BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **25**, No 1
Sofia • 2025 Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.2478/cait-2025-0007

Detecting the Inconsistency between Android Apps' Data Collection and Google Play's Data Safety Using Static Analysis

Rawan Baalous, Alanoud Althobaiti, Dareen Alyoubi, Rama Alzahrani, Mona Aljohani

Cybersecurity Department, University of Jeddah, Jeddah, Saudi Arabia
E-mails: rsbaalous@uj.edu.sa aalthubayti.stu@uj.edu.sa dalyubi.stu@uj.edu.sa ralzahrani0647.stu@uj.edu.sa maljohani0169.stu@uj.edu.sa

Abstract: *In the rapidly evolving landscape of Android mobile apps, ensuring user data privacy remains paramount. Google introduced a Data Safety section on the app listing page to display privacy and security practices in a short format. Thereby enabling users to make informed decisions regarding the app's download and usage. Google left the responsibility of providing accurate and complete information on the Data Safety section to the developers. This makes the credibility of the Data Safety section questionable. A static analysis approach has been proposed to verify the consistency between the Android app's source code and its Data Safety section to ensure that the app behaves as its Data Safety section promises. By analyzing 4980 apps, a significant 67.7% of the apps were found to have inconsistencies, indicating potential misrepresentation of data collection practices. This research highlights the need for* ***rigorous*** *verification of Data Safety information to enhance user trust and privacy.*

Keywords: *Data Safety, Android privacy, Dangerous permissions, Privacy policies.*

1. Introduction

Android users have access to a wide range of apps through the Google Play store. Each app has metadata such as description, category, and a section titled *Data Safety*, which offers information on privacy and security practices. Based on the metadata provided by an app, users decide whether it is trusted for installation and use [1].

Google Play requires app developers to provide a privacy policy for each app. Google reviews these policies and takes enforcement action for any violation [2]. Users ignore these policies because they are long and unnoticeable on the app listing page. For this reason, Google announced that by July 20, 2022, all apps must have a Data Safety section that explains their data collecting and sharing methods [3]. Unlike privacy policies, Data Safety provides a simple, readable, concise, and noticeable way to display information related to privacy and security practices taken by the app on the app listing page.

The Data Safety section presents information on data collection, sharing, and security practices in a user-friendly format. However, developers are responsible for ensuring the accuracy and completeness of this information [3], creating a risk of false or misleading disclosures. A recent study [4] highlights this concern, revealing discrepancies in 80% of reviewed apps between Google Play's Data Safety and their privacy policies.

This study evaluates the compliance of Android apps' Data Safety section with their actual privacy practices, focusing on the consistency of data collection claims with the app's source code. Data sharing and other security practices are beyond the scope of this work. To determine the data an app collects, it is crucial to identify the permissions granted to it [5]. The AndroidManifest.xml file within the Android Package Kit (APK) contains the permissions for accessing protected parts of the system or other applications [6]. This work primarily focuses on dangerous permissions, which access sensitive data like photo albums and locations [7]. Additionally, elements in the AndroidManifest.xml, such as the ⟨activity⟩ tag, may indicate data collection. Data from the AndroidManifest.xml is systematically compared with the Data Safety section to identify inconsistencies. To summarize, this paper answers the following questions:

- **RQ1.** What user privacy-related data are collected by Android apps?
- **RQ2.** What information do the developers claim to collect as stated in the Data Safety section?
- **RQ3.** How accurate is the information in the Data Safety section about data collection when compared to the user privacy-related data collected actually by apps?

This study has the following contributions:

- Creation of a dataset of Android apps' Data Safety sections and their APK files.
- Analysis of the Data Safety section to extract data collection-related information.
- Extraction of dangerous permissions from the source code using static analysis.
- Extraction of data collection indicators within the declarations of the ⟨activity⟩ elements from source code using semantic analysis.
- Comparison of information extracted from the source code with that obtained from the Data Safety section to identify inconsistencies.

The rest of this paper is organized as follows: Section 2 provides background information relevant to the study. Section 3 reviews related works. Section 4 presents the methodology, including dataset collection, extraction of data collection-related statements, source code analysis, and detection of inconsistencies. Section 5 reports the results, followed by a discussion in Section 6. Finally, Section 7 concludes the paper by summarizing the key findings.

2. Background

In this section, background information about apps' permissions, APK files, and the Data Safety section in Google Play are presented.

2.1. Apps' permissions

The Android permission system is a key security feature that controls access to system resources and user privacy [8]. Apps must request permission to access sensitive information, but many users consent without fully understanding them. Some apps exploit this by requesting additional permissions to collect user data. Android permissions are categorized into install-time, runtime, and special permissions [9].

Each permission type defines the scope of restricted data and actions an app can access upon system approval. Install-time permissions have minimal impact on the system and other apps, granting limited access [9]. When declared, these permissions are shown on the app's details page in the store, and the system automatically grants them. Runtime permissions, also known as dangerous permissions, grant apps access to sensitive data or actions during runtime, posing higher risks to the system and other apps. For instance, accessing a phone's camera is a dangerous permission. When such permission is requested, users are prompted to grant or deny access via a dialog. However, if access is granted, the app may gain additional permissions within the same group without further user consent [10]. Table 1 lists examples of dangerous permissions [11].

