ITU Journal of Wireless Communications and Cybersecurity

ITU/JWCC

Should Users Trust Their Android Devices? A Scoring System for Assessing Security and Privacy Risks of Pre-Installed Applications

Abdullah Özbay[1], **Kemal Bıçakcı**[2]

[1] TOBB University of Economics and Technology, Ankara, 06560, Turkey
[2] Informatics Institute and Computer Engineering Department, Istanbul Technical University, Istanbul, 34469, Turkey

Abstract: Android devices are equipped with many pre-installed applications which have the capability of tracking and monitoring users. Although applications coming pre-installed pose a great danger to user security and privacy, they have received little attention so far among researchers in the field. In this study, we collect a dataset comprising such applications and make it publicly available. Using this dataset, we analyze tracker Software Development Kits, manifest files and the use of cloud services and report our results. We also conduct a user survey to understand concerns and perceptions of users. Finally, we present a risk scoring system which assigns scores for smart phones consolidating our findings based on carefully weighted criteria. With this scoring system, users could give their own trust decisions based on the available concise information about the security and privacy impacts of applications pre-installed on their Android devices.

Keywords: Mobile security, privacy, Android, pre-installed apps, scoring system.

Kullanıcılar Android Cihazlara Güvenmeli mi? Ön-yüklü Uygulamaların Güvenlik ve Gizlilik Risklerini Değerlendirmek İçin Bir Puanlama Sistemi

Özet: Android cihazlarda, kullanıcıları izleme ve gözlemleme yeteneğine sahip birçok ön-yüklü uygulama bulunmaktadır. Ön-yüklü uygulamalar kullanıcı güvenliği ve gizliliği için büyük bir tehlike oluşturmasına rağmen, şimdiye kadar bu uygulamalar araştırmacıların kısıtlı ilgisini çekmiştir. Bu çalışmada, böyle uygulamaları içeren bir veri kümesi oluşturduk ve bunu herkese açık hale getirdik. Bu veri kümesini kullanarak, takipçi Yazılım Geliştirme Kitleri, manifest dosyalarını ve bulut hizmetlerinin kullanımını analiz ettik ve sonuçlarımızı raporladık. Ayrıca, kullanıcıların endişelerini ve algılarını anlamak için bir kullanıcı anketi gerçekleştirdik. Son olarak, bulgularımıza dayanan dikkatlice ağırlıklandırılmış kriterlere dayalı olarak akıllı telefonlar için risk puanlama sistemi sunuyoruz. Bu puanlama sistemi ile, kullanıcılar Android cihazlarındaki ön-yüklü uygulamaların güvenlik ve gizlilik etkileri hakkında mevcut bilgilerini kullanarak bu uygulamalara güvenip güvenemeyeceklerine karar verebilirler.

Anahtar Kelimeler: Mobil güvenlik, mahremiyet, Android, ön-yüklü uygulamalar, skorlama sistemi.

RESEARCH PAPER

Corresponding Author: Abdullah Özbay, a.ozbay@etu.edu.tr

Reference: A. Özbay, K. Bıçakcı, (2024), "Should Users Trust Their Android Devices? A Scoring System for Assessing Security and Privacy Risks of Pre-Installed Applications," *ITU Journ. Wireless Comm. Cyber.*, 1, (1) 9–28.

Submission Date: Apr, 04, 2024
Acceptance Date: Sept, 09, 2024
Online Publishing: Sept, 30, 2024

1 INTRODUCTION

Android is the most widely used mobile operating system [1] in the world mainly due to two reasons: (i) it is an open-source operating system [2], (ii) Google makes manufacturers' job of producing new devices much easier if they prefer Android [3]. Not only manufacturers, but also mobile network operators, semiconductor producers and third party companies that assist and collaborate with manufacturers can easily modify and add their own applications to mobile devices with Android.

Google provides certification programs auditing Android devices, firmware and pre-installed applications. In the Android Compatibility Program, Android Compatibility Definition Document [4] is used to check for device and firmware compatibility. The requirements can be checked using Compatibility Test Suite [5]. However, in this program, there is no privacy and security audit applied to an Android device.

Google also offers Android Certified Partners Program [6] to device manufacturers. Device manufacturers have to satisfy this program's requirements to be a Android Certified Partner [7]. As part of this program, mobile Built Test Suite (BTS) [8], Security Test Suite (STS) [8] and some other suites are applied. Within BTS, Potential Harmful Applications (PHAs) and other harmful actions are examined. Also, in STS, security patches are checked to verify that pre-installed applications are up-to-date. But, neither Android Compatibility Program nor Android Certified Partners Program guarantees security and privacy of users.

In real life, many pre-installed applications threatening security and privacy of users have been already detected. One of the well-known examples is Adups discovered by Kryptowire [9]. Adups is a Firmware Over The Air (FOTA) application that helps manufacturers to update device firmware remotely. According to the analysis, this application that exists in BLU R1 HD smartphones has the ability to collect Personally Identifiable Information (PII) and run privileged code on user's devices.

As stated in Google's Android Security & Privacy 2018 Year In Review [10], Android smartphones could be infected with ease since developers of a PHA need to deceive only one of the OEMs (Original Equipment Manufacturers) or other companies in the supply chain for the installation. There were several PHAs detected in smartphones in big Android markets such as India, USA, Brazil and Indonesia. Furthermore, researchers from Oversecured have found that pre-installed applications on Samsung devices certified in Android Certified Partner Program have multiple dangerous vulnerabilities [11]. We also note that third party applications that are not directly related with OEMs e.g., social networking, search engine, news, telecommunication, etc. may also be pre-installed in Android smartphones. For example, as reported by Bloomberg [12], Facebook apps are pre-installed and cannot be deleted from smartphones. These third party applications and their affiliated companies usually cooperate with manufacturers [13].

Until recently, studies on pre-installed application ecosystem analyze only a couple of selected applications and pre-installed applications in mobile devices did not attract much attention from researchers. However, with a recent study [14] on pre-installed Android software, the gap has begun to close. On the other hand, there are many aspects of pre-installed applications that has not been explored yet. In this paper, we identify and complete the missing spots on previous work, as described next.

First of all, because there is no public data set which consists of pre-installed Android applications (We contacted the authors of previous work [14], but they informed us that sharing their dataset is not possible), our first aim is to make such a dataset available. We believe this dataset could facilitate further research on this important topic. For this purpose, we implement an Android application (Pre-App Collector) and use it to collect pre-installed applications from the devices of volunteers. As stated in Pre-App Collector's user consent screen [15], we do not access, collect, share or analyze any kind of personal data. The data being made publicly available does not disclose any personal data.

Regarding user privacy, using the collected data set, we extract tracker SDKs from applications. Then, we analyze the goals of these trackers which could be analytics, advertisement, location tracking, profiling, identification, etc. Also, we check what kind of applications (OEM, mobile network operator, social networking, etc.) contain these trackers. This analysis is the first attempt to discover tracker SDKs ecosystem on pre-installed Android applications and the effects of trackers on user privacy.

From security point of view, we make the first study in literature on critical fields of manifest files in pre-installed applications. Within this scope, we investigate exported application components, shared UIDs, attributes such as *usesCleartextTraffic, allowBackup* and *debuggable* in manifest files and find out that if pre-installed applications follow Android security best practices. In addition, we search cloud services used by Android pre-installed applications. By doing so, we intend to find out that how securely these apps take advantage of these services.

In addition, we make a survey (with users who download and use our application [15]) to understand their concerns and perceptions regarding security and privacy of pre-installed applications. Finally, we make a comprehensive evaluation of pre-installed applications from security and privacy point of views using multiple criteria based on both our and earlier findings and present a device scoring system. Device scores aim at making our findings more understandable for average users of smart phones.

To summarize, with this study we contribute to the young literature of pre-installed mobile applications and their security and privacy implications in following ways:

- We discover tracker SDKs ecosystem that exists in Android pre-installed applications.
- We analyze manifest files of applications to check compliance to security best practices.
- We analyze cloud services that are used by pre-installed applications and check if any misconfiguration exists in these services.
- We report the results of a survey applied to users who install our application [15] to shed light on user concerns and perceptions regarding security and privacy of pre-installed applications.
- We make our preinstalled app dataset publicly available [16]. The detailed metadata information about these files is available in our website [17].
- We present a scoring system to make the results of our analysis more understandable by average users. We publish our analysis results and device scores on a website [17] to inform users and researchers.

The rest of the paper is organized as follows, Section 2 summarizes the results of earlier studies on the topic. Section 3 presents our Android application developed for collecting data on pre-installed apps and provides general information about the dataset made available. Section 4 describes the analyses we perform and presents the results we obtain. Section 5 contains user survey results and related discussion. Section 6 includes the details of our scoring system and the remarks on the scores of some devices. Section 7 lists the limitations of this study. Finally, Section 8 concludes this paper.

2 RELATED WORK

There are many previous studies on applications available at Android Application Markets (e.g., [18], [19], [20], [21], [22]) as opposed to being pre-installed. A considerable portion of these focused on application permissions due to their importance with respect to user privacy and security [23], [24], [25]. Custom permissions were also studied [26]. We note that when applications are pre-installed, users do not have the chance to grant or deny dangerous application permissions [27] as they can normally do.

Third-Party Libraries (TPLs) like SDKs are crucial for Android application development as they help developers to expedite application development process. However, these TPLs may contain codes that are related to advertising and tracking services. Earlier studies [28], [29], [30], [31] found out that these services threaten user privacy.

Misconfigurations in Android application manifest files and cloud services used by applications can cause privacy and security issues. Two recent studies [32], [33] which focused on cloud service misconfigurations indicate that unsecured cloud services may expose personal data. In addition, manifest file attributes (e.g., *allowBackup, debuggable, usesCleartextTraffic*) and shared UIDs should be configured carefully as specified in the guidelines [34]. Particularly, intentional or unintentional misuse of shared UIDs may lead to over-privileged (e.g., with *android.uid.system privilege*) execution of applications [8]. Additionally, applications that have the same shared UIDs and signed with the same keys may access each other's resources. This can lead to situations which affect security and privacy of users [35], [36]. Even though there are significant advances on standardization of secure application development [37], as we observe in our work, they are not widely adopted yet in practice.

