electronics

Article

Malware Analysis in IoT & Android Systems with Defensive Mechanism

Chandra Shekhar Yadav [1,*], Jagendra Singh [2], Aruna Yadav [2], Himansu Sekhar Pattanayak [2], Ravindra Kumar [3], Arfat Ahmad Khan [4], Mohd Anul Haq [5,*], Ahmed Alhussen [6,*] and Sultan Alharby [5]

1 Standardisation Testing and Quality Certification, MeitY, Delhi 110003, India
2 School of Computer Sciences Engineering and Technology, Bennett University, Greater Noida 201310, India; jagendrasngh@gmail.com (J.S.); aruna.yadav21@gmail.com (A.Y.); himansusekharpattanayak@gmail.com (H.S.P.)
3 CSE Department, Galgotias College of Engineering and Technology, Greater Noida 201310, India; ravindrakumarchauhan@gmail.com
4 College of Computing, Khon Kaen University, Khon Kaen 40000, Thailand; arfatkhan@kku.ac.th
5 Department of Computer Science, College of Computer and Information Sciences, Majmaah University, Al-Majmaah 11952, Saudi Arabia; sa.alharby@mu.edu.sa
6 Department of Computer Engineering, College of Computer Science and Information Sciences, Majmaah University, Al-Majmaah 11952, Saudi Arabia
* Correspondence: chandrtech15@gmail.com (C.S.Y.); m.anul@mu.edu.sa (M.A.H.); aa.alhussen@mu.edu.sa (A.A.)

Abstract: The Internet of Things (IoT) and the Android operating system have made cutting-edge technology accessible to the general public. These are affordable, easy-to-use, and open-source technology. Android devices connect to different IoT devices such as IoT-enabled cameras, Alexa powered by Amazon, and various other sensors. Due to the escalated growth of Android devices, users are facing cybercrime through their Android devices. This article aims to provide a comprehensive study of the IoT and Android systems. This article classifies different attacks on IoT and Android devices and mitigation strategies proposed by different researchers. The article emphasizes the role of the developer in secure application design. This article attempts to provide a relative analysis of several malware detection methods in the different environments of attacks. This study expands the awareness of certain application-hardening strategies applicable to IoT devices and Android applications and devices. This study will help domain experts and researchers to gain knowledge of IoT systems and Android systems from a security point of view and provide insight into how to design more efficient, robust, and comprehensive solutions. This article discusses different attack vectors and mitigation strategies available to both developers and in the open domain. Certain guidelines are also suggested for application and platform developers, as well as application databases (Google play store), to limit the risk of attack, and users can form their own defense with knowledge regarding keeping hardware and software updated and securing their system with a strong password.

Keywords: IoT; android system; malware; kernel-based attack; application attack; application hardening technique

check for updates

Citation: Yadav, C.S.; Singh, J.; Yadav, A.; Pattanayak, H.S.; Kumar, R.; Khan, A.A.; Haq, M.A.; Alhussen, A.; Alharby, S. Malware Analysis in IoT & Android Systems with Defensive Mechanism. *Electronics* **2022**, *11*, 2354. https://doi.org/10.3390/electronics11152354

Academic Editor: Maciej Ławryńczuk

Received: 2 June 2022
Accepted: 21 June 2022
Published: 28 July 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Smartphone usage and associated devices and applications are rapidly increasing because of the accessibility and effectiveness of different applications, as well as the growing development of the software and hardware of electronic objects. By 2023, it is expected that 4.3 billion people will own a smartphone. The Android operating system is the most extensively used mobile operating system (OS). Its market share was more than 75 percent in May 2021. Apple iOS has the second greatest market share of 27 percent, with Samsung, KaiOS, and other small suppliers sharing the remaining 0.81 percent. The official application database for Android-based smartphones is Google Play. As of May 2021, it had over 2.9 million applications. AppBrain has classified almost 2.5 million of these apps as standard apps, while 0.4 million have been labeled as low-quality apps. Because of its global prominence, Android is a more appealing target for cybercriminals and is more vulnerable to malware and viruses. Malware is a combination of the terms "malicious" and "software". When software performs a common task without authority, it is referred to as malevolent software. An operation is a malicious operation that occurs without the user's knowledge or that is not planned to occur at that particular time. Malware is made up of two parts: Payload (carrier) and exploits (activity).

Hackers can be classified as Rogue hackers with no prior experience and APT agents (Advanced Persistent Threat) who are highly trained and protected by safe-harbor countries and agencies. Attackers may belong to organizations, such as Julian Paul Assange, an American and former CIA operative, or may be outsiders. Hackers may have a wide range of intentions, and they may be grouped into black-hat hackers and white-hat hackers. Malware can be categorized as spyware, worms, logic bombs, viruses, rootkits, trojan horses, adware, backdoors, ransomware [1], bots, and other types of malware. Following the release of the 'Pegasus' spyware for Android and iOS phones, a debate over "National Security" erupted. These viruses spread as a result of system vulnerabilities, insecure design, the frequent usage of portable devices, homogeneity, and several other factors. An expert hacker and a persistent attack adopt a strategy to stay alive for a long duration while remaining hidden from users. Figure 1 explains how sophisticated malware attackers follow an attack plan and exploitation.

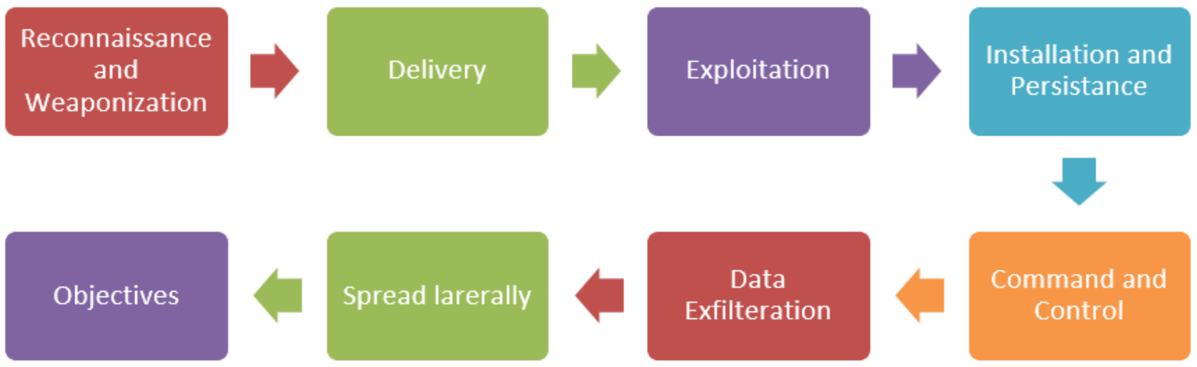

Figure 1. Malware exploitation stage.

The first stage is **Reconnaissance and weaponization** in which the attackers find the weakness and vulnerability of the system. This may belong to the hardware side, namely the CPU, memory, communication device's antenna, and sim card, or the software side, namely zero-day vulnerabilities, continued use of old patches, and a weak encryption algorithm. The next step after finding the target is the preparation of the threat and payload so that **Malware is delivered** to the user. Malware can be delivered either by a direct approach (user responsible), including E-Mail, SMS, or a malicious Link on malicious websites, or an indirect approach (attacker's experience), including an encrypted/encoded link, an active directory attack, privilege escalation, the directory access method, adware, etc. After the malware is transmitted, the following phase is **Exploitation**, or payload, which refers to the activities that malware performs. At this stage, it may participate in an active attack or passive attack. Active attacks are masquerade attacks such as login access or a bypass, session hijacking/reply attack, message modification using a change in header by taking advantage of weak cryptography algorithms, a Denial of Service/Distributed DoS (DoS/DDoS) attack causing flooding, and malformed data. Possible passive attacks are the monitoring of network traffic, scanning of open ports and vulnerabilities, eavesdropping, footprinting, spying, and dumpster driving (information collected from discarded devices and the recycle bin).

After this second phase is started, the attacker decides whether to quit or continue, but in both cases, the threat continues. They install malware and use a backdoor to obtain access to more information, as well as steal credentials that allow them to maintain a long life. They may install new malware, and the next stage is **Command & Control (C&C)** in which a persistent connection is granted to an attacker, and this may infect other connected systems, constituted as **Data Exfiltration** (the violation of data integrity and availability) and **Spread laterally** (mapped drives and connected systems/devices).

The structure of this paper is as follows: Section 2 highlights work related to Malware, the most famous attack on digital systems. Section 3 focuses on the architecture of the Internet of Things (IoT), industrial IoT (IIoT), different attacks, and mitigation proposed by several researchers, i.e., physical, network, data attacks, and software attacks. Section 4 explains details of the Android architecture, vulnerabilities in the Android system, and different malware detection techniques, and Section 5 focuses on application-hardening techniques applicable for any kind of digital system, but especially in the context of IoT and Android. Section 6 concerns the novelty and limitations of the article, and finally, Section 7 presents the conclusion and future work. The taxonomy of this presentation is shown in Figure 2.

