Sorting Ransomware from Malware Utilizing Machine Learning Methods with Dynamic Analysis

Joshua Schoenbachler*
jjschoen@iu.edu
Indiana University - Purdue University Indianapolis
Indianapolis, Indiana, USA

Vinay Krishnan*
vinayek2@illinois.edu
University of Illinois Urbana-Champaign
Urbana, Illinois, USA

Garvit Agarwal
gagarwa@iu.edu
Indiana University - Purdue University Indianapolis
Indianapolis, Indiana, USA

Feng Li
fengli@iupui.edu
Indiana University - Purdue University Indianapolis
Indianapolis, Indiana, USA

ABSTRACT

Ransomware attacks have grown significantly in the past dozen years and have disrupted businesses that engage with personal data. In this paper, we discuss the identification of ransomware, malware, and benign software from one another using machine learning techniques. We collected data samples from repositories on the internet as well as using a dataset from a previous study that provided a basis for our approach. We also collected ransomware, malware, and benign software samples manually from Cuckoo Sandbox™. We also filtered on certain feature groups to test to see if certain activity/processes in the infection process could be used to correctly distinguish ransomware from malware and benign software. These feature groups represent correlated processes within a running application: network activity, PROC memory activity, registry/events processes, and file interactions. The datasets were analyzed using several ML models which included Random Forest, SVM, Gradient Boosting, and Decision Trees using binary classification. The best classifiers for distinctly identifying ransomware from benign software were Random Forest and SVC with an F1- score of 86% and an F1-score of 82% as well as an 85% in overall accuracy for Random Forest. In addition to ransomware versus benign software, we also compared malware software to ransomware data. Yielding a 100% accuracy in performance, Gradient Boosting Classifier and Decision Trees were the best at distinguishing ransomware from malware software. This high result may partially be caused by a smaller malware and ransomware dataset. Overall, we were able to successfully distinguish ransomware from malware and benign software.

*Both authors contributed equally to this research.

ACM MobiHoc '23, October 23–26, 2023,, Washington D.C., MD

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

KEYWORDS

Dynamic Analysis, Malware, Ransomware, Benignware, Cuckoo Sandbox, Graph Learning, Machine Learning, Neural Networks

ACM Reference Format:
Joshua Schoenbachler, Vinay Krishnan, Garvit Agarwal, and Feng Li. 2023. Sorting Ransomware from Malware Utilizing Machine Learning Methods with Dynamic Analysis. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (ACM MobiHoc '23).* ACM, New York, NY, USA, 7 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Currently, malware and ransomware-based detection systems have utilized static and dynamic analysis methodologies to collect information on generalized malware and ransomware behavior. Static analysis methods have focused on anti-malware systems to check files against known and common "malicious" signatures. Static analysis methods may have limitations as new or obfuscated files may not be detected, as a signature may not exist… As malware continues to evolve and becomes more pervasive, static analysis methods have become less effective, and security researchers have shifted their focus to dynamic detection methods to better prevent malware attacks. By using a wide range of features, we have been able to detect and cluster new or obfuscated malicious files unlike the previous static systems.

In the past, researchers tended to focus on either detecting malware or ransomware. However, a real-time system would be more effective in a production environment, if it is able to detect and differentiate malware and ransomware from benign applications. Security professionals and anti-malware services will need to react to ransomware differently relative to other malicious attacks. It is necessary to distinguish ransomware from other malicious attacks to prevent disruption in business systems and stifle revenue streams.

In this paper, the focus is on dynamic analysis using features collected from a Cuckoo Sandbox™ environment. This is an isolated system in which files are run to detect API calls, assess how memory and storage is analyzed, and determine how the computer network capabilities are utilized. Using previous techniques developed by

Hernández-Álvarez et. al. [1] the number of important sandbox data features was reduced to 50 critical features. Reducing the number of features allowed for rapid and efficient analysis of the files without compromising test scores.

This paper also addresses the process of retrieving and analyzing the 50 features collected from the cuckoo analyses, along with how the machine learning algorithms were developed and tested to differentiate and separate general malicious files from ransomware files. The unique aspect of this research is to underscore the utility of dynamic models and to identify and validate minimal features evident in the signature of suspicious ransomware to proactively reduce the impact caused by these malicious software.

