Automated Analysis of Privacy Requirements for Mobile Apps

Sebastian Zimmeck[1]*, Ziqi Wang[2], Lieyong Zou[2], Roger Iyengar[3]*, Bin Liu[2], Florian Schaub[4]*, Shomir Wilson[5]*, Norman Sadeh[2], Steven M. Bellovin[1], and Joel Reidenberg[6]

[1]Department of Computer Science, Columbia University, {sebastian,smb}@cs.columbia.edu

[2]School of Computer Science, Carnegie Mellon University, {ziqiw,lieyongz,bliu1}@andrew.cmu.edu, sadeh@cs.cmu.edu

[3]Department of Computer Science and Engineering, Washington University in St. Louis, rogeriyengar@wustl.edu

[4]School of Information, University of Michigan, fschaub@umich.edu

[5]EECS Department, University of Cincinnati, shomir.wilson@uc.edu

[6]School of Law, Fordham University, reidenberg@law.fordham.edu

Abstract

Mobile apps have to satisfy various privacy requirements. App publishers are often obligated to provide a privacy policy and notify users of their apps' privacy practices. But how can we tell whether an app behaves as its policy promises? In this study we introduce a scalable system to help analyze and predict Android apps' compliance with privacy requirements. Our system is not only intended for regulators and privacy activists, but also meant to assist app publishers and app store owners in their internal assessments of privacy requirement compliance.

Our analysis of 17,991 free apps shows the viability of combining machine learning-based privacy policy analysis with static code analysis of apps. Results suggest that 71% of apps that lack a privacy policy should have one. Also, for 9,050 apps that have a policy, we find many instances of potential inconsistencies between what the app policy seems to state and what the code of the app appears to do. Our results suggest that as many as 41% of these apps could be collecting location information and 17% could be sharing such with third parties without disclosing so in their policies. Overall, it appears that each app exhibits a mean of 1.83 inconsistencies.

1 Introduction

Snapchat does "not ask for, track, or access any location-specific information." This is what Snapchat's privacy policy used to state.[1] However, in reality Snapchat's Android app transmitted Wi-Fi and cell-based location data from users' devices to analytics service providers. These discrepancies remained undetected until a researcher examined Snapchat's data deletion mechanism. His report was picked up by the Electronic Privacy Information Center and brought to the attention of the Federal Trade Commission (FTC), which launched a formal investigation requiring Snapchat to implement a comprehensive privacy program.[2]

The case of Snapchat illustrates that mobile apps are often non-compliant with privacy requirements. However, any inconsistencies can have dire consequences as they may lead to enforcement actions by the FTC and other regulators. This is especially true if discrepancies continue to exist for many years, which was the case for Yelp's collection of childrens' information.[3] These findings not only demonstrate that regulators could benefit from a system that helps them identifying potential inconsistencies, but also that it would be a useful tool for companies in the software development process. After all, researchers found that privacy violations often appear to be based on developers' difficulties in understanding privacy requirements (Balebako et al. 2014) rather than on malicious intentions.

On various occasions, the FTC, which is responsible for regulating consumer privacy on the federal level, voiced dissatisfaction with the current state of apps' privacy compliance. Three times it manually surveyed childrens' apps (FTC 2012a; 2012b; 2015) and concluded that the "results of the survey are disappointing" (FTC 2012b). Deviating from mandatory provisions, many publishers did not disclose what types of data they collect, how they make use of the data, and with whom the data is shared (FTC 2012b). A similar examination of 121 shopping apps revealed that many privacy policies are vague and fail to convey how apps actually handle consumers' data (FTC 2014b). Given that the FTC limited its investigations to small samples of apps, a presumably large number of discrepancies between apps and their privacy policies remain undetected.

In this study we are presenting a system that checks data practices of Android apps against privacy requirements derived from their privacy policies and selected laws. Our work enables app publishers to identify potential inconsistencies before they become prevalent. Moreover, our work can also aid governmental agencies, such as the FTC, to achieve a systematic enforcement of privacy laws on a large scale. The techniques presented in this paper have been customized and packaged in the form of a tool piloted by the Office of the California Attorney General over the summer of 2016. App store owners, researchers, and privacy advocates alike might also derive value from our study. Our main contribution consists of the novel combination of machine learning (ML) and static analysis techniques to analyze apps' potential non-

*Work mostly done while at Carnegie Mellon University.

[1]In the Matter of Snapchat, Inc., FTC No. 132 3078 (December 31, 2014, Complaint).

[2]In the Matter of Snapchat, Inc., FTC No. 132 3078 (December 31, 2014, Decision and Order).

[3]United States of America v. Yelp, Inc., FTC No. 132 3066 (September 17, 2014, Complaint for Permanent Injunction, Civil Penalties, and other Relief).

compliance with privacy requirements. However, we want to emphasize that we do not claim to resolve challenges in the individual techniques we leverage beyond what is necessary for our purposes. This holds especially true for the static analysis of mobile apps and its many unresolved problems, for example, in the analysis of obfuscated code. That said, the details of our contribution are as follows:

1. For a set of 17,991 Android apps we check whether they have a privacy policy. For the 9,295 apps that have one we apply machine learning classifiers to analyze policy content based on a human-annotated corpus of 115 policies. We show, for instance, that only 46% of the analyzed policies describe a notification process for policy changes. (§ 3).
2. Leveraging static analysis we investigate the actual data practices occurring in the apps' code. With a failure rate of 0.4%, a mean F-1 score of 0.96, and a mean analysis time of 6.2 seconds per app our approach makes large-scale app analyses for legally relevant data practices feasible and reliable. (§ 4).
3. Mapping the policy to the app analysis results we identify and analyze potential inconsistencies between policies and apps. With each app exhibiting a mean of 1.83 potential inconsistencies we find their occurrence to be a widespread phenomenon. (§ 5).

2 Related Work

Privacy policies disclose an organization's data practices. Despite efforts to make them machine-readable, e.g., via P3P (Cranor et al. 2002), or formalize them, for instance, using EPAL (Ashley et al. 2003), natural language policies are the de-facto standard for notifying web users of data practices. However, those policies are often long and difficult to read. Few lay users ever read them and regulators lack the resources to systematically review them. For instance, it took 26 data protection agencies one week, working together as the Global Privacy Enforcement Network (GPEN), to analyze the policies of 1,211 apps (GPEN 2015). While various works aim to make privacy policies more comprehensible, e.g., Ghazinour et al. (2009) provide a model for visualizing different practices, there is a glaring absence of an automated system to accurately analyze policy content.

It is our goal to automatically analyze natural language privacy policies at scale (Sadeh et al. 2013). Analyzing such policies presents a substantial challenge (Wilson et al. 2016a). As of now, Massey et al. (2013) provided the most extensive evaluation of 2,061 policies, however, not focusing on their legal analysis but rather their readability and suitability for identifying privacy protections and vulnerabilities from a requirements engineering perspective. In addition, Hoke et al. (2015) studied the compliance of 75 policies with self-regulatory requirements, and Cranor et al. (2013) analyzed structured privacy notice forms of financial institutions identifying multiple instances of opt out practices that appear to be in violation of financial industry laws.

