Understanding and Analyzing Privacy Risks in Mobile Consent-Management Platforms

Jingzhou Ye[†*], Fares Alharbi[‡*], Luyi Xing[§‡], Xueqiang Wang[†]
[†]*University of Central Florida, USA*
{jingzhou.ye, xueqiang.wang}@ucf.edu
[‡]*Indiana University Bloomington, USA*
fafaalha@iu.edu
[§]*University of Illinois Urbana-Champaign, USA*
lxing2@illinois.edu

***Abstract*—Today's mobile applications (apps) increasingly reuse third-party SDKs to provide essential functionalities. However, the integration of these SDKs into an app's supply chain introduces complexity, making it challenging to manage the SDKs and ensure their collective compliance with privacy regulations. Recently, consent management platforms (CMPs) have emerged, being increasingly adopted by mobile apps as a centralized mechanism and app-global configuration to help manage SDKs, particularly enabling all SDKs in an app to comply with the same user consent status for personal data processing. However, the question of whether the adoption of established CMPs ensures the validity of user consent specifically in mobile apps remains underexplored.**

This study addresses this gap in knowledge by conducting the first systematic investigation of the problems associated to obtaining user consent with CMP GUIs in real-world mobile apps (across Android and iOS). To achieve this, we developed a novel framework, DIULENS, that efficiently and comprehensively discovers CMP GUIs using a series of LLM-aided GUI analysis techniques tailored to CMPs. The framework analyzes both CMP GUIs and actual SDK usage within the apps to identify violations of privacy-accountable design, implementation, and use of CMPs (CMP DIU risks or DIU risks in short). Our findings reveal various DIU risks, such as failures to properly and consistently disclose third-party SDKs, ambiguous consent effects resulting from user interactions with CMP GUIs, and instances where consent is either difficult to withdraw or coerced. Attributing the causes, we found that both app developers' use (or configuration) of CMPs and flawed CMP implementations, or their combination, could cause DIU risks. We further observed differences in the adoption of CMPs and the DIU risks across the Android and iOS platforms, likely influenced by platform-specific privacy features, such as Apple's App Tracking Transparency (ATT) framework. This study provides new insights and bridges a critical knowledge gap regarding the assurance of CMPs in obtaining valid user consent in mobile apps.

[*]*The two authors contributed equally to this work.*

1. Introduction

As mobile apps increasingly permeate daily life, they also introduce significant privacy risks. For example, in 2024 alone, Google blocked 1.3 million apps for excessive personal data access [1], and in 2022, Apple flagged nearly 400,000 apps for collecting user data without users' knowledge [2]. These risks stem not only from the code written by app developers but also, to a large extent, from the integration of various third-party SDKs (or libraries) into the app's supply chain [3], [4]. With the complexity introduced by these SDKs, effectively managing them and ensuring consistent and collective compliance with major privacy regulations becomes a significant challenge. For instance, the GDPR [5] and COPPA [6] require that apps incorporating SDKs obtain valid user consent before any SDK collects or processes personal data, with the consent meeting key validity criteria, such as being informed, freely given, specific, and unambiguous. Yet, due to the lack of effective mechanisms for managing the consent status of SDKs, current SDK integrations in mobile apps have frequently been reported to violate these criteria [7], [8], [9], [10], thereby posing legal compliance risks. In response, one industry solution has been the development of consent management platforms (CMPs), which enable app developers to delegate the collection and management of user consent related to third-party SDK data practices within their apps. Standardization initiatives, such as the Transparency and Consent Framework (TCF) [11] developed by the IAB, have been introduced to guide and standardize CMP implementation.

Using established CMPs that follow industry standards is generally expected to enhance privacy compliance and accountability. However, research on CMPs used by websites [12], [13], [14], [15], [16], [17], [18] has shown that they can result in invalid user consent in various ways, such as through the use of pre-selected options in cookie banners. In recent years, mobile apps have increasingly adopted CMPs, with adoption rising from 6.6% in the top 100 Android apps [8] to 8.5% in this study, where each app

averaged 38.6 million downloads. Notably, top-100 apps, which tend to have larger user bases, are 41.7% more likely to adopt CMPs than those ranked between 100 and 200. This trend raises a critical question: *Does the adoption of CMPs, particularly well-established ones certified to meet industry standards, actually ensure privacy compliance by securing valid user consent in mobile apps?* Answering this question is not straightforward because mobile CMPs differ significantly from their web counterparts. For example, mobile CMPs often present more detailed privacy disclosures and request consent at a finer level of granularity for a large and diverse set of SDKs, and typically involve more complex graphical user interfaces (GUIs). In contrast, website CMPs generally focus on a small number of broad cookie categories. These differences limit the extent to which findings from web-based CMP studies can be applied to mobile apps. Currently, industry leaders often assume that using established CMPs ensures privacy compliance, as reflected in their official recommendations [19], [20]. However, empirical research that systematically evaluates this assumption in the context of mobile apps remains notably limited.

Analyzing CMPs GUIs of Mobile Apps. This study addresses the research gap by investigating potential issues in the use of CMPs in mobile apps across the two leading platforms, Android and iOS. To achieve this, we first developed a set of essential rules for the privacy-accountable design, implementation, and use of CMPs, dubbed the DIU rules, by contextualizing the key criteria for valid consent in CMPs. We then developed a new, efficient framework, called DIULENS, for identifying potential violations of these rules in CMP GUIs (refered to as DIU risks).

DIULENS, built upon the methodologies of GUI analysis, advances current research by introducing a series of targeted designs to address challenges in both CMP GUI discovery and DIU risk analysis. Specifically, unlike app-provided consent dialogs, which are often structurally simple and easy to explore, CMP GUIs tend to be more complex, requiring diverse user interactions for complete exploration and analysis. For example, CMPs that comply with TCF policies often adopt a layered GUI structure that displays varying levels of information (e.g., overall personal data usage, detailed lists of SDKs and purposes) across multiple interconnected GUIs and allows for consent choices at different granularities (e.g., accept all or itemized consent for specific SDKs or purposes). To tackle this challenge, we designed several new LLM-guided GUI analysis techniques tailored for CMPs, including a CMP-guideline-aware method for distinguishing CMP GUIs from other GUIs in apps, LLM-guided path exploration that prioritizes navigation flows leading to CMPs, and strategies for robust and comprehensive exploration of GUIs within CMPs. Another challenge arises from the fact that, although high-level guidelines for CMPs exist, the actual appearances of CMP GUIs vary across different CMPs and apps, making it infeasible to directly pinpoint DIU risks (as done in prior studies [8], [10]). To overcome the challenge, we adopted LLM reasoning rules that allow for reasoning about violations based on both CMP GUIs and the actual use of third-party SDKs in apps, thereby enabling effective and generic DIU risk detection while capturing variations in their manifestations.

DIU Risks in Real-world Mobile Apps. Our evaluation confirms the efficiency of DIULENS in discovering CMP GUIs and its effectiveness in analyzing them for DIU risks. To understand DIU risks in the real-world, we gathered two sets of popular mobile apps, including 9,332 Android apps and 3,496 iOS apps. Running DIULENS on these apps, we found that 397 apps (60.5% of those integrating CMPs, including 371 Android apps and 26 iOS apps) have at least one violation of the DIU rules. Analyzing these apps reveals new findings and evidence that demonstrate problems related to user consent persist even when CMPs are integrated, due to both privacy-nonaccountable CMP implementations and their use and configurations within apps.

For instance, 41.9% of mobile apps integrating CMPs fail to provide trustworthy disclosure information regarding third-party SDKs, such as requesting user consent for more or fewer SDKs than those actually used by the apps, or displaying inconsistent SDK disclosures across CMP GUIs, mostly due to app developers' failure to properly configure CMPs. Additionally, 14.9% of mobile apps either make it difficult to withdraw user consent by not allowing users to access CMP GUIs after providing consent, or force users to provide consent. Specifically, regarding forced consent, we found evidence that app developers manipulated CMP configurations to make non-essential SDKs (e.g., analytics) appear as strictly necessary for app functionality, thereby bypassing user consent. We also found that some of the most popular CMPs, such as OneTrust, fail to properly commit user consent choices, which contributes to forced user consent. Moreover, 8.1% of apps integrating CMPs collect user personal data before CMP GUIs are displayed to request user consent, and 11.7% contain GUI elements or navigation paths that lead to ambiguous effects on user consent, due to app developers' poor practices in configuring CMPs and the inherent limitations of CMP implementations. Finally, we observed that, compared to Android, a smaller proportion of iOS apps adopt CMPs, largely due to Apple's App Tracking Transparency (ATT) framework, highlighting the need for better integration of CMPs and ATT to ensure compliance with both platform-level (i.e., Apple) requirements and broader privacy regulations. Interestingly, we found that a smaller percentage of iOS apps were reported to have DIU risks, as their use of CMPs often involves simplified CMP GUIs, such as displaying SDKs according to ICC categories [21] without fully supporting GUIs compliant with TCF.

We are in the process of disclosing the DIU risks to affected stakeholders (app developers and CMPs), with some of them already confirmed or resolved. To support future research on CMP studies on mobile platforms, we have released the source code of DIULENS and the datasets of SDKs, CMPs, and mobile apps on Github [22].

Contributions. The contributions of this paper include:

• **New Techniques for Analyzing CMP GUIs in Mobile Apps:** We developed a new CMP GUI analysis framework, called DIULENS, with targeted designs to efficiently and effectively discover and analyze CMP GUIs.

• **New Understanding of DIU risks in Mobile Apps:** By running DIULENS on real-world apps, this study highlights various problems associated with obtaining user consent through CMP GUIs, which addresses the gap in knowledge about whether CMP adoption ensures privacy compliance in mobile apps, and further informs the privacy-accountable design, implementation, and use of CMPs.

2. Background

User Consent for Data Processing. User consent plays a central role in protecting digital privacy. For instance, Article 6(1)(a) of the GDPR [23] identifies consent as a lawful ground for processing personal data, while Section 312.5 of COPPA [6] requires parental consent for data collection from children and Section 1798.120 of CCPA [24] requires parental consent for the sale or sharing of such data. These privacy regulations require user consent to meet key criteria in order to be valid. For example, under the GDPR, valid user consent must be (1) freely given [25], with a genuine choice and no pressure or coercion, (2) informed [25], based on clear and accessible information, (3) specific [26], tied to a clearly defined purpose, (4) unambiguous [26], leaving no doubt about the individual's intent, (5) based on a clear affirmative action [26], such as clicking an "*accept*" button, (6) obtained before any personal data is processed [26], and (7) as easy to withdraw as it was given [27].

Consent Management Platforms (CMPs). Obtaining valid consent is a non-trivial task for software developers, as recent studies [10], [12], [17], [28], [29], [30], [31], [32], [33], [34] have shown frequent failures in meeting the required criteria for obtaining user consent. One of the most significant challenges is the increasing adoption of third-party SDKs, such as those used for advertising, analytics, and other purposes. To address the challenge and ensure compliance, many software developers rely on established consent management platforms (CMPs) to handle the obtaining, management, and documentation of user consent.

