Journal of Cyber Security

Tech Science Press

Doi:10.32604/jcs.2025.063711

ARTICLE

# Detecting Ransomware Using a Hybrid Detection Scheme

**David Conway and Paolina Centonze***

Department of Computer Science, Iona University, New Rochelle, NY 10804, USA

*Corresponding Author: Paolina Centonze. Email: pcentonze@iona.edu

Received: 22 January 2025; Accepted: 23 April 2025; Published: 14 May 2025

**ABSTRACT:** Ransomware is a variant of malicious software that aims to encrypt data or whole systems to lock out the intended users. The attackers then demand a ransom for the decryption key to allow the intended users access to their data or system again. Ransomware attacks have the potential to be used against industries like healthcare and finance, as well as against the public sector, have threatened and forced the operations of key infrastructure used by millions to cease, and extorted millions and millions of dollars from victims. Automated methods have been designed and implemented to detect ransomware within a system, some of which rely on created rules or heuristics to act as a blacklist for software. Others incorporate artificial intelligence, including machine learning models which use previous data to predict whether software is benign or malicious. We attempt to create a hybrid ransomware detection scheme that combines the non-artificial intelligence method of heuristic-based detection with machine learning detection to explore differences in accuracy and speed compared to solely machine learning detection.

**KEYWORDS:** Ransomware detection; heuristic; machine learning; hybrid detection; entropy

## 1 Introduction

Like other types of malware, ransomware exhibits certain characteristics and behaviors that distinguish it from benign software and allow it to be identified [1]. Characteristics commonly used for identifying ransomware include system logs and calls, network use and traffic, and interactions with other data within the file system of a device [2]. One main type of ransomware detection is manual detection, in which human operators perform system analysis and intervention. Having this interaction between trained individuals or teams and software with any relevant data is effective but is laborious and tedious [2]. This human limitation on the efficiency and speed of manual ransomware detection makes automated ransomware detection a viable and desirable alternative, especially with the rise of artificial intelligence.

Two automated approaches to ransomware detection that do not utilize artificial intelligence are signature-based and heuristic-based detection. Signature-based detection attempts to identify known patterns associated with ransomware within the code or activity of software in order to flag it as such [2]. Similar to signature-based detection is heuristic-based detection, which relies on created rules meant to fit the boundaries of normal behavior for software. If the software breaks these rules, it will be flagged as malicious. With heuristic-based detection, it is essential for whoever is creating the heuristic to be knowledgeable about what features reliably distinguish malware from benign software so they can create an effective heuristic that minimizes the number of false positives but still keeps the system safe.

A common subset of artificial intelligence used for automated ransomware detection is known as machine learning, which seeks to explore and exploit the ability of machines to "learn" from data given

to them without being specifically programmed [1]. The machine learning algorithms used to detect ransomware and other types of malware are known as supervised learning classifiers. "Supervised learning" in this context means that the data fed into the algorithms is labeled, which is useful when the machine learning model will be used for a specific question. "Classifier" refers to the fact that the output of these models will be a prediction, or a classification, of test data given to it after it has been trained on other data. Common machine learning algorithms used for ransomware detection include decision trees [3], random forests [4], k-nearest neighbors [5], and support vector machines [5]. All of these have shown much success and high accuracy in their classifications of software as benign or malicious.

Although machine learning has been shown to work exceedingly well in detecting and flagging software as benign or malicious, like ransomware, there are current limitations and room for improvement. As ransomware and other malware can spread rapidly within a singular device or throughout a whole organization's system, speed in detection and intervention is of the utmost importance. To mitigate the spreading of ransomware, a detection scheme must be able to identify malicious software quickly, as close to real-time as possible [2]. A possible way to accomplish this ideal speedy detection is to combine detection approaches, like making a hybrid detection scheme that pairs other automated approaches with a machine learning approach. An example hybrid approach would be pairing heuristic-based detection with a machine learning model. This could potentially help quicken the detection process because the heuristic-based detection could flag the software in question based on its known data or current behavior while any relevant features are compiled for the machine learning to use as test data and generate its classification.

