electronics

MDPI

Article

Cracking the Core: Hardware Vulnerabilities in Android Devices Unveiled

Antonio Muñoz

Network, Information and Computer Security Lab (NICS), University of Malaga, 29071 Malaga, Spain; anto@uma.es

Abstract: As Android devices become more prevalent, their security risks extend beyond software vulnerabilities to include critical hardware weaknesses. This paper provides a comprehensive and systematic review of hardware-related vulnerabilities in Android systems, which can bypass even the most sophisticated software defenses. We compile and analyze an extensive range of reported vulnerabilities, introducing a novel categorization framework to facilitate a deeper understanding of these risks, classified by affected hardware components, vulnerability type, and the potential impact on system security. The paper addresses key areas such as memory management flaws, side-channel attacks, insecure system-on-chip (SoC) resource allocation, and cryptographic vulnerabilities. In addition, it examines feasible countermeasures, including hardware-backed encryption, secure boot mechanisms, and trusted execution environments (TEEs), to mitigate the risks posed by these hardware threats. By contextualizing hardware vulnerabilities within the broader security architecture of Android devices, this review emphasizes the importance of hardware security in ensuring system integrity and resilience. The findings serve as a valuable resource for both researchers and security professionals, offering insights into the development of more robust defenses against the emerging hardware-based threats faced by Android devices.

Keywords: Android hardware vulnerabilities; mobile device security; system-on-chip (SoC) security; trusted execution environment (TEE); hardware vulnerability mitigations; side-channel attacks

check for updates

Citation: Muñoz, A. Cracking the Core: Hardware Vulnerabilities in Android Devices Unveiled. *Electronics* **2024**, *13*, 4269. https://doi.org/10.3390/electronics13214269

Academic Editors: Xiangjie Kong and Giovanni Pau

Received: 3 October 2024
Revised: 28 October 2024
Accepted: 29 October 2024
Published: 31 October 2024

Copyright: © 2024 by the author. 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

In the current technological landscape, the security of mobile devices is a critical concern, particularly as they are increasingly used in sectors such as healthcare, finance, and government, where sensitive data and system integrity are paramount. Operating systems (OSs), which act as the backbone of these devices, play a vital role in ensuring their functionality and security. This paper surveys attacks on Android hardware; the distribution trends are only motivation to look into Android from June 2023 to June 2024, focusing on their market shares and the implications of these trends on the broader digital ecosystem.

According to Statcounter data [1], shown in Figure 1, Android remains the dominant operating system, with a market share of 40.17% in June 2023, although it declined slightly to 36.82% by June 2024. This dominance underscores the importance of studying Android's security architecture, especially given its widespread use in mobile and tablet devices across numerous industries. Other operating systems, such as Windows, iOS, OS X, and Linux, exhibit varying trends but do not rival Android's pervasive presence, particularly in mobile sectors.

While software vulnerabilities have been extensively explored, recent developments highlight that hardware vulnerabilities are becoming increasingly relevant. This is particularly true as mobile devices are now integral to critical sectors like healthcare and finance, where any security breach could have significant consequences. Hardware vulnerabilities differ fundamentally from software vulnerabilities in that they target the physical

components of a device, often bypassing software-based defenses entirely. These vulnerabilities can compromise the entire security model of a device, leading to more severe and persistent breaches.

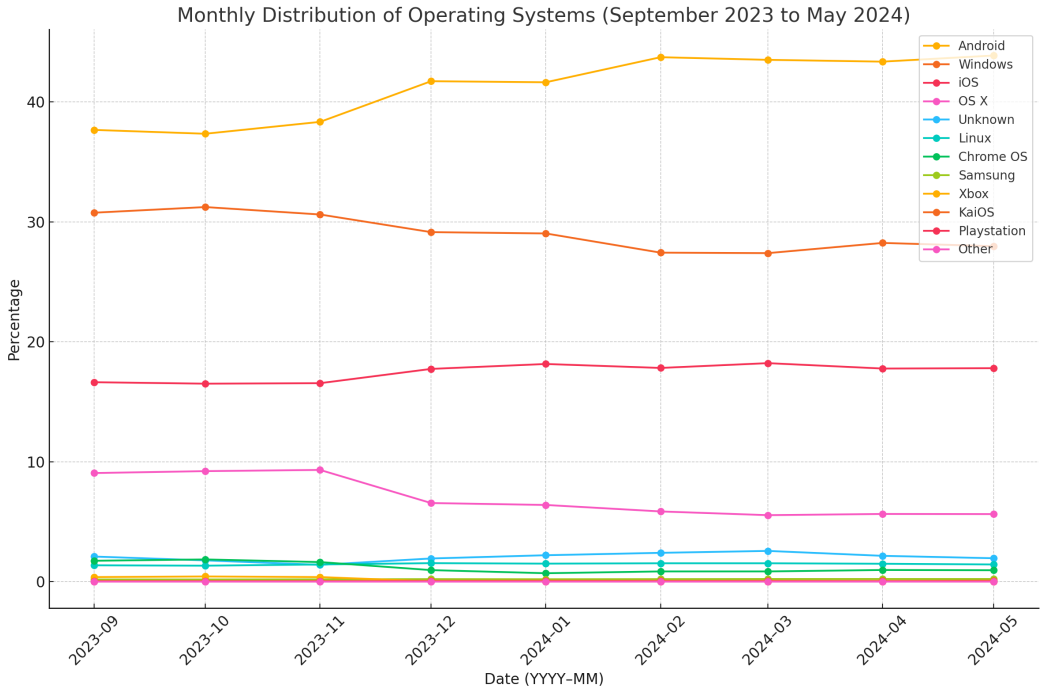


Figure 1. Monthly distribution of operating systems.

Recent advances in mobile security research have highlighted the critical role of defensive system architectures that integrate various hardware components to significantly enhance the security posture of mobile devices, particularly Android systems. These architectures go beyond software-based security measures by leveraging hardware-enforced protections that address vulnerabilities at a more fundamental level. In the context of mobile devices, hardware components such as secure boot, TEE, and hardware-backed encryption mechanisms play a pivotal role in safeguarding the integrity of the device from a wide array of sophisticated attacks.

Secure boot [2] is one of the foundational layers of these architectures. It ensures that only verified and cryptographically signed software is allowed to run during the boot process. This prevents attackers from tampering with the bootloader [3] or injecting malicious code at a critical stage in the device's operation [4]. By validating the integrity of the system at startup, secure boot mitigates the risk of rootkits and other persistent threats that can compromise system security at the hardware level.

In addition, TEE provides a secure area of the processor that isolates sensitive operations from the main operating system. TEEs are particularly effective in protecting cryptographic key management, authentication, and other critical security functions. By ensuring that these operations are handled in a secure, isolated environment, TEEs reduce the attack surface and prevent unauthorized access to sensitive data, even in the presence of malware or other threats in the main OS. Studies like [5,6] demonstrate how combining secure boot with TEE not only ensures secure boot processes but also isolates critical functions to enhance overall system security.

Another key aspect of these defensive architectures is hardware-backed encryption [7], which ensures that encryption keys are stored and processed in a secure hardware environment. Full-Disk Encryption (FDE) [8] and File-Based Encryption (FBE) [9] are commonly employed techniques that utilize hardware-backed security to protect user data at rest. These encryption mechanisms ensure that even if an attacker gains physical access to a device, they cannot decrypt sensitive data without the proper hardware-protected

keys. Research has shown that hardware-based encryption can significantly reduce the risk of data leakage, even in scenarios where attackers attempt to bypass software-based security controls.

By integrating multiple hardware components into a cohesive security architecture, these systems create a multi-layered defense that is more robust than traditional software-only approaches [10]. This layered defense is particularly effective in mitigating complex attack vectors such as side-channel attacks, memory corruption exploits, and unauthorized access to system resources. Side-channel attacks, for example, often target the physical properties of a device, such as its power consumption or electromagnetic emissions, to infer sensitive information. Hardware-backed encryption and TEE technologies can obscure these physical signals, making it more difficult for attackers to extract useful information through side-channel analysis [11]. Similarly, memory corruption vulnerabilities, which are often exploited in software attacks, are harder to leverage when hardware-level protections, such as memory isolation and secure memory management, are in place.

In comparison, software vulnerabilities typically involve flaws in applications or the operating system itself and are generally easier to patch through updates or security fixes. However, hardware vulnerabilities exploit weaknesses in the device's physical components, such as processors, memory, or communication modules. These attacks often have a broader impact as they can bypass multiple layers of software protection, rendering traditional security measures ineffective. As a result, the attack surface exposed by hardware vulnerabilities is significantly larger and more difficult to defend compared to software vulnerabilities.

Given the rise of hardware-based attacks and the increasing reliance on mobile devices in high-stakes environments, it is essential to focus on these types of vulnerabilities. Recent research has started acknowledging the importance of hardware security, but it remains fragmented, and the categorization and understanding of these vulnerabilities are still in their early stages.

The contributions of this paper are threefold:

1. *Comprehensive Cataloging*: We provide an exhaustive list of the hardware-related vulnerabilities reported and published to date. Each vulnerability is described in detail, highlighting its impact on Android devices. This catalog offers a valuable resource for researchers and practitioners to understand the growing range and severity of hardware vulnerabilities in Android devices.
2. *Proprietary Categorization*: Based on thorough analysis, we introduce a novel framework for categorizing these vulnerabilities. This framework is designed to differentiate hardware vulnerabilities from their software counterparts, focusing on the specific hardware component affected, the nature of the vulnerability, and its potential impact on the overall security architecture. This structured approach facilitates the systematic prioritization of hardware vulnerabilities.
3. *Countermeasures and Contextualization*: For each identified hardware vulnerability, we discuss feasible countermeasures and solutions. Furthermore, we provide a detailed description of the security mechanisms in Android devices to contextualize the vulnerabilities within the broader security landscape.

The structure of this paper is as follows: Section 2 provides an overview of Android architecture and security features, including the OS layers, key security mechanisms, and their evolution. Section 3 presents a comprehensive literature review, highlighting the most significant related works and clarifying our contribution. In Section 4, the main security challenges in mobile devices are discussed. Section 5 outlines the methodology for this systematic review, detailing the data sources, selection criteria, and the categorization of vulnerabilities. Section 6 shows how AI techniques enhance hardware vulnerability detection in the IoT by enabling efficient data analysis, prediction, and automated documentation. Section 7 focuses on hardware attacks specific to Android devices. Section 8 presents the results, including a detailed list of hardware-related vulnerabilities. A categorization

of these vulnerabilities is provided in Section 9. Finally, Section 10 summarizes the key findings and proposes future research directions.

2. Background

To thoroughly address the security vulnerabilities in Android devices, it is essential to first understand the architecture and security mechanisms that underpin the platform. This section highlights the key components of the Android OS [12], with a particular focus on hardware-related aspects such as secure boot, TEE [13], and hardware-based encryption, which play a critical role in defending against hardware-level attacks.

2.1. Android Architecture and Security Features

Android is built on the Linux kernel, providing the necessary foundation for resource management, including hardware drivers and security enforcement. However, the architecture's upper layers, such as the Android Runtime and application framework, rely on core hardware security mechanisms to protect the system from low-level attacks.

One of the key elements of Android's hardware security model is the implementation of secure boot. Secure boot ensures that only verified and authorized software is loaded during the boot process, preventing attackers from injecting malicious code at the hardware level. This process, managed through cryptographic signatures, safeguards the integrity of the boot sequence.

