

Privacy Bills of Materials (PriBOM): A Transparent Privacy Information Inventory for Collaborative Privacy Notice Generation in Mobile App Development

Zhen Tao
Shidong Pan*
CSIRO's Data61 & Australian
National University
Australia

Zhenchang Xing
CSIRO's Data61 & Australian
National University
Australia

Xiaoyu Sun
Australian National University
Australia

Omar Haggag
John Grundy
Monash University
Australia

Jingjie Li†
University of Edinburgh
United Kingdom

Liming Zhu
CSIRO's Data61 & UNSW
Australia

Abstract

Privacy regulations mandate that developers must provide authentic and comprehensive privacy notices, e.g., privacy policies or labels, to inform users of their apps' privacy practices. However, due to a lack of knowledge of privacy requirements, developers often struggle to create accurate privacy notices, especially for sophisticated mobile apps with complex features and in crowded development teams. To address these challenges, we introduce PriBOM (Privacy Bills of Materials), a systematic software engineering approach that leverages different development team roles to better capture and coordinate mobile app privacy information. PriBOM facilitates transparency-centric privacy documentation and specific privacy notice creation, enabling traceability and trackability of privacy practices. We present a pre-fill of PriBOM based on static analysis and privacy notice analysis techniques. We explore the perceived usefulness of PriBOM through a human evaluation with 150 diverse participants. The role of PriBOM in enhancing privacy-related communication is well received with 83.33% agreement, suggesting that PriBOM could serve as a significant solution for providing privacy support in DevOps for mobile apps.

Keywords

Transparency, Usable Privacy, Mobile Applications, Privacy Policy, Privacy Paradox

1 Introduction

Due to functional, analytical, and advertising needs, mobile application developers are increasingly expanding their collection and use of users' personal information and other privacy-related

data. Many privacy regulations, such as the General Data Protection Regulation (GDPR) [2], the California Consumer Privacy Act (CCPA) [3], and the Australian Privacy Principles (APP) [1], require developers to provide *authentic* and *understandable* privacy notices to inform users of the app's privacy practices. Privacy policy is the most prevailing format of privacy notices to mobile application users [28, 39, 40, 49–51, 60, 86, 100]. In pursuit of higher readability and conciseness, mainstream application stores in the market have also recently required app developers to remind users of app's potential privacy data practices in the form of privacy nutrition labels (*a.k.a.* privacy label) [57–59], e.g. the Data Safety Sections (DSS) in Google Play and the Apple Privacy Labels (APL) in Apple's App Store.

However, numerous studies [32, 33, 70, 82] have shown that existing app privacy notices are often problematic, as they fail to authentically align with the actual data practices of apps. While non-comprehensive and inaccurate privacy notices could harm users' trustworthiness and violate privacy regulations, software developers face various challenges when providing authoritative privacy notices. Crafting a good privacy notice is complex, requiring not only legal knowledge but also a fundamental understanding of the app's various functions and features. Developers commonly lack training or knowledge in privacy and legal fields [66, 68, 70], and often hold a passive or even negative attitude towards privacy factors during development [64, 68, 70, 92]. Additionally, even though legal teams are mainly responsible for the privacy notice documentation, they are naturally not acquainted with mobile app technical details. Such numerous challenges have promoted the development of assistance tools in generating privacy notices.

There are three types of mainstream tools to help the development team generate privacy notices: Online Automated Privacy Policy Generators (APPGs) [4, 7, 9, 19, 20, 84], Code-based Privacy Policy Generators (CPPGs) [93, 115, 116, 120] and the recently emerged Code-based IDE Plugins (CIDEPs) [66, 67, 69]. Although these tools are useful and widely adopted, most are not applicable to sophisticated mobile apps with complex features and in crowded development teams. A small change in a basic function could require significant effort to accurately reflect the privacy practices in privacy notices. Concurrently, thousands of such modifications

*Co-first author with equal contribution.

†Co-senior author.

This work is licensed under the Creative Commons Attribution 4.0 International License. To view a copy of this license visit <https://creativecommons.org/licenses/by/4.0/> or send a letter to Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.

Proceedings on Privacy Enhancing Technologies 2025(4), 392–409

© 2025 Copyright held by the owner/author(s).

<https://doi.org/10.56553/popets-2025-0136>



will occur in the software development and maintenance process, making it impossible to manually track all those privacy practice changes, ultimately leading to problematic privacy notices. These issues are further magnified in the development under multi-role collaboration. The legal team often feel like “lone wolves” carrying the company’s privacy program alone [12], given that other roles are minimally involved in managing privacy. Thus, existing generators are far from enough, a systematic and collaborative software engineering solution involving various roles is pressingly needed.

To tackle this challenge, we propose PriBOM (Privacy Bills of Materials), a systematic approach that stores privacy practices in a structured manner and facilitates the transparent, collaborative, and accurate generation of privacy notices. The concept of PriBOM is inspired by the rising of Software Bills of Materials (SBOM). Figure 1 illustrates the use cases of PriBOM. In this paper, we first retrospect the development history of privacy notice and generation tools and then conduct a literature review as a formative study. The study reveals three major challenges to software privacy factors encountered by developers in DevOps: 1) Privacy Knowledge Absence, 2) Limited Technical Knowledge, and 3) Unfriendly Organizational Environment. We then scrutinize how previous privacy notice generation tools fail to mitigate those challenges and highlight the necessity for a revolutionary solution. After, we introduce the motivation and the design of PriBOM in detail.

Focusing on facilitating collaborative privacy notice generation in mobile app development, PriBOM is a table-like privacy information inventory indexed by UI widget, documenting information regarding 1) UI Identifier, 2) Codebase and Permission, 3) Third-Party Library and 4) Privacy Notice Disclosure. Table 2 presents the format of PriBOM. The UI widget serves as both visual elements and key components in functionality and data handling, therefore it is the pivot to synergistically connect different roles of developers on privacy-related communication. Additionally, we present a pre-fill of PriBOM based on cutting-edge static analysis and privacy notice analysis techniques, demonstrating its practicality. Furthermore, we conducted a usability evaluation of PriBOM through a survey of 150 participants. By using a survey, we also aim to prompt discussions and opinions around potential adaptation to specific applications beyond only validating its usefulness. The statements about design intuitiveness, content comprehension, and information relevance of PriBOM receive positive feedback, underlined by the 85.3%, 72%, and 78.76% agreement, respectively. We observed differences in the perspectives held by different roles, with experience within the same role further shaping these viewpoints. Non-technical roles frequently highlighted PriBOM’s efficiency in streamlining workflows. Lastly, we discussed potential enhancements and ongoing refinements to ensure its practical adaptation and its alignment with real-world demands across roles. The implementation and survey questionnaire are available at: <https://github.com/ZhenTAO3059/PriBOM>.

In summary, PriBOM serves as a collaborative solution for privacy notice generation. The pre-fill of PriBOM entails privacy information indexed by UIs as a template so that different roles in the development team can work towards consistent policy generation. The key contributions are:

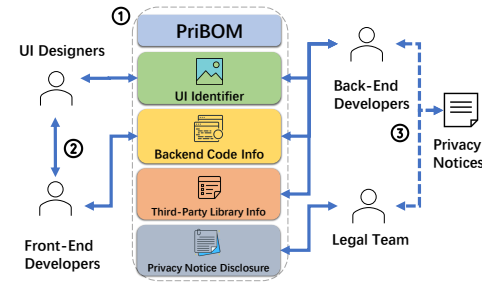


Figure 1: Use cases of PriBOM. (1) A privacy information inventory indexed by UI widgets, providing transparent privacy documentation. (2) A privacy communication platform between different roles in the development team. (3) A systematic solution for collaborative privacy notice generation.

- To the best of our knowledge, we are the first to systematically summarise the privacy notice generation tools.
- We introduce the concept of PriBOM (Privacy Bills of Materials) and propose a pre-fill for mobile app development.
- We conduct a human evaluation to comprehensively assess the usefulness of PriBOM.

2 Privacy Notice Generation for Mobile Apps

2.1 Status Quo

Although privacy policies have become the primary privacy notice approach for mobile applications [28, 39, 40, 49, 51, 60, 86], their presentation and readability have always been criticized [36, 76]. To improve the usability, Kelly et al. [57–59] introduced the privacy nutrition labels, or privacy labels, designed to facilitate consumers’ understanding of how their information is collected and utilized in a concise and structured manner. Privacy labels have been widely adopted by practitioners and have become a trend for conveying apps’ privacy practices to end users. As required by privacy regulations (e.g., GDPR), providing accurate privacy notices is equally important as providing them in an inviting way. Google¹ and Facebook². Failing to do so may cause serious legal consequences. To respond to those challenges, prior efforts have been made to assist developers create privacy notices. Based on their inherent nature, these tools can be categorized into various groups, including Online Automated Privacy Policy Generators (APPGs) [4, 7, 9, 19, 20, 84], Code-based Privacy Policy Generators (CPPGs) [115, 116, 120] and Code-based IDE Plugins (CIDEPs) [66, 67, 69]. We introduce the evolution process of these tools and summarize their key features below.

Most APPGs are questionnaire-based [84] online tools that depend on developer-provided information to generate privacy policies, as shown in Figure 2a and Figure 2b. Although APPGs can directly generate privacy policies, their qualities can be largely affected by developers’ unartistic design flaws and inaccurate completion. As the privacy policy generation process by APPGs is completely disconnected from the original software development

¹ *In re Google Assistant Privacy Litigation*, 457 F. Supp. 3d 797 - Dist. Court, ND California

² *In re Facebook, Inc. Internet Tracking Litigation*, 956 F. 3d 589 - Court of Appeals, 9th Circuit, 2020

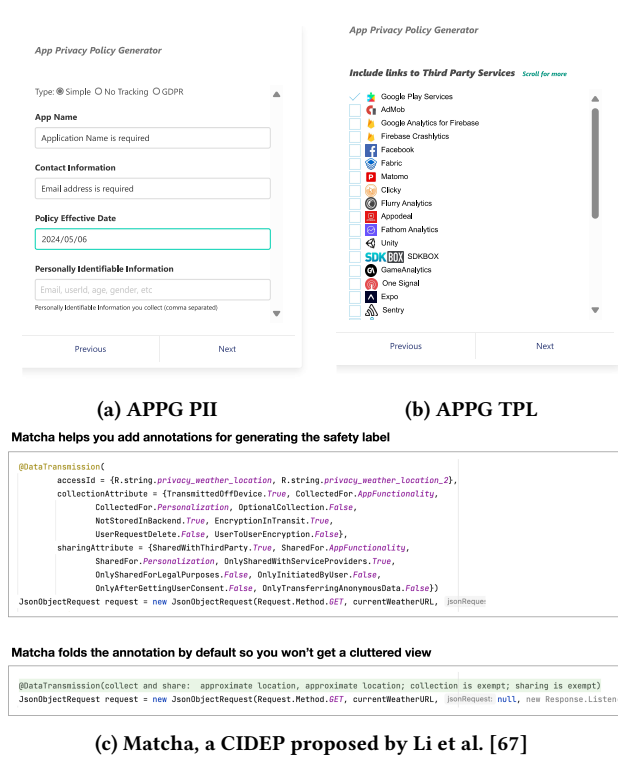


Figure 2: (a) and (b) are the interfaces of [7], one of the most popular APPGs, according to Pan et al. [84], on collecting Personally Identifiable Information (PII) and Third-party Libraries (TPL) usages. (c) is a screenshot of Matcha [67], one of the Code-based IDE Plugins (CIDEF), from its JetBrains plugin page [16].

in DevOps, it is inevitable that developers cannot appropriately maneuver these tools.

CPPGs have been proposed to further make the privacy notice generation stage “shift left” in DevOps, i.e., in the earlier stages of the software development cycle. Tools like AutoPPG [115, 116] and PrivacyFlash [120] analyze privacy-related features contained in the source code of Android and iOS applications and generate privacy descriptions or notices based on code features. However, CPPGs face significant challenges, such as inherent complexity and low explainability, which hinder their adoption and effectiveness. Also, CPPGs often fail to ensure compliance with high-level privacy regulations, particularly concerning non-functional requirements [84].

Unlike generating complete privacy notices post-development, CIDEFs are integrated into the integrated development environment (IDE) to provide code privacy annotations for developers during the development process. Figure 2c shows an example [67] of such CIDEFs. These tools, such as Coconut [66], Honeysuckle [69] and Matcha [67], help developers add privacy annotations to provide information for privacy notice creation, thereby reducing common misunderstandings among developers and easing the creation of privacy notices.

