A Flow-Based Study for Android Malware Classification IDALAB Challenge 2021 *Team #5*

Anna Lamboglia, Daniel Parisi, Francesco Ottata
University of Napoli “Federico II” (Italy)
{ann.lamboglia, danie.parisi, f.ottata}@studenti.unina.it

***Abstract*—The advent of mobile traffic has increasingly led to the growth of mobile malware that can infect the devices of users using certain applications on the network. The main purpose of this research is to test, using Machine Learning and Deep Learning techniques, various approaches in order to find a model that is able to classify malicious and benign traffic, but also to classify malicious instances as general malware or specific malware (i.e. adware). Referring to the CIC-AAGM2017 dataset, three types of models were developed and compared: a flat classifier, a hierarchical classifier and a deep network. In particular, in the first two cases, the performance of the Decision Tree and Random Forest algorithms was evaluated, while in the deep approach a convolutional network (CNN) was used. The results obtained show that the Naive Hierarchical approach is potentially the best with an F1 measure of 85% and an average precision of 97%. In conclusion, it was seen that with all three approaches the classification of the general malware class is not very accurate, which is probably due to the unbalanced reference dataset.**

***Index Terms*—Android Malware, Malware Family, Malware Detection, Adware detection, Machine Learning,Deep Learning, Hierarcal Classification, traffic classification.**

I. Introduction

In the latest change with the spread of mobile devices connected to the growth of the internet, the dramatic spread of applications available for the latter and the spread of social networks, the number and behavior of users on the network is significantly. According to the statistics, in 2021 the daily active users on the network are 4.2 billion and mobile traffic corresponds to 55% of global traffic [1]. The 50% of this traffic is generated by Android devices, the most popular mobile operating system in the world, which together with Apple devices dominate the smartphone market scene. The success of Android is partly due to the versatility offered by being an open-source system, which makes it as powerful on the market as it is dangerous as it is vulnerable to all kinds of *malicious attacks*. Indeed, the trends show that hand in hand with the exponential growth the spread of Android mobile devices and applications in the world has also resulted in an increase in the presence of malicious software capable of causing serious damage such as monitoring of user activity and theft of private information.

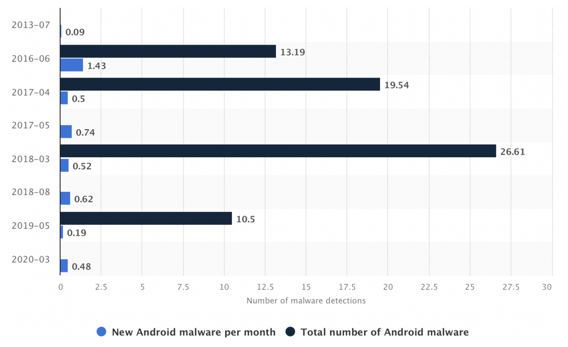

Fig. 1. Development of Android malware worldwide 2016-2020

Therefore, the *detection* and *prevention* of malware and adware of various types has become a challenge of primary importance for Android developers and for the entire Internet community. Generally, the malware detection techniques proposed by the state of the art up to now they are divided into *static techniques* and *dynamic techniques*: in the first case, the detection of symptom characteristics of malicious or incorrect code takes place without running the application; in the second, the malicious application is allowed to run in an isolated environment and by monitoring the traffic generated, an attempt is made to classify the harmfulness of the latter. The objective of the research is, therefore, to propose a model for detecting and classifying Android malware and adware traffic by making use of the power of Machine Learning and Deep Learning techniques. Hence, the studies and experiments that will be proposed in the following sections have as their main objective the identification of a model that distinguishes not only a simple discrimination of malicious traffic from benign traffic, but also the type of malicious traffic, which is the basis of the data available for this study, which we will discuss later.

