ISSN: 0067-2904

Dynamic API Call-Based Machine Learning for Early-Stage Ransomware Detection

Nameer Nail Taha*, Nada A.Z. Abdullah
Department of Computer Science, College of Science, University of Baghdad, Baghdad, Iraq

Received: 26/10/2024 Accepted: 20/3/2025 Published: 30/3/2026

Abstract

Ransomware is a serious threat while using the computer; it may cause the user's data to be encrypted or not allow them to access computers and require users to pay a ransom to open files. However, scientific work in this area has not been fully addressed, and many challenges are mentioned in previous studies. This approach employs a dedicated pipeline for feature engineering and a collection of machine-learning models for binary classification. The primary innovation in dynamic analysis is using API call functionalities to extract engineered features, which capture distinct behavioral characteristics associated with various ransomware families. This method allows the model to differentiate between benign and malicious operations with considerable accuracy. The models were trained and assessed using a recent dataset that included 2,311 samples from 13 distinct ransomware families. As for the results, experiments have shown that the models achieved accuracy rates of 99.5% in SVM, 99.7% in KNN, 99.7% in Decision Tree, and 99.9% in XGBoost, respectively, and when evaluating the evaluation criteria, the models showed outstanding performance across many metrics. These findings confirm that the models can accurately detect ransomware patterns and make reliable predictions, thereby validating the efficiency and effectiveness of the proposed methodology.

Keywords: Ransomware attacks, Early-stage detection, Machine learning, Binary classification, Cybersecurity resilience, Dynamic analysis.

التعلم الآلي القائم على استدعاء واجهة برمجة التطبيقات الديناميكية للكشف عن برامج الفدية في المرحلة المبكرة

نمير نائل طه*, ندا عبد الزهرة عبد الله
قسم الحاسوب، كلية العلوم، جامعة بغداد، بغداد، العراق

الخلاصة

إن برامج الفدية تشكل تهديدًا خطيرًا لمستخدمي اجهزة الحاسوب؛ فقد تتسبب في تشفير بيانات المستخدم أو عدم السماح له بالوصول إلى أجهزتهم وتطلب منهم دفع فدية لفتح تشفير الملفات. ومع ذلك، لم يتناول البحث العلمي هذا المجال بشكل كامل، وتم ذكر العديد من التحديات في الدراسات السابقة. لذلك، ركز هذا البحث على تطوير طريقة جديده في اكتشاف هذه البرامج لتجنب المشاكل التي يواجها مستخدمو الحاسوب من خلال دراسة ما تفتقر اليه الدراسات السابقة واستخدام طريقة لاستخراج خصائص احصائية من خصائص

*Email: Nameer.Taha2201m@sc.uobaghdad.edu.iq

البيانات الأصلية نفسها ومجموعة من نماذج التعلم الآلي للتصنيف الثنائي. حيث يتمثل الابتكار الأساسي في التحليل الديناميكي في استخدام وظائف استدعاء واجهة برمجة التطبيقات لاستخراج الميزات المصممة، والتي تلتقط السلوكيات المميزة المرتبطة بعائلات برامج الفدية المختلفة. تسمح هذه الطريقة للنموذج بالتمييز بين العمليات الحميدة والخبيثة بدقة كبيرة. تم تدريب النماذج وتقييمها باستخدام مجموعة بيانات حديثة تضمنت 2311 عينة من 13 عائلة لبرامج فدية مميزة. أما بالنسبة للنتائج، فقد أظهرت التجربة أن النماذج حققت معدلات دقة بلغت 99.5% في خوارزمية SVM، و99.7% في خوارزمية KNN، و99.7% في خوارزمية Decision Tree، و99.9% في خوارزمية XGBoost، على التوالي، وعند تطبيق معايير تقييم اخرى، أظهرت النماذج أداءً متميزًا عبر العديد من المقاييس. وتؤكد هذه النتائج أن النماذج يمكنها اكتشاف مختلف برامج الفدية بدقة وتقديم تنبؤات موثوقة، وبالتالي التحقق من كفاءة وفعالية المنهجية المقترحة.

1. Introduction

Ransomware is a serious threat that infects users by encrypting their data or preventing them from accessing their devices, requiring those affected to pay a ransom to regain control or access [1]. The malicious software uses the Command and Control (C&C) servers to control the communication channels, allowing them to encrypt their victims' data [2]. This encrypted data is held hostage and will only be decrypted after the ransom is paid [2]. In cases involving ransomware, the ransom demand cannot be overlooked, as such refusal could lead to the perpetrators destroying or publishing sensitive data on public platforms [3]. In short, the consequences could be more than financial restrictions, as the data could harm the users' reputations and affect customers' confidence in specific organizations, resulting in lengthy and complex legal issues [3].

Concerns regarding ransomware are not baseless as the risks of being attacked are high due to advanced ransomware technology and sophisticated attacking and encrypting methods [4]. The attacks usually target large corporations, government agencies, healthcare organizations, and key infrastructure that could harm the economy, disrupt access to critical online services, breach confidential information, and decrease the public's confidence in the systems [5].

The economic and operational impacts of ransomware attacks on businesses can lead to significant financial losses due to ransom payments and downtime, loss of productivity, and reputational damage [6].

The attack on the Colonial Pipeline in 2021 disrupted fuel supplies across the Southeastern United States, illustrating the potential for ransomware to impact critical infrastructure [7].

The rise in the attacks is attributed to Ransomware-as-a-Service expansion that allows the software to be sold in underground marketplaces and purchased discreetly and anonymously by financially motivated, unscrupulous individuals using cryptocurrency [8]. Therefore, as detecting and identifying ransomware attacks as early as possible is detrimental, preventive, solid measures are needed to combat the software before it could cause severe damage [9]. For early detection purposes, comprehensively devised strategies, for instance, implementing updated and strong cybersecurity measures to safeguard the servers, working collaboratively with other organizations and cybersecurity specialists globally, and introducing the latest legislative enactments to deter and convict the perpetrators [10].

Proactive strategies such as user education and regular security audits can also effectively reduce the likelihood of ransomware attacks. These strategies should not be overlooked, as cryptocurrency kept the buyers anonymous and encouraged the malware's threat to spread. Hackers know their anonymity prevents law enforcement from identifying and detecting their locations. To effectively counteract ransomware, all stakeholders must collaborate and implement strategies to safeguard digital infrastructure and protect against the growing risk of cyber extortion. [11]

Several challenges regarding ransomware detection using various techniques have been identified in the literature to create efficient machine-learning strategies for identifying ransomware. There is a critical need to enhance the robustness and generalizability of machine-learning models for ransomware detection by addressing several interrelated challenges. In the study [12], experimental results show that sub-sequences of API calls can successfully identify ransomware traits. However, this method strongly depends on the quality of the implemented emulator. The study [13] mentions that, to the best of the authors' knowledge, no prior work has implemented machine learning algorithms, and they are compared based on different sets of parameters for each algorithm. As for machine learning algorithms, they exist in numerous modifications to train the classifiers of malware detection, each suitable for different datasets. The study [14] highlighted the limitation of using a small sample size (58 ransomware and 66 benign samples). This limitation could affect the detection process. The study suggests future work should use other ML techniques or deep learning with up-to-date samples to enhance malicious software detection. The study's authors in [15] particularly refer to limited datasets and evolving ransomware threats. The paper [16] further supports this by recognizing that dynamic API call features should be utilized to improve ransomware identification.