Different from previous studies we analyze policies at scale with a legal perspective and not limited to the financial industry. We analyze whether policies are available as sometimes required by various laws and examine their descriptions of data collection and sharing practices. For our analysis we rely on the flexibility of ML classifiers (Zimmeck and Bellovin 2014) and introduce a new approach for privacy policy feature selection. Our work is informed by the study of Costante et al., who presented a completeness classifier to determine which data practice categories are included in a privacy policy (2012) and proposed rule-based techniques to extract data collection practices (2013). However, we go beyond these works in terms of both breadth and depth. We analyze a much larger policy corpus, and we focus on legal questions that have not yet been automatically analyzed. Different from many existing works that focus on pre-processing of policies, e.g. by using topic modeling (Chundi and Subramaniam 2014; Stamey and Rossi 2009) and sequence alignment (Liu et al. 2014; Ramanath et al. 2014) to identify similar policy sections and paragraphs, we are interested in analyzing policy content.

Supervised ML techniques, as used in this study, require ground-truth. To support the development of these techniques crowdsourcing has been proposed as a viable approach for gathering rich annotations from unstructured privacy policies (Sadeh et al. 2013; Wilson et al. 2016c). While crowdsourcing poses challenges due to the policies' complexity (Reidenberg et al. 2015), assigning annotation tasks to experts (Zimmeck and Bellovin 2014) and setting stringent agreement thresholds and evaluation criteria (Wilson et al. 2016c) can in fact lead to reliable policy annotations. However, as it is a recurring problem that privacy policy annotations grapple with low inter-annotator agreement (Reidenberg et al. 2015; Zimmeck and Bellovin 2014), we introduce a measure for analyzing their reliability based on the notion that high annotator disagreement does not principally inhibit the use of the annotations for ML purposes as long as the disagreement is not systematic.

The static analysis approach used in our system is inspired by TaintDroid (Enck et al. 2010) and the studies of Slavin et al. (2016) and Yu et al. (2016). However, we move beyond these contributions. First, our privacy requirements cover privacy questions previously not examined. Notably, we address whether an app needs a policy and analyze the policy itself (i.e., whether it describes how users are informed of policy changes and how they can access, edit, and delete data). Different from Slavin et al. we also analyze the collection and sharing of contact information. Second, TaintDroid, is not intended to have app store wide scale. Third, previous approaches do not neatly match to legal categories. They do not distinguish between first and third party practices (Enck et al. 2010; Slavin et al. 2016), do not take into account negative policy statements (i.e., statements that an app does *not* collect certain data, as, for example, in the Snapchat policy quoted in § 1) (Slavin et al. 2016), and base their analysis on a dichotomy of strong and weak violations (Slavin et al. 2016) unknown to the law. Fourth, we introduce techniques that achieve a mean accuracy of 0.94 and a failure rate of 0.4%, which improve over the closest comparable results of 0.8 and 21% (Slavin et al. 2016), respectively.

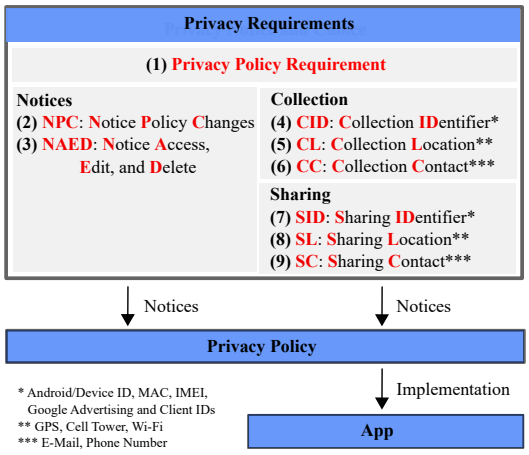

Figure 1: Per our privacy requirements, apps that process Personally Identifiable Information (PII) need to (1) have a privacy policy, (2-3) include notices about policy changes and access, edit, and deletion rights in their policy, (4-6) notify users of data collection practices, and (7-9) disclose how data is shared with third parties. The notice requirements for policy changes and access, edit, and deletion are satisfied by including the notices in the policies while the collection and sharing practices must be also implemented in the apps.

3 Privacy Policy Analysis

In this section we present our automated large-scale ML analysis of privacy policies. We discuss the law on privacy notice and choice (§ 3.1), how many apps have a privacy policy (§ 3.2), and the analysis of policy content (§ 3.3).

3.1 Privacy Notice and Choice

The privacy requirements analyzed here are derived from selected laws and apps' privacy policies. Figure 1 provides an overview of the law on notice and choice and the nine privacy requirements that our system analyzes (Privacy Policy Requirement, NPC, NAED, CID, CL, CC, SID, SL, SC). If a policy or app does not appear to adhere to a privacy requirement, we define a potential privacy requirement inconsistency to occur (which we also refer to as potential inconsistency or non-compliance). In this regard, we caution that a potential inconsistency does not necessarily mean that a law is violated. First, not all privacy requirements might be applicable to all apps and policies. Second, our system is based on a particular interpretation of the law. While we believe that our interpretation is sound and in line with the enforcement actions of the FTC and other regulatory agencies, reasonable minds may differ.[4] Third, our system is based on machine learning and static analysis and, thus, by its very nature errors can occur.

As to the privacy policy requirement, there is no generally applicable federal statute demanding privacy policies for apps (Zimmeck 2013). However, California and Delaware enacted comprehensive online privacy legislation that effectively serves as a national minimum privacy threshold given that app publishers usually do not provide state-specific app versions or exclude California or Delaware residents. In this regard, the California Online Privacy Protection Act of 2003 (CalOPPA) requires online services that collect PII to post a policy.[5] The same is true according to Delaware's Online Privacy and Protection Act (DOPPA).[6] In addition, the FTC's Fair Information Practice Principles (FTC FIPPs) call for consumers to be given notice of an entity's information practices before any PII is collected (FTC 1998). Further, the Children's Online Privacy Protection Act of 1998 (COPPA) makes policies mandatory for apps directed to or known to be used by children.[7] Thus, we treat the existence of a privacy policy as a privacy requirement.

CalOPPA and DOPPA further demand that privacy policies describe the process by which users are notified of policy changes.[8] COPPA also requires description of access, edit, and deletion rights.[9] Under the FTC FIPPs (FTC 1998) as well as CalOPPA and DOPPA those rights are optional.[10] We concentrate our analysis on a subset of data types that are, depending on the context, legally protected: device IDs, location data, and contact information. App publishers are required to disclose the collection of device IDs (even when hashed) and location data.[11] Device IDs and location data are also covered by CalOPPA[12] and for childrens' apps according to COPPA.[13] The sharing of these types of information with third parties requires consent as well.[14] Similarly, contact information, such as e-mail addresses, may be protected, too.[15]

It should be noted that we interpret ad identifiers to be PII since they can be used to track users over time and across devices. We are also assuming that a user did not opt out of ads (because otherwise no ad identifiers would be sent to opted out ad networks). We further interpret location data to refer to GPS, cell tower, or Wi-Fi location. We assume applicability of the discussed laws and perform our analysis based on the guidance provided by the FTC (1998) and the California Department of Justice (2014) in enforcement actions and recommendations for best practices. Specifically, we interpret the FTC actions as disallowing the omission of

[4] We are focusing on the US legal system as we are most familiar with it. However, in principle, our techniques are applicable to any country with a privacy notice and choice regime.

