A study on the role of feature selection for malware detection on Android applications

Catarina Palma[1]
A45241@alunos.isel.pt

Artur Ferreira[1][2]
artur.ferreira@isel.pt

Mário Figueiredo[2][3]
mario.figueiredo@tecnico.ulisboa.pt

[1] ISEL, Instituto Superior de Engenharia de Lisboa
Instituto Politécnico de Lisboa
Lisboa, PORTUGAL

[2] Instituto de Telecomunicações
Lisboa, PORTUGAL

[3] IST, Instituto Superior Técnico
Universidade de Lisboa
Lisboa, PORTUGAL

Abstract

The presence of malicious software (malware) in Android applications (apps) has harmful or irreparable consequences to the user and/or the device. Despite the protections provided by app stores, malware keeps growing in both sophistication and diffusion. This paper explores the use of *machine learning* (ML) and *feature selection* (FS) approaches to detect malware in Android applications using public domain datasets. We resort to the *relevance-redundancy FS* (RRFS) filter method using the unsupervised *mean-median* (MM) and the supervised *Fisher ratio* (FR) relevance measures. Our approach is able to reduce the dimensionality of the data, achieve high accuracy and recall values, and identify the most decisive features to classify an app as malware.

1 Introduction

The use of smartphones has grown exponentially in the past decade. This growth has been accompanied by the popularisation of Android, an open-source *operating system* (OS), being one of the most popular mobile OS, with 70% of mobile phones using Android [8]. In the third quarter of 2022, the Google Play Store had approximately 3.5 million apps available [1]. Popular apps possess thousands or millions of users and deal with substantial user-sensitive data, making them lucrative targets for malicious attacks against Android mobile devices, which have been increasing. For instance, in 2020, Kaspersky detected 5.7 million malware Android packages, about three times more than in 2019 (2.1 million) [3].

Some software and apps focus on security, and major app stores have security and detection mechanisms to mitigate malicious apps. Despite their partial success, malware keeps growing in both sophistication and diffusion, occasionally bypassing them. Thus, the need to detect malicious applications is a major issue which can be tackled by ML techniques.

This paper explores the use of ML techniques for malware detection in Android applications, focusing on FS, using public domain datasets. The RRFS filter is applied with two different relevance measures, namely the unsupervised MM and the supervised FR techniques. The goal is to reduce dimensionality and identify the most decisive features to classify an app as malicious while achieving a good performance of the ML model in malware detection for Android applications.

The remainder of this paper is organised as follows. Section 2 provides an overview of the related work. The proposed approach is described in Section 3. The experimental evaluation is reported in Section 4. Finally, Section 5 ends the paper with concluding remarks.

2 Related Work

The detection of malware in Android applications can follow these approaches: static, dynamic, and hybrid. In static analysis, the application is analysed without being executed. Dynamic analysis occurs during the app's regular operation and is usually performed in monitored or sandbox environments to analyse its behaviour. Finally, the hybrid analysis combines the previous two types of approach [8]. Wu *et al.* [9] surveyed ML-based Android malware detection papers from 2016 to November 2020 and concluded that static analysis is the most popular, followed by dynamic and hybrid analysis. Kouliaridis and Kambourakis [7] reached the same conclusion.

These different malware detection approaches are essential to build datasets for malware detection in Android applications, such as Drebin, CICAndMal2017, MalGenome, and VirusShare [9]. The Drebin dataset was first published in 2014 and contains malware apps from 179 families. The CICAndMal2017 dataset was released in 2018 [2], with samples from 42 malware families. Unlike the Drebin dataset, CICAndMal2017 has numerical and categorical features and static and dynamic features obtained via static and dynamic analysis, respectively. Alkahtani and Aldhyani [3] performed experiments with both the Drebin and CICAndMal2017 datasets and concluded that the *support vector machine* (SVM) classifier achieved the best results.

Keyvanpour *et al.* [6] conducted experiments with the Drebin dataset. The authors applied three FS strategies (effective, high weight, and effective group FS) and concluded that effective FS led to the best results. The authors proposed embedding FS with *random forest* (RF), which outperformed other models. FS has been shown to improve the performance of the RF classifier. Islam *et al.* [5] utilised the CCCS-CIC-AndMal2020 dataset, with 12 major malware categories, 53439 instances, and 141 features. To perform FS, the authors applied *recursive feature elimination* (RFE), discarding 60.2% of the features and achieving 95% accuracy. Wu *et al.* [9] found that the most popular classifier for malware detection was SVM, followed by RF. Kouliaridis and Kambourakis [7] also analysed several studies, concluding that RF is the most used classifier in the literature, followed by SVM. Thus, the SVM and RF classifiers are considered in this paper.

