International Journal of Advanced Smart Convergence Vol.13 No.4 68- 81(2024)
http://dx.doi.org/10.7236/IJASC.2024.13.4.68

IJASC 24-4-8

Construction and Evaluation of Custom Cybersecurity AI Dataset for Ransomware Detection Using Machine Learning

Niringiye Godfrey[1], Bruce Ndibanje[2], HoonJae Lee[3*], ByungGook Lee[4]

[1]*Department of Computer Engineering, Dongseo University*
[2]*Cybersecurity Consultant, TechDivision, Rome, Italy*
[3]*Professor, Dept. of Information Security, Dongseo University*
[4]*Professor, Dept. of Computer Engineering, Dongseo University*

E-mail godfreyte1@gmail.com, bruce.ndibanje@wfp.org, hjlee@dongseo.ac.kr

Abstract

Ransomware is one of the most significant cybersecurity threats facing the world. In this research we designed and constructed a custom cybersecurity AI dataset for ransomware detection. We then evaluated the dataset using different machine learning models. The dataset was constructed using Cuckoo Sandbox where raw ransomware samples were analyzed to extract key features such as API calls, DLL usage, file operations, network activity, process creation and registry changes. These were then carefully labeled as either ransomware or benign. For evaluation purposes, the custom cybersecurity AI dataset was utilized to train and test various machine learning models. The dataset was split into 80% for training and 20% for testing. Logistic Regression, Random Forest, K-Nearest Neighbors (KNN), and XGBoost models were used to evaluate the resulting custom Cybersecurity AI Dataset. We obtained higher results of accuracy, precision, recall, and F1 scores evaluation metrics. Moreover, our results demonstrate the robustness of a combination of well-designed custom Cybersecurity AI Datasets and machine learning techniques in enhancing ransomware detection mechanisms as well as providing a framework for future cybersecurity applications

Keywords: *Cybersecurity AI Dataset, Ransomware Sample Collection, Data Processing, Labeling, Machine Learning Models*

1. Introduction

In cyberspace, the world has increasingly faced many challenges. Ransomware has emerged as one of the most significant among those challenges. Ransomware is a software designed by cybercriminals to maliciously encrypt victim's files and demand for a ransom for them to be decrypted. This practice has led to substantial financial loss and operational disruptions across almost all sectors. Such sectors include industry, health care,

Manuscript Received: October. 9, 2024 / Revised: October.15 , 2024 / Accepted: October. 20, 2024
Corresponding Authors: HoonJae LEE
E-mail: hjlee@dongseo.ac.kr
Tel: +82 51-320-1730
Author's affiliation (Professor, Dept. of Information Security, Dongseo University, Busan, Korea)

business and trade, education. Traditional detection methods based on signature-based approaches have been used to combat ransomware attacks but have proven to be increasingly inadequate due to rapid evolution of ransomware variants. increasingly proving to be inadequate owing to the rapid evolution of ransomware variants. In support of this, the Federal Bureau of Investigation (FBI) has reported that ransomware attacks have targeted 14 of the 16 critical U.S. infrastructure sectors, including healthcare, manufacturing, and education [1].

Moreover, Korea Internet & Security Agency (KISA) indicated that ransomware attacks have increasingly targeted South Korean businesses leading significant financial losses and operational disruptions [2]. In response to these ransomware attacks, government agencies have often responded with tools such as the Ransomware Decryption Tool by KISA [3] and guidelines such as "How we can help you" by FBI [4]. In addition, Cybersecurity and Infrastructure Security Agency (CISA) has on several occasions provided comprehensive guidelines and resources to help organizations prevent and respond to ransomware attacks [5]. Despite these efforts, ransomware attacks have continued to increase. In support of this it is indicated that over the past year (2023) alone, ransomware attacks increased by 74% worldwide [6]. One of the state-of-art solutions to addressing this severe and fast-growing threat is Artificial Intelligence (AI) and Machine Learning (ML). This is because they can enable the analysis of vast datasets to identify patterns, anomalies, and behaviors ransomware. By applying AI and ML we can develop more effective and adaptive state of art solutions for detecting ransomware attacks. However, such solutions are usually hindered by lack of more comprehensive cybersecurity AI datasets that have dynamic features [8],[11]. To contribute to addressing this hinderance, we constructed a custom ransomware cybersecurity AI dataset and evaluated it using different Machine Learning models. Specifically, our key contributions are in three-fold. First, is the development of a custom cybersecurity AI dataset specifically designed for ransomware detection. This dataset contains dynamic analysis features such as API calls, DLL usage, file operations, network activity, process creation, and registry changes. These features were extracted from raw ransomware samples using Cuckoo Sandbox. Second, we conducted a comprehensive evaluation of multiple ML models such as Logistic Regression, Random Forest, K-Nearest Neighbors (KNN), and XGBoost to determine their performance in ransomware detection using our custom cybersecurity AI Dataset. Third, our research also contributes to the existing body of knowledge by demonstrating the effectiveness of dynamic analysis of the selected features and machine learning models in ransomware detection thus providing valuable insights for future cybersecurity research. The rest of our paper is organized as follows. Section 2 discusses Background and Motivation, Section 3 discusses our proposed method, Section 4 discusses results and analysis and lastly section 5 concludes our study.

