Computers and Electrical Engineering 117 (2024) 109233

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

Malware detection for mobile computing using secure and privacy-preserving machine learning approaches: A comprehensive survey

Faria Nawshin [a,b], Radwa Gad [b], Devrim Unal [a], Abdulla Khalid Al-Ali [b], Ponnuthurai N. Suganthan [a,*]

[a] KINDI Computing Research Center, College of Engineering, Qatar University, Doha, Qatar
[b] Department of Computer Science & Engineering, College of Engineering, Qatar University, Doha, Qatar

ARTICLE INFO

Keywords:
Mobile malware analysis
Privacy-preserving machine-learning
Secure machine-learning
Mobile security attacks
Federated learning
Mobile vulnerabilities

ABSTRACT

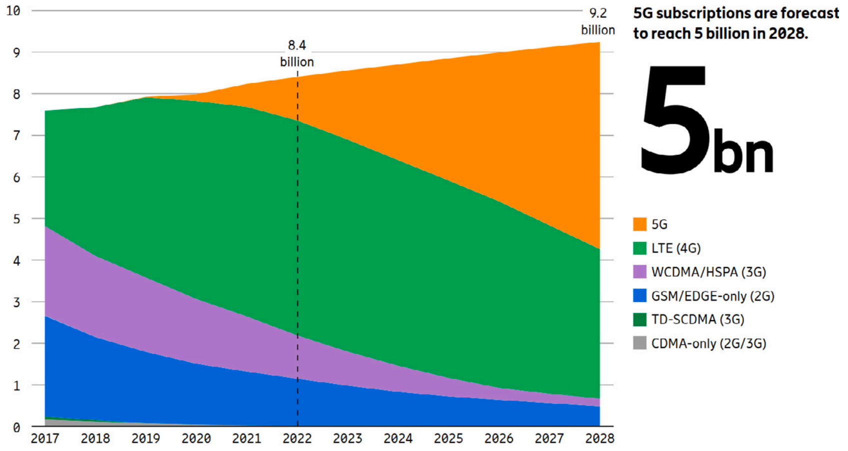
Mobile devices have become an essential element in our day-to-day lives. The chances of mobile attacks are rapidly increasing with the growing use of mobile devices. Exploiting vulnerabilities from devices as well as stealing personal information, are the principal targets of the attackers. Researchers are also developing various techniques for detecting and analyzing mobile malware to overcome these issues. As new malware gets introduced frequently by malware developers, it is very challenging to come up with comprehensive algorithms to detect this malware. There are many machine-learning and deep-learning algorithms have been developed by researchers. The accuracy of these models largely depends on the size and quality of the training dataset. Training the model with a diversified dataset is necessary to predict new malware accurately. However, this training process may raise the issue of privacy loss due to the disclosure of sensitive information of the users. Researchers have proposed various techniques to mitigate this issue, such as differential privacy, homomorphic encryption, and federated learning. This survey paper explores the significance of applying federated learning to the mobile operating systems, contrasting traditional machine learning and deep learning approaches for mobile malware detection. We delve into the unique challenges and opportunities of the architecture of in-built mobile operating systems and their implications for user privacy and security. Moreover, we assess the risks associated with federated learning in real-life applications and recommend strategies for developing a secure federated learning framework in the domain of mobile malware detection.

1. Introduction

This survey paper analyzes the in-built security mechanisms of the mobile operating systems and makes a comparative discussion between Android and iOS in terms of malware threats. After that, the methods of the conventional machine learning or deep learning models to identify the malware attacks are explained along with the security challenges of these techniques. To mitigate the issue of privacy leakage of conventional machine learning models, this survey paper explores several techniques for privacy preservation of user-sensitive data along with the advantages and drawbacks of these methods. Because of the dynamic and pervasive nature

[*] Corresponding author.
E-mail addresses: fnawshin@qu.edu.qa (F. Nawshin), rg2110609@qu.edu.qa (R. Gad), dunal@qu.edu.qa (D. Unal), abdulla.alali@qu.edu.qa (A.K. Al-Ali), p.n.suganthan@qu.edu.qa (P.N. Suganthan).

https://doi.org/10.1016/j.compeleceng.2024.109233
Received 12 February 2024; Received in revised form 23 March 2024; Accepted 2 April 2024
Available online 11 April 2024



[1] GSA (November 2022).
[2] A 5G subscription is counted as such when associated with a device that supports New Radio (NR), as specified in 3GPP Release 15, and is connected to a 5G-enabled network.
[3] Mainly CDMA2000 EVDO, TD-SCDMA and Mobile WiMAX.

Fig. 1. Mobile subscriptions by technology (Billion) [1].

of the malware applications, our survey paper not only analyzes the machine learning or deep learning models but also focuses on the significance of federated learning in augmenting privacy-preserving mechanisms in Android malware analysis. This survey paper provides a comprehensive review of how federated learning is incorporated into mobile security strategies. Besides, how this integration is better at protecting user privacy is highlighted in the paper while effectively detecting malware. However, as the malware attackers develop different innovative approaches to attack the federated learning models, this survey provides an overview of the recent security attacks of federated learning and the possible defense mechanisms to detect or mitigate those attacks. Finally, this paper outlines future research directions considering the measures found in the recent studies for developing a robust and secured federated learning solution in Android malware detection.

Due to the growing number of mobile phone users, mobile application developers are developing many eye-catching applications. People use mobile phones for various aspects of daily life, including answering calls, attending online meetings, location navigation while driving, etc. According to a statistical report by Ericsson [1], Fig. 1 shows mobile subscriptions by technology (billion), and it is noticeable that there was a sharp increase in mobile subscriptions in 2022. Mobile phones are assumed to replace computers, as they can perform almost all tasks. More than 4 million applications are available in Google Play and the Apple Store, according to the report of [2,3]. Malicious application developers are also taking advantage of this surge in usage, and their targets have shifted from computers to mobile phones to hamper people's security. According to another statistical report generated by AV-TEST Institute [4], every day, more than 450,000 new malicious applications (malware) and Potentially Unwanted Applications (PUA) are registered.

Smartphones have become unavoidable devices worldwide, serving as tools for managing daily activities. However, storing sensitive and confidential information, such as personal images, bank account details, and different website account credentials, increases the security threats to mobile phones. In some cases, developers who do not follow the rules and regulations of the applications' security framework can also cause attacks. Tangari et al. [5] showed how health applications in mobile phones can serve as entry points for attackers. He et al. [6] demonstrated that different cryptocurrency wallet applications are more vulnerable to security attacks.

Bergadano et al. [7] proposed a solution for detecting malicious applications on both Android and iOS (iPhone operating system) platforms. Additionally, Weichbroth et al. [8] explained the best practices that should be followed to protect the applications. In their research, they observed the behavior of common mobile security threats. The existing methods and algorithms are always insufficient to detect new malicious applications.

Garg et al. [9] compared the likelihood of mobile malware attacks between iOS and Android operating systems. Their study revealed that the Android operating system is more susceptible to security threats than iOS. One key factor contributing to this difference is that iOS strictly prohibits third-party application developers/providers from uploading their applications directly to the Apple Store. Instead, iOS developers must submit their applications to the official application market known as the App Store. Only after undergoing a digital signing process are these applications published in the App Store. Although Android and iOS have integrated security features to prevent these threats, as discussed in Section 2.1.2, attackers find vulnerabilities and gain unauthorized access to mobile devices, allowing them to carry out malicious activities. New malicious applications are uploaded to Google Play daily, emphasizing the urgent need for robust malware detection algorithms.

This study aims to bridge the gap in existing security measures by leveraging machine learning techniques to enhance mobile malware detection while prioritizing the protection of user privacy. By focusing on the vulnerabilities of the Android platform

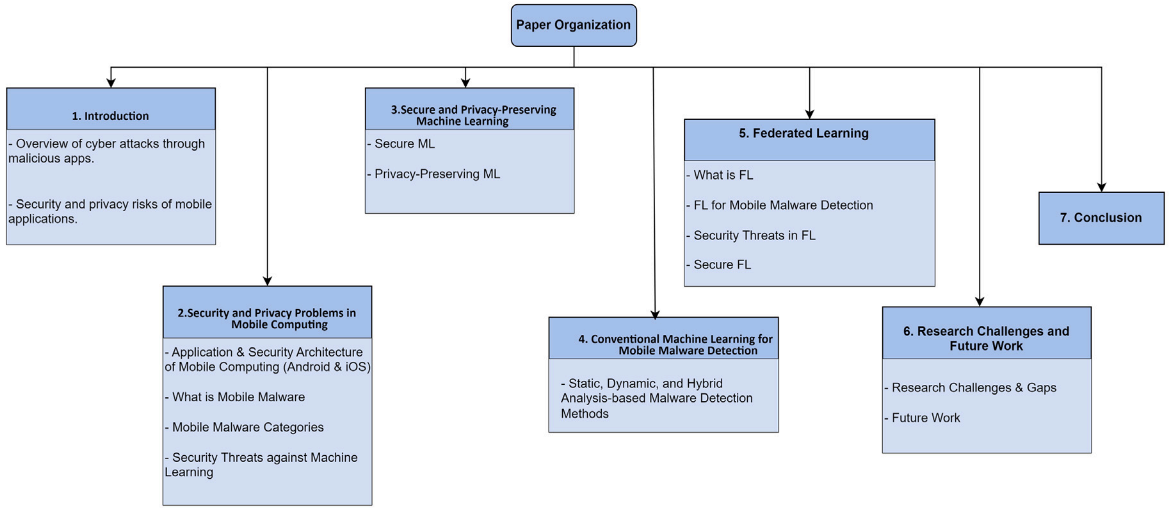

Paper Organization
1. Introduction
- Overview of cyber attacks through malicious apps.
- Security and privacy risks of mobile applications.
2.Security and Privacy Problems in Mobile Computing
- Application & Security Architecture of Mobile Computing (Android & iOS)
- What is Mobile Malware
- Mobile Malware Categories
- Security Threats against Machine Learning
3.Secure and Privacy-Preserving Machine Learning
- Secure ML
- Privacy-Preserving ML
4. Conventional Machine Learning for Mobile Malware Detection
- Static, Dynamic, and Hybrid Analysis-based Malware Detection Methods
5. Federated Learning
- What is FL
- FL for Mobile Malware Detection
- Security Threats in FL
- Secure FL
6. Research Challenges and Future Work
- Research Challenges & Gaps
- Future Work
7. Conclusion

Fig. 2. Organization of the paper.

compared to iOS and reviewing current detection methods alongside emerging threats, we emphasize the necessity of our research. This paper distinguishes the novelties from existing surveys by focusing on the integration of federated learning within the Android operating system that still remains largely unexplored in current literature. Our analysis compares conventional machine learning and deep learning models in identifying malware applications. Additionally, we include the solution mechanisms for detecting specific malware categories, such as adware, backdoors, trojans, riskware, ransomware, and more, considering the design of different attack models. Unlike previous surveys on federated learning or mobile security, our work delves into the distinct challenges and opportunities presented by the system and security architecture of Android and user base. We also address the challenges of data and model heterogeneity in federated learning systems and discuss the mechanisms of overcoming these found in recent papers. Moreover, we offer a comprehensive analysis of the potentiality of federated learning to enhance privacy and security for Android devices which is further supported by a detailed examination of current methodologies, their limitations, and future directions. This approach will provide novel insights and practical recommendations tailored to developers, researchers and other stakeholders interested in leveraging federated learning to address Android-specific security concerns. The main contributions of this article are summarized as follows:

1. We provide a detailed analysis of the application and security architectures of the Android and iOS platforms. Outlining how applications are structured and secured to offer foundational insights into the mechanisms used by these operating systems to protect against security threats and ensure user data privacy.
2. We analyze conventional machine learning algorithms for mobile malware detection and discuss the privacy and security threats associated with these methods. We also describe different security attacks on conventional machine learning models along with the design of the attack models and defense mechanisms found in recent papers. We further introduce advanced techniques for privacy-preserving and secure machine learning.
3. We present the challenges of data and model heterogeneity in Federated Learning systems and discuss strategies found in the recent studies to overcome these.
4. We highlight the potential security threats of Federated Learning when implemented in real-world applications and present the strategies of secure and privacy-preserving approaches to integrate with Federated Learning.
5. We propose the development of secure Federated Learning by synthesizing recent research, outlining the challenges faced, and suggesting directions for future research to enhance security and privacy in mobile computing.

Paper Organization: Section 2 explains security and privacy problems in mobile computing. Secure and Privacy-Preserving Machine Learning is discussed in Section 3. Section 4 discusses the existing work on conventional machine learning for mobile malware detection. Furthermore, a brief explanation of Federated Learning with malware detection is described in Section 5. Finally, Section 6 discusses the research challenges and future work, and Section 7 concludes the article. The organization of the paper is shown in Fig. 2.

2. Security and privacy problems in mobile computing

The Android operating system allows the installation of applications from unverified third-party markets, which can pose a potential security risk. Furthermore, the Android source code is openly released and accessible to all, whereas, in contrast, the iOS source code remains undisclosed. In the iOS environment, developers are granted access to develop their own applications, while unauthorized access is strictly restricted [9]. The attackers did not spare the iOS platform despite using a closed operating system. Most previous and existing research works focused on Android malware classification and detection. In contrast, researchers are currently trying to detect and classify malware for iOS, but they are not proven as solid methods. These research gaps nowadays influence attackers to attack in iOS platform [10].

2.1. Application & security architecture of mobile computing

This section discusses the application architecture in Section 2.1.1 and security architecture in Section 2.1.2 of both Android and iOS operating systems.

2.1.1. Application structure

The application structure comprises the organization and structure of application components, such as its main components, file arrangement, and interconnectivity.

Android

The description of application metadata, required permissions, external libraries, and platform requirements are mentioned in android.manifest file [11]. An Android application is available in (Android Package Kit) APK form. APK file consists of lib, assets, resource files, manifest, res, and Dalvik bytecode. These files also serve as the sources for static features. The feature vector's extraction method depends on each file's role.

- Permission Features: All the permissions that an application requires to run are mentioned in AndroidManifest.xml file and these permissions can be used as a static feature to detect whether an application is malicious or benign [12].
- Component Features: The AndroidManifest.xml registers four fundamental components found in classes.dex and AndroidManifest.xml files. classes.dex file stores system calls that are needed to declare and create those components [13].
- Intent Features: Intents transmit messages between components and intents are found in classes.dex and AndroidManifest.xml files. With the combination of intents and components, a feature vector can be made which helps in the two components' association analysis process. [14,15].
- Constant String Features: The strings which are defined by the developer are stored in strings.xml and are defined by smali code and are stored in .dex files. String frequencies and content can be extracted to analyze the application characteristics [16].
- Resource File Features: Layout files, images, and sounds from res and asset directories are not modified by malicious codes and this can be served as a static feature [17,18].
- API Features and Opcode: To generate the detection features, the frequency of API calls and Dalvik Opcodes can be considered to analyze the coding habit of the developers. It can be significantly differentiated between Malware and Benign application by counting the frequency of API calls and opcodes [17].
- Native Code Features: Native instructions are used by many malware applications from .so files to do malicious activities which makes it hard to decompile from the compiled code. The invocation frequency of system call and the extraction of arm opcode frequency help to detect malware applications [17].
- Data Flow and Control Flow Graph Features: The relationship between invocation instruction and the data flow directions can help to generate a data flow graph and control flow graph. Vector representation can be generated from these graphs which helps the detection work [17].

