Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2024.0429000

A Review of The Recent Trends in Mobile Malware Evolution, Detection, and Analysis

SEETAH ALMARRI, ALANOUD BODOKHI, and MOUNIR FRIKHA
Department of Computer Networks and Communications, College of Computer Sciences and Information Technology, King Faisal University, Al-Ahsa 31982, Saudi Arabia

Corresponding author: Seetah Almarri (e-mail: 224108483@student.kfu.edu.sa).

This paper was funded by the Deanship of Scientific Research, Vice Presidency for Graduate Studies and Scientific Research, King Faisal University, Saudi Arabia under the [GRANT].

ABSTRACT The rapid rise of smartphones and mobile applications has fostered an environment conducive to the advancement of mobile malware, presenting considerable obstacles for both detection and mitigation efforts. This analysis delves into the latest developments in mobile malware, underscoring the shift from basic threats to more complex forms of attacks, including ransomware, spyware, and banking trojans. It discusses significant progress in malware detection techniques, particularly those utilizing behavioral analysis, artificial intelligence (AI), and machine learning (ML). The paper points out deficiencies in current detection methods and stresses the importance of hybrid approaches that integrate both static and dynamic analyses to tackle issues such as obfuscation, polymorphism, and encrypted threats. Furthermore, the research investigates the impact of collaborative threat intelligence sharing and AI-driven tools in improving malware detection and analysis. This systematic review offers a thorough framework for comprehending the progression of mobile malware, assessing existing detection strategies, and pinpointing future research and security initiatives.

INDEX TERMS Mobile Malware, Malware Detection, Malware Evolution

I. INTRODUCTION

WITH the growing use of smartphones for everything from texting to banking, mobile devices have become key targets for cybercriminals. This has led to a rise in mobile malware—malicious software specifically designed to attack mobile devices. Over the years, mobile malware has changed a lot, becoming more dangerous and harder to detect. In the early days, mobile malware was fairly simple and mostly caused minor issues. But today, it has become much more advanced, including types like ransomware, spyware, and banking trojans that can steal personal data, monitor users, or demand money. This evolution is driven by the growing popularity of smartphones, mobile apps, and the value of data stored on these devices. To keep up with these threats, cybersecurity experts are constantly improving the ways they detect and analyze mobile malware. While older methods, like detecting known malware signatures, are still used, they are often not enough to catch modern, fast-evolving threats. Newer techniques involve monitoring app behavior, using Artificial Intelligence (AI), and Machine Learning (ML) to spot unusual activities. At the same time, analyzing malware involves deeper methods like code analysis, running malware in a safe environment, and sharing threat information within the cybersecurity community. Understanding how mobile malware is changing and how detection and analysis methods are evolving is critical for staying secure. This introduction gives an overview of the key trends in mobile malware and highlights the tools and techniques used to defend against these threats.

Mobile malware is evolving rapidly, with cybercriminals adopting increasingly sophisticated tactics to target smartphones and other mobile devices. One major trend is the rise of advanced malware that uses encryption, code obfuscation, and polymorphism to evade detection by traditional security systems. Another significant trend is the growth of mobile ransomware, which locks users out of their devices or encrypts data until a ransom is paid, mirroring the success of ransomware in the desktop world. Similarly, spyware and stalkerware are becoming more prevalent, allowing attackers to monitor communications, track locations, and steal sensitive information without users' knowledge. Banking trojans also continue to pose a threat by mimicking legitimate apps to steal financial data. Additionally, adware—which bombards users with unwanted ads—is being used not only to disrupt user experience but also as a gateway to more harmful malware. App exploits are another rising trend, where vulnerabilities in popular apps are targeted to deliver malicious payloads, while smishing (phishing via SMS) tricks users into clicking harmful links [1]. The growing use of AI-powered malware enables attacks to adapt dynamically to their environment, making them harder to detect. Another concerning trend is the presence of fake apps in official app stores like Google Play and Apple's App Store, which appear legitimate but hide malicious capabilities. Finally, with the increasing interconnectivity of devices through the Internet of Things (IoT), mobile devices are also becoming a primary point of attack for hackers targeting IoT ecosystems. These trends indicate that mobile malware is becoming more advanced and diversified, requiring both users and security experts to stay vigilant and adopt more proactive security measures.

Mobile malware is evolving rapidly, with cybercriminals increasingly using sophisticated tactics to target smartphones and other mobile devices. One significant trend is the rise of advanced malware, which employs encryption, code obfuscation, and polymorphism to evade detection by traditional security systems. Additionally, mobile ransomware has become a growing threat, locking users out of their devices or encrypting data until a ransom is paid, much like its desktop counterpart. Spyware and stalkerware are also becoming more prevalent, allowing attackers to monitor communications, track locations, and steal sensitive information without the user's knowledge. Another trend is the increase in banking trojans, which imitate legitimate apps to steal financial data from users. Furthermore, adware is being used not only to flood devices with unwanted ads but also to introduce more dangerous malware. Exploiting vulnerabilities in popular apps to deliver malicious payloads is another rising trend, while smishing—phishing through SMS—continues to trick users into downloading malware [1]. AI-powered malware is also emerging, enabling attacks to dynamically adapt to their environments and making them harder to detect. Fake apps in official stores like Google Play and Apple's App Store, designed to look legitimate but harboring malicious capabilities, are increasingly common. Finally, with the rise of the IoT, mobile devices are now being used as entry points for attacks targeting interconnected IoT devices, further increasing the scope and impact of mobile malware. These trends illustrate the growing sophistication and diversification of mobile threats, necessitating enhanced security measures for both users and organizations [2].

Mobile malware refers to malicious software specifically designed to target mobile devices such as smartphones and tablets. With the rapid rise in the use of mobile devices for personal, financial, and business activities, the threat posed by mobile malware has grown significantly. Mobile devices often contain sensitive data, such as banking information, personal communications, and work-related documents, making them attractive targets for cybercriminals. The importance of detecting and analyzing mobile malware lies in the increasing sophistication of attacks and their potential to cause widespread damage. Mobile malware can steal sensitive data, control devices remotely, spy on users, and even lock them out of their own devices through ransomware. This makes it essential to have advanced detection methods and analysis tools to prevent the spread of malware and mitigate its impact.

The evolution of mobile malware began with relatively simple attacks designed to exploit vulnerabilities in early mobile operating systems. Initially, these attacks caused minor disruptions such as sending unauthorized text messages or causing devices to malfunction. However, with the advancement of mobile technology and the proliferation of mobile applications, mobile malware has grown more sophisticated. Today, mobile malware encompasses a wide range of threats, including spyware, ransomware, banking trojans, and adware. These threats can steal sensitive data, lock devices until a ransom is paid, or monitor user activity, causing significant financial and reputational damage. Industries, in particular, have felt the impact of mobile malware, with sectors like finance, healthcare, and e-commerce being prime targets due to the valuable data they handle. Consumers, on the other hand, are vulnerable to identity theft, financial fraud, and loss of personal data. The growing concern over mobile malware has led to a corresponding increase in the development of security measures. However, as mobile malware continues to evolve, detection and analysis techniques must keep pace to effectively combat these threats.

Given the evolving nature of mobile malware and the rapid development of new attack methods, it is essential to keep track of the latest trends and innovations in malware detection and analysis. Conducting a Systematic Literature Review (SLR) on this topic will provide an organized and thorough examination of current research, identify gaps in the literature, and highlight areas where new techniques are emerging. This review is critical to help security professionals and researchers stay informed about the most effective strategies for combating mobile malware.

- Comprehensive overview of mobile malware evolution: By tracing the history of mobile malware, this review will provide a detailed understanding of how these threats have changed over time.
- Analysis of detection techniques: This review will compare various detection methods, including signature-based detection, behavioral analysis, ML, and hybrid approaches, identifying their strengths and limitations.
- Identification of gaps in the literature: By analyzing current research, this review will identify gaps in existing studies, such as insufficient attention to certain types of malware or emerging platforms.
- Highlighting new trends: The review will discuss new trends, such as the increasing use of AI in detection, the challenges posed by encrypted malware, and the role of threat intelligence sharing.

In our paper, we will explore mobile malware in detail and the methods used to detect it. We start with an introduction that explains why the study is important and highlights the need to understand current trends in mobile malware. After that, we provide a background section that traces the evolution of mobile malware, focusing on key developments over time. Next, the methodology section explains how the SLR was carried out, describing the search process, data sources, and selection criteria following PRISMA guidelines. The subsequent sections review different detection techniques, comparing their strengths and highlighting gaps in current research. The paper concludes by discussing the findings, emphasizing the need for more advanced detection methods, like ML and hybrid approaches, to deal with the growing complexity of mobile malware. This structure offers a clear and thorough review of existing research, providing useful insights for security experts and researchers.

A. MOTIVATION

The motive for the paper arises from the significant increase in mobile device utilization and the increasing complexity of mobile malware threats. As smartphones and tablets become essential for personal, financial, and professional activities, they have become prime targets for cybercriminals. Malware, which used to be simple and easy to detect, has become more advanced. Now, it includes threats like spyware, ransomware, and banking trojans that take advantage of weaknesses in mobile operating systems, especially Android and iOS. Traditional methods like signature-based detection are not enough to handle the fast-changing nature of these attacks. Many of these attacks use advanced techniques like encryption, hiding their code, or constantly changing their structure to avoid being detected.

In response to these growing threats, this paper emphasizes the need for stronger methods to detect and analyze these threats. It focuses on the importance of using ML, AI, and behavior-based analysis to find malware that traditional methods miss. Additionally, this paper aims to fill gaps in existing research by analyzing the latest trends in mobile malware evolution and assessing current detection strategies. It also suggests new tools and techniques to improve malware detection and prevention. This aims to help security experts and researchers stay updated and better equipped to handle the ever-changing mobile malware landscape.

It is crucial to first look at how mobile malware has changed over time, both in terms of complexity and scope, in order to completely comprehend the current threat landscape. The following section establishes the context for assessing detection techniques and gives the essential background information on the development of mobile malware.

II. BACKGROUND

The rapid adoption of mobile devices has led to a significant increase in mobile malware threats, which have evolved both in sophistication and impact. As mobile phones and tablets become increasingly central to personal, financial, and business activities, they have also become prime targets for cybercriminals. The rise of complex mobile malware such as spyware, ransomware, and banking trojans has exposed vulnerabilities in mobile operating systems, particularly Android and iOS.

Traditional malware detection techniques, like signature-based methods, are proving insufficient to combat the fast-evolving landscape of mobile [2]. In response, new approaches such as behavioral analysis, ML, and AI-driven techniques have emerged, but their effectiveness and limitations remain under active investigation. Additionally, analyzing mobile malware requires more advanced methods, including static and dynamic analysis, reverse engineering, and collaboration across the cybersecurity industry for threat intelligence sharing.

Despite progress in detection and analysis techniques, challenges such as obfuscation, platform restrictions, and the rapid adaptation of malware persist. The problem lies in understanding the latest trends in mobile malware evolution, evaluating the effectiveness of current detection techniques, and analyzing the obstacles that hinder the comprehensive understanding and mitigation of these threats. Table 1 presents the main issues in recognizing mobile malware along with the respective recommendations to solve these problems. Every row in the table offers a specific threat, indicating its description that describes the nature of the threat and its effects on the Mobile security. Also, the proposed solutions column focuses on the possible strategies and technology to avoid these challenges. The present SLR focuses on the developments in the complexity of mobile malware, the ineffectiveness of previously utilized detection methods, the importance of adopting progressive, integrated, and dynamic approaches to improve mobile security systems.

With this basic knowledge of the development of mobile malware and the difficulties it poses, the following section describes our methodical approach to choosing and evaluating relevant research studies. By systematically reviewing the latest research, this paper seeks to provide a clear understanding of how mobile malware is adapting, the current state of detection technologies, and the gaps that still exist in the analysis and mitigation of mobile threats.

III. METHODOLOGY

The approach used in this paper is the Systematic Literature Review (SLR) [3], which is an approach developed to allow for a proper evaluation of the studies within a pre-defined structure. This method starts with the formulation of definite research questions and the subsequent inclusion of relevant studies, which would allow for the extraction of relevant information. The review is limited to works published within the years **2020–2024**. Moreover, the study selection procedure strictly follows the PRISMA 2020 guideline to provide accuracy, transparency, and reliability in the identification of studies related to the evolution of mobile malware detection and analysis. The common technique that is employed in the use of SLR comes under three steps: planning, conducting, and reporting. During the planning phase, the research questions are formulated, and, therefore, inclusion and exclusion criteria are set for the review. The search strings are then used

TABLE 1: Key Challenges in Mobile Malware Detection and Proposed Solutions

Challenge	Description	Proposed Solutions
Rapid Evolution and Sophistication of Malware	Mobile malware is evolving with new techniques like polymorphism and Advanced Persistent Threats (APTs), making it difficult to detect with traditional signature-based methods.	Implement behavioral analysis to detect anomalies, use ML and AI to adapt to evolving threats.
Use of Obfuscation and Encryption	Malware developers use obfuscation and encryption to hide code, making it difficult to analyze and detect malicious activities.	Combine static and dynamic analysis methods, along with advanced reverse-engineering techniques.
Platform-Specific Challenges	Android's open ecosystem makes it more vulnerable, while iOS presents challenges due to restricted access.	Use cloud-based security solutions for real-time updates, and develop platform-specific threat detection systems.
Social Engineering and Phishing Attacks	Attackers use phishing and malvertising to trick users into downloading malware or clicking malicious links.	Increase user awareness, deploy anomaly detection systems to track unusual user behavior.
Insufficient Traditional Detection Techniques	Signature-based detection is no longer effective against zero-day attacks or new strains of malware.	Employ hybrid detection methods combining static, dynamic, and ML-based analysis.
Challenges in Malware Analysis	Static analysis is ineffective against obfuscated malware, and dynamic analysis is resource-intensive.	Combine static and dynamic methods with AI and ML to improve detection accuracy.
Lack of Comprehensive Threat Intelligence Sharing	Fragmented collaboration among different platforms and regions slows down the global response to emerging threats.	Encourage global collaboration for real-time threat intelligence sharing across organizations.

to identify potential papers to include and to evaluate papers in terms of their relevance to the research goals. This rigorous process always guarantees the review offers a focused yet coherent synthesis of the literature.

A. PLANNING

Here, we incorporated information about the research questions and the criteria for inclusion and exclusion that were applied during the planning stage.

1) Research Questions

This SLR aims to explore the following key research questions:

1) How has mobile malware evolved in recent years, and what are the latest trends?
2) What are the most effective methods for detecting mobile malware, and how do they compare?
3) What challenges do researchers face in analyzing mobile malware, and how can they be addressed?

2) Inclusion and Exclusion Criteria

Due to the guidelines highlighted in this section, readers can easily follow the selection procedure and understand why only a few research publications were selected. Papers only with high quality and relevance were included in this SLR, based on the following inclusion and exclusion criteria.

- **Inclusion Criteria**
  - **Publication Date:** In the present SLR, papers published between 2020 and 2024 are considered to ensure it is up-to-date.
  - **Field Relevance:** This review only deals with the literature that focuses on the evolution of mobile malware detection and analysis.
  - **Language:** To make the information as clear and understandable as possible to all the readers, only papers in English were considered.
  - **Peer-reviewed:** To identify peer-reviewed papers, the types included were peer-reviewed journal articles and other academic forms like conference papers and technical papers.
  - **Full-text Access:** This type of SLR includes research papers to which full text is available for a consequent review.
  - **Original research papers:** This kind of SLR selects research papers that present research results, such as outcomes or theoretical analyses that advance the detection and classification of ransomware using ML.
- **Exclusion Criteria**
  - **Irrelevant Studies:** Excluded papers included those that did not focus on the detection and analysis of the mobile malware.
  - **Non-Peer-Reviewed Sources:** Only peer-reviewed publications were used because gray literature and opinion articles do not meet academic standards. **Non-English Papers:** Only the papers published in English were included to avoid complications with translation and to capture as much content as possible.
  - **Duplicate Studies:** Replication or duplication of studies across the databases was eliminated to avoid the inclusion of similar studies.
  - **Inaccessible Papers:** It was important to have access to whole papers, so partially available papers were excluded.
  - **Insufficient Paper Length:** Studies that presented the findings in the form of short papers without adequate details or included limited information on the subject were also excluded from the SLR.

To this end, we ensured the previous criteria were met in order to identify the relevant studies that align with the topic of detection and analysis of mobile malware.

B. CONDUCTING

Here, we included additional information about the search string and data sources adopted during the conducting phase.

1) Search Strings

We employed the search string to identify the relevant studies and to refine the search outcomes. The keywords used in the search string of this paper include "malware", "malware evolution", "malware detection" and "malware analysis" to get diverse search results across different databases. Keywords are used with boolean operators like AND, OR, and NOT to improve search results and filter out information.

2) Data Sources

The review used two scientific databases, which are Google Scholar and the Saudi Digital Library, to gather data sources. Each database is chosen for its pertinence to computer science, cybersecurity, and mobile technology literature.

C. REPORTING

In this context, it is important to offer a clear and sequential account of the results obtained in the course of the screening and selection to avoid a selective reporting of the results. Furthermore, we included the number of studies identified, screened, and included in the study in the section titled "Selection process".

1) Screening Process

Initially, all the studies collected from the search phase go through a review based on their titles and abstracts to determine their relevance to our paper. Any studies that do not match the research focus criteria are excluded from consideration. When we move to the next stage of the process, which involves examining each chosen paper for its methodology, relevance, and impact in the field, any duplicated or irrelevant studies are eliminated at this phase.