"Entropy" is defined as a lack of order or predictability. In software engineering, file entropy measures the randomness of a file's data [6]. This measure can indicate whether a file has obfuscated code or encrypted code, which could help identify a file as malicious. File entropy is measured from 0 (no randomness) to 8 (totally random) [6]. Due to the link between encrypted code and malicious software, some schemes for malware detection have utilized file entropy measures of the files being examined [7]. In our experiment, file entropy was the primary feature used for our heuristic.

It is important to remember that the file entropy value of compressed and encrypted files may be similar, meaning it may be hard to distinguish between benign and malicious files if the files are compressed [7]. Despite this, promise has been shown in detection schemes utilizing entropy as a key marker for malware [8]. It is hoped that detection schemes based on file entropy will be faster than manual detection or dynamic testing of a file while not sacrificing accuracy or increasing false positives.

## 2 Materials and Methodology

All algorithms used for our research were implemented using Anaconda's Jupyter Notebook, which allows users to visualize their data and code simultaneously and is commonly used for data science and artificial intelligence research. The machine learning algorithms that we used are k-nearest neighbors and decision trees. These are all supervised learning classification machine learning algorithms included in the open-source scikit-learn or sklearn library for the Python programming language.

The data used for the experiments contains 138,047 samples of software, each with over 50 associated features, such as the size of the software's code and the number of imports and exports performed by the code [9]. The dataset is composed of 70% malicious samples and 30% samples. This dataset has been used by other researchers and their findings showed success in detecting malicious samples using machine learning based on selected features [10]. To more effectively use and simplify this dataset, we used the selected features listed in Table 1 for training and testing, including the "legitimate" feature, which shows whether or not the software sample is benign or malicious [10].

**Table 1:** Selected features

| Features* |
|---|
| SizeOfOptionalHeader |
| MajorLinkerVersion |
| AddressOfEntryPoint |
| SectionAlignment |
| MinorOperatingSystemVersion |
| SizeOfHeaders |
| SizeOfStackReserve |
| LoaderFlags |
| SectionsMinEntropy |
| SectionsMaxEntropy |
| SectionMaxRawsize |
| SectionsMinVirtualSize |
| ResourcesMinEntropy |

*These are the features selected for use by the machine learning model [10].

### *2.1 Methodology*

We created two sets of detection schemes for this research: one used solely the three machine learning algorithms separately, and the other set will be the hybrid set, where a heuristic-based detection algorithm will be paired with each machine learning algorithm. The speed of classification, accuracy, and number of false positives will be recorded for each scheme. These metrics will show how the hybrid schemes compare with the non-hybrid machine learning algorithms in their efficiency and accuracy. With the hybrid set, whether or not the heuristic correctly classified a sample from the test data will also be tracked. All machine learning models used in the experiment were trained with 70% of the dataset used and tested with the remaining 30%. The total amount of data samples in the test set was 41,415 and it was composed of 70% malicious samples and 30% benign samples, mirroring the composition of the entire dataset.

As mentioned in Section 1, file entropy has been used as a marker for malware in cybersecurity research, so we decided to have the heuristic for the hybrid schemes flag software with an average entropy greater than a certain threshold value. The values we chose to test using this heuristic were 5, 6, and 7. With this heuristic, the software currently undergoing classification will be tested to see if it fits or goes beyond this heuristic, and it will automatically be flagged as malicious ransomware if it does. If the software does not break the heuristic, it will then be passed to the machine learning algorithm that the heuristic-based detection is paired with, which will generate its classification as it normally would.

Fig. 1 shows the main loop of our hybrid detection scheme, which has a linear time complexity, the same as the pure machine learning detection that we have tested it against. Both schemes must cycle through every test data sample and predict for the current sample, similar to the scale of efficiency of most detection schemes. For the schemes that use a k-nearest neighbor machine learning model, the time complexity for N test samples, N_train training samples, and D features for every data sample is O(N × N_train × D). For our experiments, N was 41,415 test samples, N_train was 96,932 training samples, and D was the 14 features of each data sample used. The decision tree machine learning models used in our experiments had maximum depths of 33, meaning the time complexity for predicting for N test samples is O(N × depth), where N is the 41,415 test samples and depth is 33. The random forest machine learning models used had 100 trees and an average depth of 29, making the time complexity for predicting N test samples with T trees O(N × T × depth).