Table 1. Dangerous permissions

Permission group	Dangerous permissions
CALENDAR	READ_CALENDER WRITE_CALENDER
CONTACTS	READ_CONTACTS WRITE_CONTACTS GET_ACCOUNTS
LOCATION	ACCESS_FINE_LOCATION ACCESS_COARSE_LOCATION ACCESS_MEDIA_LOCATION
PHONE	READ_PHONE_STATE READ_PHONE_NUMBERS
CALL_LOG	READ_CALL_LOG WRITE_CALL_LOG
SENSORS	BODY_SENSORS
MICROPHONE	RECORD_AUDIO
ACTIVITY_RECOGNITION	ACTIVITY_RECOGNITION
SMS	READ_SMS SEND_SMS RECEIVE_SMS RECEIVE_WAP_PUSH RECEIVE_MMS
STORAGE	READ_EXTERNAL_STORAGE WRITE_EXTERNAL_STORAGE READ_MEDIA_AUDIO READ_MEDIA_IMAGES READ_MEDIA_VIDEO

2.2. APK file

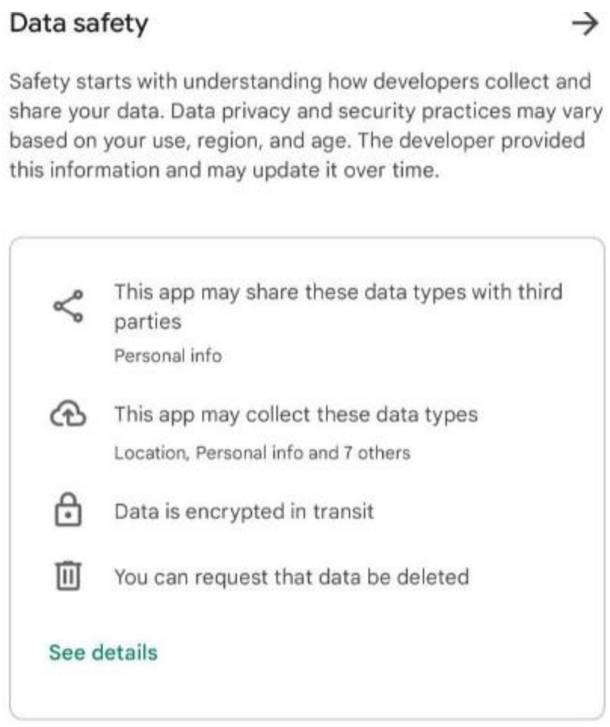
Android uses a file format called APK. Android apps are created by compiling APK files using Android Studio, the official development environment. The APK files contain all the code and assets for the software program [12]. The metadata of APK files, including permissions and manifest information, is often targeted by Android malware. Various types of Android malware exhibit differing attack capabilities and possess distinct features that facilitate their classification and identification [13].

2.3. Data safety

Data Safety is one of the information that is required from Android developers on the app’s listing page. On July 20, 2022, Google’s policy stated that all developers must disclose how they collect and use users’ data by completing the Data Safety form on the Play Console page [7].

Developers must ensure their apps have accurate and complete Data Safety forms, including a link to their privacy policy [3]. While Google Play reviews apps for policy compliance, it cannot determine how developers handle sensitive user data. Google may take enforcement action if discrepancies are found between an app’s behavior and its privacy policy declaration [2].

To publish an app on Google Play, developers must complete the Data Safety form, even if the app doesn’t collect personal user data. The Data Safety section includes details on data collection, sharing, and other security practices, as shown in Fig. 1.



Data safety

Safety starts with understanding how developers collect and share your data. Data privacy and security practices may vary based on your use, region, and age. The developer provided this information and may update it over time.

This app may share these data types with third parties
Personal info

This app may collect these data types
Location, Personal info and 7 others

Data is encrypted in transit

You can request that data be deleted

See details

Fig. 1. Data Safety section

3. Related works

Several studies have compared the actual behaviors of Android apps with the information available on Google Play's app listing page, focusing on aspects such as the privacy policy, app description, and Data Safety section.

Several studies have focused on detecting inconsistencies between privacy policies and actual app behaviors, such as [5, 14-17]. Zimmeck et al. [14] and Slavin et al. [15] employed analogous methodologies to identify privacy policy violations in Android apps by analyzing Android API calls. This approach proved valuable in the detection of instances where API calls acquired personal data from mobile devices. Different from the aforementioned methods, which primarily emphasized privacy information obtained from API methods, Wang et al. [16] developed GUILeak. It is a novel framework capable of detecting violations relating to information collected from Graphical User Interfaces (GUIs). By establishing correlations between privacy-policy phrases and user input views, GUILeak surpassed the confines of Android API-based violation detection, enabling the identification of potential violations related to user input. The PermPress tool developed by Rahman et al. [5] focused on evaluating the comprehensiveness of permissions in Android apps. It specifically evaluated whether an app's privacy policy accurately reflected its dangerous permissions. This has been achieved through a combination of machine learning techniques and human annotation of privacy policies, enabling the identification of permission-related information. In contrast, the PTPDroid tool developed by Tan and Song [17] has been designed to detect privacy disclosures to third parties in Android apps. It employed an entity-sensitive flow-to-policy consistency checking technique to identify violated privacy disclosures. PTPDroid utilized static analysis to analyze data flows within the app's code and applied natural language processing to extract relevant declarations from the privacy policy. By categorizing the identified data flows into different disclosure groups, PTPDroid provided insights into the clarity and accuracy of privacy disclosures.

Numerous studies have highlighted discrepancies between app descriptions and the permissions requested, such as [18-20]. Researchers have used deep learning and Natural Language Processing (NLP) to address this issue. For example, Feng et al. [18] have proposed AC-Net, an end-to-end framework that assessed the consistency between app descriptions and permissions. Feichtner and Gruber [19] have applied deep learning and NLP to design a Convolutional Neural Network (CNN) for text classification, identifying significant words and phrases related to dangerous permissions. This CNN also predicted whether apps required specific permissions and flagged descriptions involving sensitive data or system features not mentioned textually. Meanwhile, Wu, Chen and Lee [20] employed the Fidelity Calculation for Description-to-Permissions (FCDP) approach, using quantitative metrics to predict potential permission requests and found correlations between requested and predicted permissions in the source code.

