the behavior of normal programs and real applications to do a specific function, but in fact, do another. This is to trick users into downloading and installing them, so they can hijack users' credentials (e.g., at the time of login to the system), and run harmful tasks such as formatting hard disk, deleting files, or running applications that can defeats system security or authentication procedures [23]. The main target of Trojans is to have full control of the infected system remotely, so they can damage system resources including files and data stored in the HDD, and sometimes they can deny access to essential system services [24]. Once the attackers gain unauthorized access to a hijacked system, they become able to steal data (financial data, login credentials, etc.), install other malware software, observe user activities, modify user documents, use the system in botnets, and hide their ongoing malicious activities.

D. Spywares

This type of software is used by attackers to monitor and collect personal information of users from their devices (with or without the user's permission) and sends it to somewhere else on the Internet [25]. Spyware software has become a common issue that invades user's private data. The most frequently visited data by Spywares includes, but not limited to, user credential, email account information, credit card details, and software license keys [26]. These may be achieved by reading various documents, scripts, and files on the infected machine, or by capturing user keystrokes through a key logger. For instance, spywares may attack the settings of internet browser to change your home page or to redirect you to visit an undesired website. This type of malware gets into the system when free software is downloaded from an untrusted source on the internet [27], however, spywares do not have the feature of self-replication, so they need to be downloaded and installed on the intended machine.

E. Adware

Adware is similar to spyware, where both attempt to gather information about users and their activities. Adware software is advertising-oriented malware, which downloads, plays, or displays advertisements on the infected system once it is being used or copied into user devices [27]. This is usually happening while the malicious software is running, where the aim of Adware is to finance the distribution of the software product as a freeware application (which does not need a payment) [25]. Sometimes, this type of malware pop-ups an advertisement window, redirects users to certain websites for the purpose of marketing and making a sale, or starts installing unknown programs without users permissions. However, most of the adware software monitor users' habits and then try to show them the advertisements that fits what they are browsing or looking for [28]. For instance, searching in google for "IoT", may lead to appearing a pop-up window containing an advertisement

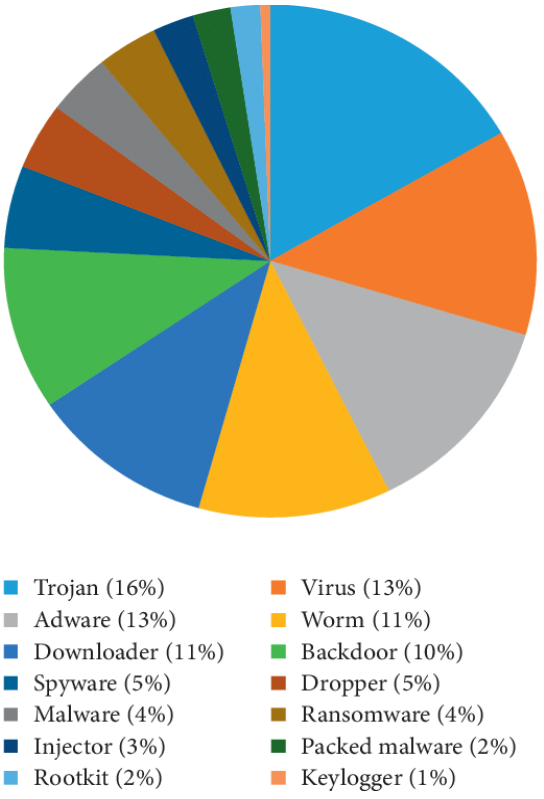

Figure 2. Breakdown of malware software (source:[15])

for "microcontroller used in IoT". Normally, adware codes are combined with some of the software/applications which are available for free on Internet (e.g., KaZaa and BearShare). However, like spyware, adware software do not have the ability to replicate themselves.

F. Others

The above-discussed software represent the main types of malwares used by attackers. In addition, there are many other types of malwares including ransomware, rootkits, grayware, malvertising, crimeware, scraper, cryptojacking, hybrid malware, etc. However, the major ways by which the malware can spread include software vulnerabilities, homogeneity, backdoors and unintended download [17]. Table I presents the names of top mobile-malwares as reported in February 2019 by Symantec [29].

TABLE I. NAMES OF TOP MOBILE-MALWARES (SOURCE: [29]).

Threat Name	Percentage (%)
Malapp	29.7
Fakeapp	9.1
MalDownloader	8.9
FakeInst	6.6
Mobilespy	6.3
HiddenAds	4.7
Premiumtext	4.4
HiddenApp	2.5
MobileSpy	2.8
Opfake	2.0

3. METHODS OF MALWARE ANALYSIS

For secured systems, we cannot say, in real scenarios, that a system or a device is fully protected, or it cannot be breached and compromised. However, it can be said that the level of security of the system or the device has been hardened and it cannot be easily breached. This means that a system may be compromised but it may take more time and higher efforts from attackers until it happens.

