Understanding Ransomware Actions Through Behavioral Feature Analysis

Caio C. Moreira, Claudomiro de S. de Sales, Jr., and Davi C. Moreira

***Abstract*—Crypto ransomware attacks have substantially increased in recent years, and owing to their highly profitable nature, this growth will evidently escalate in the future. To better understand this malware and help developers of ransomware detection systems build more robust and reliable solutions, this study investigates ransomware actions during the destruction phase through behavioral feature analysis. We used a dataset with 1524 samples and 30 967 features representing the actions conducted using 582 types of ransomware and 942 good applications (goodware). Six representative and widely used classification algorithms were applied as auxiliary tools to investigate the behavior of these attacks: Naive Bayes (NB), K-Nearest Neighbors (KNN), Logistic Regression (LR), Random Forest (RF), Stochastic Gradient Descent (SGD), and Support Vector Machine (SVM). We achieved an accuracy of 98.48%, balanced accuracy of 98.35%, precision of 98.17%, recall of 97.82%, F-measure of 97.98%, and ROC AUC of 99.87% by using RF for 462 features of the resultant dataset. We propose a new criterion to determine the feature group relevance and a method to distinguish the features that are most related to ransomware and goodware. Our main conclusions are as follows: Application Programming Interface (API) calls are the most relevant feature group, achieving alone a balanced accuracy of 96.49%; native encryption Windows APIs are not crucial for ransomware classification; and the most significant features of ransomware tend to involve handling the thread/process, physical memory operation, and communication, whereas goodware features are more likely to indicate virtual memory, files, directories, and resource operations.**

***Index Terms*—Ransomware detection, feature analysis, behavioral analysis, machine learning, cybersecurity.**

I. Introduction

DURING the COVID-19 pandemic, because working from home has required workers to connect remotely to corporate networks, attacks have significantly increased against several companies and government institutions, making this type of threat one of the 10 worst in the cybersecurity field [1], [2]. The increase in the number of such attacks has led users, organizations, and governments to protect and create backups of critical data. However, because of their highly profitable nature, ransomware attacks are constantly evolving to bypass current protection mechanisms and improve the encryption process [3]. It is possible to infer that the increase in this type of malware will remain a reality in the cyberworld in the coming years.

Ransomware can be divided into two groups: Locky and Crypto. These two groups have different characteristics in terms of malicious actions. Whereas Locky ransomware typically blocks the victim's system by displaying a login page to hinder the user from accessing files, it does not encrypt the system or user data. By contrast, Crypto ransomware encrypts the victim's files and demands a ransom payment to decrypt them. This second type constitutes the majority of the cyberworld and is also the most dangerous [4]. These attacks are composed of four phases, which can be summarized as follows: 1) an infection, in which the ransomware is delivered to the victim's system; 2) communication with Command-and-Control (C2C) servers, in which the ransomware connects to the C2C server to exchange encryption keys and system information with the attacker; 3) destruction, which encrypts files or locking systems to prevent the victim's access to their files or system; and 4) extortion, in which a ransom note informs the victim of the infection and specifies the mode of payment [5].

In general, protection software solutions are capable of defending systems against previously known ransomware. These methods are comparisons based on signatures that already exist in a database. However, in attacks hitherto unknown (zero-day), current solutions present difficulties in the accuracy of detection and return high rates of false positives and false negatives [6], [7]. The majority of these approaches use Machine Learning (ML) models that are built using a set of analysis features. ML-based ransomware detection systems use structural features, behavioral features, or both. Structural features are obtained through static analysis of ransomware binaries, in which the code of a suspicious program is inspected to recognize malicious patterns in the computer system. The major advantage of this approach is its ability to identify harmful instructions before the program is run; however, this approach is more susceptible to deception through obfuscation, polymorphism, encryption, and anti-disassembler techniques. By contrast, behavioral features are obtained through dynamic analysis, in which a suspicious program runs in an isolated environment and its behavior is monitored during its execution, seeking malicious action patterns in the Operating System (OS). The advantage of this type of analysis is that it makes evasion techniques difficult to implement. Moreover, it allows us to analyze and understand how these dangerous programs operate in an OS. Nevertheless, this approach is typically slower and more dangerous [4], [5], [8].

Protection solutions for infection and communication phases are more suitable for Intrusion Detection System (IDS) based on network traffic analysis [9], [10], whereas detection systems based on the identification of a splash screen or ransom note are suitable for the extortion phase [11].

Many studies have analyzed the ransomware destruction phase in an attempt to distinguish it from benign applications or the family to which it belongs. These studies used several

The authors are with the Electrical Engineering Graduate Program, Federal University of Pará (UFPA), Belém, Pará, Brazil (e-mail: caiomoreira@ufpa.br; cssj@ufpa.br; davi.moreira@eletronorte.gov.br).

Digital Object Identifier: 10.14209/jcis.2022.7

techniques to achieve the best performance. However, most of these studies looked at this malware behavior superficially without trying to profoundly understand the reason for the results, but instead looked for only the best metric performance.

Therefore, this paper presents a behavioral feature analysis to improve the understanding of crypto ransomware behavior during the destruction phase in Windows OS and to support developers of anti-malware systems to build more solid and reliable solutions. To achieve these goals, we formulated and answered four questions: 1) Should all groups of features be equally considered, 2) what changes from ransomware to goodware, 3) what are the most relevant APIs, and 4) where are the encryption APIs?. We used a dataset containing 30 967 behavioral actions of 582 ransomware and 942 good applications (goodware) [12]. We first observed that APIs can be grouped based on Microsoft naming conventions, and then eliminated features with low variance and used six traditional classifiers as auxiliary tools to investigate the behavior of these attacks. In addition, this study proposes a new criterion to determine the relevance of each feature group and distinguish the features most associated with ransomware and goodware, which allows us to identify the behavior patterns of both classes.

The remainder of this paper is organized as follows. Section II presents related studies and their main results, advantages, and limitations. Section III describes the dataset, data preprocessing, feature selection, and evaluation metrics. In Section IV, we present our results and answer the questions formulated by investigating the main factors associated with ransomware behavior. Finally, Section V provides the conclusions obtained throughout the study and suggests future research directions.

II. Related Works

Alruhaily et al. [13], [14] analyzed the general behavior of malware to better understand the causes of misclassification. They formulated questions about the behavior of malware and answered them using feature selection techniques and classification algorithms. These types of studies are essential for improving solutions for malware detection systems. Nevertheless, Alruhaily et al. [13], [14] did not focus on a specific type of malware, which could lead to less accurate investigations.

In recent years, several studies have investigated the ransomware issues. Most of these are related to the detection and prevention of this type of attack. Recent studies have used only API calls in dynamic analysis to detect malicious ransomware behavior [15]–[19]. The studies in [20], [21] suggest that API features alone are adequate for detecting ransomwares. Kok et al. [22] proposed a pre-encryption detection algorithm (PEDA) that can detect Crypto ransomware in the pre-encryption stage when no encryption has been conducted using API features. They identified three APIs that appeared more in ransomware and 11 that were more related to goodware. A limitation of [22] is that feature groups other than the API were not analyzed. Furthermore, the proposed PEDA relies on the use of native encryption Windows APIs, as its initial analysis only selects samples that contain APIs with the word "crypt"; thus, it is unable to detect ransomware that uses its own native encryption code.

Sgandurra et al. [12] proposed EldeRan, an ML approach to classifying ransomware based on behavior using the Regularized Logistic Regression (RLR) algorithm. They built and shared a dataset representing several Windows events such as API calls, file and directory operations, registry key modifications, and context (embedded strings). In addition, EldeRan used Mutual Information (MI) as feature selection method and achieved a detection rate of 96.34% using the top-400 features. The results were considered competitive with the top-5 anti-virus vendors. The authors also suggested that the most substantial group of features for classifying ransomware was REG, that is, registry key operation events, followed by API, STR, FILES_EXT (EXT), FILES, DIR, and DROP. The criterion used for evaluating the groups was the percentage of features remaining after feature selection; however, this may not be a good criterion for datasets with imbalanced feature groups.

In addition, using the dataset provided by [12], Abbasi et al. [23] proposed a dynamic analysis with a two-stage feature-selection method that considers the varying importance of each feature group in the dataset. To select an optimal number of features from each group and achieve better classification performance, the model applied MI for Stage I and Particle Swarm Optimization (PSO) for Stage II. They achieved an accuracy of 97.34% using the RLR classifier with 822 ± 59 features. They used the same criterion as in [12] to evaluate the groups and concluded that the API group was the most relevant, followed by REG, STR, EXT, DROP, FILES, and DIR.

Similarly, Khan et al. [24] used the dataset of [12] to evaluate their proposed DNAact-Ran, which is a digital DNA sequencing engine for detecting ransomware using ML. DNAact-Ran combines Multi-Objective Gray Wolf Optimization (MOGWO) and Binary Cuckoo Search (BCS) algorithms to select 26 features and then generates a vector that symbolizes a digital DNA sequence. This vector generates another vector based on the k-mer frequency, which is submitted to a linear regression model based on active learning training data. DNAact-Ran achieved an 87.9% detection accuracy, which is higher than the other algorithms compared; however, converting the features into the DNA sequence makes the features lose their meaning, making an in-depth study of them impossible. Hence, Khan et al. [24] could not analyze the relevance of the groups.

Table I summarizes the advantages and limitations of each related work and compares them with this work. Alruhaily et al. [13], [14] can improve solutions for malware detection systems, but they did not focus on a specific type of malware. Kok et al. [22] identified APIs related to ransomware and goodware. Their solution, however, depended on the use of native encryption Windows APIs, and they only analyzed the API feature group. Sgandurra et al. [12] and Abbasi et al. [23] achieved good performance results and evaluated different feature groups, but they used an inaccurate criterion for evaluating the feature groups. Although Khan et al. [24]

TABLE I
ADVANTAGES AND LIMITATIONS OF THE RELATED WORKS.

