AndroOBFS: Time-tagged Obfuscated Android Malware Dataset with Family Information

Saurabh Kumar
Indian Institute of Technology Kanpur, India
skmtr@cse.iitk.ac.in

Debadatta Mishra
Indian Institute of Technology Kanpur, India
deba@cse.iitk.ac.in

Biswabandan Panda
Indian Institute of Technology Bombay, India
biswa@cse.iitb.ac.in

Sandeep Kumar Shukla
Indian Institute of Technology Kanpur,India
sandeeps@cse.iitk.ac.in

ABSTRACT

With the large-scale adaptation of Android OS and ever-increasing contributions in the Android application space, Android has become the number one target of malware writers. In recent years, a large number of automatic malware detection and classification systems have evolved to tackle the dynamic nature of malware growth using either static or dynamic analysis techniques. Performance of static malware detection methods degrade due to the obfuscation attacks. Although many benchmark datasets are available to measure the performance of malware detection and classification systems, only a single obfuscated malware dataset (PRAGuard) is available to showcase the efficacy of the existing malware detection systems against the obfuscation attacks. PRAGuard contains outdated samples till March 2013 and does not represent the latest application categories. Moreover, PRAGuard does not provide the family information for malware because of which PRAGuard can not be used to evaluate the efficacy of the malware family classification systems.

In this work, we create and release AndroOBFS, a time-tagged (at month granularity) obfuscated malware dataset with familial information spanning over three years from 2018 to 2020. We create this dataset by obfuscating 16279 unique real-world malware from six categories of malware. Out of 16279 obfuscated malware samples, 14579 samples are distributed across 158 families with at least two unique malware samples in each family. We release this dataset to facilitate Android malware study towards designing robust and obfuscation resilient malware detection and classification systems.

CCS CONCEPTS

• **Security and privacy → Malware and its mitigation**; *Mobile and wireless security*.

KEYWORDS

Android, malware, obfuscation, detection, classification

MSR '22, May 23–24, 2022, Pittsburgh, PA, USA

ACM ISBN 978-1-4503-9303-4/22/05…$15.00
https://doi.org/10.1145/3524842.3528013

ACM Reference Format:
Saurabh Kumar, Debadatta Mishra, Biswabandan Panda, and Sandeep Kumar Shukla. 2022. AndroOBFS: Time-tagged Obfuscated Android Malware Dataset with Family Information. In *19th International Conference on Mining Software Repositories (MSR '22), May 23–24, 2022, Pittsburgh, PA, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3524842.3528013

1 INTRODUCTION

With the open-source nature and the extensive support of open application space, Android has become the largest shareholder in the global market of the smartphones, with more than 83% unique devices running worldwide [1]. Its open-source nature and simplicity has gained the attention of manufacturers across the world to produce low-cost smartphone devices compared to other platforms. Apart from smartphones, Android is gaining popularity in the use of other devices such as tablets, TVs, wearable, and recently, IoT devices. Furthermore, the simplicity of the Android framework with regards to the application development has resulted in substantial growth of number of mobile applications developed world wide. A study from Statista shows that every day more than 3.5K Android applications were released in the year 2020 [2].

With the large-scale adaptation of Android OS and ever-increasing contributions in the Android application space, security has become a non-trivial challenge recently. A study published by the AV-TEST shows that around 3.44 million new Android malware were counted in the year 2021 [5]. This indicates that more than 9.4K new Android malware were developed each day during the year 2021 [5]. With the rapid growth in malware in terms of number, variants, and diversity, it is challenging to analyse applications to detect and classify malware manually. Hence, automated malware detection and family classification systems have evolved to scale the dynamic nature of malware growth.

In the past many state-of-the-art automated malware detection and family classification systems ([8–13, 17, 18, 20, 22, 24, 26–29]) have been proposed based on the static and dynamic analysis techniques [14]. One of the primary requirements in designing such systems is the availability of labelled malware. Many such labelled datasets (e.g., Drebin [8], AMD [25], RmvDroid [23], AndroZoo [6]) are available which are used to evaluate the effectiveness of a malware detection and classification system.

