Detecting Ransomware Threats in Disk Storage through Behavioral Analysis using CNN2D and Flask Framework

Bhasha Pydala[1], Allampati Sireesha[2*], Peese Tejeswara Rao[3], Supriya Veluru[4], Ramireddy SaiCharan Reddy[5], V Jyothsna[6]

[1, 6] Assistant Professor, Department of CSE(DS), Mohan Babu University (Erstwhile Sree Vidyanikethan Engineering College), India
[2, 3, 4, 5] UG Scholar, Department of Computer Science and Systems Engineering, Sree Vidyankethan Engineering College, Tirupati, India.

[1]basha.chanti@gmail.com, [2*]sireeshaallampati@gmail.com, [3]teja152002@gmail.com, [4]hellosupriyaveluru@gmail.com, [5]reddycharan169@gmail.com, [6]jyothsna1684@gmail.com

Abstract. A novel strategy for combatting ransomware has emerged, aiming to circumvent the limitations of traditional antivirus software which ransomware often evades. Ransomware, by encrypting files and restricting user access to systems and data, poses a significant threat. The proposed solution involves a ransomware detection system operating within virtual machines, which collects data on processor and disk I/O activities from the host machine. Utilizing a machine learning classifier, specifically a 2D Convolutional Neural Network (CNN2D), Voting Classifier and XGBoost. Its approach seeks to minimize overhead by collectively monitoring processes rather than individually, thereby reducing the risk of data corruption induced by ransomware. The system boasts rapid detection, particularly effective against both known and unknown ransomware variants, with the random forest classifier demonstrating superior performance. Moreover, the CNN2D architecture enhances feature extraction, allowing the model to identify relevant patterns for precise classification. By selectively monitoring processor and disk I/O events, the system maintains efficiency while ensuring comprehensive coverage against ransomware activities. Across diverse user loads and ransomware types, the system consistently achieves high detection rates. Detection outcomes are conveniently presented using the Flask Framework.

Keywords: Ransomware, Encrypting, Machine learning, CNN2D, Flask.

1 Introduction

Ransomware, a type of malicious software, encrypts files or locks computers to make them inoperative, often as a way for cyber attackers to demand money. Not only criminal individuals but also governmental bodies might use ransomware assaults to disrupt the vital infrastructure of their rivals. These assaults typically involve extracting

K. R. Madhavi et al. (eds.), *Proceedings of the International Conference on Computational Innovations and Emerging Trends (ICCIET 2024)*, Advances in Computer Science Research 112,
https://doi.org/10.2991/978-94-6463-471-6_48

victims' data to force payment or trade them on the Dark Web. In 2022, about 70% of enterprises encountered ransomware attacks, a number anticipated to increase to an assault every 2 seconds by 2031, up from every 11 seconds in 2021. The financial repercussions of ransomware are significant, with expenses reaching $20 billion in 2021 and predicted to surpass $265 billion by 2031.Researchers have investigated different techniques for identifying ransomware attacks. While signature-based detection depends on antivirus generated hash values to recognize known ransomware, polymorphic and metamorphic variations can avoid detection. Consequently, behavioral and runtime detection methods are crucial additions to signature-based approaches. This mechanism suggests an original way to detect ransomware on virtual machines by concentrating on disk I/O activities and processors. By collecting data from the host machine and using a random forest classifier, the system attains prompt detection with a high likelihood, surpassing conventional methods. Moreover, the research explores the potential of utilizing 2D convolution neural networks for enhanced feature extraction.