Works	Advantages	Limitations
Alruhaily et al. [13], [14]	Can improve solutions for malware detection systems.	Does not focus on a specific type of malware.
Kok et al. [22]	Identifies API features related to ransomware and goodware.	Does not analyze feature groups other than API. Relies on using native encryption Windows APIs.
Sgandurra et al. [12] Abbasi et al. [23]	Evaluates different feature groups. Achieves good performance.	Inaccurate criterion for evaluating the feature groups.
Khan et al. [24]	Achieves fair performance.	Features lose their meaning. Does not evaluate feature groups.
This work	Can improve solutions for malware detection systems. Focus on a specific type of malware. Features retain their meaning. Evaluates different feature groups. Accurate criterion for evaluating the feature groups. Identifies API and other features related to ransomware and goodware. Does not rely on using native encryption Windows APIs. Achieves good performance.	

achieved fair performance results, feature processing caused the features to lose their meaning, rendering them unable to analyze the relevance of feature groups. Our work fills the gaps in related studies while maintaining their advantages. This study can improve the development of detection solutions for a specific type of malware by analyzing behavioral features that maintain their meaning, evaluating different feature groups with a more accurate criterion, and identifying not only APIs but also other features related to ransomware and goodware. Furthermore, our study is not limited to samples that use native encryption Windows APIs. Finally, our work achieved the best performance results compared to related works.

III. MATERIALS AND METHODS

This section describes the methodology, available dataset, data preprocessing, feature selection, tools, evaluation metrics, and experimental environment used in this study. Fig. 1 illustrates the high-level workflow of the proposed experiment. Our work aims to answer the four questions initially formulated by analyzing the available data (step 1), followed by preprocessing of data and feature selection (step 2). We use six representative classifiers to validate our model (step 3). We then split the resultant dataset into seven feature groups (step 4) to analyze each group's relevance according to our proposed criterion (step 5), and then to answer the first question (step 6). Subsequently, all groups of the resultant dataset are submitted to our proposed method to distinguish the features most associated with each class (step 7) using the highest scores found to answer the second question (step 8). Similarly, the API group is individually submitted to the method (step 9) to answer the third question (step 10). Finally, we search for native encryption Windows APIs in both the original and the resultant datasets (step 11), compare the occurrences in each other (step 12), and then submit the remaining features of the resultant dataset to our method (step 13) to answer the fourth question (step 14).

A. Data Description

The dataset used in this study contains 1524 samples: 582 ransomware and 942 goodware [12]. Each sample has 30 967 binary features, which were obtained by recording the events and operations executed by the samples through an analysis using the Cuckoo Sandbox[1]. Seven different groups of features were categorized based on the behavior analysis of each program: API, DROP, REG, FILES, EXT, DIR, and STR. The API group indicates the API invocations of the executable file, DROP is an extension of the dropped files, REG is a registry key operation, FILES indicates file operations, EXT indicates the extensions of the files involved in file operations, DIR represents the directory operations, and STR is the embedded strings in the binary. Each sample was submitted to the analysis tool and all operations were cataloged. The dataset features are Boolean; consequently, each operation or event present in a given sample is represented by 1; otherwise, it is represented by 0.

Fig. 2 shows a 30-bin histogram of the probability distribution (on a logarithmic scale) of the sum of events for each dataset feature. The first bin indicates that the vast majority of features (98.46%) have between 0 and 50 events out of 1524 possibilities. Therefore, this dataset contains a massive concentration of features with little information.

This dataset is intended to represent the most popular versions of ransomware variants and typical applications designed for Windows OS. The ransomware samples belonging to different families are identified by their codes and were collected until 2016. Ransomware has evolved primarily in its dissemination techniques, that is, in the infection phase. There has also been an evolution of the communication and extortion phases; however, the destruction phase, which we propose to investigate, exhibits similar general behaviors [5]. Cybercrime clings to old techniques as long as they work, and thus, the data are still relevant. In [12], the authors provide a complete description of this dataset, which is one of the few datasets available to the public [25]. In [25], because none of the authors of the other papers in the survey made their datasets available to the public, it was noted that it is common practice to not share datasets in this specific area.

[1]https://www.cuckoosandbox.org/

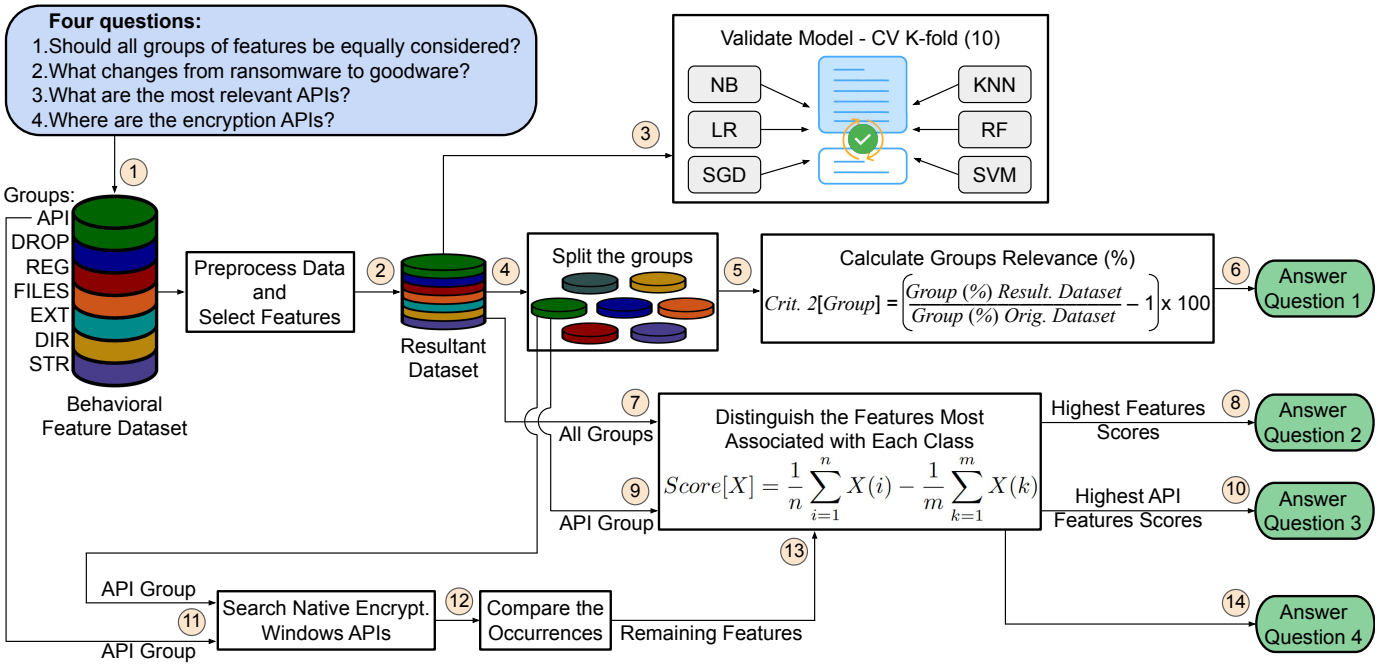


Fig. 1. High-level workflow of the proposed experiment.

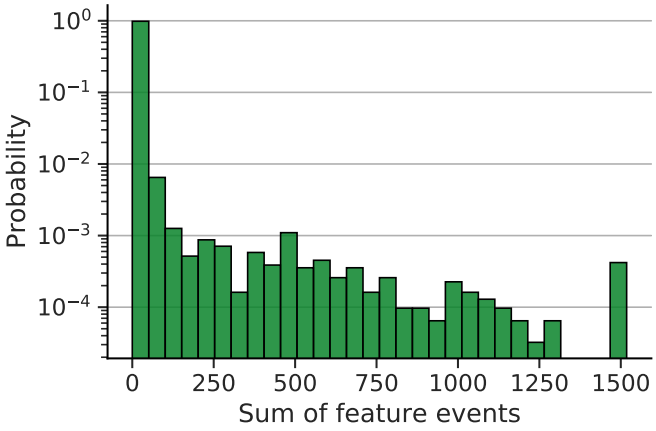


Fig. 2. Histogram of the probability distribution (on a logarithmic scale) of the sum of feature events in the original dataset.

B. Data Preprocessing and Feature Selection

We noted that some API features have similar names, differing only by certain letters at the beginning, end, or both ends of their names. According to [26] and Microsoft's documentation[2,3], an 'A' at the end indicates the API version accepting ANSI coded input strings, and 'W' indicates a Unicode variant. The addition of 'Ex' at the end indicates a new version of that function with different call pattern. The addition of 'Nt' and 'Zw' at the beginning indicates a difference in how they handle the parameter values that the caller passes to the routine. However, all of these APIs exhibit the same behavior.

To improve the understanding regarding how ransomware works, we merged the different API versions to add information gain to these functions, which, if considered separately, can be more easily removed during the feature selection process. We found 44 matches; therefore, the number of features (in the API group) decreased from 232 to 188, and the total number of features decreased to 30 923. Table XI in Appendix B lists all merged APIs. We also verified equal samples belonging to distinct classes and several nonsignificant features.

Algorithm 1 summarizes the data preprocessing and feature selection used in this dataset. After merging different versions of the same APIs (M), we removed all occurrences of equal samples belonging to both ransomware/goodware classes (P1). We then used the scikit-learn Variance Threshold (VT) feature selection method to eliminate non-essential features (F1). Finally, from F1, we removed all occurrences of equal samples belonging to both ransomware/goodware classes.

Algorithm 1 Data preprocessing and feature selection applied to the dataset.