The Problem: The accuracy of malware detection and classification systems built around static analysis techniques suffers in

Table 1: Data source and year-wise statistics of malware.

Source	Year	#Samples		
		Non-obfs	Obfs	Obfs (#Family$_{\geq 2}$)
AndroZoo [6] &	2018	9565	6525	5794 (136)
VirusShare.com [3]	2019	9090	8313	7505 (94)
AndroZoo	2020	1724	1441	1280 (44)
Total		**20379**	**16279**	**14579 (158)**

the presence of obfuscated applications. Evaluation of the detection/classification systems against possible obfuscation attacks requires up-to-date obfuscated datasets. The static analysis-based systems either use the PRAGuard [16] dataset or create a new dataset to evaluate the effectiveness against obfuscation attacks that have the following implications.

(i) Most malware detectors like DroidSieve [20] use the PRAGuard dataset to evaluate their efficiency against obfuscated malware. PRAGuard dataset contains malware till March 2013. These samples are outdated and do not represent the current state of applications. Furthermore, the author has stopped the release of the PRAGuard dataset from April 2021 due to maintenance reasons.

(ii) Malware family information is not present, or the representative sample size is small in the available PRAGuard dataset. Due to this limitation, existing malware family identification techniques [8, 10, 13, 20, 22] do not evaluate their solution against the obfuscated attacks.

(iii) Most malware detectors [12, 21] create new obfuscated samples to demonstrate their sustainability against possible obfuscation attacks. Generally, these self-created datasets are not publicly available, which hampers the reproducibility of results, a crucial aspect of progress in any research domain. **(iv)** During the evaluation, a detection system must be evaluated against samples born after the training data samples. However, the existing datasets do not contain date-time (birth) information about the malware sample on which it appears in the Android application space. A time-tagged dataset may help to choose testing samples appropriately to showcase the efficiency of the detection system in the real scenarios.

Our goal: We believe a timestamped obfuscated Android malware dataset representing current state of Android applications and family information will benefit the malware research community. Our dataset will help to develop efficient malware detection and classification tools, and evaluate the performance of different detection techniques to scale the dynamic nature of malware growth.

Our approach: We create and release a time-tagged (at the granularity of month) obfuscated malware dataset with 16279 unique real-world malware in six different categories along with family names. The dataset spans over three years, from 2018 to 2020. 14579 obfuscated malware samples out of 16279 are distributed across 158 families. To create this dataset, we use the following strategy:

(i) We collect real-world malware samples and separate them by year, quarter, and month. We exclude all malware whose detection count on VirusTotal [4] is less than 10.

(ii) We label all malware with their family name with the help of VirusTotal report and AVclass [19] tool.

(iii) We obfuscate all the malware with six different categories using the Obfuscapk [7] tool.

We store all the information related to obfuscated malware with family in two CSV files; one CSV file corresponds to 16279 samples and the other for 14579 familial malware samples. The AndroOBFS dataset along with both the CSV files are available at the web link https://www.doi.org/10.21227/9ptx-5d17.

Table 2: Obfuscators implemented in Obfuscapk[7] tool.

Category	Obfuscators
trivial	Randomize manifest file, rebuild, new alignment, re-signing
renaming	Renaming the class, fields and methods
encryption	Encryption of library, resource strings, assets, and constant strings
reflection	Invoke user defined and framework APIs using the reflection APIs
code	Junk code insertion, instruction re-ordering, calls redirection, removing debug data, insertion of goto instruction, adding new method by exploiting method overloading.

2 DATA COLLECTION AND DATASET CREATION

In this section, we describe the source of non-obfuscated malware, tools used to create the dataset, and the process of creating the obfuscated dataset.

2.1 Data Source

For this work, non-obfuscated malware samples were collected from two sources—*(i)* AndroZoo [6] Project, and *(ii)* VirusShare.com [3], spanning over three years from 2018 to 2020. Samples for the year 2020 were collected only from the AndroZoo project. All these samples were examined through VirusTotal [4] to ensure that they were malware. We eliminated all the samples flagged as malware by less than 10 antivirus (AV) engines at VirusTotal. In total, we obtained 20379 unique non-obfuscated real-world malware samples. Column name "**Non-obfs**" of Table 1 shows year-wise statistics of non-obfuscated malware.