2. Background and Motivation.

Emerging research indicates that ransomware attacks have become increasingly sophisticated and widespread, posing significant threats globally. As pointed out in the introduction, traditional signature-based detection methods often fail to detect new and evolving ransomware variants highlighting the need for AI-based detection methods to analyze dynamic features. Dynamic analysis methods therefore offer more effective solutions to ransomware detection. These methods involve executing code in a controlled environment and monitoring its behavior while capturing runtime characteristics such as API calls, DLL usage, file operations, network activity, process creation, and registry changes. In line with this, several studies in literature have focused on application of machine learning techniques for ransomware analysis and detection. Al-Dujaili et al. [7] demonstrated the effectiveness of machine learning techniques in identifying ransomware

based on dynamic features. Even though their study considered some dynamic features, they did not explore integration of other key aspects such as network activity and registry changes which could enhance detection accuracy of machine learning models. In addition, they did not evaluate their dataset with various machine learning models thus limiting generalization of their findings. Albshaier et al. [8] researched on ransomware early detection methods, emphasizing comprehensive dynamic analysis. The researchers' emphasis was on comprehensive analysis. While their approach was robust, it can be criticized for lack of detailed comparison with many dynamic analysis features. Furthermore, they did not provide details on the construction of the custom dataset that could have added value to their research. Vinayakumar et al. [9] provided a survey underscoring the significance of dynamic features. However, the survey did not cover issues to do with implementation and ransomware feature extraction techniques. The survey also did not include performance evaluation of customized AI dataset. Kumar et al. [10] discussed integrating these features in static analysis for ransomware classification. However, the researchers did not construct a custom cybersecurity AI dataset which could have provided a more comprehensive evaluation. Sgandurra et al. [24] conducted automated dynamic analysis of ransomware using Cuckoo Sandbox. Their analysis focused on API calls and file operations. While their approach is thorough, the impact of network activity and registry changes on detection accuracy was not explored. Moreover, there was no performance evaluation comparison of different machine learning models on the features that were extracted. Additionally, a custom dataset was not constructed, and this would have strengthened the analysis of ransomware features. Catak et al. [22] constructed and benchmarked Windows PE malware classification dataset using API calls. Even though their dataset is valuable, it did not specifically address ransomware detection using various machine learning models to understand its practical utilization. The research also did not involve construction of customized ransomware dataset. Oliveira et al. [23] built the RansomSet dataset for ransomware detection. By utilizing system calls and network activity. However, in their research they did not include analysis of registry changes and DLL usage which would enhance the detection of ransomware different variants. In addition, they did not evaluate the performance evaluation of the dataset with different machine learning algorithms. Yazı et al. [24] proposed a deep learning approach for metamorphic malware detection using API calls. Despite their innovation, they did not construct a custom cybersecurity AI dataset specialized for ransomware detection. To detect malware using Convolutional Neural Networks (CNN), Ahmed et al. [25] performed data augmentation. Despite the fact that5 their approach is interesting, there was no focus on detecting ransomware. In addition, they did not also evaluate the effective performance of various ML models on their augmented data.

In summary all these studies don't involve a combined focus on construction of custom ransomware AI Dataset and its performance evaluation through different ML models specifically focusing on API calls, DLL usage, file operations, network activity, process creation and registry changes features. In our research we address this gap and contribute to enhancing ransomware detection mechanisms.

3. Proposed Method

In this section, we describe the stages involved in constructing our cybersecurity AI dataset. Specifically, we describe the environment construction, ransom raw sample collection, processing, labelling, and finally evaluating the constructed dataset. Our proposed framework consists of eight modules, as shown in Figure.1.

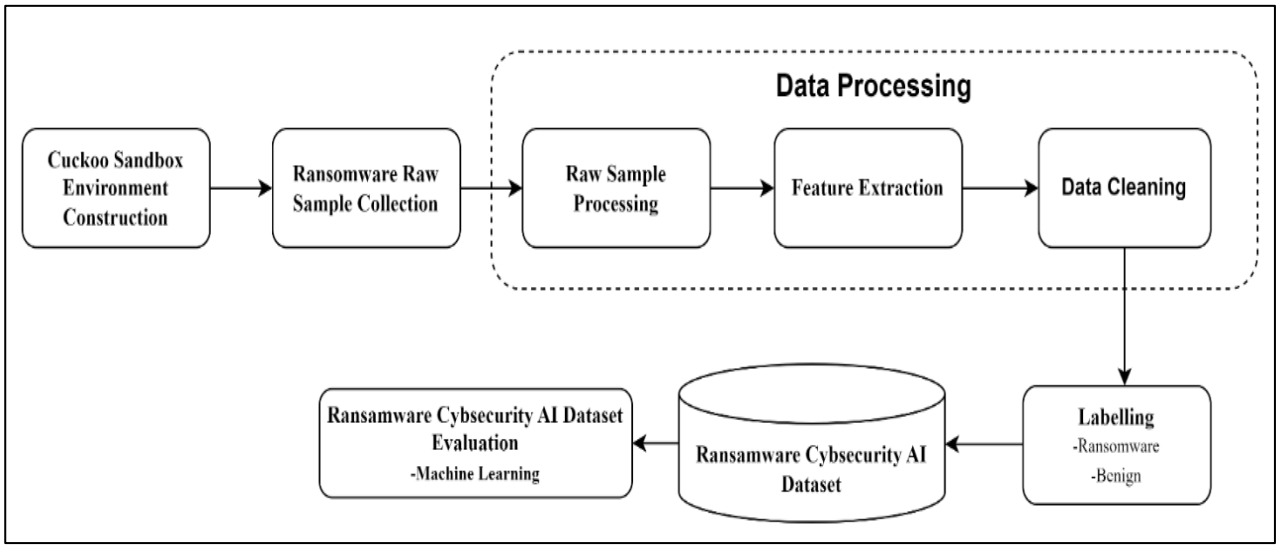

Figure 1. Proposed Framework for the Construction and Evaluation of a Ransomware Cybersecurity AI Dataset.