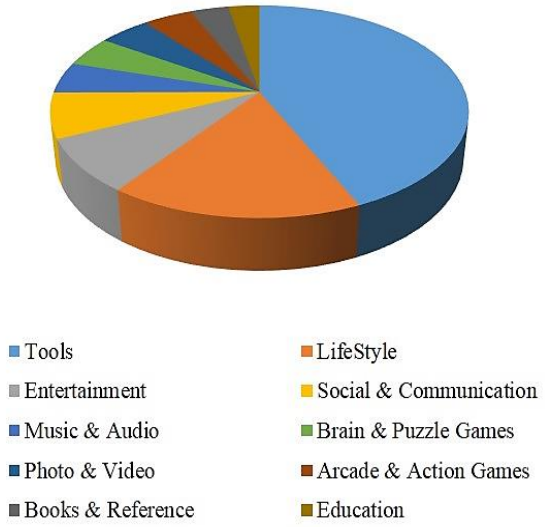

Figure 3. Top malicious mobile app categories (adapted from [29])

Such claims are concluded from observations of several researchers, which indicate that everyday attackers are continuously working for developing ways and techniques to be used in stealing information and hacking different systems. Currently, the attackers have changed their target and focus more on mobile phones and ubiquitous devices, due to the lower level of security/protection initially implemented on them, third-party involvement, and open source systems utilized in most of the platforms running such devices [30]. Figure 3 depicts the top malicious mobile app categories through which mobile devices can be attacked as reported in February 2019 by Symantec [29]. The significant need to develop novel approaches for malware detection and analysis has been encouraged recently. The aim is to know the time at which your devices were got invaded, by which type of malware, and how to get rid of them before they can cause huge damage to your private information [31].

The most common methods used in detecting and analyzing malware are the static and dynamic analysis methods. The static analysis uses some techniques that efficiently help in evaluating any codes written for a particular application by identifying whether the given application has a harmful API or any malicious behavior. This process of static analysis works on binaries without an actual run of the application. It starts by initially taking the package of the written code of the application and decompose it through utilizing the reverse engineering techniques so that the original code can be obtained. However, some of the developers make virtual functions secretly written in application packages, making it difficult to be detected and/or achieve a clear analysis [32]. The dynamic analysis can take place only when the application is in the running state. This helps in identifying and keeping records of activities happen during the execution period, including that of messages being sent and received, network bandwidth being used for accessing the internet, and the amount of energy being drained from the battery. However, this method may take more time when compared to the static analysis [33]. The testing process of this type can be performed in the virtualized environment, with particular settings. A third scheme, based on combining these two methods, is proposed by several researchers, introducing what is known as a hybrid malware analysis approach [34]. Nonetheless, many security experts, malware analysts, and researchers agreed that among these three schemes, the static analysis detection has shown efficiency and accuracy in detecting malwares in Android-based applications. However, the nature of software and system environment, in which they are used, have a significant impact on the performance and accuracy level of their results.

4. APPROACHES OF STATIC ANALYSIS

Techniques of malware static analysis have been classified into two major categories: signature-based approaches, which directly uses the sequence of binary bytes, and permission-based approaches, which focus on the disassembled binaries. Figure 4 below demonstrates the entire process in most of the applications and models that are built based on the static analysis approaches.

A. Signature-based Analysis

The common mechanism used in today's anti-malware industries is the signature-based methods [35]. This type of schemes is based on the hash values of the byte

sequence found in the application binaries, where meta-heuristic algorithms are used to build information. All the extracted semantic patterns, as well as the application

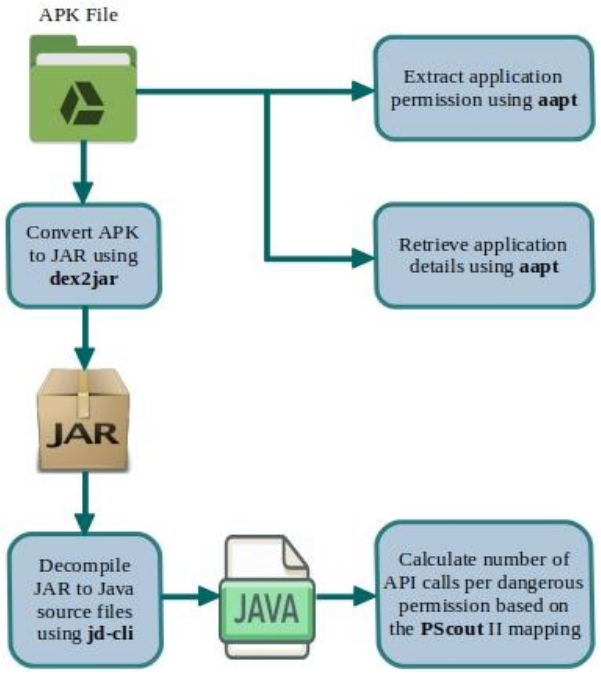

Figure 4. The process of static analysis (adapted from [30]).

