Chunk-based Malware Classifiers for Communication Networks

David F. Cevallos Salas *Member, IEEE*

***Abstract*—Cybercriminals have developed several types of malwares in order to carry out attacks against organizations and production systems. Chunk-based is the deadliest malware due to its ability to spread smoothly from one node to another through a communication network. Spyware, distributed ransomware and trojan horses are the most relevant examples of chunk-based malware. In this paper, the CIC-MalMem-2022 dataset is used in order to build classifiers using several machine learning algorithms to analyze its capability to find out chunk-based malware. 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. Also, an internal validation using unsupervised learning algorithms is presented.**

***Index Terms*—Malware, classifiers, spyware, ransomware, trojan horse, deep neuronal networks, SMOTE**

I. Introduction

MALWARE has affected communications networks since its conception [1]. Nowadays, several types of malware exist with different features, patterns and advanced technologies that continue to improve in time. However, as [2] states, chunk-based is currently considered the most important malware due to the potential loss of information and financial detriment caused to individuals, organizations, governments and social movements.

Chunk-based is a type of malware able to spread through a communications network transmitting small pieces of information until being reconstructed in the target. This feature makes it feasible not to be detected by advanced malware protection systems and antivirus solutions. Although several examples exist, spyware, distributed ransomware and trojan horses continue to be the most relevant chunk-based malware until now.

Spyware is a special type of malware which is able to steal confidential information once it has been activated.

Distributed ransomware is a type of malware with two main features: This malware encrypts user information and is able to spread through a communication network [3]. At least these two characteristics must be fulfilled by malware to be treated under this category.

As explained in [4] if malware is able to encrypt user information with a harmful purpose and not to spread, it can be considered as a ransomware although not a distributed ransomware.

The ability to encrypt allows malware to make information unreadable to its owner and claim a ransom for it [5] [6]. The payment is usually demanded in Bitcoins or another type of cryptocurrency in order to hide cybercriminals' identity from justice [7] [8].

On the other hand, the techniques applied to encrypt user information is different among the large number of ransomware families existing. The degree of affectation and loss of information will depend on the technique used by the ransomware family to encrypt information [9]. Symmetric, asymmetric and hybrid cryptography techniques are widely used [2].

However, in general terms, the last ransomware strains usually make use of the RSA (*Rivest*, *Shamir*, *Adleman*) encryption techniques with key lengths between 2048 and 4096 bits. This technique is based on the concept of asymmetric encryption and uses two keys: a public key to encrypt user information and a private key to decrypt it [10].

The second main feature of distributed ransomware corresponds to its ability to spread from one node to another through a communication network.

Generally speaking, if malware is able to spread through a communication network it is known as a worm [11]. Therefore, it can be said that distributed ransomware is a special type of worm, but not every worm can be considered as a distributed ransomware if it is not able to encrypt user information [12].

On the other hand, a third feature corresponds to the distributed ransomware's ability to hide itself in another medium in order to deceive its victim and be executed on the node to be compromised [13] [14]. This type of malware is usually known as a trojan horse [8].

This makes pretty difficult the task to identify exactly which malware is a distributed ransomware and which is exactly a trojan horse.

A few years ago, distributed ransomware analysis techniques were limited to observing its behavior once a communication network node had already been compromised [15]. Since then, advances in chunk-based malware detection and analysis tools have been of great importance [16].

In this paper, the CIC-MalMem-2022 dataset is analyzed using classifiers based on several machine learning algorithms in order to measure their ability to distinguish among a benign class and 15 chunk-based malware families distributed among the malware categories of spyware, ransomware and trojan horse.

The rest of the paper is distributed in the following way. In Section II is presented the related work carried out about this tipoc. In Section III a technical background is exposed in order to understand how chunk-based malware works. In Section IV the research methodology is illustrated. Section

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

V details the results obtained and its analysis. Finally, the research conclusion is exposed in Section VI.

II. Related Work

Since chunk-based 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 [17]. 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 [18] applying a Bidirectional Long Short-Term Memory (BiLSTM) method over three different models.

The contribution presented in [19] 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 [20] 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 [21], 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 [22] 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 [23] 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 [24] 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 [25] 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 [26] 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 [27] 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. Background

In this section a brief technical background is provided in order to understand how chunk-based malware operates. In order to exemplify this concept, TeslaCrypt will be taken as example.

Developed by expert cyber attackers, TeslaCrypt distributed ransomwares were detected for the first time in February 2015 and work with a similar approach to CryptoWall and WannaCry (other known types of distributed ransomwares) encrypting information on the node affected [28] [29].

Structurally, distributed ransomware is made up of several DLLs (*Dynamic Link Libraries*), each of which has malicious source code that, when executed, fulfills a specific function [30].

The main idea is that distributed ransomware splits its contain in several DLLs to communicate them to other nodes avoiding being detected in the process and trying to rebuild itself once it is hosted in the target node. This technique is sometimes known as *chunks hiding* [31].

Figure 1 depicts the DDLs propagation mechanism based on *chunks hiding*.

Once the distributed ransomware affects a new node in a communication network, it will be restructured from the *chunks* transmitted by the previously affected node that reported the DLLs, and executed in order to encrypt the user information in the new host affected and for trying to spread transmitting the malicious DLLs to its neighbors [32].

