IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 10 April 2024, accepted 10 May 2024, date of publication 24 May 2024, date of current version 3 June 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3404949

RESEARCH ARTICLE

Interdependency Attack-Aware Secure and Performant Virtual Machine Allocation Policies With Low Attack Efficiency and Coverage

BERNARD OUSMANE SANE[1,2], (Member, IEEE), MANDICOU BA[1,3], DOUDOU FALL[2], YUZO TAENAKA[2], (Member, IEEE), IBRAHIMA NIANG[1], AND YOUKI KADOBAYASHI[2], (Member, IEEE)

[1]Laboratoire d'Informatique de Dakar (LID), Faculty of Science and Technology, University Cheikh Anta Diop of Dakar, Dakar 630-0101, Senegal
[2]Laboratory for Cyber Resilience, Nara Institute of Science and Technology, Ikoma, Nara 630-0192, Japan
[3]Ecole Supèrieure Polytechnique, Faculty of Science and Technology, University Cheikh Anta Diop of Dakar, Dakar 630-0101, Senegal

Corresponding author: Bernard Ousmane Sane (bernardousmane.sane@ucad.edu.sn)

This work was supported in part by Industrial Cyber Security Center of Excellence (ICSCoE) Core Human Resources Development Program, and in part by Japan Society for the Promotion of Science (JSPS) KAKENHI under Grant JP24K03045.

ABSTRACT Cloud computing has completely changed IT (information technology) by providing IT resources as services on the internet. However, certain types of attacks, such as interdependency attacks, impede its wide adoption. With the latter, an attacker who succeeds in compromising the VM of a user can traverse the hypervisor to launch an attack on the VM(s) of other users on the same hypervisor. Unfortunately, we note a lack of secure and performant allocation policies against this problem. Existing policies focus on security but ignore other factors, including workload balance and energy consumption, which are vital for commercial cloud platforms. In this context, we propose different allocation policies for choosing the datacenter server to which we allocate a new virtual machine. These policies aim to minimize the interdependence of different users' VMs while keeping the system performant regarding workload balance and/or power consumption. By default, our allocation policies treat all legitimate users as attackers and host their virtual machines according to their efficiency and coverage. We first design a secure and balanced solution that increases workload balance to prevent the servers from being overused. Afterward, we propose an algorithm that addresses security, power consumption, and workload balance objectives simultaneously. Based on our simulation results, our solutions perform better than existing algorithms regarding security, workload balance, and power consumption. The balanced solution reduces the chance of an attacker to zero and increases workload balance linearly. In other words, the workload balance is between [5, 35], and it utilizes slightly more hosts than existing proposals, with gains between [2, 8]. Although our final proposal is less secure than previous algorithms, it performs better, so it has a good workload balance ([5, 30]) and consumes less energy.

INDEX TERMS Virtual machine allocation, interdependency attack, security, workload balance, power consumption, hypervisor.

I. INTRODUCTION

Cloud computing is one of the most remarkable advances in IT in the last two decades. It offers resource consumption on demand, a flexible environment, and easy to use. These facilities make it widely adopted by the customers. However,

The associate editor coordinating the review of this manuscript and approving it for publication was Nitin Gupta.

in cloud computing, the hypervisor allows multiple virtual machines (VMs) of different users to run simultaneously on the same physical server. Ideally, each of these users' virtual machines should operate in isolation to maintain optimal security conditions. Unfortunately, perfect logical isolation has not been achieved in practice, leaving attackers with the possibility of launching attacks such as interdependency attacks, etc. [1], [2], [3], [4], [5], [6], [7]. With the interdependency attack, a malicious user who has compromised the VM(s) of a user i can traverse the hypervisor to launch an attack on the VMs of another user $j \neq i$ on the same hypervisor.

Hence, to tackle this issue, most of the proposals tried to satisfy security and/or performance constraints by using optimization methods such as heuristic algorithms [8], [9], [10], game theory approaches [3], [11], [12], [13], the multi-objectives optimization [14], [15], [16] since the problem is NP-hard [14]. However, we note the absence of a secure and performant virtual machine allocation technique against the interdependency problem. For instance, in [16], the authors proposed a secure solution against the interdependency attack that minimizes both attacker's *efficiency* and *coverage*, which respectively represent the probability of success of the attacker and the probability that the virtual machine of a legitimate user will be compromised. Nevertheless, this solution overlooked essential performance constraints related to minimizing power consumption and maximizing workload balance in the datacenter. These two performance constraints are very important for commercial cloud platforms. The first one motivates a provider to allocate a lot of VMs to fewer servers to reduce the cost of energy consumption and the emission of carbon dioxide (CO2). Significant energy consumption leads to high energy costs among providers. On the other hand, maximizing the *workload balance* spreads users' virtual machines among the servers to prevent the hosts from being over-utilized. To accommodate these two performance constraints essential for commercial cloud platforms, we propose extending [16]. We address the interdependency problem and the interdependency attack interchangeably throughout the paper. We also refer to the performance by workload balance and power consumption.

This paper extends our previous work [16]. We first give a complete overview of existing secure virtual machine allocation techniques. Then, we present three algorithms that determine the data center host to which we allocate a new VM to minimize the interdependency of different users' VMs while optimizing the power consumption and workload balance. By default, our allocation policies consider all legitimate users as attackers and then proceed to host the users' virtual machines to the server where their *efficiency* and/or *coverage* are the smallest. Our simulation results show that our allocation policies perform better than the existing works.

Contributions: This paper presents two novel allocation policies: *SALAEC-B* and *SPALAEC*, which advance secure virtual machine allocation in cloud environments. With these algorithms, the interdependency attacks, an underexplored vulnerability in virtual machine allocation, are addressed by modeling attacker behavior and optimizing resource allocation based on a game-theoretic approach. Indeed, in our previous work [16], we proposed a secure solution against the interdependency attack that minimizes both attacker's *efficiency* and *coverage*. However, this solution [16] ignored the performance constraints relative to the minimization of the power consumption and the maximization of the workload balance in the data center. Regarding these two distinct points, we need to design performant algorithms to increase the attacker's difficulties. Our main contributions are:

- A secure and balanced algorithm, called *SALAEC-Balanced* (*SALAEC-B*) which is an extension of our algorithm *SALAEC* [16]. It is a virtual machine allocation policy that minimizes the possibility of attackers who use a weak/vulnerable VM as a jumping-off point to attack other VMs within the same hypervisor management area. Moreover, to avoid violating the workload balance constraints, *SALAEC-B* does not allocate additional VMs to a particular host, even if the latter can still host more hosts. This favors the use of several servers, thus resulting in the dispersion of the virtual machines to prevent the servers from being overloaded. Hence, *SALAEC-B* is a solution that prevents the servers from being overused while maintaining security.
- We propose an algorithm that simultaneously tackles the security and performance constraints. Hence, this algorithm, named secure and performant allocation for low attacker's *efficiency* and *coverage (SPALAEC)*, is an improvement of *SALAEC-Balanced*. It decreases power consumption by using the least hosts without having an important negative impact on the workload balance and security. In *SPALAEC*, the failure of the host does not have an impact on all the users' VMs, unlike in existing works [14], [15]. It also prevents virtual machines that belong to the same user from launching interdependency attacks on each other.

Our simulation results show that by switching from the secure policy (*SALAEC*) to the secure and balanced policy (*SALAEC-B*), we kept the same level of security, unlike the solution in [14]. Besides, *SALAEC-B* performs better regarding the workload balance compared to a similar algorithm in [14]. Moreover, *SPALAEC* has a high workload balance performance while being secure. It also uses less energy compared to *SALAEC-B* and the solution in [14]. This work is compared with previous ones in Table 1.

The manuscript is organized as follows: A complete overview of existing secure virtual machine allocation techniques is presented in Section II. Then, we describe some fundamental concepts before defining the studied problem and the requirements in Section III. We present three algorithms for virtual machine allocation policies against the interdependency problem that optimize the constraints related

TABLE 1. Comparative Analysis of VM allocation Policies. High and Low criteria refer to good security performance and optimum power consumption, respectively.

Paper	VM allocation Policy	Security	Workload Balance	Power Consumption
[15]	PSSF-LEAST	High	Not Balanced	Medium
[14]	PSSF-Balanced	Medium	Balanced	Low
This paper	SALAEC-B	High	Balanced	Medium
Our previous paper [16]	SALAEC	High	Not Balanced	Low
This paper	SPALAEC	Medium	Balanced	Low

to security, power consumption, and workload balance in Section IV. We analyze our proposals in section V. Section VI discusses our proposals and their limitations. We conclude this paper with a conclusion in section VII.

