ZAMBIA INFORMATION COMMUNICATION TECHNOLOGY (ICT) JOURNAL
Volume 7 (Issue 1) (2023) Pages 52-56

A Supervised Machine Learning Ransomware Host-Based Detection Framework

Yotam Mkandawire[a] and Aaron Zimba[b]

a. Department of Computer Science & IT Mulungushi University, Kabwe, Zambia, email: thby2022@gmail.com

b. Department of Computer Science, ZCAS University, Lusaka Zambia, email: aaron.zimba@zcasu.edu.zm

***Abstract*— today, the term ransomware is frequently used in cybercrime headlines, its consequences have been on the rise leaving a trail of terrible losses in its wake. Both people and businesses have been victimized by ransomware, costing the victims millions of dollars in ransom payments. In addition, victims who were unable to pay the ransom or decrypt the data experienced data losses. This study uses dynamic malware analysis artifacts and supervised machine learning to detect ransomware at the host level. It takes on a thorough examination of the operational specifics of ransomware and suggests a supervised machine learning approach to detection using various ransomware features derived from a dynamic malware analysis. According to the findings, a Logistic Regression algorithm model with a 97.7% accuracy score offers a 99% success rate in ransomware detection. This demonstrates how well machine learning and dynamic malware analysis work together to detect ransomware activity at the host level. Systems security administrators can mitigate security risks by using this method.**

Keywords*— *Ransomware, CryptoLocker, Crypto API, IDS, Machine Learning

I. Introduction

Generally, rransomware is categorized into two types: locker-ransomware and crypto ransomware [19]. Locker ransomware essentially includes corrupting or disrupting basic computer functionality while protecting the data integrity and safety of the victim; it typically locks computing devices or user interfaces and requires a ransom payment to unlock them. Crypto-ransomware on the other hand, encrypts the files of victims on a computer or network and demands a ransom to decode them. It is worth noting that crypto-ransomware assaults do not encode the entire hard disk, but rather look for imperative file extensions that have the greatest impact on victims [2].

From its first introduction, ransomware has grown to become one of the biggest threats to both individuals and enterprises globally. Over the years, the size of the ransom demanded by attackers has grown exponentially, a ransomware annual report by [16] shows that the typical cost to rectify the consequences of the most recent ransomware attacks was US$1.85 million (factoring in people and down time, network and device cost, opportunity cost and paid ransom etc.), this is more than double the US$761,106 cost described in the year 2020 report.

Not only has the frequency of ransomware attacks increased but there has been exponential growth with regards to ransomware attack tactics. Modern ransomware attack techniques have proven to be resilient due to their encryption and recovery-prevention techniques. The newer variants of ransomware use hybrid cryptosystems. Using RSA and AES, the malware creates symmetric and sub-symmetric host keys, data is then encrypted using AES keys, and the sub-RSA key is encrypted using the entrenched key before being used to encrypt the AES keys. [4]. Furthermore, newer ransomware strains have evolved to the extent of including recovery-prevention tactics such as the erasing of volume shadow copies or overwriting original target files after encryption [20]. Most mitigation frameworks have become obsolete as a result of these novel strategies, which have sparked research attention.

When faced with fresh ransomware versions that use newer and more sophisticated attack strategies, mitigation techniques may fall short due to the continually changing attack techniques. Newer ransomware variants employ stealth and evasion strategies which make detection at network level extremely difficult, host level detection has a competitive advantage of being able to view the entire set of actions that a malware program performs, allowing harmful code to be identified before it is run at all [2].

This study investigates the detection of ransomware assaults in a generic host environment by utilizing dynamic features of malware. It deals with a thorough review of the operating elements of ransomware and suggests a machine learning technique in collaboration with several ransomware characteristics generated from the dynamic malware analysis for detection. Microsoft Windows operating system is used for the host environment due to its global wide use [4].

The study is structured as follows: Section II covers relevant publications, and Section III introduces the approach and suggested detection framework. The findings and analysis in Section IV, as well as the conclusion in Section V while limitations and recommendations come in Section VI.

