Permission-Based Classification of Android Malware Applications Using Random Forest

Nikolaos Chrysikos[1], Panagiotis Karampelas[1] and Konstantinos Xylogiannopoulos[2]
[1] Hellenic Air Force Academy, Dekeleia, Greece
[2] University of Calgary, Calgary, Canada
nikoschrysikos2000@gmail.com
panagiotis.karampelas@hafa.haf.gr
kostasfx@yahoo.gr

Abstract: Android is arguably the most widely used mobile operating system in the world. Due to its widespread use, it has attracted a lot of attention of cybercriminals who attempt to exploit its architecture and outsmart innocent users to install malware applications. The number of such applications is growing every day either by alternating a basic exploitation mechanism or by creating novel mechanisms to exfiltrate users' data. As a result, there is an increasing need for detection mechanisms that can classify these applications to families based on their characteristics. A significant amount of research has already been devoted to analysing and mitigating this growing problem; however, this situation demands more efficient methods with higher precision. The paper proposes such a framework for analysing and classifying a malicious application to certain families relying on the permissions used. The proposed method involves the pre-processing of the applications to extract their permissions, the tokenization of permissions, the data cleansing and finally the application of the Random Forest Classifier to classify the applications in families. The proposed method is trained and tested with a dataset of 11,159 malicious applications categorized in 33 unique families. The precision, recall and f1-score achieved is 98%. The results of the proposed methodology are promising, since it even works in an unbalanced dataset and in many cases outperform other state-of-the-art approaches.

Keywords: Malware Classification, Android, Permissions, Machine Learning, Random Forest

1. Introduction

Presently, the prevalence of smart devices is universal constituting the most popular mean for Internet access worldwide. According to Statista (Petroc, 2023), 6.259 billion people owned a smart phone in 2021 which is translated to the 80% of the world population. Thus, people of diverse age groups with different educational backgrounds and origin have integrated smart phones in their everyday life for audio and video calling, instant messaging, carrying out financial transactions, etc. With this plethora of uses, smart phones become our personal data depositories in which all aspects of our lives are recorded. This fact has led cybercriminals to constantly target smart phones by attempting to infect them with malware in order to get hold of the personal data of the users for their personal gain. The 2021 Mobile Threat Report by Kaspersky Labs (Kaspersky, 2023) reports that more than 46 million attacks were carried out worldwide in 2021 by 3.5 million malicious mobile installation packages for Android. This operating system is installed in approximately the 70% of all the mobile devices worldwide in 2022 (Petroc, 2023) and thus, the majority of the malicious applications target Android devices.

Conversely, many security specialists identifying the activities of cybercriminals have proposed diverse defensive techniques in order to protect the mobile users and their personal data. One of the prominent techniques that many security researchers consider as more effective for protection is the automatic malware detection and classification since by recognizing the type of a malicious application, the security system can automatically take the necessary counter-measures. Usually, there are two approaches for malware classification either using dynamic or static analysis. In dynamic analysis, the behaviour of the application is monitored and specific characteristics are used as features in order to classify the application as malicious or benign while in static analysis, characteristics such as opcodes, permissions, receivers, etc. are extracted and checked. Both approaches have been proved adequately efficient to classify malicious applications but still they lack in accuracy (Ahmad et al, 2017). Moreover, various obfuscation techniques recently used by the attackers in the mobile application's code including encryption, compression, polymorphism and metamorphism have made their detection harder and thus novel detection techniques that overcome the specific countermeasures are needed.

Motivated by this gap, our approach is focused on the static analysis of permissions which is one of the most important characteristics of an Android application. The permissions are used to provide access to various resources and services that are available in the mobile phone (Amer, 2021). Usually, they are classified as normal, signature, special and dangerous. In the first category, permissions such as ACCESS_WIFI_STATE or INTERNET can be found which frequently are necessary for the operation of an application. In the second

category, permissions such as BATTERY_STATS or BIND_WALLPAPER are found which need to be signed with the same signature as the application that they ask to get access. The appearance of these permissions depends on the need for bindings with other applications or services. In the special category, permissions that require mobile user authorization are listed such as WRITE_SETTINGS. For these permissions, the users would have to provide their consent in order to be activated. As dangerous permissions are considered all those which enable the application to use potential private information or services in the mobile phone such as CAMERA, ACCESS_BACKGROUND_LOCATION, etc. Google lists 40 such permissions in the developers website[1] that they should be avoided unless it is necessary. In addition, the developers can define their own custom permissions to some users or other external applications. As it can be understood, the permissions' model is very complex and important for the security of the mobile application and that is why many researchers have focused their attempts for classifying malware applications using this characteristic (Adebayo and Aziz, 2015). In this line of research, our proposed methodology uses permissions to perform Android malware classification. Our methodology:

- uses only permissions as input data in order to analyse and classify the malware applications achieving at the same time very high accuracy. This is a competitive advantage since the vast majority of real world malicious Android applications have most of their data obfuscated, and thus an approach that would require more features could perform poorly or not at all if some of the data needed, are obfuscated or missing;
- can perform equally accurately in classifying applications independently of the homogeneity of the sample. In the experiment we run, we managed to classify accurately applications in families of more than 3,000 applications but also in families with less than 5 applications.

The paper is organized as follows: in the related work section, the contemporary research in malware classification and more specifically in permission-based malware classification is presented. In Section 3, the proposed methodology is explained and the corresponding analysis phases are described. Section 4 presents the experimental environment, the dataset and the computer specifications used for the experiment. The next section presents and discusses the results acquired presenting the key findings and the limitations of the proposed method. In the final section, the conclusions and tentative future work in this area are presented.

2. Related Work

Malware detection and classification as mentioned is one of the issues in mobile devices security that has attracted researchers attention the latest years. There are several research teams that have proposed different approaches in addressing the issue with diverse results. In our review of the related work, we approach the subject more broadly examining first some generic malware classification methods and then we delve into the those which use permissions.

2.1 Malware classification

There are diverse features of the mobile applications that can be used in order to classify malware applications. Bytecode, opcode, system API calls, permissions, etc. are typically used as classifiers in malware classification. In this section, some of the recent techniques utilizing diverse features are presented and discussed in comparison to the proposed methodology.

One approach that is frequently used in malware classification is that of detecting similarities in applications' bytecode. One such approach creates a bytecode based model, performing a frequency analysis in order to classify malicious applications to malware families (Kang et al, 2013). Eventually, machine learning algorithms such as Decision Trees, Association Rules Mining, Genetic Algorithms, etc. were used to categorize malicious applications scoring 94% of F-measure. If the Android applications are obfuscated, the specific methodology will provide poor results since each variant will have different bytecode due to the obfuscation. Another approach using system calls with call logs to classify mobile applications with Naïve Bayes classifier achieved a 99.86% accuracy applied in a sample of 5,560 applications (Ahmad et al, 2017). While the specific technique achieves very high accuracy, there are cases in which malware applications do not make phone calls and thus the specific approach may not be proved so efficient. In another approach, a sample of 13,850 mobile applications including

[1] https://developer.android.com/reference/android/Manifest.permission

3,960 malware and 1,108 benign application was used in a machine learning based technique in order to classify them as malware or benign using 36 features extracted by mobile security framework. The proposed technique achieved an accuracy of 93.63% (Sachdeva et al, 2018). However, the limitation of this approach is that the applications are not classified to their appropriate family but are detected as malware or not which in some cases is not adequate. A classification method using system calls to GPS data is used in another study on a sample of approximately 5,560 malware applications and 500 random applications in which the researchers found 32 related patterns that were used from the malware applications (Saudi, 2017). In a recent study, Arif et al (2021) used a sample of 10,000 Android applications in order to classify malware applications using static analysis. The feature used for the machine learning model was permissions and the Random Forest classification algorithm achieved accuracy 91.59% which is lower than the proposed methodology achieved using the same classification algorithm.

2.2 Permission-based classification

A first attempt to use permissions for malware description using diverse machine learning algorithms is described by Huang et al (2013). In the specific study, the researchers used 124,769 benign applications and 480 malicious applications. The achieved accuracy of the algorithms as reported is above 80%. Another similar system, called Permission-based Malware Detection System used permissions as behavioural markers and the researchers created a classifier taking into consideration the combination of application permissions to classify malicious applications. They tested their method in a relatively small dataset of 2,950 benign and malicious applications with an average precision of 92-94% (Rovelli & Vigfússon, 2014). Milosevic et al (2017) attempted to analyse and identify malicious applications for the Android operating system using a machine learning aided approach for static analysis that uses permissions. They have used a wide range of ensemble machine learning algorithms and the highest precision, recall and f1-score that they achieved was 89%. An ensemble machine learning approach for android malware classification using hybrid features was presented by Pektaş & Acarman (2018) utilizing static features such as permissions, hidden payloads and dynamic features such as API calls, network connections, etc. The authors applied a number of machine learning algorithms on a 3,339 malicious applications sample of 20 malware families achieving 92% accuracy. The whole analysis and detection process required 8 days. The SIGPID malware detection system is based on permission usage analysis that employees 22 significant permissions for malware classification (Li et al, 2018). The system is based on the Support Vector Machine classifier which achieves an accuracy 93.62% using these 22 permissions. The accuracy of the proposed methodology in this paper using a different classifier and all the relevant permissions achieves better accuracy.

