Vulnerability, Where Art Thou? An Investigation of Vulnerability Management in Android Smartphone Chipsets

Daniel Klischies
Ruhr University Bochum
daniel.klischies@ruhr-uni-bochum.de

Philipp Mackensen
Ruhr University Bochum
philipp.mackensen@ruhr-uni-bochum.de

Veelasha Moonsamy
Ruhr University Bochum
email@veelasha.org

***Abstract*—Vulnerabilities in Android smartphone chipsets have severe consequences, as recent real-world attacks [1] have demonstrated that adversaries can leverage vulnerabilities to execute arbitrary code or exfiltrate confidential information. Despite the far-reaching impact of such attacks, the lifecycle of chipset vulnerabilities has yet to be investigated, with existing papers primarily investigating vulnerabilities in the Android operating system. This paper provides a comprehensive and empirical study of the current state of smartphone chipset vulnerability management within the Android ecosystem. For the first time, we create a unified knowledge base of 3,676 chipset vulnerabilities affecting 437 chipset models from all four major chipset manufacturers, combined with 6,866 smartphone models. Our analysis revealed that the same vulnerabilities are often included in multiple generations of chipsets, providing novel empirical evidence that vulnerabilities are inherited through multiple chipset generations. Furthermore, we demonstrate that the commonly accepted 90-day responsible vulnerability disclosure period is seldom adhered to. We find that a single vulnerability often affects hundreds to thousands of different smartphone models, for which update availability is, as we show, often unclear or heavily delayed. Leveraging the new insights gained from our empirical analysis, we recommend several changes that chipset manufacturers can implement to improve the security posture of their products. At the same time, our knowledge base enables academic researchers to conduct more representative evaluations of smartphone chipsets, accurately assess the impact of vulnerabilities they discover, and identify avenues for future research.**

I. Introduction

Smartphones play an integral part of our daily lives and are entrusted with safety-critical tasks, such as emergency calls and safeguarding of users' confidential information. Most smartphones run a version of Android, which is the most popular mobile operating system in the world, with a market share of 70.5% [2]. It is thus of utmost importance to maintain the security of Android smartphones by proactively identifying particularly vulnerable components as well as ensuring timely updates after a vulnerability has been discovered. An especially security-relevant component of smartphones is the *chipset*. Chipsets consist of multiple, tightly integrated processors, some of which provide general-purpose compute to run a mobile operating system, while others provide acceleration features for graphics, or enable wireless connectivity. The functionalities implemented by chipsets inherently put them in an interesting position for an attacker, as chipsets, by design, have access to sensitive information prior to encryption as well as long-term cryptographic keys stored on the device.

As a result of the aforementioned exposure, threat actors started to actively exploit chipset vulnerabilities. In 2023, Amnesty International reported [3] that Predator, a commercial surveillance spyware capable of extracting messages, online browsing history, contact lists and location data from compromised phones, can be deployed through vulnerabilities in Samsung chipsets - without interaction by the victim or cooperation of the victim's service provider. To minimize the risk of being compromised, Amnesty International recommended that individuals at-risk should "Always update [...] as soon as any security updates are made available for your devices." However, this recommendation assumes the existence of device updates, which is dependent on the close cooperation of several entities, as the vulnerability originates from the chipset, rather than the device.

Chipset processors are closely intertwined with their software, consisting of firmware, which runs on them, and the drivers that create the interface to the Android OS. Chipset processors, firmware and drivers are developed by *chipset manufacturers* (CMs), rather than by smartphone manufacturers, i.e. *Original Equipment Manufacturers* (OEMs), nor as a part of the *Android Open Source Project* (AOSP). This means that the CMs need to continue to support and provide updated firmware and drivers to OEMs over the lifetime of a chipset, in particular, to mitigate security vulnerabilities. The primary goal when dealing with such vulnerabilities is to minimize the length of the *vulnerability lifecycle*, i.e., the overall time frame a vulnerability is exploitable, by employing successful *vulnerability management* across the supply chain. For smartphone chipsets, this time frame is divided into four phases: (i) the introduction of a vulnerability, (ii) its eventual discovery, (iii) the development of a patch removing the vulnerability by the CM, and (iv) packaging of the patch

Network and Distributed System Security (NDSS) Symposium 2025
24-28 February 2025, San Diego, CA, USA
ISBN 979-8-9894372-8-3
https://dx.doi.org/10.14722/ndss.2025.241161
www.ndss-symposium.org

into an update that is deployed to end-users' smartphones by the OEM. Each of these phases plays a significant role in safeguarding device security, from both, a technical and an organisational perspective. Each phase has a separate technical impact, as they (i) influence which vulnerabilities a chipset will be impacted by, (ii) where discovered vulnerabilities are located, (iii) reveal how many chipset models a single vulnerability affects and how severe vulnerabilities are, and (iv) how many smartphones are ultimately affected.

While prior work suspected that some chipset vulnerabilities tend to propagate across many generations [4], [5], along with speculation that vulnerabilities in chipset firmware are more severe than driver vulnerabilities [6], there is a significant lack of empirical evidence to support these claims on a large scale. Existing studies have primarily relied on limited case studies or anecdotal evidence, leaving a considerable gap in our understanding of the true prevalence and impact of these vulnerabilities. Consequently, our work seeks to bridge this gap by providing extensive empirical analysis, thereby offering a more concrete foundation for future research and practical security measures. More concretely our work aims to provide a knowledge base enabling evidence-based choice of research targets, more representative evaluations and a more accurate depiction of the impact of discovered vulnerabilities.

Analogously to the technical characteristics, each phase of the vulnerability lifecycle also encompasses important organisational aspects, as they shine light on (i) the way chipset firmware and drivers, including vulnerabilities, are developed, (ii) the factors promoting the discovery of vulnerabilities, (iii) patch development timelines, and (iv) inter-organizational coordination between CMs and OEMs. These organizational aspects have recently come under scrutiny, as both, a member of Google's Threat Analysis Group[1] and prior work [7] pointed out that some critical chipset vulnerabilities were not resolved within the 90 days responsible disclosure window, leaving end-users in danger after the vulnerabilities have been publicly disclosed. While such anecdotes highlight that vulnerability management processes occasionally fail, it is unclear whether this is a systemic flaw. The absence of a comprehensive, large-scale measurement undermines the ability to generalize observed process failures and suggest improvements of such processes. This is not only relevant to CMs, but to the entire security research community for two reasons: First, knowledge about internal processes should influence the decisions made by researchers when handling chipset vulnerabilities, such as the vulnerability disclosure timeline. Secondly, insights and suggestions for process improvements concerning the chipset vulnerability lifecycle may be applicable to dependent product categories where multiple companies must interact to discover vulnerabilities, develop patches and deploy these patches via device updates - such as IoT devices or connected vehicles.

To determine the technical and organizational characteristics of the vulnerability lifecycle and identify potential for improvements, we devise the following research questions:

[1]https://twitter.com/maddiestone/status/1636469657136959488

RQ1: Where do vulnerabilities in chipsets *originate*?
RQ2: Who *discovers* vulnerabilities in a chipset?
RQ3: When are *patches* available and how severe are the chipset vulnerabilities they mitigate?
RQ4: What are the characteristics of the *update* process utilized by OEMs to address chipset vulnerabilities in Android devices?

The current lack of information on the processes that take place in every phase of the vulnerability lifecycle, preventing us from answering our RQs based on prior work, is for two reasons: Firstly, existing papers primarily focus on understanding the final phase of the vulnerability lifecycle, in particular investigating the availability of device updates, and various factors that prolong the time it takes OEMs to provide these updates. This leaves out other phases of the vulnerability lifecycle. Secondly, existing studies focus on vulnerabilities in, and updates for, the Android mobile operating system. The sources from which they obtain their empirical data are, however, lacking comprehensive information on chipset vulnerabilities. We make the following five main **contributions**:

- We create a unified *knowledge base*, the first of its kind, aggregating comprehensive information on chipset vulnerabilities, patches, affected devices, and their update status, encompassing a majority of the Android smartphone ecosystem (Section IV).
- We then leverage our large-scale knowledge base to obtain answers to **RQ1-4**, discovering several key insights regarding the origin of vulnerabilities in newly released chipset models, limited industry transparency, and prolonged time frames for patches and updates (Section V).
- We compare our findings to similar studies in other ecosystems, highlighting similarities and differences. This allows us to determine factors that influence various aspects of the vulnerability lifecycle (Section VI).
- Given our results, we discuss several actionable changes that stakeholders could implement to improve device security and enable end-users to make informed purchase decisions resulting from improved transparency (Section VII-B).
- Finally, we describe use cases on how our newly created knowledge base can enrich and enhance future research works. We elaborate how our data enables representative experimental evaluations, provides a more accurate depiction of the impact of newly discovered vulnerabilities, and unveils promising avenues for research (Section VII-C).

We **publish** our data set in the form of a continuously self-updating website at https://www.chipsets.org to enable reproducibility, allow continuous observation of future trends, and facilitate experimental evaluation.

II. Background

A. Smartphone Chipsets

Smartphone chipsets, integral components of mobile devices, are a set of specialized processors, responsible for providing various functions essential for a smartphone's operation. These chipsets encompass a diverse set of *compo-*

nents, including an application processor running e.g. Android, graphics processing units (GPUs), modem processors for wireless connectivity such as cellular, Bluetooth and WiFi, and various other specialized co-processors. To enable end-users to make calls, play audio over Bluetooth, or use the GPU for gaming on their smartphone, a chipset-specific driver within Android interfaces with these co-processors, delegating tasks like wireless communication or graphics computation. To this end, each co-processor executes a separate, chipset model and component-specific firmware, independently from Android. Furthermore, different components of chipsets often communicate directly with each other, e. g., to synchronize on radio frequency usage [8]. For this reason, as well as for performance and energy conservation, modern smartphone chipsets are typically manufactured in a tightly integrated way by a single CM per chipset model, rather than each functionality being supplied by a component of a different manufacturer. [9]–[12]. The Android smartphone chipset market is therefore effectively an oligopoly of four CMs: Qualcomm, Mediatek, Samsung and Unisoc. These four companies cover 99% of the smartphone chipset market [13]. Because of this relatively small diversification, developing targeted exploits for specific chipsets is economically viable for threat actors [14]. This is even more of an issue, as the tight integration enables adversaries to, e. g., exploit a vulnerability in a Bluetooth component to ultimately eavesdrop on WiFi communication [8]. A holistic view on chipset vulnerabilities is thus warranted to comprehensively understand the threat-landscape and security posture of modern smartphones.

B. Vantage Points

There exists multiple vantage points from which one can obtain information on vulnerabilities and available updates.

Vulnerability Databases: The NIST National Vulnerability Database (NVD) and the CVE.org database, both operated by the US government, contain mostly identical information on vulnerabilities affecting arbitrary products, including chipsets. Due to the generic nature of such databases, they only contain a high level description of vulnerabilities, along with a Common Vulnerability Scoring System (CVSS) severeness score, which expresses the impact and exploitability of a vulnerability. The NVD also includes Common Platform Enumerations (CPEs) that, in theory, are intended to identify software and hardware affected by a vulnerability. However, not every chipset vulnerability is associated with a CPE. CPEs of chipset vulnerabilities also only specify affected chipset models, but not affected smartphone models.

Chipset Manufacturers: Bulletins on CM websites list chipset vulnerabilities including affected model numbers, severity ratings, and usually the same description that is also later used in the NVD, as well as a CVSS score. They also contain additional, chipset-specific information, such as the component of the chipset that is affected by a particular vulnerability. CMs also acknowledge who discovered a vulnerability, if the vulnerability was not discovered by one of their employees. As each CM only publishes their own vulnerabilities, the bulletin format is not standardised across CMs.

