A study on malware detection and classification using the analysis of API calls sequences through shallow learning and recurrent neural networks

Angelo Cannarile[2], Francesco Carrera[2], Stefano Galantucci[1,*], Andrea Iannacone[2] and Giuseppe Pirlo[1]

[1]*University of Bari Aldo Moro, Department of Computer Science, 70125 Bari, Italy*
[2]*BVTech SpA, 20123 Milano, Italy*

Abstract
Malware detection and classification is a critical issue in cybersecurity. Systems acting through signatures suffer the problem of not being able to detect attacks via zero-day malware. Among the approaches that can detect unknown attacks are the possibilities offered by analyzing the sequence of API calls performed by the executable. Such information can be extracted through static and dynamic analysis methods in a sandbox environment. This work proposes an analysis of different techniques to detect malware and subsequently classify them by identifying the family of belonging. Machine Learning algorithms based on trees are compared with Deep Learning algorithms based on Recurrent Neural Networks. The results obtained lead to choosing an algorithm based on RNNs for malware detection and an algorithm based on trees for malware classification.

Keywords
Malware, Classification, Detection, Trees, Neural network, malware-analysis-datasets-api-calls-sequences, APIMDS, Catak, API Calls

1. Introduction

The constant growth of the development and diffusion of new technologies expose systems to potential risks. Among the main risks is malware, malicious software created to steal, spy, or, more in general, damage infected systems. In order to mitigate this risk, it is necessary to adopt tools that can detect, classify and block these threats on time. The literature analysis shows that most malware detection systems are based on static analysis techniques such as signature verification, which are very good for all known malicious software, but at the same time, ineffective for new malware since the signatures of these applications are not yet available. An alternative approach is using sandbox information (such as CAPEv2), which provides dynamic application analysis. The sandbox software encapsulates the information extracted in reports for each file analyzed. The reports contain the ordered sequence of API calls called

ITASEC'22: Italian Conference on Cybersecurity, June 20–23, 2022, Rome, Italy
*Corresponding author.
a.cannarile@bv-tech.it (A. Cannarile); f.carrera@bv-tech.it (F. Carrera); stefano.galantucci@uniba.it (S. Galantucci); a.iannacone@bv-tech.it (A. Iannacone); giuseppe.pirlo@uniba.it (G. Pirlo)
0000-0002-3955-0478 (S. Galantucci); 0000-0002-7305-2210 (G. Pirlo)

from the analyzed file, which approximates the behavior of the software. The analysis of API calls sequences through machine learning models to detect both known malware and as-yet-unknown (zero-day) malware is of great interest to researchers. Once the threat is detected, using machine learning models, it is possible to determine the malware's type based on the information extracted from the dynamic analysis. Labeling the detected malware with the appropriate family allows experts in the field to optimize the analysis time and thus minimize the response time for each identified threat. This work is proposed as an analysis of several known techniques in order to detect and classify malware.
The goals of this paper are:

- Compare the performance of tree-based Machine Learning algorithms and Recurrent Neural Networks (RNN)-based Deep Learning algorithms for the analysis of API calls sequences;
- Evaluate and identify which basic models perform well for malware detection based on API call sequence analysis;
- Compare the performance of API calls-based models for malware classification in their category.

This paper is organized as follows. In section 2 the most interesting approaches related to malware detection and classification models based on the analysis of API calls sequences are reported. Section 3 describes the algorithms that are used in this work. Section 4 details the performed experiments and their notable results. Conclusions and possible future developments of the proposed work are presented in section 5.

