Communications of the Association for Information Systems

Volume 46 Article 26

6-2020

The Paradox of Choice: Investigating Selection Strategies for Android Malware Datasets Using a Machine-learning Approach

Shweta Sharma
National Institute of Technical Teachers Training and Research, shweta.cse@nitttrchd.ac.in

Naveen Kumar
University of Oklahoma

Rakesh Kumar
National Institute of Technical Teachers Training and Research

C. Rama Krishna
National Institute of Technical Teachers Training and Research

Recommended Citation

Sharma, S., Kumar, N., Kumar, R., & Krishna, C. (2020). The Paradox of Choice: Investigating Selection Strategies for Android Malware Datasets Using a Machine-learning Approach. Communications of the Association for Information Systems, 46, pp-pp. https://doi.org/10.17705/1CAIS.04626

Communications of the Association for Information Systems

Research Paper DOI: 10.17705/1CAIS.04626 ISSN: 1529-3181

The Paradox of Choice: Investigating Selection Strategies for Android Malware Datasets Using a Machine-learning Approach

Shweta Sharma
Department of Computer Science and Engineering
National Institute of Technical Teachers Training and Research
India
shweta.cse@nitttrchd.ac.in

Naveen Kumar
Price College of Business
University of Oklahoma
USA

Rakesh Kumar
Department of Computer Science and Engineering
National Institute of Technical Teachers Training and Research
India

C. Rama Krishna
Department of Computer Science and Engineering
National Institute of Technical Teachers Training and Research
India

Abstract:

The increase in the number of mobile devices that use the Android operating system has attracted the attention of cybercriminals who want to disrupt or gain unauthorized access to them through malware infections. To prevent such malware, cybersecurity experts and researchers require datasets of malware samples that most available antivirus software programs cannot detect. However, researchers have infrequently discussed how to identify evolving Android malware characteristics from different sources. In this paper, we analyze a wide variety of Android malware datasets to determine more discriminative features such as permissions and intents. We then apply machine-learning techniques on collected samples of different datasets based on the acquired features' similarity. We perform random sampling on each cluster of collected datasets to check the antivirus software's capability to detect the sample. We also discuss some common pitfalls in selecting datasets. Our findings benefit firms by acting as an exhaustive source of information about leading Android malware datasets.

Keywords: Malware Detection, Data Analytics, Android Package Kits (APKs), Clustering, Machine Learning, Malware Dataset Selection.

This manuscript underwent peer review. It was received 03/27/2019 and was with the authors for 3 months for 1 revision. Alvin Leung served as Associate Editor.

"It's difficult to imagine the power that you're going to have when so many different sorts of data are available."

—Tim Berners-Lee, Inventor of the World Wide Web (Michael, n.d.)

1 Introduction

In 2018, the global population of smartphone users reached about 2.53 billion (Statista, 2019c). These smartphones predominantly used the Android mobile operating system (OS), which had an 88 percent market share in that year (Statista, 2019a). Android's open-source software allows millions of developers to deploy applications (commonly known as apps) on the Google Play Store or third-party platforms (e.g., SlideMe, F-Droid, torrents, etc.) that users can download to customize their smartphones accordingly. As of 2018, the Google Play Store contains around 2.6 million apps (Statista, 2019b) that facilitate diverse tasks such as writing documents, shopping online, renting homes, or requesting food delivery.

Due to its open-source software stack, Android has gained popularity compared to other operating systems; unfortunately, this feature means cybercriminals often target it as well. Attackers create malign Android apps or malware—spoofed versions of genuine apps that they deploy on Google Play Store or third-party sources——to attract users. After users successfully install malware on their device, it can perform malicious functions such as stealing, deleting, modifying, or hiding data.

Consequently, much research has focused on analyzing and detecting Android malware with many tools and services, such as malware datasets. Thus, in order to facilitate research on malware detection, researchers have developed a secondary market of extensive lists of known malware (i.e., malware datasets). These malware datasets include Contagio, Drebin, AndRadar, Ransomware, VirusShare, AndroZoo, HelDroid, and Genome. These datasets focus on profiling malware as comprehensively as possible. In this study, we evaluate some critical aspects of the leading malware datasets and provide guidelines to help firms and researchers select the malware datasets whose malware antivirus software has the most trouble detecting. To the best of our knowledge, this work constitutes the first to critically examine an available array of Android malware datasets.

2 Motivation

In recent years, individuals have begun to use smartphones not only for communication but also for writing documentation, generating PDFs, sending emails, and so on. Further, today's smartphones from companies such as OnePlus, Samsung, HTC, and Asus have enough random access memory (RAM) that they allow users to run multiple apps simultaneously without compromising speed. Users can access various apps (such as mobile banking, social networking, online shopping, games, etc.) on smartphones that perform similar tasks to software programs on desktop computers. Cybercriminals, however, can make money more easily from smartphones than personal computers (PC). For example, malware can send an SMS from a user's phone to a premium-rate number where a network operator will debit the charging cost from the user even though the malware triggered the action (La Polla, Martinelli, & Sgandurra, 2013).

While much literature has discussed malware detection for Microsoft Windows (on PC), the research on malware for Android devices remains an emerging field due in part to the relative lack of datasets. Moreover, one cannot apply PC security solutions to Android smartphones due to the latter's limited resources (such as CPU and battery) and apps' different features (such as permissions, intents, global positioning system (GPS), etc.). In the existing literature, researchers have extracted features from Android malware and performed experiments in which they have trained machine-learning models to detect malware. However, attackers continually develop malware with new strategies, such as making it undetectable for long periods of time (known as zero-day malware). The new types of Android malware continuously emerge in different forms, including spyware, grayware, ransomware, adware, and so on.

Unique Android malware variants with different signatures and features continue to grow in number. For example, in a study in the ELE Times (ELE Times, 2018), unique malware samples grew by 43 percent, whereas the number of malicious families grew by 32 percent. Moreover, anti-malware software cannot detect newer types of malware including viruses, trojans, grayware, and ransomware (Sen, 2018). Thus, to perform experimental work using machine learning models, practitioners and researchers need updated and comprehensive Android malware datasets for feature extraction. Furthermore, given that practitioners

require high-quality data (which contains metadata, the purpose behind its creation, sample types, and measurement errors), we need to compare existing datasets in detail (Link et al., 2017).

As such, we conducted a comprehensive study on Android malware datasets to identify their key aspects. We extracted features (i.e., permissions and intents) of all the datasets we collected and grouped them into clusters according to similarities in their acquired features. We also used antivirus software to analyze random samples from each database cluster to check whether it detected malicious code. Practitioners and researchers can use our findings to select the most appropriate dataset which consists of malicious samples that remain undetected by antivirus software.

3 Contribution

One cannot consider experiments on detecting Android malware that do not use an appropriate Android malware dataset complete. For example, practitioners and researchers working on detecting Android ransomware (a type of malware that demands ransom from victims via cryptocurrency to decrypt/unlock their files on mobile phones (Sipior, Bierstaker, Borchardt, & Ward, 2018)) specifically need an Android ransomware dataset for research. Since creating a primary dataset (i.e., collecting malware oneself) can involve considerable costs in human labor and other resources, researchers and practitioners often use secondary datasets (data that others (often in cooperation) have collected) that include malware samples.

In particular, for such experiments, researchers and practitioners require raw data in the form of Android package kits (APKs) that they can reverse engineer and convert into a readable format (Java/XML). They then extract the features (such as permissions, intents, system calls, etc.) from the malicious and benign samples' XML and Java code. Finally, they employ various machine-learning models (e.g., logistic regression, support vector machine, neural network, etc.) that use these features as input to classify and detect Android malware.

