A Systematic Evaluation of Non-SDK Interface Restrictions in Android: Bridging the Gap Between Guidelines and Practice

George Silva, Norah Ridley(✉), Enrico Branca, and Natalia Stakhanova

University of Saskatchewan, Saskatchewan, Canada
norah.ridley@usask.ca

Abstract. Android non-SDK interfaces are APIs that are not part of the official SDK and are restricted. Multiple studies have indicated flaws and limitations of the usage of these non-SDK interfaces, prompting Google to introduce restrictions on non-SDK interfaces to regulate access to these interfaces. This study systematically evaluates the alignment between the official Android guidelines for non-SDK interface usage and the findings from Veridex, a Google tool that assesses the existence of these non-SDK interfaces in Android applications. Our analysis considers the three latest Android versions and reveals inconsistencies, including mismatches in non-SDK interfaces and associated restrictions, as well as contradictions in the enforcement of these restrictions. These inconsistencies highlight significant challenges in the regulatory framework, potentially undermining the effectiveness of measures intended to secure the Android platform.

Keywords: Android · Non-SDK interfaces

1 Introduction

Android is one of the most widely used mobile operating systems (OS) in the world. The convenient and extensive access to phone resources adopted by Android has quickly revealed inefficiencies of its existing protections. Android non-SDK interfaces, or APIs that are not available for third-party developers, are a part of these issues. These interfaces are not included in the official Android Software Development Kit (SDK). Non-SDK interfaces, also referred to as hidden APIs, are also a part of the Android framework. These interfaces are typically used internally by the operating system developers to ensure access to deeper system-level functionalities that are not available through the standard SDK. Unlike public SDK interfaces, which are well-documented and supported by Google, non-SDK interfaces are not intended for third-party app development. They are not publicly documented or supported for general use, and they can

K. Adi et al. (Eds.): FPS 2024, LNCS 15532, pp. 217–234, 2025.
https://doi.org/10.1007/978-3-031-87499-4_14

change or be removed without notice in future Android releases. Thus, their stability and security are not guaranteed, and their usage by third-party developers can lead to compatibility and security issues as the Android platform evolves.

Google began imposing restrictions on the use of non-SDK interfaces with the introduction of Android 9, aiming to regulate access to critical parts of the Android platform for applications and gradually block non-SDK interface use. These restrictions have been progressively tightened in subsequent versions. Several studies explored the use of non-SDK interfaces [21] and internal APIs [9] in Android apps before and after Google introduced access restrictions on non-SDK interfaces. At that time both studies noted that developers tend to ignore Googles access control restrictions. Follow-up studies noted gaps in permissions' documentation and their contradicting definitions [1] which lead to ambiguity, increasing the risk of misuse and numerous security issues [7,21].

To help restrict the use of non-SDK interfaces in third-party apps, Google provides several mechanisms to detect their presence. Among them is Veridex, an official static analysis tool maintained by Google. Veridex relies on the official restriction lists, serving to guide users in using restriction lists. While Veridex is widely employed for analysis, its effectiveness and alignment with Googles restrictions have not been systematically evaluated.

In this study, we focus on the consistency and reliability of non-SDK interfaces' analysis. Specifically, we *systematically investigate and measure the discrepancies between the documented non-SDK interfaces in the restriction lists, which serve as the official Google documentation of existing interfaces, and their actual usage in Android apps as detected by Veridex.* To achieve a comprehensive understanding of these non-SDK interfaces, we parse the official Android restriction lists to extract SDK and non-SDK interfaces and their restrictions for the latest three Android versions (12, 13 and 14). To further understand the use of these interfaces in practice, we analyze 714,616 Android apps collected from 18 sources and their use of non-SDK interfaces across the Android versions 12, 13 and 14. In summary, our contributions are as follows:

– *Systematic analysis of misalignment of Veridex results with the official restriction lists.* We studied 47,967,468 non-SDK interfaces identified by Veridex in the analyzed apps across Android versions 12 to 14. Our findings revealed a significant number of inconsistencies between Veridex's output and the official restriction lists, i.e., 72.13% of used calls were found to be inconsistent with official information in version 12, 69.95% were found in version 13, and 70.62% in version 14. Among them, we discovered 96 non-SDK interfaces that were absent from the restriction lists, but reported by Veridex. These interfaces were linked to core Android functionalities including widget, OS, and content as well as accessing JDK internal functionalities. The presence of these non-SDK interfaces poses significant security risks by exposing critical system components and sensitive operations to unauthorized usage.
  We further identified significant discrepancies between interface restrictions reported by Veridex and the restrictions indicated in the restriction lists. In version 12, 34.39% showed mismatched restrictions, including 89 interfaces

with no corresponding restrictions. In version 13, 41.26% of interfaces had mismatched restrictions with 53 interfaces lacking corresponding restrictions. Similarly, version 14 displayed a 39.62% mismatch rate, with 87 interfaces completely unmatched. This pattern highlights challenges in aligning detected non-SDK interfaces with their documented restrictions, complicating efforts to ensure secure and compliant application development.
To the best of our knowledge, we are the first to discover the discrepancies in the official guidance for the non-SDK usage and conduct a systematic study of the discrepancies between the documented interfaces in the restriction lists and their actual usage in Android apps as detected by Veridex.

– *Large-scale analysis and measurement of non-SDK interfaces present in third-party apps.* Our study examined 714,616 apps and identified the use of non-SDK interfaces in 85.79% of them. Compared to earlier study by Yang et al. [21] that investigated the use of non-SDK interfaces right after they were deployed by Google (i.e., Android 9-11) on a small set of apps, their usage remains consistently high, i.e., 88% of 66,450 apps [21].
– *Improved Veridex tool.* We identified and corrected Veridex limitation and provide the improved version to the community[1].

