KnowledgeZooClient: Constructing Knowledge Graph for Android

Li Li
Faculty of Information Technology,
Monash University
Melbourne, Australia

Jun Gao, Pingfan Kong
SnT,
University of Luxembourg
Luxembourg

Haoyu Wang
Beijing University of Posts and Telecommunications
Beijing, China

Mengyu Huang
Hong Kong University of Science and Technology
Hong Kong, China

Yuan-Fang Li
Faculty of Information Technology,
Monash University
Melbourne, Australia

Tegawendé F. Bissyandé
SnT,
University of Luxembourg
Luxembourg

ABSTRACT

In this work, we describe the design and implementation of a reusable tool named KnowledgeZooClient targeting the construction, as a crowd-sourced effort, of a knowledge graph for Android apps. KnowledgeZooClient is made up of two modules: (1) the Metadata Extraction Module (MEM), which aims at extracting metadata from Android apps and (2) the Metadata Integration Module (MIM) for importing and integrating extracted metadata into a graph database. The usefulness of KnowledgeZooClient is demonstrated via an exclusive knowledge graph called KnowledgeZoo, which contains information on over 500,000 apps already and still keeps growing. Interested users can already benefit from KnowledgeZoo by writing advanced search queries so as to collect targeted app samples.

ACM Reference Format:

Li Li, Jun Gao, Pingfan Kong, Haoyu Wang, Mengyu Huang, Yuan-Fang Li, and Tegawendé F. Bissyandé. 2020. KnowledgeZooClient: Constructing Knowledge Graph for Android. In *35th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW '20), September 21–25, 2020, Virtual Event, Australia.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3417113.3422187

1 INTRODUCTION

Google's Android mobile operating system, has steadily increased its market share since 2011 and is now leading the global market, occupying over 85% of market share. One reason contributing to this success could be the continuous and rapid growth of Android apps. Indeed, as of June 2018, the number of Android apps on Google Play, the official Android app market, has exceeded 3.3 million, with only 14% of them considered as low-quality apps[1].

The success of Android, on the one hand, brings several benefits to app developers and users, while on the other hand, it makes Android the target of choice to attackers and opportunistic developers. As demonstrated in the recent Symantec Internet Security Threat Report [1], threats in the mobile space continue to grow year on year. In 2017, the number of new malware variants has increased 54%. To cope with this, researchers have introduced various approaches (e.g., privacy leaks identification [2], ad fraud detection, etc.) to secure Android apps so as to keep users from being infected [3].

All of the above approaches require a reliable benchmark dataset to evaluate their performance. Unfortunately, it is not easy to build such a targeted dataset from scratch. As a result, researchers often apply their approaches to randomly selected apps, including both relevant and irrelevant samples. Many efforts are hence wasted to analyse the irrelevant ones. As an example, many researchers now rely on AndroZoo [4], which provides over 5.8 million Android apps for the community, to obtain experimental samples. If a researcher is only interested in apps that are obfuscated and have used reflection in their code, she still has to download all the apps and then filter in the interested ones.

To tackle this problem, Li et al. [5] attempt to facilitate the access of AndroZoo by providing to the research community sufficient metadata associated with the AndroZoo apps. Users of AndroZoo can therefore pre-select a set of apps that are all relevant to their research. Unfortunately, the metadata is shared via plain text, which is difficult to leverage for complicated cases (e.g., when many conditions are applied). Therefore, to supplement this work, we plan to represent the app metadata via a knowledge graph and share it with the community for our fellow researchers to quickly search for interested apps.

Knowledge graphs, closely related to ontology, encompass a structured and semantic representation of entities, their properties and relationships in a domain (e.g., Android). It provides an entity-centric view of the linked data that users can leverage to understand how the different artefacts are connected. For instance, the signature used to sign app_a, which is flagged by VirusTotal as malware, is also used to sign app_b. In other words, such a knowledge graph naturally facilitates the traversal via entities as well as their relationships [6]. Moreover, knowledge graphs do not require a predefined schema. As a result, they allow a more flexible update mechanism than traditional relational databases.

