Development of a Stochastic Model for the Detection of Ransomware Malware Using Hybrid Analysis and Machine Learning Techniques

Akram Albanaa, Subrata Sahana, Jabir Ali

Department of Computer Science and Engineering, Sharda University, Greater Noida,

India, 201306

akramalbanaa2016@gmail.com , subrata.sahana@gmail.com , jabirali.cse@gmail.com

ABSTRACT

The ever-increasing number of ransomware with different signatures makes it more difficult to identify ransomware than other types of malware. This renders typical signature-based detection techniques ineffective against ransomware. Current ransomware detection techniques are usually used to distinguish ransomware from benign software. On the other hand, these techniques are not useful for differentiating between ransomware and other malware. In this paper, we build ransomware detection model using hybrid analysis and machine learning. First, we extract the file headers of the sample using static analysis; next, in dynamic analysis, we run the executable using a sandbox in order to extract the Windows API call and know what this software can do. The environment was used to analyze 324 ransomware and 320 other types of malwares along with 315 goodware samples. VirusShare was the source for ransomware and malware. Different models have been compared (Random Forest, k-NN, SVM, Decision Tree, AdaBoost), showing the Random Forest as the best with an accuracy of 96% and 0.018 false positive rate for multiclass classification. The methodology that has been detailed here contributes to the achievement of a greater detection accuracy and provides the capacity to recognize new ransomware.

Keywords: Ransomware, Malware, Static Analysis, Dynamic Analysis, Machine Learning

I. Introduction

These days, cybercriminals use cunning methods in order to develop new types of malwares that are more lucrative. One of these most recent forms of malware that has become widespread is ransomware. Ransomware is a kind of malware that secretly encrypts a victim's data without authorization [1]–[3]. The goal of this piece of malware is centered on limiting access to user data by encrypting those data and demanding a payment in order to get the key to decrypt them. Bitcoins are often demanded as the mode of payment by attackers because of the anonymity that is connected with this money [4]. There has been a considerable increase in the frequency of ransomware assaults, and ransomware variants have developed more complex methods for propagating themselves, encrypting data, and avoiding protective safeguards. Antivirus programs play an essential role in defense. They maintain hashes and signatures of known ransomware samples. It is a fast and user-friendly way of detecting. Hackers do everything they can to trick antivirus software by rewriting code, using polymorphic code, or developing new variants.

The seven steps of the ransomware lifecycle are as follows: creation, campaign, infection, command and control, search, encryption and extortion [5], [6].

Researchers in the area of ransomware analysis and detection do not differentiate between ransomware and other malicious programs. Instead, they only differentiate between ransomware and normal software. Different forms of malicious programs may have similar features; thus, these programs must be analyzed to determine what they can do.

The purpose of this paper is to investigate whether it is possible to detect ransomware from other malware and benign software using hybrid analysis and machine learning. Every machine learning issue is based on data. Static analysis has been used to extract the header file from the sample. Secondly, in contrast to static analysis, samples need to be executed. For that, a safe environment has been set up to enable dynamic analyses. To conclude, the results of some classifiers were compared.

The remaining parts of this paper as follows: The earlier studies that have been done in the field of ransomware detection are discussed in Section 2; The suggested strategy is described in Section 3; The experimental findings are presented in detail in Section 4, and the conclusions is provided in Section 5.

II. Related Work

The fight against ransomware is an arms race, so security experts must do ongoing research. The level of difficulty in ransomware detection is directly proportional to the amount of information known about the sample. Signature recognition can easily block known malware families. The evolution and technological progress in this domain are going very fast, and hackers try to use methods to stay under the radar. Polymorphic code is used to change the semantics of the code but keeping the original algorithm. Researchers are always putting forth effort to investigate and identify ransomware. This section will go through some of the most significant works.

Hwang et al [7]. build a ransomware detection approach in two stages. First, they use a Markov chain model of the API calls resulting in a low FPR. During the second stage, they try to expose the undetected ransomware by using a Random Forest algorithm. 1909 of ransomware and 1139 of normal software samples has been used. The model achieved 97.28% accuracy with an 4.83% FPR.

In [8] **Azween et al**. have come up with a pre-encryption detection technique in order to identify crypto ransomware. The PDEA algorithm employs two different levels of detection in its operation. First, it investigates whether there are any matches with the known ransomware signature. Next, it employs Random Forest to identify ransomware that has not yet been seen. This work got an accuracy of 99%.

Almomani et al. [9] presented a SMOTE-tBPSO-SVM model when they used SVM to the SMOTH classification procedure. The BPSO was included in order to enhance the effectiveness of the cost coefficient in the SVM. This work achieved an accuracy of 97.5%.

Zahoora et al. [10] suggested CSPE-R to identify new ransomware threats. It changes the dynamic feature space at the base into a more stable and core semantic feature space using the CAE. Next, it searches semantic spaces at different levels to discover robust features. This work gets 93% accuracy with a 0.01 false-positive rate.