3. Methodology

The proposed methodology implements a composite classification model that analyses Android applications, extracts data and eventually classifies malicious applications to pre-defined malware families. The malware detection approach is based on extracting and analysing application features from the AndroidManifest.xml file. We have divided our methodology into three phases. The initial phase starts with the reverse engineering of the applications and data extraction from their AndroidManifest.xml file. In the second phase, the acquired data are processed and transformed to the appropriate form in order to be fed to the machine learning algorithm for parsing in the third phase. Figure 1 illustrates the overall process of the proposed methodology.

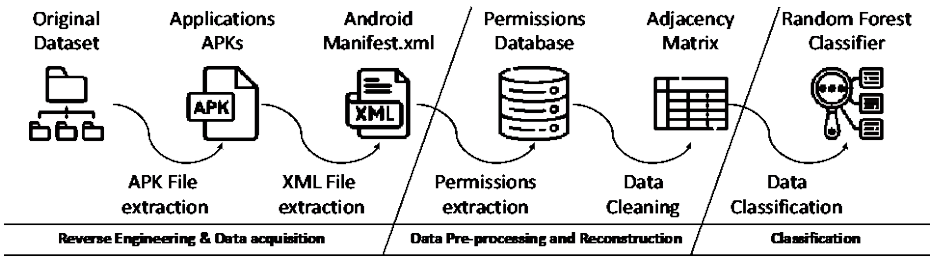


Figure 1: The overall process

3.1 Reverse Engineering and Data acquisition

The classification requires the training of the algorithm with a labelled dataset. For that purpose, we acquired a dataset of malicious Android applications (Wei et al, 2017) in order to perform the required training of the

adopted machine learning algorithm. Based on the description of the dataset, the applications are labelled and correctly classified to the corresponding malware families by the dataset creators. The training and test data ratio that was chosen was 75:25. From each family ,we removed the 25% of the applications in order to form the test dataset while the remaining apps were used to prepare the training dataset. However, in order to use the chosen features, further pre-processing is required.

The malicious applications were provided in the form of Android application PacKages (APKs) organized in families folders and thus, the AndroidManifest.xml file which contains the permissions was not immediately available. Therefore, reverse engineering was required to extract all the contents of each application package in order to acquire the necessary file. This was achieved by using Apktool[2] to extract the AndroidManifest.xml file. The second step required the collection of all the permissions of each malicious application from the manifest file. In this step, we used the list of the 206 Android permissions[3] but we also added any custom permission that was defined in the collected manifest files. In total, 358 unique permissions were collected.

3.2 Data Pre-processing and Reconstruction

In the second phase, we started the pre-processing of the data which includes the following activities:

- Duplicate Permissions Removal. We found that some malicious applications contained the same permission twice. We considered these occurrences as abnormal which will affect negatively the training of the machine learning algorithm and thus we removed them. The collected permissions were also sorted alphabetically for easier visual examination and comparison if required.
- Preparation of the Adjacency Matrix of Application and Permissions. The columns of the matrix represent the 358 permissions and each row of the matrix corresponds to each application. If a permission existed in the AndroidManifest.xml, this was denoted by 1 in the corresponding column of the permission or by 0 if it was not present. Instead of using the name of each individual application, we used as label the name of its malicious family. In Table 1 part of the adjacency matrix is demonstrated.
- Removal of unused permissions. Further data cleansing was required since there were permissions that were not used by any application and therefore they did not add any value to the next stage. Similarly, applications that do not request any permissions and consequently we would not be able to classify them were removed. More precisely, there was one such an application and 224 permissions that could influence negatively the next phase, so they were removed. Eventually, the final shape of the adjacency matrix was 11,160 by 134.
- Removal of insignificant features. In further preparation of the dataset for classification, we computed the hamming distance (Hamming, 1950) of features. If any pair of features has zero hamming distance, one of them was removed, since both columns conveyed exactly the same information. By this process 15 pairs were detected and in total 10 features were removed.

