JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

ANDROID MALWARE ANALYSIS – A LITERATURE REVIEW

Anu Varghese.[1,2], Jagadeesha S.N.[3]

[1] Research Scholar, College of Computer Science & Information Science, Srinivas University, Mangalore, India.

[2]Assistant Professor, Department of Computer Science, MES M K Mackar Pillay College for Advanced Studies, Aluva, (Mahatma Gandhi University), Aluva, Kerala, India.

[3]Research Professor, College of Computer Science and Information Science, Srinivas University, Mangalore, Karnataka, India.

ABSTRACT

Background/Purpose: *The ever-increasing presence of malicious software designed to target Android devices represents a huge risk to the security of mobile devices. Researchers are investigating a variety of cutting-edge approaches, procedures, and strategies to analyse and identify it. The purpose of this literature review is to examine recent research on Android malware analysis, with a particular emphasis on novel methodologies and the degree to which they are successful in identifying and mitigating the threat. This paper reviews the three common approaches and discuss the challenges and limitations identified.*

Objective: *This literature review aims to provide a comprehensive overview of Android malware analysis techniques and methodologies, evaluating the effectiveness of different approaches like static, dynamic, machine learning and deep learning. It also evaluates existing tools and frameworks, highlights recent studies' contributions and highlights gaps in research. The review aims to enhance detection mitigation strategies for mobile security and provide insights for researchers, practitioners, and policymakers.*

Design/Methodology/Approach*: The SWOT analysis method is used to conduct data analysis and present the results from a variety of sources, including academic papers, web articles, journals, and other sources.*

Findings/Result*: The literature study focuses on several different strategies and methods for analysing Android malware, such as static, dynamic, machine learning, and deep learning. These techniques are used to extract features, analyse code structure, and identify dangerous behaviours. It is essential for effective detection solutions to incorporate a variety of approaches as well as extensive datasets. Most of the research has been directed towards improving machine learning models rather than the malware analysis process.*

Paper type: literature review

Keywords: android malware, malware detection, HinDroid, static analysis, API, cyber security

INTRODUCTION

Computers, servers, mobile devices, electronic systems, networks, and data can all be protected from malicious attacks with the use of a practise known as cybersecurity. It requires the use of many layers of protection, which include people, procedures, and technology. It includes things like the protection of computer networks and applications, as well as information and operational security, as well as education for end users, disaster recovery, and business continuity. Cyberattacks can be thwarted by enterprises and individuals alike thanks to technological advancements such as next-

generation firewalls, DNS filtering, malware protection, antivirus software, and email security solutions. Advanced Cyber Defence programmes are advantageous for everyone in today's connected society since they avoid unwanted repercussions such as the theft of personal information, extortion, and the deletion of critical material. These are just some examples.

Researchers of cyberthreats, such as the team of 250 at Talos, investigate new and existing threats and methods of cyber-attack, thereby enhancing open-source software and boosting public awareness of the issue. The outcomes of their efforts make the internet a more trustworthy place for users of all stripes. [1]. The Fig.1 shows the statistics of no: of detected malicious installation packages on mobile devices from 2015 to 2022.

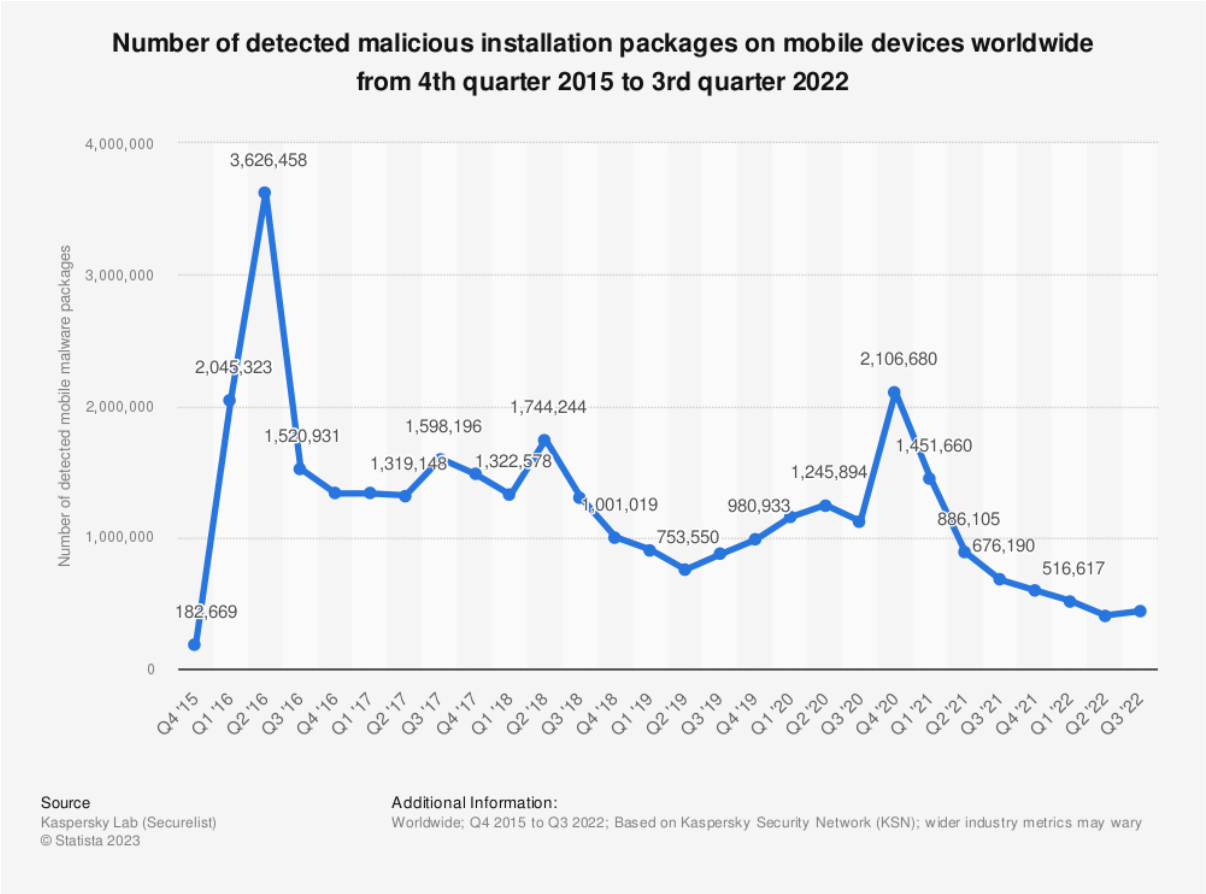

Fig.1 No.of detected malicious installation packages on mobile devices worldwide from 4th quarter 2015 to 3rd quarter 2022 (www.statistica.com)

A survey on various threats and current state of security in android platform found four forms of Android attacks: hardware, kernel, HAL, and application. Hardware-based attacks such as Rowhammer, Glitch, and Drammer are related to sensors, touch screens, communication media, and DRAM. Kernel-based attacks such as Gooligan, DroidKungfu, Return-oriented Programming are related to Root Privilege, Memory, Boot Loader, and Device Driver. HAL-based attacks such as Return to User and TocTou are related to interfaces for cameras, Bluetooth, Wi-Fi, Global Positioning System (GPS), and Radio. Application-based attacks such as AdDetect, WuKong, and LibSift are related to third-party libraries, Intra-Library collusion, and privilege escalations.

Static analysis and dynamic analysis are the primary methodologies utilised by defence systems; nevertheless, these approaches are restricted in their ability to detect new types and varieties of malware. In computer vision, natural language processing, and speech recognition, deep learning technology has demonstrated some encouraging outcomes due to its intelligence and adaptability. [2]

OBJECTIVES

- To understand the process involved in malware analysis.
- To analyse the various malware analysis techniques
- To understand current research on malware analysis and to analyse the various threats and challenges in the field.