2) Selection Process

In our research, we applied PRISMA 2020 to efficiently handle and structure the flow of data during the review process, as shown in Figure 1 [4]. The papers were extracted from two databases, which are Google Scholar and the Saudi Digital Library. Google Scholar yielded 7,750 findings, and the Saudi Digital Library presented 8,108 results. Our search was conducted using the query ("mobile malware") AND ("malware evolution" OR "malware detection" OR "malware analysis"). We also selected the papers that were published in the range of 2020 to 2024. A total of 13,291 papers were removed for different reasons, such as the paper being flagged as ineligible. After a comprehensive review process, 2,567 papers underwent detailed examination, resulting in the selection of 30 papers that aligned with the research goals. The remaining 2,537 papers were eliminated due to reasons like being irrelevant to the topic, written in a language other than English, outside the designated timeframe, or lacking accessibility.

Only the most relevant studies are included, thanks to the strict methodology used. The following sections go into greater detail about these chosen works, beginning with a targeted examination of malware evolution.

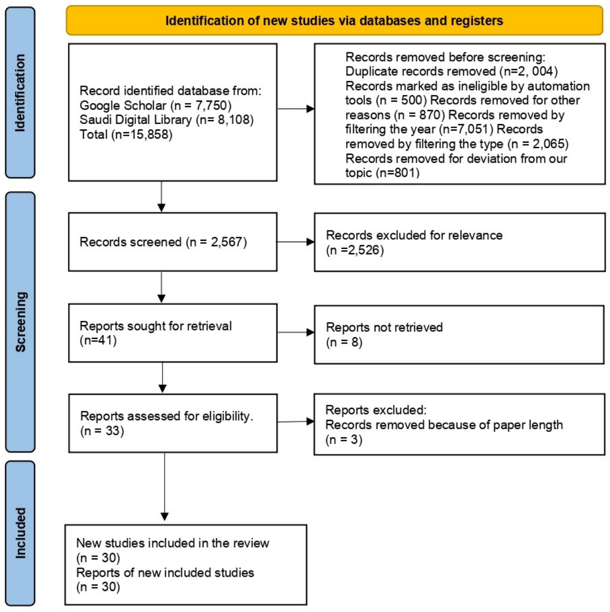
Identification of new studies via databases and registers
Identification
Record identified database from: Google Scholar (n = 7,750) Saudi Digital Library (n= 8,108) Total (n=15,858)
Records removed before screening: Duplicate records removed (n=2, 004) Records marked as ineligible by automation tools (n = 500) Records removed for other reasons (n = 870) Records removed by filtering the year (n=7,051) Records removed by filtering the type (n = 2,065) Records removed for deviation from our topic (n=801)
Screening
Records screened (n = 2,567)
Records excluded for relevance (n =2,526)
Reports sought for retrieval (n=41)
Reports not retrieved (n = 8)
Reports assessed for eligibility. (n = 33)
Reports excluded: Records removed because of paper length (n = 3)
Included
New studies included in the review (n = 30) Reports of new included studies (n = 30)

FIGURE 1: Selection of papers with PRISMA for a literature review.

IV. EVOLUTION OF MOBILE MALWARE

Mobile malware has significantly evolved from simple software that caused minor disruptions to advanced threats capable of serious harm. This progression includes a shift from early attacks to more complex forms, such as trojans and spyware, highlighting the increasing sophistication of these threats. Modern mobile malware employs advanced techniques like encryption, rootkits, and zero-day exploits, with widespread examples including spyware, ransomware, and mobile banking trojans. Attackers are also utilizing methods like malware-as-a-service, making it easier for inexperienced hackers to execute sophisticated attacks.

Recent examples, such as APTs, are particularly concerning as they can evade detection and remain on devices for extended periods. This timeline illustrates the transformation of mobile malware from simple worms to highly advanced threats that pose significant financial, personal, and security risks, underscoring the growing complexity and persistence of these attacks.

This section will cover key examples and techniques of mobile malware and illustrate how these threats have evolved and increased over time.

A. TIMELINE OF MOBILE MALWARE

The evolution of mobile malware has changed a lot over the years as shown in Table 2. It started in the mid-2000s with simple viruses and worms that targeted older mobile operating systems like **Symbian and Java ME**. Early examples, like **Cabir and Commwarrior**, spread through Bluetooth and multimedia messaging, marking the beginning of mobile threats [2]. As smartphones became more popular, especially with the rise of Android and iOS, malware grew more advanced, using the open nature of these systems to spread malicious apps widely.

During the 2010s, malware became more dangerous, with threats such as phishing, data theft, and financial fraud becoming common. Malware like **HummingBad and various banking trojans** started to appear. Ransomware also emerged, where attackers would lock users out of their phones and demand payment to unlock them. In recent years, malware has become even more advanced, with spyware like **Pegasus and malware-as-a-service** allowing even inexperienced attackers to launch complex attacks.

Today, mobile malware often takes advantage of weaknesses in apps and operating systems, using techniques like zero-day exploits and tracking user behavior. This means cybersecurity measures need to keep improving to protect against these ever-changing threats.

TABLE 2: Timeline of Mobile Malware Evolution

Generation	Period	Examples	Description
Early Generation	2000–2010	Cabir, Commwarrior.	Proof-of-concept malware targeting early mobile operating systems.
Mid Generation	2010–2015	DroidDream, Zitmo.	Trojans and spyware focused on stealing personal data.
Recent Generation	2015–Present	Pegasus, HummingBad.	Advanced malware like ransomware, banking trojans, and APTs.

To illustrate how mobile malware threats have evolved, Figure 2 presents a timeline of significant malware types and their behaviors over the past two decades. This figure illustrates the chronological progression of major mobile malware families from 2004 to 2022. It highlights how mobile threats have evolved from early Bluetooth-based worms like Cabir to advanced spyware such as Pegasus. Each entry marks a significant shift in malware capabilities, including banking fraud, rootkit-based persistence, SMS phishing, and zero-click remote access.

An overview of the important mobile malware incidents and information about their properties, consequences, and countermeasures are presented in Table 3. Every row concerns a specific type of malware, as for the description – it informs about the threat's characteristics and the way it operates. The analysis column outlines how the behavior of the malware is analyzed and the result column outlines the implication of the malware to users and organizations. Finally, the solution column shows the measures which are being employed to tackle such threats and includes the need to ensure that updates are frequent, that detection techniques are improved and users are made to be more cautious to prevent other future threats from occurring. This table is also a useful resource to refer to when studying the change patterns regarding mobile malware and the strategies to counter them.

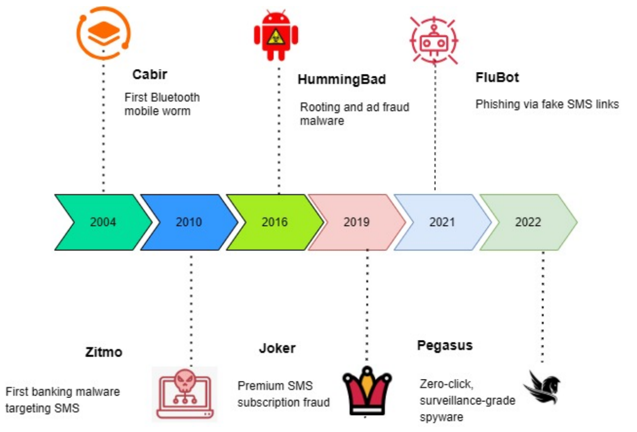
Cabir
First Bluetooth mobile worm
HummingBad
Rooting and ad fraud malware
FluBot
Phishing via fake SMS links
2004
2010
2016
2019
2021
2022
Zitmo
First banking malware targeting SMS
Joker
Premium SMS subscription fraud
Pegasus
Zero-click, surveillance-grade spyware

FIGURE 2: Timeline of Mobile Malware Evolution

In the table, AI is mentioned in the analysis of Joker Malware (2020), where ML was used to detect unusual behavior in Android apps, like accessing contacts and sending SMS without the user knowing [1]. This shows how AI, especially ML, helps detect modern malware where traditional methods may not work. Despite 20% AI usage in the examples, research on AI-based detection techniques is still limited. More studies and implementations are needed as the threat from sophisticated mobile malware grows. By improving detection using AI and ML these techniques play a vital role in detecting and quickly dealing with suspicious activity before it causes widespread damage.

B. TYPES OF MOBILE MALWARE

Malware that infects mobile devices is classified into several types as presented in Table 4, each with different capabilities and goals, each targeting specific vulnerabilities and offering unique functions. Trojans are designed to look like legitimate applications but perform malicious actions. Ransomware encrypts user data and the criminal demands payment to release it. Spyware monitors user activity and collects sensitive information. Adware displays unwanted ads, causing system performance to slow down. Rootkits give attackers elevated privileges, allowing them to hide their activities from the operating system.

C. MALWARE PROPAGATION TECHNIQUES

Table 5 shows different techniques of malware propagation that are used by cybercriminals. For example, cybercriminals employ social engineering to manipulate users into actions that compromise their security. One common tactic is phishing, where fraudulent messages, often delivered via email or SMS, are crafted to appear as though they originate from reputable sources [1]. These deceptive communications typically include links to malicious websites or encourage users to download counterfeit applications. By leveraging the element of trust, attackers can deceive individuals into inadvertently installing malware.

In the case of app repackaging, malicious actors alter

TABLE 3: Summary of Mobile Malware with Analysis and Solutions

Malware	Description	Analysis	Result	Solution
WannaCry (2017) [5]	Ransomware targeting systems with Windows vulnerabilities, infecting hundreds of thousands of devices globally.	Used static and dynamic analysis to trace the EternalBlue exploit and track propagation across networks.	Disrupted global organizations, including hospitals, demanding ransom payments.	Microsoft released a security patch, and organizations were advised to update systems and back up data.
Pegasus (2019) [6]	Spyware developed by NSO Group, targeting iOS and Android devices via app vulnerabilities for surveillance.	Behavioral analysis and reverse engineering used to trace communication with command-and-control servers.	Linked to surveillance targeting journalists, activists, and political figures globally.	Security patches were issued for iOS and Android to fix vulnerabilities, and users were warned about malicious links.
Joker (2020) [7]	Malware spread through Android apps on Google Play, stealing SMS, contacts, and enrolling users in paid services.	Google used ML techniques to detect unusual app behavior like accessing contact lists.	Joker malware was removed from Google Play after affecting millions of users.	Google improved app review processes and enhanced ML-based detection.
Dridex (2021) [8]	Banking trojan spreading through phishing emails, stealing financial data via infected documents.	Analyzed phishing emails and document macros to track the banking trojan's behavior in infected systems.	Compromised financial institutions, resulting in significant financial losses.	Security alerts were issued, email filtering was enhanced, and two-factor authentication was implemented.
FluBot (2022) [9]	Malware targeting Android devices via SMS phishing, stealing personal and financial data.	Dynamic analysis used to trace FluBot's actions after users clicked malicious links, focusing on data exfiltration.	Infected thousands of Android users, leading to data theft and financial loss.	Authorities launched campaigns to inform users about SMS phishing and advised avoiding suspicious links.

TABLE 4: Types of Mobile Malware

Type of Malware	Description	Examples
Trojans	Malicious software disguised as legitimate apps, often used to steal data or gain control of the device.	DroidDream, Zitmo
Ransomware	Blocks access to the device or data, demanding payment to unlock it.	SLocker, WannaCry (mobile version)
Spyware	Secretly monitors user activity, collecting sensitive information like passwords and location data.	Pegasus, FlexiSpy
Adware	Displays unwanted ads on the device, often slowing down the system and collecting user data without permission.	SimBad, Ewind
Rootkits	Provides unauthorized access to the device's system, allowing attackers to gain full control and hide their activities.	DroidKungFu, Ztorg
Banking Trojans	Specifically targets financial data, often intercepting SMS messages or login credentials for banking apps.	Anubis, Marcher
Worms	Self-replicating malware that spreads to other devices without any user interaction.	Cabir, Commwarrior

legitimate applications by embedding harmful code. These compromised apps are subsequently distributed through unofficial app stores or third-party websites. Users, under the impression that these applications are secure, may download them, thereby allowing malware to infiltrate their devices.

Phishing specifically targets the acquisition of personal information. Attackers design counterfeit websites or messages that closely resemble those of legitimate organizations, such as banks or social media platforms. When users input their personal information, it is captured by the attackers for purposes of identity theft or financial fraud.

All these strategies exploit users' trust and the vulnerabilities present in mobile devices and applications. For instance, users may underestimate the dangers associated with downloading apps from unofficial sources or clicking on links contained in unsolicited messages. Additionally, weaknesses within the operating system or applications can be exploited to install malware or extract sensitive data. These methods underscore the critical need for user attention and robust security protocols to safeguard against mobile malware. Effective malware threat mitigation requires a multi-faceted approach that includes user education, adherence to best practices, and implementation of strong technical safeguards. Regularly updating software and systems is crucial to addressing vulnerabilities that malware can exploit. By downloading apps exclusively from trusted sources, such as official app stores, the likelihood of unintentionally installing malware is greatly reduced. Conversely, using robust security solutions, enabling two-factor authentication, and being vigilant against phishing attempts are vital measures to prevent malware infiltration. Furthermore, using strong, unique passwords, along with limiting app permissions, can help minimize the potential impact of malware. Regular data backups are essential to minimize the consequences of attacks such as ransomware, while connecting to secure Wi-Fi networks provides additional protection against network-related threats.

D. ADVANCED MOBILE MALWARE TECHNIQUES

- **Obfuscation**
  Obfuscation is a technique where malware developers intentionally make the code difficult to understand by altering its structure while keeping its functionality in-

TABLE 5: Malware Propagation Techniques and Evasion Tactics

Propagation Technique	**Examples**	**Techniques Included**	**Evasion Techniques Used**
Social Engineering	FluBot, Joker	Phishing, Fake Apps	User deception, Zero-Day exploits
App Repackaging	DroidKungFu, Gooligan	Legitimate App Injection, Distribution via third-party stores	Obfuscation, Polymorphism
Phishing	Anubis, Cerberus	Fake Emails, Websites	URL spoofing, HTTPS encryption
Drive-by Downloads	HummingBad, Svpeng	Exploiting browser vulnerabilities, Auto-downloads	Exploiting zero-day browser flaws
Bluetooth Exploitation	Cabir, Commwarrior	Bluetooth vulnerability exploitation, Device-to-device spread	Operating at a low system level, Bypassing regular monitoring
Network Exploitation	WannaCry (mobile), NotPetya	Wi-Fi vulnerability exploitation, Network-level infection	Encryption, Polymorphism
SMS Phishing (Smishing)	FluBot, TimpDoor	Malicious links via SMS, Fraudulent messages	SMS communication evasion, Minimal scrutiny by security software

tact. This technique is used to hide the true intent of the malware, making it harder for security tools to analyze and detect it. By complicating the code, obfuscation impedes reverse engineering efforts, meaning even if the malware is discovered, understanding how it works and creating defenses against it becomes more challenging [10].

- **Encryption**
  Encryption is another method employed by advanced malware to protect its communications or payload. Malware encrypts the data it sends and receives, often from command-and-control (C2) servers, making it difficult for security systems to intercept and analyze the traffic. Additionally, some malware encrypts its own code or files, making static analysis by security tools nearly impossible. This technique prevents security software from identifying malicious behaviors or actions, allowing the malware to operate covertly [11].
- **Polymorphism**
  Polymorphic malware changes its code each time it infects a new device or is executed, enabling it to evade signature-based detection systems. Since traditional antivirus software relies on detecting known patterns or signatures, polymorphism allows malware to continuously mutate, making each version appear unique. This constant code variation prevents the creation of fixed signatures, allowing the malware to spread and persist over time without being easily identified [12].
- **Rootkits**
  Rootkits give attackers deep access to a device, often at the kernel or system level, allowing them to maintain control and hide their presence from the user and security systems. Once installed, rootkits can modify system files, log keystrokes, and even disable security software, all while remaining undetected. Rootkits make it possible for malware to persist on a device for extended periods, often requiring specialized tools to identify and remove them [13].
- **Stealth Techniques**
  Stealth techniques such as code obfuscation and polymorphism are specifically designed to avoid detection. By altering the code's structure or its execution pattern, these techniques make it harder for both static and dynamic analysis tools to detect the malware. Polymorphic malware mutates its code, while obfuscated malware hides its intent. Together, these techniques ensure that malware can operate unnoticed, even by advanced detection systems [14].
- **Evasion Techniques**
  Evasion techniques help malware avoid detection by tricking or bypassing analysis environments. For example, malware may detect when it is running in a sandbox or virtual environment, which are often used by cybersecurity researchers to analyze malware behavior. Upon detecting such environments, the malware may alter its behavior or stop functioning, making it difficult to study and detect using automated systems. These techniques help the malware stay hidden for longer periods, further delaying detection and response [15].
- **(Advanced Persistent Threats)** APTs are characterized by prolonged and highly focused attacks designed to remain undetected within a system for significant durations. In mobile contexts, APTs take advantage of weaknesses in mobile operating systems, applications, or network protocols to secure ongoing access. Unlike conventional malware, APTs prioritize the collection of intelligence or the retention of control over a device, frequently without inflicting immediate harm. Their complexity and subtlety render APTs particularly difficult to identify and eliminate, especially in mobile settings where continuous access to personal and corporate information is of great importance [16].
  These advanced techniques collectively highlight the increasing sophistication of modern mobile malware, making it necessary for cybersecurity solutions to evolve in order to effectively detect and neutralize these threats.

The evolving techniques and obfuscation methods employed by mobile malware underscore the need for robust and adaptive detection strategies. This sets the stage for the next section, which explores how current detection approaches, both static and dynamic, aim to address these increasingly evasive threats.

