Mobile Family Detection through Audio Signals Classification

Rosangela Casolare[1], Giacomo Iadarola[2], Fabio Martinelli[2], Francesco Mercaldo[2,3] and Antonella Santone[3]

[1]*Department of Biosciences and Territory, University of Molise, Pesche (IS), Italy*

[2]*Institute of Informatics and Telematics, National Research Council of Italy, Pisa, Italy*

[3]*Department of Medicine and Health Sciences "Vincenzo Tiberio", University of Molise, Campobasso, Italy*

Keywords: Malware, Audio, Android, Machine Learning, Deep Learning, Security, Classification.

Abstract: Nowadays smartphones, and generically speaking mobile devices, allow users a plethora of tasks in total mobility for instance, from checking the balance on the bank account to distance learning. In this context it is really critical the detection of malicious behaviours, considering the weaknesses of the current antimalware mechanisms. In this paper we propose a method for malicious family detection exploiting audio signal processing: in fact, an application is converted into an audio file and then is processed to generate a feature vector to input several classifiers. We perform a real-world experimental analysis by considering a set of malware targeting the Android platform i.e., 4746 malware belonging to 10 families, showing the effectiveness of the proposed approach for Android malicious family detection.

1 INTRODUCTION

In the last years, there was a huge spread of mobile devices like smartphones and tablets, which became the principal target of attacks, because these devices contain a lot of sensitive, financial and personal information. Among various software systems present in mobile devices, Android is the most popular and also the most diffuse; since Android is an open source system, it arouses more interest from malicious people as it allows you to create customized systems by rebuilding the source code (Enck et al., 2014). Furthermore, this operating system allows to install applications from third-party stores as well as from the official ones; thus, the users, who lack adequate knowledge of the dangers to which they are exposed, are subject to attacks launched by applications downloaded from unofficial stores, which are less reliable. Nevertheless, the presence of malicious applications cannot be excluded even in the official stores (i.e., Google Play Store) (Nguyen et al., 2020).

For this reason, the Android environment results to be the most attacked by cybercriminals (Canfora et al., 2018).

During May 2020, around 430,000 malware attacks were detected on Android devices, counting an increase of 3.6% compared to the previous month. Instead, in August 2020, it was observed a growth of 6.26% compared to the previous month[1].

In this regard, we propose an approach based on the detection of malware families in the Android environment, which consists of the conversion of an Android application into an audio file. Then, we extract from the audio file a series of numerical features that are used to understand which family the application belongs to.

Among the most applied analysis techniques in the literature, we decided to explore a group of them and to adopt four different supervised classification algorithms, belonging to the Machine Learning (Stochastic Gradient Descent, Random Forest) and Deep Learning fields (two different model structures of Multy-layer Perceptron).

The paper's organization is the following: in section 2 is described the proposed method to analyse the audio signals and make the malware family detection, starting from the conversion of an Android application in an audio file, which its features will be used to classify the family belonging; in section 3 is described the considered dataset, showing the effectiveness of the experimental analysis executed on it; in section 4 current state-of-the-art literature is analyzed and dis-

[1]https://news.drweb.com/show/review/?i=13991&lng=en

Casolare, R., Iadarola, G., Martinelli, F., Mercaldo, F. and Santone, A.
Mobile Family Detection through Audio Signals Classification.
DOI: 10.5220/0010543504790486
In *Proceedings of the 18th International Conference on Security and Cryptography (SECRYPT 2021)*, pages 479-486
ISBN: 978-989-758-524-1

cussed and, finally, in section 5 conclusion and future research plans are presented.

2 AUDIO SIGNALS FOR MALWARE FAMILY DETECTION

In this section, we present the method we propose for mobile family detection. In a nutshell, we convert an Android application into an audio file and we extract a series of numerical values (i.e., the features) from the audio file. Then, the features represent the input for a supervised classifier (previously trained) aimed to predict the belonging malware family. In detail the proposed method considers two distinct phases: *Training* (shown in Figure 1) and *Testing* (shown in Figure 2).