To date, to the best of our knowledge, no research has compared the Data Safety section with an app's actual behavior. However, Khandelwal et al. [21] have conducted a comprehensive study of Google's Data Safety section using a mixedmethods approach, including both quantitative and qualitative methods. They reached out to developers via email to gain insights into their disclosure practices, revealing significant inconsistencies between data collection and sharing practices. In contrast, Khandelwal et al. [22] compare privacy practices in privacy labels with those in privacy policies to assess consistency, aiming to evaluate how accurately developers disclosed their practices across Apple and Android platforms. For Android, privacy labels refer to the Data Safety section.

4. Methodology

This section outlines the approach used to detect inconsistencies between data collection information in the Data Safety section and the actual data collection practices of Android apps. An overview of the approach is shown in Fig. 2, consisting of four parts, each detailed in separate subsections. The first part covers the process of collecting a dataset of Android apps' APKs and their Data Safety information from Google Play. The second part explains how data collection-related details were extracted from the Data Safety section. The third part focuses on analyzing the app source code to extract dangerous permissions and other indicators of data collection. The fourth part discusses how the extracted source code information is mapped to the Data Safety data to detect inconsistencies. The procedures have been carried out in the Google Collaboratory environment, selected for its support of various libraries and ample storage capacity.

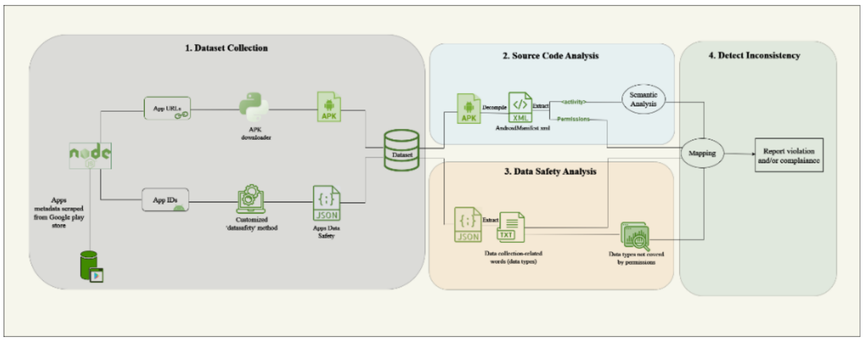


Fig. 2. Methodology for assessing the consistency between Data Safety and APK

4.1. Dataset collection

This section describes the process of building a dataset of Android app APK files and their corresponding Data Safety information. Google-Play-Scraper [23], a Node.js library, was used to crawl metadata from over 5000 apps across various Google Play categories. The List.js method was modified to iterate through all app categories and extract only app IDs and URLs. The results were manually inspected to ensure data relevance.

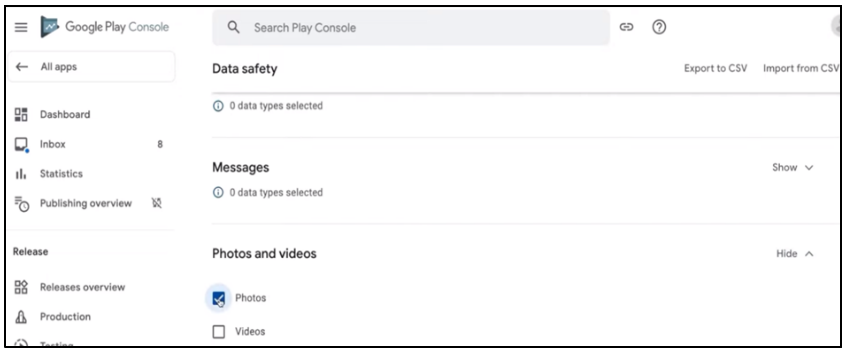
In addition to List.js, the datasafety.js method was used to retrieve Data Safety information for each app. It takes an app ID as input and returns the details in JSON format. To improve efficiency, the script was customized for automation, allowing it to process the extracted app IDs and save the Data Safety information automatically, eliminating the need for manual execution.

To enable static analysis, a Python script was developed to retrieve and download the latest APK versions from APKPure, a website offering smartphone software downloads [24]. The script automated the process, ensuring the systematic acquisition of APK files for all applications in the dataset.

4.2. Extraction of data collection-related statements

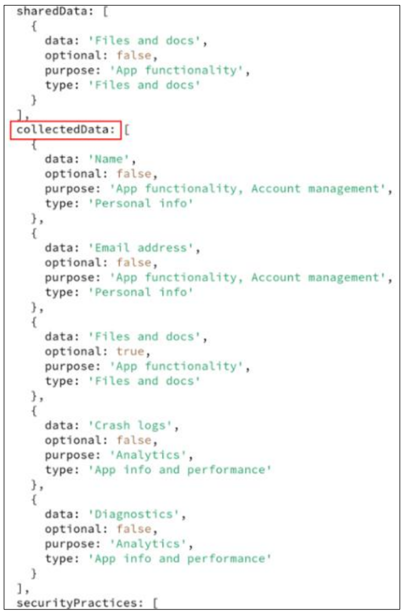
An app's Data Safety information is a standardized form filled out by developers [3], where they select the types of data their apps collect by ticking corresponding checkboxes, as shown in Fig. 3. Once published on the Google Play store, the Data Safety section appears to users in the same standardized format (Fig. 1).

If two apps collect the same data, their Data Safety sections will be identical, allowing for effective word-matching to extract data collection details.



