Salud, Ciencia y Tecnología - Serie de Conferencias. 2024; 3:933
doi: 10.56294/sctconf2024933

Category: STEM (Science, Technology, Engineering and Mathematics)

ORIGINAL

Adaptive firefly algorithm for resource allocation and modified advanced encryption standard algorithm for hypervisor attack detection on cloud computing

Algoritmo adaptable de firefly para la asignación de recursos y algoritmo estándar de cifrado avanzado modificado para la detección de ataques de hipervisor en computación en la nube

Banu Priya MR[1], Maheswari D[2]

[1]Department of Computer Science, Rathnavel Subramaniam College of Arts and Science (Autonomous). Sulur, 641042, Coimbatore.

Cite as: Banu Priya MR, Maheswari D. Adaptive firefly algorithm for resource allocation and modified advanced encryption standard algorithm for hypervisor attack detection on cloud computing. Salud, Ciencia y Tecnología - Serie de Conferencias. 2024; 3:933. https://doi.org/10.56294/sctconf2024933

Submitted: 10-02-2024 **Revised:** 30-04-2024 **Accepted:** 20-06-2024 **Published:** 21-06-2024

Editor: Dr. William Castillo-González

ABSTRACT

Introduction: the advantages and wide-ranging applications of cloud computing have made it a prominent attention for scholars these days. The dispersed structure of cloud and its whole dependence on the internet for service delivery extant security problems.
Method: hypervisor attack detection is carried out in this study using Modified Advanced Encryption Standard (MAES) system which ensures security prominently. It finds and detects the attacks earlier for secured VM migration. The proposed system includes main phases including system framework, load balancing, resource allocation and hypervisor attack detection via MAES algorithm. Initially, Over the duration of cloud computing, consider the quantity of tasks, VM, and cloud users. In this research, the MMH system is employed for load balancing to balance the total burden across the cloud. Tasks are moved from overloaded to underloaded nodes to attain load balancing. Next, the allocation of resources is carried out utilizing Adaptive Firefly (AF) optimization system which is used to select best resources optimally. It generates the best fitness values to choose the best resources.
Results: it is also focused to improve the cost metric, computational complexity, throughput and VM performance in cloud. Then, to detect hypervisor attacks, the MAES method is employed. It specializes on offering enhanced security for cloud data and is employed to identify hypervisor and VM attackers.
Conclusions: the findings produced the conclusion that the suggested MAES method superior to the current approaches according to throughput, computation cost, Mean Square Error (MSE) rate, and energy use.

Keywords: Hypervisor Attack Detection; Modified Advanced Encryption Standard (MAES) Algorithm; Adaptive Firefly (AF) Optimization Algorithm.

RESUMEN

Introducción: las ventajas y la amplia gama de aplicaciones de la computación en la nube la han convertido en una atención destacada para los académicos en estos días. La estructura dispersa de la nube y su total dependencia de Internet para la prestación de servicios plantean problemas de seguridad.
Método: la detección de ataques de hipervisor se lleva a cabo en este estudio utilizando el sistema Modified Advanced Encryption Standard (MAES) que garantiza la seguridad de manera destacada. Encuentra y detecta los ataques antes para una migración segura de VM. El sistema propuesto incluye fases principales que incluyen el marco del sistema, el equilibrio de carga, la asignación de recursos y la detección de ataques de hipervisor mediante el algoritmo MAES. Inicialmente, durante la duración de la computación en la nube,

considere la cantidad de tareas, máquinas virtuales y usuarios de la nube. En esta investigación, se emplea el sistema MMH para equilibrar la carga a fin de equilibrar la carga total en la nube. Las tareas se trasladan de nodos sobrecargados a nodos poco cargados para lograr el equilibrio de carga. A continuación, la asignación de recursos se lleva a cabo utilizando el sistema de optimización Adaptive Firefly (AF) que se utiliza para seleccionar los mejores recursos de manera óptima. Genera los mejores valores de fitness para elegir los mejores recursos.
Resultados: también está enfocado a mejorar la métrica de costos, la complejidad computacional, el rendimiento y el rendimiento de las VM en la nube. Luego, para detectar ataques de hipervisor, se emplea el método MAES. Se especializa en ofrecer seguridad mejorada para los datos en la nube y se emplea para identificar atacantes de hipervisores y máquinas virtuales.
Conclusiones: los hallazgos produjeron la conclusión de que el método MAES sugerido es superior a los enfoques actuales en cuanto a rendimiento, costo de cálculo, tasa de error cuadrático medio (MSE) y uso de energía.

Palabras clave: Detección de Ataques de Hipervisor; Algoritmo de Estándar de Cifrado Avanzado Modificado (MAES); Algoritmo de Optimización Adaptive Firefly (AF).

INTRODUCTION

The cloud computing revolutionizes information technology (IT) by offering end users virtualized, scalable assets on demand that require less maintenance and technology. These resources are delivered across the Internet utilizing established networking protocols, standards, and formats, and they are overseen by several management groups.[1] Vulnerabilities and defects in the underlying technology and legacy protocols can allow attackers to get access.

Cloud principles like multi-tenancy, resource pooling, and outsourcing present serious difficulties for the security sector. Furthermore, new security risks in the cloud environment are brought about by online technologies and trusted third parties providing cloud services. In recent times, novel security simulations, protocols, and regulations have emerged, making cloud security an important study area.[2] As a result, current study on cloud security still has issues with enhancing detection accuracy and identifying novel or unidentified cloud assaults. Many security experts have concentrated on creating cloud security frameworks with an array of innovative security techniques in order to meet the previous constraints.

Without requiring control or management of the underlying tech, executing complex computations on a big scale and providing services in a distributed environment has become increasingly common adopting this method.[3] The significant technologies in a cloud environment are task scheduling. It is a policy for task scheduling:

- Where a task is defined as work which requires completion in a set amount of time.
- Scheduling which alloting suitable resources to specific assignments that are broken down into smaller tasks.

Task scheduling is becoming a problem to achieve optimum resource utilization and minimal execution time while preserving quality of service.[4,5] Task scheduling is a vital strategy in cloud because it helps allocate jobs to the right resources for efficient resource use and overall system performance optimization.

The multitude of incoming tasks with varying resource requirements makes load balancing of jobs on VM a crucial component of task scheduling. The process of identifying overloaded and underloaded nodes between all of the nodes and then distributing the load termed as load balancing.[6] To increase user happiness, expedite task execution, and enhance system stability, load balancing intends to maximize the network's makespan, completion time, and resource usage rate. So, Meta-heuristic methods resolve to the best results are usually employed to achieve the objectives. Assigning tasks to various heterogeneous VMs in a way that reduces or maximizes at minimum one QoS parameter including makespan time, execution cost, is the ideal option.

VM to VM, Hypervisor to VM, VM to Hypervisor, and Hypervisor to Hypervisor are four groups into which assaults on virtual machines can be separated. It is not possible to limit new threats using standard network security measures including intrusion prevention systems (IPS), and intrusion detection systems (IDS) alone. A malicious VM employ a hypervisor instead of a physical network to generate new communication channels. This emphasizes how crucial it is to create a new framework that protects network security independent of the network. An attacker can target each VM on a virtual host by using a compromised hypervisor. Denial of Service (DoS), code execution, launching pointless services, memory corruption, and outdated hypervisors are examples of hypervisor vulnerabilities.