II. RELATED WORK

The papers [8], [10], [17], [18], [19] used different virtual machine allocation techniques to satisfy performance constraints such as load-balancing and/or energy consumption.

The energy consumption at the data center level is very important. In [10], the authors focused on live migration while keeping the high quality of service to reduce energy consumption. They proposed an algorithm based on the history of the data used by the virtual machines. They split the dynamic consolidation problem into four sub-problems and improved a deterministic heuristic algorithm using historical VMs' data.

The work in [20], concerning security and privacy in the interconnection between autonomous devices, gave us an idea about the interdependency in cloud computing. Replacing the network nodes with the cloud's virtual machines proves this connection between virtual machines. In [3], the authors defined the interdependency between cloud users sharing the same hypervisor as an indirect attack. Hence, when an attacker wants to compromise the user i, he first compromises a vulnerable customer, then he proceeds to the hypervisor. Unfortunately, when the hypervisor is compromised, all users' virtual machines connected to it will be vulnerable. Additionally, if the attacker can launch the interdependency attack, then it can also try to measure the utilization of CPU caches in the server [5], [6]. However, to launch an interdependency attack, the attacker must first be a co-resident of the vulnerable customer. That means that the co-resident attacks solutions [4], [14], [15], [21], [22] allow to avoid the interdependency problem. Hence, Han et al. [15] proposed a secure VM management named the previous selected server first (*PSSF*). Where given a VM from a user, a server will be selected randomly, based on the *LEAST* algorithm or based on the *MOST* algorithm when the user does not yet have a VM in the datacenter. Otherwise, the server that already hosts a VM from the user will be selected. This solution ensured the security of users in the cloud by increasing the attacker's difficulties. However, in *PSSF*, a user could lose all her virtual machines when a server fails, and *PSSF* is not performant regarding workload balance. In [14], Han et al. extended their previous work in [15] to define a more secure and performant virtual machine allocation policy. In [16], the authors focused on the interdependency attack in co-resident environments and proposed a secure VM management that decreases as much as possible the attacker's *efficiency* and *coverage*.

Based on the interconnection between cloud users, other approaches use game theory methods for secure virtual machine allocation [3], [11], [12], [13]. In [3] and [11], the authors proposed a game model in the public cloud for studying the interdependency problem. Hence, they proved that the interdependency problem is a real problem in cloud computing. The lack of investment in the security of one user can harm other users on the same hypervisor [3]. They also defined theoretic virtual machine management based on the user's investment in security [11]. In [12], Han et al. evaluated the co-resident attack in public cloud computing. They introduced a secure game model to mitigate the users' risks. They also showed that the best way for the cloud provider to secure cloud users is to use pool policies and, given a virtual machine, select one of them randomly. An evaluation of the attacker's difficulty in achieving the interdependency problem is proposed in [13]. The authors analyzed the attacker's *efficiency* under four basic virtual machine allocation policies. Moreover, they showed that the Round Robin virtual machine allocation policy is unsuitable for the interdependency problem.

However, currently, there is no secure and performant virtual machine allocation solution against the interdependency attack. In [16], the authors tried to minimize the attacker's *efficiency* and *coverage*, but they did not focus on performance factors in the datacenter such as workload and power consumption. Thus, our main research question is: "How to improve [16] to attain secure virtual machine management against the interdependency problem while increasing the workload and decreasing the power consumption?"

A. A BRIEF OVERVIEW OF THE PROBLEM OF INTERDEPENDENCE

According to [20], security and privacy are explored in the interconnection of autonomous devices. Based on the results of this work, we can conclude that the interdependency in cloud computing is strongly influenced by its structure by replacing the network nodes with virtual machines. Our study focuses on the interdependency between cloud users who share the same hypervisor. Our previous paper [16] defined

it as follows: To compromise user i, the attacker must first place his virtual machine on the same server as his potential target, named user i. Then he goes through a vulnerable user named user j, who is another client on the same server and is easy to compromise(has a big potential loss). Afterward, he uses the latter, i.e., user j, to proliferate his attack on the hypervisor. As we know, if the hypervisor is compromised, all the virtual machines connected to it will be compromised [3]. Thus, through the vulnerable user (user j), the attacker has control over all virtual machines on the same hypervisor. Therefore, if there is a large difference in potential loss between users, then one user's security can impact another user, resulting in an interdependency problem [3]. To initiate an interdependency attack, the attacker can use a brute force strategy: start as many virtual machines as possible until he secures a good target (a user with whom the difference of loss is high). Referring to [3], interdependency strongly connects to side-channel attacks. Hence, if the attacker can launch the interdependency attack, then it can also attempt to measure the CPU cache utilization in the server [5], [6].

III. PROBLEM FORMULATION AND REQUIREMENTS

In this section, we first describe some fundamental concepts before defining the studied problem and the requirements for having secure and performant VM allocation policies against the interdependency problem. However, the scope of our previous paper [16] is extended beyond security to include performance considerations, resulting in criteria (7), (8), and (9). In this expansion, we offer metrics for assessing our solutions' performance.

A. NOTATIONS AND DEFINITIONS

Prior research has shown [3], [11] that disparities in potential loss among clients hosted on a single server facilitate interdependence attacks. Through mathematical evaluation, the factors outlined here are meant to assess attackers' success and legitimate users' vulnerabilities. For instance, using the equation 2 below, we can find the attacker's VMs that share the same hypervisor as the legitimate user.

We adopt the notations and definitions as in [15] and as in our previous papers [13], [16].

- Given a set of VMs and a set of servers, we define γ as how the virtual machines are distributed in the servers.
- $\mathcal{D}$: set of possible distributions.
- A_i and U_i designate respectively attacker i and legitimate user i.
- $VM(X_i^\gamma, \delta)$: a set of virtual machines under the distribution γ launched by the entity (attacker or legitimate user) X_i at time δ.
- $PL(vm)$: potential loss of the virtual machine vm. We define it as the amount of loss a user could suffer if one of his virtual machines were compromised. For example, companies managing sensitive data, such as banks, ministers of defense, health centers, etc., can be considered as having a high potential loss.
- $Target(A_i^\gamma)$: a set of virtual machines under the distribution γ started by a user U_i and targeted by the attacker.
- $PtVM(U_i^\gamma, \delta)$: user U_i's virtual machines under the distribution γ whose potential loss difference(s) with at least one virtual machine of an attacker A_i is quite high.
- $PtVM(A_i^\gamma, \delta)$: attacker A_i's virtual machines under the distribution γ whose potential loss difference(s) with at least one virtual machine of a user U_i is quite high.
- $Hyp(vm)$: hypervisor where the virtual machine vm is hosted.
- $IdepVM(A_i^\gamma, \delta)$: a subset of $VM(A_i^\gamma, \delta)$ that contains all the virtual machines that can launch the interdependency attack.
- $IdepVM(U_i^\gamma, \delta)$: a subset of $VM(U_i^\gamma, \delta)$ that contains all the virtual machines that are susceptible to be compromised by the interdependency attack.
- *Most secure host:* a host where the attacker's *efficiency* and *coverage* are equal to zero.
- *Semi-secure host*: a host where the attacker's *efficiency* and *coverage* are equal to 0.2 and 0.15, respectively.

We confirm that $IdepVM(A_i^\gamma, \delta)$ and $IdepVM(U_i^\gamma, \delta)$ are different. Indeed, we consider any user a potential attacker, but when he is launching a VM, This precision is important. In other words, at the instant δ, when a $user_i$ starts a virtual machine, he is considered an attacker. At the time $\delta + 1$, we assume its VMs are already allocated. This means these VMs will be protected at time $\delta + 1$. In other words, the $user_j$ who will launch his VM at time $\delta + 1$ should not be able to attack the VMs of $user_i$ (U_i). These $user_i$'s VMs which must be protected are estimated with $IdepVM(U_i^\gamma, \delta)$ instead $IdepVM(A_i^\gamma, \delta)$ (where $A_i = U_j$) which estimates among the $user_j$'s VMs launched at time $\delta + 1$, the one which can launch an interdependence attack.

