WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

An Overview of Android Malware and Approaches for Detection using Machine Learning

Omkar Lokhande

Department Of Information Technology B.K Birla College (Autonomous) Kalyan, Mumbai, India.
mailto:omkar.lokhande4140@gmail.com

ABSTRACT

With the exposure of the Android smartphone technologies which has made it affordable for every section of the society. However, the emergence of the Android OS has also escalated the expansion of cybercrime through the mobile platform. Its open source OS has made it a center of attraction for the malware writers. The New generation of malware are upgraded with advanced protection tools like packing, and obfuscation which frustrate anti-virus solutions. Android malware is one among the most dangerous threats on the internet, and it has been on the rise for numerous years. Despite significant efforts in detecting and classifying android malware from innocuous android applications, there's still a long way to go. As a result, there's a need to provide a basic understanding of the behavior displayed by the most common Android malware categories.This paper attempts to provide a reasonable analysis of various Android malwares and detection techniques. The study can help a researchers gain knowledge of the Android security from various aspects and build a more complete, vigorous, and effective solution to the threats that Android is facing.

Keywords: Android Malware, Machine Learning, Cyber Crime, Open SourceOS, Obfuscation.

Introduction

With increased computing power, smartphone are becoming important part of daily life. Android dominates the smartphone and operating system markets, with market shares at all levels. It grew from a minor player when it debuted in 2010, to powering 87 percent of smartphones globally in 2019, and it appears that Android's dominance is up to 64 percent until 2022, and will continue in the coming years, with its smartphone share expected to increase to 87.4 percent in 2023 [1].

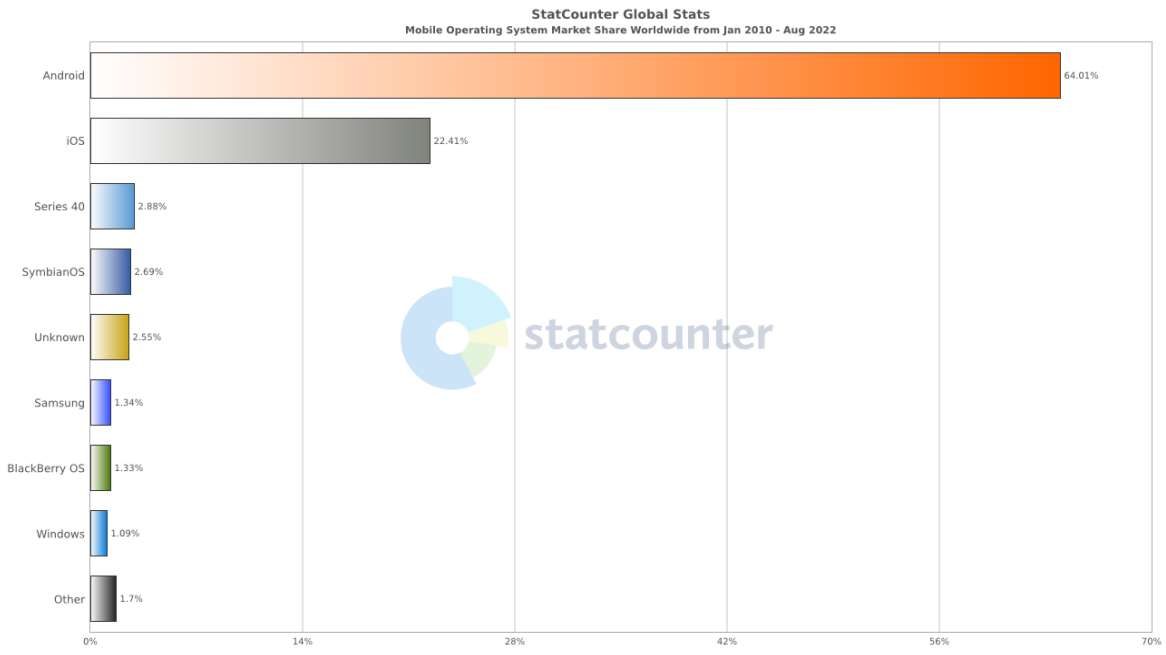

Fig 1. Market share of Android Worldwide 2010-2022

Fig 2 from Statista represents the annual number of malware attacks worldwide,2015-2022. During the first half of 2022, the number of malware attacks worldwide reached 2.8 billion. In 2021, there were 5.4 billion malware attacks detected. In recent years, the highest number of malware attacks was detected in 2018, when there were 10.5 billion such attacks reported across the globe[2].

