MAKG: A Mobile Application Knowledge Graph for the Research of Cybersecurity

Heng Zhou[1,3], Weizhuo Li[2,3(✉)], Buye Zhang[1,3], Qiu Ji[2,3], and Yiming Tan[1,3]

[1] School of Cyber Science and Engineering, Southeast University, China.
{zhouheng2020,zhangbuye,tt_yymm}@seu.edu.cn
[2] School of Modern Posts, Nanjing University of Posts and Telecommunications, China.
[3] Key Laboratory of Computer Network and Information Integration (Southeast University), Ministry of Education, China.
liweizhuo@amss.ac.cn qiuji@njupt.edu.cn

Abstract. Large-scale knowledge bases for mobile applications (a.k.a."app") such as AndroZoo++ and AndroVault have become powerful assets for malware detection and channel monitoring. However, these datasets focus on the scale of apps while most apps in them remain isolated and cannot easily be referenced and linked from other apps. To improve the situation, in this paper, we present a mobile application knowledge graph, namely MAKG, which aims to collect the apps from various resources (e.g., application markets, encyclopedias, news). We design a lightweight ontology of apps. It can bring a well-defined schema of collected apps so that these apps could share more linkage with each other. Moreover, we also evaluate the algorithms of information extraction and knowledge alignment during the process of construction. Experimental results show that several algorithms can be competent to above tasks to some extend and enrich the structured triples in MAKG. Finally, we list three use-cases about MAKG that are helpful to provide better services for security analysts and users.

Key words: Mobile Applications, Knowledge Graph, Ontology, Relation Extraction, Knowledge Alignment

1 Introduction

With the popularity of smart phones and mobile devices, the number of mobile applications (a.k.a. "app") has been growing rapidly, which provides great convenience to users for online shopping, education, entertainment, financial management etc [1]. According to a recent report[4], as of January 2021, there were over 1.8 and 3.14 million apps available on Google Play and App Store, respectively, and global downloads of mobile applications have exceeded 218 billion. With large amounts of apps, it also provided the chances for cybercriminals such as thieving private data, propagating false news and pornography, online scam. According to the statistic of National Internet Emergency Center (NIEC) in China, since the first half of 2019, there have been more than 15,000 false loan apps by mobile Internet as a carrier that cause substantial damage of numerous users. Therefore, it is essential to build a mobile application knowledge graph for cybersecurity and provide better services (e.g., semantic retrieval, sensitive detection) for security analysts and users.

[4] https://www.appannie.com/cn/insights/market-data/mobile-2021-new-records-beckon/

There exist several knowledge bases of mobile applications that are constructed in the past decade, and gain a remarkable attraction from researchers of both academia and industry [2]. Most of them are ongoing efforts to gather millions of executable apps from diverse sources including Google Play[5] and F-Droid[6], and define dozens of types of apps to share with the research community for relevant research works [3, 4]. Meanwhile, other constructed datasets focus on the specific services in cybersecurity such as malware detection [5], channel monitoring [6], vulnerability assessment [7].

Although above knowledge bases contain various apps, they suffer from two limitations. Firstly, most apps in these datasets remain isolated among application markets, which cannot easily be referenced and linked from other apps [2]. Hence, these apps without comprehensive properties and linkages may raise challenges for security analysts and users who need to judge the security of current apps. Secondly, apps in these knowledge bases lack well-defined schema, which may limit resources to share and reuse for above services in cybersecurity [7].

To fill the above gaps, we dedicate to a continuous effort to collect apps and construct **m**obile **a**pplication **k**nowledge **g**raph, namely MAKG, for the research of cybersecurity. Precisely, we design a lightweight ontology that brings a well-defined schema of collected apps including 26 basic classes, 11 relations and 45 properties. It not only can make apps share more linkage with each other, but also bring better services such as resources integration, recommendation and so on. In addition, we compare the algorithms of information extraction and knowledge alignment during the process of construction. Experimental results show that several algorithms can be competent to above tasks to some extend and make up the lacked value in MAKG.

The rest of this paper is organized as follows. Related work is introduced in Section 2. Section 3 presents the implementation details of our framework for constructing MAKG. The statistic and evaluation are reported in Section 4. Section 5 lists three usecases based on MAKG, followed by a conclusion in Section 6.

2 Related work

Many interesting insights can be learned from data on application markets and aggregations of that data, which gain a remarkable attraction from academia and industry [2].

