Privacy Malware Classifiers based on Memory Dumping Analysis

David Fabián Cevallos Salas

***Abstract*—Malware oriented to infringe the privacy of users has experienced a rise due to new regulations worldwide as well as the increase of electronic commerce and online services. This type of malware is challenging to be recognized once executed compelling the use of forensic techniques such as memory dumping analysis. In this paper, five classifiers based on this forensics technique and the dataset CIC-MalMem-2022 from the Canadian Institute for Cybersecurity of the University of New Brunswick (CIC-UNB) are presented. The implemented classifiers were: a benign/malicious binary classifier, a multiclass benign plus three malware categories (spyware, ransomware and trojan horse) classifier; and a multiclass benign plus 15 specific malware families classifier. A SMOTE oversampling technique has also been applied in the multiclass classifiers in order to compare results. The metrics obtained through the proposed deep neural network with the oversampling SMOTE technique allowed to reach practical thresholds for its use in Yara rules, advanced malware protection and antivirus systems.**

***Index Terms*—privacy, malware, classifier, memory dumping, spyware, ransomware, trojan horse**

I. Introduction

PRIVACY in the cyberworld is a people's inherent right that must be protected through both legal and technological efforts. In this context, new regulations worldwide from the *General Data Protection Reglament* (GDPR) in Europe to the *Organic Law on Personal Data Protection* (LOPDP which stands for *"Ley Orgánica de Protección de Datos Personales"*) in Ecuador seek to protect the acquisition, storage and processing of personal data [1].

Nevertheless, these legal efforts have been severely diminished through specialized malware focused on violating the privacy of its victims, either through direct attacks against the users themselves or against the agents authorized to treat their personal data [2].

As [3] describes, there are currently three main types of malware that seek to violate the privacy of users: spyware, ransomware and trojan horses.

Spyware uses techniques such as snooping and eavesdropping to collect data, and it is the type of malware that most threatens the privacy of users [4]. Ransomware, in the context of privacy, has a unique feature that allows it to open command and control communications to an attacker's work station in order to exfiltrate data prior to encrypting it [5]. Finally, trojan horse malware is able of installing backdoors that allow it to snoop and steal data while deceiving the user by pretending to be a valid application [6].

David Fabián Cevallos Salas, Member of Telecommunications and Information Networks Research Group - Electrical Faculty - Escuela Politécnica Nacional, Quito, Ecuador, e-mail: david.cevallos03@epn.edu.ec

On the other hand, cyber attacks based on these three categories of malware have potentially increased due to the rise in electronic commerce and in the provision of online services [7].

Although each category of malware has its own working mechanisms and architecture, they all share the common feature that once executed by a legitimate operating system process they are likely impossible to be identified by security controls until the malware has totally or partially finished its target [8].

In fact, existing malware detection techniques are mainly based on static or dynamic methods prior to its execution, without having a clear technique for identifying and understanding the malware's patterns and its behavior at runtime [3] [9].

As Figure 1 exposes, once a system has been affected by malware, taking a memory dumping and analyzing it can reveal certain patterns and behaviors that the system processes experienced in order to build a classifier. For instance, the number of handlers opened by the operative system on request of the process, the number of opened sockets for communication to remote sites; and the number of mutex and semaphores used are attributes that can help to identify certain malware categories and even its family subclassification. The contributions presented by [10] and [11] shows a complete guide for this procedure.

However, deducing a general rule followed by the different malware categories and families is not feasible with the traditional programming methods.

For that reason, in this paper are presented five privacy malware classifiers that based on a memory dumping analysis and deep neuronal networks allows to recognize these patterns

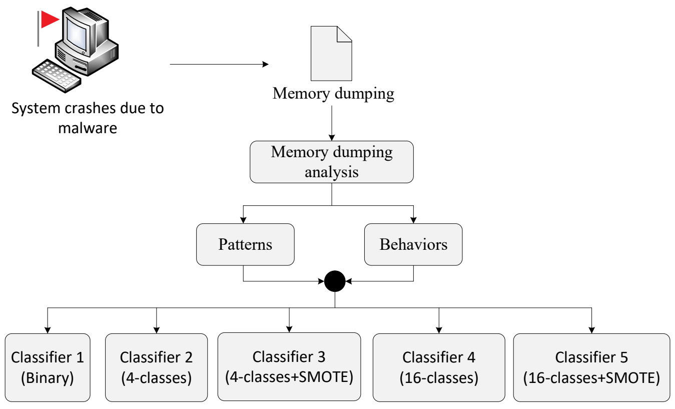

System crashes due to malware
Memory dumping
Memory dumping analysis
Patterns
Behaviors
Classifier 1 (Binary)
Classifier 2 (4-classes)
Classifier 3 (4-classes+SMOTE)
Classifier 4 (16-classes)
Classifier 5 (16-classes+SMOTE)

Fig. 1. Memory dumping analysis for privacy malware classification

and behaviors for the three privacy malware categories exposed and 15 specific families subtypes (5 foreach malware category respectively).

The classifiers have been built taking as baseline the Malware Memory Analysis CIC-MalMem-2022 dataset from the University of New Brunswick (CIC-UNB) (see section III, subsection A).

The first classifier is of binary type and allows to distinguish whether a process presents general patterns of a benign or malicious infection. The second classifier is multiclass with four categories: benign, spyware, ransomware, and trojan horse. The fourth classifier is multiclass with 16 categories: bening and 5 specific families of spyware, ransomware and trojan horse respectively.

Due to the dataset has unbalanced observations for the multiclass classifiers scenarios, a SMOTE oversampling has been applied at each one leading to two more classifiers (the third and the fifth).

This paper is organized in the following way. In section II is presented the related work that has been made by several researchers. In section III is explained the methodology followed in this research. In particular, the CIC-MalMem-2022 dataset and the materials and tools that have been used are presented in detail. In section IV are described the results and analysis of the research, and a discussion is carried out in section V. Finally the research conclusion is exposed in section VI.

II. Related Work

Since privacy malware classification is a relevant research area, prior relevant work and contributions have been made through building classifiers using forensic techniques that helps to understand malware patterns at runtime.

For instance, a classifier using Adversarial Machine Learning (AML) in a Long Short Term Memory (LSTM) model is presented by [12]. Although this contribution allows to reach a high accuracy metric it is just limited to traditional patterns and dictionary-based attacks.

A similar approach with structural variations in order to improve the classifier's time of performance has been made by [13] applying a Bidirectional Long Short-Term Memory (BiLSTM) method over three different models.

The contribution presented in [14] is able to identify malware that steals user credentials in just 2,7 minutes, using a Strange Behavior Inspection (SBI) machine learning model based on a memory dumping analysis restricted to just long-term spyware techniques.

A Heterogeneous Deep Neuronal Network (HDNN) proposed by [15] recognize malicious domain names systems (DNS) found out in volatile memory that leads to malware able to active botnets and trojan horses to perpetrate Distributed Denial of Services attacks (DDoS) and stealthy attacks respectively. The contribution is limited to just DNS names but achieves high accuracy rates for these two types of malware.

A similar work has been done by [16], although with an approach based on classical machine learning algorithms applied to a cloud-based services environment, building a system able to identify and classify trojan horses attacks.

The research carried out by [17] achieves excellent results but the RWE (Running Window Entropy) dataset does not include to spyware, the major malware type against privacy.

In the same way, the contribution of [18] showcase a self learning deep neuronal network able to extract features and patterns, but mainly from traffic seen through pcap files, that could be serve as a baseline for a memory dumping analysis approach.

Although not with a focus on data privacy but on the availability of information, the work presented in [19] demonstrates the good results that can be obtained applying deep learning techniques to analyze data from a memory dumping. So, the idea could be extended to another scenarios.

In [20] can be found out an example of this fact limited to identify and classify Cryptomining malware and special types of ransomware for this malware that could be applied within a privacy context taking into consideration the command and control sequences features able to achieve certain families of ransomware.

Finally, it is important to mention that there are relevant contributions not just based on machine learning techniques. For example, in [21] is analyzed a solution based on chip multiprocessors in order to validate memory consistency and identify malware if an inconsistency is found out.

In the same way, the work carried out by [22] presents a solution that, although it demands the use of parallel computing and multicore architectures, is able of analyzing a memory dump that feed a VCD analysis tool in order to find out malware patterns at runtime. However, the performance achieved seems not to be comparable to those contributions using machine learning and deep learning techniques.

