Mobile Networks and Applications (2023) 28:1901–1912
https://doi.org/10.1007/s11036-023-02159-x

APKOWL: An Automatic Approach to Enhance the Malware Detection

Doaa Aboshady[1] · Naglaa E. Ghannam[2] · Eman K. Elsayed[2,3] · L. S. Diab[2,4]

Accepted: 29 June 2023 / Published online: 17 July 2023

Abstract
Malicious software (malware) can steal passwords, leak details, and generally cause havoc with users' accounts. Most of the current malware detection techniques are designed to detect malware at the code level of the software, where it is actually infected and causes damage. Additionally, current malware detection techniques at the design level are done manually or semi-automatically. This research aims to enhance these methods to detect malware at the design level automatically with a big dataset. The proposed method presents an automatic system for detecting SMS (Short Message Service) malware at the design which is called APKOWL. It is based on reverse engineering of the mobile application and then automatically builds OWL (web ontology Language) ontology. The proposed system is implemented in python and Protégé, and its performance has been tested and evaluated on samples of android mobile applications including 3,904 malware and 3,200 benign samples. The experimental results successfully verify the effectiveness of the proposed method because it has good performance in detecting SMS malware at the software design level. The proposed method obtained an accuracy of 97%, precision of 97.5%, and recall of 99%, outperforming the compared model in all performance metrics.

Keywords Malware detection · OWL ontology · Mobile application · Reverse engineering · Software Quality

1 Introduction

Android is the largest operating system (OS) globally and among the different types of mobile OS, its a broadly used OS in mobile devices. Due to its open-source nature, it can be reverse-engineered and repackaged. With the increase of apps in the Google play market and third-party markets, apps have become a target for developers/users and attackers. App developers develop the apps and upload benign apps into the official markets. In the same way, attackers and hackers repack benign apps with malicious code and upload them into the app market which can cause damage in devices and perform various malicious activities. This means that hackers may convert benign programs into malicious programs, and the user may download them and be exposed to many problems, such as stealing accounts and sensitive data, poor hardware performance, etc.

Malware is intrusive software designed with malicious intent to harm mobile devices and collect sensitive data without the user's knowledge such as online accounts, credentials, etc. The common malware types include Trojan horses, spyware, viruses, worms, adware, ransomware, or any type of malicious code that penetrates a device [1]. Many empirical studies have assessed the negative impact of malware on performance [2, 3], reliability [4], energy consumption [5], and other quality elements [6]. Most of the current methods for detecting malware, such as [2, 3], are looking for more effective solutions for detecting malware, due to its great impact on the performancen [4], the authors showed that the tools used to determine software correctness, reliability, and security from malware are expensive in terms of time and at the cost of high-quality results and this makes it difficult to analyze programs. The method aims to adjust and improve the basic tools for detecting and

✉ Doaa Aboshady
doaa_aboshady@science.tanta.edu.eg

1 Mathematics Department, Faculty of Science, Tanta University, Tanta, Egypt

2 Mathematics Department, Faculty of Science, Al-Azhar University (Girls branch), Cairo, Egypt

3 School of Computer Science, Canadian International College CIC, Cairo Governorate, Egypt

4 Department of Mathematics & Statistics, College of Science 323, King Abdullah City for Female Students, Imam Mohammad Ibn Saud Islamic University, Riyadh, Saudi Arabia

classifying malware. In work [5], this approach proved that power consumption increase in a smartphone device after its being infected by malware. Because malware taps such power-consuming functions of a device as usage of GPS, Wi-Fi, and mobile Internet. It provided the comparison of the mobile power consumption level in the case of the normal uninfected condition to the one in the case of inputting a spy malware. Additionally, reference [6] suggested that malware writers tend to improve user experience and increase their malware performances, probably for 2 reasons: limiting the risk that the malware execution is obstructed and avoiding disturbing the victim to better hiding the malware behavior.

There are different levels in the software development lifecycle like requirements, design, coding development, and execution. The design level is a stage before the coding stage, which shows the model of the application using any modeling tool; hence, this will avoid time-consuming use of patterns of applications from the Internet as new versions containing malware. In most of the previous work, malware detection is done at the source code level, which is considered late because the malware will have already damaged the system. According to [7], the malware can detect early at the design level of software by designing an OWL ontology model that simulates the structure of malware. This method detects malware in a semi-automatic way, so it was tested on a small dataset. Data scarcity can be a problem in domains where data collection is difficult, time-consuming, and/or expensive. Analyzing small data sets and building statistical models using them are both challenging tasks. Data fusion methods are a step forward towards a better understanding of data by bringing multiple data observations together to increase the consistency of the information [8, 9].

This paper introduces a fully automatic method for malware detection at the design level before coding and installation. The proposed method is based on reverse engineering and building an ontology model that simulated the structure of the malware. In addition, this paper avoids the limitations of the work [7] regarding the data preprocessing and converting stages that were performed manually which resulted in the use of a small number of samples to evaluate the method.

The main contributions of this research can be summarized as follows:

- A novel automatic method (APKOWL) for detecting malware at the design level using big data to evaluate the proposed system. This method helps to enhance the performance of detecting android malware.
- Apply the suggested method to the SMS malware category which is one of the most important malware families.
- Provide a comparison between the work in [6] and the proposed APKOWL method for detecting malware in terms of AUC, precision, recall, F1-measure, true positive rate (TPR), and false positive rate (FPR).

The rest of this paper is structured as follows: Sect. 2 introduces a related work to malware detection based on machine learning and detection based on ontology. Section 3 proposes a proposed methodology for detecting malware. Section 4 describes the experimental results and analysis. Section 5 presents the conclusion and future work.

2 Related Work

Data security focused in this paper has been studied for decades and there are currently several related literatures [10–13]. Malware detection is one of the active data security research fields. Numerous static and dynamic analysis techniques have been presented by the scholarly community to detect and classify malware [14–19]. Both the techniques, static and dynamic have their benefits and limitations. So, several studies have used ensemble machine learning techniques to detect and classify malware [20– 25]. Additionally, ontology has been employed to describe the behavior of malware and build a knowledge representation of malware to achieve efficient identification. Based on the subject of this paper, this section briefly summarizes the related state-of-the-art techniques of malware detection using machine learning and ontology-based malware analysis. The existing approaches are summarized and shown in Table 1.