Various machine learning models hold different advantageous features while advancing the production of effective solutions for ransomware detection:

Support Vector Machines have been effectively used in cybersecurity and malware detection since they are particularly appropriate for dimensions of binary classification. SVMs have been employed in malware detection based on performance, system call information, file system, and network traffic [17, 18]. On the same note, SVMs have also been used in intrusion detection, where they try to discover network activities considered abnormal and may represent cyber-attacks; spam detection is categorized as spam or not [17]. In addition to cybersecurity, SVMs have also been utilized to detect malware by categorizing a file or a program as either malware or non-malware based on their features, such as the internal structure of a file, application program interfaces, and the behavioral patterns of a program [18]. Due to the ability of SVMs to work with high-dimensional and non-linear data, they are beneficial for several binary classification problems in cybersecurity and malware detection [19].

The K-Nearest Neighbors algorithm is one of the most recognizable machine learning algorithms. It finds the nearest neighbors in the training set to the given new data and then assigns the new data to the class with the most neighbors [20-22]. From previous studies, it has been observed that KNN has been applied to solve various cybersecurity problems, such as malware identification, by classifying the nature of essential files based on transaction characteristics [23]. One of the significant benefits of KNN is that it is easy to implement compared to other methods, highly flexible because it does not assume anything regarding the distribution of the data available and does not assume any probability distribution of the data sets. All these qualities make it a powerful instrument in numerous fields, including cybersecurity.

Decision trees are one of the powerful algorithms of machine learning that can be applied efficiently for binary classifiers in cybersecurity and malware detection. Their functioning implies the recursive partitioning of data based on the most informative variables and creates a tree-like structure that predicts from the root node to the end node [24-26]. In cybersecurity, decision trees should also be useful in ordering network traffic or users' actions as normal or anomalous. This classification assists in other classifications such as intrusions, malware, and phishing campaigns [24]. Likewise, decision trees can be used in malware detection through training on the contents of files or programs, including APIs, file characteristics, and behaviors. These facets define whether the files or programs are malicious or, on the

contrary, not [26, 27]. There are several advantages associated with decision trees. These are the following advantages associated with using decision trees: Easy to interpret, it can handle numerical as well as categorical data, and at the same time, the algorithm can handle outlier as well as noisy data. These qualities make decision trees an ideal tool for security analysts.
XGBoost is an efficient and optimized gradient boosting framework, which is a highly effective type of ensemble learning methods that integrate many weak classifiers to construct a strong, accurate, and highly effective classifier. Consequently, due to the system's regularization, parallel executing capability, and the way it manages missing data, it is highly appropriate for binary classification tasks in cybersecurity and malware detection [28]. XGBoost has been applied in these domains due to its ability to classify normal and anomalous circumstances, such as URL, email, network traffic, and malware [28].
A systematic evaluation and comparison of multiple machine learning algorithms, considering various parameters and configurations, are required to identify the most effective ones for different detection scenarios.
To our knowledge, previous studies did not adequately address using the latest feature engineering and machine learning techniques in ransomware detection with high accuracy and extreme effectiveness. This gap negatively affected the ability of previous models to identify ransomware patterns comprehensively and accurately. Since then, the study has sequentially defined its objectives to enhance research and analysis in this area. The goal is to develop advanced feature engineering techniques that enhance the model's ability to effectively extract data patterns, thereby improving its overall ransomware recognition performance. The researchers then directed their efforts toward developing and training four robust models for machine learning: K-Nearest Neighbors, XGBoost, Decision Tree, and Support Vector Machine. These models were trained using a resourceful and elaborate dataset to provide efficient and accurate ransomware detection. Last, the researchers assessed different models created based on the test samples in the frames of the generally approved scales. This assessment assists in validating the model's ability to handle real scenarios, appreciate the efficiency of the approach used in the study, and combat ransomware. In this sequence, the study elaborated on the absence of prior research and highlighted the goals to work towards a research avenue that fosters the improvement of models' capability to better detect and prevent ransomware.

This paper's remaining content is structured as follows: in Section 1is the Introduction. Section 2: Background section discusses previous research on ransomware detection and pre-sorting of sources to improve the search for the most relevant and effective materials. Section 3: Methodology: The research methodology is discussed and explained in detail regarding the essence of the solution and its components. Section 4: Experimental Results and Discussion is the section where the authors report the outcomes of the experiments and demonstrate the strengths of the machine learning models. Finally, Section 5: Conclusion brings together major themes presented in the paper, a reflection on the study's implications, and recommendations for further research on ransomware detection.

2. Related work

The author in [29] provides an early detection and protection system for ransomware (REDDS) that depends on sequences from the application programming interface (API). Before encrypting files and utilizing the n-gram model and the TF-IDF technique to turn them into feature vectors, REDDS first extracts API sequences from the malware. The API sequences were improved by implementing the Wasserstein GAN with Gradient Penalty (WGAN GP) technique to overcome the constraints of dynamic data collection. Subsequently, the updated data was trained using machine learning classification algorithms

to detect malware. The empirical findings demonstrate that WGAN-GP outperforms other GAN models in enhancing API sequence data. After applying data augmentation techniques, machine learning detection models' accuracy increased and peaked at 99.32%.
Study in [30] aims to provide a comprehensive analysis for selecting the most appropriate behavioral feature set using several feature selection procedures. They pinpointed the similarities and differences in traits that could be common or unique in identifying malware and ransomware. They achieved the most effective set of features for both ransomware and malware, which resulted in an accuracy of 90% for XGBoost and 96.22% for KNN.
Study in [13] uses API calls to analyze seven classifiers inside the malware detection system. In machine learning (ML), parameter optimization is done to get the best possible classification accuracy for binary files, identifying them as malicious or benign. Numerous crucial parameters in machine learning algorithms are checked using API calls. These consist of the learning rate, n-estimators, splitting criteria, tree depth, loss function, kernel function, and k value. This evaluation is done to achieve highly accurate results for the malware classifiers. Finally, the performance of supervised machine learning classification algorithms was evaluated using a dataset consisting of 6434 benign samples and 8634 malware samples. The ensemble methods used in the malware classifier achieved a remarkable accuracy of 99.1%.

The study [31], is exceptionally clear because it aims to successfully classify ransomware, malware, and normal applications using machine learning methods. The source materials used by the researchers included data samples picked from internet sources and a dataset adopted from a prior study in compiling the methodology. Speaking of the tools used to gather the material, they used Cuckoo Sandbox™ to carefully gather samples of ransomware, malware, and other more benign software. Some filter particular feature groups and are used to evaluate and determine the probable usefulness of definite activities or processes of the infection procedure for accurate segregation of ransomware from malware and other harmless applications. The necessary computations in the datasets were performed with Machine Learning (ML) models for binary classification, such as Random Forest, SVM, Gradient Boosting, and Decision Trees. The classifiers that have proven to be most effective for distinguishing between ransomware and benign programs are Support Vector Machine (SVM) and Random Forest. The Random Forest model gave an overall accuracy of 85%, whereas the SVM model gave 86% and 82% accuracy, respectively.

The method proposed for [32] involves using the support vector machine and an approach based on the principal component analysis. To measure the proposed model, the actual result of the model, when cross-checked with the detection of genuine malware, showed satisfactory results, and the decision tree accuracy was found to be 95% percent, making it necessary to employ a new model to predict the outcomes of the selected attributes and prepare the following Experiment 2: 787, and the random accuracy was 99.7%. The following is the hypothesis of the proposed methodology to identify the virus by analyzing the behavioral characteristics. The designed model in the presented study is a combination of support vector machines and principal component analysis. As for the model presented in the current paper, the result achieved an accuracy of 99.5% for authentic malware using the Decision Tree technique, 78.7% using the approach of SVM, and 99.7% utilizing Random Forest.

