FREELENS: Revealing Mobile App Feature Differences and Their Security and Privacy Implications across Geographical Regions

Jiawei Guo, Yu Nong, Zhiqiang Lin *Fellow, IEEE*, and Haipeng Cai *Senior Member, IEEE*

***Abstract*—Mobile apps are known to distribute different versions across geographic regions to accommodate local regulations and market preferences. While prior research has examined metadata-level differences such as permissions and privacy policies, there lacks systematic investigation into code-level geographic variations that may impact security. In this work, we present the first comprehensive study of geo-feature differences (GFDs) in Android apps at the code implementation level. We develop FREELENS, a novel framework that overcomes key technical challenges including code obfuscation and analysis scalability to identify and characterize security-relevant variations across regions. Using FREELENS, we conducted a large-scale study of 21,120 Android apps distributed across ten countries with diverse levels of Internet freedom. Our findings reveal that GFDs are widespread, in total 42,977 individual GFDs from 1,120 unique Apps, with significant variations in advertising, data handling, and authentication mechanisms. These differences frequently compromise security baselines and introduce disparities in privacy protections across regions. The study highlights a rising trend in GFD prevalence, emphasizing the urgency for harmonized privacy and security standards. Based on our empirical findings, we also provide actionable insights for developers, platform providers, and regulators to ensure equitable user protections.**

***Index Terms*—Android, Security, Privacy, Program Analysis**

I. INTRODUCTION

It is known that mobile applications (apps) are distributed with different versions, which may exhibit different behaviors, across geographic regions, driven by factors ranging from regulatory compliance to market adaptation. For instance, investigations into popular messaging apps revealed varying levels of encryption and content filtering across countries [1], demonstrating how geographic variations can impact user privacy and information access.

Understanding these geographic feature differences (GFDs) and their implications for security and privacy is crucial for users, developers, and regulators alike. However, identifying and analyzing such differences systematically presents significant challenges. While these variations often stem from legitimate needs to comply with local regulations or adapt to market preferences, they can also mask concerning disparities in security protections and privacy guarantees.

Previous research has primarily approached this problem through analysis of app metadata and configurations. Kumar et al. [2] conducted the first large-scale study of geodifferences in mobile apps by examining application variations in permissions, privacy policies, and basic security features in Android across regions. Yang et al. [3] investigated differences between official and third-party app markets by analyzing app metadata and market policies. While these studies provide valuable insights into surface-level differences, they cannot capture the full spectrum of behavioral variations implemented at the code level.

In this paper, we fill this gap by conducting a large-scale, semantic *code-level* characterization of feature variations between country-specific versions of the same mobile app, referred to as *geo-feature differences* (**GFD**s). Examining GFDs at code level is essential because code implementations ultimately determine an app's actual behavior, potentially revealing security and privacy relevant variations that are not apparent or cannot be identified from metadata alone. For instance, two versions of an app might declare identical permissions but implement different data collection or processing logic, leading to varying privacy implications across regions (referred to as **SPIs**, Security/Privacy Implications). Through code-level differencing of diverse regional versions of apps with GFDs (referred to as **GFD apps**), the overarching goal of our study is to systematically demystify how mobile app vendors/developers may (stealthily) differentiate their application functionalities and what the functional variations imply in security and privacy.

However, conducting such a study at scale presents several critical challenges. First (**C1**), collecting representative app samples across regions is non-trivial due to geographic restrictions and version control mechanisms enforced by app distribution services (e.g., Google Play Store). Second (**C2**), it is well known that precise, fine-grained analysis of mobile apps (e.g., in Android) is costly hence often facing scalability challenges [4], [5], [6], [7], while surface-level analysis (e.g., of metadata [2]) only offers very limited understanding of real GFDs. Third (**C3**), for legitimate (e.g., compliance/protection) purposes, mobile apps are often obfuscated [8], which makes it difficult to identify meaningful differences between app versions. Finally (**C4**), connecting low-level code to high-level feature differences and further to security/privacy understandings requires bridging significant semantic gaps.

To address these challenges, we developed FREELENS, an automated framework that combines strategic app collection, obfuscation-resilient code diffing, and code-level change-semantics characterization. Our framework employs a three-phase approach, with each phase aiming to overcome each fundamental challenge. To overcome C1, **Phase 1** conducts scalable mining of region-specific app versions using controlled app-sotre accounts. To tackle C2 and C3, **Phase 2** balances analysis scalability and effectiveness through profiling method-level control flow in apps, while identifying meaningful code differences via an obfuscation-resilient call-

Jiawei Guo, Yu Nong, and Haipeng Cai are with the Department of Computer Science and Engineering, University at Buffalo, Buffalo, NY.

Zhiqiang Lin is with the Department of Computer Science and Engineering, The Ohio State University, Columbus, Ohio.

Corresponding author: Haipeng Cai; Email: haipengc@buffalo.edu

path analysis. To address C4, **Phase 3** interprets these differences using foundation-scale large language models (LLMs) to bridge the semantic gap between code changes and SPIs. This approach enables us to analyze geographic variations systematically while maintaining scalability and accuracy.

Using this framework, we conducted a large-scale measurement study of 21,120 Android apps across ten countries representing different levels of internet freedom. Our investigation examines how apps implement geographic variations, what patterns these variations follow, and what security and privacy implications they carry. The study pays particular attention to highly-popular apps, where geographic variations can affect millions of users worldwide.

Our study reveals several important findings. **First**, we observed an evolution in customization practices: early apps (2010–2014) exhibited minimal regional variations, focusing on basic UI changes, while modern apps (2019–2021) integrate sophisticated, multi-layered adaptations involving UI, core functionality, and privacy-sensitive features. These adaptations often result in cascading security implications. **Second**, we found that security implementations across regions are inconsistent, often arising unintentionally as side effects of regional customization rather than deliberate design choices, and even seemingly benign UI changes can lead to significant security ramifications. **Third**, we identified a reactive approach to privacy, with apps implementing stronger protections only in regions with strict regulations while maintaining minimal safeguards elsewhere. Discrepancies between stated privacy policies and actual implementations are common, with features like crash reporting and analytics selectively enabled without proper user notification. **Fourth**, monetization strategies, observed in 28.6% of apps, often compromise privacy, as regions with aggressive advertising implementations tend to exhibit weaker privacy protections. User consent mechanisms also vary widely across regions, leading to inconsistent privacy standards and user experiences. **Lastly**, critical regional differences with security implications are rarely documented in privacy policies or Play Store declarations, leaving users in different regions with varying levels of information and protections. These findings underscore the urgent need for harmonized global privacy and security standards to address these disparities and ensure equitable protections for all users.

The implications of our findings extend to multiple stakeholders. For developers, our results highlight the need for more systematic approaches to managing geographic customization while maintaining consistent security standards. For platform providers, our findings suggest opportunities for improved tools and guidelines to help developers track and evaluate security implications of geographic variations. For users, our study reveals the importance of understanding potential security and privacy differences in apps based on their geographic location.

This paper extends an earlier version of our study presented in [9], and we refer interested readers to it for additional algorithmic and implementation details. It makes the following key and new contributions:

- We perform the first systematic, code-level study of geographic feature differences (GFDs) in mobile apps and their security/privacy implications (SPIs).
- We develop FREELENS, a novel framework for automatically identifying and analyzing GFDs and SPIs in Android apps, resilient to common obfuscation techniques.
- We present extensive characterization results derived from a large-scale study on GFDs and SPIs across 10 countries covering all Internet freedom levels over almost the entire history of Android, including (over the earlier version):
  1) Refined GFD (11) and SPI (15) categories offering a more granular understanding of GFD/SPI variations.
  2) Analysis distinguishing between unique and individual occurrences of GFDs/SPIs across country pairs.
  3) Detailed longitudinal analysis (2010-2023) of the evolution of individual GFD and SPI patterns over time.
  4) Nuanced insights into the impact of Internet freedom levels on GFDs and SPIs, highlighting GFD/SPI distribution divergences due to freedom level gaps.
- We provide enhanced case studies illustrating concrete code-level manifestations of GFDs and their alignment (or lack thereof) with app policies and regulations.
- We offer updated, actionable insights for developers, platform providers, researchers, and users to improve transparency and consistency in app functionalities and security and privacy across regions.

Our code and datasets are available in our artifact package.

II. BACKGROUND AND MOTIVATION

In this section, we provide the relevant background for and then motivate our study.

A. Google Play App Release Mechanisms

The Google Play Store utilizes advanced app distribution mechanisms, allowing developers to tailor application releases for specific geographic regions [10]. Developers can target production, open testing, or closed testing tracks for country-specific app rollouts, forming the foundation of our investigation into cross-regional app functionality variations, particularly in security and privacy features.

Google Play's model ensures users can only access the *latest version* available for their region. This constraint facilitates compliance with regional regulations but creates scenarios where users in different regions receive distinct app versions. These mechanisms enable developers to customize app functionality based on geographic requirements, laying the groundwork for geographic feature differences (GFDs), the focus of our study.

B. Cross-Domain App Differential Analysis

Prior work has examined cross-domain app differences from various perspectives. Wang et al. [3] explored security discrepancies between Google Play and third-party app markets, highlighting issues like excessive permissions and vulnerabilities. Dong et.al [11] investigated different behaviors of the same app on different devices. Most relevant, Kumar et al. [2] analyzed geographic app differences, identifying regional disparities in privacy policies and permissions.

Beyond metadata comparisons, prior work has explored program differencing and code-change extraction (e.g., AST/tree

differencing and change distilling) as well as Android reverse-engineering tooling for bytecode inspection and library identification. These techniques can serve as conceptual baselines for detecting code changes, but they are often not designed for large-scale cross-region APK comparison under commercial obfuscation. Our work advances the field by delving into code-level implementation differences, revealing deeper insights into how apps handle security and privacy across regions. For example, we analyzed the Google Authenticator app distributed in the U.S. and India, as contrasted in Figure 1. Despite *similar metadata—permissions, privacy policies, and third-party libraries*, *only the U.S. version implemented a "Privacy Screen" feature*, which blanks the app's preview in the system's multitasking overview (the "recent apps" list). This mitigates shoulder-surfing attacks by preventing sensitive authentication codes from being exposed when a user switches away from the app. Note that we only collect free apps. This disparity highlights how the nuance detected by metadata-level analysis fails to reveal exact functionality feature differences that significantly impact user privacy and security.

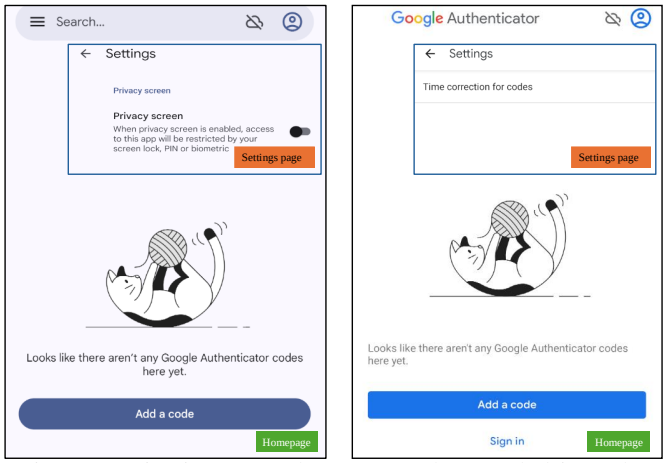

Fig. 1: Motivating example: GFDs only revealed in code.

By investigating deeper, code-level disparities, our work uniquely uncovers previously hidden implications of regional app customization, providing actionable insights for developers and platform providers to ensure consistent security/privacy standards across all regions.

III. The FreeLens Approach

In this section, we describe the overall design, implementation, and evaluation of FreeLens.

A. Design

FreeLens consists of three modules/phases: *potential GFD app mining*, *obfuscation-resilient GFD app identification*, and *semantic GFD characterization*. Figure 2 presents a high-level overview of FreeLens highlighting its architecture. The workflow begins with three **FreeLens Inputs** (1) app list: a curated list of Android apps (by package names) that meet our selection criteria; (2) country list: a list of target countries where the selected apps are distributed; and (3) country-specific user accounts: accounts registered in the target countries with associated payment methods. The first two inputs define the scope of our study, while (3) is needed for scraping the apps from the target countries. As we scrape the apps using Google Services, the accounts are Google accounts.

With these inputs, in **Phase 1**, FreeLens first scrapes the latest version of each chosen app across all target countries using our parallel app-scraping infrastructure. The resulting APKs then undergo a lightweight screening process, filtering out apps that unlikely have GFDs between their country-specific APKs. The goal of this phase is to garner *potential GFD apps* (i.e., those not filtered out).