Table 1: Excerpt from the Adjacency Matrix of the features

Applications	GLOBAL_SEARCH	DEVICE_POWER	GET_PACKAGE_SIZE	MANAGE_USB
AndroRAT	0	0	0	0
AndroRAT	0	0	0	1
AndroRAT	1	0	0	0
AndroRAT	0	0	1	0

3.3 Classification

Random Forest Classifier (Breiman, 2001) has been used as the classification model of the methodology. The specific classifier is based on an “ensemble” of independent decision trees. Each tree is presented as a flow chart in which there are branches which represent the features and nodes that represent the different instances that

[2] https://ibotpeaches.github.io/Apktool/
[3] https://developer.android.com/reference/android/Manifest.permission

occur according to the result of the test (Panigrahi et al, 2018; Biau et al, 2008). In that sense, random forests (RF) combine these decision tree classifiers and based on the overall result of the decision trees it arrives in a final result. Random Forest was selected because it outperforms other similar machine learning classification techniques (Multi-layer Perceptron classifier, Balanced Bagging Classifier, Balanced Random Forest Classifier, Support Vector Machine, K-nearest neighbors classifier) (Table 2) since it works well with high volume data with many classes and low correlation, as these in the available dataset, and produce high precision in classification tasks (Liu et al, 2012). The RF Classifier that was used is implemented in Python as part of the Scikit-Learn[4]. To apply the specific classifier, the original dataset was split into two as mentioned in a ratio of 75:25.

Table 2: f1 – score comparison of Random Forest against other known classifiers

	KNN	SVM	RF	BRFC	BBC	MLP
accuracy	0.96	0.95	0.98	0.74	0.64	0.95
macro avg	0.84	0.69	0.97	0.57	0.50	0.70
weighted avg	0.95	0.94	0.98	0.77	0.64	0.95

4. Experiments

The dataset used originally consisted of 24.650 malware samples for Android operating system categorized in 71 families (Wei et al, 2017) which were collected between 2010 and 2016. From the 71 families, we used 33 families for training and testing since some of the rest contained corrupted data and it was not possible to use them. In addition to that, it should be noted that not all samples of the 33 families contained an AndroidManifest.xml file which means that the actual number of samples that we used were less than the number of applications that a family actually contained in the original dataset. Table 3 provides an overview of the malware families we used in our dataset. For each family, its corresponding type is displayed indicating its main purpose. In total, these 33 families contained 14,437 samples from which only the 10,960 malicious applications were used, which is the 75.9% of all the applications in the 33 families.

The computing environment used for the analysis consisted of a laptop with an AMD Ryzen™ 7 3750H mobile processor, a Radeon Vega Mobile Gfx graphics processor and 16Gbs of memory. The overall experiment lasted about 2 hours starting with the initial extraction of the permissions from the AndroidManifest.xml file of each application, pre-processing, classifying and repeating the process for all samples. On average, during the experiment the resources of the laptop used were approximately 6Gbs of memory and 70% of the computing power of the processor.

Table 3: The dataset statistics

Families	Type	Samples #	Used Samples #	Percentage
AndroRat	Backdoor	46	46	100
Andup	Adware	45	45	100
Aples	Ransom	21	21	100
BankBot	Trojan-Banker	740	647	87.3
Bankun	Trojan-Banker	72	72	100
Boqx	Trojan-Dropper	221	221	100
Boxer	Trojan	44	44	100
Cova	Trojan-SMS	18	18	100