Figure 1 illustrates the workflow of CMPs in handling user consent for mobile apps. Mobile app developers start by integrating third-party SDKs, and a CMP into an app. The CMP presents GUIs to inform end users about the data practices of the SDKs and to collect their consent. Once consent is obtained, the CMP may store the corresponding consent signal in the device's storage. This signal can then be delivered to third-party SDKs, enabling them to operate in accordance with the user's consent choices. Third-party SDKs may also actively query the CMP to fetch the consent signal before proceeding with data collection. App developers are typically advised to initialize the CMP and display its GUIs as early as possible [35], ideally upon app launch, before any data processing by third-party SDKs begins. Besides that, to support consent withdrawal, CMP GUIs are also commonly integrated into other areas within an app

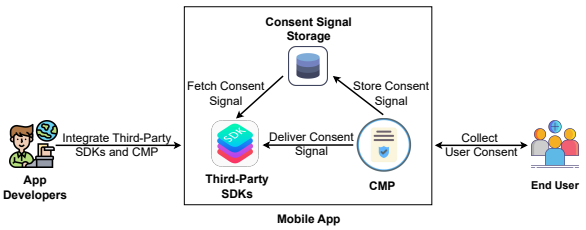


Figure 1: Workflow of CMPs in handling user consent

such as the app's privacy settings. The CMP workflow on other platforms (e.g., websites) follows a similar pattern, with CMPs essentially taking over the responsibility of managing user consent.

Transparency and Consent Framework (TCF). The Transparency and Consent Framework (TCF) [11] is the standard developed by IAB Europe for managing user consent to ensure GDPR compliance in digital advertising. TCF specifies consent strings that standardize the encoding and communication of user consent. It also provides centralized registry for participating ad tech vendors (or SDKs) through its Global Vendor List (GVL), which outlines the vendors' data processing activities and purposes. It further offers implementation guidelines for CMPs, including recommendations for GUI design. In particular, CMPs that are compliant with TCF should adopt a layered GUI approach to consent management, rather than relying on a single consent dialog. The initial GUI layer should present key information, such as the number of SDKs, a summary of their data practices, and clear consent options (e.g., "Accept All," "Deny," or "Manage options"). Subsequent layers should provide more detailed consent information, including a comprehensive list of SDKs, their specific data processing purposes, and options for users to manage consent on an itemized basis. Figure 2 presents the initial and second-layer GUIs of Usercentric [36], a leading CMP.

Today, nearly all major CMPs (e.g., 10 of the Top-10, as reported in [37]) have been certified by IAB and are verified to be compliant with TCF. To offer app developers more options and flexibility in consent management, some CMPs have also supported alternative frameworks. One example is the ICC framework [21], developed by the International Chamber of Commerce, which classifies website cookies into categories such as *strictly necessary*, *performance*, and *functionality*, thereby organizing consent options in a user-friendly manner. Although initially designed for website cookies, similar classification schemes have been adopted by some CMPs to manage SDKs on mobile platforms, i.e., classifying SDKs into those categories [38].

3. Rules for Privacy-Accountable Design, Implementation and Usage of CMP

CMPs facilitate the obtaining and management of user consent for personal data processing by the SDKs used by apps. For the consent to be valid and compliant with privacy

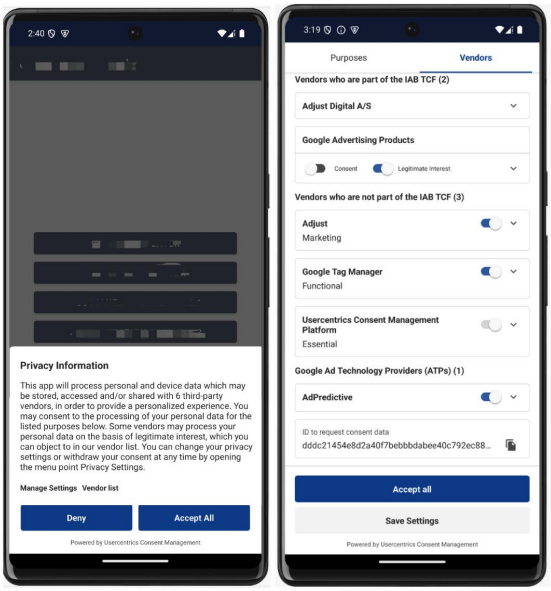


Figure 2: Usercentrics CMP: Initial and second-layer GUIs

regulations, the process of obtaining consent must meet several key criteria, as described in § 2. By contextualizing the criteria for CMPs, we propose four rules (outlined below) to guide, define, and regulate the privacy-accountable design, implementation, and use of CMPs in real-world apps. These are referred to as the CMP design, implementation, and usage rules, collectively termed the DIU Rules.

DIU Rule-1 (Timing of CMP GUI). CMP GUIs are supposed to let users specify whether they give consent (or denials) to individual SDKs for data collection or purposes of collection. Rule 1 is related to timing when CMP GUI should show up in the app. As required by major privacy regulations [5], [35], [39], user consents must be obtained before the collection of personal data. In the context of CMPs, the apps' CMP GUIs requesting consent should be displayed before any SDKs start to collect personal data.

DIU Rule-2 (Trustworthiness of Disclosure Information on CMP GUI). The CMP GUIs in an app must reliably disclose information about the individual SDKs used, enabling users to provide *informed* consent [40] for the SDKs' data collection and purposes. Rule 2 requires that the disclosure information for SDKs on CMP GUIs be accurate and correct. Based on our analysis of CMP guidelines [21], [35], the disclosed information typically includes the number of third-party SDKs, a list of the SDKs, their legal bases status, the purposes for which they process data, and the categories of data collected and processed. To ensure accuracy, the disclosed information must align with the app's actual SDK integration practices. For example, the number of SDKs disclosed on the CMP GUI must match the actual number integrated into the app.

DIU Rule-3 (Withdrawable and Uncoerced User Consent). Privacy regulations, such as GDPR Art 7(3) [27], stipulate that users shall have the right to withdraw their consent at any time. In the context of CMPs, the app should ensure that CMP GUIs remain accessible after the user has initially provided consent (or rejection), allowing the user to easily withdraw or modify their consent status with respect to the SDKs (*Withdrawable User Consent*). Moreover, user consents on CMP GUIs must be *freely given* by users [35]. For instance, users should be allowed to reject any SDK's data collections or specific purposes of the collections if they are not strictly necessary for the app's functionality (*Uncoerced User Consent*).

DIU Rule-4 (Unambiguous Consent Effect of User Interactions with CMP GUI). A key premise for consent to be informed is that users' navigation through CMP GUIs always results in clear, unambiguous consent effects. This requirement can apply to either each user interaction with CMP GUIs or the navigation path within CMPs. For instance, when users click a button or change the status of a checkbox, the resulting consent status should be clearly displayed. Also, users' navigation through multiple CMP GUIs should offer a consistent experience, free from any ambiguous user experience.

The above DIU rules are composed based on our understanding of the requirements for a valid consent (considering privacy regulations like GDPR). They are not meant to be complete, but meant to cover the minimum requirements that CMP GUIs should meet. Note that we focus on capturing rules that impact the validity of obtained user consent and manifest in end-user interactions with CMP GUIs, while placing no emphasis on other aspects, such as backend handling of consent signals, that do not directly interact with end users.

4. Design of DIULENS

To enable practical analysis of DIU risks in CMPs integrated into mobile apps, we developed a new automated framework, called DIULENS, which consists of four steps in its workflow. Figure 3 provides an overview of DIULENS. The first step is data collection, which gathers real-world mobile apps, CMPs, and third-party SDKs for analysis. The second step involves CMP and SDK usage analysis, where the actual integration of CMPs and SDKs, as well as their dynamic usage traces within the apps, are identified through static and dynamic app analysis techniques. The third step is LLM-aided CMP GUI analysis, which tailors the use of LLMs to efficiently discover CMP GUIs. The final step is DIU risk detection, where LLMs reason over CMP GUIs and SDK usage information to enable generic DIU risk detection. The output of DIULENS consists of identified DIU risks corresponding to violations of the DIU rules (§ 3), along with potential attributions for the causes of the violations. Below, we introduce each of the steps, highlighting the challenges to effective DIU risk analysis and our CMP-targeted designs to address these challenges.

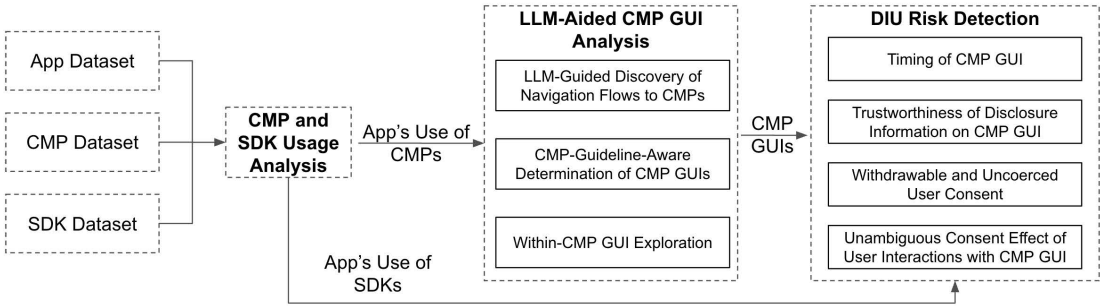


Figure 3: Overview of DIULENS

4.1. Data Collection

The data collection step builds two sets of apps, one for Android and one for iOS, along with a list of CMPs and the major SDKs they cover.

Android Apps. To collect Android apps, we use Google Play Scraper [41] to obtain the package names of the top free apps in each of the 54 Google Play categories, such as *Health & Fitness*, *Education*, *Finance*, and *Tools*. With the package names, we then download the app code in the form of APK or XAPK files using a custom crawler built for APKPure [42], an alternative app store to Google Play [43]. This alternative app store is considered a mirror of the official Google Play [43] and is commonly used by researchers [44], [45] to obtain app code, due to the lack of API access to Google Play. In total, we collected the package names of the top 200 apps in each Google Play category as of November 2024, resulting in 10,588 unique package names. We successfully downloaded the code for 9,332 unique apps; the remaining apps were not available on APKPure.

iOS Apps. We obtain the list of top free iOS apps and their app IDs from a popular App Store analytics platform, *Appfigures* [46]. For each app, we download its code (IPA) using IPATool [47]. We then install the apps on the iPhone, using the *libimobiledevice* toolkit [48], and decrypt them using *Frida iOS dump* [49]. We keep both the original and decrypted IPA files, using the former for dynamic analysis and the latter for static analysis. Also in late 2024, we successfully obtained 3,496 iOS apps.

CMPs. We collected the CMPs that (1) have passed the compliance check (or certification) of IAB Europe, and (2) have mobile SDKs available for app developer use. Specifically, we start by gathering the CMPs appearing on the CMP list of "*TCF v2.2 CMP Service (Mobile)*" [50]. Then, we verify the presence of mobile SDKs through manual exploration of CMPs' official websites. As of 2024, 53 CMPs were listed on the CMP list (Mobile), of which 21 were confirmed to have published mobile SDKs, with 21 supporting the Android platform and 19 supporting the iOS platform. For each of the 21 CMPs, we collected its website, package name (for Android), and framework name (for iOS, if the platform is supported).

SDKs. CMPs facilitate the collection of user consent for a wide range of third-party SDKs. To achieve comprehensive SDK coverage, we collected SDK names and their metadata (e.g., websites, package names, and service types) from several prominent SDK registry services: (1) IAB Global Vendor List [51], (2) Google's Ads Technology Providers (ATP) [52], and (3) Google Play SDK Index [53]. These sources cover a wide range of SDKs across various service types, such as advertising, analytics, and utility, that are commonly embedded in mobile apps. Note that the SDKs published by ATP and the Google Play SDK Index are not limited to the Android platform, as many SDKs, particularly the most popular ones, tend to cover multiple platforms. In total, we collected information for 1,858 unique SDKs.