II. Related Work

The first steps taken in our studies of research aimed at detecting Android malware and violations of user privacy through the classification of network traffic. The tests were carried out in a virtualized environment and using fictitious data such as personal data, passwords and bank details entered on an e-commerce application, which was then infected with 18 malware. Two approaches (IP / DNS blacklist, content matching) and 4 functionalities (HTTP header flag, HTTP-GET request, content and pattern of the POST request, well-structured identifiers in the POST request) were then tested to detect behaviors harmful. However, both solutions adopted had obvious limitations. Indeed, the blacklisting technique relies heavily on static malware behavior and requires frequent updates over time. Additionally, more complex attacks such as fast-flux and botnets are more difficult to detect effectively. The content matching technique infers information from simple data transmission and therefore cannot be applied in the case of encrypted traffic [2]. Subsequently new solutions were proposed, as a new network-based behavioral analysis to detect a new group of self-updating malware that were found in the Google Android marketplace, and which showed that this malware group is not detectable using classic static and dynamic analysis methods or with any approach based on signature recognition [3]. In the same year was presented a new automatic network profile generator for the detection of Android applications in HTTP. The latter argued that the traditional method of traffic classification is no longer useful for the analysis of mobile traffic and therefore introduced a new technique, which was also not free from limitations[4]. Other previous studies that are worth mentioning are those that led to the use of an Android emulator on a host with public IP to capture the traffic generated by the use of thirteen malware and then select on the basis of the previous experiments 16 features. By classification they discriminate the dataset into 3 risk levels, namely *high*, *medium* and *low* with an accuracy of 95%. However, their research had a weakness in the lack of heterogeneity of malware being 13 a small number[5]. Numerous other researches have been carried out in the years to come, and many of these are cited in great detail in the work carried out and presented in 2017 at the 15th Annual Conference on Privacy, Security and Trust by a research team of the Canadian Institute for Cybersecurity. (CIC)[6]. Our research is the specific daughter of the latter, having made use of the dataset they generated. The great contribution provided by this work was therefore the accurately labeling the collected traffic, whose samples were analyzed through the Androguard software and the ZipInfo command for the samples that were altered. Once the dataset was obtained, 24 features were selected on the basis of previous work, divided as follows: 1 based on behavior, 4 based on bytes, 4 based on packets, 4 based on time and 12 based on flow. In particular, 9 of the latter have been additionally proposed by the authors. For the extraction of this feature vector the special tool *CICFlowmeter* was deployed [7]. After dividing the dataset into testing and evaluation sets, the authors performed a features selection process using 3 different algorithms. The best results common to all led to the selection of 9 features in particular, all belonging to the flow based category. Indeed, the authors underline the importance of these features as unlike the other categories it is impossible for an attacker to modify or emulate them to evade the proposed system. At this point 3 scenarios have been defined to test and analyze the proposed model. First, a binary classification was made between benign and malignant, secondly a one vs one classification between benign and adware and benign and general malware, while in the third a classification for all classes. 5 classifiers were tested, i.e. Random Forest, Decision Tree, Random Tree, KNN and Regression in all 3 scenarios and all the results obtained were evaluated and documented. The validation of the model was then obtained on the validation dataset which represents 20% of the original dataset. Random Forest was the algorithm that generally achieves greater accuracy. This research and consequent model, although it makes a significant contribution to the state of the art in this field, also has its limits, which will be highlighted in the following sections in which we describe our work aimed precisely at improving the results obtained by the CIC team. A particular limitation, highlighted by the authors themselves in the documentation, is that of having generated the traffic through fictitious and non-manual interaction with the user, which makes the data partially insignificant for a real scenario. Another important contribution to the state of the art of mobile traffic classification and to our research is the definition of a possible structure that can be given to a dataset of this type in order to facilitate the classification tasks, as described in [8].

III. Dataset analysis

As previously explained, the dataset we used in our task is the one generated and labeled by the researchers of the CIC in 2017, named CIC-AAGM2017 [9]. Specifically, the approach used by the researchers made to collect the traffic use of the following steps. In the first instance, for the generation of the reference dataset, traffic generated not by an emulator but by a Google Nexus 5 device was captured. It was chosen to select the list of the first 1500 apps of the play store ranking for the generation of benign traffic, which were automatically installed and launched in rounds of 20 at a time. As for malicious traffic, 400 malware apps were chosen from adware to general malware. Specifically for both genres, applications belonging to popular families of malware have been chosen such as Airpsuh, Dowgin, Kemoge, Mobidash and Shuanet for adware and AVPASS, FakeAV, FakeFlash, GGtracker and Panetho for general malware. The traffic related to the latter was also captured by installing and launching the apps with the same procedure followed for benign traffic, but with the foresight to set up and configure the smartphone at each installation. We want in this section give an accurate description of them. It have a dimension of 9,5 Bb and is composed by PCAP file so divided:

- *Begnin*: 173 pcaps from 1500 apps;
- *Adware*: 62 pcaps from 250 apps;
- *General Malware*: 7 pcaps from 150 apps.

