PrivacyFlash Pro: Automating Privacy Policy Generation for Mobile Apps

Sebastian Zimmeck, Rafael Goldstein and David Baraka
Department of Mathematics and Computer Science, Wesleyan University
{szimmeck, rgoldstein01, dbaraka}@wesleyan.edu

***Abstract*—Various privacy laws require mobile apps to have privacy policies. Questionnaire-based policy generators are intended to help developers with the task of policy creation. However, generated policies depend on the generators' designs as well as developers' abilities to correctly answer privacy questions on their apps. In this study we show that policies generated with popular policy generators are often not reflective of apps' privacy practices. We believe that policy generation can be improved by supplementing the questionnaire-based approach with code analysis. We design and implement PrivacyFlash Pro, a privacy policy generator for iOS apps that leverages static analysis. PrivacyFlash Pro identifies code signatures — composed of Plist permission strings, framework imports, class instantiations, authorization methods, and other evidence — that are mapped to privacy practices expressed in privacy policies. Resources from package managers are used to identify libraries.**

We tested PrivacyFlash Pro in a usability study with 40 iOS app developers and received promising results both in terms of reliably identifying apps' privacy practices as well as on its usability. We measured an F-1 score of 0.95 for identifying permission uses. 24 of 40 developers rated PrivacyFlash Pro with at least 9 points on a scale of 0 to 10 for a Net Promoter Score of 42.5. The mean System Usability Score of 83.4 is close to excellent. We provide PrivacyFlash Pro as an open source project to the iOS developer community. In principle, our approach is platform-agnostic and adaptable to the Android and web platforms as well. To increase privacy transparency and reduce compliance issues we make the case for privacy policies as software development artifacts. Privacy policy creation should become a native extension of the software development process and adhere to the mental model of software developers.

I. Introduction

Legislators around the world are enacting new privacy laws to increase privacy protection online. Recent lawmaking activities stateside include Vermont's Data Broker Regulation [Act 171 of 2018] and Nevada's right of consumers to request their information not being sold [SB 220]. Notably, California's enactment of the California Consumer Privacy Act [CCPA] is the most comprehensive online privacy law in the US to date. Many laws are based on notice and choice: users are notified of applicable privacy practices and given the choice to opt out; at least by not using a service. The instrument to convey notice is the privacy policy, which serves as a manifestation of the privacy practices an app developer is accountable for. App developers can be subject to a host of privacy requirements they have to disclose in their policies. They are also subject to contractual privacy obligations. For example, the Apple Developer Program License Agreement requires developers to provide a policy explaining how they collect, use, disclose, share, retain, and delete user and device data [9].

While app developers are often required to provide a privacy policy, many are actually not familiar with the privacy implications of their apps and the laws they have to comply with [14]. Especially, individual developers and smaller organizations with fewer employees and resources are often not able to devote time or money to privacy considerations and may need additional help with drafting privacy policies [14]. Consequently, there is often a substantial disconnect between how privacy practices are described in privacy policies and apps' actual behavior [86]. This disconnect can lead to compliance issues and is particularly prevalent for the use of permissions by the developer and third parties as well as the integration of third party libraries [86]. A recent study of 13,796 Android apps and their privacy policies found that up to 42.4% of apps either incorrectly disclose or omit disclosing their privacy-sensitive data flows [6].

Questionnaire-based policy generators promise a low-cost solution to the problem of writing legally compliant privacy policies. Such generators, available for web and mobile apps, prompt developers with a set of privacy-related questions on their apps and generate policies based on the supplied answers. An estimated 25% of Android app developers make use of such generators [66]. In addition to alleviating developers from the task of writing privacy policies, questionnaire-based generators may be advantageous from the users' perspective as well. The standardized language and format may make it more convenient to compare different policies. Generated policies are also easier to analyze automatically. However, questionnaire-based policy generators have fundamental limitations. They are necessarily reliant on the correctness of answers provided by the developers, which creates a risk of inaccurate policies due to wrong or missing answers. Design flaws in a generator may reflect in the policies as well.

We propose to combine questionnaire- and code-based policy generation. Source code can be understood as a semantics-rich documentation of an app's privacy practices [48]. Thus, app code, including code of integrated libraries, can serve as the starting point for a privacy policy that is a traceable and faithful representation of how an app behaves [85]. However, while code analysis helps aligning policy disclosures with actual app behavior, it still remains necessary to query the developer for some input. For example, for how long personal

Network and Distributed Systems Security (NDSS) Symposium 2021
21-25 February 2021, Virtual
ISBN 1-891562-66-5
https://dx.doi.org/10.14722/ndss.2021.24100
www.ndss-symposium.org

information is retained or whether an app is subject to the CCPA cannot be solely derived from an app's codebase. Combining static code analysis and a questionnaire-based wizard with templates, we implemented PrivacyFlash Pro, a privacy policy generator for iOS apps written in Swift and integrated Swift and Objective-C libraries. We demonstrate our idea for iOS, but the same principles apply to Android, cross-platform frameworks (e.g., React Native), and web apps. We believe that especially indie developers, freelance developers, and startups would benefit the most from our work.

We believe that privacy policies should be recognized as software development artifacts [85]. The integration of legal requirements into the software development process enables compliance traceability, that is, the ability to link requirements originating from privacy laws to their corresponding software implementations. This link strengthens the synchronization between privacy-by-policy and privacy-by-architecture [23]. Just as for software licenses, an area where legal considerations became part of the software development process, privacy policies can be selected, created, and maintained by developers. In order for developers to perform this new task well it is crucial to design it based on their mental model. Certainly, policy generation will not eliminate the work of lawyers in all cases. For complex apps and apps with unusual privacy practices it will still be necessary to create individualized privacy policies. However, for the average app with standard permission uses and third party library integrations automating privacy policy generation holds the promise of increased privacy compliance and traceability. Thus, in this study, we are making the following contributions:

(1) We analyze the extent to which the use of popular questionnaire-based privacy policy generators helps iOS app developers to accurately disclose the privacy practices of their apps. Our results suggest that many apps behave differently than described and that the examined generators are inherently limited by their design and the exclusive reliance on their questionnaire-based approach. (§ III).

(2) We design and implement PrivacyFlash Pro, a privacy policy generator for iOS apps written in Swift. PrivacyFlash Pro combines questionnaire-based policy generation and standardized templates with automatic detection of privacy practices in app code via static analysis. It is available as an open source project.[1] (§ IV).

(3) We evaluate PrivacyFlash Pro in a usability study with 40 iOS app developers. The policies that the developers generated with PrivacyFlash Pro offer better coverage of the privacy practices of their apps than their previous policies. The developers also reported high levels of product usability and satisfaction. (§ V).

II. Related Work

Online privacy is fundamentally based on the principle of notice and choice.

[1]PrivacyFlash Pro GitHub repository, https://github.com/privacy-tech-lab/privacyflash-pro, accessed: January 7, 2021.

A. Notice and Choice

Users should be notified of applicable privacy practices and given the choice to opt out. However, oftentimes, it is not transparent to users how mobile and web apps use and disclose the data they collect. Especially, mobile app users are confronted with trade-offs between privacy and usability due to the constraints of mobile devices when setting their notification preferences [74]. Notifications during app use, instead of before app use, have proven to be effective [15], however, may also lead to notification fatigue. Reducing the number of privacy decisions users have to make [21] and leveraging social interaction for privacy features [4] could help prevent such [73]. Using comic-based privacy policies [68], flyers [36], or paraphrased terms of use [75] are creative new ways of engaging users' interest.

B. Privacy Policy Generators

From a developer's perspective, writing a privacy policy that correctly reflects an app's privacy practices and keeping the policy up-to-date as the app evolves over time can be a challenging task. Various solutions, most of which are commercial, generate policies based on questionnaires filled by the developer [7], [32], [35], [41], [53], [69], [70]. PAGE, a plugin for the Eclipse IDE, can be used to create questionnaire-based policies during the development process [59]. However, purely questionnaire-based generators can lead to inaccurate representations of an app's privacy practices if the questions are not answered, accurately, timely, and completely. We aim to mitigate these shortcomings by leveraging code analysis.

The closest work to ours, Polidroid-AS, is an Android Studio plugin that combines a simple privacy questionnaire with code analysis functionality [63]. However, different from Polidroid-AS, PrivacyFlash Pro has the goal of creating comprehensive and legally compliant policies for iOS apps beyond the text snippets of Polidroid-AS. PrivacyFlash Pro intends to create such policies by covering provisions of the FTC Act, Children's Online Privacy Protection Act [COPPA], California Online Privacy Protection Act [CalOPPA], CCPA, and General Data Protection Regulation [GDPR]. While no performance or usability evaluations are available for Polidroid-AS, we provide those in a usability study with iOS app developers, most of which are full-time professionals (§ V).

AutoPPG is another closely related work [80]. Similar to Polidroid-AS, AutoPPG extracts code from Android apps to create short text snippets of app behavior. A corpus of policies collected in the wild is used for generating snippets of the form `subject verb object [condition]`, which, however, entails the risk of importing non-compliant language into the generated snippets as many policies are not compliant with the law [5]. Different from AutoPPG, PrivacyFlash Pro aims to create fully legally compliant policies by mapping app analysis results and questionnaire answers to standardized legal templates. In addition, our study goes beyond both AutoPPG and Polidroid-AS with a developer usability study and a survey of questionnaire-based generators. PrivacyInformer, another related tool, also creates text snippets, though, only for apps created with the MIT App Inventor [46].

Code analysis is also used to generate templates with app privacy settings [20] and security descriptions [82]. Such

templates and descriptions can be helpful for users to understand and adjust the privacy and security settings of an app. Generally, profiling app behavior enables uncovering privacy practices [58]. Though, any templates, descriptions, and profiles are not directly usable as privacy policies.

C. Machine-readable Privacy Policies

While natural language privacy policies emerged as the standard for disclosing privacy practices, machine-readable policies would make it easier for browsers and other user agents to process and act upon on what is disclosed in policies. The Platform for Privacy Preferences (P3P) is one of the major works taking steps towards this direction [24], [25]. Privacy policy languages [78], [81] that allow a developer to specify data flows and enforce constraints are an interesting area of automating policy processing as well. However, as none of the suggested approaches received significant industry adoption, the natural language policy remains the default format and its standardization via policy generators seems a more viable avenue for making policies machine-readable rather than using specialized policy languages.

D. Static and Dynamic Code Analysis

In order to identify the privacy practices an app is performing PrivacyFlash Pro relies on signature-based static analysis. In the Android ecosystem Stowaway [30] paved the way for the static analysis of permissions. FlowDroid [11] is still the state of the art and was extended in various studies, e.g., for purposes of inter-component data-flow analysis to detect privacy leaks between app components [37]. It was shown that many apps are circumventing permissions [56]. For iOS code analysis is much less explored. PiOS [28] is one of the few static analysis frameworks for iOS. Fortunately, it is not necessary for us to leverage reverse-engineering techniques as PrivacyFlash Pro is operating on apps' source code. Even in compiled libraries, APIs are visible in plaintext. As many apps include such libraries, a comprehensive privacy analysis needs to be able to reliably identify those [13].

E. Privacy Policy Analysis

Various studies examined the extent to which information can be extracted from privacy policies [43], [55], most notably, related to opt out mechanisms [60], purposes for app permission usages [12], and sections that are relevant under the GDPR [71]. While recently neural networks were used for this purposes [33], simple machine learning techniques are often sufficient due to the limited variation in policy language [84]. Especially, financial institutions' privacy notices are usually based on a standard template [26].

One analysis found that mobile financial services often do not disclose what types of personal information they collect and store [17]. Whether in finance or other domains, the disclosure of third party practices is rare. An analysis of over 200,000 website privacy policies revealed that third party data flows are disclosed in fewer than 15% of required cases [40]. While many policies have internal contradictions [5], recent lawmaking activity seems to have a positive effect. Polices generally became more specific in the wake of the enactment of the GDPR [42].

F. Privacy Compliance Analysis

Beyond policy analysis, compliance analysis seeks to identify discrepancies between privacy practices described (or omitted) in policies and actual code functionality. Apps are potentially non-compliant due to developers' use of app building frameworks that add unnecessary permissions and API invocations, reuse of privacy-sensitive functionality by developers in multiple apps, support of secondary undocumented app functionality, or use of third party libraries [77]. Policies can be in conflict with third party library policies [79]. Thus, the ability to correctly distinguish between first and third party functionality is crucial to identify compliance issues [6] and is accounted for in PrivacyFlash Pro. Based on linking policy phrases to privacy-sensitive API invocations numerous potential compliance issues were identified for a set of top Android apps [64]. Similar findings were confirmed in a large-scale analysis of about 1 million Android apps [86].

