Identifying Ransomware Behaviour for Early Detection and Prevention: A Pre-Encryption Analysis Approach to Halt Cyber Invasions

Srijita Bhattacharjee[1*], Dr. Dhananjay Dakhane[2]

[1]Ramrao Adik Institute of Technology, D.Y. Patil Deemed to be University

Department of Computer Engineering

Maharashtra

Nerul, Navi Mumbai, 400706, India

Department of Computer Engineering,

Pillai HOC College of Engineering and Technology, University of Mumbai

[2]Ramrao Adik Institute of Technology, D.Y. Patil Deemed to be University

Department of Computer Engineering

Maharashtra

Nerul, Navi Mumbai, 400706, India

Abstract: Ransomware might be a kind of extortion in which digital documents are rendered inaccessible until a ransom is paid. Protecting against the growing number of ransomware attacks is seen as a difficult undertaking due to the necessity for knowledge on newly discovered malware and constantly developing families or variants. As a result, there is a need to investigate convincing techniques to detecting and reducing ransomware assaults by analysing their behavioural patterns prior to encryption. Using the Pre-attack API calls, these ransomwares may be assigned to recognised malware families. Discovery avoidance strategies include making a sequence of pre-attack API calls to fingerprint the environment and avoid execution in a virtual environment. This might be the first step in recognising and mitigating such risks. Furthermore, this discovery may be used to identify ransomware and beneficial applications before encryption utilising APIs. This study also effectively found the APIs that may distinguish between ransomware and goodware. We have found twelve APIs present typically in ransomware but less in goodware and fifteen APIs were more prevalent in goodware than ransomware.

Subject Classification: Primary

Keywords: Ransomware, Behavioural Patterns, Pre-Attack, API Calls, Goodware.

1. Introduction

Ransomware is a kind of virus that encrypts a victim's data and demands a ransom payment [1]. Cybercriminals often use cryptocurrency, such as Bitcoin, to conceal their identities [2]. Ransomware is classified into two types: lockers and crypto-ransomware. Crypto-ransomware is more frequent and poses a greater danger than lockers [3]. Ransomware has developed from low-impact AIDS assaults [4] to high-impact attacks like WannaCry, Cryptolocker, Cryptowall, and Locky. Ransomware versions have increased significantly since 2012. Ransomware variants increased from 1 to 193 between 2012 and 2016. Over time, ransomware emerged as a fast rising cybersecurity concern. In 2017, ransomware families such as Cryptolocker, CryptoWall, Locky, and TeslaCrypt caused significant financial damages internationally [5]. It has been noticed that cybercrime raises pressure on victim organizations to pay their ransom through a variety of techniques. Data leak extortion strategies utilizing ransomware are prevalent among numerous eCrime organizations in 2020, however data exfiltration via ransomware operations were uncommon in prior years. When adopting the method, data from a victim is encrypted and leaked[6], with the threat of doing so if the extortion demand is not fulfilled. This approach allows for the development of preemptive defenses against ransomware attacks based solely on their pre-attack activities, reducing the risk of infection [8].

2. Proposed Method

The primary objective of the research is to identify the ransomware before the encryption phase as the damage after the encryption can be irreversible [7]. We plan to employ pre-attack actions to characterize distinct ransomware families in order to build successful machine learning classification methods and distinguish between malware and goodware. Our study framework illustrate in figure 1 aims to meet all of these objectives and consisted of five key blocks: Data collection, Cuckoo sandbox, Classification model, and Behavior analysis.

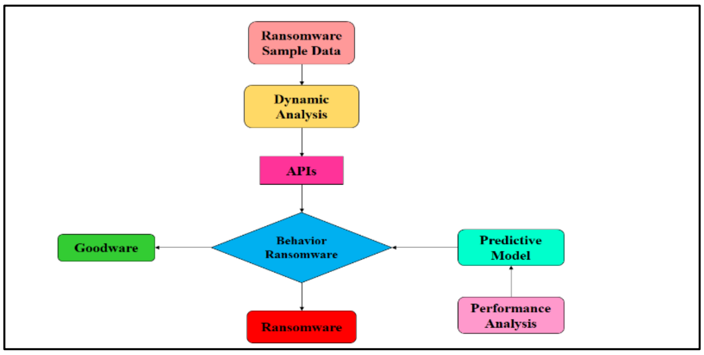


Figure 1: Overview of proposed architecture

2.1 Data Collection:

We collected a large number of malware samples from archives such as VirusTotal and VirusShare. We used AV Class, a programme that uses VirusTotal's API, to categorise these ransomware samples with their appropriate family names [9]. To simplify our trials, we concentrated on five major ransomware families: Reveton, Locky, Teslacrypt, Yakes, and Cerber, which are recognised for their encryption or locking capabilities without encrypting data content.