In addition, modern Android devices employ a TEE, a secure area of the main processor that runs isolated code and secure applications. This ensures that sensitive operations, such as authentication and cryptographic key management, are executed securely, separated from the main operating system. This isolation aims to reduce the risk of attacks that attempt to exploit the Android OS from a hardware level. However, the use of TEE is not without vulnerabilities. As highlighted in Muñoz et al. [14], a wide array of vulnerabilities have been identified within TEEs themselves, exposing them to various forms of attacks. These vulnerabilities underscore that, despite its security promises, the TEE can still be compromised, challenging its role as an impenetrable barrier in the security architecture.

Hardware-based encryption is another critical feature in Android's security architecture. FDE and FBE use hardware keys to protect user data at rest, ensuring that even if an attacker gains physical access to the device, the data remains encrypted and inaccessible without the appropriate decryption keys.

In addition to hardware and software defenses, system-level defenses play a crucial role in maintaining the confidentiality of communications within Android devices. These defenses involve deploying security protocols that ensure secure data handling across all layers of operation. For example, in high-security environments, system-level measures on Android may include secure deployment configurations that use dedicated hardware modules, secure enclave technology, and trusted third-party applications to enforce data integrity and confidentiality. Research into system-wide defenses, particularly for applications handling sensitive communications, is a growing area of focus, aiming to create frameworks that safeguard communications from unauthorized access even in hostile network environments.

2.2. Summary of Android OS Layers

While the Android OS is composed of several layers, including the application layer, application framework, and Android Runtime, this paper focuses on the aspects of the system most relevant to hardware vulnerabilities. The Dalvik Virtual Machine (DVM) [15] and its successor, Android Runtime (ART) [16], provide an environment where applications run within isolated sandboxes, leveraging hardware features to enforce security. Additionally, Android's middleware and native libraries interact closely with hardware components, such as processors and sensors, which are potential targets for hardware attacks.

Given the importance of hardware in Android's security posture, vulnerabilities within these layers, especially those that bypass software defenses, must be addressed

comprehensively. Therefore, a detailed understanding of these layers can be found at [17] to contextualize the hardware vulnerabilities.

2.3. Key Security Mechanisms in Android

Android's security architecture incorporates several mechanisms designed to protect both hardware and software layers:

- *Secure Boot*: Ensures that only trusted software is loaded during the boot process, preventing unauthorized firmware modifications.
- *Trusted Execution Environment*: Isolates sensitive operations, such as cryptographic key management and authentication, providing a secure execution space separate from the main OS.
- *Hardware-Based Encryption (HBE)*: FDE and FBE rely on hardware keys to secure user data, making it resistant to attacks even if the physical device is compromised.

Additionally, application sandboxing ensures that applications run in isolated environments, reducing the potential for one compromised app to affect the entire system. The Inter-Component Communication (ICC) [18] mechanism facilitates communication between applications while enforcing strict access control, further limiting the attack surface.

2.4. Evolution of Security Patches in Android

Security patches play a vital role in addressing both software and hardware vulnerabilities in Android devices. While software vulnerabilities can often be mitigated through regular patches, hardware vulnerabilities may require more extensive solutions, such as firmware updates or modifications to hardware components. Since the introduction of monthly security bulletins in 2015, Google has maintained a rigorous patching process, with increased emphasis on addressing vulnerabilities that arise from hardware flaws.

Project Treble [19], introduced in Android 8.0 Oreo, further streamlined the update process by separating the Android OS from vendor-specific hardware implementations. This architectural shift has improved the speed and consistency of security updates, particularly those addressing hardware-related issues. Continued efforts in this direction, alongside enhancements in hardware design and security mechanisms, are essential to mitigate future vulnerabilities.

The security of Android devices is inherently tied to both software and hardware defenses. By emphasizing secure boot, TEE, and HBE, Android's architecture provides strong protections against hardware-level attacks. However, as mobile devices continue to play a critical role in sectors like healthcare and finance, addressing hardware vulnerabilities must remain a priority, especially as these attacks have the potential to bypass even the most advanced software defenses.

3. Literature Review

This section provides a systematic review of the existing literature on hardware-related vulnerabilities in Android devices. Our work builds upon prior research by categorizing and analyzing specific hardware vulnerabilities, proposing countermeasures, and offering a detailed examination of major Android devices and their existing security mechanisms. Additionally, we reveal the results of our survey on the prevalence and nature of hardware vulnerabilities, providing quantitative insights that support our analysis. This approach ensures that our analysis is both self-contained and contextualized within the broader security landscape.

Sharma et al. [20] present a comprehensive review of malicious application detection on Android devices. They categorize malware detection techniques into static, dynamic, and hybrid methods, identifying dynamic analysis as the most effective for detecting runtime malicious behavior. However, their review focuses on software vulnerabilities, whereas our study addresses the gap by specifically analyzing hardware vulnerabilities in Android devices, providing a distinct and complementary perspective.

Sutter et al. [21] conduct a systematic review of dynamic security analysis in Android, focusing on methods for dynamically detecting security issues in applications. Their work provides a comprehensive analysis of dynamic methodologies but does not extend to hardware vulnerabilities. Our study complements theirs by presenting an in-depth analysis of hardware vulnerabilities, thus broadening the scope of Android security to encompass both software and hardware aspects.

Senanayake et al. [22] review vulnerability detection at the source code level, emphasizing the importance of early detection during development. While they focus on identifying software vulnerabilities during coding, our work expands this perspective by addressing hardware vulnerabilities, which can pose significant security risks even in well-written software.

Garg et al. [23] examine trends in Android security assessments, highlighting gaps in existing methodologies and the lack of attention to hardware vulnerabilities. Our work builds on their insights by providing an in-depth analysis of hardware vulnerabilities and their corresponding mitigation strategies, thereby addressing these overlooked areas in Android security.

Abdullah et al. [24] review the growing number of vulnerabilities and attacks in Android mobile applications as the use of mobile apps increases. While their work focuses on prevention techniques for software vulnerabilities, our research provides a targeted analysis of hardware vulnerabilities, which is essential for ensuring long-term Android device security.

Visoottiviseth et al. [25] focus on mobile application security assessment tools, identifying a lack of security awareness among developers as a key cause of vulnerabilities. They highlight the importance of educational tools like MASai [26] for addressing software vulnerabilities. In contrast, our study emphasizes hardware vulnerabilities and their mitigation through targeted countermeasures at the hardware level.

Rahman et al. [27] discuss predicting security and privacy risks in Android applications through static code analysis, arguing that bad coding practices often lead to software vulnerabilities. While their study addresses software risks, our research contributes by focusing on hardware vulnerabilities, thus providing a more holistic view of Android security.

Watanabe et al. [28] investigate the origins of vulnerabilities in mobile applications, especially those arising from third-party libraries, which they show can introduce significant security risks. Building on their findings, our study broadens the focus by analyzing hardware vulnerabilities and proposing detailed descriptions and countermeasures for these threats.

Wu et al. [29] make a significant contribution by introducing an automatic framework for vulnerability analysis focused on Android system-level vulnerabilities. Through examining 2179 vulnerabilities and their patches, they reveal that 92% of vulnerabilities are concentrated in low-level modules, such as native libraries and kernel drivers. This underscores the importance of addressing hardware vulnerabilities, a focus shared by our study. By incorporating their vulnerability clustering techniques, we extend the analysis to propose hardware-specific mitigation strategies.

Linares-Vásquez et al. [30] conduct a large-scale empirical analysis of Android vulnerabilities, focusing on those affecting the Android OS. They develop a taxonomy of Android vulnerabilities based on the Common Weakness Enumeration (CWE) [31] and examine the survivability of vulnerabilities from introduction to resolution. Their findings reveal that most vulnerabilities reside in low-level components such as drivers and native libraries, which aligns with our analysis emphasizing the critical nature of hardware vulnerabilities. Additionally, they note that hardware driver vulnerabilities can persist in systems for an extended period, with an average survivability of 724 days. Our study complements theirs by focusing specifically on hardware-level threats and proposing targeted solutions.

The existing literature provides a wealth of knowledge on software vulnerabilities, malware detection, and dynamic security analysis in Android. However, there is a notable

gap concerning hardware vulnerabilities. Our study fills this gap by conducting a comprehensive review of hardware vulnerabilities, categorizing them based on relevant criteria, and proposing viable countermeasures. These contributions are crucial for advancing the overall security posture of Android devices, addressing both software and hardware vulnerabilities within a unified framework.

4. Security Challenges in Mobile Devices

Mobile devices, spanning from low-cost to high-end smartphones, face a diverse array of security challenges stemming from variations in hardware quality, software support, and manufacturing practices. These differences critically influence their susceptibility to hardware-related vulnerabilities, which can significantly compromise the security of the devices. Understanding these issues is essential for developing robust countermeasures and ensuring the secure operation of mobile devices. This section presents an in-depth analysis of hardware vulnerabilities in low-end and mid-range devices, as well as the growing impact of widely accessible hacking tools that exploit these weaknesses.

4.1. Low-End Devices

Low-end smartphones are particularly vulnerable to hardware-related security risks, primarily due to cost-reduction measures that compromise the integrity of the hardware. Several factors contribute to the increased security risks in these devices:

Insecure Hardware Components: Low-end smartphones often incorporate lower-quality hardware components, as observed in prior analyses of cost-driven manufacturing practices [32]. These components are more prone to hardware-level vulnerabilities, including insecure boot processes and weak memory management, a pattern reported in device quality assessments [33–35]. As a result, vulnerabilities such as buffer overflows and out-of-bounds memory accesses are prevalent, largely due to the absence of essential hardware protections, making these devices prime targets for adversarial attacks.

Inadequate Software Support and Updates: Low-end devices generally receive minimal software and security updates, often limited to two years or less [36]. This lack of support leaves them vulnerable to both existing and emerging hardware exploits. The lack of ongoing firmware updates for critical hardware components leaves them exposed to both known and emerging hardware exploits. Additionally, outdated bootloaders and insecure firmware persist in these devices, further exacerbating their vulnerability to attacks.

Limited Quality Assurance and Security Testing: Due to production cost constraints, low-end devices undergo limited security testing, which may contribute to design flaws such as weak secure boot implementations [37]. This results in persistent design flaws, such as weak secure boot implementations and inadequate hardware access control mechanisms. Furthermore, manufacturers frequently overlook the implementation of hardware-based security features, such as tamper-resistant modules, increasing the risk of physical attacks.

Performance Trade-offs Affecting Security: In cost-sensitive designs, performance optimizations can outweigh security measures, as observed in recent studies on hardware trade-offs [38]. This can lead to the weakening or removal of crucial security mechanisms, such as hardware-backed encryption, leaving the device susceptible to attacks that exploit these trade-offs. For instance, reduced cryptographic protections or the insufficient isolation of critical processes are common entry points for attackers.

Given these factors, low-end devices present a significant security risk within the mobile ecosystem. Users and developers must acknowledge these limitations and implement additional protective measures, such as incorporating hardware security modules (HSMs) [39] or TEEs, to mitigate these vulnerabilities where feasible.

4.2. Mid-Range Devices

Mid-range smartphones offer a balance between cost and performance but continue to face notable security challenges. Below, we highlight the primary factors influencing the security of mid-range devices:

Enhanced Hardware with Persistent Vulnerabilities: While mid-range devices employ higher-quality components than low-end models, specific vulnerabilities, such as in firmware security and wireless interfaces, remain prevalent [40,41]. Firmware vulnerabilities, insecure wireless protocols, and weaknesses in sensors like GPS and accelerometers are common in mid-range devices, as documented in prior reviews of device classes [42]. Despite more rigorous quality control measures, mid-range devices may still harbor exploitable flaws in areas such as secure boot and firmware security.