Masum et al. [11] provide a methodology based on feature selection that employs many machine learning techniques, including neural network-based designs. The findings reveal that RF classifiers are more accurate than other approaches, with 99% accuracy.

In [12] **Talabani et la.** used Rule-Based algorithms to classify Bitcoin ransomware attacks using Bitcoin transaction data. The Bitcoin dataset included 61,004 addresses. Partial decision tree (PART) classification outperformed Decision Table classification in 96.01% accuracy.

III. Methodology

This section will describe the hybrid analysis and machine learning workflow, as presented in Figure 1. First, the available data is represented and where to find them. These data samples can't be put straight into a machine learning model. Instead, the data should be analyzed, and features must be extracted from the data. Second, we extract the file headers of the sample using static analysis; next, in dynamic analysis, we run the executable using a sandbox in order to extract the Windows API call. After preparing the dataset, data pre-processing was used to handle the missing data, transform the raw data, and split the dataset into a training set and a testing set. Feature selection used to remove unwanted feature and make the model focusing in the important feature only. In the last step of our model, 5 machine learning classifies (RF, DT, SVM, K-NN, AdaBoost) to perform the classification result.

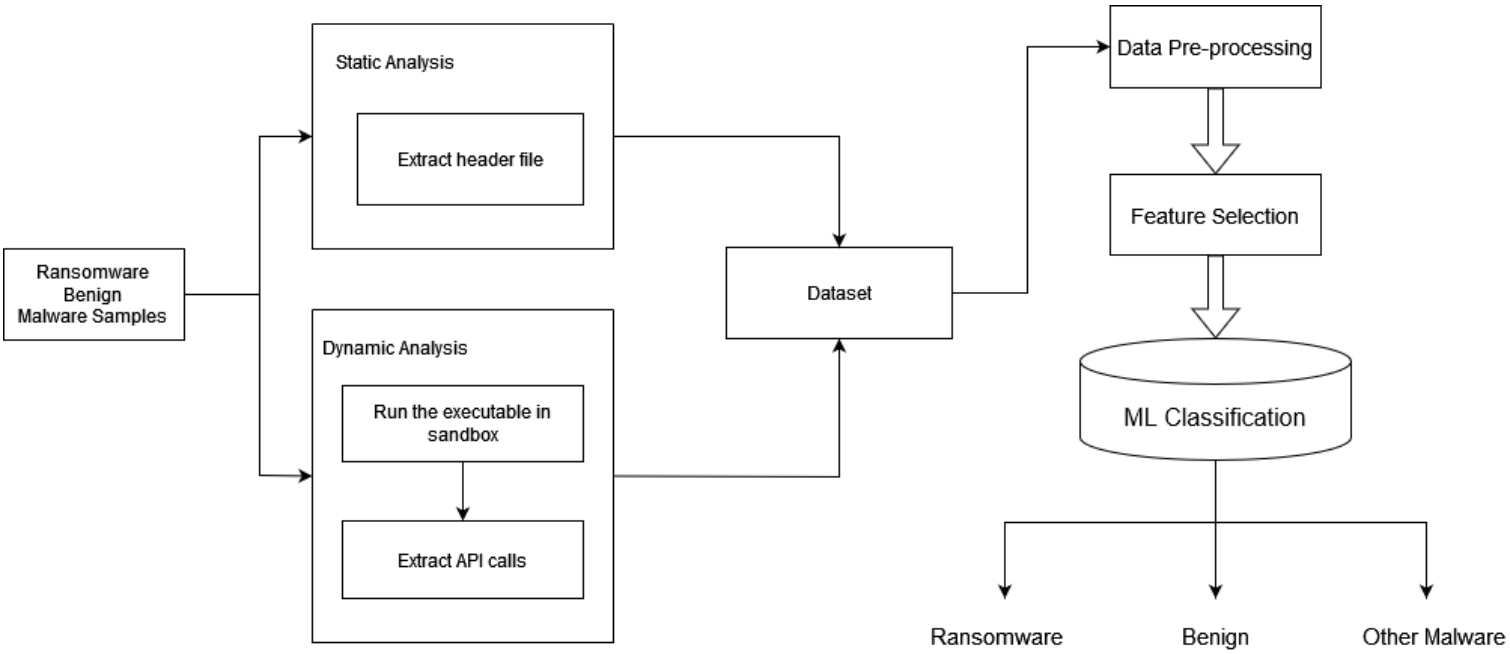

Fig. 1. Block diagram of the proposed methodology

A. Problem statement

Signature-based malware detectors can't find polymorphic malware, which can change its signatures, or new malware, for which signatures haven't been created yet. In recent studies, we've found that it distinguishes between ransomware and benign software only, and other types of malwares not taken into classification, so that means as example if there is trojan horse in the dataset it will classifieds as ransomware file.

