Deploying Android Security Updates: an Extensive Study Involving Manufacturers, Carriers, and End Users

Kailani R. Jones
kailanij@ku.edu
University of Kansas

Ting-Fang Yen
tingfang.yen@datavisor.com
DataVisor, Inc.

Sathya Chandran Sundaramurthy
sathya.chandran3@gmail.com
DataVisor, Inc.

Alexandru G. Bardas
alexbardas@ku.edu
University of Kansas

ABSTRACT

Android's fragmented ecosystem makes the delivery of security updates and OS upgrades cumbersome and complex. While Google initiated various projects such as Android One, Project Treble, and Project Mainline to address this problem, and other involved entities (e.g., chipset vendors, manufacturers, carriers) continuously strive to improve their processes, it is still unclear how effective these efforts are on the delivery of updates to supported end-user devices. In this paper, we perform an extensive quantitative study (Aug. 2015 to Dec. 2019) to measure the Android security updates and OS upgrades rollout process. Our study leverages multiple data sources: the Android Open Source Project (AOSP), device manufacturers, and the top four U.S. carriers (AT&T, Verizon, T-Mobile, and Sprint). Furthermore, we analyze an end-user dataset captured in 2019 (152M anonymized HTTP requests associated with 9.1M unique user identifiers) from a U.S.-based social network. Our findings include unique measurements that, due to the fragmented and inconsistent ecosystem, were previously challenging to perform. For example, manufacturers and carriers introduce a median latency of 24 days before rolling out security updates, with an additional median delay of 11 days before end devices update. We show that these values alter per carrier-manufacturer relationship, yet do not alter greatly based on a model's age. Our results also delve into the effectiveness of current Android projects. For instance, security updates for Treble devices are available on average 7 days faster than for non-Treble devices. While this constitutes an improvement, the security update delay for Treble devices still averages 19 days.

CCS CONCEPTS

• **Security and privacy → Mobile platform security**.

KEYWORDS

Android security updates; mobile carriers; mobile manufacturers; end-users; Project Treble; Android One

CCS '20, November 9–13, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-7089-9/20/11…$15.00
https://doi.org/10.1145/3372297.3423346

ACM Reference Format:
Kailani R. Jones, Ting-Fang Yen, Sathya Chandran Sundaramurthy, and Alexandru G. Bardas. 2020. 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 (CCS '20), November 9–13, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3372297.3423346

1 INTRODUCTION

Android's highly fragmented and inconsistent ecosystem makes the updating and operating system (OS) upgrading processes cumbersome and complex [16, 17, 21, 47]. With a Linux kernel, multiple parties partake within this scheme: operating system developers (e.g., Linux, Android Open Source Project), system-on-a-chip manufacturers (e.g., Qualcomm, NVIDIA), device model manufacturers (e.g., Samsung, LG), mobile carriers (e.g., AT&T, Verizon), carrier partners, third-party testing labs, and end users [5, 8, 12, 36, 46].

Over the years, Google initiated multiple projects that seek to increase the number of Android devices receiving security updates. For instance, in their Security & Privacy report released in 2019 [16], Android reported an 84% increase of devices receiving a security update in contrast to the last quarter in the previous year. However, this does not imply that these devices are receiving updates covering the latest Android Security Bulletins – they could actually be behind by months [5, 37]. As Android focuses on improving the availability of security patches, it is also important to assess the impact of other involved entities in their delivery and rollout to supported devices.

Providing a comprehensive study on Android security updates is a challenging task due to the fragmentation and inconsistencies across the involved parties [21, 47]. Various past efforts focused on different aspects of the Android update process. For instance, some works examined the patch latency in the Android Open Source Project (AOSP) and how it is affected by the vulnerabilities' content classifications [10, 24, 46]. Other efforts analyzed the manufacturers' involvement in this process [5, 11, 36, 46] or multifarious elements on the end-user side [2, 7, 16, 46]. In general, these efforts focused on important but limited facets of security updates. Recently, the U.S. Federal Trade Commission (FTC) released a report on the issues in mobile security updates [5]. While their goal was to provide a comprehensive perspective, this report covered a limited dataset (i.e., only eight manufacturers). Despite the important insights from these previous efforts, we did not encounter a study of the Android security update rollout process that takes into account all of the players including AOSP, manufacturers, carriers, and end users.

In this work, our goal is just that. Specifically, once a security update is available from AOSP, we examine its journey from the manufacturers and carriers to the end user devices. At each step, we measure how the frequency and latency of security updates (arguably, among the most important metrics of the update process) are affected by these players and other factors influencing this process. Moreover, we also track a recent OS upgrade. Similar to security updates, an OS upgrade also increases a device's security posture [39]. The findings from our study may benefit security-conscious consumers in making informed decisions regarding their mobile devices, and also provide policy makers and software vendors with a deeper understanding of the current Android security landscape. More generally, our study sheds light on the actual rollout process of Android security updates.

Our study is made possible by leveraging multiple data sources. We gathered information from the monthly Android Security Bulletins [38], as well as 1,953 public security update announcements available from the top four U.S. carriers (AT&T, Verizon, T-Mobile, and Sprint[1]) covering 274 unique device models across 25 manufacturers for the same time period. Both of these datasets span from August 2015 (when the Android Security Bulletins started) to December 2019, allowing us to analyze longitudinal update behavior over a period of four years. We complemented these datasets with 684 security update announcements specific to locked and unlocked models from Samsung. Additionally, we analyzed 152,156,934 HTTP requests associated with 9,163,277 unique user identifiers and 4,800,228 unique user-agent strings from a U.S.-based social network, which allows us to empirically observe update behavior by end users and correlate with the carrier/manufacturer announcements. The collection, storage, and processing of all datasets follow strict ethical and privacy considerations as detailed in Section 4.5.

To the best of our knowledge, given the fragmented nature of the Android landscape, this is the first study to adopt a comprehensive approach that results in unique quantitative insights into the Android security update ecosystem. Our findings confirm assessments, such that manufacturers and carriers tend to update older phones with lesser frequency, and complement these assessments with novel insights. For example, the latency of security update rollouts introduced by the device manufacturers and mobile carriers stays relatively unchanged over the past four years – at an additional 24 days (longer than the two weeks it takes to publish the majority of exploits [19]). These latency results vary significantly per manufacturer and carrier; as their relationship is vital. Moreover, when analyzing the carrier's role within this relationship (e.g., locked vs. unlocked devices), we notice that locked models may receive security updates faster and more frequently than their unlocked (and sometimes even "unbranded" – never used a carrier) counterparts; thus, emphasizing again, the important role carriers play in the rollout of security updates. Contrary to our expectations, the CVE severity, type of vulnerabilities, or the number of CVEs included in a security update do not generally impact the rollout latency.

By analyzing the end-user dataset, we empirically evaluate the delay for when end devices applied the update. Our results show that it takes a median value of 11 days from the carrier's security update announcement to when the update was observed on the device. If we consider that users in the end-user dataset login (on average) every 6.12 days and the effect of updates being rolled out in batches or stages, the user-incurred delay can be much lower. Examining the effects of Android's initiatives to improve the updating ecosystem, we find that carriers and manufacturers roll out security updates an average of 7 days faster for Treble [48] models compared to non-Treble models. We also observe Android One [1] and Treble devices receiving a higher number of security updates as well as being more up-to-date. However, these initiatives do not necessarily demonstrate faster OS upgrades or solve the delayed security updates as the manufacturers and carriers still influence the release process. Project Mainline [4], announced in May 2019, is in its early stages of adoption. Since it does not support OS versions prior to Android 10, its adoption rate is also directly impacted by manufacturers and carriers OS upgrade decisions.

Our main contributions can be summarized as follows:

- We perform an extensive quantitative study, at scale, on the impact manufacturers, carriers, and end-users have on the rollout of Android security updates and OS upgrades.
- We aggregate and correlate data from different data sources resulting in novel measurements on carrier and manufacturer-related factors and on user-incurred delays.
- We evaluate the current effectiveness of Android initiatives such as Android One and Project Treble, and provide a very early assessment on the more recent Project Mainline.

The rest of the paper is organized as follows. We first provide background on Android security updates and OS upgrades, and the datasets analyzed for this study. We then describe the findings from our analysis and discuss implications for the Android ecosystem.

2 BACKGROUND

Android security updates and OS upgrades are impacted by various entities [5]. This section overviews this fragmented structure and the interactions between the involved entities. These continuous interactions lead to complex dependency relationships and workflows with manufacturers and carriers serving as prime examples.

Security Updates and OS Upgrades Flow: In this paper, an update refers to a security update (applying an Android Security Bulletin) while an upgrade indicates an OS upgrade (increasing the OS version). As pictured in Figure 1, updating and upgrading an Android device follows a chain of command involving multiple entities. Security updates start with Android patching vulnerabilities within the AOSP followed by chipset vendors as needed. Next, manufacturers, based on whether a model has a customized OS version or not, integrate and test the changes as well as the alterations requested by mobile carriers. Finally, manufacturers, with (usually) approval from the carrier, release the update to the end-users [5, 36].

OS upgrades follow a similar process. First, Google releases the Platform Developer's Kit (PDK) to manufacturers and chipset vendors. The PDK contains both the AOSP and close-sourced components [3]. Next, device manufacturers continue development by ensuring compatibility with the chipset, meeting WiFi and Bluetooth standards, adding basic device components (phone calling, messaging, etc.), designing their own manufacturer customization

[1]The data was gathered when T-Mobile and Sprint were still separate entities/carriers

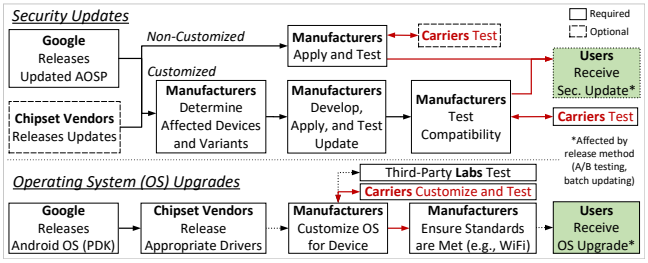

Figure 1: Android security update and OS upgrade chains – Updates follow a chain of command with multiple involved entities. The amount of work on the manufacturers' side depends on whether a model runs a customized (manufacturer-specific) or a non-customized ("stock") OS version, and on the interactions with the carrier (red arrows).

or branding (user-interface, graphics, and other features), and testing all features. During this process manufacturers work with the mobile operators/carriers and third-party testing labs to test or add additional features per request [8, 12, 44].

In this process, the manufacturers and carriers play a vital role. They are not only responsible for integrating and testing security updates and upgrades (often for tens of different models at a time) but also for their delivery to the end devices. In this paper, we explicitly attempt to measure the overhead incurred by these entities during the security update and OS upgrade rollout processes.

Challenges with Fragmentation: The interdependence between manufacturers and carriers (with the involvement of chipset vendors and testing labs) manifests itself in a plethora of back-and-forth interactions. For instance, one manufacturer supporting multiple device models may need to perform individual customizations with various mobile carriers. This can result in as many as 1,500 variations of the same update or upgrade for the manufacturer [2].

The update process is further complicated by the presence of locked and unlocked devices. Locked models are tied to a specific carrier (i.e., carrier-supported) whereas unlocked models are not. Thus, a locked device cannot use other carrier networks, unless the "owning" carrier unlocks that device. By using Samsung (which has the highest market share among Android device manufacturers [3]) as the base case, we observe all models (both locked and unlocked) receive updates from the manufacturer. However, if these devices are connected (e.g., via SIM cards) to a carrier network, then the carrier determines when updates are released [9, 15].

Unlike the structured format present in Android Security Bulletins, each manufacturer and carrier posts announcements for software updates rolled out to supported models in varying formats, if at all. Generally, these update announcements are inconsistent from a content perspective and provided (at times) in changing formats [5]. The large number of carriers and Android device manufacturers, accompanied by a lack of standard reporting mechanisms, make it challenging to perform independent, systematic measurements on the delivery process of security updates.

[2]Variations of the same software: https://www.wired.com/2017/03/good-news-androids-huge-security-problem-getting-less-huge/
[3]Market share as of May 1, 2020: https://www.appbrain.com/stats/top-manufacturers

3 RELATED WORK

Before updates are released to manufacturers, Google must patch AOSP. Farhang et al. [10] measured patch latency from Qualcomm and Linux repositories to code alterations in AOSP and also studied the lifetime of a vulnerability from discovery time to AOSP patch. Linares-Vasquez et al. [24] performed an in-depth analysis of Android vulnerabilities. They studied which system components are commonly affected and also when the vulnerabilities were introduced to when its code was fixed in AOSP. These research efforts showed that severity does not affect the patch latency (e.g., the lifetimes of critical CVEs are similar to that of moderate CVEs).

