Enhancing Ransomware Detection: A Windows API Min Max Relevance Refinement Approach

Qian Kang * and Yuanyuan Gu

Posted Date: 16 November 2023

doi: 10.20944/preprints202311.1004.v1

Keywords: ransomware; detection; dynamic analysis; feature selection; machine learning

Article

Enhancing Ransomware Detection: A Windows API Min Max Relevance Refinement Approach

Qian Kang [1,†]* and Yuanyuan Gu [1,†]*

[1] Xinshiji Cyber Academy
* Correspondence: xinshiji_academy_kang@outlook.com
† Current address: Xinshiji Cyber Academy, Shanghai, China. 200000

Abstract: Ransomware constitutes a distinctive category of pernicious software that sequesters a user's digital assets by encryption, holding them hostage until a sum is extorted from the victim. These incursions have escalated to become among the most prevalent and significant threats confronting both individuals and corporate entities. In combatting this virulent program, dynamic analysis has been established as the favored detection modality. Such analyses typically hinge on Windows API calls, the conduits through which programs requisition services from the operating system. Yet, the superfluous and unrelated Windows API calls interjected by adversaries into the execution stream of suspect binaries precipitate an excessively noisy behavioral sequence, which impairs the performance of counter-ransomware mechanisms. The research outlined herein introduces a novel non-signature-based detection paradigm that harnesses efficacious Windows API call sequences through supervised machine learning strategies. An innovative Enhanced Min Max (EmRmR) filter technique is proposed, aiming to purge noisy features and isolate the most indicative feature subset that encapsulates the ransomware's true behavior. The EmRmR method, diverging from the traditional Min Max approach, circumvents the superfluous calculations that are a hallmark of the conventional algorithms, thereby necessitating a reduced number of evaluations. Additionally, a refinement procedure has been integrated to contract the program's call trace volume by discarding those Windows API calls lacking a robust correlation with ransomware's pivotal behavior. Subsequent to rigorous experimental analyses and juxtaposition with extant behavior-based detection methodologies, the proposed strategy has demonstrated its efficacy in differentiating ransomware behavior, delivering high detection precision alongside a diminution in false-positive occurrences.

Keywords: ransomware, detection, dynamic analysis, feature selection, machine learning

1. Introduction

Ransomware has emerged as a particularly pernicious form of attack, with its capability to encrypt the digital assets of its victims, rendering them inaccessible [1,2]. This form of cyber attack represents a direct challenge to the core principles of digital security, often referred to as the CIA Triad, which encompasses the confidentiality, integrity, and availability of information [3,4]. The disruptive impact of ransomware has its roots in the latter part of the 20th century, most notably with the emergence of the AIDS Trojan towards the end of the 1980s [5,6]. This early form of ransomware was notorious for its method of operation, which involved the encryption of file identifiers and the masking of directories on the infected systems' primary storage [6,7]. This initial foray into the encryption of user data set the stage for the sophisticated ransomware attacks that are prevalent today [1,8]. The contemporary execution strategy of ransomware is typically characterized by a deceptive process that entices unsuspecting users into downloading a seemingly benign payload that, in reality, harbors the ransomware [9,10]. Once this malicious payload is activated, it proceeds to encrypt vital documents and files across various storage mediums, including local hard drives, removable media, and even networked storage systems shared among multiple users [11]. In the aftermath of this encryption, the perpetrators behind the

Citation: Lastname, F.; Lastname, F.; Lastname, F. Title. *Journal Not Specified* **2023**, *1*, 0. https://doi.org/

Received:
Revised:
Accepted:
Published:

ransomware demand financial compensation, often in the form of cryptocurrencies, for the provision of a decryption key necessary to revert the files back to their original, usable state [6,12]. The demand for a ransom, a hallmark of this cyber extortion, is not merely a monetary burden on the victims but also signifies a severe breach of the security and privacy of the affected parties [1,13]. As the digital sphere continues to evolve, the methods employed by ransomware have become increasingly sophisticated, leveraging various attack vectors to infiltrate and compromise systems [1,14]. This relentless progression of ransomware attacks requires a continuous and rigorous approach to cybersecurity, particularly in the development of dynamic and resilient strategies to counteract these threats [1,15,16].

While the digital security industry has made significant progress, the adeptness at identifying and mitigating the threats posed by ransomware, particularly the emerging strains, continues to present a substantial obstacle [17]. The age-old technique of static analysis — predicated on the detection of binary signatures without the need to run the questionable software — is now frequently outmaneuvered by the elusive tactics of ransomware developers, such as the application of obfuscation methods to cloak malicious code [9,10]. In contrast, dynamic analysis has risen to prominence, offering a more robust avenue for probing the real-time conduct of ransomware within a secure setting [11,18]. This method hinges on the meticulous observation of Windows API calls, which are the fundamental mechanisms through which ransomware interacts with and solicits services from the operating system to execute its nefarious objectives [19,20]. The sheer volume and complexity of the data produced by Windows API call logs have been known to hinder the efficacy of this analytical approach [13,21]. The dense and disordered nature of these traces often leads to a surge in computational overhead and a consequent depletion in the ability to accurately pinpoint and characterize malevolent activities [14,22]. This increase in resource consumption coupled with the challenge of accurately discerning the malicious from the benign has driven researchers and cybersecurity professionals to seek out more advanced and nuanced methods of detection and analysis [15,23,24]. As ransomware evolves, so too must the tools and techniques employed to detect it, underscoring the need for continuous innovation in the realm of cybersecurity [25,26]. The Windows API call analysis, while being a critical element in the behavioral representation of ransomware, demands a nuanced approach that can effectively distill and interpret the vast datasets generated during a ransomware attack [27,28]. These calls, when analyzed with precision, have the potential to reveal the distinctive patterns and tactics employed by ransomware, yet the task is akin to finding a needle in a haystack given the extensive and cluttered nature of the information captured [6,29]. Thus, the ongoing struggle within the cybersecurity industry is not only to enhance the capability to detect ransomware but also to refine the processes involved in such detection to ensure that they are both efficient and accurate [30,31].