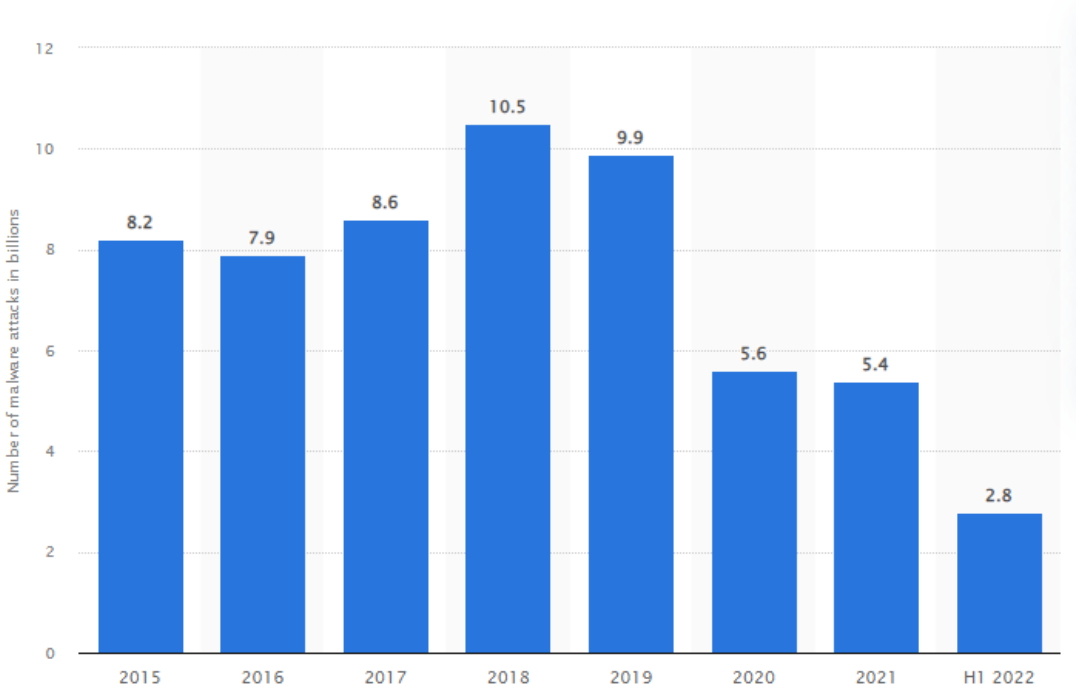

Fig 2. Annual number of malware attacks worldwide from 2015 to first half 2022(in billions)

Android malware is malicious software that is designed to focus on smartphone devices that operateon Android operating system. It behaves similarly to other malware samples that run on desktop or laptops. Android malware is additionally known as mobile malware, and it's any malicious software that designed to harm the mobile device by performing illegal activities[3]. A malware is able to execute malicious behaviours on the userssystem without any authorization or knowledge of its user. A malware performs to stealsensitive information, damage the integrity of the compromised system, and joins the computer into a part of a cybercrime as a boot and so on.

Android applications are easily penetrable with proper knowledge of Android programming if suitable security mechanisms aren't in place[4]. Android marketplaces like Google Play do not follow security protocols when new apk are published. for instance , the android game referred to as Angry Bird was hacked and implanted a malicious code that sent text messages without the knowledge of user. The value was 15 GPB to the user per message, quite a thousand users were affected.