2 Related Work and Background

Background. To control the use of non-SDK interfaces, Google has introduced a more formal restriction process and published the restriction lists for each major version of Android since Android 9 [4]. Each restriction list includes the SDK and non-SDK interfaces and the corresponding restriction categories. Officially, the restriction lists include the following categories: ① *Blocklist:* indicates interfaces that are inaccessible to third-party developers. ② *Conditionally blocked (max-target-x/greylist-max-x):* interfaces that are accessible by apps targeting an API level up to level *x*. ③ *Unsupported (greylist):* unrestricted but not included in official documentation and are therefore subject to change without notice. Google specifically states that these interfaces will be conditionally blocked in future Android versions. ④ *SDK (whitelist):* - documented and supported interfaces. ⑤ *Test APIs (test-API):* interfaces that are used for internal system testing. These are not open for third-party developers and are effectively blocked starting in Android 11.

Related Work. The landscape of Android API usage and security has been thoroughly explored, revealing challenges associated with rapid platform evolution and the resulting compatibility issues [2,6,8,13,20]. Syer et al. [19] highlighted the risks associated with platform-dependent code. Both Li et al. [8] and Linares-Vasquez et al. [11] noted that rapidly changing APIs can cause several issues for third-party applications. Liu et al.'s [12] results revealed that the

[1] https://github.com/nonsdk/revised_veridex.

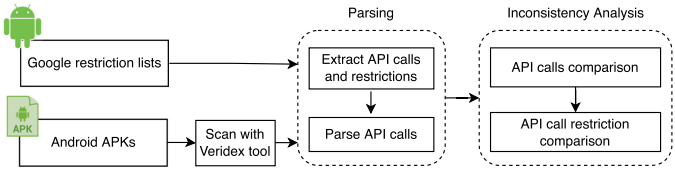

Fig. 1. The flow of the analysis

majority of third-party applications use silently evolving interfaces that lack up-to-date documentation. Other studies pointed out that this rapid API evolution make the task of keeping complete and accurate extremely difficult [10]. Cai et al. [2] and Xia et al. [20] provided insights into the strategies employed by developers to address runtime incompatibilities that are caused by API evolution.

Non-SDK APIs. Li et al. [9] conducted the first study exploring the use of interfaces not available for third-party apps before Google introduced interface restrictions, finding that 5.4% of third-party apps relied on internal API methods that were inaccessible to developers. The follow-up study by He et al. [7] discovered 112 vulnerabilities in Android 6 and identified over 25 non-SDK interfaces with inconsistent protections in Android 11 and 12. A more comprehensive analysis although on a small set of 66,450 apps was conducted by Yang et al. [21] who explored the use and design of non-SDK interfaces, focusing on restriction mechanisms, usage patterns, and potential for malicious use.

Restrictions. The gaps in app security enforcement have been widely explored at the Android permission level [3,14,16,18]. Sellwood et al. [17] pointed out that architecture changes between OS versions result in gaps that allow execution of malicious apps. Only a few studies have focused on non-SDK API restrictions. He et al. [7] examined how developers can bypass these restrictions and showed that the currently employed countermeasures might not be fully effective. Yan et al. [21] noted that restrictions are sometimes relaxed in response to developer feedback, or to incorporate new features into Android. The study by Barzolevskaia et al. [1] revealed that there are more restrictions found within the Android source code than what is available in the public documentation, suggesting a deeper layer of regulatory measures that are not publicly acknowledged.

3 Approach

To systematically identify and measure the discrepancies in the use of non-SDK calls within the Android ecosystem, we focused our analysis on two layers - official documentation and practical usage. The flow of our analysis is shown in Fig. 1. Our analysis fundamentally relies on two key data sources:

1) **Official Google restrictions lists**. For our analysis, we examined the SDK and non-SDK interfaces available within the Android platform. These interfaces are listed in official restrictions lists with their corresponding assignment

to the restriction categories. These restrictions lists are intended to serve as a guideline for third-party developers and outline the official specifications for API usage within Android applications [4]. We collected restriction lists published for the three latest Android versions[2] (12, 13, and 14).

2) **Android apps**. For this analysis, we collected a diverse dataset of Android applications from various sources to ensure a broad representation of application behaviour and characteristics. Since the restrictions lists were introduced in 2018, we focused primarily on datasets from 2020 and beyond. We relied on publicly accessible collections of benign applications gathered from the Google Play Store and alternative markets. We also included malicious Android applications from VirusShare and VirusTotal. Collected apps were checked for uniqueness to ensure that no two apps were the same. Table 1 gives an overview of the set.

3.1 Data Parsing

Processing Restriction Lists. Google's restriction lists contain API calls (SDK and non-SDK interfaces) and their associated restrictions, which indicate the call's level of access to the Android platform. Figure 2 provides an example of the relationship between calls and restrictions. We parsed the restriction lists. To facilitate our analysis, we extracted calls as well as the restriction(s) associated with the API call. We divided each API call into *caller* and *callee* segments. The *caller* segment typically specifies the class or package from which the call originates. *Callee* indicates the method or function that is being called (including any parameters it might use).

Processing Android Apps. To understand the use of calls within third-party apps, Google provides Veridex, an official static analysis tool that inspects the code of an app to detect the use of non-SDK interfaces [5]. The tool relies on the restriction lists to differentiate between SDK calls that third-party developers are permitted to use and calls that are restricted for third-party apps and non-SDK interfaces. When Veridex detects a non-SDK interface in an APK and finds a corresponding match in the restriction list, it logs the occurrence. Veridex also records the sequence number (log identifier), the mechanism Veridex uses to detect the non-SDK interfaces, the restriction that binds the API call, and the detected call. An example of a Veridex entry is shown in Fig. 3.