The research [33] focuses on analyzing ransomware's behavioral characteristics and creating an optimized model with the help of machine learning to differentiate between different types of malware families. The current deficiencies are met by a new method called

Augmented Bootstrapping proposed in the study. This method involves comparing the API calls made by the ransomware and the time these calls were made. It plays the role of orienting in the sequence of API request occurrences and is later employed in constructing the N-gram model. The API calls are then concatenated, and the so-called bigram, trigram, 4-gram, and 5-gram models are created. In addition, the literature will be examined to generate N-gram sequences, and the TF-IDF score will be calculated for each of these sequences; a machine-learning model will be developed using this data. The function analyses the sequences of the API calls from all the running programs within the system. This is confirmed by the analysis when it identifies a ransomware attack occurrence in line with the stated malicious call sequence pattern. The classifier pool used in the study included Gaussian Naive Bayes, Random Forest, K-Nearest Neighbors, Support Vector Machine (SVM), and Logistic Regression. In terms of mean accuracies, these approaches' efficiency was 91%, 89%, 92%, 89% and 89% in that order.

To classify the security level for ransomware detection and prevention, this research [34] proposes a feature selection-based framework that uses several machine learning methods, including neural network-based designs. On a subset of variables for ransomware classification, they used a variety of machine learning methods, including Decision Tree (DT), Random Forest (RF), Naïve Bayes (NB), Logistic Regression (LR), and Neural Network (NN)-based classifiers. To test their suggested approach, they run all the experiments on a single ransomware dataset. The Random Forest classifier surpasses other models by attaining the best accuracy (99%) with a low standard deviation of 0.01 and an F-beta score of 97% with a standard deviation of 0.03. Comparing the DT and NN classifiers against RF, both do reasonably well. Though the accuracy score is reasonable compared to DT, RF, and NN classifiers, LR cannot attain rewarding F-beta.

To preserve 58 API calls specific to both samples, 43 from the ransomware samples and 15 from the benign sample, study [14] used the standard API calls between the malware and benign samples. Additionally, they employed the SVM classifier to get an accuracy of 83.60%, the random forest model to get an accuracy of 86.88%, and the KNN model to acquire a prediction accuracy of 99.18%. They employed grid search to optimize all the parameters in the models they used, noting that this is crucial because it greatly impacts the performance classification model.

This study [35] included Support Vector Machines (SVM), K-Nearest Neighbors (KNN), Naive Bayes, Linear Model, Decision Tree, and Random Forest as its six classification methods. Forty API call features were examined in the experimental setup. With an accuracy of 98.45%, Support Vector Machines (SVMs) outperformed the other techniques. The K-Nearest Neighbors (KNN) algorithm's 97.65% accuracy rate indicates its effectiveness in categorization tasks. Naive Bayes came in third place with an accuracy percentage of 96.50%, proving its reliability in identifying malicious samples from benign ones. In contrast, the Random Forest and Decision Tree classifiers demonstrated competitive performance with an accuracy of 96.10%.

3. METHODOLOGY

In Figure.1, the research methodology is outlined to address threats related to ransomware attacks. This section emphasizes the importance of developing efficient detection strategies in response to the ongoing advancements in post-infection cleanup procedures. In the early stages, the methodology offers a framework that outlines a proper way of developing machine learning models for ransomware detection based on four distinct machine learning models in the research study that is discussed in the paper, each of which provides an individual means of differentiating between ransomware and non-ransomware instances. The

process includes several essential components: The specific research areas include data acquisition, data pre-processing, feature creation and scaling techniques, data analysis, classification techniques, and performance metrics. The study aims to develop a robust foundation for proactive solutions, enhancing cybersecurity resilience and effectively addressing the ever-changing threat landscape posed by ransomware attacks.

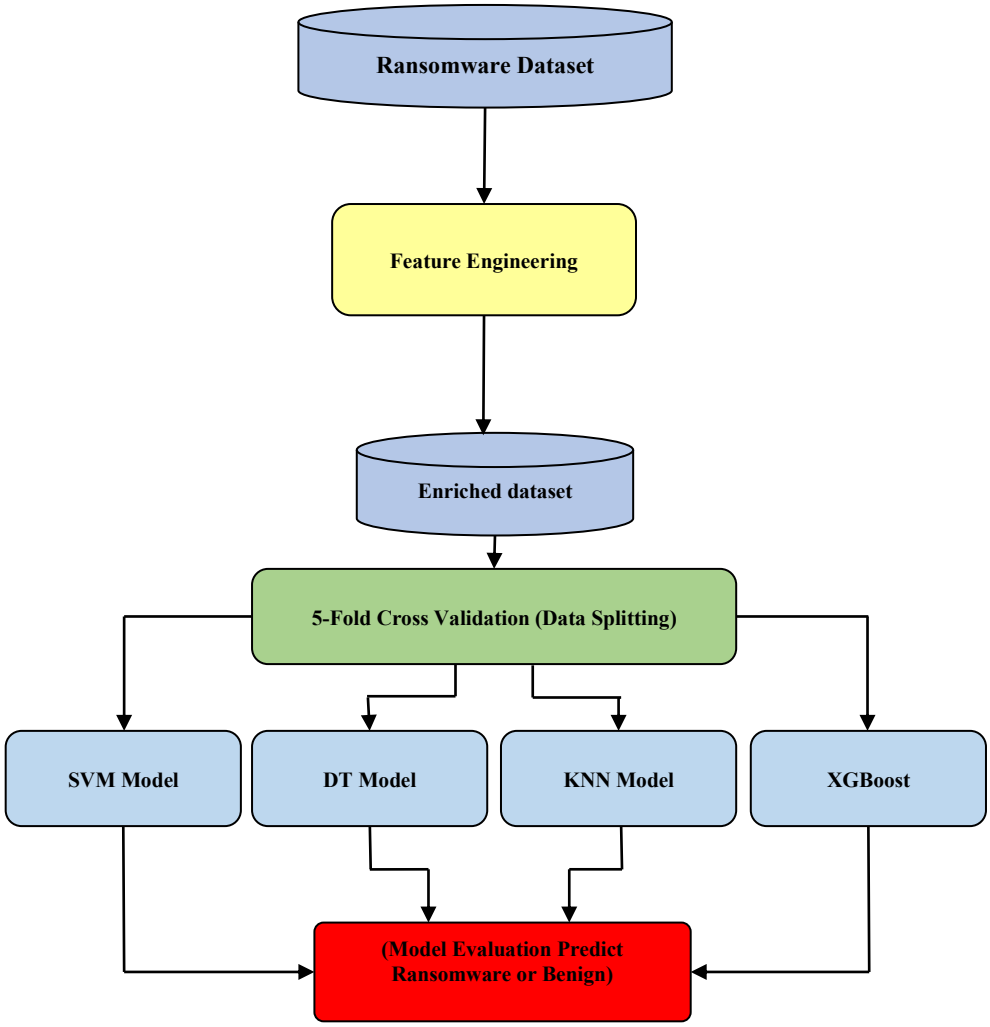

Figure 1: Proposed System Block Diagram

3.1 Dataset

This study utilizes a comprehensive and current balanced dataset comprising 2,311 samples, categorized into 1,200 ransomware and 1,111 benign instances derived from 13 families ("Cerber", "CryptoWall", "Matsnu", "Shade", "Teslacrypt", "Benign", "Hive", "Ako", "Erica", "Conti", "Matrix", "Gandcrab", "Expiro") [36]. The dataset comprises a single benign family that was identified via dynamic analysis. This identification process involved studying more than 11,000 ransomware samples and 1,200 benign samples from 23 distinct families. It represents the most up-to-date collection of its kind, providing a robust foundation for analyzing API call pattern characteristics that encompass 48 unique features, As listed in Table 1 [36] and their roles within the samples.