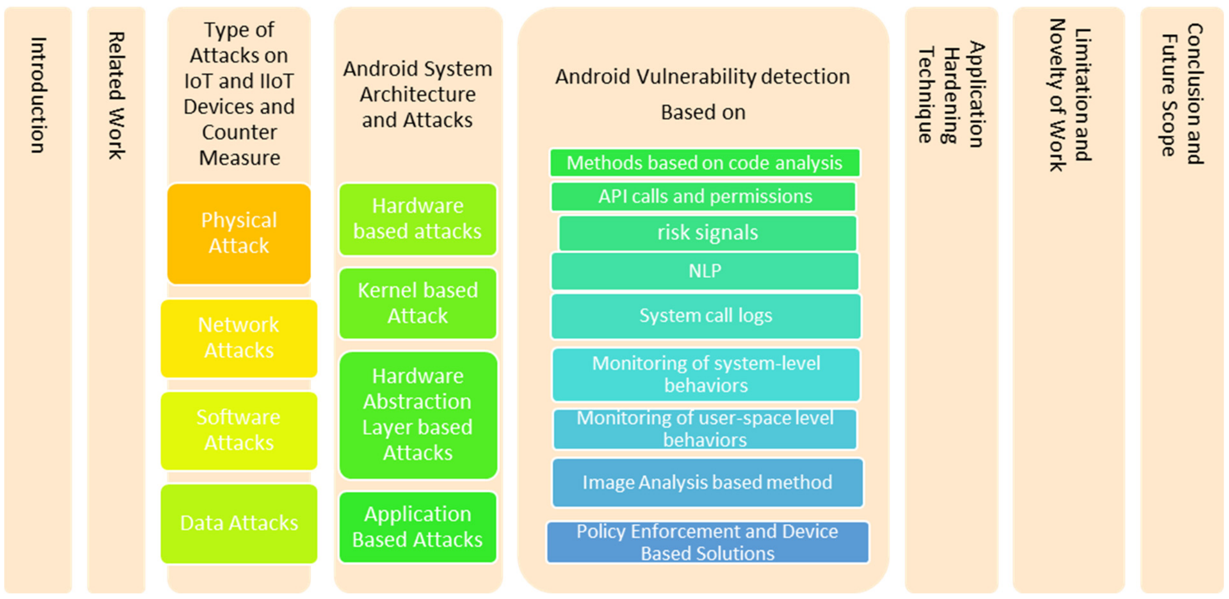
Introduction
Related Work
Type of Attacks on IoT and IIoT Devices and Counter Measure
Physical Attack
Network Attacks
Software Attacks
Data Attacks
Android System Architecture and Attacks
Hardware based attacks
Kernel based Attack
Hardware Abstraction Layer based Attacks
Application Based Attacks
Android Vulnerability detection Based on
Methods based on code analysis
API calls and permissions
risk signals
NLP
System call logs
Monitoring of system-level behaviors
Monitoring of user-space level behaviors
Image Analysis based method
Policy Enforcement and Device Based Solutions
Application Hardening Technique
Limitation and Novelty of Work
Conclusion and Future Scope

Figure 2. Taxonomy of presented article.

2. Related Work

Modern IoT and IIoT devices are controlled and monitored by Android devices. Vulnerabilities may be present either in IoT devices or Android devices and both can hamper the effectiveness of security measures implemented for the system. Therefore, some research focused on countering attacks is presented here. The authors in [2] analyzed home automation systems, i.e., motion sensors, magnetic sensors, and industrial IoT devices, and found that smart meters are vulnerable to several attacks due to the limited protection during their design and implementation. These devices communicate over an RF (Radio Frequency) channel, and they recommend using an encrypted channel as a security measure. The research in [3] addressed the issues of mutual authentication, the physically unclonable function, limited resources, side-channel analysis, and cloning attacks. Security measures for mutual authentication, session key establishment, and protocol verification are presented in their article. Mutual authentication provides good insight into inter-device and session key distribution. The proposed solution mitigates the risk of man-in-the-middle attacks and replay attacks. The author in [4] presented several security attacks and issues, privacy concerns, integration with the Blockchain for IoT devices, and different security research areas. Application repackaging is one of the most vulnerable issues in Android/iOS applications. The authors in [5] presented work on repackaged applications. They addressed five issues, namely (1) the current unfavorable repackaging practices, (2) the way adware is embedded in the code, (3) the types of apps used to repackage, (4) the reasons people download repackaged software, and (5) the way the properties of an app change in the repackaged version. Static Malware detection tools including TinyDroid, DroidFDR [6], DroidEnsemble, and NsDroid were presented in [7], but their shortcoming is that these are not suitable for dynamic analysis. NsDroid is a lightweight and fast Android malware detection tool. NsDroid is based on a local function graph [8], therefore, comparatively, it is 20× times faster than other graph-based approaches [9,10].

In [11], the author cited conflict between different sensors controlled via a smartphone and proposed a solution based on LOD (Linked Open Data). LOD enables better exploitation of the qualities of the inhabitant's profile and services, as well as defining the relationship between the various services and objects in the house. The authors of the article [12] presented good insight into mobile malware detection using signature- and anomaly-based approaches.

3. Type of Attacks on IoT and IIoT Devices

The Internet of Things, in which "things" stands as a collection of diverse controllable, recognizable, and addressable devices on the internet, constitutes connected devices with the primary task of collecting and sharing information with its peers. According to one study, applications of IoT devices are increasing in data day by day, and by 2025, there will be approximately 64 billion IoT devices [11,13,14]. These devices will have some kind of communication and sharing of data, and when this occurs, there is an obvious threat of attacks and malware in these devices. Industrially used versions of IOT may be called IIoT. IIoT is susceptible to various cyber threats, primarily due to the growth in the use of sensors, the traditional and inefficient coordination and communication approach, and the increase in the number of potential hackers. Different types of attacks and security measures are listed below.

3.1. Physical Attack and Countermeasures

In this scenario, the attacker physically accesses the device and network, and harms them in a significant way. Different physical attacks may be carried out by tampering with physical devices, i.e., disconnection of the network cable, replacing evidence instruments, rotating surveillance cameras, MCI (Malicious code Injection), RF Interference/Jamming attacks, DDoS attacks, Fake Node Injection attacks, Sleep Denial attacks, SCA (Side-Channel Attack), and PDoS (Permanent DoS). In this attack, the system becomes a completely nonoperational state.

In [2], the authors examined the security and vulnerability of both industrial and commercial IoT devices. Their experiment and analysis found that home automation systems or commercial IoT systems are more vulnerable to brute force attacks for login credentials. Smart meters have the possibility of ransomware attacks [1]. Devices used for virtual personal assistance, such as Google Home and Amazon Echo, are prone to voice masquerading and squatting attacks. An attacker can use similarly paraphrased names or pronounced names to hijack voice commands, and in this voice, masquerade an attack. The attacker uses skill to impersonate a legitimate voice in order to eavesdrop on user conversations and user data.

Countermeasures against Physical Attacks: Security measures for this issue include the inbuilt variability of the integrated circuit with the physical unclonable function (PUF) for mutual authentication. Mutual authentication takes place via challenge–response mechanics, where the output depends on the device's microstructure. This makes PUF unclonable, tamper-proof from physical attacks, and eliminates malicious code injection. The work conducted by [15] focuses on providing a method of energy-efficient architecture with increased security capabilities. To solve the energy problem, i.e., sleep denial attack, they have proposed a CUTE-mote (Customizable and Trustable End device mote) for heterogeneous architecture. The diverse architecture is developed by the hybridization of

MCU (Micro Controller Unit), Contiki OS, and RCU (reconfigurable computing unit) with an IEEE-802.15.4 radio transceiver. Experiments have shown that by considering the thread metric benchmark and smart fusion of two SOC hardware platforms, the proposed system is capable of preventing cyber-physical attacks such as jamming attacks and provides a strong countermeasure against sleep denial attacks.

To solve the issue of distributed systems, one solution is the PAuthKey (Pervasive Authentication Protocol). In the proposed protocol, the implicit certificate is obtained from the cluster head, and later, a secured link is maintained between sensor nodes, peer nodes, and the end-user. The authentication scheme depends on the respective arrangement of sensor nodes, and the scheme is liable to provide a guarantee of application-layer security. This is capable of impersonating attacks, masquerade attacks, and fake node injection attacks.

Article [16] provides a lightweight encryption algorithm to counter side-channel attacks. The lightweight and software-based solutions have been chosen by the authors due to the power concern of IoT devices, faster communication, and dynamic capability of software algorithms. According to the authors, masking techniques also make it hard to analyze encryption and decryption keys and protect against side-channel attacks. Other security measures for this kind of attack are power analysis, timing analysis, and physical unclonable function.

3.2. Network Attacks

Network attacks are performed by the maneuvering of IoT networks to cause damage. This may be launched from a remote location. Some common network attacks are Traffic Analysis attacks, RFID Spoofing (Radio Frequency Identification), RFID Unauthorized Access, Routing Information Attacks, Selective forwarding, Sinkhole attacks, Wormhole attacks, Sybil attacks, Man-in-the-Middle Attacks (MiTM), and Replay attacks. Network attacks may lead to serious consequences, and ultimately, may face a shutdown of the system and services. The denial of one service, such as OTP in banking services, makes other services inaccessible and useless.

Countermeasures against Network Attacks: The framework is named EPIC (Efficient and Privacy-preserving traffic obfuscation), a multi-hop routing protocol to achieve strong privacy. It protects commercial applications used for home automation or smart homes. Three basic properties, referred to as CIA (Confidentiality, Integrity, Authentication), are to be maintained for secure communication and to defend against Reply attacks. IBC (Identity-Based Cryptography), which is a signcryption technique, efficiently satisfies CIA. The IBC combines encryption and signature schemes and also eliminates the need to access a trusted third party to fulfill the authentication process that survives against replay attacks.