Drebin [5] provides a considerate number (5,560) of malware to the public with specific malicious behaviors inside, which were identified and classified by a machine learning method. These samples were categorized into 149 families in terms of contained malicious behaviors.

Commercial databases have mirror metadata from Google Play and other app markets and sell access to this information (e.g., appannie.com, appbrain.com and appszoom.com [8]). Most of these commercial databases contain comprehensive metadata of millions of apps but they lack links to other resources.

AndroZoo++ [3] is an ongoing effort to gather executable Android applications from as many sources as possible and make them available for analysts. In addition, the authors figured out 20 types of app metadata to share with the research community for relevant research works.

AndroVault [4] is a knowledge graph of information on over five million Android apps. It has been crawled from diverse sources, including Google Play and F-Droid

[5] https://play.google.com/store/apps

[6] https://f-droid.org/

since 2013. AndroVault computes several attributes for each app based on the downloaded android application package, in which entities can be heuristically clustered and correlated by attributes.

Recently, Kiesling et al. published a cybersecurity knowledge graph called SEPSES [7], which integrated and linked critical information on real-world vulnerabilities, weaknesses and attack patterns from various publicly available sources. In view of a semantic web research perspective, the authors applied Linked Data principles to combine local and public knowledge in a highly dynamic environment characterized by fast-changing, dispersed, and heterogeneous information.

Although there exist research efforts on building knowledge bases of apps, they suffer from some limitations. Commercial databases mainly dedicate to the scale of apps. The goal of their collecting apps is to sell their access. Therefore, its apps are not free for the research community. Drebin, AndroZoo++, AndroVault and SEPSES dedicate to provide abundant apps for malware detection and vulnerability assessment, respectively. Nevertheless, apps in these knowledge bases cannot easily be referenced and lack well-defined schema, which may limit resource integration and provide comprehensive information of apps for cybersecurity analysts. Relatively, MAKG is constructed by a well-defined schema so that these apps could share more linkage with each other. Moreover, our work is the first step towards building a high-quality mobile application dataset from various resources (e.g., application markets, encyclopedias, news), in which the algorithms of information extraction and knowledge alignment are employed during the process of construction.

3 The Construction of Mobile Application Knowledge Graph

Fig. 1 illustrates the overview of MAKG construction pipeline, which mainly includes five phases: ontology definition (described in Section 3.1) app crawling (described in Section 3.2), knowledge extraction (described in Section 3.3), knowledge alignment (described in Section 3.4), and knowledge storage (described in Section 3.5). With the help of MAKG, the security analysts and users can employ better services such as semantic retrieval, recommendation, sensitive detection and so on.

3.1 Ontology definition

To model a well-defined schema of apps for the research of cybersecurity, we discuss with analysts who worked on the China Academy of Industrial Internet, and discover that the vast majority of conceptualizations (e.g., *functionPoint*, *interactionMode*, *availableState*) described for apps are not asserted online. Therefore, we choose appropriate terms based on the survey of existing conceptualizations in view of cybersecurity, and define a set of relations and properties by protégé[7] to cover the related features of mobile applications. On the other hand, we collect the labels from different encyclopedias, and extend the synonyms for relations and properties manually, which is helpful for the models of knowledge extraction and knowledge alignment to enrich structured triples for our knowledge graph.

Fig. 2 shows an overview of the light-weight ontology, where red edges and blue ones represent *subclassof* and *rdfs:type* relations, respectively, which are two basic relations. The green ones represent the relations, and the violet ones represent properties[8].

[7] https://protege.stanford.edu/

[8] For protégé, relations and properties are called object properties and data properties.

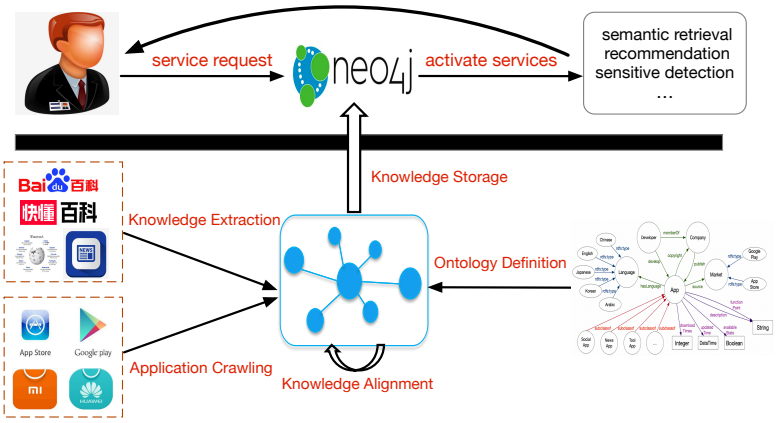

Fig. 1: The workflow of MAKG construction.