[4] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

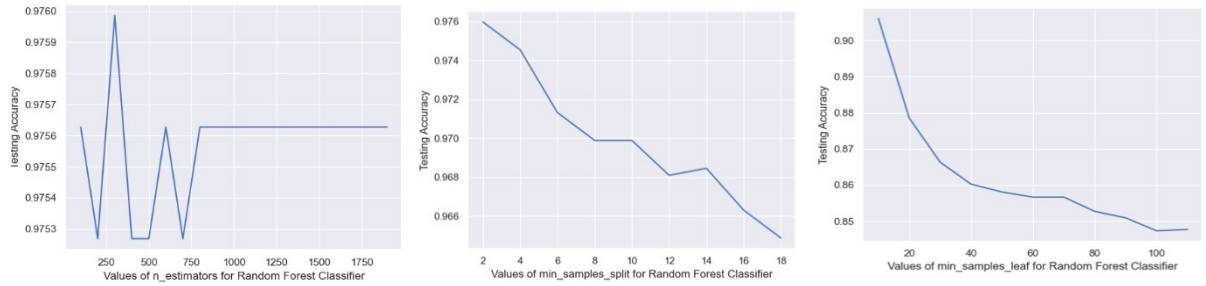
Families	Type	Samples #	Used Samples #	Percentage
Dowgin	Adware	3,410	3,410	100
DroidKungFu	Backdoor	549	549	100
Erop	Trojan-SMS	46	46	100
FakeAngry	Backdoor	10	10	100
FakeAV	Trojan	5	5	100
FakeDoc	Trojan	21	21	100
FakeInst	Trojan-SMS	2,172	1,997	91.9
FakePlayer	Trojan-SMS	21	5	23.8
Finspy	Trojan-SMS	9	9	100
Fjcon	Backdoor	20	20	100
Fobus	Trojan	4	4	100
Fusob	Ransom	1,277	1,270	99.4
GingerMaster	Backdoor	128	128	100
GoldDream	Backdoor	53	53	100
Koler	Ransom	69	4	5.8
Kuguo	Adware	1,199	10	0.8
Lotoor	HackerTool	333	15	4.5
Mecor	Trojan-Spy	1,820	1,820	100
MobileTX	Riskware	7	7	100
Penetho	Riskware	7	7	100
RuMMS	Trojan-SMS	402	350	87.1
SimpleLocker	Ransom	173	11	6.4
SlemBunk	Trojan-Banker	174	16	10.3
Youmi	Adware	1,301	76	5.8
Ztorg	Trojan-Dropper	20	3	15
Total/Average	-	14,437	10,960	76.9

As mentioned in above, the dataset consisted of 33 families and 10,960 applications that were not evenly distributed. With mean application number per family equal to 341 and median equal to 44, we can certainly conclude that this distribution is positively skewed and eventually unbalanced. Even though, we could use undersampling or oversampling to balance the dataset, it was decided to use the unbalanced data so that the results will be reproducible and reflect a real situation in which some variations of the malicious applications are more frequent than others for various reasons, e.g., the application code was publicly made available, etc. Oversampling the minority class can lead to overfitting and can also result in a disproportionate weight on the minority class, which may lead to biased predictions. Undersampling the majority class, on the other hand, can result in loss of valuable information and may lead to under-representation of important features in the model. Moreover, Random Forest is known for its ability to handle unbalanced datasets and does not require balancing of classes. Instead, it relies on the construction of decision trees on random subsets of the data, and by combining the predictions of multiple trees, it can effectively handle imbalanced data. Thus, since the dataset was unbalanced, parameters of the classification algorithm should be set accordingly. After thorough analysis for the best parameter combination with GridSearchCV method (Ranjan et al, 2019), we concluded that the best parameters for Random Forest Classifier model are those which are presented in Table 4 and are illustrated in bold characters.

Table 4: RF Initial Parameters

parameters	Option_1	Option_2	Option_3
bootstrap	**True**	False	-
oob_score	**True**	False	-
n_jobs	**-1**	None	-
warm_start	**True**	False	-
criterion	entropy	**gini**	-
max_features	auto	**sqrt**	-
min_samples_leaf	**1**	2	-
min_samples_split	**2**	10	-
n_estimators	50	**300**	1000

We can observe how accuracy fluctuates while n_estimators, min_samples_split, min_samples_leaf values' are changing in Figure 1 (a), (b) and (c) respectively.



(a) (b) (c)

Figure 1: Testing accuracy for values of n_estimators (a), min_samples_split (b) and min_samples_leaf (c)

We notice that the peak value of accuracy is detected for n_estimators = 300 and min_samples_split and min_samples_lead are negatively correlated with accuracy.