As already mentioned, most earlier work cover Android applications from Android Application Markets. Since pre-installed applications come with devices, require no further installation and most of them have more privileges beyond those available to standard developers, they demand a more elaborate and focused analysis. The effects of so called bloatware applications that come pre-installed and waste system resources like battery, disk space, memory etc. were investigated in a recent paper [38]. This paper also includes a user study conducted to understand users' knowledge and awareness regarding bloatware applications. But, it mostly focused on application permissions and their consequences. There is also a study [39] that aims to find privilege escalation vulnerabilities of pre-installed applications using taint analysis methods. In another recent study [40], pre-installed OTA applications were studied.

In another recent study [14], an analysis of pre-installed applications was presented. Although their analysis is the first large scale study on the subject, the authors admit that they were only able to scratch the surface of a much larger problem. We see that their analysis was mostly limited to third party libraries, application permissions (particularly custom permissions) and network traffic of applications.

As stated so far (and summarized in Table 1), pre-installed applications and applications from app markets differ substantially. We definitely need a better understanding of the pre-installed app ecosystem and its security and privacy implications. Our goal in this paper is to contribute in this regard and the list of our contributions is provided at the end of section 1 [1].

1 Preapp Collector app [15], which we developed independently, has a user interface and functionality comparable to the application used in [14]. But there is no repeat of analysis on the collected data set in our work, which focuses on previously unexplored aspects of pre-installed applications. On the other hand, to obtain a more comprehensive scoring system, we also consider the results of earlier work [14] as further discussed in section 6.

Table 1 Comparison of pre-installed and app market applications

Pre-installed Applications	App Market Applications
Pre-installed on devices	Installed by the user from App Markets
Runs with more privileges	User privileges
Mostly cannot be uninstalled, only disabled	Can be uninstalled
Updated less frequently	Updated more frequently
Permissions mostly automatically granted without user consent	User consent required for permissions

This paper is an extension of work originally presented in Turkish in a conference [41]. The conference paper contains essentially only a condensed and early version of our tracker analysis and user study. This paper not only presents a more elaborate discussion on these parts, but also extends our work with new sections i.e., security analysis (subsection 4.2), scoring system (section 6) and limitations (section 7).

3 PRE-APP COLLECTOR AND DATASET

In this section, we provide information about the application we develop to collect the dataset and share general statistics and some early analysis results regarding this dataset.

3.1 Android Application (Pre-App Collector)

Up to our best knowledge, no public dataset that consists of Android pre-installed applications exists. A recent study [14] has created such a dataset, but it is not publicly available. Therefore, we decide to prepare our own dataset and make it publicly available [16]. For this purpose, we implement an Android application to collect the pre-installed app data from user's devices. Our study was approved by TOBB University of Economics & Technology Human Resource Evaluation Board [42]. We make this application available on Google Play Store [15]. To announce the application, we use e-mail groups from universities, social media groups, and also share it on social media.

The application works as follows. When it starts, we inform users about our study, take their consents to start the data collection and ask a couple of questions as part of our survey to understand their concerns and perceptions regarding security and privacy of pre-installed applications. The data collected about the device includes data of manufacturer, model, product, version, timezone, SIM operator, SIM country. Then, we scan /system, /odm, /oem, /vendor, /product directories recursively to reach firmware files including pre-installed applications. Hash of these files are calculated and sent to our server to check if they already exist in our dataset. The list of files that are not in our dataset is sent to the device so that these files are also transferred to our server. Finally, we show users a summary containing the list of pre-installed applications and statistics about firmware files.

3.2 General Statistics

We present the basic statistics about our dataset as follows:

- We collect files from 22 different OEMs and 98 different devices (We distinguish non-identical devices using unique ID values. On the other hand, these values cannot be used to uniquely identify users and their devices).
- We determine using timezone information that users from at least 14 countries have installed our application.
- In total, we collect 143862 firmware files including 14178 apk files, 418 certificates and 58721 libraries.
- In total, 77 users participate in the survey (excluding survey results that have the answers as default picks or do not have a proper e-mail address).

3.3 Early Analysis and Its Results

We perform a number of early analysis. First, we use Androguard [43] which is a Python based Android reversing tool to extract certificates that are used to sign the applications [44]. We analyze the so-called Issuer field in application certificates to detect which person or company developed the application. We group these certificates because not always a single certificate is used to sign the applications developed by the same entity. We specify groups considering OEMs, OEM-related, and Third Party information (e.g., Social Networking, Web Browser, Application Marketing, Caller Identification, News, Dictionary, Cloud Service, Telecommunication Companies, Marketing & Advertising Services, etc). In total, we determine 126 certificate groups and applications under these groups.

In addition, we check what portion of determined pre-installed applications exists in Google Play Store [18] using application package name. We find out that only 9% of the applications can be accessible from Google Play. Moreover, while collecting the applications, we also obtain metadata about apps e.g., first install time and last update. The analysis of this metadata shows that 7829 out of 14178 (55%) pre-installed apps were not updated ever since they came with the devices. We note that because most of the pre-installed applications are not third party ones and located in the system partition, they can only be updated by over-the-air update mechanism released by vendors and require smartphones to be restarted. Thus, a pre-installed application cannot be easily updated like the applications from app markets.

4 ANALYSIS

We analyze pre-installed applications with respect to impacts on both user privacy and security, as detailed below.

4.1 Privacy Analysis

In privacy part, we perform a detailed analysis of tracker SDKs and privacy policies.

Tracker SDKs. Android tracker SDKs collect data about users and how they use applications. They may be embedded to pre-installed applications and have various functionalities like crash reporting, analytics, profiling, identification, advertisement, location tracking. To analyze trackers, we base our study on the work by Exodus Privacy [45], a non-profit organization working on Android trackers and their effects on user privacy. We take advantage of their tool named *exodus-standalone* [46] to detect embedded trackers in pre-installed applications. As a result, we discover tracker ecosystem and their effects to user privacy in pre-installed applications. Our early findings could be summarized as follows:

- 85 different trackers installed in 836 different applications were detected.
- We examined privacy policies of companies which use the trackers and noticed that some of them do not clearly state what kind of information they collect. (When they do not provide multi-language support, we use online translation services to investigate them.)
- In their privacy policies, most trackers stated that they track sensitive information such as PII, location-related data, log information, user behaviour, device identifiers and advertisement IDs (e.g., Google Advertising ID [47]). This practice threatens user privacy at different levels.
- Most of the trackers stated that they comply with regulations like GDPR [48] and CCPA [49], but still a few do not mention them in their privacy policies. Trackers tend to collect more data when they are not under these regulations.

Tracker Statistics. As stated above, we detected so many trackers in so many different apps. Some of these trackers are more common than the others in pre-installed applications. In Figure 1, we list the most common trackers that exist in pre-installed applications. It is not surprising to see that big technology companies such as Google, Facebook, Tencent and Amazon are dominant here.

Also, we observe that a number of applications come with excessive number of trackers which arguably makes violation of user privacy inevitable. Figure 2 lists application package names which have the highest number of tracker

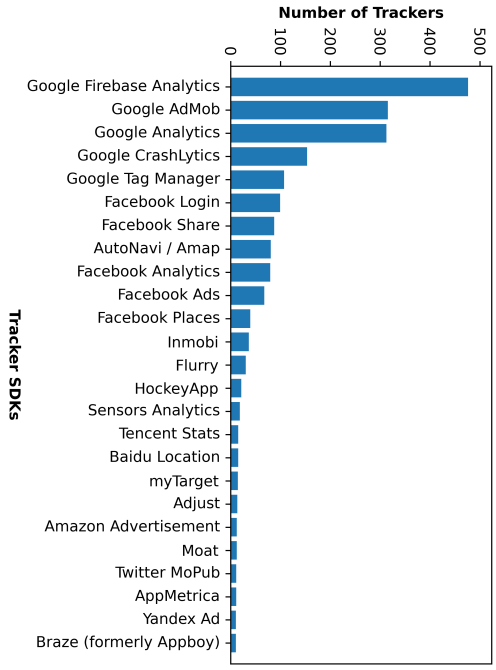


Fig. 1 Most common tracker SDKs in pre-installed applications.

Table 2 Companies and the number of tracking services related with them

Company	Number of Tracking Services
Alphabet (Parent Company of Google)	5
Facebook	5
Oath	3
Baidu	3
Microsoft	3

SDKs (different versions are considered as the same application). Interestingly, most of these applications are third party applications according to our certificate based analysis. Consequently, the devices do not actually require them to work properly.

In addition, we group trackers based on their companies. Some of the tracking services are offered by companies affiliated with big technology companies. In Table 2, we list these companies and the number of tracker-related companies that are affiliated with them.

According to our analysis (see Table 3), big technology companies acquire tracking services continuously. Once we check companies offering tracking services from Crunchbase [50], a website that provides data about companies and the people behind them, we noticed that tracking companies are acquired by other technology companies aiming to grow and expand their market share. This situation brings additional privacy risks because some tracking services state in their privacy policies that once they are ac-

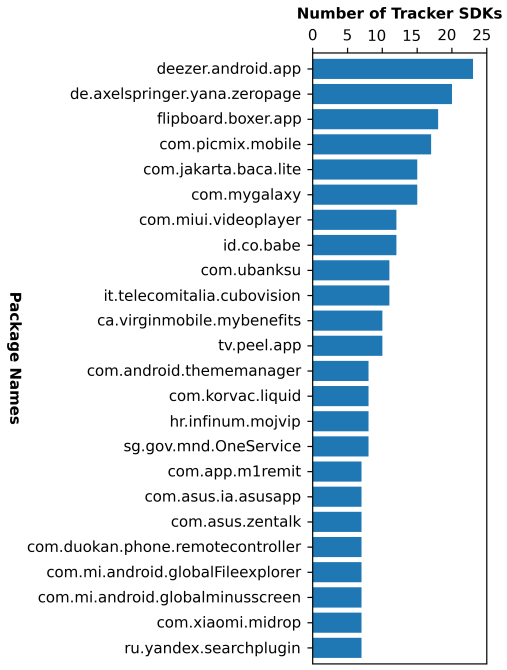


Fig. 2 Applications that contain the highest number of tracker SDKs.