Another line of work studies Android devices' patch behavior using detailed data collected directly on participating devices. Thomas et al. [46] inspected the Android source code tree to identify when fixes in upstream open-source projects are included in an Android update, and correlate that with device information collected from their Device Analyzer app (on approx. 24,000 user devices) to calculate a metric for the security posture of phone models.

Researchers from SRLabs [22, 36] analyzed the presence of security patches on end devices through a mobile app, SnoopSnitch [4]. Their approach searches for pre-compiled signatures of security patches within binary firmware files, such that source code access is not required. They showed that Android manufacturers differ widely in patch completeness. Samsung and Huawei appear to make patches available faster over time, but it is unclear how the delay is calculated in this work. Duo Labs measured the percentage of devices running the current Android Security Bulletin through the DUO app [2]. They appear to study the adoption of only one bulletin, while our analysis includes 53 bulletins.

More similar to this work are studies that consider the role of manufacturers in the security update rollout process. The FTC released a report in 2018 on mobile manufacturers' Android security update support [5]. They measured the lifetime support of models, the frequency of updates, and the time from when vulnerabilities were reported to when they were patched by the manufacturer. This report only covered models released in 2013 to 2014. Farhang et al. [11] compared the CVEs addressed in each Android Security Bulletin with those included in vendor-specific security bulletins released by manufacturers to calculate the CVE patch delay (i.e., the number of security bulletins (months) until the CVE is included).

Other efforts, not specific to Android, study open-source and third-party patching [6, 23, 29, 45] as well as user studies on users' updating behavior [13, 20, 25–27, 30, 35].

Compared to previous studies, our work takes a more holistic view on the Android security update rollout process. We consider the role of AOSP (the Android Security Bulletins), manufacturers, carriers, and end users/devices in this process. Analyzing four years of security update announcements from top U.S. carriers, Android manufacturers, and a corpus of HTTP requests from Android devices observed by a social network, our study sheds light on how the frequency and latency of updates are affected by these players and other influential factors. Our research effort extends and complements existing works by confirming some of their findings, contradicting others, and providing novel measurements.

[4]SnoopSnitch app for Android: https://play.google.com/store/apps/details?id=de.srlabs.snoopsnitch&hl=en_US

Carrier	AT&T	Sprint	T-Mobile	Verizon
Unique manufacturers	13	13	13	17
Unique models	74	93	111	149
Total number of security update announcements	394	364	658	537
Unique Android Security Bulletins addressed	49	49	53	45
Earliest security update announcement	2015-10-19	2015-10-31	2015-08-05	2015-10-08
Latest security update announcement	2019-12-17	2019-12-26	2019-12-15	2019-12-11

Table 1: Carrier dataset – Carrier security update announcements made between August 2015 and December 2019.

	Locked		Unlocked		
Carrier	Sprint	T-Mobile	Sprint	T-Mobile	Unbranded
Unique models	13	13	13	13	13
Total number of update announcements	136	127	150	120	151
Unique Android Security Bulletins	18	18	18	18	18
Earliest security update announcement	2018-08-06	2018-08-13	2018-07-25	2018-07-25	2018-07-25
Latest security update announcement	2020-01-02	2019-12-30	2020-01-14	2020-01-31	2020-01-14

Table 2: Manufacturer security updates from Samsung – Overview of the collected update announcements rolled out July 2017 to December 2019 for 15 Samsung Galaxy models.

4 DATA

We collected data from multiple data sources to study the Android security updates and OS upgrades rollout process. In this section, we describe the main datasets and review the ethical and privacy considerations. To collect information listed on publicly available websites, including Android Security Bulletins and the security update announcements, we leverage tools such as PyQt5 [5], BeautifulSoup [6], and Selenium [7] for the retrieval and parsing of the webpages. For the end-user data, the collection process was closed-source and the sensitive data fields were anonymized.

4.1 Android Security Bulletins

Since August 2015, AOSP posts Android Security Bulletins [38] on a monthly basis. Each bulletin contains information about the security update released in that month, including the patched CVEs, the type and severity of the CVEs, the affected Android OS versions and components, and the patch level(s) which is in the form of a date. A device with a specific patch level implies that it is up-to-date with all security updates released up to that level.

In studying the security updates rollout process, we use the patch level announcement date in the Android Security Bulletins as the baseline for calculating the *additional* latency imposed by the device manufacturers and mobile carriers. We also examine other factors such as the type and severity of CVEs, affected AOSP components, and number of skipped bulletins.

We collected information from 53 Android Security Bulletins spanning August 2015 to December 2019. There can be a variable number of patch levels in each monthly security bulletin (e.g., "2019-10-01", "2019-10-05"). For consistency purposes, we focus on the first patch level in each bulletin, since that one is always available.

4.2 Carrier Dataset

Similar to the Android Security Bulletins, mobile carriers also announce security updates they release to supported mobile devices, i.e., phones and tablets with cellular service. We collected security update announcements made publicly available by the top four U.S. carriers, i.e., AT&T, Verizon, T-Mobile, and Sprint, from August 2015 to December 2019. Due to inconsistent formatting of the announcements, we developed custom parsing logic for each carrier website (further detailed in Appendix A.1), specifically:

- AT&T does not announce security updates on a centralized page, but instead on separate pages for each supported device model [8]. We manually collected all URLs for each device model by traversing from the support page to each indicated model.
- Similar to AT&T, T-Mobile also lists the security update announcements separately for each device model [9]. We programmatically collected each device model URL.
- Verizon's website only displays the three most recent software updates per model, preventing users from gathering longitudinal data over time. We collected Verizon's security update announcements three times in 2019 (in May, October, and December). [10]
- Sprint posts security update announcements in a blog format. We searched Sprint's entire Android Community Boards. [11]

Each of the security update announcements made by the carriers are specific to a device model. Thus, for each announcement, we collected the device model, the corresponding Android Security Bulletin (if available), and the date of the announcement (which we use to infer the corresponding Android Security Bulletin in cases where this information is not explicit). For example, if an announcement with an unspecified Android Security Bulletin was posted on January 15, 2019, we make a conservative assumption that the corresponding Android Security Bulletin is the one from January 2019. This may result in a lower bound on the security update rollout latency, since the latency could be higher if the update was actually for previous Android Security Bulletin(s). If no release date is available in the announcement, we do not include it in our dataset. Out of 3,493 total collected Android carrier updates, 1,540 were not security updates or did not contain enough information (i.e., the addressed Android bulletin, release date). We further detail our data processing efforts in Appendix A.2. In the following sections, we analyze 1,867 security update announcements for 274 unique models across 25 device manufactures as detailed in Table 1.

4.3 End-User Dataset

End users also play a role in adopting security updates. Depending on the system settings, the user may be prompted for approval before the security update is applied on the device [13, 27, 35].

We obtained anonymized HTTP access logs from Android devices from a U.S.-based social network with over 50 million downloads on the Google Play Store. Registered users can create and share posts related to items of interest, chat in group forums, and friend each other. Users access the social network either through its website or through the mobile app. Since our study focuses on mobile systems, we only used the data collected through the latter and analyzed requests from registered users.

[5] PyQt5: https://riverbankcomputing.com/software/pyqt/intro
[6] BeautifulSoup: https://www.crummy.com/software/BeautifulSoup/bs4/doc/
[7] Selenium: https://www.selenium.dev/
[8] AT&T devices: https://www.att.com/support/contact-us/wireless/
[9] T-Mobile devices:https://www.t-mobile.com/support/phones-tablets-devices
[10] Verizon software updates: https://www.verizonwireless.com/support/software-updates/. Verizon only displays the latest three software updates *per model*, hence (while our data collection took place in 2019) the earliest security update observed is much older as shown in Table 1, e.g., models whose last update was in 2015.
[11] Sprint devices: https://community.sprint.com/t5/My-Phone/ct-p/android

The data is anonymized and exposes four fields to the researchers: the request date, a hashed value of the user account identifier, the user-agent string, and the country code based on the IP address. Only unique records are stored. The user-agent strings are generated by the social network's mobile app, and include the app version number, OS version, phone model, build number, and device carrier.

We only consider HTTP POST requests, as GET requests do not contain the user account identifier and are mostly from unregistered "guest" users for which we cannot observe their update behavior over time. To identify Android traffic, we filtered the requests by searching for keywords "Android" and "Build" in the user-agent string. The build number allows us to analyze when devices receive security updates and which updates are applied.

From January 1 to December 31, 2019 [12], a total of 152,156,934 HTTP POST requests were collected from Android devices with build number information as described above, corresponding to 9,163,277 unique user account identifiers and 4,800,228 unique user-agent strings. We further identified 7,247 unique models across 454 device manufacturers by parsing the user-agent strings.

4.4 Manufacturer Dataset

We also collected security update announcements from Samsung, a top Android manufacturer. We could not locate announcements from other manufacturers due to lack of data (e.g., release date missing, updates not posted). Specifically, we wanted to investigate the manufacturer's effect on the rollout process. Intuitively, unlocked models should not experience carrier-related delays, thus we analyze the update process between locked and unlocked models.

Since 2017, Samsung provides security update announcements for individual models on their website [13]. To access these updates, we constructed the URLs (https://doc.samsungmobile.com/model variant/carrier code/doc.html) for 15 models within the flagship Galaxy series (S7 to S10, S8+ to S10+, S7 edge, Note 8 to Note 10, A10e, A20, A50) associated with T-Mobile or Sprint. We do not include other carriers as Samsung doesn't post updates for the corresponding locked models. In certain instances, the model variant (e.g., SM-G960U) can indicate an unlocked model as indicated on Samsung's website [34]. Table 2 lists the statistics of this dataset. More details about the update process for locked, unlocked, and unbranded devices can be found in Appendix A.5.

4.5 Ethical and Privacy Considerations

While collecting public data from the Android Security Bulletins, the major U.S. carriers' and manufacturers' websites, we prioritized protecting (not overloading) these public resources. Specifically, we rate-limited the number of connections, controlled how quickly we connected (every 2-8 seconds), and locally saved the HTML code to minimize access upon testing and in case of page inconsistencies.

With T-Mobile, Sprint, and Verizon posting updates individually or per device model, we collected URLs covering all available models and updates specified in Section 4.2, then connected to those links to download the necessary code. For AT&T, their website leans more on JavaScript, wait conditions, and auto-generated tag labels which affects the traversal of HTML code. To limit our impact, we manually recorded URLs for all the models reporting a security level and saved the security update content locally for each model. Lastly, we never obfuscated our user agents or IP addresses.

The HTTP access logs were collected according to end-user agreements pertaining to the usage of the online social network. No personally-identifiable information was collected. The researchers were granted access to the anonymized data only on corporate-approved devices and networks, leveraging only the four data fields described in Section 4.3. *Our IRB Office determined that the user data is de-identified and therefore not a human subject.*

5 MEASURING CARRIER AND MANUFACTURER EFFECTS

Manufacturers and carriers play a key role in the distribution of mobile OS and software updates as previously shown in Figure 1 and also detailed in several past research efforts [36, 46]. In this section, we study the carriers' and manufacturers' role in the rollout process of security updates to supported devices.

AOSP issues security patches on a monthly basis. This means that a device model, if updated regularly by the carrier-manufacturer, should receive (at least) one update every month. Using the carrier dataset described in Section 4.2, we measure the number of security updates rolled out to each model as well as *when* those rollouts were announced by the carriers and manufacturers. We find that update patterns are not consistent across device models, e.g., some receive updates for every Android Security Bulletin, while others only receive one update per year. By leveraging Spearman correlation [28], we investigate factors affecting those difference.

5.1 Rollout Frequency

Across the carrier dataset, which spans 53 months between 2015 to 2019, the average number of security update announcements issued per device model is 4.54 (±4.67). This is a far cry from the monthly updates issued in the Android Security Bulletins. However, the carrier dataset includes models of varying "ages" with the oldest released in January 2013 (Blackberry Z10) and the newest released in October 2019 (Pixel 4, LG Stylo 5+, LG Prime 2). Naturally, the age of the model affected how many updates it could receive.

For a more fair comparison across models, we define the normalized update frequency as follows. Let a phone model's release date be T_{rel}. The number of security updates the model could potentially receive during a period ending at T_{end} can be calculated as the number of months between the two timestamps, denoted N_{pot}, while we can observe the actual number of updates the model received during this time, N_{act}. The normalized update frequency is then calculated as the ratio between these two values, $\frac{N_{act}}{N_{pot}}$.

