Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

Multi-labeling of complex, multi-behavioral malware samples

P. García-Teodoro*, J.A. Gómez-Hernández, A. Abellán-Galera

Network Engineering & Security Group (https://nesg.ugr.es), University of Granada, Spain

A R T I C L E   I N F O

Article history:
Received 7 April 2022
Revised 26 June 2022
Accepted 20 July 2022
Available online 23 July 2022

Keywords:
Android
Behavior
Dataset
Labeling
Malware

A B S T R A C T

The use of malware samples is usually required to test cyber security solutions. For that, the correct typology of the samples is of interest to properly estimate the exhibited performance of the tools under evaluation. Although several malware datasets are publicly available at present, most of them are not labeled or, if so, only one class or tag is assigned to each malware sample. We defend that just one label is not enough to represent the usual complex behavior exhibited by most of current malware. With this hypothesis in mind, and based on the varied classification generally provided by automatic detection engines per sample, we introduce here a simple multi-labeling approach to automatically tag the usual multiple behavior of malware samples. In the paper, we first analyze the coherence between the behaviors exhibited by a specific number of well-known malware samples dissected in the literature and the multiple tags provided for them by our labeling proposal. After that, the automatic multi-labeling scheme is executed over four public Android malware datasets, the different results and statistics obtained regarding their composition and representativeness being discussed. We share in a GitHub repository the multi-labeling tool developed, for public usage.

1. Introduction

The development of cyber security solutions requires an audit or assessment process before deploying them in real environments. The goal of such a procedure is to evaluate the performance of each particular solution and, from that, to conclude its benefits, effective performance and potential operational restrictions (Leszczyna, 2021; Madhavan et al., 2009).

To obtain valid conclusions from a given assessment process, it should be carried out on in-exploitation environments so that the global behavior analyzed is fully real. However, that generally does imply unacceptable security risks and threats for users, services and systems on the target environment. Hence, controlled evaluation deployments are usually considered instead. As a particular case, simulation tools can be adopted (*e.g.*, Breachlock, Cymulate, Intragen, Foresseti, AttackIQ, XM Cyber, among others) (Couretas, 2019; Hacks et al., 2021; Kavak et al., 2021; Stash, 2022). Although useful with training aims and to dig into attacks understanding, this option is however widely criticized by experts because they do not include 'real' situations and activities but just *mimic* them (BirghtTALK, 2021; Maciá-Fernández et al., 2018; Veksler et al., 2018). An intermediate solution between in-exploitation environments and simulated scenarios and tools is that of considering controlled real environments. This way, the use of virtual machines or isolated physical devices to deploy malicious users, software and/or services is widely accepted. Although more realistic than simulation, this solution still presents limitations because some attacks can behave differently depending on the observed environment they operate on, thus evading potential traps and analysis tools (CheckPoint, 2022; Gruber and Freiling, 2022; Mills and Legg, 2020; Technologies, 2021).

Whatever the case considered for evaluating a specific cyber security solution (real scenarios, sandboxing environments, etc.), both legitimate, 'normal' and malicious behaviors should be analyzed to estimate different operation performance parameters like accuracy, false positive rate, resource consumption, computation complexity, etc. (Alshaibi et al., 2022; Hassanien and Elhoseny, 2019; Highnam et al., 2021; Sarker et al., 2020). Gathering legitimate traces is not a so difficult task as we can monitor our environment to collect regular, expected activities by using a number of tools like network sniffers, system logs, etc. Regarding malicious behaviors, they can be specifically deployed and generated either manually or through the deployment of well-known malign applications or malware.

* Corresponding author.
E-mail addresses: pgteodor@ugr.es (P. García-Teodoro), jagomez@ugr.es (J.A. Gómez-Hernández), albertoabellan@ugr.es (A. Abellán-Galera).

https://doi.org/10.1016/j.cose.2022.102845

There exists a number of datasets with malware samples publicly available on the Internet which are able to be used with assessment purposes. However, most of them are not properly labeled with the specific typology or family they belong to, so that their use with evaluation aims is in some sense limited. In fact, each of the samples should be properly tagged to obtain the so-called *ground-truth* (QAnalysts, 2022; Tardiff et al., 2016; Zhang et al., 2019), which is necessary both to develop and evaluate supervised machine learning schemes (Dib et al., 2021; Kaspersky, 2021; Katrenko and Semeniak, 2022; Xin et al., 2018). However, such a labeling stage is far from being trivial in the cyber security field because of some main reasons. First, an expert, manual inspection with classification and tagging purposes would become an arduous task. Second, in case an automatic labeling process is performed, the tags will surely vary depending on the specific decisor or classifier considered Abusitta et al. (2021); Maniriho et al. (2022).

In this context, we will show how the bulk of works on this topic in the literature are intentionally aimed at unifying the decision provided by various automatic classifiers, in order to provide just a single label per sample and thus to avoid a supposed confusion (Kantchelian et al., 2015). Just on the contrary, we defend and propose here as a main novelty to take advantage of the potential provision of different labels per sample by separate classifiers to conclude multiple, complex behaviors for malware. This way, we introduce and contribute here:

- A multi-labeling scheme based on tagging a malware sample through the combination of the labels assigned to it by different existing detectors.
- An experimentation to demonstrate the coherence between the multiple labels proposed to be assigned to a malware sample and the usual multiple, complex behavior observed for it.
- Based on the above, a study about the composition and typology of malware samples of four well-known public Android malware datasets is afterwards performed.

As a direct result of our multi-labeling methodology, an auditor can conclude which datasets are the most appropriate ones for a given assessment process. In other words, audit tasks can be better adapted to specific assessment procedures than they are now.

According to the above, the organization of the rest or the paper is as follows. Section 2 presents main background on malware labeling, as well as the existence of different public malware datasets. After that, Section 3 introduces our malware multi-tagging methodology based on the labels provided by public malware classifiers. With the aim of validating the multi-behavior hypothesis for malware, we perform afterwards in the same section a detailed analysis of some malware samples manually dissected in the literature. From that, we discuss the coherence between the several real behaviors concluded for them in the corresponding works and the various tags provided by our multi-labeling methodology for those same specific samples.

Once concluded the validity of the multi-behavior hypothesis for malware, we execute in Section 4 the multi-labeling proposal over four well-known public Android malware datasets. Main results and statistics regarding the multiple tags obtained for the samples contained are subsequently shown in the same section. From the results obtained, we discuss afterwards in Section 5 the real composition of the specific malware datasets analyzed, so that the usefulness of our approach to help in labeling malware samples with assessment purposes is highlighted. Finally, the work is summarized and some future developments pointed out in Section 6.

2. Malware datasets and labeling

As previously stated, the existence of malware datasets is widely required for security developers and auditors (Adhao and Pachghare, 2021; Gençaydin et al., 2021; Highnam et al., 2021; Hreirati et al., 2018; Yang et al., 2021; Yavanoglu and Aydos, 2017). Table 1 shows a number of public malware repositories (Rokon et al., 2020; Ugarte-Pedrero et al., 2019; Zelster), where different malware typologies are considered.