1) SECURITY FACTORS

While the VMs distribution is γ, we have the attacker's *efficiency* and *coverage* defined as follows [12], [15], and [16]:

- Efficiency ($\mathcal{E}$): the probability of success of the attack when the time and the number of virtual machines started by the attacker decrease or increase.

$$\mathcal{E}(VM(A_i^\gamma, \delta)) = \frac{\#IdepVM(A_i^\gamma, \delta)}{\#VM(A_i^\gamma, \delta)} \tag{1}$$

where:

$$IdepVM(A_i^\gamma, \delta) = \{vm/vm \in PtVM(A_i^\gamma, \delta), \\ Hyp(vm) \subset \{Hyp(vm'), vm' \in Target(A_i^\gamma)\}\} \tag{2}$$

is the set of the attacker's virtual machines under the distribution γ that can launch an attack on at least one of the virtual machines of the legitimate users [3], [13].

$$PtVM(A_i^\gamma, \delta) = \{vm \mid vm \in VM(A_i^\gamma, \delta) \text{ and} \\ |PL(vm) - PL(vm')| \; and/or \\ |PL(vm) - PL(vm'')| \text{ quite high}$$

$$\text{with } (vm', vm'') \in (Target(A_i^\gamma), VM(A_i^\gamma, \delta))\} \quad (3)$$

- Coverage ($\mathcal{C}$): it gives an idea about how many of the legitimate users' virtual machines are vulnerable to the interdependency attack.

$$C(VM(U_i^\gamma, \delta)) = \frac{\#(IdepVM(U_i^\gamma, \delta))}{\#(VM(U_i^\gamma, \delta))} \quad (4)$$

where:

$$IdepVM(U_i^\gamma, \delta) = \{vm / vm \in PtVM(U_i^\gamma, \delta), Hyp(vm) \subset \{Hyp(vm'), vm' \in VM(A_i^\gamma, \delta)\}\} \quad (5)$$

is a set of the legitimate user's virtual machines under the distribution γ that are susceptible to be compromised by at least one of the virtual machines of the attacker [13].

$$PtVM(U_i^\gamma, \delta) = \{vm \mid vm \in VM(U_i^\gamma, \delta) \text{ and } |PL(vm) - PL(vm')| \; and/or \; |PL(vm) - PL(vm'')| \text{ quite high with } (vm', vm'') \in (VM(A_i^\gamma, \delta), VM(U_i^\gamma, \delta))\} \quad (6)$$

$PtVM(X_i^\gamma, t)$ allows us to find the virtual machines under the distribution γ that have a high difference of potential of loss. Unlike [13], we consider that the virtual machines coming from the same user must not be able to attack each other. That is why we redefine the $PtVM(X_i^\gamma, t)$. The $\#A$, cardinality of set A, is the total number of elements in A. For instance, $\#\{a, b, d, h, l\} = 5$. Note also that the attacker's *efficiency* and *coverage* are two dynamic factors since they depend both on the delay and on the number of virtual machines allocated. Also, In the coverage formula, we use $VM(U_i^\gamma, \delta)$, as an input instead of $VM(A_i^\gamma, \delta)$. In fact, with the coverage, we would like to estimate how many users' VMs are vulnerable to the interdependency attack. That is why we use $VM(U_i^\gamma, \delta)$ as the input variable where U_i designs a legitimate user.

2) PERFORMANCE FACTORS

The cloud service provider aims to find a VM allocation policy that reduces the attack's effectiveness and efficiency while keeping the system performing. Hence, we define the workload balance (W_k) and the power consumption at the host's level (P_k). However, we compute the sums ($\sum W_k$) and ($\sum P_k$) for having these metrics for the entire datacenter since a datacenter comprises several hosts.

- Workload balance (Wb): improves the Quality of Service (QoS) and reduces the cost. It can be defined as how the amount of processing is distributed at the host's level. In other words, it estimates how many times a server is selected. The formula is given in (7) where H_k is the k^{th} Host, λ_γ^k is the number of times that H_k is selected under the allocation policy γ and N is the total number of hosts [12].

$$Wb(\gamma) = \sum_{k=0}^{N} W_k = \sum_{k=0}^{N} \exp -(\frac{\lambda_\gamma^k}{10}) \quad (7)$$

- Usable Hosts (Uh): hosts used during the allocation. We define it as shown in (8), where α_γ is the number of selected hosts under the allocation policy γ and N is the total number of hosts available in the datacenter.

$$Uh(\gamma) = \frac{\alpha_\gamma}{N} \times 100 \quad (8)$$

- Power consumption (P): controlling energy consumption at the cloud computing level is crucial for cloud providers. However, among the main components (CPU (central processing unit), cooling unit, network interface, primary and secondary storage) in a data center that consumes electrical energy, the CPU consumes more electrical energy than the other components [10]. Hence, we provide a model for the power consumption of hosts that depends on the utilization of the critical system component (CPU). Therefore, we estimate the energy consumption by using a linear interpolation of the utilization change for a given time interval [10]. We adopt the definition from [23]. The formula is given in (9), where R_γ^k indicates the actual CPU proportion of the k^{th} host under the allocation policy γ, N is the total number of hosts, and $P_{\max}$ represents the maximum power that a host uses, and 70% of power defines the minimum percentage of power a Host uses, even when it is if in idle mode. We want to be in phase with our simulation platform CloudSim Plus [10].

$$P(\gamma) = \sum_{k=0}^{N} P_k = \sum_{k=0}^{N} (70\% \times P_{\max} + (1 - 70\%) \times P_{\max} \times R_\gamma^k) \quad (9)$$

- Gain Function: we define the gain function to quantify the exact value of gain between two allocation policies. The definition is given in (10) where X is the factor to evaluate, $\mathcal{A}_1$ and $\mathcal{A}_2$, two different algorithms.

$$\mathcal{G}_X = \frac{X_{\mathcal{A}_1} - X_{\mathcal{A}_2}}{X_{\mathcal{A}_1}} \times 100 \quad (10)$$

By the way, the security level (mentioned in the assumptions) will allow us to assess the potential loss of the VM. The efficiency and coverage use the latter, and the waiting time allows our algorithms to reach the secure server. Unlike the performance factors, we can remark that the security factors are expressed as a function of the waiting time. Therefore, the expected effect of the waiting time is more related to security than performance.

B. PROBLEM

We consider a cloud environment that runs on a virtualization technology named a hypervisor (Hyp) with n entities. We consider that the entity that manages the cloud computing resources (the provider) will act in good faith to guarantee the customers' security. On the other hand, the entity that engages the cloud provider in order to benefit from its services (the customer) will be our reference user in what follows. In other words, we will use "user" to refer to customers. Thus, each

user (U_i) can rent one or more virtual machines with different operating systems. However, the number of applications a user launches will not impact the model. When a user rents a virtual machine (*vm*), he can decide to pay (or not) an amount of *e* of investment in security. Using this investment in security for each virtual machine, we can determine the security level and the potential of loss ($PL(vm)$). Since the virtual machines are running under a hypervisor (such as Xen, VMware, or KVM), different users can share the same hypervisor. Hence, the platform is susceptible to being compromised by the interdependency attack [3]. This means that a malicious user who has compromised the VMs of a user *i* can traverse the hypervisor to launch an attack on the VMs of another user $j \neq i$ on the same hypervisor. In fact, in this situation, a lack of investment in the security of one user can impact the other users (bad neighborhood effect). Therefore, a malicious user can compromise all legitimate users on the same hypervisor. To quantify the attack's impact, we use two security factors: *efficiency* ($\mathcal{E}$) (1) and *coverage* ($\mathcal{C}$) (4). Also, to evaluate the performance of the cloud datacenter, we use the *workload balance* and the *power consumption*. So, the attacker's goal is to maximize these two security factors, while the cloud provider's aim is to define VM allocation policies that increase the attacker's difficulties while keeping the system performant. In other words, a secure and performant VM allocation policy should satisfy the following conditions at the same time:

- Minimize the attacker's possibilities as much as possible. In other words, the success of an attack, as minimal as it could be, must require the launch of many virtual machines. Therefore, a successful attack will require significant financial resources. On the other hand, increasing the number of virtual machines launched should not significantly impact the attack's efficiency. In this context, we seek to minimize *efficiency* and *coverage* (11).
- Keep the system performant by reducing the power consumption (12) and increasing the workload balance and distributing virtual machines to avoid the servers being over-utilized (13).

Hence, the problem can be formalized as follows: let $vmList = \{vm_1, \ldots, vm_n\}$ be a set of available virtual machines in a datacenter, $\mathcal{D}$ the set of all allocation possibilities. An allocation policy $\gamma \in \mathcal{D}$ is said to be secure and performant if:

$$\begin{aligned} \mathcal{E}(VM(A_i^{\gamma}, \delta)) &= \min_{\gamma \in \mathcal{D}} \mathcal{E}(VM(A_i^{\gamma}, \delta)) \\ \mathcal{C}(VM(U_i^{\gamma}, \delta)) &= \min_{\gamma \in \mathcal{D}} \mathcal{C}(VM(U_i^{\gamma}, \delta)) \qquad (11) \\ \mathcal{P}(\gamma) &= \min_{\gamma \in \mathcal{D}} \mathcal{P}(\gamma) \qquad (12) \\ \mathcal{W}b(\gamma) &= \max_{\gamma \in \mathcal{D}} \mathcal{W}b(\gamma) \qquad (13) \end{aligned}$$

The first conditions (11) mean that the distribution γ must be the distribution that most minimizes, at the same time, the attacker's *efficiency* and *coverage* compared to the other distribution $\gamma' \in \mathcal{D}$. The two last conditions signify that we cannot find another distribution $\gamma' \neq \gamma$ which best minimizes the *power consumption* or maximizes the workload balance in the datacenter.

