An Analysis of Pre-installed Android Software

Julien Gamba*†, Mohammed Rashed†, Abbas Razaghpanah‡, Juan Tapiador † and Narseo Vallina-Rodriguez*§

* IMDEA Networks Institute, † Universidad Carlos III de Madrid, ‡ Stony Brook University, § ICSI

Abstract

The open-source nature of the Android OS makes it possible for manufacturers to ship custom versions of the OS along with a set of pre-installed apps, often for product differentiation. Some device vendors have recently come under scrutiny for potentially invasive private data collection practices and other potentially harmful or unwanted behavior of the pre-installed apps on their devices. Yet, the landscape of pre-installed software in Android has largely remained unexplored, particularly in terms of the security and privacy implications of such customizations. In this paper, we present the first large-scale study of pre-installed software on Android devices from more than 200 vendors. Our work relies on a large dataset of real-world Android firmware acquired worldwide using crowd-sourcing methods. This allows us to answer questions related to the stakeholders involved in the supply chain, from device manufacturers and mobile network operators to third-party organizations like advertising and tracking services, and social network platforms. Our study allows us to also uncover relationships between these actors, which seem to revolve primarily around advertising and data-driven services. Overall, the supply chain around Android's open source model lacks transparency and has facilitated potentially harmful behaviors and backdoored access to sensitive data and services without user consent or awareness. We conclude the paper with recommendations to improve transparency, attribution, and accountability in the Android ecosystem.

I. Introduction

The openness of the Android source code makes it possible for any manufacturer to ship a custom version of the OS along with proprietary pre-installed apps on the system partition. Most handset vendors take this opportunity to add value to their products as a market differentiator, typically through partnerships with Mobile Network Operators (MNOs), online social networks, and content providers. Google does not forbid this behavior, and it has developed its Android Compatibility Program [8] to set the requirements that the modified OS must fulfill in order to remain compatible with standard Android apps, regardless of the modifications introduced.Devices made by vendors that are part of the Android Certified Partners program [5] come pre-loaded with Google's suite of apps (*e.g.*, the Play Store and Youtube). Google does not provide details about the certification processes. Companies that want to include the Google Play service without the certification can outsource the design of the product to a certified Original Design Manufacturer (ODM) [7].

Certified or not, not all pre-installed software is deemed as wanted by users, and the term "bloatware" is often applied to such software. The process of how a particular set of apps end up packaged together in the firmware of a device is not transparent, and various isolated cases reported over the last few years suggest that it lacks end-to-end control mechanisms to guarantee that shipped firmware is free from vulnerabilities [24], [25] or potentially malicious and unwanted apps. For example, at Black Hat USA 2017, Johnson *et al.* [82], [47] gave details of a powerful backdoor present in the firmware of several models of Android smartphones, including the popular BLU R1 HD. In response to this disclosure, Amazon removed Blu products from their Prime Exclusive line-up [2]. A company named Shanghai Adups Technology Co. Ltd. was pinpointed as responsible for this incident. The same report also discussed the case of how vulnerable core system services (*e.g.*, the widely deployed MTKLogger component developed by the chipset manufacturer MediaTek) could be abused by co-located apps. The infamous Triada trojan has also been recently found embedded in the firmware of several low-cost Android smartphones [77], [66]. Other cases of malware found pre-installed include Loki (spyware and adware) and Slocker (ransomware), which were spotted in the firmware of various high-end phones [6].

Android handsets also play a key role in the mass-scale data collection practices followed by many actors in the digital economy, including advertising and tracking companies. OnePlus has been under suspicion of collecting personally identifiable information (PII) from users of its smartphones through exceedingly detailed analytics [55], [54], and also deploying the capability to remotely root the phone [53], [52]. In July 2018 the New York Times revealed the existence of secret agreements between Facebook and device manufacturers such as Samsung [32] to collect private data from users without their knowledge. This is currently under investigation by the US Federal authorities [33]. Additionally, users from developing countries with lax data protection and privacy laws may be at an even greater risk. The Wall Street Journal has exposed the presence of a pre-installed app that sends users' geographical location as well as device identifiers to GMobi, a mobile-advertising agency that engages in ad-fraud activities [14], [67]. Recently, the European Commission publicly expressed concern about Chinese manufacturers like Huawei, alleging that they were required to cooperate with national intelligence services by installing backdoors on their devices [30].

Research Goals and Findings

To the best of our knowledge, no research study has so far systematically studied the vast ecosystem of pre-installed Android software and the privacy and security concerns associated with them. This ecosystem has remained largely unexplored due to the inherent difficulty to access such software at scale and across vendors. This state of affairs makes such an study even more relevant, since i) these apps – typically unavailable on app stores – have mostly escaped the scrutiny of researchers and regulators; and ii) regular users are unaware of their presence on the device, which could imply lack of consent in data collection and other activities.

In this paper, we seek to shed light on the presence and behavior of pre-installed software across Android devices. In particular, we aim to answer the questions below:

- What is the ecosystem of pre-installed apps, including all actors in the supply chain?
- What are the relationships between vendors and other stakeholders (*e.g.*, MNOs and third-party services)?
- Do pre-installed apps collect private and personally-identifiable information (PII)? If so, with whom do they share it?
- Are there any harmful or other potentially dangerous apps among pre-installed software?

To address the points described above, we developed a research agenda revolving around four main items:

1) We collected the firmware and traffic information from real-world devices using crowd-sourcing methods (§II). We obtained the firmware from 2,748 users spanning 1,742 device models from 214 vendors. Our user base covers 130 countries from the main Android markets. Our dataset contains 424,584 unique firmware files, but only 9% of the collected APKs were found in Google Play. We complement this dataset with traffic flows associated with 139,665 unique apps, including pre-installed ones, provided by over 20.4K users of the Lumen app [86] from 144 countries. To the best of our knowledge, this is the largest dataset of real-world Android firmware analyzed so far.
2) We performed an investigation of the ecosystem of pre-installed Android apps and the actors involved (§III) by analyzing the Android manifest files of the app packages, their certificates, and the Third-Party Libraries (TPLs) they use. Our analysis covers 1,200 unique developers associated with major manufacturers, vendors, MNOs, and Internet service companies. We also uncover a vast landscape of third-party libraries (11,665 unique TPLs), many of which mainly provide data-driven services such as advertisement, analytics, and social networking.
3) We extracted and analyzed an extensive set of custom permissions (4,845) declared by hardware vendors, MNOs, third-party services, security firms, industry alliances, chipset manufacturers, and Internet browsers. Such permissions may potentially expose data and features to over-the-top apps and could be used to access privileged system resources and sensitive data in a way that circumvents the Android permission model. A manual inspection reveals a complex supply chain that involves different stakeholders and potential commercial partnerships between them (§IV).
4) We carried out a behavioral analysis of nearly 50% of the apps in our dataset using both static and dynamic analysis tools (§V). Our results reveal that a significant part of the pre-installed software exhibit potentially harmful or unwanted behavior. While it is known that personal data collection and user tracking is pervasive in the Android app ecosystem as a whole [78], [84], [85], we find that it is also quite prevalent in pre-installed apps. We have identified instances of user tracking activities by pre-installed Android software – and embedded third-party libraries – which range from collecting the usual set of PII and geolocation data to more invasive practices that include personal email and phone call metadata, contacts, and a variety of behavioral and usage statistics in some cases. We also found a few isolated malware samples belonging to known families, according to VirusTotal, with prevalence in the last few years (*e.g.*, Xynyin, SnowFox, Rootnik, Triada and Ztorg), and generic trojans displaying a standard set of malicious behaviors (*e.g.*, silent app promotion, SMS fraud, ad fraud, and URL click fraud).

All in all, our work reveals complex relationships between actors in the Android ecosystem, in which user data seems to be a major commodity. We uncover a myriad of actors involved in the development of mobile software, as well as poor software engineering practices and lack of transparency in the supply chain that unnecessarily increase users' security and privacy risks. We conclude this paper with various recommendations to palliate this state of affairs, including transparency models to improve attribution and accountability, and clearer mechanisms to obtain informed consent. Given the scale of the ecosystem and the need to perform manual inspections, we will gradually make our dataset available to the research community and regulators to boost investigations.

II. Data Collection

Obtaining pre-installed apps and other software artifacts (*e.g.*, certificates installed in the system root store) at scale is challenging. As purchasing all the mobile handset models (and their many variations) available in the market is unfeasible, we decided to crowdsource the collection of pre-installed software using a purpose-built app: Firmware Scanner [34]. Using Firmware Scanner, we obtained pre-installed software from 1,742 device models. We also decided to use Lumen, an app that aims to promote mobile transparency and enable user control over their mobile traffic [86], [49] to obtain anonymized network flow metadata from Lumen's real users. This allows us to correlate the information we extract from static analysis, for a subset of mobile apps, with realistic network traffic generated by mobile users in the wild and captured in user-space. In the remainder of this section, we explain the methods implemented by each app and present our datasets. We discuss the ethical implications of our data collection in Section II-C.

A. Firmware Scanner

Publicly available on Google Play [34], Firmware Scanner is a purpose-built Android app that looks for and extracts pre-installed apps and DEX files in the `app` and `priv-app` folders located in `/system/`, libraries in the `lib` and `lib64` folders in `/system/`, any files in the `/system/vendor/` folder if that directory exists, and root certificates located in `/system/etc/security/cacerts/`. We can distinguish pre-installed apps from user-installed ones as the latter are stored in `/data/app/`. In order to reduce the scanning

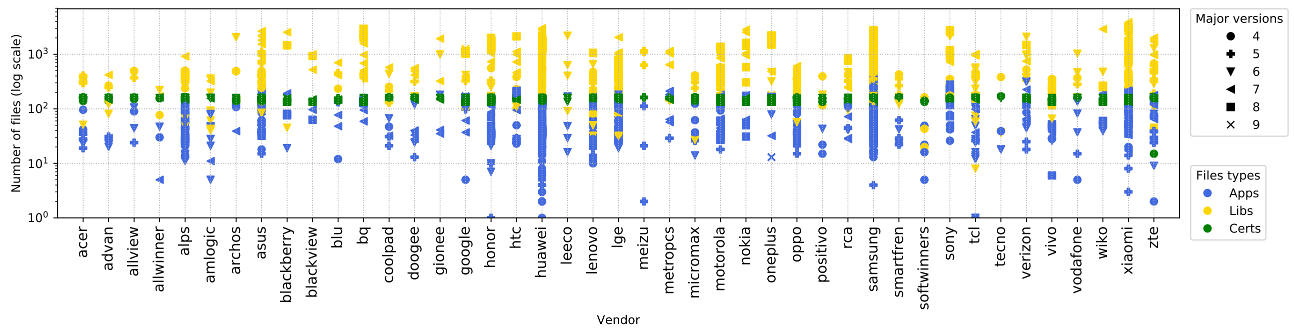

Figure 1: Number of files per vendor. We do not display the vendors for which we have less than 3 devices.

and upload time, Firmware Scanner first computes the MD5 hashes of the relevant files (*e.g.*, apps, libraries, and root certificates) and then sends the list of these hashes to our server. Only those missing in our dataset are uploaded over a Wi-Fi connection to avoid affecting the user's data plan.