Provided the generalized adoption of mobile platforms (Cisco, 2020; GobalWebIndex, 2020) and the relevance and impact of security threats and malware on that kind of devices at present (Agilie; Zimperium, 2022), the existence of mobile malware to audit security solutions for this kind of systems is of increasing interest. This way, despite some of the previously referred datasets contain such a type of malware samples (*e.g.*, VirusShare and VirusTotal, which can provide professionals with important malware related apk databases for researching and teaching purposes), there exist several available specific mobile malware datasets. Some of them are as follows, where we must remark that the bulk of them correspond to the Android OS as it is the most extended mobile platform nowadays (Statista, 2022; Webtribunal, 2022):

- *AAMG Dataset* (http://www.unb.ca/cic/datasets/android-adware.html).
  AAGM dataset is captured by installing the Android apps on the real smartphones semi-automated. The dataset is generated from 1900 applications as follows: 250 adware apps, 150 apps of general malware, 1500 benign apps.
- *AMD Project* (http://amd.arguslab.org).
  AMD contains 24,553 samples, categorized in 135 varieties among 71 malware families ranging from 2010 to 2016. The dataset provides an up-to-date picture of the current landscape of Android malware, and it is publicly shared with the community.
- *Android Malware Genome Project* (http://www.malgenomeproject.org).
  Here you can find “more than 1200 malware samples that cover the majority of existing Android malware families, ranging from their debut in August 2010 to recent ones in October 2011”.
- *Android PRAGuard Dataset* (http://pralab.diee.unica.it/en/AndroidPRAGuardDataset).
  This dataset contains 10,479 samples, obtained by obfuscating the MalGenome and the Contagio Minidump datasets with seven different obfuscation techniques.
- *AndroZoo* (https://androzoo.uni.lu).
  AndroZoo is a growing collection of Android applications collected from several sources, including the official Google Play app market. At the writting time it contains more than 19,5 million different apks, each of which has been (or will soon be, say the authors) analyzed by tens of different antivirus products to know which applications are detected as malware.
- *CICAndMal2017* (https://www.unb.ca/cic/datasets/andmal2017.html).
  Both malware and benign applications are run on real smartphones to avoid runtime behavior modification of advanced malware samples that are able to detect the emulator environment. More than 10,854 samples (4,354 malware and 6500 benign) are collected from several sources. In addition, over six thousand benign apps from Googleplay market published in 2015, 2016, 2017 are also collected. Four categories are considered: adware, ransomware, scareware, SMS malware.
- *CICMalDroid 2020* (https://www.unb.ca/cic/datasets/maldroid-2020.html).

Table 1
Examples of public malware repositories.

Dataset	Website	Description
Aposemat IoT-23	https://www.stratosphereips.org/datasets-iot23	A labeled dataset with malicious and benign IoT network traffic. This dataset was created as part of the Avast AIC laboratory with the funding of Avast Software
Awesome Open Source	https://awesomeopensource.com/project/InQuest/malware-samples	Malware samples and relevant dissection information
BODMAS Malware Dataset	https://whyisyoung.github.io/BODMAS/	BODMAS is short for Blue Hexagon Open Dataset for Malware AnalysiS
Da2dalus The-MALWARE-Repo	https://github.com/Da2dalus/The-MALWARE-Repo	A repository full of malware samples
Gibson malware research experimental repository	https://www.grc.com/malware.htm	Malware available for private forensic evaluation and experimentation to facilitate individual careful and responsible exploration
HybridAna-lysis	https://www.hybrid-analysis.com/	Offers a database of malware samples but what sets it apart is two things. First, a free malware analysis service open to all. Second, the aptly named Hybrid Analysis technology that the search uses to compare the sample. It checks multiple databases and file collections to detect some of the rarer malware samples
1.55M API Import Dataset for Malware Analysis	https://ieee-dataport.org/open-access/155m-api-import-dataset-malware-analysis	This dataset is part of my Master's research on malware detection and classification using the XGBoost library on Nvidia GPU. The dataset is a collection of 1.55 million of 1000 API import features extract from jsonl format of the EMBER dataset 2017 v2 and 2018
InQuestLabs	https://labs.inquest.net/	A malware database which offers a solid list of features: Deep file inspection (DFI), Aggregate reputation database, Indicators of compromise (IOC), Base64 regular expression generator, Mixed hex case generator, UInt() trigger generator
Malware Archaeology	https://www.malwarearchaeology.com/analysis	It provides malware reports as well as a malware management framework
Malware-Bazaar	https://bazaar.abuse.ch	A project from abuse.ch with the goal of sharing malware samples with the infosec community, AV vendors and threat intelligence providers
PacketTotal	https://packettotal.com/about.html	An engine for analyzing, categorizing, and sharing pcap files. The tool was built with the InfoSec community in mind and has applications in malware analysis and network forensics. It also provides files containing malware samples
Reddit	https://www.reddit.com/r/Malware/comments/dfis29/malware_repository	Malware reports and information
SecRepo	https://www.secrepo.com/	Attempt to keep a somewhat curated list of Security related data found, created, or pointed to
SOREL	https://ai.sophos.com/2020/12/14/sophos-reversinglabs-sorel-20-million-sample-malware-dataset/	A production-scale dataset containing metadata, labels, and features for 20 million Windows Portable Executable files, including 10 million disarmed malware samples available for download for the purpose of research on feature extraction to drive industry-wide improvements in security
TekDefense	http://www.tekdefense.com/downloads/malware-samples	For educational purposes only, this web offers files containing malware or exploits collected through honeypots and other various means
TheZoo	https://github.com/ytisf/theZoo	A project created to make the possibility of malware analysis open and available to the public
URLhaus	https://urlhaus.abuse.ch/	A project from abuse.ch with the goal of sharing malicious URLs that are being used for malware distribution
VirusBay	https://beta.virusbay.io/	A web-based, collaboration platform that connects security operations center (SOC) professionals with relevant malware researchers
VirusShare	https://virusshare.com	A repository of malware samples to provide security researchers, incident responders, forensic analysts, and the morbidly curious access to samples of live malicious code. Access to the site is granted via invitation only
VirusSign	https://www.virussign.com	A huge collection of high quality malware samples which, as previously stated, is a valuable resource for antivirus industry and threat intelligence to improve security related products
VirusTotal	https://www.virustotal.com/gui/home	A website created by the Spanish security company Hispasec Sistemas in 2004, which was acquired by Google in 2012. VirusTotal aggregates many antivirus products and online scan engines to check for viruses that the user's own antivirus may have missed, or to verify against any false positives. In addition, VirusTotal provides a number of malware samples for the community

The samples composing CICMalDroid 2020 were collected from December 2017 to December 2018, the dataset having four properties: *(i)* it has more than 17,341 Android samples; *(ii)* it includes recent and sophisticated Android samples until 2018; *(iii)* it has samples spanning between five different categories: Adware, Banking malware, SMS malware, Riskware, and Benign Comprehensive; and *(iv)* it includes the most complete captured static and dynamic features compared with other publicly available datasets.

- *Contagio Mobile Malware Mini Dump* (http://contagiominidump.blogspot.hk).
  Contagio mobile mini-dump offers an upload dropbox for you to share your mobile malware samples. Although interesting, the dataset is composed of a reduced number of samples.
- *Drebin* (https://www.sec.cs.tu-bs.de/~danarp/drebin/index.html).
  The dataset contains 5560 applications from 179 different malware families. The samples have been collected in the period of August 2010 to October 2012 and are available by the Mobile-Sandbox Project (Arp et al., 2014; Spreitzenbarth et al., 2013).
- *Kharon Malware Dataset* (http://kharon.gforge.inria.fr/dataset).

The Kharon dataset is a collection of malware totally reversed and documented. It analyzes under a microscope a (reduced) number of malware samples like SimpLocker, AndroRAT o Minecraft.

From the above datasets, other repositories are constructed to conduct some specific malware related studies, *e.g. AndroVul* (https://github.com/Zakeya/AndroVul) and *CIDRE* (https://gitlab.inria.fr/cidre-public/dada).

As already discussed, the samples in the datasets should be classified and labeled in order to precisely estimate the effective accuracy of the security solutions developed and tested through them. However, the labeling process is not a trivial task. Performing that manually by experts becomes a time consuming, arduous process, and, as a consequence, usually unapproachable. On the other hand, the use of automatic labeling procedures is not free of limitations either. The most relevant one is the existence of inconsistencies in classification, due to a number or reasons:

- The classification of a given malware sample can vary from one detector to another (Abt and Baier, 2014; Mohaisen and Alrawi, 2014; Zhu et al., 2020), as the specific patterns, filters or heuristics established to take the classification decision will directly affect the final conclusion about the nature and typology of the sample analyzed. In other words, as evidenced in the literature (Amer et al., 2022; Dasgupta et al., 2020; Fiky et al., 2021; Imtiaz et al., 2021; Kim et al., 2022a; 2021; Lashkari et al., 2018; Sihwail et al., 2018; Yerima and Sezer, 2019): static detection differs from dynamic detection, the monitoring of certain parameters or variables (traffic related, filesystem access, permissions used, etc.) will highlight different environment's states and circumstances, a signature-based detector can provide different alarm than a anomaly-based one, etc.
- In addition, there exists a lack of standard naming convention, so that vendors can assign different names to the same sample (Beck and Connolly, 2006; CARO; GdataSoftware, 2019), which evidences the necessity of some kind of naming unification.

