electronics

MDPI

Article

Privacy Assessment in Android Apps: A Systematic Mapping Study

Jose M. Del Alamo [1,*,†], Danny Guaman [1,2,†], Belen Balmori [1,†] and Ana Diez [1,†]

[1] ETSI Telecomunicación, Universidad Politécnica de Madrid, 28040 Madrid, Spain; danny.guaman@epn.edu.ec (D.G.); b.balmori@alumnos.upm.es (B.B.); ana.diezm@upm.es (A.D.)
[2] Departamento de Electrónica, Telecomunicaciones y Redes de Información, Escuela Politécnica Nacional, Quito 170517, Ecuador
* Correspondence: jm.delalamo@upm.es
† These authors contributed equally to this work.

Abstract: Android apps are daily installed by billions of users worldwide, who grant access to an extensive set of sensitive personal data. Different techniques have been developed over the years to understand how apps protect or harm their users' privacy. However, these results have been produced in different research domains and addressing privacy from different perspectives, resulting in a growing but scattered body of knowledge. To bridge this gap, we have carried out a systematic mapping study to provide practitioners and researchers with an overview of the state-of-the-art technique, published between 2016 and 2020, to assess privacy in Android apps. In this paper, we highlight the most relevant findings, identify and analyse the most pressing gaps, and discuss the promising research directions.

Keywords: Android; apps; data protection; privacy; quality assessment; software quality

Citation: Del Alamo, J.M.; Guaman, D.; Balmori, B.; Diez, A. Privacy Assessment in Android Apps: A Systematic Mapping Study. *Electronics* **2021**, *10*, 1999. https://doi.org/10.3390/electronics10161999

Academic Editor: Vijayakumar Varadarajan

Received: 9 July 2021
Accepted: 13 August 2021
Published: 18 August 2021

1. Introduction

Mobile apps pose great privacy risks to their users [1]. We carry our smartphones from dusk till dawn; they have thus become ubiquitous and gain access to a vast quantity of sensitive personal data, which are shared worldwide with different service providers in a complex ecosystem [2], even without the app developer's knowledge [3]. Android is the dominant platform in this market with a share circa 85% [4], and according to Zang et al. [5] Android apps leak more personal data to third parties than iOS applications on average.

In this context, testing or auditing Android applications trustworthiness has become a hot research topic. Researchers have leveraged static analysis to find anomalies in apps' code [6]. Dynamic analysis further supports examining the real behaviour of the apps at runtime [7]. Machine learning algorithms have recently been introduced to sign potential misbehaviours [8]. They all have been applied to spot known issues, assess new risks, or check whether some criteria or conditions are met by Android apps.

Understanding what are the available techniques and tools for assessing privacy in Android apps is significant for different stakeholders. Developers and testers can leverage the more matured techniques to identify and fix privacy issues, thus improving the quality of the apps under assessment. Data protection authorities and auditors can apply them to find privacy-related anomalies in third-party apps. Researchers need to understand the state of the art so that they can identify and pursue new research avenues.

However, having a proper overview of the available contributions in this field is challenging. To begin with, researchers approach the problem from different perspectives, e.g., addressing different privacy threats or following different privacy paradigms. Further, the contributions are scattered in different domains such as software engineering, cybersecurity, or computer networks, resulting in a growing but dispersed body of knowledge.

Some previous works have provided insight into the topic but were carried out more than five years ago [6,7] or addressed security instead of privacy [9,10].

To fill this gap, this paper presents a comprehensive overview of the different privacy assessment techniques for Android apps reported in the state of the art. The scope of our study is thus defined by the intersection of three domains, namely, software quality assessment, privacy, and Android applications. Our purpose is to identify the most promising techniques available and the current gaps and discuss promising research directions.

Following a systematic process we have identified 10,825 references published between 2016 and 2020, and after a filtering process we selected 79 papers that describe privacy assessment techniques applicable to Android apps. We have identified and classified the goals of the techniques, including the privacy threats they target and the level of detail to which they are applied. We have further investigated how the assessment is carried out and analysed the overall quality level achieved. All this evidence supports a discussion on the trends observed, which may help researchers to identify and focus their attention on new promising opportunities in this growing field. Our outcomes suggest a switch in the privacy paradigms considered, a lack of techniques assessing the app compliance with both the declared policy and the applicable legislation, and the need to support developers in this process.

The paper is organised as follows. Section 2 provides a summary of the core concepts covered in our study. Section 3 details the research method we have followed, including our research questions and classification scheme. Section 4 details the results of our mapping study, answering the research questions. Section 5 discusses the results, further analysing trends and highlighting the gaps observed. Section 6 examines the threats to the study validity. Section 7 analyses the related work detailing the main differences with our study. Finally, Section 8 concludes the paper.

2. Background

This section provides a summary of the fundamental concepts covered in this study.

2.1. Privacy Assessment

Since privacy is a plural and contestable concept there is still no single, widely accepted definition. One of the salient privacy taxonomies in the field is the one proposed by LINDDUN [11], which identifies seven common privacy threats found in software systems. Hansen et al. [12] proposed three personal data protection goals which partially overlap with LINDDUN identifiability and unawareness threats and further highlight the lack of control as a relevant threat to users' privacy. We have used these privacy threats classifications to understand what is assessed by the different techniques (Table 1).

Having different privacy threats results in distinct approaches to assess them (Table 2). For example, some techniques aim at detecting the presence of a given, previously known privacy issue succeeding when it is found, or its presence can be discarded. Others aim to verify whether an app meets a given privacy specification, whatever it is. Yet some other techniques may assess the risk posed to an individual's privacy if some feared event materialises, e.g., considering its likelihood and impact.

Even when the privacy threat and assessment approach are set, researchers can focus their attention at different levels of detail including a library, a module, the whole app, or a set of apps. App developers usually include third party libraries in their code to simplify or reuse features, but these libraries may include code that threatens the user's privacy, as demonstrated by Stevens et al. [13]. Android apps can also be organised in modules to customise features delivery, which are downloaded and installed as required [14]. Whole apps can be analysed in isolation or while collaborating, e.g., to detect collusive apps.

2.2. Assessment Techniques

The SWEBOK v3.0 identifies two major types of techniques to assess the quality of software systems, i.e., static and dynamic analysis [15]. Lately, ML (Machine Learning)-

based techniques have been introduced in software quality control where, e.g., a set of features are identified to sign a potential issue drawing from previous experiences [16].

Table 1. Privacy threats subject to assessment.

Privacy Threat	Description
Linkability	*"An adversary is able to link two items of interest without knowing the identity of the data subject(s) involved."* [11].
Identifiability	*"An adversary is able to identify a data subject from a set of data subjects through an item of interest."* [11].
Non-repudiation	*"The data subject is unable to deny a claim (e.g., having performed an action, or sent a request)."* [11].
Detectability	*"An adversary is able to distinguish whether an item of interest about a data subject exists or not, regardless of being able to read the contents itself."* [11].
Disclosure of information	*"An adversary is able to learn the content of an item of interest about a data subject."* [11].
Unawareness	*"The data subject is unaware of the collection, processing, storage, or sharing activities (and corresponding purposes) of the data subject's personal data."* [11]. Unawareness relates to the transparency measures that a system can take to guide and educate the user concerning the personal data processing (e.g., collection and disclosure) and nudge the user into a more privacy-aware use of the system [12].
Non-compliance	*"The processing, storage, or handling of personal data is not compliant with legislation, regulation, and/or policy."* [11]. We are further interested in the particular regulation/standard being evaluated (e.g., GDPR, COPPA) and, if mentioned, the concrete principle (e.g., purpose limitation of GDPR).
Lack of control	Control enables users to decide what kind or information is processed about them. Focusing on data protection rights, *"Intervenability is defined as the property that intervention is possible concerning all ongoing or planned [personal] data processing."* [12]. Intervenability enables, e.g., to exercise the individuals' rights to rectification and erasure of personal data, providing and withdrawing consent, and so on.

Table 2. Assessment approach pursued by the techniques.

Assessment Approach	Description
Issue detection	The assessment aims to find privacy-related defects, i.e., a vulnerability or deficiency that potentially may threaten a privacy aspect. A detection technique seeks to find defects or defect patterns already known. For example, dynamic permission requests with missing revocation statements.
Specification verification	The assessment aims to verify that an app or its components meet a specification, which can stem, e.g., from a policy or regulation. For example, verifying whether a personal data flow complies with a privacy policy.
Risk assessment	The evaluation aims to understand the privacy risk posed by an app. Note that the source of risk may vary, and different authors may consider different aspects to assess the risk e.g., permissions, network connections, etc. The risk assessment output may be qualitative (e.g., a risk level) or quantitative (e.g., a risk score).

Static analysis techniques do not need to execute the software. Instead, they analyse and check any readable representation of the app, such as the program source code, its config files such as the manifest or the layout files, or its policy. In fact, static analysis can be applied to different types of code, i.e., source (Java, Kotlin, JavaScript, C ...), binary (DEX) or an intermediate representation (Smali and Jimple). Li et al. documented a set of challenges for static analysis techniques in Android [6], including the use of native, obfuscated or dynamically loaded code, or dealing with inter-component communications.

Dynamic analysis techniques execute the software in order to observe its real behaviour. Kong et al. identified three main stages in dynamic analysis [17], namely the

generation of execution paths, the program execution, and the observation of the resulting behaviour. The first one describes how the interaction with the app takes place i.e., the approach used to generate the system and user events. Symbolic execution, model-based testing, fuzzing, random generation, or actual user inputs can be used for this purpose. Once generated, the tests can be executed in different environments, mainly in an emulator or an actual device. Finally, the app behaviour can be observed because the app itself logs its execution, the environment is instrumented at the virtual machine or kernel levels, or through the interception of network communications.

ML-based techniques rely on models generated by analysing a set of apps to detect privacy issues in other apps. ML techniques can be classified into three major subtypes, namely supervised, unsupervised, and reinforcement learning [16]. A supervised ML technique relies on known (labelled) training samples to learn how to assess an app, and the resulting model is applied to assess unknown apps. An unsupervised algorithm does not require labelling of the training set but seeks to find hidden groups (i.e., clustering) or to reposit the input source into a different variable space (i.e., feature extraction). Reinforcement learning techniques either drop any outcome that does not lead to favourable expectations by punishing them or reward positive outcomes, thus the "reinforcement" aspect.

A combination of the aforementioned techniques is also possible. Different techniques can be pipelined so that the output from one technique is the input for the next one. For example, the app features extracted through static analysis can be used to train a ML algorithm signalling risky apps.

Deep Learning (DL) techniques advanced these learning techniques by the use of artificial neural networks that can learn and make intelligent decisions on their own.

2.3. Technique Quality

In a growing research field sometimes, it is difficult to separate the wheat from the chaff and filter out low-quality results. As a proxy to understand the overall quality level of the techniques found, we pay attention to how well they have been assessed by their authors and whether the techniques and/or its assessment results are easy to replicate by other researchers.

Wieringa et al. [18] elaborated a taxonomy identifying the maturity of a research result based on the efforts its authors took to assess it. While Wieringa et al.'s taxonomy defines five types of assessment, we are only interested in those techniques that at least have demonstrated their feasibility (Table 3).

Table 3. Type of assessment approach.

Type of Assessment	Description
Validation	The authors demonstrate the feasibility of their contributions by means of an illustrative (demonstration) scenario. There is no formal proof of correctness nor empirical evaluation.
Formal proof	The authors assess the correctness of their contributions by means of mathematical proof.
Empirical evaluation	The authors assess the features of their proposal with experiments. Different properties can be evaluated such as the effectiveness or the performance of the proposal. The approach to report the results might be qualitative or quantitative. Different apps sets can be used in the evaluation process, e.g., ad-hoc custom sets developed/adapted by the authors, reference benchmarks containing apps with known privacy issues, or in-the-wild apps from real app stores. Finally, the evaluation may include the results comparison with other tools.

3. Methodology

This research is based on the general guidelines for conducting a systematic mapping study (SMS) proposed by Petersen et al. [19]. An SMS is a systematic approach that aims to provide an overview of a research area of interest by showing quantitative evidence to identify trends.

Our research is organised in three main stages: planning, conducting, and reporting. The planning stage included the definition of the study scope, the main goal, and the research questions. In addition, we formulate the search strategy, the exclusion and inclusion criteria for the candidate papers, the classification scheme, and the codification process. The objective of the conducting stage was to answer the research questions (RQs). For that purpose, three tasks were defined, namely, the paper selection, the application of the inclusion/exclusion criteria, and the coding of the selected papers using the classification scheme defined. Finally, we summarised and analysed the results in the reporting stage by answering the RQs and looking for trends and gaps in the field.

3.1. Scope and Research Questions

The scope of this research is the common theme between three domains: (1) software quality, (2) privacy, and (3) Android applications. Our overall goal is twofold (i) to identify the most promising techniques available to assess privacy-related aspects in Android apps and (ii) to identify current gaps and discuss promising future research directions. It has been broken down into concrete RQs:

RQ1. What privacy issues have been assessed in Android apps? We aim to gain knowledge about the privacy threats that have been evaluated, the approaches taken, and the level of detail reached.

RQ2. What techniques are used to assess these privacy issues? Our goal is to understand what type of techniques have been proposed for finding privacy related anomalies, their status and current challenges.

RQ3. What is the quality level of the techniques identified? We are interested in investigating the quality level of the techniques and tools found, to identify those worth following or applying.

3.2. Paper Search Strategy

We used Scopus and DBLP databases to find high-quality refereed research literature, as they indexed the highest number of unique articles in the computer science field [20]. Scopus indexes the most important digital libraries such as IEEE Xplore, Springer Link, Elsevier, Science Direct, or ACM. DBLP nicely complements Scopus as it also indexes conference proceedings not hosted by the aforementioned digital libraries.

We built a Scopus search string based on our goal and research questions. The terms for the search string include those related to software quality control defined in [15,21], terms proposed by a senior privacy engineering researcher who supervises this study, and synonyms obtained from taxonomies and vocabularies, e.g., 2017 IEEE Thesaurus Version 1.0 [21], ACM Computing Classification System 2012 Revision [22], and Systems and Software Engineering—Vocabulary ISO/IEC/IEEE 24765 [23].