Malware writers gain smartphone control by exploiting platform vulnerabilities , stealing sensitive user information , to extract money by exploiting the information or creating botnet . Thus, it's important to understand their operational activities, working models and usage patterns to plan detection of malwares for mobile devices. Increasing malicious apps has forced the antimalware industry to find out strong and efficient methods suitableforin-use device detection. The usual commercial anti-malware solutions employ signature based detection due to its implementation efficiency and simplicity. Global Internet connectivity and availability of private information such as contacts, messages, social network access, browsing history and banking credentials has attracted the eye of malware developers towards Android. Android malware like SMS Trojans, spyware, botnets, aggressive adware and real escalation attack exploits increased rise aside from being distributed from the secure Google Playstore and well known third-party market places. Malware authors use stealth techniques, code obfuscation methods, repackaging and encryption to bypass the existing protection mechanisms provided by the Android platform and anti-malwaresoftware's.

Machine Learning Process

Machine Learning is a method of analyzing data using software tools and algorithms to develop a model that helps in finding out patterns of regularities in datasets. It teaches machines to find out from previous experiences (data) to predict future events or data instances. Feature vectors are crucial components of machine learning, and that they are often created for the specific task associated with a particular algorithm. Machine learning is predicated on the concept of obtaining the probability data distribution. Machine learning approaches are widely used to classify and identify applications, that are meant for malware identification. The machine model is deployed to simply accept new data and produce an appropriate output. The choice of the machine learning technique depends on what kind of data to predict. The four

primary sorts of machine learning are Supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning. Machine learning in Android Malware detection reduces the time and energy required to manually create and update detection patterns .

Table 1. Classification of common machine learning algorithms based on learning method.

Learning Method	*Machine Learning Techniques*	
Supervised Learning	1.Decision Trees 2.Navie Bayesian 3.Linear Model (1)Linear Regression: Ordinary Least Squares Regression (2)Logistic Regression (3)Linear Discriminate Analysis (LDA) 4. K-Nearest Neighbor(KNN) 5. Support Vector Machine(SVM)	
Unsupervised Learning	1.Clustering Algorithm : K-means. 2. Principal Component Analysis (PCA) 3. Singular Value Decomposition (SVD) 4. Independent Component Analysis (ICA) 5. A-priori Algorithm 6. Expectation-Maximization (EM)	1. Neural Network (NN) and Deep Learning (DL) 2. Hidden Markov Model(HMM) 3. Transfer Learning 4. Ensemble Learning : Bagging, Boosting, Random Forest 5. Online Learning
Semi-supervised Learning	1. Semi-supervised Learning with Nuclear Norm Regularization (SSL-NNR) 2. Graph Inference Learning (GIL) 3. Laplacian SVM	
Reinforcement Learning	1. Q-Learning 2. Deep Q-Leaming Network (DQN) 3. Temporal Difference Learning	

This paper applies machine-learning techniques to spot the Android Malware. As installing third-party applications is the the primary source of security breaches in the Android platform, more advanced methods for static and dynamic analyses have recently been proposed, attempting to include the benefits of traditional methods while improving on their limitations.

Adware , backdoor, file infector, potentially unwanted application (PUA), ransomware , riskware, scareware , and Trojan are variety of the malware categories. Each malware category has distinguishing features that set it apart from the others. Android malware, like humans, evolves. There are Malware families are associated with each malware category. The unrivalled threat of Android malware is that the basis cause of a slew of internet security issues, posing a replacement challenge to researchers and cybersecurity experts. the only way to eliminate this threat is to detect and mitigate malware samples as soon as they detected. The key to accomplishing this is often often fundamental knowledge of Android malware categories and families. the aim of this paper is to shed light on prominent Android malware categories also as related families within each malware category. Furthermore, it familiarizes the reader with the abnormal activities administered by each malware category.Automatic analysis helps the malware analyst generate timely response to detect the unseen malware. Static analysis can quickly and precisely identify malware patterns. However, it fails against code transformations, native code and Java reflections . Thus, dynamic analysis approaches, though time consuming, is an alternate to extract malicious behavior of a stealth malware by executing them in a sandbox environment.

Literature review

This research [5] provides a good overview of Android System-Level app development, privacy issues and guideline for the developers about what measure they ought to consider while developing apps and also discussed the historical development of Android OS and the end-users role to maintain privacy and to minimize security risks.

