IDPFilter: Mitigating Interdependent Privacy Issues in Third-Party Apps

Shuaishuai Liu[1] and Gergely Biczók[12]

[1] CrySyS Lab, Dept. of Networked Systems and Services,
Budapest Univ. of Technology and Economics, Hungary
{sliu,biczok}@crysys.hu
[2] HUN-REN-BME Information Systems Research Group

Abstract. Third-party applications have become an essential part of today's online ecosystem, enhancing the functionality of popular platforms. However, the intensive data exchange underlying their proliferation has increased concerns about interdependent privacy (IDP). This paper provides a comprehensive investigation into the previously underinvestigated IDP issues of third-party apps. Specifically, first, we analyze the permission structure of multiple app platforms, identifying permissions that have the potential to cause interdependent privacy issues by enabling a user to share someone else's personal data with an app. Second, we collect datasets and characterize the extent to which existing apps request these permissions, revealing the relationship between characteristics such as the respective app platform, the app's type, and the number of interdependent privacy-related permissions it requests. Third, we analyze the various reasons IDP is neglected by both data protection regulations and app platforms and then devise principles that should be followed when designing a mitigation solution. Finally, based on these principles and satisfying clearly defined objectives, we propose IDPFilter, a platform-agnostic API that enables application providers to minimize collateral information collection by filtering out data collected from their users but implicating others as data subjects. We implement a proof-of-concept prototype, IDPTextFilter, that implements the filtering logic on textual data, and provide its initial performance evaluation with regard to privacy, accuracy, and efficiency.

Keywords: interdependent privacy · third-party apps · permissions · information filtering · Application Programming Interface

1 Introduction

Third-party applications, commonly referred to as apps, occupy a significant place within the present-day Internet ecosystem. These apps enhance the functionality and features of popular platforms, including but not limited to mobile operating systems, social media networks, web browsers, and cloud services. The proliferation of third-party apps is predicated upon the sharing of data, which is a crucial aspect of their respective platforms. The vast, diverse, and constant

data exchange, however, has given rise to increasingly pressing concerns regarding privacy.

Recently, various app platforms have undertaken the initiative to improve their privacy preservation mechanisms. For instance, since the release of iOS v14.5[3], applications have been required to obtain user consent before tracking user data from other applications and websites. Additionally, Android v12 implemented a privacy dashboard[4] enabling users to monitor app permission usage with increased accuracy, while Android v13 introduced fine-grain data sharing, e.g., granting access to a single photo instead of an entire album or all photos. These developments are encouraging steps in the ongoing quest for enhanced transparency, user control, and overall privacy. However, they do not address the issue of interdependent privacy (IDP) [1], where data shared with an app by a user may contain personal and potentially sensitive information about their friends, contacts, or colleagues; without their knowledge or control. This is particularly pertinent in the context of third-party apps.

In recent years, some of the most prominent cases of interdependent privacy have been associated with Facebook. The Cambridge Analytica scandal, which was widely reported in the media, serves as a notable example [2]. In this instance, the app "thisisyourdigitallife" harvested 87 million Facebook profiles and used the data to create comprehensive personal psychological profiles. This information was then utilized to deliver personalized political advertisements to influence the outcome of the 2016 US presidential election (and others). The app exploited the mechanism of *collateral information collection* (i.e., the information collection process enabled by privacy interdependence) on Facebook, where it was installed by $270,000$ users but was able to access the profiles of tens of millions of friends owing to the controversial design of permissions in Facebook's Graph API [2].

In February 2021, Facebook reached a settlement in a class action lawsuit involving the use of facial recognition technology in its photo tagging function. The lawsuit, which was initiated in Illinois in 2015, claimed that Facebook created and stored scans of users' faces without their permission. The photo tagging feature allowed users to tag their friends in photos uploaded to Facebook, thereby establishing a personal connection to the friends' profiles without their consent[5]. The lawsuit resulted in Facebook compensating over 1.6 million users with a total settlement amount of $650 million, which is one of the largest privacy-related settlements to date. This case serves as a clear illustration of interdependent privacy, where the uploading user consciously gave Facebook permission to display their photo, but the users who were tagged in the photo suffered a loss of privacy without even being aware of it.

In addition to high-profile incidents, there have also been a number of less publicized instances of interdependent privacy violations. In 2015, a security vul-

[3] https://developer.apple.com/app-store/user-privacy-and-data-use/

[4] https://www.androidauthority.com/android-privacy-dashboard-1233846/

[5] https://www.theguardian.com/technology/2021/feb/27/facebook-illinois-privacy-lawsuit-settlement

nerability allowed third-party applications to access Google+ user profile data, which was not discovered and patched until March 2018[6]. This bug enabled developers to also obtain non-public profile fields for both the user and their friends, affecting an estimated 500,000 profiles. Unsurprisingly, Google chose not to disclose this issue to the public, not to diminish its momentary competitive advantage over Facebook. Another concerning example is the popular TrueCaller Android app, which is capable of blacklisting spam numbers. The app requires the uploading of the installing user's address book to its servers, which constitutes an interdependent privacy issue as noted by the Article 29 Working Party in 2017[7]. TrueCaller also allows users to tag unknown numbers after calls and upload them for other users to see, which, in 2019, jeopardized the cover of an investigative journalist in a non-friendly country. Fortunately, no harm came to the journalist and their sources; however, the situation posed a considerable threat to their physical safety[8].

The above incidents naturally invite three research questions.

- *RQ1*: *are interdependent privacy issues pervasive among third-party app platforms?*
- *RQ2*: *do actual apps request the permissions enabling collateral information collection on their respective platforms?*
- *RQ3*: *is it possible to devise a technical solution to mitigate collateral information collection on third-party app platforms?*

In this paper, we answer all three research questions affirmatively. Specifically, the contribution of this paper is four-fold.

First, we conduct an analysis of the permission systems of multiple app platforms and identify those permissions that have the potential to cause interdependent privacy issues. Second, we collect datasets from the respective app platforms and characterize the extent to which existing apps request these permissions. Our findings indicate that the category of the app is a reliable predictor for the number of interdependent privacy-related permissions that are requested. Third, we discuss potential transparency and control measures for mitigating the issues above. Finally and most importantly, we design IDPFilter, a platform-agnostic API that enables application providers to filter out data collected from their users but implicating other natural persons as data subjects. We also implement a proof-of-concept prototype, IDPTextFilter, and provide its initial evaluation.

The rest of this paper is organized as follows. Section 2 briefly surveys the related work. Section 3 analyzes the permission structure of multiple widely-used third-party app platforms (Android, browser extensions, Google Workspace, and Zoom Marketplace) and identifies permissions (potentially) invoking interdependent privacy. Section 4 introduces our dataset and characterizes the proliferation of such permissions requested by real existing apps from their respective users.

[6] `https://www.wsj.com/articles/google-exposed-user-data-feared-repercussions-of-disclosing-to-public-1539017194`
[7] `https://ec.europa.eu/newsroom/article29/items/610173`
[8] `https://privacyinternational.org/node/2997`

Section 5 discusses prospective transparency- and control-enhancing techniques and devises IDPFilter, a platform-agnostic filtering API able to partially mitigate interdependent privacy issues of third-party apps. Section 6 describes our proof-of-concept prototype IDPTextFilter. Finally, Section 7 provides a discussion and concludes our paper.

2 Related work

In this section, we provide a succinct overview of existing literature in the field of interdependent privacy and third-party applications.

Interdependent privacy captures the networked characteristics of privacy-related decisions. Owing to this networked nature, the privacy of individuals is bound to be affected by the actions of others, e.g., Facebook users sharing the data of their friends [1]. In economic terms, unaware fellow users fall victim to a negative externality. Extending this interpretation, a data entry, seemingly concerning a single individual, may actually be also related to (multiple) others because of data correlation [3]. Note that the same concept is known under different monikers, such as collective privacy [4], networked privacy [5] and multiple-subject privacy [6], among others. For a comprehensive overview, we refer the interested reader to [7]. Interdependent privacy affects different types of data and data-sharing scenarios. A subset of attributes from the profile of a social network user may be harvested [1]. The location privacy of certain individuals may be threatened by sharing co-location information [8]. Photo sharing may affect the privacy of friends and bystanders captured in the photo [3]. Even the genetic profile of an individual and associated inferrable medical information might get exposed by an eager relative (i.e., kin genomic privacy) [9]. A common trait among the aforementioned scenarios is that all of them could be instantiated through a variety of third-party apps.

General privacy considerations regarding third-party apps, platforms, permissions, and ecosystems have been a strong focus area of researchers in the last decade. We do not even attempt to give a comprehensive overview here; rather, we highlight a few studies with close relations to this paper. Wang et al. studied the data collection practices of Facebook third-party apps and proposed control mechanisms that can increase transparency [10]. King et al. conducted an exploratory survey on how Facebook users interact with apps and how much they understand the privacy implications of such interaction [11]. Androidleaks uncovered how sensitive data was used once the user gave the required permissions and the Android app was installed [12]. Chia et al. studied app permissions, privacy risk signals, and community ratings on multiple app platforms [13]. FlowDroid and its follow-up works provided taint analysis for Android apps that shed light on potential unintended and malicious data leaks [14]. Reardon et al. explored the many ways apps can circumvent the Android permission system [15]. Finally, Kelley et al. (and many others building on their study) showed that users actually factor in their privacy concerns when choosing between apps if they are presented with easy-to-understand privacy facts before installation [16]. The

above selection of studies clearly demonstrates that i) permission models are imperfect, ii) various privacy leaks do occur in apps, and iii) users act on their concerns when presented with tractable information on app privacy.

Yet, there are only a handful of scientific studies dealing explicitly with interdependent privacy situations regarding third-party apps. Biczok and Chia showed that the personal, relational, and spatial privacy of Facebook users was threatened by their friends [1]. Pu and Grossklags investigated the effect of selfish and other-regarding preferences in social app adoption [17]. Harkous and Aberer analyzed Google Drive apps and pointed out that users suffered more privacy loss owing to their collaborators than their own actions [18]. Finally, Symeonidis et al. presented a comprehensive data analytics, modeling, and legal study on the *collateral information collection* practices of Facebook apps affecting the friends of the user [2]. While these studies and further anecdotal evidence suggest that interdependent privacy issues might be the norm rather than the exception on most third-party app platforms, the research community lacks a data-driven study for available, active, but previously uncharted platforms, such as Android, browser extensions, cloud services, and videoconferencing. One of the objectives of this paper is to fill this gap.

There have been some studies on the sharing of information affecting the privacy of multiple people; these studies have introduced different ideas from various domains, such as law, social sciences, and information technology. Most of these ideas were not designed to solve (partially or fully) IDP issues involving third-party applications; however, they pointed out the inherent challenges around this scene.

On the legal side, pre-GDPR studies pointed out that organizations often did not have the incentives to notify users about threats or privacy breaches [19]. Later, partly as an answer to this, new data protection regulations followed, e.g., the General Data Protection Regulation (GDPR) in Europe, the California Consumer Privacy Act (CCPA) in California, or even the Personal Information Protection Law (PIPL) in China. These regulations laid down the rights of data subjects, the obligations of organizations processing personal data, and the steps of proper information gathering and management. These groundbreaking privacy regulations increased the privacy appetite of users, but also of governments. In fact, the recent decision of the European Data Protection Board on needing explicit consent for targeted ads on Facebook and Instagram (both owned by Meta) is perceived by many as a strong step in the right direction[9]. The formulation of regulations makes service providers pay more attention to the rights of the people whose information is directly collected; however, it ignores the fact that data collected from end-users may include other people's private information. Thus, owing to the complex scope of IDP and the difficulty of formulating preventive clauses, data protection laws have not covered IDP explicitly, making it somewhat of a gray zone [20, 2].