However, similar to APPGs and CPPGs, CIDEFs are also tailored for citizen developers or small teams in which only one or several developers are responsible for privacy notice generation.

Consequently, the usability and adaptability of these approaches are significantly constrained when applied to sophisticated mobile applications with complex features, typically developed by large teams. Therefore, we emphasize the need for a systematic solution integrated into DevOps that facilitates the collaborative generation of privacy notices. Understanding the specific challenges developers face is essential for developing an improved solution. Hence, we conducted a formative study to identify these challenges.

2.2 Formative Study

Developers often encounter privacy challenges as they attempt to build a thorough understanding of the privacy practices in the application and convert them into privacy notices [67]. We first summarize the previous studies to analyze developers’ challenges relevant to privacy aspects when they are developing applications or creating privacy notices.

With the increasing attention paid to privacy issues in software development, many research works are dedicated to exposing the privacy challenges developers face in the increasingly complex software development process. To form a comprehensive understanding of the current status of privacy challenges faced by developers, we conduct a systematic literature review to summarize these privacy challenges and the research efforts on discovering and studying them. Our target venues are cybersecurity *Big-Four*. We examined the titles of these papers from 2019 to 2024, and searched for keywords related to developer privacy challenges, such as “privacy”, “developer”, “development”, and “privacy challenge”, to conduct a preliminary selection of the papers. After obtaining the preliminarily selected papers, we manually checked the abstract and introduction of these papers to filter out papers that were not related to our topic. We then employed the snowballing method to comprehensively cover the relevant literature. First, we read through the related work section to identify related papers that are not published in target venues or do not include our keywords in the title. Second, we also examined papers that cited our selected papers. We eventually obtained 11 papers that discovered and studied privacy challenges faced by developers.

Table 1 presents a comprehensive overview of the multifaceted challenges developers encounter regarding privacy. To provide a structured approach to understanding these challenges, we categorize them into the following threefold:

[Challenge-1] Privacy Knowledge Absence. Such challenges pertain to the understanding of privacy within the development community. Developers often lack privacy knowledge [33, 55, 61, 64, 66, 68, 70, 102] due to the missing of formal privacy training, e.g. interpretation of privacy-related terminology. Misunderstanding Privacy terms [33, 61, 66, 68, 70, 97] can lead to discrepancies between intended and implemented privacy practices, potentially causing privacy issues. Moreover, developers may struggle with staying informed about the ever-growing privacy requirements from platforms and laws [70]. For example, developers may find it difficult to deal with the constant evolution of requirements from Google Play’s DSS policies [61], which affects the compliance of created privacy notices.

[Challenge-2] Limited Technical Knowledge. This kind of challenge involves privacy issues in the technical aspect, especially

Table 1: Summary of developer’s privacy challenges and comparison of assistant tools along those challenges. "APPG" stands for Questionnaire-based Online Automated Privacy Policy Generators, "CPPG" stands for Code-based Privacy Policy Generators, "CIDEP" stands for Code-based IDE Plugins. ●: addressed. ○: partially addressed. ○: not addressed.

Privacy Challenges in Software Development Process	APPG	CPPG	CIDEP	PriBOM
[Challenge-1] Privacy Knowledge Absence				
Misunderstanding of privacy terms [33, 61, 66, 68, 70, 97]	●	●	●	●
Lack of knowledge in privacy and legal field [33, 55, 61, 64, 66, 68, 70, 102]	○	○	○	●
Update and iteration of privacy rules [61, 68]	○	○	●	●
[Challenge-2] Limited Technical Knowledge				
Opacity of third-party libraries and resources [33, 61, 66, 68, 70, 91]	○	●	●	●
Complicated privacy notice creation process [61, 70]	●	●	○	●
Lack of awareness of privacy-preserving alternative implementations [33, 66, 68]	○	○	●	●
Limited tool support in understanding data practices [64, 66, 68, 96, 97]	○	●	●	●
[Challenge-3] Unfriendly Organizational Environment				
Not well-maintained privacy documentations [66]	○	○	●	●
Negative and demotivated attitude towards privacy [33, 55, 64, 66, 68, 70, 91, 96, 102]	○	○	○	●
Lack of team, organization and platform support [64, 66, 70, 91, 97, 102]	○	○	○	●

related to estimating a thorough understanding of the data practices. For example, developers often integrate Third-Party Libraries (TPLs) without a full understanding of their data handling practices [33, 61, 66, 68, 70, 91], leading to a lack of transparency regarding functionality and privacy of external sources. Such opacity may cause developers to struggle to follow data minimization, which is a principle introduced by privacy regulations such as GDPR [2] and formally defined by researchers [119]. Data minimization requires developers to only collect necessary personal data in relation to specific purposes. Moreover, developers may also not be aware of alternative approaches that offer better privacy preservation [33, 66, 68]. Although existing tools [64, 66, 68, 96, 97] can provide support, such necessity checks still rely on manual checks by legal experts [99, 119].

[Challenge-3] Unfriendly Organizational Environment. These challenges reflect the broader environmental and motivational factors that influence privacy. The challenge of developers’ negative and demotivated attitude towards privacy is mentioned by most studies [33, 55, 64, 66, 68, 70, 91, 96, 102] and highlight the necessity of an urgent change of the attitude. There are mainly three reasons that contribute to this phenomenon. First, as privacy is considered to be a non-functional factor, trade-offs exist between privacy and other objectives like functionalities, user experience, business goals and model performance [64]. Second, the ownership of Privacy is uncertain. Developers may regard privacy as the responsibility of other personnel and not consider it in their own workflow [64]. Last, developers may find them in a workplace culture that may not prioritize privacy [96]. The absence of a supportive infrastructure for privacy within organizations can demotivate developers from pursuing stringent privacy standards. According to [102], less than a quarter of developers have access to security experts. A disconnect between individual developers and decision-makers in the organization may also lead to negativity towards privacy [64].

Performance of Existing Generation Tools. We compare the existing generation tools along the dimensions of summarized privacy challenges faced by developers. As presented in Table 1, none of the existing tools can address all of these privacy challenges. First, these tools often assume citizen developers as their target audience,

resulting in poor contributions to the privacy environment in large companies and organizations. Second, although CPPGs and CIDEPs can provide technical support for developers, they cannot improve the lack of privacy knowledge among developers.

Given the inherent limitations of existing privacy notice generation tools, we argue that only a systematic software engineering solution can fundamentally tackle those problems. Drawing from empirical observations and the formative study, we propose our design of PriBOM in collaborative mobile app development scenario. PriBOM enhances transparency in privacy management and provides a platform for multi-role collaboration on privacy notice generation. We specifically introduce it in the next section.

3 The PriBOM Approach

3.1 Motivation of PriBOM

Bill of Materials (BOM). To cater to increasingly complex and diverse software systems, Bill of Materials (BOM) has been introduced as the information inventory with a focus on the transparency of software materials. Proposed in the 2010s [5], SBOM has become a key strategy in response to the emerging risks in software vulnerabilities management, license compliance, and supply chain transparency [81, 106]. SBOMs record the material details of the components in developing software, providing transparency, traceability and verifiability to software building and managing process [77, 94, 117]. The US Presidential Executive Order (EO) 14028 on Improving the Nation’s Cybersecurity requires companies providing services to the US government to provide SBOMs. Approximately 78% of the organizations worldwide would use SBOMs by 2022 and 88% by 2023 [52, 94]. The Cyber Resilience Act (CRA) [8] adopted in 2024 solidifies SBOMs as a mandatory requirement for products with digital elements (PDEs) placed on the EU market after 2027. In addition to SBOM, the concept of BOM has been implemented into other forms, including HBOMs [47] and FBOMs [29, 35, 45] for hardware and firmware, DataBOMs [34] for datasets, and AIBOMs [42, 105] for AI models. Consequently, we adapt a similar design for privacy information and introduce the concept of PriBOM. PriBOM records software privacy information

in a structured way and with a focus on UI, pulling different development roles to the same page. Our work share the same name with [110], which is a form of BOM focusing on TPL accountability from a supply chain perspective, whereas we provide a collaborative solution for privacy notice creation in DevOps cycle that links UIs to privacy disclosures. Specifically, PriBOM is indexed by the UI widget, including privacy information such as associated privacy permissions, data types, third-party libraries, and corresponding privacy notice disclosures. We will elaborate the design in Section 3.2. Privacy practices are scattered across files and documents, and no single approach organizes such privacy information. Development teams often struggle with it when drafting compliant privacy notices. PriBOM inherits properties of the BOM pattern, e.g. visibility, enumeration and traceability, and provides an ingredient list that can be worked on across role boundaries. Therefore, PriBOM is not a mere engineering work but an inventory concept purposed for privacy that tackles collaborative privacy notice generation problem with the virtues that made SBOMs successful. As the regulations evolve in future stage, more privacy transparency duties may emerge. We believe PriBOM could serve the similar role for privacy that SBOMs serves under the CRA, a standardized inventory that address issues of fragmented documentation and is as actionable for privacy compliance as SBOMs are for vulnerability management.

UI Widgets - The Pivot of Mobile Privacy Communication.

The UI widgets serve as both visual elements and key components in functionality and data handling, therefore it is the pivot to synergistically connect different roles on privacy communication. First, UI components are fundamental in the software development process, which are not only visual elements but are often directly involved in the app's functionality and data handling. The transparency regarding what data is collected through specific UI components is one of the information that users want to know [83], especially when malicious design may lead to unwanted data collection and cause harm to users [38, 43]. Moreover, UI represents the tangible interface where users interact with the application, making it a critical point that can raise privacy concerns [71, 85], and discrepancy between UI intentions and actual code behavior has been studied [65, 104, 108]. Existing efforts on privacy compliance checking often leverage UIs as the entry point to detect privacy leaks [44, 101]. We acknowledge that the UI-based approach does inherently focus less on non-UI events, which may lead to gaps in capturing practices that circumvent standard permission protocols [89]. This limitation can be mitigated by leveraging non-UI permission misuses detection techniques [114]. In addition, there are also non-UI-based compliance analysis frameworks that utilize non-UI component as the breaking point in program analysis, such as iOS API and data flow [109] and code functions [99]. Such approach is likely to cover more sensitive and privacy-invasive behaviors, while the UI-based approach is more inclined to capture expected requests [72, 80, 95, 98]. In this work, we choose to focus on a UI-based approach to demonstrate the feasibility of PriBOM. This is because the approach is more straightforward for more non-technical roles such as UI designers and legal experts in evaluating PriBOM. Technical efforts involved in the non-UI-based alternative, e.g., using code element for indexing, can be more laborious and non-intuitive for them, as the lack

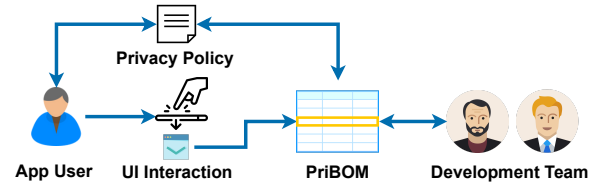


Figure 3: An overview of PriBOM in the usage scenario. The interaction between app users and UI components triggers data practices. These practices are disclosed to users through privacy notices such as the privacy policy. PriBOM helps development team create accurate privacy notices by documenting privacy information related to specific UI components.

of software knowledge leads to common obstacles in the communication [53]. Nevertheless, we argue that PriBOM can be adapted to incorporate non-UI-based permissions, such as taint tracking [80] and network monitoring [109].

3.2 Design of PriBOM

Figure 3 shows an overview of PriBOM usage, where it serves as a privacy information inventory indexed by UI widgets toward privacy notice generation. The proposed format is shown in Table 2 with a real example. Recognizing that development teams may have varying requirements and constraints, PriBOM should not be a rigid structure but a customizable approach with modifiability that teams can adjust according to their specific needs. Within the scope of this paper, the PriBOM design is tailored for mobile app scenarios, owing to the ubiquitous nature and privacy concerns of mobile applications. However, the underlying concept is sufficiently robust to extend to other software development contexts. The UI widget identifier section and codebase and permission section can be pre-filled utilizing static analysis techniques, which will be further discussed in Section 4.1. The TPL section and privacy notice disclosure section can be pre-filled using existing tools, e.g., TPL detector [103] and privacy policy segmentation framework [83, 111]. Though the current pre-filling does not include automated extraction of the latest version and publish date of TPLs, this can be easily obtained, e.g., by crawling the library's official website. One example of the pre-filled PriBOM is shown in Table 2. The elaboration of data fields in PriBOM is listed below:

UI Widget Identifier Section. As discussed in Section 3.1, UI widgets are the pivot to synergistically connect different roles of developers on privacy-related communication and collaboration. Therefore, we set the granularity of PriBOM at the UI widget level and include UI identifier in PriBOM to achieve sufficient identification.