Many studies have used primary datasets (Andronio, Zanero, & Maggi, 2015; Arp et al., 2014; Chen et al., 2018) and secondary datasets (Ab Razak et al., 2018; Saracino, Sgandurra, Dini, & Martinelli, 2016; Xiao, Zhang, Mercaldo, Hu, & Sangaiah, 2017; Yuan, Lu, & Xue, 2016) to detect Android malware. However, no study has examined existing Android malware datasets to identify their characteristics, what malware types they cover, what difficulties researchers face while selecting them, and whether antivirus software can detect the malware that they contain.

In particular, we clustered all the datasets we collected after reverse engineering them and extracting their features to identify malware with similar kinds of features. We then used VirusTotal (a Web interface to scan malware with over 70 antivirus software programs) to analyze random samples from each cluster and determine which ones the available antivirus software detected the least. We conducted this study to familiarize practitioners and researchers with these datasets and enable them to appropriately choose a dataset—one of the most useful resources for their work. For example, researchers and practitioners who develop detection methods for Android malware need datasets that comprise malicious samples that most antivirus software cannot detect. This study also helps researchers and practitioners by providing information on datasets that include benign samples.

This paper proceeds as follows: in Section 4, we review the literature on aggregating and using datasets. In Section 5, we introduce the leading Android malware datasets: Contagio, Drebin, AndRadar, Ransomware, VirusShare, and AndroZoo. In Section 6, we present implementation and experimental details. Specifically, we extract features from all malicious samples in each dataset such as permissions (allowing apps to access data stored on the smartphone) and intents (abstract objects that an app uses to request an action from another app) after performing reverse engineering. We also discuss how we performed clustering based on similarity in the extracted features from each dataset and then analyzed malware samples in VirusTotal. In Section 7, we investigate each dataset to identify the common pitfalls in data selection. In Section 8, we outline the evaluation metrics we used to determine how effectively antivirus software could detect malware in the large datasets. Finally, in Section 9, we conclude the paper.

4 Literature Review

Since the literature lacks research studies on Android malware datasets, we reviewed research papers in which authors collected datasets (from other fields) and used them to detect Android malware. As Figure 1 shows, we divided the literature survey into two parts: 1) dataset aggregation and 2) dataset utilization.

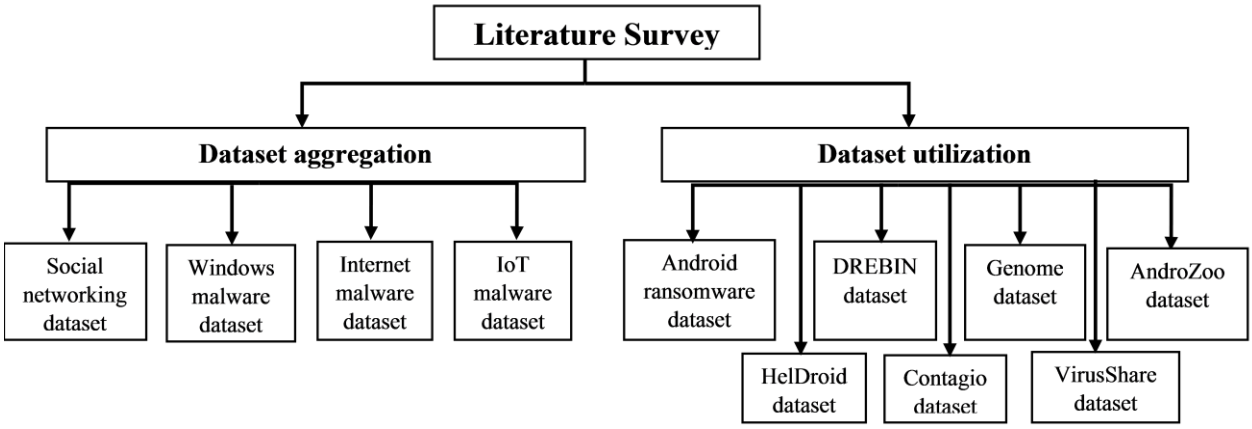

Figure 1. Literature Survey

4.1 Dataset Aggregation

Dataset aggregation refers to collecting and merging datasets from numerous sources such as the Internet, speech, image, sports, and medical records. We found several studies in which researchers performed experimental work on secondary datasets to identify malicious activities.

Smith et al. (2011) discussed the reasons behind choosing secondary datasets. They explored steps to first clarify the research topic to choose an appropriate dataset and then understood it properly to perform meaningful analysis on it. Paranthaman and Thuraisingham (2017) collected malware for Windows PC and Android OS. They focused on choosing the best tool to analyze malware rather than studying datasets. However, they did not cover all Android malware sources as they focused more on malware on Windows PC.

Galal, Mahdy, and Atiea (2016) collected the Windows PC malware dataset from VirusSign (2019) and performed dynamic analysis on it by executing the samples in a virtual environment. They analyzed malicious samples' application program interface (API) calls and employed machine-learning techniques (such as decision trees, random forest, and support vector machine) to detect malware. Sharma, Krishna, and Sahay (2019) collected the Windows PC malware dataset from Kaggle's Microsoft Malware Classification Challenge. They performed static analysis by examining how frequently opcode occurred and applied a feature-selection method (fisher score) to obtain top-most features. They employed machine-learning techniques (such as Logistic Model, J48, REPTree, and Naïve Bayes) in WEKA tool to detect the malware.

Bailey et al. (2007) collected Internet malware samples from the network security community and Arbor Malware Library. They examined these samples using antivirus software including McAfee, F-Prot, ClamAV, Trend, and Symantec to identify unique labels. They found that each antivirus labeled the same malware differently. They observed that antivirus software could not recognize the total number of unique labels that exist for malware. Guo, Cheng, and Kelley (2016) gathered a social networking dataset by crawling MySpace to detect malware propagation. They built a network in organizations by extracting information and social links from user-profiles and user-friend pages, respectively. They analyzed virus and worm features by simulating their propagation process using a susceptible infected recovered (SIR) model and calculated the risk with hierarchical regression. Xiao, Lin, Sun, and Ma (2019) collected the malware dataset from VX Heaven and performed dynamic analysis by executing the samples in Cuckoo Sandbox. They also worked on protecting Internet of things (IoT) devices from malware by employing deep-learning techniques such as neural network-stacked auto-encoders.

4.2 Dataset Utilization

Dataset utilization refers to employing a particular type of dataset such as an Android malware dataset. Chen et al. (2018) generated their own Android ransomware dataset and performed static and dynamic analyses to detect malware. They analyzed widgets (e.g., labels and list views) and activities (e.g.,

capturing photos and dialing a phone). They recorded the coordinates of users' clicks on the layout screen. During the observation, if users initiated no click operation to encrypt files, they considered the encryption operation an abnormal activity that Android ransomware performed. Andronio et al. (2015) prepared their own dataset (i.e., HelDroid) and used natural language processing to dissect and detect mobile ransomware. They classified sentences as scary, payment, porn, law, and copyright to detect Android ransomware. They searched for lockNow(), onKeyDown(), and onKeyUp() methods to detect screen-locking due to Android ransomware and performed static taint analysis on smali code to detect encryption.

Similarly, Arp et al. (2014) created their own dataset (i.e., Drebin) and performed static analysis on dalvik executable (dex) and manifest files. They collected features such as hardware components, requested permissions, intents from manifest files, and application program interface (API) calls, and network addresses from dex files. Then, they applied the support vector machine (SVM) algorithm to collected features to separate two features classes (malicious and benign) with maximum margins. Xiao et al. (2017) collected malicious samples from the Drebin dataset and analyzed system call sequences. They considered one system call equivalent to one word and one system call sequence to one sentence. They employed a deep-learning technique (long short-term memory) to detect Android malware. Onwuzurike et al. (2019) used the Drebin dataset and performed static analysis on Java bytecode to collect features such as API calls using Androguard and call graphs using Soot and FlowDroid. They built a Markov chain to create a feature vector by transitioning the abstracted calls from one possible state to another. Then, they applied random forests, the SVM algorithm, and k-nearest neighbor machine-learning algorithms to feature vectors to classify apps as malicious or benign.