[✓] Photos
[ ] Videos

Fig. 3. Data Safety form filling by developers



```
sharedData: [
  {
    data: 'Files and docs',
    optional: false,
    purpose: 'App functionality',
    type: 'Files and docs'
  }
],
collectedData: [
  {
    data: 'Name',
    optional: false,
    purpose: 'App functionality, Account management',
    type: 'Personal info'
  },
  {
    data: 'Email address',
    optional: false,
    purpose: 'App functionality, Account management',
    type: 'Personal info'
  },
  {
    data: 'Files and docs',
    optional: true,
    purpose: 'App functionality',
    type: 'Files and docs'
  },
  {
    data: 'Crash logs',
    optional: false,
    purpose: 'Analytics',
    type: 'App info and performance'
  },
  {
    data: 'Diagnostics',
    optional: false,
    purpose: 'Analytics',
    type: 'App info and performance'
  }
],
securityPractices: [
```

Fig. 4. Result of Google-Play-Scrapper

The Data Safety information retrieved from the scraper, as detailed in Section 4.1, is presented in JSON format, containing data collection, sharing, security practices, and a privacy policy link (Fig. 4). Since this study focuses on data collection, a Node.js script was developed to extract the “collectedData” object and relevant data types from the “data” key. This process resulted in a structured extraction of data collection-related terms from all apps in the dataset.

4.3. Source code analysis

To identify user data collection practices in Android apps, it is necessary to determine the permissions they request [5]. These permissions are found in the AndroidManifest.xml file within the APK. Since APK files are Java Archive (JAR) packages, they must be decompiled to access their contents. Using APKtool, a reverse engineering tool for Android apps [25], the downloaded APKs were decompiled to extract the AndroidManifest.xml files.

By the time of writing this paper, Android’s official documentation defined 40 dangerous permissions [11], grouped into 15 permission categories. This study considered 25 dangerous permissions across 10 groups, focusing on those directly related to user data collection. For example, POST NOTIFICATIONS, a dangerous permission that allows apps to post notifications but does not collect user data, was excluded. Table I lists the considered permissions, and any reference to “dangerous permission” in this paper pertains to these unless otherwise stated.

To extract dangerous permissions from AndroidManifest.xml, a script was developed to target the uses-permission tag. Fig. 5 illustrates how permissions appear in the file. If the script detects a dangerous permission, it assigns it a value of 1; otherwise, it assigns 0. For instance, if the script detects the tags shown in Fig. 5, it will return the following:

- RECORD_AUDIO = 1
- ACCESS_FINE_LOCATION = 1
- WRITE_EXTERNAL_STORAGE = 1
- READ_SMS = 1
- and for all the remaining permissions, it will assign 0.

```xml
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="android.permission.SET_WALLPAPER" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_SMS" />
```

Fig. 5. Snapshot of AndroidManifest.xml

WAKE LOCK and SET WALLPAPER were excluded as they are normal permissions and not classified as dangerous.

The extracted permissions can help detect inconsistencies in data collection-related information within the Data Safety section, as discussed in Section 4.4. However, these permissions alone do not fully capture all aspects of data collection relevant to the Data Safety section. This limitation arises because not all data types defined by Google for Data Safety are collected solely through permission grants [26]. For instance, no specific permissions exist for data categories such as "political or religious beliefs," which are typically gathered through user input forms or other interactive features within an app.

An app's source code encapsulates essential components for its functionality, including permissions, activities, and executed actions [27]. Each window displayed by an app (e.g., a "Sign In" screen) is represented by a ⟨activity⟩ tag in the source code [28]. Developers assign distinct Java class names to these activities, making it possible to predict which windows will be displayed to users. This approach facilitates the extraction of information related to data collection. Based on this, all ⟨activity⟩ declarations were automatically extracted from each app in the dataset for subsequent semantic analysis.

4.4. Detecting inconsistency

The findings from Sections 4.2 and 4.3 were used to identify inconsistencies between an app's data collection practices and the data collection-related information in the Data Safety section. These findings informed the creation of a mapping guide linking dangerous permissions to corresponding Data Safety keywords, as shown in Table 2. The verification process first identifies dangerous permissions assigned a value of "1" and then checks for the presence of relevant Data Safety keywords in the extracted data collection information. If no corresponding keyword is found for any permission marked as "1," a violation is reported. If at least one permission violation occurs, the app is classified as inconsistent.

Google categorizes all Data Safety data types into 14 distinct groups [3]. The mapping to dangerous permissions verified most of these data types; however, a subset spanning six Data Safety categories remained unverified. To assess the consistency of these unverified data types, a semantic analysis was performed on the extracted ⟨activity⟩ tag declarations.

Since ⟨activity⟩ tag declarations do not always correspond directly to a specific data type, they were instead analyzed to determine whether they aligned with any of the six unverified Data Safety categories. To facilitate this, Google's definitions of data types [3] were compiled into six category-specific texts. These texts were tokenized using the Natural Language Toolkit (NLTK) [29], producing distinct word sets for each category. For example, the "financial info" category consists of four data types defined by Google [3]:

- **User payment info.** Details about financial accounts, such as credit card numbers.
- **Purchase history.** Records of user transactions.
- **Credit score.** Information regarding a user's credit score.
- **Other financial info.** Additional financial details, such as salary or debts.