1: Read the Dataset D;
2: $M \leftarrow$ Merge different versions of the same API call.
3: $P1 \leftarrow$ Remove equal records present in different classes in M.
4: $F1 \leftarrow$ Apply VT 97% in $P1$.
5: $P2 \leftarrow$ Remove equal records present in different classes in $F1$.
6: Create new Dataset D' using the resultant samples and selected features;

VT is an unsupervised filter-type feature selection method that removes all features whose variance does not meet a certain threshold. It is based on the Bernoulli trial, which is a probabilistic experiment that can have success ($x = 1$) or

[2]https://docs.microsoft.com/en-us/windows/win32/intl/unicode-in-the-windows-api

[3]https://docs.microsoft.com/en-us/windows-hardware/drivers/kernel/using-nt-and-zw-versions-of-the-native-system-services-routines

failure ($x = 0$). The variance is given by Eq. 1, where p is the probability of success, also called the Bernoulli probability parameter, that defines the VT percentage [27].

$$Var[X] = p(1-p) \tag{1}$$

As shown in Fig. 2, most features have 0–50 events out of 1524 possibilities. Thus, VT is an appropriate method for eliminating features with little information. Furthermore, in a fair Cross-validation (CV) strategy, which we used in this study, only the unsupervised feature selection approach can be performed beforehand, keeping the same features selected in all training and testing iterations. Supervised feature selection approaches need to perform feature selection in the training set for each fold and apply it to the test set, which can vary the resulting features in each iteration [28]. Hence, we used VT to obtain a fixed number of resulting features for analysis. In addition, as an unsupervised filter-type method, VT has the advantage of evaluating features based on the intrinsic properties of data, unlike wrapper and hybrid unsupervised methods that rely on clustering algorithms and have the risk of overfitting [29].

Applying a VT of 97% (or 3%) significantly impacted the number of features. Features with 45 or fewer occurrences of either a 0 or 1 were discarded. These features were considered not to be crucial. The final number of features decreased from 30 923 to 462. To visualize the effect of feature selection on this dataset, Fig. 3 shows how the number of features is associated with the VT percentage.

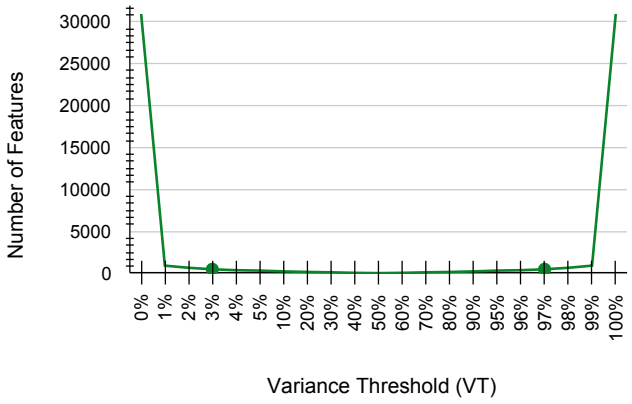


Fig. 3. Number of features versus VT percentage.

Fig. 4 shows a 30-bin histogram of the probability distribution and a Kernel Density Estimation (KDE) line (on a logarithmic scale) of the sum of events for each dataset feature after the feature selection process. The probabilities are distributed in a long-tail shape. It is worth noting the gap around the zero value of the x-axis, which were features eliminated by VT. The first and second bins have probabilities of 41.11% and 11.03%, respectively. The remaining 28 bins add up to a total of 47.86%. Therefore, the resultant dataset has a much less concentrated set of features.

Table II shows the distribution of the seven feature groups before and after the feature selection process. The API group

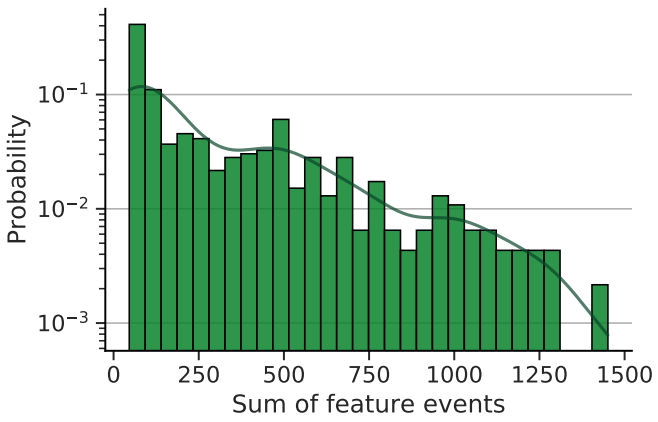


Fig. 4. Histogram of the probability distribution and a KDE line (on a logarithmic scale) of the sum of feature events in the resultant dataset.

was initially the smallest; however, after the feature selection, it represented 24.89% of the dataset, meaning a 61.17% usage of its group — the highest usage of all groups. Steps 3 and 5 of Algorithm 1 removed 40 instances, leaving 1484 samples in the dataset.

TABLE II
DISTRIBUTION OF FEATURE GROUPS BEFORE AND AFTER FEATURE SELECTION.

	Before		After	
Group	# Features	Percentage	# Features	Percentage
API	188	0.61%	115	24.89%
DROP	346	1.12%	10	2.17%
REG	6622	21.41%	225	48.70%
FILES	4141	13.39%	26	5.63%
EXT	935	3.02%	34	7.36%
DIR	2424	7.84%	16	3.46%
STR	16 267	52.60%	36	7.79%
Total	30 923	100%	462	100%

Table III lists the final amounts of each ransomware family and goodware. From this point on, the resultant dataset used in this study contains 1484 samples and 462 features after applying Algorithm 1.

TABLE III
RANSOMWARE FAMILIES AND GOODWARES RESULTING AFTER DATA PREPROCESSING.

ID	Family/Goodware	# Removed	# Resultant
1	Critroni	3	47
2	CryptLocker	2	105
3	CryptoWall	11	35
4	Kollah	2	23
5	Kovter	2	62
6	Locker	1	96
7	Matsnu	0	59
8	Pgpcoder	0	4
9	Reveton	0	90
10	TeslaCrypt	1	5
11	Trojan-Ransom	1	33
0	Goodware	17	925
-	Total:	40	1484

To effectively evaluate the dataset and generalize our model, we used six ML algorithms already evaluated for ransomware

detection in [18]: NB, KNN, LR, RF, SGD, and SVM. These techniques represent the most commonly used categories of supervised learning, including Bayesian, regression, ensemble, and instance-based learning. They are widely applied for binary classification, and each algorithm presents a different internal mechanism and method of handling data, as well as their advantages and disadvantages, aiding us to comprehensively explore the nature of the data [18], [30]. These ML algorithms are also used for multiclass classification, so future comparative studies of the multiclass domain can be based on the same techniques [31].

The performance of modern ML methods is highly dependent on their hyperparameter adjustments, which aid in maximizing the outcomes without overfitting, underfitting, or creating a high variance. Therefore, to obtain better results and compare the techniques more fairly, each classifier was tuned on hyperparameters that were frequently adjusted in the literature [32]–[34]. We used the scikit-learn GridSearchCV method with $K = 10$ and accuracy as a scoring strategy to evaluate the performance of the cross-validated model on the test set. Table X in Appendix A lists the final hyperparameters (tuned and default) used for each estimator.

For every classification, we performed an average of ten iterations of stratified K-fold CV with $K = 10$. To evaluate the results, we used well-known ML metrics such as accuracy, balanced accuracy, precision, recall, F-measure (F1), and Area Under the Curve (AUC) of the Receiver Operating Characteristic (ROC). Table IV presents the mathematical formula of the evaluation metrics. These metrics can be interpreted as relations based on: True Positive (TP) - number of ransomware samples correctly classified as ransomware; True Negative (TN) - number of goodware samples correctly classified as goodware; False Positive (FP) - number of goodware samples incorrectly classified as ransomware; and False Negative (FN) - number of ransomware samples incorrectly classified as goodware.

Accuracy is defined as the total number of samples correctly classified, divided by the total number of samples. Balanced accuracy is defined as the average of the proportion corrects of each class individually, which avoids inflated performance estimates on imbalanced datasets. Precision indicates the ability of the classifier to not label a positive sample as negative.

Recall, also known as True Positive Rate (TPR), evaluates the ability of the classifier to find all the positive samples, that is, to predict the ransomware class correctly. False Positive Rate (FPR) indicates the proportion of the negative class incorrectly classified by the classifier. F-measure (F1) is the harmonic mean of precision and recall, which can be interpreted as the performance of the model.

ROC is a graphical approach for analyzing the performance of a classifier and indicates the prediction algorithm's ability to distinguish between positive and negative outcomes. This curve is represented by FPR versus TPR at various threshold values. Therefore, by computing the area under the ROC curve (ROC AUC), the curve information can be summarized in one number where the higher the value, the better [35].

The combined interpretation of the results of these evaluation metrics provides better information about model performance [36]. In this study, the results are presented as 0.95 Confidence Interval (CI). When presenting comparative results, bold numbers indicate the best results.

TABLE IV
EVALUATION METRICS AND MATHEMATICAL FORMULAS.

Metric	Formula
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$
Balanced acc.	$\frac{1}{2}\left(\frac{TP}{TP+FN}+\frac{TN}{TN+FP}\right)$
Precision	$\frac{TP}{TP+FP}$
Recall (TPR)	$\frac{TP}{TP+FN}$
FPR	$\frac{FP}{FP+TN}$
F-measure (F1)	$2\left(\frac{Precision \times Recall}{Precision+Recall}\right)$

IV. EXPERIMENT RESULTS AND DISCUSSION

This section compares the results obtained between this study and related research, and then answers the four formulated questions investigating the main aspects of ransomware behavioral features and classification. The dataset was evaluated after data preprocessing and feature selection (Algorithm 1) using six ML estimators, the results of which are presented in Table V.

The RF algorithm outperformed all evaluation metrics, with an accuracy of 98.48%, balanced accuracy of 98.35%, precision of 98.17%, recall of 97.82%, F-measure of 97.98%, and ROC AUC of 99.87%. This result is consistent with the results of [37], in which 23 machine learning algorithms were investigated for malware detection, and it was concluded that RF provides optimal performance, with high accuracy and low training and testing times. We compared our best results with those in [12], [23], [24] using the same dataset. These studies did not provide metrics other than accuracy, and thus Fig. 5 shows a comparison of the results using this single metric.

Sgandurra et al. [12] used an RLR estimator to achieve an accuracy of 96.34% using 400 features, Abbasi et al. [23] achieved an accuracy of 97.34% using the RLR estimator and 822±59 features, and Khan et al. [24] obtained an 87.90% detection rate using a linear regression model based on active learning with 26 features. Our approach achieved the highest accuracy (98.48%) using 462 selected features; therefore, we consider that we have a consistent model to proceed with the assessments that will follow.