The analysis and the prediction concern the binary two-class of the dataset. The variety seeks to separate samples into those that are malicious ransomware and those that are non-malicious. These two classes incorporate almost all API request patterns. This dataset is

valuable for cybersecurity or ransomware identification because the comprehensive categorization means different families of threats behave differently, which helps identify possible threats.

Including a non-malicious family in the sample contributes to understanding API request patterns in normal and unsafe conditions. The dataset's content is not typical, making it possible to build machine models for task detection and enhance protection against possible cyber threats.

Combining the binary data with the API call data and the target variable is useful in enhancing analysis, identification, and pattern recognition. These are among the factors that help improve cybersecurity systems and detect ransomware. This dataset is useful for scholars and practitioners to understand how API calls work and how security risks can be detected and addressed.

Table 1: Evasion APIs

Evasion API	Description
MoveFileWithProgressW	File and directory movements comprise the children of the particular files and directories.
NtCreateFile	Creating new files or directories involves opening files and operations, which involve creating new files or directories.
NtWriteFile	Open new files by writing data
SetFileAttributesW	Sets directory or file attributes
GetDiskFreeSpaceExW	Retrieve available disk space information
GetDiskFreeSpaceW	Gets designated disk information
ShellExecuteExW	Perform operations on a designated file
DeviceIoControl	Delivers a control code to a device driver
DeviceIoControl	Obtains the hostname of the local machine
NtQuerySystemInformation	Retrieves system information
GlobalMemoryStatusEx	Retrieves system memory usage info
NtAllocateVirtualMemory	Saves a range of user-mode virtual address space pages
NtMapViewOfSection	Maps a specified segment of a Section Object into the memory space of a process
NtProtectVirtualMemory	Alters the security configurations for a designated section of assigned memory pages
NtUnmapViewOfSection	Deallocates a mapping of a segment from the virtual address space
WriteProcessMemory	Stores data in a specified memory segment of a specific process
LdrGetDllHandle	Loads a file into the computer's random-access memory (RAM)
GetAdaptersAddresses	Obtains the addresses linked to the adapters.
InternetOpenA	Launches WinINet functionalities by an application.
CreateProcessInternalW	Instantiates a fresh process along with its principal thread.
NtGetContextThread	Assign the user-mode context to the given thread.
NtResumeThread	Map the designated portion of the Section Object into the process memory.
NtSetContextThread	Assigning the user-mode context to the given thread.
NtTerminateProcess	ends a process along with all of its threads.
Process32NextW	Data regarding the subsequent procedure is documented in a snapshot.
NtLoadDriver	Drivers are installed in the system.
NtSetValueKey	Creates a new process and its primary thread
RegOpenKeyExW	Returns the user-mode context of the specified thread
RegQueryValueExW	Map specified part of Section Object into process memory

RegSetValueExW	Set the user-mode context of the specified thread
NtCreateKey	Creates or updates the value entry of a registry key
CryptGenKey	Accesses the designated registry key
CryptExportKey	Creates a fresh registry key or accesses an already existing one
LookupPrivilegeValueW	Obtains the identification used to represent the privileged name provided
CryptHashData	Appends information to a designated hash entity.
CreateServiceW	Instantiates a service object and includes it in the designated service manager
EnumServicesStatusW	lists the services stored in the designated service control management database.
SetWindowsHookExW	Attaches an application-defined hook operation to a chain of hooks
FindWindowW	Retrieves a reference to the highest-level window
CreateServiceA	install and register a new service in the Windows operating system
EnumServicesStatusA	used to retrieve information about the services currently running on the system
FindWindowA	it helps locate a window on the desktop by searching for its class name and window title

3.2 Feature Engineering

Data preparation refers to the methodologies and processes used to convert unprocessed data into a suitable format for processing or utilization in machine learning algorithms. The data preprocessing stage involves several actions to prepare the documents adequately. Text pre-processing is an essential stage in the classification process because it allows for the elimination of unnecessary words from documents, which often hinders the speed, accuracy, and efficiency of the classification process. The method of preprocessing documents used in this study consists of four steps: tokenization, elimination of stop words, stemming, and normalization. Those steps are crucial for building a text dataset that is clean, standardized, and meaningful. These methods will enhance the performance and interpretability of machine learning models when applied to text data [23].

3.2.1 Standard Scaling

Standard scaling, or z-score normalization, is a method of modifying numerical features by adjusting them to have an average of 0 and a standard deviation of 1. This strategy guarantees that the features are normalized to a comparable size, features with greater scales to prevent the learning algorithms from being overpowered. The mathematical formulation of Standard Scaling is provided in Equation (1):

$$\text{Standardized Value } (z) = \frac{(x-\mu)}{\sigma} \quad (1)$$

where x is the original value, μ is the mean, and σ is the standard deviation [37].

3.2.2 New Statistical Features

Six new statistical features were introduced for each sample to further augment the dataset's information content. These features, calculated individually for every sample, encompass quantiles at 0.25, 0.5, and 0.75, providing insights into the distribution characteristics of the API call features. Additionally, the dataset now includes three moment-based statistical features: skewness, kurtosis, and entropy.

- Quantiles:

Quantiles divide a dataset into equal parts or intervals. The quantiles at 0.25, 0.5, and 0.75 represent the values below which 25%, 50%, and 75% of the data fall, respectively. The mathematical formulation of quantiles is provided in Equation (2):

$$\text{Quantile}(q) = F^{-1}(q) \quad (2)$$

where F is the cumulative distribution function.
q is the probability value used to find the corresponding quantile [38] .

- Entropy:

Entropy quantifies the level of unpredictability or lack of organization within a dataset. The entropy of a probability distribution is computed using the given Equation (3):

$$H(X) = -\sum_{i=1}^{n} P(x_i) \log_2 P(x_i) \quad (3)$$

Where H(X) is the entropy of the random variable X and P(x) Is the probability of each possible outcome [39].

- Kurtosis:

Kurtosis measures the degree to which a probability distribution differs from a normal distribution in terms of the thickness of its tails. A positive kurtosis indicates a distribution with a higher concentration of extreme values and a more prominent peak. In contrast, a negative kurtosis suggests a distribution with fewer extreme values and a less prominent peak. The mathematical formulation of kurtosis is provided in Equation (4):

$$\text{Kurtosis}(X) = \frac{E[(X-\mu)^4]}{\sigma^4} \quad (4)$$

where E stands for the expected value, μ is the mean and σ is the standard deviation [40].