In the first analysis, the quantity of biflows by traffic categories was obtained. It is evident that there is a difference of several orders of magnitude between the types of biflows. This is a non-negligible factor in the course of our research, as will

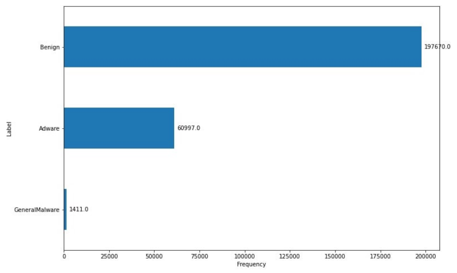

Fig. 2. Histogram representing the amount of biflows for each category

be seen in the following sections. The dataset contains 80% biflows of benign traffic and 20% of malicious traffic. As explaneid by the authors this imbalance is due to the desire to represent a scenario as realistic as possible, with the exception of the interaction with the user in the generation of the traffic itself.

A. *Information extracted from the dataset*

Other finer-grained analyzes were performed later. In particular the number of biflows for each single PCAP was extracted and added up to obtain the number of biflows for each category. It result:

- *Adware:* 60997 biflows;
- *General Malware:* 1411 biflows;
- *Benign:* 197670 biflows.

Other information that has been extrapolated is the average duration of traffic capture, equal to 2 hours and the most frequent IP address for each PCAP of each category.

IV. Proposed Logical Architecture and feature extraction

Following the analysis of the raw data set provided in PCAP format documented in the previous section, the problem arose as to how starting from it it was possible to extract useful features for subsequent classification tasks without using special tools such as FlowMeter and according to which structure to implement all for the purpose of classification. Knowing the power and versatility of Python dictionaries, using them seemed to us the best choice.

A. *Characterization of the dataset*

Before even structure the dataset, obviously the first step was to characterize the raw data to derive a series of features. This allowed us to extrapolate for each PCAPS the biflows identified by the classic quintuple made by *source address*, *destination address*, *source port*, *destination port* and *protocol*. To these 5 values another one has been added, the *timestamp*, transforming the quintuple into a sextuple. This was done to solve the problem of the uniqueness of the key if a quintuple was repeated more than once in the same PCAP. Hence, for each bi-flow then it was possible to extract some features considered particularly interesting as:

- The list of packet lengths for single biflow;
- The list of interarrival times (iat) between packets for single biflow;
- The volume of the biflow, intended as the total number of bytes;
- The total number of packets for biflows;
- The packet timestamp for single biflow;
- Payload;
- Payload length;
- The flow-rate of packets.

In particular with regards to the payload initially, this was taken entirely. However, having realized that this involved a computational power that we did not have, as well as problems due to the excessive size of the output files, the extraction of the payload was reduced to only the first 32 packets of each biflows. These features mentioned above have been identified by us as raw features being of a high level. At this point the lists obtained relating to the length of the packets and the interarrivals were characterized and analyzed in statistical terms. Therefore, again through the functions offered by the library, the following features have been obtained for both lists, defined as statistical features: (i) Minimum; (ii) Maximum; (iii) Average; (iv) Standard Deviation; (v) Variance; (vi) Madness; (vii) Skew; (viii) Curtosi indices; (ix) Percentiles from 10 to 90.

B. *Structure of the model*

To make all these features easily accessible a grafted dictionary was structured that contained both the raw and statistical characteristics and that contained all the files of a single category. This dictionary presents as a primary key that acts as a root a string representing the sextuple described above. At a first level there is then a second key which has three possible values: *"Raw Feature"*, *"Statistics Feature"*and *"Label"*. As the leaves of the tree structure there is as the last key for the raw features we have the parameter to be accessed, while as regards the statistical features a further level has been created in which the parameter is accessed first (in our case or packet length or iat) and then the statistic of interest (min, max, mean etc.). A second level has also been associated with the label key, which presents the distinction between first level and second level labels. This last choice in particular was a modification made upstream of the choice to adopt a hierarchical approach and therefore will be better explained later.