The key benefit of VM migration is the identification of hotspots in the data centers. The capacity to balance the load, perform system maintenance, etc., is provided by VM migration. The technology of virtualization

powers cloud computing. Pre-copy and post-copy approaches are the two categories into which VM migration strategies may be separated. VM migration is a method of transferring active programs, from one physical device to another. The migration procedure involves moving the network connection, storage, memory, and processor state from one host to an alternative.[7] Users are primarily concerned with two critical performance indicators: downtime and total migration time. These metrics address service deterioration and the duration of service outages.

This research's primary goal is to identify hypervisor attacks. Despite the introduction of numerous research studies and approaches, there has been little progress in detecting hypervisor attacks. To overcome the abovementioned issues, in this research, Adaptive Firefly (AF) and Modified Advanced Encryption Standard (MAES) algorithm is suggested to enhance the overall VM migration performance in cloud. The main significance is the formation of system framework, virtualization, load balancing, resource allocation and hypervisor attack detection. The suggested system leverages effective methods in a cloud context to improve attack detection efficiency.

The balance of the study is organized as follows: section 2 clarifies the studies on resource allocation and assault detection. Section 3 provides specifics on the AF+MAES method's suggested system. Section 4 contains the experimental data and a description of the performance assessment. Section 5 explains a summary of the conclusions.

Related work

Zhou et al.[8] introduced EEOM system. The three crucial elements of the EEOM system, VM selection, host location, and trigger time are improved when CPU and memory constraints are considered. Utilizing virtualization technology, the EEOM method distributes some VMs from heavily and moderately loaded hosts to other hosts. To save energy, the idle hosts are either turned off or put into low-power mode. The testing outcomes demonstrate that the EEOM method eliminates 13 % SLA violations and saves 7 % energy usage when compared to the DT method.

Thiam et al.[9] studied the issue of choosing the best migration and VM allocation strategy to reduce energy use in a data center whilst preserving QoS. A cloud is created utilizing the CloudSim simulator. It recommends an interface for interacting with both real and VM. To determine the methods, VM placement and migration were assessed and contrasted. The simulation's outcome demonstrates that VM placement and migration strategies reduce energy usage, migrations, and simulation time overall. These days, VM management techniques in a range of situations are receiving more attention. Placing and migrating VM is an optimization problem with several objectives. As a result, a data center's performance is greatly influenced by the location and method of provisioning VM. In addition to preventing QoS degradation and mitigating hotspots, an effective VM allocation policy would save data center operating costs and increase energy efficacy. Reducing the quantity of PMs where VMs are migrated helps optimize cloud server utilization and lower cloud data center energy consumption.

Adeshara et al.[10] discussed resource consumption by cloud server VMs. The system's capacity to function effectively is mostly governed by factors like system utilization, which is dependent on the use of resources by individual VMs and PM resources overall. It is essential to gather information, make decisions based on the circumstances, and then go forward by moving the VM to determine whether a PM is required in order to maximize usage. To automate this method and arrive at these judgments, a monitor machine that independently monitor the PM and VM along with the appropriate activities must be appointed. The monitor machine needs to do multi-objective optimization, taking into account other aspects like QoS and minimizing migration time and distance.

Saxena et al.[11] suggested SM-VMP technology collectively with a productive VM migration. By minimizing intercommunication latency, the architecture guarantees an energy-effective allocation of physical sources across VMs, thus highlighting that user applications should be executed securely and promptly. The WOGA implement the VMP. In comparison to current method, the performance evaluation of static and dynamic VMP showed a significant decrease in shared servers, intercommunication costs, energy use, and execution time, with reductions of up to 28,81 %, 25,7 %, 35,9 %, and 82,21 %, accordingly, along with an increase in resource utilization of up to 30,21 %.

Hung et al.[12] presented a two-stage genetic approach for cloud computing VMH load balancing utilizing migration. Prior approaches typically view this problem as a job-assignment issue and only account for the loads of the VMHs that are in operation at present. Nevertheless, these approaches can only be somewhat successful in real-world settings if the loads of the VMHs after balancing are not utilized. Two genetically based approaches are combined and described in this work. Initially, VM metrics are derived from their creation characteristics and matching cloud computing environment efficiency is evaluated. Symbolic regression algorithms are created utilizing the GEP to forecast the loads of VMMHs during load balancing. in addition to the projected VMH loads from GEP, the evolutionary method determines the ideal VM-VMH assignment for load balancing and VM migration by considering both the present and future loads of VMHs. The techniques' effectiveness is assessed in Jnet, a real cloud computing environment, where they function as a centralized load balancing system.

https://doi.org/10.56294/sctconf2024933

Nikolai et al.[13] suggested framework for utilizing hypervisor indicators of performance for intrusion detection and security while utilizing the virtualization technology. Validate the suspicious attacks and examined without OS, employing VM performance metrics collected, including packets transmitted, and CPU usage. Along with to having several advantages over network-based and host-based IDS, the hypervisor-based cloud IDS may be utilized in conjunction with these more conventional intrusion detection techniques. It also doesn't require additional applications to be loaded in VM.

Kumara et al.[14] discussed a key component of virtualization is the hypervisor, which permits resource sharing across virtual computers. An attacker may utilize VM vulnerabilities to conduct extreme persistent assaults like Trojan horses, DoS, DDoS attacks, and covert rootkits. Due to their easy availability for CSP rental, VMs are a great target for malicious cloud users or attackers looking to conduct attacks. Attacks on VM have the potential to stop cloud services from operating normally. Every VM must have a defense mechanism in place in order to detect and stop threats quickly, ensuring that the virtual setting is secured. In-and-Out-of-the-Box VMs and hypervisor-based IDS are employed to detect and remove rootkits and other threats, ensuring the VM's robust state. experimented with OSSEC. Rootkits with both Linux and Windows platforms, denial-of-service attacks, and file integrity verification tests are carried out and effectively identified by OSSEC.

Dildar et al.[15] suggested the VMHIDS method for identifying and stopping hypervisor threats. The VMHIDS has incorporated a number of characteristics from other methods, such as routine task inspections that stop questionable activity before it starts. The VMHIDS reduces the attack by the hypervisor.

Aldribi et al.[16] introduced a HIDS that finds unusual network behaviors employing online multivariate statistical change assessment. To enhance detection capabilities, develop an IOFM that takes advantage of the unique and associated behaviors of instances within a hypervisor, which deviates from the traditional monolithic network IDS feature approach. An extensive range of attack vectors are included in a newly created cloud intrusion dataset, which is utilized to assess the methodology.

Elshabka et al.[17] suggested a SDVMC method. An innovative VM placement approach designated MRI with RITH is employed in the SDVMC segment, while the SMM employs a 3D security evaluation framework. The host selected utilizing the recommended approach minimizes the total security threat increase while guaranteeing that the risk rise for every VM remains within the RITH constraint value, that is established in line with the cloud provider's standards. The tests show that using the RITH 0.8 technique enhances security; total risk was reduced by 2 % to 5 % without negatively impacting QoS or energy usage. Furthermore, the tradeoff among energy usage and total security risk can be enabled when employing this strategy with RITH smaller than 0.8. The highest possible reduction in the total risk might vary from 10 % to 40 % based on the degree of communication overhead among the VMs. Still, the maximum energy consumption was reduced to less than half when non-power-ware VM allocation rules were employed. VMs may serve as launchpads for assaults against the other co-resident VMs and the host hypervisor.

