ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

RANSOMWARE DETECTION BY COGNITIVE SECURITY

THESIS SUBMITTED AS PART OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF DOCTOR OF PHILOSOPHY IN INFORMATICS

JUAN ALBERTO HERRERA SILVA

juan.herrera@epn.edu.ec

SUPERVISOR: PHD. MYRIAM BEATRIZ HERNÁNDEZ ÁLVAREZ

myriam.hernandez@epn.edu.ec

CO-SUPERVISOR: PHD. LORENA ISABEL BARONA LÓPEZ

lorena.barona@epn.edu.ec

Quito, march 2023

ESCUELA POLITÉCNICA NACIONAL

THESIS

For the award of the degree of

DOCTOR OF PHILOSOPHY

IN INFORMATICS

Resolution RPC-SO-43-No.501-2014 of the Consejo de Educación Superior

Presented by

JUAN ALBERTO HERRERA SILVA

Thesis supervised by **Myriam Beatriz Hernández Álvarez, Professor of the National Polytechnic School** and co-supervised by **Lorena Isabel Barona López, Professor of the National Polytechnic School.**

Ransomware Detection by Cognitive Security

Oral examination by the following committee:

Sandra Patricia Sánchez Gordón, Ph.D.
Escuela Politécnica Nacional (EPN), Coordinator

Tania Elizabeth Calle Jiménez, Ph.D.
Escuela Politécnica Nacional (EPN), Opposing Member

Jorge Sá Silva, Ph.D.
Universidad de Coimbra – Portugal, External Examiner

Enrique Vinicio Carrera Erazo, Ph.D.
Universidad de las Fuerzas Armadas (ESPE), External Examiner

Luis Felipe Urquiza Aguiar, Ph.D.
Escuela Politécnica Nacional (EPN), Internal Examiner

DECLARATION

I hereby declare under oath that I am the author of this work, which has not previously been presented for obtaining any academic degree or professional qualification. I also declare that I have consulted the bibliographic references included in this document.

Through this declaration, I transfer my intellectual property rights corresponding to this thesis, to the Escuela Politécnica Nacional, as established by the Intellectual Property Law of Ecuador, its Regulations and the current institutional norms.

I declare that this work is based on the following articles of my authorship (as main author or co-author) related to the title of this thesis:

Journals:

- J. A. Herrera Silva, L. I. Barona López, Á. L. Valdivieso Caraguay, and M. Hernández-Álvarez, “A Survey on Situational Awareness of Ransomware Attacks—Detection and Prevention Parameters,” *Remote Sens.*, vol. 11, no. 10, p. 1168, May 2019.

- J. A. Herrera Silva, F. Bazante, L. I. Barona López, Á. L. Valdivieso Caraguay, and M. Hernández-Álvarez, “Dataset de Ransomware basado en Análisis Dinámico,” *RISTI - Rev. Iber. Sist. e Tecnol. Inf.*, no. E23, pp. 248–241, 2019.

- J. A. Herrera-Silva, M. Hernández-Álvarez, “Dynamic Feature Dataset for Ransomware Detection Using Machine Learning Algorithms”. *Sensors, Volume 23, Issue 3, 1053,* January 2023

Conferences:

- J. A. Herrera Silva and M. Hernandez-Alvarez, “Large scale ransomware detection by cognitive security,” in *2017 IEEE Second Ecuador Technical Chapters Meeting (ETCM)*, 2017, pp. 1–4.

- J. A. Herrera Silva, F. D. B. Veloz, L. I. B. López, Á. L. V. Caraguay, and M. Hernández-Álvarez, “Ransomware dataset based on dynamic analysis,” in *CSEI*, 2019.

- J. A. Herrera Silva and M. Hernandez-Alvarez, “Deployment of Ransomware Detection Using Dynamic Analysis and Machine Learning” in *International Conferences on Human Factors in Cybersecurity (AHFE 2023),* 2023*.* Accepted.

I also declare that I have acknowledged the collaboration of third parties, and the contribution made by other published or unpublished material.

Juan Alberto Herrera Silva

CERTIFICATION

I certify that **Juan Alberto Herrera Silva** has carried out his research under my supervision. To the best of my knowledge, the contributions of this work are novel.

Firmado electrónicamente por:
MYRIAM BEATRIZ
HERNANDEZ ALVAREZ

Myriam Beatriz Hernández Álvarez

ADVISOR

Firmado electrónicamente por:
LORENA ISABEL
BARONA LOPEZ

Lorena Isabel Barona López

CO ADVISOR

ACKNOWLEDGEMENTS

First, thanks to God for giving me health and knowledge to allow me to reach the culmination of this career.

To my supervisor and friend PhD Myriam Hernández, for his support, help, patience and valuable advice during the development of this work.

To my beloved daughters Aurita and Dianita, for being an important pillar in my life.

To my heavenly mother, for always being a guide to overcome adversity and to my father for his unconditional support.

To my students Freddy and Ronny, my great work team, for their great effort and support for the development of this work.

Thanks,

Juan Herrera

DEDICATION

I dedicate this work to my precious daughters, who are the engine of my life and the reason for my improvement. Everything that is done with perseverance always leads to success. I love you beautiful princesses.

Juan Herrera

CONTENTS

ABSTRACT 1

PROLOGUE 2

1. INTRODUCTION 4
   1.1. Justification 4
   1.2. Objectives 7
   1.3. Hypotheses 9
2. BACKGROUND 10
   2.1. Ransomware History 10
   2.2. Ransomware Taxonomy 14
      Locker Ransomware 14
      Crypto Ransomware 14
      R4IoT 15
   2.3. Stages of Ransomware Attacks 15
   2.4. Ransomware Analysis 18
      Static Analysis 18
      Dynamic Analysis 18
   2.5. Statistics of Ransomware Attacks 19
3. RELATED WORK 21
   3.1. Current Research 23
   3.2. Most used Features 27
   3.3. Dataset Repositories for Benign and Ransomware Samples 32
   3.4. Comparison with Previous Research 36
4. MATERIALS AND METHODS 38
   4.1. Cuckoo Sandbox 38
   4.2. Feature Extraction Tool 39
   4.4. Phases of Experimentation 43
      Phases Initial and Analysis 47
      Phase Final – Selected Features 50
   4.5. Test Setting 54
      Phases Previous, Initial and Analysis 54
      Phase Final 61
   4.6. Balanced Dataset 65

4.7. Machine Learning Algorithms 66
Model Generation with Machine Learning Algorithms 67
Machine Learning Parameters for Phase Final 68
Random Forest Parameters 68
Gradient Boosted Regression Trees Parameters 69
Gaussian Naive Bayes Parameters 70
Neural Networks Parameters 70
Performance of the Classifiers 71
5. DATASET, MODELING, AND DEPLOYMENT 73
5.1. Evolution of the Research to Obtain the Ransomware Features Dataset 73
Experiments in Phases Initial and Analysis 76
Combination of Features 76
Comparison of testing results 80
5.2. Final Datasets of Features of Ransomware corresponding to Phase Final 88
Dataset Global from Phase Final 89
Modeling Results with the Dataset Global 91
Dataset Extract from Phase final 93
Modeling Results with the Dataset Extract 95
5.3. Deployment 96
Repository Content 99
6. DISCUSSION 101
6.1. Contributions of This Work 101
7. CONCLUSIONS 103
8. FUTURE WORK 106
REFERENCES 107
ANNEXES 121

INDEX OF FIGURES

Figure 1. Ransomware detection scheme 8
Figure 2. Ransomware by Emotet Infection 10
Figure 3. Timeline of the evolution of ransomware 13
Figure 4. A cycle of a ransomware attack 16
Figure 5. MITRE ATT&CK Matrix for Ransomware. 17
Figure 6. Industries attacked by Ransomware during Q2 2021 20
Figure 7. Most common Ransomware Attack Vectors 20
Figure 8. General structure of the JSON reports 39
Figure 9. GUI components of the feature extraction application. 40
Figure 10. CSV file with extracted features 42
Figure 11. Test environment network topology for Phases Previous to Analysis 54
Figure 12. Features analyzed for the dataset using Cuckoo Sandbox in Phases Previous, Initial and Analysis 57
Figure 13. Average characteristics 60
Figure 14. Features for Phase final that are automatically generated. 64
Figure 15. Final Test environment network topology used in Phase Final 65
Figure 16. Comparison of performance of the algorithms over the training dataset – Phase Analysis 79
Figure 17. Comparison of performance of the algorithms over the testing dataset – Phase Analysis 81
Figure 18. Dataset rows corresponding to an ‘UDP’ feature of an artifact. 90
Figure 19. Ten-fold cross-validation accuracy results obtained in Dataset Global from Phase Final. 93
Figure 20. Information for a single artifact 94
Figure 21. Ten-fold cross-validation accuracy using the Dataset Extract from Phase Final. . 96
Figure 22. Deployment architecture 98

INDEX OF TABLES

Table 1. Summary of current research (2016 – 2022). 23
Table 2. Summary of evaluated parameters and tools. 29
Table 3. Ransomware datasets (2020 -2022)..... 32
Table 4. Characteristics of dynamic analysis solutions..... 33
Table 5. Description of the experimental phases. 43
Table 6. Feature description..... 44
Table 7. Description of dataset column (features Phase initial and analysis)..... 48
Table 8. Description of the dataset features for Phase final..... 50
Table 9. Datasets description 53
Table 10. Artifacts for Datasets in Phases Initial and Analysis 55
Table 11. Selected characteristics Phases Initial and Analysis..... 58
Table 12. Artifacts used in Phase final..... 61
Table 13. Machine learning algorithms..... 66
Table 14. Relevant attributes of the datasets generated in each phase using specific artifacts and platforms..... 73
Table 15. Processing times for a combination of 7 characteristics..... 76
Table 16. Processing times for a combination of 14 characteristics..... 77
Table 17. Processing times for a combination of 5 characteristics..... 77
Table 18. Nomenclature for the combination of characteristics 78
Table 19. Comparison of performances with different algorithms in the training dataset 79
Table 20. Comparison of performance of the algorithms over the testing dataset 80
Table 21. Performance of the classifiers using five features over the training dataset..... 81
Table 22. Performance of the classifiers using five features over the testing dataset..... 81
Table 23. Performance of the classifiers using a combination of four features in training and testing..... 82
Table 24. Performance of the classifiers using a combination of three features in training and testing..... 83
Table 25. Performance of the classifier using a combination of two features in training and testing..... 83
Table 26. Performance of the classifiers using one feature in training and testing 84
Table 27. Best accuracy in training (Phase initial) 85
Table 28. Best accuracy in testing (Phase initial) 85

Table 29. Best accuracy in training (Phase analysis) 86
Table 30. Best accuracy in testing (Phase analysis)..... 87
Table 31. Training and testing classification reports..... 87
Table 32. Performance results for Dataset Global from Phase Final 92
Table 33. Performance of the classifiers using Dataset Extract for Phase Final 95

ABSTRACT

Ransomware-related cyber-attacks have been on the rise over the last decade, disturbing organizations considerably. Developing new and better ways to detect this type of malware is necessary. This research applies dynamic analysis and machine learning to identify the ever-evolving ransomware signatures using selected dynamic features. Since most of the attributes are shared by diverse ransomware-affected samples, our study can be used for detecting current and even new variants of the threat.

This research has the following objectives: (1) Execute experiments with encryptor and locker ransomware combined with goodware to generate JSON files with dynamic parameters using a sandbox. (2) Analyze and select the most relevant and non-redundant dynamic features for identifying encryptor and locker ransomware from goodware. (3) Generate and make public a dynamic features dataset that includes these selected parameters for samples of different artifacts. (4) Apply the dynamic feature dataset to obtain models with machine learning algorithms. Five platforms, 20 ransomware, and 20 goodware artifacts were evaluated. The final feature dataset is composed of 2000 registers of 50 characteristics each. This dataset allows for a machine learning detection with a 10-fold cross-evaluation with an average accuracy superior to 0.99 for gradient boosted regression trees, random forest, and neural networks.

As a complementary tool, the present study developed an application for extracting information from the dynamic analysis of artifacts generated in a sandbox. Additionally, a client-server architecture was established for deployment and application in the protection stage. The models´ performance were evaluated with the new test data to simulate this early protection phase for deployment. The obtained results were very satisfactory.

INDEX TERMS Classification, Dataset, dynamic, analysis, Encryptor, features, Locker, Machine Learning, Ransomware

PROLOGUE

The present work contributes to the knowledge of some still open issues about ransomware detection using cognitive security. One of these issues is the necessity of a dataset containing dynamic features corresponding to all the ransomware attack patterns that could be used to train supervised algorithms and neural network models. This dynamic feature dataset should include all the relevant attributes related to the threat's behavior and be open to supporting the development of new machine learning ransomware detection solutions. Our work aims in this direction.

The author has generated a dataset comprising the dynamic features of locker and encryptor ransomware and characteristics extracted from goodware. The features were selected with the criteria that they must be related to the effects of ransomware. The literature found that a ransomware dataset with these characteristics was needed because the ones that are publicly accessible do not have dynamic features of the artifacts; it is characteristics that are extracted when the software is executed. Still, only fixed signatures or their results are challenging to replicate or use for lack of enough descriptive information.

Dynamic analysis is essential for ransomware detection because the run-time attributes have enough information for machine learning early detection of these threats. In our study, since most of these features are shared by diverse ransomware samples, our dynamic analysis can be used even for detecting new variants. The characteristics were selected using criteria related to the role of each attribute in the ransomware attacks and the results of experimentation with machine learning algorithms aiming to obtain the best performances. For better classification results that even detect variants not included in the training set, it is necessary to use a more complete description of the ransomware activities delineated by the presence of all the relevant dynamic features.

To develop the final feature dataset, this research has used three classes of classifiers: locker ransomware, encrypting ransomware, and goodware. Using our dynamic feature extraction tool, the features were tested, and 50 characteristics were selected because they comply with criteria related to ransomware attacks. They were also checked for low pairwise correlation to avoid redundant information, and the machine learning algorithms' performance was high. The researchers used 20 ransomware artifacts and 20 goodware

families tested with ten experiments, each over five platforms, to produce a dataset with 1’424.344 rows. For this dataset, there were several rows corresponding to one JSON. The best performance results were obtained with gradient boosted regression trees with values of 0.98 for 10-fold cross-evaluation accuracy.

To generate a more portable, efficient, and concise dataset without losing relevant information, the research developed a process for synthesizing all the rows corresponding to one JSON into one row. Using the information provided for the previous repository, the study obtained a second dataset with 2000 records corresponding to forty families and ten experiments for each artifact over five platforms. Using this dataset, performance results for our models improved even more for gradient boosted regression trees, random forest, and neural networks because they reached values close to perfect detection for ransomware.

In the deployment, predicting new artifacts requires applying the generated models, whether in the repository or not. The programs developed in this research allow for changing the directories of CSV JSON files and models to readily execute them in the production stage.

Those, as mentioned above, are the scientific contributions of the present doctoral thesis.

1. INTRODUCTION

The inner workings of the phenomenon known as ransomware is no longer a motley army of scammers. Its growing impact is now powered by dedicated teams working within an organized business framework. The US government manages a portfolio of risks that no corporation can imagine. Some risks are easy to guess, for example, a terrorist attack or a financial crisis; then, there is a whole new category based on cyber terrorism. In recent years in the United States, the two most remote and unexpected events were an airborne virus that claimed hundreds of thousands of lives and a random series of cyberattacks that left the country without access to vital services. The US government openly confessed that it had not kept up with the world. They have spent too long baffled that such attacks were no longer hypothetical. They had become something real. Cyber warfare has become a great leveler on the international stage. It represents an opportunity for non-state actors to give blood to any superpower; it is understandable that large corporations, like Apple, were left shaking with their intellectual property and customers now fully exposed, and it turns out that very few of them wanted to talk about it for fear that the acknowledgment of the risk would be an open invitation to be hacked.

Because of the amount of sensitive information stored on both devices and the cloud while transferring over the network, malware detection, especially ransomware, has become a primary research topic in recent years. A ransomware-like attack uses a set of stages to infect a system; it starts with the device's distribution and infection. This malware searches for files to infect. It encrypts files, requests ransom, and threatens exposure to the affected company's sensitive information in case of non-payment.

Ransomware malware continues to grow and transform; it took advantage of the anonymity provided by the growing popularity of cryptocurrencies. The researchers observed the emergence of numerous variants after 2013. After the switch to crypto-ransomware, ransomware continued to evolve, adding features like countdown timers, ransom amounts that increase over time, and infection routines that allow it to spread through networks and servers.

1.1. Justification

Cybercrime activities have grown significantly in recent years by compromising device security and jeopardizing regular business affairs. The profits obtained through intimidation and limitations for tracking illegal transactions have created a lucrative business based on hijacking users' files. In this context, ransomware takes advantage of cryptography to compromise user information or deny access to the operating system. Then, the attacker extorts the victim to pay a ransom to regain access, recover data, or keep the information private.

As of 2017, this threat had attacked hundreds of thousands of computers. According to the US Department of Justice, more than 4,000 attacks have been reported per day[1]. The situation is aggravated by the development of IoT technology that allows the availability of new devices on the Internet with open access and the continuous emergence of new variants of this virus.

The idea that a company's data is encrypted and copied is insidious. However, no organization should allow ignorance and grievance to drive policy. So it is time to rectify misconceptions about one of today's most fascinating and alarming corporate threats; that critical gap at the heart of the cyberattack came back to work when the company paid hackers millions in Ransom on May 7, 2019. A large part of the colonial oil pipeline from Texas to New York City was closed in 2021. The wall between essential and non-essential had been breached. The FBI had not seen the attack coming and, as the operator put it, after paying Ransom's price of $4.4 million.

Ransomware operators have business models and are no longer content to only target people who earn just a few dollars. At the same time, they have become an entity that, even with only moderate cyber warfare capabilities, could attack a country like the United States with a little more organization. By extension, the malware could also stop air traffic in Paris and eventually bring Philadelphia trains to a standstill. They picked a Texas pipeline that crippled South Carolina. The first real modern ransomware program dates back to 2005 with the release of the pgp encoder. Victims would visit an infected website that would take advantage of inherent flaws within browsers. Then, ransomware progressed, switched from symmetric to asymmetric encryption, and further thwarted the security industry's efforts to

[1] https://www.hhs.gov/sites/default/files/RansomwareFactSheet.pdf

create practical decryption tools. The monetization strategy is also changing, with cryptocurrencies replacing other more traceable methods.

Attackers exploited web and file servers and were deliberately positioned with public sector organizations in mind. Furthermore, no one rushed to calculate the financial risks because no one dared to make the assessment. It is as if you needed to set up an entirely new department to combat attacks or find a security guru who knew how to respond to various attacks.

The city of Atlanta has also dealt with major cyber-attacks. Hackers froze computer systems and demanded payment, and in a Tweet, the city said the cause of the attack was after the Sam Sam outbreak in 2018, the city of Atlanta faced a $51,000 demand for unlocking all computers. After accounting for outage and remediation expenses, the final Ransom Bill exceeds $2.6 million. Thus, we come to the latest generation with ransomware operators adopting a franchise model[2].

Managing risk is an act of the imagination, and government officials and businesses are somewhat good at responding to a crisis and less good at taking action to prevent it in the first place. Every measure today has to do with progress, the progress of society, and the economy. The ransomware attack can be a big shock.

Ransomware attacks have become a serious threat to information security globally, so the scientific community does constant research to detect and prevent such attacks. Despite these efforts, ransomware continues to be prevalent worldwide because antivirus and anti-malware cannot recognize them because they use polymorphism and machine learning to avoid their recognition. On the other hand, exploit kits have appeared that efficiently produce new ransomware variants, which are sold on sale and with discounts so that anyone can develop this malware. Our research develops a strategy with the same type of weapons with which this malware is presented; machine learning is used to detect the threat before it can hijack and encrypt the data.

[2] https://www.nytimes.com/2018/03/27/us/cyberattack-atlanta-ransomware.html

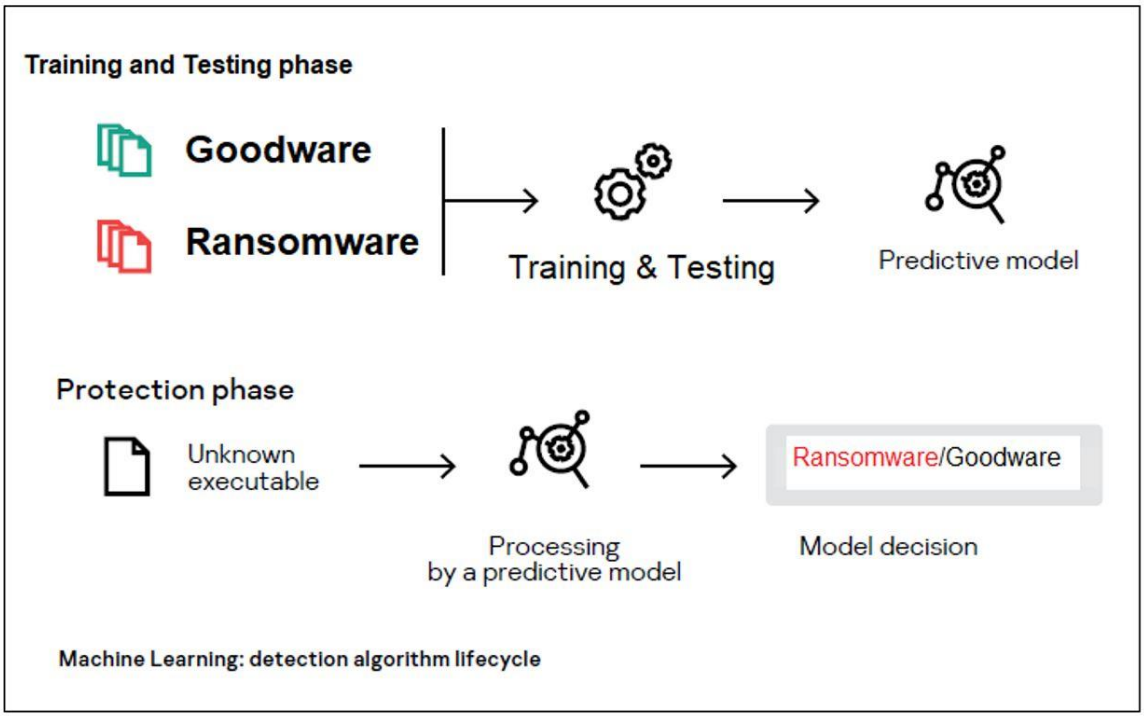
1.2. Objectives

The literature study found that there are no readily available public databases with dynamic information on this type of attack, or the existing data are challenging to use because they are not described in enough detail. In this context, this work proposes to create a dataset with all the information necessary for its use; this dataset will be publicly available. We will also indicate the parameters that have been selected as features. In our dataset, an analysis of selected characteristics is presented.

Our research aims to create a dataset that associates the ransomware samples used with the most distinctive dynamic virus features to detect them before the attack does its damage. This work presents this material to make it available to the scientific community and thus contribute to advancing the fight against this computer threat. The dataset will be used to create models that allow early detection of the virus and achieve a proactive response that minimizes the damage this malware can cause.

The parameters involved in creating the dataset are based on Cuckoo reports, considering 326 features. This information creates a Ransomware Feature Dataset. Ransomware encrypts the files of its victims' computers for a short time to hijack the information and ask for a ransom. Standard methods of discovering the malware's signature do not work because the virus has a continuous evolution, making detecting this virus's action difficult. Therefore, new protection mechanisms must focus on ransomware's operations before encrypting files.

The goal of our work, through our dataset, is to analyze the virus's behavior using machine learning algorithms, as shown in Figure 1. In the first step, we generate a feature vector that provides justified, meaningful, and relevant information about the threat. This feature vector will feed classifiers to obtain models for early risk detection. The dataset produced in this study defines the feature vector composed of relevant characteristics, tests the models, and specifies those that perform best.



Based on [1] - *Author: Juan A. Herrera Silva*

Figure 1. Ransomware detection scheme

The present research has the following objectives:

1) Execute experiments with goodware, encryptor ransomware, and locker ransomware to generate JSON files with parameters that characterize the artifacts. For this purpose, we use simulations in an isolated environment with tools like Cuckoo Sandbox.
2) Analyze and select the most relevant parameters for identifying encryptor and locker ransomware from goodware.
3) Generate a dataset that includes these selected parameters for samples of different artifacts.
4) Apply the dataset to the generation of models obtained with machine learning algorithms to detect encryptor and locker ransomware using different combinations of features to determine the selection of parameters that gives the best algorithm performance. These models will allow the ransomware to be detected before the information is encrypted and hijacked.
5) Make this dataset publicly available to contribute to advancing the fight against this malware.

1.3. Hypotheses

Our hypotheses are:

- It is possible to build a dataset containing encryptor and locker ransomware and goodware dynamic features corresponding to several artifacts in specific platforms.
- The features will deliver enough information to produce machine learning models to detect encryptor and locker ransomware, with performance over the state-of-the-art values, and their deployment will allow early detection of ransomware to minimize the damage it can cause.

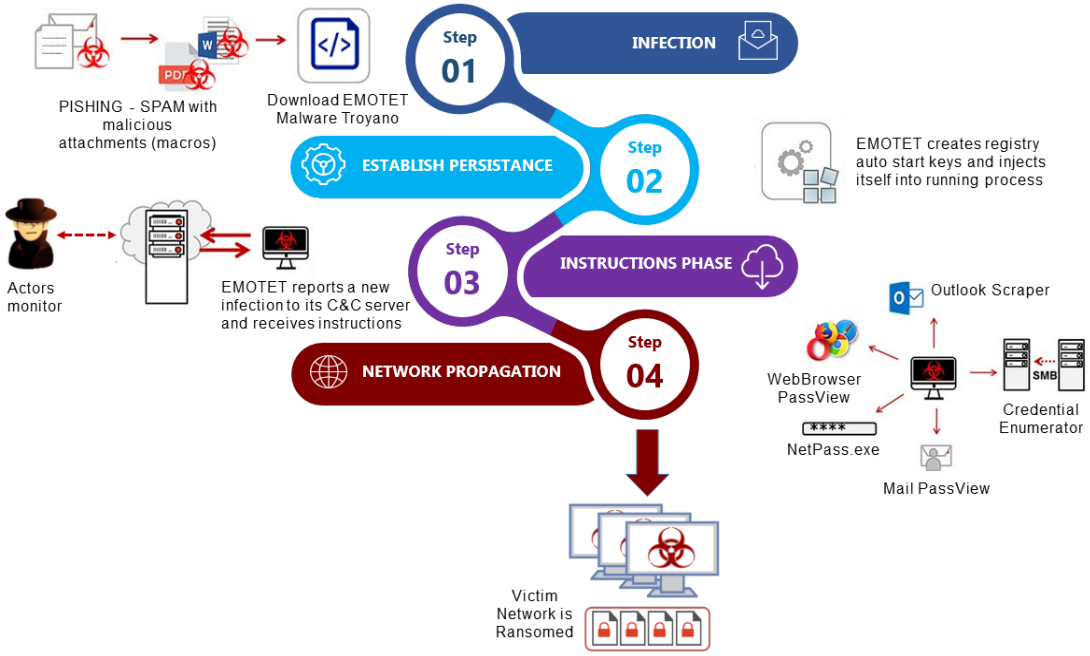
The present document consists of six sections. The first one is this introduction. Section 2 gives context to the problem; this chapter includes a description of the ransomware evolution, the attack cycle, statistics, and the definition of concepts in this topic. Section 3 is about related work, describing current research, most used features, and datasets. Section 4 describes the materials and methods used in our work. Section 5 presents the generated dataset, the modeling using the selected parameters as input to machine learning algorithms to classify goodware, encryptor, and locker ransomware and their respective results. In this chapter, it is also presented a deployment of the best models. Finally, section 6 exposes the study's conclusions.

2. BACKGROUND

In this section, we cover a history of the evolution of ransomware from its origins until date, a definition of the taxonomy of this threat, the stages of the attacks, and the types of ransomware analysis.

2.1. Ransomware History

For context, we start explaining the process of one possible infection channel: Emotet[3] infection. The virus may arrive through a script, document files with macros, or a malicious link. The process is described in Figure 2, which starts with the infection, then affects the system logs (establishes persistence) and proceeds to establish a connection with the C&C servers to receive instructions (instruction phase), after which it spreads by Network-wide infection affecting victims with ransomware (network spread). (Fig. 2).



Author: Juan A. Herrera Silva

Figure 2. Ransomware by Emotet Infection

Like any threat, ransomware is in continuous evolution. Like most malware, its goal is not to be detected or generate the most significant possible impact on infrastructure. Today, people are not only talking about cyber criminals demanding money but about threat actors

[3] https://devel.group/blog/todo-lo-que-necesitas-saber-sobre-emotet-en-2022/

(ATP - Persistent Advanced Threat), who can encrypt information and enter a system to perform espionage, capture sensitive information, or gain access to inside information.

Depending on the actor, an attack can use different techniques to enter the network. Methods include exploiting a vulnerability in a system exposed to the Internet using tools and tactics to infiltrate systems and networks such as: phishing, external remote services (VPN Virtual Private Networks, RDP Remote Desktop Protocol) or Zero-Day Exploitation, and infection of some trusted websites operated by members of an organization, among other techniques.

Consequently, with that purpose, the attacker has a wide range of malware on the black market. One such service is currently provided by the Emotet malware, which was initially known as a banking Trojan. For its polymorphic versatility and ability to reach the end-user in a more friendly way, via e-mail, an Office-type document, or some JavaScript file. It can be downloaded from Internet repositories. In this way, attackers use Emotet as a dropper; a Trojan is used to install other types of malware on the operating system. Figure 2 explains the process of ransomware by Emotet infection.

Big Game Hunting is on the rise. More groups are distributing ransomware and ransomware-as-a-Service (RaaS). They are focusing their attacks on extensive enterprise networks rather than individuals. Big Game Hunters frequently use different trojans to gain an initial foothold in the target network. In 2020 the scientists saw Ryuk operators employ Emotet and Trickbot[4]. This trend shows that phishing e-mails are still the most common technique used for initial access.

Some groups that used simple Remote Desktop Protocol (RDP) brute force as an initial access technique did not even have ransomware in their arsenals and used a legitimate encryption tool instead. Simultaneously, even some of the most advanced Big Game Hunters employed this initial access vector in some cases. Cerber, as an evolved ransomware technology, statistically surpassed the number of ransomware detected in the Asian region in 2019. WannaCry ranks first globally in 2020 and continues to pose a threat after its rapid expansion in 2017 [1].

[4] https://www.bankinfosecurity.com/emotet-ryuk-trickbot-loader-ransomware-banker-trifecta-a-14126

According to The State of Ransomware of Sophos 2020[5], attacks have skyrocketed since the big transition to remote work. Not only are attacks increasing, but they are also more sophisticated and dangerous. Ransomware attacked 51% of the organizations in the last year. The criminals managed to encrypt the data in 73% of these attacks, and 59% of attacks encrypted detailed data in the public cloud, which became the most successful ransomware attack in cybercrime. EvilQuest affected Mac operating systems (June 2020). This threat is more present than ever; it has not stopped even during pandemics. Therefore, it requires contributions that allow us to destroy this malware finally.

There were 623.3 million **ransomware attacks** worldwide in **2021** and 304.6 million detected attacks in **2020**. Between 2020 and Q2 2022, **the volume of ransomware attacks peaked** in Q2 2021 with 188.9 million attacks. The 5 most representative ransomware families in 2021 were: Stop with 51%, Revil with 34%, Cerber with 4%, Conti with 2%, Darkside with 1%. Others with 8%[6].

Figure 3 presents a timeline of the most representative changes in ransomware families and its evolution, from its appearance in 1989 to 2022. The first ransomware appeared in 1989 as AIDS since 1989, then new families have appeared such as Blockers, encryptors, ransomware as a service and extortionists who publish the kidnapped information of clients, exposing the reputation of those affected. Nowadays, they also act as denial of service over the network. Major ransomware has appeared affecting Windows operating systems such as Wanna Cry, but also affecting Linux systems such as Ransom X and EvilQuest for Macintosh systems. It should be mentioned that there are several ransomware that continue to affect systems today, despite the fact that they came out in previous years, such as: Cerber, Stop and Revil.