The *Training* phase, depicted in Figure 1, is aimed to build a model for the malicious family prediction. We start with a malware dataset (composed by malicious Android application) and the relative family label i.e., the detail about the malicious family for each sample involved. Subsequently, we extract from each Android application (stored in the *apk* file format) the executable file (i.e., the *dex* file), containing the binary of the application (we discard from the analysis all the application resources as, for instance, images and sounds). To convert binary (i.e., the *dex* file) we consider binary bytes forming a digitized raw signal, then we convert the raw signal into *wav*. In detail to generate a *wav* file from *dex* file, we developed a function aimed to firstly generate a *wav* header, subsequently the *dex* file is open and each byte of this file is converted in *wav*. For this task, we resort the *wave* module[2] available in Python; in particular, we invoked the *open* and the *writeframes* methods: the first one to open the file, while the second one for *wav* file writing. By exploiting the *setparams* methods, we also considered the following parameters for the *wav* generation: the number of audio channels equal to 1 (mono i.e., with one input which is distributed equally by the left and right speakers), the sample width set to n bytes with $n = 1$, the frame rate set to 32768, and the number of frames set to 0 and without compression.

Once obtained the audio samples related to each Android sample in the malicious dataset, a set of feature is directly computed on the audio sample.

In detail, the following features are computed:

- *Chromagram*: this feature is related to a chromagram representation automatically gathered from a waveform;
- *Root Mean Square*: this feature (i.e., *RMS*) is related to the value of the mean square of the root that is obtained for each audioframe that is gathered from the sound sample under analysis;
- *Spectral Centroid*: this feature is symptomatic of the “centre of mass” for a sound sample that is obtained as the mean related to the frequencies of the audio;
- *Bandwidth*: it is related to the bandwidth of the spectrum;
- *Spectral Rolloff*: it is expressed as the frequency related to a certain percentage of the total spectral of the energy;
- *Zero Crossing Rate*: it is expressed as the radio belonging to the sign variation relating to the sound samples;
- *Mel-Frequency Cepstral Coefficients*: this feature (i.e., *MFCC*), ranging from 10 to 20 different numerical features, is devoted to represent the shape of a spectral envelope;
- *Zero Crossing Rate*: this value is related to the rate of an audio time series;
- *Poly*: it is computed as the fitting coefficients related to an *n*th-order polynomial;
- *Tonnetz*: it is computed from the the tonal centroid.

Once we obtained the feature vectors from the *.wav* files, we export them to *.csv* files, where each row contains the feature values for each app under analysis with the relative label of the belonging family.

Subsequently, we set the parameters for the classification algorithms (i.e., model setting in Figure 1). We adopt four different supervised classification algorithm. In detail, we experiment the effectiveness of following models:

- *Stochastic Gradient Descent* (SGD): uses stochastic gradient descent that minimizes a chosen loss function with a linear function. The algorithm approximates a true gradient by considering one sample at a time, and simultaneously updates the model based on the gradient of the loss function;
- *Random Forest*: an ensemble classifier obtained from the bagging of decision trees. It consists of hundreds of thousands of decision trees. It falls under those ensemble learning algorithms, that is, algorithms that use multiple machine learning algorithms to get more precise predictions. The number of trees depends on the nature of the training set and other parameters such as the number of classes, the number of beans and the maximum depth;

[2]https://docs.python.org/3/library/wave.html

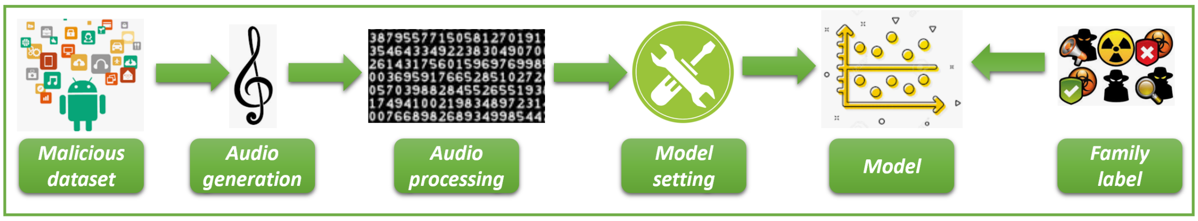


Figure 1: Training.