2. Related work

Malware analysis systems can leverage machine learning algorithms to extract meaningful patterns from data to detect the presence of malware and recognize the category to which the malware belongs. An innovative approach involves using the sequence of API calls invoked by the malware itself as a "trace" of its behavior within the system. The following section analyzes the works that represent state of the art for Malware Detection and Malware Classification systems based on API Call sequences analysis.
Ki et al. [1] propose a new approach to dynamic malware analysis. Through DNA sequence alignment algorithms, the authors can verify that malware belonging to the same category share many API calls sub-sequences. Based on these, the classification of malware into different categories is performed. The dataset created by the authors, used in the manuscript and later made public, is also used in the present work. More details about the description of the dataset can be found in the experiments section. The authors also propose *APIMDS* (API Malware Detection System), a malware detection system based on the analysis of API calls sequences. Once the API calls of the software to be analyzed are extracted, they are compared with those included in the dataset. In case of correspondence, *APIMDS* notifies the presence of the threat to the system administrator.
In a previous work [2], a study was carried out regarding the main algorithms used for Malware Detection. In detail, different shallow and deep learning techniques have been compared to

recognize if the analyzed file is malware or goodware. The analysis is based on the API Call sequences invoked by the analyzed file. These APIs are first collected by a dynamic file analysis in a sandbox, i.e., CAPEv2. Performance on two different datasets was compared using six different techniques. The results show that CatBoost performs better on the F1 Score and AUC ROC. The method developed in [3] was also used for balancing the datasets.
State-of-the-art Malware Classification systems use Deep Learning algorithms to learn patterns that can detect and classify Malware families [4]. Several Deep Learning algorithms used to classify textual sentences are used to analyze API call sequences and classify Malware families [5, 6].
Several supervised Machine Learning algorithms have been used in the literature to detect and classify malware [7]. Considering the characteristics of the API call sequences to be analyzed, traditional machine learning models may not be sufficient since the order of the calls needs to be preserved. Considering the relationships and order between API calls could improve traditional Malware Detection and Classification systems.
In [8] the classification of malware based on API calls sequences is done after n-gram extraction, a Natural Language Processing technique. Based on the formed n-grams, an unsupervised algorithm named Voting Expert extracts the patterns, which are subsequently used for training. Tran et al. [9] propose a combined approach between Memory Augmented Neural Network and Natural Language Processing techniques to solve the malware classification problem. The input for the proposed algorithm is prepared from the sequence of API calls by modeling them in a different feature space. The performance of the model has been evaluated on two datasets, including *APIMDS*, and compared with some traditional models such as Random Forest, SVN, and LSTM.
Catak et al. [10] use a Long Short-Term Memory (LSTM) architecture-based model to analyze API call sequences and classify the malware family. The performance obtained using traditional classification algorithms with single and multi-layer LSTM architectures are compared. A main contribution of the work is building a new dataset for malware detection on Windows. The dataset created for this purpose contains 7,107 samples labeled across 8 malware families. Finally, the authors show that the LSTM architecture allows the creation of a classifier that improves the performance of traditional algorithms when used to analyze sequences of API calls. An interesting development in this context is described in [11], in which the main problems affected by machine learning and deep learning models are analyzed:

- Failure to recognize malware belonging to categories not addressed in training data;
- The scarce amount of malware examples for category.

To overcome these issues, the authors present an innovative model of Malware Classification called SIMPLE. Based on the input API calls sequences, they are pre-treated through a pre-trained word embedding technique and finally modeled by a Long Short Term Memory network to preserve the sequence information. Unlike other models of the same type, the output is represented through multiple prototypes generated by clustering. The presented model achieves the best accuracy concerning the reference models.
In [12] the authors propose a methodology that uses feature extraction and feature selection techniques to process sequences for analysis by classification algorithms.
Recurrent Neural Networks and LSTM have been extensively used in Pattern Recognition to

exploit temporal sequence as in [13, 14]. Li et al. [15] propose to use Recurrent Neural Networks to analyze ordered sequences of API calls and classify malware families. Since each malware can invoke sequences of API calls of different lengths, the authors compare the use of two popular types of RNNs, which are widely used to analyze time series. In the above work, Long Short-Term Memory and Gated Recurrent Unit (GRU) algorithms are used to clarify the best RNN architecture for malware classification by analyzing long sequences of API calls. In the described experiments, a reference dataset was used to evaluate and compare the performance of the classifiers. The results showed that the LSTM model and the GRU model achieve very similar performance, and both are effective at classifying malware through the API call sequence analysis.
A classifier called "Random Transformer Forest" is proposed in [16], which uses a Transformer forest for analyzing API call sequences and categorizing malware categories. Unlike traditional machine learning and deep learning models, Transformer-based models process the sequences as a whole and learn the relationships between API calls through Multi-Head Attention mechanisms and positional embeddings. The reported experiments show that the proposed model outperforms the performance obtained using an LSTM architecture. Moreover, it is shown that BERT or CANINE, the pre-trained Transformer models, reach better performance when classifying highly unbalanced malware families.