To benefit from knowledge graph, we provide to the community a prototype research tool, namely KnowledgeZooClient, aiming to

[1] https://www.appbrain.com/stats/number-of-android-apps

ASEW '20, September 21–25, 2020, Virtual Event, Australia

ACM ISBN 978-1-4503-8128-4/20/09…$15.00
https://doi.org/10.1145/3417113.3422187

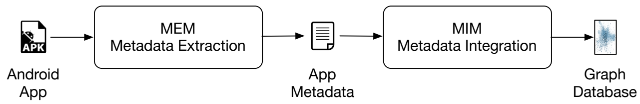

Figure 1: Knowledge Graph Construction Overview.

extract metadata from Android apps and integrate them into a graph database (i.e., knowledge graph). To demonstrate the usefulness of KnowledgeZooClient, we have applied it on 500,000 Android apps selected from AndroZoo. It takes roughly one day (in parallel with 24 instances) for KnowledgeZooClient to extract the metadata from all the apps and takes several minutes to integrate the metadata into a graph database, which then can be leveraged to support advanced search queries, e.g., to find apps that are released between 2016 and 2017, have used dynamic code loading and refection features, have accessed the *Apache Commons* library and have shared the same capability of handling incoming broadcasts.

In addition to KnowledgeZooClient, which has been made available on Github[2],we aim at also building an exclusive knowledge graph (namely KnowledgeZoo[3]) of Android apps integrating as many Android apps as possible. The 500,000 apps are our first attempt towards constructing such a graph. We also encourage crowd-sourced efforts, by running KnowledgeZooClient, to enrich KnowledgeZoo with more app metadata.

2 KNOWLEDGEZOOCLIENT

The external goal of this work is to provide a knowledge graph of Android apps for our fellow researchers working in the field of mobile app analysis to quickly search for relevant artefacts so as to facilitate their research in various means, e.g., to search for app samples exactly suitable for their experiments. To this end, we introduce a prototype tool called KnowledgeZoo, for which we share with the community to encourage crowd-sourced efforts to fulfil the aforementioned goal.

Figure 1 outlines the overview of the working process of KnowledgeZooClient, where the knowledge graph construction process is made up of two modules. In the first module, namely MEM, KnowledgeZooClient attempts to extract app metadata from Android apps. After app metadata is collected, in the second module, namely MIM, KnowledgeZooClient integrates the harvested app metadata into a growing graph database. We now detail these two modules in Section 2.1 and Section 2.2, respectively.

2.1 Metadata Extraction

The metadata extraction module takes as input an Android app and outputs its metadata information in json format, which can then be integrated into a graph database, via the KnowledgeZooClient's MIM. Table 1 enumerates the metadata types so far KnowledgeZooClient harvests.

2.1.1 APK. Android apps are released and distributed through APK files. For each Android APK file, KnowledgeZooClient collects six artefacts for our fellow researchers.

- **SHA256/SHA1/MD5:** The hash values are provided to uniquely identify an Android app.
- **APK Size:** We provide the APK size information (in byte) of a given app aiming at providing a quick means for researchers to calculate how much space they may need in order to download the actual app to their local disk.
- **Market:** For each APK, when crawling from a market, we also record from which market it is downloaded. Since we do not have a good means to check if a given APK has already been downloaded from other markets, the same APK could be downloaded from several markets. This market artefact represents the market where the app is downloaded from. If the app appears in multiple markets, the market artefact correspondingly shows all the available markets with each separated by a vertical virgule (e.g., *play.google.com|anzhi* indicates that the app is downloaded from both Google Play and the Anzhi market).
- **Certificate:** we provide the certificate signature of a given app, which is usually used to represent the signature of its developers. Literature works recurrently leverage this metadata to pinpoint repackaged (or piggybacked) Android apps, which usually involve a change of app signatures (i.e., developers) [7].

