Proceedings of the International Conference on Industrial Engineering and Operations Management

Publisher: IEOM Society International, USA DOI: 10.46254/EU08.20250413
Published: July 2, 2025

Testing a Multi-Model Artificial Intelligence System to Detect Ransomware Activity

Alexander Veach and Munther Abualkibash
School of Information Security and Applied Computing
Eastern Michigan University
Ypsilanti, MI 48198, USA
aveach1@emich.edu, mabualki@emich.edu

Abstract

Ransomware is a major threat in the modern cybersecurity field, causing billions of United States Dollars in damages each year. To combat this issue, there has been research and the application of machine learning and artificial intelligence techniques as a defensive measure. In current literature, there is a focus on using a singular model to detect ransomware in general. However, this method has an issue of accuracy decay due to the shifting methods of attack used by ransomware. While some works focus on using deep learning methods and optimization algorithms, this article investigates another method of mitigating performance decay that is less process intensive. The goal of this article is to propose a multi-model system that uses specialized models trained on specific families of ransomware to reduce decay over time. This works by reducing the amount of variance represented in each model. To test the comparative performance of this model against the common general models, a dataset is created for training using hybrid analysis of ransomware samples. A testing dataset was also created using the same methods but contained ransomware families not represented in the training dataset to test the comparative accuracy decay. The proposed system outperformed the general models at a threshold of one vote, had less decay compared to the general models, and performed comparatively to other similar works. This system, however, needs further fine-tuning to increase performance as it did not outperform all similar works, alongside rigorous testing to ensure the reduction in decay will be consistent over time.

Keywords

Ransomware; Machine learning; Artificial intelligence; Ransomware detection; Multi-model

1. Introduction

In the modern computing environment, malicious software has become a common occurrence that all must protect themselves from. Users must be vigilant against the many different viruses and attacks on the modern internet, with major organizations being especially vulnerable due to the massive size of their digital infrastructure. These organizations rely on networked services to execute their activities at a large scale, and service disruptions due to malicious software attacks can halt critical business functions, causing potentially millions, or billions, of United States Dollars in damages. Among these attacks, fewer cause more damage and disruption than that of a ransomware attack.

Ransomware is a type of malicious software that attempts to get capital from an organization. One of the most popular types of ransomware is cryptographic, which encrypts critical system data to disrupt operations and extract a ransom. To remove the encryption from affected systems, a decryption key is required to decode the data. When compromised by ransomware, a company has two choices: to pay the ransom in hopes that the attackers will provide a decryption key or initiate their system recovery strategy.If the company attempts to pay the ransom, they rely on the ransomware group exchanging a working decryption key. This does not always work, as the ransomware group can provide a fake key, alongside other unfavorable results. The second method of accepting the losses and initiating a system recovery is the suggested method of dealing with ransomware demands. The system recovery strategy lacks the risks associated with paying the ransom. However, these system recovery plans can take months or longer to restore standard functionality to the organization. To prevent this, many organizations have multiple layers of defense to proactively defend themselves.

However, major organizations operate at a massive scale with multiple different potential avenues of attack. This leads to an issue of response time to these threats. To that end, many researchers and organizations have investigated the potential of using machine learning (ML) and artificial intelligence (AI) to automate the detection of these attacks and to react accordingly. Current research is commonly focused on using a singular general model, often using specialized ensemble or deep learning methods to achieve strong performance. This practice has been criticized as these general models will experience a decay in effectiveness as the methods used in ransomware change (Oz et al. 2022), (McIntosh et al. 2022).

To combat this decay in effectiveness, this paper proposes a modular decision-making system that uses multiple specialized models to combat the decay over time. These models are trained on specific ransomware families to create specialized models, with each model implemented in the system being used to detect those families or similar families. By creating specialized models, the amount of decay over time could be limited, as the variance over time for a singular ransomware family will be less than that of multiple families. Each model will classify the data being tested and return a vote of "safe" or "unsafe," with the result determined by these votes. The system will also allow for the customization of the number of votes needed for the result of "safe" or "unsafe". By setting the number of votes to a higher or lower value, those using this system can fine-tune the sensitivity of the decision-making model.

To test the proposed system, different variants of the general model were created using different classification methods and their results were compared to the proposed system. The results of these experiments are then analyzed and compared to see the initial difference in performance.

2. Literature Review

Ransomware has been a long-standing threat that has grown more severe as the years have passed. Multiple different researchers have analyzed this topic specifically focused on utilizing new ML and AI techniques for detection and prevention and have offered their own insights into this topic.

Reviews of using AI to detect and prevent ransomware attacks were often focused on aggregating the results of others and offering their insight gained from parsing so many works of research. Oz et al. (2022) document the history and evolution of ransomware over time and categorize the types of research into the topic. They also analyze multiple studies and aggregate their results and report their true positive rate and false positive rate, if provided. Davies et al. (2021) did something similar and specifically noted that many of the methods for detection were based primarily on Shannon Entropy, which they explain has issues as a differentiator as specific types of data can trigger a false positive result, such as jpeg and mp3. Aldauiji et al. (2022) broke down the steps of the average ransomware result and broke down the methods of multiple different experiments. McIntosh et al. (2022) took their analysis and used it to outline metrics for evaluating research.