C. *ECDF extraction*

Before proceeding with the classification experiments, further information was extrapolated from the dataset for the purpose of evaluating the goodness of the features. Indeed, the ECDF graphs have been extrapolated for each category as regards the statistics of the length of the packets and the interarrival time. In order, we extract: (i) *ECDF of pcaps Adware packet length*, (ii) *ECDF of pcaps Adware iat*, (iii) *ECDF of pcaps General Malware packet length*, (iv) *ECDF of pcaps General Malware iat*, (v) *ECDF of pcaps Begnin packet length* and *ECDF of pcaps Begnin iat.*

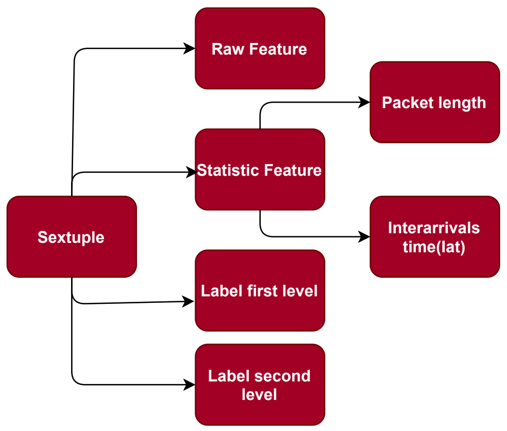

Fig. 3. Logical structure of nested dictionary

A more concise representation of the ECDFs has been made, showing them on a single graph 4 in relation to all packets of the individual classes, both for packet length and iat so as to have an immediate visual feedback.

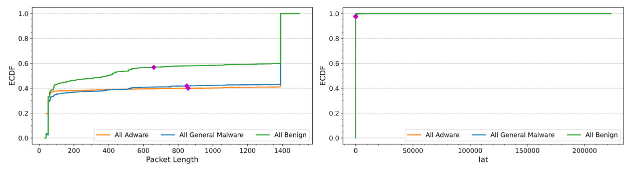

Fig. 4. ECDF Packet Length Total and ECDF Iat Total

The analysis of the ECDF has allowed us to highlight how in general the obtained statistics are significant for discriminating between benign and malware while they are probably less effective for the classification between adware and general malware.

V. Experiments

Assisting the very strong evolution of Internet and data, the dataset under consideration is a child of its time, as applications and related traffic generated a few years ago are considered. The core of this situation is to provide indicative parameters, additional information and artificial intelligence model (continuing the study carried out by those who created this dataset) that further enhance these data, with a projection of the interpretations that we are going to make towards a future of a technology that is constantly evolving.

As mentioned above, we divide our experiments into two different scenarios: the first involves the use of machine learning techniques combined with the sextuples whose features used are the statistical ones derived from the interarrival times and packet lengths associated with the sextuples themselves; the second approach involves deep learning techniques for implementing a one-dimensional convolutional neural network that processes the raw data extracted from the payloads associated with the sextuples. For both scenarios that we will propose, a splitting of the dataset into training and test set was performed according to an 80/20 proportion, keeping this stratification active also according to the available classes.

Scenario A: in a first step we propose the use of machine learning algorithms for multiclass classification. Among all the basic algorithms we have, our choice fell on the best flat classifier, namely the Random Forest Classifier, which is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset that operate as an ensemble. An analysis was conducted to observe the accuracy of the Random Forest Classifier as the number of estimators increased, in proportion also to the time spent during model training. The results saw that a hundred estimators is a very good trade-off between the execution time of training the model and the accuracy obtained, as you can see in Figure 5.