[9] https://www.euractiv.com/section/data-privacy/news/eu-squeezes-meta-on-personal-data-use-for-targeting-ads/

Without explicit mandatory regulations, the mitigation of IDP risks can only be implemented spontaneously, including both users and service providers. Kamleitner et al. proposed a new framework to better understand the IDP problem and outlined some principles for solutions [21]. Besides some potential high-level intervention mechanisms, the article also points out the social responsibility of data controllers and processors in protecting the privacy of consumers in such a complex information-gathering scenario.

Few studies have articulated technical approaches. Gnesi et al. proposed to let users create privacy policies locally and standardize the storage of IDP information [22]. Olteanu et al. designed ConsenShare, a system for users to register personal information in and upload the photos to be shared first; users can then only share photos after obtaining the consent of other people in the photos [3]. The "others" indicated above must also be registered in the system and obtain identifiers. Both of these technology-focused solutions can be interpreted as new third-party applications. On the contrary, we aim to mitigate IDP-related risks in all third-party app platforms.

Marsch et al. proposed a design for an interdependent privacy feedback system that leverages natural language generation to communicate the potential privacy impacts of app permissions regarding both the user and others. The authors concluded that increasing awareness about the interdependent privacy consequences of app permissions could lead to more responsible privacy behavior and a more privacy-friendly app ecosystem [23]. In fact, a well-designed privacy dashboard has proved to be an efficient and user-friendly transparency-enhancing tool (TET) [24]. A privacy dashboard is a feature in apps that allows users to manage their personal data and privacy settings. Such a dashboard has also been proposed in the context of Facebook apps, specifically for raising awareness on IDP [2]. In recent years, big tech companies have made a push towards more transparent data practices and providing users with greater control over their personal data[10] [11], even implementing privacy shortcuts[12], allowing users to quickly adjust their settings from within the app without having to navigate to a separate dashboard. We believe that user feedback on privacy should be a part of any practical risk reduction mechanism; yet, it is satisfactory in itself.

3 Platforms, permissions and interdependent privacy

3.1 Permissions and interdependent privacy

Permission-based access. Third-party app platforms share a common security model, which is based on requesting and granting permissions. App permissions guard the access to i) restricted data, such as location or contact information, and ii) restricted actions, such as taking photos or connecting to the Internet. Generally, the main objectives of app permissions include: i) enabling

[10] https://myaccount.google.com/dashboard
[11] https://legal.yahoo.com/us/en/yahoo/privacy/dashboard/index.html
[12] https://www.facebook.com/help/ipad-app/395495000532167

user control over data shared, ii) achieving transparency so that the user understands what data an app is using and why, and iii) promoting data minimization so that the app accesses and utilizes only the data absolutely required for a specific task the user invokes.

Platforms, e.g., Google's Android, have evolved significantly since their inception to achieve these objectives. Android has introduced install-time and run-time permissions; the latter group includes all individual permissions deemed *dangerous* by the platform. Run-time permissions can be explicitly granted (or denied) by the user through a dedicated pop-up window, shown when the execution of the app reaches a state where the permission is required. On top of this, very recently, Android has included a *privacy dashboard* that shows which apps had used sensitive permissions and for how long in the last 24 hours; also, with easy access to revoke said permissions if so desired. Despite all these improvements in Android and other permission mechanisms, there are still no specific (neither transparency nor mitigation) measures targeted at interdependent privacy.

Rubbing salt into the wound, app platforms' definitions of certain permissions are vague as to what extent the app will obtain and use sensitive private information. Combining this more general transparency issue with the specific flaws mentioned above, two sub-optimal privacy outcomes emerge. First, the user does not have sufficient knowledge of the scope of the information to be shared: others' private data might be transferred to the app without even their knowledge. Second, the user might grant excessive permissions to the app to preserve full functionality. Although this latter has been shown to be an issue with respect to one's own sensitive attributes, it could induce an even more negative impact in the context of interdependent privacy.

Permissions related to interdependent privacy. Corresponding to the above two points, when the permission involved is ambiguous, users pay more attention to protecting their own privacy while ignoring the privacy of their friends [17]. When the number of permissions granted by users to apps becomes larger, interdependent privacy issues often emerge. An obvious example is a top-rated Firefox extension called *AdBlocker Ultimate*. The permission-related warnings of this app are the following: W1) "Access browser tabs", W2) "Store unlimited amount of client-side data", W3) "Access browser activity during navigation", and W4) "Access your data for all websites". Plausibly, the combination of W1, W3, and W4 enables the extension to read the website, detect ads, and replace them with blank boxes. However, the same permissions enable the app to collect, e.g., messages sent to and received from a web-based chat; an outcome that could cause privacy loss to the communication partners of the user, an obvious interdependent privacy scenario, no user would prefer to experience. Furthermore, W2 enables the storage of unlimited personal data collected through W1, W2, and W4; this can allow for observing, e.g., personal communications over a longer period of time. Yet, not granting these requested permissions makes it impossible to install and use the app.

As we would like to quantify the extent to which interdependent privacy issues are present in third-party app platforms, we classify permissions into three pre-defined categories: invoking interdependent privacy (IDP), potentially invoking interdependent privacy (PIDP), and not invoking interdependent privacy (NIDP). If a permission *directly* enables access to private data related to a natural person other than the user herself, it is in IDP; e.g., the `READ_CONTACTS` permission in Android. If a permission *potentially* enables access to private data related to a natural person other than the user herself, it is in PIDP. Such risk can be realized through i) accessing data that *may* implicate multiple parties, such as photos or documents (e.g., `READ_EXTERNAL_STORAGE` in Android); ii) enabling a restricted action that *may create* multi-party data, such as photos or audio recordings (e.g., `RECORD_AUDIO` in Android); and iii) enabling *inference* of other's private data with reasonable efforts, such as location via co-location information from other sources (e.g., `ACCESS_FINE_LOCATION` in Android). Note that granting a PIDP permission does not automatically constitute privacy loss for a third party; the loss is context-dependent and may require additional effort from the app developer or an adversary. If a given permission belongs neither to IDP nor PIDP, then it is in NIDP and not in our focus.

3.2 Platform specifics

Here, we briefly introduce the app platforms we investigated. For practical data availability reasons, we targeted the most popular mobile app platform, Android, two well-known browsers providing an API for third-party extensions (Mozilla Firefox and Opera), and Google Workspace, a cloud-based enterprise collaboration tool bundle. Although these four platforms vary greatly in both their functionality and technical mechanisms, all of them offer the equivalent of an app store, where the access control of apps is based on the user granting permissions.

Android. Android users can download and install more than 3 million apps from the Google Play store, making Android the largest third-party app platform, both by user base and the set of available apps. This popularity has made the platform's permission model change continuously over time while trying to keep a balance between being appealing to both users and third-party developers alike. The current stable OS is v11 (with v12 right around the corner), while the API version, also defining the current permission model, is level 30. Android has evolved into a general-purpose OS with plenty of protected data objects and actions; this amounts to 91 permissions in total, offered to third-party apps). We make 91 our baseline for the total number of relevant permissions. Out of these 91, there are 4 that explicitly and 16 that potentially interfere with others' personal data instantiating interdependent privacy, see Table 1. Note that the pop-up messages, appearing when installing an app from Google Play, contain warnings that can be mapped directly to API-level permissions with reasonable effort.

Table 1. Android permissions: IDP and PIDP

IDP	PIDP
read call log_Phone	read the contents of your USB storage_Photos/Media/Files
read your contacts_Contacts	modify or delete the contents of your USB storage_Photos/Media/Files
modify your contacts_Contacts	approx. location (network-based)_Location
read your text messages (SMS or MMS)_SMS	precise location (GPS and network-based)_Location
	access extra location provider commands_Location
	take pictures and videos_Camera
	read sensitive log data_Device & app history
	read your Web bookmarks and history_Device & app history
	record audio_Microphone
	read the contents of your USB storage_Storage
	modify or delete the contents of your USB storage_Storage
	find accounts on the device_Contacts
	read cal events plus confidential information_Calendar
	add or modify cal events and send emails to guests w/o owners' knowledge_Calendar
	read cell broadcast messages_SMS
	find accounts on the device_Contacts

Browser extensions: permissions and warnings. Although referred to differently, browser extensions are very similar to apps. Extensions usually expand browser functionality and manage user operations. Owing to their objectives and architecture, browser extensions are all about interacting with their respective platforms, oftentimes resulting in obtaining large amounts of information about user operations in the browser in real-time, but also about content downloaded by the browser. Note that browsers are also used to access intranets and other non-public resources, therefore, they might leak a variety of personal (and other confidential) information if something goes wrong. Both Firefox and Opera are based on Chromium, therefore their APIs and permission models facing third-party extensions are all based on the Chrome API (along with Chrome, Edge, Brave and Safari, to be correct). Both browsers have their own extension store.

Albeit they are based on the same APIs, Firefox and Opera have some unique characteristics. They both support the majority of permissions but not all[13], and

[13] `https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/manifest.json/permissions#browser_compatibility`

they both define their own warning messages that users can see before/when they install an extension[14]. In fact, Opera does not show these warnings when installing; they are only visible on their dedicated page in the extension store. Making things more complicated, i) not all permission requests generate warning messages, and ii) warning messages and API-level permissions are not totally consistent: the platforms have decided to simplify warnings for the sake of clarity to the average user. While this is laudable from one aspect, these explanations sometimes do not fully reflect the risks of granting the requested permissions. Exact mappings between permissions and user warnings are hard to find but may be extrapolated from Chrome's official documentation[15]. Since it is only feasible to scrape the extension stores for per app warnings (and not for API-level permissions), we base our analysis on these. Note that our datasets contain information on Manifest V2 extensions; however, the changes introduced in Manifest V3 do not have a significant impact on interdependent privacy[16].

Firefox has 26 different warning messages, 19 of which are potential culprits for interdependent privacy violations, see Table 2. Opera extensions make use of 20 types of warning messages, 13 of which pose a potential threat owing to privacy interdependence, see Table 2. Note that all affected warnings (and their corresponding permissions) are in PIDP, and we omit NIDP due to space constraints. Also, note that all Opera warnings start with "This extension (can/will)".