2.1.2 Manifest. Every Android app is assembled with a global configuration file called *AndroidManifest.xml* (hereinafter referred as manifest), which plays an important role for configuring app's permissions, components, etc. In this work, we provide five artefacts that are directly extracted from the manifest file of a given app. Those five artefacts are as follows:

- **Package Name:** Package name, also known as application ID, is used to uniquely identify an app on a device and in Google Play. For example, two apps with the same application ID cannot be installed concurrently on the same device. It is also disallowed to update an app in Google Play with a different application ID.
- **Version Code/Name:** Version code/name is provided by app developers to track and name the different versions of their developed apps. Theoretically, version code increases as the app updates. Literature work leverage these information to build app lineages (i.e., sorted app versions of the same app) for supporting the investigation of app evolution.
- **Sdk Version:** Sdk version specifies the API level that the app is targeted. Since different API levels may provide different features, users of KnowledgeZoo could thus leverage this information to conduct evolution-based investigations, especially in couple with the evolution investigation of the Android operating system. As an example, Li et al. [8] have leveraged this information to investigate the evolution of inaccessible Android APIs (i.e., internal and hidden APIs) of the framework code.
- **Permission List:** Permissions, declared by app developers, are used by Android system to grant the access of protected parts and controls the interaction with other apps. Listing 1 illustrates an example of permission list. Many state-of-the-art works have leveraged permissions to perform specific analyses [9, 10]. In this work, we provide directly the used

[2]Hide for double blind reviewing.
[3]KnowledgeZoo will be shared with our fellow researchers as an independent online service.

Table 1: Metadata Overview.

Type	Name	Example
	SHA256	DFFCF9ACC7E4AC23895BF8B4AC4F12A450B68B...
	SHA1	338A00E4A27C20B139ECD751CC46614711220643
	MD5	920FE692FA1405573BFB1FE7F3B84C51
APK	Apk Size	5461580
	Market	play.google.com
	Certificate (fingerprint)	EC:05:3F:94:81:E8:29:36:3D:A2:3D:5E:0C:BA:A3...
	Certificate (owner)	CN=AndroidRCert, OU=Android, O=LGEMC, L=Seoul, ST=Seoul, C=KR
	Package Name	com.lge.friendsmanager
	Main Activity	com.lge.friendsmanager.FriendsManagerMainActivity
	Version Code	5000019
Manifest	Version Name	5.0.19
	Sdk Version (Targeted)	23
	Permission List	cf. Listing 1
	Component List	cf. Listing 2
	Dex Size	5489904
	Dex Date	01/01/2009 00:00:00
	Native Code	true
DEX	Dynamic Code	false
	Reflection	true
	Obfuscation	false
	Package List	cf. Listing 3

permissions to the research community for boosting the research of permission-based investigations. Uses of KnowledgeZoo now can collect the declared permissions of a given app without actually downloading the app.

- **Component List:** Component is the basic unit constituting an Android app. In Android, there are four types of components: (1) Activity represents user interfaces, the GUI part of an app (2) Service executes compute-intensive tasks in the background, (3) Broadcast Receiver waits to receive event messages and (4) Content Provider provides a means to share structured data within an app or between different apps. In this work, not only the component names and types are collected, KnowledgeZooClient also records the capability of every component declared in the manifest, which can then be leveraged to support advanced inter-component communication analysis.

2.1.3 DEX. DEX is the file format of the so-called Dalvik Executable format, which stores the actual code of Android apps. Every Android app should have a DEX file called *classes.dex* in the top direct of a given APK file, which is also the target where all the metadata in this group collected from.

- **Dex Size:** Similar to APK size, Dex size provides another means for users of KnowledgeZoo to pre-select their favoured apps, e.g., apps with their Dex size smaller than 1 megabyte.
- **Dex Date:** Dex (assembly) date is extracted based on the last modified time of the Dex file. It can be used to represent the assembling time of a given app and thereby to support all the time-relevant investigations.

```
android.permission.INTERNET
com.lge.permission.MANAGE_PERMISSIONS
android.permission.ACCESS_NETWORK_STATE
android.permission.BLUETOOTH
android.permission.BLUETOOTH_ADMIN
android.permission.ACCESS_FINE_LOCATION
```

Listing 1: Permission List Example.