Taking these potential GFD apps, FreeLens performs static code analysis in **Phase 2** to identify actual GFD apps and their GFDs. For each candidate GFD app, this phase profiles each pair of its country-specific APKs, resulting in pairwise profiles, followed by diffing these profiles. To balance analysis scalability and effectiveness, we profile apps at the granularity level of method-level control flows (i.e., *call-path*), while examining call paths up to framework/SDK APIs (i.e., *API-bounded* profiling). The rationale is that Android apps can only interact with platform capabilities through Android framework APIs [12], [13], [14], and the framework API signatures themselves are stable (i.e., cannot be renamed by app-level obfuscation). Importantly, our analysis is not limited to APIs that directly access sensitive data (e.g., `getContacts()`); instead, FreeLens profiles call paths bounded by *any* Android framework API, including UI/security configuration APIs (e.g., setting `FLAG_SECURE` via `android.view.Window.setFlags()`). Note that FreeLens is not a taint/data-flow analysis and therefore does not recover the semantic meaning of app-internal variables (e.g., whether a value represents an "account balance"). However, many privacy-relevant behaviors still manifest as distinct API-bounded call paths that terminate at externally observable sinks such as networking/IPC (e.g., HTTP requests via OkHttp/Retrofit that ultimately invoke standard Java/Android networking APIs). Thus, when one regional variant introduces (or removes) a code path from specific business logic to a network/IPC sink, FREELENS can flag this as a GFD even if it cannot precisely identify which app-internal fields were transmitted. This method-level diffing analysis is also more robust (e.g., than diffing at statement level) against app obfuscations, mainly renaming. For further robustness, FreeLens computes call-path signatures to capture the most significant app differences despite method-level renaming obfuscations (e.g., method renaming).

Finally, in **Phase 3**, FreeLens characterizes the GFD apps and their call-path diffs computed in **Phase 2**, generating natural-language summaries of feature variations from those diffs between each pair of country-specific APKs of each GFD app through (call-)*graph-based reasoning* (about the code diffs) on foundation LLMs. The resulting pairwise GFDs are fed to these LLMs again to summarize their associated SPIs. The *two-level summary mapping* (i.e., from code-level to natural-language-level feature differences and further to those implications) enables *explainable* characterization results, which are the main **FreeLens Outputs** for further manual confirmation/inspection in our study(§IV).

The key advantage of this multi-phase approach is its ability to efficiently process large numbers of apps while maintaining

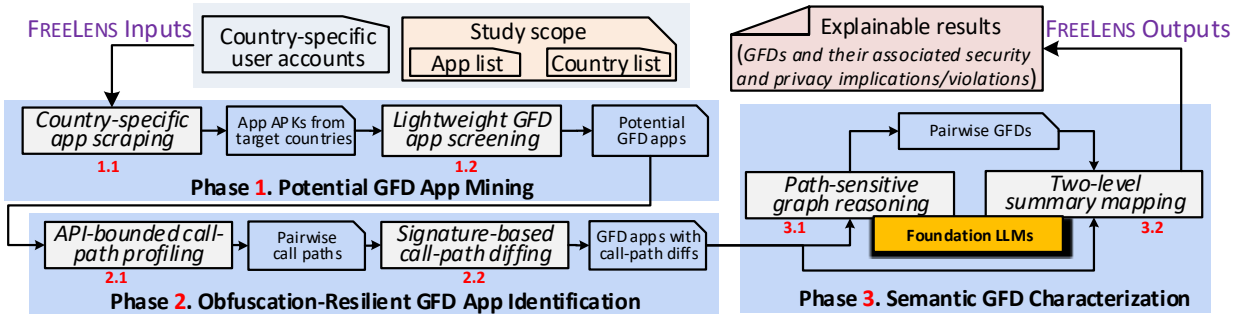

Fig. 2: An overview of FREELENS, including its inputs, three working phases and per-phase steps, and outputs.

precision in identifying security-relevant differences. **Phase 1** quickly filters out apps unlikely to contain GFDs, allowing **Phases 2** and **3** to focus computational resources on detailed analysis of promising candidates. Additionally, our approach's resilience to code obfuscation ensures reliable results even when analyzing commercial apps that employ aggressive code protection techniques. For scalability purposes, we did not consider code (whether it be third-party, native, or user code) that is not used by the app (i.e., not reachable from the app's user-code entry points as per the call graph) yet. However, the reachability analysis could be inaccurate if the two APKs inconsistently applied control-flow obfuscation. But again this is not tied with if the code is third-party or not. The design details of each phase can be found in the conference version of this paper [9].

B. Implementation and Evaluation

In **Step 1.1**, we utilized Raccoon [15] which supports batch download apps with CLI support, to assist with acquiring the given version of an app. For **Step 1.2**, we employed Jadx [16] to analyze the structural characteristics hence computing the three metrics (class count, method count, and instruction count) for each APK. Finally, for creating the standard call graph of each APK, we used FlowDroid [4], a popular static analyzer of Android apps. We analyze the entire APK, including any third-party code if present in the APK. We did not explicitly recognize third-party code, nor treat them differently from user code, in our analysis. Accordingly, when differencing the two country-specific APKs of an app, the differences in third-party code, if any, are also analyzed. Similarly, for native code, our approach handles related differences in calling relationships with native functions, as these are captured in our call graph and call-path diffing.

Importantly, given that FREELENS is an automated tool, it is critical to evaluate its accuracy in identifying GFD apps and characterizing their GFDs before conducting our study using the tool. To conduct a rigorous evaluation, we randomly sampled our entire dataset of 21,120 scraped apps(§IV) with 98% confidence level (CL), 5% margin of error (ME), and population proportion of 5% (given that 1,122 of the 21,120, or approximately 5%, were eventually identified as GFDs). This led to 103 apps as a statistically representative sample. Around these 103 apps, we build ground truth including (1) whether each app is a GFD app according to one of its country-specific APK/version pairs, (2) the code (call-path) differences between that pair, and (3) the summaries of feature differences and security/privacy implications for the same pair. Our ground truth creation involved a meticulous manual analysis pipeline: an in-depth app code review by comparing code-level differences between the two versions, reviewing their release notes and relevant documentation (e.g., apps' About and Data-Safety pages on Play Store, Google's official policies/regulations), and conducting extensive functionality testing of each APK by manual app exploration on physical devices. To ensure reliability and mitigate bias, three of the authors each independently created the ground truth, followed by cross-validation and a consensus process.

TABLE I: Effectiveness of FREELENS

Capability	Precision	Recall	F1 Score
GFD classification	98.97%	100%	99.48%
GFD characterization	92.09%	89.31%	90.68%

Using the curated ground truth, we assessed the accuracy of FREELENS in (1) identifying GFD apps (i.e., *GFD classification*) and (2) generating the summary of feature differences and that of their security/privacy implications (i.e., *GFD characterization*). For classification, the evaluation was automated by exactly matching FREELENS produced labels against the ground-truth. For characterization, three of the authors, plus an expert in Android development served as external annotator to strengthen labeling independence and avoid bias, manually validated the results based on the semantic equivalency of FREELENS produced GFDs and security/privacy implications against respective ground truth. First, each rater independently performed the evaluation. Then, per-rater results are cross-checked hence reaching consensus by resolving disagreement via negotiation. The Fleiss' kappa score [17] computed for this process is 0.78, indicating a substantial level of agreement. Table I lists the accuracy results. As shown, FREELENS can almost perfectly detect GFD apps—only one app was misclassified as its two given APKs have spurious call-path diffs due to their inconsistent obfuscation configurations. The results on GFD characterization show that FREELENS can effectively identify the vast majority of actual GFDs—the high recall suggests that FREELENS's approach of analyzing call paths and leveraging unobfuscated Android framework methods effectively captures most meaningful feature differences. Meanwhile, the strong precision indicates that our two-level mapping strategy successfully filters out coincidental differences and correctly identifies security-relevant variations.

IV. Study Methodology

To systematically investigate GFDs and their SPIs, we formulate five research questions (RQs) below:

RQ1 *How prevalent are GFD apps across regions and time?*
RQ2 *What are the common characteristics of GFDs?*
RQ3 *What are the SPIs of GFDs?*
RQ4 *How do the GFDs manifest in widely-used apps?*
RQ5 *Are GFDs aligned with relevant policies/regulations?*

Next, we describe the dataset collected and procedure followed for answering the RQs using FreeLens.

A. Dataset

TABLE II: Studied countries with their FHIF level/score

Country	Level	Score
USA	Free	76
Germany	Free	77
India	Partly Free	50
Bangladesh	Partly Free	40
Nigeria	Partly Free	59
Egypt	Not Free	28
Saudi Arabia	Not Free	25
Vietnam	Not Free	22
Turkey	Not Free	31
Pakistan	Not Free	27

Countries. To study GFDs, we need first to determine target geographic regions. Similarly to relevant prior studies [2], [18], we use the Freedom House's Internet Freedom (FHIF) score [19] as our primary selection criterion. This score provides a comprehensive metric that considers factors such as access restrictions, content limitations, and violations of user rights in different countries. Unlike the earlier study employing VPN services and physical devices across numerous countries [2], we opted for a more focused approach using registered Google accounts with associated payment methods, as documented in Google's policy of switching Play Store country [20]. This methodology, while covering fewer countries, ensures more reliable and consistent access to apps across regions. We carefully selected 10 countries that represent the full spectrum (i.e., 3 FHIF levels) of Internet freedom, as Table II shows. Note that we do not include markets where Google Play is blocked or where Play services have been disrupted in ways that impede stable access (e.g., Russia). We further emphasize that FHIF is used as one organizing lens for grouping countries, and our analyses are correlational rather than causal; other indicators (e.g., GDP, ecosystem size, alternative app stores) are complementary factors and remain future work.

Apps. For app selection, we employed a multi-pronged approach to ensure both breadth and depth in our dataset. First, we analyzed metadata from AndroZoo [21] to identify apps meeting two criteria: (1) $\geq$1 million installations, indicating significant user impact and (2) availability across all our target countries, eliminating apps with geographic restrictions that could confound our analysis. This focus on widely-installed apps improves cross-region comparability and ecosystem impact, but it may under-represent niche/localized apps that are not broadly distributed across our target countries. This yielded **20,211** apps.

To capture both current trends and historical significance in the Android ecosystem, we supplemented the above base dataset with two additional app sets: (1) top-10 trending apps from all 54 functionality categories, in total 540, on Google Play Store at the current time, and (2) top-500 most installed apps in Google Play Store history [22]. To maintain dataset independence and prevent duplicate analysis, we carefully filtered the base dataset to exclude apps present in either of these two supplementary sets. Similarly, we ensured no overlap between (1) and (2)—after removing 131 duplicates, **909** unique apps remain in these two sets. Therefore, the total number of apps we scraped is **21,120**. This meticulous process resulted in 3 distinct, complementary datasets that collectively provide a comprehensive set of apps. As in prior work [2], we scraped *latest versions* of each app per country. Particularly, we took the latest versions throughout October, 2024.

B. Procedure

We fed FreeLens with the 21,120 scraped apps, resulting in 1,902 unique apps with code differences (i.e., apps with at least one pair of their country-specific APKs being different in the set of API-bounded call paths). However, for 780 of these apps, the differences are only in third-party code not used by the apps yet. Ruling out these apps led to **1,122** GFD apps. For these GFD apps, there are in total 2,347 unique GFDs and 1,744 unique SPIs behind them considering all combinations of country pairs. Then, among the 1,122*C(10,2)=50,490 pairs of (country-specific) APKs, 20,752 have GFDs, for a total of 44,872 individual GFDs, which have 37,177 individual SPIs. Then, our investigation of GFDs follows a principled procedure to address each RQ.

For **RQ1**, which examines the prevalence and distribution of GFD apps, we first analyzed the version differences distribution between country pairs, showing what percentage of examined apps have different versions between each pair of countries. We labeled the country pairs using their Internet freedom levels to identify potential patterns between freedom levels and GFD prevalence. For temporal distribution analysis, we examined the release years (i.e., when these apps were first published on the Google Play Store) of GFD apps.

For **RQ2**, which examines the characteristics and patterns of GFDs, we employed an open coding approach [23] to categorize functionality differences, first creating a codebook and then using it to label GFDs. To create the GFD codebook, we randomly sampled 333 out of all the GFDs, a sample size of 95% confidence level (CL) and 5% margin of error (ME) (with a conservative 50% population proportion). Then, three of the authors independently analyzed the sampled GFDs to create initial categorization schemes. For functionality differences, each author (1) examined the code-level changes identified by FreeLens, (2) assessed whether each GFD fit existing categories, and (3) created new categories when needed. When establishing a new category, they defined a descriptive label, provided detailed category criteria, and included example code patterns characteristic of that category. The authors then resolved any categorization disagreements via discussion until reaching consensus on the final 15 functionality categories (F1-F11) as shown in Table III. Note that for the ease of presentation, we labeled the GFD categories in the non-ascending order of their prevalence (percentage) among all individual GFDs (e.g., we labeled the most dominating category as F1, the next as F2, and so on. We labeled the SPI categories in a similar way, based on the prevalence of each category among all individual SPIs.