3 Proposed Approach

The approach followed in this study is depicted in Figure 1. The Android app data is obtained from a dataset (Drebin or CICAndMal2017). Then, data pre-processing methods and techniques are applied, in which the use of the RRFS method is included, and data splitting is performed to prepare and organise the data properly. Three subsets result from the data splitting phase: the train, test, and validation sets. The SVM and RF classifiers are used to label the input data pattern as 'benign' or 'malicious'. Thus, we have a binary classification problem for malware detection.

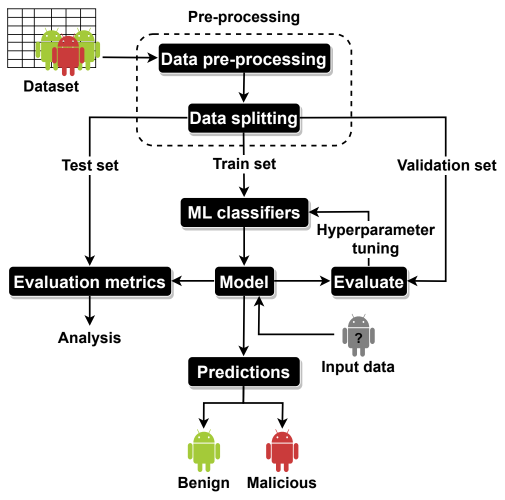

Figure 1: Block diagram of the proposed approach for malware detection.

4 Experimental Evaluation

4.1 Dataset Characteristics and Evaluation Metrics

The Drebin dataset has $n = 15036$ instances and $d = 215$ features, with no missing values, and all its features are numerical (most of them are binary). It has a ratio between class labels of approximately one-third, with the majority belonging to the 'benign' class label. The CICAndMal2017 dataset has $n = 29999$ instances and $d = 183$ features. It contains missing values (around 200) and both numerical and categorical features. It also has a ratio between class labels of approximately one-third, with the majority belonging to the 'malicious' class label. We assess the performance of the ML models with accuracy defined as

$$Acc = \frac{TN + TP}{TN + TP + FN + FP}, \tag{1}$$

in which a *true positive* (TP) means to classify a malicious app as malicious correctly, a *true negative* (TN) is to classify a benign app as benign, a *false positive* (FP) is to classify a benign app as malicious. A *false negative* (FN) refers to classifying a malicious app as benign. Although we aim for high accuracy, this can be misleading with imbalanced data. Thus, we have also assessed the sensitivity or Recall given by

$$Rec = \frac{TP}{TP + FN}. \tag{2}$$

4.2 Experimental Results

A 70-30 split for training and testing is used. In the data pre-processing stage, each categorical feature was converted to numeric through label encoding, and the mean imputation method was applied to deal with missing values. Min-Max normalisation was applied to set all feature values in the 0 to 1 range. The *relevance-redundancy FS* (RRFS) filter approach [4] was applied, with the unsupervised *mean-median* (MM) relevance metric given by

$$MM_i = |\overline{X_i} - median(X_i)|, \tag{3}$$

with $\overline{X_i}$ denoting the sample mean of feature X_i. We also consider the supervised Fisher ratio (FR) relevance metric

$$FR_i = \frac{\left|\overline{X}_i^{(-1)} - \overline{X}_i^{(1)}\right|}{\sqrt{\text{var}(X_i)^{(-1)} + \text{var}(X_i)^{(1)}}}, \tag{4}$$

where $\overline{X}_i^{(-1)}$, $\overline{X}_i^{(1)}$, $\text{var}(X_i)^{(-1)}$, and $\text{var}(X_i)^{(1)}$, are the sample means and variances of feature X_i, for the patterns of each class. The redundancy analysis between two features, X_i and X_j, is done with the *absolute cosine*

$$AC_{X_i,X_j} = |cos(\theta_{X_iX_j})| = \left|\frac{\langle X_i, X_j\rangle}{||X_i||||X_j||}\right| = \frac{\sum_{k=1}^{n} X_{ik}X_{jk}}{\sqrt{\sum_{k=1}^{n} X_{ik}^2 \sum_{k=1}^{n} X_{jk}^2}}, \tag{5}$$