[5] https://news.sophos.com/en-us/2020/05/12/the-state-of-ransomware-2020/
[6] https://www.antivirusguide.com/cybersecurity/ransomware-statistics/?gclid=CjwKCAjwq-WgBhBMEiwAzKSH6MbtY3_fLUDo8CVnDWTblLKf7g25wev2QEMizoxgS-S1A18BmFeQIBoCZD4QAvD_BwE

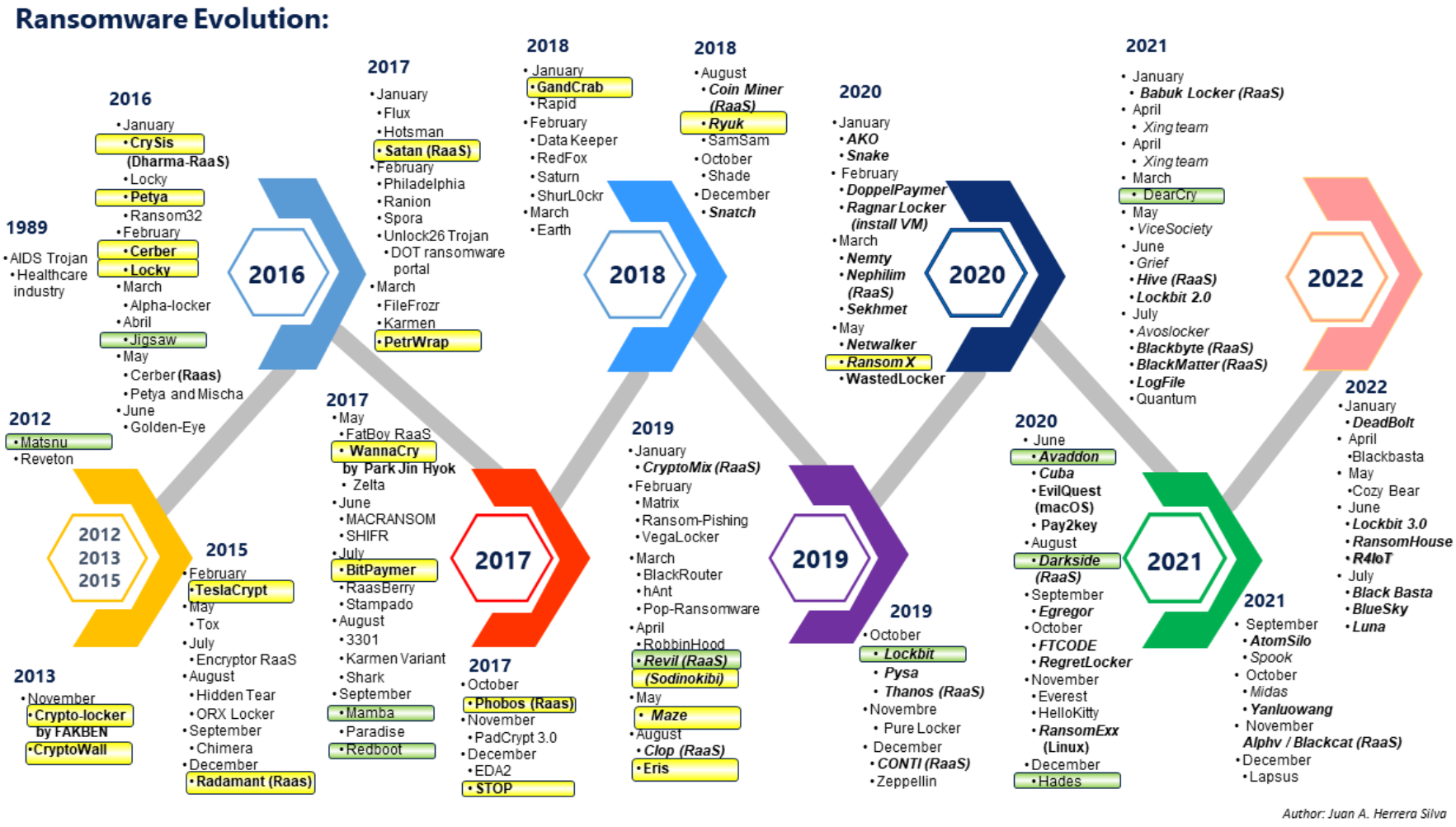


Figure 3. Timeline of the evolution of ransomware

The Ransomware selected for the construction of the final Dataset and the experimentation with learning models is highlighted in yellow and those related to detection with deployment are highlighted in green in Fig.3. Ransomware has evolved, and it is increasingly dangerous. Nowadays, there are more forms of extortion. The attackers not only hold data hostage and ask for ransom but also extort with the threat of publishing the sequestered data.

For this, REvil and others also offer a service where customers, partners, and the press are called to spread confidential information if the ransom is not paid. Furthermore, the gangs may carry out a DDoS attack to shut down companies' servers to keep the victim uncommunicated. The ransomware variant Yanluowang adds a new threat: the repetition of the attack in a few weeks, deleting all the data.

2.2. Ransomware Taxonomy

Ransomware can be classified according to the kind of victim it tries to affect, the method of infection, the mode of communication with the command-and-control server, and the type of malicious activity it performs on a computer asset [2]. For the development of our research, we focus on this last type of classification. There are two families of ransomware depending on the type of activity carried out on computer assets:

Locker Ransomware

This family blocks access to the computer system to close access to its users until they pay a sum of money [2], [3], [4]. The threat posed by this type of ransomware depends on the lock it implements. Some examples only block access to the graphical interface, which makes them less effective, while others act directly on the Master Boot Record of a system, which makes it much more dangerous [4].

Crypto Ransomware

This type encrypts files found within a computer system, rendering them completely unusable and inaccessible until a sum of money is paid [2], [3], [4], [5]. This type of ransomware represents a higher threat than the Locker family since the infected files remain completely inaccessible even if the ransomware is removed from the computer system [3]. Examples of this type of malware use symmetric, asymmetric, and hybrid encryption techniques to encrypt files and protect the cryptographic keys [4]. Some variants steal the

information hosted on a system and threaten the affected parties with the publication or sale of the information in case the demanded money is not paid [6].

R4IoT

It is a ransomware variation that demonstrates how Internet of Things (IoT) and Operational Technology (OT) exploits can be combined with a traditional ransomware campaign. It also indicates that mitigating these attacks requires solutions that enable complete visibility and greater control of all network assets[7].

This ransomware maps the different machines on the network and uses the password hashing of the administrator account and the Windows Management Instrumentation (WMI) functionality used to manage the devices and applications in a network from Windows. The virus disables Windows Firewall and Windows Defender and drops other R4IoT executables (a cryptocurrency miner and memory executable that will launch denial of service attacks against critical IoT/OT assets). A modified version provides Command-and-Control (C&C) server/agent functionality. It is a computer-controlled by the attacker that sends commands to the victim's system to obtain stolen data. At the request of the C&C server, the C&C agent can encrypt or decrypt files on the infected machine, exfiltrate files and launch arbitrary executables with administrator privileges.

This ransomware could attack Programmable Logic Controllers (PLCs), i.e., computers used to automate industrial electromechanical processes; this would have an immediate and difficult to mitigate effect. Since PLCs are rarely exposed to the outside world, it would be an internal DoS attack. Attacking PLCs could stop critical parts of business operations, be it a conveyor belt or an infusion pump. R4IoT is not a new malware development; it uses existing exploits. More worryingly, the Proof of Concept could be used by less sophisticated cybercriminals using Ransomware-as-a-Service (RaaS).

2.3. Stages of Ransomware Attacks

When a ransomware attack is accomplished, the following processes are carried out: contagion, spread, action, and warning, as shown in Figure 4. In encryption ransomware, the following phases are considered: distribution, infection, communication, file search,

[7] https://unaaldia.hispasec.com/2022/06/r4iot-el-futuro-del-ransomware-ya-esta-aqui.html

encryption, blocking, and ransom request. For blocking ransomware, only access to the computer is blocked [7].

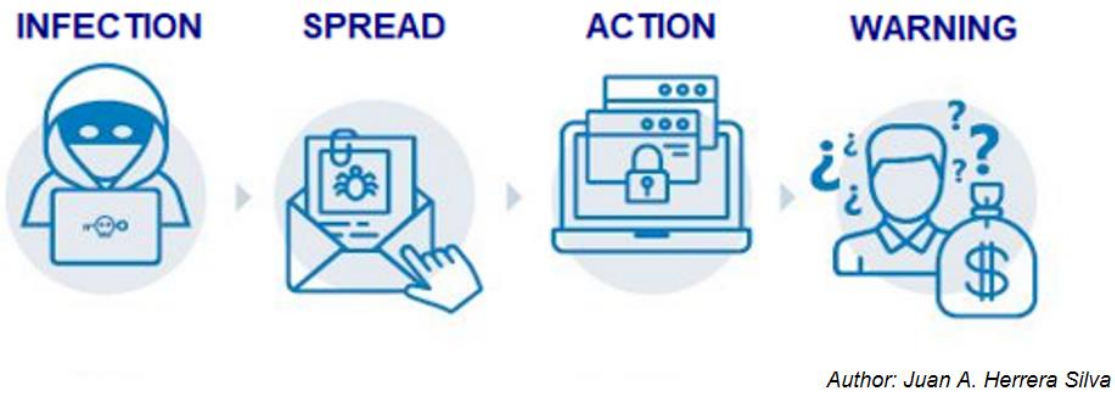


Figure 4. A cycle of a ransomware attack

In the latest methods used by the Ransomware groups [8], the researchers examined the most effective 2020 campaigns. The matrix, MITRE ATT&CK [8], is shown in Figure 5. It details their most common (highlighted in red) and also less used (highlighted in green) tactics, techniques, and procedures (TTP s). This threat uses remote access and attacks via phishing with attached files in the initial entry, then it executes commands (Power Shell), set persistence (affecting the registry), scales privileges (accesses), and applies defensive evasion techniques. Besides, ransomware does network scans with lateral infection movements, taking control and command for transferring hijacked files to the cloud, and concludes with data encryption to end inhibiting the system.

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Command and Control	Exfiltration	Impact
Drive-by Compromise (T1189)	**User Execution (T1204)**	**Registry Run Keys / Startup Folder (T1060)**	**Valid Accounts (T1078)**	**Disabling Security Tools (T1089)**	**Brute Force (T1110)**	**Network Service Scanning (T1046)**	**Remote Desktop Protocol (T1046)**	Data from Local System (T1005)	**Remote Access Tools (T1219)**	**Transfer Data Cloud Account (T1537)**	**Data Encrypted for Impact (T1486)**
External Remote Services (T1133)	**PowerShell (T1086)**	**External Remote Services (T1133)**	Exploitantion for Privilege Escalation (T1068)	Group Policy Modification (T1484)	**Credential Dumping (T1003)**	**Network Share Discovery (T1135)**	**Windows Admin Shares (T1077)**	Data from Network Shared Drive (T1039)	**Remote File Copy (T1105)**	Exfiltration over Other Network Medium (T1011)	**Inhibit System Recovery (T1490)**
Spearphisihing Attachment (T1193)	**Command-Line Interface (T1059)**	Create Account (T1136)		Redundant Access (T1108)	Credentials in files (T1081)	**Remote System Discovery (T1018)**	Windows Remote Management (T1028)		Multi-hop Proxy (T1188)	Data Encrypted (T1022)	Resource Hijacking (T1496)
Spearphisihing Link (T1192)	Scripting (T1064)	Scheduled Task (T1053)		Masquerading (T1036)	Credentials from Web Browsers (T1503)	System Information Discovery (T1082)				Exfiltration over Command an Control Channel (T1041)	
Valid Accounts (T1078)	Windows Management Instrumentation (T1047)	Valid Accounts (T1078)		Bypass User Account Control (T1088)		Permission Groups Discovery (T1069)					
Suply Change Compromise (T1195)	Explotation for Client Execution (T1203)	New Service (T1050)		NTFS Flile Attibutes (T1096)		Password Policy Discovery (T1201)					
Trusted Relationship (T1199)	Mstha (Mstha)	Modify Existing Service (T1031)		Obfuscated Files or Information (T1027)		Domain Trust Discovery (T1482)					
Exploit Public-Facing Application (T1190)	Scheduled Task (T1053)	WMI Event Suscription (T1084)		Deobfuscate/Decode Files or Information (T1140)		Network Configuration (T1016)					
				File and Directory Permissions Modification (T1222)							
				File Deletion (T1107)							

Figure 5. MITRE ATT&CK Matrix for Ransomware.

2.4. Ransomware Analysis

In general, malware analysis is studying, observing, and dissecting malicious software to determine its purpose, origin, and functionality [9], [10]. The analysis of this type of software is necessary to develop techniques that facilitate the detection of malware and tools that allow it to be counteracted [9]. The analysis could be classified as static or dynamic.

Static Analysis

This analysis focuses on studying a malicious software artifact without running it [9], [10]. Within a basic static analysis process, several activities are carried out, such as evaluating the software artifact in question within various antiviruses, searching within a binary file for readable text strings, and examining the artifact's metadata, among others.

One of the advantages of using this type of analysis is that it allows an in-depth view of the content and behavior of an artifact. However, some disadvantages can make it difficult to carry out this type of analysis, such as code obfuscation by malware authors or if the artifact in question uses self-modifying code techniques [9]. Some of the methods used in this type of analysis are:

Disassembly: It consists on using tools that allow reverse engineering to be carried out on the device in question [10]. With this technique, the intention is to obtain the instructions of the malware in assembly language from the machine code that contains the malicious software to analyze the instructions and determine the behavior of the artifact [9].

Information Extraction: Consists on extracting the information embedded in the malicious artifact without necessarily doing reverse engineering. This process includes removing readable text strings within the artifact or searching for information based on the file extension [9].

Use of antivirus: It simply passes the malicious artifact through several antiviruses from different providers [9], [11].

Dynamic Analysis

The dynamic analysis focuses on executing the malicious artifact within a controlled environment. This execution allows to observe and monitor the behavior of the malware in the controlled environment and determine the changes it has made on it [9], [10], [11]. Since

a malicious artifact is going to be executed in this analysis, it is necessary to have a controlled and safe environment to be able to guarantee that, after executing it, counterproductive results are not obtained, such as the infection of neighboring networks or the infection of the computer that is running the malware. For this purpose, simulators, emulators, or sandboxing are used [11]. In this way, the dynamic analysis seeks to obtain some information on the execution of the artifact in question, such as:

- System calls.
- Modified system registries.
- Files created, modified, or deleted.
- Network connections established.
- Network protocols used.
- Modifications to the file system.

Our research focuses on the dynamic analysis of ransomware using a sandbox to obtain information on ransomware behavior and goodware software artifacts to conduct dynamic analysis using a cuckoo tool. In addition, the authors describe a feature extraction program developed for this purpose. During execution, the artifacts yielded 326 dynamic features that describe what the artifact does while running inside an isolated operating system. Some of these features are related to ransomware activities and are pertinent for detecting this malware using machine learning techniques. The researchers analyzed ransomware behavior and chose 50 relevant and not redundant features to feed the learning algorithms to produce an accurate classification.

2.5. Statistics of Ransomware Attacks

In this section, we present some statistics demonstrating the severity of the problem created by the ransomware gangs. The attacks are rising and increased by 140% in Q3 of 2021[8]. Figure 6 shows the common industries targeted by ransomware in the second quarter of 2021. The public sector is the most affected, and nearly one in four local government organizations admitted to having no malware recovery plan in place in the 2021 Sophos survey [12]. This sector is most likely to see encrypted data and pay the extortions.

[8] https://www.pandasecurity.com/en/mediacenter/security/ransomware-statistics/

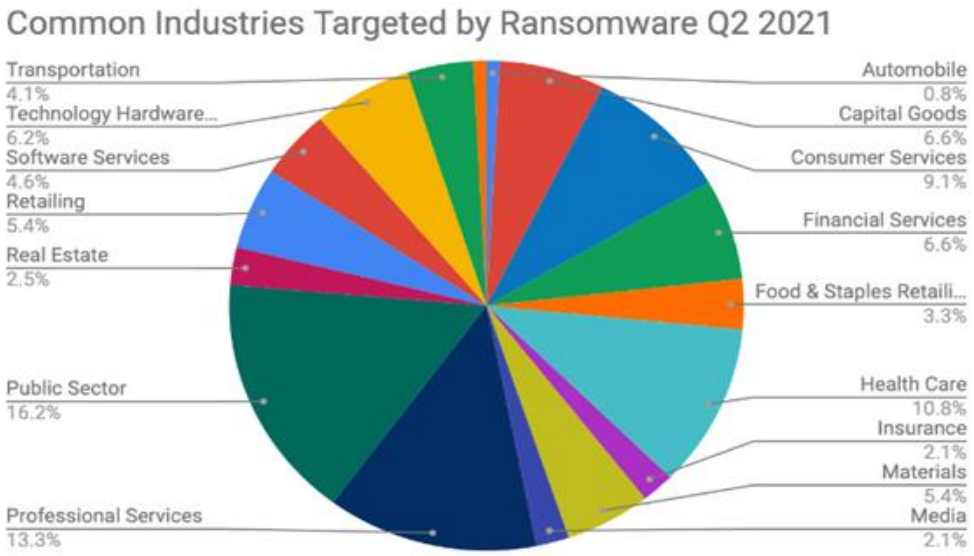


Figure 6. Industries attacked by Ransomware during Q2 2021

Figure 7 shows that in the second quarter of 2021, Remote Desktop Protocol (RDP) and Email phishing are the most common attack vectors [12].

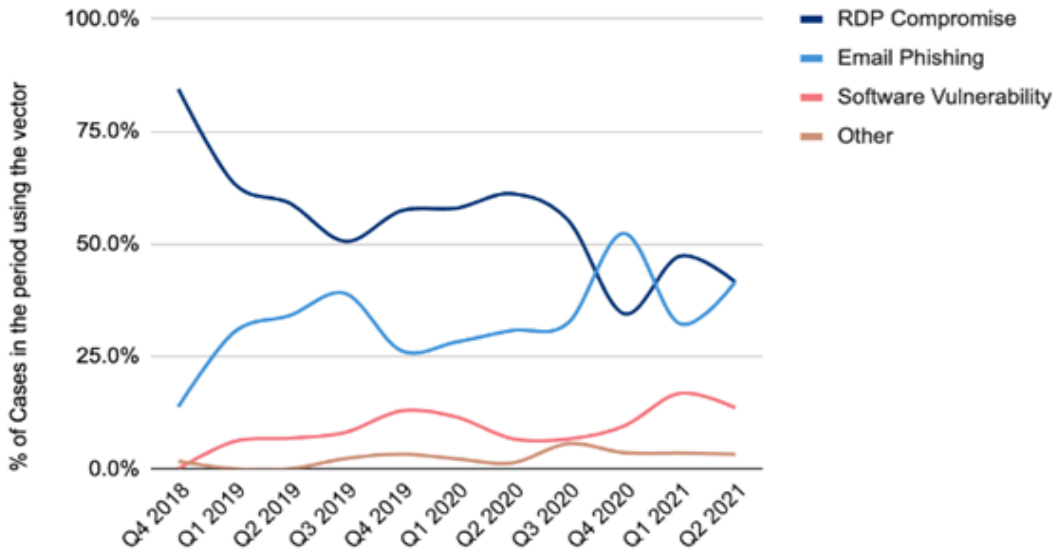


Figure 7. Most common Ransomware Attack Vectors

Since this problem is growing and has lethal effects on its victims, it is vital to develop a timely detection of this threat before it produces irreparable losses. From the review of related work detailed in chapter 3 of this thesis, we could detect that very few studies focus on dynamic analysis. Our research aims to cover this gap in diverse platforms.

3. RELATED WORK

The work on situational awareness of ransomware attacks [13] identifies parameters for detecting and preventing this attack. Besides, it presents a variety of ransomware analysis tools, including Anubis [14], VirusShare [15], VirusTotal [16], Process Monitor [17], Watchdog Module [18], and mainly Cuckoo Sandbox ([14], [16], [17], [19], [20], [21]).

Similarly, in [22], analysis is performed on a set of parameters related to ransomware attacks. The most commonly used metrics are convergence region (ROC) against file encryption, CPU utilization, valid positive rate (TPR), false-positive rate (FPR), accuracy, and recovery. On the other hand, according to the RWGuard system [23], the parameters that can influence the detection of ransomware are required packets of input and output, behavior, and CPU processing.

There are different approaches to the detection and prevention levels of such attacks. Discovery-level investigations mention the main parameters such as registry keys, system file input/output activities, process activity, entropy, API function calls [24], network activity, and network features (protocol, source, destination IP addresses, ports, packets, duration). In [15], the authors present a dataset with the following parameters: Windows API calls, registry key operations, system file operations, file operations performance set by file extension, directory operations, deleted files, and character strings. Nevertheless, it indicates that many samples used are not reflected in the dataset; there is a lack of explanation of the parameters and identifiers' description, and they do not justify why they consider them. Thus, studies that generate datasets provide only an overview of the parameters used in the ransomware attack detection process; they do not delve into their importance and are also not available to the scientific community.

Several pieces of research talk about how to scan and detect ransomware, authors in [25] propose a technique to monitor network traffic data and extract its features. These features are used in ransomware classification, and the applied algorithm is the Random Forest binary classifier. It indicates a detection rate of 86%.

On the other hand, data mining techniques are used in [26] to find unique association rules for recognizing and detecting ransomware families using a static and dynamic approach. In [21], the authors have proposed ransomware detection when making API calls. Ransomware samples run in an isolated environment to get the API call information to

create a feature database. Sample classification is performed using support vector machines. In [27], a network traffic scan has been performed for Windows ransomware. This network analysis is conversation-based, and detection accuracy is calculated using the J48 algorithm, a decision tree classifier.

In [17], a method has been introduced to detect ransomware on virtual servers. Volatile memory dumps obtained from forensic memory analyses are analyzed to create meta characteristics. The experiment was conducted using the Volatility Foundation and Random Forest Classifier as a machine learning model. According to [28], a static analysis-based approach to classify ransomware is proposed. Through inter-family discrimination, it obtains feature vectors and feeds them five machine learning methods for ransomware classification. The experiments achieve a binary classification accuracy of 91.4%, and this method can take fingerprints of the environment, which are very difficult to detect with automatic analysis.

Cuckoo sandbox is often used to isolate a working space for executing ransomware-infected files. The research papers mentioned below present the best accuracy results. In the article [29], the authors used sandboxing to obtain 64 features for 360 samples of ransomware and 532 files with other malware, and 460 samples of benign software. It is a somewhat limited dataset in terms of the number of samples. Using machine learning algorithms and values corresponding to four large feature categories: function length frequency, printable string information, and API functions, they got a maximum accuracy of 0.961. The drawback is that it is unclear which specific features inside these categories are included in the training.

Aditionally, using the Cuckoo sandbox in [30], the authors used the file and encryption features to compose a feature vector. This research achieves a 93% ransomware detection rate; accuracy is not reported. Another example is [31]. This article presents feature vector plots to distinguish a different behavior among ransomware and goodware families of artifacts. [32] proposes an active learning algorithm to detect ransomware using selected 26 features, achieving a 0.879 accuracy value for ransomware detection. None of the mentioned research papers present their datasets, which would help replicate their work.

As [33] states, the application of intelligent algorithms to detect ransomware is in an early stage but is growing. New perspectives of future developments are still ahead in this research area.

3.1. Current Research

The evolution and impact of ransomware attacks in the last decade have revealed the imperative need to discover an efficient way to mitigate or avoid this threat [34]. In this context, several studies and proposals that aid with this purpose are shown in Table 1.

Table 1. Summary of current research (2016 – 2022).

Reference	Year	Keywords/Topics	Kind of Research		
			Review	Proposal	Testing
[15]	2016	Detection, machine learning, Support Vector Machine (SVM), regularized logistic regression			X
[14]	2016	Ransomware evolution, datasets	X		X
[35]	2018	Ransomware economic impact, bitcoin trace	X		
[36]	2017	Economic analysis	X		
[37]	2017	Prevention, pattern, random forest, exploit kits, supervised machine learning			X
[38]	2016	Honeypot, detection		X	X
[39]	2017	C&C, IoT attacks	X	X	
[40]	2018	Detection methods, Decision Tree Classifier	X		X
[41]	2018	API calls, detection	X		
[42]	2017	Detection, V-detector negative selection algorithm, feature extraction			X
[43]	2018	Detection, prevention, entropy information			X
[44]	2018	Ransomware taxonomy, state of the art on prevention, detection, and prediction.	X		
[45]	2018	Unsupervised detection method, artificial neural networks, Hardware Performance Counter (HPC).			X
[46]	2018	Detection, honey file, protection			X
[47]	2018	Detection, mitigation, Software Defined Networking (SDN)			X
[48]	2018	Detection mechanism			X
[49]	2017	Analysis and detection, simple Logic (SP), SVM			X
[50]	2017	Cryptoanalysis, detection	X		X
[51]	2017	Deep learning, Long-short term memory (LSTM)			X
[52]	2018	Crypto model, encryption keys, proactive prevention			X
[53]	2018	Dynamic analysis, anomaly detection, SVM			X
[54]	2018	Backups, disaster recovery, risk assessment	X		

Reference	Year	Keywords/Topics	Kind of Research		
			Review	Proposal	Testing
[55]	2017	Deep networks, detection			X
[56]	2017	Recurrent neural network (RNN), detection			X
[57]	2018	Mitigation, detection	X	X	
[58]	2017	Ransomware evolution, safety measures	X		
[59]	2018	Detection, mitigation, SDN, NFVs		X	
[60]	2017	Crypto-Ransomware, bitcoin, Cyber currency		X	
[61]	2020	Static analysis, opcode, Machine learning			X
[62]	2018	Security, model checking, android			X
[63]	2018	Bitcoin, crypto-currency, payment	X		
[64]	2018	Volatile memory forensics memory dumps			X
[65]	2019	Deep learning, convolutional neural network, LSTM			X
[66]	2018	Remote Desktop Protocol (RDP), detection			X
[67]	2018	Behavioral detection, anomaly	X		
[68]	2018	Detection, deception systems			X
[69]	2018	Real time detection, access control, file operation			X
[70]	2018	Encryptor, file protection, document editing			X
[71]	2017	Cyber threats, security audit, penetration testing, IoT, privacy	X		
[72]	2021	Ransomware, sandbox, user-friendly model, survey	X	X	
[73]	2020	Ransomware, Detection, Prevention	X	X	
[74]	2021	Machine learning, Deep learning, Ransomware, Ransomware analysis, Dynamic analysis;		X	X
[75]	2021	Intrusion detection systems, detection rate. false alarms	X	X	
[105]	2022	Ransomware, Open dataset, Storage access pattern, Machine learning, Hypervisor		X	X
[125]	2022	crypto ransomware, data centric, process centric, event-based detection, early detection, Neural Networks, malware, machine learning-based detection	X		
[126]	2022	Network function virtualization, enterprise information system, IoT malware detection, adversarial malware, detection malware, visualization techniques, sandboxing	X	X	
[127]	2022	enterprise's private cloud, virtual machines, RAM, file system, network feature, feature selection			X

Reference	Year	Keywords/Topics	Kind of Research		
			Review	Proposal	Testing
[128]	2022	ransomware detection, cloud environment, volatile memory features, ransomware binaries and action sequences	X	X	
[129]	2022	Ransomware Classification, Feature Selection, Machine Learning, Neural Network, Cybersecurity		X	X
[130]	2022	machine learning, classification, ransomware, random-forest, elbow method		X	X

Table 1 presents a summary of the different phases followed during an investigation. These are the review of the state of the art, proposal, and testing. It also shows each paper's keywords and central topics to facilitate future research. For instance, a ransomware taxonomy and its success factors are presented in [34], [67].This state of the art is focused on ransomware counteraction from the prevention approach and detection concept. It also highlights the research direction in this field and its impact [71]. Some authors show ransomware evolution, the most common infection, and payment methods [57], [58] They also present the target users, safety measures, and the market model as a business. For instance, a ransomware economic analysis is carried out in [35], [36] . In both approaches, the economic impact of ransomware is reported. They also provide an analysis of the payment strategies, such as bitcoins, and how they contribute to ransomware proliferation.

We applied ransomware detection and prevention methods in our work, such as sandbox analysis and machine learning recognition. These kinds of approaches are detailed in [72], [73], [74], [75], [105], [125], [126], [127], [128], [129], [130]. The principal idea is to use the named methods to obtain a pattern that allows determining if an artifact is ransomware using experimentation that was carried out in different phases.

A set of suggestions to enhance the information security risk assessment guidance, specifically NIST SP 800, is given in [54].This study reviews the current backup approaches to provide a guide to address ransomware attacks, according to NIST SP 800 security management. On the other side, there are proposals focused on ransomware in Android devices [14] or the IoT area [37]. For example, [39] propose a model for analyzing incoming TCP/IP traffic (header) using a command and control server (C&C) with ransomware

blacklists. It is important to note that the analysis of ransomware threats in Android or IoT devices is out of the scope of this article.

One of the most critical challenges is to provide enhanced mechanisms to predict ransomware attacks before they happen and then apply countermeasures. On the one hand, several articles show novel detection and prediction methods [34], [38], [39]. On the other hand, some prevention mechanisms are presented to establish principles and suggestions to avoid a ransomware attack or loss of information [46], [47], [52], [54]. In [52] a key vault is proposed to protect and store the session keys. Furthermore, some proposals allow the mitigation of ransomware attacks, such as the deployment of sensors and actuators [59]. It considers the self-organized concept to provide a smart calibration and management of responses. It also contemplates the situational awareness concept of knowing the real situation of the protected environment.

It is important to highlight that big companies and organizations like the European Commission are pushing research in information security through funded projects such as RAMSES [76] or CYBECO [77]. On the one hand, Supporting Cyber insurance from a Behavioral Choice Perspective (CYBECO) develops new tools and algorithms to build more secure communication and network systems. It takes into account the behavior not only of cyber attackers but also the owners of end devices or infrastructures. On the other hand, an Internet Forensic platform for tracking the money flow of financially motivated malware (RAMSES) facilitates digital forensic research to identify internet attackers or scams. For this purpose, the RAMSES project correlates and analyzes data gathered from the Internet, particularly malware attacks such as banking trojans or ransomware. As a result of this preliminary analysis, different investigations introduce concepts like pattern recognition or prediction techniques to facilitate preventive, reactive, and proactive responses to ransomware attacks.

One of the main challenges of information security is to know what happens with the devices connected in a system and their communications, and more importantly, how to prevent and mitigate possible threats. All of these issues can be covered using a Ransomware Situational Model. In this context, several parameters are considered in the current research, such as file system operations, entropy, registry keys, checksum values, file hashes, disk usage, and open connections. Table 2 shows a summary of different parameters that were studied and evaluated in current research [13]. It also includes the tools used not only to deploy a secure environment for testing purposes but also the

programs or approaches that facilitate the gathering, correlation, and analysis of the information.

The analysis presented in Table 2 has revealed that the Cuckoo [78] sandbox is the preferred tool for testing and evaluation. Cuckoo allows the deployment of a secure environment for testing with different kinds of malware. Every action carried out by the malware is stored (logs and reports) when an attack is simulated. One of the main advantages of Cuckoo is that it works in an emulated environment which is safe for the network. It is worth mentioning that ransomware samples can be obtained in VirusTotal, VirusShare, or some authors generate or emulate their own ransomware [29].

3.2. Most used Features

In essence, to detect a ransomware attack, most of the authors take into account the following features [13]:

Content similarity and entropy: for determining how similar the data is. Entropy is based on the degree of randomness of the bytes in a file. Typical file types like HTML or doc have a lower entropy value than binary files (exe, dll). Encryption typically produces a high entropy. Therefore, if the file has been changed and is too different compared to the expected average entropy, it can be considered a potential threat. It is important to note that high entropy is not a conclusive parameter to predict ransomware attacks because other normal processes like compression imply a high entropy value. Furthermore, newer versions of ransomware reduce entropy; consequently, a lower value does not guarantee a possible infection. Entropy can be used as a part of an attack vector to predict ransomware threats.

Monitoring C&C Communications: In ransomware attacks, a C&C server propagates instructions to infect or take control of devices called bots. In this context, monitoring unusual or continuous communications with specific internet sites will be done. For this purpose, researchers are using innovative technologies such as SDN to block communication with the C&C server when it is happening. For instance, in [79] a system to monitor suspicious network traffic is proposed. It blocks infected devices, in a real-time manner, through rules applied by the SDN controller.

Moreover, concepts like Network Function Virtualization (NFV) mitigate this problem by deploying specialized network functions like Deep Packet Inspectors (DDI) or honeypots. Besides, Domain generation algorithms (DGA) generate a set of domain names used by the C&C server and leave a trace in network traffic [80]. It is important to note that some ransomware samples do not need an internet connection to encrypt files.

File system activity: Ransomware inevitably uses function calls (e.g., I/O Requests) to execute malicious operations in the OS filesystem. The system under attack can exhibit an abnormal file system activity since many equal file system access can be requested. The main suspicious activities related to the file system can include changes in Master File Table (MTF) and I/O Request Packets (IRP) [81]. The MTF can be encrypted during a ransomware attack, and the Master Boot Record (MBR) is overwritten. Thus, monitoring these elements is an effective strategy to detect ransomware.

Monitoring registry values: It has been observed that several registry values are modified during a ransomware attack. For instance, many ransomware variants modify the values of HKEY_LOCAL_MACHINE\System\CurrentControlSet\control\N1s\ComputerName\Active ComputerName and HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\WindowsNT\ CurrentVersion \WinLogon, HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows \CurrentVersion \Run.
Similarly, the value of HKLM\Software\Microsoft\Cryptography\Defaults\Provider Types\ Type 001 as the Microsoft Strong Cryptographic Provider is read [14].

Other variants remove the volume shadow copies (Volume Snapshot services VSS files) to avoid using these backups to recover the system. Finally, ransomware opens a txt instruction file, fills it with the image of attacker payment information, and changes the desktop background to the bitmap image. In other words, the HKCU\Control Panel\Desktop\

Wallpaper value is set to %CSIDL_DESKTOPDIRECTORY%_Locky_recover_instructions .bmp [70]. In the case of encryption ransomware, crypto libraries and registers are used or accessed.

Privilege Escalation: It is considered one of the most distinctive features of ransomware attacks. Once the malicious software is downloaded to the system, it monitors the environment to check its access capabilities and, if necessary, asks for administrative rights. This access request is externalized as an app authorization button in Android devices or a malicious window requesting authorization in Windows elements (update patch). Once the

attack obtains administrator privileges, it continues the attack by locking the victim device (Windows) or setting a new lock screen PIN (Android) [14].

Monitoring API and DLL calls: The use of APIs is one of the most common ways of software development. A set of procedures, protocols, and tools is provided as logical building blocks. The programmer puts the blocks together according to their particular objective through API requests and API calls. Similarly, the attacker uses the available APIs for executing malicious activities. Therefore, some characteristics of API calls (e.g., time, type, number, sequence) can be used to model the application behavior. Then, a classifier can be trained to detect suspicious activities. For example, a suspicious sequence in API Windows uses GetThreadDesktop, CreateDesktopW, and SwitchDesktop [81]. Even though the attacker could avoid using API calls, using native APIs requires significant work due to a lack of compatibility and available documentation.

