Revista Eletrônica de Iniciação Científica em Computação, 2025, 23:1
ISSN: 3085-8461 • doi: 10.5753/reic.2025.6021
This work is licensed under a Creative Commons Attribution 4.0 International License.

Research Paper

Zero-Day Ransomware Family Detection Based on Printable Character Analysis and Machine Learning

Keven Kauê Gonçalves Pinto [**Federal University of Pará** | ***keven.pinto@tucurui.ufpa.br***]
Eduardo Silva Farias [**Federal University of Pará** | ***eduardo.farias@tucurui.ufpa.br***]
Davi Carvalho Moreira [**Federal University of Pará** | ***davicm@ufpa.br***]
Caio Carvalho Moreira [**Federal University of Pará** | ***caiomoreira@ufpa.br***]

Faculty of Computer Engineering, Tucuruí Campus, Federal University of Pará, Highway BR 422 km 13, s/n, Vila Permanente, Tucuruí, PA, 68464-000, Brazil.

Abstract. This study proposes a static analysis-based method for detecting zero-day ransomware families through the extraction of printable characters from Windows binary files. The method employs a soft-voting ensemble classification composed of three machine learning techniques: Adaptive Boosting (ADB), Extra-Trees (EXT), and Logistic Regression (LR). To ensure the effectiveness of the approach, we created a dataset of 2,675 binary samples (ransomware and goodware). The training set includes 1,023 samples from 25 relevant ransomware families and 1,134 goodware samples, while the test set consists of 385 samples from 15 recent ransomware families and 133 benign samples. The Detection of New Ransomware Families (DNRF) results achieved 95.88% accuracy, 90.50% precision, 100% recall, and 94.74% F-measure, with an average analysis and prediction time of 0.45 seconds. These results highlight the method's potential as an additional layer of protection for antivirus systems, particularly on devices with limited hardware resources. Our method advances the field of zero-day ransomware detection by offering a more resilient and real-time applicable solution.

Keywords: Ransomware, Static analysis, Printable characters, Zero-day detection, Scalability

Received: 12 May 2025 • **Accepted:** 09 September 2025 • **Published:** 21 September 2025

1 Introduction

Ransomware is a malicious software that encrypts a victim's data and demands a ransom for its release. Since its emergence in the late 1990s, it has become a significant cybersecurity threat, causing substantial financial and operational damage to individuals, organizations, and governments globally. The increasing frequency and sophistication of ransomware attacks have driven extensive research into detection, mitigation, and response [Beaman *et al.*, 2021; Kapoor *et al.*, 2022]. Despite advancements in detection techniques, considerable challenges persist. Given its lucrative nature, ransomware constantly evolves to evade protection mechanisms and enhance its encryption capabilities [Kolodenker *et al.*, 2017]. Numerous subfamilies, often derived from shared base code and employing similar tactics, emerge rapidly, which perpetuates the ransomware threat and poses an ongoing risk to individuals and organizations [Hassan, 2019].

The effectiveness of ransomware has led to the emergence of a lucrative market known as Ransomware-as-a-Service (RaaS), in which developers create ransomware and offer it to others for a share of the profits from successful attacks. RaaS serves as a gateway for criminals with limited programming skills, enabling them to profit from ransomware operations. Moreover, RaaS facilitates customized ransomware, allowing attackers to conduct targeted campaigns more efficiently while reducing detection risk [Meland *et al.*, 2020].

Ransomware exhibits a distinct structure from benign files. Its typical operations include gathering system information, mapping the environment to locate user files, using Application Programming Interfaces (APIs) for file encryption, and connecting to a Command-and-Control (C&C) server [Hampton *et al.*, 2018; Hull *et al.*, 2019; Kapoor *et al.*, 2022; Moreira *et al.*, 2022]. Ransomware detection primarily uses two types of analysis: dynamic and static. Each approach has distinct strengths and limitations, offering complementary information. Dynamic analysis observes software behavior during execution, usually in an isolated environment to prevent system damage. By monitoring the program's runtime behavior, the dynamic method provides a detailed understanding of its interaction with the Operating System (OS). This makes evasion tactics more challenging. However, it is generally slower and less secure due to the program execution dependency [Kok *et al.*, 2022].

In contrast, static analysis involves examining suspicious software's code and structure without executing it. This method provides researchers with insights into the software's content and architecture and enables early identification of harmful instructions. Early detection is crucial for mitigating ransomware's impact and reducing the likelihood of successful attacks. Compared to other methods, static analysis is typically faster and safer. The outcome of static analysis is the extraction of various features, such as Portable Executable (PE) header information, hash values, API calls, entropy, Control Flow Graphs (CFG), byte sequences, opcodes, and printable characters, which can be leveraged to develop malware detection systems [Ucci *et al.*, 2019; Cen *et al.*, 2024]. While its primary goal is to detect ransomware before it encrypts files or causes harm, static analysis is vulnerable to evasion techniques like obfuscation, polymorphism, encryption, and anti-disassembly methods [Oz *et al.*, 2022; Cen *et al.*, 2024].

Host-based antivirus solutions are the most common and effective security measures against known ransomware. These systems match file signatures with a database periodically updated from a central antivirus server. However, this method is inflexible against previously unknown (zero-day) attacks. Furthermore, when an antivirus analysis center identifies new ransomware, a new signature must be generated and distributed to users for protection [Moussaileb *et al.*, 2021]. Consequently, even with up-to-date antivirus software, a host remains highly vulnerable to significant damage if it is the first target of a zero-day threat.

Most zero-day attack detection approaches rely on Machine Learning (ML) and Deep Learning (DL) techniques. These methods train models to identify specific file characteristics to predict potential threats based on available samples. However, they face significant challenges, such as assuming zero-day attacks resemble previously recorded ones, selecting effective features for analysis, and obtaining high-quality training data [Moussaileb *et al.*, 2021; Guo, 2023]. Additionally, many studies use the standard Cross-validation (CV) or traditional dataset splitting, where training and test datasets include samples from the same ransomware families. This practice inadvertently allows models to "memorize" patterns from related samples, leading to misleading results and reduced generalization. Few studies address the Detection of New Ransomware Families (DNRF), a specific case where no samples from a test family are included during model training. This separation introduces a necessary barrier to mitigate overfitting, which ensures the model learns generalizable patterns rather than simply memorizing the data.

The rise of new ransomware families driven by RaaS and the limits of current methods demand real-time detection approaches that protect clients even as first targets. This requires moving beyond signature-based solutions to techniques capable of identifying threats before execution.

Motivated by the common structural basis of ransomware, this research explores static feature analysis, focusing on strings and printable characters, to identify patterns that differentiate malicious files from benign ones using ML models. Extracting printable characters from executable binaries can reveal key elements like ransom messages, API names, Dynamic-link Libraries (DLLs), and section identifiers, offering a multifaceted view of the file's structure and behavior. This string analysis can uncover indicators of malicious intent and enhance static malware detection effectiveness.

Our approach relies on observable features in executable files, particularly the frequency of printable ASCII characters. While static analysis remains important, recent studies, such as Khan *et al.* [2024], have explored using Large Language Models (LLMs) to classify malware based on human-readable content, like ransom notes. Although promising, these models face issues like hallucinations and a lack of interpretability due to their complex architectures [Zhang *et al.*, 2025]. Given these limitations, we opted for traditional ML models rather than adopting LLMs in our classification pipeline.