The attackers target this application structure and analyze the underlying components for the purpose of exploiting vulnerabilities in different features to cause harm to the devices or users. The approaches that attackers follow to perform security attacks by using static features are discussed below.

- Permission Abuse: Malicious applications often request excessive permissions while installing [19]. Alenezi et al. [20] proposed a solution on how to select required permissions for an application. Sharmeen et al. [21] proposed a framework using deep learning and semi-supervised techniques to identify malware behavior by analyzing permissions and escalations that achieved 99.024% accuracy to find out new malware.
- Component Hijacking: AndroidManifest.xml gets manipulated and the attackers either replace or hijack the legit components with a malicious one in order to steal sensitive information of the users. Wu et al. [22] proposed a solution named SCLib to overcome the component hijacking issue. SCLib is a secure component library designed to provide in-app mandatory access control (MAC) for Android applications, thereby defending against component hijacking attacks. It defines and enforces a set of MAC policies within applications to prevent unauthorized component access that addresses weaknesses of the system.
- Intent Spoofing: The exchange of intents between components is manipulated by malicious applications for the purpose of redirecting data to malicious destinations. By performing this attack, the attackers can gain access to the system and carry out unauthorized actions [23].
- Code Injection: This involves inserting malicious codes into the executable files of an application such as the classes.dex file. Choi et al. [24] investigated the underlying conditions responsible for remote code injection attacks.

- Resource Tampering: The resource files such as images, and layout files are modified by the attackers. They insert malicious codes into the application which leads to presenting misleading content to the users. Cai et al. [25] suggested solutions for resource tampering attacks, such as ensuring application checking for camera access and alerting users if it is unavailable which secures camera usage for foreground applications, and prevents unauthorized touchscreen interactions by detecting overlay attacks.
- API Abuse: Malware attackers target different APIs also for abuse and to gain unauthorized access to the system and manipulate the functionalities of the system. Ruggia et al. [26] discussed how mobile banking applications are susceptible to phishing attacks through the abuse of "inotify" APIs which is a mechanism for monitoring file system events. They demonstrated that over 10,000 malware samples identified in 2021 exploited "inotify" APIs for malicious purposes. Ito et al. [27] proposed a framework for the detection of accessing private information by abusing APIs that are not related to the functionalities of the applications.
- Native Code Exploitation: Native libraries or native codes can be modified by security attackers by inserting malicious native code to affect the functionalities of the system. Fedler et al. [28] proposed a technique for controlling the native code execution which can mitigate this issue.
- Dynamic Code Loading: This technique allows an application to load and execute code from external sources at runtime rather than including it at compile time. Attackers can exploit this feature to bypass static analysis tools and load malicious code after an application has been installed or execute arbitrary code without the user's knowledge. Albanese et al. [29] explored how attackers exploit dynamic loading and dynamic compilation techniques to evade Android's security mechanisms.

iOS

The configuration file of iOS is called *Info.plist* and consists of the settings of the applications and metadata. Some of the components of this file are discussed below.

- Metadata: This involves information consisting of metadata about the applications, such as version number, bundle identifier of the application, display name, and more. The applications are identified by the iOS based on metadata [30].
- Permissions: The permissions such as location, camera, microphone, and more that are required to install and execute an application are specified in *info.plist* file [31].
- Localization: Localized versions of both application metadata and configuration details of the applications for different languages are mentioned in *info.plist*. It helps to display the application's language based on the users' geographical location [32].

2.1.2. Security architecture

The security architecture of Android and iOS, the two major mobile operating systems, is designed to ensure the confidentiality, integrity, and availability of user data and protect against various security threats. This section discusses the security architecture of both Android and iOS.

Android

Android, a Linux kernel-based operating system developed by Google, incorporates APIs, Java-compatible libraries, and C programming language middleware. Android utilizes the Dalvik VM, which executes multiple sandboxes simultaneously [33], ensuring the isolation of Android applications based on the structure discussed in 2.1.1. To safeguard user privacy and secure application interactions, the Android operating system includes built-in security measures, which are detailed below.

- Shared ID: Shared ID enables data sharing among application components. Applications require the same digital certificate to obtain a shared user ID and run within the same process.
- Permissions: Permissions are categorized into three groups based on the type of user data access required during run-time:
  - Normal Permission: It allows access to user data outside the sandbox with minimal risk. Explicit user approval is not required, but users can review permissions post-installation.
  - Dangerous Permission: It grants access to critical user data like location, camera, microphone, etc. Users must grant these permissions at application runtime due to potential risks.
  - Signature Permission: It is required for applications to sign the same digital certificate and acquire a shared user ID. Signature permissions are granted by the Android system during installation [34].

iOS

iOS provides default hardware and software-level security features to prevent security attacks:

- System Security: iOS ensures security through encryption, application sandboxing, and isolation. By encrypting the entire system by default, enforcing strict application sandboxing rules, and utilizing secure boot processes, iOS creates a secure environment resistant to unauthorized access and attacks [35].
- Data Security: IOS employs encryption for data at rest and in transit to safeguard user data, followed by regular security checks and updates. Application sandboxing also plays a crucial role in data security by preventing applications from accessing data outside their specified sandbox without explicit permission [36]

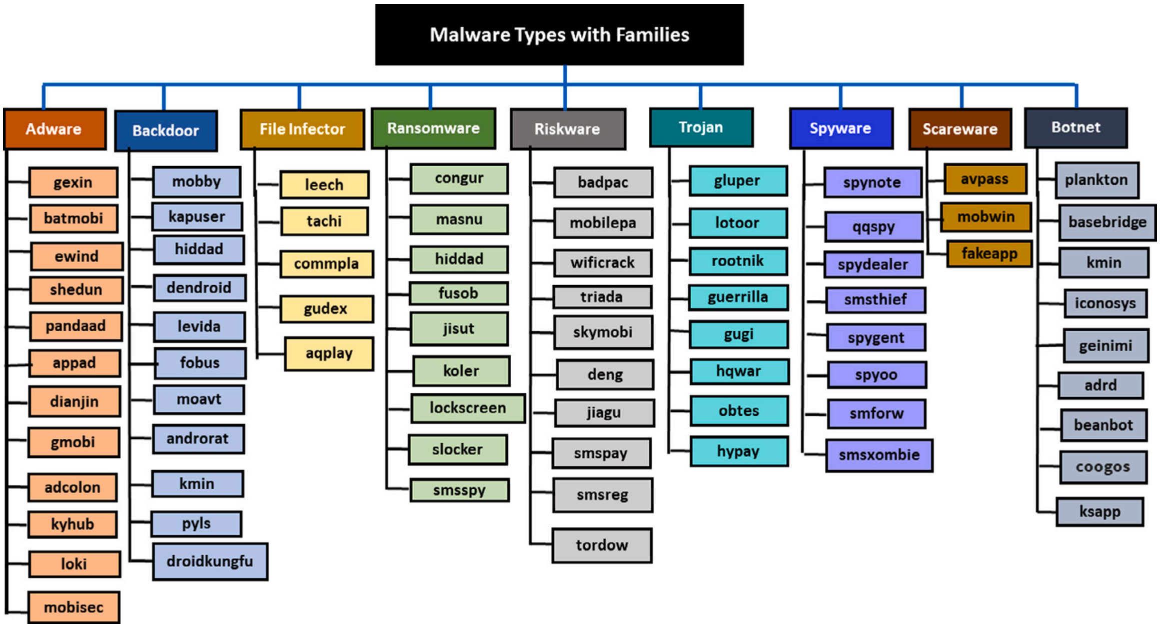


Fig. 3. Common malware types with families [40,41].

- Application Security: iOS adopts a layered approach to application security and provides developers with security APIs and frameworks to integrate robust security features directly into their applications. This includes secure storage, encryption, and secure network communications. Besides, the application review process by Apple assures that all applications available on the App Store adhere to strict security and privacy guidelines that protect users from malware and privacy breaches [37].
- Network & Internet Security: iOS restricts application access to background communication using encryption protocols, firewalls, and private addresses. This ensures secure data transmission and prevents network-related attacks [38].

2.2. What is mobile malware

Mobile malware is any malicious code that either behaves as a separate application or sometimes injects malicious codes into the source code of legitimate apps and converts them to mobile malware. Whatever the way is, the ultimate goal of mobile malware is stealing personal information, reading/writing call logs/contacts, recording call conversations, or gaining full access control to the mobile phone or locking that smartphone completely. In the last decade, mobile application usage has increased tremendously, providing various facilities for business and personal use. Malicious applications often disguise themselves as benign applications [39]. Though Android and iOS apply permission-based access control to regulate which application can access which resources, applications still declare some additional permissions that do not directly relate to the core functionality. These are called over-privileged applications. After that, users grant all the permissions required to run an application without explicitly checking, intensifying the risk of private information leakage. This way, malicious applications are successfully installed on the user's mobile and execute harmful activities.

2.3. Mobile malware categories

Understanding the different types of malware is vital to invent new solutions and algorithms. This section discusses some common categories of mobile malware, including Adware, Backdoor, Ransomware, Riskware, Trojans, Spyware, and more, together with the detection algorithms found in the existing articles. Fig. 3 shows malware categories with their families according to [40,41]

Adware

Adware performs malicious activities by showing flashing and lucrative advertisements on the user's mobile screen. When a user clicks on those malicious advertisements, the malicious application developer generates a certain revenue. Suresh et al. [42] proposed a machine-learning-based detection algorithm to detect Android adware. They used static and dynamic features for this classification and obtained 85% accuracy when considering only static features and 76% accuracy when considering only dynamic features. Then they combined static and dynamic features for classification and got 84% accuracy. Ndagi et al. [43] performed a comparative analysis of adware classification using different machine-learning algorithms. They found Random Forest was the best classifier among several, which provides an accuracy of 98% and a false positive rate of 1.7%. Bagui et al. [44] analyzed the individual adware families using the dataset CICAndMal2017. They used dynamic features for the classification algorithm in their research.

Backdoor

Backdoor attacks open hidden doors for attackers to enter smartphones. In this attack, the attackers can control the device remotely without having the mobile phones on hand. Backdoor attacks become successful by bypassing the authentication processes and getting access to smartphones. Attackers are very intelligent in embedding malicious codes in legitimate applications, and these malicious codes are executed only when a certain condition is fulfilled. Sometimes backdoor attack is linked with other malware-type attacks [45]. When a mobile user clicks on any advertisement link, some malicious codes are embedded into the system. The main purpose of a backdoor attack is to receive/record call logs, send/receive SMS, steal personal information, use space in memory, and many more. Cho et al. [46] developed a backdoor attack to unlock Android phones without the user's knowledge. Yao et al. [47] proposed a solution for identifying real-time passive backdoor behaviors by attackers. Li et al. [48] again proposed another backdoor attack to beat machine-learning-based algorithms, and they were successful for 99% out of a total of 750 malware samples to bypass all the authentication mechanisms. Zhang et al. [49] developed a transferable backdoor attack that targets CNN models for malware detection.

File infector

File infector is a type of malware that is found to be attached to the APK files. The information on how to run or execute an application is described in the APK files. The malicious applications are automatically executed once those malicious APK files are installed. The phone setting changes, stealing device ID and IMEI number are the target of the attackers [50]. To the best of our knowledge, we have not found any detection and mitigation techniques for file infector malware in recent studies.

Ransomware

Ransomware is a type of malware where mobile phones are locked by attackers with a key. After paying the attacker a certain amount, they send the key to unlock the phone. However, in some cases, users might not unlock or get access to mobile phones even after paying the ransom amount, consequently losing all data stored in the phone. According to New York Times report [51], over 900,000 mobile phones are infected due to the ScarePackage ransomware family. Another report [52] showed that 150 countries were the victim of WannaCry ransomware, which caused 35000 mobile devices to be compromised. Gharib et al. [53] proposed a solution using both static and dynamic analysis named DNA-Droid to mitigate ransomware attacks. Sharma et al. [54] analyzed permissions used by benign and malicious applications and made a comparison between those to propose a solution against this attack. Amer et al. [55] developed a model for early prediction of ransomware attacks based on behavioral analysis using static and dynamic features like API calls, system calls, and permissions. The model overcomes challenges by clustering related features and transforming them into simplified cluster classes because of massive feature sizes and complex associations. The LSTM model, along with ensemble machine learning models, is used to classify reformed API and system call sequences and Android permissions, and it showed notable performance and accuracy in detecting ransomware attacks.

Riskware

Riskware malware behaves like a legitimate application that gets installed as a malware application on the user's device. It can change the phone's settings and then redirect the user to the malicious advertisement page. Stealing personal information and changing network settings are the goals of the attackers. Mahdavifar et al. [56] proposed a solution to detect riskware using deep learning. They performed dynamic analysis for the experiment and evaluated their model on the CICMalDroid2020 dataset. They also compared the performance of their model with other machine-learning algorithms including label-propagation (LP). Their model achieved an F1 Score of 97.84% and a false positive rate of 2.76% which are higher than LP.

Scareware

Scareware is a type of malware used to deceive users through fake advertisements. This malware pretends to be a benign application that tries to convince users that installing it will protect their phones. However, users are convinced by the advertisement, and after installing this, it starts to reveal malicious activities such as stealing the user's location information. Hamid et al. [57] proposed a framework that can detect a couple of malware such as scareware, ransomware, and goodware. Bagui et al. [58] proposed a solution for classifying different malware families, including scareware, using Naïve Bayes, Decision Tree, and OneR algorithm and observed that Decision Tree achieved the highest accuracy.

Spyware

Spyware is a type of malicious software that gets installed in users' devices without the knowledge of the user. This malware tries to gain control over the network settings, data, camera, and microphone of a mobile device. The information they receive from this malicious access is sent to the attacker's database. Certainly, the attackers use the information for illicit purposes, including bank fraudulent transactions and ransomware attacks. Pierazzi et al. [59] used both conventional machine-learning and deep-learning algorithms to make a comparative study between spyware and other malware. Salih et al. [60] created a spyware application so that they could understand the malicious activities of spyware. Qabalin et al. [61] proposed a new dataset and then applied some machine-learning algorithms to detect spyware. The dataset includes network traffic data during spyware installation and was validated using the random forest classification algorithm that showcases good accuracy in both binary and multi-class classifications.

Table 1
Recent studies for the detection of each malware category.