Work conducted by [17] in the direction of IDS for IoT proposed three different models. The first model is based on the unsupervised machine learning approach K-means to find wormhole attacks, while the second is the supervised approach based on the decision tree. Both models are dependent on the clause that if any router tries to add a new node outside of its cluster or safe distance, it is declared as a wormhole. The third model is based on the hybridization of the K-mean and decision-based tree [18,19]. The RPL protocol (Routing Protocol for Low-Power and Lossy Networks), i.e., SecTrust-RPL, may be used to handle Sybil attacks.

The MiTM attack has significant potential to harm the IoT system, as it can bypass the control of the attackers. This attack can exploit the possible path of industrial robots, which leads to potential damage to the assembly line of a system. To solve MiTM issues, the work performed by [20] proposed MQTT (Message Queuing Telemetry Transport) and MQTT-SN (MQTT Sensor Network) protocols. These protocols maintain end-to-end communication. The MQTT protocol uses a key policy, whereas MQTT-SN uses an attribute-based encryption policy and elliptic curve cryptography to defend MiTM.

3.3. Software Attacks

Another type of possible attack is a software attack. This may be launched whenever vulnerabilities are associated with the software system used for IoT systems. Viruses, worms, adware, spyware, and trojan horses are malicious programs that may steal information, are capable of tampering with the data, and may launch DoS or DDoS. Some malicious programs may even infect data present in IoT devices, clouds, and data centers. According to IoT domain experts, due to the wide range of applications in commercial as well as industrial areas, we may have 75 billion devices by 2025. As it is comparatively new in w.r.t. personal computers and the internet, it has more security concerns and more possible attacks. In 2016, an IoT botnet was used for the deadliest DDoS attack on DNS service providers by Mirai Botnet and Jeep Hack.

Countermeasures against Software Attacks: In the direction of countering software-based attacks, previous research [21] presented a framework combining multiple aspects to defend hardware-implanted Trojans. The three security aspects used in the proposed model are an encryption mechanism to prevent unauthorized access, vendor diversity to gain trust among untrusted participants, and mutual auditing to check the encryption status and message content. A similar kind of work was performed by [22], in which a secure hardware-based design was proposed to prevent Trojans. This mechanism directly prevented Trojan's injection into the network.

3.4. Data Attacks

Cloud computing plays an important role in maintaining and providing resources such as virtual servers, databases, etc. Whenever a cloud server provides these services, data security becomes a concern, and it may be improved by using software updates, firmware, and proper authentication mechanisms. Some examples of data attacks are Data Inconsistency, Unauthorized Access, and Data Breaches. Most common insider attacks are due to weak access policy or weak login credentials. Examples of data attacks were seen in 2018, when 50 million Facebook users' data were breached by Cambridge Analytica, and in 2021, when the AIRINDIA data server SITA was compromised, and due to that, the details of users, including their travel history, payment card details, and date of birth, were compromised.

Protection against Data Attacks: The protection of users' data is the utmost significant task. The following countermeasures are proposed against data attacks. The integrity may be protected by a digital signature or message authentication code. However, there are challenges in signature implementation in smartphones, i.e., Android OS, iOS, and IoT sensors because these devices are computationally intensive and difficult to implement.

The model proposed by [23] used symmetric key cryptography. The model is based on a message authentication protocol and chaos-based privacy protection to ensure the integrity and secure data communication of smart home application devices. The author proposed a blockchain-based solution to ensure integrity in [24]. The proposed approach is based on three layers, namely concepts of trust, a trust space–time protocol, and the verification of data availability and integrity. Blockchain-based architecture for ABE (Attribute-Based Encryption) also ensures the nonrepudiation, integrity, confidentiality, and privacy of data transactions. The sender sends the encrypted data, and only validated users who satisfy the access control mechanism can decrypt and use the data. Another security measure is the Improved Secure Directed Diffusion protocol (ISDDP) for end-to-end data protection in the IoT domain. In this protocol, first, a shared key is generated using the bilinear pairing method; later, that key is used for the generation of multiple layers of encryption. This protocol ensures privacy protection on multiple layers, and this makes it impossible to retrieve the original data from malicious hackers. Some of these attacks and mitigation solutions are presented concisely in Table 1.

Table 1. Categorization and brief overview of possible attacks on IoT and IIoT systems with security measures.

Attack	Subcategory	Effect	Countermeasures Proposed
Physical Attacks	Tampering (perception Layer)	Disclosure of sensitive information, Unauthorized Access, DoS Attack	Authentication based on unclonable function.
	Malicious Code Injection (perception Layer)	Disclosure of sensitive info, Access gain, DoS	Authentication based on PUF.
	RF Interference/Jamming (perception Layer)	Squeeze Communication. DoS	CUTE mode (Customizable and trustable end device mote)
	Fake Node Injection (perception Layer)	Man in Middle attack Data flow control	Pervasive authentication protocol
	Sleep Denial Attack (perception Layer)	Node down	CUTE Mote SVM (SupportVector Machine)
	Side Channel Attack (Perception, Application Layer)	Gather encryption keys	PUF Authentication Masking technique
	Permanent Denial of Service (perception Layer)	Destruction of Resources	NOS middleware (NetwOrked Smart object)
Network Attacks	Traffic Analysis Attack (Network Layer)	Network Data Disclosure	Traffic obfuscation framework, Privacy protection
	RFID Spoofing and Unauthorised Access(Network Layer)	Data Modification	SRAM based PUF
	Routing Information Attack (Network Layer)	Loop routing	Authentication using Hashchain
	Selective Forwarding (Network Layer)	Message Destruction	Authentication using Hashchain, Monitor approach
	Sinkhole Attack (Network Layer)	Data leak and alteration	Authentication using Hashchain, Intrusion detection
	Wormhole Attack (Network Layer)	Packet tunneling	IDS using clustering
	Sybil Attack (Network Layer)	Partial resource allocation, Redundancy	Trust-based Protocols
	Man in the Middle Attack (Network Layer)	Violation of Data Privacy	Message Quer Telemetry Transport, Random number-based mutual authentication, Secret key and parameter-based authentication
	Replay Attack (Network Layer)	Network congestion, DoS	Signcryption
	Denial/Distributed Denial of Service(Network, Processing Layer)	Network Flooding and crash	SDN supported IoT framework, Multilevel DDoS mitigation framework, EDoS server
Software Attacks	Virus (Application Layer)	Infect system	High-level synthesis
	Malware	Data Infected	Lightweight Neural Network-based framework, Malware image classification
Data Attacks	Data Inconsistency (Network, Processing Layer)	Inconsistent Data	Blockchain-based architecture Chaos-based scheme
	Unauthorized Access (Processing Layer)	Data Privacy violation	ABE using Blockchain and Privacy Preserving
	Data Breach (Network Layer)	Data Leak	Data Protection and Privacy, Multi-factor (two) authentication

4. Android System Architecture and Attacks

In November 2007, Android was unveiled by Google and the Open Handset Alliance (OHA). As per the latest report of the International Data Corporation, nearly 85 percent of mobiles are using the Android operating system. In 2020, Google revealed that 400 million devices were running Android version 10, and the latest version is Android 11. Android has four types of components: Activities, services, content providers, and broadcast receivers. With the rising number of users, the number of threats and exploitations are also rising. According to the authors of the work [5], malware developers inject code into the service component that runs in the hidden layer without the concern of the user. The first Android Malware included DroidSMS, which targeted SMS services, and TapSnake, which transmitted GPS locations. Recently, CamScanner, which has millions of downloads from the GooglePlay store, was detected as malware in Android OS. The research conducted by Kaspersky on 5000 applications recorded more than 1000 cases of known vulnerabilities and 700 zero-day vulnerabilities. Therefore, a deep understanding of Android Devices, Architecture, and Android applications is a vital requirement. The overall Android architecture is presented in Figure 3. Attacks and exploit are possible at every layer, including hardware and drivers used to run devices.