Dataset: Thanks to 2,748 users who have organically installed Firmware Scanner, we obtained firmware versions for 1,742 unique device models[1] branded by 214 vendors[2] as summarized in Table I. Our dataset contains 424,584 unique files (based on their MD5 hash) as shown in Figure 1 for selected vendors. For each device we plot three dots, one for each type of file, while the shape indicates the major Android version that the device is running.[3] The number of pre-installed files varies greatly from one vendor to another. Although it is not surprising to see a large amount of native libraries due to hardware differences, some vendors embed hundreds of extra apps (*i.e.*, "`.apk`" files) compared to other manufacturers running the same Android version. For the rest of our study, we focus on 82,501 Android apps present in the dataset, leaving the analysis of root certificates and libraries for future work.

Our user-base is geographically distributed across 130 countries, yet 35% of our users are located in Europe, 29% in America (North and South), and 24% in Asia. Further, up to 25% and 20% of the total number of devices in our dataset belong to Samsung and Huawei ones, respectively. This is coherent with market statistics available online [35], [10]. While both manufacturers are Google-certified vendors, our dataset also contains low-end Android devices from manufacturers targeting markets such as Thailand, Indonesia, and India – many of these vendors are not Google-certified. Finally, to avoid introducing any bias in our results, we exclude 321 potentially rooted handsets from our study.[4]

B. *Lumen*

Lumen is an Android app available on Google Play that aims to promote mobile transparency and enable user control over their personal data and traffic. It leverages the Android VPN permission to intercept and analyze all Android traffic in user-space and in-situ, even if encrypted, without needing root permissions. By running locally on the user's device, Lumen is able to correlate traffic flows with system-level information and app activity. Lumen's architecture is publicly available and described in [86]. Lumen allows us to accurately determine which app is responsible for an observed PII leak from the vantage point of the user and as triggered by real user and device stimuli in the wild. Since all the analysis occurs on the device, only processed traffic metadata is exfiltrated from the device.

Dataset: For this study, we use anonymized traffic logs provided by over 20.4K users from 144 countries (according to Google Play Store statistics) coming from Android phones manufactured by 291 vendors. This includes 34,553,193 traffic flows from 139,665 unique apps (298,412 unique package name and version combinations). However, as Lumen does not collect app fingerprints or hashes of files, to find the overlap between the Lumen dataset and the pre-installed apps, we match records sharing the same package name, app version, and device vendor as the ones in the pre-installed apps dataset. While this method does not guarantee that the overlapping apps are exactly the same, it is safe to assume that phones that are not rooted are not shipped with different apps under the same package names and app versions. As a result, we have 1,055 unique pre-installed app/version/vendor combinations present in both datasets.

C. *Ethical Concerns*

Our study involves the collection of data from real users who organically installed Firmware Scanner or Lumen on their devices. Therefore, we follow the principles of informed consent [76] and we avoid the collection of any personal or sensitive data. We sought the approval of our institutional Ethics Board and Data Protection Officer (DPO) before starting the data collection. Both tools also provide extensive privacy policies in their Google Play profile. Below we discuss details specific to each tool.

Firmware Scanner: The app collects some metadata about the device to attribute observations to manufacturers (*e.g.*, its

[1]We use the MD5 hash of the IMEI to uniquely identify a user, and the build fingerprint reported by the vendor to uniquely identify a given device model. Note that two devices with the same fingerprint may be customized and therefore, have different apps pre-installed.

[2]We rely on the vendor string self-reported by the OS vendor, which could be bogus. For instance, Alps rebrands as "iPhone" some of its models, which, according to information available online, are Android-based replicas of iOS.

[3]We found that 5,244 of the apps do not have any activity, service, or receiver. These apps may potentially be used as providers of resources (*e.g.*, images, fonts) for other apps.

[4]We consider that a given device is rooted according to three signals. First, when Firmware Scanner has finished the upload of pre-installed binaries, the app asks the user whether the handset is rooted according to their own understanding (note that the user may choose not to answer the question). As a complement, we use the library RootBeer [63] to progammatically check if a device is rooted or not. If any of these sources indicates that the device is potentially rooted, we consider it as such. Finally, we discard devices where there is evidence of custom ROMs having been installed (*e.g.*, LineageOS). We discuss the limitations of this method in Section VI.

Vendor	Country	Certified partner	Device Fingerprints	Users	Files (med.)	Apps (med.)	Libs (med.)	DEX (med.)	Root certs (med.)	Files (total)	Apps (total)
Samsung	South Korea	Yes	441	924	868	136	556	83	150	260,187	29,466
Huawei	China	Yes	343	716	1,084	68	766	96	146	150,405	12,401
LGE	South Korea	Yes	74	154	675	84	385	89	150	58,273	3,596
Alps Mobile	China	No	65	136	632	56	385	46	148	29,288	2,883
Motorola	US/China	Yes	50	110	801	127	454	62	151	28,291	2,158
Total (214 vendors)	**—**	**22%**	**1,742**	**2,748**						**424,584**	**82,501**

Table I: General statistics for the top-5 vendors in our dataset.

model and fingerprint) along with some data about the pre-installed applications (extracted from the Package Manager), network operator (MNO), and user (the timezone, and the MCC and MNC codes from their SIM card, if available). We compute the MD5 hash of the device's IMEI to identify duplicates and updated firmware versions for a given device.

Lumen: Users are required to opt in twice before initiating traffic interception [76]. Lumen preserves its users' privacy by performing flow processing and analysis on the device, only sending anonymized flow metadata for research purposes. Lumen does not send back any unique identifiers, device fingerprints, or raw traffic captures. To further protect user's privacy, Lumen also ignores all flows generated by browser apps which may potentially deanonymize a user; and allows the user to disable traffic interception at any time.

III. Ecosystem Overview

The openness of Android OS has enabled a complex supply chain ecosystem formed by different stakeholders, be it manufacturers, MNOs, affiliated developers, and distributors. These actors can add proprietary apps and features to Android devices, seeking to provide a better user experience, add value to their products, or provide access to proprietary services. However, this could also be for (mutual) financial gain [32], [14]. This section provides an overview of pre-installed Android packages to uncover some of the gray areas that surround them, the large and diverse set of developers involved, the presence of third-party advertising and tracking libraries, and the role of each stakeholder.

A. Developer Ecosystem

We start our study by analyzing the organizations signing each pre-installed app. First, we cluster apps by the unique certificates used to sign them and then we rely on the information present in the `Issuer` field of the certificate to identify the organization [15]. Despite the fact that this is the most reliable signal to identify the organization signing the software, it is still noisy as a company can use multiple certificates, one for each organizational unit. More importantly, these are self-signed certificates, which significantly lowers the trust that can be put on them.

We were unable to identify the company behind several certificates (denoted as *Unknown company* in Table II) due to insufficient or dubious information in the certificate: *e.g.*, the `Issuer` field only contains the mentions `Company` and `department`. We have come across apps that are signed by 42 different *"Android Debug"* certificates on phones from 21 different brands. This reflects poor and potentially insecure development practices as Android's debug certificate is used to automatically sign apps in development environments, hence enabling other apps signed with that certificate to access its functionality without requesting any permission. Most app stores (including Google Play) will not accept the publication of an app signed with a Debug certificate [9]. Furthermore, we also found as many as 115 certificates that only mention *"Android"* in the `Issuer` field. A large part (43%) of those certificates are supposedly issued in the US, while others seem to have been issued in Taiwan (16%), China (13%), and Switzerland (13%). In the absence of a public list of official developer certificates, it is not possible to verify their authenticity or know their owner, as discussed in Section VI.

With this in mind, we extracted 1,200 unique certificates out of our dataset. Table II shows the 5 most present companies in the case of phone vendors (left) and other development companies (right). This analysis uncovered a vast landscape of third-party software in the long-tail, including large digital companies (*e.g.*, LinkedIn, Spotify, and TripAdvisor), as well as advertising and tracking services. This is the case of ironSource, an advertising firm signing pre-installed software [43] found in Asus, Wiko and other vendors, and TrueCaller, a service to block unwanted call or texts [57]. According to their website and also independent sources [40], [71], TrueCaller uses crowdsourced mechanisms to build a large dataset of phone numbers used for spam and also for advertising. Likewise, we have found 123 apps (by their MD5) signed by Facebook. These apps are found in 939 devices, 68% of which are Samsung's. We have also found apps signed by AccuWeather, a weather service previously found collecting personal data aggressively [87], Adups software, responsible for the Adups backdoor [46], and GMobi [36], a mobile-advertising company previously accused of dubious practices by the Wall Street Journal [14].

B. Third-party Services

As in the web, mobile app developers can embed in their pre-installed software third-party libraries (TPLs) provided by other companies, including libraries (SDKs) provided by ad networks, analytics services or social networks. In this section we use LibRadar++, an obfuscation-resilient tool to identify TPLs used in Android apps [91], on our dataset to examine their presence due to the potential privacy implications for users: when present in pre-installed apps, TPLs have the capacity to monitor user's activities longitudinally [90], [85]. We exclude well-known TPLs providing development support such as the Android support library. First, we classify the 11,665 unique TPLs identified by LibRadar++ according to the categories reported by Li *et al.* [83], AppBrain [51],

Company name	Number of certificates	Country	Certified partner?
Google	92	United States	N/A
Motorola	65	US/China	Yes
Asus	60	Taiwan	Yes
Samsung	38	South Korea	Yes
Huawei	29	China	Yes
Total (vendors)	**740**	—	—

Company name	Number of certificates	Country	Number of vendors
MediaTek	19	China	17
Aeon	12	China	3
Tinno Mobile	11	China	6
Verizon Wireless	10	United States	5
Unknown company	7	China	1
Total	**460**	—	**214**

Table II: **Left:** top-5 most frequent developers (as per the total number of apps signed by them), and **right:** for other companies.

Category	# libraries	# apps	# vendors	Example
Advertisement	164 (107)	11,935	164	Braze
Mobile analytics	100 (54)	6,935	158	Apptentive
Social networks	70 (20)	6,652	157	Twitter
All categories	**334**	**25,333**	**165**	—

Table III: Selected TPL categories present in pre-installed apps. In brackets, we report the number of TPLs when grouped by package name.

and PrivacyGrade [58]. We manually classified those TPLs that were not categorized by these datasets.

We focus on categories that could cause harm to the users' privacy, such as mobile analytics and targeted advertisement libraries. We find 334 TPLs in such categories, as summarized in Table III. We could identify advertising and tracking companies such as Smaato (specialized in geo-targeted ads [64]), GMobi, Appnext, ironSource, Crashlytics, and Flurry. Some of these third-party providers were also found shipping their own packages in Section III-A or are prominent actors across apps published in Google Play Store [85]. We found 806 apps embedding Facebook's Graph SDK which is distributed over 748 devices. The certificates of these apps suggests that 293 of them were signed by the device vendor, and 30 by an operator (only 98 are signed by Facebook itself). The presence of Facebook's SDKs in pre-installed apps could, in some cases, be explained by partnerships established by Facebook with Android vendors as the New York Times revealed [32].