While using Veridex, we identified several issues. First, to adapt to the fast-evolving Android restrictions, Veridex is expected to handle restriction lists corresponding to the version of Android targeted by the app. We found that Veridex uses a hardcoded restriction list for Android 12. Second, we observed segmentation faults in instances when the tool attempted to verify the presence of the non-SDK API calls with restriction 'max-target-s' (corresponds to Android 12). We modified the Veridex source code to address these issues.

We used this improved version of the tool to scan our dataset of 714,616 unique APKs to obtain the used non-SDK interfaces. For our analysis, we

[2] At the time of our analysis.

excluded 91,699 (12.83%) APKs that showed no presence of non-SDK interfaces, 15,448 (2.16%) APKs that produced errors during the Veridex scanning, and 24 APKs that showed no presence of non-SDK interfaces but also produced errors during the Veridex scanning. The overview is presented in Table 1. Consequently, our analysis proceeded with the remaining 613,072 APKs (85.79%) for which Veridex successfully generated logs with at least one non-SDK interface.

Fig. 2. An example of Android interface annotations in Android restrictions list

Fig. 3. Structure of Veridex output

3.2 Inconsistency Analysis

We identify discrepancies between the official Android guidelines for non-SDK interface use and their actual use in Android apps by comparing the non-SDK interfaces detected by Veridex to the non-SDK interfaces present in restriction lists. Since the use of non-SDK interfaces is restricted for third-party developers, we would expect Veridex to produce empty logs (i.e., Veridex did not detect any use of non-SDK interfaces). However, previous research has shown that this is not the case and that third-party developers do indeed make use of non-SDK interfaces [21]. Thus, we explore the presence of these interfaces as detected by Veridex, specifically focusing on whether or not the non-SDK interfaces and their corresponding restrictions align with the official restriction lists.

Since restriction lists differ for each Android version, we perform the comparison for each version separately. For each call present in the Veridex output, we match it to the corresponding interface in the restriction lists. We label the results of our comparison as follows:

- **Full match:** the non-SDK interface found in the Veridex log is an exact match to the non-SDK interface present in the restriction list.
- **Partial match:** either the *caller* or *callee* segments in a Veridex detected non-SDK interface match either the *caller* or *callee* in a non-SDK interface from the restriction lists.

Table 1. Summary of APK analysis

Set	Collection years	Total APKs	APKs using non-SDK interfaces	Total Non-SDK interfaces from Veridex logs
AndroGalaxy	2017, 2018, 2019	7,462	7,294 (97.75%)	1,045,655
AndroidAPKsFree	2020	1,333	1,300 (97.52%)	216,213
Anzhi	2017, 2020	5,894	5,351 (90.79%)	389,007
APKGod	2020	4,690	4,317 (92.05%)	630,607
APKMaza	2020	111	111 (100.00%)	12,825
APKPure	2020, 2021, 2023	109,216	104,862 (96.01%)	9,954,476
AppsApk	2020	6,146	5,749 (93.54%)	731,974
Appvn	2020	33,986	31,836 (93.67%)	3,727,848
CracksHash	2021, 2022	3,486	3,473 (99.63%)	540,094
F-droid	2020	7,073	5,678 (80.28%)	384,812
Googleplay	2020, 2023	5,468	5,227 (95.59%)	670,024
Mob.org	2020	1,147	1,128 (98.34%)	154,621
1Mobile	2020	1,370	1,346 (98.25%)	151,566
slideME	2020	18,052	18,049 (99.98%)	754,699
Uptodown	2020	59,717	55,999 (93.77%)	7,472,493
VirusShare	2012- 2018, 2020–2022	440,106	360,173 (81.84%)	20,720,264
VirusTotal	2020, 2021	8,160	82 (1.00%)	1,253
Xiaomi	2020	1,199	1,097 (91.49%)	120,544
Total	-	**714,616**	**613,072 (85.79%)**	**47,967,468**

– **No match:** an API call detected by Veridex tool does not fit either the 'full match' or 'partial match' categories. Non-SDK interfaces that fall under the 'no match' category are particularly significant as they may indicate either undocumented non-SDK interface usage or discrepancies in the logging or detection methods of the Veridex tool.

We refer to the list of identified discrepancies as *API calls inconsistencies*.

Similarly, we analyze the restrictions of the 'full match' calls that are present in both the Veridex logs and the restriction lists. We compared restriction categories of calls as they are labelled by the Veridex tool and defined by Google in the documentation. We expect the level of restrictions placed on API calls to be equivalent, i.e., API calls noted by Google as restrictive should align with information present in the official restriction lists. As with our analysis of API calls, we identify instances contradicting this assumption and categorize the results as follows:

– **Full Match:** the Veridex API call and the call present in the restriction list have identical restrictions.
– **Partial match:** the Veridex API call has at least one restriction in common with the corresponding API call in the restriction list. For example, if a Veridex API call is labelled with the restrictions 'unsupported, public', and the corresponding API in the restriction list is labelled with any of the combinations such as 'unsupported', 'unsupported, public' or 'unsupported, public, lo-prio, there is a partial match due to the presence of the restriction 'unsupported' that is associated with both calls.
– **No match:** the restrictions associated with an API call detected by Veridex has no overlap with the restrictions present in the restriction list for that API call.