Google Workspace (formerly GSuite). GSuite is a collaborative enterprise office platform launched by Google in September 2016 that was rebranded to Google Workspace in 2020, when it already had more than 2 billion users. Users do not need to download applications, they only need to edit and share files in the cloud, realizing remote collaboration. The platform has an app store, the Marketplace[17], where business, productivity, and educational tools are offered by third-party developers. One of Workspace's subsystems, Google Drive, has already been shown to leak others' personal information through apps owing to its collaborative nature [18]; however, its permission model has changed completely due to the integration of various Google subsystems into the Workspace. The platform has many specialized permissions catering to its intended usage as a collaborative office productivity solution. Specifically, there are 87 different permissions, out of which 3 are IDP ("See and download your contacts", "View customer-related information" and "View, edit, or permanently delete contacts" and 71 are PIDP (which we omit due to the lack of space). Note that although Workspace is a subscription-based service for enterprises and universities, it hosts huge amounts of private data. What is more, if an employee (usually

[14] e.g. Firefox: `https://support.mozilla.org/en-US/kb/permission-request-messages-firefox-extensions`

[15] `https://developer.chrome.com/docs/extensions/mv2/permission_warnings/#permissions_with_warnings`

[16] `https://developer.chrome.com/docs/extensions/mv3/intro/mv3-overview/`

[17] `https://workspace.google.com/marketplace`

Table 2. Browser extension permissions: PIDP

Firefox	Opera
Access browser tabs	Access your data on all websites
Access browser activity during navigation	Access your tabs and browsing activity
Access your data for named site	Access your data on some websites
Exchange messages with programs other than Firefox	Exchange messages with programs other than Opera
Download files and read and modify the browser's download history	Capture the content of the entire screen or of individual tabs and windows
Access your location	Access data you copy and paste
Access recently closed tabs	Allow other installed extensions and web pages to communicate with this extension
Access your data for all websites	Detect your physical location
Store unlimited amount of client-side data	Manipulate privacy-related settings
Access your data for sites in the named domain	Know which sites you're visiting most often
Read and modify bookmarks	Read and modify bookmarks
Access your data on # other sites	Store an unlimited amount of client-side data
Get data from the clipboard	Read and modify your browsing history
Extend developer tools to access your data in open tabs	
Read the text of all open tabs	
Access browsing history	
Access your data in # other domains	
Access browsing history	
Read and modify browser settings	

a system administrator) installs a third-party app, resulting in a privacy violation for other natural persons, the company can be held responsible as per the GDPR.

Zoom Marketplace. In recent years, the COVID-19 pandemic has prompted the emergence and popularity of novel tools and platforms, with some becoming indispensable in daily life, such as *Zoom*. Owing to its exceptional teleconferencing and remote collaboration capabilities, Zoom continues to thrive even after the lockdown period. Zoom Meeting, a cross-platform application for Windows, macOS, iOS, Android, Chrome OS, and Linux, is recognized for its user-friendly interface and functionality. Key features encompass one-on-one meetings, group video conferences, screen sharing, plugins, browser extensions, and the ability to record meetings and automatically transcribe them.

Zoom Marketplace is the third-part app platform of Zoom; it employs a permission-based system, requiring users to activate a pre-approval switch before installing an application. This switch is unrelated to the permissions requested by the app. Once the switch is enabled and the "add" button is clicked, specific

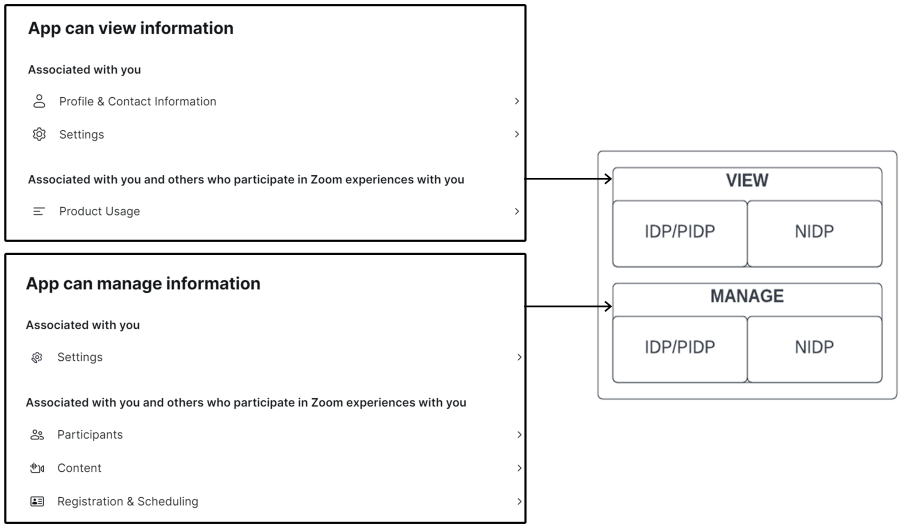

Fig. 1. Zoom Marketplace permission system structure

permissions and authorization options are displayed. A less prominent authorization option above the "authorize" button is unrelated to the app's installation. This option, termed "shared access permissions", lacks a detailed description of its scope and usage. According to Zoom Meeting, this permission means that the app will perform some actions on the user's behalf, such as scheduling a meeting, for which the app will have to obtain this permission and have access to some of the user's information, such as user profiles, and the information of other users of the account.[18]

The authorization option is the critical determinant; users must either accept all permissions or not use the app. The various permissions the app will acquire are detailed on the page. Zoom Marketplace is the first platform found to attempt to differentiate between IDP and NIDP in the permission system. Zoom categorizes permissions into "App can view information" and "App can manage information". Upon reviewing the API documentation, it was discovered that each subset contains two types of instructions. Permissions are divided into two subsets: "Associated with you" and "Associated with you and others you're allowed to access". The second set of descriptions consists of: "Associated with you, others you're allowed to access, and others included in that information" and "Associated with you and others who participate in Zoom experiences with you". Comparing the API documentation with the content displayed on the webpage reveals that for each category on the webpage, the subset displays only one description. The two descriptions have an inclusive relationship, where "Associated with you and others you're allowed to access" contains "Associated

[18] `https://support.zoom.com/hc/en/article?id=zm_kb&sysparm_article=KB0063819#h_01FMWTFQ9RM20TQB9HANS05AMQ`

Table 3. Zoom: View permissions

IDP&PIDP	NIDP
Profile & Contact Information	Settings
Product Usage	Registration Information
Device Information	Account Information
Participant Profile & Contact Information	
Content	
Calendars	

Table 4. Zoom: Manage permissions

IDP&PIDP	NIDP
Profile & Contact Information	Settings
Registration & Scheduling	Account Information
Participants	
Content	
Devices	

with you" and "Associated with you and others who participate in Zoom experiences with you". Interestingly, this classification system (see Fig. 1) essentially adheres to the IDP concept1, horizontally partitioning permissions related solely to the user or those the user can control into one category and those the user cannot control and involve others into another category. Vertically, permissions are divided into view and manage categories. This approach provides more granular control of permissions and enhances transparency and user awareness on the platform. Nevertheless, IDP/PIDP permissions are still present on the platform (see Tables 3 and 4).

It is straightforward to see that each platform has a significant proportion of its permissions and warnings connected to interdependent privacy; see Table 5. This answers *RQ1* affirmatively: *interdependent privacy issues are indeed pervasive among third-party app platforms.* Note that although new versions of platforms are rolled out periodically, the IDP issues are unlikely to disappear: they are inherently present owing to permission-based access control. Unless the access control mechanisms of app platforms are completely re-designed, interdependent privacy is here to stay.

Table 5. Summary: IDP and PIDP permissions/warnings

Platform	No. of permissions/warnings	IDP + PIDP	Ratio
Android	91	20	21.98%
Firefox	26	19	73.08%
Opera	20	13	65%
Google Workspace	87	74	85.06%
Zoom Marketplace	12	9	75%

Table 6. Number of apps with IDP/PIDP

Platform	Apps with IDP	Apps with PIDP	Total Apps	Ratio
Android	1,029	8,307	10,589	78.66%
Firefox	0	13,704	16,546	82.82%
Opera	0	1,421	1,682	84.48%
Google Workspace	29	845	882	97.62%
Zoom Marketplace	1,832	1,803	1,841	99.51%

4 Application-level statistics

4.1 Data collection

We collected datasets by scraping the app stores of 5 different third-party app platforms: Android (10,589 apps), Mozilla (16,546), Opera (1,682), and Google Workspace (882) in late 2020 and early 2021; and Zoom App Marketplace (1,841) in early 2023. Each record contains all available meta-data, e.g., app name, category, permissions/warnings, number of users, rating, etc., depending on the actual platforms. Due to the app stores' protection against scraping, i) we did not manage to collect enough data for Chrome and Edge extensions; therefore, we omitted these platforms from our analysis; ii) our Android dataset contains only a fragment of the millions of available apps (yet, large and random enough to be significant). To the best of our knowledge, we collected complete datasets (i.e., reachable by scraping) for Firefox, Opera, Google Workspace, and Zoom App Marketplace. Note that automatic scraping was infeasible for Google Workspace; we manually collected information on all available apps. By collecting URLs of different applications, we could track detailed information, including names, categories, and permissions requested by each app. Note that Zoom Marketplace does not list features like the number of users and ratings, making it hard to measure the popularity of Zoom apps. All datasets and scraping scripts are available for download.[19]

4.2 Do real apps request IDP/PIDP permissions?

Table 6 shows the number of apps that requested at least one IDP or PIDP permission. The last column calculates the proportion of the union of these apps versus all the apps in the dataset. It is clear that the vast majority of apps are affected, as evidenced by a proportion larger than 80% for all platforms. Note that the browser platforms offer only PIDP permissions.

To further study the privacy protection permissions of apps, we calculated the permissions requested by each app. Regarding Android (Google Play store), 17.2% of apps requested the permission "Contacts", which means that their users have shared their contact list with the third-party developer, directly exposing others' personal data without their knowledge. Besides, 78.66% of apps have

[19] `https://github.com/shuai20/IDP_Filter`

Table 7. Average number of IDP&PIDP permissions per app

Platform	IDP/PIDP permissions	Total permissions	Proportion
Android	4.40	11.21	39.3%
Firefox	0.83	0.85	97.6%
Opera	3.93	4.63	84.9%
Google Workspace	2.04	2.42	84.3%
Zoom Marketplace	6.20	7.90	78.5%

the potential to leak private information owing to interdependent privacy. On average, each app requests 11.21 permissions, out of which 4.4 are IDP or PIDP.

Mozilla and Opera extensions, despite their similar architecture, differ significantly in terms of PIDP warning types (0.83 vs. 3.93 on average) and total warning types (0.85 vs. 4.63) displayed. Note that, although here we observe warning instead of permissions, the difference holds, as warning-permission mappings are alike on both platforms. One reason could be that more Mozilla extensions make use of the `active_tab` permission (which does not generate a warning) instead of `_url` type permissions[20]. Some other permissions also do not generate warnings, therefore the total number of permissions requested in Table 7 is underestimated, while the proportion in the last column is overestimated for browser extensions.

Google Workspace is dedicated to collaborative enterprise features with a lot of PIDP permissions therefore we expected a high proportion of those requested by apps. Indeed, 85% of total permissions requested (2.04 out of 2.42) are IDP or PIDP. We also observed that a majority of permissions are requested only by a few apps. This can be explained by the relatively low number of available apps and the fact that permissions are very specific (especially compared to browser extensions), e.g., "View and manage your Google Slides presentations" instead of "View and manage your documents".

There are a total of $2,433$ apps on Zoom Marketplace, belonging to 32 different categories. On average, each app has 2.54 vNIDP (view NIDP) permissions, 2.7 vIDP/vPIDP permissions, 0.25 mNIDP (manage NIDP) permissions, and 2.99 mIDP/mPIDP permissions. From a statistical point of view, the number of IDP/PIDP permissions is higher, especially among manage permissions. It is generally believed that the permission level of "app can manage" is higher than that of "app can view", and the risk is also greater. Based on the results above, we can also answer *RQ2* affirmatively: *actual apps do request IDP/PIDP permissions enabling collateral information collection on all studied platforms.* While new platform versions and newly released apps might change the absolute number of IDP/PIDP permissions per app, their magnitude is unlikely to change (pending no complete re-design of the respective permission structures).