The central objective of this inquiry is to forge an advanced methodology for the selection and detection of ransomware-specific features, employing a refinement process grounded in the sequence of Windows API calls in conjunction with a dynamic programming technique [9,32] (Figure 1). This methodical approach aspires to curtail the number of evaluations necessary to develop an optimal set of features, thereby augmenting the efficiency of the detection mechanism [17,18]. An examination contrasting the proposed Enhanced Min Max Relevance (EmRmR) technique with traditional methodologies is anticipated to demonstrate the enhanced computational speed and precision of evaluation that EmRmR offers [33,34]. A suite of machine learning classifiers, which includes but is not limited to Decision Trees, K-Nearest Neighbors, Logistic Regression, Random Forests, and Support Vector Machines, has been enlisted to gauge the effectiveness of the meticulously refined feature sets [19,20,35]. The ultimate aim is the establishment of a formidable and resilient framework capable of the accurate detection of ransomware intrusions [36,37]. This study has analyzed the sequence of Windows API calls, which are instrumental in the operation of ransomware, to refine the features that are most indicative of such malevolent software [38]. By streamlining these features, the study seeks to eradicate extraneous data

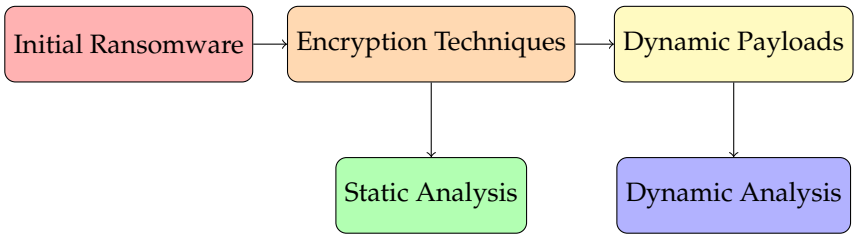


Figure 1. Conceptual diagram illustrating the evolution of ransomware and the analysis methods.

that may obfuscate the true nature of ransomware activity [10,39,40]. The EmRmR method, a cornerstone of this research, has been crafted to exclude redundant information while preserving the most critical data points, thereby ensuring that the most relevant features are highlighted [2,41]. This process is not only essential for the accurate identification of ransomware but also for the swift analysis of its behavior, which is paramount in a landscape where the speed of detection is often critical [1,12]. The selected classifiers are known for their robustness and have been deployed to validate the effectiveness of the feature selection process [15,22]. Each classifier has been rigorously trained and tested on a diverse array of data samples, with the intention of calibrating the detection system to recognize the subtle nuances of ransomware behavior [23,25]. Through this comprehensive evaluation, the classifiers are expected to exhibit a heightened sensitivity to the presence of ransomware, thus providing a crucial layer of defense against this ever-evolving digital threat [6,26,27].

The major contributions of this study include:

1. We examined the latest ransomware definition
2. We surveyed studies of static analysis and dynamic analysis of ransomware, and critiqued their shortcomings.
3. We listed and discussed major insights of this surveys.

The rest of the article is organized like this: Section 2 is about related works. Section 3 is our methodology. Section 4 is the experiment. Section 5 discusses the main implications of our study. Section 6 is the conclusion and future research direction.

2. Related works

This section explores ransomware history and definition.

2.1. Behavioral-Centric Detection Approaches

A plethora of studies have been conducted to uncover the mechanisms of ransomware identification, with a particular emphasis on behavior-centered approaches, and they have recorded and analyzed the malevolent actions executed by ransomware during its operational phase. Predominantly, these scholarly efforts have involved the application of machine learning classifiers [2]. These advanced computational models have been leveraged to scrutinize and interpret the behavioral attributes extracted from files, applying an amalgamation of filter and wrapper-based feature selection methodologies [39,40]. The classifiers' capability to identify data patterns and irregularities is essential for the detection and categorization of ransomware [10,41]. Notably, researchers have utilized the Min Max Relevance method, a strategic filtering process that isolates the most impactful features while eliminating less significant data [11,18]. In their pursuit of efficiency, these scholars have utilized the Windows API to track ransomware's actions [12,19]. These APIs act as conduits between the operating system and the ransomware, enabling the execution of malicious tasks [13,20]. By carefully monitoring these APIs, the researchers have endeavored to construct a detailed representation of ransomware's operational patterns [21,22]. The classifiers, chosen for their robustness and adaptability, range from Decision Trees, which divide the feature space into decision regions, to Support Vector Machines, known for constructing hyperplanes in a multidimensional space [14,23]. These classifiers have

undergone extensive training and testing on a wide range of data samples, with the aim of refining the detection systems to discern the faintest whispers of ransomware behavior [15,25]. This thorough evaluation of classifiers is anticipated to enhance the sensitivity to ransomware presence, thereby fortifying defenses against this evolving digital threat [26,42].

2.2. Static Analysis and Feature Reduction Techniques

Researchers have endeavored to devise sophisticated methods for identifying obfuscated ransomware. One approach that was developed involved a supervised learning methodology that focused on the extraction of mnemonic n-gram characteristics [9,39]. This technique, referred to as the Min Max Relevance filter, in conjunction with Principal Component Analysis, aimed to distill the essence of data features, thereby enhancing the classification process of potentially harmful executable files [10,17,43]. A comprehensive multi-tier feature diminution technique was also introduced to meticulously assess network traffic attributes that are frequently modified during ransomware attacks [40,41]. This method focused on the reduction of dimensional complexity while preserving the most significant indicators of ransomware behavior [2,11]. Later, a novel framework was proposed, emphasizing a non-signature-based detection system by utilizing a dual approach, integrating the robustness of support vector machine algorithms within a wrapper paradigm and the strategic selection capabilities of filter-based techniques, with the goal to isolate and identify ransomware based on its interaction with the Windows API, a crucial aspect that distinguishes benign from malignant executables [13,34]. Additionally, a distinctive detection system, referred to as the MDS, was put forward, to apply four sophisticated filter-based techniques to discriminate between malicious ransomware and legitimate files by analyzing the nuanced differences in the portable executable optional header fields, an area often exploited by ransomware for concealment and execution [12,19].