[5] Cal. Bus. & Prof. Code §22575(a).

[6] Del. Code Tit. 6 §1205C(a).

[7] 16 CFR §312.4(d).

[8] Cal. Bus. & Prof. Code §22575(b)(3), Del. Code Tit. 6 §1205C(b)(3).

[9] 16 CFR §312.4(d)(3).

[10] Cal. Bus. & Prof. Code §22575(b)(2), Del. Code Tit. 6 §1205C(a).

[11] In the Matter of Nomi Technologies, Inc., FTC No. 132 3251 (September 3, 2015, Complaint).

[12] Per the interpretation of Cal. Bus. & Prof. Code §22577(a)(6) and (7) by the California Department of Justice (2014).

[13] 16 CFR §312.2(7) and (9).

[14] In the Matter of Goldenshores Technologies, LLC, and Erik M. Geidl, FTC No. 132 3087 (April 9, 2014, Complaint).

[15] In the Matter of Snapchat, Inc., FTC No. 132 3078 (December 31, 2014, Complaint).

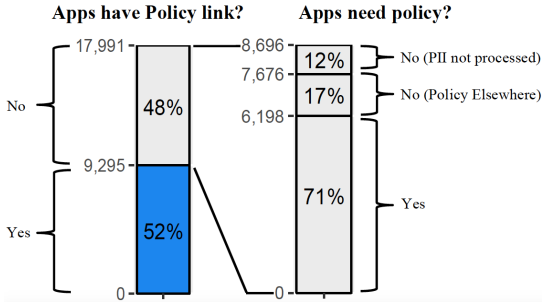

Figure 2: We analyze 17,991 free apps, of which 9,295 (52%) link to their privacy policy from the Play store (left). Out of the remaining apps, 6,198 (71%) appear to lack a policy while engaging in at least one data practice (i.e., PII is processed) that would require them to have one (right).

data practices in policies and assume that silence on a practice means that it does not occur.[16] Also, we assume that all apps in the Play store are subject to CalOPPA and DOPPA,[17] which we believe to be reasonable as we are not aware of any app in the Play store excluding California or Delaware residents or of state-specific app versions for those states.

3.2 Privacy Policy Requirement

To assess whether apps fulfill the requirement of having a privacy policy we crawled the Google Play store (February 2016) and downloaded a sample ($n = 17,991$) of free apps (full app set).[18] We started our crawl with the most popular apps and followed random links on their Play store pages to other apps. We included all categories in our crawl, however, excluded Google's Designed for Families program (as Google already requires apps in this program to have a policy) and Android Wear (as we want to focus on mobile apps). We assume that our sample is representative in terms of app categories, which we confirmed with a two-sample Kolmogorov-Smirnov goodness of fit test (two-tailed) against a sample of a million apps (Olmstead and Atkinson 2015). We could not reject the null hypothesis that both were drawn from the same distribution (i.e., $p > 0.05$). However, while the Play store hosts a long tail of apps that have fewer than 1K installs (56%) (Olmstead and Atkinson 2015), we focus on more popular apps as our sample includes only 3% of such apps.

Out of all policies in the full app set we found that $n = 9,295$ apps provided a link to their policy from the Play store (full policy set) and $n = 8,696$ apps lacked such. As shown in Figure 2, our results suggest that 71% (6,198/8,696) apps without a policy link are indeed not adhering to the policy requirement. We used the Play store privacy policy links as proxies for actual policies, which we find reasonable since regulators requested app publishers to post such links (FTC 2013; California Department of Justice 2014) and app store owners obligated themselves to provide the necessary functionality (California Department of Justice 2012). The apps in the full app set were offered by 10,989 publishers, and their app store pages linked to 6,479 unique policies.

We arrive at 71% after making two adjustments. First, if an app does not have a policy it is not necessarily non-compliant with the policy requirement. After all, apps that are not processing PII are not obligated to have a policy. Indeed, since we found that 12% (1,020/8,696) of apps without a policy link are not processing PII, we accounted for those apps. Second, despite the regulators' requests to post policy links in the Play store, some app publishers may still decide to post their policy elsewhere (e.g., inside their app). To account for that possibility we randomly selected 40 apps from our full app set that did not have a policy link in the Play store but processed PII. We found that 83% (33/40) do not seem to have a policy posted anywhere (with a Clopper-Pearson confidence interval (CI) ranging from 67% to 93% at the 95% level based on a two-tailed binomial test).[19] Thus, accounting for an additional 17% (1,478/8,696) of apps having a policy elsewhere leaves us with $100\% - 12\% - 17\% = 71\%$ out of $n = 8,696$ apps to be potentially non-compliant with the policy requirement.

Practice	$\lvert Ann \rvert$	Ag_{pol}	$\%\, Ag_{pol}$	$Fleiss_{pol}$/$Krip_{pol}$
NPC	395	86/115	75%	0.64
NAED	414	80/115	70%	0.59
CID	449	**92/115**	80%	**0.72**
CL	326	85/115	74%	0.64
CC	830	86/115	75%	0.5
SID	90	101/115	88%	**0.76**
SL	51	95/115	83%	0.48
SC	276	85/115	74%	0.58

Table 1: The table shows absolute numbers of annotations ($|$Ann$|$) as well as various agreement measures, specifically, absolute agreements (Ag_{pol}), percentage agreements (% Ag_{pol}), Fleiss' κ ($Fleiss_{pol}$), and Krippendorff's α ($Krip_{pol}$). All agreement measures are computed on the full corpus of 115 policies and on a per-policy basis (e.g., for 92 out of 115 policies the annotators agreed on whether the policy allows collection of identifiers).

3.3 Privacy Policy Content

We now move from examining whether an app has a policy to the analysis of policy content (i.e., privacy requirements (2-9) in Figure 1). As a basis for our evaluation we use manually created policy annotations.

Inter-annotator Agreement For training and testing our policy classifiers we leverage the OPP-115 corpus (Wilson et al. 2016b)—a corpus of 115 privacy policies annotated by ten law students that includes 2,831 annotations for the practices discussed in this study. The annotations, which are described in detail by Wilson et al. (2016b), serve as the

[16] In the Matter of Snapchat, Inc., FTC No. 132 3078 (December 31, 2014, Complaint).

[17] Cal. Bus. & Prof. Code §§22575–22579, Del. Code Tit. 6 §1205C.

[18] Whenever we refer to Google Play we mean its US store.

[19] All CIs in this paper are based on a two-tailed binomial test and the Clopper-Pearson interval at the 95% level.