- **[Widget ID]** *Widget ID* is the unique identifier for each widget and serves as fundamental information for referencing and mapping widgets to other data fields in PriBOM.
- **[Widget Type]** *Widget Type* records the functional type of a widget, e.g. "android.widget.ImageView" for image display, enabling developers to quickly form a preliminary understanding of the widget and achieve more fine-grained categorization and easier management.

Table 2: Design of PriBOM. One example of filled PriBOM of a real mobile application [11] is presented.

UI Widget Identifier		
Data Field	Description	Example
Widget Type	Component types of the widget.	android.view.Menuitem
Widget ID	A unique identifier for each UI widget component in the app.	2131296311
Widget Name	Names given manually for widgets to better recognize them.	action_share
Widget Src	Reference to source files, e.g. JPG or PNG images.	none
Codebase and Permission		
Data Field	Description	Example
Event	Specific events that the widget reacts to.	item_selected
Handler	The function or method that handles the event.	com.applovin.impl.mediation.debugger.ui.b.a: boolean onOptionsItemSelected(android.view.Menuitem)
Android API Level	The minimum Android API level required by the widget.	Level 1
Permission	The Android permissions required by the widget.	android.permission.ACCESS_COARSE_LOCATION
Data Type	The types of data the widget collects or processes.	Location
Method (Permissions) Path/Location	The path that accesses to the file requesting the permissions.	Landroid/location/LocationManager;-getLastKnownLocation-(Ljava/lang/String;)Landroid/location/Location;
Third-Party Library		
Data Field	Description	Example
TPL Name	The package name of third-party libraries involved in the widget.	javax.inject
TPL Version	The version of third-party libraries involved in the widget.	1
Latest TPL Version	The most recent version of third-party libraries available.	1.0.0.redhat-00012
TPL Publish Date (current version)	The release date of the current TPL version.	Oct 13, 2009
TPL Publish Date (latest version)	The release date of the latest TPL version.	Apr 16, 2024
Privacy Notice Disclosure		
Data Field	Description	Example
Privacy Policy Description	Corresponding sections in the privacy policy related to the widget's data practices.	"with your permission we may collect your geo-location information to optimize user experience, such as for localization accuracy..."
Privacy Label Declaration	Disclosure of privacy practices on related data type in privacy labels.	[Approximate Location] Optional: Yes; Purpose: App functionality, Analytics, Advertising or marketing

- **[Widget Name]** *Widget Name* is a readable common name given by developers to better recognize them during implementation and maintenance.
- **[Widget Scr]** *Widget Scr* is the reference to the widget's source files. For example, a functional button may be associated with a PNG icon image for appearance.

Codebase and Permission Section. This section focuses on the intricate relationship between the codebase specifics and the widgets. It covers data fields like the specific events that a widget reacts to, the methods of handling these events, the Android API levels, and the permissions required by each widget. Documenting these details aids developers in ensuring that the application adheres to best practices in privacy, facilitating an organized approach to managing privacy effectively. Our pre-fill of PriBOM utilizes static analysis to fill out this section, which is discussed in detail in Section 4.1.

- **[Event]** *Events* represent the triggers or user actions to which the widget responds. For example, "click" means the widget is clickable. This data field helps to understand the interactions leading to possible data processing.
- **[Handler]** *Handler* refers to the method that responds to the event, which identifies the specific code block handling the event's response. For example, the *handler* of event "click" can be "void onClick(android.view.View)".
- **[Android API Level]** This field records the minimum required Android API level the widget can operate on. *Android API Level* is considered as key information to ensure compatibility with various Android OS versions.

- **[Permission]** Android utilizes app permission mechanism [17] to protect access to both restricted data, e.g. Contacts, and restricted actions, e.g. taking pictures. More specifically, an app must require runtime permissions before it obtains access to additional data or performs actions that may affect the system, other apps or devices. Such requests may be involved in the widget's behind-the-scene behaviors [104]. For example, an elliptical coordinate icon in a food delivery app is likely to be accompanied by a request to grant geographic location permissions. Such requests are raised by the Android API calls in the widget callbacks.
- **[Method (Permissions) Location]** This field documents where in the codebase permissions are handled or required, e.g., the file path or location that accesses the permissions related to the widget.
- **[Data Type]** This field records the specific data type required by corresponding Android Permission, which is also regarded as the data that the widget collects or processes. Following the grouping strategy in previous researches [75, 88] and Official Android API Documentation [15], we map the Android Permissions to specific data types. For example, "READ_CONTACTS" and "WRITE_CONTACTS" are categorized into *data type* "Contacts". A more detailed discussion is in Section 4.1.

Third-Party Library Section. This section is dedicated to managing and documenting the use of TPLs associated with the widget, which play an important role in app functionality and data practices. It records the basic info of used TPLs. This section is crucial for maintaining up-to-date TPL usage and mitigating risks associated with outdated components.

- **[TPL Information]** Third-party libraries (TPL) pervasively serve as reusable functional components for mobile apps to improve development efficiency [103]. For example, the in-app payment

service of a shopping app may be supported by TPLs. PriBOM, therefore, records the baseline information of TPLs relevant to the widget, including *name*, *version*, *latest version*, *publish date (current version)*, and *publish date (latest version)*. PriBOM documents the publish date of the currently used and most up-to-date version of TPLs to enable assessing the currency of the version in use and informing about potential updates needed for security or functionality enhancements.

Privacy Notice Disclosure Section. In application development, a significant portion of effort is often allocated to ongoing updates rather than the outset. This principle also applies to privacy notices, given the evolving nature of legal and regulatory requirements. Consequently, PriBOM effectively manages two common forms of privacy notices, e.g. privacy policies and privacy labels, by linking widget components directly to their respective disclosures, ensuring continuous alignment between app data practices and legal requirements throughout the development lifecycle. We also realize its limitation in verifying the completeness of privacy notices, e.g., the UI approach cannot capture all sensitive behaviors. However, we believe the transparency and communication platform brought by PriBOM can help development teams create more compliant privacy notices in future, by mitigating unintentional under-disclosure and over-disclosure privacy practices [61].

- **[Privacy Policy Description]** This field refers to the disclosures in the app’s privacy policy related to the widget’s data practices. By recording relevant descriptions of the privacy behaviors, PriBOM links the technical implementation of the widget with the publicly stated privacy practices, offering a window for privacy compliance checking.
- **[Privacy Label Declaration]** Before publishing apps on markets such as Google Play, developers must declare their app’s privacy practices, e.g. data collection and handling, in a privacy nutrition label form. However, developers often struggle to create authoritative and accurate privacy labels [32, 33, 70]. Including privacy labels’ data practice declarations that match the data type in PriBOM can support developers and legal teams in aligning privacy disclosure with actual app behaviors.

Building on the UI widget inventory, PriBOM also provides possibilities to capture the semantics of sensitive user input. Specifically, following the existing research efforts [30, 54, 78, 101] on user input analysis relying on static analysis of layout resource files, PriBOM could be extended to analyze the embedded descriptive texts and labels of the UI widget to capture the semantics of user input and identify the corresponding privacy-related data type category. Recent work [44] has also explored dynamic UI analysis in detecting user personal data to overcome the coverage limitation of static analysis. By integrating these existing tools, PriBOM could analyze user input data to help align claims in privacy notices.

Violation of regulatory requirements that privacy notices must adhere, such as explicit consent before sharing users’ personal data with third parties, has been a widespread problem [79]. PriBOM has the potential to incorporate specific regulatory requirements to help avoid violation of privacy laws. For example, the rich information in PriBOM can be combined with existing GDPR compliance

tools [41, 107] to link the data right disclosures to the rights of the data subject in relevant GDPR articles, such as the right to access and right to rectification. Moreover, PriBOM could incorporate requirements from application stores, e.g. requirements from Google Play Developer Program policies [25] and guidelines of fulfilling Google Play’s Data safety section [18, 26, 27]. These integration possibilities will make PriBOM more comprehensive.

3.3 Benefits of PriBOM

Presenting the design of PriBOM reveals several unique benefits of integrating PriBOM into DevOps, responding to the challenges identified in the formative study:

[Benefit-1] Transparency. Lacking team-level and organizational support is considered a challenge faced by developers [66, 70]. Therefore, to address the privacy ambiguity caused by the lack of team support and assistant tools, the necessity of a recording and communication tool focused on privacy information has become more prominent. PriBOM can serve as a central communication tool, bridging the informational gap between developers of different roles from various teams. By providing a unified view of privacy practices at the widget level, PriBOM can help all developers have a consistent understanding of potential privacy implications, fostering informed decision-making and cohesive development strategies. Through PriBOM, transparency can be achieved as every practitioner, regardless of their role, gains clear insights into privacy aspects.

[Benefit-2] Privacy Traceability and Trackability. PriBOM connects the frontend UI widget, the backend privacy practices in source code, and the app’s privacy notices, helping developers gain traceability and trackability on this *privacy chain*. Similar to terminology in software supply chain [56], both terms in the context of PriBOM with ‘where-from’ and ‘where-used’ [87] directions respectively are as follows:

- **Tracing.** Given any point in this privacy chain, PriBOM enables developers to work backward to query privacy-relevant information. For example, if a user reports privacy issues in interacting with certain UI widgets, then the development team can easily track back and locate the problematic code files (Currently, a privacy breach takes an average of 287 days to completely resolve [24]).
- **Tracking.** PriBOM enables developers to work forwards to find corresponding UI widgets and privacy notice content based on certain privacy practices. For example, if a code change leads to a change in privacy practices, the development team can find all relative privacy notice entries and update them. Such trackability greatly contributes to the privacy maintenance after the initial software development.

[Benefit-3] Collaboration. Crafting privacy notices is a challenging task that requires both the knowledge of app features and legal requirements. Individual developers may misunderstand the app’s data practices, and misinterpretations of the relevant terminology during the creation of privacy notices may also exist. Thus, generating accurate privacy notices requires the close collaboration of the whole development team instead of relying on one or several individuals. Enabling by the PriBOM, roles like in-house legal teams or legal services providers can work closely with the developers in the task of creating privacy notices. Additionally, PriBOM acts

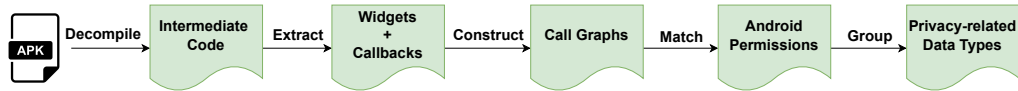


Figure 4: The pipeline of our static analysis module.

as an inventory encapsulating a detailed record of data handling practices, providing clear references for internal privacy inquiries about all roles. Although the UI-based approach falls short of serving as a comprehensive comparison tool, it still helps legal experts identify potential pattern of issues like under-disclosure and over-disclosure privacy practices [61]. Legal experts can be aware of an update whenever a developer adds a new code feature that enables an extra data flow, letting them update the privacy notices with pin-point changes. This shared inventory therefore short-circuits the hand-off bottleneck, reduces misinterpretation of code, and ensures that privacy notices remain synchronized with the software’s actual behavior, ultimately leading to better privacy notice generation.

4 Pre-fill of PriBOM

In this section, we present a pre-fill implementation of PriBOM, and describe details regarding the pipeline. Notably, although our design tailors to Android apps, the concept of PriBOM can be easily adapted on iOS apps with corresponding modifications. We emphasize that the pre-fill does not mean that PriBOM is a fully automated tool. Its collaborative nature requires different roles in the development team to contribute human efforts to the privacy notice generation where PriBOM serves as a privacy information inventory and communication platform.

4.1 Static Analysis

Following Android analysis approaches in previous icon-behavior researches [74, 104, 118], our static analysis-based pre-fill implementation consists of four steps: (1) Widget and Callback Method Extraction; (2) Call Graph Construction; (3) Android Permission Extraction; and (4) Permission-Data Mapping. Figure 4 provides an overview of the pipeline.

Widget and Callback Method Extraction. In the first step, the implementation pipeline takes the Android application package (APK) file as input, and leverages JADX [10] to disassemble it. The purpose of this step is to extract widget components, associated sources such as icons, and corresponding callback methods that respond to UI interactions. Icons (such as PNG or JPG files) are commonly bound with UI widgets in the layout configuration file. For example, XML attribute *android:background* is used to set a background image for a widget component. Therefore, we parse the layout configuration files based on a summary list [31, 118] of XML attributes used to bind icons to extract the icon sources associated with the widgets. For UI interactions like clicks, we extract events and their corresponding callback methods, such as *onClick*. Widgets in Android apps may bind to callback methods either through XML attributes or programmatically through API calls like *findViewById*, which links to a unique widget ID. To analyze these bindings comprehensively, we employ GATOR [90, 112, 113], an Android static analysis tool based on Soot [63], to extract widgets and callback methods.