All these reviews specifically found that the research into the topic often presented their results with high evaluation metrics; however, the efficacy of these solutions was often called into question due to the high numbers and an obfuscation of the data used during training, among other factors. Another commonality is that all the reviews agree that ransomware changes rapidly, and the results gained from a dataset gathered over a set period will be affected by common ransomware methods evolving. Each review highlights different methods of combating these problems, with one of the most common being that of deep learning. While deep learning does have benefits when it comes to complex classification, this comes at the cost of higher computational demands.

Experiments using AI often focus on a specific method of detection and using a public dataset or their own original dataset. These methods then report metrics that are supposed to represent the performance of their model. Noorbehbahani and Saberi (2020) used a method to modify the features of the pre-generated dataset using new families and a feature selection algorithm. They combined the resultant dataset with a random forest classifier and found it performed around sixty-five percent accuracy. Their work did not account for decay in performance over time for their general model, and the dataset used was from 2017 and did not test against ransomware not represented in the training set.

Hirano and Kobayashi (2022) used live forensics from a hypervisor to get memory and storage from their target systems and found Random Forest to be the best classifier for their dataset. Their study, however, was limited by only using data gathered from a lightweight hypervisor; alongside this, they also had limitations on the amount of information gathered that could reduce the external validity of their model.

Sharma and Sangal (2023) focused on detecting Android-based ransomware and found that their best-performing classifier could detect Ransomware, Adware, and PremiumSMS but struggled with SMS and Scareware detection in their dataset. Their work did not test the performance of their models against ransomware not represented in the original dataset, limiting the external validity of their model. Almomani et al. (2021) did something similar but focused on permissions and API calls for the features used for detection and found that Random Forest was also their best performer. Similarly, their work did not account for the decay over time and focused on the metrics provided during their training, limiting the external validity of their model. These experiments created general models using various methods which have been reported to suffer from a decay in accuracy over time. Likewise, the metrics used are not consistent across all works which can make it difficult to compare the performance of two experiments, there is also a lack of testing against ransomware families not included in the dataset which can lead to misleading results.

Then there were frameworks that focused on outlining different methodologies to approach the problem. Poudyal and Dasgupta (2020) outlined AI-powered Ransomware Detection (AIRaD), which used hybrid analysis with Cuckoo Sandbox to extract natural language processing features, association rules, and behavioral chains to detect and log potential ransomware and classify it. An issue with AIRaD is that Cuckoo Sandbox is no longer being officially supported, which causes issues when attempting to recreate the tool outlined.

While technically a review, Smith et al. (2022) analyzed proposed frameworks for ransomware detection with machine learning and analyzed their overlap and differences. Their work does not test the frameworks and models they report, alongside using models from 2016 and 2018 without testing them to see if the performance has decayed over time. Vehabovic et al. (2022) break down the potential features that could be used in ransomware analysis focusing on the common methods used by ransomware and examine the different methods of response and detection. They found that many frameworks and their provided datasets were often incompatible, which caused issues when attempting to compare performance. Their work was heavily focused on the theoretical and did not test the frameworks and their performance and did not test if there was a decay in performance against new ransomware.

Hussain et al. (2023) published the Ransomware Execution PROvenance Dataset (REPROD) which contained 933 samples of ransomware binaries from Malware Bazaar using Cuckoo Sandbox. They specifically gathered Process Monitor logs of the ransomware attacks and outlined the importance of sharing datasets. A gap in their dataset is that there was a ten-minute limit to the sandbox, which could have cut off operations that the ransomware was attempting to make. Furthermore, they reported that the process monitor would sometimes be encrypted terminating the gathering of data for their dataset.

Zhu et al. (2023) proposes a multi-model framework for detecting ransomware in Android systems that tests multiple different implementations of a voting system to detect harmful network traffic. By using a feature fusion approach mixed with unsupervised neural network learning methods they produced a novel ensemble learning algorithm. Their work performed well compared to other similar research and reported validity after testing another dataset against their model. However, their model was based on a dataset from 2015, and their validation dataset was from 2017 which could have limited the decay from changes in malware methods.

Sogut and Erdem (2023) likewise used a multi-model system to detect potential Designated Denial of Service (DDoS) attacks on Supervisory Control and Data Acquisition systems. They used a hybrid model that combined deep learning methods and traditional machine learning models. The work concludes that all methods tested had an accuracy of ninety percent or higher and would be able to detect DDoS reliably and work as a defensive measure. Their work is focused on a single aspect of defense and does not analyze other potential threats with their model.

Peppes et al. (2021) focuses on testing a hard voting and soft voting multi-model system to detect attacks on Internet of Things systems used in agriculture. In their experiments they found that the soft voting system outperformed the hard method when sample distribution was normal and under sampled datasets, while hard performed better in over sampled datasets. Their work like the prior was focused on network traffic attacks such as DDoS instead of malware. Alongside this, the external validity of the model was not tested to ensure external validity.