Our proposed technique can serve as an additional security layer within antivirus systems. It inspects each PE file at critical junctures - such as during download, upon receiving attachments, or before execution. Unlike other static analysis methods, our approach detects ransomware-specific characteristics even when the family is unknown, as demonstrated by our results. This is achieved without relying on third-party software, through a detailed examination of printable characters, which also contain information about the file's structure.

In this research, a feature dataset was created by extracting printable characters from 2675 binary samples used in our previous study [Moreira *et al.*, 2024]. The training and validation dataset included 1023 ransomware samples from 25 prominent families and 1134 goodware samples. For testing, the dataset comprised 385 ransomware samples from 15 recent families and 133 goodware samples.

The remainder of this paper is organized as follows: Section 2 reviews related works, their key findings, advantages, and limitations. Section 3 outlines the materials and methods employed in this study. Section 4 presents and discusses the experimental results. Finally, Section 5 summarizes the study's conclusions and proposes directions for future research.

2 Related Works

Recent studies in ransomware detection have shown significant advances using static, dynamic, or hybrid analysis. This often represents a promising path using behavioral and structural characteristics with ML algorithms.

Aljabri *et al.* [2024] proposes a ransomware detection model based on features extracted directly from memory. The authors concluded that memory dump analysis can be an effective detection approach. They used Random Forest (RF) model with only 16 features to achieve a detection accuracy of 97%.

Gurukala and Verma [2024] propose a hybrid approach to detect ransomware. For static analysis, they utilized the PE Parser Python library to extract Opcode features, registry references, data-defined directives, symbol frequencies, PE section features, and other miscellaneous features. For dynamic analysis, DRAKVUF sandbox was employed to monitor API calls. Their research presented an approach combining ensemble classifiers with feature selection using the Particle Swarm Optimization (PSO) algorithm. The combination of Decision Tree (DT) and K-Nearest Neighbors (KNN) classifiers achieved 98.38% accuracy.

The studies mentioned above primarily rely on dynamic or hybrid analysis approaches, which present well-known challenges like high latency due to the required behavioral analysis time. Conversely, our study focuses strictly on static analysis to facilitate real-time execution even on devices with limited hardware resources.

Zhang *et al.* [2019] employed a static analysis method based on extracting N-grams of opcodes using external tools like IDAPRO, subsequently applying a Term Term Frequency - Inverse Document Frequency (TF-IDF) filter. The authors used variable sets of N-grams $(2, 3, 4)$. Their best results were accuracies of 99.3% and 99.8% in one family using the RF classifier with $N = 3$ and 180 features; however, the authors did not provide other metrics, preventing a deeper analysis of the method's efficiency.

Zhang *et al.* [2020] started from a similar premise of using third-party programs to extract N-grams of opcodes; however, they applied a different treatment to the obtained sequences based on a mechanism called Self-Attention Convolutional Neural Network (Self-Attention-Convolutional Neural Network (SA-CNN)) on the sequences subdivided into patches. The results were then concatenated and included in a Self-Attention network for final sample classification. The main results achieved were 89.50% accuracy, 87.50% precision, 87.60% recall, and 87.30% F-measure.

Ayub and Sirai [2021] analyzed 727 active ransomware samples using static analysis based on PE file metadata. They used tools like YARA to identify cryptographic libraries and FLOSS to analyze obfuscated strings. The Local Outlier Factor (LoF) classifier yielded the best performance in detecting similarities between ransomware variants, with an accuracy of 89.96%.

Gaur *et al.* [2021] proposed a static analysis model for ransomware detection using ML and DL. For classification, they used entropy and PE file sections as well as opcode frequencies, employing tools such as Pefile and DUMPBIN. The applied models included RF, Support Vector Machine (SVM), KNN, and Neural Networks, with RF achieving the highest accuracy (99.68%). As a practical application, a command-line utility was developed to analyze executable files, display basic information, and classify them as ransomware or benign.

Ciaramella *et al.* [2023] proposed a static analysis method that converts binary code into RGB images by extracting opcodes with objdump. DL models such as LeNet, AlexNet, VGG-16, and a custom CNN were tested. The VGG-16 model achieved the best performance with 96.9% accuracy. Additionally, the Gradient-weighted Class Activation Mapping (Grad-CAM) technique was used to visualize which image regions influenced the neural network's decisions.

All the aforementioned studies employed conventional evaluation methods, such as CV or an 80/20 train-test split. These approaches include samples from the same ransomware families in training and testing sets. While often reporting strong performance metrics, they do not reflect the more challenging and practical scenario of detecting previously unseen families. The DNRF setting, which excludes any samples of a given family during training, offers a more rigorous evaluation. The DNRF setup is particularly valuable for assessing the system's ability to detect zero-day attacks—an essential feature for ensuring the robustness and reliability of modern protection mechanisms.

Other studies have employed similar DNRF testing methodologies, including [Zahoora *et al.*, 2022], [Guo, 2023], and [Cen *et al.*, 2025]. However, these studies rely on dynamic or hybrid analysis approaches, which, as previously mentioned, may not be suitable for real-time detection.

Vehabovic *et al.* [2023] proposed an ML framework for early ransomware detection using a limited set of structural features from the PE header. Their dataset included 1,240 ransomware samples from nine families and 2,000 goodware samples. Applying a DNRF-like methodology, they achieved an average accuracy of 81.04% with an RF classifier based on 15 header fields. However, the limited family diversity and small feature set likely affected the model's generalization and reduced its performance on certain families.

Moreira *et al.* [2023] employed static analysis from PE headers converted into color images using external programs, with classification through a Convolutional Neural Network (CNN) model called Xception ColSeq. They achieved weighted average results for accuracy, precision, recall, and F-measure of 93.84%, 97.50%, 89.77%, and 92.27%, respectively. However, the model's complexity may lead to increased classification time, reported as 0.66 seconds per sample.

Moreira *et al.* [2024] combined static features—header fields, imported DLLs, function calls, and section entropy—for a deeper ransomware analysis. They employed an ensemble of Logistic Regression (LR), RF, and eXtreme Gradient Boosting (XGB) models to achieve averages of 97.53% accuracy, 96.36% precision, 97.52% recall, and 96.41% F-measure. However, reliance on third-party programs increases the method's complexity, which reduces its scalability for devices with limited hardware resources.

Although previous studies show promising results in ransomware detection, more scalable static solutions for threats in the DNRF scenario offer greater application potential due to their efficiency and reliability. Table 1 presents a comparative summary of the strengths and limitations of each related work contrasted with the current study. Our work uses a static approach based exclusively on extracting printable characters, which eliminates the need for external tools or complex preprocessing methods. Unlike other methods that use hybrid or dynamic analysis, which often suffer from latency and increased operational cost, our method allows for lightweight and real-time detection, suitable for devices with limited hardware. Furthermore, in contrast to works with limited or imbalanced datasets or those with limited generalization across ransomware families, our approach emphasizes generalization through ensemble classification to improve robustness against zero-day variants. By avoiding dependencies on sandboxing, image transformation, or memory analysis, we propose a scalable and efficient solution for proactive ransomware defense on local client systems.

3 Materials and methods

This section describes the methodology, sample repositories, feature analysis, feature preprocessing, ML models, the approach for testing new ransomware families, and the performance evaluation methods.