3.1 Cuckoo Sandbox Environment Construction

The Cuckoo Sandbox was set up for the dynamic analysis of the ransomware samples. This environment allows for the extraction of detailed behavioral features. Cuckoo Sandbox is an open-source automated malware analysis system. For this research, we set up Cuckoo Sandbox in a nested virtualization manner on a dedicated machine with the following specifications. Host Operating System: Windows 11 Home Edition, Processor: 13th gen Intel(R) Core (TM) i9-13900, RAM: 128 GB. VM Hosting Cuckoo platform: VMware Workstation 17 Pro, OS: Ubuntu 18.04.6 LTS, Disk Space: 528 GB. Cuckoo Sandbox Version: 2.0.7, VM software hosting Operating System to be infected with ransomware in Cuckoo Sandbox environment: Oracle Virtual Box 5.2.42. OS. Windows 7 ultimate. The configuration settings were optimized to capture the detailed API calls, DLL usage, file operations, network activity, process creation and registry changes features. To ensure a secure and isolated environment, we implemented several precautions, including taking VM snapshots, disabling network adapters in the virtual machine hosting sandbox environment, and setting the network connection to Host-only. After the analysis of the samples in the Cuckoo sandbox, the directories containing samples were overwritten such that they were recovered using the sudo find /home/admin123/testsample/test1 -type f -exec shred -u -n 3 -v {}, and sudo find /home/admin123/testsample/test1 -type d -empty -delete. Then, we reverted to the original snapshot. These measures are crucial for the safe handling and analysis of ransomware samples.

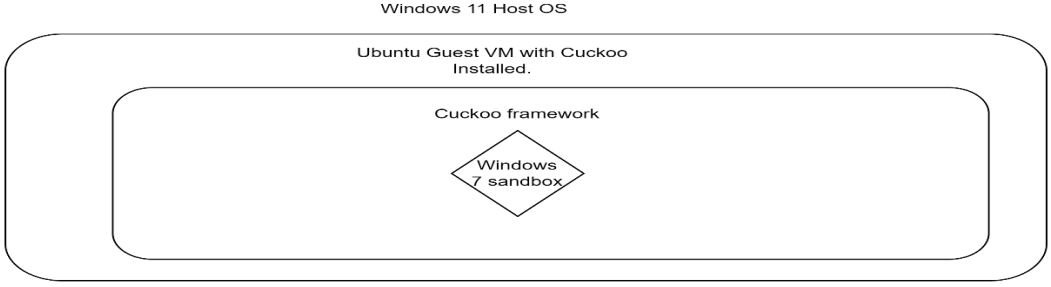

Figure 2. Cuckoo sandbox system design

3.2 Ransomware Raw Sample Collection

We collected 1,209 samples from the online repository malwares.com. To facilitate the selection process, we utilized a Python script that filters ransomware samples based on specific tags. This script leveraged the Abuse.ch API, an online repository for online repository malwares.com, to search for and download samples. The Python script was designed to automate the search and downloading processes. It includes the functionality to handle retries in the case of connection issues, ensuring reliable retrieval of the samples. The script parameters include the API key, search query, number of samples to be downloaded, and output directory for saving the files.

```
import os
import requests

# MalwareBazaar API key
api_key = 'MyAPIKey'

# Directory to save the downloaded files
ransomware_dir = "ransomware_samples"
os.makedirs(ransomware_dir, exist_ok=True)

# Load already downloaded hashes
downloaded_hashes = set(os.listdir(ransomware_dir))

# Function to download a sample
def download_sample(sha256_hash, dest_folder):
    url = 'https://mb-api.abuse.ch/api/v1/'
    data = {
        'query': 'get_file',
        'sha256_hash': sha256_hash,
        'api_key': api_key
    }
    response = requests.post(url, data=data)
    if response.status_code == 200:
        with open(os.path.join(dest_folder, sha256_hash + '.zip'), 'wb') as f:
            f.write(response.content)
        print(f"Downloaded: {sha256_hash}")
    else:
        print(f"Failed to download: {sha256_hash} - Status Code: {response.status_code}")

# Function to get ransomware samples
def get_ransomware_samples(limit=2000):
    url = 'https://mb-api.abuse.ch/api/v1/'
    count = len(downloaded_hashes)
    seen_hashes = set()
    print(f"Starting download. Already downloaded: {count}")
    while count < limit:
        print(f"Requesting samples. Current count: {count}")
        data = {
            'query': 'get_taginfo',
            'tag': 'ransomware',
            'limit': 100,
            'api_key': api_key
        }
        response = requests.post(url, data=data)
        if response.status_code == 200:
            try:
```

Figure 3. Code a snippet for scripts to download samples.

3.3 Data Processing

3.3.1 Raw Ransomware Sample Processing

Raw ransomware samples were processed using a series of structured steps to ensure comprehensive data collection and analysis. Initially, each sample was verified for integrity and labeled with unique identifiers. To avoid the duplication of samples, we considered the SHA256 of the sample as a unique identifier. The collected samples were then executed within a Cuckoo Sandbox environment configured to mimic a typical Windows 7 operating system, capturing detailed behavioral data such as API calls, DLL usage, file operations, network activity, process creation and registry changes that were processed following the algorithm in 3.1.

The sandbox environment recorded these activities in real time, and the data were exported in the JSON format for further analysis. These data were stored in a centralized repository. An initial cleaning process was applied to remove irrelevant information, focusing on malicious behaviors exhibited by the ransomware. The cleaned data were then preprocessed to standardize formats, handle missing values, and normalize features, thereby ensuring consistency and accuracy for subsequent analysis and feature extraction. The remainder of the stages follow Algorithm 3.1.

1. **Start**
2. **Initialize Libraries:** Import necessary libraries (json, os, pandas, scipy, sklearn, seaborn, matplotlib).
3. **Define extract_features Function:**

   Open Report: Load JSON report.