Moreover, Yuan et al. (2016) used the Contagio and Genome datasets to detect Android malware. They performed static analysis on extensible markup language (XML) and dex files to extract features including permissions and APIs. Then, they ran the apps in a sandbox to dynamically analyze their network communication, SMS, encryption, and phone calls. They employed deep neural networks on extracted features to detect Android malware. Saracino et al. (2016) developed a framework called MADAM by using Genome, Contagio, and VirusShare datasets to detect different types of Android malware. They analyzed system calls, admin privileges, processes, user activity, apps running in the foreground, SMS, contact list, permissions, ratings, and marketplace to detect rootkits, ransomware, spyware, botnet, installer malware, and SMS trojans. They trained collected features with the k-NN classifier ($k = 1$) to classify the app as genuine and malicious. Cai, Meng, Ryder, and Yao (2019) collected malicious samples from the VirusShare, Drebin, and Genome datasets. They performed dynamic analysis in an Android emulator to extract features such as method calls and inter-component communication calls. They employed a machine-learning technique (random forest) to detect malware.

Allix, Bissyande, Klein, and Le Traon (2016) created the AndroZoo repository by collecting millions of malicious and benign Android apps for research purpose. Ab Razak et al. (2018) collected benign apps from the AndroZoo repository and malicious apps from the Drebin dataset. They analyzed permissions as a feature to differentiate malicious and genuine apps. They used particle swarm optimization and the information gain algorithm to select the top-most features. After they selected features, they employed various machine-learning techniques (random forest, multilayer perceptron, k-NN, adaptive boosting, and J48 decision trees) to detect Android malware. They found that multilayer perceptron displayed the most accurate results. Pektaş and Acarman (forthcoming) collected benign apps from the AndroZoo repository and performed pseudo-dynamic analyses to extract call graphs from operation codes (opcodes) by modifying the Androguard tool. They employed a deep neural network that they developed by convolving opcode sequences to classify apps as malicious or benign.

To the best of our knowledge, researchers have not fully studied and analyzed Android malware datasets to identify their characteristics, what malware types they cover, what difficulties researchers face while selecting them, and whether antivirus software can detect malware that they contain. Accordingly, we address the following research question (RQ):

RQ: Which Android malware dataset contains malicious samples that antivirus software detects the least often?

5 Dataset Collection

In this section, we provide information about different Android malware datasets for practitioners and researchers to locate these datasets more conveniently. In particular, in Table 1, we provide information about the following datasets: Contagio, Drebin, AndRadar, Ransomware, VirusShare, and AndroZoo.

Some datasets require login credentials for download and/or a password to decompress downloaded files, while some datasets require an API key. For example, one needs an API key from AndroZoo repository's authors to download a bunch of android package kits (APKs) together from Github. Further, the datasets differ in size: some have many more APKs than others. Some contain all types of malware, while others contain specific types of malware. Some contain malicious samples, while others contain both malicious and benign samples. The datasets also differ in age. Based on these classifications, we discuss the Android malware datasets in the following subsections.

Table 1. Accumulated Datasets

Reference(s)	Dataset	Size	Number of samples	Collection date	Sources	Requirements		
						Login credentials	Password to decompress files	API key
Parkour (2011)	Contagio	9 GB	215 APKs	June, 2011, to March, 2018	Website 1	X	✓	X
Drebin (2016)	Drebin	6 GB	5,500 APKs	August, 2010, to October, 2012	Website 2	✓	✓	X
Lindorfer et al. (2014b)	AndRadar	25 GB	7,800 APKs	June to November, 2013	Email	X	X	X
Chen et al. (2018)	Ransomware	2.5 GB	2,300 APKs	2017	Email	X	✓	X
Virusshare (n.d.)	VirusShare	*	31 million malware samples (including APKs)	2011 ~	Website 3	✓	✓	X
Allix et al. (2016), Li et al. (2017)	AndroZoo	*	9 million APKs	2014 ~	Website 4	X	X	✓

Key: * : varies, ~: ongoing
Website 1 = "http://contagiominidump.blogspot.com/"
Website 2 = "https://www.sec.cs.tu-bs.de/$\sim$danarp/drebin/"
Website 3 = "https://virusshare.com/"
Website 4 = "https://androzoo.uni.lu/", "https://github.com/ArtemKushnerov"

5.1 Contagio

Contagio, a mini-dump repository, provides the latest malicious samples, threats, observations, and analysis for practitioners and researchers. This dataset contains 215 malicious android package kits (APKs) (approx. 3 GB). Mila Parkour (2018)—a security researcher—collected the samples from June, 2011, to March, 2018. The dataset also contains MAC OSX and Windows OS malware (approx. 3 GB), malicious traffic in packet capture (PCAP) format (approx. 2 GB) and executable (EXE) format (approx. 350 MB). One can access the dataset and download samples from http://contagiominidump.blogspot.com/. However, one needs a password to decompress files, which one can directly request from the author (the website displays her contact information).

5.2 Drebin

Drebin dataset promotes research on detecting Android malware and provides malicious samples with extracted features sets to practitioners and researchers. This dataset contains 5,500 malicious APKs from 180 distinct malware families (approx. 6 GB). Arp et al. (2014) collected the samples from August, 2010,

to October, 2012. The dataset also contains feature vectors (API calls, permissions, URLs, intents) found in malware samples. One can access the dataset and download samples from https://www.sec.cs.tu-bs.de/~danarp/drebin/download.html. One can request login credentials and the password to download and unzip the samples from the authors (the website displays their contact information).

5.3 AndRadar

AndRadar dataset promotes research on identifying malicious apps in alternative markets (i.e., third-party markets from where users download the apps) that contain specifically ad-aggressive apps (adware). This dataset contains 7,800 malicious APKs (approx. 25 GB). Lindorfer et al. (2014b) collected the samples from June to November, 2013, from approximately 10 alternative markets (such as Slideme, Lenovo, Aptoide, etc.). The dataset contains comma-separated values (CSV) files with approximately 4,500 APKs that the authors identify by their message digest (MD5) hashes with the corresponding package name and market name in which they found the APKs. The CSV files contain duplicate MD5 hash values because the dataset contains different versions of malicious apps. One can request the dataset directly from the authors (their research paper contains their contact information). One does not need any login credentials or API key to download/access the samples.

5.4 Ransomware

Ransomware dataset provides Android ransomware samples from different families (such as Fakedefender, Simplocker, Koler, and so on) to the research community. This dataset contains 2,300 malicious APKs (approx. 2.5 GB). Chen et al. (2018) collected the ransomware samples from an antivirus company (ANTIY) and blogs (Chebyshev & Unuchek, 2014; Jarvis, 2013) in 2017. The dataset contains Android ransomware samples along with their MD5 hashes. One can request the dataset and the password to decompress downloaded files from the authors (their research paper contains their contact information).

5.5 VirusShare

VirusShare.com, a malware repository, allows incident responders, practitioners, security researchers, malware analysts, and other morbidly curious people to easily access live malicious samples. This repository contains approximately 34 million malware samples (as of 2019) for all types of operating systems. For Android, the repository provides malicious APKs with MD5 hashes for all malware samples. VirusShare has collected, indexed, and freely shared malware samples to analysts, practitioners, researchers, and the information security community since 2011 (Virusshare, n.d.). However, Elish, Shu, Yao, Ryder, and Jiang (2015), Lindorfer et al. (2014a), and Saracino et al. (2016) began using this dataset from 2014. One can access the dataset and download samples from https://virusshare.com/. One can directly request login credentials from the administrators (the website displays their contact information).