2.1 Detection Based on Machine Learning

In a study [26], authors have proposed a scalable malware detection system using big data and a machine learning approach. They used Locality-sensitive hashing and achieved 99.8% accuracy. In [27] research paper, authors have proposed BigRC-EML for ransomware detection and classification based on several static and dynamic features. They applied ensemble machine learning methods on a big data to enhance the accuracy of ransomware detection. The classification models used are SVM, Random Forests, KNN, XGBoost, and Neural Network. In the study [28], the authors have designed two methods for building an ensemble classifier which is based on calculating ranks and weights for the base classifiers. This proposed scheme based on weighted voting applied on a big dataset and achieved 99.5% accuracy. In [29], the study presented an EfficientNet-B4 CNN-based method for Android malware detection. Another work presented a novel method for CNN model explainability in Android malware detection [30].

Table 1 A summary of existing related work on malware detection

Malware detection technique	Ref.	Description	Target	Data set	
				benign	**malware**
Machine learning techniques	[26]	a scalable malware detection system with big data using Locality-sensitive hashing, which significantly reduces the malware detection time. A five-stage iterative process has been used to carry out the implementation and experimental analysis.	malware of different families.	2,128 benign files from various apps and utility software like Media Player and Adobe Reader	15,952 Malware from VirsusShare,VirusTotal and theZoo.
	[27]	a hybrid ransomware detection method that analyzes image data, text, and application code to extract plain or encrypted threat text.	ransomware	-	-
	[28]	Two ensemble methods are proposed which are based on the algorithms for computing the ranks and weights of the base classifiers (used in weighted voting method), and selecting a set of base classifiers (used in stacking).	malware of different families.	98,150 benign from a clean Windows XP, 7, and 10 installations directory.	100,200 malware from diverse sources such as VirusShare, Nothink, VXHeaven,
	[29]	This model is proposed to detect Android malware using image-based malware representations of the Android DEX file. Its extracts relevant features then passed it through a global average pooling layer and fed into a softmax classifier.	Malware such as Trojan, AdWare, Clicker, SMS, Spy, Ransom, Banker, and others.	2486 sample from R2-D2 dataset [36].	3500 sample from R2-D2 data set [39].
	[30]	This work presented a method to identify locations deemed important by CNN in an Android app's opcode sequence which appears to contribute to malware detection.	malware of different families.	5,560 sample	5,560 malware from Drebin dataset [40].
	[31]	an Android malware detection system based on permission features using Bayesian classification. The permission features were extracted via the static analysis technique.	20 malware families	5,000 applications from AndroZoo,	5,000 applications from the Drebin dataset [40].
	[32]	TC-Droid, an automatic framework for Android malware detection based on text classification method.	different types of malware such as Trojans, Backdoors, Ransomware, and Virus.	Dataset1: 1,270 Dataset2: 2,515 Dataset3: 3,469 from Anzhi	Dataset1: 1,176 Dataset2: 2,517 Dataset3: 3,219 from the Contagio and Community and Android Malware Genome Project [41]

Table 1 (continued)

Ontology techniques	[33]	an effective Ontology-based intelligent model based on app permissions to detect malware apps. it extracts the relationship between the static features from the apps and builds an Apps Feature Ontology (AFO).	Different malware types	1,959 applications	2,113 applications
	[34]	APT malware detection and cognition framework called APTMalInsight used to identify and cognize APT malware by leveraging system call information and ontology knowledge.	Different malware types	1180 sample from the Windows 7 OS platform	864 sample from China Information Security Evaluation and Certification Center and VirusShare
	[35]	an ontology for dynamic analysis of malware behaviour and to capture metamorphic and polymorphic malware behaviour.	14 malware families with their sub-families	-	-
	[36]	a novel description and derivation model of Android malware intention based on the theory of intention and malware reverse engineering	Ransomware and Genome's	-	75 typical malware samples from Zhou and Jiang [41] and DroidBench samples [42]
	[37]	An Ontology based model for preventing and detecting SQL Injection Attacks (SQLIA) using ontology which implements Ontology Creation and prediction rule based vulnerabilities model.	SQL Injection Attacks (SQLIA)	-	-
	[38]	method for constructing a malware knowledge base and designing the concept classes and object properties of malware.	Eight malware families (Allaple, Mydoom, Mytop, Bagle, Netsky, Agobot, Apaj, Zbot)	2,229 sample from several desktop workstations that run Windows 7	2,229 sample from the website [43].

The research in [31] showed an Android malware detection system based on permission features using Bayesian classification. These permission features were extracted via the static analysis technique. Also in [32], the authors proposed TC-Droid, an automatic framework for Android malware detection based on text classification method.

2.2 Detection Based on Ontology

Ontology can objectively describe things in the real world, so it is often used to represent the knowledge of a domain. Ontology has been applied in many fields such as web semantics, knowledge engineering, artificial intelligence, and so on. In addition, ontology has been used to describe the knowledge of malware in the field of information security.

The study in [33] proposed a model called malicious android applications detection using ontology-based intelligent based on app permissions to detect malware. In [34], the authors proposed an effective method to identify and cognize APT malware by leveraging system call information and ontology knowledge. They leveraged the ontology model to describe the behavioral characteristics of APT malware and construct the ontological knowledge framework for APT malware. The study [35] proposed an ontology driven framework that captures recent malware behaviours. This ontology structure divided Malware in two sub-categories: malware families and malware code-structure and introduced a detailed understanding of different malware families with their behaviour. In [36] authors used the ontology to model the semantic relationship between behaviors and objects of android malware intention and automate the process of intention derivation using SWRL rules and Jess engine. The work in [37] proposed an ontology based model for preventing and detecting SQL Injection Attacks (SQLIA) using ontology (SQLIO). This method provided prevents and detects SQLIA web vulnerability to a greater extent in a cloud environment. In addition, the study in [38] introduced the ontology method into the malware detection domain, used ontology to describe malware behavior and construct the knowledge framework of malware, and used ontology reasoning technology to identify families of unknown malware.