Software and Security Update Cycles: Mid-range devices typically receive updates for two to three years, a duration shorter than high-end devices, which can leave them more vulnerable to late-discovered hardware exploits [43]. However, this support period is shorter than that of high-end devices, making mid-range smartphones more vulnerable to hardware-related exploits discovered after the update cycle ends. Delays in addressing firmware vulnerabilities further extend this window of exposure, compounding the associated security risks.

Security Testing and Quality Control: Although mid-range devices experience more rigorous testing than low-end models, they may still lack the comprehensive security evaluations that are characteristic of high-end models [41]. This gap can result in unresolved design flaws, such as inadequate access control for critical hardware registers. Moreover, concerns remain regarding vulnerabilities to side-channel attacks and sensor spoofing, especially in components that do not incorporate the advanced security measures typically found in premium devices. These issues highlight the need for ongoing improvements to the security protocols applied to mid-range hardware.

Performance–Security Trade-offs: Although capable of supporting advanced security measures, mid-range devices may omit features like hardware-backed key storage due to cost limitations [44]. Critical components, such as hardware-backed key storage or tamper-resistant cryptographic modules, are frequently limited to premium models. Consequently, mid-range smartphones remain susceptible to sophisticated attacks, particularly those targeting wireless communication protocols or exploiting physical access vulnerabilities.

Despite offering enhanced security relative to low-end models, mid-range smartphones still require careful attention, particularly in maintaining timely firmware updates and securing essential hardware interfaces. It is imperative to leverage all available hardware security features and ensure consistent firmware patching to effectively mitigate potential security risks.

4.3. High-Range Devices

High-range mobile devices, typically flagship models, incorporate sophisticated hardware and security features compared to low- and mid-range devices. These high-end devices benefit from advanced technologies such as hardware-backed key storage, tamper-resistant modules, and highly secure trusted execution environments (TEEs). Common features like secure boot and full hardware-level encryption further bolster their defenses against unauthorized access at the hardware level. Despite these enhancements, several factors continue to challenge the security of high-end devices.

Complex Hardware Architectures with Increased Attack Surface: High-range devices integrate multiple, intricate hardware layers that, while enhancing performance and security, also introduce additional attack vectors. Vulnerabilities in trusted components (e.g., secure enclaves) and side-channel attack exposure can undermine the perceived security of these devices, allowing attackers to target elements previously thought to be secure.

Prolonged Software and Hardware Update Cycles: High-end devices typically benefit from extended software and firmware support cycles compared to lower tier models. However, delays in addressing hardware vulnerabilities, particularly in critical components like baseband processors or dedicated security chips, may leave devices exposed for long periods. This prolonged exposure is especially concerning in environments where these devices store sensitive data, such as corporate or governmental contexts.

Security and Quality Assurance Gaps in Complex Systems: While high-range devices undergo rigorous testing, the increased complexity of their hardware and software ecosystems can make it difficult to identify and patch all security flaws. Even with advanced quality assurance practices, vulnerabilities in secure boot processes, firmware, and hardware interfaces can persist, leading to potential exploits that undermine the overall security architecture.

Performance–Security Trade-offs in Advanced Features: To maintain optimal performance, high-end devices may face trade-offs in implementing certain security measures. While they often support hardware-backed encryption and advanced key management, maintaining high performance can limit the depth of these protections. For instance, features such as always-on connectivity or high-speed processing can introduce subtle vulnerabilities that attackers could exploit, especially in high-risk environments.

4.4. Accessible Hacking Tools

The increasing accessibility of low-cost hacking tools represents a significant challenge to the security of mobile devices. Many of these tools can be easily acquired through public channels, allowing even individuals with limited technical expertise to exploit the vulnerabilities present in both low-end and mid-range smartphones. This growing availability lowers the entry barrier for attackers, increasing the risk of compromise. Below, we provide an analysis of some of the most widely utilized hacking tools and their potential impact on mobile security, with a focus on how these tools facilitate the exploitation of specific vulnerabilities.

Rubber Ducky USB (https://shop.hak5.org/products/usb-rubber-ducky (accessed on 2 October 2024)), *Bash Bunny* (https://shop.hak5.org/products/bash-bunny (accessed on 2 October 2024)), *and Flipper Zero* (https://shop.flipperzero.one/ (accessed on 2 October 2024)): These devices exploit vulnerabilities in unsecured USB interfaces by injecting malicious payloads or manipulating data during the boot process. Low-end devices, with outdated firmware and inadequate USB security measures, are particularly vulnerable to these attacks. Mid-range devices are also at risk if they lack strict USB interface access controls or secure boot configurations.

Proxmark3 (https://github.com/RfidResearchGroup/proxmark3/ (accessed on 2 October 2024)) *and WiFi Pineapple* (https://shop.hak5.org/products/wifi-pineapple (accessed on 2 October 2024)): Proxmark3 [45] targets poorly secured NFC chips, which are often inadequately protected in both low-end and mid-range devices. Similarly, WiFi Pineapple exploits insecure wireless protocols, such as weakly implemented Wi-Fi or Bluetooth, to conduct man-in-the-middle attacks or intercept sensitive data. Devices without robust wireless encryption or secure key management are especially vulnerable to these tools.

OMG Cable: The OMG Cable (https://docs.hak5.org/omg-cable (accessed on 2 October 2024)), a maliciously modified USB cable, can exploit insecure bootloaders or access firmware-level functionality in mobile devices. Low-end devices, often running outdated or insecure firmware, are particularly at risk, as they lack protections, such as firmware integrity checks or USB port restrictions.

The accessibility of these hacking tools underscores the urgent need for stronger hardware security in mobile devices. Both hardware and software vulnerabilities must be addressed in a unified manner to prevent exploitation by these readily available tools. Manufacturers should prioritize the integration of hardware-backed security features, such as secure boot and hardware-level encryption, across all device tiers to mitigate these risks effectively.

Note on Sources and Insights: The insights presented in this section are based on a combination of the existing literature, practical observations, and the experience of the authors in the field of mobile security. Where applicable, references to the relevant literature have been included to support specific claims, while other general statements reflect the analysis and synthesis of knowledge gathered from industry practice and research expertise.

5. Methodology

This study implements a systematic review to identify and analyze hardware-related vulnerabilities in Android devices. The methodology was designed with a focus on thoroughness, rigor, and adherence to established guidelines in information security research. A multi-stage approach was employed for identifying and selecting vulnerabilities, combining comprehensive searches across both academic and public databases, followed by the application of stringent inclusion and exclusion criteria, culminating in a detailed analysis of each identified vulnerability.

The search strategy focused on identifying relevant studies in leading academic repositories, including IEEE Xplore, ACM Digital Library, SpringerLink, and ScienceDirect. A precise combination of keywords was employed, such as 'Android hardware vulnerabilities', 'mobile device security', 'hardware exploits', and 'Android security weaknesses', to ensure the retrieval of literature addressing critical aspects of hardware security in Android devices. Simultaneously, public vulnerability repositories such as the National Vulnerability Database (NVD) [46] and Common Vulnerabilities and Exposures (CVE) [47] details were consulted to capture documented vulnerabilities reported in real-world scenarios. This dual-source strategy ensured a robust dataset that integrates both academic insights and practical vulnerability reports.

In our methodology, we have conducted a comprehensive study utilizing a wide variety of well-established sources to analyze vulnerabilities in Android systems, including hardware, firmware, and TrustZone components. This methodology has been structured around key sources such as CWE, CVE [48], NVD [46], Exploit-DB [49], VULDB [50], CERT [51], IEEE, ACM, Springer, and ScienceDirect, to ensure a robust understanding of the vulnerabilities from multiple perspectives.

The CWE (Common Weakness Enumeration) database was used to identify vulnerabilities specifically associated with TrustZone, Android hardware, and Android firmware. For example, CWE provided detailed records on TrustZone vulnerabilities with a total of 3 specific findings and 219 references regarding Android hardware, making it a critical reference for understanding weaknesses at the system-on-a-chip (SoC) level.

Furthermore, the CVE (Common Vulnerabilities and Exposures) entries allowed us to delve into Android vulnerabilities with comprehensive quantitative data, totaling 8657 records for Android and 18 for Android hardware, among others. The CVE-2016-2454 reference, for instance, is particularly noteworthy as it exemplifies how identified issues are linked to hardware components. These datasets help in understanding the depth and variety of threats affecting different Android components.

The NVD (National Vulnerability Database) contributed a total of 8555 records related to Android and its hardware vulnerabilities, while other specific sources such as Exploit-DB were also used to identify publicly known exploits. The VULDB database contributed with 14 TrustZone vulnerabilities, supporting the analysis with practical case examples of exploitations.

In addition to vulnerability databases, we incorporated data from well-respected academic and technical journals. IEEE, ACM, Springer, and ScienceDirect were used as primary academic sources, ensuring that our analysis included recent research developments. Notably, IEEE included a range of 52 conferences and 14 journals focused on Android hardware vulnerabilities, TrustZone, and firmware vulnerabilities, giving an academically rigorous background to our investigation.

To ensure the relevance and quality of the selected vulnerabilities, a set of inclusion and exclusion criteria was established. The inclusion criteria required that studies or reports be published between January 2010 and July 2024, specifically address hardware-related vulnerabilities in Android devices, and provide detailed technical descriptions. Reports focusing solely on software vulnerabilities, lacking technical depth, or duplicating previously reviewed entries were excluded. This process guaranteed that only high-quality and pertinent data were included in the analysis.

For a detailed analysis, several CWEs were selected, which include detailed descriptions and attack vectors, providing a clear understanding of the types of security weaknesses and their corresponding exploitation mechanisms. Examples include CWE-1189 [52], which describes the improper isolation of shared resources in SoC, vulnerable to side-channel speculative execution, and CWE-1240 [53], related to cryptographic primitives implemented with risky methods. The attack vectors associated with these vulnerabilities—such as insufficiently protected credentials or the use of weak cryptographic protocols—offer critical insight into how such vulnerabilities could be exploited by adversaries.

The initial exploration phase included a comprehensive examination of all potentially relevant hardware vulnerabilities as listed in the Common Weakness Enumeration (CWE) database. The analysis began by considering a broad spectrum of vulnerabilities spanning various hardware components, security features, and system configurations. The CWE database provided an extensive list of weaknesses that allowed the team to map a general overview of existing and emerging issues within Android hardware systems.

However, to ensure a focused and effective analysis, it was essential to narrow down the vulnerabilities based on relevance, impact, and practicality in real-world Android environments. This led to a refined selection of vulnerabilities, specifically centered around those directly affecting hardware security in Android system-on-a-chip (SoC) components, cryptographic implementations, memory management, and protection mechanisms.

From the initial comprehensive list, we focused specifically on the following vulnerabilities, which are of high importance due to their potential impact on Android devices and their direct relation to hardware functionality and security controls, in Table 1.