5.6 AndroZoo

AndroZoo, a repository, contains a rich collection of Android malicious and benign apps from 16 different Android markets such as Genome, Appchina, Anzhi, Fdroid, 1mobile, PlayStore, and so on. Of these, only the Genome dataset contains malicious samples. Allix et al. (2016) collected approximately nine million APKs as of August, 2019. Additionally, they provide a CSV file for APKs with their SHA256, SHA1, MD5, apk size, dex size (size of the dex file), dex date (date attached to the dex file), pkg name (name of the Android package), vercode (version code), vt detection (antivirus software in VirusTotal that detected APK as a malware), vt scan date (date of detection), and markets (to which APK belongs). One can download the dataset with a tool (az) from GitHub[1]. However, one requires an API key to access it, which one can request from the dataset creators (the website displays their contact information).

6 Implementation and Experimental Details

In this section, we discuss how we performed reverse engineering and feature extraction on the samples in each dataset to extract features such as permissions and intents. Based on the extracted features, we performed Gaussian mixture model (GMM) clustering on each dataset to group malware with similar kinds

[1] https://github.com/ArtemKushnerov/az

of features in a single cluster. GMM clustering is an unsupervised machine-learning technique that does not require a labeled dataset for experimentation (Kumar, Venugopal, Qiu, & Kumar, 2018). Subsequently, we randomly chose samples from each cluster and used VirusTotal to analyze them to determine whether anti-virus software could actually detect malware. We used a system with the following specifications to perform the experiments: Windows 8.1 Pro 64-bit operating system, Intel Core i7-4770 CPU @ 3.40GHz, and 8.00 GB memory (RAM). We show the overall structure in Figure 2, which we explain in the following subsections.

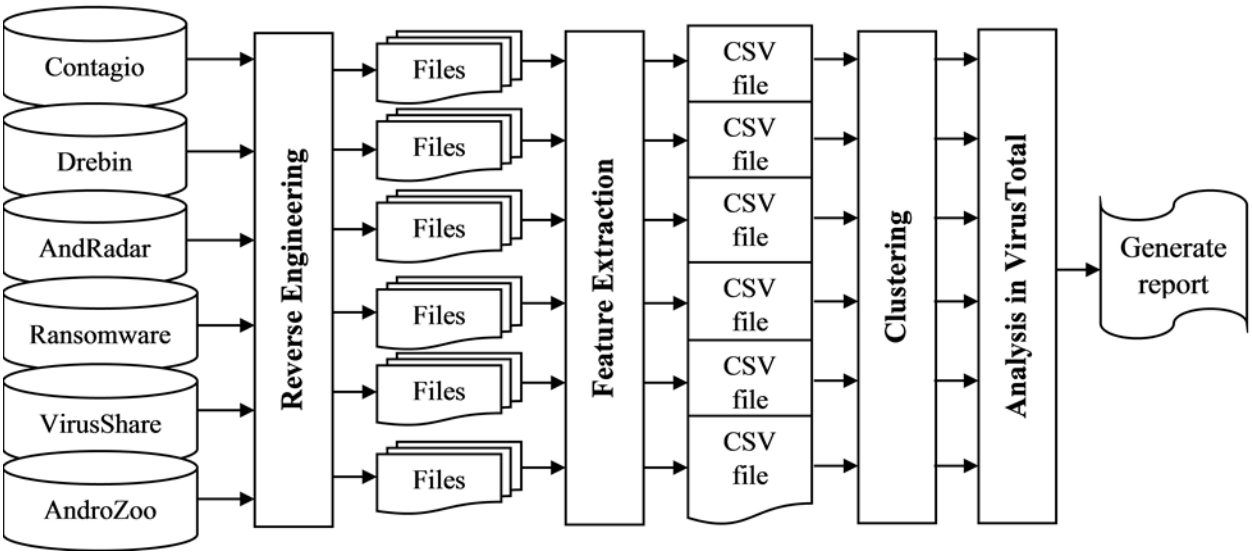

Figure 2. Overall Structure

6.1 Reverse Engineering

The datasets contained malicious samples in the form of Android package kits (APKs). Android uses the APK file format to transmit and install apps on mobile phones. We performed reverse engineering with ApkTool to disassemble all malicious APKs in each dataset into a readable format. The ApkTool (iBotPeaches, 2019) generates dex files, manifest.xml files, and smali files for each APK. We show the commands we used to disassemble the Android app with ApkTool in Figure 3 where “d” corresponds to decompiling the APK and “b” corresponds to repackaging the APK.

```
>> java –jar apktool.jar
>> apktool d appname.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Decoding values */* XMLs...
I: Baksmaling classes.dex...
...
...
>> apktool b appname
I: Smaling smali folder into classes.dex...
I: Building resources...
I: Building apk file...
I: Built apk...
```

Figure 3. ApkTool Commands

6.2 Feature Extraction

After disassembling the APK using reverse engineering, we analyzed AndroidManifest.xml file to extract features such as permissions and intents. We wrote a Python script to extract features that stored the output in comma-separated values (CSV files). We extracted permissions and intents from the AndroidManifest.xml files.

6.2.1 Permissions

All Java-written applications must receive permission from Android during installation. Android provides security to users by informing them about the permissions that any application obtains. Therefore, Android developers declare permissions in AndroidManifest.xml with the < uses−permissions > tag (see Figure 4). Thus, if an application wants access to the calendar, contacts, microphone, location, or any other API, then Android will message the user to either allow or deny the access. However, overclaiming permissions poses a drastic issue in Android that leads to information and monetary losses. For example, if a dictionary application requests unnecessary permission (e.g., READ_ PHONE_STATE), it can exploit the permission by sending the phone's state such as its IMEI number without users' consent. Attackers can use this feature to read users' private information such as SMS, logs, and so on.

Android contains approximately 300 permissions that any application can use. Of these 300, 24 pose a higher risk to users' private data (Android, 2018). We list these 24 permissions in Table 2.