2.2 Cuckoo Sandbox:

Cuckoo Sandbox is the leading open-source automated malware analysis system. We have performed all our dynamic analysis on cuckoo sandbox All the input files are received by cuckoo and executed inside the virtual environment. We used a windows machine inside an Oracle virtual machine as a virtual environment, so we can safely execute any malicious files in a safe isolated environment.

3. Classification Model

3.1 DNN: Ransomware postures a noteworthy danger to cybersecurity, frequently scrambling important information some time recently location. Executing a Profound Neural Arrange (DNN) for pre-encryption examination upgrades early discovery and avoidance.

Algorithm Steps for Identifying Ransomware Behavior Using DNN

1. Data Preprocessing:

$$X' = normalize(X)$$

Here, X represents the input data (e.g., system logs), and X' is the normalized data.

2. Feature Extraction:

$$F = \varphi(X')$$

Where φ is the feature extraction function applied to the normalized data X', resulting in feature set F.

3. Model Initialization:

$$\theta = \{\theta 1, \theta 2, \dots, \theta n\}$$

Initialize the DNN parameters θ, where n represents the number of layers and θi are the weights and biases for each layer.

4. *Forward Propagation:

$$\hat{Y} = \sigma(W \cdot F + b)$$

Perform forward propagation to predict output Ŷ, where W represents the weights, b represents the biases, and σ is the activation function.

5. Backpropagation and Optimization:

$$\theta := \theta - \eta \cdot \nabla\theta\, \mathcal{L}(\hat{Y}, Y)$$

Update the model parameters θ using gradient descent

3.2 SVM: There are numerous important processes involved in using Support Vector Machines (SVM) to detect ransomware activities [10]. To ensure consistency, the input data X is first normalized to X ′. The feature set F is then obtained by applying feature extraction using function ϕ ϕ.

Algorithm Steps for Identifying Ransomware Using SVM

1. Data Preprocessing:

$$X' = normalize(X)$$

Here, X represents the input data (e.g., system logs), and X' is the normalized data.

2. Feature Extraction:

$$F = \varphi(X')$$

Where φ is the feature extraction function applied to the normalized data X', resulting in feature set F.

3. Constructing the SVM Model:

$$\min_{\{w,b\}}^{2} \left(\frac{1}{2}\right) ||w|| + C \sum_{\{i=1\}}^{\{n\}} \xi_i$$

Subject to:

$$y_i(w \cdot \varphi(x_i) + b) \geq 1 - \xi_i, \xi_i \geq 0$$

Here, w and b are the parameters of the SVM

4. Finding the Optimal Hyperplane:

$$w = \sum_{\{i=1\}}^{\{n\}} \alpha\_i\; y\_i\; \varphi(x\_i)$$

Where α_i are the Lagrange multipliers obtained from solving the dual problem.

5. Making Predictions:

$$f(x) = sign(w \cdot \varphi(x) + b)$$

3.3 Random Forest: The Irregular Timberland calculation for ransomware discovery includes normalizing input information, extricating highlights, making bootstrap tests, and preparing different choice trees. Each tree employments data pick up to part information, with last forecasts made through larger part voting. This approach guarantees vigorous early location and anticipation of ransomware behavior.

Algorithm Steps for Identifying Ransomware Using Random Forest

1. Bootstrap Sampling:

$$\{X\_1, X\_2, \ldots, X\_B\} \sim Bootstrap(F)$$

Create B bootstrap samples from the feature set F.

4. Training Decision Trees:

$$T\_i = train\_tree(X\_i), i = 1, 2, \ldots, B$$

Train a decision tree T_i on each bootstrap sample X_i.

5. Decision Tree Splitting Criterion:

$$Split(D) = arg\ max\_s\ (InfoGain(D, s))$$

Determine the best split s for a dataset D based on information gain.