Our search string was the conjunction (AND) of the three research domains considered in our study, i.e., assessment, privacy, and Android. Each domain was represented as a disjunction (OR) of all domain-relevant terms. We used a test set of 10 relevant papers provided by a senior privacy engineering researcher to validate the search coverage of relevant literature. The initial search string was refined trying to retrieve all the papers in our test set. The final search string was obtained after five iterations. The details of the iterations and the validation are available in the replication package [24]. The final search string is presented below and was used to carry out a search on 14 October 2020, obtaining 3147 papers.

```
 ( "quality assurance" OR "quality control" OR "quality assessment" OR "validation" OR "verification"
OR "testing" OR "compliance" OR "compliant" OR ( "static" W/3 "analy*" ) OR ( "dynamic" W/3 "analy*"
) OR ( "traffic" W/3 "analy*" ) OR ( "static" W/3 "technique" ) OR ("dynamic" W/3 "technique" ) OR (
"analy*" W/3 "privacy" ) OR ( "evaluat*" W/3 "privacy" ) OR ( "asses*" W/3 "privacy" ) ) AND
( "privacy" OR "data protection" OR ( "data" W/3 "leak*" ) OR ( "information" W/3 "leak*" ) ) AND
( "android" OR "app" OR "mobile" OR "smartphone" )
```

To mitigate the threat to validity of not including all relevant papers, we included in our search the top 20 conferences in Software Systems, Computer Networks, and Computer Security categories as for GScholar (https://scholar.google.com/citations?view_op=top_venues&hl=en (accessed on 14 October 2020)) H5-Index. The H5-index is the h-index for articles published in the last five complete years. It is the largest number h such that h articles published in 2015–2019 have at least h citations each. The greater H5-index, the better the conference.

Scopus included all these conference proceedings but for TACAS, INFOCOM, ICC (Scopus does include ICC2016 proceedings), NDSS, PoPETs and ESORICS. Thus, we searched them in DBLP on 14 October 2020, obtaining 7678 papers from these conferences (Figure 1).

3.3. Inclusion and Exclusion Procedure

Once all the possible candidate papers were gathered, we conducted an inclusion and exclusion procedure to further filter the papers. This procedure consisted of an automated filter followed by a manual one.

3.3.1. Automated Inclusion—Exclusion Procedure

A paper must meet all the following inclusion criteria to pass to the screening stage:

- Research Field: computer science, engineering, decision sciences, or multidisciplinary.
- Publication date: >2015. Note that the paper selection phase of this study took place in the first half of October 2020. Only papers included in Scopus until Sept 2020 are included in our search.
- Document type: conference paper and article.
- Language: English.
- Citations: for papers published between 2016 and 2019, the minimum number of citations in a paper should have to fall in the 50% percentile of papers in computer science, as per Thomson Reuters [25]. For papers published in 2020, no citations are required. The rules are stated in Table 4.
- Number of pages: we are looking for papers proposing contributions with some form of validation, and this requires extensive works with detailed publications. Thus, we exclude short papers, i.e., heuristically, papers with less than five pages. If no page numbers are reported then the paper is included.

Scopus provides features for automatically checking all the inclusion criteria except for the number of citations and number of pages, which was checked with an Excel file. After Scopus-based filtering, 1474 papers were selected. The filter based on citations received and number of pages produced 948 papers. We found 11 duplicates; therefore, 937 papers were finally included in the next stage.

Table 4. Results for filtering by number of citations.

Publishing Year	Minimum Citations
2016	4
2017	3
2018	2
2019	1
2020	0

Conference papers from TACAS, INFOCOM, ICC, NDSS, PoPETS, and ESORICS were retrieved from DBLP. These papers are all conference papers, from the computer science field, published between 2016 and 2020, and written in English. DBLP provides details on citations only at individual paper level, thus bulk processing is not possible. This inclusion criterion was manually checked after the title-based screening. The two most-experienced researchers (first and second authors) individually read all the titles for conference papers selecting those that addressed any of the individual research domains we are targeting, i.e., assessment, privacy, or Android apps. All papers selected by any of these two researchers were included at this stage. We initially obtained 7678 papers from DBLP, which were reduced to 38 after title-base screening and to 34 papers after applying the citation-based filter.

This inclusion criterion was manually checked after a title-based screening. The two most-experienced researchers (first and second authors) individually read all the titles for conference papers selecting those that addressed any of the individual research domains we are targeting, i.e., assessment, privacy, or Android apps. All papers selected by any of these two researchers were included at this stage.

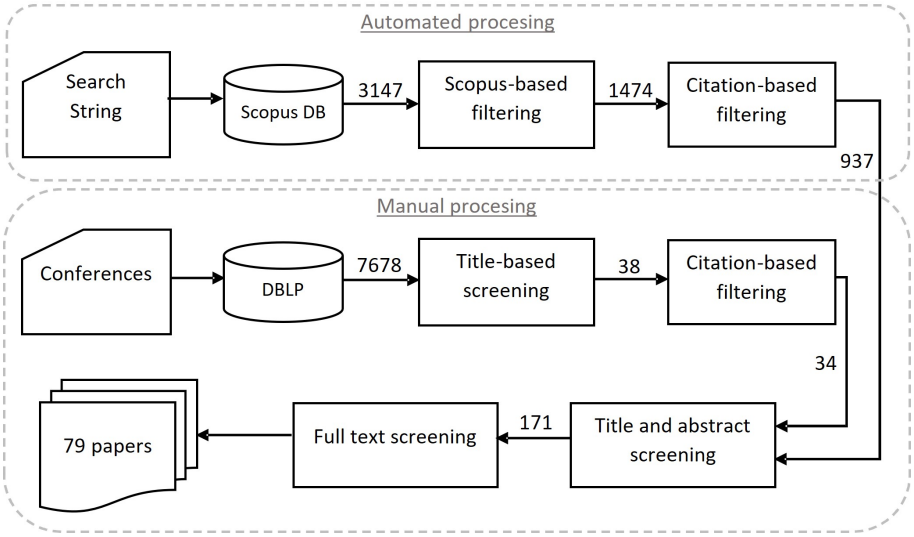

Figure 1. Filtering process.

3.3.2. Manual Inclusion—Exclusion Procedure

This procedure's goal is to include primary contributions that focused on the intersection of the software quality, privacy, and Android apps' domains. To this end we carried out two screening phases, the first one based on titles and abstract and the second one based on full texts. The tool CADIMA (https://www.cadima.info/(accessed on 16 August 2021)) was used to support the process, and the inclusion criteria was applied as a decision tree represented in Figure 2. Each paper was reviewed by two researchers of the team, made up of the four authors of the publication, and inconsistencies were discussed by the whole team to get an agreement.

The team performed a two-iteration pilot with 60 papers (ten in the first iteration and 50 in the final one) to normalise their criteria for including and excluding papers. After obtaining a 90% success rate and a "good" Krippendorff's alpha inter-coder reliability coefficient [26] of 0.78, the team then moved to the main screening stage.

In the title and abstract screening, 971 papers were evaluated and 800 of them were excluded. In the full text screening, 171 papers were evaluated and 92 of them were excluded. Once the screening phase was completed, 79 papers were selected for the classification procedure. From now on we use the paper ID to identify the papers included, e.g., ID179, which can be found in Appendix A. Details on all the papers analysed can be found in the replication package [24].

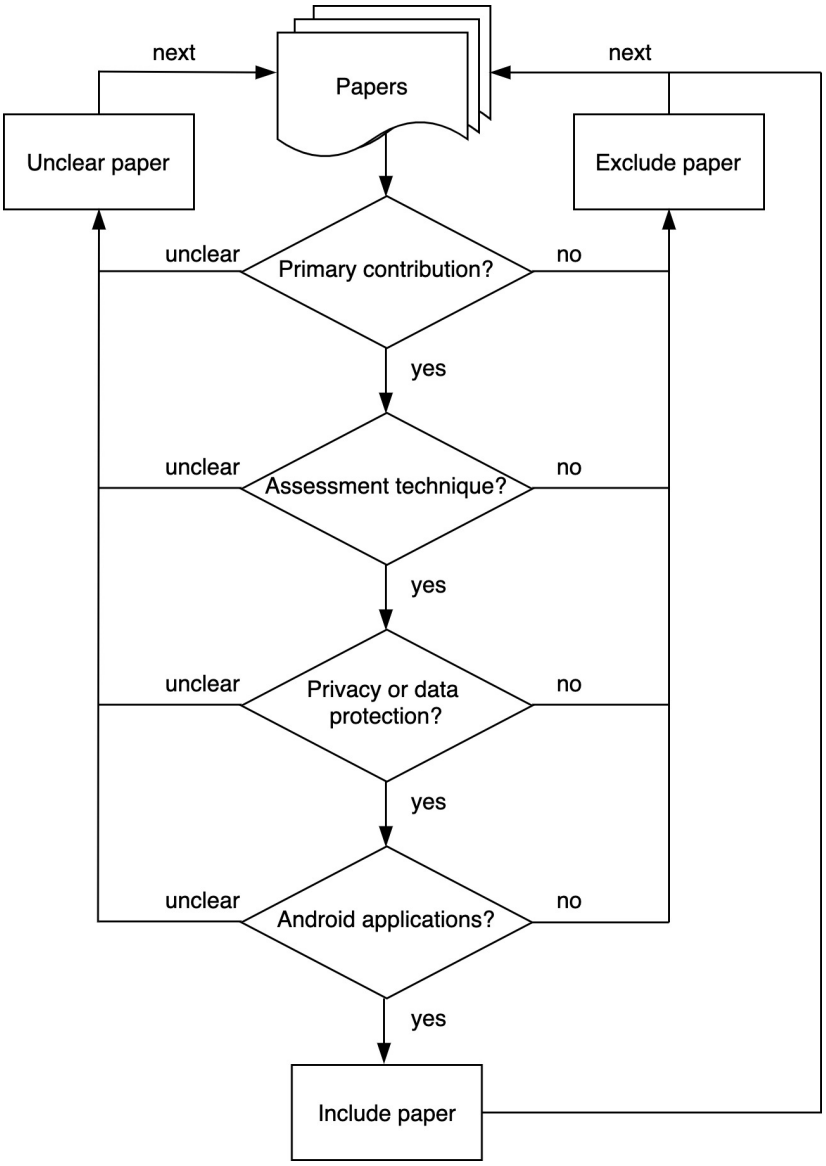

Figure 2. Inclusion and exclusion decision tree.

3.4. Classification Scheme and Procedure

As suggested in the Petersen guidelines [19], we defined our initial classification scheme based on existing taxonomies and classifications widely recognised in the research community in order to support comparability. The scheme evolved by adding new categories (e.g., risk assessment) and correcting others. The final scheme is presented in Figure 3.

The 79 papers selected in the screening stage were classified by a team of four researchers (the authors) using the scheme presented in Figure 3. Full details of the classification criteria are described in the codebook, which can be found in the replication package [24]. All the attributes were coded using an online form in Google Docs. The coders carried out a comprehensive reading of the main sections of the text which described the privacy assessment technique and its evaluation. Each paper was coded by two coders.

The first phase of the classification process was a five-iteration pilot, whose main objective was to unify the criteria of the coders. In each iteration, five papers were evaluated by all four researchers, and inconsistencies were discussed until an agreement was reached. In this phase, there were two main error types: non-systematic errors due to depth of reading and systematic errors due to disagreements in criteria. The latter were discussed

and agreed upon, clarifying the coding criteria as needed. We moved on to the next phase after reaching a Krippendorf's alpha coefficient greater than 0.75 for all criteria.

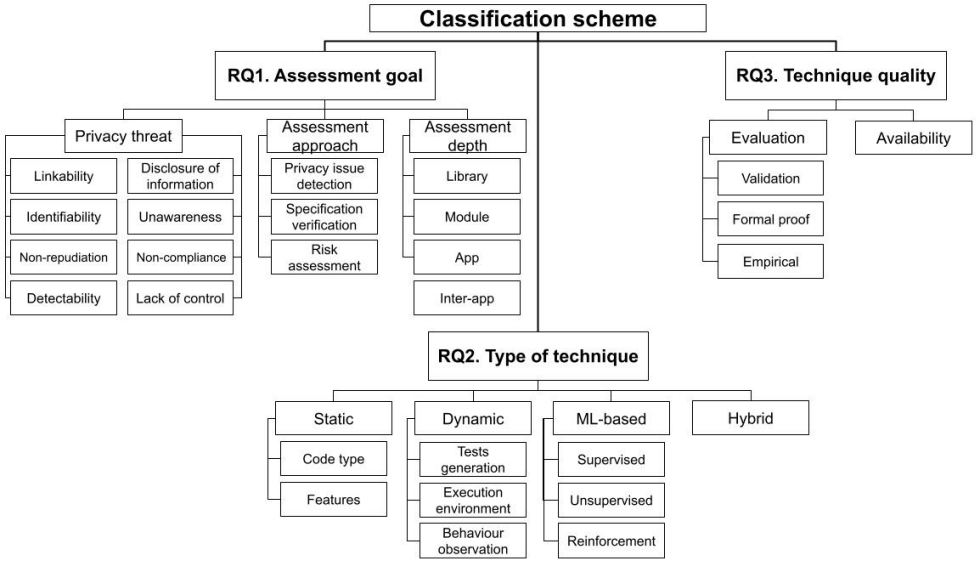

Figure 3. Classification scheme for coding the privacy assessment techniques.

After the pilot phase was over, we coded packages of eight papers. In each package, half of the papers were coded by two researchers of the team each, while the remaining four papers were coded by three researchers of the team each. This last set formed the control group, used to calculate again the Krippendorff's alpha for each coding package. Besides, daily meetings were performed during this phase to check the differences in the codification and to reach an agreement. The Krippendorff's values obtained for each criteria were in the optimal range (above 0.67).

4. Results

This section presents the results obtained from the analysis of all the papers collected in this study in response to our research questions.

4.1. RQ1: What Privacy Issues Have Been Assessed in Android Apps?

We start by analysing the 86 techniques reported in the 79 papers analysed by the assessment approach they have followed, namely, privacy issues detection (Table 5), privacy risk assessment (Table 6), or privacy specification verification (Table 7). For each approach we have identified the privacy threat targeted by the assessment technique (disclosure of information, non-compliance, or identifiability) and the depth of the assessment (library, app module, whole app, or inter-app). The number of techniques is greater than the number of papers as some papers reported different assessment techniques, e.g., when targeting more than one privacy threat or assessing privacy at several levels of depth.

We did not find any technique addressing many of the privacy threats we were looking for, namely linkability, non-repudiation, detectability, unawareness, and lack of control. These kinds of threats are often reported in relation to datasets, protocols, cryptographic algorithms, or authentication mechanisms but not in the case of consumer-oriented software systems. We found techniques assessing these threats in the screening phase, but we dismissed them as they did not meet our inclusion criteria.

Furthermore, we did not find any technique explicitly addressing privacy in Android modules. This new app publishing format was recently announced and may have a great impact in the assessment techniques. We discuss these aspects in detail in the next section.

