Technische Universität Braunschweig

IAS INSTITUTE FOR APPLICATION SECURITY

Informed Consent? A Study of “Consent Dialogs” on Android and iOS

Master’s Thesis

Benjamin Altpeter

May 04, 2022

supervised by Prof. Dr. Martin Johns

Declaration of Authorship

I hereby declare that the thesis submitted is my own unaided work. All direct or indirect sources used are acknowledged as references. I am aware that the thesis in digital form can be examined for the use of unauthorized aid and in order to determine whether the thesis as a whole or parts incorporated in it may be deemed as plagiarism. For the comparison of my work with existing sources I agree that it shall be entered in a database. Further rights of reproduction and usage, however, are not granted here. This paper was not previously presented to another examination board and has not been published.

Braunschweig, on May 04, 2022 Benjamin Altpeter

Contents

Abstract

Consent dialogs have become ubiquitous with seemingly every website and app pleading users to agree to their personal data being processed and their behaviour being tracked, often with the help of tens or even hundreds of third-party companies. They are an effort by website and app publishers to comply with data protection legislation like the GDPR, which imposes strict limits on how companies can process data. Previous research has established that companies often apply dark patterns to illegally nudge users into agreeing and that at the same time tracking is more common than ever with both websites and apps regularly automatically transmitting telemetry data.

But so far, there has been almost no research into consent dialogs on mobile. In this thesis, we study consent dialogs on Android and iOS in an automated and dynamic manner, analysing 4,388 popular apps from both platforms. We identify different types of consent elements in the apps and analyse their prevalence. We also identify dark patterns and violations by the apps based on a list of criteria for a legally compliant consent dialog that we have compiled. Finally, we measure the effect of the user's choice in the consent dialog by comparing the traffic from before any interaction with the traffic after accepting and rejecting the dialog and analysing contacted trackers and transmitted data types.
The results show that more than 90 % of consent dialogs implement at least one dark pattern and that a majority of apps transmits tracking data regardless of consent status.

1. Introduction

Your privacy is important to us!

We use cookies to improve our writing. By reading this thesis, you consent to our processing of your personal data for analytics purposes. To opt out of this processing, please stop reading now.

[OK]

Consent and cookie notices seem ubiquitous both on the web and on mobile. They are an effort by publishers to comply with data protection legislation like the *General Data Protection Regulation* (GDPR), an EU regulation that went into force in 2018 and mandates strict legal guidelines for processing personal data. The flood of consent dialogs is often wrongly attributed as a shortcoming of the GDPR, when in fact the GDPR and other applicable data protection legislation provide strong consumer protections and set a high bar for what an acceptable way of collecting consent is, that many consent dialogs fail to meet. The ubiquity of consent dialogs is a manifestation of the common belief that consent is the only possible legal basis under the GDPR, which is not true unless data is used for tracking, as well as companies deliberately violating the law, e.g. by making use of nudging and dark patterns meant to coerce the user into giving consent and discouraging them from opting out.

The danger of such dark patterns and nudging is well established [1–5], and on the web, there has already been pushback against these practices in the form of research identifying violations in consent dialogs [1; 6; 7] and consumer protection organisations like noyb bringing forward mass complaints against publishers using them [8]. Recently, in a joint decision with other EU data protection authorities, the Belgian authority ruled that the IAB *Transparency & Consent Framework*, an industry standard for collecting user consent and communicating consent status information to advertising and tracking scripts that is used on a majority of websites, violates the GDPR and is not able to acquire valid consent [9; 10].

At the same time, tracking is more common than ever not just on the web but also on mobile, with both Android and iOS apps regularly automatically transmitting telemetry data about device details (like model, settings, battery status, etc.), sensor data, events (which views are opened and buttons clicked?), or even the geolocation and which data is entered in the app [11–17]. Often, this data is sent to third-party companies and linked to the device's *advertising identifier*, a unique ID for a device that allows those companies to track users across multiple apps.

These practices are troubling from a data protection standpoint, not least since the GDPR also mandates strict legal guidelines for such data collection. According to those, any processing of personal data (meaning any data that can somehow be linked to a natural person, including pseudonymously) needs to have one of six possible legal bases. According to the data protection authorities, which are responsible for enforcing the GDPR in the EU, informed consent is the only applicable legal basis for tracking [18–20].
This is further amplified by the *ePrivacy Directive* (ePD), another EU law which mandates information, consent, and opt-out requirements for accessing information stored on a user's device even when no personal data is processed, and the European Court of Justice's *Schrems II* ruling, which invalidated the *Privacy Shield* adequacy decision that data transfers to the US were usually based on.

We are now seeing first efforts by mobile operating system providers to limit the tracking done by apps. In late 2020, Apple introduced privacy labels on iOS, which require apps to self-label the affected categories of data being processed and the purposes of the processing [21]. Since iOS 14.5, released in April 2021, Apple also provides users with a way to limit companies' ability to track them across apps: In order to gain access to the device's advertising ID (called *IDFA* on iOS), apps now need to ask the user's permission, otherwise they can only access the *IDFV*, another ID which is unique per device and app vendor and as such does not allow tracking across different vendors [22]. This was much to the dismay of advertisers and tracking companies having to face the reality that users do not want to be tracked, with one reporting opt-in rates as low as 2 % [23] and Facebook anticipating a loss of $10 billion in sales revenue as a result [24].
Since then, Google has also gone in a similar direction on Android, allowing users to opt out of the advertising ID through the settings [25] (rather than requiring opt-in as Apple did) and starting to roll out "data safety" labels in late April 2022 [26].

As previous research on consent dialogs has so far almost exclusively been limited to the web, in this thesis, we study consent dialogs on Android and iOS in an automated and dynamic manner, analysing 4,388 popular apps from both platforms.

To do so, we first identify different types of consent elements in the apps and analyse their prevalence. Then, we identify violations by the apps based on a list of criteria for a legally compliant consent dialog that we have compiled from applicable legislation, court rulings, and recommendations by the German and EU-wide data protection authorities. We look for apps that violate these criteria by using dark patterns. We also measure the effect of the user's choice in the consent dialog by comparing the traffic from before any interaction with the traffic after accepting and rejecting the dialog. For this, we identify the contacted trackers and transmitted data types in the recorded traffic. Finally, we also compare the recorded network traffic on iOS against the apps' privacy labels.

Contributions Through this thesis, we make the following contributions:

1. We present a comprehensive list of criteria that a consent dialog needs to meet in order to be compliant with applicable data protection regulation in the EU, compiled from the law, court rulings, and data protection authority recommendations.
2. We quantify the use of the IAB Transparency & Consent Framework in mobile apps.
3. We developed a device instrumentation framework for Android and iOS that can manage apps, set app permissions and extract app preferences, collect the device network traffic (including HTTPS and certificate-pinned traffic), as well as analyse and interact with elements displayed on screen, building on our previous research on the subject of data protection in mobile apps.
4. We make an automated way to reliably download large amounts of iOS apps as IPA files publicly available for the first time, based on extending an existing open source tool that we contributed our changes back to.
5. We present a method for automatically detecting consent elements in Android and iOS apps, as well as identifying dark patterns in detected consent dialogs. We further present a method for extracting the actual transmitted data from recorded network traffic using tracking endpoint-specific adapters.
6. We identify that in our dataset of 2,068 apps on Android and 2,320 apps on iOS, more than 90 % of consent dialogs implement at least one dark pattern and a majority of apps transmits tracking data regardless of consent status.

2. Legal Background

In this chapter, we introduce the legal framework that regulates data protection for mobile apps in the EU and Germany. We explain under which conditions data collection and processing are lawful or unlawful with a focus on data used for tracking purposes, as well as how consent dialogs fit into this.

2.1. Processing of Personal Data

The primary law for data protection in the EU is the *General Data Protection Regulation* (GDPR), which went into force in 2018 and mandates a data protection framework that is consistent across the whole EU and also more strict than previous laws in this area. As an EU regulation, it is applicable law in all EU member states without them needing to tranpose it into their national laws.

The GDPR deals with the *processing* of *personal data* (Article 2(1) GDPR[1]). These legal terms are defined in Article 4 GDPR:

> (1) 'personal data' means any information relating to an identified or identifiable natural person […]; [i.e.] one who can be identified, directly or indirectly, in particular by reference to an identifier such as a name, an identification number, location data, an online identifier or […] the physical, physiological, genetic, mental, economic, cultural or social identity […]
>
> (2) 'processing' means any operation […] performed on personal data […], such as collection, recording, organisation, structuring, storage, adaptation or alteration, retrieval, consultation, use, disclosure by transmission, dissemination or otherwise making available, alignment or combination, restriction, erasure or destruction

Both terms are explicitly very broad. In essence, any data that can somehow be connected to a natural person (the *data subject*) and that a company or other organisation (the *controller*) deals with in some way is covered by the GDPR. This even applies to *pseudonymous* data (Recital 26(2) GDPR), i.e. data that can only be attributed to a person using additional, separately kept information (Article 5(4) GDPR). Examples for data that can pseudonymously be linked to a person include user IDs and IP addresses. A user ID alone cannot identify a person. But that ID in combination with the database that stores the corresponding user information certainly can. Similarly, most companies will not be able to trace back an IP address to a person on their own. But in combination with the customer data held by the person's ISP, which can potentially be obtained through a court order in many jurisdictions, this is possible [28].
Conversely, data that cannot be linked to person, even when consulting additional information sources, is called *anonymous* data and does not fall under the GDPR (Recital 26(5) GDPR). This can include the settings of an application or aggregate statistical data. However, genuine anonymisation that cannot ever be reversed is hard to impossible to achieve. In many cases, even a handful of seemingly benign data points are enough to uniquely identify a person [29]. Similarly, fingerprinting can often uniquely identify a

[1]In this thesis, we cite legal norms separately from other sources in the style that is common in legal fields, following the GDPRhub style guide [27]. The references give the precise location to the cited portion of the law. The numbers in parentheses denote the specific numbered/lettered subparagraph or sentence within the norm, in hierarchical order.

device using its settings [30]. The GDPR requires taking that into consideration before classifying data as anonymous [31]. And even anonymous data can become personal data if linked with other personal data like a user ID.

Under the GDPR, processing personal data is generally prohibited unless there is a valid legal basis in the law for it. A conclusive list of those is defined in Article 6(1)(a–f) GDPR. A processing of personal data can only be lawful if it fulfils one of these conditions:

a) The data subject has given their consent.
b) The processing is necessary for performing a contract that the data subject is a party to.
c) The processing is necessary for the controller to fulfil a legal obligation they are subject to.
d) The processing is necessary to protect a person's vital interests.
e) The processing is necessary for an official task in the public interest, usually by public authorities.
f) The processing is necessary for a legitimate interest of the controller, given it outweighs the data subject's interests and fundamental rights.

In the context of tracking or other processing typically performed by apps, lit. c), d), and e) are obviously not applicable except in rare exceptions [20]. The GDPR itself does not answer which of the remaining three can be used. For that, one can look to the *data protection authorities* (DPAs). Those are public authorities in each member state tasked with enforcing the GDPR in their respective jurisdiction (Article 51 GDPR). They publish their interpretation of unclear aspects of the law like this in recommendations. While those recommendations are not legally binding in and of themselves, the DPAs can issue sanctions (including fines) to companies who do not follow them (Article 58 GDPR).
According to the data protection authorities, lit. b) and f) can typically not be used as a legal basis for tracking, either [18–20]. That leaves only consent as a potential legal basis for tracking.

In any case and regardless of the type of processing and legal basis used, controllers need to comply with the general principles set forth in the GDPR. Notably, Article 5(1) GDPR requires that controllers adhere to the principles of *data minimisation* by only processing data to the extent necessary for the particular purpose (lit. c), and *storage limitation* by only keeping stored data in a form that permits the identification of data subjects for at most as long as necessary (lit. e). This is further emphasised by Article 25(1) GDPR, which prescribes the principle of *data protection by design and by default.*

As an EU law, the GDPR is binding for all companies in the EU. But its territorial scope, as defined in Article 3 GDPR, also includes companies outside the EU if they deliberately process the data of people in the EU related to the offering of goods and services, or for the monitoring of their behaviour, as is the case with tracking [32].

2.2. Storing and Accessing Information on Terminal Equipment

Another law to consider is the *ePrivacy Directive* (ePD), which went into force back in 2002 and was last amended in 2009.
As a directive, it is not directly legally binding but needs to be transposed into national law by the member states.

In the context of this thesis, only Article 5(3) ePD is relevant:

> Member States shall ensure that the storing of information, or the gaining of access to information already stored, in the terminal equipment of a subscriber or user is only allowed on condition that the subscriber or user concerned has given his or her consent, having been provided with clear and comprehensive information […]. This shall not prevent any technical storage or access for the sole purpose of carrying out the transmission of a communication over an electronic communications network, or as strictly necessary in order for the provider of an information society service explicitly requested by the subscriber or user to provide the service.

Unlike the GDPR, Article 5(3) ePD does not deal with data protection but protects the integrity of a person's terminal equipment. It doesn't just cover personal data but *any* data that is read from or stored on a user's device [33]. While this provision is most commonly discussed in the context of cookies (earning the ePD its informal nickname of "the cookie law"), it also applies to any other reading or storing of information on a user's device, for example through fingerprinting or tracking pixels [34; 35].

Also unlike the GDPR, Article 5(3) ePD does not provide multiple possible legal bases that could apply. There are only two options: Either the reading or storing falls under the exception of being strictly necessary (a clause that has to be interpreted narrowly, with for example tracking and advertising not being strictly necessary despite many publishers' insistence [36]), or it requires prior informed consent by the user.

For a long time, Germany had not actually properly implemented Article 5(3) ePD into national law. The national implementation in § 15(3) TMG (*Telemediengesetz*) instead directly contradicted the ePD, allowing an opt-out mechanism for cookies [37]. In its October 2019 Planet49 decision, the European Court of Justice ruled that § 15(3) TMG had to be interpreted in line with Article 5(3) ePD, even against the explicit wording of that provision [33; 38].

This finally resulted in Germany changing the law. Since December 2021, Germany implements Article 5(3) ePD through the *Gesetz zur Regelung des Datenschutzes und des Schutzes der Privatsphäre in der Telekommunikation und bei Telemedien* (TTDSG). In particular, § 25 TTDSG defines privacy protections for terminal equipment. § 25(1) TTDSG mandates that information may only be stored on or accessed from a user's terminal equipment if the user has given consent based on clear and comprehensive information, which is to be provided in accordance with the GDPR. § 25(2) TTDSG then provides two exceptions to that rule, namely if (1) the sole purpose of the storing or access is to carry out the transmission of a communication over a public telecommunications network, or if (2) it is necessary to provide a telemedia service that is explicitly requested by the user.

As such, neither Article 5(3) ePD nor § 25 TTDSG allow using a contractual necessity or legitimate interest of the controller as a legal basis, placing even stricter conditions on the exception of not needing consent than the GDPR.

2.3. Transferring Personal Data to Third Countries

Finally, we need to consider the case of transferring data to a country outside the EU (a so-called *third country*). The GDPR generally forbids this, unless there is an exception in the law that allows it (Articles 44 – 50 GDPR).

Most simply, transfers to third countries can be based on an *adequacy decision*, which is one of those exceptions (Article 45 GDPR). As of the time of writing, the European Commission has issued such adequacy decisions for the following countries: Andorra, Argentina, Canada, Faroe Islands, Guernsey, Israel, Isle of Man, Japan, Jersey, New Zealand, Republic of Korea, Switzerland, the United Kingdom, and Uruguay [39].
With an adequacy decision, the European Commission attests an adequate level of data protection to the respective country considering the standards of the GDPR. Based on that, data transfers to those countries can happen without any special additional safeguards.

Previously, there was also such an adequacy decision for the United States (where most tracking providers, and many other internet infrastructure companies are based), the *Privacy Shield*. However, the Privacy Shield was invalidated by the European Court of Justice in its July 2020 Schrems II ruling due to the overbearing US surveillance programs and lack of effective judicial remedies in the US for data subjects from the EU [40; 41], just as its predecessor *Safe Harbour* was invalidated in the European Court of Justice's October 2015 Schrems I ruling [42], both over data transfers from Facebook Ireland to the US[2].

Thus, legal data transfers to the US are now significantly harder or even impossible in many cases and at the very least require additional safeguards by the controller [48–51].

2.4. Consequences for Apps

In summary, we have seen that apps need the user's consent for tracking. If the tracking data includes device identifiers like the advertising ID, the transmitted data becomes pseudonymous and thus personal data that falls under the GDPR. And while anonymous data is not covered by the GDPR, if it was accessed from the user's device without being strictly technically necessary, it falls under Article 5(3) ePD.

To obtain said consent, apps can make use of consent dialogs that prompt the user to agree to the use of their data for certain purposes. However, there are high conditions for what is considered valid consent. We will discuss these criteria in detail in the next chapter. Consent dialogs need to meet them, otherwise the supposed consent they collect is invalid, meaning that it cannot be used as a legal basis. If an app still performs tracking under these circumstances, it is in violation of applicable law.

Mere notices that only inform the user of the processing going on without actually giving them a choice in the matter cannot obtain the user's consent on the other hand. They can only be used by a controller to fulfil their information obligations under Articles 12–14 GDPR.

Finally, transfers of data to the US are very difficult to implement in a legal way in the current absence of an adequacy decision, but the automatic verification of that is beyond the scope of this thesis.

[2]A new "Trans-Atlantic Data Privacy Framework" is in the works as of the time of writing, with an "agreement in principle" having been reached between the EU and US but few details having been published and no actual legal documents having been written yet [43; 44]. The announcement has been met with cautious optimism by the European Data Protection Board and European Data Protection Supervisor [45; 46], while Max Schrems, the lead litigant behind the Schrems I and II cases questions whether the new deal actually contains any meaningful changes compared to the previous deals [47]. Regardless, for the time being, controllers cannot rely on an adequacy decision for US transfers.

3. Criteria for Compliant Consent Dialogs

Before we start to look at actual consent dialogs in the wild, in this chapter, we present a list of criteria a consent dialog needs to meet in order to be legally compliant. For this purpose, we consider two main legal sources: criteria which are set in actual law, and therefore legally binding, and criteria from recommendations of the data protection authorities (DPAs), which are not directly legally binding but rather echo the DPAs' interpretation of the law. Some of the criteria from the DPAs have also already been confirmed by court rulings.

In this thesis, we only consider EU-wide and German sources of law, though DPAs from other EU countries have also issued similar guidance and/or decisions (cf. e.g. [52–55]).

3.1. Conditions on Consent from the Law

Any law that restricts the data processing done in apps in any way can in principle introduce criteria on how to obtain consent, and thus has to be considered here. This most obviously includes all laws already discussed in Chapter 2: the GDPR, the ePD, and its national implementation in Germany, the TTDSG.

In addition, the GDPR has *opening clauses*, which allow the member states to introduce national laws that diverge from the GDPR in limited aspects. Germany has made use of these opening clauses in the BDSG (*Bundesdatenschutzgesetz*), which thus also needs to be considered.

However, in actuality, most of these laws do not introduce their own conditions on consent:

- Article 5(3) ePD delegates to Directive 95/46/EC for how consent has to be implemented. This directive was the predecessor to the GDPR, and has been replaced by it. According to Article 94(2) GDPR, all references to this repealed directive in previous legislation shall be construed as references to the GDPR.
- § 25(1) TTDSG delegates directly to the GDPR for how consent has to be implemented.
- The BDSG only talks about consent in the context of law enforcement (§ 51 BDSG in combination with § 45 BDSG) and is thus not relevant here.

This leaves the GDPR as the only law that defines applicable conditions for consent dialogs.

Consent is one of the six possible legal bases for processing personal data from Article 6(1) GDPR. Unsurprisingly, processing that can only rely on consent as a legal basis (like tracking), may thus only happen *after* consent has been given, and the controller needs to be able to demonstrate that consent has been given (Article 7(1) GDPR).

Consent itself is defined in Article 4(11) GDPR, which lists a set of basic conditions an action needs to meet in order to be considered consent, with each being further specified by the recitals to the GDPR:

freely given Consent is not *freely given* if there is a clear imbalance between the data subject and the controller, particularly in the case of public authorities (Recital 43 GDPR). The data subject needs to have a genuine and free choice to refuse (or later withdraw) consent without detriment (Recital 42 GDPR). The provision of a contract or service cannot require a data subject's consent if such consent is not necessary for the performance thereof (Article 7(4) GDPR; Recital 43 GDPR).

specific For consent to be *specific*, separate consent should be asked for different processing purposes (Recital 32 GDPR).

informed To be *informed*, a request for consent needs to contain at least the identity of the controller and the purposes of the processing (Recital 42 GDPR).

unambiguous Consent is *unambiguous* if the request for it is "clear, concise and not unnecessarily disruptive to the use of the service for which it is provided" (Recital 32 GDPR).

statement or clear affirmative action Silence, pre-ticked boxes, or inactivity do not constitute consent (Recital 32 GDPR). Instead, it has to be given by a statement "which clearly indicates [...] the data subject's acceptance of the proposed processing", like ticking a checkbox.

Article 7 GDPR then lists a number of additional conditions for consent:

- If a data subject is asked to give consent through a declaration that also concerns other matters (e.g. also needs to accept a company's terms of service at the same time), the request for consent needs to be "clearly distinguishable from the other matters, in an intelligible and easily accessible form, using clear and plain language" (Article 7(2) GDPR).
- The data subject needs to be able to withdraw consent at any time (Article 7(3) GDPR).
- Before giving consent, the data subject needs to be informed that they have the right to withdraw their consent at any time (Article 7(3) GDPR).
- Later withdrawing consent needs to be as easy as giving it in the first place (Article 7(3) GDPR).

In addition to that, the GDPR places even stricter conditions on consent for special categories of personal data (this includes, among other things, political opinions, biometric and genetic data, as well as data on a person's health and sex life, Article 9 GDPR) and third-country transfers without an adequacy decision (Article 49(1)(a) GDPR), requiring an express statement, separate for this specific purpose [56].
Children under the age of 16 years cannot give consent themselves, it instead needs to be given or authorised by their legal guardians (Article 8(1) GDPR).
We do not consider either in this thesis as we cannot reliably detect them automatically.

3.2. List of Criteria

Most of the conditions extracted directly from the GDPR are somewhat vague, making them both hard for companies to implement and difficult to check for automatically, as planned for this thesis. To alleviate this issue, the data protection authorities publish recommendations which detail their interpretation of the law and provide specific guidelines on how to follow them. In most cases, these specific guidelines are better suited for verification in specific cases and are also used in similar research [7; 57; 58].

For this thesis, we have searched all current publications regarding consent and adjacent topics from all German state data protection authorities, as well as the national DPA, the German data protection conference (*Datenschutzkonferenz*, a council of the German DPAs that develops unified recommendations), and the European Data Protection Board (an EU body tasked with ensuring consistent application of data protection law across the EU) for criteria on consent dialogs. The list below consolidates the criteria from the GDPR and DPA recommendations. Where the criteria have already been confirmed by courts, we cite those rulings as well.

It should be emphasised that *any* violation against even a single one of these criteria results in all data processing based on that supposed consent being illegal.

3.2.1. Criteria for Underlying Circumstances

- Consent needs to be given through a clear, affirmative action like clicking a button or ticking a checkbox (Recital 32 GDPR) [19, no. A.4.2; 20, p. 13; 33; 56, para. 80; 59; 60].
- Processing that needs to rely on consent may only happen after consent has been given (Article 7(1) GDPR) [19, no. B.1.1.1; 20, p. 12; 56, para. 90; 60–62].
- Consent has to be voluntary, i.e. it needs to be possible to use the app without consenting (Recital 42 GDPR) [19, no. A.4.2; 63].
- It must be possible to later withdraw consent at any time and this has to be as easy as giving consent in the first place. The data subject needs to be informed of this before giving consent. (Article 7(3) GDPR) [19, no. B.1.6; 20, p. 18; 56, para. 114; 60]
- A consent dialog may not make it impossible to access other required legal notices (like contact information or privacy policy) [19, no. A.4.1].

3.2.2. Criteria for Wording and Design of Consent Dialogs

- The consent dialog needs to have a clear heading that accurately describes the impact of the processing on the data subject, like "Data disclosure to third parties for tracking purposes." Vague headings like "We respect your privacy." are not sufficient. [19, no. B.1.3.7]
- The "consent" button cannot be highlighted compared to the "refuse" button (e.g. by making it bigger or using a more prominent colour for it) [19, no. A.4.3; 59; 60; 64; 65].
- The consent notice must be in the language of the country it addresses [19, no. B.1.3.1].
- The consent notice cannot be overly long or complex [19, nos. B.1.3.3.3, B.1.3.3.4].
- A consent notice needs to be clearly distinguishable from other matters like regular terms of service (Article 7(2) GDPR) [19, no. B.1.5.1; 56, para. 81].
- A consent notice that only mentions cookies can only receive consent under the ePD, not the GDPR [19, no. B.1.3.5.1; 20, p. 9].

3.2.3. Criteria on Information to Include in Consent Dialogs

- The consent notice needs to contain at least the following details (Recital 42 GDPR) [19, no. A.4.2; 20, p. 12; 33]:
  - Who is the controller?
  - What is the purpose of the processing?
  - If cookies are used, what is their duration?
  - Is there any access for third parties?

- Third-party recipients have to be mentioned explicitly [19, no. A.4.2; 60].
- The consent notice needs to list concrete purposes, vague wordings like "to improve user experience" are not sufficient [19, no. A.4.2; 20, p. 16; 60].

3.2.4. Criteria for Buttons and Interactive Elements in Consent Dialogs

- Refusing consent has to be possible through inaction or with the same number of clicks as consenting [19, no. A.4.3; 20, p. 14; 60; 64].
- A button with the text "Okay" does not sufficiently convey that clicking it is supposed to agree to the consent dialog, and thus does not result in valid consent [19, no. B.1.3.12.1; 20, p. 14; 60].
- It is not possible to receive consent on a page that doesn't include all necessary details (e.g. if they are hidden behind another link, or on a page deeper in the consent flow) [20, p. 14; 60].
- It needs to be possible to only consent to adequate subpurposes and/or recipients (Recital 32 GDPR) [19, no. A.4.2; 20, p. 16; 56, para. 42].
- No purposes may be pre-selected (Recital 32 GDPR) [19, no. A.4.2; 33; 59].
- Clicking an "Accept all" button may not toggle additional, previously unselected purposes [60].
- A consent dialog that saves consent but not refusal thereof (and is thus displayed over and over again when refused) is not compliant [19, no. B.2.2.3].

4. Consent Dialogs in the Wild

In this chapter, we explore how consent dialogs are actually implemented in the wild, both on the web and on mobile, to gain insights on which approach to use for the analysis in Chapter 6.

4.1. Situation on the Web

It makes sense to first take a look at consent dialogs on the web because most prior research [1; 6; 7; 66] on the topic focusses exclusively on the web.

4.1.1. Consent Management Platforms

Various companies offer *consent management platforms* (CMPs), off-the-shelf solutions that website operators can embed into their site under the promise that they will ensure legal compliance for tracking and similar processing [67–70]. Using a CMP means that individual websites don't have to implement consent dialogs themselves anymore. As such, CMPs are also conducive to research as one now only needs to handle a handful of CMPs instead of custom implementations on every website.

Theoretically, CMPs should also make it easier for websites to follow the law as the CMP is responsible for ensuring that processing that requires consent (like tracking or setting corresponding cookies) only happens after consent has been given by the user. However, most CMPs are highly configurable and allow the website operators to enable behaviour that violates the law; sometimes such settings are even the default [71].

On the web, usage of CMPs is common. Recent research detected the use of CMPs on between 6 % and 13 % of European websites, depending on their Tranco rank [6; 72]. Ad tech company Kevel reports the presence of a CMP on 44 % of the top 10k US sites for Q1 2022 [73].

4.1.2. IAB Transparency & Consent Framework

Websites often use third-party scripts from many different vendors for all kinds of purposes, including advertising and tracking. Many of these require consent (see Chapter 2). How do websites make sure that the scripts only start processing after the required consent has been given? Without a common framework, either each CMP would have to implement custom handlers for each possible third-party script or each third-party script would need to know how to communicate with all possible CMPs.

