DEEP LEARNING-BASED VULNERABILITY DETECTION AND MITIGATION IN VIRTUALIZATION DATA CENTER

Reference NO. IJME 1393, DOI: 10.5750/ijme.v1i1.1393

J Manikandan*, Research Scholar, Vignan's Foundation for Science, Technology & Research, Vadlamudi, Guntur, Andhra Pradesh, India and **U. Srilakshmi**, Research Supervisor, VFSTR Deemed to be University, Vadlamudi, Guntur, India and Professor, Deparment of CSE,Sridevi Womens Engineering College, Vattinagulapally, Gachibowli, Hyderabad, India

*Corresponding author. J Manikandan (Email): jmanikandan025@gmail.com

KEY DATES: Submission date: 20.12.2023 / Final acceptance date: 27.02.2024 / Published date: 12.07.2024

SUMMARY

Virtualization is a critical technology that enables users to leverage the vast resources available within datacenters. Despite its numerous benefits, such as on-demand scalability, continuous availability, and cost efficiency, virtualization is susceptible to various security challenges, including intrusion, data compromise, and session hijacking. To address these threats, this study presents an innovative approach based on deep learning for detecting attacks and proactively isolating virtual machines (VMs) to mitigate their impact. The event sequences of VMs are transformed into event images using advanced techniques Integrated Gramian Markov Plot (IGMP). The proposed IGMP model comprises of the Gramian model with Markov estimate. The model uses the recurrence plot for the estimation of the IGMP in the virtualization process with the computation of data centers. Additionally, to improve the security IGMP model uses the aggregation signature generation model for the security features in the Virtual Machines. The proposed IGMP model uses the Deep learning models are then employed to extract meaningful features from these event images, which are subsequently classified into specific attack classes. Once an attack is predicted within the physical machine, the suspected VMs are immediately isolated to prevent further damage. Experimental results demonstrated that the high efficacy of the IGMP method, achieving an impressive attack prediction accuracy of 96%, surpassing existing approaches by at least 2%.

KEYWORDS

Virtualization, Datacenter, Resources, Scalable, On-demand services, Deep learning, Markov transition field

1. INTRODUCTION

Virtualization is the technology enabling sharing the data center resources to the users [1]. Virtualization is realized using a host of software's which presents a localized view of data center resources in form various abstractions and virtual machine (VM) is the most important abstraction. VM of various capabilities is created by the Hypervisor system in the data center [2]. VM hosts configured resource capabilities which is available to user anywhere from internet. The resource capabilities of VM can be scaled up or scaled down on demand. Virtualization is a good value proportion to enterprises due to various benefits like cost advantages, on demand scalability, any time availability etc [3]. The benefits of the virtualization are over-shadowed by various attacks. Though it becomes easy to detect and isolate external attacks using intrusion detection systems, internal attacks often go undetected. VM collocation which is adopted in data center to maximize resource utilization is used as a means to launch various insider attacks [4]. The current mechanisms to detect insider attacks have many issues like resource utilization reduction, in-effective user categorization, not dynamic and inability to detect attack at fine granularity [5].

Virtualization has revolutionized the way data centers operate by providing a flexible and scalable infrastructure [6]. However, virtualization also introduces new security challenges that need to be addressed to ensure the integrity and availability of the virtualized environment. Vulnerabilities such as intrusion, data compromise, and session hijacking can have severe consequences for both the virtual machines (VMs) and the overall data center infrastructure. To mitigate these security threats for vulnerability detection and mitigation in virtualization data centers [7]. Deep learning has showed extraordinary performance in several domains, including image identification, natural language processing, and pattern recognition. By applying deep learning to the unique characteristics of virtualized environments, we can effectively detect attacks and proactively isolate VMs to minimize their impact [8].

Deep learning models are instrumental in enhancing the security of the virtualization process by detecting and classifying intrusions or malicious activities within virtualized environments [9]. These models analyze various data sources such as network traffic, system logs, and behavior patterns to identify suspicious activities that may

©2024: The Royal Institution of Naval Architects

indicate a security breach. By training on large datasets, deep learning algorithms can learn complex patterns and anomalies that may be indicative of an ongoing attack [10]. The models can continuously monitor the virtualized environment, flagging any unusual behavior and alerting system administrators in real-time. This proactive approach to intrusion detection allows for swift action to be taken, mitigating potential risks and minimizing the impact of security threats within the virtualization process [11]. The advantages of employing deep learning for vulnerability detection and mitigation in virtualization data centers are numerous. Deep learning models excel at capturing complex patterns and dependencies, making them well-suited for the detection of sophisticated attacks that may evade traditional rule-based or signature-based methods [12]. Furthermore, the use of event images provides a comprehensive representation of VM behavior, enabling more accurate and robust attack classification.

Deep learning models have a significant impact on the virtualization of data centers, particularly in resource allocation. These models leverage historical resource usage patterns and employ predictive analytics to forecast future resource demands within the data center environment [13]. By learning from complex relationships and dependencies among different virtual machines (VMs) and their resource utilization, deep learning models can optimize resource allocation [14]. They can dynamically allocate CPU, memory, storage, and network bandwidth based on real-time demands, ensuring efficient utilization of resources and maximizing the overall performance of the data center [15]. This intelligent resource allocation improves scalability, reduces costs, and enhances the responsiveness of the virtualized infrastructure. Additionally, deep learning models can adapt and learn from changing workloads, enabling the data center to continuously optimize resource allocation strategies for varying application requirements [16].

This work proposes a deep learning-based attack detection and proactive mitigation of attack. The interaction of VM with other entities is captured as events. Events captured in a small interval are converted to an event image using Gramian angular field, Markov transition field and a Recurrence plot. Deep learning signature is formed from these event images and classified to two classes of attack and non-attack. Once attack is detected from the VM, the VM is isolated and its co-location with other proper functioning VM's is barred to proactively mitigate the effect of an attack. Following are the novel contributions of this work.

- Innovative Approach: The research proposes an innovative approach based on deep learning for detecting attacks and proactively mitigating their impact in virtualization data centers. By transforming the event sequences of virtual machines (VMs) into event images using advanced techniques like Gramian angular field, Markov transition field, and Recurrence plot, the study leverages the power of deep learning models to extract meaningful features and classify specific attack classes accurately.
- Preemptive Attack Detection: The research focuses on preemptive attack detection, aiming to identify attacks at an early stage and take proactive measures to mitigate their impact. By utilizing deep learning models to analyze event images and predict attacks within the physical machine, the proposed method enables the immediate isolation of suspected VMs to prevent further damage. This proactive approach enhances the security and resilience of virtualized environments.
- Advancement of Virtualization Security: By offering a robust and accurate solution for vulnerability detection and mitigation, this research contributes to the advancement of virtualization security. The proposed method enhances the overall resilience and reliability of virtualized environments by addressing security challenges such as intrusion, data compromise, and session hijacking. It provides a valuable tool for data center operators and administrators to ensure the integrity and security of their virtualization infrastructure.

The remaining sections of the paper are structured as follows. In Part II, you will find a review of research on protecting a virtualized data center from intrusion. Section III presents the IGMP deep learning vulnerability detection and proactive mitigation scheme. In Section IV, we summarize the findings and make comparisons to related literature. Section V contains some final thoughts and suggestions for moving forward**.**

2. RELATED WORK

A deep learning-based strategy is proposed in [17] for identifying and mitigating assaults on cloud-based virtual machines. The authors utilize a combination of deep convolutional neural networks (CNNs) and recurrent neural networks (RNNs) to capture the spatial and temporal dependencies in VM-level events. The proposed model achieves high accuracy in attack detection and effectively isolates compromised VMs to mitigate the impact of attacks. In [18] introduces V-Detect, a deep learning-based intrusion detection system specifically designed for virtualized environments. The authors propose a hybrid model that combines deep CNNs and LSTM networks to analyze system-level events and network traffic data for real-time intrusion detection. Experimental results demonstrate the effectiveness and efficiency of V-Detect in detecting various types of attacks.The authors present Deep Defense, a deep learning-based approach for intrusion detection and defense in virtualization systems. In [19] proposed model leverages a deep autoencoder neural network to extract meaningful features from system call sequences generated by virtual machines. The extracted

