PAPER • OPEN ACCESS

An Analysis of Machine Learning-Based Android Malware Detection Approaches

To cite this article: R. Srinivasan *et al* 2022 *J. Phys.: Conf. Ser.* **2325** 012058

An Analysis of Machine Learning-Based Android Malware Detection Approaches

R. Srinivasan[1], Karpagam.S[2], M. Kavitha[3], R. Kavitha[4]

[1 3 4] Professor, Department of Computer Science and Engineering, Vel Tech University, Avadi, Chennai - 600062, Tamil Nadu, India.

[2] Associate Professor, Department of Mathematics, Vel Tech Multi Tech Dr Rangarajan Dr Sakunthala Engineering College, VelTech Rangarajan Dr Sagunthala R&D Institute of Science and Technology, Chennai, Tamil Nādu, India.

srinivasanrajkumar28@gmail.com, kavitha@veltech.edu.in, rkavitha@veltech.edu.in

Abstract: Despite the fact that Android apps are rapidly expanding throughout the mobile ecosystem, Android malware continues to emerge. Malware operations are on the rise, particularly on Android phones, it make up 72.2 percent of all smartphone sales. Credential theft, eavesdropping, and malicious advertising are just some of the ways used by hackers to attack cell phones. Many researchers have looked into Android malware detection from various perspectives and presented hypothesis and methodologies. Machine learning (ML)-based techniques have demonstrated to be effective in identifying these attacks because they can build a classifier from a set of training cases, eliminating the need for explicit signature definition in malware detection.

This paper provided a detailed examination of machine-learning-based Android malware detection approaches. According to present research, machine learning and genetic algorithms are in identifying Android malware, this is a powerful and promising solution. In this quick study of Android apps, we go through the Android system architecture, security mechanisms, and malware categorization.

Keywords: Malcode, Malware, Android Security, Machine learning, Feature extraction

1. Introduction

Due to the convenience and efficiency of many apps, as well as continuous improvements Smartphone usage and accompanying applications are rapidly increasing in today's hardware and software on smart devices. [7]. By 2023, 4.3 billion individuals are expected to own a smartphone. The Android operating system is the most popular among smartphone users (OS). In May 2021, it had a market share of 72.2 percent [8]. With a market share of 26.99 percent, Apple iOS is in second place, followed by Samsung, KaiOS, and other minor vendors with 0.81 percent[8].

The official app store for Android devices is Google Play. As of May 2021, there were around 2.9 million applications available on it. AppBrain classifies about 2.5 million of them as standard applications, whereas 0.4 million are labelled as low-quality apps[9]. Because of its global popularity, Android is a more appealing goal aimed at thieves then is extra vulnerable to malware and viruses. Many strategies for identifying these assaults have been proposed in studies, with one of the most prevalent being machine learning [10]. The taxonomical taxonomy of malware is depicted in Figure 1.

Android users and developers are notorious for making blunders that put them in danger and expose them to malware infection threats. As a result, in addition to virus detection, tactics for discovering these weaknesses are critical. The analysis of Android Application Packages (APKs) to construct an appropriate collection of features, the two basic components of malware detection utilising machine learning are training machine and deep learning (DL) algorithms on the produced characteristics to identify infected APKs. As a result, there is a discussion of the various methodologies for APK analysis, such as static, dynamic, and hybrid analysis.