Call Graph Construction. We employ AndroGuard [6], a popular Android static analysis tool, to construct a call graph for each

identified callback methods. These graphs reflect the invocation relationships between methods, with nodes representing methods and edges indicating their interactions. With a callback method as the entry point, each graph not only tracks direct API invocations but also the cascade of method calls they may initiate. As the call graph of a callback method can be considered as a subgraph of the call graph of the entire app [104], our pre-fill pipeline resorts to AndroGuard [6] to construct a complete call graph of the app, and then extract the call graph for each identified callback method. Ultimately, mappings between the widget component and its reachable methods can be obtained.

Android Permission Extraction. This step maps each widget component to its associated potential Android permission requests. Our pre-fill pipeline traverses the call graph of the widget’s callback method to extract all the API calls, and leverages AndroGuard [6] to obtain a mapping from APIs to Android permissions and determine which permissions are requested. This step associates widget components with their corresponding Android permission uses.

Permission-Data Mapping. Our pre-fill pipeline establishes associations between widget components and interaction-driven data practices by mapping Android permissions to relevant data types. Following the approach in prior works [75, 88], we group the dangerous Android permissions into ten different data type categories. For example, a widget associated with Android permission *ACCESS_COARSE_LOCATION* is considered to be related to data type *Location*, e.g., the user interaction with this widget leads to the application’s data practice of accessing the user’s location information. Dangerous Android permissions refer to permissions with a protection level of *dangerous* in the Official Android API Documentation [15].

TPL Detection. Third-party libraries (TPLs) are constitutional in mobile app development. However, insufficient understanding of TPLs used may lead to developers’ non-comprehensive understanding of the privacy behaviors of the application [33, 61, 66, 68, 70]. We adapt the existing state-of-the-art TPL detector, LibScan [103], to extract the TPLs and version information based on code features and sophisticated similarity analysis. It establishes pairwise class correspondences between app and TPL classes, computes confidence scores, and determines TPL presence.

Notably, our pre-fill pipeline is highly modular, which means each individual module, like the TPL detector, can be easily replaced by more powerful ones. Therefore, we are not overly fixated on performance but instead provide developers and companies with the freedom to modify module implementation choices through coupling in real-world scenarios.

4.2 Privacy Notice Analysis

Privacy notice generation involves not just initial creation but also continual updates and maintenance. A common scenario is maintaining consistency between existing privacy notices, evolving software, and updating regulations. To enhance the alignment between privacy prompts and actual data practices in applications, PriBOM

includes the Privacy Notice Disclosure section. The pre-fill implementation of this section analyzes and processes two common forms of privacy notices, privacy policy and privacy label, and matches them with the corresponding data types documented in PriBOM.

Privacy Policy Segmentation. The objective of this process is to obtain privacy policy segments related to certain data types. For example, segments of data type *Location* will include all descriptions related to *location* in the privacy policy, such as whether it will be collected and the purpose of such collection. We adopt the multi-level privacy policies processing strategy in [83, 111] to perform segment extraction, which takes the app’s privacy policy as input and extract sentences related to each data type through paragraph-level headings classification and sentence-level keyword searching and phrase similarity calculation. We employ prior work [83, 111] for this privacy policy extraction and segmentation process.

Privacy Label Processing. This workflow extracts privacy practice disclosures from the application’s privacy label, e.g. the DSS [22] in Google Play. The privacy label information in the DSS includes the collected data type, the purpose of collection, and whether this collection is optional. Specifically, we record the existence and collection purpose of the data type, as well as whether the granting of this data type is optional for application users. We then map the data types with the data types extracted through the static analysis pipeline discussed in section 4.1. An example of PriBOM is presented in Table 2 of a real mobile application [11] for the UI widget whose ID is “2131296311” and the type is *android.view.MenuItem*. From the PriBOM, developers can easily obtain that this UI widget is related to permission request, *android.permission.ACCESS_COARSE_LOCATION*, for coarse positioning and indicates its corresponding data type, *Location*. In addition, from the Privacy Notice Disclosure section, developers can notice the disclosure in the privacy policy about *Coarse Location*, “...may collect your geo-location information to optimize user experience...”; and corresponding privacy label *Approximate Location* information, which is *optional* for data collection and *App functionality, Analytics, Advertising or marketing* as its purposes.

Our design tailors to mobile apps, and generalizing the concept of PriBOM to arbitrary software faces challenges but also has potential. One main challenge is heterogeneous data flow, as in complex systems the privacy-relevant data flows may traverse differently, making data tracing harder and may demand hybrid static–dynamic analysis. Another challenge is software heterogeneity, as modern systems are often polyglot and multi-platform. This diversity complicates the privacy tracking. To generalize the PriBOM concept, strategies such as utilizing domain-specific privacy pivot points as anchors for documenting privacy-related information can be explored, e.g., sensor interfaces for IoT apps [48].

5 Human Evaluation

To explore PriBOM’s usefulness and prompt further requirements, we conducted an online survey to examine the perspectives of people in various software development roles on PriBOM.

5.1 User Study Design

The user study aims to evaluate the perceived usefulness of PriBOM and prompt insights and further requirements to adapt to specific

Table 3: Participants demographics. Each answer is labeled with the count of participant(s) that select it. “N.A.m.” stands for “North America”.

Role	Team Size	Gender	Continent
Junior developer (59)	<10 (89)	Male (94)	Europe (85)
Senior developer (19)	10 - 20 (29)	Female (55)	Africa (45)
Project manager (21)	20 - 50 (19)	Unknown (1)	Asia (12)
UI designer (22)	>50 (13)	-	N.A.m. (5)
Legal team (13)	-	-	Oceania (3)
Others (16)	-	-	-

needs in the real world. To ensure participants fully understand our PriBOM design, we introduce each section of PriBOM individually and provide an example, as shown in Table 2. In addition, to make sure participants are properly prompted, we deliberately put the data field and descriptions on the side for quick reference.

For the questionnaire, we developed our survey questions based on the specific section design of PriBOM and perspectives from related studies and guidelines [33, 62, 102, 105]. After iterative refinement and pilot studies with 10 participants in total, our final survey contains 26 statements, with which participants rate their agreement levels on a 5-point Likert scale ranging from 1 (strongly disagree) to 5 (strongly agree). Table 4 enumerates the statements. The questions covers six perspectives: (1) General Design of PriBOM, (2) Widget Identifier, (3) Codebase and Permission, (4) TPL, (5) Privacy Notice Disclosure, and (6) Usability and Practicality. Drawing on the pilot study results, we also include four free-text open-ended questions to seek deeper perspectives from participants. The open-ended questions are listed in Table 5.







We target participants with working experience in software development, including various roles in the development team, such as creative design, software engineering, legal, and product management. The responses of pilot study are excluded.

5.2 Participant Recruitment

Following the participant recruitment processes in prior works in Usable Privacy and Security field [73, 83, 105], we published the survey via Qualtrics [23], and recruited participants through Prolific [21], a commonly used recruitment platform of sustainably gathering survey responses at scale. We use the participant screener in Prolific to recruit participants.

We consider responses that are uncompleted or with a completion time of less than 2 minutes to be invalid. In total, we received 150 valid survey responses from participants with diverse backgrounds. The median time to complete the survey is 14 minutes and 33 seconds. Our participants span across 35 countries include 94 males and 55 females, with one participant preferring not to say, which conforms the reality that there are more male practitioners in software industry. We also include 3 demographic questions in the survey questionnaire to understand the roles of participants in software development, years of experience, and the size of the team they work in. An overview of the demographic information is presented in Table 3. The role with the highest proportion is junior developer (39%), while other participants reported to be UI designers (15%), project managers (14%), senior developers (13%), legal team members (9%), and other roles such as CEO. The team size among participants is also diverse. The most common team

Table 4: Our survey results of participants’ agreement on statements related to PriBOM. “Ave.” stands for the average Score. “Distr.” denotes the distribution of the responses. From left to right are strongly disagree (1) to strongly agree (5).

No.	Statement	Ave.	Distr.
General Design of PriBOM			
S ₁	[Intuitiveness] The data fields in PriBOM appear logically structured and are understandable.	3.91	
S ₂	[Format] The layout and format of PriBOM are intuitive for developers with varying levels of experience.	3.83	
S ₃	[Relevance] Information in the PriBOM is essential and contributes to privacy management.	4.02	
Design of the Widget Identifier Section of PriBOM			
S ₄	[Identification Precision] PriBOM's design for widget identification is precise enough to pinpoint privacy issues to a specific UI component.	3.69	
S ₅	[Clarity] The widget name and type in the PriBOM is helpful in providing clear and common terminology for team members.	4.05	
Design of the Codebase and Permission Section of the PriBOM			
S ₆	[Codebase Accessibility] The path/location fields in PriBOM intended for codebase access promote better manageability.	3.76	
S ₇	[Event & Handler] The 'Event' and 'Handler' data field in PriBOM aid in tracing data flows in response to user events.	3.92	
S ₈	[Permission] Listing 'Permission' requirements in PriBOM may increase transparency regarding data access needs and permission requests.	4.09	
S ₉	[Data Type] Including data types and associated permissions in PriBOM is instrumental for providing privacy information.	4.01	
S ₁₀	[API-Level Awareness] The inclusion of specific Android API levels in PriBOM is relevant and necessary for privacy documentation.	3.73	
Design of the TPL Section of PriBOM			
S ₁₁	[Management] Including third-party library information in PriBOM may support a more thorough privacy review.	3.85	
S ₁₂	[Discrepancy] Documenting TPL versions may help identify discrepancies of privacy practices between different versions.	3.97	
S ₁₃	[Record] The date fields of version date in PriBOM would help in maintaining a record of privacy-related updates.	3.89	
S ₁₄	[Update Awareness] The data fields of version date in PriBOM would help developers aware of the TPL updates about privacy practices.	3.87	
Design of the Privacy Notice Disclosure Section of PriBOM			
S ₁₅	[Alignment] The Privacy notice fields in PriBOM adequately guides the incremental development of accurate privacy notices.	3.71	
S ₁₆	[Traceability] This section in PriBOM can help trace data practices to relevant descriptions in privacy policy.	3.85	
S ₁₇	[Trackability] This section in PriBOM can facilitate tracking from disclosures in privacy policy to related data practices in code.	3.83	
Usability and Practicality of PriBOM			
S ₁₈	[Communication] The PriBOM is a practical solution for efficient privacy-related communication between different roles in development team.	3.94	
S ₁₉	[Privacy Notice Generation] The PriBOM can streamline privacy notice generation and management for development teams.	3.92	
S ₂₀	[Disclosure-Behaviour Alignment] The PriBOM would make it easier to align privacy notice pieces with app's actual software behaviours.	4.02	
S ₂₁	[Transparency Promotion] The PriBOM can enhance the transparency about data handling within development teams.	4.03	
S ₂₂	[Privacy Awareness] Implementing the PriBOM could lead to improved privacy awareness among the development team members.	3.99	
S ₂₃	[Risks Mitigation] The PriBOM could systematically reduce the risks of overlooking privacy concerns during the development process.	3.96	
S ₂₄	[Scalability] The PriBOM is a scalable solution that could be adapted for different project sizes and complexities.	3.69	
S ₂₅	[Consent Identification] The PriBOM effectively aids in identifying user-consent-required data collection practices.	3.81	
S ₂₆	[Inquiry Response] PriBOM could reduce the effort needed to respond to privacy-related inquiries.	3.79	

size is less than 10 people (59%), while there are also 9% of participants working in a team with more than 50 people. We believe our participants’ backgrounds are sufficiently diverse for our survey. For ethics consideration, please refer to Section 5.3.

5.3 Ethics Considerations

We recruited participants through Prolific [21], a highly regarded online platform for researchers to recruit participants for UPS studies. We only collect basic and non-identifiable demographic information. Participants receive monetary rewards at a rate of £7.49 per hour, as recommended by Prolific. We ensure the risk of participating in this research is minimum, and no disturbing content is distributed in the survey. A consent and withdrawal information sheet was presented before the survey to inform participants of their rights, including the ability to withdraw at any point before submitting by exiting the survey page. Participants give their consent by proceeding with the survey. All information provided by participants is treated confidentially and we will not share the collected data to any third parties.