Authors	Malware category	Features used	Learning model used/Developed	Performance metric
Suresh et al. [42] 2019	Adware	AndroidManifest.xml, Network traffic flow	Random forest	Accuracy of 84%
Ndagi et al. [43] 2019	Adware	–	Random forest	Accuracy of 98%
Bagui et al. [44] 2021	Adware	Network traffic flow	Gradient boosting XGB	AUC score of 98.71%
Mohammed et al. [69] 2022	Adware	Network traffic flow	AdStop	Accuracy of 98.02%
Moreira et al. [70] 2023	Ransomware	PE headers	CNN	Accuracy of 93.73%
Farnood et al. [71] 2021	Ransomware	Opcodes, API calls, System resources	RansomCare	Accuracy of 99.24%
Almohaini et al. [72] 2021	Ransomware	Reports from antivirus engines, Behavior and functionalities	Hybrid	Accuracy of 100%
Su et al. [73] 2018	Ransomware	Permission, Text, System operation calls	Ensemble	Accuracy of 99.98%
Alsoghyer et al. [74] 2019	Ransomware	API packages' calls	API-RDS	Accuracy of 97%
Faris et al. [75] 2020	Ransomware	API calls and permissions	SSA-KELM	AUC score of 98%
Gera et al. [76] 2021	Ransomware	API calls, System calls, Network flows, Permissions	J48	Recall of 99%
Manavi et al. [77] 2022	Ransomware	PE header of executable files	Proposed method	Recall of 95.11%
Hamid et al. [57] 2017	Scareware, Ransomware, Goodware	Lock detected, Text detected, Text score, Encryption detected	K-Means clustering	Accuracy of 98.12%

Trojan

Trojan is a type of malware that also acts like a benign application. Trojans can take the form of other categories of malware, such as trojan-spy, trojan-sms, and trojan-dropper [62]. Abualola et al. [62] designed a malicious Trojan application that changes the notification settings of WhatsApp, SMS, and Facebook Messenger without the user's knowledge. For this purpose, the authors used the notification listeners service. Nurul et al. [63] proposed a solution to detect trojans. They used several machine-learning algorithms like Random Forest, J45, and Naive Bayes for this detection. Ullah et al. [64] observed the correlation among different dynamic features to design another framework called TrojanDetector which detects Trojan malware.

Botnets

Botnet malware gets installed in the mobile phone remotely by a botmaster and users do not have an idea or knowledge about this malware installation in their mobile phone. Eslahi et al. [65] explained this attack and DroidDream, which can have root access to the mobile phone and how it installs other malicious applications. Hashim et al. [66] outlined a systematic review of botnet attacks. They described how reverse engineering and static analysis analyze the reasons behind botnet attacks. Suleiman et al. [67] proposed a solution for the detection of botnet attacks using deep learning and observed that their solution outperformed conventional machine-learning algorithms. In their research, they used the ISCX dataset [68] and analyzed 6802 mobile applications, where 1929 applications were botnet type and the rest were benign applications.

Tables 1 and 2 summarize existing work on specific Malware Type Classification.

Researchers keep developing efficient algorithms to detect and prevent malware attacks, and machine learning-based solutions have achieved promising results. As new malicious applications are getting uploaded in application markets every day and installed in users' mobile phones, it is essential to develop machine-learning algorithms to detect these newly spreading malware applications. However, conventional machine-learning algorithms are prone to different security threats, as discussed in Section 2.4. Conventional machine-learning algorithms refer to machine-learning models that learn different patterns from the data and then make a prediction for new unseen data. These algorithms use a centralized server to store the training data and train the global model, while privacy or security measures are not included in the algorithms.

2.4. Security threats against machine-learning

Machine-learning is a rapidly progressing field that has already proven its tremendous performance in the detection field. However, the attackers try various ways of attacking techniques to hamper the model's characteristics, ultimately affecting the prediction performance and accuracy of the model. The researchers should address the domain of security attacks on conventional machine-learning algorithms so that efficient security measures are developed to preserve the privacy and security of the training data, the efficiency of the model, and the reliability of the predictions. The existing security risks in this field have been divided into three types and discussed below.

1. Data Poisoning: A data poisoning attack is a type of adversarial attack where the attackers insert malicious data samples into the machine-learning models intentionally so that the training dataset is exploited. The attacker observes the vulnerabilities of the model during the testing period. As the malicious samples are mixed with the training data, the accuracy and performance

Table 2
Recent studies for the detection of each malware category.

Authors	Malware category	Features used	Learning model used/Developed	Performance metric
Khurram et al. [78] 2011	Adware	Sequences of opcodes	K-Nearest neighbor	AUC score of 94.9%
Fallah et al. [79] 2019	Adware, Ransomware, Scareware	Network traffic	Naïve Bayes	Adware (F1 measure of 74%), Ransomware (F1 measure of 73%), Scareware (F1 measure of 72%)
Bagui et al. [58] 2022	Scareware	Number of bytes initially sent back and forth, Packet size	Decision Tree	Accuracy of 79.5%
Pierazz et al. [59] 2020	Spyware	Permission, Structure, Network	Ensemble Late Fusion (ELF)	Accuracy of 98.2%
Qabalin et al. [61] 2022	Spyware	Network traffic	Random forest	Binary classification (Accuracy of 79%) and Multi-class classification (Accuracy of 77%)
Nurul et al. [63] 2019	Trojan	System call	Random forest	Accuracy of 81.2%
Li et al. [80] 2022	Trojan	Permissions	Voting Algorithm	Accuracy of 98.3%
Ullah et al. [64] 2022	Trojan	App-APIs, App-Services, Network Flow, System Call	SVM	Accuracy of 96.64%
Diyana et al. [81] 2020	Trojan	Request permission, API call, Service receiver, Network traffic	KNN	Accuracy of 97.83%
Suleiman et al. [67] 2021	Botnets	Permissions, API calls, Intents	DNN-2-layer-100	Accuracy of 99.1%
Hojjatinia et al. [82] 2020	Botnets	Permissions	CNN	Accuracy of 97.2%
Mohammad et al. [83] 2020	Botnets	Network traffic	J48	Accuracy of 85%
Wadi et al. [84] 2021	Botnets	Permissions	Random forest	Accuracy of 97.9%
Yerima et al. [85] 2022	Botnets	Permissions and intents	Extra trees forest	Accuracy of 96%

of the model are highly affected. In this case, the model starts making incorrect predictions for the new unseen data. The attackers try to control the predictions of the model in a way that the malicious data can be predicted as their desired class. However, it is difficult to detect this attack because it almost behaves like a legitimate one [86]. Additionally, backdoor attacks can also be executed by the attackers through data poisoning attacks [87]. Yerlikaya et al. [88] analyzed the performance of six machine-learning algorithms against data poisoning attacks using four datasets. They applied distance-based label-flipping and random label-flipping to execute the data poisoning attacks. Distance-based label Flipping selects data points for label flipping based on their distance from the decision boundary and targets those likely to impact the accuracy of the model significantly. Random Label Flipping changes the labels of randomly selected data points in the training dataset that potentially degrade the performance of the model by introducing incorrect labels.

2. Model Stealing: The attackers select different chosen inputs with confidence levels and output labels to steal machine-learning models. This attack is also called model extraction attack [86]. After stealing the model, attackers insert malicious data into the training data, and then again, they re-train the model with these malicious data, which has a great negative impact on the performance of the model's prediction. Reith et al. [89] showed that using only 854 queries that cost $0.09, the machine-learning model can be stolen. They selected SVM and SVR models for the experiment that were trained using publicly available machine-learning datasets. Lee et al. [90] used the deceptive perturbations method for protecting the machine-learning models from model stealing attacks.
3. Model Inversion: A Model inversion attack can also hamper the privacy of the training data. In this attack, the attackers try to steal sensitive information from training data after querying the model [91]. They use a reverse-engineering technique on training data after feeding inputs to the model, and then the attackers analyze the outputs. The attackers gain access to the characteristics of the model, which were not supposed to be disclosed to any unauthorized people. Model inversion attack is very dangerous for those applications such as health care and finance, which use training data that contain sensitive information [86]. Fredrikson et al. [92] developed a model inversion attack that revealed confidence values to create prediction queries to the machine-learning models. They used decision tree and neural networks for the experiment.

3. Secure and privacy-preserving machine learning

After describing the vulnerabilities of conventional machine-learning algorithms to various security threats in Section 2.4, we emphasize the critical need to shift focus towards secure and privacy-preserving machine learning approaches. Researchers are now focusing on developing algorithms for privacy and security measures in machine-learning models. Together with showing good accuracy, algorithms should also follow some privacy and security steps to protect training data and models from security breaches. Gradually, secure machine-learning and privacy-preserving machine-learning algorithms have been developed in recent years. *Secure machine-learning (Secure ML)* in Section 3.1 and *Privacy-Preserving machine-learning (Privacy-Preserving ML)* in Section 3.2 are explained.

3.1. Secure ML

Inventing new techniques for securing machine-learning algorithms is an active research area because new threats are getting discovered frequently. So, researchers also need to come up with new techniques so that they can overcome these challenges. Some methods that can ensure the security of the machine-learning models are explained below.

3.1.1. Data encryption

Encryption algorithms can be used to protect machine-learning models from various security threats. Data encryption techniques are helpful in keeping the data confidential and using it for both training and testing purposes. As many machine-learning algorithms have to deal with sensitive data, the security of users' sensitive information will be preserved in this way if a data encryption method is applied. Inference attacks can also be prevented by using encryption mechanisms [93].

3.1.2. Data de-identification

De-identification is defined as the process of identifying and hiding users' sensitive information to protect their privacy and maintain integrity as much as possible [94]. After applying this technique, machine-learning algorithms will be secured from attacks because the attackers cannot identify and exploit the vulnerabilities in the model since the original data are replaced with some random values. However, the attackers still try to infer general data distributions. Data Poisoning attacks and Model inversion attacks can also be prevented by using this method because attackers cannot alter the data with other malicious data, which forces the model to make incorrect prediction decisions [95].

3.1.3. Data access controls

Data access control is another security measure that helps to protect machine-learning algorithms from security attacks. Machine-learning algorithms often have to deal with sensitive personal or financial information. Applying access control mechanisms is one of the core techniques for ensuring the security of sensitive information [96]. This technique helps to prevent unauthorized access by the attackers. Also, some insiders can harm the models or data through social engineering attacks [97]. People accessing some parts of the system can perform these social engineering attacks to compromise the model. With access control mechanisms, there are also other techniques to check the log activities that can detect these human attacks.

3.1.4. Data segmentation

In this method, data is divided into segments that have separate access privileges [98]. This can limit the amount of data that can be comprised. In this way, the attackers will also have access to reduced data access. In the case of sensitive data such as financial or personal, if data are segmented into smaller parts, then attackers will not be able to execute a successful attack. If an attacker somehow accesses to the data then the overall impact on the whole data will be less because the attacker will access the segmented portion of the data instead of the entire dataset. Baracaldo et al. [99] used data segmentation to mitigate poisoning attacks.

3.2. Privacy-preserving ML

Privacy-preserving machine-learning means applying the mechanism of privacy-preservation of the training data in machine-learning models. It uses a technique that helps prevent data leakage outside of a pre-defined scope of the algorithm. Data are distributed across multiple parties who do not want to share their sensitive data with others [100]. Differential privacy, k-anonymity, homomorphic encryption, federated learning, and more are some techniques that help achieve that goal if integrated with a machine-learning algorithm. These privacy-preserving machine-learning algorithms also possess the same characteristics of conventional machine-learning, like building a model on training and predicting a class for new data together with the properties of the privacy-preservation of the dataset.

3.2.1. Differential privacy

Differential privacy is a type of privacy-preserving machine-learning algorithm. This algorithm uses two techniques: centralized differential privacy (CDP) and local differential privacy (LDP). In CDP, the central node inserts random noise into the private data, whereas, in LDP, all the individual nodes add noise to their private data and then send it to the server to share. LDP ensures to provide greater privacy. However, it can affect the accuracy rate of the model [100]. Current research should focus on this issue and propose a new solution.

3.2.2. Homomorphic encryption

In the homomorphic encryption method, training and computations occur on the encrypted data. There is no need for decryption for the training purpose. While integrating with machine-learning algorithms, homomorphic encryption helps to perform computations on users' sensitive data without exposing it to unauthorized people. It allows both training and evaluating models on encrypted data. This is quite a helpful technique for privacy preservation as many cases, machine-learning models need to be built on users' sensitive and confidential data. There are two types, one is fully homomorphic encryption (FHE), and another is partially homomorphic encryption (PHE). In FHE, all the computations are performed on encrypted data, while in PHE, only specific calculation types such as addition or multiplication are allowed [101].

3.2.3. K-anonymity

In k-Anonymity, identity anonymization of the users' is applied to the dataset to hide their identity from exploitation. This technique is also one of the privacy-preserving techniques that help to preserve the privacy and confidentiality of users' sensitive training data after integrating it into the machine-learning model. Each data sample must be distinguishable from at least k-1 other data samples. Generalizing and suppressing are two ways to achieve k-anonymity, making it difficult to find a person from their data. In the case of age and income-related data, this method groups the dataset into different age ranges and income brackets to prevent personal identification while allowing other useful information extraction from the dataset. This method is helpful for dealing with sensitive data like medical, personal, and financial data [102,103].

3.2.4. Secure multi-party computation (SMPC)

In this approach, computations are divided into smaller sub-computations and then each party can perform sub-computations on each part jointly with their own private data [104]. Without disclosing their private input data samples to others, each party performs the computations on the functions. The primary focus is to keep the privacy of the input data from being accessible by others, and then they perform sub-computations separately on the function. The results of all the sub-computations are combined together, and then the final result is disclosed to all other parties. In this method, no party can access the other party's individual input data, which helps preserve the privacy of the users' confidential information [105].

4. Conventional machine-learning for mobile malware detection

The in-built application and security architecture of both Android and IoS, concept of mobile malware along with different malware categories are discussed in Sections 2.1–2.3 respectively. As mobile operating systems' built-in features are insufficient to protect them from malicious applications, many solutions have been proposed using conventional machine-learning/deep-learning algorithms in the last few years. Static analysis, dynamic analysis, and hybrid analysis have been found in recent papers for the detection of mobile malware. These analyses are performed on both benign and malicious applications to observe the differences between them. This section discusses some of the significant approaches to mobile malware detection using conventional machine-learning/deep-learning algorithms found in recent studies. Section 4.1 presents mobile malware detection approaches based on static analysis, Section 4.2 presents based on dynamic analysis and Section 4.3 describes the methods based on hybrid analysis.

4.1. Static analysis-based malware detection methods

In static analysis, the code of an application is analyzed without executing for the detection of mobile malware [106,107]. This type of analysis is faster and includes observing assembly code, binary code, and the source code. It tries to discover the patterns and signatures of malicious applications, such as function calls or strings of code. However, the static analysis does not help detect encrypted and dynamically loaded malicious code [108]. The static analysis executed only in special conditions identifies the malicious codes of applications. Though static analysis is efficient and faster than dynamic analysis, it has some limitations. Static analysis may not detect new malware correctly. There is a chance of producing false positives if the code of a benign application is similar to that of known malware [109]. The detailed description of the static features is explained in Section 2.1.1.