- Skewness:

The measure of skewness implies the asymmetry of a percentage distribution. Skewness is considered appropriate in a right-skewed distribution. On the other hand, if the skewness coefficient is less than zero, the distribution is regarded as negatively skewed, whereas if the coefficient equals zero, this is an indication that the distribution is perfect. The mathematical formulation of skewness is provided in Equation (5): [41].

$$\text{Skewness}(X) = \frac{E[(X-\mu)^3]}{\sigma^3} \quad (5)$$

These new elements enhance the identification of the API call patterns' characteristics even more by indicating their average values and variability around the mean values and their deviation from other samples in the context of the central tendencies' measures. The data is very refined and prepared with the intent of improving the current dataset used to train the machine learning models.

3.3 Classification

During analysis, four ML algorithms such as SVM, Decision Tree, KNN, and XGBoot, were used to classify the dataset. The dataset was expanded by incorporating new statistical characteristics and then divided into two subsets: an 80% sample, which is used for training the model, and 20% used for testing the model. Every model was trained on the training subset to help the model discover patterns and relationships in the features. Feature engineering is the process of data preprocessing that is critical to enhancing the available dataset. Due to the over/underfitting problem, the dataset adopted a balanced design, and the features were engineered skillfully with cross-validation. During the evaluation and selection of the models, cross-validation was used, thus each classifier was tested using data it had not used to get trained on, which made the models more reliable. Thus, the equal involvement of synthesized raw and safe samples in the dataset guaranteed that the models provided reliable performance scores. This significantly improved the accuracy of the models during testing, enabling them to classify new discrete data examples not used during training and perform categorization by applying the learned patterns.

3.3.1 Decision trees

A decision tree is a widely used machine learning model designed for classification by predicting target variables based on training data. It follows a straightforward, rule-based

approach to partition data. Among its variants, C4.5 is a powerful classifier in data mining, constructing decision trees from training datasets. It determines attribute importance using gain ratio and information gain and follows four key phases [13] [42]:
I. Select an attribute as the root.
II. Generate a branch for each value.
III. Place the dataset in the branch.
IV. Iterate the second process until all classes have equal values.
The formulas utilized in C4.5 are displayed here:

$$\text{Entropy (S)} \sum_{i=1}^{n} -\text{Pi} * \log_2 \text{Pi} \quad (6)$$

S: Entropy,
P: The proportion of classes in the output.

$$\text{Gain(S, A)} = \text{Entropy (S)} \sum_{i=1}^{n} \frac{|\text{Si}|}{|\text{S}|} * \text{Entropy (S)} \quad (7)$$

S: Compilation of instances.
A: Attribute of A case or problem.
|Si|: Denotes the numerical value of instances for I.
|S|: represents the number of cases in the set.

3.3.2 K-Nearest Neighbor (KNN)

KNN is a non-parametric, instance-based machine learning algorithm that classifies a target instance based on the majority vote of its k-nearest neighbors. Unlike traditional models, it retains all training instances rather than constructing a comprehensive model. A related approach, the Radius-Neighbor classifier, categorizes instances based on neighboring data points within a specified radius instead of a fixed k value. The choice of k in KNN significantly depends on the characteristics of the training dataset. KNN distance computations predominantly employ the Euclidean distance, whose formula is as follows [43]:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i + y_i)^2} \quad (8)$$

Where d is the Euclidean distance, X denotes a distinct datum extracted from the dataset. n denotes the overall quantity of dimensions, and Y denotes a predicted data point.

3.3.3 Support Vector Machine (SVM)

Support vector machines developed by Cortes and Vapnik in 1995 are used in binary classification. The support vector machine is a highly visual method of classification and regression that is used to solve classification problems. SVM can be represented mathematically by [13] [44]:

$$\hat{\text{y}} = w^T \text{x} + \text{b} \quad (9)$$

Where w is the normal vector to the hyperplane. b is the hyperplane's distance along normal vector w from the origin. x is the input feature.

3.3.4 XGBoost

XGBoost, also known as eXtreme Gradient Boosting, is a highly resilient machine learning algorithm that is widely used for problems involving supervised learning. It builds an ensemble of weak learners, typically decision trees, to form a strong predictive model. The mathematical formulation of XGBoost involves an objective function that combines the loss function and a regularization term. The general model equation can be expressed as:

$$\mathcal{L}(\Theta) = \sum_{i=1}^{n} \text{l}(\hat{y_i}, \text{y}_i) + \sum_{k=1}^{K} \Omega(\text{f}_k) \quad (10)$$

Here, $l(\widehat{y_i}, y_i)$ is a differentiable convex loss function measuring the difference between the predicted value $\widehat{y_i}$ and the actual target y_i , and $\Omega(f_k)$ is the regularization term that prevents overfitting [45, 46].

3.4 Evaluation Metric

Throughout the code evaluation, crucial metrics were employed to comprehensively examine the performance of the models on previously unseen testing data [47, 48] . The metrics include the following:

- Accuracy: Calculated by dividing the number of correctly predicted samples by the total number of samples. Mathematically expressed as Equation (11):

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \quad (11)$$

Correctly identified positive occurrences are TP, correctly identified negative examples are TN, wrongly identified positive instances are FP, and incorrectly identified negative instances are FN. Overall accuracy refers to the number of correctly identified objects across all categories.

- Precision refers to a model's ability to correctly identify positive cases out of the total number of predicted positives. Mathematically expressed as Equation (12):

$$\text{Precision} = \frac{TP}{TP+FP} \quad (12)$$

Precision is especially important in situations where the consequences of incorrect positive results are substantial.

- Recall (Sensitivity): Recall quantifies the model's ability to accurately detect all positive instances within the actual positive samples. Mathematically expressed as Equation (13):

$$\text{Recall} = \frac{TP}{TP+FN} \quad (13)$$

Recall is crucial in scenarios where missing positive instances is a critical concern.

- F1 Score: The F1 score is a statistic that combines precision and recall by calculating their harmonic mean. Mathematically expressed as Equation (14):

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision}+\text{Recall}} \quad (14)$$

The F1 score is particularly valuable when there is a requirement to strike a balance between precision and recall during the evaluation process.

These evaluation indicators thoroughly assess the model's performance, considering factors like general precision, recall, accuracy, and a balanced measure using the F1 score.

- Confusion matrix: A confusion matrix is a tool used to measure the performance of machine learning classification tasks. The visual representation of the classification model performance juxtaposes the predicted and the actual classification.

Table 2: Confusion Matrix

	Predicted Positive Class	Predicted Negative Class
Actual Positive Class	True positive TP	False negative FN
Actual Negative Class	False positive FP	True negative TN

4. Experimental Results and Discussion

4.1 Dataset Enrichment

The dataset was enhanced through a sequence of pre-processing procedures and feature engineering methodologies. Initially, scaling was applied to ensure that all characteristics were normalized to a consistent scale. This step is essential for the effective operation of various machine learning methods, such as KNN, which is a distance-based algorithm where the distance between data points heavily influences classification decisions. Therefore,

scaling of features, including binary ones, is crucial. Without normalization, features with a broad range of numerical values may dominate the distance measure, leading to biased outcomes. Finally, additional statistical characteristics were incorporated to further enhance the dataset. As a result, the dataset contained 54 features and 2311 samples, whereas the dataset referenced in [36] included 48 features and 2311 samples. This augmentation aimed to increase the informational content and refine the data in a way that improved pattern recognition and predictive accuracy.