Modifications of Master Boot Record (MBR): A group of ransomware attacks is specialized in changing the Master Boot Record, which contains the executable boot code and the partition table. This attack takes advantage of the well-known position of the MBR (first sector of a hard disk) and the startup procedure. Then the system boot process loads the MBR instructions in memory and transfers them to the control system at boot time. In this context, the malicious software modifies the boot code with a bogus MBR that blocks the standard boot procedure and displays a message requesting a ransom.

Monitoring specific file type, file path, or directories: It includes monitoring file modifications to find an unusual increase of particular extensions, such as .locky. It also oversees the Volumen Shadow Copy service (VSC) to avoid that shadow copies of the systems can be erased. Moreover, it is crucial to monitor URLs and web pages.

Table 2. Summary of evaluated parameters and tools.

Reference	Year	Evaluated Parameters	Tools/Datasets
[15]	2016	API invocations, registry keys, file directory Operations, and dropped files.	VirusShare, Cuckoo sandbox, VirusTotal, Matlab
[14]	2016	Filesystem and registry in Windows; checking the MD5 hash values from Virus, file system and register activity, network communications	PEiD, PEView tool, Cuckoo, Anubis

Reference	Year	Evaluated Parameters	Tools/Datasets
[38]	2016	Honeypot folder monitored with an FSRM File Screen	EventSentry, FSRM
[48]	2016	The file path, time attributes, filesystem I/O activity	Cuckoo, OpenSSL, VirusTotal
[37]	2017	Listing of the file path and dropped file, ransom note, network activity, analyzing application payload	Cuckoo, Wireshark, tracewrangle3, Dionaea Honeypot
[39]	2017	C&C communication, public key, the connection established between victims and the C&C server.	Framework Proposal
[42]	2017	Hard disk reading and writing, the encryption and deletion of files, crypto APIs. Three types of features: API functions, behavioral expression (count IP address, ports, etc.) and memory feature.	Cuckoo Sandbox, Volatility
[49]	2017	API calls (GetModuleFileNameA, NtCreateSection, NtCreateFile, NtMapViewOfSection, NtWriteFile)	API Monitor tool, Weka
[50]	2017	File system, registry, process activity, entropy, API functions (ReadFile, QueryInformation), Master File Tables, System Service Descriptor Table	Cuckoo, VirusTotal, Process Monitor
[51]	2017	API calls, registry values	Cuckoo Sandbox
[52]	2018	Crypto Function Hooking, CryptoAPI, File recovery, SHA1 functions	Cuckoo sandbox, Raddar, VirusTotal
[55]	2017	121 API call functions (NtEnumerateValueKey, NtOpenSection, closesocket, CryptDecodeObjectEx, GetFileAttributesW)	Cuckoo sandbox, TensorFlow, Open Malware, VirusTotal
[56]	2017	API call sequences (NtOpenFile, RegOpenKeyExA, ioctlsocket, NtResumeThread, etc)	Cuckoo sandbox, VirusTotal
[40]	2018	Network features (Protocol, source, and destination address, ports, packets, duration)	Weka, Kali Linux
[41]	2018	API calls (CopyFile, CreateDirectory), InternetOpen, CryptoDeriveKey, SetFileAttributes,	Windows Power Shell, bash scripts, ProcMon

Reference	Year	Evaluated Parameters	Tools/Datasets
		GetFileType, GetFileSize, CryptoGenKey, CryptoDecodeObject)	
[43]	2018	The entropy value of the file was calculated (its format)	Watchdog Module
[45]	2018	Cache-references, cache-misses, branch-misses and branches.	iperf tool, sandbox, Kera,
[46]	2018	FIFO files, infinitive files	Bash-ransomware, linux suite, linux encoder, OpenSSL
[47]	2018	HTTP message sequences and their corresponding sizes.	Cuckoo, Alexa websites, POX
[53]	2018	API calls	Cuckoo sandbox
[82]	2020	C&C commands, Permissions requested by ransomware	Concurrency Workbench of New Century
[83]	2021	C&C Communication,	Recovery of Files via Cloud Backup
[84]	2020	Infected Files	Generate ransomware, Attack ransomware on _le or directory, Verify encryption by ransomware
[85]	2020	“Windows API calls, Windows Cryptographic APIs, Registry Key”	Machine Learning
[128]	2022	Ransomware binaries and action sequences	Advanced machine learning techniques
[129]	2022	SizeOfOptionalHeader, MajorLinkerVersion, AddressOfEntryPoint, SectionAlignment, MinorOperatingSystemVersion, SizeOfHeaders, SizeOfStackReserve, LoaderFlags, SectionsMinEntropy, SectionsMaxEntropy, SectionMaxRawsize, SectionsMinVirtualsize, ResourcesMinEntropy	Multiple machine learning algorithms: Decision Tree (DT), Random Forest (RF), Naïve Bayes (NB), Logistic Regression (LR) as well as Neural Network (NN)-based classifiers
[130]	2022	SectionsMaxEntropy and ResourcesMaxEntropy	Random forest

3.3. Dataset Repositories for Benign and Ransomware Samples

Many ransomware studies use samples from VirusShare[9], theZoo[10] , and hybridanalysis.com. They form repositories with different ratios between the number of benign and ransomware artifacts. Some repositories include general malware artifacts.

Table 3 presents a summary of ransomware repositories published from 2020 to 2022, with their respective number of samples [86].

Table 3. Ransomware datasets (2020 -2022)

Study/year	Tool	Sample types	Number of artifacts
[87]/2020	Cuckoo	Ransomware	1,354
		Goodware	1,358
[88]2020	Intel Pin 3.2	Ransomware	1,000
		Goodware	300
		Malware	900
[89]]/2020	Log Parser	Ransomware logs	17
		Goodware logs	103,330
[90]/2020	Cuckoo	Ransomware	NA
		Goodware	
[91]/2020	Cuckoo	Ransomware	904
		Goodware	942
[92]/2020	Events monitoring	Ransomware	80
		Goodware	76
[93]/2020	Sandbox	Ransomware	550
		Goodware	540
[94]/2020	Cuckoo	Ransomware	1,254
[95]/2020	Weka and Python to develop goodware	Ransomware	35,015
		Goodware	500
		Malware	500
[96]/2020	Cuckoo	Ransomware	1,232
		Goodware	1,308
[97]/2020	Cuckoo	Ransomware	2,000
		Goodware	2,000
[98]/2020	Not mentioned	Ransomware	35,369

[9] https://www.impactcybertrust.org/dataset_view?idDataset=1271
[10] https://github.com/ytisf/theZoo

Study/year	Tool	Sample types	Number of artifacts
		Goodware	43,191
[99]/2020	Cuckoo	Ransomware	1,000
		Goodware	1,000
[100]/2021	Cuckoo	Ransomware	80
		Non-ransomware	80
[101]/2020	Genymotion	Ransomware	400
		Goodware	400
[102]/2021	I/O from process execution	Ransomware	206
[103]/2020	Not mentioned	Ransomware	272
[104]/2020	Cuckoo	Ransomware	625
		Goodware	103
[105]/2022	Cuckoo	Ransomware	1,044
[129]/2022	Not mentioned	Ransomware	96,632
		Goodware	41,414
[130]/2022	Not mentioned	Ransomware	96,632
		Goodware	41,414

Table 4 shows relevant characteristics of previous studies to be used in subsequent sections of this thesis to compare our analysis with that of other researchers related to dynamic features selected, machine learning algorithms used or not, the number of samples of ransomware and goodware, platforms, and performance. Table 4 also demonstrates that most authors do not deliver a feature dataset.

Table 4. Characteristics of dynamic analysis solutions

Study	Features used in Dynamic Analysis	Machine Learning based/Algorithms used	Dataset is composed of samples of	Feature dataset made available	Platforms	Performance
[14]	Filesystem and registry in Windows. Permission monitoring in Android.	No	Ransomware of 25 families	No	Windows 10/Android	Not mentioned
[15]	API calls, Registry Key Operations, File /Directory System.	Yes / NB, and SVM	582 ransomware of 11 families, and 942 goodware	No	Windows	ROC: 0.995

[16]	File system, Access Patterns, and I/O Data Buffer Entropy.	No	148,223 general malware	No	Windows	Detection rate 96.3%
[17]	File System, I/O monitoring	No	715 ransomware	No	Windows 7	Detection rate 96.7%
[18]	Entropy analysis	No	Not mentioned	No	Windows	Accuracy 92%
[19]	HTTP traffic characteristics	No	750 CryptoWall 4.0 ransomware traffic - 750 Locky ransomware traffic	No	Windows	Detection rate 97%-98%
[21]	API Calls	Yes / SVM	588 logs, 312 goodware and 276 ransomware logs	No	Windows	Accuracy 97.48%
[23]	IRP	Yes / NB, LR, DT, RF	261 benign and malicious processes	No	Windows	Accuracy: NB: 80.07%, LR: 81.22%, DT: 89.27%, RF: 96.55%
[24]	API Calls	Yes / RF, SVM, SL, and NB	168 ransomware	No	Windows 7	Maximum accuracy SL: 98.2%
[25]	Command and control (C&C) server	Yes / RF	265 ransomware related flows.	No	Windows	Accuracy with 10 fold cross validation 87%
[26]	Portable Executable (PE) File	No	450 ransomware	No	Windows	Accuracy 70%
[27]	Network Traffic	Yes / DT (J48 classifier)	210 ransomware, 264 benign	Dataset sample showed	Windows	Maximum F-measure 96.8%
[28]	Ransomware Opcodes (Machine Language Instructions)	Yes / DT, RF, KNN, NB, GBDT	1787 ransomware	No	Windows	Maximum accuracy 99.3%
[29]	API Calls	Yes / SVM, DT, RF, GBDT	360 ransomware, 532 general malware, and 460 benign software	No	Windows	Maximum Accuracy 96.1%
[30]	API function calls, counts of the behavioral features, and counts of the memory features	No	1000 ransomware, 1000 benign software	No	Windows XP	Detection rate 90%
[31]	API Calls, File/Directory System, Shannon's	Yes / LR, SVM, RF, GBDT, ANN	574 ransomware	No	Windows 7, Windows 8.1	Detection rate 98.25%

	Entropy of File Writes					
[32]	Selects key features using Multi-Objective Grey Wolf Optimization (MOGWO) and Binary Cuckoo Search (BCS) algorithms	Yes / NB, RF, and SMO	582 ransomware, and 942 goodware	No	Windows	Accuracy NB: 79.3% RF: 82.67, SMO: 82%
[79]	C&C communications	No	Database of malicious URLs	No	-	Time to disrupt the connection: 100 ms
[81]	Master File Table (MTF) and I/O Request Packets (IRP)	No	Logs with 2000 user activity and 2000 ransomware activity	No	Not mentioned	Accuracy 97.4%
[105]	I/O operation, LBA, and Entropy	Yes / RF, SVM, KNN, CNN	7 ransomware families	Yes	Windows 7, Windows Server 2008	F-measure from 0.57 to 0.99
[89]	Semantic Information from Logs	Yes / Bi-LSTM	Logs	No	Linux Server, Windows 7	Accuracy 96.5% - 99.7%

Many ransomware studies use samples from sources already mentioned, such as VirusShare[11], theZoo[12], VirusTotal, Anubis, and hybridanalysis.com. They form repositories with different ratios between the number of benign and ransomware artifacts. Some repositories include general malware artifacts. Ransomware datasets are found in [87]. These datasets contain varied quantities of ransomware and goodware samples. Some use these datasets in isolated testbed tools. It is necessary to emphasize that these datasets are only a collection of malware and goodware obtained from different sources. Any of them have relevant features extracted from the artifacts. Also, as stated before, these datasets are not readily accessible.

As far as the author knows, there is no accessible dataset with a robust set of dynamic characteristics, making it challenging to develop detection and prevention solutions for the constantly evolving signature-changing ransomware. A complete dataset of dynamic

[11] https://www.impactcybertrust.org/dataset_view?idDataset=1271
[12] https://github.com/ytisf/theZoo

features is needed to be used as a basis for intelligent machine learning with the capability to produce models to detect this threat before it causes damage. For this reason, this research deals with these two issues: the generation of a relevant feature dataset and its use to generate machine learning models to differentiate ransomware from goodware.

On the other hand, when other authors use dynamic features, they only use some attributes, for example, attributes related to the network, API and DLL calls, or file systems. For better classification results that even detect new variants, it is necessary to use a more complete description of the ransomware activities delineated by the presence of all the relevant dynamic features.

The performance of the studies in Table 4 uses several metrics. It varies from an accuracy of 70% to a maximum of 99.7%; a maximum F-measure of 0.99; detection rate with values from 90% to 98.25%; one paper presents a ROC of 0.995; and another shows a response time of 100ns to disrupt the connection for C&C communication before the encryption is made.

3.4. Comparison with Previous Research

The experiments carried out by other authors cannot be reproduced because we do not have enough description of the environment, the datasets, or the specific dynamic parameters with which they work. Other papers only state the number of ransomware and goodware samples used, their sources, such as VirusTotal or VirusShare, and a not enough detailed description of the dynamic parameters applied. Therefore, the information in Table 4 helps compare the methods and results of other studies with the ones in our research. Our results are comparable to or better than those reported in other studies with an almost perfect 10-fold cross-validation accuracy using random forest and gradient boosted trees.

It is important to state that the authors of the present paper initially conducted experiments with partial sets of relevant features in the initial stages of the work. For instance, the researchers used a partial set of relevant features over the training dataset. They obtained results similar to the ones obtained with the complete set of 50 attributes, as seen in Table 21. The characteristics used correspond to procmemory: file_created; behavior (processes and apistarts): regkey_read, dll_loaded; and network: udp, command_line, domain, tcp. With these parameters, the accuracy results for training are good and go from 63.18% to 99.73%. However, using this partial set of parameters, these algorithms have a significantly

lower performance in testing with variants not present in the training set, with a higher accuracy at a value of 54% for gradient boosted trees algorithms.

Therefore, the conclusion is that it is necessary to use the 50 chosen attributes that the researchers include in the feature dataset to ensure excellent performance in detecting ransomware variants not present in the training set. This is an essential differentiation of our work, the ability to distinguish new variants due to the combination of the generation of an input vector composed of a complete set of relevant features and the use of machine learning algorithms fed with these attributes.

4. MATERIALS AND METHODS

Our research conducts dynamic analysis using a sandbox (cuckoo). Next, we present some definitions related to this tool.

4.1. Cuckoo Sandbox

A sandbox is an isolated environment that allows the malware to be executed by implementing specific security mechanisms to guarantee the environment's integrity [106]. A sandbox allows collecting information about the behavior of the artifact executed within it. This information is later sent back to the environment where the sandbox is to analyze the recorded behavior [107]. The implementation of a sandbox varies depending on what you want to monitor [106]; however, a sandbox based on virtual machines is commonly used [107].

A virtual machine can be perceived as a computer embedded within another computer. In itself, you have a host operating system that can host one or more guest operating systems so that the guest system cannot directly affect the integrity of the host system. Multiple software solutions achieve this virtualization mode, such as VMWare Player, Virtualbox, and Microsoft Hyper-V. In addition, this type of program allows you to create snapshots which are an image of a specific virtual machine at a particular time [10]. With these snapshots, the state of a virtual machine can be restored once an artifact's execution and dynamic analysis process has finished [107].For dynamic malware analysis, it is necessary to have a base snapshot to be able to reverse all the negative effects that malicious software has caused on a virtual machine. Next, the flow of the analysis of a software artifact with the use of a sandbox is described [107].

1. The host system searches for a free sandbox in case there is more than one available.

2. The host uses the base snapshot to reset the selected sandbox to its initial state and starts it.

3. The host establishes a communication channel with the sandbox to monitor and exchange information.

4. The artifact is transferred to the sandbox by the host system and executed.

5. The host uses multiple tools to monitor and record any activity or change within the sandbox at the network level, file system, and operating system, among others.

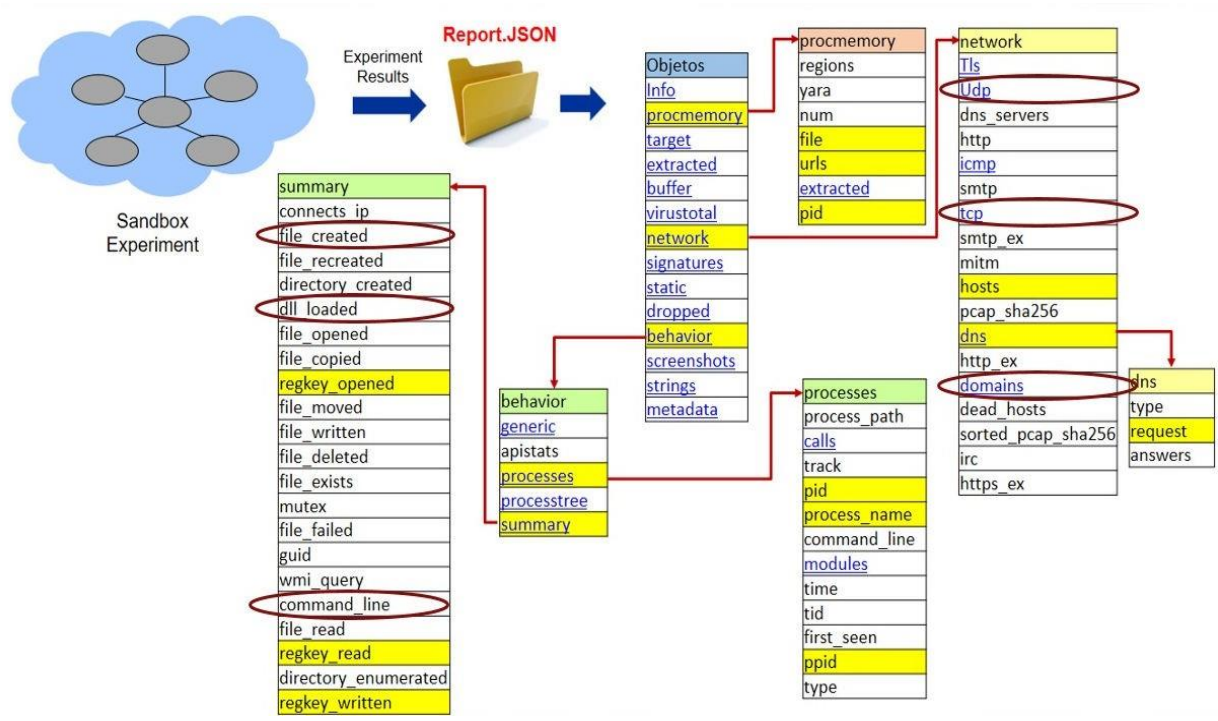
6. The host proceeds to save all the information collected from the execution of the artifact in the sandbox into one or multiple files for later review.

For this analysis process to be successful, the sandbox must be as similar as possible to a standard user's computer. Otherwise, the artifact may detect that it is being analyzed and may not run [107].

4.2. Feature Extraction Tool

Sampling artifacts (Goodware and Ransomware) and running tests on the Cuckoo Sandbox system allowed the creation of a folder containing reports of the different analyses. Figure 8 shows the general structure of the JSON reports generated in the cuckoo sandbox [108].

A report has a tree-based structure. An application to select features in the different levels was developed. For example, the features marked with yellow were chosen in the first stage. The first level contains several categories such as 'Info,' 'procmemory.' To begin the extraction process, the application visualizes the type of data stored in each category. The JSON Cuckoo Sandbox reports are recursively loaded since there were nested directories, and the program looks up every json file contained in a given directory.



Author: Juan A. Herrera Silva

Figure 8. General structure of the JSON reports

Figure 9 presents the GUI of the extraction tool used to generate the input vector for the machine learning algorithms, with the final 50 features:

Select a directory button will make a dialog box appear so that we can select the directory where our JSON reports reside.

Family and Artifact checkbox: This option allows us to obtain the 'Family' and 'Artifact' columns in our dataset. We must have a specific directory hierarchy to do this.

Select Features checkbox trees: A series of checkboxes we can use to define which features we want to extract from the JSON reports.

Extraction Method button group: To select the extraction method to use.

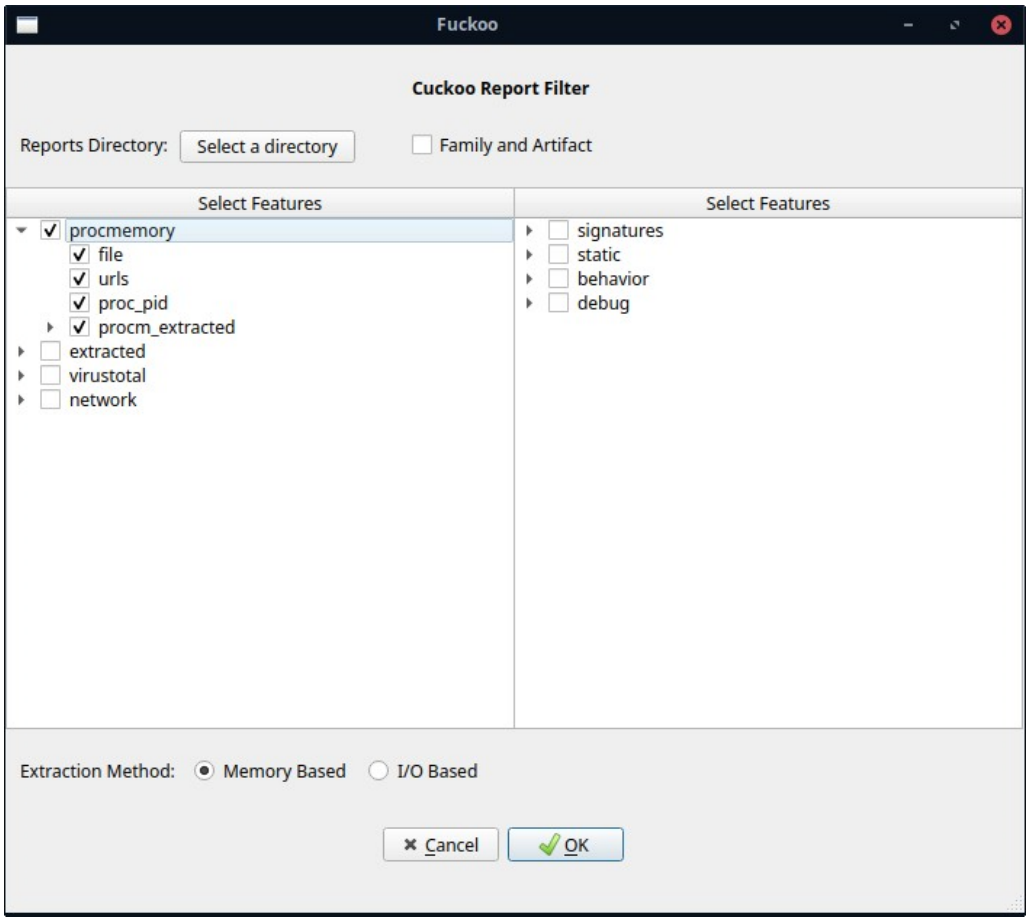


Figure 9. GUI components of the feature extraction application.

For instance, the 'network' category contains features like 'hosts' and 'dns.' 'dns' includes the 'request' feature. The program extracts all data collected in these features and writes

the data contained in a list of any primitive data type or a list of dictionaries to a CSV extension file (Figure 10). This file is the feature vector inputs in the machine learning algorithms to generate models to detect locker ransomware, encryptor ransomware, or goodware. Annex A contains more information about this application. Annex B presents examples of its use.

artifact	family	requests	udp	hosts	tcp	domains	regkey_read
Ryuk	L	slscr.update.micros	{'src': '192.168.56.50', 'dst': '192.168.56.255', 'of	172.217.2.78	{'src': '172.217.8.131', 'dst':	{'ip': '', 'domain': 'DESKTOP-49GRPRH	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Winc
Ryuk	L	www.bing.com	{'src': '192.168.56.50', 'dst': '192.168.56.255', 'of	172.217.8.131	{'src': '192.168.56.50', 'dst':	{'ip': '52.191.219.104', 'domain': 'sett	HKEY_LOCAL_MACHINE\SOFTWARE\Classes\WOW6
Ryuk	L	r2---sn-jou-0pve7.gv	{'src': '192.168.56.50', 'dst': '224.0.0.251', 'offset	13.107.246.13	{'src': '192.168.56.50', 'dst':	{'ip': '172.217.3.131', 'domain': 'upda	HKEY_LOCAL_MACHINE\SOFTWARE\Classes\WOW6
Ryuk	L	pti.store.microsoft.c	{'src': '192.168.56.50', 'dst': '224.0.0.252', 'offset	13.107.4.52	{'src': '192.168.56.50', 'dst':	{'ip': '168.61.161.212', 'domain': 'wat	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Winc
Ryuk	L	DESKTOP-49GRPRH.	{'src': '192.168.56.50', 'dst': '224.0.0.252', 'offset	13.88.21.125	{'src': '192.168.56.50', 'dst':	{'ip': '204.79.197.200', 'domain': 'ww	HKEY_LOCAL_MACHINE\SOFTWARE\Classes\WOW6
Ryuk	L	client.wns.windows.	{'src': '192.168.56.50', 'dst': '239.255.255.250', 'c	142.250.64.206	{'src': '192.168.56.50', 'dst':	{'ip': '13.107.246.13', 'domain': 'pti.s	HKEY_CURRENT_USER\Software\Microsoft\Window:
Ryuk	L	dns.msftncsi.com	{'src': '192.168.56.50', 'dst': '239.255.255.250', 'c	172.217.8.99	{'src': '192.168.56.50', 'dst':	{'ip': '13.107.4.52', 'domain': 'www.n	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Winc
Ryuk	L	ctldl.windowsupdat	{'src': '192.168.56.50', 'dst': '239.255.255.250', 'c	191.232.139.2	{'src': '192.168.56.50', 'dst':	{'ip': '52.242.101.226', 'domain': 'slsc	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Winc
Ryuk	L	go.microsoft.com	{'src': '192.168.56.50', 'dst': '239.255.255.250', 'c	192.16.58.8	{'src': '192.168.56.50', 'dst':	{'ip': '40.125.122.151', 'domain': 'fe3	HKEY_LOCAL_MACHINE\SOFTWARE\Classes\WOW6
Ryuk	L	clients2.google.com	{'src': '192.168.56.50', 'dst': '8.8.8.8', 'offset': 29	204.79.197.200	{'src': '192.168.56.50', 'dst':	{'ip': '40.126.0.67', 'domain': 'login.li	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Winc
Ryuk	L	settings-win.data.m	{'src': '192.168.56.50', 'dst': '8.8.8.8', 'offset': 29	205.185.216.10	{'src': '192.168.56.50', 'dst':	{'ip': '23.47.69.106', 'domain': 'www.	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Winc
Ryuk	L	displaycatalog.mp.m	{'src': '192.168.56.50', 'dst': '8.8.8.8', 'offset': 29	216.58.192.46	{'src': '192.168.56.50', 'dst':	{'ip': '142.250.64.206', 'domain': 'clie	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Winc
Ryuk	L	redirector.gvt1.com	{'src': '192.168.56.50', 'dst': '8.8.8.8', 'offset': 29	23.14.81.129	{'src': '192.168.56.50', 'dst':	{'ip': '131.107.255.255', 'domain': 'dr	HKEY_LOCAL_MACHINE\SOFTWARE\Classes\WOW6
Ryuk	L	www.msftconnectte	{'src': '192.168.56.50', 'dst': '8.8.8.8', 'offset': 29	23.47.68.94	{'src': '192.168.56.50', 'dst':	{'ip': '201.219.34.141', 'domain': 'r2--	HKEY_LOCAL_MACHINE\SOFTWARE\Classes\WOW6
Ryuk	L	fe3cr.delivery.mp.m	{'src': '192.168.56.50', 'dst': '8.8.8.8', 'offset': 29	23.47.69.106	{'src': '192.168.56.50', 'dst':	{'ip': '23.78.97.156', 'domain': 'go.mi	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Winc
Ryuk	L	www.microsoft.com	{'src': '192.168.56.50', 'dst': '8.8.8.8', 'offset': 29	40.125.122.176	{'src': '192.168.56.50', 'dst':	{'ip': '192.16.58.8', 'domain': 'ocsp.di	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Winc
Ryuk	L	update.googleapis.c	{'src': '192.168.56.50', 'dst': '8.8.8.8', 'offset': 29	40.126.5.36	{'src': '192.168.56.50', 'dst':	{'ip': '172.217.0.174', 'domain': 'redir	HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Con
Ryuk	L	fs.microsoft.com	{'src': '192.168.56.50', 'dst': '8.8.8.8', 'offset': 29	40.88.32.150	{'src': '192.168.56.50', 'dst':	{'ip': '52.254.96.93', 'domain': 'displa	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Winc
Ryuk	L	ocsp.digicert.com	{'src': '192.168.56.50', 'dst': '8.8.8.8', 'offset': 29	52.147.198.201	{'src': '192.168.56.50', 'dst':	{'ip': '192.16.49.143', 'domain': 'ctldl.	HKEY_LOCAL_MACHINE\SOFTWARE\Classes\WOW6
Ryuk	L	watson.telemetry.m	{'src': '192.168.56.50', 'dst': '8.8.8.8', 'offset': 29	52.167.249.196	{'src': '192.168.56.50', 'dst':	{'ip': '23.47.68.94', 'domain': 'fs.micr	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Winc
Ryuk	L	login.live.com	{'src': '192.168.56.50', 'dst': '8.8.8.8', 'offset': 29	52.177.165.30	{'src': '192.168.56.50', 'dst':	N/A	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Winc
Ryuk	L	N/A	{'src': '192.168.56.50', 'dst': '8.8.8.8', 'offset': 29	52.191.219.104	{'src': '192.168.56.50', 'dst':	N/A	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Winc
Ryuk	L	N/A	{'src': '192.168.56.50', 'dst': '8.8.8.8', 'offset': 29	52.251.11.100	{'src': '192.168.56.50', 'dst':	N/A	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Winc
Ryuk	L	N/A	{'src': '192.168.56.50', 'dst': '8.8.8.8', 'offset': 29	8.8.8.8	{'src': '192.168.56.50', 'dst':	N/A	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Winc
Ryuk	L	N/A	{'src': '192.168.56.50', 'dst': '8.8.8.8', 'offset': 29	201.219.34.141	{'src': '192.168.56.50', 'dst':	N/A	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Winc
Ryuk	L	N/A	{'src': '192.168.56.50', 'dst': '8.8.8.8', 'offset': 29	52.177.166.224	{'src': '192.168.56.50', 'dst':	N/A	HKEY_CURRENT_USER\Software\Microsoft\Window:
Ryuk	L	N/A	{'src': '192.168.56.50', 'dst': '8.8.8.8', 'offset': 29	N/A	{'src': '192.168.56.50', 'dst':	N/A	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Winc
Ryuk	L	N/A	{'src': '192.168.56.50', 'dst': '8.8.8.8', 'offset': 29	N/A	{'src': '192.168.56.50', 'dst':	N/A	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Winc
Ryuk	L	N/A	{'src': '192.168.56.50', 'dst': '8.8.8.8', 'offset': 29	N/A	{'src': '192.168.56.50', 'dst':	N/A	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Winc
Ryuk	L	N/A	{'src': '192.168.56.50', 'dst': '8.8.8.8', 'offset': 29	N/A	{'src': '192.168.56.50', 'dst':	N/A	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Winc
Ryuk	L	N/A	{'src': '192.168.56.50', 'dst': '8.8.8.8', 'offset': 29	N/A	{'src': '192.168.56.50', 'dst':	N/A	HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Winc
Ryuk	L	N/A	{'src': '192.168.56.50', 'dst': '8.8.8.8', 'offset': 29	N/A	{'src': '192.168.56.50', 'dst':	N/A	HKEY_LOCAL_MACHINE\SOFTWARE\Classes\WOW6
Ryuk	L	N/A	{'src': '192.168.56.50', 'dst': '8.8.8.8', 'offset': 29	N/A	{'src': '192.168.56.50', 'dst':	N/A	HKEY_LOCAL_MACHINE\SOFTWARE\Classes\WOW6
Ryuk	L	N/A	{'src': '192.168.56.50', 'dst': '8.8.8.8', 'offset': 29	N/A	{'src': '192.168.56.50', 'dst':	N/A	HKEY_LOCAL_MACHINE\SOFTWARE\Classes\WOW6

Author: Juan A. Herrera Silva

Figure 10. CSV file with extracted features

4.4. Phases of Experimentation

In this research, we conduct experimentation in three different phases as part of the quantitative - experimental research. Table 5 describes these different stages. Phase previous allowed generating learning models with a somewhat unrealistic prediction value, reaching 100% accuracy because it was a one-class classification that did not include goodware; this phase used a dataset generated only with ransomware artifacts and seven features.

Table 5. Description of the experimental phases.

Phases	Ransomware	Goodware	Features	Dataset (rows)	Dataset (columns)	# Sandboxing experiments
Phase previous (0)	5	-	8	6.783	10	100
Phase initial (1)	10	10	8	47.959	10	380
Phase analysis (2)	10	10	5 (group of features)	62.989	16	380
Phase final (3)	20	20	50	1.424.344	50	2000

Table 6 describes the characteristics of the objects that could be selected as features for the input vector that will be fed to the classification algorithm. A detailed description of these features can be found in Annex C. It was noticed that these predictive models would not be valid because the dataset was not balanced, and there was a bias in the classification of the only one majority class. However, this was a good first approximation for generating a ransomware detection dataset.