Based on the information gathered from these articles and other surveyed research, a major issue that needs to be addressed is reducing the decrease in accuracy over time. Deep learning is a commonly suggested method to combat this issue, with the assumption that a large and complex model can better account for the variance introduced by new methods like Zhu et al. (2023). This research takes a different approach, focusing on using simple classifiers to create multiple specialized models for use in a decision-making system. By using specialized models to detect specific families of ransomware, the loss of accuracy over time may be mitigated as the specified groups should have less variance compared to the commonly proposed general model. Alongside this, by using fewer intensive methods these systems can be less resource intensive. Alongside this, a focus on Windows-based ransomware differentiates this work from the other android focused works.

The contribution of this work is to outline a modular multi-model system for Windows-based ransomware detection that is resistant to the decay in performance over time. This system will use specialized models created for specific ransomware families to reduce the variance introduced by averaging multiple different families, alongside allowing the training of new specialized models to counter new families of ransomware. Alongside this, the system proposed will be customizable to ensure that implementation in networked systems can be easily replicated and managed.

3. Proposed System

The proposed multi-model system is designed to store multiple different trained machine learning models and use them in a waterfall method to aggregate the results from each individual model. Based on the first-generation consensus-based system outlined by Li et al. (2005), each model stored for use classifies the data passed and aggregates the resultant votes to decide on the correct course of action. The result is determined by the number of "unsafe" classifications returned by the models, with the final result being considered unsafe if the number of "unsafe" votes is equal to or greater than the decision-making threshold. The system is designed to be easily modified to better fit the task at hand by allowing users to replace the models used and by allowing the voting threshold to be changed. The visual breakdown of this system is shown in Figure 1.

This modularity allows for trained models to be changed to detect the common ransomware families being used. For example, if an organization is worried about Blacksuit ransomware they could add a model specifically trained to detect Blacksuit ransomware. This can also be used to replace models that are suffering from a severe decrease in performance. The proposed system must be able to detect ransomware quickly due to the speed and severity of the threat, which encourages the use of high-speed and high-accuracy classifiers. A major drawback of this method is that the decision-making process of the system will slow down based on the number of models used as well as the classifier used to create the model. Due to the waterfall method used for detection, the complexity of calculation will increase the time taken for prediction which must be managed to ensure efficient use.

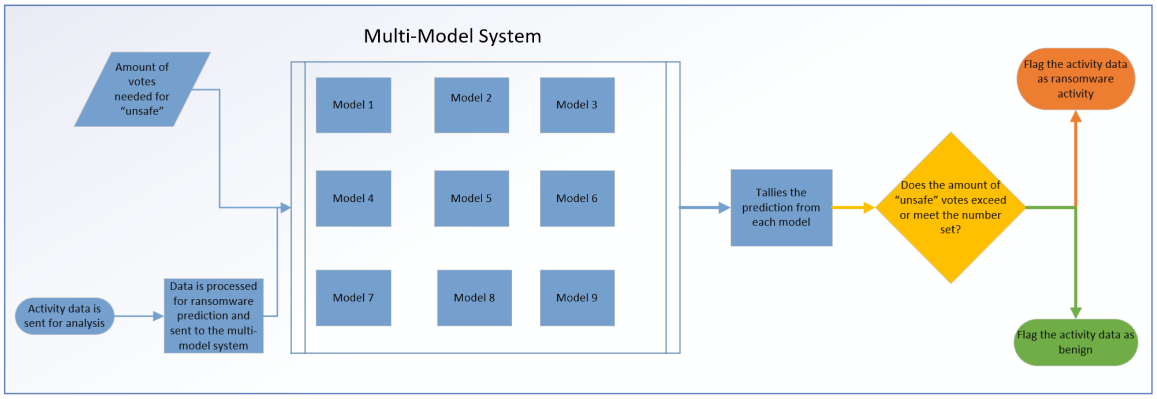
Multi-Model System
Amount of votes needed for "unsafe"
Model 1
Model 2
Model 3
Model 4
Model 5
Model 6
Model 7
Model 8
Model 9
Activity data is sent for analysis
Data is processed for ransomware prediction and sent to the multi-model system
Tallies the prediction from each model
Does the amount of "unsafe" votes exceed or meet the number set?
Flag the activity data as ransomware activity
Flag the activity data as benign

Figure 1. Proposed Multi-Model Ransomware Detection System

This system offers benefits by having multiple specialized datasets that are less prone to accuracy decay as they target a single style of ransomware, reducing the amount of variance introduced over time. Alongside this, the customizable nature of the system allows for optimization of detection and strictness. This system also has the benefit of simplicity, which can be easily replicated using virtualization techniques. Using these techniques this system could be a scalable solution if properly set up. However, for this system to work in a real-world environment the multi-model system must be implemented with high-level permissions in the client network and have the logs either pre-processed for decision-making or processed as they are sent. In exchange, the system can perform as a reactive layer of defense against ransomware, but first it must be tested to see if there is merit to the system.

