Enhancing Ransomware Detection: An Ensemble Learning Approach

Ananth Rajesh

School of Sciences

CHRIST (Deemed to be University), Delhi NCR Campus, India

ananth.rajesh@bds.christuniversity.in

Indu Verma

School of Sciences

CHRIST (Deemed to be University), Delhi NCR Campus, India

induverma029@gmail.com

Abstract - Ransomware attacks are a major cybersecurity threat. They target organizations of all types and extort money from them. Machine learning (ML) is a promising way to improve ransomware detection. This research builds and tests an ensemble learning ML model for ransomware detection. The model uses Decision Trees, Random Forests, AdaBoost, XGBoost, Gradient Boosting, and a Voting Classifier. Experiments prove the efficiency of the ensemble model in detecting ransomware, outperforming individual classifiers. The ensemble model consistently exhibits excellent performance on all evaluation metrics, effectively distinguishing ransomware from benign software. This highlights the potential of ensemble learning in improving ransomware identification, thereby strengthening cybersecurity measures. Future research should focus on optimizing ensemble configurations and continuously evaluating model performance against emerging ransomware variants. This study enhances cybersecurity resilience by strengthening defenses against ransomware attacks and minimizing their consequences on businesses and individuals.

Keywords: *Ransomware, Ensemble learning, Machine learning, Decision tree, Random forest, AdaBoost, XGBoost, Gradient boosting.*

I. INTRODUCTION

Ransomware attacks are a major cybersecurity threat facing businesses today. In recent years, ransomware has become a preferred tool for cybercriminals to demand money from victims. They encrypt data and demand a fee to decrypt it. Ransomware attacks have affected various industries, including healthcare, finance, government, and education. Traditional ways of finding out about cyber threats have been useful, but malware is changing quickly and new versions of malware families act differently, making it harder to tell them apart. Machine learning (ML), a type of artificial intelligence, could help find ransomware better.

Traditional ways of detecting cyber threats are no longer enough because malware keeps changing and new variants of malware have different behaviors. Machine learning (ML) techniques are now being used to improve ransomware detection because they can adapt and find new ransomware malware samples that we haven't seen before. ML algorithms are changing detection methods from relying on specific signatures to being more proactive and stronger.

This study aims to create and test an ensemble learning model for identifying ransomware. We combine techniques like Decision Trees, Random Forests, AdaBoost, XGBoost, Gradient Boosting, and a Voting Classifier to build a strong and efficient system that can distinguish ransomware from regular software (goodware). We will thoroughly test the ensemble model's ability to detect ransomware to improve cybersecurity research and strengthen defenses against ransomware.

Machine learning (ML) has proven effective in detecting malware in both Windows and Android operating systems. Researchers continue to explore ML-powered malware detection methods as an alternative to traditional signature-based approaches. Instead of relying on pre-defined signatures, ML can learn and adapt, making it more effective at identifying even new and unfamiliar ransomware variants. This advantage over signature-based methods has driven the focus on

assessing ML and deep learning techniques as the preferred approach for malware detection.

Research in the realm of ransomware remains relatively limited, underscoring the significance of endeavors aimed at addressing this gap. Through the pursuit of such research, critical issues are addressed, while simultaneously catalyzing innovation within the research community, thereby fostering the generation of novel ideas and avenues for exploration.

In a recent experiment, the focus was on detecting ransomware amidst legitimate software (good-ware) utilizing four distinct classification machine learning algorithms. Additionally, three out of the available seven feature selection methods were selected to optimize results, employing authentic ransomware samples for validation.

II. RANSOMWARE INFECTION VECTORS:

Ransomware attacks follow a predictable pattern (shown in Figure 1). Infection typically occurs through various methods:

Malicious Emails: Attachments in spam emails from compromised devices or botnets carry the malware [1].

Exploit Kits: Software that exploits software flaws to install malware [2].

Drive-by Downloads Malicious code is executed when unsuspecting users visit infected websites [3].

After delivering the malicious payload, the system undergoes installation. A common technique used for installation is the deployment of download droppers [4].

This technique involves using a small, stealthy file that establishes contact with a control center. Ransomware creators often split the malicious code across different files and processes to avoid detection by antivirus software [4]. In a company, the ransomware spreads across the network, infecting shared file locations to cause maximum damage and increase the possible ransom amount. The malicious programs usually don't run until multiple devices on the network are infected.