Dynamically analyzed malware samples to extract API calls then ,used sequential pattern mining algorithm to seek out API call patterns. Three machine learning algorithms are applied on API call patterns feature. Random Forest outperforms the opposite tested algorithms and achieved 99% F-measure. API call pattern-based malware detection approach is usable by many cyber-security tools to mitigate malware threats[6].

The Android platform may be a flexible OS in installing apps, enabling numerous open sources and direct downloads to put in applications in many ways like google play store, torrent files, third-party markets, and others[7]. On one side, attackers and malware conductors are already targeting Android operating platforms with intensive malicious applications. On the opposite side, Android platform developers are pursuing extensive efforts against these malicious apps. Many malware security measures are taken during the installation through the Android system's permission[8].

Five classification algorithms (Random Forest, SVM, K-NN, Nave Bayes, Logistic Regression) and three attribute selection algorithms were examined so as to choose those that would provide the most effective malware detection. Among the classification algorithms, the simplest

proved to be: random forest and k nearest neighbors. They obtained the very best scores on the percentage of correctly classified instances (at the level of 80.3% - 80.7% for Java code)[9].

Comparing Android and iOS helps to know that Android is more vulnerable to security breaches and malware attacks. the main issues are device integrity, application privacy, hardware security, ASCII text file and authentication mechanisms[10].

In big scrambled dataset, paper authors balanced the dataset using the techniques of SMOTE, random undersampling, and hybrid. KNN was ready to detect malware with an accuracy of 98.69% using SMOTE. the share of benign applications that were detected as malware (FPR) was in the range of 2.09% and 4.77%. Therefore, reducing FPR is to be considered in future works. Family detection also can be an effective step in the process of malware detection[11].

We created a dataset for supervisor call instruction traces by using Android malware and benign apps. We performed a statistical analysis of the dataset for 2001 malware and 4580 benign traces in total. We performed the first detection by cutting each system call trace, and only the primary 3000 system calls are used. The MLP classifier performed best with an accuracy of 99.34% and weighted f1-score of 0.9944 for a variety window of 3000 system calls. We also implemented an app that used the model to classify other apps as benign or malware.[12]

MDA approach is that the most anticipated and popular cross-platform approach which has been considered as a trend in global technological advancement. Android provides numerous aspects of providing security. Malware detection systems like static and dynamic analysis, sandboxing a system have proved to be efficient techniques. This paper has briefly described some aspects of the appliance development process and security systems [13].

Latent Semantic Analysis (LSA) and Paragraph Vector (PV) These models extract malicious features and try and classify malware or benign samples. The LSA model is that the most robust feature extraction method [14].

Studied the structure of Android apps, examined the origin of static features, and examined machine learning-based static malware detection for Android malware. The vertical comparison method was used to analyse the algorithm model, fundamental concepts, datasets, and performance metrics of the existing techniques and to highlight their benefits and drawbacks. Machine learning-based static detection methods have advantages in terms of thoroughness, accuracy, and less reliance on expert detection [15].

Alzaylaee et al. have suggested DL-Droid, a deep learning system that detects fraudulent Android apps with dynamic analysis and stateful input generation. More than 31,000 apps total, more than 11,000 of which were malicious. Static and dynamic analysis are both capable of being performed by the automated platform on which DL-Droid runs. The evaluation was conducted using a real Android device, and it was revealed that it took roughly 190 seconds to analyse each programme. When simply taking into account features produced from dynamic analysis, the Random Forest classifier allowed DL-Droid to achieve a detection rate of up to 97.8%, and when incorporating features derived from static analysis as well, the percentage increased to 99.6%[16].

DAN-droid is a mobile malware detection model that classifies apps using deep learning. Opcodes, permissions, and API calls are the three features that DAN-droid makes use of. Their approach was tested using almost 70K obfuscated programmes from the Drebin dataset that were built using five different obfuscation strategies. Using the CNN method, their findings showed an F-score of up to 97.3%[17].