4. Methodology and Experiment Design

To evaluate the effectiveness of the proposed multi-model system against the commonly used generic system, a dataset was created to showcase the comparative performance of the two approaches. The experiment was designed with a focus on quick response time and the performance of the trained models. The reason for this being the nature of ransomware and how fast it can spread once the malicious software gains higher-level permissions. A series of general models were created to represent the various methods showcased in other research, while a multi-model system with specialized models was created to represent the proposed method. During training each model's performance was generated by taking the average of ten different variants of the dataset achieved by randomizing an 80/20 split. These models had their training performance compared to see their performance relative to each other, and then had their performance tested against a testing dataset that contained ransomware represented in the training dataset and families excluded from the dataset.

4.1 Creating the Dataset

For the purposes of this experiment, a localized environment was necessary to gather the samples to create a dataset used for ransomware detection. A double-nested virtualized environment was utilized to ensure the safe processing of the ransomware samples. The base hardware for all testing was a desktop with a Ryzen 9 5900x 12-core processor, sixty-four gigabytes of Random Access Memory(RAM), an Nvidia GeForce 3080 graphics card, and a terabyte of solid-state storage. The first layer used VirtualBox to create an Ubuntu 22.04 machine with thirty-two gigabytes of RAM, twelve processors, and six hundred gigabytes of storage. Inside this Ubuntu machine CAPEv2[17], a fork of Cuckoo Sandbox, was used to automate deployment and analysis of the ransomware samples. The software used by CAPEv2 for virtualization was Kernel-Based Virtual Machine (KVM), which hosted a Windows 10 machine configured to their specifications listed in CAPEv2's documentation (n.d.) was used as the test environment.

The ransomware samples were gathered from online repositories such as VX-Underground (n.d.) and Malware Bazaar(n.d.). 583 samples were gathered across thirty-nine different ransomware families, with sixty control samples of standard Windows applications. These families were selected by three major aspects: the number of samples available, the recency of the ransomware family, and the impact of the ransomware. These aspects were considered for gathering to ensure that the ransomware had been active in the last five years, that there were enough samples for a trained model to predict if the data given is representative of the data, and that the ransomware gathered had a recognized effect in the cybersecurity field. This ensured that families such as REvil, BianLian, Cuba, and Phobos ransomware are included in the dataset. The standard Windows samples were gathered from common Windows software such as Microsoft Edge and GitHub.

Major limitations of the dataset come from the testing process, focusing on ransomware families with a sizable number of samples. This means that novel methods that have fewer samples were passed up for more common methods in the multi-model system and were limited to the general model. This means that the dataset created has limitations on detecting novel attacks, alongside only containing cryptographic ransomware. The dataset also makes assumptions about the data available for decision-making, as CAPEv2 uses dynamic and static analysis which may be difficult when operating at a larger scale.

These samples were processed through CAPEv2, which returned an analysis of the program's execution in JSON (JavaScript Object Notation) alongside a log file detailing the actions taken. The JSON files were analyzed to extract features such as operation signatures, the registry keys accessed, the amount of YARA: Another Recursive Acronym (YARA) detections noted by CAPEv2, and the Application Programming Interface (API) calls used. Alongside this the default log file was also mined for further features such as users involved and the operations reported.

To generate the dataset, key features were extracted from the output files using Python scripting. Using these methods, features were gathered and stored in comma-separated value (CSV) format. Several features at this point were string variables and needed to be processed for machine learning. For that purpose, WEKA 3.9.6 (Frank et al., 2016) and common-csv (Reutermann, n.d.), an unofficial WEKA package, were used to import the large CSVs for data processing. From there each dataset was modified by the default StringToWordVector method provided by WEKA, which uses a Bag of Words method to convert strings into numerical features. Using this method a training dataset was created alongside a testing dataset containing a mixture of families both represented and absent in the training dataset. The number of features per dataset varied based on the methods used, ranging from 1012 features to 1200 features.

4.2 Creating the Dataset

To create the models used, the Scikit Learn (Pedregosa et al., 2011) Python package was used to generate the models. These tests were executed on the base hardware described in Section 6.1; however, Scikit Learn did not natively support GPU use at the time of this study, so all results were generated with only the processor. The methods used for classification were Random Forest (RF), Support Vector Machine (SVM), Gaussian Naïve Bayes (GNB), Classification and Regression Tree (CART), Multilayer Perceptron (MP), and Histogram Gradient Boosted Tree (HGBT). These classifiers used their default parameters for the tests done to establish the baseline for effectiveness. These classifiers are commonly represented in the related work and used to recreate the general models described in other literature. Multilayer Perceptron is an example of a deep learning classifier and is available in the Scikit Learn module; however, its performance is not indicative of the performance of other classifiers and further research should be done comparing the performance of more deep learning methods against the multi-model system.

Specific family models were also created based on ransomware families that contained fifteen or more samples available, with control data equal to half of the samples provided (i.e., for fifteen ransomware samples there are seven control samples). These models were then implemented in the multi-model system described in Section 3 using CART as the classifier of choice. The reason for this is CART's rapid decision-making which would assist the multi-model system in quickly generating a prediction. The reason behind this is that cryptographic ransomware can rapidly encrypt a network and rapid decision-making is needed to reduce the damage caused by ransomware.