To verify discrepancies, we then group and manually inspect the instances in which the restriction information does not align.

We refer to the list of identified discrepancies as *Restriction inconsistencies*.

4 Summary of APK Analysis

Table 2. Restrictions of Android interfaces

Restriction combinations	Restriction list Version 12	Restriction list Version 13	Restriction list Version 14
blocked	247,088	280,562	337,182
lo-prio,max-target-o	97,245	95,198	92,806
public-api,sdk,system-api,test-api	66,670	72,326	79,105
core-platform-api,public-api,sdk,system-api,test-api	42,052	44,338	48,180
unsupported	20,497	20,353	20,245
sdk,system-api,test-api	13,041	15,465	17,929
max-target-r	3,010	2,949	2,884
blocked,test-api	1,432	1,577	2,005
sdk	1,555	1,558	1,660
max-target-q	810	810	804
max-target-p	760	754	753
removed,unsupported	420	420	423
blocked,core-platform-api	378	375	378
core-platform-api,lo-prio,max-target-o	190	177	177
lo-prio,max-target-o,test-api	158	154	155
core-platform-api,unsupported	141	141	141
test-api,unsupported	89	90	98
lo-prio,max-target-r	52	52	50
core-platform-api,public-api,sdk	50	51	51
max-target-r,test-api	36	34	37
max-target-o	14	14	14
core-platform-api,max-target-q	12	12	12
core-platform-api,max-target-r	8	8	8
max-target-s	0	4	4
public-api,sdk	2	2	2
core-platform-api,max-target-p	2	2	2
max-target-q,test-api	1	1	1
Total	**495,713**	**537,427**	**605,106**

Restriction Lists Analysis. To gather more insight into restrictions of Android interfaces, we analyzed API calls present in the restriction lists for the analyzed Android versions. Our analysis presented in Table 2 shows that the categorization used by Android is more complex. Officially, Android lists only 5 restriction categories. Our analysis showed that there are additional categories that are excluded from the official documentation but present in the restriction lists. These categories are 'core-platform-api', 'system-api', 'removed', and 'lo-prio'. While their restrictions are unclear, they all appear in conjunction with other documented categories.

In addition to individual categories, a significant portion of interfaces were labelled using restriction combinations, many of which are contradictory. For

instance, 24% of interfaces in each versions are public-api, sdk, system-api, test-api 'sdk,system-api, test-api', and core-platform-api, public-api, sdk, system-api, test-api combinations. Restriction categories *public* and *sdk* indicate that these interfaces are publicly accessible while *test-api* suggests that the interfaces are intended only for internal system testing (i.e., not for third-party apps).

Similarly, the combinations 'max-target-(q and r), test-api' also present contradictory labelling. Conditionally blocked, these interfaces are publicly open to third-party apps targeting Android 10 and 11, respectively, and at the same time, not intended for third-party developers ('test-api').

Veridex Analysis. In our dataset of 714,616 Android applications, Veridex identified the use of non-SDK interfaces in 613,072 APKs which is 85.79% of all analyzed apps. We documented a total of 47,967,468 non-SDK interfaces within these APKs, as detailed in Table 1. This prevalence of non-SDK interfaces in third-party apps is particularly puzzling given Google's efforts to restrict their usage. Google introduced restrictions for non-SDK interfaces, aiming to decline their use in applications to enhance platform security and maintain API consistency. We expected to see a decline in the presence of non-SDK interfaces in applications developed post-2018 (when restrictions were officially introduced).

Our examination revealed that more than 80% of the APKs in each dataset contained non-SDK interfaces. Alarmingly, within the subset of 5,468 APKs sourced from the Google Play Store, 5,227 APKs (95.59%) were found to contain non-SDK interfaces. This finding is unexpected and raises concerns about the effectiveness of Google's measures to discourage the use of non-SDK interfaces. Despite the heavy monitoring and regulation by Google, a significant number of applications in the official store still use these restricted interfaces, suggesting a gap in enforcement and developer noncompliance with the intended guidelines.

Table 3. Overview of API calls inconsistency

Set	Total APKs	Total calls	Full match calls	Partial match calls		No match calls	Total mismatched calls
				caller match	callee match		
v12	**608,954**	**15,688,459**	**4,372,357 (27.87%)**	**11,316,039 (72.13%)**	**0 (0.00%)**	**63 (0.00%)**	**11,316,102 (72.13%)**
v13	**612,556**	**16,188,834**	**4,864,281 (30.05%)**	**11,324,386 (69.95%)**	**35 (0.00%)**	**132 (0.00%)**	**11,324,553 (69.95%)**
v14	**610,911**	**16,090,175**	**4,727,085 (29.38%)**	**11,361,923 (70.61%)**	**833 (0.01%)**	**334 (0.00%)**	**11,363,090 (70.62%)**

5 Inconsistency Analysis

5.1 API Calls Inconsistencies

Across Android versions 12 to 14, we analyzed a total of 47,967,468 non-SDK interfaces detected by Veridex across 613,072 apps. The details of the API call inconsistency analysis are presented in Table 3.

Theoretically, all API calls detected by Veridex in third-party apps should be present in the official documentation, but this is not the case. The number of API calls fully matching the official restriction lists remained around 30% across the three versions, i.e., 27.87% in version 12, 30.05% in version 13, and 29.38% in version 14. Hence, a significant portion of calls used by apps did not match official documentation. We observed 72.13% (11,316,102) non-SDK calls with mismatches (partial and no match) for version 12, 69.95%(11,324,553) in version 13, and 70.62% (11,363,090) in version 14.