Several works in the literature are focused on solving or reducing inconsistences in malware labels. In this line, authors in Kantchelian et al. (2015) propose to combine multiple anti-virus vendor labels into a single authoritative ground-truth label. For that, they present both a supervised and a unsupervised technique to assign confidence weights to vendors. Instead, an attempt to obtain AV labels without any prior knowledge or pre-labeled datasets is performed in Kim et al. (2022b). Some other works in the literature deal with to cluster and unify labels from detection engines. This is the case of Sebastián et al. (2016), where authors introduce an automatic labeling tool named *AVClass* that given the AV labels for a potentially massive number of malware samples, it outputs the most likely family names for each sample. A similar proposal named *Euphony* can be found in Hurier et al. (2017), for which authors claim that, unlike AVClass, no labeled samples are requiered to distinguish family names from generic tokens. AVClass' authors introduced afterwards *AVClass2* (Silvia and Caballero, 2020), which uses, and helps building, an open taxonomy that organizes concepts in AV labels, but which is not constrained to a predefined set of tags.

Instead of trying to unify and reduce labels for a given malware sample, we propose here to take advantage of the varied classification usually provided by public classifiers to contribute a novel multi-labeling scheme for malware. It operates as follows:

1. Firstly, a set of public malware detection engines are automatically consulted about a given sample. As a result, the sample will be classified by each consulted engine according to its specific criteria and methodology.
2. From the classes obtained, and taking into the consideration a group of pre-defined expected behaviors for malware, each sample will be finally assigned with a tuple of labels aimed to gather the different potential typologies the sample corresponds to.

The central hypothesis behind the multi-labeling scheme introduced here is the usual complex behavior of current malware, so that a given sample can simultaneously behave, let's say, as a botnet, as well as a spyware, a banker or/and some other similar malicious codes. As a result of the labeling procedure, each malware sample will be automatically categorized by means of a set of terms related with its potential multiple behaviors.

This proposal clearly differs from previous existent solutions in the literature where a label merging process is defended to usually assign a unique tag per malware sample. Our proposal will ease, while strengthening, the task of creating specific assessment procedures.

3. Classification and multi-labeling of malware samples

Every specific malicious apk is usually identified in terms of its associated MD5 or SHA hash value, *e.g.* 2358a97d0f9b0e4b7d8e5a8386105c97. This cryptic name does not allow to know a priori what type of specific malware the sample corresponds to. However, just making use of the hash value we can classify the apk according to its typology by following a two-step procedure: automatic engine-based classification, and behavior-based multiple tagging. For that, we propose to operate each malware sample as follows:

1. In the first step, an automatic request to a number of well-known public detection engines is performed. To ease the process, the VirusTotal's API is used here. In particular, we consider API V2 third party Python 3.x scripts, as described in VirusTotal (2022). This way, each filehash is easily analyzed by a total of 73 detection engines like Avast, ClamAV, Comodo, DrWeb, Fortinet, McAfee, Panda, Symantec or TrendMicro (see Table 2).
   After this step, the output for a given filehash is the classification provided by each of the detection engines which positively detect the hash as a malware sample (see Table 3 for the specific classification of the filehash 00eeccd7fab472022060305819493225).
2. As expected and previously explained, it can be observed that the classification of a given malware sample can vary from a commercial engine to another. This is accepted due to the specific detection procedure implemented in each case, where different heuristics, codes or other criteria can be considered to identify a given sample. Even more, it is possible that the specimen is not detected by a specific engine because the malware does not match the detection rules considered by the detector.
   Thus, with the aim to 'normalize' and 'integrate' the potentially multiple classification provided in step 1 before, a subsequent predefined behavior-based tagging step is carried out. For that, we have defined the set of well-known behaviors related with threats and risks shown in Table 4. They are aimed to give some order from a behavioral point of view (avoiding family name, platform, and any other specific information) to the generic categories contained in the AV labels considered in Sebastián et al. (2016).
   This way, implemented in Python, after this second step each filehash will be assigned with a sequence of terms from Table 4. For example, the hash 00eeccd7fab472022060305819493225 in Table 3 is assigned (from the detection provided by the engines consulted) with the tuple of behaviors: <*adware, gray,*

Table 2
Detection engines consulted by VirusTotal.

Acronis (Static ML)	Ad-Aware	AhnLab-V3	Alibaba
ALYac	Antiy-AVL	Arcabit	Avast
Avast-Mobile	Avira (no cloud)	Baidu	BitDefender
BitDefenderFalx	BitDefenderTheta	Bkav Pro-	CAT-QuickHeal
ClamAV	CMC	Comodo	CrowdStrike Falcon
Cybereason	Cylance	Cynet	Cyren
DrWeb	eGambit	Elastic	Emsisoft
eScan	ESET-NOD32	Fortinet	F-Secure
GData	Gridinsoft	Ikarus	Jiangmin
K7AntiVirus	K7GW	Kaspersky	Kingsoft
Lionic	Malwarebytes	MAX	MaxSecure
McAfee	McAfee-GW-Edition	Microsoft	NANO-Antivirus
Palo Alto Networks	Panda	Qihoo-360	Rising
SecureAge APEX	SentinelOne (Static ML)	Sophos	SUPERAntiSpyware
Symantec	Symantec Mobile Insight	TACHYON	Tencent
Trapmine	Trellix (FireEye)	TrendMicro	TrendMicro-HouseCall
Trustlook	VBA32	VIPRE	ViRobot
Webroot	Yandex	Zillya	ZoneAlarm by Check Point
Zoner	-	-	-

Table 3
Classification results for the filehash 00eeccd7fab 472022060305819493225.

Engine	Detection result
Avira (no cloud)	ANDROID/Hiddad.AMAN.Gen
CAT-QuickHeal	Android.Airpush.G (AdWare)
Comodo	ApplicUnwnt@#u4xostpffyi0
Cynet	Malicious (score: 99)
DrWeb	Adware.Airpush.24.origin
ESET-NOD32	A Variant Of Android/Obfus.QY
Fortinet	Android/AirPush
Ikarus	PUA.AndroidOS.AirPush
Lionic	Riskware.AndroidOS.Airpush.z!c
MAX	Malware (ai Score=95)
McAfee	Artemis!00EECCD7FAB4
McAfee-GW-Edition	Artemis
Microsoft	Trojan:Win32/Bitrep.B
NANO-Antivirus	Trojan.Android.Airpush.dgwbpp
Sophos	Android Airpush (PUA)
Symantec	Trojan.Gen.MBT
Symantec Mobile Insight	AdLibrary:Airpush
Tencent	A.gray.mfpad
Trustlook	Android.PUA.General

pua, riskware, trojan>.[1] That is, the software sample with the mentioned hash corresponds to a malware which seems to behave like an *adware* (a software that includes advertisements), a *grayware* (a software in the thin line between a virus and a legitimate software), a *pua* (potentially unwanted application), a *riskware* (a legitimate program that can cause some damage in malicious hands), and a *trojan* (malware that misleads users of its true intent).

As a summary of the two-step multi-labeling procedure, we can observe that:

1. it is simple regarding implementation, as it relies on already available detection engines, and
2. it is robust and flexible, as it integrates several possible malware typologies in just one multi-behavior tuple.

This labeling methodology will ease the process of selecting malware samples to assess particular security solutions, while it will allow to clarify the real composition and representativeness of a given set of malware samples. The specific multi-labeling tool developed is publicly available at https://github.com/nesg-ugr/Multi-Labeling-Malware, where readers can also find some resources and samples used in this work.