Researchers use different subsets of the expansive malware dataset and new malware emerges daily. For the effectiveness of the malware detection solution, it is necessary to train the model using the most recent malware dataset. For these reasons, conducting experiments using a unified dataset is not possible. Table 3 provides a summary of various studies on mobile malware detection using static analysis. The authors employed different features and learning models to analyze datasets consisting of a varying number of samples. For instance, Suarez-Tangil et al. [110] introduced an Android malware classification system focusing on fast, accurate detection and resilience against obfuscation. It utilized static analysis to extract obfuscation-invariant features and identified obfuscation methods used in malware. Their method achieved up to 99.82% accuracy with zero false positives for malware detection and 99.26% for family identification of obfuscated malware. Mat et al. [111] introduced a Bayesian probability model for Android malware detection based on permission features extracted via static analysis. It utilized a dataset of 10,000 samples from AndroZoo and Drebin databases, applied information gain and chi-square algorithms for feature selection, and achieved an accuracy rate of 91.1%. Their approach signified the effectiveness of permission-based features in identifying malware. Bayazit et al. [112] presented a deep learning-based Android malware detection system employing static analysis with RNN-based LSTM, BiLSTM, and GRU algorithms. They used the CICInvesAndMal2019 dataset, which included 8115 static features, and their approach demonstrated the BiLSTM model that outperformed other models with an accuracy rate of 98.85%. Their work highlighted the effectiveness of deep learning in detecting malware and classifying complex patterns within static data for accurate malware identification. Yang et al. [113] introduced a novel Android malware detection method called AMDASE, which leveraged API semantics extraction to improve malware detection. It focused on API clustering for semantic extraction and optimized call graphs to extract robust API contextual information. Feature vectors were generated by AMDASE for machine learning classification by summarizing APIs into cluster centers. Each approach presented in the recent papers followed distinct methodologies with their respective constraints. Despite showing robustness against malware evolution, their methods face challenges in processing efficiency and scalability due to their reliance on API semantics extraction in static analysis. Some approaches only focus on permission-based features, which are limited by the evolving nature of malware threats that make the potential for adversaries to manipulate permissions to evade detection. These limitations emphasize the ongoing challenges in Android malware detection and emphasize the need for continuous refinement of detection methodologies to address the dynamic and sophisticated nature of malware threats.

Table 3
Mobile malware detection using static analysis.

Authors	Features	Learning model used/Developed	Datasets	No. of samples	Accuracy
Sandeep et al. [114] 2019	APK files/Permissions	Random forest	Google Play Store, VirusShare	–	94.65%
Singh et al. [115] 2019	Permissions and API calls	Linear-Support Vector Machine (L-SVM)	Drebin	16,100	99.6%
Mat et al. [111] 2021	Permissions	Naïve Bayes	AndroZoo [116] and Drebin [117]	10,000	91.1%
Syrris et al. [109] 2021	Permissions, receivers, services and API calls	SVM	Drebin	1,29,013	99%
Akbar et al. [118] 2022	Permissions	SVM, Rotation forest, Naïve Bayes, Random forest	Google Play Store, Virus Share [119]	10,000	86.25–89.96%
Bayazit et al. [112] 2022	Permission	RNN-based LSTM, BiLSTM and GRU	CICInvesAndMal2019 [14]	8115	98.85%
Ibrahim et al. [120] 2022	Permissions, receivers, services and API calls	XGBoost	CICMalDroid2020 [121]	17,341	97.3%
Odat et al. [122] 2023 Chaudhary et al. [123] 2023 Ding et al. [124] 2023	Permissions Permissions and intents Bytecode image	Random forest CNN CNN	Malgenome CICInvesAndMal2019 Drebin	3798 1584 4962	98% 96.4% 95.1%
Nguyen1 et al. [125] 2023	Permissions, intents, services, and API calls	1D-CNN	Drebin & CICMaldroid2020	15,130 & 17,016	99.6%
Ariffin et al. [126] 2023	Permissions	KNN	Kaggle Dataset	399	95%
Bakır et al. [127] 2023	API calls, Application metadata, Manifest components	ANN-based Auto-encoder, CNN-based Auto-encoder, VGG19-based Auto-encoder	Drebin, Malgenome and Google Play	6000	98.33%
Yang et al. [113] 2024	API	AMDASE	Custom Dataset	84,604	96.7%

4.1.1. Obfuscation procedure of static features

The static features are obfuscated by the developers of the malicious applications to a certain level so it becomes difficult to reverse engineer. The common approaches for static feature obfuscation are discussed below.

1. Rename Identifier: Developers follow some naming rules and conventions while coding and usually follow similar ways for all the identifiers. Random strings are injected by the malware developers and renamed the identifier in a way that makes it difficult to understand the logic of the code and reverse engineer. Because of not adding many unrelated nodes to the control flow graph, it is difficult to detect the obfuscated identifier. Garcia et al. [128] proposed a method to detect malicious applications in spite of having many obfuscation techniques.
2. Insert Junk Code: Operation instructions (nop), additional register, and jumps can contain junk instructions and statements by malicious application developers, which increases the code's complexity and deletes the actual static features. Because of the limited influence of the junk code on data flow, normal data flow work is observed on source–sink [129].
3. Load Dynamic Code: Remote networks, source files contain additional Dalvik bytecode and native code, which can be loaded by the Android application. The DexClassLoader function is used to load dynamic code by the malware developers. Sometimes reflection calls can be originated by using lang/Ljava/reflect/Method, which makes it hard to find the malicious code. The sensitive functions of the applications can be used to determine whether it is benign or malicious. The dynamic calls are checked and determined whether it is malicious or benign from the (super control flow graph) sCFG, which Poeplau et al. [130] designed.

4.2. Dynamic analysis-based malware detection methods

Dynamic analysis is a malware detection technique that monitors the run-time behaviors of applications in either virtual or real environments. It has been promising because it has the capability to resist code transformation. In dynamic analysis, mobile malware detection is performed by analyzing the behavior of a program or application while it is running. Monitoring system calls, network traffic, and file system changes are a few approaches to dynamic analysis. These methods are more powerful than static analysis because they can identify malicious activity that may not be detected through static analysis, which observes the code of a program or files without executing it [131]. Emulation, Sandboxing, and Honeypots are some of the dynamic analysis techniques for malware detection. Safe and isolated environments are chosen to observe the dynamic analysis of malicious applications so that other benign applications on the device are not affected.

Table 4 presents a summary of studies focusing on mobile malware detection using dynamic analysis. The authors employed various features and learning models to analyze datasets consisting of different numbers of samples. For instance, Haq et al. [132]

Table 4
Mobile malware detection using dynamic analysis.

Authors	Features	Learning model used/Developed	Datasets	No. of samples	Accuracy
Mahindru et al. [136] 2017	APK files/Permissions	Naive Bayes (NB), Decision Tree, Random Forest (RF), Simple Logistic (SL), k-star	Android Botnet [137], DroidKin [138]	11,000	67.64–99.7%
Feng et al. [139] 2018	Permissions, API calls	Naive Bayes, KNN	AndroZoo & Drebin	24,019	94.5%
Haq et al. [132] 2021	Permissions, API calls	Convolutional neural networks, Bidirectional Long ShortTerm Memory (BiLSTM)	AndroZoo, Android Malware Dataset (AMD)	38,842	98.83–99.20%
Sihag et al. [133] 2021	System calls, binder calls	Deep neural network	AndroZoo & Drebin	13,533	98.08%
Li et al. [140] 2022	API calls, system calls, and the relationships between these calls	LSTM	VirusShare	43,007	97.31%
Bhat et al. [134] 2023	System calls, binders, and complex Android objects	Stacked ensemble (Logistic regression, Support vector machine, Decision Tree and, Naive Bayes as base and Random Forest as meta classifier)	CICMal-Droid2020	11,598	98.08%
Liu et al. [135] 2024	Graph nodes	SeGDroid	CICMal2020 and MalRadar [141]	16,595 and 2092	98.37%

proposed a dynamic and robust deep learning-based model for Android malware detection that used a hybrid architecture combining Convolutional Neural Networks (CNN) and Bidirectional Long Short-Term Memory (BiLSTM) networks. Their method emphasized efficiency in identifying persistent malware across various datasets and achieved superior performance in detection accuracy and time efficiency compared to recent studies. Sihag et al. [133] proposed a deep learning-based framework for Android malware detection using dynamic features. Their approach was distinguished by the dependence on behavioral characteristics obtained through dynamic analysis in an emulated environment which proved highly effective with a detection rate of 98.08% and an F-measure of 98.84%. Bhat et al. [134] introduced a system call-based Android malware detection approach utilizing ensemble machine learning techniques. Their approach used system calls, binders, and Android-specific composite behaviors to identify malicious applications. They employed feature selection to enhance classification efficiency and achieved notable success with a stacking ensemble method which resulted in a classification accuracy rate of 98.08%. Liu et al. [135] presented a novel Android malware detection method based on sensitive function call graph (FCG) learning. A graph pruning method was employed in their approach to focus on security-related API calls that used word2vec and centrality measures for node representation and graph embeddings through a Graph Convolutional Neural Network (GCN). SeGDroid achieved high detection performance with an F-score of 98% on malware detection and 96% on malware family classification, which emphasized the ability to capture semantic knowledge of malicious behavior effectively. However, these techniques have several limitations. Some approaches face challenges in scaling to process large volumes of applications due to computational complexity and time constraints and difficulties in detecting entirely new or evolving malware without periodic retraining or updates to include novel malware samples and techniques. Despite some methods being designed to resist obfuscation, sophisticated adversaries continuously develop new evasion techniques that can bypass current detection mechanisms.

4.3. Hybrid analysis-based malware detection methods

Hybrid analysis for mobile malware combines both static and dynamic analysis techniques. This technique has been proven to have greater accuracy in malware detection in recent studies. After selecting an application, it analyzes the code of that application. Then the application is executed in an isolated environment like sandboxing to protect other applications. As hybrid analysis uses both techniques together, this technique is considered more effective in malware detection than using those two techniques separately [142]. However, these methods may also be more resource-intensive and time-consuming because they use static and dynamic features together.

Table 5 provides a summary of studies on mobile malware detection using hybrid analysis. For instance, Huanran et al. [143] proposed a hybrid Android malware detection framework that integrated static and dynamic analysis techniques. They utilized permission sequences and object reference relationships to enhance detection accuracy. By adopting machine learning algorithms, their approach achieved an F1-measure of 97.5%, which demonstrated effectiveness in resisting permission abuse behaviors and obfuscation techniques. Mahdavifar et al. [144] introduced a semi-supervised deep learning framework for Android malware detection utilizing pseudo-label stacked auto-encoders (PLSAE). This hybrid approach combined static and dynamic analysis to prepare feature vectors and evaluated them in the CICMalDroid2020 dataset. It showed notable improvements over traditional

Table 5
Mobile malware detection using hybrid analysis.

Authors	Features	Learning model used/Developed	Datasets	No. of samples	Accuracy
Surendran et al. [147] 2020	API calls, permissions and system calls	Ridge regularized logistic regression	Drebin, Androzoo Google Play Store	3300	97%
Hadiprakoso et al. [148] 2020	Manifest permission, API call signatures, intent-filter, command signature, and binaries	XGB	Malware Genome, Drebin, CICMalDroid	35,000	99%
Lu et al. [149] 2020	API call, intents, permission, dynamic code loading, system call	DBN-GRU	Public malware sharing websites, Google Play store	13,298	96.82%
Dhalaria et al. [150] 2020	API call, intents, permission, Command Strings, Cryptographic Operations, Dynamic Permissions, Information Leaks, System Calls	Random Forest	Available in Github and Kaggle	3600	98.53%
Upadhayay et al. [151] 2021	Permissions, Network Traffic	SVM	Genome, Drebin, Koodous [152], Google Play store	4900	95.96%
Huanran et al. [143] 2022	Permissions, memory	TextCNN	Google play, VirusShare	21,708	99.8%
Mahdavifar et al. [144] 2022	Intents, Permissions, System calls, Composite behaviors, Basic binders	PLSAE	CICMalDroid2020	17,341	98.28%
Chaganti et al. [142] 2023	API call sequences and PE Headers	CNN	-	40,000	97%
Taher et al. [145] 2023	Command strings, API calls, intents, and permissions, dynamic permissions, cryptographic operations, system calls, and information leakage	DroidDetectMW	Drebin and CICAndMal2017 [153]	10,450	98.1%
Dhalaria et al. [146] 2024	-	MalDetect	Drebin and AndroMD	15,036 and 3547	98.57% and 97.90%

supervised and semi-supervised methods and achieved high accuracy and efficiency in utilizing a significant volume of unlabeled data alongside a smaller set of labeled samples. Taher et al. [145] proposed a hybrid model that integrated static and dynamic analysis for Android malware detection. Their proposed model used feature selection strategies to optimize the classification process and employed a neural network optimized with an enhanced Harris Hawks Optimization (HHO) algorithm. Their approach significantly improved the accuracy of malware detection and classification and highlighted the importance of hybrid analysis in combating sophisticated malware. Dhalaria et al. [146] focused on improving Android malware detection through a classifier fusion approach, using various machine learning techniques and ranking algorithms based on predictive error rates. Drebin and AndroMD datasets were used in their experiment to demonstrate the effectiveness, which achieved better performance than individual base classifiers and traditional ensemble learning techniques. Despite achieving high-performance scores, these approaches include several limitations, such as handling new or unknown malware without frequent retraining, the need for large and diverse datasets for training, and computational demands for deep learning models and dynamic analysis in real-time scenarios.

The articles we included in this section used conventional machine-learning algorithms that do have in-built integrated privacy preserving techniques or security measures in the design of the algorithms. Though conventional machine-learning algorithms do not include security and privacy-preservation techniques by default, several privacy-preserving algorithms such as federated learning, differential privacy, k-anonymity, or more can be integrated with these algorithms. The techniques for ensuring secure machine-learning and privacy-preserving machine-learning are discussed in Section 3 and Federated Learning in 5. The model needs to be trained using a large dataset containing users' sensitive information to detect mobile malware. So, it is very crucial to integrate privacy-preserving techniques in the malware detection algorithms, which not only show greater accuracy in malware detection but also preserve the confidentiality of the data.

5. Federated learning

Federated Learning [154] is a privacy-preserving machine learning technique where the devices train the local models with their own training samples; no other device can access those data. Section 3 explains other approaches for developing secure ML and privacy-preserving ML. In Federated Learning, the local models transfer the parameters only to the central server, which is needed to

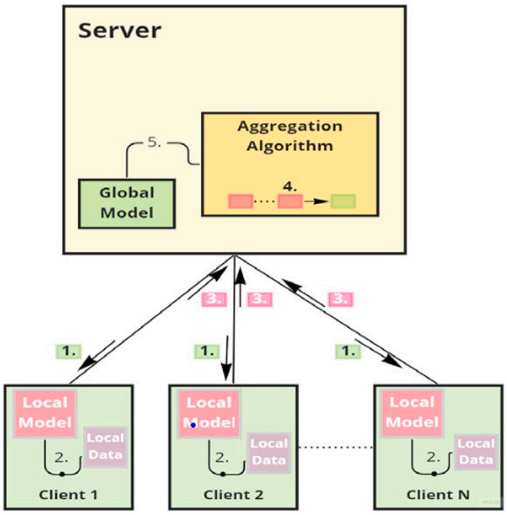


Fig. 4. General framework of FL [158].

create the global model. The global model is built by aggregating all the parameters sent by all the local devices. After aggregating, the global model is shared among all the devices to start the process again [155,156]. These processes continue until the desired accuracy is achieved by the algorithm [157]. Fig. 4 shows the general framework of Federated Learning (FL) [158]. In this method, the user's data privacy is preserved by not allowing the data to leave the devices. So, the centralized server for storing all the training data, like in the conventional machine-learning approach, is not followed here. This section describes the applications of FL in real-life scenarios along with the solution mechanisms of data and model heterogeneity in FL systems. Section 5.1 outlines the integration of FL approaches used for mobile malware detection along with the security threats and challenges. After that, Section 5.2 introduces different solution mechanisms from the literature for developing a secure FL framework.