Table 5. Distribution of contributions of the assessment approach: Privacy Issue Detection.

	Disclosure of Information	Non-Compliance	Identifiability
Library	2	0	0
Module	0	0	0
Application	51	0	0
Inter-Application	10	0	0
Total	63	0	0

Table 6. Distribution of contributions of the assessment approach: Risk Assessment.

	Disclosure of Information	Non-Compliance	Identifiability
Library	2	0	0
Module	0	0	0
Application	13	0	1
Inter-Application	0	0	0
Total	15	0	1

Table 7. Distribution of contributions of the assessment approach: Specification Verification.

	Disclosure of Information	Non-Compliance	Identifiability
Library	0	0	0
Module	0	0	0
Application	0	7	0
Inter-Application	0	0	0
Total	0	7	0

The vast majority of the techniques reported (73.26%, $n = 63$) were aimed at detecting specific privacy problems in mobile applications. In fact, all of them focus on detecting the disclosure of personal information, although at different levels of detail. The main focus is on detecting privacy issues at the application level by analysing data flows, regardless of whether these flows come from the application code itself or from any third-party library, e.g., by detecting data leakages through network traffic (ID179, ID361). A rather smaller amount of techniques focuses on analysing inter-application data leaks, e.g., due to collusive applications (ID94, ID294). Finally, only two techniques (ID362, ID452) focus on the detection of data leaks in libraries.

A smaller percentage (18.60%, $n = 16$) of techniques focus on assessing the risk to the users' privacy posed by an application. The risk score is then applied to sign potentially malicious apps (ID589, ID669) or make recommendations on alternative privacy-friendly apps to users (ID635).

Again, the main focus of the risk assessments is on personal data leaks. However, in this case we have found some papers further analysing these disclosures under the principle of data minimisation. For example, Zhang et al. (ID589) compared the execution of an application under different privacy settings claiming that if there is no perceived change when access to some data is granted/denied then the application did not actually need that piece of data, thus violating the principle of data minimisation and incurring a higher risk to the user. In another example, Oltroggle et. al. (ID846) compared the set of permissions requested and actually used by an app to detect over-privileged apps posing greater risks to their users.

We found one single paper focused on assessing identifiability risks (ID495). The authors collected the permissions granted to the application and computed a risk score considering the level of identification of the personal data obtained and the impact these may have in the users' privacy.

Finally, the remaining techniques (8.14%, $n = 7$) verify the app behaviour against a privacy specification and the non-compliance aspects. To this end, they use as a reference the app privacy policy, a specific data protection or privacy regulation, or both. For example, Yu et al. (ID281) looked for inconsistencies between the privacy policy and the permissions requested. We have found several laws being used as a reference such as the European GDPR (ID635), the USA COPPA (ID363), the privacy laws of California and Delaware (ID94), and even the old European Union Data Protection Directive (ID501) currently superseded by the GDPR.

The app manifest file is widely exploited as it encodes a declarative permission model that is significant for privacy assessment. Some techniques rely on the Android protection levels to analyse the declaration of "dangerous permissions" in the Manifest and gain an overall idea of the app privacy risk (ID272, ID281). Other techniques leverage upon the permissions declared in the Manifest as an input to detect whether an application is over-privileged, i.e., request more permissions than necessary (ID636, ID955). Other configuration files, such as the layout file, have also been used in the app assessment but to a lesser extent. This file defines the structure of an application's graphical user interface (GUI) declaring, e.g., the GUI elements that can be interacted with, and thus providing a valuable input for driving the app execution based on the extracted model.

We have also found techniques using the app metadata available in the app store including the app category, app description, or users' reviews. An app category is handy to find and classify the apps, particularly for those techniques that focus on specific privacy-sensitive categories, e.g., apps oriented to children. The users' reviews have been used to understand the privacy-to-functionality trade-off, e.g., by comparing the privacy-oriented reviews with those commenting on functionality (ID365). Some authors even leverage the privacy-related reviews as an input to calculate an app's privacy risks (ID635).

4.2. RQ2: What Techniques Have Been Used to Assess Apps' Privacy?

Static, dynamic, machine learning, and hybrid techniques have been applied in a quite different way to assess the privacy aspects described before. Nearly half of the papers use a combination of techniques instead of relying on just one of them, as shown in Figure 4. Pipelining the techniques is the preferred way of combining them and in Figure 5 we see how the techniques are preferred to be combined.

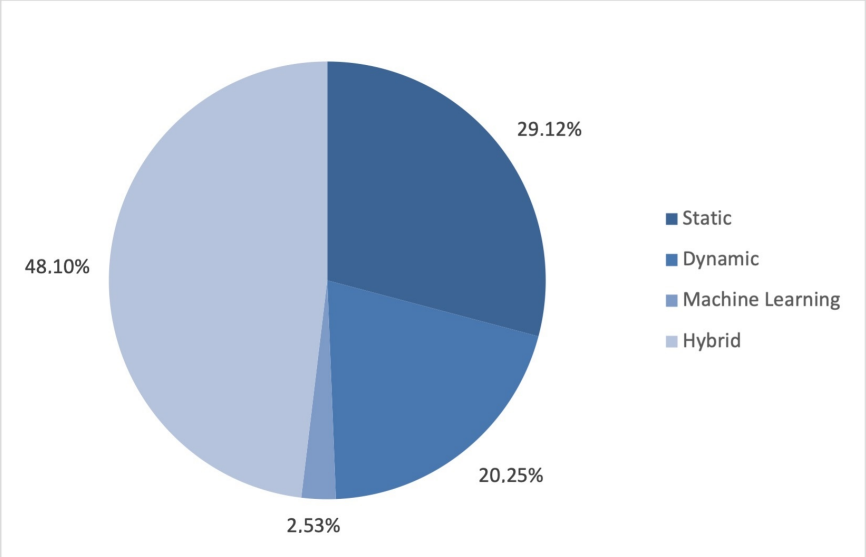

Figure 4. Privacy assessment techniques.

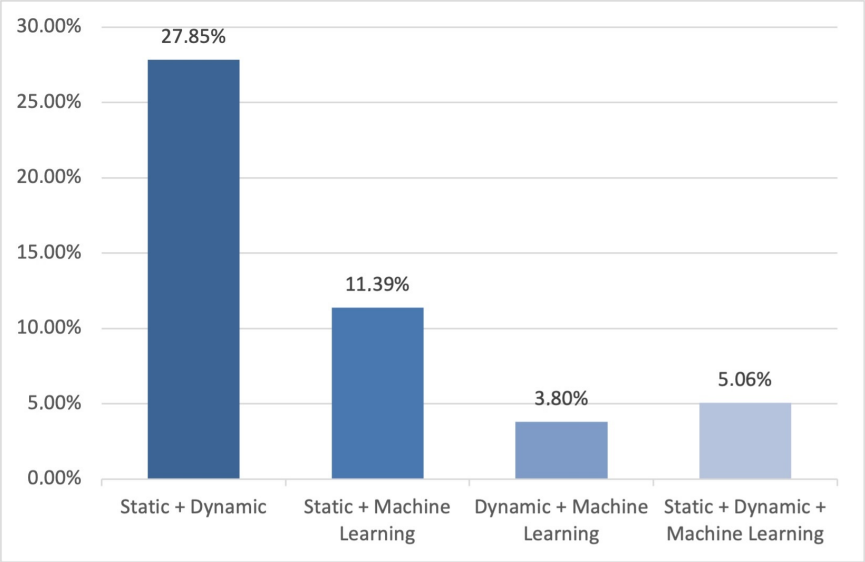

Figure 5. Hybrid assessment techniques.

Next, we further detail the different assessment techniques found and their combinations, the goals reached by each one, and the most outstanding papers in each case.

4.2.1. Static Analysis

Static analysis is carried out over different types of code, presented in the background Section 2, yet most of the papers report the analysis of some sort of intermediate code (Figure 6). Smali (https://github.com/JesusFreke/smali/wiki (accessed on 16 August 2021)) is the preferred option for intermediate code (35.9%). It provides a human-readable representation of Android binary code and can be easily obtained from an APK with tools like Apktool (https://ibotpeaches.github.io/Apktool/ (accessed on 16 August 2021)). Java bytecode is another relevant option (25.6%) but with an important trade-off: while there are tools getting the Java bytecode from Android binaries (e.g., DARE [27]) and assessing it (e.g., Soot framework [28]), reversing the process is daunting and, e.g., "problems often arise in retrieving variables names and types, among others." [29]. Finally, Jimple is also used in some cases (17.9%) as it delivers a simpler representation. Dexpler [30] obtains a Jimple representation from Android Dalvik Bytecode.

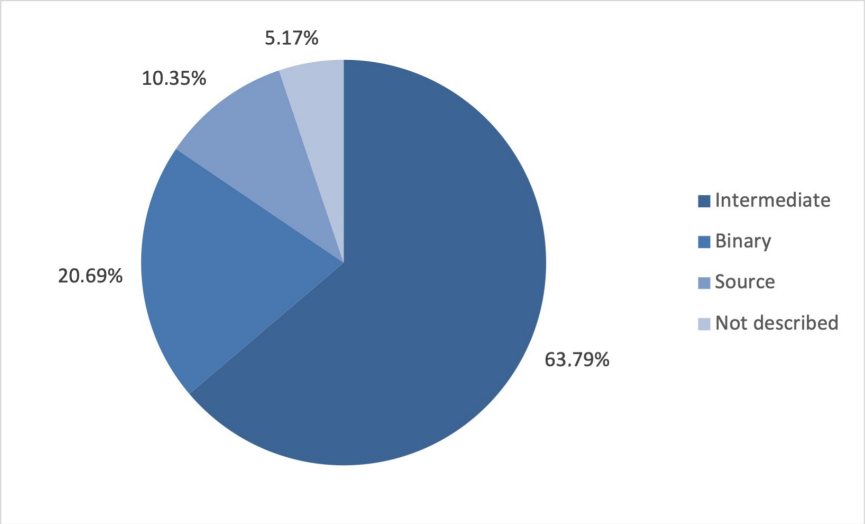

Figure 6. Type of app representation assessed by static analysis.

Previous surveys in the domain [6] identified a set of challenges for the static analysis of Android apps. Our results show that research has paid attention to some of these challenges, particularly regarding dynamically loaded code (DLC), inter-component communication (ICC), native code, and obfuscated code.

DLC is a mechanism by which applications execute code that was not included in the original APK file. This is handy to update an app over time with new features but can also become a privacy threat if it is used to load malicious code. DLC is a significant challenge to static analysis techniques as it takes place during the app execution, and thus the new code can be unknown to the static analyser. It is oftentimes implemented by reflection APIs, which are increasingly introduced in Android apps. According to ID77: *"reflection usage is widespread in Android apps, with 87.6% of apps making reflective calls"*. Although it is often claimed that static analysis cannot detect DLC, paper ID77 presents a method to detect DLC when the loaded code has been included in the apk file, and papers ID352 and ID597 describe the tools DroidRista and Ripple respectively, both partially addressing this challenge.

Other authors combine static and dynamic techniques to improve the detection of reflective code. These types of techniques basically use the static analysis to create a behavioural model (ID362), a flow graph (ID484), or to instrumentate the app (ID104), and later they use dynamic analysis techniques to focus the privacy assessment based on the results of the static analysis.

Another challenging feature is ICC, which mediates the communication between Android components either within an app or between different apps. The Android platform manages the communication leading to "discontinuities in the statically extracted app models" [7], which may hide data leaks to static analysers. Some authors (ID240, ID281, ID352, ID666, ID829, ID893, ID908) have addressed this issue by leveraging upon available tools such as Epicc [31], IC3 [32], or IccTA [33]. Other proposals "modify the code to transform the inter-component communications into function calls" so that the data leakage paths are uncovered (ID329). It is worth mentioning IIFA (ID81), which deals with both ICC data leaks intra- and inter-apps, and whose authors claim to achieve a 100% precision and recall against well-known benchmarks such as DroidBench and ICC-Bench.

Although there are different techniques coping with ICC, the main issue they usually find is that "these analysis processes consume a lot of time and memory" as reported in ID829. As the average size of applications grows, the number of methods and calls to be analysed grows too, leading on occasions to unacceptable analysis times when it comes to large-scale evaluations. We call the attention again to IIFA, whose performance was evaluated with a set of 90 real world apps reporting an average time of analysis per app of 87.9 seconds, which makes it ready for deployment at scale.

Native code makes static analysis quite challenging as code analysers usually deal with a single programming language, for example, a Java static analyser is not able to support fragments of C or C++ code [7]. However, many apps use native code to improve their performance. The approaches found either detect the native code and rely on specific tools to analyse it or create bridges to track the flow of personal data among languages. As an example of the former approach, DyDroid (ID565) is a tool to detect native code in apps, identify the libraries loading the code, and perform a malware detection based on DroidNative. As for the latter, BridgeInspector (ID89) tracks the data flows in hybrid Java-JavaScript apps, leveraging TaintDroid (ID247) for the data flows in Java, observing the Java-JavaScript bridge communication, and further analysing data flows on the JavaScript layer. The authors report a 100% precision when evaluating BridgeInspector in a sample of 38 apps that they claim to "cover all possible cross-language data paths". To deal with native code some papers (ID382, ID484) make use of tools like IDA Proo [34], which is a disassembler used to translate machine code into a human readable assembly language.

Finally, obfuscation is a common practice to protect the code from being understood. However, malicious behaviours may be also hidden behind obfuscated code, which challenges the privacy assessment. Obfuscation techniques are always evolving, and thus

no paper has claimed to fully overcome this challenge. However, some papers have solved specific cases of obfuscation such as obfuscated tracking APIs (ID771), obfuscated network transmissions (ID382), or detection of inter-app data leaks in obfuscated code (ID294). Another approach consists of using deobfuscation tools such as DeGuar, like in paper ID281. Finally, the DyDroid authors (ID565) present a system to detect and classify obfuscation techniques.

All in all, we have detected a considerable number of papers (over 60%) that report combining static analysis with another type of technique to improve their results. The main reason researchers combine static analysis with other techniques is the high number of false positives reported by this analysis. Although the analysis of all the code and associated resources of an app achieves high coverage, it may well lead to false positives due to parts of the code not executing, and in many cases researchers use other techniques to confirm this false positive, as explained in (ID294). The most common combination in this case is a pipeline of static and dynamic analysis: a way to take advantage of the best features of each technique type.

4.2.2. Dynamic Analysis

Dynamic analysis is based on the execution of an app code and the observation of its behaviour. Three main aspects to consider are the means to generate the different execution paths, i.e., (i) test generation, (ii) the environment where the tests are executed, and (iii) how the app behaviour is observed.