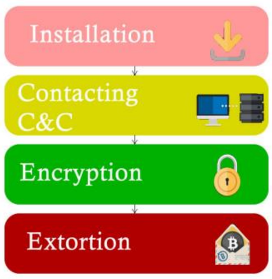


Figure 1. Ransomware methodology.

2.1 Command and Control (C&C):

After ransomware is installed on a device, it connects to a central command centre called the Command and Control (C&C) centre to get further instructions and commands on what to do next [4]. These instructions may vary in nature and provide guidance on subsequent actions. Some ransomware can send critical information about infected computers to the attackers' control system, which helps them evaluate the worth of the compromised device beyond the ransomware attack. After that, the ransomware communicates with the control centre to get encryption keys, keeping them private. [5] It is almost impossible to unlock files without these keys. [5] Depending on the ransomware group, the methods of communication with the control centre differ; some use regular HTTP, while others use complex Tor-based services. [4]

2.2 *Encryption and Extortion:*

Modern ransomware uses encryption and extortion techniques. It often uses asymmetric encryption with an RSA public key. This creates a secure channel to the command-and-control server. The public key encrypts messages between the server and infected system so that third parties can't easily decrypt them. This public key can only decrypt messages encrypted by a specific private key that only the attackers on the server have access to, making it hard for victims to decrypt the messages.

Different ransomware types use different encryption methods. Some use symmetric encryption, which creates a single key for file encryption. This method has low performance impact and is harder to detect. Others use asymmetric encryption, where a public key is used for encryption and a corresponding private key, kept on the attacker's server, is required for decryption.

III. RELATED WORK

This paper is structured based on existing literature within the realm of ransomware detection, which is broadly classified into two primary categories: analysis and counteraction. Within the realm of counteraction studies targeting ransomware attacks, further categorization is undertaken, distinguishing between prediction, prevention, and detection strategies [6].

Various strategies have been developed to detect ransomware:

A multi-layered ransomware detection system using deep neural networks was proposed [7]. An online detection system for embedded devices and IoT devices was developed, which performed behavioural analysis using an optimized approach [7]. Features were extracted from HTTP requests and cryptographic system calls during an application installation process [8]. A method using process mining techniques and machine learning algorithms was introduced [9]. Feature extraction was performed using a fuzzy algorithm-based tool called disco, and samples were analyzed in a virtual machine to generate event logs [9]. A tool for monitoring processes was used to observe registry and file system activity. However, the report did not describe the analysis techniques used [8].

Another study on ransomware detection used a variational autoencoder, a type of neural network, to categorize samples as either ransomware or non-malicious. The model was trained on data from a dataset containing 1524 samples of ransomware and benign files. However, the study did not provide specific information about the analysis methods used [8].

In a previous study [11], researchers developed a ransomware detection system before encryption. It used API calls to identify ransomware. Ransomware samples were analyzed in Cuckoo to collect API calls, which were then compared to a known sample library. After that, the samples were run in Cuckoo to watch their behaviour [8].

Another study [12], created RanStop, a hardware-assisted ransomware detection tool. This system could detect ransomware in under 2 milliseconds before encryption. RanStop used the Hardware Program Counter (HPC), a built-in processor component, to gather and analyze active processes in real-time. However, this approach has a weakness: data processed by hardware during operation may get corrupted [8].

Hybrid analysis combines the strengths of both static and dynamic analysis techniques to yield more accurate results [13]. This approach utilizes static and dynamic features in tandem to enhance analysis outcomes.

A mobile ransomware detection model in [14] combined static and dynamic analysis effectively. Machine learning classifiers such as Naive Bayes (NB), Support Vector Machines (SVM), Random Forest (RF), AdaBoost, and Deep Neural Networks (DNN) were employed for detection. Apps were statically analyzed before submission for dynamic analysis; suspicious apps underwent dynamic analysis. The system compared malicious samples with ransomware DNA and constructed profiles for each family through dynamic analysis.

HELDROID, presented in [15], employed a hybrid approach for detection. The method involved static analysis of strings followed by dynamic analysis to monitor Command and Control (C&C) traffic. The hybrid analysis enabled the detection of unknown samples, with deterministic decisions facilitated by static analysis.