Table 3 Number of tracker companies in different countries

Country	Number of Companies
United States	56
China	11
Russia	4
Germany	4
France	3
India	2
United Kingdom	1
Israel	1
Open Source	1

quired by another company, user data becomes no longer under their control and is shared with this company.

Finally, we checked the headquarters of these companies from Crunchbase. Table 3 shows the number of tracking companies located in different countries. We note that some of these countries such as Russia and China are not under any well-known regulations (e.g., GDPR, CCPA) protecting user privacy.

Purpose of Trackers. Tracker SDKs may provide different functionalities as they are designed for different purposes. Hence their impact on user privacy varies accordingly. Exodus Privacy [45] categorizes tracker SDKs in six groups:

- **Crash Reporters**: The goal of these trackers is to notify developers when applications crash.
- **Analytics**: This kind of trackers collect usage data and enable developers to learn about the users. For example, browsing behaviours are collected.
- **Profiling**: By collecting from users as much data as possible, these trackers try to build virtual profile of users. For this purpose, trackers collect data like browser history, list of installed applications, etc.
- **Identification**: The purpose of these trackers is to specify users' digital identity. Developers may associate online activities of users with their offline activities.
- **Advertisement**: The aim of these trackers is to show users targeted advertisements by using users' digital profiles and help developers to monetize their applications.
- **Location**: These trackers are used to locate users by taking advantage of Bluetooth, GPS antenna, IP address, etc.

We categorize trackers we have detected using this grouping since the effects on user privacy varies per group. Figure 3 shows the number of trackers associated with each group. As stated, each tracker group has a different functionality (some trackers perform more than one functionality). On the overall, trackers under analytics, profiling and identification groups highly threaten user privacy since they mostly need to collect personal data to fulfill their functionality. Location trackers collect location data which is also sensitive. Advertisement trackers access and collect personal data for a targeted advertisement, which might also have privacy implications. However, not all trackers are evil, crash reporters mostly do not threaten user privacy. As mentioned, they are mostly used to report application failures to help developers.

Privacy Policies. We investigate tracker companies' privacy policies and related privacy issues to understand what kind of data is collected, what data is shared with whom and whether or not these companies comply with regulations such as GDPR and CCPA.

Privacy policies confirm that all of the trackers without exception collect various types of user data. Below, we present interesting points in privacy policies of tracker companies regarding their data collection routines.

First of all, most of the tracking services collect location data in various ways. For instance, nearly all services collect IP addresses, using this information approximate location of users can be determined. Also, when available, services might access GPS data from the device to locate users. Moreover, a few of the trackers collect nearby Wi-Fi hotspots, cellular and Bluetooth information to produce the most precise location information.

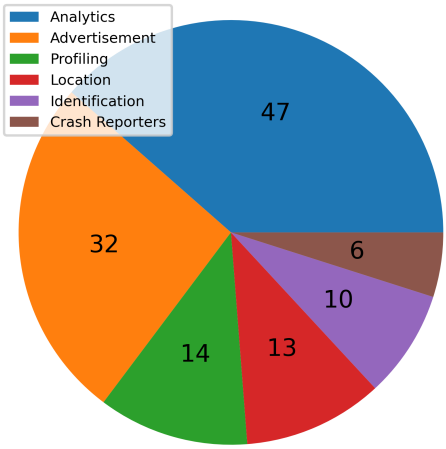


Fig. 3 Total number of trackers detected per each tracker group.

Secondly, nearly all of the trackers access advertisement IDs such as Google Advertising ID to recognize devices for advertisement purposes.

Thirdly, many of the trackers collect information about network connections such as MAC addresses, connection types (e.g., Wi-Fi, cellular) in addition to IP addresses. This information is beneficial both for location tracking and device identification.

In addition, some tracking services collect device identifiers like IMEI and IMSI numbers. This kind of data cannot be changed by users and can be used to identify the devices. The risk due to IMEI number collection is well-known [51].

Furthermore, to profile users, a number of tracking services collect information such as browser history, application log, application usage stats, cookies, etc.

Finally, some of these companies collect Personally Identifiable Information (PII) such as name, email address, gender, contact information (e.g., telephone number), etc.

Below, in order to embody the associated privacy risks, we compile a small subset of real life cases concerning trackers:

- Behavioral analytics company named Sensor Analytics, whose owner is Sang Wenfeng, former technology manager at Baidu Inc's big data department, has a partnership with Xiaomi [52] to work on tracking users.
- Citizen Lab claims that Baidu Mobile Analytics SDK causes sensitive data leaks [53]. The leak data include IMEI number, GPS location and nearby wireless access points. In addition, Baidu Map service may collect sensitive data such as IMEI number, IMSI number, MAC address, etc. [54].
- According to a research by Gizmodo, applications that use Bugly crash reporting service collect and send IMEI numbers and IP addresses to servers located in China [55].
- As stated in its privacy policy, Chinese tracking service Mintegral may collect IMEI numbers of users. Also, it cooperates with advertisement exchange platform like Google DoubleClick, Inmobi, MoPub, Tencent, Baidu, etc. [56].
- From the applications that embed its tracking code, MoEngage may obtain PII like email address, name and phone number as indicated in its privacy policy [57].
- Applications that use JPush service may send IMEI numbers, MAC addresses, serial numbers, and precise location data to Aurora Mobile's servers [58].

Data Sharing. Analysis of privacy policies shows that tracking services may share data collected from devices. In general, the data may be shared with:

- Affiliates and Subsidiaries,
- Service Providers,
- Law Enforcement Units,
- Business transfers,
- Advertisers,
- Researchers and Academics,
- Publishers,
- Data Partners.

Also, as pointed out in a study on tracking ecosystem [28], all of the ten largest tracking organizations could share collected data with third parties and subsidiaries. Because of these sharing routines, opt out chance of users is in danger since different companies have different opt out procedures. Moreover, tracking companies may share data with each other e.g., MoPub's partnership with Integral Ad Science, DoubleVerify and Moat [59].

Lastly, to the best of our knowledge, all of the tracking companies share data for legal purposes (e.g., law enforcement requirements). Even if this stems from a good intention to help law enforcement units, it can be abused by some governments [60].

Compliance with Regulations. Under the protection of regulations like CCPA, GDPR and COPPA, users have more control over their data. They can learn what kind of data is collected, with whom their data is shared or to whom it is sold, etc. Our analysis on privacy policies show that when companies are not required to comply with these

regulations, they are more likely to ignore privacy rights of users (e.g., without these regulations, as we saw in Mintegral example [56], companies continue to abuse their capabilities). High fines probably obligate the companies to adapt to these regulations and show more respect to data privacy.

4.2 Security Analysis

We analyze security practices in manifest files (*AndroidManifest.xml*) and cloud service configurations of applications.

Manifest File Analysis. A manifest file is an XML file that describes application specific essentials [61] containing app's package name, app components (activity, service, broadcast receiver, content provider), app permissions, app attributes and manifest attributes. We examine attributes such as *sharedUserId, allowBackup, usesCleartextTraffic, debuggable*, which are among the most critical fields with respect to user security. Below, we explain the security implications of misconfigurations in these fields together with our findings on the dataset.

sharedUserId. In Android, unique user ID values are assigned to each application. However, in some conditions, for instance, when the same developer or company have multiple applications on a smartphone and want to share application resources (e.g., permissions, code) with each other, the same user ID value may be assigned to these applications. For this functionality, *sharedUserId* attribute is used. But misconfiguration of this attribute may cause security vulnerabilities. Also, adversaries could take advantage of this attribute to hide their malicious codes from security analysts (because of the risk this attribute brings, it was deprecated in API level 29 by Android).

Pre-installed applications that are signed as system apps with the same certificate can run with system user privileges, one of the most privileged users in Android system. We observed that 3303 out of 14178 pre-installed applications possess shared UID value of *android.uid.system* which gives system privileges to applications. Vulnerabilities in these applications may cause adversaries to access devices with the system privileges [62]. Also, malware (e.g., Adups malware [9]) may be embedded with system privileges in devices as we have mentioned. In our analysis, we detected apps that run with system privileges without a real need (e.g., *com.caf.fmradio*). Clearly, this practice violates the least privilege principle.

allowBackup. When this attribute is valid in the manifest file and if USB debugging is enabled in an Android device, application data can be backed up by anyone who has physical access to it. Thus, all data in */data/data/package_name* can be exported from the smartphone. If any unencrypted sensitive data such as PII, passwords, keys etc. is stored in such a directory, adversaries who has physical access may easily capture it.

We examined if any application has enabled *allowBackup* attribute. We also analyzed its prevalance in each certificate group. We detected 6847 applications in total that allow backup using adb [63]. In Figure 4, vendors with the highest number applications in this configuration are presented. Almost all vendors have enabled the *allowBackup* attribute. We think this practice requires further investigation due to its security implications.

