Review

A Survey on Machine Learning Techniques in Multi-platform Malware Detection: Securing PC, Mobile Devices, IoT, and Cloud Environments

Jannatul Ferdous * and Rafiqul Islam

Posted Date: 4 December 2024

doi: 10.20944/preprints202412.0348.v1

Keywords: Machine learning; Malware detection; Multi-platform malware; Malware analysis; P.C. malware; Mobile malware; IoT malware defense; Cloud-based malware detection

 doi:10.20944/preprints202412.0348.v1

Review

A Survey on Machine Learning Techniques in Multi-platform Malware Detection: Securing PC, Mobile Devices, IoT, and Cloud Environments

Jannatul Ferdous * and Rafiqul Islam

School of Computing, Mathematics and Engineering, Charles Sturt University, Australia
* Correspondence: jferdous@csu.edu.au

Abstract: Malware has emerged as a significant threat to end-users, businesses, and governments, resulting in financial losses of billions of dollars. Cybercriminals have found malware to be a lucrative business because of its evolving capabilities and ability to target diverse platforms such as PCs, mobile devices, IoT, and cloud platforms. While previous studies have explored single platform-based malware detection, no existing research has comprehensively reviewed malware detection across diverse platforms using machine learning (ML) tactics. With the rise of malware on PC/laptop devices, it is now targeting mobile devices and IoT systems, posing a significant threat to cloud environments. Therefore, a platform-based understanding of malware detection and defense mechanisms is essential for countering this evolving threat. To fill this gap and motivate further research, we present an extensive review of malware detection using ML techniques with respect to PCs, mobile devices, IoT, and cloud platforms. This paper begins with an overview of malware, including its definition, prominent types, impacts, analysis, and features. It presents a comprehensive review of machine learning-based malware detection from recent literature, including journal articles, conference proceedings, and online resources published since 2017. This survey also offers insights into current challenges and outlines future directions for developing adaptable cross-platform malware detection techniques. This survey is crucial for understanding the evolving threat landscape and developing robust detection strategies.

Keywords: Machine learning; Malware detection; Multi-platform malware; Malware analysis; P.C. malware; Mobile malware; IoT malware defense; Cloud-based malware detection

1. Introduction

In recent years, malware has evolved into one of the most pervasive cybersecurity threats, capable of targeting not only traditional systems such as PCs, but also mobile devices, IoT, and cloud platforms. Malware is becoming increasingly complex and varied by employing methods such as code obfuscation, encryption, polymorphism, and metamorphism to avoid detection [1]. The increasing sophistication of malware and its capability to bypass conventional security measures have caused significant financial, operational, and reputational damage to individuals, businesses, and governments. As technology becomes increasingly integrated across platforms, cybercriminals can simultaneously exploit multiple systems. Therefore, a thorough investigation of multi-platform malware detection is not only timely but crucial for ensuring cybersecurity resilience.

In the context of cybersecurity, malware refers to malicious software that is intentionally designed to disrupt, damage, or gain unauthorized access to computer systems. In contrast, multi-platform malware is malware capable of infecting and spreading across various types of platforms, often simultaneously. Malware can be categorized into several forms depending on its purpose and information-sharing system, such as ransomware, spyware, adware, rootkits, worms, horses, botnets, trojans, and viruses. Machine learning (ML), in this study, refers to computational techniques that allow systems to learn from data and improve their performance over time without being explicitly programmed. The application of ML for malware detection has shown significant potential for

 doi:10.20944/preprints202412.0348.v1

automating threat identification and reducing detection latency, especially in environments with high complexity and variability.

The increasing prevalence of malware is evidenced by the increasing number of global cyberattacks. According to the 2024 Cisco Cybersecurity Readiness Index [2], 76% of firms experience malware attacks, as shown in Figure 1. Astra's Malware Statistics 2024 reports that 560,000 new malware pieces are detected daily, adding to over 1 billion existing programs. This large volume of malware thwarts organizational security, often resulting in ransomware attacks [3]. The scale and impact of ransomware attacks are expected to increase significantly in the future. Cybersecurity Ventures predict that victims could pay approximately $265 billion annually by 2031, with costs increasing by 30% each year [4]. Malware targeting Linux systems has also increased, with a 35% increase in infections and the emergence of new malware families impacting Linux-based platforms [5].

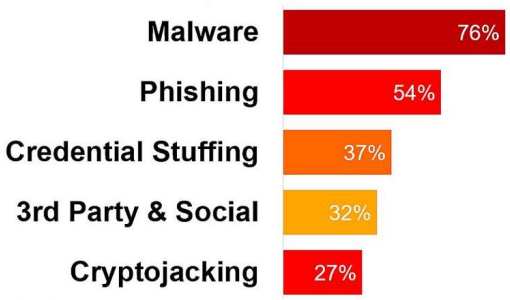


Figure 1. Types of Attacks Experienced by Companies (Published September 2024 by CISCO) [2].

Furthermore, 2023 marked a pivotal moment for IoT security threats. A report from Zscaler ThreatLabz in October 2023 showed a 400% increase in IoT malware attacks compared to the previous year [6]. Overall, the global proliferation of mobile devices, IoT systems, and cloud computing has expanded the attack surface, providing cybercriminals with new vectors for deploying malware. Hence, new challenges have arisen for malware detection. Traditional malware detection methods tailored to specific platforms, such as PCs or mobile devices, are insufficient to counter these new threats. This underscores the necessity of adopting a unified, multi-platform approach to malware detection that can provide holistic defense strategies.

In response to this evolving threat landscape, numerous studies have been conducted with an increasing focus on machine learning (ML), owing to its ability to handle the complexity of modern threats. Traditional approaches to malware detection, such as signature-based and heuristic methods, have proven inadequate for combating sophisticated and polymorphic malware, particularly in dynamic, multi-platform environments. Thus, the development of more advanced detection techniques, such as behavior-based and machine learning (ML)-driven approaches, has become essential in modern cybersecurity defenses

Despite the rising threat of multi-platform malware, existing research on malware detection remains predominantly focused on single platforms, either PCs or mobile devices, with relatively few studies addressing IoT or cloud environments. Moreover, these studies often fail to account for the growing interconnectivity between platforms, which allows malware to migrate easily from one system to another. This creates a significant gap in the literature, as there is no comprehensive review of machine learning techniques that address malware detection across PCs, mobile devices, IoT, and cloud environments.

Multi-platform malware detection is, therefore, critical for several reasons. Cyberattacks today often exploit the weakest link across interconnected systems. For instance, a single vulnerability in an IoT device can be leveraged to infiltrate broader networks, including enterprise cloud systems. Second, malware has evolved to operate across multiple platforms, with many modern malware variants designed to be adaptable to different operating environments. Mirai, a botnet initially designed to target IoT devices, was later modified to attack cloud-based systems and enterprise

 doi:10.20944/preprints202412.0348.v1

networks. Hence, developing unified defense strategies is essential, as organizations adopt hybrid environments that combine on-premises and cloud-based systems.

This survey aims to fill this gap by providing a holistic review of the recent literature on malware detection using ML methods across diverse platforms. This paper reviews the state-of-the-art ML techniques used to detect malware on PCs, mobile devices, IoT systems, and cloud environments. It also outlines the specific challenges encountered on each platform and provides insights for adapting techniques for cross-platform usage. In doing so, this survey not only serves as a valuable resource for researchers and practitioners in cybersecurity, but also offers a foundation for future research into adaptable, cross-platform malware detection strategies using machine learning.

The key contributions of this study are as follows.

- To the best of our knowledge, this is the first comprehensive review of malware detection in PCs, mobile devices, IoT systems, and cloud environments using machine-learning techniques.
- This study details the various types of features (e.g., static, dynamic, memory, and hybrid) used to train the ML models. It also discusses the malware landscape across platforms and identifies both platform-specific challenges and cross-platform issues that affect the development of effective ML-based malware detection techniques.
- This study examines existing malware detection techniques using various ML and DL models and provides the overall research trends observed for each platform.
- This study highlights gaps in the existing research and proposes future directions, such as developing adaptable, scalable, and efficient ML algorithms for multiple platforms and promoting unified cross-platform malware detection approaches.

The structure of this survey is organized as follows: Section 2 provides a comparison with previous related work. Section 3 provides an overview of malware, including malware definitions, leading malware threats, malware analysis, and features used to build the ML model for malware detection. Section 4 describes the malware landscape across diverse platforms. Section 5 presents an overview of machine learning algorithms for malware detection. Section 6 provides an extensive review of malware detection using ML techniques with respect to PCs, mobile devices, the IoT, and cloud platforms. Section 7 presents the challenges associated with platform and cross-platforms. Section 8 presents the limitations of the existing literature and future research directions. Finally, Section 9 concludes the paper.

2. Comparison with Previous Related Surveys

This section examines survey papers on malware detection via machine learning from 2017 onwards, highlighting the gaps that we intend to address. This will help researchers to establish a baseline for developing countermeasures. Table 1 compares our survey with existing surveys.

Existing surveys on malware detection using machine learning and deep learning typically focus on specific platforms such as Windows[7–9,11] or Android [10,12,13]. A small number of studies [17,18] have examined both Windows and Android. Some surveys [14–16] have addressed malware classification in IoT platforms using ML and DL techniques. However, many current studies lack a comprehensive understanding of the IoT malware. Few studies have focused on cloud malware. Belal and Sundaram [19] provided a taxonomy of ML/DL-based cloud security, addressing issues, challenges, and trends, whereas Aslan et al. [20] discussed behavior-based malware detection in the cloud. Table 1 also reveals that several surveys have focused exclusively on DL technologies for malware detection, such as those in [9,17,21,22], without focusing on traditional ML or ensemble learning techniques. However, traditional ML and ensemble learning offer distinct advantages including lower computational requirements, faster training times, and better performance on smaller datasets.

 doi:10.20944/preprints202412.0348.v1

Table 1. Summary of existing review papers for comparison with our study (myth: √ indicates complete information provided, ≈ indicates partial information provided, and × indicates no information provided).

Papers	Year	Main contribution	Insights into malware				ML-based malware detection in diverse platform					Challenges identified
			Latest prominent malware variants	Platform-based malware taxonomy	Analysis methods (Static, dynamic, memory and hybrid)	Feature details	Pcs		Mobile	IoT	Cloud	
							Windows	Linux				
[7]	2021	Survey on malware detection techniques using machine learning algorithms.	×	×	√	×	√	×	×	×	×	×
[8]	2019	Survey on sophisticated attack and evasion techniques used by the contemporary malwares.	×	×	≈	×	√	×	×	×	×	×
[9]	2022	This survey is on the use of Deep Learning-based malware detection.	×	×	≈	×	√	×	×	×	×	×
[10]	2021	Reviewed machine learning methods for Android malware detection.	×	×	√	×	×	×	√	×	×	×
[11]	2020	Study on traditional and state-of-the- art ML techniques for malware detection	×	×	≈	√	√	×	×	×	×	×
[12]	2023	DL approaches for malware defenses in the Android environment	×	×	√	√	×	×	√	×	×	×

 doi:10.20944/preprints202412.0348.v1

[13]	2020	Android malware detection using deep learning	×	×	≈	√	×	×	√	×	×	×
[14]	2023	Survey on IoT malware taxonomy and detection mechanisms.	×	≈	×	×	×	×	×	√	×	√
[15]	2023	Discussed IoT dataset use to evaluate the machine learning techniques.	×	×	≈	×	×	×	×	√	×	×
[16]	2023	Review on emerging machine learning algorithms for detecting malware in IoT.	×	×	×	×	×	×	×	√	×	×
[17]	2024	Modern deep learning technologies for identifying malware on Windows, Linux, and Android platforms.	×	×	√	≈	√	√	√	×	×	×
[18]	2021	Computer-based and mobile-based malware detection and, their countermeasures are presented.	×	×	≈	×	√	×	√	×	×	√
[19]	2022	ML and DL based defenses against attacks and security issues in cloud computing is provided.	×	×	×	×	×	×	×	×	√	√
[20]	2021	Behavior-based malware detection system in the cloud environment	×	×	≈	×	×	×	×	×	×	×
Our survey	**2024**	**Survey on malware detection in PC, mobile, IoT and cloud platform using ML techniques.**	**√**	**√**	**√**	**√**	**√**	**√**	**√**	**√**	**√**	**√**

 doi:10.20944/preprints202412.0348.v1

These benefits highlight the importance of exploring these techniques along with DL for comprehensive malware detection strategies. Moreover, existing surveys fail to comprehensively address malware detection across platforms such as Linux, macOS, iOS, IoT, and the cloud, which are also frequently targeted by malware. The lack of platform diversity in current surveys highlights the need for an inclusive review that covers various environments to thoroughly understand malware-detection methods. To fill these gaps, this study provides a comprehensive survey of recent ML and DL approaches for malware detection across Windows, Linux, macOS, Android, IoT, and cloud platforms, which are frequently targeted by malware.

3. Malware Fundamentals

This section explores the fundamental aspects of malware, including its definition, type, and disruptive impact on systems and data. It also highlights recent and significant malware threats, discusses standard analysis techniques, and examines the critical features that enable machine learning to detect and combat these threats.

3.1. What is Malware

Malware refers to malicious software designed to compromise computer systems or to gain unauthorized access. Despite advancements in cybersecurity, malware remains a significant threat, disrupting systems by stealing information, rendering services unavailable, or damaging files. Common malware categories include viruses, worms, trojans, backdoors, spyware, adware, botnets, rootkits and ransomware. Each exhibits distinct behaviors: viruses modify or delete files, worms self-replicate across networks, rootkits allow remote control, and Trojans masquerade as legitimate applications for covert activities. Adware displays unwanted ads, spyware tracks user activities, botnets exploit resources, backdoors bypass security for unauthorized access, and ransomware encrypts data, demanding payments for decryption [23]. This classification highlights the diverse operational goals of the malware.

3.2. Leading Malware Threats in the Current Cyber Landscape

The cyberthreat landscape is dominated by sophisticated malware that targets multiple platforms. Malware increasingly employs evasive, polymorphic, and adaptive tactics to evade traditional security measures, thereby posing detection and mitigation challenges. Cybercriminals also leverage AI-powered malware, further complicating defense.

In this section, we examine prevalent malware threats, their characteristics, attack methods, and associated damages, underscoring the need for cybersecurity professionals to remain informed and proactive against these evolving threats.

Ransomware: Ransomware continues to be one of the most widespread and damaging forms of malware. The COVID-19 pandemic led to an increase in ransomware activities, which have further escalated in 2023. Ransomware attacks have shifted from targeting large enterprises with complex methods to widespread attacks on small businesses facilitated by Ransomware-as-a-Service kits. Currently, LockBit is the most prevalent ransomware toolset. In February 2024, an international law enforcement operation seized 34 LockBit servers; however, LockBit3.0 quickly emerged just five days later [24].

Ransomware attacks target a wide range of computing devices, including desktops, mobiles, IoT, and cloud environments. Cybercriminals employ various attack vectors such as phishing spams and exploit vulnerabilities to deliver malicious files [25]. The ransomware then encrypts the critical files and collects information regarding the target. They frequently connect to remote servers to obtain additional components or transfer files. Victims receive recovery instructions, often through ransom notes or desktop changes, in exchange for payments.

Recent high-profile ransomware attacks by Conti, REvil, Darkside, and LockBit 3.0 have significantly impacted global infrastructure, healthcare, and businesses. For instance, Conti's attack on Costa Rica's government led to a national state of emergency [26], whereas REvil's Kaseya breach

 doi:10.20944/preprints202412.0348.v1

demanded a $70 million ransom [27]. Darkside is known for stealthy compromises such as the Colonial Pipeline incident, costing $5 million [28]. LockBit 3.0 has also carried out significant attacks, such as the Accenture breach, demanding a $50 million ransom [29].

Advanced Persistent Threats (APTs): Advanced Persistent Threats (APTs) have become a growing concern in recent malware trends, and are projected to reach a $12.5 billion market by 2025 [30]. APTs are characterized by their sophistication, advanced tactics, and prolonged, targeted campaigns against digital infrastructure for sabotage or espionage. They differ from conventional cyber threats in that they are more persistent and complex than other malware, such as ransomware.