Privacy non-compliance even persists in sensitive domains. A set of 80 health and finance apps displayed 20 "strong" and 10 "weak" violations where their functionality did not align with their privacy policies [76]. To prevent violations of Institutional Review Board policies an app platform for enforcing such policies was proposed [83]. Ultimately, current privacy compliance analysis approaches of mobile apps are grounded in the notion that an app's codebase is a semantics-rich documentation carrying meaningful privacy-related information [48]. Thus, natural language processing can be applied to automatically locate program elements of interest and perform a learning-based program structure analysis to identify those structures that are indeed carrying sensitive content [48].

G. App Development Practices and Tools

Various development practices are impacting the privacy behavior of apps. Most notably, many developers are including code from Stack Overflow [65] or other crowdsourcing resources in their app code. An analysis of 1.3 million Android apps revealed that 15.4% contained security-related code snippets from Stack Overflow and 97.9% of these contained at least one code snippet that was not secure [31]. The official Android API documentation is perceived as difficult to use; informal resources appear more accessible, though, often lead to vulnerable code [2]. Boilerplate code from app generators includes well-known security issues, such as code injection vulnerabilities, however, due to their blackbox nature, developers are often unaware of these hidden problems [51]. Individual developers and smaller organizations are more likely to run into privacy and security issues, particularly, due to the opacity of third party ad and analytics libraries [14]. Developers often do not feel responsible for managing and addressing consumer risks from integrating such libraries [45].

Developer tools may reduce the barriers for app developers to implement privacy and security best practices [14]. To support Android developers in writing better code, FixDroid, an Android Studio plugin, highlights security- and privacy-related code problems, provides an explanation to developers, and suggests quickfix options [50]. Similarly, Coconut, another Android Studio plugin, highlights potential privacy problems and offers quickfixes [38]. The information provided to the

Policy Generator	CCPA		COPPA		GDPR	
	May'20	Jan'21	May'20	Jan'21	May'20	Jan'21
iubenda	14/18	14/18	2/4	2/4	8/8	8/8
Termly	**3/18**	**15/18**	1/4	1/4	8/8	8/8
TermsFeed et al.	**5/18**	**16/18**	2/4	2/4	8/8	8/8

TABLE I: Tallies of the generators' compliance with legal requirements. The App Privacy Policy Generator does not claim CCPA, GDPR, or COPPA compliance. The individual requirements are shown in Table II (CCPA) and in Appendix Tables A.7 (COPPA) and A.8 (GDPR).

developer in FixDroid and Coconut can be used as a starting point for writing the respective apps' privacy policies. The same is true for PrivacyStreams, an implementation of a functional programming model for evaluating the access and processing of personal information in Android apps [39]. PrivacyStreams provides an API on top of the Android API that highlights privacy-related code usage, although, it does not account for third party libraries.

III. Questionnaire-based Policy Generators

Questionnaire-based policy generators guide developers through a privacy questionnaire and are intended to generate a compliant privacy policy from the supplied answers.

A. Generator and Policy Selection

We set out to review six popular questionnaire-based generators for creating mobile app privacy policies: App Privacy Policy Generator [7], FreePrivacyPolicy [32], iubenda [35], PrivacyPolicies [53], Termly [69], and TermsFeed [70]. The market is quite small but increasing. We are only considering generators covering mobile apps.[2] During our evaluation we found that three generators — TermsFeed, FreePrivacyPolicy, and PrivacyPolicies (in the following, TermsFeed et al.) — are provided by the same entity and seem to generate identical policies. This relationship is not immediately obvious but was confirmed to us by the operator, who also explained that TermsFeed is their main offering. Neither of the generators we review has any code analysis functionality. In order to collect policies from the questionnaire-based generators we started with a Google search. For example, searching for `ios iubenda site:https://itunes.apple.com/` returned numerous search results for policies of iOS apps in Apple's App Store generated with iubenda [35]. We collected policies in the order they appeared in the search results given they were (1) in English,[3] (2) associated with a free iOS app on the US App Store, (3) linked from an app's App Store page, and (4) hosted on the generator site or appearing to be generated with the generator.

To ensure that the policies were in fact generated by the generators and not modified by the developers we are relying on the generators' hosting features. All generators but one offer the option to host the generated policies on their servers so they can update the policies in case of changes in the legal environment. The App Privacy Policy Generator is the only generator not offering hosting. However, policies from this generator include a reference disclosing that "[t]his privacy policy page was created at privacypolicytemplate.net and modified/generated by App Privacy Policy Generator." All policies we are examining from this generator contained such reference and do not appear modified. We also generated multiple policies from each generator on our own to verify the absence of modifications in the policies we collected. Using the generators ourselves let us evaluate how they work and observe what policies they generate for different questionnaire answers. Leaving aside the policies we created on our own, we collected a total of 95 policies from the App Privacy Policy Generator (20), iubenda (20), Termly (20), and TermsFeed et al. (35).

B. Privacy Policy Analysis

Generators must be designed so that developers are able to generate compliant privacy policies. In particular, they must be kept up to date with changing laws. All generators we examined, except for the App Privacy Policy Generator, advertised on their websites to generate policies compliant with the CCPA, COPPA, and GDPR. However, in our initial analysis in May 2020 we found that all generators had substantial shortcomings, for example, they failed to create CCPA-compliant policies. Table I shows tallies of generators' compliance with legal requirements in May 2020. It also shows tallies of our second analysis in January 2021 after we had contacted the generator operators.[4] In May 2020 policies generated with Termly were only compliant with 3 out of 18 CCPA requirements. Similarly, policies generated with TermsFeed et al. were only compliant with 5 out of 18 CCPA requirements. For example, neither policy generated with iubenda, Termly, or TermsFeed et al. provided a list of categories of personal information disclosed for business purpose in the preceding 12 months per CCPA §1798.130(a)(5)(C).

The non-compliance was independent of the answers provided in the generators' privacy questionnaires. It was the design of the generators that did not sufficiently accommodate the CCPA. Perhaps, the generator operators needed some time to adapt their questionnaires to the new law. The CCPA became effective fairly recently in January 1, 2020 after a rather quick lawmaking process. In our analysis of January 2021 we found that Termly and TermsFeed et al. had adapted their questionnaires and their policies are now compliant with CCPA §1798.130(a)(5)(C). To cover this provision a generator would need to be adapted because it goes beyond the traditional canon of what a privacy policy usually discloses. On the other hand, the higher levels of GDPR compliance in our first analysis in May 2020 could be based on the GDPR already being effective since 2018, after it was well publicized in 2016. The requirements of the GDPR are also more general than those of the CCPA and cover disclosures that are often traditionally included in privacy policies even if they are not GDPR-specific, for example, the purposes of the data processing per GDPR Art. 13(1)(c), 14(1)(c).

[2]Thus, for example, we are not reviewing the generators of e-commerce services 3dcart [1] and Shopify [62] as they are only covering websites.

[3]Some generators support generation of policies in multiple languages.

[4]We contacted all generator operators in June 2020 and notified them of our findings. We received a response from a representative at TermsFeed et al. and further explained our findings. The representative told us that TermsFeed et al. is in a process of updating their generator.

CCPA Requirement	iubenda May'20	iubenda Jan'21	Termly May'20	Termly Jan'21	TermsFeed et al. May'20	TermsFeed et al. Jan'21
Disclosure of right to request how personal information is collected, used, sold, disclosed for a business purpose, and shared [CCPA §1798.130(a)(5)(A), 1798.110(a), 1798.115(a), Regs §999.308(c)(1)(a)]	✓	✓	✗	✓	✗	✓
Disclosure of right to request deletion of personal information [CCPA §1798.105(b), 1798.130(a)(5)(A), Regs §999.308(c)(2)(a)]	✓	✓	✗	✓	✗	✓
Disclosure of whether personal information is sold and right to opt-out of sale [Regs §999.308(c)(3)(a), 999.308(c)(3)(b), 999.306]	✓	✓	✗	✓	✗	✓
Disclosure of right to not be discriminated against when requesting any rights [CCPA §1798.130(a)(5)(A), 1798.125(a), Regs §999.308(c)(4)(a)]	✗	✗	✗	✓	✓	✓
Instructions for submitting requests and link to online form or portal if offered [Regs §999.308(c)(1)(b), 999.308(c)(2)(b), 999.308(c)(2)(c)]	✓	✓	✗	✓	✓	✓
Instructions for authorized agents to make requests [Regs §999.308(c)(5)(a)]	✓	✓	✗	✓	✗	✓
Description of the process used to verify requests [Regs §999.308(c)(1)(c)]	✓	✓	✗	✓	✗	✓
List of categories of personal information collected in preceding 12 months [CCPA §1798.130(a)(5)(B), 1798.110(c), Regs §999.308(c)(1)(d)]	✓	✓	✗	✓	✗	✓
List of categories of personal information sold in preceding 12 months [CCPA §1798.130(a)(5)(C), 1798.115(c)(1), Regs §999.308(c)(1)(g)(1)]	✓	✓	✗	✓	✗	✓
List of categories of personal information disclosed for business purpose in preceding 12 months [CCPA §1798.130(a)(5)(C), 1798.115(c)(2), Regs §999.308(c)(1)(g)(1)]	✗	✗	✗	✓	✗	✓
For each personal information category, categories of third parties to whom information was disclosed or sold [Regs §999.308(c)(1)(g)(2)]	✓	✓	✗	✓	✗	✓
Categories of sources from which personal information is collected [Regs §999.308(c)(1)(e)]	✓	✓	✓	✓	✓	✓
Business or commercial purpose for collecting or selling personal information [Regs §999.308(c)(1)(f)]	✓	✓	✗	✓	✗	✓
Whether the business has actual knowledge that it sells personal information of minors under 16 years of age and special process [Regs §999.308(c)(1)(g)(3), 999.308(c)(9)]	✓	✓	✗	✗	✗	✓
Contact information for questions or concerns [Regs §999.308(c)(6)(a)]	✓	✓	✓	✓	✓	✓
Date policy was last updated [Regs §999.308(c)(7)]	✓	✓	✓	✓	✓	✓
Special requirements for businesses buying, receiving, selling, or sharing personal information of 10,000,000 or more consumers in a calendar year [Regs §999.308(c)(8), 999.317(g)(1)]	✗	✗	✗	✗	✗	✗
For online notices, follow generally recognized industry standards, such as the W3C Web Content Accessibility Guidelines, version 2.1 of June 5, 2018 [Regs §999.308(a)(2)(d)]	✗	✗	✗	✗	✗	✗

TABLE II: CCPA privacy policy requirements and generators' compliance.

C. Privacy Compliance Analysis

Privacy policies take a long time to read [44]. However, there would not even be a point in reading a policy if it does not describe the actual behavior of an app or other piece of software. In fact, reading it could even be misleading. The potential misalignments between policies and apps motivate us to study the extent to which generated policies are susceptive to such compliance issues. All results in this subsection are current as of May 2020.

1) The Law on Privacy Compliance: We define a compliance issue to mean that an app is performing a privacy practice (e.g., uses location data) while its privacy policy does not disclose it or discloses the opposite (e.g., discloses that it is *not* using location data) [86]. The notion that a policy must accurately describe the behavior of the app it covers is both evident as well as explicitly described in the law. Per the CCPA Regulations [Regs §999.308(a)(1)], it is the purpose of the policy to provide consumers with a comprehensive description of online practices regarding the collection, use, disclosure, and sale of personal information. Using clear and plain language, notification of data processing practices must be provided in a concise, transparent, intelligible and easily accessible form [GDPR Art. 12(1)]. Not properly disclosing a practice can be an unfair or deceptive act or practice in or affecting commerce [15 USC §45] and result in a privacy enforcement action by the Federal Trade Commission. The Apple Developer Program License Agreement also specifies that developers must provide a privacy policy explaining the collection, use, disclosure, sharing, retention, and deletion of user or device data [9]. Consequently, Apple's vetting process for publishing an app on the App Store includes the identification of compliance issues as well.[5]

2) Permission Under- and Over-disclosures: The first types of compliance issues we are examining are under- and over-disclosures of permissions used by apps and their integrated third party libraries [86], [87]. The threat model is that once a user grants permission, an app or library is able to access and send the related data off of the device. The set of apps we evaluate consists of the 95 apps whose App Store pages link to the generated policies we collected (§ III-A). After we downloaded and installed the apps from the App Store on an iPhone, we explored their permission use. If necessary, we created an account with the app, navigated to all views, and tapped on every interactive element to trigger any permission uses. To avoid under-disclosures it is not sufficient that the

[5]Apple CEO Tim Cook explained in an interview with MSNBC [47]: "[Apple is] looking at every app in detail. What is it doing? Is it doing what it is saying it is doing? Is it meeting the privacy policy that they are stating, right? And so we are always looking at that."