B. Dataset description

There is no universal database for ransomware samples; researchers look online for their data from various sources. There are multiple organizations online that collect malware, such as theZoo, MalwareBazaar, VirusTotal, and VirusShare. We are using 959 samples that has been downloaded from VirusShare [13]. Out of these samples 324 Ransomware samples, 320 other malware samples, and 315 goodware samples as shown in Table 1. These files were in Portable Executables format. PE is a type of format that is used in Windows (both x86 and x64) [14]. Figure 2 shows the percentage of each class out of all the data. 33.8% ransomware, 33.4% other types of malwares, and 32.8% benign samples.

Table 1. Number of Data Samples

	Ransomware	Malware	Benign	Total
Number of Samples	324	320	315	959

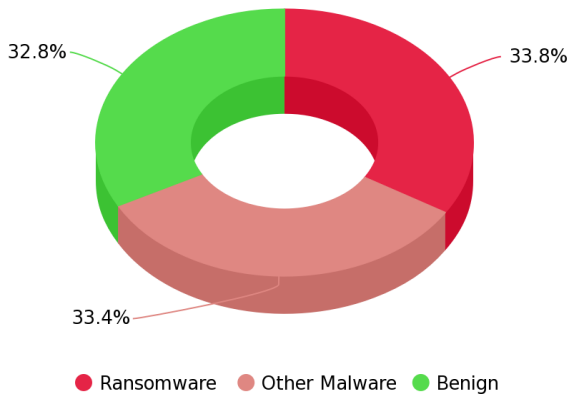

Fig. 2. Details of the dataset used

C. Static Analysis

It is a passive technique based on signature verification and source code that extracts the application's characteristics from the source code, or binary strings.

Object dump and PE parser tools are used for further investigation. Object dump is a Linux based tool to get the file headers of an object file [15]. PE parser tool helps to display import and export attributes of functions inside the code. The header file contains 45 features, which indicates the total amount of features that extracted with these tools.

Table 2. Features extracted in static analysis

Feature Name	Feature Name
MajorLinkerVersion	IMAGE_DIRECTORY_ENTRY_EXPORT
MinorLinkerVersion	MinorImageVersion
SizeOfCode	IMAGE_DIRECTORY_ENTRY_TLS
SizeOfInitializedData	Mean Entropy
SizeOfUninitializedData	IMAGE_DIRECTORY_ENTRY_SECURITY
SectionAlignment	IMAGE_DIRECTORY_ENTRY_RESOURCE
FileAlignment	MinorOSystemVersion
MajorOSystemVersion	IMAGE_DIRECTORY_ENTRY_IMPORT
ImageBase	SizeOfHeaders
MajorImageVersion	IMAGE_DIRECTORY_ENTRY_EXCEPTION
BaseOfCode	BaseOfData
AddressOfEntryPoint	IMAGE_DIRECTORY_ENTRY_BASERELOC
MinorSubsystemVersion	IMAGE_DIRECTORY_ENTRY_COPYRIGHT
Win32Version	MajorSubsystemVersion
SizeOfImage	IMAGE_DIRECTORY_ENTRY_LOAD_CONFIG
CheckSum	IMAGE_DIRECTORY_ENTRY_BOUND_IMPORT
Subsystem	IMAGE_DIRECTORY_ENTRY_IAT
DllCharacteristics	IMAGE_DIRECTORY_ENTRY_DELAY_IMPORT
SizeOfStackReserve	IMAGE_DIRECTORY_ENTRY_COM_DESCRIPTOR
SizeOfStackCommit	IMAGE_DIRECTORY_ENTRY_RESERVED
SizeOfHeapReserve	IMAGE_DIRECTORY_ENTRY_DEBUG
SizeOfHeapCommit	IMAGE_DIRECTORY_ENTRY_GLOBALPTR
Maximum_Entropy	

D. Dynamic Analysis

Dynamic analysis is a way for analyzing the behavior of malicious software during its execution. Cuckoo Sandbox used to launch malicious files in secure and isolated environment [16]. Dynamic analysis based on the API calls is used to determine what this file was created to achieve. We can tell whether a file is malicious based on its API calls. The Windows Application Programming Interface, often known as the WinAPI, is a collection of functions and procedures that may abstract a significant portion of the operations that you typically do on the Windows operating system [17]. Programmers have access to these functions via the Application Programming Interface (API), which allows them to pre-written procedures in situations when building their own may not be the most efficient option. With analyzing the samples, we have taken the common API calls used by ransomware and other malicious software as well as API calls used by benign samples. Total 49 API calls shown in the Table 3.