APTs leverage obfuscation, anti-analysis tactics, and AI to evade detection and create zero-day exploits [31]. It operates through a multistage process to infiltrate and persist within a network. They start with reconnaissance, gathering information via open-source intelligence (OSINT) and social engineering, followed by initial access by spear phishing or system exploits. Attackers establish control, escalate privileges, move laterally, and exfiltrate data while maintaining a stealth to evade detection.

Prominent APT attacks include Stuxnet, which disrupts Natanz's centrifuges through zero-day vulnerabilities and fake software signatures [32]. The SolarWinds attack is another example of APT that deploys malware via a supply chain compromise in the Orion system [31].

Cryptojacking: Cryptojacking is a stealthy cyberattack in which malware is typically injected via malicious links into a network of devices and runs covertly in the background to harness the victim's computing resources for mining cryptocurrency. In 2023, cryptojacking incidents skyrocketed, exceeding the previous year's total by early April and reaching $1.06 billion by the year-end–a 659% increase [33]. Unlike ransomware, cryptojacking avoids direct payment demands and uses obfuscation to avoid detection. Figure 2 [34] illustrates how this process works step-by-step.

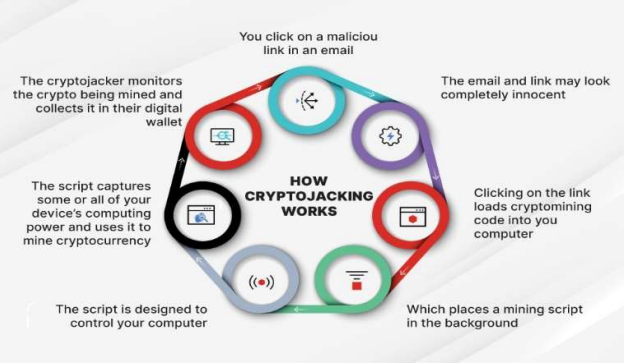


Figure 2. Step-by-step process of cryptojacking [34].

Cryptojacking targets various platforms, including desktops, servers, mobile devices, and cloud services, using different forms of malware or scripts.

Browser-based cryptojacking: This form of cryptojacking uses malicious JavaScript on websites to exploit user devices for cryptocurrency mining. It requires no software installation but may cause increased CPU usage, slowing down, or overheating devices.

Host-based cryptojacking: In host-based cryptojacking, attackers misuse the CPU or GPU of a system to mine cryptocurrencies. Unlike browser-based methods, this approach involves direct installation of malicious scripts on a host, often through phishing or bundled software. These scripts exploit the system's resources to convert cryptocurrencies.

Cloud cryptojacking: Cloud cryptojacking involves exploiting server and container vulnerabilities to mine cryptocurrency, impacting providers and customers through financial losses and reduced performance.

Notable cryptojacking incidents include the hacking of a European water utility, Tesla's cloud breach, and the cryptojacking code hidden on the Los Angeles Times website in 2018 [33,34]. Moreover, in 2020, the U.S. Department of Defense found cryptojacking malware on its servers [35], and in 2019, a Russian nuclear facility employee was fined $7000 for mining Bitcoin illegally [36].

 doi:10.20944/preprints202412.0348.v1

Spyware: Spyware enables cybercriminals to infiltrate networks by stealing sensitive data such as login credentials, screenshots, and chat histories. Pegasus, a well-known spyware variant, steals data from mobile devices and leverages BYOD policies to infiltrate secure networks. It provides cybercriminals with insider access, enabling them to locate and compromise valuable assets such as emails, SMS messages, app data, and multimedia. Its ability to bypass multi-factor authentication by extracting one-time passwords makes it even more dangerous [24].

Wiper malware: Wipers are malicious programs that permanently destroy user data and target both public and private computer networks. Threat actors use wipers to conceal their intrusion and to hinder the victim's response. Nation-state attackers deploy them to disrupt supply chains and military operations, while "hacktivists" use them to impede business activities in response to perceived injustices [37].

Recent examples include WhisperGate malware that targeted Ukraine in January 2022 [38] and HermeticWiper, which impacted various Ukrainian organizations in February 2022 [37].

Remote Access Trojans (RATs): RATs, a specific trojan type, are popular with cybercriminals for remotely controlling the endpoint devices. They trick users to run malicious codes by masking them as legitimate applications. Ghost, a Remote Access Trojan, controls the infected endpoints. Unlike typical malware, Ghost is manually deployed, suggesting that victims are already compromised by other malware [24].

Understanding the current landscape of malware threats is crucial for developing robust countermeasures and improving the detection capabilities.

3.4. Malware Analysis

In this subsection, we discuss various malware analysis methods that are crucial for the development of malware detection systems. The main goal of malware analysis is to identify the characteristics and purposes of suspicious files. Significant approaches for conducting malware analysis across platforms like Windows, Linux, and Android include [39]-

- Static analysis
- Dynamic analysis
- Memory analysis and
- Hybrid analysis

Static analysis techniques extract static signatures, features, or patterns from binary files without execution. This method is fast, secure, and efficient in identifying known malware samples, and does not require kernel privileges or a virtual machine. However, static analysis has significant limitations: it cannot examine malware strains using obfuscation techniques and is ineffective against malware that uses packers to compress and encrypt payloads [40].

Conversely, dynamic analysis involves executing malware in a controlled environment to observe runtime behavior. This enhances the understanding of malware functionality and enables the identification of previously unknown or zero-day malware. However, this approach is often slower and more time-consuming [40]. Additionally, dynamic analysis also has limitations in tracking highly sophisticated malware, such as fileless (memory-resident) malware.

Consequently, Memory analysis offers an alternative method for detecting malicious behaviors of fileless malware by capturing and examining volatile memory images during execution. While encryption and packing can conceal suspicious files, all processes are visible in the memory during runtime. Malware must disclose critical information (e.g., logs, code, and data segments) for operational functionality, making detection possible. Volatile memory analysis detects malware by examining its presence in the system's RAM, identifying fileless malware that evades detection by not leaving traces on hard drives.[41].

The hybrid malware analysis methodology combines multiple analysis approaches, offering greater effectiveness than a single analysis technique.

3.5. Features Used in ML-Based Malware Detection

 doi:10.20944/preprints202412.0348.v1

This subsection provides an overview of the features extracted from various platforms, each of which uses distinct file formats and yields different features. In Windows, malware features are extracted from executable (EXE) files, whereas Linux malware is analyzed using Executable and Linkable Format (ELF) files. In macOS, the Mach-O file format is used to analyse and extract the features. Android relies on APKs, and iOS utilizes IPA files. The APK file enables the extraction of static features from classes.dex files and dynamic features from the AndroidManifest.xml file [42]. IoT platforms derive features from firmware binaries, whereas cloud environments use container images and VM disk files, such as Docker and VMDK, for feature extraction. Table 2 classifies platform-specific features into static, dynamic, and memory-based features suited to different file formats and operating environments.

Table 2. Categorization of Platform-Specific Features: Static, Dynamic, and Memory-Based Approaches Across File Formats and Operating Environments.

	File Format	Static Features	Dynamic Features	Memory Features
Windows	Executable (EXE) files	**PE headers information:** Import/export address tables, section headers, entry point address, date timestamp, code section size**.** **File metadata**: Size, creation/modification dates, access permissions. **Strings:** IP addresses, domain names. **Opcode sequences**: An opcode is an instruction executed by a CPU, describing an executable file's behavior. Hence, opcode sequences are the specific sequences of operations extracted from the binary code.	**API Calls**: Sequence and types of Windows API calls (e.g., CreateProcess, WriteFile) **Registry modifications:** Registry key creation, deletion, or modification. **File system modifications:** Deletes, create, or overwrites the existing file, encrypts all or a subset of files in case of ransomware. **Host logs:** Events extracted from host logs. **Network activity:** Source and destination IP addresses, TCP ports, Domain Names System (DNS) requests, and network protocols (e.g., HTTP, HTTPS, SMTP etc.) **Resource usage:** Higher CPU or memory usage may indicate the presence of malware in the system.	Windows memory dumps
Linux	Executable and	**ELF header information:** Malware	**System-call patterns**: Frequency and type of	**Sections and Segments**:

 doi:10.20944/preprints202412.0348.v1

	Linkable Format (ELF): code, data, and metadata for execution.	developers manipulate ELF headers to evade or crash standard analysis tools [43]. **Internal Libraries:** Most Linux malware is statically linked to its libraries, eliminating external dependencies [43]. **Shared libraries**: List of dynamically loaded libraries. **Sections and segments**: Information on the .text (code) and .data (global variables) segments.	system calls. **Network behavior**: Monitoring outgoing/incoming connections and socket creation.	Memory segments (.text, .data, .bss).
macOS	**Mach-O files:** native executable format for macOS.	**Code signatures**: Presence and structure of code signing. **Dynamic libraries**: Information on loaded libraries (DYLIBs).	**File activity:** Monitor file creations, deletions, modifications, and access patterns. **Inbound and outbound traffic:** observe and analyze all network traffic, including DNS, HTTP requests, and other communication protocols. **Service start/stop:** Track each modification linked to service operation. **TI Reputation services:** Utilize threat intelligence feeds to detect malicious files, IP addresses, and domains.	**Sandboxing**: Memory protection through entitlements.
Android	**APK (Android Package Kit) files:** -It is a compressed archive that	**Strings:** Domain names, IP addresses, and ransom notes in case of ransomware attack **Permissions analysis**: The set of permissions requested by the app to	**Behavioural features**: Network communication, SMS, data storage behavior. **File system features:** Similar to PCs, features extracted from a mobile	**Embedded files**: Presence of assets (e.g., shared libraries) impacting

 doi:10.20944/preprints202412.0348.v1

	includes all resources needed to distribute and install applications on Android devices.	the users (e.g., camera access, network communication, Bluetooth, contacts, and more). **Manifest information**: Details about application components (e.g., activities, services, and receivers), **Intents:** Allows communication between various components of an app. **API calls:** API calls enable inter-application communication and monitoring them can detect malicious behavior.	device's file system can indicate the presence of malware. **User interaction:** Detecting ransomware can be achieved by correlating user interactions with application runtime events [44]. **System resource analysis:** CPU, memory and baterry, process reports and network usage. **Network traffic analysis:** URLs, IPs, Network Protocols, Certificates, Non-encrypted data	memory allocation. **Memory dumps:** A snapshot of Android's memory that captures all data and processes in the RAM at a specific time, including system processes, application data, and temporary data from various programs
iOS	iOS App Store Package (IPA): specific to iOS for app distribution.	**Code signing:** Verification of signatures. **Sandboxing and entitlements:** **Permission restrictions.**	**Objective-C method calls**: Runtime behavior. **Dynamic behavior**: API usage patterns (e.g., contacts, location access). **Data encryption**: Encrypted data usage.	**Entitlements**: Defines memory boundaries through sandboxing.
IoT	Various formats (e.g., BIN, HEX, Linux executables).	**Firmware version**: Metadata, updates, and patches. **Opcode sequences:** extracting operational codes after disassembling the binary file. **Control flow graph (CFG):** extracting from the assembly file **API calls**: extracting from the binary	**Network traffic**: Service type (http, smtp, ftp etc.), Device communication protocols (e.g., MQTT, CoAP), Packet size transmitted by Source IP address, etc. **Device-specific behavior**: Interactions with sensors, actuators, device ports. **System-calls:** Timestamp, return value,	**System-call sequences**: System-level commands specific to device memory. **Memory mapped IO**: Monitoring interactions with memory mapped I/O (MMIO).

 doi:10.20944/preprints202412.0348.v1

			arguments, and name of each System-calls. **CPU usage, Process usage, Ram usage and**.	**Memory buffer usage**: Analysis of memory buffers for potential overflows.
Cloud	VM disk images (e.g., VMDK, QCOW2), container formats (e.g., Docker im ages).	**VM metadata**: Hypervisor information (e.g., VM details). **Data storage patterns**: Interactions with cloud storage. **Strings and n-grams**	**API usage patterns**: Cloud-specific API calls (AWS SDK, Google Cloud API). **Container activity**: Monitoring processes, network activity in containers. **System calls:** Extracted from the interactions between applications and the OS's kernel during runtime.	**Virtual memory dumps:** Contains memory-specific features (system calls, memory access).

Static features derived from binaries or metadata without execution include file headers, opcode sequences, and metadata, which are essential for assessing executables and packages on Windows, Linux, macOS, Android, and iOS. Dynamic features capture behavior during execution, including system calls, API invocations, network activities, and registry or file system changes, aiding in the identification of complex or evasive malware. Memory features, such as memory allocation patterns and mapping, are vital for detecting sophisticated threats, particularly in IoT and cloud environments. This structured feature analysis underpins the implementation of machine learning models attached to each platform's unique characteristics.

4. Malware Landscape Across Platforms

The proliferation of digital technologies has expanded the malware threat landscape across various platforms including PCs, mobile devices, IoT, and cloud systems. Understanding the targeted operating system or device is crucial to comprehending malware behavior, as malicious software is often crafted for specific platforms that exploit system-specific vulnerabilities. In this study, the terms "platform" and "operating system" will be utilized synonymously, and we classify the target platforms for malware into four primary categories: PCs, mobile devices, IoT and cloud systems. Each platform has unique vulnerabilities, attack vectors, and security issues that require distinct detection and mitigation strategies. This section provides an overview of the malware landscape across these platforms, as shown in Figure 3.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 4 December 2024 doi:10.20944/preprints202412.0348.v1

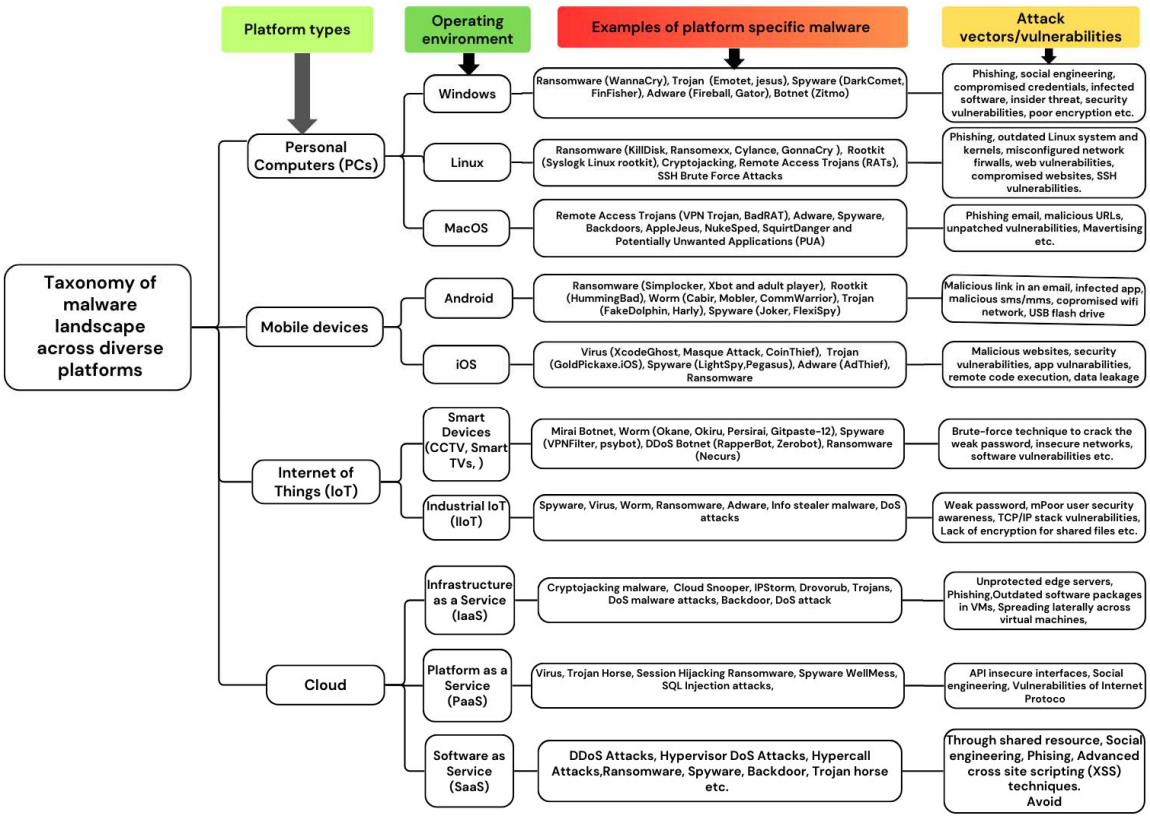


Figure 3. Taxonomy of malware landscape across various platforms.