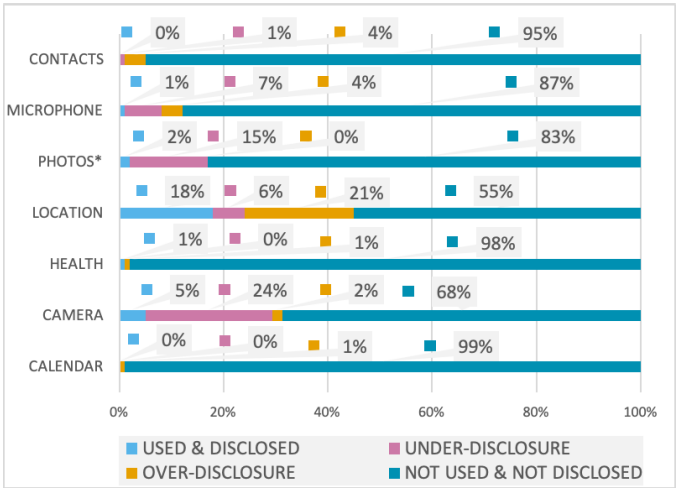

Fig. 1: Permission under- and over-disclosures (by permission). *`UIImagePickerController` allows use of some photos without Photos permission. Therefore, it is not included here.

Questionnaire-based Policy Generator	**Apps With At Least One *Under*-disclosure**	**Apps With At Least One *Over*-disclosure**
App Privacy Policy Generator	7/20 (35%)	0/20 (0%)
iubenda	5/20 (25%)	9/20 (45%)
Termly	5/20 (25%)	**14/20 (70%)**
TermsFeed et al.	**18/35 (51%)**	2/35 (6%)

TABLE III: Permission under- and over-disclosures (by generator).

policy of an app contains general disclosures (e.g., "we use your device data"). Rather, we require that a policy specifies a certain type of information (e.g., "we use your location data"). Under-disclosures can be particularly problematic for third party permission uses as users may incorrectly assume that granting a permission only refers to an app itself but not to integrated libraries. An over-disclosure refers to an app not using a permission that is actually disclosed in its policy. Arguably, over-disclosures are less harmful than under-disclosures as an app will simply use fewer permissions than disclosed. However, over-disclosures can contribute to eroding trust in privacy policies in general as those may not appear trustworthy if apps behave differently from what policies describe. While our analysis of over-disclosures aims to be as comprehensive as possible, it should be noted that there is always a possibility of missing to trigger a permission, which should be taken into account for the interpretation of our results.

Apps with policies from iubenda and Termly have fewer under-disclosures compared to apps with policies from the App Privacy Policy Generator and TermsFeed et al. The latter exhibits the maximum with 51% (Table III). One reason for iubenda and Termly covering apps' permission uses better than the other two could be the design of the generators' questionnaires. iubenda's and Termly's questionnaires include menus to select the permissions an app uses and also offer the option to manually provide additional permissions not contained in the menus. On the other hand, the App Privacy Policy Generator has just a free-form text field and TermsFeed et al. only offer Location, Contacts, and Camera permissions without any other options or free-form fields. Indeed, *any* permission use we observed for apps with policies from TermsFeed et al. — except for the location permission use — resulted in an under-disclosure.

Over-disclosures are particularly pronounced for the location permission with 21% (Figure 1). The majority of these over-disclosures are contained in privacy policies generated by Termly, which has, by far, the largest percentage of over-disclosures with 70% (Table III). A reason for these frequent location over-disclosures in Termly policies could lie in the policy text that is generated when answering affirmatively that an app will "be collecting any derivative data" from users. The rather unspecific term of "derivative data" is not tailored to apps and covers a host of data types "such as your IP address, browser and device characteristics, operating system, language preferences, referring URLs, device name, country, *location*, and information about how and when you use our Apps" (emphasis added) [69]. In addition, Termly is asking about location use in multiple other questions. Affirmative answers to those will also include location use in the generated policy. Overall, apps' permission use is not well reflected in the examined policies. Our results illustrate that the design of a generator's questionnaire can have an adverse impact on the accuracy of disclosures in generated policies.

3) Library Under- and Over-disclosures: Oftentimes, developers are not aware of third party library practices or if they are, they may not feel responsible for those [14], [45]. The integration of libraries can result in compliance issues if their use is not properly disclosed in privacy policies. We examined the integration of 10 popular libraries in our set of 95 apps and their disclosure in the generated privacy policies. While we interacted with each app, we decrypted and observed its resulting web traffic using the Fiddler web proxy [54]. After we had accessed all views and interactive elements of an app we analyzed the log files with the captured traffic. If, for instance, an app made a request to `graph.facebook.com`, we concluded that it integrates Facebook. We used the following list of ad and analytics domains to identify third party library integration.

(1) AdMob: `googleads.g.doubleclick.net`, `pubads.g.doubleclick.net`
(2) Facebook: `graph.facebook.com`
(3) Vungle: `ads.api.vungle.com`
(4) AdColony: `events3.adcolony.com`
(5) MoPub: `ads.mopub.com`
(6) Chartboost: `live.chartboost.com`
(7) Tapjoy: `ws.tapjoyads.com`
(8) UnityAds: `unityads.unity3d.com`
(9) InMobi: `sdktm.w.inmobi.com`
(10) Flurry: `data.flurry.com`

We require disclosure of the specific libraries, per GDPR Art. 13(1)(e), 14(1)(e), and not just of "other companies," for example. Apart from the GDPR, some laws generally only require the disclosure of categories of third parties and not their specific names (e.g., "ad network" instead of "AdMob"). However, as all generators provide functionality for selecting or entering specific third parties in their questionnaire, we are holding them to that standard.

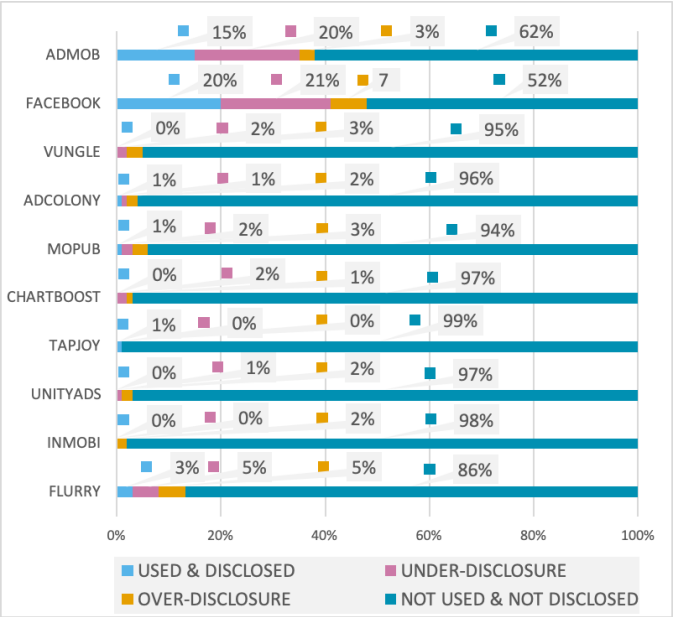

Fig. 2: Library under- and over-disclosures (by library).

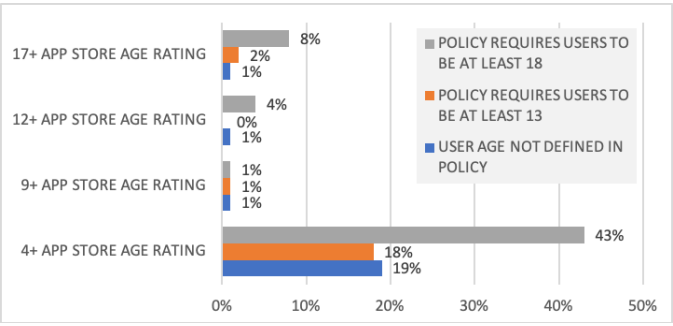

Fig. 3: App Store age rating and policy age requirements.

Our results reflect the online advertising duopoly [52] with Google's AdMob and Facebook each being integrated in about a third of the apps. However, only 15% of apps disclose in their policies that they integrate AdMob and 20% do not. The results for under-disclosures of Facebook are similar. 20% of apps disclose Facebook integration and 21% omit such disclosure. This trend holds for all other libraries as well, albeit, at lower integration levels (Figure 2). Compared to the other generators' policies, it is striking that the rate for under-disclosures in policies from TermsFeed et al. is more than twice as high with 63% (Table IV).

Termly and iubenda provide menus with hundreds of third party libraries for inclusion in the generated policies. On the other hand, the App Privacy Policy Generator and TermsFeed et al. only offer a handful of libraries. As all generators have an option to manually add libraries not included in their menus, though, it is generally possible to discloses any library. However, another difference between the generators is their pricing structure. The App Privacy Policy Generator is completely free. Termly and iubenda are offering a basic version of their generator for free and a more extensive paid version for a flat subscription fee. But TermsFeed et al. is

Questionnaire-based Policy Generator	Apps With At Least One *Under*-disclosure	Apps With At Least One *Over*-disclosure
App Privacy Policy Generator	5/20 (25%)	1/20 (5%)
iubenda	6/20 (30%)	5/20 (25%)
Termly	5/20 (25%)	2/20 (10%)
TermsFeed et al.	**22/35 (63%)**	5/35 (14%)

TABLE IV: Library under- and over-disclosures (by generator).

offering à la carte pricing where the addition of each library category, e.g., analytics, advertising, or marketing, will incur extra charges. Also, TermsFeed is the one we're focusing solely on now. Developers may feel disincentivized from extensive privacy disclosures due to the increase in fees that would come with those. Thus, the reason for TermsFeed et al.'s relatively higher levels of library under-disclosures may be rooted in its pricing structure indicating that such can have an impact on privacy compliance as well.

4) Improper Disclosures of Childrens' Apps: Online Service operators who direct their apps to children under 13 or who have actual knowledge of such users must provide COPPA disclosures in their policies (Appendix A). For example, they must disclose that a parent can review, have deleted, and refuse to permit further collection or use of the child's personal information [COPPA §312.4(d)(3)]. Whether an app is directed to children depends, among others, on its subject matter, its visual content, and the use of animated characters or child-oriented activities and incentives, music or other audio content [29]. In our set 6 of the 95 apps meet these criteria. However, neither of the apps' policies contain any COPPA disclosures. Rather, they assume teenage or adult users with 3 policies requiring a minimum age of 13 and the other 3 requiring a minimum age of 18. However, even if the developers wanted to make accurate COPPA disclosures, the generators would not allow them to do so comprehensively as they do not implement all requirements (Table I).

61% of apps' policies require users to be at least 13 or 18 years old while their apps are rated for age 4+ (Figure 3). This discrepancy is not problematic because age ratings do not mean that an app is suitable for children of a certain age but rather only that it is *un*suitable below that age [41]. However, for 2% of apps the discrepancies between their age ratings and policies' age requirements present a problem. Their age ratings require users to be 17+ while their policies only require users to be at least 13. The policies effectively allow access to violence, nudity, and other content unsuitable for younger users. Age ratings are voluntary and self-regulatory industry efforts. Still, policies should be consistent with the ratings.

IV. Generating Policies with PrivacyFlash Pro

We aim to increase privacy transparency and reduce compliance issues with a privacy policy generator that squarely fits developers' mental model and tightly integrates into the software development process and tools.[6]

[6]We presented PrivacyFlash Pro at iOSoho - New York City's largest iOS Engineer Meetup [34] and were featured on Brian Advent's iOS development YouTube channel [3]. All source code is available at https://github.com/privacy-tech-lab/privacyflash-pro/, accessed: January 7, 2021.

```
PLIST:
  LOCATION:
  - NSLocationAlwaysAndWhenInUseUsageDescription
  - NSLocationWhenInUseUsageDescription
  - NSLocationAlwaysUsageDescription
  - NSLocationUsageDescription
FRAMEWORK:
  SWIFT:
    LOCATION:
    - CoreLocation
    - MapKit
CLASS:
  SWIFT:
    LOCATION:
    - CLLocationManager
AUTHORIZATION_METHOD:
  SWIFT:
    LOCATION:
      requestAlwaysAuthorization: []
      requestWhenInUseAuthorization: []
ADDITIONAL_EVIDENCE:
  SWIFT:
    LOCATION:
      startUpdatingLocation: []
      stopUpdatingLocation: []
      requestLocation: []
      startMonitoringSignificantLocationChanges: []
      stopMonitoringSignificantLocationChanges: []
      startUpdatingHeading: []
      stopUpdatingHeading: []
      startMonitoringVisits: []
      stopMonitoringVisits: []
      startRangingBeacons:
      - satisfying
```

Fig. 4: Excerpt from PrivacyFlash Pro's iOS API evidence specification for the location framework. The full specification, about 3,000 lines of code, including comments, is available at our project's GitHub repository (https://github.com/privacy-tech-lab/privacyflash-pro/).