- **Native Code:** Native code, if true, shows that the app has accessed native code, which is usually written in C or C++.
- **Dynamic Code:** Dynamic code, if true, indicates that the app has loaded additional code at runtime.
- **Reflection:** Reflection, if true, demonstrates that the app has accessed reflective code. Usually, if the dynamic code item is true, reflection item should be also true, as the additionally loaded code should normally be accessed through reflection.
- **Obfuscation:** Obfuscation, if true, indicates that the app has been obfuscataed. Researchers such as the authors of [11] could benefit from this artefact to only collect (e.g., download from AndroZoo) obfuscated Android apps, avoiding the download of irrelevant apps and also the analysis of those potential irrelevant apps.
- **Package List:** Package list enumerates all the package names of an app (cf. Listing 3). We expect this information to be used by researchers for selecting apps with accessing specific packages. For example, researchers could leverage this information to select a set of apps with which all of them have integrated with ad library *com.wandoujia.ads*.

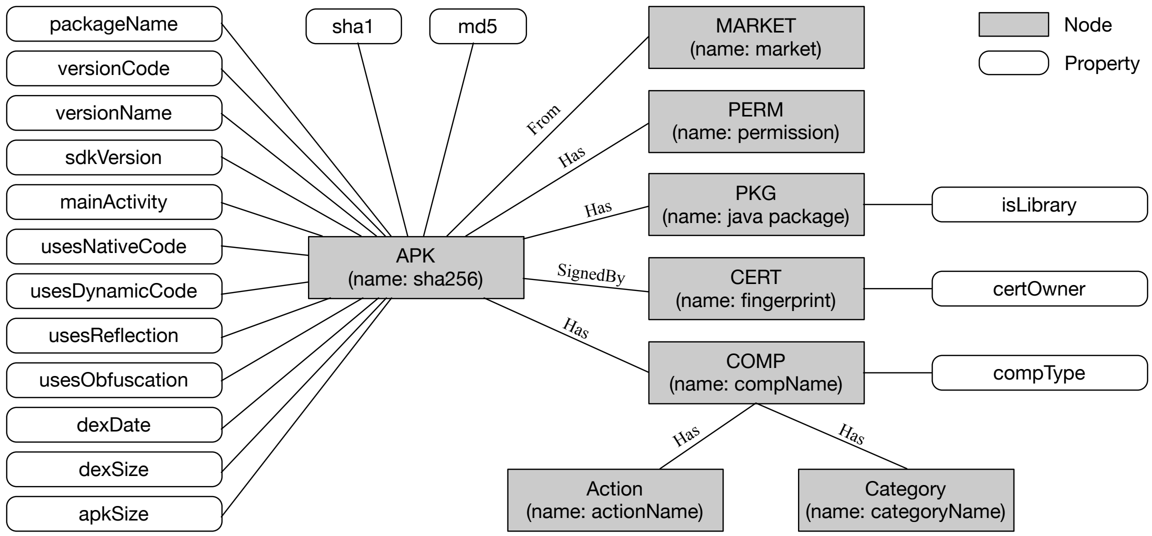

Figure 2: The Schema of KnowledgeZoo Graph Database.

```
com.lge.friendsmanager.welcome.WelcomeActivity
com.lge.app.permission.RequestPermissionsActivity
com.lge.friendsmanager.download.
    FriendsListDownloaderService
com.lge.friendsmanager.friends.FriendsReceiver
  {action: ['android.intent.action.PACKAGE_ADDED',
     'android.intent.action.PACKAGE_REMOVED',
     'android.intent.action.PACKAGE_REPLACED',
     'android.intent.action.PACKAGE_FULLY_REMOVED']}
```

Listing 2: Component List Example.

```
com.lge.bnr
com.lge.config
com.lge.resource
android.media.IAudioServiceEx
com.lge.sound
android.net.IConnectivityManagerEx
```

Listing 3: Package List Example.

2.2 MIM: Metadata Integration

Metadata Integration module runs on the server side aiming at importing and integrating the outputs of MEM (i.e., metadata of Android apps) into the current graph database. At the moment, KnowledgeZooClient supports two types of integration: (1) CSV files with pre-defined and well-structured headers and (2) Cypher scripts, a declarative, SQL-inspired language for describing patterns in graphs visually using an ASCII-art syntax. Both types are supported by modern graph databases such as *neo4j*, the one used in this work.

MIM takes into account all the extracted metadata of a given Android app and groups them into eight types. As shown in Figure 2, the schema of our graph database, each type is represented by a knowledge graph node with some properties, e.g., node *MARKET* has no property, node *COMP* has one property named *compType*, while node APK has 12 properties.