4.1. PCs

The PC platform is a primary target for malware, facing various types that exploit specific vulnerabilities in Windows, macOS, and Linux environments. This study examines the malware landscape for each operating system, emphasizing common threats, typical attack vectors, and mitigation mechanisms.

4.1.1. Windows

The Windows platform remains a primary target for malware owing to its extensive use in personal and enterprise settings. Malware types include viruses, worms, trojans, ransomware, spyware, adware, and rootkits, each threatening system integrity and data security. Cybercriminals exploit phishing emails, malicious websites, software vulnerabilities, and removable media to initiate infection. Advanced techniques such as polymorphism, obfuscation, and encryption are used to avoid traditional detection, necessitating adaptive and robust detection mechanisms [8]. The platform's broad software ecosystem provides numerous entry points for attack. Although Microsoft employs security measures, such as Windows Defender and regular updates, their effectiveness depends on users practicing safe computing and maintaining updated systems. The impact of malware on Windows can lead to system performance issues, data theft, system crashes, and financial losses, thereby highlighting the significant consequences of these attacks.

4.1.2. Linux

Linux has become the leading operating system in multi-cloud environments, powering 78% of the world's top websites. This widespread use has increased the scale and complexity of linux-based systems [45]. The Linux OS supports various distributions for diverse hardware requirements, making it integral to many Internet-based desktop devices and a target for cybercriminals. The rise

 doi:10.20944/preprints202412.0348.v1

in Linux-based malware attacks is mainly due to the prevalence of IoT devices running Linux-based firmware, such as smart home assistants, security cameras, and industrial control systems. These devices often lack robust security mechanisms, making them susceptible to attacks that can compromise the ecosystem. In addition, as more companies adopt Linux-based servers and networks, hackers increasingly target these systems for potentially greater rewards. Trend Micro's research shows that 90% of public cloud workloads run on Linux, further motivating hackers to develop Linux malware [46]. Recently, Linux-based systems have become increasingly targeted for various malware attacks. According to the VMware threat report [45], these devices face an increase in cryptojacking malware, remote access trojans (RATs), SSH brute force attacks, web shell malware, and ransomware. The Trend Micro Linux Threat Landscape Report indicated a 62% increase in ransomware attacks on Linux from 2022 to 2023. The report identified that KillDisk Ransomware, among others, specifically targeted financial institutions, exploiting phishing attacks and outdated Linux systems and kernels. This report also states that Webshell exploits are the most common Linux malware at 49.6%, followed by Trojans at 29.4%, whereas backdoors and crypto miners are less prevalent [46]. Cybercriminals primarily exploit web vulnerabilities such as SQL injection, XSS, and SSRF to compromise web resources. They also targeted cloned websites, misconfigured firewalls, and SSH vulnerabilities to execute malware attacks on Linux systems.

4.1.3. macOS

The evolving macOS threat landscape necessitates greater vigilance from users and developers. Despite macOS's reputation for robust security, it remains vulnerable to cyber threats in 2022; malware detection on macOS rose significantly by 165%, accounting for 6.2% of the total increase from the previous year [47].

MacOS employs security features such as XProtect and Gatekeeper; however, they have limitations. XProtect's signature-based detection is ineffective against unknown or modified malware and lacks the dynamic scanning capabilities of third-party EDR tools. Gatekeeper is another security feature that blocks unsigned or malicious Internet applications, verifies developer IDs, and checks for alterations after signing. However, attackers can bypass this by using stolen developer IDs or exploiting legitimate apps to run malicious code. Additionally, while Sandboxing applications limit access to vital system resources, attackers have devised techniques to escape and obtain illicit access [47].

The most prevalent malware on macOS include adware, potentially unwanted programs (PUPs), backdoor spyware, remote access Trojans, stealers, ransomware, and other emerging malware types [47,48]. Over the years, new malware threats have emerged, including AppleJeus, which shifted tactics from Windows to macOS in 2018, and NukeSped, which functions as ransomware, spyware, and stealer and was detected in 2019. SquirtDanger, a macOS-targeting malware with advanced evasion techniques, was discovered in 2022 [47]. Common attack vectors include malvertising, phishing emails, malicious URLs, and unpatched vulnerabilities with persistent macOS vulnerabilities.

4.2. Mobile Devices

The increasing prevalence of mobile devices in modern society has made them prime targets for malware, particularly for smartphones. Malware developers primarily target Android and iOS operating systems, which dominate the global mobile OS market.

4.2.1. Android

The widespread adoption of Android platforms on smartphones, tablets, and IoT devices has increased its vulnerability to malicious cyber-attacks. The flexibility, cost-effectiveness, and computing power of Android devices have increased their popularity. They offer user-friendly third-party applications, such as games, fitness, monitoring, and healthcare, accessible globally via on-demand internet connections. However, the widespread popularity of Android has made it

 doi:10.20944/preprints202412.0348.v1

susceptible to cyber-attack. A recent report revealed that over 438,000 mobile malware installation packages were detected in the third quarter of 2023, marking a 19% increase from the second quarter [49]. Another report revealed that in Q2 2024, Android led the global mobile market with a 71.65% share, while iOS accounted for approximately 27.62% [50]. Android platforms face a range of malware threats, including credential theft, privacy breaches, bank fraud, ransomware, adware, and SMS fraud. Therefore, the development of automatic Android malware detection methods is vital for protecting the system security and user privacy.

Android is an open-source Linux-based mobile OS that allows anyone to access and use its own code. Its architectural framework comprises several distinct layers: kernel, hardware abstraction layer, Android runtime, libraries, application framework, and applications. These components collectively serve to optimize the system efficiency and application performance. Android offers security mechanisms, such as sandboxing, permissions, and encryption to protect data and ensure app integrity [51]. Android apps operate in isolated sandboxes with user-approved permissions for resources like cameras and Wi-Fi. Therefore, users should exercise caution when granting permissions because malicious apps can access sensitive resources once allowed.

Various forms of malware, such as SMS Trojans, Ransomware, Adware, Backdoors, Rootkits, Spyware, Botnets, and Installer malware, significantly threaten mobile device security [52,53]. Malware spreads on mobile devices through malicious links in emails or SMS, infected apps from Google Play Stores, third-party sources, or malicious Wi-Fi networks. Significant vulnerabilities in the Android OS include information gain, code execution, denial of service (DoS), overflow, SQL Injection, and privilege escalation [53].

4.2.2. iOS

iOS, introduced in 2007, is a Unix-based operating system that powers popular Apple devices such as iPhones and iPads, ranking as the second most used mobile OS globally. The iOS architecture consists of four layers with specific functions: Core OS handles hardware interactions, Core Services provide data protection and storage features, media supports multimedia processing, and Cocoa Touch enables app development and user interface management [53].

iOS offers robust security compared with Android through a closed system design incorporating device-level protection (e.g., PINs, remote wipe), system-level features (e.g., Secure Enclave, secure boot), and mandatory data encryption. Apple's control over hardware and software makes jailbreaking and unauthorized access challenging. The iOS enhances security through sandboxing, encryption, and automatic data erasure. Applications are isolated from each other to prevent unauthorized access, whereas encryption protects files using hardware and software keys [54]. iOS automatically grants most permissions, thereby reducing user involvement. In addition, the auto-erase feature wipes data after multiple incorrect passcode attempts, offering a higher level of security than Android [55]. According to a McAfee report, iOS malware has surged in recent years, with a 70% increase in malware targeting iPhones and iPads by 2020 [53]. The most common malware on the iOS platform are ransomware, spyware, viruses, trojans, and adware [53,56]. Notable attacks include Pegasus, which exploits zero-day vulnerabilities for surveillance [57]. Additionally, LightSpy Spyware, a sophisticated iOS implant, was infiltrated via compromised news sites [58]. Common vulnerabilities, such as memory overflow, remote code execution, and data leakage, present significant risks to iOS users, highlighting the need for enhanced device security.

4.3. IoT platform

The Internet of Things (IoT), introduced by Ashton in 1999, refers to a network of interconnected devices that collect and exchange data via the Internet or other networks. This is a combination of devices, sensors, networks, computing resources, and software tools. IoT devices fall into two main categories: Consumer IoT, such as personal and wearable smart devices, and Industrial IoT (IIoT), which includes interconnected industrial machinery and energy management devices.

The number of IoT devices is increasing significantly every year. According to Statista, global IoT devices will nearly double from 15.9 billion in 2023 to over 32.1 billion by 2030. By 2033, China

 doi:10.20944/preprints202412.0348.v1

will have the highest number of IoT devices, with approximately 8 billion consumer devices [59]. However, the rapid rise of IoT coupled with insufficient security measures has made these devices prime targets for malware. Recent reports from Zscaler ThreatLabz show a 400% increase in IoT malware attacks [60]. High-profile incidents, such as the Mirai botnet in 2016, exploit weak passwords and unpatched vulnerabilities, enabling DDoS attacks and data exfiltration [60]. IoT malware also takes advantage of other vulnerabilities, such as the absence of software and security updates, insecure networks, poor user security awareness, TCP/IP stack vulnerabilities, and a lack of encryption. Modern IoT malware, including Okane, VPNFilter, and Necurs, increasingly employs brute-force methods, spyware tactics, and anti-virtualization techniques to gain access to devices [14].

4.4. Cloud Environments

Cloud computing enables remote access to computing resources such as storage, applications, networks, and servers via an Internet connection. Conversely, cloud malware is a cyberattack that targets cloud platforms with malicious code or services.

Cloud computing offers three types of services: Platform as a Service (PaaS), Software as a Service (SaaS), and Infrastructure as a Service (IaaS). PaaS provides an environment for programmers to develop, deploy, and test applications, as exemplified by the Azure and Google App Engine. SaaS supports all applications within the cloud environment such as email and office software. IaaS offers hardware resources, computing capabilities, storage, servers, networking devices, and virtual machines [19,61]. Common examples of cloud malware include DDoS Attacks, Hypervisor DoS Attacks, Hypercall Attacks (exploits the hypervisor to gain cloud control), Hyperjacking (when an attacker takes control of a virtual machine for malicious purposes), Exploiting Live Migrations (moves a VM or application without client disconnection from one physical location to another), Ransomware, Spyware, Backdoor, Trojan horse etc.[19,62].

5. Machine Learning Algorithms for Malware Detection

In this section, we present a summary of various machine learning algorithms used for malware detection on diverse platforms, including traditional, ensemble, and advanced deep learning approaches, as outlined in Table 3. Traditional algorithms such as SVM, KNN, and DT are simple yet effective in classifying malicious and benign samples. Ensemble methods, such as RF and Gradient Boosting, enhance the accuracy and robustness by combining multiple models. Deep learning algorithms, including CNN and transformers, excel in processing complex, high-dimensional, and sequential malware data. Techniques such as GAN and Transfer Learning address challenges such as limited datasets and feature extraction. The table underscores the diversity of machine learning methodologies in malware detection. The analysis outcomes of the table are reflected in the pie chart shown in Figure 4, highlighting the key trends in machine learning techniques for malware detection.

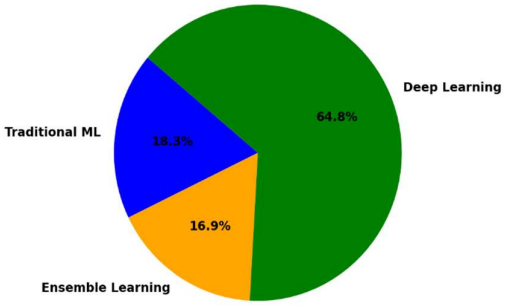


Figure 4. Proportion of algorithm categories in recent malware detection.

 doi:10.20944/preprints202412.0348.v1

Table 3. Summary of machine learning algorithms applied in various studies across diverse platforms for malware detection.

ML techniques	Algorithms	References
Traditional Machine Learning Algorithms		
Support Vector Machines (SVM): This method employs a hyperplane to maximize the margin between malicious and benign samples, proving effective for high-dimensional data.	SVM	[64–66]
K-Nearest Neighbors (KNN): This algorithm classifies samples based on the predominant class of their nearest neighbors, utilizing feature similarity as the primary criterion.	KNN	[52,67]
Logistic Regression (LR): This approach classifies malware by modelling the relationship between features and binary outcomes (malicious or benign) utilizing a sigmoid function. The sigmoid function converts input values to a range of 0 to 1, making it ideal for interpreting results as probabilities. It is used for binary classification tasks, especially in logistic regression and neural networks.	LR	[68–70]
Naïve Bayes (NB): A probabilistic approach that assumes feature independence, which is efficient for text-based malware detection.	NB	[65,66]
Decision Trees (DT): Decision trees are a supervised learning method that classify data by building a tree-like model. The process identifies the most critical features and splits the data into subsets based on these attributes to form nodes. It recursively classifies each node until a final decision is reached as benign or malware	DT	[68,67]
Ensemble Learning Algorithms		
Random Forest (RF): This approach constructs multiple decision trees and aggregates their outputs through majority voting or averaging, thereby enhancing robustness and accuracy.	RF	[65–68,70–72]
Gradient Boosting (e.g., XGBoost, LightGBM): This approach sequentially constructs weak learners, specifically decision trees, to minimize errors, thereby providing high accuracy in the analysis of structured malware data.	Gradient Boosting	[70]
	XGBoost	[67,70]

 doi:10.20944/preprints202412.0348.v1

AdaBoost: This approach focuses on challenging samples by modifying weights during the training process, thereby combining weak classifiers into a robust one.	**AdaBoost**	[66,72]
Bagging: The Bagging technique randomly divides the dataset into multiple subsets (bootstraps) based on instances, each with unique instances, and then aggregates the results from models trained on these subsets to enhance generalization		
Deep learning technique		
Convolutional Neural Networks (CNNs): This approach demonstrates efficacy in image-based malware detection, utilizing automated extraction of spatial features from transformed malware binaries.	CNN	[63,68,73–89]
Recurrent Neural Networks (RNNs): This method Facilitates the analysis of sequential data, including API call sequences and opcode patterns, for behavioral-based malware identification.	RNN	[63,88,90,91]
Long Short-Term Memory (LSTM): A variant of Recurrent Neural Network (RNN) that effectively captures long-term dependencies, particularly applicable for time-series analysis of dynamic malware features.	LSTM	[73,77,79,84,[91–100]
Gated Recurrent Unit (GRU): It is a type of recurrent neural network (RNN) designed to process sequential data, such as time series or text. This model is more computationally efficient than LSTMs due to fewer parameters and the absence of a separate output gate.	GRU	[93]
Generative Adversarial Networks (GANs): This process generates synthetic malware samples for data augmentation, thereby enhancing the efficacy of detection systems with limited datasets.	GAN	[81]
Autoencoders: Autoencoders are unsupervised neural networks used for dimensionality reduction, feature extraction, and anomaly detection. They aim to learn a compressed representation of the input data (encoding) and then reconstruct the input (decoding) as accurately as possible.	VAEs, Sparse Autoencoders etc.	[101]
Transformer Models (e.g., BERT): Transformers are advanced deep learning architectures based on attention mechanisms designed to handle sequential or contextual data effectively.	BERT (Bidirectional Encoder Representations from Transformers)	

 doi:10.20944/preprints202412.0348.v1

Transfer learning (TL): This is a deep learning approach where a model pre-trained on one task or dataset is reused and fine-tuned for a related but different task. It is particularly effective when the target dataset is small or lacks diversity.	Pre-trained CNNs like ResNet, Inception, VGG, ResNet50 etc.	[102–104]
Multilayer Perceptron (MLP): It is a type of artificial neural network (ANN) consisting of multiple layers of nodes. It is commonly used in supervised learning tasks such as classification and regression.	MLP	[52,66,68,71,82,86,105]

 doi:10.20944/preprints202412.0348.v1

Overall Research Trends on Machine Learning Algorithms for Malware Detection Across Different Platforms

Table 3 and the pie chart reveal that deep learning is the leading approach in malware detection research across platforms, with CNNs and LSTMs excelling in image-based and sequential data analysis. Traditional ML techniques such as SVM and KNN remain adequate for high-dimensional feature-based tasks. Simultaneously, ensemble learning methods, such as Random Forest and Gradient Boosting, show substantial accuracy and generalization through model aggregation. These trends highlight the increasing preference for deep learning, while acknowledging the complementary roles of traditional and ensemble models.