The decision to focus on these specific vulnerabilities was driven by the desire to contribute meaningfully to the understanding of hardware vulnerabilities in Android environments that pose the most significant risks to security. These vulnerabilities were selected based on their high likelihood of exploitation, the severity of their potential impact, and the practical challenges they present in securing Android hardware and firmware. Furthermore, during the analysis phase, it was observed that many hardware attack scenarios referenced in the literature ultimately led back to these specific CWEs as critical weaknesses. For example, CWE-1189 [52] has been extensively discussed in sources such as "Challenges and Opportunities for Hardware-Assisted Security Improvements in the Field" (IEEE) [54], "Towards Reconfigurable Hardware for In-field Hardware Bug Patches" [55] (ScienceDirect), "IP-Tag: Tag-Based Runtime 3PIP Hardware Trojan Detection in SoC Platforms" [56] (IEEE), "Security Risks and Their Mitigation Strategies: Cloud Computing Perspective" [57] (IEEE), "Hardware Phi-1.5B: A Large Language Model Encodes Hardware Domain Specific Knowledge" [58] (ACM), and "Don't CWEAT It: Toward CWE Analysis Techniques in Early Stages of Hardware Design" [59] (ACM). These references underscore the critical nature of CWE-1189 [52] and similar vulnerabilities, reinforcing the decision to concentrate on these CWEs in our detailed analysis.

The approach taken ensured that, while the broad landscape of Android hardware vulnerabilities was initially considered, the final focus remained on those weaknesses that have critical implications for system integrity, data confidentiality, and operational reliability. By concentrating on these selected CWEs, this study aims to provide a deeper analysis and develop targeted mitigation strategies that could be adopted to enhance Android hardware security.

To further enhance the understanding of these vulnerabilities, a structured data extraction process was followed for each selected CWE. This included detailed descriptions, potential attack vectors, affected hardware components, and possible mitigations. The study's ultimate goal is to offer actionable insights that both the research community and security practitioners can leverage to mitigate hardware vulnerabilities effectively.

Following data extraction, the vulnerabilities were synthesized to provide an overarching view of the current hardware vulnerability landscape in Android devices. This synthesis included a detailed list of vulnerabilities along with impact assessments and recommended countermeasures. By integrating the extracted data, the study aims to offer

a clear, actionable understanding of the state of hardware security in Android devices, contributing valuable insights to both the research community and security practitioners.

Table 1. Key Hardware-level security weaknesses in android devices: threats and root causes.

CWE	Vulnerability Description	Forms of Attack	Underlying Causes
CWE-125 [60]	Out-of-bounds Read: Reading memory outside valid boundaries, potentially exposing sensitive data or causing system crashes.	Exploitation through user-supplied data exceeding buffer limits or invalid pointer access.	Incorrect array indexing, insecure memory management practices, invalid loop conditions.
CWE-416 [61]	Use After Free: Accessing memory after it has been freed, leading to data corruption or arbitrary code execution.	Maliciously crafted requests triggering reuse of freed memory.	Premature memory deallocation, dangling pointers, improper memory management.
CWE-1189 [52]	Improper SoC Resource Isolation: Failure to isolate system resources between trusted and untrusted entities, leading to security breaches.	Memory corruption, unauthorized access to peripherals, manipulation of system states.	Inadequate isolation mechanisms, poor implementation of resource management protocols.
CWE-1191 [62]	Improper Debug Interface Access: Lack of secure access control on debug and test interfaces, allowing unauthorized tampering with internal chip settings.	Exploiting exposed JTAG interfaces to modify registers or configuration settings.	Insufficient access control mechanisms, flawed implementation of debug access protocols.
CWE-1240 [53]	Risky Cryptographic Implementation: Cryptographic primitives implemented with weaknesses that can be exploited to compromise secure operations.	Exploiting weak encryption schemes or bypassing digital signature verifications.	Use of outdated or vulnerable cryptographic algorithms, incorrect implementation of cryptographic protocols.
CWE-1256 [63]	Software–Hardware Interface Flaws: Inadequate restrictions in software interfaces lead to unauthorized hardware access.	Exploiting unprotected interfaces to manipulate power management or access cryptographic registers.	Excessive software privileges, misconfigured access control mechanisms.
CWE-1272 [64]	Uncleared Debug/Power State: Failure to clear sensitive information before transitioning between debug or power states.	Forcing debug mode to access residual encryption keys or other sensitive data.	Improper state management routines, inadequate memory clearing protocols.
CWE-1274 [65]	Boot Code Protection: Inadequate protection of volatile memory storing boot code, allowing attackers to inject malicious code.	Overwriting legitimate boot code, introducing persistent malware at the boot level.	Lack of secure boot mechanisms, insufficient protection of boot memory regions.
CWE-1300 [66]	Physical Side-Channel Protection: Insufficient protection against physical side-channel attacks, allowing attackers to extract sensitive data through indirect means.	Power analysis, electromagnetic emissions, acoustic leakage used to extract cryptographic keys.	Lack of hardware mitigations, failure to integrate side-channel protection techniques.
CWE-1332 [67]	Fault Handling: Improper handling of hardware faults, allowing attackers to bypass critical instructions via glitches or faults.	Manipulating voltage or clock signals to skip critical security instructions.	Design flaws, absence of glitch detection circuits, inadequate fault tolerance mechanisms.

To ensure replicability, every stage of the methodology, from the initial search to the final categorization of vulnerabilities, was meticulously documented, including a detailed description of the selection process and justification for the specific vulnerabilities analyzed from the CWE database. A flowchart (see Figure 2) was included to visually represent the

key stages of the review process, enhancing both transparency and reproducibility. This structured approach ensures that the findings of this review can be reliably reproduced in future studies, thus providing a foundation for ongoing research in this critical area of Android device security.

By employing a systematic and rigorous approach, this study advances the understanding of hardware vulnerabilities in Android devices. The findings provide a valuable resource for researchers and practitioners in mobile security, highlighting critical vulnerabilities and suggesting potential mitigations. As new vulnerabilities continue to surface and the classification framework evolves, this work establishes a foundation for future efforts in enhancing the security of Android devices.

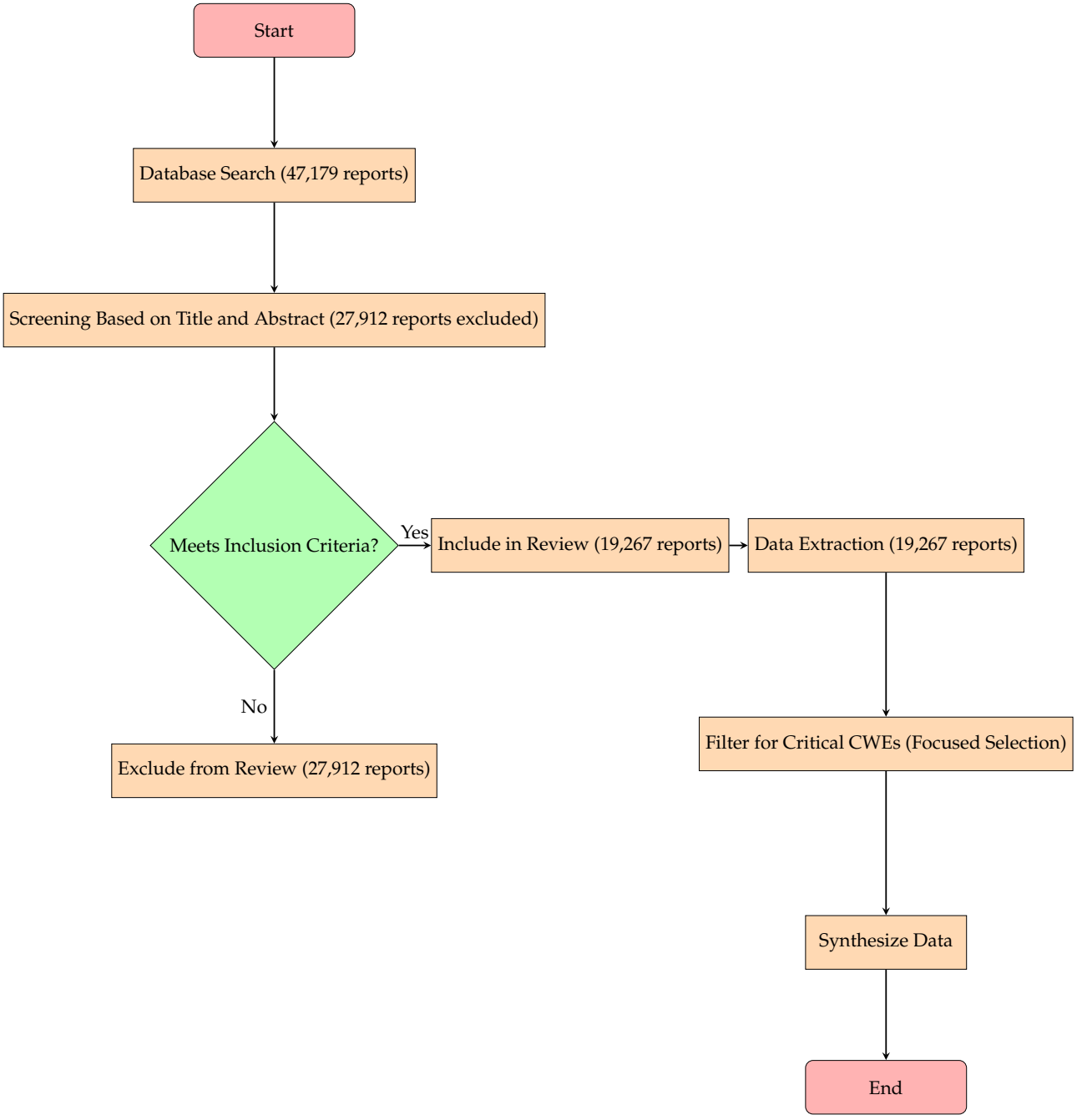


Figure 2. Flowchart of the vulnerability identification process, including the number of reports at each stage and the filtering for critical CWEs.

6. Leveraging Artificial Intelligence for the Detection of Hardware Vulnerabilities in the Context of IoT

The integration of artificial intelligence (AI) techniques in the identification and analysis of hardware vulnerabilities has gained significant attention, especially in the context of the Internet of Things (IoT). The vast quantity of data generated by scientific publications, vulnerability databases (such as CVEs and CWEs), and other reputable sources presents a unique opportunity for the deployment of AI-driven tools that can assist in the rapid and systematic documentation of potential threats to hardware. This section discusses how AI techniques can be effectively utilized to extract meaningful insights from these resources, ultimately enhancing the security posture of IoT devices.

To further validate the relevance and applicability of AI in the detection of hardware vulnerabilities, we conducted a comprehensive survey among industry professionals and researchers. The survey aimed to capture insights on the most critical types of hardware vulnerabilities and the perceived benefits of using AI for their detection and documentation. The survey results strongly support the implementation of AI-driven approaches, highlighting their role in enhancing security and efficiency in managing hardware threats. The following subsections expand on these findings and the use of AI techniques in various contexts.

6.1. Application of AI for Analyzing Scientific Literature and Databases

The growing body of scientific literature and the proliferation of public vulnerability databases provide a wealth of information regarding hardware security vulnerabilities. However, the volume and complexity of such data often make manual analysis impractical. In this context, Natural Language Processing (NLP) techniques, powered by AI, offer a powerful solution to automate the review and synthesis of information. Specifically, AI models can be trained to scan through scientific publications, security advisories, and vulnerability databases like Common Vulnerabilities and Exposures (CVE) and Common Weakness Enumeration (CWE), extracting actionable information regarding emerging threats.

Survey respondents overwhelmingly agreed that AI-based tools, particularly NLP techniques, are crucial in analyzing the sheer volume of available literature and databases. Nearly 80% of participants highlighted the challenge of keeping up with the ever-increasing number of scientific articles and advisories, reinforcing the need for automated AI-driven solutions. This strong endorsement from industry professionals underscores the practical value of AI tools in the field.