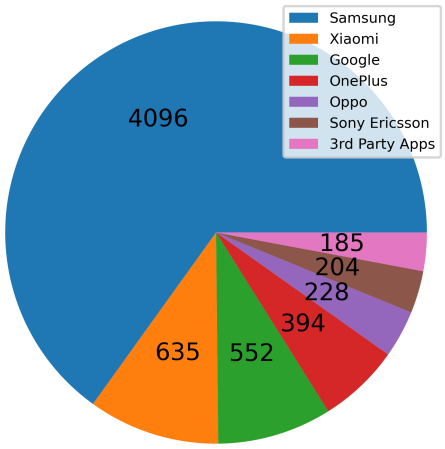


Fig. 4 Vendors and the number of pre-installed applications signed by them in our dataset that allow data backup.

usesClearTextTraffic. Applications may use cleartext traffic to connect to remote servers. This can cause private and sensitive data to be eavesdropped by adversaries [64]. With Android 6.0, application developers may prevent their applications to send cleartext data by configuring the *usesClearTextTraffic* attribute. However, we detected a considerable number of pre-installed applications with the *usesClearTextTraffic* flag set to "true". 1270 of the apps from our data set may send their data as cleartext to servers. Most of these apps are belong to OEMs and only 37 of them are belong to third parties.

debuggable. We also analyzed the configuration of *debuggagle* in pre-installed applications. When this flag is enabled in an application, it may be debugged by users who have physical access to the device with tools like jdb [65]. Using this functionality, classes and functions of apps can be easily read and even manipulated. In addition, it is possible to execute arbitrary code within the permission context of these applications. Thus, it is strongly suggested that to set this flag as "false" in production code. Fortunately, we only found 5 such applications (three variants of *com.sec.android.kiosk*, *com.trendmicro.mars.mda.httpserver*, and *com.huawei.camera2.mode.cosplay*). It was surprising to see that the later application was signed by the Android Debug Certificate [44]. Most app stores does

not accept applications that are signed with this type of certificate (recall the difference between pre-installed and app market applications).

As the conclusion of manifest file analysis, we state that the best practices regarding the use of attributes are not embraced satisfactorily by the developers of pre-installed applications.

Use of Cloud Services Almost all Android applications connect to backend servers to fulfill its functionality. These servers can be used for various purposes such as storing data, querying for information, performing actions for application, etc. Not every developer or company has the resources and time to implement their own server infrastructure. Even when they have enough resources, they might choose not to use their own server because cloud based solutions are easy to manage and provide many other advantages. These solutions offer functionalities such as data storage, notification management, analytics, API based services, etc. Due to their critical role in the mobile app ecosystem, cloud-based solutions need to be managed carefully in terms of security and privacy. Although they may be regarded as secure by default, developers should still be aware of their correct configurations and operation logic before using them. Unfortunately, considerable number of developers overlook the configuration of these solutions, that may affect millions of people.

Some of the popular cloud-based services are Google Firebase [66], Amazon Web Services (AWS) [67], Microsoft Azure [68] and Google Maps API [69]. Pre-installed applications also heavily use these services. We examined use of cloud services by these applications and misconfigurations exist in them.

These services require special keys, secrets and URL formats. Disclosure of these values may cause unauthorized access to company resources, sensitive and confidential information leaks, denial of service attacks and waste of company resources. To see whether we could extract these values, we took advantage of several tools [70, 71, 72, 73] and also wrote a few custom scripts. We also manually analyzed some of the applications by reverse engineering. As a result, we detected vulnerabilities related to Google Maps API, AWS, Firebase, Slack Webhooks [74], and OAuth [75]. Using custom Python scripts, we tested and validated our findings. Below, we discuss interesting results with respect to user security and privacy.

Google Maps API. Using this API service, developers could retrieve location-based data. Until 2018, this service was free. However, in June 2018, Google launched the pay-as-you-go pricing model [76]. In this model, the price is determined according to the number of request that is made to Product Stock-Keeping Unit (SKU) [77]. A SKU is a combination of Product API and the service or function called e.g., Place API - Photos Details.

To test whether Google Maps API key values are extractable, we used a modified version of *apkleaks* [70], a Python tool that uses special regex patterns for various URIs, endpoints and secrets for mass file scan. We tested the extracted keys using a modified version of *gmapsapis-canner* [73] so that unauthorized accesses using these keys can be verified. We present our results in Table 4 that consists of *Name of Vulnerable SKU*, *Vulnerable Application Count* that use SKU and *Impact(s)* of vulnerability that exists in SKU. These vulnerabilities may cause waste of monthly quota. Adversaries may also conduct denial of service attacks if there is a maximum bill limit.

Amazon Web Services. Since Amazon Web Services (AWS) cloud computing platform is widely used by mobile application developers and companies [78], we expect that it draws attention of attackers more than others. Mobile applications utilize Amazon Simple Storage Service (S3), which is subsidiary service of AWS to store various objects. In Amazon S3, the key concepts are Buckets, Objects, Keys and Regions. Bucket is a kind of container used to store and organize objects. Object is a fundamental entity consists of object data and metadata. To identify each object, Key is used. Finally, Region shows in which geographical region buckets are stored. For example, in the URL *https://awsexamplebucket1.s3.us-west-2.amazonaws.com/photos/puppy.jpg*, *awsexamplebucket1* is the name of the bucket, *photos/puppy.jpg* is the object and *us-west-2* is the region.

In Android ecosystem, developers need API keys (AWS access key ID and AWS secret access key) to access buckets and store objects in these buckets. Disclosure may allow adversaries to access Amazon S3 buckets and objects. Amazon has a documentation [79] that contains the best practices for managing these keys. Accordingly, these keys should not be embedded in application code directly, instead they should be stored at places suggested by Amazon or developers should use the Token Vending Machine [80]. Also, they should be renewed periodically for security reasons.

In our analysis, we detected plenty of S3 buckets, AWS access key IDs and AWS secret access keys by using *apkleaks* tool and/or by manual reverse engineering of apk files. We found a number of key pairs useful to access Amazon S3 buckets automatically. We tested them to see if any of them are still valid and can be used to access S3 buckets. We verified that accessing buckets of at least two different companies was possible. The number of valid key pairs we have found is not many but the impact could be outrageous. Using these keys, it was easy to access S3 buckets of companies which reveal not only the application information but also buckets and objects of various other applications and services. This situation clearly violates the principle of least privilege. In addition, we investigated the objects in these buckets and confirmed that sensitive information such as PII, credentials and source code of applications and ser-

Table 4 The number of vulnerable applications in our dataset for different Google Maps API SKUs

Vulnerable SKU	Vulnerable Application Count	Impact(s)
Places Photo API	199	$7 per 1000 requests
Nearby Search-Places API	198	$32 per 1000 requests
Text Search-Places API	198	$32 per 1000 requests
Find Place From Text API	196	$17 per 1000 elements
Autocomplete API	196	$2.83 per 1000 requests, Per Session - $17 per 1000 requests
Place Details API	196	$17 per 1000 requests
Staticmap API	161	$2 per 1000 requests
Geocode API	81	$5 per 1000 requests
Geolocation API	51	$5 per 1000 requests
Timezone API	36	$5 per 1000 requests
Embed (Basic) API	26	Free
Elevation API	16	$5 per 1000 requests
Streetview API	15	$7 per 1000 requests
Embed (Advanced) API	12	Free
Directions API	7	$5 per 1000 requests, (Advanced) - $10 per 1000 requests
Distance Matrix API	5	$5 per 1000 elements, (Advanced) - $10 per 1000 elements
Nearest Roads API	4	$10 per 1000 requests
Route to Traveled API	4	$10 per 1000 requests

vices could be accessed. We contacted to the companies that developed these applications about the discovered vulnerabilities via e-mail. One of them responded by confirming this vulnerability and stated that the concerned application is no longer supported by them. The other company did not respond to our e-mail. As we notified the vendors about these vulnerabilities more than a year ago, we report them in this paper in a responsible manner (without identifying them). We urge developers to use these keys securely and be aware of impacts of their disclosure.

Google Firebase Database. Google offers developers and companies a cloud based database [66] to store their data in JSON format. This database, named as Firebase Realtime Database, can be used via SDK and has some key capabilities i.e., real-time synchronization, offline response management, multiple database scalability, direct access from different clients (mobile device, web browser). To utilize this database, developers should create a database from the Firebase console. This database is named as *<database-name>.firebase.io* or if region is supplied as *<databaseName>.<region>.firebasedatabase.app*. By default, anyone can access it, hence Firebase database should be configured properly to prevent unauthorized read and write accesses.

In our work, using the *apkleaks*, we detected Firebase URLs in applications with the pattern mentioned above. We found 665 applications using Firebase databases and tested them using a custom Python script. To see if a database is readable by anyone, we simply add ".json" at the end of database URL and check the status code of the response which is 200 when readable. In addition, to find the world-writable databases, we send a put request to the database URL together with some JSON data and check the status code of response whether it is 200 or not. As a result, we found two Firebase databases that belong to two different applications everyone may read and write. Fortunately, there was no sensitive or confidential data which belong to users or companies. Developers and vendors should be careful about the Firebase database configuration as they may contain sensitive data of users and companies in other use cases.

OAuth. With *client_id* and *client_secret* values, Android applications generally use OAuth 2.0 to access different APIs or services. These values (especially *client_secret* value) should be protected against unauthorized access. For better protection, developers should take advantage of *Proof Key for Code Exchange (PKCE)* flow in which the client creates a new secret on each authorization request and uses this secret when exchanging authorization code for an access token [81]. However, in our analysis, we observed many applications that store static OAuth values belonging to services such as Google, AOL, Outlook, Office 365, Yahoo, Microsoft and mail.ru as cleartext. Thus, attackers can steal these values and use them to access APIs or services.

5 USER SURVEY

While getting help from users installing our app for building our dataset [15], we also asked them several questions to shed light on their concerns and perceptions about preinstalled applications as well as their general attitude toward smart phone usage and choices (Survey questions are provided in the Appendix APPENDIX A).

77 users attended to our survey (we eliminate results with answers all as same as the default ones and the results with an email address that has been previously used). At the be-

ginning, we asked questions on demographics. There were 40 participants in 25-34 age range and 19 in 18-24 age. 25 were female with one person chose not to provide gender information. Educational level of participants is at least Bachelor degree (70%). Only 29 of them stated they were professionally interested in cyber/mobile security. Figure 5 shows the demographic profile of survey participants.

With the survey, we try to understand user behaviour and mindset while purchasing and using their mobile phones. While 17 people did not provide any answer, most of the others (51 out of 60) bought their smartphones from online markets, technology shops or MNOs. This shows that people mostly trust large sellers when buying their phones. Arguably, this also makes sense from a privacy and security perspective. Large sellers might help users in this regard. For instance, Amazon previously suspended Blu phones which comes with pre-installed spyware [82].