[20] `https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/manifest.json/permissions#activetab_permission`

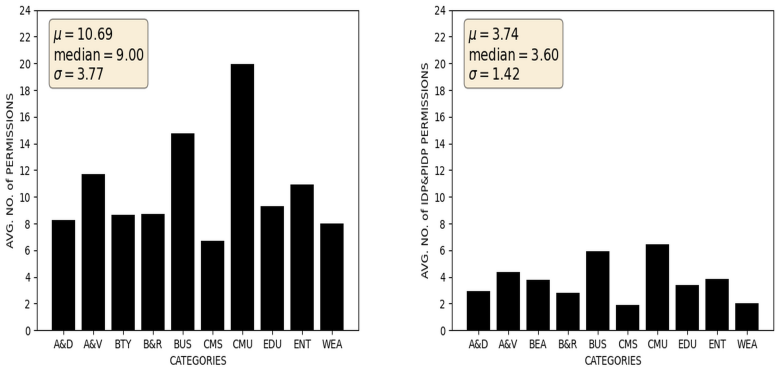

Fig. 2. Average number of permissions per app with different categories; Average number of IDP/PIDP permissions per app with different categories. Art&Design, Auto&Vehicles, Beauty, Books&Reference, Business, Comics, Communication, Education, Entertainment, Weather

4.3 Risk Signals

Users can obtain limited information when deciding upon installing third-party apps, such as category, number of users, user ratings, and permission types. Taking the Google Play store as an example, here we investigate whether the user can interpret these pieces of information as risk signals towards interdependent privacy. Previous studies found that neither popularity (number of users), nor community ratings (stars) are good indicators for privacy-conscious app behavior [13]. We also found evidence supporting this hypothesis. In fact, community ratings show a weak positive correlation with both the number of total permissions and the number of IDP/PIDP permissions requested: favorable ratings are mostly based on advanced functionality requiring more permissions.

The only promising indicator for an app to enable collateral information collection was its category. In order to demonstrate this, we selected $2,043$ apps from the Google Play dataset randomly, with ≈ 200 samples in each of the 10 major categories. The average number of total permissions (left) and IDP/PIDP permissions (right) can be seen in Figure 2.

The number of permissions varies greatly across categories. Apps belonging to "Business" and "Communication" request an average of 14.74, 19.92 permissions, while "Art&Design" and "Comics" only have 8.26, 6.71. A reasonable explanation for this result is that communications and business apps have more advanced features requiring more permissions. Interestingly, the same result holds for IDP/PIDP permissions. Categories with a high number of total permissions have a high number of IDP/PIDP permissions, and vice versa. The reasoning above can explain this result partially, but we argue that it is a characteristic of communication/business apps to involve more collaboration and multi-party interaction, a main theme behind permissions invoking interdependent privacy.

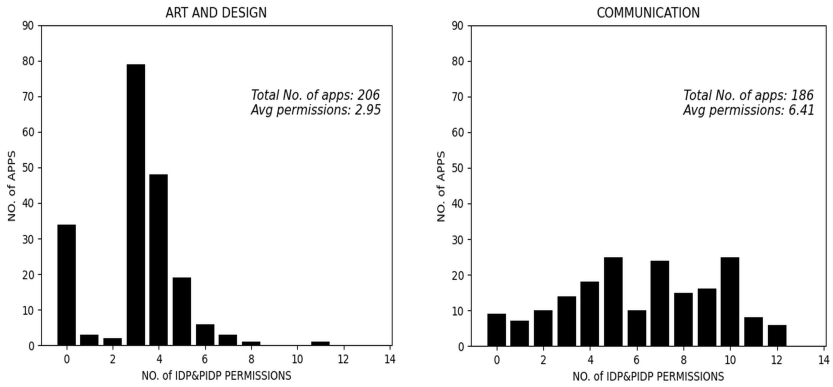

Fig. 3. Number of apps with different number of IDP/PIDP permissions in ART&DESIGN; Number of apps with different number of IDP/PIDP permissions in COMMUNICATION.

To illustrate this observation, we turn to the distribution of the number of IDP/PIDP permissions across all apps in a given category. Figure 3 shows the histogram for this metric for the categories "Art&Design" (left) and "Communication" (right). The difference between the two plots is striking: both the average number of IDP/PIDP permissions (2.95 vs. 6.41) and the shape of the histograms (top-heavy vs. normal-like) are very different. These patterns are mostly consistent for categories with a low and high number of permissions, respectively. This corroborates our previous observation, as more interactive/collaborative categories have more apps requesting a large number of IDP/PIDP permissions.

Note that our observations on risk signals are Android-specific; more narrow-focused app platforms are unlikely to have the same risky categories, as the range of apps offered is constrained.

5 Mitigation: considerations, objectives and design principles

5.1 Avoidance, transparency and control

In Section 3 and 4, we observed that i) all observed platforms offer permissions potentially invoking interdependent privacy, and ii) real apps do request a number of these permissions. Users are not particularly aware of interdependent privacy risks [2], and app platforms neither i) do a good job of informing the installing user and other persons affected by this issue, nor ii) offer control levers to influence such sharing. Therefore, it is up to the individual privacy awareness of installing users (acting as "amateur data controllers" [2]) and blind luck, that none of these platforms will experience its own Cambridge Analytica moment. Naturally, these options are neither satisfactory nor systemic; in the following, we discuss potential mitigation mechanisms, promoting risk avoidance, transparency, and control.

Risk avoidance. A visceral response by app platforms to avoid exposed interdependent privacy risks could be to banish (most) IDP/PIDP permissions from their API. In fact, this is exactly what Facebook did in 2018 in response to the Cambridge Analytica scandal: it gutted its API for third-party apps and introduced strict manual app review (hiring thousands of new employees)[21]. As evidenced by the declining popularity of Facebook apps, this might not be the most efficient way to deal with such risks. Indeed, the strong two-sided network effects characterizing app platforms require catering for both users and developers [25].

Transparency. Inspired by the GDPR and defined eloquently by Kamleitner's 3R insight framework [21], the sharing party (i.e., the amateur controller) can take three steps to reduce interdependent privacy risks: realize that there is a data transfer, recognize others' rights and respect others' rights. It is clear that transparency-enhancing technologies can facilitate the first two steps. A potential way to make the sharer aware of interdependent privacy is to add a special warning sign to the already existing permission notification dialogues [26]. Such a solution has to be platform-specific and needs the cooperation of the platform owner. If such cooperation is unlikely, a dedicated interdependent privacy dashboard app can be implemented in the manner of proposed dashboard designs for Facebook apps [2]. Note that an exact public mapping of API-level permissions to user warnings could also improve awareness (especially for browser extensions).

Following the opinion of the Article 29 Working Party and the subsequent recommendations of Privacy International[22] in the TrueCaller case, affected data subjects (i.e., "others") should/could also be notified by the app developer using SMS, using the very data it acquired unlawfully (i.e., contact list). Such notification, however, is not a general possibility: it depends on the platform and the actual data collected.

Control. There are some privacy best practices that, when adhered to, would improve the situation on the developer and the platform owner side. These include requesting the exact minimum privileges an app needs (developer) and introducing well-defined, fine-grain permissions to enable asking for the minimum privilege (platform owner, especially for browser extensions).

Best practices aside, there is potential for interdependent privacy-specific solutions that can enable better control of personal information both for affected users, privacy-conscious apps, and platforms. Notifying affected data subjects and asking for their consent can be feasible for i) specific data types (e.g., contact lists) or closed platforms such as Facebook (where all data are connected to other users of Facebook). In cases where a certain data object is clearly connected to multiple natural persons (e.g., photos, messages, collaborative documents, calls),

[21] `https://about.fb.com/news/2018/04/restricting-data-access/`
[22] `https://privacyinternational.org/node/2997`

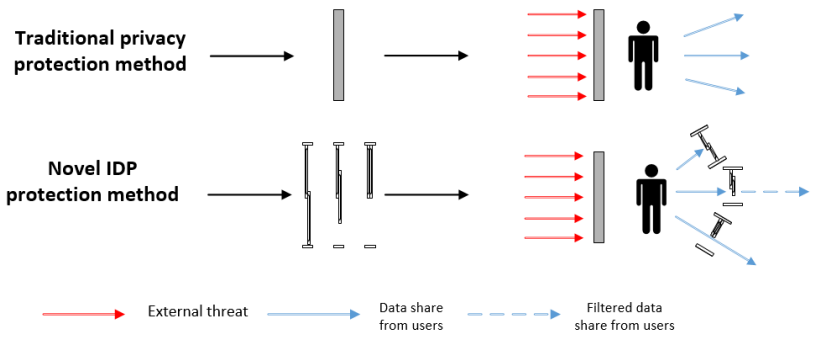

Fig. 4. Traditional privacy protection methods vs IDP protection methods.

sharing mechanisms tailored to multi-party data may be utilized [27, 3]. Whether these can be incorporated efficiently into a third-party app platform remains to be seen. Another way to go is to combine permissions with enforceable policies (in the manner of [28] but regarding privacy) and control the information flow in run-time [29] (not between components, but among platform, developer, users and others affected). An interesting restriction would be to keep data acquired through IDP/PIDP permissions locally on the user device, enabling computation (if needed for full app functionality) but restricting data transfer. There are many challenges for such a solution, starting with non-structured data that is hard to label as "multi-party".

Indeed, we can make a case for interdependent privacy being inherently present in current app platforms. A radical solution to mitigate this situation would be completely redesigning the currently widespread permission-based access for app platforms and trying different alternatives.

5.2 System objectives

Traditional privacy protection techniques in the third-party application domain predominantly focus on fine-tuning permissions and implementing usable privacy dashboards for users. In contrast, our emphasis regarding the mitigation of IDP issues lies in implementing an additional layer of protection during the transmission of user data when users willingly share their own information but may unknowingly share others' personal data. This ensures enhanced privacy even in instances of voluntary data sharing; see Figure 4.

Designing solutions to IDP problems faces multiple challenges. First, contemporary third-party application platforms have already achieved significant scale, making it impractical to fundamentally redesign or modify their internals solely to address IDP concerns. Moreover, the IDP issues faced by different platforms are similar but not identical. Second, in the absence of mandatory legal

and/or institutional mandates, both app platforms and users lack incentives and enthusiasm for addressing IDP-related matters.

Our main objectives are threefold. First, the system should not be tethered to any particular platform; it should be universally applicable to present-day third-party application platforms. Second, in light of the current absence of binding laws and regulations pertaining to IDP, the solution should be viable under the current data protection climate, offering adequate incentives for both application providers and users to adopt. Last, under the premise of voluntary user adoption, the solution should strive to protect user IDP with high efficiency.

5.3 Design principles

Since no system currently exists to protect users against IDP threats on any third-party application platform, the new system should be built to adhere to specific rules to i) ensure its effectiveness and robustness and ii) minimize the likelihood of creating new privacy risks. For tractability reasons, the new system is not intended to tear down and rebuild all the permission-related mechanisms of the original platform but to enhance and complement it.

P1: Feedback to users. Although, in general, original permission-related warnings do state the types of data the app in question would obtain upon installation/usage, this information is usually not communicated efficiently to the user. Specifically with IDP, related risks are not even acknowledged by the platform owner or the app developer (e.g., transferring a contact list with others' contact information). Therefore, our mechanism should both i) acknowledge the IDP risk and ii) convey rich information to the user. Enriching the feedback about IDP risks helps users respect others' rights, enabling them to cater to their other-regarding preferences. As such, the IDP and PIDP classifications of the requested permissions, as well as related filtering events, should be clearly displayed in the new warnings.