One example of an AI application is the use of NLP algorithms to perform entity recognition and categorization. These algorithms can identify relevant terms, such as "buffer overflow", "side-channel attack", or "system-on-chip vulnerabilities", and classify them according to established taxonomies. Machine learning models, such as Bidirectional Encoder Representations from Transformers (BERT), can be fine-tuned to recognize context-specific keywords and classify vulnerabilities based on their potential impact and affected components. By systematically analyzing text from scientific journals, technical reports, and vulnerability advisories, AI tools can create a comprehensive mapping of vulnerabilities and their relationships to specific hardware components in IoT devices.

6.2. Predictive Analysis and Pattern Recognition

AI techniques, particularly machine learning (ML), can be leveraged to detect patterns and predict potential hardware vulnerabilities by analyzing historical data. Using supervised and unsupervised learning models, AI can analyze CVE and CWE entries to identify trends that indicate the presence of certain types of vulnerabilities in hardware configurations. For example, clustering algorithms such as k-means or hierarchical clustering can be used to group vulnerabilities with similar characteristics, helping researchers and engineers understand how vulnerabilities propagate across different hardware platforms.

According to the survey results, 70% of respondents indicated that predictive analysis is a crucial area where AI can significantly enhance proactive security measures. Partic-

ipants noted that being able to anticipate potential vulnerabilities allows for the more effective allocation of resources, mitigating risks before they are exploited by attackers. This strong support highlights the importance of predictive AI tools for the continued evolution of hardware security.

Furthermore, the deployment of recurrent neural networks (RNNs) and other deep learning models can aid in forecasting the emergence of new vulnerabilities based on existing data. These models can analyze temporal trends within CVE datasets, providing insights into how certain vulnerabilities evolve over time, which components are most susceptible, and how attackers are likely to exploit these weaknesses. Such predictive capabilities are particularly valuable for IoT devices, where hardware security is often a critical concern due to the limited ability to apply regular security patches.

6.3. AI-Driven Automation in Vulnerability Documentation

Another significant benefit of using AI in the context of hardware vulnerability analysis is the automation of documentation processes. AI-driven tools can help to maintain up-to-date records of vulnerabilities, automatically generating summaries of key findings from newly published CVEs or research articles. Text summarization algorithms, utilizing techniques such as transformers or sequence-to-sequence models, can be employed to condense lengthy technical documents into concise reports that highlight the most critical information about a vulnerability, including its impact, severity, and mitigation strategies.

Survey participants emphasized the value of automation in reducing manual workload, with 85% agreeing that AI-driven automation could substantially reduce the time required to document and manage vulnerability records. This indicates a strong industry preference for incorporating AI tools that enhance operational efficiency, reduce human error, and ensure consistent documentation practices. By integrating information extracted from multiple sources, AI systems can provide a unified and comprehensive view of the current state of hardware vulnerabilities. This not only reduces the burden on cybersecurity professionals, but also ensures that vulnerability databases remain consistent and easily navigable. Moreover, AI-based text analysis tools can cross-reference newly detected vulnerabilities with known weaknesses, enabling a faster response to newly discovered threats by linking them to existing remediation strategies.

6.4. Challenges and Future Directions

Despite the potential advantages, several challenges must be addressed to fully leverage AI for vulnerability detection and documentation. One significant challenge is the availability of high-quality training datasets, as labeled data specific to hardware vulnerabilities is often scarce. Additionally, ensuring the accuracy of AI models in differentiating between subtle variations of vulnerabilities remains a complex task. Survey respondents also identified this as a key barrier, with 60% pointing to the need for improved data quality and collaboration between stakeholders to create more effective training sets.

Future research should focus on enhancing dataset quality through collaborative efforts between academic institutions, industry, and public organizations. Furthermore, explainable AI (XAI) approaches should be prioritized to enhance the interpretability of AI-driven vulnerability analyses. This is crucial for building trust in AI systems, especially among stakeholders in critical IoT applications, where understanding the rationale behind AI-generated insights is as important as the accuracy of the insights themselves.

7. Hardware Attacks on Android Devices

This section systematically categorizes and describes hardware vulnerabilities in Android devices, providing a technical understanding of each weakness without exploring mitigation strategies or detailed consequences.

Hardware vulnerabilities pose a significant and increasing threat to the security of Android devices, as they directly target physical components and bypass traditional software defenses. Given the growing reliance on Android devices across critical sectors like

finance, healthcare, and personal communications, addressing these vulnerabilities is now a pressing concern. This section systematically reviews key hardware vulnerabilities in Android devices, categorizing them by attack vectors, root causes, and their potential impacts on overall device security. Leveraging the CWE framework, this analysis provides a structured approach to understanding how these weaknesses may be exploited to compromise device integrity.

Moreover, the intricate interplay between hardware and software in Android devices makes them particularly vulnerable to hardware-based attacks that manipulate low-level system functions. These vulnerabilities often evade standard security mechanisms, complicating efforts to detect and mitigate them. Drawing from the CWE database, this analysis offers a detailed examination of critical hardware vulnerabilities, including programming errors, design flaws, and architectural weaknesses, which pose significant risks to Android device security, especially in high-stakes applications.

Table 1 offers a comprehensive summary of the most relevant hardware vulnerabilities, each identified by its CWE code and commonly known name. This table categorizes the vulnerabilities by their descriptions, typical forms of attack, and the underlying causes that lead to their emergence in hardware. This categorization aids in distinguishing between various forms of hardware attacks, such as memory corruption, improper access control, and flawed cryptographic implementations.

Vulnerability Exploitation Scenarios: Hardware vulnerabilities present unique challenges due to their low-level nature, which often makes them harder to detect and mitigate than software-based attacks. For instance, attacks targeting Improper Debug Interface Access [62] exploit debug interfaces to alter chip configurations or access sensitive system registers. Attackers often leverage on-chip debug protocols, such as JTAG [68], which, if not properly secured, can offer a direct route into the hardware. Similarly, Physical Side-Channel Protection [66] exposes vulnerabilities that allow attackers to monitor a device's electromagnetic emissions, power consumption, or acoustic signals to infer sensitive cryptographic operations.

Electromagnetic Interference Attacks

Electromagnetic interference (EMI) poses a significant and often underappreciated threat to the security of mobile devices. These attacks exploit the electromagnetic emissions from electronic components to interfere with or extract sensitive data from devices. High-frequency radiation from a nearby source can induce unintended currents in the circuit of the target device, potentially disrupting normal operations or even altering data transmissions.

EMI vulnerabilities are especially concerning in wireless devices such as smartphones, which rely heavily on electromagnetic signals for communication. Attackers can exploit these emissions to induce faults in the hardware or retrieve sensitive information, such as cryptographic keys, through techniques like differential electromagnetic analysis (DEMA). Research has shown that even well-secured hardware systems can be compromised through sophisticated EMI-based side-channel attacks, especially in environments with inadequate shielding.

One of the key vulnerabilities associated with EMI attacks is CWE-1300 [66]: Physical Side-Channel Protection, which involves insufficient safeguards against side-channel attacks that exploit physical properties like electromagnetic emissions to compromise security. EMI-based attacks such as electromagnetic fault injection (EMFI) and DEMA fall under this category. These attacks can extract cryptographic keys or sensitive data from seemingly secure systems by measuring and analyzing the electromagnetic emissions generated during cryptographic operations. Without proper shielding or noise reduction techniques, the electromagnetic footprint of devices can be exploited to undermine the confidentiality of the system.

Another significant vulnerability is CWE-1332 [67]: Fault Handling, which refers to the improper management of hardware faults, allowing attackers to manipulate system

behavior by inducing faults. In the context of EMI, attackers can use electromagnetic pulses to introduce glitches or errors in critical hardware components, bypassing security controls and enabling unauthorized actions. EMFI is a powerful technique that can exploit this vulnerability, causing unexpected system behavior by disrupting the normal electrical signals in the circuitry.

For Android devices, these vulnerabilities are exacerbated by the high degree of wireless communication that these devices engage in, such as through Wi-Fi, NFC, and Bluetooth. These wireless channels, essential for the operation of mobile devices, offer attack vectors that, when coupled with EMI techniques, can lead to the interception of secure communications, disruption of data transmission, or even manipulation of the system state. High-end Android devices, despite their advanced hardware security features, remain susceptible to such attacks if proper countermeasures, such as electromagnetic shielding and noise filtering, are not implemented.

Other vulnerabilities, such as out-of-bounds-read [60] and use-after-free [61], are particularly threatening because they compromise memory safety. When exploited, these vulnerabilities allow unauthorized access to critical system information or even enable arbitrary code execution. Devices that lack robust memory management safeguards are especially vulnerable, often leading to severe data breaches or complete system compromise.

Mitigation Strategies: Addressing hardware vulnerabilities requires a multifaceted approach, involving secure hardware design, robust firmware updates, and the strict enforcement of security mechanisms such as secure boot and hardware-backed encryption. Solutions to vulnerabilities like described in CWE-1191 [62] include enhancing access controls on debug interfaces through the use of authentication mechanisms and physically tamper-resistant designs. Vulnerability CWE-1300 [66] can be mitigated by integrating side-channel-resistant cryptographic algorithms and hardware-level protections that obscure the physical characteristics of cryptographic operations.

However, implementing these mitigations is not without challenges. Enforcing secure hardware designs or side-channel-resistant algorithms can introduce trade-offs, such as increased hardware costs or reduced performance. Balancing these factors is critical for achieving effective hardware-level security.

Further advancements are essential in the development of hardware-level security measures, particularly as threat landscapes continue to evolve. Attackers are increasingly targeting hardware components to bypass software protections. Future research should focus on strengthening the security of embedded systems, improving fault tolerance, and developing more effective detection mechanisms for side-channel and glitch-based attacks.

This analysis of hardware vulnerabilities in Android devices underscores the critical need for addressing hardware security. These vulnerabilities provide attackers with multiple avenues to compromise the core functionality of mobile devices, bypassing traditional software defenses. The insights presented in this research highlight the necessity of ongoing efforts to enhance hardware resilience, particularly as Android devices continue to play an indispensable role in critical sectors. This review lays the foundation for further exploration of hardware-based security measures and offers key insights into mitigating the risks posed by these vulnerabilities.

To mitigate some EMFI risks, manufacturers must integrate hardware shielding techniques to protect sensitive components and employ noise filtering mechanisms to reduce the susceptibility of circuits to external electromagnetic interference. Additionally, implementing side-channel-resistant cryptographic algorithms and robust fault-handling mechanisms can significantly reduce the attack surface posed by EMI. As mobile devices continue to rely heavily on wireless communication, ensuring their resilience against EMI-based attacks becomes crucial in maintaining their overall security and integrity.

8. Results and Analysis

Building upon the categorization presented in Section 6, this section synthesizes the identified vulnerabilities into actionable insights. The analysis focuses on evaluating their

potential impacts on device security, examining pathways of exploitation, and detailing mitigation strategies to address these hardware weaknesses.

This section presents the findings of our comprehensive analysis of the hardware vulnerabilities affecting Android devices. Our goal is to provide a detailed understanding of the types of vulnerabilities identified, their potential impact on system security, and the pathways through which they can be exploited. This analysis lays the foundation for a structured approach to mitigating these risks and supports the development of more secure Android environments. By systematically reviewing vulnerabilities, we aim to highlight the critical security weaknesses in Android hardware and inform further research efforts aimed at addressing these challenges.