II. RELATED WORKS

Host level detection has received a lot of interest in the security scene, despite the fact that network level detection has garnered more study. While some research efforts have focused on implementing dynamic malware analysis, most have centered on using static malware analysis. [9]. A study by Kharaz et al. [10] gave rise to "UNVEIL". The study examined 1,359 malware samples from 15 different ransomware families. First, it creates a sandbox-type analysis environment with bait files for a ransomware to targets. API hooking is then used to observe system activity in this context. The ransomware is introduced into the environment and checked for three things: a large rise in the randomness (entropy) between the read and write data buffers, the creation of new files with a high entropy signature, and the number of I/O requests related to writing or deleting the bait files.

Locky and CryptoWall ransomwares employ Microsoft CryptoAPI or OpenSSL, according to Takanari [17]. The authors decided to monitor API calls related to encryption as a means of attack detection as a result of their discovery. With this method, file encryption attempts by Encryptor are noticed as they attempt to begin, and prevention is performed by the operating system stopping the API execution as soon as detection takes place. The detection application injects a DLL file into each software process in order to track API calls made by the target software.

Kim [11] used document classification techniques to assess the performance of a machine learning strategy in malware detection. They used batch script files to extract Windows system calls as a feature and used 8-grams, 9-grams, and 10-grams to extract feature information. Weights were assigned using TF-IDF and vectors were created using Euclidean normalization. Finally, they ran tests using SVM[7] and SGD[8] which came out with a 96% accuracy.

In 2018, Takeuchi, Sakai and Fukumoto [18] proposed a ransomware detection scheme for Microsoft computers based on support vector machines. Using Cuckoo Sandbox, they dynamically retrieved characteristics from ransomware API invocation sequences. When tested, the framework generated 2-gram count vectors that had a detection accuracy of 97%.

Table I summarizes the contrast between our prosed model and existing frameworks.

TABLE I. COMPARISON WITH OTHER WORKS

Framework	API Monitoring	Registry Key Monitoring	Directory Monitoring	File rwx Monitoring	Web Application
[10]	×	×	×	×	×
[18]	✓	×	×	×	×
[8]	×	✓	×	×	×
[17]	✓	×	×	×	×
[11]	×	×	×	×	×
Proposed model	✓	✓	✓	✓	✓

As can be seen in Table I above, our detection framework has several advantages in comparison to some existing frameworks. Though API monitoring is the core of the framework, our approach factors in registry key monitoring, directory monitoring, file permission monitoring and a web-based user interface.

III. PROPOSED METHOD

Based on the sequence of system operations, such as API calls, registry key operations, directory operations and file ‘read-write-execute’ operations our proposed approach employs machine learning to identify ransomware files from innocuous ones. A significant portion of machine learning research is focused on supervised learning, the availability of annotated training data is what distinguishes supervised learning from unsupervised learning. The term suggests the existence of a "supervisor" who gives the learning system instructions for the labels to attach to training samples. In classification issues, these labels are often class labels. From these training data, supervised learning algorithms create models that may be used to categorize more unlabeled data. In supervised learning, a mapping between a set of input variables X and an output variable Y is learned, and this mapping is then used to forecast the results for hypothetical data.

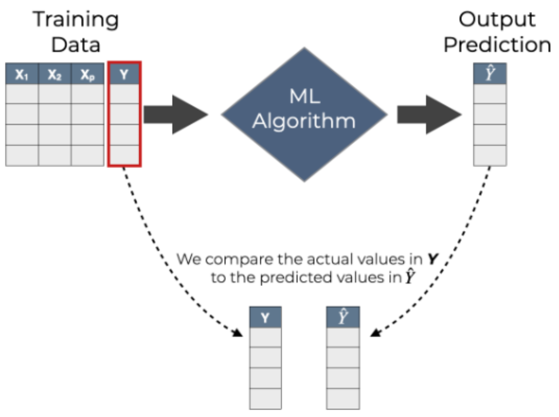

Training Data
X₁
X₂
Xₚ
Y
ML Algorithm
Output Prediction
Ŷ
We compare the actual values in Y to the predicted values in Ŷ
Y
Ŷ