Figure 1 presents the workflow of our proposed method. First, we extract printable characters from the PE files (step 1.1), then apply Natural Language Processing (NLP) processes using Regular Expressions (step 1.2) to create the n-gram sequences (step 2.1). Subsequently, we employ the TF-IDF method to calculate the top 1000 n-grams (step 2.2) for better sentence structure and flow. We populate the rows of the n-gram dataset based on the presence (1) or absence (0) of the most relevant n-grams for each column and map the goodware and ransomware families to identify the sample type in each row, generating the top n-gram dataset (step 2.3). This dataset is then processed using Variance Thresh-

Table 1. Strengths and limitations of related works

Works	**Strengths**	**Limitations**
[Aljabri *et al.*, 2024] and [Gurukala and Verma, 2024]	Enhances existing anti-ransomware strategies	Performs dynamic/hybrid analysis (not suitable for real-time detection)
[Zhang *et al.*, 2019], [Zhang *et al.*, 2020], [Ayub and Sirai, 2021], [Gaur *et al.*, 2021], and [Ciaramella *et al.*, 2023]	Performs static analysis (adequate for real-time detection)	Does not test the DNRF
[Cen *et al.*, 2025], [Cohen and Nissim, 2018], [Guo, 2023], [Shaukat and Ribeiro, 2018], and [Zahoora *et al.*, 2022]	Tests the (DNRF)	Performs dynamic/hybrid analysis (not suitable for real-time detection)
[Vehabovic *et al.*, 2023]	Tests the (DNRF) Performs static analysis (adequate for real-time detection)	Low detection rates in DNRF
[Moreira *et al.*, 2023]	Tests the (DNRF) Performs static analysis (adequate for real-time detection)	High classification time
[Moreira *et al.*, 2024]	Tests the (DNRF) Performs static analysis (adequate for real-time detection)	Requires external programs (decreases scalability)
This study	Enhances existing anti-ransomware strategies Performs static analysis (adequate for real-time detection) Tests the (DNRF) Good detection rates in DNRF Low classification time Does not require external programs (increases scalability)	

old (VT) 99% to reduce dimensionality, generating the processed 3, 4, and 5-gram datasets used for training (step 3.1).

Figure 2 illustrates the method for selecting datasets and models, as well as the methodology for testing new ransomware families. First, we split the training dataset into a ransomware family and an equal number of randomly selected goodware samples (step 1.1). These samples are then used to train and test each model, maximizing hyperparameters (step 1.2). We apply the results obtained with the hyperparameters in statistical techniques (Friedman and Nemenyi tests) to determine which datasets and models will be used (step 1.3). The most prominent models of each classification method are employed in the "soft voting" ensemble method (step 2.1). We perform comparative analyses for each test family and also randomly separate goodware samples to classify the samples (step 2.2).

The proposed method offers a comprehensive approach for ransomware detection by employing simple processing resources alongside hyperparameter selection techniques aimed at improving the performance of the applied ML models. The models are ultimately integrated into an ensemble method for sample classification.

3.1 Datasets

For this study, the dataset is derived from the feature extraction of 2,675 executable samples collected from a database provided by Moreira *et al.* [2024]. These samples are divided into two classes: a training set with 2,157 samples (80%), comprising 1,134 goodware samples and 1,023 ransomware samples from 25 families; and a test set with 518 samples (20%), composed of 133 goodware samples and 385 ransomware samples from 15 more recent families.

The selected ransomware samples exhibit significant variation in origin, spanning both small and large RaaS malicious development groups. These samples adopt diverse vulnerability exploitation techniques, algorithms, and encryption methods.

The samples' origin from the database is linked to five main sources. All ransomware executables were obtained from VirusShare and Hybrid-Analysis, while benign files were primarily collected from the PortableApps and Softonic databases, as well as from the Windows 10 standard system. With these samples, we created the 3, 4 and 5-gram datasets used to train the models and maximize hyperparameters.

3.2 Printable Characters

As part of our methodology, multiple character sets were systematically extracted from the samples and utilized as discriminative features for malicious executable detection.

Since byte sequence extraction yields a large volume of noisy data, it is necessary to filter out the most relevant strings for malware detection is necessary. We used the following regular expressions to obtain those strings

```
(?=[a-zA-Z]\{3,\})(?:[a-zA-Z0-9]) ,
```

where sequences with at least 3 characters are identified that begin with upper or lower case letters, but may contain letters or numbers after that, given the possibility of attackers using methods to disguise sensitive words such as "ransomware", "data", "encrypted" in combinations with similar characters such as "r4ns0mware", "d4ta", "3ncrypt3d" or several other possible types of combinations.

When analyzing strings extracted from ransomware samples using NLP, it is common to apply the N-gram tech-