III. Methodology

A. Dataset

The dataset used in this research was the Malware Memory Analysis CIC-MalMem-2022 created by the Canadian Institute for Cybersecurity of the University of New Brunswick (CIC-UNB). Released in April 2022, this dataset is available for downloading in [23] after registration and a complete dataset description has been made by [3] and [8].

The dataset contains a total of 58.596 balanced observations with 29.298 benign processes and 29.298 malicious processes. Having balanced benign and malicious classes makes the task of binary classification easier.

Each malicious observation belongs to one of the three categories of malware (spyware, ransomware or trojan horse) distributed in the following way: 10.020 observations belong to spyware, 9.791 observations are ransomware samples and 9.487 observations belong to trojan horse malware.

The dataset also discriminates between malware families for each malware category, as is detailed in Table I. The malware families taken into consideration by the dataset are those that exclusively threaten the privacy of users.

In total, the dataset comprises 55 features plus two columns (the first and last columns) for labels as is explained in Table II. The first column of the dataset (Category) describes whether the observation is benign or, in case of malware, the family

TABLE I
MALWARE CLASSIFICATION BY CIC-MALMEM-2022 [8]

Malware Category	Malware Familiy	Observations
Spyware	180Solutions	2.000
	Coolwebsearch	2.000
	Gator	2.200
	Transponder	2.410
	TIBS	1.410
Ransomware	Conti	1.988
	Maze	1.958
	Pysa	1.717
	Ako	2.000
	Shade	2.128
Trojan Horse	Zeus	1.950
	Emotet	1.967
	Refroso	2.000
	Scar	2.000
	Reconyc	1.570
Total		**29.298**

to which the observation belongs thus allowing to distinguish at the same time if the observation belongs to a spyware, ransomware or trojan horse. The last dataset column (Class) is redundant, and indicates in a general way whether the observation belongs to a benign or malicious process.

B. Methods

This research was carried out following a methodology in stages as is described in Figure 2. Each stage achieved results that served as inputs for the next.

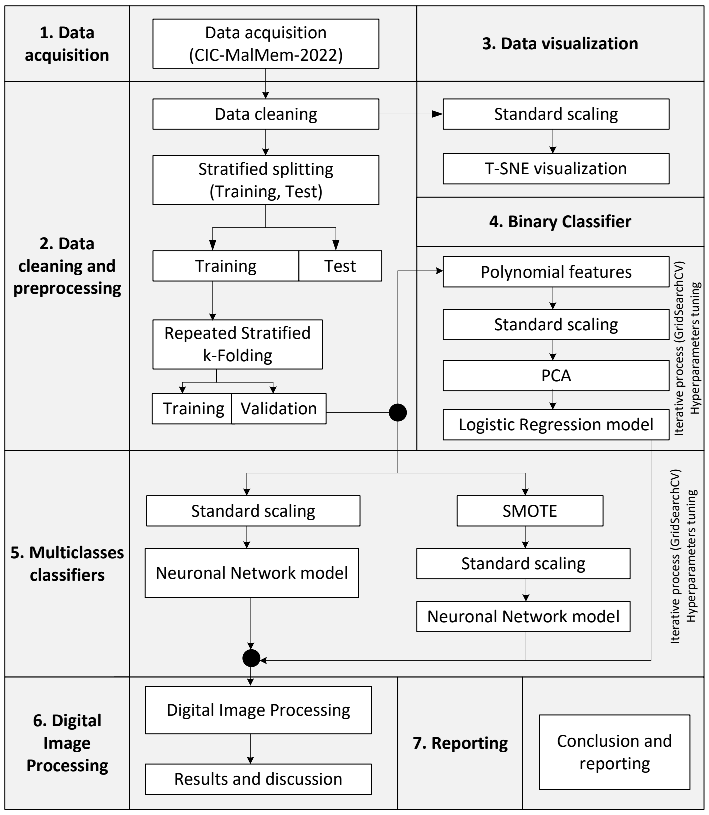


Fig. 2. Methodology

In the first stage, the search and acquisition of the CIC-MalMem-2022 dataset that significantly covers the patterns and behaviors of the three types of privacy malware was carried out.

From the dataset acquired, a data cleaning process was carried out in the second stage where, mainly, the different labels were correctly assigned to each of the dataset classes.

This process allowed to identify and discern between the observations corresponding to benign and malicious processes, as well as among the malicious processes which correspond to the three categories of malware and within each category to which specific family it belongs, achieving a dataset with three more columns for general malware categorization.

In general, 80% of the total data was taken for training and cross validation and the rest 20% for test. This data splitting was made in a stratified way among categories. On the other hand, a repeated k-fold stratified cross validation strategy was carried out over the 80% of data using 2 repetitions and 10 folds (20 iterations in total).

In stage three a data visualization was carried out using the t-SNE technique with an previous phase of standard scaling over all the features. Using a perplexity value of 100, it was possible to start to visualize classes for the binary case, the 4-classes classifiers and the 16-classes classifiers.

Thus, in stage four the binary classifier was built with the following considerations:

- Due to the dataset is balanced for the binary case, a logistic regression model was built.
- Using the sklearn tool GridSearchCV, were taken as hyperparameters of the Logistic Regression model the polynomial degree (1 or 2), the type of regularization (L1 or L2) and the value of the regularization coefficient λ with values equal to 1×10^{-3}, 1×10^{-2}, 1×10^{-1}, 1 and 100 yielding a total of 20 models evaluated for hyperparameter tunning.
- At each fold, a concrete instance of each model with its hyperparameters values was created. Also, a standard scaling and PCA dimensionality reduction was carried out on the training data of their respective fold in order to fit the model created.
- The saga solver with a maximum of 200 iterations was used for building the model.
- A general score metric corresponding to the average recall score obtained on the cross validation data through the different folds and iterations foreach model was then gathered.
- The model with the hyperparameters that yield to the best average score was taken as final result and refit in order to evaluate the data in test.
- The entire process was implemented through sklearn pipelines that facilitated the execution of each of the steps of the explained methodology for the binary classifier.

In stage five, the multiclass classifiers were built. In this stage were implemented four classifiers: two corresponding to the 4-classes classifiers and two corresponding to the 16-classes classifiers, that is four multiclass classifiers in total.

Since the dataset is no longer balanced in these scenarios unlike the binary case, a SMOTE oversampling strategy on

TABLE II
CIC-MalMem-2022 features and labels [8]

Column	Label/Feature	Description
1	Category	Category
2	pslist.nproc	Total number of processes
3	pslist.nppid	Average number of threads for the processes Total
4	pslist.avg_threads	Average number of 64 bit processes
5	pslist.nprocs64bit	Total number of 64 bit processes
6	pslist.avg_handlers	Average number of handlers
7	dllist.ndlls	Total number of loaded libraries for every process
8	dllist.avg_dlls_per_proc	Average number of loaded libraries per process
9	handles.nhandles	Total number of opened handles
10	handles.avg_handles_per_proc	Average number of handles per process
11	handles.nport	Total number of port handles
12	handles.nfile	Total number of file handles
13	handles.nevent	Total number of event handles
14	handles.ndesktop	Total number of desktop handles
15	handles.nkey	Total number of key handles
16	handles.nthread	Total number of thread handles
17	handles.ndirectory	Total number of directory handles
18	handles.nsemaphore	Total number of semaphore handles
19	handles.ntimer	Total number of timer handles
20	handles.nsection	Total number of section handles
21	handles.nmutant	Total number of mutant handles
22	ldrmodules.not_in_load	Total number of modules missing from the load list
23	ldrmodules.not_in_init	Total number of modules missing from the init list
24	ldrmodules.not_in_mem	Total number of modules missing from the memory list
25	ldrmodules.not_in_load_avg	Average amount of modules missing from the load list
26	ldrmodules.not_in_init_avg	Average amount of modules missing from the init list
27	ldrmodules.not_in_mem_avg	Average amount of modules missing from the memory
28	malfind.ninjections	Total number of hidden code injections
29	malfind.commitCharge	Total number of Commit Charges
30	malfind.protection	Total number of protection
31	malfind.uniqueInjection	Total number of unique injections
32	psxview.not_in_pslist	Total number of processes not found in the pslist
33	psxview.not_in_eprocess_pool	Total number of processes not found in the psscan
34	psxview.not_in_ethread_pool	Total number of processes not found in the thrdproc
35	psxview.not_in_pspcid_list	Total number of processes not found in the pspcid
36	psxview.not_in_csrss_handles	Total number of processes not found in the csrss
37	psxview. not_in_session	Total number of processes not found in the session
38	psxview. not_in_deskthrd	Total number of processes not found in the desktrd
39	psxview.not_in_pslist_false_avg	Average false ratio of the process list
40	psxview.not_in_eprocess_pool_false_avg	Average false ratio of the process scan
41	psxview.not_in_ethread_pool_false_avg	Average false ratio of the third process
42	psxview.not_in_pspcid_list_false_avg	Average false ratio of the process id
43	psxview.not_in_csrss_handles_false_avg	Average false ratio of the csrss
44	psxview.not_in_session_false_avg	Average false ratio of the session
45	psxview.not_in_deskthrd_false_avg	Average false ratio of the deskthrd
46	modules.nmodules	Total number of modules
47	svcscan.nservices	Total number of services
48	svcscan.kernel_drivers	Total number of kernel drivers
49	svcscan.fs_drivers	Total number of file system drivers
50	svcscan.process_services	Total number of Windows 32 owned processes
51	svcscan.shared_process_services	Total number of Windows 32 shared processes
52	svcscan.interactive_process_services	Total number of interactive service processes
53	svcscan.nactive	Total number of actively running service processes
54	callbacks.ncallbacks	Total number of callbacks
55	callbacks.nanonymous	Total number of unknown processes
56	callbacks.ngeneric	Total number of generic processes
57	Class	Benign or Malware