2 RELATED WORKS

Earlier research leverages methods utilizing a dynamic analysis and machine learning-based approach to extract features that generalize the overall behaviors of ransomware activity. The study by Hernández-Álvarez et. al. [1], utilizes diverse ransomware-affected samples to detect variant threats. They classify Locker, Encryptor, and Goodware samples. The final dataset accumulates up to 2000 samples in entry with 50 characteristics extracted.

A survey paper by U Urooj et. al. [5] discusses the contextual background of ransomware-based research. This paper provides the history of ransomware research and an overview of different approaches researchers have utilized to analyze ransomware and machine learning techniques to further classify ransomware. This paper also focuses on the significance of dynamic analysis as a tool for ransomware detection.

U. Zahoora et. al. [7] focuses on methods to counter zero day ransomware attacks which are the most unpredictable in behavior because they are unseen in data prediction. A Pareto Ensemble Classifier was used to counter against zero day. This method utilizes a Contractive AutoEncoder to define a core semantic feature space. To learn these features, the Pareto Ensemble Classifier technique is used as a way to learn various feature spaces. The proposed Pareto Ensemble Classifier technique is proven to be very effective when performing against zero day ransomware attacks.

Many studies have utilized graph learning techniques for classification on dynamic graphs. A. Pareja et. al. proposed [4] a graph convolution network model along the temporal dimension to capture dynamism of a graph sequence using a recurrent neural network architecture to evolve GCN parameters. This proposed method was coined as EvolveGCN™. Such an advanced graph learning technique was primarily utilized for link prediction, node classification, and edge classification tasks.

Graph learning techniques and malware detection systems were combined to create a much more robust and faster learning model to increase accuracy in classification. S Li et. al. [2] combines both methods to increase malware detection performance. A malware detection based classifier on a graph convolutional network(GCN) was used to adapt to various malware characteristics by utilizing an API call sequence into a directed graph. The individual nodes represent the APIs and edge connections represent the API invokes. The weight of each graph represents the number of times each API sequence was called. The graph was read as an adjacency matrix, where the rows represent the API, started and the columns refer to the API called. The weight of the matrix is stored in the cell of the matrix. The resulting GCN framework yielded a 98.32

Z Zhang et. al. [8] relies on another graph learning method by focusing on the Dynamic Evolving Graph Convolutional Network (DEGCN) model to capture dynamic evolving patterns of both local API level and global graph level software behaviors. Essentially, a graph is generated to feature different time slots based on an API segment window which are run multiple times to eventually detect whether certain software is malicious or benign. This study involves a Graph-encoding-based Gate Recurrent Unit (GGRU) network to capture the graph level evolving features which include the nodes, edges, and their attributes.

3 METHODS

3.1 Datasets

3.1.1 Dataset One - 50 Features Dataset. Utilized a 2000 sample dataset that distinguished between Ransomware and Goodware. Families of ransomware were not explicitly given and 50 features were extracted from a JSON file through a sandboxed based environment [1].

3.1.2 Dataset Two - Collected Dataset for Ransomware, Malware, and Benign Software. 257 sampled dataset were collected from the Cuckoo Sandbox™ environment. Fifty features were extracted and tested for Ransomware (57), Malware (120), and Benign software (80) samples accordingly. Families of Ransomware were specified in the data collection.

3.2 Using Cuckoo Sandbox™

The initial dataset was collected by running software samples within the Cuckoo Sandbox™ environment. To initiate the cuckoo sandbox™ installation, an Ubuntu Linux machine was utilized as the foundation. Then VirtualBox was installed along with other required software for the Cuckoo Sandbox version 2.0.7. Next, we installed Cuckoo Sandbox™ and set up the virtual machine. For the testing environment, Windows 7 was utilized with modifications to make it more vulnerable. Python 2 was installed to run the files. Once the Cuckoo Sandbox™ was configured the files could be run in Python. The internet was scoured to find malicious and benign software. Several repositories were identified that aided our research. First was the Zoo [6], a popular malware and ransomware github project that hosts both general malware and ransomware samples. The second repository was NTFS123's MalwareDatabase [3]. This repository is almost identical to theZoo, but has some variations in samples. Using these two repositories we were able to procure enough samples to generate our dataset, Dataset 2.

