International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 9
***Article Received:** 25 July 2023 **Revised:** 12 September 2023 **Accepted:** 30 September 2023*

Hypervisor-Level Ransomware Detection in Cloud Using Machine Learning

[1]Prasad Purnaye, [1]Anuj Singh, [1]Mayank Singh, [1]Suprabhath Nair, [1]Devanshu Mehta
[1] Dr Vishwanath Karad MIT World Peace University, Pune, India
prasad.purnaye@mitwpu.edu.in, anujsingh2409@gmail.com, mayank.singh011@gmail.com, suprabhathnair@gmail.com, devanshumehta8@gmail.com.

Abstract— Ransomware attack incidences have been on the rise for a few years. The attacks have evolved over the years. The severity of these attacks has only increased in the cloud era. This article discusses the evolution of ransomware attacks targeting cloud storage and explores existing ransomware detection solutions. It also presents a methodology for generating a dataset for detecting ransomware in the cloud and discusses the results, including feature selection and normalization. The article proposes a system for detecting attacks in virtualized environments using machine learning models and evaluates the performance of different classification models. The proposed system is shown to have high accuracy of 96.6% in detecting ransomware attacks in virtualized environments at the hypervisor level.

Index Terms—cybersecurity, cloud computing, virtualization, ransomware detection.

I. INTRODUCTION

Ransomware encrypts an organization's crucial data and files that are stored on servers, storage area networks, and endpoint devices. It has recently been named one of the most popular and successful viruses targeted against enterprises. After encrypting the crucial data, the attacker demands a ransom from the victim, and if payment is not made by the specified date, the victim's data would be permanently lost. Given the fast growth of cloud computing, it is not surprising that ransomware has been targeting cloud storage. Often, a user sets up their computer so that the files stored on their computer are synchronized with their other devices via cloud-based storage. Once a file is encrypted by ransomware, all copies of the file that are synchronized with the encrypted file also become encrypted. So, if a user has multiple devices the copy of the file on each device and the cloud storage all become encrypted.

II. LITERATURE SURVEY

A. Most Common Attacks

According to the technique used to extort the victim, there are two major categories of ransomware. These two varieties are Ransomware-locker and Ransomware-crypto (also known as cryptographic malware). A ransomware-locker stops the victim from accessing his or her device, i.e. The user is not permitted to use their system or device until the specified ransom is paid. Attacks using cryptocurrency-based ransomware prohibit victims from accessing files or data. This is also called a data locker since it locks data. A few of the most common ransomware attacks are discussed as follows.

a) WannaCry: The WannaCry ransomware attack was a global cyberattack that occurred in May 2017. It took advantage of a flaw in Microsoft Windows operating systems that the National Security Agency (NSA) had identified and released online. The malware encrypted files on infected computers and demanded a ransom payment in Bitcoin to restore access to the files[2].

b) Crypto Locker: It encrypts the user's files and demands payment in exchange for the decryption key to unlock the files. The attackers typically ask for payment in Bitcoin. If the ransom is not paid within the given time frame, the decryption key is permanently deleted, and the user's files remain encrypted and inaccessible. The most common ways that Cryptolocker distributes are through email attachments, dangerous links, or drive-by downloads from infected websites [3].

c) NotPetya: In June 2017, a widespread hack known as "NotPetya" took place. The attack spread across networks using a technique akin to a worm, infecting machines, and encrypting their files. To spread rapidly, NotPetya made use of a vulnerability shared by WannaCry. Once it had been installed, the ransomware requested payment in Bitcoin to unlock the files[4].

d) The AIDS trojan, the first known ransomware, was discovered in 1989. Since then, the prevalence of these attacks has increased dramatically, reaching over 304 million attacks in 2020, an increase of more than 62% from 2019[5].

e) Locky, CryptoLocker, FBI MoneyPak [6], and WannaCry are a few well-known instances. These attacks are estimated to have cost over $4 billion in damages. Locky is a well-known ransomware that was published in 2016. Malicious macros are included in the email attachment that carries the ransomware. The attachment asks the user to activate macros when they are opened by the user. Whenever a user turns on macros, malware is downloaded and run. In the year 2022, 67% of IaaS Cloud users were hit by ransomware globally.