Table 6. Feature description

Object	Description	Feature	Explanation	Reason for Choosing the Feature
PROCMEMORY	It allows the creation of memory dumps for each analyzed process (before they finish or before the analysis ends).	File	File created as a memory dump	The feature is chosen because this information allows memory forensics monitoring file modifications to find an unusual increase in particular extensions.
		URLs	URLs generated during the execution of memory processes	The feature is chosen because it stores a list of URLs that can be modeled as suspicious.
		PID	Process identifier	The feature is chosen because it identifies the generated file (file).
		name	Name of the process in memory	The feature is chosen because it identifies the name of a possible suspicious process.
		types	Artifact type	The feature is chosen because it identifies the type of artifact.
		URLs	URLs used by the process in memory	The feature is chosen because it identifies URLs used in memory by the process.
		path	Memory process storage directory	The feature is chosen because it identifies the directory.
EXTRACTED	It contains information about scripts executed by an artifact during artifact analysis.	info	Information of the script in question	The feature is chosen because it identifies information about scripts that could be used during attacks.
		program	Type of program executing the script	The feature is chosen because it identifies the program that executes possible malicious scripts.
NETWORK	Includes information on the network infrastructure used during the analyses	dns_servers	DNS servers involved in the analysis	The feature is chosen due to communication with external domain servers. DNS sub-characteristics (request).
		mitm	Network analysis to verify the type of attacks man-in-the-middle	The feature is chosen because it identifies attacks man-in-the-middle where a perpetrator is positioned in an exchange between a user and an application.
		dead_hosts	Hosts down during data transmission	The feature is chosen because it identifies hosts down, which could be one of the effects of ransomware.
		udp	network analysis of the udp protocol	The feature is chosen due to the use of communication via UDP protocol. It corresponds to the udp port number that ransomware could open.
		tcp	network analysis of the tcp protocol	The feature is chosen due to the use of communication via TCP protocol. It corresponds to the tcp port number that ransomware could open.

Object	Description	Feature	Explanation	Reason for Choosing the Feature
		hosts	hosts involved in the analysis. Help create blacklists	The feature is chosen because of the communication with a malicious host. With this information, we can create blacklists.
		domain	Domains involved in communication	The feature is chosen because communication with other domains may be a clue for identifying ransomware.
		request	Domains to which requests were sent (queries) DNS	The feature is chosen because it serves to monitor possible suspicious requests.
SIGNATURES	It contains information about tasks or processes before, during, and after the analysis and the API calls executed by the analyzed artifact.	families	A list of malware family names	The feature is chosen because it identifies requests that were sent.
		description	Signature Description	The feature is chosen because it supplements information about possible ransomware.
		name	Signature name	The feature is chosen because it supplements information about possible ransomware.
		category	API call category	The feature is chosen because it supplements information about possible ransomware. The category of the API calls can be used to model the application behavior.
		stacktrace	Execution stack related to a api call	The feature is chosen because it supplements information about possible ransomware. The stacktrace of the API calls can be used to model the application behavior.
		api	API call in question	The feature is chosen because it supplements information about possible ransomware. Some characteristics of the API calls can be used to model the application behavior.
		arguments	Arguments of the API call in question	The feature is chosen because it supplements information about possible ransomware. Arguments of the API call can be used to model the application behavior.
STATIC	Contains information about a static analysis performed by Cuckoo in case the analyzed artifact is of type Portable Executable (PE).	imported_dll_count	Number of system DLLs imported by artifact	The feature is chosen because it contains artifact information when the artifact is portable executable.
		dll	System DLL libraries used by the artifact during analysis	The feature is chosen because it contains artifact information when the artifact is portable executable.
		name	artifact name	The feature is chosen because it contains artifact information when the artifact is portable executable.

Object	Description	Feature	Explanation	Reason for Choosing the Feature
		filetype	artifact type	The feature is chosen because it contains artifact information when the artifact is portable executable.
		entropy	Entropy level of the artifact in question	Encryption changes the content. Therefore, it has a higher entropy value. This characteristic could help to detect encryption and ransomware; thus, it was selected.
		name	Sections found within the artifact	The feature is chosen because it contains artifact information when the artifact is portable executable.
BEHAVIOR	It allows seeing the behavior of ransomware, that is, to see the processes that the ransomware performs, libraries to which it makes calls, registry keys that affect	Processes	Processes carried out by the device	The feature is chosen because processes modify the infected system. The authors selected sub-characteristics processes (process_path, pid, process_name, command_line, and ppid).
		Processtree	executed child processes derived from the process tree	The feature is chosen because processtree contains subprocesses that modify the infected system. The authors selected sub-characteristics processtree (process_name, command_line, and children).
		Summary	Summary of files, log keys, directories, and commands involved during the execution of processes	The feature is chosen because it contains parameters that affect infected systems. The sub-characteristics summary (regKeys) is chosen because register values are modified during a ransomware attack. In addition, the sub-characteristics (file_created, dll_loaded, wmi_query, command_line, file_read, and directory_enumerated) are chosen because ransomware uses these function calls to execute malicious operations in the OS_file system.
DEBUG	It contains information about the analysis performed on an artifact.	action	Actions recorded during analysis	This feature was selected because it gives information about the Cuckoo and its actions during the experiments' execution.
		errors	Errors logged during analysis	This feature was selected because it gives information about the Cuckoo and its actions during the experiments' execution.
		log	Various information about the analysis executed	This feature is selected because it gives information about all the occurrences inside the cuckoo sandbox during the experiments' execution.

Annex C describes in detail the main objects such as info, procmemory, target, extracted, buffer, network, signatures, static, dropped, behavior, debug, and their respective features, which are present in a .json file. There is a total of 326 features, of which a sweep of all of them was made, reviewing their behavior, what they represented and their main occurrence in the JSON file. The main features that were selected for their Ransomware behavior in the previous, initial and analysis phases are highlighted in yellow. For the final phase, the features highlighted in light blue and purple were selected, giving a total of 64 features selected with an engineering procedure. After which, an automatic feature selection method was used, which is the Mutual Information Matrix and it can be seen which features have a high correlation with each other and these features that are more correlated are the ones that can be removed for selection. So, using this criterion we removed 14 features and worked with the final 50 features for modeling, since they have to do with the behavior of ransomware and have relevance because they are not redundant, since they are not related to others.

Phases Initial and Analysis

In phase one, it was considered the incorporation of more ransomware artifacts and, at the same time, the use of goodware to balance the dataset. We used the same seven features to obtain new predictive models, with the applied algorithms observed that the models did not exceed 85% accuracy.

During experimentation, Phase analysis was developed to extend the dataset towards accuracy improvement. It had the same amount of Ransomware and goodware artifacts, and it had six more features depending on the analysis of the MITRE ATT&CK matrix.

It should be emphasized that in the analysis of the .json files, a set of features was chosen in two levels, starting with the superior one in the .json files. From the first stage analysis, the conclusion is that the regkey_read feature most affects the system due to its density in the .json file's ransomware logs.

The same criteria were chosen, and the UDP and file_created characteristics were selected. During ransomware communication to the compromised system to identify the source and destination IP and source and destination ports, it was noted that UDP helps. The file_created feature supports identifying files created by ransomware during the infection process.

With the described background, the machine learning algorithms generated the models using three characteristics grouped into one. They used five combinations to input to the classification algorithms to achieve detection and prediction of the ransomware and goodware artifacts. Table 7 describes the objects taken from the "report.json" file obtained from Cuckoo Sandbox and the description and importance of each parameter.

The file .json consists of 15 objects, and in each item, there are features that, in turn, contain other nested data. The objects analyzed are three: proc memory, network, and behavior. We extract the characteristics of interest from these files using a developed application that allows us to obtain the nested values inside the files according to feature selection criteria. This application is introduced in section 4.2 and explained in detail in Annex A and B.

The methodological approach applied mainly for phases initial and analysis is explained below to describe the dataset generation.

Table 7. Description of dataset column (features Phase initial and analysis).

Identifier	Description	Phase
ARTIFACT	Identifiers associated with ransomware samples, for example, 1 for 7-Zip	Initial and analysis
FAMILY	Identifiers associated with the type of ransomware or goodware, i.e., locker or encryptor. E for encryptor, L para locker, and G for goodware.	Initial and analysis
REGWRITE	An identifier associated with written log keys. It has got two zeros ahead. Example: 00100	Initial and analysis
REGOPEN	An identifier related to open registry keys. It has got three zeros ahead. Example: 000100	Initial and analysis
REGREAD	An identifier associated with reading log keys. It has got four zeros ahead. Example: 0000100	Initial and analysis
PROC	Identifier associated with tit set formed by pid, process_name, and ppid. It has got a zero in front. Example: 0100	Initial and analysis
PMFILES	Identifier associated with generated memory dump files. It takes into account PID and file as a set. It has five zeros in front. Example: 00000100	Initial and analysis
PMURLS	Identifier associated with URLs generated in memory. It has got six zeros ahead. Example 000000100	Initial and analysis
NETHOSTS	Identifier associated with the host involved. It has got seven zeros ahead. Example: 0000000100	Initial and analysis
NETREQUESTS	Identifier associated with domains to which requests are made. It has got eight zeros ahead. Example 00000000100	Initial and analysis
FILECREATED	Identifier associated with different files created (tmp, ini, bat, among other types) with which it performs the infection and propagation. It has nine zeros ahead. Example: 000000000100	Analysis

Identifier	Description	Phase
DLLLOADED	Identifier associated with dlls loaded by the device. It has ten zeros ahead. Example: 0000000000100	Analysis
COMMANDLINE	Identifier associated with commands that could execute the artifact (Power Shell, CMD). It has 11 zeros ahead. Example: 00000000000100	Analysis
DOMAIN	Identifier associated with domains and IP addresses with which the device communicates. It has 12 zeros ahead. Example: 000000000000100	Analysis
TCP	Identifier associated with the tcp protocol used by the device. It has 13 zeros ahead. Example: 0000000000000100	Analysis
UDP	Identifier associated with the udp protocol used by the device. It has 14 zeros ahead. Example: 00000000000000100	Analysis

The features are chosen to obtain the best classification performance. We use feature engineering to extract features that provide enough information about the goodware and the ransomware. Characteristics with redundant information are not considered, and features that appear not to influence the results are not considered.

For Phase initial, it was considered the "DNS" features of the "network" object that contains sub-characteristics. "Request" was considered because it allows viewing domain names during a system's infection. In the case of the object "behavior," the characteristics "processes" and "summary" also contain sub-characteristics. The feature "Processes" includes "pid" (represents the process identifier), "process_name" (represents the process name), and "ppid" (represents the parent process identifier). The "summary" feature, "regkey_opened" (open registry keys), "regkey_read" (read registry keys) and "regkey_written" (written registry keys) were also considered.

On the other hand, in Phase analysis, it was added the "domain," "tcp," and "udp" features of the "network" object, which identify communications in the network. Additionally, the sub-characteristics mainly involved in the behavior of ransomware "file_created," "dll_loaded," and "command_line" were chosen.

With these considerations, 380 of the total experiments generated the corresponding json files to obtain the final dataset. The mentioned features emulate the different ransomware artifacts' behavior in an isolated and controlled testing environment, which is useful for constructing the dataset. A set of identifiers for the dataset are described in Table 8. Each identifier can, in turn, identify a characteristic or set of previously selected parameters. It is necessary to clarify that the relationship in the dataset between the artifact (ransomware sample) and the different parameters was obtained from our analysis.

Phase Final – Selected Features

With the background of phases from previous to analysis, a third phase was developed in which several datasets were generated (Phase final). To generate the dataset, 2000 experiments were performed with 20 ransomware samples and 20 Goodware samples. Characteristics were selected if they were affected during the infection process. Other characteristics were also selected that affected the infection process. These characteristics are reflected in Table 7. The identifiers are assigned depending on the number of times that features have been counted, that is, integer values starting with 0 when not there are records and from 1 onwards when there are records.

Table 8. Description of the dataset features for Phase final.

Identifier	Description	Phase
family	Identifiers associated with the type of ransomware or goodware, locker or encryptor. E for encryptor, L for locker, and G for goodware.	Final
proc_pid	Identifier associated with the process identifier. Integer values are assigned starting at 0 if no record exists	Final
file	Identifier associated with the file's name created as a memory dump of the analyzed artifact. Integer values are assigned starting at 0 if no record exists	Final
urls	Identifier associated with URLs found during the core dump process. Integer values are assigned starting at 0 if no record exists	Final
type	Identifier associated with the artifact type. Integer values are assigned starting at 0 if no record exists	Final
name	Identifier associated with the name of the process in memory. Integer values are assigned starting at 0 if no record exists	Final
ext_urls	Identifier associated with URLs used by the process in memory. Integer values are assigned starting at 0 if no record exists	Final
path	Identifier associated with the storage directory of the memory process. Integer values are assigned starting at 0 if no record exists	Final
program	Identifier associated with the type of program that executes the script. Integer values are assigned starting at 0 if no record exists	Final
info	Identifier associated with the Information of the script that executes a program. Integer values are assigned starting at 0 if no record exists	Final
families	Identifier associated with a list of malware family names. Integer values are assigned starting at 0 if no record exists	Final
description	Identifier associated with the signature description. Integer values are assigned starting at 0 if no record exists	Final
sign_name	Identifier associated with the name of the firm. Integer values are assigned starting at 0 if no record exists	Final

Identifier	Description	Phase
sign_stacktrace	Identifier associated with the execution stack related to an API call. Integer values are assigned starting at 0 if no record exists	Final
arguments	Identifier associated with arguments of the API call. Integer values are assigned starting at 0 if no record exists	Final
api	Identifier associated with the API call. Integer values are assigned starting at 0 if no record exists	Final
category	Identifier associated with the category of the API call. Integer values are assigned starting at 0 if no record exists	Final
imported_dll_count	Identifier associated with the number of system DLLs imported by the artifact. Integer values are assigned starting at 0 if no record exists	Final
dll	Identifier associated with system DLL libraries used by the artifact during analysis. Integer values are assigned starting at 0 if no record exists	Final
pe_res_name	Identifier associated with the artifact name. Integer values are assigned starting at 0 if no record exists	Final
filetype	Identifier associated with the artifact type. Integer values are assigned starting at 0 if no record exists	Final
pe_sec_name	Identifier associated with the name of sections found within the artifact. Integer values are assigned starting at 0 if no record exists	Final
entropy	Identifier associated with the artifact's entropy level. Integer values are assigned starting at 0 if no record exists	Final
hosts	Identifier associated with the IP addresses of the Hosts involved during the analysis. Integer values are assigned starting at 0 if no record exists	Final
requests	Identifier associated with domains to which DNS requests (queries) were sent. Integer values are assigned starting at 0 if no record exists	Final
mitm	Identifier associated with network analysis to verify Man-in-the-middle type attacks. Integer values are assigned starting at 0 if no record exists	Final
domains	Identifier associated with domains with which communication was established during the analysis. Integer values are assigned starting at 0 if no record exists	Final
dns_servers	Identifier associated with DNS servers used by the artifact during analysis. Integer values are assigned starting at 0 if no record exists	Final
tcp	Identifier associated with tcp connections established during the analysis. Integer values are assigned starting at 0 if no record exists	Final
udp	Identifier associated with udp connections established during the analysis. Integer values are assigned starting at 0 if no record exists	Final
dead_hosts	Identifier associated with hosts down during data transmission. Integer values are assigned starting at 0 if no record exists	Final
proc	Identifier associated with the name of the process in memory. Integer values are assigned starting at 0 if no record exists	Final
beh_command_line	Identifier associated with commands executed during the analysis. Integer values are assigned starting at 0 if no record exists	Final

Identifier	Description	Phase
process_path	Identifier associated with the directory where the process is stored on the victim system. Integer values are assigned starting at 0 if no record exists	Final
tree_command_line	Identifier associated with commands executed during the analysis. Integer values are assigned starting at 0 if no record exists	Final
children	Identifier associated with processes initialized by the artifact. Integer values are assigned starting at 0 if no record exists	Final
tree_process_name	Identifier associated with the name of the child process. Integer values are assigned starting at 0 if no record exists	Final
command_line	Identifier associated with commands executed during the analysis. Integer values are assigned starting at 0 if no record exists	Final
regkey_read	Identifier associated with registry keys read during the scan. Integer values are assigned starting at 0 if no record exists	Final
directory_enumerated	Identifier associated with directories listed by the artifact. Integer values are assigned starting at 0 if no record exists	Final
regkey_opened	Identifier associated with registry keys opened during the scan. Integer values are assigned starting at 0 if no record exists	Final
file_created	Identifier associated with files created by the artifact. Integer values are assigned starting at 0 if no record exists	Final
wmi_query	Identifier associated with Windows Administration instrumentation queries. Integer values are assigned starting at 0 if no record exists	Final
dll_loaded	Identifier associated with DLL libraries used by the artifact. Integer values are assigned starting at 0 if no record exists	Final
regkey_written	Identifier associated with registry keys written by the artifact. Integer values are assigned starting at 0 if no record exists	Final
file_read	Identifier associated with files read during the scan. Integer values are assigned starting at 0 if no record exists	Final
apistats	Identifier associated with the accounting of each API call made during the analysis. Integer values are assigned starting at 0 if no record exists	Final
errors	Identifier associated with errors logged during analysis. Integer values are assigned starting at 0 if no record exists	Final
action	Identifier associated with actions recorded during the analysis. Integer values are assigned starting at 0 if no record exists	Final
log	Identifier associated with various information about the performed analysis. Integer values are assigned starting at 0 if no record exists	Final

A tool created to select the characteristics described in the previous table was used in making the datasets, as seen in section 4.2. When choosing the features, a total of 12 were created. Table 9 shows the number of characteristics of each dataset and some observations regarding each one.

Table 9. Datasets description

Dataset	Features	Observation
DATASET 1 6 features	udp, domains, file_created, dll_loaded, command_line, regkey_read	The reason for selecting this dataset's characteristics is that with them, the best results were obtained in the models in phases initial and analysis.
DATASET 2 7 features	udp, domains, file_created, dll_loaded, command_line, regkey_read, api	The api feature is added due to its density within the report.json files
DATASET 3 8 features	udp, domains, file_created, dll_loaded, command_line, regkey_read, api, URLs, proc	The pid, process_name, and ppid (proc) features are added to understand the behavior of the processes running during the ransomware infection. With these characteristics, the id of the processes and their name are identified. Identifier associated with URLs found during the core dump process. Integer values are assigned starting at 0 if no record exists
DATASET 4 11 features	udp, domains, file_created, dll_loaded, command_line, regkey_read, api, urls, proc, tcp, hosts, request	Se agregan las características tcp, hosts y request con el propósito de identificar puertos de comunicación por tcp, hosts involucrados y respuestas dadas durante la comunicación en la red
DATASET 5 12 features	udp, domains, file_created, dll_loaded, command_line, regkey_read, api, URLs, proc, tcp, hosts, request, children	The children feature is added to see the processes within the process tree
DATASET 6 13 features	udp, domains, file_created, dll_loaded, command_line, regkey_read, api, URLs, proc, tcp, hosts, request, children, entropy	The entropy feature was added due to its contribution to entropy
DATASET 7 14 features	udp, tcp, hosts, domains, request, proc, file_created, dll_loaded, regkey_opened, command_line, regkey_read, regkey_written	The 14 features were chosen from Phase analysis– Initial Dataset Experimentation with Rapid Minder
DATASET 8 15 features	URLs, udp, tcp, hosts, domains, request, api, dll, entropy, proc, children, file_created, dll_loaded, command_line, regkey_read	The URLs and dll features were added due to their density within the json file and also because, in previous analyzes of the models, they contributed considerably
DATASET 9 16 features	file, URLs, udp, tcp, hosts, domains, request, api, dll, entropy, proc, children, file_created, dll_loaded, command_line, regkey_read	Added file feature due to a previous analysis of models
DATASET 10 17 features	file, URLs, udp, tcp, hosts, domains, request, api, dll, entropy, proc, children, file_created, dll_loaded, command_line, regkey_read,regkey_written	Added regkey_written feature due to its density within the json file and also because previous analyzes on models added significantly
DATASET 11 25 features	file, URLs, name, program, positives, udp, tcp, hosts, domains, request, sign_name, api, dll, filetype, entropy, errors, apistats, proc, file_created, dll_loaded, command_line, regkey_read,regkey_written	Added name program positives sign_name filetype errors apistats file_read features to complement features of each main object

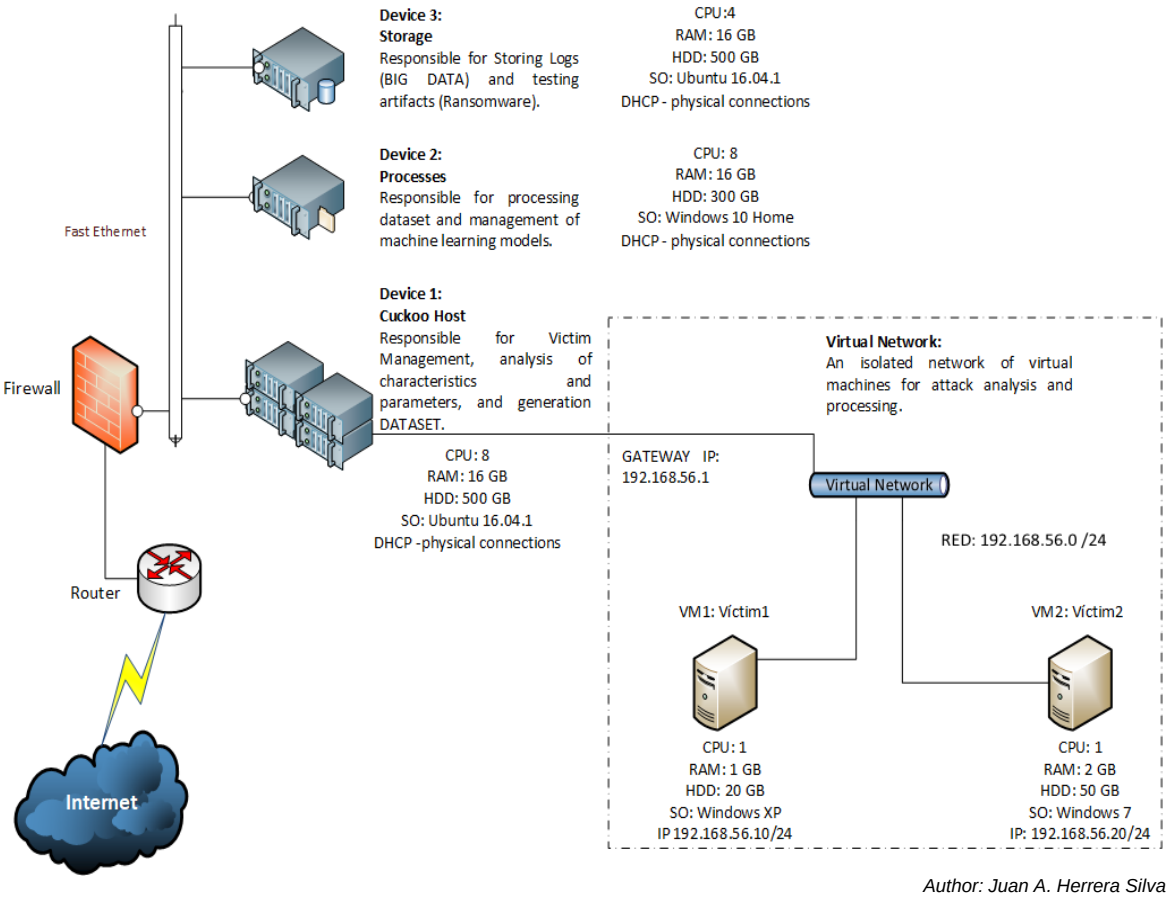
Dataset	Features	Observation
DATASET 12 50 features	All characteristics	All the characteristics were selected to know how they affect the performance of the models. SeeTable 7.

4.5. Test Setting

It was considered a test scenario in an isolated environment, allowing essential information to obtain. Then, our feature extraction tool filtered the attributes required for the dataset conformation. The deployment was based on a safe environment using the cuckoo sandbox tool [109] in the Cybersecurity Laboratory (Advanced Data Analytics).

Phases Previous, Initial and Analysis

The network topology used to generate the dataset in Phases Previous, Initial and Analysis is presented in Figure 11. This experimentation was carried out on 2 victim machines and 14 features were obtained for an initial analysis of ransomware behavior.



Author: Juan A. Herrera Silva

Figure 11. Test environment network topology for Phases Previous to Analysis

For phases initial and analysis, the following ransomware samples were considered: CryptoLocker, CryptoWall, and PetrWrap. Petya, WannaCry, Cerber, Locky, Radamant, Satana, and TeslaCrypt[13]. The goodware samples were: Windows 7, Winzip, Acrobat Reader, Chrome, Explorer, DllHost, Firefox, Services, and Svchost[14].

As part of the experiments, the report data was obtained based on Cuckoo logs (*.json, *.pcap, among others). This data generates csv files to proceed with the analysis. A total of 380 experiments are carried out, considering 20 analyses for each device. Ten scans were run for the victim using machine 1 with Windows XP, and similarly for the victim consuming the machine with Windows 7 (ten scans). Experiments on this work were conducted on the FIS-EPN Cybersecurity network, protected by a firewall and access control and service control rules. After the attack runs, the report.html and report.json files are obtained. The report.html contains summary information, and the file report.json considered the following objects: proc memory, network, and behavior. As seen in Table 9, we chose additional features to analyze each selected object.

In summary, for Phase initial and Phase experimental, the artifacts are presented in Table 10. There are 24 artifacts in total, as the ransomware applies the same artifact on Windows XP and Windows 7. The phases are due to an evolution of the systematic work that has been done. Parts of the dataset have been generated and tested in classifiers to evaluate their performance. This process has been performed in several stages to obtain the most relevant features for ransomware detection.

Table 10. Artifacts for Datasets in Phases Initial and Analysis

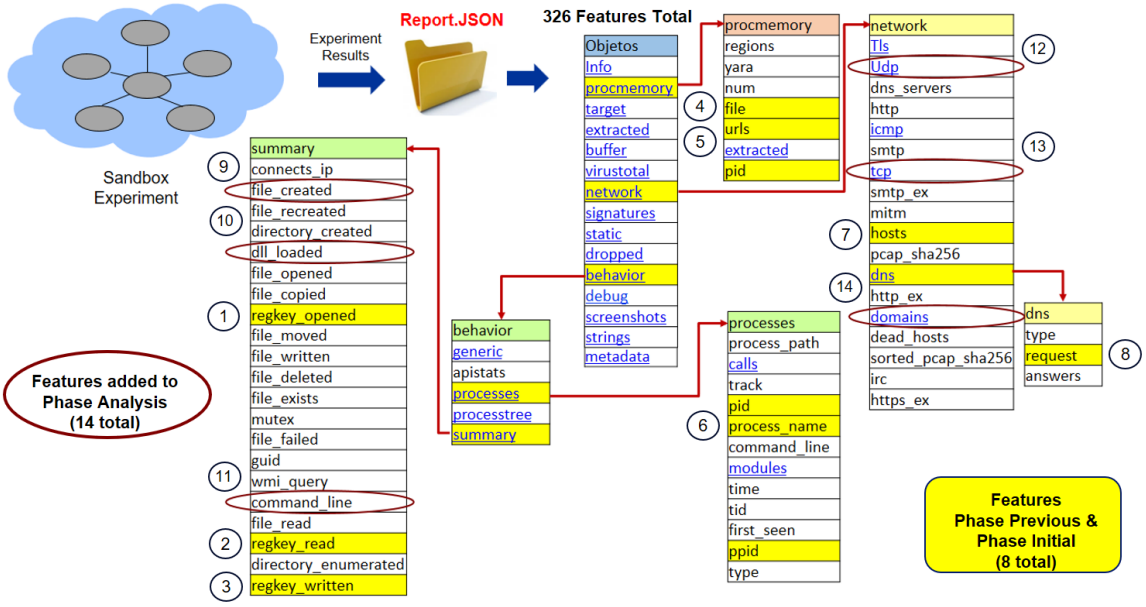
ID	Name	SHA1	MD5	Family	Experiments
1	7-zipPortable_9.20_Rev_2.paf.exe	35bcca0e8b907386ca4c7536dc55913e3c71b220	7fa4441c55a838e0691328cebde21802	G	20
2	AdbeRdr11008_es_ES.exe	aa08e431163c6129697d0aae7f4f9915bc90b2ba	3472d1522f9568534a9116400af1a1be	G	10
3	AcroRdrDC1901220036_es_ES.exe	ad998431b1ec06b2ea2087e3a2ebc65a6d23ba9e	153311a588cbbc6f45ea4401bf081fec	G	10
4	cerber.exe	c69a0f6c6f809c01db92ca658fcf1b643391a2b7	8b6bc16fd137c09a08b02bbe1bb7d670	E	20

[13] https://github.com/ytisf/theZoo/tree/master/malwares/Binaries
[14] https://www.exefiles.com/en/

ID	Name	SHA1	MD5	Family	Experiments
5	chrome.exe	04ca28f529aae1db4be4cfb4c601f57c7d08f997	da2965d0020f4156141c783ebcd64f0f	G	20
6	cryptolocker.exe	65559245709fe98052eb284577f1fd61c01ad20d	04fb36199787f2e3e2135611a38321eb	E	10
7	cryptowall.bin	ca963033b9a285b8cd0044df38146a932c838071	47363b94cee907e2b8926c1be61150c7	E	10
8	dllhost.exe	ab0af67fd000646ed231ee421e5c71798d0d86a0	0f886de058726bb6323bfd98773fad26	G	10
9	dllhost.exe	ace762c51db1908c858c898d7e0f9b36f788d2d9	a63dc5c2ea944e6657203e0c8edeaf61	G	10
10	explorer.exe	78f905f135771dec9646f6f753195adf5e7bf7c9	7522f548a84abad8fa516de5ab3931ef	G	20
11	explorer.exe	84123a3decdaa217e3588a1de59fe6cee1998004	38ae1b3c38faef56fe4907922f0385ba	G	20
12	firefox.exe	efe760ee6f516adb01e3092e78bda904df908b56	9adcb5abe8bb7e1a9355632817d23f43	G	20
13	locky	b606aaa402bfe4a15ef80165e964d384f25564e4	b06d9dd17c69ed2ae75d9e40b2631b42	E	20
14	Petrwrap.exe	34f917aaba5684fbe56d3c57d48ef2a1aa7cf06d	71b6a493388e7d0b40c83ce903bc6b04	L	20
15	petya.bin	d1c62ac62e68875085b62fa651fb17d4d7313887	a92f13f3a1b3b39833d3cc336301b713	L	20
16	radamant.ViR	05ae9c76f8f85ad2247c06d26a88bbbcfff4d62e	6152709e741c4d5a5d793d35817b4c3d	E	20
17	satana.bin	5b063298bbd1670b4d39e1baef67f854b8dcba9d	46bfd4f1d581d7c0121d2b19a005d3df	L	20
18	services.exe	7cf0d257861a23191a9d482a51783593d6a64f74	d658a8c2fc7b2ad53d1259741a09ee04	G	10
19	services.exe	ff658a36899e43fec3966d608b4aa4472de7a378	71c85477df9347fe8e7bc55768473fca	G	10
20	svchost.exe	1aae36311da414c8fd5b32956aaed1d82237ab08	4f2340f0bd5b6365c38e74dd391919a8	G	10
21	svchost.exe	4af001b3c3816b860660cf2de2c0fd3c1dfb4878	54a47f6b5e09a77e61649109c6a08866	G	10
22	teslacrypt	51b4ef5dc9d26b7a26e214cee90598631e2eaa67	6e080aa085293bb9fbdcc9015337d309	E	20

ID	Name	SHA1	MD5	Family	Experiments
23	wannacry.exe	5ff465afaabcbf0150d1a3a b2c2e74f3a4426467	84c82835a5d21bbcf 75a61706d8ab549	E	20
24	WinRAR.EXE	0d95c17831e9cd4d0d7efb 9efa866437eed186fd	b78d7b5d2fcbe117 1a3500cc2176f9c9	G	20
				TOTAL	380

Figure 12 shows the characteristics (highlighted in yellow) to generate the Phase initial; dataset and additional features (enclosed in red ellipses) to produce the Phase analysis dataset. It shows a subset of the data obtained from the reports and includes a list and quantification of the characteristics analyzed. Table 11 lists the selected characteristics for Phases initial (1) and analysis (2), with an observation explaining the selection criterion.



Author: Juan A. Herrera Silva

Figure 12. Features analyzed for the dataset using Cuckoo Sandbox in Phases Previous, Initial and Analysis

Table 11. Selected characteristics Phases Initial and Analysis

Object	Feature	Criterion
Behavior	regkey_opened	This feature was taken because of the changes they make in the OS. Phase initial.
Behavior	regkey_read	This feature was taken because of the changes they make in the OS. Phase initial.
Behavior	regkey_written	This feature was taken because of the changes they make in the OS. Phase initial.
Behavior	processes	This feature was taken because of the processes running on the OS. Phase initial.
Procmemory	files	This feature was taken because of files created by memory processes. Phase initial.
Procmemory	URLs	This feature was taken due to URLs created by memory processes. Phase initial.
Network	hosts	This feature was taken due to the communication of hosts involved. Phase initial.
Network	request	This feature was taken due to communication to domain servers (requests). Phase initial.
Behavior	file_created	This feature was taken because of the files that are created by the artifact in the OS. Phase Analysis.
Behavior	dll_loaded	This feature was taken because of the dlls that load the artifact during its execution. Phase Analysis.
Behavior	command_line	This feature was taken because of the commands the artifact uses. Phase Analysis.
Network	domains	This feature was taken because of the domains involved in communication. Phase Analysis.
Network	tcp	This feature was taken due to network analysis of the tcp protocol. Phase Analysis.
Network	udp	This feature was taken due to network analysis of the udp protocol. Phase Analysis.