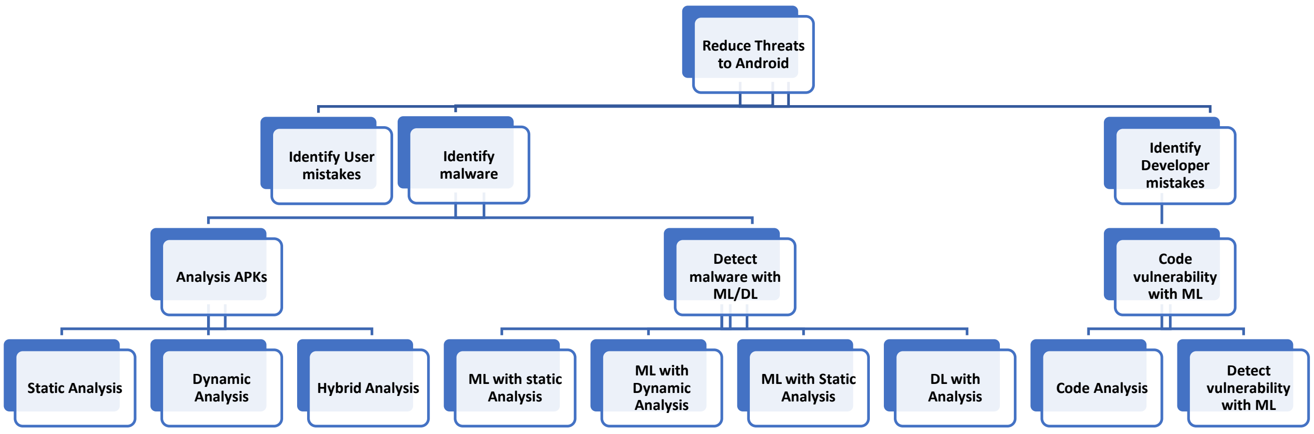

Figure 1. Taxonomy of the malware classification

Android applications are vulnerable to software theft on the open Smartphone market due to the ease with which they may be repackaged. Developers can modify or add to the original programme and then resell it as a new product on the open market. It's possible that the change is malicious. Repackaged malware accounts for 80.6 percent of all malware, illustrating its prevalence and severity, according to researchers. For detecting repackaged malware, there are two approaches: 1) similarity-based detection unique to repackaged malware, and 2) general purpose detection.

Various similarity tests are used by specific repackaged Android app solutions to find truly related apps. The similarity comparison is tied to how applications encode the user's navigation behaviours in View Droid, for example. DNA Droid examines the software dependency trees of users to assess code reuse. MassVet employs UI structures to compare the similarity of apps. Based on opcode sequence properties, Juxtapp and Droid MOSS compare code similarities.

The Android platform has become a key target for malware. Researchers have developed a variety of on-device detection ways to protect against Android malware. These on-device detection methods usually consist of two steps: I acquiring the app's behavioural characteristics from mobile devices, and (ii) sending the acquired features to remote servers for evaluation.

Methods can consistently detect malicious programmes by observing the activities of applications that operate on mobile devices (simply, software). On the other hand, in most cases, infrastructure is a barrier for mobile devices. Malware writers introduce malcode statically and change the control flow to ensure that it runs during repackaging. Repackaged malware is difficult to detect using traditional categorization methods because it behaves similarly to benign malware.

We provide a new approach for identifying repackaged malware on Android that focuses on the software's internal activities. Our method divides an app's code structure into several dependency-based structures, regions, and areas using code heterogeneity analysis (subsets of the code). Each area is defined by its own set of behavioural qualities.

We suggest a solution based on multiple reliance links that identifies security flaws, At the class and method levels, creates and builds graph partitioning code structures, and presents a solution based on numerous dependency linkages.

2. Background

This section covers the architecture and security aspects of Android, as well as potential Android attack vectors. It also contains an overview of the machine learning process and it will help non-ML readers grasp the content of this paper.

2.1 Architecture of Android

The Linux Kernel is used to create Android. Because Linux is open source, it checks the route evidence, offers networking drivers and protocols, virtual memory, device power, and security are all handled by this programme. [12]. The architecture of Android is layered [11].

The Android Runtime is used by Android apps and several system functions (ART). Before the ART, the runtime environment was Dalvik. Dalvik and ART were both created with Android applications in mind[13].

Because Android applications operate on with partial memory, memory and power management are particularly important challenges. As a result, the Android operating system is designed to effectively manage any resource[5]. If an application is not in use at the time, the Android OS, for example, will immediately suspend it in memory. The application life cycle's operating state is referred to as this. It can save power that can be used when the app reopens by doing so. Otherwise, until they are closed, the programmes are inactive [14].

2.2 Security Built-In

Security software is pre-installed on Android cellphones. It's a separate operating system for privileged users. [15]. The Android permission system, as well as the sandboxing method, assist to reduce the program's numerous risks and difficulties. Sandboxing on Android uses unique IDs based on the Linux environment to isolate executing programmes [16]. Apps can only access system resources with permissions granted by the user during installation or reconfiguration. The software will not function if specific permissions are not provided. Upgrades or enhancements to a system increase security and privacy in a variety of ways[16].