There are considerable advantages to using FL over conventional machine learning. The privacy of the users' data is preserved because the data does not leave the local device. The local devices only share the parameters. The central server also does not know about the data, and it minimizes the chances of many security risks. As training is performed based on large and diverse data samples, the model's accuracy and generalization capability are increased.

Some applications already have been using the FL technique, such as natural language processing [159], image recognition [160], healthcare [161], the internet of things [162], intrusion detection systems [163], data mining [164] and many more. In the case of medical data, FL helps to perform training on large and diverse medical datasets from multiple hospitals where any hospital cannot access another hospital's data. Same to image recognition, FL can be used to train models on a large set of images within the local devices, and it preserves the users' privacy.

Qayyum et al. [165] used clustered FL for the automatic detection of COVID-19. They used two popular datasets to evaluate their solution. Their solution obtained favorable results for both datasets. According to their result, they achieved 16% and 11% higher F1-Score for using the FL-based technique than the conventional technique for those two datasets. Alamer et al. [166] developed a privacy-preserving FL approach called PPFL-SC using IoT devices for advanced malware detection. It incorporated group-oblivious signcryption to safeguard data privacy and employed an incentive mechanism to enhance IoT device collaboration. This methodology ensures the security and privacy of data while effectively detecting malware through collective learning from diverse IoT sources.

FL systems are prone to security threats where adversaries try to change the model parameters to hamper the accuracy of the predictions' decisions. FL systems are susceptible to data poisoning attacks, eavesdropping attacks, model inversion attacks, and others [167–169]. Researchers have been developing different algorithms to protect the FL systems from different cyber-attacks and increase the model's performance. FL is still a promising technique for several real-world applications, but securing the privacy of users' data is of high priority. As FL performs training collaboratively on large datasets [170], developing efficient techniques to detect those cyber attacks is of utmost importance. Section 5.2 discusses the solution mechanisms to prevent these security attacks of FL systems.

Addressing Data and Model Heterogeneity in FL Systems

Despite having lots of advantages of using an FL approach, it still has some challenges, such as Data and model heterogeneity [171]. As there are many clients and each client can have different types and formats of data, which creates data heterogeneity issues. To address data and model heterogeneity in the FL systems, Cao et al. [172] proposed a solution using Generative Adversarial Networks (GANs) within a personalized FL framework called PerFED-GAN. This method utilizes the power of GANs to generate synthetic data samples that are representative of the client's unique data distribution. The use of GANs addresses the issue of data heterogeneity and ensures the privacy of the client's data because synthetic samples can be shared without exposing the original data. Furthermore, the solution introduces a co-training mechanism that is designed to ensure that the synthetic data generated by the GANs remains aligned with the data distribution of the client. This mechanism considers both the diversity of the client's data and model architectures and ensures that the aggregated global model can generalize across the heterogeneity present within the FL system. After that, Wang et al. [173] incorporated data generators within the FL framework. In this method, training data is supplemented with a generative model, and its learning process and the storage of the supplementary data are performed on the server side. Optimizing the global model is constrained by the distance between the training global model and the aggregated global model's distribution to ensure effective learning without divergence between real and synthetic data. Li et al. [174] proposed a solution called DynamicNet that addressed data privacy in heterogeneous environments in the FL system by dynamically adjusting model aggregation weights and privacy budgets. In this method, privacy budgets are used to determine how much data can be shared without compromising individual privacy, and aggregation weights are used to influence the importance of data from each node during model updates. The solution is applied to mobile computing, where nodes may vary widely in terms of data quality and quantity along with computational power. Chen et al. [175] introduced a decoupled model architecture called FedCMD that differentiates between cloud and edge computing resources within an FL system. It uses contrastive learning techniques to customize model architectures for edge nodes that allow for personalized model updates. It enables the model to learn from differences between nodes and improve the ability to handle heterogeneous data. This solution not only increases model generalization ability across the network but also overcomes the challenges of data and model heterogeneity. Azimi-Abarghouyi et al. [176] proposed a hierarchical structure for organizing clients in an FL system. The method is combined with quantization techniques to manage communication and computation, and clients are grouped based on their similar data characteristics or computational resources. After that, clusters are formed, and localized model updates are made. This solution addresses statistical heterogeneity by confirming that model updates are more representative of the data distribution of the network.

5.1. FL for mobile malware detection

Privacy-preserving machine-learning approaches have already been used to detect mobile malware to protect users' sensitive information. FL techniques have been used with different machine-learning or deep-learning algorithms to detect mobile malware. It can not only detect malicious applications with higher accuracy but also preserve the privacy of the training data.

The model can be trained on large and diverse datasets captured from various sources using FL. As mobile malware detection requires access to sensitive data like system logs, user behavior data, and more, FL techniques play a vital role in this respect. This technique enables a decentralized approach where the raw data do not leave out of the local devices. The global model is developed by aggregating the parameter updates from local devices.

The following are the general reasons why to use FL in mobile malware detection based on papers [177–182].

Privacy: People store confidential information on mobile devices such as photos, personal information, bank account details, locations, addresses, different websites login credentials, and many more. While detecting mobile malware, the applications must access the users' devices. As FL systems do not allow leaving data out of the devices, the privacy of these sensitive data is preserved.

Data Diversity: Malicious developers continuously develop malicious applications and make the malware dataset large and diverse. Training machine-learning models with this diverse dataset is necessary to reduce false positives or negatives. FL helps in this process of training models with large and diverse datasets.

Real-time Detection: As the FL technique means the model will be trained locally based on the local data, it is more efficient to detect malware in a particular user's device based on its own data, ensuring real-time detection of mobile malware.

Resource Efficiency: It ensures resource efficiency because FL systems do not transfer data from devices to a central server. As only the parameters are transferred, it reduces the computational burden of the central server.

Table 6 shows the research work on Mobile Malware Detection using FL. For instance, Hsu et al. [183] used privacy-preserving FL to detect Android malware. API calls, permissions, and learned local models have been used in this paper to detect Android malware. In their experiment, FL has been used with a secure multi-party computation technique and SVM. The work used the dataset from the Opera Mobile Store. Privacy of users' data has been ensured in this framework. To the best of our knowledge, this was the first work on Android malware detection using an FL system. Gálvez et al. [177] proposed a framework called Less is More (LiM) for the detection and classification of Android malware using FL. The information about the other installed applications in the mobile phone is kept private by using FL. In this experiment, the authors used a semi-supervised ensemble algorithm to achieve higher accuracy in both detection and classification. The performance metric showed that the global model achieved an F1 Score of 95% and local clients achieved 1% false positive in more than 100 applications. They used a dataset consisting of 25,000 benign applications and 25,000 malicious applications. In the security analysis section, they mentioned that their model is robust and secure against poisoning and inference attacks. Jiang et al. [178] proposed a new Android malware classification framework named FedHGCDroid using FL, which detects and classifies Android malware in a privacy-preserved way. The authors used a convolutional neural network and a graph neural network for the experiment. Though integrating these two neural network algorithms could classify the Android malware with high accuracy, they introduced the concept of FL to allow collaborative training by several Android users. This can assure both privacy of the data and the capability of real-time detection. In this work, the Androzoo dataset was used and their work proved that this method was more adaptable and accurate in the detection and classification of Android malware than other existing solutions. Chaudhuri et al. [180] proposed a solution named FedAvg using FL for Android malware detection. In this approach, the global model is developed by merging all the updates from all the local models. This approach has the limitation that if there is any poor-performing local model, it can hamper the performance of the global model because this approach gives value equally to all the local models. To overcome this issue, the authors updated their solution using a dynamic weighted federated averaging (DW-FedAvg) technique in which the local models' weights are updated based on their performance.

Table 6
Mobile malware detection using FL.

Authors	Analysis method	Features	Learning model	Datasets	No. of samples	F1 score
Hsu et al. [183] 2020	Static	Permissions and API Calls	SVM	Opera Mobile Store	61,730	95.92%
Gálvez et al. [177] 2020	Static	Permissions, Activities, Services, Intents, Hardware components, Broadcast receivers	LiM	AndroZoo [116]	50,000	95%
Jiang et al. [178] 2022	Dynamic	Behavioral features	FedHGCDroid	AndroZoo	70,000	91.25%
Chaudhuri et al. [180] 2023	Dynamic	–	DW-FedAvg	Melgenome, Drebin, Kronodroid [184] and Tuandromd [185]	Between 3799 and 78,137	Between 97.63% and 99.18%
Mahindru et al. [179] 2023	Static	Permissions	DNNdroid	–	100,000	97.8%
Amjath et al. [186] 2024	Static	API calls and Opcodes	DotGAT	AndroZoo and Maldroid [121]	2500	90%

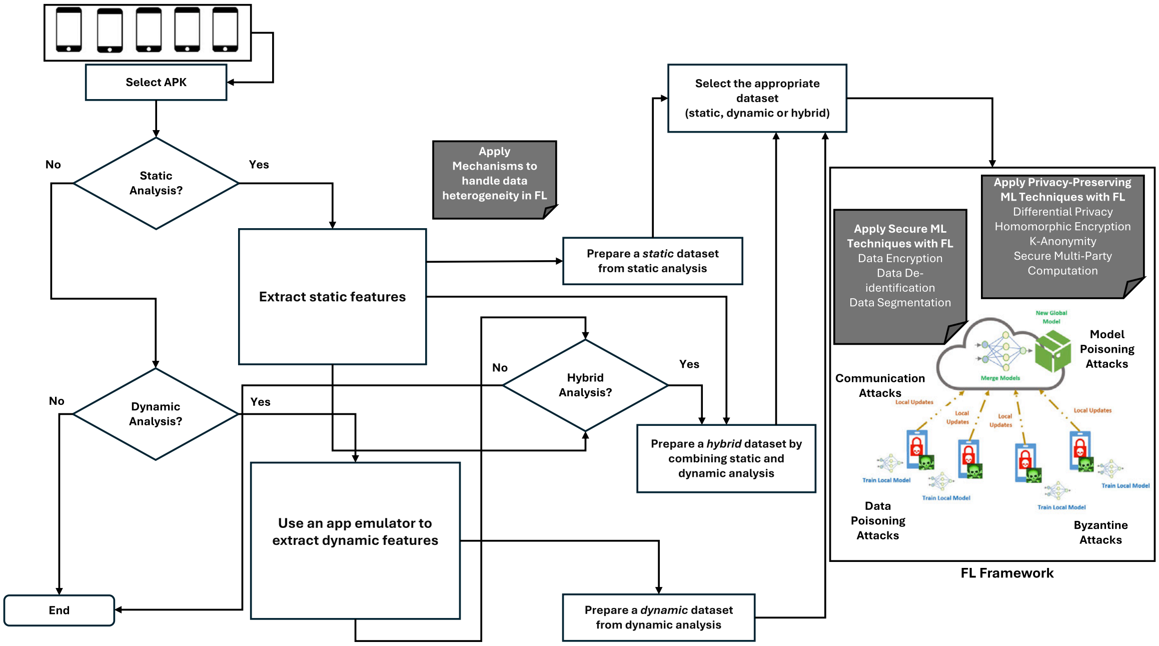


Fig. 5. Navigating security in FL for mobile malware detection.

In this experiment, Drebin, Tuandromd, Kronodroid, and Melgenome datasets were used, and the performance evaluation showed that this dynamic version outperformed the traditional one.

Security Threats in FL:

FL systems have found applications across real-world scenarios where training data is confidential and cannot be shared with external parties. Recent studies have identified several security threats to these systems. Fig. 5 highlights the critical need for integrating privacy-preserving and secure machine learning techniques with the FL framework in mobile malware detection. However, FL systems remain vulnerable to security threats even with these integrations. Section 5.2 discusses the strategies for addressing these vulnerabilities.

1. Data Leakage: The attackers try to infer private data after analyzing gradients of FL. This is called a data leakage attack and this attack can occur during the gradient-sharing process of FL. Xin et al. [187] developed an advanced data leakage attack called CAFE. They also proposed solutions to mitigate data leakage attacks.

2. Byzantine Attacks: According to recent studies, the FL system is prone to Byzantine attacks [188] where the malicious clients perform these attacks. This is one of the dangerous attacks because it can drop the accuracy of the model from 100% to 0% [189] by updating these parameters just opposite of the linear combination of other normal updates and the accuracy of the global model drops to 0%. The attackers try to perform this Byzantine attack either on training data or on parameters. If the attack is on training data then it is also called a data poisoning attack where the attackers manipulate the local training data which ultimately affects the global model's accuracy. Backdoor triggers, adding noise, and label flipping are some of the ways to attack training data. The attackers try to change the local parameters either on the model or gradient so that the global model receives the corrupted copy of parameters which also affects the accuracy of a global model.
3. Data Poisoning Attacks: In this attack, the model is modified or tampered with by the attackers. The attackers either insert malicious samples in the training set or tamper with the existing dataset by label-flipping. The accuracy of the model is affected by this attack [190,191]. Doku et al. [192] proposed a solution for mitigating data poisoning attacks for FL systems. Psychogyios et al. [193] introduced two types of data poisoning attacks in FL and also proposed a solution to mitigate this attack based on server-side clean-label training.
4. Model Poisoning Attacks: In this attack, the attackers manipulate the parameters of the chosen model and the global model starts behaving erroneously. This attack can create the chances of a backdoor attack [190]. According to [191], model poisoning attacks can be more effective than data poisoning attacks. The attackers often try the bit-flipping method to convert the model into the trojan-infected model [194]. As the local updates sharing to the central server and then the global model sharing with local clients are repetitive processes, attackers try to change the parameters of the model in between these processes.
5. Privacy Breaches: The privacy-preservation of training data is the primary target of the FL system. However, the attackers try to reveal sensitive information about the users from the local models or during the aggregation of updated parameters [195]. Chai et al. [196] proposed a framework using Hierarchical Blockchain-Enabled FL to mitigate this threat.
6. Communication Attacks: The procedures inside a FL system are iterative where the performance of the global model is improved after completion of each round. There are a huge number of communication messages that need to be exchanged between central servers and local clients. The convergence of FL is achieved after running a large number of iterations. During the exchange of communication messages, the communication channel is prone to many security attacks [188].

5.2. Secure FL

Uprety et al. [197] analyzed the effect of data poisoning attacks on the training samples such as label-flipping attacks. They proposed a method to mitigate this attack by observing each node's reputation scores using the beta probability distribution to identify and filter out malicious nodes and evaluated the model using the MNIST dataset. The approach enhances the robustness of FL models by improving the trustworthiness of contributions to the global model. Furthermore, it incorporates a blockchain-based network with FL to prevent denial of service attacks and ensure the privacy of the data and integrity of the system, which is an advantage of the method. However, this method has several challenges and drawbacks. It introduces computational overhead during the calculation and updating of reputation scores, which can affect the efficiency of the model. The method considered only one type of data poisoning attack since there are other types of adversarial attacks, which leads to the adaptability issue of the proposed solution to various attack vectors.

Unal et al. [198] proposed a solution to mitigate the FL attacks by integrating blockchain-based systems with FL in IoT-based networks. This integration enhances the privacy-preserving capabilities of FL by ensuring the integrity of the trained models through blockchain to mitigate model poisoning attacks. The approach used fuzzy hashing to detect anomalies in FL-trained models against poisoning attacks which added an additional layer of security. However, the solution introduces complexities for integrating blockchain technology, which could potentially lead to increased computational overhead and latency in processing IoT data. There is also a scalability challenge by the model because of the reliance on fuzzy hashing and maintenance of an immutable blockchain record.