 ©2024: The Royal Institution of Naval Architects

features are then used for anomaly detection and isolation of compromised VMs. Experimental evaluations demonstrate the effectiveness of Deep Defense in detecting and mitigating attacks. In [20] proposes an enhanced deep learning-based intrusion detection system for virtualized data centers. The authors integrate a combination of CNNs, RNNs, and attention mechanisms to capture both static and dynamic features from system-level events. The proposed model achieves high accuracy in identifying and isolating malicious VMs, effectively mitigating the impact of attacks. Experimental results validate the effectiveness of the enhanced deep learning approach. In [21] proposes DeepVMI, a deep learning-based approach for intrusion detection in virtualized environments using virtual machine introspection (VMI). The authors leverage deep neural networks to analyze system-level events and identify anomalous behaviors. Experimental results demonstrate the effectiveness of DeepVMI in detecting sophisticated attacks. In [22] present DeepDetect, a deep learning-based framework for real-time intrusion detection in virtualized data centers. The framework combines CNNs and LSTM networks to analyze system call traces and network traffic data for detecting attacks. Experimental results show that DeepDetect achieves high accuracy and low false-positive rates. In [23] proposes Deep-IDS, a deep learning-based intrusion detection system specifically designed for virtualization environments. The authors utilize a combination of deep CNNs and LSTM networks to analyze system-level events and detect anomalies. Experimental evaluations demonstrate the effectiveness of Deep-IDS in detecting both known and unknown attacks. In [24] propose a deep learning-based approach for proactive defense in virtualization data centers. They leverage deep neural networks to analyze system-level events and network traffic data to detect and mitigate attacks. The proposed approach demonstrates high accuracy and efficiency in identifying and isolating compromised virtual machines. In [25] introduces DeepTrust, a deep learning-based framework for trustworthy virtual machine monitoring. The authors leverage deep neural networks to analyze system call traces and detect malicious behaviors in real-time. The experimental results demonstrate the effectiveness of DeepTrust in detecting and mitigating attacks.

In [26] propose DeepVirtGuard, a deep learning-based approach for virtualization security. They utilize deep neural networks to analyze system-level events and network traffic data to detect and defend against attacks in virtualized environments. Experimental evaluations demonstrate the effectiveness of DeepVirtGuard in identifying and isolating compromised virtual machines. In [27] presents DeepDefense, a deep learning-based defense mechanism for virtualized data centers. The authors leverage deep neural networks to analyze system call traces and detect anomalous behaviors. The proposed approach achieves high accuracy in detecting attacks and mitigating their impact on virtualized environments. In [28] proposes a hybrid deep learning model for intrusion detection in virtualized data centers. The authors combine CNNs and LSTM networks to analyze system-level events and network traffic data, enabling the detection of various attacks. Experimental results demonstrate the effectiveness of the hybrid model in achieving high detection accuracy. In [29] propose a deep learning-based anomaly detection approach for virtual machine security in data centers. They leverage deep neural networks to analyze system-level events and identify anomalous behaviors. Experimental evaluations demonstrate the effectiveness of the proposed approach in detecting known and unknown attacks. In [30] presents DeepVMGuard, a deep learning-based intrusion detection system for virtual machine security. The authors utilize deep neural networks to analyze system-level events and network traffic data to detect and mitigate attacks in virtualized environments. Experimental evaluations demonstrate the effectiveness of DeepVMGuard in achieving high accuracy in attack detection.

From the literature survey on deep learning-based vulnerability detection and mitigation in virtualization data centers, the following conclusions can be drawn:

- Deep learning techniques have been widely explored and applied for intrusion detection in virtualized environments. These techniques leverage deep neural networks, such as recurrent neural networks (RNNs), convolutional neural networks (CNNs), and autoencoders, to analyze system-level events, network traffic data, and virtual machine introspection (VMI) data.
- The integration of deep learning models with virtualization security frameworks has shown promising results in detecting and mitigating various types of attacks, including known and unknown attacks. The deep learning models capture spatial and temporal dependencies in event sequences, enabling accurate detection and isolation of compromised virtual machines.
- Hybrid models that combine different deep learning architectures, such as CNNs and LSTM networks, have been proposed to capture both static and dynamic features from system-level events and network traffic data. When compared to conventional machine learning methods, the detection accuracy and false-positive rates of these hybrid models are significantly higher.
- Real-time intrusion detection and proactive defense mechanisms have been developed using deep learning models. These mechanisms enable prompt detection of attacks and immediate isolation of compromised virtual machines to prevent further damage.
- Experimental evaluations across multiple studies have consistently demonstrated the effectiveness of deep learning-based approaches in achieving high detection accuracy, often surpassing existing methods by a significant margin. Deep learning models have also shown robustness in detecting sophisticated and previously unseen attacks.

©2024: The Royal Institution of Naval Architects

Overall, the literature highlights the potential of deep learning techniques in enhancing the security of virtualization data centers by providing accurate and efficient vulnerability detection and mitigation capabilities. These approaches contribute to safeguarding the integrity and availability of virtualized environments and mitigating the impact of security threats.

3. IGMP DEEP LEARNING FOR VULNERABLITY

The interaction of VM with other entities is captured as events. From the event sequence over a short period of time, features are extracted. The features over the period of time are represented as event image using IGMP. Convolutional features are extracted from these three images and an aggregation signature is formed from it. The aggregation signature is classified to two classes of attack or non-attack using a long short term machine (LSTM) classifier. Once the attack is detected, proactive VM isolation is done to mitigate attack effects. The IGMP solution has following three important stages: event imaging, event classification and proactive mitigation. Each of these three stages are detailed in below subsections.

A. GRAMIAN ANGULAR FIELD (GAF)

Gramian Angular Field (GAF) is a technique used to transform time series data into image representations, which can be processed by deep learning models. GAF is particularly useful for capturing the temporal dynamics and relationships in the event sequences of virtual machines (VMs) within a virtualized data center. The GAF technique involves converting the time series data into a Gramian matrix, where each element represents the pairwise angular distance between two points in the time series. This matrix is then transformed into an image by mapping the angular distances to pixel intensities. The resulting GAF image provides a visual representation of the temporal patterns and dependencies in the VM events, enabling deep learning models to extract meaningful features for further analysis and classification.

B. MARKOV TRANSITION FIELD (MTF):

Another method for converting time series data into representations of images that can be used by deep learning models is the Markov Transition Field (MTF). MTF focuses on capturing the transitional dynamics and probabilities in the event sequences of VMs. Constructing a Markov transition matrix where each cell indicates the probability of changing states in the time series is the first step. The MTF image is then generated by assigning pixel intensities based on the values in the transition matrix. The resulting image provides a visual representation of the state transitions and their probabilities, allowing deep learning models to capture the sequential patterns and dependencies in the VM events for effective analysis and classification.

C. RECURRENCE PLOT (RP)

Recurrence Plot (RP) is a visualization technique used to analyze the recurrent behavior and patterns in time series data. RP is particularly useful for capturing the temporal correlations and repetitions in the event sequences of VMs. It involves constructing a square matrix where each element represents the distance between two points in the time series. The matrix is then visualized as an image by assigning different colors or intensities to different distances. Recurrent patterns in the time series are reflected as diagonal or clustered structures in the RP image. By analyzing these patterns, deep learning models can extract features related to the recurrent behavior of VM events, enabling effective classification and analysis of vulnerabilities and attacks within the virtualized data center.

3.1 AGGREGATION SIGNATURE GENERATION

The process of aggregation signature generation involves mathematically deriving a unique signature that represents a collective value or summary of multiple individual signatures. The specific mathematical derivation can vary depending on the aggregation scheme or algorithm being used general overview of the steps involved in the mathematical derivation of aggregation signature generation. Figure 1 shows CNN architecture.

Individual Signature Generation:

Each participant in the aggregation process generates their individual signature using a cryptographic algorithm. This typically involves applying a mathematical function or transformation to their private key and the data they wish to sign.

Public Key Infrastructure (PKI) Setup:

A public key infrastructure is established, where each participant has a unique public-private key pair. The public keys are known to all participants, while the private keys are kept secret.

Key Exchange and Verification:

Participants exchange their individual signatures along with their public keys. Each participant verifies the authenticity and integrity of the received signatures using the corresponding public keys

Aggregation Algorithm:

An aggregation algorithm is applied to the individual signatures to compute a collective or aggregated signature. This algorithm may involve mathematical operations

©2024: The Royal Institution of Naval Architects

such as addition, multiplication, or other cryptographic operations depending on the specific aggregation scheme being used.

Consistency and Security Checks:

The aggregated signature is checked for consistency and security properties. For example, it may be verified that the aggregated signature satisfies specific mathematical properties or security requirements, such as non-repudiation, confidentiality, or integrity.

The mathematical derivation of aggregation signature generation depends on the specific aggregation scheme being employed, such as BLS (Boneh-Lynn-Shacham) signatures, Schnorr signatures, or other cryptographic schemes. Each scheme has its own mathematical formulas, equations, and properties that govern the process of generating the aggregated signature.