the training dataset was considered for analyzing its impact on each type of classification.

The multiclass classifiers were built with the following common considerations:

- Due to the complexity of the problem and its non-linearity, a deep neural network was implemented for each classifier.
- The number of hidden layers equal to 1, 3 and 5 and the value of the regularization coefficient equal to 1×10^{-6}, 1×10^{-3} and 1×10^{-1} were taken as hyperparameters of each model. Thus, 9 different models were evaluated for hyperparameters tuning and model selection.
- The general data input was normalized prior to feeding the deep neural network.

- At each hidden layer, the non-linear activation function Leaky-ReLu was used with a factor of 0,1. This allowed avoiding the problem of dead neurons during the training phase.
- After the linear response of each layer and before its Leaky-ReLu non-linear activation function, a Batch Normalization layer was added, which allowed accelerating the learning process.
- After the Leaky-ReLu non-linear activation function a Dropout layer with a factor equal to 0,01 was added for helping to avoid overfitting and avoid memorization of the neural network.
- In the output layer, the softmax activation function was used, which allowed obtaining a vector of probabilities for each category analyzed.
- The Adam optimizer was used for these models due to it showed, in general, a better performance than other optimizers such as SGD, Adagrad, Adadelta, AdamW, among others. The sparse categorical cross entropy was taken as loss criteria for analyzing each step.
- A learning rate equal to 1×10^{-3} was used due to this rate demonstrated to decrease loss.
- Each hidden layer was built with 2.048 neurons and a total of 10 epochs with a mini-batch size of 32 were used for training.
- L2 regularization was also used in order to help to avoid overfitting.
- At each iteration and fold a concrete instance of each model was created and fit on the training dataset. One scenario did not considered the application of SMOTE on the training dataset, whereas the another did it.
- At each fold, a concrete instance of each model with its hyperparameters values was created. The model was training with the training dataset and then evaluated using the cross validation dataset for hyperparameter tuning.
- A general score metric corresponding to the average micro-accuracy score obtained on the cross validation data through the different folds and iterations for each model was then gathered.
- The model with the hyperparameters that yield to the best average score was taken as final result and refit in order to evaluate the data in test.
- The entire process was implemented through sklearn and imblearn pipelines that facilitated the execution of each of the steps of the explained methodology for the multiclass classifiers.

Then, all the models with their respective hyperparameters tuned and already trained were used to be evaluated on the test dataset.

In stage six, the images recovered from the memory dumping analysis were analyzed through the digital image process that is described in Figure 3. This process is oriented to sharpen the images' details using mainly the Sobel gradient and Laplacian components.

The digital image process proposed in this research was carried out in five steps:

- Step 1: Reading the recovered original image (point a).

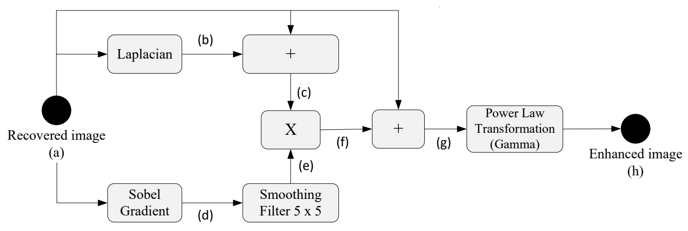


Fig. 3. Image Enhancing Process

- Step 2: Obtaining the Laplacian (point b) and adding to the original image to enhance fine aspects (sharpening) (point c).
- Step 3: Obtaining the Sobel gradient (point d) and smoothing it with a 5x5 kernel (point e).
- Step 4: Obtaining a mask (point f) as a product of (point c) and (point e) and adding it to the original image (point g)
- Step 5: Grayscale intensity transformation using the Power Law (Gamma) Transformation technique with parameters c equal to 1 and gamma equal to 0,5. The resulting image (point h) was recorded on disk.

The final research results were then gathered and discussed.

The final stage seven involved to reach the final conclusion and reporting results.

C. Performance Metrics

The average recall was used as the metric for hyperparameter tuning for the binary classifier. The malicious class has been taken as positive class for calculating the metrics.

On the other hand, the average micro-accuracy was taken as reference for the multiclass classifiers. The average final metrics acquired after applying the mean to the scores resulting of the evaluation in the cross validation dataset at each fold has been called cross validation average score in this research.

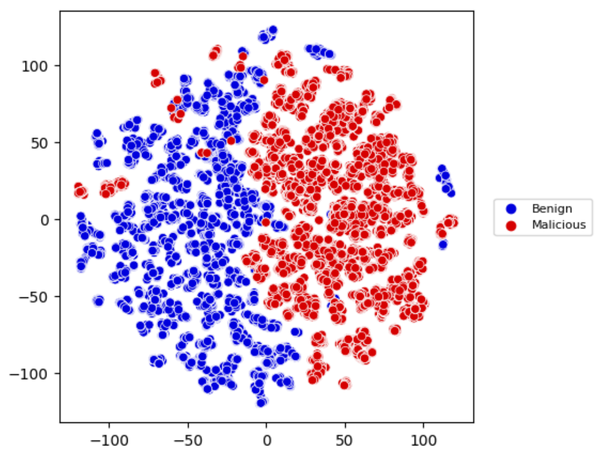
For testing evaluation, recall and accuracy metrics achieved, together with its matrix confusion and the average Area under the Curve (AUC) score of the malicious class (positive class) have been taken as reference for the binary classifier.

For multiclass classifiers the weighted values of precision, recall and F1-score have been used for testing. Also, it has been analyzed the confusion matrix and the average AUC achieved. Receiver operating characteristic (ROC) curves for each class have been also generated and analyzed.

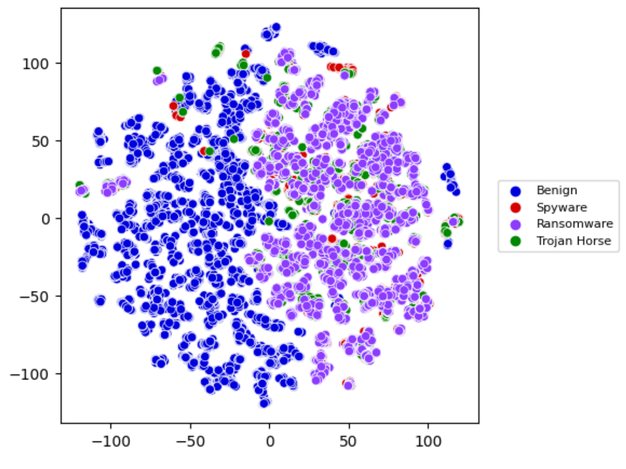
D. Materials and Tools

This research was carried out using the Google Colaboratory (Google Colab) platform and the Python programming language. The Sci-kit Learn: Machine Learning in Python framework (best known as sklearn) [24] was used to clean and preprocess data as well as to built the different machine learning models and evaluate them.