METHOD

In this work, Adaptive Firefly (AF) and Modified Advanced Encryption Standard (MAES) algorithm is proposed for hypervisor attack detection throughout VM migration over cloud setting. The suggested study includes the system model, virtualization, load balancing, hypervisor attack detection during VM migration and estimation of outcomes.

System model

The system framework comprises the number of VMs, cloud users, CPU, memory utilization, and resources. To accomplish specific objectives, a cloud is built to carry out a collection of apps. Some cloud resources are required for the execution of those apps.[18] The location of VMs is critical to the effective use of data center resources. Each host in the data center can be in one of two states: either active or in sleep mode. During its active state, the host implies this device is linked to an execution job; but, when it is in the sleep state, it suggests that it is not associated with any machine. It is categorized as a finite VM type according to factors like memory, computing power, and storage capacity.

Virtualization

The purpose of virtualization technology is to set up a hypervisor. The components of a VMH are protected by an OS known as a hypervisor, which also manages responses and requests from VMs. The app's parameters define which form of hypervisor type-1, that utilizes the VMH hardware properly, and which type-2, which utilizes the VMH OS.[19] Virtual memory (vRAM), virtual hard drives (vHDs) supplied by a VMH, and virtual processors (vCPUs) constitute an abstract computer. A VM user program cannot reach the actual hardware directly. rather the hypervisor of the VMH envelops all resources. Typically, a VMH runs and manages several VMs.

In theory, a VM consists of two files: a disk image file and a configuration file. A volatile "memory page" is formed in the VMH's primary storage when the VM is executing. These files are generated on the VMH's storage

https://doi.org/10.56294/sctconf2024933

upon the creation of a virtual machine. Thus, to delete, copy, or move a VM, one must also delete, copy, or move these files. One benefit of virtualization is the ease with which VMs can be migrated from one virtual machine health system (VMH) to additional one. VM execute on several VMH platforms through migration. Prior to starting the migration, the VM needs to be stopped. It is necessary to move the memory pages linked with the VM from the selected VMH's primary storage to the current VMH's memory. The VM can be turned on once more once all of the movements have been finished.

Load balancing using Max-Min Heuristic (MMH) algorithm

Efficient load balancing is performed by using Max-Min Heuristic (MMH) algorithm. Distributing work amongst the available resources to prevent any resource from being over or underutilized is known as load balancing. VMs, data centers, physical computers, and software are the primary sources utilized for load balancing in the clouds.[20]

Need for load balancing in cloud computing

Tasks and other endeavors to the VM are called loads in cloud. Three types are employed to classify these loads:

- Under-loaded.
- Over-loaded.
- Balanced.

Load balancing techniques are accountable for trying to maintain an equilibrium between the total workloads in the cloud. The network's throughput is improved by doing this by moving work from overloaded to underloaded nodes.

The quantity of jobs in the waiting line is known as the load, and depending on the nature of the work, it can be low, moderate, or heavy. By ensuring that every cloud computing node is uniformly employed to the greatest extent possible to maximize throughput and reduce execution times, load balancing improves the speed of computational cloud systems. Dynamic load balancing determines the scheduling decisions when a project requires to have scheduled for extra work. It is well known that load balancing can provide consumers with exceptional service productivity while optimizing the system's overall availability and utilization.

Every resource utilized by the VMs of cloud servers will have its current load calculated. Following the computation of each resource's load index, the process of allocating resources to the appropriate node to lower the load value will start the load balancing operation, which will efficiently employ the resources effectively. The assignment of resources to appropriate nodes is considered an optimal distribution problem, leading to the introduction of the Max-Min heuristic system. A scheduling is created with the goal of optimizing the scheduling queue while placing a premium on node load balancing. The method of delivering jobs requires a successful selection of the ideal physical host to achieve load balancing of cloud.

The suggested strategy starts a process that schedules the nodes in the cloud network by creating an efficient load balancing mechanism. The Max-Min heuristic method starts with a collection of jobs that are not mapped. The group of nodes with the minimum quantity of load possession is given preference throughout the scheduling procedure. Afterwards, the method selects the jobs with the largest make span and allots them to the following VM. When the efficient machine is given to this task, it is eliminated from the set of tasks. The procedure is continued until each task is finished. It is ensured that a task can be completed simultaneously with lower execution times by mapping it to a large capacity machine. The adoption of the Max-Min job scheduling technique enhances make span. However, increased make span guarantees a more evenly distributed load throughout all of the computers.

Finding the Lowest Load Node: the MMH method serves as the model for the procedures involved in finding the lowest load node. the computation of the separation among the scheduling queue nodes. Before performing a computation, it finds the node with the least load values. To determine the least distinct nodes, the queue requires the node with the least load.

The Cartesian distance is employed to compute the node's distance values:

$$dist = \sqrt{\sum_{j=1}^{k}(n_i - n_j)^2} \qquad (1)$$

The comparison node is n_j, while the selected node is n_i. The distance is displayed utilizing the expression Dist. After calculating all the distances among the node values, the nodes are shifted from the pivot node to the least distinct node. The queue at the highest level of the schedule list is the scheduling queue which is considered critical. Ultimately, the optimal server node will receive the service resource based on cloud

https://doi.org/10.56294/sctconf2024933

load balancing. It assists them in making well-informed choices about whether to assign a task to a VM with a low capacity or to wait for a machine with a high capacity to operate. Additionally, MMH is more effective at scheduling all of the jobs in the waiting mode. The primary objective of these heuristic techniques for allocating tasks to resources is to ensure quick task completion, hence reducing the overall make span of the VM. The MMH system allocate larger tasks to VMs with higher capacity and shorten their wait times. The smaller jobs are allocated to underused or idle machines after each significant work has been assigned to the proper device. By ensuring that every machine is employed equally, this lowers the possibility that any one of the machines will be either underutilized or overworked.

Algorithm 1: MMH algorithm for load balancing

1. Start
2. For all the tasks in the set G; Ti
3. For all the resources ; Rj
4. Cij=Eij+rj
5. While the tasks set G has tasks
6. Choose a task Tk with the maximum completion time.
7. Use (1)to determine the separation among nodes.
8. Arrange the materials according to their finishing time.
9. Assign the task to a resource (Rj) that will provide the shortest possible execution time.
10. Empty the task Tk from set U.
11. Until the set U is empty, continue with steps 1 through 7
12. Equilibrate the cloud loads
13. End

From the above pseudocode, rj denotes the resource Rj that is prepared to perform a task. Cij denote the expected completion time and Eij the execution time. It maximizes the utilization of the VM and CPU. Thus, The MMH method effectively equalizes loads on the cloud platform and significantly improves load balancing throughout all of the cloud computing units.

Resource allocation Adaptive Firefly (AF) algorithm

For effective resource allocation across cloud, AF is suggested. Resource Allocation (RA) is the process of distributing available resources in an effective manner among cloud applications and consumers. Utilizing Infrastructure as a Service (IaaS) as the foundation is the more problematic tasks in cloud. Additionally, RA for IaaS in cloud computing offers a number of advantages: it is economical since users don't have to install and maintain software to utilize the applications; it is flexible enough to enable users to access information and programs on any system in globally; and there are no restrictions on the standard or usage site.

It offers a method for allocating resources that makes load balancing effective. Resource allocation according to AF is employed to accomplish load balancing. It gives guidelines for the search parameters and employs fireflies to automatically change the direction of the search. While the search space is significant, AF performs optimally.