E. INDICATORS OF MALICIOUS BEHAVIOR IN MOBILE MALWARE

Mobile malware exhibits specific behavioral and structural indicators that can be leveraged during detection. These indicators include:

- **Encoded Payloads:** Many malware samples obfuscate code or strings using base64 encoding or XOR functions to evade signature-based detection. For example, the Joker malware used base64 encoding to hide malicious payloads embedded in APKs [17].
- **Embedded Credentials:** Hardcoded API keys, usernames, or tokens are often included in malicious code to gain unauthorized access to services or C2 infrastructure. Pegasus is a notable example, embedding credentials for remote control [18].
- **C2 Communications:** Command-and-Control traffic allows malware to receive instructions or exfiltrate data. Malware like Anubis uses HTTPS or DNS tunneling to maintain encrypted, hard-to-detect communications [19].
- **Suspicious URLs:** URLs hardcoded or fetched dynamically often lead to phishing pages or malware downloads. FluBot, for instance, uses shortened URLs in SMS messages to trick users into downloading APKs [20].

Table 6 summarizes real-world examples linked to these indicators.

TABLE 6: Behavioral Indicators in Mobile Malware

Indicator	Description	Real-World Example	Observed Behavior
Encoded Payloads [21]	Uses Base64 or XOR to hide malicious content.	Joker	Base64-encoded payload in APK assets to evade scanners.
Embedded Credentials [22]	Hardcoded tokens or credentials used for access.	Pegasu	Embedded keys used to connect to C2 servers.
C2 Communication [23]	Malware communicates with attacker-controlled servers over HTTPS or DNS tunneling.	Anubis	Encrypted periodic traffic to remote command servers.
Suspicious URLs [24]	Use of shortened or obfuscated links to lure users.	FluBot	SMS phishing with shortened URLs to install malware.
Stealth Techniques [25]	Detection evasion via sandbox detection or delayed execution.	Pegasus	Malware halts execution in virtual environments.

Understanding these malware evolution trends is essential to comprehending why conventional detection techniques frequently fall short. A comparative analysis of recent research that offers creative answers to this expanding problem is provided in the following section.

V. RELATED STUDY

This section examines recent studies on mobile malware detection and analysis published between 2020 and 2024. Table 7 provides a comparative analysis of these works based on key aspects such as the addressed problems, adopted methodologies, reported results, and identified limitations. Additionally, Table 8 offers a summarized comparison of intelligent mobile malware detection approaches proposed in recent literature, highlighting differences in algorithms, feature selection techniques, model architectures, and evaluation metrics.

Alazab et al. [26] proposed a classification model to differentiate benign apps from malicious apps in the Android operating system. Their model combines API calls with permission requests to better classification results. Android apps use large numbers of APIs to allow interaction with the system. For this reason, the authors prefer to classify these APIs into 3 groups to enhance the detection of malware apps in this environment. So, these groups are distributed as follows: the ambiguous group, the disruptive group, and the risky group. The authors of this study use two different datasets, one for malicious app samples, which include some reference datasets like AndroZoo, Contagio, MalShare, and VirusShare, and the other for benign app samples that were created and verified by the authors. The proposed model contains 3 phases, which are the pre-processing, extraction, and grouping phases. Furthermore, it utilizes five ML algorithms, which are Random Forest (RF), J48, Random Tree (RT), K-Nearest Neighbors (kNN), and Naïve Bayes (NB). All these algorithms were executed through fold cross-validation. The experiment results prove that malicious apps invoke several groups of API calls, and the malware of the mobile applications mostly asks for risky permissions to access users' sensitive data. In addition, the proposed model shows its effectiveness in detecting malware in mobile apps and achieves 94.3% for F-measure. Moreover, the research findings show that permissions and API calls have a critical role in classifying variants of malware.

Gong et al. [27] present a real-world mobile malware detection system that was deployed and implemented in Android T-Market. This system is called APICHECKER, which is based on machine-learning algorithms. APICHECKER examines approximately 10K apps every day to detect malware apps. The study utilizes a data set containing 500k malicious and benign apps that were collected over 10 months from the Android Store. In addition, dynamic analysis is used to track runtime API invocations. Furthermore, 9 ML algorithms are considered and finalized with the RF algorithm because it achieves high performance compared with other algorithms. Recall of 96.7% and precision of 98.6% were the results of their deployed system. In addition, their proposed system can examine 10k apps with an average processing time of 1.3 m/app. Difficulties in handling apps that used Java reflection methods to hide API calls.

Yan et al. [28] suggest a novel method to detect mobile malware by extracting the rules from Deep Neural Network

(DNN) using network traffic analysis. The study used a DroidCollector, which is a traffic data collector, to collect the dataset that contains 99,100 benign flows and 31,800 malicious flows. In addition, the paper focuses on DNN to detect malware by extracting the rules and then compares this method with different ML methods like Support Vector Machine (SVM), RF, and AdaBoost. Superior findings were achieved by the proposed method with 98.55% accuracy,98.27% recall, 97.93 % precision, and 98.04% F-measure. Complex models like CNN may outperform the proposed model due to their ability to handle and extract fine-grained features.

Taher et al. [29] proposed a novel mobile malware detection method named DroidDetectMW, which is a hybrid model that uses static and dynamic analysis. The paper proposed a model that utilizes an enhanced Deep Learning (DL) model (ANN) by Harris Hawks Optimizer (EHHO) for classification. Also, it used different feature selection techniques like fuzzy metaheuristic, chi-square, and Fisher score. The dataset used in this experiment contains 2980 benign samples and 1910 malware. Compared with traditional ML algorithms like SVM and RF, the ANN model proved its ability to improve detection performance and achieve a high accuracy rate. Lastly, the negative side of the proposed algorithm is that computational complexity is increased due to using hybrid analysis.

Casolare et al. [30] proposed a malware detection model that used dynamic analysis. The dataset contains 10 malware families: 3355 malware and 3462 legitimate applications. The proposed model represents system call traces as images and then classifies these images by utilizing machine and DL algorithms like CNN, SVM, RF, and Multi-Layer Perceptron (MLP) as benign or malware. A high accuracy of 96.72% was achieved by using this model. Dynamic analysis requires a long time to be executed, and generating images and classifying them is computationally expensive.

Chuanchang et al. [31] introduce MobiPCR, an innovative ML-based system for efficient and accurate mobile malware detection. Addressing the limitations of current methods, MobiPCR utilizes a cloud-based architecture and lightweight computational models, making it ideal for resource-constrained mobile devices. The system enhances detection accuracy through the stacking of classifiers, achieving 99.43% accuracy with a low false-positive rate of 0.13%. This system is designed to be user-friendly and integrates smoothly into the app installation process.

Giacomo et al. [32] proposes an interpretable DL model for mobile malware detection and family identification, addressing the challenge of explaining how DL models make decisions. It focuses on improving the transparency and trustworthiness of malware detection models by utilizing Gradient-weighted Class Activation Mapping (Grad-CAM) to visually represent the parts of input images that influence the model's predictions. The proposed model achieves high accuracy, ranging from 96% to 97%, tested on over 8,000 Android samples from six malware families.

Kakelli et al. [33] discusses various malware types and how they have evolved. As well as the techniques and tools developed to detect them. They focused on Android's vulnerabilities and examined modern threats like Trojan droppers and adware. It also evaluates detection methods such as static, dynamic, and hybrid analysis. Furthermore, it emphasizes the need for more robust and effective detection frameworks to combat the growing complexity of Android malware.

Gianni et al. [34] proposed a new malware detection framework that utilizes neural networks and DL techniques to identify zero-day malware on Android devices. This framework employs autoencoders to independently extract the most significant and distinguishing features from API images. It presents a novel approach by representing the sequence of API calls executed by applications as API images. These images serve as input for ML classifiers to detect malicious activities. The proposed model exhibited exceptional performance, achieving an accuracy exceeding 90% and an F-measure of 0.97 when evaluated on various Android-related datasets.

Madihah et al. [35] provide a comprehensive examination of the increasing danger posed by malware aimed at iOS devices, with particular emphasis on the exploitation of social media and online banking platforms. The research employed a hybrid analysis involving 12 malware samples within a controlled setting to track the progression of iOS malware through a phylogenetic model. Furthermore, the study established a classification model capable of identifying and detecting potential security threats within iOS applications, revealing that 4% of the applications tested exhibited vulnerabilities to exploitation.

Ababneh [36] proposed a new highly reliable ML approach for malware detection with special considerations given to Android operating systems. The research used the CCCS-CIC-AndMal-2020 dataset that incorporates a larger and updated set of malware classifications. To reduce the problem of dataset imbalance, methods like SMOTE and Spread-Sub-Sample were employed. The feature selection involved using Information Gain Attribute Evaluator (GAN) and Correlation-based Feature Subset Evaluator (CFS) to sort out the best features from the list of features that would give good prediction results while still taking a reasonable time to accomplish the task. The framework targeted dynamic features and assessed several ML classifiers such as RF, J48 Decision Trees (DT), Naïve Baye, and KNN. Experimental outcomes revealed that the mean accuracy of RF was the highest, which is 98.9%. As for model assessment, the 10-fold cross-validation was used. However, the approach may not be effective against stealth malware, which can avoid emulating actions within a sandbox. Furthermore, the job of achieving real-time detection while maintaining competitively lightweight computation remains a crucial problem for the future development of the field.

Yerima [37] proposed a novel ML framework for the detection of mobile malware using ML and DL for a better classification of malicious applications for mobile platforms.

The study was performed using four datasets, which include various kinds of malware and benign samples. In order to capture as many features as possible, both static and dynamic attributes were incorporated into the framework. Feature selection techniques such as Minimum Redundancy Maximum Relevance (mRMR) and Analysis of Variance (ANOVA) were used to select important features. Various ML and DL algorithms incorporated in the evaluation include RF, SVM, Naïve Bayes, and J48, Deep Dense Network (DDN), 1D CNN, and CNN-LSTM. As a result, RF performed significantly well; however, the DL models, namely, the CNN-LSTM and 1D-CNN models, had slightly better F1 scores with values above 98% in some tests. However some of the challenges that the framework has are the time complexity related to feature extraction and the DL algorithm's high computational complexities, and since most of the features of the datasets used in developing the framework may not necessarily be the same as those of other datasets.

A malware detection system based on the static analysis of Andriod applications was put forward by Arif et al. [38], where the ML classifier was used to detect the new malware efficiently. It builds on permission-based features derived from the application manifest files and uses two large datasets, Androzoo and Drebin, for testing. Other methods that were used to improve the detection efficacy include Particle Swarm Optimization (PSO) and Information Gain for feature optimization. Several ML classifiers such as RF, MLP, kNN, J48, and AdaBoost were used, and to control class overfitting, 10-fold cross-validation was applied. Performance results were explored in the experimentation phase, with RF registering the best accuracy score of 91.6%. Other classifiers, such as kNN and MLP, were also responsive, with an accuracy of more than 90%. However, it is essential to note that the study has found some limitations, such as the absence of permission-based features, which may neglect some behavior patterns. Furthermore, the integration with other hybrid analysis approaches based on the mixture of static and dynamic methods was mentioned as a future development.

Kaur et al. [39] proposed a machine-learning model for Android malware detection in which app permissions were used as the primary attributes. Consequently, the ability to integrate this permission data into a flexible model from a dataset of about 30,000 Android applications is another goal of the study, which is considered essential for the accurate detection of both benign and malicious apps. The approach begins with data cleaning, where the dataset is preprocessed, later followed by Exploratory Data Analysis (EDA) to discover potential patterns in this data set. In order to avoid multicollinearity and enhance computational speed, the technique called Principal Component Analysis (PCA) is employed for the reduction in dimensionality of the data. Different classifiers, such as Logistic Regression, DT, NB, RF, SVM, and MLP are used. Experimental results reveal that the RF classifier yielded the best accuracy. Highlighting a research gap in identifying advanced and complex malware variants that cannot be easily addressed by static analysis techniques, the study points towards the need for constantly evolving other dynamic proactive detection techniques. Furthermore, the real-time use of the model is prohibited too because it does not work with dynamically created features, such as real-time malware behavioral changes.

Lekssays [40] proposed a new approach to Android malware detection and categorization using CNN. The method uses a static analysis method, whereby binaries are translated to 2D arrays, and the Android application APK files are converted to grayscale images. These images capture similar structures between different malware groups in order to provide better distinction between classes. The research made use of useful benign samples derived from the Google Play Store and suspicious samples derived from public APIs and the CICAndMal2017 dataset. The proposed methodology used two algorithms, to begin with, the CNN for basic classification and the kNN for the image vector examination. The first architecture of CNN was able to yield 84.9% accuracy, while a much sparser architecture had only 68.1% accuracy, emphasizing the benefits of deeper architectures. However, the following limitations were observed; limited dataset size due to storage and computational constraints, only achieved binary classification of benign and malware without subclassification to various families of malware.

Shatnawi et al. [41] presented an Android malware detection technique that is based on static feature analysis using ML algorithms. The approach uses Android permissions and API calls and three ML algorithms: SVMs, KNN, and NB for classification. This research work has made useful contributions by introducing updated CICInvesAndMal2019 datasets that improve the existing approaches of static features and malware classification and the new feature selection technique referred to as Recursive Feature Elimination (RFE) for improvement in the detection rates. The findings indicate that SVM delivers the highest detection accuracy rate of all the algorithms, standing at 94.36%. However, the inability of static analysis to capture dynamic behaviors of malware as well as a small dataset to increase the practical applicability are the main shortcomings of the study.

Villarroel [42] looked at the utilization of ML in the enhancement of malware detection on smart mobile devices based on dynamic behavioral analysis. This research evaluated the three ML algorithms, namely LightGBM, XGBoost, and RF, using the CICMalDroid 2020 dataset. The process of selecting features was done using SelectFromModel, which gives the features according to their importance. The malware was categorized based on dynamic system call frequencies, and the LightGBM model performed the best with an accuracy of 93.95% and a precision of 94.1%. However, study limitations of imbalanced data sets and dependence on outdated tools offered only limited analyses. Moreover, the false positive issues, which label benign applications as malware, are concerns about user trust and operational concerns as well.

Arif et al. [43] presented a comprehensive multiple-attribute decision-making approach to detect Android malware through the combination of ML and fuzzy AHP for risk evaluation. Permission-based features of the Android system are utilized for static analysis for the current study. It divides the applications into four risk levels, very low, low, medium, and high, designed to inform the user of security risks and achieve a high detection rate of 90.54% malware. The dataset consists of the ten thousand samples collected from the Drebin and AndroZoo databases. With the help of Information Gain, a selection of the most important permission-based features was introduced, and the dataset was reduced to increase performance. The complete model was able to detect the malware with 90.54% accuracy, and risk levels can be accurately classified through a box plot analysis. Moreover, fuzzy AHP provided consistent and accurate evaluations with a ratio of consistency (CR < 0.1). However, the study was confined to permission-based features only, and other forms of static attributes such as Java code and intent filters, which could further improve detection, were not included in the study.

Chen et al. [44] have attempted to address the increasing threats of mobile malware, especially in Android environments, by proposing an end-to-end framework for detection, tracing, and propagation study. The proposed system also combines static and dynamic analysis techniques with a ML algorithm to detect and categorize the types of malware. One contribution includes creating a knowledge map and a 'gene recognition' system on the malware families to track their spread and facilitate identification in different network structures. The experiment was carried out using a dataset of 178,155 real mobile malware samples from China Unicum's network. The work builds behavioral graphs to aggregate related malware classes and distill out common characteristics. Moreover, it uses ML algorithms including KNN, DT, and SVM for malware detection and classification. The integration of the behavior clustering was found to actually enhance the efficacy of the family classification of malware when compared to the graph-based approach that decisively helped minimize false positives. However, the system has a limitation in that the dynamic analysis and graph clustering will, in general, be computationally expensive. Besides, inadequacies of the framework are observed when it comes to dealing with the constantly emerging malware that targets new vulnerabilities in the IoT and 5G networks.

Haq [45] discussed the detection of Android malware using an ML approach. The work introduces a new approach that combines static and dynamic analysis with ML models with optimized hyperparameters and cross-validation to improve detection rates. The dataset employed in the study has 398 records and 331 features acquired from the domain of analyses of Android malware. This performance analysis was performed on four types of ML models, including XGBoost, RF, SVM, and DT, in detecting malicious applications. RF obtained the highest accuracy at 99%. Nevertheless, the study faces several shortcomings in terms of non-application of real-time evaluation of the model and a small dataset, which does not work with more data. Furthermore, the mechanism of how to use hybrid analysis for feature selection is not being discussed in detail.

Chandra et al. [46] discussed the rising number of malware attacks aimed at IoT and Android platforms, highlighting their weaknesses in hardware, software, and network layers. It classifies different types of attacks, such as physical, network, software-related, and data breaches, and examines ways to mitigate these threats, with a focus on application hardening and effective detection methods. By employing static, dynamic, and hybrid analysis techniques, the research analyzes features like API calls, permissions, and system calls, utilizing machine learning algorithms such as Random Forest and SVM to achieve impressive accuracy rates, often above 95%. Particularly, hybrid approaches and image-based classification through convolutional neural networks show outstanding results, with detection rates exceeding 99%. However, the study also points out some limitations, including the absence of IoT-specific vulnerability classifications and alignment with standards like OWASP, as well as issues related to computational demands in practical applications. The paper provides valuable insights into improving security for IoT and Android systems through enhanced permission systems, encryption methods, and multi-factor authentication, while stressing the importance of ongoing research to tackle unresolved vulnerabilities and enhance scalability.

Jueun et al. [47] addressed the difficulties associated with identifying malware in Internet of Things (IoT) devices, particularly in light of the hardware constraints inherent to these devices, which include both novel and variant forms of malware. It introduces DAIMD, a cloud-based framework that performs dynamic malware analysis by extracting critical features such as memory usage, network activity, and system call information. These extracted features are then represented as behavioral images and classified utilizing the convolutional neural network model ZFNet. The model demonstrated impressive performance, achieving an accuracy rate of 99.28% alongside a minimal false positive rate, thereby proving its efficacy in malware detection. Nonetheless, the persistence of malware that can bypass dynamic analysis presents a challenge, underscoring the necessity for hybrid methodologies that integrate both static and dynamic analysis techniques. This proposed system provides a scalable solution aimed at enhancing the security of IoT devices.