While certain DLLs are responsible for encrypting user information, others has as objective to open a connection socket with the neighboring node for establishing a communication

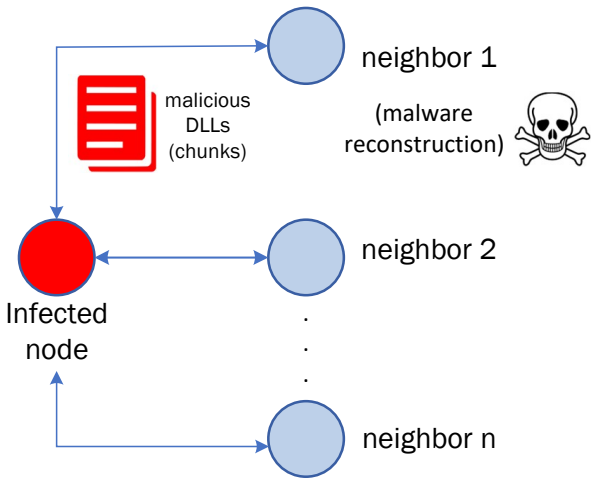


Fig. 1. DLLs distributed ransomware spread mechanism

[33]. Regardless of this, and within the scope of this research, distributed ransomware will be considered as successful in spreading from one node to another only when all of its DLLs have been successfully communicated to the neighboring node.

Therefore, the DLLs will be transmitted again from this node to another target node.

Once this happens, again, the union of the segments resulting from the execution of each one of the DLLs will lead to the restructuring of the malware in the target node. At this point, a final execution stage will start the distributed ransomware to encrypt the user information and try to replicate the DLLs to the next node [34].

A side effect of a distributed ransomware infection is an information exfiltration attack through command and control sequences [35]. An exfiltration attack comprises the steal of information in a remote way [36].

A command and control attack involves the cyber attacker executing code (commands) on the infected node in order to steal information and communicate it to the control station [37].

Although not all distributed ransomwares are capable of allowing the attacker to execute command and control sequences on an infected communication network, this is a feature of the most sophisticated distributed ransomwares and the ones that cause the most damage [38].

A similar approach is carry out by other types of chunk-based malware, mainly spyare and trojan horses with their corresponding differences. In fact, as said before, many distributed ransomware families share also common patterns with trojan horses.

However, a malicious process infected by a chunk-based malware will present certain patterns or characteristics that could allow its identification and classification when analyzed through machine learning algorithms.

IV. 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 [39] after registration and a complete dataset description has been made by [40] and [41].

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 are based on the chunk transmission mechanism.

In total, the dataset comprises 55 non-correlated features plus two columns (the first and last columns) for labels. The first column of the dataset (Category) describes whether the observation is benign or, in case of malware, the family to which the observation belongs thus allowing to distinguish at the same time if the observation is a spyware, a ransomware or a 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.

TABLE I
MALWARE CLASSIFICATION BY CIC-MALMEM-2022 [41]

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**

B. Methods

In order to analyze the CIC-MalMem-2022 through several machine learning algorithms, the methodology described in Figure 2 has been carefully followed. The methodology comprises a series of stages where each one of them serves as a basis to carry out the next one.

First and Foremost, in the first stage, the CIC-MalMem-2022 was acquired from the Canadian Institute for Cybersecurity (CIC) website.

In the second stage the data cleaning process was carried out. Mainly, each family of chunk-based malware family was labeled with a number from 0 to 15. This allowed to apply the t-SNE visualization method in order to verify the difficulty of the classification problem.

The data were then stratified splitted into training (80% of total data) and test (20%)

Using a stratified k-Fold cross validation with $k = 10$ folds, a 10-iterations enumerator was built over the training data yielding, in stage 3, foreach of the 10 iterations to define an specific model for being trained with the specific iteration's training data and evaluated with the corresponding iteration' cross-validation data.

The metrics gathered to evaluate each model with the cross-validation data were the macro: precision, recall, F1-score, accuracy, error, and area under the curve (AUC).

Thus, a mean cross-validation metric was calculated foreach of the 8 metrics after finishing the 10 folds.

Then, stage 3 finished with the model evaluation on the test data following a similar approach. Again, the 8 metrics defined above were calculated.

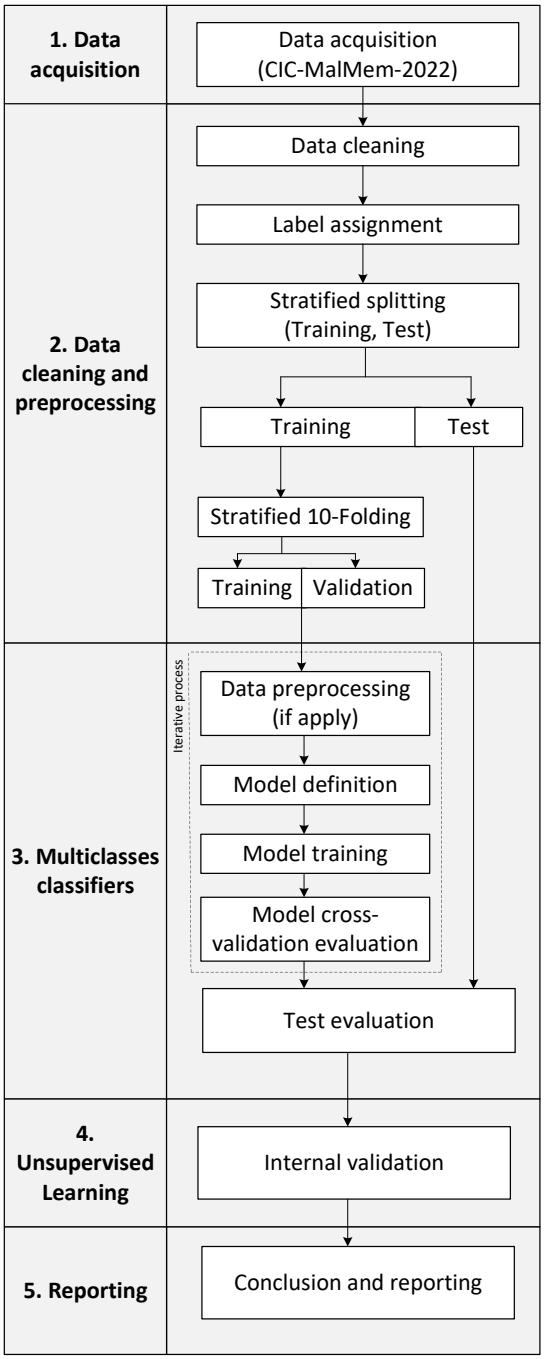


Fig. 2. Methodology

Stage 3 was performed foreach machine learning algorithm in order to determine which is able to learn the best with the available data with the CIC-MalMem-2022 dataset.