Android Open Source Project: AOSP security bulletins provide detailed insights into vulnerabilities identified within the Android OS, their severity levels, and affected Android versions. These bulletins play a pivotal role in enhancing transparency, as they contain a unique identifier, the Security Patch Level (SPL). Each Android device also displays an SPL in its settings dialog. By matching their devices' SPL to the corresponding AOSP bulletin, end-users can assess which Android OS vulnerabilities have already received a mitigating update, with vulnerabilities listed in bulletins not referenced by the devices' SPL being unmitigated. The Android Security Bulletins are therefore both: A list of vulnerabilities affecting devices, and a changelog representing which vulnerabilities have been addressed. In some cases, these bulletins also contain chipset vulnerabilities. However, AOSP security bulletins do not allow to draw an immediate conclusion on which chipset vulnerabilities a phone is vulnerable to, as this depends on the exact chipset model built into this phone. Likewise, the completeness of these bulletins w. r. t. chipset vulnerabilities is unclear. Therefore, only taking into account AOSP security bulletins in isolation is insufficient to obtain complete and correct information on the chipset vulnerabilities affecting a smartphone.

OEMs: In contrast to vulnerability databases and CM bulletins, OEM information is communicated on a per-smartphone, rather than per-chipset basis. OEMs either publish device-specific changelogs on their website, listing all vulnerabilities that have been addressed in the past, or they refer to an SPL in their changelog, consequently, ignoring vulnerabilities missing from AOSP security bulletins. OEM changelogs are thus insufficient to draw a conclusion on the security posture of a smartphone, as they only list vulnerabilities that have been already addressed, but do not enable an assessment of vulnerabilities that have not received an update.

Unifying disparate information sources: To obtain information on all chipset vulnerabilities affecting a smartphone, one currently has to (i) obtain information which chipset model is used in this smartphone, for instance from a separate data sheet, (ii) determine which vulnerabilities affect the chipset model based on AOSP, NVD, and CM information, (iii) assess the patch status of the smartphone using OEM changelogs and potentially the AOSP bulletin. Information on chipset vulnerabilities is thus scattered across multiple, disparate vantage points, necessitating a unified knowledge base in order to shed light on the chipset vulnerability lifecycle in Android smartphones.

C. Vulnerability Lifecycle

To systematically analyze the lifecycle Android chipset vulnerabilities, we adapt the linear timeline for generic software vulnerabilities, originally proposed by Schneier [15], to the Android chipset ecosystem as shown in Figure 1. We identify the following four different phases, each aligning with one of our RQs:

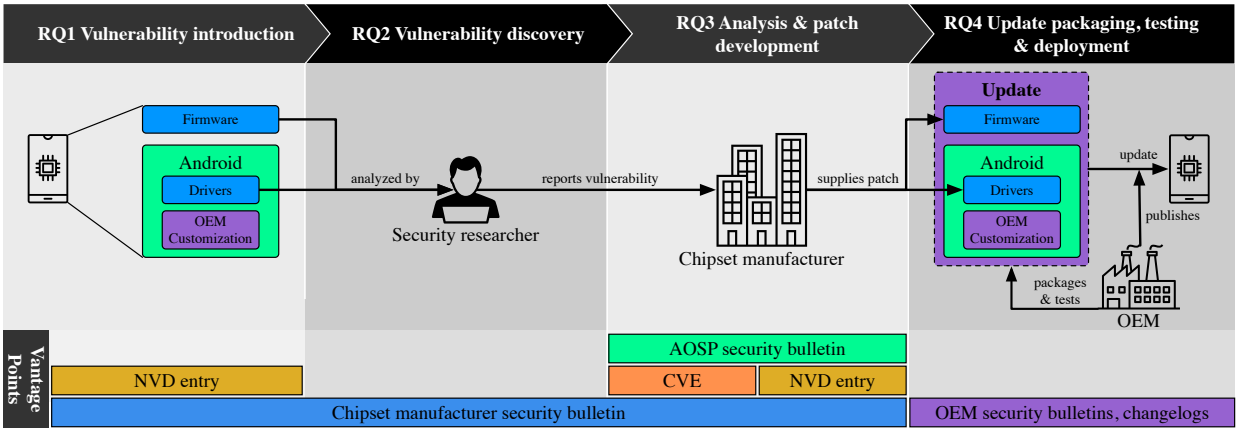

Fig. 1. Lifecycle of chipset vulnerabilities in the Android ecosystem.

Vulnerability introduction (RQ1): The lifecycle starts with a vulnerability being *introduced* into a component. Traditionally many computer programs, such as web-browsers or Android itself, follow a continuous development model, where new features and, potentially, vulnerabilities are iteratively integrated into a single "main" branch of source code, from which software releases are derived [16], [17]. As a result, developers of such software typically only support and enhance the latest (few) version(s) of this branch. In contrast, chipset firmware and drivers are often specific to each chipset model [4]. This means that CMs might develop entirely new firmware and drivers for chipsets at different price points, and for each new chipset generation. For economical reasons, CM might also reuse and adapt parts of the source code of chipset firmware and drivers across different models. Information from NVD entries and CM security bulletins enable assessing which chipset models are affected by a vulnerability. Combining this with information on chipset release dates enables a chronological assessment of a vulnerability being introduced and potentially persisting across chipset generations.

Vulnerability discovery (RQ2): Vulnerabilities are discovered either by the chipset manufacturers themselves, or by external third parties, such as bug-bounty hunters or academic researchers. As typical techniques for vulnerability discovery depend on the targeted component within the chipset, internal and external security experts must decide upon which component they target a-priori. This decision necessitates information on the chipset threat landscape to make an informed decision. Information about the discovery phase is available primarily form CM bulletins, as that is the only vantage point providing information on who discovered a vulnerability as well as the affected component.

Vulnerability analysis & patch development (RQ3): After the discovery of a new vulnerability, CMs must analyze its potential impact and severity and develop a mitigating *patch*, which they then supply to the OEMs. Finally, they publish their analysis result in the form of a CVE (that in turn is also analyzed by the NIST NVD team to assign a CVSS score), a security bulletin on their website, and, potentially, with the next AOSP security bulletin.

TABLE I
EXISTING STUDIES ON VULNERABILITIES AND UPDATES IN THE ANDROID SMARTPHONE ECOSYSTEM, IN COMPARISON TO OUR STUDY.
○= NOT COVERED, ◐= PARTIALLY COVERED, ●= COVERED

	Vantage points					Phases analysed			
Prior Work	AOSP	CVE	NVD	CMs	OEMs	Intro	Discov.	Patch	Update
Acar [7]	●	●	●	○	●	○	○	◐	●
Farhang [18]	●	●	○	○	●	○	○	◐	●
Hou [19]	○	●	●	○	◐	◐	○	○	●
Jones [20]	◐	○	○	○	◐	○	○	○	●
Vásquez [21]	●	●	●	○	○	○	◐	●	○
Zhang [22]	●	○	○	○	◐	○	○	●	◐
Our work	●	●	●	●	●	●	●	●	●

Update packaging, testing & deployment (RQ4): When a vulnerability has been discovered and the OEM has been notified by the chipset manufacturer of a patch, the OEM must assess which of their phones are affected by a vulnerability, integrate driver patches into their version of Android, and package them together with firmware patches into an *update* that they then deploy to the phones in the hands of their customers. The time frame required and rigor applied in these processes is currently unknown, and thus there is currently no single source of information comprehensively summarizing for how long which phone has been exposed to a chipset vulnerability, as OEM security bulletins only contain information on mitigated, but not on unpatched chipset vulnerabilities.

III. Related Work

Android smartphone updates: Previous research, as shown in Table I, has primarily focused on the final phase of the vulnerability lifecycle, specifically the deployment of Android OS updates [18], [20] or device updates that include Android and chipset firmware [7]. Hou et al. [19] focused on the final phase of the vulnerability lifecycle of Android vulnerabilities, but additionally discussed the first phase of the vulnerability lifecycle, investigating who are the developers of vulnerable,

pre-installed apps. Linares-Vásquez et al. focused on the second and third phase of the vulnerability lifecycle [21], investigating how many vulnerabilities affect each Android OS subsystem and how long it takes until vulnerabilities in the kernel are addressed. Lastly, Zhang et al. [22] investigated how patches propagate between Linux and different versions of Android, observing that this process is prone to delays, leaving devices exposed to kernel vulnerabilities for prolonged durations. Notably, no prior work has focused on chipset vulnerabilities or included all vantage points that are required to gain a holistic picture of chipset vulnerability lifecycle, with CMs' websites websites being a common oversight. Furthermore, the early phases of the vulnerability lifecycle have not been studied comprehensively, as existing papers mostly focus on its final phase.

Generic software: There exist studies that have investigated the lifecycle of vulnerabilities in PC software [23]–[27]. Prior work that studied whether vulnerabilities persist across several versions of software have discovered that this heavily depends on the kind of software at hand; for instance, web browsers show very different characteristics compared to operating systems [23], [27]–[30]. There exist few studies investigating how product security teams of software vendors fare in comparison to external security researchers, exclusively targeting Firefox and Chrome [31], [32]. When considering discoveries made by external security researchers, the effectiveness of bug bounty programs is an active research area [27], [33]. However, these works typically investigate bug-bounty programs that have a low barrier of entry by not requiring physical access to devices, such as bug bounties on web applications. Results of prior works might thus not translate to programs that require access to a specific phone, such as bug bounties on chipset software. Lastly, prior works on vulnerability analysis, patch development, and update deployment processes studied the reliability of vulnerability analyses [34]–[36], difficulties in replicating vulnerabilities for verification [37], [38], and update timelines [23], [32].

IV. Method

A. Data Collection

We collected vulnerability information from three independent vantage points: (i) chipset manufacturer security bulletins of Samsung, Qualcomm, Mediatek and Unisoc, (ii) NIST's NVD, and (iii) AOSP security bulletins. This allows us to later uncover any inconsistencies, if present, between the three sources when investigating our research questions. We only consider vulnerabilities of those chipset models that include at least the two basic components required to build a cellular phone, i. e., an application processor, running Android, and a cellular modem. Vulnerabilities in other components of such chipset models are also in scope. We list all components in Section B. This results in information on 3,676 different vulnerabilities across 437 chipset models, released between September 2009 and April 2024. *Our dataset thus includes chipset and vulnerability information on all relevant CMs, resulting in a market coverage of 99% [13].*

We aim to cover a diverse set of OEMs, while ensuring that our analysis results are not heavily skewed by legacy information. To avoid introducing a negative bias from defunct OEMs who have stopped updating their devices, we only included OEMs who have released a new smartphone model since the beginning of 2022. Querying GSMArena[2], a website that independently curates information on smartphones, based on this criterion resulted in smartphone to chipset mappings and release date information covering 6,866 smartphone models made by 38 OEMs. Notably, only four out of 38 OEMs (Samsung, Google Pixel, Tecno, and Fairphone) provide comprehensive changelogs of all their devices on their websites, with a third-party website offering data for Xiaomi. Three of these OEMs rank among the top four in Q2 2023 Android smartphone sales [39], while Google Pixel and Fairphone have negligible market shares [40], [41]. We therefore also include information on 16,139 device updates - covering individual device updates as well as information from OEM Security Response Centers - for all smartphone models manufactured by Samsung, Tecno, and Xiaomi into our dataset. *Our data thus covers the majority of the Android market regarding information on which chipset is used in which phone, as well as device update information [39].* Additional technical details of this process are described in Section A, with Table IV providing a summary of vantage points.

B. Data Augmentation

Since the scope of our RQs requires information that goes beyond the data immediately contained within the vantage points, we augment our dataset by inferring additional information before aggregating all collected information into a

Affected Component: Chipset manufacturers do not provide information on which component was affected by a vulnerability that uses consistent names throughout different manufacturers. Instead, CMs often refer to components by internal marketing or code names. For instance, the Trusted Execution Environment (TEE) used in Mediatek chipsets is called 'Kinibi', while Qualcomm refers to their TEE as 'QSEE'. For this reason, we identified a set of such key terms for each component based on source code fragments, data sheets, and marketing material published by each CM. We then use these key terms to automatically identify a vulnerability's component based on information provided in CM security bulletins, normalizing to a common component name that we use across all CMs (e. g.,'Trust' in the example above), according to the list in Appendix Section B.