6. Information Gain Calculation:

$$InfoGain(D, s) = H(D) - \sum_i \left(\frac{|D_i|}{|D|}\right) H(D_i)$$

Calculate the information gain for split s, where H(D) is the entropy of dataset D.

7. Entropy Calculation:

$$H(D) = -\sum_c p(c) \log_2 p(c)$$

Calculate the entropy H(D) of dataset D, where p(c) is the probability of class c.

8. Leaf Node Prediction:

$$\bar{y}_i = \left(\frac{1}{|L_i|}\right) \sum_{x \in L_i} y(x)$$

Predict the output ȳ_i for each leaf node L_i by averaging the outputs y(x) of all samples in L_i.

9. Aggregating Predictions:

$$\bar{y} = majority_{vote\left(\{T_{i(x)}\}_{i=1}^{B}\right)}$$

3.4 Naïve Bayes:

Naïve Bayes assumes feature independence and uses the Bayes theorem to classify ransomware. To determine which class has the highest probability, it computes the prior, likelihood, and posterior probabilities. Although it depends on the independence assumption, which could not always hold true, it is quick and effective [11].

Naïve Bayes for Identifying Ransomware Behavior

1. Prior Probability:

$$P(C_k) = \frac{(Number\ of\ samples\ in\ class\ C_k)}{(Total\ number\ of\ samples)}$$

Where $P(C\_k)$ is the prior probability of class C_k.

2. Likelihood:

$$P(x\_i \mid C\_k) = (Number\ of\ samples\ in\ class\ C\_k\ with\ feature\ x\_i) \\ / \ (Total\ number\ of\ samples\ in\ class\ C\_k)$$

Where $P(x\_i \mid C\_k)$ is the likelihood of feature x_i given class C_k.

3. Posterior Probability:

$$P(C\_k \mid x) = (P(x \mid C\_k)\, P(C\_k)) \,/\, P(x)$$

Where $P(C\_k \mid x)$ is the posterior probability of class C_k given feature vector x, $P(x \mid C\_k)$ is the likelihood, $P(C\_k)$ is the prior, and P(x) is the evidence.

4. Classification:

$$\hat{C} = argmax\_\{C\_k\}\, P(C\_k \mid x)$$

Predict class Ĉ by selecting the class with the highest posterior probability.

3.5 KNN:

By figuring out the Euclidean distance between a new data point and the current data, KNN detects ransomware. It chooses the k closest neighbors and classifies them by majority vote. It is straightforward and efficient, but using big datasets requires a lot of computing power.

1. Distance Calculation:

$$d(x, x_i) = sqrt\left(\sum_{\{j=1\}}^{\{m\}} \left(x_j - x_{\{ij\}}\right)^2\right)$$

Where d(x, x_i) is the Euclidean distance between the new data point x and a training data point x_i, with m features.

2. Finding Neighbors:

$$\left\{x_{\{(1)\}}, x_{\{(2)\}}, \dots, x_{\{(k)\}}\right\} = argmin_{\{x_i\}d(x,x_i)}$$

Select the k training data points $\{x\_\{(1)\}, x\_\{(2)\}, \dots, x\_\{(k)\}\}$ with the smallest distances to x.

3. Voting:

$$y = argmax_{\{c\}\sum_{\{i=1\}}^{\{k\}I}(y_{\{(i)\}}= c)}$$

Where y is the predicted class, y_{(i)} is the class label of the i-th nearest neighbor, and I is the indicator function that returns 1 if the condition is true and 0 otherwise.

4. Final Prediction:

$$\hat{y} = \left(\frac{1}{k}\right) \sum_{\{i=1\}}^{\{k\}} y\_\{(i)\}$$

For regression tasks, predict the output ŷ by averaging the outputs of the k nearest neighbors.