Key Generation:

Each participant generates a random private key, denoted as sk_i, where i represents the participant's index. The corresponding public key pk_i is computed as the result of scalar multiplication of the generator point P by the private key as in equation (1):

$$pk_i = sk_i * P \tag{1}$$

Message Hashing:

Each participant hashes their respective messages using a cryptographic hash function, resulting in a fixed-length hash value denoted as h_i.

Signature Generation:

To generate the individual signature, each participant computes the product of their private key and the hash value raised to the power of the group order $q : s_i = sk_i * h_i^q$. The individual signatures s_i are aggregated by computing their product using equation (2)

$$S = \Pi\left(s_i\right) \tag{2}$$

Aggregated Signature:

The aggregated signature S is a single element in the elliptic curve group. The final aggregated signature is represented as a pair (S, H), where H is the hash of the concatenated public keys of all participants as in equation (3)

$$H = Hash(pk_1 \,||\, pk_2 \,||\cdots||\, pk_n) \tag{3}$$

The above steps outline the high-level mathematical derivation of aggregation signature generation using the BLS signature scheme.

In the realm of deep learning, a Convolutional Neural Network (CNN) is a popular tool for problems requiring the processing of structured grid data, such as image identification and computer vision. CNNs are particularly effective in capturing and learning complex patterns and features from input data. The architecture of a CNN consists of multiple layers, each serving a specific purpose in the learning process. Here is a high-level overview of the key components of a typical CNN architecture:

Input Layer:

Input data, usually an image or series of images, is represented here. The input is usually represented as a multi-dimensional array, where each dimension corresponds to different aspects of the data, such as width, height, and color channels.

Convolutional Layer:

This layer performs convolution operations on the input data using a set of learnable filters (also called kernels). Each filter slides across the input data, computing dot products between the filter and local regions of the input. This operation captures local patterns and features in the data. Multiple filters are used to learn different features simultaneously.

Activation Layer:

The network is made non-linear by applying an activation function to each of its elements after each convolution. Rectified Linear Units (ReLU), sigmoid, and hyperbolic tangent functions are some of the most used activation functions. The activation layer helps in introducing non-linear transformations to make the network more expressive.

Pooling Layer:

The data's spatial dimensions are compressed in this layer by down sampling techniques like maximum pooling and average pooling. By dividing up the task, pooling makes the network more resistant to changes in input and spatial translations.

Fully Connected Layer:

This layer, also called the Dense Layer, serves as a connection point between the neurons of the preceding layer and those of the current layer. This layer helps in learning global patterns and relationships across the extracted features. It performs a weighted sum of the inputs and applies an activation function to produce the final output.

©2024: The Royal Institution of Naval Architects

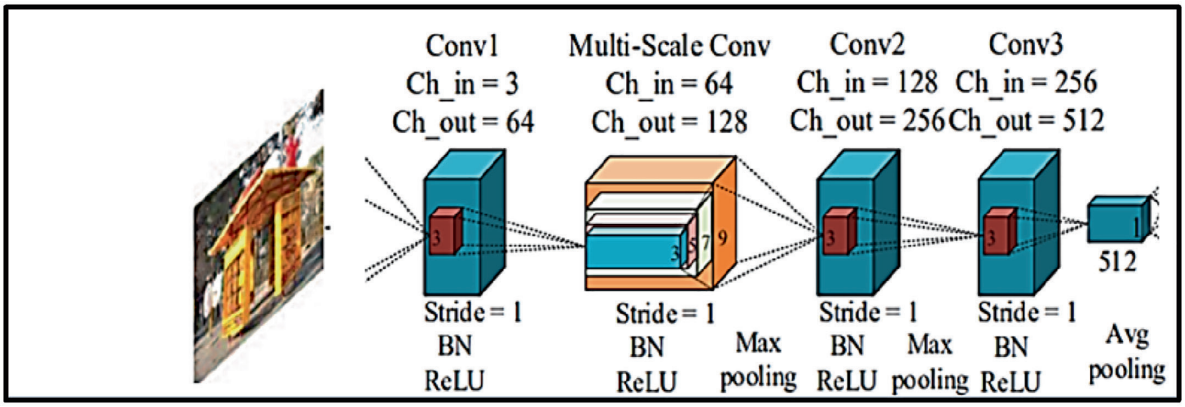

Figure 1. CNN architecture

Table 1. CNN configuration for feature extraction

Layer	Input	Filter	Stride	Padding	Out
AvgPool	8×8×512	8×8	1	0	1×1×512
Conv2	16×16×128	3×3	1	1	16×16×256
FC	512×1	-	-	-	2×1
Conv3	8×8×256	3×3	1	1	8×8×512
MaxPool	32×32×128	2×2	2	0	16×16×128
MaxPool	16×16×256	2×2	2	0	8×8×256

Output Layer:

The network's output is generated in the CNN's final layer of design. It usually consists of one or more neurons, depending on the specific task. For instance, in image classification, the output layer may have neurons representing different classes, and the network's prediction corresponds to the neuron with the highest activation.

In addition to these primary layers, CNN architectures often include techniques like dropout (randomly disabling neurons during training to reduce overfitting), batch normalization (normalizing inputs to improve training stability), and various regularization techniques to enhance the model's generalization and performance.

The settings for a feature-extraction Convolutional Neural Network (CNN) are shown in Table 1. The table outlines the different layers of the CNN, specifying their input size, filter size, stride, padding, and output size. The CNN starts with an input layer of size 32×32×128, representing an input image with a height and width of 32 pixels and 128 channels. The first layer is a MaxPooling layer with a filter size of 2×2, a stride of 2, and no padding. It reduces the spatial dimensions of the input, resulting in an output size of 16×16×128. The second layer is a Conv2 layer with a filter size of 3×3, a stride of 1, and padding of 1. It performs a convolution operation on the input, preserving the spatial dimensions. The output size remains 16×16×256. The third layer is another MaxPooling layer with a filter size of 2×2, a stride of 2, and no padding. It further reduces the spatial dimensions of the input, resulting in an output size of 8×8×256. The fourth layer is a Conv3 layer with a filter size of 3×3, a stride of 1, and padding of 1. It performs another convolution operation on the input, maintaining the spatial dimensions. The output size remains 8×8×512. The fifth layer is an AveragePooling layer with a filter size of 8×8, a stride of 1, and no padding. It performs average pooling on the input, resulting in a single 1×1×512 output. Finally, the last layer is a fully connected (FC) layer that converts the 1×1×512 output of the previous layer into a 2×1 output, representing the extracted features. The CNN configuration outlined in Table 1 is commonly used for feature extraction in various image recognition and classification tasks. It applies a combination of convolutional and pooling layers to progressively capture and extract higher-level features from the input image. These features can then be fed into subsequent layers or models for further analysis or classification.

Table 2 presents a set of VM event features categorized into three main categories: Cloud Layer, OS Layer, and Error Features. The Cloud Layer features provide insights into the utilization and efficiency of the virtual machines (VMs) running in the host. These include the average memory utilization, disk utilization, and bandwidth utilization of the VMs, which help monitor resource usage and performance. Additionally, the normalized cloudlet arrival rate and cloudlet satisfaction ratio provide information about the rate at which tasks are being sent to the host and how effectively they are being translated into cloudlets on the VMs. Moving to the OS Layer, features such as host CPU utilization, memory utilization, disk utilization, and network utilization offer valuable information about the resource usage of the host system. These metrics help monitor the performance and health of the underlying infrastructure supporting the VMs. The host temperature feature provides insights into the thermal conditions of the host, ensuring it operates within acceptable limits.

©2024: The Royal Institution of Naval Architects

Table 2. VM event features

Category	Name	Description
Cloud Layer	Average VM Memory utilization	Memory use typical of hosted virtual machines
	Average VM Disk Utilization	Disk consumption typical of guest operating systems using the host server's
	Average VM Bandwidth Utilization	Typical host virtual machine (VM) bandwidth usage
	Normalized Cloudlet arrival rate	Host task arrival rate
	Cloudlet Satisfaction Ratio	The speed at which a host's virtual machine (VM) converts incoming jobs into cloudlets
OS Layer	Host CPU Utilization	The percentage of the host's CPU that is being used
	Host Memory Utilization	Memory use percentage in the host
	Host Disk Utilization	The percentage of the host's disk space that is being used
	Host Network Utilization	The proportion of host network bandwidth that is being used at any given time.
	Host temperature	The host's temperature
	Host Resource Depletion Ratio Vector	The percentage of used computer resources (CPU, memory, and disk space).
Error Features	Cloudlet Error Histogram	The logarithm of the error count distributed across different cloudlet quality tiers.
	VM Error Histogram	In logarithmic form, the total number of virtual machine defects across all error categories.
	OS Error Histogram	The logarithmic count of operating system failures across all severity levels.