Ali et al. [199] proposed a blockchain-based solution to improve the security of FL by showing IoT-based use cases in visualized and scattered FL environments. The advantages include enhanced privacy and security through decentralization that reduces the vulnerability to attacks and unauthorized access by distributing data across a network. Additionally, transparency and immutability of the blockchain network strengthen trust among participants. However, the solution has some drawbacks, such as scalability issues, extensive computational requirements, potential latency of the blockchain network and the complexity of implementing a blockchain network with federated learning and significant overheads in terms of energy consumption.

Houda et al. [200] proposed a solution called MiTFed that enables the development of a global model for intrusion detection jointly by several software-defined networks. They did not share any information regarding model training with one another. This work also proposed a framework using the SMPC technique for the secure aggregation of updates from the local models. In addition, a blockchain-based system was also adopted by the authors so that the collaboration can be done in a trustworthy and decentralized way. They evaluated their solution using the popular network security dataset NSL-KDD. MiTFed addressed the concerns of distributed denial-of-service (DDoS) attacks and the limitations of traditional intrusion detection systems (IDSs) that are prone to zero-day attacks due to their reliance on signature-based detection methods. However, the method introduces certain challenges and drawbacks because of the integration of complex structures such as FL, blockchain, and SMPC, which can lead to increased computational overhead and latency. Additionally, the effectiveness of the framework depends on the participation and collaboration of multiple SDN domains, which raises the concern of data privacy.

Liu et al. [201] proposed a solution named BFG for mitigating attacks in decentralized federating learning using blockchain, differential privacy, and Generative Adversarial Network. The advantages include because of utilizing blockchain, BFG removes the single point of failure issue inherent in central servers, which enhances the robustness and reliability of the system, and also, the chances of inference attacks were reduced by 26% in this solution. The integration of differential privacy introduces noise to the data, which hides the details of individual data samples and strengthens the privacy of the data. However, the method has some drawbacks, including significant computational resource requirements and performance overheads that can affect the speed of the learning process of the model.

6. Research challenges and future work

The field of mobile malware detection has several research gaps and future research directions; some of these key areas include:

- Scarcity of Extensive Datasets: The possibility of accurate malware detection depends on the use of real-world extensive malware datasets. With the increasing number of new malicious applications, training detection algorithms with real-world and complete datasets become very crucial. After gaining access to these real datasets by the researchers, they can build robust machine-learning algorithms capable of accurately detecting and classifying new malware.
- Advanced Bypassing Techniques: Malicious application developers are developing various algorithms to enable malware applications to bypass detection mechanisms and gain access to mobile devices. To address this concern, researchers should focus on this issue and propose new solutions that can detect these bypassing attempts by malware applications.
- Privacy and Security Concerns: As the accuracy of detecting and classifying mobile malware depends on training machine-learning models with large and diversified real-world datasets, it arises the issue of disclosing users' private information. Though there are already some techniques proposed in recent studies for ensuring privacy-preserving machine-learning, still it needs to access large data or parameters. Researchers should work on this issue and develop algorithms for mobile malware detection that can achieve higher accuracy while minimizing the need for a large collection of datasets.

In the past few years, the importance of mobile malware detection has increased significantly. This is due to the intent to replace laptops/computers with mobile phones for their convenience and the ability to use them for almost all purposes. Different conventional machine-learning algorithms were developed initially for the detection and classification of malware applications. These conventional machine-learning algorithms achieved higher accuracy and performance metrics in differentiating between benign and malicious applications, however, these algorithms fail to preserve the privacy of the users' data. Hence there are multiple techniques discovered by the researchers such as differential privacy, k-anonymity, FL, and more which ensure the confidentiality of users' data. To the best of our knowledge, only the FL technique has been integrated with machine-learning algorithms by researchers for the detection and classification of mobile malware while also preserving the privacy of the training dataset.

However, the FL system is prone to various security attacks such as data poisoning attacks, model poisoning attacks, inference attacks, and more. Researchers have proposed some security measures to mitigate FL attacks. Still, there are some drawbacks to each approach. Many papers, such as [197–201] have proposed solutions using blockchain. However, blockchain has several disadvantages, including high implementation costs, environmental impacts, storage problems, and many more. Moreover, while blockchain is integrated with FL to secure the FL environment, the blockchain network itself is prone to many security threats. The security measures to protect the blockchain network while integrating with FL have not been addressed in these papers. Future work should focus on resolving this issue. Furthermore, most of the papers have not evaluated the algorithms using real-world use cases for ensuring secure FL. The researchers should address this part to develop solutions that can effectively deal with real-world scenarios. Additionally, future work should include the development of new innovative other privacy-preserving algorithms to detect mobile malware with higher performance metrics.

7. Conclusion

With the ever-increasing usage of mobile devices, malware developers are developing various malicious applications to cause harm to devices or users. To overcome this threat, the primary goal is to detect mobile malware. There are many machine-learning and deep-learning algorithms have been developed by researchers. This paper gives a summary of recent conventional machine-learning algorithms for this detection. Since the performance of machine-learning models depends on the quality and diversified training data, this training procedure on a large dataset can lead to the exposure of private information of the users. This paper reviews some techniques in recent studies on preserving the privacy and security of users' confidential information while applying machine-learning algorithms. Furthermore, the paper discusses some challenges associated with preserving privacy and security in the context of mobile malware detection. It gives constructive directions for future work to develop more robust and secure algorithms to identify and detect mobile malware threats effectively.

CRediT authorship contribution statement

Faria Nawshin: Conceptualization, Literature search, Writing – original draft, Editing. **Radwa Gad:** Literature search, Writing – original draft, Editing. **Devrim Unal:** Conceptualization, Supervision, Review & editing. **Abdulla Khalid Al-Ali:** Supervision, Review & editing. **Ponnuthurai N. Suganthan:** Supervision, Review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

Open Access funding provided by the Qatar National Library. This study has been partially supported by an internal GA Grant from Qatar University.

References

[1] Ericcson. Ericsson mobility report. 2022, https://www.ericsson.com/4ae28d/assets/local/reports-papers/mobility-report/documents/2022/ericsson-mobility-report-november-2022.pdf. Accessed on: 2024-01-12.

[2] AppBrain. Number of android applications. 2022, https://www.appbrain.com/stats. Accessed on: 2023-12-10.

[3] Statista. Number of apps available in leading app stores as of 3rd quarter 2022. 2022, https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/. Accessed on: 2023-12-10.

[4] AVTest. Malware. 2022, https://www.av-test.org/en/statistics/malware/. Accessed on: 2023-12-13.

[5] Tangari G, Ikram M, Sentana IWB, Ijaz K, Kaafar MA, Berkovsky S. Analyzing security issues of android mobile health and medical applications. J Am Med Inform Assoc 2021;28(10):2074–84.

[6] He D, Li S, Li C, Zhu S, Chan S, Min W, Guizani N. Security analysis of cryptocurrency wallets in android-based applications. IEEE Netw 2020;34(6):114–9.

[7] Bergadano F, Boetti M, Cogno F, Costamagna V, Leone M, Evangelisti M. A modular framework for mobile security analysis. Inf Secur J: Glob Pers 2020;29(5):220–43.

[8] Weichbroth P, Łysik Ł. Mobile security: Threats and best practices. Mob Inf Syst 2020;2020:1–15.

[9] Garg S, Baliyan N. Comparative analysis of android and iOS from security viewpoint. Comp Sci Rev 2021;40:100372.

[10] Ciaramella G, Iadarola G, Martinelli F, Mercaldo F, Santone A. A model checking-based approach to malicious family detection in iOS environment. Procedia Comput Sci 2022;207:1981–91.

[11] Lin J, Liu B, Sadeh N, Hong JI. Modeling {users'} mobile app privacy preferences: restoring usability in a sea of permission settings. 2014, p. 199–212.

[12] Zarni Aung WZ. Permission-based android malware detection. Int J Sci Technol Res 2013;2(3):228–34.

[13] Kim T, Kang B, Rho M, Sezer S, Im EG. A multimodal deep learning method for android malware detection using various features. IEEE Trans Inf Forensics Secur 2018;14(3):773–88.

[14] Taheri L, Kadir AFA, Lashkari AH. Extensible android malware detection and family classification using network-flows and API-calls. In: 2019 international carnahan conference on security technology. ICCST, IEEE; 2019, p. 1–8.

[15] Feizollah A, Anuar NB, Salleh R, Suarez-Tangil G, Furnell S. Androdialysis: Analysis of android intent effectiveness in malware detection. Comput Secur 2017;65:121–34.

[16] Wang W, Gao Z, Zhao M, Li Y, Liu J, Zhang X. DroidEnsemble: Detecting android malicious applications with ensemble of string and structural static features. IEEE Access 2018;6:31798–807.

[17] Wu Q, Zhu X, Liu B. A survey of android malware static detection technology based on machine learning. Mob Inf Syst 2021;2021:1–18.

[18] Gadyatskaya O, Lezza A-L, Zhauniarovich Y. Evaluation of resource-based app repackaging detection in android. In: Secure IT systems: 21st nordic conference, nordSec 2016, Oulu, Finland, November 2-4, 2016. Proceedings 21. Springer; 2016, p. 135–51.

[19] Harris MA, Brookshire R, Patten K, Regan B. Mobile application installation influences: have mobile device users become desensitized to excessive permission requests. In: Proceedings of the twentieth Americas conference on information systems. AMCIS 2015, 2015, p. 13–5.

[20] Alenezi M, Almomani I. Abusing android permissions: A security perspective. In: 2017 IEEE Jordan conference on applied electrical engineering and computing technologies. AEECT, IEEE; 2017, p. 1–6.

[21] Sharmeen S, Huda S, Abawajy J, Hassan MM. An adaptive framework against android privilege escalation threats using deep learning and semi-supervised approaches. Appl Soft Comput 2020;89:106089.

[22] Wu D, Cheng Y, Gao D, Li Y, Deng RH. SCLib: A practical and lightweight defense against component hijacking in android applications. In: Proceedings of the eighth ACM conference on data and application security and privacy. 2018, p. 299–306.

[23] Chin E, Felt AP, Greenwood K, Wagner D. Analyzing inter-application communication in android. In: Proceedings of the 9th international conference on mobile systems, applications, and services. 2011, p. 239–52.

[24] Choi H, Kim Y. Large-scale analysis of remote code injection attacks in android apps. Secur Commun Netw 2018;2018.

[25] Cai Y, Tang Y, Li H, Yu L, Zhou H, Luo X, He L, Su P. Resource race attacks on android. In: 2020 IEEE 27th international conference on software analysis, evolution and reengineering. SANER, IEEE; 2020, p. 47–58.

[26] Ruggia A, Possemato A, Merlo A, Nisi D, Aonzo S. Android, notify me when it is time to go phishing. In: 2023 IEEE 8th European symposium on security and privacy. EuroS&P, IEEE; 2023, p. 1–17.

[27] Ito K, Hasegawa H, Yamaguchi Y, Shimada H. Detecting privacy information abuse by android apps from API call logs. In: Advances in information and computer security: 13th international workshop on security, IWSEC 2018, Sendai, Japan, September 3-5, 2018, proceedings 13. Springer; 2018, p. 143–57.

[28] Fedler R, Kulicke M, Schütte J. Native code execution control for attack mitigation on android. In: Proceedings of the third ACM workshop on security and privacy in smartphones & mobile devices. 2013, p. 15–20.

[29] Albanese D, Casolare R, Ciaramella G, Iadarola G, Martinelli F, Mercaldo F, Russodivito M, Santone A, et al. StegWare: A novel malware model exploiting payload steganography and dynamic compilation. In: ICISSP. 2023, p. 741–8.

[30] Developer. App metadata. 2023, https://developer.apple.com/documentation/appstoreconnectapi/app_store/app_metadata. Accessed on: 2023-12-05.

[31] Developer. Requesting access to protected resources. 2023, https://developer.apple.com/documentation/uikit/protecting_the_user_s_privacy/requesting_access_to_protected_resources. Accessed on: 2023-12-05.

[32] Localization I. Requesting access to protected resources. 2018, https://medium.com/@guerrix/info-plist-localization-ad5daaea732a. Accessed on: 2023-12-05.

[33] Inc. G. Platform architecture. 2020, https://developer.android.com/guide/platform. Accessed on: 2023-12-05.

[34] Ng B. Android security overview. 2019, https://medium.com/@boshng95/android-security-overview-7386022ad55d. Accessed on: 2023-12-02.

[35] Khan S, Yusuf A, Haider M, Thirunavukkarasu K, Nand P, Rahmani MKI. A review of android and iOS operating system security. In: 2022 ASU international conference in emerging technologies for sustainability and intelligent systems. ICETSIS, IEEE; 2022, p. 67–72.

[36] Mohamed I, Patel D. Android vs iOS security: A comparative study. In: 2015 12th international conference on information technology-new generations. IEEE; 2015, p. 725–30.

[37] Thiel D. iOS application security: The definitive guide for hackers and developers. No Starch Press; 2016.

[38] Security AP. Network security overview. 2021, https://support.apple.com/en-qa/guide/security/sec79afd0274/web. Accessed on: 2023-11-25.

[39] Guo C, Xu J, Liu L, Xu S. MalDetector-using permission combinations to evaluate malicious features of android app. In: 2015 6th IEEE international conference on software engineering and service science. ICSESS, IEEE; 2015, p. 157–60.

[40] Kaur G, Lashkari AH. Understanding android malware families (UAMF) – the foundations (article 1). 2021, https://www.itworldcanada.com/blog/understanding-android-malware-families-uamf-the-foundations-article-1/441562. Accessed on: 2023-10-13.

[41] Fiky AHE, Shenawy AE, Madkour MA. Android malware category and family detection and identification using machine learning. 2021, arXiv preprint arXiv:2107.01927.

[42] Suresh S, Di Troia F, Potika K, Stamp M. An analysis of android adware. J Comput Virol Hack Tech 2019;15(3):147–60.

[43] Ndagi JY, Alhassan JK. Machine learning classification algorithms for adware in android devices: A comparative evaluation and analysis. In: 2019 15th international conference on electronics, computer and computation. ICECCO, 2019, p. 1–6. http://dx.doi.org/10.1109/ICECCO48375.2019.9043288.

[44] Bagui S, Benson D. Android adware detection using machine learning. Int J Cyber Res Educ (IJCRE) 2021;3(2):1–19.

[45] Daeef AY, Al-Naji A, Chahl J. Features engineering for malware family classification based API call. Computers 2022;11(11):160.

[46] Cho J, Cho G, Hyun S, Kim H. Open sesame! Design and implementation of backdoor to secretly unlock android devices. J Internet Serv Inf Secur 2017;7(4):35–44.

[47] Yao Y, Zhu L, Wang H. Real-time detection of passive backdoor behaviors on android system. In: 2018 IEEE conference on communications and network security. CNS, 2018, p. 1–9. http://dx.doi.org/10.1109/CNS.2018.8433190.

[48] Li C, Chen X, Wang D, Wen S, Ahmed ME, Camtepe S, Xiang Y. Backdoor attack on machine learning based android malware detectors. IEEE Trans Dependable Secure Comput 2022;19(5):3357–70. http://dx.doi.org/10.1109/TDSC.2021.3094824.