Table 2. Data safety keywords that point to dangerous permissions

Dangerous permissions	Relevant Data Safety keywords
ACCESS_ FINE_ LOCATION	Precise location
ACCESS_ COARSE_ LOCATION	Approximate location
GET_ ACCOUNTS	User IDs
READ_ MEDIA_ IMAGES	Photos
READ_ MEDIA_ VIDEO	Videos
ACCESS_ MEDIA_ LOCATION	Photos-Videos-Precise location-Approximate location
BODY_ SENSORS	Health info
ACTIVITY_ RECOGNITION	Fitness info
READ_ CALENDER	Calendar events
WRITE_ CALENDER	
READ_ CONTACTS	Contacts
WRITE_ CONTACTS	
READ_ SMS	Contacts-SMS or MMS
SEND_ SMS	
RECEIVE_ SMS	
RECEIVE_ MMS	
RECEIVE_WAP PUSH	
READ_ PHONE_ NUMBERS	Phone numbers-Contacts
READ_ CALL_ LOG	
WRITE_ CALL_ LOG	
READ_ PHONE_ STATE	Phone numbers-Contacts-Device or other IDs
READ_ MEDIA_ AUDIO	Voice or sound recordings-Music files-Other audio files
RECORD_ AUDIO	Voice or sound recordings-Music files-Other audio files
READ_ EXTERNAL_ STORAGE	Files and docs-Voice or sound recordings-Music files-Other audio files-Photos-Videos
WRITE_ EXTERNAL_ STORAGE	Files and doc-Voice or sound recordings-Music files-Other audio files-Photos-Videos

The definitions of data types were compiled into a single text to create a reference for each category. This compiled text was then tokenized using NLTK to extract distinct words relevant to the category.

The choice of using data type definitions as a reference for categories was driven by the intent to narrow down to the specific meanings of category names and avoid generality.

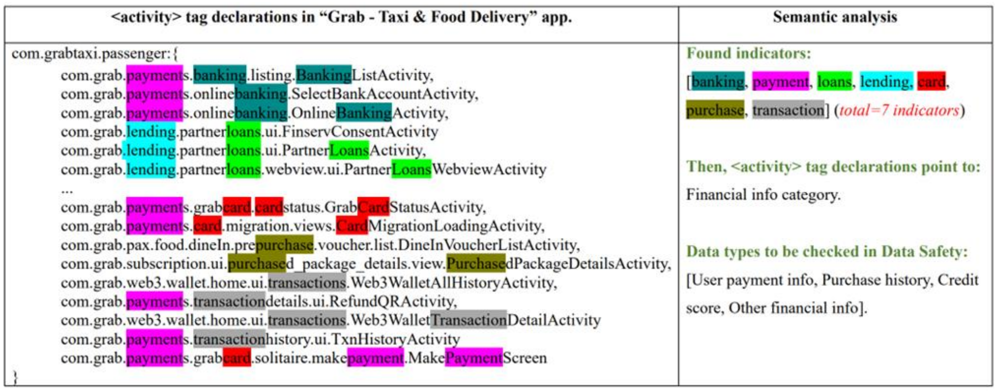
To enhance the accuracy of identifying relevant terms, Word2Vec [30] was applied to generate synonyms for each word extracted by NLTK. These synonyms acted as indicators for detecting whether an app's ⟨activity⟩ tag declarations were related to a specific Data Safety category.

To ensure reliability, an ⟨activity⟩ tag declaration was considered linked to a Data Safety category only if at least six indicators from the category's word set were present within the extracted declarations. If the threshold was met, all data types under that category were then checked against the app's reported Data Safety section.

Fig. 6 provides a visual representation of the extracted ⟨activity⟩ tag declarations and how the proposed semantic analysis method operates. The threshold of six indicators was determined based on manual inspection of a representative sample from the dataset. This inspection aimed to establish a reasonable, credible, and logical threshold to infer a category's presence within the app's functionality.

Once a category was identified, the corresponding data types were verified in the Data Safety section. If at least one data type under the category was correctly reported, the category was deemed consistent with the actual data collection practices. However, if **none** of the data types under the identified category were found in the Data Safety section, a **violation** was recorded, classifying the app as **inconsistent**. This can be illustrated in our example presented in Fig. 6 as follows:

- Indicators found in ⟨activity⟩ tag declarations point to financial info.
- The Financial info category belongs to the set of categories not covered by permissions.
- Total number of the found indicators = 7 (which exceeds the defined threshold).
- A check for the existence of "User payment info, "Purchase history", Credit score, or Other financial info" in the Data Safety has been done. If none of the data types are found to be reported in the Data Safety, then the app collection practices for data under the "financial info" category are considered inconsistent with actual collection practices.



<activity> tag declarations in "Grab - Taxi & Food Delivery" app.	Semantic analysis
com.grabtaxi.passenger:{	**Found indicators:**
com.grab.payments.banking.listing.BankingListActivity,	[banking, payment, loans, lending, card,
com.grab.payments.onlinebanking.SelectBankAccountActivity,	purchase, transaction] (*total=7 indicators*)
com.grab.payments.onlinebanking.OnlineBankingActivity,	
com.grab.lending.partnerloans.ui.FinservConsentActivity	
com.grab.lending.partnerloans.ui.PartnerLoansActivity,	**Then, <activity> tag declarations point to:**
com.grab.lending.partnerloans.webview.ui.PartnerLoansWebviewActivity	Financial info category.
...	
com.grab.payments.grabcard.cardstatus.GrabCardStatusActivity,	
com.grab.payments.card.migration.views.CardMigrationLoadingActivity,	
com.grab.pax.food.dineIn.prepurchase.voucher.list.DineInVoucherListActivity,	**Data types to be checked in Data Safety:**
com.grab.subscription.ui.purchased_package_details.view.PurchasedPackageDetailsActivity,	[User payment info, Purchase history, Credit
com.grab.web3.wallet.home.ui.transactions.Web3WalletAllHistoryActivity,	score, Other financial info].
com.grab.payments.transactiondetails.ui.RefundQRActivity,	
com.grab.web3.wallet.home.ui.transactions.Web3WalletTransactionDetailActivity	
com.grab.payments.transactionhistory.ui.TxnHistoryActivity	
com.grab.payments.grabcard.solitaire.makepayment.MakePaymentScreen	
}	