   Initialize Features Dictionary.
4. **Extract API Calls:** Count the number of API calls. API Calls = $\sum_{i=1}^{n} API_i$, where n is the total number of API calls.
5. **Extract File Operations:** Count the number of file operations (written, read, deleted). File Operations = $\sum_{i}^{n}(\text{File Written} + \text{File Read} + \text{File Deleted})_i$, where n is the total number of file operations.
6. **Extract Network Activity:** Count the number of network activities (HTTP, DNS, TCP, UDP). Network Activity = $\sum_{i=1}^{n}(HTTP + DNS + TCP + UDP)_i$. Where n is the total number of network activities.
7. **Extract Registry Changes:** Count the number of registry changes (written, deleted). Registry Changes = $\sum_{i=1}^{n}(\text{Regkey Written} + \text{Regkey Deleted})_i$. Where n is the total number of registry changes.
8. **Extract DLL Usage:** Count the number of DLLs loaded. DLL Usage = $\sum_{i=1}^{n} DDL_i$. Where (n) is the total number of DLLs loaded.
9. **Extract Process Creation:** Count the number of processes created. Process Creation = $\sum_{i=1}^{n} Process_i$. Where n is the total number of processes created.
10. **Determine Label:** Assign label based on malicious behavior score.

    Extract Ransomware Family: Get ransomware family information.
11. **Set Reports Directory:** Define the directory containing Cuckoo Sandbox reports.
12. **Initialize Data List:** Create an empty list to store extracted features.
13. **Iterate Over Analysis Directories:**

    For Each Directory:

    Check Report Existence: Verify if report.json exists.

    Extract Features: Call extract_features function.

    Append Features: Add extracted features to the data list.
14. **Create DataFrame:** Convert the data list to a Pandas DataFrame.
15. **Handle Missing Values:** Fill missing values with 0. We model this as $X_{filled} = \begin{cases} X \; if \; X \; is \; not \; missing \\ 0 \quad\quad if \; X \; is \; missing \end{cases}$

    Where X is the original value, and X_{filled} is the value after filling missing entries with 0.
16. **Remove Duplicates:** Drop duplicate rows.
17. **Remove Outliers:** Filter outliers based on z-score. Z-score, $z = \frac{(X-\mu)}{\sigma}$ Where X is the value, μ is the mean, and σ is the standard deviation.

18. **Standardize Features:** Standardize numerical features using StandardScaler. Standardization, $X_{std} = \frac{(X-\mu)}{\sigma}$, Where X is the original value, μ is the mean, and σ is the standard deviation.

19. End

3.1 Algorithm for data processing in cuckoo sandbox.

3.3.2 Feature Extraction

The extracted features were cleaned and pre-processed to ensure consistency and accuracy. In the algorithm in 3.1, feature extraction is designed to process JSON reports generated by Cuckoo Sandbox and extract various dynamic features relevant to ransomware analysis. It begins by loading the JSON report and initializing a dictionary to store extracted features. The algorithm then counts the total number of API calls, file operations (including files written, read, and deleted), network activities (such as HTTP, DNS, TCP, and UDP connections), registry changes (keys written and deleted), DLLs loaded, and the processes created. These counts were aggregated to provide a comprehensive set of features that captured the ransomware behavior. The systematic extraction of dynamic features is crucial for understanding ransomware behavior and developing effective detection mechanisms. Missing values were handled, and the features were normalized. Using Cuckoo Sandbox, we extracted dynamic features such as API calls, DLL usage, file operations, network activity, process creation and registry changes features. The Term Frequency-Inverse Document Frequency (TF-IDF) was used to quantify the importance of each feature.

$$\text{tf-idf}(t, d) = \text{tf}(t, d) \times \text{idf}(t) \quad (1)$$

where ft(t, d) is the term frequency of term t in document d and idf(t) is the inverse document frequency of term t [31].

3.3.3 Data Cleaning

The data were cleaned and pre-processed to remove noise and ensure consistency. This involved several steps: handling missing values by filling them with 0, removing duplicate rows to avoid redundancy, filtering outliers based on z-scores to maintain data integrity, and standardizing numerical features using StandardScaler to ensure uniformity across the dataset, as indicated in Algorithm 3.1.

3.4 Data Labeling

Samples were automatically labeled as ransomware or benign based on their behavior in the sandbox environment. labeling was performed using a rule-based approach, where samples were classified as ransomware if their behavior score exceeded a certain threshold. Specifically, the data were labeled as ransomware if the Cuckoo Sandbox report indicated a score greater than 0; otherwise, they were labeled as benign. This method relies on predefined criteria rather than machine-learning techniques. The tools and scripts used for labeling included custom Python scripts that processed the JSON reports generated by Cuckoo Sandbox. The labeling process can be represented as a binary classification problem, in which each sample is assigned a label based on its behavior score from the Cuckoo Sandbox report.

Our labeling function L(S) can be defined as $\text{L(S)} = \begin{cases} \mathbf{1}\ \boldsymbol{if}\ \boldsymbol{S} > \mathbf{0} \\ \mathbf{0}\ \boldsymbol{if}\ \boldsymbol{S} \leq \mathbf{0} \end{cases}$

Where L(S) = 1 indicates that the sample is labeled as ransomware, and L(S) = 0 indicates that the sample is labeled as benign.

3.5 Ransomware Cybersecurity AI Dataset

The resulting Ransomware Cybersecurity AI Dataset is a comprehensive collection of 1209 ransomware samples, capturing detailed behavioral data such as API calls, DLL usage, file operations, network activity, process creation and registry changes features. The resulting dataset contains 1209 samples (659 ransomware and 363 benign) and seven features stored in csv format. The dataset was carefully labeled and preprocessed to ensure consistency and accuracy.