6. Application of Machine Learning on Malware Detection

This section reviews recent studies that have utilized the various ML algorithms discussed in Section 5 to develop malware detection models for Windows, Linux, Android, IoT, and cloud platforms.

6.1. PC (Personal Computers) Malware Detection

This section covers malware detection on personal computers, including Windows, Linux, and macOS. Windows, the most widely used OS, are the primary target for malware. Despite Linux's robust permission-based architecture, it is facing growing threats in the server and enterprise settings. With its increasing popularity, macOS has increased the risk of malware. Detection methods employ static, dynamic, and hybrid analyses, which are frequently enhanced using machine learning, to counter evolving threats.

6.1.1. Malware Detection in Windows platform

In this subsection, we provide an extensive review of machine learning-based malware detection techniques for the Windows platform are summarized in Table 4.

 doi:10.20944/preprints202412.0348.v1

Table 4. Summary of reviewed models for Windows-based malware detection: dataset sources, feature details, and experimental result.

Reference	Data source	Feature category	Features	ML algorithms	Result (accuracy)	Limitations
			Static feature-based malware detection			
[63]	Malimg	Static	Opcode sequences	Deep RNN	96%	It requires significant computational resources
[73]	Microsoft BIG 2015	Static	Opcodes, images, byte sequence, etc	DNN, LSTM, and CNN.	98.35%	It is useless against zero-day malware.
[102]	BIG 2015, Malimg, MaleVis and Malicia dataset	Static	2D images	DenseNet	98.23%	It has high false negatives and highly imbalanced datasets
[74]	Microsoft BIG 2015	Static	Image-based opcode features	CNN	99.12%	Outdated dataset
[103]	Malimg dataset, Microsoft BIG 2015	Static	Grayscale images from PE files	VGG16, VGG19, ResNet50, and inceptionV3	98.92%	Cannot detect advanced-packed malware
[106]	Malimg	Static	Static signatures	ATT-DNNs	98.09%	Cannot detect obfuscated malware
[75]	Malware API-class	Static	Executable file to static images	CNN	98.00%	_
[76]	VirusShare, Hybrid-Analysis	Static	Executable file to static images	Xception Convolutional Neural Network (CNN)	98.20%	_
[107]	Microsoft BIG 2015	Static	Malware binary files into static images	DNN	97.80%	_
			Dynamic feature-based malware detection			
[92]	VirusShare	Dynamic	Sequences of API calls	Bi-LSTM	97.31%	Limited to execute samples in a Windows 7 environment.
[108]	Custom datasets	Dynamic	Sequences of API calls	Markov chain representation	99.7%	-
[109]	VirusTotal	Dynamic	API calls	LSTM	95%	Limited to execute samples in a Windows 7 environment.
[93]	VirusTotal	Dynamic	API call sequences	LSTM and GRU	96.8%	Highly imbalanced dataset

 doi:10.20944/preprints202412.0348.v1

[64]	CA Tech-neologises VET Zoo	Dynamic	Run time behaviour	MRed, ReliefF, SVM	99.499%	High computational complexity
[94]	Audit log events	Dynamic	Process names, action types, and accessed file	LSTM	91.05%	High false positives and lack of scalability
[77]	Multiclass dataset (Ember Dataset, private dataset)	Dynamic	loaded DLLs, registry changes, API call sequences, file changes, and	CNN-LSTM	96.8%	Susceptible to adversarial attacks
Hybrid feature-based Malware Detection Techniques						
[78]	VirusTotal	Hybrid (Static and dynamic)	Combination of static and dynamic features (PE section, PE import, PE API, and PE images)	CNN	97%	Failed to validate the robustness against adversarial attacks
[71]	The Korea Internet & Security Agency (KISA)	Hybrid	Size of file and Header, Counts of file sections. Entropy, File system changes API call, DLL loaded info, network activities, etc.	RF, MLP	85.1%	Extensive time is needed for feature extraction
[104]	VirusShare	Hybrid	Image-based static and dynamic features	VGG16	94.70%	-
[110]	VirusShare	Hybrid	Function Length Frequency Representation, Registry activities, API calls, and file operation features	SVM	97.10%	Small dataset
[79]	VirusTotal	Hybrid	Opcodes and system calls	CNN, LSTM, and an attention-based LSTM	99%	Lack of diverse features.
Memory-feature-based malware detection techniques.						
[80]	Dumpware10	Memory	Memory images of running processes	CNN	98%	Malware processing cost is high under limited resource capabilities

 doi:10.20944/preprints202412.0348.v1

[81]	Dumpware10, BIG2015 dataset	Memory	Memory images of running processes	GAN and CNN	99.86% for BIG2015 dataset	Only one type of data, like bytes, is used. Need to make the dataset more diverse.
[82]	CIC-MalMem-2022 https://www.unb.ca/cic/datasets/malmem-2022.html	Memory	Memory images of running processes	CNN and MLP	99.8%	Training time complexity and vulnerability to adversarial attacks
[68]	CIC-MalMem-2022 https://www.unb.ca/cic/datasets/malmem-2022.html	Memory	Multi-memory features	RF, DT, LR, MLP and CNN	99.89%	-

 doi:10.20944/preprints202412.0348.v1

Static feature-Based Malware Detection Techniques

Jeon and Moon [63] proposed a DL-based malware detection method using static opcode sequences with dynamic RNN and CNN. A convolutional autoencoder compresses long opcode sequences, and a recurrent neural network classifies malware using the features generated by the autoencoder. This method achieved 96% accuracy and 95% true-positive rate. However, it requires substantial computational resources owing to the inter-procedural control-flow analysis, making it less suitable for resource-limited systems. Snow et al. [73] developed a multi-modal deep-learning-based malware detection method using the Microsoft BIG 2015 dataset. Although the model achieved a high accuracy rate of 98.35%, it proved ineffective against zero-day malware that evaded detection with new static signatures. A previous study [102] employed a CNN-based pre-trained DenseNet model for malware detection by converting benign and malicious binaries into 2D images. Experiments on Malimg, BIG 2015, MaleVis, and Malicia datasets showed 98.23% accuracy on Malimg but revealed high false negatives and used highly imbalanced datasets. Darem et al. [74] implemented a deep CNN-based model to detect malware using opcode-level features from malware and benign binary files converted into images for training. The model achieved a detection rate of 99.12% %on the Microsoft BIG 2015 dataset. However, outdated datasets can affect the detection of new and unseen malware. Kumar & Janet [103] proposed an image-based deep transfer learning model for detecting Windows malware using a pre-trained deep CNN. The model efficiently extracts high-level features from grayscale images of Windows executables, conserving resources and time; however, it struggles to identify advanced-packed malware. A study in [106] introduced an attention-based deep neural network (ATT-DNN) for malware detection, extracting static features from executable files. Despite achieving a high accuracy of 98.09%, its use is limited to malware detection based on static signatures. The research in [75,76,107] also focused on malware detection via static image analysis.

Dynamic Feature-Based Malware-Detection Techniques

Li et al. [92] developed a DL model for malware detection in executables using API call sequences within a Cuckoo sandbox, achieving an F1-score of 0.9724 and 97.31% accuracy on new sequences. The limitations of this study include its focus on Windows 7 executables and the potentially reduced effectiveness against zero-day malware over time. In [108], contextual analysis of API call sequences was utilized to enhance the dynamic detection and prediction of Windows malware, thereby improving both accuracy and adaptability to evolving threats. By employing the Markov chain method, they achieved an average accuracy of 99.7%. Catak et al. [109] proposed an LSTM-based malware detection method that achieved 95% accuracy and 0.83 F1 score using a behavioral dataset of API calls. They also released a new, publicly available API call dataset for malware detection. Aditya et al. [93] used LSTM and GRU deep learning models to classify malware based on API call sequences and achieved 96.8% accuracy with LSTM. However, their dataset was highly imbalanced, with only 1,035 benign samples of 8,142. In [64], a hybrid framework combining multiple complementary filters with a wrapper feature selection method was proposed to identify critical run-time behavioral traits of malware. The ML algorithms, including MRed, SVM, and Fisher, achieved a detection accuracy of 99.499%. Ring et al. [94] used an LSTM-based model to detect malware based on audit-log features. However, it suffers from high false positive rates and lacks the evaluation of larger datasets to assess the model's scalability. Jindal et al. [77] proposed Neurlux, a stacked ensemble of CNN-LSTM with an attention mechanism to detect malware in Windows systems using dynamic features effectively but is susceptible to adversarial attacks.

Hybrid-Feature-Based Malware Detection Techniques

Hybrid feature-based learning approaches have shown promise in cybersecurity, outperforming single-type feature methods. By combining diverse feature types, such as static, dynamic, and

 doi:10.20944/preprints202412.0348.v1

memory-based, these techniques enable learning from multiple semantic perspectives, leading to an enhanced model accuracy for malware detection and classification.

The authors of article [78] created a CNN-based hybrid malware classification model for Windows, integrating static features from the executable section, static API call imports, dynamic API calls, and executable file images, achieving a detection accuracy of 97%. However, this method does not validate the effectiveness of the combined feature sets against adversarial attacks. AI-HydRa [71] represents an advanced hybrid malware classification method that combines RF and deep learning, achieving a mean detection rate of 0.851 with a standard deviation of 0.00588 over three tests. Huang et al. [104] introduced a hybrid method using static images and dynamic API call sequence visualizations to classify malicious behaviors. Utilizing a CNN (VGG16) for feature extraction, the technique attained 94.70% accuracy but had difficulty accurately identifying specific malware types, including password-stealing (PSW) Trojans and some outdated variants. RansHunt [110] integrated static and dynamic features for improved ransomware detection using an SVM, achieving an accuracy of 97.10%, outperforming traditional anti-virus solutions. Darabian et al. [79] used static and dynamic features from 1,500 cryptojacking malware samples. They used opcodes and system calls to construct CNN, LSTM, and attention-based LSTM classification models, achieving 95% accuracy in static analysis and 99% accuracy in dynamic analysis.

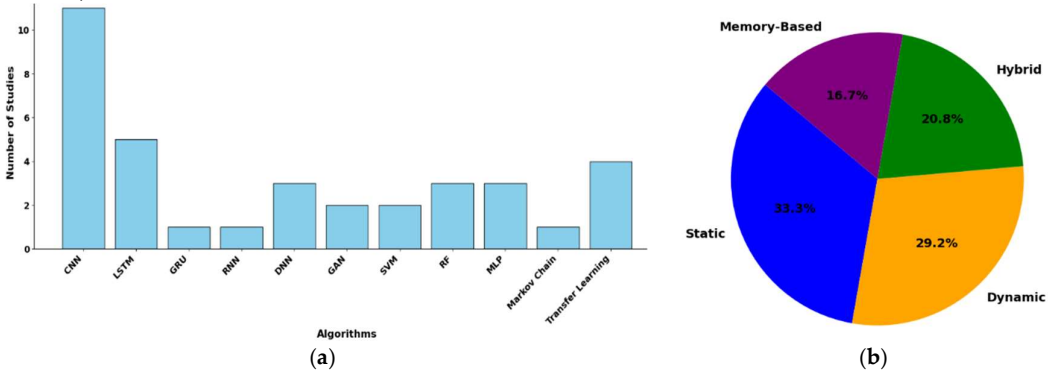
Karbab et al. [111] introduced SwiftR, an approach that detects ransomware attacks by integrating various static and dynamic features from benign and malicious executable file reports. The proposed method achieved an F1 score of 98%.

Memory-Feature-Based Malware Detection Techniques

Malware detection using static or dynamic analysis is insufficient for advanced memory-resident malware and obfuscated malware. Thus, contemporary research emphasizes memory analysis methods that are effective in detecting sophisticated malware variants [41]. The study [80] developed a CNN model was developed that uses memory images of both suspicious and benign processes to detect malware attacks with a detection rate of 98%, leveraging features extracted from grey-level co-occurrence matrices and local binary patterns. Another study [81] proposed a DL-based approach that integrated GAN and CNN models, achieving 99.60% detection accuracy on unseen samples when tested on the DumpWare10 dataset to identify advanced malware by visualizing running processes. In addition, Naeem et al. [82] developed a high-performance stacked CNN and MLP model using memory images that achieved an accuracy of 99.8%. However, it has limitations in terms of the training time complexity and susceptibility to adversarial attacks. In [68], the authors used the latest dataset, the CIC-MalMem-2022 dataset, to develop a CNN-based detection model that detects obfuscated malware in memory.

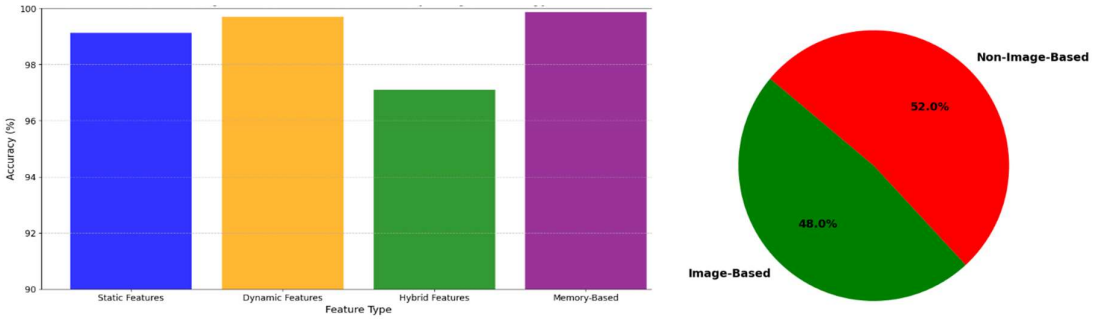
Summary of Key Trends and Insights on Malware Detection in Windows Platforms

A summary of the malware detection methods for Windows is presented in Table 4. The table outlines the studies with respect to their data collection source, feature type, features, ML algorithms used, detection accuracies, and limitations. Figure 5 illustrates the distribution of the techniques, features, and evaluation datasets used in these studies.



(**a**) (**b**)

 doi:10.20944/preprints202412.0348.v1



(c) (d)

Figure 5. Distribution of detection feature types, algorithms, accuracy by feature type, and image and non-image features. (a) Detection algorithms (b) Proportion of detection feature type (c) Accuracy of malware detection by feature type (d) Image-based vs non-image-based detection techniques.

Dataset evaluation: Table 4 reveals that many recent studies have relied on outdated datasets, such as Malimg (2011) and Microsoft BIG 2015. VirusTotal and Virus Share remain the most popular data sources for Windows malware detection systems, followed by dumpware10 and Hybrid Analysis. com. In contrast, newer datasets such as CIC-MalMem-2022 provide updated benchmarks. However, the prevalence of outdated datasets and the lack of diversity in recent studies have limited their effectiveness against zero-day malware and emerging threats.

Detection algorithms used in the studies: The bar chart in Figure 5 (a) illustrates the distribution of detection algorithms employed in various detection techniques, highlighting their relative popularity among the studies. Convolutional Neural Networks (CNNs) are the most utilized algorithms, likely because of their efficiency in handling images and spatial data. LSTM networks also stand out and are likely to be favored because of their strength in temporal or sequential data processing. Algorithms such as GANs and transfer learning are used less frequently and hint at emerging or specialized applications. However, GRU, RNN, and Markov Chains appear less favored, possibly because of their limited generalizability or lower performance in detection tasks. The chart collectively underscores the significance of choosing algorithms that align with the nature of the problem and the data characteristics.

