Dynamic Ransomware Classification: Leveraging Sandboxes and Machine Learning

Augusto Parisot, Lucila M.S. Bento, Raphael C.S. Machado

***Abstract*—The surge in ransomware attacks in recent years has elevated this malware to one of the foremost cybersecurity threats. This article presents a dynamic ransomware classification approach, leveraging the malware analysis environment provided by Cuckoo Sandbox and machine learning techniques. We introduce a methodology encompassing steps for malicious code sample collection, environment configuration for sample execution, data collection, and dataset construction for ransomware detection and experimentation. Six machine learning classifiers were employed to identify ransomware families and individual cases, furnishing valuable tools for threat detection. The results underscore the effectiveness of tree-based methods, such as Random Forests and Decision Trees, in delineating between different ransomware families.**

***Index Terms*—Ransomware, Attack Detection, Machine Learning, Sandbox.**

I. Introduction

THE cybercrime economy is burgeoning year by year. In 2023, the average cost of a data breach reached $4.45 million, marking a 2% increase from 2022 and a 15% rise over the past three years [1]. According to the World Economic Forum, the potential cost of cybercrime could soar to a staggering $10.5 trillion annually by 2025 [2]. The motivations and methods employed by malicious actors vary significantly, making the battle against this threat exceptionally challenging.

In this context, ransomware attacks have surged due to their simple yet effective techniques, facilitating quick gains through the extortion of victims. Ransomware seizes and encrypts the victim's data, demanding substantial sums of money for its release or to prevent exposure (known as double extortion [3]). The damages caused by ransomware encompass not only the ransom payment but also the costs associated with the victim's halted operations [4], making it a significant threat and a pertinent subject of interest within the scientific community.

Ransomware employs various attack vectors, including but not limited to phishing campaigns, exploiting known vulnerabilities, weak passwords, and supply chain attacks (such as the Kaseya incident [5], [6], MOVEit [7], and JetBrains [8]). Detecting such attacks demands robust tools capable of preventing or halting an ongoing attack and/or recovering the seized data. Naturally, the scientific community has explored Machine Learning algorithms for detecting various types of malware attacks [9], [10], particularly ransomware, given the characteristics of the attacks and the need for swift and adaptive responses.

One of the most critical steps in producing effective Machine Learning-based detection tools is constructing the dataset used to train the classifiers, which can be used either in isolation or combined with other tools for attack detection. As far as we know, there is no ransomware dataset containing the most recently active threats. Among the candidate information to support detection, Application Programming Interface (API) calls are widely used in the literature [11]–[13]. Monitoring API calls provides comprehensive traces of the actions executed by the software, allowing for the creation of API profiles for malware detection and classification [14].

In this article, we analyze the detection of attacks from five ransomware families targeting Windows operating systems, which have been responsible for a substantial number of recent attacks [5], [15]–[17]. Our proposal is based on using the *Cuckoo Sandbox* tool to execute and investigate the behavior of samples from the considered families. Utilizing the reports generated by the *Cuckoo Sandbox*, we construct a dataset containing the API calls made by the considered families and employ six Machine Learning algorithms for attack detection. Summarizing our results:

- Description of the environment configuration used for executing the samples.
- Implementation and provision of tools for collecting malware samples and generating datasets from the reports produced by the *Cuckoo Sandbox*.
- Creation and provision of the produced dataset to ensure the reproducibility of the presented results and enable further studies on these ransomware families.
- Analysis of the performance of six Machine Learning algorithms in detecting attacks from the five considered families.

By creating a ransomware dataset addressing the latest threats, this research fills a critical gap and lays a solid foundation for future studies on ransomware detection. The comparative analysis of machine learning algorithms contributes to understanding the effectiveness of different approaches in ransomware detection and can help identify which algorithms are best suited to address the specific characteristics of ransomware attacks, thereby guiding the development of more effective and adaptable solutions. Furthermore, making the

A.P. is with the Federal Fluminense University (UFF), Institute of Computing, Rio de Janeiro, RJ, Brazil. email: aparisot@id.uff.br, https://orcid.org/0009-0001-6238-3604

L.M. de Souza Bento is with the State University of Rio de Janeiro, Institute of Mathematics and Statistics (IME UERJ), Rio de Janeiro, RJ, Brazil. email: lucila.bento@ime.uerj.br, https://orcid.org/0000-0002-3688-5838

R.C. Santos Machado is with the Federal Fluminense University (UFF), Institute of Computing, Rio de Janeiro, RJ, Brazil. email: raphael-machado@ic.uff.br, https://orcid.org/0000-0003-3339-9735

Submission: 2024-03-04, First decision: 2024-07-29, Acceptance: 2025-11-25, Publication: 2025-12-19.

Digital Object Identifier: 10.14209/jcis.2025.12

dataset and comparative analysis results available can promote research reproducibility and encourage additional studies on ransomware detection.

The work is organized as follows. Section II explores related works found in the literature. The methodology employed to conduct the research is detailed in Section III. The results of the conducted experiments and the evaluation of the obtained classifications are presented in Section IV. Finally, Section VI delves into future prospects and offers conclusions drawn from this study.

II. Related Works

Over the last five years, ransomware campaigns have proven highly lucrative for their operators [18]. Despite international efforts to investigate and combat these extortion practices, their profitability persists. Consequently, there is a continuous dissolution and reformation of new malicious actor groups, often incorporating components from previous software and implementing updates to enhance malware capabilities. This dynamic results in a proliferation of active variants of the same ransomware. In this context, sandbox environments are widely used in malware analysis for their ability to automate the process. They enable the evaluation of a large number of samples, facilitate the comparison of different variants, and enable the creation of datasets for Machine Learning algorithms, as discussed in the following sections.

Maniriho et al. [19] conducted a comprehensive systematic review of literature about malware detection techniques spanning from 2009 to 2022, providing an overview of the current landscape of Windows malware detection techniques. The literature review highlighted that behavior-based techniques, such as the one adopted in this work, use analysis tools, like sandboxes, to monitor and capture behavioral characteristics of malware and benign EXE files. The paper also identifies Support Vector Machine and Random Forest as the most widely used Machine Learning algorithms in malware detection.

The importance of combining machine learning algorithms with analysis tools was emphasized in the work of Cen et al. [13] specifically in the context of detecting ransomware attacks.

Bhagwat and Pati [9] detail the process of extracting behavioral traces from malware using the *Cuckoo Sandbox* and a parser to transform the reports into a dataset. They focused on data related to file access, processes, and registry changes. However, the paper lacks specific details on the selected parts or their representation in the overall content of the reports. Additionally, there is no information about the parser used, the features selected after applying the proposed reduction methods, and the resulting dataset is not available for reference. Other recent works [20]–[22] have also employed sandboxing to detect ransomware attacks.

Dinh et al. [10] conducted three types of data extraction from analysis reports generated by the *Cuckoo Sandbox*. The first method treated the report file of each sample as a 1-gram. The second considered only the categories of network, signatures, behaviors, and others (unspecified) for feature extraction in the dataset. In the third method, a new dictionary was created, incorporating each new key-value pair from the described sections, except in some cases where the occurrence count of a specific key was included.

Observing that malware executes unique sequences of API calls distinguishing them from other programs, [14] proposed a method where the quantity of calls to a particular API is considered, and these quantities are transformed into a vector for each sample. The study used samples from three categories: malicious, benign, and benign cryptographic programs. While the paper outlines the techniques for extracting the best features, there is a gap regarding the process that transforms the reports from the sandbox into the initial dataset. Additionally, the data is not made available for access.

The work in [11] utilizes sequences of API calls, file accesses, and process trees represented through a vector model. Among the reviewed works, this paper is the only one providing some details about the configuration carried out on the virtual machine used for sample analysis. In the study, 10 folders and 1,000 subfolders were randomly distributed within Windows working folders such as “My Documents” and “Desktop”. No file or program sharing occurred with the authors’ developed/utilized tools.

Chen et al. [23] proposed a malware detection method considering runtime information with the names of called APIs, obtained using a sandbox, and integrating with Cyber Threat Intelligences (CTIs) knowledge. Furthermore, the method extracts a set of indicators to help identify the security sensitivity levels of API calls, using deep neural networks to train the malware detection model. Experimental results indicate similar success rates to approaches that use simpler machine learning algorithms.