2.2 Tools Used

We utilized two tools to produce an obfuscated malware dataset annotated with their respective family names—*(i)* AVclass [19] and *(ii)* Obfuscapk [7]. The following is a description of these tools:

AVclass [19] is a tool for labeling the malware with their family name. It operates on the AV labels for a large number of malware samples, e.g., VirusTotal JSON reports, and outputs the family name of each sample from the AV labels. Please refer to the paper [19] for more details about the AVclass.

Obfuscapk [7] is an open-source black-box obfuscation tool for Android applications. In Obfuscapk, obfuscation techniques are classified into five categories. The possible obfuscators implemented in different obfuscation category of Obfuscapk are shown in Table 2. Please refer to the paper [7] for more details about the Obfuscapk.

2.3 Dataset Creation Process

We use four steps to create obfuscated dataset with familial information as explained below.

(i) Data Collection: We obtained non-obfuscated malware samples through the AndroZoo project and VirusShare.com. After collecting samples, we eliminated samples that were detected by less than 10 AV engines at VirusTotal, leaving 20379 unique non-obfuscated samples.

(ii) Separating in Time-tagged Structure: We separated each malware sample depending on the year, quarter, and month by looking at the last modification date of Dex file. The malware's time of birth was determined by the Dex file modification date.

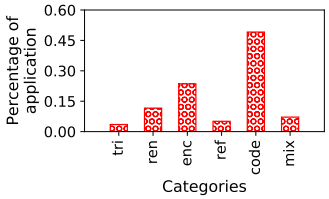

Figure 1: Distribution of malware samples across obfuscation categories [tri: trivial, ren: renaming, enc: encryption, ref: reflection].

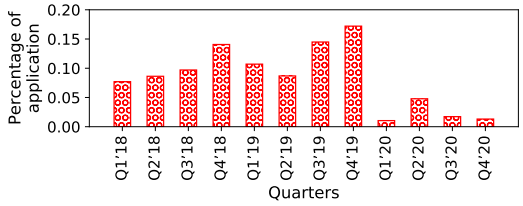

Figure 2: Quarter-wise distribution of malware samples spanning over three years (2018 to 2020).

(iii) Labeling with Family Information: After separating malware into the time-tagged structure, we looked for the family name of each malware sample. To determine the family name, we first obtained the VirusTotal JSON reports by sending the SHA-256 hash of each sample to VirusTotal. After getting the VirusTotal reports, we passed these reports to the AVclass tool to assign family names to each sample and store them in CSV files.

(iv) Obfuscating Malware Samples: Finally, using the Obfuscapk tool, every non-obfuscated malware sample was obfuscated month-by-month for each year (i.e., 2018, 2019, and 2020) in six different categories. The first five categories are the same as offered by Obfuscapk (see Table 2), while the sixth category includes a combination of two or more obfuscation techniques (referred to as mix). The Obfuscapk tool failed for several malware samples due to the application complexity during the obfuscation process. As a result, we correctly acquired 16279 unique obfuscated samples from 20379 non-obfuscated samples. The column name "**Obfs**" of Table 1 shows year-wise statistics of obfuscated malware, including families with at least one sample. However, after removing all malware families containing a single sample, we were left with 14579 unique obfuscated malware samples distributed across 158 families. The column name "**Obfs (#Family$_{\geq 2}$)**" of Table 1 shows year-wise statistics of obfuscated malware where each malware family contains at least two unique samples.

3 OVERVIEW OF DATASET

In this section, we provide an overview of AndroOBFS dataset along with the distribution of malware and malware families.