4.2. CMP and SDK Usage Analysis

Detecting CMP and SDK Integrations in Apps. We conduct static app analysis to determine whether CMPs and SDKs are integrated into apps. Specifically, for an Android app, we decompile and transform the code using Soot [54], then analyze the packages included in the app. We check for the presence of CMPs and SDKs by detecting whether their package names appear in the app code. For an iOS app, we unpack its IPA file and traverse the `Payload` directory to identify frameworks (`.framework` files) or dynamic libraries (`.dylib` files), which correspond to the dynamically linked SDKs typically supported by leading package managers such as *CocoaPods*, *SPM*, and *Carthage*. We check for the presence of CMPs by verifying whether the frameworks used by apps contain the framework names of CMPs. In less common cases, iOS apps may integrate SDKs via static linking. To accommodate this, we use *otool* [55] on the iOS app binary (Mach-O file at `Payload/<app name>.app/<app name>`) to extract symbols and check for the presence of SDK names.

This step reports the list of CMPs and SDKs integrated into the apps by checking appearances of their code. Such information will (1) help DIULENS narrow down the scope of analysis to apps integrating CMPs, and (2) provide a reference for SDK disclosure information on CMP GUIs to detect DIU risks.

Analyzing Dynamic Usage Trace of SDKs and CMPs. For apps that include both CMP and SDK code, we conduct dynamic analysis to understand how these CMPs and SDKs are used at runtime, particularly in relation to the accessing of personal data and the management of user

consent. For this purpose, we perform both traffic and UI analysis. Specifically, we first gather a list of APIs that access personal data from several open-source privacy tools, such as Camille [56] and PrivacySentry [57] for Android, and Lalaine [58] for iOS. This list includes 73 Android APIs across 12 categories, such as location data access and device identifiers, and 140 iOS APIs across 5 categories, such as access to photos and contacts. When an app is executed through UI interaction (§ 4.3), we use Frida to hook into these APIs and log the stack trace for each invocation. We match the package names or framework names in the stack trace frames to determine whether the personal data access originates from SDK code. At the same time, we identify when CMP GUIs are displayed to users during app execution (as detailed in § 4.3). The output of this step is a sequence of time-stamped events, including personal data processing and the display of CMP GUIs.

4.3. LLM-Aided CMP GUI Analysis

In this section, we present our CMP GUI analysis techniques, featuring targeted designs to ensure both efficiency and comprehensive GUI discovery.

CMP-Guideline-Aware Determination of CMP GUIs. Typical methods for determining whether an app GUI is privacy-related involve analyzing privacy-related content, such as detecting privacy dialogs using keywords like "`privacy`," "`cookie`," and "`data protection`" [8], [10]. However, as evaluated in § 5.1, these methods cannot reliably identify CMP GUIs. While privacy-related keywords may also appear in CMP GUIs, they alone are insufficient, as many other GUIs such as privacy policy pages and non-CMP privacy dialogs implemented by app developers also contain these keywords. Our observation is that CMPs typically implement GUIs based on high-level guidelines (or requirements) specified by privacy-related frameworks, such as the TCF Policies [35]. However, the detailed layout and content of CMP GUIs can vary across different CMP providers, or even between app developers as CMPs are often configurable (as demonstrated in § 6). As a result, *determining CMP GUIs requires an approach that is aware to the CMP guidelines*, focusing on the nature of CMP GUIs described in the guidelines rather than specific UI content (such as privacy-related keywords).

Inspired by this, we design an LLM-based approach where an app UI is evaluated by LLMs to determine its relevance to CMP. Specifically, we use Appium [59], an open-source tool for automating mobile apps across both Android and iOS platforms, to drive the exploration of app UIs and extract UI information. For each app UI, we first extract all UI elements from the page source (an intermediate layout file) provided by Appium, including their attributes such as name, label, coordinates, size, etc. Next, we input these UI elements along with a prompt summarizing the nature of CMP GUIs into LLMs to assess whether the UI qualifies as a CMP GUI. Figure 9 (in Appendix) illustrates the prompt derived from the User Interface Requirements outlined in the TCF Policies [35] and the ICC guidelines [21]. Evaluation results (detailed in § 5.1) confirm that this approach achieves both high coverage (84.8%) and precision (100.0%) in determining CMP GUIs, compared to the keyword-based approach [8], which has a coverage of 55.9% and precision of 37.9%.

LLM-Guided Discovery of Navigation Flows to CMPs. The general recommendation for displaying CMP GUIs is to present them upon app launch to ensure privacy compliance [60], e.g., consent is obtained before any personal data is collected. However, as required by privacy regulations [27] and industry standards such as TCF policies [35], apps must resurface CMP GUIs to allow users to withdraw their consent. The navigation flows (or paths) to resurface the GUIs are not defined and often vary across different apps, depending on the specific app UI designs. Discovering these CMP GUIs within an app is non-trivial, yet crucial for a comprehensive analysis.

Most publicly available UI exercisers, however, are coverage-driven [61], [62], [63], [64], with their primary goal being to increase app testing coverage, rather than to specifically target the discovery of particular types of GUIs. To bridge this gap, we design an LLM-guided GUI exerciser aimed at the efficient discovery of navigation flows to CMPs. This exerciser consists of two key components: (1) prior knowledge of the navigation flows leading to known CMP GUIs, and (2) a prompt that guides the prioritization of GUI exploration. Specifically, we first built prior knowledge by reviewing 24 apps that integrate CMPs (selected from partners or customers listed on the websites of 5 CMPs and not covered by our mobile app dataset) and identified 11 navigation flows leading to CMP GUIs. Each flow is described with a list of text labels corresponding to the user interactions with the GUI (e.g., labels of buttons clicked during the navigation flow). Next, we crafted a prompt (as shown in Figure 10 in the Appendix) that takes as input the navigation flow leading to the current GUI, all the *clickable* elements on the GUI (extracted by Appium), and the prior knowledge. The prompt then generates an output that ranks the *clickable* elements based on the likelihood of advancing the navigation flow closer to CMP GUIs. Appium with interact with the current GUI based on the ordered elements. With this prompt, the exploration of navigation flows follows a largely depth-first search strategy, driven by CMP-specific knowledge. The exploration process terminates when a new CMP GUI is surfaced and restarts once the app navigates away from a CMP GUI to a non-CMP GUI. The exploration concludes when all app GUIs have been explored or when a time limit is reached (which is a common practice in mobile app analysis [65]). Our evaluation in § 5.1 demonstrates the significant advantage of this LLM-guided discovery process over state-of-the-art GUI exploration methods.

Within-CMP GUI Exploration. Unlike consent dialogs developed by app developers, which typically consist of simple, single-page GUI with limited interactions, CMPs have a more intricate GUI structure that usually involves multiple interconnected screens. Analyzing CMPs requires

thoroughly exploring all CMP GUIs and the interactions within them. Issuing random UI events [66] is a potential solution, as it ensures broad coverage of UIs. A significant challenge, however, hinders the effective application of this method when navigating within CMPs. CMP GUIs often contain elements, such as "*accept*" or "*reject*" options, which, when interacted with, can cause the exploration to end prematurely by navigating away from the current CMP. To address this problem and ensure both efficiency and coverage in CMP GUI exploration, we developed two strategies for managing the GUI interactions within CMPs.

The first is a *deferred interaction* strategy, where we delay interacting with CMP-exiting options until the latest possible moment. To implement this, we manually compiled a list of eight keywords (i.e., `accept`, `approve`, `allow`, `reject`, `deny`, `save`, `confirm`, `close`) associated with UI elements likely to cause the exit of CMPs, based on manual analysis of the documents of the top-10 CMPs [37]. For each GUI within a CMP, we identify GUI elements containing these keywords and reserve interaction with them until the very end of the exploration. The second is a *record-and-replay* strategy. Specifically, for each GUI, we generate a GUI fingerprint by hashing the combination of static GUI elements that have both non-empty name and label attributes in Appium. We then record each navigation path at runtime as a sequence of GUI fingerprints, where each fingerprint is associated with a series of in-GUI events, such as clicks on clickable elements, scroll actions, and the amount of vertical scrolling. If the CMP has not been fully explored before exiting, the Appium script will replay the GUI interactions based on the recorded navigation path and resume interaction with the remaining GUI elements. These combined strategies achieve high efficiency and coverage for CMP GUIs, with an overall CMP GUI coverage of 84.8%, and substantially more GUIs identified in the same amount of time compared to random clicking, which often results in abnormally exiting the CMP GUIs.

4.4. DIU Risk Detection

Detecting violations using high-level DIU rules (§ 3) presents practical challenges due to the wide diversity of CMP implementations and their inconsistent usage across mobile apps. For example, prior work such as Koch et al. [8] relies on a rule-based system using regular expressions and hardcoded heuristics to detect consent dialog violations. While effective for large-scale analyses of CMPs used on websites, their approach lacks flexibility and often fails to generalize across different CMP GUIs in mobile apps. Therefore, effectively detecting DIU risks requires a new approach that robustly evaluates CMP GUIs and SDK usage according to the rules, while ensuring generalizability to capture the varied manifestations of these violations. To address this need, we propose an LLM-based approach for DIU risk detection, leveraging the generalizability introduced by LLMs' ability to semantically reason about GUIs in relation to rules.

```
You are an expert in identifying problems in the GUIs
of consent management platforms (CMPs) within mobile
apps. You will be provided with the following
information:

  - CMP GUI Screenshots: A list of URLs for CMP GUI
screenshots, with timestamps indicating when the GUIs
were displayed. For example, [("https://a.com/1st_layer
.png", "2025-02-12T14:30:00"), ("https://a.com/2
nd_layer.png", "2025-02-12T14:31:00")] represents two
GUI screenshots.

  - CMP GUI Navigation Flows: A list of UI navigation
events, each including the source GUI, destination GUI,
 and the triggering action. For example, navigation
between two CMP layers: [("1st_layer.png", "2nd_layer.
png", "click 'Manage options' button")].

  - SDK Usage Data: Sentences listing the number of
SDKs, their names, and service types. For example: This
 app uses 50 third-party SDKs, including Adjust (
analytics), ..., and <ignored to reduce space>.

  - SDK Data-Access Events: A sequence of data-access
events initiated by the SDKs with timestamps indicating
 when the events occur, e.g., [("2025-02-12T14:29:30Z",
 "AppLovin", "Accessing Device Identifiers")].

Using the above information, check for violations of
the rules below: <DIU rules with exemplary examples.>

Note that:
  - You should evaluate the CMP GUIs, navigation flows,
 SDK usage data and data-access events against each
rule.

  - For each app, your response should be numbered
according to the rules, starting with "YES" (violation)
 or "NO" (no violation), followed by a one-sentence
description if "YES".

  - Do not report "YES" for low-confidence violations.
```

Figure 4: LLM prompt used in DIU risk detection

LLM Prompt Combining CMP and SDK Usage Data. At the core of the approach is an LLM prompt that describes the CMP GUIs and SDK usage information of an app, effectively making the mobile app code suitable for LLM-based analysis. As shown in Figure 4, the prompt begins by defining the role of LLMs in analyzing CMP GUIs, specifying the input format, listing the detection rules, and outlining the expected output format. We design the input to cover different perspectives, including individual CMP GUIs, across different GUIs, or between SDK usage and CMP GUIs. Specifically, the input includes a list of CMP GUI screenshots extracted in § 4.3. These screenshots are labeled with the timestamps of when they appear during CMP GUI analysis and are made accessible to LLMs via URLs pointing to the images stored in Dropbox [67]. We also provide LLMs with the navigation flows leading to each CMP GUI, in the form of a list of preceding GUIs and the corresponding GUI interactions that trigger the display of the next GUI. The prompt also includes SDK usage information, such as the number and list of SDKs used, their service types, along with a timestamped sequence of data-access events initiated by the SDKs. With both CMP GUIs, GUI navigation flows (identified through CMP GUI analysis, § 4.3), and SDK usage data (gathered through static

and dynamic app analysis, § 4.2) included in the input, the prompt will enable the detection of DIU risks through an extensible set of LLM reasoning rules (as we will illustrate later).