2 Literature Survey

This study integrates hardware execution profiles with ML techniques to classify Microsoft-Windows ransomware, achieving by using a dataset of both ransomware and non-ransomware. Challenges include dependency on hardware counters and handling new ransomware variants. Despite these, our research underscores the effectiveness of leveraging hardware profiles for enhanced ransomware detection in cyber security efforts. [5]

The system uses Sequential Pattern Mining on 1,624 ransomware samples to extract Maximal Frequent Patterns (MFP), achieving high accuracy in detection and family classification. However, it faces challenges with zero-day attacks and adapting to new variants, affecting real-time detection and scalability. Nonetheless, our research highlights the effectiveness of pattern mining in enhancing ransomware detection and cyber security intelligence. [6]

RanStop utilizes hardware performance monitoring and an LSTM model for early crypto-ransomware detection within 2ms. With 97% accuracy against benign programs, it swiftly identifies ransomware, mitigating financial and data security threats. Challenges include resource overhead, updates for new ransomware variants, and compatibility issues. Nonetheless, RanStop proves effective in mitigating ransomware attacks' impact. [7]

RATAFIA introduces a two-step unsupervised ransomware detection framework using Fast Fourier Transformation and Deep Neural Network. It offers accurate, fast, and reliable detection without OS kernel modification, adaptable to modern systems. Challenges include potential false positives/negatives, requiring continuous updates for evolving ransomware techniques. Nonetheless, RATAFIA proves effective in enhancing software security by reliably detecting ransomware without OS kernel modification. [8]

3 Methodology

3.1 Proposed Work

To overcome the aforementioned issue, the paper's author proposes employing VMWARE on the host system to read Hardware Performance Counters (HPC) and IO EVENTS data. This data is then utilized in machine learning models to predict whether executing scripts are benign or ransomware. By extracting HPC and IOEVENTS features using VMWARE, system performance remains unaffected, while achieving over 90% accuracy in ransomware prediction. Various machine learning algorithms including XGBOOST, LSTM, and Voting Classifier were experimented, with Random Forest and XGBOOST consistently providing high accuracy. Additionally, deep learning models such as DNN and LSTM were employed. The extension includes experimentation with CNN2D to optimize dataset features in multiple layers, potentially improving detection accuracy. However, the paper did not explore advanced feature optimization algorithms.

3.2 System Architecture

The system architecture leverages VMware on the host system to extract Hardware Performance Counters (HPC) and IO EVENTS data. This information is fed into machine learning models, including, XGBOOST, LSTM, and CNN2D. The proposed architecture ensures minimal impact on system performance during feature extraction. Various experiments with machine learning algorithms demonstrate Random Forest, Voting Classifier and XGBOOST consistently achieving over 98% accuracy in predicting Ransomware. Additionally, the extension introduces CNN2D for advanced feature optimization through multi-layered 2D convolution neural networks, enhancing dataset relevance and improving overall detection accuracy. Notably, the system does not employ advanced feature optimization algorithms beyond traditional techniques.

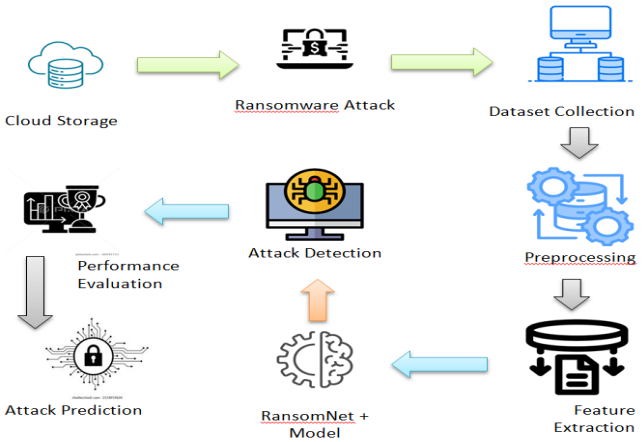

Fig. 1. Proposed System Architecture

3.3 Dataset

The dataset, namely 'HPC_41Events_5Rounds' for HPC events and 'IO_41Events_5Rounds' for IO events, encompasses a total of 6000 samples. These samples are characterized by 12 features, namely as Instructions, Last Level Cache, L1 icache load misses, Branch load misses, node load misses, read requests, read bytes, write requests, write bytes, data explicitly remove, read total time, write total time, and flush total time. Among these samples, 4800 are designated for testing purposes, while the remaining 1200 are allocated for training. The datasets were procured from the Harvard Dataverse.[9][10]