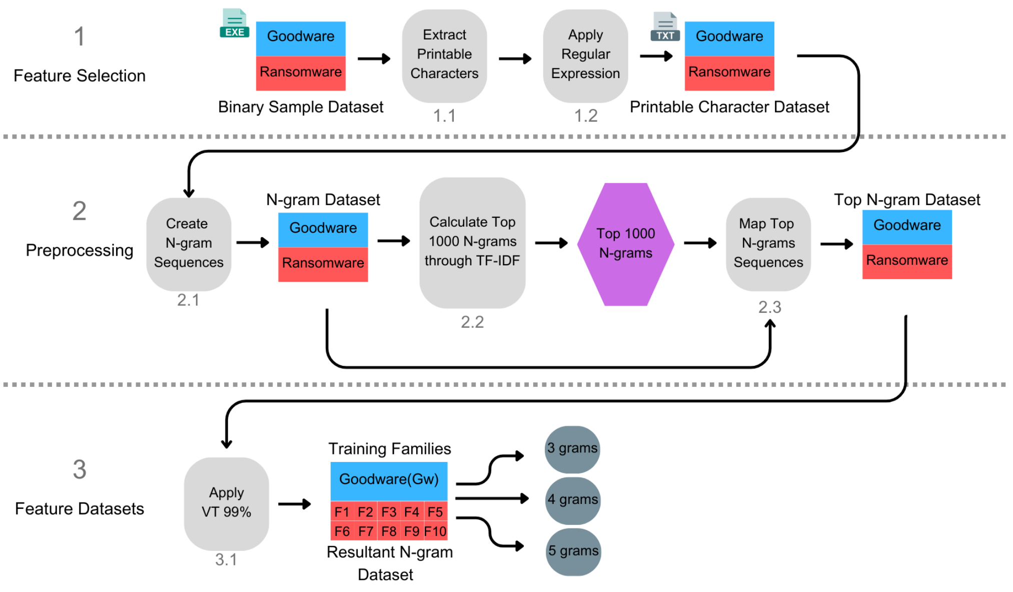

Figure 1. Workflow

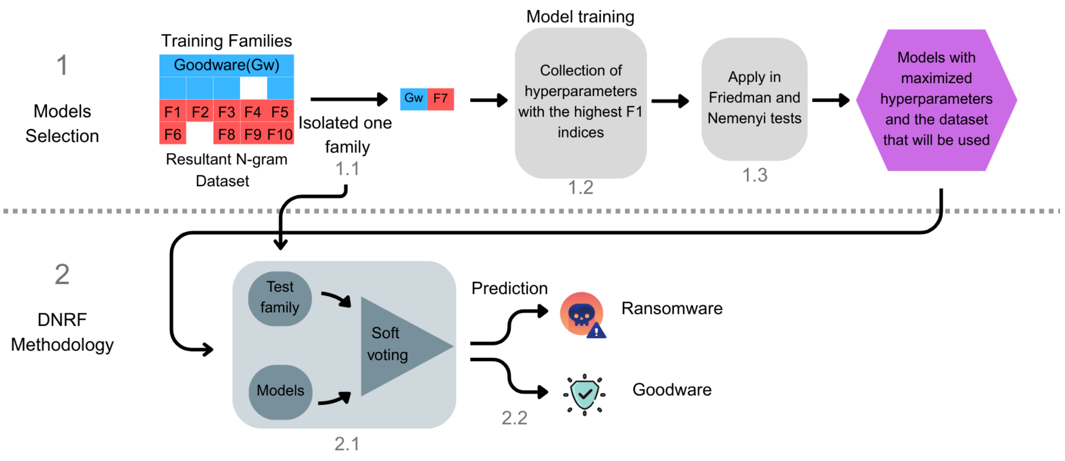

Figure 2. Model selection and joint application to test families

nique, with weights calculated using TF-IDF [Zhang *et al.*, 2019, 2020].

An N-gram model is a probabilistic language model that predicts the next item in a sequence. For this work, we chose to use N values of $\{3, 4, 5\}$, as they provide a good balance between performance and contextualization, as indicated by previous studies [Poudyal *et al.*, 2019; Zhang *et al.*, 2019].

The score obtained through TF-IDF reflects an N-gram's importance within a document. The Term Frequency (TF) indicates how many times the N-gram appears in the document, while the Inverse Term Frequency (IDF) shows how often the N-gram occurs across all documents. By multiplying TF by IDF, the TF-IDF value is obtained. Mathematically, TF is calculated as

$$TF_{(i,j)} = \frac{n_{(i,j)}}{\sum_k n_{(k,j)}}, \tag{1}$$

where $TF_{(i,j)}$ represents the frequency with which N-gram i is found in the N-gram sequence j, $n_{(i,j)}$ indicates the number of times N-gram i occurs in the N-gram sequence j, and finally, $\sum_k n_{(k,j)}$ corresponds to the total frequency of all i N-grams in the N-gram sequence j.

The IDF is calculated as

$$IDF(i) = \log_2 \left(\frac{|D|}{|\{j : i \in j \wedge j \in D\}|} \right), \tag{2}$$

where $IDF_{(i)}$ shows the frequency with which N-gram i appears across all N-gram sequences, D represents the set of all N-gram sequences, and $|j : i \in j \wedge j \in D|$ represents the number of N-gram sequences that contain N-gram i.

Thus, TF-IDF is calculated as

$$TF\text{-}ID_{(i,j)} = TF_{(i,j)} \times IDF_{(i)}, \tag{3}$$

in which $TF\text{-}IDF_{(i,j)}$ represents the value of N-gram i in the N-gram sequence j. The higher the TF-IDF value of the N-gram, the more important it is.

To extract string sequences, we used the *re* Python library to define regular expressions and Scikit-learn to construct N-gram sequences and compute TF-IDF values. Due to the large number of unique N-grams, we selected the top 1,000 most relevant N-grams for each N to ensure a better balance between performance and computational efficiency.

3.3 Data preprocessing

Preprocessing is a key step to ensure the quality of the input dataset, as well as to enhance performance and results. To handle features and eliminate those with low information content, the VT method was applied. VT is an unsupervised filtering model that removes features whose variance does not meet a predefined threshold. The variance is calculated as

$$Var_{(X)} = \frac{1}{m}\sum_{a=1}^{m}(x_a - \bar{x})^2, \qquad (4)$$

where Var(X) represents the variance of sample X, m is the sample size, x_a is the occurrence at position a, m is the sample size, and finally, $\bar{x}$ is the mean of the sample Fida *et al.* [2021].

A special case is the use of VT in binary features. In this scenario, the Bernoulli probabilistic experiment is used, where there are only two possible outcomes: success $(x = 1)$ or failure $(x = 0)$. Thus, the variance for binary features is given by

$$Var_{(X)} = p(1 - p), \qquad (5)$$

p stands for the chance of success, also called the Bernoulli probability parameter Forbes *et al.* [2010].

Table 2 summarizes the results after applying VT with a parameter p = 0.99%, which indicates the minimum variance threshold defined for this study, applied across the different datasets created from 3, 4, and 5 N-grams.

Table 2. Before and after VT

Dataset	Before	After	%Reduction
3-grams	2000	1449	27.55%
4-grams	2000	1671	16.45%
5-grams	2000	1692	15.40%

Given its role in eliminating features irrelevant to training ML models, which avoids overfitting and reduces training time, VT has become an essential tool in this study for creating datasets.

3.4 Proposed method for Detection of New Ransomware Families

The specific case analysis for detecting zero-day ransomware attacks involves the DNRF model, which is designed to handle uncatalogued samples. This capability is vital for any defense system, especially given the rapid evolution of RaaS. The emergence of new samples not yet recorded in databases poses a significant challenge for any identification system.

Figure 3 illustrates the process that starts with categorizing the sample set into two groups: goodware and ransomware families. For each ransomware family under evaluation, we isolate that family to create a test set. During this step, we include an equal number of randomly selected goodware samples to ensure a minimum threshold of 30 samples (Step 1). The remaining dataset is then reserved exclusively for model training (Step 2). After training the model, we apply it to samples from the ransomware family under evaluation (Step 3). Finally, we generate binary classification results, distinguishing between benign and malicious files (Step 4).

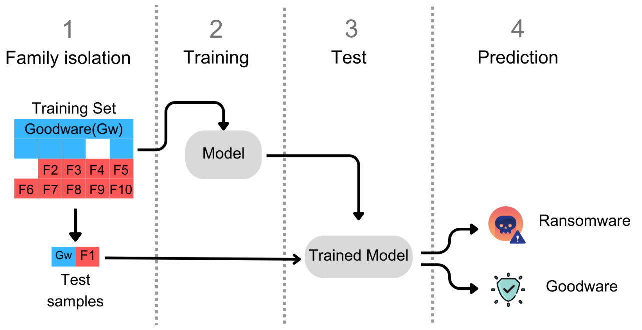

Figure 3. Detection of New Ransomware Families Methodology

3.5 Machine Learning Algorithms

To achieve the objectives of this work, it was necessary to first select the most suitable dataset among those created from N-grams and investigate ways to improve performance within the DNRF framework. The adopted approach is based on using ML methods combined with hyperparameter tuning, with an emphasis on the most prominent results. In this context, nine ML models were selected: Adaptive Boosting (ADB), DT, Extra-Trees (EXT), Gradient Boosting (GB), LR, Naive Bayes (NB), RF, Support Vector Classification (SVC), and XGB.

These models are widely known and used for binary classification problems, each employing different approaches and mechanisms, including decision trees, ensemble learning, linear regression and class clustering. The diversity of methods allows for a better exploration of dataset results, given the advantages and limitations of each ML model, providing a deeper understanding of the data.

For each test, all samples from the target ransomware family were used, along with an equal number of randomly selected goodware samples. For families with fewer than 15 samples, additional benign samples were included until a minimum of 30 samples was reached, establishing the required threshold. The remaining goodware and family samples were used for training. To optimize the hyperparameters, we performed ten iterations per ransomware family, reporting average metrics to reflect model performance in DNRF.