P2: Configurability. Data under IDP risks can be complex and of different types. Furthermore, users have different sharing preferences and requirements regarding different apps and use cases. Therefore, the system should be able to process/filter IDP-relevant information in a fine-grained manner. Suppose that the sharing of IDP-relevant information is necessary (to provide a complex functionality or better user experience. Yet, Alice may want to share her friend's location with an app but not her name, e-mail, or phone number (Scenario 1). Or, she may want to share part of her address book with a productivity app (e.g., company colleagues) but not everything (e.g., personal contacts) (Scenario 2). Scenario 1 shows that the processing of IDP-relevant information should be fine-grained, and the system should be able to classify and process different types of information. Additionally, Scenario 2 shows that the same type of information may have different scopes and user expectations. The system should be able to treat the same type of information as a range and extract the part the user wants. Moreover, users should be able to determine which of their own data could be collected through apps installed by others.

P3: Sufficient incentives and voluntary usage. The new system requires additional technical means to achieve IDP protection. This affects both third-party app providers and app users. For app providers, there is a financial incentive to avoid fines imposed by data protection authorities. Given that IDP is a gray area in the GDPR and other data protection regulations, it could be wise to adopt the proposed system. Nevertheless, we propose completely voluntary participation: this signals a privacy-conscious attitude towards users and potentially results in the enhanced reputation of the participating app providers. Note that the system should be capable of functioning without the active participation of users with a sub-optimal default configuration. However, users' input is vital for the system to reach its potential. Users have a dual incentive: they can satisfy both their self-regarding and other-regarding privacy preferences by enabling protection against IDP issues for themselves and others, respectively.
P4: Usability. It is essential that the IDP mitigation mechanism i) is easy to use (both for app providers and users), and ii) does not impede the users' quality of experience regarding normal app usage. First, providing a mechanism that can be used on any app platform is a significant step in the right direction: this way, both app providers and users can use the same one-stop-shop IDP mitigation service across all their apps and platforms. Second, a simple interface for both app providers (e.g., a standard REST API) and users (e.g., a simple web interface for setting their preferences) facilitates easy usage. Furthermore, it is important that the service does not impose a large overhead with respect to delay to preserve the normal user experience.
P5: Incremental deployability. We have already stated that we do not intend to redesign the permission systems of (popular) third-party application platforms. Although such an approach could attack the root of IDP issues, it is not practical: only platform owners (e.g., Google, Apple, Zoom, etc.) have the rights and power for such fundamental re-engineering. Instead, our IDP mitigation mechanism should operate on top of the existing app architecture, requiring zero change in the underlying mechanisms. Coupled with voluntary usage (P3), this ensures flexibility and gradual transition, where even a single app provider can opt-in to IDP protection, improving the privacy of its users instantly. This way, no critical mass (neither app providers nor users) is needed for the mitigation solution to start operating.

Note that our principles map well to the foundational principles and strategies of Privacy-by-Design [30–32]. The only notable absence is that of the principle of privacy embedded into the design; in fact, we aim to mitigate the lack of IDP-related protection mechanisms in third-party app platforms and the apps themselves, utilizing the exact strategies outlined in [31].

5.4 System design

We propose IDPFilter, an API designed to mitigate the risks associated with IDP in third-party applications. Drawing inspiration from established filtering methods, such as those evident in our proof-of-concept Flask-driven application (see Section 6), this API endeavors to provide a robust layer of protection (see

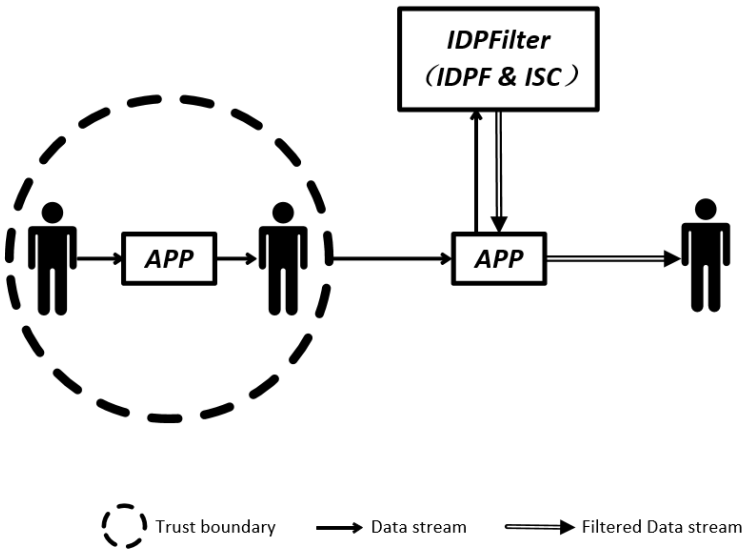

Fig. 5. When sending information outside the trust boundary, IDP information will be filtered

Figure 5. IDPFilter dynamically filters and masks sensitive information based on user preferences and predefined categories, thereby restricting inadvertent interdependent data leaks (see Figure 6). By marrying user-defined customization with system-driven presets, our solution encapsulates a forward-thinking approach to IDP. The crux of our system lies in its ability to improve on automated interdependent privacy safeguards, ensuring more efficient, customized protection for interdependent data and a trustworthy environment for app users.

The system consists of 4 main entities: users, applications, Interdependent Privacy Filters (IDPF), and Interdependent Privacy Settings Collector (ISC). IDPF will detect the content uploaded by the user, and if the data is IDP-related and matches the filter settings made by the user, IDPF will filter it. ISC collects IDP filter settings from the user. The default setting is to opt-in to the filter service. Of course, users have different needs in different scenarios, and ISC gives fine-grained choices, which means users can define how the filter actually works. IDPF and ISC together form IDPFilter.

5.5 IDPFilter: mitigation scheme

Our IDP mitigation scheme comprises three distinct interactions with regard to a given pair (app, user) during IDPFilter's operation (see Fig. 8):

1. **Filter setup.** Initially, the IDP service provides a standard API to all interested third-party applications, encompassing various filtering schemes under

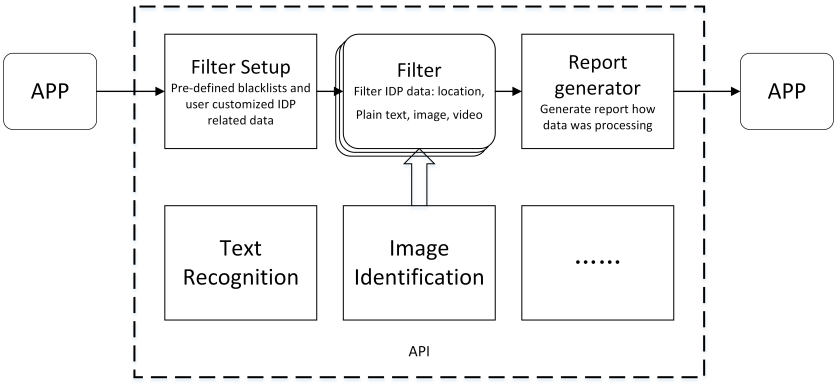

Fig. 6. Overview of the IDPFilter API framework

different constraints. Third-party application developers can select and integrate the services that best align with their requirements. The Filter Setup can be done at any time the users use the service.

2. **Granting permissions and customization.** When users opt to install a third-party application, the app has to request a set of permissions related to IDPFilter. These permissions include i) whether the user allows the IDPFilter service to filter the information they share with the app in the future and ii) the extent of information filtering, e.g., *whether the user permits others to share her information*; if not or only partially, the user can establish her own blacklist (whitelist), ensuring that the information referred to in the list is (not) filtered when other IDPFilter users share her information via the app in question. Note that the user only has to compile a blacklist (whitelist) once, and it is valid for all data transmission by the single app in question flowing through IDPFilter. User-compiled lists are further elaborated in Section 6.1.
3. **Data Flow.** Upon completing the first two stages, users share information through the third-party application, which utilizes the IDPFilter service. Based on the granted permissions and the chosen filtering scheme, the information is filtered before being shared with other users or third parties[23] as in Figure 5.

6 Proof-of-Concept: IDPTextFilter API

Users share multi-modal data via third-party applications, including text, voice, images, and videos. Technologies for image/voice/video recognition exist that could enable us to flag and reduce IDP-related leaks in all information modalities above. Nevertheless, in this section, we focus on basic textual data to illustrate the mechanism of mitigating IDP issues.

[23] third parties are stakeholders outside of users, app provider, and IDPFilter

The reasons why we chose to focus on text filtering over other forms of data are the following:

1. **Ubiquity and relevance.** Textual data is the most common type of data people share and interact with online. From social media posts to emails to articles, text is everywhere. Hence, text is indeed a relevant data modality for our proof-of-concept (PoC).
2. **Simplicity and speed.** Text processing is generally simpler and faster than processing images or video. It requires less computational power and storage, which has enabled us to focus on demonstrating IDPFilter in action and avoid making processing cost and efficiency our primary concern.
3. **Maturity of technology.** Natural Language Processing (NLP) has matured significantly over the past decades. We have advanced tools and libraries at our disposal that can be used directly to process text based on natural languages effectively and without developing algorithms from scratch. This has allowed for a streamlined development of our PoC.

IDPFilter can be deployed as a web service and hosted by us or any trusted providers. Following the design principles P1-P5 (see Section 5.3), we provide a series of interfaces available to *any* third-party application; this ensures that IDP protection is available and hassle-free so that app developers can use it easily as opposed to coding their own IDP protection functionality. The app needs to call the API to realize the function of IDP text filtering. If app developers integrate the IDPFilter API into their app, then users can both i) customize and ii) enjoy IDPFilter's features. From the users' perspective, APIs are presented as a (graphical) user interface integrated into the app. Therefore, users do not have to have technical knowledge in order to be protected by IDPFilter.

6.1 IDPTextFilter: structure

Our PoC is comprised of the User Interface, the Privacy Settings Collector, the Text Filter, and three types of user-defined lists. Note that the User Interface we developed serves only demonstration purposes; it is not strictly a part of the IDP(Text)Filter. In a real-world deployment, the User Interface part is implemented *within* the app.

User Interface (UI). The user interface provides an intuitive and seamless gateway for individuals to navigate and adjust their interdependent privacy settings (Principle P4: usability). Tailored to be user-friendly, this front-end component ensures that users have direct access and control over their personal data preferences.

Text Filter (TF). The Text Filter is responsible for dynamically applying the privacy settings and matching filter expressions, decided by the user, onto data that is being accessed by an app. By referencing parameters from the Privacy Settings Collector, TA ascertains which information can be shared and which should be withheld. TF is also responsible for generating filtering reports that form the basis of user feedback.

Interdependent Privacy Settings Collector (ISC). The Privacy Setting Collector plays the crucial role of gathering, storing, and updating the interdependent privacy settings of the users. To implement the core functionality of allowing users to choose which data to filter, the ISC design hinges on three foundational lists: the Self-Regarding Blacklist, the Other-Regarding Blacklist, and the Self-Regarding Whitelist. When integrated into the PoC, these lists ensure configurability and proper user incentives, resonating with Principles P2 and P3.

Self-Regarding Blacklist (SRB). A dynamic and customizable list where a user (User 1) designates private attributes that the app handles, but she does not want them to be shared via the app by any other user. As a data flow *from another user using any app integrating IDPFilter containing information on User 1* passes through TA, this list acts as a primary checkpoint in flagging and removing any matches from getting to the app, other users, and third parties. There is a single SRB for every (user, app) pair. Note that entries of the SRB are i) limited to data handled by the app in question and ii) verified by the app. For example, in the case of the user's e-mail address, the app requests confirmation from the user and then verifies whether the user entered the same e-mail address to the SRB.