[49] Zhang Y, Feng F, Liao Z, Li Z, Yao S. Universal backdoor attack on deep neural networks for malware detection. Appl Soft Comput 2023;143:110389.

[50] Kaur G, Lashkari AH. Understanding android malware families: file infector and potentially unwanted applications (article 6). 2021, https://www.itworldcanada.com/blog/understanding-android-malware-families-file-infector-and-potentially-unwanted-applications-article-6/455415. Accessed on: 2023-11-03.

[51] Perlroth N. Android phones hit by 'ransomware'. 2014, https://archive.nytimes.com/bits.blogs.nytimes.com/2014/08/22/android-phones-hit-by-ransomware/. Accessed on: 2023-10-10.

[52] News S. Cyberattack has hit at least 74 countries'. 2017, https://news.sky.com/video/cyberattack-has-hit-at-least-74-countries-10874889. Accessed on: 2023-10-04.

[53] Gharib A, Ghorbani A. Dna-droid: A real-time android ransomware detection framework. In: International conference on network and system security. Springer; 2017, p. 184–98.

[54] Sharma S, Kumar R, Krishna CR. RansomAnalysis: The evolution and investigation of android ransomware. In: Proceedings of international conference on ioT inclusive life (ICIIL 2019), NITTTR Chandigarh, India. Springer; 2020, p. 33–41.

[55] Amer E, El-Sappagh S. Robust deep learning early alarm prediction model based on the behavioural smell for android malware. Comput Secur 2022;116:102670.

[56] Mahdavifar S, Abdul Kadir AF, Fatemi R, Alhadidi D, Ghorbani AA. Dynamic android malware category classification using semi-supervised deep learning. In: 2020 IEEE intl conf on dependable, autonomic and secure computing, intl conf on pervasive intelligence and computing, intl conf on cloud and big data computing, intl conf on cyber science and technology congress. DASC/piCom/cBDCom/cyberSciTech, 2020, p. 515–22. http://dx.doi.org/10.1109/DASC-PICom-CBDCom-CyberSciTech49142.2020.00094.

[57] Hamid IRA, Khalid NS, Abdullah NA, Ab Rahman NH, Wen CC. Android malware classification using K-means clustering algorithm. In: IOP conference series: materials science and engineering. Vol. 226, IOP Publishing; 2017, 012105.

[58] Bagui S, Brock H. Machine learning for android scareware detection. J Inf Technol Res (JITR) 2022;15(1):1–15.

[59] Pierazzi F, Mezzour G, Han Q, Colajanni M, Subrahmanian V. A data-driven characterization of modern android spyware. ACM Trans Manage Inf Syst (TMIS) 2020;11(1):1–38.

[60] Salih HM, Mohammed MS. Spyware injection in android using fake application. In: 2020 international conference on computer science and software engineering. CSASE, IEEE; 2020, p. 100–5.

[61] Qabalin MK, Naser M, Alkasassbeh M. Android spyware detection using machine learning: A novel dataset. Sensors 2022;22(15):5765.

[62] Abualola H, Alhawai H, Kadadha M, Otrok H, Mourad A. An android-based trojan spyware to study the notificationlistener service vulnerability. Procedia Comput Sci 2016;83:465–71.

[63] Aminuddin NI, Abdullah Z. Android trojan detection based on dynamic analysis. Adv Comput Intell Syst 2019;1(1):1–7.

[64] Ullah S, Ahmad T, Buriro A, Zara N, Saha S. TrojanDetector: A multi-layer hybrid approach for trojan detection in android applications. Appl Sci 2022;12(21):10755.

[65] Eslahi M, Salleh R, Anuar NB. Mobots: A new generation of botnets on mobile devices and networks. In: 2012 international symposium on computer applications and industrial electronics. ISCAIE, IEEE; 2012, p. 262–6.

[66] Hashim HA-B, Mohd Saudi M, Basir N. A systematic review analysis of root exploitation for mobile botnet detection. Adv Comput Commun Eng Technol 2016;113–22.

[67] Yerima SY, Alzaylaee MK, Shajan A. Deep learning techniques for android botnet detection. Electronics 2021;10(4):519.

[68] Canadian Institute for Cybersecurity. Android botnet dataset. 2014, https://www.unb.ca/cic/datasets/android-botnet.html. Accessed on: 2023-12-07.

[69] Alani MM, Awad AI. AdStop: Efficient flow-based mobile adware detection using machine learning. Comput Secur 2022;117:102718.

[70] Moreira CC, Moreira DC, de Sales Jr CdS. Improving ransomware detection based on portable executable header using xception convolutional neural network. Comput Secur 2023;130:103265.

[71] Faghihi F, Zulkernine M. RansomCare: Data-centric detection and mitigation against smartphone crypto-ransomware. Comput Netw 2021;191:108011.

[72] Almohaini R, Almomani I, AlKhayer A. Hybrid-based analysis impact on ransomware detection for android systems. Appl Sci 2021;11(22):10976.

[73] Su D, Liu J, Wang X, Wang W. Detecting android locker-ransomware on chinese social networks. IEEE Access 2018;7:20381–93.

[74] Alsoghyer S, Almomani I. Ransomware detection system for android applications. Electronics 2019;8(8):868.

[75] Faris H, Habib M, Almomani I, Eshtay M, Aljarah I. Optimizing extreme learning machines using chains of salps for efficient android ransomware detection. Appl Sci 2020;10(11):3706.

[76] Gera T, Singh J, Mehbodniya A, Webber JL, Shabaz M, Thakur D. Dominant feature selection and machine learning-based hybrid approach to analyze android ransomware. Secur Commun Netw 2021;2021:1–22.

[77] Manavi F, Hamzeh A. Ransomware detection based on PE header using convolutional neural networks.. ISeCure 2022;14(2):181–92.

[78] Shahzad RK, Lavesson N, Johnson H. Accurate adware detection using opcode sequence extraction. In: 2011 sixth international conference on availability, reliability and security. IEEE; 2011, p. 189–95.

[79] Fallah S, Bidgoly AJ. Benchmarking machine learning algorithms for android malware detection. Jordanian J Comput Inf Technol 2019;5(3):216–29.

[80] Li G. Research on smartphone trojan detection based on the wireless sensor network. Secur Commun Netw 2022;2022.

[81] Dehkordy DT, Rasoolzadegan A. DroidTKM: Detection of trojan families using the KNN classifier based on manhattan distance metric. In: 2020 10th international conference on computer and knowledge engineering. ICCKE, IEEE; 2020, p. 136–41.

[82] Hojjatinia S, Hamzenejadi S, Mohseni H. Android botnet detection using convolutional neural networks. In: 2020 28th Iranian conference on electrical engineering. ICEE, IEEE; 2020, p. 1–6.

[83] Rasheed MM, Faieq AK, Hashim AA. Android botnet detection using machine learning. Ingén Syst Inf 2020;25(1):127–30.

[84] Alqatawna J, Ala'M A-Z, Hassonah MA, Faris H, et al. Android botnet detection using machine learning models based on a comprehensive static analysis approach. J Inf Secur Appl 2021;58:102735.

[85] Yerima SY, Bashar A. A novel android botnet detection system using image-based and manifest file features. Electronics 2022;11(3):486.

[86] Xue M, Yuan C, Wu H, Zhang Y, Liu W. Machine learning security: Threats, countermeasures, and evaluations. IEEE Access 2020;8:74720–42.

[87] Cinà AE, Grosse K, Demontis A, Biggio B, Roli F, Pelillo M. Machine learning security against data poisoning: Are we there yet? Computer 2024;57(3):26–34.

[88] Yerlikaya FA, Bahtiyar Ş. Data poisoning attacks against machine learning algorithms. Expert Syst Appl 2022;208:118101.

[89] Reith RN, Schneider T, Tkachenko O. Efficiently stealing your machine learning models. In: Proceedings of the 18th ACM workshop on privacy in the electronic society. 2019, p. 198–210.

[90] Lee T, Edwards B, Molloy I, Su D. Defending against machine learning model stealing attacks using deceptive perturbations. 2018, arXiv preprint arXiv:1806.00054.

[91] Al-Rubaie M, Chang JM. Privacy-preserving machine learning: Threats and solutions. IEEE Secur Priv 2019;17(2):49–58.

[92] Fredrikson M, Jha S, Ristenpart T. Model inversion attacks that exploit confidence information and basic countermeasures. In: Proceedings of the 22nd ACM SIGSAC conference on computer and communications security. 2015, p. 1322–33.

[93] Thambiraja E, Ramesh G, Umarani DR. A survey on various most common encryption techniques. Int J Adv Res Comput Sci Softw Eng 2012;2(7):226–33.

[94] Yadav S, Ekbal A, Saha S, Bhattacharyya P. Deep learning architecture for patient data de-identification in clinical records. In: Proceedings of the clinical natural language processing workshop. clinicalNLP, 2016, p. 32–41.

[95] Löbner S, Tronnier F, Pape S, Rannenberg K. Comparison of de-identification techniques for privacy preserving data analysis in vehicular data sharing. In: Proceedings of the 5th ACM computer science in cars symposium. 2021, p. 1–11.

[96] Zhou L, Su C, Li Z, Liu Z, Hancke GP. Automatic fine-grained access control in SCADA by machine learning. Future Gener Comput Syst 2019;93:548–59.

[97] Salahdine F, Kaabouch N. Social engineering attacks: A survey. Future Internet 2019;11(4):89.

[98] Hiter S. What is data segmentation? 2021, https://www.datamation.com/security/data-segmentation/. Accessed on: 2023-10-16.

[99] Baracaldo N, Chen B, Ludwig H, Safavi JA. Mitigating poisoning attacks on machine learning models: A data provenance based approach. In: Proceedings of the 10th ACM workshop on artificial intelligence and security. 2017, p. 103–10.

[100] Xu R, Baracaldo N, Joshi J. Privacy-preserving machine learning: Methods, challenges and directions. 2021, arXiv preprint arXiv:2108.04417.

[101] Sun X, Zhang P, Liu JK, Yu J, Xie W. Private machine learning classification based on fully homomorphic encryption. IEEE Trans Emerg Top Comput 2018;8(2):352–64.

[102] Vijayarani S, Tamilarasi A, Sampoorna M. Analysis of privacy preserving k-anonymity methods and techniques. In: 2010 international conference on communication and computational intelligence. INCOCCI, IEEE; 2010, p. 540–5.

[103] Friedman A, Wolff R, Schuster A. Providing k-anonymity in data mining. VLDB J 2008;17:789–804.

[104] Kim W, Seok J. Privacy-preserving collaborative machine learning in biomedical applications. In: 2022 international conference on artificial intelligence in information and communication. ICAIIC, IEEE; 2022, p. 179–83.

[105] Zhao C, Zhao S, Zhao M, Chen Z, Gao C-Z, Li H, Tan Y-a. Secure multi-party computation: theory, practice and applications. Inform Sci 2019;476:357–72.

[106] Amro B. Malware detection techniques for mobile devices. 2018, arXiv preprint arXiv:1801.02837.

[107] Milosevic N, Dehghantanha A, Choo K-KR. Machine learning aided android malware classification. Comput Electr Eng 2017;61:266–74.

[108] Arshad S, Shah MA, Wahid A, Mehmood A, Song H, Yu H. SAMADroid: A novel 3-level hybrid malware detection model for android operating system. IEEE Access 2018;6:4321–39. http://dx.doi.org/10.1109/ACCESS.2018.2792941.

[109] Syrris V, Geneiatakis D. On machine learning effectiveness for malware detection in android os using static analysis data. J Inf Secur Appl 2021;59:102794.

[110] Suarez-Tangil G, Dash SK, Ahmadi M, Kinder J, Giacinto G, Cavallaro L. DroidSieve: Fast and accurate classification of obfuscated android malware. CODASPY '17, New York, NY, USA: Association for Computing Machinery; 2017, p. 309–20. http://dx.doi.org/10.1145/3029806.3029825.

[111] Mat SRT, Ab Razak MF, Kahar MNM, Arif JM, Firdaus A. A Bayesian probability model for android malware detection. ICT Express 2022;8(3):424–31.

[112] Bayazit EC, Sahingoz OK, Dogan B. A deep learning based android malware detection system with static analysis. In: 2022 international congress on human-computer interaction, optimization and robotic applications. HORA, IEEE; 2022, p. 1–6.

[113] Yang H, Wang Y, Zhang L, Cheng X, Hu Z. A novel android malware detection method with API semantics extraction. Comput Secur 2024;137:103651.

[114] HR S. Static analysis of android malware detection using deep learning. In: 2019 international conference on intelligent computing and control systems. ICCS, 2019, p. 841–5. http://dx.doi.org/10.1109/ICCS45141.2019.9065765.

[115] Singh AK, Jaidhar C, Kumara M. Experimental analysis of android malware detection based on combinations of permissions and API-calls. J Comput Virol Hack Tech 2019;15(3):209–18.

[116] AndroZoo. AndroZoo. 2023, https://androzoo.uni.lu/. Accessed on: 2023-12-24.

[117] DREBIN. The drebin dataset. 2023, https://www.sec.tu-bs.de/∼danarp/drebin/download.html. Accessed on: 2023-12-24.

[118] Akbar F, Hussain M, Mumtaz R, Riaz Q, Wahab AWA, Jung K-H. Permissions-based detection of android malware using machine learning. Symmetry 2022;14(4):718.

[119] VIRUSSHARE. VIRUSSHARE. 2023, https://www.secrepo.com/Datasets%20Description/PE_malware/VirusShare.html. Accessed on: 2023-12-24.

[120] İbrahim M, Issa B, Jasser MB. A method for automatic android malware detection based on static analysis and deep learning. IEEE Access 2022;10:117334–52.

[121] Mahdavifar S, Alhadidi D, Ghorbani AA. Effective and efficient hybrid android malware classification using pseudo-label stacked auto-encoder. J Netw Syst Manage 2022;30:1–34.

[122] Odat E, Yaseen QM. A novel machine learning approach for android malware detection based on the co-existence of features. IEEE Access 2023;11:15471–84.

[123] Chaudhary M, Masood A. RealMalSol: real-time optimized model for android malware detection using efficient neural networks and model quantization. Neural Comput Appl 2023;35(15):11373–88.

[124] Ding Y, Zhang X, Hu J, Xu W. Android malware detection method based on bytecode image. J Ambient Intell Humaniz Comput 2023;14(5):6401–10.

[125] Nguyen C-D, Khoa NH, Doan KN-D, Cam NT. Android malware category and family classification using static analysis. In: 2023 international conference on information networking. ICOIN, IEEE; 2023, p. 162–7.

[126] Ariffin NAM, Casinto HP. Android malware detection using permission based static analysis. J Adv Res Appl Sci Eng Technol 2023;33(3):86–97.

[127] Bakır H, Bakır R. DroidEncoder: Malware detection using auto-encoder based feature extractor and machine learning algorithms. Comput Electr Eng 2023;110:108804.

[128] Garcia J, Hammad M, Pedrood B, Bagheri-Khaligh A, Malek S. Obfuscation-resilient, efficient, and accurate detection and family identification of android malware. Tech. rep., 202, Department of Computer Science, George Mason University; 2015.

[129] Zhang X, Breitinger F, Luechinger E, O'Shaughnessy S. Android application forensics: A survey of obfuscation, obfuscation detection and deobfuscation techniques and their impact on investigations. Forensic Sci Int: Digit Invest 2021;39:301285.