In line with the previously mentioned works, this study employs sandboxing and API calls to obtain malware behavior. However, unlike the previous works, the present study provides detailed configuration information about the experimental environment, the process of feature extraction and selection, and makes all the developed data and tools available to the community through GitHub, ensuring the reproducibility of the results and making the resources available for further analysis.

III. Methodology

To address the existing limitations in the literature and bridge the gap, this study focuses on examining and categorizing API calls as either ransomware or benign, leveraging a dataset constructed specifically for this purpose. Relevant attributes were extracted from the dataset to optimize performance. The study adhered to a consistent methodology, encompassing several key steps: identification of the most active ransomware families and sample acquisition, configuration of the environment for running ransomware samples, execution of samples within a sandbox for data acquisition, pre-processing, classification using various machine learning models, and subsequent analysis.

To construct the dataset, we scrutinized threat reports from reputable organizations to identify the five most active ransomware families. Samples from these families were then sourced from public repositories. Subsequently, we meticulously configured the execution environment to replicate

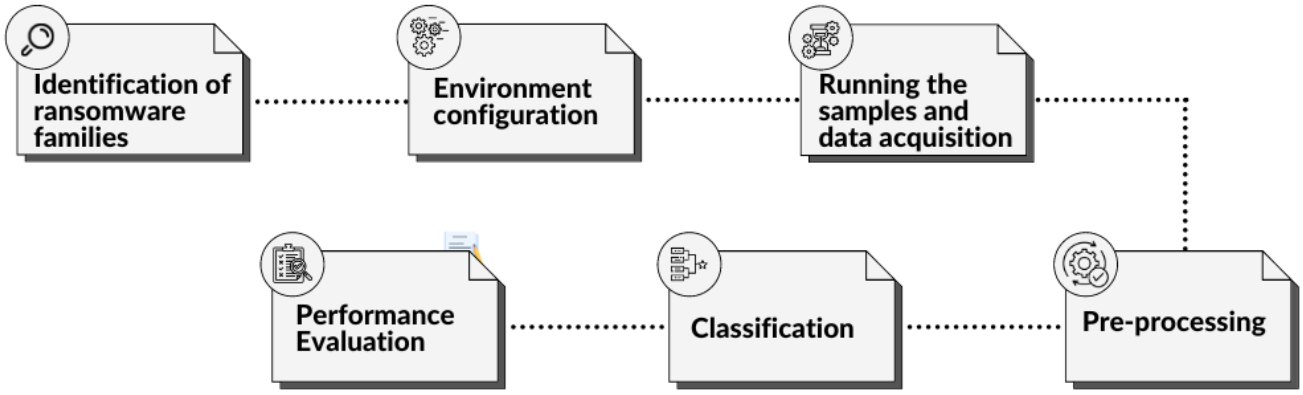

Fig. 1. Methodology pipeline.

real-world computing environments and executed the samples within a sandbox, gathering relevant data throughout the process.

Upon dataset acquisition, we conducted initial visualization and evaluation to identify prevalent features. Subsequent preprocessing steps aimed to enhance data quality and optimize model performance and reliability. The dataset was partitioned into training-test sets in 70:30 and 50:50 ratios to facilitate classification using a variety of machine learning models, including Decision Trees, Random Forests, K-Nearest Neighbors, Naive Bayes, Support Vector Machines, and Multilayer Perceptrons. Finally, the obtained results were analyzed comprehensively.

A visual representation of the methodology pipeline is depicted in Fig. 1, illustrating the sequential flow of the methodological steps adopted in this study.

Detailed descriptions of each stage, including activities performed and tools utilized, are provided in subsequent sections.

A. Ransomware families

The landscape of malware, particularly ransomware, is in constant flux, necessitating a focus on families responsible for significant recent attacks rather than those active in the current month. Drawing from various threat reports [5], [15]–[17], our analysis centers on five prominent ransomware families: REvil, Ryuk, LockBit, Conti, and Clop. These families, which orchestrated numerous attacks in 2022 and 2023, form the cornerstone of our investigation. In this context, a "ransomware family" refers to distinct groups or variants of ransomware characterized by similar attributes and behaviors often associated with specific malicious actors involved in their development and/or distribution.

- **REvil:** Typically infiltrates systems through phishing campaigns, brute-force attacks on Remote Desktop Protocol, or software vulnerabilities [24].
- **Ryuk:** Upon infecting a system, initiates service and process shutdowns to streamline uninterrupted data encryption [25].
- **LockBit:** Esteemed as one of the most lucrative ransomware families, LockBit boasts rapid encryption speeds by encrypting only portions of files [26].
- **Conti:** Employs a variety of tools and techniques, including custom malware and a multi-stage intrusion model [27]. It leverages public search services like *Shodan* and tools such as *Shelter Project*.
- **Clop:** Evolved from *CryptoMix*, initially disseminated via email spear-phishing campaigns. Despite the apprehension of six suspects in Ukraine in 2021, attacks persisted after a brief hiatus [28], [29].

To identify samples from these families, we conducted searches across four repositories: VirusTotal, VirusShare, Malware Bazaar, and Hybrid-Analysis. Due to licensing constraints, we utilized VirusTotal as a catalog to obtain the SHA256 hashes of samples belonging to the considered families. Subsequently, the samples were downloaded from VirusShare, Malware Bazaar, and Hybrid-Analysis using the provided hashes.

To automate the search and download tasks, we developed two Python scripts. The first script retrieves hashes from VirusTotal, while the second script queries the VirusTotal database for additional information and checks for sample availability in other repositories.

Following the acquisition, evaluation, and processing of hash lists, the total sample quantity per ransomware family is detailed in Tab. III-A. Given our focus on analyzing malware targeting the Windows environment and the capabilities of the Cuckoo Sandbox configuration, only EXE and DLL files were selected for analysis.

Family	Sample Quantity
Ryuk	52
Revil	629
LockBit	49
Conti	104
Clop	15

TABLE I
SAMPLE QUANTITY PER RANSOMWARE FAMILY

B. Environment configuration

Malware analysis involves studying malware samples to comprehend their functionalities and behavior. This encompasses two primary approaches: Static Analysis and Dynamic

Analysis. Static analysis involves scrutinizing program code without executing it, which may entail inspecting source code or analyzing compiled binaries' machine code. Dynamic Analysis, on the other hand, involves running malware samples in a controlled virtualized environment to monitor their behavior, observing actions such as registry key alterations, changes to execution modes, network communications, and other system interactions.

In our study, dynamic analysis serves as the primary approach for malware analysis due to its capability to observe malware behavior in a controlled environment, enabling the detection of malicious activities not always evident through static analysis alone.

1) Techniques for Hiding Virtualization: To evade analysis and bypass security controls, malware authors often design their code to terminate execution when they detect analysis environments. Due to the behavior of ransomwares and recommendations from different authors [30]–[32], this study focuses on hiding the virtualization of the considered families. As part of the virtual machine (VM) preparation, the following steps were taken:

- Virtual Machine Preparation: Configured as an end-user machine, the VM simulated user activity by downloading files and installing programs [33], populating the system with various applications and files.
- Paranoid Fish Tool [34]:Executed to detect characteristic virtualization traces, resulting in changes to Windows Registry entries and replacing the default MAC address of the VirtualBox network interface.
- INETSIM: Used to simulate common internet services, convincing the ransomware under analysis that network services are available.

2) Changes to Windows Group Policy: Several changes were made to the Windows group policy to reduce the security barriers of the operating system itself. These changes included:

1) Behavior of the elevation prompt for administrators in Admin Approval Mode (Elevate without prompting);
2) Detect application installations and prompt for elevation (Disabled);
3) Run all administrators in Admin Approval Mode (Disabled);
4) Configure Automatic Updates (Notify for download and notify for install);
5) Protect all network connections (Disabled);
6) Turn off Windows Defender Antivirus (Enabled).

3) Cuckoo Sandbox Analysis: Cuckoo Sandbox, created in 2012 by Guarnieri [35], was utilized for analyses. Once a ransomware sample is submitted, Cuckoo loads a snapshot of the VM (configured as described previously) to return the system to a malware-free state before transferring and executing the file under analysis. While the sample is being executed, Cuckoo collects execution information such as API calls, downloaded files, network traffic, strings contained in the file, packers, memory dumps, and other information. It is worth noting that Cuckoo Sandbox has a built-in REST API that allows operations such as file submission, analysis tracking, machine monitoring, and report downloading, facilitating integration with other systems and data mining from the analysis results (used in building the dataset, described in Section III-C1).