Vulnerability Location: The *location* specifies whether the vulnerability is located within a driver executing as a part of Android, or if it the vulnerable code is contained in the component's firmware, executing on a separate co-processor. We identify a vulnerability's location using the same approach as we use for components.

External reports: Only Qualcomm explicitly distinguishes between external and internal sources in their security bulletins, while Mediatek, Unisoc, and Samsung credit external

[2]https://GSMArena.com

discoverers, but not their own employees, by name for each vulnerability if the researcher consents to this. Vulnerabilities with named discoverers are, therefore, always external discoveries. Depending on how many external reporters want to remain anonymous, the true number of externally found vulnerabilities might be higher than the estimated heuristic for Samsung, Mediatek, and Unisoc. As we will see in Section V-B1, this possible under-approximation does not invalidate but strengthens our empirical results.

Knowledge Base: Afterwards, we build our knowledge base by combining information from various vantage points to fit the canonical schema illustrated in Figure 10 (Appendix Section C). To do so, all collected and augmented data is stored in a relational database. We link individual data points using common identifiers, such as CVE IDs for vulnerabilities or model numbers for chipsets. This allows us to correlate and compare information obtained from different vantage points.

V. Empirical Analysis

A. RQ1: Vulnerability Introduction

Where do vulnerabilities in chipsets originate?

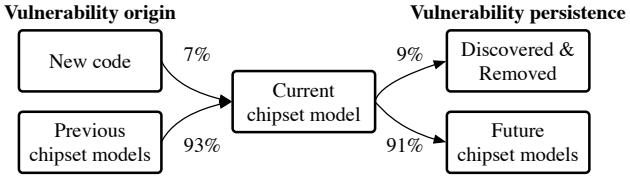

Fig. 2. Origin and persistence of discovered vulnerabilities in the average chipset model. Arrows symbolize the direction in which vulnerabilities propagate.

1) Vulnerability origin: To assess if vulnerabilities introduced through *code re-use* from previous chipset generations outweigh newly introduced ones, we measure the amount of vulnerabilities that have been newly introduced into every chipset, and compare this to the amount of all vulnerabilities affecting each chipset. To do so, we utilize the information about which chipsets are affected by a vulnerability, as published in CM security bulletins and NVD entries, combined with the release date of each chipset. We associate each chipset $c \in C$ with a set of vulnerabilities $V(c)$ that affected the chipset, and the chipset's release date $T_{\text{rel}}(c)$. Then a vulnerability v has been *newly introduced* in chipset c iff. $\forall c' \in C.(v \in V(c') \rightarrow T_{\text{rel}}(c') \geq T_{\text{rel}}(c))$.

We observe that each chipset c is affected by $|V(c)| = 204$ vulnerabilities on average, with a median of 149. However, on average, only 7% of the vulnerabilities affecting a chipset have been newly introduced in a particular generation of chipset models, while 93% of vulnerabilities are inherited from previous generations, as shown on the left side of Figure 2. In the median case, a new chipset model does not introduce any new vulnerabilities. Thus, the distribution is heavily skewed, and there are a few chipset models that, upon release, introduce the majority of new vulnerabilities. The primary origin for vulnerabilities in newly released chipset models are thus vulnerabilities inherited from previous chipset generations through code re-use, rather than new vulnerabilities, which only occur in very specific chipset model releases.

We also found that the introduction of support for new cellular, WiFi or Bluetooth protocols does not always lead to the highest percentage of newly introduced vulnerabilities. Instead, we observe that chipset models supporting identical protocols have vastly different shares of newly introduced vulnerabilities. For instance, we discovered that Qualcomm's SM8475 chipset exhibited 21% newly introduced vulnerabilities, surpassing the 6% found in its predecessor, the SM8450, despite both supporting the same cellular, WiFi and Bluetooth protocols. In comparison, Qualcomm's first 5G supporting chipset models SM4350 (7.5%), SM8150 (8%) and SM8350 (13.5%) also exhibit a smaller share of newly introduced vulnerabilities than the SM8475. The same holds for other CMs: 18% of the vulnerabilities in Mediatek's MT6889, released in the first quarter of 2020 and their first model to support 5G, were newly introduced. The MT6762, only supporting LTE and released in 2018, a time when LTE was already widely adopted, saw a comparable rate of 17.5% of new vulnerabilities. Therefore, we suspect that, while the introduction of 5G led to above-average rates of newly introduced vulnerabilities, similar percentages of newly introduced vulnerabilities can be likewise caused by internal changes made by the CM.

2) Vulnerability persistence: We determine whether vulnerabilities *persist* from one chipset generation to the next. Formally, if $T_{\text{patch}}(v)$ is the date at which vulnerability v has been patched, we consider v to persist into chipset c iff v was not newly introduced in c and $T_{\text{rel}}(c) \leq T_{\text{patch}}(v) \wedge v \in V(c)$ holds. We find that, on average, only 9% of vulnerabilities of a chipset model are removed before the release of the next chipset model by the same CM ($T_{\text{rel}}(c) \geq T_{\text{patch}}(v)$), while 91% of vulnerabilities persist ($T_{\text{rel}}(c) < T_{\text{patch}}(v)$), as shown on the right side of Figure 2. The distribution is heavily skewed, as in the median case no vulnerabilities are removed. Combined with our previous results, we observe a decreasing trend in the amount of vulnerabilities in chipsets: On average, each new chipset generation leads to the addition of 6% of new vulnerabilities, while 9% of the vulnerabilities are removed.

Summary RQ 1: Vulnerability introduction

Observations:
- For new chipset models, 93% of vulnerabilities are inherited from previous chipset generations, rather than newly introduced by novel features.
- On average, new chipset models tend to exhibit less vulnerabilities than the previous generation, as there are fewer vulnerabilities introduced than discovered.

New insight:
- Vulnerabilities affecting cutting-edge chipset models may be found by testing legacy devices, demonstrating that it is often-times unnecessary to acquire new devices to find practically relevant vulnerabilities.

B. RQ2: Vulnerability Discovery

Who discovers vulnerabilities in a chipset?

1) Growth in vulnerability discovery: We depict the amount of newly assigned CVEs per chipset manufacturer and year in the bar-chart of Figure 3. Overall, we notice that the amounts of yearly discovered vulnerabilities affecting Qualcomm and Samsung chipsets have stayed constant since 2018, while Mediatek and Unisoc chipsets have seen a significant increase in discovered vulnerabilities. At the same time, we see that Mediatek and Unisoc discover almost no vulnerabilities internally, as illustrated by the line-chart in Figure 3. The growth in absolute numbers of discovered vulnerabilities for both CMs is thus driven by increased scrutiny from external researchers. We see two potential reasons for this increase in external discoveries: First, while Samsung's and Qualcomm's market shares have stayed the same, Mediatek and Unisoc significantly increased their market shares in the period from 2018 to 2023 [2], likely attracting more research on their products. Second, while Samsung and Qualcomm have maintained bug bounty programs since 2017 [42], [43], Mediatek and Unisoc vulnerabilities only became eligible for bug bounties in 2023 [44], [45], potentially further incentivizing external researchers to discover vulnerabilities in their products.

2) Internal discovery rates: In 2023, the products of Mediatek and Qualcomm were both eligible for similar bug bounties [43], [44], and for both CMs around 300 vulnerabilities were published. However, based on the information in CM security bulletins, Qualcomm was able to discover 57% of these internally, while Mediatek only found 10% internally. Even more strikingly, Unisoc published 465 vulnerabilities, of which they found only around 7% internally. One would expect that the ratio between internal and external discoveries would be roughly equal for all CMs, as long as there are no external factors (e. g., bug bounties) that would pull external researchers more towards particular CMs. There are two potential reasons for the observed discrepancy between CMs: Either, Mediatek and Unisoc indeed almost never find vulnerabilities internally, or they find more vulnerabilities internally than they publicly disclose. We also observe a positive trend: Samsung's ratio of internally discovered vulnerabilities has more than doubled from 25% in 2022 to 60% in 2023. Although it might be an outlier, as was 2020, this spike could be caused by increased scrutiny following critical vulnerabilities uncovered by Google Project Zero [46] and Samsung's establishment of a dedicated chipset product security team in 2023 [47].

3) Number of discovered vulnerabilities by component: To understand if the probability of discovering a new vulnerability differs between a chipset's components, we group the number of discovered vulnerabilities by component and manufacturer. The result is shown in Table II. Most vulnerabilities are discovered in cellular connectivity followed by WiFi and GPUs. This holds true when looking at all manufacturers combined, as well as for every manufacturer independently, except for Samsung. Table II also allows us to determine the components in which chipset manufacturers' internal security teams are predominantly discovering vulnerabilities. We observe that, similar to the overall discovery rates, Qualcomm and Samsung surpass the internal discovery rates of Mediatek and Unisoc on almost all components, underlining that their internal discovery rate is universally higher.

TABLE II
NUMBER OF TOTAL VULNERABILITIES GROUPED BY COMPONENT AND MANUFACTURER. PERCENTAGE OF INTERNALLY DISCOVERED VULNERABILITIES SHOWN IN PARENTHESIS.

Chipset component	Samsung	Qualcomm	Unisoc	Mediatek	Total
Cellular	54 (57.4%)	305 (78.4%)	173 (13.9%)	70 (20.0%)	602
WiFi	4 (0.0%)	356 (59.3%)	91 (1.1%)	111 (13.5%)	562
GPU	21 (23.8%)	254 (58.7%)	26 (0.0%)	107 (5.6%)	408
Trust	29 (37.9%)	186 (79.6%)	2 (0.0%)	42 (31.0%)	259
Audio	0 (-)	90 (53.3%)	11 (0.0%)	19 (47.4%)	120
Vision	2 (0.0%)	39 (51.3%)	20 (5.0%)	40 (10.0%)	101
Bluetooth	3 (66.7%)	53 (56.6%)	7 (0.0%)	22 (13.6%)	85
Debug	3 (0.0%)	32 (50.0%)	24 (4.2%)	15 (0.0%)	74
Boot	12 (8.3%)	37 (48.6%)	0 (-)	16 (18.8%)	65
IPC	1 (0.0%)	33 (84.8%)	0 (-)	29 (24.1%)	63
Machine learning	10 (30.0%)	10 (10.0%)	1 (0.0%)	41 (2.4%)	62
Position	0 (-)	18 (61.1%)	4 (25.0%)	17 (0.0%)	39
Memory management	3 (33.3%)	7 (57.1%)	3 (0.0%)	24 (12.5%)	37
Power	0 (-)	6 (83.3%)	17 (0.0%)	11 (36.4%)	34
Virtualization	1 (100.0%)	11 (72.7%)	0 (-)	2 (50.0%)	14
NFC	0 (-)	4 (75.0%)	0 (-)	0 (-)	4
Total	143	1,441	379	566	2,529

Summary RQ 2: Vulnerability discovery

Observations:

- The amount of internally and externally discovered vulnerabilities vary greatly between CMs, with two CMs discovering less than 15% of reported vulnerabilities internally.
- There exists a large disparity in number of discovered vulnerabilities between components, with cellular and WiFi vulnerabilities being discovered most frequently.

New insights:

- Independent bug bounty hunters, security research companies and academic researchers, rather than in-house product security teams, appear to be the driving force behind vulnerability discovery in the chipsets of some CMs.
- The increase in discovered vulnerabilities in recent years is due to external researchers targeting more CMs, rather than improving techniques for a single CM's chipsets. This suggests that adapting existing techniques to additional CMs' products is more effective than refining techniques for one CM.

C. RQ3: Vulnerability Patching

When are patches available and how severe are the chipset vulnerabilities they mitigate?