As was previously stated, in our experiments, N was 41,415 test samples, T was 100 trees, with an average depth of tree of 29. Introducing the heuristic with our hybrid schemes focuses on decreasing the volume of test samples that must be predicted for by the machine learning model used, as the complexity of each scheme is dominated by the machine learning prediction. Therefore, with an effective heuristic that catches samples before they are sent to the associated machine learning model for prediction, a portion of the test set would not have to be predicted for by the dominant, costly machine learning prediction computations.

```
Function: HybridScheme
Input: X_test - test dataset
       ML - trained machine learning model
       entVal - value that average entropy must be above in order to be flagged.
Output: hybridPred - collection of predictions
        heuristicCaught - integer counter of samples caught by heuristic
        heuristicCorrect - integer counter of samples caught and correctly flagged by heuristic

1 for sample in X_test:
2       if (((sample['SectionsMaxEntropy'] + sample['SectionsMinEntropy']) / 2) > entVal):
3               heuristicCaught += 1
4               hybridPred.append(0)
5               if sample['legitimate'] == 0:
6                       heuristicCorrect += 1
7       else:
8               hybridPred.append(ML.predict(sample))
```

**Figure 1:** Pseudo-code of hybrid detection scheme

### *2.2 Limitations and Expected Findings*

The limitations of this research and experiment include the difficulty in finding more public and realistic data to use for training and testing the machine learning models and for the experiment in general. With limited data, the heuristic created may be too specific to the data used and may not be widely applicable to software samples outside of the used dataset. It is hoped that the heuristic used is general enough to be applicable outside of the used dataset and that the findings will reveal features of software that can be quickly collected and tested to make hybrid detection schemes viable.

## 3 Results

Table 2 shows that our 'Average Entropy Above 5' heuristic caught the most data samples from the test set but also had the most false positives, with each hybrid using the heuristic making over 1400 false positives. Still, the 'Average Entropy Above 5' heuristic had the highest percentage of correct flaggings as well, with all three such hybrids correctly flagging roughly 54% of the total malicious samples in their test sets. The 'Average Entropy Above 6' heuristic caught fewer samples overall compared to the 'Average Entropy Above 5' heuristic but made much fewer false positives, with no such hybrid making over 160 false positives, as shown in Table 3. The 'Average Entropy Above 6' heuristic hybrids each correctly flagged roughly 41% of the malicious samples in the test set. Finally, Table 4 shows that the 'Average Entropy Above 7' heuristic caught a negligible number of samples overall, with all three hybrids using the heuristic correctly flagging less than 1% of the total malicious samples within their test sets. These results indicate that our heuristics for average entropies above 5 and 6 were effective at catching malicious samples, but that the 'Average Entropy Above 6' heuristic especially was able to catch and correctly flag samples, all while minimizing the number of false positives.

**Table 2:** Hybrid detection results—average entropy above 5

| Machine learning model used | Total flags by heuristic | Correct flags by heuristic | False flags by heuristic | Total accuracy of hybrid | Malicious predict samples | Benign predict samples | Time elapsed (s) |
|---|---|---|---|---|---|---|---|
| K-nearest neighbor | 17,180 | 15,756 | 1424 | 0.94 | 28,994 | 12,421 | 91.02 |
| Decision trees | 16,930 | 15,494 | 1436 | 0.95 | 28,923 | 12,492 | 58.28 |
| Random forest | 17,078 | 15,655 | 1423 | 0.96 | 28,942 | 12,473 | 401.06 |

**Table 3:** Hybrid detection results—average entropy above 6

| Machine learning model used | Total flags by heuristic | Correct flags by heuristic | False flags by heuristic | Total accuracy of hybrid | Malicious predict samples | Benign predict samples | Time elapsed (s) |
|---|---|---|---|---|---|---|---|
| K-nearest neighbor | 12,246 | 12,097 | 149 | 0.97 | 28,994 | 12,421 | 113.88 |
| Decision trees | 12,083 | 11,926 | 157 | 0.98 | 28,923 | 12,492 | 69.90 |
| Random forest | 12,130 | 11,998 | 132 | 0.99 | 28,942 | 12,473 | 473.52 |