After establishing the codebook, we categorized all identified GFDs. To ensure reliable categorization (i.e., coding),

TABLE III: Main Categories of GFDs Found in Our Study

GFD Category (label)	Description (*what features are changed and how they are changed*)
User Interface & Experience (F1)	Modified user interface experience through dialog prompts, navigation controls, browser interactions, etc.
Core System & Infrastructure (F2)	Refactored core system components including app initialization, intent handling, runtime configurations, etc.
Monetization & Advertising Systems (F3)	Added/removed advertising and monetization features including ad SDK, purchase flows, and consent handling.
Content & Media (F4)	Changed media handling capabilities through audio controls, image loading, video playback, etc.
Security & Access Control (F5)	Altered user privacy and security through permission controls, consent management, authentication, etc.
Data & Analytics (F6)	Modified analytics tracking and data management through legacy integrations and state persistence.
Specialized Features (F7)	Mixed feature modification including context management, storage access, encryption mechanisms, etc.
Communication & Messaging (F8)	Added/removed communication features through push notifications, social login flows, in-app messaging, etc.
Business & Commerce (F9)	Modified payment and billing systems with subscription flows, authentication methods, and purchase handling.
Development & Quality (F10)	Optimizing app performance through background processing, lazy loading, memory management, etc.
Device & Hardware Integration (F11)	Added/removed camera, barcode scanning, and location-related hardware functionalities.

we employed negotiated agreement [24]. The three authors who developed the codebooks worked together to categorize each GFD, reaching consensus through discussion when initial categorizations differed. For labeling (remaining) GFDs, this process involved examining code changes to identify characteristic patterns matching our established categories.

For **RQ3**, we followed a similar coding approach. Using the same statistical parameters as for **RQ2**, we determined a significant sample size of 323 cases to create the SPI codebook, as shown in Table V. During the coding process, for labeling remaining SPIs, we analyzed how each functional difference might impact security properties of the respective apps, such as data protection, user privacy, and system integrity. When the differences exhibited multiple characteristics, we categorized them according to their primary impact while noting secondary effects for further analysis.

To address **RQ4**'s focus on code-level manifestation of GFDs in widely-used apps, we conducted detailed code analysis examining the call path differences identified in **Phase 2** of FREELENS. We studied multiple apps from each major category identified in **RQ2**, analyzing their specific code patterns and technical mechanisms for implementing GFDs.

To answer **RQ5**, we focused on apps exhibiting privacy-sensitive variations identified in our previous analysis. For each selected app, we conducted a thorough documentation review including: (1) privacy policies data safety declarations from both the Google Play Store and developer websites, (2) release notes describing version changes, and (3) any region-specific documentation or user notifications. We compared these documented policies against observed code-level differences to identify potential discrepancies between what are stated and actual implementations across regions.

All large-scale quantitative results for RQ1–RQ3 are produced automatically by FREELENS over the full dataset. Manual inspection is only used for (i) the 103-app ground-truth evaluation and (ii) selecting and explaining a small number of representative illustrative case studies (RQ4/RQ5), which are drawn from FREELENS's automated outputs.

V. PREVALENCE OF GFD APPS (RQ1)

5.3% of the (21,120) examined apps are GFD apps. This overall presence can be zoomed in *geographically* (across countries and freedom levels) and *longitudinally* (over years).

Geographic prevalence. Figure 3 shows the percentages of the examined apps that are GFD apps for each of the country pairs studied. For instance, 4.2% of the examined apps have GFDs between Pakistan and USA, and 1.2% between Germany and Pakistan. We also color the countries as per their freedom levels (e.g., green for *free*) as in Table II.

When paired with any other countries, some of the countries (e.g., USA, India, Vietnam, and Turkey) have a generally higher percentage of GFD apps, while some (e.g., Germany, Bangladesh, Nigeria, Egypt, and Saudi Arabia) have notably lower, than the average. A potential reason for this contrast may be *market-specific requirements/preferences*. For instance, USA's large/competitive market [25] may lead to experimentation with features/functionalities to cater to specific user segments, while India and Vietnam's diverse languages and socio-economic demographics may necessitate additional localization and tailored features. The more homogeneous user preferences, which reduce the need for country-specific features, may justify the low prevalence of GFD apps in respective countries (e.g., Germany, Egypt). Another potential factor could be *regulatory and legal influences*. For instance, countries such as the USA, India, and Turkey operate under national-level regulatory regimes that differ substantially from the more harmonized GDPR-based requirements observed across many European countries; this divergence across national legal systems may contribute to GFD prevalence when comparing these countries [26], [27].

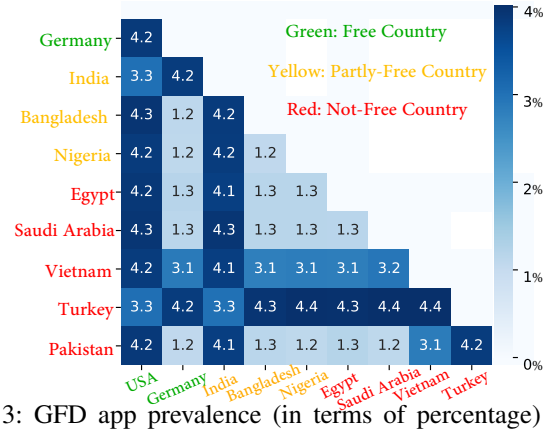

Fig. 3: GFD app prevalence (in terms of percentage) across country pairs and internet freedom levels.

The low GFD app presence in countries like Germany and Saudi Arabia can be due to their enforcing uniform standards across app versions to simplify compliance. Disparities in terms of *economic/technical factors*, *developer strategies*, and *localization complexity* can further explain the contrast.

Yet, there is no consistent pattern between the country pairs with the same and different freedom levels. First, the percentages of GFD apps between countries at different levels are not always higher or lower than those between countries at the same levels. Second, across different levels, such percentages are not necessarily higher (lower) between countries with larger (smaller) level gaps, nor the other way around.

Longitudinal prevalence. Figure 4 shows the percentages of GFD apps among examined apps released (first published) in different years on the Google Play Store. As shown, the GFD apps studied span release years from 2010 to 2023.

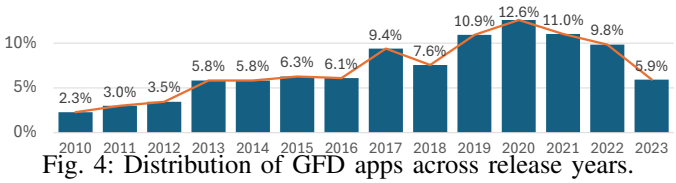

Fig. 4: Distribution of GFD apps across release years.

Overall, the past 14 years have seen growing GFD app prevalence. It started off relatively low in the early years (2010-2012), due to the nascent app market and limited global smartphone adoption—Android was officially launched in 2008. The prevalence picked up in the next four year (2013-2016), as app stores expanded region-specific features while smartphone penetration grew globally and localization (e.g., languages, cultural adjustments) became a competitive necessity. During 2019–2021, the prevalence peaked, potentially driven by heightened app usage (e.g., partly due to the COVID-19 pandemic), stricter local regulations (e.g., data privacy, content standards), and improved tools enabling easier development of geo-specific versions.

The lower percentages for recently released apps (9.8% in 2022 and 5.9% in 2023) plausibly reflect that newer apps have had less time to reach the study's installation threshold and establish regional variations, rather than indicating an actual decline in geographic feature customization practices.

This distribution provides important context for understanding the temporal scope of our dataset, as we conduct deeper analysis of GFD patterns and trends in subsequent sections.

While geographic variations are common, they do not follow simple patterns based on internet freedom levels and specific country pairs, and their prevalence has increased significantly over time, particularly in recent years.

VI. Characteristics of GFDs (RQ2)

We analyze GFDs from two perspectives: *unique GFDs*, which count distinct feature differences between app versions (i.e., country-specific APKs), and *individual GFDs*, which account for all country-pair combinations where these differences occur. For example, if an app has two versions distributed across four and six countries respectively, three *unique* GFDs would result in 72 *individual* GFDs (24 country pairs × 3 GFDs). We also explore the distribution of *unique* GFDs across functionality categories and *individual* GFDs across country pairs and app release years.

Categories of GFDs. We identified **11** GFD categories, as shown in Table III. How the unique GFDs we discovered are distributed across these categories is depicted in Figure 5, where we also show the percentage of GFD apps that have GFDs of each category (noted as *app coverage*).

User Interface & Experience (**F1**) stands out, not only accounting for the largest share of GFD entries (21.9%) but also exhibiting the highest app coverage at 41.2%. This pattern suggests that, while developers might prioritize a consistent *global* look-and-feel, regional UI/UX customizations (e.g., layout modifications and user flow adjustments) remain prevalent. Intuitively, user-facing elements are highly sensitive to regional preferences, cultural norms, and language differences; thus, developers often tailor UI/UX to improve accessibility, usability, and engagement for diverse user bases [28].

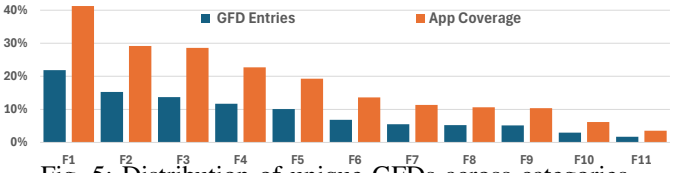

Fig. 5: Distribution of unique GFDs across categories.

Core System & Infrastructure (**F2**) related feature variations are the next dominant kind of GFDs, found in nearly 30% of the GFD apps. Looking into specific cases revealed that these GFDs manifest in how certain foundational services (e.g., background process management, storage handling) are enabled or disabled for particular country-specific releases, typically reflecting legal or performance-driven constraints in different locales. Thus, this dominance may be explained by differences in network infrastructure, compliance requirements (e.g., data storage, encryption), and regional backend services (e.g., payment systems, cloud providers), which necessitate modifications in app architecture and system-level functionalities [29]. Similarly, Monetization & Advertising Systems (**F3**) ranks high, indicating that developers systematically tailor subscription plans, ad placements, or payment gateways to conform with local revenue models.

In contrast, categories such as Device & Hardware Integration (**F11**) account for the smallest fraction of GFD entries and app coverage. A main reason may be that deeper OS-level or hardware-driven functionalities are comparatively more uniform across regions——either because device integrations are heavily constrained by the Android platform itself or because these features translate poorly to smaller or region-specific user segments and thus see fewer variations. In fact, Android enforces a standardized API for hardware and device-level functionalities (e.g., sensors, camera, Bluetooth); these APIs provide consistent behavior across devices to ensure compatibility with the diverse Android ecosystem [30]. Regional disparities in mobile hardware adoption also influence the usage of hardware-dependent app features [29]. For similar reasons, Specialized Features (**F7**) and Communication & Messaging (**F8**) are also less prominent, though they still appear in more than one in ten apps.

Local tailoring in GFD apps primarily focuses on user interfaces, core infrastructure, and monetization—areas that are easier to adapt. In contrast, GFD categories that require extensive low-level modifications or rely heavily on uniform global services, such as deeper system, specialized features, and device-level integrations, appear less frequently.

Distribution of individual GFDs. Comparing the category distribution of individual GFDs (shown in Figure 6) to that of unique GFDs (in Figure 5) reveals that different types of feature variations between regions tend to recur across multiple apps with similar frequencies. For instance, individual F1, F2, and F3 GFDs have similar dominance (21.9%, 15.2%, and 13.9%, respectively) to those among unique GFDs. This implies that certain types of features are consistently targeted for regional customization across apps, and it is justifiable. For example, UI/UX (F1) and core infrastructure (F2) are inherently flexible and responsive to regional preferences or constraints, leading to their repeated appearance across apps. Another rationale is that developers often address common regional requirements (e.g., language, payment systems, or regulatory compliance) in similar ways across different apps [31], [32], [29], resulting in overlapping patterns of feature variations.

Category distribution of individual GFDs largely resembles that of unique GFDs, suggesting that feature variations are consistently adapted across apps for regional needs.