Detection feature type: The chart 5 (b) highlights the distribution of feature types in malware detection techniques. Static features lead by 33.3%, favoring their simplicity and effectiveness against known malware. Dynamic features follow closely at 29.2%, offering strong runtime analysis capabilities but requiring controlled execution environments. Hybrid features, at 20.8%, integrate static and dynamic methods for comprehensive detection but involve higher computational demands. Memory-based features, representing 16.7%, are powerful for analyzing runtime data, such as API calls, but are less commonly used because of their resource-intensive nature.

Accuracy of malware detection techniques by feature type: The bar chart as outlined in Figure 5 (c) compares the accuracy of malware detection techniques based on four feature types: static, dynamic, hybrid, and memory. Memory-based features achieved the highest accuracy (~99.89%), demonstrating their effectiveness in capturing runtime behaviors, although they may require higher computational resources. Dynamic features also perform well (~99.49%), leveraging runtime analysis, whereas static features (~99.12%) offer robust results through code and signature analysis. Hybrid features (~99%) combine static and dynamic methods but do not significantly outperform individual approaches. Overall, memory-based and dynamic features demonstrated the highest potential for accurate malware detection.

Image-based vs. non-image-based detection techniques: The pie chart in Figure 5 (d) shows a close competition between the non-image-based (52%) and image-based (48%) detection methods. While non-image-based methods lead slightly because of their flexibility with diverse data types, image-based approaches are emerging as powerful tools in malware detection. By converting malware binaries into images, image-based methods use CNNs to analyze spatial patterns and effectively identify complex obfuscated malware. The availability of labelled malware datasets,

 doi:10.20944/preprints202412.0348.v1

efficient pre-trained models, and generalization capabilities further drive their adoption, reflecting the growing significance and scalability of image-based methods in modern malware detection.

6.1.2. Malware Detection in Linux OS

Researchers have also utilized various ML algorithms to detect malware attacks on Linux systems. The ML-based Linux detection techniques are listed in Table 5. Xu et al. [105] developed a graph-based Linux malware detection system called HawkEye that achieved 96.82% accuracy. Hwang et al. [112] also demonstrated the effectiveness of deep learning for Linux threat detection, using a large dataset of 10,000 malicious and 10,000 benign files to train and test a DNN model. Bai et al. [72] proposed a Linux malware detection method that analyzes system calls in ELF executable symbol tables using 756 benign and 763 malware ELF samples. They achieved up to 98% accuracy with various classifiers, including J48, random forest, AdaBoostM1, and IBk. Landman and Nissim's Deep-Hook [83] used CNNs to analyze VM-captured memory dumps and identify Linux malware with up to 99.9% accuracy. Similarly, another study [43] classified malware using behavioral features from volatile memory.

Table 5. Summary of reviewed models for Linux-based malware detection: dataset sources, feature details, and experimental result.

Reference	Data source	Feature category	Features	ML algorithms	Result (accuracy)
[105]	AndroZoo, VirusShare, and clean Ubuntu libraries.	Static	Assembly instructions (control flow graphs)	MLP	96.82%
[112]	VirusShare	Static	Strings from binary data	DNN	94%
[72]	VX heavens	Dynamic	System calls	J48, random forest, AdaBoostM1 (J48), and IBk	98%
[83]	VirusShare	Memory	Memory dumps	CNN	99.9%
[43]	VirusTotal and ViruShare.	Memory	Multi-memory features	DNNs	98.8%

Summary of Key Trends on Malware Detection in Linux Platform

Most current studies have focused on malware detection in Windows and Android platforms, with few addressing advanced ML-based malware detection in Linux. According to current literature, Linux-based malware detection has advanced through the integration of diverse machine learning algorithms, feature types, and datasets, achieving high accuracy. Memory-based detection, in particular, has gained popularity owing to its effectiveness in identifying sophisticated threats. Owing to 's widespread adoption of the Linux OSin online supercomputers and devices globally, cybercriminals have increasingly targeted Linux-based devices. Thus, the success of deep learning in Windows and Android indicates its potential for Linux malware detection. Additionally, exploring hybrid models and cross-platform techniques could enhance the detection capabilities and adapt to the evolving landscape of Linux malware.

6.1.3. Malware Detection in macOS

Despite the rising threats of OS X malware, research on its detection remains scarce, with only a few studies focusing on malware detection on the macOS platform. For example, a study [113] proposed OS X malware and rootkit detection by analyzing static file structures and tracing memory activities. Pajouh et al. [114] developed an SVM model with novel library call weighting for OS X

 doi:10.20944/preprints202412.0348.v1

malware detection, attaining 91% accuracy on a balanced dataset. SMOTE-enhanced datasets increased the accuracy to 96%, with slight false alarm increases, indicating that larger synthetic datasets enhance accuracy, but may impact false-positive rates.

Summary of Key Trends on Malware Detection in macOS Platform

The application of machine learning to OS X malware detection is underexplored, likely owing to the scarcity of suitable datasets and the difficulty in collecting malware samples. Future research should focus on overcoming these challenges to enhance machine learning techniques for detecting OS X malware.

6.2. Malware Detection in Mobile Platform

The increase in mobile device usage, mainly Android, has led to increased malware threats. This section reviews machine learning techniques for detecting malicious applications on both Android and iOS platforms.

6.2.1. Android Malware Detection

This subsection examines ML-based Android malware detection techniques categorized by APK file features, with the dataset details summarized in Table 6.

Table 6. Summary of reviewed models for Android-based malware detection: dataset sources, feature details, experimental result, and limitation.

Reference	Data source	Feature category	Features	ML algorithms	Accuracy	Limitations
Static feature-based Android malware detection techniques.						
[115]	MalGenome	Static	Call graphs	GCN	98.99 %	Lack of representative of real-world scenarios.
[84]	Contagio Mobile	Static	Opcode sequences	CNN-LSTM	91.42 %	Unable to manage obfuscated malware
[69]	MalDroid-2020 dataset	Static	Opcode sequences (histograms of n-grams)	LR	93.56 %	Adversarial attack resistance and handling evolving malware are not addressed.
[95]	CIC-Inves2017	Static	Opcode sequences	LSTM	96%	Small dataset (1,500 apps)

 doi:10.20944/preprints202412.0348.v1

[70]	Drebin, VirusShare, AndroZoo	Static	Permissions, Intents	Base models (LR, MLP, and SGD), Ensemble learning	99.1%	-
[85]	Drebin dataset	Static	Opcode sequences, Permissions, API calls	CNN	99.92 %	lack of malware diversity and scalability
Dynamic feature-based Android malware detection techniques						
[65]	McAfee	Dynamic	Actions/Events	Base models (NB, SL, SVM Linear, SVM RBF, J48, PART, RF), deep learning	97.8%	-
[96]	Google Play Store https://play.google.com/store/games?pli=1	Dynamic	API calls	Bi-LSTM	97.22 %	High detection time
[116]	Drebin dataset	Dynamic	Network traffic Permissions, Intents, API calls	C4.5	97.89 %	Small dataset
[97]	MalGenome	Dynamic	System call sequences	LSTM	99.23 %.	-
[98]	Custom dataset	Dynamic	API and system call sequences	LSTM	96.8%	
Hybrid feature-based Android malware detection techniques						
[65]	McAfee	Hybrid	Permissions, Intents, API Calls, Actions/Events	Base models (NB, SL, SVM Linear,	99.6% detection	-

 doi:10.20944/preprints202412.0348.v1

				SVM RBF, J48, PART, RF), deep learning		
[52]	Contagio Mobile, http://contagiominidump.blogspot.com/ VirusShare and Genome	Hybrid	Runtime behaviors across various levels—kernel, application, user, and package	K-NN, LDC, QDC, MLP, Parzen Classifier (PARZC) and RBF	96%	This method is susceptible to mimicry attacks and ineffective against unknown malware.
[99]	VirusShare, Drebin, DroidAnalytics and CICInvesAndMal2019/2000 https://www.unb.ca/cic/datasets/invesandmal 2019.html.	Hybrid	Permissions requests, API and system call sequences, opcode sequences, and graph structures, including abstract syntax trees, control-flow, and data-flow graphs.	Bi-LSTM and GNN	95.94 %	Need more scalable static analyses
[101]	CICMal- Droid2020	Hybrid	Permissions, intents, system calls, composite behaviors, and network traffic packets.	Pseudo-label stacked auto-encoder (PLSAE)	98.28 %	-
	Memory feature-based Android malware detection techniques.					
[86]	AndroZoo project https://androzoo.uni.lu/	Memory	Process memory dumps	Ensemble of MLP	94.3%	Vulnerable to

 doi:10.20944/preprints202412.0348.v1

	and CNN	adversarial attacks

Static Feature-Based Malware Detection Techniques

A study [115] proposed GDroid, a graph convolutional neural network model for detecting Android malware through API call patterns using static analysis. Although effective in detecting malicious apps, its accuracy decreases in real-world Android devices.

Pektaş and Acarman [84] proposed a CNN- and LSTM-based model utilizing static features, including opcodes, API calls, and call graphs, for Android malware detection. Despite achieving 91.42% accuracy and 91.91% F-measure on unknown samples, the model's dependence on static features may restrict its effectiveness against obfuscated malware. Similarly, in [69], the authors proposed H-LIME, a novel XAI method that enhances LIME by incorporating opcode-sequence hierarchy for better Android malware detection explanations. They evaluated H-LIME using the MalDroid-2020 dataset, and H-LIME outperformed LIME in explanation quality and efficiency, but they faced challenges with shorter programs in real-world malware. Lakshmanarao and Shashi [95] created an LSTM-based malware detection model using opcode sequences from Android apps, achieving 96% accuracy, albeit on a limited dataset of 1,500 apps. Potha et al. [70] created an ensemble model combining LR, MLP, and Stochastic Gradient Descent (SGD), demonstrating that larger, homogeneous ensembles with feature selection outperformed smaller ones, achieving strong AUC and accuracy on Android malware datasets. Furthermore, Aamir et al. [85] introduced the AMDDL model, achieving a 99.92% accuracy in malware detection using CNNs. This study highlights the challenges related to limited malware diversity, deep learning interpretability, and scalability.

Dynamic Feature-Based Malware Detection Techniques

Ma et al. [96] proposed Droidect, a Bi-LSTM-based model for classifying malicious Android apps, achieving 97.22% accuracy on a dataset of 11,982 benign and 9,616 malicious files. Despite its success, this model suffers from long detection times. Wang et al. [116] presented a malware detection technique employing network traffic analysis and the C4.5 algorithm, achieving a 97.89% detection rate on the Drebin dataset, outperforming current methods. The study in [97] introduced MemDroid, an LSTM-based detection method trained on Androzoo malware samples. Apps were run in a sandbox to capture system call sequences, which were used to train the LSTM classifier, achieving 99.23% malware detection accuracy. The study in [98] used LSTM to develop classifiers for detecting Android malware via dynamic API and system calls, achieving F1-scores of 0.967 and 0.968, respectively, across different datasets.

Hybrid-Feature-Based Malware Detection Techniques

Alzaylaee et al. [65] introduced DL-Droid, a deep learning-based framework for Android malware detection using static and dynamic analysis. They achieved 97.8% detection with dynamic features and 99.6% with combined features, taking 190 s/app on average. Saracino et al. [52] introduced MADAM, an Android malware detection system analyzing kernel, application, user, and package-level features. MADAM detected over 96% of malicious apps in a 2800-app test but is susceptible to mimicry attacks and cannot identify unknown malware. Wu et al. [99] presented DeepCatr, a hybrid learning approach for Android malware detection, which combines text mining and call graphs with bidirectional LSTM and graph neural networks, achieving accuracies of 95.94% and 95.83% on 18,628 samples. Mahdavifar et al. [101] created a semi-supervised deep learning model for Android malware detection, employing a stacked auto-encoder trained on hybrid features, obtaining a 98.28% accuracy and 1.16% false positive rate.

Memory-Feature-Based Malware Detection Techniques

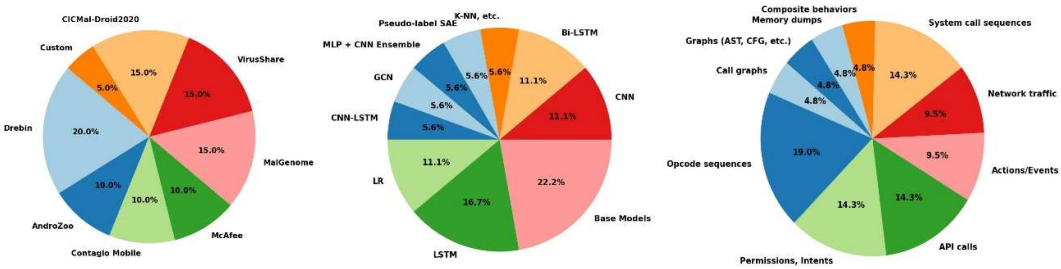
Memory analysis has been utilized to develop deep learning models for detecting obfuscated and memory-resident Android malware. A framework combining weak learners (CNNs) and a meta-

 doi:10.20944/preprints202412.0348.v1

learner (MLP) to create a deep-stacked ensemble model along with an explainable AI approach for result interpretation and validation was proposed by Naeem et al. [86]. The framework achieved an accuracy of 94.3% using 2,375 images in an empirical evaluation.

Summary of Research Trends on Malware Detection in the Android Platform

Table 6 summarizes mobile device malware detection systems, including datasets, features, detection algorithms, and study accuracy. Figure 6 shows the distribution of the dataset sources, techniques, and features used in these studies.



(**a**) (**b**) (**c**)

Figure 6. Distribution of detection techniques, detection features, and evaluation datasets used in mobile malware detection solutions. (a) Proportion of dataset sources (b) Detection algorithms (c) Detection features.

Proportion of datasets used: The pie chart in Figure 6 (a) reveals a clear preference for established datasets in Android malware detection studies. Drebin emerges as the most-used dataset (31%), owing to its extensive malware diversity and widespread acceptance as a benchmark in the field. Medium-utilized datasets, including MalGenome, VirusShare, and CICMal-Droid2020 (23% each), are valued for their reliability and growing prominence in the evaluation of detection techniques. The relatively low adoption of custom datasets highlights the focus on standardized datasets, limiting opportunities for novel malware detection approaches tailored to evolving threats.

Detection algorithms: The pie chart in Figure 6 (b) demonstrates that base models (31%), including Logistic Regression and Random Forest, are the most commonly used detection algorithms, valued for their reliability, simplicity, and ease of implementation. Deep learning methods, such as LSTM (23%) and CNN (15%), are gaining popularity owing to their ability to process complex and large-scale malware patterns effectively. Hybrid techniques (8%), ensemble models (8%), and exploratory approaches, such as Pseudo-label SAE and K-NN (8% each), showcase ongoing innovations aimed at improving detection accuracy and robustness. This distribution underscores the balance between traditional dependable methods and modern complexity-driven approaches to malware detection**.**

Detection features: As shown in Figure 6 (c), the chart indicates that opcode sequences (31%) are the most commonly used features because of their effectiveness in static analysis. Permissions, intents, and API calls (23% each) are essential for identifying behavioral anomalies. System call sequences (23%) and network traffic (15%) are gaining prominence in runtime analyses. Composite behaviors and memory dumps (8% each) remain underexplored, likely due to their complexity and resource demands

6.2.2. Malware Detection in iOS

In [117], the authors focused on identifying iOS malware using static analysis and machine learning, achieving a high precision of 0.971 and recall of 1.0. It addresses the underexplored domain of iOS malware detection owing to the platform's closed source nature. Zhou et al. [118] examine the risks of legitimate applications being hijacked for malware communication. They presented the ChanDet model to identify potential channel applications and proposed mitigation strategies. Mercaldo and Santone[119]successfully classified 50,000 Android and 230 iOS malware samples using deep learning on grayscale images of executables, tackling obfuscation and false positives.

 doi:10.20944/preprints202412.0348.v1

Summary of Research Trends on Malware Detection in iOs Platform

The current literature highlights advancements in iOS malware detection, leveraging machine learning and static analysis to address the platform's closed-source challenges. Researchers have introduced high-precision models and deep learning techniques, such as those using executable images, to mitigate obfuscation. Despite these advancements, challenges such as limited datasets, lack of hybrid analysis, and insufficient attention to real-time cross-platform threats persist. Future work should focus on expanding the datasets, utilizing transfer learning, enhancing anti-obfuscation methods, and developing comprehensive detection frameworks.