	1	2	3	4	5	6	7	8
1	api_calls	dll_usage	file_operations	network_activity	process_c	registry_cl	sha256	label
2	62342	12	25	17	2	39	4c000fe92ad033c543d371434eb75cc0cec69b7b4af02ccbc73dfad60e6c474f	1
3	0	0	0	16	1	0	22461a09d47bcd5701c4d135a2ff94e014aa3c04cc78447980be37498048f543	0
4	42	3	0	17	2	4	8d6135b6945bcf772cd883391533f3f6b1175f6c5ad65f0f241a695b68ed9278	1
5	0	0	0	3	0	0	7ac41fc734444e60a46b6bbfc37dd0d0c53627b7a8242f702f021e9efb3081c3	1
6	0	0	0	16	1	0	f8c8a18905afedd141b99b3f769292d3d12e3be4bd028e0dbf9e8d234df010e1	0
7	0	0	0	4	0	0	2ddf303378247d56e7be2f2da85ccc61de32d47b26652372de8b5c98c78432f7	0
8	49104	52	48	19	5	23040	23f03ca7290d6bc77a6ede705835c85ac89b8d27ad22d15d7b63b0e5ca687ef7	1
9	0	0	0	3	0	0	4d39b4d345bb7b8ba8ce61a8b9a95e87ab15845f799df1b470520b193b72f133	1
10	1108	27	0	46	2	354	7bd45b7d823cb40a6991995d1b695d28d9c6154739e801c55b0635f5c707d00f	1

Figure 4. Dataset generated from raw ransomware samples in a cuckoo sandbox environment.

In Figure 3, we plot the correlation matrices among our dataset features in a heatmap. This heatmap visualizes the correlation coefficients between different features, which helps to understand the relationships among them. It can support various cybersecurity applications, including ransomware detection, behavioral analysis, and threat intelligence, and is available to the research community under ethical guidelines to promote further advancements in the cybersecurity field.

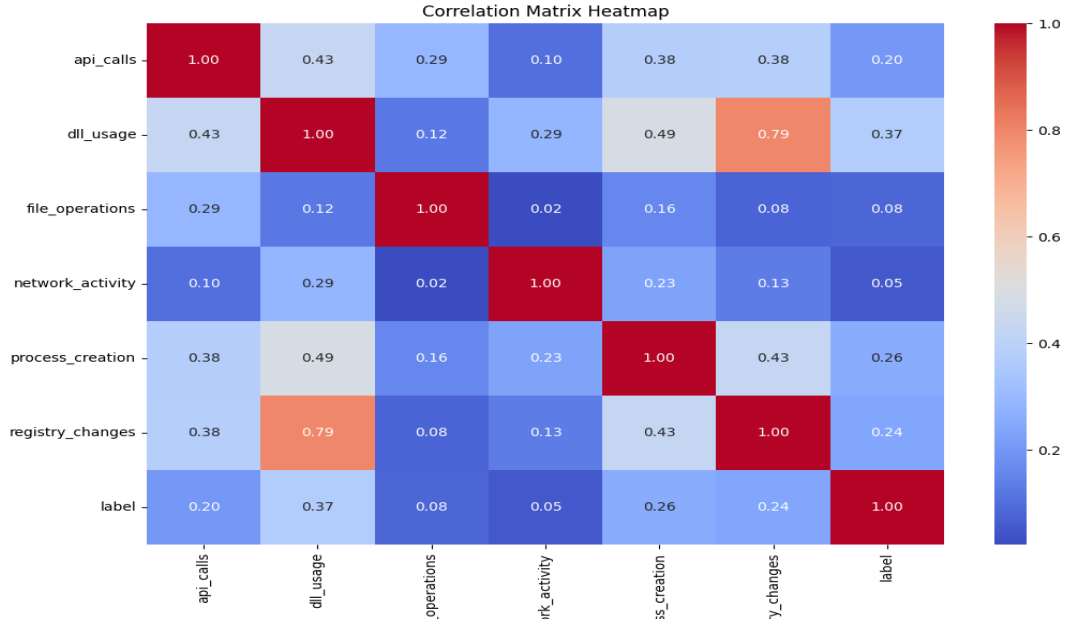

Figure 5. Correlation Matrices Among Dataset Features

3.6 Ransomware Cybersecurity AI Dataset Evaluation through Machine Learning

To ensure the suitability of our dataset for real-time ransomware detection, we evaluated it using various

machine learning algorithms. The evaluation process followed the flowchart described in Section 3.2.

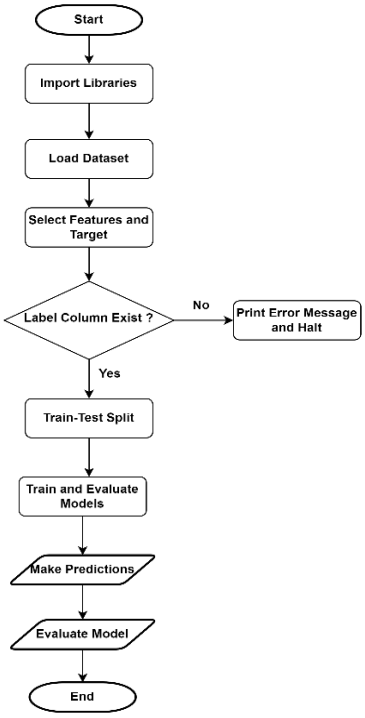

Figure 6. Ransomware Cybersecurity AI Dataset evaluation framework.

Our dataset evaluation process began by importing the necessary libraries and loading the dataset, which consisted of 1209 samples. We performed descriptive statistics to understand the data distribution, focusing on key features, such as API calls, DLL usage, file operations, network activity, process creation and registry changes features. After verifying the existence of the selected features and target columns, we proceed to split the dataset into training and testing sets to facilitate model evaluation. The dataset was split into 80% for training and 20% for testing. We then employ various ML models to analyze the data. The modes included Logistic Regression, Random Forest, K-Nearest Neighbors (KNN) and XGBoost. These models help in predicting the probability of a sample being ransomware through the following equations.