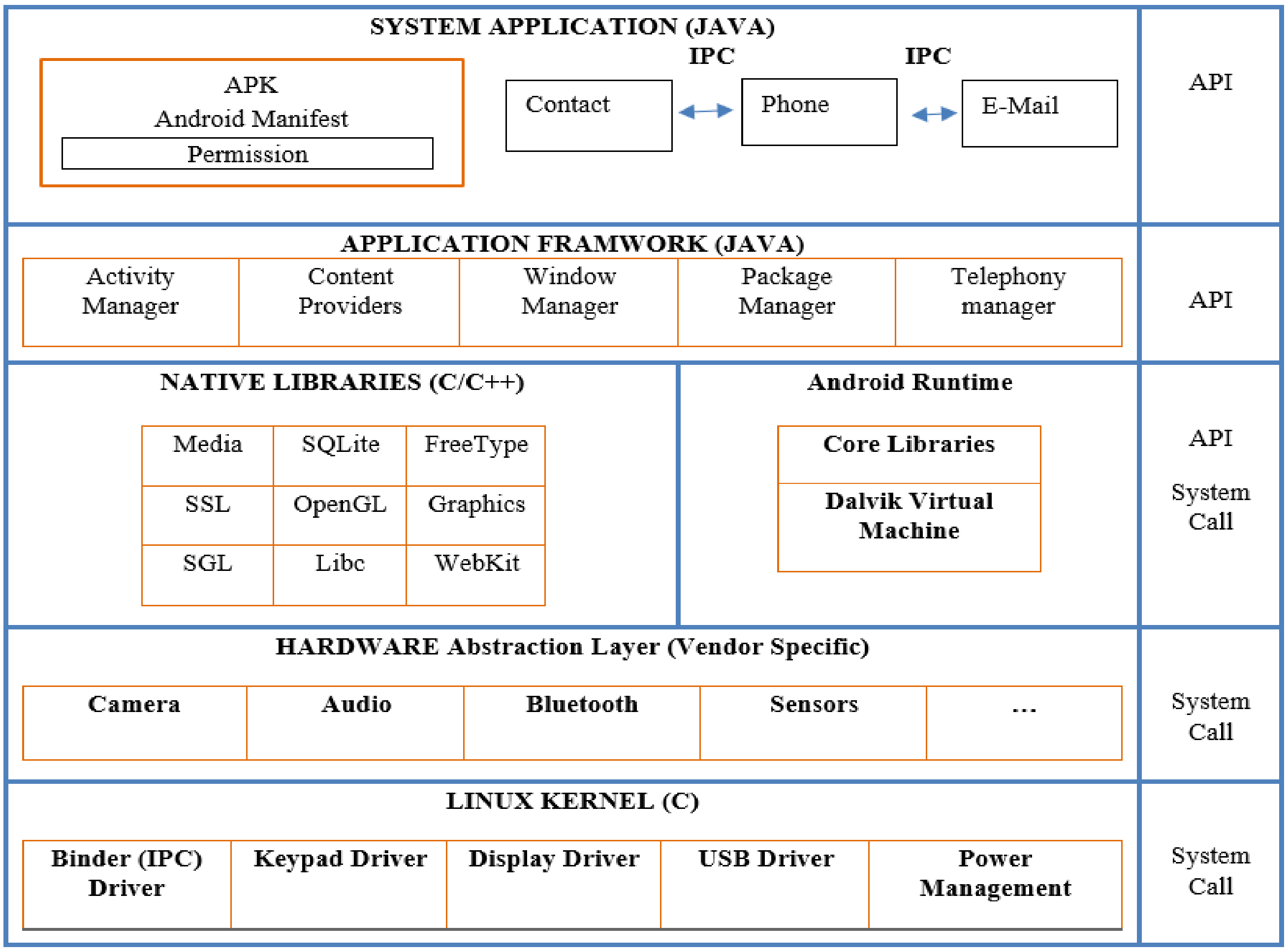

Figure 3. Android framework layers.

Attacks on Android devices may be categorized into four categories, namely hardware-based attacks, kernel-based attacks, attacks based on the hardware abstraction layer, and application-based attacks. In this subsection, we mention certain threats, challenges, and corresponding solutions, and this is summarized in Table 2.

Table 2. Layered attacks on Android Architecture with proposed security measures.

Classification of Attack	Type of Attacks	Vulnerabilities Exploited by Attack	Strategy and Effect of Attack	Security Measure
Hardware-Based Attack	Rowhammerattack	DRAM	Memory leakage.	GuardION
	Drammer attack	DRAM chip	Gain root access	GuardION
	Glitch attack	DDR3 and DDR4	Use graphics code to exploit DDR3 and DDR4 loopholes	GuardION
	QuadRooter Attacks	Linux IPC	Root Access	Patch with Distributer
	RAMpage attack	RAM component	Gain access to sensitive memory location. Leak Information.	GuardION
Kernel-Based Attacks	DroidKungFu (Root Privilege)	Root privilege of adb and udev	Privilege escalation, unlock all system files and functions.	Security enhanced
	ROP (Code reuse)	The executable code present in the library	Overwriting of the return address, point to arbitrary code.	Instruction set randomization
	Gooligan (Root Privilege)	Kernel API put_user/get_user	Inappropriate validation, Root access, privilege escalation	NA
	SOP (Code reuse)	Format String	Overwrite GOT entry, stack function pointer	Stack canaries
	JOP (Code reuse)	Vulnerable jump instruction	Overwriting stack, setjmp pointer, function pointer	Program shepherding
	Return to user attack (Code reuse)	The shared address between user and kernel	Overwrite kernel pointer, redirect execution flow.	kGuard
	TOCTOU (Zero Day)	The user and Driver havedirect access to the buffer	Gaining device driver control, corrupt memory, privilege escalation	NA
	Kernel Space Mirroring Attack (Permission Exploitation)	Read/Write data access to EL (0) user mode	Accessing kernel	NA
	Boot Loader Attack (DoS)	Non-verifiable address	Flood of traffic	BOOTSTOMP
Hardware Abstraction-based attack	Invisible man attack (sensor-based attack)	Security weakness	Launch Phishing attack	Download applications from Trusted sources.
	Brute force attack (Encryption system)	Weak passwords and Authentication system.	Guessing of right key pair.	Strong encryption algorithm. Two-factor Authentication.
	Key reinstallation attacks (Replay attack)	Allow reconnection using the same key.	Reset the encryption key.	New Patch
	Audio channel attacks (media-based attacks)	A security weakness in Android accessibility application.	Use services of Talkback.	AuDroid policy enforcement.
	Convert channel attacks (sensor-based attacks)	Exploit IPC and permission system.	Retrieve data from the sensor.Steal sensitive data.	Security policy enforcement. Strong permission.
Application-Based Attacks	Energy-Based Attack (DoS)	Complex inter relationship between network and hardware	Resource consumption.	Strong Permission
	Runtime information Gathering Attack (permission exploit)	Default permission system	Malicious application use.	App Guardian
	Code Injection Attack (Permission exploit)	Third-party libraries	The attacker gains excess permission.	Profile matching algorithm
	Ad Libraries attack (Permission exploit)	Intra library collusion	The attacker exploits additional permission.	Justifiable permission
	Mam in the middle attack (Network Attack)	Non-authentication	The attacker pretends legitimate host	TLS public key Infrastructure.
	Cloak and dagger attack (Permission exploit)	Default permission system	Control over the user interface.	Strong Permission.
	Dirty COW (Permission exploit)	Privilege Escalation Attacks	Create race conditions.	New Patch

4.1. Hardware-Based Attacks

Any device is made of a range of hardware, such as a printed circuit board, processor (central processing unit, baseband processor, application processor, digital signal processor, graphical processing unit), memory, touch screen, radio frequency interface, Wi-Fi/Bluetooth device, SIM card, USB device, camera, and different sensors: GPS, motion detection, environmental sensor, heat detector, network sensor, etc. In a hardware attack, the attacker tries to exploit the preexisting vulnerabilities present in the hardware design component. These kinds of vulnerabilities are hard to patch because the patch is only available from distributors, and user-end updating is not possible. If these hardware components are repaired by a third party, then it causes a question mark and threat regarding the integrity of the device. This issue may be solved by cross-checking the communication integrity between the repaired device and the central processing unit.

The Rowhammer is a type of hardware attack in which malicious JavaScript is used to attack the Android device. After successful installation of the script, the attacker gains access to GPU (general processing unit) and DRAM (dynamic random-access memory) and causes the leakage of electrical current, i.e., Glitch, using the system library WebGL. The triggered Glitch can harm DDR3 and DDR4 memory chips and exploit the GPU. RAMpage classified as CVE-2018-9442 is an attack similar to Rowhammer where attackers gain access to the restricted memory location and the complete system. GuardION is an open-source technique that protects the system from authorized memory-access leaks and Glitches. QuadRooter is a set of four vulnerabilities identified as CVE-2016-2059 (Common Vulnerabilities and Exposures), CVE-2016-5340, CVE-2016-2503, and CVE-2016-2504, which are present in the Qualcomm chipset. The attacker can gain access to the root of the application and may exploit the vulnerabilities present. The patch is available from the distributor or Qualcomm.

4.2. Kernel-Based Attack

Android is implemented over the kernel, and the Linux kernel is responsible for every action and the management of different activities. The Linux kernel takes care of Android runtime environment functionalities. The attacker tries to exploit the kernel and related components such as the memory, device driver, and run time environment. Even though Android uses several security measures such as Secure IPC (Inter-Process Communication), Android Sandbox, cryptography, and an encrypted file system, vulnerabilities continue to be exploited, and some recent examples may be broadly classified into (1) an attack that targets the root privilege, (2) a memory target, (3) a boot loader target, and (4) a device driver. A brief attack pattern, description, and solutions are mentioned below.

The attacker tries to access the critical section using the keyclt() function of the Linux kernel and exploits the memory leakage, i.e., CVE-2016-0728.This provides unprivileged access to the system. The zero-day vulnerability CVE-2013-6282 known as a Gooligan Attack exists due to an error in put_user/get_user, a kernel API. The attacker performed unauthorized read and write operations in memory using malicious code. This leads to the exploitation, access, and execution of malicious code. This vulnerability was present in Android versions 4 and 5. The second root privilege attack malware for Android devices is DroidKungfu. It uses vulnerabilities identified by CVE-2010-EASY and CVE-2009-1185. The motivation of this malware is to silently root into the mobile device and unlock the file system and functionalities. It exploits the system resources without requesting services from the kernel. This is performed by exploiting a vulnerability present in the root level code "udev" by sending a NETLINK udev event ping. The message hint to udev runs into an arbitrary binary as the root.