A. PrivacyFlash Pro Architecture

Current questionnaire-based generators have design weaknesses (§ III). They also necessarily rely on developers' ability to answer questions on the privacy practices of their apps accurately, comprehensively, and over time upon any privacy-relevant code change. We suggest automating privacy policy generation by using, as far as possible, first, standardized templates (§ IV-A1), second, automatic code analysis (§ IV-A2), and third, a questionnaire-based wizard (§ IV-A3).

1) Standardized Templates: Different apps will often be subject to the same disclosure requirements. For example, any app subject to the GDPR must disclose that users can request access to the data a controller has stored on them [GDPR, Art. 13(2)(b), 14(2)(c)].[7] These types of disclosures are included via templates in any policy that needs to comply with the GDPR. PrivacyFlash Pro contains templates for provisions of the GDPR, CCPA, CalOPPA, and COPPA. Such standardization eases policy comparison and comprehensibility by use of familiar terminology and placement of information resources [23]. Using standardized templates also has the advantage of enabling machine-readability of policies allowing browsers and other user agents to consume policy data and take actions on a user's behalf [23].[8] Making natural language policies machine-readable seems more promising at this point than developing dedicated machine-readable policy formats due to the wide adoption of the former. The more standardized language a policy contains, the easier it will become to make it machine-readable.

2) Code Analysis: Using Python for the code analysis logic and JavaScript for the UI, PrivacyFlash Pro runs as a macOS desktop app locally in the web browser and generates policies for iOS apps written in Swift and their libraries in Swift and Objective-C. The signature-based static analysis approach is language-agnostic as long as the analyzed code is sufficiently expressive to reveal information about its privacy-sensitive behavior when executed.

[7]PrivacyFlash Pro generates policies for GDPR "controllers" and CCPA "businesses" as opposed to "processors" and "service providers."

[8]The GDPR reintroduced this idea by providing for policy disclosures in combination with machine-readable icons [GDPR, Art. 12(7)].

a) From Evidence to Signatures to Policies: The detection of a privacy practice by PrivacyFlash Pro depends on the evidence found in the analyzed codebase. To that end, PrivacyFlash Pro contains a specification layer, distinct from the analysis logic, that defines the evidence that will be searched for. The specification contains hundreds of evidence items from the iOS Swift and Objective-C APIs [8]. Figure 4 shows an excerpt. As Swift and Objective-C continue to evolve, the specification can be updated without restructuring the analysis logic. The following are the different types of evidence items and their uses in the analysis:

(1) *Plist Permission Strings*. Since iOS 10.0 apps must include in their `Info.plist` file a `UsageDescription` key explaining why access to a certain permission is requested. This explanation is used in the generated policy to describe the purpose for why personal data is processed [e.g., GDPR, Art. 13(1)(c), 14(1)(c)].

(2) *Framework Imports*: All functionality we are interested in is bundled in frameworks. For example, an app must import the `CoreLocation` or `MapKit` frameworks to obtain the location of a device.

(3) *Class Instantiations*. At least one class from the framework must be instantiated. Otherwise, the framework's functionality could not be used.

(4) *Authorization Methods*. There must be instance method declarations for authorization methods (for example, `requestAlwaysAuthorization` to request the device location whenever the app is running). Sometimes they are also called access methods (for example, `requestAccess` to request access to the device's microphone). Parameters disambiguate methods with the same name (for example, both microphone and camera have a `requestAccess` method but microphone has an `audio` parameter and camera has a `video` parameter).

(5) *Entitlements*. Using particularly sensitive resources that are expanding beyond the sandbox of an app requires an entitlement, for example, use of `HealthKit`. Those are

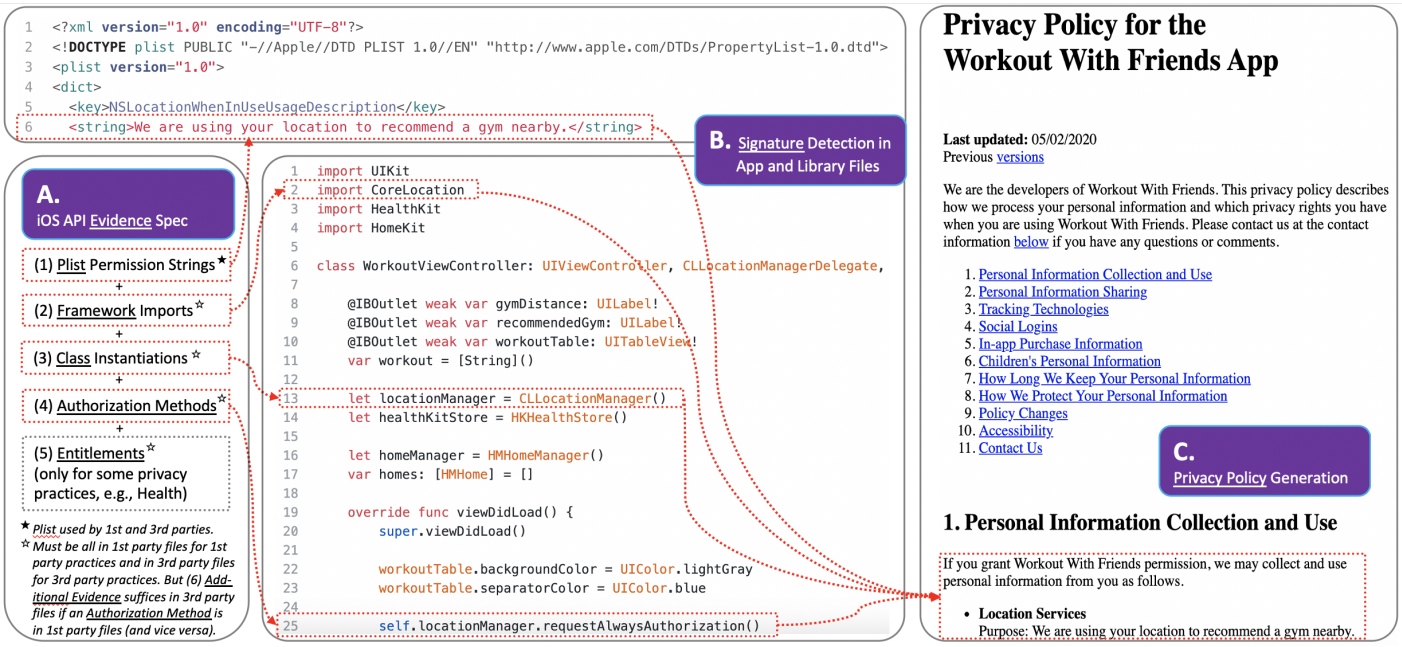

Fig. 5: Detection of a location signature in an app's source files causing the privacy policy to be populated with the location practice.

controlled by Apple through the app signing process and must be declared in the `Entitlements.plist` file.

(6) *Additional Evidence*. Instance methods (for example, `startUpdatingLocation`) and instance properties (for example, `AVCaptureDeviceInput`) require an authorization method to be used. Just as for authorization methods, parameters are used to disambiguate instance methods with the same name.

Once the evidence is completed to a signature, it is inferred that a privacy practice is performed that should be disclosed in a privacy policy. The signatures for first parties (i.e., apps) and third parties (i.e., libraries) are composed of the same evidence items. For a signature of a first party practice, the first party (i.e., app) code must contain (1) a Plist permission string, (2) a framework import, (3) a class instantiation, (4) an authorization method, and (5) an entitlement, to the extent required for the practice, where (6) additional evidence can be used in place of an authorization method if and only if an authorization method is present in the third party (i.e., library) code. For a signature of a third party practice, the first party code must contain (1) and the third party code must contain (2) through (5), where (6) additional evidence can be used in place of an authorization method if and only if an authorization method is present in the first party code. As third parties do not have their own `Info.plist`, they reuse the first party's. Similarly, third parties can reuse the first party's authorization method, in which case additional evidence in the third party code is required to show that the practice is actually used by the third party. Although, unlikely to occur, the first party can also reuse a third party's authorization method, in which case the first party code needs additional evidence. Figure 5 illustrates the idea. Generally, the evidence-based approach is also used to search for functionality within libraries, such as federated Facebook or Google Login.

b) Detecting Libraries and their Purposes: The code analysis distinguishes first and third parties since various legal requirements depend on such distinction. For example, apps subject to CalOPPA must disclose whether third parties may collect personally identifiable information [CalOPPA, §22575(b)(6)]. As described, third party libraries can make use of the same APIs as the app itself. Whether libraries are written in Swift or Objective-C, all APIs remain visible in plaintext even when the libraries are compiled. Thus, libraries in both source and compiled format can be analyzed.

PrivacyFlash Pro scans through the project directory of an app, e.g., searching for the `Google-Mobile-Ads-SDK` that would indicate the presence of Google's AdMob library. As app developers in the iOS ecosystem usually include libraries via package managers, such as CocoaPods [22] or Carthage [19], libraries are recognized based on Pods and Carthage directories as well as other framework resources from package managers used in an app. For example, the name of a third party can be reliably identified from the `Podfile` if the third party library was integrated via CocoaPods. Use of a package manager ensures that the library directory is present and named accordingly as otherwise the build process for the app would fail. While the third party library identification necessarily hinges on the use of a package manager and does not cover libraries manually included in an app, our performance analysis indicates that only a tiny fraction of libraries might be missed (§ IV-B1).

The purpose for a permission use by a first party can be inferred from the Plist permission string. However, the purpose for a third party use is usually not explicitly specified in the app or library code. Thus, the specification layer of PrivacyFlash Pro also contains a third party purpose specification. At this point, the specification contains the purposes for 300 popular third party libraries we identified on the analytics service Apptopia [10]. It is our goal to grow this specification with

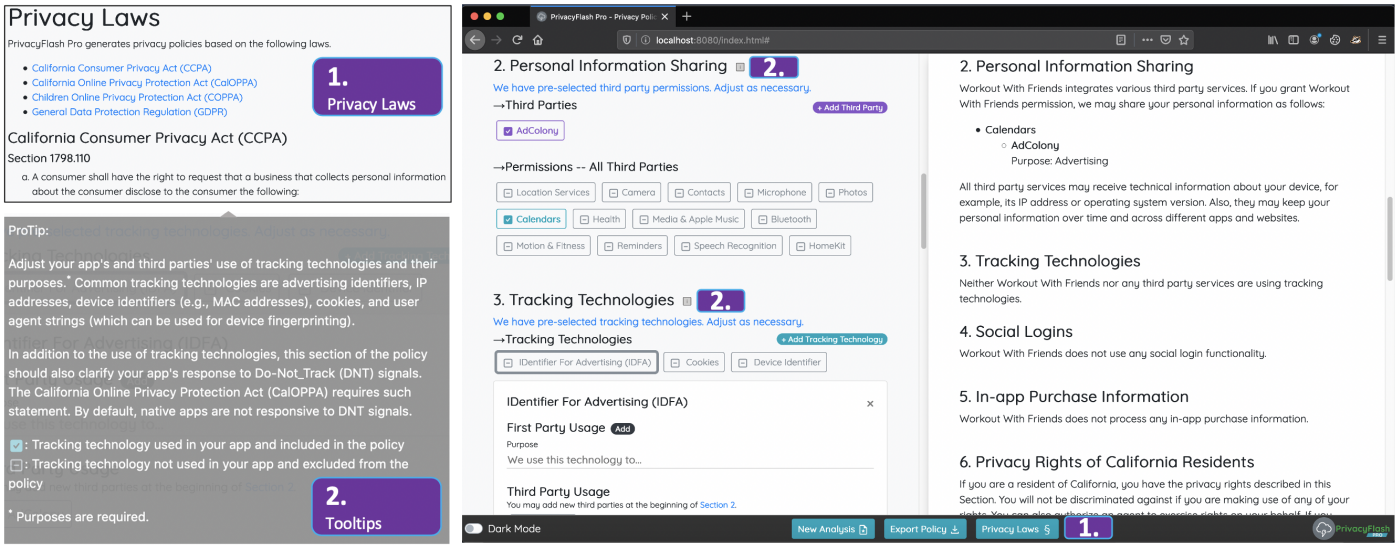

Fig. 6: Screenshot of the wizard and policy generation UI including an excerpt from the privacy law overview (1) and a tooltip (2).

open source contributions. The purpose categories we use so far (and the quantity of libraries) are: authentication (9), advertising (105), analytics (38), social network integration (31), payment processing (10), and developer support (107). It should be noted, though, that even if the purpose of a library is not included in the specification, the integration of the library will still be detected as long as it was done with a package manager. The developer can also always specify the purpose manually in the questionnaire wizard.