- *MLP 1*: it is model of computation based on biological neural networks. A neural network is an interconnection of a group of nodes called neurons. An artificial neural network receives external signals on a layer of input nodes (processing units), each of which is connected with numerous internal nodes, organized in several layers. Each node processes the received signals and transmits the result to subsequent nodes. In detail we exploit a multi-layer perceptron (MLP) algorithm with backpropagation, a class of feedforward artificial neural network. We consider two different networks exploiting the MLP architecture: the *MLP 1* model consists of three layers of nodes: an input layer, a hidden layer and an output layer. This model consider 100 neurons for the hidden layer, by exploiting the ReLU activation and the Adam solver;
- *MLP 2*: this model is based, as the *MLP 1*, the MLP algorithm with backpropagation. Differently from the *MLP 1* model, the *MLP 2* one is composed by an input layer, three hidden layers each one considering 100 neurons and the output layer (while the *MLP 1* considers only one hidden layer).

Four different models are considered for conclusion validity i.e., to demonstrate that the proposed feature set, obtained from audio samples, can be effective in the discrimination of different malicious families.

Once built the predictive model, its effectiveness is evaluated in the *Testing* phase, shown in Figure 2.

The idea of the *Testing* phase is the evaluation of the effectiveness of the model built in the *Training* phase. For this reason, considering an application not considered in the model generation (i.e., app under analysis in Figure 2), its *dex* file is obtained from the *apk* one and it is converted into an audio sample. Thus, from the audio sample the features are extracted and then are considered as input for the model that will generated a prediction (i.e., malicious family in Figure 2).

3 STUDY DESIGN AND EXPERIMENTAL ANALYSIS

We design a study composed by two steps: the first one is the descriptive statistics, aimed to provide a graphical impact about the feature value distributions for all the involved families and the second one is the classification results, devoted to confirm the effectiveness of the proposed model for the mobile family detection task.

With regard to the descriptive statistics we exploit boxplots, a method for graphically depicting groups of numerical data through their quartile, to display variation in samples of a statistical population without making any assumptions of the underlying statistical distribution.

The classification analysis is aimed to compute a set of well-known metrics to provide a numerical measurement to evaluating the performances of the proposed models.

3.1 The Real-world Dataset

As stated into the introduction, we consider a real-world dataset composed by 4796 Android malicious applications belonging to 10 different families, as shown in Table 1.

The dataset considered in the experiment was gathered from three different repositories: the first one is the Drebin dataset (Arp et al., 2014; Michael et al., 2013), a very well-known collection of malware largely considered by malware analysis researchers, including the most widespread Android families. The malware dataset is freely available for research purposes [3]. The second malware repositoty is Contagio Mobile [4], a web site containing malicious samples with the relative technical report about the malicious behaviour. The third malicious repository we exploited is the Android Malware repository (AMD) (Iadarola et al., 2020).

[3]https://www.sec.cs.tu-bs.de/~danarp/drebin/
[4]http://contagiominidump.blogspot.com/

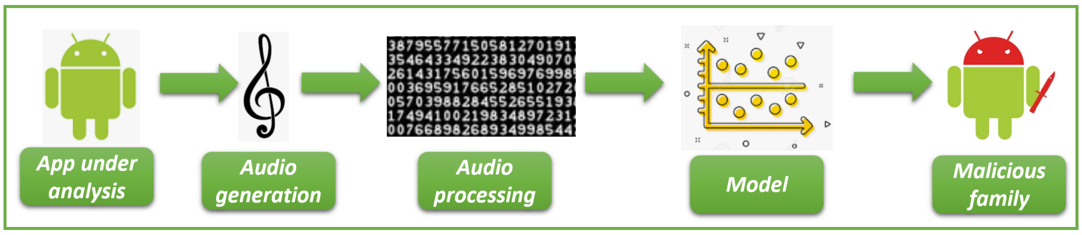


Figure 2: Testing.

Table 1: The real-world malicious dataset involved in the study.