3. Related Work

Xu Jiang et al. propose a new method for detecting Android malware that is based on a fine-grained permission system that reflects the differences between malicious and benign applications as machine learning characteristics and, for the first time, provides information about the components' harmful permissions[1].

Tajuddin Manhar Mohammed et al, examine the permissions employed by malicious programmes in depth, as well as the efficacy of permission characteristics used to distinguish between harmful and benign apps[2].

FDP employs static techniques to capture all characteristics and analyse an application in an acceptable amount of time, according to Deqiang Li et al[3].

FDP is more successful at identifying more malware kinds and harmful programmes that declare a small number of damaging rights or permissions that are identical to those claimed by benign applications, according to Amiangshu Bosu et al tests[4].

4. Machine Learning to Detect Android Malware

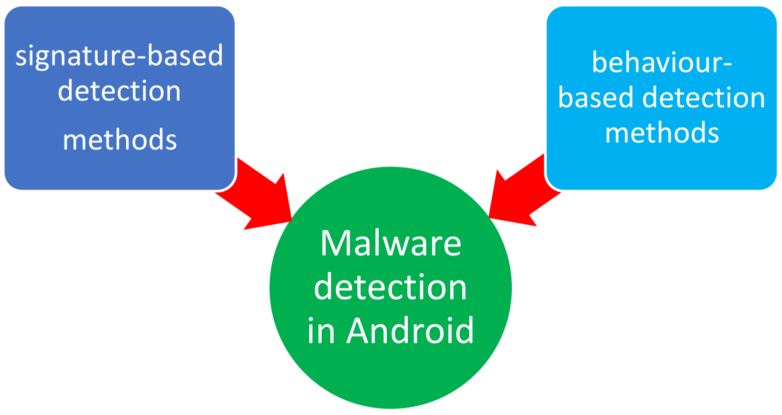
signature-based detection methods
behaviour-based detection methods
Malware detection in Android

Figure 2. Types of detect Android Malware

Figure 2 shows types of malware to detected in android application and the signature-based detection method is simple, efficient, and produces minimal false positives. The binary code of the programme is compared to signatures in a known malware database. This technique, on the other hand, excludes the identification of unknown malware. As a result, behavior-based/anomaly-based detection is the most often used approach. In this case, machine learning and data science methodologies are widely applied [17-18].

4.1 Static, Dynamic, and Hybrid Analysis

As previously stated, some of the suggested ML algorithms in the literature involve analyzing APKs to extract characteristics. Static, dynamic, and hybrid analysis methods are characterized as three analysis strategies [19–21]. Instead of executing it on a mobile device, static analysis may be conducted by analyzing the bytecode and source code or re-engineered APK. Malware is detected via dynamic analysis, which analyses a programme while it is executing in a simulated or real-world context.

4.1.1 Static analysis with Machine Learning

Many research have been undertaken to identify Android malware using both standard ML-based techniques like Decision Trees (DT) and Support Vector Machines (SVM) as well as innovative DL-based models like Deep Convolutional Neural Network (Deep-CNN) [17] and Generative Adversarial Networks (GENAN) [18].

4.1.2 Dynamic Analysis with Machine Learning

The second sort of analysis is dynamic analysis. Malware may be discovered using machine learning after the application has been launched in a runtime environment using this approach. A network-based technique for detecting malware on Android was described in [23].

A detecting application was created as part of this effort. It is divided into three modules: network tracing, network feature extraction, and detection. Running programmers' network activity were tracked, and network traces were gathered on a regular basis in the traces collecting module. The features extraction module gathered information about the network that the apps used.

The detection module made use of the DT, LR, KNN, Bayes Network, and RF algorithms. The RF algorithm was the most accurate of the four (98.7 percent). This technique, on the other hand, was founded on network analysis. If dangerous programmes employed encrypted data, the malware detection accuracy would be lowered.