characteristics, are used to generate a unique digital signature [36]. In case the created signature is identical to any of the malware signatures already predefined in the market, then the application is considered to be malware. This type of analysis is identified as very effective to detect malware in the versatile applications due to the low rate of false positives outcomes in case the signature is already known. Once the malicious software is detected the created signature is added to the database. However, these approaches face difficulties to identify any new malware types, especially with the quick change of mobile malware. This raises the need for steady periodical updating of the antivirus database and overhauling the already created signature database.

B. Permission-based analysis

Installing third-party applications on Android-based platforms requires, sometimes, certain access to user data for the application to be installed and work perfectly [37]. There exist several APIs in the Android operating system that permit different applications to utilize mobile resources including hardware such as network ports, camera, and storage, as well as the other software including smart-mobile settings [38]. Such permission requests can be considered as important indicators for users to know whether the application is beneficial or malware. The permissions accessed are controlled by the application itself at the time of installation, at which users should be aware of what permissions can violate their privacy or harm mobile devices. Technically, the access permissions are coded in the androidmanifest.xml file in most of the applications [39]. Also, by default, the applications have no permission to access or use the data stored on mobile phones, which limits their functionalities in most of the cases. However, some applications request for permissions that are not likely to be used by the given application. Permitting such applications to access the data may harm the security and expose the private information. So, users should make sure that any application has declared all the permission it needs, and be careful when allowing access to their data during the installation process. The permission-based analysis addresses only the adroidmanifest.xml file in order to extract the various permissions from the applications instead of analyzing all the application files [40].

5. REVIEW OF STATIC ANALYSIS METHODS

In this section, we provide a literature review of some research studies proposed recently for malware static-based analysis in both the signature-based and permission-based approaches. We conclude the section with brief highlighting of machine-learning based static analysis techniques, indicating their relevance and rapid growth recently.

In [41], the authors introduced a new application called Risk-Ranker which has been designed specifically to proactively detect android malware. Risk ranker is an automatic scheme that sifts through various applications to detect security threats on user devices. The authors classified the probable risks into three levels: low, medium, and high. Each has different effects on the devices and a different way to compromise them without permission form users. Based on this categorization, Risk-Ranker is designed to detect the risk that may come from untrusted applications. A systematic method is provided to map each application to one of these risk categories, so by this, it can minimize the number of applications that may need further verification. Two analysis modules are used to detect malicious behaviors. One is for handling non-obfuscated applications, while the other is for detecting apps' behaviors that may harm user devices. They used a set of signatures corresponding to different known vulnerabilities, which contains their essential characteristics, so malicious apps can be detected once they try to exploit these vulnerabilities. The output of the two modules are, then, used for prioritizing suspicious applications that need more analysis. The authors evaluated the Risk-Ranker prototype on a large number of apps and showed a better performance. In [42], Zheng, et al. introduced an implementation of an automatic system for signature-based static malware analysis. The system, which is called DroidAnalytics, helps in obtaining opcode-level information of the applications so the malicious logic can be quickly retrieved, associated, and revealed, and signature is easily generated. The authors

used a parsing algorithm to extract information from .apk package file and retain it in a predefined data structure; while a mechanism with three-phases is utilized for the signature generation to identify applications based on API calls' sequence. This new scheme first extracts classes and methods form .dex file of the application package, then an API calls' table is used to generate signatures of all the methods and classes, and lastly, the application signature is created as a composition of all these already generated signatures. For evaluation, the authors used 150,368 samples to identify malware apps from different families and showed the efficiency of the proposed system.