According to the survey results, only 10% use devices which cost less than $100 US dollars. 44 users prefer $351-$700 devices and 27 prefer those costing $701-$1400. We remind that in general there are more security and privacy risks in less expensive smartphones [83].

We also asked questions to learn how long users have been using their phones and how often they change them. Nearly half of the users (40%) stated that their devices were between 2 and 5 years old. Even worse, a remarkable portion (11.7%) have not change their smartphones for at least 5 years. As most vendors support security updates only in their most recent models (two years on average [84]), a significant number of users are at great risk for potential security vulnerabilities. We also asked how often users change their smartphones (this is not asking the previous question again because users may have bought their devices recently). Most users (68 out of 77) change their phones after at least 2 years. As already pointed above, this brings considerable risks. The survey results about the age of smartphones used by participants can be seen in Figure 6.

In order to learn about the criteria in user choices when purchasing smartphones, we asked another question. As expected, price and model are important for most people. Only 14 participants reported they care about privacy and security policies of vendors. 13 users stated that they consider the country of the vendor as part of their purchasing decision. With these results, we argue that users should be informed better about the importance of privacy policies.

We also aim at measuring user knowledge on pre-installed applications and their impacts. We observed that the knowledge of users on the number of pre-installed applications on their device is far from the actual numbers. In Figure 7, we present the number of pre-installed applications users thought they have in their phones. Most of guesses are underestimates (more than half (%55.8) assume only 0-20 pre-installed applications). We note that we calculate the average number of pre-installed applications per device as 294 which is far more than these guesses. In addition, we asked users whether they are informed at any time about pre-installed applications. 31 users (40%) stated that they did not pay any attention to this subject. 27 of them (35%) thought that they were not informed about pre-installed applications.

To understand and compare user behaviour when managing Android permissions, we asked two additional questions. Almost half of them (38 out of 77) stated they checked application permissions before installation. On the other hand, 71% does not bother with periodic regular checks. We remind that even when application permissions are checked by users, permissions given to pre-installed applications cannot be seen.

Do users update applications in their devices when an update is available? Most of them (81%) indicated that they pay attention that applications are up-to-date. However, according to our metadata analysis, as previously mentioned, more than half of the pre-installed applications have not been updated since they came with the devices.

Finally, we asked users if they have heard about regulations like GDPR [48] or KVKK [85] (Personal Data Protection Authority in Turkey). Unfortunately, more than half of the users did not hear any of these regulations. As we discussed, these regulations have an important role for user privacy. Thus, users should be informed better about these regulations and their importance regarding user privacy.

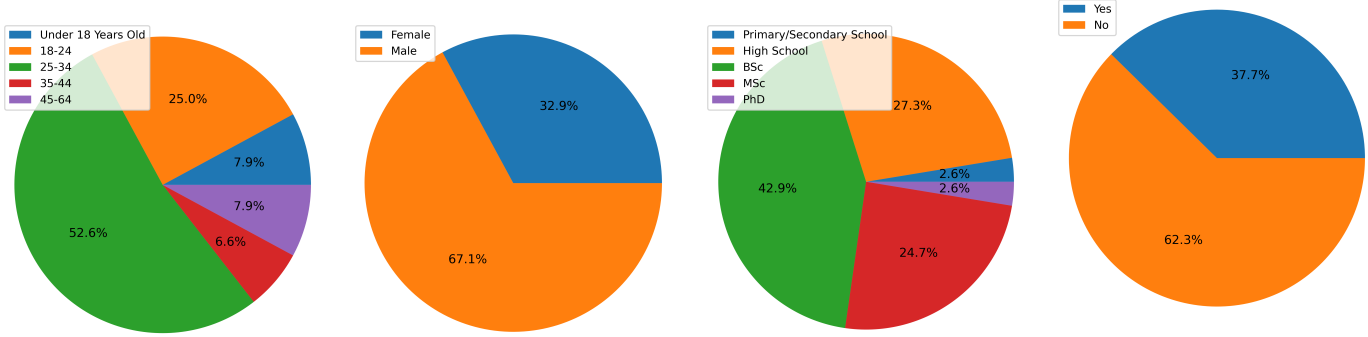
At the end of our survey, we collect email addresses of users to send our analysis results. In our view, users should be notified about pre-installed applications on their phones and their impacts on security and privacy.

6 RISK SCORING SYSTEM

As a result of series of analysis, we obtain various findings regarding pre-installed applications on smart phones that have varying degrees of effects on user security and privacy. However, these findings cannot easily be grasped by an average smart phone user especially when presented and discussed technically. For this reason, we aim at having a scoring system to provide users with information about their devices and pre-installed applications with respect to security and privacy risks in a more clear and concise way. Although, the scores seem inevitably fraught with issues of subjectivity, we believe the end result is still helpful in a certain extent.

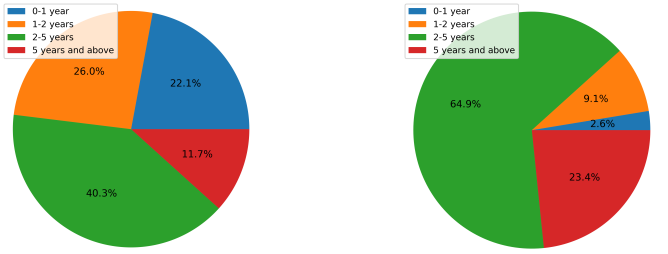
While designing our scoring system, we are inspired from the Common Vulnerability Scoring System (CVSS) [86], which assigns scores for vulnerabilities and the quantitative risk assessment methodology for IT systems proposed by Aksu et al. [87], which is built on top of CVSS scores.

The calculations in the scoring system we present essentially start with the basic risk formula as given in eq. 1.



(a) Age **(b)** Gender **(c)** Educational background **(d)** Interested in cyber/mobile security?

Fig. 5 Demographic profile of survey participants.



(a) How long have you been using your smartphone? **(b)** How often do you change your smartphone?

Fig. 6 Survey results about the age of smartphones.

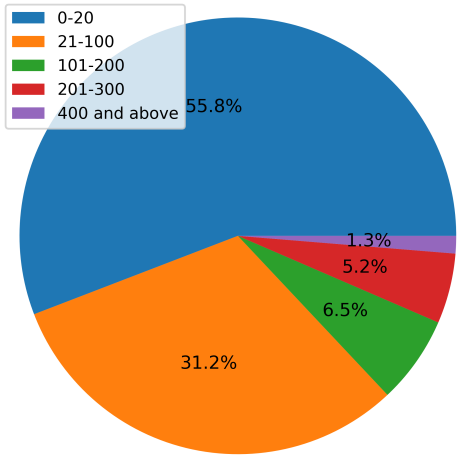


Fig. 7 How many applications were pre-installed you think when you first bought your phone?

Table 5 What is the level of difficulty to exploit the finding?

Criterion	Coefficient (d_i)
Easy (Almost no requirement)	1.00
Medium (One of either physical access, an available vulnerability or user interaction is required)	0.50
Hard (Two of physical access, an available vulnerability and user interaction are required)	0.25
Very Hard (Physical access, an available vulnerability and user interaction are all required)	0.10

$$Risk = Probability \times Cost \quad (1)$$

With this formula in mind, we first consider each finding separately and calculate a score per finding for each device. Then, we consolidate these scores to obtain a final risk score for each device we analyze.

From this perspective, for each finding we consider, three components contribute to the first parameter (Probability) of the device risk: the number of pre-installed applications (n_i) that has the concerned finding i, the difficulty level to exploit the concerned finding (d_i) and likelihood the user being aware of the exploit (once it happened) (a_i). For the later two, we grade each finding according to Tables 5 and 6 where relevant subjective coefficients are determined according to our expertise and experience. To finalize the calculation of the first parameter in the risk formula, we multiply the three components and normalize the result between 0 and 1. We emphasize that the coefficients shown have relative meanings, they do not reflect the absolute values e.g., $d_i = 1.0$ does not mean the concerned exploit is certain.

For the second parameter of the risk, we consult to Figure 7 where subjective coefficients (I_i) are available. The first and second parameters are multiplied as shown in eq. 2.

Finally, to obtain consolidated risks per device, we perform one final normalization to the sum to have device scores between 0 and 100. This is captured in eq. 3.

Table 6 Could the user be aware of the finding and its effects?

Criterion	Coefficient (a_i)
The user is unlikely aware of the finding and its effect	1.00
The user is possibly aware of the finding and its effect	0.50
The user is likely aware of the finding and its effect	0.25
The user is most likely aware of the finding and its effect	0.10

Table 7 What is the level of impact (cost) on user security and privacy?

Criterion	Coefficient (I_i)
Affects user privacy or security directly and has very high impact.(Very High)	1.00
Affects user privacy or security directly and has high impact. (High)	0.50
Possibly affects user privacy or security with high impact. (Medium)	0.25
Possibly affects user privacy or security with low impact. (Low)	0.10

$$score_i = Normalize(n_i * d_i * a_i) * I_i \tag{2}$$

$$Total\ Device\ Score = Normalize(\sum_{i=1}^{10} score_i) \tag{3}$$

Below, numerical values assigned to d_i, a_i and I_i in our scoring system are given for all of the ten findings, which is split into two groups.

6.1 New Findings

The first group contains the findings we analyze and discuss in this work.

Privileged pre-installed applications. System user is one of the most privileged users in Android devices and its use by device manufacturers is common. We detect pre-installed applications that run with system user privilege by checking if *sharedUserId* value is *android.uid.system* or not. Even though not directly affecting user privacy and security, unnecessary usage of this privilege definitely opens new attack vectors: d_1=0.25 a_1=0.50 I_1=0.25.

Applications with *allowBackup* flag enabled. In Android applications, *allowBackup* flag is used by applications to allow users to backup application data. This feature can be exploited by adversaries to reach application data but only if they have physical access: d_2=0.25 a_2=0.25 I_2=0.25.