Real fireflies' social and biological characteristics served as influence of the Firefly Algorithm.[21] Genuine fireflies release rhythmic light that acts as a warning system to keep predators away and helps in luring potential mates. This flashing behavior is articulated by FA utilizing the problem's objective function as the basis for optimization. The flashing lights of the firefly serve as the basis for FA's operation. The light's brightness stimulates a firefly swarm to go to areas that are bright and enticing, and these places are tracked to get the greatest resolution possible throughout the search area.

A few firefly characteristics are standardized by this method, shown by Liu et al.[22]:

a) Whichever a person's sex, fireflies are drawn to each other.

b) The firefly's illumination is directly correlated with its attraction; The more brilliant firefly prefers to attract the less bright one when there are two of them. If a firefly cannot find a brighter firefly nearby, it will move randomly.

The objective function regulates the firefly's brilliance in quantitative form.

The Firefly algorithm was selected due to its capacity to yield optimal solutions for issues involving several objectives. The intensity of an optimization issue can be expressed as a simple function of the objective function. It is predicated for simplification that a firefly's brightness, which is connected to the primary function conveyed, establishes the attractive.

The task that has been requested is represented by the fireflies, and the cloud wherein the application is running is represented by its brightness. It is linked to a distinct ID, and the cloud broker utilizes AF to link jobs

https://doi.org/10.56294/sctconf2024933

to available resources.[23] Between 300 and 400 applications are created for a simulation by various users. The form's length, the tasks it contains, and the server's volume are all stated. Iteratively repeating these stages continues until the convergence requirements are achieved.

a) Light Intensity and Attractiveness at the source: the light intensity improves according to the inverse square law.

$$I(r) = \frac{I_0}{r^2} \qquad (2)$$

Where I(r) is the brightness of the light at attractive r^2 appealing produced by giving VMs jobs at arbitrary.

b) The light intensity is when the intermediate is provided:

$$I(r) = I_0 \exp(-\gamma r)0 \qquad (3)$$

Where I_0 is the medium's absorption coefficient.

c) To prevent singularities, the following Gaussian approximation form is taken into consideration.

$$I(r) = I_0 \exp(-\gamma r^2) \qquad (4)$$

A firefly's beauty is directly correlated with the amount of light it perceives relative to its nearby counterparts. By taking variance into account and adjusting the allocation at arbitrary, an innovative approach is produced. Certain jobs are shown as appealing when they need to be distributed across the resources in a batch. Thus, a firefly's attraction coefficient ß is as follows.

$$\beta = \beta_0 \exp(-\gamma r^m) \qquad (5)$$

Here $ß_0$ is the attractiveness at r=0.

The distance is calculated using the following formula for any two fireflies that are positioned i and j.

$$r_{i,j} = \sqrt{\sum_{k=1}^{d}(x_{i,k} - x_{j,k})^2} \qquad (6)$$

Where $x_{i,k}$ is the kth is spatial match factor x_i of the kth firefly and d is the quantity of dimensions. Adaptation parameter is introduced for absorption and random variable, resulting in AF. These modifications increase the capacity for both local and global searches by altering variables linearly over the course of the iterations. AF is employed to schedule virtual machines among data centers to produce the best resource allocation possible in cloud computing. By employing load balancing, the burden is dispersed among all of the nodes in a dynamic manner. In addition to maximizing throughput, minimizing data center processing times, minimizing user response times, and optimizing resource usage, load balancing helps prevent any one resource from becoming overloaded.

Compute the variable α:

$$\alpha(t+1) = \left(1 - \frac{t}{MaxG}\right)\alpha(t) \qquad (7)$$

a adjusts the value cantered the optimization's level of distance variation to improve the solution's correctness and rate of convergence. The increase population flexibility:

$$\alpha = \alpha_{min} + (\alpha_{max} - \alpha_{min}) \times ||x_i - x_{best}||/L_{max} \qquad (8)$$

$$L_{max} = (x_{worst} - x_{best}) \qquad (9)$$

$α_{max}$ and $α_{min}$ are the maximum and minimum resources individually. In equation (9) x_{worst} represents the place of the worst individual at the generation t firefly, and L_{max} is the separation among the worst individual x_{worst} and the global optimal individual x_{best}. In early process, the fireflies are scattered over the entire area, and most of

https://doi.org/10.56294/sctconf2024933

them are not at all the perfect people that people would have all over the globe. Currently, the value of $\|x_i - x_{best}\|$ is larger, L_{max} and $(\alpha_{max}-\alpha_{min})$ are fixed values. Consequently, equation (8) demonstrated that an early stage with a greater value of α had a greater global optimization effect. Fireflies' individual is fascinated to fireflies that are brighter than oneself and in close proximity to global ideal resources as a result of the method's implementation. Later, firefly people that will assemble around the world's best people, the value of $\|x_i - x_{best}\|$ is smaller at this time, which is advantageous to enhance the cloud's best resource search. The method's convergence speed is increased by varying the value of α in each iteration reliant on the optimal position. The research above indicates that the step size factor α varies dynamically and adaptively reliant on the separation amongst firefly individuals, balancing the search and algorithm development capabilities.

Energy consumption and delay are assumed by a new fitness function:

$$f(x) = \frac{\left({}^{m_d}/_{m_t}\right)\times\left({}^{P_i^r}/_{P_{init}}\right)}{exp^{-{}^{e_D}/_{e_M}}} \qquad (10)$$

Here m_d is the quantity of dropped messages. m_t is the total quantity of messages sent with greater throughput.

P_i^r is the remaining power in node i.

P_{init} is the initial power.

e_D is the end to end delay and e_M is the maximum allowable delay.

$$x_i = x_i + \beta_0 e^{-\gamma r^2}\left(x_j - x_i\right) + \alpha\left(rand - \frac{1}{2}\right) \qquad (11)$$

Where x_i and x_j is distance among two firefly nodes.

Every VM's fitness value is computed inside the population. The OpenVMS is randomly assigned to a batch of jobs in the first generation. Every firefly's fitness value is computed. The next step is to select two fireflies utilizing the selection process. The firefly, which is preferred for the following generation, has the highest fitness rating along with more brightness.

Algorithm 2: AF for resource allocation

Input: number of tasks and resources.

Output: optimal resource selection.

1. Objective function (x), $x = (x1,\ldots,)T$ consider lower energy consumption and higher throughput as objective function
2. Generate initial population of fireflies xi (i = 1, 2, . . . , n)
3. Light intensity Ii at xi is initiate via $f(xi)$
4. State light absorption coefficient γ
5. while (t < MaxGeneration)
6. for i=1:n all n fireflies
7. for j=1:i all n fireflies
8. if ($Ij > Ii$), Transfer firefly i to j in d-dimension;
9. end if
10. The change in attractiveness with range r via exp[$-\gamma r$]
11. Estimate fitness function utilizing (8) and (9)
12. Estimate objective model utilizing
13. (4)Evaluation new solutions and update light intensity utilizing
14. (1)Minimize the data transmission overhead utilizing
15. (8)Update the optimal RA task utilizing
16. (6)end for j
17. end for i
18. Sort the fireflies and discover the present best
19. end while
20. A firefly shifts to a more attractive

The AF system is applied to improve the energy and delay metrics and generate optimal solutions. The fireflies are organized utilizing the AF method, and the best firefly are selected based on their best fitness ratings. The selected fireflies utilize crossover and mutation to proliferate amongst individuals. The firefly iteration continues after the unique finest ideas are added to the firefly pool. Therefore, the AF method is

https://doi.org/10.56294/sctconf2024933

utilized for selecting RA tasks according to factors such as higher packet delivery, reduced energy consumption, and distance traveled. The goal is to apply a multi-objective function to choose the finest resource allocation. It consumes minimum energy and increases the VM cloud performance for improving the data transmission.