3.1 Overview

Table 1 shows the year-wise statistics of AndroOBFS dataset. We have created this dataset by obfuscating 16279 unique real-word malware in six different obfuscation categories that have been

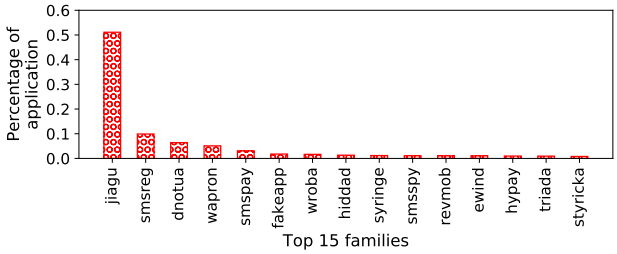

Figure 3: Distribution of top 15 families in familial AndroOBFS.

flagged as malware by at least 10 antivirus engines of VirusTotal. To obfuscate malware, we have used Obfuscapk Tool. Furthermore, for the familial obfuscated malware dataset, we only consider those families with at least two unique samples. After that, we were left with 14579 unique obfuscated malware with family information tagged with their time of birth. We release the entire dataset including all time-tagged samples and samples with family information at the web link https://www.doi.org/10.21227/9ptx-5d17.

3.2 Malware Distribution

We show the malware distribution in AndroOBFS dataset by considering the entire set of obfuscated samples, i.e., 16279 unique real-word malware sample.

3.2.1 Distribution Across Obfuscation Categories. Figure 1 shows the distribution of malware samples under six different obfuscation categories—(i) trivial (tri), (ii) renaming (ren), (iii) encryption (enc), (iv) reflection (ref), (v) code, and (vi) mix (a mix of two or more obfuscation method from (i) to (v)). In our dataset, ~50% of samples fall under code obfuscation as code obfuscation techniques are widely used by malware authors to bypass the static analysis process. We give second priority to encryption where ~23% malware are encrypted.

3.2.2 Time-tagged Distribution. A time-tagged dataset should contain enough samples to perform a longitudinal study to understand the attack scenarios and the effectiveness of counter measures used by any security system. Figure 2 shows that AndroOBFS contain enough samples for each quarter spanning over three years from 2018 to 2020. The portion of samples for 2020 however is small but sufficient to perform a longitudinal study. In future, we will add more samples for the year 2020 when we collect new malware samples of that year.

3.3 Distribution of Malware Families

One of our aims is to create familial obfuscated malware samples. Hence, we show the distribution of malware families by using 14579 unique samples distributed across 158 families.

3.3.1 Malware Families Distribution. Malware in AndroOBFS are distributed among 158 different families having at least two or more unique samples. Out of 158 families, we show the distribution of the top 15 malware families in Figure 3. The `jiagu` family alone holds more than 51% of the entire familial obfuscated dataset, while the share of top 5 families (i.e., `jiagu`, `smsreg`:~9.9%, `dnotua`:~6.4%,

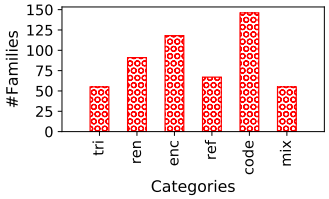

Figure 4: Number of unique families across six different obfuscation categories.

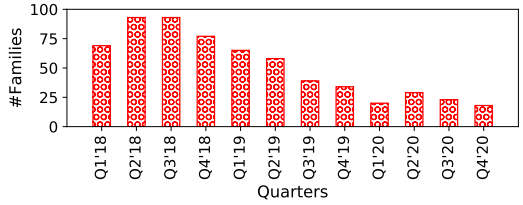

Figure 5: Quarter-wise number of families spanning over three years (2018 to 2020).

wapron:~5%, and smspay:~3%) in the AndroOBFS dataset is ~75%. Out of 158 families, 146 families individually holds less than 1% samples of AndroOBFS.

3.3.2 Number of Families Across Obfuscation Categories. We have shown the distribution of malware samples into families. Our dataset contains obfuscated samples in six different categories. Figure 4 shows the number of unique malware families with different obfuscation techniques. Similar to malware distribution across categories (see Section 3.2.1), the highest number of families are represented by the code obfuscation method, followed by the encryption method. The code obfuscated and encrypted malware samples are distributed across 146 and 118 families, respectively.

3.3.3 Number of Families in Time-tagged Familial Dataset. We further categorize quarter-wise distribution of malware samples into for years 2018, 2019, and 2020, as shown in Figure 5. Similar to the time-tagged malware distribution, our familial dataset also contains enough families for every quarter. A sufficient number of malware families will help to evaluate the family classification system against obfuscation attacks.