LITERATURE REVIEW/RELATED WORKS

Slno	Features used	Model	Accuracy	Reference
1	API	KNN	99%	Aafer et al.,[3]
2	API & Permissions	RF, ANN	94%	Qiao et al.,[4]
3	API & Permissions	RF	92.3%	Chan & Wen Kai Song[5]
4`	API & Permissions	DBN	93%	Wang et al.,[6]
5	API & Permissions	SVM	99.6%	Singh et al.,[7]
6	API & Permissions	SVM	86%	Li et al.,[8]
7	API, Intents	NB	98%	Kumar et al.,[9]
8	Permissions, Intents	RF	98%	Koli[10]
9	API calls	RF	94%	Onwuzurike et al.,[11]
10	Permissions, API calls, native calls, opcode	DT	97.7%	Cai et al.,[12]

LITERATURE SURVEY

Malware assaults target Android devices, which account for 70% off mobile phone users. Classic signature-based detection methods failed with huge numbers of users and applications, but machine learning can detect zero-day assaults. Machine learning to identify Android malware has been studied using supervised, unsupervised, deep learning, and online methods. Android device growth has spurred the development of ML- based malware detection methods that may avoid zero-day assaults. However, obsolete data sets and inadequate metrics limit present techniques. Reimplementation and re-evaluation of existing methodologies using an independent, up to date data set extending online learning approaches to incorporate dynamic and hybrid features and ongoing Android dynamic analytic tool development are recommendations to overcome these concerns. Automation for updating datasets add enhanced reporting are also needed. The rise of this discipline is encouraging, and new machine methods offer great space for additional research [13]. Malware designed to infect Android devices has evolved into a substantial problem, endangering not just the functionality of devices but also the privacy and safety of their users. The threat landscape is always shifting; thus, researchers have been concentrating on building efficient tools for analysing malware for Android devices. The purpose of this literature review is to investigate recent research that has contributed to the field of Android malware analysis, with a particular emphasis on the application of machine learning strategies.

Guerra-Manzanares et al. [14] addresses time, dynamic data sources, and real device and emulator characteristics to improve Android malware detection systems. By integrating benign add malicious data, a broader time frame and 489 static and dynamic features are obtained. The largest hybrid featured Android data set, KronoDroid, offers timestamps for each data sample from 2008 through 2020. Time stamped sample from over 209 Android malware types make it the largest and only data set. With KronoDroid , the authors will study concept drift ,dynamic differences between emulators and real devices and malware family evolution.

Shatnawi et al.[15]proposes a static base classification strategy for malware detection utilising SVM, K nearest neighbours and Naïve Bayes. The strategy seeks strong malware detection rates and mobile information access development protection. The SVM classifier has the highest accuracy, averaging 94% using permission features and 83% using API call features. This strategy aids mobile malware prevention and reduction.

A framework for the identification of malicious software on Android devices dubbed DeepAndroid, which is based on deep learning, is proposed in Jiang et al. [16]. To extract information from Android applications and obtain a high level of detection accuracy, the authors make use of convolutional neural networks (CNNs). The results of the experiments show that DeepAndroid is effective in detecting both known and undiscovered examples of malicious software.

Using Random Forest and XGBoost, Ismail et al.,[17] uses machine learning to sort and guess Distributed Denial of Service (DDoS) attacks. We utilised the UNWS-np-15 dataset and Python as an emulator. Compared to previous studies, the model was about 85% to 79% more accurate. In the first group, the model was 89% accurate, and in the second group, it was 90% accurate on average. This all-around system makes it easier to guess when DDoS attacks will happen

Utilising a learning-based classifier, Chen et al.,[18] investigates the safety of machine learning in finding malware on Android. Asses the classifier's defences against escape attacks and suggests a strong secure-learning approach. The model enhances system defences against different evasion attacks and can be used for other security jobs, such as spam blocking and fraud detection. In this paper, the rising interest in cybersecurity to protect Android users from malware is shown.

Zhai et al.,[19] aims to make use of deep learning strategies to identify potentially harmful intentions in Android applications. DeepIntent is a model that the authors suggest; it makes use of networks with bidirectional long short-term memory (BiLSTM) to capture the sequential patterns of intentions. The findings of this experimental work reveal that DeepIntent is effective in precisely identifying harmful intent patterns.

Shabir & Sabahat,[20] is a hybrid approach to feature selection is presented by the authors as a method for Android malware detection. They use a combination of information acquisition and evolutionary algorithms to pick features, in addition to combining static and dynamic characteristics, such as permissions, system calls, and API requests. The suggested method delivers increased detection performance by picking the features that provide the most informative information.

Wang et al.,[21] presents DroidEnsemble, an ensemble learning-based method for detecting malicious software on Android devices. The authors use a combination of different machine learning classifiers, such as random forests, support vector machines, and gradient boosting, to increase the accuracy of the detection process as a whole. Experiment findings reveal that DroidEnsemble works better than individual classifiers when it comes to detecting malicious software on Android.

Feizollah et al.,[22] proposes A framework for analysing Android malware families that is based on deep learning is called AndroDialysis, and it is presented in this study. In order to categorise malware samples according to their respective families, the authors make use of several different types of deep neural networks, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs). The findings suggest that AndroDialysis is successful in precisely recognising malware types and comprehending the behavioural characteristics of malicious programmes.

Haq et al.,[23] presents a hybrid approach to Android malware detection that combines static and dynamic analysis approaches. The authors of this research offer MalDroid as a solution. During the static analysis phase, features such as permissions, API calls, and manifest entries are extracted. During the dynamic analysis phase, runtime behaviour is observed. The authors make use of machine learning algorithms like random forests and logistic regression to achieve high detection rates while simultaneously reducing the number of false positives.

Zhu et al.,[24] presented a stacking ensemble called SEDMDroid for identifying sophisticated Android malware. The framework employed bootstrap and Principal Component Analysis (PCA) to generate random feature subspaces. Multi-Layer Perception (MLP) and Support Vector Machine (SVM) were trained to learn additional information. The experiment was performed on official benign Android applications and malicious apps from the virus share repository.

Bibi, I., et al.,[25] proposed a dynamic deep learning-based architecture to identify Android malware using Gated Recurrent Unit (GRU). The scheme achieved 98.99% detection accuracy. Millar et al. implemented DAN for identifying malicious Android applications, with 97% average detection accuracy. Amin et al.,[26] presented a VM-based open-source mobile anti-malware system, with an accuracy of 99%.

Xiao et al.,[27] introduced a malware identification mechanism using systems calls while using LSTM, with a detection rate of 97.7%. Iram et al. proposed an effective DL-based ransomware detection scheme using Long Short-Term Memory (LSTM). The study investigated permission-induced risks in android applications using Gated Recurrent Unit and Convolutional Neural Network.

An ensemble learning strategy for identifying malicious software for Android is proposed by the authors of Li et al.,[28] To determine which characteristics are the most important, they use a variety of feature selection methods, such as information gain and chi-square. An ensemble model is produced by the authors by combining several different classifiers, including random forests and AdaBoost. The experimental findings show that the overall detection accuracy is better compared to the separate classifiers.

Zhang et al.,[29] investigates the application of deep learning techniques, more specifically deep belief networks (DBNs), for the detection of malware on Android. The authors take information gleaned from opcode sequences and use them to train a DBN model that can differentiate between malicious and benign applications. The results illustrate the usefulness of deep learning as a tool for analysing malicious Android software and show promise in terms of detection accuracy.

Suarez-Tangil et al.,[30] concentrates on the examination of manifest files for Android applications with the goal to identify malicious software. The authors suggest using DroidMat, a programme that does an analysis of the permissions, intents, activities, and services that are defined in the manifest files. DroidMat delivers accurate malware detection by identifying potentially malicious patterns and then comparing these to prevalent malware behaviours.