Family	Description	Inst.	#
accutrack	it tracks down the GPS location of the device on which it was installed	*R*	500
airpush	it aggressively pushes advertising content to the device's notification bar	*R*	500
basebridge	it sends SMS and personal information	*R*, *U*	600
droidkungfu	it uses exploits in its attempt to root a device to install other applications	*R*	667
fakeinstaller	it sends SMS messages to premium-rate services	*S*	606
hummingbad	it establishes a persistent rootkit and installs fraudulent applications	*R*	500
judy	an auto-clicking adware relying on the communication with its C&C server	*R*	84
opfake	it hides its presence by installing the Opera browser and can monitor SMS	*S*	610
overlay	a fake bank application using overlay technique to steal user credentials	*U*	56
plankton	it installs a JAR file obtained from an external server	*U*	623

The considered malware dataset consists of 10 Android malicious families characterized by different installation methods (column *Inst.* in Table 1): (i) *standalone* (i.e., *S* in Table 1), applications that intentionally include malicious functionalities; (ii) *repackaging* (i.e., *R* in Table 1), known and common (legitimate) applications that are first disassembled, then the malicious payload is added, and finally are reassembled and distributed as a new version (of the original application); and (iii) *update attack* (i.e., *U* in Table 1), applications that initially do not show harmful behaviors and download an update containing the malicious payload, at runtime.

In detail the *basebridge*, the *droidkungfu*, the *fakeinstaller*, the *opfake* and the *plankton* families were obtained from the Drebin dataset, the *hummingbad*, the *judy* and the *overlay* ones from the Contagio Mobile website. The *accutrack*, *airpush* families were gathered from the AMD dataset.

The malware dataset is also partitioned according to the *malware family* (Zhou and Jiang, 2012).

We analyzed the dataset with the VirusTotal service[5], a web service able to run 61 commercial and free antimalware: this analysis confirmed that the malicious applications were actually recognized as malware.

In Table 1 we indicate also the details about the number of samples considered for each malware family (i.e., column # in Table 1).

[5]https://www.virustotal.com/

For each application of the dataset we gathered the audio sample and the feature set with the procedure explained in the previous section.

3.2 Descriptive Statistics

Figure 3 shows the box-plots related for the spectral chromogram features. For reason space we show only this plot, but similar considerations can be done for the remaining ones.

In Figure 3 each boxplot is related to a single family. On the top of each boxplot we indicate the family name and the median value, while below, for each boxplot, from the left the value of the first quartile, the average and the value of the third quartile.

From the *Spectral Centroid* boxplot in Figure 3, it emerges that the values for this feature for the application to the malicious dataset ranging into different values. For instance, the numerical values for the *accutrack* family are ranging in a smaller range if compared to the *airpush* family. A similar trend is exhibited by the *droidkungfu*, the *plankton* and the *hummingbad* families. For this reason, from this analysis, it seems that the *spectral centroid* can not be of interest for the discrimination of these families. Differently, the *overlay* family assumes values whose first quartile is greater than the third quartile of all other families, making this feature very discriminatory in identifying this family. The *judy* family boxplot in the part between the first quartile and the average dif-