4.3 The Metrics Used

The most important test of the multi-model system is to ensure that performance is like other commonly used classifiers in decision-making speed and accuracy. To determine this each model is assessed to gather the key metrics related to machine learning models: accuracy(eq1), precision(eq2), recall(eq3), true negative rate (eq4), F-score(eq5) and Precision-recall curve(eq6). These scores are determined by taking the confusion matrix of true positive, false positive, true negative, and false negative then applying the values to the formulas below. True positive represents an unsafe sample marked as unsafe, false positive represents a safe sample marked as unsafe, true negative represents a safe sample marked as safe, and a false negative represents an unsafe sample marked as safe.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (1)$$

$$Recall = \frac{TP}{TP+FN} \quad (2)$$

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

$$True\ Negative\ Rate = \frac{TN}{TN+FN} \quad (4)$$

$$F-Score = \frac{\left(2*\left(\frac{TP}{TP+FP}\right)*\left(\frac{TP}{TP+FN}\right)\right)}{\left(\frac{TP}{(TP+FP)}\right)+\left(\frac{TP}{TP+FN}\right)} \quad (5)$$

$$Matthews\ Correlation\ Coefficeint\ (MCC) = \frac{TN*TP-FN*FP}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \quad (6)$$

Accuracy represents the number of correct decisions made by the model overall and is a common metric to showcase performance; however, it does not give the entire picture of a model. Precision detects the rate of correct positive classifications against the total amount of positive samples in the dataset. In this dataset precision represents how often an unsafe value is truly unsafe. Recall measures the amount of misclassification of positive results; in the scope of this work it would be marking safe results as unsafe. The true negative rate is focused on the correct identification of negative results. F-Score is the relative performance of Recall and Precision used to analyze the overall performance when it comes to classification for a model. F-Score is often argued to be a better measure of a model's performance as it focuses on the reliability of the classification rather than the result. Finally, the MCC is a value that shows the model's performance when it comes to determining actual values and predicted values, and the correlation of the variables used for prediction. For all these metrics the closer the value is to one the better the performance.

6. Results and Discussion

After executing the experiment, the following results were gathered. During training, standard classifiers all performed well returning F-Scores of .96 (seen in table 1) and higher, which means that the model can accurately predict the contents of the training dataset. The best-performing classifier in the training dataset was RF, which had a value of 1 in all metrics. This value indicates that using the default training method has led to overfitting of the dataset. The other standard classifiers showcase high metrics as well, which again indicates that the overfitting could be caused by the current dataset lacking in variance. To combat this issue in future research, cross-validation or a larger dataset can help reduce overfitting and ensure that the models can predict external data more reliably. Alongside this, other methods that modify feature weights based on statistical methods such as genetic algorithms can be used to mitigate overfitting.

Table 1. Average Standard Classifier Training Results

Classifier Used	Accuracy	Precision	Recall	True Negative Rate	F-Score	MCC
RF	1.0	1.0	1.0	1.0	1.0	1.0
SVM	0.998	1.0	0.986	1.0	0.992	0.992
GNB	0.998	1.0	0.976	1.0	0.988	0.987
CART	0.992	0.989	0.941	0.998	0.964	0.96
MP	0.998	0.981	1.0	0.998	0.99	0.989
HGBT	0.994	0.982	0.962	0.998	0.971	0.969

Comparatively, the multi-model methods performed worse than the standard classifiers in training in each category (seen in table 2 and 3). The only family models system performed best at the twenty-five percent threshold (four unsafe votes required) for determining a value as "unsafe" with an F-Score of .975, bringing it below most of the standard classifiers during the training phase. This lower value could indicate that unlike its standard counterparts the system is less affected by overfitting. An exception to this is the ninety-five-percentage (13/14 votes needed for unsafe classification) threshold which had an F-score of 0.816 which indicates an issue in reliability. Alongside this the fifty and seventy-five percent thresholds (7 and 11, or 8 and 12 votes required respectively) performed the same. This means that performance was consistent across those thresholds and problems did not occur until almost full confidence was needed for an unsafe result.

Adding the general model to the multi-model system caused a change in performance depending on the number of votes needed for a decision of "unsafe". For a strict one-vote for unsafe systems the recall decreased, which in turn decreased the F-score and MCC once the general model was added. This means that during training, the general model incorrectly identified safe samples as unsafe. However, when the voting threshold was set to four votes the general model increased performance slightly, resulting in a higher F-score and MCC. This means that requiring more of a consensus from the trained models prevented the loss of accuracy and increased it by correctly determining classification of data.