IAB Europe, an association that represents the interests of the digital advertising and marketing industry in Europe [74], maintains the *Transparency & Consent Framework* (TCF), a standard that defines a common interface for CMPs and third-party scripts to communicate. It defines how websites should store consent and legitimate interest records, as well as conditions on how to prompt the user for consent and inform them through consent dialogs. For that, IAB Europe maintains a list of purposes and vendors the website can ask users to consent to. The first version, v1.1[1], was launched in April 2018, shortly before the GDPR went into force, and in August 2019, a revised v2.0 was published, with v1.1 now being deprecated [75]. Most CMPs implement the TCF [76–79]. Note that in February 2022, the Belgian Gegevensbeschermingsautoriteit found the TCF to violate the GDPR in a joint decision with other DPAs but gave the IAB a grace period of six months to implement the necessary changes [10].

The TCF distinguishes between the publisher (the entity running the website), the CMP (the entity providing the CMP—this can also be a publisher's in-house CMP if it is registered with IAB Europe), and vendors (the third parties embedded in the publisher's website, e.g. ad or tracking providers). Both CMPs and vendors need to register with IAB Europe, for which there is an annual fee of 1,500 € [80].

On a technical level, the TCF mainly regulates two aspects. For one, it mandates that CMPs store consent records in a so-called *TC string*, which encodes[2] the following information [81]:

- Metadata: TCF version the TC string is based on, last update time
- Consent records: per purpose and per vendor
- Legitimate interest records: objections to them by the user[3]
- Publisher restrictions: allow the publisher to specify custom requirements to restrict how vendors may process personal data
- Publisher transparency and consent data: allows the publisher to store consent and legitimate interest data for their own purposes
- Jurisdiction data: country publisher is based in

TC strings may only be created or changed by the CMP [81], and the CMP is free to choose where and how to store the TC string [82].

Secondly, it defines mechanisms for the different parties to communicate. For that, the CMP must expose API commands through the `__tcfapi(command, version, callback, parameter)` function on the global `window` object [83], most notably a command to receive an object representing the parsed TC string, one to check whether the CMP has finished loading and whether it believes the GDPR applies in the context of this visit[4], and another one to register an event listener for changes to the TC string.

If a script then wants to start processing, it has to [83]:

1. Determine whether a CMP is loaded on the page by checking for the presence of the `__tcfapi()` function. If no CMP is loaded, it has to assume that there is no consent or legitimate interest.

   If the script is loaded in an iframe instead of directly on the page (common for ads), it can use a `postMessage()` API for the same purpose.

2. Request the TC data, and check whether a legal basis for the desired processing is available. Only if that's the case, may it start processing.

[1]Version 1.0 was never published.

[2]IAB Europe offers a JavaScript library (`https://github.com/InteractiveAdvertisingBureau/iabtcf-es`) and an online tool (`https://iabtcf.com`) to decode and encode TC strings.

[3]The fact that the TCF allows a publisher to use a legitimate interest for a certain processing does not necessarily mean that this is legal [57].

[4]The TCF does not mandate how the CMP is supposed to determine whether the GDPR applies, it only mentions using the user's geolocation as one option. Vendors are required to adhere to the CMP's determination. [83] However, as explained in Section 2.1, the user's location alone is not necessarily sufficient for determining whether the GDPR applies.

3. Subscribe to change events to notice if the user withdrew their consent for example and then stop processing accordingly.

While not the TCF's intended purpose, the fact that the CMP data can be easily read programmatically through the API that has to be provided has also enabled research on consent dialogs on the web [6; 72; 84], and is even used by consumer protection organisations to automatically find violations in popular websites to pursue legal action against [8].

4.2. Situation on Mobile

In this section, we look at whether mobile apps also commonly implement the TCF or at least use a limited number of CMPs, which would make the desired analysis easy.

4.2.1. Use of IAB TCF

Version 2.0 of the IAB TCF also supports CMPs in mobile apps on Android and iOS [83]. As native apps do not support running JavaScript code, a different API mechanism is used here. CMPs must instead store consent details (including the TC string) via the platform-specific per-app storage interfaces, namely `SharedPreferences` on Android and `NSUserDefaults` on iOS. Vendor SDKs can also access these stores to read the TCF data and set listeners to be informed of updates. Thus the same steps as on the web can be used to determine whether an SDK can process the user's data on mobile, just with a slightly different implementation.

The TCF mandates the keys to be used for storing the consent details, most importantly [83]:

- `IABTCF_PolicyVersion`: The version of the TCF that is the basis for these properties.
- `IABTCF_gdprApplies`: The CMP's determination of whether the GDPR applies to the particular use of the app, same as on the web, with `0` meaning that the GDPR does not apply, `1` meaning that the GDPR does apply, and an unset value meaning that the CMP has not (yet) determined this.
- `IABTCF_TCString`: The TC string containing all consent information as explained in the previous section.
- Various other properties containing values already represented in the TC string but in parsed form to avoid SDKs needing to parse the string themselves.

For analysis purposes, it is necessary to read those properties from outside the app. That is possible, both for Android and iOS, using *Frida*[5], a dynamic instrumentation toolkit for native apps that allows the user to inject JavaScript code into running applications and thus interact with native functions and runtime memory [85].

On Android, the `SharedPreferences` of an app can be accessed by injecting Frida and running the following script:

```
var app_ctx =
    Java.use('android.app.ActivityThread').currentApplication().getApplicationContext();
```

[5]`https://frida.re/`

```
var pref_mgr =
    Java.use('android.preference.PreferenceManager')
        .getDefaultSharedPreferences(app_ctx);
console.log(pref_mgr.getAll());
```

Similarly, on iOS the `NSUserDefaults` of a running app can be accessed using the following Frida script:

```
var prefs = ObjC.classes.NSUserDefaults.alloc().init();
console.log(prefs.dictionaryRepresentation());
```

In both cases, for automated analysis, the objects would still need to be converted from their respective platform-native representation to a common format, but that step is omitted here for brevity.

Unlike on the web, no prior research on how common TCF usage is on mobile exists as far as we are aware. To gauge whether reading the TCF preferences is a viable approach for this thesis, we performed a simple analysis on 823 apps from a dataset of popular Android apps from November 2021: The apps were run in the Android emulator and left running for five seconds. Afterwards, a screenshot was taken and the `SharedPreferences` of the respective app were saved. Then, we manually looked at the screenshots to determine whether the app showed any consent element (like a consent dialog, a privacy notice, or even just a link to a privacy policy).

181 of the 823 apps (21.99 %) displayed such a consent element on screen after five seconds. However, only 21 of the 823 apps (2.55 %) had set a corresponding privacy-related preference (i.e. one with a key that includes `IABTCF` or `GDPR`). This suggests that the TCF is not commonly implemented on mobile.

4.2.2. Use of CMPs

Even though it seems like mobile apps do not tend to make use of the TCF, it would still be possible that they use off-the-shelf CMPs that just don't implement the TCF. To find out whether that is actually the case, we ran a simple static analysis to detect the presence of common CMP libraries in Android and iOS apps.

On Android, we use the same approach that the *Exodus Privacy*[6] project uses for checking for the presence of tracking libraries [86]. We run the `dexdump` tool on the APK file, which is the Android counterpart to the Linux `objdump` tool and can statically extract class and method names from an APK, among other things. Then we compare the namespaces of the listed classes to a list of CMP libraries. For example, the classes of the Didomi CMP library are in the `io/didomi` namespace. If we detect classes with this namespace in an app, we can assume that it uses the CMP[7].

On iOS, we are not aware of any similar work. It is however possible to list the shared libraries in an IPA file using the `otool` command [87], where the lines starting with `@rpath` are the libraries included in the IPA (lines without this prefix are system libraries). The symbol table can be listed using the `nm` and `symbols` commands [88]. All of these tools only run on macOS.

[6] `https://exodus-privacy.eu.org/`

[7] Of course, this approach is very fuzzy. Even if an app includes a CMP library, that does not mean it actually uses it. In addition, it is possible that the list of namespaces we use is not strict enough and matches other non-CMP libraries. None of that is a problem for the purposes of this analysis, though. Its goal is only to provide an upper bound on the CMP usage in mobile apps and to evaluate whether it is viable at all to rely on CMP-specific code for this thesis.

Nonetheless, the required functionality can be replicated on any operating system: An IPA file is just a ZIP archive. All libraries included in an IPA are in subdirectories of `/Payload/<app name>.app/Frameworks`. We simply list those directories and compare them against a list of CMP libraries.

For this analysis, we compiled a list of identifiers for the names of 18 CMPs (based on [89–91]). For Android, we used the same dataset as in Section 4.2.1, but ran the analysis on all 3,271 apps. For iOS, we used a dataset of 1,001 apps from the App Store top charts from May 2021.

We detected a potential CMP use in 234 of the 3,271 Android apps (7.15 %) and 28 of the 1,001 iOS apps (2.8 %). We also checked for the presence of IAB's TC string library (`com/iabtcf`), which is only available for Android. We detected that in 38 of the 3,271 apps (1.16 %). This suggests that mobile apps do not commonly use off-the-shelf CMPs either, especially given that the simple analysis we performed is even an overapproximation.

4.2.3. Consequences for Analysis

As established, we can neither rely on the TCF, which would have provided a standardized and machine-readable way to detect the presence of a CMP in an app, read the settings of CMPs, and even interact with them, nor on custom adapters for the internal implementations and dialog design characteristics of a limited number of off-the-shelf CMP solutions. As such, we need to use a much more general approach that detects and works with any kind of CMP, regardless of implementation details. This in turn necessarily means a loss in the amount of details we can extract from CMPs, as we have to expect a large amount of completely different implementations. It also means that we will likely miss some consent elements.
The details of the method we use for the analysis are described in Chapter 6.

For those consent dialogs that *do* implement the TCF, we still extract the data from `NSUserDefaults` or `SharedPreferences` and perform an analysis on that.

5. Device Instrumentation Framework

We developed a framework for automated instrumentation of Android and iOS apps for this analysis, using an emulator on Android and a physical device on iOS. The framework can manage (install, run, and uninstall) apps, set app permissions and extract app preferences, collect the device network traffic (including HTTPS and certificate-pinned traffic) while an app is running, as well as analyse and interact with elements displayed on screen. It builds on and extends previous work on data protection in mobile apps that we conducted and participated in [16; 17].
The framework is written in TypeScript running on Node.js. The full source code is available on GitHub: `https://github.com/baltpeter/thesis-mobile-consent-dialogs`

We also present an approach to extract top chart data from the Play Store on Android and App Store on iOS, as well as download the corresponding apps.

5.1. Traffic Collection

To record the apps' network traffic, we use *mitmproxy*[1], an open-source set of proxy tools written in Python, which can already deal with HTTPS traffic given its certificate authority is installed in the system certificate stores [92]. However, we also need to deal with apps that implement certificate pinning. For that, we use *objection*[2] on Android and *SSL Kill Switch 2*[3] on iOS. These hook known certificate pinning functions provided by the OS itself, in common libraries, and even some custom implementations in apps. Nonetheless, it is still possible that apps use other pinning mechanisms not recognized by them. In those cases, we will miss the corresponding requests.

We execute apps for one minute and record their traffic in the meantime. For that, we use a mitmproxy add-on that stores the encountered requests, as well as their headers and cookies, in a PostgreSQL database. After the one-minute timeout has expired, we check whether the app is still running and discard the results otherwise. This is necessary because some apps quit immediately after launch, either because they detected that the device we are using is rooted/jailbroken and they consider that a security problem (often the case for banking apps, for example), or because of problems with the certificate pinning bypass by objection on Android (see Section 8.2).

Using mitmproxy, we can only record the whole device's network traffic and not an individual app's traffic. Both Android and iOS regularly send requests to Google and Apple, respectively, in the background, doing connectivity checks, time synchronisation, and their own tracking for example[4]. We need to filter out this background noise. To do so, we recorded the network traffic from idle devices for several days and created an SQL view to filter out all requests from these runs. For some endpoints, it is difficult or even impossible

[1]`https://mitmproxy.org/`
[2]`https://github.com/sensepost/objection`
[3]`https://github.com/nabla-c0d3/ssl-kill-switch2`
[4]Apple at least provides a list explaining most of the background requests: `https://support.apple.com/en-us/HT210060`

to figure out whether a request was caused by an app or by the OS, e.g. when both Android itself and apps use the same Google trackers. In those cases, we have opted to rather remove too much than too little traffic to avoid wrongly attributing OS traffic to apps. The full list of filters we employ can be seen in Appendix A.2.

5.2. Device Management Automation

For managing the devices, we have implemented platform-specific interfaces that provide the following common commands for both Android and iOS:

Ensure device On Android, this starts the emulator and waits for it to finish booting. For that, it reads the boot animation state using `adb -e shell getprop init.svc.bootanim` and waits until that returns `stopped` [93].
We also run this command when the emulator has stopped responding (this seems to be caused by emulator bugs that happen with some apps). Thus, it first stops a potential previous running emulator before starting a new one.

On iOS, this is a no-op since we are using a physical device that we cannot start automatically as the jailbreak requires manual steps (in particular, putting the device into DFU mode). The iPhone has to be started and jailbroken before the analysis is run (see Section 5.4).

Reset device On Android, this uses the emulator's snapshot features [94] to restore the emulator into a clean state by loading a predefined snapshot.
After that, it checks whether Frida is already running and starts it otherwise.

On iOS, this is once again a no-op since it is not possible to reset the device automatically either. Instead, we rely on the *uninstall app* and *clear stuck modals* commands to put the device into a clean state again.

Clear stuck modals Sometimes, modals or opened browser windows are stuck on the screen after uninstalling an app. On Android, we can get rid of those by pressing the back button followed by the home button. This is done through the `adb shell "input keyevent $event_number"` command, where `4` is the event number for the back button, and `3` the one for the home button [95].

On iOS, we use *Activator*[5] and the `activator send libactivator.system.clear-switcher` and `activator send libactivator.system.homebutton` commands [96]. We send those to the device through SSH.

Install or uninstall app Nowadays, many apps on the Play Store are distributed as *Android App Bundles*. This means that the app is not bundled as a single `.apk` file but instead consists of multiple `.apk` files, so-called *split APKs*, to allow Google to optimize app delivery for different devices [97]. Split APKs can be correctly installed by passing all their parts to the `adb install-multiple` command. We also pass the `-g` flag which grants all permissions to the app [98].
To uninstall apps, we use `adb shell "pm uninstall --user 0 $appId"` [99]. In the case of an error, unlike `adb uninstall`, this allows us to determine whether there was an actual problem or the app just wasn't installed to begin with.

[5]https://cydia.saurik.com/package/libactivator/

On iOS, we use the `ideviceinstaller --install` and `ideviceinstaller --uninstall` commands from *libimobiledevice*[6], an open source cross-platform library for interacting with iOS devices.

Set app permissions As we want to find out all data apps would transmit if given the opportunity, we need to give them all permissions. Granting an operating system permission only refers to the local device access the apps have and not to data protection, thus it cannot be considered consent under the GDPR or ePD.

On Android, we have already granted all runtime permissions through the `-g` flag to `adb install`. This does not, however, include so-called dangerous permissions like reading phone numbers or SMS messages. To grant those, we first obtain a list of dangerous permissions by using the command `adb shell "pm list permissions -g -d -u"`, and then grant them individually using the command `adb shell "pm grant $appId $permissionId"` [100].

On iOS, there is no intended way to grant permissions other than through the GUI. We discovered that permission data is stored in the `access` table of an SQLite database that is located at `/private/var/mobile/Library/TCC/TCC.db`. A list of possible permission IDs can be reconstructed by using `/System/Library/PrivateFrameworks/TCC.framework/en.lproj/Localizable.strings`, a translation file for the *Transparency, Consent, and Control* framework [101]. Setting the `auth_value` column to `2` grants the permission, setting it to `0` denies it. A list of which permissions we set can be found in Table A.1 in Appendix A.1.

The location permission is not handled by that table. To grant an app access to the location, we inject the following Frida script into the settings application:

```
ObjC.classes.CLLocationManager
    .setAuthorizationStatusByType_forBundleIdentifier_(
        authorization_status, app_id
    );
```

The possible values for the authorization status are: `0` (ask every time), `2` (never), `3` (always), and `4` (while using the app)[7].

Set clipboard To seed the clipboard, we again use Frida scripts. On Android, we inject the following script into the launcher process (`com.google.android.apps.nexuslauncher`):

```
var app_ctx = Java.use('android.app.ActivityThread').currentApplication()
    .getApplicationContext();
var cm = Java.cast(
    app_ctx.getSystemService("clipboard"),
    Java.use("android.content.ClipboardManager")
);
cm.setText(Java.use("java.lang.StringBuilder").$new(text));
```

On iOS, we inject the following script into the `SpringBoard` process:

```
ObjC.classes.UIPasteboard.generalPasteboard().setString_(text);
```

[6]`https://libimobiledevice.org/`

[7]A value of `1` would mean that the status is restricted and cannot be changed by the user, e.g. due to parental controls [102].

Start app On Android, we start apps through objection. That way, we can enable early instrumentation for the certificate pinning bypasses and ensure we do not miss any requests [103].

On iOS, SSL Kill Switch 2 is running system-wide and doesn't need to be manually injected into apps. We use the *Open* package[8], which enables us to start apps from the command line using the `open` command, which we do via SSH.

Reset app We need to run apps with dialogs multiple times so we can capture the different network traffic after accepting and rejecting them. This command also prepares the app for analysis.

The steps for that are the same on both platforms as they rely on the other commands:

1. Uninstall the app if it was previously installed. This will also clear all app data and settings.
2. Install the app.
3. Set the desired app permissions.
4. Clear any stuck modals.
5. Seed the clipboard to a known value.
6. Start the app.

Get app preferences This fetches the app's preferences from the operating system's per-app storage interfaces (`SharedPreferences` on Android, and `NSUserDefaults` on iOS) by injecting the Frida scripts described in Section 4.2.1 and converting their results to JSON. The preferences contain the IAB TCF data.

Get app version On Android, this returns an APK's version by running `aapt dump badging $apk_path` and extracting the `versionName` field [104].

On iOS, we use the *ipa-extract-info*[9] library to parse the IPA files and then read the version from `$.info.CFBundleShortVersionString`.

5.3. App Instrumentation

For reading and interacting with elements on the screen, we leverage *Appium*[10], an open source test automation framework. Appium provides a common interface over platform-specific user interface testing APIs, in particular *UiAutomator2* on Android and *XCUITest* on iOS [105]. This way, we can use the same code for both platforms.

We disable several of Appium's features, namely automatically starting apps, waiting for their launch, and resetting, as those are intended for software testing and not flexible enough for the purposes of this thesis. Instead, we use our own platform APIs described in Section 5.2.

The Appium docs previously mentioned that Appium does not work with jailbroken iOS devices, though we found that to be wrong[11]. We were however not able to rely on Appium's automatic configuration for iOS and had to resort to using the fully manual configuration [106]. We additionally had to pass the

[8]`http://cydia.saurik.com/package/com.conradkramer.open/`
[9]`https://github.com/nowsecure/ipa-extract-info`
[10]`http://appium.io/`
[11]We have submitted a correction and the docs have since been changed: `https://github.com/appium/appium/issues/16211`

`-allowProvisioningUpdates` flag to `xcodebuild`, which the docs previously did not mention[12]. This may be due to the fact that we are using a free Apple developer account.

Further, we noticed that a persistent Appium server tends to break after a few runs on iOS. This can be mitigated by restarting the Appium server for each app, though that adds a bit of overhead. Nonetheless, after analysing a few hundred apps, the iPhone consistently got into a broken state where Appium couldn't communicate with the device anymore, and sometimes it was not possible to uninstall or start apps manually through the UI anymore, either. The Appium developers describe known issues with the underlying WebDriverAgent on real devices [107]. We do not know whether these would also cause the problems we saw in iOS itself. Either way, they could be remedied by manually restarting the device, which then also required reapplying the jailbreak.

Finally, we noticed that the first `findElements()` call in a session does not find elements inside webviews, for some reason. As a workaround, we found that we can do an initial bogus `findElements()` call before any other ones and discard its results.

5.4. Device Preparation

For Android, Google has long offered an established emulator that is well supported by tooling. It is possible to install arbitrary apps into the emulator. Thus, we can utilise it for our analysis. We use Android 11 on the x86_64 architecture. This provides a good compromise between performance and compatibility, as the emulator provides hardware acceleration support for x86 [108] and we can still run apps only compiled for ARM as the emulator is capable of translating ARM instructions to x86 [109].

While Apple also offers an iOS Simulator, it cannot run apps packaged for distribution as downloaded from the App Store [110; 111]. We would need to have the apps' source code to create a development build, which is not feasible. And even then, the iOS Simulator is much more locked down than the Android Emulator and does not allow us to access everything we would need for the analysis. There are a number of commercial providers that offer running native iOS apps like *Appetize.io* and *RunThatApp*, but those only resell the iOS Simulator and are thus suitable either [112; 113].
As such, we need to use a real iPhone for the analysis. We are using an iPhone 7 that is running iOS 14.8. The choice of the iOS version is heavily constrained by two factors: jailbreak and device availability. As of the time of writing, the newest version of iOS is 15.4.1, but no jailbreak, which is necessary for the kinds of instrumentation we are doing, is available for iOS 15 yet [114; 115]. At the same time, Apple heavily restricts the versions that can be installed on an iPhone. It is currently only possible to upgrade to the very latest version, iOS 15.4.1, and no downgrades are possible at all[13] [117]. Thus, one has to resolve to buying used devices that happen to not have been upgraded yet by their previous owner. This makes choosing a particular version of iOS to base the analysis on practically impossible. The only requirement we were able to impose is a version of at least 14.5 (which introduced the *App Tracking Transparency* changes [118]) and lower than 15.0 (so a jailbreak is available).

In the following, we give an overview of the steps needed to prepare the device/emulator for instrumentation.

[12]We have also submitted a correction that has since been incorporated for that: `https://github.com/appium/appium/issues/16212`, `https://github.com/appium/appium/pull/16215`

[13]It *is* possible to downgrade to any version for which one has backed up Apple's signatures (so-called SHSH blobs), but those are tied to a specific device [116].

5.4.1. Android

On Android, the general device setup steps are:

1. Install the mitmproxy certificate as a root certificate authority. On newer versions of Android, including Android 11, which we are using, this requires starting the emulator with the `-writable-system` flag, disabling *Android Verified Boot* (AVB) and *device-mapper-verity* (dm-verity), and copying the certificate to the `/system/etc/security/cacerts/` folder [119–121].
2. Install Frida server 15.1.12.
3. Uninstall unnecessary Google apps to avoid their background traffic.
4. Disable Android's default captive portal checks [122] to further reduce background noise.

5.4.2. iOS

On iOS, the steps are:

1. Jailbreak the device using *checkra1n*[14] 0.12.4 beta.
2. Install the following Cydia packages: *Activator*[15], *SSL Kill Switch 2*[16], *Frida*[17], *Open*[18], *OpenSSH*[19], *sqlite3*[20].
3. Adjust the device settings to keep background traffic to a minimum, disabling background app refresh, automatic OS updates, iPhone analytics, and automatic app downloads and updates.
4. Uninstall all unnecessary third-party apps to avoid their background traffic.
5. Configure the machine mitmproxy is run on as the proxy and import mitmproxy's profile to trust its certificate authority [123].

5.4.3. Honey Data

On both platforms, we plant honey data so we can detect if apps transmit this data. We use randomly generated values with sufficient entropy to make sure they cannot appear in traffic by chance. We also read relevant device identifiers that apps may track. Table 5.1 shows the honey data we use.

Table 5.1.: Overview of the honey data we set on the devices. Most values are either manually placed on the device by us beforehand or already present on the system. We automatically set the location through Appium and seed the clipboard through Frida. Some values are only present on one of the platforms.

Value	Kind	Notes
Contacts	set manually	
Location	set through Appium	
Messages	set manually	
Calls	set manually	Android only

[14]`https://checkra.in/`
[15]`https://cydia.saurik.com/package/libactivator/`
[16]`https://github.com/nabla-c0d3/ssl-kill-switch2`
[17]`https://frida.re/docs/ios/#with-jailbreak`
[18]`http://cydia.saurik.com/package/com.conradkramer.open/`
[19]`https://cydia.saurik.com/package/openssh/`
[20]`http://apt.bingner.com/debs/1443.00/sqlite3_3.24.0-1_iphoneos-arm.deb`

Value	Kind	Notes
Clipboard	set through Frida	
Calendar	set manually	iOS only
Reminders	set manually	iOS only
Notes	set manually	iOS only
Health details	set manually	iOS only
Apple Home data	set manually	iOS only
WiFi SSID	set manually	
Device name	set manually	
Phone number	set manually	Android only
Operating system	device parameter	
Device model	device parameter	
Serial number	device parameter	
MAC addresses	device parameter	
BSSID	device parameter	
Advertising ID	set automatically by OS	
Local IP addresses	set automatically by OS	

5.5. App Dataset

To give an accurate picture of consent dialogs in mobile apps, we need a large dataset of apps on Android and iOS. We restrict our dataset to apps from the respective platform's top charts to only include apps that are actually commonly used in the wild and avoid potential outliers in insignificant apps.

5.5.1. App Selection

To get a sufficient amount of apps, on both platforms we cannot just rely on the overall top charts as they do not contain enough apps. Instead, we rely on the top charts per category and merge the results. A list of the categories is given in Table A.2 in Appendix A.1.

Android

Google does not offer an API for the top apps from the Play Store. The Play Store web UI does have top lists for apps and games[21], though, distinguishing between "top for 0€", "top selling", and "top grossing", with 200 apps each. Using only those would yield at most 400 free apps (assuming the top free lists for apps and games are disjunct), which is not sufficient for this thesis. Going through the pages for individual categories[22] offers various sublists on different topics, but those are not top charts and the number of apps in them varies wildly between a handful and up to around 50. Various third-party companies do sell top chart APIs for the Play Store with more results (e.g. [124; 125]), but they don't disclose how those are compiled, and it is preferable to rely on an official source.

[21] `https://play.google.com/store/apps/top`
[22] e.g. `https://play.google.com/store/apps/category/TOOLS`

We found that Google *does* in fact publish top charts per category, though those are not linked anywhere in the web UI as far as we can tell and can only be found through search engines. Those lists again contain 200 apps each. The links for those pages follow this pattern:

`https://play.google.com/store/apps/top/category/<category ID>?gl=<country code>`

The category IDs can be determined from the links in the category dropdown at the top of the web page, which are of the form: `https://play.google.com/store/apps/category/<category ID>`. From the category top pages, one has to click on the "top for 0€" link (which contains obfuscated parameters and thus can't be constructed directly by us).

Knowing that, we were able to write a scraper that extracts the top free apps per category using Microsoft's *Playwright*[23] browser automation framework. The scraper iterates over all category IDs and, for each one:

1. Visits the top list page.
2. Clicks the "top for 0€" link.
3. Continues scrolling to the bottom of the page until it doesn't get larger anymore (to defeat infinite scrolling as results are loaded dynamically using JavaScript[24]).
4. Extracts ID, name and top list position for each app on the page.

We scraped the top apps on Google Play on March 22, 2022. Counting all results, we found 6,970 apps in total, with 6,817 apps remaining after deduplication. For this analysis, we restrict our dataset to the top 100 apps per category. For that, we found 3,500 apps in total, with 3,421 apps remaining after deduplication.

iOS

Apple offers an RSS feed generator[25] for the top charts of various media types they sell (including apps). Using that, it is possible to obtain an XML or JSON file (despite the tool's name) of the top free or top paid apps per country on the App Store. The generator only returns up to 50 apps, but it is possible to retrieve up to 200 apps by manually adjusting the result limit parameter in the URL:
`https://rss.applemarketingtools.com/api/v2/de/apps/top-free/<limit>/apps.json`
Requesting more than 200 apps will result in an internal server error.

It used to be possible to get up to 1,200 top apps through an endpoint that was used in old versions of the iOS App Store: `https://itunes.apple.com/WebObjects/MZStore.woa/wa/topChartFragmentData` [126]. However, that endpoint now only provides the top 100 apps. iTunes on Windows[26] can however display top charts for each category (called "genre" by Apple), with up to 200 results each.