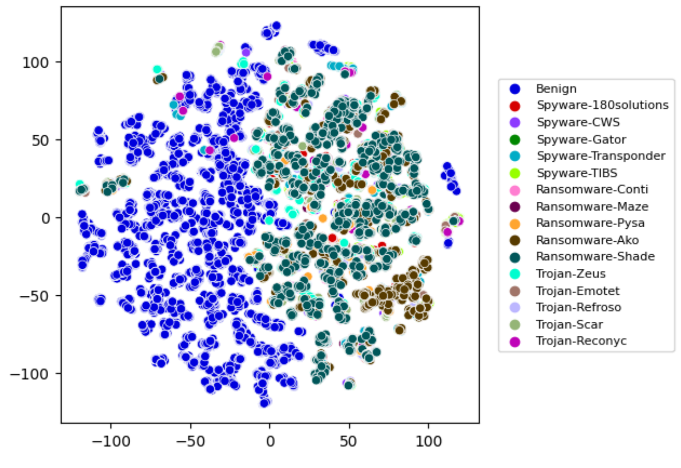
Google TensorFlow and its API Keras [25] were used for the building of the deep neuronal networks, which were embedded in a sklearn pipelines using the technique described in [26].



(a)



(b)



(c)

Fig. 4. t-SNE data visualization: (a) Binary case (b) Malware categories (c) Malware Families

Also, the deep neural networks of this research were implemented using the Keras Wrapper solution for Python [27], which allowed the models achieved with TensorFlow to be adapted and integrated into data structures supported by sklearn.

The information recovery through memory dumping was made using the tool Volatility, available at [28].

For the digital image processing the software Matlab version 2023a was used.

IV. Results and Analysis

A. t-SNE Data Visualization

t-SNE Data visualization with dimension reduction to 2 components demonstrates that the CIC-MalMem-2022 dataset is able of clearly distinguishing between benign and malicious samples as Figure 4a suggests.

However, when trying to discriminate between different categories of malware, it is not feasible to find out such a clear border as in the binary case, as is illustrated in Figure 4b.

This demonstrates the difficulty that currently exists in distinguishing between different categories of malware. As suggested by [29], many ransomwares families have similar patterns to trojan horses and several behaviours that resemble spyware attacks.

Similarly, Figure 4c suggests that the classification of different malware families is even a more difficult task due to the fact that each family share common properties with others, more likely if they are of the same malware category.

B. Binary Classifier

The hyperparameters values that allowed obtaining the best logistic regression model among the 20 possible binary classifiers analyzed were a regularization parameter equal to 1×10^{-3} with L2 regularization and polynomial degree equal to 1.

Therefore, the results suggest that it is not necessary to generate polynomial features to obtain a high metric for the binary classification with the CIC-MalMem-2022.

These hyperparameters allowed to reach a cross validation score of 0,9984 as is detailed in Table III.

Figure 5 presents the confusion matrix achieved for the binary classifier's performance evaluation in the testing dataset. As can be seen, 26 observations are classified as false positive

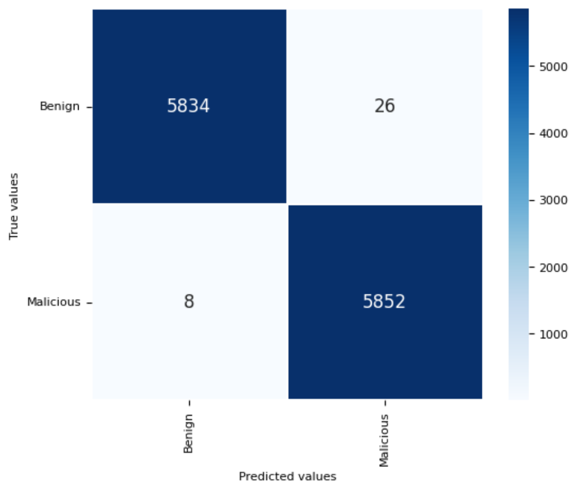


Fig. 5. Binary classifier's test confusion matrix

TABLE III
CROSS VALIDATION BEST SCORES ACHIEVED BY EACH CLASSIFIER

Classifier	Cross validation average score
Binary	0,9984
4-classes	0,7538
4-classes+SMOTE	0,7686
16-classes	0,6202
16-classes+SMOTE	0,6830

whereas just 8 observations are classified as false negative cases, which demonstrates that the binary classifier built is able to reduce the false negative cases in order to achieve a high recall metric.

This confusion matrix reached an accuracy of 0,9971 and an average AUC metric of 0,9970.

C. Multiclass Classifiers

For all the multiclass classifiers, it was obtained that the values of the best hyperparameters were a number of hidden layers equal to 3 and a value of the regularization coefficient equal to 1×10^{-6}.

These values of hyperparameters allowed to achieve a cross validation average score of 0,7538 and 0,7686 for the 4-classes classifiers without and with SMOTE respectively; as well as a metric of 0,6202 and 0,6802 for the 16-classes classifiers without and with SMOTE respectively, as is summarized in Table III.

Table IV summarizes the resulting metrics obtained on the test dataset for the multiclass classifiers.

Figures 6a and 6b illustrates the confusion matrix obtained for the 4-classes classifier without and with the oversampling SMOTE technique applied in the test dataset respectively.

The results show that the SMOTE technique helped to improve the true positive rate of the trojan horse malware category penalizing, however, the rate of the other categories of malware.

The 4-classes classifier without SMOTE reached for testing a weighted precision of 0,7814, weighted recall of 0,7433 and a weighted F1-score of 0,7325. The 4-classes classifier with SMOTE reached for testing a weighted precision of 0,7868, weighted recall of 0,7611 and a weighted F1-score of 0,7544.

It is important to note that the built neural networks try to obtain the best F1-score metric while keeping the recall and precision values almost at the same level.

On the other hand, the 4-classes classifier reached an average AUC metric equal to 0,9160 without SMOTE, whereas an average AUC metric equal to 0,9230 was reached with SMOTE.

Figures 7a and 7b exposes the ROC curves achieved for the 4-classes classifier without and with SMOTE in the test dataset respectively. The figure suggests that the AUC metric for each privacy malware category rises when SMOTE is applied mainly for the ransomware and trojan Horse categories.

As can be seen, the SMOTE technique allowed to slightly increase all the metrics for the case of the 4-class classifier.

Although the increase in metric values was not substantial for this scenario, SMOTE was able to generate ROC curves much more arched towards the upper left of the graph, achieving a greater AUC for each malware category which is a desired feature for every malware classifier.

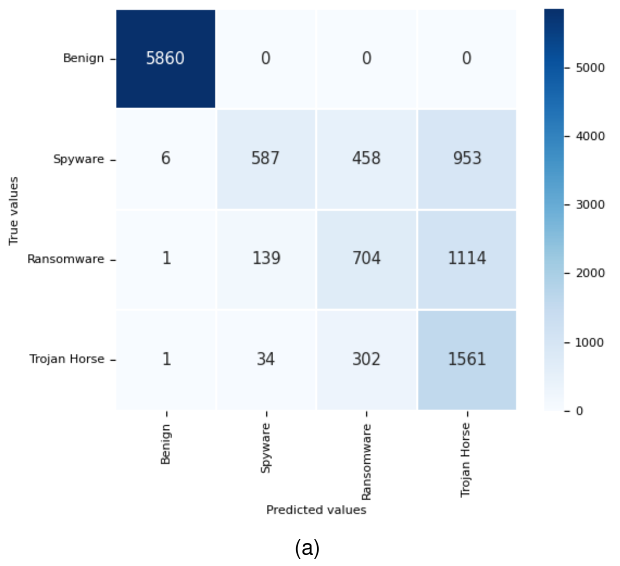
Figures 8a and 8b exposes the confusion matrix for the 16-classes classifiers with and without SMOTE in the test dataset respectively. As the mentioned figures suggest, the SMOTE technique allows to increase the number of true positive observations for certain families such as the Spyware-CWS and Trojan-Reconyc, at the cost of reducing these values for other classes, that is, the SMOTE oversampling technique tries to achieve more balanced and equitable metrics for the different classes under analysis.

The 16-classes classifier without SMOTE achieved a weighted precision of 0,6788, weighted recall of 0,6220 and a weighted F1-score of 0,6208; whereas the 16-classes classifier with SMOTE achieved a weighted precision of 0,7115, weighted recall of 0,6789 and a weighted F1-score of 0,6815.