In practice, we apply the later of T_{rel} and the start of our data collection in 2015 to compute N_{pot}, and use the time of the last security update rolled out to the device as T_{end}. We rely on GSMArena [14] and PhoneArena [15] for each model's release date [16].

[12] Due to a bug in the data collection process, we were not able to collect data from June 11 to July 23, 2019.

[13] For example, the announcement page for Galaxy S9 (SM-G960U) on T-Mobile is https://doc.samsungmobile.com/SM-G960U/TMB/doc.html

[14] Release dates from GSMArena: https://www.gsmarena.com/

[15] Release dates from PhoneArena: https://www.phonearena.com/

[16] When only the month and year of a model's release date is available, we use the first day of the month and year.

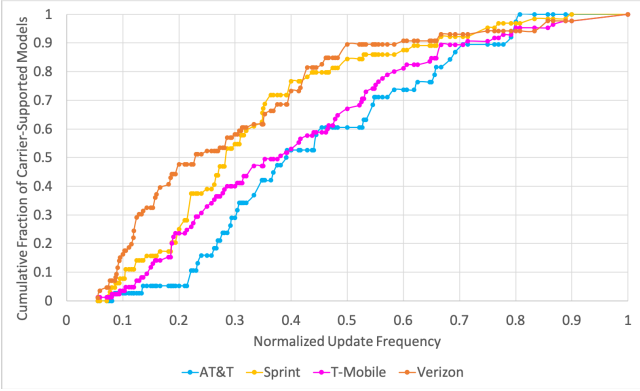

Figure 2: Cumulative distribution of the normalized update frequency per model, for each carrier – The average update frequency is 0.35 across all models, indicating that a carrier-supported device receives slightly more than one-third of updates issued in the monthly Android Security Bulletins.

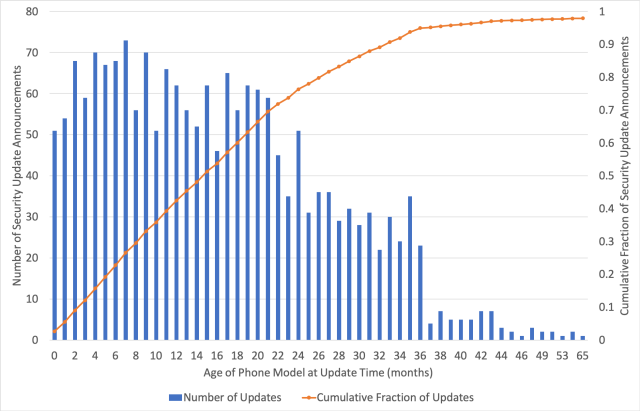

Figure 3: Age distribution of carrier-supported models at the release time of security updates– The age of the model plays a big part in both its update frequency as well as update support duration. The number of security updates rolled out to models older than 36 months drops sharply.

Figure 2 shows the cumulative distribution of the normalized update frequency per model for each carrier. The average update frequency across all models is 0.35, indicating that a carrier-supported device only receives slightly more than one-third of updates issued in the monthly Android Security Bulletins. This observation is consistent across the four major carriers, with T-Mobile having the highest average update frequency at 0.40.

The FTC [5] reported that manufacturers and carriers allocate support towards newer devices, such that newer devices receive security update faster and for a longer period of time. Correlating the age of the model (i.e., as of December 1, 2019) with its normalized update frequency, we observed that the two variables exhibit a moderate negative correlation of -0.3265 with a p-value of 3.3541^{-8}. With high confidence, this result shows carriers do have a preference for newer models. This effect is even more obvious when we examine each carrier separately. For example, T-Mobile shows a strong negative correlation between model age and normalized update frequency of -0.5054 with a p-value of 8.0681^{-7}.

The age of a model appears to play a big part in its update support duration, as well. For each security update announcement issued by a carrier, we examine the age of the model at that time, shown in Figure 3. While the number of updates rolled out to models released within 24 months remain relatively consistent, this value drops sharply at around 36 months. Considering that models older than three years (i.e., released prior to 2017) make up 43.59% of all models in the carrier dataset, the drop is disproportionately large and confirms empirical observations that devices are typically supported by the carriers for 2-3 years.

In addition to the age of the models, we observe that relationships between certain carriers and manufacturers also affect the update process – sometimes in a positive way. In our data, LG models have a median update frequency of 0.42 when associated with T-Mobile, much higher than with other carriers (which range from 0.12 to 0.33). Interestingly, the update frequency for LG models seem to be capped at 0.75, while this does not appear to be the case for other brands (see Figures 19 and 20 in Appendix A.4). This suggests that manufacturers are also responsible for introducing overhead in the update rollout process.

Previous work also reported on interdependent relationships between carrier and manufacturers [5, 36]. Specifically, that carrier involvement in the security update rollout process can influence manufacturers to speed up patching for *popular* models or maintain a routine update schedule. The influence can come from other sources as well, as shown in the initiatives pushed forward by Google encouraging manufacturers to support security updates for designated models and versions. We discuss Google-led initiatives in the update rollout process in Section 7.

5.2 Rollout Latency

An equally important metric to the security update frequency is *when* updates are rolled out. We define *rollout (patching) latency* as the number of days after the publication of the Android Security Bulletin (usually the 1st of the month) until the update is rolled out by the carrier/manufacturer. This quantifies the additional delay introduced by these actors before the updates could reach end devices. The higher the rollout latency, the higher the risk as devices are exposed to potential vulnerabilities during the unpatched time.

Figure 4 shows the update latency for each Android Security Bulletin during our study period, August 2015 to December 2019. The boxplots represent the distribution of the latency across security update announcements issued by the carriers for that bulletin (the carriers have separate update announcements for each model, such that the latency can vary by model even for the same bulletin). Across all Android Security Bulletins, a median delay of 24 days is introduced before carrier rollout – longer than than the two weeks it takes to generally publish the majority of exploits [19].

Looking at each carrier separately, we can see in Figure 5 that all of them introduce delays on the order of weeks to months, with large differences across carriers. AT&T appears to have the highest median delay at 29.5 days, compared to 19 days for T-Mobile.

While the age of the model greatly affected its security update frequency, as shown in Section 5.1, it does not appear to be a strong

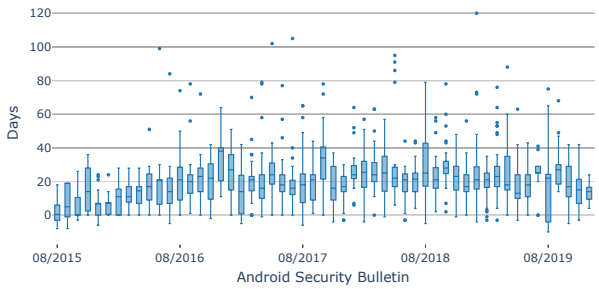

Figure 4: Update latency for each Android Security Bulletin from August 2015 to December 2019 – Each data point represents a security update announcement from the carriers. The rollout latency remains largely unchanged over four years with a median of 24 days.

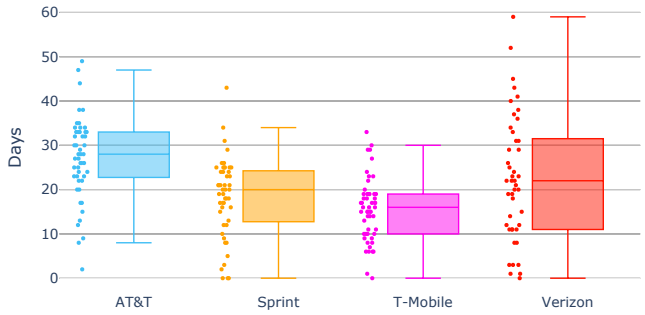

Figure 5: Distribution of update latency per carrier – Each data point corresponds to an Android Security Bulletin, where we calculate the average latency across carrier security update announcements for that bulletin.

factor for update latency. No significant correlation exists between the update latency and the age of the model (a correlation of 0.0502 with p-value 0.0321). Thus even though carriers and manufacturers update models with less frequency over time, they do not discriminate against older devices when an update is available for rollout.

Carrier and manufacturer relationship, on the other hand, remains an important factor. We examine models supported by all four carriers, and compare their update latency across carriers. There are 21 models from Samsung, LG, Motorola, and HTC that match this criteria. If the manufacturer is the main factor, we would expect the latency to be consistent across carriers for the same model. Figure 6 shows that this is not the case. The same model can experience varying amounts of latency depending on the carrier. In particular, the median latency for Motorola models range widely from 10 days (if with T-Mobile) to 37 days (if with AT&T).

Perhaps a more obvious explanation for the update latency is related to the carriers themselves. When rolling out security updates, manufacturers not only customize the updates for the models they support but also commonly require carrier-specific modifications and tests. Using the manufacturer dataset collected from Samsung, as described in Section 4.4, we examine a total of 684 security update announcements for 13 Samsung models across locked, unlocked, and unbranded variants. Locked models receive updates faster than

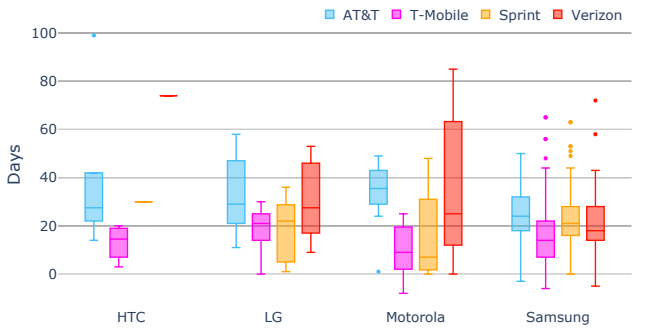

Figure 6: Update latency for models supported by all four carriers – 21 models from 4 manufacturers (Samsung, LG, Motorola, HTC). With different testing mechanisms between manufacturer and carrier [5], a model may experience inconsistent update latency depending on the carrier.

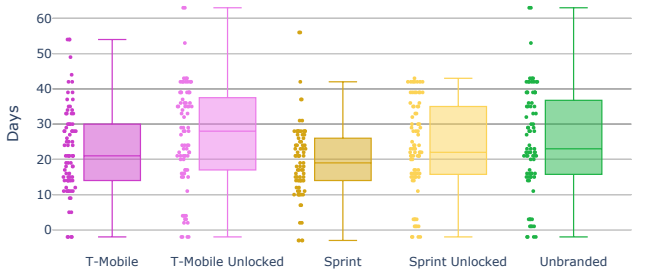

Figure 7: Update latency of Samsung locked, unlocked, and unbranded devices – Locked models, on average, receive updates four days faster than their unlocked counterparts.

their unlocked counterparts, as shown in Figure 7. On average, a locked model receives updates four days earlier.

Our findings on locked models is contradictory to the result reported in FTC's report [5], which found that unlocked devices tend to be patched quicker than locked versions. However, the report also noted that while they observed unlocked models with shorter overall latency, particular locked models may still receive faster updates. We conjecture that since our data from Samsung focuses on their flagship Galaxy models (and hence are popular models), they may be prioritized in the update rollout process.

Lastly, we also examine other information contained in the Android Security Bulletins that could impact update latency. For each security update announcement for a given model, we analyzed the number of "skipped" Android Security Bulletins since its last security update, as well as the number of days since its last security update. Each Android security update patches all CVEs in prior patch levels, hence we expect a larger value for either should correspond to a longer rollout latency. An Android Security Bulletin also includes a list of CVEs patched in that security update, i.e., the CVE ID, severity, vulnerability type, and the Android components affected (the category). We expect updates that address more severe vulnerabilities or that affect core components to be prioritized.

Calculating the Spearman correlation coefficient between the update latency and these variables show that, surprisingly, most of them have no effect (no significant correlation). However, a weak

positive correlation is observed for the number of days since last security update (i.e., the more time passed since the device received its last update, the longer it took the carrier/manufacturer to roll out its current security update), as well as the number of CVEs addressed in the "System" category. The latter has a correlation between 0.45 to 0.61 in three out of four carriers. Detailed statistics for all tested variables can be found in Appendix A.6.

5.3 Summary of Carrier and Manufacturer Measurements

We find that carriers and manufacturers roll out slightly over one-third of all monthly Android Security Bulletins to supported devices, and introduce a median delay of 24 days from the date of the Android Security Bulletin to update rollout. This provides time for adversaries to leverage these known CVEs and develop exploits, e.g., the majority of exploits are published within two weeks from vulnerabilities' public release [19]. The median 24 day latency is for the current Android Security Bulletin (released that month). When some bulletins are skipped for a model, those patches are packaged into subsequent updates [38], such that the delay before a patch reaches the end device could be much longer in practice.