Fig. 6. Semantic analysis of tag declarations of Grab – Taxi & Food Delivery app

5. Results

Out of the +5000 collected apps, 4980 APKs were successfully downloaded. Fig. 7 illustrates that only 3768 (75.7%) of the downloaded apps reported data collection in Data Safety, constituting the set that will be evaluated in this section. The remaining 1212 (24.3%) apps were excluded from the study. 1222 (32.3%) of the evaluated apps reported their data collection in Data Safety consistent with the actual collection practices extracted from the source code. Among these 1222 apps, 8.7% were found to be over-reporting. In this context, over-reporting refers to situations where apps report the collection of a data type in Data Safety while not requesting the corresponding permission. Despite the fact that 8.7% of consistent apps represent a relatively small number in the dataset, it still has the potential to affect users' trust in Data Safety.

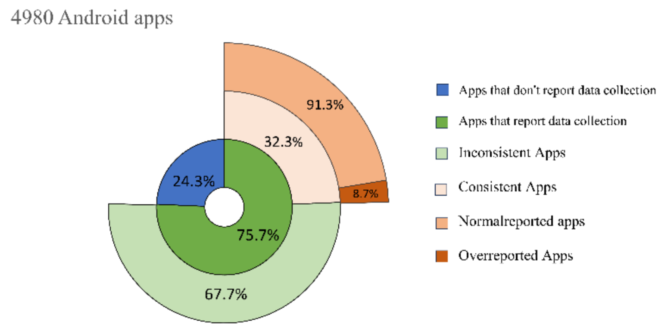


Fig. 7. Percentage of apps reporting data collection in Data Safety consistently/inconsistently with actual data collection (The Two Inner Layers). Percentage of consistent apps that over-report data collection (Outer Layer)

Fig. 7 also shows that 2546 (67.7%) of evaluated apps were found to be inconsistent. As previously explained, the proposed methodology deems an app inconsistent if at least one permission or at least one category is found to be in violation. To delve deeper into the reported inconsistencies, Fig. 8 illustrates the violation rate per distinct permission. Permissions within the STORAGE permission group show the highest rate of violations. Specifically, WRITE_EXTERNAL_STORAGE and READ_EXTERNAL_STORAGE permissions as they mapped to the largest number of data types across different categories within Data Safety. Sequentially, permissions within the LOCATION permission group indicate the second-highest rate of violations. This suggests that developers can access sensitive data, including photos, videos, files, location, etc., without properly reporting their collection practices in the Data Safety. Thereby, users may be misled into believing that these apps do not pose a threat to their privacy. Moreover, a noticeable gap was observed between permissions belonging to the SMS and CALL._LOG permission groups and other permissions. The violation rates of permissions belonging to the SMS and CALL_LOG permission groups range between 0%-0.32%. At the same time, other permissions violation rates range gradually between 1.09%-18.36%. When a violation rate is low, that means either the permission is consistent with Data Safety or the permission is not requested in the first place. The low violation rate of the permissions belonging to the SMS and CALL_LOG is mostly because they are not frequently requested, as discussed later in the Discussion section.

Similarly, Fig. 8 shows the violation rate per distinct Data Safety category. The high violation rates were observed in the Audio files, Photos and videos, and Files and docs categories. On the other hand, Health and fitness, Device or other IDs, and Personal info categories have lower violation rates, indicating their alignment with the actual collection practices of the apps. For further examination, Table 3 enumerates the total number of detected violations related to both sets of data types covered and not covered by dangerous permissions. It clearly states that most violations (66%) were detected during the consistency verification between requested dangerous permissions and relevant data types in Data Safety. At the same time, only 34% of the violations pertain to data types not covered by dangerous permissions.

Table 3. Total number of detected violations

Number	Data types covered by dangerous permissions	Data types not covered by dangerous permissions
Number of detected violations (10,444)	6895 (66%)	3549 (34%)