In knowledge graphs, an edge between two nodes can be associated with some terms specifying the relationships between them. As illustrated in Figure 2, three basic relationships (i.e., "Has", "From" and "SignedBy") have been defined. Based on these relationships, one can clearly understand, from the schema of the graph database, the relationships between different nodes, e.g., a market (*MARKET* node) can have (i.e., via "From" relationship) Android apps (*APK* node) while an app can have permissions (*PERM* node).

It is worth mentioning that, in addition to existing edge types, which reflect direct relationship between nodes, one can introduce additional relationships to enhance the knowledge graph. Some of these relationships could be derived from existing ones through graph mining. For example, based on the java packages and the signing certificate, we can introduce "similar" or "repackaging" relationships to APK nodes.

3 KNOWLEDGEZOO

To demonstrate the usefulness of KnowledgeZooClient and show the advantages of having a knowledge graph of Android apps, we apply KnowledgeZooClient on 578,580 Android apps, which represent all the malicious apps available in AndroZoo, to construct a preliminary knowledge graph (hereinafter referred to as KnowledgeZoo). With a modern server (24 cores), it takes roughly one days for KnowledgeZooClient to extract metadata from the half million Android apps and about two minutes to import the metadata into the graph database. Among the 578,580 Android malware considered, KnowledgeZoo contains 5,035,760 distinct nodes and 39,997,923 relationships.

We believe the KnowledgeZoo knowledge graph has great potential to support various applications. The most basic application would be to search for app samples under specific conditions. For example, suppose that a researcher wants to apply her/his approach on a set of Android apps that are released between 2016 and 2017 on Google Play, have leveraged the *Apache http* library and listens to the *BOOT_COMPLETED* system event. Instead of downloading

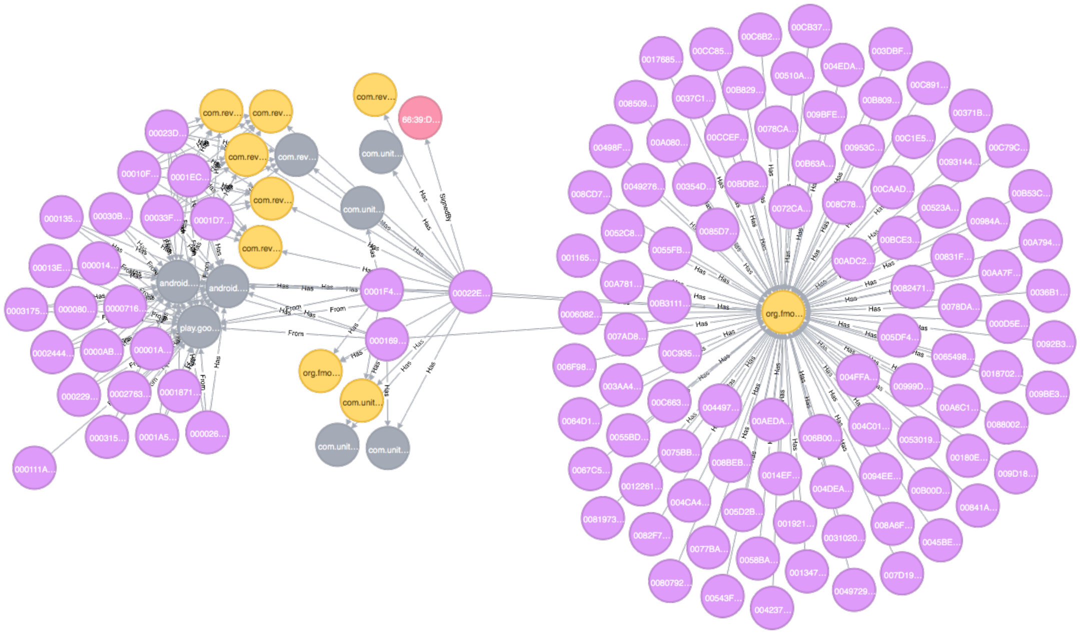

Figure 3: An example of interactive graph representation.

```
MATCH (n:APK)-[:Has]->(p:PKG),
      (n)-[:From]->(m:MARKET),
      (n)-[:Has]->()-[:Has]->(a:ACTION) where
      n.dexDateInMillis > 1451602800000 and
      n.dexDateInMillis < 1483225200000 and p.name
      starts with "org.apache.http" and m.name
      contains "play.google.com" and  a.name ends with
      "BOOT_COMPLETED"  return n;
```