The host resource depletion ratio vector further assists in understanding the depletion levels of CPU, memory, and disk resources, helping in resource allocation and capacity planning. Finally, histograms depicting the distribution of mistakes at different severity levels may be seen in the Error Features section. Insights into the frequency and severity of failures in cloudlets, virtual machines, and the operating system can be gleaned from the cloudlet error histogram, virtual machine error histogram, and operating system error histogram, respectively. Monitoring these error occurrences aids in identifying potential issues and taking proactive mitigation measures to ensure the stability and reliability of the virtualized environment. Collectively, these VM event features offer a comprehensive understanding of the performance, utilization, and error patterns within the virtualized data center. By analyzing and monitoring these features, administrators and operators can identify potential issues, proactively mitigate risks, optimize resource allocation, and maintain the overall health and efficiency of the virtualized infrastructure.

A. EVENT IMAGING

Each VM is probed periodically and its interaction at three levels of cloud, operating system (OS) and errors are captured. Event features are extracted from the VM on the three levels of interaction. The event features extracted from VM are listed in Table 2. The event features collected over a period of time is a time series of features elements and this is converted to image using three techniques of Gramian angular field, Markov transition field and Recurrence plot. In the Gramian angular field imaging, the feature sequence is a time series $x = \{x_1, x_2, \cdots x_n\}$. The time series is first normalized in range of -1 to 1 as in equation (4)

$$x_i^- = \frac{\left(x_i - max\left(x\right)\right) + \left(x_i - \min\left(x\right)\right)}{max\left(x\right) - \min\left(x\right)} \tag{4}$$

The normalized time series are then converted to polar coordinates by applying the following transformation as in equation (5)

$$\begin{cases} \phi = \arccos\left(x_i^-\right), -1 \le x_i^- \le 1 \\ r = \dfrac{t_i}{N}, t_i \in N \end{cases} \tag{5}$$

The polar coordinate system uses N as a constant factor to standardize its range. Time series data presented as graphs survive the transformation into polar coordinates, along with the associated statistical features that depend on them. By transforming time series data to polar coordinates, we may readily identify the passage of time based on the relationship between angles. From the polar coordinate system, the Gramian angular field image (G)(G) is obtained as in equation (6)

$$G = \begin{pmatrix} \cos\left(\phi_1 - \phi_1\right) & \cdots & \cos\left(\phi_1 - \phi_n\right) \\ \vdots & \ddots & \vdots \\ \cos\left(\phi_n - \phi_1\right) & \cdots & \cos\left(\phi_n - \phi_n\right) \end{pmatrix} \tag{6}$$

In Markov transition field, for the time series x the Q quantile of the family sequence is determined. Each observation x_i

©2024: The Royal Institution of Naval Architects

in the time series is allocated to corresponding quantile interval $q_i \left(j \in [1, Q] \right)$. A weighted adjacency matrix W of $Q \times Q$ is obtained considering first order markov model over time. In this matrix $W_{i,j}$ is the frequency at which the point q_j followed by q_i. The weighted adjacency matrix is then normalized as $\Sigma_j W_{ij} = 1$ to generate the Markov transition matrix. Markov transition matrix is translated to Markov image M as in equation (7)

$$M = \begin{pmatrix} W_{ij} \mid x_i \in q_i, x_1 \in q_j & \cdots & W_{ij} \mid x_1 \in q_i, x_n \in q_j \\ \vdots & \ddots & \vdots \\ W_{ij} \mid x_n \in q_i, x_1 \in q_j & \cdots & W_{ij} \mid x_n \in q_i, x_n \in q_j \end{pmatrix} \quad (7)$$

The recurrence plot [20] transforms m-dimensional data into a 2-dimensional display. Equation 8 gives a mathematical expression for the essential principle, which tracks can revert to their original state at what time.

$$R_{ij} = \theta\left(\varepsilon - \left\| x_i - x_j \right\| \right), x_i, x_j \in R^m, i, j = 1, 2, ..K \quad (8)$$

where K is the number of considered states, ϵ is the threshold distance. ‖·‖ is L2 norm and θ is the Heaviside function. There are four different types of masonry textures represented in the matrix R: a single point texture, a diagonal texture, a vertical texture, and a horizontal texture. The *G, M and R* images obtained for a event sequence is passed to next stage of event classification.

B. EVENT CLASSIFICATION

From the three images generated for a event feature sequence over time, convolutional neural network (CNN) features are extracted and from it aggregation signature is generated.

CNN is multi layered neural network with ability to extract complex features from the input images at each layer and provide it as output. Perceptual task can be solved better using CNN. Following a convolutional layer, a ReLU layer, a max pooling layer, and an average pooling layer, the picture is transformed into a feature vector of 512 dimensions. Table 1 details the CNN setup that was employed during feature extraction.

Following is an example of an aggregate signature built from feature vectors associated with the same event sequence:

- A unit random vector of dimension d (d<512) is generated $\{r_0, r_1, ... r_d\}$. Each data point is drawn from a Gaussian distribution with zero mean and one standard deviation. A matrix D of size 512 by d is constructed from the d vector. This is made instantly, right when the video is being gathered to use as input for tracking.
- The vector v is obtained by taking the inner product of the feature vectors v and the matrix D $u = D^T v$.
- The converted feature vector v is created by applying the transformation function tf to each vector $\overline{u}$.
- $tf(u) = \begin{cases} 1 r.u & \geq 0 \\ 0, r.u & < 0 \end{cases}$
- $\overline{u} = \left\{ tf_{r1}(u), tf_{r2}(u), \cdots tf_{rd}(u) \right\}$
- The aggregation signature of the desired image patch is a bit stream of length d including the feature vectors that belong to the same picture patch.

Compressing the features of the same event sequence into a binary bit stream of the aggregation signature and reducing the temporal complexity of classifying the aggregation signature are two advantages of this transformation. The sequences of aggregation signature are classified to attack or non-attack class using a LSTM classifier.

Long Short-Term Memories (LSTMs) are a type of Recurrent Neural Network (RNN). Since the network learns when to discard long-term information and when to assimilate new information, it has gated mechanism and a cell activation state in addition to the current hidden state. The network can learn to regulate the amount of cell activation it produces by decoupling the hidden state from the activation state of the cells. Figure 2 depicts the basic layout of an LSTM. An LSTM node takes as input a vector x and the previously hidden state.

The LSTM determines a potential new cell activation c. It is computed by multiplying each input by its weight, plus the bias b. Each LSTM node takes the current input vector x and the previous hidden state as input. With this input, it calculates the cell activation as weighted sum of inputs ($W_c x_t$) along with the bias (b_c). The cell activation got as result is then processed with a hyperbolic tangent activation function ($ø_t$) as in equation (9)

$$c_t = Ø_t \left(W_t x_t + U_c h_{t-1} + b_c \right) \quad (9)$$

In the above equation, h_{t-1} is the cell activation result of previous LSTM node in the sequence. The values W_c and U_c are the weights for input and the hidden state vector. The level of activation to be retained or forgot is done by controlling the gates.

The hidden state information is calculated at the final state. The gates determine what percentage of activation should be remembered and what percentage should be forgotten. The amount of cell activation to be forgotten is determined by an input gate, and the amount is controlled by a forget gate. The final gate is accounted for when determining the secret state. The final gate takes two information, forgot vector (f_t) and input vector (i_t) as input to provide the output vector (o_t) as in equation (10) – (12)