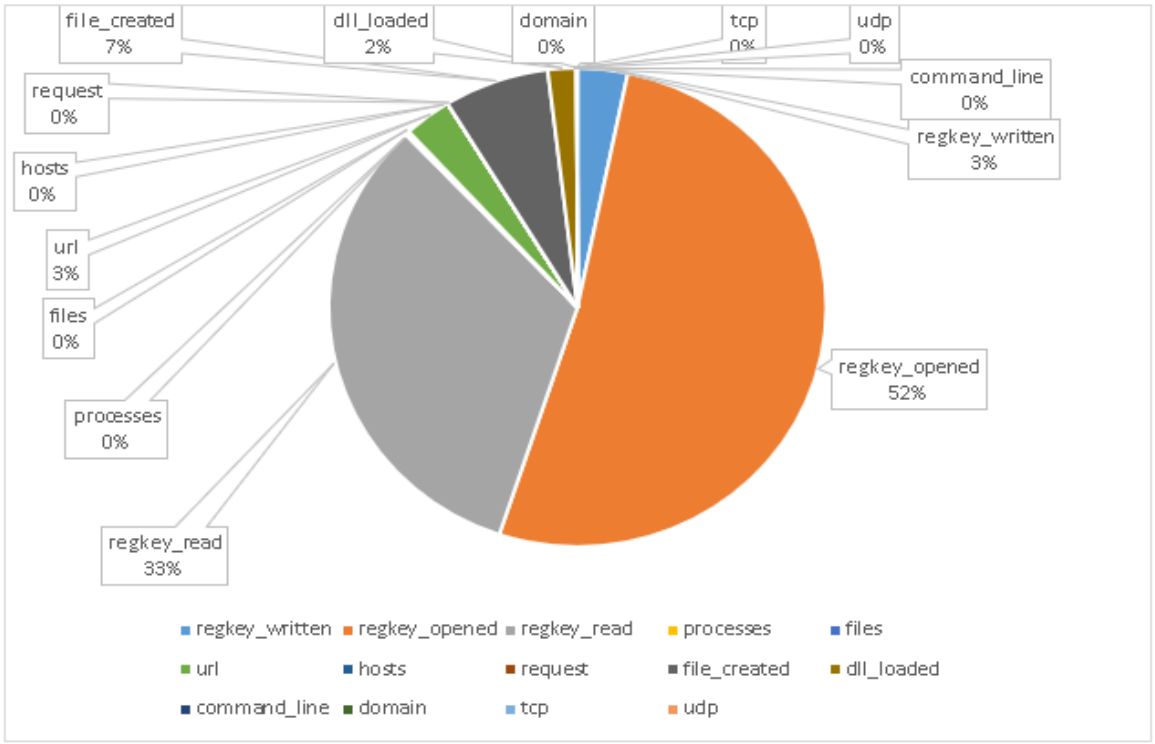
This study determined a set of candidate characteristics to elaborate the required dataset from the original information obtained and carried out (without processing) data filtering. The features taken into account come from different objects: procmemory, behavior, and network. In turn, these objects contain a set of characteristics and nests of the same so-called sub-characteristics.

The object "procmemory" is characterized by creating logs about modifications in the memory of infected devices and considering features such as memory dump files, URLs, processes executed in memory, and memory regions affected among the most relevant. This object's analysis sets the maximum and minimum number of memory dump files (dmp files) with their process identifiers and the five artifacts used in the experiments.

We determined with this data that the ransomware that most generates dump files is WannaCry. Simultaneously, it was observed that large quantities of URLs were also stored with the same ransomware. The object called “network” characterizes ransomware’s behavior concerning communication in the network (requests and protocols involved, servers, domain names, and hosts that interact in communication). Because of this object's analysis, the maximum and minimum number of IPs, domain names, and requests have been made to set them. The attributes considered are hosts (IPs of hosts involved) and claims (domain names of requests).

The "behavior" object characterizes ransomware behavior (triggered processes, library calls, and invoked registry keys, among others). As a result of this object’s analysis, it was set a maximum and a minimum number of affected records and processes executed by the five types of ransomware samples. The attributes considered are open, read, and written processes and registers.

The research gets the most representative average ransomware occurrence with the selected features with ransomware and goodware artifacts as shown in Annex D. You can see Figure 13 for the representation of average characteristics for ransomware as obtained in the experiments for Phase analysis that produce this data.



Author: Juan A. Herrera Silva

Figure 13. Average characteristics

WannaCry ransomware affects victim systems in processes that involve registry keys. The lists of the chosen features allow forming the dataset from identifiers of each list. The dataset associates artifacts with processes, registry keys, memory dump files, URLs stored in memory, and IPs of hosts involved in communication during ransomware attacks and domain names. The details of each parameter (Phase initial y Phase analysis) and all Annexes are publicly available at the following link:

https://drive.google.com/open?id=1vgOi2jchr_a0HrRhK1KOa6_UjuaczMaf.

Phase Final

Table 12 lists the 40 artifacts used in Phase Final: goodware, encryptor, and locker. In this table are also some observations about the behavior of goodware that could lead to an erroneous detection as malware because they have behaviors similar to Ransomware. The selected Ransomware is highlighted in yellow in Fig. 3. Ransomware evolution timeline.

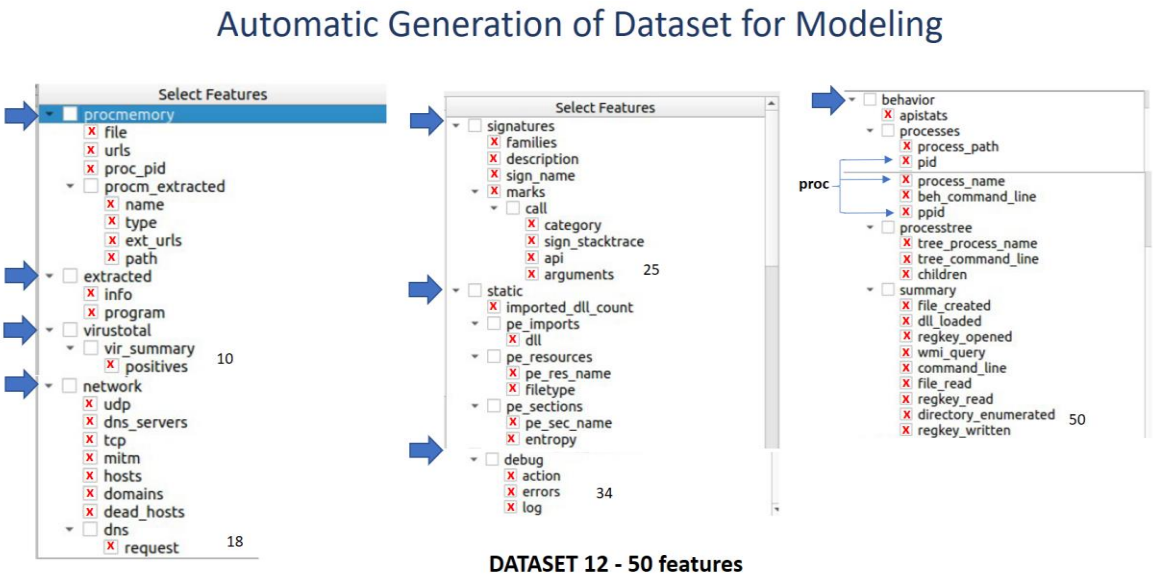
Table 12. Artifacts used in Phase final

ID	Artifact	Family	Comments
1	7Zip	Goodware	This goodware was selected due to its behavior related to file encryption.
2	Task Manager (taskmgr)	Goodware	This goodware was selected due to its access to process and system tasks.
3	API WINDOWS SECURITY CRYPTOGRAPHY (cipher)	Goodware	This goodware was selected due to its operating system file encryption behavior.
4	API WINDOWS SYSTEM INFORMATION REGISTRY (regedit)	Goodware	This goodware was selected due to its interactions with the registry keys.
5	API WINDOWS VOLUME MANAGEMENT (diskpart)	Goodware	This goodware was selected due to its access to disk volumes and partitions.
6	Bitlocker	Goodware	This goodware was selected due to its ability to encrypt disks and directories.
7	BitPaymer	Encryptor	BitPaymer allows cybercriminals to carry out a ransomware and data theft attack at the same time as it has a feature to remotely access the victim's files before encrypting them.
8	Cerber	Encryptor	This encryption ransomware was selected due to it encrypts only specific files from the infected device. Leaves additional ransom notes, such as an audio file that is addressed aloud to the victim, both on the desktop of the affected computer and inside encrypted folders.
9	cmd	Goodware	This goodware was selected due to its ability to execute scripts and commands.
10	Cryptolocker	Encryptor	This encryption ransomware was selected due to it encrypts only specific files from the infected device.
11	Cryptowall	Encryptor	This Ransomware (system blocker) was selected due to it infiltrates the user's operating system through an infected email message or a fraudulent download.
12	Crysis	Encryptor	Crysis ransomware uses brute force to infect computers.
13	dllhost	Goodware	This goodware was selected due to its access to dlls during different stages of the software use (execution, installation).

ID	Artifact	Family	Comments
14	Eris	Encryptor	ERIS renames all encrypted files and changes their extensions to ".ERIS" and uses both Salsa20 and RSA-1024 encryption.
15	Windows Remote Desk	Goodware	This goodware was selected due to its control interaction when there are permissions.
16	GandCrab	Encryptor	GandCrab is a virus for rent for other cybercriminals to spread attacks with it.
17	gpg	Goodware	This goodware was selected due to the execution of public and private keys.
18	IPScan	Goodware	This goodware was selected because it allows scanning IP addresses in several environments.
19	Locky	Encryptor	This ransomware was selected because it is distributed via email or exploit kit with Microsoft Word attachment.
20	Maze	Encryptor	This ransomware threatens to leak information from encrypted files, if the demanded ransom is not paid. It is designed to attack Windows operating systems.
21	Microsoft SQL Server Compact	Goodware	This goodware was selected due to its use for database management.
22	Nmap	Goodware	This goodware was selected because it allows different scanning parameters such as open ports and IP addresses, among others.
23	Petrwrap	Locker	PetrWrap, a variant of Petya that takes into account WannaCry's Eternal Blue exploit
24	Petya	Locker	This ransomware was selected because overwrites the main boot record of the infected computer
25	Phobos	Encryptor	This ransomware encrypts data to demand payment for decryption. During the encryption process, files are renamed according to this pattern: original file name, unique ID assigned to victims, cybercriminals' email address, and the extension ".iso" (not to be confused with the format genuine ISO disc image)
26	Radamant	Encryptor	The ransomware encrypts data using AES-256 encryption and the file extensions ".RDM" or ".RRK" are appended to infected files. It spreads via spam email attachments, corrupted links, fake advertisements and so on. (RaaS)
27	RansomX	Encryptor	This ransomware is used in targeted attacks against government agencies and companies.
28	Ryuk	Locker	This ransomware encrypts the data on an infected system, making the data on it inaccessible until a ransom is paid in Bitcoin. It expressly seeks high-profile targets capable of paying large sums, such as large public entities.

ID	Artifact	Family	Comments
29	Satana	Locker	This ransomware encrypts files and prevents from starting Windows (injects the same into TaskHost.exe) and starts data encryption. (RaaS)
30	services	Goodware	This goodware was selected due to its interaction with operating system services.
31	Sodinokibi	Encryptor	Sodinokibi ransomware exploits a vulnerability in Oracle WebLogic to gain access to the target machine. Once inside, the malware attempts to deploy itself with elevated legal user rights to access all files as well as system resources without restrictions.
32	STOP	Encryptor	This ransomware uses a combination of AES and RSA algorithms to encrypt data and adds the .STOP file extension.
33	svchost	Goodware	This goodware was selected because it checks the operating system and is possibly the first victim of malware attacks.
34	Team Viewer	Goodware	This goodware was selected due to its interaction with remote control.
35	Teslacrypt	Encryptor	This ransomware is appeared as a threat targeted users to computer games, now has several versions, and affects many files
36	VNC	Goodware	This goodware was selected due to its interaction with remote control.
37	WannaCry	Encryptor	WannaCry takes advantage of the vulnerability of the SMB device sharing protocol
38	WhatsAppWeb	Goodware	This goodware was selected due to the use of encryption in sending and receiving messages.
39	Winrar	Goodware	This goodware was selected due to the use of file and directories encryption.
40	Wireshark	Goodware	This goodware was selected because it allows the obtention of important information through a network using pcap files.

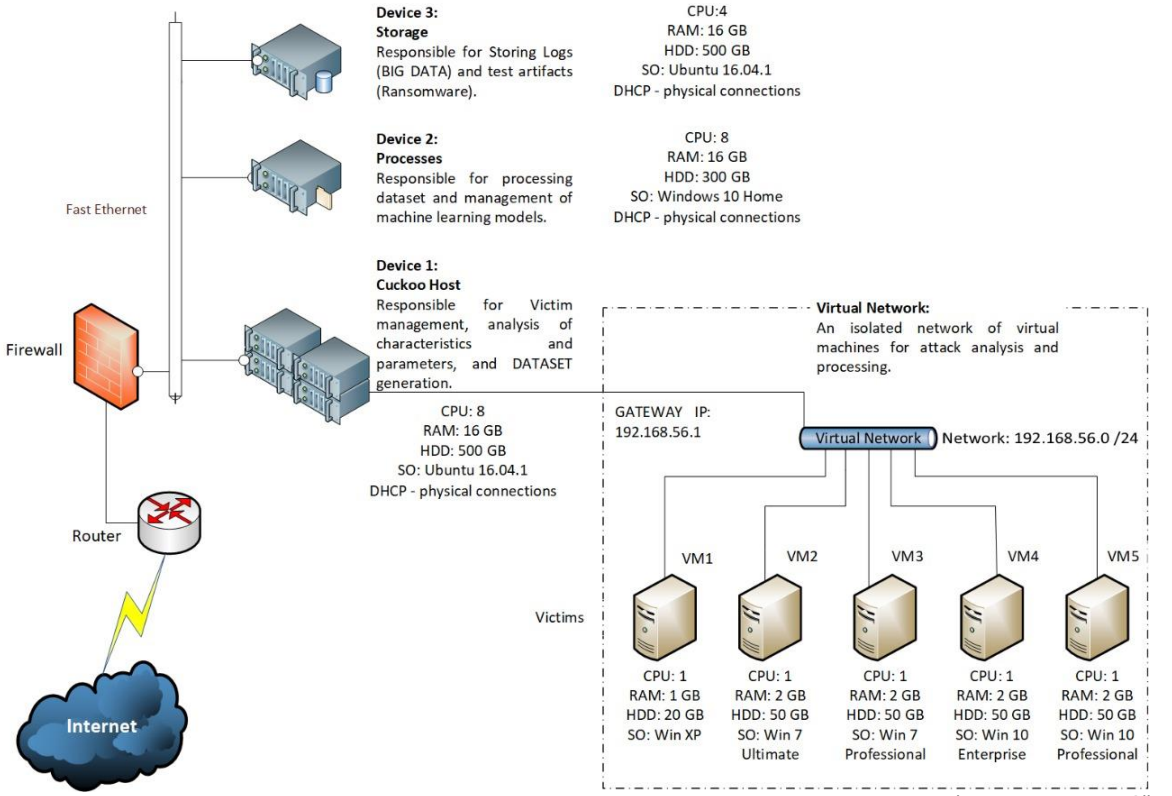
In Phase final, 50 features were used, as shown in Figure 14, which shows the GUI of the extraction tool generated in the present work. These characteristics are generated from the .json file produced in the cuckoo sandbox using the application described in section 4.2 and Annexes A and B. Phase final included Windows 10 as the platform and new threats and goodware, as shown in Table 12.



Author: Juan A. Herrera Silva

Figure 14. Features for Phase final that are automatically generated.

Figure 15 shows the Phase final test environment network topology. In this configuration, we have three machines; the first hosts cuckoo, the second CPU processes the models with machine learning, and the third machine is responsible for storing logs (big data) and artifacts for testing. Cuckoo communicates with an isolated virtual network for ransomware processing and analysis composed of CPUs in five platforms (victims): Windows XP Service Pack3, Windows 7 Ultimate, Windows 7 Professional, Windows 10 Enterprise and Windows 10 Professional. In this experimentation, the main 50 features selected for the generation of the 12 Datasets were obtained and consequently the different learning models were obtained.



Author: Juan A. Herrera Silva

Figure 15. Final Test environment network topology used in Phase Final

4.6. Balanced Dataset

The first dataset obtained consisted only of malware generated within the cuckoo sandbox. For balancing data, goodware records were included to avoid bias for having an unbalanced dataset with more malware than goodware records that could affect the classification when using this data within machine learning algorithms.

The column named Objects represents the JSON file's main features created by the Cuckoo Sandbox tool after analyzing artifacts (Ransomware or Goodware). The total number of main objects is 15, each with nested features. The total number of features in the JSON file is 326. Of these 326 characteristics, some have been chosen in Phases previous, initial and analysis under the criteria explained in Table 11. In Phase final, 50 features are selected (Figure 13) after several experiments with machine learning algorithms applied to diverse combinations of attributes. This process is detailed in Annexes D and E, where the different generated models are listed with their performance when using various combinations of features for supervised machine learning (Annex E) and Neural Networks (Annex F), both include detection times.

4.7. Machine Learning Algorithms

In this study, we tested machine learning algorithms for the generation of the models to recognize ransomware, as shown in Table 13. For the generation of the machine learning models, first, we used the RapidMiner tool[15] and obtained models for the mentioned algorithms as an approximation for evaluating performances. In Phase final, we took the algorithms with the best performances and applied the Scikit-learn library and the Python programming language with different parameters until the best results were obtained.

Table 13. Machine learning algorithms

ALGORITHM / TECHNICAL	REFERENCES	CHARACTERISTICS
Decision Tree	[110], [111]	1. It is a classifier in the form of a tree structure that includes branch nodes and leaf nodes 2. The decision tree is a supervised machine learning (ML) algorithm commonly used in regression analysis and classification.
Neural networks	[112]	Neural networks work similarly to a biological brain to recognize patterns of large amounts of data. Multi-layer neural network algorithms received raw data and performed internal processes to extract and select features. For this reason, they had an embedded feature extraction and selection process. A simple neural network includes an input layer, an output layer with the classified variables, and a hidden layer. The layers are connected and form a network of neurons.
Fast Large Margin	[113]	1. Quick learning method for high margin optimizations 2. It is based on the linear support vector learning scheme3. You can work on a dataset with millions of examples and attributes
Generalized Linear Model	[114]	1. GLMs are a class of models applied in cases where linear regression is not applicable or does not make appropriate predictions. 2. It consists of three components: 2.1. Random component: an exponential family of probability distributions; 2.2. Systematic component: a linear predictor; and 2.3. Link function: which generalizes the linear regression.
Gradient Boosted Trees	[115], [116], [117]	This algorithm is based on an ensemble of decision trees to improve the performance of each separate tree, considered individually as weak learners. The algorithm applies gradient augmentation algorithms and generates trees sequentially in a way that complements the errors of the previous tree, and this model is not random. Instead, it uses powerful pre-pruning. The trees combined their output results in better models. In the case of regression, the final result is generated from the average of all weak learners.

[15] https://rapidminer.com/downloads/

ALGORITHM / TECHNICAL	REFERENCES	CHARACTERISTICS
Logistic Regression	[118], [119]	1. the nominal attributes are transformed into numerical attributes 2. This algorithm is optimized for conditional probability.
Naive Bayes	[119], [120]	This algorithm generates probabilistic models on target variables. It assumes that input features are independent without pairwise correlation, which is not entirely accurate in most cases. This assumption of uncorrelated attributes makes this algorithm “naive”. The name Bayes comes from the famous probabilistic theorem on which this algorithm bases the generation of the probabilistic model.
Random Forest	[118], [119] , [121], [122], [123]	This algorithm is an ensembled method combining tree predictors so that each tree depends on the values of an independently sampled random vector and has the same distribution for all trees in the forest. It can improve performance compared to independent decision trees. The random forest algorithm uses a collection of decision trees to vote and predict the input data class.
Support Vector Machines	[122], [124]	1. It is a machine learning mechanism based on the concept of structural risk minimization of the Statistical Learning Theory 2. Separate data points as much as possible 3. It is based on the concept of decision planes that define decision limits

Model Generation with Machine Learning Algorithms

Phases previous through analysis used RapidMiner to generate the machine learning models. In Phase final, we used the Scikit-Learn library and Python programming language to define each algorithm's parameters flexibly. Two particular variables are “estimators” and “versions.” The estimator’s variable contains an array of integers listing the number of trees to consider in each algorithm. This way, a Random Forest model and, a GBRT model with five trees, another model with ten trees will be trained.

The “versions” variable allows assigning an identifier to each pair of models. The value '1' will correspond to the first two models generated by the script, and the value '2' to the next two. Then, the part that will be used for training and the part that will be used for evaluation is obtained from the dataset. The “test_size” parameter denotes the percentage of the dataset that will be taken for assessment. Also, the dataset for cross-evaluation is divided into ten folds, with 10% of the dataset for testing and 90% for training, a process repeated ten times to obtain performance scores that can be averaged. Using the results of the precedent phases, we used the following machine learning algorithms: Random forest, Gradient boosted regression trees, Gaussian Naïve Bayes, and Neural Networks.

Machine Learning Parameters for Phase Final

The class and parameters described below are used to generate each model for Phase final since, in the other phases, we used a free version of RapidMiner that was useful as a first approximation to the problem. However, with this tool, we did not specify personalized parameters as in Python. In Phase final, we used the algorithms that yielded better performances in the other phases and programmed them in Python. We implemented Random Forest, Gradient Boosted Regression Trees, Neural Networks, and for comparison, kept an algorithm with not that good performance for our problem: Gaussian Naïve Bayes.

Once the models are developed, cross-validation is carried out with a total of 10 splits to validate each model effectively. Once this validation is done, we obtain the metrics of Precision, Recall, F1, and the confusion matrix to validate the results of each model.

Random Forest Parameters

For the generation of this model, the Python sklearn library was used. The RandomForestClassifier class allows us to create models of the type Random Forest. The following parameters were used to generate machine learning models with the Random Forest algorithm:

- Estimators (5-100): The number of estimators varies from 5 to 100. A maximum of one hundred has been defined to avoid overfitting.
- Criterion (gini): This function was used to measure the efficiency of each tree division since it measures each node's impurity.
- Maximum Depth (none): A maximum depth was not defined due to the nature of the data set and the available number of records used for model generation.
- The minimum number of records for node division (2): The default property of the library was used since each record contains the necessary information to identify each artifact.
- A minimum number of records in leaf nodes (1): This number is the minimum given that it is enough for a record to be labeled as software of a particular type to have the certainty of the prediction.
- A maximum number of features (auto): A maximum number of features to be considered was not defined since they were manually selected for each dataset.

- A maximum number of leaf nodes (none): A maximum number of leaf nodes is not defined due to the nature of the dataset.
- Number of jobs (none): A maximum of 1 job was used for a model generation because there was no large dataset to consider parallel processing.
- Random state (6): This state allows for consistency between all the models generated with all the datasets.
- A maximum number of records (None): Limiting the number of records for a model generation was not desired.

Gradient Boosted Regression Trees Parameters

Like Random Forest models, sklearn allows us to generate GBRT models with the GradientBoostingClassifier class.

- Estimators (5-100): The number of estimators varies from 5 to 100. A maximum of one hundred has been defined to avoid overfitting.
- Learning Rate (0.1): It was decided to use the default value recommended by the library since it presents favorable results.
- Subsample (1.0): Indicates the number of records each tree will use, so it was decided to use all the records.
- Criterion (Friedman Mean Squared Error): It was decided to use the mse as the quality criterion for each division within each tree since it is an improved version of the standard Mean Squared Error criterion.
- A minimum number of records for node division (2): The default property of the library was used since each record contains the necessary information to identify each artifact.
- A minimum number of records in leaf nodes (1): This number is the minimum given that it is enough for a record to be labeled as software of a specific type to have the certainty of the prediction.
- Maximum Depth (none): A maximum depth was not defined due to the nature of the data set and the available number of records used for model generation.
- Random state (6): This state allows for consistency between all the models generated with all the datasets.
- A maximum number of features (auto): A maximum number of features to be considered was not defined since they were manually selected for each dataset.

- A maximum number of leaf nodes (none): A maximum number of leaf nodes is not defined due to the nature of the dataset.
- Tolerance (0.004): This tolerance measures the loss calculated between each estimator. This value was used because it is the one recommended by the library.

Gaussian Naive Bayes Parameters

Like the previously mentioned models, sklearn allows us to generate GBRT models with the GaussianNB class. Gaussian Naïve Bayes is a probabilistic classification model, as shown in (7), which assumes that the features are independent. Even when this assumption is invalid, the model works reasonably well in most cases.

$$P(c|x) = \frac{\mathrm{P(X|C)} * P(c)}{\mathrm{P(x)}} \qquad (1)$$

Where:

P(c|x) = Posterior probability

P(x|c) = Likelihood

P(c) = Class prior probability

P(x) = Predictor prior probability

In Phase final, we applied the default parameters to give flexibility to the algorithm because if the prior probabilities of the classes are specified, the priors are not adjusted according to the data.

Neural Networks Parameters

We used the Python programming language and the Tensorflow and Keras library for Neural Networks to create artificial neural networks. The number of splits to be considered for the cross-validation process is defined. The following variable defines the dataset to train and save the models. The third variable is an array containing the metrics to consider when evaluating the generated models and the optimizer used. In this case, the “Adam” optimizer is used with a learning rate of 0.001 to get a good model precision and simultaneously obtain a solution quickly.

Then there is another series of variables also used for the training and evaluating variables where the architectures to be built are defined. The program will perform a permutation of the activation functions, the number of neurons, and the number of layers of each model to be generated.

There is also a function to encode the classes of the dataset. This output layer is necessary for the artificial neural networks to predict the three categories we have in our research: encryptor ransomware, locker ransomware, and goodware. Next, the models to be generated are built, trained, and validated. A cross-validation process is executed, and each trained model is saved. The parameters to use are:

- Activation functions: Commonly used functions were used to generate DL models and that were available in the library of tensorflow. These are sigmoid, selu, relu and tanh.
- Number of neurons: The number of neurons range from 25 to 300. Each layer of each network will always have the same number of neurons defined for the model. No permutations between quantities of neurons per layer to facilitate the process of generating the Models. The maximum number of neurons is 300 to avoid overfitting.
- Number of layers: The models range from 1 layer to 4 layers. Increasing the number of layers was also avoided due to the nature of the dataset.
- Learning Rate: A learning rate of 0.001 was used since it is the recommended by several authors and, after experimentation, it was enough to obtain models with good results.
- Epochs: Limited to a maximum of 20 epochs for all given models that the loss between epochs during training did not vary from drastic way.
- Metrics: As with ML models, for DL models extracted Precision, Accuracy and Recall. For the accuracy of the stage of training, the accuracy obtained in the last epoch of each is taken into account model.
- Output function: softmax was used since it is recommended by various authors for general problems.

Performance of the Classifiers

Our study evaluated the machine learning algorithms' performance using several metrics listed below and a classifier's confusion matrix to calculate these metrics.

$$\text{True positive rate (TPR)} = \frac{TP}{TP+FN} \quad (2)$$

$$\text{False positive rate (FPR)} = \frac{FP}{FP+TN} \quad (3)$$

$$\text{Precision} = \frac{TP}{TP+FP} \quad (4)$$

$$\text{Recall} = \frac{TP}{TP+FN} \quad (5)$$

$$\text{F} - \text{measure} = \frac{2*Precision*Recall}{Precision+Recall} \quad (6)$$

$$\text{Accuracy } = \frac{TP+TN}{TP+TN+EP+FN} \quad (7)$$

Equations two to seven show the following descriptions: TP is a true positive representing the number of ransomware samples classified correctly. TN is a true negative, meaning the number of standard samples categorized accurately. FP is a false positive that represents regular binaries incorrectly classified as ransomware. FN is a false negative that represents ransomware incorrectly classified. TPR gives the predicted ransomware value correctly classified as ransomware, while FPR gives the value of files incorrectly classified as ransomware.

Precision defines the machine learning model's precision in categorizing relevant instances. Recall establishes the ability to find pertinent instances of the data set. F-measure is the harmonic mean of accuracy and recovery and estimates the given machine learning model’s performance. The equations (two to seven) compute the performance of the algorithms using five features. The same is done for the other combinations of characteristics.

5. DATASET, MODELING, AND DEPLOYMENT

The dataset and its final features were applied to machine learning algorithms to detect locker ransomware and encryptor ransomware to differentiate them from goodware. The combination of characteristics used was the one that yielded the best algorithm performances. This way, we detect this computer threat to minimize the damage that it can cause. The work's hypothesis was confirmed, and the objectives of this investigation were achieved.

In this chapter, we will describe results obtained in the different phases to explain the evolution of the research to get the Ransomware Features Dataset, the resulting datasets, machine learning models generated with the final version of the dataset, and the deployment using the best models.

5.1. Evolution of the Research to Obtain the Ransomware Features Dataset

It was mentioned that the present work had four phases (previous, initial, analysis and final). In each stage, a dataset was generated using some of the features of the .json file obtained after processing artifacts in the cuckoo file and adding goodware files. Each of the resulting datasets was evaluated using machine learning algorithms. Table 14 summarizes the features, artifacts, platforms, and the number of generated registers of the CSV dataset generated in each phase.

Table 14. Relevant attributes of the datasets generated in each phase using specific artifacts and platforms

Phase	Selected features	Artifacts	Platforms	Number of registers of the CSV dataset file
Previous	regwrite, regopen, regread, proc, pmfiles, pmurls, nethosts, netrequest.	Cryptolocker, Cryptowall, Petrwrap, Petya , Wannacry	Windows XP, Windows 7	6.783
Initial	regwrite, regopen, regread, proc, pmfiles, pmurls, nethosts, netrequest.	7-zipPortable_9.20_ Rev_2.paf.exe, AdbeRdr11008_es_ES.exe, AcroRdrDC1901220036_es_ ES.exe, cerber.exe, chrome.exe, cryptolocker.exe, cryptowall.bin,	Windows XP, Windows 7	47.959

Phase	Selected features	Artifacts	Platforms	Number of registers of the CSV dataset file
		dllhost.exe (W7), dllhost.exe (WXP), explorer.exe (W7), explorer.exe (WXP), firefox.exe, locky, Petrwrap.exe, petya.bin, radamant.ViR, satana.bin, services.exe (W7), services.exe (WXP), svchost.exe (W7), svchost.exe (WXP), teslacrypt, wannacry.exe, WinRAR.EXE.		
Analysis	regwrite, regopen, regread, proc, pmfiles, pmurls, nethosts, netrequest, file_created, dll_loaded, command_line, udp, tcp, domains	7-zipPortable_9.20_ Rev_2.paf.exe, AdbeRdr11008_es_ES.exe, AcroRdrDC1901220036_es_ ES.exe, cerber.exe, chrome.exe, cryptolocker.exe, cryptowall.bin, dllhost.exe (W7), dllhost.exe (WXP), explorer.exe (W7), explorer.exe (WXP), firefox.exe, locky, Petrwrap.exe, petya.bin, radamant.ViR, satana.bin, services.exe (W7), services.exe (WXP), svchost.exe (W7), svchost.exe (WXP), teslacrypt, wannacry.exe, WinRAR.EXE.	Windows XP, Windows 7	62.989
Final	family, proc_pid, file, urls, type, name, ext_urls, path, program, info, families, description, sign_name, sign_stacktrace, arguments, api, category, imported_dll_count, dll, pe_res_name, filetype,	7Zip, Task Manager (taskmgr), API WINDOWS SECURITY CRYPTOGRAPHY (cipher), API WINDOWS SYSTEM INFORMATION REGISTRY (regedit), API WINDOWS VOLUME MANAGEMENT (diskpart), Bitlocker,	Windows XP_SP3 Windows 7_Ultimate Windows 7_Professional Windows 10_Enterprise Windows 10_Professional	1.424.344

Phase	Selected features	Artifacts	Platforms	Number of registers of the CSV dataset file
	pe_sec_name, entropy, hosts, requests, mitm, domains, dns_servers, tcp, udp, dead_hosts, proc, beh_command_line, process_path, children, tree_command_line, tree_process_name, command_line, regkey_read, wmi_query, directory_enumerated, regkey_opened, log, file_created, action, dll_loaded, file_read, regkey_written, apistats, errors	BitPaymer, Cerber, cmd, Cryptolocker, Cryptowall, Crysis, dllhost, Eris, Windows Remote Desk, GandCrab, gpg, IPScan, Locky, Maze, Microsoft SQL Server Compact, Nmap, Petrwrap, Petya, Phobos, Radamant, RansomX, Ryuk, Satana, services, Sodinokibi, STOP, svchost, Team Viewer, Teslacrypt, VNC, WannaCry, WhatsAppWeb, Winrar, Wireshark.		
	family, proc_pid, file, urls, type, name, ext_urls, path, program, info, families, description, sign_name, sign_stacktrace, arguments, api, category, imported_dll_count, dll, pe_res_name, filetype, pe_sec_name, entropy, hosts, requests, mitm, domains, dns_servers, tcp, udp, dead_hosts, proc, beh_command_line, process_path, children, tree_command_line, tree_process_name, command_line, regkey_read, wmi_query, directory_enumerated, regkey_opened, log, file_created, action, dll_loaded, file_read, regkey_written, apistats, errors	7Zip, Task Manager (taskmgr), API WINDOWS SECURITY CRYPTOGRAPHY (cipher), API WINDOWS SYSTEM INFORMATION REGISTRY (regedit), API WINDOWS VOLUME MANAGEMENT (diskpart), Bitlocker, BitPaymer, Cerber, cmd, Cryptolocker, Cryptowall, Crysis, dllhost, Eris, Windows Remote Desk, GandCrab, gpg, IPScan, Locky, Maze, Microsoft SQL Server Compact, Nmap, Petrwrap, Petya, Phobos, Radamant, RansomX, Ryuk, Satana, services, Sodinokibi, STOP, svchost, Team Viewer, Teslacrypt, VNC, WannaCry, WhatsAppWeb, Winrar, Wireshark.	Windows XP_SP3 Windows 7_Ultimate Windows 7_Professional Windows 10_Enterprise Windows 10_Professional	2.000

Experiments in Phases Initial and Analysis

This section will explain the processes in phases initial and analysis to obtain different combinations of features to feed the machine learning algorithms to evaluate their performance to establish the best possible configuration for the ransomware dataset. We divided the dataset by a Split operator in 75% for training and 25% for testing.