In this paper, the following machine algorithms were analyzed:

- **Support Vector Machine Classifier (SVC):** A SVM classifier with the rbf kernel and a default regularization parameter C equal to 1 and an autoscaling value of gamma was built. A z-score data normalization step was carried out as a step prior to feeding the classifier.
- **Decision Tree (DT):** A single decision tree was built using the entropy criterion and a max depth of 15.
- **Random Forest (RF):** Using a total of 130 decision trees and each one with a max depth equal to 15, a random forest solution was built in order to compare it with the singular decision tree.
- **Linear Discriminant Analysis (LDA):** A LDA classifier has been implemented in order to identify malware families using a previous step of features normalization.
- **Naive Bayes (NB):** A Naive Bayes classifier using the gaussian technique with a previous feature normalization layer has been implemented.
- **Deep Neuronal Network (DNN):** In this research, a fully connected deep neuronal network using 2.048 neurons, 10 epochs, 32 samples as mini-batch size, and L2 regularization in order avoid overfitting is proposed. The best lambda regularization hyperparameter was tested among the values $1e-6$, $1e-3$ and $1e-1$.
  Also, the best number of hidden layers hyperparameter was tested among the values 1, 3 and 5; leading to a total of 9 different models analyzed.
  Each hidden layer used the Leaky ReLu non-linear activation function with an alpha value equal to 0,1.
  The model also considered a Batch Normalization layer applied previous to the activation function and a Dropout layer with parameter equal to 0,01 after its application.
  A z-score data normalization step was carried out as a previous step to feeding the classifier.
  Finally, the Ada optimizer helped to reach the best results with the proposed architecture.
- **Adaptive Neuro Fuzzy Inference System (ANFIS):** An ANFIS networks was used to estimate its functionality classifying the malware families. For this, the TSNE dimensionality reduction was used, which served to feed the network.
  The best ANFIS model was achieved using 3 initials membership functions foreach descriptor and 100 epochs. The hybrid optimizer (gradient descent plus iterative least squares) has been employed in this classifier in order to optimize the model parameters.
- **SMOTED Deep Neuronal Network (S-DNN):** The same deep neuronal network proposal in the previous point was also tested using a SMOTE layer at the training dataset in order to verify if better metrics can be achieved using this oversampling strategy for data generation.

In stage 4 the dataset was analyzed in order to perform an internal validation with the unsupervised algorithms k-means, k-medoids, DBSCAN, Hierarchical clustering and Fuzzy c-means.

To carry out this task, all the descriptors that make up the dataset were used.

Stage 5 was the final step. In this stage the final conclusion of the research was reached and the reporting was carried out.

C. 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

TABLE II
CROSS-VALIDATION MEAN METRICS

Algorithm	Precision	Recall	F1-score	Accuracy	Error	Specificity	False Positive Rate	AUC
SVC	0,4315	0,2879	0,2908	0,6215	0,3785	0,9755	0,0245	0,8760
DT	0,4850	0,4759	0,4748	0,7200	0,2800	0,9819	0,0181	0,8855
RF	0,5443	0,5284	0,5229	0,7484	0,2516	0,9837	0,0163	0,9541
LDA	0,3325	0,2457	0,2387	0,5970	0,4030	0,9738	0,0262	0,8547
NB	0,2944	0,2006	0,1492	0,5647	0,4352	0,9718	0,0282	0,8321
DNN	0,6202	0,6202	0,6202	0,6202	0,3798	0,9867	0,0133	0,8981
ANFIS	0,2031	0,0941	0,2406	0,3115	0,6885	0,9246	0,0754	0,7078
S-DNN	0,6830	0,6830	0,6830	0,6830	0,3170	0,9921	0,0079	0,9344

framework (best known as sklearn) [42] was used to clean and preprocess data as well as to built the different machine learning models and evaluate them.

The deep neural networks of this research were implemented using the Keras Wrapper solution for Python [43], which allowed the models achieved with TensorFlow to be adapted and integrated into data structures supported by sklearn.

On the other hand, MatLab has been used for ANFIS and unsupervised learning algorithms.

V. RESULTS AND ANALYSIS

Figure 3 demonstrates that chunk-based malware family classification is a difficult task. Although the CIC-MalMem-2022 dataset is able to distinguish between benign and malicious samples, it is not able to show a clear pattern among the different malware families.

The main problem arises from the fact that the different strains of spyware, ransomware and trojan horses show common patterns that can be confused or misinterpreted which makes it difficult to classify them in a certain category.

Indeed, the dataset observations overlap by reducing the dimensionality of the problem, therefore making it difficult for the problem to be solved in this way.

Table II summarizes the cross-validation mean metrics obtained after the 10-folds iterative process. The best F1-score obtained for the training phase on the cross-validation dataset corresponds to the S-DNN classifier reaching a value of 0,6830.