6.3. Malware Detection in IoT Platform

This section compares the surveys on machine-learning-based malware detection in IoT, which are summarized in Table 7.

Ali et al. [66] used machine learning algorithms on the IoT-23 dataset to detect IoT network anomalies. The RF algorithm demonstrated the highest efficacy, achieving 99.5% accuracy. Sudheera et al. [87] introduced Adept, a distributed framework that detects and classifies attack stages in IoT networks through local anomaly detection, pattern mining for correlated alerts, and machine learning-based classification. This method can identify five times more attack patterns with 99% accuracy in classifying attack stages. Vasan et al. [88] proposed a cross-architectural malware detection method suitable for diverse IoT processor architectures, such as MIPS, PowerPC, and SPARC.

Researchers have used sandboxing as a dynamic method to detect malware in IoT environments. However, existing sandboxes are inadequate for resource-limited IoT settings, lack support for diverse CPU architectures, and do not offer library sharing options [120]. Hai-Viet et al. [89] proposed an IoT botnet detection approach using system call graphs and a one-class CNN classifier, which improved sandboxing to capture system behaviors and utilized graph features for robust detection, overcoming dataset imbalance and architectural constraints, attaining 97% accuracy. Jeon et al. [100] introduced HyMalD, a hybrid IoT malware detection method using Bi-LSTM and SPP-Net to analyze static and dynamic features, extracting opcode and API call sequences for classification. It achieved 92.5% accuracy, surpassing the 92.09% accuracy of the static analysis. Researchers have now converted network traffic or OpCode into 2D images for malware detection using visual methods. Shire et al. [90] utilized visual detection techniques in IoT malware detection, transforming network traffic into 2D images for machine learning analysis. He et al. [67] proposed an efficient and scalable lightweight IoT intrusion detection method utilizing feature grouping, which attained over 99.5% accuracy on three public IoT datasets while consuming fewer computational resources than baseline methods. Jiang et al. [91] proposed FGMD, a framework that protects IoT intrusion detectors from adversarial attacks, preserving efficacy and performance. Conversely, Zhou et al. [121] introduced HAA, a hierarchical adversarial attack strategy for GNN-based IoT detectors, which reduces the classification accuracy by over 30% through minor perturbations and node prioritization techniques.

 doi:10.20944/preprints202412.0348.v1

Table 7. Summary of reviewed models for IoT-based malware detection: dataset sources, feature details, experimental result, and limitation.

Reference	Data source	Feature category	Features	ML algorithms	Accuracy (%)
[66]	IoT-23 dataset	**Static**	Network capture files include IP address, ID of the capture, protocol, etc.	RF, NB, MLP, SVM, and AdaBoost**.**	99.5
[87]	NSS Mirai Dataset latest relevant, balanced data sets https://www.stratosphereips.org/datasets-iot23	Static	Alert level (Source and Destination IP Addresses, C&C activities, Protocol) and packet level features ((IP address or port number, packet size, etc.)	CNN	99
[88]	ARM-based IoT	Static	OpCode features	RNN and CNN	99.98
[89]	Executable and Linkable Format (ELF) file templates are executed in the QEMU sandbox.	Dynamic	System call graph	CNN	97
[100]	KISA-data challenge 2019-Malware.04, provided by the Korea Internet & Security Agency	Hybrid	Opcode and API call sequences	Bi-LSTM and spatial pyramid pooling network (SPP-Net)	92.09
[90]	Network traffic is collected from external repositories.	Dynamic	2D Image-Based Network Traffic Features	Neural network	91.32
[67]	Bot-IoT, MedBIoT, and MQTT-IoT-IDS2020 datasets	Dynamic	Packet-level metadata of the raw PCAP file	DT, RF, K-nearest neighbor (KNN), and extreme gradient boosting (XGB)	99.5 with RF
[91]	MedBIoT dataset [122]. IoTID (IoT network intrusion dataset) http://dx.doi.org/10.21227/q70p-q449.	Dynamic	PCAP files	LSTM, RNN and DT, respectively.	98.71
[121]	UNSW-SOSR2019	Static	Network packets (source IP, destination IP, timestamp, traffic flows, etc.)	Graph neural network (GNN)	-

 doi:10.20944/preprints202412.0348.v1

Summary of Research Trends on Malware Detection in IoT Platform

Table 7 summarizes IoT malware detection systems, detailing data sources, features, detection models, and accuracies.

Dataset utilization and challenges: Table 7 shows that datasets such as IoT-23, MedBIoT, and Bot-IoT are frequently utilized. However, issues such as dataset imbalances and limited architectural diversity remain unclear.

Diverse feature categories: IoT malware detection employs a mix of static (e.g., network packet features and opcode sequences) and dynamic features (e.g., system call graphs and network traffic metadata), with some methods integrating hybrid approaches (e.g., opcode and API calls).

Machine learning algorithms: A wide range of ML algorithms, including RF, CNN, RNN, Bi-LSTM, and GNN, has been utilized. The RF and CNN models dominate owing to their high accuracy and adaptability to IoT-specific constraints.

Image-based detection advances in sandboxing: Image-based approaches and sandboxing improvements, such as QEMU-based execution and system behavior capture, have addressed limitations in resource-constrained IoT environments while enhancing the malware detection performance.

Adversarial vulnerabilities: Many existing models are susceptible to adversarial attacks, which reduces their real-world applicability.

6.4. Malware Detection in Cloud Platform

Malware detection in cloud platforms is becoming increasingly vital as organizations move data and services to the cloud. Unlike traditional systems, cloud environments pose unique challenges due to their distributed architecture, multi-tenancy, and scalability. The dynamic and large-scale nature of the cloud enables rapid malware propagation, outpacing traditional detection methods. Detection agents on cloud servers provide security services, allowing users to upload files and receive malware reports.

Xiao et al. [123] proposed a cloud-based malware detection scheme utilizing Q-learning to optimize the offloading rate for mobile devices without prior knowledge of trace generation or radio bandwidth. They employed the Dyna architecture and post-decision state learning to enhance performance and expedite the reinforcement learning process. Testing revealed that their scheme improved detection accuracy by 40%, reduced delay by 15%, and increased mobile device utility by 47% with 100 devices, thereby enhancing overall performance. Additionally, Yadav R. Mahesh [61] introduced a malware detection system for cloud environments using a novel consolidated Weighted Fuzzy K-means clustering algorithm with an Associative Neural Network (WFCM-AANN). The proposed classifier identified malware with a high detection precision of 92.45%, surpassing existing classifiers.

Summary of research Trends on Malware Detection in IoT Platform

According to the studies reviewed in this work, advanced methods such as Q-learning and Weighted Fuzzy K-means clustering combined with neural networks have shown promising results. However, large-scale and highly variable cloud environments render malware detection challenging. Hence, existing solutions often lack scalability to handle the rapid increases in traffic and malware propagation. To overcome this issue, adaptive ML models that can efficiently handle the dynamic and multi-tenant nature of cloud systems can be developed.

7. Challenges Associated with Platform-Specific and Cross-Platform

This section discusses platform-specific research challenges related to malware detection, such as Windows, Linux, macOS, Android, iOS, IoT and Cloud. We also present the cross-platform challenges in ML-based malware detection.

Windows platform-

 doi:10.20944/preprints202412.0348.v1

- The use of outdated Windows versions, which no longer receive official support, exposes the systems to unpatched vulnerabilities.
- The variety of third-party applications on Windows expands the attack surface, thereby increasing the risk of exploitation.
- The rise of fileless malware, which primarily lives in memory, presents challenges for traditional detection and mitigation methods.
- Inconsistent user behavior and poor adherence to security best practices increases vulnerability.

Linux platform: The primary challenges in Linux malware research include the following.

- Linux systems support diverse computer architectures, requiring analysts to create specific malware analysis codes for each architecture, leading to high costs and operational complexity owing to extensive code management.
- The analysis environment may lack the necessary loader for the ELF file format, thereby preventing sample execution.
- Constructing refined datasets is difficult because of the varied devices, vendors, and architectures of Linux systems.
- Moreover, complexity demands expert manual analysis.

macOS platform-

- As macOS gains market share, it increasingly targets malware, which requires continuous advancements in detection techniques.
- The limited tools available for malware analysis of macOS hinder large-scale studies.
- Owing to the historically low prevalence of malware, MacOS users may be less vigilant about security risks.

Android platform-

- Android's dependency on multiple manufacturers slows OS updates, leaving many outdated devices and exposed to security risks.
- Third-party Android apps elevate malware risks, thereby threatening device security and user privacy.
- Unlike iOS, Android allows users to control permissions, potentially enabling malicious apps to misuse the granted access.
- The variety of Android devices and OS versions complicate uniform patching and security protocols.
- Android's open-source framework enables adversaries to examine their code, facilitating reverse engineering and exploitation creation.

iOS platform-

- iOS's auto-erase feature of iOS enhances security but may cause unintended data loss following unsuccessful login attempts.
- Despite advanced Face ID, earlier iOS versions were vulnerable to photos or masks, compromising security.
- iOS apps use obfuscation to prevent reverse engineering; however, skilled attackers can bypass these defenses to access sensitive data.

In summary, Android's flexibility through open-source and diverse devices creates scalability but risks security, whereas iOS enhances security with strict policies, thus limiting flexibility. Future research should focus on balancing security, usability, and standardization.

IoT platform: The rapid growth and heterogeneity of IoT devices introduce significant security challenges, especially in combating malware threats.

- Most IoT devices use the Android operating system, which is open-sourced and, unlike iOS, is more vulnerable to exposure.
- IoT devices possess considerably less computing power than x86-architecture PCs, making them highly vulnerable to malware due to their limited resources.
- In machine and deep learning, larger datasets facilitate faster model learning and improvement. However, there is a significant shortage of valid datasets of IoT malware.

Cloud platform: The dynamic and distributed nature of cloud environments poses distinct challenges to the malware landscape.

 doi:10.20944/preprints202412.0348.v1

- The interconnected nature of the cloud infrastructure increases the impact of malware.
- A shared responsibility model for cloud security can obscure responsible security tasks. This can lead to organizations having limited visibility and control, thus hindering threat detection and response.
- Attackers can leverage the automation and scalability features of the cloud to quickly launch large-scale attacks.

Cross-platform issues-

- *Data heterogeneity:* Variations in file formats, system call sequences, and behavioral patterns across platforms make it challenging to create generalized models.
- *Lack of unified datasets*: The absence of a standardized, diverse, and large-scale dataset that incorporates samples from Windows, macOS, Linux, Android, IoT, and cloud environments.
- *Inconsistent feature representations:* Differences in how features like static metadata, dynamic behavior, and memory traces are extracted and represented across platforms.
- *Transferability of models:* ML models trained on one platform (e.g., Windows) may not generalize well to others (e.g., Linux or IoT) because of differences in malware characteristics.
- *Performance scalability:* Ensuring scalability and efficiency of detection techniques when applied to cloud and IoT systems with resource limitations.

These challenges emphasize the need for a multi-platform approach to malware detection that considers platform-specific constraints while addressing overarching cross-platform issues.

8. Limitations in the Existing Literature and Future Research Directions

Based on a comprehensive review, we identified some significant research gaps or limitations and list some significant future works to enhance malware detection using machine learning techniques

Lack of Unified Cross-Platform Detection Frameworks

Current ML models are typically designed for a single platform, limiting their ability to detect malware across interconnected systems like PCs, mobile devices, IoT, and cloud environments. To mitigate this issue, we can develop adaptable unified ML frameworks that may use transfer learning and cross-platform training strategies to allow models to generalize effectively across platforms. This approach enables a single model to adapt to different environments, reducing the need for platform-specific datasets.

Insufficient Model Adaptability to Emerging Malware Variants

Malware continually evolves, with new variants employing polymorphic and metamorphic techniques to evade detection, rendering static or narrowly trained ML models ineffective. Transfer learning and continual learning strategies can be used to train models incrementally using new data from different platforms and emerging malware types. This can help maintain model accuracy without retraining from scratch, thereby addressing the limitations of static ML approaches in dynamic threat landscapes.

High Computational Demands of ML Models in Resource-Constrained Environments

The IoT and mobile devices typically have limited processing power and memory, making it challenging to implement advanced ML-based detection techniques that require substantial computational resources. We need to develop lightweight, energy-efficient ML algorithms that are explicitly optimized for IoT and mobile environments. Techniques such as model pruning, quantization, and distillation can reduce the computational load, enabling robust malware detection, even in devices with limited resources.

Limited Transparency and Interpretability of ML-Driven Detection Systems

 doi:10.20944/preprints202412.0348.v1

The opacity of complex ML models' intense learning methods hinders their adoption in security-critical applications, where interpretability is essential for understanding detection decisions and incident responses. Incorporate explainable AI (XAI) techniques are used in malware detection models, making it easier for cybersecurity professionals to interpret ML-driven decisions. XAI methods, such as Shapley Additive exPlanations) or Local Interpretable Model-agnostic Explanations (LIME), could provide insights into model predictions, enhancing trust and transparency.

Lack of Comprehensive, Labelled Datasets for Multi-Platform Malware Detection

Effective multi-platform malware detection models require large, labelled datasets that encompass diverse attack patterns and behaviors across PCs, mobile devices, IoT, and cloud platforms, which are currently scarce. To mitigate this issue, foster collaborative data-sharing frameworks among academic and industry researchers can be built to develop large and diverse datasets that include benign and malicious samples from various platforms. Moreover, synthetic data generation techniques such as Generative Adversarial Networks (GANs) can also be employed to augment existing datasets and enhance model robustness and performance.

Vulnerability to Adversarial Attacks

Adversarial examples, where attackers subtly alter input data to deceive ML models, remain a challenge for malware detection systems, particularly in high-stakes environments, such as IoT and cloud systems. Adversarial training techniques can be integrated to improve model robustness against adversarial samples. Additionally, anomaly detection methods can help identify abnormal patterns that adversarial examples might exhibit, thereby strengthening defense mechanisms.

Limited Research on Hybrid Detection Approaches Combining Static, Dynamic, and Memory Analysis

Most malware detection techniques rely on either static or dynamic analysis, omitting the combined potential of hybrid models that use static, dynamic, and memory features for enhanced detection. To approach this issue, hybrid models can be implemented that integrate static, dynamic, and memory-based analysis techniques, thereby creating a more comprehensive view of malware behaviors across platforms. This approach can improve the detection accuracy and adaptability, especially against complex and evasive malware that may bypass single-method detection.

The proposed solutions offer a roadmap for future research aimed at enhancing the resilience, adaptability, and effectiveness of ML-based malware detection across interconnected digital environments.

9. Conclusion

This comprehensive review examined machine learning (ML)-based malware detection techniques across diverse platforms, including PCs, mobile devices, IoT systems, and cloud environments, each presenting unique security challenges owing to distinct vulnerabilities, operational constraints, and resource limitations. ML techniques utilizing static, dynamic, memory, and hybrid features show considerable promise for identifying malware effectively; however, they face challenges in handling adaptive and polymorphic malware that exploit platform-specific weaknesses. Furthermore, a significant gap persists in cross-platform detection capabilities because most ML models are optimized for specific platforms and lack adaptability to other environments without retraining or additional data. To address these limitations, future research should prioritize adaptable frameworks that leverage transfer and federated learning, enhance model resilience across platforms, and reduce the need for platform-specific labelled datasets. Additionally, advancements in explainable AI (XAI) are critical for improving transparency in ML-driven detection systems, particularly in complex threat scenarios. Lightweight models tailored to resource constrained IoT and edge devices are also essential for effective deployment across increasingly interconnected ecosystems. Ultimately, this survey underscores the necessity of developing robust, unified ML-

 doi:10.20944/preprints202412.0348.v1

based malware detection systems capable of defending against sophisticated, multi-platform threats, thus advancing cybersecurity resilience across digital environments.