Table 2. Average Multi-Model (Family Models Only) Training Results

Votes needed for "unsafe"	Accuracy	Precision	Recall	True Negative Rate	F-Score	MCC
1	0.989	1.0	0.883	1.0	0.938	0.934
4	0.992	0.966	0.95	0.996	0.958	0.954
7	0.982	0.845	1.0	0.980	0.916	0.91
11	0.982	0.845	1.0	0.980	0.916	0.91
13	0.956	0.69	1.0	0.952	0.816	0.81

Table 3. Average Multi-Model (Family Models + General Model) Training Results

Votes needed for "unsafe"	Accuracy	Precision	Recall	True Negative Rate	F-Score	MCC
1	0.985	1.0	0.85	1.0	0.919	0.915
4	0.994	0.967	0.967	0.996	0.967	0.963
8	0.982	0.845	1.0	0.980	0.916	0.91
12	0.982	0.845	1.0	0.980	0.916	0.91
14	0.956	0.69	1.0	0.952	0.816	0.81

While the training results on average performed well, the external validity of the model must be proven. The models were tested against data containing a mixture of families represented in training and families absent from training. Alongside this, recent ransomware such as Blacksuit was included to see how well these models perform against new methods. The results of these tests are shown in tables 4 and 5.

The decision-making models had a decrease in performance when used to predict the testing dataset. GNB had the biggest decrease in performance as it commonly marked safe samples as unsafe. SVM had a decrease across both precision and recall, which means that it incorrectly identified both safe and unsafe data. RF had a major decrease in precision, which indicates that the model misidentified unsafe data as safe. The high true negative rate for these classifiers does show that safe data was commonly identified as safe.

The multi-model system had a similar decrease in performance, most evident when the number of votes required for an "unsafe" value were higher than one. However, compared to the other classifiers the multi-model system with a one-vote threshold retained its performance as the best. This shows that the multi-model system has potential to combat the decay over time; however, further research is necessary for anything conclusive. Setting the voting threshold for four performed the second best but had classification issues overall resulting in a lower MCC and F-Score compared to the training performance. The later thresholds struggled with precision and detecting true negatives which resulted in the lowest scores outside of GNB and SVM.

Table 4. Standard Classifier Testing Results

Classifier Used	Accuracy	Precision	Recall	True Negative Rate	F-Score	MCC
RF	0.862	0.60	1.0	0.826	0.75	0.704
SVM	0.862	0.667	0.667	0.913	0.667	0.580
GNB	0.828	1.0	.167	1.0	0.286	0.370
CART	0.862	0.6	1.0	0.826	0.75	0.704
MP	0.862	0.6	1.0	0.826	0.75	0.704
HGBT	0.862	0.6	1.0	0.826	0.75	0.704

Table 5. Multi-Model (Both Versions) Testing Results

Votes needed for "unsafe" (with general / without general)	Accuracy	Precision	Recall	True Negative Rate	F-Score	MCC
1/1	0.931	0.833	0.883	0.957	0.833	0.79
4/4	0.862	0.833	0.833	0.870	0.714	0.637
7/8	0.828	0.545	1.0	0.783	0.706	0.653
11/12	0.828	0.545	1.0	0.783	0.706	0.653
13/14	0.793	0.5	1.0	0.739	0.667	0.608

Beyond accuracy, another major component of detecting ransomware is in the response speed. For training purposes HGBT took the longest to train at 20.491 seconds, with the next highest being the multi-model system at 2.511 seconds needed, which is almost ten times faster. Beyond this, the other classifiers trained rapidly with training times below one second. For time to predict RF took the longest at a tenth of a second until a decision was made, and the next highest being the multi-model systems at 0.061 and 0.059 seconds respectively. All other classifiers predicted below that it took 0.45 to 0.54 seconds to predict the result.

Table 6**.** Train and Prediction Times per Classifier

Classifier	Time to Train (Training)	Time to Predict (Testing)
RF	0.710 seconds	0.104 seconds
SVM	0.701 seconds	0.046 seconds
GNB	0.076 seconds	0.045 seconds
CART	0.066 seconds	0.045 seconds
MP	0.680 seconds	0.052 seconds
HGBT	20.491 seconds	0.054 seconds
MM (Fam)	2.509 seconds	0.061 seconds
MM (Fam+Gen)	2.511 seconds	0.059 seconds

This means overall, the best performing method in testing was the multi-model system with a one-vote threshold. It managed to be affected the least when classifying the testing dataset and managed to predict the result in under a second. However, the current method is still imperfect as there is still a decrease in accuracy in testing compared to the training performance.

Compared to the other reported works, this multi-model system performed similarly in training to the models proposed in other research scoring in the higher side of the ninety percentiles. As mentioned in the related work section, the reported values were often based on internal validity rather than external validity which could modify the results in real-world scenarios. Even then, comparatively the multi-model systems performed worse than Zhu et al. (2023) when it came to MCC. This means that there are currently issues in the simple implementation that need to be addressed to ensure the system performs better against data not represented in training. To do this a fusion method with neural networks could be used to raise that value. Another method to increase the performance could be in changing the grouping used for the specialized models, which could optimize their detection capabilities. Finally, further tuning of the dataset and features used could reduce the variance in detection and strengthen the capabilities of the proposed multi-model system.