1) Vulnerability severeness in firmware and drivers: It has previously been speculated that vulnerabilities in firmware, running on chipset processors outside of the Android OS, are more severe than vulnerabilities in drivers. Prior work argues

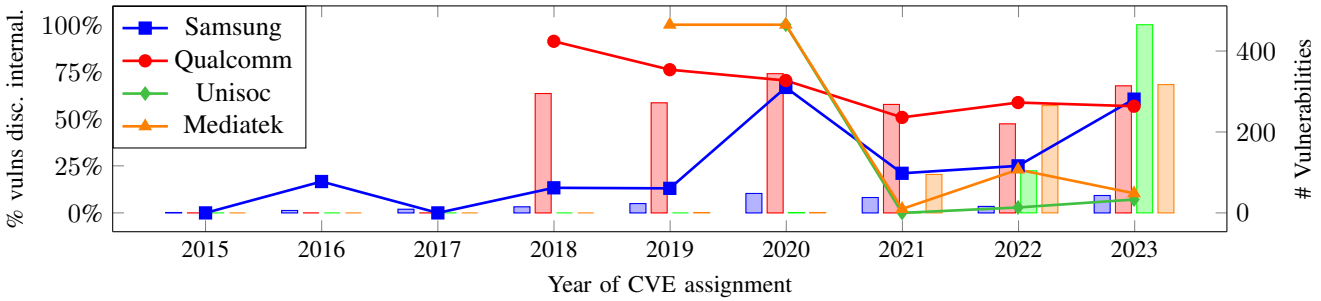

Fig. 3. Vulnerabilities published per year per CM. Bars show the total number of published vulnerabilities, lines show the fraction of vulnerabilities discovered internally by each CM.

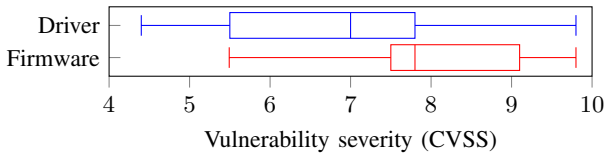

Fig. 4. Severity distribution of firmware and driver vulnerabilities. Severity information is based on NIST analysis.

that this is because drivers, interfacing with chipset processors from inside Android, inherit security features from Android, which firmware does not [6]. We now validate whether this claim can be verified empirically by comparing severity scores of driver and firmware vulnerabilities in Figure 4. While the most severe vulnerabilities of firmware and drivers are capable of enabling a whole system compromise, we observe that the medians of both populations differ, with firmware vulnerabilities showing a median that is 0.8 CVSS points higher than driver vulnerabilities, implying that, on average, firmware vulnerabilities are indeed more severe than driver vulnerabilities. We confirmed this statistically significant difference at $p = 0.05$ using a Kruskal-Wallis test.

2) Time until patch availability: To verify how long it takes chipset manufacturers to develop a patch and provide it to OEMs, we compare the date $T_{\text{report}}(v)$ a vulnerability v has been (externally) reported to the date the chipset manufacturer notified the OEM and provided a patch, or they published the vulnerability on their website $T_{\text{patch}}(v)$. Here we are particularly interested in whether the industry-standard 90-day responsible disclosure period is adhered to, i. e., $T_{\text{patch}}(v) - T_{\text{report}}(v) \leq 90$ days. As evident from Figure 5, neither Qualcomm nor Samsung can consistently meet the 90-day responsible disclosure deadline. Within 90 days, Samsung has patched 46.9%, while Qualcomm has patched 19.9% of vulnerabilities reported by external researchers. Instead, Samsung requires 185 days to address 95% of their vulnerabilities. For Qualcomm, the discrepancy is even more significant, as they require 348 days to supply patches for 95% of their vulnerabilities. Even more worrisome, we observe that a few remaining outlier vulnerabilities required several years to be addressed. There is no strict correlation between the affected component and the time it takes to develop a patch. Interestingly, our data also shows there is no meaningful correlation between vulnerability severity and patching time frame. This leads us to conclude that high severity vulnerabilities are seemingly not prioritized.

3) Information availability and consistency: After a patch has been developed, information on the vulnerability is published by CMs. To verify whether all vulnerability information sources provide the same coherent information, we first compare where vulnerabilities are being published and then assess if the information across all sources is identical. This is important, as inconsistent information on the same vulnerability might lead to confusion and a misjudgement of a vulnerability's severity by OEMs, researchers, and end-users. To do so, we leverage the fact that our knowledge base aggregates information from the NVD, CM's, and the AOSP in directly comparable structure, including publishing dates.

Availability: First, we assess if manufacturers reliably publish vulnerabilities on all three platforms. Our goal is to capture a recent, rather than a historical, picture of the vulnerability publication practises. Over our entire data set, we found that one year is sufficient time to propagate between vantage points for 99.9% of all vulnerabilities that are eventually included in all vantage points. We therefore base our report on vulnerabilities published in the time frame from June 2022 to May 2023, such that the vulnerabilities published in May 2023 had until April 2024, i. e., one year, to propagate to all vantage points. Vulnerabilities that have not reached all vantage points within this time frame are, according to our data, extremely unlikely to propagate at all ($< 0.1\%$ of cases). The result of this measurement is shown in Figure 6. From this chart, it becomes clear that vulnerabilities published on chipset manufacturer websites are almost always also published in the NVD, and only very few Samsung vulnerabilities are missing from the NVD. Notably, no vulnerabilities are missing from the CM websites, i. e., vulnerabilities published in CM security bulletins are a superset of the vulnerabilities published via all other vantage points. However, we notice that Unisoc and Mediatek publish less than 25% of chipset vulnerabilities in AOSP security bulletins, and Samsung has not published any vulnerability in the AOSP bulletins. The AOSP security bulletins thus are incomplete, and must be augmented with CM (or NVD) website information to gain a holistic picture on which vulnerabilities affect a smartphone.

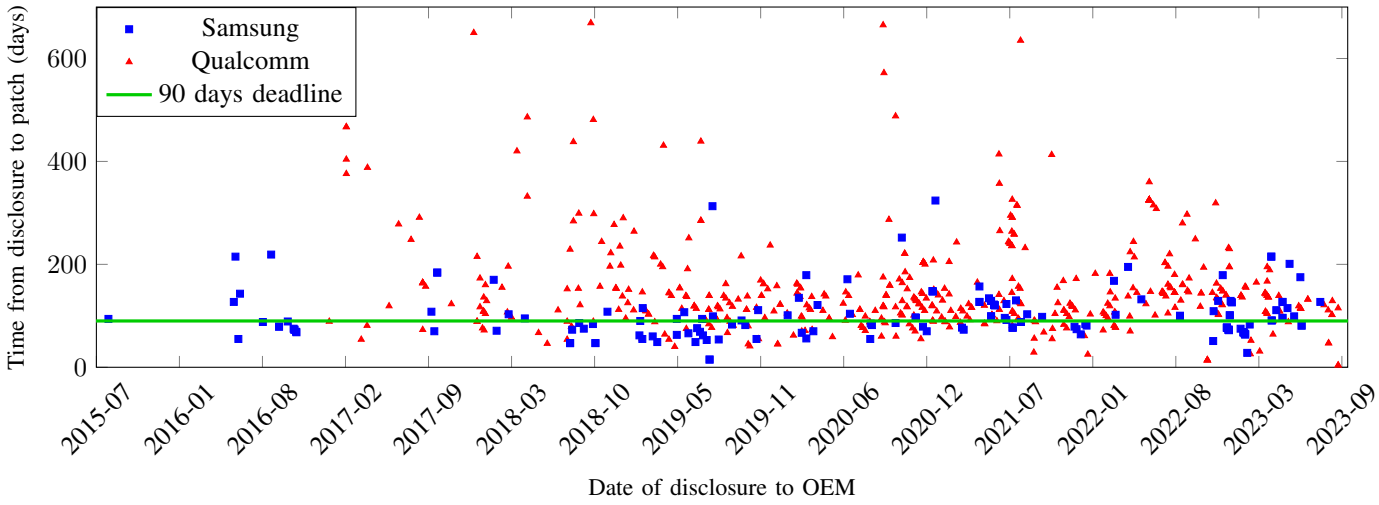

Fig. 5. Time frame from vulnerability disclosure to patch availability, each marker representing a single vulnerability. Graph cut-off at $y = 700$ days for readability. Mediatek and Unisoc are excluded due to missing discovery date information.

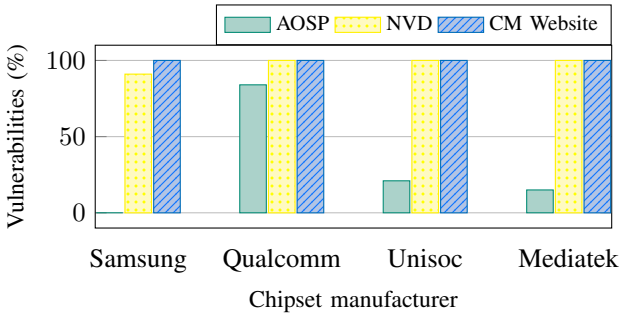

Fig. 6. Comparison of vulnerability information availability.

Consistency: As described in Section II-C, not only is the severity of a vulnerability analyzed by the chipset manufacturer but also by NIST, which assigns a CVSS score independently. This analysis by NIST has recently been criticized for exaggerating the severity of vulnerabilities in open source projects [48]–[50]. However, out of the 2,249 vulnerabilities for which we obtained a CVSS severeness rating from the CMs' websites as well as from NIST, we observe that for 10% of these vulnerabilities NIST assigned a lower severity than the CM, and in 15% of the cases NIST assigned a higher severity. In the realm of chipset vulnerabilities, our data thus does not support the aforementioned criticism that NIST systematically exaggerates severities.

Summary RQ 3: Vulnerability patching

Observations:

- On average, vulnerabilities in chipset firmware are more severe than driver vulnerabilities.
- The 90-day responsible disclosure period is commonly not adhered to by Qualcomm and Samsung.
- Vulnerability information is often missing from monthly AOSP security bulletins.

New insights:

- A 90-day disclosure period is inapplicable to vulnerabilities in chipset drivers and firmware, given that it is violated on a regular basis. Researchers should not rely on vulnerabilities being addressed within 90 days to assess when to publicly disclose vulnerabilities.
- AOSP security bulletins lack comprehensive information on chipset vulnerabilities. This could mislead users and researchers alike to underestimate the amount of vulnerabilities affecting a device.

D. RQ4: Vulnerability Updating

> *What are the characteristics of the update process utilized by OEMs to address chipset vulnerabilities in Android devices?*

1) Number of affected smartphones: The amount of smartphone models affected by a chipset vulnerability v is determined by two factors: (i) how many chipset models are affected by the vulnerability $|\{c \in C | v \in V(c)\}|$ and (ii) the amount of smartphone models $s \in S$ that contain one of these affected chipset models. Assuming that $B(s) = c$ iff. smartphone model s uses chipset model c, then the number of smartphones affected by v is $|\{s \in S | v \in V(B(s))\}|$. We illustrate the resulting number of affected phones per vulnerability and CM in Figure 7. Overall, the median number of

affected phones per Mediatek vulnerability (652 phone models) is significantly higher than per Qualcomm vulnerability (277 phone models). Similarly, the most widespread Mediatek vulnerability affected 2,222 smartphone models, while the most prevalent Qualcomm vulnerability affected 1730 smartphone models. In contrast, vulnerabilities affecting Samsung or Unisoc chipsets generally affect the fewer different smartphone models. This is because Samsung predominantly produces chipsets for use in their own smartphones, as well as very few smartphone models made by HTC, Meizu and Motorola. This limits the amount of potentially affected smartphone models to approximately 450. Likewise Unisoc, being a relatively new CM, only produces chipsets for use in 277 different smartphone models, although supplying to 23 different OEMs.

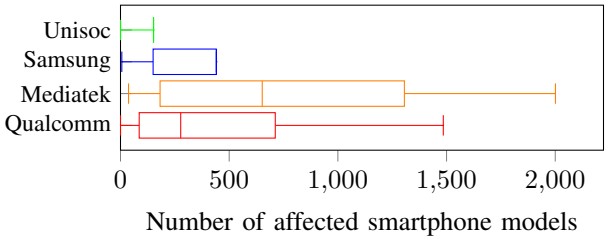

Fig. 7. Number of affected smartphones by a vulnerability, per CM.

These results show that a single vulnerability usually has quite drastic ripple effects, as it typically affects multiple chipset models through code re-use (cf. Section V-A), that are used in a variety of smartphone models. The extent of this effect depends on the CM of the chipset.