As noted, most of these studies detected malware by tracing the software's source code. Additionally, no works have studied the possibility of malware detection at the design stage of the software. Based on the work in [6], which proved that it is possible to detect malware at the design stage using ontology and reverse engineering. As this method achieved satisfying results, in this paper we present an improvement for this ontology method by applying it automatically with big data.

3 The Proposed Methodology

This section discusses the detailed APKOWL methodology used to detect malware based on reverse engineering and design ontology model for infected android applications. Figure 1 presents an overview of the APKOWL detection system architecture and the subsequent subsections describe its various phases in detail.

3.1 The Preprocessing Phase (Dataset Preparation)

It is important to note the basic android application structure. The application programming kit (apk) files are similar to a compressed (zip) or archived file, that contains several important resource files. The five main components of an apk file are; METF − INF, res, Classes.dex, resources.arsc, and AndroidManifest.xml. Classes.dex is an executable file that contains the Dalvik bytecode that can be executed on the Dalvik Virtual Machine (DVM). AndroidManifest.xml is an XML file that stores the meta-data such as package name, permissions required, definitions of one or more components (i.e., activities, services, broadcast receivers, or content providers), min and max version support, linked libraries, etc. it describes the application metadata and how it connects and interacts with the Android system.

To prevent being decompiled, Android malware applications often used software packers to protect themselves. After decompilation, the only available file is the AndroidManifest.xml file. Therefore, when detecting malware applications, the researchers can only rely on the features that provided by the AndroidManifest.xml file, including intent filters and permissions. At this phase, the decompilation process of the apk files is done automatically using apkx converter [44]. The decompilation process includes reverse engineering of the apk files to extract its components and find the AndroidManifest.xml files.

3.2 OWL Ontology Design Phase

An ontology broadly describes concepts within a domain through classes and properties. These properties include property between defined classes and their attributes. An ontology is usually designed around a few main classes that cover the domain whose scope has been predefined. These classes may have sub-classes (more specific), and super-classes (more general). The relationship among classes determines the kind of interaction between them. Instances are individual instantiated examples of a class, which means each instance can have different values for data properties and be connected to other instances via object properties. Using ontology techniques has contributed to many works related to the detection of malware, as we explained in Sect. 2.

At this phase, an ontology model is generated automatically in OWL format for each XML file extracted in the previous phase. The ontmalizer converter [45, 46] is used to transform XML file to OWL ontology. It is an open-source tool that converts XML data to RDF and XSD schemas to ontologies.

3.3 Malware Detection Phase

This work aims to detect malicious programs early in the design stage of the software before it is installed, and not at the code stage like other current works. Although there are different types of malware, so far only infected applications can be detected at the design stage. Because it is controlled by the user and it is possible to detect malware before the infected application is installed. Also, some malware such as worms infected the device through spam emails. which were not controlled by the user. Therefore, not all malware types can be detected at the design level.

After generate the OWL ontology model, APKOWL can detect the malware files by running certain SPARQL queries on the ontology. SPARQL is a semantic (RDF) query language for databases and its able to retrieve and manipulate data stored in Resource Description Framework format. The knowledge is retrieved from an ontology graph using structured queries. SPARQL utilizes the relationship to navigate any linked data quickly. So, queries can traverse

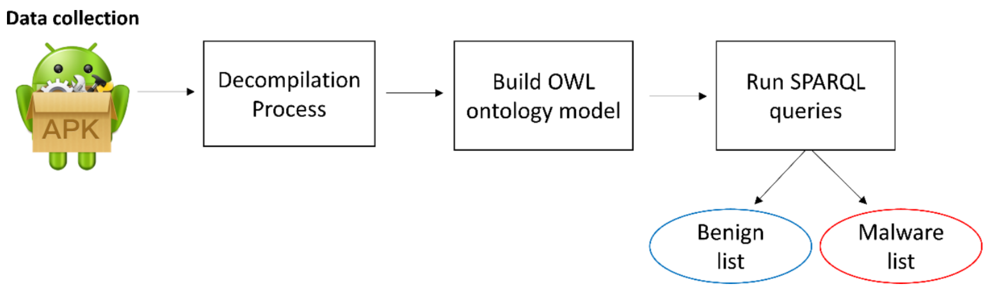


Fig. 1 APKOWL system architecture for malware detection

across multiple RDF at once. In the proposed APKOWL method, SPARQL queries are defined based on the malware behavior. we have chosen the SMS malware to evaluate the system and build a SPARQL query for them according to their behavior as shown in subsection 4.2. besides, SMS malware can be detected at the design stage of infected software. APKOWL showed acceptable results for SMS malware detection and it can be used for detecting other types of Android malware.

4 Experiments and Results

4.1 Dataset Description

The proposed ontology method for malware detection is evaluated on a dataset of 3,904 malware and 3,200 benign apk files. The benign file means that the application is not malware. Totally, there were 7,104 applications in our dataset. The malware and benign files are collected from the CICMalDroid 2020 dataset presented by the Canadian Institute for Cybersecurity. The CICMalDroid 2020 dataset has 11,598 android samples with five distinct categories (Adware, Banking, SMS malware, Riskware, and Benign). This dataset can be found in [47]. Some benign samples are collected from Google Play using a crawler that randomly searched in the dictionary file and downloads top applications from the found results. For balance, our dataset contains nearly amount of malware and benign files.

In this work, we choose SMS malware category from the CICMalDroid 2020 dataset to apply the proposed method, the following describes the SMS malware.

SMS Malware The android malware used in this research is SMS malware. Mobile SMS is one of the most used functions on a mobile device. SMS is a text messaging service that can be used by any messaging application on phones or computers. Most of the time, SMS malware is active from the SMS to the mobile device. the SMS malware intercepts the SMS payload to conduct attacks through the SMS service. First, the hackers upload malware to their hosting sites to be connected with the SMS. Second, they use the command-and-control server for controlling the attack instructions, i.e., send malware SMS then intercept SMS and finally steal the user data. For example, the user received a text message from somebody they didn't know or an SMS requesting they click on an URL link of an infected application to download or dial a specific number to get a voicemail. All of these techniques are tricks the hackers can use to insert malware onto the mobile or try to trick users into handing over personal data.