Applications not signed by the manufacturer/vendor. We examine application certificates and detect the applications not belonging to device manufacturers. These applications mostly do not conform to the security best practices and contain tracker SDKs. Moreover, they are not strictly necessary for the normal operation of the device. When pre-installed, they run with more privileges and permissions as compared to when installed from application markets: d_3=0.50 a_3=0.50 I_3=0.25.

Applications not updated for more than two years. In our survey, most users state that they have been using their phones more than two years. Thus, their devices are open to vulnerabilities if pre-installed applications are not updated at least for two years: d_4=0.25 a_4=0.50 I_4=0.10.

Applications with *usesClearTextTraffic* flag enabled. One of the best practices in network communication is the use of TLS protocols. After Android API Level 27, applications are not allowed to make cleartext communication unless they set *usesClearTextTraffic* flag as "true" in their manifest file. However this choice is dangerous since network attacks such as man-in-the-middle becomes possible: d_5=0.50 a_5=0.50 I_5=0.25.

Applications with *debuggable* flag enabled. Use of this flag in production code is extremely dangerous. In fact, applications with *debuggable* flag set are not allowed to be uploaded to Google Play Store. When this flag is set, application methods and classes can be listed and application behaviour can be manipulated by adversaries having physically access: d_6=0.25 a_6=0.25 I_6=0.50.

Trackers (excluding crash reporters). As previously discussed in detail, tracker SDKs that come with pre-installed applications collect various kinds of user data. Users mostly are not aware of them and their activities: d_7=1.00 a_7=1.00 I_7=1.00.

Vulnerabilities in cloud services. As discussed earlier, we found a number of vulnerabilities on cloud service configurations used by pre-installed applications. We take into account the difference with respect to the impact of vulnerabilities in Google Maps API and other cloud services: d_8=1.00 a_8=1.00 I_8=0.25 (Google Maps API), I_8=1.00 (Others).

6.2 Findings from Earlier Research

Our scoring system is enriched further with the results of previous studies. The second group is composed of findings that were analyzed and discussed in previous work (but not considered in a scoring system).

We take advantage of especially one of the most comprehensive study [14] on Android pre-installed applications and include findings regarding dangerous application permissions and exported application components in our scoring system. These findings are analyzed and discussed in the previous work [14]. We use the reported procedure to obtain our results.

Exported application components not requiring permission(s). Exported application components can be used by applications to share data and functionality with other applications that are also installed on the device. But, insecure usage of these components may cause various security vulnerabilities. Our analysis on the dataset revealed that many pre-installed applications use exported components without permissions. While being not a direct threat, vulnerabilities in these components still pose a non-

negligible risk for device owners: d_9=0.25 a_9=0.25 I_9=0.10.

Dangerous permissions. In Android, dangerous permissions are those which are given to perform actions which may affect user security and privacy. After the Android API Level 23, user consent for the permissions is received at runtime. In theory, this is applied to both third-party and pre-installed applications, however vendors can enable exceptions for pre-installed applications. This can be applied by whitelisting dangerous permissions for specific pre-installed applications [88]. Also, privileged applications which are located in */system* for Android 8.1 and lower, and */system, /product, /vendor* for Android 9.0 and higher can take advantage of privilege permission allowlisting [89]. Moreover, pre-installed apps may expose critical services and data by using custom permissions [14]. This feature allows applications to use runtime permissions without user consent: d_{10}=0.25 a_{10}=0.25 I_{10}=0.25.

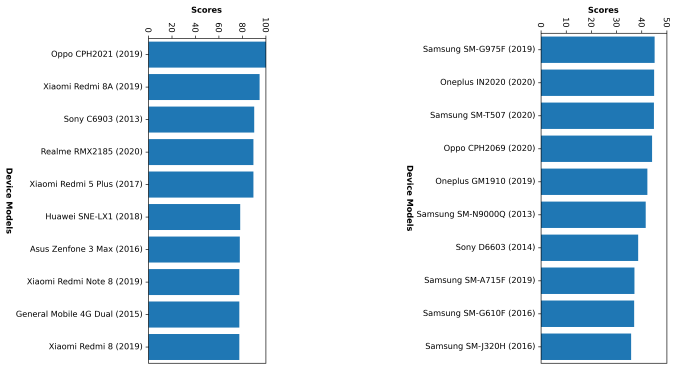
6.3 Device Scores

We use 10 different criteria as listed above and eq. 2 and eq. 3 to calculate the final device scores. Devices with the highest scores are the worst with respect to security and privacy impacts of pre-installed applications.

In our analysis, we only consider devices where we could collect more than 50 pre-installed applications since lack of enough data is most likely due to network connection problems. We also do not have sufficient data for some other devices due to various other reasons.

We determine the devices with the highest scores as seen in Figure 8 (a). Sony Xperia Z1 is the device with the highest score in our dataset. Our results also show that 7 of the 10 highest score phones are Samsung devices. Asus and General Mobile devices are also among the devices with the highest scores.

We also determine the best devices with the lowest scores. As seen in Figure 8 (b), most of these devices (6 out of 10) are released in 2019 or later.



(a) Devices with the highest-scores.

(b) Devices with the lowest scores.

Fig. 8 Devices with the highest and lowest risk scores.

With a conjecture that the total number of pre-installed applications on a device might be one of the most significant factors in risk scores, we calculate Pearson Correlation and see that the coefficient value is -0.22 which indicates that there is actually a weak negative correlation between the risk scores and the number of pre-installed applications. We illustrate this correlation in Figure 9. We could infer from this figure that risk scores reflect a more complex mix of contributing factors.

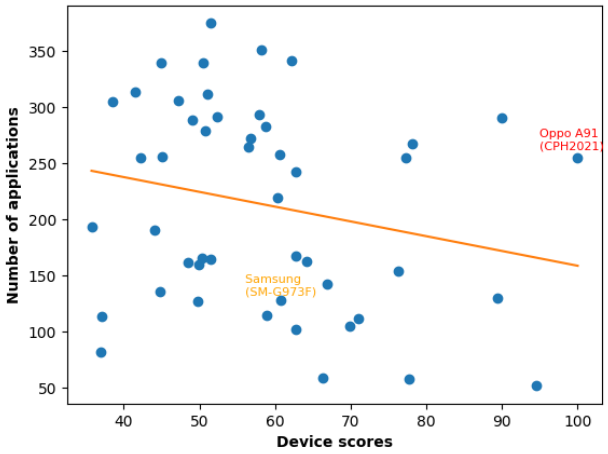


Fig. 9 The relationship between the number of pre-installed applications on devices and their risk scores.

To conclude this section, we note that with the results of our proposed risk scoring system [17], users could easily compare devices with respect to pre-installed applications and their effects on user security and privacy.

7 LIMITATIONS

In this section, we report various limitations of our study as follows:

Scope. Our study involves a dataset comprising 14178 apk files and a user study with 77 participants. These numbers are far from being sufficient to draw ultimate results on pre-installed application ecosystem. Our observation is that most people is reluctant on installing an unknown application to their smartphone even when the application is developed for an ethically approved research study. We encourage researchers to use our Android application [15] to conduct follow-up studies to obtain larger datasets of pre-installed applications [2].

Analysis. We only analyzed pre-installed applications using static analysis methods, however full functionality of applications cannot be understood only by this method be-

2 On the other hand, for obvious reasons, we do not recommend making it pre-installed.

cause these applications could take advantage of techniques such as reflection, dynamic code loading, native libraries, obfuscation and encryption. Therefore, future work may focus on developing and adopting dynamic analysis methods and platforms for pre-installed applications. Another promising future work could be analyzing privacy policies of pre-installed applications using automation tools [90].

Scoring System. The scoring system is designed using the results obtained from a limited number of devices and pre-installed applications. By adding more criteria, the scoring system can be made more comprehensive. We also note that while dangerous permissions may sometimes be actually required by the applications to fulfill their functionality, some others may use these permissions just to access and leak sensitive user data. This distinction can be detected with techniques like taint flow analysis. Additionally, different findings could be merged (e.g., tracker SDKs and dangerous permissions) to improve the reliability of scores.

8 CONCLUSION

In this work, we presented a dataset made publicly available for Android pre-installed applications. We analyzed pre-installed applications in various aspects and developed a scoring system, grading and consolidating the effects on user security and privacy. We also conducted a user study to understand and measure the knowledge and perceptions of users about pre-installed applications and their activities.

In our tracker SDK analysis, we observed that most of the tracker SDKs exist in third-party applications. However, users cannot uninstall these applications, they could only deactivate them. Although these applications are not required for proper device operation, they have serious security and usability impacts. Also, we detected tracker SDKs on vendor pre-installed applications, which confirms that vendors and third-party firms collaborate with each other.

We analyzed critical manifest file attributes and flags such as *sharedUserId, allowBackup, debuggable, usesClearTextTraffic* and determined various pre-installed applications having critical security vulnerabilities. Vendors are urged to follow security best practices while developing pre-installed applications.

We examined cloud services in pre-installed applications and detected various vulnerabilities that affect user security and privacy in varying levels. Some of these allow even unauthorized access to data of other applications and users.

The user survey we conducted to learn knowledge and perception of users about pre-installed applications showed that most of the participants have limited knowledge about them. One takeaway is that users should be informed better about pre-installed applications and their effects on user security and privacy. For this purpose, we developed a website [17] and published our analysis results for each device we analyzed.

The scoring system we developed takes into account the difficulty of exploiting, the awareness level of users and the impact on security and privacy. We evaluated the devices with respect to ten different findings and the normalized sum of scores for findings gave us a total device score. With this score, users may easily form an opinion concerning the security and privacy impacts of mobile devices and pre-installed applications.

To sum up, pre-installed applications in Android devices can affect security and privacy of users in multiple ways. However, this topic has not drawn much attention in academic literature. We encourage researchers to take advantage of our available dataset [16]. We believe there are still many aspects of pre-installed applications awaiting to be uncovered.

COMPLIANCE WITH ETHICAL STANDARDS

Data collection and user survey that are made as part of this study is ethically approved by TOBB University of Economics & Technology Human Research Evaluation Board [42]. Collected information does not contain any personal data and is not shared with any third party.