Rana et al. [48] focused on the critical issue of identifying Android ransomware, which presents considerable threats by either locking devices or encrypting user information. It introduces a hybrid detection framework that integrates both static and dynamic analysis to enhance the accuracy of detection. This approach utilizes over 70 antivirus engines for static analysis, alongside the MobSF tool for dynamic analysis, capitalizing on various features such as permissions, API calls, and encryption patterns. The findings indicate that static analysis yields an accuracy rate between 40% and 55%, whereas dynamic analysis consistently achieves a perfect score of 100%. Nonetheless, the method faces challenges, including the lengthy process associated with dynamic analysis and the complications arising from ransomware that employs sophisticated evasion strategies.

Amira Sallow et al. [49] examined the increasing prevalence of Android malware attributed to the platform's open-source characteristics. Their research emphasizes the improvement of detection methods through both static and dynamic analyses, alongside the application of sophisticated techniques such as machine learning. The study utilizes datasets that encompass various features, including API calls, permissions, and control flow graphs, implementing Principal Component Analysis (PCA) for effective feature selection. Additionally, it employs algorithms such as Support Vector Machines (SVM), Random Forest, and Neural Networks to enhance malware detection capabilities. Although the findings demonstrate a high level of accuracy, the research underscores the difficulties associated with identifying new malware variants and the resource demands of hybrid detection methods, thereby highlighting the necessity for lightweight and scalable solutions to bolster Android security.

Moses et al. [50] examined the progression and infection methodologies of mobile malware, with a specific emphasis on Android devices, spanning the years 2000 to 2020. It underscores the increasing complexity of malware attacks and the significant challenges they present to information security. This research offers a contemporary evaluation of malware trends, categorizes mobile malware according to their sophistication, attributes, and underlying motivations, and identifies various factors that have influenced their proliferation, including early deterrents such as the absence of operating system standardization and recent facilitators like the integration of the Internet of Things. Nonetheless, the research is constrained by its dependence on secondary data, a predominant focus on the Android platform, and the absence of empirical validation for the suggested solutions.

Yu Kyung et al. [51] investigated the difficulties associated with identifying mobile malware in the context of the increasing prevalence of 5G technology and advanced cyber threats. It analyzes 154 studies focused on machine learning techniques, utilizing datasets such as Drebin and MalGenome, which incorporate static, dynamic, and hybrid feature extraction methods. The evaluation of algorithms, including Support Vector Machines (SVM) and K-means, indicates that supervised learning approaches tend to be more effective; however, they face challenges such as zero-day vulnerabilities and a lack of diversity in datasets. The findings underscore the necessity for the development of enhanced algorithms and more comprehensive datasets to bolster mobile cybersecurity efforts.

Abijah et al. [52] examined the increasing intricacy of malware attacks and offers an extensive overview of the various tools and strategies employed to combat malware on both computers and mobile devices. It tackles the difficulties associated with identifying emerging threats, particularly zero-day attacks, which pose a considerable risk to cybersecurity. Among its contributions, the paper provides a thorough classification of malware detection techniques, including static, dynamic, vision-based, and hybrid methods, along with an analysis of their effectiveness. It assesses datasets such as Drebin and MalGenome, emphasizing key features like permissions, API calls, and behavioral patterns, while also exploring the application of machine learning algorithms, including SVM and deep learning, for the classification of malware. The findings suggest that hybrid approaches tend to yield greater accuracy; however, they also face challenges such as high computational demands, difficulties in detecting sophisticated obfuscated malware, and limitations in generalizability stemming from dataset issues.

Roopak et al. [53] addressed the difficulties in detecting Android malware by using a hybrid Tree Augmented Naive Bayes (TAN) model that merges both static and dynamic analysis methods. Their research addresses the issue of multicollinearity among various features such as API calls, permissions, and system calls, which can negatively affect the performance of classifiers. By utilizing datasets like Drebin, AMD, and AndroZoo, they implement ridge-regularized logistic regression for identifying anomalies and combine these findings with TAN for predicting malware, resulting in an impressive accuracy rate of 97%. However, the study faces challenges in identifying new malware due to concept drift, which necessitates frequent retraining. Future research is focused on improving Bayesian models to better handle adversarial strategies.

Raden et al. [54] examines the shortcomings of current Android malware detection techniques and introduces a hybrid model that integrates both static and dynamic analysis to enhance accuracy and efficiency. Utilizing datasets such as Drebin, Malware Genome, and CICMalDroid, a total of 261 features were extracted and evaluated through various machine learning algorithms, including Gradient Boosting (GB), Random Forest, and Multi-Layer Perceptron (MLP). The proposed hybrid model achieved an impressive accuracy rate of 99%, with Gradient Boosting demonstrating the highest effectiveness. Although this approach significantly enhances detection capabilities by merging static and dynamic methodologies, it does face challenges such as the risk of overfitting due to noisy data and substantial computational demands. The research underscores the importance of optimizing the model for real-time performance.

Corentin et al. [55] presented BrainShield model an innovative malware detection model for Android devices that combines machine learning techniques. It tackles the difficulties associated with identifying malware that exhibits a wide range of characteristics and relies on outdated datasets. The research utilizes the Omnidroid dataset, which comprises 22,000 samples and features over 30,000 static and dynamic attributes. Additionally, it introduces a client-server framework that incorporates neural networks for conducting static, dynamic, and hybrid analyses. To enhance training efficiency and minimize computational demands, the model implements feature selection methods, such as Pearson correlation, along with strategies like dropout regularization and binary cross-entropy to boost accuracy. The findings indicate that this hybrid approach achieves an impressive accuracy rate of 91.1%, surpassing traditional static and dynamic detection methods. However, the study acknowledges certain limitations, including dependence on the quality of the dataset and the risk of becoming outdated as malware evolves. It highlights the need for ongoing development to ensure scalability and the ability to respond to emerging threats.

Li et al. [56] presented a useful threat model to assess how resilient deep learning-based modulation classifiers are in adverse conditions. Proposing an Intra-Class Universal Adversarial Perturbation (IC-UAP) technique that uses class-specific information to improve the efficacy of universal adversarial attacks and an extended physical IC-UAP technique that takes into consideration actual unsynchronization between adversarial perturbations and transmitted signals constitutes their primary contribution. Methodologically, the study generates perturbations using two carefully designed algorithms, a modified projected gradient descent and an Adam optimizer, which are then assessed on the benchmark RadioML2016.10a dataset using the VT-CNN2 model. According to experimental results, the suggested IC-UAP performs better than current UAP crafting methods in a range of SNR and PNR scenarios, resulting in a 50% reduction in classifier accuracy even at low perturbation power. However, there are drawbacks in situations involving random time shifting, where all methods' performance decreases, underscoring the difficulty of maintaining attack efficacy in realistic channel conditions and sensitivity to synchronization.

The following section classifies the different detection methods and offers a systematic examination of their guiding principles, advantages, and disadvantages, building on this overview of current developments and their constraints.

VI. MOBILE MALWARE DETECTION TECHNIQUES

This section explores different mobile malware detection technologies, including signature-based detection, anomaly-based detection, static and dynamic analysis, ML and DL-based detection, and cloud-based detection as shown in Figure 3.

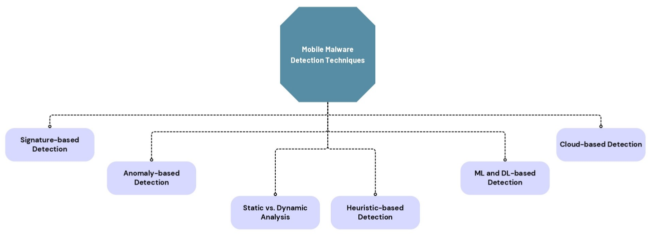
Mobile Malware Detection Techniques
Signature-based Detection
Anomaly-based Detection
Static vs. Dynamic Analysis
Heuristic-based Detection
ML and DL-based Detection
Cloud-based Detection

FIGURE 3: Mobile Malware Detection Techniques.

A. SIGNATURE-BASED DETECTION

Signature-based detection methods depend on recognizing malware by detecting byte sequences called signatures that are distinct to particular types of malware. However, this approach falls short when it comes to uncovering newer or disguised forms of malware due to the need for constant signature database updates and the difficulty in thwarting advanced evasion tactics such, as code obfuscation [57].

While detection models help in identifying malware, forensic techniques are essential to investigate, understand, and respond to mobile malware incidents. The following subsection outlines commonly used forensic tools and methods in mobile malware analysis.

B. FORENSIC TOOLS AND TECHNIQUES IN MOBILE MALWARE ANALYSIS

Forensic analysis plays a crucial role in understanding the behavior and origin of mobile malware. The key techniques and tools used include:

- **Static Analysis Tools:** Tools such as *JADX* and *APKTool* are used to decompile Android packages and examine their structure, manifest, and permissions.
- **Dynamic Analysis Tools:** *DroidBox* and *CuckooDroid* monitor runtime behavior in a sandbox environment, capturing API calls, file modifications, and network requests.
- **Memory Forensics:** Tools like *Volatility* analyze RAM dumps to extract hidden processes, strings, and injected code segments.
- **Network Forensics:** *Wireshark* enables packet capture and traffic inspection, which is particularly useful for tracing C2 traffic and data exfiltration.
- **Hybrid Platforms:** *MobSF* integrates static and dynamic analysis to provide a complete security assessment of Android and iOS apps.

Figure 4 illustrates the standard stages in analyzing mobile malware from initial static or dynamic inspection to memory forensics and threat classification. Tools such as APKTool, Wireshark, and Volatility support these phases by enabling code inspection, traffic analysis, and memory dump evaluation. A detailed comparison of these tools in terms of analysis type, usage, and platform compatibility is presented in Table 9.

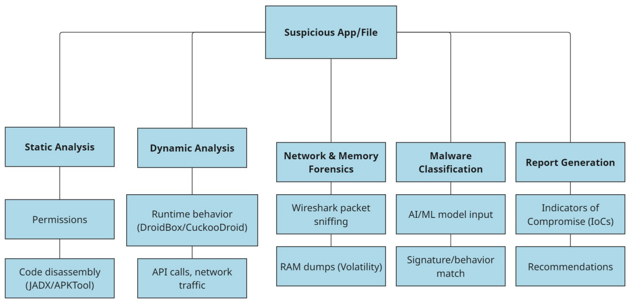
Suspicious App/File
Static Analysis
Permissions
Code disassembly (JADX/APKTool)
Dynamic Analysis
Runtime behavior (DroidBox/CuckooDroid)
API calls, network traffic
Network & Memory Forensics
Wireshark packet sniffing
RAM dumps (Volatility)
Malware Classification
AI/ML model input
Signature/behavior match
Report Generation
Indicators of Compromise (IoCs)
Recommendations

FIGURE 4: Mobile Malware Forensic Analysis Workflow

C. ANOMALY-BASED DETECTION

Anomaly-based detection emphasizes spotting behaviors to detect malware of normal actions undertaken by software applications or systems that can unveil unseen or novel threats

TABLE 7: Existing work in this field.

Ref.	Addressed Problems	Methodology	Results	Limitations
[26]	• Differentiate between benign apps and malicious apps in the Android operating system.	• Combines API calls with permission requests to better classification results. • Classify APIs into 3 groups to enhance the detection of malware apps in this environment. (the ambiguous group, the disruptive group, and the risky group). • Use two different datasets, one for malicious app samples, which include some reference datasets like AndroZoo, Contagio, MalShare, and VirusShare, and the other for benign app samples that were created and verified by the authors. • The proposed model contains 3 phases, which are the preprocessing, extraction, and grouping phases. Furthermore, it utilizes five ML algorithms, which are RF, J48, RT, kNN, and NB. All these algorithms were executed through fold cross-validation.	• The experiment results prove that malicious apps invoke several groups of API calls, and the malware of the mobile applications mostly asks for risky permissions in order to access users' sensitive data. • The proposed model shows its effectiveness in detecting malware in mobile apps and achieves 94.3% for F-measure. • The research findings show that permissions and API calls have a critical role in classifying variants of malware.	• The proposed model faces challenges in detecting malware that changes its behavior or uses obfuscation techniques. • The proposed model depends on permissions and API calls which may not be effective in detecting complex or new malware variants.
[27]	• A real-world mobile malware detection system that was deployed and implemented in Android T-Market is presented. • This system is called APICHECKER, which is based on machine-learning algorithms. • APICHECKER examines approximately 10K apps every day in order to detect malware apps.	• The study utilizes a data set containing 500k malicious and benign apps that were collected over 10 months from the Android Store. • Dynamic analysis is used to track runtime API invocations. • 9 ML algorithms are taken into consideration and finalized with the RF algorithm because it achieves high performance compared with other algorithms.	• RF achieved recall of 96.7% and precision of 98.6% . • Their proposed system is able to examine 10k apps with a processing time of 1.3 m/app on average.	• Difficulties in handling apps that used Java reflection methods to hide API calls.
[28]	• A novel method to detect mobile malware by extracting the rules from DNN using network traffic analysis was proposed.	• The study used a DroidCollector, which is a traffic data collector, to collect the dataset that contains 99,100 benign flows and 31,800 malicious flows. • The paper focuses on DNN to detect malware by extracting the rules and then compares this method with different ML methods like SVM, RF, and AdaBoost.	• Superior findings were achieved by the proposed method with 98.55% accuracy,98.27% recall, 97.93 % precision, and 98.04% F-measure.	• Complex models like CNN may outperform the proposed model due to their ability to handle and extract fine-grained features.

TABLE 7: *Cont.*

Ref.	Addressed Problems	Methodology	Results	Limitations
[29]	• A novel mobile malware detection method named DroidDetectMW was proposed to detect and classify malware in smart devices.	• A hybrid model that uses static and dynamic analysis was proposed. • The paper proposed a model that utilizes an enhanced DL model (ANN) by EHHO for classification. • Their model used different feature selection techniques like fuzzy metaheuristic, chi-square, and Fisher score. • The dataset used in this experiment contains 2980 benign samples and 1910 malware.	• Compared with traditional ML algorithms like SVM and RF, the ANN model proved its ability to improve detection performance and achieve a high accuracy rate.	• The negative side of the proposed model is that computational complexity is increased due to using hybrid analysis.
[30]	• A malware detection model that used dynamic analysis was proposed.	• The dataset contains 10 malware families: 3355 malware and 3462 legitimate applications. • The proposed model represents system call traces as images and then classifies these images by utilizing machine and DL algorithms like CNN, SVM, RF, and MLP as benign or malware.	• A high accuracy of 96.72% was achieved by using this model.	• Dynamic analysis requires a long time to be executed, and generating images and classifying them is computationally expensive.
[31]	• Inefficiency in malware detection on mobile devices due to high computational costs. • Low detection accuracy, leading to false positives and missed malware threats. • High resource consumption, which is unsuitable for resource-limited mobile devices. • Lack of practical, user-friendly malware detection tools for real-world use. • Difficulty in keeping up with new and evolving malware, making detection systems quickly outdated.	• MobiPCR system a cloud-based architecture that offloads the heavy detection tasks to the cloud, reducing the load on mobile devices. • CbTiFS (Classification-based TF-IDF Feature Selection): A novel feature extraction algorithm used to intelligently select and prioritize important features for malware detection. • Dynamic Ensemble Selection (DES): An ensemble learning technique to select the best classifiers based on the features, improving the detection accuracy.	• High detection accuracy of 99.43% achieved in detecting mobile malware. • Maintained a false-positive rate of 0.13%, outperforming existing malware detection systems.	• The system needs regular updates to stay effective against new and evolving malware. • Issues such as network latency and mobile power consumption could affect the system's overall efficiency in large-scale deployments. • MobiPCR relies on cloud-based detection, which may introduce delays or reliability issues in regions with poor connectivity.
[32]	• The lack of explainability in DL models used for mobile malware detection. Traditional models function as "black boxes," making it hard for security analysts to understand or trust the predictions, especially when distinguishing between different malware families.	• Using Grad-CAM to provide visual explanations for the decisions made by a DL model.	• The proposed model achieved an accuracy ranging from 96% to 97% when detecting malware and classifying samples into malware families. • When leveraging Grad-CAM and cumulative heatmaps, the model offers insights into how decisions are made, which enhances trust and usability for security analysts. • The proposed model tested on 8,446 Android samples, including six malware families and one family of trusted applications, demonstrating its effectiveness in both detecting malware and identifying its family.	• The model can only detect malware belonging to families present in the training data. It cannot generalize to new or unknown families without retraining. • Requires a large and diverse dataset to perform effectively. • The need to test how well the model handles obfuscated or modified malware that aims to evade detection.