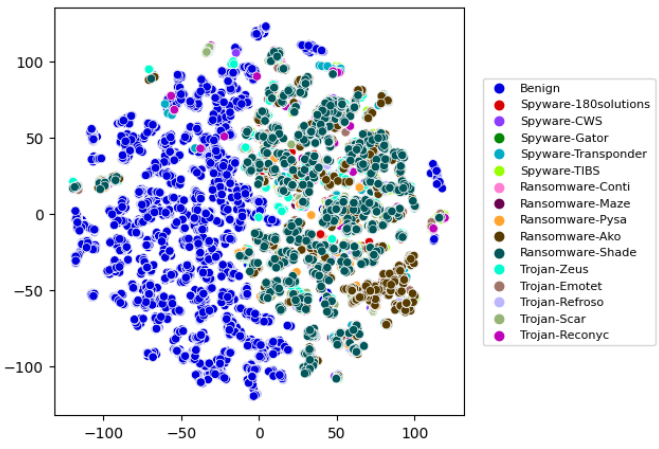


Fig. 3. t-SNE data visualization (perplexity equal to 100)

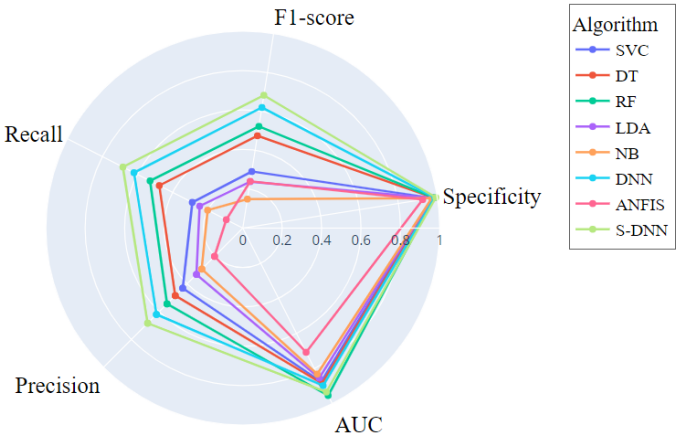


Fig. 4. Cross-validation metrics comparison

The best deep neuronal networks were achieved using 3 hidden layers and a lambda regularization parameter equal to $1e-6$, both for the case in which SMOTE was used and for the one in which SMOTE was not used.

Figure 4 depicts a comparison of the metrics achieved among the different machine learning and deep learning techniques. As the figure suggests, the best precision, recall and F1-score was achieved using the SMOTE oversampling technique together with deep neuronal networks, followed by the technique of deep learning without SMOTE and the random forest classifier.

The metric F1-score is extremely important in the present analysis scenario, since chunk-based malware families is a multiclass classification problem, this metric represents a general vision for each class.

Although the metrics achieved by the decision tree are lower than those of these three classifiers, the values achieved at the cost of the required computational cost make it a good choice.

The worst metric values were obtained in the classifiers implemented using the ANFIS, NB, SVM and LDA classification techniques, which are not suitable for malware identification and classification.

In particular, the results achieved demonstrates that fuzzy logic is not suitable in order to classify advanced malware families.

However, it is worth noting that during the cross-validation phase it managed to obtain an F1-score metric better that the achieved by the LDA classifier, even with the t-SNE dimensionality reduction applied in order to apply the ANFIS

TABLE III
TEST EVALUATION METRICS

Algorithm	Precision	Recall	F1-score	Accuracy	Error	Specificity	False Positive Rate	AUC
SVC	0,4277	0,2814	0,2845	0,6184	0,3819	0,9752	0,0248	0,8752
DT	0,4836	0,4729	0,4723	0,7189	0,2811	0,9818	0,0182	0,8856
RF	0,5429	0,5241	0,5209	0,7464	0,2536	0,9836	0,0164	0,9546
LDA	0,3251	0,2411	0,2337	0,5949	0,4051	0,9737	0,0263	0,8541
NB	0.2956	0.2016	0.1491	0.5660	0.4340	0,9719	0,0281	0.8319
DNN	0,6788	0,6220	0,6201	0,6220	0,3780	0,9853	0,0147	0,8981
ANFIS	0,1110	0,0766	0,1266	0,1918	0,8082	0,9494	0,0506	0,7077
S-DNN	0,7115	0,6788	0,6815	0,6788	0,3212	0,9901	0,0099	0,9344

neuronal network.

Although all the classifiers analyzed reach high mean AUC values, likewise the best values correspond to the three previously indicated classifiers. However, the random forest classifier achieved a better average AUC metric than deep neural networks with and without SMOTE.

Table III summarizes the metrics obtained after evaluate each model on the test dataset. Overall, the metrics achieved are similar to those achieved in the cross-validation dataset, which reflects that the built models do not fail in the generalization process. Again, the best F1-score obtained corresponds to the S-DNN classifier reaching a value of 0,6815.

Figure 5 illustrates a comparison of the metrics obtained with the test dataset among the different classifiers. The result obtained is a graph pretty similar to that of the metrics achieved for the cross-validation case, which suggests that the classifiers manage to interpret the generalization quite well.

However, in this ocasion the F1-score metric achieved by the ANFIS classifier was the worst achieved among all the classifiers analyzed.

On the other hand, the best metrics were reached by the DNN classifier and the S-DNN classifier, what is more important S-DNN was able to achieve an F1-score metric better than 0,65 which relects that this classifier can be used together with Yara rules and other industrial protection systems.