Other-Regarding Blacklist (ORB). Recognizing the intertwined nature of digital privacy, the ORB is dedicated to safeguarding the privacy of other individuals related to the user creating the list (e.g., friends, family, or simply other users). The ORB enables users who are willing to protect others' privacy to achieve this effectively. When a user enters a sensitive word that might be related to others' privacy into its ORB, the text sent by this user (only this user) using the respective app will be filtered accordingly. There is a single ORB for every (user, app) pair.

Self-Regarding Whitelist (SRW). Serving as a counterpoint to the SRB, this is where users earmark data segments they deem shareable *by other users of the same app*. The SRW ensures the smoothness of information transmission when users agree to others sharing their information. There is a single SRW for every (user, app) pair.

By allowing users to define their own blacklists and whitelists, our prototype provides granular control over data sharing. As the user inputs data, the API checks it against these lists to filter out any sensitive or non-consented information. This reduces the potential risks associated with interdependent privacy in third-party applications.

6.2 IDPTextFilter: implementation

We have developed the PoC using Python[24]. We utilized the Flask framework[25] for development, accompanied by an HTML interface for displaying the privacy

[24] Code available at `https://github.com/shuai20/IDP_Filter`

[25] `https://flask.palletsprojects.com/en/3.0.x/`

dashboard. We utilized the FlashText algorithm, which performs well when keywords are complete and exhibits linear complexity in relation to the length of the search text. This makes it particularly useful for many keywords, as all can be matched simultaneously in a single pass over the input string [33]. Each user can register to select different text filtering schemes and save their preferences. The TA itself offers a sensitive vocabulary database, categorized into names, links, countries, disease names, and street names. Users may opt for more stringent filtering schemes, such as filtering all numerals. Additionally, users can establish their own sensitive word blacklist, SRB and ORB.

Vocabulary in the SRB is added to a public blacklist, effective for all registered users, ensuring terms from the SRB are filtered during IDPTextFilter usage. On the other hand, ORB vocabulary is only effective for the individual user: only this user will have her own ORB words filtered when using the service. In addition, users can create a sensitive word whitelist SRW and incorporate it into their filtering scheme. SRW words are given the highest priority, ensuring their preservation regardless of the selected filtering scheme. The UI is a user-centric dashboard designed for both i) user input of interdependent privacy settings and configurations and ii) feedback on filtering to the user. Users are greeted with personalized messages at a glance, ensuring a tailored experience. Users can effortlessly update their lists of sensitive and non-sensitive words via an intuitive mechanism. Beyond personal data settings, the dashboard provides robust filtering options, allowing users to specify categories of data they wish to shield, such as names, numbers, links, etc. After processing, the dashboard showcases a clear representation of the filtered text and provides a concise report on the number of words filtered. Navigation is streamlined with easily accessible options for logging out or returning to the homepage. Overall, the dashboard seamlessly bridges user preferences with advanced privacy features, ensuring fine-grained data protection, see Figure 7.

6.3 IDPTextFilter: User operation

The operation of this prototype is mainly composed of three mechanisms: registration, text filter setup, and text filtering, as seen in Figure 8.

Registration. The prototype handles user registration and login based on a username and password pair. The password is not stored directly; the PoC rather transforms it using the Password-Based Key Derivation Function (PBKDF2) before storing it locally[26], implementing the current best practice in password storage. During user login, the user submits her username and password through the login form, and then the prototype calculates its PBKDF2 transform and matches it to the stored value in its SQLite database.

Filter setup and execution. We consider three users in this system: Alice, Bob, and Jack. Alice wants to share a piece of text information with Bob through

[26] using the `pbkdf2hmac()` function of the `hashlib` package in python

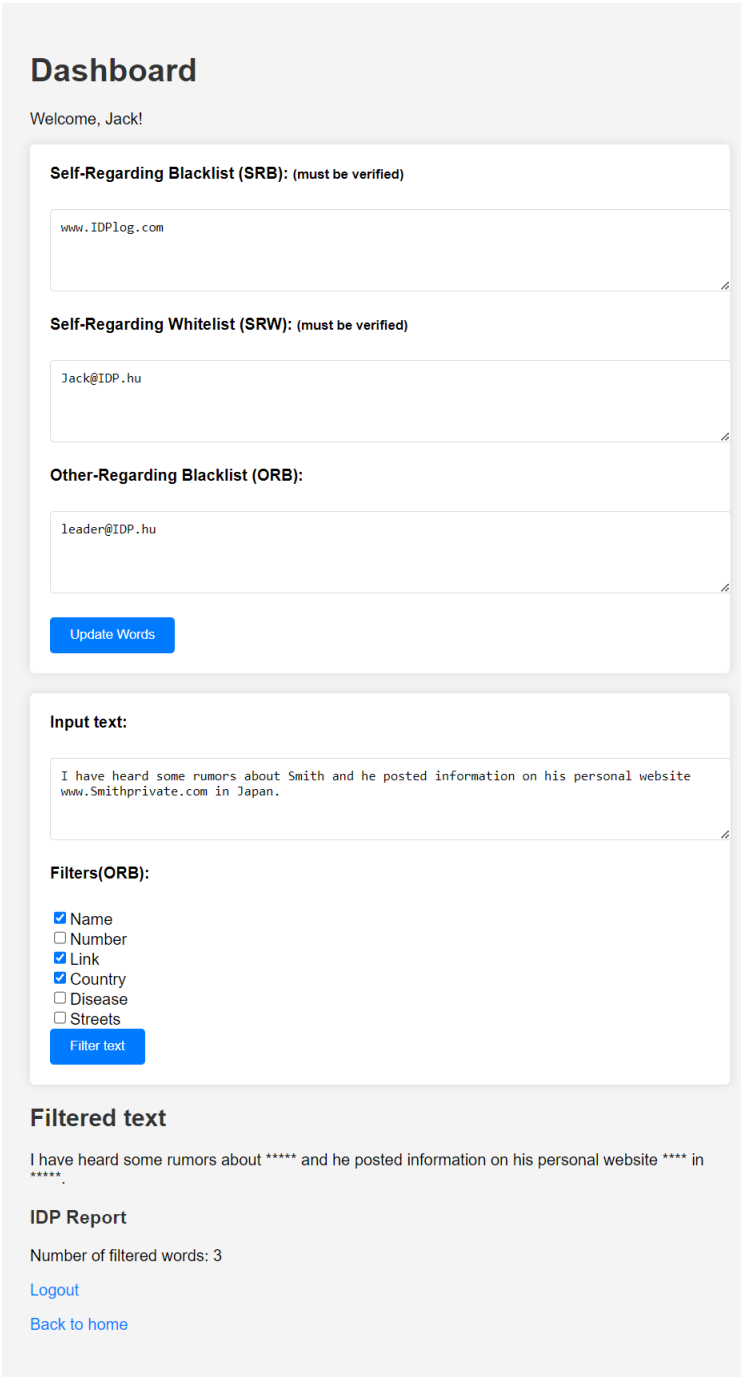

Fig. 7. PoC: User Interface (dashboard)

an app. The text contains content related to Jack's personal information. Alice, Bob, and Jack can all add content to the SRB. In the content Alice sends to Bob, the content related to Jack's SRB will be filtered out. Alice can add sensitive words in the ORB that may involve Jack's personal information to protect Jack's privacy proactively. Jack can add content to the SRW, which is equivalent to announcing that others are allowed to share certain information involving Jack himself. SRB, SRW, and ORB constitute each user's text processing preferences. Users can modify them at any time. IDPFilter saves the user's current settings to ensure that users do not need to reconfigure each time.

6.4 IDPTextFilter: Initial Evaluation

We briefly evaluate the PoC from three aspects: privacy, filtering accuracy, and filtering efficiency.

Privacy. While it is not possible to quantify the IDP risk reduction of IDPTextFilter at this stage, we believe it is important to summarize its privacy-enhancing characteristics and provide a brief *qualitative* evaluation.

IDPTextFilter provides both transparency and the ability to control how one's private textual data is shared by others (i.e., interdependent privacy) in the context of third-party applications. In terms of risk, the increased transparency of data sharing/filtering (delivered in the form of detailed reports via the UI) enables users to estimate IDP risks more accurately. On the other hand, providing a fine-grained control mechanism (composed of blacklists/whitelists and a text-filtering component) enables users to effectively reduce both the likelihood and impact aspects of IDP risks. It is important to emphasize that IDPFilter caters both to individual users and the whole user base of an app by operationalizing both own and other-regarding privacy preferences.

Accuracy. Although implementing the perfect pattern-matching tool is out of the scope of this paper, to illustrate the feasibility of the PoC, we devised accuracy tests for IDPTextFilter. To be able to perform an initial performance evaluation, we set up a local test environment (AMD Ryzen 7 4800H CPU 2.90 GHz, 32GB RAM with Windows 10 Education 22H2). Compared to traditional pattern-matching tools, the main feature to test is the users having different requirements for IDP protection; even for an individual user, requirements might change every time he/she shares information. The solution should be able to deal with multiple requirements and respond to modification quickly. For instance, every time users share data with others, they might choose to filter different categories like surnames or URLs, customize the blacklists and whitelists, or both. To check whether the accuracy is satisfactory, we implemented three different filter methods: regular expression, KMP algorithm [34], and FlashText [35]. For further comparison of filter results, we also implemented another Named Entity Recognition (NER) script by using the spacy library in Python [27]. Spacy NER has been proven to be an accurate and efficient tool to recognize the named entities in the text; in the following, we show the comparison of filtering results

[27] `https://pypi.org/project/spacy/`

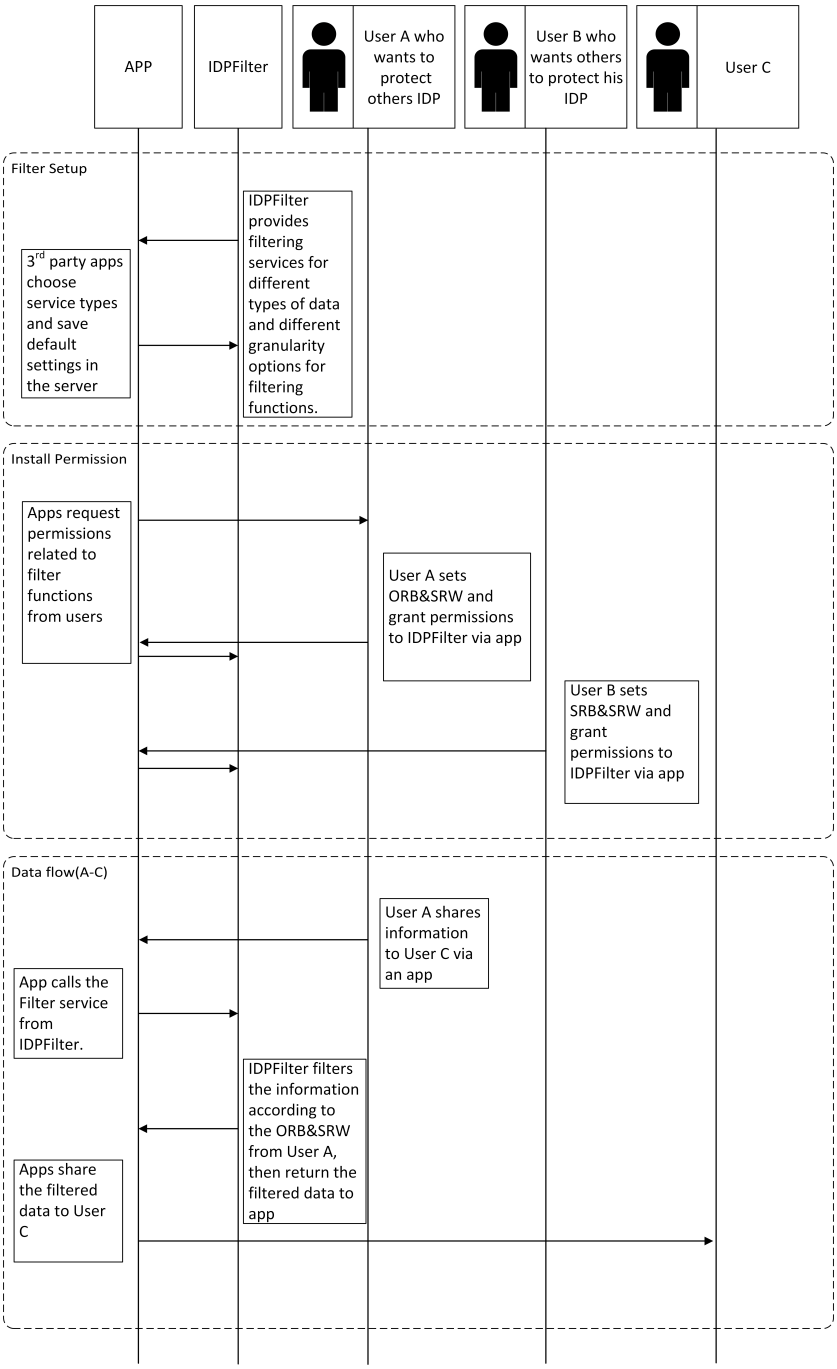

Fig. 8. PoC data flow

Table 8. Number of words filtered in 10 sentence sets: IDPFilter (800 most common surnames) vs. NER (PERSON)