Jump-oriented programming, Return-oriented programming, Return to libc, and ret-2-usr are some examples of such kinds of attacks. These attacks exploit the address space of the code segment and address space of the kernel. A great deal of work was conducted to secure devices against these attacks, including the randomization of the user address space and kernel address space.

The bootloader is responsible for scheduling vital tasks at the initialization of the operating system and kernel process. Mobile phone's bootloaders have the special responsibility of initializing the system, maintaining the device's security, and ensuring the integrity of CoT (Chain of Trust). There are several cases and malware that hinder the process of bootloaders from performing their function and also make them vulnerable to attacks. Most of the vulnerabilities are from the insecure design and coding errors categorized as per OWASP Top 10 2021 (Open Web Application Security Project). The malware BOOTSTMP exploits the vulnerabilities CVE-2014-9798 and CVE-2015-8893 that are present in Qualcomm, Huawei, and NVIDIA's bootloader. These vulnerabilities allow attackers to run malicious code, hence imposing memory corruption, buffer overflow, privilege escalation, and denial-of-service attacks on the device. Designing a bootloader while keeping in mind the following issue is very crucial for developers.

The vulnerability present in the device driver allows an attacker to breach the driver, cause the device to become unstable, gain access control, and crash devices. During development, the designer's primary focus is on driver functionality, not security. When a new or updated driver appears, these are made available in the market by the developer, distributor, or using reverse engineering, and an attacker finds security gaps available in that driver. The researcher of Zlab discussed two vulnerabilities, CVE-2016-2411 and CVE-2016-2435, which were present in the MSM thermal driver, and the NVIDIA video driver affects Android version 6.0 in Nexus devices. The implementation flaw and vulnerabilities were present in both the kernel and external driver. The attacker used malicious applications and successfully gained root access and launched attacks, i.e., privilege access, root access, complete access gain, compromised confidential data, etc. The solution relies on the developer understanding what to do and what not to do during the design and implementation of drivers.

4.3. Hardware Abstraction Layer-Based Attacks

This layer contains library modules for the implementation of different devices such as Wi-Fi, Bluetooth, Camera, GPS, and other components of Android devices. Here, the attacker targets the interface and uses these components. Most of the devices contain a Wi-Fi chipset designed by the M/S Broadcom company, and these chipsets are highly vulnerable to a Booby-Trapped Signal. The same kind of vulnerability was present in Wi-Fi security protocol WPA2 in Android version 6.0 and higher, IOS, OpenBSD, and Windows. This vulnerability was exploited using an advanced attack method named KRACKs, i.e., Key Reinstallation Attack. To exploit the vulnerability, the attacker needs to be in the Wi-Fi range. This attack can steal confidential data, decrypt encrypted data, and undertake spying activity using a camera, voice recording, or GPS. The GooglePlay service also offered common vulnerability challenges, e.g., default service enabling and disabling, which affects the normal functioning of the android device.

Attackers also use rooting and jailbreaking tools, i.e., Aircrackng, to generate passwords. After gaining kernel control, the attacker gains full access to all the data and system resources. The solution to this issue is a stronger encryption algorithm. This process is slightly challenging to execute when considering brute force attacks due to the requirements of memory, computation resources, and intensive knowledge.

4.4. Application-Based Attacks

Android applications are available in the Google Play store and other database stores. These applications require certain default permissions and many other users, which are never used by applications. Later, the Malware designer exploits the vulnerabilities present in this application, and the user suffers attacks such as privilege escalation. The attacker can leverage the third-party library and native libraries used for applications. Attackers gain root privileges and can cause irreparable damage to devices, perform spying activity leading to critical damage, or sell the gathered information. Upon the running/installation of a malicious application, if it gathers sensitive information, then it is a kind of RIG attack (Run time Information Gathering). This attack can result in a potential threat to the IoT device control administered by Android devices and can harm the CPU, files, battery, disk, and memory of the device. App Guardian is a tool that handles RIG attacks, in which it monitors the suspected application and suspicious activities and closes that application, cleaning the memory to make sure nothing is left in the memory for the attacker to access. WHYPER is a Natural Language Processing-based tool that analyzes the requirements of different applications and the permission used by the application. When application-based attacks use more power and devices are exhausted quickly, it is termed energy-based attacks.

Other library-based attacks were present in OpenSSL and Boring SSL and considered critical, allowing the attacker to corrupt the memory. The solution to this is available in the updated version of OpenSSl1.0.2. The GNU C library used the third-party function getaddrinfo(), which may be exploited for CVE-2015-7574 from are mote system. These vulnerabilities may breach the user's privacy, expose personal information to the dark web, code injection attacks, and can even hijack the system's control, compromise the Dropbox account, etc. Table 2 represents a summary of the above section.

5. Android Vulnerability Detection

Vulnerabilities may be detected by static analysis, dynamic analysis, or a hybrid of the two. Some techniques have been presented in this section.

5.1. Methods Based on Code Analysis

TinyDroid and DroidMoSS [7] are based on a static malware detection technique. The input of TinyDroid is the application/.apk files, and machine learning is used for parameter learning. By using the Apktool, first .apk file is decompiled into Smali code. Smali is a higher-level explanation of the Dalvik bytecode, which may be further abstracted into instruction representation. TinyDroid finds n-grams from the instructions and uses them for classification. These n-grams are matched with benign and malicious applications' n-grams. Based on this technique, classification takes place. The authors claimed that some antivirus performances are below fifty percent, but TinyDroid's classification rate is 95.6%. The clone detection technique is another approach for Android malware. The process first uses dex2jar for the conversion of the Dalvik VM (virtual machine) bytecode into the JVM bytecode (Java VM). In the next step, the Java bytecode was decompiled using the JD-CORE java decompiler. Decompilation yields higher-level code, which is used for clone detection. The limitation of this approach is that it can detect the malware from defined classes and similar malicious files. DroidMOSS is presented to measure the similarity between applications to find the repackaged applications. To do that, applications are considered from a third-party marketplace, their distinguishable features extracted, and application-specific fingerprints are generated. Fingerprint generation is performed by the fuzzy hashing technique. The third-party application marketplace chooses from 6296 apps in USA, 12,595 apps in China, and4015 applications in Europe considered for experiments. These applications are measured against 68,187 apps from the Android market. The author concludes that 5–10% of applications are repackaged, and 13–30% of applications are just redistributed on the Android marketplace. These repackaged applications not only harm the economic interest of a developer/nation, but repackaged applications are also more vulnerable to attacks such as backdoors and Trojans. The benefit of this approach is that it operates directly on Dalvik code without source code requirements.

5.2. Android Vulnerability Detection Methods Based on API Calls and Permissions

The algorithm DroidSieve used extreme gradient boosting to perform various trial-and-error runs to trace and segregate Android malware pertaining to the fixed inflow of data. DroidSieve has evaluated over 1 million faulty apps, achieving a very high detection rate of over 99%. However, since it performs malware detection by checking for similar patterns in the app's code, it may not be robust against mimicry attacks and app cloning.

Code conversions [25] from one form to the other enable an analysis of the performance of any application in terms of the time taken to carry out the conversion process. A sequence of API calls also can be associated with a set of permissions to keep the malicious access away from creating any havoc across the network. However, the use of various machine-learning algorithms to estimate API-based detection on any real-time dataset can achieve better accuracy rates, but, at the same time, can also be computationally expensive concerning retrieving the set of permissions associated with the specific call, DroidMat, another approach based on fixed information analysis, includes permissions, intents, which are the activating objects that contain information about other elements, and API calls, in addition to the type of components integrated with the service. The use of various machine-learning algorithms such as K-Means and k-nearest neighbors (KNN) to infer conclusions about the static analysis are not applicable in dynamic analysis. Hence, it should be implemented in a manner that supports both fixed and variable analyses.

5.3. Android Vulnerability Detection Based on Risk Signals

Before the installation of an app, a thorough analysis should be carried out by the users to prevent malicious access from running in the background, eliminating the stakes of a confidential data breach. Furthermore, in many real-time scenarios, the setup of an account asks the user to grant permission to access their contacts, location, camera, and many other features, which may exploit confidential data to derive commercial benefits if falling into the wrong hands. The authors in [26] developed a model that relies on app-based permissions as supported by Google Store environments. In addition to this, a sequence of additional apps has also been tracked to perform efficient malware-free analysis, as unusual permissions increase with the risk of app installation.

Probabilistic approaches used to detect and work on the elimination of malware play a prominent role in terms of mathematical model design and applications. However, the implication is thatthe outcome for the problem under consideration has an acceptance rate of nearly 50%. The work presented in [27] uses a rank-based approach to grade the access permissions. The adopted approach uses a mix of classification techniques that enable the categorization of apps installed in the user's environment into harmful and harmless apps. However, the methodology is only applicable against malware detection and resolution for a set of operating systems, making it inefficient to assess the risk factor against a wide set of emerging systems.