3.1. Multi-behavior hypothesis and preliminary labeling results

This section is devoted to roughly validate the multi-behavior hypothesis for malware, so that the introduced multi-labeling methodology is concluded to be coherent with it. For that, some well-known particular malware samples dissected and analyzed in the literature are presented here and, after that, labeled through our proposal in order to compare the supossed behaviors they have with the tags obtained from their classification.

The first malware sample to be studied is one of the fearest and most famous Android malware: the *Anubis Android Banking Trojan* (Ning et al., 2020). In particular, the pandemidestek version:

1. Malware name: Anubis-pandemidestek
2. Apk name: pandemidestek.apk
3. Apk download link: GitHub_ChickenHook_Apk
4. SHA256: 231d970ea3195b3ba3e11e390b6def78a1c8eb5f0a8b7dccc0b4ec4aee9292ec

Based on some detailed analysis performed in the literature (AndroidReverse; GitHub_ChickenHook), we can observe that pandemidestek can present, at least, the following behaviors:

- *Trojan*: Once executed by the user, the app icon disappears from the launcher. Additionally, this app implements mechanisms that prevent it to be uninstalled. It can also prevent with-

[1] We should also probably include in the tuple the term *ransomware*, as *Artemis*, provided by McAfee's detector, corresponds to a variant of this kind of malware.

Table 4
Malicious behaviors or terms considered for malware operation.

adware	apt	backdoor	bank	bomb	bootkit	bot	bug
click	crypto	denial	dial	download	drop	exploit	fraud
gray	grey	hijack	hoax	inject	joke	keylogger	lock
phish	porn	pua	pup	ransom	rat	risk	rootkit
scam	scrap	shim	skim	sms	spam	spy	stego
trojan	vish	worm	zombie	-	-	-	-

Table 5
Malware samples analyzed.

Apk sample (family)	Behaviors reported in the literature (references)	Multi-label according to our proposal (#engines)
Pandemidestek (Anubis)	*Trojan, Bank, Spy, Download, Phish, Keylogger, SMS, Spam* (AndroidReverse; GitHub_ChickenHook)	Trojan (24), Bank (8), Spy (4), Download (2), Adware (1), Pua (1)
bb0hhzjs1 / Her Aile'ye 2000 TL Pandemi Devlet Desteği (Anubis)	*Trojan, Bank, Drop, SMS, Phish, Keylogger* (Blog; Irmak)	Trojan (26), Drop (8), Bank (5), Download (1), Ransom (1), Pup (1), Adware (1)
Fedex (FluBot)	*Trojan, Bank, SMS, Backdoor, Phish* (Incibe-Cert; Prodaft_Flubot)	Trojan (11), Bank (5), Backdoor (3), Risk (2), Drop (1), Download (1), Pua (1)
05637 / Sex xonix (SimpleLocker)	*Trojan, Ransom* (GroDDViewer_SimpleLocker; Kharon_SimpleLocker)	Trojan (16), Ransom (6), Adware (2), Lock (1), Spy (1)
Durak (MobiDash)	*Adware, Trojan* (GroDDViewer_MobiDash; Kharon_MobiDash)	Adware (15), Trojan (7), Pua (4), Pup (3), Risk (1), Spy (1)
6slmduc1o (Toddler)	*Trojan, Bank, SMS, Keylogger, Bot* (Prodaft_Toddler)	Trojan (25), Bank (6), Drop (3), Adware (2), Spy (1), Risk (1), Download (1)

draw permissions. Furthermore, it can boot and reboot, uninstall other apps, disable Play Protect and Lock Screen, etc.
- *Bank*: Pandemidestek, as other samples belonging to the Anubis family, is particularly designed for stealing banking information and performing bank related actions. In this sense, it includes a lot of fake phishing portals for the most famous banks in the world. It uses several techniques (such as looking for the installed apps, faking SMS or geolocation) to detect the possible user's bank apps and accounts to carry out the phishing attack (*e.g.*, with push injections, notification injections combined with screenshots or keyboard record).
- *Spy*: Furthermore, with the ability of taking screenshots and location tracking, the spyware behavior is also exhibited by this malware.
- *Download*: The malware also implements the functionality of finding and uploading files and exfiltrating information, as well as downloading (malicious) files, which is often used to add more functionality to the app.
- *SMS*: The app implements functionality related to SMS stealing too. Moreover, it can enter USSD codes, read the double factor authentication received by SMS, etc.
- *Spam*: In addition, this sample can send SMS spam, even with a link to download the app itself to infect more users (through the contact list).
- *Phish*: As we have previously described for the 'bank' behavior, this sample can behave as a phishing related malware.
- *Keylogger*: Finally, with the possibility of recording keystrokes, this malware sample could also be cataloged within the keylogger typology.

On the other hand, regarding the labels obtained with our multi-labeling procedure for the malware sample, a total of 40 engines positively detect and classify the apk in the following categories:

1. *Trojan*: 24 engines.
2. *Bank*: 8 engines.
3. *Spy*: 4 engines.
4. *Download*: 2 engines.
5. *Adware*: 1 engine.
6. *Pua*: 1 engine.

As observed, four of the eight behaviors potentially exhibited by the Anubis-pandemidestek malware according to the literature are also included in the multi-label tuple derived by our methodology.

A similar analysis is performed for some additional malware samples corresponding to different families. Table 5 shows them as well as the results associated to each, both regarding the behaviors concluded by the community from the dissection of the samples, and the tags derived by applying our approach to each of them. Again, we can see the high correlation and coherence between both classifications and, from that, the validity of our multi-labeling methodology. At this point, however, we can point out the possibility of avoiding some of the labels, specially those provided by a number of engines less than a fixed number, let's say 1 or 2, in order to reduce potential erroneous classifications. The consideration of a given percentage of positive detections over the total obtained could also be considered to limit the number of tags to be assigned to a sample. However, our actual aim now is not to conclude which labels are appropriate or not; instead, that decision will rely on the user/auditor who is thinking about the possibility of using a certain labeled sample in a particular assessment procedure.

From the previous analysis, we can conclude that despite most of malware samples usually have a principal behavior (*Trojan* in the cases above), we can take advantage of other simultaneous behaviors to create more complex and complete malware datasets. In summary, two main facts can be highlighted at this point:

i) a given malware sample can present a number of different behaviors, while
ii) the detection procedures implemented by various engines can help in complementing and hence completing the classification of the sample.

4. Experimentation: Multi-labeling of android malware datasets

Once validated the hypothesis about the usual multi-behavioral nature of malware, in this Section we analyze some public malware datasets in order to obtain the associated multi-tags for the component samples. This will allow us to conclude the real composition and representativeness regarding typology and variety of malware samples for each of the datasets and, from that, the applicability and usability of them.

The specific malware datasets analyzed correspond to four well-known Android malware datasets already mentioned in Section 2:

- AMD (A), with a total of 24,553 samples from 2010 to 2016.
- Drebin (D), composed of 5560 malware samples from 2010 to 2012 and manually identified by the family name they belong to.
- VirusShare (VS), with 51,632 Android malware samples (from a total of around 45,000,000 samples in the dataset), collected between 2016 and 2018.
- VirusTotal (VT), a set of 19,092 Android malware samples from 2017 to 2020 extracted from the corresponding global malware dataset.