We found other companies that provide mobile analytics and app monetization schemes such as Umeng, Fyber (previously Heyzap), and Kochava [85]. Moreover, we also found instances of advanced analytics companies in Asus handsets such as Appsee [17] and Estimote [28]. According to their website, Appsee is a TPL that allows developers to record and upload the users' screen [16], including touch events [84]. If, by itself, recording the user's screen does not constitute a privacy leak, recording and uploading this data could unintentionally leak private information such as account details. Estimote develops solutions for indoors geo-localization [28]. Estimote's SDK allows an app to react to nearby wireless beacons to, for example, send personalized push notifications to the user upon entering a shop

Finally, we find TPLs provided by companies specialized in the Chinese market [91] in 548 pre-installed apps. The most relevant ones are Tencent's SDK, AliPay (a payment service) and Baidu SDK [20] (for advertising and geolocation / geo-coding services), the last two possibly used as replacements for Google Pay and Maps in the Chinese market, respectively. Only one of the apps embedding these SDKs is signed by the actual third-party service provider, which indicates that their presence in pre-installed apps is likely due to the app developers' design decisions.

C. Public and Non-public Apps

We crawled the Google Play Store to identify how many of the pre-installed apps found by Firmware Scanner are available to the public. This analysis took place on the 19th of November, 2018 and we only used the package name of the pre-installed apps as a parameter. We found that only 9% of the package names in our dataset are indexed in the Google Play Store. For those indexed, few categories dominate the spectrum of pre-installed apps according to Google Play metadata, notably communication, entertainment, productivity, tools, and multimedia apps.

The low presence of pre-installed apps in the store suggests that this type of software might have escaped any scrutiny by the research community. In fact, we have found samples of pre-installed apps developed by prominent organizations that are not publicly available on Google Play. For instance, software developed and signed by Facebook (*e.g.*, `com.facebook.appmanager`), Amazon, and CleanMaster among others. Likewise, we found non-publicly available versions of popular web browsers (*e.g.*, UME Browser, Opera).

Looking at the last update information reported by Android's package manager for these apps, we found that pre-installed apps also present on Google Play are updated more often than the rest of pre-installed apps: 74% of the non-public apps do not seem to get updated and 41% of them remained unpatched for 5 years or more. If a vulnerability exists in one of these applications (see Section V), the user may stay at risk for as long as they keep using the device.

IV. Permission Analysis

Android implements a permissions model to control apps' access to sensitive data and system resources [56]. By default, apps are not allowed to perform any protected operation. Android permissions are not limited to those defined by AOSP: any app developer – including manufacturers – can define their own *custom permissions* to expose their functionality to other apps [26]. We leverage Androguard [4] to extract and study the permissions, both declared and requested, by pre-installed apps. We primarily focus on custom permissions as i) pre-installed services have privileged access to system resources, and ii) privileged pre-installed services may (involuntarily) expose critical services and data, even bypassing Android's official permission set.

A. Declared Custom Permissions

We identify 1,795 unique Android package names across 108 Android vendors defining 4,845 custom permissions. We exclude AOSP–defined permissions and those associated with Google's Cloud Messaging (GCM) [37]. The number of custom permissions declared per Android vendor varies across brands and models due to the actions of other stakeholders in the supply chain. We classify the organizations declaring custom permissions in 8 groups as shown in Table IV: hardware vendors, MNOs (*e.g.*, Verizon), third-party services (*e.g.*, Facebook), AV firms (*e.g.*, Avast), industry alliances (*e.g.*, GSMA), chipset manufacturers (*e.g.*, Qualcomm), and browsers (*e.g.*, Mozilla). We could not confidently identify the organizations responsible for 9% of all the custom permissions.[5]

As shown in Table IV, 63% of all declared custom permissions are defined by 31 handset vendors according to our classification. Most of them are associated with proprietary services such as Mobile Device Management (MDM) solutions for enterprise customers. Yet three vendors account for over 68% of the total custom permissions; namely Samsung (41%), Huawei (20%), and Sony (formerly Sony-Ericsson, 7%). Most of the custom permissions added by hardware vendors – along with chipset manufacturers, and MNOs – are exposed by Android core services, including the default browser `com.android.browser`. Unfortunately, as demonstrated in the MediaTek case [79], exposing such sensitive resources in critical services may potentially increase the attack surface if not implemented carefully.

An exhaustive analysis of custom permissions also suggests (and in some cases confirms) the presence of service integration and commercial partnerships between handset vendors, MNOs, analytics services (*e.g.*, Baidu, ironSource, and Digital Turbine), and online services (*e.g.*, Skype, LinkedIn, Spotify, CleanMaster, and Dropbox). We also found custom permissions associated with vulnerable modules (*e.g.*, MediaTek) and potentially harmful services (*e.g.*, Adups). We discuss cases of interest below.

VPN solutions: Android provides native support to third-party VPN clients. This feature is considered as highly sensitive as it gives any app requesting access the capacity to break Android's sandboxing and monitor users' traffic [68], [80]. The analysis of custom permissions reveals that Samsung and Meizu implement their own VPN service. It is unclear why these proprietary VPN implementations exist but it has been reported as problematic by VPN developers for whom their clients, designed for Android's default VPN service, do not run on such handsets [1], [86], [80]. A complete analysis of these VPN packages is left for future work.

Facebook: We found 6 different Facebook packages, three of them unavailable on Google Play, declaring 18 custom permissions as shown in Table V. These permissions have been found in 24 Android vendors, including Samsung, Asus, Xiaomi, HTC, Sony, and LG. According to users' complaints, two of these packages (`com.facebook.appmanager` and `com.facebook.system`) seem to automatically download other Facebook software such as Instagram in users' phones [69], [70]. We also found interactions between Facebook and MNOs such as Sprint.

Baidu: Baidu's geo-location permission is exposed by pre-installed apps, including core Android modules, in 7 different vendors, mainly Chinese ones. This permission seems to be associated with Baidu's geocoding API [19] and could allow app developers to circumvent Android's location permission.

Digital Turbine: We have identified 8 custom permissions in 8 vendors associated with Digital Turbine and its subsidiary LogiaGroup. Their privacy policy indicates that they collect personal data ranging from UIDs to traffic logs that could be shared with their business partners, which are undisclosed [27]. According to the SIM information of these devices, Digital Turbine modules are mainly found in North-American and Asian users. One package name, `com.dti.att` ("dti" stands for Digital Turbine Ignite), suggests the presence of a partnership with AT&T. A manual analysis confirms that this is the case. By inspecting their source-code, this package seems to implement comprehensive software management service. Installations and removals of apps by users are tracked and linked with PII, which only seem to be "masked" (*i.e.*, hashed) discretionally.

ironSource: The advertising company ironSource exposes custom permissions related to its AURA Enterprise Solutions [44]. We have identified several vendor-specific packages exposing custom ironSource permissions, in devices made by vendors such as Asus, Wiko, and HTC (the package name and certificate signatures suggest that those modules are possibly introduced with vendor's collaboration). According to ironSource's material [45], AURA has access to over 800 million users per month, while gaining access to advanced analytics services and to pre-load software on customers' devices. A superficial analysis of some of these packages (*e.g.*, `com.ironsource.appcloud.oobe.htc`, `com.ironsource.appcloud.oobe.asus`) reveals that they provide vendor-specific out-of-the-box-experience apps (OOBE) to customize a given user's device when the users open their device for the first time and empower user engagement [44], while also monitoring users' activities.

Other Advertising and Tracking Services: Discussing every custom permission introduced by third-party services individually would require an analysis beyond the scope of this paper. However, there are a couple of anecdotes of interest that we discuss next. One is the case of a pre-installed app signed by Vodafone (Greece) and present in a Samsung device that exposes a custom permission associated with Exus [31], a firm specialized in credit risk management and banking solutions. Another service declaring custom permissions in Samsung and LG handsets (likely sold by Verizon) is the analytics and user engagement company Synchronoss. Its privacy policy acknowledges the collection, processing and sharing of personal data [65].

Call protection services: We identify three external com-

[5]While Android's documentation recommends using reverse-domain-style naming for defining custom permissions to avoid collisions. [26], 269 of them – many of which are declared by a single hardware vendor – start with AOSP prefixes such as *android.permission.**. The absence of good development practices among developers complicated this classification, forcing us to follow a semi-manual process that involved analyzing multiple signals to identify their possible purpose and for attribution.

	Custom	Providers							
	permissions	Vendor	Third-party	MNO	Chipset	AV / Security	Ind. Alliance	Browser	Other
Total	4,845 (108)	3,760 (37)	192 (34)	195 (15)	67 (63)	46 (13)	29 (44)	7 (6)	549 (75)
Android Modules									
`android`	494 (21)	410 (9)	—	12 (2)	4 (13)	—	6 (7)	—	62 (17)
`com.android.systemui`	90 (15)	67 (11)	1 (2)	—	—	—	—	—	22 (8)
`com.android.settings`	87 (16)	63 (12)	—	1 (1)	—	—	—	—	23 (8)
`com.android.phone`	84 (14)	56 (9)	—	5 (2)	3 (5)	—	—	—	20 (10)
`com.android.mms`	59 (11)	35 (10)	—	1 (2)	—	—	1 (1)	—	22 (8)
`com.android.contacts`	40 (7)	32 (3)	—	—	—	—	—	—	8 (5)
`com.android.email`	33 (10)	18 (4)	—	—	—	—	—	—	15 (17)

Table IV: Summary of custom permissions per provider category and their presence in selected sensitive Android core modules. The value in brackets reports the number of Android vendors in which custom permissions were found.

Package	Public	# Vendors	# Permissions
`com.facebook.system`	No	18	2
`com.facebook.appmanager`	No	15	4
`com.facebook.katana` (Facebook)	Yes	14	8
`com.facebook.orca` (Messenger)	Yes	5	5
`com.facebook.lite` (FB Lite)	Yes	1	1
`com.facebook.pages.app`	No	1	4
Total	3	24	18

Table V: Facebook packages on pre-installed handsets.

panies providing services for blocking undesired and spam phone calls and text messages: Hiya [38], TrueCaller [57], and PrivacyStar [59]. Hiya's solution seems to be integrated by T-Mobile (US), Orange (Spain), and AT&T (US) in their subsidized Samsung and LG handsets according to the package signatures. Hiya and TrueCaller's privacy policies indicate that they collect personal data from end users, including contacts stored in the device, UIDs, and personal information [39]. [6] PrivacyStar's privacy policy, instead, claims that any information collected from a given user's contacts is "NOT exported outside the App for any purpose" [60].

B. Used Permissions

The use of permissions by pre-installed Android apps follows a power-law distribution: 4,736 of the package names request at least one permission and 55 apps request more that 100. The fact that pre-installed apps request many permissions to deliver their service does not necessarily imply a breach of privacy for the user. However, we identified a significant number of potentially over-privileged vendor- and MNO-specific packages with suspicious activities such as `com.jrdcom.Elabel` – a package signed by TCLMobile requesting 145 permissions and labeled as malicious by Hybrid Analysis (a free online malware analysis service) – and `com.cube26.coolstore` (144 permissions). Likewise, the calculator app found on a Xiaomi Mi 4c requests user's location and the phone state, which gives it access to UIDs such as the IMEI. We discuss more instances of over-privileged apps in Section V-C.