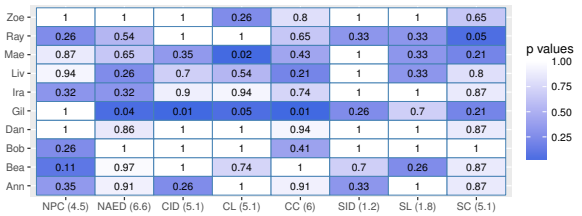

Figure 3: Analysis of disagreement among annotators for the different data practices with binomial tests. Larger p values mean fewer disagreements. If there are no disagreements, we define $p = 1$. The numbers in parentheses are the average absolute disagreements for the respective practices.

ground-truth for our classifiers. Each annotator annotated a mean of 34.5 policies (median 35). We select annotations according to majority agreement (i.e., two out of three annotators agreed on it). As it is irrelevant from a legal perspective how often a practice is described in a policy, we measure whether annotators agree that a policy describes a given practice at least once.

High inter-annotator agreement signals the reliability of the ground-truth on which classifiers can be trained and tested. As agreement measures we use Fleiss' κ and Krippendorff's α, which indicate that agreement is good above 0.8, fair between 0.67 and 0.8, and doubtful below 0.67 (Manning, Raghavan, and Schütze 2008). From our results in Table 1 it follows that the inter-annotator agreement for collection and sharing of device IDs with respective values of 0.72 and 0.76 is fair. However, it is below 0.67 for the remaining classes. While we would have hoped for stronger agreement, the annotations with the observed agreement levels can still provide reliable ground-truth as long as the classifiers are not misled by patterns of systematic disagreement, which can be explored by analyzing the disagreeing annotations (Reidsma and Carletta 2008).

To analyze whether disagreements contain systematic patterns we evaluate the number of each annotator's disagreements with the other two annotators. If he or she is in a minority position for a statistically significant number of times, there might be a misunderstanding of the annotation task or other systematic reason for disagreement. However, if there is no systematic disagreement, annotations are reliable despite low agreement levels (Reidsma and Carletta 2008). Assuming a uniform distribution each annotator should be in the minority in 1/3 of all disagreements. We test this assumption with the binomial test for goodness of fit. Specifically, we use the binomial distribution to calculate the probability of an annotator being x or more times in the minority by adding up the probability of being exactly x times in the minority, being $x+1$ times in the minority, up to $x+n$ (that is, being always in the minority), and comparing the sum to the expected probability of 1/3. We use a one-tailed test as we are not interested in finding whether an annotator is fewer times in the minority than in 1/3 of the disagreements.

We only found few cases with systematic disagreement. More specifically, for 7% (11/160) of an annotator's disagreements we found statistical significance ($p \leq 0.05$) for rejecting the null hypothesis that the disagreements are equally distributed. An annotator can be in the minority when omitting an annotation that the two other annotators made or adding an extra annotation. Figure 3 shows the former. However, excluding affected annotations from the training set for our classifiers had only little noticeable effect. Thus, we believe that our annotations are sufficiently reliable to serve as ground-truth for our classifiers. As other works have already explored, low levels of agreement in policy annotations are common and do not necessarily reflect their unreliability (Reidenberg et al. 2015; Zimmeck and Bellovin 2014). In fact, different from our approach here, it could be argued that an annotator's addition or omission of an annotation is not a disagreement with the others' annotations to begin with.

```
def location_feature_extraction(policy):

 data_type_keywords = ['geo', 'gps']
 action_keywords = ['share', 'partner']
 relevant_sentences = ''
 feature_vector = ''

 for sentence in policy:
  for keyword in data_type_keywords:
   if (keyword in sentence):
    relevant_sentences += sentence

 words = tokenize(relevant_sentences)
 bigrams = ngrams(words,2)

 for bigram in bigrams:
  for keyword in action_keywords:
   if (keyword in bigram):
    feature_vector += bigram, bigram[0],
        bigram[1]

 return feature_vector
```

Listing 1: Pseudocode for the location sharing practice.

Feature Selection One of the most important tasks for correctly classifying data practices described in privacy policies is appropriate feature selection. Using information gain and tf-idf we identified the most meaningful keywords for each practice and created sets of keywords. One set of keywords refers to the data type of the practices (e.g., keywords for the SL practice are "geo" and "gps") and is used to extract all sentences from a policy that contain at least one of the keywords. On these extracted sentences we are using a second set of keywords that refers to the actions of a data practice (e.g., for the SL practice "share" and "partner") to create unigram and bigram feature vectors (Zimmeck and Bellovin 2014). Listing 1 shows a simplified use of our algorithm for the SL practice. Thus, for example, if the keyword "share" is encountered, the bigrams "not share" or "will share" would be extracted if the words before the keyword are "not" and "will," respectively. The feature vectors created from bigrams (and unigrams) are then used to classify the practices. If no keywords are extracted, the clas-

Practice	*Classifier*	*Parameters*	***Base*** *n=40*	***Acc***$_{pol}$ *n=40*	***95% CI*** *n=40*	***Prec***$_{neg}$ *n=40*	***Rec***$_{neg}$ *n=40*	***F-1***$_{neg}$ *n=40*	***F-1***$_{pos}$ *n=40*	***Pos*** *n=9,050*
NPC	SVM	RBF kernel, weight	0.7	0.9	0.76–0.97	0.79	0.92	**0.85**	0.93	**46%**
NAED	SVM	linear kernel	0.58	0.75	0.59–0.87	0.71	0.71	**0.71**	0.78	**36%**
CID	Log. Reg.	LIBLINEAR solver	0.65	0.83	0.67–0.93	0.77	0.71	**0.74**	0.87	46%
CL	SVM	linear kernel	0.53	0.88	0.73–0.96	0.83	0.95	**0.89**	0.86	34%
CC	Log. Reg.	LIBLINEAR, L2, weight	0.8	0.88	0.73–0.96	0.71	0.63	**0.67**	0.92	56%
SID	Log. Reg.	LBFGS solver, L2	0.88	0.88	0.73–0.96	0.94	0.91	**0.93**	**0.55**	**10%**
SL	SVM	linear kernel, weight	0.95	0.93	0.8–0.98	0.97	0.95	**0.96**	-	**12%**
SC	SVM	poly kernel (4 degrees)	0.73	0.78	0.62–0.89	0.79	0.93	**0.86**	**0.47**	**6%**

Table 2: Classifiers, parameters, and classification results for the policy test set (n=40) and the occurrence of positive classifications (Pos) in a set of n=9,050 policies (full app/policy set). We obtained the best results by always setting the regularization constant to $C = 1$ and for NPC, CC, and SL adjusting weights inversely proportional to class frequencies with scikit-learn's `class_weight` (weight). Except for the SL practice, all classifiers' accuracies (Acc$_{pol}$) reached or exceeded the baseline (Base) of always selecting the most often occurring class in the training set. $Prec_{neg}$, Rec_{neg}, and *F-1*$_{neg}$ are the scores for the negative classes (e.g., data is *not* collected or shared) while *F-1*$_{pos}$ is the F-1 score for positive classes.

sifier will select the negative class. We applied the Porter stemmer to all processed text.

For finding the most meaningful features as well as for the subsequent classifier tuning we performed nested cross-validation with 75 policies separated into ten folds in the inner loop and 40 randomly selected policies as held out test set (policy test set). We used the inner cross-validation to select the optimal parameters during the classifier tuning phase and the held out policy test set for the final measure of classification performance. We stratified the inner cross-validation to avoid misclassifications due to skewed classes. After evaluating the performance of our classifiers with the policy test set we added the test data to the training data for the final classifiers to be used in our large-scale analysis.