Reasoning Rules for Detecting DIU risks. We design LLM reasoning rules for detecting DIU risks, based upon the DIU rules outlined in § 3. Specifically, for DIU Rule-1, LLMs compare the timestamps of data-access events in SDK usage information with the timestamps of CMP GUI screenshots to determine whether any SDKs access user personal data before obtaining user consent. The reasoning rule for DIU Rule-2 compares SDK usage information and CMP GUI screenshots to identify instances of over- and under-disclosure of SDKs in CMP GUIs, relative to the SDKs actually used by the app. It also checks SDK disclosures across CMP GUIs to identify any inconsistencies in the SDK information. The reasoning rule for DIU Rule-3 focuses on checking whether apps allow users to withdraw their consent by resurfacing the CMP GUIs, specifically through an easily accessible path. Additionally, the rule will check whether CMP GUIs allow users to freely set their consent status for SDKs and data processing purposes that are not strictly necessary, and whether the current CMP GUIs enable users to explicitly confirm their changes (e.g., through a save button). Finally, the reasoning rule for DIU Rule-4 checks both individual CMP GUIs and their navigation paths to identify: (1) whether any user actions (such as button clicks) could result in a misleading consent status, and (2) whether any user interactions across GUIs along the navigation paths are visually or semantically misleading. The rule section in Figure 4 provides a summary of the reasoning rules.

Attribution of DIU risks. DIU risks may arise from various stakeholders, including CMP developers who implement CMPs, app developers who integrate CMPs into their applications, or sometimes both. For each DIU risk reported, we make best-effort attempts to identify potential sources through cross-app analysis. Intuitively, we can group apps based on the CMPs they use. For each group of apps using the same CMP, if all apps report the same DIU risk, the DIU risk can be attributed to CMP developers; otherwise, it would be attributed to app developers. However, in practice, app developers can decide which CMP GUIs to display [15], causing some DIU risks (even through they are caused by CMP developers) to appear in some apps while not appearing in others. To complement this, we further refine the grouping of apps based not only on the CMP name but also on the number of CMP GUIs displayed in the apps. The DIU risks attributed to app developers are primarily caused by misconfigurations during the integration of CMPs. To further investigate the potential causes of these misconfigurations, we review the integration process to identify any CMP-related factors that may make correct configurations difficult. § 6 reports the DIU risks identified in this study as well as their attribution to app and SDK developers.

5. Evaluation

Unless otherwise specified, all evaluation and measurement activities are conducted by running the Android and iOS apps from an IP located in Norway, where GDPR and its consent requirements are enforced. We run Android apps on a rooted Redmi Note 12 with Android 13, and iOS apps on a jailbroken iPhone X running iOS 16.7.10. We prototyped DIULENS based on the flagship model of ChatGPT, GPT-4o [68]. The evaluation aims to answer two key research questions: **RQ1**: What is the effectiveness of DIULENS in discovering CMP GUIs, and how does it compare to alternative methods? **RQ2**: What is the effectiveness of DIULENS in detecting DIU risks?

5.1. Effectiveness of Discovering CMP GUIs (RQ1)

Groundtruth Data. We randomly sampled 60 apps (30 Android and 30 iOS) from the pool of apps that integrate any CMP, as identified by static app analysis, and manually ran them to discover as many GUIs as possible. Through this process, we found that 25 of the 30 Android apps and 26 of the 30 iOS apps presented CMP GUIs during our analysis. More specifically, these apps presented a total of 165 distinct CMP GUIs, with each app having an average of 3.2 GUIs. Additionally, the Android apps presented 87 CMP GUIs (with an average of 3.5 per app), while iOS apps presented 78 CMP GUIs (with an average of 3.0 per app). We use these manually identified CMP GUIs as ground truth data for evaluating the effectiveness of DIULENS in discovering CMP GUIs.

Evaluation Results. We ran DIULENS on all 60 apps and found that it successfully discovered CMP GUIs in 47 of them. Specifically, DIULENS successfully discovered CMP GUIs in 24 out of 25 Android apps, covering a total of 76 CMP GUIs, and in 23 out of 26 iOS apps, covering 64 CMP GUIs. Notably, none of these CMP GUIs were false positives, i.e., all of them appeared in the manually discovered CMP GUIs. In terms of false negatives, we observed that one Android app does not allow taking screenshots of its app GUIs, while Appium fails to interact with three of the iOS apps, preventing GUI analysis. Overall, DIULENS successfully discovered 140 CMP GUIs across 47 apps, achieving an app-level coverage of 92.2% and a GUI-level coverage of 84.8%, with no false positives. With this high precision and reasonable coverage, we ensure that DIU risks reported later are indeed associated with CMP GUIs.

DIULENS vs. Keyword-Based Determination of CMP GUIs. Compared to prior approaches for determining CMP GUIs, the advantage of DIULENS is that it leverages a CMP-guideline-aware method, which accommodates variations in CMP implementations. To evaluate this, we compare DIULENS to a keyword-based approach similar to [8]. Specifically, for the 30 Android apps used to create the ground truth data, we exercise them using Appium and, for each GUI, search for the presence of the keywords used to identify privacy dialogs (including CMP GUIs) in [8]. This process reports 124 GUIs containing at least one of the keywords.

However, upon manual analysis, we found that only 47 of these GUIs are actual CMP GUIs, resulting in a coverage of only 54.0% of all CMP GUIs in the ground truth data. The remaining 77 GUIs identified by the keyword-based approach are not related to CMPs, such as those associated with privacy policies and app onboarding screens, leading to a precision of 37.9%. Compared to the precision (100.0%) and coverage (84.8%) of DIULENS, this stark difference highlights the need for more effective approaches to determining CMP GUIs, as we proposed in § 4.3. Note that we chose only Android apps for evaluation, as the determination of CMP GUIs is agnostic to mobile platforms once provided with the page source for GUIs by Appium.

DIULENS vs. FastBot for Discovering Navigation Flows to CMPs. Fastbot is a model-based GUI testing tool widely used for automated app GUI exploration [69]. We compared it to our LLM-guided approach for discovering CMP navigation flows. Specifically, we ran both FastBot and DIULENS for 15 minutes on each of the 30 Android apps, and observed the number of CMPs reached and the time spent to reach them. We found that Fastbot discovers CMP navigation flows in only 10.0% of cases within the time limit, with an overall CMP GUI coverage of 28.7%. This result confirms the infeasibility of using existing tools similar to Fastbot, which are not guided by the semantics of CMPs, for discovering CMP GUIs, compared to the 84.8% GUI coverage achieved by DIULENS. In particular, we found that Fastbot fails in two aspects. First, when CMPs are displayed in privacy settings with complex navigation flows, Fastbot often fails to discover them due to its strategy of maximizing overall GUI coverage, rather than being optimized for CMPs. Second, even when some CMPs are automatically presented during app launch, we found that Fastbot often quickly navigates away from them, failing to explore further CMP GUIs. This finding substantiates the challenge of covering CMP GUIs through robust within-CMP exploration, which we address by introducing two intuitive yet effective navigation strategies: deferred interaction and record-and-replay, as described in § 4.3.

5.2. Effectiveness of Detecting DIU risks (RQ2)

Groundtruth Data. We built the ground truth data for DIU risks based on 30 Android apps sampled in § 5.1. Specifically, we recruited two privacy researchers and provided them with the 87 CMP GUIs for these apps, along with the SDK usage data obtained from static analysis. We trained both researchers on the DIU rules and asked them to independently report whether the CMP GUIs in these apps violate any of these rules. The researchers reported 32 and 30 DIU risks in 18 apps, respectively, with both identifying potential violations for all four rules. Calculating Cohen's Kappa for the researchers' reports indicates that they achieved substantial agreement, with a Kappa value of 85.4%. Afterward, we asked them to review the potential DIU risks reported by only one of them and resolve any discrepancies through face-to-face discussions. This process resulted in ground truth data of 32 DIU risks across 18 apps.

Evaluation Results. We ran DIULENS on the above apps and found that it reported all the DIU risks, but with two false positives, resulting in a precision of 94.1% in detecting DIU risks. Both false positives are caused by the implicit meaning of CMP GUI elements. For example, an app, *CNN*, was reported to violate DIU rule-3 due to the lack of an option for users to explicitly save their consent choices. However, manual analysis confirmed that an "*OK*" button exists, which essentially serves the purpose of "*Save changes*" or "*Confirm*." We consider this only related to usability, rather than a true violation, since it does not force users to provide consent, and the "*OK*" button is neither directly related to consent status (which does not constitute a violation of DIU rule-4 either).

Generalizability Across LLMs. To further validate that our methodology for detecting DIU risks (§ 4.4) is general and not tied to a specific model, we replaced GPT-4o in the prototype implementation of DIULENS with two other LLMs, GPT-5 [70] and DeepSeek-V3 [71], and report their precision and recall in detecting DIU risks. When evaluated with GPT-5, DIULENS successfully identified all 32 ground-truth DIU risks while reducing false positives to one, achieving a precision of 97.0%. In contrast, with DeepSeek-V3, DIULENS also detected all DIU risks but introduced four false positives, resulting in a precision of 88.9%. We believe that the differences between DeepSeek-V3 and GPT-5 are primarily due to GPT-5's better multimodal semantic understanding compared to DeepSeek-V3 [72]. Nevertheless, despite these differences, the strong performance of all evaluated models demonstrates their practical potential for analyzing DIU risks, showing that DIULENS is not strictly tied to a specific LLM and can be adapted to different backend models.

6. Measuring DIU risks in the Wild

6.1. Landscape of DIU risks

TABLE 1: CMP SDK Detection Results

	Android	iOS	Total	CMPs
Total	9,332	3,496	12,828	15
Apps w/ CMP Code	675 (7.2%)	159 (4.5%)	834 (6.5%)	12
Apps w/ CMP GUIs	546 (5.9%)	110 (3.1%)	656 (5.1%)	12
Apps w/ DIU risks	371 (67.9%)	26 (23.6%)	397 (60.5%)	12
DIU Rule-1	48 (8.8%)	2 (1.8%)	50 (7.6%)	5
DIU Rule-2	256 (46.9%)	19 (17.3%)	275 (41.9%)	7
DIU Rule-3	90 (16.5%)	8 (7.3%)	98 (14.9%)	4
DIU Rule-4	73(13.4%)	4 (3.6%)	77 (11.7%)	6

Usage of CMPs by Apps. We ran DIULENS on 9,332 Android apps and 3,496 iOS apps to identify those using CMPs, at an average LLM query cost of approximately $0.01 per app. Through static app analysis (§ 4.2), we found that 675 (7.2%) Android apps and 159 (4.5%) iOS apps integrated at least one CMP. The integration rate of CMPs in iOS apps is lower than in Android apps, which we believe is linked to the introduction of Apple's App Tracking Transparency (ATT) framework [73]. Since iOS 14.5, apps on iOS have been required to request user consent through the ATT

framework specifically for tracking users via the device's IDFA. Integrating both the ATT framework and CMPs could result in an app requesting similar consent twice, potentially causing a confusing and frustrating user experience [74]. Additionally, the situation has also confused developers, as shown by the fact that even leading CMP platforms such as Google UMP have previously provided ambiguous guidance on whether CMPs are necessary when ATT is already implemented [75]. Given this, we suspect that some iOS app developers may have chosen not to integrate CMPs due to the presence of ATT. However, it is important to note that ATT does not negate the requirement for user consent to data processing (except for the use of personal data for tracking purposes) and is not intended to replace CMPs [74], [76]. Hence, more engineering and research attention could be needed to better integrate CMPs (for complying with privacy regulations) with ATT (for compliance on the iOS platform) to ensure genuine privacy compliance related to user consent.