Among them, the most worrisome are the not matching instances. For version 12, we found 63 interfaces used in the third-party Android apps that did not have any corresponding entries in the restriction list. The number of interfaces with no corresponding matches increased across subsequent versions: 132 interfaces in version 13, and 334 interfaces in version 14. The presence of non-SDK API calls that are neither documented nor restricted is puzzling. This contradicts Google's guidelines which state, "To avoid crashes and unexpected behavior, apps should only use the officially documented parts of the classes in the SDK" [4].

We manually analyzed the 'no match' non-SDK interfaces. In version 12, the interface 'Ljdk/internal/misc/Unsafe' was flagged 62 times as a 'no match'. This particular interface, known for providing low-level and unsafe operations within Java, allows for memory manipulation and the execution of arbitrary code [15]. Despite its critical nature, this interface remains undocumented in the official lists, and Veridex merely notes its presence without associating any specific restrictions. These instances are particularly alarming as they clearly indicate that third-party developers are well aware of these undocumented interfaces and use them. In subsequent versions 13 and 14 (Table 4), we identified the top five occurring 'no match' interfaces. These interfaces impact core system functionalities and are similarly logged by Veridex without restrictions.

Table 4. Top five 'No match' non-SDK interface occurrences

Android version	Non-SDK interfaces	Number of occurrences
v13	Lcom/android/internal/widget/FloatingToolbar$FloatingToolbarPopup;->mParent	5
	Lcom/android/internal/widget/FloatingToolbar;->mPopup	5
	Lcom/android/internal/widget/FloatingToolbar;->mWidthChanged	5
	Lcom/android/internal/content/PackageHelper;->APP_INSTALL_EXTERNAL	3
	Lcom/android/internal/os/BatterySipper;->sumPower	2
v14	Lcom/android/internal/os/BatteryStatsImpl;->computeBatteryRealtime	12
	Lcom/android/internal/os/BatteryStatsImpl;->CREATOR	11
	Lcom/android/internal/os/BatteryStatsImpl$Uid;->getUid	11
	Lcom/android/internal/os/BatteryStatsImpl;->getUidStats	11
	Lcom/android/internal/os/BatteryStatsImpl;->readLocked	11

During our analysis, we identified a significant number of partial matches, i.e., the calls matched only on the caller or callee side, which highlights inconsistencies in API interactions. We flagged 11,316,039 (72.13%) non-SDK interfaces with caller matches for version 12. Versions 13 and 14 had 11,324,386 (69.95%) and 11,361,923 (70.61%) caller matches, respectively. In contrast, the number of calls

with matching callees was much lower: 0 matches for version 12, 35 for version 13, and 833 for version 14. Partial matches suggest a gap in the documentation, or inconsistent enforcement of restrictions in which certain parts of the API calls are acknowledged but not fully regulated through restrictions. Partial matches cause incomplete enforcement of restrictions, which complicates the efforts of security teams to comprehensively secure API usage. Such inconsistencies may expose the system to potential security issues where certain aspects of the API calls are controlled, but other aspects remain unrestricted and exploitable.

Table 5. Restriction inconsistencies of fully matched non-SDK calls

Set	APKs	Total calls	Full match	Mismatches		
				Partial match	No match	Total
Android v12	481,358	4,372,357	2,868,595 (65.61%)	1,503,673 (34.39%)	89 (0.00%)	1,503,762 (34.39%)
Android v13	491,098	4,864,281	2,857,197 (58.74%)	2,007,031 (41.26%)	53 (0.00%)	2,007,084 (41.26%)
Android v14	487,881	4,727,085	2,854,440 (60.38%)	1,872,558 (39.61%)	87 (0.00%)	1,872,645 (39.62%)

Collectively, our findings demonstrate the inconsistencies in non-SDK interface enforcement and documentation. Over 69% of the Veridex logs that we analyzed showed mismatched non-SDK interfaces compared to the official restriction lists. This suggests that many of the restrictions associated with these non-SDK interfaces may not be enforced as stated by Google and are lacking proper documentation.

5.2 Restriction Inconsistencies

To explore restriction inconsistencies, we analyzed only fully matching non-SDK interfaces across three Android versions. The summary is presented in Table 5.

Calls with 'Full Match' Restrictions. Similarly to non-SDK calls, we expected the restrictions indicated by Veridex to match information present in the official restricted lists. Surprisingly, this was not the case. In version 12, 65.61% of calls had restrictions fully matching the corresponding restrictions in official lists, 58.74% in version 13, and 60.38% in version 14.

Calls with 'Partial Match' Restrictions. Out of 4,372,357 'full match' non-SDK interfaces, 34.39% of calls had only partially matching restrictions for version 12. In versions 13 and 14, these numbers were slightly higher (41.26% and 39.61%, respectively). Approximately one-third of the identical calls had at least one (but not all) restriction category the same in both the official restriction lists and the corresponding Veridex logs. The occurrences are listed in Table 6.