B. Existing Ransomware detection solutions

Although ransomware poses a concern, there are several ways to lessen its effects and guarantee some level of safety. Below, we go over several of these widely utilized defenses.

a) Repositories and Data Sharing: Repositories and data sharing can be effective countermeasures to ransomware attacks [7]. Organizations can lessen the effects of a ransomware attack by having the ability to restore their data quickly and easily by keeping up-to-date backups of crucial data in repositories that are not directly available to users. Additionally, by sharing data across multiple locations, organizations can prevent a single point of failure and reduce the likelihood of a successful attack.

b) Behavior-based detection: It is a method of detecting and preventing ransomware attacks by analyzing the behavior of software and network traffic [8]. It entails keeping an eye on system events and comparing them to recognized attack patterns to spot suspicious behavior. When a device is infected, the system can instantly isolate it to stop the ransomware from spreading.

c) Reverse Engineering: Reverse engineering can be used as a countermeasure to identify the specific malware used, the attack methods used to deliver it, and any vulnerabilities in the system that were exploited. This information can be used to develop mitigation strategies to prevent future attacks, such as developing and implementing patches or deploying security controls. The most important use case for this method is when prevention techniques used by antivirus software fail, resulting in system damage [9].

d) Risk Disclosure and Awareness: Risk disclosure involves identifying potential vulnerabilities and weaknesses in an organization's systems and taking proactive measures to address them before they are exploited by cybercriminals. Employees must be made aware of the dangers posed by ransomware and the precautions they can take to protect themselves, including avoiding suspicious emails, not opening links from unfamiliar sources, and not downloading attachments.

e) Decentralization and All-Or-Nothing Transforms: Instead of depending on a single centralized server or network, decentralization is a method for preventing ransomware attacks by distributing data and resources across multiple systems or nodes. All-or-nothing transforms are cryptographic techniques used to protect data from ransomware attacks by encrypting it in such a way that the entire dataset must be decrypted or none of it can be accessed. This prevents attackers from selectively encrypting parts of the data and demanding payment for each piece. There is a research gap observed for ransomware detection at the hypervisor level in the cloud[14].

III. Methodology

In this paper, we present a methodology to detect the ransomware attack at the hypervisor level. The ransomware attack mainly focuses on encrypting the files and data of the victim. This act almost always calls for an unprecedented rise in resource activities. This behavior signature can be tapped for detection of the attack at the hypervisor level. We have applied an AI agent-based technique to continuously monitor the activity and trigger the evidence collection module for the further process.

A. Design of the experiment

To generate the dataset a private cloud was set up. The system configuration includes AMD Ryzen 9, 5900HS Processor with 16 GB of RAM with 1TB of SSD. The private cloud setup was done using a KVM type-1 hypervisor along with OpenNebula (version 5.12) as a cloud management platform. To simulate the real-time cloud environment a script generating a synthetic workload was deployed on the virtual machines of the cloud. The virtual machines were running Ubuntu Linux OS.

To simulate the usual workload a python script was run on the virtual machine. The Python script simulates network activity by retrieving data in JSON format from a URL using the requests library. It then writes the data to a file named "testing.txt" to represent read and write operations on RAM. Additionally, the script makes use of the random and time libraries to add some randomness to how long each iteration of the loop that gets the data and writes it to the file takes to complete. This happens every two minutes. The program continuously checks for active scheduled tasks while sleeping for one second in between checks. On these virtual machines, a ransomware Python script was deployed. The Libvirt API runs for 157 minutes to collect data at regular intervals of a second.