This shows that the SMOTE technique was able to substantially increase the value of the metrics obtained, mainly the F1-score.

TABLE IV
MULTICLASS CLASSIFIERS'S TEST RESULTING METRICS

Classifier	Weighted-Precision	Weighted-Recall	Weighted-F1-score	Average-AUC
4-classes	0,7814	0,7433	0,7325	0,9160
4-classes+SMOTE	0,7868	0,7611	0,7544	0,9230
16-classes	0,6788	0,6220	0,6208	0,8981
16-classes+SMOTE	0,7115	0,6789	0,6815	0,9344



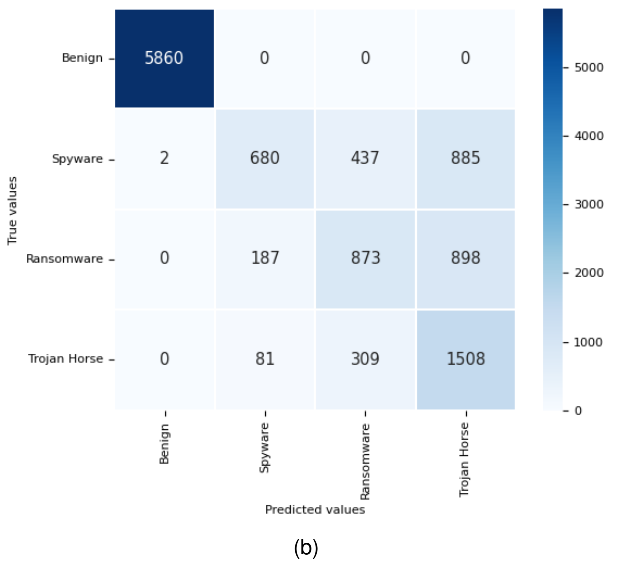
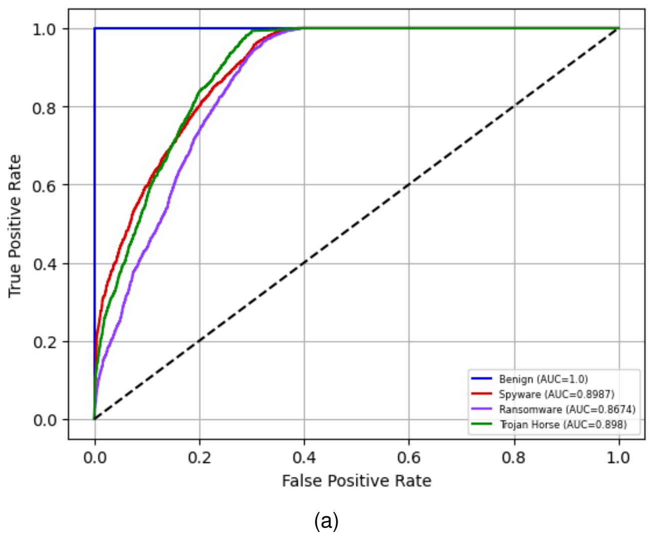


Fig. 6. 4-classes classifier's test confusion matrix (a) Without SMOTE (b) With SMOTE



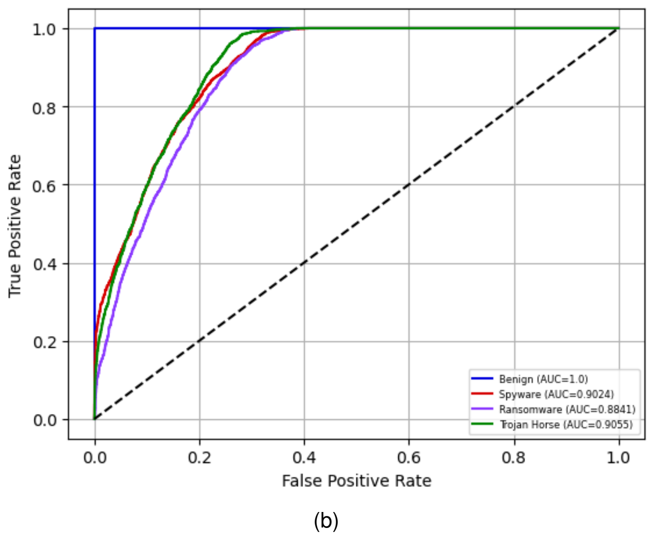


Fig. 7. 4-classes classifier's test ROC curves (a) Without SMOTE (b) With SMOTE

Finally, Figures 9a and 9b illustrates the ROC curves achieved for the 16-classes classifiers with and without SMOTE in the test dataset respectively.

In this case, the average AUC achieved for the 16-classes classifier in the test dataset without SMOTE was equal to 0,8981 whereas the AUC metric using SMOTE was equal to 0,9344.

Again, the SMOTE technique was able to substantially increase the AUC metric, allowing to reach almost the same AUC to each class.

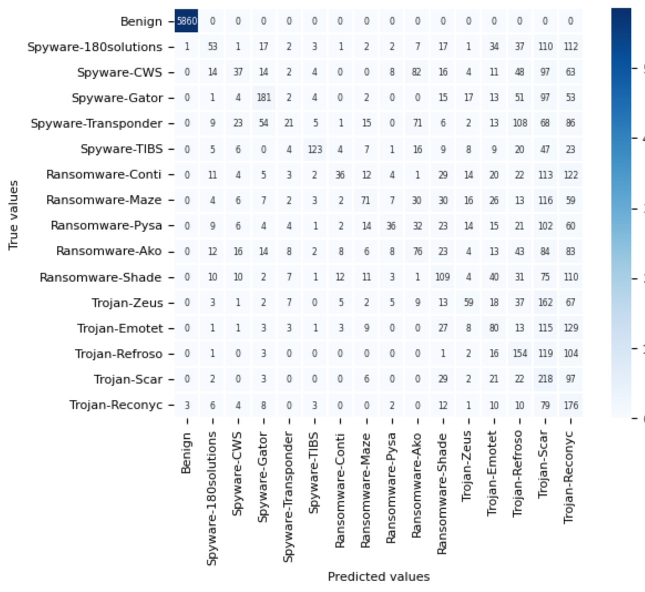
The ROC curves without SMOTE, although achieve good AUC values, appear scattered among them. When SMOTE is applied, the ROC curves bow even more towards the upper left corner of the graph, which demonstrates that the SMOTE oversampling technique allows to improve the AUC metric especially of those classes with fewer observations.

As can be seen, the SMOTE technique allowed obtaining good results, improving the values obtained, especially in the case of 16 classes, at the cost of a higher computational cost due to the synthetic generation of data.

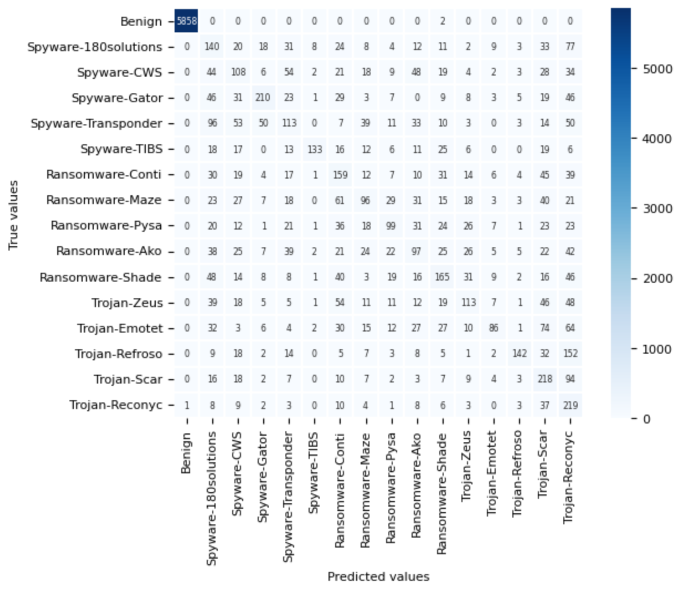
D. Digital Image Processing

The RAM memory dumping allowed the recovery of various types of files such as .exe executables, word processor documents and mainly images.

Although the images recovered using Volatility correspond to a best effort, the image processing technique suggested in the present investigation allowed to improve the details of the recovered images. This can be used, for example, to obtain details such as moles, tattoos, scars or other indications for forensic analysis in the case of crimes.