The age of a model is the most significant factor affecting its update frequency. Most models are supported by the carrier for less than 36 months, with the update frequency decreasing as the model ages. By contrast, update latency does not appear to be biased toward model age. This shows that carriers and manufacturers do not discriminate against older devices once an update for a model has been made available and tested (indicating that the development of an update constitute the main overhead in the update rollout process). Prioritizing newer phone models can impact a large population of consumers. Studies have shown that a significant fraction of users keep their mobile devices for three years or more [17].

Partnerships between carriers and manufacturers have a strong impact on both the frequency and latency of updates. Carrier involvement can influence manufacturers to speed up patching for popular models or maintain a routine update schedule, as well as pressure from software vendors such as Google. Partnerships and business decisions can (in this case) trump technical challenges.

Contrary to our expectations, the time since last update, number of skipped Android Security Bulletins, and the addressed CVE severity, type, and category do not exhibit a high correlation with the update latency. Previous work has also reported the lack of relationship between CVE severity and patch speed at the code level [10, 22, 24, 36]. Our study independently validates results from previous work, and also brings to light new findings that were previously challenging to quantify due to fragmentation.

6 MEASURING END-USER UPDATE BEHAVIOR

In this section, we analyze the security update behavior of end devices using the dataset described in Section 4.3. Specifically, we are interested in observing the update frequency and latency on end-user devices, such that we can put them in perspective with

[17] According to a 2019 poll from PhoneArena, one-third of users switch phones every three years or longer https://www.phonearena.com/news/How-often-do-you-upgrade-to-a-new-phone_id119501.

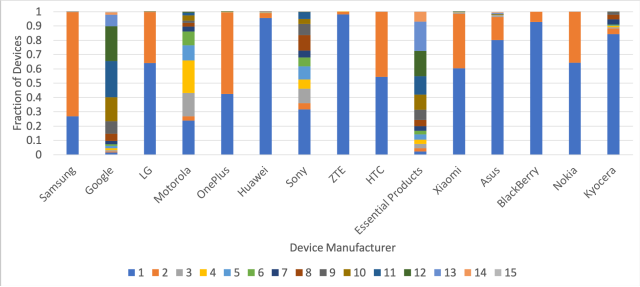

Figure 8: End-user unique build number distribution – Per device, the top 15 manufacturers with the most devices.

our measurements from Section 5 and provide a more complete picture of the Android security update rollout process.

We start by analyzing the number of security updates received by devices in the end-user dataset in 2019. Next, we join this dataset with the carrier dataset to measure (for carrier-supported devices) the update latency introduced by end users and/or the deployment of updates *in addition* to the carrier-manufacturer delay previously calculated. Lastly, we evaluate the rollout of Android 10.

Due to the nature of this data and analysis, we make two major considerations. First, we consider an end "device" as a unique pair of hashed user account identifier and the model variant [18]. Second, we can only observe the user-agent strings when the users access the mobile app, hence we focus our analysis on "active" devices that exhibit continued activities throughout the duration of the data. Our results emerge from 1,273,571 active devices that accessed the app at least once a month for more than 10 months in 2019. [19] Additional statistics on the end-user dataset are available in Appendix A.3.

6.1 Security Updates Frequency and Latency on End-User Devices

We parse the user-agent strings from the mobile app to obtain the phone model variant, carrier, build number, and Android OS version. The model variant is a code name that can be mapped to a specific model, e.g., "starlte" for Samsung Galaxy S9 [20]. The build number specifies the Android code branch from which the build was derived from, the date when the release is branched from or synchronized with the development branch, as well as the version numbers and hotfixes [21]. For example, the build number "QQ1A.191205.011" corresponds to the 2019-12-05 security patch level. Lacking visibility on the devices, we leverage changes to the build numbers as a proxy for observing security updates.

Over the 12 months in 2019, we observe a device associated with 3.44 build numbers on average. Considering that there are (at least) 12 monthly security updates per year, this result is consistent with our measurements from Section 5.1 which showed that the average model receives 35% of all potential Android Security Bulletins. The distribution of build numbers is largely bi-modal, with 76.99% of devices having two or less unique build numbers and 16.55%

[18] Assuming that a user is not using multiple devices of the same model.
[19] We picked 10 months as threshold due to a bug that resulted in a 1.5 month gap in the data. We analyze the potential effects of the gap in Section 8.
[20] We use the models supported by Google Play for this mapping https://support.google.com/googleplay/answer/1727131?hl=en.
[21] AOSP build numbers: https://source.android.com/setup/start/build-numbers

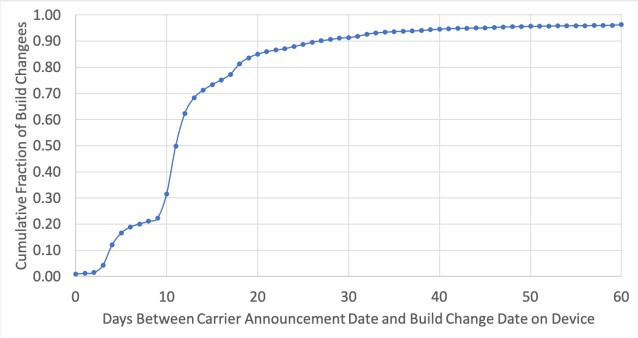

Figure 9: Device update latency – Cumulative distribution of the days between carrier posted date and the build change observed on the device.

having more than nine unique build numbers. A closer look in Figure 8 shows that some manufacturers do not tend to change build numbers, but instead have manufacturer-specific versions that only change upon OS upgrades (e.g., "R16NW" for Samsung). On the other hand, devices from Google, Motorola, Sony and Essential switch build numbers when applying security updates.

For devices with multiple build numbers, we keep track of the earliest date a build number was observed. On average, a device changes build numbers every 47.72 days, with a median of 33 days. This shows updates take place with a monthly cadence corresponding to the monthly release of Android Security Bulletins.

To measure the latency between the carrier update announcement and the update arriving on the device, we match the phone model variant and carrier information in the user-agent strings with the carrier dataset. This latency could be from the end user, who postpones installing available updates, and from updates rolled out in batches (such as an A/B-testing-based approach [40]) by the carrier and manufacturer. The latter is an industry practice to detect issues early in the software release process, where the update is made available to increasingly more devices over time. A device only receives updates when its "batch" is addressed, which may be well after the carrier announcement date.

Out of the active devices, we matched 877,499 (68.90%) with the carrier dataset[22]. This "joined" dataset largely includes Samsung (75.90%), Google (15.94%), LG (5.39%), and Motorola (2.41%) devices. Each time the build number changes on a device, we look up the security update announcements from the corresponding carrier and identify the announcement that most closely occurred prior to the build change. The time distribution between those two dates is shown in Figure 9 with an 11 day median and 15.12 day average.

Lacking information about the batch rollout process, such as the number of batches or release schedule, it is challenging to attribute this latency to the end user or batch rollout. Nonetheless, Figure 9 shows a noticeable trend: most devices are updated shortly after carrier announcement. The curve rises sharply at days 3-5 and 10-12 (likely due to the 6 days average access interval in this dataset), and flattens out soon after. Around when the curve flattens, e.g., day 14, the vast majority (70%) of devices have already updated.

[22]The vast majority of active devices that we were not able to join with the carrier dataset have no carrier information (the value for the carrier field in the user-agent string is "None") or are associated with carriers outside of this study.

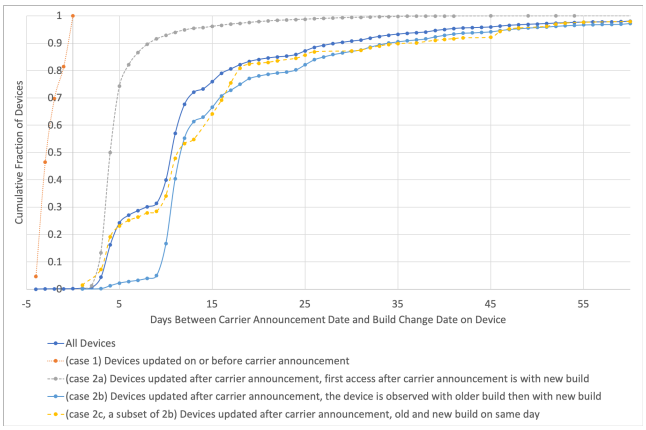

Figure 10: Device access behavior versus latency – Cumulative distribution of the days between carrier announcement and device build change, for Pixel 2 XL devices associated with Verizon updating to build "QP1A.191005.007.A1".

To investigate the effect of batch rollout further, we examine devices that share the exact same phone model, carrier, and receive the same security update. Using the 30,545 Pixel 2 XL devices associated with Verizon that updated to "QP1A.191005.007.A1" as an example, their latency distribution is shown in Figure 10. We can categorize the devices' update behavior into the following cases:

(1) Update prior to carrier release (0.14% of the devices).
(2) Update after carrier release (99.86% of the devices).
  (a) The first time the device accesses the app after the carrier announcement is with the new build number (30.36%).
  (b) After the carrier announcement, the device is observed with an older build number, then subsequently with the new build number (69.50%).

A special scenario in case 2-b (1.58% of devices under case 2-b) is where the device is observed with an older build and the new build number on the same day, i.e., we can determine the exact day the update occurred on that device (illustrated as case 2-c in Figure 10). Update behaviors correspond to different update latency. For example, 90% of devices in case 2-a were updated by day 8. They could be in earlier rollout batches compared to devices in case 2-b, who do not have significant update adoption until day 9. Across the cases, the curves flatten around day 14, similar to Figure 9.

This suggests that batch rollout likely takes place over a period of around two weeks, agreeing with public reports [31]. While measuring user delay (users intentionally delaying updating) is challenging, the fact that the vast majority of devices are updated within this period shows the user delay is significantly smaller than either the delay introduced by the carrier/manufacturer or from the batch rollout process. Whether it is due to carrier and manufacturer delay, their batch rollout process, or even due to the user, a device is susceptible to known vulnerabilities prior to receiving updates.

6.2 OS Upgrades on End-User Devices

The end-user dataset, spanning the duration of 2019, additionally provides us a unique vantage point to observing the rollout of a major Android operating system upgrade: Android 10. Similar to

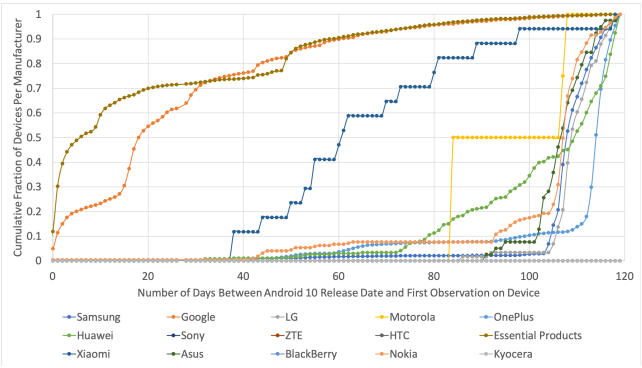

Figure 11: Android 10 adoption per manufacturer – time delta between Android 10 release (September 3, 2019) and the date that Android 10 was first observed on a device.

security updates, Android's OS upgrade also increases a device's security posture [16]. We measure the latency in Android 10 upgrades similar to our approach for measuring security update rollouts in the previous section. Specifically, we quantify upgrade latency as the time between the first observation of Android 10 on a device, and either the official Android OS release date (September 3, 2019) or the date of the corresponding carrier update announcement, for those devices upgraded to Android 10 in 2019.

We first examine the time between the first observation on the device and the official Android OS release date. Among devices that were eventually upgraded to Android 10, 19.93% did so during the first week since its release on September 3, 2019, and 25.09% did so within two weeks. The median time to first observation on end-user devices is 20 days. Compared to security updates, OS upgrades appear to take longer to reach end devices.