6.2.2 Intents

Intents are abstract objects in Android that an app uses to request an action from another app component. It switches the user from one app to another app based on action (e.g., showing a map, taking a photo, sending a message) it would like to perform. Basically, intents represent the glue that coordinates interactions between activities, services, and broadcast receivers. Just like permissions, Android developers declare intents in AndroidManifest.xml with the < action > tag (see Figure 4). Intents contain operations that an application component will perform (Tam, Feizollah, Anuar, Salleh, & Cavallaro, 2017). Attackers can use this feature to launch activities such as to obtain device administration privileges.

```
<?xml version="1.0" encoding="utf-8" standalone="no"?><manifest xmlns:android="
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
    <uses-permission android:name="android.permission.READ_SMS"/>
    <uses-permission android:name="android.permission.SEND_SMS"/>
    <uses-permission android:name="android.permission.RECEIVE_SMS"/>
    <application android:debuggable="true" android:icon="@drawable/icon" androi
        <receiver android:name=".Notificator">
            <intent-filter>
                <action android:name="android.intent.action.BOOT_COMPLETED"/>
                <category android:name="android.intent.category.HOME"/>
            </intent-filter>
        </receiver>
        <activity android:configChanges="keyboardHidden|orientation" android:la
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Figure 4. AndroidManifest.xml File

Table 2. Dangerous Permissions in Android OS

Dangerous permission group	Dangerous permissions	Description
CALENDAR	READ/WRITE_CALENDAR	Read/write the user's calendar data
CALL_LOG	READ /WRITE_CALL_LOG	Read/write the user's call log
	PROCESS_OUTGOING_CALLS	Read the dialed number during an outgoing call
CAMERA	CAMERA	Access the camera
CONTACTS	READ /WRITE_CONTACTS	Read/write the user's contacts data
	GET_ACCOUNTS	Access to the list of accounts
LOCATION	ACCESS_FINE/COARSE_LOCATION	Access precise/approximate location
MICROPHONE	RECORD_AUDIO	Record audio
PHONE	READ_PHONE_STATE	Read access to phone state (ongoing calls, list of Phone Accounts)
	CALL_PHONE	Start a phone call without using the Dialer to confirm the call
	ADD_VOICEMAIL	Add voicemails in the voice box
	USE_SIP	Use SIP service
SENSORS	BODY_SENSORS	Acquire data collected by sensors (heart rate sensor)
SMS	SEND/RECEIVE/READ_SMS	Send/receive/read short message service
	RECEIVE_WAP_PUSH	Receive wireless application protocol push messages
	RECEIVE_MMS	Monitor incoming multimedia messaging service
STORAGE	READ/WRITE_EXTERNAL_STORAGE	Read/write external storage

6.3 Clustering

After accumulating features (i.e., permissions and intents) from each dataset sample's AndroidManifest.xml file in CSV files, we performed clustering based on the extracted features to bifurcate each dataset's malware samples into clusters. We performed the clustering in such as a way that a malware sample in one cluster resembled the other malware samples in the same cluster and did not resemble the malware samples in other clusters. We applied GMM clustering to group the malware with similar features in the same cluster. We performed the experiment with Python 3.6 on Colaboratory (a free GPU cloud service from Google).

Gaussian mixture modeling is a machine learning-based clustering algorithm that models datasets into clusters according to different Gaussian distributions. It calculates the probability of each sample belonging to a cluster after computing mean (to determine the data's center), variance (to determine the data's spread), and mixing probability (to determine the Gaussian function's size).

We used the probability density function for one-dimensional Gaussian distribution as follows:

$$G(X|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}, \quad (1)$$

where X shows the data points, μ is the mean, σ^2 is the variance, and π is the mixing probability.

We used the probability density function for multi-dimensional Gaussian distribution as follows:

$$G(X|\mu,\sum) = \frac{1}{\sqrt{2\pi|\sum|}} \exp\left(-\frac{1}{2}(X-\mu)^T\sum^{-1}(X-\mu)\right), \quad (2)$$

where X shows the data points, μ is the d-dimensional vector, $\sum$ is the covariance matrix, and π is the mixing probability.

GMM follows the probabilistic model to automatically learn the subpopulation from the total population. It implements the EM algorithm, which contains two steps (an expectation step and maximization step) to calculate the model's parameters (Kumar, Venugopal, Qiu, & Kumar, 2019). The expectation step calculates weights, whereas the maximization step updates the location and shape of all samples.

In our experimental work, several Android malware samples possessed common features in each dataset. For example, approximately 2,200 samples in the ransomware dataset asked for the same permission (i.e., RECEIVE_BOOT_COMPLETED). Thus, we used the GMM clustering algorithm, which allows soft assignments, to allow overlapping between clusters.

6.4 Analysis in VirusTotal

After grouping malware with similar types of features in a single cluster, we performed random sampling on it to select malware samples from different clusters for analysis in VirusTotal and, thus, check whether antivirus software could actually detect the malware.

The website VirusTotal (VirusTotal, n.d.) analyzes files and URLs to detect malware. It has a Web interface and API through which it scans the malware on over 70 antivirus and URL/domain blacklisting services. It notifies the user whether the submitted file/URL is malicious or not, provides the detection label for each malware, states whether a URL belongs to a particular botnet, and provides other information. We submitted APKs after we performed random sampling on each dataset cluster to VirusTotal for analysis. VirusTotal computes the hash values of each APK and submits the APK to antivirus software (such as Avast, Fortinet, McAfee, Microsoft, etc.), which provides a detection output (whether the APK is malicious or not).

7 Common Pitfalls in Dataset Selection

In this section, we discuss common pitfalls in selecting the datasets in terms of sample size, standardization, sample demarcation, and sample depletion:

Sample size: some Android malware datasets do not support robust statistics due to their small size. For example, Contagio contains only 215 APKs. Researchers may not find this data self-sufficient for use in research studies. Therefore, Mercaldo, Nardone, Santone, and Visaggio (2016), Saracino et al. (2016), and Yuan et al. (2016) integrated this dataset with other datasets such as Genome, VirusShare, and HelDroid. Further, the reverse-engineering process also eradicates some malware.

Insufficient standardization: researchers need proper standardization to create and use Android malware datasets. For example, the Drebin, AndRadar, VirusShare, and Ransomware datasets use hash values to store each sample's file name. While Contagio and AndroZoo use the name of corresponding malware to store each sample's file name. The file name stored as hash value suits research studies more than the name of malware because using hashing removes duplicate samples.