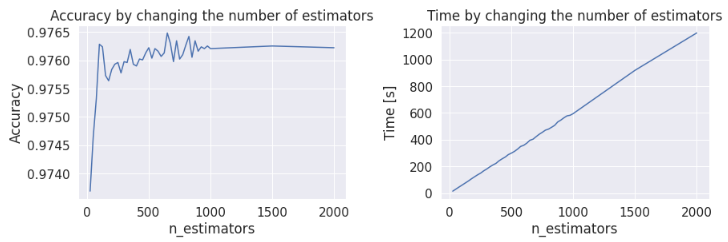

Fig. 5. Analysis of Random Forest Classifier accuracy and execution time with increasing number of estimators

Our experiments continue in search of a better performing model, both in considering an optimization of the hyperparameters of the two classifiers mentioned before, and in looking for an approach that better discriminates the three classes. The hierarchical classification approach comes into play, where the classification tree starts from a binary classifier that discriminates a benign or malware class (where for malware we considered instances whose labels are either adware or general malware) [10]. If an instance is predicted to be malware, then

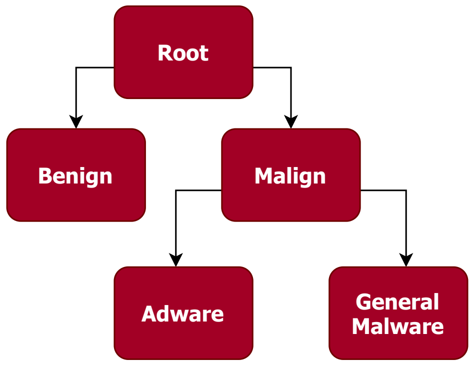

Fig. 6. Hierarchical classification structure

we move to the next level of classification, where an additional binary classifier discriminates between adware and general malware.

Scenario B: in this scenario, differently from the previous one, we implement a deep learning architecture that takes as input the first n bytes of the payload associated with a sextuple. Taking a step back, the first thing that was done is a manipulation of the data available so as to concatenate all the payloads associated with a sextuple, make a conversion from hexadecimal to decimal of the payload itself, and then call the first n bytes desired for the construction of the dataset to be fed to the neural network convolutional. We propose, in this case, three variants of the dataset that see the extraction of the first *512*, *784* and *1024* bytes of the payloads. For the sextuples that have empty associated payloads, a zero-padding function has been applied. For the structure of the dataset that we have of reference, we make use of a one-dimensional CNN sequential model, which sees the replication of a block consisting of a convolution layer and a max-pooling layer three times, with the introduction of a dropout layer to avoid over-training phenomena [11]. The choice to replicate these blocks three times also comes from the number of convolutional filters that are gradually increasing, going to identify patterns in our data at a finer grain. To deepen on what was done in this scenario, a further improvement of the model has been made, inserting a layer of Embedding that would provide additional semantics and expressiveness to the input.

A. Performance Metrics

For both scenarios we're going to look at some key metrics to evaluate their performance. To be more precise, we will consider the metrics *accuracy* (acc), *precision* (prec), *recall* (rec) - calculated from the indices True Positive (TP), False Positive (FP), False Negative (FN), True Negative (TN) - and F-measure, resulting from the calculation of the first two metrics. In addition, we will use confusion matrices to provide a graphical representation that showcases classification patterns at a finer grain. From this table, then, it is possible to understand the performance of a predictive classification model in order to determine how accurate and effective our model is.

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

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

$$F1 = \frac{2 * Prec * Rec}{Prec + Rec} = \frac{2 * TP}{2 * TP + FP + FN} \tag{4}$$

VI. Experimental Evaluation

In this section, we show the experimental results for the scenarios defined in the previous section on finding the best approach.

*1) **Scenario A**:* In this scenario the statistical features extracted from the dataset as explained in section IV-A, related to packet length and interarrival times. They were used to train and compare the results of the Decision Tree algorithm with those of the Random Forest. With regard to the "flat" approach, i.e. for the classification of all three classes: *Adware*, *Benign* and G*eneral Malware*, it was seen that RF gives us the best results with 98% of accuracy. The results in terms of accuracy, F-measure are shown in the table VI-1.

Also for the hierarchical approach we tried to find the best classifier for both levels. As for the flat approach, the algorithm RF turns out to be a better solution than the simple decision tree. From the Figure 7 it can be highlighted how the performances are very good in classifying between Benign and Malware, while they are not entirely satisfactory at the second level of the hierarchy where the instances classified as malicious are discriminated between adware and general malware. It is also necessary to take into account the inevitable phenomenon of error propagation. Indeed, it is clear how benign instances incorrectly classified as malicious at the first level negatively affect the performance of the second classifier.