3.3 Feature Extraction

After running the cuckoo samples, python programming interface [1] was utilized to extract 50 features from each malware sample. These features were selected based on certain characteristics of a program that were affected in the infection process while the malware was running. The selected features were extracted from a JSON report contained within the cuckoo report that was generated after a sample was run. The python program generated a GUI tool that presented the 50 final features as a list of checkboxes. As

a result, certain features could be filtered out that were related to specific processes such as network activity, registry /events, file processes, etc. After each extraction was complete, a .csv for each malware sample was generated displaying information on the features selected. The headers represented the features and the number associated with each header represented the number of occurrences of each process during the malware analysis. A total of 257 samples were run consisting of Malware, Ransomware, and Goodware. Each sample representing a .csv file was generated. All .csv files were combined into one to represent a dataset set for machine learning analysis. This dataset represented Dataset Two. We also utilized a GitHub repository [1] to serve as a training set for both Benign Software versus Ransomware analysis which is represented as Dataset one.

3.4 Feature Analysis

After converting our JSON files into .csv programs, all samples were analyzed by each feature through scatter plot graphs. We started with a large, but unusable graph that contained all 50 features, and broke it down into the following categories:

The first category created was the networking category. Out of the network features that are documented by cuckoo sandbox and are within the 50 features we collected from the json file, only three were noteworthy during this analysis. The three noteworthy features were hosts, dns-servers, and udp connections as displayed in Figure 1. Each of these features are distinctly shown in each category with udp being accessed most in benign software, dns servers being accessed most by malware, and hosts being utilized by ransomware.

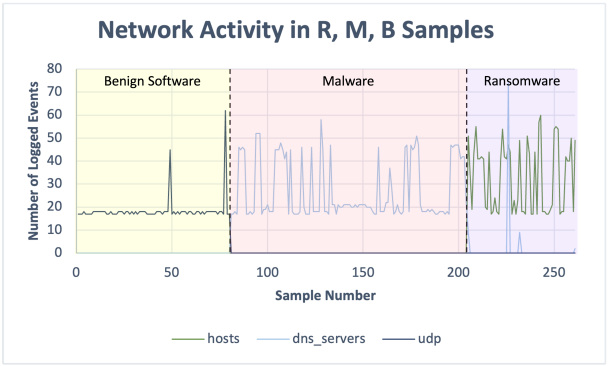

Figure 1: Most expressed networking features displayed. (R = Ransomware, B = Benign software, and M = Malware Samples)

The next category created was the api and dll calls. These are most commonly utilized for ransomware and malware detection by other researchers as shown in these sources [1, 2, 7, 8], hence it is important to monitor them more rigorously. In Figure 2, five features are identified to be notable, API, apistats, dll, dll-loaded, and importer-dll-count. API which is the amount of api calls made, apistats reflects the statistics about those api calls. The dll feature is the amount of dll's that are called, dll-loaded is the number of dlls that were run by the code, and were able to be found through static analysis, while importer-dll-count refers to dll's that were imported during dynamic analysis which is normally an obfuscation technique. In this analysis features unique to benign software were dll, features added with malware were api calls and dll loading, and features intensified with ransomware being used logged occurring more frequently than with malware.

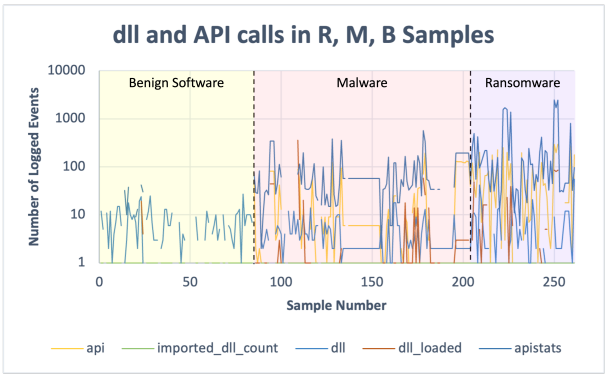

Figure 2: Most expressed API and dll features displayed. (R = Ransomware, B = Benign software, and M = Malware Samples)

Another important category that was developed in this research was registry events. As shown in Image 3, this graph consisted of registry reads, registry writes, and registry accesses with the registry-opened stat. We also included command-line to show commands within the program that are sent directly to windows and it integrates with the registry events. We see little activity from any of these features in benign software, though we see a lot of registry openings and command line operations in malware, and in ransomware we see registry reads and writes along with the registry opening and command line events that we also saw in malware.