Figure 1: Classical supervised learning [8]

As shown in figure 1, the training data (also referred to as the dataset), is split in two i.e., the X-set and Y-set. The target set, which might be binary (0 and 1) or many (multi classification), is represented by the Y set, whereas the X set is a collection of feature values. The key goal is to identify the function "f" that, by reducing the classification error, will more accurately generalize equation 1 and ensure that equation 2 is correct.

$$f(X) \Rightarrow Y \qquad (1)$$

$$f(x) = \hat{y} \qquad (2)$$

During the learning process, six classification algorithms were employed to train the model and the best performing algorithm was chosen as the classifier for our model [1]. The six classification algorithms used are Decision Tree Classifier, Random Forest Classifier, Gradient Boosting Classifier, Ada Boost Classifier, Gaussian Naïve Bayes, and Logistic Regression. The performance metric used is accuracy, though metrics such as precision and recall are also considered.

$$Accuracy = \frac{NumberofCorrectPredictions}{TotalNumberofPredictions} \quad (3)$$

$$Precision = \frac{TP}{TP+FP} \quad (4)$$

$$Recall = \frac{TP}{TP+FP} \quad (5)$$

Dynamic analysis is achieved via virtualization-based sandbox techniques [13]. We can identify whether a program is harmful or not by looking at the altered operating system structure that results from running software. For our proposed framework we use Cuckoo [16], a malware analysis tool that offers a thorough behavior report on a Windows executable file. The sandbox-based system's present stage of development is sufficient for reporting input executable files' behavioral activities in the form of behavioral reports. Following the extraction of the X-set characteristics from the report, the model makes a prediction.

Algorithm 1: Feature selection and machine learning model

Input: X - Standard dataset
Output: P - Feature list, Q - Classifier

```
1.  Read $X$ with no labels
2.  $f_{et}(X) \Rightarrow P$
3.  Save $P$
4.  $algorithms = \{f_{dt}(), f_{rf}(), f_{gb}(), f_{ad}(), f_{gn\_b}(), f_{lg}()\}$
5.  for i in algorithms:
6.      clf = algorithms[i]
7.      $clf.fit(P) \Rightarrow Q$
8.      Save accuracy score
9.  Algorithm with highest accuracy score is saved.
10. End
```

Algorithm 2: Supervised learning ransomware detection

Input: X - Windows executable file, P – Feature, Q – Classifier, C – Cuckoo sandbox instance, t - detection program
Output: Z – Dynamic analysis report, k – prepped sample, Y – Prediction,

```
1. Initialize $C$
2. $C(X) \Rightarrow Z$
3. $t.P(Z) \Rightarrow k$
4. $t.Q(k) \Rightarrow Y$
5. End
```

IV. Experiment Results and Discussion

A. Dataset Description

A dataset created because of the work of [15] is used as the study case throughout the experimental session. The collection includes 942 benign applications and 582 functional instances of ransomware from 11 distinct families. The dataset features include registry keys operations, API statistics, strings, file extensions, files operations, directory operations and dropped files extensions. This dataset was chosen because it allows for the collection of API information and registry key manipulations, both of which are common in ransomware activity.

B. B. Feature selection

The feature selection process employs Extremely Randomized Trees Classifier, sometimes referred to as Extra Trees Classifier. The Extra Tree classifiers integrates the results of numerous de-correlated decision trees that have been gathered in a "forest" as part of an ensemble learning strategy. To accomplish feature selection utilizing the forest structure, the normalized total decrease in the mathematical criteria (Information Gain is our preferred criterion) is computed for each feature throughout the forest's building [3]. Information gain is a term used to describe how much knowledge a characteristic imparts about a class.

$$Gain = E_{\text{parent}} - E_{\text{children}} \quad (6)$$

Where 'E' in equation 6 above represents the entropy. Entropy is a metric used in information theory to gauge how impure or uncertain a set of observations is.

$$E = -\sum_{i=1}^{N} p_i log_2 p_i \quad (7)$$

Equation 7 above is the formula to find the entropy of data where 'N' is the number of classes and 'ρι' is the likelihood of picking a class example 'ι' at random. The feature selection process also serves the purpose of rectifying the curse of dimensionality (overfitting) [5].