OS upgrade behaviors appear to be different depending on the devices' manufacturer, as shown in Figure 11. The upgrades were either rolled out very quickly after the official Android OS release (i.e., for Google and Essential devices), or months later (e.g., Samsung, LG, OnePlus, and Huawei devices). The former is unique in that both manufacturers release software updates directly, bypassing the carrier. Pixel devices receive software updates directly from Google [23], and Essential devices run stock Android with limited modifications and are known for quickly releasing software improvements and updates [24]. By contrast, other manufacturers work closely together with carriers in rolling out updates. According to the update announcements in the carrier dataset, none of the carriers rolled out Android 10 upgrades until mid-December 2019 — over 100 days after the official Android 10 release date. Without carriers and manufacturers batch rollout schedule, using the same rationale from Section 6.1, we observe that after 20 days roughly 70% Essential and 55% Google devices performed OS upgrades (Figure 11) compared to 85% of users adopting security updates (Figure 9). This could mean that the batch rollout window is longer for OS upgrades than for security updates and/or end users tend to delay OS upgrades more than updates.

[23] Update schedule for Pixel phones: https://support.google.com/pixelphone/answer/4457705?hl=en

[24] Essential devices: https://www.theverge.com/2020/2/12/21134995/essential-phone-software-updates-security-android-startup

Android OS version	<8	8 (Oreo)	8.1	9 (Pie)	10 (Q)
Treble devices	0%	7.84%	4.07%	35.65%	52.43%
Android One devices	0%	0%	0.29%	75.19%	24.52%
All active devices	11.56%	11.62%	3.46%	54.16%	19.20%

Table 3: OS distributions for Treble, Android One, and all active devices – applies to active user devices that accessed the mobile app at least once a month, for at least 10 months in 2019. There are a total of 1,273,571 active devices, including 427,291 Treble devices and 1,048 Android One devices.

In the carrier dataset, only T-Mobile and AT&T announced upgrades for Android 10, both applied only to Samsung Galaxy S10, s10+, and S10e models. Inspecting these three Samsung models associated with either AT&T and T-Mobile in the end-user data, we find the difference between the carrier release date and the date of the first Android 10 observation on the device to be 7.67 days on average. Again, considering that each device accesses the mobile app once every 6.12 days on average, this shows most users do upgrade soon after Android 10 was made available on their devices.

While the nature of our end-user data (i.e., driven by when users choose to login to the app) prevents us from measuring user-incurred update delays at a more granular level, the results show that any uptake delay from the end users is negligible compared to that posed by the manufacturers and carriers.

7 EFFECTIVENESS OF ANDROID INITIATIVES

To assist the fragmented ecosystem, Google and its collaborators initiated various programs to increase the availability of security updates to end devices. We discuss each of them and attempt to empirically analyze their effectiveness using the end-user dataset.

Android One: Android One is a hardware and software standard created to run a near-stock version of Android on participating phone models. Participating models are promised at least two OS upgrades and three years of monthly security updates [25]. However, phone models in this program are selected by Google on a case-by-case basis. As of December 10, 2019, there are 23 participating model listed on the Android One website [1].

We identified 1,048 user devices participating in Android One. Of those devices, 75.19% are on Android 9 by the end of 2019, and 24.52% are on Android 10. No devices run versions older than 8.1. Compared to all active devices in the end-user dataset, where 54.16% are on Android 9 and 19.19% on Android 10, Android One models are indeed much more up-to-date, see Table 3.

However, even though Android One devices receive OS upgrades, they do not necessarily receive them faster than other devices. On average, 105.48 days passed since the official Android 10 release date until we observe it on an Android One device [26]. Additionally, despite that the program was initially launched in 2014 and evolved to include mid-range and high-end smartphones in 2017 [1], its footprint in our end-users dataset is less than 1%.

[25] Android One "standard": https://www.xda-developers.com/best-android-one/

[26] A separate article, posted in February 2020, discusses how Android One devices receive OS upgrades at variable times, and that ultimately the manufacturers still control the release of OS upgrades, https://www.notebookcheck.net/The-Android-One-program-is-a-shambles-and-here-s-why.454848.0.html.

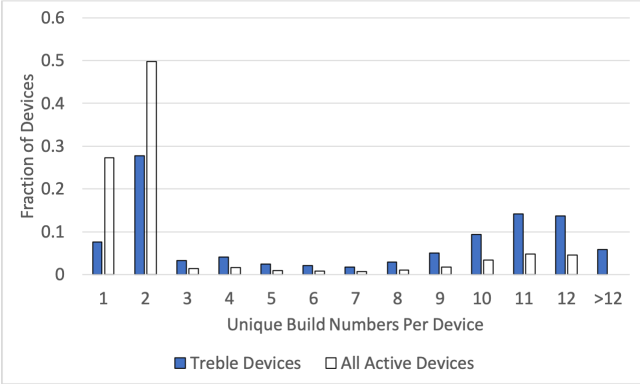

Figure 12: Treble vs. all Active Devices - Distribution of unique build numbers per device in 2019, comparing Treble devices with all active devices.

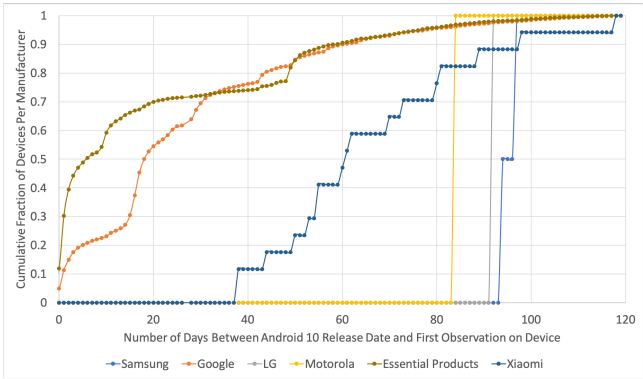

Figure 13: Android 10 release date on *Treble* end-user devices – The time difference between Android 10 release (September 3, 2019) and the date that Android 10 was first observed on a Treble device, per device manufacturer.

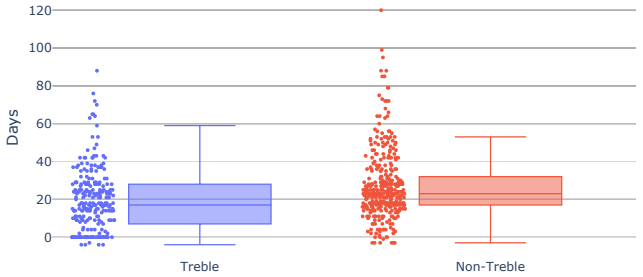

Figure 14: Distribution of update rollout latency for Treble and non-Treble devices – From the carrier dataset, Treble's average latency is 19 days versus 26 days for non-Treble.

Project Treble: Project Treble fundamentally alters the software stack by introducing a hardware abstraction layer (HAL), which separates lower-level, device-specific vendor implementations from the Android OS framework [48]. This helps reduce the overhead for manufacturers and chipset vendors when updating and upgrading devices. We identified 427,291 devices in the end-user data where the phone model is part of Project Treble support [27]. A first look at the number of unique build numbers per device (Figure 12) shows that Treble devices do appear to be receiving more frequent security updates. While only 16.55% of all active devices have more than nine unique build numbers over the course of 2019, 48.15% of Treble devices have more than nine build numbers.

In addition to a higher frequency of security updates, Treble devices also appear to be much more up-to-date in terms of OS upgrades compared to the general device population. Among end-user Treble devices, 35.65% are on Android 9 by the end of 2019, and 52.43% are on Android 10, as shown in Table 3. Considering that Treble includes phones shipped with Android 8 or later, the 11.91% of Treble devices that are still on Android 8 is an approximate of the end users who do not upgrade. Figure 13 shows the distribution of the latency between the Android 10 official release date and the date when we first observed Android 10 on the device in the end-user dataset, for each device manufacturer associated with Treble devices. Devices from Samsung, LG, and Motorola (18,460, 27,917, and 4,653 devices, respectively) appear to receive their OS upgrades on the same day, in contrast to the general population of devices where the rollout is staggered over time, as shown in Figure 11.

Compared to the general population, Treble devices do receive a higher number of security updates (approximated by the number of build changes) as well as access to new OS versions. It also appears that they are getting those updates faster from carriers and manufacturers. As shown in Figure 14, security updates for Treble devices are rolled out 7 days quicker than non-Treble devices on average, though there is still a median delay of 19 days.

This delay may be grounded into the current testing procedures. AOSP, manufacturers, and carriers try to quicken testing for all devices (regardless of Treble or not) by differentiating between security updates typically needing less than one week versus service updates lasting more than six weeks [5]. This may explain why Treble models do *not* experience a significant decrease in the average security update latency. Overall, Treble is a step in the right direction. However, the 19 day rollout delay is of concern.

Project Mainline: A more recent initiative, Project Mainline, further modularizes Android OS components. Announced in May 2019 and supported on devices launched with Android 10, it allows users to update certain system components directly through Google Play [4]. Mainline delivers those components as APK or APEX files, such that they can be updated like apps. This effectively removes device manufacturers and carriers in the software rollout process.

Lacking visibility on the end devices, our dataset does not allow us to directly measure the effect of Project Mainline (build numbers do not change). Nevertheless, we identified 140,720 active Mainline-supported devices, i.e., whose model is listed in the Mainline-supported devices [4]. This makes up 11.05% of the active devices in the end-user dataset — though the majority, 86.58%, of these Mainline-supported devices are Google Pixel phones. The outcome promised by Mainline could reduce the 24 days update latency incurred by manufacturers and carriers as described in Section 5.1. However, Mainline's low adoption among non-Google devices corroborated with the lack of support for older OS versions may prolong viewing its benefits by a considerable amount of time.

[27] Treble models: https://github.com/phhusson/treble_experimentations/wiki

8 STUDY LIMITATIONS

Even though we believe that our work is representative in the current Android landscape, there are a few considerations and limitations that require additional discussion.

Completeness of the study and data bias: Our study is geared towards carrier-supported devices and covers only the top U.S. carriers with their associated manufacturers, while the end-user dataset also consists largely of U.S.-based users. Many smaller carriers exist in the U.S., but the top four carriers we studied covered over 98% of the subscribers in 2018 [18]. When attempting to include other countries, we could only locate carriers posting limited information about updates (e.g., Vodafone-AU, Telus-CA, Orange-EU [28]).

Our work focuses on the update process after a patch is available in the AOSP. We do not consider delays at the code-commit level nor do we capture the delays caused by chipset vendors in developing a patch. Due to the fragmentation of the Android ecosystem, such a "boundary-setting" approach allows us to solely concentrate on entities such as manufacturers, carriers, and end users.

Accuracy in the measurements: As described in Section 4.2, we found many formatting inconsistencies across carriers' websites. Overall, our assumptions prioritize conservative calculations of the carrier's frequency. This means that a model could potentially receive more updates than our measurements report. Also, sometimes the release date of a security update announcement is not explicitly provided, thus we infer this information from other fields if possible. More details on data processing are available in Appendix A.2.

Lacking visibility on the end-user devices, we infer the presence of security updates from user-agent strings. Our results are based on devices where the build numbers either follow the Android convention or within the carrier and manufacturer update announcement (more details located in Appendix A.3).

The lack of transparency in the reporting mechanism and, as a matter of fact, in the entire software supply chain as noted in previous work [14, 32] makes it cumbersome and challenging to track all software changes/updates on devices.

Model vs. device updates: Not all CVEs in a security bulletin apply to the same model across all carriers [11]. Furthermore, because we do not know whether a bulletin is applicable to each model, we assume a "skipped" bulletin is packaged into a later security update as bulletins should not be skipped [38].

Batch updates: Batch rollout by manufacturers-carriers could lengthen the amount of time before an update reaches the end device. Lacking visibility on the updating schedule, we attempt to quantify the batch rollout period using our data and discuss it with respect to the carrier and manufacturer delay in Section 6.1.

The blackbox of carrier-manufacturer relationships: Usually information about vendor relationships and their business decisions are not made public by the involved entities. Thus, the findings in our work are made based on empirical observations. Despite these circumstances, lack of data transparency, and the heavy fragmentation of the Android ecosystem, our work attempts to provide a grounded and balanced assessment of Android's security updates.

[28] International carriers: https://www.vodafone.com.au/support/device/software-updates, https://forum.telus.com/t5/Mobility/Software-Update-Schedule/ta-p/53566, https://www.orange.ro/info/gadgets/article/2115813

9 ACTIONABLE ADVICE & FUTURE WORK

Our results suggest several actions can be taken to improve the current Android security landscape:

Manufacturer/carrier consistency and transparency: While multiple factors affect the delivery of security updates, the main bottleneck remains the carrier-manufacturer delay, including delay from batch rollout. Synchronizing all entities in the Android landscape may not be feasible (e.g., business decisions, too many entities, etc.), but what can help is *consistency and transparency in update announcements, specifically what and when vulnerabilities are patched.* For example, including the Android Security Bulletin, addressed CVEs, the initial release date and when devices have access to the update would yield more precise measurements. This would also enable independent auditing, potentially alleviating pressure for carriers and manufacturers to perform in-house measurements.