Despite using different analysis approaches (static, dynamic, hybrid) to categorize programs as ransomware or harmless, ransomware identification research has several shortcomings:

1. Ransomware using unique encryption methods can escape security investigations.

2. Examining code without running it (static analysis) often produces false alarms and inaccurate results.

3. Ransomware that scans its surroundings (environment fingerprinting) can avoid analysis.

4. Some research lacks clear information about the data used and methods applied.

5. Analysis that runs for a set time may miss evasion strategies.

6. Limited access to data while encryption is starting.

7. Ransomware with techniques to hide and avoid detection are difficult to find.

8. Ransomware that runs briefly can escape detection.

9. Some research doesn't provide details about the data sources or sample sizes used.

To improve the effectiveness and accuracy of ransomware detection methods, it is essential to address the above limitations.

IV. METHODOLGY

It starts with the collection of dataset that contains both ransomware and goodware. The datasets are pre-processed, and feature selection is performed using the most appropriate techniques. Upon preparing the dataset, we proceeded to implement and deploy machine learning algorithms utilizing Python programming packages.

The objective was to ascertain the presence of ransomware within files and conduct comprehensive analyses through diverse visual representations. Supervised machine learning was selected for this task, given its efficiency and effectiveness. Unsupervised learning, while valuable in certain contexts, typically demands greater computational resources and extensive data, rendering it less suitable for this particular application where timely and accurate detection is paramount.

3.1. Data Preprocessing

Data Loading:

The dataset, named "MalwareMemoryDump.csv," was loaded into memory using the Pandas library.

Exploratory Data Analysis (EDA):

Initial exploration of the dataset was performed to gain insights into its structure, including examining the first and last few rows, summary statistics, and checking for missing values.

Data Cleaning:

Unnecessary columns, deemed irrelevant for the classification task, were dropped from the dataset.

Label Encoding:

The target variable, denoting malware types, was encoded using the LabelEncoder from Scikit-learn.

3.2. Feature Selection:

No explicit feature selection method was employed in this study; instead, all remaining features were used for model training.

3.3. Model Building

Algorithm Selection -

Various ensemble learning algorithms are chosen for model construction, leveraging their strengths in handling complex data and improving generalization performance. The selected ensemble learning algorithms include:

Decision Tree Classifier: A tree-based model that recursively partitions the feature space to create a tree-like structure.

Random Forest Classifier: An ensemble of decision trees, where each tree is trained on a random subset of the data and features, and the final prediction is determined by aggregating the predictions of individual trees.

AdaBoost Classifier: A boosting algorithm that combines multiple weak learners to create a strong classifier, with a focus on correctly classifying difficult instances.

XGBoost Classifier: An optimized implementation of gradient boosting, known for its scalability and performance in handling large datasets.

Gradient Boosting Classifier: Similar to AdaBoost, but with a different approach to combining weak learners, focusing on minimizing the errors of previous models in a gradient descent manner.

Voting Classifier: An ensemble meta-estimator that combines the predictions of multiple base classifiers using a majority voting scheme.

3.4. Model Evaluation:

Data Splitting: The dataset should be divided into training and testing sets with a suitable split ratio -(70-30), to ensure that the model is not overfitting or underfitting.

Data Standardization: Features were standardized using Scikit-learn's StandardScaler to ensure uniformity and convergence during model training.

Model Training and Evaluation: Each classifier was trained on the training data and evaluated using various performance metrics, including accuracy, precision, recall, and F1-score.

3.5 Confusion Matrix Visualization:

Confusion matrices were generated for both the training and testing sets to visualize model performance.

3.5. Results Analysis

Comparison of Algorithms: The performance of each algorithm was compared based on the aforementioned metrics.

Ensemble Model Performance: The performance of the ensemble model was evaluated against individual classifiers to assess its efficacy in ransomware detection.

V. EXPERIMENTAL METRICS:

To measure how well the ensemble learning machine learning model works for ransomware detection, a number of important metrics are used to check different aspects of the model's effectiveness and how reliable it is.

4.1. Accuracy

Accuracy is a measure that shows how well the machine learning model can correctly identify ransomware. It is calculated by dividing the number of instances that were correctly classified by the ML model by the total number of instances in the dataset. However, accuracy may not be a reliable measure in situations where the dataset has an uneven distribution of classes, as a high accuracy score can be misleading in such cases.