Combination of Features

We applied feature engineering and produced different feature vectors as a combination of characteristics. With these inputs, we evaluated processing times and algorithms' performances. The combination spreadsheets refer to the number of non-repeatable combinations made with the selected attributes to form the dataset and the number of models generated for each combination. Also, the investigation has measured the time necessary to get the models. This time is, on average, 4 hours. The number of combinations must be multiplied by this number to obtain the time in hours and then divided by 24 to get the time in days.

In the phase initial, it was chosen a total of 7 characteristics to form the dataset. Table 15 covers these considerations for the seven features. In the second phase, seven features were added to the dataset for a total of 14. Table 16 shows the corresponding calculations.

The time that all models would finish with all the features is too high (24.574,5 days), so it was decided to choose certain more significant features in the analysis. From stage 1, the regkey_read feature was selected, which affects the application layer models' prediction behavior. With a broader network knowledge criterion, the UDP feature and the file feature were also considered because they describe the action at the application layer level. With these three characteristics added in one, the research obtained five factors to analyze. Table 17 shows the necessary time for the calculations.

Table 15. Processing times for a combination of 7 characteristics

Combination	# Combinations Without Repetition	# Models	
7	7	1	9
7	6	7	63
7	5	21	189
7	4	35	315
7	3	35	315

Combination		# Combinations Without Repetition	# Models	
7	2		21	189
7	1		7	63
TOTAL			127	1143
		Time by a combination (Hours)		4
		TOTAL Time (Hours)		4572
		TOTAL Time (Days)		190.50

Table 16. Processing times for a combination of 14 characteristics

Combination		# Combinations Without Repetition	# Models
14	14	1	9
14	13	14	126
14	12	91	819
14	11	364	3276
14	10	1001	9009
14	9	2002	18018
14	8	3003	27027
14	7	3432	30888
14	6	3003	27027
14	5	2002	18018
14	4	1001	9009
14	3	364	3276
14	2	91	819
14	1	14	126
TOTAL		16383	147447
		Time by a combination (Hours)	4
		TOTAL Time (Hours)	589788
		TOTAL Time (Days)	24574.50

Table 17. Processing times for a combination of 5 characteristics

Combination		# Combinations Without Repetition	# Models
5	5	1	9
5	4	5	45
5	3	10	90
5	2	10	90
5	1	5	45

Combination	# Combinations Without Repetition	# Models
TOTAL	31	279
	Time by a combination (Hours)	4
	TOTAL Time (Hours)	1116
	TOTAL Time (Days)	46.50

In these phases, 279 learning models were processed in 60 days to obtain algorithm analysis, results, and confusion matrices. The discussion includes comparative analysis. Table 18 represents the combinations of characteristics used to generate the models. We have marked these mixtures with an identifier to present the performance of the classifiers to which these input vectors are applied. The comparison of performances for training is shown in Table 19, which gives the accuracy percentages to predict the type of artifact taken during model training.

Table 18. Nomenclature for the combination of characteristics

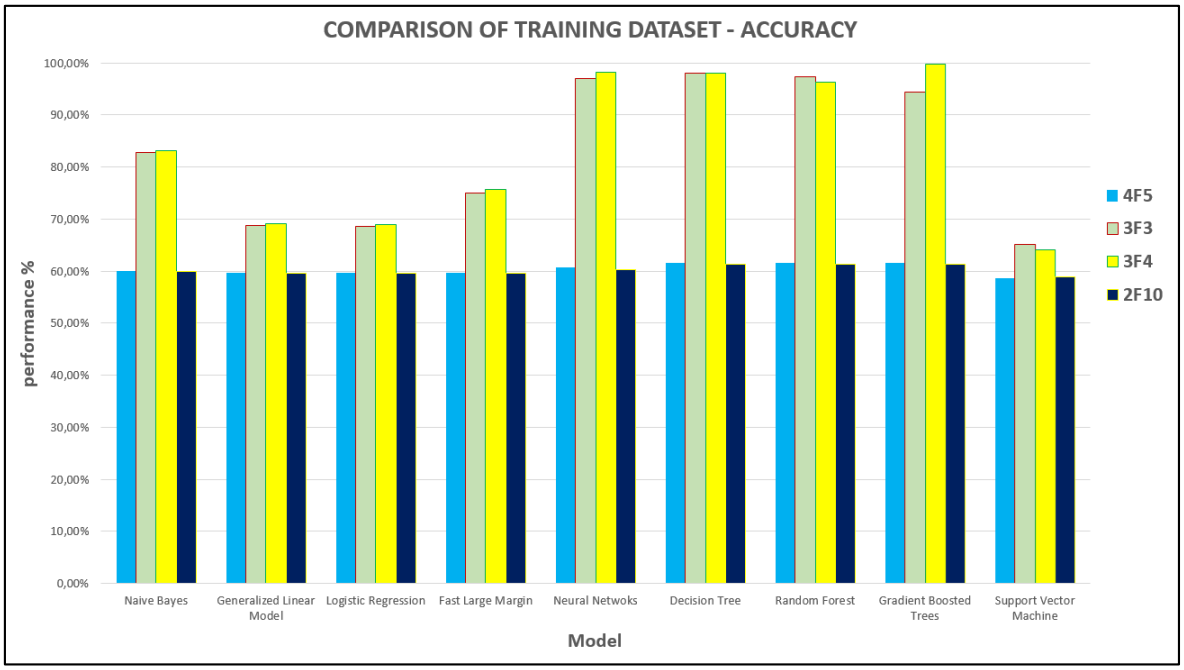
ID	**FEATURES**
5F1	(regkey_read, udp, file_created), dll_loaded, comand_line, domain, tcp
4F1	(regkey_read, udp, file_created), command_line, domain, tcp
4F2	(regkey_read, udp, file_created), dll_loaded, command_line, domain
4F3	(regkey_read, udp, file_created), dll_loaded, command_line, tcp
4F4	(regkey_read, udp, file_created), dll_loaded, domain, tcp
4F5	dll_loaded, comand_line, domain, tcp
3F1	(regkey_read, udp, file_created), comand_line, domain
3F2	(regkey_read, udp, file_created), comand_line, tcp
3F3	(regkey_read, udp, file_created), dll_loaded, comand_line
3F4	(regkey_read, udp, file_created), dll_loaded, domain
3F5	(regkey_read, udp, file_created), dll_loaded, tcp
3F6	(regkey_read, udp, file_created), domain, tcp
3F7	comand_line, domain, tcp
3F8	dll_loaded, comand_line, domain
3F9	dll_loaded, comand_line, tcp
3F10	dll_loaded, domain, tcp
2F1	(regkey_read, udp, file_created), comand_line
2F2	(regkey_read, udp, file_created), dll_loaded
2F3	(regkey_read, udp, file_created), domain
2F4	(regkey_read, udp, file_created), tcp
2F5	comand_line, domain
2F6	comand_line, tcp
2F7	dll_loaded, comand_line

ID	FEATURES
2F8	dll_loaded, domain
2F9	dll_loaded, tcp
2F10	domain, tcp
1F1	(regkey_read, udp, file_created)
1F2	comand_line
1F3	dll_loaded
1F4	domain
1F5	tcp

Table 19. Comparison of performances with different algorithms in the training dataset

Model	5F1	4F1	4F2	4F3	4F4	4F5	3F1	3F2	3F3	3F4	3F5
Naive Bayes	83,29%	83,23%	83,28%	82,77%	83,25%	60,01%	83,22%	82,71%	82,77%	83,22%	82,71%
Generalized Linear Model	69,40%	69,87%	69,27%	68,89%	69,50%	59,75%	69,70%	69,14%	68,84%	69,18%	68,92%
Logistic Regression	69,34%	69,77%	69,00%	68,62%	69,28%	59,74%	69,60%	68,92%	68,60%	68,93%	68,63%
Fast Large Margin	75,87%	75,85%	75,76%	74,96%	75,85%	59,60%	75,71%	74,91%	74,98%	75,65%	74,85%
Neural Networks	98,26%	98,34%	98,31%	97,13%	98,36%	60,68%	98,31%	97,11%	97,01%	98,24%	97,05%
Decision Tree	98,09%	98,09%	98,09%	98,09%	98,09%	61,50%	98,09%	98,09%	98,09%	98,09%	98,09%
Random Forest	93,18%	94,46%	94,33%	94,17%	94,97%	61,51%	96,69%	98,16%	97,30%	96,33%	97,96%
Gradient Boosted Trees	99,73%	99,72%	99,73%	98,48%	99,72%	61,49%	99,72%	98,48%	94,48%	99,74%	98,48%
Support Vector Machine	63,18%	62,55%	64,49%	64,30%	66,58%	58,65%	61,12%	62,46%	65,07%	64,07%	64,00%

Figure 16 presents graphs with the comparison of the training dataset sheet. These graphs help establish which algorithm and features perform better by pinpointing a type of artifact.



Author: Juan A. Herrera Silva

Figure 16. Comparison of performance of the algorithms over the training dataset – Phase Analysis

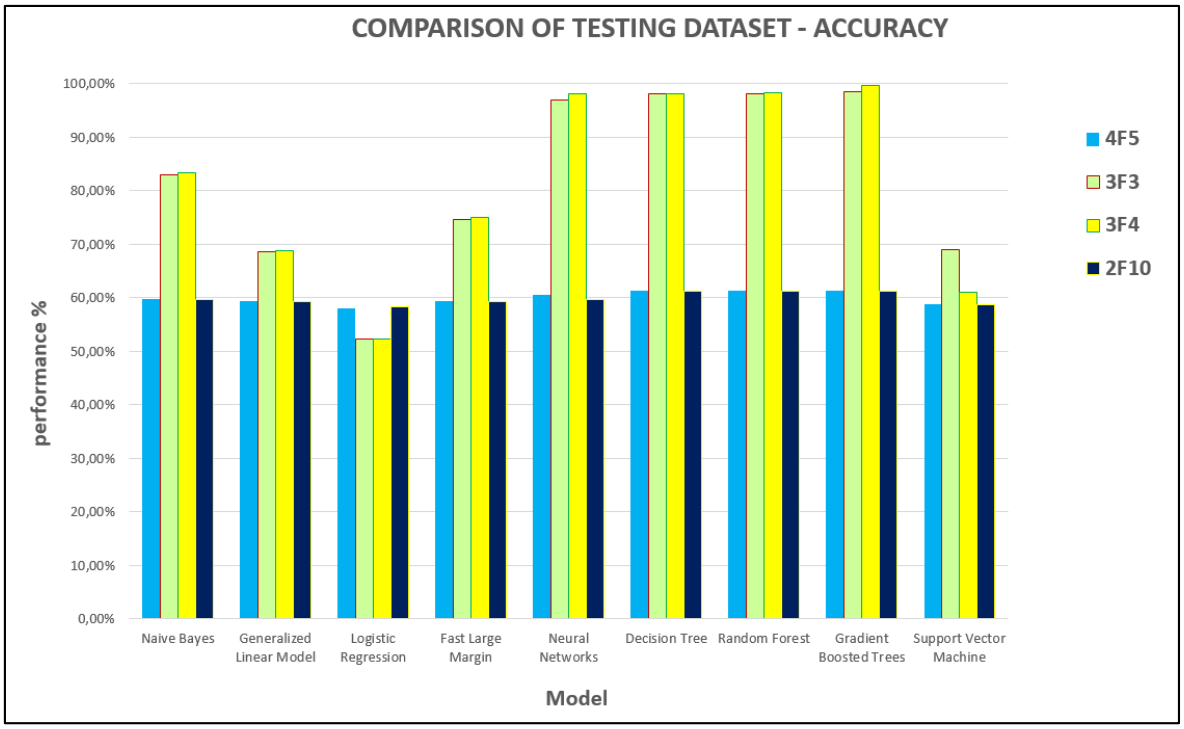
The comparison of testing datasets presents the accuracy percentages to predict the artifact during model testing. There is a relationship between the parameters chosen and the models used. Table 20 indicates the performance of the different algorithms over the testing dataset.

Table 20. Comparison of performance of the algorithms over the testing dataset

Model	5F1	4F1	4F2	4F3	4F4	4F5	3F1	3F2	3F3	3F4	3F5
Naive Bayes	83.39%	83.30%	83.39%	82.81%	83.32%	59.83%	83.31%	82.74%	82.85%	83.30%	82.76%
Generalized Linear Model	69.09%	69.48%	68.93%	68.62%	69.14%	59.44%	69.39%	68.96%	68.62%	68.83%	68.65%
Logistic Regression	52.29%	52.25%	52.27%	52.30%	52.29%	58.03%	52.25%	52.27%	52.27%	52.29%	52.29%
Fast Large Margin	75.31%	75.23%	75.24%	74.56%	75.26%	59.36%	75.16%	74.47%	74.61%	75.07%	74.38%
Neural Networks	98.06%	98.00%	98.18%	96.45%	98.07%	60.60%	98.26%	96.53%	96.85%	98.16%	96.18%
Decision Tree	98.05%	98.05%	98.05%	98.05%	98.05%	61.27%	98.05%	98.05%	98.05%	98.05%	98.05%
Random Forest	92.88%	93.06%	96.89%	92.90%	96.89%	61.30%	98.22%	98.06%	98.05%	98.22%	98.23%
Gradient Boosted Trees	99.68%	99.68%	99.67%	98.38%	99.68%	61.27%	99.67%	98.38%	98.38%	99.68%	98.38%
Support Vector Machine	65.39%	62.75%	59.72%	59.79%	62.24%	58.77%	66.71%	58.55%	68.99%	60.99%	64.96%

Comparison of testing results

Figure 17 presents graphs associated with comparing the algorithms over the testing datasets. These graphs help establish which algorithm performs best in the generation of the models and with which features pinpoint the prediction of a type of artifact.



Author: Juan A. Herrera Silva

Figure 17. Comparison of performance of the algorithms over the testing dataset – Phase Analysis

Table 21. Performance of the classifiers using five features over the training dataset

(Features: regkey_read, udp, file_created), dll_loaded, command_line, domain, tcp)				
Model	**Accuracy**	**Precision**	**Recall**	**Classification Error**
Naive Bayes	83.29%	69.96%	57.63%	16.71%
Generalized Linear Model	69.40%	47.06%	44.86%	30.60%
Logistic Regression	69.34%	46.93%	44.84%	30.66%
Fast Large Margin	75.87%	59.99%	48.65%	24.13%
Neural Networks	98.26%	98.76%	95.32%	1.74%
Decision Tree	98.09%	97.86%	97.06%	1.91%
Random Forest	93.18%	96.55%	65.40%	6.82%
Gradient Boosted Trees	99.73%	99.83%	98.47%	0.27%
Support Vector Machine	63.18%	53.49%	37.16%	36.82%

Table 22. Performance of the classifiers using five features over the testing dataset

(Features: regkey_read, udp, file_created), dll_loaded, command_line, domain, tcp)				
Model	**Accuracy**	**Precision**	**Recall**	**Classification Error**
Naive Bayes	83.39%	68.15%	57.86%	16.61%

(Features: regkey_read, udp, file_created), dll_loaded, command_line, domain, tcp)				
Generalized Linear Model	69.09%	46.99%	44.69%	30.91%
Logistic Regression	52.29%	40.68%	38.79%	47.71%
Fast Large Margin	75.31%	56.52%	48.31%	24.69%
Neural Networks	98.06%	98.78%	94.71%	1.94%
Decision Tree	98.05%	97.70%	96.86%	1.95%
Random Forest	92.88%	96.39%	65.28%	7.12%
Gradient Boosted Trees	99.68%	99.81%	98.11%	0.32%
Support Vector Machine	65.39%	54.31%	39.34%	34.61%

Our study considers the algorithms specified at the beginning of this section, as shown in Tables 21 and 22. Tables 23 through 26 present the performance of the classifiers using different combinations of features both in training and testing. The Gradient Boosted Trees algorithm has better accuracy during the training process with 99.73%, and the testing process has 99.68%.

Table 23. Performance of the classifiers using a combination of four features in training and testing

Characteristics	Algorithm / Model	Accuracy training	Precision training	Recall training	Classification Error Training	Accuracy testing	Precision testing	Recall testing	Classification Error testing
(regkey_read, udp, file_created), command_line, domain, tcp	Gradient Boosted Trees	99.72%	99.80%	98.48%	0.28%	99.68%	99.81%	98.11%	0.32%
(regkey_read, udp, file_created), dll_loaded, command_line, domain	Gradient Boosted Trees	99.73%	99.82%	98.48%	0.27%	99.67%	99.81%	98.11%	0.33%
(regkey_read, udp, file_created), dll_loaded, domain, tcp	Gradient Boosted Trees	99.72%	99.81%	98.48%	0.28%	99.68%	99.81%	98.11%	0.32%
(regkey_read, udp, file_created), dll_loaded, command_line, tcp	Gradient Boosted Trees	98.48%	99.13%	97.38%	1.52%	98.38%	99.10%	96.94%	1.62%
dll_loaded, comand_line, domain, tcp	Random Forest	61.51%	86.72%	36.71%	38.49%	61.30%	86.67%	36.86%	38.70%

Table 24. Performance of the classifiers using a combination of three features in training and testing

Characteristics	Algorithm / Model	Accuracy training	Precision training	Recall training	Classification Error training	Accuracy testing	Precision testing	Recall testing	Classification Error testing
(regkey_read, udp, file_created), comand_line, domain	Gradient Boosted Trees	99.72%	99.82%	98.48%	0.28%	99.67%	99.81%	98.11%	0.33%
(regkey_read, udp, file_created), comand_line, tcp	Gradient Boosted Trees	98.48%	99.12%	97.37%	1.52%	98.38%	99.10%	96.94%	1.62%
(regkey_read, udp, file_created), dll_loaded, comand_line	Gradient Boosted Trees	94.48%	99.12%	97.38%	1.52%	98.38%	99.10%	96.94%	1.62%
(regkey_read, udp, file_created), dll_loaded, domain	Gradient Boosted Trees	99.74%	99.82%	98.48%	0.27%	99.67%	99.81%	98.11%	0.33%
(regkey_read, udp, file_created), dll_loaded, tcp	Gradient Boosted Trees	98.48%	99.13%	97.38%	1.52%	98.38%	99.10%	96.94%	1.62%
(regkey_read, udp, file_created), domain, tcp	Gradient Boosted Trees	99.72%	99.80%	98.48%	0.28%	99.68%	99.81%	98.11%	0.32%
comand_line, domain, tcp	Random Forest	61.35%	86.24%	36.54%	38.65%	61.14%	86.63%	36.72%	38.86%
dll_loaded, comand_line, domain	Random Forest	61.39%	86.69%	36.23%	38.61%	61.17%	83.23%	36.39%	38.83%
dll_loaded, comand_line, tcp	Random Forest	59.85%	86.10%	35.21%	40.15%	59.64%	86.42%	35.23%	40.36%
dll_loaded, domain, tcp	Random Forest	61.52%	86.81%	36.73%	38.48%	61.30%	86.67%	36.86%	38.70%

Table 25. Performance of the classifier using a combination of two features in training and testing

Characteristics	Algorithm / Model	Accuracy training	Precision training	Recall training	Classification Error training	Accuracy testing	Precision testing	Recall testing	Classification Error testing
(regkey_read, udp, file_created), comand_line	Gradient Boosted Trees	98.47%	99.12%	97.37%	1.53%	98.38%	99.10%	96.94%	1.62%

Characteristics	Algorithm / Model	Accuracy training	Precision training	Recall training	Classification Error training	Accuracy testing	Precision testing	Recall testing	Classification Error testing
(regkey_read, udp, file_created), dll_loaded	Gradient Boosted Trees	98.48%	99.13%	97.38%	1.52%	98.38%	99.10%	96.94%	1.62%
(regkey_read, udp, file_created), domain	Gradient Boosted Trees	99.72%	99.82%	98.48%	0.28%	99.67%	99.81%	98.11%	0.33%
(regkey_read, udp, file_created), tcp	Gradient Boosted Trees	98.48%	99.13%	97.38%	1.52%	98.38%	99.10%	96.94%	1.62%
comand_line, domain	Random Forest	61.09%	85.94%	35.95%	38.91%	60.84%	83.00%	36.09%	39.16%
comand_line, tcp	Random Forest	59.69%	85.61%	35.07%	40.31%	92.88%	96.39%	65.28%	7.12%
dll_loaded, comand_line	Random Forest	59.64%	85.68%	34.56%	40.36%	59.41%	81.38%	34.68%	40.59%
dll_loaded, domain	Random Forest	61.39%	86.08%	36.23%	38.61%	61.16%	82.39%	36.38%	38.84%
dll_loaded, tcp	Random Forest	59.85%	85.37%	35.24%	40.15%	59.65%	85.33%	35.33%	40.35%
domain, tcp	Gradient Boosted Trees	61.36%	85.83%	36.54%	38.64%	61.13%	86.72%	36.57%	38.87%

Table 26. Performance of the classifiers using one feature in training and testing

Characteristics	Algorithm / Model	Accuracy training	Precision training	Recall training	Classification Error training	Accuracy testing	Precision testing	Recall testing	Classification Error testing
(regkey_read, udp, file_created)	Gradient Boosted Trees	98.47%	99.12%	97.37%	1.53%	98.38%	99.10%	96.94%	1.62%
comand_line	Random Forest	59.26%	85.30%	34.14%	40.74%	59.02%	71.72%	34.10%	40.98%
dll_loaded	Random Forest	59.63%	84.74%	34.52%	40.37%	59.42%	81.38%	34.68%	40.58%
domain	Gradient Boosted Trees	60.97%	80.67%	35.51%	39.03%	60.71%	76.52%	35.58%	39.29%
Tcp	Random Forest	59.69%	85.25%	35.07%	40.31%	59.47%	86.08%	35.03%	40.53%

To achieve a contextual view of our experiments' findings, we present the best results for accuracy, using selected features, for training and testing during Phase initial and analysis, from Table 27 through 30. Tables 27 and 28 below show the best accuracy using different

algorithms and considering the features that improve the classifiers' performance in Phase initial.

Table 27. Best accuracy in training (Phase initial)

Algorithm	Best Accuracy training	Number Of Features	Features
Naive Bayes	77,82%	8	Regkey_written, regkey_ opened, regkey_read, processes, files, URLs, hosts, requests
Gradient Boosted Trees	91,10%	7	Regkey_written, processes, files, URLs, requests, regkey_opened, regkey_read
Neural Networks	79,07%	6	Regkey_written,processes, file, URLs, hosts, regkey_opened
Neural Networks	89,27%	5	Files, URLs, hosts, requests, regkey_read
Gradient Boosted Trees	90,13%	4	Processes, files, URLs, regkey_read
Gradient Boosted Trees	93,14%	3	Regkey_written, processes, regkey_read
Neural Networks	91,42%	2	Processes, regkey_read
Support Vector Machine	95,38%	1	Regkey_read

Table 28. Best accuracy in testing (Phase initial)

Algorithm	Best Accuracy testing	Number Of Features	Features
Naive Bayes	76,88%	8	Regkey_written, regkey_opened, regkey_read, processes,files, URLs, hosts, requests
Neural Networks	88,82%	7	Regkey_written, processes, files, URLs, requests, regkey_opened, regkey read
Fast Large Margin	73,56%	6	Regkey_written, processes, file, URLs, hosts, regkey_opened
Support Vector Machine	86,98%	5	Files, URLs, hosts, requests, regkey read
Gradient Boosted Trees	80,68%	4	Processes, files, URLs, regkey, read

Algorithm	Best Accuracy testing	Number Of Features	Features
Gradient Boosted Trees	93,14%	3	Regkey_written, processes, regkey_read
Neural Networks	89,43%	2	Processes, regkey_read
Random Forest	80,58%	1	Regkey_read

Tables 28, 29 and 30 show the algorithm's performance that obtained the best accuracy using different combinations of features, including a grouping of various features. It can be observed that performance was improved with the use of new features in this phase.

Table 29. Best accuracy in training (Phase analysis)

Algorithm	Best Accuracy Training	Number Of Features	Features
Gradient Boosted Trees	99,73%	5	(regkey_read, udp, file_created), dll_loaded, command_line, domain, tcp
Gradient Boosted Trees	99,73%	4	(regkey_read, udp, file_created), command_line, domain, tcp
Gradient Boosted Trees	99,73%	4	(regkey_read, udp, file_created), dll_loaded, command_line, domain
Gradient Boosted Trees	99,73%	4	(regkey_read, udp, file_created), dll_loaded, domain,tcp
Gradient Boosted Trees	99,74%	3	(regkey_read, udp, file_created), dll_loaded,domain
Gradient Boosted Trees	99.74%	2	(regkey_read, udp, file_created), domain
Gradient Boosted Trees	98,48%	1	(regkey_read, udp, file_created)

Table 30. Best accuracy in testing (Phase analysis)

Algorithm	Best Accuracy testing	Number Of Features	Features
Gradient Boosted Trees	99,68%	5	(regkey_read, udp, file_created), dll_loaded, command_line, domain, tcp
Gradient Boosted Trees	99,68%	4	(regkey_read, udp, file_created), command_line, domain, tcp
Gradient Boosted Trees	99,68%	4	(regkey_read, udp, file_created), dll_loaded, command_line, domain
Gradient Boosted Trees	99,68%	3	(regkey_read, udp, file_created), dll_loaded, domain
Gradient Boosted Trees	99,67%	2	(regkey_read, udp, file_created),domain
Gradient Boosted Trees	98,38%	1	(regkey_read, udp, file_created)

Table 31 presents the classification reports for all the machine learning considered algorithms, i.e., the table shows data for True Positive Rate (TPR), False Positive Rate (FPR), Accuracy, Precision, recall, and Classification Error.

Table 31. Training and testing classification reports

Features: (regkey_read, udp, file_created), dll_loaded, domain								
Training dataset								
Model	**TPR 1**	**TPR 2**	**FPR 1**	**FPR 2**	**Accuracy**	**Precision**	**Recall**	**Classification Error**
Naive Bayes	0.9511	0.7296	0.8367	0.8282	83.22%	69.69%	57.50%	16.78%
Generalized Linear Model	0.8965	0.4417	0.6814	0.7284	69.18%	47.00%	44.61%	30.82%
Logistic Regression	0.8935	0.4395	0.68	0.7211	68.93%	46.71%	44.43%	31.07%
Fast Large Margin	0.9964	0.4573	0.7088	0.9877	75.65%	56.55%	48.46%	24.35%
Neural Networks	0.9946	0.9708	0.9783	0.8383	98.24%	98.55%	95.85%	1.76%
Decision Tree	0.9997	0.9534	0.9712	0.9995	98.09%	97.86%	97.06%	1.91%
Random Forest	1	0.9352	0.9415	0.9999	96.33%	98.09%	87.50%	3.67%
Gradient Boosted Trees	0.9999	0.9975	0.9956	0.9997	99.74%	99.82%	98.48%	0.27%

Support Vector Machine	0.9925	0.15	0.6229	0.9269	64.07%	49.69%	38.08%	35.93%

Testing dataset								
Model	**TPR 1**	**TPR 2**	**FPR 1**	**FPR 2**	**Accuracy**	**Precision**	**Recall**	**Classification Error**
Naive Bayes	0.9518	0.7347	0.8362	0.8329	83.30%	68.53%	57.78%	16.70%
Generalized Linear Model	0.8983	0.4343	0.6767	0.7297	68.83%	46.89%	44.42%	31.17%
Logistic Regression	0.4308	0.7282	0.7167	0.4195	52.29%	39.39%	38.78%	47.71%
Fast Large Margin	0.9967	0.4468	0.7029	0.9885	75.07%	56.39%	48.12%	24.93%
Neural Networks	0.9929	0.9717	0.9791	0.9844	98.16%	98.62%	95.90%	1.84%
Decision Tree	0.9998	0.953	0.9709	0.9993	98.05%	97.70%	96.86%	1.95%
Random Forest	0.9999	0.9574	0.9707	0.9998	98.22%	99.02%	97.06%	1.78%
Gradient Boosted Trees	0.9998	0.9977	0.9948	0.9995	99.68%	99.81%	98.11%	0.33%
Support Vector Machine	1	0.615	0.6008	1	60.99%	53.36%	35.38%	39.01%

Datasets for Phases Initial and Analysis are included in Annexes M and N, respectively.

5.2. Final Datasets of Features of Ransomware corresponding to Phase Final

For the construction of the final dataset, it is necessary to have a series of analyses of Ransomware and Goodware artifacts so that the information can be extracted from the information collected regarding the behavior of these artifacts. This analysis execution also takes much time since it is required to perform a dynamic analysis of several artifacts for as long as possible to collect all possible information about their behavior within different environments. Annex L shows the use of CPU and memory resources of different artifacts in the used platforms.

The results compile two thousand dynamic analyses with the cuckoo sandbox tool, including twenty non-malicious artifacts or goodware and twenty ransomware-type malicious artifacts. We used the same number of ransomware and goodware artifacts to build a balanced dataset. For balance, it is necessary to have the same number of samples for each class. The dataset was split for cross-evaluation, i.e., ten folds with 10% of the dataset

for testing and 90% for training, a process repeated ten times to obtain performance scores that can be averaged. Table 11 lists the artifacts that were considered for that analysis. The following platforms were considered (Figure 14):

- Windows XP Service Pack 3
- Windows 7 Ultimate
- Windows 7 Professional
- Windows 10 Enterprise
- Windows 10 Professional

Each artifact was analyzed ten times for each selected operating system to collect all possible information that can be recorded within different environments. This way, we have the results of fifty dynamic analyses (json files) for each artifact. As a total of forty artifacts were listed, this results in a total of two thousand executed analyzes from which the json files with the information of each analysis will be taken to build the respective data set.

Dataset Global from Phase Final

The information of the dataset is taken from the json files generated in the sandbox. The extraction tool explained in 4.2 allows extracting any number of features from each json generated by an artifact. For instance, if we need to get information for one specific characteristic such as “udp” that corresponds to the connections established through UDP during dynamic analysis, this feature is contained within an object called “network”. It can be observed that this feature does not have one register but multiple rows of information. It is a list of objects.

The extraction tool accedes to this list's content and saves each record in a row within the dataset, as shown in Figure 18.

udp
{'src': '192.168.56.20', 'dst': '13.86.101.172', 'offset': 1184, 'time': 3.1674599647521973, 'dport': 123, 'sport': 123}
{'src': '192.168.56.20', 'dst': '192.168.56.255', 'offset': 11049, 'time': 3.2924230098724365, 'dport': 137, 'sport': 137}
{'src': '192.168.56.20', 'dst': '192.168.56.255', 'offset': 20985, 'time': 9.29214596748352, 'dport': 138, 'sport': 138}
{'src': '192.168.56.20', 'dst': '224.0.0.252', 'offset': 30015, 'time': 3.2200019359588623, 'dport': 5355, 'sport': 49252}
{'src': '192.168.56.20', 'dst': '224.0.0.252', 'offset': 30379, 'time': 45.782649993896484, 'dport': 5355, 'sport': 50100}
{'src': '192.168.56.20', 'dst': '224.0.0.252', 'offset': 30707, 'time': 53.46725010871887, 'dport': 5355, 'sport': 56804}
{'src': '192.168.56.20', 'dst': '224.0.0.252', 'offset': 31035, 'time': 50.90259313583374, 'dport': 5355, 'sport': 58349}

Author: Juan A. Herrera Silva

Figure 18. Dataset rows corresponding to an 'UDP' feature of an artifact.

The same process is applied to extract the rest of the features from the artifact's json file, which is saved in a CSV file. This process is carried out in this stage for different combinations of characteristics. Each corresponding dataset is evaluated with machine learning algorithms to obtain the optimal number of combination of attributes to generate high-performance models.

As described in detail in Annex C the main objects such as: info, procmemory, target, extracted, buffer, network, signatures, static, dropped, behavior, debug, and their respective characteristics, which are present in a .json file. There are a total of 326 features, of which a sweep was made of all of them, reviewing their behavior, what they represented and their main occurrence in the JSON file. Highlighted in yellow are the main features that were selected for their behavior as Ransomware in the pre-, initial and analysis phases. For the final phase, the features highlighted in light blue and purple were selected, giving a total of 64 features selected with an engineering procedure. Once the features have been extracted with the tool developed in this work, 64 dynamic features are selected. According to the analysis, these are the features related to ransomware.

These characteristics extracted for the different artifacts form a matrix that is used to select the most relevant ones using the Mutual Information Matrix method that allows us to detect if a pair of attributes has a high correlation that would lead us to conclude that the information is redundant. In that case, we will choose only one of them. The threshold considered is a 75% pair-wise correlation between attributes.

X and Y are a pair of features with a joint probability mass function p(x,y) and marginal probability mass function p(x) and p(y). The mutual information matrix M(X, Y) is the relative

entropy considering the joint distribution and the product of the marginal distribution as presented in the equation. [131]

$$\mathrm{MI}(X,Y) = \sum_{x\in X}\sum_{y\in Y} p(x,y)\log\frac{p(x,y)}{p(x)p(y)} \quad (8)$$

After which, an automatic feature selection method was used, which is the Mutual Information Matrix and it can be seen which features have a high correlation with each other and these features that are more correlated are the ones that can be removed for selection. So, using this criterion we removed 14 features and worked with the final 50 features for modeling, since they have to do with the behavior of ransomware and have relevance because they are not redundant, since they are not related to others. After this process, the 50 features of the final dataset were obtained.