4.2 Result of the Classification Phase

Based on the experiment, the utilization of the improved dataset was able to display peculiarly high performance of four models, including SVM, KNN, Decision Tree, and XGBoost, in ransomware prediction. SVM achieved 99.5% accuracy by utilizing the method's ability to separate classes. The accuracy of KNN was 99.7%. It used a neighborhood approach and stood to gain from better data. The Decision Tree achieved a 99.7% result, leveraging its decision tree structure to provide a method for engaging data. Lastly, XGBoost achieved the highest accuracy of 99.9%, capitalizing on the strength of small tree decision-making. In comparison to the results before the use of feature engineering techniques in terms of accuracy —SVM attained 90.5%, KNN accomplished 91%, Decision Tree secured 91.5%, and XGBoost achieved 92.8%, it is evident that the feature engineering approaches markedly enhanced the performance of all models. These results reflect the power and efficiency these models can provide in ransomware detection after applying the proposed feature engineering pipeline and confirm the effectiveness and reliability of the models adopted in the study based on a comprehensive analysis of the various evaluation scales used. Table 3 and Table 4 Shows classifier performance before and after feature engineering

Table 3: Classifiers Performance Before Applying Feature Engineering.

Classifiers	Accuracy%	Recall %	Precision%	F1 Score %
SVM	90.5	90.3	90.7	90.5
KNN	91	90.9	91.2	91.1
DT	91.5	91.7	91.2	91.4
XGBoost	92.8	93	92.6	92.8

Table 4: Classifiers Performance After Applying Feature Engineering.

Classifiers	Accuracy%	Recall %	Precision%	F1 Score %
SVM	99.5	99.5	99.5	99.5
KNN	99.7	99.7	99.7	99.7
DT	99.7	99.9	99.5	99.7
XGBoost	99.9	99.9	99.9	99.9

These measurements demonstrate how well the models can work to identify these patterns and how to make the right predictions to prove how practical and efficient our approach is.
The enhancement of accuracy regarding our methodological approach demonstrate that the proposed solution works effectively by using all four models of machine learning algorithms for binary classification. Thus, by generating the models to detect the detailed patterns of data, advancing the number of statistical features, and having a better pre-processing technique, the models became capable of giving precise predictions. The confusion matrices for each of the machine learning models used (SVM, KNN, Decision Tree, and XGBoost) are illustrated in Figure 2, displaying true positives, false positives, true negatives, and false negatives for each machine learning model after feature engineering is applied.

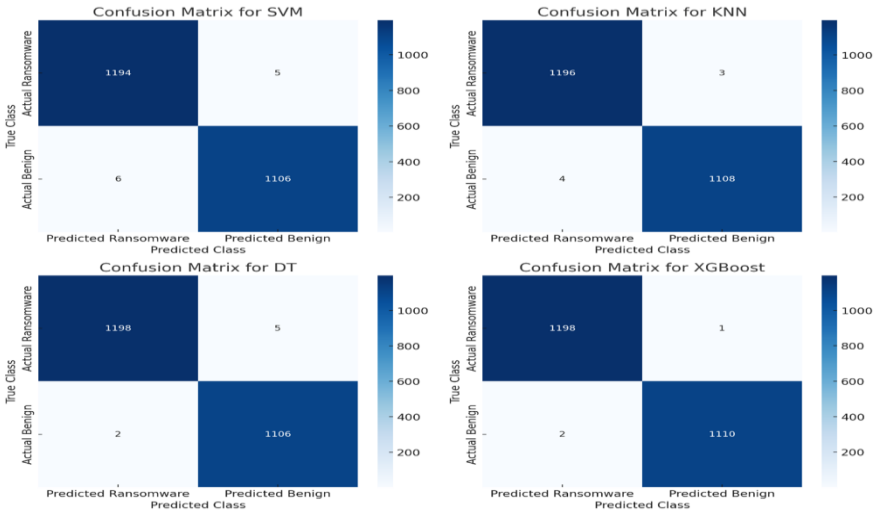

Figure 2: All Models Confusion matrices

4.3 Performance Comparison and Generalization of the Proposed Methodology

To compare the proposed methodology with other studies in the literature, Figure 3 illustrates the comparative performance of the XGBoost and KNN models on a recent dataset [30] versus the methodology that integrates advanced feature engineering techniques for ransomware detection. The figure highlights how the proposed approach significantly enhances the performance of these machine learning models. In the previous study [30]The XGBoost and KNN models achieved 90% accuracy for XGBoost and 96.22% accuracy for KNN, respectively. These results were based on one of the most recent and challenging datasets in ransomware detection. However, by applying the advanced feature engineering methodology to the same dataset, considerable improvements were achieved: 99.3% accuracy for XGBoost and 98.5% accuracy for KNN. The results demonstrate that the methodology generalizes well across different machine learning algorithms, enhancing model performance and making this approach highly efficient in detecting ransomware. This improvement indicates that the advanced feature engineering process is crucial for extracting meaningful information, allowing the models to detect ransomware more accurately and robustly compared to previous approaches.

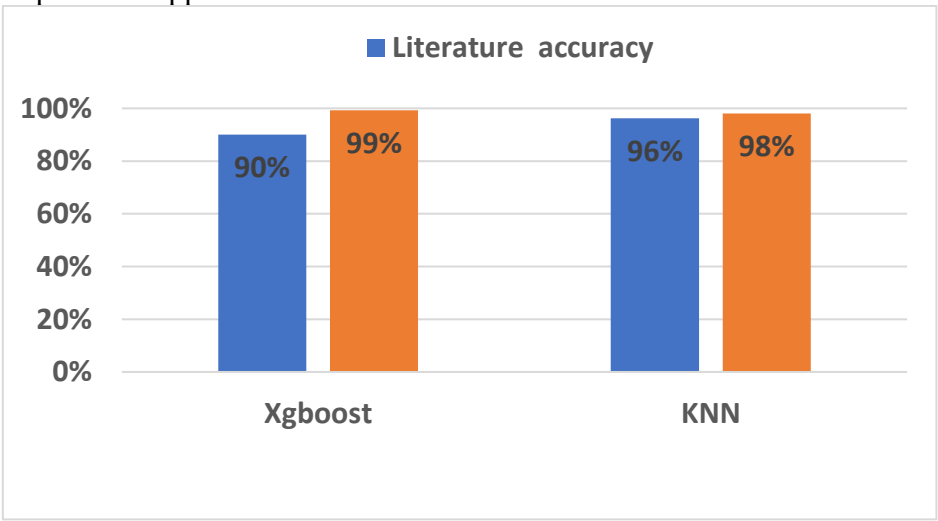

Figure 3: Performance Comparison with Literature Result

4.4 Comparison with Existing Literature

The proposed methodology requires comparison to recent studies to prove its

effectiveness. Previous research on ransomware detection employed different machine learning models and datasets. The proposed approach achieves better model performance through its combination of feature engineering with API call functionalities.
The results presented in Table 5 demonstrate better performance than recent studies, which validates the developed framework's reliability and efficiency.