4.1.3 Hybrid Analysis with Machine Learning

The third approach that may be employed in ML-based Android malware detection is hybrid analysis. [93] outlined three ways to malware detection: signature-based, anomaly-based, and topic modeling-based tactics.

The behavioral-based technique outperformed the signature-based method. According to the findings of this investigation, the SVM classifier utilizing the hybrid analytic methodology beat the other ML approaches.

According to the research on ML-based techniques to malware detection, 65 percent of studies utilised static analysis, 15% used dynamic analysis, and the other 20% employed a mixed analytic strategy. This is seen in Figure 3. Because of the benefits it delivers, such as the ability to uncover more vulnerabilities, localise issues, and save money, static analysis may be more tempting than dynamic analysis. Table 3 shows the chart representation of ML algorithms commonly used for Android malware detection

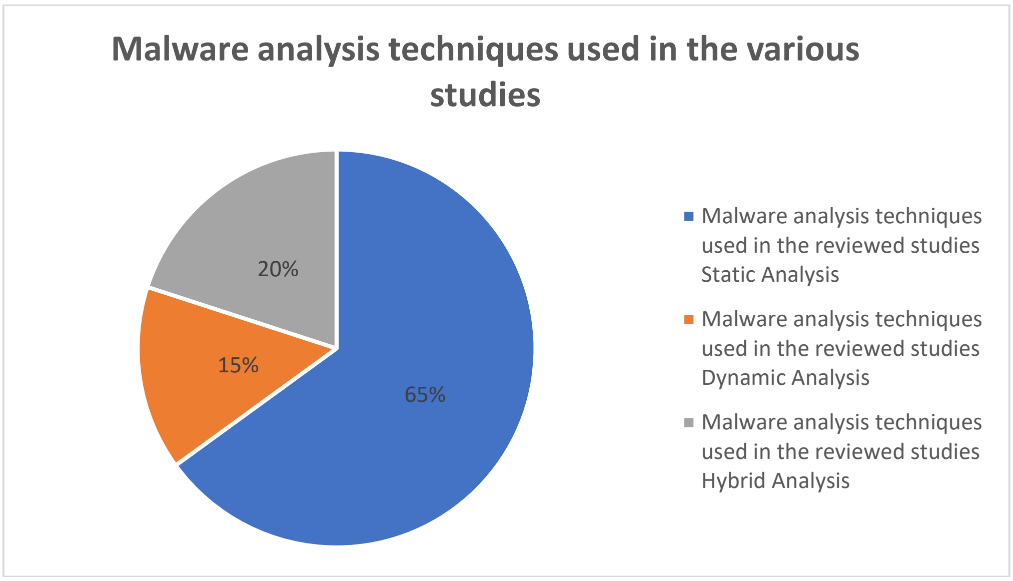

Figure 3. Malware analysis techniques used from various studies.

Table 1. ML algorithms commonly used for Android malware detection

Algorithm	Advantages	Disadvantages
Decision Trees(DT)	Possible with values Simple to comprehend	overfitting problem
k-nearest neighbours	Simple to implement Simple & quick	Sensitive to outliers
convolutional neural network(CNN)	Weight sharing and down sampling help to reduce non-essential factors.	High cost of calculation
Regression Models	Study with statistics	High-dimensional characteristics are difficult to deal with.
SVM	solve nonlinear high-dimensional small-scale issues.	Processing data has a high overhead.

5. Proposed System

The genetic algorithm (GA) is a search-based optimization method based on natural selection and genetic principles. It's a part of the method we recommend. It is widely used to find optimal or near-optimal solutions to tough problems that would otherwise take an eternity to solve. The suggested technique leverages an evolving Genetic Algorithm[5] to build the best-optimized feature subset that may be used to efficiently train machine learning algorithms.