In [43], the authors focused on the process of increasing users' awareness of the permissions granted to the applications for accessing their data on Android mobile phones. They introduced APK Auditor, a permission-based lightweight malware detection system for Android-based smartphones. The system has three main components of its static analysis: a database for storing apps information and results of operation, a user client, and a central server. The client provides the server with information about the app that needs to be analyzed. The server downloads the app and communicates with the database to manage and control the entire analysis process, then gives feedback to the client. By this, the user can avoid installing malware on his/her device before checking it, and hence avoiding any malicious behavior as well as maintaining the system performance by saving the system resources from being used during the analysis process. APK Auditor client on the user device gives the user two options: analyzing applications on the Play-Store (through https using its package-name), which can be downloaded by the central server, or analyzing the local applications already downloaded on the user device. The server extract permissions of apps from the manifest file. It then calculates PMS (permission-malware-score) for all permissions including dangerous ones depending on their existence in the apps, and AMS (application-malware-score) as the sum of PMSs. AMS is used as an indication to judge whether an apps is malicious or not. If its values exceed the given threshold set by APK Auditor, then the app is considered as harmful and reported as malware application. The authors experimented with 8762 apps (6909 are malicious apps and 1853 benign apps) and showed a higher detection accuracy. In [44], the author proposed a new accurate, and scalable malware detection scheme based on family-signature for flexible static analysis of android malicious applications. In this solution, extracted binary-patterns of applications are used as a representative signature of its family, where various applications are classified into grouped (families) by estimating signature similarity based on the concept of that app variants mostly retain the same code and resources. The structure of signature includes name, character stings, method name, and their bodies, which are extracted from DEX file in .apk package. The proposed method consists of two consecutive stages: signature-creation stage and malware-detection stage. In the first stage, the character strings and binary-patterns are extracted from DEX file of known malware with calculating a weight for each based on its sharing level in representing the family (PSR), where the signatures with weight values less than a threshold are deleted to keep higher accuracy. In the detection process stage, hashed values and hash map are used with a dictionary-search method for family-signature matching with the set of patterns of DEX file for the targeted application in a constant searching time. In their evaluating comparison, the authors used 5846 samples of android malware correspond to 48 families and showed that this approach exhibits a high level of accuracy in detecting malicious applications and a linear time-complexity with respect to the number of apps.

In [45], a new system is proposed to improve users' security and help in protecting their privacy from malicious applications installed on their android devices. The scheme works statically offline and uses permission-based analysis through clustering and categorization techniques in order to recognize the harmful apps and discard them form the main storage. The proposed system consists of five phases: installed apps identification, permission-extraction, apps-clustering, apps-classification, and apps-removal phases. The authors used package-manager in the first phase of the system, and package-info in the second phase to get apps and permission details respectively. K-mean algorithm is used in the clustering phase in which the permissions of the installed apps are classified into various categories of malware. Then, the naïve Bayesian algorithm is used in the fourth phase for further accurate classification to decide whether the malware is malicious or benign app. So, this helps the user to make his/her decision in the last phase (by listing the malicious applications) to retain or uninstall the apps from its device. In [46], the authors focused on using multi-features for static malware analysis. They proposed a scalable scheme for detecting malware in android devices based on features of the malicious application and probability theory. Their new mechanism extracts two features: the API calls and applications' requested permissions from the .apk packages in order to build their feature sets. Different clustering algorithms are considered with these feature sets based on a polynomial-time method to yield predictions for provided instances. The given predictions, then, are fused through MCDF collaborative approach to make the final decision. Their evaluation results indicate that their solution has better performance even with a larger number of data sources. In [47], the applications' malicious behavior was analyzed by Singh et al. using the static analysis approach. The authors proposed a characterization method by which the applications were examined in order to find out whether the permissions coded in the package of applications (APK byte-code) are correct or not. This was done by matching the permissions being requested to the application programming interface (API) calls. They used reverse engineering software to extract components of .apk package to be analyzed and to filter permissions. The authors found out that most of the developers of Android applications write permissions code with some mistakes which lead to poor requirements of the security.

Shahid at el. [48] proposed a new solution for detecting android-based malware called DroidNative decoder. This static signature-based method works at the level of native-code and aims to detect malicious codes that are hidden either in native-code or in byte-code. For cod representation, the authors utilized MAIL (Malware Analysis Intermediate Language), which, in turn, employs two methods: ACFG and SWOD to build cross-platform (i.e., ARM and x86) malware signatures, and uses a number of patterns to create information about behaviors and structures of assembly codes. This information is then used to optimize the process of detection and analysis. In the testing process, a decision-tree method is used to detect the malicious apps based on the signatures sets with a given empirical threshold. If the match occurs, then the application is marked as malware. The DroidNative evaluation shows improved results with respect to other solutions. Idrees et al, in [49], introduced permissions and intents based mechanism called PIndroid to detect malware applications in android systems. The main aim

TABLE II. SUMMARY OF THE PERMISSION-BASED SCHEMES REVIEWED

Author and Reference	Main focus of the paper	Methodology employed

TABLE III. SUMMARY OF THE SIGNATURE-BASED SCHEMES REVIEWED

Author and Reference	**Main focus of the paper**	**Methodology employed**
M. Grace et al. [41],	Detecting zero-day malware in the existing Android applications.	Using two-order risk analysis modules based on vulnerabilities signatures to detect dangerous behaviors.
Zheng, et al [42]	Quick identification of the high volume zero-day malicious mobile malware.	Using a multi-level signature-generation algorithm at the op-code level (i.e., class/method/app level)
J. Lee et al. [44]	Increasing the robustness and speed of detecting malware variants.	Using automatic family-signature design and matching based on PSR, Tpsr, hash values, and hash map.
Shahid et al. [48]	Utilizing control-flow patterns to achieve automation and reduce obfuscations.	Using MAIL, ACFG, and SWOD for signatures-creation and "decision-tree" for decoding and tagging.
S. Ngamwitroj et al [52]	Exploiting the existing features of the applications to detect malwares	Using the frequencies of feature's usages extracted to designing malicious signature and detecting malware.
A. Jalilian et al . [53]	Extracting the file features in a static mode and without running them.	Using of N-grams distribution and Top-k algorithm in identifying unknown files.