The significant existence of partial matches raises substantial security concerns. It suggests that, while the non-SDK interfaces are recognized and restricted to some extent, their full restrictions are not known in advance to

Table 6. 'Partial match' restriction inconsistencies

Android version	Veridex	Restriction list	Occurrences
v12	unsupported	removed,unsupported	1,040,092
	unsupported	public-api,sdk,system-api,test-api	318,840
	max-target-o	lo-prio,max-target-o	143,639
	max-target-o,core-platform-api	core-platform-api,lo-prio,max-target-o	661
	max-target-o,test-api	lo-prio,max-target-o,test-api	320
	max-target-r	lo-prio,max-target-r	90
	unsupported,test-api	unsupported	14
	max-target-o,test-api	lo-prio,max-target-o	12
	max-target-o	lo-prio,max-target-o,test-api	2
	unsupported	sdk,system-api,test-api	2
	blocked,test-api	blocked	1
	Total	**-**	**1,503,673**
v13	unsupported	removed,unsupported	1,861,259
	max-target-o	lo-prio,max-target-o	142,428
	unsupported	public-api,sdk,system-api,test-api	2,073
	max-target-o,core-platform-api	core-platform-api,lo-prio,max-target-o	657
	max-target-o,test-api	lo-prio,max-target-o,test-api	371
	max-target-r	lo-prio,max-target-r	90
	unsupported	core-platform-api,public-api,sdk,system-api,test-api	79
	unsupported,test-api	unsupported	32
	unsupported	sdk,system-api,test-api	22
	max-target-o	lo-prio,max-target-o,test-api	17
	max-target-o,test-api	lo-prio,max-target-o	2
	unsupported,test-api	public-api,sdk,system-api,test-api	1
	Total	**-**	**2,007,031**
v14	unsupported	removed,unsupported	1,730,161
	max-target-o	lo-prio,max-target-o	141,114
	max-target-o,core-platform-api	core-platform-api,lo-prio,max-target-o	657
	max-target-o,test-api	lo-prio,max-target-o,test-api	388
	max-target-r	lo-prio,max-target-r	90
	unsupported	core-platform-api,public-api,sdk,system-api,test-api	86
	unsupported	test-api,unsupported	27
	unsupported	sdk,system-api,test-api	19
	unsupported	public-api,sdk,system-api,test-api	15
	blocked	blocked,test-api	1
	Total	**-**	**1,872,558**

developers. This can potentially lead to contradictions during the execution of the restrictions. For example, among the partially matched restrictions, we identified multiple non-SDK interfaces across the three versions where Veridex marked the calls as 'unsupported' while the restriction list specified these calls under various combinations that include 'public-api' and 'sdk'. These labels indicate that these interfaces are officially documented and open for use by third-party developers, i.e., technically these are SDK-interfaces. Yet Veridex sees them as unreliable interfaces that can change at any time.

Aside from this issue, the presence of contradictory restriction combinations causes further confusion about the actual restrictions that are being enforced for

an interface. Although this issue was noted by previous studies [1,21], our analysis further emphasizes the extent of this problem. The restriction combination 'public-api,sdk,system-api,test-api' is one of the examples of this contradiction, while 'sdk' points to the official availability of the interfaces, 'test-api' says that interface is reserved for internal system testing and hence is not available for third-party apps. Resolving these cases through Veridex is not always possible as the corresponding Veridex output may also contradict the official guidance.

Table 7. 'No match' restriction inconsistencies

Android version	Veridex	Restriction list	Occurrences
v12	blocked	unsupported	71
	blocked	max-target-r	16
	blocked	max-target-p	2
	Total	-	**89**
v13	blocked	unsupported	37
	max-target-r	blocked	4
	unsupported	blocked	4
	max-target-p	blocked	2
	max-target-p	public-api,sdk,system-api,test-api	2
	unsupported	max-target-s	2
	max-target-o	public-api,sdk,system-api,test-api	1
	max-target-r	public-api,sdk,system-api,test-api	1
	Total	**-**	**53**
v14	unsupported	blocked	75
	max-target-p	public-api,sdk,system-api,test-api	3
	max-target-r	blocked	3
	max-target-r	public-api,sdk,system-api,test-api	3
	unsupported	max-target-s	2
	max-target-o	public-api,sdk,system-api,test-api	1
	Total	**-**	**87**

Calls with 'No match' restrictions We observed that for a small number of calls Veridex reported differences from official documentation restrictions. Specifically, 89 calls had not-matching restrictions in version 12, 53 calls in version 13, and 87 in version 14. These discrepancies in restrictions are presented in Table 7.

The majority of these cases (187) include blocked instances on one side and unsupported instances on the other. These combinations reveal some obvious contradictions between Veridex's findings and the official documentation with respect to how the same non-SDK interfaces should be restricted. For instance,

we observed occurrences of calls labelled as 'blocked' in the official documentation, yet listed as 'unsupported' by Veridex (4 occurrences in version 13 and 75 occurrences in version 14), and vice versa, listed as 'unsupported' by restriction list and seen as 'blocked' by Veridex (71 occurrences in Android version 12).

In theory, these instances highlight critical gaps in the Android official documentation. In practice, they convey a false sense of available calls when, in fact, they will be blocked by the operating system. A similar false sense of availability is conveyed by conditionally blocked calls, which Veridex lists as 'max-target-x' where x indicates the target API level after which the app can no longer access these non-SDK interfaces. Veridex detected 9 non-SDK interfaces as conditionally blocked while restriction lists state that these calls are blocked regardless of an app's target API level. Such inconsistencies raise concerns regarding potential misclassifications, undocumented API usage, or errors in restriction mapping.

Other discrepancies involve 11 cases of non-SDK interfaces labelled by Veridex as 'max-target-x' while the corresponding entries in the restriction lists indicate that these interfaces are 'public-api, sdk, system-api, test-api', implying that they can be used by developers. Although, we have to acknowledge that even the official restriction listed as a combination of public-api, sdk, system-api is confusing and does not give a clear message about the availability of the calls. To assess the severity of these contradictions, we manually analyzed the affected interfaces, observing that non-SDK interfaces exist within critical Android packages such as 'bluetooth', 'location', 'telephony', 'os', 'net', 'view', 'text', and 'app'.