Overall, we define 26 basic concepts, 11 relations and 45 properties in the ontology. Benefited from a well-defined schema, it not only can make apps present more comprehensive properties to security analysts and users, but also can generate more shared linkages among apps.

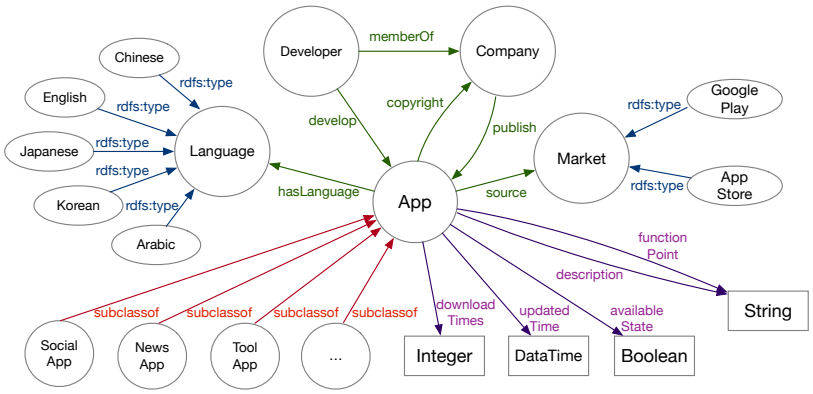

Fig. 2: The overview of lightweight ontology

3.2 App crawling

With the help of scrapy framework, we crawl the descriptive information of apps published from Google Play, App Store[9], Xiaomi App Store[10] and Huawei App Market[11]. We simply give an introduction to these four markets.

- **Google Play**. Its former is called Android Market, which is the official distribution storefront for Android applications and other digital media. It is available on mobile devices and tablets that run the Android operating system (OS).
- **App Store**. It is a digital distribution platform for mobile applications on iOS and iPadOS operating systems that are developed and maintained by Apple Inc.
- **Xiaomi App Store**. It is an android app recommendation software in the ROM of MIUI Android system of Xiaomi technology, which is one of the largest platforms that can discover fun Android apps and games for users.
- **Huawei App Market**. It is the official app distribution platform developed by Chinese technology company Huawei, which is one of the largest platforms that provide Android apps run in the EMUI operating system.

As some application markets (e.g., Google Play) do not provide classification of apps, so we treat some common apps as seeds, and utilize their recommendation results to crawling more apps. Consider the storage space of MAKG, we do not download the application packages and focus on their descriptive information defined in the ontology.

3.3 Knowledge extraction

Crawling description information of apps from application markets is the most direct way to build KG. However, existing labels in the application markets are not inadequate to cover the value of properties in our designed ontology, which impedes the discovery of the shared linkage among apps. Therefore, we try to collect related web pages from encyclopedias (e.g., Baidu Baike[12], Toutiao Baike[13], Wikipedia[14]), textual descriptions and news related to apps that are published on the website so as to fill the lacked value of these properties.

We mainly consider the following strategies to parse the web pages and textual descriptions of apps to obtain some useful structured triples of apps and enrich their lacked value.

- **Infobox-based completion**. It is a common skill that complements the lacked value of properties. We employ the string matching method (e.g., Levenshtein measure) to calculate the similarity of labels in Infobox and properties in ontology, and further construct their correspondences such as *Market*) in ontology *Platform*.
- **Named entity recognition** [9]. For the defined concepts (e.g., *Developer* and *Company*) in our designed ontology, we evaluate several methods named entity recognition and select NcrfPP [10] to capture the related value of apps, which is designed for quick implementation of different neural sequence labeling models with a CRF inference layer and obtain the best performance in our evaluation (see Section 4.2.1).