3.4 Data preprocessing

In data preprocessing, pandas and numpy efficiently handle dataset manipulation tasks. The dataset is loaded into a pandas dataframe for easy management, and unnecessary columns are dropped. Normalization techniques such as Min-Max scaling or Z-score normalization ensure consistent feature ranges. This optimization addresses data cleanliness, relevance, and numerical stability concerns. It lays a robust foundation for subsequent machine learning model training and evaluation processes.

3.5 Feature Selection

Feature selection is essential for identifying and retaining relevant features that enhance a machine learning model's predictive power. It involves visualization tools like seaborn and matplotlib to understand feature distributions and relationships, aiding in pattern identification. Label encoding converts categorical data into numerical format for model interpretation. Techniques like Recursive Feature Elimination (RFE) or feature importance from tree-based models objectively rank and select features, reducing dataset dimensionality and improving model efficiency and accuracy.

3.6 Algorithms

XGBOOST

Initialize $f_0(p)$;

For $j = 1,2 \ldots . m$ do

Calculate $g_j = \frac{\partial L(q,f)}{\partial f}$;

Calculate $h_j = \frac{\partial^2 L(q,f)}{\partial f^2}$;

create splits with maximum gain $A = \frac{1}{2}\left[\frac{M_L^2}{N_L} + \frac{M_R^2}{N_R} - \frac{M^2}{N}\right]$

leaf weights Calculation

$$W = -\frac{M}{N}$$

Identifying the base learner $b\hat{}(p) = \varepsilon_{j=1}^{T} wl$;
Add trees $f(p) = f_{k-1}(p) + b^{(p)}$;
end for
Result:$f(p) = \varepsilon_{K=0}^{L} f_k(p)$

CNN2D.

Input:
Ransomware Sample Data
Training : (Xtrain1, ytrain1)
Testing : (Xtest1, ytest1)
Output:
Predict: ypred
Evaluate: performance , Accuracy
Preprocess data:
Split dataset into training and testing : (Xtrain1, ytrain1), (Xtest1, ytest1)
Reshape the data for CNN input
Xtrain1 = reshape(Xtrain1, (n, m, 1, 1))
Xtest1 = reshape(Xtest1, (n', m', 1, 1))
Define CNN Architecture:
Initialize a CNN model: CNN = Sequential()
CNN.add(Convolution2D(64, (1, 1), input_shape=(m, 1, 1), active tion='relu'))
CNN.add(MaxPooling2D((1, 1)))
Flatten the feature maps: CNN.add(Flatten())
Compile CNN Model:
CNN.compile(loss='categorical_crossentropy',optimizer='adam', metrics=['accuracy'])
Evaluating model:
y_pred = argmax(CNN.predict(Xtest1))
Metrics = evaluate(ypred, ytest)

VOTING CLASSIFIER.

Applying 3 classifiers (Random Forest, voting classifier,Adaboost) on training data:

```
Clf1=AdaBoostclassifier(n_estimators=10
            ,random_state=0)
Clf2=RandomForestclassifier(n_estimators
          =5, random_state=1)
eclf = VotingClassifier(estimators=[('ad',
     clf1), ('rf', clf2)], voting='soft')
```

Comparing performance:

```
calculateMetrics("Voting Classifier", predict, y_test)
```

Perform Majority Voting:

```
eclf = VotingClassifier (estimators=[('ad', clf1), ('rf', clf2)], voting='soft')
```

4 Experiment Results

4.1 Accuracy

The accuracy of a ransomware detection method is its capacity to correctly identify infected and uninfected systems. To gauge accuracy, it's crucial to calculate the ratio of true positives and true negatives among all instances. Mathematically, this is represented as:

$$\boldsymbol{Accuracy} = \frac{\boldsymbol{TP + TN}}{\boldsymbol{TP + TN + FP + FN}}$$