A. Should all groups of features be equally considered?

Because of their malicious nature, most ransomwares are expected to have determinant operating characteristics, which

TABLE V
EVALUATION OF SIX ML ALGORITHMS: METRIC (%) ± 0.95 CI (%)

Metric	NB	KNN	LR	RF	SGD	SVM
Accuracy	85.90 ± 0.13	96.23 ± 0.07	98.09 ± 0.05	**98.48 ± 0.05**	97.97 ± 0.06	98.32 ± 0.05
Balanced Acc.	87.78 ± 0.11	96.12 ± 0.07	97.92 ± 0.06	**98.35 ± 0.06**	97.83 ± 0.07	98.15 ± 0.06
Precision	74.62 ± 0.20	94.47 ± 0.14	97.74 ± 0.09	**98.17 ± 0.10**	97.02 ± 0.11	98.07 ± 0.09
Recall	95.40 ± 0.14	95.67 ± 0.12	97.21 ± 0.10	**97.82 ± 0.09**	97.50 ± 0.12	97.48 ± 0.10
F-measure	83.66 ± 0.13	95.03 ± 0.09	97.46 ± 0.07	**97.98 ± 0.07**	97.23 ± 0.08	97.75 ± 0.07
ROC AUC	89.78 ± 0.10	98.33 ± 0.05	99.66 ± 0.02	**99.87 ± 0.01**	99.59 ± 0.02	99.67 ± 0.02

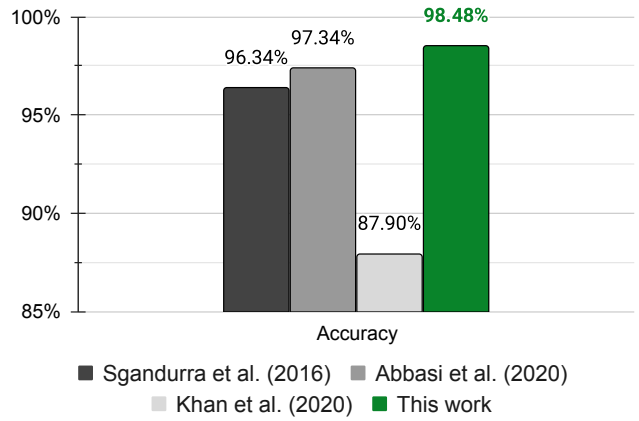


Fig. 5. Highest ransomware/goodware classification accuracy in [12], [23], [24] and the present study.

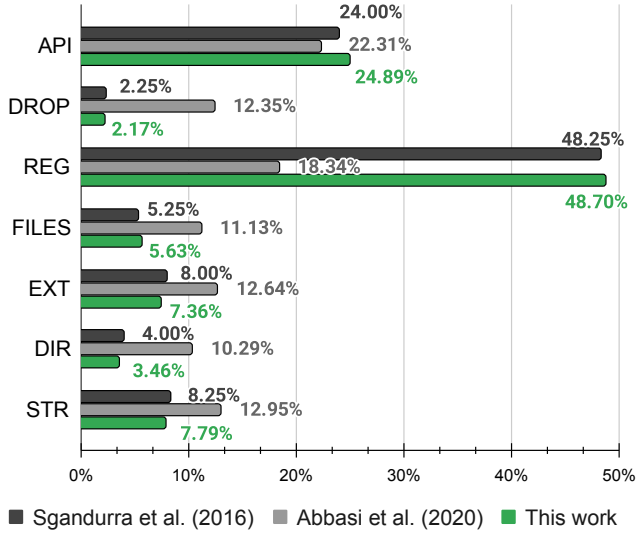


Fig. 6. Criterion 1: percentage per group in the resultant datasets.

can be analyzed by dividing them into seven feature groups and determining their relevance. Khan et al. [24] did not analyze the relevance of the groups. Sgandurra et al. [12] and Abbasi et al. [23] used the percentage per group criterion in the dataset. Other mentions to this criterion will be referenced as Criterion 1.

Fig. 6 shows Criterion 1 applied for this and related studies. Our results indicate that REG is the most important group. Although Sgandurra et al. [12] concluded the same, Abbasi et al. [23] deduced that the API group is the most significant. All comparison studies found that API, REG, and STR are the top-three groups using Criterion 1. However, it is important to note that the groups in the dataset were initially imbalanced (see Table II), thus, defining the group relevance by this criterion may lead to inaccurate results.

To improve the criterion for evaluating these groups, we propose that the variance in percentage of the group from the original dataset determines its relevance. Criterion 2 will be used for further mentions of the proposed criterion. Eq. 2 shows how to calculate the importance (%) of each group using Criterion 2.

$$Crit.\ 2[Group] = \left(\frac{Group\ (\%)\ Result.\ Dataset}{Group\ (\%)\ Orig.\ Dataset} - 1 \right) \times 100 \quad (2)$$

Fig. 7 shows the application of Criterion 2. The results of our study pointed that the most substantial group of features is the API, having a high difference from the other groups. Among the 188 API features in the original dataset (merged), 115 were considered significant. The percentage in the dataset increased from ≈0.61% to 24.89%, indicating a variance of +3994%. The STR usage group ranged from 52.60% to 7.79% with a variance of -85%. The API and STR groups showed the best and worst variance, respectively. The other groups, i.e., DROP, REG, and EXT, showed positive variances, whereas DIR and FILES showed a negative variance. The results of the related studies were also modified, showing that the most important group to be considered is the API. This result corresponds with that reported in the literature [20], [21]. Furthermore, the DROP, EXT, and DIR groups increased their relevance, whereas the REG, FILES, and STR groups had the opposite result.

We also split the dataset into seven groups of features (according to Table II) for evaluation. Table VI presents the balanced accuracies of the six ML estimators for each group. This metric was chosen because it avoids inflated performance estimates on imbalanced datasets. Table VI shows that the API is the best individual feature group. With 115 API features, it was possible to achieve a balanced accuracy of 96.49% with RF; however, to assess the significance of a group more accurately, we must use the mean of the estimators to avoid

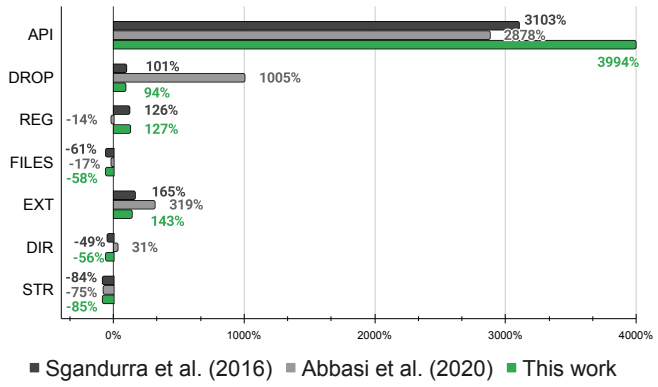


Fig. 7. Criterion 2: Percentage of variance per group in the resultant datasets.

favoring a single technique. The mean of the API group was 3.38%, which was still the highest. The worst group was FILES, presenting a mean of 82.28%, whereas the other five groups achieved similar results between ≈84% and 87%.

Table VII provides the ranking of the group features using Criterion 1, Criterion 2, and the classification mean. We can observe a better correlation between the classification mean with Criterion 2 than with Criterion 1. The results of our study indicate that the API group is the best individual group for ransomware/goodware classification.

The EXT group, despite using only 34 features, was shown to be the second best group based on both the classification as well as Criterion 2. The DROP group achieved third place in mean classification and fourth place in Criterion 2. The REG group was ranked fourth in terms of classification and third based on Criterion 2. The STR group was considered the least relevant when using Criterion 2, which was fifth when using the mean calculation.

DROP, REG, and STR had similar classification results, which can make the significance of the results less representative. The DIR group diverged from sixth when considering the mean classification to fifth after applying Criterion 2. The FILES group was the worst in terms of classification and sixth when using Criterion 2, and this group was considered the least relevant.

TABLE VII
RANKING OF GROUPS BY CRITERIA 1 AND 2 AND MEAN CLASSIFICATION.

	Criterion 1			Criterion 2			Classif. Mean
#	[12]	[23]	This work	[12]	[23]	This work	This work
1st	REG	API	REG	API	API	API	API
2nd	API	REG	API	EXT	DROP	EXT	EXT
3rd	STR	STR	STR	REG	EXT	REG	DROP
4th	EXT	EXT	EXT	DROP	DIR	DROP	REG
5th	FILES	DROP	FILES	DIR	REG	DIR	STR
6th	DIR	FILES	DIR	FILES	FILES	FILES	DIR
7th	DROP	DIR	DROP	STR	STR	STR	FILES

We found that all groups of features should not be considered equally. Although the API group has a major importance, an appropriate combination of groups of features can provide a better detection performance. However, it is crucial to emphasize that the results found are pointers to the binary classification of ransomware and goodware; thus, to better understand the behavior of ransomware, we can investigate individual features associated with ransomware and goodware.