4.2 Implementation

The APKOWL method for malware detection was designed and successfully implemented. PyCharm 2021.3 was used to implement the system architecture. Google Colaboratory (Colab) Notebook was the IDE that has been used to design and implement the system architecture. Colab is a free research tool with two various models offered by Google, for research and machine learning education. It's a Jupyter notebook environment which requires no additional setup to employ. Code is written on browser interface and executed in a virtual machine dedicated to user account. It provides CPU, Graphical Processing Unit GPU, and Tensor Processing Unit TPU. The offered models of Colab are GPU NVIDIA K80/T4 for free usage and 100 GB for free space for data, and 16 GB of RAM for which can provide high-performance computing tasks. In addition, it supports Linux and Windows operating system commands [48]. The data set used in this architecture to achieve the goal is described in the previous section.

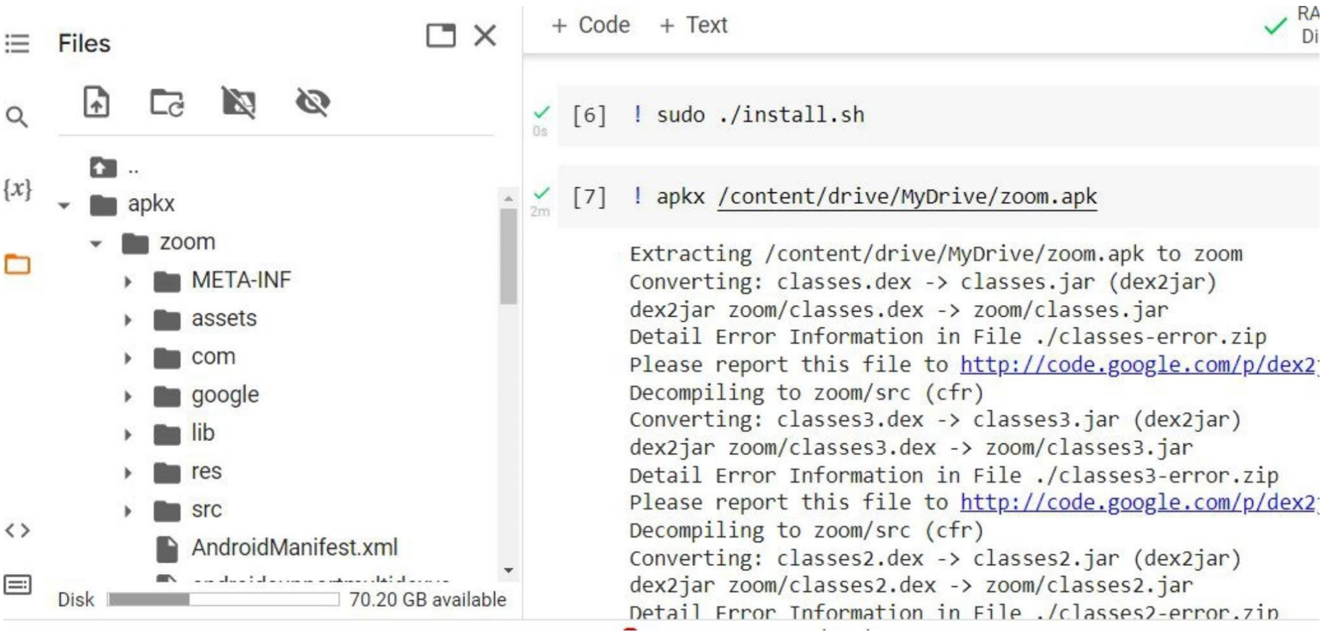
The detection process has done by converting each apk file in the dataset to an XML file using the apkx converter. Figure 2 shows the result of apkx converter for the zoom application as an example of the converting step. After that, used the ontmalizer converter to generate the OWL ontology. Then, the file is imported into the Protégé which is the editor of ontology. By applying certain SPARQL queries, the proposed method detects SMS malware.

We propose an OWL ontology model that represents the SMS malware as a case study for applying the SPARQL queries. The OWL model presented in Fig. 3 describes the construction of the SMS malware.

The model contains four main classes, which are (User, Application or SMS, Server, and Hacker). Every class has datatype properties that describe it and object properties as relations between the class and other classes. The main idea of this malware is based on the inverse property, which gets user information once he clicks the link to download the app. The "Download_Application" Property has inverse property of name "record_user _information" which implies that all users download the application their information is recorded automatically without any permission as in Fig. 4.

The domain of the "Download_Appliction" object property is the "User" class and the range is the "Application" class. The first step of the detection is getting all apps in the dataset that contains an inverse property. This is through using the next query Q1.

Q1. Retrieving the inverse properties query:



```
[6] ! sudo ./install.sh

[7] ! apkx /content/drive/MyDrive/zoom.apk

    Extracting /content/drive/MyDrive/zoom.apk to zoom
    Converting: classes.dex -> classes.jar (dex2jar)
    dex2jar zoom/classes.dex -> zoom/classes.jar
    Detail Error Information in File ./classes-error.zip
    Please report this file to http://code.google.com/p/dex2
    Decompiling to zoom/src (cfr)
    Converting: classes3.dex -> classes3.jar (dex2jar)
    dex2jar zoom/classes3.dex -> zoom/classes3.jar
    Detail Error Information in File ./classes3-error.zip
    Please report this file to http://code.google.com/p/dex2
    Decompiling to zoom/src (cfr)
    Converting: classes2.dex -> classes2.jar (dex2jar)
    dex2jar zoom/classes2.dex -> zoom/classes2.jar
    Detail Error Information in File ./classes2-error.zip
```