Dangerous Android permissions. The median pre-installed Android app requests three dangerous AOSP permissions. When we look at the set of permissions requested by a given

[6]Note: the information rendered in their privacy policy differs when crawled from a machine in the EU or the US. As of January 2019, none of these companies mention the new European GDPR directive in their privacy policies.

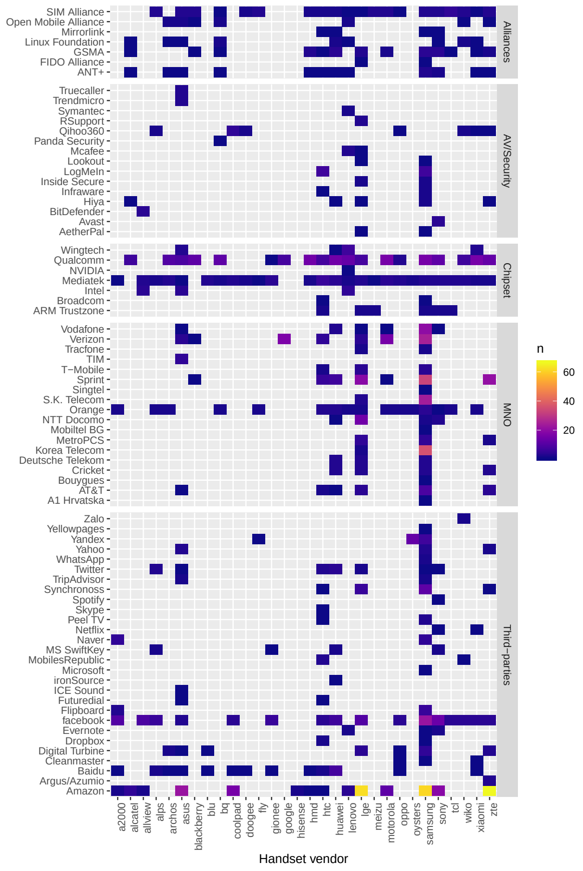

Figure 2: Permissions defined by AV firms, MNOs, chipset vendors and third parties, requested by pre-installed apps.

app (by its package name) across vendors, we can notice significant differences. We investigate such variations in a subset of 150 package names present at least in 20 different vendors. This list contains mainly core Android services as well as apps signed by independent companies (*e.g.*, Adups) and chipset manufacturers (*e.g.*, Qualcomm).

Then, we group together all the permissions requested by a given package name across all device models for each brand. As in the case of exposed custom permissions, we can see a tendency towards over-privileging these modules in specific vendors. For instance, the number of permissions requested by the core `android` module can range from 9 permissions in a Google-branded Android device to over 100 in most Samsung devices. Likewise, while the median

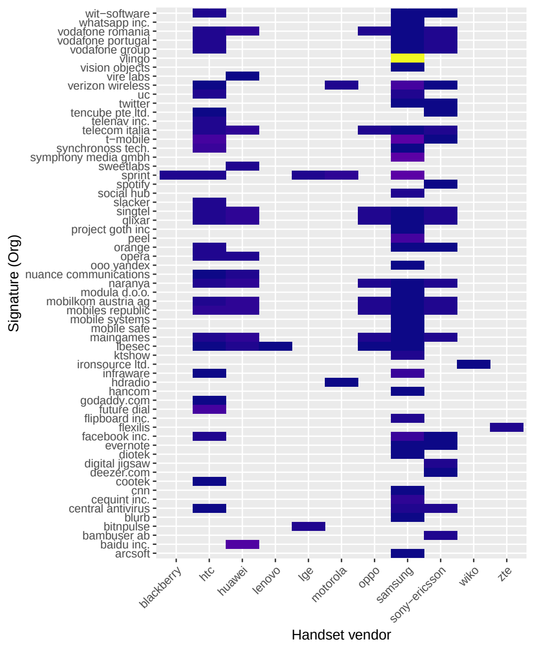

Figure 3: Apps accessing vendors' custom permissions.

`com.android.contacts` service requests 35 permissions, this number goes over 100 for Samsung, Huawei, Advan, and LG devices.

Custom permissions. 2,910 pre-installed apps request at least one custom permission. The heatmap in Figure 2 shows the number of custom permissions requested by pre-installed packages in a hand-picked set of popular Android manufacturers (x-axis). As we can see, the use of custom permissions also varies across vendors, with those associated with large third-party analytics and tracking services (*e.g.*, Facebook), MNOs (*e.g.*, Vodafone), and AV/Security services (*e.g.*, Hiya) being the most requested ones.

This analysis uncovers possible partnerships beyond those revealed in the previous sections. We identify vendor-signed services accessing ironSource's, Hiya's, and AccuWeather's permissions. This state of affairs potentially allows third-party services and developers to gain access to protected permissions requested by other pre-installed packages signed with the same signature. Further, we found Sprint-signed packages resembling that of Facebook and Facebook's Messenger APKs (`com.facebook.orca.vpl` and `com.facebook.katana.vpl`) requesting Flurry-related permissions (a third-party tracking service owned by Verizon).

Commercial relationships between third-party services and vendors appear to be bi-directional as shown in Figure 3. This figure shows evidence of 87 apps accessing vendor permissions, including packages signed by Facebook, ironSource, Hiya, Digital Turbine, Amazon, Verizon, Spotify, various browser, and MNOs – grouped by developer signature for clarity purposes. As the heatmap indicates, Samsung, HTC and Sony are the vendors enabling most of the custom permissions requested by over-the-top apps. We found instances of apps listed on the Play Store also requesting such permissions. Unfortunately, custom permissions are not shown to users when shopping for mobile apps in the store – therefore they

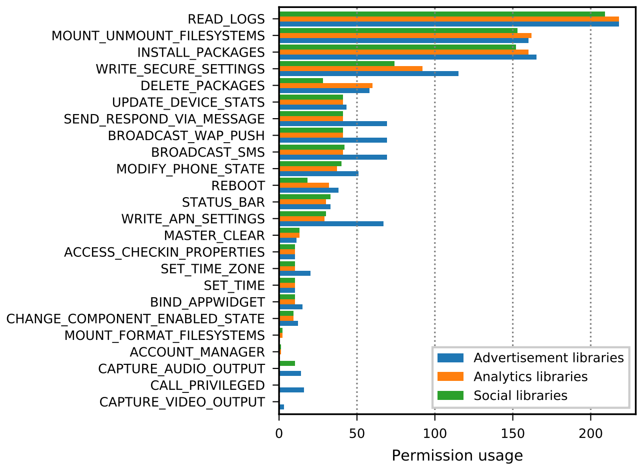

Figure 4: System permissions requested by pre-installed apps embedding TPLs.

are apparently requested without consent – allowing them to cause serious damage to users' privacy when misused by apps.

C. Permission Usage by TPLs

We look at the permissions used by apps embedding at least one TPL. We study the access to permissions with a protection level of either `signature` or `signature|privileged` as they can only be granted to system apps [50] or those signed with a system signature. The presence of TPLs in pre-installed apps requesting access to a signature or dangerous permission can, therefore, give it access to very sensitive resources without user awareness and consent. Figure 4 shows the distribution of signature permissions requested across apps embedding TPLs. We find that the most used permissions – `READ_LOGS` – allow the app (and thus the TPLs within it) to read system logs, mount and unmount filesystems, or install packages. We find no significant differences between the three types of TPLs of interest. For completeness, we also find that 94 apps embedding TPLs of interest request custom permissions as well. Interestingly, 53% of the 88 custom permissions used by these apps are defined by Samsung.

D. Component Exposing

Custom permissions are not the only mechanism available for app developers to expose (or access) features and components to (or from) other apps. Android apps can also interact with each other using *intents*, a high-level communication abstraction [42]. An app may expose its component(s) to external apps by declaring `android:exported=true` in the manifest without protecting the component with any additional measure, or by adding one or more intent-filters to its declaration in the manifest; exposing it to a type of attack known in the literature as a confused deputy attack [79]. If the `exported` attribute is used, it can be protected by adding a permission to the component, be it a custom permission or an AOSP one, through checking the caller app's permissions programmatically in the component's Java class.

We sought to identify potentially careless development practices that may lead to components getting exposed without any additional protection. Exporting components can lead to:

i) harmful or malicious apps launching an exposed activity, tricking users into believing that they are interacting with the benign one; ii) initiating and binding to unprotected services; and iii) malicious apps gaining access to sensitive data or the ability to modify the app's internal state.

We found 6,849 pre-installed apps that potentially expose at least one activity in devices from 166 vendors and signed by 261 developer signatures with `exported=true`. For services, 4,591 apps (present in 157 vendors) signed by 183 developers including manufacturers, potentially exposed one or more of their services to external apps. The top-10 vendors in our dataset account for over 70% of the potentially exposed activities and services. Other relevant examples include an app that potentially exposes several activities related to system configurations (device administration, networking, *etc.*), hence allowing a malicious developer could access or even tamper a users' device settings. The core package `com.android.mms` found in customized firmware versions across several vendors also expose services to read WAP messages to other apps. We also found 8 different instances of a third-party app, found in handsets built by two large Android manufacturers, whose intended purpose is to provide remote technical support to customers. This particular service provides remote administration to MNOs, including the ability to record audio and video, browse files, access system settings, and upload/download files. The key service to do so is exposed and can be misused by other apps.

We leave the detailed study of apps vulnerable to confused deputy attacks and the study of the access to these resources by apps publicly available on Google Play for future work.

V. Behavioral Analysis

We analyze the apps in our dataset to identify potentially harmful and unwanted behaviors. To do this, we leverage both static and dynamic analysis tools to elicit behavior and characterize purpose and means. This section describes our analysis pipeline and evidence of potentially harmful and privacy-intrusive pre-installed packages.

A. Static Analysis

We triage all apps to determine the presence of potentially harmful behaviors. This step allows us to obtain a high-level overview of behaviors across the dataset and also provides us with the basis to score apps and flag those potentially more interesting. This step is critical since we could only afford to manually inspect a limited subset of all available apps.

Toolkit. Our analysis pipeline integrates various static analysis tools to elicit behavior in Android apps, including Androwarn [12], FlowDroid [74], and Amandroid [92], as well as a number of custom scripts based on the Apktool [13] and Androguard [4] frameworks. In this stage we do not use dynamic analysis tools, which prevents us from identifying hidden behaviors that rely on dynamic code uploading (DEX loading) or reflection. This means that our results present a lower-bound estimation of all the possible potentially harmful behaviors. We search for apps using DEX loading and reflection to identify targets that deserve manual inspection.