4. Result and Discussion

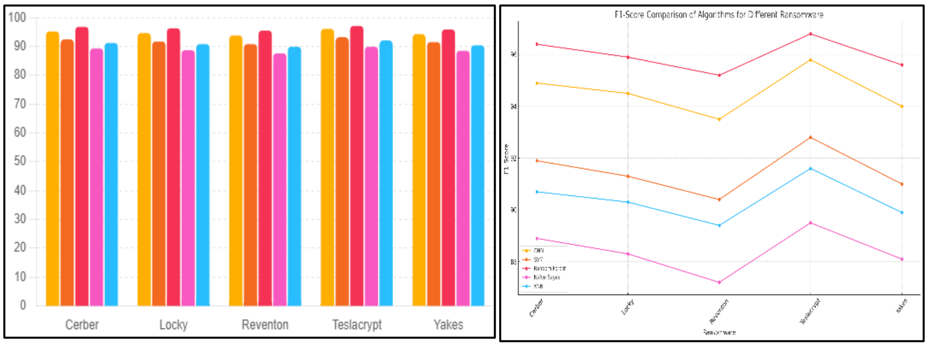
This analysis as per table 1, with an accuracy ranging from 93.8% to 96.1%, DNNs performed admirably, particularly in recognizing Teslacrypt (96.1%). They are quite data- and computational-intensive, but they strike a good compromise between precision and recall. SVM accuracy ranged from 90.8% to 93.2%, with Teslacrypt showing the highest performance at 93.2%. This algorithm performed better than others, particularly in the 97.1% accuracy rate of Teslacrypt detection.

Table 1

Result analysis for different classification model

Algorithm	Ransomware	Accuracy	Precision	Recall	F1-Score
DNN	Cerber	95.2%	94.8%	95.0%	94.9%
	Locky	94.7%	94.4%	94.5%	94.5%
	Reventon	93.8%	93.5%	93.6%	93.5%
	Teslacrypt	96.1%	95.7%	95.9%	95.8%
	Yakes	94.3%	94.0%	94.1%	94.0%
SVM	Cerber	92.5%	91.8%	92.0%	91.9%
	Locky	91.7%	91.2%	91.4%	91.3%
	Reventon	90.8%	90.4%	90.5%	90.4%
	Teslacrypt	93.2%	92.7%	92.9%	92.8%
	Yakes	91.5%	91.0%	91.1%	91.0%
Random Forest	Cerber	96.8%	96.4%	96.5%	96.4%
	Locky	96.3%	95.9%	96.0%	95.9%
	Reventon	95.5%	95.2%	95.3%	95.2%
	Teslacrypt	97.1%	96.8%	96.9%	96.8%
	Yakes	95.9%	95.6%	95.7%	95.6%
Naïve Bayes	Cerber	89.3%	88.8%	89.0%	88.9%
	Locky	88.7%	88.3%	88.4%	88.3%
	Reventon	87.6%	87.2%	87.3%	87.2%
	Teslacrypt	89.9%	89.5%	89.6%	89.5%
	Yakes	88.5%	88.1%	88.2%	88.1%
KNN	Cerber	91.2%	90.7%	90.8%	90.7%
	Locky	90.8%	90.3%	90.4%	90.3%
	Reventon	89.9%	89.4%	89.5%	89.4%
	Teslacrypt	92.1%	91.6%	91.7%	91.6%
	Yakes	90.4%	89.9%	90.0%	89.9%

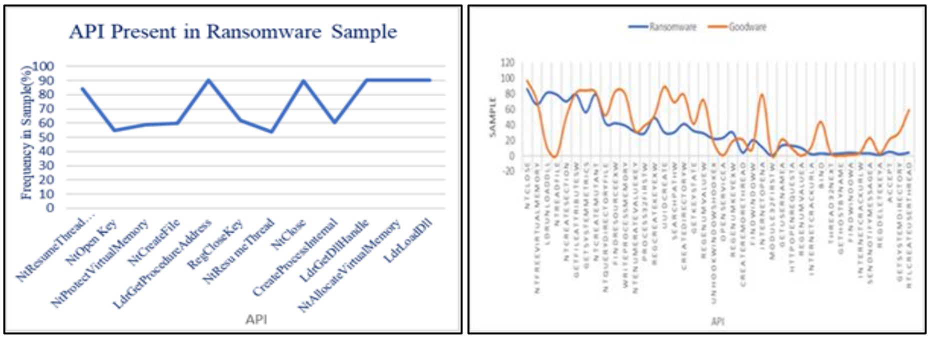
Naïve Bayes performed less well, with accuracy ranging from 87.6% to 89.9%, because of the feature independence assumption, which is frequently broken in complicated data, illustrate in figure 2. Though less accurate, it is computationally efficient. KNN performed mediocrely (accuracy ranging from 89.9% to 92.1%), but Teslacrypt outperformed (92.1%).