4) Hardware and Software Configuration: The host machine runs Ubuntu 20.04.4 LTS (64-bit) on an Intel Core i7-8550U CPU with 16GB DDR4 RAM. The virtual machine (VM) is equipped with Windows 7 Ultimate SP1 x64 on VirtualBox version 6.1.32, allocated 4 cores with 4GB of RAM. The architecture of the environment configuration is presented in Fig. 2.

C. Running Samples and Data Acquisition

Once the samples collected, as described in Section III-A, are stored locally, we initiate the analysis process using Cuckoo Sandbox and VirtualBox tools. With the environment configurations detailed in Section III-B, we submit the samples for analysis. The analysis is performed in three steps.

1) Setup and Launch:
   a) Launch the main program of Cuckoo Sandbox and the VM.
   b) Start the Cuckoo API and INETSIM services.
2) Analysis Process:
   a) Cuckoo Sandbox injects the monitoring script file into the client, where the sample executes.
   b) The monitoring script captures various information throughout the sample run.
   c) Upon completion, the monitoring script sends the recorded results back to Cuckoo, residing in the VM software via the virtual network.
   d) The analysis component within the host parses and generates a report file.
   e) The VM software utilizes a snapshot to restore the VM to its initial state.
3) Data Extraction and Processing:
   a) We utilize the Cuckoo Sandbox API to retrieve reports for each analyzed ransomware sample.
   b) Reports containing detailed information about the behavior of each sample during execution are saved in JSON format.
   c) The JSON reports are divided into different sections, with the "behavior" section being the most relevant for the scope of this study. It contains the count of API calls made by the processes created by the malware, process trees, used DLLs, created and removed files and registry entries, among other information. All API calls made during 600 seconds of monitoring were considered, following the recommendation by Black et al. [14].
   d) In addition to capturing the JSON reports, our developed scripts perform the necessary filtering to transform the raw report data into features to compose a Pandas dataset.
   e) The NaN-filled data, resulting from the fact that a certain sample did not make a call to that particular API, was replaced with zeros in the produced Pandas dataset.

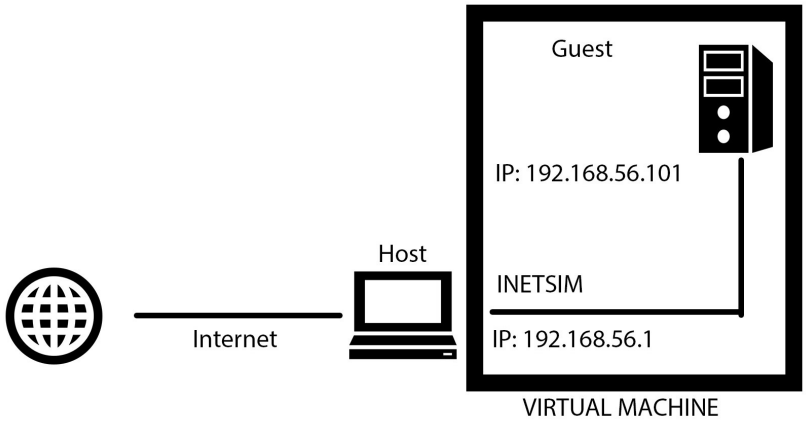

Fig. 2. Test environment setup

By following this process, we gather comprehensive data on the behavior of each ransomware sample under analysis, facilitating the subsequent stages of feature extraction, classification, and evaluation.

1) Dataset Description: Following the steps outlined in Section III-C, we obtained the Pandas dataset comprising the ransomware samples detailed in Section III-A, along with the analysis of 100 samples of benign applications. This benign set encompasses applications like Mozilla Thunderbird, Mozilla Firefox, MS Office 2016, WinRAR, VLC Media Player, Java, and generic Windows utilities such as drivers, network utilities, multimedia tools, and program installers.

In total, the dataset consists of 261 columns representing various APIs called during the analysis, including NtQueryKey, LdrUnloadDll, and LdrGetDllHandle, and 849 rows representing each of the analyzed samples.

D. Pre-Processing Phase

With our datasets prepared, we are ready to apply data optimization techniques and move forward with classification. It's crucial to note that many Machine Learning algorithms require data standardization to perform effectively. Without standardization to a normal distribution (Gaussian with zero mean and unit variance), these algorithms may underperform. In our API calls dataset, we observe significant variability in values, particularly in calls to the Windows Crypto library (such as CryptGenKey, CryptEncrypt, CryptDecrypt, and CryptExportKey), which are heavily utilized by ransomware for file encryption. When a feature's variance vastly outweighs others, it can dominate the objective function, hindering the algorithm's ability to learn other features effectively.

To establish comparison parameters and evaluate potential improvements in classification accuracy, we applied the StandardScaler [36] standardization technique and the Principal Component Analysis (PCA) [37] dimensionality reduction method during the pre-processing phase.

StandardScaler scales the data so that each feature has a mean of zero and a variance of one. This normalization is crucial for algorithms like Support Vector Machines and K-Nearest Neighbors, which perform optimally when features are standardized. Given the varying counts of API calls across samples, StandardScaler helps mitigate these differences, ensuring comparability across the dataset.

PCA, on the other hand, is a dimensionality reduction technique that reduces the number of variables in a dataset while retaining as much information as possible. With our dataset containing numerous API calls, PCA identifies the directions of greatest variance and projects the data into a new, lower-dimensional space, preserving relevant information. This not only aids in reducing model training time but also mitigates the risk of overfitting.

Therefore, employing StandardScaler to standardize API call counts and PCA to reduce dataset dimensionality proves invaluable for data preparation before classification, particularly when dealing with a large number of API calls and aiming to address issues related to dimensionality and disparate scales between features. In this work we used PCA with its parameter n=100, therefore reducing the 261 columns to 100.

E. Classification Phase

Machine Learning algorithms play a crucial role in classifying and clustering different types of malware, as indicated by recent studies [13], [38]–[40]. For this research, we selected classical algorithms widely used in the literature [41]–[43], including Decision Tree (DT), Random Forest (RF), K-Nearest Neighbors (KNN), Naive Bayes (NB), Support Vector Machines (SVM), and Multilayer Perceptron (MLP). All algorithms were implemented using the SciKit-Learn library.

1) Decision Trees: DTs [36] are a supervised learning technique known for their non-parametric approach, which involves constructing a hierarchical structure of if–else conditions based on input data properties. They segment the dataset into subsets using features that offer the most information gain. Each internal node signifies a feature, while each branch denotes a decision based on that feature. The leaf nodes represent class labels. While DTs are susceptible to overfitting, techniques like pruning can regularize them.

2) Random Forest: RF [36] is an ensemble learning method derived from DTs. It constructs multiple decision trees and amalgamates their predictions to enhance accuracy and mitigate overfitting. RF introduces randomness in the tree-building

process by considering random subsets of features and samples for each tree. It exhibits robustness to noise and outliers, typically yielding high accuracy across diverse datasets.

3) K-Nearest Neighbors: The KNN [36] algorithm is renowned as one of the simplest Machine Learning algorithms. It's commonly utilized for classification but can be extended for regression tasks. KNN is a non-parametric algorithm wherein the majority class label determines the class label of a new data point among its nearest 'k' neighbors in the feature space. During training, KNN stores labeled instances, serving as the "'knowledge" for predictions. It computes distances, identifies neighbors, and assigns class labels based on majority voting. In cases of ties, additional rules may dictate label selection.

4) Naive Bayes: NB [36] is a probabilistic classifier grounded in Bayes' theorem, assuming strong independence among features. It posits that the presence of one feature in a class is unrelated to the presence of another. Despite its simplicity and assumptions, NB often excels in text classification and other domains. It requires relatively little training data to estimate necessary parameters accurately.

5) Support Vector Machines: SVMs [36] are effective supervised learning techniques applicable to regression and classification tasks, particularly adept at delineating distinct class boundaries. SVM locates an optimal hyperplane that maximally separates classes in feature space. During training, SVMs discern the significance of each data point in illustrating class boundaries, with support vectors being crucial in this process. Classification decisions are based on the importance of support vectors and distances discovered during training.

6) Multilayer Perceptron: MLP [44] is a type of artificial neural network featuring multiple layers of nodes (neurons), including input, hidden, and output layers. Neurons within layers are densely connected, facilitating information flow. MLPs employ backpropagation and gradient descent to update weights, minimizing error between predicted and actual outputs.