DroidMat, which offers malware detection using manifest and API call tracing, was implemented by Wu et al. [19]. The manifest file and disassembly codes are used by the writers to extract app information. They primarily gather data from the app's manifest file, which is an abstract description of an action to be taken, as well as from Inter-Component Communications (ICC) and API calls pertaining to permissions. In their testing of DroidMat, the authors gathered 238 malicious Android apps and 1,500 good ones. Their findings revealed an up to 97.87% accuracy rate in mobile malware detection.[18]

Malware can evade security mechanisms, collect sensitive user information, display unnecessary advertisements, or can interrupt with the normal functioning of the mobile device. [19]Researchers have been studying the nature of malware applications for many years and have categorizedthem into different families —

- **Trojan** — It is a type of malware, which does not self-replicate. Trojans are typically disguised as genuine software in order to evade detection. These applications seem to be helpful while secretly stealing the user's personal information. Trojan can enter into the system via numerous vectors, such as navigating untrusted websites (drive-by-download), by clicking attachments in phishing emails, other forms of social engineering, etc
- **Ransomware** — It is type of malicious software, which restricts the access to the system resources and extorts money from the victims . Most popular ransomware in mobile platform is crypto-ransomware, which encrypts files of a system thereby restricting the users to access the files. Attackers then demand ransom for the key used to decrypt the files so as to resume the access.
- **Backdoor** — It is a malware that evades the authentication mechanism of the system. As a result, it can remotely access the database and file systems. Backdoor installation requires administrator privileges by rooting Android devices and jailbreaking Apple devices
- **Spyware** —Spyware is classified as a type of malware — malicious software designed to gain access to or damage your computer, often without your knowledge. Spyware gathers your personal information and relays it to advertisers, data firms, or external users. Spyware infection in the mobile device can be viewed differently in Android and iOS. In Android, social engineering attack can used to trick the users to download an app from the Google play store or from a third-party app store. Manin-the-Middle (MitM) attack is used to perform remote infection, where the attacker intercepts the user's mobile communications to inject the malware. In iOS, spyware needs a physical access to the device or through exploitation of JailbreakMEexploit .
- **Adware** — Adware displays unwanted advertisements, which can alter the browser settings, steal personal information or can execute an arbitrary code . Fig. 16 shows the timeline of Android and iOS malware from 2015–2019. The description of these malware is listed in Table 10. Fig. 17 depicts the growth of Android malware worldwide as of May 2019 . It is seen that the total

number of Android malware detections amounted to over 10.5 million programs by May, 2019. There is a steep decline in the growth of malware from 2018 to 2019 due to the deployment of effective machine learning and deep learning malware detection techniques.

- **Backdoors**—It is a malware that evades the authentication mechanism of the system. As a result, it can remotely access the database and file systems These exploit root grant privileges and aim to gain control over the device andperform any operation without the user's knowledge. Backdoor installation requires administrator privileges by rooting Android devices and jailbreaking Apple devices
- **Worms**— This malware creates copies of itself and distributes them over the mobile device'snetworks.
- **Spyware**—Spyware is unwanted software that infiltrates your computing device, stealing your internet usage data and sensitive information . These appear as benign apps designed to monitor the user's confidential information,such as messages, contacts, location, bank information, etc., for undesirable consequences.
- **Botnets**—A botnet is a network of compromised Android devices controlled by a remote server.
- **Ransomware**—This malware prevents users from accessing their data by locking the mobile phoneuntil a ransom amount is paid.
- **Riskwares**—These are legitimate that malicious authors exploit to reduce the device's performanceor harm their data.

Methodology

Methodology to achieve the goal can be either static or dynamic analysis based approaches to detect malware. Control-flow and data-flow analysis are the examples of formal static analysis . In dynamic analysis, apps are executed/emulated in a sandboxed environment, in order to monitor their activities andidentify anomalous behaviors, that are otherwise difficult with static analysis.Depending on the sort of analysis used, anomaly-based detection can be further divided into static, dynamic, and hybrid categories. Static analysis examines the internal organisation of a programme in a non-runtime context. However, dynamic analysis takes the opposite approach and occurs when the app is being used normally.The manifest, Dalvik bytecode, native code, sound, picture, and other inverted APK files are used to extract static features. Run APK files in a controlled environment to gather dynamic features from the log records, code execution pathways, variable value tracking, sensitive function calls, and other behaviours during programme operation..