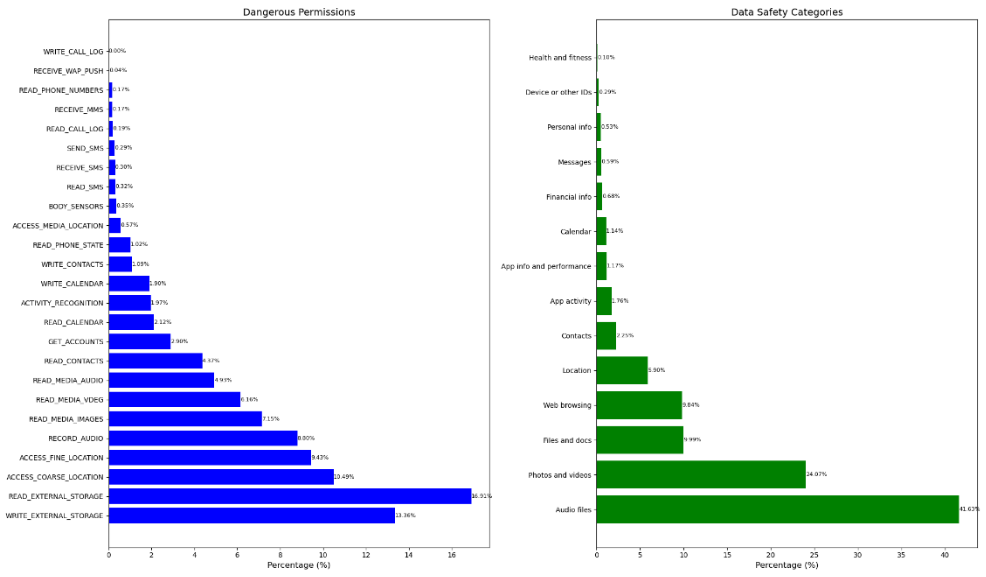


Fig. 8. Number of violations per dangerous permissions and Data Safety category

6. Discussion

This research detects violations in data collection-related information within Google Play’s Data Safety section when compared to the source code of Android apps. The paper employs static analysis of the source code of Android apps to extract dangerous permissions and other indicators related to the user’s data collection. While the majority of data types in Data Safety were validated by mapping them to dangerous permissions, there was a subset of data types that remained unverified by the extracted dangerous permissions. To address them, a semantic analysis was performed on the ⟨activity⟩ tags declarations extracted from the source code to map the result to those data types.

The proposed methodology reveals that 67% of the examined apps show inconsistencies in the data collection-related information within the Data Safety section when compared to the source code. This percentage is notably similar to and slightly higher than the findings of K h a n d e l w a l et al. [22] and K h a n d e l w a l et al. [21]. In their respective studies, 55% of 346K and 41% of 539K apps had violations in Data Safety when compared against the privacy policies and developers' claims, respectively. According to K h a n d e l w a l et al. [22], the most inconsistent categories with privacy policies were Personal info and App activity, with rates of 66% and 67%, respectively. Previous studies have also shown violations when comparing privacy policies with source code [14-16]. This research reveals that Audio files, Photos and videos, and Files and docs are among the most inconsistent categories when compared to the app source code. Since our approach relies on mapping data types to their corresponding permissions, it provides more accurate verification when compared to prior works that rely on privacy policies or interviewing developers.

Previous studies by F e n g et al. [18] and W u, C h e n and L e e [20] identified that the most inconsistent permissions, when compared with app descriptions, were from the LOCATION, SMS, and STORAGE permission groups. Similarly, R a h m a n et al. [5] found that the most inconsistent permissions, when compared to privacy policies, belonged to the STORAGE and CONTACTS permission groups. Our results, illustrated in Fig. 8, mirror these findings. Permissions from STORAGE, LOCATION, and CONTACTS permission groups have a relatively high violation rate when compared to Data Safety. This implies that developers tend not to reveal the privacy practices of such permissions. Thus, users who trust Data Safety to guide their decisions on app downloads may be misled into believing that their privacy is safeguarded.

Fig. 8 also shows that there is a notable disparity in the violation rate of the permissions belonging to SMS and CALL_LOG groups compared to other permissions. At the end of 2018 [31], Google classified permissions belonging to SMS and CALL_LOG groups have restricted permissions, resulting in reduced requests for these permissions. Google announced that apps not qualified for access to these permissions must be removed from their manifest within 90 days following the announcement. The variance in our findings, compared to those of F e n g et al. [18], regarding these permission groups may be attributed to the temporal proximity of their study conducted in 2019 with the Google announcement.

7. Conclusion

This study aims to detect the inconsistency between the data collection-related information within the Data Safety section and actual data collection practices in the app's source code to ensure alignment and accuracy between the stated information and the implemented practices. Following the proposed method, 4980 Android apps' APKs were downloaded and compared against data collection-related information in the Data Safety section. The comparison was conducted through an analysis of the source code to extract the dangerous permissions and activity declarations and map them to the relevant Data Safety statements. The results indicate that 32.3% of the apps' actual collection practices are consistent with the Data Safety section, whereas a significant 67.7% of the analysed apps are inconsistent with the Data Safety section.

The approach implemented in this paper is subject to certain limitations. In particular, the methodology employed in this study focuses on static analysis of the source code, disregarding runtime behavior and user interactions. Furthermore, this study solely focuses on data collection within Data Safety. Future work may focus on other key sections, such as data sharing and handling.