3. Algorithms used

Consider the goal of implementing a malware detection system. Such a system should also be able to classify the detected malware. It is desired that the system performs the operations by evaluating information extracted from the dynamic analysis of files. Therefore, it is necessary to obtain a workflow that can perform the operations mentioned earlier, which are, in order: dynamic analysis of the executable, identification of its type (malware/goodware), and, if it is a malware, classification in its specific family. An isolated and secure environment is used for the dynamic analysis, i.e., CAPEv2, which examines the file and returns a report containing the sequence of API calls invoked by the analyzed file. Thus, it is desired to find which Machine Learning and Deep Learning algorithms achieve the best performance when used to analyze API call sequences.
The proposed system uses a two-step approach to increase the detection rate of malicious files and improve the accuracy in classifying malware families. In detail, the system uses in the first step a Malware Detection algorithm to detect malware, which, regardless of the family of belonging, must be blocked and analyzed by experts in the field. During the second step, a Malware Classification algorithm is used to support the analysis of malicious files; it inputs the sequence of API calls captured for the malware in question and outputs the corresponding family.
Therefore, according to the Malware Detection model, it is necessary to build a binary classifier, which allows for detecting as many malware as possible; instead, in the second step, it is necessary to build a multi-class classifier, which accurately assigns to each malware the appropriate class.
The classification algorithms analyzed in this paper are now reported. All the algorithms exam-

ined can be used both to build binary classification models, i.e., to build the Malware Detection model; and for multiple classification problems, i.e., to build the Malware Classification model. From the review of the literature conducted in the previous section, it can be seen that some of the most interesting approaches are tree-based classifiers and deep learning algorithms generally used to classify textual sequences; based on this study, a choice of algorithms to be analyzed was made.
The algorithms examined in the proposed work are:

- **Random Forest** [17]: classifier obtained from an aggregation, through bagging technique, of decision trees;
- **CatBoost** [18]: machine learning method based on Gradient Boosting applied to decision trees;
- **XGBoost** [19]: algorithm for creating prediction models from an ensemble of smaller prediction models, represented by decision trees;
- **ExtraTrees** [20]: algorithm that aggregates the results of multiple unrelated decision trees into a forest that produces the classification output. The idea behind it is most similar to the Random Forest algorithm and differs from it only in the way the decision trees are constructed in the forest;
- **TabNet** [21]: Neural network for tabular data processing;
- **Bidirectional Long Short-Term Memory (Bi-LSTM)**: evolution of the model Long Short Term Memory [22], belongs to the category of RNN. It is defined bidirectional since the flow of information propagates in both directions;
- **Bidirectional Gated Recurrent Unit (Bi-GRU)**: evolution of the Gated Recurrent Unit [23], belongs to the category of RNNs. It uses a bidirectional approach to analyze sequences in both directions as well.

4. Experimental setup

This section describes the experiments performed in the proposed work. All experiments were executed on a machine with 16 cores, 38 GB of RAM, and Ubuntu 20.04.3 operating system. Section 4.1 describes the datasets used to evaluate the algorithms considered in this work. In sections 4.2 and 4.3, the results obtained in the experiments performed for the construction of the Malware Detection and Malware Classification model, respectively, are presented. The obtained results and the achievement of the proposed goals are discussed in section 4.4.

4.1. Dataset

The *malware-analysis-datasets-api-call-sequences* and *APIMDS* datasets were examined to evaluate malware detection algorithms. The *malware-analysis-datasets-api-call-sequences* [24] dataset contains 42,797 malware-related API call sequences and 1079 goodware API call sequences. Each API call sequence is made of the first 100 consecutive non-repeating calls associated with the parent process, extracted from the "calls" elements of the reports obtained from Cuckoo Sandbox. For *malware-analysis-datasets-api-call-sequences* dataset, there are no labels for the malware categories. The *APIMDS* [1] dataset was constructed from the random selection of records belonging to the Malicia-Project and VirusTotal malware datasets. It collects the API calls sequences of 23080 malware and 300 goodware. The extracted sequences contain 2727 different API calls. The following classes represent the malware types:

- Backdoor;
- Worm;
- Packed;
- PUP;
- Trojan;
- Misc.