2) Update availability: CM's may release security advisories containing vulnerability information before OEMs have packaged the patch for the published vulnerabilities into a smartphone update. This would enable threat actors to analyse the, now publicly known, vulnerability and potentially use this information to built exploits, putting the security of end-users at risk. To quantify this risk, we assess how many vulnerabilities in our dataset seem to have never been mitigated by an update and may still affect devices. All of these vulnerabilities fulfill the following criteria:

- affect at least one smartphone model for which we have update information
- are never included in any update changelog, or AOSP security bulletin referenced by an update changelog
- published after at least one phone with an affected chipset, for which we have update information, has been released
- published before January 2023, to give smartphone manufacturers on a bi-annual update schedule enough time to provide an update

Only 951 (60.2%) out of 1,546 vulnerabilities matching these criteria have received at least one mitigating update. For the remaining 631 (40.8%) vulnerabilities, affected users cannot assess whether their devices are still exposed.

3) Update timeline: For the vulnerabilities that have been addressed by updates, we assess for how long end-users have been at risk before an update has been available for their devices. To this end, for every vulnerability addressed by a device update, we calculate the time from vulnerability announcement on the chipset manufacturer's website until the corresponding update became available. Formally, we define $T_{\text{OEM}}(v, s)$ as the point in time when an OEM has released an update containing the patch that mitigates vulnerability v in smartphone s, and then compute $T_{\text{OEM}}(v, s) - T_{\text{patch}}(v)$ for all $(v, s) \in S \times V$ with $v \in V(B(s))$. We analyzed all 24,226 pairs of vulnerabilities and affected smartphones (v, s) in our knowledge base. The majority of phones will receive an update in under 3 months after a vulnerability is published, as 25% of affected smartphone models have received an update within 44 days after a vulnerability has been published, and within 71 days 50% have received updates. However, we observe that updates are sometimes heavily delayed, with the 95% quantile at 266 days. We also observe that updates for the same vulnerability do not reach all smartphones at the same time. Instead, the median of the interval between the first phone model receiving an update and the last affected phone model receiving an update $\max_{s,s' \in S} T_{\text{OEM}}(v, s) - T_{\text{OEM}}(v, s')$ is at 182 days. When we compare the median interval between the first phone model to receive an update, and half of all phones having received an update, we observe a 32 day delay.

Our study thus illustrates two things: First, the update process is not well coordinated between OEMs and CMs, as the CMs publicly announce the existence of a vulnerability before OEMs are able to provide an update to end-users. This allows threat actors to analyze published vulnerabilities, using the time-frame between vulnerability publication and update availability for exploitation. Secondly, it shows that the update process is fragmented, as not all phone models receive an update at once. This could enable threat actors to obtain an update of a phone model that received a timely update and analyze it to gain additional information on exploitability of phone models that are yet-to-receive updates.

Summary RQ 4: Vulnerability updating

Observations:

- A single vulnerability typically affects hundreds to thousands of different smartphone models.
- 41% of vulnerabilities affecting devices in our knowledge base are never addressed in any changelog.
- Updates reach end-users with a median delay of 71 days after vulnerability publication, with some updates being delayed by more than 8 months.

New insights:

- End-users are left in the dark regarding a large amount of chipset vulnerabilities, which are seemingly never mitigated via an update.
- Vulnerability announcements by CMs and update availability by OEMs are not well coordinated. This leaves users at risk, when a CM publicly announces a vulnerability, but the OEM takes another 2-8 months to provide an update to end-users.

VI. Comparative Analysis

We now compare our findings to the results of similar studies in other domains, highlighting key differences, the importance of a better understanding on vulnerability management of chipsets and the gaps that our proposed unified knowledge base fills.

A. Vulnerability Introduction

In our analysis, we found that code re-use is *the* primary origin of chipset vulnerabilities. As shown in Figure 8, code re-use commonly leads to the introduction of vulnerabilities in other ecosystems as well. However, chipset firmware and drivers seem to be particularly prone to this, as their percentage of vulnerabilities from code re-use follows closely those of Firefox [28] and OpenJDK [27]. Firefox however follows a rapid release development paradigm, with a new version released every four weeks [16], limiting the amount of potential code changes that can be made in this short time frame. In contrast, chipset development operates on a longer cycle, typically releasing new generations annually with derivative versions appearing every few months. Notably, chipset firmware and drivers are the closed-source products exhibiting the highest proportion of vulnerabilities due to code re-use, compared to Android apps [30], closed-source PC software [29], and Microsoft Windows [31]. *This implies that transitions between versions in closed-source products often result in distinct sets of vulnerabilities for each version. In contrast, chipset firmware and drivers seem to often inherit vulnerabilities, potentially due to chipsets' necessity for backward compatibility with older specifications, such as cellular, Bluetooth, or WiFi standards.* This need for compatibility mirrors that of programming languages like Java (OpenJDK) and PHP, presumably explaining the similarity in vulnerabilities related to code re-use.

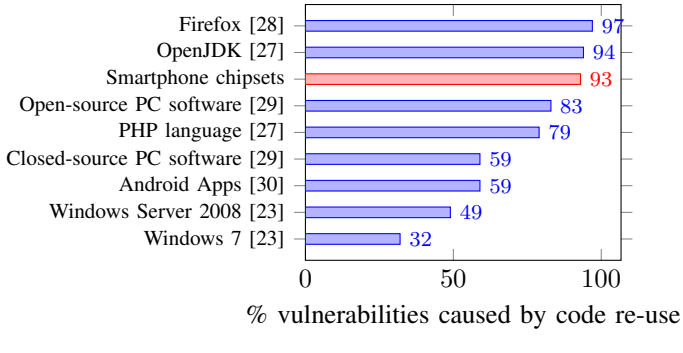

Fig. 8. Prevalence of vulnerabilities inherited from previous software versions through code re-use.

B. Vulnerability Discovery

Most affected components: As illustrated in Table II, we found that most vulnerabilities in chipset models are discovered in the cellular modem, followed by WiFi, GPU and trust related functionality. In their study on vulnerabilities in the Android ecosystem, Linares-Vásquez et al. [21] reported that most vulnerabilities found in Android drivers stem from the GPU, WiFi or the camera subsystem, with the cellular modem and trust components never being mentioned. We suspect that this is because the authors obtained their information from the AOSP bulletins which, as we showed in Section V-C3, lack information on the majority of chipset vulnerabilities. *Prior work therefore severely underestimated the amount of vulnerabilities in various chipset components, such as cellular modems or trust-related components. In contrast, our knowledge base incorporates information from multiple vantage points, enabling us to compile a comprehensive statistic of affected components.*

Internal versus external discovery: In Section V-B1, we observed that Mediatek and Unisoc find less than 15% of vulnerabilities in their products internally. Compared to web browsers, another type of high-risk targets, this rate of internally discovered vulnerabilities appears to be strikingly low. Sivagnanam et al. [31] measured that 56% of all Chromium vulnerabilities are found internally by the Chromium development team, Google employees or automated processes. Similarly, Atefi et al. [32] find that 69% of all Firefox vulnerability reports come from the Firefox development team internally, rather than external researchers. This shows that, rather than Qualcomm and lately Samsung having an outstandingly well performing internal security team, *Mediatek and Unisoc seem to be under-performing w. r. t. internally discovered vulnerabilities, compared to other high-impact and high-exposure product categories, such as web browsers.*

Impact of bug bounties: Several studies [27], [33] have shown that in general, the establishment of bug bounty programs leads to an initial increase in discovered vulnerabilities for products covered by the bounty, followed by a decline in discovered vulnerabilities. The authors argue that this is a sign of increasing code quality, as it becomes harder for bug-bounty hunters to discover vulnerabilities. Given Qualcomm's and Samsung's bug bounty programs have existed for 6 to 8 years, one would expect to see a similar decrease in discovered vulnerabilities in their products. However, the amount of vulnerabilities discovered in Qualcomm chipsets is constant, while the quantity of discovered vulnerabilities for Samsung chipsets is following an increasing trend. *Therefore, it seems that the current bug-bounty programs run by CMs do not significantly reduce the amount of chipset vulnerabilities.* We suspect that this might be due to two factors. First, discovering vulnerabilities in chipsets commonly requires access to hardware for testing purposes (smartphones, software-defined radios etc.), a requirement that does not apply for almost all other software in bug bounty programs and might deter bug hunters. Secondly, chipset vulnerabilities are relatively valuable to parties competing with bug bounty programs for the attention of qualified security researchers, such as exploit brokers. For instance, Zerodium pays up to 200,000 USD for a remote code execution vulnerability in a cellular modem, or up to 100,000 USD for a remote code execution vulnerability in a WiFi component [14].

C. Patch Development

In Section V-C2 we found that Qualcomm and Samsung are, on average, able to provide patches for discovered security vulnerabilities within the 90-day responsible disclosure period. This, again, is in stark contrast to the browser ecosystem. On average, the Firefox and Chrome developer teams are able to develop a patch even within 80 days or less [32], demonstrating that the 90 days disclosure period is not entirely unrealistic in other ecosystems. Even more strikingly, Zhao et al. [33] report that 50% of vulnerabilities reported via bug bounty programs in the web ecosystem are patched within 7 days. In fairness to the CMs, there are several additional steps in the patch development process of chipsets, compared to browsers or web applications: Web browsers or websites do not have do undergo integration tests for hundreds of different phone models, as they typically only target a handful of operating systems. They also do not have to potentially undergo the same regulatory (re-)certification for radio emitters and protocol compliance that chipsets do. *Nonetheless, this highlights that patching chipset vulnerabilities takes significantly longer than patching other high-value targets.*

D. Updates

Web browsers and operating systems: As illustrated in Section V-D3, OEMs require 52 days to supply 50% of affected smartphone models with an update after a vulnerability has been published by a CM. This sharply differs from other types of software. Shazat et al. [23] found that 96% of Chrome and 58% of Firefox vulnerabilities receive a mitigating update on the day they are published, while Microsoft and Apple are able to immediately provide updates for their operating systems for 76%-78% of the vulnerabilities they publish. We conclude that the additional 52 days required by OEMs are due to the supply chain from CM to OEM that does not exist for web browsers or operating systems, where the party developing a patch is also packaging the patch into an update.

Android updates: When comparing our findings to prior work on Android device updates, we observe that prior work has been underestimating the amount of vulnerabilities impacting a smartphone. Acar et al. [7] have highlighted, as part of a case study, that the Qualcomm SM8350 used in the Samsung Galaxy Z Fold3 5G phone suffers from 11 chipset vulnerabilities. However, in their study, the NVD CVE database was used as the sole source, where only one of two possible CPE identifiers to filter for CVEs affecting the SM8350 was employed. Our unified knowledge base reveals that this chipset is actually affected by, at least, 131 vulnerabilities, an increase by more than a factor of 11. This highlights the relevance of cross-validating information from several vantage points, thus validating our proposal for the need to consolidate all the information in one single place. Prior work on Qualcomm chipset vulnerabilities [18] also found that, while some Qualcomm chipset vulnerabilities are never resolved in specific phones, every CVE is at least addressed by one OEM. In contrast, we observe that for 43% of CVEs no OEM publishes an update, and thus highlight that the situation has been previously underestimated. A large contributing factor to this underestimation is that Samsung, Unisoc and Mediatek - absent from studies in prior work - only report very few of their vulnerabilities in AOSP security bulletins. However, in the updating phase of the vulnerability lifecycle OEMs rely heavily on these AOSP bulletins to communicate which vulnerabilities have been addressed in their devices, and thus *the absence of chipset vulnerabilities in AOSP bulletins results in missing information for end-users.*

VII. Discussion

A. Changes in the Android ecosystem

We observe two primary ways in which chipsets directly impacted security-relevant changes in the AOSP within the last years. First, the Android version developed by the AOSP is heavily customized by the CMs and OEMs to run on the devices' chipset. Prior to Android 8, every Android update had to be customized for each phone model, such that it properly runs on the respective phone model's chipset. As our results show, resolving 95% of all chipset-related security vulnerability takes OEMs and CMs between 180 days and a full year. To reduce the dependence of Android system updates on CMs, the APIs that are used within the chipset-specific customizations have been gradually standardized since Android 8 [51]. This decoupling allows OEMs to deliver Android updates without waiting for CMs to adapt the new version to the chipset, accelerating deployment of general Android system updates.