Table 3. Features extracted in dynamic analysis

API call Name	API call Name	API call Name
Sleep	CryptAcquireContextA	ChangeServiceConfig2A
FindFirstFileA	FindNextFile	CopyFileA
ExitProcess	SetFilePointer	LockResource
GetProcAddress	GetFileSize	recv
GetLastError	SetFileAttributes	WSAStartup
CloseHandle	GetFileAttributesA	GetAdaptersInfo
LoadLibraryA	VirtualAlloc	GetComputerNameA
GetStartupInfoA	CryptUnprotectData	CryptGenRandom
InternetOpen	GetSystemInfoA	CreateServiceA
RegDeleteKeyW	SystemParametersInfoW	StartServiceA
RegOpenKeyExW	SwitchDesktop	WSAGetLastError
RegCreateKeyExW	CreateWindowStationW	Socket
GetUserName	CryptDestroyKey	NtShutdownSystem
RegEnumKeyW	CryptImportKey	ShellExecute
CreateRemoteThread	CryptAcquireContextW	ExitWindowsEx
Process32First	Process32Next	CreateFileA
VirtualProtect		

E. Data Pre-processing and Feature Selection

In this context, "data pre-processing" refers to the processes of cleaning, converting, and integrating data in order to make it suitable for analysis. Data pre-processing used here to handling the missing value, after that splitting the data to 75% training set and 25% testing set, last step is scaling the features by converting different scales to a standard scale.

The process of isolating the characteristics that will be used in model creation that are the most consistent, non-redundant, and important is referred to as "feature selection". The major objective of the feature selection procedure is to improve the accuracy of a predictive model while reducing the amount of computational work necessary for the modeling process. Pearson's correlation method used to remove the features with high correction (features with correlation more than 85%). With applying feature selection, the number of features has been reduced from 94 to 85 features including header file and API calls. Table 4 contains the high correlated features that has been removed.

Table 4. High correlated features

Feature Name	Feature Type
CryptGenRandom	API
IMAGE_DIRECTORY_ENTRY_DELAY_IMPORT	File header
MinorSubsystemVersion	File header
Process32Next	API
SizeOfHeaders	File header
Subsystem	File header
WSAGetLastError	API

F. ML Classification

In this study, the detection of ransomware on Windows was handled by a total of five machine-learning classifiers. When these classifiers were put through their

paces, the primary objective was to evaluate how well they could identify ransomware when combined with data preparation and feature selection strategies. These are the ML classifiers:

- **Random Forest:**

The random forest algorithm is an ensemble approach comprised of many decision trees and bagging processes. Every tree is classified, and the categorization is completed by majority voting on the findings of the Decision trees. The most important parameters are max depth, which says how deep the tree can grow, and n estimators, which define the number of trees in the forest [18], [19]. In our research, the number of the trees in the forest was 50.

- **K- Nearest Neighbor:**

One of the supervised machine learning algorithms that can be utilized for both regression and classification is called K-NN. Input to the k-NN method is the k- closest training samples in the dataset, which in this study is equal to 5. The prediction for each test sample is computed using the Euclidean metric.

- **Support Vector Machine:**

SVMs are a group of supervised learning algorithms that may be used in a variety of contexts, including classification, regression, and the detection of outliers.

- **Decision Tree:**

It is a supervised ML sequential model that splits data according to a parameter and tests it, like a flow chart where the inner node represents a feature test. The leaf node has a class label, and each branch reflects a test result. Adding new sample features to the tree initiates the DT flow. To manage tree effectiveness, the max depth option is crucial.

- **AdaBoost:**

Adaptive boosting was the first really successful boosting strategy developed for binary classification. AdaBoost is used with decision trees that are quite short.

IV. Experiment

The approach that we have suggested distinguish ransomware from other forms of malware as well as legitimate programs. In order to demonstrate that our suggested strategy is successful, we carried out a number of tests.

A. Experimental environment

While conducting experiments, we make use of a computer that has a CPU with an Intel core i7, 16 gigabytes of RAM, and Ubuntu 22.04 as the operating system of the host machine and Windows 7 32-bit as the operating system of the guest machine.

B. Evaluation metrics

For Evaluation the efficiency of our proposed method, the following equations have been used:

Equation number (1) to calculate the true positive rate:

$$TPR = \frac{\mathrm{TP}}{\mathrm{TP+FN}} \quad (1)$$

To calculate the false positive rate by:

$$FPR = \frac{\mathrm{FP}}{\mathrm{FP+TN}} \quad (2)$$

To calculate the precision is given by equation number 3.

$$Precision = \frac{\mathrm{TP}}{\mathrm{TP+FP}} \quad (3)$$

Recall, also known as sensitivity, is a measurement of the proportion of true positives that are properly classified as true positive. The formula for calculating recall is as follows:

$$Recall = \frac{\mathrm{TP}}{\mathrm{TP+FN}} \quad (4)$$

To calculate accuracy with this method:

$$Accuracy = \frac{\mathrm{TP+TN}}{\mathrm{TP+FP+TN+FN}} \quad (5)$$

The F1 Score is determined by calculating the weighted mean of the Precision and Recall categories. The F1 score calculated using the following formula:

$$F1 - Score = 2 * \frac{(\mathrm{Precision} * \mathrm{Recall})}{(\mathrm{Precision+Recall})} \quad (6)$$

C. Results and discussion

Many works have been done for ransomware detection before. However, we have done the classification four times. In the first time of the classification, we used only two classes that are (ransomware and goodware samples) and compared that with the recent and notable studies that used same two classes in Table 5 and Figure 3. This table shows the type of features used, ML classification, accuracy, and FPR of the suggested work.