Samples demarcation: the AndroZoo repository does not separate malicious APKs from benign ones, although its creators provide a CSV file with ratings to show how many antiviruses consider that the app is malware. However, some malware (e.g., zero-day) evades VirusTotal's security checks (Saracino et al., 2016). Similarly, VirusTotal does not differentiate different types of malware in the datasets we collected (i.e., Drebin, AndRadar, VirusShare, and AndroZoo) as spyware, trojans, ransomware, grayware, rootkits, and so on. Thus, if, for example, practitioners and researchers want to perform experiments to detect only trojans, then they have to run each sample from these datasets in VirusTotal to know its category. In the same way, the Ransomware dataset does not differentiate samples according to their type (i.e., locker or crypto). Thus, if researchers or practitioners use this dataset to conduct a study entirely on locker ransomware, then they may find it difficult to differentiate different ransomware types.

Samples depletion: the reverse-engineering process obliterates some APKs in each dataset. For example, the Ransomware dataset initially contained 2,288 APKs; however, after we applied reverse engineering, that number fell to 2,076 malicious samples. The depleted samples did not preserve any XML, smali, or Java files in them, and, thus, we could not use them for analysis.

8 Performance Evaluation and Experimental Results

In this section, we discuss several performance metrics to evaluate how well the antivirus software could detect malware in the large datasets that we present in Section 5. To the best of our knowledge, no study has compared Android malware datasets in detail. Thus, we referred to related research papers from another field on the same theme (i.e., dataset collection and analysis) in order to choose the performance metrics. For example, Sakar et al. (2013) used true positive (TP), false positive (FP), true negative (TN), and false negative (FN) values to calculate how accurately Parkinson disease can be diagnosed using Parkinson speech dataset. They used an indistinguishable dataset (i.e. a dataset with all types of samples) and, hence, merged these values to classify instances as positive and negative.

In comparison to the literature, we examined datasets with only malicious samples (e.g., Contagio, Drebin, AndRadar, Ransomware, VirusShare, and the Genome dataset in the AndroZoo repository) and with only benign samples (e.g., several datasets in the AndroZoo repository: PlayStore, 1mobile, fdroid, torrents, etc.). Thus, the TP and FP values belong to the datasets with malicious samples, whereas the TN and FN values belong to datasets with benign samples.

True positives (TP): While submitting APKs, we examined the malicious apps that antivirus software detected—that is, the true positives (TP). Figure 5 shows the TP percentage for 10 dataset samples[2] that we randomly selected from clusters in each dataset. Out of these datasets, the AndRadar dataset had the fewest TP followed by VirusShare, Ransomware, Contagio, Drebin, and Genome (in the AndroZoo repository).

False positives (FP): While submitting APKs, we examined the malicious apps that antivirus software mistakenly considered benign—that is, the false positives (FP). K7antivirus, Malwarebytes, Microsoft, and other anti-malware software could not detect some Android malware. Figure 6 shows the FP percentage for 10 dataset samples[3] randomly selected from clusters in each dataset. Out of these datasets, the Genome dataset (in the AndroZoo repository) had the fewest FP followed by Drebin, Contagio, Ransomware, VirusShare, and AndRadar.

True negatives (TN): While submitting APKs, we examined the number of benign apps that antivirus software successfully detected—that is, the true negatives (TN). Since only the AndroZoo repository contained benign samples (refer to Section 5.6), we calculated the TN values for 10 dataset samples[4] that we randomly selected from clusters in benign datasets in the AndroZoo repository. We considered the other datasets to have a TN value of 0 since they did not contain benign samples.

False negatives (FN): While submitting APKs, we examined the number of benign apps that antivirus software mistakenly considered malicious—that is, the false negatives (FN). Since only the AndroZoo repository contained benign samples (refer to Section 5.6), we calculated the FN values for 10 dataset samples[5] that we randomly selected from clusters in benign datasets in the AndroZoo repository. We considered the other datasets to have a FN value of 0 since they did not contain benign samples.

[2] Table 3 shows the average of TP values.

[3] Table 3 shows the average of FP values.

[4] Table 3 shows the average of TN values.

[5] Table 3 shows the average of FN values.

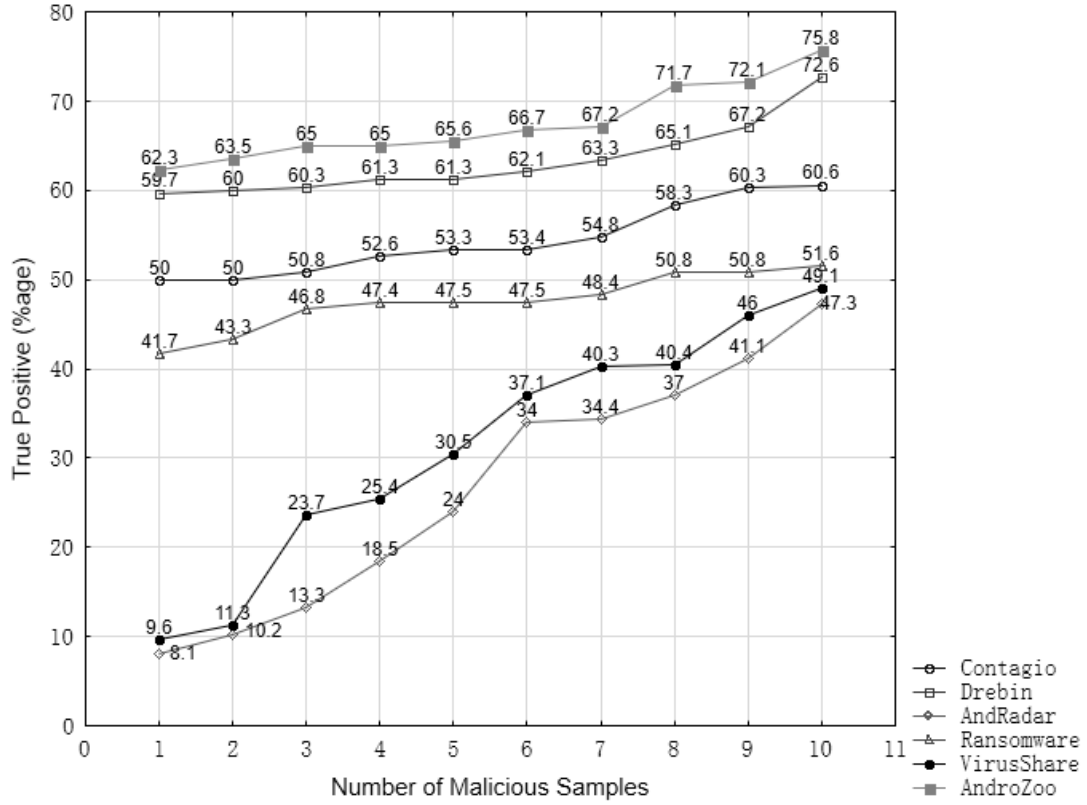

Figure 5. True Positive (Percentage) of Six Malicious Datasets

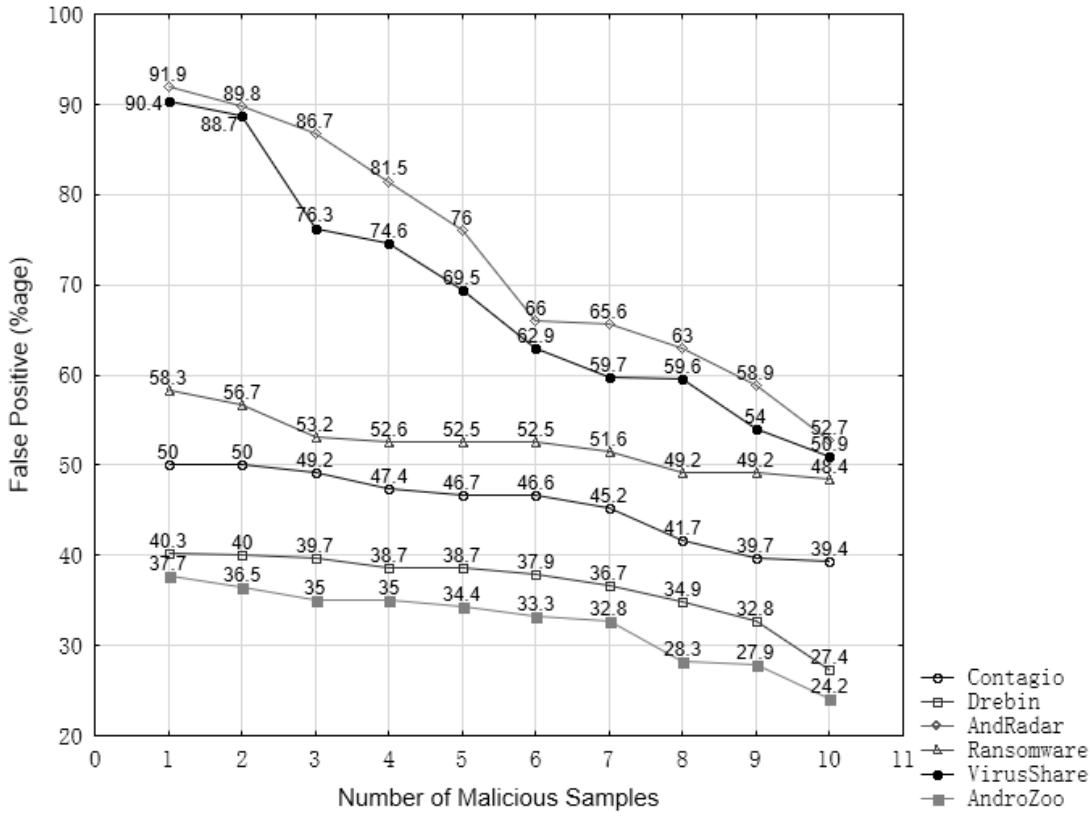

Figure 6. False Positive (percentage) of Six Malicious Datasets

We now explain why some datasets achieved high TP and some achieved low FP, which one can see in Figures 5 and 6.

The AndroZoo repository contains malicious samples in the Genome dataset file, which includes Android trojans (malicious apps that masquerade as benign apps to perform damaging actions such as stealing SMS, contacts, IMEI number, and photos and sending such things to a remote server without users' consent). Commercial antivirus software can easily detect Android trojans. Hence, these samples had the most TP and correspondingly the fewest FP.

The Drebin dataset primarily contains Android trojans that antivirus software can easily detect. Therefore, this dataset had a high antivirus detection rate and, thus, a high number of TP. However, it also contains adware and riskware samples that had a low TP percentage; and can cause harm if cybercriminals exploit them. Hence, Microsoft, Malwarebytes, K7AntiVirus, and other antivirus software did not detect these apps.

The Contagio dataset contains Android trojans samples. It also contains grayware, adware, and potentially unwanted program (PUP) samples that Microsoft, Malwarebytes, K7AntiVirus, and other commercial antivirus software did not detect. Hence, it had fewer TP and more FP compared to the AndroZoo and Drebin datasets.

The Ransomware dataset contains Android ransomware samples in many variants such as Koler, locker, Jisut, and more. These samples had fewer TP and more FP compared to AndroZoo, Drebin, and Contagio as antivirus software did not detect most ransomware samples. Because locker ransomware obtains only one dangerous permission from users (READ_PHONE_STATE), which many legitimate apps also use, Malwarebytes, Microsoft, K7AntiVirus, and other antivirus software failed to detect these samples.

The VirusShare dataset contains many different Android malicious samples, such as riskware, adware (responsible for throwing advertisements on screen), PUA, trojans, and so on. Out of these samples, riskware, adware, and PUA samples had few TP and many FP. In particular, the PUA and riskware samples had few TP since Avast, McAfee, Microsoft, Fortinet, Kaspersky, and other antivirus software did not detect them.