The analysis carried out comprises several aspects which are simultaneously performed for each of the datasets for a proper comparison. As a first step, it should be mentioned that despite all the samples provided are supposedly malware, more than 2300 samples in the VirusShare dataset are undetected by all the engines

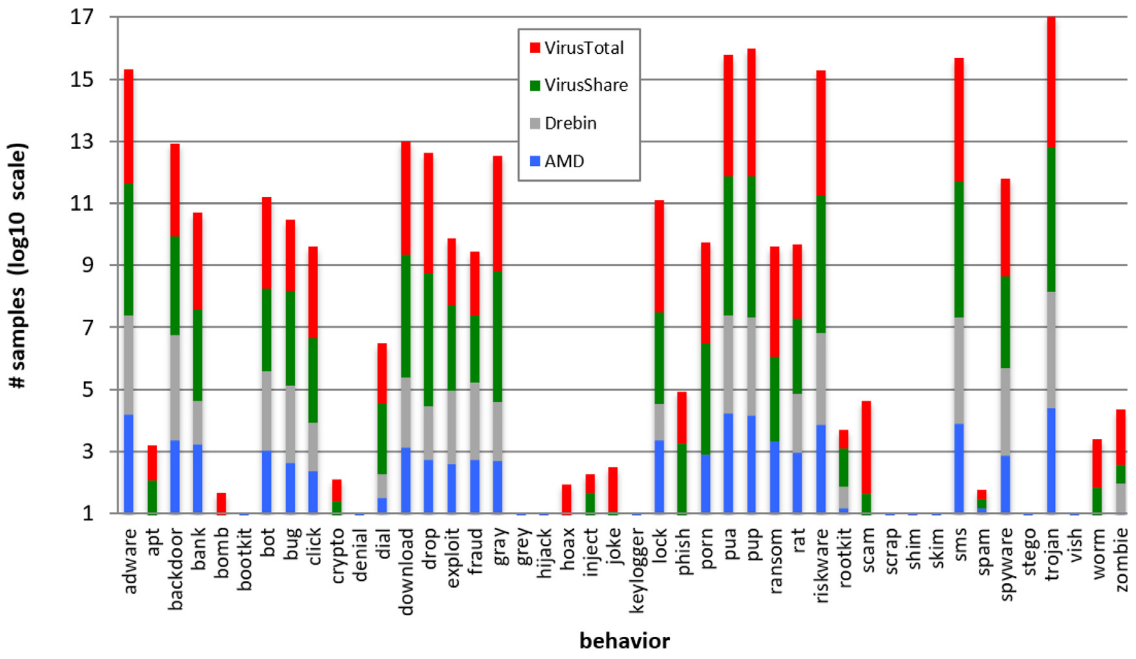

Fig. 1. Distribution of behaviors observed for the malware datasets analyzed.

consulted, while around 1600 are detected just by one or two engines and generally classified as *'suspicious'* or *'apprisk'*. Also Drebin contains 30 malware samples of this second type. They all are removed from the datasets for the study that follows.

Figure 1 shows the number of filehashes (in logarithmic scale for a better visualization) assigned to some of the individual behaviors/typologies considered in Table 4. We can see that behaviors like *adware, backdoor, bank, bot, bug, click, download, drop, exploit, fraud, lock, rat, sms, spyware* and *trojan* appear much more than the rest in all the datasets. Although in a lower quantity (especially for AMD), also the behaviors *dial* and *rootkit* appear in the four cases.

In addition to the mentioned behaviors, it is remarkable that the generic categories *gray, pup/pua* (potentially unwanted program/application) and *riskware* are also recurrent in all the datasets.

In the case of AMD, VirusShare and VirusTotal, the behaviors *porn* and *ransom* appear a number of times too, while none of these two behaviors appear in Drebin. Moreover, VirusShare and VirusTotal have a significant number of *phish* samples, and in a lower quantity *apt, crypto, inject, scam* and *worm* samples. In addition, VirusTotal is the only dataset with an appreciable number of *bomb, hoax* and *joke* samples. AMD and Drebin contain an insignificant number of all of the previous samples, if any.

Some behaviors or types like *keylogger* and *spam* have around a dozen of observations in each of the datasets. Finally, it is remarkable the complete absence in all the datasets of behaviors corresponding to some categories like *bootkit, denial, grey, hijack, scrap, shim, skim, stego* and *vish*.

Besides the appearance of individual behaviors, most of the filehashes are usually assigned with more than one behavior, as previously explained. In fact, some of the samples are assigned with up to 11 behaviors. Figure 2 shows this point, where the most usual number of behaviors per filehash is in the range from 4 to 5 for AMD and Drebin, and in the range from 5 to 8 for VirusShare and VirusTotal. Beyond the values for the typical number of multiple behaviors, it can be observed that VirusShare and VirusTotal present a wider spectrum in terms of number of behaviors per sample. That is, the malware samples in VirusShare and VirusTotal seem to be more complex than those contained in AMD and Drebin.

Table 6
Multi-behaviors differentiated per sample.

Simultaneous behaviors Total (2–11)	AMD 527	Drebin 328	Virus Share 1912	Virus Total 1271
3	56	50	162	102
4	117	89	288	240
5	154	94	382	322
6	107	56	426	276
7	64	20	311	196
8	16	4	183	76
9	1	3	69	32

In the case of AMD, we obtain a total of 527 different tuples or groups of (more than 2) behaviors, this number being equal to 154 when 5 behaviors per filehash is considered, and 117 in the case of 4 behaviors per filehash. Please, compare such numbers of multiple simultaneous behaviors observed with the 135 varieties of 71 malware families in the AMD dataset according to authors. That is, a similar granularity in the number of total (complex) behaviors is obtained through our proposal. Likewise, 328 different tuples or groups of behaviors are obtained for Drebin, this number being equal to 94, 89 and 50 when tuples of 5, 4 and 3 behaviors are considered, respectively. In the case of VirusShare, a total of 1912 different behaviors are observed, which are roughly distributed as follows: 162 for 3 simultaneous behaviors, 288 for 4, 382 for 5, 426 for 6, 311 for 7, and 183 in the case of 8 simultaneous behaviors. Regarding VirusTotal, a total of 1271 different behaviors are observed, which are mainly distributed as follows: 102 for 3 simultaneous behaviors, 240 for 4, 322 for 5, 276 for 6, 196 for 7, and 76 for 8. A summary of the multi-behaviors observed in each case is shown in Table 6.

Finally, Table 7 shows the most typical tuples observed with 2, 3, 4 and 5 behaviors. Due to the generic nature of *pua, pup* and *riskware*, these behaviors often appear in addition to the rest in Table 5 to produce longer tuples. For example, the tuples <*adware, download, drop, pua, pup, riskware, sms, trojan*> and <*adware, bank, download, drop, gray, lock, pua, pup, riskware, trojan*>, with 8 and 10 behaviors respectively, are also observed in the datasets.

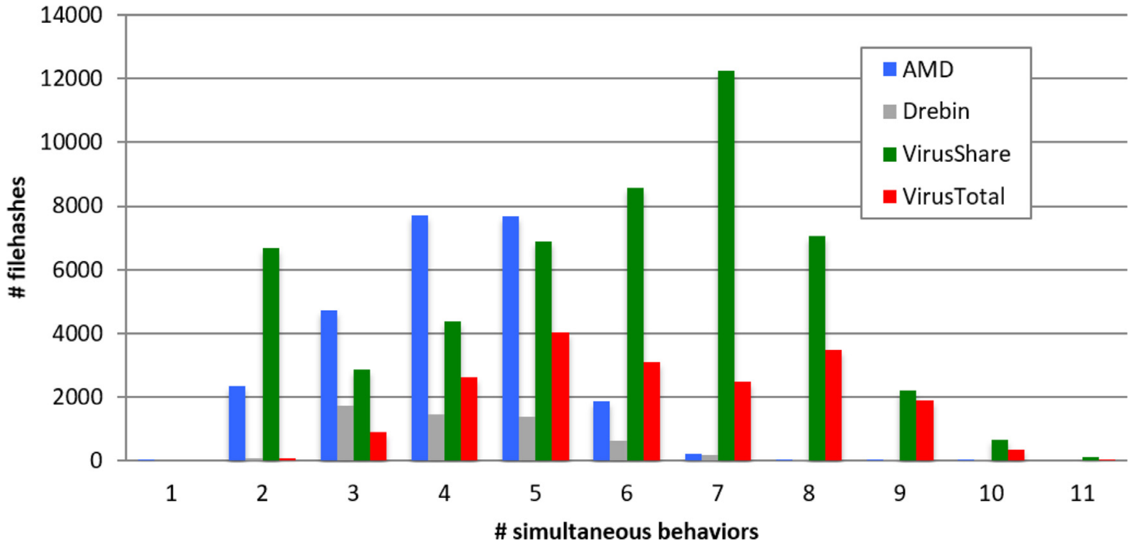

Fig. 2. Distribution of simultaneous behaviors observed per filehash.