4 USAGE SCENARIO

In this section, first, we describe the prior work that utilizes the part of this dataset, followed by the future usage scenario of the entire dataset.

4.1 Prior Usage

To enable on-device Android malware detection, we have proposed DeepDetect [12]. DeepDetect is a practical on-device malware detector that employs a machine learning algorithm on static features, which consumes significantly less processing time and device energy. To study the resiliency of DeepDetect against the possible obfuscation attacks, we have used 4993 obfuscated malware samples from the AndroOBFS dataset for the year 2019. Other than this work, we have not used this dataset in the past.

4.2 Future Usage

We believe our dataset can be utilized in the following scenarios:
(i) Robust Malware Detection and Classification: As described earlier, most malware detectors use the PRAGuard dataset to study the robustness of their technique against the obfuscated malware. PRAGuard is outdated and does not represent the latest applications in use. Also, the existing family classification models have never been evaluated against the possible obfuscation attacks due to the non-availability of obfuscated familial malware samples. Hence, AndroOBFS will help in designing robust and obfuscation resilient malware detection and classification systems.
(ii) Studying the Efficacy of Existing Analysis System: The efficacy of existing detection systems has been evaluated on outdated obfuscated datasets. The impact of the current application design approaches along with obfuscation is unknown on the existing malware detection solutions. Hence, our dataset can be utilized to study the efficacy of existing malware detection and family classification system against the current state of Android applications and obfuscation methods.
(iii) Temporal Study: This study is important to understand how long a security system can work without requiring an update. Existing malware detection and classification systems have performed a temporal study on the non-obfuscated datasets. However, none of the existing solutions have opted for temporal study against the obfuscation attacks due to the non-availability of time-tagged obfuscated samples. Therefore, AndroOBFS dataset will enable temporal study against possible obfuscation attacks.

5 LIMITATIONS

The time-tagged obfuscation dataset released as part of this work faces several limitations which can be improved further, as discussed below:
(i) Our dataset enables temporal study by providing a dataset tagged with time till the granularity of months. We use Dex file modification time to separate malware in timeline order. Release time inconsistency may threaten our dataset where the malware author intentionally alters the Dex modification date. Our dataset can be further improved by using the MoonlightBox [15] tool which analyses API release history to overcome this limitation.
(ii) As mentioned in Section 3.2.2, the year 2020 contains fewer samples, resulting in no obfuscated malware sample under a specific obfuscation method. For example, in February 2020, we did not have any obfuscated samples under the mix category and had only one sample under the reflection category. However, this limitation is easily addressable depending on the availability of new malware samples.
(iii) Our familial dataset contains malware families with at least two unique samples. Therefore, some malware families do not hold obfuscated samples in all categories. However, it can be further improved by increasing the family size threshold from two to some value decided based on the requirement.

(iv) It is be possible that some of the obfuscated malware samples may crash during the execution. Information regarding malware samples that crashed during execution is not available with the AndroOBFS dataset. In the future, we will annotate the dataset with execution status to overcome this limitation.

6 CONCLUSION

This paper presents AndroOBFS, a time-tagged (at the granularity of month) obfuscated malware dataset with familial information based on real-world malware, to aid the research study on malware detection and family classification systems. AndroOBFS was created by obfuscating 16279 unique malware samples in six different obfuscation categories, spanning over three years from 2018 to 2020. 14579 out of 16279 obfuscated malware samples are distributed among 158 families with at least two unique malware samples in the entire dataset. We believe the Android malware research community will benefit from this dataset to design robust obfuscation resilient malware detection and family classification systems.

ACKNOWLEDGMENTS

This work is partially supported by Visvesvaraya Ph.D. Fellowship grant MEITY-PHD-999, SERB and DST through C3i center and C3i hub projects at IIT Kanpur.