was to protect users from the rapid increase in malicious attacks. The method is integrated with an ensemble scheme to increase detection accuracy. The authors analyzed a number of malware and benign applications collected from different sources. To determine the correlation level (e.g., Pearson correlation coefficient (r)) between the two parameters: permissions and intents, they used correlation coefficient methods and indicated a robust relationship between dangerous intents and dangerous permissions. The proposed system has three major phases: extracting features (permissions and intents), preprocessing, and categorization. Their evaluation results show an improved efficiency in the detection process with a higher accuracy level.

In [50], the authors studied the impact of broadcast-receivers usage patterns by android-based malware through analyzing onReceive() source-code manually. They proposed a static malware detection scheme that uses static analysis of registered broadcast-receivers, which are largely exploited by malicious applications in comparison to benign apps. Both the manifest files (containing permissions and broadcast-receivers) and Java source code are analyzed in this mechanism, where tools of reverse engineering are employed on APKs to retrieve them. Apktool is used to extract classes.dex, and XMLs from APS's resources, while dex2jar and JD tool are used for getting Java source code (the parts related to receivers are used to find the occurrence of specific actions). The analysis process considered 5,723 malicious applications and 48,262 benign from which they extract permissions and actions by parsing manifest files. The prediction stage assumes three different configurations: permissions only, action items only, and permissions and action items together. Then, applies a new data mining processing scheme based on SVM algorithm (which has a statistical origin) to create balanced data-sets and use CV (cross-validation) in order to obtain the proper combinations. The authors indicated in their evaluation that using broadcast-receivers along with permissions in their solution provide better detection results. In [51], a new forensic analysis of the security and privacy implementations in the vault applications of the Android-based devices is introduced. These applications are usually designed to enhance user privacy but are misused in hiding various incriminating files and considered as anti-digital forensic applications. The authors proposed a new method that helps in breaking into a number of vault applications and reconstructing files hidden by applications without login requirements. In addition to reverse engineering APK files, their method extracts forensic artifacts from such applications and then examines them. For this, they installed the applications and changed their setting such as passwords, lock pattern, recovery email, etc., and store some well numbered files in them. They then use "adb pull" command to get the required artifacts and store them in predefined folders, then manually analyzed them. The experimentations and case study they presented show that security holes still there in such privacy applications and can be exploited by practitioners using reverse engineering.

In [52], a signature-based malware detection mechanism is proposed by S. Ngamwitroj and B. Limthanmaphon based on static analysis. In order to build the signature of the malicious apps, the author used data of broadcast-receivers and frequencies of permission usage extracted from the manifest file. The proposed method consists of two main parts: signature creation of malicious apps, and malware-detection process. The first part calculates the frequencies of permissions & broadcast-receiver usage and ranks them in decreasing order. Only some top-most of them, then, are used in designing the malicious signature in a compare-and-trim process. These extracted information and created signatures are used in the second stage of the method (i.e., malware-detection process). The authors evaluated their proposed scheme by examining 1,447 application and showed a significantly improved performance. In [53], a new signature-based scheme for detecting malware is provided which focuses on features' extraction of application (i.e., op-codes, binary-sequences, or API-calls) statically before running them on the user device. The method uses the N-gram distribution and Top k algorithm to define the top most-similar k files when identifying unknown files. The process has three main stages: binary-sequence and op-code extraction, N-gram generation, and file classification. For N-grams algorithm, which is simple and stable when obfuscation exists, each N-gram (N= 1, 2, 3, …, k) denotes an application feature, where deciding the appropriate N will help in maintaining the desired efficiency and performance. The author considered Top-K similar-files approach and used the features taken from .apk package along with features of op-code in order to enhance the strength of feature space and improve detection efficiency. The evaluation results show a significant improvement in accuracy when the Top-K approach, a combination of op-code sequence, and binary-files are employed in classifying benign-files/malware apps.

Recently, data science and machine-learning have revealed efficient potential in developing innovative solutions for automating different aspects of malware investigation and analysis techniques. Several designing and implementation challenges emerge, which makes the need for new solutions and standard mechanisms crucial. The aim is to cope with the rapid increase of malware threats, particularly in parallel with the increase in big data generation. However, this domain is a very broad area of its own, which is our separate work following this progress study. The reader can find more about machine learning mechanism in [9, 54 – 60].