Dataset. Because of scalability limitations – our dataset comprises 82,501 APK files with 6,496 unique package names – we randomly select one APK file for each package name and analyze the resulting set of apps, obtaining an analysis report for 48% of them. The majority of the remaining packages could not be analyzed due to the absence of a `classes.dex` for odexed files. Even though in some cases we had the corresponding `.odex` file, we generally could not deodex it since the device's Android framework file was needed to complete this step but Firmware Scanner did not collect it. Moreover, we could not analyze a small subset of apps due to the limitations of our tools, including errors generated during analysis, file size limitations, or analysis tools becoming unresponsive after hours of processing. Instead, we focused our analysis on the subset of apps for which we could generate reports.

Results. We processed the analysis reports and identified the presence of the 36 potentially privacy intrusive behaviors or potentially harmful behaviors listed in Table VI. The results suggest that a significant fraction of the analyzed apps could access and disseminate both user and device identifiers, user's location, and device current configuration. According to our flow analysis, these results give the impression that personal data collection and dissemination (regardless of the purpose or consent) is not only pervasive but also comes pre-installed. Other a priori concerning behaviors include the possible dissemination of contacts and SMS contents (164 and 74 apps, respectively), sending SMS (29 apps), and making phone calls (339 apps). Even though there are perfectly legitimate use cases for these behaviors, they are also prevalent in harmful and potentially unwanted software. The distribution of the number of potentially harmful behaviors per app follows a power-law distribution. Around 25% of the analyzed apps present at least 5 of these behaviors, with almost 1% of the apps showing 20 or more. The bulk of the distribution relates to the collection of telephony and network identifiers, interaction with the package manager, and logging activities. This provides a glimpse of how pervasive user and device fingerprinting is nowadays.

B. Traffic Analysis

While static analysis can be helpful to determine a lower bound of what an app is capable of, relying on this technique alone gives an incomplete picture of the real-world behavior of an app. This might be due to code paths that are not available at the time of analysis, including those that are within statically- and dynamically-linked libraries that are not provided with apps, behaviors determined by server-side logic (e.g., due to real-time ad-bidding), or code that is loaded at runtime using Java's reflection APIs. This limitation of static approaches is generally addressed by complementing static analysis with dynamic analysis tools. However, due to various limitations (including missing hardware features and software components) it was unfeasible for us to run all the pre-installed apps in our dataset in an analysis sandbox. Instead, we decided to use the crowd-sourced Lumen mobile traffic dataset to find evidence of dissemination of personal data from the pre-installed apps by examining packages that exist in both datasets.

Accessed PII type / behaviors		Apps (#)	Apps (%)
Telephony identifiers	IMEI	687	21.8
	IMSI	379	12
	Phone number	303	9.6
	MCC	552	17.5
	MNC	552	17.5
	Operator name	315	10
	SIM Serial number	181	5.7
	SIM State	383	12.1
	Current country	194	6.2
	SIM country	196	6.2
	Voicemail number	29	0.9
Device settings	Software version	25	0.8
	Phone state	265	8.4
	Installed apps	1,286	40.8
	Phone type	375	11.9
	Logs	2,568	81.4
Location	GPS	54	1.7
	Cell location	158	5
	CID	162	5.1
	LAC	137	4.3
Network interfaces	Wi-Fi configuration	9	0.3
	Current network	1,373	43.5
	Data plan	699	22.2
	Connection state	71	2.3
	Network type	345	10.9
Personal data	Contacts	164	11
	SMS	73	2.31
Phone service abuse	SMS sending	29	0.92
	SMS interception	0	0
	Disabling SMS notif.	0	0
	Phone calls	339	10.7
Audio/video interception	Audio recording	74	2.4
	Video capture	21	0.7
Arbitrary code execution	Native code	775	24.6
	Linux commands	563	17.9
Remote conn.	Remote connection	89	2.8

Table VI: Volume of apps accessing / reading PII or showing potentially harmful behaviors. The percentage is referred to the subset of triaged packages ($N = 3,154$).

Results. Of the 3,118 pre-installed apps with Internet access permissions, 1,055 have at least one flow in the Lumen dataset. At this point, our analysis of these apps focused on two main aspects: uncovering the ecosystem of organizations who own the domains that these apps connect to, and analyzing the types of private information they could disseminate from user devices. To understand the ecosystem of data collection by pre-installed apps, we studied where the data that is collected by these apps makes its first stop. We use the Fully-Qualified Domain Names (FQDN) of the servers that are contacted and use the web crawling and text mining techniques described in our previous work [85] to determine the parent organization who own these domains.

The Big Players. Table VII shows the parent organizations who own the most popular domains contacted by pre-installed apps in the Lumen dataset. Of the 54,614 domains contacted by apps, 7,629 belong to well-known Advertising and Tracking Services (ATS) [85]. These services are represented by organizations like Alphabet, Facebook, Verizon (now owner of Yahoo!, AOL, and Flurry), Twitter (MoPub's parent organization), AppsFlyer, comScore, and others. As expected,

Organization	# of apps	# of domains
Alphabet	566	17052
Facebook	322	3325
Amazon	201	991
Verizon Communications	171	320
Twitter	137	101
Microsoft	136	408
Adobe	116	302
AppsFlyer	98	10
comScore	86	8
AccuWeather	86	15
MoatInc.	79	20
Appnexus	79	35
Baidu	72	69
Criteo	70	62
PerfectPrivacy	68	28
Other ATS	221	362

Table VII: Top 15 parent ATS organizations by number of apps connecting to all their associated domains.

Alphabet, the entity that owns and maintains the Android platform and many of the largest advertising and tracking services (ATS) in the mobile ecosystem [85], also owns most of the domains to which pre-installed apps connect to. Moreover, vendors who ship their devices with the Google Play Store have to go through Google's certification program which, in part, entails pre-loading Google's services. Among these services is Google's own `com.google.backuptransport` package, which sends a variety of information about the user and the device on which it runs to Google's servers.

Traffic analysis also confirms that Facebook and Twitter services come pre-installed on many phones and are integrated into various apps. Many devices also pre-install weather apps like AccuWeather and The Weather Channel. As reported by previous research efforts, these weather providers also gather information about the devices and their users [87], [85].

C. Manual Analysis: Relevant Cases

We used the output provided by our static and dynamic analysis pipeline to score apps and thus flag a reduced subset of packages to inspect manually. Our goal here was to confidently identify potentially harmful and unwanted behavior in pre-installed apps. Other apps were added to this set based on the results of our third-party library and permission analysis performed in Sections III and IV, respectively. We manually analyzed 158 apps using standard tools that include DEX disassemblers (baksmali), dex-to-java decompilers (jadx, dex2jar), resource analysis tools (Apktool), instrumentation tools (Frida), and reverse engineering frameworks (radare2 and IDA Pro) for native code analysis. Our main findings can be loosely grouped into three large categories: 1) known malware; 2) potential personal data access and dissemination; and 3) potentially harmful apps. Table VIII provides some examples of the type of behaviors that we found.

Known Malware. We came across various isolated instances of known-malware in the system partition, mostly in low-end devices but also in some high-end phones. We identified variants of well-known Android malware families that have been prevalent in the last few years, including Triada, Rootnik, SnowFox, Xinyin, Ztorg, Iop, and dubious software developed by GMobi. We used VirusTotal to label these samples. According to existing AV reports, the range of behaviors that such

Family	Potential Behavior and Prevalence
Known Malware	
Triada	Disseminates PII and other sensitive data (SMS, call logs, contact data, stored pictures and videos). Downloads additional stages. Roots the device to install additional apps.
Rootnik [62]	Gains root access to the device. Leaks PII and installs additional apps. Uses anti-analysis and anti-debugging techniques.
GMobi [11], [67]	Gmobi Trade Service. Leaks PII, including device serial number and MAC address, geolocation, installed packages and emails. Receives commands from servers to (1) send an SMS to a given number; (2) download and install apps; (3) visit a link; or (4) display a pop-up. It has been identified in low-end devices.
Potentially Dangerous Apps	
Rooting app	Exposes an unprotected receiver that roots the device upon receiving a telephony secret code (via intent or dialing `*#*#9527#*#*`).
Blocker	If the device does not contain a signed file in a particular location, it loads and enforces 2 blacklists: one containing 103 packages associated with benchmarking apps, and another with 56 web domains related to phone reviews.
Potential Personal Data Access and Dissemination	
TrueCaller	Sends PII to its own servers and embedded third-party ATSes such as AppsFlyer, Twitter-owned MoPub, Crashlytics, inMobi, Facebook, and others. Uploads phone call data to at least one of its own domains.
MetroName ID	Disseminates PII to its own servers and also to third-party services like Piano, a media audience and engagement analytics service that tracks user's installation of news apps and other partners including those made by CNBC, Bloomberg, TechCrunch, and The Economist, among others, the presence of which it reports to its own domains.
Adups [47]	FOTA app. Collects and shares private and PII with their own servers and those of embedded third-party ATS domains, including Advmob and Nexage. Found worldwide in 55 brands.
Stats/Meteor	Redstone's FOTA service. Uses dynamic code uploading and reflection to deploy components located in 2 encrypted DEX files. Disseminates around 50 data items that fully characterize the hardware, the telephony service, the network, geolocation, and installed packages. Performs behavioral and performance profiling, including counts of SMS/MMS, calls logs, bytes sent and transmitted, and usage stats and performance counters on a package-basis. Silently installs packages on the device and reports what packages are installed / removed by the user.

Table VIII: Examples of relevant cases and their potential behaviors found after manual analysis of a subset of apps. When referring to personal data dissemination, the term PII encompasses items enumerated in Table VI.

samples exhibit encompass banking fraud, sending SMS to premium numbers or subscribing to services, silently installing additional apps, visiting links, and showing ads, among others. While our method does not allow us to distinguish whether potentially malicious apps are indeed pre-installed or took advantage of system vulnerabilities to install themselves in the system partition, it is important to highlight that the presence of pre-installed malware in Android devices has been previously reported by various sources [66], [6], [67]. Some of the found samples use Command and Control (C2) servers still in operation at the time of this writing.

Personal Data Access and Potential Dissemination. Nearly all apps which we identified as able to access PII, appear to disseminate it to third-party servers. We also observed instances of apps with capabilities to perform hardware and network fingerprinting, often collected under the term "device capability," and even analytics services that track the installation and removal of apps (notably news apps, such as those made by CNBC, Bloomberg, TechCrunch, and The Economist, among others). More intrusive behaviors include apps able to collect and send email and phone call metadata. The most extreme case we analyzed is a data collection service contained in a FOTA service associated with Redstone Sunshine Technology Co., Ltd. [61], an OTA provider that "supports 550 million phone users and IoT partners in 40 countries" [22]. This app includes a service that can collect and disseminate dozens of data items, including both user and device identifiers, behavioral information (counts of SMS and calls sent and received, and statistics about network flows) and usage statistics and performance information per installed package. Overall, this software seems to implement an analytics program that admits several monetization strategies, from optimized ad targeting to providing performance feedback to both developers and manufacturers. We emphasize that the data collected is not only remarkably extensive and multi-dimensional, but also very far away from being anonymous as it is linked to both user and device IDs.