Observing iTunes' network traffic when loading these pages revealed the following endpoint:

```
GET https://itunes.apple.com/WebObjects/MZStore.woa/wa/viewTop
    ?cc=<country code>&genreId=<genre ID>&l=<language code>&popId=<top list type>
```

[23]`https://playwright.dev/`

[24]It would probably be possible to use the underlying network requests to extract the top list data directly without scraping, but those requests are heavily obfuscated.

[25]`https://rss.applemarketingtools.com/`

[26]Newer versions of iTunes don't include support for the iOS App Store anymore, but Apple offers a special, unsupported (but continuing to work as of the time of writing) version of iTunes (12.6.5.3) that still contains this feature and doesn't prompt the user to update to newer versions: `https://support.apple.com/HT208079`

The cc and l GET parameters control the country and language, respectively. The popId parameter determines the type of top chart returned, with the following possible values (again determined by observing the iTunes network traffic): 27 (top free apps for iPhone), 30 (top paid apps for iPhone), 38 (top grossing apps for iPhone), 44 (top free apps for iPad), 46 (top grossing apps for iPad), 47 (top paid apps for iPad). Finally, the genreId parameter controls the category the returned top list is for. A list of all possible categories can be found at https://itunes.apple.com/WebObjects/MZStoreServices.woa/ws/genres [127]. 36 is the first-level category for all apps on the App Store. The second level then has the actual app categories, e.g. 6000 for "Business". There are also third-level categories but only for "Games" and "Newsstand", so they are excluded here.

In addition to the GET parameters, the X-Apple-Store-Front header also needs to be set. It consists of between one and three numbers and has the following format: `<country>-<language>,<platform>` (only `<country>` is required, the others can be left off). Setting `<country>` to 143443 means Germany for example; a list of possible countries used to be available in the iTunes affiliate partner documentation and can still be accessed through the Internet Archive [128]. Among the available values for `<language>` are 1 (US English), 2 (British English), 3 (French), and 4 (German) [129; 130]. It is not possible to combine country and language arbitrarily, for example setting US English as the language but Germany as the country does not work, whereas British English does work for Germany. Finally, the `<platform>` value determines the Apple application the request is (supposedly) coming from, with 28 meaning iTunes 12 for example [131].

The endpoint's response format changes depending on the platform set. When setting a graphical client like iTunes, an HTML site will be returned that contains a script element which assigns the actual API return value to its.serverData. Through trial and error, we discovered that setting the platform to 26 or 29 makes the endpoint instead return the API result directly as JSON, making it a lot easier to consume programmatically.

In this result, $.storePlatformData.lockup.results then has a list of the respective top apps and their associated metadata. However, this list only contains 84 results. To get the full list of 200 apps, one has to use $.pageData.segmentedControl.segments[0].pageData.selectedChart.adamIds instead, which is a list of the numerical app IDs without metadata.

We retrieved the top charts of the iOS App Store on March 22, 2022. Counting all results, we found 5,205 apps in total, with 4,968 apps remaining after deduplication. Again restricting our dataset to the top 100 apps per category, we found 2,605 apps in total[27], with 2,486 apps remaining after deduplication.

5.5.2. App Acquisition

The download process on Android is already well known and implemented in various tools[28], so we omit a description of it here and only describe the existing tool we use. This is not the case on iOS however, so that is explained in detail.

[27]One category, "Catalogues", only has five apps in its top charts, which explains the count not being divisible by 100.

[28]Examples include: https://github.com/ClaudiuGeorgiu/PlaystoreDownloader, https://github.com/onyxbits/raccoon4, https://github.com/89z/googleplay, https://github.com/rehmatworks/gplaydl, and https://github.com/matlink/gplaycli

Android

To download Android apps from the Google Play Store, we use *PlaystoreDownloader*[29]. For this, a Google account is necessary that needs to be prepared with these steps [132]:

1. Enable two-factor authentication for the Google account.
2. Create an Android Emulator with Google Play support, use the Google account to login with that emulator, and download at least one app from the Play Store.
3. Install the *Device ID* app[30] and use that to read the emulator's assigned *Google Service Framework (GSF) ID* there.
4. Create an app password[31] for the Google account.
5. Temporarily allow signing in from new devices[32].
6. Update PlaystoreDownloader's `credentials.json` file with the correct `USERNAME` (the full email address of the account), `PASSWORD` (the app password generated before), and `ANDROID_ID` (the ID read using the Device ID app).

We then use a bash script to download the top 100 apps per category as collected in Section 5.5.1. We pass the `-s` flag to PlaystoreDownloader to correctly download split APKs.
If downloading fails for five or more consecutive apps, the script pauses for as many minutes to avoid exceeding potential undocumented rate limits in the Play Store endpoints.

The download script ran between March 22, 2022 at 18:54 and March 23, 2022 at 17:54. For 108 apps, the download failed. In all of these cases, the error code was `DF-DFERH-01`. Trying to download the failed apps through other APK download tools was unsuccessful as well. Viewing them in the Play Store on the emulator used to prepare the Google account showed the following error message: "Your device isn't compatible with this app." These apps were excluded from the analysis.

iOS

Prior to this thesis, there was no reliable automated way to download arbitrary iOS apps as IPA files. Older versions of iTunes supported manually downloading iOS apps as a native feature [133], which previous research on this topic leveraged by way of an AutoHotkey script[33] [15]. Apple has since ended support for downloading iOS apps through iTunes, but the feature continues to work in iTunes 12.6.5.3 for the time being [134]. *3uTools*[34], a third-party program for managing iOS devices, also includes the capability to download iOS apps [135]. We confirmed through traffic analysis that it uses the same endpoints as iTunes. These days, it is still possible to download IPA files using *Apple Configurator*[35], a program for companies that need to manage many iOS devices, which temporarily stores the IPA files of provisioned apps on the user's computer [136]. This however only works for apps that are already "purchased" (the same term is used for free apps) by the user's Apple ID and cannot be used to acquire new apps. The same applies

[29]`https://github.com/ClaudiuGeorgiu/PlaystoreDownloader/`
[30]`https://play.google.com/store/apps/details?id=com.evozi.deviceid`
[31]`https://myaccount.google.com/apppasswords`
[32]`https://accounts.google.com/DisplayUnlockCaptcha`
[33]`https://github.com/OxfordHCC/platformcontrol-ios-downloader/blob/b6a2038e57863bbdf6b98304883cf698c1579db8/instrumentor.ahk`
[34]`http://3u.com/`
[35]`https://support.apple.com/apple-configurator`

for *iMazing*[36], another third-party iOS device management software that can also only download already purchased apps [137], as it internally uses the same endpoints as Apple Configurator.

Thus, none of these methods provide a reliable way to download large amounts of iOS apps as needed for this thesis. Finally, *IPATool*[37] is the only command line tool we are aware of for downloading IPA files, but it was previously also only able to download already purchased apps, since it used the same Apple Configurator endpoints. Based on extensive reverse-engineering of the other described tools through network analysis, we were able to extend IPATool to also support purchasing new apps and thus use IPATool for this analysis. We contributed our changes back to IPATool[38] and they are already part of a new release of the software[39].

With our changes, the flow for downloading apps through IPATool now looks like this (the requests are incomplete here for brevity) [138]:

1. IPATool first requests the app metadata to find out the numerical app ID for the given bundle ID:

   ```
   GET https://itunes.apple.com/lookup?entity=software&media=software
           &bundleId=<bundle ID>&country=<country code>&limit=1 HTTP/2.0
   ```

   The response is a JSON object, where `$.results[0].trackId` is the numerical app ID.

2. It then logs in with a unique ID for the device, which is its MAC address without the colons, and the given Apple ID in the POST body:

   ```
   POST https://p25-buy.itunes.apple.com/WebObjects/MZFinance.woa/wa/authenticate
            ?guid=<unique device ID> HTTP/1.1
   ```

   It uses an Apple Configurator user agent for this instead of an iTunes one. The reason is that authenticating for iTunes requires an additional `X-Apple-ActionSignature` header and the process for generating that is not known.

   The response sets the authentication cookies.

3. Now authenticated, IPATool can try to purchase the desired app for the Apple ID through an endpoint used by iTunes, specifying the numerical app ID as `salableAdamId` in the POST body:

   ```
   POST https://buy.itunes.apple.com/WebObjects/MZBuy.woa/wa/buyProduct HTTP/1.1
   ```

   This request will fail with a status code of 500 if the Apple ID already owns the app[40]. This can simply be ignored.

4. If the previous request had set an iTunes user agent (and corresponding authentication cookies for iTunes), the reply would have already contained the download link for the IPA. For an Apple Configurator user agent however, this is not the case. Thus, IPATool needs to use the Apple Configurator endpoint for downloading already owned apps, again specifying the numerical app ID as `salableAdamId` in the POST body:

[36]`https://imazing.com/`

[37]`https://github.com/majd/ipatool`

[38]See this pull request: `https://github.com/majd/ipatool/pull/51`

[39]`https://github.com/majd/ipatool/releases/tag/v1.1.0`

[40]By first doing a request to `https://se-edge.itunes.apple.com/WebObjects/MZStoreElements.woa/wa/buyButtonMetaData` and using the returned `buyParams` in the `buyProduct` request, this could be avoided. But in the context of IPATool, the 500 error is not a problem, so the extra request is not necessary.

```
POST https://p25-buy.itunes.apple.com/WebObjects/MZFinance.woa/wa
         /volumeStoreDownloadProduct?guid=<unique device ID> HTTP/1.1
```

The response is a PLIST, where `$.songList[0].URL` is the download URL for the IPA file.

5. With that, IPATool can actually download the IPA file:

```
GET https://iosapps.itunes.apple.com/itunes-assets/<path>/<filename>.ipa
        ?accessKey=<unique access key> HTTP/1.1
```

6. Finally, IPATool signs the IPA file, which is necessary to run it on iOS devices. The signature is contained in `$.songList[0].sinfs[0].sinf` of the response from the `volumeStoreDownloadProduct` endpoint. It needs to be base64-decoded and written in the IPA file (which is just a ZIP archive) to `/Payload/<app name>.app/SC_Info/<app name>.sinf`.

   IPATool also fills the `/iTunesMetadata.plist` file in the IPA with the `$.songList[0].metadata` data from the `volumeStoreDownloadProduct` response, additionally appending the `apple-id` and `userName` properties, both set to the user's Apple ID email address.

We made one additional change to IPATool: As the top list data we gathered in Section 5.5.1 already has the numerical app IDs, we patched out the code for getting them from the bundle IDs, thus eliminating the first step from above[41].

We then also use a bash script to download the top 100 apps per category, just like on Android. In addition to the steps described there, the script also requests an app's privacy label for Section 6.4 immediately after downloading it through the following request:

```
GET https://amp-api.apps.apple.com/v1/catalog/DE/apps/<numerical app ID>
        ?platform=iphone&extend=privacyDetails&l=en-gb HTTP/2.0

Authorization: Bearer <token>
```

A token for this request can be obtained by visiting any app on the web version of the iOS App Store and observing the network traffic while clicking the "See Details" link next to "App Privacy".

The download script ran between March 22, 2022 at 21:00 and March 25, 2022 at 12:06. The download was delayed due to an App Store downtime. We also noticed that after downloading around 300 apps, the `buyProduct` endpoint would still return a valid response, but the app would not appear in the Apple ID's list of purchased apps, causing the subsequent `volumeStoreDownloadProduct` request to fail. This also applied to downloading apps through the App Store on the iPhone. After a few days, it would work again. To work around this problem, we used two Apple IDs for the download[42].

Downloading failed for five apps. Manual checking revealed that they had all since been taken down from the App Store. Those apps were excluded from the analysis.

[41] The code of our patched version is available at: `https://github.com/baltpeter/ipatool/tree/b_dev`

[42] In general, iOS will only allow apps signed for the logged-in Apple ID to run. If apps signed for another Apple ID are opened, a prompt saying 'To open "`<app>`", sign in with the Apple ID that purchased it.' is displayed. If one signs in with the other Apple ID in this prompt, running apps from both Apple IDs is subsequently possible (but the second Apple ID does not seem to be displayed anywhere in the UI).

6. Analysis Method

In this chapter, we present our method for detecting consent dialogs and violations in them, as well as how we extract the actual data being transmitted in the recorded traffic.

The source code is available in the same repository as the device instrumentation framework.

6.1. Consent Dialog Detection

As it is not feasible to detect consent dialogs on mobile based on the IAB TCF framework or CMP library-specific adapters (see Section 4.2), we need to use an approach that is based on common elements of consent dialogs. Looking at existing research for the web and disregarding TCF- and library-based methods [6–8; 139], most approaches for consent dialog detection are purely manual [1; 11; 140; 141]. Some research relies on adblock filter lists like *Easylist Cookies*[1] and *I don't care about cookies*[2] to detect HTML elements belonging to consent dialogs based on their ID or class [66; 84]. These lists are *very* broad, e.g. detecting any element with `CNIL` (the French data protection authority) or `Cookie` in its ID. Finally, privacy policy detection tends to be keyword-based [142; 143], matching on general terms like "policy" and "GDPR"[3], or use natural language processing [144].

The approach we use for our analysis should be automatic, with manual steps required at most for validation. We found that while some elements in mobile apps have descriptive IDs (e.g. `consent_button_accept_all`), this is not as common as on the web, with most elements IDs not containing enough information to discern whether they contain a consent dialog (e.g. `content_tv` does contain one). This means that we cannot rely on element IDs alone and will not be able to use adblock filter lists.
We found many apps displaying a dialog that, at first glance, looks like a consent dialog but actually just concerns the company's terms of service or similar. These need to be filtered out correctly[4].

Based on those considerations, we use a text-based approach that matches on the elements' text content. We encountered three types of consent elements we want to distinguish, motivating the following taxonomy:

Link Some apps only contain a link to a privacy policy in a menu or the footer. While this can be enough to satisfy a controller's information obligations under Articles 12–14 GDPR, a link can obviously not be used to obtain consent from a user.

Notice Some apps inform users that the app processes their data, not seldom claiming that the user agrees to this by continuing to use the app. The notices are often in the form of a banner or a short sentence tucked away under a form. As established in Chapter 3, consent in the context of data protection cannot be given through inaction, so apps may not assume consent based on only such a notice. It can, however, be used by the controller to meet their duty to inform the user of the processing.

[1] `https://secure.fanboy.co.nz/fanboy-cookiemonster.txt`
[2] `https://www.i-dont-care-about-cookies.eu/abp/`
[3] See: `https://github.com/RUB-SysSec/we-value-your-privacy/blob/181cbffb62ce2dcc89ff9b467401093aa10f0cd8/privacy_wording.json`
[4] Even if those apps bury data protection information somewhere in their terms of service, it doesn't make a terms of service dialog a consent dialog (cf. Article 7(2) GDPR), so we can safely ignore those cases.

Dialog Finally, some apps not only inform the user about their data processing but actively solicit their consent through a button or a checkbox that needs to be clicked. This is the only way that apps can actually obtain valid consent under the GDPR.

6.1.1. Our Approach

Based on manually looking at many apps, we have collected a list of common phrases that German and English apps use to refer to data protection, like "we care about your privacy" or "by continuing to use our app, you acknowledge that we may process your data in line with our data protection statement". We extracted the key elements from these phrases and compiled them into compact regexes that only match on the important words, leaving out as much of the app-specific wording as possible.

We detect a dialog or notice in an app if we encounter at least one match for one of those regexes in an element. We distinguish between notices and dialogs only by whether they contain an interactive element: In addition to the criteria for a notice, a dialog also needs to have at least one button. For that, we have compiled another list of regexes for buttons typically found in consent dialogs, matching on labels like "accept", "okay", or "reject".
If an app has neither a dialog nor a notice, but we detect a link to a privacy policy, we classify it as "link". Unfortunately, searching for privacy policy links also has to be done based on text only, as Appium doesn't support reading link targets.

To weed out notices and dialogs not referring to data protection, we introduce another criterion: We compiled a list of keywords commonly found in consent dialogs and assign a keyword score based on how many of those we find in an app. We differentiate between keywords that are clearly related to data protection and a clear indicator of a dialog like `/(ad(vertising|s)?|content|experience) personali(s|z)ation/` or `/(necessary|essential|needed) cookies/`, which yield one point, and ones that are commonly but not necessarily related to data protection like `'geolocation data'` or `'IP address'`, which only yield half a point.
We only detect a dialog or notice if the keyword score is at least one. Alternatively, if we find a privacy policy link, that is also sufficient. Conversely, even if we do not detect one of the dialog phrases, if an app reaches a keyword score of at least three, we classify it as "maybe dialog/notice".

One of those phrase regexes looks like this for example:

```
/have read( and understood)? [^.]{3,35}
    (privacy|cookie|data protection|GDPR) (policy|notice|information|statement)/
```

This regex tries to anticipate all possible word choices for "privacy policy" that could come up. As explained, we leave out any words that are not strictly necessary to classify a sentence as coming from a consent dialog. For our purposes, it is not important whether a dialog says "You hereby confirm that you have read our aforementioned privacy policy." or simply "I have read the privacy policy." However, it is important that the "read" and "privacy policy" parts belong to a single statement and aren't parts of entirely separate sentences. Thus, we limit the number of characters that may occur between them and disallow periods between the parts.
The main criterion when compiling the regexes was to avoid false-positives under the assumption that it is better to provide an under-approximation of consent dialog prevalence than to wrongly detect other elements as consent dialogs. A full list of the regexes we use can be found in Appendix A.3.

All text matching is done case-insensitively. We only consider visible elements. Button and privacy policy link texts additionally need to be at word boundaries, to avoid matching "acknowledge" as "no" for example.
Appium has no general way of distinguishing between buttons and other elements. Of course, a text element that happens to contain the word "no" should not be detected as a reject button, either. Thus, we additionally only match buttons if their text is at most twice as long as the respective regex.
Finally, we need to make sure not to match an "I do not consent" button as an "accept" button. For that, we use a list of negator regexes containing words like "don't" and "refuse". An element that matches one of these negator regexes is never classified as an "accept" button.

6.1.2. Interaction with Consent Dialogs

We also automatically interact with the detected consent dialogs to measure the difference in (tracking) behaviour between the app not having received any input and after having consented or refused consent. For that, we first reset the app completely and wait ten seconds to allow for the consent dialog to appear.

We then click the first "accept" button we detected, preferring ones with a clear label. As before, we record the network traffic for 60 seconds but mark the run as an "accepted" one. After the timeout is over, we also save the preferences of the app.

We repeat the same steps for the "reject" button analogously.

6.2. Dark Pattern Identification

We detect the following violations and dark patterns in apps determined to show a consent dialog:

Processing before consent Processing that can only rely on consent as a legal basis may of course only occur after consent has been given (cf. Section 3.1). As such, we consider any tracking that happens before we have interacted with a consent dialog or after we have refused consent a violation.

This also automatically applies to all apps performing tracking without a consent dialog or with only a notice or privacy policy link.

Ambiguous button labels If a consent dialog has an "accept" button, it needs to have a clear label that unambiguously communicates to the user that clicking the button will result in consent to the described processing (cf. Section 3.2.4). To detect violations, we sort the button regexes (see Section 6.1.1) into clear (like "allow" or "consent") and ambiguous (like "okay" or "continue") ones and record a violation if the dialog has at least one "accept" button with an ambiguous label button but none with a clear label.

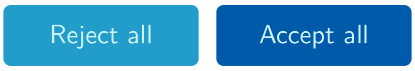
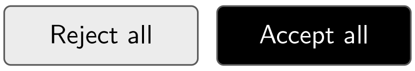
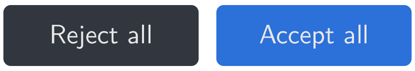
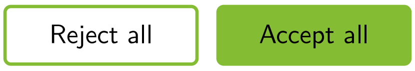
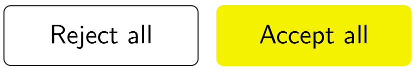
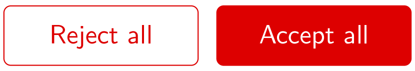
We proceed analoguously for "reject" buttons, which can also have clear (like "decline" or "refuse") and ambiguous (like "options" or "cancel") labels.

"Accept" button without "reject" button If a consent dialog has an "accept" button on the first layer, it also needs to have a "reject" button on the same layer (cf. Section 3.2.4). We record a violation if we detect an "accept" but no "reject" button on the screen shown to the user without any interaction.

“Accept” button highlighted compared to “reject” button by size An app may not nudge a user into consenting by highlighting the “accept” button compared to the “reject” button, for example by making it bigger (cf. Section 3.2.2). To detect violations, we look at the bounding rectangle of both buttons. We record a violation if the product of the “accept” button’s width and height is at least 1.5 times bigger than that of the “reject” button.

If there is more than one of each button, we cannot know which ones to check against each other to detect whether one is highlighted. Thus, for an “accept” button, we only record a violation it is highlighted compared to *every* “reject” button. But it is enough if there is one “accept” button that is highlighted, not all of them need to be.

Table 6.1.: Comparison of various button combinations and the deltaE CMC difference between their most prominent colours. All buttons are adapted from real designs we encountered.

Buttons	Most prominent colours	deltaE
	`#229ccb`, `#005aaa`	26.12
	`#ececec`, `#000000`	64.64
	`#32363f`, `#2c70d9`	68.10
	`#ffffff`, `#84bc33`	113.36
	`#ffffff`, `#f4f200`	146.07
	`#ffffff`, `#dd0000`	151.51

“Accept” button highlighted compared to “reject” button by colour Similarly, a consent dialog may also not have an “accept” button that is more prominent than the “reject” button through its colour (cf. Section 3.2.2). To detect violations, we take screenshots of both buttons (cropped to just the respective button’s bounding rectangle). We then use the *get-image-colors*[5] library to extract the most prominent colour from each screenshot using colour quantisation. Finally, we compute the two colours’ *deltaE CMC difference*, a numeric measure for the distance between two colours, [145] using the *chroma.js*[6] library and record a violation if it is more than 30. See Table 6.1 for a reference of different colour differences in buttons we encountered.

This of course does not guarantee that it is the “accept” and not the “reject” button that’s highlighted. To ensure that, we would have to also compare the button colour against the background colour. Unfortunately, Appium does not have the capability to extract the background colour and trying to guess which pixels to screenshot to get just the button background without catching other elements would be too error-prone. Thus, we manually review all detected violations of this type.

In the case of more than one of each button type, we proceed exactly as for the previous violation.

[5]`https://github.com/colorjs/get-image-colors`
[6]`https://vis4.net/chromajs/`

App stops after refusing consent It needs to be possible to use an app without consenting (potentially with a reduced feature set), so an app may not quit after the user has refused their consent (cf. Section 3.2.1). To detect violations, we first ensure that the app is still running and in the foreground, then click the "reject" button, wait for ten seconds and record a violation if the app is not running and in the foreground anymore afterwards. In the case of multiple "reject" buttons, we click the first one, preferring a "reject" button with clear label, if available.

Even if we don't detect a violation or dark pattern in a consent dialog, that does not mean that it is compliant. An absence of violations only represents a minimum of compliance that can be reliably checked using automated methods. As such, our findings will only provide a lower bound in terms of violations but, conversely, an upper bound of compliance in mobile apps.

6.3. Tracking Content Extraction

To extract and classify the tracking data that apps send from the recorded network traffic, we use a two-fold approach: We wrote a series of adapters for the most common trackers which understand the actual tracking protocol and can thus precisely extract the transmitted data. For those requests that are not matched by one of our adapters, we employ indicator matching to check for the presence of common data types. For each request, we decide whether the contained data is pseudonymous or anonymous. We consider the data in a request pseudonymous if the request contains at least one unique identifier for the device or user like the device's advertising ID (including the IDFV on iOS and hashed forms thereof) or the user's public IP address[7]. Otherwise, we consider the data in the request anonymous.

In addition to that, we also analyse the cookies that are set in requests. For that, we leverage the Open Cookie Database [146], a list of 710 cookies as of the time of writing, mapped to the platform they are set by and a category they can be attributed to, among other things.

6.3.1. Endpoint-Specific Tracking Request Adapters

We noticed that there is a comparatively small number of tracking endpoints which make up a large portion of the app traffic. We developed 26 adapters than can extract the tracking data in a common schema that we can easily reason about for the most common endpoints. For our purposes, an endpoint is uniquely identified by the scheme, host and path without GET parameters.

Each adapter consists of these parts:

Endpoint URLs Each adapter has a list of endpoints that it works for. Endpoints can either be specified as simple strings or as regexes to accommodate for URLs with parameters in hostname or path.

Match function Optionally, adapters can have an additional match function used to filter out requests to the same endpoint that the adapter cannot handle, e.g. based on the request method or body.

Prepare function Tracking data can be included in the URL or request body. In addition, we have observed a variety of data formats and encodings used by trackers, sometimes even different ones for the same

[7]It is of course not technically possible for a server to handle a user's request without at least temporarily processing their IP address. We only consider cases where the IP address is literally included in the request body or path.

endpoint. The prepare function parses as much as possible of the raw request into a JavaScript object with plain text values.

Steps of the processing that need to happen in the prepare function include: Parsing JSON or query strings, decoding Protobuf blobs, combining data from the body and GET parameters, and decoding base64 strings. Often, different formats and encodings are nested. For example, the bodies of requests to Supersonic can either be a plain JSON object or a base64 string that holds a GZIP which in turns holds the actual JSON object. Meanwhile, ironSource sends a base64-encoded JSON as a query string parameter. And some requests to Facebook have a query string as the body which holds a JSON with one property being an array of events that are JSON strings, while others have a JSON object as the body that holds a JSON string of an array of objects where the actual data is query-string-encoded in each object's value.

Extract function Finally, the extract function extracts the known data types from the prepared request and brings it into a unified schema. In many cases, one data type can be present in different properties, so we use the first one that actually holds data.

Sometimes, it is not obvious what data a property holds because it has no name or the name is not descriptive. In these cases, we have looked at all instances of the respective property across all requests and only extracted the properties we were able to definitively identify. This means that we can once again only provide a lower bound on the data that is being transmitted.

6.3.2. Indicator Matching in Network Traffic

For the requests that cannot be handled by any of our endpoint-specific adapters, we perform indicator matching on the path and request body. We search for the honey data values described in Section 5.4.3.

In addition to matching against the plain text, we also match base64-encoded values. This cannot be done by simply encoding the indicator value and matching the traffic against that. The actual base64 encoding of a value depends on its offset within the whole string that is being encoded as well as the string's length. We ported a PowerShell script [147] that generates all possible ways a value can be base64-encoded and builds a regex for that to JavaScript[8].

6.4. Apple Privacy Labels

App developers on iOS are supposed to inform users about what data their apps process through privacy labels. Among other things, they need to declare the following two details [21]:

Types of data The privacy label needs to list the data types collected by the app, regardless of whether they are collected by the app developer themselves or by third-party companies. As of the time of writing, there are 32 possible data types across 14 categories. While the meaning of some data types is obvious, e.g. for "email address" and "phone number", others are not well defined and lack a clear description, e.g. "Other Data Types" which is simply described as "[a]ny other data types not mentioned".

[8]We have published our port as a library: `https://github.com/baltpeter/base64-search`

The data types have to be sorted into “Data Linked to You”, “Data Used to Track You”, and “Data Not Linked to You”. Apps that do not collect any data declare an empty “Data Not Collected” list.

Purposes In addition, the privacy label needs to list the purposes that the data types are collected for. The possible values as of the time of writing are: “Third-Party Advertising”, “Developer’s Advertising or Marketing”, “Analytics”, “Product Personalization”, “App Functionality”, and “Other Purposes”.

To analyse the privacy labels, for each app, we go through the privacy types and record a list of the data types and purposes declared in the label, distinguishing between ones declared as pseudonymous (i.e. with a privacy type of “Data Linked to You” or “Data Used to Track You”) and anonymous (i.e. with a privacy type of “Data Not Linked to You”).

Table 6.2.: Mapping from data types that can appear in privacy labels [21] to data types we detect and consider equivalent. Note that unlike Apple, we do not distinguish between precise and coarse location.