C. ASSUMPTION

We make the following assumptions:

- If a user does not have a VM on the same hypervisor as the attacker, he will not suffer the consequences of the attack.
- Any user is considered as an attacker $\mathcal{A}_i$ when he starts a virtual machine at time *t*. However, "launching/starting a VM" does not mean "turning on a VM". We mean by "Launching a VM" a request from a client who wants to subscribe for a VM in the cloud provider.
- We assume that the attacker can try to launch an interdependency attack at any time.
- Given a user's virtual machine and his amount of investment in security, we can determine the security level and the potential loss ($PL(vm)$).
- Given a user's virtual machine, his amount of investment in security does not vary over time.
- We consider only suitable hosts (servers that have enough resources to host at least one VM) given a new virtual machine. In fact, given a virtual machine and its characteristics, we can recognize the list of hosts that have the capacity to host it. In each suitable host, we are looking for the top-ranked server so that the three above conditions (11), (12), and (13) can be satisfied.
- To minimize the power consumption, we use the straightforward method, which consists of reducing the number of running servers.

D. REQUIREMENTS

To propose a secure and performant solution against the interdependency problem, we should define a VM allocation policy that satisfies at the same time the previous conditions (11), (12) and (13). That means that our solution should:

- increase the attacker's difficulties by minimizing his *efficiency* and *coverage*.
- keep the system performant, by increasing the *workload* and decreasing the *power-consumption*.

However, satisfying these constraints is equivalent to solving an NP-hard combinatorial problem [8]. It includes both the knapsack problem (the security and the workload balancing constraints) and the bin packing problem (the energy consumption constraint) [15]. As a solution, we use the multi-objectives approach by minimizing the security metrics while treating the power consumption and the workload balance as constraints. In other words, when the choice arises, we will choose a secure server instead of a server with good power consumption and/or workload balance. This approach considers all legitimate users as attackers who attempt to hack the host's hypervisor, gain unauthorized privileges on the VMs it contains, and then proceed to host the users' virtual machines. Hence, we define allocation policies based

on the attacker's *efficiency* and *coverage* focusing on different objectives separately and simultaneously:

- The first allocation policy is the secure allocation for low attacker's *efficiency* and *coverage* (*SALAEC*). It focuses on the security and the power consumption (Algorithm 2).
- The second allocation is named *SALAEC-Balanced* (*SALAEC-B*), and it focuses on the security and the workload balance (Algorithm 3).
- The third allocation policy is the secure and performant allocation for low attacker's *efficiency* and *coverage* (*SPALAEC*). It focuses on security, power consumption, and workload balance simultaneously (Algorithm 4).

In the next section, we will provide more details about these algorithms.

IV. PROPOSED ALGORITHMS FOR MEETING SECURITY AND PERFORMANCE FACTORS

A summary of the proposed algorithms can be found in figure 1.

A. SECURE ALLOCATION FOR LOW ATTACKING EFFICIENCY AND COVERAGE

We start by only considering the security constraints defined in (11). To minimize the attacker's possibilities, we try to allocate any user's VM to a suitable host (a server that has enough resources to host at least one VM) where the security is optimal. Hence, we consider any user malicious when he starts his virtual machine. Then, we leverage the list of hosts with enough resources to host a new virtual machine and process to compute each host's *efficiency* and *coverage*. Finally, the new virtual machine will be hosted in a suitable host where the *efficiency/coverage* is the smallest. Hence, each VM will be allocated to the optimal security host. Given a virtual machine and a list of hosts, Algorithm 2 works as follows:

- We first consider the first suitable host on the list of hosts as the temporary secure server. Then, we evaluate the *efficiency* and the *coverage* by considering the VM's owner as an attacker and the users that already have VMs in the datacenter as the targets. When we reach the time when the latter is no longer suitable, we will look for another suitable host. This continuous process creates a loop (Algorithm 1).
- We consider the secure host as the host where the *efficiency/coverage* is the smallest, and this *efficiency/coverage* will be called "(min_e/min_c)". To find this secure host ("secHost"), we evaluate the *efficiency* and *coverage* (*tmp_e*, *tmp_c*) in all suitable remaining hosts again. To avoid checking all available hosts, we define sub-lists of hosts from the start index to the number of hosts to check. In other words, the checked hosts' size will equal the number of virtual machines currently and already allocated (Algorithm 1).
- Finally, the secure and suitable host is the one where the attacker's *efficiency* and *coverage* is the smallest. Additionally, if the secure host is not suitable, we remove it from the list of hosts and look for another one on the updated list of hosts. When it is "null" (not instantiated), the first free suitable host will be supposed as the secure host. Because the virtual machine will be alone inside the host, i.e., it cannot compromise other virtual machines (Algorithm 2).

Algorithm 1 Find Potential Secure Host for Virtual Machine

Input: A virtual machine *vm* launched by user U_i at time δ and a list of hosts available in the datacenter (HostList) that will be used by the allocation policy to place the virtual machine *vm*.

Output: Return the most secure host for the virtual machine

Initialization

```
A_i ← user i
min_e ← 1
min_c ← 1
secHost ← null
for each host in HostList do
    {Searching for first secure host}
  if (host is Suitable for vm) then
    min_e ← E(VM(A_i^γ, δ))
    min_c ← C(VM(U_i^γ, δ))
    secHost ← host
    break;
  end if
end for
{Due to performance concerns, we do not want to check all hosts, so we define a sublist that will vary based on the number of VMs}
if (the number of vms in the datacenter is higher than the number of available hosts) then
  NbrHostTocheck ← HostList.size ()
else
  NbrHostTocheck ← number of vms already hosted + 1
end if
for each host in HostList.subList (startIndex,NbrHostTocheck) do
  if (host is Suitable for vm and host! =secHost) then
    tmp_e ← E(VM(A_i^γ, δ))
    tmp_c ← C(VM(U_i^γ, δ))
    min_e←min(min_e, tmp_e)
    min_c ← min(min_c, tmp_c)
    secHost ← HostList.getHost (min_e,min_c)
  else
    startIndex ← startIndex +1
  end if
end for
return secHost
```

Besides, we consider efficiency as the most important factor compared to the coverage since it works directly with the attacker's VMs, unlike the coverage where the