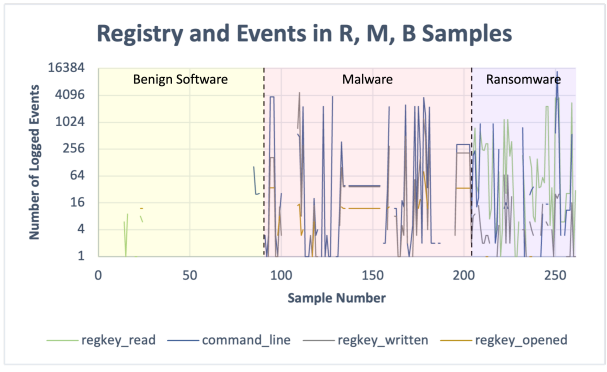

Figure 3: Most expressed Registry and Events features displayed. (R = Ransomware, B = Benign software, and M = Malware Samples)

The next major category analyzed was File Interactions. Total files referenced, file paths referenced, and files read and created

were reviewed. In malware a significant amount of action with file reading is identified, but with ransomware all files referenced in this category are relevant. From a visual standpoint, this may be the most promising category for detecting and distinguishing ransomware from malware.

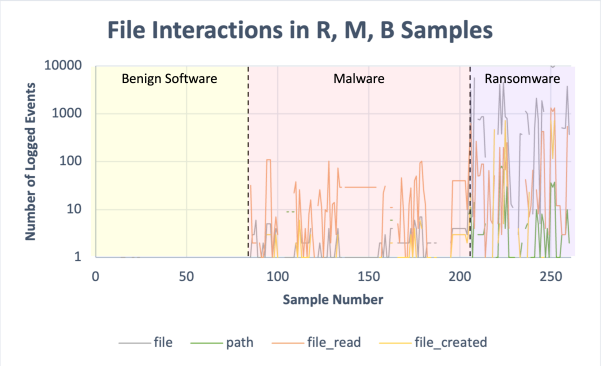

Figure 4: Most expressed file features displayed. (R = Ransomware, B = Benign software, and M = Malware Samples)

3.5 Machine Learning

Machine Learning was tested from a Random Forest Classifier, Gradient Boosting Classifier, Decision Tree Classifier, and Support Vector Machine (SVM) classifier. Binary classification was performed on each training and testing set for both malware and ransomware analysis and benign software and ransomware analysis. Performance on all 50 features was conducted as well as additional training and testing from selected features. Features selected included: process identifier (proc-pid) to represent PROC memory; registry key read (regkey-read), command line (command-line), registry key written (regkey-written), and registry key open (regkey-open) to represent registry and event processes; hosts, dns servers(dns-servers), and udp to represent network activity; file, path, file types (filetype), file readings (file-read), and file creations (file-created) to represent file interactions; and API calls in question (api), number of imported dynamic link libraries (imported-dll-count), number of system DLL libraries used by the malware during analysis (dll), and APIStats (apistats) to represent API and DLL activity. We chose to filter from these features to test if certain isolated processes could yield improved classification. Accuracy scores were computed for ransomware and malware comparison, benign software and ransomware comparison, ransomware and malware comparison with selected features, and benign software and ransomware with selected features. Each classifier computed a precision score with labels specified. For example, Malware and Ransomware Analysis showed a precision score that was just specified towards Malware and just specified towards Ransomware. The same label calculations were specified for Recall and F1-scores as well. Simply stated each classifier calculated an accuracy for performance as well as a precision, recall, and f1-score with proper label specifications. The following modules, scores, and classifiers were imported from a scikit python learning module for testing and training.