**Table 4:** Hybrid detection results—average entropy above 7

| Machine learning model used | Total flags by heuristic | Correct flags by heuristic | False flags by heuristic | Total accuracy of hybrid | Malicious predict samples | Benign predict samples | Time elapsed (s) |
|---|---|---|---|---|---|---|---|
| K-nearest neighbor | 62 | 26 | 36 | 0.97 | 28,994 | 12,421 | 152.12 |
| Decision trees | 52 | 24 | 29 | 0.98 | 28,923 | 12,492 | 83.52 |
| Random forest | 48 | 26 | 22 | 0.99 | 28,942 | 12,473 | 655.89 |

### *Compared with Pure Machine Learning*

All our hybrid schemes were able to yield comparable total accuracies to the results of the pure corresponding k-nearest neighbor model, decision trees model, and random forest model, the metrics of which are shown in Table 5. The hybrid schemes using a k-nearest neighbor model had accuracies of 0.94 by the 'Average Entropy Above 5' hybrid and 0.97 by both the 'Average Entropy Above 6' and 'Average Entropy Above 7' hybrids. The k-nearest neighbor model alone was also able to yield an accuracy of 0.97. The hybrid schemes using a decision trees model had an accuracy of 0.95 by the 'Average Entropy Above 5' hybrid, and

0.98 by the 'Average Entropy Above 6' hybrid and the 'Average Entropy Above 7' hybrid, compared with the accuracy of 0.99 yielded by the non-hybrid decision trees model. The hybrid random forest models had a total accuracy of 0.96 yielded by the 'Average Entropy Above 5' hybrid and 0.99 by the 'Average Entropy Above 6' and 'Average Entropy Above 7' hybrids. The accuracy of the pure random forest model was 0.99.

**Table 5:** Pure machine learning detection results

| Machine learning model used | Accuracy | Malicious predict samples | Benign predict samples | Time elapsed (s) |
|---|---|---|---|---|
| K-nearest neighbor | 0.97 | 28,994 | 12,421 | 125.77 |
| Decision trees | 0.99 | 28,923 | 12,492 | 78.69 |
| Random forest | 0.99 | 28,942 | 12,473 | 683.77 |

The elapsed times for both the hybrid and pure machine learning schemes were also comparable. The hybrids using a k-nearest neighbor model had run times of roughly 91,114, and 152 s in order of increasing average entropies used as the threshold for the heuristic, compared to the pure k-nearest neighbor model taking almost 126 s. This represents a 27.78% and a 9.52% increase in speed compared to the pure k-nearest neighbor model by the k-nearest neighbor hybrid schemes using the 'Average Entropy Above 5' and 'Average Entropy Above 6' heuristics, respectively. The heuristic-decision trees model hybrids took roughly 58, 70, and 83 s for the heuristics using thresholds of average entropy of 5, 6, and 7, respectively, while the pure decision trees model took about 77 s. This makes the decision trees hybrid schemes have speed increases of 24.67% and 9.09% for the 'Average Entropy Above 5' and 'Average Entropy Above 6' heuristics, respectively. The biggest difference between a pure model and the related hybrids in elapsed time for predictions was between the random forest models. The pure random forest model had a run time of about 684 s, while the hybrids using a random forest model ran for roughly 401 s for the 'Average Entropy Above 5' heuristic, a 41.37% increase in speed, 474 s for the 'Average Entropy Above 6' heuristic, a 30.70% increase in speed, and 656 s for the 'Average Entropy Above 7' heuristic, a 4.09% increase in speed.

What is notable in these run time results is that the fastest times and largest increases in speed for each machine learning model came from either the 'Average Entropy Above 5' or the 'Average Entropy Above 6' hybrids. These two heuristics catch and flag more data samples than the 'Average Entropy Above 7' heuristic; and, as the pure machine learning models by necessity have no such heuristic, this indicates that there is a correlation between using an effective heuristic and a quicker run time. This is in keeping with our prior reasoning that decreasing the number of test samples being sent to machine learning models would improve time complexity by cutting out the dominant prediction process for test samples that break the heuristic.