Classification During the tuning phase we prototyped various classifiers with scikit-learn (Pedregosa et al. 2011), a Python library. Support vector machines and logistic regression had the best performance. We selected classification parameters individually for each data practice. The classification results for our policy test set, shown in Table 2, suggest that the ML analysis of privacy policies is generally feasible. For the negative classifications our classifiers achieve *F-1*$_{neg}$ scores between 0.67 and 0.96. These scores are the most important measures for our task because the identification of a potential inconsistency demands that a practice occurring in an app is *not* covered by its policy. Consequently, it is less problematic that the sharing practices, which are skewed towards the negative classes, have relatively low *F-1*$_{pos}$ scores of 0.55 (SID) and 0.47 (SC) or could not be calculated (SL) due to a lack of true positives in the policy test set.

We applied our classifiers to the policies in the full app/policy set with $n = 9,050$ policies. We obtained this set by adjusting our full policy set ($n = 9,295$) to account for the fact that not every policy link might actually lead to a policy: for 40 randomly selected apps from our full policy set we checked whether the policy link in fact lead to a policy, which was the case for 97.5% (39/40) of links (with a CI of 0.87 to 1 at the 95% level). As the other 2.5%, that is, one link, lead to some other page and would not contain any data practice descriptions, we randomly excluded from the full policy set 2.5% = 245 of policies without any data practice descriptions leaving us with $n = 9,295 - 245 = 9,050$ policies in the full app/policy set. We emphasize that this technique does not allow us to determine whether the 245 documents actually did not contain a policy or had a policy that did not describe any practices. However, in any case the adjustment increases the occurrence of positive data practice instances in the full app/policy set and keeps discrepancies between apps and policies at a conservative level as some apps for which the analysis did not find any data practice descriptions are now excluded.[20]

It appears that many privacy policies fail to satisfy privacy requirements. Most notably, per Table 2, only 46% describe the notification process for policy changes, a mandatory requirement for apps that do not exclude California and Delaware residents. Similarly, only 36% of policies contain a statement on user access, edit, and deletion rights, which COPPA requires for childrens' apps, that is, apps intended for children or known to be used by children. For the sharing practices we expected more policies to engage in the SID, SL, and SC practices. The respective 10%, 12%, and 6% are rather small percentages for a presumably widely occurring practice, especially, given that we focus on policies of free apps that often rely on targeted advertising.

4 Mobile App Analysis

In order to compare our policy analysis results to what apps actually do according to their code we now turn to our app analysis approach. We first discuss our system design (§ 4.1) and follow up with our analysis results (§ 4.2).

4.1 App Analysis System Design

Our app analysis system is based on Androguard (2012), an open source static analysis tool written in Python that

[20]We also checked the random sample of 40 apps for policies dynamically loaded via JavaScript because for such policies the feature extraction would fail. We had observed such dynamic loading before. However, as neither of the policies in the sample was loaded dynamically, we do not make an adjustment in this regard.

3rd Party Library
Crashlytics/Fabric
Crittercism/Aptel.
Flurry Analytics
Google Analytics
Umeng
AdMob*
InMobi*
MoPub*
MillennialMedia*
StartApp*

Table 3: Analytics and ad* libraries.

Pract	*Base* n=40	*Acc$_{app}$* n=40	*95% CI* n=40	*Prec$_{pos}$* n=40	*Rec$_{pos}$* n=40	*F-1$_{pos}$* n=40	*F-1$_{neg}$* n=40	*Pos$_{w/\ pol}$* n=9,295	*Pos$_{w/o\ pol}$* n=8,696
CID	0.8	0.9	0.76–0.97	0.89	1	**0.94**	0.67	95%	87%
CL	0.55	0.8	0.64–0.91	0.73	1	**0.85**	0.71	66%	49%
CC	0.78	1	0.91–1	1	1	**1**	1	**25%**	12%
SID	0.68	0.95	0.83–0.99	1	0.93	**0.96**	0.93	**71%**	**62%**
SL	0.93	1	0.91–1	1	1	**1**	1	20%	16%
SC	0.98	1	0.91–1	1	1	**1**	1	**2%**	0%

Table 4: App analysis results for the app test set (n=40) and the percentages of practices' occurrences in the full app set (n=17,991). More specifically, $Pos\_w/\ pol$ and $Pos\_w/o\ pol$ are showing what percentage of apps engage in a given practice for the subset of apps in the full app set with a policy (n=9,295) and without a policy (n=8,696), respectively. We measure precision, recall, and F-1 scores with the $_{pos}$ and $_{neg}$ subscripts referring to the scores for the positive and negative classes.

provides extensible analytical functionality. Apart from the manual intervention in the construction and testing phase our system's analysis is fully automated. A brief example for sharing of device IDs will convey the basic program flow of our data-driven static analysis. For each app our system builds an API invocation map, which is utilized as a partial call graph. To illustrate, for sharing of device IDs all calls to the `TelephonyManager.getDevice.Id` API are included in the call graph because the caller can use it to request a device ID. All calls to this and other APIs that can be used to request a device ID are included in the call graph and passed to the identification routine, which checks the package names of the callers against the package names of selected third party libraries that we want to analyze, listed in Table 3. In order to make use of the `getDeviceId` API a library needs the `READ_PHONE_STATE` permission. Only if the analysis detects that the library has the required permission, the app is classified as sharing device IDs with third parties.[21] We identified relevant Android API calls for the types of information we are interested in and the permission each call requires by using PScout (Au et al. 2012).

Our static analysis is informed by a manual evaluation of Android and third party APIs. Because sharing of data most often occurs through third party libraries (Enck et al. 2011), we can leverage the insight that the observation of data sharing for a given library allows extension of that result to all apps using the same library (Gibler et al. 2012). As the top libraries have the farthest reach (Gibler et al. 2012) we focus on those. We used AppBrain (2015) to identify the ten most popular libraries by app count that process device ID, location, or contact data. To the extent we were able to obtain them we also analyzed previous library versions dating back to 2011. After all, apps sometimes continue to use older library versions even after the library has been updated. For each library we opened a developer account, created a sample app, and observed the data flows from the developer perspective. For these apps as well as for a sample of Google Play store apps that implement the selected libraries we additionally observed their behavior from the outside by capturing and decrypting packets via a man-in-the-middle attack and a fake certificate (Progress Software Corporation 2016). We also analyzed library documentations. These exercises allowed us to see or deduce which data types were sent out to which third parties.

[21] Android's permission model as of Android 6.0 does not distinguish between permissions for an app and permissions for the app's libraries, which, thus, can request all permissions of the app.