Since the *APIMDS* dataset contains both malware family labels and records labeled as goodware, it is used in the present work for both the malware detection task and the malware classification task. For the evaluation of the malware classification algorithms, the *Catak* and *APIMDS* datasets were considered.
To build the *Catak* dataset [10], the author analyzed within an isolated Cuckoo Sandbox environment several malware samples, for each of which the ordered sequence of API calls made to the Windows operating system was recorded. In each sequence, API calls can be present in a repeated form, even consecutively. The extracted sequences contain 162 different API calls. The dataset is labeled with different types of malware and does not contain any information related to goodware analysis. For this reason, if a supervised approach is used, the examined dataset can be used to perform Malware Classification but cannot be used for Malware Detection.
The dataset contains 7,107 samples, which are labeled using the following Malware categories:

- Spyware;
- Downloader;
- Worms;
- Adware;
- Trojan;
- Backdoor.
- Virus;
- Dropper.

4.2. Malware detection experiments

Starting from the state of the art study on algorithms carried out in [2], in the present section it is aimed to compare the results obtained in the previous work with Deep Learning algorithms used to analyze temporal sequences. The previous work is now extended by repeating the experiments, under the same conditions, with Bi-LSTM and Bi-GRU models. The experiments were performed by analyzing the *malware-analysis-datasets-api-call-sequences* and *APIMDS* datasets. Tests were not repeated on the *Catak* dataset because, as expressed earlier, it does not contain API call sequences related to goodware files.
Tables 1 and 2 shows the results obtained from the Malware Detection experiment on the *malware-analysis-datasets-api-call-sequences* (Table 1) and *APIMDS* (Table 2) datasets. Specifically, the results are computed through stratified cross-validation with k equal to 10 and

averaged across them. Accuracy, Recall, and F1-Score metrics are calculated with macro averaging. The tables also contain the training and prediction times measured by the Malware Detection experiment. The training time represents the average training times measured to build the Malware Detection model in the 10 k-Fold experiments. In the same way, the prediction time represents the average of the prediction times measured to identify the malware contained in the test dataset in the 10 experiments of the k-Fold.

Table 1
Average metrics computed for the 10 folds in the malware detection experiment on the *malware-analysis dataset-api-call-sequences* dataset. Training and prediction time (in seconds) are the average of the respective times measured in the stratified k-Fold

Algorithm	Precision	Recall	F1 Score	AUC ROC	Accuracy	Training time	Prediction time
Random Forest	0.965	0.809	0.869	0.809	0.990	15.957	0.088
CatBoost	0.962	0.820	0.877	0.820	0.990	8.991	**0.011**
XGBoost	0.957	0.781	0.847	0.782	0.988	8.740	0.032
ExtraTrees	**0.970**	0.741	0.816	0.741	0.986	**6.673**	0.089
TabNet	0.898	0.767	0.817	0.767	0.986	198.780	0.188
Bi-LSTM	0.926	0.882	0.903	0.882	0.991	1319.353	3.600
Bi-GRU	0.950	**0.893**	**0.918**	**0.893**	**0.993**	2013.649	3.287

Table 2
Average metrics computed for the 10 folds in the malware detection experiment on the *APIMDS* dataset. Training and prediction time (in seconds) are the average of the respective times measured in the stratified k-Fold

Algorithm	Precision	Recall	F1 Score	AUC ROC	Accuracy	Training time	Prediction time
Random Forest	0.991	0.921	0.953	0.921	0.986	6.314	0.034
CatBoost	0.995	0.940	0.965	0.940	0.990	15.553	0.034
XGBoost	0.994	0.962	0.978	0.963	0.998	7.158	**0.014**
ExtraTrees	0.995	0.913	0.950	0.913	0.984	**1.481**	0.036
TabNet	0.984	0.926	0.953	0.926	0.997	2950.518	1.550
Bi-LSTM	0.990	0.970	0.979	0.970	**0.999**	1198.291	4.057
Bi-GRU	**0.996**	**0.979**	**0.988**	**0.979**	**0.999**	326.068	2.678