TABLE 7: *Cont.*

Ref.	Addressed Problems	Methodology	Results	Limitations
[33]	• The increasing complexity and sophistication of Android malware make detection more challenging. • Modern, dynamic, and stealthy malware cannot be effectively detected by traditional antivirus and static analysis techniques. • The diversity of Android devices and OS versions creates challenges for developing universal malware detection systems. • Inefficiency in malware analysis and detection.	• The paper analyzes the latest trends in Android malware development, focusing on modern threats like trojans, and dropper tools. • Different malware detection approaches, including ML-based methods, antivirus systems, and permission-based approaches, are evaluated for their effectiveness in detecting evolving threats.	• Evaluation of detection techniques of hybrid approaches are more effective against modern threats.	• Requires large datasets and significant processing power. • Some methods are fast or lightweight for real-time malware detection on mobile devices.
[34]	• Ineffective detection methods against new and invisible malware • Static and dynamic analysis methods face limitations with obfuscation techniques and dynamically loaded malicious code.	• The sequences of API calls made by the application during its execution are represented as sparse arrays called API images. These images serve as fingerprints of the application's behavior over time. • Autoencoders are used to independently extract the most discriminative features from the API images without the need for labeled training data.	• This paper achieved over 90% accuracy and up to 97% F-measure in detecting malware across multiple datasets. • It performed well even when the training set was small, making it suitable for mobile environments with constrained resources.	• Autoencoders require fine-tuning and can be computationally intensive, which can impact real-time performance on resource-limited mobile devices. • The model performance improves with larger datasets. However, limited training resources have significant challenges.
[35]	• Increasing threats of iOS-targeted malware, especially in social media and online banking applications. • Limited in-depth analysis of iOS malware compared to Android platforms. • The increasing sophistication of iOS malware, such as zero-day exploits and jailbreak-related vulnerabilities.	• Conducted hybrid analysis (static and dynamic) of 12 malware samples in a controlled environment. • Integrated a phylogenetic analysis method to trace the evolution and origins of iOS malware. • Created 30 classification patterns based on identified functions and behaviors of the analyzed malware.	• An iOS malware detection model was created that effectively categorized malware into 30 distinct patterns. • This detection model was subsequently applied to 150 iOS applications, revealing that 4% of them were potentially susceptible to exploitation.	• Limited dataset which analysis conducted on 12 malware samples and 150 applications, which constrains the ability to generalize the results effectively. • A significant portion of the research concentrated on vulnerabilities and exploits unique to jailbroken iOS devices. • Potential for false positives: Certain applications might be incorrectly identified as malicious due to the use of shared functions. • Absence of real-time detection.
[36]	• A new highly reliable ML approach for malware detection with special considerations given to Android operating systems. • The research used the CCCS-CIC-AndMal-2020 dataset that incorporates a larger and updated set of malware classifications.	• To reduce the problem of dataset imbalance, methods like SMOTE and Spread-Sub-Sample were employed. • GAN and CFS are used as feature selection algorithms to sort out the best features. • The framework targeted dynamic features and assessed several ML classifiers such as RF, J48 DT, NB, and K-NN.	• Experimental outcomes revealed that the mean accuracy of RF was the highest, which is 98.9%. • As for model assessment, the 10-fold cross-validation was used.	• The approach may not be effective against stealth malware, which can avoid emulating actions within a sandbox. • The job of achieving real-time detection while maintaining competitively lightweight computation remains a crucial problem for the future development of the field.

TABLE 7: *Cont.*

Ref.	Addressed Problems	Methodology	Results	Limitations
[37]	• A novel ML framework for the detection of mobile malware using ML and DL for a better classification of malicious applications for mobile platforms. • The study was performed using four datasets, which include various kinds of malware and benign samples.	• In order to capture as many features as possible, both static and dynamic attributes were incorporated into the framework. • Feature selection techniques such as mRMR and ANOVA were used to select important features. • Various ML and DL algorithms incorporated in the evaluation include RF, SVM, Naïve Bayes, and J48, Deep Dense Network (DDN), 1D CNN, and CNN-LSTM.	• RF performed significantly well. • CNN-LSTM and 1D-CNN models, had slightly better F1 scores with values above 98	• Some of the challenges that the framework has are the time complexity related to feature extraction and the DL algorithm's high computational complexities. • Most of the features of the datasets used in developing the framework may not necessarily be the same as those of other datasets.
[38]	• A malware detection system based on the static analysis of Andriod applications, where the ML classifier was used to detect the new malware efficiently. • It builds on permission-based features derived from the application manifest files and uses two large datasets, Androzoo and Drebin, for testing.	• To improve the detection efficacy, they used PSO and Information Gain for feature optimization. • Several ML classifiers such as RF, MLP, kNN, J48, and AdaBoost were used, and to control class overfitting, 10-fold cross-validation was applied.	• Performance results were explored in the experimentation phase, with RF registering the best accuracy score of 91.6%. • Other classifiers, such as kNN and MLP, were also responsive, with an accuracy of more than 90%.	• The absence of permission-based features, which may neglect some behavior patterns. • The integration with other hybrid analysis approaches based on the mixture of static and dynamic methods was mentioned as a future development.
[39]	• A machine-learning model for Android malware detection in which app permissions were used as the primary attributes. • The ability to integrate this permission data into a flexible model from a dataset of about 30,000 Android applications is another goal of the study, which is considered essential for the accurate detection of both benign and malicious apps.	• The approach begins with data cleaning, where the dataset is preprocessed, later followed by EDA to discover potential patterns in this data set. • To avoid multicollinearity and enhance computational speed, the technique called PCA is employed for the reduction in dimensionality of the data. • Different classifiers, such as LR, DT, NB, RF, SVM, and MLP are used.	• Experimental results reveal that the RF classifier yielded the best accuracy.	• Identifying advanced and complex malware variants that cannot be easily addressed by static analysis techniques. • The real-time use of the model is prohibited too because it does not work with dynamically created features, such as real-time malware behavioral changes.
[40]	• A new approach to Android malware detection and categorization using CNN. • The research made use of useful benign samples derived from the Google Play Store and suspicious samples derived from public APIs and the CICAndMal2017 dataset.	• The method uses a static analysis method, whereby binaries are translated to 2D arrays, and the Android application APK files are converted to grayscale images. • These images capture similar structures between different malware groups to provide better distinction between classes. • The proposed methodology used two algorithms, to begin with, the CNN for basic classification and the kNN for the image vector examination.	• The first architecture of CNN was able to yield 84.9% accuracy, while a much sparser architecture had only 68.1% accuracy, emphasizing the benefits of deeper architectures.	• limited dataset size due to storage and computational constraints. • Only achieved the binary classification of benign and malware without subclassification to various families of malware.

TABLE 7: *Cont.*

Ref.	Addressed Problems	Methodology	Results	Limitations
[41]	• An Android malware detection technique that is based on static feature analysis using ML algorithms. • This research work has made useful contributions by introducing updated CICInvesAndMal2019 datasets that improve the existing approaches of static features and malware classification and RFE for improvement in the detection.	• The approach uses Android permissions and API calls and three ML algorithms, which are SVMs, KNN, and NB for classification.	• The findings indicate that SVM delivers the highest detection accuracy rate of all the algorithms, standing at 94.36%.	• The inability of static analysis to capture the dynamic behaviors of malware as well as a small dataset to increase the practical applicability are the main shortcomings of the study.
[42]	• The utilization of ML in the enhancement of malware detection on smart mobile devices based on dynamic behavioral analysis.	• This research evaluated the three ML algorithms, namely LightGBM, XGBoost, and RF, using the CICMalDroid 2020 dataset. • The process of selecting features was done using SelectFromModel, which gives the features according to their importance. • The malware was categorized based on dynamic system call frequencies.	• The LightGBM model performed the best with an accuracy of 93.95% and a precision of 94.1%.	• However, study limitations of imbalanced data sets and dependence on outdated tools offered only limited analyses. • The false positive issues, which label benign applications as malware, are concerns about user trust and operational concerns as well.
[43]	• A comprehensive multiple-attribute decision-making approach to detect Android malware through the combination of ML and fuzzy AHP for risk evaluation. • Permission-based features of the Android system are utilized for static analysis for the current study.	• It divides the applications into four risk levels, very low, low, medium, and high, designed to inform the user of security risks and achieve a high detection rate of 90.54% malware. • The dataset consists of the ten thousand samples collected from the Drebin and AndroZoo databases. • With the help of Information Gain, a selection of the most important permission-based features was introduced, and the dataset was reduced to increase performance.	• The complete model was able to detect the malware with 90.54% accuracy, and risk levels can be accurately classified through a box plot analysis. • Fuzzy AHP provided consistent and accurate evaluations with a ratio of consistency ($CR < 0.1$).	• The study was confined to permission-based features only, and other forms of static attributes such as Java code and intent filters, which could further improve detection, were not included in the study.
[44]	• The paper addressed the increasing threats of mobile malware, especially in Android environments, by proposing an end-to-end framework for detection, tracing, and propagation study. • The proposed system also combines static and dynamic analysis techniques with a ML algorithm to detect and categorize the types of malware. • One contribution includes creating a knowledge map and a 'gene recognition' system on the malware families to track their spread and facilitate identification in different network structures.	• The experiment was carried out using a dataset of 178,155 real mobile malware samples from China Unicum's network. • The work builds behavioral graphs to aggregate related malware classes and distill out common characteristics. • It uses ML algorithms, including KNN, DT, and SVM, for malware detection and classification.	• The integration of the behavior clustering was found to actually enhance the efficacy of the family classification of malware when compared to the graph-based approach that decisively helped minimize false positives.	• The dynamic analysis and graph clustering will, in general, be computationally expensive. • Inadequacies of the framework are observed when it comes to dealing with the constantly emerging malware that targets new vulnerabilities in the IoT and 5G networks.

TABLE 7: *Cont.*

Ref.	Addressed Problems	Methodology	Results	Limitations
[45]	• The work introduces a new approach that combines static and dynamic analysis with ML models with optimized hyperparameters and cross-validation to improve detection rates.	• The dataset employed in the study has 398 records and 331 features acquired from the domain of analyses of Android malware. • This performance analysis was performed on four types of ML models, including XGB, RF, SVM, and DT, in detecting malicious applications.	• RF obtained the highest accuracy at 99%.	• Non-application of real-time evaluation of the model and a small dataset, which does not work with more data. • The mechanism of how to use hybrid analysis for feature selection is not being discussed in detail.
[46]	• The rise in malware attacks targeting IoT and Android devices is taking advantage of weaknesses in hardware, software, and network layers. • The absence of thorough integration of protective measures within the IoT and Android environments.	• Examination of malware through static, dynamic, and hybrid detection methodologies. • Employs API calls, permissions, system calls, application metadata, and image analysis. • Tools such as TinyDroid, DroidSieve, and NSDroid utilize machine learning techniques including Random Forest, SVM, K-Nearest Neighbors, and neural networks. • Integrates image classification via convolutional neural networks and behavioral analysis through monitoring of system and user-space activities.	• High accuracy rates in malware detection were attained, frequently surpassing 95%, with certain hybrid methods achieving levels as high as 99%. • Image-based methodologies utilizing convolutional neural networks demonstrated outstanding effectiveness in malware classification. • Hybrid techniques that integrated both static and dynamic analysis, providing an optimal balance between accuracy and robustness.	• Lack of specific vulnerability classifications for IoT, as well as a need for better alignment with established guidelines such as OWASP and NIST. • Some of the suggested methods require significant computational power and resources, which can make them difficult to implement in practical scenarios. • The essential to develop scalable and lightweight solutions to effectively tackle new threats in various IoT and Android settings.
[47]	• The swift rise in IoT malware threats can be attributed to the extensive proliferation of IoT devices that possess constrained hardware capabilities and inherent security weaknesses.	• Implement dynamic analysis within a cloud-based virtual environment, where features such as memory usage, network activity, and system calls are extracted, transformed into images, and subsequently classified using a convolutional neural network (CNN) model, specifically ZFNet. • This involves debugging, feature extraction, image pre-processing, and automated classification through the CNN.	• Attained an accuracy rate of 99.28%, accompanied by a minimal False Positive Rate of 0.63% and a False Negative Rate of 0.72%, surpassing the performance of current models.	• The system relies on cloud resources, faces challenges with malware that can bypass virtual environments, and necessitates considerable computational power.
[48]	• The effective learning of feature representations from unprocessed data. • The minimization of dependence on pre-existing knowledge or manual feature engineering.	• A data-flow dataset comprising 3,733 features from the MUD-FLOW project, characterized by an imbalance with approximately 17,897 samples. • A dataset of permissions and suspicious API calls gathered from benign and malicious applications between 2019 and 2020, containing 645 features.	• Dataset one Stacked Hybrid Learning MSAE and SDAE (SHLMD) achieved an accuracy of 94.46%, surpassing other methodologies. • For second dataset SHLMD attained an accuracy of 90.57%.	• The second dataset's relatively small size may restrict generalization. • The limited feature sets arise from constraints in available tools and technology, such as absent network addresses and function call graphs. • Additional evaluation on highly imbalanced datasets is required to validate the findings.
[49]	• The increasing incidence of Android malware attributed to its open-source framework. • The challenges in identifying novel and advanced malware techniques that frequently bypass conventional detection systems.	• A diverse array of datasets was utilized across various studies, typically , Applications categorized as either benign or malicious. - Features derived from API calls, permissions, and behavioral logs.	• End-to-end deep learning models achieved accuracy rates of up to 93%. • Hybrid methods that utilize both static and dynamic features demonstrated superior resilience against sophisticated malware.	• Certain studies faced constraints due to small or unbalanced datasets, which restricted the applicability of their findings. • There exists a challenge in modifying models to identify newly developed and obfuscated malware.

TABLE 7: *Cont.*

Ref.	Addressed Problems	Methodology	Results	Limitations
[50]	• The increasing complexity and advancement of mobile malware, especially on Android devices, which aim to compromise sensitive user information and exploit vulnerabilities within the device. • Need for Comprehensive Analysis	• A total of 243 papers were examined through scientometric techniques, concentrating on the development.	• Initial malware, such as Cabir, utilized Bluetooth and basic propagation techniques. Contemporary malware adopts sophisticated methods, including code obfuscation, zero-day vulnerabilities, and compatibility with IoT systems.	• The analysis is confined to existing research and may not encompass all variants of malware. • There is a restricted examination of other platforms, such as iOS. • Dynamic Nature of Malware •
[51]	• The escalating risks associated with mobile malware due to the advent of 5G technology and the proliferation of mobile applications. • Existing detection methods encounter difficulties such as recognizing zero-day malware, managing obfuscation techniques, and ensuring scalability.	• Reviewed 154 studies published between 2016 and 2021. • Emphasized mobile malware detection techniques that utilize machine learning.	• Supervised learning techniques, such as Support Vector Machines (SVM) and Random Forest, demonstrated superior effectiveness compared to unsupervised methods. • High accuracy in malware detection was achieved through the use of feature-rich datasets and optimized learning algorithms.	• Insufficient availability of diverse and high-quality datasets. • Challenges associated with processing large volumes of data. • Challenges in identifying newly emerging and obfuscated malware. • Supervised learning models rely on labeled data, which may limit their generalizability.
[52]	• The increasing diversity, complexity, and impact of computer and mobile malware.	• Malware detection methods were categorized into static, dynamic, vision-based, and hybrid analyses. • An examination of datasets, detection methodologies, and the challenges associated with real-time detection was conducted.	• Machine Learning Integration improved detection accuracy and minimized false positives through adaptable algorithms. • Vision-Based Methods innovative techniques utilizing image representations of malware for detection purposes.	• Malware evolves to circumvent sandbox environments and static detection systems. • Scalability of managing large volumes of data continues to pose significant challenges. • A deficiency of diverse, labeled datasets hampers accurate evaluation. • Automated detection systems are not entirely dependable and still necessitate manual verification.
[53]	• The significant rise in Android malware that takes advantage of security vulnerabilities through tactics such as malicious payloads and application tampering.	• Integrates static features (API calls and permissions) with dynamic features (system calls) to enhance the accuracy of malware detection. • Employs a Tree-Augmented Naive Bayes (TAN) model to identify and model the conditional dependencies among features, thereby mitigating multicollinearity.	• Achieved an accuracy rate of 97% with minimal false positive and negative rates. • Showed resilience across a variety of datasets from 2010 to 2019. • Surpassed existing static, dynamic, and hybrid methodologies in detecting Android malware, particularly in relation to evolving threats.	• Certain malware can evade detection by utilizing adversarial strategies or by circumventing runtime execution. • Reliance on labeled datasets; the evolving nature of malware necessitates frequent retraining with updated samples. • Increased computational demands compared to simpler detection models due to the complexities of hybrid analysis.
[55]	• The increasing prevalence of malware targeting Android devices, coupled with the inadequacies of current detection methods regarding dataset diversity, performance, and the ability to generalize to new malware instances.	• A hybrid approach to malware detection that leverages machine learning techniques. • Incorporates static, dynamic, and hybrid analyses, utilizing the Omnidroid dataset for training. • Employs neural networks across static, dynamic, and hybrid categories for predictive analysis. • Features a client-server architecture for the extraction and analysis of features.	• Static model of 92.9% accuracy (an increase from 89.3%). • Dynamic model of 81.1% accuracy (an increase from 78.6%). • Hybrid model of 91.1% accuracy (an increase from 89.7%). • Static features decreased from 25,999 to 840. • Dynamic features decreased from 5,932 to 3,722.	• Reliance on the quality and diversity of the dataset. • Possibility of diminished performance when encountering entirely new malware not included in the dataset. • Challenges in implementation related to the client-server architecture and ensuring smooth operation across various Android devices. • Limited assessment of the model's effectiveness in real-world deployment scenarios.