B. Ransomware

Due to the inability to get ahold of ransomware software a python script was used to stimulate the attack on the VM. To simulate the attack 20000 files were created on the virtual machine with data sizes ranging from 0.1 to 1 MB. A Python script was written as a substitute for ransomware. The script generates an RSA [15]key pair with a key size of 2048 bits and saves the private key and public key to separate files named 'private.pem' and 'public.pem', respectively. The private key is saved in PEM format, which is a text-based format for storing cryptographic keys, while the public key is saved in X.509 format, which is a standard format for public key certificates.

This script recursively scans a directory and encrypts each file using AES [2] encryption with a randomly generated session key that is then encrypted with an RSA public key. The encrypted session key, nonce, tag, and ciphertext are then saved to the original file. The original unencrypted file is removed.

It's important to note that running this script on your PC could potentially cause irreversible data loss since it will encrypt all files in the specified directory and remove the unencrypted versions. Then, a script proceeds to decrypt the encrypted files using a private key to decrypt the session key and then using it to decrypt the ciphertext. The decrypted data is saved to a new file with a decrypted text extension overwriting the original encrypted files. The whole process is repeated in an infinite loop until interrupted.

IV. Results and Discussion

A. Dataset Generation

With the help of OpenNebula[12], we have set up a private cloud. With the Libvirt API [13], these virtualized domains' resources are monitored. The purpose of Libvirt is to have a library that performs all required hypervisor management operations without implementing functionality that is designed for a particular virtualization solution and may not be of universal interest. The disc, network, and memory properties of the VM are connected to the monitored parameters. Table IV contains a list and a description of the parameters [1]. The datasets are periodically stored in the MySQL database.

B. Preprocessing

As all the features provided by the libvirt were cumulative, for the normalization process. Observing the resultant features, it's clear that features such as major Faults, minorFaults, memUsable, memlastUpdate, hda_read_requests, hda_read_bytes, hda_write_requets, and hda_write_bytes are not required as their values do not change overall, thus their values do not have a significant impact on deciding the final output. In some cases, it could also create bias. To reduce any dominant effect of any parameter, each of the feature parameters is normalized. Equational normalization transforms values to a common scale.

C. Feature Selection

Bivariate analysis that assesses the direction and degree of the relationship between two variables is called correlation. The value of the correlation coefficient varies between +1 and -1 depending on how strong the association is. The most popular correlation statistic for evaluating the strength of a link between variables that are linearly related is the Pearson r correlation. To determine the best parameters for efficient machine learning performance, we experimented with the Pearson correlation coefficient [10] and Information gain [11], gain ratio, and Gini index. Fig. 3 shows the correlation score for our dataset and Table I shows the ranking of the features which was used for feature selection.

Table I Ranking of the feature

Sr No	Parameter	Information Gain	Gain ratio	Gini
1	vda_write_requests	0.6924	0.3463	0.3451
2	cputime	0.6864	0.3432	0.3336
3	vda_write_bytes	0.6835	0.3419	0.3412
4	usertime	0.6692	0.3358	0.3284
5	systime	0.1169	0.0609	0.0719
6	rxbytes	0.0007	0.0003	0.0004
7	txbytes	0.0006	0.0003	0.0004
8	rxpackets	0.0004	0.0002	0.0002
9	txpackets	0.0003	0.0002	0.0002

D. Distributions

The one-dimensional standard Gaussian distribution is the most basic type of Gaussian distribution. The dataset's selected parameters all exhibit Gaussian distribution. Another method for finding abnormalities in the dataset is to use a box plot. The distribution and box plot of the top features that were chosen from the prior stage is shown in the figures below. It so happens that the distribution is gaussian in nature. The boxplots suggest that there are no outliers in the datasets. Fig. 1 and Fig. 2 show the box plot and distribution of one of the features cputime of the dataset.