Privacy label data type	Our corresponding data types
Email Address	Apple ID email address
Phone number	Phone number
Health	Apple Health honey data
Location	Coordinates or address of device location
Contacts	Contacts honey data
Emails or Text Messages	SMS honey data
Other User Content	Clipboard content, honey data in reminders, calendar, notes, Apple Home
Product Interaction	Viewed pages in the app, whether the app is in the foreground
Performance Data	RAM usage, disk usage, device uptime
Device ID	IDFA, IDFV, hashed IDFA, hashed IDFV
Other Diagnostic Data	Device root status, device emulator status, network connection type, signal strength, charging status, battery percentage, sensor data
Other Data Types	Device name, carrier, roaming status, local IP address(es), MAC address(es), BSSID, volume

Then, we look at the detected tracking content as described in the previous section. We compare the data types that we observed being transmitted by the app against the data types declared in the label. As some of the possible data types in privacy labels are not clearly defined, we have created a mapping between them and the data types we detect, which can be seen in Table 6.2. We can only check a subset of the data types that can be declared. For each privacy label data type, we determine whether it was correctly declared, correctly not declared, wrongly declared as anonymous, wrongly undeclared, unnecessarily declared, or unnecessarily declared as pseudonymous. Of course, detections of unnecessary declarations or a correct lack of a declaration are only within the context of the traffic we recorded. It is possible that apps do in fact transmit this data, but we did not observe that.

Finally, we look at the declared purposes. Here, we judge whether the app correctly declares the purposes “Analytics” and “Third-Party Advertising”/“Developer’s Advertising or Marketing” by comparing the contacted hosts against the EasyList and EasyPrivacy adblock filter lists[9] [149].

[9]EasyList and EasyPrivacy are for adblockers in browsers and contain a lot more than hostnames. We use the Firebog versions of both lists [148]: `https://v.firebog.net/hosts/Easylist.txt` and `https://v.firebog.net/hosts/Easyprivacy.txt`

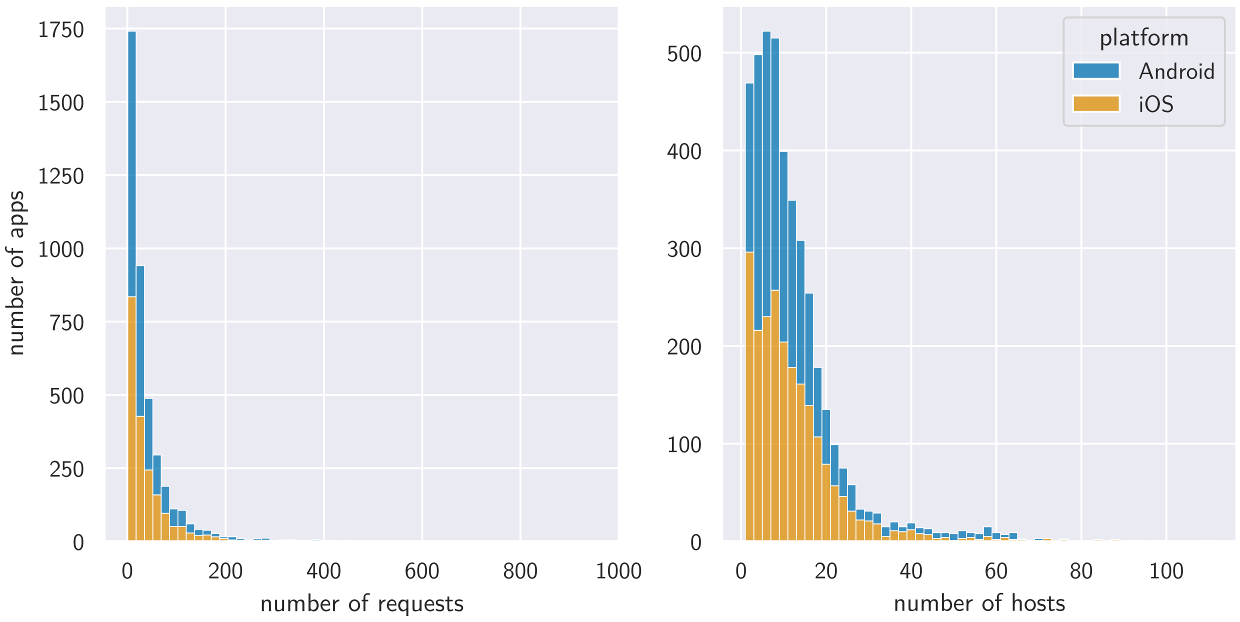
7. Results

We successfully analysed 4,388 apps with 2,068 apps on Android and 2,320 apps on iOS, corresponding to 62.42 % and 93.51 % of the downloaded apps, respectively. On Android, the high number of apps we could not analyse is caused for the most part by problems with the certificate pinning bypass through objection. 1,049 of the Android apps failed to launch or quit immediately after being launched through objection. These apps were excluded from the analysis. We discuss this further in Section 8.2. On iOS, only 65 apps failed to launch and 18 apps could not be installed because they require a newer version of iOS than we can use. The remaining failures on both platforms were mostly due to Appium or Frida commands failing even after multiple retries.

In the interest of reproducibility, the processed data behind all graphs is available in our GitHub repository.

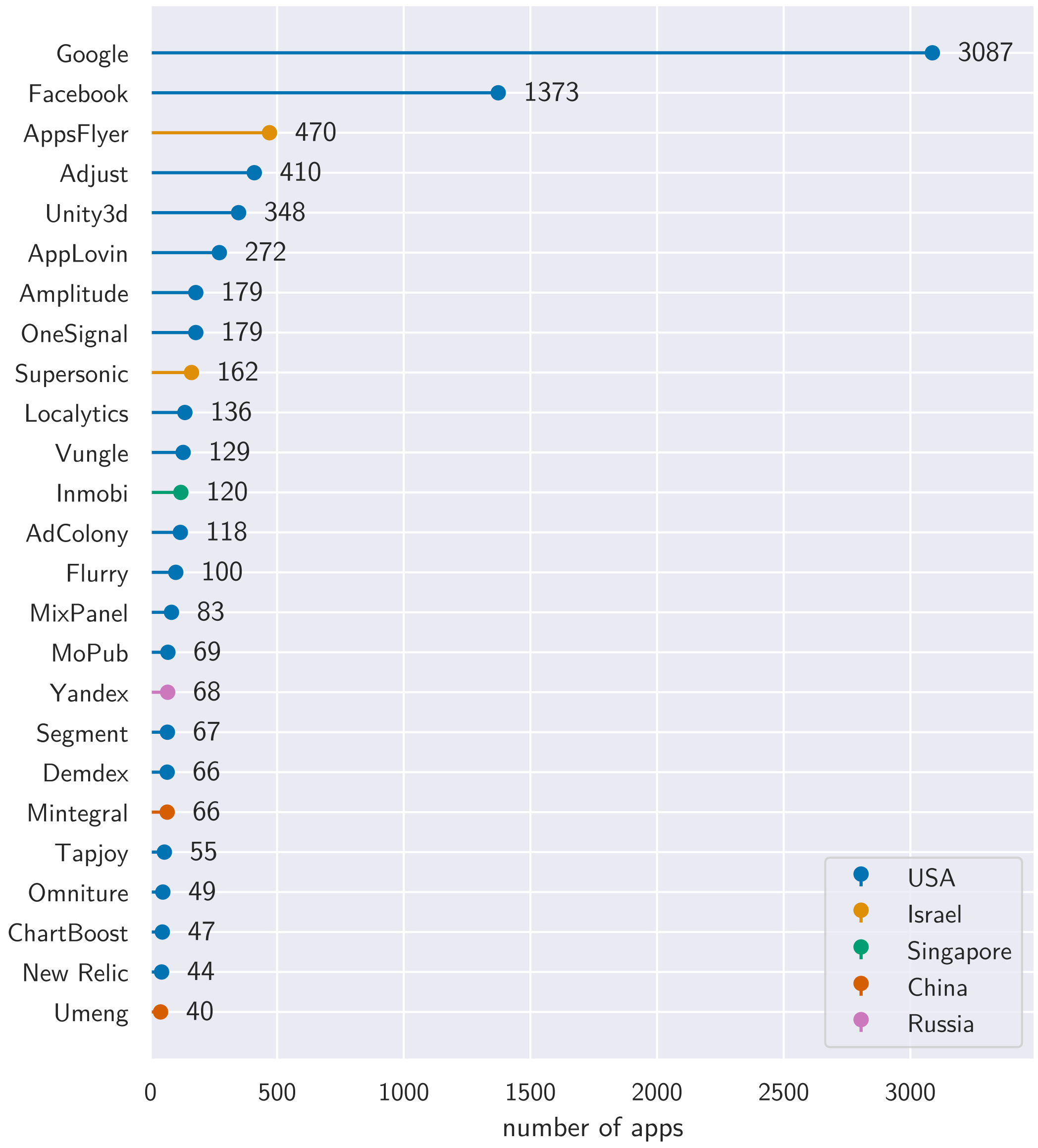
7.1. Network Traffic and Tracking

Figure 7.1.: Number of requests and unique hosts contacted per app without any user interaction. Three apps with more than 1,000 requests are omitted in this graph: `com.prequel.app` on Android with 2,500 requests, and `com.audiomack.iphone` and `com.storycover` on iOS with 2,383 and 1,019 requests, respectively.



In total, we recorded 194,817 requests after filtering out the operating systems' background traffic. Figure 7.1 illustrates the amount of requests and unique hosts per app in the initial run before we interacted with the apps. 50 % of apps did less than 23 requests and 75 % of apps did less than 50 requests, but there were also some outliers with up to 2,500 requests from a single app and 19 apps doing more than 500 requests. On average, apps on Android did 44.27 requests and apps on iOS did 44.66 requests. There were 65 apps on Android and 158 apps on iOS with no requests at all.
53 % of apps contacted less than 10 unique hosts, with 11.85 hosts on average across both platforms.

Figure 7.2.: Number of apps that sent requests to the 25 most common trackers in our dataset according to Exodus [149] (without user interaction). The trackers are coloured by the country they are based in. We compiled the mapping from tracker to country by looking at the trackers' privacy policies. When a policy listed multiple establishments, we chose the country of the main one.



61,700 (33.32 %) of the requests that happened without user interaction were identified as going to trackers when we compared their hostnames against the Exodus tracker database [150], with 78.08 % of apps making at least one request to a tracker. Figure 7.2 shows the 25 most common tracker companies that we encountered. Google and Facebook were the most common tracker companies by far, receiving traffic from 70.35 % and 31.29 % of apps, respectively. Notably, Google's trackers were the most common across Android *and* iOS. On Android, 81.21 % of apps sent traffic to Google trackers, and on iOS, 67.11 % did. The remaining trackers were all only contacted by 10 % or less of the apps. The majority of the contacted trackers are in the US, with only six of the 25 most common trackers being based in different countries, namely Israel, Singapore, China, and Russia.

Figure 7.3.: Number of times that the observed data types were transmitted per app and tracker without any user interaction, grouped by whether they were transmitted linked to a unique device ID (i.e. pseudonymously) or without identifiers for the device (i.e. anonymously). Note that we are also using the term "IDFA" for the Android advertising ID here.

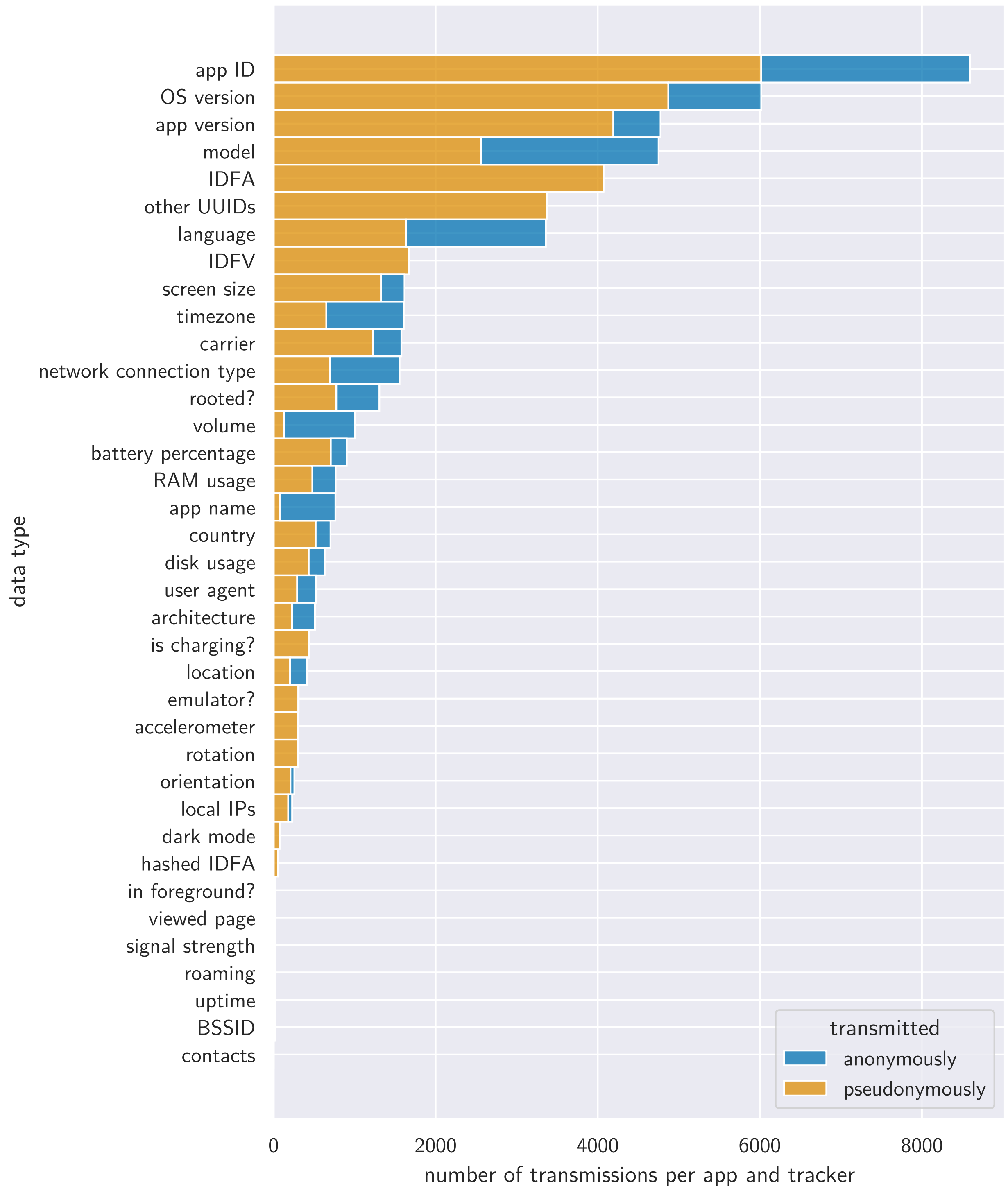
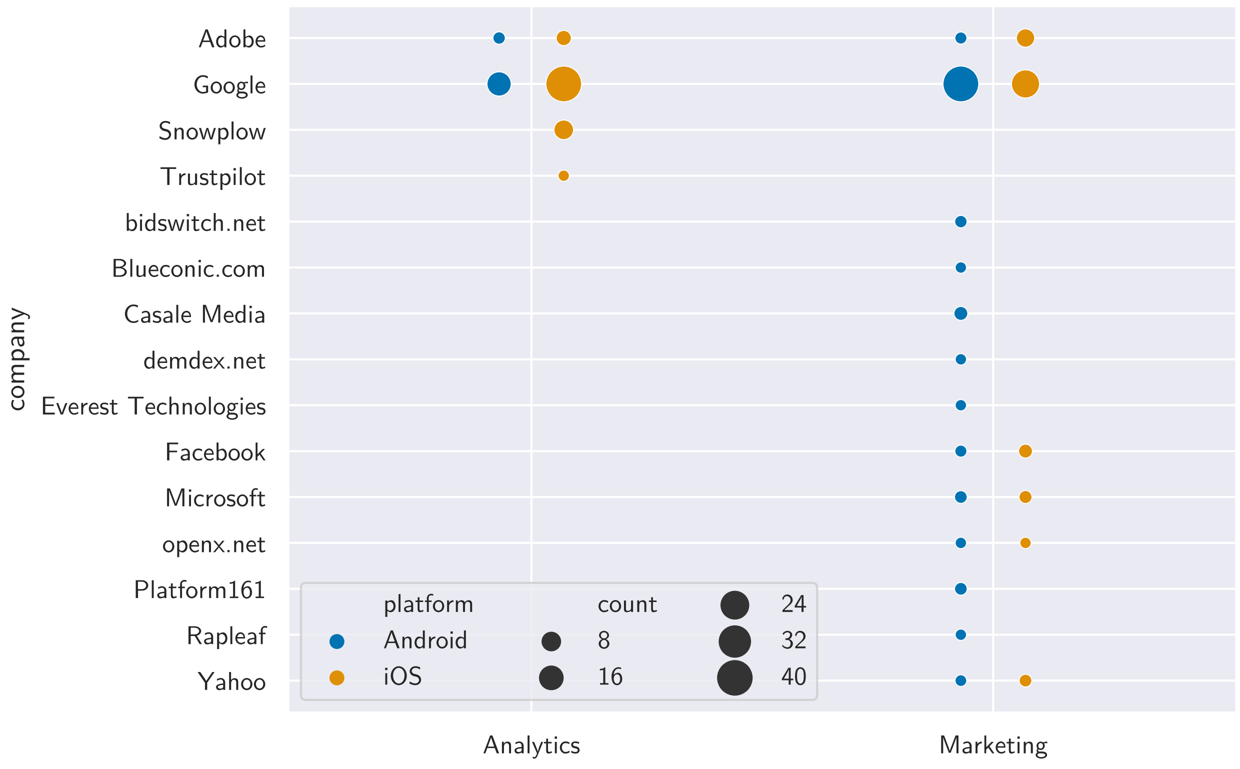


Figure 7.4.: Prevalence of cookies by various companies and which category they can be attributed to (across all runs) according to [146]. Each point represents the number of times a cookie by the company in the respective row and belonging to the category in the respective column was set by an app to a different value, with the size of the point indicating how often the cookie was set. The points are coloured according the apps' platform.



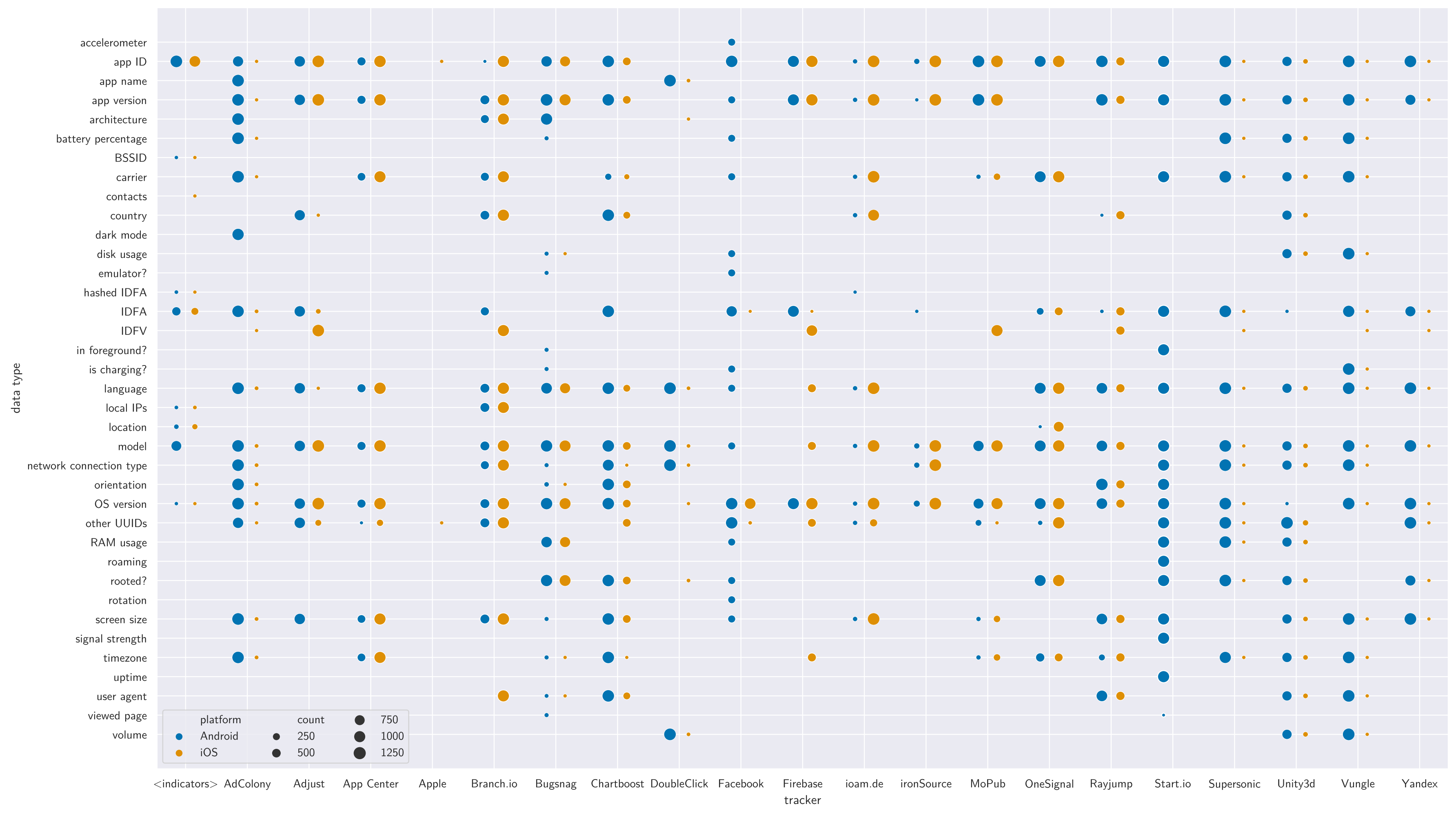
Looking at the data transmitted to trackers, 3,201 apps (72.95 %) sent at least one request containing a unique device identifier like the advertising ID, IDFV, or another UUID in the initial run, making all other data included in those requests pseudonymous and thus personal data that falls under the GDPR. Our 26 endpoint-specific tracking request adapters were enough to process 20,465 of 194,817 requests (10.50 %). Using those and indicator matching on the requests not covered by an adapter, we also observed a wide array of other data types being transmitted to trackers, including for example the location, jailbreak status, volume, battery percentage, sensor data, and disk usage. Figure 7.3 lists how often each data type was transmitted per app and tracker. Indeed, even benign data types like the operating system or phone model are linked to the specific user and device through unique IDs in most cases.

Figure 7.5 further plots the observed data types against the trackers they were sent to, highlighting that some trackers are only active on one platform and others transmit different types of data depending on the platform. For example, we saw Facebook receiving significantly fewer data types on iOS compared to Android. Conversely, Google Firebase received more data types on iOS.
From this data, we can also infer that some trackers are more common on one platform. For example, we observed significantly more transmissions to AdColony on Android compared to iOS, while it is the other way around for ioam.de and ironSource.

Finally, we also analysed the cookies that were set in the requests against the Open Cookie Database. The results are shown in Figure 7.4. We only observed cookies from the *Analytics* and *Marketing* categories but

Figure 7.5.: Observed transmissions of various types of data to trackers without interaction, grouped by platform. Note that we are also using the term “IDFA” for the Android advertising ID here. Each point represents a number of apps transmitting the respective row’s data type to the tracker in the respective column, with the size of the point indicating how many apps performed this transmission at least once. The points are coloured according to the apps’ platform.

The observations in the “<indicators>” column came from string-matching plain and base64-encoded text in the requests not covered by an endpoint-specific tracking request adapter.



none from the *Functional* and *Preferences* categories, which is likely due to the database's focus on websites. Most cookies we saw were marketing cookies. We saw a more diverse set of companies setting cookies on Android than on iOS. Once again, Google was the most prevalent company on both platforms.

7.2. Prevalence of Consent Dialogs

Table 7.1.: Number of apps where the different consent elements were detected by platform. The percentages are relative to all apps in the respective column.

Classification	Detections on Android	Detections on iOS	Detections in total
dialog	132 (6.38 %)	199 (8.58 %)	331 (7.54 %)
maybe dialog	17 (0.82 %)	36 (1.55 %)	53 (1.21 %)
notice	103 (4.98 %)	82 (3.53 %)	185 (4.22 %)
maybe notice	5 (0.24 %)	5 (0.22 %)	10 (0.23 %)
link	103 (4.98 %)	103 (4.44 %)	206 (4.69 %)
neither	1,708 (82.59 %)	1,895 (81.68 %)	3,603 (82.11 %)

Table 7.1 lists the number of apps where our analysis detected a consent element. Across all apps, we detected a consent dialog in 384 apps (8.75 %), a consent notice in 195 apps (4.44 %), and a link to a privacy policy in 206 apps (4.69 %). Thus, in total, 785 apps (17.89 %) had one of the consent elements we detect.

There appears to be little difference in the prevalence of the consent elements between platforms. Across all types, the relative counts differ by no more than 2.3 percentage points. We detected slightly more consent dialogs on iOS compared to Android, whereas we detected slightly more notices on Android. In total, we detected any consent element in 18.32 % of apps on iOS compared to 17.41 % on Android.

7.3. Violations in Consent Dialogs

Looking at the individual dark patterns, 43.2 % of the dialogs did not have a "reject" button on the first layer. Ambiguous labels for the "accept" and "reject" buttons were also common with 37.5 % and 32.8 % of dialogs exhibiting them, respectively. "Accept" buttons were most commonly highlighted compared to "reject" buttons by colour with 31.2 % of dialogs, compared to only 10.7 % of dialogs highlighting the "accept" button by size. Finally, 16 apps (4.2 %) quit after refusing consent.

Figure 7.6 illustrates the observed combinations of dark patterns in consent dialogs and compares them against the apps' top chart positions. We most commonly observed "accept" buttons with an ambiguous label in combination with no "reject" button on the first layer (22.7 % of dialogs). Consent dialogs also often have an ambiguous "reject" button and highlight the "accept" button by colour (14.3 %). Both of those were slightly more frequently the case for apps ranked highly in the top charts. Other than that, most of the dark patterns occurred on their own and with no significant correlation to the apps' top chart position.