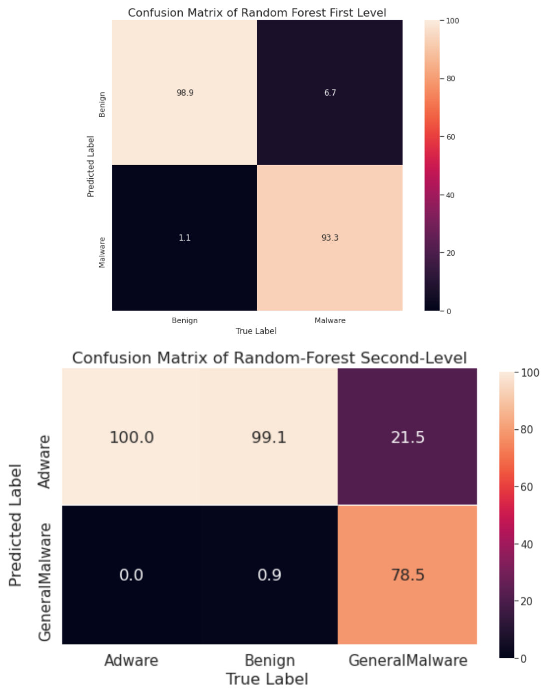

Fig. 7. Confusion Matrix of Random-Forest First-Level and Second-Level

A. Naive Hierarchical vs Best Flat Classifier

We now compare the resulting matrices and reports using both approaches.

Taking into consideration the respective total confusion matrices (Figure 8) and following the analyzes reported, the results appear quite similar. However, through the hierarchical approach, a small increase in performance is obtained and furthermore it has a much better potential.

*1) **Scenario B**:* The convolutional neural network architecture considered for this scenario is described in the table VI-A1; this network has been trained considering active the callbacks of early stopping and learning rate reduction. Among the best optimizers [12], Adam was considered in this case, using its default parameters to compile our model.

Scenario	**A (Machine Learning Classifiers)**				**B (Deep Learning)**	
Case	**RF - Flat**	**RF - Naive Hierarchical**	**DT - Flat**	**DT - Naive Hierarchical**	**1D - CNN**	**1D - CNN + Embedding Layer**
Precision	0.80	0.97	0.80	0.77	0.97	0.97
Recall	0.97	0.80	0.79	0.80	0.75	0.77
F1-Measure	0.86	0.85	0.80	0.79	0.82	0.83

TABLE I
TABLE COMPARING THE CLASSIFIERS AND MODELS USED IN THE SCENARIOS

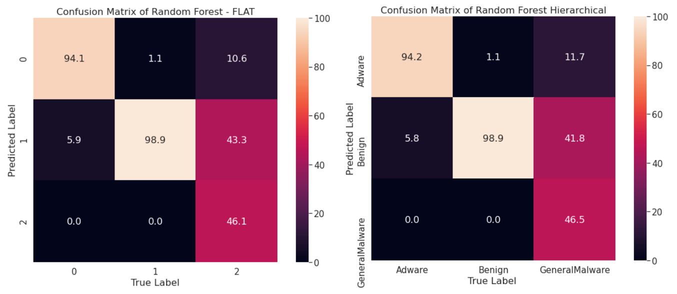

Fig. 8. Confusion Matrix of Random-Forest Flat vs Confusion Matrix Total of Random-Forest Hierarchical

Layer	**Type**	**Filters**	**Other Parameters**
1	Input	-	Input_shape = (n_bytes,1)
2	Embedding	-	Input_dim=256, output_dim=10
3	Convolutional1D	16	Activation = ReLU
4	MaxPooling1D	-	Pool Size = 3
5	Convolutional1D	32	Activation = ReLU
6	MaxPooling1D	-	Pool Size = 3
7	Convolutional1D	64	Activation = ReLU
8	MaxPooling1D	-	Pool Size = 3
9	Flatten	-	
10	Dropout	-	Rate = 0.25
11	Dense	256	
12	Dropout	-	Rate = 0.25
13	Dense	3	

TABLE II
CONVOLUTIONAL NEURAL NETWORK ARCHITECTURE PROPOSED FOR THIS SCENARIO

Note that the use of the Embedding layer was added at a later moment, as a further refinement to provide a better representation of the input.