[9] https://www.apple.com/app-store/
[10] https://app.mi.com/game
[11] https://appgallery.huawei.com/
[12] https://baike.baidu.com
[13] https://www.baike.com/
[14] http://en.wikipedia.org/wiki/Wiki

– **Relation extraction** [11]. For several important relations (e.g., *Headquarter*) and properties (e.g., *publishedTime*) in our designed ontology, we employ promising relation extraction models from two platforms (i.e., OpenNRE[15] and DeepKE[16]), and select the best one based on our dataset to obtain the structured descriptive information of apps (see Section 4.2.2). As several labeled relations in the corpus are not enough, we try to utilize few-shot relation extraction to achieve our goal.

3.4 Knowledge alignment

We notice that most apps are usually published in various application markets (e.g., Android ones), but their descriptive labels and value from application markets are heterogeneous (e.g., the property *name* and its value). Moreover, there exist differences in the perspectives of apps' relations and properties from different application markets. Therefore, it is essential to find the correspondences among entities from different application markets, which can share and reuse this information to provide better services for security analysts and users.

To achieve this goal, we try to employ the following approaches for knowledge graph alignment.

– **Rule miner method** [12]. It is one kind of semi-supervised learning algorithm to iteratively refine matching rules and discover new matches of high confidence based on these rules.
– **Knowledge graph embedding-based method** [13]. It encodes entities and relations of knowledge graph in a continuous embedding space and measures entity similarities based on the learned embeddings.

3.5 Knowledge storage

After we utilized the crawled data to instantiate the properties based on our designed ontology and employed knowledge extraction and knowledge alignment to enrich the structured triples to our knowledge graph, we transform them into structured triples $\{(h, r, t)\}$ by Jena[17]. For knowledge storage, we employ Neo4j[18] to store the transformed triples, which is one of the efficient graph bases for storing the RDF triples and provide one convenience query language called Cypher. To keep MAKG in sync with the evolving apps, we will periodically update the descriptive information of apps by crawling above sources and record the updated logs.

4 Statistic and Evaluation

4.1 The statistic of MAKG

Table 1 lists the statistics of crawled apps from application markets, in which more than 347 thousand apps are collected and they are transformed into about 7 million triples. Notice that the number of apps in Google Play is larger than others because the

[15] https://github.com/thunlp/OpenNRE
[16] https://github.com/zjunlp/deepke
[17] http://jena.apache.org/
[18] https://neo4j.com/

language of their names and descriptions is multilingual, including Chinese, English, Japanese, French and German. In addition, we discover that the sum of relations and properties is less than the defined ones of ontologies. Therefore, we try to complement the lacked value from Baidu Baike and Toutiao Baike. Finally, we retrieve 2493 and 2503 apps from Baidu Baike and Toutiao Baike, and add 47001 and 22091 triples from their infoboxes to MAKG.

Table 1: The statistics of crawled apps from application markets

Application Market	Google Play	App Store	Xiaomi App Store	Huawei App Market
♯Apps	273802	3137	27694	43287
♯Relations	4	3	3	5
♯Properties	15	5	12	14
♯Entities	382358	5724	47478	70954
♯Triples	5210417	25088	384648	1384854

4.2 Evaluation of information extraction

In this section, we mainly evaluate techniques of named entity recognition and relation extraction that are utilized to obtain abundant structured triples of apps and enrich MAKG. The evaluation was conducted on a personal workstation with an Intel(R) Xeon(R) CPU E5-1650 V4 CPU which has 128GB memory and RTX 2080Ti GPU. The evaluated datasets can be downloaded together with the technical report.

Evaluation of named entity recognition We focus on nine entity types defined in ontologies, including *APP*, *City*, *Country*, *Location*, *Company*, *Developer*, *Platform*, *Time* and *Others*. For the dataset, we randomly selected 3676 unstructured textual descriptions of apps from encyclopedias, and invite three undergraduates to annotate the entities IOB (short for Inside, Outside, Beginning) format. The annotated data set is divided into a training set and test set according to the ratio of 9:1. To decide the best method for this task, we employ four NER models based on CRF [14], BERT+BiLSTM+CRF [15], SpERT [16], NcrfPP [10] to evaluate their performances.

Table 2 lists the results of models in the NER task. We observe that NcrfPP obtains the performance than others in terms of best precision and F1-measure. Therefore, we employ it to provide entities for relation extraction.