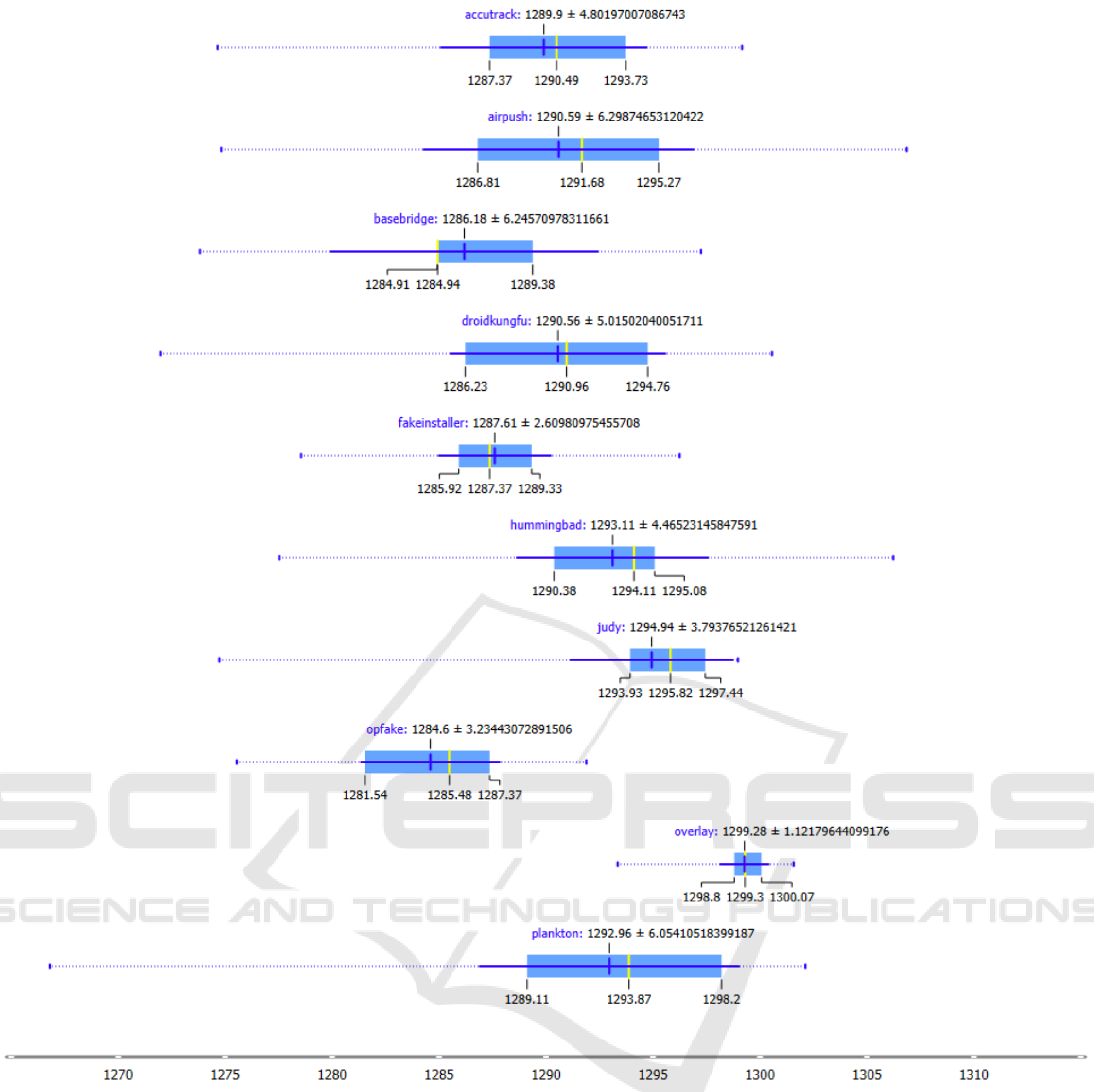


Figure 3: Box-plots for the *Spectral Centroid* feature.

ferent values overlapping with those of other families, but from the average up to the third quartile there is no overlap with any other family, for this reason the feature can be considered discriminating enough to distinguish this family from others. Also the *opfake* family boxplot is of interest in fact, there is only a slight overlap with some values near the third quartile with the first quartile of the remaining families.

Obviously, the more the boxplots of each family are not superimposed, the higher the probability that the models will be able to correctly discriminate the different families. From this visual analysis it emerges that this feature can actually be valid to distinguish some families from others, but as has been said for some families there is overlap. This is the reason why we consider a set of features, in order to increase this possibility.

3.3 Classification Analysis

With regard to the classification analysis, for different metrics are exploited to measure the effectiveness of the proposed method in Android family detection: Precision, Recall, F-Measure and Accuracy.

Table 2 shows the classification results.

As emerges from the results in Table 2 the model obtaining the best performances is *MLP 2* with an average accuracy for family identification equal to 0.988.

Also the *MLP 1* and the *Random Forest* classifica-

Table 2: Classification results.

Model	Precision	Recall	F-Measure	Accuracy
SGD	0.583	0.611	0.586	0.779
MLP 1	0.831	0.833	0.831	0.974
Random Forest	0.905	0.905	0.905	0.986
MLP 2	0.907	0.907	0.907	0.988

tion algorithms obtain interesting performances with an average accuracy equal to 0.974 for the *MLP 1* and equal to 0.986 for the *Random Forest*.