4.3. Malware classification experiments

This section describes the experiments performed to compare classification algorithms for malware family identification. Specifically, the performance obtained using the Machine Learning and Deep Learning algorithms described in section 3 were compared. The experiments were performed using the *Catak* and *APIMDS* datasets. Tests were not repeated on the *malware-analysis-datasets-api-call-sequences* dataset since it does not contain malware family labels.
To build the Malware Classification models, all sub-sequences with consecutively repeated API calls were removed, leaving only one call for each sub-sequence. There is a high concentration of null values since not all records have the same amount of calls. In order to mitigate this problem, the columns in the dataset in which there is a presence of null values greater than 20% have been removed.

Tables 3 and 4 presents the calculated results for each algorithm derived from the stratified cross-validation experiment with k equal to 10 and averaged with each other. The Accuracy, Recall, F1-Score metrics are calculated with macro averaging, instead the AUC ROC value was calculated according to the "one vs rest" approach. The reference dataset for table 3 is *APIMDS*, while table 4 presents the results obtained on the dataset *Catak*.
The tables also include the training and prediction times measured by the Malware Classification experiment. As in previous experiments, the training time is the average of the training times measured to build the Malware Classification model in the 10 k-Fold experiments. Similarly, the prediction time represents the average of the prediction times measured to classify the malware family contained in the test dataset for the 10 k-Fold experiments.

Table 3
Average metrics computed for the 10 folds in the malware classification experiment on the *APIMDS* dataset. Training and prediction time (in seconds) are the average of the respective times measured in the stratified k-Fold

Algorithm	Precision	Recall	F1 Score	AUC ROC	Accuracy	Training time	Prediction time
Random Forest	0.884	0.694	0.760	0.821	**0.904**	783.183	1.973
CatBoost	0.901	0.663	0.737	0.804	0.899	52.084	**0.030**
XGBoost	**0.911**	0.615	0.688	0.776	0.888	**45.911**	0.057
ExtraTrees	0.878	0.698	**0.762**	0.824	**0.904**	2289.088	7.329
TabNet	0.778	0.362	0.683	0.786	0.877	4136.281	2.026
Bi-LSTM	0.770	**0.715**	0.737	**0.833**	0.863	1456.415	5.299
Bi-GRU	0.766	0.704	0.730	0.826	0.865	1827.840	5.317

Table 4
Average metrics computed for the 10 folds in the malware classification experiment on the *Catak* dataset. Training and prediction time (in seconds) are the average of the respective times measured in the stratified k-Fold

Algorithm	Precision	Recall	F1 Score	AUC ROC	Accuracy	Training time	Prediction time
Random Forest	0.591	0.568	0.575	0.752	0.555	1822.669	1.939
CatBoost	0.521	0.509	0.512	0.718	0.497	97.479	**0.025**
XGBoost	0.469	0.452	0.451	0.685	0.439	**19.450**	0.028
ExtraTrees	**0.593**	**0.571**	**0.578**	**0.753**	**0.557**	754.477	2.332
TabNet	0.334	0.336	0.326	0.619	0.318	11454.760	2.175
Bi-LSTM	0.521	0.514	0.516	0.721	0.505	947.194	4.047
Bi-GRU	0.506	0.506	0.504	0.716	0.494	788.007	2.827