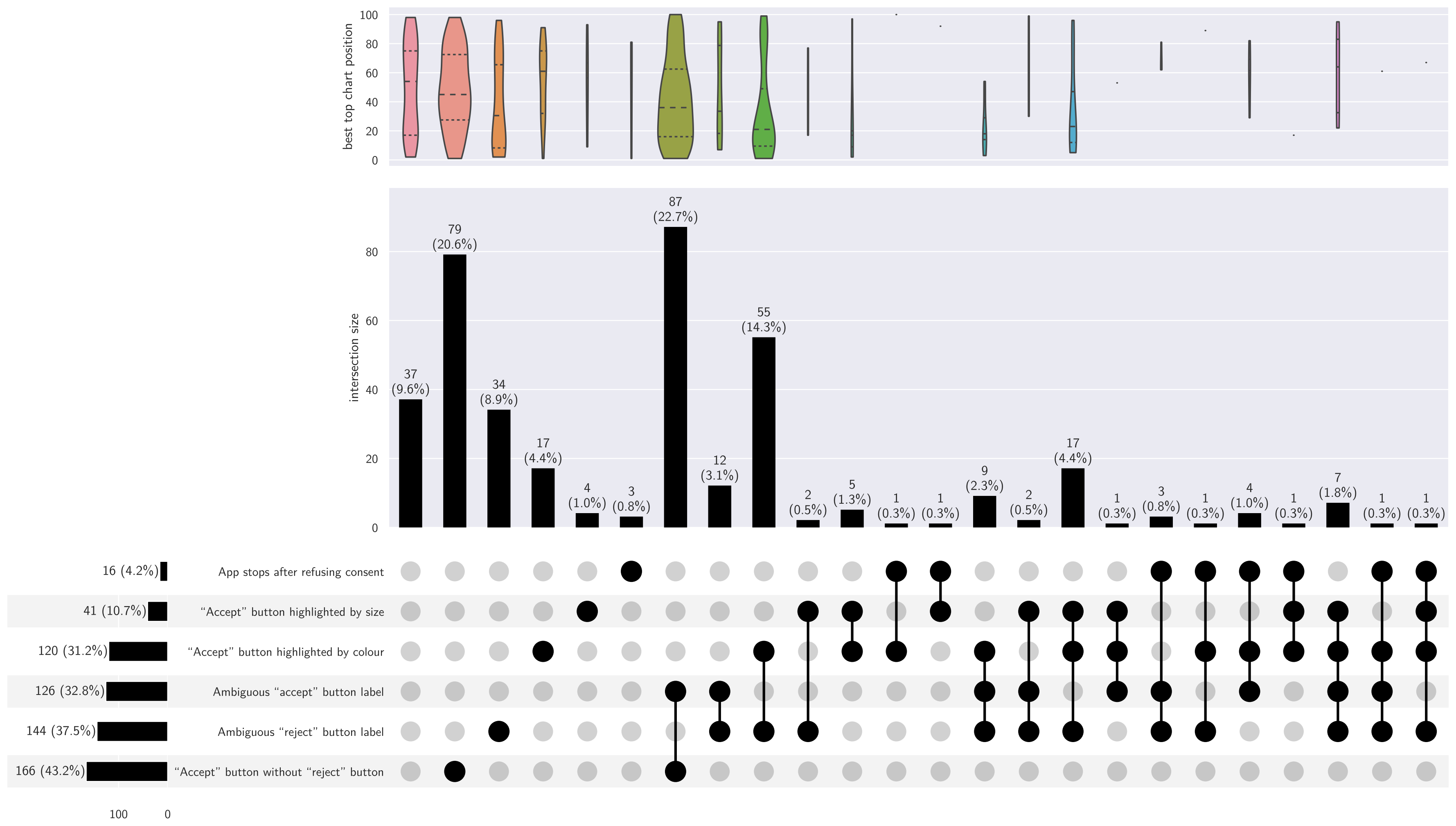


Figure 7.6.: UpSet plot [151] showing the different combinations of dark patterns we have detected in consent dialogs. Note that not all combinations are possible. Most of the dark patterns refer to the “reject” button and thus of course cannot occur if there was no “reject” button to begin with.

The upper violin plot illustrates the distribution of top chart positions among the apps in the respective set. If an app was listed in multiple top charts, we recorded its highest position.

In total, we have detected at least one dark pattern in 347 of the 384 apps with a dialog (90.36 %). The share of dark patterns in dialogs is slightly higher on Android with 136 of 149 dialogs (91.28 %) compared to 211 of 235 (89.79 %) on iOS.

On their own, the dark patterns we detect are not necessarily violations of data protection law. Using dark patterns in a consent dialog just results in the consent acquired through it being invalid. As such, the actual violation that we can detect is the transmission of tracking data based on such invalid consent[1].

We found that 328 of the 384 apps with a dialog (85.42 %) transmitted pseudonymous data in any of our runs. Further, 297 of the 347 apps with a detected dark pattern in their dialog (85.59 %) transmitted pseudonymous data in any of our runs. Taking that into consideration, we have identified that 77.34 % of the 384 detected dialogs failed to acquire valid consent for the tracking that they perform.

7.4. Effect of User Choices

To gain insights into how different choices in the consent dialogs affect the tracking going on, we collected the app's network traffic, distinguishing between the initial run without any user input, and the runs after accepting and rejecting the dialog if present. We collected traffic for 330 apps after accepting and 28 apps after rejecting. The latter number might seem low but can be explained by the fact that most dialogs we found either did not contain a first-layer "reject" button at all or only had one with an ambiguous label, and we only clicked ones with a clear label.
We collected 185,152 requests in the initial runs, 9,342 requests in the accepted runs, and 323 requests in the rejected runs. Note that for the accepted and rejected runs, we only collected the traffic *after* clicking the respective button. The initial traffic before any interaction was not recorded again in those runs to only capture the change in traffic after interaction.
Given the low number of apps for which we were able to collect traffic after rejecting and the low number of corresponding requests, the results for those are most likely not representative.

In the traffic before interaction, 33.32 % of requests were identified as trackers by Exodus. In the traffic after accepting the dialogs, this percentage slightly dropped to 31.90 %, while after rejecting, there was actually a higher percentage of the traffic that was identified as tracking with 47.06 %. Meanwhile, 78.08 % of apps contacted at least one Exodus-identified tracker in the initial runs. In the accepted runs, 25 additional apps (7.58 % of the accepted apps) contacted a tracker that previously didn't. In the rejected runs, 16 of 28 apps (57.14 %) continued contacting trackers, and one additional app did for the first time.

Furthermore, in the initial runs, 3,201 of the 4,388 apps (72.95 %) transmitted pseudonymous data. Of the 384 apps with a detected dialog, 282 (73.44 %) already transmitted pseudonymous data before receiving a consent choice. In the accepted runs, 46 additional apps started transmitting pseudonymous data. In the rejected runs, 12 of 28 apps (42.85 %) continued transmitting pseudonymous data and one app started doing so for the first time.

Figure 7.7 lists how often each data type was transmitted per app and tracker after accepting. Comparing that to the transmissions without user interaction in Figure 7.3 shows little difference in the data types that are transmitted to trackers after consent was given.

[1]Though presenting the user with a consent dialog that uses dark patterns without ever actually requiring consent for any processing would arguably run afoul of the principle of lawfulness, fairness, and transparency set forth by Article 5(1)(a) GDPR and thus be a violation in and of itself.

Figure 7.7.: Number of times that the observed data types were transmitted per app and tracker after accepting the consent dialogs, grouped by whether they were transmitted linked to a unique device ID (i.e. pseudonymously) or without identifiers for the device (i.e. anonymously). Note that we are also using the term "IDFA" for the Android advertising ID here.

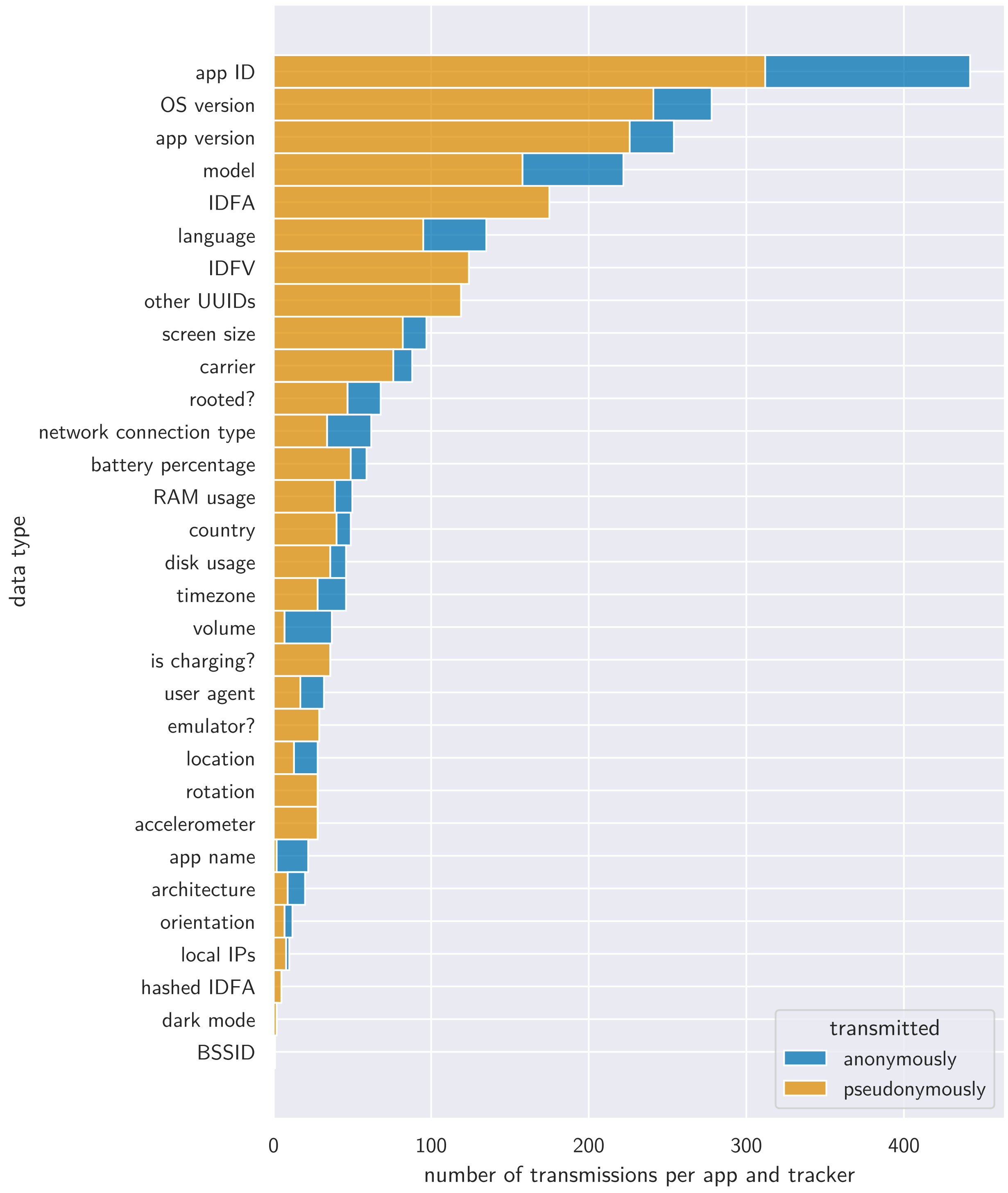
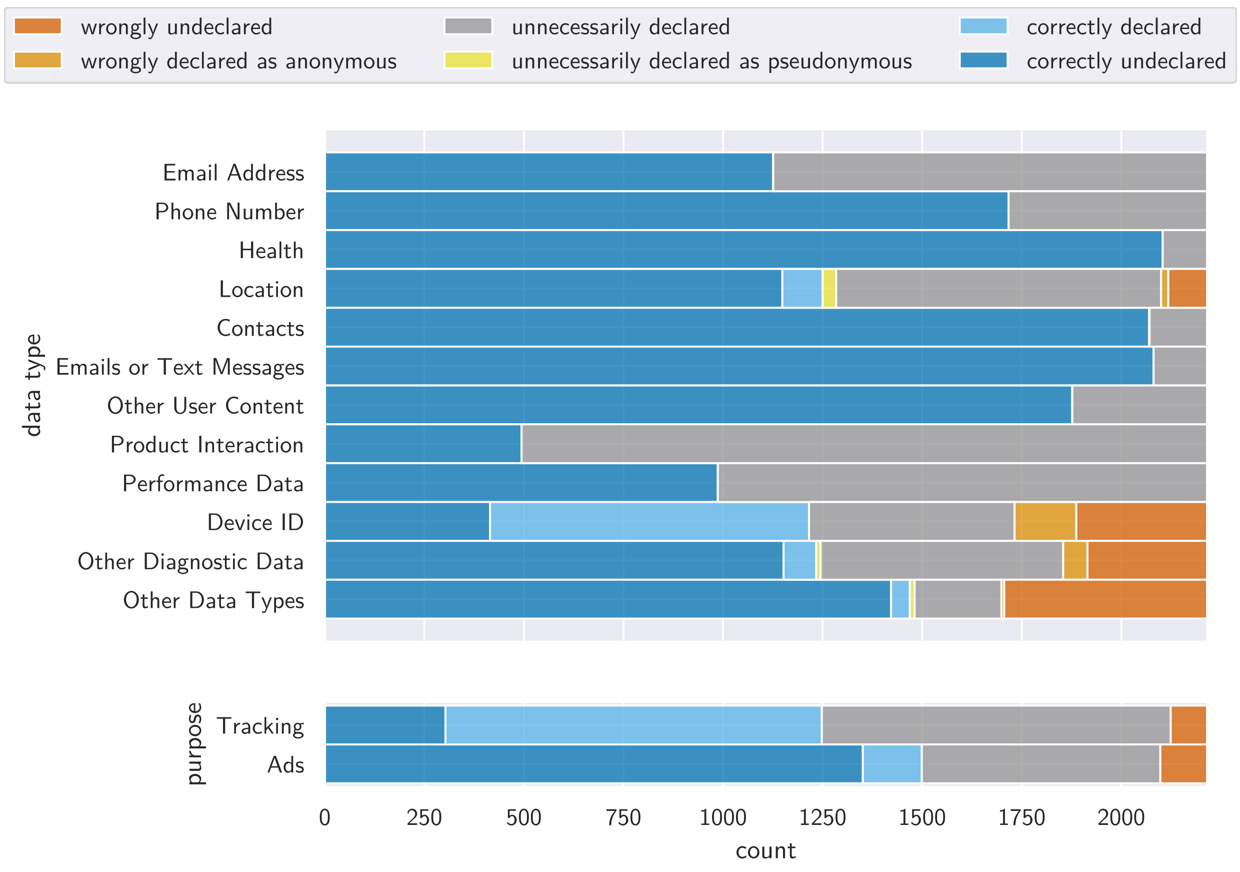


Figure 7.8.: Evaluation of the correctness of data types and purposes in privacy labels on iOS. Remember that we can only definitively say when data *is* collected but if we don't observe data being transmitted, it does not necessarily mean that it is never collected.



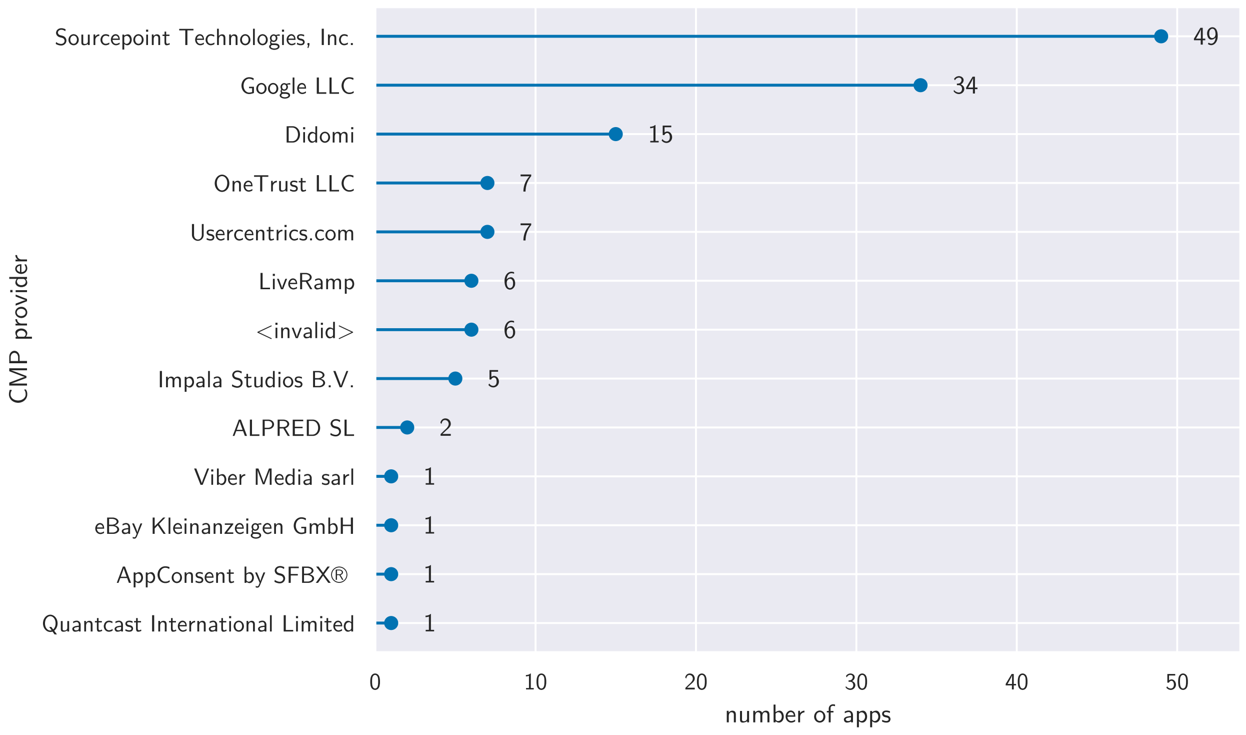
7.5. Apple Privacy Labels

112 of the 2,481 apps on iOS (4.51 %) had an empty privacy label. 182 of them (7.68 %) claimed not to collect any data.

Figure 7.8 shows the comparison of the observed and declared data types and purposes. For most of the data types that we can check, we did not observe apps that incorrectly omitted them from their privacy label or misdeclared them as anonymous. Most notably, we saw 329 apps (13.26 %) that transmitted the IDFA, IDFV, or a hashed version thereof without declaring that in their privacy label. Further, 155 apps (6.25 %) claimed to collect such a device ID in a way that is not linked to the user, which seems like an obvious contradiction. 98 apps (3.95 %) also transmitted the device's location but omitted that in their privacy label and a further 18 apps (0.73 %) declared that they only collected the location anonymously even though we observed them linking it to a unique identifier for the user or device.

In terms of the purposes, most apps correctly declared whether they used ads and tracking. 118 (4.76 %) and 92 apps (3.71 %) wrongly claimed not to use ads and tracking, respectively, despite doing so after all.

Figure 7.9.: Prevalence of CMP providers according to IAB TCF data.



7.6. IAB TCF data

163 of the analysed apps have saved `IABTCF` preferences (64 on Android, and 99 on iOS). Of those, 61 were not detected as having a consent dialog by our approach. Manually analysing those showed that 17 do in fact show a dialog that we did not detect but the remaining 44 do not. It could be that those only show a dialog later in the user flow or maybe they include CMP libraries without actually using them.
Conversely, 282 apps were detected as showing a dialog but have not saved `IABTCF` preferences, confirming our assumption that only relying on TCF data for the analysis would not have been viable (cf. Section 4.2.1).

24 apps only saved `IABTCF` preferences after accepting or rejecting the dialog but not initially, the remaining 138 saved them even without any interaction with the consent dialog.

The apps most often set the `IABTCF_gdprApplies` property, with 125 apps setting the property initially, another 27 only setting it after accepting the dialog, and one app setting it only after rejecting. In total, 145 apps determine the GDPR to be applicable, 6 apps (incorrectly) determine it not to be, and 2 apps set non-spec-compliant values[2]. None of the apps changed their determination after accepting or rejecting the dialog.

`IABTCF_CmpSdkID` specifies which CMP is being used and is set by 111 apps, with six apps specifying an invalid value. Figure 7.9 shows the distribution of the different CMP providers. In our dataset, Sourcepoint[3] and Google's Funding Choices[4] are the most used CMPs by far.

[2]The values in question are `-5828135500133229487` and `-6437494263561806870`, with both apps being on iOS coming from the same vendor (`de.prosiebensat1digital.sat1` and `de.prosiebensat1digital.prosieben`). All other `IABTCF` properties these two apps set were either empty or also nonsensical.

[3]`https://www.sourcepoint.com/cmp/`

[4]`https://blog.google/products/admanager/helping-publishers-manage-consent-funding-choices/`

`IABTCF_PublisherCC` specifies the app publisher's country. 62 apps are from Germany according to this, for 22 the CMP didn't know the country, seven are from the US, five from the Netherlands, and three from Spain. The following countries are each represented once: France, Hong Kong, Luxembourg, Japan, United Kingdom, and Australia.

Finally, using `IABTCF_TCString`, it is possible to determine the exact consent state the apps are saving. We have collected the accepted state for 60 apps. The TCF allows apps to request consent for ten different purposes like "Store and/or access information on a device" or "Measure ad performance". Most apps store consent for all ten purposes, with an average of 9.10 and a median of 10. Apps can also request consent for vendors, with 860 possible vendors on the global vendor list[5] as of the time of writing. The average for the amount of stored vendor consents is 361.75, the median is 158. All possible vendors were requested by at least seven apps. Table 7.2 lists the vendors that more than 45 apps stored consent for.

Table 7.2.: Counts of vendors apps saved consent for according to IAB TCF data. Only vendors requested by more than 45 apps are included.

Vendor	Count
Google Advertising Products	52
The Trade Desk	50
Smart Adserver	50
Adform	50
Flashtalking, Inc.	50
Amazon Advertising	50
RTB House S.A.	49
Yahoo EMEA Limited	49
Xandr, Inc.	49
Magnite, Inc.	49
Sizmek by Amazon	49
OpenX	49
PubMatic, Inc.	49
MediaMath, Inc.	49
Criteo SA	49
Meetrics GmbH	49
SpotX, Inc	49
advanced store GmbH	49
Publicis Media GmbH	49
TabMo SAS	49
Exactag GmbH	49
Otto (GmbH & Co KG)	49
Index Exchange, Inc.	48
Amobee Inc.	48
Active Agent (ADITION technologies GmbH)	48
emetriq GmbH	48
AudienceProject Aps	48
ADITION technologies GmbH	47
Taboola Europe Limited	47

[5] `https://vendor-list.consensu.org/v2/archives/vendor-list-v139.json`

Vendor	Count
Yieldlab AG	47
Platform161 B.V.	47
Improve Digital	47
Semasio GmbH	46
LiveRamp, Inc.	46
Teads	46
Mobile Professionals BV	46
Adobe Audience Manager, Adobe Experience Platform	46
Online Solution	46

The TC string also encodes the language of the consent dialog. Of the 68 apps that initially store a TC string, 63 showed an English consent dialog (our devices were set to English), and five showed a dialog in German.

There is also an older, deprecated TCF specification specifically for mobile apps, the *Mobile In-App CMP API v1.0*[6], which uses `IABConsent` as the prefix for the saved preferences. Only four apps set preferences for this specification without also setting `IABTCF` preferences for the new TCF 2.0 specification. Of those, three only set `IABConsent_SubjectToGDPR` (with one wrongly determining the GDPR not to be applicable), disregarding empty properties. One app additionally set `IABConsent_CMPPresent` to `true` but did not actually show a consent dialog.

7.7. Validation

For each app, we saved a screenshot immediately after all elements on screen had been analysed to allow us to validate the results afterwards. Apps can prevent screenshots from being taken [152], in these cases we were not able to take one. This was the case for 42 apps on Android and 50 apps on iOS.

Table 7.3.: Counts of wrong classifications from manually validating a random set of 250 apps.

Detected	Actual	Count
neither	link	1
neither	notice	2
neither	dialog	15
link	notice	2
link	dialog	5

We manually validated the classification for a random set of 250 apps with screenshots. Table 7.3 shows the results of this validation. Notably, we did not encounter a single false positive (meaning detecting something as a consent element that isn't actually one). All classifications were either correct or our analysis missed the consent elements. 25 of the 250 classifications were false negatives.

[6] `https://github.com/InteractiveAdvertisingBureau/GDPR-Transparency-and-Consent-Framework/blob/b7164119d6b281ac0efb06cb9717e0793fc1f9d0/Mobile%20In-App%20Consent%20APIs%20v1.0%20Final.md`

The discovered false negatives are expected and do not impact the validity of the detected violations. As explained in Section 4.2.3, our approach necessarily misses consent elements due to more detailed information to base an analysis on not being sufficiently available in mobile apps. Not detecting a consent dialog does not cause us to wrongly attribute violations to an app. In these cases, all detected tracking has happened without any user interaction. This means that the apps, regardless of whether a consent dialog is being shown on screen, cannot have obtained valid consent and thus have no legal basis for the tracking. We only perform detection of the other violations in apps where we detected a consent dialog.

We also manually validated all cases where we detected the “accept” button having a significantly different colour than the “reject” button, as our approach cannot determine which of the two is actually highlighted compared to the other. We were able to confirm that it is indeed the “accept” button that is highlighted in all cases.

Finally, we manually validated the remaining violations for 25 randomly selected apps. We found no false positives here, either. There was one app where the “accept” button was larger than the “reject” button but we did not detect the violation.

8. Discussion

8.1. Comparison with Results for the Web

Here, we compare our results with the findings from previous research for the web to gauge the differences between the two platforms.

Prevalence of consent elements Eijk et al. found 40.2 % of 1,500 from the top 100 websites across multiple countries having any consent dialog or notice in 2019 [66], and Sanchez-Rola et al. report around 50 % in 2,000 websites the same year [140], while Mehrnezhad reports 91 % in the top 116 EU websites in 2020 [141].
Meanwhile, we found 17.89 % of all apps having a dialog, notice, or link. Even taking into account our observed false negatives and disregarding the result by Mehrnezhad as an outlier likely influenced by the significantly smaller sample size, these numbers are noticeably higher than our results, suggesting that consent elements are less common on mobile than on the web.

Some research also looked at CMP providers according to IAB TCF data. Matte et al. found Quantcast, OneTrust, Didomi, and Sourcepoint being the most common CMPs in 2020 [6], while Aerts encountered Didomi, Sourcepoint, LiveRamp, and OneTrust most frequently in Belgian, Dutch, and French sites in 2021 [84].
These results are similar to our findings. We also encountered all of these CMPs on mobile, though with different distributions. Notably, the second most common CMP in our data was Google, which Matte et al. did not encounter at all and Aerts encountered less frequently. This can be explained by the fact the Google first adopted the TCF in July 2020 [153].

Dark patterns in consent dialogs Nouwens et al. found only 12.6 % of consent dialogs having a "reject all" button on the first layer in 2020 [7], while Mehrnezhad found 35.4 % of websites only having an "accept" and no "reject" button the same year [141]. Similarly, Aerts found significantly more first-layer "accept" than "reject" buttons, and they also found the "accept" button commonly having a different colour than the "reject" button in 2021 [84]. Of the violations that noyb sent complaints about in 2021, 81 % were about dialogs on websites with no "reject" button on the first layer and 73 % were about "accept" buttons that were highlighted by colour [8].
In comparison, we found 43.2 % of dialogs on mobile having an "accept" but no "reject" button on the first layer and 31.2 % of dialogs highlighting the "accept" button by colour.

8.2. Limitations

There are some limitations in our approach that need to be considered when looking at the results.

Detection of consent dialogs and dark patterns As explained, our analysis can only provide a lower bound on the prevalence of consent elements in apps and of dark patterns in consent dialogs. This is in part due to the limited tooling for dynamically analysing apps on mobile devices. For one, Appium can

only detect text that apps expose in a machine-readable way, which is not always the case. We especially saw games rendering text as bitmaps that are opaque to Appium and thus missed by us.
Also, Appium can only access a limited amount of element attributes. For example, it cannot extract a link's target URL or even reliably classify an element as a link. This especially impacts our ability to detect privacy policy links, meaning that we have to rely on the link text alone.

Additionally, we are limited by our very general approach of detecting arbitrary consent elements based on string matching, which greatly restricts the details we can extract from apps and forces us to err on the side of caution at the cost of missing consent elements that our strict regexes do not catch (cf. Section 4.2.3).
We also only match strings in English or German, though the impact of that is likely within reason as we only downloaded apps from the top charts for Germany anyway.

Finally, we do not interact with apps beyond their consent dialogs. This means that we can only find consent elements that are displayed on the initial screen after launching the apps and will miss those that are only displayed later in the user flow, e.g. after a first-run wizard.

Possible differences in consent dialog behaviour due to analysis environment Even though we clear all app data between runs, it is possible that apps try to re-identify the device for example using our IP address or through fingerprinting, which could impact our results on changes between the user accepting and rejecting the dialog. New techniques continue outsmarting fingerprinting protections in browsers time and time again [154–156] and mobile devices are no strangers to fingerprinting, either [157; 158]. Given the vast array of vectors for fingerprinting and the fact that there likely even exist many that are not publicly known, we cannot possibly control all of them and thus have to assume that apps are able to track us even across resets.

In addition, we deny the iOS permission for tracking across other companies' apps and websites as apps might otherwise construe that as consent (regardless of whether it would be legally). Conversely, it is however possible that apps interpret us denying this permission as a refusal of consent and thus don't display a consent dialog. While this would actually be in the users' interest, it would skew our results on the prevalence of consent dialogs as we would miss those only shown when the tracking permission is granted.

App instrumentation framework We were not able to launch a significant number of apps on Android: 31.66 % of the apps quit immediately or shortly after being started. Manual investigation of a random sample of those revealed that this was caused by objection's certificate pinning bypass. Instead starting the apps without the bypass worked fine. Other Frida scripts for bypassing certificate pinning[1] had the same problem.
We suspect that this is due to some sort of anti-tamper protection built into these apps, perhaps even one built into a common network library and thus automatically included in this many apps. Ultimately, we decided to exclude the affected apps from our analysis rather than running them without a certificate pinning bypass as we feared they would otherwise skew our results with regard to the network traffic and tracking analysis.