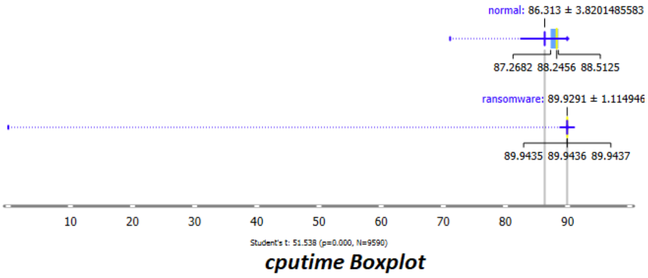


Fig. 1 Box plot: cputime

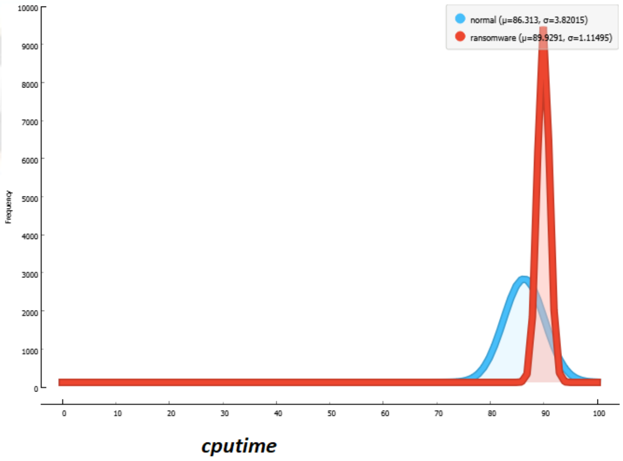


Fig. 2 Distribution of the dataset: cputime

We have plotted the graphs and boxplots for all the selected features from the feature selection stage and used them for further training.

E. Attack Detection

The Libvirt dataset contains 3110 records in total. We trained SVM, Naive Bayes, Random Forest, and KNN models using 10% of the fixed sampling technique. With rigorous empirical investigation, the parameter setting that produces higher performance is chosen. None of these classification models will be changed because of this research. The performance of the AI agent in the suggested system has improved because of the proposed changed feature set, which was utilized to train the models and revealed higher classification results.

With training data samples of 10, 20, 30, 40, 50, 60, and 70%, the suggested system is put to the test. The proposed agent was trained with SVM, Naive Bayes, Random Forest, and KNN models, with the KNN model (with Mahalanobis distance) providing the best classification accuracy in nearly all training iterations from 10% to 70%. The data show a concern with class imbalance. We have considered sensitivity and specificity to gauge performance through the evaluation findings due to the issue of class imbalance. The classification evaluation findings for the Libvirt dataset are shown in Table II.

Table II Evaluation Matrix for ransomware detection

Model	**AUC**	**CA**	**F1**	**Sensitivity**	**Specificity**
kNN_manhattan	0.992	0.992	0.992	0.992	0.992

kNN_mahalanobis	0.991	0.991	0.991	0.992	0.991
kNN_euclidean	0.992	0.992	0.992	0.992	0.992
SVM	0.998	0.993	0.993	0.993	0.993
Random Forest	0.997	0.996	0.996	0.996	0.996
Naive Bayes	0.996	0.971	0.971	0.973	0.971

For training the ransomware attack detection abovementioned models were trained with different hyperparameter tuning. The best results are considered here. The hyperparameter setting used for the experiments is mentioned in Table III.

Table III Hyperparameter tuning for ransomware detection.

Model	**Parameter setting for Ransomware Dataset**
Random Forest	Train: test split - 10%:90%, Number of features - 9, Number of trees - 5, Number of attributes considered at each split - 7, Limit depth of each tree - 4
Naive Bayes	Train: test split - 10%:90%, Number of features - 9, Classifier - Gaussian
SVM	Train: test split - 10%:90%, Number of features - 9, Kernel - RBF, exp(-0.35|x-y|2), Regression Loss epsilon (ε) - 0.10, Cost (C) - 0.50, Numerical tolerance - 0.0010
kNN_euclidean	Train: test split - 10%:90%, Number of features - 9, Number of neighbors - 3, Metric - Euclidean Weight - Uniform
kNN_manhattan	Train: test split - 10%:90%, Number of features - 9, Number of neighbors - 5, Metric - Manhattan, Weight - Uniform
kNN_mahalanobis	Train: test split - 10%:90%, Number of features - 9, Number of neighbors - 5, Metric - Mahalanobis, Weight - Uniform