In Figure 4 we show the ROC curve plot relating to the *accutrack* family. The ROC curve is created by plotting the True Positive Rate (TPR, fraction of true positives) versus the False Positive Rate (FPR, fraction of false positives) at various threshold settings. In Figure 4 the green line is related to the Random Forest algorithm, the orange one to the MLP 1 model, the purple to the SGD model and, the pink one to the MLP 2 network.

As shown from the ROC Area in Figure 4 with the exception of the *SDG* model, the remaining ones exhibit equally good performances.

Starting from this results, we focus our analysis on the model obtaining the best results in the classification analysis i.e., the deep learning one (*MLP 2* in Table 2). For understand the performances of the *MLP 2* model at a family grain, in Figure 5 we show the confusion matrix.

All the family are generally correctly detected as belonging to the right malicious family. We highlight 67 (on a total of 667 samples of this family) droidkungfu samples erroneously detected as belonging to the accutrack family and 85 (on 500 samples analysis of this family) accutrack samples predicted as belonging to the droidkungfu one. These two examples represent the main cases of misclassifications. This aspect is also visible from the descriptive analysis, where in the boxplots shown in Figure 3 we highlighted the overlapping between the droidkungfu and the hummingbad malware families.

4 RELATED WORK

The proposed techniques rely on the vivid research branch that studies signal features to detect malware. In this paper, we exploit the audio signal, but more approaches were proposed in the literature, such as texture (Nataraj et al., 2011), network (Kim et al., 2018) or behavioural features analysis (Popli and Girdhar, 2019).

The paper (Farrokhmanesh and Hamzeh, 2019) proposes a similar technique, which extracts the program's bytes and converts them to an audio signal. In detail, the byte of executable files are converted to musical notes (MIDI note) and then audio files are generated. Then, audio features such as MFCC and Chromagram are used to classify music and machine learning classifiers (KNN) is applied.

The main issues of such approaches regard the size of byte sequences to analyze. Most of the time, the static analyses required are time-consuming and computationally expensive. The paper (Bakhshinejad and Hamzeh, 2017) presents an approach that tries to mitigate this problem by applying compression algorithms to the sequences to study. Similarly, the approach proposed by Jerome Q. et al. (Jerome et al., 2014) works directly on the binary sequences, by extracting k-gram and classifying the malware with an SVM.

The method presented in (Vasan et al., 2020) converts malware binaries into colour images, and then use a CNN model, pre-trained on the ImageNet dataset, to distinguish between malware and benign samples. The approach proposed in (Iadarola et al., 2021) applies a similar methodology but improves the robustness of the classification, by analysing also the inference phase, exploiting the use of a Grad-CAM.

The approach proposed in (Azab and Khasawneh, 2020) exploits the use of both audio signal processing and image classification techniques. The initial program's bytes are cast to audio signals, and then applied Fourier Transformation to convert the signal from time-domain to frequency-domain and generate spectrograms. Then, the spectrograms are analysed by a Convolutional Neural Network such as a standard image-classification task.

5 CONCLUSION AND FUTURE WORK

Mobile malware is continuously plaguing users, that are unaware of the malicious behaviour that silently are able to perpetrate harmful action as, for instance, sending sensitive and private information (as, for instance, the samples belonging to the *accutrack* family) but also to install undesired apps (behaviour exhibited by the *droidkungfu* and *hummingbad* fami-