Different techniques are used for test generation, but the ones that stand out are "Random generation" and "Tester inputs" (Figure 7). In fact, the choice of the technique for the test generation heavily depends on what behaviours the researcher favours, with some papers even applying multiple techniques. Android Monkey is the preferred tool for the generation of inputs. The tool has demonstrated a good code coverage when compared with other input generation tools [35]. Furthermore, being part of the Android framework makes it a robust and always updated alternative. This contrasts with many of the results we have found which are outdated and abandoned shortly after being published. However, Android Monkey is not able to deal with smart user inputs. As described by Xu et al. (ID249): *"We do not adopt automated testing (e.g., using Monkey) because such tools cannot handle user registration and login issues, and hence cannot trigger some spyware behaviours effectively"*.

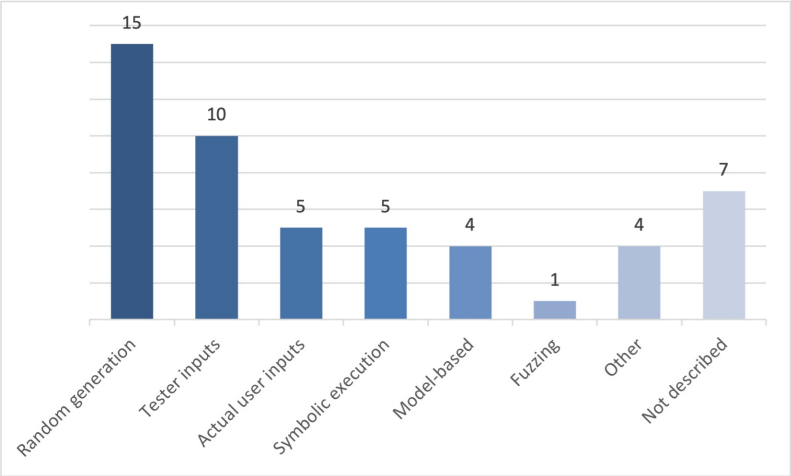

Figure 7. Types of test generation.

Regarding the execution environment, researchers slightly favour physical devices (55.3%) over emulators. Devices are expensive and sometimes they must be rooted to unlock relevant features, which can be difficult to emulate. On the other hand, current emulators are easy to configure, launch, and reuse although they do not have full capabilities. Some of the most commonly used emulators are GenyMotion, AndroidX86, and Android Emulator.

Finally, we have found four major approaches to observe an app behaviour, with important differences between them (Figure 8):

- Application instrumentation: this is one of the most complex techniques because it requires the modification of the app code to introduce code snippets that log the app status or behaviour at different points. This implies obtaining the application code, changing it, and repackaging it. This procedure heavily depends on the API version. As an advantage, this technique gives a greater insight on the app internals.
- Virtual Machine instrumentation: this technique requires an instrumented version of the Android Virtual Machine to observe the different API calls of the tested app. Unfortunately, this instrumentation is tight to the system version, thus making it very difficult to keep updated. The paradigmatic example was TaintDroid (ID247), which has been extensively used for Android apps assessment but is now deprecated.
- Kernel instrumentation: These tools observe the behaviour of the app at the kernel level. To this end, a modification to the operating system is required, generating again a great dependency on the system version. One of the most widespread tools was Xposed, which is now deprecated too. Some alternatives currently available are Frida or Magisk.
- Network interception: this technique is one of the easiest ways of behavioural observation. It does not require modifying the app code, can be used with both devices and emulators, and does not depend on the Android version. Tools such as tcpdump, Wireshark, or a Man In The Middle (MITM) Proxy can be used to monitor the network traffic. On the downside, the communication payload can be encoded or the channel secured, thus making the detection of data leaks cumbersome.

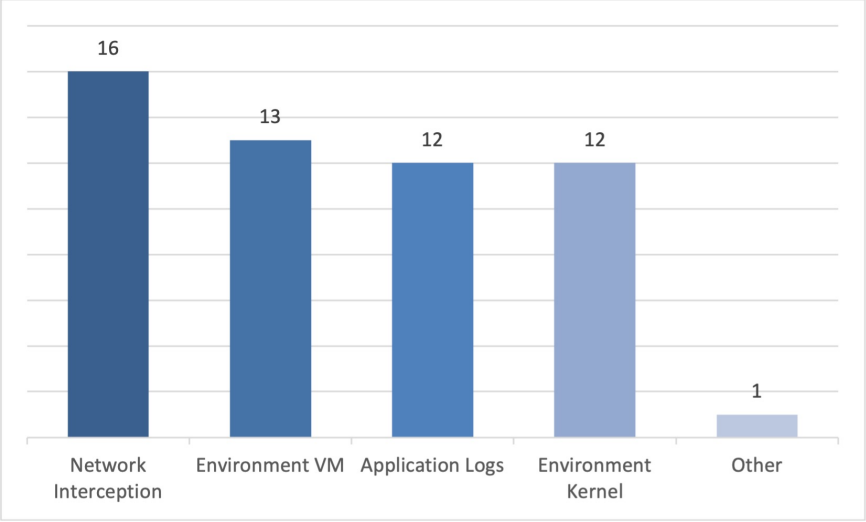

Figure 8. Types of behavioural observation.

4.2.3. Machine Learning-Based Analysis

Of all the coded papers, only 18 explore the use of machine learning to assess the privacy of Android apps. The features to train the models are usually extracted from static analysis, dynamic analysis, or both. Papers which only base their privacy assessment on features extracted from the app meta-data are an exception, e.g., paper ID342 uses the app reviews available in Google Play Store to train and validate a privacy risk model.

The use of machine learning brings important benefits to the privacy assessment of Android apps. First, machine learning can help to discover the most relevant features to focus the available resources on, avoiding useless efforts. For example, some studies (ID129, ID363, ID908) have discovered that the combination of simple features can be highly relevant for uncovering problematic apps, thus reducing the overall analysis time while keeping the effectiveness.

In our research, nearly all papers used supervised learning algorithms. Furthermore, all these algorithms were classifiers. This fits well with the nature of the object of analysis, since the purpose is to create a model that predicts if an application has or has not privacy issues, which is indeed a classification problem. The distribution of the different supervised algorithms is described in Figure 9. It is worth noting that the authors usually test different algorithms and finally choose to apply the one showing the best performance. Among them, Support Vector Machine has demonstrated a better performance.

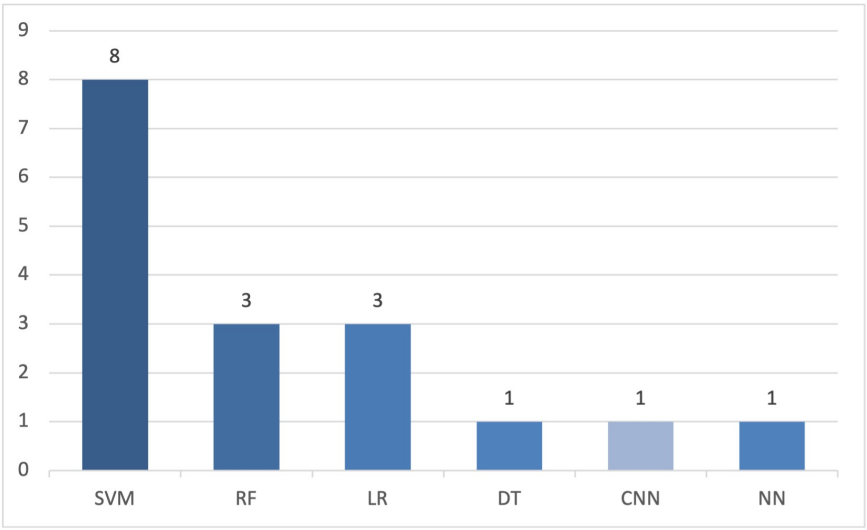

Figure 9. Supervised algorithms used to uncover privacy issues: Support Vector Machines (SVM), Random Forest (RF), Logistic Regression (LR), Decision Trees (DT), Convolutional Neural Networks (CNN), Nearest Neighbours (NN).

The major hurdle of supervised classification algorithms is the need of a ground truth, i.e., a reference dataset where each element is labelled. The creation of a good dataset can be one of the most time-consuming tasks, and an incomplete dataset can lead to a poor model. While some papers create their own dataset, others make use of publicly available ones. Among the latter, there are mainly three types: the datasets of labeled privacy policies, the datasets of malware samples, and the datasets of labeled apps based on their previous analysis. Table 8 presents a compilation of the public datasets still available and their main features.

Table 8. Public datasets labelled for the privacy assessment of Android apps.

Name	Description	Samples
OPP-115 (https://usableprivacy.org/data)	Website privacy policies annotated with privacy practices	115
APP-350 (https://usableprivacy.org/data)	Android app privacy policies annotated with privacy practices	350
Androsec (http://androsec.rit.edu/home)	Dataset of Android applications including details on quality metrics and security characteristics of the applications such as adherence to coding standards, file size, lines of code, over and under permissions, and any defects or security vulnerabilities that may exist	1179
Drebin (https://www.sec.cs.tu-bs.de/~danarp/drebin/)	Malware repository including apps labelled with malware family, and other features extracted from the applications	5560
VirusShare (https://virusshare.com/	Repository of malware samples for cybersecurity research	36,643,433 *

* This set can include multiple versions of the same application.

4.3. RQ3: What Is the Quality Level of the Techniques Identified?

All the papers analysed have either validated or evaluated their contributions (Figure 10). We intentionally excluded papers not showing efforts to assess their contributions. We did not find any paper carrying out formal proofs, which is reasonable since this would require having a formal model of apps unknown to the evaluator.

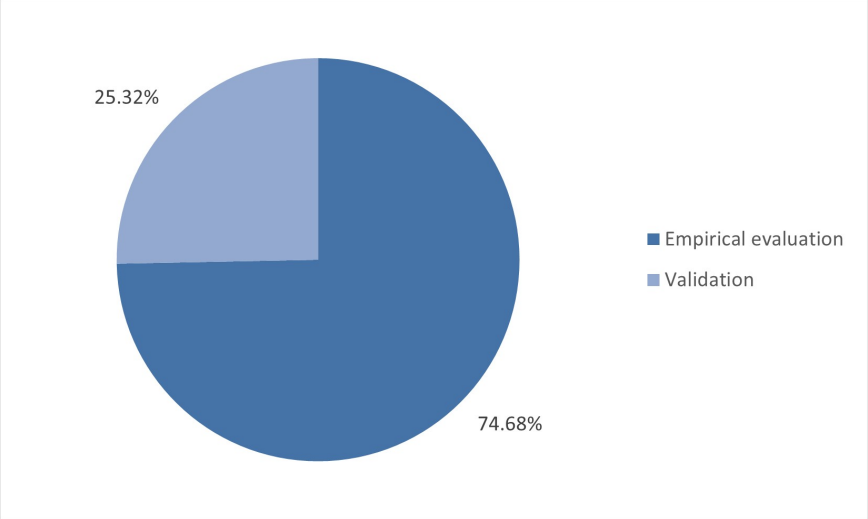

Figure 10. Type of assessment carried out on the technique by its authors.

Validations provide less information on the quality of the technique, as these basically demonstrate the feasibility of the proposal, but neither its effectiveness nor its performance is discussed. Validations are usually related to large-scale demonstration of the technique in the wild, where the newly created technique is used to assess the privacy features of a large set of apps. We have found techniques applied to as much as one million apps (ID94, ID196).

Effectiveness is the main goal pursued by empirical evaluations (Figure 11), i.e., understanding how well the technique assesses privacy aspects. Approximately 70% of the papers use standard metrics to describe their effectiveness such as precision, accuracy, recall, or F-score. This also means that the remaining 30% do not follow standard practices, which hinder comparing these tools with others. Time of analysis is the most reported metric of performance, although we have also found measures of CPU usage or overhead.

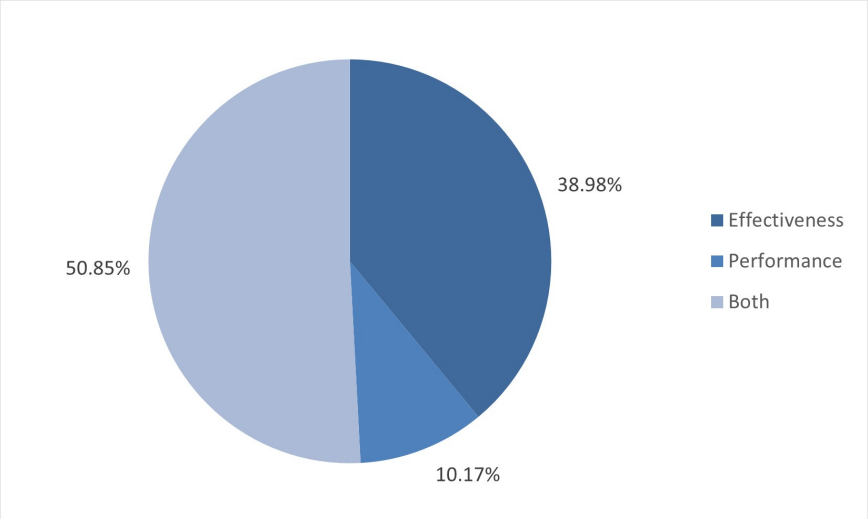

Figure 11. Goal pursued by empirical evaluations.

Authors have used different application sets to carry out their empirical evaluations (Figure 12). For example, they develop their own apps from scratch or adapt those already available to show a privacy behaviour their techniques are challenged to detect. These custom sets are usually reduced, including tens of applications at best. Sometimes, a benchmark is already available containing apps with known behaviour, thus saving researchers a great amount of time. Table 9 summarises the most relevant benchmarks available for evaluating privacy assessment techniques in Android apps. Finally, in a few cases researchers use actual apps available in different stores, which need to be reverse engineered to confirm the technique verdict. Real applications are usually downloaded from Google Play Store as there are REST APIs available to this end. We have also found other markets being used such as Baidu, Wandoujia, Anzhi, or QQ.