These fluctuations in restriction labels for the same non-SDK interface pose significant security risks. For developers, these inconsistencies can lead to confusion. If an interface is deemed 'blocked' in one instance, a developer might avoid using it under the belief that it cannot be used. However, if the same interface is classified as 'unsupported', it might suggest that the interface can be used without official support. These contradictions, or misclassifications, might encourage developers to use risky interfaces without realizing the potential threats, due to inaccurate restriction information.

6 Case Study

During our analysis, we identified 159 unique APKs across Android versions 12, 13, and 14 that used non-SDK interfaces (as identified by Veridex) not present in any corresponding restriction lists. Specifically, we observed 63 APKs in version 12, 19 in version 13, and 80 in version 14.

For manual investigation, we decided to focus on apps that use 'Lcom/android/internal/os/BatteryStatsImpl' interface. This is an internal API meant to be used by system apps only. The interface requires the `BATTERY_STATS` permission, which is only available to system apps. Hence, the use of this interface by any third-party apps is prohibited and should not be possible. We randomly selected 15 APKs using this interface. Among them, 3 were from Androgalaxy, 1 from apkgod, 7 from apkpure, and 4 from appvn. There were no apps with 'no match' calls from the Google Play Store.

These 15 APKs were installed on Android smartphones running versions 12 (Samsung SM G990W2), 13 (Google Pixel 7 Pro), and 14 (Google Pixel 8). Out of them, 8 APKs were successfully installed across all three versions, 2 only on devices with Android 14, 3 on Android 12 and 13, 1 only on Android 12, and 1 failed to install on any version. We imported each app into Android Studio (Version 2024.1.2 Canary 8) and installed on the three smartphones connected to Android Studio through USB debugging. Apps were installed one at a time and the phone was rebooted with a fresh image between apps. For each app, we monitored the logs generated through *Android Debug Bridge (adb)*, a command-line tool that enabled us to observe the logs generated by the smartphone and the installed app. To trigger app functionality, we manually browsed each app for several minutes.

Our findings revealed several critical points. *First*, the accessed battery information was correctly displayed by all 14 apps. *Second*, the adb *logcat* logs indicated that a system-level communication was triggered to retrieve battery statistics through 'BatteryStatsImpl' calls. In all 14 APKs, the 'BatteryStatsImpl' calls originated from the system server (a system process spawned by zygote), and not under the PID of the running app as expected. This behaviour was consistent across all analyzed APKs. In Android 12, adb also displayed a warning about using this interface, which was not present in Android 13 and 14. The calls, although not officially present or allowed for use by third-party apps as correctly indicated by Veridex, were executed successfully. Further investigation of Android's platform code revealed that the Android OS broadcasts battery statistics that are accessed by system apps. It appears that there are no additional checks performed by the OS to prevent third-party apps from accessing battery information, which is likely because this interface is not available to such apps.

This finding is concerning as it reveals a significant security oversight: *although these APKs functioned as expected in providing battery statistics, they did so by bypassing Android's security mechanisms.* This scenario highlights a critical security gap within the Android ecosystem where non-SDK interfaces that are not officially documented are still being accessed, potentially exposing sensitive device data. Theoretically, such communications should be blocked by the system to prevent exposure of sensitive data, but our observations confirm that these APKs could fetch battery details and display them to the user without any system-level restrictions.

7 Discussion

In our analysis, we uncovered several key aspects:
① *Widespread violation of restrictions by third-party developers.* The widespread use of non-SDK interfaces in third-party apps raises significant security concerns. Of the 613,072 APKs that use non-SDK interfaces, we found a core group of 477,984 APKs using non-SDK interfaces *and* are present in all three Android versions. Non-SDK interfaces should not be present since their inclusion contradicts Google's guidelines on non-SDK API usage. Even in the Google Play Store

where security measures are expected to be stringent, the number of APKs with non-SDK interfaces is high: 95.13% (4,963 APKs) in version 12, 98.34% (5,139 APKs) in version 13, and 98.37% (5,141 APKs) in version 14. Widespread usage of non-SDK interfaces, even when matching with restriction lists, highlights gaps in Android's enforcement of its security policies.

② *Significant misalignment between the non-SDK interfaces detected by Veridex and the documented interfaces in the official restriction lists.* There were a large number of inconsistencies between Veridex's reports and the restriction lists. For example, 72.13% of the used interfaces reported by Veridex were either partially matched or completely missing from the corresponding restriction list in version 12 (69.95% and 70.62% of interfaces for versions 13 and 14, respectively).

③ *The use of undocumented non-SDK interfaces absent from restriction lists.* We discovered a small number of apps that used interfaces not present in any official restrictions lists, i.e., 63 APKs in version 12, 19 APKs in version 13, and 80 APKs in version 14. Overall, we found 96 unique non-SDK interfaces used by third-party apps, but absent from the restrictions lists.

④ *Absent non-SDK interfaces are successfully used by third-party apps.* We manually executed and analyzed 15 apps containing undocumented non-SDK interfaces. Of these, 14 apps were successfully installed and executed on the phones, and they retrieved information through non-SDK interfaces. These instances indicate gaps in Android's security since APKs use interfaces that are not in the restriction lists, opening up avenues for severe security exploitation.

⑤ *Mismatching restrictions for identical non-SDK interfaces.* There were significant discrepancies between the non-SDK interfaces detected by Veridex and their corresponding entries in the official restriction lists. Restrictions of 34.39% non-SDK interfaces reported by Veridex in version 12 were either a 'partial match' or 'no match' with their corresponding restrictions in the restriction lists. The number of mismatches increased in version 13 (41.26%) and version 14 (39.61%). The quality of official Android documentation is not improving over time, and critical guidelines for access control are becoming more confusing.