Table 7
Tuples observed per dataset.

#Behaviors	Tuples / Datasets
2	<{*adware, backdoor, sms*}, *trojan*> / A,D,VS,VT
	<*bank*, {*bot, spyware*}> / A,VS
	<{*bot, download, drop*}, *trojan*> / VS
	<*fraud, sms*> / D
3	<*adware, backdoor, trojan*> / A,D,VS
	<*adware*,{*drop, smsv*},*trojan*> / VS,VT
	<{*backdoor, bank*}, *sms, trojan*> / A,D,VS
	<*download, drop, trojan*> / VS,VT
	<{*bot, drop*}, *sms, trojan*> / VS,VT
	<*fraud, sms, trojan*> / D
	<*lock, ransom, trojan*> A,VS
	<*sms, spyware, trojan*> / D,VS
4	<*bank*, *bot*, {*sms, spyware*}, *trojan*> / A,VS
	<*bank, bot*, {*drop, sms*}, *trojan*> / VT
	<*download, drop, sms, trojan*> / VS
	<*lock, porn, ransom, trojan*> / A,VT
5	<*adware, bank, bot, sms, trojan*> / VT
	<*backdoor, drop, rat, spyware, trojan*> / VT

5. Further discussion

After verifying the multi-behavior hypothesis for malware samples in Section 3.1, the analysis carried out afterwards for some Android malware datasets in Section 4 concludes as a main fact that VirusShare and VirusTotal are the most complex datasets of all the studied ones, both from the point of view of the volume of malware samples contained as well as from the perspective of the spectrum or typology of malicious behaviors observed. Instead, Drebin contains the lowest quantity of malware samples as well as the lowest number of different malicious behaviors. This can be seen as a natural consequence of the age of each dataset.

The above conclusion is extracted thanks to the behavior-based multi-labeling scheme introduced by authors. Some additional conclusions regarding current labeling procedures for malware samples can be highlighted:

- The classification process usually performed at present to label malware samples assigns just one label per sample, which does not explicitly represent the general multi-behavior exhibited nowadays by malware.
- The classification provided by detection engines can be seen as contradictory regarding the typology concluded (if any!) for the samples analyzed.
- The simple automatic multi-labeling procedure proposed here tries to bypass the previous limitations by complementing the labels provided by different engines to obtain a more global, multiple and realistic tag for each sample. In fact, the results provided are consistent with the variety of families and variants the malware samples analyzed belong to.

In summary, it can be concluded that malware datasets should be improved regarding labeling and often variety. Our automatic multi-labeling procedure can be useful with this purpose, thus allowing to improve current assessment activities.

An additional comment regarding the computational cost involved in the labeling process should also be remarked. The cost of the proposed classification methodology is so low both in time and also regarding resource consumption, as it only consists on a search demand to the VirusTotal's website and a subsequent processing to extract the multiples tags derived by the (73) engines consulted. An estimation of the total classification time concludes a mean value of around 1.05 s/sample, where 0.37 s correspond to the consult to VirusTotal and 0.68 s to the subsequent tags processing, by using a VM with Quad-Core Parrot OS running on a 16GB i7Core-8565U laptop. Beyond these values, the labeling process (multiple or simple) is performed only once (at the beginning) per dataset, so that the cost involved is usually not a relevant issue.

As a potential improvement of our labeling proposal, a smoothing process can be performed by fixing the minimum number of times (either in absolute or in relative value) a given label must be generated by the consulted engines to accept it as a valid tag within the multi-label tuple to be assigned to the sample. For instance, it seems reasonable to remove those labels that appear only once in the classification process. However, this is not a specific goal of the labeling approach and, thus, the prune decision is relegated to the final user interested in utilizing the multi-label scheme.

6. Conclusion and future work

In this work, we present an automatic multi-labeling methodology to classify malware samples. The proposal is supported on the hypothesis that malware samples can present a complex, multi-behavioral nature, which is first validated with some malware samples dissected in the literature. Such a multi-behavior is also concluded for the analyzed samples by our multi-labeling approach, which evidences the consistency, coherence and validity of the proposal.

Once accepted the multi-behavioral nature of malware samples and the coherence with the tuple of labels provided for them, our multi-labeling tool is applied over four well-known Android malware datasets. The analysis shows that the malware datasets should be improved in two principal aspects:

1. Variety, as the behaviors exhibited by some datasets do not cover the current maliciousness spectrum.
2. Labeling, since the tag/class assigned to a given sample does not explicitly represent its usual multiple behavior.

According to the analysis and experimentation performed, two main benefits should be remarked for our proposal:

- Novelty, as, on the contrary to the (mono-)labeling approaches available in the literature, a multi-behavior related tag can be provided per sample.
- Simplicity, as it relies on gathering the labels provided by public detection engines.

As a direct consequence of the above, we conclude that the labeling approach introduced is useful for security testing purposes, as it can help auditors to better adjust at a low cost scenarios and test sets to assess specific security solutions. With this aim, the tool developed is publicly available at https://github.com/nesg-ugr/Multi-Labeling-Malware.

The labeling procedure introduced can be still improved, for instance, by adapting the classes or terms defined in Table 4 to particular goals; possibly combining some of them, including new ones or removing others. Another option is to fix them through the analysis of the labels provided the detection engines and clustering them, so that some order is introduced in the global tags space. All of this, however, is postponed for future work.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

P. García-Teodoro: Conceptualization, Formal analysis, Writing – original draft, Writing – review & editing. **J.A. Gómez-Hernández:** Funding acquisition, Writing – original draft, Writing – review & editing. **A. Abellán-Galera:** Funding acquisition.

Acknowledgement

This work is partially supported by the Spanish Ministry of Economy and Competitiveness and ERDF (European Regional Development Fund) funds through projects TIN2017-83494-R and PID2020-114495RB-I00. Funding for open access charge: Universidad de Granada / CBUA.

References

Abt, S., Baier, H., 2014. Are we missing labels? a study of the availability of ground-truth in network security research. In: Third International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS), pp. 40–55. doi:10.1109/BADGERS.2014.11.

Abusitta, A., Li, M.Q., Fung, B.C.M., 2021. Malware classification and composition analysis: asurvey of recent developments. J. Inf. Secur. Appl. 59, 102828. doi:10.1016/j.jisa.2021.102828.

Adhao, R.B., Pachghare, V., 2021. Network traffic classification using feature selections and two-tier stacked classifier. Int. J. Next-Gener. Netw. 12 (5), 544–550. doi:10.47164/ijngc.v12i5.422.

Agilie. Top mobile security trends to watch in 2022. 2022. Available at https://agilie.com/blog/top-mobile-security-trends-to-watch-in-2020.

Alshaibi, A., Al-Ani, M., Al-Azzawi, A., Konev, A., Shelupanov, A., 2022. The comparison of cybersecurity datasets. Data 7, 1–18. doi:10.3390/data7020022.

Amer, E., Mohamed, S.E., Ashaf, M., Ehab, A., Shereef, O., Metwale, H., Mohammed, A., 2022. Using machine learning to identify android malware relying on API calling sequences and permissions. J. Comput. Commun. 1 (1), 38–47. doi:10.21608/JOCC.2022.218454.

AndroidReverse. Reverse engineering of the anubis malware (pandemidestek) - intended for the Turkish market. Available at https://androidreverse.wordpress.com/2020/06/30/reverse-engineering-of-the-anubis-malware%E2%80%8A-%E2%80%8Apandemistek-intended-for-the-turkish-market/.