GFDs across Internet freedom levels. Our results in RQ1 revealed no clear correlation between these levels and the prevalence of GFD apps. Now we explore this further but looking at *individual* GFDs. Figure 6 shows the distribution of these GFDs across different country's freedom level pairs. For a balanced view/comparison, we collapse the three levels into two more sharply contrasted ones: with freedom (including free and partly free, or **FPF**) vs. without (i.e., no freedom, or **NF**), so that each level is represented by 5 countries.

The results suggest a clear impact of freedom levels on the prevalence of individual GFDs: consistently within any GFD category, the majority (e.g., 11.8% for F1) of GFDs are between countries of FPF vs. NF, largely outstripping those within the same levels (e.g., 5.2% for FPF and 4.9% for NF, in F1). This contrast suggests a strong correlation between differing political, regulatory, and cultural environments—reflected in the varying freedom levels—and patterns of app localization and functional feature variations in Android. One possible interpretation is that these contextual factors may shape how developers adapt app functionalities across regions, which may be explained in two ways. First, countries with no freedom often impose stricter controls on content, privacy, and app functionalities, which may require developers to make significant modifications [33], while those with (full or partial) freedom often have more diverse and open markets, requiring apps to align with varied cultural norms, languages, and monetization strategies [28], [29]. Second, countries within the same freedom level tend to share similar legal and cultural expectations [34], [35], which may reduce the need for localized feature differentiation.

We further examined how FHIF levels relate to the prevalence of GFDs by aggregating, for each feature category, the percentage of total GFD instances observed among country pairs grouped by freedom level: within free/partly-free (FPF–FPF), within not-free (NF–NF), and cross-freedom (FPF–NF). For each group, we computed the mean and standard deviation (SD) of these percentages to capture overall prevalence and variability across categories. We also derived a relative ratio (FPF–NF vs. FPF–FPF) to express the magnitude of cross-freedom disparity, a Cohen's d to quantify the standardized effect size between these two distributions, and a Pearson correlation coefficient (r) to measure how each pair type aligns with the overall GFD distribution. As summarized in Table IV, cross-freedom-level comparisons (FPF–NF) exhibit a markedly higher prevalence of GFDs than within-freedom-level comparisons. The large effect size and strong correlation indicate that cross-freedom disparity strongly aligns with functional variation intensity, reinforcing that Internet-freedom differences are a major correlate of GFD prevalence, even though causality cannot be asserted.

TABLE IV: Statistics of GFD Distribution by Freedom-Level Pair

Metric	FPF–FPF	FPF–NF	NF–NF
Mean (%)	2.13	6.45	2.00
Standard Deviation (SD)	1.23	2.89	1.04
Relative Ratio (vs. FPF–FPF)	1.00×	3.03×	0.94×
Cohen's d (FPF–NF vs. FPF–FPF)	-	2.05	-
Pearson Correlation with Overall (%)	0.94	0.99	0.92

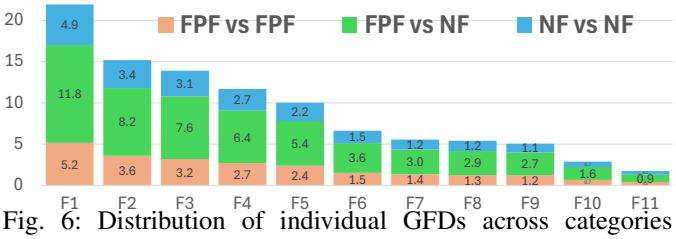

Fig. 6: Distribution of individual GFDs across categories stacked with country pairs of Internet freedom levels.

Both the depth and breadth of GFDs increase as the gap in Internet freedom widens, with UI/UX related GFDs leading the way. For the prevalence of individual GFDs, having freedom or not matters less than whether having the same freedom level between the respective countries.

GFDs across (app first release) years. To further understand how the (individual) GFDs evolve over time, both in total and different categories, we examine the temporal breakdown of GFD apps first released in each year over top-5 dominating categories while collapsing the rest as `Others`, as shown in Figure 7. In terms of the total GFDs, the evolutionary pattern resembles, and can be intuitively explained by, that of the GFD apps (Figure 4). Given the overall growing trend of GFD app prevalence, total feature differences (GFDs) grow in magnitude accordingly since the GFDs stem from associated GFD apps. Meanwhile, this similarity implies that the GFD apps contribute to individual GFDs almost evenly.

Now looking into the evolutionary characteristics of specific GFD categories, our temporal analysis reveals distinct patterns in Android app regionalization from 2010 to 2023. Overall, for most of the years that we studied, the yearly prevalence ranking of these categories is largely consistent with the

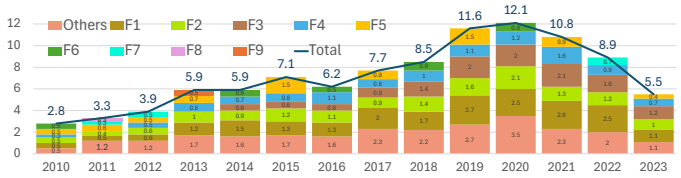

Fig. 7: Distribution of individual GFDs across years.

general ranking across all years (Figure 6), while in the other (mainly, earliest and recent few) years the pattern deviated.

Specifically, early apps (2010) showed minimal regional variations distributed evenly across the GFD categories, with even the overall-top-3 categories (F1, F2, and F3) accounting only for 0.5% of all GFDs, suggesting a basic, more uniform approach to regional customization. A significant evolution occurred between 2015-2018, with F1 variations increasing from 1.3% to 2.3% and F2 from 0.9% to 1.4%, indicating growing sophistication in regional adaptations. This trend peaked during 2019-2021, when F1 reached 2.5-3.5%, F2 rose to 2.0-2.6%, and F3 to 1.6-2.1%, demonstrating the most comprehensive approach to regional customization. Recent apps (2022-2023) show more balanced but lower variation levels across all categories, potentially indicating either streamlined adaptation strategies or the emergence of new customization mechanisms beyond our analysis framework. This evolution reflects developers' growing understanding of regional adaptation needs, balanced with implementation efficiency.

Illustrations of GFDs. We present several examples with GUI screenshots that help illustrate the visible GFDs to better understand each GFD category. We selected Top-5 categories, apart from F2, where most are internal modifications that are not perceivable to end-users on GUI level.

User Interface & Experience (F1) variations often manifest in how apps adapt their interfaces to accommodate different regional preferences and cultural norms. These modifications can include changes in layout structure, navigation patterns, and content presentation. For example, in social gaming app `in.dradhanus.liveher` [36], we observe distinct UI layouts between its USA and India versions, as seen in Figure 8. The India version employs a simpler, more structured grid layout for game selection with clear categorization ("Match game", "Game room"), while the USA version features a more dynamic interface with multiple tabbed sections ("Top Room", "Top Sender") and a mixed layout combining different content types. These variations reflect different regional preferences in content organization and navigation patterns - the Indian version prioritizing straightforward game access and the USA version emphasizing social features and user rankings. Notably, both versions maintain the core gaming functionality while adapting their presentation to better align with local user expectations and interaction patterns.

Monetization & Advertising Systems (F3) differences represent how apps implement varying revenue generation strategies across regions. These variations are particularly evident in advertising implementations, where the same app often employs markedly different advertising approaches across regions. A common pattern we observe is the contrast between test advertisements and sophisticated interactive ads, as seen

Package Name: in.dradhanus.liveher
Countries: USA(left)/India(right)

Fig. 8: An illustrating example of GFD category F1 (User Interface & Experience).

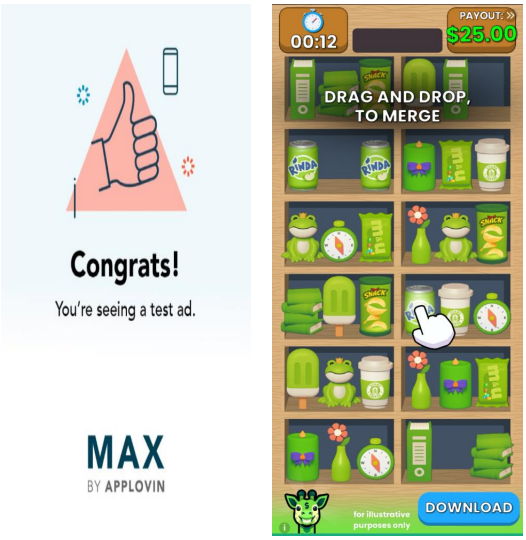

Fig. 9: An illustrating example of GFD category F3 (Monetization & Advertising Systems).

in Figure 9. In many apps, one regional version displays simple test ad placeholders (typically showing "You're seeing a test ad" with minimal interaction), while another version uses complex, interactive advertising experiences by means of third-party advertisement library. For example, as shown in the screenshots, one version shows an engaging "drag and drop" merge game as an advertisement, complete with timer and reward indicators ($25.00 payout), while its counterpart in another region receives only a basic test ad with a thumbs-up illustration. This stark contrast in advertising sophistication reflects regional differences in monetization strategies, likely influenced by factors such as market maturity, user engage-

ment patterns, and advertising revenue potential in different geographical regions.

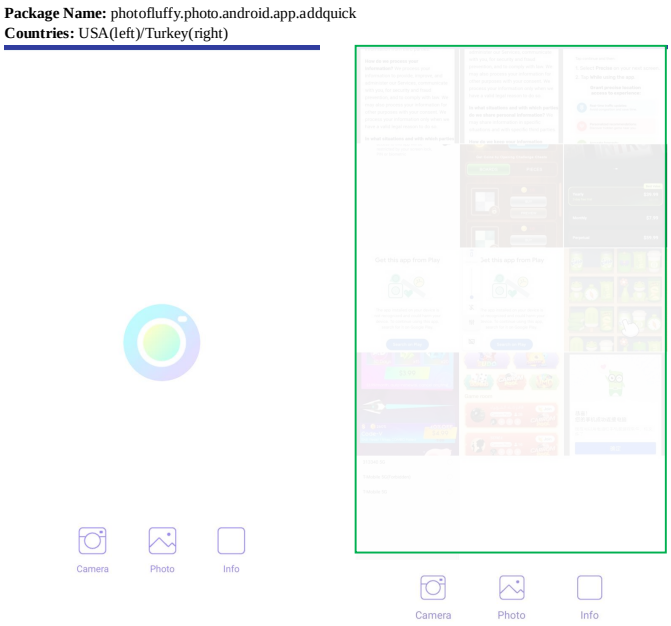

Fig. 10: An illustrating example of GFD category F4 (Content & Media).

Content & Media (F4) modifications primarily involve differences in how apps present and organize their media content across regions. These variations can include different content layouts, media organization strategies, and viewing options. This is exemplified in `photofluffy.photo.android.app.addquick` [37], as seen in Figure 10 where comparing its USA and Turkey versions reveals distinct approaches to media content presentation. The Turkey version implements a grid-based image gallery view directly on the homepage, providing users with an immediate overview of their location-related media content. In contrast, the USA version maintains a simpler interface without the grid view, requiring users to access media through dedicated camera and photo sections. This variation demonstrates how apps adapt their media presentation strategies to align with regional user preferences and consumption patterns - the Turkish version favoring immediate visual content access while the USA version opts for a more streamlined, navigation-based approach to media content.

Overall, GFDs experienced a long-term continuous growth both in total and in specific categories. Geographic customization in Android apps evolved from simple adaptations to more sophisticated regional variations, culminating in the most comprehensive adaptations across multiple categories.

VII. GFDs' Security/Privacy Implications (RQ3)

Similarly to RQ2, we explore the distribution of *unique SPIs* across SPI categories and *individual SPIs* across country pairs and app-release years. Additionally, we examine the relationship (*mapping*) between GFDs and SPIs.

Categories of SPIs. We identified **15** distinct categories of SPIs, as shown in Table V. In a format similar to Figure 5, Figure 11 depicts how the unique SPIs are distributed over these categories, as well as the app coverage (i.e., percentage of GFD apps that have SPIs of each category).

Data Security & Privacy (**S1**) dominates all other categories, accounting for 24.0% of all SPI entries and exhibiting the broadest app coverage at 33.4%. This highlights the prevalence of variations in data handling policies and storage practices across locales, potentially driven by region-specific regulations such as Europe's GDPR, USA's CCPA, and other localized privacy laws—which mandate app-specific compliance [34]. Authentication & Access Control (**S2**) ranks second, both in the dominance (13.6%) and app coverage (19.4%), potentially reflecting frequent customization of credential management to meet local legal requirements, or adapt to varying user risk profiles and regional differences in authentication norms (e.g., multifactor authentication requirements) [38].