For virus detection, static and dynamic analysis is the most used technique. In the case of static analysis, the application does not get executed. Binaries are screened throughout this analysing process utilising signatures, or patterns, of harmful applications. Although the primary analysis programme is pattern matching, decompilation and decryption are also a part of static analysis. Static analysis includes reviewing system calls before recognising them. Dynamic analysis entails running the system in a controlled environment and observing its operations. A bug report for malicious programmes is generated if abnormalities are found while monitoring the device state and network activity. The most popular technique for performing dynamic analysis is called "sandboxing," in which the system is separated to avoid using essential functionality by way of system calls.

a. Static Approach

With this method, it is possible to examine an application's functionality and maliciousness without actually running it by disassembling and examining its source code.[20] Even if the detection approach based on static characteristics has several drawbacks compared to those based on dynamic ones, such as the difficulty in overcoming code obfuscation, it also has clear benefits:

I. **Full code coverage**— By scanning code or performing symbolic pseudoexecution, static feature extraction may cover all code and all resource files. The full range of code execution pathways can barely be covered by dynamic feature extraction, though. Numerous programmes demand login information from users in order to access the majority of its features, making it challenging to identify all of the actions which are executed dynamically and leading to incomplete feature extraction.

II. **Reliable detection efficiency**— Since static feature extraction doesn't require the programme to be running, it will do the detection work in the anticipated amount of time. In contrast, dynamic feature extraction necessitates the execution of several functions, which takes a long time. It takes some time for the programme to mimic a click-through interface while it is running. The software could carry out a difficult calculation or go into an endless loop. Due to these circumstances, it is challenging to perform the detection assignment within the allotted time range.

III. **Unperceived by malicious code**—Malicious apps are unable to sense that they are being checked since static detection does not need the execution of the code. Even while some malware attempts to complicate static analysis by adding interference codes, these extra codes may serve as an indicator to help find maliciousssoftware.

IV. **Easier to generate generic fingerprint identification**— Static malicious sample analysis is more inclined to extract features with invariance and universality. In contrast, the operational environment is quite likely to have an impact on dynamic analysis. The quick identification of widespread harmful apps may be accomplished using statically extracted traits, which are suited for fingerprinting.

b. Dynamic Approach

In this approach, the application is examined during execution and can help identify undetected malware by static analysis techniques due to code obfuscation and encryption of the malware.

For Android applications, dynamic analysis has many advantages over static analysis[21]:

I. **Behavioral Runtime Features**—The runtime behavioural aspects that are gathered when executing Android apps in actual contexts or in emulation settings, such a sandbox, are known as dynamic features, and the accompanying technique of analysis is

known as dynamic analysis. System calls, API calls, network traffic, and CPU data are all examples of the dynamic analysis items specifically for Android applications. We give a summary of the dynamic characteristics employed in Android malware detection based on machine learning in accordance with our literature study.

II. **API requests and network activity**— For the purpose of detecting Android malware, many research integrate graph theory with Dalvik opcode. These examples, however, are not largely based on machine learning techniques and fall outside the purview of this work. API calls and network trafficthis event highlights the intimate relationship between these dynamic elements and the presence of malware in Android applications. Despite the fact that malware detection is based on a single class of dynamic characteristic, several researchers have various worries or ways of looking at the data. Analyze the programme based on the sequence of system calls made while it is operating, using the feature of system calls as an example. Analyze the programme instead based on how frequently certain system calls occur.

III. **Runtime Environment**—Many objects or events, such as lifecycle callbacks, GUI handling, control flow, and data flow at runtime, cannot be studied by static analysis alone due to the event-driven structure of the Android system and must rely on the runtime environment. Some permissions or APIs that are stated in the Android application code, such as in AndroidManifest.xml, may not actually be used or used.

IV. **Dynamic Permission Support**— Dynamic permission support has been added to the Android system since version 6.0 (API 23), therefore detection based only on static analysis may be prone to false positives. Through static analysis, it might be challenging to identify malicious activity in Android applications due to techniques like code obfuscation and dynamic code loading, however dynamic analysis can partially circumvent these restrictions. As was already noted, dynamic analysis has numerous benefits, but it also has the drawback of taking up more time and resources than static analysis.