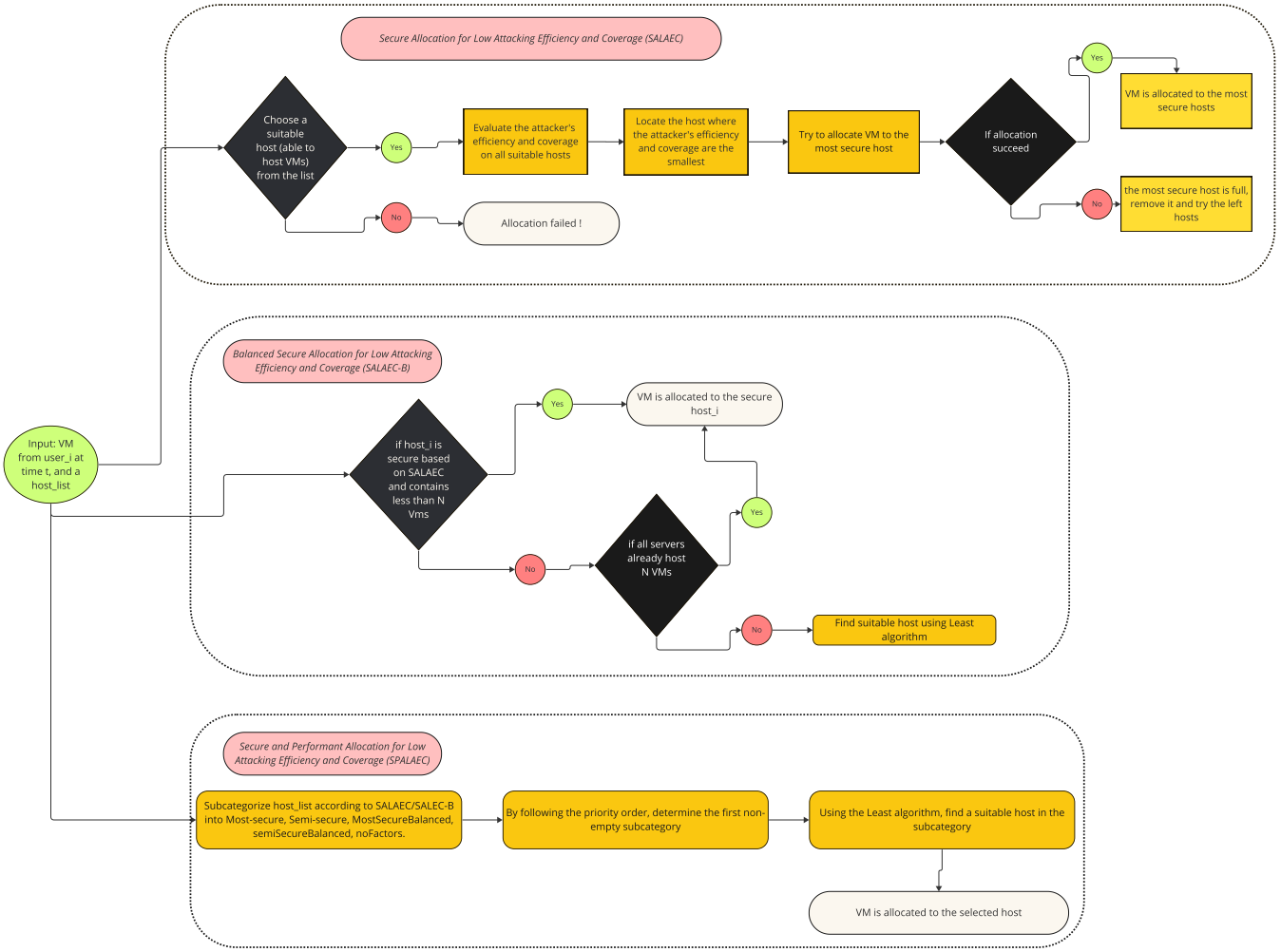


FIGURE 1. Proposed virtual allocation policy workflow.

input parameters are the VMs of the legitimate user hence, as mentioned in the original paper [16], we first used the attacker's efficiency for searching the secure host. However, by using the attacker's *efficiency* we can meet the situation where the efficiency is the same when a virtual machine is supposed to be in $host_i$ and also when it is supposed to be in $host_j$. In this case, we will use the coverage to decide which host to choose. More explanations, details, and an example can be found in the original paper [16].

B. POWER CONSUMPTION-AWARE SECURE ALLOCATION FOR LOW ATTACKER'S EFFICIENCY AND COVERAGE

We know that the *MOST-POLICY* (an algorithm that allocates a new virtual machine to a suitable host that contains more VMs) performs better in terms of power consumption compared to the *LEAST-POLICY* (where a new virtual machine will be allocated to a suitable host that has the least VMs) [12]. For the reason that the *MOST-POLICY* uses fewer servers because it tries to allocate a new virtual machine to a server that has more virtual machines until that server reaches its full capacity. However, we remark that *SALAEC* (Algorithm 2) adopts the same philosophy by avoiding launching new servers in the following ways:

- When the attacker's *efficiency* and *coverage* are the same in two servers, we chose the server that has the largest number of virtual machines. This allows us to use between $30-48\%$ less hosts as shown in our previous paper [16].
- There is no limit to the number of virtual machines per server. That means a server can host a new virtual machine if it has enough resources.
- A server without a virtual machine will be turned off.

Hence, we propose *SALAEC* as a candidate when the constraints relative to the security (11) and the power consumption (12) are considered.

C. BALANCED SECURE ALLOCATION FOR LOW ATTACKER'S EFFICIENCY AND COVERAGE

In this subsection, we define an algorithm that will focus on the security and workload constraints by slightly modifying *SALAEC*:

- We prevent a server from being overused by defining a limited number of virtual machines (N) per server.

Algorithm 2 Secure Allocation for Low Attacking Efficiency and Coverage (SALAEC)

Input:
A virtual machine vm launched by user U_i at time δ and a list of hosts available in the datacenter (HostList), that will be used by the allocation policy to place the virtual machine *vm*.

Output:
Return the potential secure and suitable host for the virtual machine

Initialization

```
HostList ← getHostList ()
secHost←FindPotentialSecHostForVm(HostList, vm)
while IndexMostSecHost ()< HostList.size () do
    if (secHost! =null and secHost is Suitable for vm) then
        secHost.add (vm)
        return secHost
    else
        if secHost!=null && secHost is not Suitable for vm then
            HostList.remove (secHost)
            secHost←FindPotentialSecHostForVm (HostList, vm)
        else
            IndexMostSecHost← IndexMostSecHost () + 1
            if (IndexMostSecHost() <HostList.size()) then
                secHost← HostList.get (IndexMostSecHost())

                return secHost
            end if
        end if
        IndexMostSecHost(HostList.indexOf(secHost))
    end if
end while
```

Hence, we assume that no server can host more than N virtual machines from any users even if it has enough remaining resources.

- We prevent the VMs from the same user from being at the same host by using *efficiency*/*coverage* as criteria for allocation. Otherwise, if the host crashes, the user will lose his virtual machines.

The Algorithm 3 works as follows:

- We use the *SALAEC* algorithm to verify the host's security. Then, we check some constraints relative to the workload balance. Hence,
  - If a secure host contains less than N virtual machines, it will be considered as a secure and balanced host.
  - Otherwise, we check if all servers already have N virtual machines. If yes, the virtual machine can be hosted at the current secure host even if it has N virtual machines.
  - Additionally, if the secure host and some other hosts (not all hosts) contain N virtual machines. Then, the host will be considered secure but not balanced. In this case, we will use the *LEAST* algorithm to find the host with the least virtual machines. We chose the *LEAST* algorithm based on its performance about the workload. Additionally, it performs better concerning security compared to the *MOST* algorithm and the *RANDOM* algorithm [12]. When we use the *MOST* algorithm, we will probably overuse some servers, which favors the co-location between the attacker and the legitimate user. On the other hand, choosing a server randomly among several servers with a different level of security is not helpful. For the reason that among the chosen servers some of them may already have the attacker's VM(s). Moreover, most of the time, the "least server" will be free (contains no VM), which is crucial for the security and the workload at the same time.

D. SECURE AND PERFORMANT ALLOCATION FOR LOW ATTACKING EFFICIENCY AND COVERAGE (SPALAEC)

In this section, we consider the security, the power consumption, and the workload balance constraints at the same time. Hence, we introduce the *SPALAEC* algorithm that optimizes these three factors as follows:

- Security: *SPALAEC* considers any user as a potential attacker when he launches his virtual machine. Then, it allocates the latter's virtual machine to the host where his *efficiency* and *coverage* are the smallest. Additionally, as the allocated host will be selected from a list of hosts using the *LEAST* algorithm, we contend that the hosts inside the lists are homogeneous (they have the same level of security). Hence, we define *Most-secure* host as the host where the attacker's *efficiency* and *coverage* are both equal to zero. A host is *Semi-secure* when the attacker's *efficiency* and *coverage* are in the intervals]0, 0.2] and]0, 0.15], respectively.
  These values are from the simulation results of "*PSSF-Balanced*" [14], which is considered secure enough when the attacker's *efficiency* and *coverage* is equal to 0.2 and 0.15, respectively. The allocated host can be insecure without this homogeneity due to utilizing the *LEAST* algorithm.
- Workload balance: The workload balance aims to prevent a server from being overused. Hence, we define a number N of virtual machines per server to solve that issue. That threshold will help us to distribute the users' virtual machines across the servers. It also does not harm security since if the number of virtual machines per host is limited, then the number of virtual machines able to launch attacks will be small.
- Power consumption: we should use as few hosts as possible to decrease the power consumption. But, using the least hosts can harm the workload balance and

Algorithm 3 Balanced Secure Allocation for Low Attacking Efficiency and Coverage (SALAEC-B)

Input:
A virtual machine VM launched by user U_i at time δ and a list of hosts available in the datacenter (HostList), that will be used by the allocation policy to place the virtual machine *vm*.