Singh et al.,[31] presents a hybrid strategy for detecting malware on Android devices by combining machine learning and static analysis strategies. When classifying malware samples, the authors make use of a variety of machine learning algorithms, such as support vector machines and k-nearest neighbours, as well as static features like permissions and API

calls. The hybrid strategy is stronger than the separate techniques in terms of its ability to achieve increased detection accuracy.

Arzt et al.,[32] enables exact taint analysis for Android applications. It does static analysis to follow the flow of sensitive information and identify any data leaks or malicious behaviour. In order to assist in the detection of malicious software, FlowDroid does an analysis of the dynamic that exists between the various application components. It then offers insights into the behaviour of Android apps.

Hou et al.,[33] suggests graph analysis as a method for detecting malicious software on Android devices. The idea of a HinDroid graph, which illustrates the relationships and dependencies between the various parts of Android applications, is presented in this study for the first time. Through the use of both static and dynamic analysis to generate the graph, HinDroid is able to discover malicious behaviour by capturing the structural properties and patterns of the system. The research reveals that HinDroid is effective in precisely detecting malicious software for Android and separating it from applications that are legitimately available.

Gao et al.,[34] introduces GDroid is a tool that detect malicious software on Android devices via a dynamic analysis. The research presents a dynamic analysis tool that, when applied to Android applications, allows them to be run in a predetermined setting while the runtime behaviour of those apps is observed. GDroid is able to detect suspected malicious software activity and anomalies since it monitors system calls, network communication, and file access. The study demonstrates how good GDroid is at identifying dangerous behaviour in real-time, which contributes to a deeper comprehension of Android malware.

Fan et al., [35] provides a novel method that analyses and detects Android malware by utilising graph embeddings and unsupervised learning. The results of the experiments show that it is superior to existing methods and highlight its potential for improving malware detection and family analysis in the Android ecosystem.

Singh et al.,[36] establish a method for detecting malware on Android devices by utilising static properties such as normal permissions, nonstandard permissions, and API-calls. The key features were chosen with the help of FSTs. According to the findings of the study, integrated features are superior to individual features when it comes to detecting malicious software on Android. Using the BI-Normal Separation FST and L-SVM classifier allowed for the achievement of the best possible detection accuracy, which was 99.6%.

Singh & Singh,[37] investigate seven different machine learning classifiers that detect malware by making API calls. The research shows that ensemble algorithms have the highest accuracy due to the fact that they are optimised versions of conventional ML methods. The choice of parameters is determined by the type of dataset and its distribution. The balance that needs to be struck between parameters such as the n-estimator and the learning rate in ensemble methods and the regularisation parameter value and the margin between distinct classes is discussed. The research analyses modified machine learning algorithms by testing them on 6434 benign and 8634 malicious samples. Random forest demonstrates the greatest accuracy of 99.1% in binary classification.

Zhu et al.,[38] presents a classification model that improves feature extraction and classification capabilities by making use of several convolutional neural networks. A backtracking method that can provide high-fidelity explanations of deep learning detection algorithms is also shown here. The transparent, multimodal CNN-based Android malware detection framework has been built, with the goal of delivering large time cost improvements while yet keeping superior classification performance.

Ye et al. [39]proposes to prevent ever-evolving Android malware threats by extracting API call sequences from runtime Android applications and conducting semantic research. They suggest the HG-Learning approach for effective categorization and introduce HG as a way for modelling complex interactions among entities of multiple types. AiDroid is a DNN classifier that was developed specifically for Android malware detection in real time, and it outperforms other systems. The solution has been incorporated into Tencent Mobile Security, which will safeguard millions of users throughout the globe.

Hou et al., [40](Make Evasion Harder: An Intelligent Android Malware Detection System)This study examines Android malware assaults by examining the relationships that exist between apps and application programming interfaces (APIs) through the utilisation of a structured heterogeneous information network (HIN). The meta-path-based approach is used to characterise semantic relatedness, and HinDroid outperforms existing Android malware detection systems. The findings from the experiments show some encouraging results

Xiao et al.,[41]In this paper, a behavior-based deep learning malware detection system for IoT contexts is presented. The framework combines behaviours with Stack AutoEncoder for maximum performance. There is a potential for SAE-based models to be applied in classification, as they increase detection accuracy by 1.5%.(Malware Detection Based on Deep Learning of Behavior Graphs)

The significance of dataset construction and evaluation approaches is emphasised in the review paper. To properly train and assess malware detection models, Garg & Yadav[42]stress the importance of using vast and diverse datasets. Accuracy, precision, recall, and F1-score are only some of the evaluation metrics they cover.

Sample Android apps were classified as benign or malicious using a combination of permissions and API calls, as well as machine learning algorithms, as demonstrated by Peiravian and Zhu [43]. They may train a classifier to determine if an app is safe or dangerous based on the permissions and API calls it makes. In order to verify their method, they conducted experiments on real-world apps using 1,260 harmful samples and 1,250 benign samples. Tests used 10-fold cross validation using Support Vector Machines (SVM), Decision Tree (DT) of J48, and Bagging to compare these three categorization strategies. WEKA's J48 is a decision tree method that uses the C4.5 algorithm. They used 1,456 features (130 permissions + 1,326 APIs) to reach a 96.88% accuracy rate, which is rather good.

Aafer et al.,[44] retrieved API-level malware behaviour features to distinguish dangerous and benign programmes. APIs and frequency analysis identified malicious patterns. 71% of benign programmes had advertising packages, and certain APIs were identical in malicious instances. The K-Nearest Neighbour classifier had 99% accuracy and 2.2% FPR after data flow analysis.

Goyal et al.,[45] created SafeDroid, an open-source Android malware detection service. Based on API calls, the micro-service classifies apps as good or bad. 743 dangerous APIs and 300 top-ranked features were found by SafeDroid. With 99.51% accuracy and 0.017 FPR, Random Forest performed best.

Jung et al. [46] provides an efficient machine learning-based Android malware detection approach. It ranks APIs from 30,159 benign and 30,084 harmful apps. The classifier uses the top 50 benign and harmful APIs. The Random Forest classifier detects the top 50 API list better than the malicious list. The paper will compare SVM and ANN malware detection methods.

In Kim et al. [47] ,Convolution neural networks (CNN) are used by MAPAS to analyse API call graphs of malicious apps. CNN is exclusively used by MAPAS to find common malware API call graph features. MAPAS is a lightweight classifier

to detect malware by comparing API call graphs used for harmful activities to those of apps to be classed. MAPAS's efficacy and efficiency are demonstrated by implementing and evaluating a prototype. MaMaDroid, a cutting-edge Android malware detection method, is compared to MAPAS. Our test shows that MAPAS classifies applications 145.8% faster and consumes 10 times less RAM than MaMaDroid. MAPAS outperforms MaMaDroid in detecting unknown malware (91.27% vs. 84.91%). Additionally, MAPAS can accurately detect any malware.

Zhang et al.,[48] presents a method-level behavioural semantic analysis Android malware detection system. Static analysis extracts method-level information from Android apps to create a behavioural model of normal app behaviour. It detects viruses using similarity-based matching.

Sequence-based analysis and natural language processing are used to detect Android malware in Zhang et al., [49]. It analyses Android API requests using an n-gram model and NLP methods like word embeddings. Using modified sequences, the CNN model classifies apps as harmful or benign. The hybrid approach detects malware more accurately than separate methods.

Naval et al.,[50] suggested an updated API sequence model. A series of trials showed that the method was effective when compared to existing malicious code detectors. Using function call graphs of apps as social networks, Wu et al.,[51] suggests a simple graph-based method for finding malware on Android. A total of 15,285 good samples and 15,430 bad samples are used to test the system called MalScan. There are results that show it can find Android malware faster than two state-of-the-art methods, with up to 98% success in just one second. Through finding 18 examples of zero-day malware in a Google Play app market, MalScan also showed that it could be used to scan entire markets for malware.