5.4 Result Analysis

The results of the survey are detailed in Table 4. We calculate the average agreement score and present the distribution of Likert-scale responses. Overall, participants largely agree with our statements regarding PriBOM. For open-ended responses, we employ the methodology of deductive thematic analysis [37, 46] to identify key insights. Given that the free-text questions are already structured around specific sections of our survey (e.g., Q_1 corresponds to Design of the Codebase and Permission Section in Table 4), we align our qualitative analysis with predefined two-layer (sections and statements) a priori rather than emerging new codes and themes from responses. Below, we discuss the results in more detail.

5.4.1 Design of PriBOM. Most participants agree or strongly agree with the statements related to the general design. The intuitiveness of PriBOM’s design is well received with 85.33% agreement, suggesting that most participants found the data fields in PriBOM logically organized and conducive to understanding. The responses regarding the precision and clarity of Widget Identification show a trend toward agreement. Notably, most participants (83.33%) agree or strongly agree PriBOM is helpful in providing clear terminology.

Table 5: The free text open-ended questions in our survey.

No.	Open-Ended Question
Q_1	How would implementing PriBOM in your projects impact the management and documentation of permission request and data collection? Please briefly describe any potential advantages or challenges.
Q_2	Reflecting on your previous projects, how would detailed tracking of Third-Party Libraries (TPLs) as proposed by PriBOM facilitate to manage privacy compliance?
Q_3	What potential benefits do you see in the PriBOM method of documenting data practices and their associated disclosures in the privacy notice?
Q_4	Reflecting on your own experiences, how do you think implementing PriBOM in the development team might influence the collaboration between different roles toward privacy notice (e.g., privacy policy) management and generation?

The results of the Codebase and Permission Section suggest a favorable perception among participants, particularly in their recognition of PriBOM's functional contributions to privacy management. The positive attitude (80.67%) toward the inclusion of *Permission* and *Data Type* fields highlight their importance. Interestingly, the statement regarding the necessity of permission information in improving data transparency received the highest average agreement score (4.09) among all statements. PriBOM's approach has also been recognized. For example, one participant stated *"I believe a system like this would be helpful to give our clients more peace of mind on the quality of our privacy policies"* (Q_1), indicating participants believed PriBOM was useful for creating accurate privacy policies.

The results of the TPL Section illustrate a strong positive perception regarding PriBOM's capabilities in managing TPL information. 74.67% of the participants agree or strongly agree that documenting TPL versions may help identify discrepancies in privacy practices between different TPL versions. One participant wrote *"I believe the key terms here are the dates and update version records. With privacy laws changing all the time, it would be a way developers could track the changes necessary to TPLs to ensure compliance"* (Q_2). Participants also expressed their agreement with the trackability of PriBOM, as stated, *"...if this is done properly, you have a concrete evidence where things went wrong"*. Such support could relieve the challenge of limited technical knowledge discussed in Section 2.2.

The results of the Privacy Notice Disclosure Section also indicate a positive perception, particularly in terms of traceability (72%) and trackability (70.67%). As participants mentioned in Q_3 , *"I believe the tracking and traceability components benefit the user by providing clear and concise management opportunities for accuracy"*. Moreover, the role of PriBOM in regulatory compliance of applications has been supported, as *"It ensures that the developers access and/or control the information in line with the law or regulations set"*. Since the privacy notice generation involves collaboration, the collaborative nature of PriBOM has also been recognized, *"Seems like the greatest benefit of this, in particular, is to facilitate communication between devs and other departments"*. Such collaboration with legal teams helps address the challenges of privacy knowledge absence.

As for the usability and practicality, the results indicate strong approval (83.33%) for PriBOM in enhancing privacy-related communication across different development roles. Stated in Q_4 , *"I think it will influence a culture within the workspace that is focused on privacy and security and creating awareness in this regard"*, indicating that PriBOM may address the organizational environment challenges discussed in Section 2.2. The role of PriBOM toward creating privacy policy is also well recognized:

"this common language investors clear communication and understanding among team members, facilitating more efficient collaboration in drafting and updating privacy policies."

Highlight-1: Participants highly value the intuitiveness and clarity of PriBOM's design, with high agreement on its capability to provide structured data fields and transparent privacy information. Results affirm that PriBOM significantly bolsters privacy-related communication, exhibiting a high level of usability.

5.4.2 Viewpoint Comparison Across Roles. During the data analysis, we observed that different roles hold different views on PriBOM. To further investigate this difference, we separated the participants based on their roles and calculated the agreement scores for each group on the statements separately. Although different roles showed similar agreement in many statements, we concluded two findings.

The first finding is the discrepancies in the perspectives between frontend UI designers and other roles. For instance, The legal team's agreement score for S_{10} is significantly higher than that of UI designers, as shown in Figure 5(a). This statement describes the necessity of API-level information for privacy information management. The legal team's average score reached 4.00, while the average score for UI designers was 3.63, indicating a divergence of opinions between them, with the former largely agreeing on it and the latter not. We believe this result is reasonable since legal experts typically have a higher privacy awareness, while UI designers may consider this issue less. This relatively low level of agreement may lead to UI designers lacking a rigorous and serious attitude towards privacy when dealing with such information.

Similarly, UI designers scored an average of 3.82, while legal teams scored significantly higher at 4.31 on the statement concerning the impact of PriBOM on enhancing privacy awareness among team members (Figure 5(d)). This means that though legal experts may see PriBOM as a critical tool for privacy practicing, UI designers might appreciate its role but to a lesser extent, as their direct interaction with privacy as a compliance or educational tool is less pronounced. Distinct differences in perception between the project managers and UI Designers were observed in the disclosure-behaviour alignment aspect of PriBOM (Figure 5(c)). The project managers, who need to ensure the development aligns with business and compliance objectives, show a higher agreement score of 4.19, suggesting a strong recognition of the utility of PriBOM in streamlining and clarifying the alignment between privacy disclosures and the actual behaviors. Regarding the maintenance of a record of privacy-related updates (Figure 5(b)), UI designers gave a lower score of 3.64, compared to senior developers who scored it at 4.16. While senior developers are likely to value features that aid in documenting and archiving changes, which supports better version

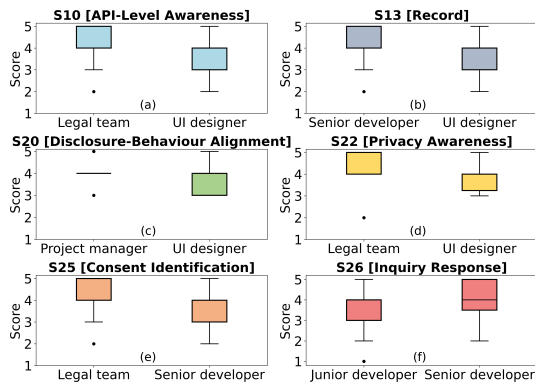


Figure 5: Examples of the differences of agreement scores regarding survey statements among different roles.

control and compliance tracking, UI designers may not perceive it as impactful to their role.

While UI designers often give scores that differ significantly from other raters, differences in opinion also occur between other roles. For instance, in the case of aiding in identifying practices that require user consent (Figure 5(e)), legal teams rated PriBOM’s effectiveness at 4.00, whereas senior developers gave it a lower score of 3.68. This difference can be attributed to the legal team’s acute need for precise tools in drafting and verifying privacy policies and consent protocols. Conversely, one of senior developers, who often focus on actual coding following privacy rules, commented “*I think it is complicated to follow these steps...*”.

Our second finding is differences in experience can lead to changes in viewpoints. For example, senior developers agree more than junior developers that PriBOM could reduce the effort needed to respond to privacy-related inquiries (S_{26}), as presented in Figure 5(f). This statement relates to whether PriBOM could provide support in addressing privacy inquiries. Since PriBOM serves as a privacy information inventory, we believe the development team can respond faster based on the information in PriBOM, rather than manually extracting them after receiving requests. The senior developers’ average score was 4.05, while the average score for UI designers was 3.78, suggesting that senior developers largely agree with our statement, while junior developers may not. We believe this could be due to the more experience of senior developers who may have a better understanding of privacy inquiry and the role PriBOM can play in it. Differently, junior developers may not have much exposure to privacy inquiries, and the uncertainty may guide them to choose more neutral options.

Overall, the human evaluation of PriBOM across multiple dimensions has garnered largely positive feedback, underscoring its usefulness. However, participants also provided feedback on the potential challenges of PriBOM in practical adoption. “*It would be challenging to let my company’s elderly employees get up to date on this since most of them are stuck to their old ways.*” Another participant also wrote “*it would be a challenge and a matter of time for everyone to learn how to work based on PriBOM*”. These feedbacks indicate that PriBOM should continue to improve usability and reduce the learning curve. Nevertheless, participants still expressed affirmation of PriBOM, demonstrating that PriBOM is well-received

by the development community for its comprehensive approach to embedding privacy into the software development lifecycle, as

“I think it’s a good Model that deserves to be tested, it may have a very positive impact and improve the way we look at privacy policies and management of projects in future.”

Highlight-2: Although PriBOM is widely recognized for its advantages, perceptions of its value differ notably across various roles. Experience within the same role can further shape these viewpoints.

5.4.3 Perspectives Held by Non-technical Roles. PriBOM provides a systematic solution for communication and collaboration between technical and non-technical roles. As we discussed in the formative study, existing research has intensively discussed the opinions held by technical roles. To understand the perspectives of non-technical roles, namely the legal team, UI designers, and project managers, on PriBOM, we analyzed their free-text responses and summarized the frequently mentioned topics.

Efficiency. All three non-technical roles mentioned the efficiency benefits of PriBOM. What the legal team cares about is that PriBOM can reduce their heavy workload and speed up their work. A legal expert working in a team of 10 to 20 people commented “*Our team can get a quick guide to be compliant with all the ness...PriBOM will help us with fast answer...*”, and another legal expert also stated “*it could speed up and automatize some processes*”. Differently, two UI designers mentioned “*high rates of turnover within projects*” and “*team dynamics*”, which means team members may frequently join or leave the project due to various reasons such as changing job roles or shifts in project focus. In such environment, PriBOM could maintain consistency and enhance knowledge transfer by ensuring new members can quickly come up to speed on the existing privacy practices and policies without a steep learning curve. The project manager’s thinking focuses on the role of PriBOM in decision-making. “*I like the clarity and effectiveness of it...*”, commented by a product manager working in a team with more than 50 people,

“it would certainly ease decision making since it has very in depth record and makes it easier to trace things”.

All three non-technical roles mentioned challenges in the practical usage and adaptation of PriBOM. A legal expert in a team with over 50 people commented that “*Each department had its own unique ideas of how privacy notice works*”, and challenges pointing by UI designers include “*initial setup complexity, potential integration issues with existing systems*” and “*it was something people would need to get used to first*”. Differently, the project manager’s concern includes the effort required to maintain PriBOM within a large team. “*seems like a lot of effort to get started and to maintain it.*” Fulfilling PriBOM is indeed a chore, therefore we introduce a pre-fill that can save developers from adding extra burden.

Traceability and Trackability. This discussion mainly focuses on the benefits of constructing privacy chains for compliance analysis. As stated by a legal team member, “*It enhances the traceability in the privacy chain connecting front-ed UI and the product’s policy*”. One UI designer also mentioned “*efficient chain flows and better root cause analysis*”, and another one stated that “*It’s like a trust-builder because it ensures that what’s promised in the notice matches up with*

what's actually happening with their data, making compliance efforts smoother too". A product manager considered the communication between technical and non-technical roles involved in the privacy chain, and wrote

"Imagine a central platform where developers link data practices to specific components. This transparency would streamline communication between developers and privacy specialists".

Communication. The discussion of this topic by the three roles focuses on different perspectives. Product managers think of responsibility and time management as key aspects. One product manager mentioned the benefit of *"clear and consistent segmentation of responsibility that impact liability"*, indicating his belief on PriBOM in facilitating responsibility assignment within the team. Another manager wrote *"improve time management, brings members to closer because they can see everything ..."*, supporting the use of PriBOM to improve communication efficiency. For UI designers, they agree with the potential of PriBOM as a common communication platform, as *"It's like a common language that everyone can understand, making it easier for developers, privacy officers, and legal teams to work together effectively"*. One participant from the legal team also mentioned *"I think we could get a better collaboration between the departments where we all are more clear about our roles"*. Overall, many responses support the idea of PriBOM, a systematic solution, can facilitate communication between different roles.