B. What changes from Ransomware to Goodware?

The feature selection methods used in this study and the related research aim to achieve the best performance; however, they consider the two targets equally, and thus the resulting features are correlated to both ransomware and goodware in an indistinguishable manner. Hence, we proposed Eq. 3 to distinguish the features most related to ransomware and goodware for this specific dataset. First, the ransomware and goodware samples were separated into two sets. Subsequently, for each feature, the mean of each set was calculated. Finally, the mean of each feature for each set was subtracted.

$$Score[X] = \frac{1}{n}\sum_{i=1}^{n} X(i) - \frac{1}{m}\sum_{k=1}^{m} X(k) \tag{3}$$

In Eq. 3, X is a feature, $X(i)$ is the value of feature X in the ransomware sample i, and n is the total number of ransomware samples. Similarly, $X(k)$ denotes the value of feature X in the goodware sample k, and m is the total number of goodware samples. The results can vary within the range of -1.0 and 1.0. A feature calculated at -1.0 means a feature correlated to goodware. The opposite is also true, i.e., a feature calculated at 1.0, which is wholly associated with ransomware. For classification algorithms, the most relevant features are at

TABLE VI
EVALUATION OF THE SEVEN GROUPS OF FEATURES: BALANCED ACCURACY (%) ± 0.95 CI (%)

Group	# Features	NB	KNN	LR	RF	SGD	SVM	Mean
API	115	83.07 ± 0.16	94.95 ± 0.11	95.07 ± 0.08	**96.49 ± 0.08**	95.34 ± 0.09	95.35 ± 0.10	93.38 ± 0.10
DROP	10	86.76 ± 0.12	84.15 ± 0.48	86.58 ± 0.11	**87.35 ± 0.12**	86.81 ± 0.14	86.92 ± 0.13	86.43 ± 0.18
REG	225	80.58 ± 0.12	85.38 ± 0.21	87.90 ± 0.13	**88.69 ± 0.14**	86.28 ± 0.20	88.24 ± 0.13	86.18 ± 0.16
FILES	26	75.06 ± 0.27	81.78 ± 0.30	83.25 ± 0.16	**86.32 ± 0.12**	83.35 ± 0.21	83.94 ± 0.15	82.28 ± 0.20
EXT	34	86.00 ± 0.12	84.61 ± 0.28	87.22 ± 0.13	**87.93 ± 0.11**	86.86 ± 0.14	87.35 ± 0.13	86.66 ± 0.15
DIR	16	82.03 ± 0.18	76.92 ± 0.70	86.88 ± 0.11	**87.82 ± 0.12**	86.94 ± 0.16	86.75 ± 0.11	84.56 ± 0.23
STR	36	82.08 ± 0.14	88.27 ± 0.14	85.31 ± 0.16	**88.86 ± 0.13**	84.80 ± 0.15	84.34 ± 0.17	85.61 ± 0.15

the ends of the range. However, for a better understanding of ransomware behavior, the most substantial features are those close to 1.0, which can be interpreted as a higher occurrence of ransomware events and a lower occurrence of goodware events.

Our proposal for distinguishing features is purely statistical and independent of algorithms. Eq. 3 emphasizes the more significant presence or absence of a certain feature individually, without relating features to each other, which makes the experiment reproduction always obtain the same results, even when the dataset is split. Hence, it is appropriate to distinguish features between the analyzed ransomware and goodware classes.

Applying Eq. 3 to the dataset with 462 features showed that 154 features obtained scores higher than 0, that is, these features are more likely to be observed in ransomware events. By contrast, 308 of the remaining features with scores lower than 0 were more likely to be seen in goodware events. Table VIII lists the top-15 features most frequently found in ransomware and goodware.

The results in Table VIII represent the typical behavior of ransomware and goodware in an OS. We can see much higher scores on goodware than on ransomware; in addition, the highest ransomware feature score is lower than the lowest goodware feature score, which means that the features in this dataset are more capable of being associated with goodware than with ransomware. Analyzing all of these features, we can note the predominance of the API group with 10 out of 15 occurrences in ransomware and 7 out of 15 in goodware, which corroborates the results of the relevance of groups found thus far.

We observed a behavior change from ransomware to goodware because the API features of the former are generally related to the execution of a process in the OS (see Table XII in Appendix C). Also related to the execution of a process, the OPENED:Manifest feature of the EXT group indicates an opened file containing settings that inform Windows how to handle a program when it is started; the string KERNEL32.dll is associated with an essential Dynamic-link Library (DLL) that handles memory management, input/output operations, and system interrupts; similarly, the MSVCRT.dll contains program code that enables applications written in Microsoft Visual C++ to run properly; and the mscoree.dll is a Microsoft Runtime Execution Engine, which contains fundamental functions of the Microsoft.NET framework, providing the possibility to connect information, systems, people and devices through software. The OPENED:Device\ of feature the FILES group may indicate an attempt to search the user files on other devices such as a pen drive or external hard drive.

Meanwhile, the features of goodware are related to file and directory operations: the APIs CreateDirectoryW, SetFilePointer, SearchPathW, and GetFileAttributes (see Table XIII in Appendix C); the temporary files in DROP tmp, EXT OPENED:tmp, and EXT WRITTEN:tmp; and the enumeration of a standard directory in DIR. The high score obtained by API CreateDirectoryW, the handling of temporary files, and the two REG features in goodware might indicate that many of the samples collected were software installers, which needed to build their directory structure, obtain system and user information for the installation process, and then remove unnecessary files.

TABLE VIII
TOP-15 FEATURES THAT APPEAR THE MOST IN RANSOMWARE (LEFT) AND GOODWARE (RIGHT).

	Ransomware			Goodware		
#	Score	Group	Name	Score	Group	Name
1	0.402	API	NtTerminateProcess	-0.748	API	CreateDirectoryW
2	0.351	STR	KERNEL32.dll	-0.712	DIR	ENUMERATED:C:\Documents and Settings\MyUser\Desktop\test-personal-files\img\
3	0.331	API	SetUnhandledExceptionFilter	-0.645	EXT	OPENED:tmp
4	0.234	API	WSAStartup	-0.642	DROP	tmp
5	0.207	EXT	OPENED:Manifest	-0.591	API	CoInitializeEx
6	0.194	API	WriteProcessMemory	-0.567	API	SetFilePointer
7	0.191	API	NtGetContextThread	-0.555	API	NtReadVirtualMemory
8	0.190	API	NtSetContextThread	-0.548	EXT	WRITTEN:tmp
9	0.175	API	__exception__	-0.538	API	SearchPathW
10	0.166	API	NtResumeThread	-0.481	API	GetFileAttributesW
11	0.165	API	GetComputerNameW	-0.455	API	SizeofResource
12	0.162	STR	mscoree.dll	-0.454	EXT	WRITTEN:dll
13	0.154	API	NtDeviceIoControlFile	-0.459	DROP	dll
14	0.142	FILES	OPENED:Device\	-0.444	REG	OPENED:HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\
15	0.132	STR	MSVCRT.dll	-0.439	REG	OPENED:HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Uninstall\

C. What are the most relevant APIs?

To find the behavior patterns in ransomware during the destruction phase, we applied Eq. 3 for the most relevant group of features. Among the 115 API features, 40 were mostly associated with ransomware and 75 with goodware. Figs. 8 and 9 show the comparative presence of the 20 API features that appear the most in ransomware and goodware, respectively. The scores obtained and the descriptions of these APIs are provided in Appendix C.

Analyzing Fig. 8, it is possible to identify a feature pattern in ransomware, in which 11 out of 20 features involve the thread/process. In addition, the presence of WriteProcessMemory and ReadProcessMemory shows that ransomware is more likely to use physical memory than virtual memory. The WSAStartup feature represents a function that opens a communication socket on the system, which can be interpreted as an attempt to connect with the C2C server. GetComputerName(A, W) can help cybercriminals identify their targets after communication and even customize the ransom note shown to the user. GetFileType, the only API related to file operation, is an essential function used to discover what type of file is important to the user, and thus the malware will likely not encrypt system files non-relevant to the ransom (e.g., a .BAT or .COM file), but will encrypt valuable personal files instead (e.g., a .DOC or .JPG file). We also noticed that WriteProcessMemory, NtGetContextThread, NtSetContextThread, and NtDeviceIoControlFile appear fairly well in ransomware, but only a few appear in goodware. These features are determinant in ransomware actions and must be carefully inspected by anti-malware systems.

By contrast, the main patterns linked to goodware are the operations of files, directories, and resources, which are present in 12 of the 20 features in Fig. 9. An overwhelming difference in the CreateDirectoryW feature indicates that goodware tends to create new directories whereas ransomware does not. We suppose that the only directory created by ransomware is for installing itself in the system and placing a ransom note file. NtReadVirtualMemory shows that goodware tends to use more virtual memory than ransomware does. CoInitializeEx is the only thread-related feature, while the other features have a broad general meaning. These findings corroborate the answers to the title question in Section IV-B.

D. Where are the encryption APIs?

Of the 232 API features in the entire original dataset (not merged), only eight were directly associated with encryption. We searched for the word ”crypt” in the feature description file containing the name of each variable and found CryptAcquireContextA, CryptAcquireContextW, CryptExportKey, CryptDecodeObjectEx, CryptEncrypt, CryptCreateHash, CryptGenKey, and CryptHashData. All of these API