If signature-based approaches fail to detect new types of malware that pose zero-day threats, machine learning-based algorithms are used to identify malware more successfully. Figure 4. Shows architecture Diagram of the proposed method

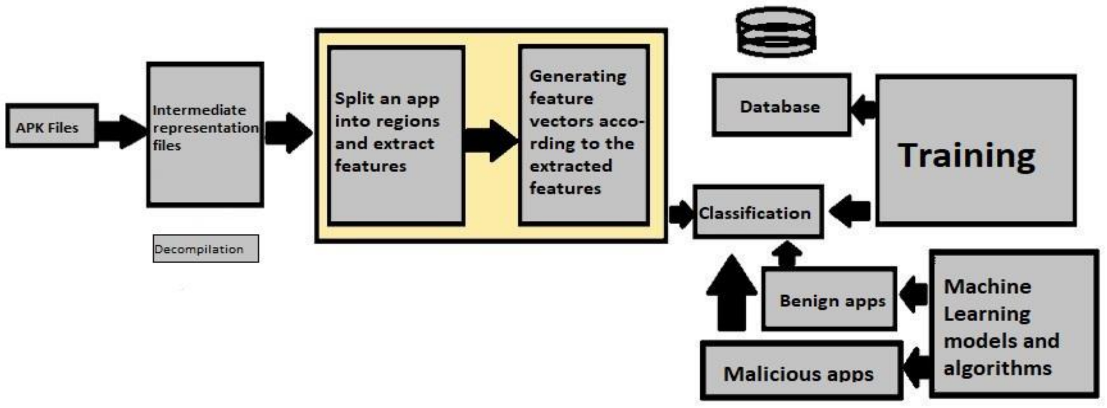
APK Files
Intermediate representation files
Decompilation
Split an app into regions and extract features
Generating feature vectors acco-rding to the extracted features
Database
Training
Classification
Benign apps
Malicious apps
Machine Learning models and algorithms

Figure 4. Architecture Diagram of the proposed method

As illustrated in the flowchart above, we'll start our tests at the beginning of each stage. The first section will concentrate on data and its preparation. Then, as a first step, we'll use Data Augmentation to try to enhance accuracy. To enhance capacity, the next stage is to make some modest changes to network infrastructures. After that, we'll look at a number of different loss functions, the majority of which are based on a single picture (and maybe with pair and triple of image). After that, armed with all of the knowledge we've gathered, we'll be able to select the best architecture for our requirements[6]. Figure 5. Show the flow of methodologies used in proposed work

6. Methodologies

Module 1: Data Set preparation

• Executable files (benign of malware file) are dissembled using a dissembler

• Assembly docs are parsed, and the data set is collected

Module 2: Using androguard tool.

To decode the AndroidManifest.xml or resources. arsc, use androguard axml and androguard arsc.

• APK or DEX files are analyzed and loaded

• Get a list of all the activities defined in the AndroidManifest.xml

• Update the data set

Module 3: Classification using Machine Leaning Algorithms

• Tree, SVM, K-Fold, KNN, Genetic Algorithm (GA)

• The steps involved in selecting features with GA:

• Step 1: Fill the method with binary-encoded feature subsets.

• Step 2: Begin the technique by creating a randomly generated starting population set.

• Step 3: Based on the supplied fitness function, assign a fitness score to the genetic algorithm.

• Step 4: Parental Selection: High-fitness chromosomes are given precedence over others in the next generation of kids.

• Step 5: To generate offspring, use the specified probability of crossover and mutation to run crossover and mutation operations on the selected parents.

• Iteratively repeat Steps 3–5 until convergence is reached and the best chromosome from the population, i.e. the best feature subset, is acquired.

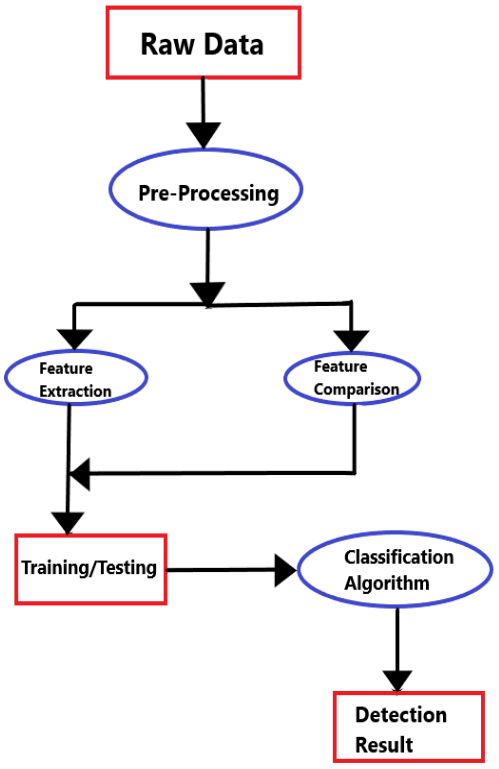

Figure 5. Flow of methodologies used in proposed work

6.1 Input and Output Model

When both the data and the correct answers for each object are provided, supervised learning is applied. There are two stages to supervised learning:

• Fitting a model to available training data and training a model.