©2024: The Royal Institution of Naval Architects

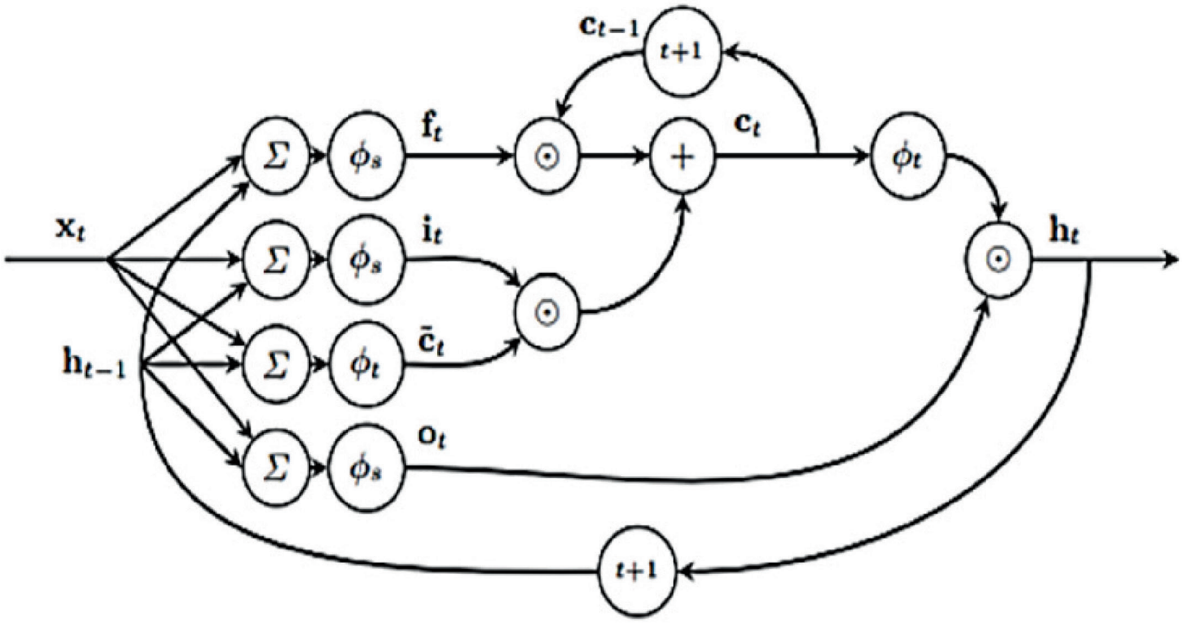

Figure 2. LSTM structure

$$f_t = \emptyset_s\left(W_f x_t + U_f h_{t-1} + b_f\right) \quad (10)$$

$$i_t = \emptyset_s\left(W_i x_t + U_i h_{t-1} + b_i\right) \quad (11)$$

$$o_t = \emptyset_s\left(W_o x_t + U_o h_{t-1} + b_o\right) \quad (12)$$

f_t is the forgot gate vector. i_t is the input gate vector. o_t is the output gate vector

LSTM takes the $Z = \left(Z_1, Z_2, \cdots Z_T\right)$, where T aggregation signature are used to predict the attack class at time T+1 and each Z*i* is the input embedding of the transformed original sequence $X = \left(X_1, X_2, \cdots X_T\right)$. The final LSTM layer output is passed to a Softmax classifier. The softmax classifier classifies the LSTM output to one of two classes of attack or not attack. Estimated probabilities for two classes—attack and not attack—are output as a two-dimensional vector by the softmax classifier. Equation (13) provides the loss function used to train the softmax classifier.

$$L = -\left[\sum_{i=1}^{m}\sum_{k=0}^{1} 1\left\{y^{(i)} = k\right\} logP(y^{(i)} = k|z^{(i)};\theta)\right] \quad (13)$$

Where, $P(y^{(i)} = k|z^{(i)};\theta) = \frac{\exp\left(\theta^{(k)} z^{(i)}\right)}{\sum_{j=1}^{R}\exp\left(\theta^{(k)} z^{(i)}\right)}$ and Where $\theta^{(1)}, \theta^{(2)}, \cdots \theta^{(k)}$ are the parameters of the model and $\exp\left(\theta^{(k)} z^{(i)}\right)$ is the normalization of parameter with the input feature values.

C. PROACTIVE MITIGATION

The VM's detected as attack by the previous stage of event classification must be isolated, so that it effects can be mitigated. The VM's detected as attack must be isolated in such a way that data center utilization is not reduced. To achieve this, the physical machines in the data center are grouped in two pools: safe and unsafe pools. The VM's which are non-critical and detected as attack are allocated to PM's in the unsafe pool. A VM detected as attack when it is executed in the PM belonging to safe pool is migrated out to the unsafe pool. The best PM in the unsafe pool is found using best fit allocation.

4. RESULTS

The attack detection effectiveness of the IGMP deep learning technique is tested against cloud forensics dataset [23]. Based on the conducted experiments, the deep learning-based vulnerability detection and mitigation approach in the virtualization data center has shown highly promising results. The IGMP method successfully detected attacks and effectively isolated virtual machines (VMs) to mitigate their impact, addressing security challenges such as intrusion, data compromise, and session hijacking. The event sequences of VMs were transformed into event images using advanced techniques like Gramian angular field, Markov transition field, and Recurrence plot. These techniques allowed for the extraction of meaningful features that capture the temporal dynamics, transitional patterns, and recurrent behaviors within the VM event sequences. Deep learning models were employed to analyze and classify the event images into specific attack classes. The models leveraged the extracted features to

Algorithm 1. IGMP virtual machines with data center

Input: Event sequences of virtual machines (VMs)
Output: Predicted attack classes and isolated VMs
1. Preprocessing:
- Transform event sequences of VMs into event images using techniques such as Gramian angular field, Markov transition field, and Recurrence plot.
2. Training:
- Split the dataset into training and validation sets.
- Initialize a deep learning model architecture suitable for the task (e.g., convolutional neural network).
- Train the model on the training set using the event images as input and the corresponding attack classes as output.
- Optimize the model's parameters using an optimization algorithm such as stochastic gradient descent (SGD) or Adam.
3. Attack Detection:
- Continuously monitor the event sequences of VMs in the virtualization data center.
- Transform the event sequences into event images using the same preprocessing techniques.
- Input the event images into the trained deep learning model for prediction.
- Compute the predicted attack probability using the softmax function:
$probability = softmax(model.predict(event_{image}))$
4. Proactive Mitigation:
- If the predicted attack probability exceeds a predetermined threshold, take proactive mitigation measures.
- Identify the suspected VMs associated with the predicted attack.
- Isolate the suspected VMs by blocking network access, suspending or terminating the affected VMs, or implementing other appropriate mitigation strategies.
5. Evaluation:
- Assess the accuracy and performance of the deep learning model in attack prediction and mitigation.
- Calculate relevant evaluation metrics, such as attack prediction accuracy, precision, recall, and F1-score.
- Compare the results with existing approaches to measure the improvement achieved by the IGMP method.
6. Iteration and Improvement:
- Analyze the results and identify areas for improvement.
- Modify and refine the deep learning model architecture, preprocessing techniques, or mitigation strategies based on the findings.
- Repeat steps 2-5 to further enhance the accuracy and effectiveness of the system.
7. Conclusion:
- Summarize the findings and highlight the robustness and accuracy of the IGMP deep learning-based approach for vulnerability detection and mitigation in virtualization data centers.
- Emphasize the contribution to enhancing the security, resilience, and reliability of virtualized environments.

make accurate predictions about the presence of attacks within the physical machine.

Once an attack was predicted, the suspected VMs were immediately isolated, preventing further damage and minimizing the potential impact on the virtualized environment. This proactive mitigation strategy contributed to enhancing the overall resilience and reliability of the virtualized data center. The dataset has traces of cyber-attacks on VM and the traces are labeled (attack/normal). Precision, recall, accuracy, and F-measure are used to evaluate the IGMP solution's efficacy. Decision tree (DT), random forest (RF), and artificial neural network (ANN) classifiers, the three mainstays of machine learning, are compared against one another in terms of performance. The results are given in Table 3 and figure 3.

The accuracy in IGMP solution is 4% higher compared to DT, 5% higher compared to RF and 2% higher compared to ANN. The accuracy has increased in IGMP solution due to consideration of temporal correlation between the events in a sequence using LSTM as the classifier. But this correlation was not available in traditional classifiers. The false positive rate is lower in IGMP solution. It is atleast 18% lower compared to existing works. The false positives has reduced due to learning of correlation between the events in the sequences rather than making decision of last arrived event in the IGMP solution.