Potentially dangerous apps. We found 612 pre-installed apps that potentially implement engineering- or factory-mode functions according to their package and app names. Such functions include relatively harmless tasks, such as hardware tests, but also potentially dangerous functions such as the ability to root the device. We found instances of such apps in which the rooting function was unprotected in their manifest (*i.e.*, the component was available for every other app to use). We also identified well-known vulnerable engineering mode apps such like MTKLogger [82]. Such apps expose unprotected components that can be misused by other apps co-located in the device. Other examples include a well known manufacturer's service, which under certain conditions blacklists connections to a pre-defined list of 56 web domains (mobile device review and benchmarking websites, mostly) and disables any installed package that matches one of a list of 103 benchmarking apps.

VI. Study Limitations

Completeness and coverage. Our dataset is not complete in terms of Android vendors and models, even though we cover those with a larger market share, both in the high- and low-end parts of the spectrum. Our data collection process is also best-effort. The lack of background knowledge and documentation required performing a detailed case-by-case study and a significant amount of manual inspection. In terms of analyzed apps, determining the coverage of our study is difficult since we do not know the total number of pre-installed apps in all shipped handsets.

Attribution. There is currently no reliable way to accurately find the legitimate developer of a given pre-installed app by its self-signed signature. We have found instances of certificates with just a country code in the `Issuer` field, and others with strings suggesting major vendors (*e.g.*, Google) signed the app, where the apps certainly were not signed by them. The same applies to package and permission names, many of which are opaque and not named following best-practices. Likewise, the lack of documentation regarding custom permissions prevented us from automatizing our analysis. Moreover, a deeper study of this issue would require checking whether those permissions are granted in runtime, tracing the code to fully identify their purpose, and finding whether they are actually used by other apps in the wild, and at scale.

Package Manager. We do not collect the `packages.xml` file from our users' devices as it contains information about all installed packages, and not just pre-installed ones. We consider that collecting this file would be invasive. This, however, limits our ability to see if user-installed apps are using services exposed by pre-installed apps via intents or custom permissions. We tried to compensate for that with a manual search for public apps that use pre-installed custom permissions, as discussed in Section IV-D.

Behavioral coverage. Our study mainly relies on static analysis of the samples harvested through Firmware Scanner, and we only applied dynamic analysis to a selected subset of 1,055 packages. This prevents us from eliciting behaviors that are only available at runtime because of the use of code loading and reflection, and also code downloading from third-party servers. Despite this, our analysis pipeline served to identify a considerable amount of potentially harmful behaviors. A deeper and broader analysis would possibly uncover more cases.

Identifying rooted devices. There is no sure way of knowing whether a device is rooted or not. While our conservative approach limits the number of false negatives, we have found occurrences of devices with well-known custom ROMs that were not flagged as rooted by RootBeer. Moreover, we have found some apps that allow a third party to root the device on-the-fly to, for example, install new apps on the system partition as discussed in Section V-C. Some of these apps can then unroot the phone to avoid detection. Under the presence of such an app on a device, we cannot know for sure if a given package – particularly a potentially malicious app – was pre-installed by an actor in the supply chain, or was installed afterwards.

VII. Related work

Android images customization. Previous work has been focused on studying modifications made to AOSP images, whether by adding root certificates [89], customizing the default apps [73], or the OS itself [95]. In [72], Aafer *et al.* introduced a new class of vulnerability caused by the firmware customization process. If an app is removed but a reference to it remains in the OS, a malicious app could potentially impersonate it which could lead to privacy and security issues. While these studies have focused on Android images as a whole rather than pre-installed apps, they all show the complexity of the Android ecosystem and underline the lack of control over the supply chain.

Android permissions. Previous studies on Android permissions have mainly leveraged static analysis techniques to infer the role of a given permission [75], [78]. These studies, however, do not cover newer versions of Android [94], or custom permissions. In [81], Jiang *et al.* demonstrated how custom permissions are used to expose and protect services. Our work complements this study by showing how device makers and third parties alike declare and use custom permissions, and make the first step towards a complete and in-depth analysis of the whole custom permissions' landscape.

Vulnerabilities in pre-installed apps. A recent paper by Wu *et al.* [93] also used crowdsourcing mechanisms to detect apps that listen to a given TCP or UDP port and analyze the vulnerabilities that are caused by this practice. While their study is not limited to user-installed apps, they show evidence of pre-installed apps exhibiting this behavior.

VIII. Discussion and Conclusions

This paper studied, at scale, the vast and unexplored ecosystem of pre-installed Android software and its potential impact on consumers. This study has made clear that, thanks in large part to the open-source nature of the Android platform and the complexity of its supply chain, organizations of various kinds and sizes have the ability to embed their software in custom Android firmware versions. As we demonstrated in this paper, this situation has become a peril to users' privacy and even security due to an abuse of privilege or as a result of poor software engineering practices that introduce vulnerabilities and dangerous backdoors.

The Supply Chain. The myriad of actors involved in the development of pre-installed software and the supply chain range from hardware manufacturers to MNOs and third-party advertising and tracking services. These actors have privileged access to system resources through their presence in pre-installed apps but also as third-party libraries embedded in them. Potential partnerships and deals – made behind closed doors between stakeholders – may have made user data a commodity before users purchase their devices or decide to install software of their own.

Attribution. Unfortunately, due to a lack of central authority or trust system to allow verification and attribution of the self-signed certificates that are used to sign apps, and due to a lack of any mechanism to identify the purpose and legitimacy of many of these apps and custom permissions, it is difficult to attribute unwanted and harmful app behaviors to the party or parties responsible. This has broader negative implications for accountability and liability in this ecosystem as a whole.

The Role of Users and Informed Consent. In the meantime regular Android users are, by and large, unaware of the presence of most of the software that comes pre-installed on their Android devices and their associated privacy risks. Users are clueless about the various data-sharing relationships and partnerships that exist between companies that have a hand in deciding what comes pre-installed on their phones. Users' activities, personal data, and habits may be constantly monitored by stakeholders that many users may have never heard of, let alone consented to collect their data. We have demonstrated instances of devices being backdoored by companies with the ability to root and remotely control devices without user awareness, and install apps through targeted monetization and user-acquisition campaigns. Even if users decide to stop or delete some of these apps, they will not be able to do so since many of them are core Android services and others cannot be permanently removed by the user without root privileges. It is unclear if the users have actually consented to these practices, or if they were informed about them before using the devices (*i.e.*, on first boot) in the first place. To clarify this, we acquired 6 popular brand-new Android devices from vendors including Nokia, Sony, LG, and Huawei from a large Spanish retailer. When booting them, 3 devices did not present a privacy policy at all, only the Android terms of service. The rest rendered a privacy policy that only mentions that they collect data about the user, including PII such as the IMEI for added value services. Note that users have no choice but to accept Android's terms of service, as well as the manufacturer's one if presented to the user. Otherwise Android will simply stop booting, which will effectively make the device unusable.

Consumer Protection Regulations. While some jurisdictions have very few regulations governing online tracking and data collection, there have been a number of movements to regulate and control these practices, such as the GDPR in the EU [29], and California's CCPA [21] in the US. While these efforts are certainly helpful in regulating the rampant invasion of users' privacy in the mobile world, they have a long way to go. Most mobile devices still lack a clear and meaningful mechanism to obtain informed consent, which is a potential violation of the GDPR. In fact, it is possible that many of the ATSes that come pre-installed on Android devices may not be COPPA-compliant [88] – a US federal rule to protect minors from unlawful online tracking [23] –, despite the fact that many minors in the US use mobile devices with pre-installed software that engage in data collection. This indicates that even in jurisdictions with strict privacy and consumer protection laws, there still remains a large gap between what is done in practice and the enforcement capabilities of the agencies appointed to uphold the law.

Recommendations. To address the issues mentioned above and to make the ecosystem more transparent we propose a number of recommendations. which are made under the assumption that stakeholders are willing to self-regulate and to enhance the status quo. We are aware that some of these suggestions may inevitably not align with corporate interests of every organizations in the supply chain, and that an independent third party may be needed to audit the process. Google might be a prime candidate for it given its capacity for licensing vendors and its certification programs. Alternatively, in absence of self-regulation, governments and regulatory bodies could step in and enact regulations and execute enforcement actions that wrest back some of the control from the various actors in the supply chain. We also propose a number of actions that would help independent investigators to detect deceptive and potentially harmful behaviors.

• *Attribution and accountability:* To combat the difficulty in attribution and the resulting lack of accountability, we propose the introduction and use of certificates that are signed by globally-trusted certificate authorities. Alternatively, it may be possible to build a certificate transparency repository dedicated to providing details and attribution for self-signed certificates used to sign various Android apps, including pre-installed ones.

• *Accessible documentation and consent forms:* Similar to the manner in which open-source components of Android require any modified version of the code to be made publicly-available, Android devices can be required to document the specific set of apps that have pre-installed, along with their purpose and the entity responsible for each piece of software, in a manner that is accessible and understandable to the users. This will ensure that at least a reference point exists for users (and regulators) to find accurate information about pre-installed apps and their practices. Moreover, the results of our small-scale survey of consent forms of some Android vendors leaves a lot to be desired from a transparency perspective: users are not clearly informed about third-party software that is installed on their devices, including embedded third-party tracking and advertising services, the types of data they collect from them by default, and the partnerships that allow personal data to be shared over the Internet. This necessitates a new form of privacy policy suitable for pre-installed apps to be defined (and enforced) to ensure that such practices are at least communicated to the user in a clear and accessible way. This should be accompanied by mechanisms to enable users to make informed decisions about how or whether to use such devices without having to root them.

Final Remarks. Despite a full year of efforts, we were only able to scratch the surface of a much larger problem. This work is therefore exploratory, and we hope it will bring more attention to the Android supply chain ecosystem and its impact on users' privacy and security. We have discussed our results with Google which gave us useful feedback. Our work was also the basis of a report produced by the Spanish Data Protection Agency (AEPD) [3]. We will also improve the capabilities and features of both Firmware Scanner and Lumen to address some of the aforementioned limitations and develop methods to perform dynamic analysis of pre-installed software. Given the scale of the ecosystem and the need for manual inspections, we will gradually make our dataset (which keeps growing at the time of this writing) available to the research community and regulators to aid in future investigations and to encourage more research in this area.

Acknowledgments

We are deeply grateful to our Firmware Scanner users for enabling this study, and ElevenPaths for their initial support in this project. We thank the anonymous reviewers for their helpful feedback. This project is partially funded by the US National Science Foundation (grant CNS-1564329), the European Union's Horizon 2020 Innovation Action program (grant Agreement No. 786741, SMOOTH Project), the Spanish Ministry of Science, Innovation and Universities (grants DiscoEdge TIN2017-88749-R and SMOG-DEV TIN2016-79095-C2-2-R), and the Comunidad de Madrid (grant EdgeData-CM P2018/TCS-4499). Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors and do not reflect the views of the funding bodies.

References

[1] AdGuard - Meizu Incompatibilities. https://github.com/AdguardTeam/AdguardForAndroid/issues/800. [Online; accessed 31-March-2019].