Arp, D., Spreitzenbarth, M., Huebner, M., Gascon, H., Rieck, K., 2014. DREBIN: efficient and explainable detection of android malware in your pocket. In: 21th Annual Network and Distributed System Security Symposium (NDSS), pp. 1–15. doi:10.14722/ndss.2014.23247.

Beck, D., Connolly, J., 2006. The common malware enumeration initiative. In: Virus Bulletin Conference. Available at https://www.virusbulletin.com/conference/vb2006/abstracts/common-malware-enumeration-initiative/.

BirghtTALK. Simulation vs emulation: why real attacks matter. 2021. Available at https://www.brighttalk.com/webcast/7451/512384.

Blog. Anubis android malware analysis. Available at https://0x1c3n.tech/anubis-android-malware-analysis.

CARO. A new virus naming convention. Available at http://www.caro.org/articles/naming.html.

CheckPoint. Invisible sandbox evasion. 2022. Available at https://research.checkpoint.com/2022/invisible-cuckoo-cape-sandbox-evasion/.

Cisco, 2020. Cisco Annual Internet Report (2018–2023). White paper. Available at https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html.

Couretas, J.M., 2019. An Introduction to Cyber Modeling and Simulation. Wiley.

Dasgupta, D., Akhtar, Z., Sen, S., 2020. Machine learning in cybersecurity: a comprehensive survey. J. Def. Model. Simul.Appl. Methodol. Technol. 19 (1), 57–106. doi:10.1177/1548512920951275.

Dib, M., Torabi, S., Bou-Harb, E., Assi, C., 2021. A multi-dimensional deep learning framework for IoT malware classification and family attribution. IEEE Trans. Netw. Serv. Manage. 18 (2), 1165–1177. doi:10.1109/TNSM.2021.3075315.

Fiky, A.H.E., Shenawy, A.E., Madkour, M.A., 2021. Android malware category and family detection and identification using machine learning. Cryptogr. Secur. 1–20. doi:10.48550/arXiv.2107.01927.

GdataSoftware. Malware naming hell Part 1: taming the mess of AV detection names. 2019. Available at https://www.gdatasoftware.com/blog/2019/08/35146-taming-the-mess-of-av-detection-names.

Gençaydin, B., Kahya, C.N., Demirkiran, F., Düzgün, B., Çayir, A., Dag, H., 2021. New datasets for dynamic malware classification. Cryptogr. Secur. 1–5. doi:10.48550/arXiv.2111.15205.

GitHub_ChickenHook. Analysis of the anubis malware variant pandemidestek discovered on 12.06.2020. Available at https://github.com/ChickenHook/Anubis-pandemidestek.

GitHub_ChickenHook_Apk. Apk download link of pandemidestek. Available at https://github.com/ChickenHook/Anubis-pandemidestek/blob/master/apk/pandemidestek.apk.

GobalWebIndex. Device globalwebindex's flagship report on device ownership and usage. 2020. Available at https://www.globalwebindex.com/reports/device.

GroDDViewer_MobiDash. Analysis of mobidash. Available at https://cidre.gitlabpages.inria.fr/malware/malware-website/dataset/MobiDash_sample_com.cardgame.durak.html.

GroDDViewer_SimpleLocker. Analysis of a simplelocker sample. Available at https://cidre.gitlabpages.inria.fr/malware/malware-website/dataset/SimpLocker_sample_fd694cf5ca1dd4967ad6e8c67241114c.html.

Gruber, J., Freiling, F., 2022. Fighting evasive malware. Datenschutz und Datensicherheit 46, 284–290. doi:10.1007/s11623-022-1604-9.

Hacks, S., Butun, I., Lagerstrom, R., Buhaiu, A., Georgiadou, A., Michailits, A., 2021. Integrating security behavior into attack simulations. In: 16th International Conference on Availability, Reliability and Security, pp. 1–13. doi:10.1145/3465481.3470475.

Cybersecurity and secure information systems: challenges and solutions in smart environments, 2019. In: Hassanien, A.E., Elhoseny, M. (Eds.), Advances Sciences and Technologies for Security Applications. Springer.

Highnam, K., Arulkumaran, K., Hanif, Z., Jennings, N.R., 2021. BETH dataset: Real cybersecurity data for anomaly detection research. In: Workshop on Uncertainty and Robustness in Deep Learning, pp. 1–8.

Hreirati, O., Iqbal, S., Zulkernine, M., 2018. An adaptive dataset for the evaluation of android malware detection techniques. In: International Conference on Software Security and Assurance (ICSSA), pp. 62–66. doi:10.1109/ICSSA45270.2018.00024.

Hurier, M., Suarez-Tangil, G., Santanu, S.K., Bissyandé, T.F., Traon, Y.L., Cavallaro, L., 2017. Euphony: harmonious unification of cacophonous anti-virus vendor labels for android malware. In: 14th IEEE International Conference on Mining Software Repositories, pp. 425–435. doi:10.1109/MSR.2017.57.

Imtiaz, S.I., Rehman, S., Javed, A.R., Jalil, Z., Liu, X., Alnumay, W.S., 2021. DeepAMD: detection and identification of android malware using high-efficient deep artificial neural network. Future Gener. Comput. Syst. 115, 844–856. doi:10.1016/j.future.2020.10.008.

Incibe-Cert. Flubot malware analysis report. Available at https://www.incibe-cert.es/sites/default/files/contenidos/estudios/doc/incibe-cert_flubot_analysis_study_2021_v1.pdf.

Irmak Y.B.. Anubis android malware analysis report. Available at https://0x1c3n.tech/Anubis%20Android%20Malware%20Analysis%20Report.pdf.

Kantchelian, A., Tschantz, M.C., Afroz, S., Miller, B., Shankar, V., Bachwani, R., Joseph, A.D., Tygar, J.D., 2015. Better malware ground truth: techniques for weighting anti-virus vendor labels. In: Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security (AISec), pp. 45–56. doi:10.1145/2808769.2808780.

Kaspersky. Machine learning for malware detection. report. 2021. Available at https://media.kaspersky.com/en/enterprise-security/Kaspersky-Lab-Whitepaper-Machine-Learning.pdf.

Katrenko, A., Semeniak, E., 2022. Implementing Artificial Intelligence and Machine Learning in Cybersecurity Solutions. Apriorit eBooks.

Kavak, H., Padilla, J.J., Vernon-Bido, D., Diallo, S., Gore, R., Shetty, S., 2021. Simulation for cybersecurity: state of the art and future directions. J. Cybersecur. 7 (1), 1–13. doi:10.1093/cybsec/tyab005.

Kharon_MobiDash. Mobidash analysis study. Available at https://cidre.gitlabpages.inria.fr/malware/malware-website/dataset/malware_MobiDash.html.

Kharon_SimpleLocker. Simplocker analysis study. Available at https://cidre.gitlabpages.inria.fr/malware/malware-website/dataset/malware_SimpLocker.html.

Kim, J., Ban, Y., Ko, E., Cho, H., Yi, J.H., 2022. MAPAS: a practical deep learning-based android malware detection system. Int. J. Inf. Secur. 1–14. doi:10.1007/s10207-022-00579-6.

Kim, S., Jung, W., Lee, K., Oh, H., Kim, E.T., 2022. Sumav: fully automated malware labeling. ICT Express doi:10.1016/j.icte.2022.02.007. In press.

Kim, M., Kim, D., Hwang, C., Cho, S., Han, S., Park, M., 2021. Machine-learning-based android malware family classification using built-in and custom permissions. Appl. Sci. 11, 1–24. doi:10.3390/app112110244.

Lashkari, A.H., Kadir, A.F.A., Taheri, L., Ghorbani, A.A., 2018. Toward developing a systematic approach to generate benchmark android malware datasets and classification. In: Proceedings of the 52nd IEEE International Carnahan Conference on Security Technology (ICCST), pp. 1–7. doi:10.1109/CCST.2018.8585560.

Leszczyna, R., 2021. Review of cybersecurity assessment methods: applicability perspective. Comput. Secur. 108, 1–28. doi:10.1016/j.cose.2021.102376.