7. Future work

While these experiments outlined above show the potential in a multi-model system, there are many avenues of exploration needed for a complete understanding. Currently, the method uses CAPEv2 under the assumption that a business parsing their computers data will have a security endpoint capable of processing the information into a dataset and feeding it to the model. Creating a system that requires less preprocessing before decision-making would make it easier for organizations to adopt this system. Another avenue of potential research for this system focuses on the customization aspect. Further research should focus on testing the change in performance as the number of models used in the voting process increases.

Likewise, the type of models used in the system can be changed from CART to another classifier or a mixture of classifiers. Deep learning methods are more intensive but offer potential in multi-model systems as seen in Zhu et al. (2023). Further works can test the performance of deep learning methods in the outlined hard voting system to compare the performance against the ensemble methods showcased in this work. Another critical point that was tested in a limited way is the reduction in detection due to the natural change in attack methods for ransomware. While Blacksuit was new there is no more than a few months difference between it and some of the other samples in the dataset. Further research should be done when more time has passed to see how resilient the proposed system is against the issue of performance decay.

This work is also focused heavily on the theoretical; thus, future work should focus on more accurately representing an organization's infrastructure and how this method performs under different levels of strain. Creating a simulated network and with this multi-model system and seeing how fast it can detect and stop ransomware activity on the network would show the efficacy in real-world environments. Alongside this, different styles of implementation can be experimented with such as hosting the trained models via servers accessible by APIs. By doing this, it can make it easier for users to select the models they wish to use in their implementation via an online repository. Similarly, pre-designed versions of this system could be implemented into various virtualization methods to create scalable ransomware detection infrastructure.

Another area of potential research is testing new methods of grouping for ransomware detecting models. The current method of grouping by family theoretically creates specialized methods of detection against said family; however, it can lead to misclassification of other ransomware if the methods of attack are different enough. By grouping ransomware by other features, such as behaviors could assist in decreasing the performance decay of general models. Likewise, other methods of grouping based on other qualitative features may also assist with this problem.

Other methods of preventing decay can also be used in combination with this system, such as using cross-validation and other training techniques to decrease overfitting of the model. Optimization algorithms are one such method, as they can be used to optimize feature weights and would help reduce misclassification. The dataset used is also currently limited by its size, adding further data can assist in reducing the amount of error when it comes to predictions. Changing the specialized models to be grouped around other features, as mentioned in the prior paragraph, would also assist in reducing the overfitting issue currently afflicting this system.

8. Conclusion

Ransomware is a complicated threat that has, and will, continue to dominate discourse in the field of cybersecurity. The failure to detect and respond to ransomware can cause untold damage in the millions or more to a major organization. Alongside this there has been increased pressure on major organizations as hacking groups target major organizations for their cyberattacks. The multi-model system proposed in this paper shows potential in resisting the decay caused by the evolution in ransomware compared to ensemble general models. However, the proposed system does not manage to completely overcome the decay in the experiment, and currently some deep learning methods report better performance. Further research should focus on adding and testing other methods to increase the efficacy of the proposed method. Other avenues of research include using deep learning and other advanced machine learning techniques in conjunction with the proposed system and testing different methods of implementation in real-world environments. Application of this system can be a useful layer of defense against the ever-increasing ransomware threat and reduce the amount of time spent retraining general ransomware detection models. The application of the methods outlined in this paper alongside more traditional preventative methods can assist organizations in defending against these costly attacks.

References

Oz, H., Aris, A., Levi, A., and Uluagac, A. S., “A Survey on Ransomware: Evolution, Taxonomy, and Defense Solutions,” ACM Comput. Surv. 54, 11s, Article 238, 2022, 37 pages. doi:10.1145/3514229

McIntosh, T., Kayes, A. S. M., Chen P. P-Y., Ng, A., and Watters, P. “Ransomware Mitigation in the Modern Era: A Comprehensive Review, Research Challenges, and Future Directions”. ACM Comput. Surv. 54, 9, Article 197, Dec 2022, 36 pages. doi:10.1145/3479393

S. R. Davies, R. Macfarlane and W. J. Buchanan, "Review of Current Ransomware Detection Techniques," 2021 International Conference on Engineering and Emerging Technologies (ICEET), Istanbul, Turkey, 2021, pp. 1-6, doi:10.1109/ICEET53442.2021.9659643

F. Aldauiji, O. Batarfi and M. Bayousef, "Utilizing Cyber Threat Hunting Techniques to Find Ransomware Attacks: A Survey of the State of the Art," in IEEE Access, vol. 10, pp. 61695-61706, 2022, doi:10.1109/ACCESS.2022.3181278

Noorbehbahani, F., and Saberi, M. "Ransomware Detection with Semi-Supervised Learning," *2020 10th International Conference on Computer and Knowledge Engineering (ICCKE)*, Mashhad, Iran, 2020, pp. 024-029, doi:10.1109/ICCKE50421.2020.9303689

Hirano, M. and Kobayashi, R., "Machine Learning-based Ransomware Detection Using Low-level Memory Access Patterns Obtained From Live-forensic Hypervisor," *2022 IEEE International Conference on Cyber Security and Resilience (CSR)*, Rhodes, Greece, 2022, pp. 323-330, doi:10.1109/CSR54599.2022.9850340