Hypervisor attack detection via Modified Advanced Encryption Standard (MAES) algorithm

The MAES method successfully executes hypervisor attack detection. AES is created on a substitution-permutation network layout.[16] With a predetermined block size of 128 bits and a key size of 128, 192, or 256 bits, AES is a Rijndael variation. By contrary, Rijndael per se is defined by block and key sizes, where any combination of 32 bits with a minimum value of 128 and a maximum of 256 bits. For 128-bit keys, ten rounds are needed; for 192-bit keys, 12 rounds; and for 256-bit keys, 14 rounds. There are several processing steps in a cycle, and the encryption key determines one of them. Reverse rounds are used to transform the ciphertext into the original plaintext employing an identical encryption key. Authorization is the method by which a user limits what they may do on the system after they are authenticated. Authentication is the procedure by which a user verifies their connection to the platform.

The term "hypervisor" refers to the software used to manage VMs in cloud computing. It looks for resources that the VMs share. Additionally, it prevents any illegal VMs from accessing the cloud environment.[24] The creation, termination, relocation, and resource allocation of VMs are the primary tasks of hypervisors.

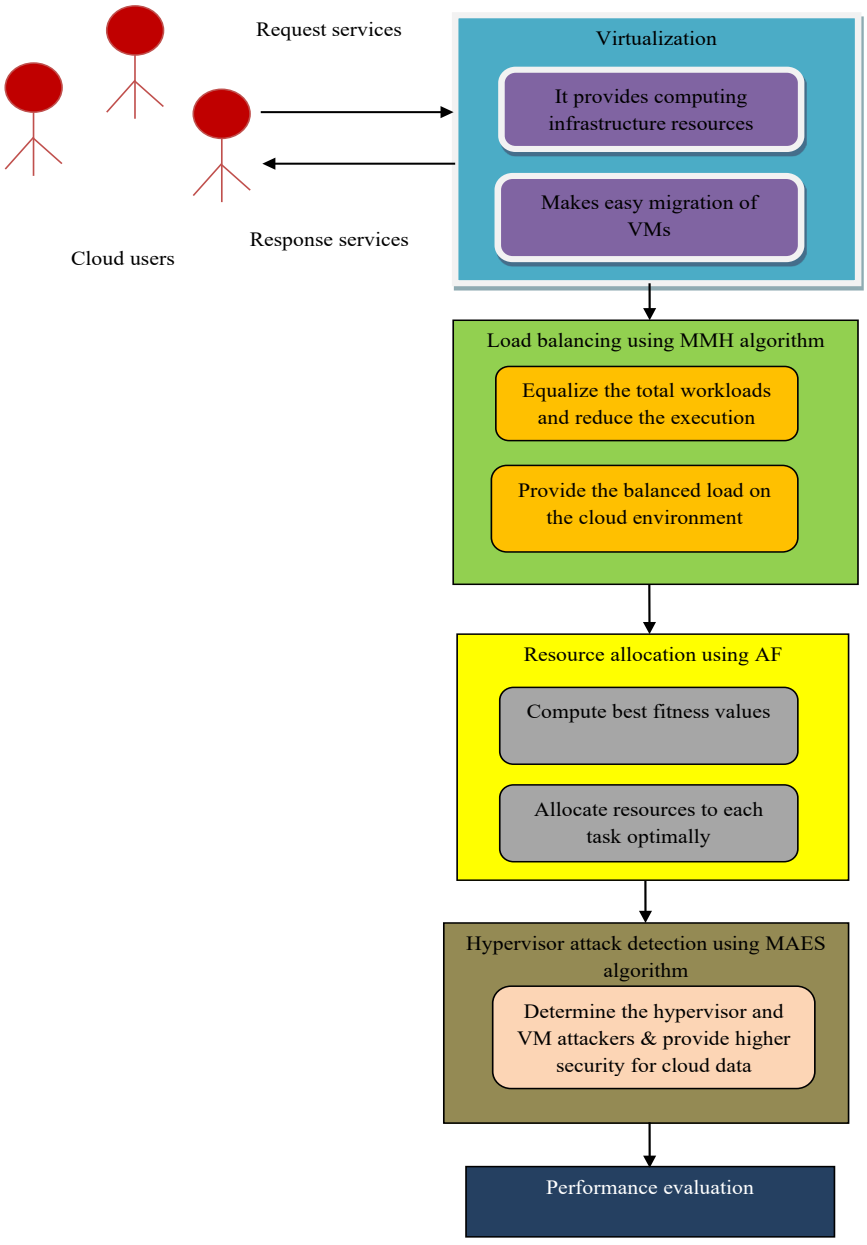

Figure 1. Overall block diagram of the proposed framework

https://doi.org/10.56294/sctconf2024933

The definition of a hypervisor attack, which falls under the category of external attacks, is the exploitation of a hypervisor's vulnerabilities to grant attackers access to and control over the hypervisors. The attackers initially targeted a specific VM. Subsequently, the malicious VM will carry out the exchange with the hypervisor. The attackers then seize the chance to spread their attacks to every VM running on the compromised hypervisor. Eventually, under the corrupted hypervisor, all VM will progressively become infected. The introduction of the double key centered method of AES allows for the detection and elimination of hypervisor assaults. The recommended framework's overall illustration is displayed in figure 1.

Hypervisor-based attacks leverage cloud requests from the hypervisor to monitor the system metrics and identify any patterns of potential misuse. The Hypervisor-based attack platform monitors and records the exchange of data among VMs, hypervisors, and virtual networking. This detection method, which runs inside the hypervisor, may work well in cloud.

Keyword Generation

The creation of AES encryption keys is a crucial step in the encryption process. The input's size and the encryption key's size are the same. Thus, the suggested method encrypts data utilizing the 16 bytes of the encryption key.[25] The 16 bytes key is organized in the form of 4 × 4matrices, which is shown below.

$$\text{Key}=\begin{matrix} k0 & k4 & k8 & k12 \\ k1 & k5 & k9 & k13 \\ k2 & k6 & k10 & k14 \\ k3 & k7 & k11 & k15 \end{matrix} \qquad (12)$$

When it comes to the encryption of the input data files, the AES method typically goes through several processing stages. It is additionally an iterated block cypher with a changeable key length and a block size of 128. The intermediate outcomes, referred to as a state, are the focus of the many transformations; at its core, rectangular bytes constitute the state. When 16 bytes, or 128 bits, constitute the block size, as illustrated in below table, the dimension of a rectangular array is 4 × 4.

Block size of the AES algorithm

b0	b1	b2	b3	b4	b5	b6	.....	b15

As seen below, the input files are arranged in a 4 x 4 matrix. The input's first four bytes are:

$$\begin{matrix} b0 & b4 & b8 & b12 \\ b1 & b5 & b9 & b13 \\ b2 & b6 & b10 & b14 \\ b3 & b7 & b11 & b15 \end{matrix} \qquad (13)$$

Every round, the state conducts a small number of operations. Among the operations:

- Sub-bytes.
- Shift row.
- Mix column.
- Add round key.

Sub-bytes operation

Every byte in the state is affected independently by the sub-bytes operation, which is a nonlinear byte replacement. Two transformations are employed for creating the reversible substitution table (S-Box).