TABLE 7: *Cont.*

Ref.	Addressed Problems	Methodology	Results	Limitations
[56]	• The study discusses how adversarial attacks can affect DL-based modulation classifiers in real-world, unsynchronized signal scenarios.	• It suggests IC-UAP and physical IC-UAP algorithms that use the Adam optimizer and projected gradient descent to create class-specific perturbations.	• The suggested approaches outperform current adversarial techniques in a range of SNR and PNR settings and drastically reduce classifier accuracy (by up to 50%).	• Attacks become less effective when time shifts randomly occur, indicating that physical environments are sensitive to signal unsynchronization.
[54]	• The swift rise of Android malware that takes advantage of third-party app stores and employs code obfuscation techniques.	• Integrates static features (such as permissions and API calls) with dynamic features (including system calls and behavioral traces). • Employs machine learning and deep learning algorithms for malware detection, utilizing Support Vector Machine (SVM), Random Forest, Gradient Boosting (GB), and Multi-Layer Perceptron (MLP). • Implements oversampling (SMOTE) to balance class distributions in the dataset and applies Principal Component Analysis (PCA) for feature reduction. • Utilizes the Malware Genome and Drebin projects, comprising 18,835 samples. • Incorporates the CICMalDroid dataset, which contains 17,341 samples.	• The hybrid analysis improved detection accuracy by 3-5% compared to using static or dynamic methods in isolation. • The Gradient Boosting (GB) model achieved the highest accuracy at 99.3% and the shortest prediction latency. • Notable enhancements in resource efficiency were realized through the optimized hybrid analysis.	• Hybrid models, particularly those incorporating dynamic features, necessitate substantial computational resources and may not be suitable for real-time detection on devices with limited resources. • Malware employing adversarial strategies or circumventing dynamic feature extraction may successfully evade detection. • The performance of the model is contingent upon the quality and diversity of the dataset, which may not always be guaranteed.

TABLE 8: Comparative Overview of mobile malware detection approaches

Ref.	Algorithms	Feature Selection Processes	Model Architectures	Evaluation Metrics
[26]	Random Forest, J48, Naïve Bayes, k-NN, Random Tree)	Permissions and API calls grouped into ambiguous, risky, and disruptive categories	A classification model combining grouped API calls with permission features	F-measure (94.3%) using cross-validation on 27,891 Android app samples
[27]	Random Forest, SVM, k-NN, Naïve Bayes	Dynamic API invocation tracking; 426 critical APIs selected; static features from permissions and intents	APIChecker system; hybrid architecture combining static and dynamic features; scalable for app market deployment	Precision (98%), Recall (96.7%), Average detection time (1.3 min/app), 10k apps processed daily
[28]	Explainable Deep Neural Network (EDNN)	Behavior-based features from dynamic analysis logs using system calls, API sequences, and memory usage patterns	Attention-based DNN architecture with explainability layer to interpret model predictions	Accuracy: 98.6%, Precision: 98.2%, Recall: 98.7%, F1-score: 98.4%, evaluated on a real-world Android malware dataset
[29]	Hybrid model combining Random Forest and Deep Neural Network	Static and dynamic features selected using Information Gain and mutual correlation analysis	Hybrid intelligent framework integrating RF for feature reduction and DNN for final classification	Accuracy: 99.12%, Precision: 99.10%, Recall: 99.14%, F1-score: 99.11%; tested on CIC-InvesAndMal2019 dataset
[30]	CNN applied to image representations of system calls	System call traces transformed into grayscale images, implicitly encoding behavioral patterns	CNN architecture designed to process image-based input from dynamic execution traces	Accuracy: 96.72%, Precision: 96.30%, Recall: 96.80%, F1-score: 96.55%;
[31]	Random Forest, Gradient Boosting, Logistic Regression, and ensemble voting	Permission-based and behavior-based features selected using principal component analysis (PCA) and mutual information	MOBIPCR: Ensemble classifier integrating multiple ML models for strict and accurate malware classification	Accuracy: 99.43%, Precision: 99.32%, Recall: 99.44%, F1-score: 99.36%; evaluated on Drebin and Androzoo datasets
[32]	Deep Neural Network (DNN) with interpretability via SHAP	API call sequences and system events extracted from dynamic analysis logs	Interpretable DNN model for both malware detection and family classification using SHAP for feature contribution	Accuracy: 97.51%, Precision: 97.60%, Recall: 97.40%, F1-score: 97.50%; tested on labeled malware families dataset

TABLE 8: *Cont.*

Ref.	Algorithms	Feature Selection Processes	Model Architectures	Evaluation Metrics
[34]	Autoencoder	API call sequences converted into images	Autoencoder detects behavioral anomalies in API-image space	Accuracy: 94.7%, Precision: 94.3%, Recall: 95.0%, F1-score: 94.6%
[36]	Random Forest	Static features (permissions, intents) selected via ranking and correlation	Lightweight RF-based ML detection framework	Accuracy: 98.5%, Precision: 98.2%, Recall: 98.6%, F1-score: 98.4%
[37]	Random Forest, SVM, XGBoost	Static + dynamic features selected using IG and PCA	Comparative framework with hyperparameter tuning	Accuracy: 99.23%, Precision: 99.20%, Recall: 99.25%, F1-score: 99.22%
[38]	RF, Naive Bayes, SVM, Decision Tree	Static permissions and manifest attributes	Traditional ML pipeline based on Android manifest file	Accuracy: 94.3%, Precision: 93.7%, Recall: 94.9%, F1-score: 94.3%
[39]	SVM, Decision Tree, Random Forest	Features extracted using APKtool and static disassembly	ML classifiers compared for malware detection efficiency	Accuracy: 96.2% (Random Forest), best among tested models
[40]	Convolutional Neural Network (CNN)	Features from APK files converted into binary image representations	CNN-based classifier for detection and multiclass classification	Accuracy: 97.6%, F1-score: 97.2%, tested on balanced dataset
[41]	Random Forest, k-NN, SVM	Static features from permissions, intents, and components	ML classification pipeline with cross-validation	Accuracy: 95.8% (RF), Recall: 95.5%, Precision: 96.1%
[42]	Decision Tree, SVM, Random Forest	Features extracted from dynamic runtime behavior logs	Comparison of detection performance on dynamically executed apps	Accuracy: 94.7%, F1-score: 94.1%
[43]	Fuzzy AHP with ML algorithms	Multi-criteria evaluation of static features (permissions, code structure)	Hybrid Fuzzy AHP-based scoring + ML detection	Accuracy: 93.4%, enhanced interpretability reported
[44]	Supervised ML and rule-based propagation models	Traceability features + propagation flow patterns	Framework combining traceability, detection, and behavior profiling	Detection rate: 95.2%, emphasis on threat evolution and tracking
[45]	Random Forest, SVM, k-NN, Naive Bayes	Static features from APKs and dynamic system logs	Hybrid comparison of static and dynamic analysis in ML context	Accuracy: 96.8%, Precision: 96.5%, F1-score: 96.6%
[46]	Hybrid ML-based detection + defensive modeling	Feature selection based on behavior logs and system call analysis for IoT and Android	Proposed framework combining malware detection and anomaly-based defense	Accuracy: 96.4%, Precision: 96.1%, Recall: 96.5%, F1-score: 96.3%
[47]	Convolutional Neural Network (CNN)	Behavior patterns from dynamic execution, including network, memory, and system state	CNN architecture trained on dynamically executed IoT malware samples	Accuracy: 98.2%, Precision: 97.9%, Recall: 98.3%, F1-score: 98.1%
[48]	Hybrid static-dynamic ML model using Decision Tree, Random Forest, SVM	Combined static and dynamic features (permissions, behavior logs, system events)	Comparative hybrid detection framework to improve ransomware classification	Accuracy: 97.9%, Precision: 97.6%, Recall: 98.0%, F1-score: 97.8%
[49]	Comparative study of ML algorithms (SVM, RF, NB, DT)	Behavioral features extracted from Android APKs and runtime monitoring	Survey-based comparative framework analyzing detection techniques	Accuracy: 95.2% (RF), Recall: 94.8%, Precision: 95.5%, F1-score: 95.1% (varies by classifier)
[53]	Tree-Augmented Naive Bayes (TAN)	Hybrid features from static code structure and dynamic API behavior	TAN-based hybrid classifier combining static and dynamic indicators	Accuracy: 97.4%, Precision: 97.1%, Recall: 97.5%, F1-score: 97.3%
[54]	Hybrid ML-based approach	Static + dynamic feature set including permissions and behavioral signatures	Hybrid detection system emphasizing performance efficiency and low resource usage	Accuracy: 96.1%, F1-score: 95.9% (tested on custom Android malware samples)
[55]	Brainshield: Hybrid ML combining Random Forest and k-NN	Features from static disassembly and runtime system logs	Multi-phase hybrid model integrating ensemble learning and behavior tracing	Accuracy: 98.9%, Precision: 98.6%, Recall: 99.1%, F1-score: 98.8%

that have not been recognized before. Also known as zero-day threats in the cybersecurity realm. ML and AI prominently feature in this area by utilizing models that are educated on how legitimate applications behave to point out any deviations as security risks. This method is flexible and efficient at tackling forms of malware. However, it may lead to a number of false alarms compared to conventional signature-based detection techniques [66].

D. STATIC VS. DYNAMIC ANALYSIS

Analyzing malware code without running it is known as static analysis. It is an effective method, particularly useful for scanning numerous files at once. However, it faces challenges with obfuscated or encrypted malware instances. Signature-based approaches represent a form of static analysis but are inherently constrained in identifying only already-known attacks. In contrast to static analysis methods that examine malware without execution, dynamic analysis involves running malware in a controlled environment like a sandbox to observe its real-time behavior. This approach is effective in bypassing obfuscation tactics employed by malware. However, dynamic analysis demands significant resources, and it may be ineffective against malware that possesses anti-sandbox or anti-virtualization capabilities. Merging static and dynamic analyses is known as hybrid analysis, which enables detection capabilities to address the shortcomings of each approach individually. These combined strategies have the potential to enhance detection precision for discreet malware threats [67].

TABLE 9: Comparison of Mobile Forensic Tools

Tool Name	Type of Analysis	Main Usage	Platform
JADX [58]	Static	APK decompilation to Java source	Android
APKTool [59]	Static	Reverse engineering APK resources	Android
Androguard [60]	Static + Code Analysis	Permissions and control-flow analysis	Android (Python-based)
DroidBox [61]	Dynamic	Monitor runtime behavior of APKs	Android (AVD)
CuckooDroid [62]	Dynamic	Malware sandboxing	Android
Wireshark [63]	Network	Packet capture, analyze C2 traffic	Cross-platform
Volatility [64]	Memory	RAM dump analysis	Android (rooted), Linux
MobSF [65]	Hybrid	Static + dynamic app security analysis	Android, iOS

E. HEURISTIC-BASED DETECTION

The heuristic approach aims to recognize malware by examining how it operates and behaves using rules instead of fixed signatures. These approaches are capable of detecting unfamiliar or unidentified malware by spotting unusual traits commonly associated with malware. These heuristic approaches may encounter difficulties like false positives and the difficulty of detecting complex obfuscated malware [68].

F. ML AND DL-BASED DETECTION

ML techniques, particularly DL, are now commonly applied in detecting malware in mobile devices due to their capability to analyze extensive data sets and identify intricate patterns in how applications operate. These approaches involve spotting anomalies in data sets and utilizing classification techniques as well as DNN. DL systems encounter challenges such as data availability and the complexities involved in selecting features and training the model effectively while maintaining a balance between accuracy and minimizing false positive rates to prevent inundating users with unnecessary alerts [69].

G. CLOUD-BASED DETECTION

Cloud-based detection systems shift the burden of malware detection to cloud platforms. This will help to analyze these applications in the space for real-time detection and quicker analysis of extensive data sets without overloading mobile device resources. It became more prevalent due to its adaptability and effectiveness in managing evolving malware strains [70].

This analysis of detection techniques offers an insightful view for comprehending how modern tools react to threats from sophisticated malware. A more comprehensive discussion of the findings, conclusions, and possible lines of inquiry derived from this assessment is provided in the section that follows.

VII. MOBILE MALWARE ANALYSIS TECHNIQUES

After detection, forensic analysis plays a crucial role in understanding attack impact, origins, and propagation. This section explores tools and techniques used in mobile malware investigations.

Mobile malware analysis is the examination and prevention of the behavior and characteristics of mobile device attacking programs. Approaches in this domain are reverse engineering wherein one disassembles the malware and studies its behavior, behavioral analysis which tools analyze the behavior of a program and identify that it has malware, and forensic analysis wherein one gathers evidence of an infection. Malware has become more sophisticated, and this makes it difficult for researchers as well as practitioners to deal with challenges such as; obfuscation, evasion, and the detection of zero-day attacks.

A. STATIC (REVERSE ENGINEERING)

Reverse engineering is at the base of the methodologies for analyzing mobile malware. It is a process of taking an application apart and analyzing characteristics such as internal flow structures, architectures of the used packers and/or obfuscators, and code payloads. Through reverse-engineering the malware, analysts are able to note its functionality, purpose, and weaknesses [71].

- **Key Objectives:**
  1) Code Analysis: Looking for suspicious patterns for coding like encodings, credentials in the code, C2 URLs, etc.
  2) Payload Identification: Separating parts that are involved in the execution of various damaging uses, for instance, information leakage or unauthorized access.
  3) Obfuscation Analysis: Elimination of disguising and concealing forms such as control flow flattening and string encryption schemes.
- **Tools and Frameworks:**
  -- APKTool: Used to decode APK files for a better methodology of scrutinizing resources and manifest files.
  -- JADX: APK files are decompiled to provide a readable form of Java source code.
  -- Ghidra: A holistic approach to binary file reverse engineering that aims at the Windows, Linux, and OSX platforms.
- **Importance:** The process is very important in understanding the potential threats posed by malware as well as in designing means of protection against it. It also helps create signatures for the installation of antivirus tools.

B. DYNAMIC (BEHAVIOR ANALYSIS)

It is a process whereby the dynamic behavior of malware is analyzed in a controlled or live environment. This technique of detection centers on how the malware functions in relation to the system, the network, and the data it is expected to work with while performing its given tasks [72], [73].

- **Key Techniques:**
  1) Network Traffic Monitoring: Monitors stripped down sent and received messages to block evil communication with C2 servers or unauthorized information exchange.
  2) System Call Analysis: Supervises the use of system calls to observe the interaction between the malware and the operating system.
  3) Dynamic Permission Analysis: Discusses how permissions are used at runtime to determine when a request can be deemed suspicious.
- **Tools and Frameworks:**
  -- Cuckoo Sandbox: It is an open-source malware analysis system that provides an environment for running and, therefore, studying the behavior of the malware.
  -- TaintDroid: A real-time monitoring tool to track the usage of what is defined as sensitive data in Android applications using dynamic analysis.
  -- Wireshark: Examines network traffic and then identifies irregularities and potentially suspicious patterns.
- **Applications:** Behavioral analysis is to help find malware behavior that is likely to be missed during static analysis, such as encrypted payload carrying, evasion techniques, or a delayed action.

C. FORENSIC ANALYSIS

Forensic analysis is the examination of infected devices after the attack with the main aim of looking for other signs of intrusion and trying to reconstruction of the malware activities. This technique is important when determining the scope and effect of an attack [74], [75].

- **Key Techniques:**
  1) Data Recovery: It is possible to retrieve some files that have been either deleted or hidden and may contain malware.
  2) Memory Dump Analysis: Analyzing memory dumps infected with malware that exist only in runtime.
  3) Network Forensics: Analyzing network activities in order to detect how malware got into the network and which external servers are engaged.
- **Tools and Frameworks:**
  -- FTK Imager: A digital forensic tool that enables one to make images of devices and analyze the objects contained within them.
  -- XRY: Focused on mobile device data extraction and decoding, it retrieves data from hacked systems.
- **Significance:** The timeline and methodology of an attack can only be understood by doing a forensic analysis. It is also utilized in courts as proof.

VIII. MOBILE MALWARE ANALYSIS TOOLS AND FRAMEWORKS

This section reviews tools and methods used for analyzing mobile malware. It emphasizes the importance of combining static and dynamic analysis techniques to effectively detect and understand malware threats on mobile devices.

A. TOOLS FOR STATIC ANALYSIS

Static analysis refers to the process of reviewing the source code or binary of a software application without running it. This technique is employed to uncover vulnerabilities, coding mistakes, or harmful code by scrutinizing the code's structure, syntax, and logic. Static analysis tools typically function by decompiling the code into a format that is easier to read, facilitating a thorough investigation of its elements, including control flow, dependencies, and API utilization. This approach is especially proficient in identifying problems such as hardcoded credentials, insecure configurations, and vulnerabilities present at the code level before deployment. The key points for static analysis tools include:

- **Android Package Tool** is an advanced utility utilized for the reverse engineering of Android APK files. It allows for the extraction and decoding of both resources and bytecode, thereby enabling modifications and the reassembly of the APK. By reconstructing the application's resources and code, researchers can investigate app behaviors, uncover potential vulnerabilities, and analyze malicious intents embedded within Android applications. This functionality renders APKTool an indispensable resource for static analysis and malware research [76].
- **JADX** is a decompiler that converts Android application bytecode into a more comprehensible Java source code. It is widely employed in static analysis to examine the structure and functionality of Android applications. By facilitating straightforward navigation through an application's code, JADX assists researchers and analysts in detecting malicious patterns, unintended functionalities, and code obfuscation techniques. It is a crucial tool for static malware analysis and source code evaluation [77].
- **Androguard** is a robust Python-based framework tailored for the static analysis and reverse engineering of Android applications. It provides functionalities for disassembling, decompiling, and analyzing APKs, with a focus on bytecode examination and permission analysis. Androguard includes advanced features such as graph-based analysis, which is essential for visualizing the interrelationships within an application's architecture. This capability aids in identifying malicious activities and comprehending app behavior [78].