REFERENCES

[1] 2021. *IDC: Smartphone Market Share-OS.* https://www.idc.com/promo/smartphone-market-share/os Accessed: 12 January 2022.
[2] 2021. *Number of daily Android app releases worldwide | statista.com.* https://www.statista.com/statistics/276703/android-app-releases-worldwide/ Accessed: 15 January 2022.
[3] 2021. *VirusShare.com.* https://virusshare.com/ Accessed: 10 October 2021.
[4] 2021. *VirusTotal.* https://www.virustotal.com/ Accessed: 10 October 2021.
[5] 2022. *Malware Statistics & Trends Report | AV-TEST.* https://www.av-test.org/en/statistics/malware/ Accessed: 1 January 2022.
[6] Kevin Allix, Tegawendé F. Bissyandé, Jacques Klein, and Yves Le Traon. 2016. AndroZoo: Collecting Millions of Android Apps for the Research Community. In *2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR)*. 468–471.
[7] Simone Aonzo, Gabriel Claudiu Georgiu, Luca Verderame, and Alessio Merlo. 2020. Obfuscapk: An open-source black-box obfuscation tool for Android apps. *SoftwareX* 11 (2020), 100403.
[8] Daniel Arp, Michael Spreitzenbarth, Malte Hübner, Hugo Gascon, and Konrad Rieck. 2014. DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket. In *Symposium on Network and Distributed System Security (NDSS)*. NDSS. https://doi.org/10.14722/ndss.2014.23247
[9] Anam Fatima, Saurabh Kumar, and Malay Kishore Dutta. 2021. Host-Server-Based Malware Detection System for Android Platforms Using Machine Learning. In *Advances in Computational Intelligence and Communication Technology*, Xiao-Zhi Gao, Shailesh Tiwari, Munesh C. Trivedi, and Krishn K. Mishra (Eds.). Springer Singapore, Singapore, 195–205.
[10] Hye Min Kim, Hyun Min Song, Jae Woo Seo, and Huy Kang Kim. 2018. Andro-Simnet: Android Malware Family Classification using Social Network Analysis. In *2018 16th Annual Conference on Privacy, Security and Trust (PST)*. 1–8. https://doi.org/10.1109/PST.2018.8514216
[11] Saurabh Kumar, Debadatta Mishra, Biswabandan Panda, and Sandeep K. Shukla. 2020. STDNeut: Neutralizing Sensor, Telephony System and Device State Information on Emulated Android Environments. In *Cryptology and Network Security*, Stephan Krenn, Haya Shulman, and Serge Vaudenay (Eds.). Springer International Publishing, Cham, 85–106.
[12] Saurabh Kumar, Debadatta Mishra, Biswabandan Panda, and Sandeep Kumar Shukla. 2021. DeepDetect: A Practical On-device Android Malware Detector. In *2021 IEEE 21st International Conference on Software Quality, Reliability and Security (QRS)*. 40–51. https://doi.org/10.1109/QRS54544.2021.00015
[13] Saurabh Kumar, Debadatta Mishra, and Sandeep Kumar Shukla. 2021. Android Malware Family Classification: What Works – API Calls, Permissions or API Packages?. In *2021 14th International Conference on Security of Information and Networks (SIN)*, Vol. 1. 1–8. https://doi.org/10.1109/SIN54109.2021.9699322
[14] Saurabh Kumar and Sandeep Kumar Shukla. 2020. *The State of Android Security*. Springer Singapore, Singapore, 17–22. https://doi.org/10.1007/978-981-15-1675-7_2
[15] Li Li, Tegawendé Bissyandé, and Jacques Klein. 2018. MoonlightBox: Mining Android API Histories for Uncovering Release-Time Inconsistencies. In *2018 IEEE 29th International Symposium on Software Reliability Engineering (ISSRE)*. 212–223. https://doi.org/10.1109/ISSRE.2018.00031
[16] Davide Maiorca, Davide Ariu, Igino Corona, Marco Aresu, and Giorgio Giacinto. 2015. Stealth Attacks: An Extended Insight into the Obfuscation Effects on Android Malware. *Comput. Secur.* 51, C (jun 2015), 16–31. https://doi.org/10.1016/j.cose.2015.02.007