6. CONCLUSION

Malware software are mostly transferred to users' mobile phones when applications are being installed. This may lead to granting attackers unauthorized access to

sensitive and private data stored on the users' devices. Attackers and malware developers work hard in order to develop hacking techniques, applications, and software to steel private date from ubiquitous devices and systems of those who do not care much about securing their information. Malware analysis is known to be the study of functionality, purpose, origin and the impact of malicious applications and software. Then deciding on how to avoid or treat them. This task is, by tradition, known to be exhausting and complicated. It requires expert knowledge of analytics and of reverse engineering together with understanding of software architecture. As technology continues to develop rapidly, it is important to keep our knowledge, and technical skills, in this domain, up to date. The need is to improve user awareness of such malware applications so they can make the right decision at the time of downloading and installation. In addition, innovative solutions, models, standards, and tools are to be developed to facilitate malware detection, analysis, and treatment. In this paper, we have introduced a progress study of malware analysis in Android-based operating systems, centered on static approaches. Common classes of malware software have been presented, followed by discussions on the major techniques used for malware detection and analysis. A literature review is, then, presented on several static analysis solutions including signature-based and permission-based schemes proposed in the last few years. The intended work in the future focuses on considering recent machine learning techniques along with big data analytics within the data-science domain in order to develop a practical multidimensional joint optimization approach to achieve better performance and higher accuracy level of malware analysis.

REFERENCES

[1] M. Dachyar, Teuku Yuri M. Zagloel, and L. Ranjaliba Saragih, "Knowledge growth and development: internet of things (IoT) research, 2006–2018", Heliyon, Vol. 5, No. 8, 2019, pp. 1-14.

[2] Abdul-Qawy, A.S.H., Srinivasulu, T. SEES: a scalable and energy-efficient scheme for green IoT-based heterogeneous wireless nodes. J Ambient Intell Human Computing, Springer, 2019, pp. 10, 1571–1596.

[3] Antar S. H. Abdul-Qawy and T. Srinivasulu, "Greening Trends in Energy-Efficiency of IoT-based Heterogeneous Wireless Nodes", International Conference on Electrical, Electronics, Computers, Communication, Mechanical and Computing (EECCMC), Jan 2018, India, pp. 118 – 427.

[4] Stiborek J., Pevný T., and Rehák, M., "Multiple instance learning for malware classification", Expert Systems with Applications, Vol. 93 No. 1, 2018, pp. 346-357.

[5] John Sammons, and Michael Cross, "Chapter 3 - Software problems and solutions", The Basics of Cyber Safety, Syngress, 2017, pp. 53-74.

[6] Sudhakar Kumar S., "An emerging threat Fileless malware: a survey and research challenges", Cybersecur, vol. 3, No. 1, 2020.

[7] Xu Jiang et al., "Android Malware Detection Using Fine-Grained Features", Scientific Programming, Hindawi, pp. 1-13.

[8] N. Lageman, M. Lindsey and W. Glodek, "Detecting malicious Android applications from runtime behavior", IEEE Military Communications Conference (MILCOM), USA, 2015, pp. 324-329.

[9] Daniele Ucci, Leonardo Aniello, and Roberto Baldoni, "Survey of machine learning techniques for malware analysis", Computers & Security, Vol. 81, 2019, pp. 123-147.

[10] Freeman D., and Chio C., "Chapter 4: Malware Analysis", Machine Learning and Security, O'Reilly Media, Inc., 2018, pp. 125-79.

[11] Darrel Rendell, "Understanding the evolution of malware", Computer Fraud & Security, Vol. 2019, No. 1, 2019, pp. 17-19.

[12] P. Michalopoulos, V. Ieronymakis, M.T. Khan, and D. Serpanos, "An Open Source, Extensible Malware Analysis Platform", MATEC Web of Conferences 188, 05009, 2018.

[13] Jha A. K., Lee, and W. J., "An empirical study of collaborative model and its security risk in Android", Journal of Systems and Software. Vol. 137, No. 1, 2018, pp. 550-562.

[14] Malware AV-TEST-The Independent IT-Security Institute, online: https://www.av-test.org/en/statistics/malware/

[15] Aslan Ömer. Samet Refik and Tanrıöver, Ömer, "Using a Subtractive Center Behavioral Model to Detect Malware", Security and Communication Networks, 2020, pp. 1-17.

[16] Lysne Olav, "Static Detection of Malware", the Huawei and Snowden Questions, Springer International Publishing, 2018, pp. 57-66.

[17] Neil DuPaul, "Common Malware Types: Cybersecurity 101", online:https://www.veracode.com/blog/2012/10/common-malware-types-cybersecurity-101.

[18] Jang J. et al., "Andro-Dumpsys: Anti-malware system based on the similarity of malware creator and malware centric information", Computers & Security, Vol. 58, No. 1, 2016, pp. 125-138.