V. Result Analysis

The dataset was generated for ransomware detection by running scrips ransomware scripts at the VM and collecting hypervisor-level activities. The dataset was analyzed for its distribution and outliers. The dataset is imbalanced, However, the real-time behavior of the ransomware attacks would generate fewer data samples for attack than normal. Hence we have not used any technique for the data imbalance problem. Feature selection was done on the hypervisor level activities. The features were used to train various ML models on the dataset for ransomware detection. Our novel approach to detecting ransomware attacks at the hypervisor level is giving detection accuracy of 99.6%.

Conclusion

The paper sheds light on the evolution of ransomware attacks in the cloud and the pressing need for robust detection solutions. The proposed system, employing machine learning models, demonstrates promising results in effectively detecting and mitigating ransomware attacks in virtualized environments. The research contributes to the field of cybersecurity by providing valuable insights into combating ransomware in the cloud era. Future work of the research could be extended to using AI agents to automate this process.

REFERENCES

[1] Purnaye, Prasad and Kulkarni, Vrushali. "BiSHM: Evidence detection and preservation model for cloud forensics" Open Computer Science, vol. 12, no. 1, 2022, pp. 154-170. https://doi.org/10.1515/comp-2022-0241

[2] Trautman, Lawrence J., and Peter C. Ormerod. "Wannacry, ransomware, and the emerging threat to corporations." Tenn. L. Rev. 86 (2018): 503.

[3] Liao, Kevin, et al. "Behind closed doors: measurement and analysis of CryptoLocker ransoms in Bitcoin." 2016 APWG symposium on electronic crime research (eCrime). IEEE, 2016.

[4] Okereafor, Kenneth, and Rania Djehaiche. "A Review of Application Challenges of Digital Forensics." International Journal of Simulation Systems Science and Technology 21.2 (2020): 35-1.

[5] Sajjan, Rajani S., and Vijay R. Ghorpade. "Ransomware attacks: Radical menace for cloud computing." 2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET). IEEE, 2017.

[6] Costandache, Mihai-Andrei, Marian-Stefan Mihalache, and Emil Simion. "New directions in the ransomware phenomenon." Cryptology ePrint Archive (2020).

[7] Abraham, Sherly, and InduShobha Chengalur-Smith. "An overview of social engineering malware: Trends, tactics, and implications." Technology in Society 32.3 (2010): 183-196.

[8] Muhtadi, Adib Fakhri, and Ahmad Almaarif. "Analysis of malware impact on network traffic using behavior-based detection technique." International Journal of Advances in Data and Information Systems 1.1 (2020): 17-25.

[9] Alsharabi, Naif, Mariam F. Alshammari, and Yasser Alharbi. "Analysis of Ransomware Using Reverse Engineering Techniques to Develop Effective Countermeasures." Journal of Advances in Information Technology 14.2 (2023).

[10] Benesty, J., Chen, J., Huang, Y., Cohen, I. (2009). Pearson Correlation Coefficient. In: Noise Reduction in Speech Processing. Springer Topics in Signal Processing, vol 2.

Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-00296-0_5

[11] Berrar, D., Dubitzky, W. (2013). Information Gain. In: Dubitzky, W., Wolkenhauer, O., Cho, KH., Yokota, H. (eds) Encyclopedia of Systems Biology. Springer, New York, NY. https://doi.org/10.1007/978-1-4419-9863-7_719

[12] Calle-Romero, Paúl E., et al. "Virtual Desktop Infrastructure (VDI) deployment using OpenNebula as a private cloud." Applied Technologies: First International Conference, ICAT 2019, Quito, Ecuador, December 3–5, 2019, Proceedings, Part I 1. Springer International Publishing, 2020.