Initiatives independent from manufacturers/carriers: Android initiatives such as Treble, Android One, or Mainline are steps in the right direction. However, a reduced footprint, restrictive support, and the influence of carriers and manufacturers limit their benefits. In the end-user data, only 1.5% (excluding Google Pixel phones) of devices are Mainline-compatible, and less than 1% are Android One devices. Furthermore, as discussed in Section 7, updates for Treble devices are directly impacted by manufacturers' and carriers' update release processes. Thus, *Android-led initiatives should strive for more independence from manufacturers/carriers.*

Continuously tracking progress: Security update latency is relatively stable over 4 years. As shown in Section 5.2, CVE severity, type of vulnerabilities, or the number of CVEs included in a security update do not generally impact the rollout latency. To ensure change, it is important (especially for the research community) *to continuously track progress in this complex ecosystem.*

10 CONCLUSIONS

This paper presents an extensive quantitative study on Android's security updates. We provide measurements on the impact of manufacturers, carriers, end users and the effectiveness of Google-led projects on security updates and OS upgrades. The paper empirically quantifies the scale of the problem (e.g., rollout latency, frequency) and provides additional insights (e.g., locked vs. unlocked, Treble vs. non-Treble devices) from new perspectives.

Availability: Our collection scripts and all data gathered from public resources are available at https://github.com/undo-lab/Deploying-Android-Security-Updates-Carrier-Dataset.

ACKNOWLEDGMENTS

We are thankful to the anonymous reviewers for their insightful feedback and their suggestions for improving this paper. This research is supported by the National Science Foundation (NSF) under Award No. 1915824. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

REFERENCES

[1] Android. Accessed: 5/2020. Android One. https://www.android.com/one/.

[2] Olabode Anise. Accessed: 04/2020. Thirty Percent of Android Devices Susceptible to 24 Critical Vulnerabilities. https://duo.com/decipher/thirty-percent-of-android-devices-susceptible-to-24-critical-vulnerabilities.

[3] Macy Bayern. Accessed: 04/2020. Xperia owners: Here's when you'll get Android Pie. https://www.techrepublic.com/article/xperia-owners-heres-when-youll-get-android-pie-and-why-updates-take-so-long/.

[4] Android Developers Blog. Accessed: 05/2020. Fresher OS with Projects Treble and Mainline. https://android-developers.googleblog.com/2019/05/fresher-os-with-projects-treble-and-mainline.html.

[5] US Federal Trade Commission. Accessed: 03/2020. Mobile Security Updates. https://www.ftc.gov/system/files/documents/reports/mobile-security-updates-understanding-issues/mobile$_{s}$ecurity$_{u}$pdates$_{u}$nderstanding$_{t}$he$_{i}$ssues$_{p}$ublication$_{f}$inal.pdf.

[6] Erik Derr, Sven Bugiel, Sascha Fahl, Yasemin Acar, and Michael Backes. 2017. Keep Me Updated: An Empirical Study of Third-Party Library Updatability on Android. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (Dallas, Texas, USA) *(CCS '17)*. Association for Computing Machinery, New York, NY, USA, 2187–2200. https://doi.org/10.1145/3133956.3134059

[7] Android Developers. Accessed: 2/2020. Build anything on Android. https://developer.android.com.

[8] Droidlife. Accessed: 5/2020. Awesome Infographic: HTC Shows Us "The Anatomy of an Android OS Update" From PDK to OTA. https://www.droid-life.com/2013/12/26/awesome-infographic-htc-shows-us-the-anatomy-of-an-android-os-update-from-pdk-to-ota/.

[9] DroidViews. Accessed: 5/2020. Change CSC on Samsung Devices – Samsung CSC Codes. https://www.droidviews.com/how-to-change-csc-in-samsung-galaxy-phones/#What8217s$_{C}$SC$_{o}$n$_{S}$amsung.

[10] Sadegh Farhang, Mehmet Bahadir Kirdan, Aron Laszka, and Jens Grossklags. 2019. Hey Google, What Exactly Do Your Security Patches Tell Us? A Large-Scale Empirical Study on Android Patched Vulnerabilities. arXiv:1905.09352

[11] Sadegh Farhang, Mehmet Bahadir Kirdan, Aron Laszka, and Jens Grossklags. 2020. An Empirical Study of Android Security Bulletins in Different Vendors. In *Proceedings of The Web Conference 2020* (Taipei, Taiwan) *(WWW 2020)*. Association for Computing Machinery, New York, NY, USA, 3063–3069. https://doi.org/10.1145/3366423.3380078

[12] Jon Fingas. Accessed: 05/2020. Sony explains why Android updates take so long. https://www.engadget.com/2018/08/19/sony-explains-long-wait-for-android-updates/.

[13] Alain Forget, Sarah Pearman, Jeremy Thomas, Alessandro Acquisti, Nicolas Christin, Lorrie Faith Cranor, Serge Egelman, Marian Harbach, and Rahul Telang. 2016. Do or Do Not, There is No Try: User Engagement May Not Improve Security Outcomes. In *Proceedings of the Twelfth USENIX Conference on Usable Privacy and Security* (Denver, CO, USA) *(SOUPS '16)*. USENIX Association, USA, 97–111.

[14] Julien Gamba, Mohammed Rashed, Abbas Razaghpanah, Juan Tapiador, and Narseo Vallina-Rodriguez. 2019. An Analysis of Pre-installed Android Software. arXiv:1905.02713

[15] Chris Gold. Accessed: 05/2020. Why Buy an Unlocked Phone: A Primer. https://www.bhphotovideo.com/explora/portable-entertainment/features/unlocked-phones-primer.

[16] Google/Android. Accessed: 05/2020. Android Security & Privacy 2018 Year In Review. https://source.android.com/security/reports/Google$_{A}$ndroid$_{S}$ecurity$_{2}$018$_{R}$eport$_{F}$inal.pdf.

[17] Simon Hill. Accessed: 05/2020. What is Android fragmentation, and can Google ever fix it? https://www.digitaltrends.com/mobile/what-is-android-fragmentation-and-can-google-ever-fix-it/.

[18] Arne Holst. Accessed: 05/2020. Number of subscribers to wireless carriers in the U.S. from 1st quarter 2013 to 3rd quarter 2018, by carrier. https://www.statista.com/statistics/283507/subscribers-to-top-wireless-carriers-in-the-us/.

[19] Kenna Security & Cyentia Institute. Accessed: 05/2020. Prioritization To Prediction. https://www.kennasecurity.com/prioritization-to-prediction-report/images/Prioritization$_{t}$o$_{P}$rediction.pdf.

[20] Iulia Ion, Rob Reeder, and Sunny Consolvo. 2015. "...No one Can Hack My Mind": Comparing Expert and Non-Expert Security Practices. In *Eleventh Symposium On Usable Privacy and Security (SOUPS 2015)*. USENIX Association, Ottawa, 327–346. https://www.usenix.org/conference/soups2015/proceedings/presentation/ion

[21] Azzief Khaliq. Accessed: 05/2020. Android Fragmentation: The Story So Far. https://www.hongkiat.com/blog/android-fragmentation/.

[22] Jakob Lell and Karsten Nohl. Accessed: 05/2020. Mind the Gap - Uncovering the Android patch gap through binary-only patch analysis. https://conference.hitb.org/hitbsecconf2018ams/sessions/mind-the-gap-uncovering-the-android-patch-gap-through-binary-only-patch-level-analysis/.

[23] Frank Li and Vern Paxson. 2017. A Large-Scale Empirical Study of Security Patches. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (Dallas, Texas, USA) *(CCS '17)*. Association for Computing Machinery, New York, NY, USA, 2201–2215. https://doi.org/10.1145/3133956.3134072

[24] Mario Linares-Vásquez, Gabriele Bavota, and Camilo Escobar-Velásquez. 2017. An empirical study on android-related vulnerabilities. In *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)* (Buenos Aires, Argentina). IEEE, Institute of Electrical and Electronics Engineers, New York, NY, USA, 2–13.

[25] Arunesh Mathur and Marshini Chetty. 2017. Impact of User Characteristics on Attitudes Towards Automatic Mobile Application Updates. In *Thirteenth Symposium on Usable Privacy and Security (SOUPS 2017)*. USENIX Association, Santa Clara, CA, 175–193. https://www.usenix.org/conference/soups2017/technical-sessions/presentation/mathur

[26] Arunesh Mathur, Josefine Engel, Sonam Sobti, Victoria Chang, and Marshini Chetty. 2016. "They Keep Coming Back Like Zombies": Improving Software Updating Interfaces. , 43–58 pages. https://www.usenix.org/conference/soups2016/technical-sessions/presentation/mathur

[27] Arunesh Mathur, Nathan Malkin, Marian Harbach, Eyal Péer, and Serge Egelman. 2018. Quantifying Users' Beliefs about Software Updates. arXiv:1805.04594 http://arxiv.org/abs/1805.04594

[28] Jerome L Myers, Arnold Well, and Robert Frederick Lorch. 2010. *Research design and statistical analysis*. Routledge, Abingdon-on-Thames, England, UK.

[29] A. Nappa, R. Johnson, L. Bilge, J. Caballero, and T. Dumitras. 2015. The Attack of the Clones: A Study of the Impact of Shared Code on Vulnerability Patching. In *2015 IEEE Symposium on Security and Privacy*. Institute of Electrical and Electronics Engineers, New York, NY, USA, 692–708.

[30] Jema David Ndibwile, Edith Talina Luhanga, Doudou Fall, Daisuke Miyamoto, and Youki Kadobayashi. 2018. Smart4Gap: Factors That Influence Smartphone Security Decisions in Developing and Developed Countries. In *Proceedings of the 2018 10th International Conference on Information Management and Engineering* (Salford, United Kingdom) *(ICIME 2018)*. Association for Computing Machinery, New York, NY, USA, 5–15. https://doi.org/10.1145/3285957.3285980

[31] Bogdan Petrovan. Accessed: 08/2020. Google engineer explains how Android updates roll out and why you shouldn't force them. https://www.androidauthority.com/how-android-updates-roll-out-force-clear-gcf-318744/.

[32] Joel Reardon, Álvaro Feal, Primal Wijesekera, Amit Elazari Bar On, Narseo Vallina-Rodriguez, and Serge Egelman. 2019. 50 Ways to Leak Your Data: An Exploration of Apps' Circumvention of the Android Permissions System. In *28th USENIX Security Symposium*. USENIX Association, Santa Clara, CA, 603–620. https://www.usenix.org/conference/usenixsecurity19/presentation/reardon

[33] Samsung. Accessed: 05/2020. About Bring Your Own Device (BYOD) for Galaxy Phones. https://www.samsung.com/us/support/answer/ANS00078492/.

[34] Samsung. Accessed: 05/2020. Home / Phones / All Phones / Unlocked by Samsung. https://www.samsung.com/us/mobile/phones/all-phones/?carrier=Unlocked+by+Samsung.

[35] Armin Sarabi, Ziyun Zhu, Chaowei Xiao, Mingyan Liu, and Tudor Dumitraş. 2017. Patch me if you can: A study on the effects of individual user behavior on the end-host vulnerability state. , 113-125 pages.

[36] Ben Schlabs. Accessed: 08/2020. Mind the Gap. https://www.qualcomm.com/sites/ember/files/uploads/180516.srlabs-mind$_{t}$he$_{g}$ap-android$_{p}$atch$_{g}$ap-qualcomm$_{s}$ummit.ben$_{s}$chlabs.pdf.

[37] Michael Simon. Accessed: 05/2020. Android security: Why Google's demands for updates don't go far enough. https://www.pcworld.com/article/3316717/android-security-requirements-partners.html.

[38] Android Source. Accessed: 05/2020. Android Security Bulletins. https://source.android.com/security/bulletin.

[39] Android Source. Accessed: 05/2020. Security & Privacy in Android 10. https://source.android.com/security/enhancements/enhancements10.

[40] Android Source. Accessed: 08/2020. A/B (Seamless) System Updates. https://source.android.com/devices/tech/ota/ab.

[41] Laerd Statistics. Accessed: 04/2020. Spearman's Rank-Order Correlation. https://statistics.laerd.com/statistical-guides/spearmans-rank-order-correlation-statistical-guide-2.php.

[42] T-Mobile Support. Accessed: 05/2020. My Galaxy S7 still says AT&T software and firmware after switch. https://support.t-mobile.com/thread/140331.