References

[1] M. H. Nguyen, D. Le Nguyen, X. M. Nguyen, and T. T. Quan, "Auto-detection of sophisticated malware using lazy-binding control flow graph and deep learning," Comput. Secur., vol. 76, pp. 128–155, 2018, doi: 10.1016/j.cose.2018.02.006.

[2] O. Companies, "2024 Cisco Cybersecurity Readiness Index," 2024. [Online]. Available: https://newsroom.cisco.com/c/dam/r/newsroom/en/us/interactive/cybersecurity-readiness-index/documents/Cisco_Cybersecurity_Readiness_Index_FINAL.pdf.

[3] N. J. Palatty, "Top Malware Attack Statistics, astra 2024. https://www.getastra.com/blog/security-audit/malware-statistics/ (accessed October 15, 2024).

[4] Forbes, "Why Ransomware Should Be On Every Cybersecurity Team's Radar," 2022. https://www.forbes.com/councils/forbestechcouncil/2022/04/12/why-ransomware-should-be-on-every-cybersecurity-teams-radar/#:~:text=According to Cybersecurity Ventures%2C victims,business up and running again. (accessed October 15, 2024).

[5] B. Toulas, "Linux malware sees 35% growth during 2021," 2022. https://www.bleepingcomputer.com/news/security/linux-malware-sees-35-percent-growth-during-2021/ (accessed October 19, 2024).

[6] V. GANDH, "2023 ThreatLabz Report Indicates 400% Growth in IoT Malware Attacks," 2023. https://www.zscaler.com/blogs/security-research/2023-threatlabz-report-indicates-400-growth-iot-malware-attacks (accessed October 15, 2024).

[7] J. Singh and J. Singh, "A survey on machine learning-based malware detection in executable files," J. Syst. Archit., vol. 112, no. March 2020, p. 101861, 2021, doi: 10.1016/j.sysarc.2020.101861.

[8] S. Sibi Chakkaravarthy, D. Sangeetha, and V. Vaidehi, "A Survey on malware analysis and mitigation techniques," Comput. Sci. Rev., vol. 32, pp. 1–23, 2019, doi: 10.1016/j.cosrev.2019.01.002.

[9] U.-H. Tayyab, F. B. Khan and M. H. Durad, A. Khan, and Y. S. Lee, "A Survey of the Recent Trends in Deep Learning Based Malware Detection," J. Cybersecurity Priv., vol. 2, no. 4, pp. 800–829, 2022, doi: 10.3390/jcp2040041.

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

[11] D. Gibert, C. Mateu, and J. Planes, "The rise of machine learning for detection and classification of malware: Research developments, trends and challenges," J. Netw. Comput. Appl., vol. 153, p. 102526, March 2020, doi: 10.1016/J.JNCA.2019.102526.

[12] Y. Liu, C. Tantithamthavorn, L.; Li, Y. Liu, "Deep Learning for Android Malware Defenses: A Systematic Literature Review," ACM Comput. Surv., vol. 55, no. 8, pp. 1–36, 2023, doi: 10.1145/3544968.

[13] Z. Wang, Q. Liu, and Y. Chi, "Review of Android malware detection based on deep learning," IEEE Access, vol. 8, pp. 181102–181126, 2020, doi: 10.1109/ACCESS.2020.3028370.

[14] P. Victor, A. Habibi, L. Rongxing, L. Tinshu, S. Pulei, and X. Shahrear, IoT malware : An attribute - based taxonomy , detection mechanisms and challenges. Springer US, 2023.

[15] C. Alex, G. Creado, W. Almobaideen, O. A. Alghanam, and M. Saadeh, "A Comprehensive Survey for IoT Security Datasets Taxonomy , Classification and Machine Learning Mechanisms," Comput. Secur., p. 103283, 2023, doi: 10.1016/j.cose.2023.103283.

[16] A. Gaurav, B. B., Gupta, and P. K. Panigrahi, "A comprehensive survey on machine learning approaches for malware detection in IoT-based enterprise information system," Enterp. Inf. Syst., vol. 17, no. 3, 2023, doi: 10.1080/17517575.2021.2023764.

[17] P. Maniriho, A. N. Mahmood, and M. J. M. Chowdhury, "A Survey of Recent Advances in Deep Learning Models for Detecting Malware in Desktop and Mobile Platforms," ACM Comput. Surv., vol. 56, no. 6, 2024, doi: 10.1145/3638240.

[18] S. Abijah Roseline and S. Geetha, "A comprehensive survey of tools and techniques mitigating computer and mobile malware attacks," Comput. Electr. Eng., vol. 92, no. October 2020, p. 107143, 2021, doi: 10.1016/j.compeleceng.2021.107143.

[19] M. M. Belal and D. M. Sundaram, "Comprehensive review on intelligent security defences in cloud: Taxonomy, security issues, ML/DL techniques, challenges and future trends," J. King Saud Univ. - Comput. Inf. Sci., vol. 34, no. 10, pp. 9102–9131, 2022, doi: 10.1016/j.jksuci.2022.08.035.

[20] O. Aslan, M. Ozkan-Okay, and D. Gupta, "Intelligent Behavior-Based Malware Detection System on Cloud Computing Environment," IEEE Access, vol. 9, pp. 83252–83271, 2021, doi: 10.1109/ACCESS.2021.3087316.

[21] M. Gopinath and S. C. Sethuraman, "A comprehensive survey on deep learning-based malware detection techniques," Comput. Sci. Rev., vol. 47, p. 100529, 2023, doi: 10.1016/j.cosrev.2022.100529.

[22] O. Sanda, M. and Pavlidis, N. Polatidis, "A deep learning approach for host-based cryptojacking malware detection," Evol. Syst., vol. 15, no. 1, pp. 41–56, 2024, doi: 10.1007/s12530-023-09534-9.
[23] J. Ferdous et al., "Malware–Resistant Data Protection in Hyper−connected Networks : A Survey," https://arxiv.org/pdf/2307.13164, 2023, [Online]. Available: https://arxiv.org/abs/2307.13164.
[24] A. Mitchell, "Current Malware Trends: 5 Most Common Types of Malware in 2024," lumifi, 2024. https://www.lumificyber.com/blog/current-malware-trends-5-most-common-types-of-malware-in-2024/ (accessed October 23, 2024).
[25] J. Ferdous, R. Islam, A. Mahboubi, and M. Z. Islam, "A Review of State-of-the-Art Malware Attack Trends and Defense Mechanisms," IEEE Access, vol. 11, no. October, pp. 121118–121141, 2023, doi: 10.1109/access.2023.3328351.
[26] M. BURGESS, "Conti's Attack Against Costa Rica Sparks a New Ransomware Era," WIRED, 2022. https://www.wired.com/story/costa-rica-ransomware-conti/ (accessed October 23, 2024).
[27] B. Toulas, "REvil ransomware member extradited to the U.S. to stand trial for Kaseya attack," BLEEPING COMPUTER, 2022. https://www.bleepingcomputer.com/news/security/revil-ransomware-member-extradited-to-us-to-stand-trial-for-kaseya-attack/ (accessed October 23, 2024).
[28] M. S. and N. Perlroth, "DarkSide, Blamed for gas pipeline attack, Says It Is Shutting Down. New York Times 2021. https://www.nytimes.com/2021/05/14/business/darkside-pipeline-hack.html (accessed October 23, 2024).
[29] S. Gatlan, "Accenture confirms data breach after August ransomware attack," BLEEPING COMPUTER, 2021. https://www.bleepingcomputer.com/news/security/accenture-confirms-data-breach-after-august-ransomware-attack/ (accessed October 23, 2024).
[30] E. Kost, "What is an Advanced Persistent Threat (APT)?" UpGuard, 2024. https://www.upguard.com/blog/what-is-an-advanced-persistent-threat (accessed October 23, 2024).
[31] A. Sharma, B. B. Gupta, A. K. Singh and V. K. Saraswat, "Orchestration of APT malware evasive manoeuvers employed for eluding anti-virus and sandbox defense," Comput. Secur., vol. 115, p. 102627, 2022, doi: 10.1016/j.cose.2022.102627.
[32] Z. Masood, R. Samar, and M. A. Z. Raja, "Design of a mathematical model for the Stuxnet virus in a network of critical control infrastructure," Comput. Secur., vol. 87, p. 101565, 2019, doi: 10.1016/j.cose.2019.07.002.
[33] A. Jain, "Decoding cryptojacking: What is it and how can you protect yourself," Crypto.news, 2024. https://crypto.news/what-is-cryptojacking-how-does-it-work/ (accessed October 24, 2024).
[34] FORTINET, "Cryptojacking (learns how cryptojacking works and gains access to and abuses computer resources).," FORTINET, 2024. https://www.fortinet.com/resources/cyberglossary/cryptojacking#:~:text=Cryptojacking is also referred to,overall health of your network. (Accessed October 24, 2024).
[35] R. Stevens, "Crypto mining botnet found on Defense Department web server," 2020. https://decrypt.co/18738/crypto-mining-botnet-found-on-defense-department-web-server (accessed October 24, 2024).
[36] R. Stevens, "Man fined $7,000 for using Russian supercomputer to mine Bitcoin," Decrypt, 2019. https://decrypt.co/9751/man-fined-for-using-russian-supercomputer-to-mine-crypto (accessed October 24, 2024).
[37] A. Wolf, "13 Types of Malware Attacks — and How You Can Defend Against Them," 2024. https://arcticwolf.com/resources/blog/8-types-of-malware/ (accessed October 24, 2024).
[38] K. Baker, "The 12 Most Common Types of Malwares," CROWDSTRIKE, 2023. https://www.crowdstrike.com/en-us/cybersecurity-101/malware/types-of-malware/ (accessed October 24, 2024).
[39] P. Maniriho, A. N. Mahmood, and M. J. M. Chowdhury, "A study on malicious software behaviour analysis and detection techniques: Taxonomy, current trends and challenges," Futur. Gener. Comput. Syst., vol. 130, pp. 1–18, 2022, doi: 10.1016/j.future.2021.11.030.
[40] J. Ferdous, R. Islam, A. Mahboubi, and M. Z. Islam, "AI-based Ransomware Detection: A Comprehensive Review," IEEE Access, vol. 12, no. September 2024, doi: 10.1109/ACCESS.2024.3461965.
[41] I. Kara, "Fileless malware threats: Recent advances, analysis approach through memory forensics and research challenges," Expert Syst. Appl., vol. 214, no. April 2022, p. 119133, 2023, doi: 10.1016/j.eswa.2022.119133.
[42] R. Kumar, X. Zhang, W. Wang, R. U. Khan, J. Kumar, and A. Sharif, "A Multi-modal Malware Detection Technique for Android IoT Devices Using Various Features," IEEE Access, vol. 7, pp. 64411–64430, 2019, doi: 10.1109/ACCESS.2019.2916886.
[43] T. Panker and N. Nissim, "Leveraging malicious behavior traces from volatile memory using machine learning methods for trusted unknown malware detection in Linux cloud environments," Knowledge-Based Syst., vol. 226, August 2021, doi: 10.1016/j.knosys.2021.107095.
[44] H. Oz, A. Aris, A. Levi, and A. S. Uluagac, "A Survey on Ransomware: Evolution, Taxonomy, and Defense Solutions," ACM Comput. Surv., vol. 1, no. 1, 2022, doi: 10.1145/3514229.

[45] T. A. Unit, "VMware Threat Report – Exposing Malware in Linux-Based Multi-Cloud Environments," 2022. https://blogs.vmware.com/security/2022/02/2022-vmware-threat-report-exposing-malware-in-linux-based-multi-cloud-environments.html (accessed November 04, 2024).
[46] R. Walsh, "Linux Malware Stats and Facts for 2024," 2024. https://www.comparitech.com/blog/vpn-privacy/linux-malware-stats-and-facts/ (accessed November 04, 2024).
[47] Y. E. T. M. Meshi, "Battling macOS Malware with Cortex AI," 2023. https://www.paloaltonetworks.com/blog/security-operations/battling-macos-malware-with-cortex-ai/ (accessed November 04, 2024).
[48] B. Report, "macOS Threat Landscape Report," 2023.
[49] Ani Petrosyan, "Number of detected malicious installation packages on mobile devices worldwide from 4th quarter 2015 to 3rd quarter 2023," Statista, 2024. https://www.statista.com/statistics/653680/volume-of-detected-mobile-malware-packages/ (accessed October 27, 2024).
[50] A. Sherif, "Market share of mobile operating systems worldwide from 2009 to 2024, by quarter," Statista, 2024. https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/ (accessed November 01, 2024).
[51] H. Haidros Rahima Manzil and S. Manohar Naik, "Detection approaches for android malware: Taxonomy and review analysis," Expert Syst. Appl., vol. 238, no. PF, p. 122255, 2024, doi: 10.1016/j.eswa.2023.122255.
[52] A. Saracino, D. Sgandurra, G. Dini, and F. Martinelli, "MADAM: Effective and Efficient Behavior-based Android Malware Detection and Prevention," IEEE Trans. Dependable Secur. Comput., vol. 15, no. 1, pp. 83–97, 2018, doi: 10.1109/TDSC.2016.2536605.
[53] S. Garg and N. Baliyan, "Comparative analysis of Android and iOS from a security viewpoint," Comput. Sci. Rev., vol. 40, p. 100372, 2021, doi: 10.1016/j.cosrev.2021.100372.
[54] Y. Shen and H. Wuhan, "Enhancing data security of iOS client by encryption algorithm," IEEE 2nd Adv. Inf. Technol. Electron. Autom. Control Conf., pp. 366–370, 2017.
[55] M. Lutaaya, "Rethinking app permissions on iOS," Conf. Hum. Factors Comput. Syst. - Proc., vol. 2018-April, pp. 1–6, 2018, doi: 10.1145/3170427.3180284.
[56] J. Phungglan, "Most common viruses on iPhone," 2023. https://macpaw.com/how-to/most-common-iphone-viruses (accessed November 03, 2024).
[57] R. Walsh, "iOS Malware Stats and Facts for 2024," 2024. https://www.comparitech.com/blog/vpn-privacy/ios-malware-stats-and-facts/ (accessed November 03, 2024).
[58] K. O'Flaherty, "New 'Dangerous' iPhone Spyware Attack Warning Issued To iOS Users," 2024. https://www.forbes.com/sites/kateoflahertyuk/2024/04/19/new-dangerous-iphone-spyware-attack-warning-issued-to-ios-users/ (accessed November 03, 2024).
[59] L. S. Vailshery, "Number of Internet of Things (IoT) connections worldwide from 2022 to 2023, with forecasts from 2024 to 2033," Statista, 2024. https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/ (accessed November 05, 2024).
[60] C. San Jose, "Zscaler ThreatLabz Finds a 400% Increase in IoT and OT Malware Attacks Year-over-Year, Underscoring Need for Better Zero Trust Security to Protect Critical Infrastructures," Zscaler, 2023. https://www.zscaler.com/press/zscaler-threatlabz-finds-400-increase-iot-and-ot-malware-attacks-year-over-year-underscoring (accessed November 05, 2024).
[61] R. M. Yadav, "Effective analysis of malware detection in cloud computing," Comput. Secur., vol. 83, pp. 14–21, 2019, doi: 10.1016/j.cose.2018.12.005.
[62] F. Kilonzi, "Cloud Malware: Types of Attacks and How to Defend Against Them," 2023. https://thenewstack.io/cloud-malware-types-of-attacks-and-how-to-defend-against-them/ (accessed November 25, 2024).
[63] S. Jeon and J. Moon, "Malware-Detection Method with a Convolutional Recurrent Neural Network Using Opcode Sequences," Inf. Sci. (Ny)., vol. 535, pp. 1–15, 2020, doi: 10.1016/j.ins.2020.05.026.
[64] S. Huda, R. Islam, J. Abawajy, J. Yearwood, M. M. Hassan, and G. Fortino, "A hybrid-multi filter-wrapper framework to identify run-time behaviour for fast malware detection," Futur. Gener. Comput. Syst., vol. 83, pp. 193–207, June 2018, doi: 10.1016/j.future.2017.12.037.
[65] M. K. Alzaylaee, S. Y. Yerima, S. Sezer, "DL-Droid : Deep learning based android malware detection using real devices," vol. 89, 2020, doi: 10.1016/j.cose.2019.101663.
[66] N. A. Stoian, "Machine Learning for Anomaly Detection in IoT networks: Malware analysis on the IoT-23 Data set," Univ. Twente, 2020.
[67] M. He, Y. Huang, X. Wang, P. Wei, and X. Wang, "A Lightweight and Efficient IoT Intrusion Detection Method Based on Feature Grouping," IEEE Internet Things J., vol. 11, no. 2, pp. 2935–2949, 2024, doi: 10.1109/JIOT.2023.3294259.
[68] A. Mezina and R. Burget, "Obfuscated malware detection using dilated convolutional network," Int. Congr. Ultra Mod. Telecommun. Control Syst. Work., vol. 2022-Octob, pp. 110–115, 2022, doi: 10.1109/ICUMT57764.2022.9943443.