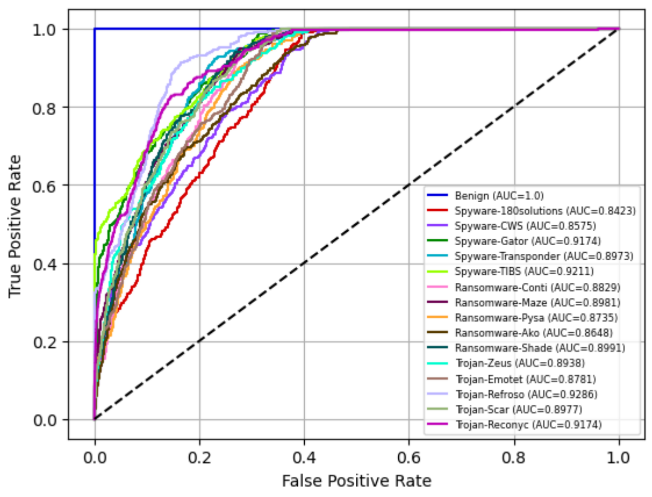


(a)

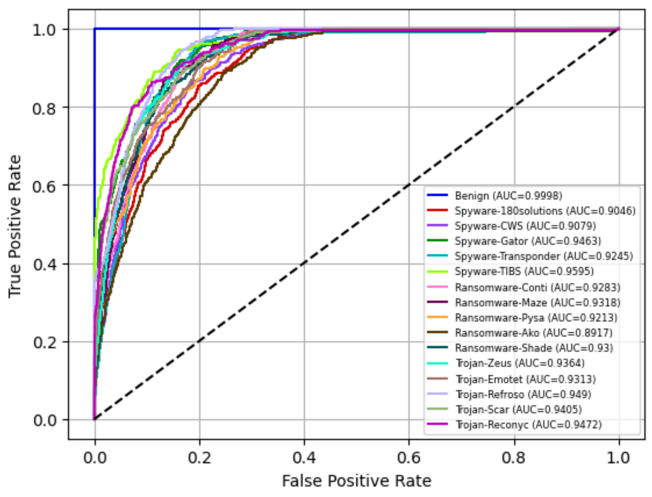


(b)

Fig. 8. 16-classes classifier's test confusion matrix (a) Without SMOTE (b) With SMOTE



(a)



(b)

Fig. 9. 16-classes classifier's test ROC curves (a) Without SMOTE (b) With SMOTE

The sequence of Figures from 10a to 10l presents some examples of the original images obtained with Volatility (first column) and the improved images after applying the image sharpening process (second column).

As can be seen, the digital process that has been given to each of the images allows them to be softened, making it possible to improve the details of the faces.

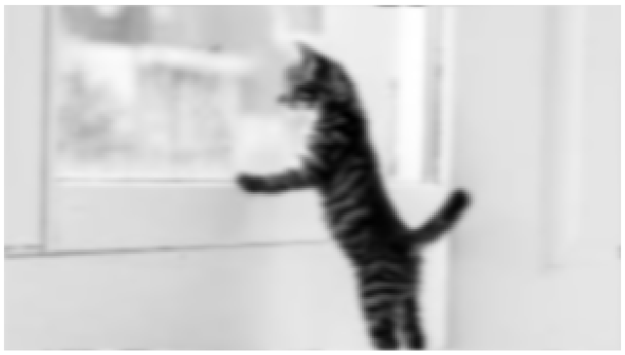
V. Discussion

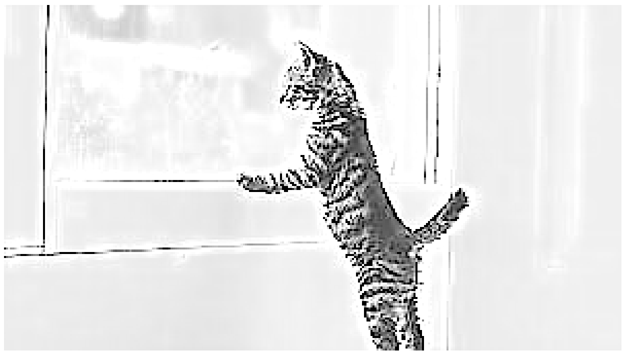
Privacy malware identification and classification at runtime continue to be a widely exploited field of research since distinguishing among the different patterns and behaviors of malware with benign processes and even more among different categories and families of malware is a challenge.

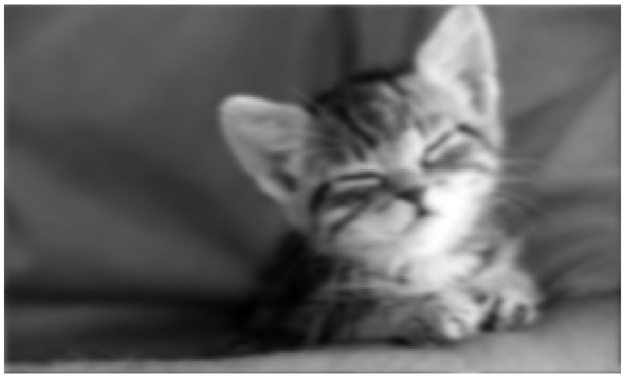
The five classifiers presented in this research allowed to achieve important results to determine how effective the CIC-MalMem-2022 dataset is in this task and compare them with other researchers.

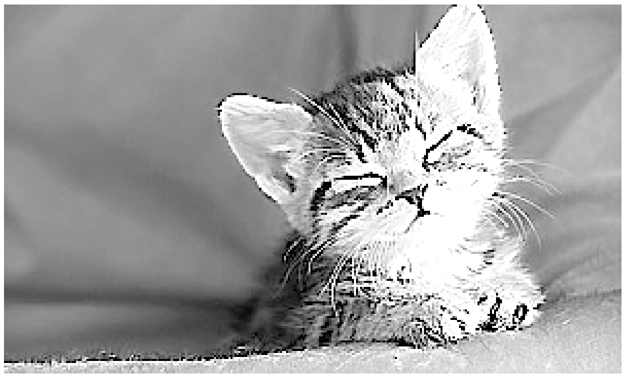
As of the writing date of this document, the researches proposed by [3] and [8] are the only ones that make a relevant analysis of the binary classification using the CIC-MalMem-2022 dataset.

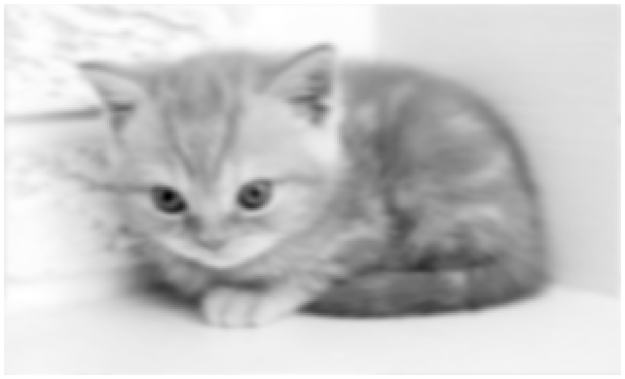
Overall, the best accuracy level of 0.9700 achieved by [3] using a random forest classifier is lower than the 0.9971 presented in this research, achieved through logistic regression with determined preprocessing techniques. In the same way, logistic regression is also applied by [8] reaching exactly the same accuracy that [3].