COMPETING INTERESTS

The authors declare that they have no competing interests.

RESEARCH DATA POLICY AND DATA AVAILABILITY STATEMENT

The data set is created as part of this study can be accessed from Kaggle [16]. Also, analyses results are available in our website [17].

REFERENCES

[1] “Mobile operating system market share worldwide | statcounter global stats,” https://gs.statcounter.com/os-market-share/mobile/worldwide, (Accessed on 29/11/2022).

[2] “Android open source project,” https://source.android.com/, (Accessed on 29/11/2022).

[3] “Android compatibility program overview - android open source project,” https://source.android.com/compatibility/overview?hl=en, (Accessed on 29/11/2022).

[4] “Android compatibility definition document,” https://source.android.com/compatibility/cdd, (Accessed on 29/11/2022).

[5] “Compatibility test suite - android open source project,” https://source.android.com/compatibility/cts, (Accessed on 29/11/2022).

[6] "Android - certified," https://www.android.com/certified/, (Accessed on 29/11/2022).

[7] "Android certified partners," https://www.android.com/certified/partners/, (Accessed on 29/11/2022).

[8] "Securing the system: A deep dive into reversing android preinstalled apps," https://i.blackhat.com/USA-19/Thursday/us-19-Stone-Securing-The-System-A-Deep-Dive-Into-Reversing-Android-Preinstalled-Apps.pdf, (Accessed on 29/11/2022).

[9] "Android firmware sending private information without consent - kryptowire," https://www.kryptowire.com/kryptowire-discovers-mobile-phone-firmware-transmitted-personally-identifiable-information-pii-without-user-consent-disclosure/, (Accessed on 29/11/2022).

[10] "Google android security 2018 report final.pdf," https://source.android.com/security/reports/Google_Android_Security_2018_Report_Final.pdf, (Accessed on 29/11/2022).

[11] "Two weeks of securing samsung devices: Part 1 - oversecured blog," https://blog.oversecured.com/Two-weeks-of-securing-Samsung-devices-Part-1/, (Accessed on 29/11/2022).

[12] "Facebook app can't be deleted from certain samsung phones - bloomberg," https://www.bloomberg.com/news/articles/2019-01-08/samsung-phone-users-get-a-shock-they-can-t-delete-facebook, (Accessed on 29/11/2022).

[13] "Facebook gave device makers deep access to data on users and friends - the new york times," https://www.nytimes.com/interactive/2018/06/03/technology/facebook-device-partners-users-friends-data.html?mtrref=undefined&gwh=DFAE7B3996870E0D2452CDBF4B2F1154&gwt=pay&assetType=PAYWALL, (Accessed on 29/11/2022).

[14] J. Gamba, M. Rashed, A. Razaghpanah, J. Tapiador, and N. Vallina-Rodriguez, "An analysis of pre-installed android software," in *2020 IEEE Symposium on Security and Privacy (SP)*, 2020, pp. 1039–1055.

[15] "Pre-app collector application - google play," https://play.google.com/store/apps/details?id=com.preappcollector, (Accessed on 29/11/2022).

[16] "Android pre-installed applications | kaggle," https://www.kaggle.com/abdullahzbay/android-preinstalled-applications, (Accessed on 29/11/2022).

[17] "Pre-app collector website," https://preappcollector.com/, (Accessed on 29/11/2022).

[18] "Google play store," https://play.google.com/store, (Accessed on 29/11/2022).

[19] "Galaxy store apps - the official samsung galaxy site," https://www.samsung.com/global/galaxy/apps/galaxy-store/, (Accessed on 29/11/2022).

[20] "The amazon app," https://www.amazon.com/gp/mas/get/amazonapp, (Accessed on 29/11/2022).

[21] "F-droid - free and open source android app repository," https://f-droid.org/en/, (Accessed on 29/11/2022).

[22] "Apkpure.com," https://apkpure.com/, (Accessed on 29/11/2022).

[23] K. W. Y. Au, Y. F. Zhou, Z. Huang, and D. Lie, "Pscout: Analyzing the android permission specification," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, ser. CCS '12. New York, NY, USA: Association for Computing Machinery, 2012, pp. 217–228. [Online]. Available: https://doi.org/10.1145/2382196.2382222

[24] A. P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner, "Android permissions demystified," in *Proceedings of the 18th ACM Conference on Computer and Communications Security*, ser. CCS '11. New York, NY, USA: Association for Computing Machinery, 2011, pp. 627–638. [Online]. Available: https://doi.org/10.1145/2046707.2046779

[25] C. Gibler, J. Crussell, J. Erickson, and H. Chen, "Androidleaks: Automatically detecting potential privacy leaks in android applications on a large scale," in *Trust and Trustworthy Computing*, S. Katzenbeisser, E. Weippl, L. J. Camp, M. Volkamer, M. Reiter, and X. Zhang, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 291–307.

[26] G. Tuncay, S. Demetriou, K. Ganju, and C. Gunter, "Resolving the predicament of android custom permissions," 01 2018.

[27] B. Liu, J. Lin, and N. Sadeh, "Reconciling mobile app privacy and usability on smartphones: Could user privacy profiles help?" in *Proceedings of the 23rd International Conference on World Wide Web*, ser. WWW '14. New York, NY, USA: Association for Computing Machinery, 2014, pp. 201–212. [Online]. Available: https://doi.org/10.1145/2566486.2568035

[28] A. Razaghpanah, R. Nithyanand, N. Vallina-Rodriguez, S. Sundaresan, M. Allman, C. Kreibich, and P. Gill, "Apps, trackers, privacy, and regulators: A global study of the mobile tracking ecosystem," in *NDSS*, 2018.

[29] R. Binns, U. Lyngs, M. Van Kleek, J. Zhao, T. Libert, and N. Shadbolt, "Third party tracking in the mobile ecosystem," in *Proceedings of the 10th ACM Conference on Web Science*, ser. WebSci '18. New York, NY, USA: Association for Computing Machinery, 2018, pp. 23–31. [Online]. Available: https://doi.org/10.1145/3201064.3201089

[30] H. Wang, H. Li, and Y. Guo, "Understanding the evolution of mobile app ecosystems: A longitudinal measurement study of google play," in *The World Wide Web Conference*, ser. WWW '19. New York, NY, USA: Association for Computing Machinery, 2019, pp. 1988–1999. [Online]. Available: https://doi.org/10.1145/3308558.3313611

[31] B. Hu, Q. Lin, Y. Zheng, Q. Yan, M. Troglia, and Q. Wang, "Characterizing location-based mobile tracking in mobile ad networks," in *2019 IEEE Conference on Communications and Network Security (CNS)*, 2019, pp. 223–231.

[32] "Unsecured cloud configurations exposing information in thousands of mobile apps," https://blog.zimperium.com/unsecured-cloud-configurations-exposing-information-in-thousands-of-mobile-apps/, (Accessed on 29/11/2021).

[33] "Mobile app developers' misconfiguration of third party services leave personal data of over 100 million exposed - check point research," https://research.checkpoint.com/2021/mobile-app-developers-misconfiguration-of-third-party-services-leave-personal-data-of-over-100-million-exposed/, (Accessed on 29/11/2021).

[34] "Mobile security testing guide," https://mobile-security.gitbook.io/mobile-security-testing-guide/, (Accessed on 29/11/2021).

[35] D. Barrera, J. Clark, D. McCarney, and P. C. van Oorschot, "Understanding and improving app installation security mechanisms through empirical analysis of android," in *Proceedings of the Second ACM Workshop on Security and Privacy in Smartphones and Mobile Devices*, ser. SPSM '12. New York, NY, USA: Association for Computing Machinery, 2012, pp. 81–92. [Online]. Available: https://doi.org/10.1145/2381934.2381949

[36] E. Ratazzi, Y. Aafer, A. Ahlawat, H. Hao, Y. Wang, and W. Du, "A systematic security evaluation of android's multi-user framework," *ArXiv*, vol. abs/1410.7752, 2014.

[37] S. M. Dye and K. Scarfone, "A standard for developing secure mobile applications," *Computer Standards & Interfaces*, vol. 36, no. 3, pp. 524–530, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0920548913001268

[38] H. Elahi, G. Wang, and J. Chen, "Pleasure or pain? an evaluation of the costs and utilities of bloatware applications in android smartphones," *J. Netw. Comput. Appl.*, vol. 157, no. C, May 2020. [Online]. Available: https://doi.org/10.1016/j.jnca.2020.102578

[39] M. Elsabagh, R. Johnson, A. Stavrou, C. Zuo, Q. Zhao, and Z. Lin, "FIRMSCOPE: Automatic uncovering of privilege-escalation vulnerabilities in pre-installed apps in android firmware," in *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, Aug. 2020, pp. 2379–2396. [Online]. Available: https://www.usenix.org/conference/usenixsecurity20/presentation/elsabagh

[40] E. Blázquez, S. Pastrana, A. Feal, J. Gamba, P. Kotzias, N. Vallina-Rodriguez, and J. Tapiador, "Trouble over-the-air: An analysis of fota apps in the android ecosystem," in *2021 IEEE Symposium on Security and Privacy (SP)*, 2021, pp. 1606–1622.

[41] A. Ozbay and K. Bicakci, "Android pre-installed applications effects on user's privacy," in *2021 International Conference on Information Security and Cryptology (ISCTURKEY)*, 2021, pp. 12–17.

[42] "ethical_approval.pdf," https://preappcollector.com/static/ethical_approval.pdf, (Accessed on 29/11/2022).

[43] "Androguard," https://github.com/androguard/androguard, (Accessed on 29/11/2022).

[44] "Application signing," https://developer.android.com/studio/publish/app-signing, (Accessed on 29/11/2022).

[45] "exodus," https://reports.exodus-privacy.eu.org/en/, (Accessed on 29/11/2022).

[46] "exodus-standalone," https://github.com/Exodus-Privacy/exodus-standalone, (Accessed on 29/11/2022).