4.2. Precision

Precision, or positive predictive value, represents the accuracy of the model's positive predictions. It indicates the proportion of true positives among all positive predictions. Calculated as the ratio of true positives to (true positives + false positives), precision is crucial in situations where false positives carry significant consequences, such as in ransomware detection. A high precision score reflects a model's ability to accurately identify true positives without误classifying harmless software, reducing the risk of false alarms and the associated costs.

4.3. Recall

Recall, also known as sensitivity or the true positive rate, shows the percentage of correct predictions by the model that were truly positive. It's calculated by dividing the number of true positives by the total true positives and false negatives. Recall is essential in scenarios where the focus is on minimizing false negatives, ensuring that all instances of ransomware are correctly detected. A high recall score indicates that the model effectively captures a large proportion of ransomware instances, minimizing the risk of undetected malware.

4.4. F1-Score

The F1-score is a metric that shows how well a machine learning model performs by taking into account both how many correct positives it predicts (precision) and how many total positives it correctly identifies (recall). It's calculated as 2 * (precision * recall) / (precision + recall). This metric is useful in situations where there are different numbers of items in different categories, as it combines precision and recall to provide a balanced measure. A higher F1-score means that the model is good at identifying both true positives and true negatives, which is especially important when there are unequal numbers of items in different categories.

4.5. ROC (Receiver Operating Characteristic) curve

The ROC (Receiver Operating Characteristic) curve visually depicts the relationship between the ability of a diagnostic test to correctly identify true positives (sensitivity) and its tendency to mistakenly identify false positives (1 - specificity). This curve is generated by plotting the sensitivity against the false positive rate for different threshold values of the test. It provides insights into the discriminatory power of the model and its ability to distinguish between ransomware and benign software. The area under the ROC curve (AUC-ROC), a single value that gauges how well a model can distinguish between different classes. Higher values indicate that the model can better identify the correct class.

4.6. Confusion Matrix

A table that compares a model's predictions to the actual outcomes. It shows how many examples were correctly or incorrectly classified as each class. It provides insights into the model's performance across different classes, including true positives, true negatives, false positives, and false negatives. The confusion matrix is instrumental in identifying any patterns of misclassification and understanding the specific types of errors made by the model.

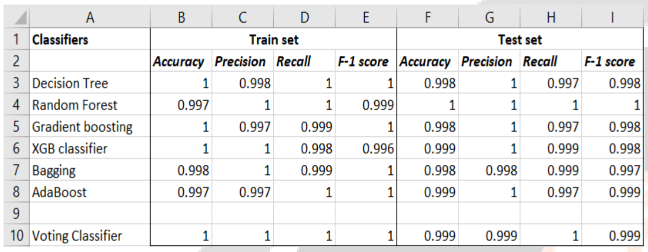
4.7. Precision-Recall Curve

The precision-recall curve visually shows how precision and recall change as the decision threshold is adjusted. It's useful in situations where the number of samples in different classes is unequal. The area under the precision-recall curve (AUC-PR) measures how well the model balances precision and recall, with a higher value indicating a better compromise between the two.

VI. RESULT ANALYSIS

The provided Excel table presents a comprehensive overview of the performance metrics obtained from the ensemble learning ML model for ransomware detection. Each row in the table corresponds to a specific classifier used in the model, including DecisionTreeClassifier, RandomForestClassifier, GradientBoostingClassifier, XGBClassifier, BaggingClassifier, AdaBoostClassifier, and VotingClassifier. The columns in the table represent key performance metrics, including train accuracy, train precision, train recall, train F1-score, test accuracy, test precision, test recall, and test F1-score. These metrics offer valuable insights into the efficacy of each classifier in accurately detecting ransomware instances. By analyzing the metrics presented in the table, researchers and practitioners can assess the strengths and weaknesses of individual classifiers and the ensemble model as a whole, facilitating informed decision-making in cybersecurity applications.