It is also possible that the way we run the apps affects their behaviour. Apps can trivially detect that they are running on a rooted/jailbroken device and in an emulator on Android, as well as the

[1]For example these: `https://github.com/httptoolkit/frida-android-unpinning` and `https://codeshare.frida.re/@pcipolloni/universal-android-ssl-pinning-bypass-with-frida/`

fact that we are injecting Frida scripts into them[2]. For example, we saw apps that displayed a screen informing us that we are using a rooted device and refused to function as a result. This was especially common with banking apps.

Similarly, our HTTPS proxy could also alter some apps' behaviour. If an app employs certificate pinning in a way that is not bypassed by objection or SSL Kill Switch 2, the corresponding requests will fail, which may have an effect on the app behaviour.

[2]There are Frida scripts to bypass these checks as well, e.g.: `https://codeshare.frida.re/@dzonerzy/fridantiroot/` and `https://codeshare.frida.re/@enovella/anti-frida-bypass/`

However, we tried to keep the amount of Frida scripts we inject to a minimum because they can in fact break apps as we saw. The only script that we inject into the apps during the actual runs is the certificate pinning bypass on Android. All other scripts are either injected into system processes (clipboard seeding and granting location permission on iOS), where we have confirmed that they do not cause problems, or are only injected into the app after the rest of analysis is done (reading the preferences).

9. Related Work

A lot of research has been done on data protection in websites and apps. We focus this overview on research on consent dialogs in the wild, their design (especially with regard to dark patterns and nudging), and detecting privacy violations through traffic analysis. Interest in these topics has resurged after the GDPR came into force in 2018, with much of recent research using the GDPR as a basis for investigations.

9.1. Analysis of Consent Dialogs and Privacy Policies

Prior research into consent dialogs in the wild has so far been almost exclusively limited to the web. Degeling et al. [143] monitored changes in privacy practices on 6.5k websites between December 2017 and October 2018 to measure the GDPR's impact. They found 16 % of websites adding new privacy polices and 73 % of websites updating their existing ones. They further found an increase in consent dialogs being shown on websites by 16 %. In a follow-up study the same year [1], they manually looked at screenshots of 1,000 consent dialogs on websites and identified eight variables in which they differ, including size and position.

Still in 2019, Eijk et al. [66] detected consent dialogs in websites based on an adblock filter list, finding 40 % of sites having a cookie dialog or notice. They discovered differences in cookie dialog prevalence between different countries (including ones in the EU), with those differences mostly being based on the sites' instead of the users' location. In 2020, Mehrnezhad [141] manually analysed 116 websites and their corresponding mobile apps, finding consent dialogs in 91 % of websites but only in 35 % of apps.

Touching on the legal aspects, a 2020 paper by Matte et al. [57] studied the purposes of the IAB TCF and their usage by advertisers. They found that several of the purpose descriptions were not specific enough and that many advertisers relied on a questionable legitimate interest, thus likely lacking a legal basis for their processing. The same year, Matte et al. [6] also analysed websites using consent dialogs following the TCF. They detected violations like consent being registered without user interaction or even after opt-out, and preselected options in half of the sites. Similarly, Nouwens et al. [7] also found common dark patterns, namely pre-ticked checkboxes, implicitly assumed consent, and rejecting the dialog being harder than accepting it, by scraping 680 from the top 10k sites in the UK for their consent dialog designs using CMP-specific adapters for the five most common ones.

A 2021 thesis by Aerts [84] detected consent dialogs on EU websites also based on filter lists and the TCF. They detected a vast majority of sites either not offering a first-level "reject" button at all or highlighting the "accept" button. In May 2021, consumer protection organisation noyb analysed popular websites using the OneTrust CMP and found violations in more than 500 sites, with the most common ones being sites making it harder to refuse and withdraw consent than giving it and highlighting the "accept" over the "reject" button [8].

For privacy policy analysis, Harkous et al. [159] and Hossein et al. [144] propose machine-learning approaches. Zimmeck et al. specifically analysed the privacy policies of Android apps and compared them against the apps' permissions, API usage, and library inclusion [160].

9.2. Consent Dialog Design and Dark Patterns

Consent fatigue and the effect of nudging on consent rates are well established in relevant literature: Users will often use the easiest choice to dismiss a consent dialog as quickly as possible, without thinking about what it entails. Companies commonly exploit this by highlighting the most privacy-invasive choice and making it harder to choose anything else.

A 2010 experiment by Böhme et al. [161] already found that users are trained to accept EULAs and tend to blindly accept anything that resembles them. They further found that button labels had the largest effect on consent rates, with button labels indicating an actual choice resulting in much fewer opt-ins. A 2013 experiment by Bauer et al. [162] investigated users' perception of and willingness to use "log in with <social network>" functionality, also looking at the dialogs that prompt for consent to forward the user's information to the requesting website. They confirmed that consent dialogs don't work as a medium for conveying privacy-critical information to users, even when those users are concerned about their privacy. And in 2016, Bösch et al. [163] introduced a series of dark patterns common in websites and apps to deceive users into agreeing to more privacy-invasive practices than they actually want.

There is also already a plethora of work on the human aspects on consent dialogs specifically under the GDPR. Experiments by both Utz et al. in 2019 [1] and Nouwens et al. in 2020 [7] investigated the effect of various design variables in consent dialogs on user choices and found that nudging, even in the form of seemingly small details, heavily increases consent rates.

Research on this topic also frequently discusses the ethical aspects of consent dialogs. A 2020 experiment by Machuletz et al. [2] that explored users' interactions with consent dialogs again found that a highlighted "accept all" button results in significantly higher consent rates but at the same time users are not aware of its effects and regret their choice after being informed of them. Based on that, they call the morality and legitimacy of "accept all" buttons into question. And a 2021 experiment by Graßl et al. [4] found most participants agreeing to all consent requests regardless of dark patterns. They hypothesise that nudging and dark patterns in consent dialogs have for a long time been so common that people became conditioned to them and their behaviour is influenced even in the absence of nudges. Conversely, they found that "bright patterns", i.e. nudging towards the privacy-friendly option, was effective in making people choose that.

A 2018 report by the Norwegian Consumer Council analyses dark patterns in Facebook, Google, and Windows 10 [3]. Fassl et al. compiled a literature review in 2021 [164], also looking at potential solutions to the discovered problems. Still in the same year, Gray et al. [5] presented a transdisciplinary *interaction criticism* of dark patterns in consent dialogs, providing arguments for further policy refinement and advancement.

9.3. Traffic Analysis of Websites and Apps

Finally, we look at research that analyses the traffic of websites and apps to detect privacy violations. The GDPR coming into force has resulted in a decrease in cookie and third-party tracking use, to a certain degree also profiting people in non-EU countries, as reported by Hu et al. and Dabrowski et al. in 2019 [165; 166]. Nonetheless, tracking is still common. Also in 2019, Trevisan et al. introduced CookieCheck [11], a tool that visits websites without providing consent and checks whether tracking cookies (as classified by Ghostery [167] and Disconnect [168]) have been set. They found that 74 % of websites still use third-party cookies. Increasingly, websites also use “post-cookie” tracking techniques like fingerprinting and tracking ID synchronisation. A 2021 study by Papadogiannakis et al. [139] found websites using those with 75 % of the detected tracking happening before the user interacted with a consent prompt or even after explicitly rejecting it.

Looking at the Android ecosystem, a 2018 study by Ren et al. [169] compared various versions of 512 Android apps across eight years, finding increased collection of personal data over time. In 2020, Liu et al. implemented MadDroid, a framework for automatically detecting malicious ad content in Android apps and found 6 % of apps showing malicious ads. And a 2021 paper by Nguyen et al. [12] presented an analysis of the network traffic of 86k Android apps with no interaction. They classified the contacted domains by whether they belong to third-party trackers and found that 25k apps contacted trackers without consent. They also performed a survey among the violating app developers, finding common misconceptions about controllers’ obligations under the GDPR.
The Exodus Privacy project [13] has collected a list of tracker signatures, using static analysis to check for their presence in Android apps. The TrackerControl app [170] uses a local VPN, allowing Android users an insight into which domains their apps contact.

Meanwhile on iOS, until recently, research into privacy violations by apps, has been scarce and outdated. A 2011 paper by Egele et al. [171] explored using static analysis on app binaries to detect privacy leaks and a 2015 paper by Dehling et al. [172] crawled App Store pages of health apps.
This changed in 2021 with AppChk by Geier et al. [14], an iOS app that also utilises a local VPN to them to collect the domains contacted by apps. Starting with iOS 15.2, Apple has since integrated the *App Privacy Report*, a similar feature, into the operating system itself [173]. Most recently, a 2022 study by Kollnig et al. [15] compared the privacy practices of 12k apps each on Android and iOS. They found widespread violations against applicable privacy regulation, with little difference between the platforms.

10. Conclusion

In this thesis, we explored the implementation of consent dialogs in mobile apps. We looked at the legal framework that regulates data protection in mobile apps, identifying the GDPR and ePD as the primary laws which limit the processing that apps can perform and mandate that apps need the user's informed consent in order to legally perform tracking.

We then established the strict criteria for a legally compliant consent dialog stemming from the GDPR, which prohibit dark patterns and nudging, e.g. prioritising the "accept" button, using vague purpose descriptions like "to improve user experience", making it harder to refuse consent than accept it, and preselecting purposes. Any consent dialog that violates even just one of these criteria cannot capture valid consent, leaving the app without a legal basis for the data processing.

Looking at consent dialogs in the wild, we found that while websites commonly make use of the IAB TCF standard to implement consent dialogs, this is not the case on mobile, with only around 3 % of apps adhering to the TCF. Similarly, apps less frequently use off-the-shelf CMP solutions compared to the web.

To perform a large-scale analysis of apps on Android and iOS, we developed a device instrumentation framework that can manage apps, set app permissions and extract app preferences using Frida, collect the device network traffic through mitmproxy (including HTTPS and certificate-pinned traffic thanks to objection and SSL Kill Switch 2) while an app is running, as well as analyse and interact with elements displayed on screen by leveraging Appium. We also showed how to collect top chart data for the Google Play Store and the Apple App Store. To actually download the apps, we used PlaystoreDownloader for Android and extended IPATool with support for purchasing apps for iOS.

For the actual detection of references to data protection in the apps, we employed a list of regexes to find the relevant elements and distinguished the discovered references between dialogs, notices, and links. In the detected dialogs, we checked for ambiguous button labels, missing "reject" buttons, "accept" buttons highlighted by colour or size, and apps that quit after refusing consent. While running the apps, we recorded their network traffic and extracted the transmitted data using endpoint-specific tracking requests adapters and indicator matching.

Finally, we evaluated the collected data and found that more than 30 % of the total network traffic was tracking, with Google and Facebook being the most prevalent tracking companies by far. Almost 73 % of apps sent requests containing pseudonymous data even before any user interaction.
We further found that around 18 % of apps displayed any consent element on screen in the first run, with 8.75 % showing a consent dialog, 4.44 % showing a notice, and 4.69 % showing a link to a privacy policy. Of those dialogs, more than 90 % exhibited at least one of the dark patterns we detect, missing "reject" buttons being the most common, followed by ambiguous button labels and "accept" buttons that were highlighted by colour. We did however not observe all of the apps with a dialog transmitting pseudonymous data. In total, we found 77.34 % of the apps with a dialog employing dark patterns *and* transmitting pseudonymous data, thus violating the GDPR. Given the nature of our approach, these numbers are only a lower bound and there are likely even more violations.

Given that the vast majority of consent dialogs are in direct violation of the GDPR, it seems fair to conclude that the GDPR is not at fault for the flood of annoying consent dialogs and that it is in fact the lack of enforcement of the GDPR that allows these user-hostile practices to continue.

10.1. Future Work

The analysis presented in this thesis gave a first snapshot into consent dialogs in mobile apps as of early 2022. Future research could monitor changes over time as has already been done for the web. It could be especially interesting to monitor how Google following Apple's footsteps with the potential to opt-out of the advertising ID and the introduction of a privacy label counterpart influences the situation on Android. We hope that we can contribute to that by releasing our source code.

We encountered limitations restricting us to presenting lower bounds, and subsequent work should tackle the challenges identified in the previous chapter. The certificate pinning bypass problems on Android are especially unfortunate and force us to exclude a significant amount of apps from analysis. Improving certificate pinning bypasses could also benefit research into other subjects on mobile. In addition, there are hopefully more reliable ways to inspect app elements and extract more details about consent dialogs.

Going further, we only interacted with consent dialogs, which means that the vast majority of app functionality is left untouched by us. App interaction beyond consent dialogs for data protection research could not rely on naive monkey testing but would need to be context-aware to avoid acidentally granting unwanted consent.

Finally and orthogonal to our research, until enforcement catches up with the law and progress in tracking technology, defenses against tracking in general and metadata extraction in particular are needed to preserve users' privacy.

11. Bibliography

[1] C. Utz, M. Degeling, S. Fahl, F. Schaub, and T. Holz, “(Un)informed Consent: Studying GDPR Consent Notices in the Field,” in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, New York, NY, USA, 2019, pp. 973–990, doi: 10.1145/3319535.3354212.

[2] D. Machuletz and R. Böhme, “Multiple Purposes, Multiple Problems: A User Study of Consent Dialogs after GDPR,” *Proceedings on Privacy Enhancing Technologies*, vol. 2020, no. 2, pp. 481–498, Apr. 2020, doi: 10.2478/popets-2020-0037.

[3] Forbrukerrådet, “Deceived by design,” Jun. 2018. `https://fil.forbrukerradet.no/wp-content/uploads/2018/06/2018-06-27-deceived-by-design-final.pdf`. [Accessed: 26-Apr-2022]

[4] P. Graßl, H. Schraffenberger, F. Z. Borgesius, and M. Buijzen, “Dark and Bright Patterns in Cookie Consent Requests,” *Journal of Digital Social Research*, vol. 3, no. 1, pp. 1–38, Feb. 2021, doi: 10.33621/jdsr.v3i1.54.

[5] C. M. Gray, C. Santos, N. Bielova, M. Toth, and D. Clifford, “Dark Patterns and the Legal Requirements of Consent Banners: An Interaction Criticism Perspective,” in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, New York, NY, USA, 2021, pp. 1–18, doi: 10.1145/3411764.3445779. `https://hal.inria.fr/hal-03117307/document`

[6] C. Matte, N. Bielova, and C. Santos, “Do Cookie Banners Respect my Choice? : Measuring Legal Compliance of Banners from IAB Europe’s Transparency and Consent Framework,” in *2020 IEEE Symposium on Security and Privacy (SP)*, 2020, pp. 791–809, doi: 10.1109/SP40000.2020.00076.

[7] M. Nouwens, I. Liccardi, M. Veale, D. Karger, and L. Kagal, “Dark Patterns after the GDPR: Scraping Consent Pop-ups and Demonstrating their Influence,” in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, New York, NY, USA, 2020, pp. 1–13, doi: 10.1145/3313831.3376321. `https://people.csail.mit.edu/ilaria/papers/Midas-MITCHI2020.pdf`

[8] noyb.eu, “noyb aims to end ‘cookie banner terror’ and issues more than 500 GDPR complaints,” 31-May-2021. `https://noyb.eu/en/noyb-aims-end-cookie-banner-terror-and-issues-more-500-gdpr-complaints`. [Accessed: 28-Oct-2021]

[9] Irish Council for Civil Liberties, “GDPR enforcer rules that IAB Europe’s consent popups are unlawful,” 05-Feb-2022. `https://www.iccl.ie/news/gdpr-enforcer-rules-that-iab-europes-consent-popups-are-unlawful/`. [Accessed: 29-Apr-2022]

[10] Gegevensbeschermingsautoriteit Litigation Chamber, “Decision 21/2022 on case number DOS-2019-01377,” 02-Feb-2022. `https://www.gegevensbeschermingsautoriteit.be/publications/beslissing-ten-gronde-nr.-21-2022-english.pdf`. [Accessed: 29-Apr-2022]

[11] M. Trevisan, S. Traverso, E. Bassi, and M. Mellia, “4 Years of EU Cookie Law: Results and Lessons Learned,” *Proceedings on Privacy Enhancing Technologies*, vol. 2019, no. 2, pp. 126–145, Apr. 2019, doi: 10.2478/popets-2019-0023.

[12] T. T. Nguyen, M. Backes, N. Marnau, and B. Stock, “Share First, Ask Later (or Never?) Studying Violations of GDPR’s Explicit Consent in Android Apps,” presented at the 30th USENIX Security Symposium (USENIX Security 21), 2021, pp. 3667–3684. `https://www.usenix.org/system/files/sec21-nguyen.pdf`

[13] Exodus contributors, “What Exodus Privacy does,” *Exodus Privacy*, 23-Jul-2020. https://exodus-privacy.eu.org/en/page/what/. [Accessed: 18-Apr-2022]

[14] O. Geier and D. Herrmann, “The AppChk Crowd-Sourcing Platform: Which Third Parties are iOS Apps Talking To?” in *ICT Systems Security and Privacy Protection*, Cham, 2021, pp. 228–241, doi: 10.1007/978-3-030-78120-0_15. https://arxiv.org/pdf/2104.06167.pdf

[15] K. Kollnig, A. Shuba, R. Binns, M. V. Kleek, and N. Shadbolt, “Are iPhones Really Better for Privacy? A Comparative Study of iOS and Android Apps,” *Proceedings on Privacy Enhancing Technologies*, vol. 2022, no. 2, pp. 6–24, Apr. 2022, doi: 10.2478/popets-2022-0033.

[16] B. Altpeter and M. Wessels, “Do they track? Automated analysis of Android apps for privacy violations,” 09-Mar-2021. https://benjamin-altpeter.de/doc/presentation-android-privacy.pdf. [Accessed: 02-May-2022]

[17] B. Altpeter, “iOS watching you: Automated analysis of ‘zero-touch’ privacy violations under iOS,” 03-Aug-2021. https://benjamin-altpeter.de/doc/presentation-ios-privacy.pdf. [Accessed: 02-May-2022]

[18] European Data Protection Board, “Guidelines 2/2019 on the processing of personal data under Article 6(1)(b) GDPR in the context of the provision of online services to data subjects,” Version 2.0, Oct. 2019. https://edpb.europa.eu/sites/default/files/files/file1/edpb_guidelines-art_6-1-b-adopted_after_public_consultation_en.pdf. [Accessed: 10-Apr-2022]

[19] Der Landesbeauftragte für den Datenschutz und die Informationsfreiheit Baden-Württemberg, “FAQ: Cookies und Tracking durch Betreiber von Webseiten und Hersteller von Smartphone-Apps,” Mar. 2022. https://www.baden-wuerttemberg.datenschutz.de/wp-content/uploads/2022/03/FAQ-Tracking-online.pdf. [Accessed: 14-Mar-2022]

[20] Konferenz der unabhängigen Datenschutzaufsichtsbehörden des Bundes und der Länder, “Orientierungshilfe der Aufsichtsbehörden für Anbieter:innen von Telemedien ab dem 1. Dezember 2021,” OH Telemedien 2021, Dec. 2021. https://www.datenschutzkonferenz-online.de/media/oh/20211220_oh_telemedien.pdf. [Accessed: 14-Mar-2022]

[21] Apple Inc., “App Privacy Details,” *Apple Developer*, 11-Nov-2021. https://developer.apple.com/app-store/app-privacy-details/. [Accessed: 28-Apr-2022]

[22] Apple Inc., “User Privacy and Data Use,” *Apple Developer*, 2021. https://developer.apple.com/app-store/user-privacy-and-data-use/. [Accessed: 29-Apr-2022]

[23] E. Laziuk, “iOS 14.5 Opt-in Rate - Daily Updates Since Launch,” 29-Apr-2021. https://www.flurry.com/blog/ios-14-5-opt-in-rate-att-restricted-app-tracking-transparency-worldwide-us-daily-latest-update/. [Accessed: 29-Apr-2022]

[24] D. Newman, “Apple, Meta And The $10 Billion Impact Of Privacy Changes,” *Forbes*, 10-Feb-2022. https://www.forbes.com/sites/danielnewman/2022/02/10/apple-meta-and-the-ten-billion-dollar-impact-of-privacy-changes/. [Accessed: 29-Apr-2022]

[25] Vungle, Inc., “What Is Limit Ad Tracking on Android?” 11-Feb-2022. https://vungle.com/blog/limit-ad-tracking-android/. [Accessed: 29-Apr-2022]

[26] S. Frey, “Get more information about your apps in Google Play,” 26-Apr-2022. https://blog.google/products/google-play/data-safety/. [Accessed: 29-Apr-2022]

[27] M. Schrems *et al.*, "GDPRhub style guide," *GDPRhub*, 08-Mar-2022. https://gdprhub.eu/index.php?title=GDPRhub_style_guide&oldid=24397. [Accessed: 30-Apr-2022]

[28] European Court of Justice, Patrick Breyer v Bundesrepublik Deutschland (Case C-582/14), 19-Oct-2016, ECLI:EU:C:2016:779. https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:62014CJ0582

[29] L. Rocher, J. M. Hendrickx, and Y.-A. de Montjoye, "Estimating the success of re-identifications in incomplete datasets using generative models," *Nat Commun*, vol. 10, no. 1, p. 3069, Jul. 2019, doi: 10.1038/s41467-019-10933-3.

[30] lschatzkin, W. Budington, maximillianh, and C. Antaki, "About Cover Your Tracks," 03-Apr-2021. https://coveryourtracks.eff.org/about. [Accessed: 13-Apr-2022]

[31] T. Blair, P. Campbell Jr., and V. Catanzaro, "The eData Guide to GDPR: Anonymization and Pseudonymization Under the GDPR," *JD Supra*, 09-Dec-2019. https://www.jdsupra.com/legalnews/the-edata-guide-to-gdpr-anonymization-95239/. [Accessed: 13-Apr-2022]

[32] A. K. Muniz Schiebert and B. Altpeter, "Territorial scope of the GDPR," 04-Aug-2020. https://www.datarequests.org/blog/gdpr-territorial-scope/. [Accessed: 10-Apr-2022]

[33] European Court of Justice, Bundesverband der Verbraucherzentralen und Verbraucherverbände - Verbraucherzentrale Bundesverband e. V. v Planet49 GmbH (Case C-673/17), 01-Oct-2019, ECLI:EU:C:2019:801. https://eur-lex.europa.eu/legal-content/en/TXT/?uri=CELEX:62017CJ0673

[34] European Data Protection Board, "Guidelines 8/2020 on the targeting of social media users," Version 2.0, Apr. 2021. https://edpb.europa.eu/system/files/2021-04/edpb_guidelines_082020_on_the_targeting_of_social_media_users_en.pdf. [Accessed: 11-Apr-2022]

[35] D. Clifford, "EU Data Protection Law and Targeted Advertising: Consent and the Cookie Monster - Tracking the crumbs of online user behaviour," *jipitec*, vol. 5, no. 3, Dec. 2014. http://www.jipitec.eu/issues/jipitec-5-3-2014/4095

[36] Article 29 Data Protection Working Party, "Opinion 04/2012 on Cookie Consent Exemption (WP 194)," 00879/12/EN, Jun. 2012. https://ec.europa.eu/justice/article-29/documentation/opinion-recommendation/files/2012/wp194_en.pdf. [Accessed: 10-Apr-2022]

[37] S. Hessel, "In force as of 1 December 2021: the TTDSG and the 'new' cookie rules," Nov-2021. https://www.reuschlaw.de/en/news/in-force-as-of-1-december-2021-the-ttdsg-and-the-new-cookie-rules/. [Accessed: 10-Apr-2022]

[38] M. Schrems, K. Bygnes, A. Dahi, FA, and Gb, "CJEU - C-673/17 - Planet49," *GDPRhub*, 24-Mar-2022. https://gdprhub.eu/index.php?title=CJEU_-_C-673/17_-_Planet49&oldid=24953. [Accessed: 10-Apr-2022]

[39] European Commission Directorate-General for Communication, "Adequacy decisions," 2021. https://ec.europa.eu/info/law/law-topic/data-protection/international-dimension-data-protection/adequacy-decisions_en. [Accessed: 10-Apr-2022]

[40] European Court of Justice, Data Protection Commissioner v Facebook Ireland Limited and Maximillian Schrems (Case C-311/18), 16-Jul-2020, ECLI:EU:C:2020:559. https://eur-lex.europa.eu/legal-content/en/TXT/?uri=CELEX:62018CJ0311

[41] C. Fennessy, "The 'Schrems II' decision: EU-US data transfers in question," 16-Jul-2020. https://iapp.org/news/a/the-schrems-ii-decision-eu-us-data-transfers-in-question/. [Accessed: 10-Apr-2022]

[42] European Court of Justice, Maximillian Schrems v Data Protection Commissioner (Case C-362/14), 06-Oct-2015, ECLI:EU:C:2015:650. https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:62014CJ0362

[43] European Commission, "Trans-Atlantic Data Privacy Framework Factsheet," 25-Mar-2022. https://ec.europa.eu/commission/presscorner/api/files/attachment/872132/Trans-Atlantic%20Data%20Privacy%20Framework.pdf. [Accessed: 10-Apr-2022]

[44] The White House, "FACT SHEET: United States and European Commission Announce Trans-Atlantic Data Privacy Framework," 25-Mar-2022. https://www.whitehouse.gov/briefing-room/statements-releases/2022/03/25/fact-sheet-united-states-and-european-commission-announce-trans-atlantic-data-privacy-framework/. [Accessed: 10-Apr-2022]

[45] European Data Protection Board, "Statement 01/2022 on the announcement of an agreement in principle on a new Trans-Atlantic Data Privacy Framework," 06-Apr-2022. https://edpb.europa.eu/system/files/2022-04/edpb_statement_202201_new_trans-atlantic_data_privacy_framework_en.pdf. [Accessed: 10-Apr-2022]

[46] European Data Protection Supervisor, "#EDPS welcomes, in principle, the announcement from @vonderleyen and @POTUS on the new transatlantic data transfer agreement," *Twitter*, 25-Mar-2022. https://twitter.com/EU_EDPS/status/1507382700575010816. [Accessed: 10-Apr-2022]

[47] noyb.eu, "'Privacy Shield 2.0'? - First Reaction by Max Schrems," 25-Mar-2022. https://noyb.eu/en/privacy-shield-20-first-reaction-max-schrems. [Accessed: 10-Apr-2022]

[48] European Data Protection Board, "Recommendations 01/2020 on measures that supplement transfer tools to ensure compliance with the EU level of protection of personal data," Version 2.0, Jun. 2021. https://edpb.europa.eu/system/files/2021-06/edpb_recommendations_202001vo.2.0_supplementarymeasurestransferstools_en.pdf. [Accessed: 10-Apr-2022]

[49] M. Blocher *et al.*, "DSB (Austria) - 2021-0.586.257 (D155.027)," *GDPRhub*, 18-Feb-2022. https://gdprhub.eu/index.php?title=DSB_(Austria)_-_2021-0.586.257_(D155.027)&oldid=23449. [Accessed: 10-Apr-2022]

[50] FA, Cms, and M. Blocher, "CNIL (France) - Google Analytics (no case number)," *GDPRhub*, 24-Feb-2022. https://gdprhub.eu/index.php?title=CNIL_(France)_-_Google_Analytics_(no_case_number)&oldid=23759. [Accessed: 10-Apr-2022]

[51] Der Landesbeauftragte für den Datenschutz und die Informationsfreiheit Baden-Württemberg, "Orientierungshilfe: Was jetzt in Sachen internationaler Datentransfer?" Sep. 2021. https://www.baden-wuerttemberg.datenschutz.de/wp-content/uploads/2021/10/OH-int-Datentransfer.pdf. [Accessed: 10-Apr-2022]

[52] Commission Nationale de l'Informatique et des Libertés, "Refuser les cookies doit être aussi simple que de les accepter : une vingtaine d'organismes mis en demeure," 25-May-2021. https://www.cnil.fr/fr/cookies-une-vingtaine-organismes-mis-en-demeure. [Accessed: 07-Apr-2022]

[53] Commission Nationale de l'Informatique et des Libertés, "Cookies et autres traceurs : la CNIL publie des lignes directrices modificatives et sa recommandation," 01-Oct-2020. https://www.cnil.fr/fr/cookies-et-autres-traceurs-la-cnil-publie-des-lignes-directrices-modificatives-et-sa-recommandation. [Accessed: 07-Apr-2022]

[54] Rose and FD, "Datatilsynet (Denmark) - 2021-431-0125," *GDPRhub*, 27-Oct-2021. https://gdprhub.eu/index.php?title=Datatilsynet_(Denmark)_-_2021-431-0125&oldid=21008. [Accessed: 07-Apr-2022]

[55] FA, S. Rossetti, C. Villarroel, and Gb, “CNIL (France) - SAN-2021-023,” *GDPRhub*, 06-Apr-2022. `https://gdprhub.eu/index.php?title=CNIL_(France)_-_SAN-2021-023&oldid=25131`. [Accessed: 07-Apr-2022]

[56] European Data Protection Board, “Guidelines 05/2020 on consent under Regulation 2016/679,” Version 1.1, May 2020. `https://edpb.europa.eu/sites/default/files/files/file1/edpb_guidelines_202005_consent_en.pdf`. [Accessed: 14-Mar-2022]

[57] C. Matte, C. Santos, and N. Bielova, “Purposes in IAB Europe’s TCF: which legal basis and how are they used by advertisers?” presented at the APF 2020 - Annual Privacy Forum, 2020, p. 1. `https://hal.inria.fr/hal-02566891`

[58] C. Santos, N. Bielova, and C. Matte, “Are cookie banners indeed compliant with the law? : Deciphering EU legal requirements on consent and technical means to verify compliance of cookie banners,” *Technology and Regulation*, vol. 2020, pp. 91–135, Dec. 2020, doi: 10.26116/techreg.2020.009.