8. References

1. B i l a l, A., H. T. M i r z a, I. H u s s a i n, A. A h m a d. Investigating Influence of Google-Play. Application Titles on Success. – ACM Digital Library, Vol. **36**, 2024, No C, p. 302.
2. Google Play Developer Help Community. Managing and Reporting Police Violations. Google Play. Online. Accessed 14-May-2023. **https://support.google.com/googleplay/androiddeveloper/answer/9899142?hl=en**
3. Google Play. Provide Information for Google Play's Data Safety Section. Google Play, 14 December 2021. Online. Accessed 14-May-2023. **https://support.google.com/googleplay/androiddeveloper/answer/10787469?hl=en**
4. Mozilla. Mozilla Study: Data Privacy Labels for Most Top Apps in Google Play Store are False or Misleading. Mozilla, 23 February 2023. **https://foundation.mozilla.org/en/blog/mozilla-study-data-privacy-labels-for-most-top-apps-in-google-play-store-are-false-or- misleading/**
5. R a h m a n, M., P. N a g h a v i, B. K o j u s n e r, S. A f r o z, B. W i l l i a m s, S. R a m-p a z z i, V. B i n d s c h a e d l e r. Permpress: Machine Learning-Based Pipeline to Evaluate Permissions in App Privacy Policies. IEEE, 2022, p. 22.
6. Android Developers. App Manifest Overview. Online. Accessed 27-April-2023. **https://developer.android.com/guide/topics/manifest/manifest-intro**
7. Y a n g, X., X. Z h a n g. A Study of User Privacy in Android Mobile AR Apps. – In: Proc. of 37th IEEE/ACM International Conference on Automated Software Engineering, 2022.
8. A l m o m a n i, I. M., A. A. K h a y e r. A Comprehensive Analysis of the Android Permissions System. – IEEE Access, Vol. **8**, 2020, pp. 216671-216688. DOI:10.1109/access.2020.3041432.
9. Android Developers. Permissions on Android. Online. Accessed 19 May 2023. **https://developer.android.com/guide/topics/permissions/overview**
10. K h a t o o n, A., P. C o r c o r a n. Android Permission System and User Privacy – A Review of Concept and Approaches. – In: 7th IEEE International Conference on Consumer Electronics – Berlin (ICCE-Berlin'17), 2017. DOI:10.1109/icce-berlin.2017.8210616.
11. Android Developer. Manifest.permission. Online. Accessed 29 May 2023. **https://developer.android.com/reference/android/Manifest.permission**
12. G i l l i s, A. S. What Is an APK File (Android Package Kit File Format)?: Definition from TechTarget. WhatIs.com. Online. Accessed May 19, 2023. **https://www.techtarget.com/whatis/definition/APK-file-Android-Package-Kit-file-format**
13. N w a s r a, N., M. D a o u d, Z. H. Q a i s a r. ANFIS-AMAL: Android Malware Threat Assessment Using Ensemble of ANFIS and GWO. – Cybernetics and Information Technologies, Vol. **24**, 2024, No 3, pp. 39-58.
14. Z i m m e c k, S., Z. W a n g, L. Z o u, R. I y e n g a r, B. L i u, F. S c h a u b, J. R e i d e n b e r g. Automated Analysis of Privacy Requirements for Mobile Apps. – In: 2016 AAAI Fall Symposium Series, 2016.
15. S l a v i n, R., X. W a n g, M. B. H o s s e i n i, J. H e s t e r, R. K r i s h n a n, J. B h a t i a, T. B r e a u x, J. N i u. Toward a Framework for Detecting Privacy Policy. – In: Proc. of 38th International Conference on Software Engineering, 2016, pp. 25-36.
16. W a n g, X., X. Q i n, M. H o s s e i n i, R. S l a v i n, T. B r e a u x, J. N i u. GUILeak: Tracing Privacy Policy Claims on User Input Data for Android Applications. – In: Proc. of 40th International Conference on Software Engineering, 2018, pp. 37-47.

17. T a n, Z., W. S o n g. PTPDroid: Detecting Violated User Privacy. – In: Proc. of 45th International Conference on Software Engeneering (ICSE), IEEE, 2023, pp. 473-485. DOI: 10.1109/ICSE48619.2023.00050.
18. F e n g, Y., L. C h e n, A. Z h e n g, C. G a o, Z. Z h e n g. AC-Net: Assessing the Consistency of Description and Permission in Android Apps. – IEEE Access, Vol. **7**, 2019, pp. 57829-57842.
19. F e i c h t n e r, J., S. G r u b e r. Understanding Privacy Awareness in Android App Descriptions Using Deep Learning. – In: Proc. of 10th ACM Conf. Data Appl. Secur. Priv. (CODASPY'20), 2020, pp. 203-214. DOI: 10.1145/3374664.3375730.
20. W u, Z., X. C h e n, S. U. J. L e e. FCDP: Fidelity Calculation for Description-to-Permissions in Android Apps. – IEEE Access, Vol. **9**, 2021, pp. 1062-1075. DOI: 10.1109/ACCESS.2020.3047019.
21. K h a n d e l w a l, R., A. N a y a k, P. C h u n g, K. F a w a z. Unpacking Privacy Labels: A Measurement and Developer Perspective on Google's Data Safety Section. – arXiv Preprint arXiv:2306.08111, 2023, p. 25.
22. K h a n d e l w a l, R., A. N a y a k, P. C h u n g, K. U. F a w a z. The Overview of Privacy Labels and Their Compatibility with Privacy Policies. – ArXiv.Org, 2023. **https://arxiv.org/abs/2303.08213.**
23. O l a n o, F. F a c u n d o o l a n o. – GitHub, 4.2.2019. **https://github.com/facundoolano/google-play-scraper**.
24. APKPure.com. AboutUs. Online. Accessed 19 May 2023. **https://m.apkpure.com/ar/about.html.**
25. Apktool. Apktool – A Tool for Reverse Engineering 3rd Party, Closed, Binary Android Apps. **https://ibotpeaches.github.io/Apktool/.**
26. Google Play Help. Understand App Privacy Security Practices with Google Play's Data Safety Section. Online. Accessed 5 Juny 2023. **https://support.google.com/googleplay/answer/11416267?sjid=2407870662 863064307-EU#data-collection&zippy=%2Cdata-collection.**
27. Appdome. Structure of an Android App Binary (.apk). 9 Aug 2022. Online. Accessed 4 Juny 2023. **https://www.appdome.com/how-to/appsec-release-orchestration/appdome-basics/structure-of-an-android-app-binary-apk/.**
28. Android Developers. Introduction to Activities. Online. Accessed 4 Juny 2023. **https://developer.android.com/guide/components/activities/intro- activities**
29. B i r d, S., E. L o p e r, E. K l e i n. Natural Language Processing with Python. O'Reilly Media Inc., 2009.
30. TensorFlow. Word2Vec. TensorFlow Text Tutorials. Online. Accessed 13 November 2023. **https://www.tensorflow.org/text/tutorials/word2vec**
31. Play Console Help. Use of SMS or Call Log Permission Groups. **https://support.google.com/googleplay/androiddeveloper/answer/10208820?sjid8311827 649165247607-EU**

Received: 06.12.2025, Revised Version: 10.02.2025, Accepted: 18.02.2025