Figure 6 illustrates the confusion matrix achieved with S-DNN on the test dataset. Although some error are made by this classifier, the metrics achieved through this confusion matrix are able to reach the industrial threshold required for

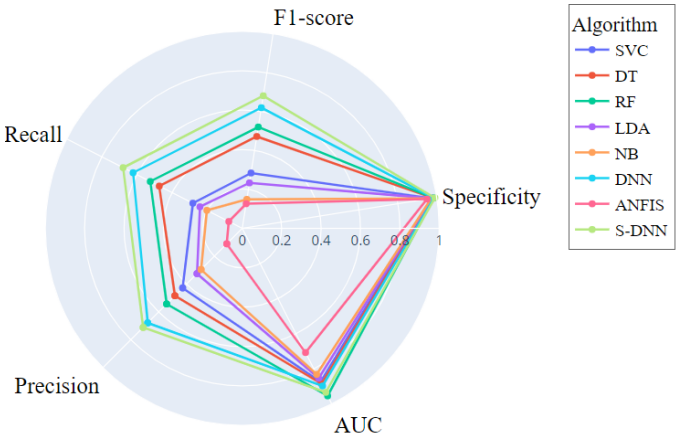


Fig. 5. Test metrics comparison

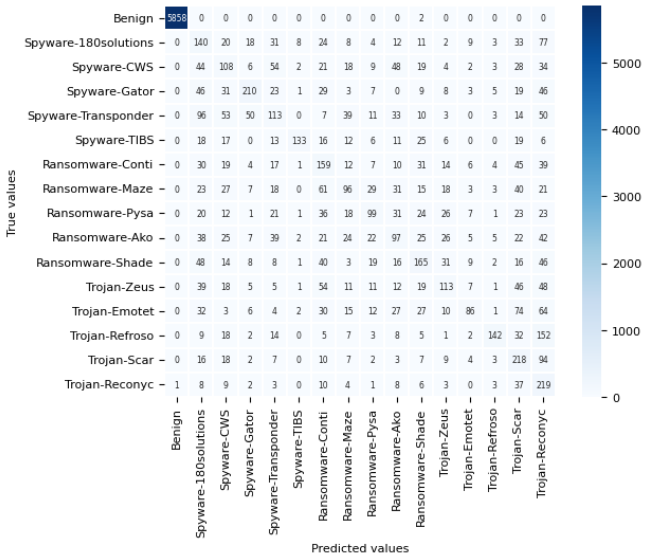


Fig. 6. Test S-DNN confusion matrix

the building of Yara rules as well as antivirus and advanced malware protection mechanisms.

Nevertheless, it is important to emphasize that the classifier also achieves an almost perfect classification for the binary case. In fact, the majority of cases with a true positive scenario have been achieved for the benign category, whereas the malware categories show a significant reduction in this metric.

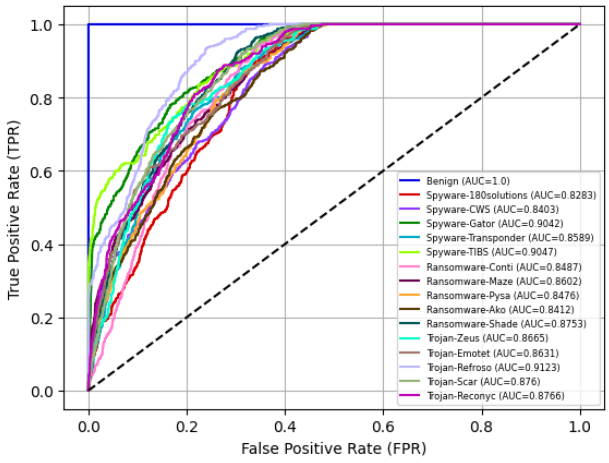
Figures 7a, 7b, 7c, 7d, 7e, 7f, 7g and 7h depicts the ROC curves achieved by the SVC, DT, RF, LDA, NB, DNN, ANFIS and S-DNN classifiers respectively on the test dataset.

Overall, all the classifiers were able to reach high AUC which demonstrates that CIC-MalMem-2022 is useful for identifying chunk-based malware families.

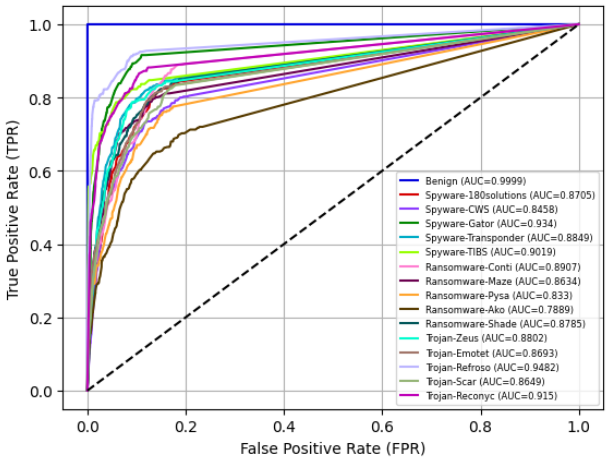
The best ROC curves were achieved through the RF classifier reaching an average AUC score equal to 0,9546 and surpassing even to the S-DNN classifier that reached an average AUC metric equal to 0,9344.

On the other hand, the classifier with the worst ROC curves achieved was ANFIS demonstrating again the limitations that the fuzzy logic have when identifying and comparing malware families.