Table 2: The results of models in the named entity recognition task

Models	Precision	Recall	F1-measure
CRF [14]	0.66	0.67	0.66
BERT+BiLSTM+CRF [15]	0.76	0.81	0.79
SpERT [16]	0.72	0.64	0.68
NcrfPP [10]	0.82	0.76	0.79

Evaluation of relation extraction After selected the model of NER task, we utilize the techniques of relation extraction to obtain the semantic relations among entities, including *Headquarter*, *Copyright*, *publishedBy*, *developedBy*, *appliedPlatform*, *publishedTime*, *userNumber*, *interactionMode*, *functionPoint* For the data set, we further invite three undergraduates to annotate the relations in above 3676 text descriptions of apps. The annotated data set is divided into the training set and test set according to the ratio of 9:1. To select the best model for this task, we employ OpenNRE[19] and DeepKE[20] to evaluate performances of relation extraction models, which are two famous platforms that integrated promising relation extraction models.

Table 3 lists the results of relation extraction models. We observe that relation extraction based on CNN [17] from DeepKE obtains the performance than others in terms of best precision and F1-measure. Therefore, we employ it to extract structured triples from textual descriptions of apps.

Table 3: The results of relation extraction models

Platforms	Model	Precision	Recall	F1-measure
OpenNRE	BERT [18]	0.94	0.88	0.91
OpenNRE	PCNN [19]	0.62	0.54	0.58
DeepKE	CNN [17]	0.97	0.95	0.96
DeepKE	Bi-LSTM [20]	0.91	0.80	0.84
DeepKE	Capsule [21]	0.96	0.93	0.94

As the several labeled relation is few, we also try to utilize few-shot relation extraction models from FewRel[21] platform to achieve our goal. In Table 4, we discover that combining BERT and Siamese models can also be competitive with several transitional ones for relation extraction tasks. For few annotated relations such as *userNumber*, *publishedTime*, some few-shot models of relation extraction can obtain better than transitional ones.

Table 4: The results of few-shot relation extraction models

Models	Proto [22]	Siamese [23]	GNN [24]	Snail [25]	MLMAN [26]	BERT+Siamese [27]
Accuracy	0.66	0.65	0.52	0.38	0.49	0.89

Finally, we totally obtain 17310, 6933, 3255, 2153 structured triples based on our selected relations from *Huawei App Market*, *Xiaomi App Store*, *Google Play*, *App Store*, respectively.

[19] https://github.com/thunlp/OpenNRE
[20] https://github.com/zjunlp/deepke
[21] https://github.com/thunlp/FewRel

4.3 Evaluation of knowledge alignment

In this section, we evaluate the rule miner method and KG embedding-based methods for knowledge alignment, which can share and reuse the description information of apps so as to provide better services based on MAKG for security analysts and users.

To evaluate the effectiveness of models, we select some apps from three application markets and build an alignment benchmark data set listed in Table 5.[22] The equivalent entities compose of apps and their tail entities. The former is roughly obtained by the names of installation package because all of these apps are developed based on Android systems. The latter is simply obtained based string matching method. Notice that the apps in Google Play are multilingual, we also translate the description information of apps into Chinese for knowledge alignment.

Table 5: The statistic of evaluating data set for knowledge alignment

Market	♯ APP	♯ entities	♯ triples	♯ equivalent entities in Huawei App Market	♯ equivalent entities in Xiaomi App Store	♯ equivalent entities in Google Play
Huawei App Market	43287	70954	1384854	–	27397	1526
Xiaomi App Store	27694	47478	384648	27397	–	979
Google Play	273802	382358	5210417	1526	979	–

Table 6 and Table 7 list results of the rule miner method [12] and three states of the art KG-embedding methods (i.e., MultiKE [28], RDGCN [29], NMN [30]) for knowledge alignment[23]. From these two tables, we can observe that:

- Rule miner method can finish all the tasks and cost about 1 minute for each task. KG-embedding methods are limited by memory so that some tasks have to reduce the scale of data sets.
- As the equivalent entities from Google Play to other application markets are not enough, the performances of all the alignment models are not very well.
- The whole performances of the rule miner method are better than KG embedding-based methods, especially for finding the equivalent apps from different markets. We analyze that the sparse contexts of apps may be one of the important reasons to limit the power of KG embedding-based methods.

Table 6: The results of the rule miner method

Alignment task	♯ Num	♯ Correct	Precision	Recall	F1-measure	Time (s)
Xiaomi App Store-Huawei App Market	25715	24639	0.958	0.899	0.928	66.9
Xiaomi App Store-Google Play	1036	584	0.564	0.597	0.580	49.8
Huawei App Market-Google Play	1128	898	0.796	0.589	0.677	55.8

[22] As the number of App Store is fewer than other application markets, we do not employ it.