`

The steps involved in this standard shift row procedure are as follows:

- First row to 0 positions to the left.
- Second row to 1 position to the left.
- Third row to 2 positions to the left.
- Fourth row to 3 positions to the left.

The AES technique typically performs ten-round operations for 128-bit. By altering the shift operation in this case to reflect the round operation, the unique system is enhanced. The sub-bytes operation is performed first in each cycle. Next, the shift operation according to rounds is carried out. With each round, the recommended method checks if the number of rounds is odd or even. The rows of the state array are affected by the shift

https://doi.org/10.56294/sctconf2024933

rows transformation if the round number is 1 (odd). The shift rows transformation travels on two locations in the matrix if the round number is two (even). The shift procedure, as seen in figures 2 and 3.

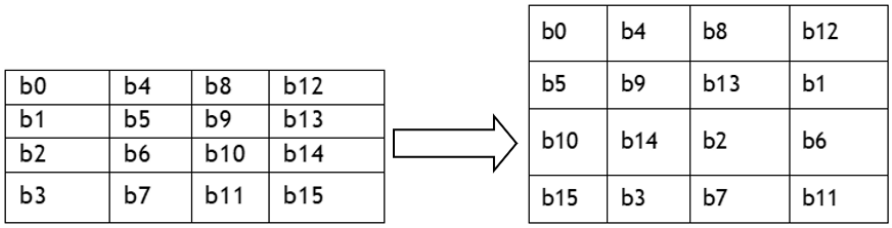

Figure 2. Shift Operation for Round Number 1 (odd)

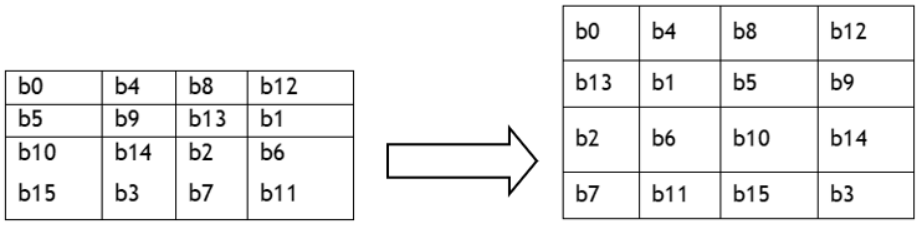

Figure 3. Shift Operation for Round Number 2 (Even)

Mix-column operation

The complex calculations are employed in the mix column operation.

Add round key

Bitwise XOR is used to determine the condition of the organization key in add round key. The round key can be attained from the cipher key making use of key schedule.

The VMs and the hypervisor are both shielded from external and insider attacks in cloud environments by the MAES technique. It is probable to examine real-time events for automatic detection and blocking of harmful occurrences by continuous monitoring employing MAES from hypervisors. MAES tracks and monitors every file and process that exchanges information inside the hypervisor. Additionally, because MAES is installed on hypervisors and VMs, it is simple to identify any recent or suspicious attacks on the hypervisors for quicker mitigation.

RESULTS

In this work, the proposed methods are implemented in Jnet. About 200 individuals can access Jnet, a private intranet made up of more than 20 different kinds of servers. Jnet was linked to the load balance monitor and the VMHs. Four VMH servers running CentOS, containing three HP ProLiant BL460c blades and one HP ProLiant ML350, comprised the trial setup. Running FreeNAS, the HP ProLiant ML110 served as the shared storage server. One gigabyte of Ethernet connects each machine.

Figures 4 and 5 list the specifications of the VM and physical hosts that were used.[17] The impact of solution is studied on hypervisor attack detection and QoS. The following metrics are applied to each variable:

Host Type	Type 1	Type 2
Total MIPS	2660	1860
Total processor units	2	2
Total RAM	8 GB	8 GB
Network bandwidth	1 GB/s	1 GB/s
Total storage size	80 GB	80 GB

Figure 4. Physical Hosts Specification

VM Type	Type 1	Type 2	Type 3	Type 4
Total MIPS	2500	2000	1000	500
Total processor units	1	1	1	1
Total RAM	1 GB	1 GB	1 GB	1 GB
Network bandwidth	100 Mbit/s	100 Mbit/s	100 Mbit/s	100 Mbit/s
Total storage size	2.5 GB	2.5 GB	2.5 GB	2.5 GB

Figure 5. VM specifications

https://doi.org/10.56294/sctconf2024933

Performance metrics are compared between the suggested AF+MAES method and the GA-GEP, ACO, ESA+WSVMCVM, and MMH+EACO systems. Metrics including Mean Square Error (MSE) rate, energy usage, throughput, cost complexity, and time complexity are taken into consideration.

Time complexity

As the suggested approach runs more quickly, the system performs effectively.

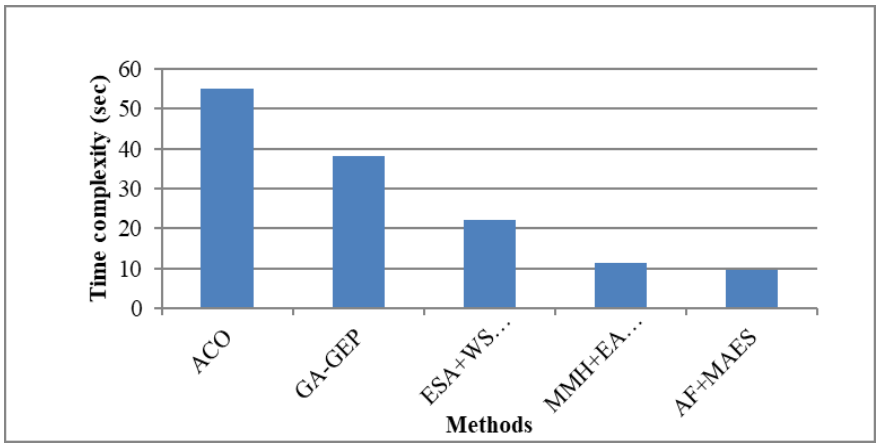

Figure 6. Time complexity

Figure 6 shows that the current and new methods are employed to assess the comparison measure by means of time complexity. The systems are represented on the x-axis, while the time complexity value is displayed on the y-axis. While the AF+MAES method offers reduced time complexity, GA-GEP, ACO, ESA+WSVMCVM, and MMH+EACO systems, provides increased time complexity. The MMH algorithm is employed in the recommended study project for load balancing, and the AF method is employed for resource allocation. Thus, the result concluded that the AF+MAES improves the hypervisor attack detection performance.

Cost complexity

Cloud is great when a recommended plan offers less expense and complexity.