VII. CONCLUSION

In our research, therefore, we have proposed a malware traffic classification model following three approaches, namely a flat classifier, a hierarchical classifier and a neural network, characterizing the traffic with flow-based features as the latter are difficult to modify by potential attackers who want to try to evade the model. All three approaches return similar results, with particular difficulty in classifying general malware. This problem is due to the strong class imbalance given by the dataset we've worked with. In future work, it would certainly be interesting to further develop both the flat and the hierarchical approach, carrying out a more precise hyperparameters optimization and testing the best combinations of algorithms not tested in this research. A further improvement would be to insert a per-classifier *Reject Option* before each level of the hierarchy, in order to limit the errors propagated from level to level.

In the same way, the deep learning approach could be expanded using RNN networks using GRU or LSTM cells in combination with *Attention layers* in such a way as to recognize patterns along the payload sequence.

ACKNOWLEDGMENT

We would like to sincerely thank Professor Antonio Pescapé, Full Professor at the Department of Electrical Engineering and Information Technology of the University of Naples Federico II, as well as lecturer of the Internet Data Analysis course, and Engineers Antonio Montieri and Alfredo Nascita who tutored us during the course of our work, with great cordiality and availability. Our thanks also go to engineers Valerio Persico, Gianpaolo Bovenzi, Giuseppe Aceto and Domenico Ciuonzo who helped provide us with the knowledge we needed to complete our research, as well as to all the colleagues who took part in the ”20/21” Internet Data Analysis class for their healthy discussions and opinions.

REFERENCES

[1] J. Johnson. Development of new android malware worldwide from june 2016 to march 2020. [Online]. Available: https://www.statista.com/statistics/680705/global-android-malware-volume/

[2] D. Iland, A. Pucher, and T. Schäuble, “Detecting android malware on network level,” 12 2011.

[3] L. Tenenboim, O. Barad, A. Shabtai, D. Mimran, L. Rokach, B. Shapira, and Y. Elovici, “Detecting application update attack on mobile devices through network features,” 04 2013.

[4] S. Dai, A. Tongaonkar, X. Wang, A. Nucci, and D. Song, “Networkprofiler: Towards automatic fingerprinting of android apps,” in *2013 Proceedings IEEE INFOCOM*, 2013, pp. 809–817.

[5] A. Arora and S. K. Peddoju, “Minimizing network traffic features for android mobile malware detection,” in *Proceedings of the 18th International Conference on Distributed Computing and Networking*, ser. ICDCN ’17. New York, NY, USA: Association for Computing Machinery, 2017. [Online]. Available: https://doi.org/10.1145/3007748.3007763

[6] A. Habibi Lashkari, A. F. Abdul kadir, H. Gonzalez, K. Mbah, and A. Ghorbani, “Towards a network-based framework for android malware detection and characterization,” 08 2017, pp. 233–23 309.

[7] A. Habibi Lashkari, “Cicflowmeter-v4.0,” https://github.com/ahlashkari/CICFlowMeter, 2017.

[8] G. Aceto, D. Ciuonzo, A. Montieri, V. Persico, and A. Pescapè, “Mirage: Mobile-app traffic capture and ground-truth creation,” 10 2019.

[9] C. I. for Cybersecurity, “Android adware and general malware dataset (cic-aagm2017),” 2017, data retrieved for the first time in 03/31/2021 at https://www.unb.ca/cic/datasets/android-adware.html.

[10] A. Montieri, D. Ciuonzo, G. Bovenzi, V. Persico, and A. Pescapé, “A dive into the dark web: Hierarchical traffic classification of anonymity tools,” *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 3, pp. 1043–1054, 2020.

[11] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang, “End-to-end encrypted traffic classification with one-dimensional convolution neural networks,” 07 2017, pp. 43–48.

[12] F. Chollet *et al.*, “Keras,” 2015, https://keras.io/api/optimizers/.