	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
IDPFilter(Regex)	1455	1394	1432	1424	1390	1241	1394	1360	1304	1364
IDPFilter(KMP)	1455	1394	1432	1424	1390	1241	1394	1360	1304	1364
IDPFilter(FlashText)	1455	1394	1432	1424	1390	1241	1394	1360	1304	1364
Spacy NER	2255	2097	2272	2258	2227	2068	2161	2195	2044	2223

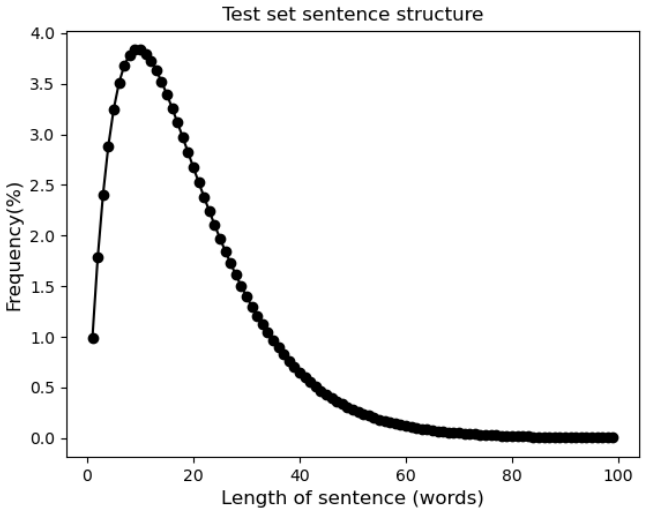

Fig. 9. Empirical PDF of sentence length (words) in each test set

made by IDPTextFilter and Spacy NER. Spacy NER now could recognize limited types of entities such as PERSON, ORG, or MONEY; therefore, it cannot fully implement the filtering requirements of IDPFilter (e.g., a user would like to filter three street names in Budapest), but it can provide an industry-grade baseline for comparing the results when the same sentences are processed with the same IDP-related words. To construct the test sets, we picked 10×10^5 random sentences from the Brown corpus[28]. To make these sentences realistic, in each set, sentence lengths conform to the Zipf distribution ($f_{\text{exp}} = 1.1 \cdot L^1 \cdot 0.90^L$) [36] visualized in Figure 9.

Due to the inflexibility of NER, we chose to recognize (filter) PERSON entities and compare the result when IDPTextFilter filters the name category. We show the number of filtered words for the four different methods in Table 8 with the 800 most common English surnames as the IDP-related blacklist.

After checking the filtered sentences, we can draw two conclusions. First, IDPTextFilter accurately filters out all the words in the list. On the contrary, NER makes mistakes when recognizing people's names. Some names are not rec-

[28] `http://korpus.uib.no/icame/brown/bcm.html`

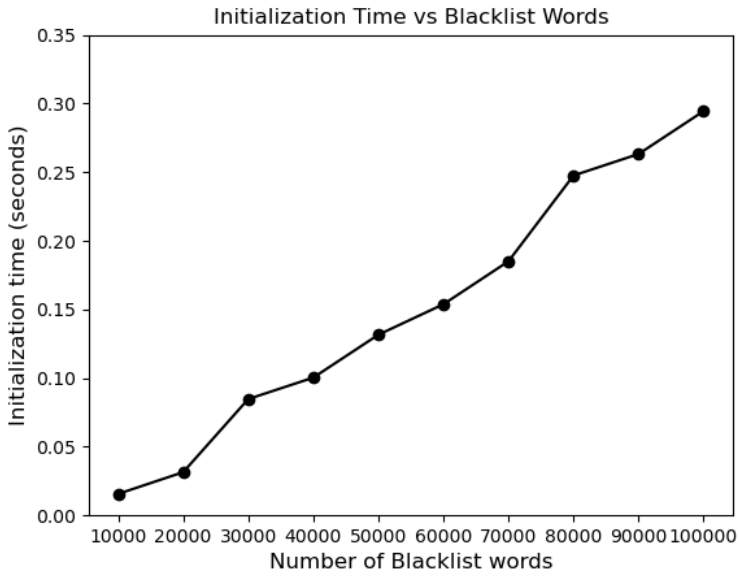

Fig. 10. Initialization time cost

ognized, and some other entities are mistakenly recognized as names. Second, NER is trained on a large amount of data; therefore, it performs well in approximate string matching and can identify fictitious names and nicknames, while IDPTextFilter cannot identify names outside the data set. In summary, IDPTextFilter can properly filter text based on user preferences if fine-grained blacklists are provided. Nevertheless, future work might consider adding NER-inspired approximate matching capabilities for (semi-)automatically creating sensible filtering rules; note that approximate expressions might create IDP issues themselves.

Efficiency. Considering that information sharing occurs all the time, mitigating the IDP issue requires an efficient solution that can promptly process a large amount of information. We first utilize the Knuth-Morris-Pratt (KMP) algorithm [34] to improve efficiency. To quantify efficiency, first, we show that when IDPFilter is provided with a large dataset to filter, the initialization time of the system is sufficiently short. We generated multiple blacklists (10,000 to 100,000 words per set) for measuring the initialization time. We obtained the ratio of the time required for pre-processing to the size of the blacklist text; see Figure 10. Then, we obtain the time costs when IDPTextFilter processes the test sets with IDP-related blacklists of sizes 100, 200, and 400. Figure 11 shows the average filtering time costs for 10,000-sentence test sets: $33.27s$, $67.28s$, and $127.34s$, respectively. The FlashText solution we ultimately adopted is far superior to other solutions in terms of efficiency. The time cost of filtering sentences under the same conditions is only about one-tenth of that of other solutions. Figure 12 shows the average filtering time costs for 10,000-sentence test sets: $3.45s$, $6.22s$,

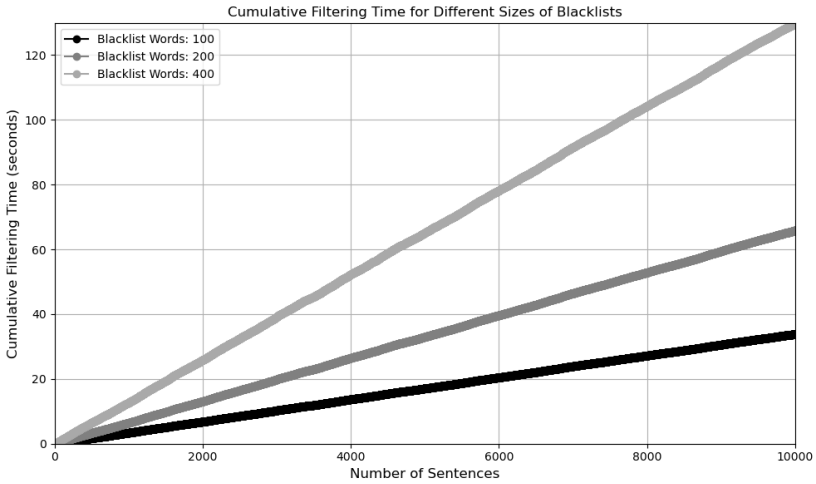

Fig. 11. Time cost (KMP) with different number of blacklist entries

and 11.95s, respectively. Note that IDPTextFilter is designed to reload the user's filter settings every time it is invoked for the user, so the filtering time values shown also include $10,000$ initializations. Note that we did not take further steps to improve the efficiency of IDPTextFilter as the prototype's performance was already sufficient to justify its promise regarding real-world deployment. Naturally, turning IDPTextFilter into a carrier-grade service would require more performance optimization.

7 Conclusion

Summary. In this paper, we carried out a comprehensive study on the interdependent privacy issues that inherently plague third-party applications in general. Specifically, our three research questions regarding RQ1) the pervasiveness of the privacy interdependence issues across platforms, RQ2) the collateral information collection of real-life apps, and RQ3) the possibility of designing and implementing a common mitigation mechanism were answered affirmatively. First, we provided data and reasoning about the proliferation of privacy interdependence on app platforms (RQ1) and in existing apps (RQ2). Then, most importantly, we designed IDPFilter, a platform-agnostic API that enables privacy-conscious app providers *and* end-users to filter out collateral information, i.e., data collected from certain app users but (also) implicating other natural persons as data subjects. We also provided a proof-of-concept implementation, IDPTextFilter, that realizes the proposed filtering logic on textual data (RQ3).

Benefits of IDPFilter. IDPFilter was constructed with five design principles in mind. First, it provides rich IDP-related information to the end-user to improve transparency and raise awareness. Second, the filtering mechanism is configurable to balance between privacy and app utility carefully. Third, IDPFilter can be used voluntarily, yet it aligns with the economic incentives of both app

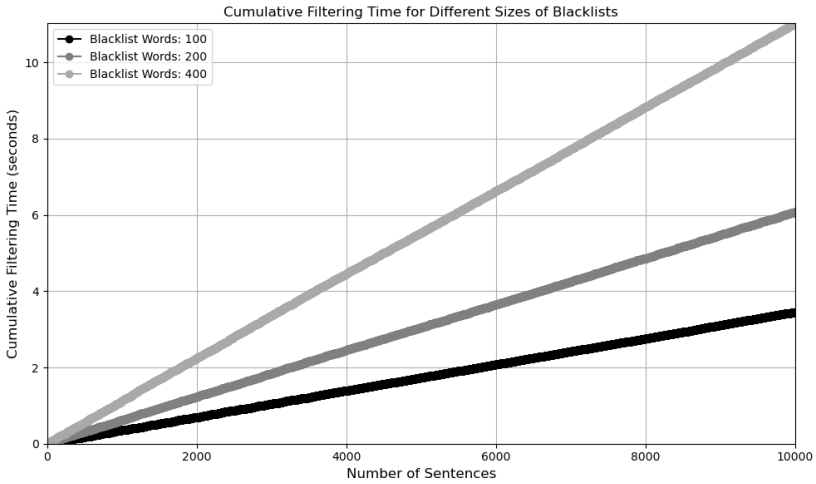

Fig. 12. Time cost (FlashText) with different number of blacklist entries

providers and end-users. Fourth, IDPFilter is platform-agnostic and easy to use for all stakeholders, providing a usable one-stop shop for IDP mitigation. Last, realizing that the complete re-design of third-party app permission systems is out of our reach, IDPFilter is compatible with the current mechanisms of app platforms, not requiring any major change regarding how apps and platforms operate.