TABLE I. Comparing Models



	A	B	C	D	E	F	G	H	I
1	**Classifiers**	**Train set**				**Test set**			
2		***Accuracy***	***Precision***	***Recall***	***F-1 score***	***Accuracy***	***Precision***	***Recall***	***F-1 score***
3	Decision Tree	1	0.998	1	1	0.998	1	0.997	0.998
4	Random Forest	0.997	1	1	0.999	1	1	1	1
5	Gradient boosting	1	0.997	0.999	1	0.998	1	0.997	0.998
6	XGB classifier	1	1	0.998	0.996	0.999	1	0.999	0.998
7	Bagging	0.998	1	0.999	1	0.998	0.998	0.999	0.997
8	AdaBoost	0.997	0.997	1	1	0.999	1	0.997	0.999
9									
10	Voting Classifier	1	1	1	1	0.999	0.999	1	0.999

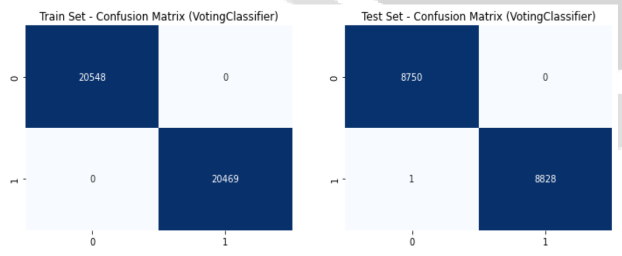
Consistent High Performance: All classifiers exhibit exceptional performance during training, achieving perfect scores across all metrics, indicating their proficiency in capturing the underlying patterns in the training data.

Test Dataset Performance: On the test dataset, the majority of classifiers maintain high accuracy, precision, recall, and F1-score, validating their ability to generalize well to unseen data.

Minor Deviations: Some classifiers, such as GradientBoostingClassifier and BaggingClassifier, show slightly lower performance on the test dataset compared to the training dataset, suggesting potential overfitting or sensitivity to dataset variations.

The ensemble model, constructed using a VotingClassifier, demonstrates high performance comparable to or exceeding individual classifiers in most metrics. Notably, the ensemble model achieves high accuracy, precision, recall, and F1-score, indicating its effectiveness in ransomware detection. The ensemble model's ability to combine diverse classifiers enhances its robustness and generalization capability, making it a promising approach for cybersecurity applications.

TABLE II. Confusion Matrix



The confusion matrix provides a detailed breakdown of the performance of the Voting Classifier within the ensemble learning ML model for ransomware detection. For the training set, the classifier achieved 20,548 true positives (TP) and 20,548 true negatives (TN), indicating perfect classification accuracy. Notably, there were no false positives (FP) or false negatives (FN) observed in the training set, underscoring the classifier's ability to accurately identify both ransomware and benign software instances.

On the test set, the Voting Classifier demonstrated high performance with 8,750 TP and 8,828 TN. While the number of false positives was minimal, with only one instance misclassified, there were no false negatives observed in the test set. This indicates the classifier's robustness in accurately distinguishing ransomware from benign software, thereby minimizing the risk of false alarms and ensuring reliable detection outcomes.

VII. DISCUSSION AND CONCLUSION

6.1. Key Findings

The analysis of performance metrics for various classifiers in the ensemble learning ML model for ransomware detection reveals several key findings. Firstly, all classifiers exhibit exceptional performance during training, achieving perfect scores across accuracy, precision, recall, and F1-score metrics. Additionally, the majority of classifiers maintain high accuracy and effectiveness in generalizing to unseen data, as evidenced by their performance on the test dataset. Notably, the ensemble model constructed using a Voting Classifier demonstrates comparable or superior performance to individual classifiers, highlighting the efficacy of ensemble learning in enhancing model robustness and accuracy.

6.2. Implications for Ransomware Detection and Cybersecurity Practices.

The findings have significant implications for ransomware detection and cybersecurity practices. The high performance of the ensemble learning ML model underscores its potential as a reliable tool for identifying ransomware instances amidst benign software. By leveraging diverse classifiers and combining their predictions, the ensemble model can effectively mitigate false positives and false negatives, thereby enhancing overall detection accuracy. Moreover, the ability of the ensemble model to generalize well to unseen data enhances its applicability in real-world cybersecurity scenarios, where encountering novel ransomware variants is commonplace.