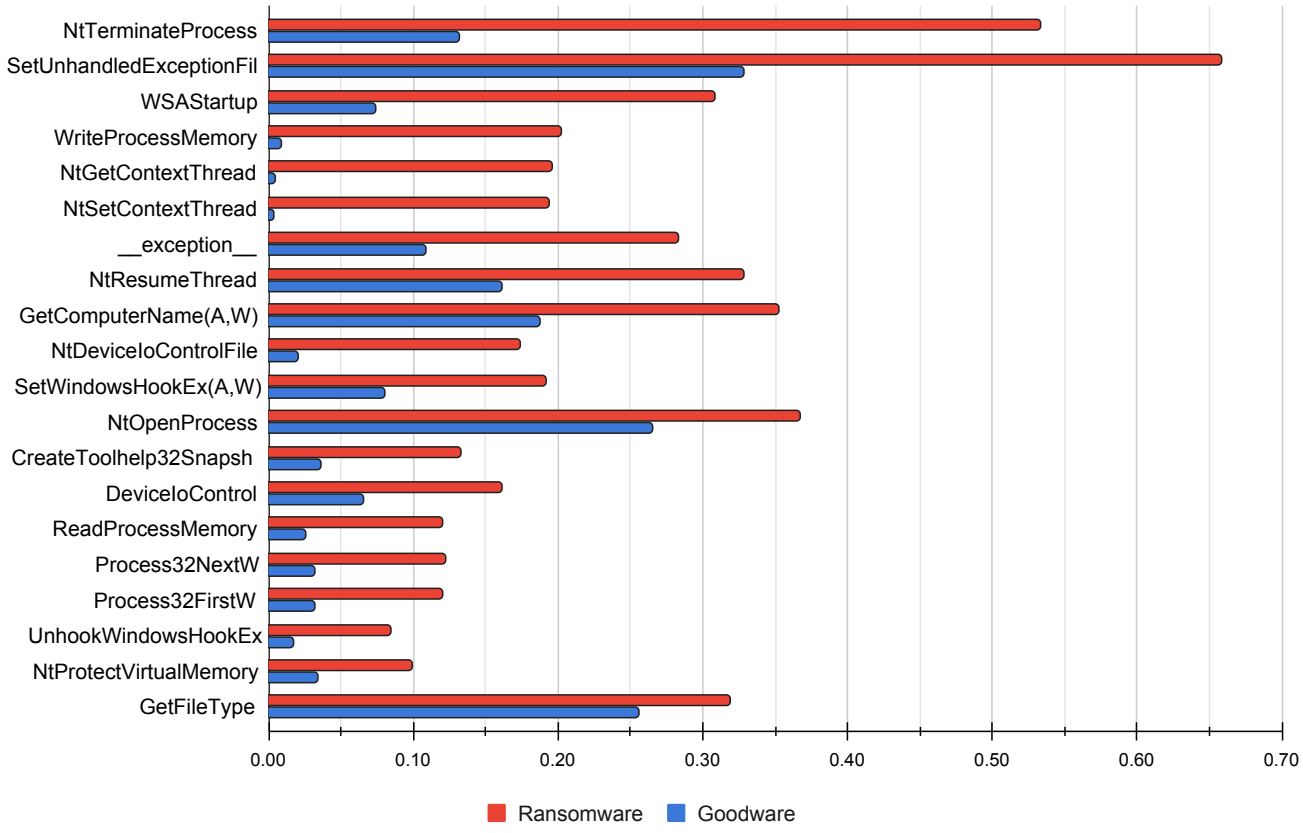


Fig. 8. Comparative top-20 features that appear the most in ransomware.

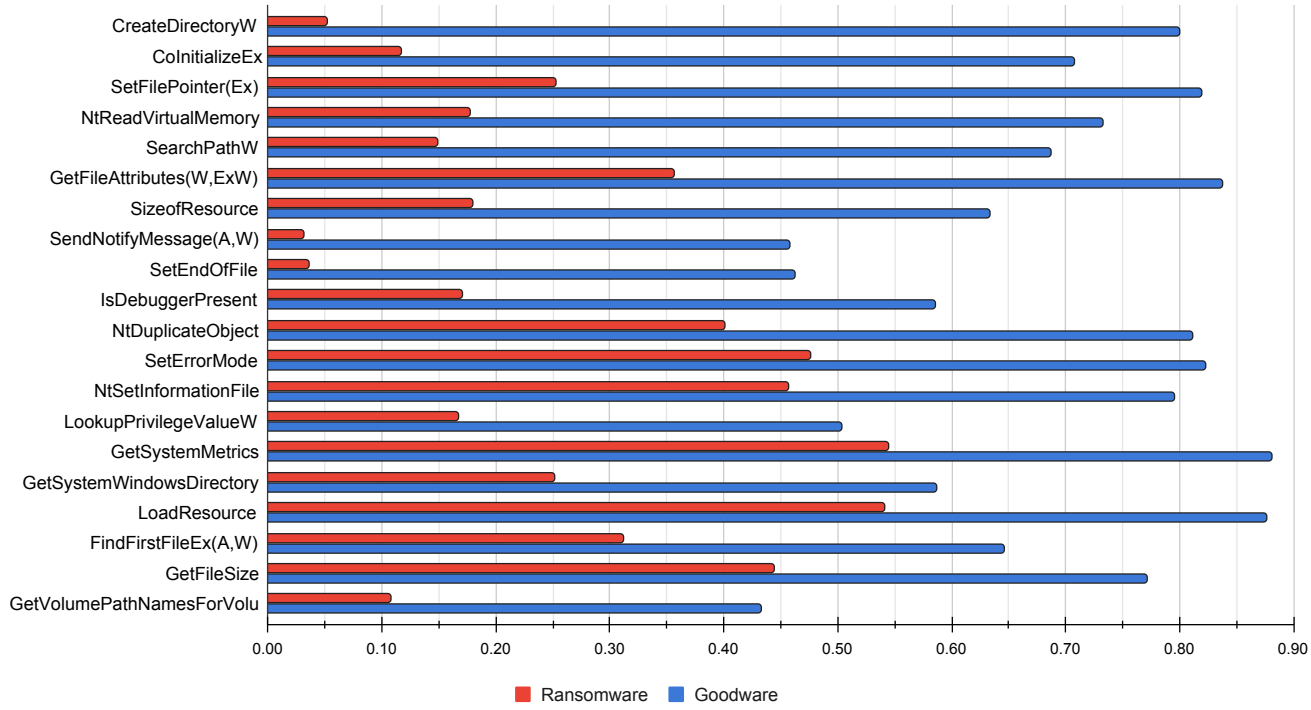


Fig. 9. Comparative top-20 features that appear the most in goodware.

features belong to Wincrypt.h[4], a native API of Windows OS. We also found five API features linked to asymmetrical encryption and present in Wincrypt.h: CertCreateCertificateContex, CertControlStore, CertOpenStore, CertOpenSystemStoreA, and CertOpenSystemStoreW. As discussed in Section III-B, we merged CryptAcquireContextA with CryptAcquireContextW and CertOpenSystemStoreA with CertOpenSystemStoreW.

Table IX provides a summary of the occurrence of those features found in ransomware and goodware. Because the primary goal of ransomware is to encrypt the user files as much as possible, this type of feature was expected to appear quite often in the ransomware samples. However, only 48 (out of 582) of the ransomware cases called at least one of those encryption APIs, thus, most of the ransomware did not use native Windows encryption code. Goodware samples that used those APIs numbered 66 (out of 942). Ransomware and goodware samples that called at least one of those APIs totaled 114, or approximately 7.48% of the entire original dataset (merged) — the resultant dataset from step 2 of Algorithm 1.

Some of the APIs in Table IX are perfectly correlated, that is, they appear precisely in the same samples. This is the case for CryptExportKey with CryptEncrypt and CryptGenKey; and CryptCreateHash with CryptHashData. After applying Algorithm 1, CryptExportKey, CryptEncrypt, CryptGenKey, CertCreateCertificateContex, CertControlStore, CertOpenStore, and CertOpenSystemStore(A,W) features were removed because they have low variance. Algorithm 1 did not remove any of the 114 samples that used native encryption Windows API.

TABLE IX
OCCURRENCE OF NATIVE ENCRYPTION WINDOWS APIS IN ORIGINAL DATASET (MERGED).

API	Ransomware	Goodware
CryptAcquireContext(A,W)	45	66
CryptDecodeObjectEx	21	38
CryptCreateHash	28	42
CryptHashData	28	42
CryptExportKey	11	0
CryptEncrypt	11	0
CryptGenKey	11	0
CertCreateCertificateContex	0	1
CertControlStore	0	35
CertOpenStore	0	37
CertOpenSystemStore(A,W)	3	0

Applying Eq. 3 to calculate the scores of the remaining features, CryptAcquireContext(A,W), CryptDecodeObjectEx, CryptCreateHash, and CryptHashData returned scores close to zero: 0.0091, 0.0047, 0.0047, and -0.0035, respectively. Hence, native encryption Windows API features are not significant for binary ransomware/goodware classification. Perhaps, for family ransomware classification, these features could add some information gain to the model. In addition, it should be noted that this dataset represents executed or non-executed

[4]https://docs.microsoft.com/en-us/windows/win32/api/wincrypt/

events, but does not indicate the frequencies of such events; therefore, because of the essence of ransomware actions, some of these features can be frequently called by the executable binary, which can add valuable information to distinguish ransomware from goodware.

V. Conclusions

In this study, we analyzed Crypto ransomware behavioral features to better understand its actions during the destruction phase. In this analysis, we merged different API versions with similar names, resulting in information gain during the investigation. Likewise, we proposed an improvement in the criterion for evaluating groups in datasets with imbalanced feature groups. Four questions were formulated: Should all groups of features be equally considered, what changes from ransomware to goodware, what are the most relevant APIs, and where are the encryption APIs?. The answers to these questions are as follows:

- The API is the best individual feature group to distinguish ransomware from goodware; however, an anti-ransomware system should inspect some key features from other groups to improve the detection rate, such as the manifested files, strings mentioning DLLs, and the attempts to access other storage devices.
- Changes to the pattern behavior between ransomware and goodware were identified. Whereas the first tends to involve thread/process manipulation, context strings and file operations related to process execution, physical memory operations, a socket connection, computer identification, and file type discovery operations, the second is more likely to present virtual memory, directories, and resource operations, including temporary files and DLLs.
- The 10-most relevant APIs identified for ransomware events were NtTerminateProcess, SetUnhandledExceptionFilter, WSAStartup, WriteProcessMemory, NtGetContextThread, NtSetContextThread, __exception__, NtResumeThread, GetComputerName(A,W), and NtDeviceIoControlFile. These API calls must be carefully monitored through protection systems.
- As much as cryptographic APIs are expected to be decisive for identifying ransomware, native encryption Windows API have proven to be less crucial for binary classification, with approximately equivalent amounts of samples from both classes using them.

All of these findings are relevant to improving the understanding of ransomware behavior for the design of more robust and reliable protection systems against such malware. However, this study focused on binary ransomware/goodware classification using the presence or absence of certain events. Thus, future research can analyze the frequencies of such events during process execution and use not only a dynamic but also static analysis. Researchers can also use other metrics of system behavior during ransomware execution, such as CPU, memory, and input/output devices.

Appendix A

TABLE X
HYPERPARAMETERS USED FOR EACH CLASSIFIER. TUNED IN **'BOLD'** AND DEFAULT IN 'REGULAR' TEXT.