4 RESULTS AND DISCUSSION

4.1 Ransomware Versus Malware

4.1.1 All 50 Features. In Table 1, this dataset was primarily tested with 56 malware and 56 ransomware samples with training and testing being split. The 112 samples came from Dataset Two which consists of samples that were collected manually. Best performing classifiers are Random Forest, Decision Tree, and Gradient Boosting showing a 100% accuracy. The worst performing is SVC, showing low model accuracy (0.609) but no false positive identification rate in ransomware labeling. In machine learning, typically a 100% is a score that is conjectural in nature. However, since our sample size for training and testing is relatively small and malware and ransomware are very distinct in nature, proven from our Graph analysis data, it is possible that such scores would appear in accuracy, precision, recall, and f1 scores. From this data alone, each classifier computed no false positive for ransomware labeling and scored a 100% at predicting a positive rate regarding recall on malware. With regards to precision on malware, three out of the four models score no false positives with the exception of SVC (0.571). Recall on ransomware is almost perfect; however, SVC fails to predict ransomware activity through 0.18 score. F1-scores are similar for ransomware and malware labeling; however, all four classifiers are better at classifying malware than ransomware activity due to SVC scoring 0.73 on malware labeling.

Model	Accuracy	Precision on R	Precision on M	Recall on R	Recall on M	F1-Score on R	F1-Score on M
Random Forest	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Decision Tree	1.0	1.0	1.0	1.0	1.0	1.0	1.0
SVC	0.608695652	1.0	0.571428571	0.181818182	1.0	0.307692308	0.727272727
Gradient Boosting	1.0	1.0	1.0	1.0	1.0	1.0	1.0

Table 1: R = ransomware as true positive label, M = malware with true positive label

4.1.2 Feature Selection Testing. Featuring network activity with Table 2, all four classifiers were perfect in accuracy, precision, recall, and f1-score. Even though a 100% score is very unrealistic in a large dataset setting, it is possible with the small sample size as is presented in this particular dataset. Similar logic can be applied to Table 1. This can be considered the best for prediction for classification though somewhat speculative in performance. These scores amongst all classifiers show no false positives with precision scores of 100%. Additionally, a 100% positive prediction rate amongst both labels of ransomware and malware is present as well. These scores show that ransomware is most definitely distinguishable from malware regarding network activity.

Model	Accuracy	Precision on R	Precision on M	Recall on R	Recall on M	F1-Score on R	F1-Score on M
Random Forest	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Decision Tree	1.0	1.0	1.0	1.0	1.0	1.0	1.0
SVC	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Gradient Boosting	1.0	1.0	1.0	1.0	1.0	1.0	1.0

Table 2: R = ransomware as true positive label, M = malware with true positive label

In Table 3, Dynamic Link Library and API activity yielded relatively high scores in performance with three out of four models scoring above an 85% in accuracy. There is also a zero false positive rate regarding ransomware performance. There is a relatively

low false positive performance in precision testing for malware, especially in Decision Tree and Gradient Boosting; however, SVC scores are high in false positive rate. The percentage of correctly identified positive labels in malware labeling is 100% for all four models. Similarly, the same can be said for three out of the four models in Random Forest, Decision Tree, and Gradient Boosting amongst ransomware labeling with SVC performing at its lowest. SVC performs the worst in this particular testing set. Regarding F1-scores, for DLL and API-based features there is better model performance in malware identification as a true positive label rather than ransomware which is shown through higher F1-scores. The best performing models are Decision tree (accuracy = 96% and f1 >= 0.95) and Gradient Boosting (accuracy = 96% and f1 =0.95) with random forest coming in third (accuracy = 87% and f1 = 0.89). Due to variability in numerical data, it would be appropriate to test from other models as well to yield higher scores in numerical data overall.

Model	Accuracy	Precision on R	Precision on M	Recall on R	Recall on M	F1-Score on R	F1-Score on M
Random Forest	0.869565217	1.0	0.8	0.727272727	1.0	0.842105263	0.888888889
Decision Tree	0.956521739	1.0	0.923076923	0.909090909	1.0	0.952380952	0.96
SVC	0.565217391	1.0	0.545454545	0.090909091	1.0	0.166666667	0.705882353
Gradient Boosting	0.956521739	1.0	0.923076923	0.909090909	1.0	0.952380952	0.952380952

Table 3: R = ransomware as true positive label, M = malware with true positive label

In Table 4, file processes yielded very similar scores in model accuracy for three out of the four models in this particular testing set with SVC as the exception with 65%. With a model accuracy of 91% on Random forest, Decision tree, and Gradient Boosting, the classifiers are shown to have a very high performance in testing overall. There is also no false positive prediction in ransomware labeling which is very optimal. There is also a very low false positive rate on malware labeled data with SVC as the exception again given a 0.6 precision score. SVC overall is performing the worst consistently throughout most of these tests in ransomware versus malware performance which is seen in Table 1, 3, 4 and 5. There is a 100% positive label prediction rate regarding recall on malware samples. Similarly, there is also a very high positive prediction rate with SVC performing the worst.