TP ← True positive, TN ← True Negative, FP←False Positive, FN ← False Negative

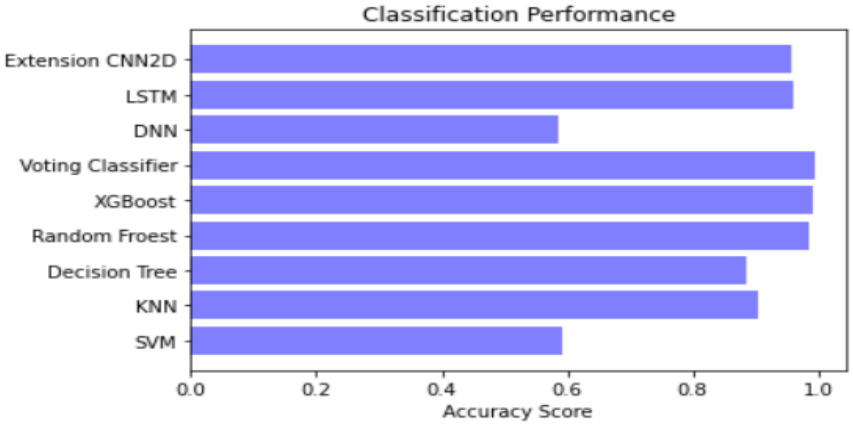

Fig. 2. Accuracy comparison graph

4.2 Precision

Precision assesses the proportion of accurately classified instances or samples among those identified as positives. Therefore, the formula to compute precision is as follows:

$$\boldsymbol{Precision} = \frac{\boldsymbol{True\ Positive}}{\boldsymbol{True\ Positive + False\ Positive}}$$

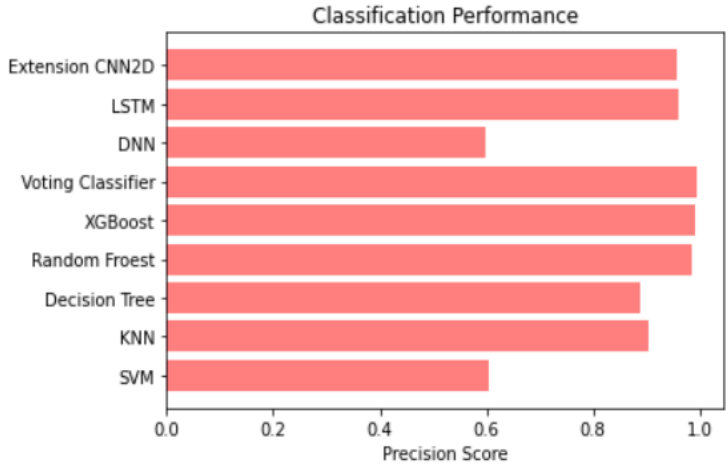

Fig. 3. Precision Comparison graph

The above graph is about the performance of different algorithms versus the Precision score.

4.3 Recall

Recall serves as a vital metric in machine learning, measuring the model's aptitude in identifying all pertinent instances of a particular class. It quantifies the proportion of accurately predicted positive observations among the total actual positives, providing valuable insights into the model's thoroughness in capturing instances of the specified class:

$$\boldsymbol{Recall} = \frac{\boldsymbol{True\ Positive}}{\boldsymbol{True\ Positive} + \boldsymbol{False\ Negative}}$$