After identifying the best F-measure results and their corresponding hyperparameters for each ML model, we proceeded to search for the best overall results, both in the aforementioned models and in the datasets used to achieve them. Once these essential combinations were determined, they were applied to the test dataset created from the remaining 15 new ransomware families, as shown in Figure 3.

3.6 Evaluation of the performance of classification algorithms

We used well-established metrics for binary classification to evaluate each technique's results. We divide the target class into two categories: positive and negative. In this study, we define the positive class as ransomware and the negative class as goodware. Additionally, using different evaluation methods provides a more detailed understanding of each classifier's performance. These metrics serve as important performance indicators and can be interpreted as relationships studied by [Hossin and Sulaiman, 2015].

- True Positives (TP): ransomware correctly classified as ransomware;
- True Negatives (TN): goodware correctly classified as goodware;
- False Positives (FP): number of goodware samples incorrectly classified as ransomware; and
- False Negatives (FN): number of ransomware samples incorrectly classified as goodware.

The most basic metric used in our study is Accuracy (ACC), which measures the rate of correct predictions out of the total number of instances analyzed, mathematically defined as

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}. \tag{6}$$

Precision (PREC) indicates a classifier's success rate in not treating a positive sample as negative, where high precision signifies few false positives, given by

$$PREC = \frac{TP}{TP + FP}. \tag{7}$$

Recall (REC), on the other hand, assesses the success in finding all positive samples correctly classified; a high value of this metric indicates few false negatives, mathematically described as

$$REC = \frac{TP}{TP + FN}. \tag{8}$$

Although the previously mentioned metrics are highly important for analyzing the behavior of the classes, in this study, the main target metric is the F1-score (F1), which combines the results of precision and recall into a harmonic mean, also known as the balanced F-measure, given by

$$F1 = 2 \times \left(\frac{Precision \times Recall}{Precision + Recall} \right). \tag{9}$$

Strategies focused on optimizing the F1 score were employed to maximize results across the datasets and avoid overfitting and underfitting. Table 3 describes the results obtained from the hyperparameters used during training, with the highest values highlighted in bold.

For a complete assessment of an ML model's performance, it is essential to jointly analyze the results of various metrics, as each one offers a distinct perspective on the model's effectiveness [Powers, 2020].

Additionally, for a broader approach to the performances of each model, it is necessary to employ statistical tests, given the probabilistic nature inherent to learning models. In this sense, two non-parametric statistical techniques were used for this analysis of ML algorithms in their application to DNRF: Friedman and Nemenyi. The Friedman test is guided by a null hypothesis H_0, which states that the results are statistically equivalent, and an alternative hypothesis H_1, which states that they are different. In this article, a confidence interval (CI) of 95% demonstrates that the relationship is significant. If the observed value (p-value) is less than 0.05, the null hypothesis should be rejected [Demšar, 2006].

In case of rejection, the post-hoc Nemenyi test is performed, which is used to detect significant differences between the datasets and the learning models. For this, the difference between the average ranks of the two analyzed groups must differ by at least the Critical Distance (CD), given by:

$$CD = q_\alpha \sqrt{\frac{c(c+1)}{6M}}, \tag{10}$$

where q_α represents the critical value outlined by the studies addressed by [Demšar, 2006] following the Studentized range statistic divided by $\sqrt{2}$, c is the number of groups, and M is the number of metrics analyzed in these groups.

4 Experiments and results

This section presents the results obtained from the nine ML models and identifies the dataset that achieves the most prominent results. Subsequently, it presents the selection process of the models used in the ensemble method, which applies to the remaining 15 families reserved exclusively for testing.

The experiments were conducted in the Google Colab environment, equipped with an Intel® Xeon® dual-core processor @2.20GHz, 12 GB of RAM, and 225 GB of disk storage.

Table 3 summarizes the F1 values obtained for each ML technique applied to datasets composed of 3, 4, and 5-grams, with the best-performing results highlighted in bold. The results demonstrate promising rates, particularly for the 3- and 4-gram sets, whose classifiers achieve values above 94% in the vast majority, DT and SVC in the 3-gram set notably achieved F1 rates above 97%. For the 4-gram set, the classifiers SVC and EXT showed good results, also around 97%. Additionally, the 5-gram set still presents a good F1 rate despite being inferior in most algorithms except for NB.

Table 3. Performance comparison between nine ML models in different datasets

Models	3-grams	4-grams	5-grams
ADB	95.51 ± 0.41	**95.77 ± 0.31**	94.73 ± 0.42
DT	**97.07 ± 0.24**	95.86 ± 0.29	91.89 ± 0.85
EXT	96.47 ± 0.32	**96.69 ± 0.28**	91.85 ± 0.32
GB	**96.70 ± 0.29**	92.06 ± 0.40	92.02 ± 0.55
LR	94.66 ± 0.23	**94.77 ± 0.23**	94.42 ± 0.28
NB	92.46 ± 0.31	92.25 ± 0.49	**94.99 ± 0.33**
RF	96.28 ± 0.43	**96.35 ± 0.36**	94.15 ± 0.24
SVC	**97.06 ± 0.25**	97.02 ± 0.24	96.91 ± 0.27
XGB	**95.04 ± 0.47**	94.16 ± 0.55	93.93 ± 0.19

The results were statistically analyzed using the Friedman and Nemenyi tests. Table 4 presents the organization and results of these analyses. The Friedman test rejects the null hypothesis H_0 with a p-value of 1.3×10^{-2}, which indicates statistically significant differences between the performances of the evaluated models. In the Nemenyi post-hoc test, the CD between the mean ranks is 1.12. The results indicate a statistically significant distinction between the 3-gram (1.44) and 5-gram (2.77) datasets, while no significant difference is observed between the 3-gram (1.44) and 4-gram (1.75) datasets.

Table 4. Friedman and Nemenyi test results

Dataset	Average Rank
3-grams	1.55
4-grams	1.66
5-grams	2.77

Given this scenario, and considering the need to balance predictive performance and computational efficiency, the 4-gram dataset is selected for subsequent experiments. To choose the model with the best computational performance, we consider the average training time. The 4-gram dataset took 0.6 seconds, while the 3-gram dataset required an average of 2.12 seconds to converge, demonstrating a higher processing time.

Table 5 presents the performance comparison between the metrics ACC, PREC, REC, and F1 for each classifier on the 4-gram dataset. When applying statistical tests, the null hypothesis H_0 of the Friedman test was rejected with an observed p-value of 2.7×10^{-2}, suggesting statistically significant differences among the models.

Table 5. Comparison of performances between the metrics of each model in training

Models	Accuracy	Precision	Recall	F1
ADB	95.96 ± 0.32	95.33 ± 0.60	96.61 ± 0.03	95.77 ± 0.31
DT	96.17 ± 0.29	96.67 ± 0.54	95.70 ± 0.06	95.86 ± 0.29
EXT	96.77 ± 0.29	96.77 ± 0.29	96.91 ± 0.05	96.69 ± 0.28
GB	94.70 ± 0.31	98.00 ± 0.54	90.93 ± 0.31	92.06 ± 0.40
LR	96.05 ± 0.24	97.39 ± 0.56	94.41 ± 0.02	94.77 ± 0.23
NB	93.45 ± 0.45	94.07 ± 0.73	92.32 ± 0.21	92.25 ± 0.49
RF	96.51 ± 0.35	96.62 ± 0.57	96.42 ± 0.16	96.35 ± 0.36
SVC	97.11 ± 0.25	97.00 ± 0.49	97.24 ± 0.00	97.02 ± 0.24
XGB	95.49 ± 0.4	97.84 ± 0.51	93.09 ± 0.36	94.16 ± 0.55

We applied the Nemenyi post-hoc test with the average classifications in Table 6 to assess significant differences between models. By calculating the CD (5.36), we found statistically relevant differences between SVC (1.75) and EXT (2.75) when compared to NB (8.50).