[2] Amazon suspends sales of Blu phones for including preloaded spyware, again. https://www.theverge.com/2017/7/31/16072786/amazon-blu-suspended-android-spyware-user-data-theft. [Online; accessed 31-March-2019].

[3] Análisis del software preinstalado en dispositivos Android y riesgos para la privacidad de los usuarios. https://www.aepd.es/prensa/2019-03-18.html. [Online; accessed 31-March-2019].

[4] Androguard. https://github.com/androguard/androguard/. [Online; accessed 31-March-2019].

[5] Android — Certified. https://www.android.com/certified/. [Online; accessed 31-March-2019].

[6] Android Adware and Ransomware Found Preinstalled on High-End Smartphones. https://www.bleepingcomputer.com/news/security/android-adware-and-ransomware-found-preinstalled-on-high-end-smartphones/. [Online; accessed 31-March-2019].

[7] Android Certified Partners. https://www.android.com/certified/partners/. [Online; accessed 31-March-2019].

[8] Android Compatibility Program Overview. https://source.android.com/compatibility/overview. [Online; accessed 31-March-2019].

[9] Android Developer Documentation. https://developer.android.com/. [Online; accessed 31-March-2019].

[10] Android Trackers. https://fiksu.com/resources/android_trackers/. [Online; accessed 31-March-2019].

[11] Android.Gmobi.1. https://vms.drweb.com/virus/?_is=1&i=7999623&lng=en. [Online; accessed 31-March-2019].

[12] Androwarn–Yet another static code analyzer for malicious Android applications. https://github.com/maaaaz/androwarn. [Online; accessed 31-March-2019].

[13] Apktool–A tool for reverse engineering Android apk files. https://ibotpeaches.github.io/Apktool/. [Online; accessed 31-March-2019].

[14] App Traps: How Cheap Smartphones Siphon User Data in DevelopingmCountries. https://www.wsj.com/articles/app-traps-how-cheap-smartphones-help-themselves-to-user-data-1530788404. [Online; accessed 31-March-2019].

[15] Application signing. https://developer.android.com/studio/publish/app-signing. [Online; accessed 31-March-2019].

[16] Appsee — Features. https://www.appsee.com/features. [Online; accessed 31-March-2019].

[17] Appsee Mobile App Analytics. https://www.appsee.com/. [Online; accessed 31-March-2019].

[18] Asurion. https://www.asurion.com/. [Online; accessed 31-March-2019].

[19] Baidu Geocoding API. http://api.map.baidu.com/lbsapi/geocoding-api.htm. [Online; accessed 31-March-2019].

[20] Baidu SDK. https://developer.baidu.com/. [Online; accessed 31-March-2019].

[21] California Consumer Privacy Act. https://leginfo.legislature.ca.gov/faces/billTextClient.xhtml?bill_id=201720180AB375. [Online; accessed 31-March-2019].

[22] China Mobile Network Partner Redstone Moves into Robotics. https://www.prweb.com/releases/2017/04/prweb14212503.htm. [Online; accessed 31-March-2019].

[23] COPPA - Children's Online Privacy Protection Act. http://coppa.org/. [Online; accessed 31-March-2019].

[24] CVE-2017-2709. https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-2709. [Online; accessed 31-March-2019].

[25] CVE-2017-2709. https://cve.mitre.org/cgi-bin/cvename.cgi?name=cve-2015-0864. [Online; accessed 31-March-2019].

[26] Define a Custom Permission. https://developer.android.com/guide/topics/permissions/defining. [Online; accessed 31-March-2019].

[27] Digital Turbine - Privacy Policy. https://www.digitalturbine.com/privacy-policy/. [Online; accessed 31-March-2019].

[28] Estimote — indoor location with bluetooth beacons and mesh. https://estimote.com/. [Online; accessed 31-March-2019].

[29] EU General Data Protection Regulation (GDPR). https://eugdpr.org/. [Online; accessed 31-March-2019].

[30] Europe should be wary of Huawei, EU tech official says. https://www.reuters.com/article/us-eu-china-huawei-idUSKBN1O611X. [Online; accessed 31-March-2019].

[31] EXUS. https://www.exus.co.uk. [Online; accessed 31-March-2019].

[32] Facebook Gave Device Makers Deep Access to Data on Users and Friends. https://www.nytimes.com/interactive/2018/06/03/technology/facebook-device-partners-users-friends-data.html. [Online; accessed 31-March-2019].

[33] Facebook's Data Deals Are Under Criminal Investigation. https://www.nytimes.com/2019/03/13/technology/facebook-data-deals-investigation.html. [Online; accessed 31-March-2019].

[34] Firmware Scanner. https://play.google.com/store/apps/details?id=org.imdea.networks.iag.preinstalleduploader. [Online; accessed 31-March-2019].

[35] Global market share held by leading smartphone vendors. https://www.statista.com/statistics/271496/global-market-share-held-by-smartphone-vendors-since-4th-quarter-2009/. [Online; accessed 31-March-2019].

[36] GMobi — General Mobile Corporation. http://www.generalmobi.com/en/. [Online; accessed 31-March-2019].

[37] Google Cloud Messaging. https://developers.google.com/cloud-messaging/android/android-migrate-fcm. [Online; accessed 31-March-2019].

[38] Hiya. https://hiya.com/. [Online; accessed 31-March-2019].

[39] Hiya Partners. https://hiya.com/hiya-data-policy. [Online; accessed 31-March-2019].

[40] How does Truecaller get its data? https://support.truecaller.com/hc/en-us/articles/212638485-How-does-Truecaller-get-its-data. [Online; accessed 31-March-2019].

[41] Infinum Inc. https://infinum.co. [Online; accessed 31-March-2019].

[42] Intents and Intent Filters - Android Developers. https://developer.android.com/guide/components/intents-filters. [Online; accessed 31-March-2019].

[43] IronSource — App monetization done right. https://www.ironsrc.com/. [Online; accessed 31-March-2019].

[44] IronSource - AURA. https://company.ironsrc.com/enterprise-solutions/. [Online; accessed 31-March-2019].

[45] IronSource - Aura for Advertisers. https://www.slideshare.net/ironSource/aura-for-advertisers. [Online; accessed 31-March-2019].

[46] Kryptowire Discovers Mobile Phone Firmware that Transmitted Personally Identifiable Information (PII) without User Consent or Disclosure. https://www.kryptowire.com/adups_security_analysis.html. [Online; accessed 31-March-2019].

[47] Kryptowire Provides Technical Details on Black Hat 2017 Presentation: Observed ADUPS Data Collection & Data Transmission. https://www.kryptowire.com/observed_adups_data_collection_behavior.html. [Online; accessed 31-March-2019].

[48] locationlabs by Avast. https://www.locationlabs.com/. [Online; accessed 31-March-2019].

[49] Lumen Privacy Monitor. https://play.google.com/store/apps/details?id=edu.berkeley.icsi.haystack. [Online; accessed 31-March-2019].

[50] Manifest permissions. https://developer.android.com/reference/android/Manifest.permission. [Online; accessed 31-March-2019].

[51] Monetize, advertise and analyze Android apps. https://www.appbrain.com. [Online; accessed 31-March-2019].

[52] OnePlus Device Root Exploit: Backdoor in EngineerMode App for Diagnostics Mode. https://www.nowsecure.com/blog/2017/11/14/oneplus-device-root-exploit-backdoor-engineermode-app-diagnostics-mode/. [Online; accessed 31-March-2019].

[53] OnePlus left a backdoor in its devices capable of root access. http://www.androidpolice.com/2017/11/15/oneplus-left-backdoor-devices-capable-root-access/. [Online; accessed 31-March-2019].

[54] OnePlus OxygenOS built-in analytics. https://www.chrisdcmoore.co.uk/post/oneplus-analytics/. [Online; accessed 31-March-2019].

[55] OnePlus Secret Backdoor. https://www.theregister.co.uk/2017/11/14/oneplus_backdoor/. [Online; accessed 31-March-2019].

[56] Permissions overview. https://developer.android.com/guide/topics/permissions/overview.html. [Online; accessed 31-March-2019].

[57] Phone Number Search — TrueCaller. https://www.truecaller.com/. [Online; accessed 31-March-2019].

[58] Privacy Grade. http://privacygrade.org. [Online; accessed 31-March-2019].

[59] PrivacyStar. https://privacystar.com. [Online; accessed 31-March-2019].

[60] PrivacyStar Privacy Policy. https://privacystar.com/privacy-policy/. [Online; accessed 31-March-2019].

[61] Redstone. http://www.redstone.net.cn/. [Online; accessed 31-March-2019].

[62] Rootnik Android Trojan Abuses Commercial Rooting Tool and Steals Private Information. https://unit42.paloaltonetworks.com/rootnik-android-trojan-abuses-commercial-rooting-tool-and-steals-private-information/. [Online; accessed 31-March-2019].

[63] Simple to use root checking Android library. https://github.com/scottyab/rootbeer. [Online; accessed 31-March-2019].

[64] Smaato Blog. https://blog.smaato.com/everything-you-need-to-know-about-location-based-mobile-advertising. [Online; accessed 31-March-2019].

[65] Synchronoss Technologies - Privacy Policy. https://synchronoss.com/privacy-policy/#datacollected. [Online; accessed 31-March-2019].

[66] Triada Trojan Found in Firmware of Low-Cost Android Smartphones. https://www.bleepingcomputer.com/news/security/android-adware-and-ransomware-found-preinstalled-on-high-end-smartphones/. [Online; accessed 31-March-2019].

[67] Upstream - Low-end Android smartphones sold with pre-installed malicious software in emerging markets. https://www.upstreamsystems.com/pre-installed-malware-android-smartphones/. [Online; accessed 31-March-2019].

[68] VPN Service. https://developer.android.com/reference/android/net/VpnService. [Online; accessed 31-March-2019].

[69] What is "com,facebook,app manager" and why is it trying to download Instagram, Facebook, and Messenger. https://forums.androidcentral.com/android-apps/547447-what-com-facebook-app-manager-why-trying-download-instagram-facebook-messenge.html. [Online; accessed 31-March-2019].

[70] XDA-Developers Forum (Galaxy Note 4). com.facebook.appmanager. https://forum.xda-developers.com/note-4/themes-apps/com-facebook-appmanager-t2919151. [Online; accessed 31-March-2019].

[71] Your Data Is Our Data: A Truecaller Breakdown. https://techcabal.com/2018/05/02/your-data-is-our-data-a-truecaller-breakdown/. [Online; accessed 31-March-2019].

[72] Aafer, Y., Zhang, N., Zhang, Z., Zhang, X., Chen, K., Wang, X., Zhou, X., Du, W., and Grace, M. Hare Hunting In The Wild Android: A Study On The Threat Of Hanging Attribute References. In *Proceedings of the ACM Conference on Computer and Communication Security (CCS)* (2015).

[73] Aafer, Y., Zhang, X., and Du, W. Harvesting Inconsistent Security Configurations In Custom Android ROMs Via Differential Analysis. In *Proceedings of the USENIX Security Symposium* (2016).