Apposcopy[52] is a semantics-based approach for identifying Android malware that steals user information. It uses a high-level language for specifying signatures and a static analysis to determine if an application matches a malware signature. The algorithm uses static taint analysis and Inter-Component Call Graph to efficiently detect Android applications with specific control and data-flow properties. Apposcopy can detect malware with high accuracy and is resilient to various program obfuscations. Future work includes improving efficiency and precision, de-obfuscating apps, and learning malware signatures from labelled apps.

Feng et al. [53] provides a minimal-sample method for learning Android semantic malware signatures. It finds maximally suspicious common subgraphs (MSCS) shared by all malware families. The method uses static analysis and a new approximate signature matching algorithm to match Android apps. The ASTROID-implemented method improves malware detection accuracy, precision, interpretability, and resistance to behavioural obfuscation.

In the article Zou et al.,[54] detects Android malware using social-network-analysis and graph-based approaches. Social-network-based centrality analysis identifies important nodes in function call graphs by treating them as complicated social networks. IntDroid tests 3,988 benign and 4,265 malicious samples. It identifies Android malware with a 97.1% F-measure and 99.1% True-positive Rate. IntDroid can find 28 GooglePlay zero-day viruses faster than MaMaDroid.

Attackers target Android OS-based mobile devices because of their convenience and features. For real-world apps, researchers are creating Android malware analyzer frameworks. Permissions, intentions, and API requests are covered in Taheri et al.,[55]The two-layer Android malware analyzer has 95.3% static-based binary classification, 83.3% dynamic-based category classification, and 59.7% dynamic-based family classification

A graph convolutional network malware classifier was developed to adapt to different characteristics by Li et al.,[56]. The method extracts API call sequences, generates directed cycle graphs, extracts feature maps, and designs a classifier. The results show superior performance in detection and accuracy, with a 98.32% FPR and stability.

Amer et al.,[57] introduces multi-perspective malware detection methods utilising statistical, contextual, and graph mining. It performs well with shifting API calling sequences. The models cluster API calls and visualise malware and goodware. These models are 0.997 and 0.977 accurate against Windows and Android malware, respectively. A algorithm for detecting bogus goodware sequences is proposed to adapt to new malware threats.

Feng et al., [58] suggests a new way to find malware on Android devices that uses lightweight static analysis and the graph neural network (GNN). Instead of getting information about API calls, the writers look at the source code of Android apps to get high-level semantic data. As a result of function invocation relationships, they make rough call graphs and pull out characteristics within functions. A graph neural network creates a vector representation of the programme. Malware detection is then done on this matrix. According to the results of experiments, this method works better than the best current identification methods. Malware attacks are becoming easier to do on Android because it is becoming more popular. Based on static analysis of the Android APK, this study entitled Graph Approach for Android Malware Detection Using Machine Learning Techniques,[59] shows how to use machine learning to find malware. The technique makes use of the Drebin and Malgenome files, which have a lot of malware and goodware in them. It did a great job of classifying data: 98.19% of the time with the Drebin dataset, 96.27% with the RF dataset, and 98.84% with the Malgenome dataset. This method works better than the best recognition methods currently available.

Because Android malware is getting smarter and more complex, machine learning methods are a better way to find it. Existing feature representations for system call analysis methods, on the other hand, have a lot of dimensions and don't show how one thing depends on another. Surendran et al., [60] suggests a new way to find Android malware apps using low-dimensional features that are represented and extracted using graph signals. A smart expert system based on machine learning uses this low-dimensional feature to make the detection job automatic. Using random forest classifiers, experiments show that a feature vector with 16 dimensions is enough to classify malware with a maximum accuracy of 0.99.

CYBERSECURITY

Computers, servers, mobile devices, electronic systems, networks, and data can all be protected from malicious attacks with the use of a practise known as cybersecurity. It requires the use of many layers of protection, which include people, procedures, and technology. Computer network security, application security, information security, operational security, business continuity and disaster recovery, and end-user education are among the most important aspects of operational security. These precautions make it easier for companies to deal with breaches in their cyber security and the loss of data. The three phases of threat management—detection, investigation, and remediation—can all be automated using a single threat management system. It is essential for users to have a fundamental understanding of data security and to adhere to its best practises in order to protect their devices and systems[61]

CYBER SECURITY THREATS:

Phishing: Phishing involves sending fake emails from trusted sources. Sensitive data like Credit card and login data are targeted. It is the most prevalent cyberattack.

Ransomware: Ransomware is harmful malware. It blocks access to files or the computer system to extort money. Ransom payment does not ensure file recovery or system restoration.

Malware: Malware is a sort of malicious software that is meant to obtain unauthorised access to a computer or to cause damage to the system.

Social engineering: Social engineering is a trick used by opponents to get sensitive information. They can demand money or access your private info. Social engineering can make you click on links, download malware, or believe a fraudulent source when combined with any of the other risks.

SQL injection: SQL (structured language query) injections steal database data and seize control. Data-driven apps are vulnerable to SQL injection attacks by cybercriminals. They can access crucial database information.

MIM attack: A cybercriminal intercepts two people's communication to steal data in a man-in-the-middle attack. An attacker could intercept data from the victim's device and the network on an unprotected WiFi network.

Denial-of-service attack: By flooding networks and servers with traffic, fraudsters block a computer system from delivering genuine requests. The system is unusable, preventing an organisation from performing key operations.

Dridex malware: Dridex, a financial trojan, has many skills. Since 2014, it has infected systems using phishing emails or malware. It has caused hundreds of millions of dollars in financial damages by collecting passwords, banking details, and personal data. This harmful campaign affects the people, government, infrastructure, and business globally. In reaction to the Dridex attacks, the U.K.'s National Cyber Security Centre encourages the public to "ensure devices are patched, anti-virus is turned on and up to date and files are backed up".

Dating scams: The FBI warned Americans in February 2020 about cybercriminals' confidence fraud on dating sites, chat rooms, and smartphones. Victims are duped into giving up personal data by perpetrators. The FBI claims that romance cyber threats cost New Mexico 114 victims $1.6 million in 2019.

Malware Emotet: In late 2019, the Australian Cyber Security Centre cautioned national organisations about Emotet malware's global cyber danger. Emotet, a complex trojan, may steal data and load additional infections. Emotet thrives on simple passwords, emphasising the need for strong passwords to protect against cyberattacks.

End user protection: Endpoint security or end-user protection is vital to cyber security. The end-user often unwittingly uploads malware or other cyber threats to their desktop, laptop, or mobile device.

Level of difficulty caused by Cybercrime

Data breaches continue to grow as the global cyber threat evolves. Data breaches compromised 7.9 billion records in the first nine months of 2019, according to RiskBased Security. In 2018, 112% of records were revealed. Medical services, shops, and public bodies were most breached by malevolent attackers. Because they collect financial and medical data, some of these sectors are more attractive to cybercriminals, but all firms that use networks can be targeted for customer data, corporate espionage, or customer attacks. As the cyber threat grows, worldwide cybersecurity investment will climb. Cybersecurity investment will reach $188.3 billion in 2023 and $260 billion globally by 2026, according to Gartner. The growing cyber threat has prompted governments worldwide to provide advice on cyber-security.

The U.S. National Institute of Standards and Technology (NIST) developed a cyber-security architecture. Continuous, real-time monitoring of all electronic resources is recommended by the framework to counteract malicious code and enable early discovery.[62]

CYBERTHREAT TYPES

Cyber-security addresses three threats:

1. Cybercriminals attack systems for profit or disruption.

2. Cyberattacks often collect political data.

3. Cyberterrorism destabilises electronic systems to generate panic.

SAFETY TIPS

1. Update the software and operating system.
2. Use anti-virus software.
3. Use strong passwords.
4. Don not open email attachments/click on links in email from unknown senders or unfamiliar websites.
5. Avoid using unprotected public WiFi networks.