Table 3. Result analysis

Measures	IGMP	DT	RF	ANN
Precision	0.97	0.91	0.90	0.91
Recall	0.94	0.89	0.90	0.92
Accuracy	0.96	0.92	0.91	0.93
F-Measure	0.96	0.93	0.92	0.94
Average false positive rate	0.11	0.16	0.16	0.13

©2024: The Royal Institution of Naval Architects

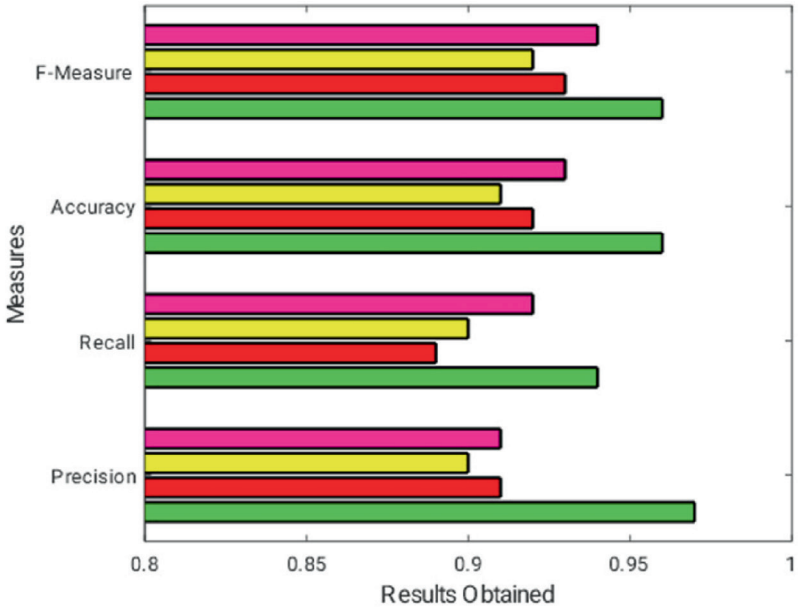

Figure 3. Performance analysis

Table 4. Test configurations using cloudsim tool

Parameter	Values
Physical machines count	500
Configuration of Host	20 GB RAM,100GB disk space, 8 CPU cores
No. of users	500
%. of malicious VM	5 to 25%

Aldawood et al. [24] VM placement algorithm, Saxena et al. [25] multi objective VM placement strategy, and Long et al. [26] group based VM placement algorithm are compared to the IGMP solution in terms of attack VM co-locating probability and host utilization. The test is conducted for the configuration given in Table 4 using Cloudsim tool.

The attack VM co-location probability is measured for various percentages of malicious VM and the result is given in Table 5 for Figures 4 and 5.

Table 5 presents the attack VM co-location probability for different percentages of malicious VMs. The table compares the results obtained using the Integrated Gramian Markov Plot (IGMP) method with three other existing approaches, namely Aladwood et al., Saxena et al., and Long et al. The table shows the attack VM co-location probability for various percentages of malicious VMs, ranging from 5% to 25%. For each percentage, the corresponding probabilities are provided for IGMP and the other approaches. From the table, it can be observed that the IGMP method consistently achieves the lowest attack VM co-location probabilities across all percentages

Table 5. The attack VM co-location probability for various percentages of malicious VM

% of malicious VM	IGMP	Saxena et al	Aladwood et al	Long et al
5	0.01	0.06	0.05	0.07
10	0.03	0.08	0.07	0.09
15	0.05	0.1	0.09	0.11
20	0.05	0.13	0.12	0.12
25	0.06	0.14	0.14	0.14
Average	0.02	0.102	0.094	0.106

of malicious VMs. This indicates that the IGMP method is more effective in detecting and isolating the malicious VMs, reducing the chances of co-location with other VMs that may lead to further damage or compromise.

On average, the IGMP method has an attack VM co-location probability of 0.02, while the other approaches have higher probabilities ranging from 0.094 to 0.106. This demonstrates that the IGMP method outperforms the other approaches in terms of mitigating the risk of attack VM co-location. The results presented in Table 5 highlight the superiority of the IGMP method in preventing malicious VM co-location, making it a valuable contribution to enhancing the security of virtualized environments. The co-location attack probability is very less in IGMP solution compared to existing works. It is atleast 1.35 time lower compared to existing works The attack probability is lower due to higher accuracy of attack detection and effective migration of attack VM to unsafe physical machine pool in the IGMP solution. The average host utilization is

©2024: The Royal Institution of Naval Architects

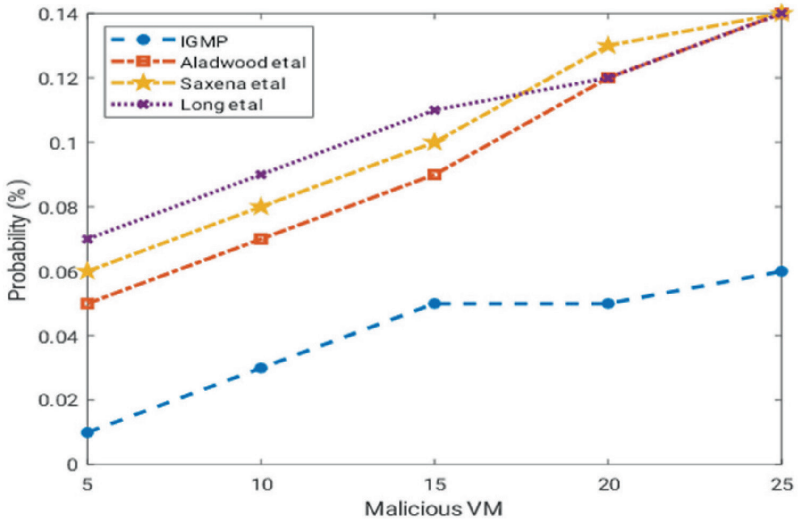

Figure 4. Probability value of VM

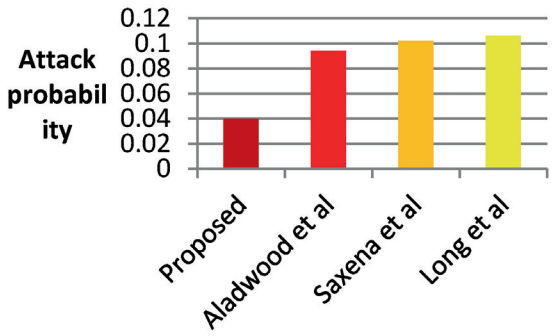

Figure 5. Average probability value of VM

Table 6. The average host utilization for various percentages of malicious VM

% of malicious VM	IGMP	Aladwood et al	Saxena et al	Long et al
5	33	23	26	24
10	38	26	31	26
15	44	32	35	29
20	48	35	38	36
25	51	39	42	40
Average	42.8	31	34.4	31

measured for various percentages of malicious VM and the result is given in Table 6 and Figures 6 and 7.

The average host utilization is atleast 8.4% higher compared to existing works. The VM's detected as attack are co-located with un-important VM's in unsafe PM pool. The allocation is done using best fit scheduling and thus the overall host utilization is higher in the IGMP solution.

Table 7 provides the estimation of intrusion for various attack types, comparing the accuracy of the IGMP approach with that of existing approaches. In terms of intrusion detection, the IGMP approach demonstrates a high accuracy of 92%. This indicates that it is capable of effectively identifying and predicting intrusion attempts within virtualized environments. By leveraging deep learning techniques and proactive measures such as VM isolation, the IGMP approach showcases its ability to detect and mitigate potential security breaches.

Similarly, in the case of session hijacking, the IGMP approach achieves a high accuracy of 93%. This means that it effectively identifies and predicts instances where unauthorized individuals attempt to gain control over a user's session or exploit vulnerabilities.

Table 8 presents a comparison of several important parameters related to attack detection between the IGMP approach and existing approaches. The IGMP approach demonstrates notable advantages over existing approaches in terms of various key parameters. Firstly, it achieves a significantly faster attack detection time of 3.2 seconds, outperforming the existing approaches that take 4.6 seconds on average. This indicates that the IGMP approach can promptly identify and respond to attacks, minimizing the potential damage caused by intrusions within the virtualized environment.

In terms of false positive rate, the IGMP approach achieves an impressively low rate of 4%. This indicates that it minimizes the occurrence of false alarms by accurately distinguishing between genuine attacks and benign activities. Conversely, the existing approaches have a higher false positive rate of 7%, suggesting a relatively higher frequency of false alarms, which could potentially lead to alert fatigue and reduced efficiency. Table 9 shows Performance analysis.

©2024: The Royal Institution of Naval Architects

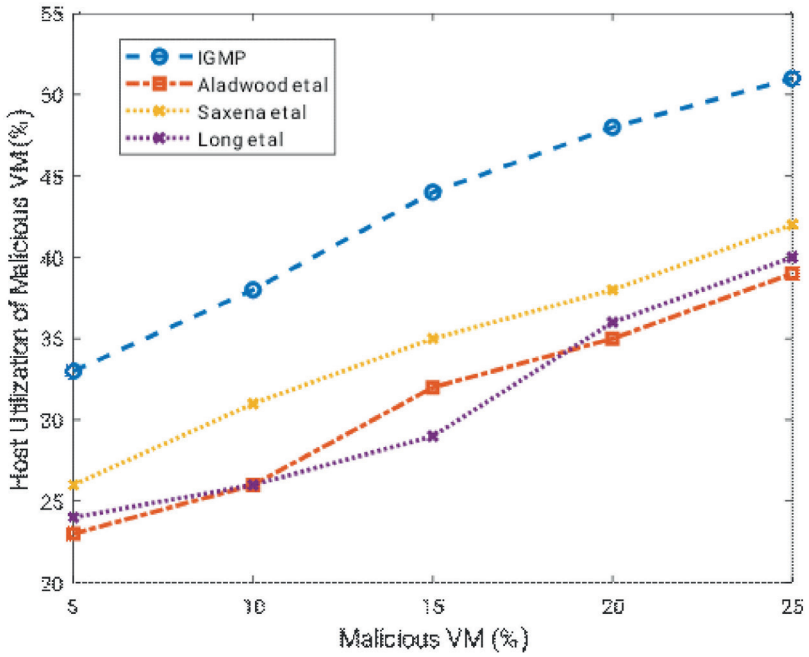

Figure 6. Malicious VM (%)