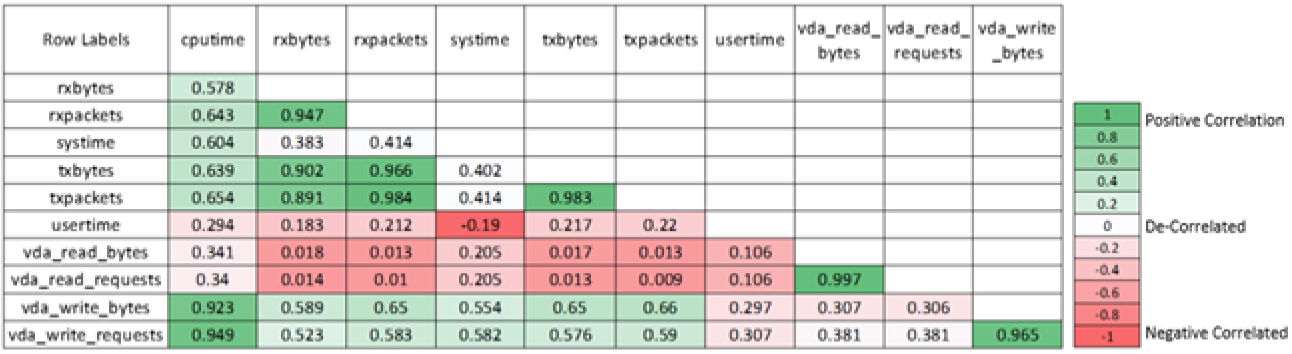
[13] Ashley, W. David. Foundations of Libvirt Development. New york: Apress, 2019

[14] Aslan, Ömer Aslan, and Refik Samet. "A comprehensive review on malware detection approaches." IEEE Access 8 (2020): 6249-6271.

[15] Baker, Matthew. "Transport and Encryption." Secure Web Application Development: A Hands-On Guide with Python and Django. Berkeley, CA: Apress, 2022. 59-93.

Table IV Dataset Description [1]

Sr No	Category	Feature	Description
1	Meta-data	LAST_POLL	Epoch timestamp
2		VMID	The ID of the VM
3		UUID	Unique identifier of the domain
4		Dom	Domain name
5	Network	Rxbytes	Received bytes from the network
6		rxpackets	Received packets from the network
7		txbytes	Transmitted bytes from the network
8		txpackets	Transmitted packets from the network
9	Memory	timecpu	Time spent by vCPU threads executing guest code
10		timesys	Time spent in kernel space
11		timeusr	Time spent in userspace
12		memmajor_fault	The number of page faults where disk IO was required
13		memminor_fault	The number of other page faults
14		memusable	The amount of memory that can be reclaimed by balloon without causing host swapping (in KiB)
15		memlast_update	The timestamp of the last update of statistics (in seconds)
16	Disk	vdard_req	Number of read-requests on the vda block device
17		vdard_bytes	Number of read-bytes on the vda block device
18		vdawr_reqs	Number of write requests on the vda block device
19		vdawr_bytes	Number of write requests on vda the block device
20		hdard_req	Number of read requests on the hda block device
21		hdard_bytes	Number of read bytes on the had block device
22		hdawr_reqs	Number of write requests on the hda block device
23		hdawr_bytes	Number of write bytes on the hda block device



Row Labels	cputime	rxbytes	rxpackets	systime	txbytes	txpackets	usertime	vda_read_bytes	vda_read_requests	vda_write_bytes
rxbytes	0.578									
rxpackets	0.643	0.947								
systime	0.604	0.383	0.414							
txbytes	0.639	0.902	0.966	0.402						
txpackets	0.654	0.891	0.984	0.414	0.983					
usertime	0.294	0.183	0.212	-0.19	0.217	0.22				
vda_read_bytes	0.341	0.018	0.013	0.205	0.017	0.013	0.106			
vda_read_requests	0.34	0.014	0.01	0.205	0.013	0.009	0.106	0.997		
vda_write_bytes	0.923	0.589	0.65	0.554	0.65	0.66	0.297	0.307	0.306	
vda_write_requests	0.949	0.523	0.583	0.582	0.576	0.59	0.307	0.381	0.381	0.965

Fig. 3 Pearson correlation coefficient