[59] M. Blocher, I. Hahn, A. Dahi, Cvl, and FD, “LG Rostock - 3 O 762/19,” *GDPRhub*, 20-Sep-2021. `https://gdprhub.eu/index.php?title=LG_Rostock_-_3_O_762/19&oldid=19832`. [Accessed: 30-Apr-2022]

[60] Die Landesbeauftragte für den Datenschutz Niedersachsen, “Handreichung: Datenschutzkonforme Einwilligungen auf Webseiten – Anforderungen an Consent-Layer,” Nov. 2020. `https://lfd.niedersachsen.de/download/161158`. [Accessed: 14-Mar-2022]

[61] European Court of Justice, Fashion ID GmbH & Co. KG v Verbraucherzentrale NRW e. V. (Case C-40/17), 29-Jul-2019, ECLI:EU:C:2019:629. `https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:62017CJ0040`

[62] LG Frankfurt am Main, 3-06 O 24/21, 19-Oct-2021, openJur 2021, 44407. `https://openjur.de/u/2378854.html`

[63] Konferenz der unabhängigen Datenschutzaufsichtsbehörden des Bundes und der Länder, “Einwilligung nach der DS-GVO,” Kurzpapier Nr. 20, Feb. 2019. `https://www.datenschutzkonferenz-online.de/media/kp/dsk_kpnr_20.pdf`. [Accessed: 14-Mar-2022]

[64] Bayerisches Landesamt für Datenschutzaufsicht, “Pressemitteilung Länderübergreifende Prüfung: Einwilligungen auf Webseiten von Medienunternehmen sind meist unwirksam,” Jun. 2021. `https://www.lda.bayern.de/media/pm/pm2021_06.pdf`. [Accessed: 14-Mar-2022]

[65] Die Landesbeauftragte für den Datenschutz Niedersachsen, “Telekommunikations-Telemediendatenschutz-Gesetz (TTDSG) – Fragen und Antworten,” Dec-2021. `https://lfd.niedersachsen.de/startseite/infothek/faqs_zur_ds_gvo/faq-telekommunikations-telemediendatenschutz-gesetz-ttdsg-206449.html`. [Accessed: 14-Mar-2022]

[66] R. van Eijk, H. Asghari, P. Winter, and A. Narayanan, “The Impact of User Location on Cookie Notices (Inside and Outside of the European Union),” Social Science Research Network, Rochester, NY, SSRN Scholarly Paper 3361360, Mar. 2019. `https://papers.ssrn.com/abstract=3361360`. [Accessed: 10-Apr-2022]

[67] OneTrust, LLC., “OneTrust Consent Management Platform.” `https://www.onetrust.com/solutions/consent-management-platform/`. [Accessed: 07-Apr-2022]

[68] Usercentrics GmbH, “Website Consent Management Product,” 20-Jan-2022. `https://usercentrics.com/website-consent-management/`. [Accessed: 07-Apr-2022]

[69] Cookiebot, “Cookiebot Consent Management Platform (CMP),” 21-Jan-2020. https://www.cookiebot.com/en/consent-management-platform-cmp-cookiebot-cmp/. [Accessed: 07-Apr-2022]

[70] Piwik PRO SA, “Piwik PRO Consent Management Platform,” 2022. https://piwik.pro/gdpr-consent-manager/. [Accessed: 07-Apr-2022]

[71] noyb.eu, “WeComply! Guide for OneTrust,” May 2021. https://wecomply.noyb.eu/static/app/pdf/OneTrustGuide.766f4ff956c0.pdf. [Accessed: 07-Apr-2022]

[72] M. Hils, D. W. Woods, and R. Böhme, “Privacy Preference Signals: Past, Present and Future,” *Proceedings on Privacy Enhancing Technologies*, vol. 2021, no. 4, pp. 249–269, Oct. 2021, doi: 10.2478/popets-2021-0069.

[73] Kevel, “Consent Management Platform (CMP) 2022 Tracker,” 2022. https://www.kevel.com/cmp/. [Accessed: 29-Mar-2022]

[74] IAB Europe, “About Us,” 2021. https://iabeurope.eu/about-us/. [Accessed: 29-Mar-2022]

[75] IAB Europe, “TCF – Transparency & Consent Framework,” 2021. https://iabeurope.eu/transparency-consent-framework/. [Accessed: 07-Apr-2022]

[76] OneTrust, LLC., “OneTrust PreferenceChoice CMP for TCF v2.0,” 2022. https://www.onetrust.com/solutions/consent-management-platform/iab-tcf-2-0/. [Accessed: 29-Mar-2022]

[77] Usercentrics GmbH, “TCF 2.0 explained - the most important new features,” 07-May-2020. https://usercentrics.com/knowledge-hub/tcf-2-0-explained-new-features-at-a-glance/. [Accessed: 29-Mar-2022]

[78] iubenda s.r.l, “The complete guide to iubenda Consent Management Platform (CMP) and IAB TCF 2.0 & 2.1,” 2020. https://www.iubenda.com/en/help/7440-iab-framework-cmp. [Accessed: 29-Mar-2022]

[79] Cookiebot, “IAB Transparency and Consent Framework (TCF 2.0) | Integration and compliance with Cookiebot CMP,” 02-Mar-2022. https://www.cookiebot.com/en/iab-cookies/. [Accessed: 29-Mar-2022]

[80] IAB Europe, “Join the TCF,” 2021. https://iabeurope.eu/join-the-tcf/. [Accessed: 29-Mar-2022]

[81] IAB Tech Lab, “Transparency and Consent String with Global Vendor & CMP List Formats: IAB Europe Transparency & Consent Framework,” 22-Feb-2022. https://github.com/InteractiveAdvertisingBureau/GDPR-Transparency-and-Consent-Framework/blob/b7164119d6b281ac0efb06cb9717e0793fc1f9d0/TCFv2/IAB%20Tech%20Lab%20-%20Consent%20string%20and%20vendor%20list%20formats%20v2.md. [Accessed: 29-Mar-2022]

[82] IAB Tech Lab, “IAB Europe Transparency and Consent Framework Implementation Guidelines,” 21-Sep-2021. https://github.com/InteractiveAdvertisingBureau/GDPR-Transparency-and-Consent-Framework/blob/b7164119d6b281ac0efb06cb9717e0793fc1f9d0/TCFv2/TCF-Implementation-Guidelines.md. [Accessed: 29-Mar-2022]

[83] IAB Tech Lab, “Consent Management Platform API: IAB Europe Transparency & Consent Framework,” 21-Sep-2021. https://github.com/InteractiveAdvertisingBureau/GDPR-Transparency-and-Consent-Framework/blob/b7164119d6b281ac0efb06cb9717e0793fc1f9d0/TCFv2/IAB%20Tech%20Lab%20-%20CMP%20API%20v2.md. [Accessed: 29-Mar-2022]

[84] K. Aerts, “Cookie Dialogs and their Compliance: The quest for an automated audit process to enhance privacy regulation,” Open Universiteit, 2021. https://www.open.ou.nl/hjo/supervision/2021-koen-aerts-msc-thesis.pdf

[85] The Frida contributors, “Frida Docs,” 25-Feb-2022. https://frida.re/docs/home/. [Accessed: 29-Mar-2022]

[86] Exodus contributors, “Exodus static analysis,” 17-Aug-2018. https://exodus-privacy.eu.org/en/post/exodus_static_analysis/. [Accessed: 07-Apr-2022]

[87] benaneesh, “Answer to ‘How to search public or private API’s in ‘nm‘ tool for all the libraries in the binary .ipa’,” *Stack Overflow*, 23-Sep-2016. https://stackoverflow.com/a/39668318. [Accessed: 07-Apr-2022]

[88] Columbo, “Answer to ‘Find size contributed by each external library on iOS’,” *Stack Overflow*, 17-Aug-2015. https://stackoverflow.com/a/32053076. [Accessed: 07-Apr-2022]

[89] IAB Europe, “CMP List,” 2021. https://iabeurope.eu/cmp-list/. [Accessed: 30-Mar-2022]

[90] Udonis, “Top 14 Consent Management Tools for Mobile Apps and Games,” 24-Jan-2022. https://www.blog.udonis.co/mobile-marketing/mobile-games/consent-management-tools. [Accessed: 30-Mar-2022]

[91] Instabug, “Top Mobile App Consent Management Tools,” 29-Nov-2021. https://instabug.com/blog/top-mobile-app-consent-management-tools/. [Accessed: 30-Mar-2022]

[92] A. Cortesi, M. Hils, and T. Kriechbaumer, *mitmproxy: A free and open source interactive HTTPS proxy.* 2021. https://mitmproxy.org/

[93] O. Spryn, “Automated Android Emulator Setup and Configuration,” 26-Feb-2020. https://proandroiddev.com/automated-android-emulator-setup-and-configuration-23accc11a325. [Accessed: 09-Apr-2022]

[94] Android Open Source Project contributors, “Start the emulator from the command line,” *Android Developers*, 25-Aug-2020. https://developer.android.com/studio/run/emulator-commandline. [Accessed: 09-Apr-2022]

[95] LionCoder, “Answer to ‘ADB Shell Input Events’,” *Stack Overflow*, 13-Dec-2011. https://stackoverflow.com/a/8483797. [Accessed: 09-Apr-2022]

[96] geekdada, “URL Schemes,” *JBguide*, 23-Jan-2016. https://jbguide.me/2016/url-schemes-after-jailbreak. [Accessed: 09-Apr-2022]

[97] Android Open Source Project contributors, “About Android App Bundles,” *Android Developers*, 17-Mar-2022. https://developer.android.com/guide/app-bundle. [Accessed: 02-Apr-2022]

[98] Android Open Source Project contributors, “Android Debug Bridge (adb),” *Android Developers*, 30-Mar-2022. https://developer.android.com/studio/command-line/adb. [Accessed: 09-Apr-2022]

[99] neverever415, “Answer to ‘adb shell command to make Android package uninstall dialog appear’,” *Stack Overflow*, 12-Dec-2012. https://stackoverflow.com/a/13833600. [Accessed: 09-Apr-2022]

[100] I. Latif, “Answer to ‘List of ADB settable permissions’,” *Android Enthusiasts Stack Exchange*, 13-Jan-2020. https://android.stackexchange.com/a/220297. [Accessed: 03-Apr-2022]

[101] K. Johnson, “A deep dive into macOS TCC.db,” 09-Feb-2021. https://www.rainforestqa.com/blog/macos-tcc-db-deep-dive. [Accessed: 09-Apr-2022]

[102] Apple Inc., “CLAuthorizationStatus,” *Apple Developer*, 2022. `https://developer.apple.com/documentation/corelocation/clauthorizationstatus`. [Accessed: 02-May-2022]

[103] L. Jacobs, “Early Instrumentation,” *sensepost/objection Wiki*, 14-Sep-2017. `https://github.com/sensepost/objection`. [Accessed: 09-Apr-2022]

[104] Android Open Source Project contributors, “AAPT2,” *Android Developers*, 25-Aug-2020. `https://developer.android.com/studio/command-line/aapt2`. [Accessed: 09-Apr-2022]

[105] J. Lipps, K. Matsuo, S. Sekar, S. Dixon, M. Virani, and Jonahss, “Introduction,” *Appium Docs*, 27-Feb-2022. `http://appium.io/docs/en/about-appium/intro/?lang=en`. [Accessed: 09-Apr-2022]

[106] K. Matsuo *et al.*, “XCUITest Real Devices (iOS),” *Appium Docs*, 07-Dec-2021. `http://appium.io/docs/en/drivers/ios-xcuitest-real-devices/`. [Accessed: 09-Apr-2022]

[107] I. A. Murchie, K. Matsuo, M. Mokhnach, J. Lipps, and D. Graham, “README.md – appium-xcuitest-driver,” 24-Mar-2022. `https://github.com/appium/appium-xcuitest-driver/blob/f1c2b14473fc6a5ef67d027c1fe2b3896071bf58/README.md`. [Accessed: 04-Apr-2022]

[108] Android Open Source Project contributors, “Configure hardware acceleration for the Android Emulator,” *Android Developers*, 14-Sep-2021. `https://developer.android.com/studio/run/emulator-acceleration`. [Accessed: 08-Apr-2022]

[109] M. Hazard, “Run ARM apps on the Android Emulator,” *Android Developers Blog*, 26-Mar-2020. `https://android-developers.googleblog.com/2020/03/run-arm-apps-on-android-emulator.html`. [Accessed: 08-Apr-2022]

[110] “Installing .ipa/.app inside iOS simulator on Silicon M1,” *r/MacOS*, 01-Aug-2021. `https://www.reddit.com/r/MacOS/comments/ovubxa/installing_ipaapp_inside_ios_simulator_on_silicon/`. [Accessed: 08-Apr-2022]

[111] grg, “Answer to ‘How can I install .ipa file to my iPhone simulator?’” *Ask Different*, 07-Dec-2013. `https://apple.stackexchange.com/a/113055`. [Accessed: 08-Apr-2022]

[112] Appetize.io, LLC, “Uploading apps,” *Appetize.io Docs*, 28-Feb-2022. `https://docs.appetize.io/core-features/uploading-apps`. [Accessed: 02-Apr-2022]

[113] RunThatApp.com, Inc., “RunThatApp File Submission Upload,” 2021. `http://upload.runthatapp.com/upload.html`. [Accessed: 02-Apr-2022]

[114] emiyl, “iPhone 7,” 30-Mar-2022. `https://appledb.dev/device/iPhone-7.html`. [Accessed: 08-Apr-2022]

[115] A. Bouchard, “CoolStar confirms that the upcoming iOS 15 jailbreak will be her last,” 20-Mar-2022. `https://www.idownloadblog.com/2022/03/20/coolstar-confirms-that-the-upcoming-ios-15-jailbreak-will-be-her-last/`. [Accessed: 08-Apr-2022]

[116] PJ09, iAdam1n, and fattyffat, “faq/downgrading,” *r/jailbreak wiki*, Oct-2021. `https://www.reddit.com/r/jailbreak/wiki/faq/downgrading`. [Accessed: 08-Apr-2022]

[117] DanTheMann15, AS967, and IAdam1n, “SHSH,” *The iPhone Wiki*, 08-Apr-2022. `https://web.archive.org/web/20220408121323/https://www.theiphonewiki.com/wiki/SHSH`. [Accessed: 08-Apr-2022]

[118] Apple Inc., “iOS 14.5 offers Unlock iPhone with Apple Watch, diverse Siri voices, and more,” 26-Apr-2021. `https://www.apple.com/newsroom/2021/04/ios-14-5-offers-unlock-iphone-with-apple-watch-diverse-siri-voices-and-more/`. [Accessed: 08-Apr-2022]

[119] L. Jose, "Answer to 'adb remount fails - mount: "system" not in /proc/mounts'," *Stack Overflow*, 20-Jul-2020. https://stackoverflow.com/a/62687021. [Accessed: 03-Apr-2022]

[120] ropnop, "Configuring Burp Suite With Android Nougat," 18-Jan-2018. https://blog.ropnop.com/configuring-burp-suite-with-android-nougat/. [Accessed: 03-Apr-2022]

[121] 0x10f2c, "Intercepting Traffic on Android 9 Pie (Emulated) with Burp Suite," 13-Nov-2019. https://web.archive.org/web/20210126205441/https://anothernetsecblog.com/intercepting-traffic-on-android-9-0/. [Accessed: 03-Apr-2022]

[122] A. Yan, "Answer to 'Captive Portal parameters'," *Android Enthusiasts Stack Exchange*, 27-Nov-2017. https://android.stackexchange.com/a/186995. [Accessed: 03-Apr-2022]

[123] A. Ibanez, "Intercepting Network Traffic with mitmproxy," 20-Nov-2019. https://www.andyibanez.com/posts/intercepting-network-mitmproxy/. [Accessed: 09-Apr-2022]

[124] Appfigures, Inc., "Top Apps for Android on Google Play in Germany," 09-Apr-2020. https://appfigures.com/top-apps/google-play/germany/top-overall. [Accessed: 09-Apr-2022]

[125] Sensor Tower, Inc., "Top Grossing Apps | GERMANY | Top App Store Rankings for Android," 09-Apr-2022. https://app.sensortower.com/android/rankings/top/mobile/germany/overall. [Accessed: 09-Apr-2022]

[126] A. Johnson, "Answer to 'How to get top 400 lists from iTunes'," *Stack Overflow*, 09-May-2015. https://stackoverflow.com/a/30134648. [Accessed: 03-Apr-2022]

[127] Apple Inc., "Genre IDs Appendix – Partner Resources," 20-Sep-2019. https://web.archive.org/web/20190920135004/https://affiliate.itunes.apple.com/resources/documentation/genre-mapping/. [Accessed: 30-Mar-2022]

[128] Apple Inc., "Advanced Partner Linking – Partner Resources," 06-Dec-2019. https://web.archive.org/web/20191206001952/https://affiliate.itunes.apple.com/resources/documentation/linking-to-the-itunes-music-store/. [Accessed: 30-Mar-2022]

[129] Ölbaum, "How iTunes selects the storefront," *Ölbaum's Delirium*, 27-May-2006. https://ithink.ch/blog/2006/05/27/how_itunes_selects_the_storefront.html. [Accessed: 30-Mar-2022]

[130] konsumer, "Answer to 'What does X-Apple-Store-Front means in apple http headers?'" *Stack Overflow*, 09-Nov-2019. https://stackoverflow.com/a/58776183. [Accessed: 03-Apr-2022]

[131] ZhouFyk, "Scrape ratings from iTunes store," 23-Jul-2019. https://gist.github.com/sgmurphy/1878352. [Accessed: 30-Mar-2022]

[132] G. C. Georgiu, rooky-c3bo, and F. Collova, "README.md – PlaystoreDownloader," 08-Jan-2022. https://github.com/ClaudiuGeorgiu/PlaystoreDownloader/blob/58a74a9ab2749f932c32bc8204a3a323d115629e/README.md. [Accessed: 02-Apr-2022]

[133] W. Pearson, "Where iOS Apps Are Stored Locally in Mac OS X and Windows," *OS X Daily*, 15-Dec-2011. https://osxdaily.com/2011/12/15/where-ios-apps-stored-locally-in-mac-os-x-and-windows/. [Accessed: 06-Apr-2022]

[134] Apple Inc., "Deploy apps in a business environment with iTunes," *Apple Support*, 14-Aug-2019. https://support.apple.com/en-us/HT208079. [Accessed: 06-Apr-2022]

[135] 3uTools, "How to Download Apps Using 3uTools?" 19-Jul-2017. http://3u.com/tutorial/articles/126/how-to-download-apps-using-3utools. [Accessed: 06-Apr-2022]

[136] M. Anderson, "How To Download IPA Files for iOS," *AppleToolBox*, 22-Sep-2020. https://appletoolbox.com/download-ipa-files-ios/. [Accessed: 06-Apr-2022]

[137] DigiDNA SARL, “Download, Install & Back up your iOS Apps to Mac and PC,” 01-Apr-2022. `https://imazing.com/ios-app-management`. [Accessed: 06-Apr-2022]

[138] M. Alfhaily *et al.*, *majd/ipatool*. 2022. `https://github.com/majd/ipatool/tree/1b65463007b7e5a160d1c83e32d92f4e18cde6da`. [Accessed: 07-Apr-2022]

[139] E. Papadogiannakis, P. Papadopoulos, N. Kourtellis, and E. P. Markatos, “User Tracking in the Post-cookie Era: How Websites Bypass GDPR Consent to Track Users,” in *Proceedings of the Web Conference 2021*, New York, NY, USA, 2021, pp. 2130–2141, doi: 10.1145/3442381.3450056. `https://publications.ics.forth.gr/_publications/3442381.3450056.pdf`

[140] I. Sanchez-Rola *et al.*, “Can I Opt Out Yet? GDPR and the Global Illusion of Cookie Control,” in *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*, New York, NY, USA, 2019, pp. 340–351. `https://www.eurecom.fr/publication/5941/download/sec-publi-5941.pdf`. [Accessed: 10-Apr-2022]

[141] M. Mehrnezhad, “A Cross-Platform Evaluation of Privacy Notices and Tracking Practices,” in *2020 IEEE European Symposium on Security and Privacy Workshops (EuroS PW)*, 2020, pp. 97–106, doi: 10.1109/EuroSPW51379.2020.00023. `https://eusec20.cs.uchicago.edu/eusec20-Mehrnezhad.pdf`

[142] T. Libert, “An Automated Approach to Auditing Disclosure of Third-Party Data Collection in Website Privacy Policies,” in *Proceedings of the 2018 World Wide Web Conference*, Republic and Canton of Geneva, CHE, 2018, pp. 207–216, doi: 10.1145/3178876.3186087.

[143] M. Degeling, C. Utz, C. Lentzsch, H. Hosseini, F. Schaub, and T. Holz, “We Value Your Privacy ... Now Take Some Cookies: Measuring the GDPR’s Impact on Web Privacy,” in *Proceedings 2019 Network and Distributed System Security Symposium*, San Diego, CA, 2019, doi: 10.14722/ndss.2019.23378.

[144] H. Hosseini, M. Degeling, C. Utz, and T. Hupperich, “Unifying Privacy Policy Detection,” *Proceedings on Privacy Enhancing Technologies*, vol. 2021, no. 4, pp. 480–499, Oct. 2021, doi: 10.2478/popets-2021-0081.

[145] B. J. Lindbloom, “Delta E (CMC),” 07-Apr-2017. `http://www.brucelindbloom.com/index.html?Eqn_DeltaE_CMC.html`. [Accessed: 05-Apr-2022]

[146] J. Kwakman, pareyesv, baudev, n3t, meetinthemiddle-be, and afrancht, *Open Cookie Database*. 2022. `https://github.com/jkwakman/Open-Cookie-Database`. [Accessed: 28-Apr-2022]

[147] L. Holmes, “Searching for Content in Base-64 Strings,” 21-Sep-2017. `https://www.leeholmes.com/searching-for-content-in-base-64-strings/`. [Accessed: 28-Apr-2022]

[148] WaLLy3K, “The Big Blocklist Collection,” 12-Feb-2022. `https://firebog.net/`. [Accessed: 28-Apr-2022]

[149] The EasyList authors, “EasyList - Overview,” 07-Oct-2021. `https://easylist.to/`. [Accessed: 28-Apr-2022]

[150] Exodus contributors, “Exodus tracker investigation platform,” 2022. `https://etip.exodus-privacy.eu.org/trackers/all?tracker_name=&trackers_select=exodus`. [Accessed: 23-Apr-2022]

[151] A. Lex, N. Gehlenborg, H. Strobelt, R. Vuillemot, and H. Pfister, “UpSet: Visualization of Intersecting Sets,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 1983–1992, Dec. 2014, doi: 10.1109/TVCG.2014.2346248.

[152] Android Open Source Project contributors, “WindowManager.LayoutParams,” *Android Developers*, 10-Feb-2022. https://developer.android.com/reference/android/view/WindowManager.LayoutParams. [Accessed: 16-Apr-2022]

[153] A. Cartee, “Google Added to IAB TCF 2.0 Global Vendor List (GVL),” 24-Jul-2020. https://www.cookiepro.com/blog/google-iab-tcf-gvl/. [Accessed: 29-Apr-2022]

[154] G. Acar, C. Eubank, S. Englehardt, M. Juarez, A. Narayanan, and C. Diaz, “The Web Never Forgets: Persistent Tracking Mechanisms in the Wild,” in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, New York, NY, USA, 2014, pp. 674–689, doi: 10.1145/2660267.2660347. https://securehomes.esat.kuleuven.be/~gacar/persistent/the_web_never_forgets.pdf

[155] G. krishna R, “Are you anonymous?” 09-Oct-2021. https://www.nothingprivate.ml/. [Accessed: 20-Apr-2022]

[156] K. Solomos, J. Kristoff, C. Kanich, and J. Polakis, “Tales of Favicons and Caches: Persistent Tracking in Modern Browsers,” in *Proceedings 2021 Network and Distributed System Security Symposium*, Virtual, 2021, doi: 10.14722/ndss.2021.24202.

[157] J. Zhang, A. R. Beresford, and I. Sheret, “SensorID: Sensor Calibration Fingerprinting for Smartphones,” in *2019 IEEE Symposium on Security and Privacy (SP)*, 2019, pp. 638–655, doi: 10.1109/SP.2019.00072.

[158] S. Copland, “Device Fingerprinting For Android Developers,” *FingerprintJS*, 08-Dec-2020. https://fingerprintjs.com/blog/device-fingerprinting-android/. [Accessed: 20-Apr-2022]

[159] H. Harkous, K. Fawaz, R. Lebret, F. Schaub, K. Shin, and K. Aberer, “Polisis: Automated Analysis and Presentation of Privacy Policies Using Deep Learning,” in *USENIX Security Symposium*, 2018. https://www.usenix.org/system/files/conference/usenixsecurity18/sec18-harkous.pdf

[160] S. Zimmeck *et al.*, “MAPS: Scaling Privacy Compliance Analysis to a Million Apps,” *Proceedings on Privacy Enhancing Technologies*, vol. 2019, no. 3, pp. 66–86, Jul. 2019, doi: 10.2478/popets-2019-0037.

[161] R. Böhme and S. Köpsell, “Trained to accept? a field experiment on consent dialogs,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, New York, NY, USA, 2010, pp. 2403–2406, doi: 10.1145/1753326.1753689. https://www1.icsi.berkeley.edu/pubs/networking/trainedto10.pdf

[162] L. Bauer, C. Bravo-Lillo, E. Fragkaki, and W. Melicher, “A comparison of users’ perceptions of and willingness to use Google, Facebook, and Google+ single-sign-on functionality,” in *Proceedings of the 2013 ACM workshop on Digital identity management*, New York, NY, USA, 2013, pp. 25–36, doi: 10.1145/2517881.2517886.

[163] C. Bösch, B. Erb, F. Kargl, H. Kopp, and S. Pfattheicher, “Tales from the Dark Side: Privacy Dark Strategies and Privacy Dark Patterns,” *Proceedings on Privacy Enhancing Technologies*, vol. 2016, no. 4, pp. 237–254, Oct. 2016, doi: 10.1515/popets-2016-0038.