2.3. Dynamic Analysis Utilizing Windows API Calls

Investigators have shed light on ransomware's characteristics by examining Windows API call patterns across diverse strains of this malware [17,36,44]. For instance, a study highlighted the use of Windows API call features to outline the behavior of 13.7 distinct ransomware variants [33,34]. A detection approach utilizing the history of API call sequences combined with an SVM classifier has been introduced, offering a novel perspective on thwarting ransomware threats [2,9]. Building on this framework, an analysis detailed a spectrum of ransomware behaviors captured through Windows API calls, alongside registry key and file system operations observed during the initial stages of an attack [39,40]. The critical role of Microsoft's Cryptographic API in the encryption of sensitive data has been underscored, revealing a method by which ransomware generates encryption keys through a series of eight specific API calls [10,41]. Furthermore, a detection mechanism has been proposed, leveraging the weaknesses in the chaining modes employed by certain ransomware, thus enabling a potential reversal of the encryption with the cipher algorithm [11,18]. This mechanism has demonstrated the feasibility of intercepting ransomware during its encryption process, offering a glimmer of hope for data recovery [1,12].

3. Methodology

In this section, the methodology of this study is explored.

3.1. Environment setup

A comprehensive behavioral analysis of ransomware requires the deployment of samples within a controlled setting. Thus, the construction of the ransomware analysis infrastructure was carried out adhering to the high standards recommended in prominent research. The Cuckoo Sandbox, renowned for its proficiency in automating the analysis of harmful software, served as the cornerstone of the analysis toolkit. The host environment was established on an Ubuntu 22.04 LTS Desktop, carefully updated to ensure a stable and

Table 1. Configuration of the guest environment for ransomware analysis.

Component	Configuration
Operating System	Windows 11 23H2, 64bit
Security Measures	Deactivated (Anti-virus, Firewall, etc.)
Third-Party Applications	Microsoft Office, Acrobat Reader, Google Chrome
Virtualization Software	VirtualBox with Host-Only Adapter
Network Settings	Configured for C&C communication
Python Agent	Installed for Cuckoo communication
User Files	Created in standard directories
Browsing History	Established for analysis

secure analysis platform. The guest environment configuration is summarized in Table 1. This setup facilitates a comprehensive analysis by allowing a broader observation of ransomware activities under deliberately vulnerable conditions.

Following the setup detailed in Table 1, a Python agent was integrated, operating within the guest system to establish a seamless conduit for communication and data exchange between the Cuckoo Sandbox and the guest operating system. The intention behind these configurations was to offer a rich dataset for the analysis and to closely monitor the interaction of ransomware with user data. The anticipated outcome was to acquire a detailed understanding of the ransomware's behavioral patterns, which includes but is not limited to, the encryption of files, alteration of file structures, and any attempts to establish unauthorized communications.

3.2. Dataset

The composition of the data set employed in this investigation encompassed both ransomware and benign exemplars. A total of 353 ransomware instances, encompassing 14 distinct categories, were amassed from a multitude of sources that are publicly accessible, including but not restricted to, repositories such as VirusShare and VirusTotal. Additional sources like Malwarebytes and offensive-computing forums, known for their distribution of ransomware, were also utilized. Concurrently, 357 benign applications were amassed, sourced from reliable providers such as software-informer and inclusive of system files from the "system32" directory of an unblemished installation of Windows 11. To craft a data set that mirrors real-world conditions, applications that exhibit behavior similar to ransomware, yet are deemed benign, were included as detailed in Table 9. Each sample was meticulously stored in segregated files corresponding to their designated group. Ensuring the benign nature of the applications entailed a rigorous verification process, wherein the MD5 hash values of the collected software were cross-referenced with the Virus Total service, which integrates 36 prevalent antivirus programs. The identification of ransomware by family name posed significant challenges due to the voluminous nature of the malicious files. The strategy employed relied upon the labelling schema utilized by antivirus vendors, which is based on the frequency at which ransomware classes are identified across antivirus engines. The prevalent issue of erroneous labelling by some antivirus engines necessitated a systematic approach to parsing labels. A script, written in Python, facilitated the collation of labels from the antivirus programs commonly engaged in ransomware classification, with a benchmark value of 90% adopted as the criterion for family assignment: if 90% of the antivirus engines aligned in identifying a ransomware sample as part of a specific family, it was then catalogued accordingly.

3.3. Generating Windows API calls

This subsection discusses how to generate Windows API calls.

3.3.1. Tracing Windows API Calls

The process of Windows API call tracing plays a critical role in understanding the behavior of ransomware. To instigate an infection on the host system, ransomware must successfully initiate a sequence of Windows API calls. These calls are instrumental in differentiating between malicious and non-malicious files due to their role as an interface through which programs request services from the operating system's kernel. Therefore, monitoring these calls can be indicative of a program's ultimate objective.

Definition 1. *Windows API Call Trace* is an analytical method employed to dynamically record the execution of a process within a secure environment. Given a process K with an input I, and assuming the process K invokes a Windows API call C, where $K \in C$ with input I, the Windows API call trace C_t can be formally described as:

$$C_t = \{(K, C, I)\} \tag{1}$$

The comprehensive trace compiles a sequence of Windows API calls, each with its respective parameters and returned values from the process. To capture these dynamic behavioral traces, each ransomware sample was executed within an insulated environment. For a precise duration, ranging from 70.1 to 89.9 seconds, the behavior of the ransomware was scrutinized to reveal any malicious indicators. The utility of virtualization software was indispensable, as it allowed for the creation of system snapshots prior to the execution of ransomware. Post ransomware attack, the environment was restored to its prior uninfected state. A parallel was drawn for benign applications, executing them within the same insulated environment to accrue their behavioral profiles. To accurately trace Windows API calls, it is necessary for both ransomware and benign samples to run to completion. However, certain ransomware requires user interaction, such as mouse clicks or keyboard inputs, to activate their payload. To overcome this, a custom Python script was implemented to simulate basic user activities within the sandbox environment. These activities included web browsing, file manipulation, and desktop interaction. At the conclusion of the execution, the sandbox outputted a log file in JSON format, which is readable by humans and provides a record for each sample analyzed. While these reports encompass various dimensions of ransomware analysis, the focus was narrowed to the behavioral category, which elucidates the dynamic attributes of the ransomware.