Finding 1: iOS apps tend to integrate fewer CMPs than Android apps potentially due to the enforcement of Apple's ATT framework, which highlights the need for better integration of CMPs with ATT to ensure compliance with both platform-level requirements and privacy regulations.

These apps integrating CMPs are among the most popular, with an average of 28.65 million downloads on Android and over 165,000 reviews on iOS [1]. DIULENS further identified CMP GUIs in 546 Android apps and 110 iOS apps, and no CMP GUIs were identified in the other apps. We observed that CMP GUI analysis was unsuccessful for 37 Android apps for several reasons: nineteen apps could not be executed from our testing location in Norway, 10 were incompatible with the testing device, and the remaining eight either required users to download unavailable resources (e.g., game assets) or were blocked by a paywall that prevented the rendering of real app GUIs. Similarly, CMP GUI analysis was unsuccessful for 28 iOS apps due to challenges with installation on the iPhone (15 apps), limitations in Appium's ability to analyze GUIs and perform interactions (10 apps), and the removal of some apps from the App Store during our evaluation period (3 apps).

Overview of Detected DIU risks. Among these apps, 371 Android apps (67.9% of 546) and 26 iOS apps (23.6% of 110) were reported to violate at least one of the DIU rules. The most violated rule is DIU Rule-2 (Trustworthiness of Disclosure Information on CMP GUI), accounting for 46.9% of DIU risks on Android and 17.3% of DIU risks on iOS. The lower rate of DIU risks on iOS could be attributed to the adoption of more simplified CMP GUI designs. For example, on Android, 366 (67.0%) apps present CMP GUIs compliant with TCF, which include layered GUIs and a rich set of content as defined by TCF policies [35], such as a detailed list of SDKs, their purposes, and consent requests. However, only 31 (28.2%) iOS apps present CMP GUIs compliant with TCF, while a large portion (71.8%) display simplified GUIs that allow users to manage consent for several predefined categories of SDKs (similar to ICC cookie categories [21]), rather than itemized consent for each SDK, as shown in Figure 8 in the Appendix. Such CMP GUIs, which may violate the data transparency principle by omitting detailed SDK information (e.g., actual data recipients) [77], tend to trigger fewer violations in DIULENS due to the limited GUI content available for evaluation against the DIU rules.

Finding 2: Compared to Android apps, which often display CMP GUIs compliant with TCF, iOS apps tend to favor a simplified CMP GUI design that fails to provide detailed SDK information (and true data transparency), except for predefined SDK categories.

6.2. DIU Risks Violating DIU Rule-1

50 apps (7.6%) include third-party SDKs that begin collecting personal data before displaying their CMP GUIs, violating DIU Rule-1 (Timings of CMP GUI). This occurs because app developers often mistakenly initialize one or more SDKs before the CMPs are loaded. Also, although all apps were tested within the EU (where the GDPR applies), twelve apps were found to display CMP GUIs designed for the CCPA [24], which is intended to protect California residents. Specifically, instead of requesting consent before data collection begins (e.g., upon app launch), these apps assume data collection is allowed by default and only present CMP GUIs within privacy settings to offer users an opt-out option. Further analysis shows that these apps, while also serving EU residents, consistently present CCPA-related GUIs through hard-coded settings IDs, suggesting that developers may have misunderstood which privacy regulation applies.

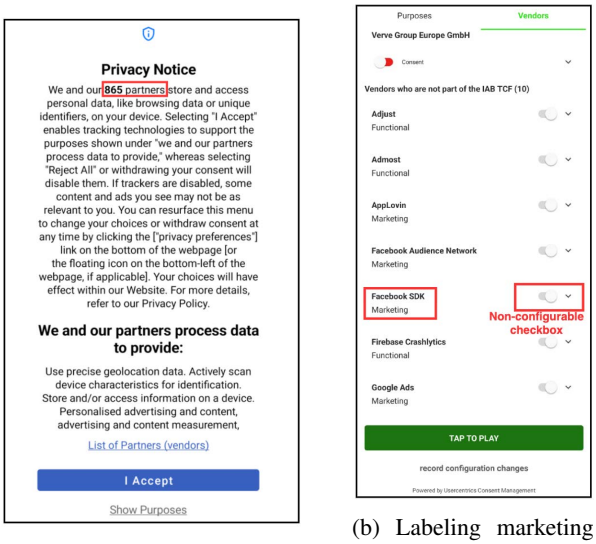
`Flip Master` is a real-world example that uses the Usercentrics CMP. The app launches directly into gameplay without displaying any CMP GUI upfront. Users can only access the CMP GUI by navigating to "*Settings – Privacy Settings*," where they find a "*Targeted Ads*" option turned on by default. This setup implies that the app assumes user consent for SDK data collection as long as users engage with the app. The CMP GUIs within the app are included solely to let users opt out, rather than to obtain their consent.

Finding 3: 7.6% of apps integrating CMPs access personal data before collecting user consent, due to app developers' failure to (1) properly coordinate the execution order of CMPs and SDKs, and (2) configure CMPs for compliance with applicable privacy regulations.

6.3. DIU Risks Violating DIU Rule-2

In total, 256 (46.9%) Android apps and 19 (17.3%) iOS apps were reported to contain DIU risks due to untrustworthy (inaccurate or incorrect) SDK disclosures. These DIU risks mainly manifest in two aspects: (1) SDK disclosures on CMP GUIs that are inconsistent with SDK usage, and (2) inconsistent SDK disclosures across different CMP GUIs.

1. The number of app downloads is not available for iOS



(a) Requesting consent for excessive SDKs in the 9GAG app

(b) Labeling marketing SDKs as "*essential*" in the Animal Hunter app

Figure 5: CMP GUI screenshots for (1) over-disclosure of SDK information, and (2) labeling marketing SDKs as essential.

SDK Disclosures Inconsistent with SDK Usage. 214 (32.6%) of all analyzed apps contain DIU risks due to the SDKs for which CMPs request user consent do not match the SDKs actually used by the app. The vast majority of these DIU risks appear as over-disclosure (95.4%), where the CMP GUI seeks consent for more SDKs than are integrated, while a minority are cases of under-disclosure (4.6%), where fewer SDKs are presented in the CMP GUI than are actually used. The former fails to provide specific and informed consent requests, causes decision fatigue by overwhelming users with unnecessarily long SDK lists, and exposes them to unnecessary privacy risks [78], while the latter leads to direct privacy violations [27], [39] due to the lack of consent for data processing by specific SDKs.

An example of over-disclosure is the 9GAG app, which has over 44 million installs. Despite using only 76 SDKs, its CMP GUI presented by OneTrust [79] requests consent for as many as 865 SDKs (as illustrated in Figure 5a). This issue stems from the fact that OneTrust, by default, includes all vendors from the IAB Vendor 2.2 framework [11], and requires app developers to actively review and configure the vendor list based on the SDKs actually used in their app. However, the developer of the 9GAG app adopted the default settings without tailoring them to the app's actual SDK usage. An example of under-disclosure is Clock Launcher, an alarm clock and launcher app that uses Iubenda as a CMP [80]. The GUI presented by Iubenda requests consent for an empty list of SDKs, while the app actually integrates third-party SDKs such as Google Mobile Ads SDK and Adjust SDK, both of which process personal data (as noted in the IAB vendor list [51]). Our analysis of the other affected apps reveals similar causes: app developers either accepted default CMP configurations without tailoring them to their apps (consistent with the *default effect bias* observed in behavioral studies, where users tend to stick to default settings [81], [82]), or omitted SDKs when customizing the CMPs (though it remains unclear whether these omissions were accidental or intentional).

Finding 4: Almost one-third of apps integrating CMPs disclose untrustworthy SDK information that does not align with actual SDK usage. The vast majority of these apps request user consent for an excessive number of SDKs, due to app developers failing to properly configure the list of SDKs when integrating CMPs.

Besides the 214 apps with misaligned SDK information, we also found that 165 apps do not disclose SDK information on their CMP GUIs. Specifically, these apps configure CMPs to only display SDK categories (similar to Figure 8 in Appendix), while hiding the detailed SDK list from users (as we noted earlier in § 6.1). This practice is not compliant with TCF policies [35], which mandate that CMP GUIs provide information about the "*list of named third parties*," as well as privacy regulations such as GDPR Article 13 [77], which requires transparency regarding the *identity of data controllers*. However, we exclude these 214 apps from our report as they do not explicitly disclose untrustworthy SDK information on the CMP GUIs, but instead represent a special case where no SDKs are disclosed at all.

Inconsistent SDK Disclosures Across CMP GUIs. A total of 61 apps (9.3%) were found to contain inconsistent information within or across CMP user interfaces. A prominent example involves 50 apps, all developed by Voodoo [83], which integrate SourcePoint as their CMP. In the initial CMP GUI, these apps inform users that they work with 36 partners (i.e., SDKs). However, upon clicking the "*View All Providers*" button, users are shown another GUI that includes both a text view and a list view. While the text view still states that 36 partners are used, the list view displays 70 SDKs. This DIU risk gives users the false impression that the apps rely on far fewer third-party SDKs than they actually do. We reported this issue to Voodoo, who confirmed that the discrepancy was due to "*a linking issue caused by a canceled integration between the text and the partner count module*," and have since resolved the problem. This issue affected all 57 apps in our dataset that integrate SourcePoint. Additionally, we found four apps using other CMPs, such as iubenda and OneTrust, that appeared to deliberately manipulate the number of disclosed SDKs, further contributing to user misperceptions.

Finding 5: Implementation flaws in some CMPs, along with deliberate manipulation of SDK disclosures by app developers, result in inconsistent SDK information across CMP GUIs.

6.4. DIU Risks Violating DIU Rule-3

A total of 90 (16.5%) Android apps and 8 (7.3%) iOS apps were reported to contain DIU risks violating DIU Rule-3 (Withdrawable and Uncoerced User Consent).

Unwithdrawable User Consent. As major privacy regulations stipulate [24], [27], [39], users should be able to withdraw their consent as easily as they give it. The TCF Implementation Guidelines [35] further require that app developers (i.e., publishers) provide accessible settings that allow users to withdraw consent. However, running DIU risks reveals that 15 (2.5%) apps do not include any GUIs that allow users to modify their consent once given. For instance, `C-SPAN Now`, a news app that uses the OneTrust CMP, presents consent options only during the initial app launch. Once the app is running, users cannot revisit or adjust their consent preferences. The option to withdraw consent is entirely absent from the app's settings, effectively forcing users to either accept their initial decision permanently or uninstall and reinstall the app to reset their preferences.

Finding 6: 2.5% of apps fail to provide accessible CMP GUIs for users to withdraw their consent.

Coerced User Consent. A total of 90 apps (13.7%) were found to force users into providing consent, either due to app developers' practices or flaws in CMP implementation.

The primary cause of forced user consent is the *failure (and potential manipulation) by app developers to classify SDKs*, i.e., to set up SDK categories in CMP configurations. Take the `Animal Hunter` app, which uses the Usercentrics CMP, as an example. In addition to SDKs from the IAB vendor list [51] and Google ad technology providers [52] (both of which are inherently integrated into Usercentrics), the CMP also allows app developers to manually add SDKs that are not part of these lists. Developers can specify the name, description, and, most importantly, the category of each SDK, such as *Essential*, *Marketing*, or *Functional*. Notably, SDKs labeled as *Essential* are treated as strictly required and are therefore exempt from consent requests, i.e., users are considered to have granted consent by default, with no option to modify this setting. We observed that the `Animal Hunter` app exploits this CMP feature, which is intended to support a wider range of SDKs, by labeling marketing SDKs such as AppLovin and Facebook SDK as *Essential* (as shown in Figure 5b), thereby forcing users to consent to their use without the option to opt out. Similar DIU risks were identified in 78 apps, with slightly varying appearances. For example, OneTrust, another CMP, classifies SDKs into five cookie categories: *Strictly Necessary*, *Performance*, *Functional*, *Targeting*, and *Social Media*. By default, only *Strictly Necessary* cookies (and their associated SDKs) are marked as "*Always Active*," preventing users from rejecting them. However, OneTrust allows app developers to configure any cookie category as "*Always Active*," effectively forcing users to consent to non-essential SDKs. Forty-five apps, such as `Forever 21`, were found to have exploited the configuration as described above. We could not pinpoint a definitive cause for such developer practices, whether they are technical failures or intentional manipulations. But a review of the research literature indicates that some developer practices may stem from

Figure 6: Ambiguous navigation flows, from an "*Always Active*" CMP setting to an SDK list allowing user choices.

a misunderstanding of the category labels. For example, as illustrated in [84], the *Functional* category, different to what the name suggests, is not necessarily required for the app to function properly; rather, it pertains to what could be considered "Extra Functional Cookies".