F. Evaluation Phase

The evaluation phase involved assessing the performance of binary and multiclass classifications of the selected ransomware families, using the dataset constructed as detailed in Section III-C1.

In multiclass classification, family A is designated as the positive class, while the remaining families (x) are considered the negative class. Hence, we define TP (*True Positive*) as the samples accurately classified as belonging to family A, TN (*True Negative*) as samples correctly identified as not belonging to family A, FP (*False Positive*) as samples inaccurately classified as belonging to family A, and FN (*False Negative*) as samples belonging to family A but not recognized as such.

The metrics *Precision*, *Recall*, *F1-measure*, and *Accuracy* are used for evaluation. *Precision* assesses the classifier's accuracy in predicting a positive family and is calculated as:

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

Recall measures how many true positives were successfully identified as positives by the model:

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

The *F1-Score*, a harmonic mean between *Precision* and *Recall*, signifies overall model quality:

$$F1score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{3}$$

Accuracy indicates the correct classification of positive and negative instances:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

For multiclass classification, Accuracy represents the correct classifications across all families divided by the total number of samples:

$$Accuracy = \frac{\text{correctly classified samples}}{\text{total number of samples}} \tag{5}$$

In this study, *Accuracy* serves as the primary metric, reflecting correct classifications relative to the total number of classifications. Additionally, the highest *Recall* is considered to prioritize detecting all positives in ransomware detection.

In the context of *Precision* and *Recall* metrics, high Precision signifies a close percentage of true positives (TP) to all instances actually positive (TP + FP), which is achievable when false positives (FP) are minimal or nonexistent. Conversely, high Recall denotes a close percentage of true positives (TP) to all instances classified as positive (TP + FN), feasible when false negatives (FN) are minimal or absent.

IV. Results and Further Discussion

The experiments were conducted using Python version 3.9 on the same device outlined in Section III-B4. For training and testing, the data was split into two divisions: one with a 70:30 ratio and another with a 50:50 ratio. Two experiments were carried out for each test size, employing DT, RF, KNN, NB, SVM, and MLP. In Experiment 1, all classifiers were utilized for binary classification across both test sizes. In Experiment 2, all classifiers were employed for multiclass classification across both test sizes. Throughout the experiments, four evaluation metrics — Precision, Recall, F1-score, and accuracy — were recorded, and the models were evaluated based on these metrics.

A. Binary Classification

This section outlines the evaluation focused on identifying whether a sample is benign or malicious. The experiment was conducted individually for each ransomware family, enabling the identification of the specific family to which a malicious sample belongs. Classifications were executed under two scenarios: one with a 70:30 data split (where 70% of the data is used for training and 30% for testing) and another with a 50:50 data split. Additionally, the dataset was evaluated

without optimization, post application of *StandardScaler* for data standardization and PCA with $n = 100$ for dimensionality reduction.

Tab. II, III, IV, V, VI, and VII illustrate the results of binary classification with a 70:30 data split. The rows categorize ransomware families and benign samples (identified in the tables as Goodware), while the columns group dataset optimization scenarios, displaying the observed results for the considered metrics in each scenario. The tables adopt a heatmap-style color scheme, where the classification results are represented by varying shades of blue, with darker shades indicating better performance.

Examining Tab. II, it becomes evident that precision and recall metrics consistently demonstrate strong performance across various malware families, including Revil, Ryuk, LockBit, and Clop. This indicates the algorithm's ability to accurately classify both benign and malicious samples, effectively minimizing false positives and false negatives. Furthermore, the algorithm achieves an impressive global Accuracy of 96%, underscoring its proficiency in categorizing the majority of samples correctly, regardless of their nature. This highlights the algorithm's robustness and effectiveness in identifying discriminatory patterns between benign and malicious samples. Moreover, Precision, Recall, and F1-Score metrics maintain consistently high levels across different data preprocessing configurations, such as normalization (StandardScaler) and dimensionality reduction (PCA). This consistency suggests that the algorithm's performance remains stable and is not significantly influenced by the preprocessing techniques applied to the data.

In Tab. III, the overall accuracy of the RF algorithm matches that of the DT algorithm (i.e., 96%), indicating that RF performs comparably well in correctly categorizing the majority of samples, regardless of their nature as benign or malicious. Notably, for the Revil and Clop ransomware families, both Precision and Recall values consistently remain high. Similar to the observations with the DT algorithm, RF demonstrates consistent high performance across various data preprocessing configurations, including normalization (StandardScaler) and dimensionality reduction (PCA), as evidenced by sustained high levels of Precision, Recall, and F1-Score.

Looking at Tab. IV, we observe significant variations in Precision, Recall, and F1-Score metrics across different ransomware families. For instance, Clop achieves perfect Precision and Recall scores of 100% whereas LockBit exhibits lower scores (69% Precision). These metrics also vary depending on the data preprocessing method applied. For LockBit, Precision drops notably when no data transformation is applied (69%), but escalates to 100% with the StandardScaler method. The overall accuracy of the KNN model ranges from 85% to 100% for various data preprocessing configurations, suggesting sensitivity to data preparation methods while demonstrating high accuracy under specific conditions. Clop proves particularly challenging to classify accurately, highlighting either a necessity for model refinement or a notable difficulty in detecting this specific ransomware strain.

In Tab. V, one can observe significant variations in Precision, Recall, and F1-Score results across different malware families and data preprocessing methods. For instance, the Precision and Recall values for benign samples exhibit considerable fluctuation depending on the preprocessing technique employed. While Recall achieves 100% with StandardScaler, Precision only reaches 15% with PCA. Conversely, for the Revil ransomware family, Precision and Recall exhibit more consistent performance across all preprocessing methods. Nevertheless, the overall accuracy displays substantial variance across preprocessing techniques, ranging from 47% with PCA to 91% with raw data (unprocessed). These findings suggest a high sensitivity of the NB algorithm to data preparation methods, indicating that certain preprocessing approaches may be more suitable for specific ransomware families than others.

By analyzing Tab. VI, it becomes evident that, similar to NB, SVM is sensitive to the data preparation methods, suggesting that specific preprocessing techniques might be better suited for certain ransomware families than others. Furthermore, there is a notable diversity in the Precision, Recall, and F1-score outcomes across various ransomware families and data preprocessing approaches.

When examining Tab. VII, we can see that Precision, Recall, and F1-Score outcomes vary significantly across various malware families and data preprocessing methods. For instance, in benign samples, Precision and Recall values exhibit notable differences based on the chosen data preprocessing approach. With StandardScaler, Recall achieves 100%, whereas with PCA, Precision drops to 0%. On the other hand, for the Revil ransomware family, results appear more consistent, maintaining consistently high Precision and Recall across all preprocessing methods. Nevertheless, the overall accuracy shows considerable variation among different preprocessing techniques, ranging from 85% with PCA to 93% with raw data (no preprocessing).

It can be observed that, overall, the utilization of StandardScaler and PCA enhances classification performance. In specific instances, there was a notable improvement of 10% in the Accuracy metric with 70:30 data spli, particularly evident in LockBit detection (classified by KNN and SVM) and Ryuk. Additionally, Clop (classified by MLP) attained flawless classification performance when StandardScaler and PCA were applied.

Experiments were conducted to evaluate the performance of algorithms with a 50:50 data split. For clarity, the graphic depicted in Fig. 3 illustrates the average values of each metric across datasets without preprocessing, with StandardScaler applied, and with PCA dimensionality reduction.

Upon reviewing the results illustrated in Fig. 3, it becomes evident that, similar to the 70:30 data split, DT exhibits the most favorable average performance concerning Precision, Recall, F1-Score, and Accuracy compared to other classifiers. RF also demonstrates robust average performance across all metrics, albeit slightly below that of the DT. Both SVM and KNN display comparable average performance, with Precision, Recall, and F1-score ranging between 0.75 to 0.8. Although NB yields the lowest average performance, possibly due to its known difficulty in generalization in some scenarios [45]–[47].