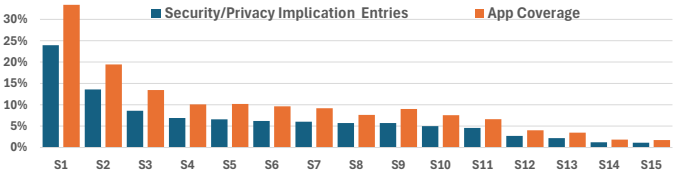

Fig. 11: Distribution of unique SPIs across categories.

Privacy Compliance & Consent (**S3**) follows, highlighting the regional adjustments required for explicit disclosures and opt-in processes, such as additional consent forms in jurisdictions with stricter privacy regulations. Meanwhile, System & Infrastructure Security (**S4**), Monitoring & Analytics (**S5**), and Storage & Resource Management (**S6**) exhibit moderate coverage, indicating that while infrastructure-level settings like logging and data retention are less frequently altered, they still undergo meaningful cross-country adjustments.

On the opposite end, Browser Security (**S14**) and Software Management (**S15**) account for the smallest share of SPI entries and app coverage, likely reflecting apps' reliance on standard Android mechanisms for in-app browsing and software updates, reducing the need for region-specific modifications of respective app components. Overall, the prevalence of security and privacy adjustments, particularly in dominant categories, underscores that cross-country variations extend beyond superficial design changes, encompassing essential protective measures and compliance-driven updates with significant implications for user safety and trust.

GFDs have most dominantly data security/privacy implications, reflecting region-specific compliance needs. In contrast, browser security and software management SPIs are least common, indicating related countermeasures are more standardized, hence fewer regional modifications.

Distribution of individual SPIs. Overall, this distribution across the 15 SPI categories is quite consistent with that of unique SPIs, as evidenced in Figure 12 vs. Figure 11. For instance, S1 again emerges as the most frequently differentiated category, making up 24.2% of all individual SPIs, followed by S2 at 13.5% and S3 at 8.5% (vs. 24%, 13.6%, and 8.6%, respectively, in unique SPIs), highlighting their centrality to the SPIs of region-specific feature variations. This consistency reaffirms the dominance ranking of all the SPI categories.

TABLE V: Main Categories of Security/Privacy Implications of GFDs Found in Our Study

Category (label)	**Description (*what security & privacy implications are and how they may be realized*)**
Data Security & Privacy (S1)	Handling sensitive user data biometrics and permissions compromises privacy and security.
Authentication & Access Control (S2)	Changing authentication flows and verification mechanisms like login, payment, messages compromises security issues.
Privacy Compliance & Consent (S3)	Modifying permission handling and consent management affects user authorization and data protection compliance.
System & Infrastructure Security (S4)	Modify system settings, access device information, and manage broadcast receivers impacts device security and privacy.
Monitoring & Analytics (S5)	Modifying monitoring and logging mechanisms affects security auditing, user tracking, and advertisement analytics.
Storage & Resource Management (S6)	Inappropriate system resource management causes unauthorized data exposure, resource leaks, and vulnerabilities.
Advertising Security & Privacy (S7)	Modifying advertising integrations and tracking mechanisms affects user privacy and cause potential attack.
Third-Party Security (S8)	Include third-party integrations weakens system protections.
Transaction Security (S9)	Insecure handling of financial transactions and secure validation causes unauthorized access and data leakage.
Network & Communication Security (S10)	Altering network connections and permissions for ads, licensing, and VPN functionality causes security risks.
UI & Navigation Security (S11)	Altering UI components and navigation flows, such as back navigation, can lead to phishing attacks, unauthorized access.
Content & Media Security (S12)	Modifying access to multimedia content can lead to data leaks, copyright violations, and unauthorized content sharing.
Protection & Prevention (S13)	Changes to protection mechanisms like firewalls, malware detection, and application sandboxing.
Browser Security (S14)	Insecure WebView modification exposes applications and users to web-based vulnerabilities and privacy risks.
Software Management (S15)	Other core feature modification like cryptography and third-party integrations weakens system protections.

Note: The SPI descriptions summarize potential security/privacy implications under common threat models and may depend on app-specific configurations; they should not be interpreted as confirmed vulnerabilities without deeper, app-specific analysis.

It also suggests that unique SPIs of every category recur across multiple country-specific APK pairs of a GFD app and/or across multiple GFD apps in a *largely even/uniform manner*. The rationale lies likely in the standardization of security/privacy practices across regions, aligned with established guidelines and global frameworks for app security [39], [40].

Individual SPIs are distributed consistently with unique SPIs, where no category dominates disproportionately in its recurrence, suggesting systematic, balanced efforts in app localization across all the security/privacy dimensions.

SPIs across Internet freedom levels. While SPIs stem from GFDs, not every GFD results in an SPI as GFDs are not necessarily related to security/privacy. Nevertheless, two similar observations (to those for GFDs) on the distribution of individual SPIs over the two collapsed freedom levels can be made from Figure 12. First, both across all and within individual SPI categories (including the minor ones), FPF vs. NF comparisons consistently account for the largest share of cross-country security/privacy variations (e.g., 13.2% for S1), substantially outpacing the other two level pairs. Second, the latter two level pairs generally have very close shares (e.g., 5.7% FPF vs. FPF and 5.3% NF vs. NF, in S1).

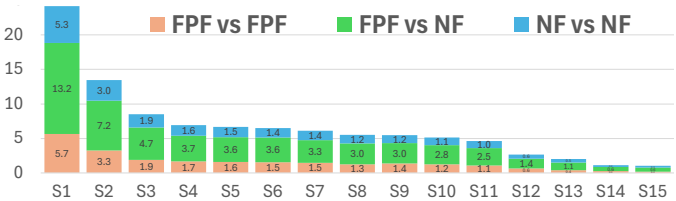

Fig. 12: Distribution of individual SPIs across categories stacked with country pairs of Internet freedom levels.

These patterns indicate that, in terms of how Internet freedom levels impact the SPIs of functional feature variations, level gaps matter more than specific levels, similarly to what we found for GFDs, which may be justified in two parts. First, FPF countries (e.g., those governed by GDPR in Europe) enforce stricter security/privacy standards, whereas NF countries often lack similar frameworks, leading to substantial adaptations [34]. Also, shared infrastructural and regulatory frameworks in NF regions (e.g., censorship-focused or surveillance-driven policies) and FPF regions (focused on user autonomy and privacy rights) appear to correlate with fewer SPI adjustments within each group [41].

Freedom level differences significantly influence security/privacy adaptations, while there are fewer variations within each freedom category due to shared constraints.

SPIs across (app first release) years. Similar to how we examine the evolutionary characteristics of GFDs (Figure 7), we examine the same for SPIs, focusing on the top-5 SPI categories while collapsing the rest as `Others` for each year. The results, shown in Figure 13, reveal the overall trend of SPIs which mirrors, and can be explained by, that of GFDs as well as the general distribution pattern of GFD apps (Figure 4), reflecting how adjustments in security/privacy considerations have evolved alongside functionality feature variations. For one thing, the growing importance of regional regulatory compliance, such as GDPR and sector-specific privacy laws. A plausible explanation for the generally rising GFD and SPI trends over time could be the growing importance of regional regulatory compliance [34]. For another, increasing user awareness of privacy/security issues amplifies the need for SPI adaptations in tandem with feature-level customizations [42].

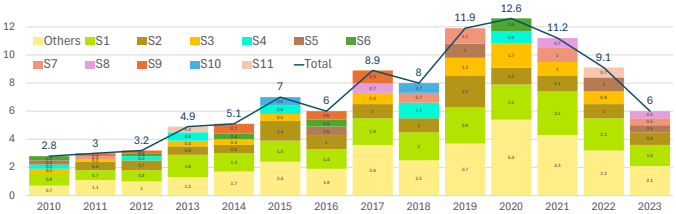

Fig. 13: Distribution of individual SPIs across years.

Early apps (2010–2012) exhibited minimal regional security adaptations, with S1 and S2 each accounting for only 0.7–0.9% of SPIs, reflecting simpler security needs across regions. From 2015–2017, SPIs began growing significantly, with S1 rising from 1.4% to 2.0%, and S3 emerging as a notable category (0.6% to 1.0%), likely driven by anticipation of stricter privacy regulations. The peak SPI concentration occurred during 2019–2021, with S1 reaching 3.1%, S2 climbing to 2.5%, and S4 growing to 1.7%, signaling heightened attention to data protection and system-level safeguards. S3 also maintained a strong presence at 1.2%, reflecting sustained compliance efforts. In 2022–2023, SPI distribution became

more balanced across categories, albeit at reduced levels, with S1 still leading at 1.5%. This shift suggests a transition toward a more holistic security approach, reflecting mature development practices and broader awareness of regional security/privacy requirements.

The temporal trend underscores a clear evolution in addressing security/privacy in regional app variations, moving from basic safeguards to nuanced, multi-dimensional strategies that align with advancing regulations and user expectations.

SPI evolution follows GFD trends, reflecting how geographic feature variations inherently incorporates security/privacy considerations. While dominant SPIs in regional adaptations remain consistent across app ages, specific implementation approaches and emphasis areas have evolved.

Mapping from GFDs to SPIs. To see which functional feature variations underlie which security/privacy adaptations, we identified 140 types of connections between the 11 GFD and 15 SPI categories, totaling 31,916 connection instances across all regional pairs, as Figure 14 shows. The mapping reveals critical patterns in how GFDs translate to SPIs.

First, there are a few strong, high-volume GFD-SPI connections. The most prominent mapping is between F5 and S2, with over 2,400 observed connections. This underscores the direct influence of functionality variations in user-security features, such as login mechanisms, multi-factor authentication, and session management, on corresponding security adaptations. Such alignment may be justified by the essential need for mitigating region-specific risks or addressing compliance requirements tied to credential management (e.g., password policies and biometric authentication) [43].

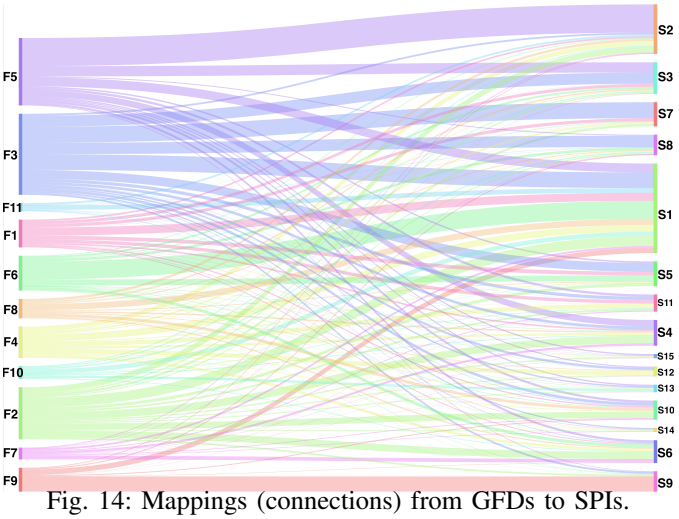

Fig. 14: Mappings (connections) from GFDs to SPIs.

Another substantial connection appears between F3 and S7, reflecting the prevalence of region-specific ad integrations that trigger corresponding compliance or protective measures. In the same vein, F3 also shows sizable flows toward S1 and S3, indicating that local monetization strategies often demand adjustments to user-data handling and disclosure practices [34].

Second, GFDs in core functionalities drive multi-domain SPIs. For instance, F2 exhibits large flows toward multiple security/privacy domains—most notably S1, S10, and S4. This trend reflects the foundational role of app infrastructure in determining security behaviors, which can be justified by that regional variations in foundational app behaviors (e.g., resource management, OS-level permissions, network protocols) often necessitate robust protective measures tailored to local operating environments [44]. Similarly, F6 shows frequent connections to S1, suggesting that cross-country functional variations in how apps collect/process user data often translate directly to heightened privacy and compliance concerns [45].

In contrast, categories like S14 and S15 receive minimal flows, consistent with earlier observation of their reliance on globally standardized frameworks. They are less influenced by regional distinctions because they often depend on Android's core mechanisms or widely accepted global standards, leaving limited scope or necessity for regional adaptations.