Model	Accuracy	Precision on R	Precision on M	Recall on R	Recall on M	F1-Score on R	F1-Score on M
Random Forest	0.913043478	1.0	0.857142857	0.818181818	1.0	0.9	0.923076923
Decision Tree	0.913043478	1.0	0.857142857	0.818181818	1.0	0.9	0.923076923
SVC	0.652173913	1.0	0.6	0.272727273	1.0	0.428571429	0.75
Gradient Boosting	0.913043478	1.0	0.857142857	0.818181818	1.0	0.9	0.923076923

Table 4: R = ransomware as true positive label, M = malware with true positive label

For registry and event activity in Table 5, three out of four models show at least 85% in model accuracy with SVC performing the lowest with 57%. Decision Tree(96%) and Gradient Boosting (96%) perform the best in classification regarding metrics in registry and event. There is no false positive identification in ransomware labeling and a 100% recall performance amongst all four models in malware identification. Three out of four models again seem to show a low false positive rate with SVC as the exception with malware labeling. Regarding recall on ransomware labeling, there is a very high sensitivity among Random Forest (precision score = 0.73), Gradient Boosting (precision score = 0.91), and Decision Tree (precision score = 0.91) data. Overall, there is better classification on malware labeling than ransomware given F1-scores.

Model	Accuracy	Precision on R	Precision on M	Recall on R	Recall on M	F1-Score on R	F1-Score on M
Random Forest	0.869565217	1.0	0.8	0.727272727	1.0	0.842105263	0.888888889
Decision Tree	0.956521739	1.0	0.923076923	0.909090909	1.0	0.952380952	0.96
SVC	0.565217391	1.0	0.545454545	0.090909091	1.0	0.166666667	0.705882353
Gradient Boosting	0.956521739	1.0	0.923076923	0.909090909	1.0	0.952380952	0.96

Table 5: R = ransomware as true positive label, M = malware with true positive label

4.2 Ransomware Versus Benign Software

4.2.1 All 50 Features. In Table 6, we compare ransomware to benign software samples. Training was run on Dataset One and testing was run on Dataset Two which consists of 137 samples (57 = number of ransomware, 80 = number of benign software). Random Forest performing the best compared to other models is seen here yielding an 85% accuracy score for binary classification between benign software and ransomware samples. Precision on ransomware software was highest on SVC (0.96) showing little false positive rates in benign software identification. However, the same cannot be said for benign software precision labeling where it yielded the highest false positive rate compared to other models (0.70) with Random Forest performing the best (0.93). Random Forest also performs best against other models on recall for ransomware labeling showing the highest sensitivity. For recall on benign software, SVC performs the best (PR=0.9583) with Random Forest coming (PR= 0.7647) in second. For F1-scores on ransomware data and benign Software, Random Forest performs the best. SVC and Random Forest are the best performing models in classification refer to Table 4.2. However, the variance in scores should be noted. Therefore, it's safe to assume other models should be tested for further classification such as a graph neural network model. This should be done to yield much more conclusive results; however, Random Forest performs the best overall.

Model	Accuracy	Precision on R	Precision on B	Recall on R	Recall on B	F1-Score on R	F1-Score on B
Random Forest	0.846715328	0.764705882	0.927536232	0.912280702	0.8	0.832	0.859060403
Decision Tree	0.664233577	0.575342466	0.765625	0.736842105	0.6125	0.646153846	0.680555556
SVC	0.744525547	0.958333333	0.699115044	0.403508772	0.9875	0.567901235	0.81865285
Gradient Boosting	0.635036496	0.575342466	0.741935484	0.719298246	0.575	0.621212121	0.647887324

Table 6: R = ransomware as true positive label, B = benign software with true positive label

4.2.2 Feature Selection Testing. In Table 7, we measured network activity, which was the worst performing classification regarding filtered features yielding scores of at most 59% accuracy. Due to the variance in numerical data, we cannot conclude that network activity alone can properly detect differences in ransomware and benign software.