Logistic Regression classifies samples with:

$$P(y=1|X) = \frac{1}{e^{-(\beta 0+\beta 1X1+\cdots+\beta pXp)}} \quad (2)$$

where y is the binary outcome (ransomware or not), X is the feature vector and are the coefficients [26].
Random Forest classifies samples based on multiple decision trees with the equation:

$$\hat{y} = \frac{1}{T}\sum_{t=1}^{T} h_t(x) \quad (3)$$

where T is the number of trees, h_t is the prediction of the t^{th} tree, and x is the feature vector.

K-Nearest Neighbors (KNN) classifies samples based on the similarity to their k-nearest neighbors using:

$$\hat{y} = \frac{1}{k}\sum_{i=1}^{k} y_i \quad (4)$$

where k is the number of nearest neighbors and y_i are the labels of the nearest neighbors [27].

XGBoost efficiently classifies samples with

$$\hat{y} = \sum_{m=1}^{M} \alpha_m h_m(x) \quad (5)$$

where M is the number of trees, α_m are the weights, and h_m are the predictions of the trees [28].

The models help predict the likelihood of a sample being ransomware based on the extracted features. After training the models, we make predictions on the test set and evaluate their performance using metrics such as accuracy, precision, recall, and F1-score. This comprehensive evaluation ensures that our models generalize well to unseen data and are effective in detecting ransomware in real-time scenarios. Feature engineering is crucial, utilizing API calls, DLL usage, file operations, network activity, process creation and registry changes for model training, with hyperparameter tuning optimizing performance. Finally, we evaluate the models using metrics such as Accuracy $= \frac{TP+TN}{TP+TN+FP+FN}$, Precision = $\frac{TP}{TP+FP}$, Recall = $\frac{TP}{TP+FN}$, and F1-score = 2 x $\frac{PrecisionxRecall}{Precision+Recall}$, [29],[30] ensuring a comprehensive assessment of our ransomware detection capabilities. TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives, and FN is the number of false negatives. Relevant charts and plots, such as feature importance graphs, ROC curves, confusion matrices, heat maps, and Precision-Recall Curves, are used to visualize the performance and significance of each model, providing deeper insights into the detection process.

4. Results and Analysis

The table compares the performance metrics of various machine learning models used for ransomware detection, including Logistic Regression, Random Forest, K-Nearest Neighbors (KNN) and XGBoost. The achieved F1 score of 83%, precision of 81%, recall of 84%, and accuracy of 82%, align well with recent research findings in machine learning model evaluation. For example, literature review by Albshaier et al. [8] reports F1 scores ranging from 0.75 to 0.85, indicating that our model performs within the upper range of these values. Similarly, Alraizza and Algarni [11] discuss models achieving precision and recall values typically between 0.75 and 0.90, which matches our achieved precision and recall metrics. Additionally, the survey by Alraizza and Algarni [11] on machine learning techniques for ransomware detection highlights that advanced models can achieve F1 scores above 0.80, further validating our results. Lastly, the review by Albshaier et al. [8] provides a broad range of performance metrics for various detection mechanisms, with F1 scores generally between 0.70 and 0.85, placing our model at the higher end of this spectrum. These comparisons demonstrate that our dataset and machine models perform well in the context of existing research.

Table 1. Machine Learning performance evaluation in ransomware detection (percentages are rounded up to nearest whole numbers)

Model	Accuracy (%)	Precision (%)	Recall (%)	F1 score (%)
Logistic Regression	82	81	84	83
Random Forest	83	81	84	83
K-Nearest Neighbors (KNN)	82	76	93	83
XGB	83	81	85	83

Figure 7 shows the performance of each machine learning algorithm as a graph in detection ransomware. As can be seen, all the four machine learning algorithms generally performed well on our dataset in all the four metrics. KNN lowest performance precision.

Figure 7. Machine Learning algorithms performance in ransomware detection

Feature Performance

A plot of feature importance indicates that Network work activities contributed the highest in dynamic feature analysis followed by API calls and Process Creation.