3.3.2. Windows API feature abstraction

Following the acquisition of the behavioral log, the resultant report size from the sandbox was substantial, often spanning several hundred megabytes. The manual scrutiny of each report was deemed impractical due to the sheer volume of data, thus necessitating the development of a specialized parser engine. This engine, outlined in a comprehensive algorithm, was tasked with extracting behavioral characteristics from the output files. The parsing operation entailed reading the sandbox's extensive reports and systematically sifting through them to isolate the essential Windows API calls, thereby diminishing the search space. To streamline the trace sequences, a focused extraction was performed, retrieving only the list of Windows API call function identifiers linked to the process, while overlooking the parameters and the values returned. A selection of critical cryptographic Windows API calls, along with their associated parameters, were represented visually. In the effort to condense the behavioral logs derived from the sample's execution trace, a set of rules was implemented: Initially, the process identifier for the ransomware samples was pinpointed. Subsequently, an examination was conducted to determine the precise juncture at which the process instigated a new thread or spawned a child process, achieved by traversing through the logs. These calls were then ordered in a chronological fashion based on their timestamps. Ultimately, pertinent function calls were semantically distilled from the logs to construct an illustrative feature vector.

Algorithm 1: Trace Parsing and Feature Extraction

Input: Behavioral logs L
Output: Feature vector F
1 : Load logs L into parser engine
2 : $F \leftarrow \emptyset$ (Initialize feature vector as empty)
3 : for each log l in L do
4 : Identify process identifier PID of ransomware sample
5 : for each Windows API call c in l do
6 : if c is linked to PID then
7 : Extract function identifier FID from c
8 : if FID is not extraneous then
9 : $F \leftarrow F \cup \{FID\}$ (Add to feature vector)
10 : end if
11 : end for
12 : Sort F chronologically
13 : end for
14 : Refine F to eliminate noisy calls
15 : return F

(2)

The output traces harvested not only encapsulated the authentic Windows API calls but were also interspersed with extraneous and obfuscating calls fabricated by the ransomware to camouflage its deleterious intentions.

3.3.3. Constructing n-gram

In the process of constructing a sequential behavior profile based on the Windows API call log for each sample, subsequences exerting minimal influence on detection were identified and omitted. N-grams with lengths ranging from 2 to 5 were synthesized from the Windows API call traces of each scrutinized sample. Only n-grams that were derived from identical categories were extracted. For example, the sequence comprising NtTerminateThread, NtTerminateThread, and NtSuspendThread represents a 3-gram sequence associated with the process and thread operations aiming to cease and suspend a specific threat. The culmination of this n-gram Windows API call trace construction is encapsulated in the vector $\mathbf{V}$, which embodies the count of every conceivable n-gram.

Upon the generation of n-grams, the aggregate count of issues derived from the Windows API traces utilizing 2-gram, 3-gram, 4-gram, and 5-gram structures amounted to 32,134; 76,917; 186,909; and 227,152 respectively. Nevertheless, the magnitude of these features imposes a considerable demand on training duration and memory allocation. Recognizing that not all features are integral to the categorization of malicious samples, those subsequences deemed to be noise-inducing and of negligible impact on detection precision were pruned from the sequence trace.

$$V_{ngram} = \left\{ (g_1, g_2, \ldots, g_n) \mid g_i \in G, \sum_{i=1}^{n} \text{count}(g_i) \right\} \tag{3}$$

Here, G represents the set of all possible grams derived from the Windows API call logs, and $\text{count}(g_i)$ refers to the frequency of the individual grams within the log sequence. The refinement of the n-gram vector aims to retain only the most indicative sequences that contribute to the effective discrimination between benign and ransomware-related behaviors.

3.3.4. API refining process

In their attempts to elude detection, ransomware creators embed an extensive array of non-essential and redundant dynamic call sequences to obscure their runtime malevolent behavior and misdirect execution flow. Consequently, the volume of malevolent Windows API call sequences burgeons, engendering a highly noisy behavioral sequence. Windows API calls that fail to enhance the quality of detection are regarded as noise due to their non-contributory nature to detection efficacy. The performance of machine learning algorithms is contingent upon the presence or absence of such noise. Noise-laden datasets could deleteriously affect the learning models, manifesting in augmented processing times, escalated storage needs, and convoluted analysis of genuine malevolent intent, potentially leading to operational inefficiencies and compromised predictive capacities.

Definition 2. A Windows API call A is characterized as redundant A_R if the feature vector f_i and its correlated counterpart f_j possess a redundancy index R_{ij} approaching or equaling 1, indicating a complete association with no additive value towards the target outcome. Typically, the squared cosine similarity is utilized to ascertain the correlation between two Windows API call vectors (f_i, f_j), delineated as:

$$R_{ij} = \cos^2 \theta(f_i, f_j) \tag{4}$$

The goal is to simplify the model's complexity and bolster the algorithm's performance by excluding irrelevant and redundant Windows API calls that fail to contribute to heightened detection accuracy.

Syntax Definitions: Let P represent a process, F_S denote first seen, log file $\{L\}$, Windows API Call $\{A\}$, Database $\{DB\}$; Dynamic behavior $\{B_d\}$, timestamps $\{T\}$ and directory D. **Input:** File containing the Windows API traces $A_t = \{(P \in A; I)\}$ and features $\{f_1, f_2, f_3, \ldots, f_n\}$. **Output:** List of Windows API calls $L \in \{a_1 a_2 a_3 \ldots a_t\}$.

Algorithm 2: Trace Parsing and Feature Extraction

```
1 : Procedure FindAPICalls (A_t, Condition){
2 :   for each P in {L}['B'_d]['P'_d] do
3 :     if {L}[P_d] = Condition then
4 :       F_Stemp ← F_S
5 :       if F_S ≥ F_Stemp
6 :         F_S ← F_Stemp
7 :     for each trace in {L}[P_d][A_t = (P ∈ A; I)] do
8 :       if trace[i] does not include A then
9 :         ID ← A, modify DB & T[i] ← {T}[i]
10 :        DB[i][count] ← 1
11 :      else DB[i][T] ← append [T[i]]
12 :        DB[i][count] ← DB[i][count] + 1
13 : return DB[i]
14 :    end for
15 :  end for
16 :end
17 : Procedure Loadfile (D, PR){
18 :  for (D, PR) in G_file() do
19 :    for i, N in enumerate [all_files] do
20 :      if N ends with ('.json') then
21 :        FN ← N
22 :        Sdata ← load (D + FN)
23 :        FindAPICalls(Sdata, true)
24 :      end if
25 :    end for
26 :  end for
27 :end}
```