[164] M. Fassl, L. T. Gröber, and K. Krombholz, “Stop the Consent Theater,” in *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, New York, NY, USA, 2021, pp. 1–7, doi: 10.1145/3411763.3451230. https://publications.cispa.saarland/3370/7/Stop_the_Consent_Theater__alt_chi_2021.pdf

[165] X. Hu and N. Sastry, “Characterising Third Party Cookie Usage in the EU after GDPR,” in *Proceedings of the 10th ACM Conference on Web Science*, New York, NY, USA, 2019, pp. 137–141, doi: 10.1145/3292522.3326039. `https://kclpure.kcl.ac.uk/portal/files/110158213/Characterising_Third_Party_Cookie_HU_Accepted6April2019_GREEN_AAM.pdf`

[166] A. Dabrowski, G. Merzdovnik, J. Ullrich, G. Sendera, and E. Weippl, “Measuring Cookies and Web Privacy in a Post-GDPR World,” in *Passive and Active Measurement*, Cham, 2019, pp. 258–270, doi: 10.1007/978-3-030-15986-3_17. `https://www.johannaullrich.eu/assets/papers/dabrowski2019_pam.pdf`

[167] Ghostery GmbH, “We make privacy easy,” 2022. `https://www.ghostery.com/`. [Accessed: 18-Apr-2022]

[168] Disconnect, Inc., “Tracking protection lists,” 2015. `https://disconnect.me/trackerprotection`. [Accessed: 18-Apr-2022]

[169] J. Ren, M. Lindorfer, D. J. Dubois, A. Rao, D. Choffnes, and N. Vallina-Rodriguez, “Bug Fixes, Improvements, ... and Privacy Leaks - A Longitudinal Study of PII Leaks Across Android App Versions,” in *Proceedings 2018 Network and Distributed System Security Symposium*, San Diego, CA, 2018, doi: 10.14722/ndss.2018.23143.

[170] K. Kollnig, “TrackerControl for Android,” *TrackerControl for Android*, 10-Apr-2022. `https://trackercontrol.org/`. [Accessed: 18-Apr-2022]

[171] M. Egele, C. Krügel, E. Kirda, and G. Vigna, “PiOS: Detecting Privacy Leaks in iOS Applications,” in *NDSS*, 2011. `https://www.ndss-symposium.org/wp-content/uploads/2017/09/egel.pdf`

[172] T. Dehling, F. Gao, S. Schneider, and A. Sunyaev, “Exploring the Far Side of Mobile Health: Information Security and Privacy of Mobile Health Apps on iOS and Android,” *JMIR mHealth and uHealth*, vol. 3, no. 1, p. e3672, Jan. 2015, doi: 10.2196/mhealth.3672.

[173] Apple Inc., “About App Privacy Report,” *Apple Support*, 13-Dec-2021. `https://support.apple.com/en-us/HT212958`. [Accessed: 18-Apr-2022]

A. Appendix

A.1. Additional Figures and Tables

Table A.1.: List of the permissions we set on iOS. The labels were determined by setting the permissions on a test app and observing the change in the Settings app.

ID	Label in Settings	Granted?
`kTCCServiceCalendar`	Calendars	granted
`kTCCServiceAddressBook`	Contacts	granted
`kTCCServiceReminders`	Reminders	granted
`kTCCServicePhotos`	Photos	granted
`kTCCServiceMediaLibrary`	Media & Apple Music	granted
`kTCCServiceBluetoothAlways`	Bluetooth	granted
`kTCCServiceMotion`	Motion & Fitness	granted
`kTCCServiceWillow`	Home Data	granted
`kTCCServiceExposureNotification`	Exposure Notifications	granted
`kTCCServiceLiverpool`	*no visible effect*	granted
`kTCCServiceUbiquity`	*no visible effect*	granted
`kTCCServiceCamera`	Camera	denied
`kTCCServiceMicrophone`	Microphone	denied
`kTCCServiceUserTracking`	Allow Tracking	denied

Table A.2.: Matching of the categories on the App Store (iOS) and Google Play Store (Android). Multiple categories in a single row are delimited by semicolons. Not every category is available on both platforms.

iOS	Android
	Auto & Vehicles
	Beauty
Books; Reference	Books & Reference
Business	Business
Catalogues	
	Comics
	Dating
Developer Tools	
Education	Education
Entertainment	Entertainment
	Events
Finance	Finance
Food & Drink	Food & Drink
Games	Games

iOS	Android
Graphics & Design	Art & Design
Health & Fitness	Health & Fitness
	House & Home
	Kids
	Libraries & Demo
Lifestyle	Lifestyle
Medical	Medical
Music	Music & Audio
Navigation	Maps & Navigation
News; Magazines & Newspapers	News & Magazines
	Parenting
	Personalization
Photo & Video	Photography; Video Players & Editors
Productivity	Productivity
Shopping	Shopping
Social Networking	Social; Communication
Sports	Sports
Stickers	
Travel	Travel & Local
Utilities	Tools
	Watch apps
Weather	Weather

A.2. SQL Query to Create the Background Request Filter View

We use the following SQL view to filter out background noise from the operating systems. We compiled the filters by recording the idle traffic of both platforms for several days and then writing queries to filter out all traffic.

```
create view filtered_requests as
select name, platform, version, run_type, requests.*,
  regexp_replace(concat(
    requests.scheme, '://', requests.host, requests.path), '\?.+$', ''
  ) endpoint_url from apps
join runs r on apps.id = r.app join requests on r.id = requests.run where

(
  platform = 'ios'
  and not requests.host = 'albert.apple.com' and not requests.host = 'captive.apple.com'
  and not requests.host = 'gs.apple.com'  and not requests.host = 'humb.apple.com'
  and not requests.host = 'sq-device.apple.com' and not requests.host = 'tbsc.apple.com'
  and not requests.host = 'time-ios.apple.com' and not requests.host = 'time.apple.com'
  and not requests.host ~~ '%.push.apple.com' and not requests.host = 'gdmf.apple.com'
  and not requests.host = 'gg.apple.com' and not requests.host = 'identity.apple.com'
  and not requests.host = 'iprofiles.apple.com' and not requests.host = 'mesu.apple.com'
  and not requests.host = 'appldnld.apple.com' and not requests.host = 'ppq.apple.com'
  and not requests.host = 'xp.apple.com' and not requests.host ~~ '%.itunes.apple.com'
  and not requests.host = 'doh.dns.apple.com' and not requests.host = 'crl.apple.com'
  and not requests.host = 'crl.entrust.net' and not requests.host = 'crl3.digicert.com'
  and not requests.host = 'crl4.digicert.com' and not requests.host = 'ocsp.apple.com'
  and not requests.host = 'ocsp.digicert.com' and not requests.host = 'ocsp.entrust.net'
  and not requests.host = 'ocsp.verisign.net' and not requests.host = 'valid.apple.com'
  and not requests.host = 'ocsp2.apple.com' and not requests.host ~~ '%smoot.apple.com'
  and not requests.host = 'ns.itunes.apple.com' and not requests.host = 'fba.apple.com'
  and not requests.host ~~ '%.apps.apple.com' and not requests.host ~~ '%.mzstatic.com'
  and not requests.host = 'itunes.apple.com' and not requests.host = 'setup.icloud.com'
  and not requests.host = 'pancake.apple.com' and not requests.host = 'csig.apple.com'
  and not requests.host = 'gs-loc.apple.com' and not requests.host ~~ 'p%-%.icloud.com'
  and not requests.host = 'deviceenrollment.apple.com'
  and not requests.host = 'deviceservices-external.apple.com'
  and not requests.host = 'static.ips.apple.com'
  and not requests.host = 'mdmenrollment.apple.com'
  and not requests.host = 'vpp.itunes.apple.com'
  and not requests.host = 'updates-http.cdn-apple.com'
  and not requests.host = 'updates.cdn-apple.com'
  and not requests.host = 'serverstatus.apple.com'
  and not requests.host ~~ '%.appattest.apple.com'
  and not requests.host = 'cssubmissions.apple.com'
  and not requests.host = 'diagassets.apple.com'
  and not requests.host = 'configuration.apple.com'
  and not requests.host = 'configuration.ls.apple.com'
  and not requests.host ~~ 'gspe%-ssl.ls.apple.com'
  and not requests.host ~~ 'gsp%-ssl.ls.apple.com'
  and not requests.host = 'weather-data.apple.com'
  and not requests.host = 'token.safebrowsing.apple'
  and not requests.host = 'apple-finance.query.yahoo.com'
  and not requests.host = 'keyvalueservice.icloud.com'
  and not requests.host = 'gateway.icloud.com'
  and not requests.host = 'metrics.icloud.com'
  and not requests.host = 'calendars.icloud.com'
)

or

(
  platform = 'android'
  and not (requests.host = 'android.clients.google.com'
           and requests.path = '/c2dm/register3')
  and not
    (requests.host = 'android.googleapis.com' and requests.path = '/auth/devicekey')
  and not (requests.host ~~ '%.googleapis.com' and requests.path ~~ '/google.internal%')
  and not
    (requests.host ~~ 'www.googleapis.com' and requests.path ~~ '/androidantiabuse/%')
  and not (requests.host ~~ 'www.googleapis.com' and
          requests.path ~~ '/androidcheck/v1/attestations%')
  and not (requests.host ~~ 'play.googleapis.com' and requests.path = '/log/batch')
  and not (requests.host ~~ 'www.googleapis.com'
           and requests.path ~~ '/experimentsandconfigs/%')
  and not requests.host = '172.217.19.74'
  and not (requests.host ~~ 'firebaseinstallations.googleapis.com' and
          requests.path ~~ '/v1/projects/google.com%')
  and not (requests.host ~~ 'firebaseinstallations.googleapis.com' and
          requests.path ~~ '/v1/projects/metal-dimension-646%')
  and not (requests.host ~~ 'firebaseinstallations.googleapis.com' and
          requests.path ~~ '/v1/projects/zillatest-20296%')
  and not (requests.host ~~ '%gvt1.com' and requests.path ~~ '/edgedl/%')
  and not requests.host ~~ 'update.googleapis.com'
  and not (requests.host ~~ 'www.gstatic.com' and requests.path ~~ '/android%')
  and not (requests.host = 'www.google.com' and requests.path = '/loc/m/api')
  and not (requests.host = 'ssl.gstatic.com' and requests.path ~ '/suggest-dev/yt')
  and not (requests.host = 'android.googleapis.com' and requests.path = '/checkin')
  and not (requests.host = 'www.gstatic.com' and requests.path ~ '/commerce/wallet')
  and not (requests.host = 'app-measurement.com' and
          requests.path ~ '/config/app/1%3A357317899610%3Aandroid%3A4765c0ded882c665')
  and not (requests.host = 'app-measurement.com' and
          requests.path ~ '/config/app/1%3A1086610230652%3Aandroid%3A131e4c3db28fca84')
  and not (requests.host = 'ssl.google-analytics.com'
           and requests.content ~ 'UA-61414137-1')
  and not (requests.host = 'www.googletagmanager.com'
           and requests.content ~ 'GTM-K9CNX3')
  and not requests.host = 'accounts.google.com'
  and not requests.host = 'safebrowsing.googleapis.com'
  and not requests.path ~ '/v1/projects/chime-sdk/installations'
)

and not (requests.host = 'app-measurement.com'
         and not encode(requests.content_raw, 'escape')
           like concat('%', apps.name, '%'));
```

A.3. Regexes Used for Detecting Consent Elements

We use the following regexes and strings to find clear “accept” buttons:

```
/(accept|agree|allow|consent|permit) and continue/, 'accept', 'agree', 'allow',
'consent', 'permit', /(select|choose) all/, 'yes',

/(akzeptieren?|zustimmen|zulassen|annehmen|erlauben?|einwilligen|genehmigen?) und
  weiter/, /akzeptieren?/, 'zustimmen', 'zulassen', 'annehmen', /erlauben?/,
'einwilligen', /genehmigen?/, /alle (aus)?wählen/, /stimm[^.]{0,4} zu/,
/nehm[^.]{0,4} an/, /willig[^.]{0,4} ein/, 'ja'
```

We use the following strings to find ambiguous “accept” buttons:

```
'ok', 'okay', 'got it', 'confirm', 'next', 'continue',
'yes, continue to see relevant ads',

'weiter', 'fortfahren', 'bestätigen'
```

We use the following regexes and strings to find clear “reject” buttons:

```
'disagree', 'decline', 'reject', 'refuse', 'deny', /opt(- )?out/, 'no',
/(do not|don't|without) (accept|agree|allow|consent|permit)(ing)?/,
/no,? thanks?( you)?/, 'i want to opt out',

/widersprechen?/, 'ablehnen', /verweiger(n|e)/, 'nein',
/nicht (akzeptieren?|zustimmen|zulassen|annehmen|erlauben?|einwilligen|genehmigen?)/,
/ohne (akzept|zustimm|zulass|annehm|erlaub|einwillig|genehmig)/, /lehn[^.]{0,4} ab/,
/nein,? danke/
```

We use the following regexes and strings to find ambiguous “reject” buttons:

```
/customi(z|s)e/, /personali(z|s)e/, /more (choices|details|info|information)/,
'settings', 'options', 'preferences', 'configure',
/(adjust|change|manage|view|show|more)[^.]{0,12}
    (details|settings|options|preferences|cookies|choices)/,
/(confirm|save)[^.]{0,8} selection/, 'later', 'skip', 'exit', 'cancel',
/(learn|read) more/, 'not now', 'no, see ads that are less relevant',

'anpassen', 'personalisieren', 'einstellungen', 'einstellen', 'konfigurieren',
'optionen', /(details|einstellungen|optionen|cookies|auswahl)[^.]{0,5}
    (anpassen|ändern|verwalten)/, /mehr (details|infos|information)/,
'details anzeigen', 'anpassen', /auswahl (bestätigen|speichern)/, 'später',
/schlie(ß|ss)en/, 'beenden', 'abbrechen', /mehr (erfahren|lesen)/, 'jetzt nicht',
'überspringen'
```

The following negator regexes and strings prevent an element from being classified as an affirmative button:

```
'disagree', 'decline', 'reject', 'refuse', 'deny', /opt(- )?out/, 'no', "don't", 'not',

/widersprechen?/, 'ablehnen', /verweiger(n|e)/, 'nein', 'nicht', 'kein'
```

We use the following regexes to detect dialogs and notices:

```
/(we care about|comitted|respect)[^.]{0,10} (privacy|data protection)/,
/(privacy|data protection) [^.]{0,35} important/,
/can( always| later)? revoke[^.]{0,15} consent ?(at any time|later)?/,
/(use|utilise|need|have|set|collect|ask)[^.]{0,25} (cookie|consent|tracking)/,
/by (sign|logg|continu|creat|us|tapp|click|select|choos)ing [^.]{0,75},?
    (you|I) (agree|accept|consent|acknowledge|confirm)/,
/(accept|agree|consent) [^.]{3,35} (privacy|cookie|data protection|GDPR)
    (policy|notice|information|statement)/,
/(accept|agree|consent) [^.]{3,35} processing [^.]{3,20} data/,
/(learn|read|more|acknowledge) [^.]{2,40} (privacy|cookie|data protection|GDPR)
    (policy|notice|information|statement)/,
/have read( and understood)? [^.]{3,35} (privacy|cookie|data protection|GDPR)
    (policy|notice|information|statement)/,

/(Datenschutz|Privatsphäre) (ist uns wichtig|liegt uns am Herzen)/,
/respektier[^.]{0,20} (Datenschutz|Privatsphäre)/,
/wir nehmen[^.]{0,10} (Datenschutz|Privatsphäre) ernst/,
/(kannst|können)[^.]{0,10} Einwilligung jederzeit[^.]{0,20} widerrufen/,
/(benutz|verwend|nutz|brauch|benötig|hab|setz|sammel|frag)[^.]{0,25}
    (Cookie|Zustimmung|Einwilligung|Einverständnis|Tracking)/,
/(mit|durch|bei|wenn) [^.]{2,30} (tipp|klick|(aus)?wähl)[^.]{2,65}
    (akzeptier|stimm|nimm|nehm|bestätig)/,
/(akzeptier|stimm|nimm|nehm) [^.]{3,35}
    (Datenschutz|Cookie|DSGVO|Privatsphäre)-?(hinweis|erklärung|information)/,
/(Datenschutz|Cookie|DSGVO|Privatsphäre)-?(hinweis|erklärung|information) [^.]{3,35}
    (akzeptier|stimm|nimm|nehm)/,
/(akzeptier|stimm|nimm|nehm) [^.]{3,35}
    ((Verarbeit(ung|en) [^.]{3,20} Daten)|(Daten(-| )?[^.]{0,10}Verarbeit(ung|en)))/,
/((Verarbeit(ung|en) [^.]{3,20} Daten)|(Daten(-| )?[^.]{0,10}Verarbeit(ung|en)))
    [^.]{3,35} (akzeptier|stimm|nimm|nehm)/,
/(Informationen|mehr)( dazu)? [^.]{0,25}in [^.]{0,20}
    (Datenschutz|Cookie|DSGVO|Privatsphäre)-?(hinweis|erklärung|information)/,
/(Datenschutz|Cookie|DSGVO|Privatsphäre)-?(hinweis|erklärung|information)
    [^.]{3,35} (gelesen|Kenntnis)/,
/(Informationen|mehr)( dazu)? [^.]{0,30}in [^.]{0,25}
    (Datenschutz|Cookie|DSGVO|Privatsphäre)-?(hinweis|erklärung|information)/,
/(Datenschutz|Cookie|DSGVO|Privatsphäre)-?(hinweis|erklärung|information)
    (gelesen|Kenntnis)/,
/(mit|durch|bei|wenn) [^.]{2,30}
```

```
    (fortf(a|ä)hr|weitermach|anmeld|registrier|erstell|nutz|tipp|klick|(aus)?wähl)
        [^.]{2,65} (akzeptier|stimm|nimm|nehm|bestätig)/,
```

We use the following regexes to detect privacy policy links:

```
/(privacy|cookie|data protection|GDPR) (policy|notice|information|statement)/,

/(Datenschutz|Cookie|DSGVO|Privatsphäre)-?(hinweis|erklärung|information)(e|en)?/
```

The following keywords are worth one point:

```
/third-party ad(vertising|s)?/, /(read|store) cookies/,
/(ad(vertising|s)?|content|experience) personali(s|z)ation/,
/personali(s|z)ed?[^.]{0,10} (ad(vertising|s)?|content|experience)/,
/(ad(vertising|s)?|content) (measurement|performance)/, 'analytics',
'data processing purposes', 'audience insights', 'personal data',
/user (behaviou?r|data)/, 'GDPR', 'data protection regulation',
'insufficient level of data protection', 'mobile identifiers', /(advertising|ad)-?ID/,
/(necessary|essential|needed) cookies/, 'data processing', /(pseudo|ano)nymi(s|z)ed/,
/(data protection|privacy) (settings|controls|preferences)/, 'legitimate interest',
'crash data', /(collect|transmit) (information|data)/,

/Drittanbieter-?(Werbung|Anzeige|Werbeanzeige)/, /Cookies ((aus)?lesen|speichern)/,
/personalisierte (Werbung|Anzeige|Werbeanzeige|Inhalt|Erfahrung)/,
/(Werbungs?|Anzeigen|Werbeanzeigen|Inhalt(s|e)?|Erfahrungs)-?Personalisierung/,
/(Werbungs?|Werbe|Anzeigen|Werbeanzeigen|Inhalt(s|e)?|Erfahrungs)-?
    (Messung|Performance|Leistung|Zahlen)/, 'Analysetools',
/(Zwecke? der Verarbeitung|Verarbeitungszweck)/, 'Zielgruppenwissen', 'personenbezogen',
/Nutz(er|ungs)(verhalten|daten)/, /DS-?GVO/, /Datenschutz-?Grundverordnung/,
/gleiches? Datenschutzniveau/, /mobile (ID|Kennungs?)-?Nummer/,
/(notwendige|erforderliche) Cookies/,
/Datenverarbeitung|Verarbeitung (Deiner|Ihrer) Daten/, /(pseudonymisiert|anonymisiert)/,
'Datenschutzeinstellungen', /berechtigte(n|s)? Interesse/,
/Crash-?(Daten|Bericht|Information)/,
/(Daten|Informationen) (sammeln|übertragen|übermitteln)/
```

The following keywords are worth half a point:

```
/(optimal|better) user experience/, 'European Court of Justice',
/(without( any)?|effective) (legal|judicial) remedy/, 'geolocation data',
'third countries', 'IP address', 'app activity', 'consent', 'privacy',
'data protection', /\bprocess(ed|ing)?\b/,

/(bessert?e|optimale) Nutz(er|ungs)erfahrung/, 'EuGH', /Europäische(r|n)? Gerichtshof/,
/wirksame(r|n) Rechtsbehelf/, /(Standort|Geo)-?daten/, 'Drittländer', 'IP-Adresse',
'Aktivitätsdaten', 'Einwilligung', 'Datenschutz', 'Privatsphäre', /verarbeit(en|ung)/
```

A.4. Thesis Proposal

The following exposé was submitted as the proposal for this master's thesis. It served as the task definition.

Introduction and Motivation

Previous research has shown that automated data collection in the background is common in both Android and iOS apps. A large portion of apps transmits telemetry data about device details (model, settings, battery status, etc.), sensor data, events (which pages are opened and buttons are clicked), or even the geolocation and which data is entered in the app. Often, this data is sent to third-party companies and linked to the device's *Advertising ID* that allows those companies to track users across multiple apps.

These practices are troubling from a data protection standpoint, especially since the *General Data Protection Regulation* (GDPR) that mandates strict legal guidelines for such data collection went into force in 2018. According to the GDPR, any processing of personal data (meaning any data that can somehow be linked to a natural person, including pseudonymously) needs to have one of six possible legal bases (Art. 6(1) GDPR). According to the supervisory authorities, which are responsible for enforcing the GDPR in the EU, usually informed consent is the only applicable legal basis for tracking[1].
This is further amplified by the *ePrivacy Directive* (ePD), which mandates even stricter information, consent, and opt-out requirements for accessing information stored on a user's device and applies even when no personal data is processed (Art. 5(3) ePD), and the European Court of Justice's (CJEU) *Schrems II* ruling, which invalidated the *Privacy Shield* adequacy decision that data transfers to the US were usually based on.

In an effort to circumvent these legal problems, websites and mobile apps alike tend to use *consent dialogs*, which are supposed to inform the user about the processing that is taking place and to either get their consent or allow them to opt out. But these consent dialogs give rise to new legal and technical questions themselves. Research into cookie banners on the web showed that nudging and dark patterns meant to coerce the user into giving consent and discouraging them from opting out are common[2]. Many websites also start tracking before the user has made a choice in the consent dialog or continue tracking even after they have opted out[3].

And the legal requirements for consent are high. Consent needs to be given by a "clear affirmative act" (Recital 32 to the GDPR) and different purposes require individual consent (Art. 7(2) GDPR). The CJEU's *Planet49* ruling further confirmed that pre-ticked checkboxes in cookie banners do not constitute valid consent, even when no personal data is processed.

Conversely, consent dialogs are often also shown where they aren't actually needed. In cases where the processing of certain data is necessary for the website or app to function, both the GDPR and ePD give alternate legal bases that do not require consent from the user (Art. 6(1)(f) GDPR, Art. 5(3) ePD).

[1] `https://www.datenschutzkonferenz-online.de/media/ah/201804_ah_positionsbestimmung_tmg.pdf`

[2] `https://noyb.eu/en/noyb-aims-end-cookie-banner-terror-and-issues-more-500-gdpr-complaints`, `https://dl.acm.org/doi/abs/10.1145/3442381.3450056`, `https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9152617`

[3] `https://www.sciendo.com/article/10.2478/popets-2019-0023`

Related Work

Prior research into consent dialogs in the wild has so far been almost exclusively limited to the web. Utz et al. manually looked at screenshots of 1,000 cookie dialogs on websites and identified eight variables in which they differ in 2019[4]. In 2020, Matte et al. crawled 1,426 websites with cookie banners that implement the *Transparency & Consent Framework* by IAB Europe, an industry self-regulation organization, which allowed them to automatically extract data on suspected violations.

Hu et al. (2019) studied the effect the introduction of the GDPR had on cookie usage across the web[5] and Eijk et al. (2019) compared the impact of the user's location (in particular inside and outside the EU) on the cookie notices they were shown[6].

There are also various user studies that explore how different consent dialog designs and wordings influence the user's decisions, touching on dark patterns and nudging, for example[7].

More generally, for privacy violations on Android, two 2019 studies by Liu et al.[8] and He et al.[9] detected privacy leaks from third-party libraries based on static and dynamic analysis. Two 2016 studies by Slavin et al.[10] and Yu et al.[11] compared apps' privacy policies with the tracking code used in them based on text analysis.
On iOS, research into privacy violations by apps, is scarce and outdated. A 2011 paper by Egele et al. explored using static analysis on app binaries to detect privacy leaks[12] and a 2015 paper by Dehling et al. crawled app store pages of health apps[13].

Goals

Given the lack of research into consent dialogs on mobile devices, this thesis will study consent dialogs on Android and iOS in an automated and dynamic manner, analysing a large number of apps from both platforms.

In the first step, different consent dialog types and frameworks (*Consent Management Platforms*, CMPs), as well as their settings will be identified in the apps. For the identified dialogs, the listed options (consent for which processing is requested?, which third-party tracking companies are used?) will be extracted. These findings will then be compared to the existing research for the web.

In the second step, the collected data will be used to identify violations by the apps. For this, a list of criteria for a legally compliant consent dialog will be compiled and the discovered consent dialogs will be checked against that. This will include nudging and dark patterns, like pre-ticked checkboxes, overly prioritizing the "Accept all" button, only giving the user the option to agree to the entire processing without the ability to

[4]https://www.researchgate.net/profile/Martin-Degeling/publication/334965379_Uninformed_Consent_Studying_GDPR_Consent_Notices_in_the_Field/links/5d638e6c458515d610253bb1/Uninformed-Consent-Studying-GDPR-Consent-Notices-in-the-Field.pdf
[5]https://dl.acm.org/doi/abs/10.1145/3292522.3326039
[6]https://pure.tudelft.nl/ws/files/57080768/vaneijk_conpro19.pdf
[7]https://arxiv.org/pdf/1908.10048.pdf, https://dl.acm.org/doi/abs/10.1145/1753326.1753689
[8]https://ieeexplore.ieee.org/abstract/document/8660581
[9]https://www.sciencedirect.com/science/article/abs/pii/S2214212618304356
[10]https://dl.acm.org/doi/abs/10.1145/2884781.2884855
[11]https://ieeexplore.ieee.org/abstract/document/7579770
[12]http://www.syssec-project.eu/m/page-media/3/egele-ndss11.pdf
[13]https://mhealth.jmir.org/2015/1/e8/PDF

opt out of individual purposes, etc.
Additionally, the effect of the user's choice in the consent dialog will also be measured. For this, first, any tracking before consent is given will be identified. Then, different approaches for interacting with consent dialogs will be evaluated. The best scaling one will be used to conduct a post-consent study, checking the difference in tracking between the user giving or denying consent, as well as any more granular options the consent dialogs may offer.
Where conclusive violations against the law are found, the app's publisher will be notified of the findings and given the ability to rectify them. Where this doesn't happen after some period of time, complaints will be filed with the supervisory authorities.

The results from the analysis will also be compared to the *privacy labels* on iOS, which require apps to self-label the affected categories of data being processed and the purposes for the processing in the App Store.

The thesis will build on existing research into tracking on Android and iOS without user interaction that the author has participated in. The existing system will be extended to allow for extracting information displayed by the apps and interacting with them. In addition, the system for identifying tracking, which currently only looks at the domains, will be improved to also factor in the actual data and any identifiers being transmitted.

Research Tasks

- Compile list of criteria for compliant consent dialogs, based on applicable legislation, court rulings, supervisory authority recommendations, and previous work.
- Extend existing analysis framework to allow extraction of elements on screen and interaction with apps.
- Extend existing analysis framework for better identification of tracking, based on looking at actual transmitted data.
- Identify consent dialogs as well as their frameworks and settings in apps. Compare results with research for the web.
- Identify violations in consent dialogs (dark patterns, nudging).
- Interact with consent dialogs and measure effect of the choices on the tracking going on.
- Evaluate results and compare with privacy labels on iOS.