TABLE VI: Chi-square test comparing the distribution of SPI categories between each two years. (***: p<0.001, **: p<0.01, *: p<0.05)

	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023	2024
2010	0.341	0.853	0.520	0.314	0.129	0.152	0.057	0.226	0.098	0.071	0.058	0.014*	0.024*	0.977
2011		0.675	0.621	0.740	0.742	0.974	0.975	0.901	0.711	0.508	0.759	0.084	0.668	0.997
2012			0.798	0.627	0.219	0.353	0.778	0.504	0.311	0.270	0.168	0.029*	0.180	0.981
2013				0.673	0.300	0.175	0.470	0.234	0.679	0.867	0.276	0.492	0.384	0.979
2014					0.848	0.631	0.702	0.328	0.956	0.294	0.751	0.068	0.761	0.995
2015						0.803	0.133	0.663	0.792	0.009**	0.867	0.026*	0.657	0.993
2016							0.340	0.597	0.786	0.014*	0.512	0.008**	0.805	0.994
2017								0.637	0.357	0.642	0.376	0.044*	0.547	0.997
2018									0.326	0.029*	0.523	0.028*	0.561	0.999
2019										0.202	0.899	0.334	0.840	0.988
2020											0.120	0.567	0.343	0.993
2021												0.098	0.725	0.996
2022													0.293	0.989
2023														0.997

Overall, SPIs and GFDs are strongly connected in a many-to-many manner, highlighting (1) the intricate relationship between app functionality and region-specific security/privacy measures and (2) the critical interplay between functional differences and the security/privacy adaptations required to address the risks, compliance demands, and challenges introduced by these variations in diverse global contexts.

Statistical significance of temporal shifts. To assess whether the apparent changes in SPI patterns across release years are meaningful beyond descriptive trends, we compare the distribution of SPI categories between each pair of years using a Pearson χ^2 test. Table VI reports the resulting pairwise p-values. The results indicate that several comparisons involving recent years are statistically significant (e.g., around 2020–2022), suggesting that the composition of SPIs has shifted in more recent app cohorts.

Illustrating Examples of SPIs associated with GFDs Similar to Section VII, we select examples to illustrate how security/privacy implications arise with GFDs.

Data Security & Privacy (S1) GPS tracker application `com.locator.gpstracker.phone` [46] across USA and Egyptian versions reveals insights into regional approaches to Data Security & Privacy(S1). As seen in Figure 15, despite the UI layout differences, the most prominent functional distinction lies in how the Egyptian version implements a live map view showing real-time user location, a feature absent in the USA version. To enable this functionality, the Egyptian version requires the additional `FOREGROUND_SERVICE_LOCATION` permission, which allows the app to access the user's location while actively running in the foreground.

This GFD in this case raises noteworthy data security and privacy considerations. The foreground service permission ensures that location tracking only occurs when the user is

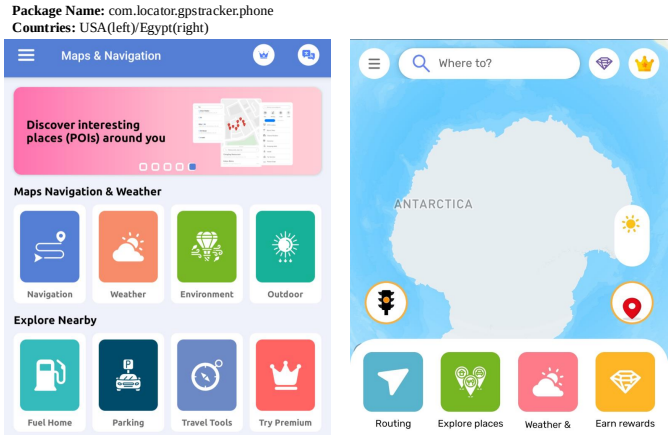

Fig. 15: An illustrating example of SPI category S1 (Data Security & Privacy).

actively using the app, providing a level of transparency in data collection. However, the continuous access to location data during active app usage could still potentially expose users to risks if the collected data is not properly secured or if the service is vulnerable to exploitation by malicious actors attempting to intercept real-time location information.

Authentication & Access Control (S2) Several apps we observed shows some authentication implications. Specifically, a common pattern is the lack of license checking in some country's versions of the same app. For example, `com.pairip.licensecheck3` is a license verification module commonly used in Android apps to validate legitimate app purchases and prevent unauthorized app usage. It provides authentication services to ensure that apps are being used according to their licensing terms.

The absence of this license verification mechanism raises several security concerns. First, without license verification, unauthorized copies of the app could run without validation, potentially violating the developer's intellectual property rights and licensing terms. Second, the verification process includes signature validation using public key cryptography, and its removal eliminates a crucial integrity check that ensures the app hasn't been tampered with or repackaged. Third, the licensing system helps prevent revenue loss by ensuring only legitimate purchases can use the app, and its absence in some versions could lead to unauthorized distribution and usage.

Furthermore, the presence of different authentication mechanisms across versions creates inconsistencies in how the app manages user access and validates installations, as we noticed that this integrity check mechanism is replaced by Play Integrity API [47]. We also observed that most of apps that did not enforce license check will also be blocked by Google Play Service to be installed, which is shown in Figure 16. We will also explore the underlying code difference in Listing 2 in Section VIII-B.

Advertising Security & Privacy (S7) As we have seen an example in Figure 9, the implementation of complex, interactive ads through third-party ad libraries raises several security and privacy concerns. Such interactive advertisements require more extensive permissions and access to device resources to function properly, potentially exposing users to greater data collection and tracking capabilities. These ads often need to monitor user interactions, track engagement time, and collect behavioral data to optimize ad performance and verify engagement metrics.

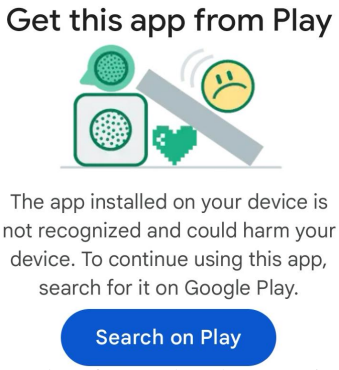

Fig. 16: A example of Google Play Service blocking App without license check.

From a security perspective, the integration of third-party ad libraries expands the application's attack surface. These complex ad implementations could potentially be exploited as vectors for malicious activities, such as clickjacking [48], data harvesting [49], or even malware distribution [50] if the ad network's security is compromised. Additionally, the interactive nature of these ads increases the risk of accidental clicks and unintended user interactions, which could lead to unauthorized data collection or unwanted redirects. For example, recent studies [51] discovered that library-based Ad promotion can be exploited by malicious developers.

GFDs clearly shape the security/privacy landscape of cross-country app variations, with the heaviest SPI concentrations revolving around data handling, authentication, and privacy compliance, aligning closely with legal and commercial pressures in different regions, although certain security/privacy areas like browser security remain standardized.

VIII. Code-Level GFD Manifestations (RQ4)

To understand concrete manifestations of GFDs, we conducted case studies of representative apps, which reveal how common GFDs are implemented at code level.

A. Advertising and Monetization

Our analysis reveals that advertising-related GFDs is significantly prevalent, and often the most discernible one by users. By examining code differences identified in **Phase 2** of FreeLens, we observed several notable patterns in how advertising implementations vary across regions.

Ad Complexity Variations. Some versions of a GFD app implement basic or test advertisements, while others deploy complex advertising systems. This is evidenced by the presence of distinctive advertising-related components in certain versions. An example is given in Figure 9. In the versions with interstitial ads, we see the presence of `com.applovin.adview.AppLovinFullscreenThemedActivity`, `com.vungle.ads.internal.ui.AdActivity`, or `com.unity3d.ads.adplayer.FullScreenWebViewDisplay`, etc. These classes indicate the implementation of advanced ad features such as full-screen interstitial ads

or rewarded video ads. The presence of methods like `dispatchTouchEvent` in these classes suggests interactive ad experiences that can capture and respond to user interactions.A recent study [52] has shown that third-party libraries with interstitial advertising capabilities may have silence installation, inappropriate ads to children, and ad fraud. The optional integration of these libraries exposes these concerns to users in certain regions.

Unexpected Ads. We observed significant variations in both where and when ads appear within apps. A common pattern involves integrating ads into every major user interface component, typically placed at the creation or the destruction of an `Activity` class. In the user's view, it will be displayed when the user enters or leaves some interfaces. For example, in the app `com.locator.gpstracker.phone`, we found extensive differences in back-button handling, as shown in Listing 1. This implementation demonstrates a strategy where interstitial ads are triggered by navigation events, specifically when users attempt to leave an activity using the back button. This approach maximizes ad exposure by intercepting common user interactions. As per Google Play's official policy [53], showing ads that are displayed to users in unexpected ways harms the mobile user experience.

```
onBackPressed()
|_ AdsHelper.showInterBack()
   |_ Admob.showInterAds()
      |_ onAdClosedByUser()
```

Listing 1: An example of ad placement strategy.

Paid Features. A prevalent pattern emerges in apps with consistent core features but varying monetization implementations. In these cases, one regional version typically offers users the option to remove ads or unlock premium features through in-app purchases, while the other version either lacks this option entirely or implements it differently. For example, in apps that support payment options to remove ads, we consistently observe that these versions also implement particularly aggressive advertising strategies.

```
onCreate()
|_BaseBannerAdActivity.S0()
   |_BillingClient.g()
```

Listing 2: An example of access control implication.

The implementation typically manifests through in-app purchase validation mechanisms. Our analysis reveals common patterns where apps integrate with Google Play's billing system to manage feature access. For instance, consider the implementation pattern in Listing 2. This pattern reveals a monetization strategy where ads are conditionally displayed based on in-app purchase status, with the billing client verification determining whether a user should see the ads. This disparity raises questions about equitable access to app functionalities across different geographic regions and highlights how monetization strategies can lead to different user experiences despite apps sharing the same core functionalities.

B. Authentication

```
    onCreate(android.os.Bundle)
    |_onActivityCreate()
      |_initializeLicenseCheck()>
        |_connectToLicensingService()>
           |_handleError()
              |_showErrorDialog()
           |_validateResponse()
              |_ verifySignature()
```

Listing 3: An example of authentication implication.

We noted a common authentication concern in one of the illustrating examples as shown in Figure 16. Now we examine its code implementation. As shown in Listing 3, the code path reveals a license verification process that is present in one version but not in another. During `Activity` creation (i.e., in `onCreate`), the app initializes the license check through `LicenseClientV3`. The client then connects to a licensing service and validates the response through several steps. First, it establishes a connection with the licensing service and receives a response bundle containing license information. Then, it performs response validation using cryptographic signatures, which involves decoding a Base64-encoded public key, verifying the signature of the license response, and validating the JSON response containing license status. If any errors occur during this process, the app shows an error dialog and handles the exception appropriately.

GFDs are strongly manifested in app code implementations, and the regional variations link clearly to the associated SPIs, via user interaction and authentication, among others.

IX. GFD's Policy/Regulation Alignments (RQ5)

Our analysis of how GFDs align with relevant policies and regulations reveals concerning disparities between implemented features and their disclosure to users. We manually examined privacy policies and documentation of apps exhibiting privacy-relevant GFDs, particularly those categorized under Data Security and Privacy (S1) and Permission and Consent (S3), focusing on three key aspects: privacy policy disclosures, regulatory compliance, and platform policy adherence.

Data Privacy. The optional integration of third-party libraries raises concerns as they might violate data privacy [54] [55] [56].A representative case highlighting these issues is `com.wildberries.ru` [57], one of the largest online retail store based in Russia and has over 100 million downloads. It incorporates the Sentry [58] crash analysis functionality in one of its regional versions. Sentry provides detailed crash diagnostics via session replay, which captures user interactions leading to crashes, including sequential screenshots of the user interface, as shown in Listing 4. While Sentry's default configuration masks sensitive text and images [59], and we confirmed that the developer has not enabled the unmask option, this implementation still raises privacy concerns.

```
onResume()>
|_<getValue()>
   |_<io.sentry.android.replay.ReplayIntegration$rootViewsSpy$2: io.sentry.
         android.replay.RootViewsSpy invoke()>
      |_<io.sentry.android.replay.RootViewsSpy$Companion: io.sentry.android
            .replay.RootViewsSpy install()>
         |_postAtFrontOfQueue(), getMainLooper()
```

Listing 4: How the app collects crash report via Sentry.

Though Google Play Store's data safety page indicates that the app may collect crash logs and diagnostic data, this broad categorization understates the extent and nature of the data collection. Session replay, even with masking enabled, can

reveal sensitive interaction patterns and user behaviors. For instance, the sequence and timing of user actions, screen navigation patterns, and general usage habits are still captured and transmitted to third-party servers. This granular behavioral data could be used to profile users or reconstruct their activities within the app, raising privacy concerns.

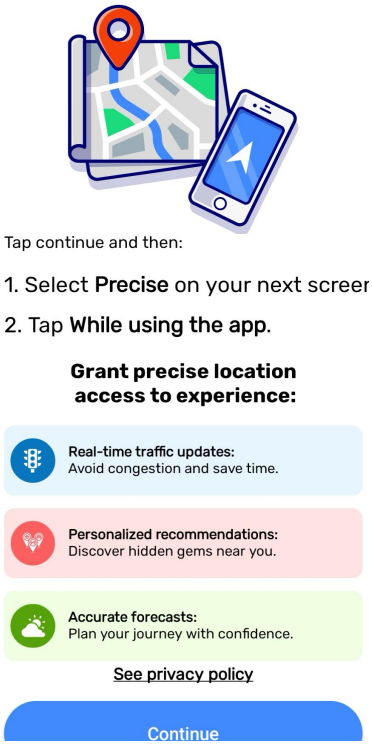

Fig. 17: The consent form of app `com.jvstudios.gpstracker`.