Through an extensive review, we identified several key hardware vulnerabilities that pose significant threats to Android devices. Each vulnerability was examined based on its technical characteristics, the risks it introduces, and the methods attackers can use to exploit it. These vulnerabilities span a broad spectrum of issues, ranging from memory management flaws to weaknesses in cryptographic implementations. The vulnerabilities identified have direct implications for the confidentiality, integrity, and availability of Android devices, making them a critical concern for device security.

The identified vulnerabilities include serious risks associated with improper memory management. For example, *CWE-125 (Out-of-bounds Read)* [60] exposes sensitive information by allowing access to memory areas outside the allocated range, leading to potential data leakage and system instability. Similarly, *CWE-416 (Use After Free)* [61] involves the re-use of memory after it has been freed, which can result in data corruption or arbitrary code execution, compromising system integrity. These types of vulnerabilities are particularly concerning because they undermine the core memory architecture of Android devices, which is essential for maintaining system stability and security.

In addition to memory-related issues, we identified vulnerabilities in system-on-chip (SoC) components, where improper resource isolation between trusted and untrusted agents can lead to unauthorized access and data manipulation. *CWE-1189 (Improper SoC Resource Isolation)* [52] exemplifies this problem, as inadequate isolation mechanisms within the SoC components allow attackers to exploit shared resources, leading to potential breaches in data confidentiality and integrity. SoC vulnerabilities pose a significant risk because they target the hardware-level interactions that are typically shielded from software-based defenses, making them harder to detect and mitigate.

Another critical category of vulnerabilities relates to insecure access to debug interfaces. *CWE-1191 (Improper Debug Interface Access)* [62] highlights the risks associated with weak access controls on on-chip debug interfaces, which attackers can exploit to manipulate internal system configurations, escalate privileges, or gain unauthorized control over the device. Debug interfaces are often overlooked as a potential entry point for attacks, but they offer a direct route to the device's underlying hardware, making them a high-priority target for exploitation.

Cryptographic vulnerabilities also pose a serious threat to the security of Android devices. *CWE-1240 (Risky Cryptographic Implementation)* [53] describes the risks introduced by weak or improperly implemented cryptographic primitives. Inadequate encryption can lead to data leakage and the compromise of secure communications, putting user data at risk. The use of outdated or flawed cryptographic algorithms makes Android devices vulnerable to attacks that bypass encryption protections, undermining the confidentiality of sensitive information.

Table 2 summarizes the key vulnerabilities identified in our analysis, including their CWE codes, descriptions, potential impacts, and the affected components. The vulnerabilities affect various critical parts of Android devices, from memory management systems to SoC components and cryptographic mechanisms. Understanding the nature and impact of these vulnerabilities is essential for designing effective countermeasures that reduce the attack surface and improve the overall resilience of the device.

Table 2. Summary of key hardware vulnerabilities in Android devices.

CWE Code	Vulnerability Description	Impact	Affected Component
CWE-125 [60]	Out-of-bounds Read: Memory access outside the allocated range, potentially exposing sensitive information.	Data leakage, System crashes	Memory Management
CWE-416 [61]	Use After Free: Reusing memory after it has been freed, leading to unpredictable behavior.	Data corruption, Arbitrary code execution	Memory Management
CWE-1189 [52]	Improper SoC Resource Isolation: Lack of isolation between trusted and untrusted agents on the SoC.	Unauthorized access, Data manipulation	SoC
CWE-1191 [62]	Debug Interface Access: Insecure access to on-chip debug interfaces, allowing modification of internal configurations.	System manipulation, Privilege escalation	Debug Interface
CWE-1240 [53]	Risky Cryptographic Implementation: Cryptographic primitives with vulnerabilities that weaken overall security.	Data leakage, Encryption compromise	Cryptography

The vulnerabilities presented in Table 2 represent critical security weaknesses that must be addressed to improve the security posture of Android devices. By analyzing these vulnerabilities, we gain valuable insights into the common failure points within hardware components, which in turn allows for the development of targeted mitigation strategies. Addressing these vulnerabilities requires a multi-layered approach that includes strengthening hardware design, implementing more robust cryptographic standards, and enhancing memory management practices. This understanding will not only contribute to a reduction in the attack surface but also promote greater resilience against emerging hardware-based threats in Android devices.

Through this systematic review and analysis, we have identified key vulnerabilities that pose significant risks to the security of Android devices. By addressing these weaknesses, developers and security professionals can build more resilient hardware infrastructures capable of defending against both current and future threats. Furthermore, the insights provided by this analysis offer a solid foundation for future research aimed at enhancing the security of Android devices in an increasingly hostile threat landscape.

Results of the Survey on Hardware Vulnerabilities

To supplement our review of hardware vulnerabilities, we conducted a survey targeting security professionals and researchers in the field. The survey aimed to assess the perceived criticality of various hardware vulnerabilities in Android devices, as well as to identify areas of greatest concern within the industry. A total of 150 respondents provided their insights, offering a broad perspective on the current state of hardware security.

Table 3 summarizes the key findings of the survey, highlighting the vulnerabilities deemed most critical by respondents.

Table 3. Summary of survey results: critical hardware vulnerabilities.

Vulnerability Type	Percentage of Responses as Critical (%)
System-on-Chip (SoC) Resource Isolation	65
Trusted Execution Environment (TEE) Vulnerabilities	45
Side-Channel Attacks	40
Insecure Debug Interfaces	30
Memory Management Flaws	25

Figure 3 provides a graphical representation of the respondents' perceptions regarding the most significant hardware vulnerabilities. Notably, SoC resource isolation and TEE vulnerabilities were identified as the most pressing concerns, aligning with our review findings.

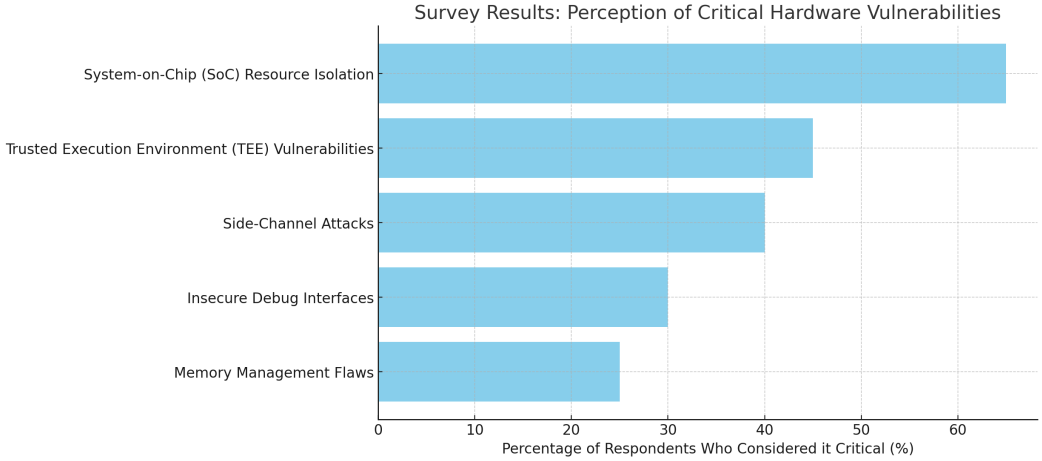


Figure 3. Survey results: perception of critical hardware vulnerabilities.

The survey results indicate a strong alignment with the vulnerabilities discussed in the literature review, particularly regarding the importance of securing SoC and TEE components. Respondents expressed significant concern about the improper isolation of system resources, with 65% considering it a critical risk. This feedback underscores the necessity for enhanced isolation mechanisms and secure hardware design practices to mitigate these vulnerabilities.

9. Classification of Vulnerabilities

The growing complexity of Android devices and their reliance on both hardware and software components make them increasingly vulnerable to sophisticated attacks. Among these, hardware vulnerabilities present unique challenges due to their ability to bypass traditional software-based security defenses. To address these challenges, we propose a comprehensive classification framework that organizes hardware vulnerabilities based on key criteria, such as the nature of the issue, the attack vector, system impact, and severity. This classification provides a structured approach to analyzing hardware risks, offering researchers and practitioners a systematic methodology to prioritize and mitigate vulnerabilities in Android devices.

To enhance the understanding and prioritization of these vulnerabilities, we propose a detailed categorization based on affected components, the nature of the vulnerability, and its potential impact on both the device and the user. This structured approach not only aligns vulnerabilities with the appropriate mitigation strategies, but also helps focus security efforts where they are most needed. Our categorization is divided into four main types: memory and resource management vulnerabilities, information disclosure issues, hardware and SoC weaknesses, and cryptographic vulnerabilities.

One of the most prevalent categories in this classification is memory and resource management vulnerabilities, which arise from the improper handling of system memory and critical resources. These issues can lead to serious security risks such as out-of-bounds access or the reuse of freed memory. For instance, *CWE-125 (Out-of-bounds Read)* [60] exposes sensitive data by accessing memory outside of its allocated boundaries, which can result in data leakage or corruption. Similarly, *CWE-416 (Use After Free)* [61] allows an attacker to exploit memory that has already been deallocated, potentially resulting in arbitrary code execution or system crashes. These vulnerabilities are critical because they directly compromise the integrity and stability of the system's memory architecture, leading to widespread system failures and security breaches. Robust memory management

strategies are essential to mitigate these risks by enforcing strict controls on memory access and allocation.

Another important category includes information disclosure vulnerabilities, which occur when sensitive data is improperly handled or unintentionally exposed. This type of vulnerability poses significant risks to user privacy, as sensitive information may remain accessible in memory or resources even after use. An example of this is *CWE-226 (Sensitive Information Not Removed Before Reuse)* [69], where sensitive data are left in memory after a process completes, making it vulnerable to unauthorized access. Additionally, *CWE-1189 (Incorrect Initialization)* [52] highlights how the improper initialization of variables can expose sensitive information, further amplifying the risks to user privacy. These vulnerabilities underscore the importance of implementing rigorous data handling and disposal procedures to prevent residual data from being accessed by unauthorized entities.

Hardware and system-on-chip (SoC) vulnerabilities represent another significant category, where weaknesses in the physical components of Android devices can be exploited. *CWE-1191 (Improper Debug Interface Access)* [62] is an example, where weak protections on debug interfaces allow attackers to manipulate system configurations, potentially compromising the entire device. Similarly, *CWE-1189 (Improper SoC Resource Isolation)* [52] highlights how inadequate isolation mechanisms between trusted and untrusted components within the SoC can lead to unauthorized access to shared resources, opening the door to further exploitation. The importance of securing these hardware interfaces is paramount, as hardware-level attacks often bypass software defenses, directly undermining system integrity.

In addition, cryptographic issues present another layer of complexity, particularly when cryptographic implementations are flawed. Vulnerabilities in cryptographic mechanisms, such as *CWE-1240 (Risky Cryptographic Implementation)* [53], occur when weak cryptographic algorithms or the improper use of cryptographic primitives expose the system to potential data leaks or encryption bypass. Ensuring robust and up-to-date cryptographic implementations is critical for protecting sensitive communications and data in Android devices. These vulnerabilities emphasize the need for secure, hardware-backed cryptographic protocols that can withstand modern attack techniques.