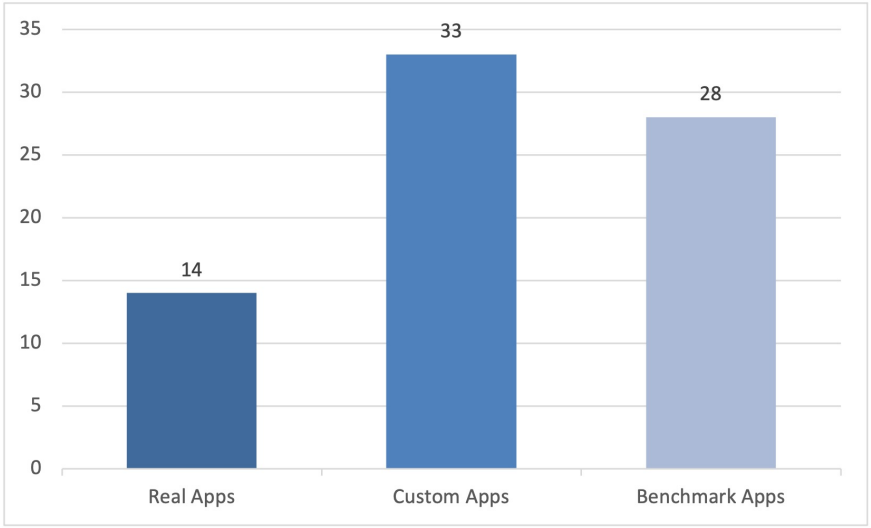

Figure 12. Types of app sets used for empirical evaluation.

Only 10 out of the 79 papers analysed (13%) have publicly released their tools (Table 10). This result strongly suggests that nowadays, although there have been sustained research efforts in privacy assessment techniques, there is very limited dissemination of these efforts to the scientific and more importantly the practitioner community. Figure 13 shows the tools used for comparison more often and the techniques comparing them.

Table 9. Reference benchmarks for evaluating privacy assessment techniques in Android.

Name	Description	Samples	Last Updated
Droidbench (https://github.com/secure-software-engineering/DroidBench) 2.0/3.0	Apps covering several types of data leaks	120 (2.0); 146 (3.0)	2016
Icc bench (https://github.com/fgwei/ICC-Bench)	Apps with inter-component data leaks	24	2017
F-droid (https://f-droid.org/en/packages/)	General apps (no privacy-related)	3696	2020
Mudflow (https://www.st.cs.uni-saarland.de/appmining/mudflow/)	Apps with abnormal personal data use	2866	2018
Drebin (https://www.sec.cs.tu-bs.de/~danarp/drebin/)	Malware apps	5560	2014
VirusShare (https://virusshare.com/)	Malware apps	36,643,433	2020
Contagio Mobile (http://contagiominidump.blogspot.com/)	Malware apps	131	2018
MobSecLab (http://mobseclab.gazi.edu.tr/datasets/)	Bening apps	1074	2017

Table 10. Contributions that released the developed tools.

Contribution	Open Source	Technique Type	Compared
SpyAware (ID249)	No	Dynamic + ML based	No
SoProtector (ID484)	No	Static + Dynamic	Yes
Agrigento (ID38)	Yes	Dynamic	Yes
DroidRA (ID77)	Yes	Static	Yes
NDroid (ID108)	Yes	Dynamic	Yes
SCDFLOW (ID126)	Yes	Static + ML based	Yes
Pluto (ID306)	Yes	Dynamic	No
HybriDroid (ID362)	Yes	Static	Yes
Reaper (ID842)	Yes	Dynamic	No
AppWalker (ID893)	Yes	Static + Dynamic	Yes

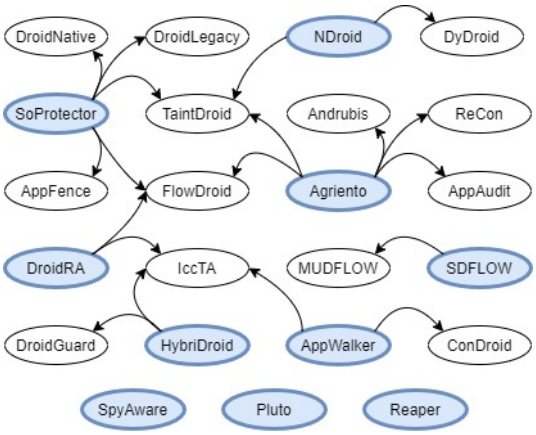

Figure 13. Publicly available tools found in our study (blue) and other tools to which they compare (white).

5. Discussion

In this section we further elaborate on the most relevant trends observed.

5.1. A Switch in Privacy Paradigms

Based on the results of what aspects of privacy have been assessed (RQ1), one main observation is that research efforts in the last five years have focused on two privacy threats: disclosure of personal information (83%) and non-compliance (12.5%). These two groups of papers roughly correspond to two very distinct privacy paradigms: privacy as confidentiality and privacy as contextual integrity.

Privacy as confidentiality is based on the binary criterion that any disclosure of a piece of personal data is a violation of the individual's privacy [36]. That is, it assumes that there are no other actors with other interests or needs to collect pieces of personal data (e.g., an app provider's need to collect the location to provide its core service). For example, some of the techniques included in this category alert of a privacy leak when an app merely accesses pieces of personal data on the mobile device even when these are not necessarily sent off-device (e.g., ID81), others do so when the data indeed leaves the device but does not take into account who the recipient is (e.g., ID141). Thus, these techniques by themselves are useful for alerting about (potential) personal data leaks but fail when certain personal data flows are indeed expected in a particular context.

On the other hand, the idea that individuals do not always act in isolation but rather transact in a variety of social contexts is known as contextual integrity [37]. It argues that an individual's activities take place in differentiated social contexts, and each of these contexts has a different set of norms that balance the multiple interests. Such norms prescribe context-dependent appropriate or inappropriate flows of personal data. For example, in a location-based service, the individual's location needs to be collected by the app provider

to operate correctly and provide the service to the individual. In that particular context, the disclosure of the individual's location to the app provider is appropriate. However, disclosure of the individual's location to other parties not involved in the provision of the service (e.g., advertisers) is not appropriate and may lead to a privacy violation. We observed that some recent works rely on the apps' privacy policies (ID596), regulations (ID501), or both (ID94) as the source to extract the (in) appropriate personal data flows. They are then used to analyse their alignment with the personal data flows performed by apps and to detect privacy violations in case of misalignment.

We argue that the increasing and tightened privacy regulation of mobile apps in the past few years has led to a growing interest in techniques embraced by the contextual integrity approach. Both the European Union [38] and the United States [39] authorities have agreed on the obligations for app providers to inform users about their personal data collection and processing practices, e.g., in terms of a privacy policy. Following on these requirements, the Google Play Store mandates that any application with access to personal and sensitive information must provide a privacy policy. However, apps' compliance with privacy and data protection regulations still remains a hot research topic, as further elaborated in the next subsection.

5.2. From Policy Compliance to Legal Compliance

Researchers have addressed apps' privacy compliance following two main approaches. Some authors seek to check the app compliance with some specific privacy law. Other researchers focus on checking that the app behaviour actually complies with its privacy policy.

The first group checks whether the app meets some mandatory legal aspect. At the basic level, Zimmeck et al. checked whether a privacy policy was available (ID323), showing that as much as 48% of applications in Google Play do not even provide a policy link. Assuming a privacy policy is available, others apply natural language processing (NLP) and other machine learning techniques to the app policy to address specific data protection principles, for example, assessing whether the policy is easy to read and understand (ID365). The completeness of the privacy policy has been evaluated too (ID323), that is, whether it includes some important information such as changes over previous versions and where to find them, the data subjects' rights, the data collection practices, and the information about with whom and how personal data is shared. Advanced papers even touch on requirements for international transfers (ID501), i.e., assessing whether the policy properly declares to send personal data to third countries with different (and potentially incompatible) data protection laws.

The second group of papers compares the practices described in the policy with the app behaviour. For this purpose, the most convenient approach is a hybrid analysis, combining supervised machine learning techniques to understand the policy statements with either static or dynamic analysis for understanding the app behaviour and comparing both.

Supervised machine learning requires a dataset of annotated policies to train policy models. Generating such annotated dataset is a human-intensive process, thus some approaches focus on extending datasets already available [40]. Once the new dataset is ready, the Sklearn library is usually employed to generate and train the models. Once the models are trained and validated, they are applied to identify some data practices in apps policies, typically data access and sharing. These practices will be compared with the actual behaviour observed in the app.

For understanding the app behaviour, the researchers use static and/or dynamic techniques. In the simplest cases, they check the permissions granted to the app and map them to the personal data they protect (ID635), assuming that the apps will indeed collect the data. More sophisticated studies perform a call graph analysis to trace relevant data access and sharing (ID94). While the static analysis approaches may yield false positives, dynamic techniques are used to detect actual behaviours by executing the application, typically using Android Monkey, and collecting system and network logs (ID596).

Checking the consistency between the app policy and its behaviour is a first step but far from being enough for assessing app compliance. Thus, we posit that the applicable legislation must be considered in this process, to understand whether the data practices also meet them.

5.3. App Developers Need More Support

Privacy compliance assessment is a priority issue for application providers, as certain regulations carry significant financial penalties in case of privacy violations [41]. Although sometimes developers ignore what privacy risks are introduced in their applications by the libraries they use [13], testing or auditing mobile apps against legal requirements is challenging for developers as there is a need for tools available to test, verify, or audit apps, libraries, and services [42]. As shown in Section 4, there have been a significant number of contributions in recent years; only a few have made their tools available. This then remains a barrier for developers to leverage those advancements and perform an effective assessment of the privacy status of their applications.

We believe that the provision of privacy-as-a-service assessment, supported by a flexible architecture for assessing different privacy requirements, could be a viable alternative to support developers and partly break down this barrier. Zimmeck et al. (ID94) moves closer to this approach, although it only deals with the data collection practices.

5.4. Changes on Apps Delivery Practices

In 2018 Google introduced a new app publishing format for Android known as Android App Bundle (.aab) [14]. Basically, a developer submits to Google Play a package including all the app code and resources and delegates in Google Play the generation of an optimised APK upon a user request. Some of the advantages of using the app bundle approach include the automated generation and distribution of custom APKs fitting the device and users' needs and the dynamic delivery of new modules as needed.

Having custom apps makes the privacy assessment process challenging as it shall be tailored to each APK too. In fact, Han et al. (ID60) already assessed differences between free and paid versions of apps (ID60), discovering that only 32% of the paid apps exhibit the same data processing behaviours as their free counterpart. Aligned with these findings, Ren et al. (ID813) carried out a longitudinal study of data leaks across app versions finding that the privacy risks vary greatly with versions. As demonstrated by these results, conclusions from a given custom version assessment cannot be generalised to the others.

In addition, bundle apps can optionally include feature modules, which are not included in the first download of the app but requested and installed later on either on-demand or when certain requirements are met. For example, a feature module can include new code and resources for an additional language package or a fancy add-on. Dynamically loaded code is a known challenge for static analysis techniques (ID362). Although the state-of-the-art reports early advances on this challenge, mainly through the combination of static and dynamic techniques (ID77, ID352, ID597), they have not been validated in the wild yet but in controlled environments.

As exposed, these novelties pose new challenges for the assessment of apps' privacy, which are not being properly addressed by the state of the art yet. In fact, we have not been able to find any research focused on this new app publishing format, even though it will become mandatory from August 2021. Thus, future research should provide privacy assessment techniques targeting user- and device-specific apps while also addressing their dynamic nature.

6. Threats to Validity

This section presents the potential threats to validity of our study and the steps we have followed in order to mitigate them.

6.1. Construct Validity

One of the first threats that was presented was the definition of the scope of the study, specifically regarding the frontiers between privacy and security in mobile applications. Although privacy is a multidisciplinary concept, within computer science it is often considered a cybersecurity knowledge area [43]. As such, identifying malware can imply also identifying a privacy threat. On the other hand, not all malware seeks to compromise privacy. To clearly set the scope, we decided to exclude from our analysis those malware-related papers which were not targeting privacy-specific issues.

In the paper search phase, we dealt with two threats related to the completeness of the study: the database, which could not index all relevant papers of the target domain, and the search string, which may lead to the exclusion of relevant papers. Regarding the database, we used Scopus since it indexes the most important digital libraries used in our area of research, as mentioned in Section 3, Methodology. Furthermore, for the sake of completeness, we also retrieved papers from the DBLP database which indexes conference proceedings not hosted by Scopus. As for the search string, we made a great effort to define it thoroughly, using keywords from well-known standards, vocabularies, and taxonomies in the research field. Furthermore, the string creation was an iterative process, validated with a test set of relevant papers provided by a senior privacy engineering researcher.

Despite all the preventions and actions taken, admittedly our study could have some limitations regarding its coverage. The number of citations and the number of page criteria significantly reduced the number of papers included in the screening phase. In addition, we only used one search string and it is possible that some papers may be missing. All in all, we started with a total amount of 10,825 papers published in the last five years, and after analysing our results we consider that our study gathers all the actual knowledge and provides an overview of the state of the art regarding the assessment of privacy-related aspects in Android apps.

6.2. Internal Validity

As in any systematic literature study, we dealt with three internal threats presented due to individual researcher's bias, particularly in (i) resolving whether to include or exclude a paper, (ii) classifying them according to the scheme, and (iii) analysing the results and drawing the conclusions. Regarding (i), early in the screening procedure, the researchers carried out an iterative pilot for validating and normalising their understanding on the inclusion and exclusion criteria. Only after obtaining a 90% of success and a 0.77 value of the Krippendorff's alpha coefficient they proceeded to the main screening phase. Furthermore, in this phase every paper was screened by at least two researchers and disagreements were resolved by all the team.

As for the classification procedure, the researchers elaborated a complete and clear classification scheme. It was built using existing recognised classifications, as for the best-practices and guidelines set in [19]. Once more a pilot was carried out in order to ensure the understanding and convergence of classification criteria. After the inter-coder reliability was assured (Krippendorff's alpha coefficient greater than 0.75 for all criteria), the classification phase was carried out and, like in the previous phase, every paper was coded by at least two researchers and inconsistencies were discussed by the entire team.

Finally, for the analysis of the results, two of the researchers cleaned and presented the results to the team. In an iterative way, the team discussed the meaning and the relevant facts in the results until an agreement was reached. The final results and conclusions presented in this paper stem from consolidated agreements and not from individual thoughts and ideas. The detailed results obtained in the coding phase are available in the replication package in case any reader wants to reproduce or validate our work [24].

6.3. External Validity

The scope of this study is the set of techniques reported in the state of the art to assess privacy-related aspects in Android applications. Its external validity is further framed

by the context in which they are intended to operate, i.e., the assessment is performed in a (semi-)automated fashion on final versions of any Android app available. Thus, the results and conclusions obtained are only valid within this scope and context. Accordingly, our conclusions do not apply to techniques focused on detecting common malware rather than specific privacy-related malware, techniques based on fixing some privacy issues rather than assessing them, or techniques based on a manual evaluation of some group of applications.