C. Experiment setup

80% of the data in the optimized dataset will be utilized for training, while the remaining 20% will be used for testing. Every sample has the same opportunity to take part in the trials as a training or testing sample. The same performance indicators (accuracy, precision, and recall) are considered during each cycle (training and testing).

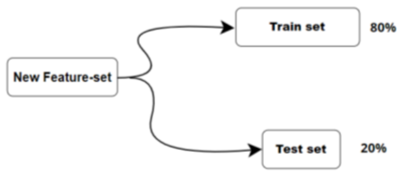


Figure 2: Dataset split to training and testing set.

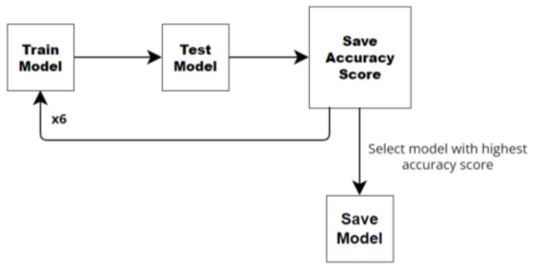
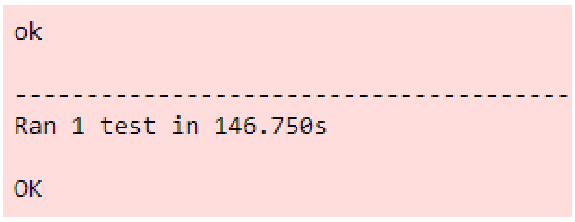


Figure 3: Find best performing algorithm.

The simulation program is written in python programming language, the machine learning model was trained and tested in Jupyter notebook, PyCharm community 2021.3.3 was used as the IDE where the Streamlit server was launched to host the web application on a computer with an Intel core i5 CPU, 2.5 GHz with 12gb RAM and running windows 10 64bit.

D. Experiment Results and Discussion

The experiment's initial phase focuses on improving the dataset and developing the machine learning model. The complexity of the execution (training/testing) is recorded in seconds and the accuracy in percentage. Metrics such as precision, recall, f1-score, ROC, and AUC are determined.



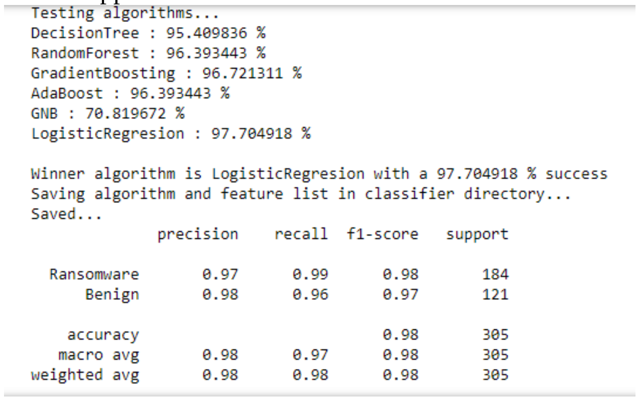
```
ok

----------------------------------------
Ran 1 test in 146.750s

OK
```

Figure 4: Execution time for modal training.

On figure 4. Both during training and during testing, the execution time is assessed. Because the testing data set's dimension is smaller, testing takes less time.

On figure 5. The accuracy scores of the classifiers we trained our model on are displayed together with the classification report of the classifier with the highest accuracy score. The classification report includes the precision score, the recall, the f1-score, and support. It also shows the macro and the weighted average of the accuracy [12].