Listing 4: A query example written in the Cypher language, which is supported by our graph database.

all the apps from an app repository like AndroZoo to search for such apps, which is time-consuming, one can leverage KnowledgeZoo to quickly achieve that, e.g., by sending the query shown in Listing 4. Indeed, KnowledgeZoo is capable of understanding the query and outputs the results (i.e., 312 out of 578,580 apps match the aforementioned conditions) in a short time.

Moreover, as demonstrated in Figure 3, in addition to the contextual results, KnowledgeZoo can also visually and interactively present the results in a graph, for which analytics can interact with so as to have a better understanding of the results.

Recall that with KnowledgeZooClient, our external goal is to provide a comprehensive and growing knowledge graph of Android apps for our research community. Hence, in our future work, we will continuously integrate more apps into the KnowledgeZoo knowledge graph. We also encourage crowd-sourced efforts, by running KnowledgeZooClient, to enrich KnowledgeZoo with more apps. We plan to also extend the ability of KnowledgeZooClient to include more artifacts of Android apps so as to enhance KnowledgeZoo with more types of nodes and relationships.

4 RELATED WORK

Since 2018, the first time when Android releases, Android has become the most successful mobile system in the industry and one of the most popular research targets in the research community. Researchers have spent tremendous efforts in analyzing and improving Android apps and their running framework systems, as well as the overall Android ecosystem [3, 12, 13]. They have attempted to detect security issues of Android apps [14–16], dissect malicious behaviors of Android malware [7, 17], characterize compatibility issues of Android apps [18–21], improve code qualities of Android apps [22, 23], mitigate energy concerns of Android apps [24, 25], etc.

The aforementioned research studies have involved large sets of Android apps, either benign or malicious apps. To prepare the app sets, the authors of these studies either crawl directly from app markets or reuse existing ones that are carefully crafted for supporting Android-based researches. Our community has actually proposed various app datasets [4, 5, 26–29]. The most representative one would be the AndroZoo dataset [4], which has collected over 10 million Android apps and is no doubt the largest dataset of close-sourced Android apps. In line with AndroZoo, Liu et al. [30] have recently proposed another dataset called AndroZooOpen, which contains over 70,000 open-sourced Android apps.

In addition to providing the artifacts of Android apps, our community starts to offer easily accessible fine-grained app datasets. These datasets, often provided as knowledge graphs, include not only the apps' metadata (including basic metadata such as the apps' attributes and advanced metadata such as experimental results returned by static/dynamic analysis approaches) but also pre-computed relationships of those apps, aiming at facilitating researchers to conduct automated Android app analyses. For example, Meng et al. [31] have presented to the community a knowledge graph called AndroVault and demonstrated that their knowledge graph helps to promote productive research studies such as malware detection and malware generation [32]. Our approach, although targeting different types of metadata, could be supplementary to theirs, and the knowledge graph constructed in this work should also be capable of facilitating Android-based research.

5 CONCLUSION

In this work, we present a research prototype tool called KnowledgeZooClient, which extracts metadata (over 20 types of attributes) from Android apps and integrates them into a graph database. We have applied KnowledgeZoo to more than 500,000 apps, resulting in a knowledge graph, for which we call it as KnowledgeZoo, containing 5,035,760 nodes and 39,997,923 relationships. So far, we demonstrate the usefulness of KnowledgeZoo with a simple example. We believe that the KnowledgeZoo knowledge graph will have great potential to support various applications. In our future work, while enlarging the knowledge graph in terms of the number of apps, we will also introduce more applications (such as revealing repackaged apps) on top of KnowledgeZoo. We also encourage our fellow researchers to leverage KnowledgeZoo to provide creative applications.

ACKNOWLEDGMENT

This work was supported by the Australian Research Council (ARC) under projects DE200100016 and DP200100020, by the National Natural Science Foundation of China (No. 61702045), and by the Fonds National de la Recherche (FNR), Luxembourg, under project CHARACTERIZE C17/IS/11693861.