Secondly, AOSP offers to include vulnerabilities affecting chipsets into the Android Security Bulletins to mitigate some of the heterogeneity and offer users a CM and OEM independent way of assessing the patch level of their device [52]. According to our knowledge base, 35% of all chipset vulnerabilities were included in AOSP security bulletins in 2019 and the inclusion ratio rose to 59% in 2021, but has since declined to 25% in 2023. This drop is caused by increasing number of discovered vulnerabilities affecting Mediatek and Unisoc, who do not publish all their vulnerabilities in AOSP bulletins. As we have seen, vulnerabilities included into AOSP bulletins are less likely to be never mitigated. In this sense, the AOSP bulletins are an organizational measure to improve transparency, but also positively correlate with update availability in cases where CMs commit to publishing all vulnerabilities within these bulletins.

B. Recommendations to Industry

Bolster CMs' internal security teams. Our study shows that vulnerabilities exist in several chipset generations before discovery and remain in the firmware and drivers until identified (cf. Section V-A). There is an urgent need for increased efforts and resources for faster vulnerability discovery. Currently, half of the CMs discover less than 15% of vulnerabilities internally, relying on external researchers (cf. Section V-C). This contrasts with the security practices of other high-impact targets like web browsers (cf. Section VI). We recommend that CMs with low internal discovery rates conduct internal

security testing, such as employing a "red-team" of penetration testers, and publish all internally found vulnerabilities in their security bulletins.

Prioritize chipset firmware. Vulnerabilities in firmware are more severe than driver vulnerabilities, putting devices at greater risk (cf. Section V-C). We recommend that CMs deploy appropriate technical measures to reduce the impact of firmware vulnerabilities. Potential technical measures that could reduce the severity gap between drivers and firmware include defense techniques such as process isolation and memory-tagging for address sanitization. These are already deployed in Android itself [53] and thus promote driver security, but are often missing in chipset firmware [4].

Ensure completeness of AOSP security bulletins. Information on previously discovered vulnerabilities, patches, and updates is often unavailable or scattered across multiple sources. We find the incomplete information in AOSP security bulletins particularly problematic, as they are the premier way for Android users and researchers to determine if a device has received all the latest security updates. If particular vulnerabilities never occur in these bulletins, it is impossible for users to judge the security of their device. Per our analysis, this is the case for more than 75% of vulnerabilities affecting chipsets of some CMs (cf. Sections IV-A, V-C and V-D). Completeness of AOSP bulletins could be achieved by enforcing that CMs report all vulnerabilities to the AOSP by making this a requirement for Android compatibility certifications.

Establish an industry-wide responsible disclosure timeframe. While the 90-day responsible disclosure period might be insufficient for hardware deployed in hundreds of different device models, we suggest that an appropriate timeframe (around 200 days, as per our analysis) should be established and followed to minimize outlier vulnerabilities that remain unpatched for extended periods. Ideally, this embargo timeframe would not only include the patch development, but also the update packaging phase of the vulnerability lifecycle, as we have demonstrated in our analysis that this phase adds another significant delay until updates finally arrive on end-users' devices (cf. Section V-D), putting them at risk of threat actors developing exploits based on publicly available vulnerability information.

C. Use Cases in Research

Our data enables multiple practical use cases that we believe to benefit the research community.

More representative evaluations. Evaluating novel vulnerability discovery techniques warrants a representative set of devices to empirically test the success of said techniques. As we observed in Section V-C, many chipset models share the same vulnerabilities through code re-use. Manually testing chipsets affected by mostly overlapping sets of vulnerabilities is time consuming, unnecessarily expensive and thus inefficient. Our website offers a tool (www.chipsets.org/devices/pick) to effortlessly, and via a graphical user interface, select a variety of devices with chipsets that share fewer vulnerabilities, increasing the likelihood of testing novel implementations rather than re-used ones.

TABLE III
PRESENTED AND ACTUAL NUMBER OF SMARTPHONE MODELS AFFECTED BY CHIPSET CVES DISCOVERED BY ACADEMIA. † IDENTIFIER AS PER ORIGINAL PAPER.

Paper	Identifier	# aff. (paper)	# aff. (actual)
Braktooth [54]	CVE-2021-30348	1	293
Braktooth [54]	CVE-2022-20021	-	609
Instructions Unclear [55]	CVE-2022-26446	2	1492
Instructions Unclear [55]	CVE-2022-32591	1	1397
Owfuzz [56]	CVE-2021-1903	3	572
HW-backed Heist [57]	CVE-2018-11976	1	1481
DoLTEst [5]	CVE-2019-2289	17	1381
Signal Overshadowing [58]	SigOver †	10	330
DIKEUE [59]	E1 †	7	420
DIKEUE [59]	E13 †	3	115

Accurate depiction of the impact of newly discovered vulnerabilities. Researching chipset vulnerabilities is a dynamic field, and understanding their real-world impact is crucial. Typically, researchers gauge this impact by assessing how many devices are affected by a vulnerability they uncover. However, CMs usually only disclose information on affected chipset models, not the specific smartphone models impacted. Consequently, many research papers on novel vulnerabilities tend to underestimate the number of affected smartphone models due to the cascading effects discussed in Section V-D, and only report the devices they manually tested. In contrast, our unified knowledge base allows for automatic cross-referencing of chipset vulnerability information published by CMs with affected smartphone models. This provides a comprehensive number of affected device models. We summarize the difference between several papers' reported and actual number of affected devices in Table III, further highlighting the extent of this under-estimation in current literature.

Identifying avenues for future research. Since our unified knowledge base helps to paint a complete picture of chipset vulnerabilities, this information can therefore further facilitate the identification of outliers that warrant future research. *Underrepresented components:* As described in Section V-C, so far, extremely few vulnerabilities in the Near Field Communication (NFC) stack have been discovered, although all of these were rather severe. Therefore, NFC might call for further scrutiny. *High severity targets:* Similarly, we found that firmware vulnerabilities tend to be more severe than driver vulnerabilities. However, currently, our data set contains roughly twice as many driver than firmware vulnerabilities. We thus identify a need for more approaches on firmware security testing.

D. Threats to Validity

Undiscovered vulnerabilities. As we exclusively analyze vulnerabilities acknowledged by CMs or OEMs, this precludes the estimation of total vulnerabilities in chipsets or phones due to undiscovered vulnerabilities. Potential impact: For this reason, any absolute number associated with vulnerability introduction (RQ1) does not represent the overall number of

existing vulnerabilities. We believe our conclusion is valid nonetheless because the relative numbers in Section V-A and the significant differences in vulnerability origin and persistence are unaffected, as both discovered and undiscovered vulnerabilities come from the same underlying distribution of chipset vulnerabilities.

Zero-day vulnerabilities. Our data set does not contain information on vulnerabilities that were discovered by threat-actors, and that CMs are currently unaware of. Such Zero-day vulnerabilities might be particularly severe vulnerabilities, as threat-actors are typically interested in exploitability. Potential impact: Within RQ2 we only consider vulnerabilities discovered by CMs (internal) and external researchers who report their findings to the CMs. Therefore, our results do not allow claims on the total amount of discovered vulnerabilities. Likewise, if there would be a significant amount of high-severity zero-day vulnerabilities, the absolute numbers of high-severity vulnerabilities within RQ3 and especially Figure 4 might be lower-bounds. Nonetheless the conclusion that firmware vulnerabilities are, on average, more severe than driver vulnerabilities prevails, as the underestimation would equally affect firmware and drivers.

Silent patching. Our knowledge base does not contain information on vulnerabilities that are known to CMs, but have not been publicly disclosed and instead were patched without a public announcement. Given that Qualcomm and Samsung already published hundreds of internally discovered vulnerabilities, some of which being very severe and exploitable, the upside of additionally engaging in silent patching seems to be rather minimal for these two CMs. Potential impact: We assume that silently patched vulnerabilities would be internally discovered, as preventing external researchers from publishing information on their discoveries is not generally possible. This would skew the results presented in Section V-B. Likewise, the results on timeframes for patch development (Section V-C) and update availability (Section V-D) may not apply for silently patched vulnerabilities. Lastly, CMs may be particularly inclined to silently patch high severity vulnerabilities. The absolute numbers presented in Figure 4 likely represent a lower bound, yet this does not alter the conclusion that firmware vulnerabilities are generally more severe than driver vulnerabilities, as both are equally impacted by this limitation.

Data collection and augmentation. Some our our data is obtained from websites, requiring us to automatically extract relevant information. Since this process is based on carefully written, individual parsers and validators for each vantage point, we believe our results to be as reliable as state of the art papers on this topic, that all employ a similar data collection method [7], [18], [19], [21]. Potential impact: As information on vulnerabilities is gathered from several independent vantage points (CM security bulletins, AOSP security bulletins and NVD data), we believe that this information is comprehensive. Information that is collected from a single source, such as chipset release dates and assignments of chipset models to phone models, may occasionally be incorrect. Augmented information may also have inaccuracies. However, given the large quantities of vulnerabilities, chipsets, and phones suggested, the primary outcomes of this study will not be affected by a limited number of data imperfections.

VIII. Conclusion

We create the first comprehensive knowledge base of 3,676 smartphone chipset vulnerabilities, tightly linking chipset models and vulnerabilities to devices and their updates. By empirically analyzing this data set, we discover that chipset vulnerabilities often persist across many chipset generations, such that each chipset vulnerability exposes hundreds or even thousands of different smartphone models. We identified significant shortcomings in current vulnerability management processes, such as inconsistently published information and OEMs frequently failing to inform users about vulnerabilities affecting their devices, leaving them at risk of exploitation. Our findings highlight unique characteristics of chipset vulnerabilities, including extensive code reuse and prolonged patch propagation times, which set them apart from vulnerabilities in other high-risk targets like web browsers and operating systems. To prevent chipset firmware and drivers from remaining a critical weakness in the security architecture of Android smartphones, we propose several measures to enhance their security. We envision that our unified knowledge base will enable future research to tackle these problems by identifying particularly problematic aspects of the vulnerability lifecycle.

Acknowledgment

We would like to thank our anonymous reviewers for their valuable comments and suggestions. We also thank Paul Staat, Alyssa Milburn and Marius Muench for their feedback. This work was supported by the German Federal Office for Information Security (FKZ: Pentest-5GSec - 01MO23025B) and the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2092 CASA - 390781972.

References

[1] U.S. Department of Homeland Security, "CISA Adds Four Known Exploited Vulnerabilities to Catalog," https://www.cisa.gov/news-events/alerts/2023/12/05/cisa-adds-four-known-exploited-vulnerabilities-catalog, 2023, [Online; accessed December 6, 2024].

[2] StatCounter, "Global market share held by mobile operating systems from 2009 to 2023, by quarter." https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/, October 2023, [Online; accessed December 6, 2024].

[3] Security Lab, "Predator Files: Technical deep-dive into Intellexa Alliance's surveillance products," https://securitylab.amnesty.org/latest/2023/10/technical-deep-dive-into-intellexa-alliance-surveillance-products/, Amnesty International, 2023, [Online; accessed December 6, 2024].

[4] G. Hernandez, M. Muench, D. Maier, A. Milburn, S. Park, T. Scharnowski, T. Tucker, P. Traynor, and K. Butler, "FIRMWIRE: Transparent dynamic analysis for cellular baseband firmware," in *Network and Distributed Systems Security Symposium (NDSS) 2022*, 2022.

[5] C. Park, S. Bae, B. Oh, J. Lee, E. Lee, I. Yun, and Y. Kim, "DoLTEst: In-depth downlink negative testing framework for LTE devices," in *31st USENIX Security Symposium (USENIX Security 22)*. Boston, MA: USENIX Association, Aug. 2022, pp. 1325–1342.