• Obtaining predictions by applying the trained model to new samples.

The task:

• We're given a set of things to work with.

• Feature set X is used to represent each object.

• If an object relates to the right answer, it is marked with a Y. X might represent some file content or behaviour attributes, such as file statistics or a list of API calls executed, in the event of malware detection. Malware, innocuous, or a more precise categorization like virus, Trojan-Downloader, or adware might all be labelled Y.

The Android platform has the largest global market share because to its open-source nature and

Google's backing. It has drawn the attention of cyber thieves who exploit it to transmit dangerous malware since it is the most extensively used operating system on the globe. We use an evolutionary genetic strategy to detect Android malware, in which we use selected features to train machine learning classifiers, and then evaluate their ability to detect malware before and after feature selection.

According to the findings, the genetic algorithm produces the most efficient feature subset, reducing the feature dimension to less than half of the original feature set. Machine learning-based classifiers retain a classification accuracy of more than 90-91 percent after feature selection despite operating on a substantially reduced feature dimension, decreasing learning classifier compute complexity.

7. Conclusion:

It is vital to design a framework that can correctly identify malware as the number of threats posed to Android devices rises every day. These threats are transmitted mostly through malicious applications or malware. To produce the most optimised feature subset that may be utilised to train machine learning algorithms in the most efficient manner, the proposed technique employs an evolving Genetic Algorithm. Experiments show that employing, when working with lower dimension feature sets, use Support Vector Machine and Neural Network classifiers, a respectable classification accuracy of more than 90-91 percent may be maintained, decreasing the classifiers' training complexity. More work can be done to improve results by using larger data sets and evaluating the impact of other machine learning algorithms when used in conjunction with the Genetic Algorithm.

References:

[1] Xu Jiang, Baolei Mao, Jun Guan, Xingli Huang, 2020," Android Malware Detection Using Fine-Grained Features", Scientific Programming, vol. 2020, Article ID 5190138.

[2] Tajuddin Manhar Mohammed, Lakshmanan Nataraj, Satish Chikkagoudar, Shivkumar Chandrasekaran, B.S. Manjunath, Jan 2021" Malware Detection Using Frequency Domain-Based Image Visualization and Deep Learning", Hawaii International Conference on System Sciences (HICSS).

[3] Deqiang Li, Qianmu Li, June 2020" Adversarial Deep Ensemble: Evasion Attacks and Defenses for Malware Detection", in IEEE Transactions on Information Forensics and Security.

[4] Amiangshu Bosu, Fang Liu, Danfeng (Daphne) Yao, Gang Wang, April 2017," Collusive Data Leak and More: Large-scale Threat Analysis of Inter-app Communications", in ACM, Pages 71–85.

[5] Fang Liu, Haipeng Cai, Gang Wang, Danfeng Yao, Karim O. Elish, Barbara G. Ryder, December 2017," MR-Droid: A Scalable and Prioritized Analysis of Inter-App Communication Risks", in IEEE Security and Privacy Workshops (SPW).

[6] Muhammad Furqan Rafique, Muhammad Ali, Aqsa Saeed Qureshi, Asifullah Khan, Anwar Majid Mirza, October 2019," Malware Classification using Deep Learning based Feature Extraction and Wrapper based Feature Selection Technique", in arxiv.

[7]. Number of Mobile Phone UsersWorldwide from 2016 to 2023 (In Billions). Available online: https://www.statista.com/statistics/ 330695/number-of-smartphone-users-worldwide/ (accessed on 19 May 2021).

[8]. Mobile Operating System Market Share Worldwide. Available online: https://gs.statcounter.com/os-market-share/mobile/ worldwide/ (accessed on 19 May 2021).