Output:
Return the most secure and balanced suitable host for the virtual machine

Initialization

allocatedHost ← null
for host in HostList **do**
  if host is secure based on *SALAEC* and it contains less than N virtual machines) **then**
    allocatedHost ← secHost
  else if all servers already host N virtual machines **then**
    allocatedHost ← secHost
  else
    allocatedHost ← FindLeastSuitableHost (vm, δ, HostList)
  end if
end for
return allocatedHost

the security. To avoid this negativity, we ensure our VM allocation policy does not launch new free hosts and select ''insecure'' hosts. Hence, the allocated host will be chosen among a group of balanced *Most-secure/Semi-secure* not-empty hosts (hosts that contain at least one virtual machine). Or it will be from a group of not balanced *Most-secure/Semi-secure* not-empty hosts. Otherwise, it will be chosen from a list containing some empty hosts. As you can remark, we are always looking for secure hosts that have fewer virtual machines among non-empty hosts while respecting the limit of N virtual machines per server.

Algorithm 4 works as follows:

- It considers any user as an attacker when he launches his virtual machine.
- Then, it sorts the list of available hosts from the most secure to the least secure by using *SALAEC*.
- Finally, the workload balance constraints will be checked based on *SALAEC-B*. The allocated host will be chosen based on the *LEAST* algorithm among a list of hosts such as each host is balanced and the attacker's *efficiency* and *coverage* are null, no host is balanced and the attacker's *efficiency* and *coverage* are null, each host is balanced and *Semi-secure* host or, not balanced *Semi-secure* or, each host is simple; a host which we ignore his security and performance level.

E. COMPLEXITY ANALYSIS

We consider computational complexity. In fact, the complexity of space is a concern when using devices with low memory, in contrast to cloud computing. The complexity of *SALAEC*, *SALAEC-B* and *SPALAEC* are polynomials which is synonymous with ''feasible'' and ''efficient''.

- The complexity of *SALAEC* is equal to $1+p^c+p(\theta(1)+p^c) = 1 + p + p^c + p^{c+1} \approx \theta(p^k)$, k constant, p be the total number of hosts. In fact, the complexity of $\mathcal{E}(VM(A_i^{\gamma}, \delta))$ and $\mathcal{C}(VM(U_i^{\gamma}, \delta))$ are both equal to $2p^2$ as shown in [16]. In Algorithm 1, the total number of operations regarding the instruction ($A_i \leftarrow$ user i) and the first loop is $1 + p(\theta(1) + 2p^2) = 1+p+2p^3$. After the first loop, we have some elementary operations ($\theta(1)$) and another loop with $p(\theta(1) + 2p^2) = p + 2p^3$ operations. The total number of operations for Algorithm 1 is

Algorithm 4 Secure and Performant Allocation for Low Attacking Efficiency and Coverage (SPALAEC)

Input:
A virtual machine vm launched by user U_i at time δ and a list of hosts available in the datacenter (HostList), that will be used by the allocation policy to place the virtual machine *vm*.

Output:
Return the most secure and balanced suitable host for the virtual machine

Initialization

secureBalancedHostList ← null
securenobalancedHostList ← null
mostSecureBalancedHostList ← null
semiSecureBalancedHostList ← null
mostSecHostList ← null
semiSecHostList ← null
noFactorsHostList ← null
allocatedHost ← null
potentialAllocatedHostList ← null
for host in HostList **do**
  Categorize it based on *SALAEC* using ''Most-secure'', ''Semi-secure'', ''noFactors''
  sort it according to (mostSecHostList, semiSecHostList, noFactorsHostList)
  if host is balanced based on *SALAEC-B* **then**
    sort it according to (mostSecureBalancedHostList, semiSecureBalancedHostList)
  end if
end for
potentialallocatedHostList ←determine the first non-empty list by following the priority order as follows: mostSecureBalancedHostList, semiSecureBalancedHostList, MostSecHostList, semiSecHostList, noFactorsHostList
if ! (potentialAllocatedHostList.isEmpty ()) **then**
  allocatedHost←FindLeastSuitableHost(vm, δ, potentialallocatedHostList)
end if
return allocatedHost

$1+p+2p^3+p+2p^3 = 1+2p+4p^3$. The complexity is $\theta(p^c)$, where c is a constant. In Algorithm 2, the number of operations is $1+p^c$ where p^c is given by the function **FindPotentialSecHostForVm**. IndexMostSecHost()< hostlist.size() will be executed p times. Inside the loop "**while**", the number of operations is $\theta(1)+p^c$. Hence, the complexity is $1+p^c+p(\theta(1)+p^c) = 1+p+p^c+p^{c+1} \approx \theta(p^k)$, k constant.

- The complexity of *SALAEC-B* is polynomial and equal to $\approx \theta(p^{k+1})$, k constant. In fact, in *SALAEC-B*, the Algorithm 2 (*SALAEC*) will be executed as many times as we have a host. Since the number of hosts is equal to p, then the number of operations in the loop is equal to $p(\theta(p^k)+\theta(1))$, where $\theta(1)$ is the complexity of the elementary operations inside the loop.
- The complexity of *SPALAEC* is polynomial and equal to $\approx \theta(p^{k+2})+\theta(p^{k+1})$, k constant. In fact, in *SPALAEC*, *SALAEC* and *SALAEC-B* will be executed p times then the number of operations in the loop is equal to $p(\theta(p^k)+\theta(p^{k+1})+\theta(1))$, where $\theta(1)$ is the complexity of the elementary operations inside the loop.

V. EVALUATION

In this section, we perform the analysis of the resource allocation policies to evaluate the security of virtual machines and the performance in the datacenters. Hence, we consider different allocation policies such as *PSSF-LEAST* [15], and *PSSF-BALANCED* [14], because they are close to our work and are recent proposals concerning secure and/or performant virtual machine allocation policies. We also consider *SALAEC* [16], *SALAEC-B* and *SPALAEC*. Hence, we present the attacker's *efficiency* and *coverage*, and the datacenter performances such as the power consumption and the workload-balance under these VM allocation policies by using CloudSim Plus.

A. SIMULATION SETTINGS

As a simulator, we use CloudSim Plus [10], which is an open-source project developed in the Java programming language. It provides a flexible environment where it is possible to test all cloud services. We compare our different algorithms with *PSSF-LEAST* [15] and *PSSF-Balanced* [14] by using the factors such as the *efficiency* $\mathcal{E}$, the *coverage* $\mathcal{C}$, the workload balance $\mathcal{W}_b$, the power consumption $\mathcal{P}$, and the Usable hosts $\mathcal{U}_h$. Using inheritance, we can define the potential loss criterion as a feature for each virtual machine. Also, for each virtual machine, we attribute a random score between 0 and 10 as its security level. Note that in real cloud computing, based on the amount of investment in security, the provider can define the security level. We consider that the difference in loss between the two virtual machines is high when it is greater than or equal to 4. The different devices that we use in our simulation and the configuration parameters are given in Table 2. The previous works [14] and [15] used exactly the same settings as us.

TABLE 2. Devices configuration.

	Type	Quantity	Parameters	Charac-teristics	Values
Datacenter	1	1	Arch	x86	N/A
			OS	LINUX	
			VM	XEN	
Host	1	150	Cores	N/A	16
			MIPS		2600
			RAM		24576
			Storage		128000
Virtual machine	1	Random	Cores	N/A	1
			VMM		XEN
			MIPS		2500
			RAM		870
			Storage		10000
Virtual machine	2	Random	Cores	N/A	1
			VMM		XEN
			MIPS		2000
			RAM		1740
			Storage		10000
Virtual machine	3	Random	Cores	N/A	1
			VMM		XEN
			MIPS		1000
			RAM		1740
			Storage		10000
Virtual machine	4	Random	Cores	N/A	1
			VMM		XEN
			MIPS		600
			RAM		513
			Storage		10000

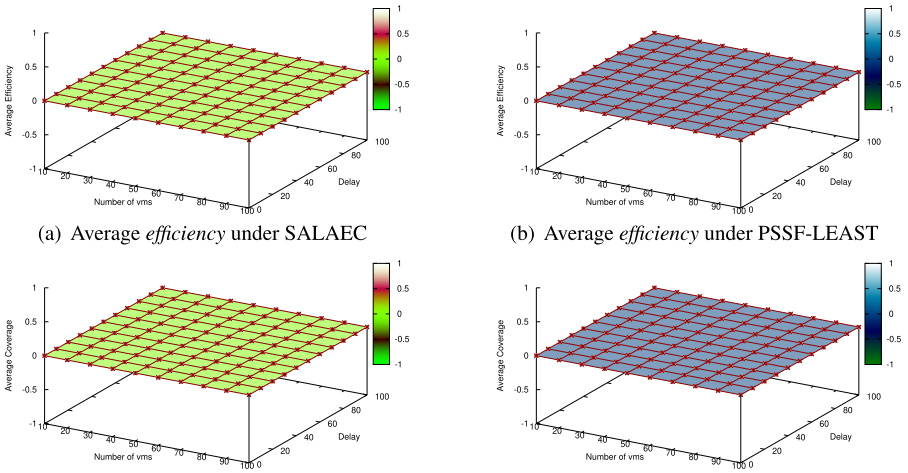
For the simulation, we note δ the waiting time between the start of the virtual machines of the legitimate user and those of the attacker [12]. This delay varies between 0 to 100. The simulation works as follows: the legitimate user starts 25 virtual machines; after a waiting time, $\delta = t$, the attacker starts 10 virtual machines. This operation is repeated 100 times. Then, we pass to the scenario where the virtual machines of the legitimate user are equal to 25, $\delta = t$, and the virtual machines of the attacker are 20; this is repeated 100 times. We continue until the virtual machines of the legitimate user are 25, $\delta = t$, and the virtual machines of the attacker are 100. We conduct the previous procedure for all $\delta \in \{0, 10, \ldots, 100\}$ [16].