(5)

3.3.5. Removing noise

The extraction of sequences from the N-Grams often encompasses numerous instances that fail to single out the functions of malicious files. Isolating these distinctive subset features from the Windows API call logs is a paramount task in the analysis phase, being both time-intensive and necessitating considerable effort for data cleansing. The mRmR has seen wide application across various domains, including intrusion detection. The Maximum Relevance process singles out the most relevant feature associated with the behavior of a malicious class without accounting for relationships among features. Given a set G of features, and collective information $MI(f_i; M)$ representing the mutual information among feature f_i and the ransomware class, the Maximum Relevance is quantified as:

$$\text{MaxRelevance}(f_i; M) = \frac{1}{|G|} \sum_{f_i \in G} MI(f_i; M) \tag{6}$$

Table 2. Comparative accuracy of classifiers with varying n-gram sequences.

n-gram	SVM	kNN	DT	RF
2-g	66.7%	58.7%	63.7%	67.7%
3-g	98.7%	88.7%	86.7%	78.7%
4-g	96.7%	76.7%	74.7%	61.7%
5-g	97.7%	78.7%	76.7%	59.7%

Opting for features based on Maximum Relevance (mR) can discern behaviors relevant to ransomware. To mitigate the issue of repetition in mR, the Minimum Redundancy (mR) criterion is employed and articulated as:

$$\text{MinRedundancy}(f_i; C) = \frac{1}{|S|^2} \sum_{f_i, f_j \in S} MI(F_i; F_j) \tag{7}$$

Where $MI(F_i; F_j)$ is the mutual information between feature F_i and feature F_j. By amalgamating these two conditions—Maximum Relevance and Minimum Redundancy—into a singular score function denoted by mRMR, one achieves a balance, employing the mutual information difference (mRMR MID) defined as:

$$\text{MID} = \max_{S} \left[MI(f_i; \Omega) - \frac{1}{|S|} \sum_{f_i, f_j \in S} MI(F_i; F_j) \right] \tag{8}$$

Separate features including redundancy and relevance quantified by computing the common information between selected characters. Two Windows API call features are deemed independent if their mutual information score is the minimal value $MI(f_i; f_j)$. The joint probability distribution $p(x, y)$ alongside the marginal probabilities $p(x)$ and $p(y)$, allows for defining the mutual information of feature pairs as:

$$MI(f_i; f_j) = \sum_{x,y} p(x, y) \log \left(\frac{p(x, y)}{p(x)p(y)} \right) \tag{9}$$

4. Results

This section shows the results of this study.

4.1. Classifier Performance Analysis Based on N-Gram Feature Lengths

The efficacy of various classifiers was scrutinized based on n-gram lengths of 2, 3, 4, and 5, derived from each Windows API call sequence. This analysis was facilitated by the train-test split methodology, which bifurcates the dataset into a training set and a testing set. Prior to the construction of the model, the dataset was divided, ensuring a uniform distribution where 80.7% constituted the training set and 19.3% formed the testing set. The classifiers' precision was evaluated by metrics such as accuracy and the values of the Area under the Receiver Operating Characteristic Curve (AUC).

The Support Vector Machine (SVM) classifier, when utilizing a 3-gram approach, manifested superior accuracy, with a spectrum from 66.7% to 98.7%, and a reduced false-positive rate of approximately 0.0267. The robustness of this classifier was observed to be consistent with the implementation of 4-gram and 5-gram feature sequences. According to the data presented in Table 2, the k-Nearest Neighbors (kNN) and Decision Tree (DT) algorithms revealed almost parallel accuracy. These classifiers, with a 2-gram feature set, exhibited a lower accuracy threshold, whereas a significant increase was noted with the application of a 3-gram sequence.

The Random Forest (RF) classifier displayed the least accurate results across the spectrum of classifiers when the models were subjected to various lengths of n-grams. Specifically, the accuracy was moderately satisfactory with 67.7% for 2-grams and 78.7%

Table 3. Classifier Performance Evaluation with Varied N-Gram Feature Lengths

n-gram Length	SVM	kNN	DT	RF
2-gram	66.1%	58.1%	63.1%	67.1%
3-gram	98.1%	88.1%	86.1%	78.1%
4-gram	96.1%	76.1%	74.1%	61.1%
5-gram	97.1%	78.1%	76.1%	59.1%

for 3-grams. However, a decline in accuracy to 61.7% and 59.7% was observed for 4-grams and 5-grams, respectively. This decrease could be attributed to the increased computational complexity required by the RF classifier to construct the model with larger n-gram sizes, leading to overfitting issues during the training phase and consequently a lack of generalization for new features during testing. In contrast, Logistic Regression (LR) exhibited remarkable performance, with an accuracy of 72.3% and a lower false-positive rate of 0.413 for 2-gram features. This can be ascribed to the classifier's lower computational demands with smaller n-gram sizes.

The Receiver Operating Characteristic (ROC) curves show that overall AUC values underwent a marked decrease when 2-gram based features were utilized for model training. This is likely due to shorter n-grams' inability to encapsulate ransomware behavior adequately, thus increasing false positive alerts. A broadening of feature dimensionality, achieved by expanding the n-gram window to a 3-gram, resulted in a dramatic rise in AUC values for the classifiers. As seen in Figure **??** (c), a marginal decline in AUC by an average of 0.167% was observed for n = 4. Furthermore, Figure **??** (d) presents a comparative analysis of the ROC curves for longer sequences with n = 5 across the classifiers, indicating a significant drop in AUC values. This could be because extended sequences of n-grams are not efficacious in differentiating ransomware behavior from benign processes. Moreover, the increment in the length of n-gram sequences has a direct correlation with the training duration required for the model.