6.3. Recommendations for Future Research.

While the ensemble learning ML model demonstrates promising results, there are opportunities for further research and refinement. Future studies could explore the integration of additional classifiers or feature selection techniques to enhance model performance further. Moreover, investigating the impact of ensemble size and composition on detection accuracy could provide valuable insights into optimizing ensemble model configurations for specific ransomware detection tasks. Additionally, ongoing monitoring and evaluation of the model's performance against evolving ransomware threats are essential to ensure its continued effectiveness in real-world settings.

6.4. Potential Areas for Improvement or Refinement of the Ensemble Model.

To improve the ensemble model's efficacy, researchers could focus on addressing potential challenges such as class imbalances or dataset biases that may affect model performance. Additionally, fine-tuning hyperparameters and optimizing model training procedures could lead to enhanced detection accuracy and efficiency. Furthermore, incorporating domain-specific knowledge and expertise into the model development process may facilitate the identification of unique ransomware characteristics and behaviors, thereby improving detection capabilities.

In conclusion, the ensemble learning ML model presents a promising approach to ransomware detection, offering high accuracy and robustness in identifying malicious software instances. By addressing current limitations and exploring future research avenues, the model holds considerable potential to advance cybersecurity practices and mitigate the impact of ransomware attacks on organizations and individuals.

REFERENCES

1. Zimba, A. Malware-Free Intrusion: A Novel Approach to Ransomware Infection Vectors. Int. J. Comput. Sci.Inform. Secur. 2017, 15, 317–325.
2. CyberPedia. What Is an Exploit Kit. 2018. Available online: Https://www.paloaltonetworks.com/cyberpedia/what-is-an-exploit-kit (accessed on 21 November 2018).
3. Zakaria, W.Z.A.; Mohd, M.F.A.O.; Ariffin, A.F.M. The Rise of Ransomware. In Proceedings of the 2017 International Conference on Software and e-Business, ICSEB 2017, Hong Kong, 28–3 December 2017; pp. 66–70.
4. Liska, A.; Gallo, T. Ransomware: Defending Against Digital Extortion, 1st ed.; O'Reilly M., Ed.; O'Reilly Media:
   Sebastopol, CA, USA, 2017.
5. Sophos Knowledge Base: Ransomware: How an Attack Works. 2016. Available online: Https://community.sophos.com/kb/en-us/124699 (accessed on 21 November 2018)
6. Al-rimy, B.A.S.; Maarof, M.A.; Shaid, S.Z.M. Ransomware threat success factors, taxonomy, and countermeasures: A survey and research directions. Comput. Secur. 2018, 74, 144–166.
7. Alrawashdeh, K.; Purdy, C. Ransomware detection using limited precision deep learning structure in fpga. In Proceedings of the NAECON 2018-IEEE National Aerospace and Electronics Conference, Dayton, OH, USA, 23–26 July 2018; pp. 152–157.
8. Urooj, 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.
9. Bahrani, A.; Bidgly, A.J. Ransomware detection using process mining and classification algorithms. In Proceedings of the 2019 16th International ISC (Iranian Society of Cryptology) Conference on Information Security and Cryptology (ISCISC), Mashhad, Iran, 28–29 August 2019; pp. 73–77.
10. Lee, K.; Lee, S.-Y.; Yim, K. Machine learning based file entropy analysis for ransomware detection in backup systems. IEEE Access 2019, 7, 110205–110215.
11. Kok, S.; Azween, A.; Jhanjhi, N. Evaluation metric for crypto-ransomware detection using machine learning. J. Inf. Secur. Appl. 2020, 55, 102646.
12. Pundir, N.; Tehranipoor, M.; Rahman, F. RanStop: A Hardware-assisted Runtime Crypto-Ransomware Detection Technique. arXiv 2020, arXiv:2011.12248.
13. Damodaran, A.; Di Troia, F.; Visaggio, C.A.; Austin, T.H.; Stamp, M. A comparison of static, dynamic, and hybrid analysis for malware detection. J. Comput. Virol. Hacking Tech. 2017, 13, 1–12.
14. Gharib, A.; Ghorbani, A. Dna-droid: A real-time android ransomware detection framework. In International Conference on Network and System Security; Springer: Cham, Switzerland, 2017; pp. 184–198.
15. Zhang, H.; Xiao, X.; Mercaldo, F.; Ni, S.; Martinelli, F.; Sangaiah, A.K. Classification of ransomware families with machine learning based on N-Gram of opcodes. Future Gener. Comput. Syst. 2019, 90, 211–221.