[19] Bimal Kumar Mishra, and Samir Kumar Pandey, "Dynamic model of worm propagation in computer network", Applied Mathematical. 38 Modelling, Vol, No. 7, 2014, pp. 2173-2179.

[20] Touchette F., "The evolution of malware", Network Security, 2016, pp. 11-14.

[21] Veracode, "COMPUTER WORM, online: https://www.veracode.com/security/computer-worm

[22] Roger A. Grimes, " types of malware and how to recognize them", CSO, May 2019, online: https://www.csoonline.com/article/2615925/security-your-quick-guide-to-malware-types.html.

[23] Z. Huang et al., "A Survey on Machine Learning Against Hardware Trojan Attacks: Recent Advances and Challenges", IEEE Access, vol. 8, 2020, pp. 10796-10826.

[24] Sheen S., Anitha R., and Sirisha P., "Malware detection by pruning of parallel ensembles using harmony search", Pattern Recognition Letters. Vol. 34, No. 14, 2013, pp. 1679-1686.

[25] Ido Dubrawsky, "Chapter 1 - Systems Security", Eleventh Hour Security+, Syngress, 2010, pp. 1-16.

[26] Todd G. Shipley, and Art Bowker, "Chapter 7 - Online Digital Officer Safety", Investigating Internet Crimes, Syngress, 2014, pp. 149-169.

[27] Razak M. F. A. et al., "The rise of "malware": Bibliometric analysis of malware study. Journal of Network and Computer Applications, Vol. 75, No. 1, pp. 58-76.

[28] J. Gao et al., "Should You Consider Adware as Malware in Your Study?", IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER), Hangzhou, China, 2019, pp. 604-608.

[29] Symantec Corporation, "Internet security threat report," Symantec, online: https://docs.broadcom.com/doc/istr-24-2019-en

[30] Abdullah Talha Kabakus, and Ibrahim Alper Dogru, "An in-depth analysis of Android malware using hybrid techniques, Digital Investigation, Vol. 24, 2018, pp. 25-33.

[31] Almin S. B., and Chatterjee M., "A Novel Approach to Detect Android Malware", International Conference on advanced computing technologies and applications, Vol. 45, No. 1, 2015, pp. 407-417.

[32] Weijie Han et al., "MalDAE: Detecting and explaining malware based on correlation and fusion of static and dynamic characteristics", Computers & Security, Vol. 83, 2019, pp. 208-233.

[33] Fan W. et al., "DroidInjector: A process injection-based dynamic tracking system for runtime behaviours of Android applications", Science Direct., Vol. 70, No. 1, 2017, pp. 224-237.

[34] Tong F., and Yan, Z., "A hybrid approach of mobile malware detection in Android", Journal of Parallel and Distributed Computing, Vol. 103, No. 1, 2017, pp. 22-31.

[35] Alireza Souri, and Rahil Hosseini, "A state-of-the-art survey of malware detection approaches using data mining techniques", Hum. Cent. Comput. Inf. Sci., Vol. 8, No. 3, 2018, pp. 1-22.

[36] Shinde S. V., and Manjrekar A. A., "A Review Paper on Effective Behavioural Based Malware Detection and Prevention Techniques for Android Platform", International Journal of Engineering Research and Technology, Vol. 10, No. 1, 2017, pp. 901-907.

[37] Talha K. A., Alper D. I., and Aydin, C., "APK Auditor: Permission-based Android malware detection", Digital Investigation, Vol. 13, No. 1, 2015, pp. 1-14.

[38] Wu S. et al., "Effective detection of android malware based on the usage of data flow APIs and machine learning", Information and Software Technology, Vol. 75, No. 1, 2016, pp. 17-25.

[39] Kaur P., and Sharma S., "Literature Analysis on Malware Detection", International Journal of Electronic and Electrical Engineering, Vol. 7, No. 7, 2014, pp. 717-722.

[40] Rao V., and Hande K., "A comparative study of static, dynamic and hybrid analysis techniques for android malware detection", International Journal of Engineering Development and Research, Vol. 5, No. 2, 2017, pp. 1433-1436.

[41] M. Grace et al., "RiskRanker: scalable and accurate zero-day android malware detection", 10th international conference on Mobile systems, applications, and services (MobiSys), Association for Computing Machinery, USA, 2012 pp. 281–294.

[42] M. Zheng, M. Sun and J. C. S. Lui, "Droid Analytics: A Signature Based Analytic System to Collect, Extract, Analyze and Associate Android Malware", 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, Australia, 2013, pp. 163-171.

[43] K. A. Talha, D. I. Alper, and C. Aydin, "APK Auditor: Permission-based Android malware detection system", Digital Investigation, Vol. 13, 2015, pp. 1-14.