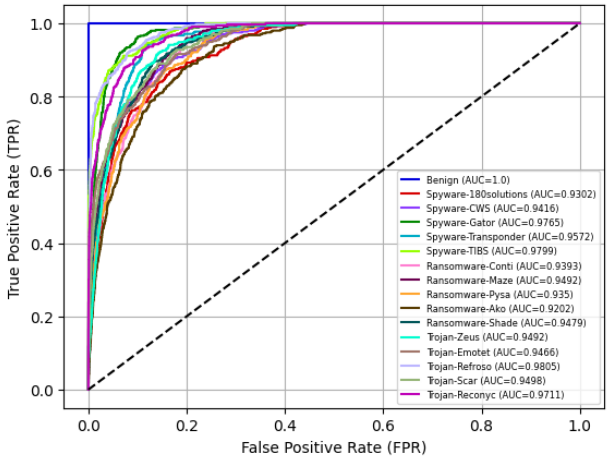
So, in general, it can be observed that classifiers based on some technique of dimensionality reduction or plane and hyperplane formation are not a good strategy for the task of identifying and classifying malware, since their application



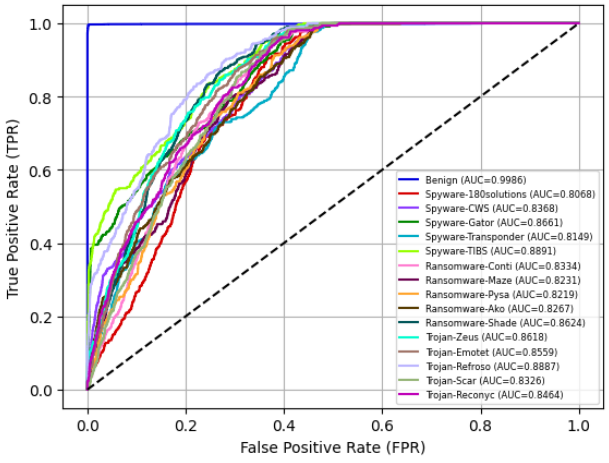
(a) SVC



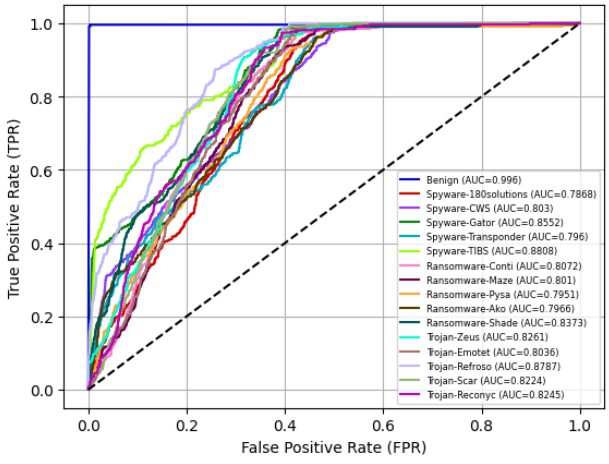
(b) DT



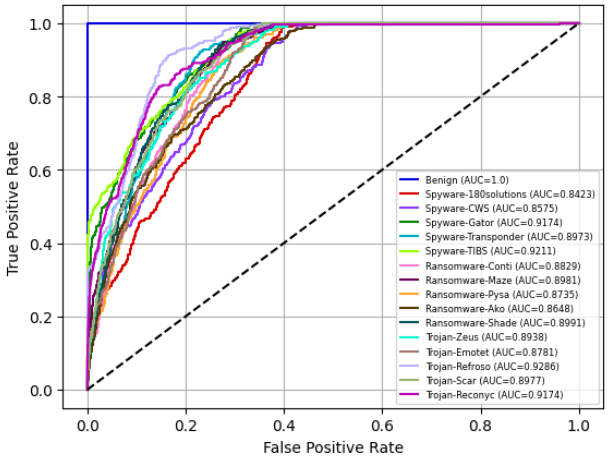
(c) RF



(d) LDA

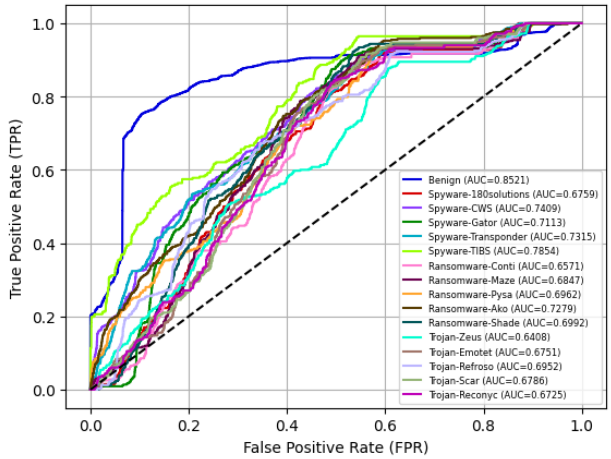


(e) NB

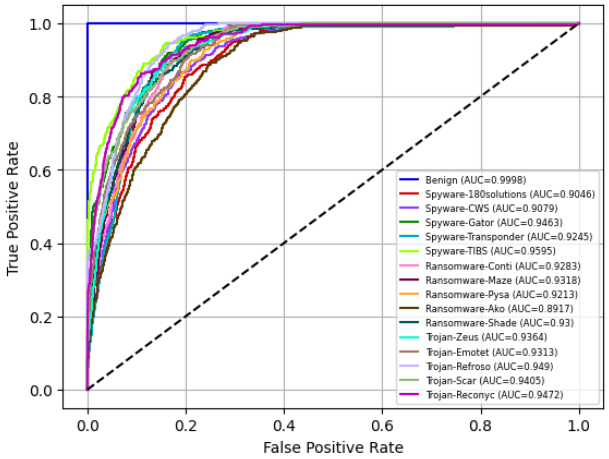


(f) DNN

Fig. 7. Test ROC curves



(g) ANFIS



(h) S-DNN

Fig. 7. Test ROC curves

involves loss of relevant information. leading then to obtaining very poor performance metrics.