Table 6. Average Rank Result for each model

Models	Average rank
SVC	1.75
EXT	2.75
RF	4.25
DT	4.75
LR	5.00
ADB	5.50
XGB	5.75
GB	6.75
NB	8.50

The NB model presented the weakest statistical performance among the evaluated classifiers and was discarded. Since no statistically significant differences were found among the remaining models, we implemented an ensemble learning approach based on Soft Voting. In this method, each model contributes its predictive probabilities, and the final decision is made based on the class with the highest aggregated probability. This approach ensures a balance among the different learning strategies. To compose the Soft Voting ensemble, we selected the top-ranking models based on their mean rank while also ensuring diversity in their learning approaches. Initially, the highest-ranking models were SVC (1.75), EXT (2.75), LR (5.00), and ADB (5.50).

We analyzed the classifiers individually before combining them to form the classification model and evaluating their performance in the DNRF methodology. Table 7 shows a particular case of the SVC algorithm. Although the model obtained the best overall classification, its specific performance when applied to the DNRF method was unsatisfactory. This indicates that, despite its superior overall performance, it does not adapt well to this specific approach, which would compromise the ensemble's effectiveness. Thus, we chose to exclude it from the Soft Voting ensemble.

Table 7. Result(%) of the SVC model

Model	Accuracy	Precision	Recall	F1
SVC	65.90 ± 2.98	56.31 ± 1.94	100.00 ± 0.00	70.96 ± 1.62

Based on model behavior, statistical equivalence, and internal characteristics, the ADB, EXT, and LR techniques were selected to compose the weighted voting on the 4-gram dataset. To avoid data leakage, training was restricted to the 25 known ransomware families, while the remaining 15 were reserved exclusively for testing, ensuring a realistic DNRF scenario.

Table 8 presents the weighted average results for the proposed model and the highest-scoring classifiers, whose performance is significantly improved compared to the individual algorithms that compose it, highlighting the benefit of leveraging each model's strengths. These results reinforce the method's potential for detecting unknown ransomware and its adaptability to evolving threats.

Table 8. Result(%) of the 3 best models (EXT, LR and ADB) and the proposed model

Models	Accuracy	Precision	Recall	F1
EXT	95.09 ± 1.83	89.18 ± 3.31	99.62 ± 0.57	93.74 ± 2.18
LR	95.58 ± 1.44	89.64 ± 3.00	100.00 ± 0.00	94.19 ± 1.76
ADB	94.54 ± 1.66	88.03 ± 3.35	100.00 ± 0.00	93.29 ± 2.03
Prop.	$\mathbf{95.88 \pm 1.59}$	$\mathbf{90.50 \pm 3.39}$	$\mathbf{100.00 \pm 0.00}$	$\mathbf{94.74 \pm 1.99}$

Figure 4 illustrates the confusion matrix for the proposed model.The matrix was created from the 518 test samples. The results obtained reflect the model's performance in analyzing all samples, without division by families and in a single iteration. The elements on the main diagonal (111 and 385) represent the high number of true positives for each class, which demonstrates the model's high accuracy in both ransomware detection and goodware identification. The small number of off-diagonal elements (22 and 0) indicates that the classifier maintains low misclassification rates, further reinforcing its robustness and reliability in real-world scenarios.

Table 9 presents the results obtained with the proposed method in the DNRF context for the test families. As

Table 9. Individual results for each of the 15 test families with soft voting

Family	Samples (R-G)	Accuracy	Precision	Recall	F1
Avoslocker	50 - 50	96.70 ± 1.31	93.91 ± 2.34	100.00 ± 0.00	96.83 ± 1.24
Bianlian	11 - 19	96.33 ± 2.85	91.68 ± 6.17	100.00 ± 0.00	95.46 ± 3.44
Blackbasta	32 - 32	96.88 ± 1.18	94.20 ± 2.06	100.00 ± 0.00	96.99 ± 1.10
Blackbyte	13 - 17	95.33 ± 2.01	90.55 ± 3.75	100.00 ± 0.00	94.97 ± 2.08
Blackcat	50 - 50	96.80 ± 1.06	94.05 ± 1.84	100.00 ± 0.00	96.92 ± 0.98
Bluesky	34 - 34	96.47 ± 1.66	93.56 ± 2.89	100.00 ± 0.00	96.63 ± 1.54
Clop	46 - 46	95.54 ± 0.93	91.86 ± 1.58	100.00 ± 0.00	95.75 ± 0.86
Hive	50 - 50	96.10 ± 0.98	92.82 ± 1.72	100.00 ± 0.00	96.26 ± 0.92
Holyghost	4 - 26	94.00 ± 2.71	71.38 ± 10.09	100.00 ± 0.00	82.61 ± 6.74
Karma	13 - 17	94.33 ± 2.76	88.91 ± 4.85	100.00 ± 0.00	94.01 ± 2.76
Lorenz	16 - 16	95.94 ± 2.12	92.73 ± 3.62	100.00 ± 0.00	96.17 ± 1.95
Maui	3 - 27	93.67 ± 3.27	65.57 ± 12.79	100.00 ± 0.00	77.95 ± 9.29
Nightsky	14 - 16	95.00 ± 2.32	90.64 ± 4.00	100.00 ± 0.00	95.01 ± 2.22
Playcrypt	43 - 43	95.93 ± 1.37	92.58 ± 2.35	100.00 ± 0.00	96.12 ± 1.27
Quantum	6 - 24	93.33 ± 2.75	76.55 ± 8.57	100.00 ± 0.00	86.26 ± 5.30
Weighted Avg.		95.88 ± 1.59	90.50 ± 3.39	100.00 ± 0.00	94.74 ± 1.99

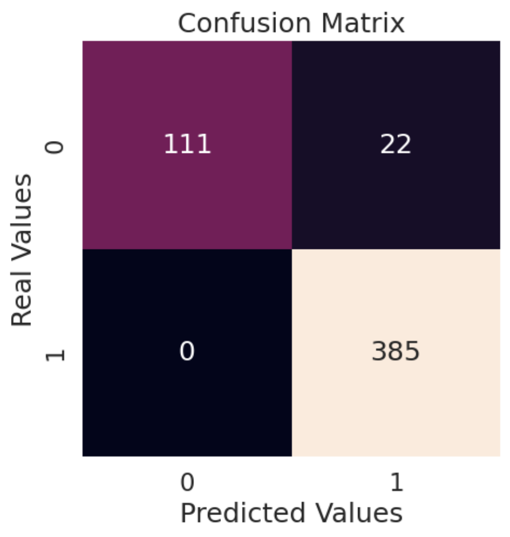

Figure 4. Confusion Matrix

described, the overall weighted F1 score reaches 94.74%, which illustrates the proposed model's efficiency. Furthermore, 12 of the 15 families achieve values equal to or greater than 95% for the same metric.