Permission. Android permission provides fine-grained security features to enhance the restriction of the specific operations in applicationFor example, android.permission.READ_SMS allows an application to read SMS messages, and android.permission.INTERNET allows an application to open network sockets. Each permission has a ‘‘protection level’’ attribute to define its security level.API call (API). In Android, critical information or sensitive data can be accessed by certain API calls, such as getAccount to get access to user accounts and updateNetwork to update system network time. That is, if we trace or locate those API, it is possible to supervise the key behaviors of malware.

Fig 3. Mentions commonly used Techniques for Malware Detection that can help to achieve the estimated result. RF has proven to be more efficient and fast amongst other algorithms .Random Forest is a supervised learning algorithm. The forest which is built in the algorithm is a group of Decision Trees which are trained with the bagging method. Bagging is similar to polling. It compares results produced by all the trees in forest and the label which has been produced by more trees will be the outcome and that label is assigned to given entity.

Algorithm	Advantages	Disadvantages
DT	• Possible handle samples with missing values • Easy to understand	• Might cause the overfitting problem
NB	• Easily and quickly trainable	• Need to calculate prior probability • Not applicable if the feature variables are corelated
Regression Models	• Widely used in statistics based studies • Direct and Fast	• Not possible to deal well with high dimensional features
KNN	• Suitable to solve multiclassification problems	• Computation overhead is relatively high • Issues with the skewness of data
SVM	• Possible to solve high dimensional nonlinear small scale problems	• High overhead in data processing • Might face some issues when there are missing values in the sample
K-Means	• Easy to implement • Fast and simple	• Sensitive to outliers
RF	• Reduces overfitting • Normalising of data is not required	• Requires much time to train • Requires high computational power
Neural Networks	• Highly accurate • Strong fault tolerance	• Requires much time to train • Require a large number of data to train the model
LSTM	• Capable to remember facts for lengthy interval	• Requires high computational resources
CNN	• Reduce unimportant parameters by weight sharing and downsampling	• High computational cost
Ensemble Learning	• Accuracy is high	• Overhead on model training and maintenance

Fig 3. Commonly used ML algorithms for Android Malware Detection

Conclusion

Android malware is one of the most dangerous threats on the internet, and its prevalence has increased dramatically in recent years. Experts in cybersecurity face an open problem. There are a variety of machine learning-based approaches for detecting and classifying Android malware.We briefly introduced the background to Android malware and gave a comprehensive review of machine learning-based

approaches for detecting Android malware.The review provided better knowledge of the status with respect to android malware detection like the common methods used, the malware analysis techniques, various features used for malware analysis, algorithms. In future a researcher may use the algorithms mentioned in this paper for discovering malwares and their types.

Acknowledgements

A special gratitude is conveyed to Prof. Swapna Augustine Nikale, Department of Information Technology of B.K. Birla College of Arts, Science and Commerce (Autonomous) Kalyan, Thane.

References

[1] "Mobile Operating System Market Share Worldwide," *StatCounter Global Stats*. https://gs.statcounter.com/os-market-share/mobile/worldwide/ (accessed Sep. 15, 2022).

[2] "Number of malware attacks per year 2022," *Statista*. https://www.statista.com/statistics/873097/malware-attacks-per-year-worldwide/ (accessed Sep. 15, 2022).

[3] A. H. E. Fiky, A. E. Shenawy, and M. A. Madkour, "Android Malware Category and Family Detection and Identification using Machine Learning," p. 20.

[4] J. Senanayake, H. Kalutarage, and M. O. Al-Kadri, "Android Mobile Malware Detection Using Machine Learning: A Systematic Review," *Electronics*, vol. 10, no. 13, p. 1606, Jul. 2021, doi: 10.3390/electronics10131606.

[5] R. Sikder, M. S. Khan, M. S. Hossain, and W. Z. Khan, "A survey on android security: development and deployment hindrance and best practices," *TELKOMNIKA Telecommun. Comput. Electron. Control*, vol. 18, no. 1, p. 485, Feb. 2020, doi: 10.12928/telkomnika.v18i1.13288.