MACHINE LEARNING & MALWARE ANALYSIS

Malware analysis is the process of understanding the behaviour and the objective of a suspicious file or URL. It is vital to do Android malware analysis to understand and mitigate the hazards posed by malicious software that targets Android devices. It utilises both static and dynamic methodologies, such as signature-based identification, behavioural analysis, machine learning, and static and dynamic analyses, respectively. The difference between static analysis and dynamic analysis is that the former examines the code, resources, and manifest files without executing them, while the latter monitors runtime behaviour in controlled circumstances. The behaviour of an application is monitored using behavioural analysis, which identifies potentially hostile activity such as data exfiltration, unauthorised access, and aggressive advertising. Signature-based detection examines apps' signatures to see if they match known malware signatures, however it may have trouble identifying zero-day or polymorphic varieties of malware. Techniques such as machine learning and deep learning improve the accuracy and effectiveness of detection. In general, Android malware analysis is necessary for the purpose of protecting user data and devices from the ever-evolving dangers that are present.[63] The fig 2 shows the various malware detection approaches and features.

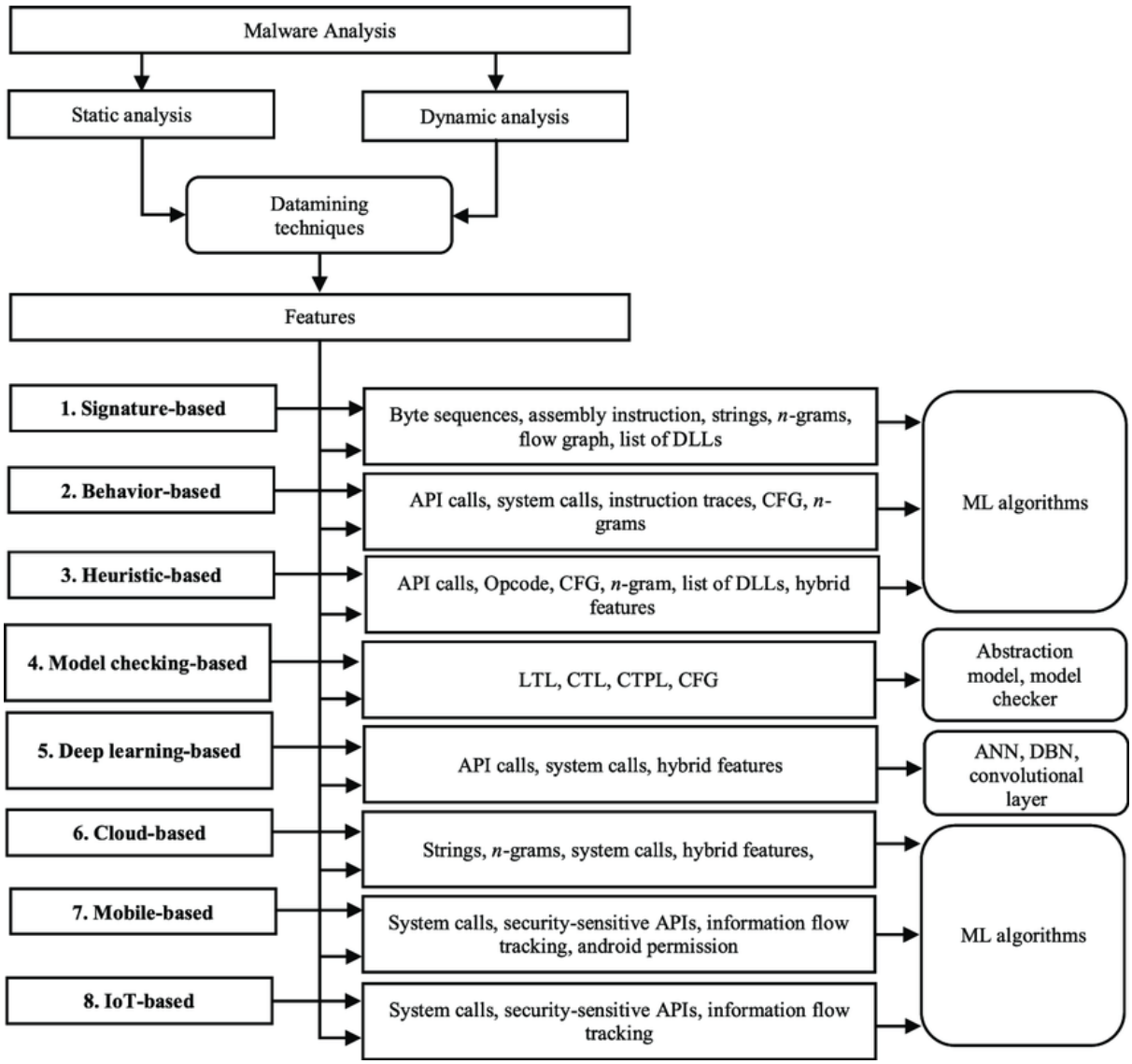

Fig 2: A flow chart of malware detection approaches and features.
(https://images.app.goo.gl/qkg4hiGCQLMnJJ2YA)

TYPES OF MALWARE ANALYSIS

Different types of analysis can be performed on Android malware:

1. Static analysis

2. Dynamic analysis

3. Hybrid analysis

Static Analysis

Analysing code can also be referred to as static analysis. The programme code of the malicious software is seen and then the process is repeated, going through the code one instruction at a time. It does this without actually running the code, so it may review the files for any indications of malicious intent. It is possible that doing so will help you detect malicious infrastructure, libraries, or compressed files. Analysis that is static is known as signature-based analysis. It is also comparable to analyses that are statistically based. Virus scans, fingerprinting, and other security measures are involved. Static analysis relies on the practise of reverse engineering.

Dynamic Analysis

When it comes to the delivery of results, dynamic analysis is superior to static analysis. Analysis of behaviour can also be referred to as dynamic analysis. The malicious code is run in a protected and managed setting known as a sandbox for the purpose of performing dynamic analysis, which investigates the behaviour of the malware when it is run. It is more difficult to undertake dynamic analysis due to the fact that they may make unforeseen modifications to the system, and the majority of malicious software is able to mask their run time actions to some level.

Hybrid Analysis

Static analysis and dynamic analysis, when combined into a single process known as hybrid analysis, can compensate for each other's shortcomings. In other words, it does an analysis on the malware's signature, and then it continues the study by combining the signature with several different behavioural patterns. The advantages of static analysis and the drawbacks of dynamic analysis can be mitigated with the help of hybrid analysis.

MALWARE ANALYSIS PROCESS

There are various steps involved in the process of detecting malware on Android, and these steps can change based on the method or system that is being used to identify the malware. [64]. The fig 3 is an outline of the overall procedure and organisational structure of Android malware detection:

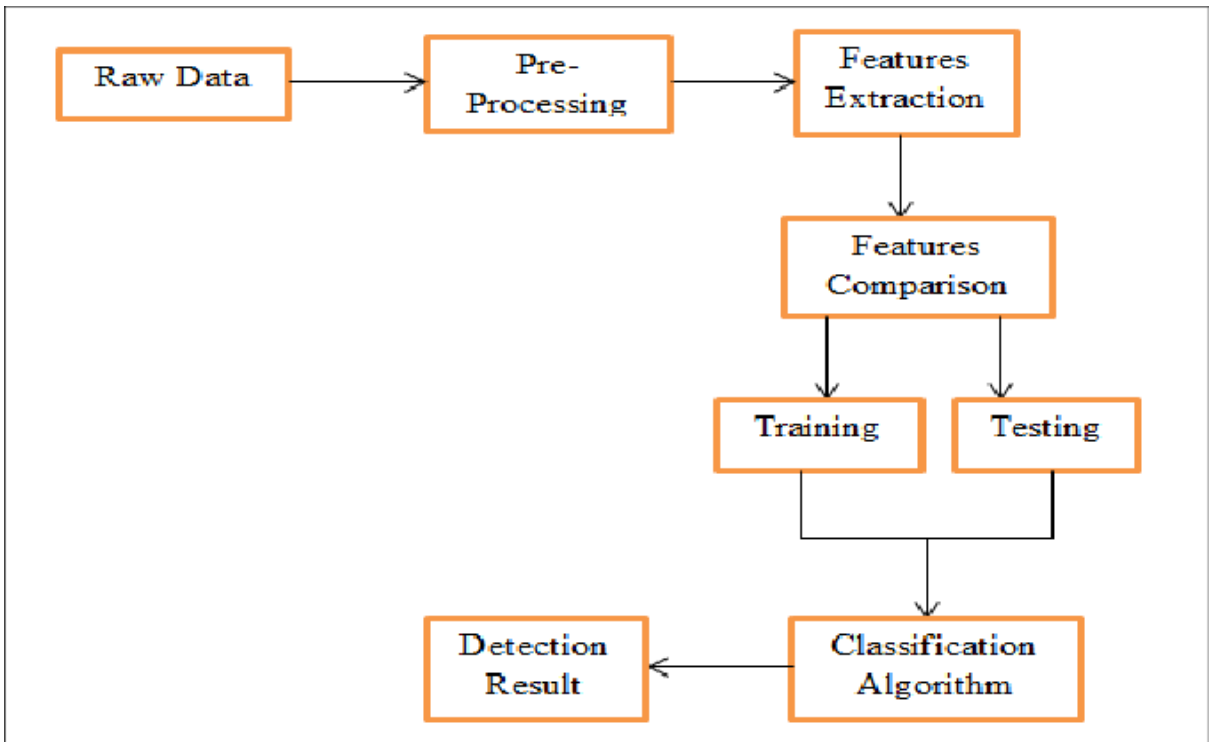

Fig 3: Android malware detection model(https://images.app.goo.gl/1BhX5yabGW4zL92v6)

Data Collection: The very first thing that must be done in order to complete the detection process is to gather data about Android applications. This data may comprise the application's code, permissions requested, API calls made, system call sequences, network traffic, and any other features that are pertinent to the situation. Data can be collected from a wide variety of sources, including programme stores, malware repositories, user devices, and so on.

Preprocessing: After the data have been collected, they are put through preprocessing so that relevant features can be extracted from them, and they are made ready for analysis. This may require parsing the code of the programme, extracting permissions, locating API calls, and translating the data into a format that is acceptable for further analysis.

Feature Extraction: The process of detecting and extracting relevant traits or patterns from the data is referred to as "feature extraction." During this stage, we will attempt to identify the characteristics that set malicious programmes apart from legitimate software and distinguish them from one another. API call sequences, permission usage patterns, code structure, resource file behaviour, and network activity are all examples of common features that are employed in Android malware detection.

Model Training: Training the Models During this stage of the process, either machine learning or deep learning models are trained using labelled datasets. The labelled collection includes examples of both malicious software and programmes that are safe to use. In order for the models to be trained, the extracted features are utilised as input, which enables the models to learn the patterns and characteristics of malware. When it comes to training, you have the option of using a number of different machine learning methods, such as decision trees, support vector machines (SVM), random forests, or even deep learning models, such as convolutional neural networks (CNN) or recurrent neural networks (RNN).

Model Evaluation: After training, the performance of the detection model is then evaluated using evaluation measures such as accuracy, precision, recall, and F1-score. This takes place after the training phase. Evaluation is often carried out on separate datasets, each of which contains examples of known and unknown forms of malware. During this step, the model's capacity to accurately classify malicious software and benign programmes is evaluated, and insights into the model's efficiency and areas in which it could be improved are provided.

Detection and Classification: After the model has been educated and assessed, it may then be used to detect malicious software on Android devices. When a new programme is encountered, its features are parsed out, and the trained model is used to determine whether or not the application is dangerous. The decisions that the model makes are determined by the patterns and traits that it has acquired throughout the training phase. The outcome of the categorization decides whether or not the application is marked as having the potential to be malicious or as being safe.

Post-processing and Remediation: The post-processing step is when further analysis or checks to validate the categorization results can be carried out. This could involve conducting more behavioural analysis, running the programme in a sandbox, or running the application in a controlled environment to observe its behaviour. On the basis of the results of the categorization and any extra analysis that may be performed, suitable actions may be made. These may include isolating or destroying any malware that has been discovered, informing users, or applying remediation steps.

It may include additional phases or modifications depending on the malware detection system or approach that is being utilised. The detection of malware on Android is an ever-evolving field, and researchers and practitioners are continually looking into new approaches and methods to enhance the accuracy and effectiveness of detection systems.

Android malware analysis today

Analysis of Android malware is a fast-developing area that makes use of a wide variety of strategies to identify, investigate, and neutralise potential dangers. Feature extraction and behaviour analysis, obfuscation and evasion techniques, collaborative and crowdsourced analysis, mobile threat intelligence platforms, and app store security are some of the essential components of this field. These techniques assist detect and neutralise dangers posed by malware based on learnt patterns and behaviours. As a result, detection is made more accurately and efficiently. It is necessary for

researchers, security businesses, and the larger community to work together in order to stay current on evolving threats and develop effective solutions. Mobile threat intelligence services give real-time information on newly discovered threats, while app store security measures and analyses assist in preventing the dissemination of dangerous mobile applications. It is essential to maintain an up-to-date knowledge of the most recent research and breakthroughs to effectively resist threats, as the field of Android malware analysis is continuously undergoing evolution.

Research Gap

Android malware analysis can reveal research gaps. Sophisticated evasion techniques, zero-day malware detection, context-aware analysis, sophisticated machine learning, malware analysis automation, dynamic analysis scalability, and privacy-preserving analysis are lacking. Addressing these restrictions improves malware analysis systems' detection accuracy, scalability, and efficacy. Research could improve machine learning, automate analysis, optimise emulator performance, and ensure privacy throughout analysis.

Research questions

(a) whether the use of the ranked list of API calls in apps as the feature of a machine learning-based algorithm is effective for Android malware detection

(b) how to choose the effective subset of API calls for Android malware detection.

(c) What is the ML/DL based methods that can be used to detect malware in Android?

SWOT ANALYSIS

The term "SWOT analysis" is commonly used to refer to a strategic planning tool that is utilised in the business world to determine a company's strengths, weaknesses, opportunities, and threats respectively. Here we discuss the strengths, weaknesses, opportunities and threats in the Android Malware Analysis field.

STRENGTHS: makes the researchers to gain deep insights into malware behaviour and characteristics.

- Advanced techniques
- Growing expertise
- Automated analysis platforms

WEAKNESS: Evasion strategies can produce false negatives and undetected viruses.

- Evasion methods
- Changing threat landscape
- Imbalanced data

The requirement for continuous advancements in machine learning and context aware analysis

OPPORTUNITIES

Opportunities for Android malware analysis include advancements in machine learning, which can improve the accuracy and efficiency of malware detection and analysis. Context-aware analysis, which considers user behaviour, device settings, and network conditions, can lead to more precise and contextual detection

THREATS: threats include zero-day malware, which exploits unknown vulnerabilities and requires continuous research and development of proactive analysis techniques. Privacy and ethical concerns are essential in the analysis process, as accessing sensitive user data and interacting with malicious code is crucial.

CONCLUSION

In Android malware research, graph-based algorithms are helpful for recognising known and unknown samples, finding harmful patterns, and capturing runtime behaviours. Scalability, intricacy, and evasion methods continue to be difficulties, though. In the future, research should concentrate on areas such as scalability, efficient algorithms, and machine learning. Malware detection that is based on the GCN can accommodate variations in viruses by extracting API call sequences, directed cyclic graphs, characteristics, and the GCN itself for categorization purposes. It's possible to adapt the technology. To reduce malware detection staff expenses, GCN-based adaptive detection methods will be studied.