The application of the techniques reported in this paper to other OSs is not straightforward either. Most of them cannot be reused due to programming language and OS differences, which would make Java-based static analysers and dynamic OS- and virtual machine-based techniques useless. The most promising techniques to be reused in other OSs are those leveraging machine learning models based on OS-agnostic features such as metadata available in the app market (e.g., reviews) or network traffic metadata (e.g., destination, flow patterns), as demonstrated e.g., by McIlroy et al. [44]. Finally, the techniques assessing apps' privacy policies can be easily reused by researchers targeting other OSs.

The techniques reported targeted different Android API versions, thus threatening the generalisation of their results to other versions. Specifically, we have found 10 different API versions reported in the papers, from API level 14 released in 2011 to API level 28 released in 2018. Current reports on Android version market share [45] show that these versions are in use by slightly over a half of Android devices but sharply decreasing. Whether or not a given technique is able to deal with newer API versions remains unclear and must be studied individually. Unfortunately, most of the papers analysed do not provide a way of replication, which may hinder this checking.

7. Related Work

As far as we know, there are no recent secondary studies focused on the intersection of the three domains of our scope (i.e., privacy, software quality, and Android applications). The related work is either outdated [6,7] or has a different scope, e.g., by focusing on security instead of privacy [9,10] or missing the overall set of quality assessment techniques we have covered [46,47]. We analyse these studies next.

There have been many secondary studies focused on specific techniques to assess quality in Android apps. For instance, Li et al. [6] reviewed the use of static analysis for the quality assessment of Android apps. In turn, Sadeghi et al. [7] focused on dynamic analysis techniques. While both studies provided a complete overview of quality assessment techniques and their challenges, they both reviewed papers published by 2016 and privacy was only slightly mentioned.

More recent works have provided an updated perspective on the different techniques, though their scope still differs from ours. For example, the literature review by Pan et al. [9] analysed Android malware detection through static analysis techniques. This is a very complete study providing an updated overview of static analysis techniques and its combination with machine learning models, yet it misses privacy issues. Something similar happens with the study by Garg et al. [10], as their review covers all the papers published from 2013 to 2020 in the field of security assessment in Android. They have classified security objectives, assessment techniques, code representations, tools, and frameworks used but again do not include techniques addressing specific privacy issues.

Some recent studies have paid due attention to privacy issues, yet in this case they have missed the focus on Android apps. For example, Guamβín et al. [46] carried out a systematic mapping study on privacy assessment techniques in information systems to provide an overall picture of the broad landscape of techniques available. In this study the authors slightly touched upon mobile apps (i.e., they identified only 17 out of the 79 papers we have analysed) but pointed out the relevance of further analysing this domain as apps "involve higher privacy risks due to their ubiquity, the vast quantity of the personal data they handle and their sensitive nature, as well as the amount of service providers involved in the processing".

Finally, Ebramini et al. [47] might be considered the closest work to ours, as they partially focused on privacy assessment techniques in Android apps and also detected a majority of works reporting techniques for information disclosure detection. Their systematic mapping study on mobile apps' privacy was published in 2019, analysing papers published from 2010 to 2018 in software engineering venues. We claim that our study is more comprehensive and exhaustive. First, we claim comprehensiveness because by considering only software engineering venues Ebramini et al. ignored relevant papers published, e.g., in top conferences and journals in Computers Security and Computer Networks such as NDSS (seven papers analysed) or PoPETS (four papers analysed). As a result they identified 85 papers and coded 54 papers (often not even getting to the citation threshold we set), while our research considered an initial set of 10,825 papers and finally analysed 79 papers. Second, we claim exhaustiveness because they worked on different privacy aspects, namely privacy policy analysis, privacy requirements specification, users' perception on privacy, and privacy leaks detection, from which only the first and the last categories are related to our study. However, unlike our papers on the specification verification category, the papers in their privacy policy category were rather limited to policy text analysis, missing the detection of policy-to-behaviour mismatches, which is only recommended as future work. Finally, their Privacy Leaks Detection category is well aligned with our Disclosure of Information Privacy Threat category including static, dynamic, and hybrid approaches, but even in this case they found only 33 papers and we found more than twice that and also included machine learning based approaches.

All in all, our study advances the state of the art since we have focused on privacy assessment for Android applications considering all possible techniques (static, dynamic, ML-based, and hybrid approaches) and extensively covering the papers published in the last five years in major conferences and journals. We think our study is complementary to the aforementioned studies and other similar ones, and in combination they provide a better and more complete view on the dimension of Android privacy assessment.

8. Conclusions

This systematic mapping study has presented an overview of the state-of-the-art techniques to assess privacy in Android apps. For researchers, we have provided an overview of research efforts to assess different privacy threats and the most successful techniques. We have further analysed research gaps and discussed promising trends, such as the focus switch towards new privacy paradigms, the need to assess the app compliance with the declared policy and the applicable legislation, and the need to support developers in this process. For testers, auditors, and data protection authorities we have identified the most mature techniques and investigated the quality of their results, highlighting those that could be closer to their application in practice.

Our next steps point towards addressing the new challenges introduced by recent changes in the Android apps delivery model, which will be mandatory from August 2021. To this end, we seek to combine different techniques, which is a promising path according to our findings, and we apply interdisciplinary knowledge to advance from mere policy compliance to true app compliance with privacy regulations.

Author Contributions: Conceptualisation, J.M.D.A.; Investigation, J.M.D.A., D.G., B.B., and A.D.; Supervision, J.M.D.A.; Writing—original draft, J.M.D.A., D.G., B.B., and A.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research has been partially supported by the CLIIP project (grant reference APOYO-JOVENES-QINIM8-72-PKGQ0J) funded by the Comunidad de Madrid and Universidad Politécnica de Madrid under the V-PRICIT research programme 'Apoyo a la realización de Proyectos de I+D para jóvenes investigadores UPM-CAM', and by the Escuela Politécnica Nacional in Ecuador.

Data Availability Statement: The data presented in this study are openly available in Mendeley Data at doi:10.17632/jr349zxzcg.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. The Full List of Examined Publications

Table A1. List of paper references.

ID1	Y. Nan, Z. Yang, X. Wang, Y. Zhang, D. Zhu, M. Yang, Finding Clues for Your Secrets: Semantics-Driven, Learning-Based Privacy Discovery in Mobile Apps, in: Proc. 2018 Netw. Distrib. Syst. Secur. Symp., Internet Society, Reston, VA, 2018. https://doi.org/10.14722/ndss.2018.23092.
ID3	Y. He, B. Hu, Z. Han, Dynamic privacy leakage analysis of android third-party libraries, in: Proc.—2018 1st Int. Conf. Data Intell. Secur. ICDIS 2018, Institute of Electrical and Electronics Engineers Inc., 2018: pp. 275–280. https://doi.org/10.1109/ICDIS.2018.00051.
ID8	A. Rahman, P. Pradhan, A. Partho, L. Williams, Predicting AndroApplication Security and Privacy Risk with Static Code Metrics, in: Proc.—2017 IEEE/ACM 4th Int. Conf. Mob. Softw. Eng. Syst. MOBILESoft 2017, Institute of Electrical and Electronics Engineers Inc., 2017: pp. 149–153. https://doi.org/10.1109/MOBILESoft.2017.14.
ID38	A. Continella, Y. Fratantonio, M. Lindorfer, A. Puccetti, A. Zand, C. Kruegel, G. Vigna, Obfuscation-Resilient Privacy Leak Detection for Mobile Apps Through Differential Analysis, in: Proc. 2017 Netw. Distrib. Syst. Secur. Symp., Internet Society, Reston, VA, 2017. https://doi.org/10.14722/ndss.2017.23465.
ID60	C. Han, I. Reyes, βü. Feal, J. Reardon, P. Wijesekera, N. Vallina-Rodriguez, A. Elazari, K.A. Bamberger, S. Egelman, The Price is (Not) Right: Comparing Privacy in Free and PaApps, Proc. Priv. Enhancing Technol. 2020 (2020) 222-242. https://doi.org/10.2478/popets-2020-0050.
ID77	L. Li, T.F. Bissyandé, D. Octeau, J. Klein, DroidRA: Taming reflection to support whole-program analysis of androapps, in: ISSTA 2016—Proc. 25th Int. Symp. Softw. Test. Anal., Association for Computing Machinery, Inc, 2016: pp. 318–329. https://doi.org/10.1145/2931037.2931044.
ID81	A. Tiwari, S. Groβf, C. Hammer, IIFA: Modular Inter-app Intent Information Flow Analysis of AndroApplications, in: Lect. Notes Inst. Comput. Sci. Soc. Telecommun. Eng. LNICST, Springer, 2019: pp. 335–349. https://doi.org/10.1007/978-3-030-37231-6_19.
ID88	V. Sihag, A. Swami, M. Vardhan, P. Singh, Signature based malicious behavior detection in android, in: Commun. Comput. Inf. Sci., Springer, 2020: pp. 251–262. https://doi.org/10.1007/978-981-15-6648-6_20.
ID89	J. Bai, W. Wang, Y. Qin, S. Zhang, J. Wang, Y. Pan, BridgeTaint: A Bi-Directional Dynamic Taint Tracking Method for JavaScript Bridges in AndroHybrApplications, IEEE Trans. Inf. Forensics Secur. 14 (2019) 677–692. https://doi.org/10.1109/TIFS.2018.2855650.
ID93	V. Jain, S. Bhandari, V. Laxmi, M.S. Gaur, M. Mosbah, SniffDroid: Detection of Inter-App Privacy Leaks in Android, in: 2017 IEEE Trust., IEEE, 2017: pp. 331–338. https://doi.org/10.1109/Trustcom/BigDataSE/ICESS.2017.255.
ID94	S. Zimmeck, P. Story, D. Smullen, A. Ravichander, Z. Wang, J. Reidenberg, N. Cameron Russell, N. Sadeh, MAPS: Scaling Privacy Compliance Analysis to a Million Apps, Proc. Priv. Enhancing Technol. 2019 (2019) 66–86. https://doi.org/10.2478/popets-2019-0037.
ID104	Y. Yang, W. Luo, Y. Pei, M. Pan, T. Zhang, Execution enhanced static detection of androprivacy leakage hidden by dynamic class loading, in: Proc. - Int. Comput. Softw. Appl. Conf., IEEE Computer Society, 2019: pp. 149-158. https://doi.org/10.1109/COMPSAC.2019.00029.

Table A1. *Cont.*

ID108	L. Xue, C. Qian, H. Zhou, X. Luo, Y. Zhou, Y. Shao, A.T.S. Chan, NDroid: Toward tracking information flows across multiple andro-contexts, IEEE Trans. Inf. Forensics Secur. 14 (2019) 814–828. https://doi.org/10.1109/TIFS.2018.2866347.
ID117	K. Zhu, X. He, B. Xiang, L. Zhang, A. Pattavina, How dangerous are your Smartphones? App usage recommendation with privacy preserving, Mob. Inf. Syst. 2016 (2016). https://doi.org/10.1155/2016/6804379.
ID119	G. Bai, Q. Ye, Y. Wu, H. Botha, J. Sun, Y. Liu, J.S. Dong, W. Visser, Towards Model Checking AndroApplications, IEEE Trans. Softw. Eng. 44 (2018) 595–612. https://doi.org/10.1109/TSE.2017.2697848.
ID126	P. Feng, J. Ma, C. Sun, Selecting Critical Data Flows in AndroApplications for Abnormal Behavior Detection, Mob. Inf. Syst. 2017 (2017). https://doi.org/10.1155/2017/7397812.
ID129	N. Wongwiwatchai, P. Pongkham, K. Sripanidkulchai, Detecting personally identifiable information transmission in androapplications using light-weight static analysis, Comput. Secur. 99 (2020) 102011. https://doi.org/10.1016/j.cose.2020.102011.
ID141	K. Alkhattabi, A. Alshehri, C. Yue, Security and Privacy Analysis of AndroFamily Locator Apps, in: Proc. 25th ACM Symp. Access Control Model. Technol., ACM, New York, NY, USA, 2020: pp. 47–58. https://doi.org/10.1145/3381991.3395612.
ID179	M. Backes, S. Bugiel, E. Derr, S. Gerling, C. Hammer, R-Droid: Leveraging androapp analysis with static slice optimization, in: ASIA CCS 2016—Proc. 11th ACM Asia Conf. Comput. Commun. Secur., Association for Computing Machinery, Inc, 2016: pp. 129–140. https://doi.org/10.1145/2897845.2897927.
ID196	R. Binns, U. Lyngs, M. Van Kleek, J. Zhao, T. Libert, N. Shadbolt, Third party tracking in the mobile ecosystem, in: WebSci 2018—Proc. 10th ACM Conf. Web Sci., Association for Computing Machinery, Inc, 2018: pp. 23–31. https://doi.org/10.1145/3201064.3201089.
ID224	M. Junaid, D. Liu, D. Kung, Dexteroid: Detecting malicious behaviors in Androapps using reverse-engineered life cycle models, Comput. Secur. 59 (2016) 92–117. https://doi.org/10.1016/j.cose.2016.01.008.
ID240	S. Bhandari, F. Herbreteau, V. Laxmi, A. Zemmari, P.S. Roop, M.S. Gaur, SneakLeak: Detecting Multipartite Leakage Paths in AndroApps, in: 2017 IEEE Trust., IEEE, 2017: pp. 285–292. https://doi.org/10.1109/Trustcom/BigDataSE/ICESS.2017.249.
ID247	W. Enck, P. Gilbert, S. Han, V. Tendulkar, B.G. Chun, L.P. Cox, J. Jung, P. McDaniel, A.N. Sheth, TaintDroid: An information-flow tracking system for realtime privacy monitoring on smartphones, ACM Trans. Comput. Syst. 32 (2014) 1–29. https://doi.org/10.1145/2619091.
ID249	H. Xu, Y. Zhou, C. Gao, Y. Kang, M.R. Lyu, SpyAware: Investigating the privacy leakage signatures in app execution traces, in: 2015 IEEE 26th Int. Symp. Softw. Reliab. Eng. ISSRE 2015, Institute of Electrical and Electronics Engineers Inc., 2016: pp. 348–358. https://doi.org/10.1109/ISSRE.2015.7381828.
ID272	S. Pooryousef, M. Amini, Enhancing accuracy of andromalware detection using intent instrumentation, in: ICISSP 2017—Proc. 3rd Int. Conf. Inf. Syst. Secur. Priv., SciTePress, 2017: pp. 380–388. https://doi.org/10.5220/0006195803800388.
ID281	L. Yu, X. Luo, C. Qian, S. Wang, H.K.N. Leung, Enhancing the Description-to-Behavior Fidelity in AndroApps with Privacy Policy, IEEE Trans. Softw. Eng. 44 (2018) 834–854. https://doi.org/10.1109/TSE.2017.2730198.