Table 5: Performance Comparison of Ransomware Detection Models

Study/ Source	Classifier	Accuracy
Our proposed models	SVM	99.5%
	KNN	99.7%
	Decision Tree	99.7%
	XGBoost	**99.9%**
Singh and Singh, 2020	Ensemble Methods	99.1%
Almousa et al., 2021	SVM	83.6%
	Random Forest	86.88%
	KNN	99.18%
Hwang et al., 2022	SVM	98.45%
	KNN	97.65%
	Naive Bayes	96.5%
	Random Forest	96.1%
	Decision Tree	96.1%
Masum et al., 2022	Random Forest	99%
Sukul et al., 2022	Gaussian NB	91%
	Random Forest	89%
	KNN	92%
	SVM	89%
	Logistic Regression	89%
Zhang et al., 2023	WGAN-GP + ML	99.32%
Schoenbachler et al., 2023	Random Forest	85%
	SVM	86%
Rahman et al., 2024	Decision Tree	99.5%
	SVM	78.7%
	Random Forest	99.7%
Bhagwat and Patil, 2024	XGBoost	90%
	KNN	96.22%

5. Conclusion

To summarize, ransomware poses a significant risk when using computers, as it can encrypt user data or restrict access to computer systems, demanding payment in exchange for recovery. Prior research has inadequately addressed this subject, resulting in numerous unresolved issues. Hence, this study investigates and assesses this phenomenon to mitigate the challenges encountered by computer users. The shortcoming identified in previous research was addressed by examining the utilization of API calls from the most recent and comprehensive dataset available, as referenced in this paper, for ransomware detection. The findings indicate that the models used on this dataset achieved 99.5% accuracy in SVM, 99.7% in KNN, 99.7% in the decision tree, and 99.9% in XGBoost, demonstrating high precision in identifying ransomware. Additionally, to validate the generalization capability of the proposed approach, the feature engineering technique was applied to another recent dataset. This resulted in improved model performance, with XGBoost achieving 99% accuracy and KNN reaching 98%, significantly outperforming a previous study that reported 90% and 91% accuracy for the same models, respectively. These results confirm the effectiveness of the methodology across different datasets and scenarios. In terms of

performance evaluation, the models have demonstrated high efficiency across multiple criteria, further validating the suitability of the proposed approach. This strategy highlights the ability of models to identify patterns and make accurate predictions, thereby enhancing the overall performance of the study.

Future work could focus on integrating multi-class classification to improve efficiency and accuracy. Additionally, using hybrid machine learning models and exploring different modern datasets could further enhance the robustness and adaptability of the detection system. The dataset used remains open to further investigation and discussion in future studies. Addressing these factors and strengthening defense measures against ransomware attacks will contribute to safeguarding computer systems and user data.

References:

[1] A. Vardalaki and V. Vlachos, "Emerging malware threats: The case of ransomware," in *Cybersecurity Issues in Emerging Technologies*, 1st ed. Boca Raton, FL, USA: CRC Press, 2021, pp. 153-170.

[2] M. Keshavarzi and H. R. Ghaffary, "I2CE3: A dedicated and separated attack chain for ransomware offenses as the most infamous cyber extortion," *Comput. Sci. Rev.*, vol. 36, p. 100233, May 2020, doi: 10.1016/j.cosrev.2020.100233.

[3] S. R. Matthijsse, M. S. van ‘t Hoff-de Goede, and E. R. Leukfeldt, "Your files have been encrypted: A crime script analysis of ransomware attacks," *Trends Organ. Crime*, pp. 1-27, Apr. 27, 2023, doi: 10.1007/s12117-023-09496-z.

[4] V. Vasani et al., "Comprehensive analysis of advanced techniques and vital tools for detecting malware intrusion," *Electronics*, vol. 12, no. 20, p. 4299, Oct. 17, 2023, doi: 10.3390/electronics12204299.

[5] A. Minnaar and F. J. Herbig, "Cyberattacks and the cybercrime threat of ransomware to hospitals and healthcare services during the COVID-19 pandemic," *Acta Criminol. Afr. J. Criminol. Victimol.*, vol. 34, no. 3, pp. 155-185, Dec. 1, 2021, doi: 10.10520/ejc-crim_v34_n3_a10.

[6] N. Pattnaik et al., "It’s more than just money: The real-world harms from ransomware attacks," in *Proc. Int. Symp. Human Aspects Inf. Security Assurance*, Jul. 26, 2023, vol. 674, pp. 261-274, doi: 10.1007/978-3-031-38530-8_21.

[7] N. A. Hassan, "Introduction: Understanding digital forensics," in *Digital Forensics Basics: A Practical Guide Using Windows OS*, Cham, Switzerland: Springer, 2019, pp. 1-33, doi: 10.1007/978-1-4842-3838-7_1.

[8] M. Ryan, *Ransomware Revolution: The Rise of a Prodigious Cyber Threat*, vol. 85. Berlin, Germany: Springer, Feb. 24, 2021.

[9] T. Reshmi, "Information security breaches due to ransomware attacks—a systematic literature review," *Int. J. Inf. Manage. Data Insights*, vol. 1, no. 2, p. 100013, Apr. 15, 2021, doi: 10.1016/j.jjimei.2021.100013.

[10] S. Razaulla et al., "The age of ransomware: A survey on the evolution, taxonomy, and research directions," *IEEE Access*, vol. 11, pp. 40698-40723, Apr. 19, 2023, doi: 10.1109/ACCESS.2023.3268535.

[11] R. Chaganti, V. Ravi, and T. D. Pham, "A multi-view feature fusion approach for effective malware classification using deep learning," *J. Inf. Security Appl.*, vol. 72, p. 103402, Dec. 14, 2022, doi: 10.1016/j.jisa.2022.103402.

[12] G. Balan, "Using sequences of API calls to identify and classify ransomware families," in *Proc. 2023 25th Int. Symp. Symbolic Numeric Algorithms Sci. Comput. (SYNASC)*, Sep. 14, 2023, pp. 178-185, doi: 10.1109/SYNASC61333.2023.00031.

[13] J. Singh and J. Singh, "Assessment of supervised machine learning algorithms using dynamic API calls for malware detection," *Int. J. Comput. Appl.*, vol. 44, no. 3, pp. 270-277, Feb. 26, 2020, doi: 10.1080/1206212X.2020.1732641.

[14] M. Almousa, S. Basavaraju, and M. Anwar, "API-based ransomware detection using machine learning-based threat detection models," in *Proc. 2021 18th Int. Conf. Privacy, Security, Trust (PST)*, Dec. 15, 2021, pp. 1-7, doi: 10.1109/PST52912.2021.9647816.

[15] A. Alraizza, A. Algarni, and C. Computing, "Ransomware detection using machine learning: A survey," *Big Data Cogn. Comput.*, vol. 7, no. 3, p. 143, Aug. 16, 2023, doi: 10.3390/bdcc7030143.

[16] K. S. Kader et al., "Ransomware detection using binary classification," in *Proc. 2021 IEEE Int. Conf. Dependable, Autonomic Secure Comput. (DASC)*, Oct. 28, 2021, pp. 979-984, doi: 10.1109/DASC-PICom-CBDCom-CyberSciTech52372.2021.00163.

[17] S. Naz and D. K. Singh, "Review of machine learning methods for Windows malware detection," in *Proc. 2019 10th Int. Conf. Comput., Commun. Netw. Technol. (ICCCNT)*, Jul. 6-8, 2019, pp. 1-6, doi: 10.1109/ICCCNT45670.2019.8944796.

[18] M. Wadkar, F. Di Troia, and M. Stamp, "Detecting malware evolution using support vector machines," *Expert Syst. Appl.*, vol. 143, p. 113022, Apr. 1, 2020, doi: 10.1016/j.eswa.2019.113022.

[19] M. M. Rahma and A. D. Salman, "Heart disease classification–based on the best machine learning model," *Iraqi Journal of Science*, vol. 63, pp. 3966-3976, Sep. 30, 2022, doi: 10.24996/ijs.2022.63.9.28.