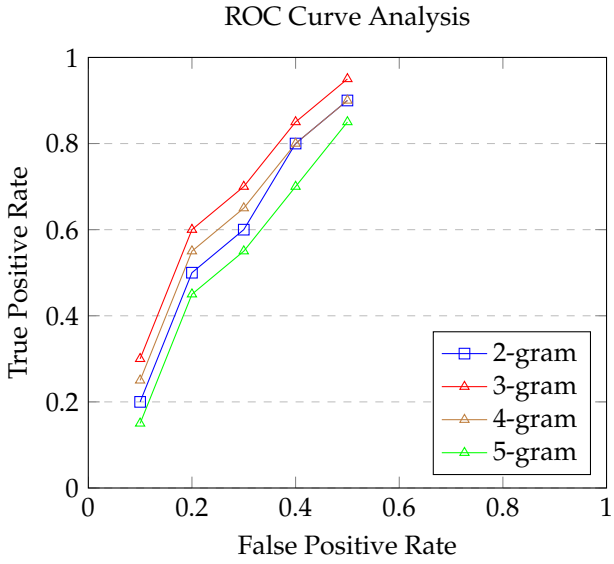
4.2. Enhanced Analysis of Classifier Efficacy Based on Variable N-Gram Feature Sizes

The objective of this investigation was to assess the classifiers' effectiveness when applied to an array of n-gram feature sizes, utilizing a tenfold cross-validation method. A notable limitation observed with the train-test splitting approach is the potential for class imbalance post-dataset division, which may lead to an erroneous estimate of the holdout error rate. To circumvent this issue, tenfold cross-validation was employed, which serves to mitigate overfitting and more accurately evaluate the model's performance. In this method, the dataset was thoroughly shuffled and partitioned into ten equal-sized segments. During each of the ten iterations, nine segments were utilized to develop the model, and the remaining segment was used for evaluation.

The Support Vector Machine (SVM), when employing a 3-gram scheme, demonstrated remarkable accuracy levels, varying from 66.1% to 98.1%, accompanied by a diminished false-positive rate of approximately 0.0261. The SVM's consistency was notable across the board, with the 4-gram and 5-gram feature sequences showing stable performance metrics. According to Table 3, k-Nearest Neighbors (kNN) and Decision Tree (DT) algorithms exhibited commensurate levels of accuracy. The 2-gram feature set resulted in lower accuracy rates for these classifiers, which noticeably improved upon the adoption of a 3-gram sequence.

Random Forest (RF) classifiers yielded the least precise outcomes among the spectrum of classifiers when evaluated across diverse n-gram lengths. In particular, the RF classifier's accuracy was moderately acceptable with a 67.1% for 2-grams and 78.1% for 3-grams. A downward trend in accuracy was noted for 4-grams and 5-grams, registering at 61.1% and 59.1% respectively. Such a trend suggests an escalation in computational demands as the n-gram size increases, potentially causing overfitting during the training phase and a subsequent inability to generalize when confronted with new features in the testing phase.

In stark contrast, Logistic Regression (LR) outshone other classifiers with an accuracy of 72.3% and a lower false-positive rate of 0.0413 for 2-gram features, attributed to its lesser computational requirements for smaller n-gram sizes.



5. Discussions

This section provides an in-depth interpretation of the ransomware detection study results across 4 key dimensions. The first 3 subsections discuss the major insights uncovered regarding the efficacy of feature engineering strategies, comparative classifier performance, and the role of enhanced feature selection techniques. The final subsection examines the limitations present within the current study.

5.1. Efficacy of N-Gram Based Feature Engineering

The n-gram based feature engineering approach proved instrumental in constructing an informative behavioral profile for both ransomware and benign files. Shorter length 2-grams were unable to adequately encapsulate the intricacies of ransomware behavior, resulting in suboptimal detection rates. This limitation necessitated an expansion of the n-gram window to 3-grams, which dramatically enhanced feature dimensionality. The 3-gram sequences enabled a more thorough characterization of ransomware behavioral attributes, significantly boosting detection accuracy levels. However, further lengthening n-grams beyond 3 led to a decline in efficacy, implying that excessive n-gram sizes fail to provide additional discriminative power. Overall, 3-gram feature sequences offered an optimal balance between dimensionality and informational content regarding ransomware activities. The study demonstrates the importance of meticulous feature engineering to extract indicators with high relevance towards identifying ransomware. Mindful tuning of n-gram sizes is essential to construct feature vectors that maximize detection performance.

5.2. Comparative Analysis of Classifier Performance

The spectrum of classifiers exhibited varying levels of accuracy across the different n-gram feature lengths. SVM consistently outperformed other models, underscoring its suitability for discerning ransomware behaviors within the high-dimensional feature space produced by n-grams. SVM's maximal margin property enables an effective separation between benign and ransomware classes. The kNN and DT classifiers also showed competency, although their accuracy was inferior compared to SVM. In contrast, RF classifiers displayed subpar outcomes, which further deteriorated as n-gram sizes increased. The elevated feature dimensionality likely overwhelmed RF's decision tree constituents, resulting in overfitting. Furthermore, RF's bagging approach failed to ameliorate the overfitting problem. The study highlights SVM's superiority in modeling ransomware behaviors

from Windows API call traces. Tailoring classifiers to the intricacies of feature engineering choices is imperative for maximizing detection efficacy.

5.3. Relevance of Enhanced Feature Selection

The integration of feature selection techniques such as mRMR helped to refine the core indicators differentiating ransomware from benign entities. By pruning redundant and non-informative features from the n-gram vectors, mRMR reduced complexity and retained only the most decisive attributes. This process of enhancement enabled easier interpretation of how specific API call sequences characterize ransomware operations. The mRMR decreased computational overhead by diminishing the volume of features fed into classifiers. The improved efficiency facilitated quicker model training to rapidly adapt to new ransomware strains. Overall, the strategic application of feature selection is integral to elevate both the speed and accuracy of ransomware detection frameworks.

5.4. Limitations of the Study

While providing valuable insights, the study possesses certain limitations that warrant acknowledgement. The dataset composition relied exclusively on Windows PE files, limiting the extension of findings across other platforms. Additionally, the scope was confined to static analysis, without considering run-time interactions such as file encryption. There is a need for continued research with expanded datasets covering diverse file types. Execution in sandboxes could reveal supplementary dynamic indicators beyond API calls. Finally, testing across adversarial evasion attempts can further validate the detection framework's resilience. This study provided a rigorous examination of feature engineering and selection strategies for optimizing ransomware detection from Windows API calls. The techniques discussed will serve as a foundation for developing robust anti-ransomware systems. Future work should focus on building ensemble classifiers that synthesize the strengths of multiple algorithms. With continuous enhancement, ransomware detection will grow more potent in safeguarding against cyber extortion.