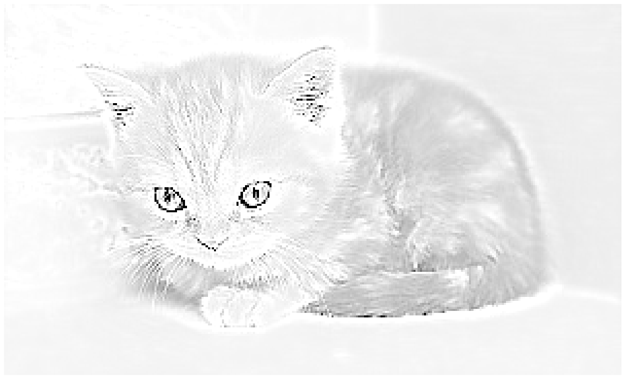

(a)

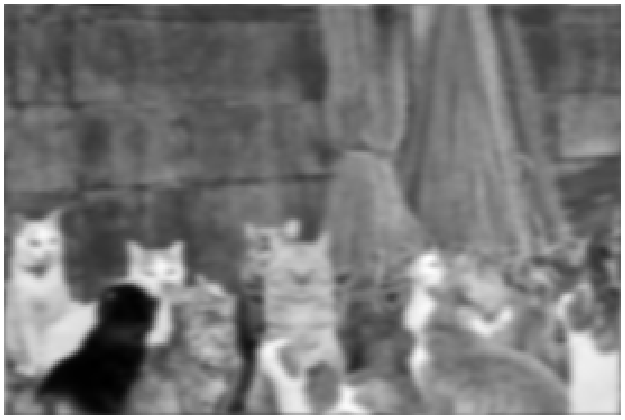

(b)


(c)


(d)


(e)


(f)


(g)

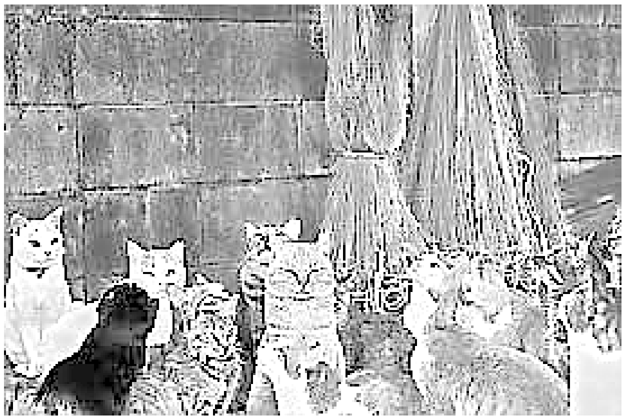

(h)

Fig. 10. Recovered (first column) and enhanced images (second column) from memory dumping

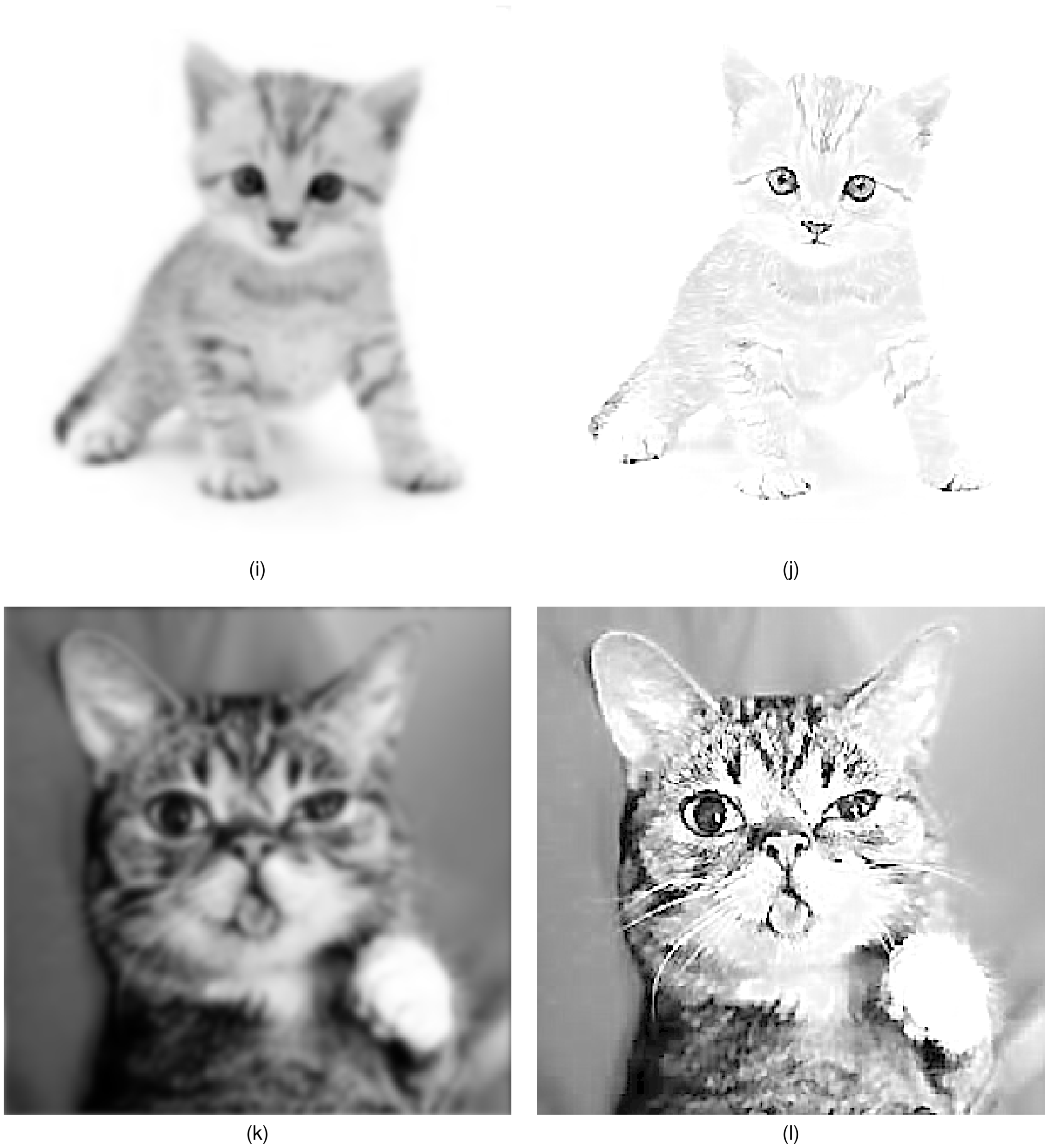


Fig. 10. Recovered and enhanced images

Even, the overall accuracy value achieved in this research is greater than the achieved by [15] equal to 0,9773 without using parallel techniques, although this last uses the SDGA domain names dataset.

This demonstrates that the data preprocessing techniques suggested in this research can help to obtain better results in privacy malware binary classification.

On the other hand, the SMOTE oversampling technique applied in this research for building the 4-classes and 16-classes classifiers has been able to improve the results obtained in this task.

With regards to 4-classes privacy malware categories, the SMOTE technique helped to increase the AUC metric for each malware category, specially the spyware and ransomware categories. However, with respect to the case where SMOTED is not applied, there is no an important rise in the testing metric values achieved between privacy malware categories.

In general, the testing F1-score achieved in this research (0,7544 using SMOTE) is slightly less than that achieved by [30] (0,7690) although their research is oriented to threats spread through URLs and/or email and subject to just three categories.

This was not the case for the 16-classes classifier where SMOTED helped to reach better metrics values including to AUC values. In this case, the metrics reached were even better than those achieved by [31], who carried out a pretty similar research although restricted to malware identified by malicious domain name systems achieving for different scenarios metrics with testing weighted F1-score between 0,5906 and 0,622 in comparison to the value of 0,6815 achieved in this research.

VI. Conclusion

Privacy malware identification and classification is a complex problem that depends in large extent on the quality of data to be solved. In this paper, it has been demonstrated that the CIC-MalMem-2022 dataset which employs a pattern and behavior's volatile memory analysis is able to clearly distinguish between benign and malicious samples (mainly due to its balanced data) and achieves reasonable results in classifying between privacy malware categories and privacy malware families.

The SMOTE technique allowed for both multiclass classifiers to achieve a weighted F1-score inside the threshold available for its use in Yara rules as well as advanced malware protection solutions and antivirus (metric greater than 0,65).

However, the dataset information must still be worked in order to achieve better models able to detect and classify malware at runtime.

Thus, for example, it can be concluded that adding new features capable of distinguishing more clearly between ransomwares and trojan horses could help to considerably improve the metrics for multiclass classifiers.

References

[1] V. Morales and A. Robalino-Lopez, "Framework for the Evaluation of Internet Development. Case Study: Application of Internet Universality Indicators in Ecuador," *2020 7th International Conference on eDemocracy and eGovernment, ICEDEG 2020*, pp. 291–296, 2020.

[2] A. N. Jahromi, S. Hashemi, A. Dehghantanha, R. M. Parizi, and K. K. R. Choo, "An Enhanced Stacked LSTM Method with No Random Initialization for Malware Threat Hunting in Safety and Time-Critical Systems," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 4, no. 5, pp. 630–640, 2020.