Finally, the CIC-MalMem-2022 dataset was analyzed using the unsupervised learning algorithms k-means, k-medoids, DBSCAN, Hierarchical clustering and Fuzzy c-means in order to perform an internal validation of the metrics that can be obtained.

Figure 8 details the elbow diagram from 1 to 20 clusters for the CIC-MalMem-2022 with the k-means, k-medoids and Fuzzy c-means algorithms, demonstrating that a value of 16 clusters seems to be the optimal value from which the objective function decreases to its minimum value.

So, a value of 16 clusters has been taken as reference for these three algorithms as well as for the Hierarchical clustering algorithm. For the DBSCAN algorithm a value of 3 minimum neighbors with a value of epsilon equal to 2.049,10 lead to the formation of 16 clusters.

Table IV exposes the metrics obtained with the parameters explained. The metrics that have been considered are: Sum of Squared Within (SSW), Sum of Squared Between (SSB), WB index and Silhouette.

TABLE IV
UNSUPERVISED LEARNING METRICS FOR 16 CLUSTERS

Algorithm	SSW	SSB	WB-index	Silhouette
k-means	418,5362	3276,6000	2,0437	0,6815
k-medoids	420,7788	3275,8000	2,0552	0,6631
DBSCAN	297,8611	770,2884	6,1870	0,8945
Hierarchical clustering	515,3139	1097,8000	7,5107	0,8410
Fuzzy c-means	387,7675	3273,2000	1,8955	0,5233

Figure 9 compares the metrics achieved previously normalizing them to a scale between 0 and 1.

Overall, k-means and k-medoids reached the best metrics values in comparison to the rest of algorithms.

This internal validation study shows that the dataset used is robust and correctly structured, helping to confirm that the results obtained are valid.

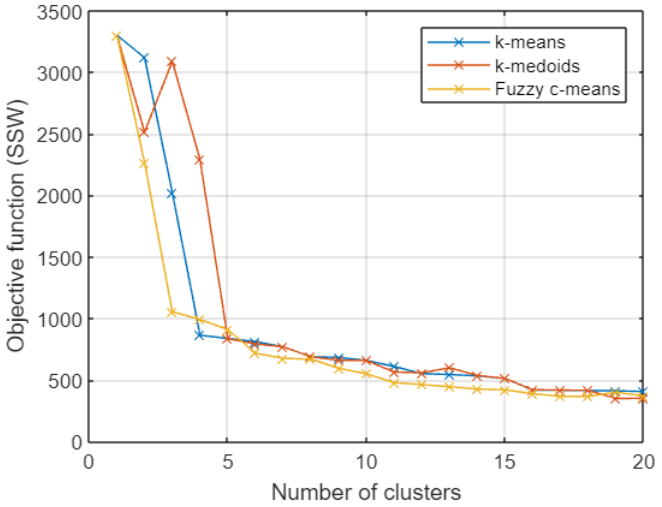


Fig. 8. CIC-MalMem-2022 Elbow diagram

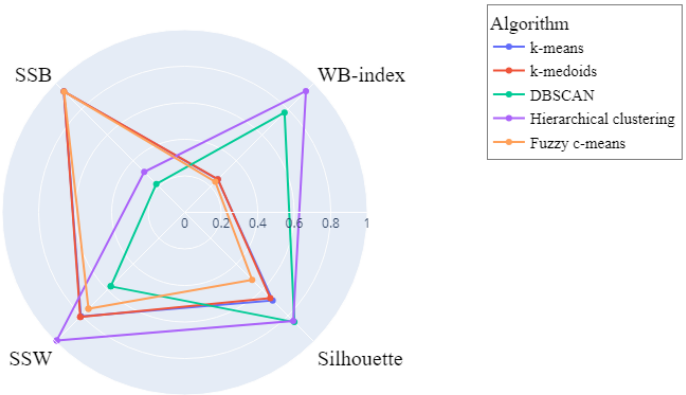


Fig. 9. Unsupervised learning algorithms' metrics comparison

VI. Conclusion

This research allows to conclude that deep neural networks comprise a much more robust technique for the identification and classification of chunk-based malware families in comparison to traditional machine learning algorithms such as SVC, decision trees, random forest, among others.

In addition to allowing a higher F1-score, deep neural networks achieved precision and recall values balanced and almost equal, which is a desirable feature in the building of Yara rules, advanced malware protection and antivirus systems.

In conjunction with an oversampling technique for the generation of synthetic samples that help to solve the problem of an unbalanced dataset, it can be concluded that deep neural networks can improve their performance metrics even in a complex problem such as chunk classification.

Finally, it is worth to mention that the CIC-MalMem-2022 proves to be an effective mean for detecting malware as was demonstrated through the internal validation using unsupervised learning algorithms, but in order to be used at an industrial level, it must improve its descriptors and the contribution that each one offers to the different models able to be built on the data.

References

[1] L. J. García Villalba, A. L. Sandoval Orozco, A. López Vivar, E. A. Armas Vega, and T. H. Kim, "Ransomware Automatic Data Acquisition Tool," *IEEE Access*, vol. 6, no. c, pp. 55 043–55 051, 2018.

[2] 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.

[3] W. Liu, "Modeling Ransomware Spreading by a Dynamic Node-Level Method," *IEEE Access*, vol. 7, pp. 142 224–142 232, 2019.

[4] E. Berrueta, D. Morato, E. Magana, and M. Izal, "A Survey on Detection Techniques for Cryptographic Ransomware," *IEEE Access*, vol. 7, pp. 144 925–144 944, 2019.

[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. Zollner, K. K. R. Choo, and N. A. Le-Khac, "An Automated Live Forensic and Postmortem Analysis Tool for Bitcoin on Windows Systems," *IEEE Access*, vol. 7, pp. 158 250–158 263, 2019.