Finding 7: Some leading CMPs allow developers to specify which SDKs are essential, thereby assuming consent for them without offering users a choice. Evidence shows that app developers have indeed exploited this feature, whether intentionally or unintentionally, to bypass user consent for marketing SDKs.

Another cause of forced user consent lies in flaws in CMP implementation. Specifically, 56 apps (8.5%) were found to include a link to a "*List of Partners (Vendors)*" on the initial CMP GUI. However, when users navigate to the list and attempt to modify their consent preferences for certain partners (i.e., SDKs), there is no explicit option to save the changes (e.g., a "*Save*" or "*Confirm my Choices*" button), nor are the changes saved automatically. As a result, re-entering the partner list always shows that consent has been granted, regardless of any prior user choices. This DIU risk may mislead users into believing that their consent preferences have been recorded, when in fact they have not taken effect. Closer analysis of the apps reveals that they all use OneTrust as their CMP. Figure 7 (in Appendix) shows the initial GUI and the vendor list presented by OneTrust. We reported this issue to OneTrust, and they are currently investigating it.

Finding 8: Implementation flaws in some CMPs, such as OneTrust, prevent users from saving their consent choices on certain CMP GUIs.

6.5. DIU Risks Violating DIU Rule-4

A total of 30 apps (4.6%) were found to have CMP GUIs that support user interactions leading to ambiguous effects on user consent choices, potentially violating the requirement that user consent must be obtained through a clear, affirmative action [27], [39], [85]. A de facto standard implementation of CMPs is to provide built-in GUIs that allow users to make affirmative actions regarding third-party SDKs, such as "*Accept All*" or "*Manage Settings*." However, seven CMPs were found to allow app developers to freely customize the appearance of these actions, opening the door for abuse and consent manipulation [36], [86]. An example is the `Animal Hunter` app that uses the Usercentrics CMP. The app was found to replace the two standard options with "*Tap to Play*" and "*Details*," where "*Tap to Play*" suggests that clicking it will lead to the main game content. In reality, however, clicking this option immediately provides consent for the use of all SDKs, a fact that may not be immediately clear to all users. The customizations made by app developers vary, for example, replacing "*Accept All*" with "*OK*", "*Yes*", or "*Continue*". It is debatable whether such customizations actually lead to privacy violations, but according to prior studies on users' privacy perceptions [15], they could indeed cause confusion or mislead users when making consent choices.

Finding 9: App developers may customize the appearance of CMP GUI elements, such as buttons, at their discretion, potentially causing confusion or misleading users when making consent choices.

Ambiguity not only happen on individual CMP GUIs, but also manifest in the navigation paths between multiple CMP GUIs, which were found in 47 (7.2%) apps. An analysis reveals that all of these apps use OneTrust as their CMP. Notably, we observed that the apps list certain data processing purposes (e.g., "*Match and combine data from other data sources*") as "*Always Active*" in the CMP GUI, suggesting that users cannot reject these purposes. However, when users click on these purposes, they are redirected to another CMP GUI that presents a list of SDKs with toggle options to either consent to or reject their use for these purposes. This creates a discrepancy, as the initial CMP GUI leads users to believe that the purpose cannot be rejected, while the subsequent GUI allows them to reject it, resulting in confusion and an ambiguous experience for app users. Figure 6 shows the two CMP GUIs involved in this DIU risk. We have reported this issue to OneTrust, who is currently investigating the issue.

Finding 10: CMP implementations may create confusion and an ambiguous experience for app users when navigating across multiple CMP GUIs, which calls for more user-centered evaluation of these GUIs in addition to their technical implementation.

7. Discussion

Technical Contribution Discussion. Prior approaches to analyzing CMP GUIs, such as those based on regular expressions and heuristics [8], [10], [18], work well for other platforms, especially the Web, whose cookie banner layouts and contents have been standardized or uniform across websites. However, reusing these approaches to evaluate mobile CMP GUIs against DIU rules is difficult, as mobile CMP GUIs are often extensively customized by both CMPs and apps in layout, text, and navigation flows. This study makes two technical contributions to address this problem: (1) the introduction of a set of generic and foundational rules that define the expected (or compliant) design, implementation, and usage of mobile CMPs for obtaining user consents, i.e., the DIU rules (§ 3); and (2) the design of a new LLM-based semantic analysis pipeline, DIULENS, that enables generic assessment of these rules on customized and varied CMP GUIs across both Android and iOS.

Limitations and Future Work. While the DIU rules offer useful insights, they are not exhaustive, so our findings capture only a subset of current DIU risks. Other DIU risks, such as those involving the internal handling of consent signals by CMPs, go beyond what manifests in GUIs and will be explored in future work. The implementation of DIULENS involves several practical considerations. First, we observe that Android developers often apply obfuscation rules that retain CMP initialization classes [87], [88], whereas iOS developers typically leave framework file names unchanged. Hence, we detect CMP integration in apps by matching package names on Android and framework names on iOS (§ 4.2), noting that more advanced techniques, such as anti-obfuscation library detection, are not practically required. Second, testing each CMP individually (without apps hosting it) offers an alternative way to identify some DIU risks related to CMP implementations. However, we choose to analyze CMPs through the apps that integrate them, primarily due to practical constraints, which include many CMPs requiring licensing agreements, contracts, or explicit authorization to access their full functionality, as well as non-trivial configuration efforts, which hinder scalable and representative analysis in real-world scenarios.

Responsible Disclosure. We are in the process of responsibly disclosing our findings. Due to the large number of stakeholders involved (e.g., app developers and CMPs), our strategy is to prioritize reporting to app developers with the highest number of affected apps and CMPs most impacted by the DIU risks. So far, we have sent detailed DIU risk reports, including CMP identifiers, screenshots, and descriptions, to three app developers (covering 94 distinct apps) and one CMP provider covering 26 apps. In total, we have received two responses: one (e.g., Voodoo) confirmed the DIU risks and has already implemented fixes, while the other is still investigating.

8. Related Work

Prior research [12], [13], [14], [15], [16], [17], [18] has extensively examined the privacy risks associated with website cookie banners managed by CMPs. These studies highlight a range of compliance issues, including the storage of consent before users make any choice (i.e., consent by default), the absence of opt-out options, and the use of preselected choices to steer users toward agreement [12], [13], [15]. Researchers also found that websites often fail to honor users' preferences, continuing to collect and share personal data even after users opt out [12], [13], [14], [16]. To detect and analyze such violations, several tools have been developed. For example, studies [14], [17] propose client-side instrumentation to identify issues in consent string generation and transmission. CookieBlock [18] takes this further by applying machine learning to classify cookies and enforce user consent directly in the browser, revealing that over 94% of websites exhibit at least one type of GDPR violation. While effective in web environments, these techniques rely heavily on browser-specific features such as JavaScript instrumentation, HTTP cookie headers, and client-side consent strings – features often absent in CMPs integrated into mobile apps. Instead, mobile apps typically rely on local storage, proprietary SDK interfaces, or backend CMP APIs, making existing web-based CMP analysis techniques difficult to apply directly.

On mobile platforms, several studies have shown that consent dialogs implemented by app developers can present similar risks [7], [8], [9], [10], including a lack of meaningful opt-out options and non-compliance with user choices. However, these studies mainly focus on in-house implementations, rather than CMPs, which are becoming increasingly important due to their wider adoption. For instance, one study [8] conducted in 2023 found that that only 6.6% of top-100 apps adopted CMPs. In contrast, our study using apps collected at the end of 2024 (around one year later than [8]), reveals a notable increase in CMP adoption among top-ranked apps (7.2% on Android and 4.5% on iOS), likely driven by regulatory pressure and the widespread use of third-party SDKs that process personal data. This trend underscores the need for a systematic investigation into the design and use of CMPs in mobile apps. To address this gap, we introduce new tools for CMP GUI analysis and present novel empirical findings on their compliance behaviors.

9. Conclusion

This study sheds light on the problems associated with the use of consent management platforms (CMPs) in mobile apps. We begin by designing rules for the privacy-accountable design, implementation, and use of CMPs, by contextualizing the key criteria for valid user consent. Next, we develop a new CMP GUI analysis framework, DIULENS, to analyze violations of these rules in CMPs within mobile apps. Our findings show that, despite the adoption of CMPs, significant problems persist, including untrustworthy disclosures of SDKs in CMP GUIs, SDK data processing before consent is requested, non-withdrawable and forced user consent, and ambiguous consent effects caused by user interactions with CMP GUIs. These results highlight the need for app developers to improve CMP implementation and configuration to ensure compliance with privacy regulations.

Acknowledgments

The authors would like to thank the anonymous reviewers and shepherd for their insightful and constructive feedback. The UCF authors are supported in part by the UCF Seed Funding Program and NSF Grant No. 2520321. IU authors are supported in part by NSF grant CNS-2330265.

References

[1] A. Vakulov, "Google play's security: 2.36 million apps blocked for violations in 2024," *Forbes*, 2025. [Online]. Available: https://www.forbes.com/sites/alexvakulov/2025/02/02/google-plays-security-236m-apps-blocked-for-violations-in-2024/

[2] S. Gatlan, "Apple blocked 1.7 million apps for privacy, security issues in 2022," *BleepingComputer*, 2023. [Online]. Available: https://www.bleepingcomputer.com/news/apple/apple-blocked-17-million-apps-for-privacy-security-issues-in-2022/

[3] 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.

[4] I. Reyes, P. Wijesekera, J. Reardon, A. Elazari Bar On, A. Razaghpanah, N. Vallina-Rodriguez, and S. Egelman, ""won't somebody think of the children?" examining coppa compliance at scale," in *The 18th Privacy Enhancing Technologies Symposium (PETS 2018)*, 2018.

[5] European Union. (2016) General data protection regulation (gdpr). [Online]. Available: https://gdpr-info.eu/

[6] Federal Trade Commission, "Children's online privacy protection rule (coppa)," https://www.ftc.gov/legal-library/browse/rules/childrens-online-privacy-protection-rule-coppa, 2023.

[7] S. Li, Z. Yang, Y. Nan, S. Yu, Q. Zhu, and M. Yang, "Are we getting well-informed? an in-depth study of runtime privacy notice practice in mobile apps," in *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, 2024, pp. 1581–1595.

[8] S. Koch, B. Altpeter, and M. Johns, "The {OK} is not enough: A large scale study of consent dialogs in smartphone applications," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 5467–5484.

[9] 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 *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 3667–3684.

[10] T. T. Nguyen, M. Backes, and B. Stock, "Freely given consent? studying consent notice of third-party tracking and its violations of gdpr in android apps," in *CCS 2022*, 2022, pp. 2369–2383.