Scores varied throughout testing but scores were relatively close in range in Table 8 for API and dll calls. SVC had the best performance in model accuracy and other metrics, yielding a 90% F1-score which is quite impressive. SVC (88% accuracy) performs the best when dealing with API and DLL related features. Precision on ransomware and Recall on benign software are also showing

Model	Accuracy	Precision on R	Precision on B	Recall on R	Recall on B	F1-Score on R	F1-Score on B
Random Forest	0.583941606	0	0.583941606	0.98245614	1.0	0	0.737327189
Decision Tree	0	0	0	0	0	0	0
SVC	0.583941606	0	0.583941606	0	1.0	0	0.737327189
Gradient Boosting	0.569343066	0	0.577777778	0	0.975	0	0.725581395

Table 7: R = ransomware as true positive label, B = benign software with true positive label

a 100% which translates to no false negative identification when true positives is seen for ransomware classification and 100% recall for ransomware data comparison. All classifiers scores are relatively variable but no concerningly low score to be labeled as a bad performance. Random Forest also performed very well showing the best Precision on benign software(PR = 0.9286) and Recall on ransomware classification (0.912).

Model	Accuracy	Precision on R	Precision on B	Recall on R	Recall on B	F1-Score on R	F1-Score on B
Random Forest	0.854014599	0.776119403	0.928571429	0.912280702	0.8125	0.838709677	0.866666667
Decision Tree	0.802919708	0.767857143	0.827160494	0.754385965	0.8375	0.761061947	0.832298137
SVC	0.875912409	1.0	0.824742268	0.701754386	1.0	0.824742268	0.903954802
Gradient Boosting	0.810218978	0.776119403	0.821428571	0.736842105	0.8625	0.763636364	0.841463415

Table 8: R = ransomware as true positive label, B = benign software with true positive label

In Table 9, we primarily tested process-related features, Random Forest performs the best overall in terms of model accuracy. Decision Tree with an 83% accuracy, Gradient Boosting with an 84% accuracy, and Random Forest are marginally close in performance, which shows no dominant classifier throughout despite Random Forest performing the best yielding about an 85% accuracy score. It is important to note that Gradient Boosting and Random Forest are relatively similar in nature as both classifiers work with decision tree processes; however, gradient boosting decision trees are processed sequentially while random forests are processed independently.

The worst performing classifier shown was SVC, displaying the most varied data seen in Table 9. Important measurements to note Precision on benign software outperformed Precision on ransomware software showing a higher false positive rate when identifying benign software compared to ransomware software. The opposite can be said for recall as identification for ransomware software outperformed recall on benign software with three out of four classifiers generating above a 90%. F1-score performance is better on benign software performance considering all scores yielded above an 80% for each classifier.

Model	Accuracy	Precision on R	Precision on B	Recall on R	Recall on B	F1-Score on R	F1-Score on B
Random Forest	0.846715328	0.743243243	0.968253968	0.964912281	0.7625	0.839694656	0.853146853
Decision Tree	0.832116788	0.736111111	0.938461538	0.929824561	0.7625	0.821705426	0.84137931
SVC	0.737226277	1.0	0.689655172	0.368421053	1.0	0.538461538	0.816326531
Gradient Boosting	0.839416058	0.743243243	0.967741935	0.964912281	0.75	0.833333333	0.845070423

Table 9: R = ransomware as true positive label, B = benign software with true positive label

Registry and event activity yielded the best performance in classification amongst all classifiers tested in Table 10. Random Forest and Gradient Boosting performed best in model accuracy with both yielding 91%. In fact, Gradient Boosting and Random Forest yielded the same score for Precision, Recall, and F1-scores for both ransomware and benign software activity. Registry and Events features yielded the highest model accuracy overall compared to other selected features. Registry and event testing also yielded the best scores for precision testing in ransomware showing at least a 93% in precision along with showing a low false positive identification rate. Precision on benign software (0.9523) showed relatively high scores with SVC as the exception. Recall on ransomware performed very well showing at least a 70% performance. Recall on benign software performed substantially better with SVC as the exception. F1-scores on both ransomware and benign software performed the best amongst other selected feature sets. Scores on benign software had three out of four models showing at least a 0.9 or greater score with SVC as the exception. Overall, registry and event activity showed the best performance amongst all metrics with all four classifiers tested.