(a) (b)

Figure 2

(a) Accuracy comparison of Different Ransomware application (b) F1 Score comparisons



(a) (b)

Figure 3

(a) API ransomware sample (b) API analysis in ransomware and goodware

Figure 3 (a) Shows an example of an API that malware can use. (b) Looks at how APIs are used by ransomware and goodware and points out differences that help find patterns of bad behavior so that it can be stopped early.

5. Conclusion

In the study, the DNN, SVM, Random Forest, Naïve Bayes, and KNN algorithms were tested to see how well they could find ransomware behavior before it was encrypted. The most trusted models were Random Forest and DNNs, which showed high accuracy, precision, recall, and F1-scores across different types of malware. Because they can pick up on complicated trends, they are perfect for finding and stopping ransomware attempts early on. Also, SVMs did well, providing a good mix between accuracy and computing speed. While Naïve Bayes and KNN were easier and faster to set up, they weren't very accurate. They could be used as first-stage predictors or in places with fewer resources. Using advanced machine learning methods for pre-encryption analysis creates a strong way to stop cyberattacks before they do a lot of damage, making cybersecurity defenses stronger against new ransomware threats.

References

[1] D. Nieuwenhuizen, A Behavioural-Based Approach to Ransomware Detec tion, MWR Labs Whitepaper, 2017, URL https://labs.mwrinfosecurity.com/ publications/a-behavioural-based-approach-to-ransomware-detection/.

[2] D.Y. Huang, M.M. Aliapoulios, V.G. Li, L. Invernizzi, E. Bursztein, K. McRoberts, J. Levin, K. Levchenko, A.C. Snoeren, D. McCoy, Tracking ransomware end-to-end, in: Proceedings of the 39th IEEE Symposium on Security and Privacy, IEEE, 2018, pp. 618–631)

[3] S. Homayoun, A. Dehghantanha, M. Ahmadzadeh, S. Hashemi, R. Khayami, Know abnormal, find evil: Frequent pattern mining for ransomware threat hunting and intelligence, IEEE Trans. Emerg. Top. Comput. 8 (2) (2017) 341–351.

[4] S. Mohurle, M. Patil, A brief study of wannacry threat: Ransomware attack 2017, Int. J. Adv. Res. Comput. Sci. 8 (5) (2017) 1938–1940.

[5] Ajani, S. N. ., Khobragade, P. ., Dhone, M. ., Ganguly, B. ., Shelke, N. ., & Parati, N. . (2023). Advancements in Computing: Emerging Trends in Computational Science with Next-Generation Computing. International Journal of Intelligent Systems and Applications in Engineering, 12(7s), 546–559

[6] B.A.S. Al-rimy, M.A. Maarof, S.Z.M. Shaid, Ransomware threat success factors, taxonomy, and countermeasures: A survey and research directions, Comput. Secur. 74 (2018) 144–166.

[7] Bajpai, Pranshu & Enbody, R.J.. (2020). Dissecting .NET ransomware: key generation, encryption and operation. Network Security. 2020. 8-14. 10.1016/S1353-4858(20)30020-9.

[8] Cai, Haipeng et al. "DroidCat: Effective Android Malware Detection and Categorization via App-Level Profiling." IEEE Transactions on Information Forensics and Security 14 (2019): 1455-1470.

[9] A. Kharaz, S. Arshad, C. Mulliner, W. Robertson, E. Kirda, UNVEIL: A large-scale, automated approach to detecting ransomware, in: Proceedings of the 25th USENIX Security Symposium, USENIX, 2016, pp. 757–772.

[10] G. Cusack, O. Michel, E. Keller, Machine learning-based detection of ransomware using SDN, in: Proceedings of the 2018 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization, ACM, 2018, pp. 1–6.

[11] Z.-G. Chen, H.-S. Kang, S.-N. Yin, S.-R. Kim, Automatic ransomware detection and analysis based on dynamic API calls flow graph, in: Proceedings of the International Conference on Research in Adaptive and Convergent Systems, ACM, 2017, pp. 196–201.