[23] As the outputs of two kinds of models are different, we lists their results in two tables.

Table 7: The results of KG-embedding methods

Alignment task	♯ entities	Model	hit@1	hit@5	hit@10
Xiaomi App Store-Huawei App Market	47478-70954	MultiKE	84.59	90.37	91.60
		RDGCN	71.11	87.45	89.16
		NMN	76.87	87.83	89.74
Xiaomi App Store-Google Play	47478-382358	MultiKE	40.77	44.60	45.30
	47478-16456	RDGCN	31.76	44.02	49.73
	47478-16456	NMN	32.94	44.65	52.63
Huawei App Market-Google Play	70954-382358	MultiKE	36.89	42.67	45.78
	70954-16456	RDGCN	25.24	34.29	37.32
	70954-16456	NMN	29.40	38.10	41.85

5 Use-Cases

We list three use-cases based on MAKG (shown in Fig. 1) about cybersecurity

Firstly, it can achieve semantic retrieval based on MAKG. For example, if one user queries one app, MAKG can present comprehensive to the user. Moreover, we can further employ entity linking techniques [31] to link the apps to their appearing textual descriptions. Benefited from above cases, users can fully understand the information of apps and avoid downloading some invalid apps.

Secondly, it can help security analysts to detect some sensitive apps, which own more conditions or plausibility than normal apps that become the hotbeds for related cybercriminals [32]. With the help of comprehensive relations and properties, analysts can define some prior rules or employ more promising algorithms to evaluate the sensitivity and rank them. It can prevent the risk of some sensitive apps in advance.

Thirdly, it can recommend some similar apps for users and security analysts when they request one service. Similarly, if some suitable algorithms [33, 34] are utilized for the recommendation, it can also reduce the potential risks and maintain the security of the mobile internet.

6 Conclusion

In this resource paper, we presented a mobile application knowledge graph, namely MAKG. Our work is to collect apps from application markets and external resources. Then, lots of structured triples are obtained by information extraction techniques according to our designed ontology for enriching MAKG. In addition, we further employ knowledge alignment models to generate the correspondences among apps from different application markets in order to share and reuse the description information of apps, which can provide better services based on MAKG for security analysts and users. The result is a high-quality mobile application dataset, which provides an open data resource to the researchers from The Semantic Web and CyberSecurity.

As future work, we plan to broaden the apps and enrich their structured information, so that MAKG becomes more comprehensive and covers more topics. In addition, we try to implement the mentioned use-cases and explore promising algorithms to provide better services for security analysts and users.