Vulnerabilities can also be categorized based on the attack vectors used to exploit them. Attack vectors such as input manipulation, memory corruption, and fault injection provide insights into how adversaries can exploit weaknesses. For instance, input manipulation involves tricking the system into accepting malicious inputs, while memory corruption exploits flaws in memory handling, as seen in *CWE-125* [60] and *CWE-416* [61]. Fault injection attacks, like *CWE-1191* [62], disrupt normal hardware operations by introducing faults or glitches that allow attackers to bypass security mechanisms. Each attack vector presents unique challenges for security solutions, requiring tailored strategies to defend against the specific methods attackers use to exploit these vulnerabilities.

Classifying vulnerabilities by their system impact offers valuable insight into the broader security implications of each issue. Vulnerabilities that compromise confidentiality, such as *CWE-125 (Out-of-bounds Read)* [60], allow unauthorized access to sensitive data, undermining the privacy of users and exposing critical information. Integrity-related vulnerabilities, such as *CWE-416 (Use After Free)* [61], enable attackers to modify system data or execute malicious code, threatening the reliability and stability of the device. Availability-based vulnerabilities, like *CWE-1191 (Improper Debug Interface Access)* [62], can cause system instability or resource exhaustion, potentially leading to denial-of-service conditions. By understanding these different impacts, security practitioners can prioritize vulnerabilities based on their potential to disrupt core system functions.

Severity classification provides an essential framework for prioritizing mitigation efforts. Vulnerabilities are classified into high, medium, and low severity categories, considering factors such as the ease of exploitation, potential system damage, and the availability of effective countermeasures. High-severity vulnerabilities, such as *CWE-125* [60] and *CWE-416* [61], pose the greatest risk, often leading to system compromise or

data exposure, and require immediate attention. Medium-severity vulnerabilities, like *CWE-226* [69] and *CWE-1191* [62], can cause partial system failures or unauthorized access, while low-severity vulnerabilities, such as *CWE-1189* [52], may only lead to minor performance degradation but should still be addressed as part of a comprehensive security strategy.

By adopting this structured classification of hardware vulnerabilities, we gain a deeper understanding of the specific risks they pose to Android devices and their users. This approach allows for more targeted and effective mitigation strategies, helping to prioritize security efforts where they are most needed. Furthermore, it contributes to the broader development of resilient hardware security frameworks for Android devices, which are critical in safeguarding sectors such as finance, healthcare, and communications. As Android devices continue to play a central role in daily life, it is imperative to continue refining hardware security to address the evolving threat landscape.

Discussion on Advantages of the Categorization: The categorization framework presented in this section provides multiple benefits, particularly in the structured analysis and mitigation of hardware vulnerabilities. By dividing vulnerabilities based on their impact, attack vector, and severity, researchers and security practitioners can prioritize efforts efficiently. This approach facilitates the identification of the most pressing threats, enabling targeted mitigation strategies that align with the available resources and urgency of the vulnerabilities. Moreover, this structured categorization provides a clear roadmap for understanding how different hardware flaws can affect device security holistically, promoting a layered security approach that is essential for safeguarding Android devices against evolving threats.

10. Conclusions and Future Works

This paper presents a comprehensive and systematic review of hardware-related vulnerabilities in Android devices, a field that has historically received less attention compared to software vulnerabilities. By providing a detailed categorization of hardware weaknesses based on attack vectors, system impact, and severity, we have offered a structured framework for understanding the diverse risks that hardware vulnerabilities pose to Android systems. This classification framework, distinguishing vulnerabilities across various hardware components and their corresponding attack surfaces, serves as a valuable resource for both researchers and practitioners. Such categorization is instrumental in identifying, prioritizing, and developing targeted countermeasures for mitigating these vulnerabilities.

The analysis has demonstrated that hardware vulnerabilities, unlike software vulnerabilities, often bypass traditional software-based defenses, underscoring the critical need for robust hardware security mechanisms. Sectors such as healthcare, finance, and communications, where security is paramount, are particularly vulnerable to these hardware attacks. Consequently, addressing these vulnerabilities with secure boot mechanisms, trusted execution environments (TEE), and hardware-backed encryption has become essential in enhancing the resilience of Android devices against emerging threats. These countermeasures provide actionable strategies for improving hardware security, and their adoption will play a key role in safeguarding sensitive data and ensuring system integrity.

Despite these advances, the evolving nature of hardware attacks demands continuous research efforts. Future works should expand the catalog of vulnerabilities to include newly discovered issues, especially as advancements in hardware design and mobile technologies introduce novel attack vectors. An important avenue for future exploration is the development of more advanced detection and mitigation mechanisms that operate at the hardware level. This includes enhancing fault tolerance, refining side-channel attack prevention techniques, and improving secure boot processes to defend against increasingly sophisticated hardware-based attacks.

Furthermore, integrating artificial intelligence (AI) and machine learning (ML) into hardware-based security solutions represents a promising direction for real-time threat detection and mitigation. As attackers exploit more complex vectors, collaboration between hardware manufacturers, software developers, and the cybersecurity community will be

essential. Additionally, ongoing research must investigate the role of standardization in hardware security, ensuring that future Android devices are equipped with consistent and effective defenses against hardware-level threats.

Lastly, emerging technologies such as 5G, the Internet of Things (IoT), and edge computing bring new security challenges. Future studies should explore the specific hardware vulnerabilities associated with these technologies, ensuring that security strategies remain adaptive and proactive. By focusing on these critical areas, the research community and industry professionals can stay ahead of the evolving threat landscape and ensure that Android devices remain secure in an increasingly interconnected world. Ultimately, this work fills a critical gap in the existing literature by systematically addressing hardware vulnerabilities and laying the foundation for future research and practical implementations in Android device security.

Funding: This research received no external funding.

Data Availability Statement: Data is contained within the article.

Conflicts of Interest: The author declares no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AI	Artificial Intelligence
ART	Android Runtime
CVE	Common Vulnerabilities and Exposures
CWE	Common Weakness Enumeration
DVM	Dalvik Virtual Machine
FBE	File-Based Encryption
FDE	Full-Disk Encryption
IoT	Internet of Things
JTAG	Joint Test Action Group
HBE	Hardware-Based Encryption
HSM	Hardware Security Module
ICC	Inter-Component Communication
ML	Machine Learning
NVD	National Vulnerability Database
OS	Operating Systems
SOC	System-On-Chip
TEE	Trusted Execution Environment
LD	Linear Dichroism

References

1. StatCounter. StatCounter Global Stats. Available online: https://gs.statcounter.com/ (accessed on 22 July 2024).
2. Ling, Z.; Yan, H.; Shao, X.; Luo, J.; Xu, Y.; Pearson, B.; Fu, X. Secure boot, trusted boot and remote attestation for ARM TrustZone-based IoT Nodes. *J. Syst. Archit.* **2021**, *119*, 102240. [CrossRef]
3. Redini, N.; Machiry, A.; Das, D.; Fratantonio, Y.; Bianchi, A.; Gustafson, E.; Shoshitaishvili, Y.; Kruegel, C.; Vigna, G. {BootStomp}: On the security of bootloaders in mobile devices. In Proceedings of the 26th USENIX Security Symposium (USENIX Security 17), Vancouver, BC, Canada, 16–18 August 2017; pp. 781–798.
4. Qamar, A.; Karim, A.; Chang, V. Mobile malware attacks: Review, taxonomy & future directions. *Future Gener. Comput. Syst.* **2019**, *97*, 887–909.
5. Hoang, T.T.; Duran, C.; Serrano, R.; Sarmiento, M.; Nguyen, K.D.; Tsukamoto, A.; Suzaki, K.; Pham, C.K. Trusted execution environment hardware by isolated heterogeneous architecture for key scheduling. *IEEE Access* **2022**, *10*, 46014–46027. [CrossRef]
6. Ménétrey, J.; Göttel, C.; Pasin, M.; Felber, P.; Schiavoni, V. An exploratory study of attestation mechanisms for trusted execution environments. *arXiv* **2022**, arXiv:2204.06790.
7. Ryan, K. Hardware-backed heist: Extracting ECDSA keys from qualcomm's trustzone. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, London, UK, 11–15 November 2019; pp. 181–194.

8. Khati, L.; Mouha, N.; Vergnaud, D. Full disk encryption: Bridging theory and practice. In Proceedings of the Topics in Cryptology—CT-RSA 2017: The Cryptographers' Track at the RSA Conference 2017, San Francisco, CA, USA, 14–17 February 2017; Springer: Berlin/Heidelberg, Germany, 2017; pp. 241–257.
9. Galindo, D.; Liu, J.; Stone, C.M.; Ordean, M. SoK: Untangling file-based encryption on mobile devices. *arXiv* **2021**, arXiv:2111.12456.
10. Sheibani, M.; Konur, S.; Awan, I.; Qureshi, A. A Multi-Layered Defence Strategy against DDoS Attacks in SDN/NFV-Based 5G Mobile Networks. *Electronics* **2024**, *13*, 1515. [CrossRef]
11. Zhao, L.; Shuang, H.; Xu, S.; Huang, W.; Cui, R.; Bettadpur, P.; Lie, D. Sok: Hardware security support for trustworthy execution. *arXiv* **2019**, arXiv:1910.04957.
12. Nouman, N.; Noreen, Z.; Naz, F. Vulnerabilities in Android OS: Challenges and Mitigation Techniques. In Proceedings of the International Conference on Digital Technologies and Applications, Vienna, Austria, 12–14 May 2022; Springer: Berlin/Heidelberg, Germany, 2022; pp. 256–266.
13. Sabt, M.; Achemlal, M.; Bouabdallah, A. Trusted execution environment: What it is, and what it is not. In Proceedings of the 2015 IEEE Trustcom/BigDataSE/Ispa, Helsinki, Finland, 20–22 August 2015; IEEE: Piscataway, NJ, USA, 2015; Volume 1, pp. 57–64.
14. Muñoz, A.; Rios, R.; Román, R.; López, J. A survey on the (in) security of trusted execution environments. *Comput. Secur.* **2023**, *129*, 103180. [CrossRef]
15. Oh, H.S.; Kim, B.J.; Choi, H.K.; Moon, S.M. Evaluation of Android Dalvik virtual machine. In Proceedings of the 10th International Workshop on JAVA Technologies for Real-Time and Embedded Systems, Copenhagen, Denmark, 24–26 October 2012; pp. 115–124.
16. Backes, M.; Bugiel, S.; Schranz, O.; von Styp-Rekowsky, P.; Weisgerber, S. Artist: The android runtime instrumentation and security toolkit. In Proceedings of the 2017 IEEE European Symposium on Security and Privacy (EuroS&P), Paris, France, 26–28 April 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 481–495.
17. Singh, R. An overview of android operating system and its security. *Int. J. Eng. Res. Appl.* **2014**, *4*, 519–521.
18. Jenkins, J.; Cai, H. ICC-inspect: Supporting runtime inspection of Android inter-component communications. In Proceedings of the 5th International Conference on Mobile Software Engineering and Systems, Gothenburg, Sweden, 27–28 May 2018; pp. 80–83.
19. Veer, R.; Patil, R.; Mhatre, A.; Gaikwad, S. Android Development based on Linux. *Int. J. Adv. Res. Innov. Ideas Educ. (IJARIIE)* **2018**, *4*, 39–46.
20. Sharma, T.; Rattan, D. Malicious application detection in android—A systematic literature review. *Comput. Sci. Rev.* **2021**, *40*, 100373. [CrossRef]
21. Sutter, T.; Kehrer, T.; Rennhard, M.; Tellenbach, B.; Jacques Klein, J. Dynamic security analysis on android: A systematic literature review. *IEEE Access* **2024**, *12*, 57266–57267. [CrossRef]
22. Senanayake, J.; Kalutarage, H.; Al-Kadri, M.O.; Petrovski, A.; Piras, L. Android source code vulnerability detection: A systematic literature review. *ACM Comput. Surv.* **2023**, *55*, 187. [CrossRef]
23. Garg, S.; Baliyan, N. Android security assessment: A review, taxonomy and research gap study. *Comput. Secur.* **2021**, *100*, 102087. [CrossRef]
24. Abdullah, H.; Zeebaree, S.R. Android Mobile Applications Vulnerabilities and Prevention Methods: A Review. In Proceedings of the 2nd International Conference of Information Technology to Enhance E-Learning and Other Application (IT-ELA2021), Baghdad, Iraq, 27–28 December 2022; IEEE: Piscataway, NJ, USA, 2021.
25. Visoottiviseth, V.; Kotarasu, C.; Cheunprapanusorn, N.; Chamornmarn, T. A Mobile Application for Security Assessment Towards the Internet of Thing Devices. In Proceedings of the 2019 IEEE 6th Asian Conference on Defence Technology (ACDT), Bali, Indonesia, 13–15 November 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 1–7.
26. Arora, D.; Sonwane, A.; Wadhwa, N.; Mehrotra, A.; Utpala, S.; Bairi, R.; Kanade, A.; Natarajan, N. MASAI: Modular Architecture for Software-engineering AI Agents. *arXiv* **2024**, arXiv:2406.11638.
27. Rahman, A.; Pradhan, P.; Partho, A.; Williams, L. Predicting Android application security and privacy risk with static code metrics. In Proceedings of the 2017 IEEE/ACM 4th International Conference on Mobile Software Engineering and Systems (MOBILESoft), Buenos Aires, Argentina, 22–23 May 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 149–153.
28. Watanabe, T.; Akiyama, M.; Kanei, F.; Shioji, E.; Takata, Y.; Sun, B.; Ishi, Y.; Shibahara, T.; Yagi, T.; Mori, T. Understanding the origins of mobile app vulnerabilities: A large-scale measurement study of free and paid apps. In Proceedings of the 2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR), Buenos Aires, Argentina, 20–21 May 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 14–24.
29. Wu, D.; Gao, D.; Cheng, E.K.; Cao, Y.; Jiang, J.; Deng, R.H. Towards understanding Android system vulnerabilities: Techniques and insights. In Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security, Auckland, New Zeland, 7–12 July 2019; pp. 295–306.
30. Linares-Vásquez, M.; Bavota, G.; Escobar-Velásquez, C. An empirical study on android-related vulnerabilities. In Proceedings of the 2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR), Buenos Aires, Argentina, 20–21 May 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 2–13.
31. Christey, S.; Kenderdine, J.; Mazella, J.; Miles, B. *Common Weakness Enumeration*; MITRE Corporation: McLean, VA, USA, 2013.
32. Liu, K.L.C. Supply Chain for Mobile Network Operator. Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2010.
33. Dejon, N. Design of a Secure Kernel for Constrained Devices. Ph.D. Thesis, Université de Lille, Lille, France, 2022.