[74] Arzt, S., Rasthofer, S., Fritz, C., Bodden, E., Bartel, A., Klein, J., Le Traon, Y., Octeau, D., and McDaniel, P. Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps. *Proceedings of the ACM Special Interest Group on Programming Languages (SIGPLAN)* (2014).

[75] Au, K. W. Y., Zhou, Y. F., Huang, Z., and Lie, D. PScout: Analyzing The Android Permission Specification. In *Proceedings of the ACM Conference on Computer and Communication Security (CCS)* (2012).

[76] Dittrich, D., and Kenneally, E. The Menlo Report: Ethical principles guiding information and communication technology research. *US Department of Homeland Security* (2012).

[77] Dr Web. Trojan preinstalled on Android devices infects applications' processes and downloads malicious modules. http://news.drweb.com/news/?i=11390&lng=en. [Online; accessed 31-March-2019].

[78] Felt, A. P., Chin, E., Hanna, S., Song, D., and Wagner, D. Android Permissions Demystified. In *Proceedings of the ACM Conference on Computer and Communication Security (CCS)* (2011).

[79] Felt, A. P., Wang, H. J., Moshchuk, A., Hanna, S., and Chin, E. Permission Re-Delegation: Attacks And Defenses. In *Proceedings of the USENIX Security Symposium* (2011).

[80] Ikram, M., Vallina-Rodriguez, N., Seneviratne, S., Kaafar, M. A., and Paxson, V. An analysis of the privacy and security risks of android vpn permission-enabled apps. In *Proceedings of the Internet Measurement Conference (IMC)* (2016).

[81] Jiang, Y. Z. X., and Xuxian, Z. Detecting Passive Content Leaks And Pollution In Android Applications. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)* (2013).

[82] Johnson, Ryan and Stavrou, Angelos and Benameur, Azzedine. All Your SMS & Contacts Belong to ADUPS & Others. https://www.blackhat.com/docs/us-17/wednesday/us-17-Johnson-All-Your-SMS-&-Contacts-Belong-To-Adups-&-Others.pdf. [Online; accessed 31-March-2019].

[83] Li, L., Bissyandé, T. F., Klein, J., and Le Traon, Y. An investigation into the use of common libraries in android apps. In *Proceedings of the International Conference on Software Analysis, Evolution, and Reengineering (SANER)* (2016).

[84] Pan, E., Ren, J., Lindorfer, M., Wilson, C., and Choffnes, D. Panoptispy: Characterizing Audio and Video Exfiltration from Android Applications. *Proceedings of the Privacy Enhancing Technologies Symposium (PETS) 2018*.

[85] Razaghpanah, A., Nithyanand, R., Vallina-Rodriguez, N., Sundaresan, S., Allman, M., Kreibich, C., and Gill, P. Apps, Trackers, Privacy, and Regulators: A Global Study of the Mobile Tracking Ecosystem. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)* (2018).

[86] Razaghpanah, A., Vallina-Rodriguez, N., Sundaresan, S., Kreibich, C., Gill, P., Allman, M., and Paxson, V. Haystack: In situ mobile traffic analysis in user space. *arXiv preprint arXiv:1510.01419* (2015).

[87] Ren, J., Lindorfer, M., Dubois, D. J., Rao, A., Choffnes, D., and Vallina-Rodriguez, N. Bug Fixes, Improvements,... and Privacy Leaks. *Proceedings of the Network and Distributed System Security Symposium (NDSS)* (2018).

[88] Reyes, I., Wijesekera, P., Reardon, J., On, A. E. B., Razaghpanah, A., Vallina-Rodriguez, N., and Egelman, S. "Won't Somebody Think of the Children?" Examining COPPA Compliance at Scale. *Proceedings of the Privacy Enhancing Technologies Symposium (PETS)* (2018).

[89] Vallina-Rodriguez, N., Amann, J., Kreibich, C., Weaver, N., and Paxson, V. A Tangled Mass: The Android Root Certificate Stores. In *Proceedings of the International Conference on Emerging Networking EXperiments and Technologies (CoNEXT)* (2014).

[90] Vallina-Rodriguez, N., Shah, J., Finamore, A., Grunenberger, Y., Papagiannaki, K., Haddadi, H., and Crowcroft, J. Breaking for commercials: characterizing mobile advertising. In *Proceedings of the Internet Measurement Conference (IMC)* (2012).

[91] Wang, H., Liu, Z., Liang, J., Vallina-Rodriguez, N., Guo, Y., Li, L., Tapiador, J., Cao, J., and Xu, G. Beyond Google Play: A Large-Scale Comparative Study of Chinese Android App Markets. In *Proceedings of the Internet Measurement Conference (IMC)* (2018).

[92] Wei, F., Roy, S., Ou, X., and Robby. Amandroid: A Precise and General Inter-component Data Flow Analysis Framework for Security Vetting of Android Apps. In *Proceedings of the ACM Conference on Computer and Communication Security (CCS)* (2014).

[93] Wu, D., Gao, D., Chang, R. K. C., He, E., Cheng, E. K. T., , and Deng, R. H. Understanding Open Ports In Android Applications: Discovery, Diagnosis, And Security Assessment. *Proceedings of the Network and Distributed System Security Symposium (NDSS)* (2019).

[94] Zhauniarovich, Y., and Gadyatskaya, O. Small Changes, Big Changes: An Updated View On The Android Permission System. In *Research in Attacks, Intrusions, and Defenses* (2016).

[95] Zhou, X., Lee, Y., Zhang, N., Naveed, M., and Wang, X. The Peril Of Fragmentation: Security Hazards In Android Device Driver Customizations. In *IEEE Symposium on Security and Privacy (SP)* (2014).

Appendix

A. Userbase distribution

Table IX describes our userbase geographical distribution.

Country (N=130)	Samples	Vendors		Vendor's share
		Total	Unique	
USA	12%	36	11	17%
Spain	6%	24	3	11%
Indonesia	6%	26	7	12%
Italy	5%	15	6	7%
UK	4%	19	6	9%
Mexico	3%	17	3	8%
Thailand	3%	28	12	13%
Germany	3%	21	2	10%
Belgium	2%	17	4	8%
Netherlands	2%	16	2	8%
Total countries	**130**	**—**	**214**	

Table IX: Geographical distribution of our users. Only the top 10 countries are shown.

B. Custom permissions

Table X reports a subset of custom permissions defined by device vendors, MNOs, third-party services, and chipset manufacturers.

MANUFACTURER PERMISSIONS

Package name	Developer Signature	Vendor(s)	Permission
`com.sonyericsson.facebook.proxylogin`	Sony Ericsson (SE)	Sony	*com.sonyericsson.permission.FACEBOOK*
`com.sonymobile.twitter.account`	Sony Ericsson (SE)	Sony	*com.sonymobile.permission.TWITTER*
`android`	Sony Ericsson (SE)	Sony	*com.sonymobile.googleanalyticsproxy.permission.GOOGLE_ANALYTICS*
`com.htc.socialnetwork.facebook`	Android (TW)	HTC	**.permission.SYSTEM_USE*
`com.sonymobile.gmailreaderservice`	Sony Ericsson (SE)	Sony	*com.sonymobile.permission.READ_GMAIL*
`com.sec.android.daemonapp`	Samsung Corporation (KR)	Samsung	**.ap.accuweather.ACCUWEATHER_DAEMON_ACCESS_PROVIDER*
`android`	Lenovo (CN)	Lenovo	*android.permission.LENOVO_MDM*
`com.asus.loguploaderproxy`	AsusTek (TW)	Asus	*asus.permission.MOVELOGS*
`com.miui.core`	Xiaomi (CN)	Xiaomi	*miui.permission.DUMP_CACHED_LOG*
`android`	Samsung (KR)	Samsung	*com.sec.enterprise.knox.KNOX_GENERIC_VPN*
`com.sec.enterprise.permissions`	Samsung (KR)	Samsung	*android.permission.sec.MDM_ENTERPRISE_VPN_SOLUTION*
`com.android.vpndialogs`	Meizu (CN)	Meizu	*com.meizu.permission.CONTROL_VPN*

MNO PERMISSIONS

Package name	Developer Signature	MNO	Permission
`com.android.mms`	ZTE	T-Mobile US	*com.tmobile.comm.RECEIVE_METRICS*
`com.lge.ipservice`	LG	T-Mobile US	*com.tmobile.comm.RECEIVE_METRICS*
`hr.infinum.mojvip`	Infinum (HR) [41]	H1 Croatia	*hr.infinum.mojvip.permission.RECEIVE_ADM_MESSAGE*
`com.locationlabs.cni.att`	AT&T (US)	AT&T (US) [48]	*com.locationlabs.cni.att.permission.BROADCAST*
`com.asurion.android.verizon.vms`	Asurion (US) [18]	Verizon (US)	*com.asurion.android.verizon.vms.permission.C2D_MESSAGE*
`jp.naver.line.android`	Naver (JP)	South Korea Telekom	*com.skt.aom.permission.AOM_RECEIVE*

THIRD-PARTY SERVICE PERMISSIONS

Package name	Developer Signature	Provider	Permission
`com.facebook.system`	Facebook	Facebook	**.ACCESS*
`com.amazon.kindle`	Amazon	Amazon	*com.amazon.identity.auth.device.perm.AUTH_SDK*
`com.huawei.android.totemweather`	Huawei (CN)	Baidu	*android.permission.BAIDU_LOCATION_SERVICE*
`com.oppo.findmyphone`	Oppo (CN)	Baidu	*android.permission.BAIDU_LOCATION_SERVICE*
`com.dti.sliide`	Logia	Digital Turbine	*com.digitalturbine.ignite.ACCESS_LOG*
`com.dti.att`	Logia	Digital Turbine	*com.dti.att.permission.APP_EVENTS*
`com.ironsource.appcloud.oobe.wiko`	ironSource	ironSource	*com.ironsource.aura.permission.C2D_MESSAGE*
`com.vcast.mediamanager`	Verizon (US)	Synchronoss	*com.synchronoss.android.sync.provider.FULL_PERMISSION*
`com.myvodafone.android`	Vodafone (GR)	Exus	*uk.co.exus.permission.C2D_MESSAGE*
`com.trendmicro.freetmms.gmobi`	TrendMicro (TW)	GMobi	*com.trendmicro.androidmup.ACCESS_TMMSMU_REMOTE_SERVICE*
`com.skype.rover`	Skype (GB)	Skype	*com.skype.android.permission.READ_CONTACTS*
`com.cleanmaster.sdk`	Samsung (KR)	CleanMaster	*com.cleanmaster.permission.sdk.clean*
`com.netflix.partner.activation`	Netflix (US)	Netflix	**.permission.CHANNEL_ID*

CHIPSET PERMISSIONS

Package name	Developer Signature	Provider	Permission
`com.qualcomm.location`	ZTE (CN)	Qualcomm	*com.qualcomm.permission.IZAT*
`com.mediatek.mtklogger`	TCL (CN)	MediaTek	*com.permission.MTKLOGGER*
`com.android.bluetooth`	Samsung (KR)	Broadcom	*broadcom.permission.BLUETOOTH_MAP*

Table X: Custom permission examples. The wildcard * represents the package name whenever the permission prefix and the package name overlap.