[43] T-Mobile Support. Accessed: 05/2020. Switched S9 from Verizon to T-Mobile - Phone won't rebrand. https://support.t-mobile.com/thread/146826.

[44] T-Mobile. Accessed: 12/2019. Support Devices. https://support.t-mobile.com/community/phones-tablets-devices#phone/.

[45] Daniel R. Thomas. 2015. The Lifetime of Android API Vulnerabilities: Case Study on the JavaScript-to-Java Interface (Transcript of Discussion). In *Security Protocols 2015 (Lecture Notes in Computer Science)*. Springer, Cham, 139–144. https://doi.org/10.1007/978-3-319-26096-9$_{1}$4

[46] Daniel R. Thomas, Alastair R. Beresford, and Andrew Rice. 2015. Security Metrics for the Android Ecosystem. In *Proceedings of the 5th Annual ACM CCS Workshop on Security and Privacy in Smartphones and Mobile Devices* (Colorado, USA)

(SPSM '15). Association for Computing Machinery, New York, NY, USA, 87–98. https://doi.org/10.1145/2808117.2808118

[47] Davey Winder. Accessed: 05/2020. Smartphone Security Surprise As Samsung Shows Google How Android Updates Can Be Done. https://www.forbes.com/sites/daveywinder/2020/01/06/smartphone-security-surprise-as-samsung-shows-google-how-android-updates-can-be-done-note10-galaxys10-pixel/#6752ce7f1d6e.

[48] Keun Soo Yim, Iliyan Malchev, Andrew Hsieh, and Dave Burke. 2019. Treble: Fast Software Updates by Creating an Equilibrium in an Active Software Ecosystem of Globally Distributed Stakeholders. *ACM Transactions on Embedded Computing Systems (TECS)* 18, 5s (2019), 1–23.

A APPENDIX

The Appendix section provides more details and supporting information for previously discussed matters.

A.1 Carrier Data: Collection Process

Each carrier's announcement style for Android updates presented its own unique challenges during the collection process. In this appendix section, we detail our methodology for collecting data from the top four U.S. carriers (AT&T, Sprint, T-Mobile, Sprint).

Accessing and saving update announcements: AT&T, T-Mobile, and Verizon provided one page with separate links to each models (similar to a table of index). These individual links contained additional links to the update announcements for each of those models. For T-Mobile and Verizon, we programmatically traversed through these URLs and locally downloaded the HTML code using PyQt5 [29]. On the other hand, AT&T employed a different set of features (e.g., Javascript, wait conditions, cookies, pop-ups). Thus, for AT&T, we manually recorded each device announcement page that indicated updates for Android Security Bulletins and leveraged Selenium[30] to download these announcements.

Sprint posts update announcements in a blog-like style, meaning each individual blog post contained an announced update. We programmatically identified update announcements by traversing through each post preview (prevented us from accessing each individual post) and by checking if the post contained one of the following: "Software Version:",‟software update - version", or "Software Update". Similar to T-Mobile and Verizon, we locally saved each update post with PyQt5. At all times, we employed various measures to minimize access upon collecting the data (as mentioned in Section 4.5).

Extracting specific data fields: Each carrier required a slightly different method to best capture each data field. Overall, we used regular expressions (regex) to match the fields, relied on HTML tags to traverse through each update, or combined the two. Aside from the model's name, which is always captured via traversal of HTML tags, collecting other data features varied across carriers.

AT&T. Each model announcement page contains the current update and a table of all the earlier/previous updates. To handle this discrepancy, we leveraged both regex and HTML tags to collect each update. The current update was located upon matching, "Here is the current update". We then used additional regex strings to match other fields such as OS version, software version, file size, baseband version, build number, security level, and extra details. In the previous/historical updates, these tend to vary and contain less information, but we could still use the same regex strings as the current update. These previous updates are captured by traversing the HTML table row tags and then the fields are identified via regex.

Sprint. Each post typically provides the software version (an extended version of the build number), the release date, the release method (typically Over-The-Air, OTA), and what was fixed in the announcement. Each of these fields are captured with the respective string match where newlines are not matched: "Software Version:(.*)", "Release Date:(.*)", "Method:(.*)", and "Fixes:(.*)". Occasionally, for older posts, these strings slightly alter and tend to be more verbose (e.g., "the update starts on ([0-9/]+)").

T-Mobile. The announcements, provided in a table format, typically contain the version (build number), release date, enhancements and status. Because the updates are displayed in a table, we collected them by traversing each row indicated by the HTML code.

Verizon. With announced updates separated into sections, we relied on regular expressions. We collected the release date, patch level (if provided), software version, the date when the announcement was last updated, and the details (if provided).

We also manually validated our programmatically collected data fields for all update announcements. If a field was not captured or not formatted correctly (e.g., release date, security level), we manually collected the data for that field.

[29] PyQt5: https://pypi.org/project/PyQt5/

[30] Selenium: https://www.selenium.dev/

Normalization Criteria	AT&T	Sprint	T-Mobile	Verizon	Running Total
Raw Android Updates	1022	483	1021	967	3,493
Contains Release Date	1018	479	823	921	3,241
Contains Android Bulletin Indicator	533	380	815	893	2,621
Released within analysis window (8/15 - 12/19)	504	382	709	891	2,486
"Non-duplicate" Prepaid&Contract Models	504	382	709	860	2,455
Reducing Duplicates and Noise	440	369	664	612	2,085
Located Market Release Year for Models	408	368	678*	570	2,024
Level 1 Bulletins	394	364	658	537	1,953
Removal of outliers	381	363	654	469	**1,867**

Total number of "valid" security updates after normalization **1,867**

Table 4: Processing security updates data from carriers – Each (normalization) step removes non-conforming data from each carrier. We process all Android updates from carriers and continue to remove updates at each step based on the stated criteria. It's important to note the need for transparency and consistency across these update announcements as quite a few updates are dropped in our analysis due to incomplete information (e.g., from missing release dates to what is addressed within an update).
***Increased due to Verizon posting one update but indicating multiple models.**

A.2 Carrier Data: Processing & Normalization

We collected a total of 3,493 raw Android updates (OS upgrades, security updates, and miscellaneous updates) across the four mobile carriers. In this appendix section, we describe and discuss how our normalization and filtering methods directly impact the data summarized in Table 4. Overall, we excluded 1,627 updates due to various reasons such as non-security nature of updates, incomplete data in update announcements, etc. This resulted in a total number of 1,867 "valid" updates that we used in our analysis. Below we discuss our data normalization process.

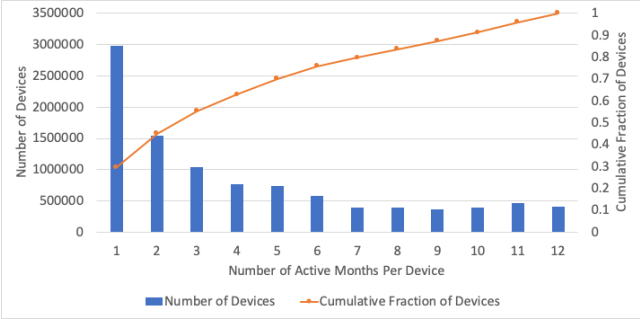

Figure 15: The distribution of the number of active months per device in the end-user dataset.

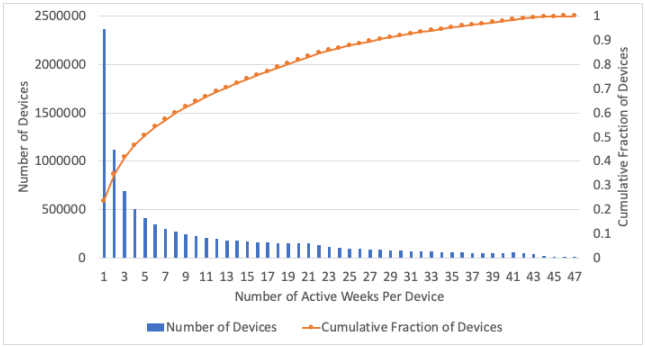

Figure 16: The distribution of the number of active weeks per device in the end-user dataset.

As shown in Table 4, our carrier data normalization can be summarized as a series of subsequent steps (each "removal" step inherits the remaining number of updates from the previous ones):

(1) Gather all Android raw updates from carriers' public websites
(2) Remove updates without an identifiable release date
(3) Remove updates without Android Bulletin indicators (e.g., "Android update")
(4) Remove updates released prior to August 2015 (before the start of Android bulletins) and after December 2019.
(5) Remove duplicate prepaid and contract models.
(6) Remove duplicate and non-security updates.
(7) Remove updates from models without a market release year.
(8) Remove updates applied to higher Android bulletin levels if level 1 of the same bulletin was addressed (as higher levels include level 1).
(9) Remove outliers (updates with a delay less than -10 and greater than 170 days as we cannot verify the accuracy of these updates, e.g., possible typos.)

To calculate the carrier and manufacturer incurred delay, we use the time from when Android posted their bulletin to when the manufacturer and carrier released the corresponding update for that bulletin. Thus, we can only leverage updates containing both the release date and the Android bulletin (or an indication if not explicitly stated). If an update contains "Android updates" or "Google updates", we consider the update applies to the closest, prior bulletin based on the release date. For Verizon, in cases where the release date of the software update is not explicitly provided, we infer this information from the "Updated:" field, which is the date the post was posted or revised. We also filter out updates with a rollout latency higher than 150 days as this could be due to the estimation error. Thus, this resulted in 2,621 updates (see Table 4).

We also removed updates released prior to August 2015 and after December 2019. Google started releasing Android Security Bulletins in August 2015 while our end-user data ends on December 31, 2019. Next we observed that carriers (specifically, Verizon) listed updates for some prepaid models separate from contract models. Thus, we removed 41 overlapping updates where prepaid models had corresponding contract device models.

Due to our collection method, a few updates for the same model and carrier indicated the same Android Security Bulletin. This means models received miscellaneous updates whilst on the same bulletin. We removed the updates that were not specifically security updates which is labelled as noise as labeled in Table 4. Furthermore, because we collected data from Verizon three times, certain updates overlapped. We also removed these duplicates.

To calculate the update frequency of each model, we use the model's marketplace release date. Since carriers do not always release this information, we leveraged other data sources (such as GSMArena [31] and PhoneArena [32]) to manually collect 431 device release dates. We could not locate the device release date for 32 models thus excluding 61 updates, as shown in Table 4.

Some models received multiple updates covering different levels in the same Android bulletin. Because we focus on the first level bulletins, we drop updates to higher levels if their corresponding first level was also addressed. In the event only a "higher level" was addressed, we keep the update as higher levels must also address all lower levels. We dropped 87 updates based on this criteria.

Lastly, we remove any outliers within our corpus based on the delay. When manually validating our programmatic collection and normalization methods, we noticed a update's release date almost a year prior to the release of a bulletin. We cannot validate the accuracy of this, thus we removed an update if the delay was less than -10 days[33] or greater than 170 days. This resulted in the exclusion of 86 updates.

It is important to note that there were instances where carriers "packaged" update announcements such that one post contained multiple model names, thus meaning multiple updates were released. In our normalization method, we did not break this update apart until we located models' marketplace release dates, thus the total number of updates slightly increase during this step. In total, 12 announced updates applied to two different models. This increases the total number of updates from 12 to 24. Notice in Table 4, T-Mobile's count increased due to this issue as 10 of the 12 updates were from this carrier. The remaining 2 were from AT&T.

A.3 End-User Dataset Statistics

We provide additional statistics for the end-user dataset related to our measurement study.

[31] GSMArena: https://www.gsmarena.com/
[32] PhoneArena: https://www.phonearena.com/
[33] The negative refers to days prior to the release of the Android bulletin

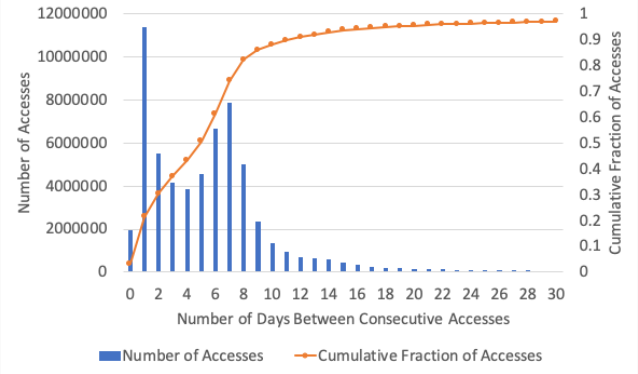

Figure 17: The distribution of the time (in days) between consecutive accesses to the app from the same end-user device.