[47] "Google advertising id - play console help," https://support.google.com/googleplay/android-developer/answer/6048248, (Accessed on 29/11/2022).

[48] "General data protection regulation (gdpr)," https://gdpr-info.eu/, (Accessed on 29/11/2022).

[49] "California consumer privacy act (ccpa)," https://oag.ca.gov/privacy/ccpa, (Accessed on 29/11/2022).

[50] "Crunchbase: Discover innovative companies and the people behind them," https://www.crunchbase.com/, (Accessed on 29/11/2022).

[51] "Why do you even need the imei?" https://blog.appcensus.io/2019/04/26/why-do-you-even-need-the-imei/, (Accessed on 29/11/2022).

[52] "Exclusive: Warning over chinese mobile giant xiaomi recording millions of people's 'private' web and phone use," https://www.forbes.com/sites/thomasbrewster/2020/04/30/exclusive-warning-over-chinese-mobile-giant-xiaomi-recording-millions-of-peoples-private-web-and-phone-use/?sh=7579d95c1b2a, (Accessed on 29/11/2022).

[53] "Baidu's and don'ts: Privacy and security issues in baidu browser," https://citizenlab.ca/2016/02/privacy-security-issues-baidu-browser/, (Accessed on 29/11/2022).

[54] "Data leakage found from android apps on google play with millions of downloads," https://unit42.paloaltonetworks.com/android-apps-data-leakage/, (Accessed on 29/11/2022).

[55] "Dji releases security findings it hopes will quash 'chinese spying' fears," https://gizmodo.com/dji-releases-security-findings-it-hopes-will-quash-chin-1825469976, (Accessed on 29/11/2022).

[56] "Privacy policy - mintegral," https://www.mintegral.com/en/privacy/, (Accessed on 29/11/2022).

[57] "Privacy policy - moengage," https://www.moengage.com/privacy-policy/, (Accessed on 29/11/2022).

[58] "Report: Aurora mobile's jpush sdk - the appcensus blog," https://blog.appcensus.io/2020/09/15/report-aurora-mobiles-jpush-sdk/, (Accessed on 29/11/2022).

[59] "Industry collaborations - mopub," https://www.mopub.com/en, (Accessed on 29/11/2022).

[60] Z. Wang, "Systematic government access to private-sector data in China," *International Data Privacy Law*, vol. 2, no. 4, pp. 220–229, 07 2012. [Online]. Available: https://doi.org/10.1093/idpl/ips017

[61] "App manifest overview | android developers," https://developer.android.com/guide/topics/manifest/manifest-intro, (Accessed on 29/11/2022).

[62] "Nvd - cve-2018-14825," https://nvd.nist.gov/vuln/detail/CVE-2018-14825, (Accessed on 29/11/2022).

[63] "Android debug bridge (adb) | android developers," https://developer.android.com/studio/command-line/adb, (Accessed on 29/11/2022).

[64] "Android developers blog: Protecting against unintentional regressions to cleartext traffic in your android apps," https://android-developers.googleblog.com/2016/04/protecting-against-unintentional.html, (Accessed on 29/11/2022).

[65] "jdb - the java debugger," https://docs.oracle.com/javase/7/docs/technotes/tools/windows/jdb.html, (Accessed on 29/11/2022).

[66] "Firebase," https://firebase.google.com/, (Accessed on 29/11/2022).

[67] "Amazon web services (aws) - cloud computing services," https://aws.amazon.com/, (Accessed on 29/11/2022).

[68] "Cloud computing services | microsoft azure," https://azure.microsoft.com/en-us/, (Accessed on 29/11/2022).

[69] "Google maps platform | google developers," https://developers.google.com/maps, (Accessed on 29/11/2022).

[70] "dwisiswant0/apkleaks: Scanning apk file for uris, endpoints & secrets." https://github.com/dwisiswant0/apkleaks, (Accessed on 29/11/2022).

[71] "skylot/jadx: Dex to java decompiler," https://github.com/skylot/jadx, (Accessed on 29/11/2022).

[72] "Apktool - a tool for reverse engineering 3rd party, closed, binary android apps." https://ibotpeaches.github.io/Apktool/, (Accessed on 29/11/2022).

[73] "gmapsapiscanner," https://github.com/ozguralp/gmapsapiscanner, (Accessed on 29/11/2022).

[74] "Sending messages using incoming webhooks | slack," https://api.slack.com/messaging/webhooks, (Accessed on 29/11/2022).

[75] "Oauth 2.0 - oauth," https://oauth.net/2/, (Accessed on 29/11/2022).

[76] "Billing: Mapping previous skus to new skus | google maps platform," https://developers.google.com/maps/billing/sku-mapping-old-to-new, (Accessed on 29/11/2022).

[77] "Google maps platform billing | google developers," https://developers.google.com/maps/billing/gmp-billing, (Accessed on 29/11/2022).

[78] "Global cloud infrastructure market share 2021 | statista," https://www.statista.com/statistics/967365/worldwide-cloud-infrastructure-services-market-share-vendor/, (Accessed on 29/11/2022).

[79] "Aws general reference - reference guide," https://docs.aws.amazon.com/general/latest/gr/aws-general.pdf#aws-access-keys-best-practices, (Accessed on 29/11/2022).

[80] “Authenticating users of aws mobile applications with a token vending machine - aws articles,” https://aws.amazon.com/tr/articles/authenticating-users-of-aws-mobile-applications-with-a-token-vending-machine/, (Accessed on 29/11/2022).

[81] “Protecting mobile apps with pkce - oauth 2.0 simplified,” https://www.oauth.com/oauth2-servers/pkce/, (Accessed on 29/11/2022).

[82] “Amazon suspends sales of blu phones for including preloaded spyware, again - the verge,” https://www.theverge.com/2017/7/31/16072786/amazon-blu-suspended-android-spyware-user-data-theft, (Accessed on 29/11/2022).

[83] “Buying a smart phone on the cheap? privacy might be the price you have to pay - privacy international,” https://privacyinternational.org/long-read/3226/buying-smart-phone-cheap-privacy-might-be-price-you-have-pay, (Accessed on 29/11/2022).

[84] “Mobile security updates: Understanding the issues,” https://www.ftc.gov/system/files/documents/reports/mobile-security-updates-understanding-issues/mobile_security_updates_understanding_the_issues_publication_final.pdf, (Accessed on 29/11/2022).

[85] “Kişisel verileri koruma kurumu | kvkk | personal data protection authority,” https://www.kvkk.gov.tr/en/, (Accessed on 29/11/2022).

[86] “What are cvss scores | balbix,” https://www.balbix.com/insights/understanding-cvss-scores/, (Accessed on 29/11/2022).

[87] M. U. Aksu, M. H. Dilek, E. I. Tatli, K. Bicakci, H. I. Dirik, M. U. Demirezen, and T. Aykir, “A quantitative cvss-based cyber security risk assessment methodology for it systems,” in *2017 International Carnahan Conference on Security Technology (ICCST)*. IEEE, 2017, pp. 1–8.

[88] “Runtime permissions | android open source project,” https://source.android.com/devices/tech/config/runtime_perms?hl=en#creating-exceptions, (Accessed on 29/11/2022).

[89] “Privileged permission allowlisting | android open source project,” https://source.android.com/devices/tech/config/perms-allowlist?hl=en, (Accessed on 29/11/2022).

[90] H. Harkous, K. Fawaz, R. Lebret, F. Schaub, K. G. Shin, and K. Aberer, “Polisis: Automated analysis and presentation of privacy policies using deep learning,” in *27th USENIX Security Symposium (USENIX Security 18)*, 2018, pp. 531–548.

APPENDIX A Survey Questions

1) Please select your age range.
a. Under 18 years old
b. 18-24
c. 25-34
d. 35-44
e. 45-64

2) Please select your gender.
a. Female
b. Male
c. Prefer not to answer

3) What is your educational background?
a. Primary School-Secondary School
b. High school
c. Bachelor (BSc)
d. Master of Science (MSc)
e. Doctorate (PhD)

4) Are you professionally interested in cyber security / mobile security?
a. Yes
b. No

5) Where did you buy your smartphone?
Technology Store, Telecommunication Company, Local Store, 2nd Hand Seller, Online Market etc. (Text Box)

6) How much money did you pay for your smartphone?
a. 0-130 $
b. 131-350 $
c. 351-700 $
d. 701-1400 $
e. 1400 $ and above

7) How long have you been using your smartphone?
a. 0-1 Year
b. 1-2 Years
c. 2-5 Years
d. 5 Years and above

8) How often do you change your smartphone?
a. 0-1 Year
b. 1-2 Years
c. 2-5 Years
d. 5 Years and above

9) How many pre-installed applications (already installed on the device when the device came out of the box) do you think there are when you first bought your phone?
a. 0-20
b. 21-100
c. 101-200
d. 201-300
e. 301-400
f. 400 and above

10) When purchasing a smartphone, select the factors that affect your purchasing decision. (Note: Users can choose multiple choices)
a. Price
b. Model
c. Popularity
d. Country of manufacturer (Samsung: South Korea, Huawei: China, etc.)
e. Security and Privacy Policy of the Manufacturer / Seller
f. Whether it is Sold / Manufactured by large and well-known companies

11) While setting up your smartphone, have you been informed about the pre-installed applications and the operations these applications perform and the data they collect?
a. Yes, I have been informed.
b. No, I havenŕ been informed.
c. I did not pay attention / I did not read.

12) Did you give any permission for pre-installed apps?
a. No.
b. Yes.
c. I did not pay attention / I did not recall.

13) Do you pay attention to what permissions the apps you install on your phone use?
a. No, I don't pay attention.
b. Yes, I pay attention.

14) Do you regularly check these permissions?
a. Yes, I'm checking.
b. No, I'm not checking.

15) Do you check that applications on your smartphone are up-to-date?
a. Yes
b. No

16) Do you think you know enough about General Data Protection Regulation (GDPR) or Personal Data Protection Authority in Turkey (KVKK)?
a. Yes
b. No