DroidAPIMiner performs classification using four commonly used classification algorithms, namely ID5 DT, C4.5 DT, KNN, and SVMs. The optimal results of 99% accuracy and a 2.2% false-positive rate were obtained when using the KNN classifier. However, as the dataset size increased further by adding more features, the accuracy did not remain the same and was degraded, thereby defeating the overall purpose of the classification techniques used. The DroidRanger tool uses a variety of malicious families to detect the behavioral patterns present in malware. It performs the first operation based on a permission-based behavioral footprint for classifiable malware, and another, for previously unknown malware, that is based on a heuristic analysis of the app's behavior. Suspicious applications are then executed and monitored to verify whether they display malicious behavior at runtime. If this is the case, the associated behavioral fingerprint is extracted and included in the first detection process database. However, DroidRanger only covers free applications and only five Android markets, with a false-negative rate of 4.2%, which is highly insignificant for a dynamically increasing set of applications.

5.4. Android Vulnerability Detection Using NLP

Natural Language Processing (NLP) paves a path to identifying and processing abundant approaches to deal with the widely emerging text and other data formats originating. The experiments can also be efficiently conducted through the inbuilt library support offered by a variety of programming domains to process the language semantics. The analysis is more productive as a pair of attributes are considered to identify the correlation among them. Such models can be developed irrespective of the source code. Though the probabilistic approaches serve to provide accurate results to estimate the correlation amongst the attributes, they do not infer any assured outcomes for any combination of data samples. The authors in [28] proposed a Long Short-Term Memory model (LSTM) to eradicate the aforementioned difficulties by invoking system calls. LSTM assigns a probability to the occurrence of a sentence (i.e., the sequence of system calls of the application being monitored) in both the valid and malicious models. Upon testing the model under different conditions with varying lengths of system call sequences from 50 to 50,000, the authors achieved an accuracy rate of above 90%. However, if the number of samples was found to increase further, the accuracy may either drop or remain constant for streaming analysis. Furthermore, the removal of noise from the samples is not covered while dealing with the malware identification and removal process.

5.5. Android Vulnerability Detection Using System Call Logs

The model proposed in [29] is used to detect malicious behavior at runtime by executing the apps in a controlled environment for a static time and recording the system call occurring during this time. The most common systems used for the classification of malware using learning methods include the Naive Bayes algorithm, the Random Forest algorithm, and the stochastic descent gradient algorithm. This work showcased a malware detection rate of over 95% with a misclassification rate as low as 8%. The work used a particular Chi-square model to cater to the analysis of malware, which may not serve the accurate traceability aspect of every recorded system call.

Previous researchers [30] relied upon system calls to perform malware detection. Their method draws upon the fact that malware tends to evolve through an iterative process of borrowing and modifying code from other malware. As a result, malware samples are likely to share common behavioral features. The authors performed an extensive experimental evaluation using a real device on which they executed more than 1000 apps for a total of nearly 20 K runs, which led to deriving an accuracy of over 95%. However, a malware classification technique alone could not judge the validity of the call being traced. There is a need to integrate the approach with malware elimination techniques.

Social media plays a prominent role as a data generator source that may increase the fear of safe data falling into unsafe hands. Furthermore, the use of low-cost servers as a deployment environment is significantly preferred over high-end servers to support scalability to a greater extent. This phenomenon can be applied to the data emerging via call logs associated with smartphones, irrespective of the nature of the operating system. A similar idea underpins the static methods of previous authors [31] who looked for code segments in multiple apps. A dynamic approach benefits from the fact that the same behavior can be encoded in a number of different, but semantically identical, ways. An analysis of over 7000 Smartphone application binaries showed that the developed schemes detected all instances of plagiarism from a set of real-world malware incidents, with low false-positives and on a scale of millions of applications, using only commodity servers. The meta-information of nearly 158,000 applications from the Android Market was analyzed, and it was found that nearly 29% of applications are likely to be plagiarized. However, the approach is only proven to be efficient for one class of systems, uncovering the malware issues identified with a generic set of classes. Furthermore, the assumption of random methods to perform the analysis is insignificant when applied to real-world data samples in the long run.

5.6. Android Vulnerability Detection Using Monitoring of System-Level Behaviors

EnDroid is a malware detection system based on several types of dynamic behavior at the system level. A feature selection algorithm is adopted to eliminate irrelevant features and extract critical features from the behavior. In this approach, various application actions such as cryptographic operations, network operation, file operation, information leaks, SMS messages sent, telephone calls, receiver actions, receiver startup, .dex class loading, and system calls were first monitored. Experimental results show that this approach detected nearly more than 95% of malware with a good ratio of false-positives. The proposed approach makes it hard to examine the flooding traffic over the device.

Andromaly repeatedly applies supervised anomaly detection techniques to continuously monitor various system measures to detect suspicious activities. The various service components synchronize feature collection, malware detection, and the alerting process. The features upon which Andromaly relies include CPU consumption, the number of packets sent over the network, the number of processes running, and battery usage. However, malware detection alone does not serve the recovery and removal approaches that exist at a higher requirement rate for a particular class of systems. Moreover, the supervised approaches are limited to dealing with the emerging data samples due to high-speed bandwidth across the network. Hence, there is a need to devise a generic model that works well irrespective of the sparse or dense availability of the samples.

5.7. Android Vulnerability Detection Using Monitoring of User-Space Level Behaviors

The design of a device is often integrated with its user interface functionality to attract users and increase sales. The tool RespassDroid combines grammatical and lexical analysis to automatically detect malicious Android programs. This synthesizes the API used in the application as a semantic function and the essential permissions as a syntactic function with an emphasis on two prime parameters. First, it generates a call graph of each application, and then it extracts the application's features in terms of APIs and permissions from the graph to form feature vectors. The accuracy rate of the proposed technique is above 90%. However, the limited set of algorithms against which accuracy tests are conducted remains to be proven for the scalable set of applications dynamically spanned across the distributed environment.

The authors in [32] proposed a malware detection scheme for Android devices based on the SVM (support vector machine) automatic learning classifier. The system is efficiently designed for optimal smartphone utilization service. A warning is issued to the user upon installation of an infected app. Experimental results show that this system yields an accuracy rate of over 95% and a false-positive rate of over 13%. However, the chosen dynamic features vary from one model to the other, and it is difficult to assess the behavior of the device in such instances. The XManDroid (eXtended Monitoring on Android) tool analyzes variable application permission percentages to detect a varying notion of attacks at dynamic phases. The aforementioned attack can creep in when the invocation method is nested into layers. At the other end, the interactions between a set of devices need to be monitored to keep away the intruders. XManDroid was tested using a custom-made suite of malware using a reasonably good number of benign apps taken from the Android market. The results showed that XManDroid exhibits a low false-positive rate. However, the coupling of authentication with hash-based security mechanisms in both fixed and transition-based environments is a great focus to perform effective malware analysis.

5.8. Image Analysis-Based Method

DeepVisDroid is an amalgam of Image fusion with a one-dimensional convolutional-layer neural network. Multiple image-based local and global features are extracted and used to train a convolutional neural network in multiple scenarios. Multiple files from the Android app's contents are converted into grayscale images, such as the Manifest.xml file-based dataset Manifest, DEX code file-based dataset, Manifest and Resources.arsc file-based dataset, and Manifest and Dex file-based image dataset, to construct four grayscale image datasets. Then, the four different image-based local features and three different image-based global features were extracted to train the proposed model. After that, a lightweight, one-dimensional, convolutional layers-based CNN model was proposed and trained using the extracted image-based global and local features. Furthermore, two classical CNN models were proposed, and two well-known deep learning models, namely ResNet and Inception V3, were tested to compare the results of the proposed DeepVisDroid with the results of the state-of-the-art deep learning model [33]. The obtained results show that the proposed model outperformed the classical CNN models in terms of classification accuracy and computational cost.

Based on attributes collected from images and a manifest file, the authors in [34] offer a unique mobile botnet detection system. The approach combines features derived from the manifest files with a Histogram of Oriented Gradients and byte histograms produced from images representing the executable app. The best features are then used for categorization with ML algorithms after feature selection. The suggested system was tested on the ISCX botnet dataset, and the findings show that it works, with F-1 varying from 0.923 to 0.96, employing standard Machine-Learning techniques. Furthermore, employing an 80:20 train–test split and 10-fold cross-validation, the Extra Trees model [35] achieved up to 97.5 percent overall accuracy and 96 percent overall accuracy.

5.9. Policy Enforcement and Device-Based Solutions

The authors [36] proposed a privacy tool labeled Paranoid Android, which efficiently identifies several malicious access mechanisms on remote servers hosting another replica of the user's device. The design is unique in its role of proving compatibility with the cloud environment. A log of all actions is maintained on the user's device, as well as on the cloud, to match the sequences in terms of fraudulent access. The designed system is used for various kinds of analysis such as instant and variable analysis, anti-viruses, memory scanners, and system call-based analysis. The benefit of this is the capacity to perform multiple detection mechanisms simultaneously to achieve a better detection rate. However, it has been identified that the battery life is drained due to the transmission of data from one mode to the other while also retrieving the encrypted versions.