The recall metric also stands out significantly, reaching a score of 100% for all families. This result highlights the model's ability to detect ransomware accurately and reduces the occurrence of FN, which represents the most critical classification errors.

On the other hand, despite excellent recall rates, the distinction between malicious and benign samples was unsatisfactory for three families: Holyghost, Maui, and Quantum, whose precision scores reached at most 76.55%.

The low precision performance for these families can be attributed to the scarcity of ransomware samples, which necessitated the inclusion of benign samples to reach the minimum threshold of 30 samples for testing. This increased the likelihood of benign samples being misclassified as ransomware (FP), consequently impacting precision for those families.

Additionally, execution times were analyzed for feature extraction and prediction on the 518 test samples using the already trained model. These measurements are essential to simulate a real-world scenario of analyzing a single sample submitted to the model. Feature extraction took an average of 4.52×10^{-1} seconds, while prediction time is 1.8×10^{-3} seconds, resulting in a total execution time of 4.52×10^{-1} seconds. This result indicates the advantages of the proposed model, even on devices with limited hardware resources. Another noteworthy point is its compact size, occupying only 1.3 MB.

After these results, we compare our research with previous works by Moreira *et al.* [2023] and Moreira *et al.* [2024] under a similar execution environment. Table 10 presents the methods, required tools, F1-scores, and execution times of the referenced studies.

Table 10. Comparison among [Moreira *et al.*, 2023], [Moreira *et al.*, 2024], and this study under similar execution conditions.

Works	Method	Required Tools	F1-score	Exec. Time
Moreira *et al.* [2023]	Xception CNN	-	92.27%	2.87s
Moreira *et al.* [2024]	Ensemble: LR, RF, XGB	pefile, Detect It Easy	96.41%	0.37s
This study	Ensemble: EXT, LR, ADB	-	94.74%	0.45s

Moreira *et al.* [2023] employed robust computational settings to enhance the performance of their methodology, as CNN-based models require intensive operations to analyze images generated from executables, which slows down classification. Originally, their experiments were conducted using Tensor Processing Units (TPU), a resource not typically available on conventional user devices. To ensure a fair comparison, we reproduced their method using hardware equivalent to that used in our study. Under these conditions, their approach reached an average preprocessing and prediction time of 2.87 seconds and achieved similar results in terms of F1-score. In contrast, our method predicts each sample in 0.45 seconds, resulting in a performance gain of approximately 85%. While Moreira *et al.* [2023] presents an effective solution, their approach remains less suitable for resource-constrained environments. Our method, by contrast, offers faster processing and better adaptability to low-power platforms.

While Moreira *et al.* [2024] achieves a higher F1-score with reduced execution time, it requires auxiliary tools such as pefile for extracting header fields, DLLs, and function calls, as well as external software like Detect It Easy for section entropy analysis, which increases its implementation complexity. In contrast, both Moreira *et al.* [2023] and our study operate without such dependencies. This absence of additional tools is a key advantage of our approach, as it greatly simplifies its implementation across diverse environments, including Linux, Android, and IoT platforms.

Our work provides an effective solution for scenarios with limited hardware, eliminating the need for external programs. These attributes, combined with easy resource extraction, make the proposed model suitable for large-scale applications, which enhances its potential as an additional layer of protection on devices.

5 Conclusion

This research contributes by developing a static analysis model capable of effectively detecting previously unseen ransomware samples. The proposed model integrates printable strings extracted from executable files with a weighted voting ensemble classification method, composed of the ADB, EXT, and LR algorithms. This approach maintains a compact size and low execution overhead, making it suitable for real-time detection and large-scale deployment, even on devices with limited hardware resources. As a result, it can function as an additional protection layer for systems by analyzing downloaded and received files before they are executed within the operating system.

In the DNRF scenario, the proposed method shows minor performance degradation, particularly in identifying benign software, which highlights the complexity of the testing environment. Nevertheless, the model achieves perfect recall across all previously unseen families, demonstrating high efficacy in ransomware classification and confirming its robustness in adapting to the continuous evolution of such threats.

The methodology applied in this study opens new avenues for future research, as its simplistic yet effective approach can be extended to develop solutions across various operational and industrial domains. Therefore, it is undeniable that ransomware's propagative and adaptive capabilities require a continuous search for improved cybersecurity and the development of anti-malware tools capable of combating these evolving threats. Indeed, a string-based approach presents certain limitations, such as obfuscation and encryption methods for sensitive ransom note strings like ”ransomware,” ”encrypted,” and ”bitcoin,” during runtime. These obfuscation methods can impact what is extracted from each executable file, limiting model performance. This also paves the way for future studies focused on the impact of these strings on potential adversarial techniques.

Finally, the experimental results underscore the predictive capability of the developed model in identifying new ransomware families, thus supporting the creation of reliable and robust solutions. This new line of investigation shows promise for enhancing the efficiency and resilience of DNRF approaches in broader, more complex scenarios.

Declarations

Funding

This research was funded by the Institutional Scientific Initiation Scholarship Program (PIBIC) and by the Support Program for Doctoral Researchers (PRODOUTOR) of the Federal University of Pará (UFPA), Brazil.

Authors' Contributions

Keven Pinto: Writing – original draft, preparation, creation of the work and software methodology development. Eduardo Farias: Writing – review & editing, software development, manuscript writing and validation. Davi Moreira: Writing – review & editing, manuscript review. Caio Moreira: Writing – review & editing, methodology conception, research supervision, and manuscript review. Keven Pinto and Eduardo Farias are the manuscript's main contributors and writers. All authors read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests.

Availability of data and materials

The datasets generated and analyzed during the current study are available in Mendeley Data at the following link: https://data.mendeley.com/datasets/ghpy6kdhx5.

References

Aljabri, M., Alhaidari, F., Albuainain, A., Alrashidi, S., Alansari, J., Alqahtani, W., and Alshaya, J. (2024). Ransomware detection based on machine learning using memory features. *Egyptian Informatics Journal*, 25:100445. DOI: 10.1016/j.eij.2024.100445.

Ayub, M. A. and Sirai, A. (2021). Similarity analysis of ransomware based on portable executable (pe) file metadata. In *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–6. DOI: 10.1109/SSCI50451.2021.9660019.

Beaman, C., Barkworth, A., Akande, T. D., Hakak, S., and Khan, M. K. (2021). Ransomware: Recent advances, analysis, challenges and future research directions. *Computers & Security*, 111:102490. DOI: 10.1016/j.cose.2021.102490.

Cen, M., Jiang, F., and Doss, R. (2025). Ransoguard: A rnn-based framework leveraging pre-attack sensitive apis for early ransomware detection. *Computers & Security*, 150:104293. DOI: 10.1016/j.cose.2024.104293.

Cen, M., Jiang, F., Qin, X., Jiang, Q., and Doss, R. (2024). Ransomware early detection: A survey. *Computer Networks*, 239:110138. DOI: 10.1016/j.comnet.2023.110138.

Ciaramella, G., Iadarola, G., Martinelli, F., Mercaldo, F., and Santone, A. (2023). Explainable ransomware detection with deep learning techniques. *Journal of Computer Virology and Hacking Techniques*, :1–14. DOI: 10.1007/s11416-023-00501-1.