Fig. 2 The output of apkx converter

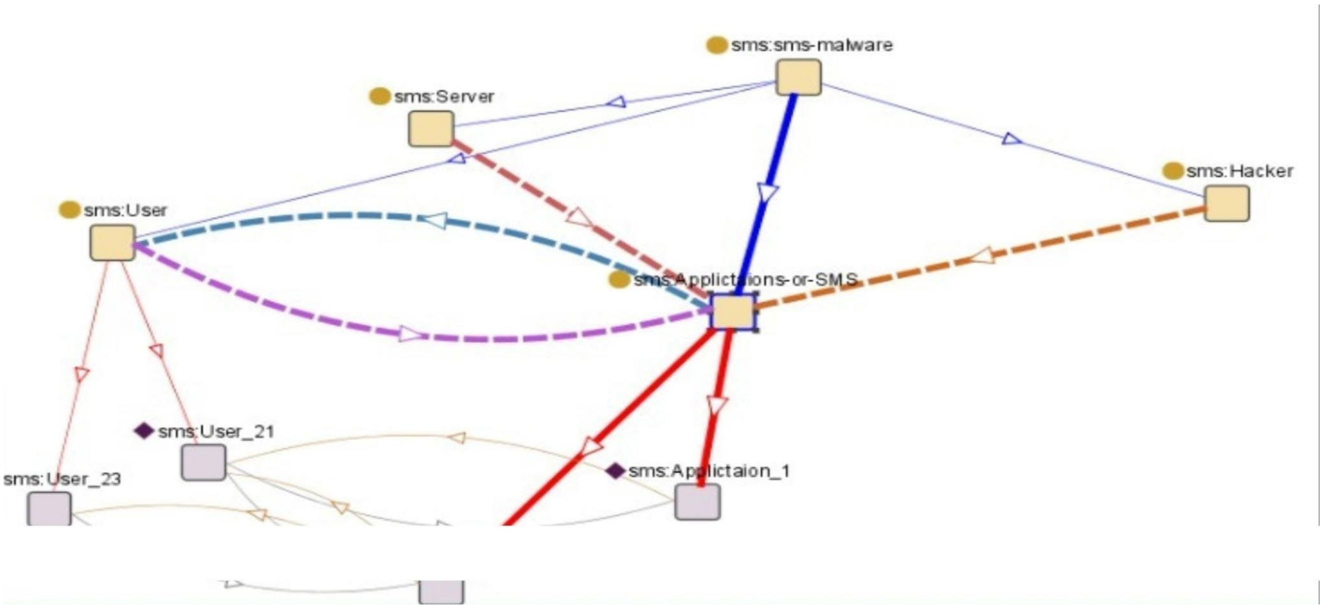
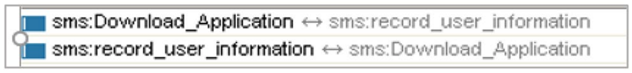


Fig. 3 The construction of SMS malware



sms:Download_Application ↔ sms:record_user_information
sms:record_user_information ↔ sms:Download_Application

Fig. 4 The inverse properties

SELECT ?property ?inverse.
WHERE{.
?inverse a owl:ObjectProperty.
?inverse ?v ?property.
?property ?v ?inverse.

}

The next step of the detection is ensuring that User information is recorded at every download process. By using query Q2, all User individuals appear and their personal information too.

Q2. The query of the downloaded applications and the Users.

SELECT ?Application ?User ?PeronalInformation.

Where{.
?Properties a owl:ObjectProperty.
? Properties ?n sms:Application-or-SMS.
?Application ?Properties ?Users.
?Y a owl:ObjectProperty.
?Users ?Y ?Application.
?Information a owl:DatatypeProperty.
?Users ?Information ?PersonalInformation.
}

The result of query Q2 is in Fig. 5.

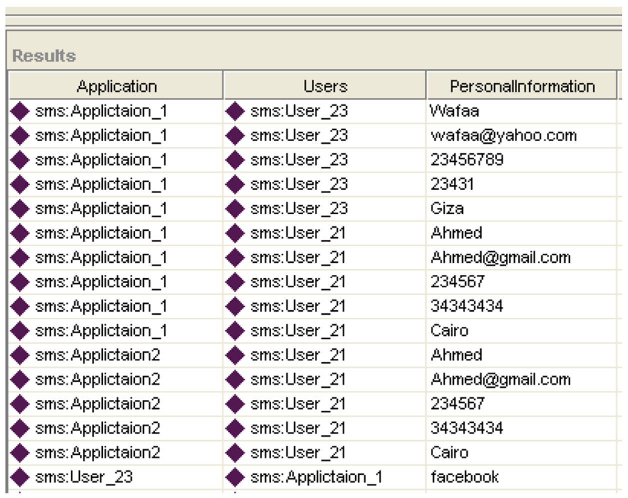
Now, it is clear that user information is available at the. application website that the hacker will receive easily.

4.3 Evaluation Measures

For evaluating the proposed method, we applied it on a dataset of 3,104 malware and 3,200 benign apk files. The performance of the proposed model was evaluated using a confusion matrix or also called an error matrix. A confusion matrix is a summary of prediction results by a classifier. It provides the values of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). In the context of this study, TP refers to the malware classified as malware, and TN indicates the benign identified as benign. FP gives the number of benign files misclassified as malware, and FN refers to the count of malware misclassified as benign.

Based on these values, the following four performance Metrics were computed:

Precision: It is a measure of quality and the model's ability to not predict a benign (negative) sample as malware (positive).



Results

Application	Users	PersonalInformation
sms:Applictaion_1	sms:User_23	Wafaa
sms:Applictaion_1	sms:User_23	wafaa@yahoo.com
sms:Applictaion_1	sms:User_23	23456789
sms:Applictaion_1	sms:User_23	23431
sms:Applictaion_1	sms:User_23	Giza
sms:Applictaion_1	sms:User_21	Ahmed
sms:Applictaion_1	sms:User_21	Ahmed@gmail.com
sms:Applictaion_1	sms:User_21	234567
sms:Applictaion_1	sms:User_21	34343434
sms:Applictaion_1	sms:User_21	Cairo
sms:Applictaion2	sms:User_21	Ahmed
sms:Applictaion2	sms:User_21	Ahmed@gmail.com
sms:Applictaion2	sms:User_21	234567
sms:Applictaion2	sms:User_21	34343434
sms:Applictaion2	sms:User_21	Cairo
sms:User_23	sms:Applictaion_1	facebook

Fig. 5 The result of query Q2

$$Precision = \frac{TP}{TP+FP} \tag{1}$$

Recall: It is a measure of the model's ability to predict all malware (Positive) samples as malware.

$$Recall = \frac{TP}{TP+FN} \tag{2}$$

F-score: It is the symmetric average of the values of precision and recall.

$$F-score = \frac{2*Recall*Precision}{Recall+Precision} \tag{3}$$

Accuracy: It is the percentage of correctly identified samples (both malware and benign).