5. Results

The outcomes of performance metrics are shown in Table 5. The evaluation of the method is performed by considering detection parameters or in terms of accurately distinguishing among malicious families by considering accuracy, f1-score, recall and precision that are displayed in the Figure 2, and True Positive, True Negative, False Positive, False Negative ratio that are described in the confusion matrix generated by the predictions of RF model (Figure 2). Precision is defined as the probability of the proposed system correctly identifying the malicious application and is calculated based on True Label by predicted Labels. Recall refers to the number of total malicious apps of each family, which are labelled correctly to the family that they belong and can be determined by considering True Labelled by Total Labelled. Furthermore, accuracy of machine learning classifier has also been determined to demonstrate the effectiveness of Random Forest. In the experiment, the precision that was achieved was 98% with 97.52% recall and 97.56% f1-score. From the confusion matrix, we can deduce that 96.4% of applications were correctly classified (true labelled) to the family that they belong while only a 3.6% were misclassified (false labelled) according to the results.

Table 5: Evaluation metrics

Families	Precision	Recall	F1-score	Support
AndroRAT	1	1	1	12
Andup	1	0.909091	0.952381	11
Aples	1	1	1	5
BankBot	1	0.7625	0.865248	160
Bankun	1	0.882353	0.9375	17
Boqx	0.796875	0.944444	0.864407	54
Boxer	1	1	1	11
Cova	1	1	1	4
Dowgin	0.982415	0.982415	0.982415	853
DroidKungFu	1	0.992647	0.99631	136
Erop	1	1	1	11
FakeAV	1	1	1	1
FakeAngry	1	1	1	3
FakeDoc	1	1	1	5
FakeInst	1	1	1	541
FakePlayer	1	1	1	5
Finspy	1	1	1	2
Fjcon	1	1	1	5
Fobus	1	1	1	1
Fusob	1	1	1	317
GingerMaster	0.964286	0.870968	0.915254	31
GoldDream	1	1	1	12
Koler	1	1	1	1
Kuguo	1	0.666667	0.8	3
Lotoor	1	1	1	6
Mecor	1	1	1	454
MobileTX	1	1	1	2
Penetho	1	1	1	1
RuMMS	0.696	1	1	87
SimpleLocker	1	1	1	3
SlemBunk	1	1	1	4
Youmi	0.97	0.9	0.93	31
Ztorg	1	1	1	1
accuracy	-	-	0.975269	2,790
macro avg	0.980928	0.975254	0.970735	2,790
weighted avg	0.98005	0.975269	0.975666	2,790
Accuracy : 97.75%				

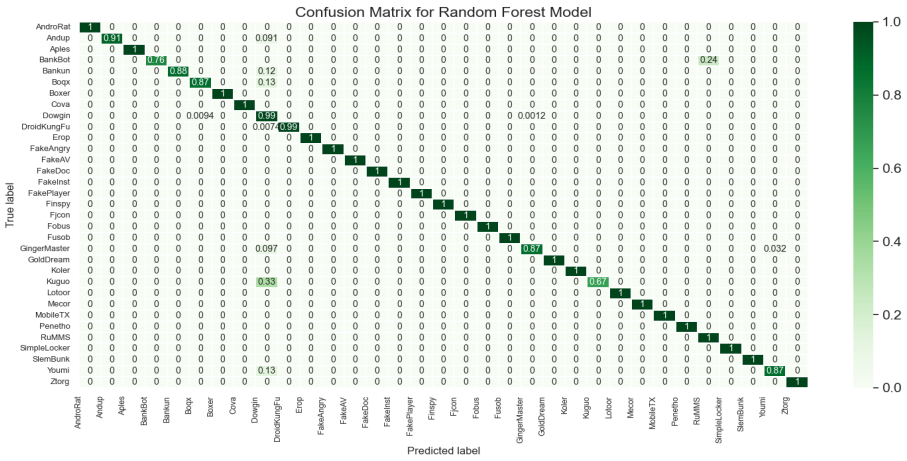
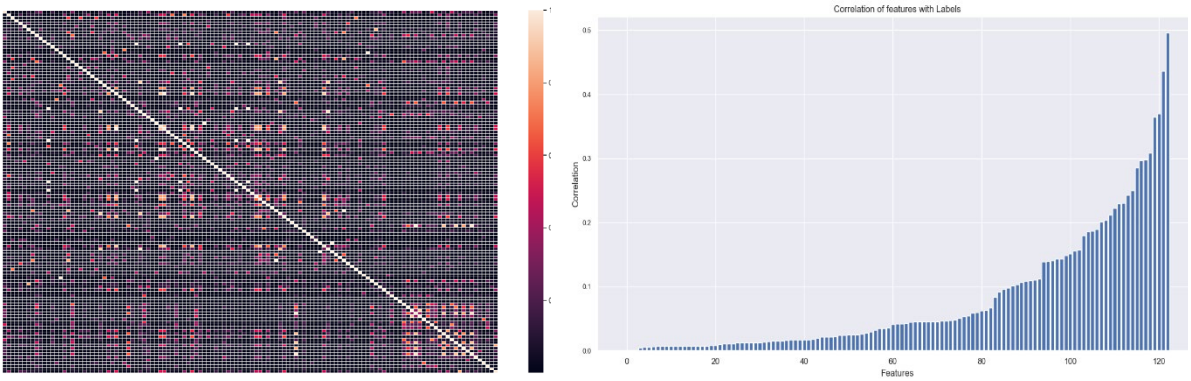