Table 5. Comparison our work against existing approaches for two classes.

Ref.	ML Classification	Feature	Accuracy	FPR
[7]	RF	API calls	97.2%	0.04
[8]	RF	API calls	99%	NA
[9]	SVM	API calls, Permissions	97.5%	NA
[10]	RF, SVM, LR, DNN	API calls, Registry Key, Strings, File directories	93%	0.01
[11]	RF, DT, NB, LR, NN	File header	99%	NA
[12]	DT	Bitcoin network transactions	90.01%	0.007
Our	RF, K-NN, SVM, DT, AB	File header, API calls	99.3%	0.004

RF: Random Forest, DT: Decision Tree, SVM: Support Vector Machine, AB: AdaBoost, LR: Logistic Regression, K-NN: K-Nearest Neighbors, DNN: Deep Neural Network. NB: Naive Bayes, NN: Neural Network.

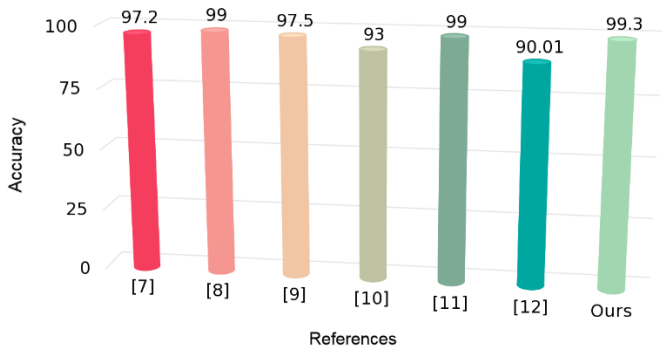

Fig. 3. Comparison the accuracy of our work against existing approaches for two classes.

The second classification is the primary work that we did with three classes (ransomware, benign, and other malicious software samples). The methods Random Forest, K-NN, SVM, Decision Tree, and AdaBoost were used to accomplish this task. Table 6 and Figure 4 presents the results of a comparison of these different methods.

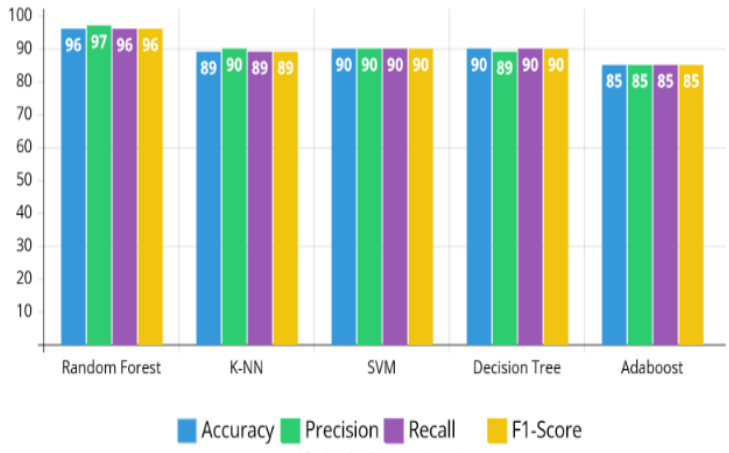

Fig. 4. Comparison the result of different ML methods.

As we see in the results random forest algorithm gets the best accuracy which is 96% and 0.018 false positive rate.

The Receiver Operating Characteristic (ROC) curve is a likelihood plot that illustrates how well a classification model performs at various thresholds. The curve is drawn between two variables, namely TPR and FPR. The ROC curves of several machine learning techniques are shown in figure 5. where class 0 means benign, class 1 means other malware and class 2 means ransomware.

The Area Under the Curve (AUC) is a statistic that is used to summarize the ROC curve. This measure is used to determine how well a binary classifier can differentiate between different classes. When AUC is equal to one, the classifier is able to make an accurate distinction among all positive and negative possible values. If the AUC had been 0, then when it made its predictions, all negatives would have been considered positives and all positives would have been considered negatives.

Table 6. ML algorithms' evaluation for hybrid analysis with multi class classification.