6. Conclusion and Future Research Directions

This study presented a novel approach for ransomware detection centered on strategic feature engineering and selection techniques applied to Windows API call traces. The efficacy of diverse n-gram feature lengths was analyzed, with 3-grams offering optimal characterization of ransomware behaviors. SVMs demonstrated exceptional accuracy in classifying ransomware based on the n-gram vectors. Enhanced feature selection via mRMR further refined the predictive capabilities by isolating the most decisive API call indicators. The proposed methodology provides a robust foundation for analyzing and constraining ransomware operations.

While this study marks an important step towards combating ransomware threats, additional research is warranted to build on the current findings. Future work should encompass dynamic analysis within sandboxes to uncover supplementary behavioral characteristics beyond API calls. Testing across a broader spectrum of platforms and file types would improve generalizability. Furthermore, adversarial evasion attempts can help bolster the detection framework's resilience. Advanced ensembles synthesizing multiple classifiers could yield improvements over individual techniques. With emerging strains growing increasingly sophisticated, continuous innovation in feature engineering and selection is imperative to sustain high ransomware detection accuracy. This study lays a methodical groundwork to be expanded through multi-pronged behavioral modeling and enhanced machine learning algorithms. Going forward, ransomware mitigation frameworks must persist in their evolution to counteract the unrelenting progression of cyber extortion.

Conflicts of Interest: The authors declare that there is no conflict of interest.

References

1. Almashhadani, A.O.; Kaiiali, M.; Sezer, S.; O'Kane, P. A multi-classifier network-based crypto ransomware detection system: a case study of Locky ransomware. *Ieee Access* **2019**, *7*, 47053–47067.
2. Ahmed, U.; Lin, J.C.W.; Srivastava, G. Mitigating adversarial evasion attacks of ransomware using ensemble learning. *Computers and Electrical Engineering* **2022**, *100*, 107903.
3. McIntosh, T.; Kayes, A.; Chen, Y.P.P.; Ng, A.; Watters, P. Dynamic user-centric access control for detection of ransomware attacks. *Computers & Security* **2021**, *111*, 102461.
4. Eliando, E.; Purnomo, Y. LockBit 2.0 Ransomware: Analysis of infection, persistence, prevention mechanism. *CogITo Smart Journal* **2022**, *8*, 232–243.
5. Hwang, J.; Kim, J.; Lee, S.; Kim, K. Two-stage ransomware detection using dynamic analysis and machine learning techniques. *Wireless Personal Communications* **2020**, *112*, 2597–2609.
6. Davies, S.R.; Macfarlane, R.; Buchanan, W.J. Evaluation of live forensic techniques in ransomware attack mitigation. *Forensic Science International: Digital Investigation* **2020**, *33*, 300979.
7. McIntosh, T.; Kayes, A.; Chen, Y.P.P.; Ng, A.; Watters, P. Applying Staged Event-Driven Access Control to Combat Ransomware. *Computers & Security* **2023**, p. 103160.
8. Zuhair, H.; Selamat, A.; Krejcar, O. A Multi-Tier Streaming Analytics Model of 0-Day Ransomware Detection Using Machine Learning. *Applied Sciences* **2020**, *10*, 3210.
9. Ahmed, M.E.; Kim, H.; Camtepe, S.; Nepal, S. Peeler: Profiling kernel-level events to detect ransomware. In Proceedings of the Computer Security–ESORICS 2021: 26th European Symposium on Research in Computer Security, Darmstadt, Germany, October 4–8, 2021, Proceedings, Part I 26. Springer, 2021, pp. 240–260.
10. Al-Hawawreh, M.; Sitnikova, E.; Aboutorab, N. Asynchronous peer-to-peer federated capability-based targeted ransomware detection model for industrial IoT. *IEEE Access* **2021**, *9*, 148738–148755.
11. Al-Dwairi, M.; Shatnawi, A.S.; Al-Khaleel, O.; Al-Duwairi, B. Ransomware-Resilient Self-Healing XML Documents. *Future Internet* **2022**, *14*, 115.
12. AlMajali, A.; Qaffaf, A.; Alkayid, N.; Wadhawan, Y. Crypto-Ransomware Detection Using Selective Hashing. *2022 International Conference on Electrical and Computing Technologies and Applications (ICECTA)* **2022**, pp. 328–331.
13. Axon, L.; Erola, A.; Agrafiotis, I.; Uuganbayar, G.; Goldsmith, M.; Creese, S. Ransomware as a Predator: Modelling the Systemic Risk to Prey. *Digital Threats: Research and Practice*.
14. Bekkers, L.; van't Hoff-de Goede, S.; Misana-ter Huurne, E.; van Houten, Y.; Spithoven, R.; Leukfeldt, E.R. Protecting Your Business against Ransomware Attacks? Explaining the Motivations of Entrepreneurs to Take Future Protective Measures against Cybercrimes Using an Extended Protection Motivation Theory Model. *Computers & Security* **2023**, *127*, 103099.
15. Bold, R.; Al-Khateeb, H.; Ersotelos, N. Reducing False Negatives in Ransomware Detection: A Critical Evaluation of Machine Learning Algorithms. *Applied Sciences* **2022**, *12*, 12941.
16. Davies, S.R.; Macfarlane, R.; Buchanan, W.J. Review of Current Ransomware Detection Techniques. In Proceedings of the Proc. of the 7 th International Conference on Engineering and Emerging Technologies (ICEET). Institute of Electrical and Electronics Engineers, 2022.
17. Abbasi, M.S.; Al-Sahaf, H.; Mansoori, M.; Welch, I. Behavior-based ransomware classification: A particle swarm optimization wrapper-based approach for feature selection. *Applied Soft Computing* **2022**, *121*, 108744.
18. Albulayhi, K.; Al-Haija, Q.A. Early-stage Malware and Ransomware Forecasting in the Short-Term Future Using Regression-based Neural Network Technique. In Proceedings of the 2022 14th International Conference on Computational Intelligence and Communication Networks (CICN). IEEE, 2022, pp. 735–742.
19. Almeida, F.; Imran, M.; Raik, J.; Pagliarini, S. Ransomware Attack as Hardware Trojan: A Feasibility and Demonstration Study. *IEEE Access* **2022**, *10*, 44827–44839.
20. Aurangzeb, S.; Rais, R.N.B.; Aleem, M.; Islam, M.A.; Iqbal, M.A. On the classification of Microsoft-Windows ransomware using hardware profile. *PeerJ Computer Science* **2021**, *7*, e361.
21. Baksi, R.P. Pay or Not Pay? A Game-Theoretical Analysis of Ransomware Interactions Considering a Defender's Deception Architecture **2022**. pp. 53–54.
22. Beaman, C.; Barkworth, A.; Akande, T.D.; Hakak, S.; Khan, M.K. Ransomware: Recent advances, analysis, challenges and future research directions. *Computers & security* **2021**, *111*, 102490.
23. Berrueta, E.; Morato, D.; Magaña, E.; Izal, M. Crypto-ransomware detection using machine learning models in file-sharing network scenarios with encrypted traffic. *Expert Systems with Applications* **2022**, *209*, 118299.
24. McIntosh, T.; Kayes, A.; Chen, Y.P.P.; Ng, A.; Watters, P. Ransomware mitigation in the modern era: A comprehensive review, research challenges, and future directions. *ACM Computing Surveys (CSUR)* **2021**, *54*, 1–36.
25. Borah, P.; Bhattacharyya, D.K.; Kalita, J.K. Cost effective method for ransomware detection: an ensemble approach **2021**. pp. 203–219.
26. Celdrán, A.H.; Sánchez, P.M.S.; Scheid, E.J.; Besken, T.; Bovet, G.; Pérez, G.M.; Stiller, B. Policy-based and Behavioral Framework to Detect Ransomware Affecting Resource-constrained Sensors **2022**. pp. 1–7.
27. Connolly, A.Y.; Borrion, H. Reducing ransomware crime: analysis of victims' payment decisions. *Computers & Security* **2022**, *119*, 102760.