where $\langle,\rangle$ and $||.||$ denote the inner product and L_2 norm, respectively. The baseline results of SVM and RF are reported in Table 1. FR yields better results than MM for both Acc and Rec metrics, which seems to indicate that the use of the class label provides valuable information for the FS task. The baseline results and the results obtained after applying RRFS with FR do not differ much, but mostly, they worsen slightly by using RRFS with FR. However, these slight drops in Acc and Rec are arguably compensated by the dimensionality reduction. Table 2 reports the original number of features, d, and the number of reduced features denoted as m. In both datasets, the number of features was significantly reduced with RRFS. The Acc and Rec metrics show slight differences as with the original number of features. The RRFS approach also allows insight into the most relevant features of each dataset. We present the four most relevant according to RRFS with FR and MM to classify malware on each dataset. For the Drebin dataset, RRFS (MM) selects: *Landroid.content.Context.registerReceiver*, *android.telephony.SmsManager*, *GET_TASKS*, and */system/app*. RRFS (FR) selects: *transact*, *SEND_SMS*, *Ljava.lang.Class.getCanonicalName*, and *android.telephony.SmsManager*. On the CICAndMal2017 dataset, RRFS (MM) selects: *Network communication : view network state*, *Services that cost you money : directly call phone numbers*, *Hardware controls : take pictures and videos*, and *Your accounts : discover known accounts*. RRFS (FR) selects: *Category*, *Price*, *Network communication : view network state*, and *Your personal information : write Browser's history and bookmarks*. As compared to Keyvanpour *et al.* [6], the accuracy results of the RF classifier are similar to ours, and two features coincide with our five most relevant features by RRFS (FR): *SEND_SMS* and *android.telephony.SmsManager*.

Table 1: Experimental results Acc (%) | Rec (%) with baseline and after applying RRFS approach with MM and FR for each classifier and dataset.

Classifier	Dataset	Baseline (%)	RRFS (MM)	RRFS (FR)
RF	Drebin	98.60 | 96.97	95.03 | 90.20	96.85 | 94.95
RF	CICAndMal2017	81.22 | 86.50	75.28 | 83.33	81.54 | 86.20
SVM	Drebin	98.63 | 97.57	95.28 | 90.91	96.36 | 94.77
SVM	CICAndMal2017	71.52 | 82.51	67.27 | 93.61	70.42 | 84.37

Table 2: Number of original features (d) and after RRFS (m).

Dataset	d, Original	m, RRFS (MM)	m, RRFS (FR)
Drebin	215	100	94
CICAndMal2017	183	58	64

5 Conclusions

Malware in Android apps affects millions of users worldwide and is evolving, making its detection a relevant problem. ML approaches have been proposed to detect malware in mobile applications. This paper followed an ML approach with FS techniques. Experiments were performed with the RRFS algorithm with two relevance metrics. The supervised measure, FR, provided better results than the MM unsupervised technique. The use of RRFS proved to be beneficial in this problem. Namely, it substantially reduced the number of features in both datasets while keeping similar accuracy and recall metrics. Furthermore, through RRFS, the names of the most relevant features for malware identification on each dataset were obtained. In future work, we intend to fine-tune the parameters of the RRFS algorithm and explore other FS techniques.

References

[1] Biggest app stores in the world 2022 | statista. `https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/`. (Accessed on 13/02/2023).

[2] Android Malware 2017 | Datasets | Research | Canadian Institute for Cybersecurity | UNB. `https://www.unb.ca/cic/datasets/andmal2017.html`. (Accessed on 27/02/2023).

[3] Hasan Alkahtani and Theyazn HH Aldhyani. Artificial intelligence algorithms for malware detection in Android-operated mobile devices. *Sensors*, 22(6):2268, 2022.

[4] Artur Ferreira and Mário Figueiredo. Efficient feature selection filters for high-dimensional data. *Pattern Recognition Letters*, 33(13):1794 – 1804, 2012. ISSN 0167-8655. doi: http://dx.doi.org/10.1016/j.patrec.2012.05.019.

[5] Rejwana Islam, Moinul Islam Sayed, Sajal Saha, Mohammad Jamal Hossain, and Md Abdul Masud. Android malware classification using optimum feature selection and ensemble machine learning. *Internet of Things and Cyber-Physical Systems*, 3:100–111, 2023. ISSN 2667-3452. doi: https://doi.org/10.1016/j.iotcps.2023.03.001.

[6] Mohammad Reza Keyvanpour, Mehrnoush Barani Shirzad, and Farideh Heydarian. Android malware detection applying feature selection techniques and machine learning. *Multimedia Tools and Applications*, 82(6):9517–9531, 2023. doi: https://doi.org/10.1007/s11042-022-13767-2.

[7] Vasileios Kouliaridis and Georgios Kambourakis. A comprehensive survey on machine learning techniques for Android malware detection. *Information*, 12(5):185, 2021.

[8] Ali Muzaffar, Hani Ragab Hassen, Michael A Lones, and Hind Zantout. An in-depth review of machine learning based Android malware detection. *Computers & Security*, page 102833, 2022.

[9] Qing Wu, Xueling Zhu, and Bo Liu. A survey of Android malware static detection technology based on machine learning. *Mobile Information Systems*, 2021.