Figure 2: Confusion Matrix for RF Model

Thus, the proposed methodology proves to be highly efficient even though the dataset used is highly unbalanced. RF is a machine-learning method that generally works well with high-dimensional problems and allows for nonlinear relationships between features. In Figure 3 (a), where the heatmap depicts the intercorrelation of the labels and features, we can discern by the light colour that there are only a few instances in which features are highly correlated among them. Similarly in Figure 3 (b) we can see the correlation of each future with the applications' family labels.



(a) (b)

Figure 3: Intercorrelation heatmap (a) and correlation (b) of labels and features

Comparing our results with other published methodologies, we can see that our method outperforms several other methods in terms of higher precision, recall and f1-score. As for example, in Milosevic et al (2017), as it was mentioned, the researchers use permissions in a wide range of ensemble machine learning algorithms and the highest precision, recall and f1-score achieved was 89%. Pektaş and Acarman (2018) used a hybrid system for malware classification based on various features including permissions and they achieved 92% accuracy after 8 days of processing. Kang et al (2013) using the bytecode of the applications for classification with Decision Trees, they managed to score 94% of F-measure. Compared to the above-mentioned methods, our proposed model not only performs better and faster, but also uses a highly unbalanced dataset of 10,960 malicious applications, categorized in 33 families and requires only the permissions in order to classify these applications.

The proposed methodology even though it correctly classifies the malware applications in the appropriate family with very high precision, it has some limitations. If, for example, a new malware application of a known family will enter the test dataset then the method will work efficiently and the malware application will be classified to the correct family with high precision. However, a malware application which belongs to a new family or a family that was excluded from the training dataset will be classified to an existing family especially if the permissions are correlated with those of an existing one. Another limitation is that it is unable to classify a malware application if there are no permissions in the AndroidManifest.xml. We noticed in the given dataset, there are cases in which in the same family there was a small percentage of malware applications that they did not have any permission. In Table 1, it can be seen that FakeInst family has 5 varieties with 2,172 malware applications in

total, but from those only the 1,997 of them had AndroidManifest.xml file and could be used in the dataset. The remaining 197 malware applications could not be classified at all using the specific methodology. Finally, since Random Forests Classifier has been used to classify the malware applications which is a probabilistic classifier, statistical error is introduced. However, the error depends on the correlation between two trees in the forest which in our case is very low as depicted in Figure 2 and the strength of each tree in the forest (Breiman, 2001) which has been optimized based on the initial parameters selected.

6. Conclusions

Classification of malicious applications to malware families can be a difficult task due to their great diversity. There is an undeniable need for accurate and efficient categorization of malicious applications in order to determine the abilities of these applications and the possible ways that they may affect the victim. Therefore, we developed an effective mechanism that extracts information from the applications' permissions and by utilizing a well-established machine learning algorithm, Random Forest, it was possible to predict accurately in which family the sample application belongs to. The model was applied on a dataset of 10,960 applications distributed in 33 families and with hyperparameter tuning it reached an accuracy of 97.75%. Due to the simplicity and customizability of the method, it can be generalized to operate on other types of input data (receivers, intents, etc.) or even to tackle other types of classification problems such us to distinguish benign from malicious applications. In addition, the specific research work can be further expanded to other malicious data sources in order to further develop the classification model and achieve even better results. Furthermore, the technique can be enhanced by incorporating applications' features from dynamic analysis which could address the limitation of missing permissions in some malicious applications. These improvements we believe that can further enhance detection and classification of malicious Android applications and boost further research in this field.