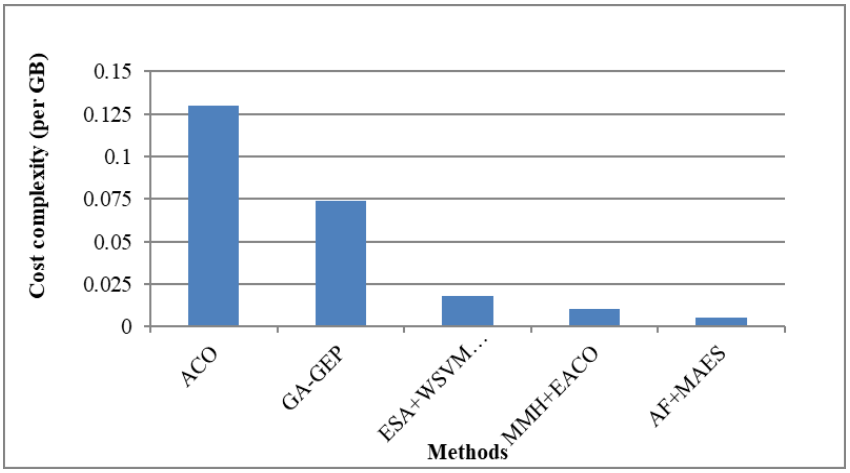

Figure 7. Cost complexity

It is evident from figure 7 that the cost complexity is evaluated utilizing the current and recommended approaches. The system are represented on the x-axis, while the cost complexity value is displayed on the y-axis. While the AF+MAES method provides low cost complexity, GA-GEP, ACO, ESA+WSVMCVM, and MMH+EACO systems serve increased cost complexity. The AF system is employed in the recommended study to allocate resources based on the best firefly values. Thus, the result concluded that the AF+MAES improves the hypervisor attack detection performance using effective encryption scheme.

https://doi.org/10.56294/sctconf2024933

Throughput

Throughput is the term employed to describe the speed at which data packets travel effectively over networks.

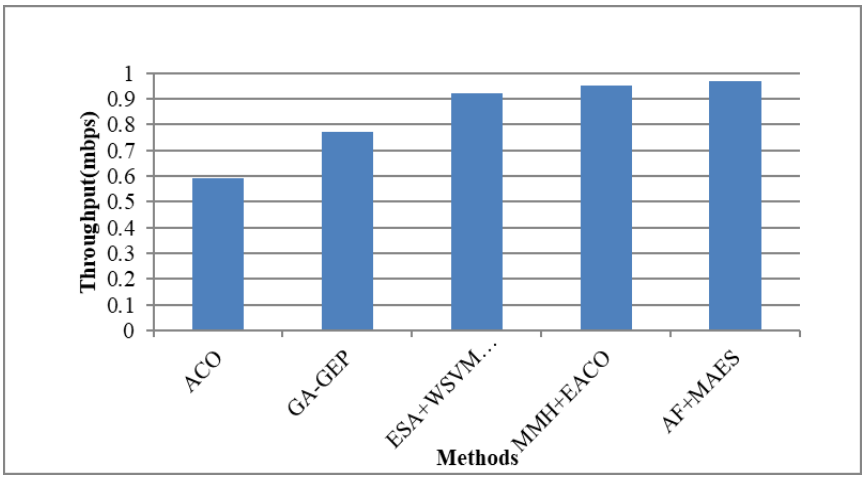

Figure 8. Throughput comparison

The evaluation of the GA-GEP, ACO, MMH+EACO, ESA+WSVMCVM, and AF+MAES approaches for the throughput metric is shown in figure 8. It demonstrates that while the suggested AF+MAES scheme offers higher throughput, the GA-GEP, ACO, ESA+WSVMCVM, and MMH+EACO approaches generate lesser throughput. The anticipated system improves the huge amount of data transmission speed by balancing efficient loads over cloud environment. AF algorithm is used to optimize the resource utilization and reduced the makespan time significantly.

Energy consumption

The term “energy consumption” describes the average quantity of energy needed over a given duration of time for a packet’s sending, receiving, or forwarding actions to a network node.

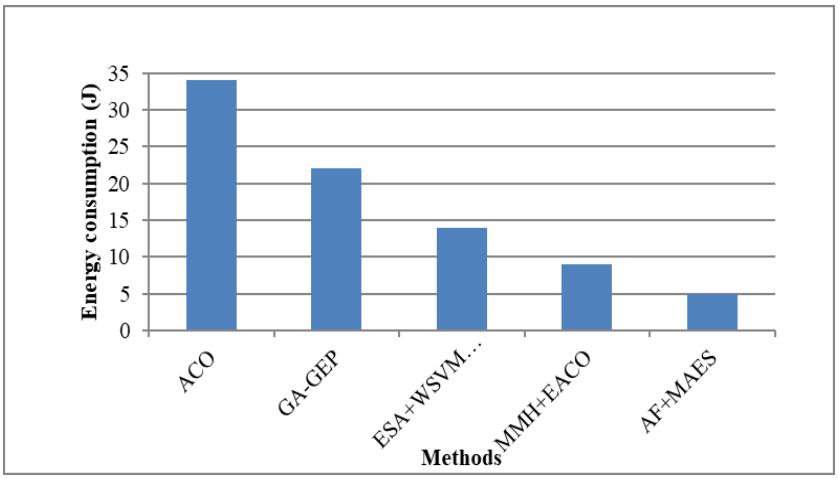

Figure 9. Energy consumption comparison

Figure 9 displays a evaluation of energy usage between the suggested AF+MAES technique and the GA-GEP, ACO, ESA+WSVMCVM, and MMH+EACO approaches. It demonstrates that while the suggested AF+MAES strategy offers lower energy consumption, the current approaches generate higher energy usage. With the creation of the optimal energy model, the suggested technique utilized less energy when transmitting huge amounts of data.

Mean Squared Error (MSE)

The MSE of a statistical estimator measures the average squared difference among the expected and actual values.

https://doi.org/10.56294/sctconf2024933

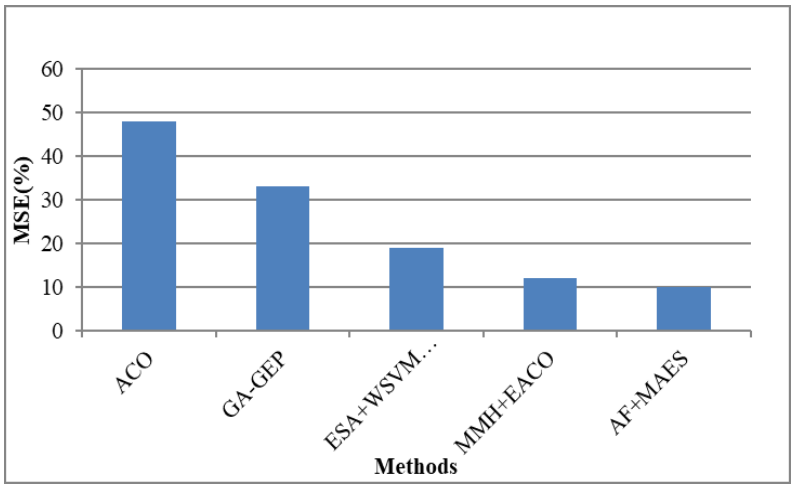

Figure 10. MSE

It is evident from figure 10 that the comparison metric is evaluated with respect to MSE utilizing the contemporary systems. The systems are represented on the x-axis, whereas the MSE value is displayed on the y-axis. The GA-GEP, ACO, ESA+WSVMCVM, and MMH+EACO approaches yield higher MSE rates than the suggested AF+MAES approach. Thus, the result concludes that the proposed AF+MAES approach increases the hypervisor attack detection performance

CONCLUSIONS

In this work, AF+MAES technique is suggested to identify and eliminate VMs and hypervisor threats in cloud. The AF technique is employed to allocate resources after the system model has been established. The best resources are chosen considering both local and global fitness values. Once the allocation procedure is finished, to imagine that hackers are attacking the hypervisor itself. While the hypervisor is under attack, the attackers are also targeting the VMs running beneath it. Then, utilizing a double key encryption method, the MAES method is suggested to significantly identify and eliminate VMs and hypervisor threats. Therefore, it utilized the MAES procedure to boost security. Ultimately, the analysis shows that compared to the current methods, the suggested AF+MAES method offers better security, throughput, and reduced energy usage, cost complexity, and time complexity. Future research can focus on developing efficient machine learning and soft computing algorithms to effectively address a variety of threats.