The SeqDroidtool, assigning an identity to every device being sold on the market, enables us to trace the device in the event of theft or loss attempts. This sequencing varies from one model to another and is further encrypted to prevent malicious access by hackers. The user purchasing a particular handset should digitally sign a certificate that acts as a source during the transmission of ownership and also to prevent fraud. The representation of data samples using mathematical models is of utmost interest to the researchers to provide an effective understanding of the design via a set of flow models or equations. It is also essential to preserve the consensus across all the participating entities to ensure validity and transparency in terms of log access from a third-party virtualized environment. However, the application of mathematical models should be justifiable across a varied set of operations to identify the malicious attempts made to access the data samples. This may come with the integration of several domains and increase the maintenance costs. According to experts in [37], billions of users throughout the world install Android apps on a daily basis, granting access to a vast array of sensitive personal data. Over time, several methods have been developed to better understand how apps protect or compromise the privacy of their users. These findings, on the other hand, have come from a variety of research disciplines and approaches to privacy, resulting in a growing but dispersed body of information. To close this gap, researchers conducted a thorough mapping study to offer practitioners and scholars an overview of the state-of-the-art techniques for assessing privacy in Android apps, released between 2016 and 2020. They highlight the most important discoveries, identify and analyze the most critical gaps, and outline promising research. A summary of the proposed techniques is presented in Table 3.

Table 3. Recapitulative table, where RF stands for Random Fo22rest, and Dynamic tool is prescribed by the keyword Dynamic.

Tool	**Efficiency**			**Feature and Technique Applied for Vulnerability and Malware Analysis**
	Accuracy	**Precision**	**Recall**	
NSDroid	95%	96%	95%	The similarity between Application, Function call graph, RF, SVM, Decision Tree
TinyDroid	95	92	95	Disassembled application using smali code, Naïve Bayes, RF, SVM, kNN
DroidSieve	99	99	99	Static analysis, synthetic features used, SVM, Extrac Tree, XGBoost, RF
Qiao et al. [25]	94	-	-	API call and Permission, Source code analysis, RF, SVM, Neural Network
DroidAPI Miner	99	-	-	Permission and API call, SVM, KNN, C4.5, ID3
DroidMat	97	96	87	Clustering, Activity service and receiver component, Singular value decomposition, K-Means, KNN
DroidRanger	-	-	-	Behavioral footprint, Permission
Sarma et al. [26]	-	-	80	Attribute risk level, permission, SVM
Xiao et al. [28]	93	91	96	System call trace, LSTM, SVM, KNN, Random Forest
RepassDroid (Dynamic)	97	99	96	API call and semantic information, Decision Tree, SVM. KNN
EnDroid (Dynamic)	97	95	97	System call input output, Decision Tree, Stochastic Descent algorithm, Boosted Tree
Chaba et al. [29] (Dynamic)	-	95	95	Runtime monitoring, Random Forest, Naïve Forest

6. Application Hardening Technique

Vulnerabilities of some sort always remain with in the system, and developers are unaware of this. As a result, the application-hardening technique [14,38,39] offers a viable alternative. There are multiple methods for accomplishing this, and some of the existing solutions are discussed below.

Data Execution Prevention (DEP): DEP is host-dependent system-level memory protection technology that is used to protect against malicious code and injection attacks. This feature designates certain memory locations as protected areas, where no code is performed. If an application attempts to execute code in those protected areas, DEP raises an exception as a memory access violation. Both hardware and software can be used to implement DEP. The hardware-based implementation is performed at the processor level. Address Space Layout Randomization (ASLR) improves security by limiting resource access and preventing current weaknesses from being exploited. During the booting process of some operating systems, engineers fix the original and starting addresses of each segment. Certificate and Public key Pinning (CPP) refers to pinning an X509 certificate and its public key to a root. This aids in the establishment of trust between the sender and the receiver, as well as ensuring the integrity of the website following the verification of a valid certificate and removing harmful targets that break the pinning rules. Null Page Protection is a valid address that refers to storage locations that are currently empty. Null page protection enables PAGE GUARD on a specific location and reallocates virtual memory at 0X00000000 to address such issues. Heap Spray Protection is an attack in which malicious code populates a heap, causing any memory vulnerabilities to be diverted to malicious code. Heap spray prevention aims to reduce the number of heap spray attacks. Regarding CPU-enhanced application security, modern processors have isolated instruction-level execution and runtime memory allocation to address these difficulties. Some of the processors' enhanced security features in modern processors include the Virtual Machine, VT-X/AMD-V, ARM Trust Zone, and Intel SGX. In Virtual Secure Mode (VSM), a spume develops above the hypervisor, and this spume acts as an isolated operating system, tagging connected memory spots and specific programs. Hypervisor code and the kernel mode for integrity and control, a non-physical platform for increased key-based access, and local security authority are among the four components of the virtual secure mode. The local security authority (LSA) component is responsible for protecting user credentials and authentication within a private hypervisor environment [13]. As a result, people with local admin privileges are unable to view the hashed passwords of registered users. Devices communicate with one another in real-time using pre-programmed mechanisms or algorithms. The trusted platform module (TMP) is a hardware-based security solution for the host that uses key-based access mechanisms. TMP is installed on the motherboard and uses unique connections to communicate with the host. Endorsement, Storage Root, and Attestation are the three keys in this module. An endorsement key is a pair of RSA keys that are implemented on a semiconductor and are inaccessible to software. The storage root key is produced by the administrator during the ownership process.

7. Limitation and Novelty of Work

This work proposed the integration of security measures for Smartphones and IoT devices. During the control, communication, and authentication, devices communicate similar to peer devices. If any device, either IoT or smartphone device, has vulnerability (hardware or software), then that may be exploited by attackers. In this work, security measures for IoT devices against attacks (physical, network, software, and data) and for Android devices/operating systems against attacks (hardware-induced, kernel-induced, hardware abstraction-induced, and application-based) are enlisted. Corresponding security measures proposed in the literature are also enlisted here.

The limitations of this article are that threat classification and solutions are not discussed as per the OWASP Top 10 vulnerabilities (Open Web Security project). Vulnerability classification for IoT devices is not presented here. In the future, we will try to set up vulnerability classification for IoT devices as prescribed for Android OS, i.e., OWASP Top 10 and NIST (National Institute of Standards and Technology) guidelines.

8. Conclusions and Future Scope

The primary objective of this work is to present the current security state of IoT and Android systems. Developers and researchers are developing new technologies to safeguard users and devices from attackers. This work discussed several attacks on various layers of IoT and Android devices, as well as countermeasures offered by researchers. This work tabulated explanations of threats and the solutions to prevent them. Several important findings, including secure hardware architecture design, a two-step malware detection approach, i.e., static followed by a dynamic approach, or hybrid techniques for malware identification, are suggested. A strong suggestion is provided for developers to follow security protocols, proper authentication and authorization, two-step or multi-step verification, and a strong encryption algorithm. Along with these proposed techniques, some additional steps that can provide a more secure environment are:

1. In order to solve the privacy issue, application developers should restrict permissions, and only essential permissions should be granted. A robust permission system and a standard should be followed.
2. To limit the common vulnerabilities and zero-day exploits, the database, i.e., the Google Play Store, must come with robust permissions, malware detection tools, and clone detection techniques, to protect users from exploitation.
3. Whenever a new application from a third party is installed on an Android device, it must warn the user about the threat and the application must submit to Google for static and dynamic analysis of potential threat identification.
4. Android must design an application where the user can trace the requested IP address, webpages, port request, memory use, and CPU analysis in an accessible way, as is available in Windows OS.
5. In order to comply with the CIA triad, developers must follow the latest SSL/TLS security protocol on every page of the application.

6. In order to solve the authentication and authorization issue, users must follow the direction and advice of strong passwords, phishing sites, URLs, and not sharing personal and confidential information, i.e., OTP, Login id, password, etc.
7. Applications related to the financial sector must be checked by CERT-in (The Indian Computer Emergency Response Team) and STQC (Standardisation Testing and Quality Certification) impaneled organizations.
8. Continuous innovation and research are required to protect against unforeseen threats and vulnerabilities in known and unknown environments.

Author Contributions: Writing—original draft, C.S.Y.; conceptualization, writing—review & editing, J.S.; visualization, validation A.Y.; visualization, resources, specifically visualization/data presentation, H.S.P.; data curation, resources. conceptualization R.K.; funding acquisition, analysis A.A.K.; project administration, supervision, funding acquisition, M.A.H.; funding acquisition, methodology, writing—editing, A.A.; funding acquisition, analysis, writing—review, S.A. All authors have read and agreed to the published version of the manuscript.

Funding: S.A. like to thank the Deanship of Scientific Research at Majmaah University for supporting this work under Project No. R-2022-214.

Acknowledgments: S.A. like to acknowledge the Deanship of Scientific Research at Majmaah University for supporting this work under Project No. R-2022-214.

Conflicts of Interest: The authors declare that they have no conflict of interest.

Ethical Approval: This article does not contain any studies with human participants performed by any of the authors.