[44] Jehyun L., Suyeon L., and Heejo L., "Screening smartphone applications using malware family signatures", Computers & Security,Vol. 52, 2015, pp. 234-249.

[45] S. B Almin, and M. Chatterjee, "A Novel Approach to Detect Android Malware", Procedia Computer Science, Volume 4, 2015, pp. 407-417.

[46] Sheen S., Anitha R., and Natarajan, "Android based malware detection using a multi feature collaborative decision fusion approach", Neurocomputing, Vol. 151, No. 2, 2015, pp. 905-912.

[47] Singh P., Tiwari P., and Singh, S., "Analysis of Malicious Behaviour of Android Apps", International Conference on Communication, Computing and Virtualization. Vol. 79, No. 3, 2016, pp. 215 – 220.

[48] Shahid Alam et al., "DroidNative: Automating and optimizing detection of Android native code malware variants", Computers & Security, Vol. 65, 2017, pp 230-246.

[49] Idrees F. et al., "PIndroid: A novel Android malware detection system using ensemble learning methods", Computers & Security. Vol. 68, No. 1, 2017, pp. 36-46.

[50] F. Mohsen et al., "Detecting Android Malwares by Mining Statically Registered Broadcast Receivers", IEEE 3rd International Conference on Collaboration and Internet Computing (CIC), CA, 2017, pp. 67-76.

[51] Zhang X., Baggili I., and Breitinger F., "Breaking into the vault: Privacy, security and forensic analysis of Android vault applications", Computers & Security, Vo. 70, No. 1, 2017, pp. 516-531.

[52] S. Ngamwitroj, and B. Limthanmaphon, "Adaptive Android Malware Signature Detection", International Conference on Communication Engineering and Technology (ICCET-18), Association for Computing Machinery, USA, 2018, pp. 22–25.

[53] Jalilian A., Narimani Z., and Ansari E., "Static Signature-Based Malware Detection Using Opcode and Binary Information". 7th International Conference on Contemporary Issues in Data Science (CiDaS-2019), Lecture Notes on Data Engineering and Communications Technologies, vol 45. Springer, 2019, pp. 24-35.

[54] Shalaginov A., et al., Machine Learning Aided Static Malware Analysis: A Survey and Tutorial. In: Dehghantanha A., Conti M., Dargahi T. (eds) Cyber Threat Intelligence. Advances in Information Security, vol 70. Springer, 2018, pp. 7-45.

[55] Daniel Gibert, Carles Mateu, and Jordi Planes, "The rise of machine learning for detection and classification of malware: Research developments, trends and challenges", Journal of Network and Computer Applications, Vol. 153, 2020, pp 1-22,

[56] Quan Le et al., "Deep learning at the shallow end: Malware classification for non-domain experts, Digital Investigation, Vol. 26, 2018, pp. S118-S126.

[57] Milosevic N., Dehghantanha A., and Choo K.-K. R., "Machine learning aided Android malware classification", Computers & Electrical Engineering, Vol. 61, No. 1, pp. 266-274.

[58] Matilda Rhode, Pete Burnap, and Kevin Jones, "Early-stage malware prediction using recurrent neural networks", Computers & Security, Vol. 77, 2018.

[59] ElMouatez Billah Karbab, and Mourad Debbabi, "MalDy: Portable, data-driven malware detection using natural language processing and machine learning techniques on behavioral analysis eports", Digital Investigation, Vol. 28, Supplement, 2019.

[60] Gianni D'Angelo, Massimo Ficco, and Francesco Palmieri, "Malware detection in mobile environments based on Auto-encoders and API-images", Journal of Parallel and Distributed Computing, Vol. 137, 2020, pp. 26-33.

Mr. Abdulhamid Ahmed Ali has received his bachelor degree in Information Technology from Asia Pacific University of Technology and Innovation (APU), Malaysia in 2017, and master degree in Data Science and business analytics in 2019 from APU, Malaysia. He is now working as an Assistant Lecturer of Information Technology at the Faculty of Science, SUMAIT University, Zanzibar, Tanzania. His areas of interest include Big Data, Information System Security, Cloud Computing, System Analysis and System Design.

Dr. Antar Shaddad H. Abdul-Qawy Has received his B.E in Computer Engineering in 2005 from Hodeidah University, Yemen, Master of Technology in Computer Science in 2014 form University of Hyderabad, India, and Ph.D. in Internet of Things, in April 2019 from Department of Electronics and Communication Engineering, Kakatiya University, India. He worked as Assistant Lecturer at Hodeidah University from 2005 to 2012. Now, he is working as a Senior Lecturer of Information Technology at the Faculty of Science, SUMAIT University, Zanzibar, Tanzania. His areas of interest include Internet of Things, Wireless Sensor Networks, Data Science, Green loT, and Energy-Efficient Networks