[7] A. Adamov and A. Carlsson, "The state of ransomware. Trends and mitigation techniques," *Proceedings of 2017 IEEE East-West Design and Test Symposium, EWDTS 2017*, 2017.

[8] T. N. Witte, "Phantom malware: Conceal malicious actions from malware detection techniques by imitating user activity," *IEEE Access*, vol. 8, pp. 164 428–164 452, 2020.

[9] M. Al-Hawawreh, F. D. Hartog, and E. Sitnikova, "Targeted Ransomware: A New Cyber Threat to Edge System of Brownfield Industrial Internet of Things," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 7137–7151, 2019.

[10] D. Javaheri, M. Hosseinzadeh, and A. M. Rahmani, "Detection and elimination of spyware and ransomware by intercepting kernel-level system routines," *IEEE Access*, vol. 6, no. c, pp. 78 321–78 332, 2018.

[11] Z. Li, A. L. G. Rios, and L. Trajkovic, "Detecting Internet Worms, Ransomware, and Blackouts Using Recurrent Neural Networks," *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*, vol. 2020-October, pp. 2165–2172, 2020.

[12] J. Castiglione and D. Pavlovic, "Dynamic Distributed Secure Storage against Ransomware," *IEEE Transactions on Computational Social Systems*, vol. 7, no. 6, pp. 1469–1475, 2020.

[13] J. Chen, C. Wang, Z. Zhao, K. Chen, R. Du, and G. J. Ahn, "Uncovering the Face of Android Ransomware: Characterization and Real-Time Detection," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 5, pp. 1286–1300, 2018.

[14] I. Almomani, R. Qaddoura, M. Habib, S. Alsoghyer, A. A. Khayer, I. Aljarah, and H. Faris, "Android Ransomware Detection Based on a Hybrid Evolutionary Approach in the Context of Highly Imbalanced Data," *IEEE Access*, vol. 9, pp. 57 674–57 691, 2021.

[15] 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.

[16] B. A. S. Al-Rimy, M. A. Maarof, M. Alazab, F. Alsolami, S. Z. M. Shaid, F. A. Ghaleb, T. Al-Hadhrami, and A. M. Ali, "A Pseudo Feedback-Based Annotated TF-IDF Technique for Dynamic Crypto-Ransomware Pre-Encryption Boundary Delineation and Features Extraction," *IEEE Access*, vol. 8, pp. 140 586–140 598, 2020.

[17] 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.

[18] 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.

[19] 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.

[20] Y. Yang and H. Zhang, "Mathematical Models and Control Methods of Infectious Diseases," *Proceedings - 5th International Conference on Automation, Control and Robotics Engineering, CACRE 2020*, pp. 383–388, 2020.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] S. C. Hsiao and D. Y. Kao, "The static analysis of WannaCry ransomware," *International Conference on Advanced Communication Technology, ICACT*, vol. 2018-February, pp. 153–158, 2018.

[29] D. Y. Kao, S. C. Hsiao, and R. Tso, "Analyzing WannaCry Ransomware Considering the Weapons and Exploits," *International Conference on Advanced Communication Technology, ICACT*, vol. 2019-February, no. 1, pp. 1098–1107, 2019.

[30] B. Abrath, B. Coppens, S. Volckaert, and B. D. Sutter, "Obfuscating Windows DLLs," *Proceedings - International Workshop on Software Protection, SPRO 2015*, pp. 24–30, 2015.

[31] B. Qin, Y. Wang, and C. Ma, "API Call Based Ransomware Dynamic Detection Approach Using TextCNN," *Proceedings - 2020 International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering, ICBAIE 2020*, pp. 162–166, 2020.

[32] S. Poudyal and D. Dasgupta, "AI-Powered Ransomware Detection Framework," *2020 IEEE Symposium Series on Computational Intelligence, SSCI 2020*, pp. 1154–1161, 2020.
[33] F. Khan, C. Ncube, L. K. Ramasamy, S. Kadry, and Y. Nam, "A Digital DNA Sequencing Engine for Ransomware Detection Using Machine Learning," *IEEE Access*, vol. 8, pp. 119 710–119 719, 2020.
[34] A. Zimba, Z. Wang, and H. Chen, "Reasoning crypto ransomware infection vectors with Bayesian networks," *2017 IEEE International Conference on Intelligence and Security Informatics: Security and Big Data, ISI 2017*, pp. 149–151, 2017.
[35] M. Casenove, "Exfiltrations using polymorphic blending techniques: Analysis and countermeasures," *International Conference on Cyber Conflict, CYCON*, vol. 2015-January, pp. 217–230, 2015.
[36] A. Iacovazzi, S. Sarda, D. Frassinelli, and Y. Elovici, "DropWat: An Invisible Network Flow Watermark for Data Exfiltration Traceback," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 5, pp. 1139–1154, 2018.
[37] A. Zahra and M. A. Shah, "IoT based ransomware growth rate evaluation and detection using command and control blacklisting," *ICAC 2017 - 2017 23rd IEEE International Conference on Automation and Computing: Addressing Global Challenges through Automation and Computing*, no. September, pp. 7–8, 2017.
[38] V. Zaccaria, M. C. Molteni, F. Melzani, and G. Bertoni, "Darth's Saber: A Key Exfiltration Attack for Symmetric Ciphers Using Laser Light," *Proceedings - 2018 Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2018*, vol. 2, pp. 23–26, 2018.
[39] 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.
[40] 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.
[41] 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.
[42] Scikit Learn, "scikit-learn machine learning in python," 2023, https://scikit-learn.org/stable/, Last accessed on 2023-07-28.
[43] 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.

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.