ML Classifier	Precision	Recall	F1-Score	Accuracy (%)	TPR	FPR
Random Forest	0.97	0.96	0.96	96	0.963	0.018
K-NN	0.90	0.89	0.89	89.7	0.886	0.055
SVM	0.90	0.90	0.90	90	0.89	0.051
Decision Tree	0.90	0.90	0.90	90	0.896	0.049
AdaBoost	0.85	0.85	0.85	85	0.846	0.075

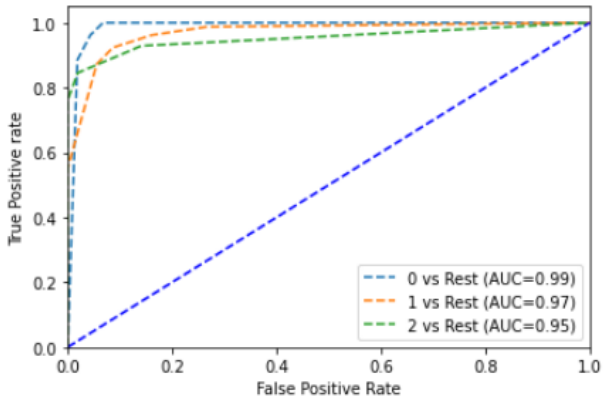

Fig. 5.(A) Random Forest Multiclass ROC curve

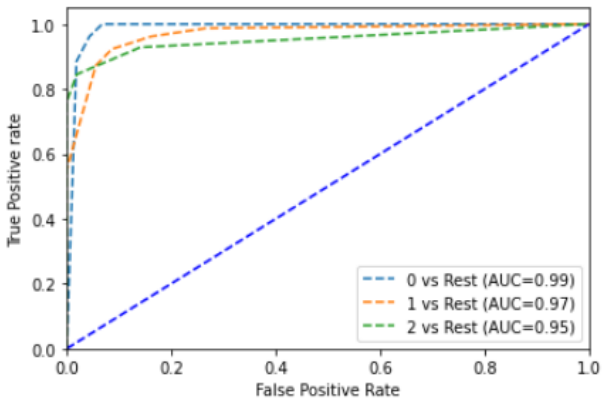

Fig. 5.(B) K-NN Multiclass ROC curve

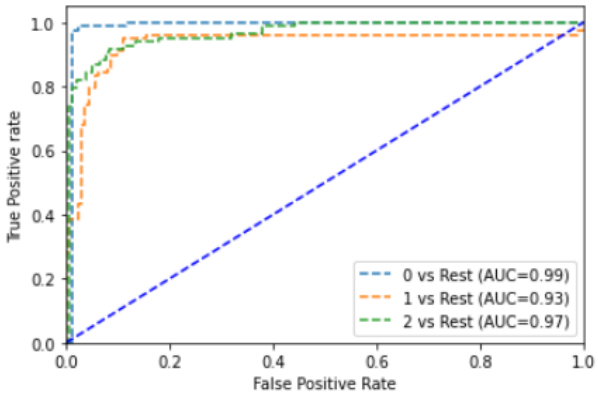

Fig. 5.(C) SVM Multiclass ROC curve

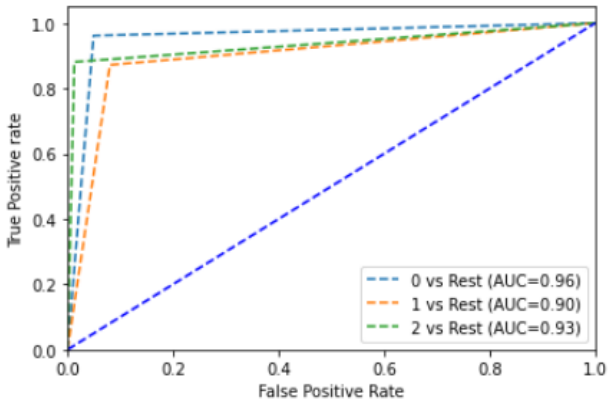

Fig. 5.(D) Secision Tree Multiclass ROC curve

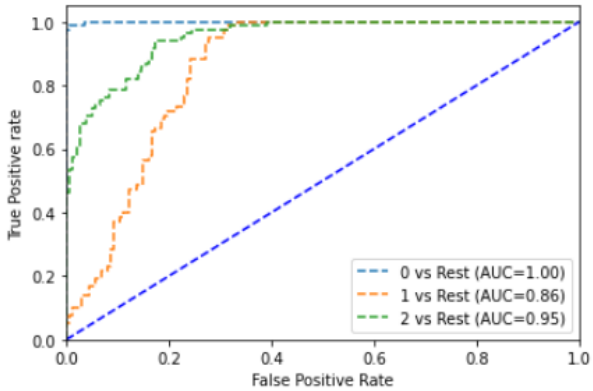

Fig. 5.(E) AdaBoost Multiclass ROC curve

Fig. 5. Comparison the ROC curve of different ML methods.

In Figure 5.A, we see the Receiver Operating Characteristic Curve of the Random Forest algorithm for each class against other classes. The AUC result of class benign is 1, class malware is 0.99, and class ransomware is 0.99, which means the model is perfect.