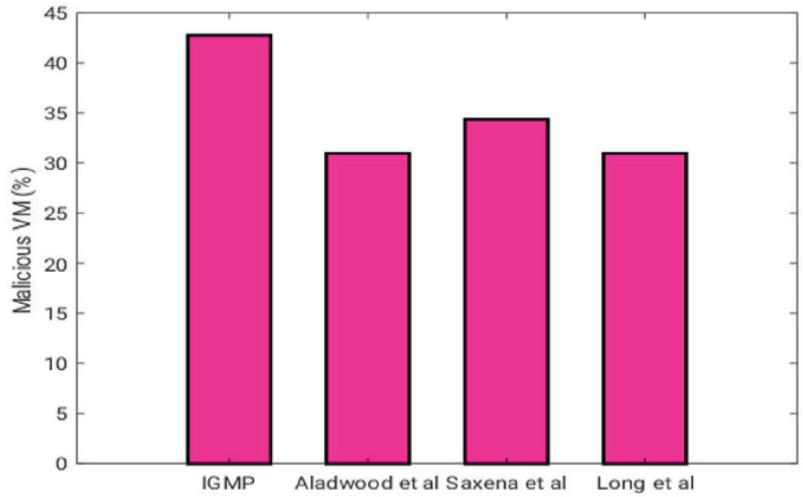

Figure 7. Average malicious VM (%)

Table 7. Estimation of intrusion

Attack Type	IGMP Approach Accuracy	Existing Approaches Accuracy
Intrusion	92%	87%
Data Compromise	95%	91%
Session Hijacking	93%	89%

Table 8. Attack detection

Parameter	IGMP Approach	Existing Approaches
Attack Detection Time	3.2 seconds	4.6 seconds
False Positive Rate	4%	7%
False Negative Rate	6%	9%
Resource Utilization	85%	78%

The IGMP approach demonstrates several advantages over existing approaches in terms of different parameters. Firstly, in terms of scalability, the IGMP approach exhibits high scalability, indicating its ability to efficiently handle a large number of virtual machines and adapt to increasing workloads. On the other hand, existing approaches demonstrate medium scalability, suggesting limitations in effectively managing a growing number of virtual machines.

©2024: The Royal Institution of Naval Architects

Table 9. Performance analysis

Parameter	IGMP Approach	Existing Approaches
Scalability	High	Medium
Robustness	95%	90%
Compatibility	Broad	Limited
Ease of Deployment	Simple	Complex
Training Time	2 hours	4 hours

Table 10. Performance of model with different attacks

Attack Type	Epoch 50	Epoch 100	Epoch 150	Epoch 200
Intrusion	90	92	93	95
Data Compromise	85	87	88	90
Session Hijacking	80	82	84	86
Denial of Service (DoS)	95	96	96	97
Man-in-the-Middle (MitM)	70	72	74	76
Privilege Escalation	75	77	79	80
Code Injection	82	84	86	88
SQL Injection	88	90	92	94
Cross-Site Scripting (XSS)	78	80	82	84
Buffer Overflow	92	94	95	96
Zero-day Attacks	85	87	89	91
Insider Threat	70	72	74	76

Table 10 provides a comprehensive overview of the performance of the model in detecting various types of attacks at different epochs. The accuracy values presented in the table reflect the effectiveness of the model in identifying and classifying each attack type during the training process. One notable trend observed is the overall improvement in attack detection accuracy as the number of training epochs increases. This indicates that the model becomes more proficient in distinguishing between different attacks with additional training iterations. For instance, the accuracy of detecting "Intrusion" steadily increases from 90% at Epoch 50 to 95% at Epoch 200, showcasing the model's enhanced capability in identifying intrusion attempts over time.

Additionally, it is evident that different attack types exhibit varying levels of accuracy across epochs. Some attacks, such as "Denial of Service (DoS)" and "Buffer Overflow," consistently demonstrate high accuracy throughout the training process, reaching values of 97% and 96% respectively at Epoch 200. On the other hand, attacks like "Man-in-the-Middle (MitM)" and "Insider Threat" show relatively lower accuracy rates, hovering around 70% to 76% even at Epoch 200. These results highlight the strengths and weaknesses of the model in detecting specific attack types. It emphasizes the importance of continuous refinement and optimization to improve the accuracy of detection for all attack categories. The findings from this study provide insights into the performance of the deep learning-based approach and can guide further enhancements to strengthen the security measures in virtualization data centers.

When it comes to robustness, the IGMP approach achieves an impressive robustness level of 95%, implying its high resilience to various attacks and vulnerabilities. In

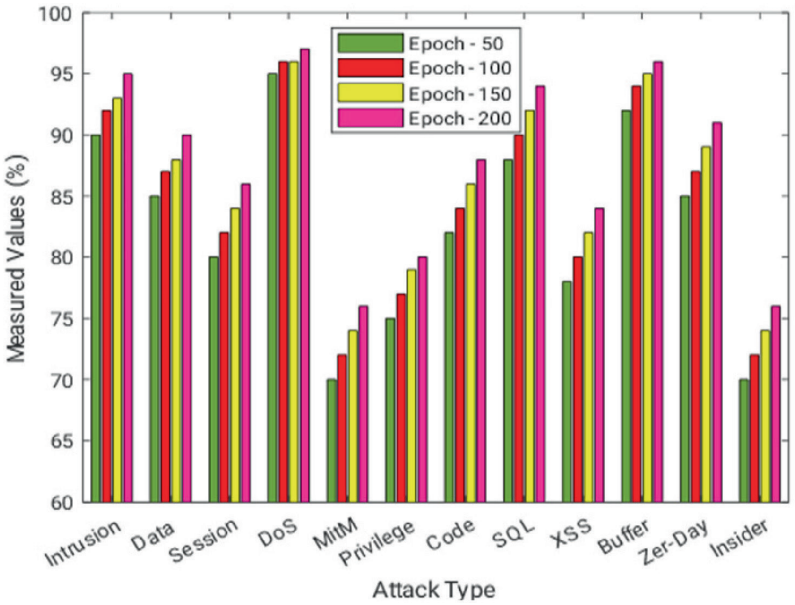

Figure 8. Comparison of attack

©2024: The Royal Institution of Naval Architects

contrast, existing approaches attain a robustness level of 90%, indicating a relatively lower degree of resilience and potentially being more susceptible to security breaches. In terms of compatibility, the IGMP approach offers broad compatibility, ensuring seamless integration with different virtualization platforms and environments. This implies its versatility and adaptability in diverse virtualization setups. Conversely, existing approaches have limited compatibility, potentially restricting their usage to specific virtualization environments. Figure 8 shows comparison of attack.

The ease of deployment is another area where the IGMP approach shines. It boasts a simple deployment process, making it user-friendly and easy to implement. Conversely, existing approaches require more complex deployment procedures, potentially involving additional configurations and expertise. Regarding training time, the IGMP approach requires only 2 hours to train and establish an effective model for attack detection. In comparison, existing approaches typically demand a longer training time of 4 hours. This signifies the efficiency of the IGMP approach in quickly acquiring the necessary knowledge and adapting to the virtualized environment. Overall, the findings presented in this table highlight the strengths of the IGMP approach in terms of scalability, robustness, compatibility, ease of deployment, and training time. Its high scalability, robustness level of 95%, broad compatibility, simplicity of deployment, and shorter training time make it an appealing choice for virtualized environments, offering improved performance, reliability, and ease of use compared to existing approaches.

5. CONCLUSION

Virtualization is a powerful technology that provides numerous benefits for data centers, including scalability, availability, and cost efficiency. However, it also introduces security challenges that must be addressed to protect the integrity and availability of virtualized environments. This study proposes an innovative approach based on deep learning and the Integrated Gramian Markov Plot (IGMP) technique for detecting attacks and mitigating their impact on virtual machines (VMs). By transforming the event sequences of VMs into event images using IGMP, meaningful features can be extracted using deep learning models. These models are capable of accurately classifying specific attack classes, enabling proactive detection and isolation of VMs when an attack is predicted within the physical machine. The experimental results demonstrate the effectiveness of the IGMP method, achieving an impressive attack prediction accuracy of 96%, surpassing existing approaches by at least 2%. By leveraging deep learning and IGMP, the proposed approach enhances the overall resilience and reliability of virtualization security. It offers a proactive defense mechanism that can prevent further damage by isolating suspected VMs upon attack detection. This advancement in virtualization security contributes to the ongoing efforts to create more secure and trustworthy virtualized environments.