Maciá-Fernández, G., Camacho, J., Magán-Carrión, R., García-Teodoro, P., Therón, R., 2018. UGR'16: a new dataset for the evaluation of cyclostationarity-based network IDSs. Comput. Secur. 73, 411–424. doi:10.1016/j.cose.2017.11.004.

, 2009. In: Madhavan, R., Tunstel, W., Messina, E. (Eds.), Performance Evaluation and Benchmarking of Intelligent Systems. Springer.

Maniriho, P., Mahmood, A.N., Chowdhury, M.J.M., 2022. A study on malicious software behaviour analysis and detection techniques: taxonomy, current trends and challenges. Future Gener. Comput. Syst. 130 (C). doi:10.1016/j.future.2021.11.030.

Mills, A., Legg, P., 2020. Investigating anti-evasion malware triggers using automated sandbox reconfiguration techniques. J. Cybersecur. Privacy 1–21. doi:10.3390/jcp1010003.

Mohaisen, A., Alrawi, O., 2014. AV-meter: an evaluation of antivirus scans and labels. In: 11th International Conference Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA), pp. 112–131. doi:10.1007/978-3-319-08509-8_7.

Ning, B., Zhang, G., Zhong, Z., 2020. An evolutionary perspective: a study of anubis android banking trojan. In: 7th International Conference on Dependable Systems and Their Applications (DSA), pp. 141–150. doi:10.1109/DSA51864.2020.00026.

Prodaft_Flubot. Flubot malware analysis report. Available at https://www.prodaft.com/resource/detail/flubot-new-masssive-mobile-malware-ring-targeting-europe.

Prodaft_Toddler. Toddler malware analysis report. Available at https://www.prodaft.com/m/reports/Toddler___TLPWHITE_V2.pdf.

QAnalysts. An overview of ground truth data collection. report. 2022. Available at https://qanalysts.com/an-overview-of-ground-truth-data-collection/.

Rokon, M.O.F., Islam, R., Darki, A., Papalexakis, E.E., Faloutsos, M., 2020. SourceFinder: finding malware source-code from publicly available repositories. RAID 1–16.

Sarker, I.H., Kayes, A.S.M., Badsha, S., Alqahtani, H., Watters, P., Ng, A., 2020. Cybersecurity data science: an overview from machine learning perspective. J. Big Data 7 (41), 1–29. doi:10.1186/s40537-020-00318-5.

Sebastián, M., Rivera, R., Kotzias, P., Caballero, J., 2016. AVClass: a tool for massive malware labeling. In: 19th International Symposium on Research in Attacks, Intrusions, and Defenses (RAID), LNCS 9854, pp. 230–253. doi:10.1007/978-3-319-45719-2_11.

Sihwail, R., Omar, K., Ariffin, K.A.Z., 2018. A survey on malware analysis techniques: static, dynamic, hybrid and memory analysis. Int. J. Adv. Sci.Eng. Inf. Technol. 8 (4-2), 1662–1671. doi:10.18517/ijaseit.8.4-2.6827.

Silvia, S., Caballero, J., 2020. AVclass2: massive malware tag extraction from AV labels. In: Annual Computer Security Applications Conference, pp. 42–53. doi:10.1145/3427228.3427261.

Spreitzenbarth, M., Echtler, F., Schreck, T., Freling, F.C., Hoffmann, J., 2013. Mobile-sandbox: looking deeper into android applications. In: 28th International ACM Symposium on Applied Computing (SAC), pp. 1808–1815. doi:10.1145/2480362.2480701.

Stash. Top 20 breach and attack simulation (BAS) tools. 2022. Available at https://startupstash.com/breach-and-attack-simulation-bas-tools/.

Statista. Share of global smartphone shipments by operating system from 2014 to 2023. 2022. Available at https://www.statista.com/statistics/272307/market-share-forecast-for-smartphone-operating-systems/.

Tardiff, M., Bonheyo, G.T., Cort, K.A., Edgar, T.W., 2016. Applying the scientific method to cybersecurity research. In: IEEE Symposium on Technologies for Homeland Security (HST), pp. 1–9.

Technologies P.. Sandbox detection and evasion techniques. how malware has evolved over the last 10 years. 2021. Available at https://www.ptsecurity.com/ww-en/analytics/antisandbox-techniques/.

Ugarte-Pedrero, X., Graziano, M., Balzarotti, D., 2019. A close look at a daily dataset of malware samples. ACM Trans. Privacy Secur. 22 (1), 1–30. doi:10.1145/3291061.

Veksler, V.D., Buchler, N., Hoffman, B.E., Cassenti, D.N., Sample, C., Sugrim, S., 2018. Simulations in cyber-security: a review of cognitive modeling of network attackers, defenders, and users. Front. Psychol. doi:10.3389/fpsyg.2018.00691.

VirusTotal. API scripts and client libraries. 2022. Available at https://support.virustotal.com/hc/en-us/articles/360006819798-API-Scripts-and-client-libraries#h_10f07166-4521-4730-9910-da4e810ffaf1.

Webtribunal. Mobile and desktop operating systems market share. 2022. Available at https://webtribunal.net/blog/operating-systems-market-share/#gref.

Xin, Y., Kong, L., Liu, Z., Chen, Y., 2018. Machine learning and deep learning methods for cybersecurity. IEEE Access 35365–35381. doi:10.1109/ACCESS.2018.2836950.

Yang, L., Ciptadi, A., Laziuk, I., Ahmadzadeh, A., Wang, G., 2021. BODMAS: an open dataset for learning based temporal analysis of PE malware. In: IEEE Security and Privacy Workshops (SPW), pp. 78–84. doi:10.1109/SPW53761.2021.00020.

Yavanoglu, O., Aydos, M., 2017. A review on cyber security datasets for machine learning algorithms. In: IEEE International Conference on Big Data, Symposium on Data Analytics for Advanced Manufacturing, pp. 2186–2193. doi:10.1109/BigData.2017.8258167.

Yerima, S.Y., Sezer, S., 2019. DroidFusion: a novel multilevel classifier fusion approach for android malware detection. IEEE Trans. Cybern. 49 (2), 453–466. doi:10.1109/TCYB.2017.2777960.

Zelster L.. Free malware sample sources for researchers. Available at https://zeltser.com/malware-sample-sources/.

Zhang, Y., Sui, Y., Zheng, Z., Ning, B., Tsang, I., Zhou, W., 2019. Familial clustering for weakly-labeled android malware using hybrid representation learning. IEEE Trans. Inf. Forensics Secur. 15, 3401–3414. doi:10.1109/TIFS.2019.2947861.

Zhu, S., Shi, J., Yang, L., Qin, B., Zhang, Z., Song, L., Wang, G., 2020. Measuring and modeling the label dynamics of online anti-malware engines. In: 29th USENIX Conference on Security Symposium, pp. 2361–2378.

Zimperium. 2022 global mobile threat report. 2022. Available at https://www.zimperium.com/global-mobile-threat-report/.

Pedro García-Teodoro is a Professor of the Department of Signal Theory, Telematics and Communications of the University of Granada (Spain), and head of the research group "Network Security and Engineering Group (NESG)" of this university. His current professional interest is in the field of computer and network security, especially focused on anomaly-based intrusion detection and denial of service attacks.

José Antonio Gómez-Hernández is an Assistant Professor at the Department of "Languages and Computer Systems" of the University of Granada, and member of the research group "Network Security & Engineering Group (NESG)" and "UGR Cyber Security Group (UCyS)" of the same university. His professional interest is related with operating system security and computer forensics.

Alberto Abellán-Galera is a Telecommunications engineer. He rceived his BSc and MSc in telecommunications engineering from the University of Granada in 2018 and 2020, respectively. He received a starting research grant from this university. He is a collaborator of "Network Engineering and Security Group (NESG)" research group, in the Department of Signal Theory, Telematics and Communication, University of Granada. His research interest is mainly focused on darknets, ethical hacking and network security.