The F1 score is a weighted harmonic mean of precision and recall, where the highest score is 1.0 and the poorest is 0.0. Precision is the capacity of a classifier not to label an instance positive that is negative, and recall is the ability of a classifier to discover all positive occurrences [6]. The number of real instances of the class in the given dataset is known as support.



```
Testing algorithms...
DecisionTree : 95.409836 %
RandomForest : 96.393443 %
GradientBoosting : 96.721311 %
AdaBoost : 96.393443 %
GNB : 70.819672 %
LogisticRegresion : 97.704918 %

Winner algorithm is LogisticRegresion with a 97.704918 % success
Saving algorithm and feature list in classifier directory...
Saved...
              precision    recall  f1-score   support

  Ransomware       0.97      0.99      0.98       184
      Benign       0.98      0.96      0.97       121

    accuracy                           0.98       305
   macro avg       0.98      0.97      0.98       305
weighted avg       0.98      0.98      0.98       305
```

Figure 5: Accuracy scores and classification report.

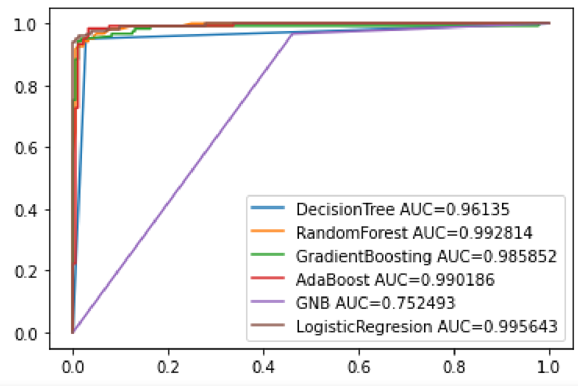
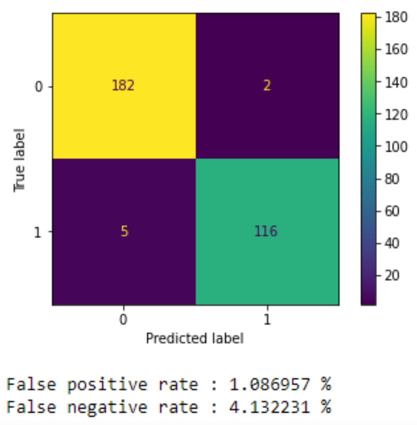


Figure 6: ROC and AUC

The figure 6 above shows ROC (Receiver Operating Characteristics) and AUC (Area under the Curve). AUC is a measure of degree of separateness, whereas ROC is a probability curve. It reveals how well the model can differentiate across classes. The AUC measures how well a model predicts 0 classes as 0 and 1 classes as 1. [14].



```
False positive rate : 1.086957 %
False negative rate : 4.132231 %
```

Figure 7: Confusion matrix

An algorithm's performance may be shown using a special table structure called a confusion matrix, often referred to as an error matrix, in the subject of machine learning, specifically the problem statistical classification [14].

V. Conclusion

This study proposes a robust system for detecting ransomware in Windows executable files using supervised machine learning. The results show that supervised learning combined with dynamic malware analysis may successfully identify ransomware at the host level. Our model's efficiency and reliability were enhanced by first reducing dimensionality and testing multiple classification algorithms to determine which algorithm yields the highest accuracy and precision.

The operations and generated artifacts exhibited in dynamic malware analysis accurately model real-life environments. The presence of such API statistics and registry key operations are high indicators of ransomware compromise. With a false positive rate of 1.1% and false negative rate of 4.1 the proposed framework is candidate for real-time implementation.

The benefit of our suggested detection methodology is that it makes use of sandboxing technology, which insulates the victim device and has broad applications in systems running Microsoft Windows operating systems like Windows Server and Windows 7/8/10.

VI. LIMITATION AND RECOMMENDATION

The scarcity of an updated open-source dataset that captures ransomware behavior on a Microsoft Windows system was the greatest challenge encountered. The scope of input files needs to be expanded such that more file extensions such as zip, iso, bat etc. can be analyzed as well. Additionally, portability (across different operating systems) and multiclass detection is an area to be explored, the current framework can only detect ransomware and benign software, future works can be extended to implement multiclass classification to detect if a file is ransomware, benign or virus.