4.4. Discussion

The above section results show that the considered algorithms constitute the state of the art for malware detection and classification systems. In the related experiment for Malware Detection, it can be observed that Deep Learning algorithms do not always achieve better performance than Machine Learning algorithms and vice versa. It is notable that, for both datasets analyzed for Malware Detection, Bi-LSTM and Bi-GRU algorithms (which are generally used to classify textual sequences) always achieve higher performances than tree-based algorithms. According to the results reported in Tables 1 and 2, the Bi-LSTM and Bi-GRU algorithms outperform

the CatBoost algorithm, which was identified as the most suitable algorithm for building the Malware Detection system in previous work [2]. Maximizing the Recall metric is necessary for the Malware Detection system, as it is an index that expresses how many of the malware instances are correctly classified as malicious files and how many of the goodware instances are detected as non-malicious files. In both datasets analyzed for Malware Detection, the Bi-GRU algorithm performed best in terms of Recall, AUC ROC, F1-Score, and Accuracy.
To identify the algorithm to be used in the second phase, it is necessary to analyze the results reported in section 4.3. The ExtraTrees algorithm outperformed the performance obtained for the *Catak* dataset in the [15, 10] works. Even for experiments related to malware family classification, the results prove that Machine Learning algorithms do not always outperform Deep Learning algorithms and vice versa. Unlike the Malware Detection model, the Accuracy index must also be maximized for the Malware Classification model. Indeed, the label on the type of malware assigned by the classification model must be correct to make this information helpful to experts who analyze files that are considered malicious.
For this reason, when choosing the Malware Classification model, it is necessary to consider the F1-score index, which is a harmonic average of the Accuracy and Recall scores. From the results shown in tables 3 and 4, it can be observed that the algorithm that overall gets the best results is ExtraTrees. For the *APIMDS* dataset, the ExtraTrees algorithm scores the highest values for the F1-score and Accuracy metrics and one of the highest AUC ROC index scores. For the *Catak* dataset, all metrics computed in the experiments using the ExtraTrees algorithm are higher than the metrics obtained using the other surveyed algorithms.
For completeness, observing the temporal performances, it emerges how the best techniques allow recognition in relatively short times, even if not suitable for a context with large volumes of data. The training phase is the heaviest; however, this can be carried out cyclically, such as during the hours in which there is less traffic to analyze.

5. Conclusions and future work

Aiming at realizing a workflow for malware analysis, experimental analysis was carried out between the performances of malware detection and classification algorithms based on the analysis of API call sequences. From the obtained results, it is possible to state that defining the best algorithm for the addressed tasks is not always possible. In the experiments carried out for Malware Detection, the Bi-GRU algorithm has obtained the best performances in terms of Recall, AUCROC, F1-Score, and Accuracy in both analyzed datasets. On the other hand, the analysis of the experiments performed for Malware Classification shows the best results by the ExtraTrees decision tree-based algorithm.
This study shows that the algorithms based on the RNN architecture achieve great performance in malware detection. RNNs can work on sequences of an arbitrary length, overcoming the limitations imposed by other tree-based ones such as CatBoost. This feature allows Malware Detection models to generalize over new records and identify sub-sequences of API calls that can be traced back to malware. The results reported in section 4.2 show that RNN-based algorithms detect more malicious files than tree-based algorithms, as they achieve a significantly higher Recall score at the cost of comparable accuracy with the models analyzed in previous work.

Unlike the detection model, higher classification accuracy is needed in malware classification cases. Since many types of malware invoke similar API sequences, tree-based algorithms can better discriminate the particular malware type.
Future developments of this work will focus on studying pre-processing techniques that can remove the noise in the API calls sequences and optimize the discrimination ability of the classification models. Moreover, it is of great interest to investigate the use of ensemble learning techniques based on innovative RNN architectures. Future analysis will also evaluate the applicability of such solutions in contexts with large volumes of data to be analyzed, i.e., in systems where time is critical.

Acknowledgments

This work was partially supported by the Fondo Europeo di Sviluppo Regionale Puglia Programma Operativo Regionale (POR) Puglia 2014–2020 Axis I–Specific Objective 1a–Action 1.1 (Research and Development) – Project Title: CyberSecurity and Security Operation Center (SOC) Product Suite by BV TECH S.p.A., under Grant CUP/CIG B93G18000040007.