Assumptions: The attackers can decide when to launch their virtual machines and the number of virtual machines to launch.

B. PERFORMANCE EVALUATION AND DISCUSSION

In Fig. 2, we only consider the security constraint, and we evaluate the attacker's *efficiency* and *coverage* under *SALAEC* and *PSSF-LEAST* [15]. In Figs. 3 and 4, we consider the security and workload balance, then we evaluate respectively the attacker's *efficiency* (Fig. 3(a), Fig. 3(b)), *coverage* (Fig. 3(c), Fig. 3(d)), the workload balance Fig. 4(b), Fig. 4(a)). Also, we compute the gain of workload balance (Fig. 4(c)) between *SALAEC-B* and *PSSF-Balanced*. Finally, we evaluate the attacker's *efficiency* (Fig. 6(a)), *coverage* (Fig. 6(b)), the workload balance (Fig. 6(c)), the power consumption (Fig. 6(d)) according to the variation of $VM(A_i^{\gamma}, \delta)$ when all security and performance factors are considered.

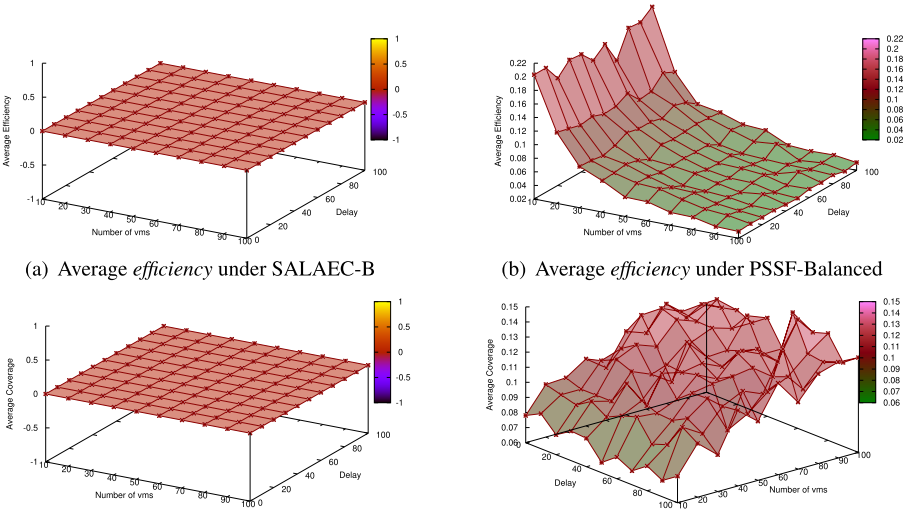
We note that the attacker's *efficiency* and *coverage* are reduced to zero under *SALAEC* and *PSSF-LEAST* as shown in



(a) Average *efficiency* under SALAEC (b) Average *efficiency* under PSSF-LEAST

(c) Average *coverage* under SALAEC (d) Average *coverage* under PSSF-LEAST

FIGURE 2. Average *efficiency* and *coverage* according to the number of virtual machines and the delay under different VM allocation policies that only consider security factors.



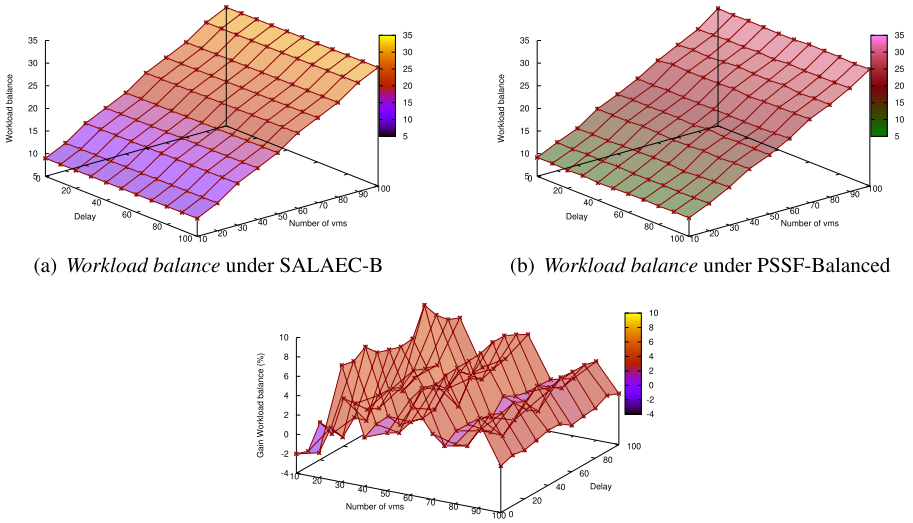
(a) Average *efficiency* under SALAEC-B (b) Average *efficiency* under PSSF-Balanced

(c) Average *coverage* under SALAEC-B (d) Average *coverage* under PSSF-Balanced

FIGURE 3. Average *efficiency* and *coverage* according to the number of virtual machines and the delay under different VM allocation policies that only consider security and workload balance.

Fig. 2. This means that the virtual machines are secure under these two policies. The reason is that *SALAEC* considers any user as a potential attacker, and then, it looks for a host where the security factors are the smallest. This process guarantees that a server will only host virtual machines with the same security level. Hence, the probability of launching an interdependency attack will be too low. Concerning *PSSF-LEAST*, the attacker's possibilities are null because they use dedicated hosts. That means that the virtual machines from the same user will share the same host. Nevertheless, the situation can have a negative impact when such kind of host goes down, unlike *SALAEC*, where the allocation is done by using the attacker's *efficiency* and *coverage*. Therefore, the VMs from the same user are mostly distributed among the hosts.

As shown in Fig. 3, when we pass from *SALAEC* to *SALAEC-B*, we do not lose on security. In fact, *SALAEC-B* is *SALAEC* with a limited number of virtual machines per server. Hence, as the number of virtual machines per server decreases, the probability of co-location decreases, i.e., it becomes difficult to launch interdependency attacks. However, in Han et al. [14], from *PSSF-LEAST* to *PSSF-Balanced*, the attacker's *efficiency* and *coverage* are not null as shown in Fig. 3(b) and Fig. 3(d). The reason is that for a new user (for example, the attacker), his virtual machines will be allocated to the servers chosen randomly from a group of servers. That means if the servers inside the group already host some users' virtual machines, the attackers can share the same host with some legitimate users to launch interdependency attacks. Unfortunately, this group of servers



(a) *Workload balance* under SALAEC-B

(b) *Workload balance* under PSSF-Balanced

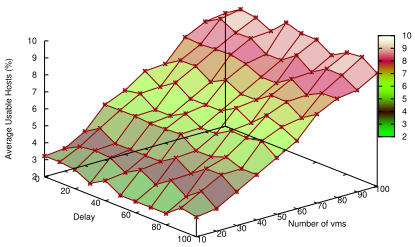
(c) Average *Gain Workload balance* between SALAEC-B and PSSF-Balanced

FIGURE 4. ***Workload balance*** **according to the number of virtual machines and the delay under different VM allocation policies that only consider security and workload balance.**

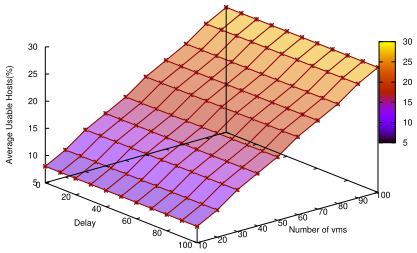
often already contains virtual machines from other users. Switching from one group of servers to another will be possible only if the servers of the first group are all fully used [14]. In this context, the random choice is favorable to the attacker. This justifies the non-null values of the attacker's *efficiency* and *coverage*. However, there is a linear decrease of the attacker's *efficiency*. This decrease is due to the passage of the other groups of hosts. Indeed, once the first virtual machines of the attacker fill a group of hosts, those remaining will be the first to be hosted in new groups. As they will be alone there, the probability that they will launch attacks decreases.