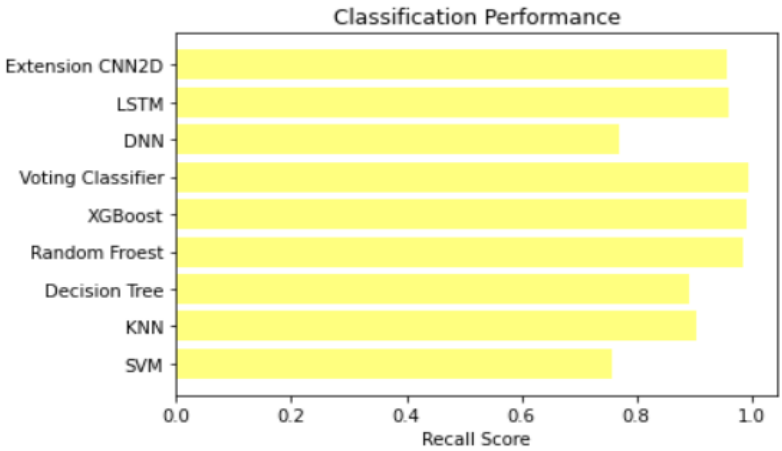

Fig. 4. Recall comparison graph

The above graph is about the performance of different algorithms versus the Recall score.

4.4 F1- Score

The F1 score combines precision and recall to assess a model's accuracy in machine learning, providing a balanced evaluation of performance:

$$F1\ Score = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}}$$

$$F1\ Score = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

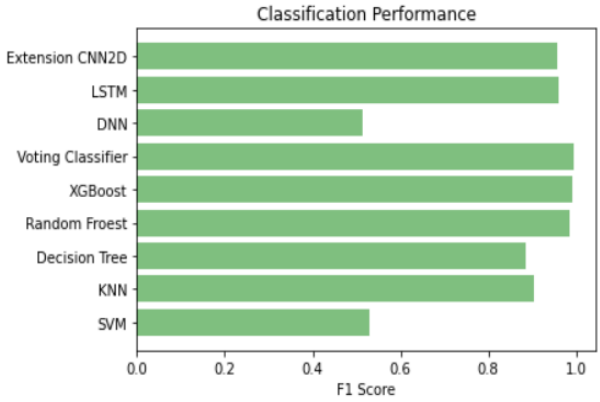

Fig.5. F1 comparison graph

The above graph is about the performance of different algorithms versus the F1 score.

4.5 Algorithms comparison table

The table presents a comparison of different algorithms alongside the proposed ones, showing Accuracy, Precision, Recall, and F1 score.

Table 1. Performance Evaluation Table

ML Model	Accuracy	Precision	Recall	F1-score
SVM	0.593	0.605	0.738	0.329

KNN	0.904	0.903	0.903	0.904
Decision Tree	0.883	0.887	0.891	0.883
Random Forest	0.985	0.983	0.985	0.983
XG Boost	0.992	0.991	0.992	0.992
Voting Classifier	0.993	0.995	0.993	0.993
DNN	0.383	0.597	0.769	0.313
LSTM	0.960	0.961	0.961	0.960
Extension CNN2D	0.936	0.937	0.937	0.936

5 Conclusion

This research introduces a new method for quickly and accurately detecting ransomware in a virtual environment. The approach involves closely monitoring the processor and disk I/O activities on the host machine and then using machine learning techniques for analysis. To gather data, the perf tool and hardware performance counters (HPCs) capture processor event data, while virsh domblkstats (domain block statistics) is used to obtain disk I/O event data. The evaluation phase of the study involved testing five machine learning (ML) and two deep learning (DL) classifiers. Each classifier was tested with three models: one utilizing HPC data, another utilizing disk I/O data, and a third combining both types of data. In addition to these models, the study incorporated an algorithm voting classifier. Impressively, this classifier achieved an accuracy of 0.993, surpassing existing models such as Support Vector Machine (SVM) with an accuracy of 0.593, K-Nearest Neighbors (KNN) with an accuracy of 0.904, Decision Trees with an accuracy of 0.883, and Random Forest with an accuracy of 0.983.