Cohen, A. and Nissim, N. (2018). Trusted detection of ransomware in a private cloud using machine learning methods leveraging meta-features from volatile memory. *Expert Systems with Applications*, 102:158–178. DOI: 10.1016/j.eswa.2018.02.039.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine learning research*, 7:1–30. Available at: `https://dl.acm.org/doi/10.5555/1248547.1248548`.

Fida, M. A. F. A., Ahmad, T., and Ntahobari, M. (2021). Variance threshold as early screening to boruta feature selection for intrusion detection system. In *2021 13th International Conference on Information & Communication Technology and System (ICTS)*, pages 46–50. IEEE. DOI: 10.1109/ICTS52701.2021.9608852.

Forbes, C., Evans, M., Hastings, N., and Peacock, B. (2010). *Statistical Distributions*. John Wiley & Sons, Ltd, Hoboken, New Jersey, 4 edition.

Gaur, K., Kumar, N., Handa, A., and Shukla, S. K. (2021). Static ransomware analysis using machine learning and deep learning models. In Anbar, M., Abdullah, N., and Manickam, S., editors, *Advances in Cyber Security*, pages 450–467, Singapore. Springer Singapore. DOI: 10.1007/978-981-33-6835-4_30.

Guo, Y. (2023). A review of machine learning-based zero-day attack detection: Challenges and future directions. *Computer Communications*, 198:175–185. DOI: 10.1016/j.comcom.2022.11.001.

Gurukala, N. K. Y. and Verma, D. K. (2024). Feature selection using particle swarm optimization and ensemble-based machine learning models for ransomware detection. *SN Computer Science*, 5:1093. DOI: 10.1007/s42979-024-03454-4.

Hampton, N., Baig, Z., and Zeadally, S. (2018). Ransomware behavioural analysis on windows platforms. *Journal of Information Security and Applications*, 40:44–51. DOI: 10.1016/j.jisa.2018.02.008.

Hassan, N. A. (2019). *Ransomware Families*. Apress, Berkeley, CA.

Hossin, M. and Sulaiman, M. N. (2015). A review on evaluation metrics for data classification evaluations. *International journal of data mining & knowledge management process*, 5(2):1. DOI: 10.5121/ijdkp.2015.5201.

Hull, G., John, H., and Arief, B. (2019). Ransomware deployment methods and analysis: views from a predictive model and human responses. *Crime Science*, 8:2. DOI: 10.1186/s40163-019-0097-9.

Kapoor, A., Gupta, A., Gupta, R., Tanwar, S., Sharma, G., and Davidson, I. E. (2022). Ransomware detection, avoidance, and mitigation scheme: A review and future directions. *Sustainability*, 14(1):8. DOI: 10.3390/su14010008.

Khan, M. A.-Z., Al-Karaki, J., and Omar, M. (2024). Llms for malware detection: Review, framework design, and countermeasure approaches. *Framework Design, and Countermeasure Approaches*. DOI: 10.2139/ssrn.4995252.

Kok, S., Abdullah, A., and Jhanjhi, N. (2022). Early detection of crypto-ransomware using pre-encryption detection algorithm. *Journal of King Saud University - Computer and Information Sciences*, 34(5):1984–1999. DOI: 10.1016/j.jksuci.2020.06.012.

Kolodenker, E., Koch, W., Stringhini, G., and Egele, M. (2017). Paybreak: Defense against cryptographic ransomware. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ASIA CCS '17, page 599–611, New York, NY, USA. Association for Computing Machinery. DOI: 10.1145/3052973.3053035.

Meland, P. H., Bayoumy, Y. F. F., and Sindre, G. (2020). The ransomware-as-a-service economy within the darknet. *Computers & Security*, 92:101762. DOI: 10.1016/j.cose.2020.101762.

Moreira, C., Sales, Jr., C., and Moreira, D. (2022). Understanding ransomware actions through behavioral feature analysis. *Journal of Communication and Information Systems*, 37(1):61–76. DOI: 10.14209/jcis.2022.7.

Moreira, C. C., Moreira, D. C., and de S. de Sales Jr., C. (2023). Improving ransomware detection based on portable executable header using xception convolutional neural network. *Computers & Security*, 130:103265. DOI: 10.1016/j.cose.2023.103265.

Moreira, C. C., Moreira, D. C., and Sales, C. (2024). A comprehensive analysis combining structural features for detection of new ransomware families. *Journal of Information Security and Applications*, 81:103716. DOI: 10.1016/j.jisa.2024.103716.

Moussaileb, R., Cuppens, N., Lanet, J.-L., and Bouder, H. L. (2021). A survey on windows-based ransomware taxonomy and detection mechanisms. *ACM Comput. Surv.*, 54(6):117. DOI: 10.1145/3453153.

Oz, H., Aris, A., Levi, A., and Uluagac, A. S. (2022). A survey on ransomware: Evolution, taxonomy, and defense solutions. *ACM Comput. Surv.*, 54(11s). DOI: 10.1145/3514229.

Poudyal, S., Dasgupta, D., Akhtar, Z., and Gupta, K. (2019). A multi-level ransomware detection framework using natural language processing and machine learning. In *14th International Conference on Malicious and Unwanted Software (MALCON)*. Available at: `https://scholar.google.com/scholar?cluster=15597576849736198O9`.

Powers, D. M. (2020). Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *arXiv preprint arXiv:2010.16061*. DOI: 10.48550/arXiv.2010.16061.

Shaukat, S. K. and Ribeiro, V. J. (2018). Ransomwall: A layered defense system against cryptographic ransomware attacks using machine learning. In *2018 10th International Conference on Communication Systems & Networks (COMSNETS)*, pages 356–363. DOI: 10.1109/COMSNETS.2018.8328219.

Ucci, D., Aniello, L., and Baldoni, R. (2019). Survey of machine learning techniques for malware analysis. *Computers & Security*, 81:123–147. DOI: 10.1016/j.cose.2018.11.001.

Vehabovic, A., Zanddizari, H., Ghani, N., Shaikh, F., Bou-Harb, E., Pour, M. S., and Crichigno, J. (2023). Data-centric machine learning approach for early ransomware detection and attribution. In *NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium*, pages

1–6. DOI: 10.1109/NOMS56928.2023.10154378.

Zahoora, U., Rajarajan, M., Pan, Z., and Khan, A. (2022). Zero-day ransomware attack detection using deep contractive autoencoder and voting based ensemble classifier. *Applied Intelligence*, 52(12):13941–13960. DOI: 10.1007/s10489-022-03244-6.

Zhang, B., Xiao, W., Xiao, X., Sangaiah, A. K., Zhang, W., and Zhang, J. (2020). Ransomware classification using patch-based cnn and self-attention network on embedded n-grams of opcodes. *Future Generation Computer Systems*, 110:708–720. DOI: 10.1016/j.future.2019.09.025.

Zhang, H., Xiao, X., Mercaldo, F., Ni, S., Martinelli, F., and Sangaiah, A. K. (2019). Classification of ransomware families with machine learning based on n-gram of opcodes. *Future Generation Computer Systems*, 90:211–221. DOI: 10.1016/j.future.2018.07.052.

Zhang, Z., Wang, C., Wang, Y., Shi, E., Ma, Y., Zhong, W., Chen, J., Mao, M., and Zheng, Z. (2025). Llm hallucinations in practical code generation: Phenomena, mechanism, and mitigation. *Proceedings of the ACM on Software Engineering*, 2(ISSTA):481–503. DOI: 10.1145/3728894.