References

[1] Alazab, Mamoun *et al.* (2010) ‘Zero-day malware detection based on supervised learning algorithms of API call signatures’.

[2] Canfora, G. *et al.* (2014) ‘Metamorphic malware detection using code metrics’, *Information Security Journal: A Global Perspective*, 23(3), pp. 57–67.

[3] Cho, J.H. and Kurup, P.U. (2011) ‘Decision tree approach for classification and dimensionality reduction of electronic nose data’, *Sensors and Actuators B: Chemical*, 160(1), pp. 542–548. doi:10.1016/j.snb.2011.08.027.

[4] Casen, M., Li, F. and Williams, D. (2021) ‘Friend or Foe: An Investigation into Recipient Identification of SMS-Based Phishing’, in Furnell, S. and Clarke, N. (eds) *Human Aspects of Information Security and Assurance*. Cham: Springer International Publishing (IFIP Advances in Information and Communication Technology), pp. 148–163. doi:10.1007/978-3-030-81111-2_13.

[5] Cuckoo Sandbox-https://cuckoosandbox.org/.

[6] Daka, E. and Fraser, G. (2014) ‘A survey on unit testing practices and problems’, in *2014 IEEE 25th International Symposium on Software Reliability Engineering*. IEEE, pp. 201–211.

[7] Ebner, J. (2021) ‘Supervised vs Unsupervised Learning, Explained’, *R-Craft*, 12 April. Available at: https://r-craft.org/r-news/supervised-vs-unsupervised-learning-explained/ (Accessed: 29 October 2022).

[8] Honda *et al.* (2016) ‘${$UNVEIL$}$: A large-scale, automated approach to detecting ransomware’, in *25th ${$USENIX$}$ Security Symposium (${$USENIX$}$ Security 16)*, pp. 757–772.

[9] Hampton, N. and Baig, Z.A. (2015) ‘Ransomware: Emergence of the cyber-extortion menace’.

[10] Kharaz, A. *et al.* (2016) ‘${$UNVEIL$}$: A large-scale, automated approach to detecting ransomware’, in *25th ${$USENIX$}$ Security Symposium (${$USENIX$}$ Security 16)*, pp. 757–772.

[11] Kim, C.W. (2018) ‘Ntmaldetect: A machine learning approach to malware detection using native api system calls’, *arXiv preprint arXiv:1802.05412* [Preprint].

[12] Luque, A. *et al.* (2019) ‘The impact of class imbalance in classification performance metrics based on the binary confusion matrix’, *Pattern Recognition*, 91, pp. 216–231.

[13] M. Neugschwandtner, C. Platzer, P. M. Comparetti, and U. Bayer, “Danubis–dynamic device driver analysis based on virtual machine introspection,” in International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. Springer, 2010, pp. 41– 60.

[14] Narkhede, S. (2021) *Understanding AUC - ROC Curve*, *Medium*. Available at: https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5 (Accessed: 4 June 2022).

[15] Sgandurra, D. *et al.* (2016) ‘Automated Dynamic Analysis of Ransomware: Benefits, Limitations and use for Detection’, *arXiv preprint arXiv:1609.03020* [Preprint].

[16] Sophos (2021) ‘State of ransomware’. Available at: https://secure2.sophos.com/en-us/medialibrary/pdfs/whitepaper/sophos-state-of-ransomware-2021-wp.pdf.

[17] Takanari Shigeta *et al.* (2016) ‘Encryption Processing of Ransomware’.

[18] Takeuchi, Y., Sakai, K. and Fukumoto, S. (2018) ‘Detecting ransomware using support vector machines’, in *Proceedings of the 47th International Conference on Parallel Processing Companion*, pp. 1–6.

[19] You, I. and Yim, K. (2010) ‘Malware obfuscation techniques: A brief survey’, in *2010 International conference on broadband, wireless computing, communication and applications*. IEEE, pp. 297–300.

[20] Zimba, A. and Chishimba, M. (2019) ‘Understanding the evolution of ransomware: paradigm shifts in attack structures’, *International Journal of computer network and information security*, 11(1), p. 26.