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} \tag{4}$$

To further evaluate the quality of the proposed system results, we use the Receiver Operating Characteristic (ROC) metric. ROC plots false positive rate (FPR) on X axis, and true positive rate (TPR) on Y axis. TPR is the rate of correctly predicted malware samples and is also known as recall or sensitivity. FPR refers to the rate of benign (negative) samples that are incorrectly identified. The curve depicts a trade-off between TPR and FPR. TPR and FPR are defined as below:

$$TPR = \frac{TP}{TP+FN} \tag{5}$$

$$FPR = \frac{FP}{TN+FP} \tag{6}$$

Additionally, the region under the ROC curve is known as an area under curve (AUC) and its ranges in value from 0 to 1. The higher the AUC value or close to 1 implies better performance by the model.

5 Experimental Results and Discussion

This experiment is performed on Google CoLab using rdflib (Python RDF library for representing information) and lxml libraries. We apply APKOWL on 7,104 samples of android mobile applications. Colab has space limitations in addition to execution time limitations that was the reason that we prove the proposed detection method only on SMS malware type.

Table 2 The detection results of the APKOWL malware detection system

Malware category	Sample number	Results
SMS malware	3,904	3,785 times
Benign	3,200	3,319 times

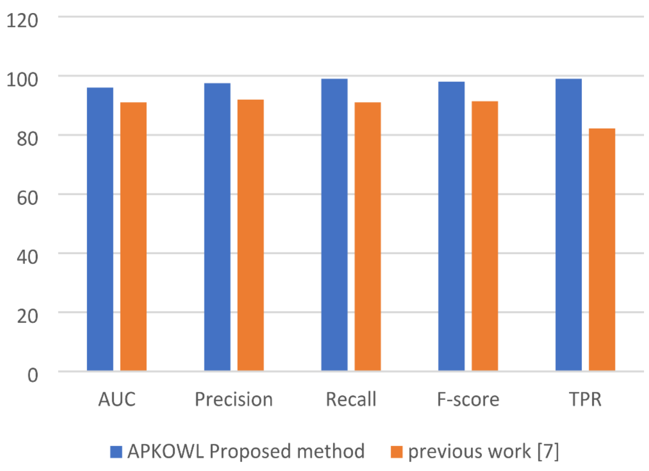


Fig. 7 Comparison of APKOWL and previous work [7]

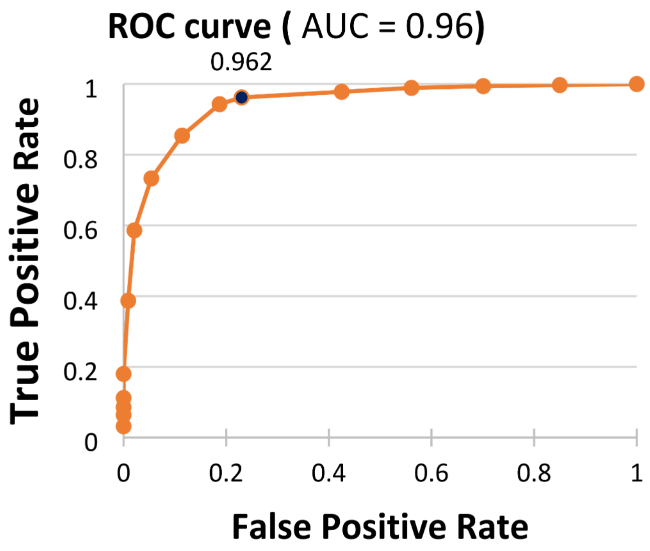


Fig. 6 The ROC curve of the APKOWL system

Table 2 presented the results of detection according to the application type. As noted, the number of correctly detected samples was 3,785 for SMS malware. The system was applied to the 3,200 benign files and 5090 of them were correctly identified by the system. The rest of the 110 benign files were identified as false positive (i.e., malware).

The perfect detection is when the TPR = 100% and FPR = 0%. In truth, it is impossible to achieve 100% accuracy of TP. However, with a larger amount of data, the analysis may possibly provide a near accuracy of the positive measurement. Table 3 depicts that the values of TPR and FPR are 0.99 and 0.02 for the proposed APKOWL model respectively. As shown in Table 3, the following results are produced: accuracy = 97%, precision = 97.5%, recall = 99% and F-score = 98%. We note that our proposed ontology method achieves excellent detection performance on the dataset.

Additionally, The ROC curve for the proposed model is given in Fig. 6. As noted in Fig. 6, the closeness of the ROC curve to the top-left corner indicates better performance by the proposed model and the AUC score obtained is 96%. This suggests that the quality of detection results by the proposed model is good.

Whereas the APKOWL proposed to detect Android malware at the design stage of the software, we compare it with the state-of-the-art Android malware detection method at the design stage [7]. The work [7], proposed a semi-automatic model based on reverse engineering and unified modeling language (UML) environment, and the ontology to detect Android malware at the design stage. Table 4 shows the evaluation results of this comparison in terms of AUC, Precision, Recall, F-score, TPR, and FPR. As noted in Table 4, the automatic method and the use of a larger number of data led to an improvement in the results and performance in detecting Android malware using the proposed APKOWL approach.

Figure 7 shows the evaluation metrics-based comparative analysis of the APKOWL proposed method with a recent state-of-the-art detection method at the design level. As can be seen from Fig. 7, the APKOWL method has achieved higher performance compared to the other method. The APKOWL yields 96% AUROC, 97.5% precision, 99% recall, 98% F-score, and 99% TPR.