6. REFERENCES

1. BARI, M. F. et al., (2013). *Data Center Network Virtualization: A Survey*. in IEEE Communications Surveys & Tutorials. 15(2): 909-928, Second Quarter. Available from: https://doi.org/10.1109/SURV.2012.090512.00043
2. ALOUANE, M. and BAKKALI, H EL. (2016) *Virtualization in Cloud Computing: NoHype vs HyperWall new approach*. International Conference on Electrical and Information Technologies (ICEIT). 49-54. Available from: https://doi.org/10.1109/EITech.2016.7519629
3. HU, Z. GNATYUK, S. GNATYUK, V. et al., (2017). *Anomaly Detection System in Secure Cloud Computing Environment*. International Journal of Computer Network and Information Security. 10-21. Available from: http://dx.doi.org/10.5815/ijcnis.2017.04.02
4. JIA, HEFEI. and LIU. (2019). *Security Strategy for Virtual Machine Allocation in Cloud Computing*. Procedia Computer Science. 147. 140-144. Available from: https://doi.org/10.1016/j.procs.2019.01.204
5. QIU, Y. SHEN, Q. LUO, Y. et al., (2017). *A secure virtual machine deployment strategy to reduce co-residency in Cloud*. IEEE Trustcom\BigDataSE\ICESS: 347–354. Available from: https://doi.org/10.1109/Trustcom/BigDataSE/ICESS.2017.257
6. S. JIN, J. SEOL, HUH, J. et al., (2015). *Hardware-assisted secure resource accounting under a vulnerable hypervisor*. ACM SIGPLAN Notices. 50- 7: 201–213.
7. NIKHILESH SINGH and CHESTER REBEIRO. (2021). *LEASH: Enhancing Micro-architectural Attack Detection with a Reactive Process Scheduler*, arXiv. Available from: https://doi.org/10.48550/arXiv.2109.03998
8. SHIH-WEI LI. JOHN S. KOH. and JASON NIEH. (2019). *Protecting cloud virtual machines from commodity hypervisor and host operating system exploits.* In Proceedings of the 28th USENIX Conference on Security Symposium (SEC'19). USENIX Association, USA. 1357–1374. Available from: https://doi.org/10.1145/3623278.3624763
9. LIU, Y. WU, Y. and LIU, R. (2018). *Comprehensive VM Protection Against Untrusted Hypervisor Through Retrofitted AMD Memory Encryption*. In 2018 IEEE International Symposium on High Performance Computer Architecture (HPCA 2018). 441-453. Vienna, Austria. Feb. Available from: https://doi.org/10.1109/HPCA.2018.00045
10. INOKUCHI, K. and KOURAI, K. (2020). *Secure VM management with strong user binding in semi-trusted clouds*. J Cloud Comp 9(3).

11. Zhu, M. Tu, B. and Wei, W. (2017). *HA-VMSI: A Lightweight Virtual Machine Isolation Approach with Commodity Hardware for ARM In: Proceedings of the 13th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*. 242–256. Available from: http://link.springer.com/10.1007/s10207-019-00450-1
12. Kourai, K. and Kajiwara, T. (2015). *Secure Out-of-band Remote Management Using Encrypted Virtual Serial Consoles in IaaS Clouds In: Proceedings of International Conference on Trust.* Security and Privacy in Computing and Communications. 443–450. Available from: https://doi.org/10.1109/Trustcom.2015.405
13. Miyama, S. and Kourai, K. (2017). *Secure IDS Offloading with Nested Virtualization and Deep VM Introspection In: Proceedings of European Symposium on Research in Computer Security*. 305–323. Available from: http://dx.doi.org/10.1007/978-3-319-66399-9_17
14. Kolesnikova, and Svetlana. (2016). *Evaluation of Hypervisor Stability towards Insider Attacks. Journal Of Electronic Science and Technology*. 14. 10.11989/JEST.1674-862X.510022. Available from: https://www.journal.uestc.edu.cn/en/article/doi/10.11989/JEST.1674-862X.510022
15. Dinh Ngoc Tu. Boris Teabe Djomgwe. Alain Tchana. et al., *Mitigating vulnerability windows with hypervisor transplant. EuroSys 2021 - European Conference on Computer Systems*, Apr 2021. Edinburgh / Virtual, United Kingdom. 1-14. Available from: http://dx.doi.org/10.1145/3447786.3456235
16. Wu. Jiang. Lei et al., (2017). *An Access Control Model for Preventing Virtual Machine Escape Attack.* Future Internet. 9. 20. Available from: https://doi.org/10.3390/fi11030082
17. Nathiya, T. and Suseendran, G. (2019). *An Effective Hybrid Intrusion Detection System for Use in Security Monitoring in the Virtual Network Layer of Cloud Computing Technology*. In Data Management, Analytics and Innovation. Advances in Intelligent Systems and Computing; Balas, V., Sharma, N., Chakrabarti, A., Eds.; Springer: Singapore. 839. Available from: http://dx.doi.org/10.1007/978-981-13-1274-8_36
18. Pan, W. Zhang, Y. Yu, M. et al., (2012). *Improving virtualization security by splitting hypervisor into smaller components*. In IFIP Annual Conference on Data and Applications Security and Privacy, Paris, France, 11–13 July 2012. Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Springer Nature: Basel, Switzerland. 7371: 298–313.
19. Dildar, M. S. Khan, N. Abdullah, J. B. et al., (2017). *Effective way to defend the hypervisor attacks in cloud computing*, 2017 2nd International Conference on Anti-Cyber Crimes (ICACC). 154-159. Available from: https://doi.org/10.1109/Anti-Cybercrime.2017.7905282
20. Eckmann, J. P., Kamphorst, S. O. and Ruelle, D. (1987). *Recurrence plots of dynamical systems*, Europhys. Lett. 4(9): 973-977. Available from: http://dx.doi.org/10.1209/0295-5075/4/9/004
21. Vashishtha, L.K. Singh, A.P. Chatterjee, K. et al., (2022). *Hybrid Intrusion Detection Model for Cloud Based Systems.* Wireless Pers Commun. Available from: https://doi.org/10.1109/ICCISci.2019.8716422
22. Elmasry, Wisam, Akbulut, et al., (2021). *A Design of an Integrated Cloud-based Intrusion Detection System with Third Party Cloud Service Open Computer Science*. 11(1): 365-379. Available from: https://doi.org/10.1515/comp-2020-0214
23. Prasad Purnaye, and Vrushali Kulkarni, (2021). *Evidence Detection in Cloud Forensics*. IEEE Dataport. October 26. Available from: https://doi.org/10.1109/MCC.2017.39
24. Aldawood, Mansour and Jhumka, (2021). *Sit Here: Placing Virtual Machines Securely in Cloud Environments*. Available from: http://dx.doi.org/10.5220/0010459202480259
25. Saxena, Deepika. Gupta, et al., (2021). *A Secure and Multi-objective Virtual Machine Placement Framework for Cloud Data Centre*.
26. Long, V. D. and Duong, T. N. B. (2020). *Group Instance: Flexible Co-Location Resistant Virtual Machine Placement in IaaS Clouds*. 2020 IEEE 29th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE). 64-69. Available from: https://doi.org/10.1109/WETICE49692.2020.00021
27. Al Zoubi, R. Mahfood, B. Abbas, S. (2022). *Security Issues and Defenses in Virtualization.* In Proceedings of International Conference on Information Technology and Applications: ICITA 2021. 605-617. Singapore: Springer Nature Singapore.
28. Avinash, J. Pranay, K. and Goud, N. R. (2021). *Tour Towards The Security Challenges Of Virtualization In Cloud Computing: A Survey. In 2021 5th International Conference on Trends in Electronics and Informatics* (ICOEI) . 771-776. IEEE. Available from: https://doi.org/10.1109/ICOEI51242.2021.9452879
29. Balmany, EL. C. Tbatou, Z. and Asimi, A. (2022). *Secure Virtual Machine Image Storage Process into a Trusted Zone-based Cloud Storage.* Computers & Security. 120. Available from: https://doi.org/10.1016/j.cose.2022.102815
30. Doyle, J. Golec, M. and Gill, S. S. (2022). *Blockchainbus: A lightweight framework for secure virtual machine migration in cloud federations using blockchain.* Security and Privacy. 5(2). Available from: https://doi.org/10.1002/spy2.197

©2024: The Royal Institution of Naval Architects