Limitations. Naturally, our IDP mitigation mechanism has some limitations. First, from a trust perspective, the entity providing the IDPFilter service observes all collected data. This can be improved in different ways: i) using advanced cryptographic solutions, such as homomorphic/functional/searchable encryption that enables computation on encrypted data, or ii) by letting the app platforms themselves deploy the filtering service. Second, we provide neither a full-fledged implementation (realizing filtering on other, more complex data modalities) nor a comprehensive performance evaluation of our proof-of-concept prototype, IDPTextFilter. We believe that our contributions are sufficient to answer RQ3 affirmatively and leave the comprehensive treatment regarding both a full implementation and performance evaluation to future work. Third, recently brought into the foreground by the increasing popularity of applied large language models, we acknowledge that simple text filtering cannot remove all potential privacy implications owing to the correlated nature of natural languages: sensitive information can be revealed even if complete lines of text were removed [37]. All in all, IDPFilter cannot mitigate all IDP issues emerging in the third-party app context; nevertheless, it provides a first-of-its-kind solution that is able to improve the current situation considerably[38].

Acknowledgments

Project no. 138903 has been implemented with the support provided by the Ministry of Innovation and Technology from the NRDI Fund, financed under the FK_21 funding scheme.

References

1. Gergely Biczók and Pern Hui Chia. Interdependent privacy: Let me share your data. In Ahmad-Reza Sadeghi, editor, *Financial Cryptography and Data Security - 17th International Conference, FC 2013, Okinawa, Japan, April 1-5, 2013, Revised Selected Papers*, volume 7859 of *Lecture Notes in Computer Science*, pages 338–353. Springer, 2013.
2. Iraklis Symeonidis, Gergely Biczók, Fatemeh Shirazi, Cristina Pérez-Solà, Jessica Schroers, and Bart Preneel. Collateral damage of facebook third-party applications: a comprehensive study. *Comput. Secur.*, 77:179–208, 2018.
3. Alexandra-Mihaela Olteanu, Kévin Huguenin, Italo Dacosta, and Jean-Pierre Hubaux. Consensual and privacy-preserving sharing of multi-subject and interdependent data. In *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*. The Internet Society, 2018.
4. Anna Cinzia Squicciarini, Mohamed Shehab, and Federica Paci. Collective privacy management in social networks. In Juan Quemada, Gonzalo León, Yoëlle S. Maarek, and Wolfgang Nejdl, editors, *Proceedings of the 18th International Conference on World Wide Web, WWW 2009, Madrid, Spain, April 20-24, 2009*, pages 521–530. ACM, 2009.
5. Danah Boyd. Networked privacy. *Surveillance & society*, 10(3/4):348, 2012.
6. Stefania Gnesi, Ilaria Matteucci, Corrado Moiso, Paolo Mori, Marinella Petrocchi, and Michele Vescovi. My data, your data, our data: Managing privacy preferences in multiple subjects personal data. In Bart Preneel and Demosthenes Ikonomou, editors, *Privacy Technologies and Policy - Second Annual Privacy Forum, APF 2014, Athens, Greece, May 20-21, 2014. Proceedings*, volume 8450 of *Lecture Notes in Computer Science*, pages 154–171. Springer, 2014.
7. Mathias Humbert, Benjamin Trubert, and Kévin Huguenin. A survey on interdependent privacy. *ACM Comput. Surv.*, 52(6):122:1–122:40, 2020.
8. Alexandra-Mihaela Olteanu, Kévin Huguenin, Reza Shokri, Mathias Humbert, and Jean-Pierre Hubaux. Quantifying interdependent privacy risks with location data. *IEEE Trans. Mob. Comput.*, 16(3):829–842, 2017.
9. Mathias Humbert, Erman Ayday, Jean-Pierre Hubaux, and Amalio Telenti. Addressing the concerns of the lacks family: quantification of kin genomic privacy. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, pages 1141–1152. ACM, 2013.
10. Na Wang, Heng Xu, and Jens Grossklags. Third-party apps on facebook: privacy and the illusion of control. In *Proceedings of the 5th ACM symposium on computer human interaction for management of information technology*, pages 1–10, 2011.
11. Jennifer King, Airi Lampinen, and Alex Smolen. Privacy: Is there an app for that? In *Proceedings of the Seventh Symposium on Usable Privacy and Security*, pages 1–20, 2011.

12. Clint Gibler, Jonathan Crussell, Jeremy Erickson, and Hao Chen. Androidleaks: Automatically detecting potential privacy leaks in android applications on a large scale. In *International Conference on Trust and Trustworthy Computing*, pages 291–307. Springer, 2012.
13. Pern Hui Chia, Yusuke Yamamoto, and N. Asokan. Is this app safe?: a large scale study on application permissions and risk signals. In Alain Mille, Fabien Gandon, Jacques Misselis, Michael Rabinovich, and Steffen Staab, editors, *Proceedings of the 21st World Wide Web Conference 2012, WWW 2012, Lyon, France, April 16-20, 2012*, pages 311–320. ACM, 2012.
14. Steven Arzt, Siegfried Rasthofer, Christian Fritz, Eric Bodden, Alexandre Bartel, Jacques Klein, Yves Le Traon, Damien Octeau, and Patrick McDaniel. Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps. *Acm Sigplan Notices*, 49(6):259–269, 2014.
15. Joel Reardon, Álvaro Feal, Primal Wijesekera, Amit Elazari Bar On, Narseo Vallina-Rodriguez, and Serge Egelman. 50 ways to leak your data: An exploration of apps' circumvention of the android permissions system. In *28th {USENIX} Security Symposium ({USENIX} Security 19)*, pages 603–620, 2019.
16. Patrick Gage Kelley, Lorrie Faith Cranor, and Norman Sadeh. Privacy as part of the app decision-making process. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 3393–3402, 2013.
17. Yu Pu and Jens Grossklags. Towards a model on the factors influencing social app users' valuation of interdependent privacy. *Proc. Priv. Enhancing Technol.*, 2016(2):61–81, 2016.
18. Hamza Harkous and Karl Aberer. "if you can't beat them, join them": A usability approach to interdependent privacy in cloud apps. *CoRR*, abs/1702.08234, 2017.
19. Maria Karyda and Lilian Mitrou. Data breach notification: Issues and challenges for security management. In *10th Mediterranean Conference on Information Systems, MCIS 2016, Paphos, Cyprus, 4-6 September 2016*, page 60. University of Nicosia / AISeL, 2016.
20. Brendan Van Alsenoy. Regulating data protection: the allocation of responsibility and risk among actors involved in personal data processing. *PhD thesis*, 2016.
21. Bernadette Kamleitner and Vince Mitchell. Your data is my data: a framework for addressing interdependent privacy infringements. *Journal of Public Policy & Marketing*, 38(4):433–450, 2019.
22. Stefania Gnesi, Ilaria Matteucci, Corrado Moiso, Paolo Mori, Marinella Petrocchi, and Michele Vescovi. My data, your data, our data: Managing privacy preferences in multiple subjects personal data. In *Annual Privacy Forum*, pages 154–171. Springer, 2014.
23. Maximilian Marsch, Jens Grossklags, and Sameer Patil. Won't you think of others?: Interdependent privacy in smartphone app permissions. *Proc. ACM Hum.-Comput. Interact.*, 5(CSCW2), oct 2021.
24. Philip Raschke, Axel Küpper, Olha Drozd, and Sabrina Kirrane. Designing a gdpr-compliant and usable privacy dashboard. In Marit Hansen, Eleni Kosta, Igor Nai Fovino, and Simone Fischer-Hübner, editors, *Privacy and Identity Management. The Smart Revolution - 12th IFIP WG 9.2, 9.5, 9.6/11.7, 11.6/SIG 9.2.2 International Summer School, Ispra, Italy, September 4-8, 2017, Revised Selected Papers*, volume 526 of *IFIP Advances in Information and Communication Technology*, pages 221–236. Springer, 2017.
25. Geoffrey G Parker and Marshall W Van Alstyne. Two-sided network effects: A theory of information product design. *Management science*, 51(10):1494–1504, 2005.

26. Anjuli Franz and Alexander Benlian. Exploring interdependent privacy – empirical insights into users' protection of others' privacy on online platforms. *Electronic Markets*, 32(4):2293–2309, Dec 2022.
27. Jose M Such, Joel Porter, Sören Preibusch, and Adam Joinson. Photo privacy conflicts in social media: A large-scale empirical study. In *Proceedings of the 2017 CHI conference on human factors in computing systems*, pages 3821–3832, 2017.
28. Elli Fragkaki, Lujo Bauer, Limin Jia, and David Swasey. Modeling and enhancing android's permission system. In *European Symposium on Research in Computer Security*, pages 1–18. Springer, 2012.
29. Limin Jia, Jassim Aljuraidan, Elli Fragkaki, Lujo Bauer, Michael Stroucken, Kazuhide Fukushima, Shinsaku Kiyomoto, and Yutaka Miyake. Run-time enforcement of information-flow properties on android. In *European Symposium on Research in Computer Security*, pages 775–792. Springer, 2013.
30. Ann Cavoukian. Privacy by design: The 7 foundational principles. *Information and privacy commissioner of Ontario, Canada*, 5:12, 2009.
31. Jaap-Henk Hoepman. Privacy design strategies. In Nora Cuppens-Boulahia, Frédéric Cuppens, Sushil Jajodia, Anas Abou El Kalam, and Thierry Sans, editors, *ICT Systems Security and Privacy Protection*, pages 446–459, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.
32. Seda Gürses, Carmela Troncoso, and Claudia Diaz. Engineering privacy by design. *Computers, Privacy & Data Protection*, 14(3):25, 2011.
33. Vikash Singh. Replace or retrieve keywords in documents at scale. *CoRR*, abs/1711.00046, 2017.
34. Hossein Dehghani, Neerja Mhaskar, and W. F. Smyth. Practical KMP/BM style pattern-matching on indeterminate strings. *CoRR*, abs/2204.08331, 2022.
35. V. Singh. Replace or Retrieve Keywords In Documents at Scale. *ArXiv e-prints*, October 2017.
36. Bengt Sigurd, Mats Eeg-Olofsson, and Joost Van Weijer. Word length, sentence length and frequency–zipf revisited. *Studia linguistica*, 58(1):37–52, 2004.
37. Hannah Brown, Katherine Lee, Fatemehsadat Mireshghallah, Reza Shokri, and Florian Tramèr. What does it mean for a language model to preserve privacy? In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, pages 2280–2292, 2022.
38. Shuaishuai Liu, Barbara Herendi, and Gergely Biczók. Interdependent privacy issues are pervasive among third-party applications. In Joaquín García-Alfaro, Jose Luis Muñoz-Tapia, Guillermo Navarro-Arribas, and Miguel Soriano, editors, *Data Privacy Management, Cryptocurrencies and Blockchain Technology - ESORICS 2021 International Workshops, DPM 2021 and CBT 2021, Darmstadt, Germany, October 8, 2021, Revised Selected Papers*, volume 13140 of *Lecture Notes in Computer Science*, pages 70–86. Springer, 2021.