Influence. Several responses mentioned the influence of PriBOM on the software development environment. For example, one legal expert wrote *"It allows you to have a complete record which encourages healthy and smart data practices"*, indicating that PriBOM could promote a proactive approach to privacy management, encouraging teams to consider privacy implications continuously rather than as a periodic compliance exercise. One UI designer stated

"I currently find the knowledge regarding privacy policy, permissions etc. to be lacking within the team, especially on a management level. I feel like it is currently regarded as an afterthought ... this could hopefully increase awareness among all team members and facilitate communication as well as implementing a proper workflow for managing privacy settings".

This comment reveals a critical gap in privacy awareness and management at the management level within teams, where privacy considerations are not fully integrated into the development life-cycle. The need for a systematical solution like PriBOM for various roles is emphasized in such context.

Highlight-3: The efficiency benefit of PriBOM is widely endorsed for its ability to streamline workflows, facilitate swift onboarding, bridge communication gaps between diverse roles, and enhance both efficiency and privacy environment. However, the feedback highlights a need for ongoing refinements to ensure its practical adaptation and alignment with real-world demands across different roles.

Future Enhancements. Participants also proposed additional information to include in PriBOM in different use cases. For example, one product manager suggested the potential inclusion of *"security level for guarding PII fields"*. This information may benefit the implementation of more targeted, effective data protection strategies based on data sensitivity and decision-making. Another

valuable suggestion from a project manager is *"Data about TPL should include github or web page of the project"*, facilitating quick investigation and smooth work in due diligence and verifying the credibility and security practices of TPLs.

Current findings reveal the promising potential of PriBOM to effectively bridge the gap between various roles, enhancing communication, compliance, and efficiency within development teams. Nonetheless, the deployment of the PriBOM approach within diverse environments underscores the necessity for further efforts to refine its adaptability and to better meet the specific real-world needs highlighted by various roles.

6 Limitations

While we propose a pre-fill of PriBOM and assess the usefulness, there are still limitations. First, the pre-fill generation is dependent on the performance of static analysis tools, and limited performance could result in restricted quality of pre-fill. Second, our user study design can be improved by involving a in-depth interview to mitigate the possibility of social desirability or acquiescence biases. Third, the study sample is skewed toward small development teams. This imbalance may affect the generalizability of the results. But we believe our study sample reflects the real world situation that small development teams account for a larger proportion. Our analysis results do not stem solely from the small-team participants. The free text responses reveal the topics frequently mentioned by participants in teams larger than 20 people, such as transparency, regulatory compliance pressure, and efficiency. Although we propose this promising privacy enhancing mechanism in software development process, real practicability needs to be further verified. As our current design is UI-centric, it can miss non-UI-triggered permission use. Future work will therefore incorporate with non-UI analyses to enhance its completeness. Still, as the first of such kind, we believe this work can ignite deeper thinking on the aspect of software engineering at the systematic level, considering all practitioners to solve the long-standing privacy policy challenges.

7 Conclusion and Future Work

Privacy regulations commonly require developers to provide authentic privacy notices, e.g., privacy policies or labels, in communicating their apps' privacy practices. However, developers often struggle to create and maintain accurate privacy notices. We introduce PriBOM as a systematic and collaborative solution for the development team to craft accurate privacy notices. As a privacy information inventory, we leverage static analysis and privacy notice analysis techniques to implement PriBOM. The role of PriBOM in enhancing privacy-related communication is well received with 83.33% agreement, underscoring the usefulness and practicability of PriBOM in fostering a privacy-conscious development workflow. Lastly, we discuss in depth the implications of our results, including differences in views held by different roles and privacy notice generation, especially non-technical roles. Furthermore, this structured approach can be extended beyond privacy contexts to address data confidentiality challenges relevant in ESG (Environmental, Social, and Governance) reporting and compliance. By clearly documenting and managing sensitive data elements systematically, PriBOM

could help organizations maintain consistent and transparent standards for data confidentiality, crucial for ESG governance.

Acknowledgments

We thank Ze Shi Li for useful feedback and help on earlier versions of this paper. Dr. Haggag is supported by a National Intelligence Post-doctoral Fellowship. Prof. Grundy and Dr.Haggag are supported by ARC Laureate Fellowship FL190100035.

References

- [1] 2014. *Australian Privacy Principles (APP)*. <https://www.oaic.gov.au/privacy/australian-privacy-principles> Accessed: 2022-05-03.
- [2] 2016. *General Data Protection Regulation (GDPR)*. <https://gdpr-info.eu/> Accessed: 2022-04-25.
- [3] 2018. *California Consumer Privacy Act of 2018 (CCPA)*. <https://oag.ca.gov/privacy/ccpa> Accessed: 2022-04-25.
- [4] 2022. *Iubenda*. <https://www.iubenda.com/en/> Accessed: 2022-03-28.
- [5] 2023. *SPDX Overview*. <https://spdx.dev/about/>
- [6] 2024. *Androguard*. <https://github.com/androguard/androguard> Accessed: 2024-02-27.
- [7] 2024. *App privacy policy generator*. <https://app-privacy-policy-generator.firebaseapp.com/> Accessed: 2024-02-27.
- [8] 2024. *Cyber Resilience Act*. <https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32024R2847> Accessed: 2025-05-13.
- [9] 2024. *Iubenda*. <https://termly.io/> Accessed: 2022-02-27.
- [10] 2024. *JADX*. <https://github.com/skylot/jadx> Accessed: 2024-02-27.
- [11] 2024. *Lep's World 2*. https://play.google.com/store/apps/details?id=at.ner.lepsWorld2&hl=en_US Accessed: 2024-05-20.
- [12] 2024. *LinkedIn Post Kuijt*. https://www.linkedin.com/posts/dr-emilie-kuijt-baa79a50_privacy-activity-718878625558492160-KSxy?utm_source=share&utm_medium=member_desktop Accessed: 2024-06-07.
- [13] 2024. *LinkedIn Post Nini*. <https://www.linkedin.com/pulse/5-challenges-life-data-protection-officer-nini-prasad/> Accessed: 2024-06-07.
- [14] 2024. *LinkedIn Post Tricia Higgins*. https://www.linkedin.com/posts/higginstricia_managing-gdpr-non-compliance-activity-7081663097794371584-ac20?utm_source=share&utm_medium=member_desktop Accessed: 2024-06-07.
- [15] 2024. *Manifest.permission | Android Developers*. <https://developer.android.com/reference/android/Manifest.permission> Accessed: 2024-03-31.
- [16] 2024. *Matcha - A Google Play Safety Label Generator*. <https://plugins.jetbrains.com/plugin/20141-matcha--a-google-play-safety-label-generator> Accessed: 2024-05-06.
- [17] 2024. *Permissions on Android*. <https://developer.android.com/guide/topics/permissions/overview> Accessed: 2024-04-1.
- [18] 2024. *Prepare for Google Play's data disclosure requirements*. <https://developers.google.com/android/guides/play-data-disclosure> Accessed: 2025-05-11.
- [19] 2024. *Privacy policy online*. <https://www.privacypolicyonline.com/> Accessed: 2022-02-27.
- [20] 2024. *Privacypolicies*. <https://www.privacypolicies.com/> Accessed: 2022-02-27.
- [21] 2024. *Prolific*. <https://www.prolific.com/> Accessed: 2024-03-25.
- [22] 2024. *Provide information for Google Play's Data safety section*. <https://support.google.com/googleplay/android-developer/answer/10787469?hl=en> Accessed: 2024-03-25.
- [23] 2024. *Qualtrics*. <https://www.qualtrics.com/au/> Accessed: 2024-03-25.
- [24] 2024. *simplelegal*. <https://www.simplelegal.com/blog/corporate-legal-departments-data-privacy> Accessed: 2024-04-9.
- [25] 2025. *Developer Program Policy*. <https://support.google.com/googleplay/android-developer/answer/16070163?hl=en> Accessed: 2025-05-11.
- [26] 2025. *Provide information for Google Play's Data safety section*. <https://support.google.com/googleplay/android-developer/answer/10787469?hl=en&zippy=> Accessed: 2025-05-11.
- [27] 2025. *User Data*. <https://support.google.com/googleplay/android-developer/answer/10144311?hl=en> Accessed: 2025-05-11.
- [28] Paul C Adams. 2020. Agreeing to surveillance: Digital news privacy policies. *Journalism & Mass Communication Quarterly* 97, 4 (2020), 868–889.
- [29] Vincent Zimmer Amy Nelson, Jiewen Yao. 2021. Traceable Firmware Bill of Materials Overview. <https://uefi.org/node/4950>.
- [30] Benjamin Andow, Akhil Acharya, Dengfeng Li, William Enck, Kapil Singh, and Tao Xie. 2017. Uiref: analysis of sensitive user inputs in android applications. In *Proceedings of the 10th acm conference on security and privacy in wireless and mobile networks*. 23–34.
- [31] Vitalii Avdiienko, Konstantin Kuznetsov, Isabelle Rommelfanger, Andreas Rau, Alessandra Gorla, and Andreas Zeller. 2017. Detecting behavior anomalies in graphical user interfaces. In *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*. IEEE, 201–203.
- [32] Rebecca Balebako and Lorrie Cranor. 2014. Improving app privacy: Nudging app developers to protect user privacy. *IEEE Security & Privacy* 12, 4 (2014), 55–58.
- [33] Rebecca Balebako, Abigail Marsh, Jialiu Lin, Jason Hong, and Lorrie Faith Cranor. 2014. The privacy and security behaviors of smartphone app developers. In *Workshop on Usable Security*. Citeseer, 1–10.
- [34] Iain Barclay, Alun Preece, Ian Taylor, and Dinesh Verma. 2019. Towards traceability in data ecosystems using a bill of materials model. *arXiv preprint arXiv:1904.04253* (2019).
- [35] Eliot Beer. 2022. Firmware security in the spotlight after novel ransomware attacks. <https://thetack.technology/firmware-attacks-focus/>.
- [36] Jarni Blakkarly and Daniel Graham. 2024. *Privacy policy comparison reveals half have poor readability*. <https://www.choice.com.au/consumers-and-data/protecting-your-data/data-laws-and-regulation/articles/privacy-policy-comparison> Accessed: 2024-03-25.
- [37] Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. *Qualitative research in psychology* 3, 2 (2006), 77–101.
- [38] Harry Brignull. 2010. Types of deceptive design. *Deceptive Design* (2010).
- [39] Duc Bui, Brian Tang, and Kang G Shin. 2023. Detection of inconsistencies in privacy practices of browser extensions. In *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2780–2798.
- [40] João Caramujo and Alberto Manuel Rodrigues Da Silva. 2015. Analyzing privacy policies based on a privacy-aware profile: The Facebook and LinkedIn case studies. In *2015 IEEE 17th Conference on Business Informatics*, Vol. 1. IEEE, 77–84.
- [41] Orlando Amaral Cejas, Sallam Abualhaija, and Lionel C Briand. 2024. CompAi: A Tool for GDPR Completeness Checking of Privacy Policies using Artificial Intelligence. In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*. 2366–2369.
- [42] Brian Ka Chan. 2017. Artificial Intelligence Bill of Materials (AI-BOM). <https://minddata.org/bill-of-artificial-intelligence-materials-boaimBrian-Ka-Chan-AI>.
- [43] Jieshan Chen, Jiamou Sun, Sidong Feng, Zhenchang Xing, Qinghua Lu, Xiwei Xu, and Chunyang Chen. 2023. Unveiling the Tricks: Automated Detection of Dark Patterns in Mobile Applications. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*. 1–20.
- [44] Jiatao Cheng, Zhefan Chen, Xun Zhu, Jiaxing Yang, Yifan Chen, and Yuhong Nan. 2025. A Privacy-driven UI Exploration Framework for Mobile App Compliance Check. In *Proceedings of the 2025 4th International Conference on Cryptography, Network Security and Communication Technology*. 43–49.
- [45] Andrei Costin. 2022. Securing Your Iot Device With Fboms From Devastating Cyberattacks. <https://euhubs4data.eu/blog/securing-iot-device-with-fboms/>.
- [46] Benjamin F Crabtree and William F Miller. 1992. A template approach to text analysis: developing and using codebooks. (1992).
- [47] CycloneDX. 2022. Hardware Bill of Materials (HBOM). <https://github.com/CycloneDX/bom-examples/tree/master/HBOM>.
- [48] Earlene Fernandes, Justin Paupore, Amir Rahmati, Daniel Simionato, Mauro Conti, and Atul Prakash. 2016. {FlowFence}: Practical data protection for emerging {IoT} application frameworks. In *25th USENIX security symposium (USENIX Security 16)*. 531–548.
- [49] Carlos Flavián and Miguel Guinalíu. 2006. Consumer trust, perceived security and privacy policy: three basic elements of loyalty to a web site. *Industrial management & data Systems* 106, 5 (2006), 601–620.
- [50] Omar Haggag, Alessandro Pedace, Shidong Pan, and John Grundy. 2025. An analysis of privacy regulations and user concerns of finance mobile applications. *Information and Software Technology* (2025), 107756.
- [51] Hamza Harkous, Kassem Fawaz, Rémi Lebrete, Florian Schaub, Kang G Shin, and Karl Aberer. 2018. Polisis: Automated analysis and presentation of privacy policies using deep learning. In *27th {USENIX} security symposium ({USENIX} security 18)*. 531–548.
- [52] Stephen Hendrick. 2022. Software Bill of Materials (SBOM) and Cybersecurity Readiness. <https://tinyurl.com/293v3xte>.
- [53] Stefan Albert Horstmann, Samuel Domiks, Marco Gutfleisch, Mindy Tran, Yasemin Acar, Veelasha Moonsamy, and Alena Naiakshina. 2024. “Those things are written by lawyers, and programmers are reading that.” Mapping the Communication Gap Between Software Developers and Privacy Experts. *Proceedings on Privacy Enhancing Technologies* (2024).
- [54] Jianjun Huang, Zhichun Li, Xusheng Xiao, Zhenyu Wu, Kangjie Lu, Xiangyu Zhang, and Guofei Jiang. 2015. {SUPOR}: Precise and scalable sensitive user input detection for android apps. In *24th USENIX Security Symposium (USENIX Security 15)*. 977–992.
- [55] Dilara Keküllüoğlu and Yasemin Acar. 2023. “We are a startup to the core”: A qualitative interview study on the security and privacy development practices in Turkish software startups. In *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2015–2031.