Table 5 shows the accuracy of the proposed approach compared to recent Android malware detection approaches including malware type and the techniques used. These existing approaches detected the Android malware at the source

Table 3 Performance of the proposed APKOWL system

	Accuracy	Precision	Recall	F-score	TPR	FPR
APKOWL system	97%	97.5%	99%	98%	99%	2%

Table 4 Comparison of the previous method with the APKOWL method

	AUC	Precision	Recall	F-score	TPR	FPR
[7]	91%	92%	91%	91.4%	82.2	17.2
APKOWL	96%	97.5%	99%	98%	99%	2%

Table 5 comparison of the proposed method and state-of-the-art methods on Android malware detection

Method	Accuracy	Malware type	Techniques Used
Proposed method (APKOWL)	97%	SMS malware	Ontology
DroidMD [24]	95.5	Different malware types	Machine learning
MGOP-Droid [25]	96.35%	Different malware families	Machine learning
EfficientNet-B4 [29]	95.7%	Malware such as Trojan, AdWare, Clicker, SMS, Spy, Ransom, Banker, and others.	convolutional neural network (CNN)
Reference [31]	91.1	Different malware types	Bayesian probability
TC-Droid [32]	96.6%	Different types of Android malware such as Trojans, Backdoors, Ransomware, and Virus.	Deep Learning (DL)
Reference [33]	94.11	Different malware types	Ontology

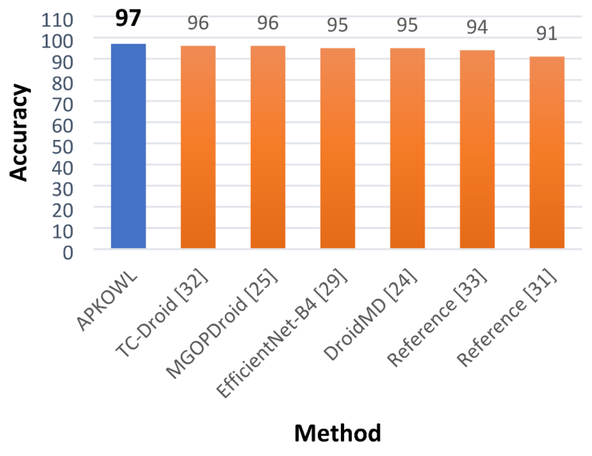


Fig. 8 Comparison of APKOWL with state-of-the-art existing detection methods

code level. As seen from Table 5, the APKOWL method scored the highest performance. It recorded an accuracy of 97% which exceeds that of other current Android detection methods [24, 25, 29, 31–33].

Figure 8 shows the accuracy-based comparative analysis of the APKOWL proposed method with recent state-of-the-art methods. As can be seen from Fig. 8, the APKOWL method has achieved higher accuracy compared to the other methods. It can be concluded that using a full dynamic ontology model with a large amount of data improved Android malware detection at the design level. Detecting malware at the design stage is better than detecting it at the source code stage in terms of software performance and security. The experimental results have demonstrated the superiority of the APKOWL suggested method compared to the current methods.

6 Conclusions and Future Work

In this paper, we introduce an APKOWL ontology technique to contribute to the malware detection process. This method depends on detecting SMS malware at the.

stage of software design. By doing reverse engineering for the infected application, extracting the XML file, generating ontology, and executing SPARQL queries, malware detection is done. Making all these steps automatic and using a larger number of data, we have achieved an improvement in results and performance over the latest work in this area. The proposed method achieves a high detection score based on the evaluation metrics. It records 96% of AUC score, 97% accuracy, 97.5% precision, 99% recall, 98% F-score, and 99% TPR.

Several future works could be extended from this work. First, the malware detection problem could be tackled early by designing an ontology model that simulates the structure of malware. Second, we aim to apply the proposed detection method to detect other types of malware. Third, we intend to use APKOWL for the identification of android malware types.

Funding Open access funding provided by The Science, Technology & Innovation Funding Authority (STDF) in cooperation with The Egyptian Knowledge Bank (EKB).

Data Availability Data will be made available on reasonable request.

References

1. Al-Marghilani A (2021) Comprehensive Analysis of IoT Malware Evasion techniques. Eng Technol Appl Sci Res 11(4):7495–7500
2. Darabian H, Dehghantanha A, Hashemi S, Taheri M, Azmoodeh A, Homayoun S, …, Parizi RM (2020) A multiview learning method for malware threat hunting: Windows, IoT and android as case studies. World Wide Web 23(2):1241–1260