Model	Accuracy	Precision on R	Precision on B	Recall on R	Recall on B	F1-Score on R	F1-Score on B
Random Forest	0.905109489	0.94	0.885057471	0.824561404	0.9625	0.878504673	0.922155689
Decision Tree	0.883211679	0.936170213	0.855555556	0.771929825	0.9625	0.846153846	0.905882353
SVC	0.722627737	0.952380952	0.681034483	0.877192982	0.350877193	0.512820513	0.806122449
Gradient Boosting	0.905109489	0.94	0.885057471	0.824561404	0.9625	0.878504673	0.922155689

Table 10: R = ransomware as true positive label, B = benign software with true positive label

5 CONCLUSION

By utilizing a dynamic analysis-based approach and sampled datasets, we have identified a model that efficiently discerns ransomware from benign software and malware using binary classification. In terms of drawbacks of the study, the dataset for our ransomware versus malware comparison was much smaller in size due to the lack of obtainable large sets of ransomware samples. Our benign and ransomware samples for training were primarily obtained from an external repository. However, our testing set consisted of manually collected samples.

When comparing ransomware versus benign software, we were able to show that registry and event related features were the most optimal in performance in identifying ransomware versus benign software. We were able to select features on registry and event related inputs by looking from a graphical spectral analysis to map out the activity of such features amongst benign software, malware, and ransomware. Similarly, we were able to discern certain features in network activity from these graphs. As a result, when comparing ransomware and malware, we were able to show that network activity scaled the highest in model performance overall.

For ransomware versus malware testing and training, most classifiers were able to yield a distinguishable difference in malware versus ransomware just from accuracy scores alone with Decision Tree and Gradient Boosting classifiers performing the best overall. Additionally, most classifiers were able to show differences in benign software and ransomware where Random Forest and SVC performed the most optimally. We can identify several features that can be used in tandem to identify ransomware from malware and benign programs.

REFERENCES

[1] Juan A. Herrera-Silva and Myriam Hernández-Álvarez. 2023. Dynamic feature dataset for ransomware detection using machine learning algorithms. *Sensors* 23, 3 (2023), 1053. https://doi.org/10.3390/s23031053

[2] Shanxi Li, Qingguo Zhou, Rui Zhou, and Qingquan Lv. 2022. Intelligent Malware Detection Based on Graph Convolutional Networks. *The Journal of Supercomputing* 78, 3 (2022), 4182–4198. https://doi.org/10.1007/s11227-021-04020-y

[3] NTFS123. 2018. NTFS123/Malwaredatabase. https://github.com/NTFS123/MalwareDatabase

[4] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao Schardl, and Charles Leiserson. 2020. EVOLVEGCN: Evolving graph convolutional networks for dynamic graphs. *Proceedings of the AAAI Conference on Artificial Intelligence* 34, 04 (2020), 5363–5370. https://doi.org/10.1609/aaai.v34i04.5984

[5] Umara Urooj, Bander Ali Al-rimy, Anazida Zainal, Fuad A. Ghaleb, and Murad A. Rassam. 2021. Ransomware detection using the dynamic analysis and Machine Learning: A Survey and Research Directions. *Applied Sciences* 12, 1 (2021), 172. https://doi.org/10.3390/app12010172

[6] Ytisf. 2014. YTISF/thezoo: A repository of Live Malwares for your own joy and pleasure. thezoo is a project created to make the possibility of malware analysis open and available to the public. https://github.com/ytisf/theZoo

[7] Umme Zahoora, Asifullah Khan, Muttukrishnan Rajarajan, Saddam Hussain Khan, Muhammad Asam, and Tauseef Jamal. 2022. Ransomware detection using deep learning based unsupervised feature extraction and a cost sensitive pareto ensemble classifier. *Scientific Reports* 12, 1 (2022). https://doi.org/10.1038/s41598-022-19443-7

[8] Zikai Zhang, Yidong Li, Wei Wang, Haifeng Song, and Hairong Dong. 2022. Malware detection with dynamic evolving graph convolutional networks. *International Journal of Intelligent Systems* 37, 10 (2022), 7261–7280. https://doi.org/10.1002/int.22880