References

1. Guozhu Meng, Matthew Patrick, Yinxing Xue, Yang Liu, and Jie Zhang. Securing Android App Markets via Modeling and Predicting Malware Spread Between Markets. *IEEE Trans. Information Forensics and Security*, 14(7):1944–1959, 2019.
2. Franz-Xaver Geiger and Ivano Malavolta. Datasets of Android Applications: a Literature Review. *CoRR*, abs/1809.10069, 2018.
3. 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. *CoRR*, abs/1709.05281, 2017.
4. 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.
5. Daniel Arp, Michael Spreitzenbarth, Malte Hubner, Hugo Gascon, and Konrad Rieck. DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket. In *NDSS*, 2014.
6. Xin-Yu Yang and XU Guo-Ai. Construction method on mobile application security ecological chain {J}. *Journal of Software*, 28(11):3058–3071, 2017.
7. Elmar Kiesling, Andreas Ekelhart, Kabul Kurniawan, and Fajar Ekaputra. The sepses knowledge graph: an integrated resource for cybersecurity. In *ISWC*, pages 198–214, 2019.
8. Daniel E. Krutz, Mehdi Mirakhorli, Samuel A. Malachowsky, Andres Ruiz, Jacob Peterson, Andrew Filipski, and Jared Smith. A dataset of open-source android applications. In *MSR*, pages 522–525, 2015.
9. Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. A survey on deep learning for named entity recognition. *CoRR*, abs/1812.09449, 2018.
10. Jie Yang and Yue Zhang. NCRF++: an open-source neural sequence labeling toolkit. In *ACL*, pages 74–79, 2018.
11. Meiji Cui, Li Li, Zhihong Wang, and Mingyu You. A survey on relation extraction. In *CCKS*, pages 50–58, 2017.
12. Xing Niu, Shu Rong, Haofen Wang, and Yong Yu. An effective rule miner for instance matching in a web of data. In *CIKM*, pages 1085–1094, 2012.
13. Zequn Sun, Qingheng Zhang, Wei Hu, Chengming Wang, Muhao Chen, Farahnaz Akrami, and Chengkai Li. A benchmarking study of embedding-based entity alignment for knowledge graphs. *Proc. VLDB Endow.*, 13(11):2326–2340, 2020.
14. Andrew McCallum and Wei Li. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *NAACL*, pages 188–191, 2003.
15. Liyuan Liu, Jingbo Shang, Xiang Ren, Frank Fangzheng Xu, Huan Gui, Jian Peng, and Jiawei Han. Empower sequence labeling with task-aware neural language model. In *AAAI*, pages 5253–5260, 2018.
16. Markus Eberts and Adrian Ulges. Span-based joint entity and relation extraction with transformer pre-training. In *ECAI*, pages 2006–2013, 2020.
17. Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. Relation classification via convolutional deep neural network. In *COLING*, pages 2335–2344, 2014.
18. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*, pages 4171–4186, 2019.
19. Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. Distant supervision for relation extraction via piecewise convolutional neural networks. In *EMNLP*, pages 1753–1762, 2015.
20. Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. Attention-based bidirectional long short-term memory networks for relation classification. In *ACL*, 2016.
21. Ningyu Zhang, Shumin Deng, Zhanling Sun, Xi Chen, Wei Zhang, and Huajun Chen. Attention-based capsule networks with dynamic routing for relation extraction. In *EMNLP*, pages 986–992, 2018.

22. Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical networks for few-shot learning. In *NIPS*, pages 4077–4087, 2017.
23. Xu Han, Hao Zhu, Pengfei Yu, Ziyun Wang, Yuan Yao, Zhiyuan Liu, and Maosong Sun. Fewrel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation. In *EMNLP*, pages 4803–4809, 2018.
24. Victor Garcia Satorras and Joan Bruna Estrach. Few-shot learning with graph neural networks. In *ICLR*, 2018.
25. Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. In *ICLR*, 2018.
26. Zhi-Xiu Ye and Zhen-Hua Ling. Multi-level matching and aggregation network for few-shot relation classification. In *ACL*, pages 2872–2881, 2019.
27. Tianyu Gao, Xu Han, Hao Zhu, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. Fewrel 2.0: Towards more challenging few-shot relation classification. In *EMNLP-IJCNLP*, pages 6249–6254, 2019.
28. Qingheng Zhang andZequn Sun, Wei Hu, Muhao Chen, Lingbing Guo, and Yuzhong Qu. Multi-view knowledge graph embedding for entity alignment. In *IJCAI*, pages 5429–5435, 2019.
29. Yuting Wu, Xiao Liu, Yansong Feng, Zheng Wang, Rui Yan, and Dongyan Zhao. Relation-aware entity alignment for heterogeneous knowledge graphs. In *IJCAI*, pages 5278–5284, 2019.
30. Yuting Wu, Xiao Liu, Yansong Feng, Zheng Wang, and Dongyan Zhao. Neighborhood matching network for entity alignment. In *ACL*, pages 6477–6487, 2020.
31. Priya Radhakrishnan, Partha Talukdar, and Vasudeva Varma. Elden: Improved entity linking using densified knowledge graphs. In *NAACL*, pages 1844–1853, 2018.
32. Weizhuo Li, Buye Zhang, Liang Xu, Meng Wang, Anyuan Luo, and Yan Niu. Combining knowledge graph embedding and network embedding for detecting similar mobile applications. In *NLPCC*, pages 256–269, 2020.
33. Ning Chen, Steven C. Hoi, Shaohua Li, and Xiaokui Xiao. SimApp: A Framework for Detecting Similar Mobile Applications by Online Kernel Learning. In *WSDM*, pages 305–314, 2015.
34. Da Cao, Xiangnan He, Liqiang Nie, Xiaochi Wei, Xia Hu, Shunxiang Wu, and Tat-Seng Chua. Cross-platform app recommendation by jointly modeling ratings and texts. *ACM Transactions on Information Systems (TOIS)*, 35(4):1–27, 2017.