[3] T. Carrier, P. Victor, A. Tekeoglu, and A. Lashkari, "Detecting Obfuscated Malware using Memory Feature Engineering," *Proceedings ofthe 8th International Conference on Information Systems Security and Privacy (ICISSP 2022)*, no. Icissp, pp. 177–188, 2022.

[4] H. Huseynov, K. Kourai, T. Saadawi, and O. Igbe, "Virtual Machine Introspection for Anomaly-Based Keylogger Detection," *IEEE International Conference on High Performance Switching and Routing, HPSR*, vol. 2020-May, 2020.

[5] S. Homayoun, A. Dehghantanha, M. Ahmadzadeh, S. Hashemi, and R. Khayami, "Know Abnormal, Find Evil: Frequent Pattern Mining for Ransomware Threat Hunting and Intelligence," *IEEE Transactions on Emerging Topics in Computing*, vol. 8, no. 2, pp. 341–351, 2020.

[6] S. Shukla, G. Kolhe, P. D. Sai Manoj, and S. Rafatirad, "Stealthy malware detection using RNN-Based automated localized feature extraction and classifier," *Proceedings - International Conference on Tools with Artificial Intelligence, ICTAI*, vol. 2019-November, pp. 590–597, 2019.

[7] Y. Lee, S. Woo, Y. Song, J. Lee, and D. H. Lee, "Practical Vulnerability-Information-Sharing Architecture for Automotive Security-Risk Analysis," *IEEE Access*, vol. 8, pp. 120 009–120 018, 2020.

[8] M. Dener, G. Ok, and A. Orman, "Malware Detection Using Memory Analysis Data in Big Data Environment," *Applied Sciences (Switzerland)*, vol. 12, no. 17, 2022.

[9] C. W. Chen, C. H. Su, K. W. Lee, and P. H. Bair, "Malware Family Classification using Active Learning by Learning," *International Conference on Advanced Communication Technology, ICACT*, vol. 2020, pp. 590–595, 2020.

[10] A. H. Lashkari, B. Li, T. L. Carrier, and G. Kaur, "VolMemLyzer: Volatile Memory Analyzer for Malware Classification using Feature Engineering," *2021 Reconciling Data Analytics, Automation, Privacy, and Security: A Big Data Challenge, RDAAPS 2021*, no. Cic, 2021.

[11] D. Mu, Y. Du, J. Xu, J. Xu, X. Xing, B. Mao, and P. Liu, "POMP++: Facilitating postmortem program diagnosis with value-set analysis," *IEEE Transactions on Software Engineering*, vol. 47, no. 9, pp. 1929–1942, 2021.

[12] I. Yilmaz, A. Siraj, and D. Ulybyshev, "Improving DGA-Based malicious domain classifiers for malware defense with adversarial machine learning," *4th IEEE Conference on Information and Communication Technology, CICT 2020*, 2020.

[13] Girinoto, H. Setiawan, P. A. W. Putro, and Y. R. Pramadi, "Comparison of LSTM Architecture for Malware Classification," *Proceedings - 2nd International Conference on Informatics, Multimedia, Cyber, and Information System, ICIMCIS 2020*, pp. 93–97, 2020.

[14] N. Mohamed and B. Belaton, "SBI Model for the Detection of Advanced Persistent Threat Based on Strange Behavior of Using Credential Dumping Technique," *IEEE Access*, vol. 9, pp. 42 919–42 932, 2021.

[15] L. Yang, G. Liu, Y. Dai, J. Wang, and J. Zhai, "Detecting stealthy domain generation algorithms using heterogeneous deep neural network framework," *IEEE Access*, vol. 8, pp. 82 876–82 889, 2020.

[16] P. Mishra, P. Aggarwal, A. Vidyarthi, P. Singh, B. Khan, H. H. Alhelou, and P. Siano, "VMShield: Memory Introspection-Based Malware Detection to Secure Cloud-Based Services against Stealthy Attacks," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 10, pp. 6754–6764, 2021.

[17] K. J. Jones and Y. Wang, "Malgazer: An Automated Malware Classifier with Running Window Entropy and Machine Learning," *2020 6th International Conference on Mobile and Secure Services, MOBISECSERV 2020*, pp. 1–6, 2020.

[18] J. Yang and Y. Guo, "AEFETA: Encrypted traffic classification framework based on self-learning of feature," *2021 IEEE 6th International Conference on Intelligent Computing and Signal Processing, ICSP 2021*, no. Icsp, pp. 876–880, 2021.

[19] J. Zhang, M. Kwon, S. Han, N. S. Kim, M. Kandemir, and M. Jung, "FastDrain: Removing Page Victimization Overheads in NVMe Storage Stack," *IEEE Computer Architecture Letters*, vol. 19, no. 2, pp. 92–96, 2020.

[20] G. Mani, V. Pasumarti, B. Bhargava, F. T. Vora, J. Macdonald, J. King, and J. Kobes, "DeCrypto Pro: Deep learning based cryptomining malware detection using performance counters," *Proceedings - 2020 IEEE International Conference on Autonomic Computing and Self-Organizing Systems, ACSOS 2020*, pp. 109–118, 2020.

[21] B. Kumar, S. Thakur, K. Basu, M. Fujita, and V. Singh, "A low overhead methodology for validating memory consistency models in chip multiprocessors," *Proceedings - 33rd International Conference on VLSI Design, VLSID 2020 - Held concurrently with 19th International Conference on Embedded Systems*, pp. 101–106, 2020.

[22] D. Appello, P. Bernardi, A. Calabrese, S. Littardi, G. Pollaccia, S. Quer, V. Tancorre, and R. Ugioli, "Accelerated Analysis of Simulation Dumps through Parallelization on Multicore Architectures," *Proceedings - 2021 24th International Symposium on Design and Diagnostics of Electronic Circuits and Systems, DDECS 2021*, pp. 69–74, 2021.

[23] Canadian Institute for Cybersecurity, "Malware memory analysis cic-malmem-2022," 2022, https://www.unb.ca/cic/datasets/malmem-2022.html, Last accessed on 2023-07-02.

[24] Scikit Learn, "scikit-learn machine learning in python," 2023, https://scikit-learn.org/stable/, Last accessed on 2023-07-28.

[25] Google, "Google tensorflow," 2023, https://www.tensorflow.org/?hl=es-419, Last accessed on 2023-07-28.

[26] Jason Brownlee , "Use keras deep learning models with scikit-learn in python," 2023, https://machinelearningmastery.com/use-keras-deep-learning-models-scikit-learn-python/, Last accessed on 2023-07-28.

[27] Jason Brownlee, "Use keras deep learning models with scikit-learn in python," 2023, https://machinelearningmastery.com/use-keras-deep-learning-models-scikit-learn-python/, Last accessed on 2023-07-28.

[28] Volatitlity, "Volatitlity foundation," 2023, https://github.com/volatilityfoundation/volatility, Last accessed on 2023-07-28.

[29] A. O. Almashhadani, M. Kaiiali, S. Sezer, and P. O'Kane, "A Multi-Classifier Network-Based Crypto Ransomware Detection System: A Case Study of Locky Ransomware," *IEEE Access*, vol. 7, no. c, pp. 47 053–47 067, 2019.

[30] R. Vinayakumar, K. P. Soman, P. Poornachandran, S. Akarsh, and M. Elhoseny, *Deep Learning Framework for Cyber Threat Situational Awareness Based on Email and URL Data Analysis*. Cham: Springer International Publishing, 2019, pp. 87–124. [Online]. Available: https://doi.org/10.1007/978-3-030-16837-7_6

[31] R. Vinayakumar, K. P. Soman, P. P., S. Akarsh, and M. Elhoseny, *Improved DGA Domain Names Detection and Categorization Using Deep Learning Architectures with Classical Machine Learning Algorithms*. Cham: Springer International Publishing, 2019, pp. 161–192. [Online]. Available: https://doi.org/10.1007/978-3-030-16837-7_8

David F. Cevallos Salas was born on March 30th 1989 in Quito-Ecuador. He received his bachelor degree in Electronics and Information Networks from the Escuela Politécnica Nacional with Summa cum laude distinction in 2014. In 2020 he earned his Master of Science degree in Information Systems with a mention in Information Security Management from the Universidad UTE. He has also earned several certifications in the field of networking. As a computer programmer he has experience working for private and public institutions in Ecuador. His main research areas are Security, Distributed Computing, Software Defined Networking, Routing, Digital Television and Cloud Computing.