This process is carried out in this stage for different combinations of characteristics. Each corresponding dataset is evaluated with machine learning algorithms to obtain the optimal number y combination of attributes to generate high-performance models.

Modeling Results with the Dataset Global

Annexes E and F present this process for several feature combinations and their corresponding algorithms' performances for supervised and Neural Networks, including detection times (Runtimes). With this data, the author selected 50 features that yield the best results because they are related to the typical behavior of the analyzed artifacts. We have 50 attributes, 40 artifacts, and ten experiments for each artifact in five victim's device, giving a total of 2000 json files. Because each json file has several rows, this first dataset generated in Phase final has 1'424.344 registers after a cleaning procedure to eliminate redundant rows (Dataset Global).

The previous phases established that the algorithms that produce the best performances are Random Forest and Gradient Boosted Regression Trees. Also, Gaussian Naive Bayes and Neural Networks were included in this experimentation, although the yields are lower with these algorithms. The modeling results for this dataset are presented in Table 31 and Annex G, including detection times (Runtimes), which contains the logs for the generation of the models for Phase final, Dataset Global using supervised algorithms.

In Annex H are detailed the results for the different models obtained with Neural Networks, including detection times (Runtimes), several configurations of layers, and the number of

neurons in each layer. The best configuration results for 3 layers with 200 neurons, sigmoid activation, and softmax output functions are also shown in Table 32. For Random Forest and Gradient Boosted Regression Trees, the best results, without overfitting, are obtained for 100 estimators, i.e., trees in the forest. G, E, and L mean Goodware, Encryptor, and Locker. Gradient Boosted Regression Trees is the algorithm with the best performance, but its processing time is around four hours, which makes it challenging to deploy for the operation cycle.

The metrics used to evaluate the performance of the machine learning algorithms are accuracy, precision, recall, and F1.

Table 32. Performance results for Dataset Global from Phase Final

Algorithm	**Average ten-fold cross-validation Accuracy**	**Precision (%)**			**Recall (%)**			**F1 (%)**			**Processing time (segs.)**
		G	E	L	G	E	L	G	E	L	
Random Forest	99.0	87.40	99.40	96.98	91.11	99.28	93.43	89.25	99.34	85.15	5193.67
Gradient Boosted Regression Trees	98.00	83.00	98.85	98.98	85.19	99.07	90.37	84.08	98.96	94.48	14755.79
Gaussian Naive Bayes	89.00	46.08	92.98	16.47	40.38	96.16	07.19	43.04	94.54	1.00	76.50
Neural Networks	91.92		92.31			90.55			92.12		2804.61

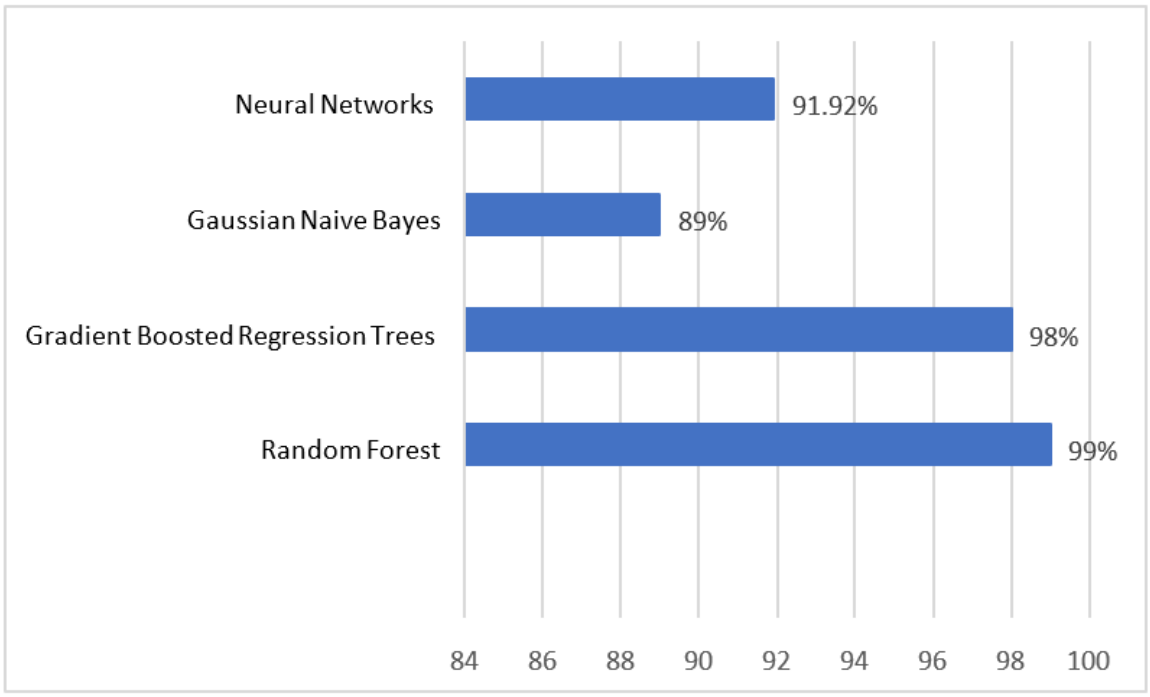


Figure 19. Ten-fold cross-validation accuracy results obtained in Dataset Global from Phase Final.

The Dataset Global generated in Phase final is included in Annex O.

For choosing the best option, we generated datasets for 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, and 50 features. The best results were obtained for 50 features, as shown in Annex K for supervised learning and Annex J for Neural Networks, both include detection times (Runtimes). Therefore, we choose 50 features for this final dataset.

Dataset Extract from Phase final

The results with the dataset shown in the previous section are satisfactory. However, the file size of this dataset produces longer processing times and is neither portable nor efficient to be implemented in the deployment stage. For this reason, the previous dataset was processed to obtain one row for each json file corresponding to an artifact.

For constructing this new dataset used for the generation of machine learning models, we start from the extraction of the previously described JSON content. The following image shows an example the information extracted from a single artifact analyzed, as seen in Figure 20.

urls	hosts	tcp	udp	domains	regkey_written	dll_loaded	file_created
http://127.0.0.1:8000	192.16.58.8	{'src': '192.168.56.50', 'dst': '172.217.2.78', 'offset': 16968, 'time': 96.92912888526917, 'dport': 80, 'sport': 49727}	{'src': '192.168.56.50', 'dst': '224.0.0.252', 'offset': 2107346, 'time': 3.254694938659668, 'dport': 5355, 'sport': 50511}	{'ip': '172.217.8.142', 'domain': 'redirector.gvt1.com'}	N/A	N/A	N/A
http://127.0.0.1:8000/status	201.219.34.141	{'src': '192.168.56.50', 'dst': '192.16.58.8', 'offset': 18351, 'time': 33.49315905570984, 'dport': 80, 'sport': 49688}	{'src': '192.168.56.50', 'dst': '224.0.0.252', 'offset': 2107528, 'time': 1.2354998588562012, 'dport': 5355, 'sport': 56790}	{'ip': '172.217.3.67', 'domain': 'update.googleapis.com'}	N/A	N/A	N/A
http://7z.sparanoid.com/	N/A	{'src': '192.168.56.10', 'dst': '192.16.48.200', 'offset': 115557, 'time': 21.20490002632141, 'dport': 80, 'sport': 1053}	N/A	N/A	HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\Browse For Folder Width	C:\DOCUME~1\WINXP-~1\CONFIG~1\Temp\nsl2D.tmp\LangDLL.dll	C:\7-ZipPortable\App\7-Zip64\descript.ion
http://acraiz.icpbrasil.gov.br/DPCacraiz.pdf0?	N/A	{'src': '192.168.56.40', 'dst': '192.16.58.8', 'offset': 1479898, 'time': 46.68064308166504, 'dport': 80, 'sport': 49690}	{'src': '192.168.56.40', 'dst': '239.255.255.250', 'offset': 1734967, 'time': 6.180221080780029, 'dport': 3702, 'sport': 58726}	{'ip': '52.184.216.246', 'domain': 'geo-prod.do.dsp.mp.microsoft.com'}	N/A	N/A	N/A
http://acraiz.icpbrasil.gov.br/DPCacraiz.pdf0?	186.42.100.145	{'src': '192.168.56.40', 'dst': '172.217.2.78', 'offset': 39040, 'time': 76.06003999710083, 'dport': 80, 'sport': 49712}	{'src': '192.168.56.40', 'dst': '239.255.255.250', 'offset': 1374705, 'time': 20.467479944229126, 'dport': 3702, 'sport': 60140}	N/A	HKEY_CURRENT_USER\Local Settings\MuiCache\1b\63C768CF\LanguageList	N/A	C:\Users\Administrador\AppData\Local\Temp\nsaAF5D.tmp\System.dll
http://acraiz.icpbrasil.gov.br/LCRacraizv2.crl0	N/A	{'src': '192.168.56.40', 'dst': '205.185.216.42', 'offset': 1330387, 'time': 38.0294508934021, 'dport': 80, 'sport': 49680}	{'src': '192.168.56.40', 'dst': '8.8.8.8', 'offset': 1684788, 'time': 63.033758878707886, 'dport': 53, 'sport': 52056}	N/A	HKEY_CURRENT_USER\Software\Microsoft\Speech\AudioOutput\TokenEnums\MMAudioOut\{0.0.0.00000000}.{9af2d554-5f7b-403b-b5e3-983174a2c89c}\(Default)	N/A	c:\tmpjp07qn\lib\common_HELP_HELP_HELP_CT8XG_.hta
http://api.blo	205.185.216.42	{'src': '192.168.56.50', 'dst': '142.250.64.195', 'offset': 5785567, 'time': 43.875773191452026, 'dport': 443, 'sport': 49716}	{'src': '192.168.56.50', 'dst': '8.8.8.8', 'offset': 8182792, 'time': 45.638493061065674, 'dport': 53, 'sport': 52100}	{'ip': '65.8.246.105', 'domain': 'updates.bravesoftware.com'}	HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Internet Settings\5.0\Cache\History\CachePrefix	wship6.dll	c:\tmpyu6hmr\lib\core_HELP_HELP_HELP_UCDURIRT_.hta
http://appmap.trafficmanager.net/api/v1/parse?url=	40.90.137.120	{'src': '192.168.56.50', 'dst': '204.79.197.200', 'offset': 5915784, 'time': 58.11036920547485, 'dport': 443, 'sport': 49730}	{'src': '192.168.56.50', 'dst': '8.8.8.8', 'offset': 8184738, 'time': 38.87125110626221, 'dport': 53, 'sport': 55617}	{'ip': '52.179.216.235', 'domain': 'geo.prod.do.dsp.mp.microsoft.com'}	HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\UNCAsIntranet	wldp.dll	C:\Users\Administrador\AppData\Local\Temp\nsd6038.tmp
http://btc.blo	13.107.246.10	{'src': '192.168.56.50', 'dst': '104.43.193.48', 'offset': 13985, 'time': 44.623791217803955, 'dport': 443, 'sport': 49720}	{'src': '192.168.56.50', 'dst': '224.0.0.251', 'offset': 7660450, 'time': 11.123236179351807, 'dport': 5353, 'sport': 5353}	{'ip': '52.255.188.83', 'domain': 'watson.telemetry.microsoft.com'}	HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Internet Settings\5.0\Cache\Content\CachePrefix	api-ms-win-rtcore-ntuser-wmpointer-l1-1-0.dll	c:\tmpyu6hmr\lib\common_HELP_HELP_HELP_V4A93Z0_.hta
http://ca2.mtin.es/mtin/crl/MTINAutoridadRaiz0	N/A	{'src': '192.168.56.40', 'dst': '40.90.137.124', 'offset': 1508093, 'time': 47.303240060806274, 'dport': 443, 'sport': 49684}	{'src': '192.168.56.40', 'dst': '8.8.8.8', 'offset': 1686576, 'time': 38.581517934799194, 'dport': 53, 'sport': 59059}	N/A	HKEY_CURRENT_USER\Software\Microsoft\Speech\AudioOutput\TokenEnums\MMAudioOut\{0.0.0.00000000}.{9af2d554-5f7b-403b-b5e3-983174a2c89c}\DeviceName	N/A	C:\Users\Administrador\AppData\Local\Temp\tmp2AB6.bmp
http://crl.ssc.lt/root-b/cacrl.crl0	N/A	{'src': '192.168.56.20', 'dst': '188.165.164.184', 'offset': 5697, 'time': 34.832813024520874, 'dport': 80, 'sport': 49172}	{'src': '192.168.56.20', 'dst': '192.168.56.255', 'offset': 15454, 'time': 11.834764957427979, 'dport': 138, 'sport': 138}	{'ip': '', 'domain': 'teredo.ipv6.microsoft.com'}	N/A	CRYPTSP.dll	N/A
http://crl.startcom.org/sfsca-crl.crl0	51.104.167.255	{'src': '192.168.56.40', 'dst': '205.185.216.42', 'offset': 1331911, 'time': 55.969918966293335, 'dport': 80, 'sport': 49699}	{'src': '192.168.56.40', 'dst': '8.8.8.8', 'offset': 1685110, 'time': 67.04167890548706, 'dport': 53, 'sport': 54184}	{'ip': '40.90.137.125', 'domain': 'login.live.com'}	HKEY_CURRENT_USER\Software\Microsoft\Speech\AudioOutput\TokenEnums\MMAudioOut\{0.0.0.00000000}.{9af2d554-5f7b-403b-b5e3-983174a2c89c}\CLSID	N/A	c:\tmpjp07qn\modules\auxiliary_HELP_HELP_HELP_6JW63G7_.hta

Author: Juan A. Herrera Silva

Figure 20. Information for a single artifact

A matrix is created where the columns correspond to each of the characteristics extracted from the analyzed artifact. This way, all the information collected throughout the analysis is grouped. Once this is done, the number of records by columns is counted. A cell with the value “N/A” is not counted. If it has a value other than "N/A," it is calculated. Taking the previous image as an example, the result of the accounting for the artifact in question produces a unique vector. The number of registers for each column in the considered categories is found in each cell.

We proceed to do this with the two thousand experiments of which we have the reports, and we obtain a matrix where each row has information about an artifact, and each row cell corresponds to a feature of that artifact. This process produces a matrix of 2000 rows and 50 columns. This dataset generates the models using machine learning algorithms and is included in Annex P.

Modeling Results with the Dataset Extract

For the machine learning algorithms, we used the parameters specified in section 4.6, which are the ones that produce the best performances. For Random Forest and Gradient Boosted Trees, it is shown the performances for 100 estimators or trees. For Neural Networks, all the models have high performances. We chose one similar to the parameters used for Neural Networks for Dataset Global of Phase final, i.e., with three layers, 100 neurons in each. See Table 33.

However, we selected SELU as an activation function in this case because it runs a little faster. Annex J and K show the complete log with all the experiments for different numbers of attributes and several estimators' values for supervised and Neural Networks.

Table 33. Performance of the classifiers using Dataset Extract for Phase Final

Algorithm	**Average ten-fold cross-validation Accuracy**	**Precision (%)**			**Recall (%)**			**F1 (%)**			**Processing time (segs.)**
		G	E	L	G	E	L	G	E	L	
Random Forest	100	99.86	100	100	100	99.831	100	99.93	99.91	100	3.9
Gradient Boosted Regression Trees	100	99.74	100	100	100	99.66	100	99.86	99.98	100	25.47

Algorithm	Average ten-fold cross-validation Accuracy	Precision (%)			Recall (%)			F1 (%)			Processing time (segs.)
		G	E	L	G	E	L	G	E	L	
Gaussian Naive Bayes	74.00	71.11	88.86	52.43	93.62	58.03	38.29	80.83	70.21	4.26	0.15
Neural Networks	99.8		99.8			99.8			99.8		6.99

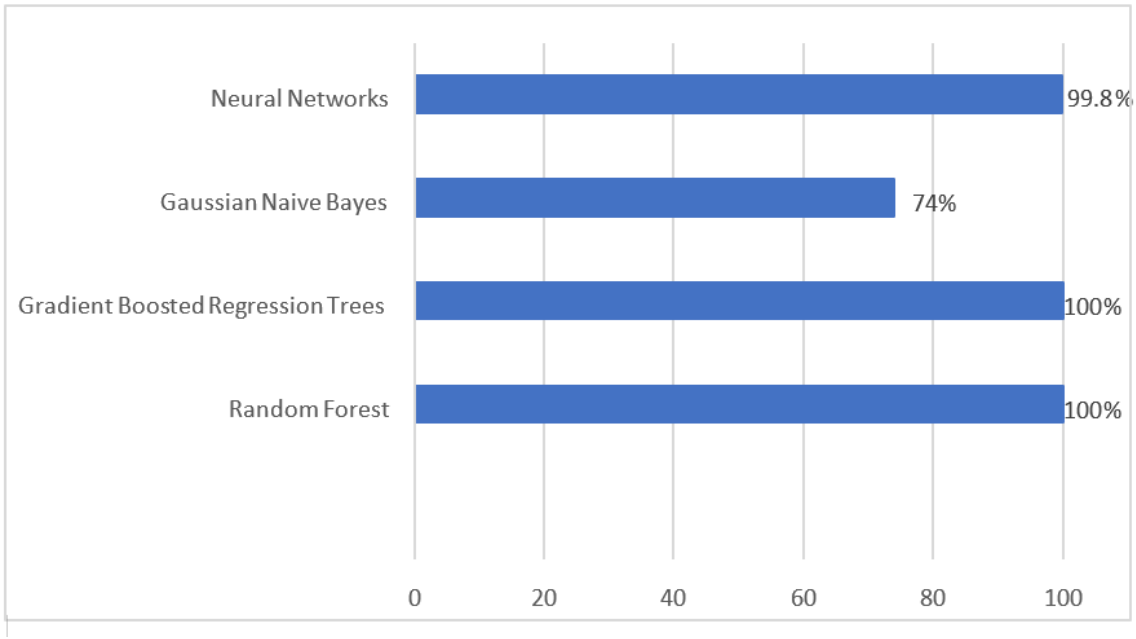


Figure 21. Ten-fold cross-validation accuracy using the Dataset Extract from Phase Final.

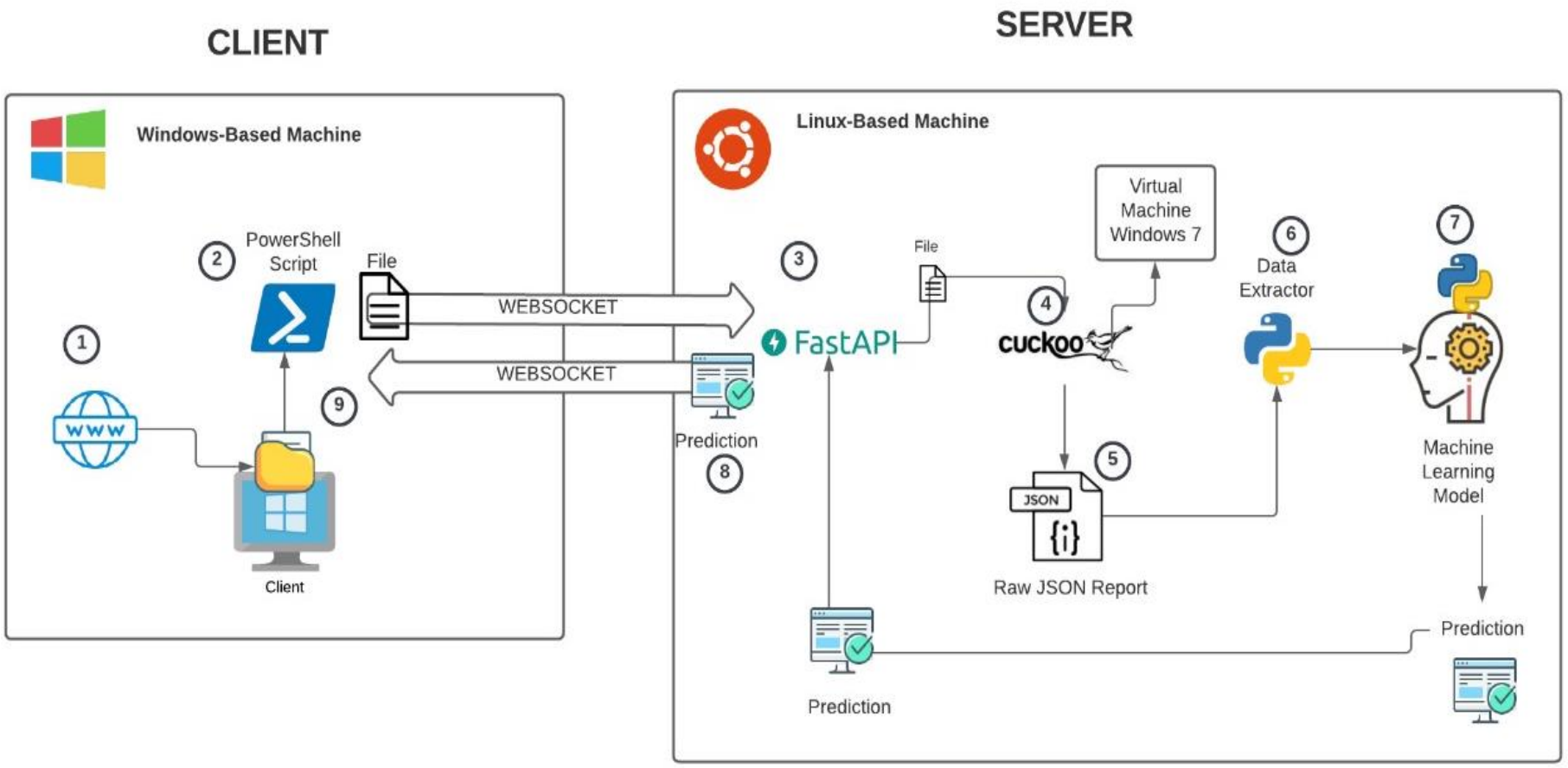
The best results are obtained with the second dataset, as seen in Figure 21. Also, processing times for the model obtaining are significantly lower than with the previous dataset. Again, the best performance algorithms are Random Forest and Gradient Boosted Regression Trees, and slightly lesser values were obtained using Neural Networks with three layers with 100 neurons each. Bayes reduces performance values from 89 obtained in Dataset Global to 74 obtained with the Dataset Extract for 10-fold cross-validation accuracy.

5.3. Deployment

The prediction of new artifacts requires generating a csv file with the previously described tool. Once you have the corresponding csv file, we use the ml_predictor.py and dl_predictor.py programs to make predictions with any generated models, whether in the

repository or not. The content of these files is concise enough to change the directories of csv files and models to execute the deployment.

Our architecture allows analyzing the behavior of an artifact since it is created in a file system. It considers the sandbox environment for the dynamic analysis of an artifact, the information extraction tool obtained from the analysis, and the machine learning models to be used to classify the analyzed artifact, as shown in Figure 22.



Author: Juan A. Herrera Silva

Figure 22. Deployment architecture

The process of analyzing an artifact by deploying the models is detailed below:

1. A file is introduced into the computer, for example, through a network.

2. Using a Powershell script, the introduction (creation) of the file to the file system of the operating system is detected.

3. Using the Powershell script, the client opens a WebSocket-type connection with the server and proceeds to send the file in question.

4. Once it has received the entire file, the server starts the dynamic analysis process using the cuckoo sandbox tool.

5. After completing the dynamic analysis process, Cuckoo Sandbox collects all the information and saves it in a file in json format.

6. Once the creation of this file is detected, a variation of the information extraction tool is used to extract the relevant information that will serve as input for the machine learning models.

7. Once the information has been extracted, the feature vector is built and sent to one of the previously trained machine learning models to obtain the classification (prediction) of the analyzed file.

8. The classification (prediction) provided by the model is sent through the WebSocket connection to the client to take actions depending on whether it is Ransomware or not.

Repository Content

Inside the repository, there are two folders:

1. The filter folder contains the source code of the information extraction tool and the dataset generated under which the Machine Learning and Neural Networks models were developed. This folder has the following files:

    • filter.py: Source code of the program in charge of extracting the information from one or more json files obtained with cuckoo.

• gui.py: Source code of the graphical interface that invokes the methods of the file filter.py

• simplified_dataset_shuffled.csv: The dataset was built from the extraction of information from the two thousand JSON files obtained in the initial experimentation

2. Machine Learning directory that contains two folders for Machine Learning and Neural Networks Logs.

   Machine Learning - Logs:

   - Logs_ML.xslx: This file contains a table with information regarding the generated Machine Learning models and the results obtained.
   - my_dataset_ml.txt: This file contains more information about each of the generated models.
   - Models: This folder contains the generated machine learning models that can be used in conjunction with the files to predict new artifacts.
   - ml_predictor.py: Python script that can be used to classify new artifacts.

   Neural Networks - Logs:

   - Logs_DL_NewDataset.ods: This file contains a table with information regarding the generated Neural Networks models and the results obtained.
   - new_dataset_dl_logs.txt: This file contains more information about each generated model.
   - Models: This folder contains the generated Neural Networks models that can be used in conjunction with the files to predict new artifacts.
   - Predictions: Contains prediction results of new artifacts from some previously generated Neural Networks models.
   - dl_predictor.py: Python script that can be used to classify new artifacts.

6. DISCUSSION

6.1. Contributions of This Work

From Tables 4 and 13, comparing the characteristics of other research with the present work, it can be inferred that our experiment has several advantages:

- ✓ Unlike all the other studies analyzed in the Related Work section, which only use around three types of features, our research uses the full range of related attributes to study artifacts. This full use of different characteristics allows for the recognition of behavior patterns common to ransomware. Therefore, even new variants not initially present in the training set can be detected.

  The attributes used by this research are:

  - PROCMEMORY: memory management information;
  - EXTRACTED: information on executed scripts;
  - NETWORK: network data;
  - SIGNATURES: predefined patterns that might represent malicious behavior;
  - STATIC: static analysis data, including entropy level obtained by the cuckoo sandbox software;
  - BEHAVIOR: libraries to which the artifact makes calls, suspicious processes, and affected registry keys;
  - DEBUG: actions, errors, and log information recorded during the dynamic analysis.

- ✓ Table 4 shows that most researchers use only a fraction of all possible types of features available in dynamic analysis, for example, attributes related to the network or API calls that are part of the behavior parameters. For better classification results, it is necessary to use a more complete description of the ransomware activities delineated by the presence of all the related types of dynamic features. For this reason, we chose 50 attributes related to the before-mentioned group of dynamic parameters related to ransomware effects.

- ✓ The 10-fold cross-validation accuracy, precision, recall, and F1 values obtained with the final dataset, using random forest and gradient boosted regression trees, are practically perfect, ensuring the threat's detection with a processing time in the range of seconds.

Other studies have detection results comparable to or lower than the ones obtained in our research.

- ✓ The dataset that our study delivers are a feature dataset; that is, it is information that is already ready to be used as input to a machine learning classifier to obtain models that can be tested on new data to be categorized. Most studies only mention the ransomware sample sources, e.g., VirusTotal, and the number of ransomware and goodware used in their datasets; they do not deliver their samples dataset nor the features dataset generated with their work. Unlike other studies, we present in the paper the information we produced in a GitHub repository for community use (https://github.com/Juan-Herrera-Silva/PaperSENSORS, accessed on 2 December 2022).

- ✓ The fact that we apply machine learning gives flexibility to our research because this technique allows for the discovery of hidden patterns in the ransomware behavior. Because this study uses the full range of relevant dynamic features without redundant information (with low correlation pairwise), it generates models that recognize patterns corresponding to the locker and crypto-ransomware variants not present in the training set.

- ✓ The time it takes for our classifiers to process the samples is in the order of seconds, making it possible to detect the threat and stop it before any damage is achieved.

- ✓ The range of platforms used for our study is more complete than the ones used in other studies. The sandbox implementation is executed in Windows XP, Windows 7 Ultimate, Windows 7 Professional, Windows 10 Enterprise, and Windows 10 Professional.

7. CONCLUSIONS

The fact that ransomware attacks continue to produce millions in losses worldwide shows that there is much room for improvement in ransomware detection. The present work contributes to some of the still open issues. One of these issues is the necessity of a dataset containing features corresponding to all the ransomware attack patterns that could be used to train supervised algorithms and neural network models. This feature dataset should include all the relevant attributes related to the threat's behavior and should be open for the development of new machine learning ransomware detection solutions. Our work aims in that direction.

In the present research, the authors have developed a dataset composed of the dynamic features of locker and encryptor ransomware and characteristics extracted from goodware. The features were selected with the criteria that they are related to the effects of ransomware. In the literature, it was found that a ransomware dataset with these characteristics was needed because the ones that are publicly accessible do not have dynamic features of the artifacts but only fixed signatures, or their results are challenging to replicate or use for lack of enough descriptive information.

Dynamic analysis is essential for ransomware detection because the run-time attributes have enough information for machine learning early detection of these threats. In our study, since most of these features are shared by diverse ransomware samples, our dynamic analysis can be used even for detecting new variants. For dynamic analysis, the experimentation must be conducted in an isolated environment to protect the network from using a sandbox for artifact execution. For this purpose, cuckoo sandbox was used to create JSON files with nested information of the dynamic features. The features were selected using criteria related to the role of each attribute in the ransomware attacks and the results of experimentation with machine learning algorithms aiming to obtain the best performances. The JSON file's total number of features was 326, and the chosen characteristics were 50.

In Phase Previous, we extracted seven features and tested them on five ransomware artifacts over two platforms to obtain a dataset of 6783 rows. In Phase initial, we added more ransomware families and goodware artifacts for dataset balancing, with a total of 24 families to get a 47959 register dataset.

In Phase Analysis, we started feature engineering testing combinations of features to obtain the performance of the machine learning algorithms with each one. We found that the best performance was consistently yielded using Random Forest, Gradient Boosted Trees, and Neural Networks algorithms. In this stage, using combinations of 14 features, we tested 24 families of artifacts over the same two platforms and generated a 62989 record dataset.

On the other hand, when other authors use dynamic features, they only use some attributes, for example, attributes related to the network, API and DLL calls, or file systems. For better classification results that even detect variants not included in the training set, it is necessary to use a more complete description of the ransomware activities delineated by the presence of all the relevant dynamic features.

In Phase Final, we developed two features datasets, Dataset Global and Dataset Extract. This research has gone through two steps to categorize three classes: locker ransomware, encryptor ransomware, and goodware.

The first step, using our dynamic feature extraction tool, the features were tested, and 50 characteristics were selected because they comply with criteria related to ransomware attacks. They were also tested to have a low pairwise correlation to avoid redundant information. In the trials, the study found that high performances for the machine learning algorithms were obtained for these 50 characteristics and the machine learning algorithms mentioned in Section 5. The researchers used 20 ransomware artifacts and 20 goodware families tested with ten experiments, each over five platforms, to produce a dataset named Dataset Global with 1’424.344 rows. For this dataset, there were several rows corresponding to one JSON. The best performance results were obtained with gradient boosted regression trees with values of 0.98 for 10-fold cross-evaluation accuracy. However, processing times for machine learning model generation were high because it took in the range of 4 h to obtain the models.

The second step, to generate a more portable, efficient, and concise dataset without losing relevant information, the research developed a process for synthesizing all the rows corresponding to one JSON into one row. This way, using the information provided for the previous repository, the study obtained a second dataset named Dataset Extract with 2000 records, corresponding to forty families and ten experiments for each artifact over five platforms. Using this dataset, performance results for our models improved even more for gradient boosted regression trees, random forest, and neural networks because they

reached values close to perfect detection for ransomware. The reported accuracy presented in the literature for ransomware detection gives 0.997 as a maximum value; thus, our models have comparable or better performance. Additionally, processing times were reduced from hours (using the Dataset Global) to seconds using the summary dataset (Dataset Extract).

In the deployment, predicting new artifacts requires applying the generated models, whether in the repository or not. The programs allow changing the directories of csv json files and models to execute them in the production stage.

This dataset is available for public access along with the present article and in the GitHub repository[16]. The dataset we deliver will allow the researchers to summarize which malware parameters affect a system more. Therefore, this information can be used as a starting point for generating new methods of detecting ransomware. As the dataset will be of public access, the scientific community can improve, modify, and share this knowledge.

The present research's objectives and hypothesis were achieved and confirmed.

[16] https://github.com/Juan-Herrera-Silva/Paper-SENSORS

8. FUTURE WORK

Ransomware detection will remain one of the highest priority challenges for individuals and organizations for the years to come. Therefore, it is still necessary to find solutions that are really effective and can stop these attacks that are constantly evolving with new variants.

Our approach using machine-learning models and ransomware feature datasets allows this detection to occur. However, it still requires work for general application outside the lab. As future work, we consider it is necessary to develop an application that can be executed in real-time to generate a program that obtains the JSON files using the cuckoo structure of new samples to form a feature vector. These feature vectors would be entered into the machine learning models to quickly detect the virus before it starts to encrypt the files. This requires a response time of fewer than 45 minutes because, according to Microsoft[17], close to 97% of all ransomware infections take less than 4 hours to successfully infiltrate their target. The fastest can take over systems in less than 45 minutes.

Additionally, the feature dataset should be constantly updated with new ransomware available data to produce machine-learning models capable of responding effectively to this threat. As the final feature dataset is public access, the author hopes the scientific community can use, improve, modify, and share this knowledge.

[17] https://blogs.microsoft.com/on-the-issues/2020/09/29/m icrosoft-digital-defense-report-cyber-threats/

REFERENCES

[1] W. X. Tee, W. Jung, and A. A. Radlet, “ASEAN Cyberthreat Assessment Report 2020,” *Interpol*, p. 30, 2020, [Online]. Available: https://www.interpol.int/en/content/download/14922/file/ASEAN_CyberThreatAssessment_2020.pdf.