We can notice that in some cases, the algorithms were able to achieve perfect classification, where all ransomware samples are considered malicious and all benign samples are

	Normal			StandardScaler			PCA		
Malware	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score
Goodware	0.76	0.86	0.81	0.76	0.86	0.81	0.76	0.86	0.81
Revil	0.99	0.97	0.98	0.99	0.97	0.98	0.99	0.97	0.98
Accuracy		0.96			0.96			0.96	
Goodware	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Ryuk	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Accuracy		1.00			1.00			1.00	
Goodware	1.00	0.93	0.96	1.00	0.93	0.96	0.98	1.00	0.99
LockBit	0.86	1.00	0.92	0.86	1.00	0.92	1.00	0.95	0.97
Accuracy		0.95			0.95			0.95	
Clop	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Goodware	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Accuracy		1.00			1.00			1.00	
Conti	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95	0.95
Goodware	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
Accuracy		0.93			0.93			0.93	

TABLE II
BINARY CLASSIFICATION WITH 70:30 DATA SPLIT AND DT

	Normal			StandardScaler			PCA		
Malware	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score
Goodware	0.95	0.95	0.95	0.91	0.95	0.93	1.00	0.82	0.91
Revil	1.00	1.00	1.00	1.00	0.99	0.99	0.98	1.00	0.99
Accuracy		0.99			0.99			0.98	
Goodware	1.00	0.93	0.96	1.00	0.93	0.96	1.00	0.96	0.98
Ryuk	0.87	1.00	0.93	0.87	1.00	0.93	0.93	1.00	0.96
Accuracy		0.95			0.95			0.98	
Goodware	1.00	0.96	0.98	0.98	1.00	0.99	1.00	0.93	0.96
LockBit	0.92	1.00	0.96	1.00	0.95	0.97	0.87	1.00	0.93
Accuracy		0.97			0.98			0.95	
Conti	0.95	0.95	0.95	0.95	0.95	0.95	0.84	1.00	0.92
Goodware	0.9	0.9	0.9	0.9	0.9	0.9	1.00	0.67	0.8
Accuracy		0.93			0.93			0.88	
Clop	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Goodware	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Accuracy		1.00			1.00			1.00	

TABLE III
BINARY CLASSIFICATION WITH 70:30 DATA SPLIT AND RF

	Normal			StandardScaler			PCA		
Malware	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score
Goodware	0.76	0.86	0.81	0.87	0.91	0.89	0.91	0.91	0.91
Revil	0.99	0.97	0.98	0.99	0.99	0.99	0.99	0.99	0.99
Accuracy		0.96			0.98			0.98	
Goodware	0.93	0.93	0.93	1.00	0.96	0.98	1.00	0.96	0.98
Ryuk	0.85	0.85	0.85	0.93	1.00	0.96	0.93	1.00	0.98
Accuracy		0.90			0.98			0.98	
Goodware	0.96	0.82	0.88	1.00	0.93	0.96	0.97	0.97	0.97
LockBit	0.69	0.92	0.79	0.86	1.00	0.92	0.95	0.95	0.95
Accuracy		0.85			0.95			0.97	
Conti	0.94	0.89	0.92	0.93	1.00	0.96	0.93	1.00	0.96
Goodware	0.83	0.9	0.86	1.00	0.86	0.92	1.00	0.86	0.92
Accuracy		0.90			0.95			0.95	
Clop	0.00	0.00	0.00	1.00	1.00	1.00	1.00	1.00	1.00
Goodware	0.93	1.00	0.96	1.00	1.00	1.00	1.00	1.00	1.00
Accuracy		0,93			1.00			1.00	

TABLE IV
BINARY CLASSIFICATION WITH 70:30 DATA SPLIT AND KNN

	Normal			StandardScaler			PCA		
Malware	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score
Goodware	0.5	0.27	0.35	0.49	1.00	0.66	0.15	1.00	0.26
Revil	0.93	0.97	0.95	1.00	0.89	0.94	1.00	0.41	0.58
Accuracy		0.91			0.9			0.47	
Goodware	0.88	1.00	0.93	0.93	0.46	0.62	1.00	0.93	0.96
Ryuk	1.00	0.69	0.82	0.44	0.92	0.6	0.87	1.00	0.93
Accuracy		0.9			0.61			0.95	
Goodware	0.96	0.96	0.96	1.00	1.00	1.00	1.00	0.85	0.92
LockBit	0.92	0.92	0.92	1.00	1.00	1.00	0.77	1.00	0.83
Accuracy		0.95			1.00			0.85	
Conti	0.75	1.00	0.85	0.79	1.00	0.88	0.95	0.97	0.96
Goodware	1.00	0.38	0.55	1.00	0.52	0.69	0.95	0.9	0.93
Accuracy		0.78			0.83			0.95	
Clop	0.08	1.00	0.14	0.11	1.00	0.19	0.20	1.00	0.33
Goodware	1.00	0.11	0.20	1.00	0.37	0.54	0.20	0.7	0.83
Accuracy		0.17			0.41			0.72	

TABLE V
BINARY CLASSIFICATION WITH 70:30 DATA SPLIT AND NB

considered benign. This observation is particularly relevant when considering the classification performed by DT for Ryuk and KNN for Clop, where we achieved 100% accuracy for both data splits (70:30 and 50:50).

Malware	Normal			StandardScaler			PCA		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score
Goodware	0.70	0.73	0.71	0.91	0.91	0.91	0.91	0.91	0.91
Revil	0.97	0.97	0.97	0.99	0.99	0.99	0.99	0.99	0.99
Accuracy		0.94			0.98			0.98	
Goodware	1.00	0.89	0.94	1.00	0.96	0.98	1.00	0.96	0.98
Ryuk	0.81	1.00	0.9	0.93	1.00	0.96	0.93	1.00	0.96
Accuracy		0.93			0.98			0.98	
Goodware	0.96	0.86	0.91	0.97	1.00	0.98	0.96	1.00	0.98
LockBit	0.73	0.92	0.81	1.00	0.92	0.96	1.00	0.91	0.96
Accuracy		0.88			0.97			0.97	
Conti	0.9	0.96	0.92	0.96	0.97	0.96	0.96	0.97	0.96
Goodware	0.89	0.81	0.85	0.96	0.9	0.93	0.96	0.9	0.93
Accuracy		0.9			0.95			0.95	
Clop	1.00	1.00	1.00	0.67	1.00	0.8	1.00	1.00	1.00
Goodware	1.00	1.00	1.00	1.00	0.96	0.98	1.00	1.00	1.00
Accuracy		1.00			0.97			1.00	

TABLE VI
BINARY CLASSIFICATION WITH 70:30 DATA SPLIT AND SVM

Malware	Normal			StandardScaler			PCA		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score
Goodware	0.73	0.86	0.79	0.96	1.00	0.98	1.00	0.86	0.93
Revil	0.99	0.97	0.98	1.00	1.00	1.00	0.99	1.00	0.99
Accuracy		0.96			1.00			0.99	
Goodware	0.87	0.96	0.92	1.00	0.96	0.98	1.00	0.93	0.96
Ryuk	0.9	0.69	0.78	0.93	1.00	0.96	0.87	1.00	0.93
Accuracy		0.88			0.98			0.95	
Goodware	0.9	0.93	0.91	0.96	0.93	0.95	0.95	0.91	0.92
LockBit	0.82	0.75	0.78	0.85	0.92	0.88	0.82	0.91	0.86
Accuracy		0.88			0.93			0.90	
Conti	0.91	0.84	0.88	0.93	0.97	0.95	0.86	0.97	0.91
Goodware	0.75	0.86	0.8	0.95	0.86	0.9	0.94	0.71	0.81
Accuracy		0.85			0.93			0.88	
Clop	0.00	0.00	0.00	1.00	1.00	1.00	1.00	1.00	1.00
Goodware	0.93	1.00	0.96	1.00	1.00	1.00	1.00	1.00	1.00
Accuracy		0.93			1.00			1.00	