REFERENCES BIBLIOGRAPHIC

1. Zekri M, El Kafhali S, Aboutabit N, and Saadi Y. DDoS attack detection using machine learning techniques in cloud computing environments. In 3rd international conference of cloud computing technologies and applications (CloudTech), pp. 1-7. https://doi.org/10.1109/CloudTech.2017.8284731.

2. Rawashdeh A, Alkasassbeh M, and Al-Hawawreh M. An anomaly-based approach for DDoS attack detection in cloud environment. International Journal of Computer Applications in Technology, 57(4), pp.312-324. https://doi.org/10.1504/IJCAT.2018.093533.

3. Tsai JT, Fang JC, and Chou JH. Optimized task scheduling and resource allocation on cloud computing environment using improved differential evolution algorithm. Computers & Operations Research, 40(12), pp. 3045-3055. https://doi.org/10.1016/j.cor.2013.06.012.

4. Ding D, Fan X, Zhao Y, Kang K, Yin Q, and Zeng J. Q-learning based dynamic task scheduling for energy-efficient cloud computing. Future Generation Computer Systems, 108, pp. 361-371. https://doi.org/10.1016/j.future.2020.02.018.

5. Yakubu IZ, Musa ZA, Muhammed L, Ja'afaru B, Shittu F, and Matinja ZI. Service level agreement violation preventive task scheduling for quality of service delivery in cloud computing environment. Procedia Computer Science, 178, pp. 375-385. https://doi.org/10.1016/j.procs.2020.11.039.

6. Gamal M, Rizk R, Mahdi H, and Elnaghi BE. Osmotic bio-inspired load balancing algorithm in cloud computing. IEEE Access, 7, pp. 42735-42744. https://doi.org/10.1109/ACCESS.2019.2907615.

7. Chandrakala N, and Rao BT. Migration of Virtual Machine to improve the Security in Cloud Computing. International Journal of Electrical & Computer Engineering,(2088-8708) 8(1), pp. 210-219. http://doi.org/10.11591/ijece.v8i1.pp210-219.

8. Zhou, Z., Yu, J., Li, F. and Yang, F., 2018. Virtual machine migration algorithm for energy efficiency optimization in cloud computing. Concurrency and Computation: Practice and Experience, 30(24), pp. 1-10. https://doi.org/10.1002/cpe.4942.

9. Thiam C, and Thiam F. An energy-efficient VM migrations optimization in cloud data centers. In 2019 IEEE AFRICON, pp. 1-5. https://doi.org/10.1109/AFRICON46755.2019.9133776.

10. Adeshara N, Rede A, Jain S, Dhoot K, and Mhamane S. Optimizing Resource Utilization by Vm Migration Among Virtual Machines of a Cloud Server. In 5th International Conference on Communication and Electronics Systems (ICCES), pp. 671-677. https://doi.org/10.1109/ICCES48766.2020.9138010.

11. Saxena D, Gupta I, Kumar J, Singh AK, and Wen X. A secure and multiobjective virtual machine placement framework for cloud data center. IEEE Systems Journal, 16(2), pp. 3163-3174. https://doi.org/10.1109/JSYST.2021.3092521.

12. Hung LH, Wu CH, Tsai CH, and Huang HC. Migration-based load balance of virtual machine servers in cloud computing by load prediction using genetic-based methods. IEEE Access, 9, pp. 49760-49773. https://doi.org/10.1109/ACCESS.2021.3065170.

13. Nikolai J, and Wang Y. Hypervisor-based cloud intrusion detection system. In International Conference on Computing, Networking and Communications (ICNC), pp. 989-993. https://doi.org/10.1109/ICCNC.2014.6785472.

14. Kumara A, and Jaidhar CD. Hypervisor and virtual machine dependent Intrusion Detection and Prevention System for virtualized cloud environment. In 1st international conference on telematics and future generation networks (TAFGEN), pp. 28-33. https://doi.org/10.1109/TAFGEN.2015.7289570.

15. Dildar MS, Khan N, Abdullah JB, and Khan AS. Effective way to defend the hypervisor attacks in cloud computing. In 2017 2nd International Conference on Anti-Cyber Crimes (ICACC), pp. 154-159. https://doi.org/10.1109/Anti-Cybercrime.2017.7905282.

16. Aldribi A, Traoré I, Moa B, and Nwamuo O. Hypervisor-based cloud intrusion detection through online multivariate statistical change tracking. Computers & Security, 88, pp. 101646. https://doi.org/10.1016/j.cose.2019.101646.

17. Elshabka MA, Hassan HA, Sheta WM, and Harb HM. Security-aware dynamic VM consolidation. Egyptian Informatics Journal, 22(3), pp. 277-284. https://doi.org/10.1016/j.eij.2020.10.002.

18. Annadanam CS, Chapram S, and Ramesh T. Intermediate node selection for Scatter-Gather VM migration in cloud data center. Engineering Science and Technology, an International Journal, 23(5), pp. 989-997. https://doi.org/10.1016/j.jestch.2020.01.008.

19. Maipan-Uku JY, Muhammed A, Abdullah A, and Hussin M. Max-average: An extended max-min scheduling algorithm for grid computing environtment. Journal of Telecommunication, Electronic and Computer Engineering (JTEC), 8(6), pp. 43-47.

20. Elzeki OM, Reshad MZ, and Elsoud MA. Improved max-min algorithm in cloud computing. International Journal of Computer Applications, 50(12), pp. 22-27.

21. Annadanam CS, Chapram S, and Ramesh T. Intermediate node selection for Scatter-Gather VM migration in cloud data center. Engineering Science and Technology, an International Journal, 23(5), pp. 989-997. https://doi.org/10.1016/j.jestch.2020.01.008.

22. Liu J, Mao Y, Liu X, and Li Y. A dynamic adaptive firefly algorithm with globally orientation. Mathematics and Computers in Simulation, 174, pp. 76-101. https://doi.org/10.1016/j.matcom.2020.02.020.

23. Kaur G, and Kaur K. An adaptive firefly algorithm for load balancing in cloud computing. In Proceedings of Sixth International Conference on Soft Computing for Problem Solving: SocProS ,1, pp. 63-72. https://doi.org/10.1007/978-981-10-3322-3_7.

24. Szefer J, Keller E, Lee RB, and Rexford J. Eliminating the hypervisor attack surface for a more secure cloud. In Proceedings of the 18th ACM conference on Computer and communications security, pp. 401-412. https://doi.org/10.1145/2046707.2046754.

25. Pendli V, Pathuri M, Yandrathi S, and Razaque A. Improvising performance of advanced encryption standard algorithm. In second international conference on mobile and secure services (MobiSecServ), pp. 1-5. https://doi.org/10.1109/MOBISECSERV.2016.7440224.

FINANCING

The authors did not receive financing for the development of this research.

CONFLICT OF INTEREST

The authors declare that there is no conflict of interest.

AUTHORSHIP CONTRIBUTION

Conceptualization: Banu Priya MR.
Data curation: Maheswari D.
Formal analysis: Maheswari D.
Research: Banu Priya MR.
Methodology: Banu Priya MR.
Drafting - original draft: Maheswari D.
Writing - proofreading and editing: Banu Priya MR.