ROC curve for the K-NN model as shown in figure 5.B, the result of the AUC in class benign is 0.99, class malware is 0.97, and class ransomware is 0.95. The average AUC for all classes is 0.97, indicating that the model is also perfect.

Figure 5.C shows the ROC curve for all classes using support vector machines (SVM). The result of the AUC for class benign is 0.99, for class malware it is 0.93, and for class ransomware it is 0.97. The result means the model is perfect, with an average of 0.96.

The ROC curve of the Decision Tree in figure 5.D shows that the AUCs of classes benign and ransomware are perfect, while the AUC of class malware is very good. The average AUC of all classes is 0.92.

Table 7. Random forest evaluation for each class.

Class	Precision	Recall	F1-Score	TPR	FPR	AUC	Support
Benign	1	1	1	1	0	1	78
Other Malware	0.90	1	0.95	1	0.055	0.99	78
Ransomware	1	0.89	0.94	0.89	0	0.99	84

The ROC curve for the AdaBoost model is shown in Figure 5.E. The result of AUC in class benign is 1, and class ransomware is 0.95, which means perfect, while the AUC of class malware is 0.86. The average AUC for all classes is 0.93.

We have compared the performance of the random forest approach for each class of data in Table 7. These results include precision, recall, f1-score, true positive rate, false positive rate, area under curve, and the number of test samples in each class. As we can see in confusion matrix of random forest algorithm in figure 6, the error between benign class and other classes is zero. While the error rate presented only between ransomware class and other malware class.

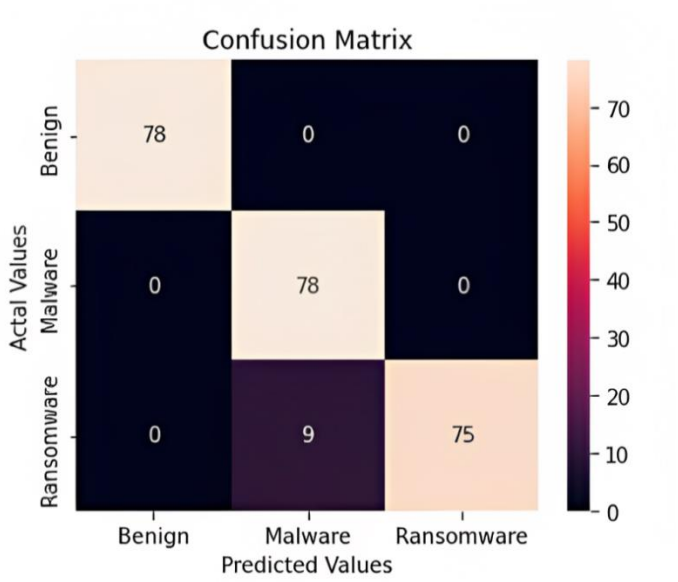

Fig. 6. Multi class confusion matrix using random forest algorithm.

We have applied the same models using static features (file header) alone and dynamic features (API calls) alone and compared the result when using the both features together as hybrid features in figure 7. Using hybrid analysis gets higher accuracy against using static and dynamic only for multi class classification. For classifying between two types of malwares we have to execute the sample to know what this program might be designed to do.

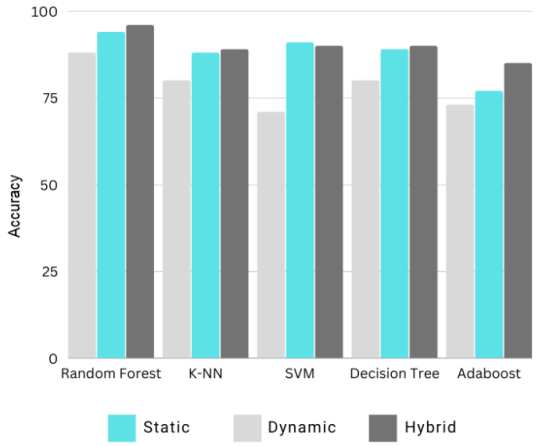

Fig. 7. Comparison the accuracy of each model using static, dynamic and hybrid analysis.

V. Conclusion

This paper has aimed to investigate the extent to which it is possible to distinguish ransomware from goodware and other malware through hybrid analysis. Static analysis used to extract file header and dynamic analysis used to execute the samples to extract the API calls. Ransomware can cause enormous damage if not handled with caution. A notable contribution to the research domain was creating a safe research environment. VirtualBox allows creating a safe research environment using any operating system. Cuckoo Sandbox helps in the automation of the analyzes. Together they assure the necessary virtualization. The environment was used to analyze 324 ransomware and 320 other types of malwares along with 315 goodware samples. VirusShare was the source for ransomware and malware. Object dump and PE parser used to extract the file header of the samples, while sandbox generates a report for each sample. These reports contain information about the executed code's behavior, for example, statistics about the API calls. Different models have been compared, showing the Random Forest as the best with an accuracy of 96% and 0.018 false positive rate. The technique that has been described helps achieve higher detection accuracy and offers the capability to identify new ransomware.