TABLE VII
BINARY CLASSIFICATION WITH 70:30 DATA SPLIT AND MLP

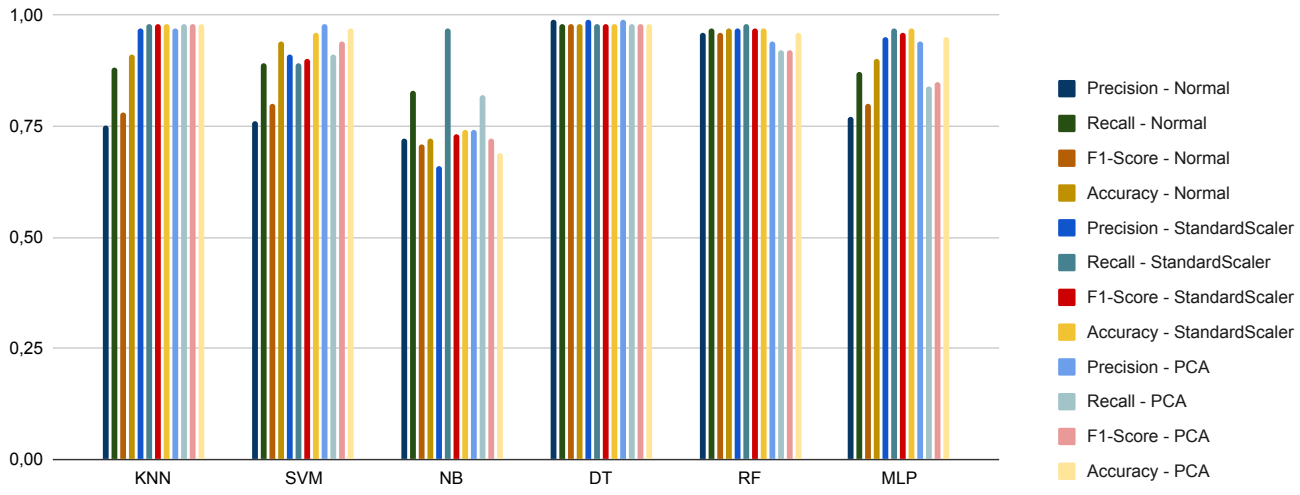

Fig. 3. Binary classification with 50:50 data split

In general, the application of StandardScaler and PCA seems to enhance the performance of evaluation metrics for most classifiers when compared to utilizing the original (Normal) dataset.

B. Muticlass Classification

Tab. VIII, IX, X, XI, XII, and XIII illustrate the performance of the algorithms in differentiating ransomware samples among the selected families. As outlined in Section IV-A, the dataset was divided into 70:30 and 50:50 data splits, considering the dataset without optimization, with the application of StandardScaler, and with the application of PCA with $n = 100$. The organization and formatting of the tables adhere to the description provided in Section IV-A.

Based on Tab. VIII, it is evident that, for certain metrics and ransomware classes, the DT model performed better with the original dataset (Normal) in comparison to data processed using preprocessing techniques such as StandardScaler and PCA. This is highlighted by the cells shaded in darker blue, signifying higher Precision, Recall, and F1-Score values for

Malware	Normal			StandardScaler			PCA		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score
Goodware	1.00	0.90	0.95	0.76	0.91	0.83	0.76	0.91	0.83
Revil	0.98	0.97	0.97	1.00	0.94	0.97	1.00	0.94	0.97
Ryuk	1.00	0.94	0.97	0.92	0.92	0.92	0.92	0.92	0.92
LockBit	0.85	0.94	0.89	0.84	0.91	0.87	0.84	0.91	0.87
Clop	0.50	1.00	0.67	0.67	0.86	0.75	0.67	0.86	0.75
Conti	1.00	0.94	0.97	0.96	0.94	0.95	0.96	0.94	0.95
Accuracy	0.95			0.93			0.93		

TABLE VIII
MULTICLASS CLASSIFICATION WITH 70:30 DATA SPLIT AND DT

the original dataset when contrasted with the corresponding cells in the StandardScaler and PCA columns. For instance, the benign samples (Goodware) exhibit higher Precision, Recall, and F1-Score values in the original dataset than in the processed datasets. A similar trend is noted for the Revil and Conti classes.

In Tab. IX, we observe that RF achieved its best performance in classifying the Ryuk family, falling short of 100% only in the Recall and F1-Score metrics when using the PCA preprocessing technique. Regarding benign samples, the model attained high Precision, Recall, and F1-Score values with the original dataset (Normal), albeit slightly lower values were observed when considering preprocessing techniques like StandardScaler and PCA. Across all preprocessing techniques, the performance metrics for the Revil family consistently remained high, underscoring the model's effectiveness in multiclass classification of this ransomware family. Conversely, the Clop family exhibited significant performance variations across preprocessing techniques, emerging as the family with the poorest performance.

Tab. X suggests that, overall, KNN performance seems to improve with preprocessing techniques, particularly StandardScaler and PCA, where there is a trend of increasing performance metrics compared to using the normal dataset. Regarding the classification of families, benign samples and the Clop family exhibited very poor performance across all metrics, except when StandardScaler and PCA were utilized.

Tab. XI indicates that while the utilization of StandardScaler can enhance multiclass classification with NB, the overall performance of the model remains low.

The SVM achieves an average accuracy of 0.89, as shown in Tab. XII, signifying strong multiclass classification capability. Regarding specific ransomware families, Precision, Recall, and F1-Score metrics remain consistent across all preprocessing techniques for Revil and Conti, while they vary based on preprocessing techniques for Ryuk, LockBit, and Clop.

When analyzing Tab. XIII, it becomes evident that, similar to other models, MLP exhibits varied performance across different ransomware families. While some families achieve high precision and recall rates, others yield lower results. The utilization of preprocessing techniques like StandardScaler and PCA notably enhances MLP's performance compared to raw data. Notably, families such as Revil and Conti are distinguished with remarkable precision and recall, underscoring MLP's effectiveness in identifying these ransomware variants. Despite performance fluctuations across families and preprocessing methods, MLP demonstrates consistent results in multiclass classification, underscoring its proficiency in discriminating between distinct ransomware families.

Just as in binary classification, we conducted experiments to evaluate algorithms' performance with a 50:50 data split for multiclass classification. The graph depicted in Fig. 4 illustrates the average metric values across datasets without preprocessing, with StandardScaler applied, and with PCA dimensionality reduction. The findings indicate that RF exhibited the strongest performance, followed by KNN and DT. Conversely, MLP and NB demonstrated lower performance, with NB recording the weakest results among the algorithms. Notably, NB displayed considerably lower values of precision, recall, and F1-Score across all preprocessing techniques.

We observed promising results regarding the classifiers' performance across different scenarios. RF demonstrated the highest discriminative ability among ransomware families, achieving an impressive overall accuracy of 97% across both 70:30 and 50:50 data splits. Conversely, NB algorithm performed the poorest in multiclass classification, exhibiting inferior performance across both data splits, mirroring its performance in binary classification.

V. LIMITATIONS OF THIS STUDY

In many studies focused on detecting malware using ML algorithms, defining the parameters and characteristics poses significant challenges. While our study employs classical ML models for detection, the process of collecting samples and assembling datasets is subject to continuous development. Consequently, providing a definitive dataset for training and testing ML algorithms remains problematic. It's conceivable that other researchers may introduce new ransomware samples from the same families or incorporate additional steps or activities into the methodology, potentially enhancing model performance through novel combinations of models.

Furthermore, we acknowledge that our dataset is relatively small and lacks balanced proportions between benign and malicious samples. This imbalance could introduce biases into our classification results. Moreover, expanding the benign sample pool to include applications exhibiting behavior akin to ransomware, such as heavy encryption usage, could provide a more comprehensive dataset for analysis.

VI. CONCLUSION AND FUTURE WORK

The utilization of Cuckoo Sandbox and virtualized environments for malware analysis has proven invaluable in cybersecurity. The comprehensive reports provided by Cuckoo offer deep insights into malware behavior, forming a solid foundation for defense strategies. In this study, we construct a new dataset for ransomware classification, leveraging API call data extracted from Cuckoo reports. Using six machine learning classifiers, we successfully identified ransomware families and individual samples, offering valuable tools for

	Normal			StandardScaler			PCA		
Malware	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score
Goodware	1.00	0.95	0.98	0.83	0.85	0.84	0.97	0.85	0.91
Revil	0.97	1.00	0.98	0.97	0.99	0.98	0.92	0.98	0.95
Ryuk	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.96	0.98
LockBit	0.94	0.94	0.94	0.84	0.91	0.87	0.91	0.91	0.91
Clop	0.33	1.00	0.50	1.00	0.14	0.25	1.00	0.86	0.92
Conti	0.94	0.97	0.95	0.94	0.96	0.95	0.86	0.78	0.82
Accuracy		0.97			0.95			0.93	

TABLE IX
MULTICLASS CLASSIFICATION WITH 70:30 DATA SPLIT AND RF

	Normal			StandardScaler			PCA		
Malware	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score
Goodware	0.59	0.62	0.60	0.89	0.91	0.90	0.86	0.91	0.89
Revil	0.92	0.95	0.94	0.97	0.96	0.96	0.98	0.97	0.97
Ryuk	1.00	0.94	0.97	0.96	0.96	0.96	0.93	0.96	0.94
LockBit	0.93	0.78	0.85	0.95	0.91	0.93	0.91	0.91	0.91
Clop	0.00	0.00	0.00	1.00	0.86	0.92	1.00	0.86	0.92
Conti	0.78	0.94	0.85	0.83	0.88	0.85	0.93	0.86	0.89
Accuracy		0.89			0.93			0.94	