3) Questionnaire Wizard: Once the code analysis is finished, the developer is presented with a wizard for adjusting any (un)detected practices. The wizard helps the developers to determine which laws are applicable to their apps and provides explanations of the law. Despite the evidently large impact that different laws can make on what must be disclosed (Appendix A), existing generators are sparse in this regard. Implementation details for the policy, e.g., where it is to be posted, and related topics, e.g., whether the developer is required to provide Do Not Sell functionality, are covered in tooltips. Once the developer has finalized the policy, it can be exported into an HTML page that can be readily posted on the developer's website and that is accessible as per CCPA Regs §999.308(a)(2)(d). Figure 6 shows a screenshot of the wizard with the AdColony library detected. Since making it available to the public, we start to see privacy policies created with PrivacyFlash Pro in the field (§ V-D3).

B. PrivacyFlash Pro Performance

To test its practicability we evaluated the code analysis and runtime performance of PrivacyFlash Pro.

1) Code Analysis Performance: We started evaluating PrivacyFlash Pro on an app we created with 13 permissions and 5 libraries. Running PrivacyFlash Pro on this app resulted in a fully correct analysis. We also randomly selected 10 apps from the Collaborative List of Open-Source iOS Apps [27] and other public repositories covering 18 permission, at least one from each of the 13 permissions, and 45 unique libraries.

Permission Category	True Positives	False Positives	False Negatives
Bluetooth	2	0	0
Calendars	4	0	0
Camera	15	0	2
Contacts	3	0	1
Health	0	0	0
HomeKit	0	0	0
Location	21	0	0
Microphone	1	0	0
Motion & Fitness	0	0	0
Media & Apple Music	2	0	1
Photos	14	0	1
Reminders	0	0	1
Speech Recognition	0	0	0
Sum	**62**	**0**	**6**

TABLE V: Detection of permission uses for the 40 apps analyzed by the participants in our usability study (first and third party uses combined). With 62 true positives and 6 false negatives the analysis achieves a precision of 1 and recall of 0.91 for an F-1 score of 0.95.

Running these apps, analyzing their permission and library uses (§ III-C2, § III-C3), and comparing them against the results of PrivacyFlash Pro did not result in any discrepancy. In addition, we asked the 40 participants in our usability study if they had encountered any analysis errors when using PrivacyFlash Pro. We also asked them to provide us with the policy they generated so we could observe the results.[9] Table V shows the permission performance reported by the participants.

We find it noteworthy that our signature-based code analysis does not produce any false positives, which could occur due to unreachable code, for instance. The false negatives are likely due to APIs not contained in our specification. From the feedback we received, this is at least true for 1 out of the 6 false negatives, which was based on an older Swift API. Another reason may be our categorization of evidence items, e.g., there may be a few cases where an additional evidence item should have been categorized as authorization

[9]The complete usability questionnaire is attached in Appendix B.

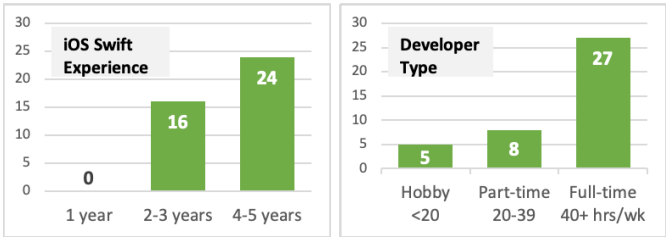

Fig. 7: Participants' qualifications for developing Swift iOS apps.

Third Party Code	True Positives	False Positives	False Negatives
Libraries	525	0	**1**
Facebook Login	9	0	**2**
Google Login	6	0	**3**

TABLE VI: Detection of third party libraries, Facebook Login, and Google Login.

method. The library detection performed well with just 1 false negative. However, the false negative rates for Facebook and Google Login implementations are higher with 2/11 (18%) and 3/9 (33%) instances (Table VI). Login detection goes beyond identifying a library as it aims to reason about its functionality. While we are following the same principles as for detecting other API uses (§ IV-A2), there are many different ways for integrating Facebook and Google Login. At least in 2 instances the reported APIs were not included in our specification. Thus, performance could be improved by increasing the number of APIs in the specification possibly combined with a slight re-categorization of evidence items.

2) Runtime Performance: We ran PrivacyFlash Pro on a small (91MB) and a large (624MB) app. Running the analysis 5 times for each, we measure an average runtime of 13.6s for the small and 46.3s for the large app on a 13-inch MacBook Pro (2017) with 2.3GHz Intel Core i5 CPU and 8GB RAM.

3) Limitations: PrivacyFlash Pro's code analysis may lead to false positives and false negatives that the developer would need to correct in the wizard. If left uncorrected, false positives would lead to over-disclosures and false negatives to under-disclosures. It is the developer's responsibility that the privacy policy reflects the app's practices correctly. Further, PrivacyFlash Pro's code analysis does not account for server-side data sharing, which may require taint tracking. It also does not take into account different library configurations.

V. Evaluating PrivacyFlash Pro's Usability

We performed a usability study of PrivacyFlash Pro with 40 iOS developers. After the participants had used PrivacyFlash Pro, we asked them in an online survey whether they found it helpful for policy creation as well as easy to use.

A. Participant Recruitment and Experience

We obtained our institution's IRB approval and recruited participants on the freelance platform Upwork [72], from developer websites, such as the iOS programming community on Reddit [57] and in person at iOSoho - New York City's largest iOS Engineer Meetup [34]. We asked the participants

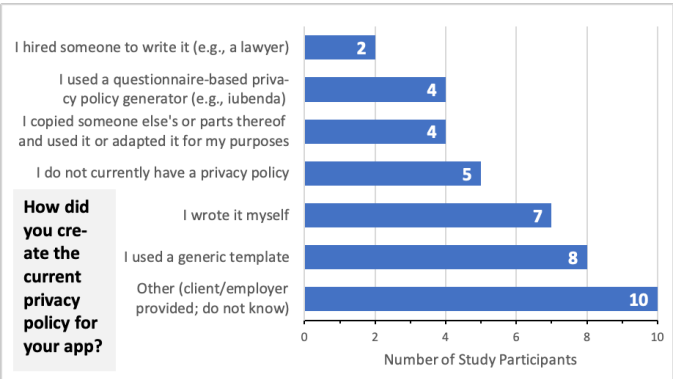

Fig. 8: The different methods that participants used to create their current privacy policy, if any.

to use PrivacyFlash Pro on an app they had written in Swift. We also required that they have an app published on Apple's App Store for the US. To ensure that participants were proficient in developing iOS apps in Swift we checked their Upwork profiles, especially, reviews of prior work in this area. For participants outside of Upwork we asked them for the e-mail address they use in their apps to verify that they are indeed the developers and sent their compensation to this address. To ensure that answers are reliable we included an attention question in the survey. The System Usability Scale, which is part of our survey (§ V-D1), contains a mix of positively and negatively worded statements that also forces attentiveness [61]. We further required participants to submit the policy they generated to ensure they actually used PrivacyFlash Pro. We paid every participant $20.[10] All participants were at least 18 years old; most ranging 20-29 (21) and 30-39 (14). 1 participant identified as female and 39 as male. Most were full time developers and had Swift experience of 4-5 years (Figure 7). 13 participants were from the US and 27 from other countries.

B. Non-compliance of Current Policies

As Figure 8 shows, 10 out of the 40 participants in our study were provided a policy for their app by their employer or client (or did not know how the policy was created). 5 participants did not have a policy, which can happen as the App Store only seems to require a policy link without Apple enforcing that it actually leads to a policy. The remaining participants created their policy themselves indicating a need for a policy generation tool. 26 participants provided us with their policy or a link to such. Upon examining those, many policies do not sufficiently cover their apps.

We observed similar compliance issues as those discussed for policies from questionnaire-based generators (§ III-C). 15/26 (58%) policies have at least one permission under-disclosure and 4/26 (15%) at least one permission over-disclosure. Library under-disclosures occur in 13/26 (50%) of cases. 2/26 (8%) policies also exhibit library over-disclosures. While 1 app was directed at children, its policy was not compliant with COPPA. These rates of compliance issues

[10]We increased the amount from $5 to $20 to motivate participation. Participants who participated before the increase received an additional $15.

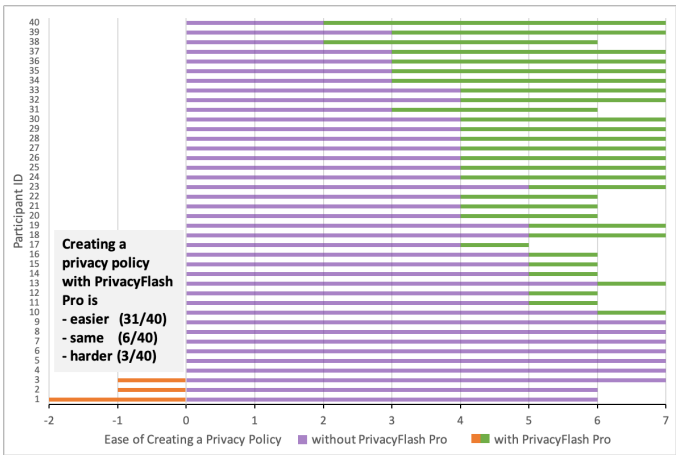

Fig. 9: For example, the participant with ID 40 at the top found creating their current privacy policy relatively difficult (2) and creating a policy with PrivacyFlash Pro very easy (7). On the other hand, the participant with ID 1 found the creation of their current policy relatively easy (6) and using PrivacyFlash Pro actually more difficult (6-2=4).

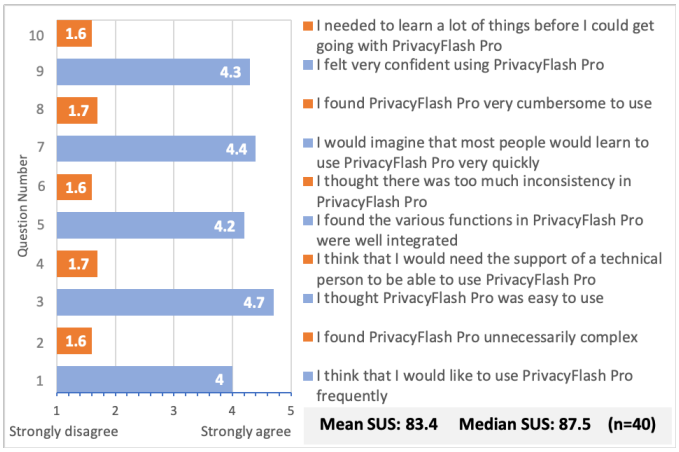

Fig. 10: The System Usability Scale results for PrivacyFlash Pro.

are generally a bit higher than those for the policies from questionnaire-based generators (§ III-C), which may well be taken as an indicator that generators can principally help to create compliant policies. Indeed, the policies that the participants generated with PrivacyFlash Pro have better coverage of their apps' permission and library usages than their current policies (§ IV-B1).

C. Easing the Policy Creation Process

Beyond ensuring that policies are compliant and have good coverage, PrivacyFlash Pro is intended to ease the policy creation process. Most participants expressed that privacy policy creation became substantially easier with PrivacyFlash Pro compared to their current method. When asked to rate the level of difficulty for creating their current policy on a scale of 1 (very difficult) to 7 (very easy), the mean difficulty across all participant responses converged to 4.675 and the median difficulty to 4. For creating a privacy policy with PrivacyFlash Pro, the mean increased by 1.875 to 6.55 and the median by 3 to 7. Figure 9 shows the differences in ratings for every individual participant. For 6 participants it was already very easy (7) to create their current policy. However, for 31 participants PrivacyFlash Pro provided an improvement and, except for participant 17, eased the difficulty to at least a level of 6. A number of developers expressed, unprompted for, that they were pleased with PrivacyFlash Pro and found it easy to use. Appendix D contains the complete set of comments.

D. Usability Measurement Results

We analyzed the usability of PrivacyFlash Pro based on the System Usability Scale (SUS) and Net Promoter Score (NPS).