Sharma, N. and Sangal, A. L., "Machine Learning Approaches for Analysing Static features in Android Malware Detection," *2023 Third International Conference on Secure Cyber Computing and Communication (ICSCCC)*, Jalandhar, India, 2023, pp. 93-96, doi:10.1109/ICSCCC58608.2023.10176445

Almomani, I., Al-Khayer, A., Ahmed, M., "An Efficient Machine Learning-based Approach for Android v.11 Ransomware Detection," *2021 1st International Conference on Artificial Intelligence and Data Analytics (CAIDA)*, Riyadh, Saudi Arabia, 2021, pp. 240-244, doi:10.1109/CAIDA51941.2021.9425059

Poudyal, S. and Dasgupta, D., "AI-powered Ransomware Detection Framework," *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, Canberra, ACT, Australia, 2020, pp. 1154-1161, doi:10.1109/SSCI47803.2020.9308387

Smith, D., Khorsandroo, S., and Roy, K., "Machine Learning Algorithms and Frameworks in Ransomware Detection," in *IEEE Access*, vol. 10, pp. 117597-117610, 2022, doi: 10.1109/ACCESS.2022.3218779

Vehabovic, A., Ghani, N., Bou-Harb, E., Crichingno, J., and Yayimili, A., "Ransomware Detection and Classification Strategies," *2022 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*, Sofia, Bulgaria, 2022, pp. 316-324, doi:10.1109/BlackSeaCom54372.2022.9858296

Hussain, S., Musa, M., Neeshat, T., Batool, R., Ahmed, O., Zaffar, F., Gehani, A., Poggio, A., and Yadav, M., "Towards Repoducible Ransomware Analysis*," Proceedings of the 16th Cyber Security Experimentation and Test Workshop (CSET '23)*, Association for Computing Machinery, New York, NY, USA, pp. 1-9, 2023, doi:10.1145/3607505.3607510

Zhu, H., Li, Y., Wang, L., Sheng, V.S., "A multi-model ensemble learning framework for imbalanced android malware detection", *Expert Systems with Applications,* vol. 234, Dec 2023, doi: 10.1016/j.eswa.2023.120952

Sogut, E. and Erdem, O., "A Multi-Model Proposal for Classification and Detection of DDoS Attacks on SCADA Systems", Appl. Sci. 2023, vol. 13, May 2023, doi:10.3390/app13105993

Peppes, N., Daskalakis, E., Alexakis, T., Adamopoulou, E., Demestichas, K., "Performance of Machine Learning-Based Multi-Model Voting Ensemble Methods for Network Threat Detection in Agriculture 4.0", Sensors 2021, vol. 21(22), Nov. 2021, doi:10.3390/s21227457

Li, X.R., Zhao, Z., and Li, X.B., "General model-set design methods for multiple-model approach," *IEEE Transactions on Automatic Control*, vol. 50, no. 9, pp. 1260-1276, Sept. 2005, doi:10.1109/TAC.2005.854581

O'Reilly, K., and Brukhovetskyy, A., "CAPE: Malware Configuration And Payload Extraction (Version 2)", available at: https://github.com/kevoreilly/CAPEv2

CAPE Sandbox., "CAPEv2 Documentation: Preparing the Guest", Available at: https://capev2.readthedocs.io/en/latest/installation/guest/index.html

VX-Underground, vx-underground, available at: https://vx-underground.org/

Abuse.ch, Malware Bazar, available at: https://bazaar.abuse.ch/

Frank, E., Hall, M.A., and Witten, I.H. "The WEKA Workbench," in *Data Mining: Practical Machine Learning Tools and Techniques*, Morgan Kaufmann, Fourth edition, 2016.

Reutermann, P. "common-csv-weka-package", available at: https://github.com/fracpete/common-csv-weka-package.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E. "Scikit-learn: Machine Learning in Python", *Journal of Machine Learning Research, vol. 12, pp. 2825-2830,* 2011.

Farfoura, M.E., Alkhatib, A., Alsekait, D.M., Alshinwan, M., El-Rahman, S.A., Rosiyadi, D., and AbdElminaam, D.S. "A low complexity ML-based methods for Malware Classification", *TSP, Computers, Materials & Continua, vol. 80, pp. 4833-4857, 2024.*

Biographies

Alexander Veach is a graduate student from Eastern Michigan University in Ypsilanti, Michigan with a bachelor's degree in information technology and a master's in cybersecurity. He has a penchant for computers and has done work on a few papers on machine learning and artificial intelligence and now smart mobility. His current interests in research includes cybersecurity, smart mobility, artificial intelligence, and cloud computing.

Munther Abuelita's is an associate professor in the School of Information Security and Applied Computing at Eastern Michigan University. His interests and expertise include computer and network security, cloud computing, and machine learning. He received his master's degree from the University of Bridgeport, in Bridgeport, Connecticut. There, he also earned his Ph.D. in computer science and engineering.