6 Feature Scope

In future, the intention is to employ the developed models for real-time ransomware detection during execution. Although the current model is tailored for VMs, it aims to modify it for use on standalone machines in upcoming research. The system have yet

to assess whether the models optimized for one machine configuration perform effectively on different configurations, such as those with increased memory or additional CPU cores. This aspect will be a focus of future investigations.

7 References

1. SR Department, Ransomware victimization rate 2022. Accessed: Apr. 6, 2022. [Online]. Available: https://www.statista. com/statistics/204457/businesses-ransomware-attack-rate/ (2022).
2. D. Braue, Ransomware Damage Costs. Accessed: Sep. 16,2022. Available: https://cybersecurityventures.com/globalransomware-damage-costs-predicted-to-reach-250-billion-usd-by-2031/ (2022).
3. Logix Consulting, What is Signature Based Malware Detection. Accessed: Apr. 3, 2023. [Online]. Available: https://www.logixconsulting. com/2020/12/15/what-is-signature-based-malware-detection/ (2020).
4. W. Liu, P. Ren, K. Liu, and H.-X. Duan, Behavior-based malware analysis and detection. Proc. 1st Int. Workshop Complex, Data Mining, Sep. 2011, pp. 39–42.
5. S. Aurangzeb, R. N. B. Rais, M. Aleem, M. A. Islam, and M. A. Iqbal, On the classification of microsoft-windows ransomware using hardware profile. PeerJ Comput. Sci., vol. 7, p. e361, (02/2021).
6. S. Homayoun, A. Dehghantanha, M. Ahmadzadeh, S. Hashemi, and R. Khayami, Know abnormal, find evil: Frequent pattern mining for ransomware threat hunting and intelligence. IEEE Trans. Emerg. Topics Comput., vol. 8, no. 2, pp. 341–351, (04,2020).
7. N. Pundir, M. Tehranipoor, and F. Rahman RanStop: A hardwareassisted runtime crypto-ransomware detection technique. arXiv:2011.12248 (2020).
8. M. Alam, S. Bhattacharya, S. Dutta, S. Sinha, D. Mukhopadhyay, and A. Chattopadhyay, RATAFIA: Ransomware analysis using time and frequency informed autoencoders. Proc. IEEE Int. Symp. Hardw. Oriented Secur. Trust (HOST), pp. 218–227 (05/2019).
9. Reddy Madhavi, K., A. Vinaya Babu, G. Sunitha, and J. Avanija. "Detection of concept-drift for clustering time-changing categorical data: An optimal method for large datasets." In Data Engineering and Communication Technology: Proceedings of 3rd ICDECT-2K19, pp. 861-871. Springer Singapore, 2020.
10. K. Thummapudi, R. Boppana, and P. Lama, IO 41 events 5 rounds, Harvard Dataverse, DOI: 10.7910/DVN/GHJFUT (2022).
11. Durre Zehra Syeda, Mamoona Naveed Asghar, Dynamic Malware Classification and API Categorisation of Windows Portable Executable Files Using Machine Learning. Applied Sciences, (2024).
12. V. Jyothsna, E. Sandhya, Thammisetty Swetha, P. Lokesh Kumar Reddy, B. Jyothsna, P.Bhasha, Deep Learning Model for Intrusion Detection. SDN Network, 1st International Conference on Optimization Techniques for Learning (ICOTL), (2023).
13. Seunghye Lee, Taeseop Kim, Qui X. Lieu, Thuc P. Vo, Jaehong Lee, A novel data-driven analysis for sequentially formulated plastic hinges of steel frames, Computers & Structures, (2023).
14. M. Rhode, P. Burnap, and A. Wedgbury, Real-time malware process detection and automated process killing. Secur. Commun. Netw., vol. 2021, pp. 1–23, (12/2021).
15. Seunghye Lee, Taeseop Kim, Qui X. Lieu, Thuc P. Vo, Jaehong Lee, A novel data-driven analysis for sequentially formulated plastic hinges of steel frames, Computers & Structures, (2023).