References

[1] D. Stiawan, S. M. Daely, A. Heryanto, N. Afifah, M. Y. Idris, and R. Budiarto, "Ransomware detection based on opcode behaviour using k-nearest neighbours algorithm," *Information Technology and Control*, vol. 50, no. 3, 2021, doi: 10.5755/j01.itc.50.3.25816.

[2] S. Alsoghyer and I. Almomani, "Ransomware detection system for android applications," *Electronics (Switzerland)*, vol. 8, no. 8, 2019, doi: 10.3390/electronics8080868.

[3] R. Almohaini, I. Almomani, and A. Alkhayer, "Hybrid-based analysis impact on ransomware detection for android systems," *Applied Sciences (Switzerland)*, vol. 11, no. 22, 2021, doi: 10.3390/app112210976.

[4] B. Jethva, I. Traoré, A. Ghaleb, K. Ganame, and S. Ahmed, "Multilayer ransomware detection using grouped registry key operations, file entropy and file signature monitoring," *J Comput Secur*, vol. 28, no. 3, 2020, doi: 10.3233/JCS-191346.

[5] S. H. Kok, A. Abdullah, N. Z. Jhanjhi, and M. Supramaniam, "Ransomware, Threat and Detection Techniques: A Review," 2019.

[6] B. A. S. Al-Rimy *et al.*, "A Pseudo Feedback-Based Annotated TF-IDF Technique for Dynamic Crypto-Ransomware Pre-Encryption Boundary Delineation and Features Extraction," *IEEE Access*, vol. 8, 2020, doi: 10.1109/ACCESS.2020.3012674.

[7] J. Hwang, J. Kim, S. Lee, and K. Kim, "Two-Stage Ransomware Detection Using Dynamic Analysis and Machine Learning Techniques," *Wirel Pers Commun*, vol. 112, no. 4, 2020, doi: 10.1007/s11277-020-07166-9.

[8] S. H. Kok, A. Azween, and N. Z. Jhanjhi, "Evaluation metric for crypto-ransomware detection using machine learning," *Journal of Information Security and Applications*, vol. 55, 2020, doi: 10.1016/j.jisa.2020.102646.

[9] I. Almomani *et al.*, "Android Ransomware Detection Based on a Hybrid Evolutionary Approach in the Context of Highly Imbalanced Data," *IEEE Access*, vol. 9, 2021, doi: 10.1109/ACCESS.2021.3071450.

[10] U. Zahoora, A. Khan, M. Rajarajan, S. H. Khan, M. Asam, and T. Jamal, "Ransomware detection using deep learning based unsupervised feature extraction and a cost sensitive Pareto Ensemble classifier," *Sci Rep*, vol. 12, no. 1, Dec. 2022, doi: 10.1038/s41598-022-19443-7.

[11] M. Masum, M. J. H. Faruk, H. Shahriar, K. Qian, D. Lo, and M. I. Adnan, "Ransomware Classification and Detection with Machine Learning Algorithms," in *2022 IEEE 12th Annual Computing and Communication Workshop and Conference, CCWC 2022*, 2022. doi: 10.1109/CCWC54503.2022.9720869.

[12] H. S. Talabani and H. M. T. Abdulhadi, "Bitcoin Ransomware Detection Employing Rule-Based Algorithms," *Science Journal of University of Zakho*, vol. 10, no. 1, 2022, doi: 10.25271/sjuoz.2022.10.1.865.

[13] "VirusShare. [Online]. Available: https://virusshare.com/login/torrents/VirusShare_CryptoRansom_20160715."

[14] N. A. Azeez, O. E. Odufuwa, S. Misra, J. Oluranti, and R. Damaševičius, "Windows PE malware detection using ensemble learning," *Informatics*, vol. 8, no. 1, 2021, doi: 10.3390/informatics8010010.

[15] Michael Kerrisk, "Object dump tool. [Online] Available: https://man7.org/linux/man-pages/man1/objdump.1.html," 2022.

[16] "Cuckoo automated malware analysis. [Online] Available: https://cuckoosandbox.org."

[17] "Windows API. [Online] Available: https://learn.microsoft.com/en-us/windows/win32/apiindex/windows-api-list."

[18] J. Saxe and H. Sanders, *Malware data science : attack detection and attribution*. No Starch Press, Inc., 2018.

[19] F. A. Narudin, A. Feizollah, N. B. Anuar, and A. Gani, "Evaluation of machine learning classifiers for mobile malware detection," *Soft comput*, vol. 20, no. 1, 2016, doi: 10.1007/s00500-014-1511-6.

[20] Clarence. Chio and D. Freeman, *Machine Learning and Security: Protecting Systems with Data and Algorithms.* O'Reilly Media, Inc., 2018.