TABLE X
MULTICLASS CLASSIFICATION WITH 70:30 DATA SPLIT AND KNN

	Normal			StandardScaler			PCA		
Malware	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score
Goodware	0.44	0.19	0.27	0.78	0.41	0.54	0.07	0.12	0.09
Revil	0.98	0.82	0.89	1.00	0.94	0.97	0.95	0.45	0.61
Ryuk	0.92	0.71	0.80	0.96	0.85	0.90	0.44	0.58	0.50
LockBit	1.00	0.89	0.94	0.84	0.91	0.87	0.27	0.48	0.34
Clop	0.00	0.00	0.00	0.19	1.00	0.33	0.32	1.00	0.48
Conti	1.00	0.42	0.59	0.98	0.84	0.90	0.33	0.82	0.47
Accuracy		0.72			0.89			0.51	

TABLE XI
MULTICLASS CLASSIFICATION WITH 70:30 DATA SPLIT AND NB

	Normal			StandardScaler			PCA		
Malware	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score
Goodware	0.78	0.86	0.82	0.83	0.88	0.86	0.78	0.85	0.82
Revil	0.97	0.94	0.96	0.97	0.93	0.95	0.99	0.92	0.95
Ryuk	0.92	0.65	0.76	0.96	0.96	0.96	0.93	0.96	0.94
LockBit	0.70	0.78	0.74	0.94	0.65	0.77	0.81	0.57	0.67
Clop	0.11	1.00	0.20	1.00	0.86	0.92	1.00	0.86	0.92
Conti	0.93	0.87	0.90	0.81	0.90	0.85	0.81	0.94	0.87
Accuracy		0.89			0.90			0.89	

TABLE XII
MULTICLASS CLASSIFICATION WITH 70:30 DATA SPLIT AND SVM

	Normal			StandardScaler			PCA		
Malware	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score
Goodware	0.60	0.14	0.23	0.81	0.88	0.85	0.86	0.91	0.89
Revil	0.89	0.93	0.91	0.98	0.95	0.97	0.95	0.97	0.96
Ryuk	0.57	0.76	0.65	0.93	0.96	0.94	1.00	0.92	0.96
LockBit	0.93	0.78	0.85	0.87	0.87	0.87	0.90	0.83	0.86
Clop	0.00	0.00	0.00	0.86	0.86	0.86	1.00	0.86	0.92
Conti	0.84	0.68	0.75	0.85	0.92	0.88	0.80	0.92	0.86
Accuracy		0.81			0.93			0.93	

TABLE XIII
MULTICLASS CLASSIFICATION WITH 70:30 DATA SPLIT AND MLP

threat detection. Our research also considered the limitations of dynamic analysis-based approaches against ransomware with anti-VM capabilities, shaping our analysis environment.

Among the classification techniques, tree-based methods (RF and DT) demonstrated promising outcomes in terms of Precision, Recall, and F1 metrics, making them prime candidates for constructing effective ransomware detection systems.

To advance this research, we can conduct a detailed analysis of classifier parameters, particularly focusing on MLP, to enhance classification accuracy. Furthermore, developing and deploying a ransomware detection and prevention system based on the outlined approach, evaluating the efficacy of different classifiers, and assessing the runtime performance of algorithms in the detection system to ensure efficiency in real-time scenarios are crucial steps forward. Additionally, extending the study to classify other types of malware beyond ransomware, exploring the adaptability of the proposed approach, and expanding the proposed dataset by considering new ransomware samples (potentially from other families) and benign samples would enrich future research efforts.

REFERENCES

[1] IBMSecurity. (2023) Cost of a data breach report 2023. [Online]. Available: https://www.ibm.com/reports/data-breach

[2] W. E. Forum, "Why we need global rules to crack down on cybercrime," 2023. [Online]. Available: https://www.weforum.org/agenda/2023/01/global-rules-crack-down-cybercrime/

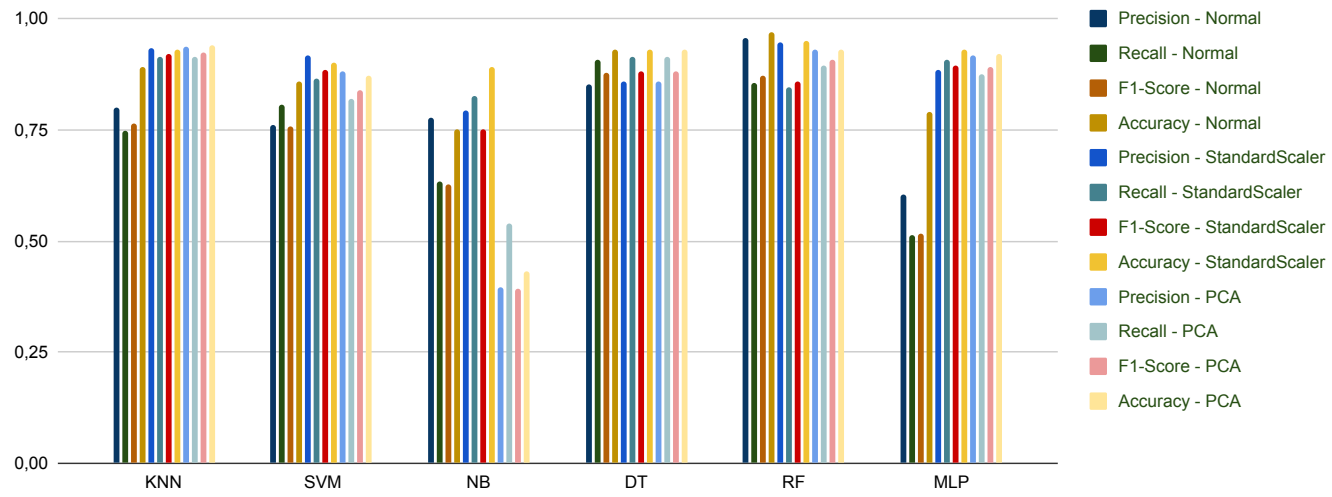

Fig. 4. Multiclass classification with 50:50 data split

[3] Kaspersky, "Ransomware double extortion and beyond: Revil, clop, and conti," 2021. [Online]. Available: https://www.kaspersky.com/resource-center/threats/ransomware-attacks-and-types

[4] Kaspersrky, "Principais ataques de ransomware," 2021. [Online]. Available: https://www.kaspersky.com.br/resource-center/threats/top-ransomware-2020

[5] IBMSecurity, "X-force threat intelligence index 2022," 2022. [Online]. Available: https://www.ibm.com/downloads/cas/ADLMYLAZ

[6] CISA, "Alert (aa21-265a) - conti ransomware," 2022. [Online]. Available: https://www.cisa.gov/uscert/ncas/alerts/aa21-265a

[7] BBC, "Moveit hack: Bbc, ba and boots among cyber attack victims," 2023. [Online]. Available: https://www.bbc.com/news/technology-65814104

[8] C. U. C. . I. S. Agency, "Russian foreign intelligence service (svr) exploiting jetbrains teamcity cve globally," 2023. [Online]. Available: https://www.cisa.gov/news-events/cybersecurity-advisories/aa23-347a

[9] L. B. Bhagwat and B. M. Patil, "Detection of ransomware on windows system using machine learning technique: Experimental results," in *Advanced Computing*, D. Garg, K. Wong, J. Sarangapani, and S. K. Gupta, Eds. Singapore: Springer Singapore, 2021, pp. 417–423, "DOI: 10.1007/978-981-16-0401-0_32".

[10] P. V. Dinh, N. Shone, P. H. Dung, Q. Shi, N. V. Hung, and T. Nguyen Ngoc, "Behaviour-aware malware classification: Dynamic feature selection," in *2019 11th International Conference on Knowledge and Systems Engineering (KSE)*, 2019, pp. 1–5, doi: 10.1109/KSE.2019.8919491.

[11] Y. Takeuchi, K. Sakai, and S. Fukumoto, "Detecting ransomware using support vector machines," in *Workshop Proceedings of the 47th International Conference on Parallel Processing*, ser. ICPP Workshops '18. New York, NY, USA: Association for Computing Machinery, 2018, doi: 10.1145/3229710.3229726.