[69] J. Mitchell, N. McLaughlin, and J. Martinez-del-Rincon, "Generating sparse explanations for malicious Android opcode sequences using hierarchical LIME," Comput. Secur., vol. 137, no. July 2023, p. 103637, 2024, doi: 10.1016/j.cose.2023.103637.
[70] N. Potha, V. Kouliaridis, G. Kambourakis, "An extrinsic random-based ensemble approach for android malware detection," Conn. Sci., vol. 33, no. 4, pp. 1077–1093, 2021, doi: 10.1080/09540091.2020.1853056.
[71] S. Yoo, S. Kim, S. Kim, and B. B. Kang, "AI-HydRa: Advanced hybrid approach using random forest and deep learning for malware classification," Inf. Sci. (Ny)., vol. 546, pp. 420–435, 2021, doi: 10.1016/j.ins.2020.08.082.
[72] "No Title."
[73] E. Snow, M. Alam, A. Glandon, and K. Iftekharuddin, "End-to-end Multimodel Deep Learning for Malware Classification," Proc. Int. Jt. Conf. Neural Networks, 2020, doi: 10.1109/IJCNN48605.2020.9207120.
[74] A. Darem, J. Abawajy, A. Makkar, A. Alhashmi, and S. Alanazi, "Visualization and deep-learning-based malware variant detection using OpCode-level features," Futur. Gener. Comput. Syst., vol. 125, pp. 314–323, 2021, doi: 10.1016/j.future.2021.06.032.
[75] F. O. Catak, J. Ahmed, K. Sahinbas, and Z. H. Khand, "Data Augmentation based Malware Detection Using Convolutional Neural Networks," PeerJ Comput. Sci., vol. 7, pp. 1–26, 2021, doi: 10.7717/PEERJ-CS.346.
[76] C. C. Moreira, D. C. Moreira, C., de S. d. Sales, "Improving ransomware detection based on portable executable header using Xception convolutional neural network," Comput. Secur., vol. 130, p. 103265, 2023, doi: 10.1016/j.cose.2023.103265.
[77] C. Jindal, C. Salls, H. Aghakhani, K. Long, C. Kruegel, and G. Vigna, "Neurlux: Dynamic malware analysis without feature engineering," ACM Int. Conf. Proceeding Ser., pp. 444–455, 2019, doi: 10.1145/3359789.3359835.
[78] R. Chaganti, V. Ravi, T. D. Pham, "A multi-view feature fusion approach for effective malware classification using Deep Learning," J. Inf. Secur. Appl., vol. 72, no. December 2022, p. 103402, 2023, doi: 10.1016/j.jisa.2022.103402.
[79] H. Darabian et al., "Detecting Cryptomining Malware: a Deep Learning Approach for Static and Dynamic Analysis," J. Grid Comput., vol. 18, no. 2, pp. 293–303, 2020, doi: 10.1007/s10723-020-09510-6.
[80] M. R. Naeem et al., "A Malware Detection Scheme via Smart Memory Forensics for Windows Devices," Mob. Inf. Syst., vol. 2022, 2022, doi: 10.1155/2022/9156514.
[81] A. Tekerek and M. M. Yapici, "A novel malware classification and augmentation model based on convolutional neural network," Comput. Secur., vol. 112, p. 102515, 2022, doi: 10.1016/j.cose.2021.102515.
[82] H. Naeem, S. Dong, O. J. Falana, and F. Ullah, "Development of a deep stacked ensemble with process based volatile memory forensics for platform independent malware detection and classification," Expert Syst. Appl., vol. 223, no. February, p. 119952, 2023, doi: 10.1016/j.eswa.2023.119952.
[83] T. Landman and N. Nissim, "Deep-Hook: A trusted deep learning-based framework for unknown malware detection and classification in Linux cloud environments," Neural Networks, vol. 144, pp. 648–685, 2021, doi: 10.1016/j.neunet.2021.09.019.
[84] A. Pektaş and T. Acarman, "Learning to detect Android malware via opcode sequences," Neurocomputing, vol. 396, pp. 599–608, 2020, doi: 10.1016/j.neucom.2018.09.102.
[85] M. Aamir et al., "AMDDLmodel : Android smartphones malware detection using deep learning model," pp. 1–16, 2024, doi: 10.1371/journal.pone.0296722.
[86] H. Naeem, S. Dong, O. J. Falana, and F. Ullah, "Development of a deep stacked ensemble with process based volatile memory forensics for platform independent malware detection and classification," Expert Syst. Appl., vol. 223, no. October 2022, p. 119952, 2023, doi: 10.1016/j.eswa.2023.119952.
[87] K. L. K. Sudheera, D. M. Divakaran, R. P. Singh, and M. Gurusamy, "ADEPT: Detection and Identification of Correlated Attack Stages in IoT Networks," IEEE Internet Things J., vol. 8, no. 8, pp. 6591–6607, 2021, doi: 10.1109/JIOT.2021.3055937.
[88] D. Vasan, M. Alazab, S. Venkatraman, J. Akram, and Z. Qin, "MTHAEL: Cross-architecture iot malware detection based on neural network advanced ensemble learning," IEEE Trans. Comput., vol. 69, no. 11, pp. 1654–1667, 2020, doi: 10.1109/TC.2020.3015584.
[89] H. V. Le, Q. D. Ngo, and V. H. Le, "Iot botnet detection using system call graphs and one-class CNN classification," Int. J. Innov. Technol. Explor. Eng., vol. 8, no. 10, pp. 937–942, 2019, doi: 10.35940/ijitee.J9091.0881019.
[90] R. Shire, S. Shiaeles, K. Bendiab, B. Ghita, and N. Kolokotronis, Malware Squid: A Novel IoT Malware Traffic Analysis Framework Using Convolutional Neural Network and Binary Visualisation, vol. 11660 LNCS. Springer International Publishing, 2019.
[91] H. Jiang, J. Lin, and H. Kang, "FGMD: A robust detector against adversarial attacks in the IoT network," Futur. Gener. Comput. Syst., vol. 132, pp. 194–210, 2022, doi: 10.1016/j.future.2022.02.019.
[92] C. Li, Q. Lv, N. Li, Y. Wang, D. Sun, and Y. Qiao, "A novel deep framework for dynamic malware detection based on API sequence intrinsic features," Comput. Secur., vol. 116, 2022, doi: 10.1016/j.cose.2022.102686.

 doi:10.20944/preprints202412.0348.v1

[93] W. R. Aditya, Girinoto, R. B. Hadiprakoso, and A. Waluyo, "Deep Learning for Malware Classification Platform using Windows API Call Sequence," Proc. - 3rd Int. Conf. Informatics; Multimedia; Cyber; Inf. Syst. ICIMCIS 2021, pp. 25–29, 2021, doi: 10.1109/ICIMCIS53775.2021.9699248.

[94] M. Ring, D. Schlör, S. Wunderlich, D. Landes, and A. Hotho, "Malware detection on windows audit logs using LSTMs," Comput. Secur., vol. 109, p. 102389, 2021, doi: 10.1016/j.cose.2021.102389.

[95] M. S. A Lakshmanarao, "Android Malware Detection with Deep Learning using RNN from Opcode Sequences.," Int. J. Interact. Mob. Technol., vol. 16, 2022.

[96] H. Ge, Z. Wang, Y. Liu, and X. Liu, "D ROIDETEC: Android Malware Detection and Malicious Code," pp. 1–13.

[97] S. K. Sasidharan and C. Thomas, "MemDroid - LSTM based Malware Detection Framework for Android Devices," 2021 IEEE Pune Sect. Int. Conf., pp. 1–6, 2021, doi: 10.1109/PuneCon52575.2021.9686531.

[98] E. Amer and S. El-sappagh, "Robust deep learning early alarm prediction model based on the behavioral smell for Android malware," Comput. Secur., vol. 116, p. 102670, 2022, doi: 10.1016/j.cose.2022.102670.

[99] Y. Wu, J. Shi, P. Wang, D. Zeng, and C. Sun, "Android malware detection," No. January 2022, pp. 118–130, 2023, doi: 10.1049/ise2.12082.

[100] J. Jeon, B. Jeong, S. Baek and Y. S. Jeong, "Hybrid Malware Detection Based on Bi-LSTM and SPP-Net for Smart IoT," IEEE Trans. Ind. Informatics, vol. 18, no. 7, pp. 4830–4837, 2022, doi: 10.1109/TII.2021.3119778.

[101] S. Mahdavifar, D. Alhadidi, and A. A. Ghorbani, Effective and Efficient Hybrid Android Malware Classification Using Pseudo - Label Stacked Auto - Encoder, vol. 30, no. 1. Springer US, 2022.

[102] J. Hemalatha, S. A. Roseline, S. Geetha, S. Kadry, and R. Damaševičius, "An efficient densenet-based deep learning model for malware detection," Entropy, vol. 23, no. 3, pp. 1–23, 2021, doi: 10.3390/e23030344.

[103] S. Kumar and B. Janet, "DTMIC: Deep transfer learning for malware image classification," J. Inf. Secur. Appl., vol. 64, no. December 2021, p. 103063, 2022, doi: 10.1016/j.jisa.2021.103063.

[104] X. Huang, L. Ma, W. Yang, and Y. Zhong, "A Method for Windows Malware Detection Based on Deep Learning," J. Signal Process. Syst., vol. 93, no. 2–3, pp. 265–273, 2021, doi: 10.1007/s11265-020-01588-1.

[105] P. Xu, Y. Zhang, C. Eckert, and A. Zarras, HawkEye: Cross-Platform Malware Detection with Representation Learning on Graphs, vol. 12893 LNCS. Springer International Publishing, 2021.

[106] S. K. J. Rizvi, W. Aslam, M. Shahzad, S. Saleem, and M. M. Fraz, "PROUD-MAL: static analysis-based progressive framework for deep unsupervised malware classification of windows portable executable," Complex Intell. Syst., vol. 8, no. 1, pp. 673–685, 2022, doi: 10.1007/s40747-021-00560-1.

[107] M. Khan, D. Baig, U. S. Khan, and A. Karim, "Malware Classification Framework using Convolutional Neural Network," 1st Annu. Int. Conf. Cyber Warf. Secur. ICCWS 2020 - Proc., 2020, doi: 10.1109/ICCWS48432.2020.9292384.

[108] E. Amer and I. Zelinka, "A dynamic Windows malware detection and prediction method based on contextual understanding of API call sequence," Comput. Secur., vol. 92, p. 101760, 2020, doi: 10.1016/j.cose.2020.101760.

[109] F. O. Catak, A. F. Yazi, O. Elezaj, and J. Ahmed, "Deep learning based Sequential model for malware analysis using Windows exe API Calls," PeerJ Comput. Sci., vol. 6, pp. 1–23, 2020, doi: 10.7717/PEERJ-CS.285.

[110] M. M. Hasan and M. M. Rahman, "RansHunt: A support vector machines based ransomware analysis framework with integrated feature set," 20th Int. Conf. Comput. Inf. Technol. ICCIT 2017, vol. 2018-Janua, pp. 1–7, 2018, doi: 10.1109/ICCITECHN.2017.8281835.

[111] E. M. B. Karbab, M. Debbabi, and A. Derhab, "SwiftR: Cross-platform ransomware fingerprinting using hierarchical neural networks on hybrid features," Expert Syst. Appl., vol. 225, no. March, p. 120017, 2023, doi: 10.1016/j.eswa.2023.120017.

[112] C. Hwang, J. Hwang, J. Kwak, and T. Lee, "Platform-independent malware analysis applicable to windows and linux environments," Electron., vol. 9, no. 5, 2020, doi: 10.3390/electronics9050793.

[113] T. Set and T. Set, "Mac Malware Detection via Static File Structure Analysis," pp. 1–5.

[114] H. H. Pajouh, A. and Dehghantanha, R. Khayami, and K. K. R. Choo, "Intelligent OS X malware threat detection with code inspection," J. Comput. Virol. Hacking Tech., vol. 14, no. 3, pp. 213–223, 2018, doi: 10.1007/s11416-017-0307-5.

[115] H. Gao, S. Cheng, and W. Zhang, "GDroid : Android malware detection and classification with graph convolutional network," Comput. Secur., vol. 106, p. 102264, 2021, doi: 10.1016/j.cose.2021.102264.

[116] S. Wang, Z. Chen, Q. Yan, B. Yang, L. Peng, and Z. Jia, "A mobile malware detection method using behavior features in network traffic," J. Netw. Comput. Appl., vol. 133, no. December 2018, pp. 15–25, 2019, doi: 10.1016/j.jnca.2018.12.014.

[117] A. Cimitile, F. Martinelli, and F. Mercaldo, "Machine learning meets ios malware: Identifying malicious applications on apple environment," ICISSP 2017 - Proc. 3rd Int. Conf. Inf. Syst. Secur. Priv., vol. 2017-Janua, no. Icissp, pp. 487–492, 2017, doi: 10.5220/0006217304870492.

[118] G. Zhou, M. Duan, Q. Xi, and H. Wu, "ChanDet: Detection Model for Potential Channel of iOS Applications," J. Phys. Conf. Ser., vol. 1187, no. 4, 2019, doi: 10.1088/1742-6596/1187/4/042045.

 doi:10.20944/preprints202412.0348.v1

[119] F. Mercaldo and A. Santone, "Deep learning for image-based mobile malware detection," J. Comput. Virol. Hacking Tech., vol. 16, no. 2, pp. 157–171, 2020, doi: 10.1007/s11416-019-00346-7.
[120] H. V. Le and Q. D. Ngo, "V-Sandbox for Dynamic Analysis IoT Botnet," IEEE Access, vol. 8, pp. 145768–145786, 2020, doi: 10.1109/ACCESS.2020.3014891.
[121] X. Zhou, W. Liang, W. Li, K. Yan, S. Shimizu, and K. I. K. Wang, "Hierarchical Adversarial Attacks Against Graph-Neural-Network-Based IoT Network Intrusion Detection System," IEEE Internet Things J., vol. 9, no. 12, pp. 9310–9319, 2022, doi: 10.1109/JIOT.2021.3130434.
[122] A. Guerra-Manzanares, J. Medina-Galindo, H. Bahsi, and S. Nomm, "MedBIoT: Generation of an IoT Botnet Dataset in a Medium-sized IoT Network," Int. Conf. Inf. Syst. Secur. Priv., no. Icissp 2020, pp. 207–218, 2020, doi: 10.5220/0009187802070218.
[123] L. Xiao, Y. Li, X. Huang, and X. Du, "Cloud-based malware detection game for mobile devices with offloading," IEEE Trans. Mob. Comput., vol. 16, no. 10, pp. 2742–2750, 2017, doi: 10.1109/TMC.2017.2687918.