28. Davies, S.R.; Macfarlane, R.; Buchanan, W.J. NapierOne: A modern mixed file data set alternative to Govdocs1. *Forensic Science International: Digital Investigation* **2022**, *40*, 301330.
29. Davies, S.R.; Macfarlane, R.; Buchanan, W.J. Differential area analysis for ransomware attack detection within mixed file datasets. *Computers & Security* **2021**, *108*, 102377.
30. De Gaspari, F.; Hitaj, D.; Pagnotta, G.; De Carli, L.; Mancini, L.V. Evading behavioral classifiers: a comprehensive analysis on evading ransomware detection techniques. *Neural Computing and Applications* **2022**, *34*, 12077–12096.
31. Du, J.; Raza, S.H.; Ahmad, M.; Alam, I.; Dar, S.H.; Habib, M.A. Digital Forensics as Advanced Ransomware Pre-Attack Detection Algorithm for Endpoint Data Protection. *Security and Communication Networks* **2022**, *2022*, 1–16.
32. Davies, S.R.; Macfarlane, R.; Buchanan, W.J. Majority Voting Ransomware Detection System. *Journal of Information Security* **2023**, *14*.
33. Ahmed, Y.A.; Koçer, B.; Al-rimy, B.A.S. Automated Analysis Approach for the Detection of High Survivable Ransomware. *KSII Transactions on Internet and Information Systems (TIIS)* **2020**, *14*, 2236–2257.
34. Ahmed, Y.A.; Koçer, B.; Huda, S.; Al-rimy, B.A.S.; Hassan, M.M. A system call refinement-based enhanced Minimum Redundancy Maximum Relevance method for ransomware early detection. *Journal of Network and Computer Applications* **2020**, *167*, 102753.
35. Faghihi, F.; Zulkernine, M. RansomCare: Data-centric detection and mitigation against smartphone crypto-ransomware. *Computer Networks* **2021**, *191*, 108011.
36. Adamov, A.; Carlsson, A. Reinforcement learning for anti-ransomware testing. In Proceedings of the 2020 IEEE East-West Design & Test Symposium (EWDTS). IEEE, 2020, pp. 1–5.
37. Iqbal, M.J.; Aurangzeb, S.; Aleem, M.; Srivastava, G.; Lin, J.C.W. RThreatDroid: A Ransomware Detection Approach to Secure IoT Based Healthcare Systems. *IEEE Transactions on Network Science and Engineering* **2022**.
38. McIntosh, T.; Liu, T.; Susnjak, T.; Alavizadeh, H.; Ng, A.; Nowrozy, R.; Watters, P. Harnessing GPT-4 for generation of cybersecurity GRC policies: A focus on ransomware attack mitigation. *Computers & Security* **2023**, *134*, 103424.
39. Al-rimy, B.A.S.; Maarof, M.A.; Alazab, M.; Alsolami, F.; Shaid, S.Z.M.; Ghaleb, F.A.; Al-Hadhrami, T.; Ali, A.M. A Pseudo Feedback-Based Annotated TF-IDF Technique for Dynamic Crypto-Ransomware Pre-Encryption Boundary Delineation and Features Extraction. *IEEE Access* **2020**.
40. Al-rimy, B.A.S.; Maarof, M.A.; Alazab, M.; Shaid, S.Z.M.; Ghaleb, F.A.; Almalawi, A.; Ali, A.M.; Al-Hadhrami, T. Redundancy coefficient gradual up-weighting-based mutual information feature selection technique for crypto-ransomware early detection. *Future Generation Computer Systems* **2020**.
41. Al-Haija, Q.A.; Alsulami, A.A. High performance classification model to identify ransomware payments for heterogeneous bitcoin networks. *Electronics* **2021**, *10*, 2113.
42. Zhang, C.; Luo, F.; Ranzi, G. Multistage Game Theoretical Approach for Ransomware Attack and Defense. *IEEE Transactions on Services Computing* **2022**.
43. Davies, S.R.; Macfarlane, R.; Buchanan, W.J. Exploring the Need For an Updated Mixed File Research Data Set. In Proceedings of the 2021 International Conference on Engineering and Emerging Technologies (ICEET). IEEE, 2021, pp. 1–5.
44. A. Alissa, K.; H. Elkamchouchi, D.; Tarmissi, K.; Yafoz, A.; Alsini, R.; Alghushairy, O.; Mohamed, A.; Al Duhayyim, M. Dwarf mongoose optimization with machine-learning-driven ransomware detection in internet of things environment. *Applied Sciences* **2022**, *12*, 9513.