[20] S. Lakshminarayana and P. I. Basarkod, "Unification of K-nearest neighbor (KNN) with distance aware algorithm for intrusion detection in evolving networks like IoT," *Wireless Pers. Commun*., vol. 132, no. 3, pp. 2255-2281, Sep. 11, 2023, doi: 10.1007/s11277-023-10722-8.

[21] A. P. Singh, "Analysis of variants of KNN algorithm based on preprocessing techniques," in *2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN)*, Oct. 12-13, 2018, IEEE, pp. 186-191, doi: 10.1109/ICACCCN.2018.8748429.

[22] D. Lopez-Bernal, D. Balderas, P. Ponce, and A. Molina, "Education 4.0: Teaching the basics of KNN, LDA and simple perceptron algorithms for binary classification problems," *Future Internet*, vol. 13, no. 8, p. 193, Jul. 27, 2021, doi: 10.3390/fi13080193.

[23] S. Zhang, X. Li, M. Zong, X. Zhu, R. Wang, and L. Systems, "Efficient kNN classification with different numbers of nearest neighbors," *IEEE Transactions on Neural Networks*, vol. 29, no. 5, pp. 1774-1785, Apr. 12, 2017, doi: 10.1109/TNNLS.2017.2673241.

[24] A. Yeboah-Ofori, "Classification of malware attacks using machine learning in decision tree," *International Journal of Security*, vol. 11, no. 2, pp. 10-25, Aug. 2020.

[25] S. Judy and R. Khilar, "Detection and classification of malware for cybersecurity using machine learning algorithms," in *2023 Eighth International Conference on Science Technology Engineering and Mathematics (ICONSTEM)*, Apr. 6-7, 2023, IEEE, pp. 1-6, doi: 10.1109/ICONSTEM56934.2023.10142575.

[26] S. Choudhary and A. Sharma, "Malware detection & classification using machine learning," in *2020 International Conference on Emerging Trends in Communication, Control and Computing (ICONC3)*, Feb. 21-22, 2020, IEEE, pp. 1-4, doi: 10.1109/ICONC345789.2020.9117547.

[27] Q. Abu Al-Haija, A. Odeh, and H. Qattous, "PDF malware detection based on optimizable decision trees," *Electronics*, vol. 11, no. 19, p. 3142, Sep. 30, 2022, doi: 10.3390/electronics11193142.

[28] Q. H. Vu, D. Ruta, and L. Cen, "Gradient boosting decision trees for cybersecurity threats detection based on network event logs," in *2019 IEEE International Conference on Big Data (Big Data)*, Dec. 9-12, 2019, IEEE, pp. 5921-5928, doi: 10.1109/BigData47090.2019.9006061.

[29] S. Zhang, T. Du, P. Shi, X. Su, and Y. Han, "Early detection and defense countermeasure inference of ransomware based on API sequence," *International Journal of Advanced Computer Science Applications*, vol. 14, no. 10, 2023, doi: 10.14569/IJACSA.2023.0141067.

[30] L. B. Bhagwat and B. M. Patil, "Behavioral analysis and results of malware and ransomware using optimal behavioral feature set," *International Journal of Information Computer Security*, vol. 23, no. 1, pp. 57-78, Feb. 19, 2024, doi: 10.1504/IJICS.2024.136719.

[31] J. Schoenbachler, V. Krishnan, G. Agarwal, and F. Li, "Sorting ransomware from malware utilizing machine learning methods with dynamic analysis," in *Proceedings of the Twenty-Fourth International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*, Oct. 16, 2023, pp. 516-521, doi: 10.1145/3565287.3617632.

[32] M. S. Rahman, M. S. A. Sabbir, and S. Ghosh, "Ransomware attack detection using machine learning approaches," in *2024 3rd International Conference for Innovation in Technology (INOCON)*, Mar. 1-3, 2024, IEEE, pp. 1-7, doi: 10.1109/INOCON60754.2024.10512276.

[33] M. Sukul, S. A. Lakshmanan, and R. Gowtham, "Automated dynamic detection of ransomware using augmented bootstrapping," in *2022 6th International Conference on Trends in Electronics and Informatics (ICOEI)*, Apr. 28-30, 2022, IEEE, pp. 787-794, doi: 10.1109/ICOEI53556.2022.9777099.

[34] M. Masum et al., "Ransomware classification and detection with machine learning algorithms," in *2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*, Jan. 26-29, 2022, IEEE, pp. 0316-0322, doi: 10.1109/CCWC54503.2022.9720869.

[35] J. Hwang, J. Kim, S. Lee, and K. Kim, "Two-stage ransomware detection using dynamic analysis and machine learning techniques," *Wireless Personal Communications*, vol. 112, no. 4, pp. 2597-2609, Jan. 2022, doi: 10.1109/CCWC54503.2022.9720869.

[36] F. Coglio, A. Lekssays, B. Carminati, and E. Ferrari, "Early-stage ransomware detection based on pre-attack internal API calls," in *International Conference on Advanced Information Networking and Applications*, Mar. 15, 2023, Springer, pp. 417-429, doi: 10.1007/978-3-031-28451-9_36.

[37] A. C. Müller, *Introduction to Machine Learning with Python: A Guide for Data Scientists*. O'Reilly Media, 2017.

[38] R. S. Witte and J. S. Witte, *Statistics*. John Wiley & Sons, Jan. 5, 2017.

[39] D. J. MacKay, *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, Sep. 25, 2003.

[40] P. Newbold, W. L. Carlson, and B. M. Thorne, *Statistics for Business and Economics. Pearson*, 2013.

[41] R. E. Walpole, R. H. Myers, S. L. Myers, and K. Ye, *Probability and Statistics for Engineers and Scientists*, vol. 5. Macmillan, Jan. 1993.

[42] R. Elhassan and M. Ahmed, "Arabic text classification review," *Evaluation*, vol. 4, no. 1, pp. 1-5, Jan. 2015.

[43] F. Liu, C. He, and Z. Chang, "An improved local weighted mean-based k-nearest neighbor classifier," in *2023 IEEE 6th International Conference on Automation, Electronics and Electrical Engineering (AUTEEE)*, Dec. 15-17, 2023, IEEE, pp. 832-837, doi: 10.1109/AUTEEE60196.2023.10408698.

[44] N. Patel, B. Mehtre, and R. Wankar, "Detection of intrusions using support vector machines and deep neural networks," in *2022 10th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO)*, Oct. 13-14, 2022, IEEE, pp. 1-5, doi: 10.1109/ICRITO56286.2022.9964756.

[45] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug. 13, 2016, pp. 785-794, doi: 10.1145/2939672.2939785.

[46] A. A. Abdualrahman and M. K. Ibrahem, "Intrusion detection system using data stream classification," *Iraqi Journal of Science*, vol. 62, pp. 319-328, Jan. 30, 2021, doi: 10.24996/ijs.2021.62.1.30.

[47] A. R. Mahmood and S. M. Hameed, "A Smishing detection method based on SMS contents analysis and URL inspection using Google Engine and VirusTotal," *Iraqi Journal of Science*, vol. 64, pp. 5376-5391, Oct. 30, 2023, doi: 10.24996/ijs.2023.64.10.41.

[48] H. S. Abdullah, N. H. Ali, and N. A. Abdullah, "Evaluating the performance and behavior of CNN, LSTM, and GRU for classification and prediction tasks," *Iraqi Journal of Science*, vol. 65, no. 3, pp. 1741-1751, Mar. 1, 2024, doi: 10.24996/ijs.2024.65.3.43.