- [56] Thomas Kelepouris, Katerina Pramataris, and Georgios Doukidis. 2007. RFID-enabled traceability in the food supply chain. *Industrial Management & data systems* 107, 2 (2007), 183–200.
- [57] Patrick Gage Kelley, Joanna Bresee, Lorrie Faith Cranor, and Robert W Reeder. 2009. A "nutrition label" for privacy. In *Proceedings of the 5th Symposium on Usable Privacy and Security*, 1–12.
- [58] Patrick Gage Kelley, Lucian Cesa, Joanna Bresee, and Lorrie Faith Cranor. 2010. Standardizing privacy notices: an online study of the nutrition label approach. In *Proceedings of the SIGCHI Conference on Human factors in Computing Systems*, 1573–1582.
- [59] Patrick Gage Kelley, Lorrie Faith Cranor, and Norman Sadeh. 2013. Privacy as part of the app decision-making process. In *Proceedings of the SIGCHI conference on human factors in computing systems*, 3393–3402.
- [60] Katharine Kemp. 2020. Concealed data practices and competition law: why privacy matters. *European Competition Journal* 16, 2-3 (2020), 628–672.
- [61] Rishabh Khandelwal, Asmit Nayak, Paul Chung, and Kassem Fawaz. 2023. Unpacking Privacy Labels: A Measurement and Developer Perspective on Google's Data Safety Section. *arXiv preprint arXiv:2306.08111* (2023).
- [62] Barbara A Kitchenham and Shari L Pfleeger. 2008. Personal opinion surveys. In *Guide to advanced empirical software engineering*, Springer, 63–92.
- [63] Patrick Lam, Eric Bodden, Ondrej Lhoták, and Laurie Hendren. 2011. The Soot framework for Java program analysis: a retrospective. In *Cetus Users and Compiler Infrastructure Workshop (CETUS 2011)*, Vol. 15.
- [64] Hao-Ping Hank Lee, Lan Gao, Stephanie Yang, Jodi Forlizzi, and Sauvik Das. 2024. "I Don't Know If We're Doing Good. I Don't Know If We're Doing Bad": Investigating How Practitioners Scope, Motivate, and Conduct Privacy Work When Developing AI Products. In *USENIX Security Symposium*.
- [65] Jiawei Li, Jiahao Liu, Jian Mao, Jun Zeng, and Zhenkai Liang. 2025. UI-CTX: Understanding UI Behaviors with Code Contexts for Mobile Applications. (2025).
- [66] Tianshi Li, Yuvraj Agarwal, and Jason I Hong. 2018. Coconut: An IDE plugin for developing privacy-friendly apps. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 2, 4 (2018), 1–35.
- [67] Tianshi Li, Lorrie Faith Cranor, Yuvraj Agarwal, and Jason I Hong. 2024. Matcha: An IDE Plugin for Creating Accurate Privacy Nutrition Labels. *arXiv preprint arXiv:2402.03582* (2024).
- [68] Tianshi Li, Elizabeth Louie, Laura Dabbish, and Jason I Hong. 2021. How developers talk about personal data and what it means for user privacy: A case study of a developer forum on reddit. *Proceedings of the ACM on Human-Computer Interaction* 4, CSCW3 (2021), 1–28.
- [69] Tianshi Li, Elijah B Neundorfer, Yuvraj Agarwal, and Jason I Hong. 2021. Honysuckle: Annotation-guided code generation of in-app privacy notices. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 5, 3 (2021), 1–27.
- [70] Tianshi Li, Kayla Reiman, Yuvraj Agarwal, Lorrie Faith Cranor, and Jason I Hong. 2022. Understanding challenges for developers to create accurate privacy nutrition labels. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, 1–24.
- [71] Jiali Lin, Shahriyar Amini, Jason I. Hong, Norman Sadeh, Janne Lindqvist, and Joy Zhang. 2012. Expectation and purpose: understanding users' mental models of mobile app privacy through crowdsourcing. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing (Pittsburgh, Pennsylvania) (UbiComp '12)*, Association for Computing Machinery, New York, NY, USA, 501–510. <https://doi.org/10.1145/2370216.2370290>
- [72] Jiali Lin, Shahriyar Amini, Jason I Hong, Norman Sadeh, Janne Lindqvist, and Joy Zhang. 2012. Expectation and purpose: understanding users' mental models of mobile app privacy through crowdsourcing. In *Proceedings of the 2012 ACM conference on ubiquitous computing*, 501–510.
- [73] Yanzi Lin, Jaideep Juneja, Eleanor Birrell, and Lorrie Cranor. 2023. Data Safety vs. App Privacy: Comparing the Usability of Android and iOS Privacy Labels. *arXiv preprint arXiv:2312.03918* (2023).
- [74] Vikas K Malviya, Yan Naing Tun, Chee Wei Leow, Ailys Tee Xynyn, Lwin Khin Shar, and Lingxiao Jiang. 2023. Fine-Grained In-Context Permission Classification for Android Apps Using Control-Flow Graph Embedding. In *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, IEEE, 1225–1237.
- [75] Ryan McConkey and Oluwafemi Olukoya. 2023. Runtime and design time completeness checking of dangerous android app permissions against GDPR. *IEEE Access* (2023).
- [76] Alecia M McDonald and Lorrie Faith Cranor. 2008. The cost of reading privacy policies. *Isilp* 4 (2008), 543.
- [77] Mehdi Mirakhorli, Derek Garcia, Schuyler Dillon, Kevin Laporte, Matthew Morrison, Henry Lu, Viktoria Kosinski, and Christopher Enoch. 2024. A Landscape Study of Open Source and Proprietary Tools for Software Bill of Materials (SBOM). *arXiv preprint arXiv:2402.11151* (2024).
- [78] Yuhong Nan, Min Yang, Zheming Yang, Shunfan Zhou, Guofei Gu, and Xiaofeng Wang. 2015. {UIPicker} : {User-Input} Privacy Identification in Mobile Applications. In *24th USENIX Security Symposium (USENIX Security 15)*, 993–1008.
- [79] Trung Tin Nguyen, Michael Backes, Ninja Marnau, and Ben Stock. 2021. Share first, ask later (or never?) studying violations of {GDPR's} explicit consent in android apps. In *30th USENIX Security Symposium (USENIX Security 21)*, 3667–3684.
- [80] Trung Tin Nguyen, Duc Cuong Nguyen, Michael Schilling, Gang Wang, and Michael Backes. 2021. Measuring user perception for detecting unexpected access to sensitive resource in mobile apps. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*, 578–592.
- [81] NTIA. 2019. Roles and Benefits for SBOM Across the Supply Chain. https://www.ntia.gov/files/ntia/publications/ntia_sbom_use_cases_roles_benefits-nov2019.pdf.
- [82] Shidong Pan, Thong Hoang, Dawen Zhang, Zhenchang Xing, Xiwei Xu, Qinghua Lu, and Mark Staples. 2023. Toward the cure of privacy policy reading phobia: Automated generation of privacy nutrition labels from privacy policies. *arXiv preprint arXiv:2306.10923* (2023).
- [83] Shidong Pan, Zhen Tao, Thong Hoang, Dawen Zhang, Tianshi Li, Zhenchang Xing, Xiwei Xu, Mark Staples, Thierry Rakotoarivelo, and David Lo. 2024. A NEW HOPE: Contextual Privacy Policies for Mobile Applications and An Approach Toward Automated Generation. In *33rd USENIX Security Symposium (USENIX Security 24)*, USENIX Association, Philadelphia, PA, 5699–5716. <https://www.usenix.org/conference/usenixsecurity24/presentation/pan-shidong-hope>
- [84] Shidong Pan, Dawen Zhang, Mark Staples, Zhenchang Xing, Jieshan Chen, Xiwei Xu, and Thong Hoang. 2024. Is It a Trap? A Large-scale Empirical Study And Comprehensive Assessment of Online Automated Privacy Policy Generators for Mobile Apps. In *33rd USENIX Security Symposium (USENIX Security 24)*, USENIX Association, Philadelphia, PA, 5681–5698. <https://www.usenix.org/conference/usenixsecurity24/presentation/pan-shidong-trap>
- [85] Xiang Pan, Yinzhi Cao, Xuechao Du, Boyuan He, Gan Fang, Rui Shao, and Yan Chen. 2018. FlowCog: Context-aware Semantics Extraction and Analysis of Information Flow Leaks in Android Apps. In *27th USENIX Security Symposium (USENIX Security 18)*, USENIX Association, Baltimore, MD, 1669–1685. <https://www.usenix.org/conference/usenixsecurity18/presentation/pan>
- [86] Alfredo J Perez, Sherali Zeadally, and Jonathan Cochran. 2018. A review and an empirical analysis of privacy policy and notices for consumer Internet of things. *Security and Privacy* 1, 3 (2018), e15.
- [87] John N Petroff and Arthur V Hill. 1991. A framework for the design of lot-tracing systems for the 1990s. *Production and Inventory Management Journal* 32, 2 (1991), 55.
- [88] Muhammad Sajidur Rahman, Pirouz Naghavi, Blas Kojusner, Sadia Afroz, Byron Williams, Sara Rampazzi, and Vincent Bindschaedler. 2022. Permpress: Machine learning-based pipeline to evaluate permissions in app privacy policies. *IEEE Access* 10 (2022), 89248–89269.
- [89] Joel Reardon, Álvaro Feal, Primal Wijesekera, Amit Elazari Bar On, Narseo Vallina-Rodriguez, and Serge Egelman. 2019. 50 ways to leak your data: An exploration of apps' circumvention of the android permissions system. In *28th USENIX security symposium (USENIX security 19)*, 603–620.
- [90] Atanas Rountev and Dacong Yan. 2014. Static reference analysis for GUI objects in Android software. In *Proceedings of Annual IEEE/ACM International Symposium on Code Generation and Optimization*, 143–153.
- [91] William Seymour, Noura Abdi, Kopo M Ramokapane, Jide Edu, Guillermo Suarez-Tangil, and Jose Such. 2023. Voice App Developer Experiences with Alexa and Google Assistant: Juggling Risks, Liability, and Security. *arXiv preprint arXiv:2311.08879* (2023).
- [92] Yashothara Shanmugarasa, Shidong Pan, Ming Ding, Dehai Zhao, and Thierry Rakotoarivelo. 2025. Privacy Meets Explainability: Managing Confidential Data and Transparency Policies in LLM-Empowered Science. In *Proceedings of the Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, 1–8.
- [93] Meixue Si, Shidong Pan, Dianshu Liao, Xiaoyu Sun, Zhen Tao, Wenchang Shi, and Zhenchang Xing. 2024. A solution toward transparent and practical AI regulation: Privacy nutrition labels for open-source generative AI-based applications. *arXiv preprint arXiv:2407.15407* (2024).
- [94] Trevor Stalnaker, Nathan Wintersgill, Oscar Chaparro, Massimiliano Di Penta, Daniel M German, and Denys Poshyvanyk. 2024. BOMs Away! Inside the Minds of Stakeholders: A Comprehensive Study of Bills of Materials for Software Systems. In *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering*, 1–13.
- [95] Mohammad Tahaei, Ruba Abu-Salma, and Awais Rashid. 2023. Stuck in the permissions with you: Developer & end-user perspectives on app permissions & their privacy ramifications. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, 1–24.
- [96] Mohammad Tahaei, Alisa Frik, and Kami Vaniea. 2021. Privacy champions in software teams: Understanding their motivations, strategies, and challenges. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, 1–15.