REFERENCES

[1] Symantec. Executive Summary - 2018 Internet Security Threat Report. Technical report, 2018. https://www.symantec.com/content/dam/symantec/docs/reports/istr-23-executive-summary-en.pdf.

[2] Li Li, Alexandre Bartel, Tegawendé F Bissyandé, Jacques Klein, Yves Le Traon, Steven Arzt, Siegfried Rasthofer, Eric Bodden, Damien Octeau, and Patrick Mcdaniel. IccTA: Detecting Inter-Component Privacy Leaks in Android Apps. In *ICSE*, 2015.

[3] Li Li, Tegawendé F Bissyandé, Mike Papadakis, Siegfried Rasthofer, Alexandre Bartel, Damien Octeau, Jacques Klein, and Yves Le Traon. Static analysis of android apps: A systematic literature review. *Information and Software Technology*, 2017.

[4] Kevin Allix, Tegawendé F Bissyandé, Jacques Klein, and Yves Le Traon. Androzoo: Collecting millions of android apps for the research community. In *Proceedings of the 13th International Conference on Mining Software Repositories*, pages 468–471. ACM, 2016.

[5] Li Li, Jun Gao, Médéric Hurier, Pingfan Kong, Tegawendé F Bissyandé, Alexandre Bartel, Jacques Klein, and Yves Le Traon. Androzoo++: Collecting millions of android apps and their metadata for the research community. *arXiv preprint arXiv:1709.05281*, 2017.

[6] Guozhu Meng, Yinxing Xue, Jing Kai Siow, Ting Su, Annamalai Narayanan, and Yang Liu. AndroVault: Constructing Knowledge Graph from Millions of Android Apps for Automated Analysis. *CoRR*, abs/1711.07451, 2017.

[7] Li Li, Daoyuan Li, Tegawendé F Bissyandé, Jacques Klein, Yves Le Traon, David Lo, and Lorenzo Cavallaro. Understanding android app piggybacking: A systematic study of malicious code grafting. *IEEE Transactions on Information Forensics & Security (TIFS)*, 2017.

[8] Li Li, Tegawendé F Bissyandé, Yves Le Traon, and Jacques Klein. Accessing inaccessible android apis: An empirical study. In *The 32nd International Conference on Software Maintenance and Evolution (ICSME 2016)*, 2016.

[9] Adrienne Porter Felt, Erika Chin, Steve Hanna, Dawn Song, and David Wagner. Android permissions demystified. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 627–638. ACM, 2011.

[10] Michael Backes, Sven Bugiel, Erik Derr, Patrick D McDaniel, Damien Octeau, and Sebastian Weisgerber. On demystifying the android application framework: Revisiting android permission specification analysis. In *USENIX Security Symposium*, pages 1101–1118, 2016.

[11] Shuaike Dong, Menghao Li, Wenrui Diao, Xiangyu Liu, Jian Liu, Zhou Li, Fenghao Xu, Kai Chen, Xiaofeng Wang, and Kehuan Zhang. Understanding android obfuscation techniques: A large-scale investigation in the wild. *arXiv preprint arXiv:1801.01633*, 2018.

[12] Pingfan Kong, Li Li, Jun Gao, Kui Liu, Tegawendé F Bissyandé, and Jacques Klein. Automated testing of android apps: A systematic literature review. *IEEE Transactions on Reliability*, 2018.

[13] William Martin, Federica Sarro, Yue Jia, Yuanyuan Zhang, and Mark Harman. A survey of app store analysis for software engineering. *IEEE transactions on software engineering*, 43(9):817–847, 2016.

[14] Fenghao Xu, Wenrui Diao, Zhou Li, Jiongyi Chen, and Kehuan Zhang. Badbluetooth: Breaking android security mechanisms via malicious bluetooth peripherals. In *NDSS*, 2019.

[15] Tianming Liu, Haoyu Wang, Li Li, Xiapu Luo, Feng Dong, Yao Guo, Liu Wang, Tegawendé F Bissyandé, and Jacques Klein. Maddroid: Characterising and detecting devious ad content for android apps. In *The Web Conference 2020 (WWW 2020)*, 2020.