3. Kadiyal a SP, Jadhav P, Lam SK, Srikanthan T (2020) Hardware performance counter-based fine-grained malware detection. ACM Trans Embedded Comput Syst (TECS) 19(5):1–17
4. Sebastio S, Baranov E, Biondi F, Decourbe O, Given-Wilson T, Legay A, …, Quilbeuf J (2020) Optimizing symbolic execution for malware behavior classification. Computers & Security 93:101775
5. Maevsky DA, Maevskaya EJ, Stetsuyk ED, Shapa LN (2017) Malicious software effect on the mobile devices power consumption. Green IT Engineering: components, networks and Systems implementation. Springer, Cham, pp 155–171
6. Mercaldo F, Di Sorbo A, Visaggio CA, Cimitile A, Martinelli F (2018) An exploratory study on the evolution of Android malware quality. J Software: Evol Process 30(11):e1978
7. Aboshady D, Ghannam N, Elsayed E, Diab L (2022) The Malware Detection Approach in the design of Mobile Applications. Symmetry 14(5):839
8. Wang, S., Celebi, M. E., Zhang, Y. D., Yu, X., Lu, S., Yao, X., … Tyukin, I. (2021).Advances in data preprocessing for biomedical data fusion: An overview of the methods,challenges, and prospects. *Information Fusion*, *76*, 376–421
9. Zhang YD, Dong Z, Wang SH, Yu X, Yao X, Zhou Q…, Gorriz JM (2020) Advances in multimodal data fusion in neuroimaging: overview, challenges, and novel orientation. Inform Fusion 64:149–187
10. Tang S, Huang S, Zheng C, Liu E, Zong C, Ding Y (2021) A novel cross-project software defect prediction algorithm based on transfer learning. Tsinghua Sci Technol 27(1):41–57
11. Sandhu AK (2021) Big data with cloud computing: discussions and challenges. Big Data Mining and Analytics 5(1):32–40
12. Wei D, Ning H, Shi F, Wan Y, Xu J, Yang S, Zhu L (2021) Dataflow management in the internet of things: sensing, control, and security. Tsinghua Sci Technol 26(6):918–930
13. Li F, Yu X, Ge R, Wang Y, Cui Y, Zhou H (2021) BCSE: Blockchain-based trusted service evaluation model over big data. Big Data Mining and Analytics 5(1):1–14
14. Abusitta A, Li MQ, Fung BC (2021) Malware classification and composition analysis: a survey of recent developments. J Inform Secur Appl 59:102828
15. Singh J, Thakur D, Gera T, Shah B, Abuhmed T, Ali F (2021) Classification and analysis of android malware images using feature fusion technique. IEEE Access 9:90102–90117
16. Reddy V, Kolli N, Balakrishnan N (2021) Malware detection and classification using community detection and social network analysis. J Comput Virol Hacking Techniques 17(4):333–346
17. da Costa, F. H., Medeiros, I., Menezes, T., da Silva, J. V., da Silva, I. L., Bonifácio,R., … Ribeiro, M. (2022). Exploring the use of static and dynamic analysis to improve the performance of the mining sandbox approach for android malware identification.*Journal of Systems and Software*, *183*, 111092
18. Chanajitt R, Pfahringer B, Gomes HM (2021), October Combining Static and Dynamic Analysis to Improve Machine Learning-based Malware Classification. In *2021 IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA)* (pp. 1–10). IEEE
19. Huang X, Ma L, Yang W, Zhong Y (2021) A method for windows malware detection based on deep learning. J Signal Process Syst 93(2):265–273
20. Wyrwinski P, Dutkiewicz J, Jedrzejek C (2020), October Ensemble malware classification using neural networks. In *International conference on multimedia communications, services and security* (pp. 125–138). Springer, Cham
21. Azeez NA, Odufuwa OE, Misra S, Oluranti J, Damaševičius R (2021, February) Windows PE malware detection using ensemble learning. Informatics, vol 8. MDPI, p 10. 1
22. Jain M, Andreopoulos W, Stamp M (2020) Convolutional neural networks and extreme learning machines for malware classification. J Comput Virol Hacking Techniques 16(3):229–244
23. Narayanan BN, Davuluru VSP (2020) Ensemble malware classification system using deep neural networks. Electronics 9(5):721
24. Akram J, Mumtaz M, Jabeen G, Luo P (2021) DroidMD: an efficient and scalable android malware detection approach at source code level. Int J Inf Comput Secur 15(2–3):299–321
25. Tang J, Li R, Jiang Y, Gu X, Li Y (2022) Android malware obfuscation variants detection method based on multi-granularity opcode features. Future Generation Computer Systems 129:141–151
26. Kumar M (2022) Scalable malware detection system using big data and distributed machine learning approach. Soft Comput 26(8):3987–4003
27. Aurangzeb S, Anwar H, Naeem MA, Aleem M (2022) BigRC-EML: big-data baseds structurethe malware' ransomware classification using ensemble machine learning. Cluster Comput, 1–18
28. Gupta D, Rani R (2020) Improving malware detection using big data and ensemble learning. Comput Electr Eng 86:106729
29. Yadav P, Menon N, Ravi V, Vishvanathan S, Pham TD (2022) EfficientNet convolutional neural networks-based Android malware detection. Computers & Security 115:102622
30. Kinkead M, Millar S, McLaughlin N, O'Kane P (2021) Towards explainable CNNs for android malware detection. Procedia Comput Sci 184:959–965
31. Mat SRT, Razak A, Kahar MF, Arif MNM, J. M., Firdaus A (2021) A bayesian probability model for android malware detection. ICT Express.
32. Zhang N, Tan YA, Yang C, Li Y (2021) Deep learning feature exploration for android malware detection. Appl Soft Comput 102:107069
33. OS JN (2021) Detection of malicious android applications using ontology-based intelligent model in mobile cloud environment. J Inform Secur Appl 58:102751
34. Han W, Xue J, Wang Y, Zhang F, Gao X (2021) APTMalInsight: identify and cognize APT malware based on system call information and ontology knowledge framework. Inf Sci 546:633–664
35. Chowdhury IR, Bhowmik D (2022), July Capturing Malware Behaviour with Ontology-based Knowledge Graphs. In *IEEE Conference on Dependable and Secure Computing (IEEE DSC 2022)*. IEEE
36. Jiao J, Liu Q, Chen X, Cao H (2018) Behavior Intention Derivation of Android Malware Using Ontology Inference. *Journal of Electrical and Computer Engineering*, *2018*
37. Durai KN, Subha R, Haldorai A (2021) A novel method to detect and prevent SQLIA using ontology to cloud web security. Wireless Pers Commun 117(4):2995–3014
38. Ding Y, Wu R, Zhang X (2019) Ontology-based knowledge representation for malware individuals and families. Computers & Security 87:101574
39. Hsien-De Huang T, Kao HY (2018), December R2-d2: Color-inspired convolutional neural network (cnn)-based android malware detections. In *2018 IEEE international conference on big data (big data)* (pp. 2633–2642). IEEE
40. Arp D, Spreitzenbarth M, Hubner M, Gascon H, Rieck K, Siemens CERT (2014), February Drebin: Effective and explainable detection of android malware in your pocket. In *Ndss* (Vol. 14, pp. 23–26)
41. Zhou Y, Jiang X (2012), May Dissecting android malware: Characterization and evolution. In *2012 IEEE symposium on security and privacy* (pp. 95–109). IEEE
42. Arzt, S., Rasthofer, S., Fritz, C., Bodden, E., Bartel, A., Klein, J., … McDaniel,P. (2014). Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps. *Acm Sigplan Notices*, *49*(6), 259–269

43. Malware L (2019) Malicious code samples available from: http://malware.lu (Accessed 20 May 2019)
44. Bernhard Mueller. [n.d.]. b-mueller/apkx: one-step APK decompilation with multiple backends. https://github.com/b-mueller/apkx
45. https://github.com/srdc/ontmalizer
46. https://github.com/rhizomik/redefer-xsd2owl
47. Mahdavifar S, Kadir AFA, Fatemi R, Alhadidi D, Ghorbani AADynamic Android Malware Category Classification using Semi-Supervised Deep Learning. In Proceedings of the 18th IEEE International Conference on Dependable, Autonomic, and, Computing S (DASC), Calgary, AB, Canada, 17–24 August 2020; Available online: https://www.unb.ca/cic/datasets/maldroid-2020.html (accessed on 10 March 2021)
48. Colab. https://colab.research.google.com/