[2] H. Oz, A. Aris, A. Levi, and A. S. Uluagac, “A Survey on Ransomware: Evolution, Taxonomy, and Defense Solutions,” *ACM Comput. Surv.*, vol. 1, no. 1, 2022, doi: 10.1145/3514229.

[3] N. A. Hassan, “Ransomware Revealed: A Beginner’s Guide to Protecting and Recovering from Ransomware Attacks,” *Access, IEEE*, p. 2, 2019.

[4] R. Richardson and M. North, “Ransomware: Evolution, Mitigation and Prevention,” *Int. Manag. Rev.*, vol. 13, no. 1, pp. 10–21, 2017.

[5] D. Gonzalez and T. Hayajneh, “Detection and prevention of crypto-ransomware,” *2017 IEEE 8th Annu. Ubiquitous Comput. Electron. Mob. Commun. Conf. UEMCON 2017*, vol. 2018-Janua, pp. 472–478, 2017, doi: 10.1109/UEMCON.2017.8249052.

[6] J. Dimaggio, “A history of Revil,” vol. 1, no. January 27, pp. 1–6, 2022.

[7] M. A. Sotelo Monge, J. M. Vidal, and L. J. García Villalba, “A novel self-organizing network solution towards crypto-ransomware mitigation,” *ACM Int. Conf. Proceeding Ser.*, 2018, doi: 10.1145/3230833.3233249.

[8] Group IB Experts, “Ransomware Uncovered,” no. March, pp. 1–22, 2021.

[9] S. Gadhiya, K. Bhavsar, and P. D. Student, “Techniques for Malware Analysis,” *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 3, no. 4, pp. 2277–128, 2013.

[10] M. Sikorski and A. Honig, *Practical malware analysis: the hands-on guide to dissecting malicious software. no starch press*. 2012.

[11] A. Ray and A. Nath, “International Journal of Advance Research in Computer Science and Management Studies Introduction to Malware and Malware Analysis: A brief overview,” no. November, 2016, [Online]. Available: www.ijarcsms.com.

[12] Sophos, “The State of Ransomware.” pp. 1–21, 2021.

[13] J. A. H. Silva, L. I. B. López, Á. L. V. Caraguay, and M. Hernández-álvarez, “A survey on situational awareness of ransomware attacks-detection and prevention parameters,” *Remote Sens.*, vol. 11, no. 10, 2019, doi: 10.3390/rs11101168.

[14] Monika, P. Zavarsky, and D. Lindskog, “Experimental Analysis of Ransomware on Windows and Android Platforms: Evolution and Characterization,” *Procedia Comput. Sci.*, vol. 94, pp. 465–472, 2016, doi: 10.1016/j.procs.2016.08.072.

[15] D. Sgandurra, L. Muñoz-González, R. Mohsen, and E. C. Lupu, “Automated Dynamic Analysis of Ransomware: Benefits, Limitations and use for Detection,” 2016, [Online]. Available: http://arxiv.org/abs/1609.03020.

[16] A. Kharraz, S. Arshad, C. Mulliner, W. Robertson, and E. Kirda, “A Large-Scale, Automated Approach to Detecting Ransomware,” *Metalurgija*, vol. 54, no. 1, pp. 286–288, 2016.

[17] A. B. KARDILE, “Crypto ransomware analysis and detection using process monitor,” no. December, pp. 1–14, 2017.

[18] S. Jung and Y. Won, “Ransomware detection method based on context-aware entropy analysis,” *Soft Comput.*, vol. 22, no. 20, pp. 6731–6740, 2018, doi: 10.1007/s00500-018-3257-z.

[19] K. Cabaj, M. Gregorczyk, and W. Mazurczyk, “Software-defined networking-based crypto ransomware detection using HTTP traffic characteristics,” *Comput. Electr. Eng.*, vol. 66, pp. 353–368, 2018, doi: 10.1016/j.compeleceng.2017.10.012.

[20] E. Kolodenker, W. Koch, G. Stringhini, and M. Egele, “PayBreak : Defense against cryptographic ransomware,” *ASIA CCS 2017 - Proc. 2017 ACM Asia Conf. Comput. Commun. Secur.*, pp. 599–611, 2017, doi: 10.1145/3052973.3053035.

[21] Y. Takeuchi, K. Sakai, and S. Fukumoto, “Detecting ransomware using support vector machines,” *ACM Int. Conf. Proceeding Ser.*, pp. 1–6, 2018, doi: 10.1145/3229710.3229726.

[22] A. M. Maigida, S. M. Abdulhamid, M. Olalere, J. K. Alhassan, H. Chiroma, and E. G.

Dada, “Systematic literature review and metadata analysis of ransomware attacks and detection mechanisms,” *J. Reliab. Intell. Environ.*, vol. 5, no. 2, pp. 67–89, 2019, doi: 10.1007/s40860-019-00080-3.

[23] S. Mehnaz, A. Mudgerikar, and E. Bertino, *RWGuard: A real-time detection system against cryptographic ransomware*, vol. 11050 LNCS, no. March 2019. Springer International Publishing, 2018.

[24] Z. G. Chen, H. S. Kang, S. N. Yin, and S. R. Kim, “Automatic ransomware detection and analysis based on dynamic API calls flow graph,” *Proc. 2017 Res. Adapt. Converg. Syst. RACS 2017*, vol. 2017-Janua, pp. 196–201, 2017, doi: 10.1145/3129676.3129704.

[25] G. Cusack, O. Michel, and E. Keller, “Machine learning-based detection of ransomware using SDN,” *SDN-NFVSec 2018 - Proc. 2018 ACM Int. Work. Secur. Softw. Defin. Networks Netw. Funct. Virtualization, Co-located with CODASPY 2018*, vol. 2018-Janua, pp. 1–6, 2018, doi: 10.1145/3180465.3180467.

[26] K. P. Subedi, D. R. Budhathoki, and D. Dasgupta, “Forensic analysis of ransomware families using static and dynamic analysis,” *Proc. - 2018 IEEE Symp. Secur. Priv. Work. SPW 2018*, pp. 180–185, 2018, doi: 10.1109/SPW.2018.00033.

[27] O. M. K. Alhawi, J. Baldwin, and A. Dehghantanha, “Leveraging Machine Learning Techniques for Windows Ransomware Network Traffic Detection,” vol. 70, pp. 1–11, 2018, doi: 10.1017/CBO9781107415324.004.

[28] H. Zhang, X. Xiao, F. Mercaldo, S. Ni, F. Martinelli, and A. K. Sangaiah, “Classification of ransomware families with machine learning based on N-gram of opcodes,” *Futur. Gener. Comput. Syst.*, vol. 90, pp. 211–221, 2019, doi: 10.1016/j.future.2018.07.052.

[29] M. Hasan, M., Rahman, “RansHunt_ A support vector machines based ransomware analysis framework with integrated feature set - IEEE Conference Publication.pdf,” pp. 22–24, 2017.

[30] T. Lu, L. Zhang, S. Wang, and Q. Gong, “Ransomware detection based on V-detector negative selection algorithm,” in *2017 International Conference on Security, Pattern*

Analysis, and Cybernetics (SPAC), Dec. 2017, vol. 2018-Janua, pp. 531–536, doi: 10.1109/SPAC.2017.8304335.

[31] S. K. Shaukat and V. J. Ribeiro, “This paper is a preprint (IEEE ‘ accepted ’ status). IEEE copyright notice : RansomWall : A Layered Defense System against Cryptographic Ransomware Attacks using Machine Learning,” pp. 356–363, 2018.

[32] 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. 119710–119719, 2020, doi: 10.1109/ACCESS.2020.3003785.

[33] I. Bello *et al.*, “Detecting ransomware attacks using intelligent algorithms: recent development and next direction from Neural Networks and big data perspectives,” *J. Ambient Intell. Humaniz. Comput.*, vol. 11, no. 10, p. 1168, Nov. 2020, doi: 10.1007/s12652-020-02630-7.

[34] R. Enbody, A. K. Sood, and P. Bajpai, “A key-management-based taxonomy for ransomware,” *eCrime Res. Summit, eCrime*, vol. 2018-May, pp. 1–12, 2018, doi: 10.1109/ECRIME.2018.8376213.

[35] M. Conti, A. Gangwal, and S. Ruj, “On the economic significance of ransomware campaigns: A Bitcoin transactions perspective,” *Comput. Secur.*, vol. 79, pp. 162–189, 2018, doi: 10.1016/j.cose.2018.08.008.

[36] J. Hernandez-Castro, E. Cartwright, and A. Stepanova, “Economic Analysis of Ransomware,” *SSRN Electron. J.*, pp. 1–14, 2017, doi: 10.2139/ssrn.2937641.

[37] K. Gangwar, S. Mohanty, and A. K. Mohapatra, *Analysis and detection of ransomware through its delivery methods*, vol. 799. Springer Singapore, 2018.

[38] C. Moore, “Detecting ransomware with honeypot techniques,” *Proc. - 2016 Cybersecurity Cyberforensics Conf. CCC 2016*, pp. 77–81, 2016, doi: 10.1109/CCC.2016.14.

[39] A. Zahra and M. A. Shah, “IoT based ransomware growth rate evaluation and detection using command and control blacklisting,” *ICAC 2017 - 2017 23rd IEEE Int. Conf. Autom. Comput. Addressing Glob. Challenges through Autom. Comput.*, no. September, pp. 7–8, 2017, doi: 10.23919/IConAC.2017.8082013.

[40] S. Sheen and A. Yadav, "Ransomware detection by mining API call usage," 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2018, pp. 983-987, doi: 10.1109/ICACCI.2018.8554938.

[41] N. Hampton, Z. Baig, and S. Zeadally, “Ransomware behavioural analysis on windows platforms,” *J. Inf. Secur. Appl.*, vol. 40, pp. 44–51, 2018, doi: 10.1016/j.jisa.2018.02.008.

[42] JI, Zhou; D. DASGUPTA, "V-detector: An efficient negative selection algorithm with probably adequate detector coverage". Information sciences, 2009, vol. 179, no 10, p. 1390-1406. [43] L. P. Sendagorta, “Optimal Power Flow for Distribution Networks,” no. July, pp. 2–4, 2017, [Online]. Available: https://www.iit.comillas.edu/pfc/resumenes/595bd3755cc6c.pdf.

[43] V. B. Singh and K. K. Chaturvedi, "Entropy based bug prediction using support vector regression," *2012 12th International Conference on Intelligent Systems Design and Applications (ISDA)*, 2012, pp. 746-751, doi: 10.1109/ISDA.2012.6416630.

[44] B. A. S. Al-Rimy, M. A. Maarof, Y. A. Prasetyo, S. Z. M. Shaid, and A. F. M. Ariffin, “Zero-day aware decision fusion-based model for crypto-ransomware early detection,” *Int. J. Integr. Eng.*, vol. 10, no. 6, pp. 82–88, 2018, doi: 10.30880/ijie.2018.10.06.011.

[45] M. Alam, S. Sinha, S. Bhattacharya, S. Dutta, D. Mukhopadhyay, and A. Chattopadhyay, “RAPPER: Ransomware Prevention via Performance Counters.” 2020, [Online]. Available: http://arxiv.org/abs/2004.01712.

[46] J. A. Gómez-Hernández, L. Álvarez-González, and P. García-Teodoro, “R-Locker: Thwarting ransomware action through a honeyfile-based approach,” *Comput. Secur.*, vol. 73, pp. 389–398, 2018, doi: 10.1016/j.cose.2017.11.019.

[47] K. Cabaj, M. Gregorczyk, and W. Mazurczyk, “Software-defined networking-based crypto ransomware detection using HTTP traffic characteristics,” *Comput. Electr. Eng.*, vol. 66, pp. 353–368, 2018, doi: 10.1016/j.compeleceng.2017.10.012.

[48] A. SOURI, R. HOSSEINI, "A state-of-the-art survey of malware detection approaches using data mining techniques". Human-centric Computing and Information Sciences,

2018, vol. 8, no 1, p. 1-22.

[49] T. Inoue, K. Hasegawa, M. Yanagisawa and N. Togawa, "Designing hardware trojans and their detection based on a SVM-based approach," 2017 IEEE 12th International Conference on ASIC (ASICON), 2017, pp. 811-814, doi: 10.1109/ASICON.2017.8252600.

[50] Ö. Aslan and R. Samet, "Investigation of Possibilities to Detect Malware Using Existing Tools," 2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA), 2017, pp. 1277-1284, doi: 10.1109/AICCSA.2017.24.

[51] S. Maniath, A. Ashok, P. Poornachandran, and S. Jan, “Deep Learning LSTM based Ransomware Detection.,” vol. 3.

[52] A. PEKTAS, T. ACARMAN, "Malware classification based on API calls and behaviour analysis". IET Information Security, 2018, vol. 12, no 2, p. 107-117.

[53] G. KRISHNA, V. RADHA, K. RAO, V. Gopala, "Evolutionary Binary Classification using Cuckoo Search for Malware Perception in API Call Sequences". En *2017 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*. IEEE, 2017. p. 1-8.

[54] J. E. Thomas and G. C. Galligher, “Improving Backup System Evaluations in Information Security Risk Assessments to Combat Ransomware,” *Comput. Inf. Sci.*, vol. 11, no. 1, p. 14, 2018, doi: 10.5539/cis.v11n1p14.

[55] R. Vinayakumar, K. P. Soman, K. K. S. Velan, and S. Ganorkar, “Evaluating shallow and deep networks for ransomware detection and classification,” *2017 Int. Conf. Adv. Comput. Commun. Informatics, ICACCI 2017*, vol. 2017-Janua, pp. 259–265, 2017, doi: 10.1109/ICACCI.2017.8125850.

[56] I. Kwon and E. G. Im, “Extracting the representative API call patterns of malware families using recurrent neural network,” *Proc. 2017 Res. Adapt. Converg. Syst. RACS 2017*, vol. 2017-Janua, pp. 202–207, 2017, doi: 10.1145/3129676.3129712.

[57] P. O’Kane, S. Sezer, and D. Carlin, “Evolution of ransomware,” *IET Networks*, vol. 7, no. 5, pp. 321–327, 2018, doi: 10.1049/iet-net.2017.0207.

[58] A. K. Maurya, N. Kumar, A. Agrawal, and R. A. Khan, “Ransomware Evolution, Target and Safety Measures,” *Int. J. Comput. Sci. Eng.*, vol. 6, no. 1, pp. 80–85, 2018, doi: 10.26438/ijcse/v6i1.8085.

[59] M. MONGE, J. VIDAL, L. VILLALBA, "A novel self-organizing network solution towards crypto-ransomware mitigation". *En Proceedings of the 13th International Conference on Availability, Reliability and Security*. 2018. p. 1-10.

[60] B. A. S. Al-rimy, M. A. Maarof, and S. Z. M. Shaid, “A 0-day aware crypto-ransomware early behavioral detection framework,” *Lect. Notes Data Eng. Commun. Technol.*, vol. 5, pp. 758–766, 2018, doi: 10.1007/978-3-319-59427-9_78.

[61] B. ZHANG, et al. "Ransomware classification using patch-based CNN and self-attention network on embedded N-grams of opcodes". *Future Generation Computer Systems,* 2020, vol. 110, p. 708-720.

[62] A. Cimitile, F. Mercaldo, V. Nardone, A. Santone, and C. A. Visaggio, “Talos: no more ransomware victims with formal methods,” *Int. J. Inf. Secur.*, vol. 17, no. 6, pp. 719–738, 2018, doi: 10.1007/s10207-017-0398-5.

[63] D. Y. Huang *et al.*, “Tracking Ransomware End-to-end,” *Proc. - IEEE Symp. Secur. Priv.*, vol. 2018-May, no. 1, pp. 618–631, 2018, doi: 10.1109/SP.2018.00047.

[64] A. Cohen and N. Nissim, “Trusted detection of ransomware in a private cloud using machine learning methods leveraging meta-features from volatile memory,” *Expert Syst. Appl.*, vol. 102, pp. 158–178, 2018, doi: 10.1016/j.eswa.2018.02.039.

[65] S. Homayoun *et al.*, “DRTHIS: Deep ransomware threat hunting and intelligence system at the fog layer,” *Futur. Gener. Comput. Syst.*, vol. 90, pp. 94–104, 2019, doi: 10.1016/j.future.2018.07.045.

[66] Z. H. Wang, C. G. Liu, J. Qiu, Z. H. Tian, X. Cui, and S. Su, “Automatically Traceback RDP-Based Targeted Ransomware Attacks,” *Wirel. Commun. Mob. Comput.*, vol. 2018, 2018, doi: 10.1155/2018/7943586.

[67] B. A. S. Al-rimy, M. A. Maarof, and S. Z. M. Shaid, “Ransomware threat success factors, taxonomy, and countermeasures: A survey and research directions,” *Comput. Secur.*, vol. 74, pp. 144–166, 2018, doi: 10.1016/j.cose.2018.01.001.

[68] A. El-Kosairy and M. A. Azer, “Intrusion and ransomware detection system,” *2018 1st Int. Conf. Comput. Appl. Inf. Secur.*, pp. 1–7, 2018, doi: 10.1109/cais.2018.8471688.

[69] D. Y. Kim, G. Y. Choi, and J. H. Lee, “White list-based ransomware real-time detection and prevention for user device protection,” *2018 IEEE Int. Conf. Consum. Electron. ICCE 2018*, vol. 2018-Janua, pp. 1–5, 2018, doi: 10.1109/ICCE.2018.8326119.

[70] T. Honda, K. Mukaiyama, T. Shirai, T. Ohki, and M. Nishigaki, “Ransomware detection considering user’s document editing,” *Proc. - Int. Conf. Adv. Inf. Netw. Appl. AINA*, vol. 2018-May, pp. 907–914, 2018, doi: 10.1109/AINA.2018.00133.

[71] J. Saleem, B. Adebisi, R. Ande, and M. Hammoudeh, “A state of the art survey - Impact of cyber attacks on SME’s,” *ACM Int. Conf. Proceeding Ser.*, vol. Part F1305, 2017, doi: 10.1145/3102304.3109812.

[72] A. Kamal *et al.*, “A User-friendly Model for Ransomware Analysis Using Sandboxing,” *Comput. Mater. Contin.*, vol. 67, no. 3, pp. 3833–3846, 2021, doi: 10.32604/cmc.2021.015941.

[73] L. Alhathally and E. Alsuwat, “RANSOMWARE ATTACK DETECTION AND PREVENTION,” 2020.

[74] U. Urooj, B. A. S. Al-rimy, A. Zainal, F. A. Ghaleb, and M. A. Rassam, “Ransomware Detection Using the Dynamic Analysis and Machine Learning: A Survey and Research Directions,” *Appl. Sci.*, vol. 12, no. 1, p. 172, Dec. 2021, doi: 10.3390/app12010172.

[75] A. H. Azizan *et al.*, “A machine learning approach for improving the performance of network intrusion detection systems,” *Ann. Emerg. Technol. Comput.*, vol. 5, no. Special issue 5, pp. 201–208, 2021, doi: 10.33166/AETiC.2021.05.025.

[76] H. Nitsch, J. Hernandez-castro, and D. Hurley-smith, “RAMSES2020 – Internet Forensic platform for tracking the money flow of financially-motivated malware,” pp. 141–146, [Online]. Available: http://ramses2020.eu/.

[77] A. Couce and D. Ríos, “Delft University of Technology CYBECO Supporting cyber-

insurance from a behavioural choice perspective," 2019.

[78] cuckoosandbox, "Automated Malware Analysis," *december 12*, 2018. .

[79] K. Cabaj and W. Mazurczyk, "Using software-defined networking for ransomware mitigation: The case of cryptowall," *IEEE Netw.*, vol. 30, no. 6, pp. 14–20, 2016, doi: 10.1109/MNET.2016.1600110NM.

[80] S. Pletinckx, C. Trap, and C. Doerr, "Malware coordination using the blockchain: An analysis of the cerber ransomware," *2018 IEEE Conf. Commun. Netw. Secur. CNS 2018*, pp. 1–9, 2018, doi: 10.1109/CNS.2018.8433199.

[81] M. Almgren, V. Gulisano, and F. Maggi, "Cutting the Gordian Knot: A Look Under the Hood of Ransomware Attacks," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9148, pp. 3–24, 2015, doi: 10.1007/978-3-319-20550-2.

[82] H. Zuhair, A. Selamat, and O. Krejcar, "A multi-tier streaming analytics model of 0-day ransomware detection using machine learning," *Appl. Sci.*, vol. 10, no. 9, 2020, doi: 10.3390/app10093210.

[83] H. Oz, A. Aris, A. Levi, and A. S. Uluagac, "A Survey on Ransomware: Evolution, Taxonomy, and Defense Solutions," *ACM Comput. Surv.*, 2022, doi: 10.1145/3514229.

[84] A. Patel and J. Tailor, "A malicious activity monitoring mechanism to detect and prevent ransomware," *Comput. Fraud Secur.*, vol. 2020, no. 1, pp. 14–19, 2020, doi: 10.1016/S1361-3723(20)30009-9.

[85] A. Alzahrani, A. Alshehri, H. Alshahrani, and H. Fu, "Ransomware in Windows and Android Platforms." 2020, [Online]. Available: http://arxiv.org/abs/2005.05571.

[86] M. I. Eduardo Berrueta, Daniel Morató , Eduardo Magaña, "Open Repository for the Evaluation of Ransomware Detection Tools," *February 24*, 2020. https://ieee-dataport.org/open-access/open-repository-evaluation-ransomware-detection-tools.

[87] Y. A. Ahmed, B. Koçer, S. Huda, B. A. Saleh Al-rimy, and M. M. Hassan, "A system call refinement-based enhanced Minimum Redundancy Maximum Relevance

method for ransomware early detection," *J. Netw. Comput. Appl.*, vol. 167, p. 102753, 2020, doi: 10.1016/j.jnca.2020.102753.

[88] S. Il Bae, G. Bin Lee, and E. G. Im, "Ransomware detection using machine learning algorithms," *Concurr. Comput. Pract. Exp.*, vol. 32, no. 18, pp. 1–11, 2020, doi: 10.1002/cpe.5422.

[89] K. C. Roy and Q. Chen, "DeepRan: Attention-based BiLSTM and CRF for Ransomware Early Detection and Classification," *Inf. Syst. Front.*, vol. 23, no. 2, pp. 299–315, 2021, doi: 10.1007/s10796-020-10017-4.

[90] H. Zhou, G. Yang, Y. Xu, and W. Wang, *Science of Cyber Security*, vol. 11933. 2019.

[91] S. H. Kok, A. Azween, and N. Z. Jhanjhi, "Evaluation metric for crypto-ransomware detection using machine learning," *J. Inf. Secur. Appl.*, vol. 55, p. 102646, 2020, doi: 10.1016/j.jisa.2020.102646.

[92] "Rahman, F. RanStop A Hardware-assisted Runtime Crypto-Ransomware Detection Technique.pdf." .

[93] S. Poudyal and D. Dasgupta, "AI-Powered Ransomware Detection Framework," *2020 IEEE Symp. Ser. Comput. Intell. SSCI 2020*, pp. 1154–1161, 2020, doi: 10.1109/SSCI47803.2020.9308387.

[94] Y. A. Ahmed, B. Koçer, and B. A. S. Al-Rimy, "Automated Analysis Approach for the Detection of High Survivable Ransomware," *KSII Trans. Internet Inf. Syst.*, vol. 14, no. 5, pp. 2236–2257, 2020, doi: 10.3837/tiis.2020.05.021.

[95] C. Galen and R. Steele, "Performance Maintenance Over Time of Random Forest-based Malware Detection Models," 2020 11th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), 2020, pp. 0536-0541, doi: 10.1109/UEMCON51285.2020.9298068.

[96] S. Sharmeen, Y. A. Ahmed, S. Huda, B. S. Kocer, and M. M. Hassan, "Avoiding Future Digital Extortion through Robust Protection against Ransomware Threats Using Deep Learning Based Adaptive Approaches," *IEEE Access*, vol. 8, pp. 24522–24534, 2020, doi: 10.1109/ACCESS.2020.2970466.

[97] G. O. Ganfure, C. F. Wu, Y. H. Chang, and W. K. Shih, “DeepGuard: Deep Generative User-behavior Analytics for Ransomware Detection,” *Proc. - 2020 IEEE Int. Conf. Intell. Secur. Informatics, ISI 2020*, 2020, doi: 10.1109/ISI49825.2020.9280508.

[98] F. Ullah *et al.*, “Modified Decision Tree Technique for Ransomware Detection at Runtime through API Calls,” *Sci. Program.*, vol. 2020, 2020, doi: 10.1155/2020/8845833.

[99] B. Qin, Y. Wang, and C. Ma, “API Call Based Ransomware Dynamic Detection Approach Using TextCNN,” *Proc. - 2020 Int. Conf. Big Data, Artif. Intell. Internet Things Eng. ICBAIE 2020*, pp. 162–166, 2020, doi: 10.1109/ICBAIE49996.2020.00041.

[100] S. Aurangzeb, R. N. Bin Rais, M. Aleem, M. A. Islam, and M. A. Iqbal, “On the classification of Microsoft-Windows ransomware using hardware profile,” *PeerJ Comput. Sci.*, vol. 7, pp. 1–24, 2021, doi: 10.7717/peerj-cs.361.

[101] S. Verma, A. Chug, and A. P. Singh, *Android Ransomware Detection Based on Dynamic Obtained Features*, vol. 978 AISC, no. Scdm. 2020.

[102] M. E. Ahmed, H. Kim, S. Camtepe, and S. Nepal, “Peeler: Profiling Kernel-Level Events to Detect Ransomware,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 12972 LNCS, pp. 240–260, 2021, doi: 10.1007/978-3-030-88418-5_12.

[103] M. A. Ayub, A. Continella, and A. Siraj, “An I/O Request Packet (IRP) Driven Effective Ransomware Detection Scheme using Artificial Neural Network,” *Proc. - 2020 IEEE 21st Int. Conf. Inf. Reuse Integr. Data Sci. IRI 2020*, pp. 319–324, 2020, doi: 10.1109/IRI49571.2020.00053.

[104] B. Jethva, I. Traoré, A. Ghaleb, K. Ganame, and S. Ahmed, “Multilayer ransomware detection using grouped registry key operations, file entropy and file signature monitoring,” *J. Comput. Secur.*, vol. 28, no. 3, pp. 337–373, 2020, doi: 10.3233/JCS-191346.

[105] M. Hirano, R. Hodota, and R. Kobayashi, “RanSAP: An open dataset of ransomware

storage access patterns for training machine learning models," *Forensic Sci. Int. Digit. Investig.*, vol. 40, p. 301314, 2022, doi: 10.1016/j.fsidi.2021.301314.

[106] T. Gutierrez, "Malware Sandbox Deployment , Analysis and Development."

[107] A. Mohanta and A. Saldanha, *Malware Analysis Lab Setup*. 2020.

[108] J. A. H. Silva, F. D. B. Veloz, L. I. B. López, Á. Leonardo, V. Caraguay, and M. Hernández-Álvarez, "Dataset de Ransomware basado en análisis dinámico," *Rev. Ibérica Sist. e Tecnol. Informação*, no. E23, pp. 248–261, 2019.

[109] C. Sandbox, "Cuckoo Sandbox Book," 2011.

[110] S. Satish and Z. Ramzan, "Using sequencing and timng information of behavior events in machine learning to detected malware." p. 1, 2013.

[111] Y. L. Wan, J. C. Chang, R. J. Chen, and S. J. Wang, "Feature-Selection-Based Ransomware Detection with Machine Learning of Data Analysis," *2018 3rd Int. Conf. Comput. Commun. Syst. ICCCS 2018*, pp. 392–396, 2018, doi: 10.1109/CCOMS.2018.8463300.

[112] A. Tseng *et al.*, "Deep Learning for Ransomware Detection," *IEICE Tech. Report; IEICE Tech. Rep.*, vol. 116, no. 282, pp. 87–92, 2016, [Online]. Available: https://yunchunchen.github.io/papers/IEICE-16/ieice-papers.pdf.

[113] RapidMiner, "RapidMiner Documentation," 2022. https://docs.rapidminer.com/latest/studio/operators/modeling/predictive/support_vector_machines/fast_large_margin.html.

[114] Mahbubul Alam, "Linear Regression or Generalized Linear Model?," *May 31*, 2020. https://towardsdatascience.com/linear-regression-or-generalized-linear-model-1636e29803d0.

[115] C. G. Akcora, Y. Li, Y. R. Gel, and M. Kantarcioglu, "BitcoinHeist: Topological Data Analysis for Ransomware Prediction on the Bitcoin Blockchain," pp. 4439–4445, 2020, doi: 10.24963/ijcai.2020/612.

[116] K. Lee, S. Y. Lee, and K. Yim, "Machine Learning Based File Entropy Analysis for Ransomware Detection in Backup Systems," *IEEE Access*, vol. 7, pp. 110205–

110215, 2019, doi: 10.1109/ACCESS.2019.2931136.

[117] S. H. Kok, A. Abdullah, N. Z. Jhanjhi, and M. Supramaniam, “Prevention of crypto-ransomware using a pre-encryption detection algorithm,” *Computers*, vol. 8, no. 4, pp. 1–15, 2019, doi: 10.3390/computers8040079.

[118] S. Poudyal, K. P. Subedi, and D. Dasgupta, “A Framework for Analyzing Ransomware using Machine Learning,” *Proc. 2018 IEEE Symp. Ser. Comput. Intell. SSCI 2018*, pp. 1692–1699, 2019, doi: 10.1109/SSCI.2018.8628743.

[119] ABORISADE, Opeyemi; ANWAR, Mohd. Classification for authorship of tweets by comparing logistic regression and naive bayes classifiers. En 2018 IEEE International Conference on Information Reuse and Integration (IRI). IEEE, 2018. p. 269-276.

[120] B. Tang, S. Kay, and H. He, “Toward Optimal Feature Selection in Naive Bayes for Text Categorization,” *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 9, pp. 2508–2521, 2016, doi: 10.1109/TKDE.2016.2563436.

[121] ROSELINE, S. Abijah, et al. Intelligent vision-based malware detection and classification using deep random forest paradigm. IEEE Access, 2020, vol. 8, p. 206303-206324.

[122] A. Azmoodeh, A. Dehghantanha, M. Conti, and K. K. R. Choo, “Detecting crypto-ransomware in IoT networks based on energy consumption footprint,” *J. Ambient Intell. Humaniz. Comput.*, vol. 9, no. 4, pp. 1141–1152, 2018, doi: 10.1007/s12652-017-0558-5.

[123] MORALES-MOLINA, Carlos Domenick, et al. Methodology for malware classification using a random forest classifier. En 2018 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC). IEEE, 2018. p. 1-6.

[124] B. ALAHMADI, I. MARTINOVIC, "MalClassifier: Malware family classification using network flow sequence behaviour", *En 2018 APWG Symposium on Electronic Crime Research (eCrime)*. IEEE, 2018. p. 1-13.

[125] A. Alqahtani and F. T. Sheldon, “A Survey of Crypto Ransomware Attack Detection Methodologies: An Evolving Outlook,” Sensors, vol. 22, no. 5, p. 1837, Feb. 2022, doi: 10.3390/s22051837.

[126] A. Gaurav,B. Gupta, P. Kumar, “A comprehensive survey on machine learning approaches for malware detection in IoT-based enterprise information system” https://doi.org/10.1080/17517575.2021.2023764

[127] Prachi, Kumar, “An effective ransomware detection approach in a cloud environment using volatile memory features”. J Comput Virol Hack Tech 18, 407–424 (2022). https://doi.org/10.1007/s11416-022-00425-2

[128] A. Vehabovic, N. Ghani, E. Bou-Harb, J. Crichigno and A. Yayimli, "Ransomware Detection and Classification Strategies," 2022 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom), 2022, pp. 316-324, doi: 10.1109/BlackSeaCom54372.2022.9858296.

[129] M. Masum, M. J. Hossain Faruk, H. Shahriar, K. Qian, D. Lo and M. I. Adnan, "Ransomware Classification and Detection With Machine Learning Algorithms," 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC), 2022, pp. 0316-0322, doi: 10.1109/CCWC54503.2022.9720869.

[130] C. Gao, H. Shahriar, D. Lo, Y. Shi and K. Qian, "Improving the Prediction Accuracy with Feature Selection for Ransomware Detection," 2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC), 2022, pp. 424-425, doi: 10.1109/COMPSAC54236.2022.00072.

[131] T. M. Cover and J. A. Thomas. “Elements of Information Theory”. John Wiley & Sons, 2nd edition, 2006.).

ANNEXES

ANNEX A: Tool for feature extraction

ANNEX B: Examples of the use of feature extraction tool

ANNEX C: Objects and features in json files

ANNEX D: Average occurrences of features for ransomware

ANNEX E: Model performances for Neural Networks models for 12 Datasets in Phase Final

ANNEX F: Model performances for Supervised Learning for 12 Datasets in Phase Final

ANNEX G: Model performances for Supervised Learning for Dataset Global in Phase Final

ANNEX H: Model performances for Neural Networks for Dataset Global in Phase Final

ANNEX I: Generation for Dataset Global in Phase Final

ANNEX J: Model performances for Neural Networks for Dataset Extract in Phase Final

ANNEX K: Model performances for Supervised Learning for Dataset Extract in Phase Final

ANNEX L: Artifacts' use of CPU and memory

ANNEX M: Dataset Phase Initial

ANNEX N: Dataset Phase Analysis

ANNEX O: Dataset Global Phase Final

ANNEX P: Dataset Extract Phase Final

Note: Annexes at:

https://drive.google.com/drive/folders/1dLrRr4U8J3gFPkP5u0Is7QmgjgAuaD-X

The Dataset Extract Phase Final is available for public access along with the present article and in the GitHub repository: https://github.com/Juan-Herrera-Silva/Paper-SENSORS