Furthermore, while Sentry's documentation explicitly states that developers should provide appropriate notices [60] to users about session replay functionality, we found no explicit mention of this feature in the app's privacy policy or user interface. This lack of transparency is particularly concerning because users in regions where this feature is enabled are unknowingly having their interactions recorded and analyzed at a level of detail beyond what would reasonably be expected for basic crash reporting. The disparity between regions where some users are subject to this detailed behavioral monitoring while others are not raises questions about equitable privacy protections across geographic boundaries.

This pattern reveals how geographic variations in implementation of features can lead to policy compliance issues, particularly regarding privacy-impacting features deployed without proper disclosures. Such practices raise two key concerns: potential violations of platform policies requiring transparent privacy disclosures, and compliance issues with regional privacy regulations that mandate explicit user notification and consent. The collection of detailed interaction data, even when masked, constitutes a form of user surveillance that requires both clear disclosure and user opt-out options.

Permission and User Consent. Although Android's runtime permission system provides a standardized mechanism for users to control app permissions, we observed significant

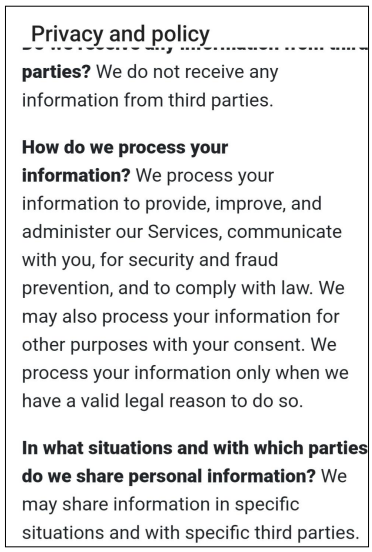

Fig. 18: The privacy policy that does not match with Play Store About page.

variations in how apps supplement this system with additional user notifications and consent mechanisms. Some regional versions of apps implement explicit informational interfaces, such as dedicated UI pages explaining permission requirements or custom widgets displaying detailed consent information, while their counterparts in other regions rely solely on Android's default permission dialogs. This raises concerns as study [61] has shown that apps may send personal data towards data controllers without the user's explicit prior consent.

For instance, the app `com.jvstudios.gpstracker` [62] showed inconsistent disclosure practices across regions. In some regions, users receive a customized consent form and privacy policy link upon first use or language change. However, versions in other countries provide minimal transparency, showing only system-default permission dialogs without contextual explanations or in-app access to privacy policies.

Further analysis reveals discrepancies between the developer's Play Store claims and their privacy policy. While the *Play Store About* page states the app "does not collect or share user data", their privacy policy indicates they "may share information in specific situations and with specific third parties". Previous studies [63] [64] also have shown that this discrepancy is alarmingly common. This sharing aligns with our observation of multiple third-party ad libraries in the app. However, only users in certain regions receive proper notification of this data collection via explicit consent forms. This regional disparity in privacy transparency suggests selective compliance—meeting stricter privacy regulations in certain markets while maintaining minimal disclosure in others——raising ethical concerns about informed consent. Figures 17–18 show the consent form and privacy policy.

Regional variations in consent implementation raise policy compliance concerns: they (1) foster information asymmetry, leaving users in some regions with inadequate context for privacy decisions and (2) reflect a compliance-driven, not universally consistent, approach to privacy protections.

X. Discussion

A. Implication of Results

Our findings reveal several important implications for different stakeholders in the mobile app ecosystem:

For **app developers**, the prevalence of functionality variations across regions calls for more systematic approaches to managing geographic customizations. While regional adaptations are often necessary for business or regulatory reasons [29], [31], [32], [33], [34], [38], our findings show that these modifications frequently introduce unintended security disparities. Such disparities can arise when a new, region-specific feature inadvertently creates a new vulnerability, or when security patches are not consistently propagated across all regional codebases (e.g., due to the increased maintenance overhead of managing multiple versions). Developers should establish clear security baselines that remain consistent across all regional versions, only varying security implementations when explicitly required by local regulations. The high occurrence of advertising-related security differences particularly suggests that developers should carefully evaluate how regional monetization strategies impact their app's overall security posture and user experience.

For **platform providers**, the significant number of apps exhibiting security-relevant GFDs (82.6% of identified variations) indicates a need for better tools and guidelines from platform providers like Google. Current app store policies primarily focus on individual app versions rather than cross-regional consistency. Platform providers should consider implementing mechanisms to help developers track and evaluate security implications of geographic variations. For instance, the app submission process could include automated comparison of security-relevant features across regional versions.

For **security researchers**, the diversity of SPIs we uncovered suggests fertile ground for future research. Particularly noteworthy is how seemingly benign feature differences often carry subtle security implications. Our methodology of dealing obfuscated methods also provides insights for developing more sophisticated tools for cross-version security analysis.

For **end users**, the variations in security and privacy implementations across regions indicate that users should be more aware of potential differences in protection levels based on their location. While users might assume that globally distributed apps maintain consistent security standards, our findings suggest this isn't always true. This highlights the importance of users understanding security and privacy features available in their regional versions of apps.

B. Limitations

Technical Limitations. Our technique is primarily designed to handle renaming obfuscation, which is the most common type of code obfuscation in Android apps. However, more sophisticated obfuscation techniques, such as control flow obfuscation, string encryption, or dynamic code loading, may lead to false positives in our analysis. When apps employ these advanced obfuscation methods, FreeLens might incorrectly identify or characterize feature differences due to its inability to fully comprehend the obfuscated code patterns.

Modern commercial Android apps often exhibit high complexity in their architecture and dependencies, which can pose challenges for static analysis tools. In particular, we found that FlowDroid sometimes fails to process these complex apps. When call-graph construction fails, FreeLens cannot proceed with its analysis, limiting our coverage of the Android apps. Even when call-graph construction succeeds, Java/Android call graphs are known to be incomplete/unsound in practice [65], which can omit real call edges or introduce spurious ones. As a result, our call-graph-based reachability filtering may exclude code that is actually executable or include code that is infeasible at runtime. Additionally, many modern Android apps utilize dynamic feature modules and on-demand content loading to optimize app size and performance. Since FreeLens performs static analysis, it cannot capture differences in dynamically loaded features or content that may vary by region.

Regarding substantial native code components, while FreeLens effectively captures differences in calling relationships with native functions through our call graph and call path diffing, it cannot analyze implementation differences within the native code (C/C++ and other compiled languages) itself. This limitation would lead to missing GFDs that are reflected primarily in changes to native function implementations rather than their calling patterns. Additionally, native code involved in obfuscated control flow (i.e., when the native code is invoked via reflective calls) is subject to the same reachability related limitation as mentioned above.

Threats to Validity. Our analysis provides a snapshot of geographic differences at a specific point in time, rather than a comprehensive view of how these differences evolve. This limitation stems from two factors: (1) Google Play's API does not provide access to the metadata of historical app versions. Even though we can get the historical app versions, we cannot align what versions are the latest at the specific time due to the lack of upload date; (2) Developers' app update policies and schedules vary significantly across regions. Consequently, we cannot determine whether identified differences are persistent across versions or represent temporary variations. However, it is important to note that our study evaluates these differences from an end-user's perspective, for whom even a temporary variation in security or functionality can be a user-experienced discrepancy. This limits our ability to understand the long-term patterns and motivations behind geographic feature differences. Additionally, although Android framework API signatures are stable, apps may invoke framework APIs via reflection and dynamically reconstruct the target class/method names at runtime (e.g., via string packing/encryption). In such cases, FreeLens may not statically resolve the reflective targets, which can cause missed API-bounded call paths and thus an under-approximation of certain GFDs. Moreover, because we cannot inspect build-time obfuscation settings, we treat potential cross-variant obfuscation inconsistency as a threat to validity; however, our evaluation and the prevalence of coherent, interpretable call-path differences in practice suggest that identifier-renaming-style obfuscation is sufficiently consistent for most studied apps.

Our approach relies on matching apps across regions using package names, which means we cannot automatically detect regional variants of apps that

use different package names. For example, Amex app (`com.americanexpress.android.acctsvcs.us`) [66] and its other countries' variants, e.g., Amex United Kingdom (`com.americanexpress.android.acctsvcs.uk`) [67] are essentially regional versions of the same app but are treated as separate apps in our analysis. This means our study might miss important GFDs implemented through separate app variants. Additionally, our study focuses exclusively on free apps, so our findings on the prevalence and nature of GFDs may not be generalizable to paid apps, which could exhibit different patterns of geographic variation due to their possibly distinct monetization and distribution strategies.

XI. Related Work

Android App Measurement. Kumar et al. [2] conducted the first large-scale investigations into GFDs in mobile apps, focusing on app availability and metadata. Several studies examined the evolutionary patterns of benign apps revealing their relatively rare use of refactoring or obfuscation [68], the evolution of behavioral differences between benign and malicious apps [69]. A recent study [70] explored the security relevancy of app incompatibilities, revealing that malware suffers lesser run-time compatibility issues than benign apps.

Software Differencing. Program differencing has been a fundamental research area in software engineering. For traditional software systems, techniques such as [71], [72] analyze abstract syntax trees to identify fine-grained code changes.

In the Android ecosystem, several specialized differencing techniques have been proposed. Androguard [73] pioneered Android app differencing to identify code changes between app versions. VerLog [74] leverages LLM to generate natural-language description of functionality feature differences for adjacent versions of Android apps. However, it is targeted at developers who have access to codebases.

Static Analysis for Android Security and Privacy. FlowDroid [4], Droidsafe [5], and Amandroid [75] represent foundational works that enable static taint analysis and inter-component data-flow tracking in Android. These tools precisely model Android's complex lifecycle and inter-component communication, allowing detection of sensitive data leaks, permission misuse, and insecure inter-app communication.

In contrast, FreeLens operates at a higher abstraction level. Rather than analyzing single-version data flows, it compares cross-version and cross-region code behaviors to reveal how functionalities diverge geographically. Using API-bounded call-path profiling and LLM-based semantic interpretation, FreeLens bridges low-level code changes with high-level SPIs, extending prior static analyses into a comparative, semantics-aware framework.

XII. Conclusion

This work presents the first large-scale, *code-level* study of GFDs in Android apps and their SPIs, analyzing over 21,000 apps across ten countries. We designed the FreeLens framework which effectively identifies GFDs and SPIs automatically, and found that most of identified GFDs translate to SPIs, revealing significant regional functional feature variations and their strong impact on data protection, ethical disclosures, regulatory compliance, and security/privacy concerns.

XIII. Ethics and Disclosures

All apps downloaded were performed using legitimate Google accounts registered in each target country, following Google's Terms of Service without using any circumvention techniques. Only publicly available apps were analyzed. We avoided any automatic dynamic testing that might interfere with app operations or backend services.

Following responsible disclosure practices, we notified both the Google Play security team and affected app developers of several identified apps with SPIs, providing technical details and reproducible evidence. Our communications offered concrete recommendations to mitigate risks and ensure corrective action before public dissemination of our findings.

Acknowledgments

We thank the reviewers for constructive comments. J. Guo, Y. Nong, and H. Cai were supported by National Science Foundation (NSF) under grant CCF-2505223 and Office of Naval Research (ONR) under grant N000142512252. Z. Lin was supported by NSF under grant 2330264.