Classifier	Hyperparameters and values
NB	**alpha=0.0**, **fit_prior=True**, binarize=0.0, class_prior=None
KNN	**n_neighbors=5**, **weights='distance'**, **metric='manhattan'**, algorithm='auto', leaf_size=30, p=2, metric_params=None, n_jobs=None
LR	**penalty='l2'**, **C=1.0**, **solver='lbfgs'**, **max_iter=100**, dual=False, tol=0.0001, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, multi_class='auto', verbose=0, warm_start=False, n_jobs=None, l1_ratio=None
RF	**n_estimators=150**, **criterion='gini'**, **max_depth=150**, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False, class_weight=None, ccp_alpha=0.0, max_samples=None
SGD	**loss='hinge'**, **penalty='l1'**, **tol=0.001**, alpha=0.0001, l1_ratio=0.15, fit_intercept=True, max_iter=1000, shuffle=True, verbose=0, epsilon=0.1, n_jobs=None, random_state=None, learning_rate='optimal', eta0=0.0, power_t=0.5, early_stopping=False, validation_fraction=0.1, n_iter_no_change=5, class_weight=None, warm_start=False, average=False
SVM	**C=15**, **kernel='rbf'**, **gamma=0.01**, degree=3, coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=- 1, decision_function_shape='ovr', break_ties=False, random_state=None

Appendix B
Merged APIs

Table XI lists all 44 API versions that we found with similar functions, but with different nomenclature. The API on the left side is the one that remained with the final label (position) after the merge. Labels and names according to the description text file *VariableNames.txt* of the original dataset.

TABLE XI
LIST OF ALL MERGED APIS.

Label	Name	Label	Name
4	GetSystemDirectoryA	25	GetSystemDirectoryW
5	WriteConsoleA	9	WriteConsoleW
11	RemoveDirectoryA	49	RemoveDirectoryW
13	FindFirstFileExA	105	FindFirstFileExW
15	OpenServiceW	40	OpenServiceA
19	HttpOpenRequestA	23	HttpOpenRequestW
20	HttpSendRequestA	24	HttpSendRequestW
21	GetUserNameA	54	GetUserNameExW
21	GetUserNameA	56	GetUserNameExA
21	GetUserNameA	88	GetUserNameW
30	GetDiskFreeSpaceExW	233	GetDiskFreeSpaceW
34	FindWindowExW	46	FindWindowExA
34	FindWindowExW	182	FindWindowA
34	FindWindowExW	235	FindWindowW

Continued on the next column

Continued from the previous column

Label	Name	Label	Name
37	RegQueryInfoKeyW	38	RegQueryInfoKeyA
41	InternetOpenA	53	InternetOpenW
48	RegEnumValueA	59	RegEnumValueW
50	RegDeleteKeyA	60	RegDeleteKeyW
52	WSASocketA	58	WSASocketW
67	SetFilePointer	153	SetFilePointerEx
69	FindResourceA	87	FindResourceW
69	FindResourceA	146	FindResourceExA
69	FindResourceA	160	FindResourceExW
71	DeleteFileW	125	NtDeleteFile
73	CryptAcquireContextA	95	CryptAcquireContextW
74	CreateServiceA	83	CreateServiceW
91	GetSystemWindowsDirectoryA	107	GetSystemWindowsDirectoryW
94	MessageBoxTimeoutA	96	MessageBoxTimeoutW
99	RegEnumKeyExW	100	RegEnumKeyExA
99	RegEnumKeyExW	149	RegEnumKeyW
121	OpenSCManagerW	135	OpenSCManagerA
124	GetComputerNameW	141	GetComputerNameA
128	SetWindowsHookExW	148	SetWindowsHookExA
130	RegSetValueExW	143	RegSetValueExA
138	RegCreateKeyExW	155	RegCreateKeyExA
147	InternetConnectW	168	InternetConnectA
151	GetFileAttributesW	161	GetFileAttributesExW
152	RegDeleteValueA	163	RegDeleteValueW
181	GetFileSize	213	GetFileSizeEx
184	CertOpenSystemStoreW	201	CertOpenSystemStoreA
198	InternetCrackUrlW	216	InternetCrackUrlA
206	RegQueryValueExA	228	RegQueryValueExW
217	RegOpenKeyExW	230	RegOpenKeyExA
219	SendNotifyMessageW	229	SendNotifyMessageA

Concluded

APPENDIX C

TABLE XII
SCORE AND DESCRIPTION OF THE APIS FEATURES MOSTLY PRESENT IN RANSOMWARE.

Score	Name	Description
0.4023	NtTerminateProcess	Terminates the specified process and all of its threads.
0.3308	SetUnhandledExceptionFilter	Enables an application to supersede the top-level exception handler of each thread of a process.
0.2342	WSAStartup	Initiates use of the Winsock DLL by a process.
0.1935	WriteProcessMemory	Writes data to an area of memory in a specified process. The entire area to be written to must be accessible or the operation fails.
0.1907	NtGetContextThread	Retrieves the context of the specified thread.
0.1900	NtSetContextThread	Sets the context for the specified thread.
0.1745	__exception__	Raises an exception in the calling thread.
0.1663	NtResumeThread	Decrements a thread's suspend count. When the suspend count is decremented to zero, the execution of the thread is resumed.
0.1654	GetComputerName(A,W)	Retrieves the NetBIOS name of the local computer. This name is established at system startup, when the system reads it from the registry.
0.1541	NtDeviceIoControlFile	Builds descriptors for the supplied buffer(s) and passes the untyped data to the device driver associated with the file handle.
0.1114	SetWindowsHookEx(A,W)	Installs an application-defined hook procedure into a hook chain. You would install a hook procedure to monitor the system for certain types of events. These events are associated either with a specific thread or with all threads in the same desktop as the calling thread.
0.1019	NtOpenProcess	Opens a handle to a process object and sets the access rights to this object.
0.0967	CreateToolhelp32Snapshot	Takes a snapshot of the specified processes, as well as the heaps, modules, and threads used by these processes.
0.0961	DeviceIoControl	Sends a control code directly to a specified device driver, causing the corresponding device to perform the corresponding operation.
0.0950	ReadProcessMemory	Copies the data in the specified address range from the address space of the specified process into the specified buffer of the current process.
0.0903	Process32NextW	Retrieves information about the next process recorded in a system snapshot.
0.0885	Process32FirstW	Retrieves information about the first process encountered in a system snapshot.
0.0679	UnhookWindowsHookEx	Removes a hook procedure installed in a hook chain by the SetWindowsHookEx function.
0.0649	NtProtectVirtualMemory	Locks the committed virtual address to the specific process.
0.0633	GetFileType	Retrieves the file type of the specified file.

TABLE XIII
SCORE AND DESCRIPTION OF THE APIS FEATURES MOSTLY PRESENT IN GOODWARE.

Score	Name	Description
-0.7481	CreateDirectoryW	Creates a new directory.
-0.5907	CoInitializeEx	Initializes the COM library for use by the calling thread, sets the thread's concurrency model, and creates a new apartment for the thread if one is required.
-0.5672	SetFilePointer(Ex)	Moves the file pointer of the specified file.
-0.5548	NtReadVirtualMemory	Reads a virtual memory block.
-0.5380	SearchPathW	Searches for a specified file in a specified path.
-0.4808	GetFileAttributes(W,ExW)	Retrieves file system attributes for a specified file or directory.
-0.4546	SizeofResource	Retrieves the size, in bytes, of the specified resource.
-0.4269	SendNotifyMessage(A,W)	Sends the specified message to a window or windows.
-0.4258	SetEndOfFile	Sets the physical file size for the specified file to the current position of the file pointer.
-0.4149	IsDebuggerPresent	Determines whether the calling process is being debugged by a user-mode debugger.
-0.4101	NtDuplicateObject	Creates a handle that is a duplicate of the specified source handle.
-0.3469	SetErrorMode	Controls whether the system will handle the specified types of serious errors or whether the process will handle them.
-0.3384	NtSetInformationFile	Changes various kinds of information about a file object.
-0.3363	LookupPrivilegeValueW	Retrieves the locally unique identifier (LUID) used on a specified system to locally represent the specified privilege name.
-0.3362	GetSystemMetrics	Retrieves the specified system metric or system configuration setting.
-0.3355	GetSystemWindowsDirectory(A,W)	Retrieves the path of the shared Windows directory on a multi-user system.
-0.3354	LoadResource	Retrieves a handle that can be used to obtain a pointer to the first byte of the specified resource in memory.
-0.3341	FindFirstFileEx(A,W)	Determines the location of a resource with the specified type and name in the specified module.
-0.3272	GetFileSize	Retrieves the size of the specified file, in bytes.
-0.3251	GetVolumePathNamesForVolumeNameW	Retrieves a list of drive letters and mounted folder paths for the specified volume.

REFERENCES

[1] T. Ahmad, "Corona virus (covid-19) pandemic and work from home: Challenges of cybercrimes and cybersecurity," *SSRN Electron. J.*, 4 2020, doi: 10.2139/ssrn.3568830.

[2] N. A. Khan, S. N. Brohi, and N. Zaman, "Ten deadly cyber security threats amid covid-19 pandemic," 5 2020, doi: 10.36227/TECHRXIV.12278792.V1.

[3] E. Kolodenker, W. Koch, G. Stringhini, and M. Egele, "Paybreak: Defense against cryptographic ransomware," in *Proc. 2017 ACM Asia Conf. Comput. Commun. Security*, ser. ASIA CCS '17. New York, NY, USA: CM, 2017, p. 599–611, doi: 10.1145/3052973.3053035.

[4] S. H. Kok, A. Abdullah, N. Z. Jhanjhi, and M. Supramaniam, "Ransomware, threat and detection techniques: A review," *Int. J. Comput. Sci. Netw. Security*, vol. 19, pp. 136–146, 2019. [Online]. Available: http://paper.ijcsns.org/07_book/201902/20190217.pdf

[5] H. Oz, A. Aris, A. Levi, and A. S. Uluagac, "A survey on ransomware: Evolution, taxonomy, and defense solutions," 2021. [Online]. Available: https://arxiv.org/abs/2102.06249

[6] A. Arrott, A. Lakhotia, F. Leitold, and C. LeDoux, "Cluster analysis for deobfuscation of malware variants during ransomware attacks," in *2018 Int. Conf. Cyber Situational Awareness Data Analytics Assessment*, 2018, pp. 1–9, doi: 10.1109/CyberSA.2018.8551432.