References

1. Alsoghyer, S.; Almomani, I. Ransomware detection system for android applications. *Electronics* **2019**, *8*, 868. [CrossRef]
2. Wurm, J.; Hoang, K.; Arias, O.; Sadeghi, A.R.; Jin, Y. Security analysis on consumer and industrial IoT devices. In Proceedings of the Asia and South Pacific Design Automation Conference, ASP-DAC, Macao, China, 25–28 January 2016; pp. 519–524. [CrossRef]
3. Aman, M.N.; Chua, K.C.; Sikdar, B. A Light-Weight Mutual Authentication Protocol for IoT Systems. In Proceedings of the 2017 IEEE Global Communications Conference, GLOBECOM 2017—Proceedings, Singapore, 4–8 December 2017; Volume 2018, pp. 1–6. [CrossRef]
4. Sengupta, J.; Ruj, S.; Bit, S.D. A Comprehensive Survey on Attacks, Security Issues and Blockchain Solutions for IoT and IIoT. *J. Netw. Comput. Appl.* **2019**, *149*, 102481. [CrossRef]
5. Khanmohammadi, K.; Ebrahimi, N.; Hamou-Lhadj, A.; Khoury, R. Empirical study of android repackaged applications. *Empir. Softw. Eng.* **2019**, *24*, 3587–3629. [CrossRef]
6. Yang, Z.; Chao, F.; Chen, X.; Jin, S.; Sun, L.; Du, X. DroidFDR: Automatic Classification of Android Malware Using Model Checking. *Electronics* **2022**, *11*, 1798. [CrossRef]
7. Razgallah, A.; Khoury, R.; Hallé, S.; Khanmohammadi, K. A survey of malware detection in Android apps: Recommendations and perspectives for future research. *Comput. Sci. Rev.* **2021**, *39*, 100358. [CrossRef]
8. Yang, Y.; Du, X.; Yang, Z.; Liu, X. Android malware detection based on structural features of the function call graph. *Electronics* **2021**, *10*, 186. [CrossRef]
9. Yadav, C.S.; Sharan, A. Automatic Text Document Summarization Using Graph Based Centrality Measures on Lexical Network. *Int. J. Inf. Retr. Res.* **2018**, *8*, 14–32. [CrossRef]
10. Yadav, C.S.; Sharan, A.; Joshi, M.L. Semantic graph based approach for text mining. In Proceedings of the 2014 International Conference on Issues and Challenges in Intelligent Computing Techniques, Ghaziabad, India, 7–8 February 2014. [CrossRef]
11. Guebli, W.; Belkhir, A. Inconsistency Detection-Based LOD in Smart Homes. *Int. J. Semant. Web Inf. Syst.* **2021**, *17*, 56–75. [CrossRef]
12. Kouliaridis, V.; Barmpatsalou, K.; Kambourakis, G.; Chen, S. A survey on mobile malware detection techniques. *IEICE Trans. Inf. Syst.* **2020**, *103*, 204–211. [CrossRef]
13. Meddeb, M.; Dhraief, A.; Belghith, A.; Monteil, T.; Drira, K.; Al-Ahmadi, S. Named data networking: A promising architecture for the Internet of Things (IoT). *Int. J. Semant. Web Inf. Syst.* **2018**, *14*, 86–112. [CrossRef]
14. Boukhalfa, S.; Amine, A.; Hamou, R.M. Border Security and Surveillance System Using IoT. *Int. J. Inf. Retr. Res.* **2022**, *12*, 21. [CrossRef]
15. Gomes, T.; Salgado, F.; Tavares, A.; Cabral, J. Cute mote, a customizable and trustable end-device for the internet of things. *IEEE Sens. J.* **2017**, *17*, 6816–6824. [CrossRef]

16. Choi, J.; Kim, Y. An improved LEA block encryption algorithm to prevent side-channel attack in the IoT system. In Proceedings of the 2016 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA), Jeju, Korea, 13–15 December 2016; pp. 1–4.
17. Shukla, P. ML-IDS: A machine learning approach to detect wormhole attacks in Internet of Things. In Proceedings of the 2017 Intelligent Systems Conference, IntelliSys 2017, London, UK, 7–8 September 2018; pp. 234–240. [CrossRef]
18. Yadav, C.S.; Sharan, A. Feature learning using random forest and binary logistic regression for ATDS. In *Applications of Machine Learning*; Springer: Cham, Switzerland, 2020; pp. 341–352.
19. Yadav, M.; Verma, V.K.; Yadav, C.S.; Verma, J.K. MLPGI: Multilayer perceptron-based gender identification over voice samples in supervised machine learning. In *Applications of Machine Learning*; Springer: Cham, Switzerland, 2020; pp. 353–364.
20. Singh, M.; Rajan, M.A.; Shivraj, V.L.; Balamuralidhar, P. Secure MQTT for Internet of Things (IoT). In Proceedings of the 2015 5th International Conference on Communication Systems and Network Technologies, CSNT 2015, Gwalior, India, 4–6 April 2015; pp. 746–751. [CrossRef]
21. Liu, C.; Cronin, P.; Yang, C. A mutual auditing framework to protect IoT against hardware Trojans. In Proceedings of the Asia and South Pacific Design Automation Conference, ASP-DAC, Macao, China, 25–28 January 2016; pp. 69–74. [CrossRef]
22. Konigsmark, S.T.C.; Chen, D.; Wong, M.D.F. Information dispersion for trojan defense through high-level synthesis. In Proceedings of the 2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC), Austin, TX, USA, 5–9 June 2016. [CrossRef]
23. Song, T.; Li, R.; Mei, B.; Yu, J.; Xing, X.; Cheng, X. A privacy preserving communication protocol for IoT applications in smart homes. *IEEE Internet Things J.* **2017**, *4*, 1844–1852. [CrossRef]
24. Machado, C.; Frohlich, A.A. IoT data integrity verification for cyber-physical systems using blockchain. In Proceedings of the 2018 IEEE 21st International Symposium on Real-Time Computing, ISORC 2018, Singapore, 29–31 May 2018; pp. 83–90. [CrossRef]
25. Qiao, M.; Sung, A.H.; Liu, Q. Merging permission and api features for android malware detection. In Proceedings of the 2016 5th IIAI International Congress on Advanced Applied Informatics, IIAI-AAI 2016, Kumamoto, Japan, 10–14 July 2016; pp. 566–571. [CrossRef]
26. Sarma, B.; Li, N.; Gates, C.; Potharaju, R.; Nita-Rotaru, C.; Molloy, I. Android permissions: A perspective combining risks and benefits. In Proceedings of the ACM Symposium on Access Control Models and Technologies, SACMAT, Newark, NJ, USA, 20–22 June 2012; pp. 13–22. [CrossRef]
27. Peng, H.; Gates, C.; Sarma, B.; Li, N.; Qi, Y.; Potharaju, R.; Nita-Rotaru, C.; Molloy, I. Using probabilistic generative models for ranking risks of Android apps. In Proceedings of the ACM Conference on Computer and Communications Security, Raleigh, NC, USA, 16–18 October 2012; pp. 241–252. [CrossRef]
28. Xiao, X.; Zhang, S.; Mercaldo, F.; Hu, G.; Sangaiah, A.K. Android malware detection based on system call sequences and LSTM. *Multimed. Tools Appl.* **2019**, *78*, 3979–3999. [CrossRef]
29. Chaba, S.; Kumar, R.; Pant, R.; Dave, M. Malware detection approach for android systems using system call logs. *arXiv* **2017**, arXiv:1709.08805.
30. Canfora, G.; Mercaldo, F.; Medvet, E.; Visaggio, C.A. Detecting Android malware using sequences of system calls. In Proceedings of the 3rd International Workshop on Software Development Lifecycle for Mobile, DeMobile 2015—Proceedings, Bergamo, Italy, 31 August 2015; pp. 13–20. [CrossRef]
31. Potharaju, R.; Newell, A.; Nita-Rotaru, C.; Zhang, X. Plagiarizing smartphone applications: Attack strategies and defense techniques. In Proceedings of the International symposium on engineering secure software and systems, Eindhoven, The Netherlands, 16–17 February 2012; pp. 106–120.
32. Wen, L.; Yu, H. An Android malware detection system based on machine learning. *AIP Conf. Proc.* **2017**, *1864*, 020136. [CrossRef]
33. Yerima, S.Y.; Alzaylaee, M.K.; Shajan, A. Deep learning techniques for android botnet detection. *Electronics* **2021**, *10*, 519. [CrossRef]
34. Yerima, S.Y.; Bashar, A. A Novel Android Botnet Detection System Using Image-Based and Manifest File Features. *Electronics* **2022**, *11*, 486. [CrossRef]
35. Goswami, A.; Sharma, D.; Mathuku, H.; Gangadharan, S.M.P.; Yadav, C.S.; Sahu, S.K.; Pradhan, M.K.; Singh, J.; Imran, H. Change Detection in Remote Sensing Image Data Comparing Algebraic and Machine Learning Methods. *Electronics* **2022**, *11*, 431. [CrossRef]
36. Portokalidis, G.; Homburg, P.; Anagnostakis, K.; Bos, H. Paranoid android: Versatile protection for smartphones. In Proceedings of the Annual Computer Security Applications Conference, ACSAC, Austin, TX, USA, 6–10 December 2010; pp. 347–356. [CrossRef]
37. Del Alamo, J.M.; Guaman, D.; Balmori, B.; Diez, A. Privacy Assessment in Android Apps: A Systematic Mapping Study. *Electronics* **2021**, *10*, 1999. [CrossRef]
38. Medhi, S.; Bora, A.; Bezboruah, T. Security Impact on e-ATM Windows Communication Foundation Services using Certificate based Authentication and Protection: An implementation of Message Level Security based on. NET Technique. *Int. J. Inf. Retr. Res.* **2016**, *6*, 37–51. [CrossRef]
39. Sengan, S.; Khalaf, O.I.; Sharma, D.K.; Hamad, A.A. Secured and privacy-based IDS for healthcare systems on E-medical data using machine learning approach. *Int. J. Reliab. Qual. E-Healthc.* **2022**, *11*, 1–11. [CrossRef]