References

[1] "Encrypted Messaging Applications and Political Messaging: How They Work and Why Understanding Them is Important for Combating Global Disinformation - Center for Media Engagement - Center for Media Engagement," https://mediaengagement.org/research/encrypted-messaging-applications-and-political-messaging/, Jun. 2023.
[2] R. Kumar, A. Virkud, R. S. Raman, A. Prakash, and R. Ensafi, "A large-scale investigation into geodifferences in mobile apps," in *USENIX Security*, 2022.
[3] S. Yang, G. Bai, R. Lin, J. Guo, and W. Diao, "Beyond the horizon: Exploring cross-market security discrepancies in parallel Android apps," in *ISSRE*, 2024, pp. 558–569.
[4] S. Arzt, S. Rasthofer, C. Fritz, E. Bodden, A. Bartel, J. Klein, Y. Le Traon, D. Octeau, and P. McDaniel, "Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for Android apps," *ACM sigplan notices*, vol. 49, no. 6, 2014.
[5] M. Gordon, D. Kim, J. Perkins, L. Gilhamy, N. Nguyen, and M. Rinard, "Information-flow analysis of Android applications in droidsafe," in *NDSS*, vol. 15, no. 201, 2015, p. 110.
[6] J. Zhang, C. Tian, and Z. Duan, "Fastdroid: efficient taint analysis for android applications," in *ICSE-Companion*, 2019, pp. 236–237.
[7] D. He, H. Li, L. Wang, H. Meng, H. Zheng, J. Liu, S. Hu, L. Li, and J. Xue, "Performance-boosting sparsification of the ifds algorithm with applications to taint analysis," in *ASE*, 2019, pp. 267–279.
[8] P. Wang, Q. Bao, L. Wang, S. Wang, Z. Chen, T. Wei, and D. Wu, "Software protection on the go: A large-scale empirical study on mobile app obfuscation," in *ICSE*, 2018, pp. 26–36.
[9] J. Guo, Y. Nong, Z. Lin, and H. Cai, "Code speaks louder: Exploring security and privacy relevant regional variations in mobile applications," in *IEEE Symposium on Security and Privacy (S&P)*, 2025.
[10] "Distribute app releases to specific countries - Play Console Help," https://support.google.com/googleplay/android-developer/answer/7550024?hl=en.
[11] Z. Dong, Y. Zhao, T. Liu, C. Wang, G. Xu, G. Xu, L. Zhang, and H. Wang, "Same app, different behaviors: Uncovering device-specific behaviors in Android apps," in *ASE*, 2024, pp. 2099–2109.
[12] A. Developers, *Permissions Overview*, 2023. [Online]. Available: https://developer.android.com/guide/topics/permissions/overview
[13] W. Enck, P. Gilbert, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth, "TaintDroid: An information-flow tracking system for real-time privacy monitoring on smartphones," in *OSDI*, 2010, pp. 393–407.
[14] J. Lin, J. Liu, N. Sadeh, and J. I. Hong, "Android permissions: User attention, comprehension, and behavior," in *SOUPS*, 2014, pp. 1–14.
[15] "APK Downloader for PC," https://raccoon.onyxbits.de/.
[16] skylot, "Skylot/jadx," Nov. 2024.
[17] J. L. Fleiss and J. Cohen, "The equivalence of weighted kappa and the intraclass correlation coefficient as measures of reliability," *Educational and psychological measurement*, vol. 33, no. 3, pp. 613–619, 1973.

[18] C. T. Ramos, "From the freedom of the press to the freedom of the internet: A new public sphere in the making?" in *Politics and technology in the post-truth era*. Emerald Publishing Limited, 2019.

[19] "Countries," https://freedomhouse.org/countries/freedom-net/scores.

[20] "How to change your Google Play country - Google Play Help," https://support.google.com/googleplay/answer/7431675?hl=en.

[21] K. Allix, T. F. Bissyandé, J. Klein, and Y. Le Traon, "Androzoo: Collecting millions of Android apps for the research community," in *MSR*, 2016, pp. 468–471.

[22] "List of most-downloaded Google Play applications," *Wikipedia*, Sep. 2024.

[23] J. Corbin and A. Strauss, *Basics of qualitative research: Techniques and procedures for developing grounded theory*. Sage publications, 2014.

[24] J. L. Campbell, C. Quincy, J. Osserman, and O. K. Pedersen, "Coding in-depth semistructured interviews: Problems of unitization and intercoder reliability and agreement," *Sociological methods & research*, vol. 42, no. 3, 2013.

[25] "Google play store statistics by device traffic," https://www.enter priseappstoday.com/stats/google-play-store-statistics.html.

[26] "Internet censorship in vietnam," https://en.wikipedia.org/wiki/Internet _censorship_ in_Vietnam.

[27] "Censorship in turkey," https://en.wikipedia.org/wiki/censorship_in_Turkey.

[28] A. Marcus, *Design, User Experience, and Usability: User Experience Design for Diverse Interaction Platforms and Environments: Third International Conference, DUXU 2014, Held as Part of HCI International 2014, Heraklion, Crete, Greece, June 22-27, 2014, Proceedings, Part II*. Springer, 2014, vol. 8518.

[29] GSMA Intelligence, "The mobile economy 2020," 2020. [Online]. Available: https://www.gsma.com/mobileeconomy/

[30] Google, "Android compatibility definition document (cdd)," 2023. [Online]. Available: https://source.android.com/compatibility/cdd

[31] A. Bhattacharya, "User-centric mobile app localization trends in 2024," 2024. [Online]. Available: https://www.linkedin.com/pulse/user-centric-mobile-app-localization-trends-2024-avishek-bhattacharya-tvq8c/

[32] D. Schiller, *Digital Capitalism: Networking the Global Market System*. The MIT Press, 1999.

[33] R. MacKinnon, "Consent of the networked: The worldwide struggle for internet freedom," *Politique étrangère*, vol. 50, no. 2, 2012.

[34] C. Kuner, L. A. Bygrave, C. Docksey, C. Docksey, L. Drechsler, and L. Tosoni, "The eu general data protection regulation: A commentary/update of selected articles," May 2021, available at SSRN: https://ssrn.com/abstract=3839645.

[35] E. Meyer, *The culture map: Breaking through the invisible boundaries of global business*. Public Affairs, 2014.

[36] "YouStar–Group Voice Chat Room - Apps on Google Play," https:// play.google.com/store/apps/details?id=in.dradhanus.liveher.

[37] "Soft Focus : Soft Photo Fluffy - Apps on Google Play," https://play.go ogle.com/store/apps/details?id=photofluffy.photo.android.app.addquick.

[38] M. McLennan *et al.*, "The global risks report 2022 17th edition." World Economic Forum Cologny, Switzerland, 2022.

[39] D. M. Mendez, I. Papapanagiotou, and B. Yang, "Internet of things: Survey on security and privacy," *arXiv preprint arXiv:1707.01879*, 2017.

[40] C. I. Cybersecurity, "Framework for improving critical infrastructure cybersecurity," *URL: https://nvlpubs. nist. gov/nistpubs/CSWP/NIST. CSWP*, vol. 4162018, 2018.

[41] G. Pulkkis, "Surveillance in the digital age: Methods, opportunities, and threats," in *International Handbook of Media Literacy Education*. Routledge, 2017.

[42] A. Marthews and C. E. Tucker, "Government surveillance and internet search behavior," *Available at SSRN 2412564*, 2017.

[43] J. Bonneau, C. Herley, P. C. Van Oorschot, and F. Stajano, "The quest to replace passwords: A framework for comparative evaluation of web authentication schemes," in *S&P*, 2012.

[44] D. Gollmann, "Computer security," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, no. 5, 2010.

[45] O. Reis, N. E. Eneh, B. Ehimuan, A. Anyanwu, T. Olorunsogo, and T. O. Abrahams, "Privacy law challenges in the digital age: a global review of legislation and enforcement," *International Journal of Applied Research in Social Sciences*, vol. 6, no. 1, 2024.

[46] "GPS Tracker: GPS Phone Locator - Apps on Google Play," https://play .google.com/store/apps/details?id=com.locator.gpstracker.phone.

[47] "Overview of the Play Integrity API | Google Play," https://developer. android.com/google/play/integrity/overview.

[48] L.-S. Huang, A. Moshchuk, H. J. Wang, S. Schecter, and C. Jackson, "Clickjacking: Attacks and defenses," in *USENIX Security*, 2012, pp. 413–428.

[49] J. Wang, Y. Xiao, X. Wang, Y. Nan, L. Xing, X. Liao, J. Dong, N. Serrano, H. Lu, X. Wang *et al.*, "Understanding malicious cross-library data harvesting on Android," in *USENIX Security*, 2021, pp. 4133–4150.

[50] B. J. Kwon, J. Mondal, J. Jang, L. Bilge, and T. Dumitraş, "The dropper effect: Insights into malware distribution with downloader graph analytics," in *CCS*, 2015, pp. 1118–1129.

[51] S. Ma, C. Chen, S. Yang, S. Hou, T. J.-J. Li, X. Xiao, T. Xie, and Y. Ye, "Careful about what app promotion ads recommend! detecting and explaining malware promotion via app promotion graph," *arXiv preprint arXiv:2410.07588*, 2024.

[52] F. P. Research, "AppLovin (APP) – Formers Allege Ad Fraud; Is DTC Hype Actually 'Stealing' Meta's Data; Illegal Tracking of Children & Serving Sex Ads to Kids," Feb. 2025.

[53] "Mobile Unwanted Software - Play Console Help," https://support. google.com/googleplay/android-developer/answer/9970222?sjid= 5354602318567233593-NA#ProtectUserDataAndPrivacy.

[54] K. Zhao, X. Zhan, L. Yu, S. Zhou, H. Zhou, X. Luo, H. Wang, and Y. Liu, "Demystifying privacy policy of third-party libraries in mobile apps," in *ICSE*, 2023, pp. 1583–1595.

[55] Y. Shen, P.-A. Vervier, and G. Stringhini, "Understanding world-wide private information collection on Android," *arXiv preprint arXiv:2102.12869*, 2021.

[56] M. H. Meng, C. Yan, Y. Hao, Q. Zhang, Z. Wang, K. Wang, S. G. Teo, G. Bai, and J. S. Dong, "A large-scale privacy assessment of Android third-party sdks," *arXiv preprint arXiv:2409.10411*, 2024.

[57] "Wildberries - Apps on Google Play," https://play.google.com/store /apps/details?id=com.wildberries.ru&hl=en.

[58] "Application Performance Monitoring & Error Tracking Software," https://sentry.io/welcome/.

[59] "Set Up Session Replay | Sentry for Android," https://docs.sentry.io /platforms/android/session-replay/.

[60] "Protecting User Privacy in Session Replay," https://docs.sentry.io /security-legal-pii/scrubbing/protecting-user-privacy/.

[61] T. T. Nguyen, M. Backes, N. Marnau, and B. Stock, "Share first, ask later (or never?) studying violations of {GDPR's} explicit consent in Android apps," in *USENIX Security*, 2021, pp. 3667–3684.

[62] "GPS+ Maps, Navigation, Traffic - Apps on Google Play," https:// play.google.com/store/apps/details?id=com.jvstudios.gpstracker &hl=en.

[63] I. Arkalakis, M. Diamantaris, S. Moustakas, S. Ioannidis, J. Polakis, and P. Ilia, "Abandon all hope ye who enter here: A dynamic, longitudinal investigation of Android's data safety section," in *USENIX Security*, 2024, pp. 5645–5662.

[64] L. Yu, X. Luo, X. Liu, and T. Zhang, "Can we trust the privacy policies of Android apps?" in *DSN*, 2016, pp. 538–549.

[65] M. Reif, F. Kübler, M. Eichberg, D. Helm, and M. Mezini, "Judge: Identifying, understanding, and evaluating sources of unsoundness in call graphs," in *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2019, pp. 251–261.

[66] "Amex - Apps on Google Play," https://play.google.com/store/apps /details?id=com.americanexpress.android.acctsvcs.us&hl=en_US.

[67] "Amex United Kingdom – Apps on Google Play," https://play. google.com/store/apps/details?id=com.americanexpress.android. acctsvcs.uk&hl=en_GB.

[68] H. Cai and B. Ryder, "A longitudinal study of application structure and behaviors in Android," *TSE*, vol. 47, no. 12, pp. 2934–2955, 2020.

[69] H. Cai, X. Fu, and A. Hamou-Lhadj, "A study of run-time behavioral evolution of benign versus malicious apps in Android," *Information and Software Technology (IST)*, vol. 122, p. 106291, 2020.

[70] J. Guo, X. Fu, L. Li, T. Zhang, M. Fazzini, and H. Cai, "Characterizing installation-and run-time compatibility issues in Android benign apps and malware," *TOSEM*, 2024.

[71] B. Fluri, M. Wursch, M. PInzger, and H. Gall, "Change distilling: Tree differencing for fine-grained source code change extraction," *TSE*, vol. 33, no. 11, 2007.

[72] J.-R. Falleri, F. Morandat, X. Blanc, M. Martinez, and M. Monperrus, "Fine-grained and accurate source code differencing," in *ASE*, 2014, pp. 313–324.

[73] "Androguard/androguard," androguard, Nov. 2024.

[74] J. Guo, H. Yang, and H. Cai, "VerLog: Enhancing release note generation for Android apps using large language models," in *ISSTA*, 2025.

[75] F. Wei, S. Roy, and X. Ou, "Amandroid: A precise and general inter-component data flow analysis framework for security vetting of Android apps," in *CCS*, 2014, pp. 1329–1341.