[9]. Number of Android Applications on the Google Play Store. Available online: https://www.appbrain.com/stats/number-ofandroid- apps/ (accessed on 19 May 2021).

[10]. Gibert, D.; Mateu, C.; Planes, J. The rise of machine learning for detection and classification of

malware: Research developments, trends and challenges. J. Netw. Comput. Appl. 2020, 153, 102526. [CrossRef]

[11].PlatformArchitecture.Availableonline:https://developer.android.com/guide/platform(accessed on 19 May 2021).

[12]. Khan, J.; Shahzad, S. Android Architecture and Related Security Risks. Asian J. Technol. Manag. Res. [ISSN: 2249–0892] 2015, 5, 14–18. Available online: http://www.ajtmr.com/papers/Vol5Issue2/Vol5Iss2_P4.pdf(accessed on 19 May 2021).

[13].Cai, H.; Ryder, B.G. Understanding Android application programming and security: A dynamic study. In Proceedings of the 2017 IEEE International Conference on Software Maintenance and Evolution (ICSME), Shanghai, China, 17–22 September 2017; pp. 364–375. [CrossRef]

[14]. Liu, K.; Xu, S.; Xu, G.; Zhang, M.; Sun, D.; Liu, H. A Review of Android Malware Detection Approaches Based on Machine Learning. IEEE Access 2020, 8, 124579–124607. [CrossRef]

[15].Gilski, P.; Stefanski, J. Android os: A review. Tem J. 2015, 4, 116. Available online:https://www.temjournal.com/content/41/14/ temjournal4114.pdf(accessed on 19 May 2021).

[16]. Privacy in Android 11 | Android Developers. Available online: https://developer.android.com/about/versions/11/privacy (accessed on 19 May 2021).

[17]. Chen, T.; Mao, Q.; Yang, Y.; Lv, M.; Zhu, J. TinyDroid: A lightweight and efficient model for Android malware detection and classification. Mob. Inf. Syst. 2018, 2018. [CrossRef]

[18]. Nisa, M.; Shah, J.H.; Kanwal, S.; Raza, M.; Khan, M.A.; Damaševi ̌cius, R.; Blažauskas, T. Hybrid malware classification method using segmentation-based fractal texture analysis and deep convolution neural network features. Appl. Sci. 2020, 10, 4966. [CrossRef]

[19]. Alqahtani, E.J.; Zagrouba, R.; Almuhaideb, A. A Survey on Android Malware Detection Techniques Using Machine Learning Algorithms. In Proceedings of the 2019 Sixth International Conference on Software Defined Systems (SDS), Rome, Italy, 10–13 June 2019; pp. 110–117. [CrossRef]

[20]. Lopes, J.; Serrão, C.; Nunes, L.; Almeida, A.; Oliveira, J. Overview of machine learning methods for Android malware identification. In Proceedings of the 2019 7th International Symposium on Digital Forensics and Security (ISDFS), Barcelos, Portugal, 10–12 June 2019; pp. 1–6. [CrossRef]

[21]. Choudhary, M.; Kishore, B. HAAMD: Hybrid analysis for Android malware detection. In Proceedings of the 2018 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India, 4–6 January 2018; pp. 1–4. [CrossRef]

[22]. Garg, S.; Baliyan, N. Android Security Assessment: A Review, Taxonomy and Research Gap Study. Comput. Secur. 2020, 100, 102087. [CrossRef]

[23]. Garg, S.; Peddoju, S.K.; Sarje, A.K. Network-based detection of Android malicious apps. Int. J. Inf. Secur. 2017, 16, 385–400. [CrossRef]

[24]. Janaka Senanayake 1,* , Harsha Kalutarage 1 and Mhd Omar Al-Kadri, "Android Mobile Malware Detection Using Machine Learning: A Systematic Review", 10, Electronics 2021.

[25]. Xu, K.; Li, Y.; Deng, R.H.; Chen, K. Deeprefiner: Multi-layer android malware detection system applying deep neural networks. In Proceedings of the 2018 IEEE European Symposium on Security and Privacy (EuroS&P), London, UK, 24–26 April 2018; pp. 473–487. [CrossRef]