[7] B. A. Saleh Al-rimy, M. A. Maarof, Y. A. Prasetyo, S. Z. Mohd Shaid, and A. F. Mohd Ariffin, "Zero-day aware decision fusion-based model for crypto-ransomware early detection," *Int. J. Integr. Eng.*, vol. 10, no. 6, Nov. 2018. [Online]. Available: https://publisher.uthm.edu.my/ojs/index.php/ijie/article/view/2828

[8] E. Torres and S. G. Yoo, "Detecting and neutralizing encrypting ransomware attacks by using machine-learning techniques: A literature review," *Int. J. Appl. Eng. Res.*, vol. 12, pp. 7902–7911, 5 2017. [Online]. Available: https://ripublication.com/ijaer17/ijaerv12n18_105.pdf

[9] B. Silva, R. Silveira, M. Silva Neto, P. Cortez, and D. Gomes, "A comparative analysis of undersampling techniques for network intrusion detection systems design," *J. Commun. Inf. Syst.*, vol. 36, no. 1, pp. 31–43, Feb. 2021, doi: 10.14209/jcis.2021.3.

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

[11] A. Atapour-Abarghouei, S. Bonner, and A. S. McGough, "A king's ransom for encryption: Ransomware classification using augmented one-shot learning and bayesian approximation," in *2019 IEEE Int. Conf. Big Data*, 2019, pp. 1601–1606, doi: 10.1109/BigData47090.2019.9005540.

[12] 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: https://arxiv.org/abs/1609.03020

[13] N. Alruhaily, B. Bordbar, and T. Chothia, "Towards an understanding of the misclassification rates of machine learning-based malware detection systems," in *Proc. 3rd Int. Conf. Inf. Syst. Security Privacy*, vol. 2017-January. SciTePress, 2017, pp. 101–112, doi: 10.5220/0006174301010112.

[14] N. Alruhaily, T. Chothia, and B. Bordbar, "A better understanding of machine learning malware misclassifcation," in *Commun. Comput. Inf. Sci.*, vol. 867. Springer Verlag, 2018, pp. 35–58, doi: 10.1007/978-3-319-93354-2_3.

[15] Y. Liu and Y. Wang, "A robust malware detection system using deep learning on api calls," in *Proc. 2019 IEEE 3rd Inf. Technol. Netw. Electron. Autom. Control Conf.* IEEE, 3 2019, pp. 1456–1460, doi: 10.1109/ITNEC.2019.8728992.

[16] J. Zhou, M. Hirose, Y. Kakizaki, and A. Inomata, "Evaluation to classify ransomware variants based on correlations between apis," in *Proc. 6th Int. Conf. Inf. Syst. Security Privacy*. SciTePress, 2020, pp. 465–472, doi: 10.5220/0008959904650472.

[17] F. Ullah, Q. Javaid, A. Salam, M. Ahmad, N. Sarwar, D. Shah, and M. Abrar, "Modified decision tree technique for ransomware detection at runtime through api calls," *Sci. Program.*, vol. 2020, 2020, doi: 10.1155/2020/8845833.

[18] S. I. Bae, G. B. Lee, and E. G. Im, "Ransomware detection using machine learning algorithms," *Concurrency Comput.: Practice Experience*, vol. 32, p. e5422, 4 2020, doi: 10.1002/cpe.5422.

[19] P. Black, A. Sohail, I. Gondal, J. Kamruzzaman, P. Vamplew, and P. Watters, "Api based discrimination of ransomware and benign cryptographic programs," in *Neural Inf. Process.*, H. Yang, K. Pasupa, A. C.-S. Leung, J. T. Kwok, J. H. Chan, and I. King, Eds. Cham: Springer, 2020, pp. 177–188, doi: 10.1007/978-3-030-63833-7_15.

[20] S. Kok, A. Abdullah, N. Jhanjhi, and M. Supramaniam, "Prevention of crypto-ransomware using a pre-encryption detection algorithm," *Comput.*, vol. 8, p. 79, 11 2019, doi: 10.3390/computers8040079.

[21] M. Scalas, D. Maiorca, F. Mercaldo, C. A. Visaggio, F. Martinelli, and G. Giacinto, "On the effectiveness of system api-related information for android ransomware detection," *Comput. Security*, vol. 86, pp. 168–182, 4 2019, doi: 10.1016/j.cose.2019.06.004.

[22] S. H. Kok, A. Abdullah, and N. Z. Jhanjhi, "Early detection of crypto-ransomware using pre-encryption detection algorithm," *J. King Saud University - Comput. Inf. Sci.*, 7 2020, doi: 10.1016/j.jksuci.2020.06.012.

[23] M. S. Abbasi, H. Al-Sahaf, and I. Welch, "Particle swarm optimization: A wrapper-based feature selection method for ransomware detection and classification," in *Appl. Evol. Comput.*, vol. 12104 LNCS. Cham: Springer, 4 2020, pp. 181–196, doi: 10.1007/978-3-030-43722-0_12.

[24] F. Khan, C. Ncube, L. K. Ramasamy, S. Kadry, and Y. Nam, "A digital dna sequencing engine for ransomware detection using machine learning," *IEEE Access*, vol. 8, pp. 119 710–119 719, 2020, doi: 10.1109/ACCESS.2020.3003785.

[25] D. W. Fernando, N. Komninos, and T. Chen, "A study on the evolution of ransomware detection using machine learning and deep learning techniques," *IoT*, vol. 1, no. 2, pp. 551–604, 2020, doi: 10.3390/iot1020030.

[26] N. Hampton, Z. Baig, and S. Zeadally, "Ransomware behavioural analysis on windows platforms," *J. Inf. Security Appl.*, vol. 40, pp. 44–51, 6 2018, doi: 10.1016/j.jisa.2018.02.008.

[27] C. Forbes, M. Evans, N. Hastings, and B. Peacock, *Statistical Distributions*. John Wiley & Sons, Inc., 12 2010, pp. 53–54, doi: 10.1002/9780470627242.ch7.

[28] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference, and prediction*, 2nd ed. New York: Springer, 2009, pp. 245–247, isbn: 978-0-387-84858-7.

[29] S. Solorio-Fernández, J. A. Carrasco-Ochoa, and J. F. Martínez-Trinidad, "A review of unsupervised feature selection methods," *Artif. Intell. Rev.*, vol. 53, no. 2, pp. 907–948, Feb 2020, doi:10.1007/s10462-019-09682-y.

[30] S. Ray, "A quick review of machine learning algorithms," in *2019 Int. Conf. Mach. Learn., Big Data, Cloud Parallel Comput. (COMITCon)*, 2019, pp. 35–39, doi: 10.1109/COMITCon.2019.8862451.

[31] P. Chaitra and D. R. S. Kumar, "A review of multi-class classification algorithms," *Int. J. Pure Appl. Math*, vol. 118, no. 14, pp. 17–26, 2018. [Online]. Available: https://acadpubl.eu/jsi/2018-118-14-15/articles/14/3.pdf

[32] J. N. van Rijn and F. Hutter, "Hyperparameter importance across datasets," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, ser. KDD '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 2367–2376, doi: 10.1145/3219819.3220058.

[33] J. Singh and J. Singh, "Assessment of supervised machine learning algorithms using dynamic api calls for malware detection," *Int. J. Comput. App.*, vol. 0, no. 0, pp. 1–8, 2020, doi: 10.1080/1206212X.2020.1732641.

[34] D. Paper, *Hands-on Scikit-Learn for Machine Learning Applications: Data Science Fundamentals with Python*. Berkeley, CA: Apress, 2020, pp. 165–188, doi: 10.1007/978-1-4842-5373-1_6.

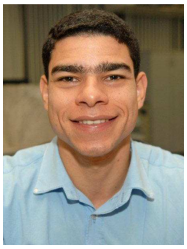
[35] M. Hossin and S. M.N, "A review on evaluation metrics for data classification evaluations," *Int. J. Data Mining Knowl. Manag. Process*, vol. 5, pp. 01–11, 03 2015, doi: 10.5121/ijdkp.2015.5201.

[36] D. M. W. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," *Int. J. Mach. Learn. Technol.*, pp. 37–63, 2011. [Online]. Available: https://arxiv.org/abs/2010.16061

[37] D. Carlin, P. O'Kane, and S. Sezer, "A cost analysis of machine learning using dynamic runtime opcodes for malware detection," *Comput. Security*, vol. 85, pp. 138–155, 2019, doi: 10.1016/j.cose.2019.04.018.

Caio Carvalho Moreira received the M.Sc. degree in electrical engineering from the Federal University of Pará (UFPA), Belém, Pará, Brazil, in 2013. He served as a computer system analyst at the Brazilian Air Force from 2013 to 2017. He is currently a Ph.D. student in the Electrical Engineering Graduate Program at UFPA. His contemporary research addresses ML techniques applied to network and system security, specifically, ransomware detection.

Claudomiro de Souza de Sales, Jr. received the Ph.D. in electrical engineering from the Federal University of Pará (UFPA), Belém, Pará, Brazil, in 2009. Since 2010, he has been with UFPA, where he is currently with the Computer Science Department and Electrical Engineering and Computer Science Graduate Programs. He is involved in developing new dimension reduction and data visualization techniques, applying ML algorithms for ransomware detection and bioinformatics, and developing new metaheuristics and variants for PSO and GA.



Davi Carvalho Moreira received the Ph.D. in electrical engineering from the Federal University of Pará (UFPA), Belém, Pará, Brazil, in 2021. Since 2005, he has been with the Electrical Center of North Brazil, Tucuruí, Pará, Brazil, where he is currently with the HPP Tucuruí in the Electrical Maintenance Department. He has experience in the operation of hydroelectric plants, maintenance planning, equipment design/specifications, and maintenance of generators, transformers, and gas insulated substations.