34. Mihalcea, I. *Prototyping Memory Integrity Tree Algorithms for Internet of Things Devices*; Technical Report; Information Security Group, Royal Holloway University of London: Egham, UK, 2022.
35. Makhdoom, I.; Abolhasan, M.; Franklin, D.; Lipman, J.; Zimmermann, C.; Piccardi, M.; Shariati, N. Detecting compromised IoT devices: Existing techniques, challenges, and a way forward. *Comput. Secur.* **2023**, *132*, 103384. [CrossRef]
36. Ojo, M.O.; Giordano, S.; Procissi, G.; Seitanidis, I.N. A review of low-end, middle-end, and high-end IoT devices. *IEEE Access* **2018**, *6*, 70528–70554. [CrossRef]
37. Schulz, S.; Schaller, A.; Kohnhäuser, F.; Katzenbeisser, S. Boot attestation: Secure remote reporting with off-the-shelf IoT sensors. In Proceedings of the Computer Security—ESORICS 2017: 22nd European Symposium on Research in Computer Security, Oslo, Norway, 11–15 September 2017; Proceedings, Part II 22; Springer: Berlin/Heidelberg, Germany, 2017; pp. 437–455.
38. Göttel, C.; Pires, R.; Rocha, I.; Vaucher, S.; Felber, P.; Pasin, M.; Schiavoni, V. Security, performance and energy trade-offs of hardware-assisted memory protection mechanisms. In Proceedings of the 2018 IEEE 37th Symposium on Reliable Distributed Systems (SRDS), Salvador, Brazil, 2–5 October 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 133–142.
39. Sommerhalder, M. Hardware Security Module. In *Trends in Data Protection and Encryption Technologies*; Springer: Berlin/Heidelberg, Germany, 2023; pp. 83–87.
40. de Faveri Tron, A.; Longari, S.; Carminati, M.; Polino, M.; Zanero, S. Canflict: Exploiting peripheral conflicts for data-link layer attacks on automotive networks. In Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, Los Angeles, CA, USA, 7–11 November 2022; pp. 711–723.
41. Udvarhelyi, B.; van Wassenhove, A.; Bronchain, O.; Standaert, F.X. On the security of off-the-shelf microcontrollers: Hardware is not enough. In Proceedings of the Smart Card Research and Advanced Applications: 19th International Conference, CARDIS 2020, Virtual Event, 18–19 November 2020; Revised Selected Papers 19; Springer: Berlin/Heidelberg, Germany, 2021; pp. 103–118.
42. Lopez, A.B. *Security Modeling and Analysis for Intelligent Transportation Systems*; University of California: Irvine, CA, USA, 2020.
43. Jones, K.R.; Yen, T.F.; Sundaramurthy, S.C.; Bardas, A.G. Deploying android security updates: An extensive study involving manufacturers, carriers, and end users. In Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, 9–13 November 2020; pp. 551–567.
44. Dupré, G. Energy Efficiency in AES Encryption on ARM Cortex CPUs: Comparative Analysis Across Modes of Operation, Data Sizes, and Key Lengths (Dissertation). 2024. Available online: https://urn.kb.se/resolve?urn=urn:nbn:se:his:diva-24099 (accessed on 2 October 2024).
45. Quarkslab. RFID: New Proxmark3 Tear-Off Features and New Findings. 2023. Available online: https://blog.quarkslab.com/rfid-new-proxmark3-tear-off-features-and-new-findings.html (accessed on 27 September 2024).
46. National Institute of Standards and Technology (NIST). National Vulnerability Database (NVD). 2005. Available online: https://nvd.nist.gov/ (accessed on 23 October 2024).
47. Martin, R.; Christey, S.; Baker, D.; Corporation, M. *The Common Vulnerabilities and Exposures (CVE) Initiative*; MITRE Corporation: McLean, VA, USA, 2002.
48. MITRE Corporation. Common Vulnerabilities and Exposures (CVE). 1999. Available online: https://cve.mitre.org/ (accessed on 23 October 2024).
49. Offensive Security. Exploit Database (Exploit-DB). 2009. Available online: https://www.exploit-db.com/ (accessed on 23 October 2024).
50. Vulnerability Database (VULDB). Vulnerability Database (VULDB). 2002. Available online: https://vuldb.com/ (accessed on 23 October 2024).
51. Computer Emergency Response Team (CERT). CERT Vulnerability Database. 1988. Available online: https://www.kb.cert.org/vuls/ (accessed on 23 October 2024).
52. CWE-1189: Improper Isolation of Shared Resources on System-on-a-Chip. Available online: https://cwe.mitre.org/data/definitions/1189.html (accessed on 24 March 2024).
53. CWE-1240: Use of a Cryptographic Primitive with a Risky Implementation. Available online: https://cwe.mitre.org/data/definitions/1240.html (accessed on 20 March 2024).
54. Tan, B. Challenges and opportunities for hardware-assisted security improvements in the field. In Proceedings of the 2022 23rd International Symposium on Quality Electronic Design (ISQED), Santa Jose, CA, USA, 6–7 April 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 90–95.
55. Dharavathu, A. Towards Reconfigurable Hardware for In-Field Hardware Bug Patches. Master's Thesis, University of Calgary, Calgary, AB, Canada, 2024.
56. Chen, K.; Arias, O.; Guo, X.; Deng, Q.; Jin, Y. IP-Tag: Tag-Based Runtime 3PIP Hardware Trojan Detection in SoC Platforms. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2022**, *42*, 68–81. [CrossRef]
57. Pandey, S.; Pathak, P.C.; Tripathi, S.; Halwai, S.; Aggarwal, S.; Nishant, N. Security Risks and Their Mitigation Strategies: Cloud Computing Perspective. In Proceedings of the 2023 10th IEEE Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON), Gautam Buddha Nagar, India, 1–3 December 2023; IEEE: Piscataway, NJ, USA, 2023; Volume 10, pp. 1015–1019.
58. Fu, W.; Li, S.; Zhao, Y.; Ma, H.; Dutta, R.; Zhang, X.; Yang, K.; Jin, Y.; Guo, X. Hardware phi-1.5 b: A large language model encodes hardware domain specific knowledge. In Proceedings of the 2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC), Incheon, Republic of Korea, 22–25 January 2024; IEEE: Piscataway, NJ, USA, 2024; pp. 349–354.

59. Ahmad, B.; Liu, W.K.; Collini, L.; Pearce, H.; Fung, J.M.; Valamehr, J.; Bidmeshki, M.; Sapiecha, P.; Brown, S.; Chakrabarty, K.; et al. Don't cweat it: Toward cwe analysis techniques in early stages of hardware design. In Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design, San Diego, CA, USA, 30 October–3 November 2022; pp. 1–9.
60. CWE-125: Out-of-Bounds Read. Available online: https://cwe.mitre.org/data/definitions/125.html (accessed on 25 March 2024).
61. CWE-416: Use After Free. Available online: https://cwe.mitre.org/data/definitions/416.html (accessed on 21 March 2024).
62. CWE-1191: On-Chip Debug and Test Interface with Improper Access Control. Available online: https://cwe.mitre.org/data/definitions/1191.html (accessed on 20 March 2024).
63. CWE-1256: Improper Restriction of Software Interfaces to Hardware Features. Available online: https://cwe.mitre.org/data/definitions/1256.html (accessed on 23 March 2024).
64. CWE-1272: Sensitive Information Uncleared Before Debug/Power State Transition. Available online: https://cwe.mitre.org/data/definitions/1272.html (accessed on 24 March 2024).
65. CWE-1274: Improper Access Control for Volatile Memory Containing Boot Code. Available online: https://cwe.mitre.org/data/definitions/1274.html (accessed on 22 March 2024).
66. CWE-1300: Improper Protection of Physical Side-Channels. Available online: https://cwe.mitre.org/data/definitions/1300.html (accessed on 23 March 2024).
67. CWE-1332: Improper Handling of Faults That Lead to Instruction Skips. Available online: https://cwe.mitre.org/data/definitions/1332.html (accessed on 21 March 2024).
68. Nguyen, V.T.; Popova, T.A.; Uvaysova, S.S. JTAG-Technology as a Tool for Testing Complex Components of Printed Circuit Boards. In Proceedings of the Information Innovative Technologies, Kottayam, India, 13–14 February 2020; pp. 203–208.
69. CWE-226: Sensitive Information in Resource Not Removed Before Reuse. Available online: https://cwe.mitre.org/data/definitions/226.html (accessed on 24 March 2024).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.