B. TOOLS FOR DYNAMIC ANALYSIS

Dynamic analysis is a method employed to assess software by monitoring its behavior while it is running. In contrast to static analysis, which reviews the code without execution, dynamic analysis offers valuable insights into an application's behavior during runtime, encompassing interactions with system elements, network communications, and data flows. This methodology is essential for detecting security vulnerabilities, including harmful payloads and privacy violations, that may not be visible through static code examination. The tools that are frequently utilized in the dynamic analysis of Android applications include:

- **CuckooDroid** serves as an extension of the Cuckoo sandbox specifically designed for the analysis of Android applications. It establishes a secure virtual environment in which applications can be executed without risk, allowing for the monitoring and documentation of their runtime activities. The tool meticulously records comprehensive logs detailing various actions, including API calls, file alterations, and network communications. These features render CuckooDroid an essential asset for detecting malicious behaviors, comprehending malware operations, and producing forensic documentation. Its capacity to replicate and scrutinize harmful actions provides profound insights into application performance, thereby aiding in the formulation of robust security strategies [78].
- **DroidBox** is focused on observing the runtime activities of Android applications, providing an in-depth examination of data leaks, file system interactions, cryptographic functions, and network communications. Its graphical interface enables researchers to monitor behavioral variations over time, thereby assisting in the detection of potential threats. The tool has gained significant traction in both academic and industrial settings due to its user-friendly nature and efficiency in dynamic behavior analysis. By identifying critical issues, such as unauthorized data transfers, DroidBox contributes to enhancing the security and privacy of applications [78].
- **TaintDroid** is a dynamic taint tracking system specifically developed for the Android platform. It enables real-time surveillance of sensitive data flows within applications, effectively identifying any instances of misuse or leakage of user information. By labeling sensitive data and tracking its trajectory throughout the application, TaintDroid uncovers privacy infringements, including unauthorized data transfers to external servers. Its incorporation into the Android operating system guarantees a high level of accuracy and dependability, rendering it an essential instrument for research focused on privacy preservation and adherence to regulatory standards [78].

Every tool is essential for comprehending application behaviors, enhancing application security, and safeguarding user privacy.

C. HYBRID TOOLS

Hybrid analysis integrates both static and dynamic analysis to capitalize on the advantages of each approach. Static analysis entails reviewing source code or binaries without executing them, while dynamic analysis focuses on monitoring the application's behavior during runtime within a controlled setting. By merging these methodologies, hybrid analysis provides a more thorough framework for detecting malware, offering insights at both the code level and in terms of runtime behavior. This synthesis effectively mitigates the limitations associated with each individual technique, such as static analysis's inability to uncover threats that manifest only during execution and the high resource demands of dynamic analysis. Here is a description of the hybrid tools shown below:

- **Mobile Security Framework (MobSF)** is a powerful hybrid tool tailored for extensive security evaluations and malware analysis of mobile applications, encompassing both Android and iOS platforms. It initiates the process with static analysis, which involves decompiling the source code and scrutinizing elements like manifest files and permissions. Subsequently, dynamic analysis is performed using an integrated emulator to monitor the application's behavior during runtime. This two-phase methodology facilitates the identification of vulnerabilities that might be overlooked when relying solely on one method. The adaptability of MobSF and its compatibility with various platforms contribute to its widespread use in research and security assessments [79].
- **SandDroid** represents another hybrid tool utilized for malware detection, employing both static and dynamic analysis techniques. During the static phase, it extracts features such as permissions and API calls, which are then integrated with insights gained from runtime behavior analysis. This combination enhances the tool's capacity to classify applications and detect potentially harmful activities more efficiently. SandDroid has played a significant role in the evolution of hybrid analysis techniques, particularly in the realm of Android malware detection [79].

IX. CHALLENGES AND OPEN ISSUES

There are still many challenges and open issues in the area of mobile malware analysis and its mitigation. These challenges are due to the advanced technology in malware, the drawbacks in existing approaches, and other issues that concern the mobile environment. These factors are significant to overcome in order to enhance the security of mobile devices.

- **Evasion techniques:** Mobile malware uses different ways of hiding, such as code hiding, code encryption, and compression, as well as anti-debugging and sandbox awareness. Obfuscation conceals malware's processes, and encryption conceals its operations until the program runs. Malware capable of self-modifying typically conceals its nature from the analyst and can identify and eliminate sandbox environments or debugging tools. These strategies pose a new threat to the conventional approaches to static and dynamic analysis in that they are difficult to decipher even at runtime using techniques such as de-obfuscation and monitoring [80], [81].

- **Adversarial attacks on ML-based systems:** AI security systems are vulnerable to adversarial attacks due to manipulation of the detection mechanisms of ML-based malware detection systems. Evasion attacks modify malware characteristics in a rather sneaky manner so as to avoid being detected by classifiers while poisoning attacks compromise training data so as to decrease the performance of models. So, adversaries can then look at the same model to reverse-engineer it and find other weaknesses. These threats decrease the dependability of the present-day ML systems, which demand high availability solutions such as adversarial training and an optimistic approach to detection techniques that can improve the overall system security [73], [82].
- **Resource constraints on mobile devices:** In the context of malware analysis, constraints imposed by the deployment of mobile devices become a concern. Behavioral monitoring, or sandboxing, is a kind of dynamic analysis tool that consumes more CPU, memory, and battery, and of course, it has performance overhead. Another challenge to local malware analysis results from storage constraints. These constraints, however, can potentially be addressed by employing cloud-based solutions and optimizing detection algorithms to reduce resource utilization [74], [75].

X. RECENT TRENDS AND FUTURE DIRECTIONS

This section examines the profound influence of Artificial Intelligence (AI) and Machine Learning (ML) on the detection of mobile malware. It emphasizes the enhancements brought about by the incorporation of these technologies, which have significantly increased both the precision and efficiency of identifying mobile malware threats.

A. THE IMPACT OF AI AND ML ON MOBILE MALWARE DETECTION

The integration of Artificial Intelligence (AI) and Machine Learning (ML) has greatly improved the detection of mobile malware, boosting both accuracy and speed. ML algorithms are capable of processing large datasets and uncovering intricate patterns in mobile applications, allowing for the identification of both familiar and unfamiliar malware. Techniques such as dynamic and hybrid analysis facilitate real-time monitoring of application behavior, leading to higher detection rates. For example, methods like Support Vector Machines (SVM) and neural networks have shown effectiveness in spotting anomalies and harmful patterns in how applications operate [83], [84].

Moreover, AI-based approaches offer scalability and automation, which are essential for managing the increasing volume of mobile applications and associated threats. By utilizing strategies like neural networks and ensemble learning, contemporary malware detection systems can effectively predict and categorize malicious activities while minimizing false positives. Even with these improvements, adversarial attacks present a significant hurdle for machine learning-based malware detection systems. These attacks take advantage of weaknesses in ML models by presenting obfuscated or adversarial samples, which can successfully evade detection. For instance, polymorphic and metamorphic malware can alter their code structure to escape static analysis methods while still retaining their harmful purpose [85], [86].

Additionally, the absence of standardized datasets and the fast-paced evolution of malware make it even more difficult to create strong ML models. Approaches such as adversarial training and robust feature extraction are being investigated to address these issues, but their application is still under development.

B. CLOUD AND EDGE COMPUTING IN MALWARE DETECTION

The advancement of cloud and edge computing has profoundly impacted the field of malware detection. Cloud computing offers scalable and centralized resources that facilitate the real-time analysis of extensive datasets, allowing for the identification of intricate malware patterns. By harnessing the computational capabilities of the cloud, sophisticated methodologies such as machine learning and big data analytics can be utilized to enhance both the accuracy and speed of detection. Nevertheless, cloud-based solutions encounter challenges, including latency issues, bandwidth constraints, and the necessity for stringent data privacy protocols.

In contrast, edge computing has surfaced as a complementary approach to address these challenges. It processes data in proximity to its origin, thereby minimizing latency and improving the responsiveness of malware detection systems. The decentralized nature of edge computing ensures that devices, particularly those within the Internet of Things (IoT), receive real-time protection. However, this approach also presents security challenges, including heightened vulnerability at the device level and the requirement for lightweight yet effective security measures.

The combination of cloud and edge computing is transforming mobile security by creating hybrid detection models that leverage the strengths of both approaches. For example, cloud systems take on heavy computational tasks such as training deep learning models, while edge devices focus on real-time threat detection and response. This hybrid strategy allows for effective malware detection while alleviating the resource limitations of edge devices. Moreover, the use of Artificial Intelligence (AI) and Machine Learning (ML) in these settings boosts the flexibility and accuracy of detection systems. AI-powered anomaly detection and encryption management at the edge offer strong protection against new threats. At the same time, cloud systems provide a broader view, spotting new malware variants and equipping edge devices with the latest threat intelligence [87], [88].

C. CROSS-LAYER ADVERSARIAL THREATS

Recent advancements in adversarial machine learning have reached the physical layer of wireless systems, uncovering weaknesses in deep learning-based modulation classifiers. For example, [56], [89] intra-class universal adversarial attacks can deceive modulation identification models with slight alterations. While this field focuses on communication signals, it also presents a wider threat model where adversarial examples could potentially evade upper-layer malware detection systems. Future studies might investigate these cross-layer adversarial threats and how AI-driven mobile malware detectors could be modified to resist them.

D. EVOLUTION OF MALWARE AS A SERVICE (MAAS)

Malware as a Service (MaaS) is an emerging trend in the world of cybercrime, making it easier for even those without technical expertise to carry out advanced malware operations. This model operates on a subscription basis, where cybercriminals provide malware tools and services for sale or lease on the dark web, complete with support, updates, and customization options. As a result, it has lowered the entry barriers for those looking to engage in cyberattacks. The rise of MaaS is fueled by the increasing professionalism within the cybercrime landscape. Criminal organizations are now offering user-friendly platforms, technical assistance, and even guarantees for their services, resembling legitimate software companies. This shift enables individuals with limited technical knowledge to execute various malware attacks, including ransomware, banking Trojans, and phishing schemes [90], [91].

MaaS presents considerable threats to the security of mobile devices, as it allows attackers to exploit the extensive and varied ecosystem of mobile applications and devices. The accessibility of MaaS reduces both the cost and effort involved in generating malware variants, rendering mobile platforms an attractive target for cybercriminals. Services specific to MaaS frequently take advantage of vulnerabilities found in mobile operating systems and applications, employing obfuscation and evasion strategies to circumvent conventional security protocols. In addition, MaaS accelerates the creation and dissemination of new threats, including advanced persistent threats (APTs) and zero-day exploits. This situation poses significant challenges to mobile security frameworks, which must adapt through ongoing updates to their detection and response strategies. The dependence on mobile devices for critical functions, such as banking and healthcare, further heightens the potential repercussions of attacks enabled by MaaS [92], [93].

Our study underscores various promising directions for future investigations focused on improving malware detection methodologies and tackling new challenges in the realm of cybersecurity. A central emphasis will be placed on the formulation of detection techniques that seamlessly combine static, dynamic, and hybrid approaches to effectively recognize intricate and zero-day malware threats. This endeavor encompasses the development of hybrid models that capitalize on the advantages of diverse detection strategies while integrating sophisticated techniques such as behavioral analytics and machine learning.

These initiatives are designed to enhance the detection of anomalies and reveal malicious activities that traditional detection systems might overlook. Furthermore, it is imperative to confront the challenges introduced by adversarial attacks that take advantage of weaknesses in detection frameworks, especially concerning polymorphic and metamorphic malware. Future investigations should prioritize the enhancement of these methodologies to attain improved accuracy and efficiency, thus facilitating a more resilient and adaptable response to the continuously evolving landscape of mobile malware.

In addition, promoting international collaboration for real-time threat intelligence sharing will be vital in strengthening the collective response to emerging threats. Ongoing surveillance of mobile applications and heightened user awareness regarding mobile security risks will also be crucial in prevention efforts. By amalgamating these components, our research aspires to contribute to a thorough understanding of mobile malware and to enhance detection and prevention strategies in light of the swiftly changing cyber threat environment.

XI. CONCLUSIONS

The threats posed by mobile malware have not only diversified but have also become far more serious, ranging from simple malware to ransomware, spyware, banking trojans, and others. It involves smishing, app repackaging, and zero-day exploits to capitalize on the weaknesses. As a result, some detection problems still remain despite advances in static and dynamic analysis, ML, and methods that combine elements of both categories. These are mainly the following: The first one covers such techniques as obfuscation and encryption; the second one is related to the adversarial attacks on the detection models; the third one worth mentioning is the ones caused by resource constraints of the mobile devices. However, the legal factors as well as ethical concerns add to the issues involved in research. To address these problems, organizational use of AI, behavioral analytics, and threat intelligence from across the globe is essential. Since there are possibilities for the utilization of resources that are cost-effective, the solutions must involve successful implementation of preventive security measures that do not violate ethical concerns. The complexity of mobile malware will continually rise in the future, and therefore it will be crucial to advance collaboration in innovation in order to protect users and organizations.

ACKNOWLEDGMENT

The authors extend their appreciation to the Deanship of Scientific Research, Vice Presidency for Graduate Studies and Scientific Research, King Faisal University, Saudi Arabia [GRANT No.]. The authors would like to thank the anonymous reviewers for their insightful scholastic comments and suggestions, which improved the quality and clarity of the paper.

ABBREVIATIONS

The following abbreviations are used in this manuscript:

AI	Artificial Intelligince
ANOVA	Analysis of Variance
APTs	Advanced Persistent Threats
DL	Deep Learning
DNN	Deep Neural Network
DT	Decision Trees
EDA	Exploratory Data Analysis
EHHO	Harris Hawks Optimizer
Grad-CAM	Gradient-weighted Class Activation Mapping
IoT	Internet of Things
KNN	K-Nearest Neighbors
ML	Machine Learning
MLP	Multi-Layer Perceptron
mRMR	Minimum Redundancy Maximum Relevance
NB	Naive Bayes
PCA	Principal Component Analysis
PCO	Particle Swarm Optimization
RF	Random Forest
RT	Random Tree
SLR	Systematic Literature Review
SVM	Support Vector Machine

REFERENCES

[1] K. Lab, "Flubot malware: Sms phishing and propagation on android devices," *Kaspersky Security Blog*, 2021. [Online]. Available: https://www.kaspersky.com/blog/flubot-malware

[2] G. S. Team, "Joker malware: A look into the mobile threat landscape," *Google Security Blog*, 2020. [Online]. Available: https://security.googleblog.com/2020/01/joker-malware.html

[3] H. Snyder, "Literature review as a research methodology: An overview and guidelines," *Journal of business research*, vol. 104, pp. 333–339, 2019.

[4] M. J. Page, J. E. McKenzie, P. M. Bossuyt, I. Boutron, T. C. Hoffmann, C. D. Mulrow, L. Shamseer, J. M. Tetzlaff, E. A. Akl, S. E. Brennan *et al.*, "The prisma 2020 statement: an updated guideline for reporting systematic reviews," *International journal of surgery*, vol. 88, p. 105906, 2021.

[5] K. Manuel, "Wannacry ransomware: A threat to global cybersecurity," *International Journal of Information Security*, vol. 16, no. 6, pp. 571–572, 2017.

[6] J. Scott-Railton, B. Marczak, B. Razzak, K. Mabna, and R. Mohd, "The pegasus spyware: The emergence of a global surveillance threat," *Citizen Lab Reports*, 2019. [Online]. Available: https://citizenlab.ca/2019/10/pegasus-spyware-global-surveillance-threat/

[7] N. Moran and D. O'Brien, "The rise of joker malware: Stealthy threats on android devices," *Google Threat Analysis Group*, 2020. [Online]. Available: https://blog.google/threat-analysis-group/joker-malware-threat/

[8] S. Thompson and P. Howard, "The evolution of dridex malware and its impact on financial services," *Cybersecurity and Finance Journal*, vol. 14, pp. 101–112, 2021.

[9] L. Martínez and J. Cruz, "Flubot malware: A rising threat to android users via sms phishing," *Cybersecurity Review*, vol. 19, pp. 45–56, 2022.

[10] Y. Song, D. Zhang, J. Wang, Y. Wang, Y. Wang, and P. Ding, "Application of deep learning in malware detection: a review," *Journal of Big Data*, vol. 12, no. 1, p. 99, 2025.

[11] Z. Fu, M. Liu, Y. Qin, J. Zhang, Y. Zou, Q. Yin, Q. Li, and H. Duan, "Encrypted malware traffic detection via graph-based network analysis," in *Proceedings of the 25th International Symposium on Research in Attacks, Intrusions and Defenses*, 2022, pp. 495–509.

[12] X. Liu, X. Du, Q. Lei, and K. Liu, "Multifamily classification of android malware with a fuzzy strategy to resist polymorphic familial variants," *IEEE Access*, vol. 8, pp. 156 900–156 914, 2020.

[13] E. Praditya, S. Maarif, Y. Ali, H. J. R. Saragih, R. Duarte, F. A. Suprapto, and R. Nugroho, "National cybersecurity policy analysis for effective decision-making in the age of artificial intelligence," *Journal of Human Security*, vol. 19, no. 2, pp. 91–106, 2023.

[14] L. Elluri, V. Mandalapu, P. Vyas, and N. Roy, "Recent advancements in machine learning for cybercrime prediction," *Journal of Computer Information Systems*, vol. 65, no. 2, pp. 249–263, 2025.

[15] S. Yan, J. Ren, W. Wang, L. Sun, W. Zhang, and Q. Yu, "A survey of adversarial attack and defense methods for malware classification in cyber security," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 1, pp. 467–496, 2022.

[16] Z. Muhammad, Z. Anwar, A. R. Javed, B. Saleem, S. Abbas, and T. R. Gadekallu, "Smartphone security and privacy: a survey on apts, sensor-based attacks, side-channel attacks, google play attacks, and defenses," *Technologies*, vol. 11, no. 3, p. 76, 2023.

[17] Q. Han, S. Mandujano, S. Porst, V. Subrahmanian, S. D. Tetali, and Y. Xiong, *The Android Malware Handbook: Detection and Analysis by Human and Machine*. No Starch Press, 2023.

[18] O. Hussien, U. Butt, and R. B. Sulaiman, "Critical analysis and countermeasures tactics, techniques and procedures (ttps) that targeting civilians: A case study on pegasus," *arXiv preprint arXiv:2310.00769*, 2023.

[19] E. Macedo, "Signature-based ids for encrypted c2 traffic detection," Master's thesis, Universidade do Porto (Portugal), 2022.