[6] A. Pektas, E. N. Pektas, and T. Acarman, "Mining Patterns of Sequential Malicious APIs to Detect Malware," *Int. J. Netw. Secur. Its Appl.*, vol. 10, no. 4, pp. 01–09, Jul. 2018, doi: 10.5121/ijnsa.2018.10401.

[7] W. Azim, "Android as open source operating system," p. 9.

[8] A. S. Shatnawi, Q. Yassen, and A. Yateem, "An Android Malware Detection Approach Based on Static Feature Analysis Using Machine Learning Algorithms," *Procedia Comput. Sci.*, vol. 201, pp. 653–658, 2022, doi: 10.1016/j.procs.2022.03.086.

[9] M. Kedziora, P. Gawin, M. Szczepanik, and I. Jozwiak, "Malware Detection Using Machine Learning Algorithms and Reverse Engineering of Android Java Code," *Int. J. Netw. Secur. Its Appl.*, vol. 11, no. 01, pp. 01–14, Jan. 2019, doi: 10.5121/ijnsa.2019.11101.

[10] S. Garg and N. Baliyan, "Comparative analysis of Android and iOS from security viewpoint," *Comput. Sci. Rev.*, vol. 40, p. 100372, May 2021, doi: 10.1016/j.cosrev.2021.100372.

[11] D. T. Dehkordy and A. Rasoolzadegan, "A new machine learning-based method for android malware detection on imbalanced dataset," *Multimed. Tools Appl.*, vol. 80, no. 16, pp. 24533–24554, Jul. 2021, doi: 10.1007/s11042-021-10647-z.

[12] X. Zhang *et al.*, "An Early Detection of Android Malware Using System Calls based Machine Learning Model," in *Proceedings of the 17th International Conference on Availability, Reliability and Security*, Vienna Austria, Aug. 2022, pp. 1–9. doi: 10.1145/3538969.3544413.

[13] A. Sarkar, A. Goyal, D. Hicks, D. Sarkar, and S. Hazra, "Android Application Development: A Brief Overview of Android Platforms and Evolution of Security Systems," in *2019 Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, Palladam, India, Dec. 2019, pp. 73–79. doi: 10.1109/I-SMAC47947.2019.9032440.

[14] M. Mimura and R. Yamamoto, "A Feasibility Study on Evasion Attacks Against Nlp-Based Malware Detection Algorithms," *SSRN Electron. J.*, 2022, doi: 10.2139/ssrn.4061102.

[15] Q. Wu, X. Zhu, and B. Liu, "A Survey of Android Malware Static Detection Technology Based on Machine Learning," *Mob. Inf. Syst.*, vol. 2021, pp. 1–18, Mar. 2021, doi: 10.1155/2021/8896013.

[16] "DL-Droid: Deep learning based android malware detection using real devices," *Comput. Secur.*, vol. 89, p. 101663, Feb. 2020, doi: 10.1016/j.cose.2019.101663.

[17] "DANdroid | Proceedings of the Tenth ACM Conference on Data and Application Security and Privacy," *ACM Conferences*. https://dl.acm.org/doi/10.1145/3374664.3375746 (accessed Sep. 15, 2022).

[18] "DroidMat: Android Malware Detection through Manifest and API Calls Tracing." https://ieeexplore.ieee.org/document/6298136/ (accessed Sep. 15, 2022).

[19] P. Faruki*et al.*, "Android Security: A Survey of Issues, Malware Penetration, and Defenses," *IEEE Commun. Surv. Tutor.*, vol. 17, no. 2, pp. 998–1022, 2015, doi: 10.1109/COMST.2014.2386139.

[20] V. Kouliaridis, K. Barmpatsalou, G. Kambourakis, and S. Chen, "A Survey on Mobile Malware Detection Techniques," *IEICE Trans. Inf. Syst.*, vol. E103.D, no. 2, pp. 204–211, Feb. 2020, doi: 10.1587/transinf.2019INI0003.

[21] K. Liu, S. Xu, G. Xu, M. Zhang, D. Sun, and H. Liu, "A Review of Android Malware Detection Approaches Based on Machine Learning," *IEEE Access*, vol. 8, pp. 124579–124607, 2020, doi: 10.1109/ACCESS.2020.3006143.