⑥ *Conflicting documentation.* A significant portion of interfaces are labelled with restrictions that are not in the documentation (e.g., 'lo-prio'), or labelled with contradictory restriction combinations (e.g., 'core-platform-api, public-api, sdk, system-api, test-api'). Our results align with previous research finding that Android documentation is often incomplete, irrelevant, and inaccurate [1,12].

⑦ *Inadequate detection of non-SDK interfaces.* Google offers Veridex and adb as reliable tools for identifying the use of non-SDK interfaces [4]. Our case study revealed significant discrepancies between Veridex's results and adb's *logcat* functionalities. This clearly indicates that, despite Google's guidance, these tools are inadequate and not equivalent in detection of non-SDK use in apps. Our observations underscore significant challenges in Android's current security framework's capability to accurately detect and report non-SDK interface usage.

8 Conclusion

In this study, we conducted a systematic analysis of non-SDK interfaces, focusing specifically on the discrepancies between the official documentation in the restriction lists and the practical usage as detected by the Veridex tool. Our investigation into the existence of non-SDK interfaces across Android 12, 13, and 14 revealed significant inconsistencies in how these interfaces are documented in restriction lists, and how they are actually used within applications as reported by Veridex. The severity of the issues we uncovered ranges from minor misclassification, which could lead to performance inefficiencies, to critical security issues. This underscores the urgent need for more rigorous and consistent documentation and stricter enforcement of API usage policies.

References

1. Barzolevskaia, A., Branca, E., Stakhanova, N.: Measuring and characterizing (mis)compliance of the android permission system. IEEE Trans. Software Eng. **01**, 1–23 (2024)
2. Cai, H., Zhang, Z., Li, L., Fu, X.: A large-scale study of application incompatibilities in android. In: ISSTA 2019, pp. 216–227. ACM, New York, NY, USA (2019)
3. Calciati, P., Kuznetsov, K., Gorla, A., Zeller, A.: Automatically granted permissions in android apps: an empirical study on their prevalence and on the potential threats for privacy. In: MSR 2020, pp. 114–124 (2020)
4. Google: Restrictions on non-sdk interfaces. https://developer.android.com/guide/app-compatibility/restrictions-non-sdk-interfaces
5. Google: Veridex. https://android.googlesource.com/platform/prebuilts/runtime/+archive/main/appcompat.tar.gz
6. He, D., Li, L., Wang, L., Zheng, H., Li, G., Xue, J.: Understanding and detecting evolution-induced compatibility issues in android apps. In: ASE 2018, pp. 167–177 (2018)
7. He, Y., et al.: A systematic study of android non-SDK (hidden) service API security. IEEE Trans. Dependable Secur. Comput. **20**(2), 1609–1623 (2023)
8. Li, L., Bissyandé, T.F., Wang, H., Klein, J.: CiD: automating the detection of API-related compatibility issues in Android apps. In: ISSTA 2018, pp. 153–163. ACM, New York, NY, USA (2018)
9. Li, L., Bissyandé, T.F., Le Traon, Y., Klein, J.: Accessing inaccessible android APIs: an empirical study. In: ICSME 2016, pp. 411–422 (2016)
10. Li, L., Gao, J., Bissyandé, T.F., Ma, L., Xia, X., Klein, J.: Characterising deprecated android APIs. Empir. Softw. Eng. **25**, 2058–2098 (2020)
11. Linares-Vásquez, M., Bavota, G., Bernal-Cárdenas, C., Di Penta, M., Oliveto, R., Poshyvanyk, D.: API change and fault proneness: a threat to the success of android apps. In: FSE 2013, pp. 477–487. ACM, New York, NY, USA (2013)
12. Liu, P., Li, L., Yan, Y., Fazzini, M., Grundy, J.: Identifying and characterizing silently-evolved methods in the android API. In: ICSE-SEIP 2021, pp. 308–317 (2021)
13. McDonnell, T., Ray, B., Kim, M.: An empirical study of API stability and adoption in the android ecosystem. In: ICSM 2013, pp. 70–79 (2013)
14. Oishwee, S.J., Codabux, Z., Stakhanova, N.: Decoding android permissions: a study of developer challenges and solutions on stack overflow. In: ESEM 2024 (2024)

15. Oracle: The unsafe class: unsafe at any speed. https://blogs.oracle.com/javamagazine/post/the-unsafe-class-unsafe-at-any-speed
16. Scoccia, G.L., Peruma, A., Pujols, V., Christians, B., Krutz, D.E.: An empirical history of permission requests and mistakes in open source android apps. In: MSR 2019, pp. 597–601 (2019)
17. Sellwood, J., Crampton, J.: Sleeping android: the danger of dormant permissions. In: SPSM 2013, pp. 55–66 (2013)
18. Shao, Y., Chen, Q.A., Mao, Z.M., Ott, J., Qian, Z.: Kratos: discovering inconsistent security policy enforcement in the android framework. In: NDSS 2016 (2016)
19. Syer, M.D., Nagappan, M., Adams, B., Hassan, A.E.: Studying the relationship between source code quality and mobile platform dependence. Softw. Qual. J. **23**, 485–508 (2015)
20. Xia, H., et al.: How android developers handle evolution-induced API compatibility issues: a large-scale study. In: ICSE 2020, pp. 886–898. ACM, New York, NY, USA (2020)
21. Yang, S., Li, R., Chen, J., Diao, W., Guo, S.: Demystifying android non-SDK APIs: measurement and understanding. In: ICSE 2022, pp. 647–658 (2022)