References

Adebayo, O.S. and Abdul Aziz, N., (2015) Static Code Analysis of permission-based features for android malware classification using apriori algorithm with Particle Swarm Optimization.

Ahmad, I.N., Ridzuan, F., Saudi, M.M., Pitchay, S.A., Basir, N. and Nabila, N.F., (2017) Android mobile malware classification using a tokenization approach. In *The World Congress on Engineering and Computer Science* (pp. 271-285). Springer, Singapore.

Amer, E., (2021) May. Permission-based approach for android malware analysis through ensemble-based voting model. In *2021 International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC)* (pp. 135-139). IEEE.

Mohamad Arif, J., Ab Razak, M.F., Awang, S., Tuan Mat, S.R., Ismail, N.S.N. and Firdaus, A., (2021) A static analysis approach for Android permission-based malware detection systems. *PloS one*, *16*(9), p.e0257968.

Biau, G., Devroye, L. and Lugosi, G., (2008) Consistency of random forests and other averaging classifiers. *Journal of Machine Learning Research*, *9*(9).

Breiman, L., (2001) Random forests. *Machine learning*, *45*(1), pp.5-32.

Hamming, R.W., (1950) Error detecting and error correcting codes. *The Bell system technical journal*, *29*(2), pp.147-160.

Huang, C.Y., Tsai, Y.T. and Hsu, C.H., (2013) Performance evaluation on permission-based detection for android malware. In *Advances in intelligent systems and applications-volume 2* (pp. 111-120). Springer, Berlin, Heidelberg.

Kaspersky (2022) "2021 Mobile threats report", [online], Kaspersky.com, https://www.kaspersky.com/about/ press-releases/2022_2021-mobile-threats-report-cybercriminals-forego-low-hanging-fruit-to-go-after-banking-and-gaming.

Li, J., Sun, L., Yan, Q., Li, Z., Srisa-An, W. and Ye, H., (2018) Significant permission identification for machine-learning-based android malware detection. *IEEE Transactions on Industrial Informatics*, *14*(7), pp.3216-3225.

Liu, Y., Wang, Y. and Zhang, J., (2012) New machine learning algorithm: Random forest. In *International Conference on Information Computing and Applications* (pp. 246-252). Springer, Berlin, Heidelberg.

Milosevic, N., Dehghantanha, A. and Choo, K.K.R., (2017) Machine learning aided Android malware classification. *Computers & Electrical Engineering*, *61*, pp.266-274.

Ranjan, G.S.K., Verma, A.K. and Radhika, S., (2019) K-nearest neighbors and grid search cv based real time fault monitoring system for industries. In *2019 IEEE 5th international conference for convergence in technology (I2CT)* (pp. 1-5). IEEE.

Panigrahi, R., Borah, S., Day, N., Babo, R. and Ashour, A.S., (2018) Classification and analysis of facebook metrics dataset using supervised classifiers. *Social Network Analytics: Computational Research Methods and Techniques*, *1*.

Pektaş, A. and Acarman, T., (2018) Ensemble machine learning approach for android malware classification using hybrid features. In *International conference on computer recognition systems* (pp. 191-200). Springer, Cham.

Petroc, T. (2023) "Smartphone mobile network subscriptions worldwide 2016-2028", [online], Statista.com, https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/.

Petroc, T. (2023) "Market share of mobile operating systems worldwide 2009-2022", [online], Statista.com, https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/

Ranjan, G.S.K., Verma, A.K. and Radhika, S., (2019) K-nearest neighbors and grid search cv based real time fault monitoring system for industries. In *2019 IEEE 5th international conference for convergence in technology (I2CT)* (pp. 1-5). IEEE.

Rovelli, P. and Vigfússon, Ý., (2014) PMDS: permission-based malware detection system. In *International conference on information systems security* (pp. 338-357). Springer, Cham.

Sachdeva, S., Jolivot, R. and Choensawat, W., (2018) Android malware classification based on mobile security framework. *IAENG International Journal of Computer Science*, *45*(4), pp.514-522.

Saudi, M.M., 2017. Mobile malware classification via system calls and permission for GPS exploitation. *International journal of advanced computer science and applications*, *8*(6).

Wei, F., Li, Y., Roy, S., Ou, X. and Zhou, W., (2017) Deep ground truth analysis of current android malware. In *International conference on detection of intrusions and malware, and vulnerability assessment* (pp. 252-276). Springer, Cham.