Table A1. *Cont.*

ID294	N.T. Cam, V.H. Pham, T. Nguyen, Detecting sensitive data leakage via inter-applications on Androusing a hybranalysis technique, Cluster Comput. 22 (2019) 1055–1064. https://doi.org/10.1007/s10586-017-1260-2.
ID302	V. Jain, V. Laxmi, M.S. Gaur, M. Mosbah, APPLADroid: Automaton based inter-app privacy leak analysis for android, in: Commun. Comput. Inf. Sci., Springer Verlag, 2019: pp. 219–233. https://doi.org/10.1007/978-981-13-7561-3_16.
ID306	S. Demetriou, W. Merrill, W. Yang, A. Zhang, C.A. Gunter, Free for All! Assessing User Data Exposure to Advertising Libraries on Android, in: Proc. 2016 Netw. Distrib. Syst. Secur. Symp., Internet Society, Reston, VA, 2016. https://doi.org/10.14722/ndss.2016.23082.
ID323	S. Zimmeck, Z. Wang, L. Zou, R. Iyengar, B. Liu, F. Schaub, S. Wilson, N. Sadeh, S.M. Bellovin, J. Reidenberg, Automated analysis of privacy requirements for mobile apps, in: AAAI Fall Symp.—Tech. Rep., AI Access Foundation, 2016: pp. 286–296. https://doi.org/10.14722/ndss.2017.23034.
ID329	H. Cui, S. Shao, S. Niu, W. Zhang, Y. Yuan, Container-based privacy preserving scheme for androapplications, Chinese J. Electron. 29 (2020) 731–737. https://doi.org/10.1049/cje.2020.06.001.
ID342	M. Hatamian, J. Serna, K. Rannenberg, Revealing the unrevealed: Mining smartphone users privacy perception on app markets, Comput. Secur. 83 (2019) 332–353. https://doi.org/10.1016/j.cose.2019.02.010.
ID349	Y. Li, G. Xu, H. Xian, L. Rao, J. Shi, Novel andromalware detection method based on multi-dimensional hybrfeatures extraction and analysis, Intell. Autom. Soft Comput. 25 (2019) 637–647. https://doi.org/10.31209/2019.100000118.
ID352	A. Alzaidi, S. Alshehri, S.M. Buhari, DroidRista: a highly precise static data flow analysis framework for androapplications, Int. J. Inf. Secur. 19 (2020) 523–536. https://doi.org/10.1007/s10207-019-00471-w.
ID361	J. Qiu, J. Zhang, W. Luo, L. Pan, S. Nepal, Y. Wang, Y. Xiang, A3CM: Automatic Capability Annotation for AndroMalware, IEEE Access. 7 (2019) 147156–147168. https://doi.org/10.1109/ACCESS.2019.2946392.
ID362	H. Chen, H.F. Leung, B. Han, J. Su, Automatic privacy leakage detection for massive androapps via a novel hybrapproach, in: IEEE Int. Conf. Commun., Institute of Electrical and Electronics Engineers Inc., 2017. https://doi.org/10.1109/ICC.2017.7996335.
ID363	K. Basu, S.S. Hussain, U. Gupta, R. Karri, COPPTCHA: COPPA Tracking by Checking Hardware-Level Activity, IEEE Trans. Inf. Forensics Secur. 15 (2020) 3213-3226. https://doi.org/10.1109/TIFS.2020.2983287.
ID365	J. Wettlaufer, H. Simo, Decision support for mobile app selection via automated privacy assessment, in: IFIP Adv. Inf. Commun. Technol., Springer, 2020: pp. 292–307. https://doi.org/10.1007/978-3-030-42504-3_19.
ID382	J. Reardon, βü. Feal, P. Wijesekera, A.E.B. On, N. Vallina-Rodriguez, S. Egelman, 50 Ways to Leak Your Data: An Exploration of Apps' Circumvention of the AndroPermissions System, in: 28th USENIX Secur. Symp. (USENIX Secur. 19), USENIX Association, Santa Clara, CA, 2019: pp. 603–620. https://www.usenix.org/conference/usenixsecurity19/presentation/reardon.
ID384	M. Hatamian, J. Serna, K. Rannenberg, B. Igler, FAIR: Fuzzy alarming index rule for privacy analysis in smartphone apps, in: Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), Springer Verlag, 2017: pp. 3–18. https://doi.org/10.1007/978-3-319-64483-7_1.

Table A1. *Cont.*

ID412	G. Ascia, V. Catania, R. Di Natale, A. Fornaia, M. Mongiovi, S. Monteleone, G. Pappalardo, E. Tramontana, Making androapps data-leak-safe by data flow analysis and code injection, in: Proc.—25th IEEE Int. Conf. Enabling Technol. Infrastruct. Collab. Enterp. WETICE 2016, Institute of Electrical and Electronics Engineers Inc., 2016: pp. 205–210. https://doi.org/10.1109/WETICE.2016.53.
ID452	L.H. Tuan, N.T. Cam, V.H. Pham, Enhancing the accuracy of static analysis for detecting sensitive data leakage in Androby using dynamic analysis, Cluster Comput. 22 (2019) 1079-1085. https://doi.org/10.1007/s10586-017-1364-8.
ID484	G. Xu, W. Wang, L. Jiao, X. Li, K. Liang, X. Zheng, W. Lian, H. Xian, H. Gao, SoProtector: Safeguard Privacy for Native SO Files in Evolving Mobile IoT Applications, IEEE Internet Things J. 7 (2020) 2539–2552. https://doi.org/10.1109/JIOT.2019.2944006.
ID495	A. Hamed, H.K. Ben Ayed, Privacy risk assessment and users' awareness for mobile apps permissions, in: Proc. IEEE/ACS Int. Conf. Comput. Syst. Appl. AICCSA, IEEE Computer Society, 2016. https://doi.org/10.1109/AICCSA.2016.7945694.
ID501	M. Eskandari, B. Kessler, M. Ahmad, A.S. de Oliveira, B. Crispo, Analysing Remote Server Locations for Personal Data Transfers in Mobile Apps, Proc. Priv. Enhancing Technol. 2017 (2016) 118–131. https://doi.org/10.1515/popets-2017-0008.
ID539	M. Sun, T. Wei, J.C.S. Lui, TaintART: A practical multi-level information-flow tracking system for AndroRunTime, in: Proc. ACM Conf. Comput. Commun. Secur., Association for Computing Machinery, 2016: pp. 331–342. https://doi.org/10.1145/2976749.2978343.
ID541	T. Watanabe, M. Akiyama, T. Sakai, T. Mori, Understanding the Inconsistencies between Text Descriptions and the Use of Privacy-sensitive Resources of Mobile Apps, in: Elev. Symp. Usable Priv. Secur. (SOUPS 2015), USENIX Association, Ottawa, 2015: pp. 241–255. https://www.usenix.org/conference/soups2015/proceedings/presentation/watanabe.
ID565	Z. Qu, S. Alam, Y. Chen, X. Zhou, W. Hong, R. Riley, DyDroid: Measuring Dynamic Code Loading and Its Security Implications in AndroApplications, in: Proc.—47th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Networks, DSN 2017, Institute of Electrical and Electronics Engineers Inc., 2017: pp. 415–426. https://doi.org/10.1109/DSN.2017.14.
ID589	L.L. Zhang, C.J.M. Liang, Z.L. Li, Y. Liu, F. Zhao, E.H. Chen, Characterizing Privacy Risks of Mobile Apps with Sensitivity Analysis, IEEE Trans. Mob. Comput. 17 (2018) 279–292. https://doi.org/10.1109/TMC.2017.2708716.
ID593	Q. Qian, J. Cai, M. Xie, R. Zhang, Malicious Behavior Analysis for AndroApplications, 2016.
ID596	B. Andow, S.Y. Mahmud, J. Whitaker, W. Enck, B. Reaves, K. Singh, S. Egelman, Actions Speak Louder than Words: Entity-Sensitive Privacy Policy and Data Flow Analysis with PoliCheck, in: 29th USENIX Secur. Symp. (USENIX Secur. 20), USENIX Association, 2020: pp. 985–1002. https://www.usenix.org/conference/usenixsecurity20/presentation/andow.
ID597	Y. Zhang, T. Tan, Y. Li, J. Xue, Ripple: Reflection analysis for androapps in incomplete information environments, in: CODASPY 2017—Proc. 7th ACM Conf. Data Appl. Secur. Priv., Association for Computing Machinery, Inc, 2017: pp. 281–288. https://doi.org/10.1145/3029806.3029814.

Table A1. *Cont.*

ID611	N.W. Lo, K.H. Yeh, C.Y. Fan, Leakage Detection and Risk Assessment on Privacy for AndroApplications: LRPdroid, IEEE Syst. J. 10 (2016) 1361–1369. https://doi.org/10.1109/JSYST.2014.2364202.
ID626	W. Choi, J. Kannan, D. Babic, A scalable, flow-and-context-sensitive taint analysis of androapplications, J. Vis. Lang. Comput. 51 (2019) 1–14. https://doi.org/10.1016/j.jvlc.2018.10.005.
ID627	E.P. Papadopoulos, M. Diamantaris, P. Papadopoulos, T. Petsas, S. Ioannidis, E.P. Markatos, The long-standing privacy debate: Mobile websites Vs mobile apps, in: 26th Int. World Wide Web Conf. WWW 2017, International World Wide Web Conferences Steering Committee, 2017: pp. 153–162. https://doi.org/10.1145/3038912.3052691.
ID635	M. Hatamian, N. Momen, L. Fritsch, K. Rannenberg, A Multilateral Privacy Impact Analysis Method for AndroApps, in: Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), Springer Verlag, 2019: pp. 87–106. https://doi.org/10.1007/978-3-030-21752-5_7.
ID636	S. Wu, P. Wang, X. Li, Y. Zhang, Effective detection of andromalware based on the usage of data flow APIs and machine learning, Inf. Softw. Technol. 75 (2016) 17–25. https://doi.org/10.1016/j.infsof.2016.03.004.
ID657	S. Lee, J. Dolby, S. Ryu, Hybridroid: Static analysis framework for androhybrapplications, in: ASE 2016—Proc. 31st IEEE/ACM Int. Conf. Autom. Softw. Eng., Association for Computing Machinery, Inc, New York, NY, USA, 2016: pp. 250-261. https://doi.org/10.1145/2970276.2970368.
ID658	J. Malik, R. Kaushal, CREDROid: Andromalware detection by network traffic analysis, in: PAMCO 2016—Proc. 2nd MobiHoc Int. Work. Privacy-Aware Mob. Comput., Association for Computing Machinery, Inc, 2016: pp. 28–36. https://doi.org/10.1145/2940343.2940348.
ID666	S. Bhandari, F. Herbreteau, V. Laxmi, A. Zemmari, M.S. Gaur, P.S. Roop, SneakLeak+: Large-scale klepto apps analysis, Futur. Gener. Comput. Syst. 109 (2020) 593–603. https://doi.org/10.1016/j.future.2018.05.047.
ID669	R.S. Arslan, I.A. Doğru, N. Barişçi, Permission-Based Malware Detection System for AndroUsing Machine Learning Techniques, Int. J. Softw. Eng. Knowl. Eng. 29 (2019) 43–61. https://doi.org/10.1142/S0218194019500037.
ID671	J. Xie, X. Fu, X. Du, B. Luo, M. Guizani, AutoPatchDroid: A framework for patching inter-app vulnerabilities in androapplication, in: IEEE Int. Conf. Commun., Institute of Electrical and Electronics Engineers Inc., 2017. https://doi.org/10.1109/ICC.2017.7996682.
ID692	B. Buddhadev, P. Faruki, M.S. Gaur, S. Kharche, A. Zemmari, FloVasion: Towards Detection of non-sensitive Variable Based Evasive Information-Flow in AndroApps, IETE J. Res. (2020). https://doi.org/10.1080/03772063.2020.1721338.
ID704	Z. Cheng, X. Chen, Y. Zhang, S. Li, J. Xu, MUI-defender: CNN-driven, network flow-based information theft detection for mobile users, in: Lect. Notes Inst. Comput. Sci. Soc. Telecommun. Eng. LNICST, Springer Verlag, 2019: pp. 329–345. https://doi.org/10.1007/978-3-030-12981-1_23.
ID706	E. Pan, J. Ren, M. Lindorfer, C. Wilson, D. Choffnes, Panoptispy: Characterizing Audio and Video Exfiltration from AndroApplications, Proc. Priv. Enhancing Technol. 2018 (2018) 33–50. https://doi.org/10.1515/popets-2018-0030.