The AndRadar dataset also contains many different Android malicious samples such as PUA, adware, riskware, and more. These samples had the fewest TP and most FP compared to other datasets since Avast, Microsoft, Malwarebytes, Symantec, and other antivirus software did not detect them.

Based on these values, we calculated how accurately the antivirus software performed for each dataset.

Accuracy here refers to the ratio of how accurately anti-virus software can correctly detect malicious and benign samples compared to the total number of input samples. We calculated it using Equation 3:

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \quad (3)$$

The equation produces an accuracy value: low values suggest a dataset contained samples that antivirus software detected the least often. Table 3 shows the results for malicious and benign samples from the six malicious datasets (i.e., Contagio, Drebin, AndRadar, Ransomware, VirusShare, and the Genome dataset in AndroZoo) and one benign dataset (PlayStore, 1mobile, fdroid, torrents, and so on in Androzoo) that we randomly selected from clusters that contained similar kinds of samples. Applied to these samples, the AndRadar dataset had the lowest accuracy value (26.5%) followed by VirusShare, Ransomware, Contagio, Drebin, and AndroZoo.

Table 3. Dataset Accuracy

Dataset	Avg. TP values	Avg. FP values	Avg. TN values	Avg. FN values	Accuracy (%)
AndRadar	14.9	41.4	0	0	26.5
VirusShare	18.4	40.3	0	0	31.3
Ransomware	29.2	32.1	0	0	47.6
Contagio	32.7	27.4	0	0	54.4
Drebin	38.5	22.3	0	0	63.3
AndroZoo	41.3	19.9	60.4	0.2	83.6

9 Discussion and Conclusions

Due to global smartphone usage's ubiquity and Android's rise as the dominant mobile operating system, we collected and compared samples of leading Android malware datasets. In each dataset, we characterized applications according to their availability, which ranged from free direct downloads and open-source software apps to protected applications, which may include services or subscriptions that require user credentials or API keys. We stored each sample's extracted features (i.e., requested permissions and intents that prompt users for access privileges) in a file after we performed reverse engineering. In each dataset, we grouped samples with similar features in a single cluster using machine-learning techniques. Using VirusTotal, we investigated random samples that we collected from each cluster to check how well antivirus software detected them.

We compared the results among different datasets to find the dataset with malicious samples that antivirus software detected the least often. To the best of our knowledge, we present the first study that focuses on collecting and examining Android-specific malware datasets in order to help practitioners and researchers select a dataset that contains malicious samples that antivirus software detects the least often and on evaluating malware-detection techniques to improve the security on Android devices. We found that, in the case of malicious dataset samples, antivirus software detected AndRadar samples the least often since that dataset contains extreme adware samples.

Our findings act as an exhaustive information source about the leading Android malware datasets. Our results provide key information to managers and technical experts working in various organizations who work on securing Android-based smartphones. Decision makers can use the information we provide to choose the best dataset to implement machine-learning algorithms for detecting and preventing Android malware. In the future, we plan to extend this work to include more features. Moreover, we need a robust mechanism to detect further types of Android malware (such as adware and grayware), which antivirus software cannot detect as yet.

References

Ab Razak, M. F., Anuar, N. B., Othman, F., Firdaus, A., Afifi, F., & Salleh, R. (2018). Bio-inspired for features optimization and malware detection. *Arabian Journal for Science and Engineering*, *43*(12), 6963–6979.

Allix, K., Bissyande, T. F., Klein, J., & Le Traon, Y. (2016). Androzoo: Collecting millions of android apps for the research community. In *Proceedings of the International Conference on Mining Software Repositories* (pp. 468-471).

Android. (2018). *Permissions overview.* Retrieved from https://developer.android.com/guide/topics/permissions/overview

Andronio, N., Zanero, S., & Maggi, F. (2015). Heldroid: Dissecting and detecting mobile ransomware. In *Proceedings of the International Symposium on Research in Attacks, Intrusions and Defenses* (pp. 382-404).

Arp, D., Spreitzenbarth, M., Hubner, M., Gascon, H., Rieck, K., & Siemens, C. (2014). DREBIN: Effective and explainable detection of Android malware in your pocket. In *Proceedings of the International Symposium on Network and Distributed System Security* (pp. 23-26).

Bailey, M., Oberheide, J., Andersen, J., Mao, Z. M., Jahanian, F., & Nazario, J. (2007). Automated classification and analysis of Internet malware. In *Proceedings of the International Workshop on Recent Advances in Intrusion Detection* (pp. 178-197).

Cai, H., Meng, N., Ryder, B., & Yao, D. (2019). DroidCat: Effective android malware detection and categorization via app-level profiling. *IEEE Transactions on Information Forensics and Security*, *14*(6), 1455-1470.

Chebyshev, V., & Unuchek, R. (2014). *Mobile malware evolution: 2013. Kaspersky.* Retrieved from https://securelist.com/mobile-malware-evolution-2013/58335/

Chen, J., Chiheng, W., Zhao, Z., Chen, K., Du, R., & Ahn, G.-J. (2018). Uncovering the face of Android ransomware: Characterization and real-time detection. *IEEE Transactions on Information Forensics and Security*, *13*(5), 1286-1300.

Drebin. (2016). *The Drebin dataset.* Retrieved from https://www.sec.cs.tu-bs.de/~danarp/drebin/

ELE Times. (2018). *Mobile malware attacks on the rise as holiday season nears*. Retrieved from https://www.eletimes.com/mobile-malware-attacks-on-the-rise-as-holiday-season-nears

Elish, K. O., Shu, X., Yao, D. D., Ryder, B. G., & Jiang, X. (2015). Profiling user-trigger dependence for Android malware detection. *Computers & Security*, *49*, 255-273.

Galal, H. S., Mahdy, Y. B., & Atiea, M. A. (2016). Behavior-based features model for malware detection. *Journal of Computer Virology and Hacking Techniques*, *12*(2), 59-67.

Guo, H., Cheng, H. K., & Kelley, K. (2016). Impact of network structure on malware propagation: A growth curve perspective. *Journal of Management Information Systems*, *33*(1), 296-325.

iBotPeaches. (2019). A*PKTOOL.* Retrieved from https://ibotpeaches.github.io/Apktool/

Jarvis, K. (2013). Cryptolocker ransomware. *Secureworks*. Retrieved from https://www.secureworks.com/research/cryptolocker-ransomware

Kumar, N., Venugopal, D., Qiu, L., & Kumar, S. (2018). Detecting review manipulation on online platforms with hierarchical supervised learning. *Journal of Management Information Systems*, *35*(1), 350-380.

Kumar, N., Venugopal, D., Qiu, L., & Kumar, S. (2019). Detecting anomalous online reviewers: An unsupervised approach using mixture models. *Journal of Management Information Systems*, *36*(4), 1313-1346.

La Polla, M., Martinelli, F., & Sgandurra, D. (2013). A survey on security for mobile devices. *IEEE Communications Surveys & Tutorials*, *15*(1), 446-471.