[11] IAB Europe, "Iab europe transparency & consent framework policies," 2024. [Online]. Available: https://iabeurope.eu/transparency-consent-framework/

[12] C. Matte, N. Bielova, and C. Santos, "Do cookie banners respect my choice?: Measuring legal compliance of banners from iab europe's transparency and consent framework," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 791–809.

[13] A. Bouhoula, K. Kubicek, A. Zac, C. Cotrini, and D. Basin, "Automated {Large-Scale} analysis of cookie notice compliance," in *33rd USENIX Security Symposium (USENIX Security 24)*, 2024, pp. 1723–1739.

[14] M. A. B. Aziz and C. Wilson, "Johnny still can't opt-out: Assessing the iab ccpa compliance framework," *Proceedings on Privacy Enhancing Technologies*, 2024.

[15] M. Nouwens, I. Liccardi, M. Veale, D. Karger, and L. Kagal, "Dark patterns after the gdpr: Scraping consent pop-ups and demonstrating their influence," in *Proceedings of the 2020 CHI conference on human factors in computing systems*, 2020, pp. 1–13.

[16] Z. Liu, U. Iqbal, and N. Saxena, "Opted out, yet tracked: Are regulations enough to protect your privacy?" *Proceedings on Privacy Enhancing Technologies*, 2024.

[17] M. Zhang, W. Meng, Y. Zhou, and K. Ren, "Cschecker: revisiting gdpr and ccpa compliance of cookie banners on the web," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, 2024, pp. 1–12.

[18] D. Bollinger, K. Kubicek, C. Cotrini, and D. Basin, "Automating cookie consent and {GDPR} violation detection," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 2893–2910.

[19] Google, "Set up your consent banner with a consent management platform or a content management system," 2024. [Online]. Available: https://support.google.com/analytics/answer/14546213?hl=en

[20] Google., "Google consent management requirements for serving ads in the eea, the uk, and switzerland for cmps," https://support.google.com/admanager/answer/13554020, 2024.

[21] I. C. of Commerce, "Icc uk cookie guide," 2025. [Online]. Available: https://www.mqmentalhealth.org/wp-content/uploads/icc_uk_cookiesguide_revnov.pdf

[22] SPIRIT-security, "DIUlens," https://github.com/SPIRIT-security/DIUlens, 2025.

[23] European Union, "Art. 6 lawfulness of processing," https://gdpr-info.eu/art-6-gdpr/, 2016.

[24] California Department of Justice, "California consumer privacy act (ccpa)," https://oag.ca.gov/privacy/ccpa, 2024.

[25] European Parliament and Council, "Recital 42 of the General Data Protection Regulation (GDPR)," https://gdpr.eu/recital-42-consent-requirements/, 2016.

[26] E. Parliament and Council, "Recital 32 of the General Data Protection Regulation (GDPR)," https://gdpr.eu/recital-32-consent/, 2016.

[27] European Union, "Art. 7 gdpr – conditions for consent," https://gdpr-info.eu/art-7-gdpr/, 2016.

[28] 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 *33rd USENIX Security Symposium (USENIX Security 24)*, 2024, pp. 5645–5662.

[29] N. Alomar and S. Egelman, "Developers say the darnedest things: Privacy compliance processes followed by developers of child-directed apps," *Proceedings on Privacy Enhancing Technologies*, 2022.

[30] C. Tagliaro, F. Hahn, R. Sepe, A. Aceti, and M. Lindorfer, "I still know what you watched last sunday: Privacy of the hbbtv protocol in the european smart tv landscape," in *30th Annual Network and Distributed System Security, NDSS 2023*, 2023.

[31] Y. Zhang, Z. Hu, X. Wang, Y. Hong, Y. Nan, X. Wang, J. Cheng, and L. Xing, "Navigating the privacy compliance maze: Understanding risks with {Privacy-Configurable} mobile {SDKs}," in *33rd USENIX Security Symposium (USENIX Security 24)*, 2024, pp. 6543–6560.

[32] Á. Feal, P. Calciati, N. Vallina-Rodriguez, C. Troncoso, and A. Gorla, "Angel or devil? a privacy study of mobile parental control apps," *Proceedings on Privacy Enhancing Technologies*, 2020.

[33] K. Kollnig, A. Shuba, R. Binns, M. Van Kleek, and N. Shadbolt, "Are iphones really better for privacy? comparative study of ios and android apps," *arXiv preprint arXiv:2109.13722*, 2021.

[34] Z. Liu, U. Iqbal, and N. Saxena, "Opted out, yet tracked: Are regulations enough to protect your privacy?" *arXiv preprint arXiv:2202.00885*, 2022.

[35] IAB Europe, "Iab europe transparency & consent framework policies," 2024. [Online]. Available: https://iabeurope.eu/iab-europe-transparency-consent-framework-policies/

[36] Usercentrics, "Usercentrics - automate privacy compliance. drive marketing growth," https://usercentrics.com/, 2025.

[37] Cookiebot, "10 of the best consent management platforms (cmps) compared," 2025. [Online]. Available: https://www.cookiebot.com/en/best-consent-management-platforms/

[38] OneTrust, "View and categorize scan results," https://developer.onetrust.com/onetrust/docs/view-and-categorize-scan-results, 2025.

[39] European Data Protection Supervisor, "eprivacy directive," https://www.edps.europa.eu/data-protection/our-work/subjects/eprivacy-directive_en, 2024.

[40] European Union. (2016) General data protection regulation (gdpr) – informed consent. [Online]. Available: https://gdpr-info.eu/issues/consent/

[41] F. Olano, "google-play-scraper," https://github.com/facundoolano/google-play-scraper, 2022.

[42] APKPure, https://apkpure.com, 2014.

[43] G. P. Store, https://play.google.com/store, 2012.

[44] P. Calciati, K. Kuznetsov, A. Gorla, and A. Zeller, "Automatically granted permissions in android apps: An empirical study on their prevalence and on the potential threats for privacy," in *Proceedings of the 17th International Conference on Mining Software Repositories*, 2020, pp. 114–124.

[45] A. Razagallah, R. Khoury, and J.-B. Poulet, "Twindroid: A dataset of android app system call traces and trace generation pipeline," in *Proceedings of the 19th International Conference on Mining Software Repositories*, 2022, pp. 591–595.

[46] Appfigures, "Appfigures – aso tools, app intelligence, and analytics," https://appfigures.com/, 2025.

[47] Majd, "ipatool: Command-line tool for downloading and managing ios apps," 2025. [Online]. Available: https://github.com/majd/ipatool

[48] libimobiledevice contributors, "libimobiledevice: A cross-platform library for interacting with ios devices," 2025. [Online]. Available: https://github.com/libimobiledevice/libimobiledevice

[49] AloneMonkey, "frida-ios-dump: A tool for dumping ios app executables," 2025. [Online]. Available: https://github.com/AloneMonkey/frida-ios-dump

[50] IAB Europe, "Consent management platform(cmp) list," https://iabeurope.eu/cmp-list/, 2025.

[51] I. Europe, "Tcf vendor list," https://iabeurope.eu/vendor-list-tcf/, 2025.

[52] Google, "Ad technology providers," https://support.google.com/admanager/answer/9012903?hl=en, 2025.

[53] Google, "Google play sdk index," https://play.google.com/sdks, 2025.

[54] S. Team, "Soot: A java optimization framework," 2025. [Online]. Available: https://soot-oss.github.io/soot/

[55] Otool, "otool - a tool for extracting objective-c runtime information from mach-o files," https://github.com/Imposter/otool, 2020.

[56] J. Zheng, "Camille: Detecting and characterizing location exposure in android apps," 2021. [Online]. Available: https://github.com/zhengjim/camille

[57] Y. Allen, "Privacysentry: Federated learning for detecting privacy violations in mobile apps," 2021. [Online]. Available: https://github.com/allenymt/PrivacySentry

[58] Y. Xiao, Z. Li, Y. Qin, X. Bai, J. Guan, X. Liao, and L. Xing, "Lalaine: Measuring and characterizing non-compliance of apple privacy labels at scale," *arXiv preprint arXiv:2206.06274*, 2022.

[59] A. Contributors, "Appium: A mobile automation framework," 2025. [Online]. Available: https://github.com/appium/appium

[60] Usercentrics, "Integrating usercentrics," https://docs.usercentrics.com/cmp_in_app_sdk/latest/integration/intro-collect/, 2024.

[61] Google, "Monkey: Randomized testing," https://developer.android.com/studio/test/other-testing-tools/monkey, 2025.

[62] A. Machiry, R. Tahiliani, and M. Naik, "Dynodroid: An input generation system for android apps," in *Proceedings of the 2013 9th joint meeting on foundations of software engineering*, 2013, pp. 224–234.

[63] T. Cai, Z. Zhang, and P. Yang, "Fastbot: A multi-agent model-based test generation system," in *Proceedings of the IEEE/ACM 1st International Conference on Automation of Software Test*, 2020, pp. 93–96.

[64] C. Peng, Z. Zhang, Z. Lv, and P. Yang, "Mubot: Learning to test large-scale commercial android apps like a human," in *Proceedings of the 38th International Conference on Software Maintenance and Evolution (ICSME 2022)*, 2022.

[65] H. Wang, Y. Fang, Y. Liu, Z. Jin, E. Delph, X. Du, Q. Liu, and L. Xing, "Hidden and lost control: on security design risks in iot user-facing matter controller," 2025.

[66] S. R. Choudhary, A. Gorla, and A. Orso, "Automated test input generation for android: Are we there yet?(e)," in *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2015, pp. 429–440.

[67] Dropbox User, "Dropbox," https://www.dropbox.com/, 2008.

[68] OpenAI, "Gpt-4o technical report," https://openai.com/index/gpt-4o, 2024.

[69] Z. Lv, C. Peng, Z. Zhang, T. Su, K. Liu, and P. Yang, "Fastbot2: Reusable automated model-based gui testing for android enhanced by reinforcement learning," in *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, 2022, pp. 1–5.

[70] OpenAI, "Gpt-5 technical report," https://platform.openai.com/docs/models/gpt-5, 2025.

[71] DeepSeek-AI, "Deepseek-v3 technical report," 2024. [Online]. Available: https://arxiv.org/abs/2412.19437

[72] A. Rahman, S. H. Mahir, M. T. A. Tashrif, A. A. Aishi, M. A. Karim, D. Kundu, T. Debnath, M. A. A. Moududi, and M. Eidmum, "Comparative analysis based on deepseek, chatgpt, and google gemini: Features, techniques, performance, future prospects," *arXiv preprint arXiv:2503.04783*, 2025.

[73] Apple Inc., "App tracking transparency," 2024. [Online]. Available: https://developer.apple.com/documentation/apptrackingtransparency

[74] Didomi, "App tracking transparency (ios 14.5+)," Aug. 2024. [Online]. Available: https://developers.didomi.io/cmp/mobile-sdk/ios/app-tracking-transparency-ios-14

[75] A. Vaish, "Implementation of google cmp along with att for ios apps," 2024. [Online]. Available: https://groups.google.com/g/google-admob-ads-sdk/c/OYoMmgwstNE

[76] Usercentrics GmbH, "Does att replace a cmp?" https://docs.usercentrics.com/cmp_in_app_sdk/latest/overview/att/#does-att-replace-a-cmp, 2025.

[77] European Union, "Art. 13 gdpr – information to be provided where personal data are collected from the data subject," https://gdpr-info.eu/art-13-gdpr/, 2016.

[78] B. W. Schermer, B. Custers, and S. Van der Hof, "The crisis of consent: How stronger legal protection may lead to weaker consent in data protection," *Ethics and information technology*, vol. 16, no. 2, pp. 171–182, 2014.

[79] OneTrust, "Onetrust – trust intelligence platform," https://www.onetrust.com/, 2024.

[80] iubenda, "iubenda – compliance software for websites and apps," https://www.iubenda.com/en/, 2024.