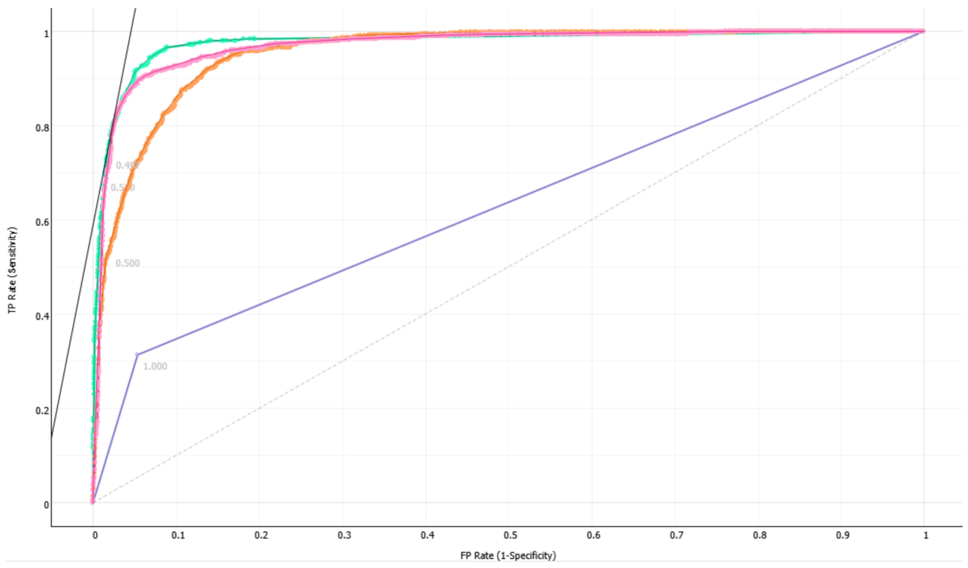
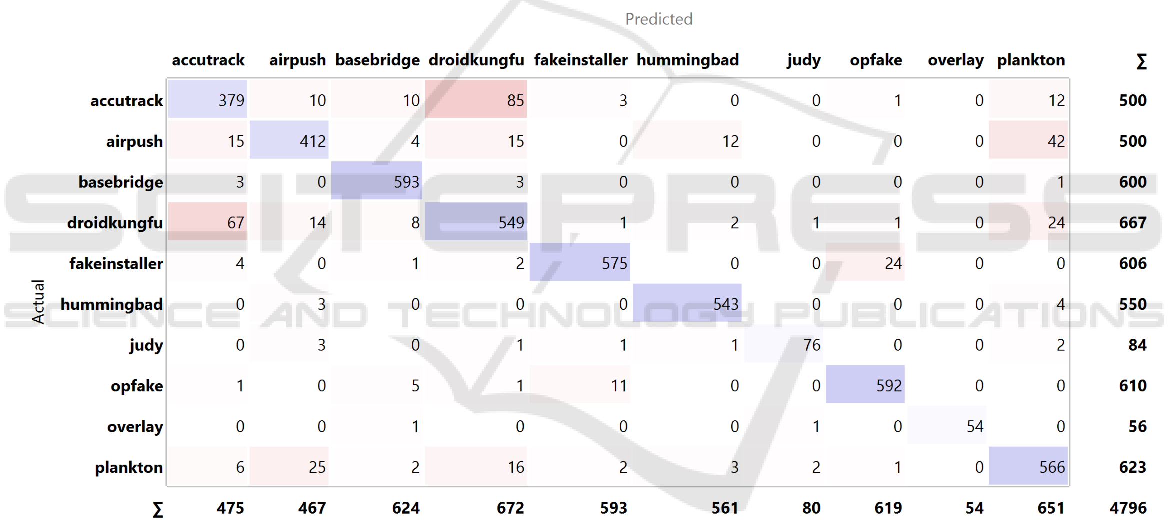


Figure 4: ROC curve for the *accutrack* family.



Actual \ Predicted	accutrack	airpush	basebridge	droidkungfu	fakeinstaller	hummingbad	judy	opfake	overlay	plankton	Σ
accutrack	379	10	10	85	3	0	0	1	0	12	500
airpush	15	412	4	15	0	12	0	0	0	42	500
basebridge	3	0	593	3	0	0	0	0	0	1	600
droidkungfu	67	14	8	549	1	2	1	1	0	24	667
fakeinstaller	4	0	1	2	575	0	0	24	0	0	606
hummingbad	0	3	0	0	0	543	0	0	0	4	550
judy	0	3	0	1	1	1	76	0	0	2	84
opfake	1	0	5	1	11	0	0	592	0	0	610
overlay	0	0	1	0	0	0	1	0	54	0	56
plankton	6	25	2	16	2	3	2	1	0	566	623
Σ	475	467	624	672	593	561	80	619	54	651	4796

Figure 5: Confusion Matrix for the *MLP 2* model.

lies). In this paper we propose a technique for mobile malware classification into malicious belonging family. In detail we propose the analysis of an audio stream, obtained from the Android application under analysis, to extract a set of numerical features. These features are the input for several machine learning classifiers that we evaluate with more than 4500 malware targeting the Android environment. We obtain an accuracy equal to 0.988 using a deep learning model designed by authors, showing that the proposed method can be effective for Android malware family detection. As future work, we plan to try to localise inside the audio wave the frame related to the malicious behaviour. Moreover, we will experiment the proposed method by using a dataset composed by iOS applications.