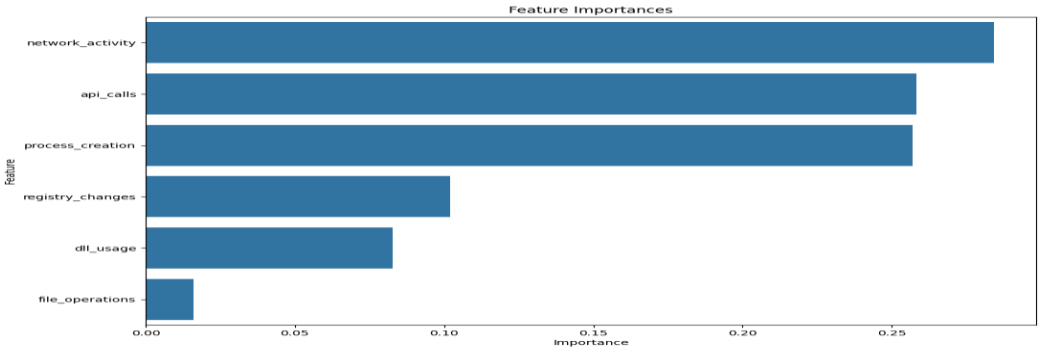

Figure 8. A plot of feature performance

ROC curves

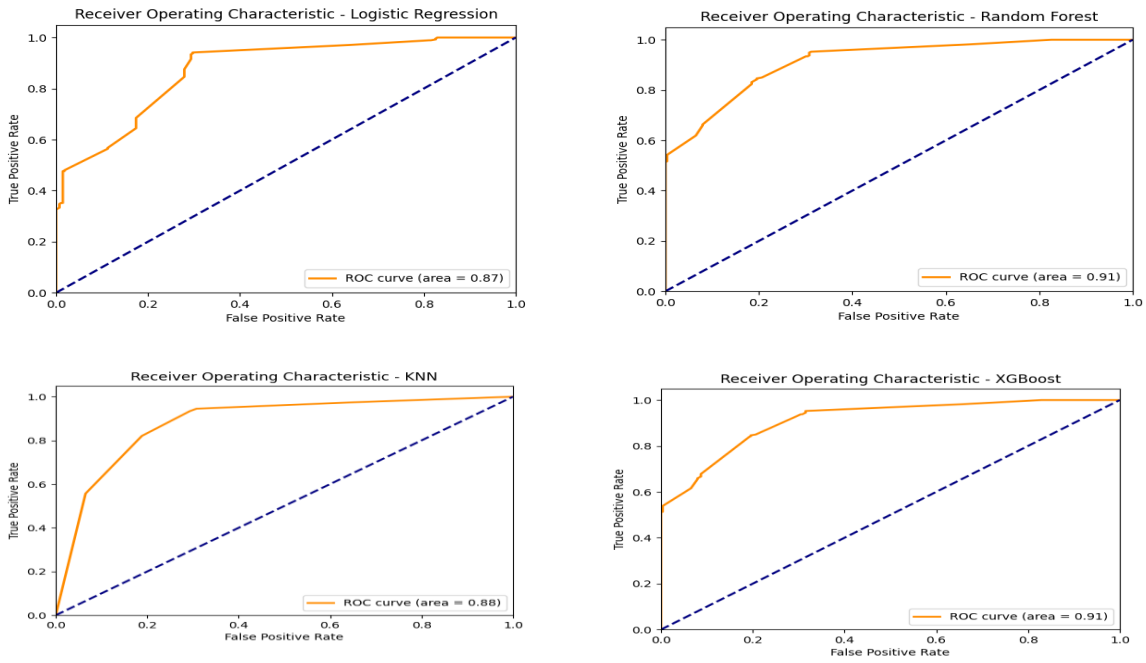

Figure 9. ROC curves of the models

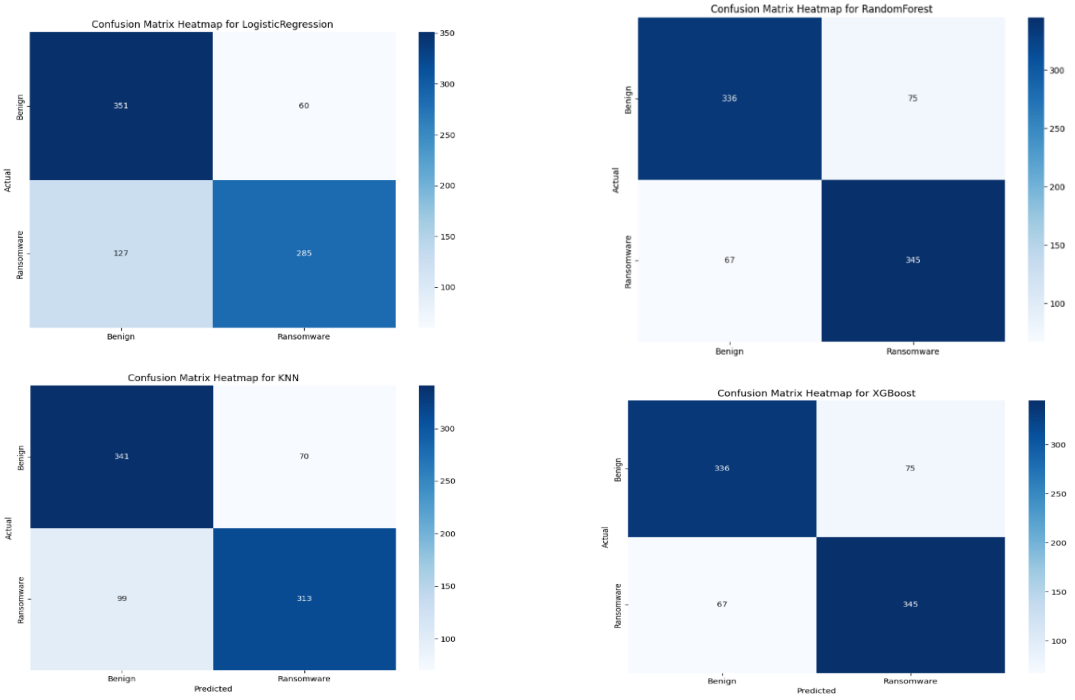

Figure 10. Confusion matrices of the models

5. Conclusion and Future Research

In conclusion, we successfully constructed a comprehensive cybersecurity AI dataset and evaluated it using machine learning models. The results of the ML performance evaluation metrics that were obtained are evidence of the effectiveness of our setup in detecting and mitigating ransomware. Moreover, our research emphasized the importance of dynamic analysis features and diverse custom cybersecurity AI datasets towards enhancing ransomware detection accuracy. In future we intend to explore the use of additional features, such as memory usage and evaluate them using deep leaning models.

6. Acknowledgement

"The Construction Project for Regional Base Information Security Cluster", grant funded by Ministry of Science, ICT and Busan Metropolitan City in 2024 & This research was also supported by "Regional Innovation Strategy (RIS)" through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (MOE).(2023RIS-007).

7. References

[1] United States Government Accountability(GAO), "Critical Infrastructure Protection", Report to Congressional Addressees, 2024, January (Access date 2024.09.07), https://www.gao.gov/assets/d24106221.pdf

[2] Yang Sung-jin, "Ransomware attacks on Korean companies on the rise: KISA", The Korea Herald, 2024, 2021/August/2 (Access date 2024.09.07), https://www.koreaherald.com/view.php?ud=20210802000863

[3] Korea Internet & Security Agency (KISA), "Directions for Using Decryption Tool" Rhysida Ransomware Decryption Tool User Manual_v1.0, 2023 December (Access date 2024.09.07), https://seed.kisa.or.kr/async/MultiFile/download.do?FS_KEYNO=FS_0000000471&MNK=MN_0000001279

[4] Federal Bureau of Investigation. "Ransomware" FBI. (Access date 2024.09.07). https://www.fbi.gov/how-we-can-help-you/scams-and-safety/common-frauds-and-scams/ransomware.