[20] B. Sousa, D. Dias, N. Antunes, J. Cámara, R. Wagner, B. Schmerl, D. Garlan, and P. Fidalgo, "Mondeo-tactics5g: Multistage botnet detection and tactics for 5g/6g networks," *Computers & Security*, vol. 140, p. 103768, 2024.

[21] W. Wang *et al.*, "Android malware detection using static analysis based on deep learning," in *2021 IEEE International Conference on Communications (ICC)*. IEEE, 2021, pp. 1–6.

[22] Amnesty International, "Forensic methodology report: How to catch pegasus," 2021. [Online]. Available: https://www.amnesty.org/en/latest/research/2021/07/forensic-methodology-report-how-to-catch-pegasus/

[23] W. Li *et al.*, "Understanding android malware families using static code analysis," *IEEE Access*, vol. 9, pp. 38 391–38 403, 2021.

[24] CERT-EU, "Flubot malware advisory," 2021, available at: https://www.cert.europa.eu/publications/flubot-malware.

[25] B. Marczak *et al.*, "The nso pegasus spyware: Government surveillance of smartphones," *Citizen Lab Report*, 2021. [Online]. Available: https://citizenlab.ca/2021/07/pegasus-project-nso/

[26] M. Alazab, M. Alazab, A. Shalaginov, A. Mesleh, and A. Awajan, "Intelligent mobile malware detection using permission requests and api calls," *Future Generation Computer Systems*, vol. 107, pp. 509–521, 2020.

[27] L. Gong, Z. Li, F. Qian, Z. Zhang, Q. A. Chen, Z. Qian, H. Lin, and Y. Liu, "Experiences of landing machine learning onto market-scale mobile malware detection," pp. 1–14, 2020.

[28] A. Yan, Z. Chen, H. Zhang, L. Peng, Q. Yan, M. U. Hassan, C. Zhao, and B. Yang, "Effective detection of mobile malware behavior based on explainable deep neural network," *Neurocomputing*, vol. 453, pp. 482–492, 2021.

[29] F. Taher, O. AlFandi, M. Al-kfairy, H. Al Hamadi, and S. Alrabaee, "Droiddetectmw: a hybrid intelligent model for android malware detection," *Applied Sciences*, vol. 13, no. 13, p. 7720, 2023.

[30] R. Casolare, C. De Dominicis, G. Iadarola, F. Martinelli, F. Mercaldo, A. Santone *et al.*, "Dynamic mobile malware detection through system call-based image representation." *J. Wirel. Mob. Networks Ubiquitous Comput. Dependable Appl.*, vol. 12, no. 1, pp. 44–63, 2021.

[31] C. Liu, J. Lu, W. Feng, E. Du, L. Di, and Z. Song, "Mobipcr: Efficient, accurate, and strict ml-based mobile malware detection," *Future Generation Computer Systems*, vol. 144, pp. 140–150, 7 2023.

[32] G. Iadarola, F. Martinelli, F. Mercaldo, and A. Santone, "Towards an interpretable deep learning model for mobile malware detection and family identification," *Computers and Security*, vol. 105, 6 2021.

[33] K. A. Kumar, A. Raman, C. Gupta, and R. R. Pillai, "The recent trends in malware evolution, detection and analysis for android devices," *Journal of Engineering Science and Technology Review*, vol. 13, pp. 240–248, 2020.

[34] G. D'Angelo, M. Ficco, and F. Palmieri, "Malware detection in mobile environments based on autoencoders and api-images," *Journal of Parallel and Distributed Computing*, vol. 137, pp. 26–33, 3 2020.

[35] M. M. Saudi, M. A. Husainiamer, A. Ahmad, and M. Y. I. Idris, "ios mobile malware analysis: a state-of-the-art," *Journal of Computer Virology and Hacking Techniques*, 2023.

[36] M. Ababneh, A. Al-Droos, and A. El-Hassan, "Modern mobile malware detection framework using machine learning and random forest algorithm." *Computer Systems Science & Engineering*, vol. 48, no. 5, 2024.

[37] S. Y. Yerima, "High accuracy detection of mobile malware using machine learning," *Electronics*, vol. 12, no. 6, p. 1408, 2023.

[38] J. Mohamad Arif, M. F. Ab Razak, S. Awang, S. R. Tuan Mat, N. S. N. Ismail, and A. Firdaus, "A static analysis approach for android permission-based malware detection systems," *PloS one*, vol. 16, no. 9, p. e0257968, 2021.
[39] A. Kaur, S. Lal, S. Goel, M. Pandey, and A. Agarwal, "Android malware detection system using machine learning," pp. 186–191, 2024.
[40] A. Lekssays, B. Falah, and S. Abufardeh, "A novel approach for android malware detection and classification using convolutional neural networks." pp. 606–614, 2020.
[41] A. S. Shatnawi, Q. Yassen, and A. Yateem, "An android malware detection approach based on static feature analysis using machine learning algorithms," *Procedia Computer Science*, vol. 201, pp. 653–658, 2022.
[42] E. Villarroel, "Dynamic malware analysis using machine learning-based detection algorithms," *Interfases*, no. 19, pp. 110–120, 2024.
[43] J. M. Arif, M. F. Ab Razak, S. R. T. Mat, S. Awang, N. S. N. Ismail, and A. Firdaus, "Android mobile malware detection using fuzzy ahp," *Journal of Information Security and Applications*, vol. 61, p. 102929, 2021.
[44] L. Chen, C. Xia, S. Lei, and T. Wang, "Detection, traceability, and propagation of mobile malware threats," *IEEE Access*, vol. 9, pp. 14 576–14 598, 2021.
[45] M. A. Haq and M. Khuthaylah, "Leveraging machine learning for android malware analysis: Insights from static and dynamic techniques," *Engineering, Technology & Applied Science Research*, vol. 14, no. 4, pp. 15 027–15 032, 2024.
[46] C. S. Yadav, J. Singh, A. Yadav, H. S. Pattanayak, R. Kumar, A. A. Khan, M. A. Haq, A. Alhussen, and S. Alharby, "Malware analysis in iot android systems with defensive mechanism," *Electronics (Switzerland)*, vol. 11, 8 2022.
[47] J. Jeon, J. H. Park, and Y. S. Jeong, "Dynamic analysis for iot malware detection with convolution neural network model," *IEEE Access*, vol. 8, pp. 96 899–96 911, 2020.
[48] R. Almohaini, I. Almomani, and A. Alkhayer, "Hybrid-based analysis impact on ransomware detection for android systems," *Applied Sciences (Switzerland)*, vol. 11, 11 2021.
[49] A. B. Sallow, M. A. M. Sadeeq, R. R. Zebari, M. B. Abdulrazzaq, A. B. Sallow, M. R. Mahmood, H. M. Shukur, and L. M. Haji, "An investigation for mobile malware behavioral and detection techniques based on android platform," vol. 22, pp. 14–20. [Online]. Available: www.iosrjournals.org
[50] M. Ashawa and S. Morris, "Analysis of mobile malware: A systematic review of evolution and infection strategies," *Journal of Information Security and Cybercrimes Research*, vol. 4, pp. 103–131, 12 2021.
[51] Y. K. Kim, J. J. Lee, M. H. Go, H. Y. Kang, and K. Lee, "A systematic overview of the machine learning methods for mobile malware detection," *Security and Communication Networks*, vol. 2022, 2022.
[52] S. A. Roseline and S. Geetha, "A comprehensive survey of tools and techniques mitigating computer and mobile malware attacks," *Computers and Electrical Engineering*, vol. 92, 6 2021.
[53] R. Surendran, T. Thomas, and S. Emmanuel, "A tan based hybrid model for android malware detection," *Journal of Information Security and Applications*, vol. 54, 10 2020.
[54] R. B. Hadiprakoso, H. Kabetta, and I. K. S. Buana, "Hybrid-based malware analysis for effective and efficiency android malware detection," pp. 8–12, 11 2020.
[55] C. Rodrigo, S. Pierre, R. Beaubrun, and F. E. Khoury, "Brainshield: A hybrid machine learning-based malware detection model for android devices," *Electronics (Switzerland)*, vol. 10, 12 2021.
[56] R. Li, H. Liao, J. An, C. Yuen, and L. Gan, "Intra-class universal adversarial attacks on deep learning-based modulation classifiers," *IEEE Communications Letters*, vol. 27, no. 5, pp. 1297–1301, 2023.
[57] R. Feng, S. Chen, X. Xie, G. Meng, S.-W. Lin, and Y. Liu, "A performance-sensitive malware detection system using deep learning on mobile devices," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 1563–1578, 2020.
[58] S. Piskunov, "Jadx - dex to java decompiler," 2013, available at: https://github.com/skylot/jadx.
[59] Brut.alll, "Apktool - a tool for reverse engineering android apks," 2012, available at: https://ibotpeaches.github.io/Apktool/.
[60] A. Desnos, "Androguard - reverse engineering, malware analysis, and more for android," 2013, available at: https://github.com/androguard/androguard.
[61] A. Desnos and G. Guadard, "Droidbox: An android application sandbox for dynamic analysis," in *Workshop on Cyber Security*, 2011.
[62] M. Alam, S. Vuong, M. Alazab, and A. Beheshti, "Cuckoodroid: Automated android malware analysis framework," in *Proceedings of the 16th International Conference on Information Integration and Web-based Applications Services*. ACM, 2014, pp. 421–426.
[63] G. Combs, "Wireshark network protocol analyzer," 2006, available at: https://www.wireshark.org/.
[64] A. Walters and N. Petroni, "The volatility framework: Volatile memory artifact extraction utility framework," in *Black Hat DC*, 2007.
[65] MobiSec, "Mobile security framework (mobsf)," 2016, available at: https://github.com/MobSF/Mobile-Security-Framework-MobSF.
[66] S. Barbhuiya, P. Kilpatrick, and D. S. Nikolopoulos, "Droidlight: Lightweight anomaly-based intrusion detection system for smartphone devices," in *Proceedings of the 21st international conference on distributed computing and networking*, 2020, pp. 1–10.
[67] H. Shahriar, C. Zhang, M. A. Talukder, and S. Islam, "Mobile application security using static and dynamic analysis," *Machine intelligence and big data analytics for cybersecurity applications*, pp. 443–459, 2021.
[68] R. A. Yunmar, S. S. Kusumawardani, F. Mohsen *et al.*, "Hybrid android malware detection: A review of heuristic-based approach," *IEEE Access*, vol. 12, pp. 41 255–41 286, 2024.
[69] I. U. Haq, T. A. Khan, and A. Akhunzada, "A dynamic robust dl-based model for android malware detection," *IEEE Access*, vol. 9, pp. 74 510–74 521, 2021.
[70] Ö. Aslan, M. Ozkan-Okay, and D. Gupta, "Intelligent behavior-based malware detection system on cloud computing environment," *IEEE Access*, vol. 9, pp. 83 252–83 271, 2021.
[71] A. Pathak, T. S. Kumar, and U. Barman, "Static analysis framework for permission-based dataset generation and android malware detection using machine learning," *EURASIP Journal on Information Security*, vol. 2024, no. 1, p. 33, 2024.
[72] A. Mohanraj and K. Sivasankari, "Android traffic malware analysis and detection using ensemble classifier," *Ain Shams Engineering Journal*, p. 103134, 2024.
[73] P. Bhooshan, N. Sonkar *et al.*, "Comprehensive android malware detection: Leveraging machine learning and sandboxing techniques through static and dynamic analysis," pp. 580–585, 2024.
[74] B. Hammi, J. Hachem, A. Rachini, and R. Khatoun, "Malware detection through windows system call analysis," p. 7, 2024.
[75] D. Spiekermann and J. Keller, "Challenges of digital investigations in nowadays communication networks," pp. 872–877, 2024.
[76] Y. Pan, X. Ge, C. Fang, and Y. Fan, "A systematic literature review of android malware detection using static analysis," *IEEE Access*, vol. 8, pp. 116 363–116 379, 2020.
[77] J. Senanayake, H. Kalutarage, and M. O. Al-Kadri, "Android mobile malware detection using machine learning: A systematic review," *Electronics (Switzerland)*, vol. 10, 7 2021.
[78] A. D. Lorenzo, F. Martinelli, E. Medvet, F. Mercaldo, and A. Santone, "Visualizing the outcome of dynamic analysis of android malware with vizmal," *Journal of Information Security and Applications*, vol. 50, 2 2020.
[79] A. A. Hamza, I. T. A. Halim, M. A. Sobh, and A. M. Bahaa-Eldin, "Hsas-md analyzer: A hybrid security analysis system using model-checking technique and deep learning for malware detection in iot apps," *Sensors*, vol. 22, 2 2022.
[80] C. Wang, T. Liu, Y. Zhao, L. Zhang, X. Du, L. Li, and H. Wang, "Towards demystifying android adware: Dataset and payload location," in *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering Workshops*, 2024, pp. 167–175.
[81] A. Guerra-Manzanares and H. Bahsi, "Experts still needed: boosting long-term android malware detection with active learning," *Journal of Computer Virology and Hacking Techniques*, pp. 1–18, 2024.
[82] S. Udeze, H. RAFIQ, D. Jeremiah, V.-T. TA, and M. USMAN, "Image-based android malware detection using deep learning," in *16th International Conference On Global Security, Safety & Sustainability, ICGS3-24: Cybersecurity and Human Capabilities through Symbiotic Artificial Intelligence*. Springer, 2024, pp. 1–13.
[83] F. A. Aboaoja, A. Zainal, F. A. Ghaleb, B. A. S. Al-rimy, T. A. E. Eisa, and A. A. H. Elnour, "Malware detection issues, challenges, and future directions: A survey," *Applied Sciences (Switzerland)*, vol. 12, 9 2022.
[84] U. Urooj, B. A. S. Al-Rimy, A. Zainal, F. A. Ghaleb, and M. A. Rassam, "Ransomware detection using the dynamic analysis and machine learning: A survey and research directions," *Applied Sciences (Switzerland)*, vol. 12, 1 2022.
[85] F. A. Aboaoja, A. Zainal, F. A. Ghaleb, B. A. S. Al-rimy, T. A. E. Eisa, and A. A. H. Elnour, "Malware detection issues, challenges, and future directions: A survey," *Applied Sciences (Switzerland)*, vol. 12, 9 2022.

[86] D. Gibert, C. Mateu, and J. Planes, "The rise of machine learning for detection and classification of malware: Research developments, trends and challenges," *Journal of Network and Computer Applications*, vol. 153, 3 2020.

[87] D. Alsadie, "Artificial intelligence techniques for securing fog computing environments: Trends, challenges, and future directions," *IEEE Access*, 2024.

[88] I. Gulatas, H. H. Kilinc, A. H. Zaim, and M. A. Aydin, "Malware threat on edge/fog computing environments from internet of things devices perspective," *IEEE Access*, vol. 11, pp. 33 584–33 606, 2023.

[89] F. Restuccia and T. Melodia, "Deepradioid: Real-time channel-resilient optimization of deep learning-based radio fingerprinting systems," *IEEE Transactions on Mobile Computing*, vol. 20, no. 3, pp. 1085–1098, 2020.

[90] J. H. Verhagen, R. A. Fouchier, and N. Lewis, "Highly pathogenic avian influenza viruses at the wild–domestic bird interface in europe: Future directions for research and surveillance," *Viruses*, vol. 13, 2 2021.

[91] S. Rao, A. K. Verma, and T. Bhatia, "A review on social spam detection: Challenges, open issues, and future directions," *Expert Systems with Applications*, vol. 186, 12 2021.

[92] L. Caviglione, M. Choras, I. Corona, A. Janicki, W. Mazurczyk, M. Pawlicki, and K. Wasielewska, "Tight arms race: Overview of current malware threats and trends in their detection," *IEEE Access*, vol. 9, pp. 5371–5396, 2021.

[93] A. R. de Araujo Zanella, E. da Silva, and L. C. P. Albini, "Security challenges to smart agriculture: Current state, key issues, and future directions," *Array*, vol. 8, p. 100048, 12 2020.

SEETAH ALMARRI Department of Computer Networks and Communications, College of Computer Sciences and Information Technology, King Faisal University, Al-Ahsa, Saudi Arabia Seetah Almarri received a B.S. degree in computer science and information technology from King Faisal University in 2019, where she is currently pursuing the M.S. degree in sciences in cybersecurity with the Department of Computer Networks and Communications. Her research interests include artificial intelligence, cybersecurity, software-defined networks, blockchain, the Internet of Things, and lightweight cryptography.

ALANOUD BUDOKHI Department of Computer Networks and Communications, College of Computer Sciences and Information Technology, King Faisal University, Al-Ahsa, Saudi Arabia Alanoud Budokhi received the B.S. degree in computer science from King Faisal University, Al-Ahsa, Saudi Arabia, in 2022, where she is currently pursuing the M.S. degree in sciences in cybersecurity with the Department of Computer Networks and Communications. Her research interests include artificial intelligence, cybersecurity, the Internet of Things, cloud computing, and blockchain.cryptography.

MOUNIR FRIKHA received the Diploma degree in engineering from the Technical University of Braunschweig, Brunswick, Germany, and the Ph.D. degree in telecommunications from the Technical University of Braunschweig, in 1999, and the Habilitation Universitaire (HD) degree from the Higher School of Communication of Tunis (SUP'COM), Tunis, Tunisia, in 2004.,He is currently a Full Professor with the Higher School of Communication of Tunis (SUP'COM), University of Carthage. He has conducted research on the management of the quality of service and mobility with the MEDIATRON Research Unit. He has contributed to numerous scientific events and has been the president and a member of numerous conferences and symposiums' program committees. His research areas are mobility in IP networks, wireless networks, AD HOC networks, and the Internet of things.

• • •