References

[1] Y. Ki, E. Kim, H. K. Kim, A novel approach to detect malware based on api call sequence analysis, International Journal of Distributed Sensor Networks 11 (2015) 659101.
[2] A. Cannarile, V. Dentamaro, S. Galantucci, A. Iannacone, D. Impedovo, G. Pirlo, Comparing deep learning and shallow learning techniques for api calls malware prediction: A study, Applied Sciences 12 (2022) 1645.
[3] V. Dentamaro, D. Impedovo, G. Pirlo, Licic: less important components for imbalanced multiclass classification, Information 9 (2018) 317.
[4] B. Kolosnjaji, A. Zarras, G. Webster, C. Eckert, Deep learning for classification of malware system call sequences, in: Australasian joint conference on artificial intelligence, Springer, 2016, pp. 137–149.
[5] Q. Li, H. Peng, J. Li, C. Xia, R. Yang, L. Sun, P. Yu, L. He, A text classification survey: from shallow to deep learning, arXiv preprint arXiv:2008.00364 (2020).
[6] S. Minaee, N. Kalchbrenner, E. Cambria, N. Nikzad, M. Chenaghlu, J. Gao, Deep learning–based text classification: a comprehensive review, ACM Computing Surveys (CSUR) 54 (2021) 1–40.
[7] D. Ucci, L. Aniello, R. Baldoni, Survey of machine learning techniques for malware analysis, Computers & Security 81 (2019) 123–147.
[8] A. Pektaş, T. Acarman, Malware classification based on api calls and behaviour analysis, IET Information Security 12 (2018) 107–117.
[9] K. Tran, H. Sato, M. Kubo, Mannware: A malware classification approach with a few samples using a memory augmented neural network, Information 11 (2020) 51.
[10] F. O. Catak, A. F. Yazı, O. Elezaj, J. Ahmed, Deep learning based sequential model for malware analysis using windows exe api calls, PeerJ Computer Science 6 (2020) e285.
[11] P. Wang, Z. Tang, J. Wang, A novel few-shot malware classification approach for unknown

family recognition with multi-prototype modeling, Computers & Security 106 (2021) 102273.

[12] U. Nandagopal, S. Thirumalaivelu, Classification of malware with mist and n-gram features using machine learning, International Journal of Intelligent Engineering & Systems (2021).

[13] V. Dentamaro, P. Giglio, D. Impedovo, L. Moretti, G. Pirlo, Auco resnet: an end-to-end network for covid-19 pre-screening from cough and breath, Pattern Recognition 127 (2022) 108656.

[14] G. Cicirelli, D. Impedovo, V. Dentamaro, R. Marani, G. Pirlo, T. D'Orazio, Human gait analysis in neurodegenerative diseases: a review, IEEE Journal of Biomedical and Health Informatics (2021).

[15] C. Li, J. Zheng, Api call-based malware classification using recurrent neural networks, Journal of Cyber Security and Mobility (2021) 617–640.

[16] F. Demirkıran, A. Çayır, U. Ünal, H. Dağ, An ensemble of pre-trained transformer models for imbalanced multiclass malware classification, arXiv preprint arXiv:2112.13236 (2021).

[17] L. Breiman, Random forests, Machine learning 45 (2001) 5–32.

[18] J. T. Hancock, T. M. Khoshgoftaar, Catboost for big data: an interdisciplinary review, Journal of big data 7 (2020) 1–45.

[19] C. Bentéjac, A. Csörgő, G. Martínez-Muñoz, A comparative analysis of gradient boosting algorithms, Artificial Intelligence Review 54 (2021) 1937–1967.

[20] P. Geurts, D. Ernst, L. Wehenkel, Extremely randomized trees, Machine learning 63 (2006) 3–42.

[21] S. O. Arık, T. Pfister, Tabnet: Attentive interpretable tabular learning, in: AAAI, volume 35, 2021, pp. 6679–6687.

[22] M. Schuster, K. K. Paliwal, Bidirectional recurrent neural networks, IEEE transactions on Signal Processing 45 (1997) 2673–2681.

[23] K. Cho, B. Van Merriënboer, D. Bahdanau, Y. Bengio, On the properties of neural machine translation: Encoder-decoder approaches, arXiv preprint arXiv:1409.1259 (2014).

[24] A. Oliveira, R. Sassi, Behavioral malware detection using deep graph convolutional neural networks, TechRxiv (2019).