[12] Prachi., N. Dabas, and P. Sharma, "Malanalyser: An effective and efficient windows malware detection method based on api call sequences," *Expert Systems with Applications*, vol. 230, p. 120756, 2023, doi: 10.1016/j.eswa.2023.120756.

[13] M. Cen, F. Jiang, X. Qin, Q. Jiang, and R. Doss, "Ransomware early detection: A survey," *Computer Networks*, vol. 239, p. 110138, 2024, doi: 10.1016/j.comnet.2023.110138.

[14] P. Black, A. Sohail, I. Gondal, J. Kamruzzaman, P. Vamplew, and P. Watters, "Api based discrimination of ransomware and benign cryptographic programs," in *International Conference on Neural Information Processing*. Springer, 2020, pp. 177–188, doi: 10.1007/978-3-030-63833-7_15.

[15] IBMSecurity, "X-force threat intelligence index 2023," 2023. [Online]. Available: https://www.ibm.com/br-pt/reports/threat-intelligence

[16] M. Horowitz, "Check point 2023 security report," 2023. [Online]. Available: https://go.checkpoint.com/2023-cyber-security-report/chapter-01.php

[17] ThreatDown, "Threatdown state of ransomware a playbook to counter the latest ransomware gangs and attacks," 2024. [Online]. Available: https://www.threatdown.com/wp-content/uploads/2024/09/2024-TD-State-of-Ransomware-US-update-09182024.pdf

[18] K. Team, "Ataques de ransomware direcionados crescem 700%," 2021. [Online]. Available: https://www.kaspersky.com.br/blog/ataques-ransomware-direcionados-crescem-700/17470/

[19] P. Maniriho, A. N. Mahmood, and M. J. M. Chowdhury, "A systematic literature review on windows malware detection: Techniques, research issues, and future directions," *Journal of Systems and Software*, vol. 209, p. 111921, 2024, doi: 10.1016/j.jss.2023.111921.

[20] K. Hammadeh and M. Kavitha, "Unraveling ransomware: Detecting threats with advanced machine learning algorithms," *International Journal of Advanced Computer Science and Applications*, vol. 14, no. 9, p. 484 – 491, 2023, cited by: 0; All Open Access, Gold Open Access.

[21] A. Kamal, M. Derbali, S. Jan, J. I. Bangash, F. Q. Khan, H. Jerbi, R. Abbassi, and G. Ahmad, "A user-friendly model for ransomware analysis using sandboxing," *Computers, Materials and Continua*, vol. 67, no. 3, p. 3833 – 3846, 2021, doi: 10.32604/cmc.2021.015941.

[22] N. W. Pérez-Díaz, J. O. Chinchay-Maldonado, H. I. Mejía-Cabrera, D. E. Bances-Saavedra, and J. A. Bravo-Ruiz, "Ransomware identification through sandbox environment," *Lecture Notes in Networks and Systems*, vol. 560 LNNS, p. 326 – 335, 2023, doi: 10.1007/978-3-031-18458-1_23.

[23] T. Chen, H. Zeng, M. Lv, and T. Zhu, "Ctimd: Cyber threat intelligence enhanced malware detection using api call sequences with parameters," *Computers & Security*, vol. 136, p. 103518, 2024, doi: 10.1016/j.cose.2023.103518.

[24] REUTERS, "Meatpacker jbs says it paid equivalent of 11 mln dollars in ransomware attack," 2021. [Online]. Available: https://www.reuters.com/technology/jbs-paid-11-mln-response-ransomware-attack-2021-06-09/

[25] TrendMicro, "O que é o ransomware ryuk?" https://www.trendmicro.com/pt_br/what-is/ransomware/ryuk-ransomware.html, 2021.

[26] B. Tulas, "Ten notorious ransomware strains put to the encryption speed test," 2022. [Online]. Available: https://www.bleepingcomputer.com/news/security/ten-notorious-ransomware-strains-put-to-the-encryption-speed-test/

[27] ESENTIRE, "Conti ransomware gang claims 50+ new victims including oil terminal operator sea-invest disrupting operations at 24 seaports across europe and africa," 2022.

[28] IPDForum, "South korean police help take down cyber threat in ukraine," 2021. [Online]. Available: https://ipdefenseforum.com/2021/08/south-korean-police-help-take-down-cyber-threat-in-ukraine/

[29] CybeReason, "Cl0p ransomware gang tries to topple the house of cards," 2021. [Online]. Available: https://www.cybereason.com/blog/ceo-series/cl0p-ransomware-gang-tries-to-topple-the-house-of-cards

[30] A. Mills and P. Legg, "Investigating anti-evasion malware triggers using automated sandbox reconfiguration techniques," *Journal of Cybersecurity and Privacy*, vol. 1, no. 1, pp. 19–39, 2020, doi: 10.3390/jcp1010003.

[31] D. E. Garcia and N. DeCastro-Garcia, "Optimal feature configuration for dynamic malware detection," *Computers & Security*, vol. 105, p. 102250, 2021, doi: 10.1016/j.cose.2021.102250.
[32] A. Trafimchuk, A. Bukhteyev, and R. Ladutska, "Checkpoint research evasion techniques," 2022. [Online]. Available: https://evasions.checkpoint.com/
[33] A. Mohanta and A. Saldanha, *Malware Analysis and Detection Engineering: A Comprehensive Approach to Detect and Analyze Modern Malware*. Springer, 2020, doi: 10.1007/978-1-4842-6193-4.
[34] A. Ortega, "Pafish," 2021. [Online]. Available: https://github.com/a0rtega/pafish
[35] C. Guarnieri, A. Tanasi, J. Bremer, and M. Schloesser, "Cuckoo malware analysis," *Packt Publishing*, vol. 16, p. 2018, 2013.
[36] D. Freeman and C. Chio, *Machine Learning and Security: Protecting Systems with Data and Algorithms*. O'Reilly Media, 2018.
[37] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and Intelligent Laboratory Systems*, vol. 2, no. 1, pp. 37–52, 1987, doi: /10.1016/0169-7439(87)80084-9.
[38] J. Singh and J. Singh, "A survey on machine learning-based malware detection in executable files," *Journal of Systems Architecture*, vol. 112, p. 101861, 2021, doi: 10.1016/j.sysarc.2020.101861.
[39] K. Chang, N. Zhao, and L. Kou, "A survey on malware detection based on api calls," in *2022 9th International Conference on Dependable Systems and Their Applications (DSA)*, 2022, pp. 464–471, doi: 10.1109/DSA56465.2022.00067.
[40] K. Begovic, A. Al-Ali, and Q. Malluhi, "Cryptographic ransomware encryption detection: Survey," *Computers & Security*, vol. 132, p. 103349, 2023, doi: 10.1016/j.cose.2023.103349.
[41] K. Faceli, A. C. Lorena, J. Gama, and A. C. P. d. L. F. d. Carvalho, *Inteligência artificial: uma abordagem de aprendizado de máquina*. LTC, 2021.
[42] P. Maniriho, A. N. Mahmood, and M. J. M. Chowdhury, "A systematic literature review on windows malware detection: Techniques, research issues, and future directions," *Journal of Systems and Software*, vol. 209, p. 111921, 2024, doi: 10.1016/j.jss.2023.111921.
[43] M. Benmalek, "Ransomware on cyber-physical systems: Taxonomies, case studies, security gaps, and open challenges," *Internet of Things and Cyber-Physical Systems*, vol. 4, pp. 186–202, 2024, doi: 10.1016/j.iotcps.2023.12.001.
[44] R. Vang-Mata, *Multilayer Perceptrons: Theory and Applications*, ser. Computer Science, Technology and Applications Series. Nova Science Publishers, 2020.
[45] L. Chen, C.-Y. Yang, A. Paul, and R. Sahita, "Towards resilient machine learning for ransomware detection," *arXiv preprint arXiv:1812.09400*, 2018, doi: 10.48550/arXiv.1812.09400.
[46] H. Zhao, M. Li, T. Wu, and F. Yang, "Evaluation of supervised machine learning techniques for dynamic malware detection," *International Journal of Computational Intelligence Systems*, vol. 11, no. 1, pp. 1153–1169, 2018, doi: 10.2991/ijcis.11.1.87.
[47] H. Harahsheh, M. Shraideh, and S. Sharaeh, "Performance of malware detection classifier using genetic programming in feature selection," *Informatica*, vol. 45, no. 4, 2021, doi: 10.31449/inf.v45i4.3819.