[130] Poeplau S, Fratantonio Y, Bianchi A, Kruegel C, Vigna G. Execute this! analyzing unsafe and malicious dynamic code loading in android applications. In: NDSS. Vol. 14, 2014, p. 23–6.

[131] Yan P, Yan Z. A survey on dynamic mobile malware detection. Softw Qual J 2018;26(3):891–919.

[132] Haq IU, Khan TA, Akhunzada A. A dynamic robust DL-based model for android malware detection. IEEE Access 2021;9:74510–21. http://dx.doi.org/10.1109/ACCESS.2021.3079370.

[133] Sihag V, Vardhan M, Singh P, Choudhary G, Son S. De-LADY: Deep learning based android malware detection using dynamic features. J Internet Serv Inf Secur 2021;11(2):34–45.

[134] Bhat P, Behal S, Dutta K. A system call-based android malware detection approach with homogeneous & heterogeneous ensemble machine learning. Comput Secur 2023;130:103277.

[135] Liu Z, Wang R, Japkowicz N, Gomes HM, Peng B, Zhang W. SeGDroid: An android malware detection method based on sensitive function call graph learning. Expert Syst Appl 2024;235:121125.

[136] Mahindru A, Singh P. Dynamic permissions based android malware detection using machine learning techniques. In: Proceedings of the 10th innovations in software engineering conference. ISEC '17, New York, NY, USA: Association for Computing Machinery; 2017, p. 202–10. http://dx.doi.org/10.1145/3021460.3021485.

[137] Abdul Kadir AF, Stakhanova N, Ghorbani AA. Android botnets: What urls are telling us. In: Network and system security: 9th international conference, NSS 2015, New York, NY, USA, November 3-5, 2015, proceedings 9. Springer; 2015, p. 78–91.

[138] Gonzalez H, Stakhanova N, Ghorbani AA. Droidkin: Lightweight detection of android apps similarity. In: International conference on security and privacy in communication networks: 10th international ICST conference, secureComm 2014, Beijing, China, September 24-26, 2014, revised selected papers, part i 10. Springer; 2015, p. 436–53.

[139] Feng P, Ma J, Sun C, Xu X, Ma Y. A novel dynamic android malware detection system with ensemble learning. IEEE Access 2018;6:30996–1011. http://dx.doi.org/10.1109/ACCESS.2018.2844349.

[140] Li C, Lv Q, Li N, Wang Y, Sun D, Qiao Y. A novel deep framework for dynamic malware detection based on API sequence intrinsic features. Comput Secur 2022;116:102686.

[141] Wang L, Wang H, He R, Tao R, Meng G, Luo X, Liu X. MalRadar: Demystifying android malware in the new era. Proc ACM Meas Anal Comput Syst 2022;6(2):1–27.

[142] Chaganti R, Ravi V, Pham TD. A multi-view feature fusion approach for effective malware classification using deep learning. J Inf Secur Appl 2023;72:103402.

[143] Wang H, Zhang W, He H. You are what the permissions told me! android malware detection based on hybrid tactics. J Inf Secur Appl 2022;66:103159.

[144] Samaneh M, Dima A, Ghorbani AA. Effective and efficient hybrid android malware classification using pseudo-label stacked auto-encoder. J Netw Syst Manage 2022;30(1):22.

[145] Taher F, AlFandi O, Al-kfairy M, Al Hamadi H, Alrabaee S. DroidDetectMW: A hybrid intelligent model for android malware detection. Appl Sci 2023;13(13):7720.

[146] Dhalaria M, Gandotra E. MalDetect: A classifier fusion approach for detection of android malware. Expert Syst Appl 2024;235:121155.

[147] Surendran R, Thomas T, Emmanuel S. A TAN based hybrid model for android malware detection. J Inf Secur Appl 2020;54:102483.

[148] Hadiprakoso RB, Kabetta H, Buana IKS. Hybrid-based malware analysis for effective and efficiency android malware detection. In: 2020 international conference on informatics, multimedia, cyber and information system. ICIMCIS, IEEE; 2020, p. 8–12.

[149] Lu T, Du Y, Ouyang L, Chen Q, Wang X. Android malware detection based on a hybrid deep learning model. Secur Commun Netw 2020;2020:1–11.

[150] Dhalaria M, Gandotra E. A hybrid approach for android malware detection and family classification. 2020, p. 174–88.

[151] Upadhayay M, Sharma A, Garg G, Arora A. Rpndroid: Android malware detection using ranked permissions and network traffic. In: 2021 fifth world conference on smart trends in systems security and sustainability. WorldS4, IEEE; 2021, p. 19–24.

[152] KOODOUS. KOODOUS. 2023, https://koodous.com/. Accessed on: 2023-12-11.

[153] Lashkari AH, Kadir AFA, Taheri L, Ghorbani AA. Toward developing a systematic approach to generate benchmark android malware datasets and classification. In: 2018 international carnahan conference on security technology. ICCST, IEEE; 2018, p. 1–7.

[154] Asad M, Moustafa A, Aslam M. CEEP-FL: A comprehensive approach for communication efficiency and enhanced privacy in federated learning. Appl Soft Comput 2021;104:107235.

[155] Li L, Fan Y, Tse M, Lin K-Y. A review of applications in federated learning. Comput Ind Eng 2020;149:106854.

[156] Li T, Sahu AK, Talwalkar A, Smith V. Federated learning: Challenges, methods, and future directions. IEEE Signal Process Mag 2020;37(3):50–60.

[157] Rieke N, Hancox J, Li W, Milletari F, Roth HR, Albarqouni S, Bakas S, Galtier MN, Landman BA, Maier-Hein K, et al. The future of digital health with federated learning. NPJ Digit Med 2020;3(1):119.

[158] Khan M, Glavin FG, Nickles M. Federated learning as a privacy solution - an overview. Procedia Comput Sci 2023;217:316–25. http://dx.doi.org/10.1016/j.procs.2022.12.227, URL https://www.sciencedirect.com/science/article/pii/S1877050922023055. 4th International Conference on Industry 4.0 and Smart Manufacturing.

[159] Lin BY, He C, Zeng Z, Wang H, Huang Y, Soltanolkotabi M, Ren X, Avestimehr S. Fednlp: A research platform for federated learning in natural language processing. 2021, arXiv preprint arXiv:2104.08815.

[160] Toldinas J, Venčkauskas A, Liutkevičius A, Morkevičius N. Framing network flow for anomaly detection using image recognition and federated learning. Electronics 2022;11(19):3138.

[161] Xu J, Glicksberg BS, Su C, Walker P, Bian J, Wang F. Federated learning for healthcare informatics. J Healthc Inform Res 2021;5:1–19.

[162] Zhang T, Gao L, He C, Zhang M, Krishnamachari B, Avestimehr AS. Federated learning for the internet of things: applications, challenges, and opportunities. IEEE Internet Things Mag 2022;5(1):24–9.

[163] Nguyen TD, Rieger P, Miettinen M, Sadeghi A-R. Poisoning attacks on federated learning-based IoT intrusion detection system. In: Proc. workshop decentralized IoT syst. secur.. 2020, p. 1–7.

[164] Yu B, Mao W, Lv Y, Zhang C, Xie Y. A survey on federated learning in data mining. Wiley Interdiscip Rev: Data Min Knowl Discov 2022;12(1):e1443.

[165] Qayyum A, Ahmad K, Ahsan M, Al-Fuqaha A, Qadir J. Collaborative federated learning for healthcare: Multi-modal COVID-19 diagnosis at the edge. IEEE Open J Comput Soc 2022;3(01):172–84. http://dx.doi.org/10.1109/OJCS.2022.3206407.

[166] Alamer A. A privacy-preserving federated learning with a secure collaborative for malware detection models using internet of things resources. Internet Things 2024;25:101015.

[167] Tolpegin V, Truex S, Gursoy ME, Liu L. Data poisoning attacks against federated learning systems. In: Computer security–ESORICs 2020: 25th European symposium on research in computer security, ESORICs 2020, Guildford, UK, September 14–18, 2020, proceedings, part i 25. Springer; 2020, p. 480–501.

[168] Zhou X, Xu M, Wu Y, Zheng N. Deep model poisoning attack on federated learning. Future Internet 2021;13(3):73.

[169] Bhagoji AN, Chakraborty S, Mittal P, Calo S. Analyzing federated learning through an adversarial lens. In: International conference on machine learning. PMLR; 2019, p. 634–43.

[170] Li L, Fan Y, Tse M, Lin K-Y. A review of applications in federated learning. Comput Ind Eng 2020;149:106854.

[171] Ding J, Tramel E, Sahu AK, Wu S, Avestimehr S, Zhang T. Federated learning challenges and opportunities: An outlook. In: ICASSP 2022-2022 IEEE international conference on acoustics, speech and signal processing. ICASSP, IEEE; 2022, p. 8752–6.

[172] Cao X, Sun G, Yu H, Guizani M. PerFED-GAN: Personalized federated learning via generative adversarial networks. IEEE Internet Things J 2022;3749–62.

[173] Wang X, Zhu T, Zhou W. Supplement data in federated learning with a generator transparent to clients. Inform Sci 2024;120437. http://dx.doi.org/10.1016/j.ins.2024.120437.

[174] Li Z, Duan M, Yu S, Yang W. DynamicNet: Efficient federated learning for mobile edge computing with dynamic privacy budget and aggregation weights. IEEE Trans Consum Electron 2024. http://dx.doi.org/10.1109/TCE.2024.3372696, 1–1.

[175] Chen X, Du T, Wang M, Gu T, Zhao Y, Kou G, Xu C, Wu. DO. Towards optimal customized architecture for heterogeneous federated learning with contrastive cloud-edge model decoupling. 2024, arXiv preprint arXiv:2403.02360.

[176] Azimi-Abarghouyi, Mohammad S, Fodor V. Quantized hierarchical federated learning: A robust approach to statistical heterogeneity. 2024, arXiv preprint arXiv:2403.01540.

[177] Gálvez R, Moonsamy V, Diaz C. Less is more: A privacy-respecting android malware classifier using federated learning. 2020, arXiv preprint arXiv:2007.08319.

[178] Jiang C, Yin K, Xia C, Huang W. FedHGCDroid: An adaptive multi-dimensional federated learning for privacy-preserving android malware classification. Entropy 2022;24(7):919.

[179] Mahindru A, Arora H. Dnndroid: Android malware detection framework based on federated learning and edge computing. In: Advancements in smart computing and information security: first international conference, ASCIS 2022, Rajkot, India, November 24–26, 2022, revised selected papers, part II. Springer; 2023, p. 96–107.

[180] Chaudhuri A, Nandi A, Pradhan B. A dynamic weighted federated learning for android malware classification. In: Soft computing: theories and applications: proceedings of soCTA 2022. Springer; 2023, p. 147–59.

[181] Rey V, Sánchez PMS, Celdrán AH, Bovet G. Federated learning for malware detection in iot devices. Comput Netw 2022;204:108693.

[182] Xia Q, Ye W, Tao Z, Wu J, Li Q. A survey of federated learning for edge computing: Research problems and solutions. High-Confid Comput 2021;1(1):100008.

[183] Hsu R-H, Wang Y-C, Fan C-I, Sun B, Ban T, Takahashi T, Wu T-W, Kao S-W. A privacy-preserving federated learning system for android malware detection based on edge computing. In: 2020 15th Asia joint conference on information security. AsiajCIS, 2020, p. 128–36. http://dx.doi.org/10.1109/AsiaJCIS50894.2020.00031.

[184] Guerra-Manzanares A, Bahsi H, Nõmm S. Kronodroid: Time-based hybrid-featured dataset for effective android malware detection and characterization. Comput Secur 2021;110:102399.

[185] TUANDROMD. TUANDROMD (tezpur university android malware dataset). 2023, https://www.archive.ics.uci.edu/dataset/855/tuandromd+(tezpur+university+android+malware+dataset). Accessed on: 2023-12-20.

[186] Amjath M, Henna S. Differential Privacy Preservation for Graph-based Federated Learning under Malware Attacks.

[187] Jin X, Chen P-Y, Hsu C-Y, Yu C-M, Chen T. CAFE: Catastrophic data leakage in vertical federated learning. Adv Neural Inf Process Syst 2021;34:994–1006.

[188] Benmalek M, Benrekia MA, Challal Y. Security of federated learning: attacks, defensive mechanisms, and challenges. Rev Sci Technol l'Inf-Sér RIA: Rev d'Intell Artif 2022;36(1):49–59.

[189] Shi J, Wan W, Hu S, Lu J, Zhang LY. Challenges and approaches for mitigating byzantine attacks in federated learning. In: 2022 IEEE international conference on trust, security and privacy in computing and communications. TrustCom, IEEE; 2022, p. 139–46.

[190] Chen Y, Gui Y, Lin H, Gan W, Wu Y. Federated learning attacks and defenses: A survey. 2022, arXiv preprint arXiv:2211.14952.

[191] Bhagoji AN, Chakraborty S, Mittal P, Calo S. Analyzing federated learning through an adversarial lens. In: International conference on machine learning. PMLR; 2019, p. 634–43.

[192] Doku R, Rawat DB. Mitigating data poisoning attacks on a federated learning-edge computing network. In: 2021 IEEE 18th annual consumer communications & networking conference. CCNC, IEEE; 2021, p. 1–6.

[193] Psychogyios K, Velivassaki T-H, Bourou S, Voulkidis A, Skias D, Zahariadis T. GAN-driven data poisoning attacks and their mitigation in federated learning systems. Electronics 2023;12(8):1805.

[194] Rakin AS, He Z, Fan D. Tbt: Targeted neural network attack with bit trojan. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020, p. 13198–207.

[195] Zhang J, Zhu H, Wang F, Zhao J, Xu Q, Li H, et al. Security and privacy threats to federated learning: Issues, methods, and challenges. Secur Commun Netw 2022;2022.

[196] Chai H, Leng S, Chen Y, Zhang K. A hierarchical blockchain-enabled federated learning algorithm for knowledge sharing in internet of vehicles. IEEE Trans Intell Transp Syst 2020;22(7):3975–86.

[197] Uprety A, Rawat DB. Mitigating poisoning attack in federated learning. In: 2021 IEEE symposium series on computational intelligence. SSCI, IEEE; 2021, p. 01–7.

[198] Unal D, Hammoudeh M, Khan MA, Abuarqoub A, Epiphaniou G, Hamila R. Integration of federated machine learning and blockchain for the provision of secure big data analytics for internet of things. Comput Secur 2021;109:102393. http://dx.doi.org/10.1016/j.cose.2021.102393.

[199] Ali M, Karimipour H, Tariq M. Integration of blockchain and federated learning for internet of things: Recent advances and future challenges. Comput Secur 2021;108(C):102355. http://dx.doi.org/10.1016/j.cose.2021.102355.

[200] Abou El Houda Z, Hafid AS, Khoukhi L. MiTFed: A privacy preserving collaborative network attack mitigation framework based on federated learning using SDN and blockchain. IEEE Trans Netw Sci Eng 2023;1985–2001.

[201] Liu W, He Y, Wang X, Duan Z, Liang W, Liu Y. BFG: privacy protection framework for internet of medical things based on blockchain and federated learning. Connect Sci 2023;35(1):2199951.