- [97] Mohammad Tahaei, Kami Vaniea, and Naomi Saphra. 2020. Understanding privacy-related questions on stack overflow. In *Proceedings of the 2020 CHI conference on human factors in computing systems*. 1–14.
- [98] Daniel Votipka, Seth M Rabin, Kristopher Micinski, Thomas Gilray, Michelle L Mazurek, and Jeffrey S Foster. 2018. User comfort with android background resource accesses in different contexts. In *Fourteenth Symposium on Usable Privacy and Security (SOUPS 2018)*. 235–250.
- [99] Lun Wang, Usmann Khan, Joseph Near, Qi Pang, Jithendaraa Subramanian, Neel Somani, Peng Gao, Andrew Low, and Dawn Song. 2022. {PrivGuard}: Privacy regulation compliance made easier. In *31st USENIX Security Symposium (USENIX Security 22)*. 3753–3770.
- [100] Liu Wang, Dong Wang, Shidong Pan, Zheng Jiang, Haoyu Wang, and Yi Wang. 2025. A Big Step Forward? A User-Centric Examination of iOS App Privacy Report and Enhancements. In *2025 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 3878–3896.
- [101] Xiaoyin Wang, Xue Qin, Mitra Bokaei Hosseini, Rocky Slavin, Travis D Breaux, and Jianwei Niu. 2018. Guileak: Tracing privacy policy claims on user input data for android applications. In *Proceedings of the 40th International Conference on Software Engineering*. 37–47.
- [102] Charles Weir, Ben Hermann, and Sascha Fahl. 2020. From needs to actions to secure apps? the effect of requirements and developer practices on app security. In *29th USENIX security symposium (USENIX security 20)*. 289–305.
- [103] Yafei Wu, Cong Sun, Dongrui Zeng, Gang Tan, Siqi Ma, and Peicheng Wang. 2023. {LibScan}: Towards More Precise {Third-Party} Library Identification for Android Applications. In *32nd USENIX Security Symposium (USENIX Security 23)*. 3385–3402.
- [104] Shengqu Xi, Shao Yang, Xusheng Xiao, Yuan Yao, Yayuan Xiong, Fengyuan Xu, Haoyu Wang, Peng Gao, Zhuotao Liu, Feng Xu, et al. 2019. Deepint: Deep icon-behavior learning for detecting intention-behavior discrepancy in mobile apps. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 2421–2436.
- [105] Boming Xia, Tingting Bi, Zhenchang Xing, Qinghua Lu, and Liming Zhu. 2023. An empirical study on software bill of materials: Where we stand and the road ahead. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 2630–2642.
- [106] Boming Xia, Dawen Zhang, Yue Liu, Qinghua Lu, Zhenchang Xing, and Liming Zhu. 2023. Trust in software supply chains: Blockchain-enabled sbom and the aibom future. *arXiv preprint arXiv:2307.02088* (2023).
- [107] Anhao Xiang, Weiping Pei, and Chuan Yue. 2023. PolicyChecker: Analyzing the GDPR completeness of mobile apps' privacy policies. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*. 3373–3387.
- [108] Xusheng Xiao, Xiaoyin Wang, Zhihao Cao, Hanlin Wang, and Peng Gao. 2019. Iconintent: automatic identification of sensitive ui widgets based on icon classification for android apps. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 257–268.
- [109] Yue Xiao, Zhengyi Li, Yue Qin, Xiaolong Bai, Jiale Guan, Xiaojing Liao, and Luyi Xing. 2023. Lalaine: Measuring and characterizing {Non-Compliance} of apple privacy labels. In *32nd USENIX Security Symposium (USENIX Security 23)*. 1091–1108.
- [110] Yue Xiao, Adwait Nadkarni, and Xiaojing Liao. 2023. Enhancing Transparency and Accountability of TPLs with PBOM: A Privacy Bill of Materials. In *Proceedings of the 2024 Workshop on Software Supply Chain Offensive Research and Ecosystem Defenses*. 1–11.
- [111] Fuman Xie, Yanjun Zhang, Chuan Yan, Suwan Li, Lei Bu, Kai Chen, Zi Huang, and Guangdong Bai. 2022. Scrutinizing privacy policy compliance of virtual personal assistant apps. In *Proceedings of the 37th IEEE/ACM international conference on automated software engineering*. 1–13.
- [112] Shengqian Yang, Haowei Wu, Hailong Zhang, Yan Wang, Chandrasekar Swaminathan, Dacong Yan, and Atanas Rountev. 2018. Static window transition graphs for Android. *Automated Software Engineering* 25 (2018), 833–873.
- [113] Shengqian Yang, Dacong Yan, Haowei Wu, Yan Wang, and Atanas Rountev. 2015. Static control-flow analysis of user-driven callbacks in Android applications. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*. Vol. 1. IEEE, 89–99.
- [114] Wei Yang, Xusheng Xiao, Benjamin Andow, Sihan Li, Tao Xie, and William Enck. 2015. AppContext: Differentiating Malicious and Benign Mobile App Behaviors Using Context. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*. Vol. 1. 303–313. <https://doi.org/10.1109/ICSE.2015.50>
- [115] Le Yu, Xiapu Luo, Jiachi Chen, Hao Zhou, Tao Zhang, Henry Chang, and Hareton KN Leung. 2018. Ppchecker: Towards accessing the trustworthiness of android apps' privacy policies. *IEEE Transactions on Software Engineering* 47, 2 (2018), 221–242.
- [116] Le Yu, Tao Zhang, Xiapu Luo, and Lei Xue. 2015. Autoppg: Towards automatic generation of privacy policy for android applications. In *Proceedings of the 5th Annual ACM CCS Workshop on Security and Privacy in Smartphones and Mobile Devices*. 39–50.
- [117] Nusrat Zahan, Elizabeth Lin, Mahzabin Tamanna, William Enck, and Laurie Williams. 2023. Software Bills of Materials Are Required. Are We There Yet? *IEEE Security & Privacy* 21, 2 (2023), 82–88.
- [118] Yanjie Zhao, Li Li, Xiaoyu Sun, Pei Liu, and John Grundy. 2021. Icon2Code: Recommending code implementations for Android GUI components. *Information and Software Technology* 138 (2021), 106619.
- [119] Lu Zhou, Chengyongxiao Wei, Tong Zhu, Guoxing Chen, Xiaokuan Zhang, Suguo Du, Hui Cao, and Haojin Zhu. 2023. {POLICYCOMP}: Counterpart Comparison of Privacy Policies Uncovers Overbroad Personal Data Collection Practices. In *32nd USENIX Security Symposium (USENIX Security 23)*. 1073–1090.
- [120] Sebastian Zimmeck, Rafael Goldstein, and David Baraka. 2021. PrivacyFlash Pro: Automating Privacy Policy Generation for Mobile Apps.. In *NDSS*, Vol. 2. 4.

8 Appendix

8.1 The Methodology and Scope of Literature in Formative Study

Table 6 presents the methodology and scope of studies on privacy-related challenges faced by developers. The most common methodology is the semi-structured interview, which is used in over half of the listed studies. Other studies utilize qualitative methods such as email interactions and analysis of online community discussions. A rich tapestry of collected data reflects the multifaceted nature of privacy challenges in the development process. Participant backgrounds vary widely, encompassing a spectrum from Android and iOS app developers to users of online communities like Stack Overflow and Reddit. This variety underscores the breadth of privacy concerns across different platforms and development environments. It is worth noting that participants of some studies include project managers and legal teams, emphasizing the cross role nature of addressing privacy issues in software development and reflecting the interaction between technical and non-technical factors in privacy management.

8.2 Challenges for Legal Experts

As shown in Figure 6, legal experts like DPO in enterprises emphasized the severe challenges they face, including various factors such as the lack of resources, zero cooperation among organisational units, and unstructured data. Among them, the necessity for support from other organizational units is highlighted, underscoring the importance of distributing privacy tasks across various development roles to balance the workload effectively. PriBOM serves as a systematic approach to liberating legal experts from heavy and difficult tasks.

Table 6: Methodology and scope of studies on privacy-related challenges faced by developers.

Study	Methodologies	Participant Background
Khandelwa et al. [61]	Qualitative study involving email interactions	More than 3,500 Android app developers
Li et al. [70]	Remotely observational study on Zoom, semi-structured interview	12 iOS app developers from various platforms
Li et al. [66]	Semi-structured interview	9 Android app developers, including independent and full-time developers, researchers, and others
Li et al. [68]	Qualitative analysis of discussions from the /r/androiddev subreddit	Users of the /r/androiddev subreddit
Balebako et al. [33]	Semi-structured interview, online survey	13 app developers varied in terms of company size and app types, 228 U.S. app developers and product managers
Lee et al. [64]	Semi-structured interview	35 industry practitioners, including researchers, software engineers, and designers
Seymour et al. [91]	Semi-structured interview	30 developers varied in terms of experience and professionals
Weir et al. [102]	Online Survey	345 Google Play Android developers
Keküllüoğlu et al. [55]	Semi-structured interview	16 developers in Turkish software startups
Tahaei et al. [96]	Semi-structured interview	12 Privacy Champions who promote privacy in development teams
Tahaei et al. [97]	Qualitative analysis of privacy-related questions from Stack Overflow	Users of Stack Overflow

3. Zero cooperation among other organisational units

Being a DPO is no easy task, and their hands are always full. They have to uphold data protection laws and practices, monitor compliance, support business operations and data handling, notify other teams and authorities about data breaches, and foster a security-aware culture. It is also impossible for a single person who is a DPO to handle everything on their own which is why it is crucial to divide all the tasks among other organisational units to balance the workload. This way they can cooperate with the other units, too.

So it is essential that other organisational units cooperate with the DPO to identify the problems and work on them together to ensure a better working environment for them. It would also be helpful if the top management could lend a helping hand to the DPOs as that would help them immensely.

5. Unstructured Data

Unstructured data is termed as information that is unorganised and does not have a fixed data model. It's like unmanaged data that is like a nightmare for every DPO who is out there working on it. It is text-heavy and contains text-based information such as date, place, etc. It cannot be stored in any form of predefined rational data structure and neither can it be organised or processed properly. According to Waterford Technologies "Unstructured data accounts for approximately 80% of data." This makes the job of a DPO extremely difficult.

In larger organizations, the DPO has their own team in place.
In an enterprise, sometimes an entire department.

But many DPOs and privacy professionals are lone wolves.

They usually carry the weight of the company's privacy program entirely on their own shoulders: responsible for the privacy of customers, users, partners and employees.

This is why as a DPO, you need a support system:

It could be appointing departmental privacy champions as your go-to point of contact in each department.

It could be creating a privacy executive forum (council, committee - whatever works) that includes executives from Legal, Security, IT, HR, Product, R&D, Marketing and Sales to meet on a recurrent basis.

Preferably, it should be both.

This will help you share that huge weight on your shoulders.

Make privacy a team effort.

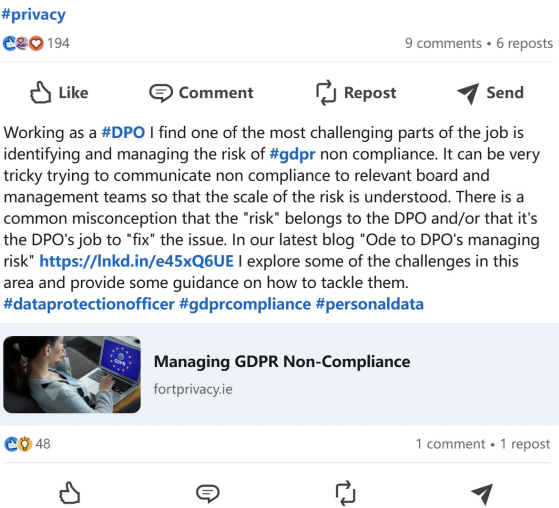


Figure 6: The screenshots of data protection officers' posts on LinkedIn [12–14].