ACKNOWLEDGEMENTS

This work has been partially supported by MIUR - SecureOpenNets, EU SPARTA, CyberSANE and E-CORRIDOR projects.

REFERENCES

Arp, D., Spreitzenbarth, M., Hubner, M., Gascon, H., Rieck, K., and Siemens, C. (2014). Drebin: Effective and explainable detection of android malware in your pocket. In *Ndss*, volume 14, pages 23–26.

Azab, A. and Khasawneh, M. (2020). Msic: malware spectrogram image classification. *IEEE Access*, 8:102007–102021.

Bakhshinejad, N. and Hamzeh, A. (2017). A new compression based method for android malware detection using opcodes. In *2017 Artificial Intelligence and Signal Processing Conference (AISP)*, pages 256–261. IEEE.

Canfora, G., Martinelli, F., Mercaldo, F., Nardone, V., Santone, A., and Visaggio, C. A. (2018). Leila: formal tool for identifying mobile malicious behaviour. *IEEE Transactions on Software Engineering*, 45(12):1230–1252.

Enck, W., Gilbert, P., Han, S., Tendulkar, V., Chun, B.-G., Cox, L. P., Jung, J., McDaniel, P., and Sheth, A. N. (2014). Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones. *ACM Transactions on Computer Systems (TOCS)*, 32(2):1–29.

Farrokhmanesh, M. and Hamzeh, A. (2019). Music classification as a new approach for malware detection. *Journal of Computer Virology and Hacking Techniques*, 15(2):77–96.

Iadarola, G., Martinelli, F., Mercaldo, F., and Santone, A. (2020). Evaluating deep learning classification reliability in android malware family detection. In *2020 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, pages 255–260. IEEE.

Iadarola, G., Martinelli, F., Mercaldo, F., and Santone, A. (2021). Towards an interpretable deep learning model for mobile malware detection and family identification. *Computers & Security*, page 102198.

Jerome, Q., Allix, K., State, R., and Engel, T. (2014). Using opcode-sequences to detect malicious android applications. In *2014 IEEE International Conference on Communications (ICC)*, pages 914–919.

Kim, H. M., Song, H. M., Seo, J. W., and Kim, H. K. (2018). Andro-simnet: Android malware family classification using social network analysis. In *2018 16th Annual Conference on Privacy, Security and Trust (PST)*, pages 1–8. IEEE.

Michael, S., Florian, E., Thomas, S., Felix, C. F., and Hoffmann, J. (2013). Mobilesandbox: Looking deeper into android applications. In *Proceedings of the 28th International ACM Symposium on Applied Computing (SAC)*.

Nataraj, L., Karthikeyan, S., Jacob, G., and Manjunath, B. S. (2011). Malware images: visualization and automatic classification. In *Proceedings of the 8th international symposium on visualization for cyber security*, pages 1–7.

Nguyen, T., Mcdonald, J., Glisson, W., and Andel, T. (2020). Detecting repackaged android applications using perceptual hashing. In *Proceedings of the 53rd Hawaii International Conference on System Sciences*.

Popli, N. K. and Girdhar, A. (2019). Behavioural analysis of recent ransomwares and prediction of future attacks by polymorphic and metamorphic ransomware. In *Computational Intelligence: Theories, Applications and Future Directions-Volume II*, pages 65–80. Springer.

Vasan, D., Alazab, M., Wassan, S., Naeem, H., Safaei, B., and Zheng, Q. (2020). Imcfn: Image-based malware classification using fine-tuned convolutional neural network architecture. *Computer Networks*, 171:107138.

Zhou, Y. and Jiang, X. (2012). Dissecting android malware: Characterization and evolution. In *Proceedings of 33rd IEEE Symposium on Security and Privacy (Oakland 2012)*.