1) System Usability Scale: The System Usability Scale is a 10-question scale for obtaining a global view of subjective assessments of usability [18]. It covers a variety of aspects, such as the need for technical support, training, and complexity, and, thus, is considered to have a high level of face validity for measuring usability of a system [18]. Each of the 10 questions is answered based on a Likert scale ranging from 1 (strongly disagree) to 5 (strongly agree). Then, the individual scores are added. Questions with odd numbers are positively formulated and contribute their score minus 1. Even-numbered questions are negatively formulated and contribute 5 points minus their score. Multiplying the sum of the scores by 2.5 puts each SUS score in a range between 0 and 100. Thus, given a score, s, of an individual question,

$$SUS = 2.5 \left(\sum_{i=0}^{4} (s_{2i+1} - 1) + \sum_{j=1}^{5} (5 - s_{2j}) \right). \quad (1)$$

PrivacyFlash Pro reached a mean SUS score of 83.4 and a median SUS score of 87.5 (Figure 10). For comparison, for 3,500 surveys of 273 studies covering, among others, web, UI, and hardware tools and systems, a total mean score of 69.5 was reported [16]. Mean scores of 71.4 are considered good (standard deviation 11.6) and scores of 85.5 excellent (standard deviation 10.4) [16], thus, placing PrivacyFlash Pro close to the mean of excellent scores. A number of participants provided suggestions for further improvements. The most frequent suggestion was to implement PrivacyFlash Pro as a native desktop app (6). The browser-based implementation raised concerns with some developers that their source code would be sent off of their devices. In a similar vein, more transparency about what is being done with the source code during the analysis was requested (3), which is a point that we will address with more and detailed explanations of how PrivacyFlash Pro works in the UI. The complete set of improvement suggestions is available in Appendix C.

2) Net Promoter Score: We asked study participants: "How likely is it that you would recommend PrivacyFlash Pro to a friend or colleague?" This single multiple choice question allows answers on a scale of 0 (not at all likely) to 10 (very likely). Participants scoring a 9 or 10 are considered promoters who will likely recommend PrivacyFlash Pro. Those who gave a score of 7 or 8 are reasonably satisfied but would not go out of their way. Detractors give scores between 0-6 and are dissatisfied. The Net Promoter Score is the difference

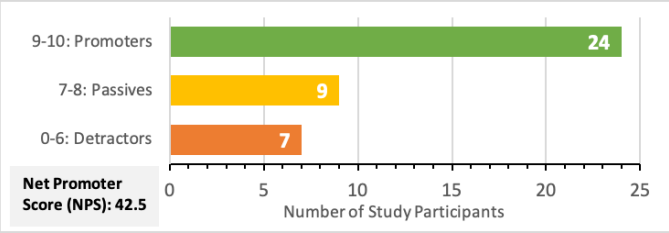

Fig. 11: PrivacyFlash Pro has 24 promoters and 7 detractors.

between the number of promoters, p, and detractors, d, given n participants, i.e.,

$$NPS = \frac{(p-d)}{n}100. \quad (2)$$

Net Promoter Scores can range from -100 to 100. PrivacyFlash Pro reaches an NPS of 42.5 (Figure 11). According to global benchmark data from SurveyMonkey, which accounts for the scores of more than 150,000 organizations, the average NPS is 32 with technology companies scoring a bit higher with an average of 35 and a median of 40 [67]. PrivacyFlash Pro places above these scores. Indeed, without prompting, one participant explicitly mentioned that they will recommend PrivacyFlash Pro (Appendix D).

3) Use of PrivacyFlash Pro in the Field: Since making it available to the public, we started to see privacy policies created with PrivacyFlash Pro in the field.[11]. As of January 7, 2021, the PrivacyFlash Pro GitHub repository was starred by 107 users, forked by 7, and 5 are watching it. A sample privacy policy generated with PrivacyFlash Pro is available in the GitHub repository of our project.[12]

VI. Conclusions

Privacy policies are the standard instruments for disclosing online privacy practices. They should provide users with transparent disclosures as to what happens with their data and which rights they have. Privacy policy generators can help developers creating legally compliant privacy policies. However, current generators are questionnaire-based and partly flawed in their design as well as fundamentally limited by the error-prone nature of the manual question-answering process. We believe that the combination of standardized policy templates, code analysis, and a wizard-based questionnaire has the potential to improve the current situation. To that end, we created PrivacyFlash Pro and introduced it as an open source project to the iOS developer community. The performance and feedback on the usability of PrivacyFlash Pro encourages us to move further in this direction.

A host of privacy-sensitive app functionalities can be surfaced from their code. Going forward it would be interesting to analyze additional app features and generate corresponding policy descriptions, for example, for an app's use of tracking technologies. Also, as Apple announced at its Worldwide Developers Conference 2020 that developers now need to provide privacy labels for their apps on the App Store [49], we are interested in exploring the extent to which our approach can be used for automating the label creation process. We also think that our approach can be used beyond iOS in the Android and web ecosystems. In fact, many privacy policies are intended to cover more than just one platform or app. Thus, generator results for multiple platforms and apps would need to be combined, possibly also with offline data collection practices. We see the potential for integrating policy generators like ours into developer tools. Generally, we understand our study as an initial step towards the automatic generation of privacy documentation.

We see a privacy tech trend that transcends disciplines of privacy law and software engineering. The emergence of web and app technologies has created new privacy challenges. However, those same technologies can be used to create new privacy solutions. If an app's codebase is not aligned with its privacy policy, one way to improve its alignment is to make privacy policies traceable by generating them as far as possible from the codebases they are intended to cover. In order to avoid non-compliance, whenever possible, it is not only advisable to design software with privacy built-in at its inception, but rather to synchronize it over the course of its life cycle with its privacy policy and other privacy documentation.

Software is not written in a vacuum; there are social, ethical, and legal considerations involved. By infusing the software development process and its tools with privacy requirements, developers are more likely to appreciate the need for compliance enabling them to write more compliant code. We think that the establishment of privacy policy generation as a native extension of the software development process will be beneficial for both privacy transparency and compliance. Software developers have embraced legal concepts before. With the open source movement as its catalyst, many developers came to appreciate software licensing as an integral component of their work. We think that a similar development is possible and desirable for privacy. Software developers can and should be stakeholders for creating compliant privacy policies. Adherence to a developer's mental model is critical. If developers can do what they are already doing — writing code — in a programming language they already know with tools they already use, privacy policies can become indeed software development artifacts.

Acknowledgment

We would like to thank the anonymous reviewers for their detailed reviews and most excellent suggestions. We would also like to thank Peter Story, Shaoyan Sam Li, and Yuanyuan Feng for their invaluable assistance in the early stages of this project and Kuba Alicki for his unit tests. Finally, PrivacyFlash Pro would not exist without the generous funding from Wesleyan University, its Department of Mathematics & Computer Science, and the Anil Fernando Endowment.

[11] See, e.g., https://codewithchris.com/cwc-mobile-privacy-policy/, https://www.simpleflights.app/privacy-policy/, https://perspect.photos/privacy/, or https://www.elitebaseballacademy.net/app-privacy-policy/ (each accessed: January 7, 2021).

[12] It can be viewed at https://htmlpreview.github.io/?https://github.com/privacy-tech-lab/privacyflash-pro/blob/master/iOS-sample-projects/Workout-with-Friends-Sample-Privacy-Policy.html (accessed: January 7, 2021).

References

[1] 3dcart. Create a GDPR personalized privacy policy template. https://www.3dcart.com/personalized-policy.html. Accessed: January 7, 2021.

[2] Yasemin Acar, Michael Backes, Sascha Fahl, Doowon Kim, Michelle L. Mazurek, and Christian Stransky. You get where you're looking for: The impact of information sources on code security. In *S&P*, 2016.

[3] Brian Advent. Privacy and software development. https://www.youtube.com/watch?v=7tpq0v4j_vM, May 24, 2020. Accessed: January 7, 2021.

[4] Zaina Aljallad, Wentao Guo, Chhaya Chouhan, Christy Laperriere, Jess Kropczynski, Pamela Wisnewski, and Heather Lipford. Designing a mobile app to support social processes for privacy and security decisions. In *USEC*, 2019.

[5] Benjamin Andow, Samin Yaseer Mahmud, Wenyu Wang, Justin Whitaker, William Enck, Bradley Reaves, Kapil Singh, and Tao Xie. PolicyLint: Investigating internal privacy policy contradictions on Google Play. In *USENIX Security*, 2019.

[6] Benjamin Andow, Samin Yaseer Mahmud, Justin Whitaker, William Enck, Bradley Reaves, Kapil Singh, and Serge Egelman. Actions Speak Louder than Words: Entity-Sensitive Privacy Policy and Data Flow Analysis with PoliCheck. In *USENIX Security*, 2020.

[7] App Privacy Policy Generator. Generate a generic privacy policy and terms & conditions for your apps. https://app-privacy-policy-generator.nisrulz.com/. Accessed: January 7, 2021.

[8] Apple. Apple Developer Documentation. https://developer.apple.com/documentation. Accessed: January 7, 2021.

[9] Apple. Apple Developer Program License Agreement, 06/22/20, §3.3.10. https://developer.apple.com/terms/. Accessed: January 7, 2021.

[10] Apptopia. https://apptopia.com/. Accessed: January 7, 2021.

[11] 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. In *PLDI*, 2014.

[12] Rawan Baalous and Ronald Poet. How dangerous permissions are described in android apps' privacy policies? In *Proceedings of the 11th International Conference on Security of Information and Networks*, SIN, 2018.

[13] Michael Backes, Sven Bugiel, and Erik Derr. Reliable third-party library detection in Android and its security applications. In *CCS*, 2016.

[14] Rebecca Balebako, Abigail Marsh, Jialiu Lin, Jason Hong, and Lorrie F. Cranor. The privacy and security behaviors of smartphone app developers. In *USEC*, 2014.

[15] Rebecca Balebako, Florian Schaub, Idris Adjerid, Alessandro Acquisti, and Lorrie Cranor. The impact of timing on the salience of smartphone app privacy notices. In *SPSM*, 2015.

[16] Aaron Bangor, Philip Kortum, and James Miller. Determining what individual sus scores mean: Adding an adjective rating scale. *J. Usability Studies*, 4(3):114–123, May 2009.

[17] Jasmine Bowers, Bradley Reaves, Imani N. Sherman, Patrick Traynor, and Kevin R. B. Butler. Regulators, mount up! analysis of privacy policies for mobile money services. In *SOUPS*, 2017.

[18] John Brooke. *SUS-A quick and dirty usability scale. Usability evaluation in industry*. CRC Press, June 1996.

[19] Carthage. https://github.com/Carthage/Carthage. Accessed: January 7, 2021.

[20] Xin Chen, Heqing Huang, Sencun Zhu, Qing Li, and Quanlong Guan. SweetDroid: Toward a context-sensitive privacy policy enforcement framework for Android os. In *WPES*, 2017.

[21] Saksham Chitkara, Nishad Gothoskar, Suhas Harish, Jason I. Hong, and Yuvraj Agarwal. Does this app really need my location?: Context-aware privacy management for smartphones. *ACM IMWUT*, 1(3):42:1–42:22, September 2017.

[22] CocoaPods. https://cocoapods.org/. Accessed: January 7, 2021.

[23] Lorrie Faith Cranor. Necessary but not sufficient: Standardized mechanisms for privacy notice and choice. *J. on Telecomm. and High Tech. L.*, 10(2):273–307, 2012.

[24] Lorrie Faith Cranor, Brooks Dobbs, Serge Egelman, Giles Hogben, Jack Humphrey, Marc Langheinrich, Massimo Marchiori, Martin Presler-Marshall, Joseph M. Reagle, Matthias Schunter, David A. Stampley, and Rigo Wenning. The Platform for Privacy Preferences 1.1 (P3P1.1) specification. World Wide Web Consortium, Note NOTE-P3P11-20061113, November 2006.

[25] Lorrie Faith Cranor, Marc Langheinrich, Massimo Marchiori, Martin Presler-Marshall, and Joseph M. Reagle. The Platform for Privacy Preferences 1.0 (P3P1.0) specification. World Wide Web Consortium, Recommendation REC-P3P-20020416, April 2002.

[26] Lorrie Faith Cranor, Pedro Giovanni Leon, and Blase Ur. A large-scale evaluation of U.S. financial institutions standardized privacy notices. *ACM Trans. Web*, 10(3):17:1–17:33, August 2016.

[27] dkhamsing. Collaborative list of open-source iOS apps. https://github.com/dkhamsing/open-source-ios-apps. Accessed: January 7, 2021.

[28] Manuel Egele, Christopher Kruegel, Engin Kirda, and Giovanni Vigna. PiOS: Detecting privacy leaks in iOS applications. In *NDSS*, 2011.