Table A1. *Cont.*

ID730	J.C.J. Keng, L. Jiang, T.K. Wee, R.K. Balan, Graph-aided directed testing of androapplications for checking runtime privacy behaviours, in: Proc.—11th Int. Work. Autom. Softw. Test, AST 2016, Association for Computing Machinery, Inc, 2016: pp. 57–63. https://doi.org/10.1145/2896921.2896930.
ID771	X. Liu, J. Liu, S. Zhu, W. Wang, X. Zhang, Privacy risk analysis and mitigation of analytics libraries in the androecosystem, IEEE Trans. Mob. Comput. 19 (2020) 1184–1199. https://doi.org/10.1109/TMC.2019.2903186.
ID783	H. Fu, Z. Zheng, S. Bose, M. Bishop, P. Mohapatra, LeakSemantic: identifying abnormal sensitive network transmissions in mobile applications, in: Proc.—IEEE INFOCOM, Institute of Electrical and Electronics Engineers Inc., 2017. https://doi.org/10.1109/INFOCOM.2017.8057221.
ID813	J. Ren, M. Lindorfer, D.J. Dubois, A. Rao, D. Choffnes, N. Vallina-Rodriguez, Bug Fixes, Improvements, ... and Privacy Leaks—A Longitudinal Study of PII Leaks Across AndroApp Versions, in: Proc. 2018 Netw. Distrib. Syst. Secur. Symp., Internet Society, Reston, VA, 2018. https://doi.org/10.14722/ndss.2018.23143.
ID827	Y. He, L. Zhang, Z. Yang, Y. Cao, K. Lian, S. Li, W. Yang, Z. Zhang, M. Yang, Y. Zhang, H. Duan, TextExerciser: Feedback-driven text input exercising for androapplications, in: Proc.—IEEE Symp. Secur. Priv., Institute of Electrical and Electronics Engineers Inc., 2020: pp. 1071–1087. https://doi.org/10.1109/SP40000.2020.00071.
ID829	Z. Meng, Y. Xiong, W. Huang, L. Qin, X. Jin, H. Yan, AppScalpel: Combining static analysis and outlier detection to identify and prune undesirable usage of sensitive data in Androapplications, Neurocomputing. 341 (2019) 10–25. https://doi.org/10.1016/j.neucom.2019.01.105.
ID842	M. Diamantaris, E.P. Papadopoulos, E.P. Markatos, S. Ioannidis, J. Polakis, Reaper: Real-time app analysis for augmenting the andropermission system, in: CODASPY 2019—Proc. 9th ACM Conf. Data Appl. Secur. Priv., Association for Computing Machinery, Inc, 2019: pp. 37–48. https://doi.org/10.1145/3292006.3300027.
ID846	M. Oltrogge, E. Derr, C. Stransky, Y. Acar, S. Fahl, C. Rossow, G. Pellegrino, S. Bugiel, M. Backes, The Rise of the Citizen Developer: Assessing the Security Impact of Online App Generators, in: Proc.—IEEE Symp. Secur. Priv., Institute of Electrical and Electronics Engineers Inc., 2018: pp. 634–647. https://doi.org/10.1109/SP.2018.00005.
ID893	T. Wu, Y. Yang, Detecting AndroInter-App Data Leakage Via Compositional Concolic Walking, Intell. Autom. Soft Comput. (2019). https://doi.org/10.31209/2019.100000079.
ID902	M. Graa, N. Cuppens-Boulahia, F. Cuppens, J.L. Lanet, R. Moussaileb, Detection of side channel attacks based on data tainting in androsystems, in: IFIP Adv. Inf. Commun. Technol., Springer New York LLC, 2017: pp. 205–218. https://doi.org/10.1007/978-3-319-58469-0_14.
ID908	X. Pan, X. Wang, Y. Duan, X. Wang, H. Yin, Dark Hazard: Learning-based, Large-Scale Discovery of Hidden Sensitive Operations in AndroApps, in: Proc. 2017 Netw. Distrib. Syst. Secur. Symp., Internet Society, Reston, VA, 2017. https://doi.org/10.14722/ndss.2017.23265.
ID915	R. Salvia, P. Ferrara, F. Spoto, A. Cortesi, SDLI: Static Detection of Leaks Across Intents, in: Proc.—17th IEEE Int. Conf. Trust. Secur. Priv. Comput. Commun. 12th IEEE Int. Conf. Big Data Sci. Eng. Trust. 2018, Institute of Electrical and Electronics Engineers Inc., 2018: pp. 1002–1007. https://doi.org/10.1109/TrustCom/BigDataSE.2018.00141.

Table A1. *Cont.*

ID955	L. Yu, T. Zhang, X. Luo, L. Xue, H. Chang, Toward Automatically Generating Privacy Policy for AndroApps, IEEE Trans. Inf. Forensics Secur. 12 (2017) 865–880. https://doi.org/10.1109/TIFS.2016.2639339.
ID967	G. Barbon, A. Cortesi, P. Ferrara, E. Steffinlongo, DAPA: Degradation-aware privacy analysis of Androapps, in: Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), Springer Verlag, 2016: pp. 32–46. https://doi.org/10.1007/978-3-319-46598-2_3.

References

1. Privacy and Data Protection in Mobile Applications—ENISA. Available online: https://www.enisa.europa.eu/publications/privacy-and-data- (accessed on 16 August 2021).
2. Gamba, J.; Rashed, M.; Razaghpanah, A.; Tapiador, J.; Vallina-Rodriguez, N. An analysis of pre-installed android software. In Proceedings of the IEEE Symposium on Security and Privacy, San Francisco, CA, USA, 18–21 May 2020; pp. 1039–1055.
3. Balebako, R.; Marsh, A.; Lin, J.; Hong, J.; Faith Cranor, L. The Privacy and Security Behaviors of Smartphone App Developers. In Proceedings of the 2014 Workshop on Usable Security, Reston, VA, USA, 23 February 2014, doi:10.14722/usec.2014.23006.
4. IDC—Smartphone Market Share—OS. Available online: https://www.idc.com/promo/smartphone-market-share (accessed on 16 August 2021).
5. Zang, J.; Dummit, K.; Graves, J.; Lisker, P.; Sweeney, L. Who Knows What About Me? A Survey of Behind the Scenes Personal Data Sharing to Third Parties by Mobile Apps. *Technol. Sci.* **2015**, *30*, 1–53.
6. Li, L.; Bissyandé, T.F.; Papadakis, M.; Rasthofer, S.; Bartel, A.; Octeau, D.; Klein, J.; Traon, L. Static analysis of android apps: A systematic literature review. *Inf. Softw. Technol.* **2017**, *88*, 67–95, doi:10.1016/j.infsof.2017.04.001.
7. Sadeghi, A.; Bagheri, H.; Garcia, J.; Malek, S. A Taxonomy and Qualitative Comparison of Program Analysis Techniques for Security Assessment of Android Software. *IEEE Trans. Softw. Eng.* **2017**, *43*, 492–530, doi:10.1109/TSE.2016.2615307.
8. Liu, K.; Xu, S.; Xu, G.; Zhang, M.; Sun, D.; Liu, H. A Review of Android Malware Detection Approaches Based on Machine Learning. *IEEE Access* **2020**, *8*, 124579–124607, doi:10.1109/ACCESS.2020.3006143.
9. Pan, Y.; Ge, X.; Fang, C.; Fan, Y. A Systematic Literature Review of Android Malware Detection Using Static Analysis. *IEEE Access* **2020**, *8*, 116363–116379, doi:10.1109/ACCESS.2020.3002842.
10. Garg, S.; Baliyan, N. Android security assessment: A review, taxonomy and research gap study. *Comput. Secur.* **2021**, 102087, doi:10.1016/j.cose.2020.102087.
11. Wuyts, K. Privacy Threats in Software Architectures. 2015; p. 192. Available online: https://limo.libis.be/primo-explore/fulldisplay?docid=LIRIAS1656390&context=L&vid=Lirias&search_scope=Lirias&tab=default_tab&lang=en_US&fromSitemap=1 (accessed on 16 August 2021)
12. Hansen, M.; Jensen, M.; Rost, M. Protection goals for privacy engineering. In Proceedings of the 2015 IEEE Security and Privacy Workshops, San Jose, CA, USA, 21–22 May 2015; pp. 159–166, doi:10.1109/SPW.2015.13.
13. Stevens, R.; Gibler, C.; Crussell, J.; Erickson, J.; Chen, H. Investigating User Privacy in Android Ad Libraries. In *Workshop on Mobile Security Technologies (MoST)*; Citeseer: Philadelphia, PA, USA; 2012; Volume 10, pp. 195–197.
14. About Android App Bundle | Android developers. Available online: https://developer.android.com/guide/app-bundle (accessed on 16 August 2021).
15. Bourque, P.; Dupuis, R.; Abran, A.; Moore, J.W.; Tripp, L. *Guide to the Software Engineering Body of Knowledge, Version 3.0*; IEEE: Piscataway, NJ, USA, 2014.
16. Alsharif, M.H.; Kelechi, A.H.; Yahya, K.; Chaudhry, S.A. Machine Learning Algorithms for Smart Data Analysis in Internet of Things Environment: Taxonomies and Research Trends. *Symmetry* **2020**, *12*, 88, doi:10.3390/sym12010088.
17. Kong, P.; Li, L.; Gao, J.; Liu, K.; Bissyandé, T.F.; Klein, J. Automated testing of Android apps: A systematic literature review. *IEEE Trans. Reliab.* **2019**, *68*, 45–66, doi:10.1109/TR.2018.2865733.
18. Wieringa, R.; Maiden, N.; Mead, N.; Rolland, C. Requirements engineering paper classification and evaluation criteria: A proposal and a discussion. *Requir. Eng.* **2006**, *11*, 102–107, doi:10.1007/s00766-005-0021-6.
19. Petersen, K.; Vakkalanka, S.; Kuzniarz, L. Guidelines for conducting systematic mapping studies in software engineering: An update. In *Information and Software Technology*; Elsevier: Amsterdam, The Netherlands; 2015; Volume 64, pp. 1–18, doi:10.1016/j.infsof.2015.03.007.
20. Cavacini, A. What is the best database for computer science journal articles? *Scientometrics* **2015**, *102*, 2059–2071, doi:10.1007/s11192-014-1506-1.
21. 2017 IEEE Thesaurus Version 1.0 Created by The Institute of Electrical and Electronics Engineers (IEEE). Technical Report, IEEE. Available online: https://www.ieee.org/publications/services/thesaurus-access-page.html (accessed on 16 August 2021)
22. Computing Classification System. Available online: https://dl.acm.org/ccs (accessed on 16 August 2021).

23. ISO—ISO/IEC/IEEE 24765:2017—Systems and Software Engineering—Vocabulary. Available online: https://standards.iso.org/ittf/PubliclyAvailableStandards/c071952_ISO_IEC_IEEE_24765_2017.zip (accessed on 16 August 2021).
24. Del Alamo, J.M.; Guaman, D.S.; Diez, A.; Balmori, B. Privacy Assessment in Android Apps: A Systematic Mapping Study. *Mendeley Data* **2021**, doi:10.17632/jr349zxzcg.2.
25. InCites—Clarivate Analytics. Available online: https://esi.clarivate.com/ (accessed on 16 August 2021).
26. Krippendorff, K. Testing the reliability of content analysis data: What is involved and why. In *The Content Analysis Reader*; SAGE Publications: New York, NY, USA; 2009 .
27. Octeau, D.; Jha, S.; McDaniel, P. Retargeting Android applications to Java bytecode. In Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering, Cary North, CA, USA, 11–16 November 2012, doi:10.1145/2393596.2393600.
28. Vallée-Rai, R.; Hendren, L.; Co, P.; Lam, P.; Gagnon, E.; Sundaresan, V. Soot—A Java bytecode optimization framework. In CASCON '10: CASCON First Decade High Impact Papers, Toronto, ON, Canada, 1–4 November 2010; pp. 214–224, doi:10.1145/1925805.1925818.
29. Miecznikowski, J.; Hendren, L. Decompiling Java bytecode: Problems, traps and pitfalls. In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*; Springer Verlag: Berlin/Heidelberg, Germany, 2002; Volume 2304, pp. 111–127, doi:10.1007/3-540-45937-5_10.
30. Bartel, A.; Klein, J.; Monperrus, M. Dexpler: Converting android dalvik bytecode to jimple for static analysis with soot. In Proceedings of the ACM SIGPLAN International Workshop on State of the Art in Java Program Analysis, Beijing, China, 14 June 2012; pp. 27–38, doi:10.1145/2259051.2259056.
31. Octeau, D.; McDaniel, P.; Jha, S.; Bartel, A.; Bodden, E.; Klein, J.; Le Traon, Y. Effective inter-component communication mapping in android with epicc: An essential step towards holistic security analysis. In Proceedings of the 22nd USENIX Security Symposium, Washington, DC, USA, 14–16 August 2013; pp. 543–558.
32. Octeau, D.; Luchaup, D.; Dering, M.; Jha, S.; McDaniel, P. Composite constant propagation: Application to android inter-component communication analysis. In Proceedings of the International Conference on Software Engineering, Florence, Italy, 16–24 May 2015; Volume 1, pp. 77–88, doi:10.1109/ICSE.2015.30.
33. Li, L.; Bartel, A.; Bissyandé, T.F.; Klein, J.; Traon, Y.L.; Arzt, S.; Rasthofer, S.; Bodden, E.; Octeau, D.; McDaniel, P. IccTA: Detecting inter-component privacy leaks in android apps. In Proceedings of the International Conference on Software Engineering, Florence, Italy, 16–24 May 2015; Volume 1, pp. 280–291, doi:10.1109/ICSE.2015.48.
34. IDA Pro—Hex Rays Available online: https://hex-rays.com/ida-pro/ (accessed on 16 August 2021).
35. Choudhary, S.R.; Gorla, A.; Orso, A. Automated Test Input Generation for Android: Are We There Yet? In Proceedings of the 2015 30th IEEE/ACM International Conference on Automated Software Engineering, Lincoln, NE, USA, 9–13 November 2015; pp. 429–440.
36. Gürses, S. Can you engineer privacy? *Commun. ACM* **2014**, *57*, 20–23, doi:10.1145/2633029.
37. Nissenbaum, H. Privacy as contextual integrity. *Wash. Law Rev.* **2004**, *79*, 119–157.
38. ARTICLE 29 DATA PROTECTION WORKING PARTY Opinion 02/2013 on Apps on Smart Devices. Available online: https://ec.europa.eu/justice/article-29/documentation/opinion-recommendation/files/2013/wp202_en.pdf (accessed on: 16 August 2021)
39. Trade Commission, F. Mobile privacy disclosures: Building trust through transparency. In *Mobile Privacy Disclosures: Recommendations of the Federal Trade Commission*; Federal Trade Commission: Washington, DC, USA, 2013; pp. 1–34.
40. Guaman, D.S.; Del Alamo, J.M.; Caiza, J.C. GDPR Compliance Assessment for Cross-border Personal Data Transfers in Android Apps. *IEEE Access* **2021**, *9*, doi:10.1109/ACCESS.2021.3053130.
41. GDPR Fines & Data Breach Penalties. Available online: https://www.enforcementtracker.com/ (accessed on 16 August 2021).
42. Castelluccia, C.; Gürses, S.; Hansen, M.; Hoepman, J.H.; Hoboken, J.V.; Vieira, B. *Privacy and Data Protection in Mobile Applications: A Study on the App Development Ecosystem and the Technical Implementation of GDPR*; ENISA: Athens, Greece, 2017; p. 70.
43. Rashid, A.; Chivers, H.; Danezis, G.; Lupu, E.; Martin, A. *The Cyber Security Body of Knowledge (CyBoK) 1.0*; University of Bristol: Bristol, UK, 2019; p. 299.
44. McIlroy, S.; Ali, N.; Khalid, H.; E. Hassan, A. Analyzing and automatically labelling the types of user issues that are raised in mobile app reviews. *Empir. Softw. Eng.* **2016**, *21*, 1067–1106, doi:10.1007/s10664-015-9375-7.
45. Mobile Android Version Market Share Worldwide | StatCounter Global Stats. Available online: https://gs.statcounter.com/android-version-market-share/mobile/worldwide/ (accessed on 16 August 2021).
46. Guaman, D.S.; Alamo, J.M.; Caiza, J.C. A Systematic Mapping Study on Software Quality Control Techniques for Assessing Privacy in Information Systems. *IEEE Access* **2020**, *8*, 74808–74833, doi:10.1109/ACCESS.2020.2988408.
47. Ebrahimi, F.; Tushev, M.; Mahmoud, A. Mobile App Privacy in Software Engineering Research: A Systematic Mapping Study. *Inf. Softw. Technol.* **2019**, *14*, 106466.