## 4 Discussion and Conclusion

The results of our experiments support the idea that file entropy can be an effective marker to identify malicious software, including ransomware. Additionally, our experiments have shown that a hybrid scheme for ransomware detection that utilizes both a heuristic and a machine learning model can provide results comparable to a lone machine learning model and offer a faster linear time complexity solution. Further work could include expanding these experiments to larger datasets with more features, which could help identify other features that can serve as the basis for effective heuristics. Building a more realistic simulation could be another direction to take further research, as this could more accurately measure the difference in speed of detection between a hybrid detection scheme and a pure machine learning model. A more realistic

simulation would have to simulate the collection of data from a software sample to be used for prediction, as the information helpful for accurate detection is not always immediately available as it is in a dataset.

## 5 Conclusion

In this paper, we designed and tested a hybrid detection framework utilizing a heuristic and a machine learning model. Using three different machine learning models and comparing our framework's metrics on a ransomware dataset against a pure machine learning model, we found that our framework maintained comparable accuracy with slightly faster prediction time. Two challenges faced with our experiment are the need for a realistic and extensive dataset and the need for a more realistic simulation in which to test our framework. Future work should focus on simulating data collection for the sample being predicted for, replicating how not all the information necessary for accurate detection is known immediately by a system.

**Acknowledgement:** Thank you to Dr. Paolina Centonze of the Computer Science Department at Iona University for suggesting the research topic and providing guidance throughout the research project. Thank you to Iona University for allowing this project to be done as part of my honors thesis for my final years as an undergraduate.

**Funding Statement:** The authors received no specific funding for this study.

**Author Contributions:** The authors confirm contribution to the paper as follows: Conceptualization, David Conway and Paolina Centonze; methodology, David Conway; software, David Conway; validation, David Conway; writing—original draft preparation, David Conway; writing—review and editing, David Conway and Paolina Centonze. All authors reviewed the results and approved the final version of the manuscript.

**Availability of Data and Materials:** The data that support the findings of this study are openly available in Hybrid Heuristic-ML Ransomware Detection at: https://github.com/dconway0/Hybrid-Heuristic-ML-Ransomware-Detection (accessed on 1 January 2025).

**Ethics Approval:** Not applicable.

**Conflicts of Interest:** The authors declare no conflicts of interest to report regarding the present study.

## References

1. Gorment NZ, Selamat A, Cheng LK, Krejcar O. Machine learning algorithm for malware detection: taxonomy, current challenges, and future directions. IEEE Access. 2023;11(12):141045–89. doi:10.1109/ACCESS.2023.3256979.
2. Alraizza A, Algarni A. Ransomware detection using machine learning: a survey. Big Data Cogn Comput. 2023;7(3):143. doi:10.3390/bdcc7030143.
3. Ullah F, Javaid Q, Salam A, Ahmad M, Sarwar N, Shah D, et al. Modified decision tree technique for ransomware detection at runtime through API calls. Sci Program. 2020;2020:1–10. doi:10.1155/2020/8845833.
4. Khammas BM. Ransomware detection using random forest technique. ICT Express. 2020;6(4):325–31. doi:10.1016/j.icte.2020.11.001.
5. Akhtar MS, Feng T. Malware analysis and detection using machine learning algorithms. Symmetry. 2022;14(11):2304. doi:10.3390/sym14112304.
6. IBM QRADAR security intelligence platform 7.4 [Internet]. [cited 2025 Jan 1]. Available from: https://www.ibm.com/docs/en/qsip/7.4?topic=content-analyzing-files-embedded-malicious-activity.
7. Zainodin ME, Zakaria Z, Hassan R, Abdullah Z. Entropy based method for malicious file detection. JOIV Int J Inform Vis. 2022;6(4):856. doi:10.30630/joiv.6.4.1265.
8. Lee K, Lee J, Lee SY, Yim K. Effective ransomware detection using entropy estimation of files for cloud services. Sensors. 2023;23(6):3023. doi:10.3390/s23063023.

9. Muditmathur. GitHub—muditmathur2020/ransomware detection: ransomware detection using machine learning models and ensemble technique [Internet]. [cited 2025 Jan 1]. Available from: https://github.com/muditmathur2020/RansomwareDetection.
10. Masum M, Faruk MJH, Shahriar H, Qian K, Lo D, Adnan MI. Ransomware classification and detection with machine learning algorithms. In: 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC); 2022 Jan 26–29; Las Vegas, NV, USA.