[29] Federal Trade Commission. Complying with COPPA: Frequently asked questions. https://www.ftc.gov/tips-advice/business-center/guidance/complying-coppa-frequently-asked-questions, March 20, 2015. Accessed: January 7, 2021.

[30] Adrienne Porter Felt, Erika Chin, Steve Hanna, Dawn Song, and David Wagner. Android permissions demystified. In *CCS*, 2011.

[31] Felix Fischer, Konstantin Böttinger, Huang Xiao, Christian Stransky, Yasemin Acar, Michael Backes, and Sascha Fahl. Stack Overflow considered harmful? the impact of copy&paste on Android application security. *CoRR*, abs/1710.03135, 2017.

[32] Free PrivacyPolicy.com. Create a free privacy policy. https://www.freeprivacypolicy.com/. Accessed: January 7, 2021.

[33] Hamza Harkous, Kassem Fawaz, Rémi Lebret, Florian Schaub, Kang G. Shin, and Karl Aberer. Polisis: Automated analysis and presentation of privacy policies using deep learning. In *USENIX Security*, 2018.

[34] iOSoho - New York City's largest iOS Engineer Meetup. Developing privacy policies for iOS. https://www.meetup.com/iOSoho/events/268790508, February 24, 2020. Accessed: January 7, 2021.

[35] iubenda. We help with the legal requirements, so you can focus on the business. https://www.iubenda.com/en/. Accessed: January 7, 2021.

[36] Oksana Kulyk, Paul Gerber, Karola Marky, Christopher Beckmann, and Melanie Volkamer. Does this app respect my privacy? design and evaluation of information materials supporting privacy-related decisions of smartphone users. In *NDSS*, 2019.

[37] Li Li, Alexandre Bartel, Jacques Klein, Yves Le Traon, Steven Arzt, Siegfried Rasthofer, Eric Bodden, Damien Octeau, and Patrick D. McDaniel. I know what leaked in your pocket: uncovering privacy leaks on Android apps with static taint analysis. *CoRR*, abs/1404.7431, 2014.

[38] Tianshi Li, Yuvraj Agarwal, and Jason I. Hong. Coconut: An ide plugin for developing privacy-friendly apps. *ACM IMWUT*, 2(4):178:1–178:35, December 2018.

[39] Yuanchun Li, Fanglin Chen, Toby Jia-Jun Li, Yao Guo, Gang Huang, Matthew Fredrikson, Yuvraj Agarwal, and Jason I. Hong. PrivacyStreams: Enabling transparency in personal data processing for mobile apps. *ACM IMWUT*, 1(3):76:1–76:26, September 2017.

[40] Timothy Libert. An automated approach to auditing disclosure of third-party data collection in website privacy policies. In *WWW*, 2018.

[41] I. Liccardi, M. Bulger, H. Abelson, D. J. Weitzner, and W. Mackay. Can apps play by the COPPA rules? In *2014 Twelfth Annual International Conference on Privacy, Security and Trust*, pages 1–9, 2014.

[42] Thomas Linden, Rishabh Khandelwal, Hamza Harkous, and Kassem Fawaz. The privacy policy landscape after the GDPR. In *PETS*, 2020.

[43] Frederick Liu, Shomir Wilson, Peter Story, Sebastian Zimmeck, and Norman Sadeh. Towards Automatic Classification of Privacy Policy Text. Technical Report CMU-ISR-17-118R and CMU-LTI-17-010, School of Computer Science Carnegie Mellon University, Pittsburgh, PA, June 2018.

[44] Aleecia M. McDonald and Lorrie F. Cranor. The cost of reading privacy policies. *I/S: A Journal of Law and Policy for the Information Society*, 4(3):540–565, 2008.

[45] Abraham H. Mhaidli, Yixin Zou, and Florian Schaub. "We can't live without them!" App developers' adoption of ad networks and their considerations of consumer risks. In *SOUPS*, 2019.

[46] Daniela Yidan Miao. PrivacyInformer: An automated privacy description generator for the MIT App Inventor. Ms thesis, Massachusetts Institute of Technology, Cambridge, US, 2014.

[47] MSNBC. Tim Cook slams Facebook's Zuckerberg: I wouldn't be in this situation. https://www.msnbc.com/msnbc/watch/tim-cook-slams-facebook-s-zuckerberg-i-wouldn-t-be-in-this-situation-1197130819683, March 2018. Accessed: January 7, 2021.

[48] Yuhong Nan, Zhemin Yang, Xiaofeng Wang, Yuan Zhang, Donglai Zhu, and Min Yang. Finding clues for your secrets: Semantics-driven, learning-based privacy discovery in mobile apps. In *NDSS*, 2018.

[49] Alfred Ng. Apple's new iOS privacy updates will show how apps are tracking you. https://www.cnet.com/news/apples-new-ios-privacy-updates-will-show-how-apps-are-tracking-you/, June 2020. accessed: January 7, 2021.

[50] Duc Cuong Nguyen, Dominik Wermke, Yasemin Acar, Michael Backes, Charles Weir, and Sascha Fahl. A stitch in time: Supporting Android developers in writing secure code. In *CCS*, 2017.

[51] Marten Oltrogge, Erik Derr, Christian Stransky, Yasemin Acar, Sascha Fahl, Christian Rossow, Giancarlo Pellegrino, Sven Bugiel, and Michael Backes. The rise of the citizen developer: Assessing the security impact of online app generators. In *S&P*, 2018.

[52] Nicole Perrin. US advertisers still eager to target at scale with duopoly. https://www.emarketer.com/content/us-advertisers-still-eager-to-target-at-scale-with-duopoly, May 21, 2019. Accessed: January 7, 2021.

[53] PrivacyPolicies.com. Privacy policies are required by law. get compliant today. https://www.privacypolicies.com/. Accessed: January 7, 2021.

[54] Progress Software Corporation. Fiddler. http://www.telerik.com/fiddler. Accessed: January 7, 2021.

[55] Rohan Ramanath, Fei Liu, Norman Sadeh, and Noah A. Smith. Unsupervised alignment of privacy policies using hidden markov models. In *ACL*, 2014.

[56] 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 *S&P*, 2019.

[57] Reddit, r/iOSProgramming. https://www.reddit.com/r/iOSProgramming/comments/em86k4/generate_privacy_policies_from_app_code/. Accessed: January 7, 2021.

[58] Sanae Rosen, Zhiyun Qian, and Z. Morely Mao. AppProfiler: A flexible method of exposing privacy-related behavior in Android applications to end users. In *CODASPY*, 2013.

[59] Mark Rowan and Josh Dehlinger. Encouraging privacy by design concepts with privacy policy auto-generation in Eclipse (Page). In *Proceedings of the 2014 Workshop on Eclipse Technology eXchange*, ETX '14, pages 9–14, New York, NY, USA, 2014. ACM.

[60] Kanthashree Mysore Sathyendra, Shomir Wilson, Florian Schaub, Sebastian Zimmeck, and Norman Sadeh. Identifying the provision of choices in privacy policy text. In *EMNLP*, 2017.

[61] Jeff Sauro and James R. Lewis. When designing usability questionnaires, does it hurt to be positive? In *CHI*, 2011.

[62] Shopify. Free privacy policy generator. https://www.shopify.com/tools/policy-generator. Accessed: January 7, 2021.

[63] Rocky Slavin. PoliDroid-AS. https://github.com/rslavin/PoliDroid-AS, 2019. Accessed: January 7, 2021.

[64] Rocky Slavin, Xiaoyin Wang, Mitra Bokaei Hosseini, James Hester, Ram Krishnan, Jaspreet Bhatia, Travis D. Breaux, and Jianwei Niu. Toward a framework for detecting privacy policy violation in Android application code. In *ICSE*, 2016.

[65] Stack Overflow. https://stackoverflow.com/. Accessed: January 7, 2021.

[66] Yi Ping Sun. Investigating the effectiveness of Android privacy policies. Master of applied science thesis, University of Toronto, Toronto, Canada, 2018. Accessed: January 7, 2021.

[67] SurveyMonkey. What is a good net promoter score? and how does it vary across industries? https://www.surveymonkey.com/curiosity/what-is-a-good-net-promoter-score/. Accessed: January 7, 2021.

[68] Madiha Tabassum, Abdulmajeed Alqhatani, Marran Aldossari, and Heather Richter Lipford. Increasing user attention with a comic-based policy. In *CHI*, 2018.

[69] Termly. Free privacy policy generator. https://termly.io/products/privacy-policy-generator/. Accessed: January 7, 2021.

[70] TermsFeed. Trusted legal agreements. https://www.termsfeed.com/. Accessed: January 7, 2021.

[71] Welderufael B. Tesfay, Peter Hofmann, Toru Nakamura, Shinsaku Kiyomoto, and Jetzabel Serna. I read but don't agree: Privacy policy benchmarking using machine learning and the eu gdpr. In *WWW*, 2018.

[72] Upwork. https://www.upwork.com/. Accessed: January 7, 2021.

[73] Christine Utz, Martin Degeling, Sascha Fahl, Florian Schaub, and Thorsten Holz. (un)informed consent: Studying GDPR consent notices in the field. In *CCS*, 2019.

[74] Raj Vardhan, Ameya Sanzgiri, Dattatraya Kulkarni, Piyush Joshi, and Srikanth Nalluri. Notify assist: Balancing privacy and convenience in delivery of notifications on Android smartphones. In *WPES*, 2017.

[75] T. Franklin Waddell, Joshua R. Auriemma, and S. Shyam Sundar. Make it simple, or force users to read?: Paraphrased design improves comprehension of end user license agreements. In *CHI*, 2016.

[76] Xiaoyin Wang, Xue Qin, Mitra Bokaei Hosseini, Rocky Slavin, Travis D. Breaux, and Jianwei Niu. Guileak: Identifying privacy practices on gui-based data. https://pdfs.semanticscholar.org/ced1/313acaacd3897b5b231cdccb1383d01d20c4.pdf, 2017. Accessed: January 7, 2021.

[77] Takuya Watanabe, Mitsuaki Akiyama, Tetsuya Sakai, and Tatsuya Mori. Understanding the inconsistencies between text descriptions and the use of privacy-sensitive resources of mobile apps. In *SOUPS*, 2015.

[78] Jean Yang, Kuat Yessenov, and Armando Solar-Lezama. A language for automatically enforcing privacy policies. In *POPL*, 2012.

[79] L. Yu, X. Luo, X. Liu, and T. Zhang. Can we trust the privacy policies of Android apps? In *DSN*, 2016.

[80] Le Yu, Tao Zhang, Xiapu Luo, and Lei Xue. AutoPPG: Towards automatic generation of privacy policy for Android applications. In *SPSM*, 2015.

[81] W. D. Yu and S. Murthy. PPMLP: A special modeling language processor for privacy policies. In *ICCE*, 2007.

[82] Mu Zhang, Yue Duan, Qian Feng, and Heng Yin. Towards automatic generation of security-centric descriptions for Android apps. In *CCS*, 2015.

[83] Yanyan Zhuang, Albert Rafetseder, Yu Hu, Yuan Tian, and Justin Cappos. Sensibility testbed: Automated irb policy enforcement in mobile research apps. In *HotMobile*, 2018.

[84] Sebastian Zimmeck and Steven M. Bellovin. Privee: An architecture for automatically analyzing web privacy policies. In *USENIX Security*, 2014.

[85] Sebastian Zimmeck, Peter Story, Rafael Goldstein, David Baraka, Shaoyan Li, Yuanyuan Feng, and Norman Sadeh. Compliance traceability: Privacy policies as software development artifacts. https://petsymposium.org/2019/files/workshop/abstracts/PUT_2019_paper_21.pdf, July 2019. Accessed: January 7, 2021.

[86] Sebastian Zimmeck, Peter Story, Abhilasha Ravichander, Daniel Smullen, Ziqi Wang, Joel Reidenberg, N. Cameron Russell, and Norman Sadeh. MAPS: Scaling privacy compliance analysis to a million apps. In *PETS*, 2019.

[87] Sebastian Zimmeck, Ziqi Wang, Lieyong Zou, Roger Iyengar, Bin Liu, Florian Schaub, Shormir Wilson, Norman Sadeh, Steven M. Bellovin, and Joel Reidenberg. Automated analysis of privacy requirements for mobile apps. In *NDSS*, 2017.

APPENDIX

A. Legal Requirements for Privacy Policies

The following privacy policy requirements are covered by PrivacyFlash Pro and, where indicated, by the examined questionnaire-based generators.[13] Also, an omission or wrong disclosure can be an unfair or deceptive act or practice [15 USC §45].