[81] C. Santos, M. Nouwens, M. Toth, N. Bielova, and V. Roca, "Consent management platforms under the gdpr: processors and/or controllers?" in *Annual Privacy Forum*. Springer, 2021, pp. 47–69.

[82] N. C. Council, "Deceived by design: How tech companies use dark patterns to discourage us from exercising our rights to privacy.(2018)," 2018.

[83] Voodoo, "Voodoo | entertain the world," https://voodoo.io/, 2013.

[84] S. Jiwani, R. Sasheendran, A. Abhyankar, E. Bouma-Sims, and L. Cranor, "Crumbling cookie categories: Deconstructing common cookie categories to create categories that people understand," *Proceedings on Privacy Enhancing Technologies*, 2024.

[85] European Union, "Art. 4 gdpr – definitions," https://gdpr-info.eu/art-4-gdpr/, 2016.

[86] UniConsent, "Uniconsent – universal consent management platform," https://www.uniconsent.com/, 2017.

[87] SourcePoint, "android-cmp-app: Android consent management platform sdk," 2025. [Online]. Available: https://github.com/SourcePointUSA/android-cmp-app

[88] "Android app implementation sdk," https://support.inmobi.com/choice/how-to-guide/integrate-the-tag-or-sdk/android-app-implementation-sdk#overview, InMobi Support, 2025.

[89] F. He, Y. Jia, J. Zhao, Y. Fang, J. Wang, M. Feng, P. Liu, and Y. Zhang, "Maginot line: Assessing a new cross-app threat to pii-as-factor authentication in chinese mobile apps," in *Network and Distributed System Security Symposium (NDSS)*, 2024.

[90] B. Tang, D. Bui, and K. G. Shin, "Navigating cookie consent violations across the globe," in *Proceedings of the 34th USENIX Conference on Security Symposium*, 2025, pp. 5817–5836.

Appendix A. Additional Figures and Prompts

Figure 7 presents the CMP GUIs of OneTrust, which allow users to modify their consent choices but do not provide an explicit option to save these modifications. As a result, users' consent choices are not properly recorded. Figure 8 presents an iOS CMP GUI that allows users to manage consent for several predefined SDK categories, similar to the ICC cookie categories [21]. Figure 9 shows the prompt used to determine whether a UI qualifies as a CMP GUI, guided by the User Interface Requirements in the TCF Policies [35] and the ICC guidelines [21]. Figure 10 shows the prompt used to identify clickable UI elements that are likely to advance the navigation flow toward CMP GUIs.

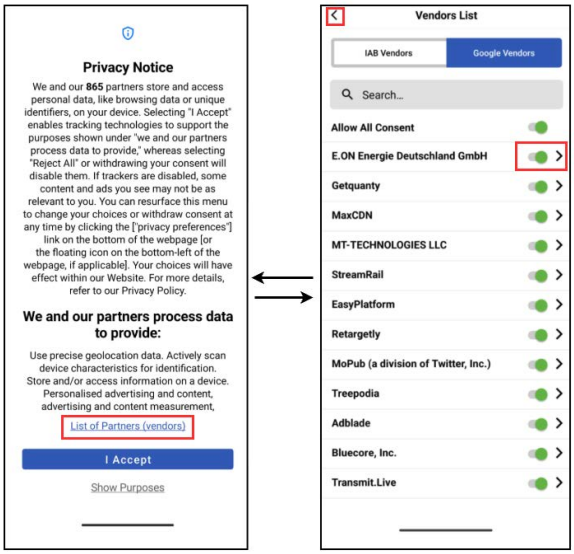


Figure 7: OneTrust CMP GUIs failing to save consent choices.

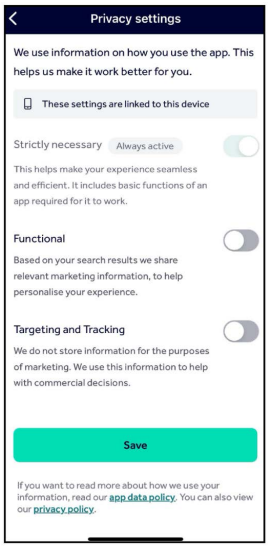


Figure 8: An iOS CMP GUI showing predefined SDK categories

```
You are a mobile GUI analysis expert who determines
whether a GUI is presented by consent management
platforms (CMPs) based on the page source of the GUI.
When making decisions, keep in mind that CMP GUIs often
 serve one of the following purposes:

(1) As an initial CMP GUI that provides information
about:
  - the fact that information is stored on and accessed
 from the user's device (e.g., cookies, device
identifiers, and other device data);
  - the fact that personal data is processed and the
nature of the personal data processed;
  - the fact that third-party vendors will be involved
in data processing, the number of vendors, and a link
to the full list of vendors;
  - the list of purposes for data processing;
  - information about the special features used by
vendors when processing data;
  - information about the scope of the consent choice;
  - the fact that the user can withdraw their consent
at any time and how to resurface the Framework UI to do
 so;
  - a call to action for the user to express their
consent (e.g., "Accept", "Okay", "Approve");
  - a call to action for the user to customize their
choices (e.g., "Advanced Settings", "Customise Choices
").

(2) As a secondary CMP GUI that allows users to:
  - review the following information: a list of named
vendors and links to their privacy policies, the
purposes and special purposes of data processing along
with their associated legal bases and retention periods
, the features and special features involved, and the
categories of data being collected and processed;
  - review detailed information about the purposes,
special purposes, features, and special features,
including their standard names, user-friendly text,
illustrations (where applicable), the number of vendors
 seeking consent for each purpose, and a way to see
those vendors;
  - make granular consent choices with respect to each
vendor and, separately, each purpose for which the
publisher seeks consent on behalf of one or more
vendors;
  - make granular opt-in choices for each special
feature for which the publisher seeks opt-ins on behalf
 of one or more vendors;
  - view information about vendors processing data
based on legitimate interest (if applicable) and the
user's right to object to such processing. It should
also provide details about the consequences of
consenting or not consenting, including how to withdraw
 consent (if not disclosed in the first layer);
  - review vendor-specific details, such as the maximum
 device storage duration and whether it refreshes, as
well as any additional purpose-specific storage and
access information provided by the vendor.

(3) As a CMP GUI that displays cookie/SDK categories
such as "strictly necessary," "performance," "
functionality," "targeting/advertising," and "social
media," and allows users to consent to or reject these
categories.

You will be given the page source of a GUI, and your
response should be limited to one of the following two
options:
  - "YES": if the page source corresponds to a CMP GUI.
  - "NO": if the page source corresponds to a non-CMP
GUI.
```

Figure 9: LLM prompt used to determine CMP GUIs

```
You are a mobile app GUI tester who explores navigation
 paths that are most likely to lead to consent
management platform (CMP) GUIs. These CMP GUIs are
typically found in the app's privacy settings, where
users can manage privacy preferences, cookies, or
consents. You will receive a list of GUI elements from
the current app screen, and your task is to order these
 elements based on how likely they are to lead to CMP
GUIs, from high likelihood to low.

Here is a list of GUI elements to select from:
<list of GUI elements from the current app screen>

Here is the sequence of GUI events leading to current
app screen:
<list of GUI events leading to current app screen>

Please use the following instructions when generating
response:
  - Only select GUI elements of the following types: {
Helper.get_clickable_elements_types()}.
  - Below are the navigation paths found to lead to CMP
 GUIs in some apps. Please use these as reference to
infer the likelihood of GUI elements leading to CMP
GUIs.
    - Cookie & Ad Preferences
    - Account -> Privacy Settings
    - Menu -> About -> Cookies Policy -> Manage your
consents
    - Menu -> Settings -> Cookie Settings
    - Menu -> Settings & Privacy -> Do not sell or
share my personal information
    - Menu -> Your Privacy Choices
    - Profile -> Account -> Data Protection -> Cookie
Settings
    - Profile -> Settings -> Data Protection
    - Profile -> Manage My Consents
    - Profile -> Cookie Settings
    - More -> Settings -> Privacy Settings
  - If the current app screen is a login/signup screen,
 please prioritize GUI elements such as "Continue as
guest", "Maybe later", "Skip", or similar.

Your response should follow the format of <list of GUI
elements from the current app screen> that is ordered
based on their likelihood to lead to CMP GUIs. Do not
add any explanatory text, or punctuation marks,
formatting, or backticks.
```

Figure 10: LLM prompt used to explore navigation flows to CMP GUIs

Appendix B. Meta-Review

The following meta-review was prepared by the program committee for the 2026 IEEE Symposium on Security and Privacy (S&P) as part of the review process as detailed in the call for papers.

B.1. Summary

This paper evaluates the privacy regulation compliance of consent management platforms (CMPs) in mobile Android and iOS apps, motivated mainly by GDPR. It introduces DIULENS, a tool that identifies violations through an LLM-guided approach, which takes as input context extracted from the static and dynamic analysis of the apps for SDK information, GUI elements via Appium, and screenshots.

B.2. Scientific Contributions

- Provides a New Data Set For Public Use
- Creates a New Tool to Enable Future Science
- Provides a Valuable Step Forward in an Established Field

B.3. Reasons for Acceptance

1) The paper introduces a LLM-guided approach to evaluating privacy regulation compliance of consent management platforms (CMPs) on two popular mobile platforms: Android and iOS. DIULENS is primarily automated and can be extended to other privacy regulations and platforms. It automates the triggering of CMPs, navigating through the CMP graphical interfaces, and detecting any noncompliance.
2) The paper conducts a large-scale evaluation of both mobile platforms (9,332 Android apps and 3,496 iOS apps that utilize CMPs) and discovers that 371 Android apps and 26 iOS apps had at least one violation.

B.4. Noteworthy Concerns

1) Small-scale manual evaluation: The paper relies on a small set of ground truth to evaluate their work: (1) 30 Android Apps in Sec. 5.2 (no iOS apps) for risk detection, (2) 60 apps for identifying CMP GUIs in Sec. 5.1 (30 Android, 30 iOS apps). Although the 60 apps cover about half of CMPs, this does not translate to thousands of apps on Android and iOS, especially since mobile apps are heterogeneous, providing new challenges as compared to the web ecosystem.
2) Evaluation of LLM-guided approach. The paper provides a high-level performance evaluation, such as the percentage of identifying apps with CMP GUI, risk detection, and some false positive explanations. However, it offers few insights into LLM-guided approach, including how every section of the prompt individually impacts the performance of DIULENS. This should help inform future work as CMPs get more prevalent in the mobile ecosystem and when DIULENS extends to future privacy regulations and platforms.

Appendix C. Response to the Meta-Review

We thank the anonymous reviewers and the shepherd for their valuable feedback and guidance on our paper. Below, we provide our responses to the "Noteworthy Concerns" raised in the meta-review.

1) Small-scale manual evaluation: The goal of this study is to investigate new privacy risks and previously unreported instances of these risks in CMPs used by mobile apps. As a result, we are inherently limited in gathering a large-scale ground truth for evaluation and therefore opted to use a small set of apps that we manually confirmed to exhibit these risks. We believe that the evaluation results on these apps are indicative of the effectiveness of DIULENS because (1) the apps were randomly sampled from a large set (that minimizes manual biases), (2) they cover more than half (11) of the 21 mobile CMPs registered with the IAB, as well as all seven mobile CMPs appearing in the top-10 CMP list [37], and (3) the sample size is comparable to prior studies [8], [89], [90] that also rely on manually created ground truth.
2) Evaluation of LLM-guided approach: We found it hard to soundly evaluate and understand the impacts of in-prompt components, partly due to the non-transparency of LLMs. Hence, the evaluation of DIULENS has primarily focused on an end-to-end assessment of its overall effectiveness in detecting DIU risks.