4.2 App Analysis Results

Table 4 shows our results for the app analysis, specifically, the occurrence of practices in the full app set and the performance for a set of 40 apps (app test set), which we selected randomly from the publishers in the policy test set to obtain corresponding app/policy test pairs (for our later performance analysis of potential inconsistencies in § 5). To check whether the data practices in the test apps were correctly analyzed by our system we dynamically observed and decrypted the data flows from the test apps to first and third parties, performed a manual static analysis for each test app with Androguard (2012), and studied the documentations of third party libraries. Thus, for example, we were able to infer from the proper implementation of a given library that data is shared as explained in the library's documentation. We did not measure performance based on micro-benchmarks, such as DroidBench (Arzt et al. 2014), as those do not fully cover the data practices we are investigating.

In the context of potential inconsistencies correctly identifying positive instances of apps' collection and sharing practices is more relevant than identifying negative instances because only practices that are occurring in an app need to be covered in a policy. Thus, the results for the data practices with rarely occurring positive test cases are especially noteworthy: CC, SL, and SC all reached *F-1$_{pos}$* $= 1$ indicating that our static analysis is able to identify positive practices even if they rarely occur. Further, the F-1$_{pos}$ scores, averaging to a mean of 0.96, show the overall reliability of our approach. For all practices the accuracy is also above the baseline of always selecting the test set class that occurs the most for a given practice.

For all six data practices we find a mean of 2.79 occurring practices per app for apps with policies and 2.27 occurrences for apps without policies. As all practices need to be described in a policy per our privacy requirements (§ 3.1), it is already clear that there are substantial amounts of potential inconsistencies between apps and policies simply due to missing policies. For example, the SID practice was de-

Pract	*Acc* *n=40*	*Acc$_{pol}$ · Acc$_{app}$* *n=40*	*95% CI* *n=40*	*Prec$_{pos}$* *n=40*	*Rec$_{pos}$* *n=40*	*F-1$_{pos}$* *n=40*	*F-1$_{neg}$* *n=40*	*MCC* *n=40*	*TP, FP, TN, FN* *n=40*	*Inconsistency* *n=9,050*
CID	**0.95**	0.74	0.83–0.99	0.75	1	**0.86**	0.97	0.84	6, 2, 32, 0	**50%**
CL	**0.83**	0.7	0.67–0.93	**0.54**	1	**0.7**	0.88	0.65	8, 7, 25, 0	**41%**
CC	**1**	0.88	0.91–1	-	-	-	1	-	0, 0, 40, 0	**9%**
SID	**0.85**	0.84	0.7–0.94	0.93	0.74	**0.82**	0.87	0.71	14, 1, 20, 5	**63%**
SL	**1**	0.93	0.91–1	1	1	**1**	1	1	3, 0, 37, 0	**17%**
SC	**1**	0.78	0.91–1	1	1	**1**	1	1	1, 0, 39, 0	**2%**

Table 5: Results for identifying potential inconsistencies in the app/policy test set (n=40) and the percentage of potential inconsistencies for all 9,050 app/policy pairs (Inconsistency). Assuming independence of policy and app accuracies, Acc$_{pol}$· Acc$_{app}$, that is, the product of policy analysis accuracy (Table 2) and app analysis accuracy (Table 4), indicates worse results than the directly measured accuracy. However, the Matthews correlation coefficient (MCC), a measure that is particularly insightful for evaluating classifier performance in skewed classes, indicates a positive correlation between the observed and predicted classes.

tected in 62% of apps that did not have a policy (Table 4), which, consequently, are potentially non-compliant with privacy requirements. Furthermore, for apps that had a policy only 10% disclosed the SID practice (Table 2) while it occurred in 71% of apps (Table 4). Thus, 61% of those apps are potentially non-compliant in this regard. The only practices for which we cannot immediately infer the existence of potential inconsistencies are the CC and SC practices with policy disclosures of 56% and 6% and occurrences in apps of 25% and 2%, respectively.

We want to point out various limitations of our static analysis. At the outset our approach is generally subject to the same limitations that all static analysis techniques for Android are facing, most notably, the difficulties of analyzing native code, obfuscated code, and indirect techniques (e.g., reflection). It is a further limitation that the identification of data practices occurs from the outside (e.g., server-side code is not considered). While this limitation is not a problem for companies' analysis of their own apps, which we see as a major application of our approach, it can become prevalent for regulators, for instance. Also, our results for the sharing practices only refer to the ten third parties listed in Table 3. The percentages for sharing of contacts, device IDs, or locations would almost certainly be higher if we would consider additional libraries. In addition, our definition of sharing data with a third party only encompasses sharing data with ad networks and analytics libraries.

5 Identifying Potential Inconsistencies

In this section we marry our policy (§ 3) and app (§ 4) analyses. We explore to which extent apps are potentially non-compliant with privacy requirements. We emphasize that app developers were found to lack an understanding of privacy-best practices (Balebako et al. 2014), and it could be that many of the potential inconsistencies that we found are a result of this phenomenon. Especially, many developers struggle to understand what type of data third parties receive, and with limited time and resources even self-described privacy advocates and security experts grapple with implementing privacy and security protection (Balebako et al. 2014). In this regard, our analysis can provide developers with a valuable indicator for instances of potential non-compliance. For identifying those instances only positive app classes and negative policy classes are relevant. In other words, if a data practice does not occur in an app, it does not need policy coverage because there can be no potential inconsistency to begin with. Similarly, if a user is notified about a data practice in a policy, it is irrelevant whether the practice is implemented in the app or not. Either way, the app is covered by the policy. Based on these insights we analyze the performance of our approach. Table 5 shows our results.

To check the performance of our system for correctly identifying potential inconsistencies we use a test set with corresponding app/policy pairs (app/policy test set). The set contains the 40 apps from our app test set (§ 4.2) and their associated policies from our policy test set (§ 3.3). We associate an app and a policy if the app or its Play store page links to the policy or if the policy explicitly declares itself applicable to mobile apps. As only 23 policies satisfy this requirement we associated some policies with multiple apps to which the respective policies are applicable. Making 240 classifications in the app/policy test set—that is, classifying six practices for each of the 40 app/policy pairs—our system correctly identified 32 potential inconsistencies (TP). It also returned five false negatives (FN), 10 false positives (FP), and 193 true negatives (TN). As shown in Table 5, accuracy results range between 0.83 and 1 with a mean of 0.94. Although not fully comparable, AsDroid achieved an accuracy of 0.79 for detecting stealthy behavior (Huang et al. 2014) and Slavin et al. (2016) report an accuracy of 0.8 for detecting discrepancies between app behavior and policy descriptions.

The F-1$_{pos}$ scores for our analysis, ranging from 0.7 to 1, indicate the overall reliable identification of potential inconsistencies. While we think that these results are generally promising, we obtain a relatively low precision value of $Prec_{pos} = 0.54$ for the CL practice. This result illustrates a broader point that is applicable beyond location collection. False positives seem to occur because our analysis takes into account too many Android system APIs that are only occasionally used for purposes of the data practice in question. Despite our believe that it is better to err on the side of false positives, which is especially true for an au-

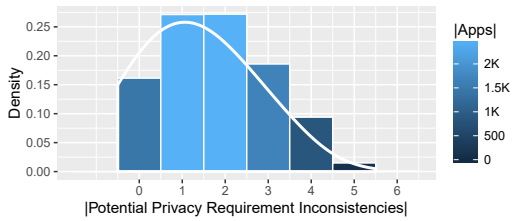

Figure 4: For the full app/policy set (n = 9,050) we found that 2,455 apps have one potential inconsistency, 2,460 have two, and only 1,461 adhere completely to their policy. Each app exhibits a mean of 1.83 (16,536/9,050) potential inconsistencies (with the following means per data practice: CID: 0.5, CL: 0.41, CC: 0.09, SID: 0.63, SL: 0.17, SC: 0.02).

diting system (Gibler et al. 2012), in hindsight we probably would have left out some APIs. The opposite problem seems to occur in the SID practice. We included too few relevant APIs. In this regard, we acknowledge the challenge of identifying a set of APIs that captures the bulk of cases for a given practice without being over-inclusive.