[13]The App Privacy Policy Generator [7] does not claim to be compliant with any of the laws and, thus, is not included in the analysis.

COPPA Requirement	iubenda	Termly	TermsFeed et al.
Name, address, telephone number, and email address of all operators collecting or maintaining personal information from children or of one operator who will respond to all inquiries if names of all operators are also listed [§312.4(d)(1)]	✗	✗	✗
Description of the operator's collection, use, and disclosure practices [§312.4(d)(2)]	✓	✓	✓
Description of whether the site or service enables a child to make personal information publicly available [§312.4(d)(2)]	✗	✗	✗
Statement that parent can review, have deleted, and refuse to permit further collection or use of the child's personal information and procedures for doing so [§312.4(d)(3)]	✓	✗	✓

TABLE A.7: COPPA privacy policy requirements. We analyzed each generator twice, in May 2020 and January 2021. The results were identical.

GDPR Requirement	iubenda	Termly	TermsFeed et al.
Identity and contact details of the data controller and their representative, if any [Art. 13(1)(a), 14(1)(a)]	✓	✓	✓
Purposes of the processing for which the personal data are intended and legal basis for the processing [Art. 13(1)(c), 14(1)(c)]	✓	✓	✓
Categories of personal data concerned [Art. 14(1)(d)]	✓	✓	✓
Recipients or categories of recipients of the personal data [Art. 13(1)(e), 14(1)(e)]	✓	✓	✓
Period for which the personal data will be stored, or if that is not possible, the criteria used to determine that period [Art. 13(2)(a), 14(2)(a)]	✓	✓	✓
Existence of the rights to request data access, rectification, erasure, and data portability as well as the rights to restrict and object to processing [Art. 13(2)(b), 14(2)(c)]	✓ ✓	✓ ✓	✓ ✓
Right to withdraw consent for processing at any time [Art. 13(2)(c), 14(2)(d)]	✓	✓	✓
Right to lodge complaint with a supervisory authority [Art. 13(2)(d), 14(2)(e)]	✓	✓	✓

TABLE A.8: GDPR privacy policy requirements. We analyzed each generator twice, in May 2020 and January 2021. The results were identical.

CalOPPA Requirement
Identify categories of personally identifiable information collected and categories of third parties with whom it is shared [§22575(b)(1)]
If the operator maintains a process to review and request changes to personally identifiable information, provide a description of that process [§22575(b)(2)]
Description of the process by which consumers are notified of material changes to the operator's privacy policy [§22575(b)(3)]
Identify the policy's effective date [§22575(b)(4)]
Disclosure how the operator responds to Do Not Track (DNT) signals or other mechanisms that provide consumers' choice as to the collection of personally identifiable information across sites and over time [§22575(b)(5)]
Disclosure whether other parties may collect personally identifiable information across sites and over time [§22575(b)(6)]

TABLE A.9: PrivacyFlash Pro complies with CalOPPA's requirements, which were not reviewed for the questionnaire-based policy generators.

B. Usability Survey Questionnaire

In our usability study (§ V) we used the following questionnaire.

1) Demographic Information:

- What is your age range? [Multiple choice; answer required]
  ○ 18–19 ○ 20–29 ○ 30–39 ○ 40–49 ○ 50–59 ○ 60–69 ○ 70 or older
- What is your gender? [Multiple choice; answer required]
  ○ Female ○ Male ○ Other [Short answer text; answer required if selected]
- Which country do you live in? [Short answer text; answer required]

2) Swift Development Experience:

- For how long have you been developing iOS apps in Swift? [Multiple choice; answer required]
  ○ 1 year ○ 2 to 3 years ○ 4 to 5 years
- Please provide the App Store link to your iOS app written in Swift available in the United States with the most user ratings. [Short answer text; answer required]

- What is your contribution to the app to which you linked in the previous question? [Long answer text; answer required]
- Tell us what type of app developer you are. [Multiple choice; answer required]
  - ○ Full-time developer (40 or more hours/week of mobile app development)
  - ○ Part-time developer (20-39 hours/week of mobile app development)
  - ○ Hobby developer (less than 20 hours/week of mobile app development)

3) Generating and Writing Privacy Policies:

- How did you create the current privacy policy for your app? [Multiple choice; answer required]
  - ○ I wrote it myself
  - ○ I used a generic template
  - ○ I copied someone else's or parts thereof and used it or adapted it for my purposes
  - ○ I hired someone to write it (e.g., a lawyer)
  - ○ I used a questionnaire-based privacy policy generator (e.g., iubenda)
  - ○ I do not currently have a privacy policy
  - ○ Other [Short answer text; answer required if selected]
- Please paste the text of your current privacy policy (plain text format). [Long answer text; answer not required]
- Please paste the link to your current privacy policy. [Short answer text; answer not required]
- Overall, how difficult or easy was creating your current privacy policy?
  [Multiple choice; answer not required; answer on a scale of "Very difficult 1 – 2 – 3 – 4 – 5 – 6 – 7 Very easy"]
- Please paste the privacy policy that you created for your app with PrivacyFlash Pro. [Long answer text; answer required]
- Please select the extent to which you agree. [System Usability Scale (SUS); multiple choice; all answers required; each answer on a scale of "Strongly disagree 1 – 2 – 3 – 4 – 5 Strongly agree"]
  - ○ I think that I would like to use PrivacyFlash Pro frequently
  - ○ I found PrivacyFlash Pro unnecessarily complex
  - ○ I thought PrivacyFlash Pro was easy to use
  - ○ I think that I would need the support of a technical person to be able to use PrivacyFlash Pro
  - ○ I found the various functions in PrivacyFlash Pro were well integrated
  - ○ I thought there was too much inconsistency in PrivacyFlash Pro
  - ○ I would imagine that most people would learn to use PrivacyFlash Pro very quickly
  - ○ I found PrivacyFlash Pro very cumbersome to use
  - ○ I felt very confident using PrivacyFlash Pro
  - ○ I needed to learn a lot of things before I could get going with PrivacyFlash Pro
- Overall, how difficult or easy was creating a privacy policy with PrivacyFlash Pro?
  [Multiple choice; answer required; answer on a scale of "Very difficult 1 – 2 – 3 – 4 – 5 – 6 – 7 Very easy"]
- Sample Privacy Policy: This privacy policy discloses our privacy practices. In order to use our website, a user must first complete the registration form to log in. You can log in using your Facebook or other social network account.
- Which type of company is mentioned in the sample privacy policy above? [Multiple choice; answer required]
  - ○ Social networks ○ Analytics services ○ Advertising networks ○ Hosting services
- Did PrivacyFlash Pro miss to flag one of the following resources that your app is using? [Checkboxes; answer required]
  - ○ PrivacyFlash Pro did not flag the BLUETOOTH permission although my app is using it
  - ○ PrivacyFlash Pro did not flag the CALENDARS permission although my app is using it
  - ○ PrivacyFlash Pro did not flag that my app integrates FACEBOOK LOGIN
  - ○ PrivacyFlash Pro did not flag that my app integrates GOOGLE LOGIN
  - ○ PrivacyFlash Pro did not flag an SDK that my app integrates
  - ○ None of the above; PrivacyFlash Pro flagged all of the above resources that are used by my app
- If PrivacyFlash Pro did not flag a resource your app is using per the previous question, please provide a link to the site with the instructions that you followed to integrate the resource in your app. [Long answer text; answer not required]
- Did PrivacyFlash Pro flag the presence of an aforementioned resource despite your app NOT using it?
  [Checkboxes; answer required]
  - ○ PrivacyFlash Pro flagged the BLUETOOTH permission, but my app is not using it
  - ○ PrivacyFlash Pro flagged the CALENDARS permission, but my app is not using it
  - ○ PrivacyFlash Pro flagged FACEBOOK LOGIN, but my app is not using it
  - ○ PrivacyFlash Pro flagged GOOGLE LOGIN, but my app is not using it
  - ○ PrivacyFlash Pro flagged use of an SDK, but my app is not using it
  - ○ None of the above; all resources that PrivacyFlash Pro flagged are used by my app
- If PrivacyFlash Pro flagged a resource that your app is not using per the previous question, did you still find it useful that PrivacyFlash Pro flagged it? (In particular, Privacy Flash Pro is intended to help developers maintaining a clean codebase by flagging unreachable code.) [Multiple choice; answer not required]
  - ○ Yes ○ No ○ Other [Short answer text; answer required if selected]
- How likely is it that you would recommend PrivacyFlash Pro to a friend or colleague? [Net Promoter Score (NPS); multiple choice; answer required; answer on a scale of "Not at all likely 0 – 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 10 Very likely"]
- We want to help app developers. How could we improve PrivacyFlash Pro? [Long answer text; answer not required]

- Please enter your e-mail address where you would like to receive your gift card or, if you are participating via Upwork, please enter the name as used on Upwork. [Short answer text; answer required]

C. Improvement Suggestions from the iOS Developer Community

We received the following responses to our question "How could we improve PrivacyFlash Pro?" in the usability survey.

(1) No suggestion (16)
(2) Implement PrivacyFlash Pro as a native desktop app (6)
(3) Improve the UI/UX (e.g., the color scheme or toggle switches) (4)
(4) Add support for multiple policy file formats (e.g., Word and pdf) (3)
(5) Be more transparent about what is being done with the source code during the analysis (3)
(6) Make the tooltips more specific, extensive, or explain more of the law (2)
(7) Add to the third party library analysis more details on what they are doing and what their risks are (2)
(8) Implement PrivacyFlash Pro as a web app for analyzing code repositories online (2)
(9) Allow developers to manually include or exclude files or directories from the analysis (2)
(10) Create a marketplace for lawyers to confirm the generated policies (1)
(11) Add options "Maybe" or "I don't know/not sure" to the wizard (1)
(12) Add functionality to store and modify generated policies (1)
(13) Make the setup process easier (1)
(14) Add multi-language support (1)

D. Comments from the iOS Developer Community

We received the following comments from study participants via the Upwork chat [72] or e-mail. Additional comments were provided on Reddit's r/iOSProgramming [57] where we solicited participants for our study. Neither comment was prompted for.

(1) "Thanks for this awesome application, very helpful"
(2) "As a developer, I am very pleased to use PrivacyFlash Pro."
(3) "It is great. Awesome work I am really impressed."
(4) "I found [PrivacyFlash Pro] very useful for future porpose [sic]."
(5) "I'll use the application while developing iOS apps. It is great."
(6) "This tool is amazing! Congratulations. I really mean that."
(7) "i do really like privacy flash pro"
(8) "Your project is very interested [sic], but honestly I have never written the privacy info myself. But I realize how to make privacy doc easily. if I need to write [...] that, I will recommend to use your program"
(9) "it is quite useful"
(10) "Neat tool you got there. It would be awesome to have something similar for GDPR too.[14] One thing tho, for the flags, in my opinion, it's a bit confusing to add/remove any flags. I got a flag for a 3rd party lib, but it was a bit misleading on how to remove id [sic]. Anyway, congrats on the cool tool and good luck !"
(11) "Overall it was very easy to use your app for creating privacy policy."
(12) "Really cool thing for Privacy Policy creation"
(13) "I'll give it a try when I will release a personal project in about 4 weeks[.] It's a good tool, but in the past when I released a large number of apps under my name, I did not pay so much attention to the privacy policy. I was just using a template[.]"
(14) "I've found your app very useful. It is really easy to use."
(15) "I just expected it to run as a native app instead of opening a browser[.] Anyways, seems like a really cool app, congratulations[.] I will return to it if I ever need a new privacy policy[.]"
(16) "Awesome tool, help [sic] to avoid routine work."
(17) "I like this product and idea itself. I did not create many policies by myself but I've never heard/used something like your product. It can save a lot of time and help [...] many people, I think."
(18) "I like your product, and will consider using it in the future."
(19) "thanks for letting me know about PrivacyFlash Pro software, it can really help every Swift developer out there."
(20) "This looks awesome! Well done. Looking forward to giving this a try on a project I'm working on soon." [r/iOSProgramming]
(21) "[author name], congratulations to you and your team for the great work! PFP looks (and acts) more like a commercial product that [sic] a research prototype :)" [r/iOSProgramming]
(22) "Saved this one for later. I cancelled working on my first app because of privacy policies" [r/iOSProgramming]
(23) "This is pretty cool! Would it be possible to skip the code analysis and allow for creating a privacy policy by just filling out the questionnaire? I currently use Termly and I'd love to switch over to this and host it myself. Good job, y'all!" [r/iOSProgramming]
(24) "This is really cool! Are there any plans for React Native support?" [r/iOSProgramming]

[14] Note that GDPR provisions are included in PrivacyFlash Pro.