[6] R.-P. Weinmann, "Baseband attacks: Remote exploitation of memory corruptions in cellular protocol stacks," in *6th USENIX Workshop on Offensive Technologies (WOOT 12)*. Bellevue, WA: USENIX Association, Aug. 2012.

[7] A. Acar, G. S. Tuncay, E. Luques, H. Oz, A. Aris, and S. Uluagac, "50 shades of support: A device-centric analysis of android security updates," in *Network and Distributed Systems Security Symposium (NDSS) 2024*, 2024.

[8] J. Classen, F. Gringoli, M. Hermann, and M. Hollick, "Attacks on wireless coexistence: Exploiting cross-technology performance features for inter-chip privilege escalation," in *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2022, pp. 1229–1245.

[9] S. Swanson and M. B. Taylor, "Greendroid: Exploring the next evolution in smartphone application processors," *IEEE Communications Magazine*, vol. 49, no. 4, pp. 112–119, 2011.

[10] M. B. Taylor, "Is dark silicon useful? harnessing the four horsemen of the coming dark silicon apocalypse," in *Proceedings of the 49th annual design automation conference*, 2012, pp. 1131–1136.

[11] Ginny, C. Kumar, and K. Naik, "Smartphone processor architecture, operations, and functions: current state-of-the-art and future outlook: energy performance trade-off: Energy–performance trade-off for smartphone processors," *The Journal of Supercomputing*, vol. 77, pp. 1377–1454, 2021.

[12] A. Cabrera, S. Hitefield, J. Kim, S. Lee, N. R. Miniskar, and J. S. Vetter, "Toward performance portable programming for heterogeneous systems on a chip: A case study with qualcomm snapdragon soc," in *2021 IEEE High Performance Extreme Computing Conference (HPEC)*, 2021.

[13] Counterpoint Research, ""Smartphone application processor (AP)/system-on-chip (SoC) vendor shipment share worldwide from 2020 to 2022" Chart." https://www.statista.com/statistics/1226674/smartphone-application-processor-vendor-shipment-share-worldwide/, September 2022, [Online; accessed December 6, 2024].

[14] Zerodium, "Zerodium ," https://zerodium.com/program.html, 2024, [Online; accessed December 6, 2024].

[15] B. Schneier, "Crypto-Gram: Full Disclosure and the Window of Exposure," https://www.schneier.com/crypto-gram/archives/2000/0915.html#1, Counterpane Internet Security, Inc., September 2000, [Online; accessed December 6, 2024].

[16] Mozilla Project, "Mozilla Firefox: Development Process," https://wiki.mozilla.org/Release_Management/Release_Process, 2022, [Online; accessed December 6, 2024].

[17] Android Open Source Project, "Android software management," https://source.android.com/docs/setup/about/codelines, 2024, [Online; accessed December 6, 2024].

[18] S. Farhang, M. B. Kirdan, A. Laszka, and J. Grossklags, "An empirical study of android security bulletins in different vendors," in *Proceedings of The Web Conference 2020*, 2020, pp. 3063–3069.

[19] Q. Hou, W. Diao, Y. Wang, C. Mao, L. Ying, S. Liu, X. Liu, Y. Li, S. Guo, M. Nie *et al.*, "Can we trust the phone vendors? comprehensive security measurements on the android firmware ecosystem," *IEEE Transactions on Software Engineering*, 2023.

[20] K. R. Jones, T.-F. Yen, S. C. Sundaramurthy, and A. G. Bardas, "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*, 2020, pp. 551–567.

[21] M. Linares-Vásquez, G. Bavota, and C. Escobar-Velásquez, "An empirical study on android-related vulnerabilities," in *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, 2017.

[22] Z. Zhang, H. Zhang, Z. Qian, and B. Lau, "An investigation of the android kernel patch ecosystem," in *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, Aug. 2021, pp. 3649–3666.

[23] M. Shahzad, M. Z. Shafiq, and A. X. Liu, "Large scale characterization of software vulnerability life cycles," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 4, pp. 730–744, 2019.

[24] N. Alexopoulos, M. Brack, J. P. Wagner, T. Grube, and M. Mühlhäuser, "How long do vulnerabilities live in the code? a {Large-Scale} empirical measurement study on {FOSS} vulnerability lifetimes," in *31st USENIX Security Symposium (USENIX Security 22)*. Boston, MA: USENIX Association, Aug. 2022, pp. 359–376.

[25] W. A. Arbaugh, W. L. Fithen, and J. McHugh, "Windows of vulnerability: A case study analysis," *Computer*, vol. 33, no. 12, pp. 52–59, 2000.

[26] S. Frei, M. May, U. Fiedler, and B. Plattner, "Large-scale vulnerability analysis," in *Proceedings of the 2006 SIGCOMM Workshop on Large-Scale Attack Defense*, ser. LSAD '06. New York, NY, USA: Association for Computing Machinery, 2006, p. 131–138.

[27] N. Alexopoulos, S. M. Habib, S. Schulz, and M. Mühlhäuser, "The tip of the iceberg: On the merits of finding security bugs," *ACM Transactions on Privacy and Security (TOPS)*, vol. 24, no. 1, pp. 1–33, 2020.

[28] S. Clark, M. Collis, M. Blaze, and J. M. Smith, "Moving targets: Security and rapid-release in firefox," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, 2014, pp. 1256–1266.

[29] S. Clark, S. Frei, M. Blaze, and J. Smith, "Familiarity breeds contempt: The honeymoon effect and the role of legacy code in zero-day vulnerabilities," in *Proceedings of the 26th annual computer security applications conference*, 2010, pp. 251–260.

[30] K. Huang, J. Zhang, W. Tan, and Z. Feng, "Shifting to mobile: Network-based empirical study of mobile vulnerability market," *IEEE Transactions on Services Computing*, vol. 13, no. 1, pp. 144–157, 2020.

[31] A. Sivagnanam, S. Atefi, A. Ayman, J. Grossklags, and A. Laszka, "On the benefits of bug bounty programs: A study of chromium vulnerabilities," in *Workshop on the Economics of Information Security (WEIS)*, 2021.

[32] S. Atefi, A. Sivagnanam, A. Ayman, J. Grossklags, and A. Laszka, "The benefits of vulnerability discovery and bug bounty programs: Case studies of chromium and firefox," in *Proceedings of the ACM Web Conference 2023*, 2023, pp. 2209–2219.

[33] M. Zhao, J. Grossklags, and P. Liu, "An empirical study of web vulnerability discovery ecosystems," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 1105–1117.

[34] A. Anwar, A. Abusnaina, S. Chen, F. Li, and D. Mohaisen, "Cleaning the nvd: Comprehensive quality assessment, improvements, and analyses," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 6, pp. 4255–4269, 2021.

[35] Y. Dong, W. Guo, Y. Chen, X. Xing, Y. Zhang, and G. Wang, "Towards the detection of inconsistencies in public security vulnerability reports," in *28th USENIX Security Symposium (USENIX Security 19)*. Santa Clara, CA: USENIX Association, Aug. 2019, pp. 869–885.

[36] Y. Jiang, M. Jeusfeld, and J. Ding, "Evaluating the data inconsistency of open-source vulnerability repositories," in *Proceedings of the 16th International Conference on Availability, Reliability and Security*, 2021, pp. 1–10.

[37] D. Mu, A. Cuevas, L. Yang, H. Hu, X. Xing, B. Mao, and G. Wang, "Understanding the reproducibility of crowd-reported security vulnerabilities," in *27th USENIX Security Symposium (USENIX Security 18)*. Baltimore, MD: USENIX Association, Aug. 2018, pp. 919–936.

[38] R. Croft, M. A. Babar, and L. Li, "An investigation into inconsistency of software vulnerability severity across data sources," in *2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2022, pp. 338–348.

[39] IDC, ""Smartphone shipments by vendor worldwide from 4th quarter 2009 to 2nd quarter 2023 (in million units)." Chart." https://www.statista.com/statistics/271490/quarterly-global-smartphone-shipments-by-vendor/, July 2023, [Online; accessed December 6, 2024].

[40] F. Richter, "Google Remains a Niche Player in the Smartphone Market ," https://www.statista.com/chart/25463/popularity-of-google-smartphones/, Statista, 2023, [Online; accessed December 6, 2024].

[41] M. Lempers, "Fairphone Impact Report 2022," https://www.fairphone.com/wp-content/uploads/2023/05/Fairphone-Impact-Report-2022.pdf, Fairphone B.V., 2023, [Online; accessed December 6, 2024].

[42] SAMSUNG, "Samsung to Launch Mobile Security Rewards Program, Welcoming Security Research Community ," https://news.samsung.com/global/samsung-to-launch-mobile-security-rewards-program-welcoming-security-research-community, 2016, [Online; accessed December 6, 2024].

[43] Qualcomm Technologies, Inc., "Qualcomm announces launch of bounty program, offering up to $15,000 usd for the discovery of vulnerabilities," https://www.qualcomm.com/news/releases/2016/11/qualcomm-announces-launch-bounty-program-offering-15000-usd-discovery, 2016, [Online; accessed December 6, 2024].

[44] MediaTek Inc., "Internet Archive Wayback Machine, August 2023: MediaTek Report Vulnerability ," https://web.archive.org/web/20230812015002/https://corp.mediatek.com/security-contact, 2023, [Online; accessed December 6, 2024].

[45] Google LLC, "Hardening Firmware Across the Android Ecosystem ," https://security.googleblog.com/2023/02/hardening-firmware-across-android.html, 2023, [Online; accessed December 6, 2024].
[46] M. Clark, "Google says hackers could silently own your phone until samsung fixes its modems," https://www.theverge.com/2023/3/16/23644013/samsung-exynos-modem-security-issue-project-zero, Vox Media, LLC., 2023, [Online; accessed December 6, 2024].
[47] SAMSUNG, "Semiconductor Product Security ," https://semiconductor.samsung.com/support/quality-support/disclosure-policy/, 2016, [Online; accessed December 6, 2024].
[48] J. Edge, "The bogus CVE problem," https://lwn.net/Articles/944209/, LWN.net by Eklektix, Inc., September 2023, [Online; accessed December 6, 2024].
[49] PostgreSQL Global Development Group, "CVE-2020-21469 is not a security vulnerability," https://www.postgresql.org/about/news/cve-2020-21469-is-not-a-security-vulnerability-2701/, August 2023, [Online; accessed December 6, 2024].
[50] D. Stenberg, "NVD makes up vulnerability severity levels," https://daniel.haxx.se/blog/2023/03/06/nvd-makes-up-vulnerability-severity-levels/, March 2023, [Online; accessed December 6, 2024].
[51] K. S. Yim, I. Malchev, A. Hsieh, and D. Burke, "Treble: Fast software updates by creating an equilibrium in an active software ecosystem of globally distributed stakeholders," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 18, no. 5s, pp. 1–23, 2019.
[52] Android Open Source Project, "Android Security Bulletins," https://source.android.com/docs/security/bulletin/asb-overview, 2024, [Online; accessed December 6, 2024].
[53] Android Platform Hardening Team, "System hardening in Android 11," https://android-developers.googleblog.com/2020/06/system-hardening-in-android-11.html, June 2020, [Online; accessed December 6, 2024].
[54] M. E. Garbelini, V. Bedi, S. Chattopadhyay, S. Sun, and E. Kurniawan, "BrakTooth: Causing havoc on bluetooth link manager via directed fuzzing," in *31st USENIX Security Symposium (USENIX Security 22)*. Boston, MA: USENIX Association, Aug. 2022, pp. 1025–1042.
[55] D. Klischies, M. Schloegel, T. Scharnowski, M. Bogodukhov, D. Rupprecht, and V. Moonsamy, "Instructions unclear: Undefined behaviour in cellular network specifications," in *32nd USENIX Security Symposium (USENIX Security 23)*. Anaheim, CA: USENIX Association, Aug. 2023, pp. 3475–3492.
[56] H. Cao, L. Huang, S. Hu, S. Shi, and Y. Liu, "Owfuzz: Discovering wi-fi flaws in modern devices through over-the-air fuzzing," in *Proceedings of the 16th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2023, pp. 263–273.
[57] K. Ryan, "Hardware-backed heist: Extracting ecdsa keys from qualcomm's trustzone," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 181–194.
[58] H. Yang, S. Bae, M. Son, H. Kim, S. M. Kim, and Y. Kim, "Hiding in plain signal: Physical signal overshadowing attack on LTE," in *28th USENIX Security Symposium (USENIX Security 19)*. Santa Clara, CA: USENIX Association, Aug. 2019, pp. 55–72.
[59] S. R. Hussain, I. Karim, A. A. Ishtiaq, O. Chowdhury, and E. Bertino, "Noncompliance as deviant behavior: An automated black-box noncompliance checker for 4g lte cellular devices," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 1082–1099.