As indicated by the high percentages, potential inconsistencies seem to be a widespread phenomenon. Specifically, collection of device IDs and locations as well as sharing of device IDs are practices that appear to be inconsistent for 50%, 41%, and 63% of apps, respectively. It is further noteworthy that for SL and SC nearly every detection of the practice goes hand in hand with a potential inconsistency. For the apps that share location information (20%, per Table 4) nearly all (17%, per Table 5) do not properly disclose such sharing. Similarly, for the 2% of apps that share contact data only a handful provide sufficient disclosure. For the majority of those cases it is not even necessary to perform a policy analysis to detect potential inconsistencies.

From a big picture view, the average number of 1.83 potential inconsistencies per app is high compared to the closest previous averages with 0.62 (113/182) cases of stealthy behavior (Huang et al. 2014) and potential privacy violations of 1.2 (24/20) (Enck et al. 2010) and 0.71 (341/477) (Slavin et al. 2016). Figure 4 shows the details. It should also be noted that for apps without a policy essentially every data collection or sharing practice causes a potential inconsistency. For example, all 62% apps without a policy that share device IDs (Table 4) are potentially inconsistent. Thus, overall our results suggest a broad level of potential inconsistencies between apps and policies.

6 Future Directions

The law of notice and choice is intended to enable enforcement of data practices in mobile apps and other online services. However, verifying whether an app actually behaves according to the law and its privacy policy is decisively hard. To alleviate this problem we propose the use of an automated analysis system based on machine learning and static analysis. Our system advances app privacy in three main thrusts: it increases transparency for otherwise largely opaque data practices, offers the scalability necessary for potentially making an impact on the app eco-system as a whole, and provides a first step towards the automation of privacy requirement checks.[22]

Our results suggest that potential privacy requirements inconsistencies are quite common in mobile apps. Results from our analysis could be used to prioritize further manual analysis of apps for compliance with relevant regulations. While we focused on the Android platform, our approach is, in principle, adaptable to other mobile platforms, for example, for iOS using previous works (Deng et al. 2015; Kurtz et al. 2014). Our approach can also be made workable for analyzing website practices, e.g., leveraging the work of Sen et al. (2014), for which first and third party cookies and other tracking mechanisms can be observed to evaluate collection and sharing of data. The Internet of Things and sensor data represent other rich use cases. Fitness trackers with APIs for monitoring the heart rate and other body sensor data could be a first step towards exploring these areas.

We believe that it is necessary to develop public policy and law alongside the privacy requirement analysis system we propose. In our opinion, regulators are moving in the right direction by pushing for app store standardization (California Department of Justice 2012) and early enforcement of potentially invasive privacy practices (FTC 2014a). Approaches like the one proposed here can relieve regulators through automation and allow them to focus their limited resources to move from a purely reactionary approach towards more systematic oversight. As we also think that many software publishers do not intend non-compliance with privacy requirements, but rather lose track of their obligations or are unaware of them, we also advocate for implementation of a privacy law check in software development tools and as part of the app vetting process in app stores. Given their broad access to app code, app stores are in a unique position to leverage the approach described in this paper.

7 Acknowledgments

This material is based upon work supported in part by the National Science Foundation under grants CNS-1330596, CNS-1330214, and SBE-1513957, as well as by DARPA and the Air Force Research Laboratory, under agreement number FA8750-15-2-0277. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes not withstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA, the Air Force Research Laboratory, the National Science Foundation, or the U.S. Government.

[22] We started a collaboration with the California Attorney General to further validate a version of our system customized to automatically identify mobile apps that may be in violation of CalOPPA (for example, because a company shares personal contact information collected by its app with third parties but does not disclose this practice in its privacy policy).

References

Androguard. 2012. http://doc.androguard.re/html/index.html. Last accessed: September 8, 2016.

AppBrain. 2015. Appbrain. http://www.appbrain.com/stats/libraries/. Last accessed: September 8, 2016.

Arzt, S.; Rasthofer, S.; Fritz, C.; Bodden, E.; Bartel, A.; Klein, J.; Le Traon, Y.; Octeau, D.; and McDaniel, P. 2014. Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps. In *PLDI '14*. ACM.

Ashley, P.; Hada, S.; Karjoth, G.; Powers, C.; and Schunter, M. 2003. Enterprise Privacy Authorization Language (EPAL 1.2). Technical report, IBM.

Au, K. W. Y.; Zhou, Y. F.; Huang, Z.; and Lie, D. 2012. Pscout: Analyzing the android permission specification. In *CCS '12*. ACM.

Balebako, R.; Marsh, A.; Lin, J.; Hong, J.; and Cranor, L. F. 2014. The privacy and security behaviors of smartphone app developers. In *Workshop on Usable Security (USEC)*.

California Department of Justice. 2012. Attorney general Kamala D. Harris secures global agreement to strengthen privacy protections for users of mobile applications. http://www.oag.ca.gov/news/press-releases/attorney-general-kamala-d-harris-secures-global-agreement-strengthen-privacy. Last accessed: September 8, 2016.

California Department of Justice. 2014. Making your privacy practices public. https://oag.ca.gov/sites/all/files/agweb/pdfs/cybersecurity/making_your_privacy_practices_public.pdf. Last accessed: September 8, 2016.

Chundi, P., and Subramaniam, P. M. 2014. An Approach to Analyze Web Privacy Policy Documents. In *KDD Workshop on Data Mining for Social Good*.

Costante, E.; Sun, Y.; Petković, M.; and den Hartog, J. 2012. A machine learning solution to assess privacy policy completeness. In *WPES '12*. ACM.

Costante, E.; den Hartog, J.; and Petkovic, M. 2013. What websites know about you: Privacy policy analysis using information extraction. In *Data Privacy Management '13*. Springer.

Cranor, L. F.; Langheinrich, M.; Marchiori, M.; Presler-Marshall, M.; and Reagle, J. M. 2002. The Platform for Privacy Preferences 1.0 (P3P1.0) specification. World Wide Web Consortium, Recommendation REC-P3P-20020416.

Cranor, L. F.; Idouchi, K.; Leon, P. G.; Sleeper, M.; and Ur, B. 2013. Are they actually any different? comparing thousands of financial institutions' privacy practices. In *WEIS '13*.

Deng, Z.; Saltaformaggio, B.; Zhang, X.; and Xu, D. 2015. iris: Vetting private api abuse in ios applications. In *CCS '15*. ACM.