[5] Cybersecurity & Infrastructure Security Agency (CISA), "#StopRansomware: Akira Ransomware", Joint Cybersecurity Advisory, 2024/April/18 (Access date 2024.09.07), https://www.cisa.gov/sites/default/files/2024-04/aa24-109a-stopransomware-akira-ransomware_2.pdf

[6] Smith, J, Ransomware Attacks Surge in 2023; Attacks on Healthcare Sector. Cybersecurity Journal, 15(3), 45-60, 2024/February/28, (Access date 2024.09.07) https://www.dni.gov/files/CTIIC/documents/products/Ransomware_Attacks_Surge_in_2023.pdf

[7] Al-Dujaili, A., et al. "Ransomware Detection Using Machine Learning Techniques." IEEE Access, 2020.

[8] Albshaier, L., et al. "Earlier Decision on Detection of Ransomware Identification: A Comprehensive Systematic Literature Review." Information, 2024.

[9] Vinayakumar, R., et al. "A Comprehensive Survey of Ransomware Detection Techniques." Computers & Security, 2019.

[10] Kumar, R., et al. "Static Analysis and Machine Learning for Ransomware Classification." 2021.

[11] Alraizza, A., and Algarni, A. "Ransomware Detection Using Machine Learning: A Survey." Big Data Cogn. Comput., 2023.

[12] Sgandurra, D., et al. "Automated Dynamic Analysis of Ransomware: Benefits, Limitations and Use for Detection." arXiv preprint arXiv:1609.03020, 2016.

[13] Ndibanje, B., et al. "Cross-Method-Based Analysis and Classification of Malicious Behavior by API Calls Extraction." Applied Sciences, 2019.

[14] Urooj, U.; Al-rimy, B.A.S.; Zainal, A.; Ghaleb, F.A.; Rassam, M.A. Ransomware Detection Using the Dynamic Analysis and Machine Learning: A Survey and Research Directions. Appl. Sci. **2022**, 12, 172. https://doi.org/10.3390/app12010172

[15] Mercaldo, F. A framework for supporting ransomware detection and prevention based on hybrid analysis. J Comput Virol Hack Tech **17**, 221–227 (2021). https://doi.org/10.1007/s11416-021-00388-w

[16] Damodaran, A., et al., A Comparison of Static, Dynamic, and Hybrid Analysis for Malware Detection. San Jose State University, 2024 (Access date 2024.09.09) https://www.cs.sjsu.edu/faculty/stamp/papers/Anusha.pdf.

[17] Singh, A., et al Enhancing Ransomware Attack Detection Using Transfer Learning and Deep Learning Ensemble Models on Cloud-Encrypted Data. Electronics **2023**, 12, 3899. https://doi.org/10.3390/electronics12183899

[18] Latifa Albshaier, Seetah Almarri, and M. M. Hafizur Rahman. "Earlier Decision on Detection of Ransomware Identification: A Comprehensive Systematic Literature Review." Journal of Cybersecurity, 2024.

[19] Amjad Alraizza and Abdulmohsen Algarni. "Ransomware Detection Using Machine Learning: A Survey." Journal of Information Security and Applications, 2023.

[20] "A State-of-the-Art Survey on Ransomware Detection using Machine Learning and Deep Learning Techniques." IEEE Access, 2023.

[21] "Systematic Literature Review and Metadata Analysis of Ransomware Attacks and Detection Mechanisms." Computers & Security, 2019.

[22] Catak, F. O., et al. "Benchmark Dataset for Windows PE Malware Classification Using API Calls." Journal of Information Security and Applications, 2020. DOI: 10.1016/j.jisa.2020.102528

[23] Oliveira, L., et al. "RansomSet: A Dataset for Ransomware Detection." Journal of Computer Virology and Hacking Techniques, 2021. DOI: 10.1007/s11416-021-00378-1

[24] Yazı, A., et al. "Deep Learning Approach for Metamorphic Malware Detection Using API Calls." IEEE Transactions on Information Forensics and Security, 2020. DOI: 10.1109/TIFS.2020.2975741

[25] Ahmed, F., et al. "Data Augmentation Techniques for Malware Detection Using Convolutional Neural Networks." IEEE Access, 2021. DOI: 10.1109/ACCESS.2021.3055741

[26] Hosmer, David W., Jr., Stanley Lemeshow, and Rodney X. Sturdivant. Applied Logistic Regression. 3rd ed. Hoboken, NJ: Wiley, 2013. https://doi.org/10.1002/9781118548387.

[27] Cover, Thomas, and Peter Hart. "Nearest Neighbor Pattern Classification." IEEE Transactions on Information Theory 13, no. 1 (1967): 21-27. https://doi.org/10.1109/TIT.1967.1053964.

[28] Chen, Tianqi, and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System." In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785-794. New York, NY: ACM, 2016. https://doi.org/10.1145/2939672.2939785.

[29] Sokolova, Marina, and Guy Lapalme. "A systematic analysis of performance measures for classification tasks." Information Processing & Management 45, no. 4 (2009): 427-437. https://doi.org/10.1016/j.ipm.2009.03.002.

[30] Powers, David M. W. "Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness and Correlation." Journal of Machine Learning Technologies 2, no. 1 (2011): 37-63. https://doi.org/10.48550/arXiv.2010.16061.

[31] Jones, Karen Sparck. "A Statistical Interpretation of Term Specificity and Its Application in Retrieval." Journal of Documentation 28, no. 1 (1972): 11-21. https://doi.org/10.1108/eb026526