[16] Jun Gao, Li Li, Pingfan Kong, Tegawendé F Bissyandé, and Jacques Klein. Borrowing your enemy's arrows: the case of code reuse in android via direct inter-app code invocation. In *ESEC/FSE 2020*, 2020.

[17] Li Li, Tegawendé F Bissyandé, and Jacques Klein. Simidroid: Identifying and explaining similarities in android apps. In *The 16th IEEE International Conference On Trust, Security And Privacy In Computing And Communications (TrustCom 2017)*, 2017.

[18] Li Li, Tegawendé F Bissyandé, Haoyu Wang, and Jacques Klein. Cid: Automating the detection of api-related compatibility issues in android apps. In *The ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA 2018)*, 2018.

[19] Haipeng Cai, Ziyi Zhang, Li Li, and Xiaoqin Fu. A large-scale study of application incompatibilities in android. In *The 28th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA 2019)*, 2019.

[20] Lili Wei, Yepang Liu, and Shing-Chi Cheung. Pivot: learning api-device correlations to facilitate android compatibility issue detection. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pages 878–888. IEEE, 2019.

[21] Lili Wei, Yepang Liu, Shing-Chi Cheung, Huaxun Huang, Xuan Lu, and Xuanzhe Liu. Understanding and detecting fragmentation-induced compatibility issues for android apps. *IEEE Transactions on Software Engineering*, 2018.

[22] Li Li, Jun Gao, Tegawendé F Bissyandé, Lei Ma, Xin Xia, and Jacques Klein. Cda: Characterising deprecated android apis. *Empirical Software Engineering (EMSE)*, 2020.

[23] Luis Corral and Ilenia Fronza. Better code for better apps: a study on source code quality and market success of android applications. In *2015 2nd ACM International Conference on Mobile Software Engineering and Systems*, pages 22–32. IEEE, 2015.

[24] Luis Cruz, Rui Abreu, John Grundy, Li Li, and Xin Xia. Do energy-oriented changes hinder maintainability? In *The 35th IEEE International Conference on Software Maintenance and Evolution (ICSME 2019)*, 2019.

[25] Mario Linares-Vásquez, Gabriele Bavota, Carlos Bernal-Cárdenas, Rocco Oliveto, Massimiliano Di Penta, and Denys Poshyvanyk. Mining energy-greedy api usage patterns in android apps: an empirical study. In *Proceedings of the 11th Working Conference on Mining Software Repositories*, pages 2–11, 2014.

[26] Haoyu Wang, Junjun Si, Hao Li, and Yao Guo. Rmvdroid: towards a reliable android malware dataset with app metadata. In *MSR*, pages 404–408. IEEE, 2019.

[27] Fengguo Wei, Yuping Li, Sankardas Roy, Xinming Ou, and Wu Zhou. Deep ground truth analysis of current android malware. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 252–276. Springer, 2017.

[28] Daniel Arp, Michael Spreitzenbarth, Hugo Gascon, Konrad Rieck, and CERT Siemens. Drebin: Effective and explainable detection of android malware in your pocket. 2014.

[29] Li Li, Tegawendé F Bissyandé, and Jacques Klein. Rebooting research on detecting repackaged android apps: Literature review and benchmark. *IEEE Transactions on Software Engineering (TSE)*, 2019.

[30] Pei Liu, Li Li, Yanjie Zhao, Xiaoyu Sun, and John Grundy. Androzooopen: Collecting large-scale open source android apps for the research community. In *The 2020 International Conference on Mining Software Repositories, Data Track (MSR 2020)*, 2020.

[31] Guozhu Meng, Yinxing Xue, Jing Kai Siow, Ting Su, Annamalai Narayanan, and Yang Liu. Androvault: Constructing knowledge graph from millions of android apps for automated analysis. *arXiv preprint arXiv:1711.07451*, 2017.

[32] Guozhu Meng, Yinxing Xue, Chandramohan Mahinthan, Annamalai Narayanan, Yang Liu, Jie Zhang, and Tieming Chen. Mystique: Evolving android malware for auditing anti-malware tools. In *Proceedings of the 11th ACM on Asia conference on computer and communications security*, pages 365–376, 2016.