Enck, W.; Gilbert, P.; Chun, B.-G.; Cox, L. P.; Jung, J.; McDaniel, P.; and Sheth, A. N. 2010. Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones. In *OSDI'10*. USENIX Assoc.

Enck, W.; Octeau, D.; McDaniel, P.; and Chaudhuri, S. 2011. A study of android application security. In *USENIX '11*. USENIX Assoc.

FTC. 1998. Privacy online: A report to congress. https://www.ftc.gov/reports/privacy-online-report-congress. Last accessed: September 8, 2016.

FTC. 2012a. Mobile apps for kids: Current privacy disclosures are disappointing. http://www.ftc.gov/os/2012/02/120216mobile_apps_kids.pdf. Last accessed: September 8, 2016.

FTC. 2012b. Mobile apps for kids: Disclosures still not making the grade. https://www.ftc.gov/reports/mobile-apps-kids-disclosures-still-not-making-grade. Last accessed: September 8, 2016.

FTC. 2013. Mobile privacy disclosures. www.ftc.gov/os/2013/02/130201mobileprivacyreport.pdf. Last accessed: September 8, 2016.

FTC. 2014a. FTC warns children's app maker BabyBus about potential COPPA violations. https://www.ftc.gov/news-events/press-releases/2014/12/ftc-warns-childrens-app-maker-babybus-about-potential-coppa. Last accessed: September 8, 2016.

FTC. 2014b. What's the deal? a federal trade commission study on mobile shopping apps. https://www.ftc.gov/reports/whats-deal-federal-trade-commission-study-mobile-shopping-apps-august-2014. Last accessed: September 8, 2016.

FTC. 2015. Kids' apps disclosures revisited. https://www.ftc.gov/news-events/blogs/business-blog/2015/09/kids-apps-disclosures-revisited.

Ghazinour, K.; Majedi, M.; and Barker, K. 2009. A model for privacy policy visualization. In *2009 33rd Annual IEEE International Computer Software and Applications Conference*, volume 2, 335–340. IEEE.

Gibler, C.; Crussell, J.; Erickson, J.; and Chen, H. 2012. Androidleaks: Automatically detecting potential privacy leaks in android applications on a large scale. In *TRUST'12*. Springer.

GPEN. 2015. 2014 annual report. https://www.privacyenforcement.net/node/513.

Hoke, C.; Cranor, L.; Leon, P.; and Au, A. 2015. Are They Worth Reading? An In-Depth Analysis of Online Trackers Privacy Policies. *I/S : a journal of law and policy for the information society*.

Huang, J.; Zhang, X.; Tan, L.; Wang, P.; and Liang, B. 2014. Asdroid: Detecting stealthy behaviors in android applications by user interface and program behavior contradiction. In *ICSE '14*. ACM.

Kurtz, A.; Weinlein, A.; Settgast, C.; and Freiling, F. 2014. Dios: Dynamic privacy analysis of ios applications. Technical Report CS-2014-03, Friedrich-Alexander-Universität Erlangen-Nürnberg, Dept. of Computer Science.

Liu, F.; Ramanath, R.; Sadeh, N.; and Smith, N. A. 2014. A step towards usable privacy policy: Automatic alignment of privacy statements. In *COLING '14*.

Manning, C. D.; Raghavan, P.; and Schütze, H. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

Massey, A. K.; Eisenstein, J.; Antón, A. I.; and Swire, P. P. 2013. Automated text mining for requirements analysis of policy documents. In *RE '13*.

Olmstead, K., and Atkinson, M. 2015. Apps permissions in the Google Play store. `http://www.pewinternet.org/2015/11/10/apps-permissions-in-the-google-play-store/`. Last accessed: September 8, 2016.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; and Duchesnay, E. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

Progress Software Corporation. 2016. Fiddler. `http://www.telerik.com/fiddler`. Last accessed: September 8, 2016.

Ramanath, R.; Liu, F.; Sadeh, N.; and Smith, N. A. 2014. Unsupervised alignment of privacy policies using hidden markov models. In *ACL '14*.

Reidenberg, J. R.; Breaux, T.; Cranor, L. F.; French, B.; Grannis, A.; Graves, J. T.; Liu, F.; McDonald, A.; Norton, T. B.; Ramanath, R.; Russell, N. C.; Sadeh, N.; and Schaub, F. 2015. Disagreeable privacy policies: Mismatches between meaning and users' understanding. *Berkeley Technology Law Journal* 30(1):39–88.

Reidsma, D., and Carletta, J. 2008. Reliability measurement without limits. *Comput. Linguist.* 34(3):319–326.

Sadeh, N.; Acquisti, A.; Breaux, T. D.; Cranor, L. F.; McDonald, A. M.; Reidenberg, J. R.; Smith, N. A.; Liu, F.; Russell, N. C.; Schaub, F.; and Wilson, S. 2013. The usable privacy policy project. Tech. report CMU-ISR-13-119, Carnegie Mellon University.

Sen, S.; Guha, S.; Datta, A.; Rajamani, S. K.; Tsai, J.; and Wing, J. M. 2014. Bootstrapping privacy compliance in big data systems. In *SP '14*. IEEE Comp. Soc.

Slavin, R.; Wang, X.; Hosseini, M.; Hester, W.; Krishnan, R.; Bhatia, J.; Breaux, T.; and Niu, J. 2016. Toward a framework for detecting privacy policy violation in android application code. In *ICSE '16*.

Stamey, J. W., and Rossi, R. A. 2009. Automatically identifying relations in privacy policies. In *SIGDOC '09*. ACM.

Wilson, S.; Schaub, F.; Dara, A.; Cherivirala, S. K.; Zimmeck, S.; Andersen, M. S.; Leon, P. G.; Hovy, E.; and Sadeh, N. 2016a. Demystifying privacy policies with language technologies: Progress and challenges. In *Proceedings of LREC 1st Workshop on Text Analytics for Cybersecurity and Online Safety*, TA-COS '16. Portorož, Slovenia: ELRA.

Wilson, S.; Schaub, F.; Dara, A. A.; Liu, F.; Cherivirala, S.; Leon, P. G.; Andersen, M. S.; Zimmeck, S.; Sathyendra, K. M.; Russell, N. C.; Norton, T. B.; Hovy, E.; Reidenberg, J.; and Sadeh, N. 2016b. The creation and analysis of a website privacy policy corpus. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, ACL '16. Berlin, Germany: ACL.

Wilson, S.; Schaub, F.; Ramanath, R.; Sadeh, N.; Liu, F.; Smith, N. A.; and Liu, F. 2016c. Crowdsourcing annotations for websites' privacy policies: Can it really work? In *WWW '16*.

Yu, L.; Luo, X.; Liu, X.; and Zhang, T. 2016. Can we trust the privacy policies of android apps? In *DSN '16*.

Zimmeck, S., and Bellovin, S. M. 2014. Privee: An architecture for automatically analyzing web privacy policies. In *USENIX '14*. USENIX Assoc.

Zimmeck, S. 2013. The information privacy law of web applications and cloud computing. *Santa Clara Computer & High Tech. L.J.* 29(3):451–487.