REFERENCES

1. https://www.cisco.com/c/en_in/products/security/what-is-
2. Bhat, P., & Dutta, K. (2019, February 13). A Survey on Various Threats and Current State of Security in Android Platform. *ACM Computing Surveys*, *52*(1), 1–35. https://doi.org/10.1145/3301285
3. Aafer, Y., Du, W., & Yin, H. (2013). DroidAPIMiner: Mining API-Level Features for Robust Malware Detection in Android. *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, 86–103. https://doi.org/10.1007/978-3-319-04283-1_6
4. Qiao, M., Sung, A. H., & Liu, Q. (2016, July). Merging Permission and API Features for Android Malware Detection. 2016 5th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI). https://doi.org/10.1109/iiai-aai.2016.237
5. Chan, P. P. K., & Wen-Kai Song. (2014, July). Static detection of Android malware by using permissions and API calls. 2014 International Conference on Machine Learning and Cybernetics. https://doi.org/10.1109/icmlc.2014.7009096
6. Wang, Z., Cai, J., Cheng, S., & Li, W. (2016, September). DroidDeepLearner: Identifying Android malware using deep learning. *2016 IEEE 37th Sarnoff Symposium*. https://doi.org/10.1109/sarnof.2016.7846747
7. Singh, A. K., Jaidhar, C. D., & Kumara, M. A. A. (2019, May 30). Experimental analysis of Android malware detection based on combinations of permissions and API-calls. *Journal of Computer Virology and Hacking Techniques*, *15*(3), 209–218. https://doi.org/10.1007/s11416-019-00332-z
8. Li, W., Ge, J., & Dai, G. (2015, November). Detecting Malware for Android Platform: An SVM-Based Approach. *2015 IEEE 2nd International Conference on Cyber Security and Cloud Computing*. https://doi.org/10.1109/cscloud.2015.50
9. Kumar, R., Zhang, X., Wang, W., Khan, R. U., Kumar, J., & Sharif, A. (2019). A Multimodal Malware Detection Technique for Android IoT Devices Using Various Features. *IEEE Access*, *7*, 64411–64430. https://doi.org/10.1109/access.2019.2916886
10. Koli, J. D. (2018, March). RanDroid: Android malware detection using random machine learning classifiers. *2018 Technologies for Smart-City Energy Security and Power (ICSESP)*. https://doi.org/10.1109/icsesp.2018.8376705
11. Onwuzurike, L., Mariconti, E., Andriotis, P., Cristofaro, E. D., Ross, G., & Stringhini, G. (2019, April 9). MaMaDroid. *ACM Transactions on Privacy and Security*, *22*(2), 1–34. https://doi.org/10.1145/3313391

12. Cai, L., Li, Y., & Xiong, Z. (2021, January). JOWMDroid: Android malware detection based on feature weighting with joint optimization of weight-mapping and classifier parameters. *Computers & Security*, *100*, 102086. https://doi.org/10.1016/j.cose.2020.102086
13. Muzaffar, A., Ragab Hassen, H., Lones, M. A., & Zantout, H. (2022, October). An in-depth review of machine learning based Android malware detection. Computers & Security, 121, 102833. https://doi.org/10.1016/j.cose.2022.102833
14. Guerra-Manzanares, A., Bahsi, H., & Nõmm, S. (2021, November). KronoDroid: Time-based Hybrid-featured Dataset for Effective Android Malware Detection and Characterization. Computers & Security, 110, 102399. https://doi.org/10.1016/j.cose.2021.102399
15. Shatnawi, A. S., Yassen, Q., & Yateem, A. (2022). An Android Malware Detection Approach Based on Static Feature Analysis Using Machine Learning Algorithms. Procedia Computer Science, 201, 653–658. https://doi.org/10.1016/j.procs.2022.03.086
16. McLaughlin, N., Martinez del Rincon, J., Kang, B., Yerima, S., Miller, P., Sezer, S., Safaei, Y., Trickel, E., Zhao, Z., Doupé, A., & Joon Ahn, G. (2017, March 22). Deep Android Malware Detection. *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy*. https://doi.org/10.1145/3029806.3029823
17. Ismail, Mohmand, M. I., Hussain, H., Khan, A. A., Ullah, U., Zakarya, M., Ahmed, A., Raza, M., Rahman, I. U., & Haleem, M. (2022). A Machine Learning-Based Classification and Prediction Technique for DDoS Attacks. *IEEE Access*, *10*, 21443–21454. https://doi.org/10.1109/access.2022.3152577
18. Chen, L., Hou, S., Ye, Y., & Chen, L. (2017). An Adversarial Machine Learning Model Against Android Malware Evasion Attacks. *Web And Big Data*, 43–55. https://doi.org/10.1007/978-3-319-69781-9_5
19. Zhai, S., Chang, K. H., Zhang, R., & Zhang, Z. M. (2016, August 13). DeepIntent. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. https://doi.org/10.1145/2939672.2939759
20. Shabir, A. G., & Sabahat, N. (2020, November 5). A Review of Hybrid Malware Detection Techniques in Android. *2020 IEEE 23rd International Multitopic Conference (INMIC)*. https://doi.org/10.1109/inmic50486.2020.9318117
21. Wang, W., Gao, Z., Zhao, M., Li, Y., Liu, J., & Zhang, X. (2018). DroidEnsemble: Detecting Android Malicious Applications With Ensemble of String and Structural Static Features. *IEEE Access*, *6*, 31798–31807. https://doi.org/10.1109/access.2018.2835654
22. Feizollah, A., Anuar, N. B., Salleh, R., Suarez-Tangil, G., & Furnell, S. (2017, March). AndroDialysis: Analysis of Android Intent Effectiveness in Malware Detection. *Computers & Security*, *65*, 121–134. https://doi.org/10.1016/j.cose.2016.11.007
23. Haq, I. U., Khan, T. A., Akhunzada, A., & Liu, X. (2021, August 2). MalDroid: Secure DL-enabled intelligent malware detection framework. *IET Communications*, *16*(10), 1160–1171. https://doi.org/10.1049/cmu2.12265
24. Zhu, H., et al.: Sedmdroid: An enhanced stacking ensemble of deep learning framework for android malware detection. IEEE Trans. Netw. Sci. Eng.(2020)
25. Bibi, I., et al.: A dynamic dl-driven architecture to combat sophisticated android malware. IEEE Access 8, 129600–129612 (2020)
26. Amin, M., et al.: Static malware detection and attribution in android byte-code through an end-to-end deep system. Future Gener. Comput. Syst.102, 112–126 (2020)