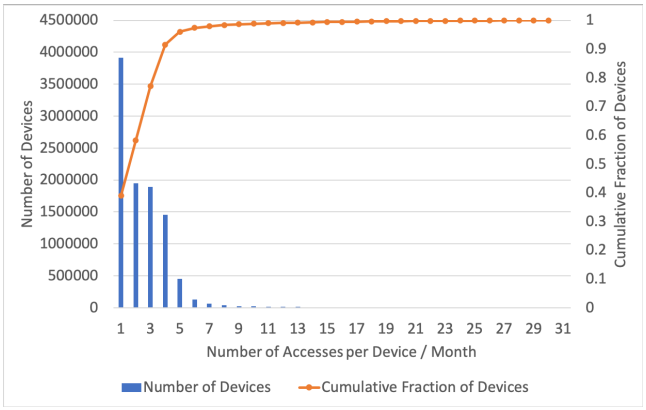

Figure 18: The distribution of number of accesses per month for each device. For a device, this is the number of total accesses over the year divided by the number of months in which the device is active.

The end-user dataset only included Android traffic containing build number information, as determined by the presence of "Android" and "Build" in the user-agent strings. This excludes traffic from Android devices without these keywords. Minus more intrusive data collection from the devices, our methodology is guided by available information, i.e., build and OS version in user-agent strings, that would allow us to observe security update behavior.

In Section 6, we focus our analysis on "active" devices that exhibit continued activities throughout the duration of the data. Figure 15 and 16 show the distribution of the number of active months and weeks per device, respectively. We define an "active" device as one that is active for at least 10 months out of the year, which results in 1,273,571 devices considered for the analysis in Section 6. We select 10 months due to the presence of a 1.5 months gap in our HTTP requests data collection, and also consider using month as a unit (rather than week) since Android Security Bulletins are issued on a monthly basis. As an additional data point, Figure 18 shows the distribution of monthly accesses per device, calculated by taking the number of total accesses over the year from that device divided by the number of months in which the device is active.

In measuring update latency in the end-user dataset, our observations may be affected by when devices access the mobile app. Figure 17 shows the number of days between consecutive accesses from the same device. On average, each device accesses the app once every 6.12 days, with a median value of six days. 21.4% of the accesses occurred within one day after the immediately preceding access, and overall 92.9% of the accesses occurred within two weeks after the immediately preceding access.

We also acknowledge that the log timestamps in the end-user dataset are entirely driven by the users of the mobile app. The user-incurred latency measurements is hence affected by user behavior, and the activities we observe may not be consistent across all devices.

Additionally, the end-user dataset has a gap from June 11 to July 23, 2019. To assess the gap's impact when measuring latency, we repeated the measurements in Section 6.1 but removed device build changes observed after the data gap that correspond to carrier released updates before the gap (0.3% of devices are affected). This analysis yielded a median delay of 11 days, consistent with our earlier results. In addition, we split the dataset by examining only the first 5 months (prior to the gap) and the last 5 month (after the gap), remove the criteria to only consider "active" devices, and repeat our measurements. Our findings are still consistent - the median is 11 days, though the average value from the last 5 months is 12.10 days, slightly shorter than the 15.12 observed across the entire dataset.

A.4 Security Update Frequency

To better highlight how manufacturers and carriers' relationships impact frequency, we compare Samsung and LG. Figures 19 and 20 show the normalized update frequency across carriers for models from Samsung and LG, respectively. Samsung models have a median update frequency of 0.54 when associated with AT&T, much higher than the 0.28 to 0.32 range with other carriers. A similar trend can be observed for LG models with T-Mobile (0.42) where other carriers range 0.12 to 0.33. Interestingly, the update frequency for LG models across all carriers seem to be capped at 0.75, while this does not appear to be the case for Samsung.

A.5 Security Updates on Locked and Unlocked Samsung Devices

When analyzing locked and unlocked Samsung models, we observed slight variations occurring with models tied to carriers. Updates still flow from Android Security Bulletins to manufacturers; however, when the carrier intervenes in the process, three possibilities can occur. First, locked models purchased from a carrier still receive updates from Samsung; however, the carrier determines when those updates are released. Furthermore, if these models are unlocked by that carrier and remain on the same network, the model continues to receive updates from Samsung but can be delayed by request of the mobile carrier [15]. Second, models purchased directly from Samsung without carrier branding are labelled as unbranded if and only if these have yet to be connected to a mobile network. These models receive updates from Samsung without the carrier's involvement [9]. Lastly, models purchased unlocked then placed

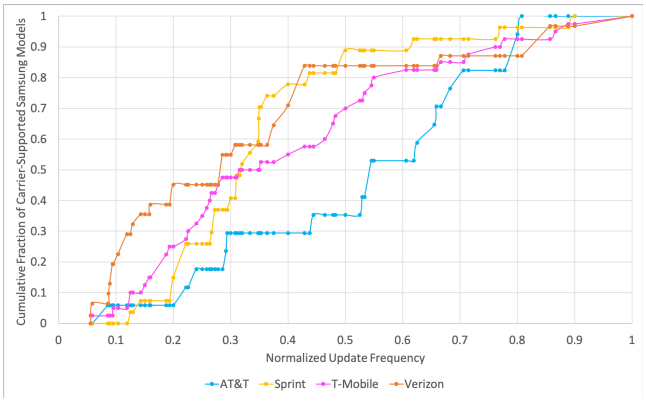

Figure 19: Cumulative distribution of the normalized security update frequency for Samsung models, for each carrier – Samsung models on AT&T appear to receive updates more frequently than those on other carriers.

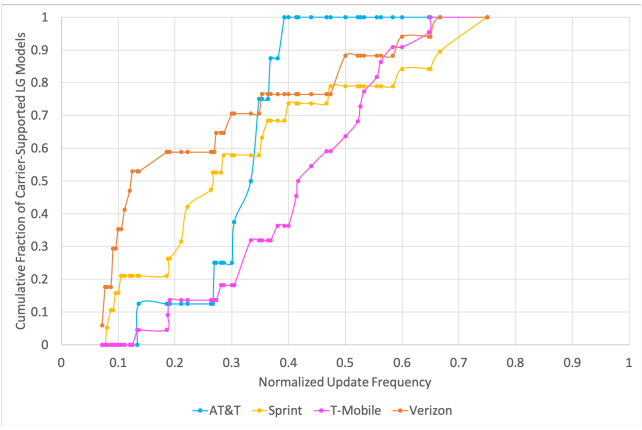

Figure 20: Cumulative distribution of the normalized security update frequency for LG models, for each carrier – LG models on T-Mobile appear to generally receive updates more frequently than those on other carriers.

onto the carrier's network receive updates from the Samsung, yet the carrier can still choose to delay updates [34].

It is worth pointing out that devices can transfer from one carrier to another. However, many issues can arise which makes determining the update process dependent on the specific carrier and manufacturer [33, 42, 43].

A.6 Correlating Variables Affecting Latency

Using Spearman's correlation, we calculated the correlation between security update latency from the manufacturer and carrier against multiple factors: the total CVE counts, CVE severity, CVE type, CVE classification, prior missed Android Security Bulletins, and the number of days since the prior update.

[34] Samsung community updates: https://us.community.samsung.com/t5/Galaxy-S9/Once-a-phone-is-unlocked-who-does-the-software-updates/m-p/563811#M17572

Category	Sub-category	carrier	r-value	p-value	n(points)
CVE Severity	High	AT&T	-0.1188	0.4425	44
CVE Severity	High	Sprint	-0.0279	0.8491	49
CVE Severity	High	T-Mobile	0.0173	0.9022	53
CVE Severity	High	Verizon	0.0462	0.7658	44
CVE Severity	Critical	AT&T	-0.295	0.0519	44
CVE Severity	Critical	Sprint	-0.1991	0.1703	49
CVE Severity	Critical	T-Mobile	-0.3884	0.0041	53
CVE Severity	Critical	Verizon	0.0727	0.6393	44
CVE Severity	Moderate	AT&T	0.0413	0.7901	44
CVE Severity	Moderate	Sprint	-0.3143	0.0279	49
CVE Severity	Moderate	T-Mobile	-0.4025	0.0028	53
CVE Severity	Moderate	Verizon	-0.5518	0.0001	44
CVE Severity	Low	AT&T	-0.2432	0.1116	44
CVE Severity	Low	Sprint	0.1372	0.3473	49
CVE Severity	Low	T-Mobile	-0.3091	0.0243	53
CVE Severity	Low	Verizon	-0.3048	0.0443	44
CVE Type	EoP	AT&T	0.116	0.4535	44
CVE Type	EoP	Sprint	0.314	0.028	49
CVE Type	EoP	T-Mobile	0.4149	0.002	53
CVE Type	EoP	Verizon	0.6463	0	44
CVE Type	RCE	AT&T	0.2135	0.164	44
CVE Type	RCE	Sprint	0.1898	0.1915	49
CVE Type	RCE	T-Mobile	0.0562	0.6895	53
CVE Type	RCE	Verizon	0.5186	0.0003	44
CVE Type	DoS	AT&T	-0.0639	0.6803	44
CVE Type	DoS	Sprint	0.3914	0.0054	49
CVE Type	DoS	T-Mobile	0.4595	0.0005	53
CVE Type	DoS	Verizon	0.3459	0.0214	44
CVE Type	ID	AT&T	0.0219	0.888	44
CVE Type	ID	Sprint	0.2405	0.096	49
CVE Type	ID	T-Mobile	0.2417	0.0812	53
CVE Type	ID	Verizon	0.5123	0.0004	44
CVE Classification	Android runtime	AT&T	0.4012	0.007	44
CVE Classification	Android runtime	Verizon	0.3487	0.0204	44
CVE Classification	Media framework	Sprint	0.311	0.0296	49
CVE Classification	Media framework	Verizon	0.5558	0.0001	44
CVE Classification	System	Sprint	0.4566	0.001	49
CVE Classification	System	T-Mobile	0.6164	0	53
CVE Classification	System	Verizon	0.467	0.0014	44
CVE Classification	Framework	T-Mobile	0.4098	0.0023	53
CVE Classification	Framework	Verizon	0.3819	0.0105	44
CVE Classification	Elevation of Privilege Vulnerability in Qualcomm Wi-Fi Driver	AT&T	-0.339	0.0244	44
CVE Classification	Remote Code Execution Vulnerabilities in Mediaserver	Verizon	-0.3081	0.0419	44
CVE Classification	Remote Code Execution Vulnerability in Mediaserver	AT&T	-0.4972	0.0006	44
CVE Classification	Remote Code Execution Vulnerability in Mediaserver	T-Mobile	-0.3126	0.0227	53
CVE Classification	Elevation of Privilege Vulnerability in Mediaserver	AT&T	-0.339	0.0244	44
CVE Classification	Information Disclosure Vulnerability in Mediaserver	AT&T	-0.3396	0.0241	44
CVE Classification	Elevation of Privilege Vulnerability in Wi-Fi	AT&T	-0.327	0.0303	44
CVE Classification	Elevation of Privilege Vulnerability in Wi-Fi	Sprint	-0.3219	0.0241	49
CVE Classification	Denial of service vulnerability in Mediaserver	AT&T	0.329	0.0292	44
CVE Classification	Denial of service vulnerability in Mediaserver	Verizon	-0.3318	0.0278	44
CVE Classification	Information disclosure vulnerability in Mediaserver	Verizon	-0.3113	0.0397	44
CVE Classification	Elevation of privilege vulnerability in Framework APIs	AT&T	0.3783	0.0113	44
CVE Classification	Elevation of privilege vulnerability in Audioserver	AT&T	0.3086	0.0415	44
Total CVE Count	N/A	Bulletin Average	-0.2812	0.0414	53
Prior Missed Android Security Bulletins	N/A	All	-0.0355	0.1859	1387
Days Since Last Update	N/A	All	0.1809	0	1387

Table 5: Correlation between latency and other factors – The results of Spearman's correlation (r-value) between security update rollout latency and various variables. Under CVE Classification, System was conclusive with a moderate correlation across three carriers. For CVE Classification, we only include the category with a p-value of 0.05 or less.

As shown in Table 5, we consider any correlation (r-value) with a p-value greater than 0.05 to be inconclusive [41]. Upon interpretation, a positive correlation means a higher category quantity leads to a higher security update latency. A negative correlation represents the opposite behavior (a higher category quantity leads to a lower security update latency). We leveraged 64 CVE classifications, but we only include the correlation coefficients with a p-value less than or equal to 0.05.