[17] Luca Massarelli, Leonardo Aniello, Claudio Ciccotelli, Leonardo Querzoni, Daniele Ucci, and Roberto Baldoni. 2017. Android malware family classification based on resource consumption over time. In *2017 12th International Conference on Malicious and Unwanted Software (MALWARE)*. 31–38. https://doi.org/10.1109/MALWARE.2017.8323954
[18] Andrea Saracino, Daniele Sgandurra, Gianluca Dini, and Fabio Martinelli. 2018. MADAM: Effective and Efficient Behavior-based Android Malware Detection and Prevention. *IEEE Transactions on Dependable and Secure Computing* 15, 1 (Jan 2018), 83–97. https://doi.org/10.1109/TDSC.2016.2536605
[19] Marcos Sebastián, Richard Rivera, Platon Kotzias, and Juan Caballero. 2016. AVclass: A Tool for Massive Malware Labeling. In *Research in Attacks, Intrusions, and Defenses*, Fabian Monrose, Marc Dacier, Gregory Blanc, and Joaquin Garcia-Alfaro (Eds.). Springer International Publishing, Cham, 230–253.
[20] Guillermo Suarez-Tangil, Santanu Kumar Dash, Mansour Ahmadi, Johannes Kinder, Giorgio Giacinto, and Lorenzo Cavallaro. 2017. DroidSieve: Fast and Accurate Classification of Obfuscated Android Malware. In *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy* (Scottsdale, Arizona, USA) *(CODASPY '17)*. Association for Computing Machinery, New York, NY, USA, 309–320. https://doi.org/10.1145/3029806.3029825
[21] Caijun Sun, Hua Zhang, Sujuan Qin, Jiawei Qin, Yijie Shi, and Qiaoyan Wen. 2020. DroidPDF: The Obfuscation Resilient Packer Detection Framework for Android Apps. *IEEE Access* 8 (2020), 167460–167474. https://doi.org/10.1109/ACCESS.2020.3010588
[22] Sercan Türker and Ahmet Burak Can. 2019. AndMFC: Android Malware Family Classification Framework. In *2019 IEEE 30th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC Workshops)*. 1–6. https://doi.org/10.1109/PIMRCW.2019.8880840
[23] Haoyu Wang, Junjun Si, Hao Li, and Yao Guo. 2019. RmvDroid: Towards A Reliable Android Malware Dataset with App Metadata. In *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. 404–408. https://doi.org/10.1109/MSR.2019.00067
[24] Xiaolei Wang, Yuexiang Yang, and Yingzhi Zeng. 2015. Accurate mobile malware detection and classification in the cloud. *SpringerPlus* 4 (12 2015). https://doi.org/10.1186/s40064-015-1356-1
[25] Fengguo Wei, Yuping Li, Sankardas Roy, Xinming Ou, and Wu Zhou. 2017. Deep Ground Truth Analysis of Current Android Malware. In *Detection of Intrusions and Malware, and Vulnerability Assessment*, Michalis Polychronakis and Michael Meier (Eds.). Springer International Publishing, Cham, 252–276.
[26] Ke Xu, Yingjiu Li, Robert H. Deng, and Kai Chen. 2018. DeepRefiner: Multi-layer Android Malware Detection System Applying Deep Neural Networks. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*. 473–487.
[27] Lok Kwong Yan and Heng Yin. 2012. DroidScope: Seamlessly Reconstructing the OS and Dalvik Semantic Views for Dynamic Android Malware Analysis. In *21st USENIX Security Symposium (USENIX Security 12)*. USENIX Association, Bellevue, WA, 569–584.
[28] Suleiman Y. Yerima, Sakir Sezer, and Igor Muttik. 2014. Android Malware Detection Using Parallel Machine Learning Classifiers. In *2014 Eighth International Conference on Next Generation Mobile Apps, Services and Technologies*. 37–42. https://doi.org/10.1109/NGMAST.2014.23
[29] Zhenlong Yuan, Yongqiang Lu, and Yibo Xue. 2016. DroidDetector: Android malware characterization and detection using deep learning. *Tsinghua Science and Technology* 21, 1 (2016), 114–123. https://doi.org/10.1109/TST.2016.7399288