APPENDIX

In this section we explain further technical details on the implementation that automatically performs the data collection process described in Section IV.

A. Data collection

As shown in Table IV, we incorporate information from CMs, OEMs, the AOSP, NIST, as well as websites listing device and chipset information. The entire data collection is implemented in NodeJS[3], using the NestJS[4] framework, using a MariaDB[5] instance to store collected information using a relational data model. As the websites differ in data formats, we employ different techniques to obtain structured information. In the cases of Tecno and the NIST NVD, vantage points are available to us as easily processable JSON documents, which we download using NodeJS built-in functionality. The other vantage points require processing of HTML-based websites. All vantage points that are HTML-based websites use a reoccurring HTML structure within the same vantage point. We download their HTML files, again using built-in functionality, and then use the Cheerio library[6] to extract specific fields from the HTML structure via their CSS selector.

For instance, all Qualcomm security bulletins use the same, tabular format. This enables us to obtain their information by traversing the HTML structure using CSS selectors, taking account labels - such as HTML column headers, to detect structural changes. Figure 9 depicts a part of a Qualcomm security bulletin. To obtain the text associated with the Technology Area field, we use the `td:contains(Technology Area) + td` CSS selector, which selects the table cell adjacent to the cell that contains the text "Technology Area".

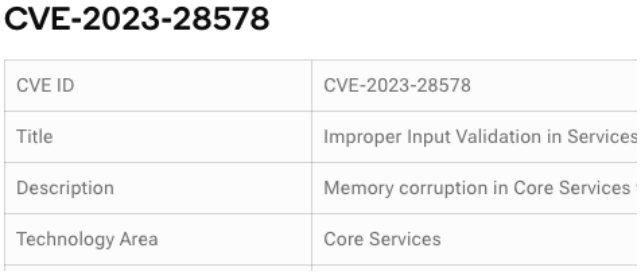

Fig. 9. Excerpt from a Qualcomm security bulletin. Source: https://docs.qualcomm.com/product/publicresources/securitybulletin/march-2024-bulletin.html.

For each vantage point we developed a separate parser, taking into account the vantage point's specific structure. This is in-line with the data collection methods employed in prior work [7], [18], [19], [21]. Any data point obtained is automatically checked for plausibility, allowing us to catch changes should the HTML structure change in the future. For instance, CVE numbers should always follow the `CVE-<YEAR>-<NUM>` pattern, where the first set of digits `<YEAR>` must be a valid year in the past.

In cases where the security bulletins are not available as static HTML, and instead are rendered via Javascript, we use Puppeteer[7]. This allows us to automate a browser which is running in the background to open the relevant website, execute any Javascript to generate HTML, and then process the HTML as described before. This is currently only needed to process security bulletins published by Unisoc. In contrast to prior work, all collection procedures, are performed periodically, and followed by a transformation into the schema illustrated in

[3]https://nodejs.org/
[4]https://nestjs.com/
[5]https://mariadb.org/
[6]https://cheerio.js.org/
[7]https://pptr.dev/

Figure 10. The unified knowledge base is therefore constantly evolving as vantage points are updated by NIST, CMs, OEMs and the AOSP, rather than being a snapshot of the situation at the time the paper was written.

B. Data augmentation

After the individual data points have been collected, some information is missing or inconsistently represented across CMs. Below, we provide a list of all data fields that we augment to mitigate this issue, and details on the augmentation procedures.

Chipset components. When comparing vulnerabilities across CMs, we have to normalize the name of the component affected by a chipset, as different CMs use different names for components implementing the same functionality. For instance, while MediaTek licenses the ARM Mali GPU architecture, Qualcomm uses their in-house Adreno GPU design. As both fulfil the same functional objective, we want them to be assigned a common component name. To fulfil this requirement, our list of components is inherently based on the differentiation used by CMs. We build a list of common component names, and CM specific key-terms identifying components, by iteratively scanning CM security bulletins of all CMs for the field containing the CM specific component name, e.g. "Technology Area" in Qualcomm Security Bulletins. We then perform the following steps:

1) Check if the CM specific component name is already in our set of key-terms. If so, stop the process for the current bulletin and continue with the next bulletin.
2) If not, look up the CM specific component name in technical documentation, source code fragments and marketing material to identify its functionality.
3) Identify if there is already a common component name for the identified functionality, if not: add it to the list.
4) Add the CM specific component name as a key term for the common component name either identified or added in step 3.
5) Continue this process with the next bulletin.

Using this approach, we establish a nomenclature of Android smartphone chipset components that differentiates between the following components:

Bluetooth. Bluetooth connections and packets, including Bluetooth low energy.

WiFi. IEEE-802.11 (WiFi) connectivity

Cellular. Cellular connectivity subsystem, including signal processing, baseband processor and Radio Interface Layer (RIL), used to establish connections to e. g., 4G and 5G networks.

GPU. Subsystem responsible for graphics processing and rendering, including Digital Rights Management (DRM) for visual content (i. e., WideVine).

Vision. Camera, facial recognition and similar visual perception systems.

NFC. Near-Field-Communication, for instance used for digital payment and smartcard applications.

Boot. Bootloader and other early-stage firmware used to start the operating system.

Position. Subsystem to determine device position, e. g., via the Global Positioning System.

Audio. Input and output audio processing pipelines.

Virtualization. Hardware support for virtual machine monitors and hypervisors.

Machine Learning. Support for the implementation of inference systems,e. g., acceleration of matrix operations for neural network computations via dedicated hardware.

Trust. Isolation functionalities for programs with elavated security requirements (i. e., via ARM TrustZone), signature as well as efuse mechanisms, used for instance to authenticate device integrity.

Power. Power management facilities, for instance used to reduce power consumption while the device is not actively used.

IPC. Inter-processor communication, used by primary components to communicate with each other, e.g. via mailbox mechanisms or queues transmitted via serial busses.

Memory. Memory management and protection mechanisms.

Debug. Logging and debugging interfaces.

C. Data integration

After the data has been collected and augmented, information collected from different vantage points is still separated, and not adhering to a common schema. During the data integration step, we integrate individually collected data points into the shared relational schema shown in Figure 10. To do so, we first map the individual descriptors used by the vantage points to our own, canonical descriptors. For instance, Qualcomm's `CVE ID` in Figure 9 becomes `Identifier` in the `Vulnerability` relation. The mapping between canonical and vantage point-specific attribute names is based on a set of static rules. We then insert the collected information into the relational database. For potentially conflicting information, such as vulnerability severity, we store the values separately for each vantage point. This allows us to analyze conflicting information in Section V-C. Lastly, we establish foreign key relationships between the different entities, which allows us to analyze the vulnerability lifecycle based on data from different vantage points in a unified way. For instance, we establish a many-to-many relationship between vulnerabilities and the chipset models they affect. To do so, we use the list of affected chipset model names published in the CM security bulletin and NVD entry of each vulnerability to identify matching chipset models in our database. We then establish the foreign key relationship between the vulnerability, and all identified chipset models. Since we also establish a one-to-to-many relationship between every chipset model and the devices it is used in, this allows us to determine which vulnerability affects which devices by traversing the foreign key relationships. All foreign keys we establish are depicted as arrows in Figure 10.

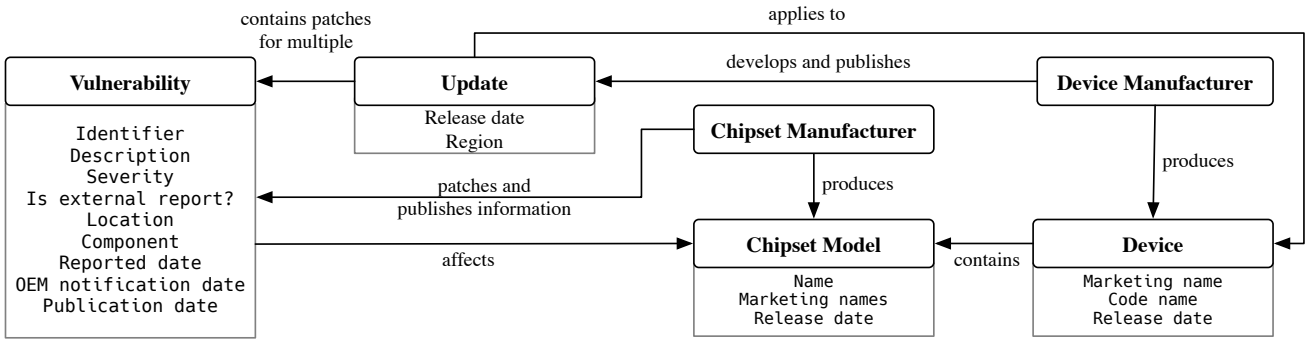

Fig. 10. Relational schema employed by our knowledge base. Boxes correspond to relations, and contain the attributes in that relation. Arrows correspond to foreign key references.

TABLE IV
LIST OF VANTAGE POINTS

Name	Vantage Points	Format	Relevant for Phase			
			Intro.	Discovery	Patching	Updating
Qualcomm Security Bulletin[1]	CM	HTML	✓	✓	✓	✗
Mediatek Security Bulletin[2]	CM	HTML	✓	✓	✓	✗
Samsung Security Bulletin[3]	CM, OEM	HTML	✓	✓	✓	✓
Samsung Semicon. Security Bulletin[4]	CM	HTML	✓	✓	✓	✗
Unisoc Security Bulletin[5]	CM	Javascript	✓	✓	✓	✗
Samsung Updates[6]	OEM	HTML	✗	✗	✗	✓
Xiaomi Updates[7]	OEM	HTML	✗	✗	✗	✓
Tecno Updates[8]	OEM	JSON	✗	✗	✗	✓
Tecno Changesets[9]	OEM	JSON	✗	✗	✗	✓
Android Security Bulletin[10]	AOSP	HTML	✗	✗	✓	✓
NIST NVD Database[11]	NIST, CVE	JSON	✓	✓	✓	✗
GSMArena[12]	Device information	HTML	✗	✗	✗	✓
Wikipedia[13]	Chipset release dates	HTML	✓	✗	✗	✗

Exemplary URLs:
[1] https://docs.qualcomm.com/product/publicresources/securitybulletin/march-2024-bulletin.html
[2] https://corp.mediatek.com/product-security-bulletin/March-2024
[3] https://security.samsungmobile.com/securityUpdate.smsb
[4] https://semiconductor.samsung.com/support/quality-support/product-security-updates/
[5] https://www.unisoc.com/en_us/secy/announcementDetail/1754320321801945089
[6] https://doc.samsungmobile.com/SM-A415F/PHN/doc.html
[7] https://xiaomifirmwareupdater.com/archive/miui/agate/
[8] https://security.tecno.com/slm/deviceScope?lang=en-US
[9] https://security.tecno.com/slm/patchInfo?lang=en_US&year=2022&quarter=01
[10] https://source.android.com/docs/security/bulletin/2023-09-01
[11] https://services.nvd.nist.gov/rest/json/cves/2.0?cveId=CVE-2022-33251
[12] https://www.gsmarena.com/samsung_galaxy_s20_fe_5g-10377.php
[13] https://en.wikipedia.org/w/api.php?action=parse&page=List_of_Qualcomm_Snapdragon_systems_on_chips&prop=text&section=1&format=json