Li, L., Gao, J., Hurier, M., Kong, P., Bissyande, T. F., Bartel, A., Klein, J., & Le Traon, Y. (2017). *AndroZoo++: Collecting millions of Android apps and their metadata for the research community.* arXiv:1709.05281.

Lindorfer, M., Neugschwandtner, M., Weichselbaum, L., Fratantonio, Y., Van Der Veen, V., & Platzer, C. (2014a). Andrubis—1,000,000 apps later: A view on current Android malware behaviors. In *Proceedings of the International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security* (pp. 3-17).

Lindorfer, M., Volanis, S., Sisto, A., Neugschwandtner, M., Athanasopoulos, E., Maggi, F., Platzer, C., Zanero, S., & Ioannidis, S. (2014b). AndRadar: Fast discovery of Android applications in alternative markets. In *Proceedings of the International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment* (pp. 51-71).

Link, G. J. P., Lumbard, K., Conboy, K., Feldman, M., Feller, J., George, J., Germonprez, M., Goggins, S., Jeske, D., Kiely, G., Schuster, K., & Willis, M. (2017). Contemporary issues of open data in information systems research: Considerations and recommendations. *Communications of the Association for Information Systems*, *41*, 587-610.

Mercaldo, F., Nardone, V., Santone, A., & Visaggio, C. A. (2016). Ransomware steals your phone: Formal methods rescue it. In *Proceedings of the International Conference on Formal Techniques for Distributed Objects, Components, and Systems* (pp. 212-221).

Michael, M. (n.d.). *How to think differently about data, insights, strategy, and analytics.* Retrieved from https://www.markmichael.io/insights/forget-big-data-data-types-that-really-matter/

Onwuzurike, L., Mariconti, E., Andriotis, P., Cristofaro, E. De, Ross, G., & Stringhini, G. (2019). MaMaDroid: Detecting android malware by building Markov chains of behavioral models. *ACM Transactions on Privacy and Security*, *22*(2), 1-34.

Paranthaman, R., & Thuraisingham, B. (2017). Malware collection and analysis. In *Proceedings of the International Conference on Information Reuse and Integration* (pp. 26-31).

Parkour, M. (2011). *Contagio mobile. Retrieved* from http://contagiominidump.blogspot.com/

Parkour, M. (2018). *Blogger*. Retrieved from https://www.blogger.com/profile/09472209631979859691

Pektaş, A., & Acarman, T. (Forthcoming). Learning to detect Android malware via opcode sequences. *Neurocomputing*.

Sakar, B. E., Isenkul, M. E., Sakar, C. O., Sertbas, A., Gurgen, F., Delil, S., Apaydin, H., & Kursun, O. (2013). Collection and analysis of a Parkinson speech dataset with multiple types of sound recordings. *IEEE Journal of Biomedical and Health Informatics*, *17*(4), 828-834.

Saracino, A., Sgandurra, D., Dini, G., & Martinelli, F. (2016). Madam: Effective and efficient behavior-based Android malware detection and prevention. *IEEE Transactions on Dependable and Secure Computing*, *15*(1), 83-97.

Sen, R. (2018). Challenges to cybersecurity: Current state of affairs. *Communications of the Association for Information Systems*, *43*, 22-44.

Sharma, S., Krishna, C. R., & Sahay, S. K. (2019). Detection of advanced malware by machine learning techniques. In M. Pant, T. K. Sharma, O. P. Verma, R. Singla, & A. Sikander (Eds.), *Soft computing: Theories and applications* (pp. 333-342). Jhansi, India: Springer.

Sipior, J. C., Bierstaker, J., Borchardt, P., & Ward, B. T. (2018). A ransomware case for use in the classroom. *Communications of the Association for Information Systems*, *43*, 598-614.

Smith, A. K., Ayanian, J. Z., Covinsky, K. E., Landon, B. E., McCarthy, E. P., & Wee, Christina C., & Steinman, M. A. (2011). Conducting high-value secondary dataset analysis: An introductory guide and resources. *Journal of General Internal Medicine*, *26*(8), 920-929.

Statista. (2019a). *Global market share held by the leading smartphone operating systems in sales to end users from 1st quarter 2009 to 2nd quarter 2018.* Retrieved from https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/

Statista. (2019b). *Number of available applications in the Google Play store from December 2009 to December 2018.* Retrieved from https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/

Statista. (2019c). *Number of smartphone users worldwide from 2014 to 2020 (in billions).* Retrieved from https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/

Tam, K., Feizollah, A., Anuar, N. B., Salleh, R., & Cavallaro, L. (2017). The evolution of Android malware and Android analysis techniques. *ACM Computing Surveys*, *49*(4), 1-41.

Virusshare. (n.d.). *Virusshare*. Retrieved from https://virusshare.com

VirusSign. (2019). *MALWARE-LIST*. from https://www.virussign.com/index.html

VirusTotal. (n.d.). *VirusTotal*. Retrieved from https://www.virustotal.com

Xiao, F., Lin, Z., Sun, Y., & Ma, Y. (2019). Malware detection based on deep learning of behavior graphs. *Mathematical Problems in Engineering.* Retrieved from https://www.hindawi.com/journals/mpe/2019/8195395/

Xiao, X., Zhang, S., Mercaldo, F., Hu, G., & Sangaiah, A. K. (2017). Android malware detection based on system call sequences and LSTM. *Multimedia Tools and Applications*, *76*, 1-21.

Yuan, Z., Lu, Y., & Xue, Y. (2016). Droiddetector: Android malware characterization and detection using deep learning. *Tsinghua Science and Technology*, *21*(1), 114-123.

About the Authors

Shweta Sharma received BTech degree in Information Technology from Himachal Pradesh University, Shimla, India, in 2013 and MTech degree in Computer Science and Technology (specialization in Cyber Security) from Central University of Punjab, Bathinda, India, in 2016. She is pursuing doctoral research on Android Security from Department of Computer Science & Engineering at National Institute of Technical Teachers Training & Research (NITTTR) Chandigarh.

Naveen Kumar is an Assistant Professor of Management Information Systems in the School of Business, University of Oklahoma, Norman. He received his PhD from the University of Washington, Seattle. His research focuses on applying deep learning and other artificial intelligence techniques in social media and information systems. His work has been published in Information Systems Research, Journal of Management Information Systems, Strategic Management Journal, and others. Before joining academia, he worked as a researcher in the high-tech industry, solving complex business problems in IT, Finance, and Manufacturing using advanced machine-learning techniques.

Rakesh Kumar is an Assistant Professor in the Department of Computer Science and Engineering at NITTTR Chandigarh. He received his PhD in Computer Engineering from N.I.T. Kurukshetra. His key area of interest includes MANET, Cloud Computing, Scheduling Algorithms, and Wireless Sensor Networks. He has number of International/National conference/journal publications to his credit.

C. Rama Krishna is a Professor in the Department of Computer Science & Engineering at NITTTR, Chandigarh. He received his PhD from IIT Kharagpur. His research area includes Wireless Communications & Networks, Computer Networks, Distributed Computing, Cryptography and Cyber Security. Dr. Krishna has published numerous papers in different International journals and conferences.