27. Xiao, X., et al.: Android malware detection based on system call sequences and lstm. Multimedia Tools Appl. 78(4), 3979–3999 (2019)
28. Li, J., et al.: Significant permission identification for machine-learningbased android malware detection. IEEE Trans. Ind. Inf. 14(7), 3216–3225 (2018)
29. Zhang, C., et al.: Deep learning in mobile and wireless networking: A survey. IEEE Commun. Surv. Tutorials 21(3), 2224–2287 (2019)
30. Suarez-Tangil, G., Stringhini, G.: Eight years of rider measurement in the android malware ecosystem. IEEE Trans. Dependable Secure Comput. 99 (2020)
31. Singh, A. K., Jaidhar, C. D., & Kumara, M. A. A. (2019, May 30). Experimental analysis of Android malware detection based on combinations of permissions and API-calls. *Journal of Computer Virology and Hacking Techniques*, *15*(3), 209–218. https://doi.org/10.1007/s11416-019-00332-z
32. Arzt, S., Rasthofer, S., Fritz, C., Bodden, E., Bartel, A., Klein, J., Le Traon, Y., Octeau, D., & McDaniel, P. (2014, June 5). FlowDroid. *ACM SIGPLAN Notices*, *49*(6), 259–269. https://doi.org/10.1145/2666356.2594299
33. Hou, S., Ye, Y., Song, Y., & Abdulhayoglu, M. (2017, August 13). HinDroid. *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. https://doi.org/10.1145/3097983.3098026
34. Gao, H., Cheng, S., & Zhang, W. (2021, July). GDroid: Android malware detection and classification with graph convolutional network. *Computers & Security*, *106*, 102264. https://doi.org/10.1016/j.cose.2021.102264
35. Fan, M., Luo, X., Liu, J., Wang, M., Nong, C., Zheng, Q., & Liu, T. (2019, May). Graph Embedding Based Familial Analysis of Android Malware using Unsupervised Learning. *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. https://doi.org/10.1109/icse.2019.00085
36. Singh, A. K., Jaidhar, C. D., & Kumara, M. A. A. (2019, May 30). Experimental analysis of Android malware detection based on combinations of permissions and API-calls. *Journal of Computer Virology and Hacking Techniques*, *15*(3), 209–218. https://doi.org/10.1007/s11416-019-00332-z
37. Singh, J., & Singh, J. (2020, February 26). Assessment of supervised machine learning algorithms using dynamic API calls for malware detection. *International Journal of Computers and Applications*, *44*(3), 270–277. https://doi.org/10.1080/1206212x.2020.1732641
38. Zhu, D., Xi, T., Jing, P., Wu, D., Xia, Q., & Zhang, Y. (2019, November 25). A Transparent and Multimodal Malware Detection Method for Android Apps. *Proceedings of the 22nd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*. https://doi.org/10.1145/3345768.3355915
39. Ye, Y., Hou, S., Chen, L., Lei, J., Wan, W., Wang, J., Xiong, Q., & Shao, F. (2019, August). Out-of-sample Node Representation Learning for Heterogeneous Graph in Real-time Android Malware Detection. *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*. https://doi.org/10.24963/ijcai.2019/576
40. Hou, S., Ye, Y., Song, Y., & Abdulhayoglu, M. (2018, July). Make Evasion Harder: An Intelligent Android Malware Detection System. *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*. https://doi.org/10.24963/ijcai.2018/737
41. Xiao, F., Lin, Z., Sun, Y., & Ma, Y. (2019, February 11). Malware Detection Based on Deep Learning of Behavior Graphs. *Mathematical Problems in Engineering*, *2019*, 1–10. https://doi.org/10.1155/2019/8195395

42. Garg, V., & Yadav, R. K. (2019, November). Malware Detection based on API Calls Frequency. *2019 4th International Conference on Information Systems and Computer Networks (ISCON)*. https://doi.org/10.1109/iscon47742.2019.9036219
43. Peiravian, N., & Zhu, X. (2013, November). Machine Learning for Android Malware Detection Using Permission and API Calls. *2013 IEEE 25th International Conference on Tools With Artificial Intelligence*. https://doi.org/10.1109/ictai.2013.53
44. Aafer, Y., Du, W., & Yin, H. (2013). DroidAPIMiner: Mining API-Level Features for Robust Malware Detection in Android. *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, 86–103. https://doi.org/10.1007/978-3-319-04283-1_6
45. Goyal, R., Spognardi, A., Dragoni, N., & Argyriou, M. (2016, November). SafeDroid: A Distributed Malware Detection Service for Android. *2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA)*. https://doi.org/10.1109/soca.2016.14
46. Jung, J., Kim, H., Shin, D., Lee, M., Lee, H., Cho, S. J., & Suh, K. (2018, September). Android Malware Detection Based on Useful API Calls and Machine Learning. *2018 IEEE First International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*. https://doi.org/10.1109/aike.2018.00041
47. Kim, J., Ban, Y., Ko, E., Cho, H., & Yi, J. H. (2022, February 9). MAPAS: a practical deep learning-based android malware detection system. *International Journal of Information Security*, *21*(4), 725–738. https://doi.org/10.1007/s10207-022-00579-6
48. Zhang, H., Luo, S., Zhang, Y., & Pan, L. (2019). An Efficient Android Malware Detection System Based on Method-Level Behavioral Semantic Analysis. *IEEE Access*, *7*, 69246–69256. https://doi.org/10.1109/access.2019.2919796
49. Zhang, N., Xue, J., Ma, Y., Zhang, R., Liang, T., & Tan, Y. (2021, July 12). Hybrid sequence-based Android malware detection using natural language processing. *International Journal of Intelligent Systems*, *36*(10), 5770–5784. https://doi.org/10.1002/int.22529
50. Naval, S., Laxmi, V., Rajarajan, M., Gaur, M. S., & Conti, M. (2015, December). Employing Program Semantics for Malware Detection. *IEEE Transactions on Information Forensics and Security*, *10*(12), 2591–2604. https://doi.org/10.1109/tifs.2015.2469253
51. Wu, Y., Li, X., Zou, D., Yang, W., Zhang, X., & Jin, H. (2019, November). MalScan: Fast Market-Wide Mobile Malware Scanning by Social-Network Centrality Analysis. *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. https://doi.org/10.1109/ase.2019.00023
52. Feng, Y., Anand, S., Dillig, I., & Aiken, A. (2014, November 11). Apposcopy: semantics-based detection of Android malware through static analysis. *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*. https://doi.org/10.1145/2635868.2635869
53. Feng, Y., Bastani, O., Martins, R., Dillig, I., & Anand, S. (2017). Automated Synthesis of Semantic Malware Signatures using Maximum Satisfiability. *Proceedings 2017 Network and Distributed System Security Symposium*. https://doi.org/10.14722/ndss.2017.23379
54. Zou, D., Wu, Y., Yang, S., Chauhan, A., Yang, W., Zhong, J., Dou, S., & Jin, H. (2021, May 8). IntDroid. *ACM Transactions on Software Engineering and Methodology*, *30*(3), 1–32. https://doi.org/10.1145/3442588

55. Taheri, L., Kadir, A. F. A., & Lashkari, A. H. (2019, October). Extensible Android Malware Detection and Family Classification Using Network-Flows and API-Calls. *2019 International Carnahan Conference on Security Technology (ICCST)*. https://doi.org/10.1109/ccst.2019.8888430

56. Li, S., Zhou, Q., Zhou, R., & Lv, Q. (2021, August 24). Intelligent malware detection based on graph convolutional network. *The Journal of Supercomputing*, *78*(3), 4182–4198. https://doi.org/10.1007/s11227-021-04020-y

57. Amer, E., Zelinka, I., & El-Sappagh, S. (2021, November). A Multi-Perspective malware detection approach through behavioral fusion of API call sequence. *Computers & Security*, *110*, 102449. https://doi.org/10.1016/j.cose.2021.102449

58. Feng, P., Ma, J., Li, T., Ma, X., Xi, N., & Lu, D. (2021, June 4). Android Malware Detection via Graph Representation Learning. *Mobile Information Systems*, *2021*, 1–14. https://doi.org/10.1155/2021/5538841

59. Graph Approach for android malware detection using machine learning techniques. (2021, November 1). *Humanitarian and Natural Sciences Journal*, *2*(11). https://doi.org/10.53796/hnsj21115

60. Surendran, R., Thomas, T., & Emmanuel, S. (2020, November). GSDroid: Graph Signal Based Compact Feature Representation for Android Malware Detection. *Expert Systems With Applications*, *159*, 113581. https://doi.org/10.1016/j.eswa.2020.113581

61. https://u-next.com/blogs/cyber-security/malware-analysis/#:~:text=Malware%20analysis%20can%20be%20described,incident%20responders%20and%20security%20analysts.

62. https://www.kaspersky.co.in/resource-center/definitions/what-is-cyber-security

63. https://u-next.com/blogs/cyber-security/malware-analysis/#:~:text=Malware%20analysis%20can%20be%20described,incident%20responders%20and%20security%20analysts.

64. https://www.researchgate.net/publication/344455629_Deep-Droid_Deep_Learning_for_Android_Malware_Detection