Concerning the *workload balance*, it is represented mathematically by a decreasing function depending on the number of times that a server is selected. This means that the more a server is solicited, the more its workload balance decreases. So, when we increase the workload balance, we reduce the possibility of a server being overloaded. Hence, from *SALAEC* to *SALAEC-B* (Fig. 4(a)) and from *PSSF-LEAST* to *PSSF-Balanced* (Fig. 4(b)), we note a linear increase of the workload balance (for space reasons, the workload balance graphs under *SALAEC* and *PSSF-LEAST* are omitted). More precisely, the workload balance is between $[1.5, 6.5]$ under *SALAEC* and $[5, 35]$ under *SALAEC-B*. It is is between $[4, 16]$ under *PSSF-LEAST* and $[5, 35]$ under *PSSF-Balanced*. These variations are due to the limit of the number of virtual machines per server. This limit favors the dispersion of the virtual machines among several servers to avoid over-utilization. As a result, a server will be selected just a few times. Consequently, the number of usable hosts experience a linear increase by switching from non-balanced policies (*SALAEC (Fig. 5(a)), PSSF-LEAST (Fig. 5(c))*) to balanced policies (*SALAEC-B (Fig. 5(b)), PSSF-Balanced (Fig. 5(d))*). Hence, it is between $[2, 10]$ and $[5, 30]$ under *SALAEC* and *SALAEC-B*, respectively. It is equal to $[4, 18]$ and $[6, 28]$ under *PSSF-LEAST* and *PSSF-Balanced*, respectively. As you can remark, *SALAEC-B* uses slightly more hosts than *PSSF-Balanced* with a gain that varies between $[-2, 8]$ (Fig. 5(e)). We can remark that we don't even use 30% of the servers yet. Moreover, 30% servers mean we have allocated all 125 VMS using 45 servers. So, indirectly, we remain efficient even with more VMs than servers. Since *SALAEC-B* uses more hosts compared to *PSSF-Balanced*, *SALAEC-B* performs better than *PSSF-Balanced* regarding the workload balance as shown in Fig. 4(c). In fact, the gain between *SALAEC-B* and *PSSF-Balanced* is equal to $[-4, 10]$. The negative values represent the few times that *SALAEC-B* has a bad workload compared to *PSSF-Balanced*, and they are represented by the few purple colors in Fig. 4(c).

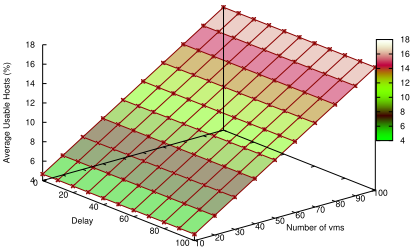
In Fig. 6, we take into account the security (Fig. 6(a), Fig. 6(b)), the workload balance (Fig. 6(c)) and the power consumption (Fig. 6(d)). We observe that the attacker's *efficiency* and *coverage* are not null. These two metrics oscillate between $[0.16, 0.3]$ and $[0.15, 0.5]$ therefore in terms of security *PSSF-Balanced* and *SALAEC-B* are better than *SPALEC*. Indeed, *SPALAEC* tries to minimize power consumption, so it will try to use fewer hosts. Consequently, it promotes co-location and, thus, the interdependence attack (which increases the attacker's efficiency and coverage), unlike SALAEC-B and PSSF-balanced. In other words, this was expected since performance has a cost on security. Besides, we aim to ensure that the virtual machines from the same account cannot attack each other. Unfortunately, this increases the number of target virtual machines and, therefore, increases the coverage. Other factors are also added, such as the limit N of virtual machines per server.



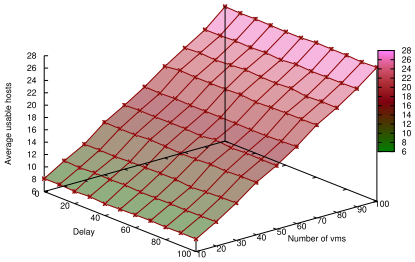
(a) Average *Usable Hosts* under SALAEC



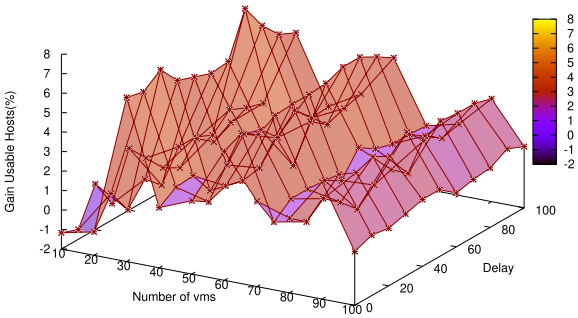
(b) Average *Usable Hosts* under SALAEC-B



(c) Average *Usable Hosts* under PSSF-LEAST



(d) Average *Usable Hosts* under PSSF-Balanced



(e) *Gain Usable Host* between SALAEC-B and PSSF-Balanced

FIGURE 5. ***Usable Host*** **according to the number of virtual machines and the delay under different VM allocation policies that only consider security and workload balance.**

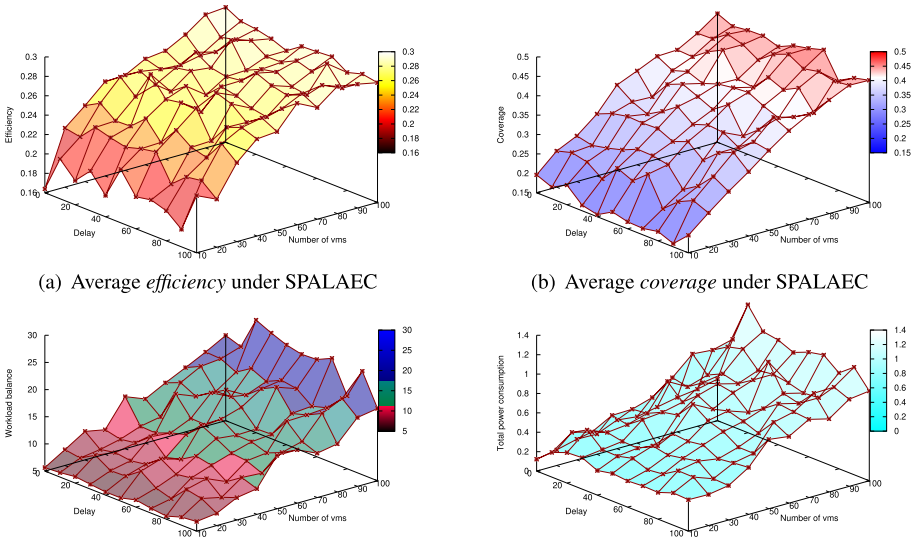
Indeed, with this limit, a secure server that already contains N virtual machines will not be chosen. This results in choosing a less secure server instead. On the other hand, the reduction in the level of security favors the performance. Indeed, the workload balance has a good variation ($[5, 30]$). This variation is almost equal to that of an algorithm that is only balanced, such as *PSSF-Balanced*, *SALAEC-B*. Also, the power consumption is controlled since the servers consume less compared to *PSSF-Balanced* and *SALAEC-B*. An evaluation of the gain of power consumption shows that *SPALAEC* uses less energy with a gain equal to $[15, 55]$, $[20, 50]$ compared to *PSSF-Balanced* and *SALAEC-B*, respectively.

However, the evaluation shows that if we remove the condition that the virtual machines from the same account cannot attack each other, then the attacker's possibilities will be reduced to zero while keeping the same performance relative to the workload balance and the power consumption.

SALAEC-B and *SPALAEC* are two innovative allocation policies that go beyond security considerations by addressing critical factors such as power consumption and workload balance, reflecting a holistic approach tailored to real-world deployments in which multiple demands compete for resources. With the foundation of game theory and their polynomial time complexity, these algorithms provide scalable solutions that can be deployed in large-scale cloud environments.

VI. DISCUSSIONS AND LIMITATIONS

One cloud providers' biggest challenge is identifying a malicious user from an honest user. In a sense, our virtual machine allocation policies consider any user a potential attacker when he launches his virtual machine. This consideration allows us to ignore the process of distinction between an attacker and a legitimate user. Because if all the users are considered attackers, the ''real attacker'' is not spared. Indeed, those who can do more can do less. Furthermore, even if the attacker uses many accounts, it will not impact the efficiency of his attack. On the other hand, at the level of commercial cloud platforms (Example: Amazon Web Services (AWS)), the best practice requires that the services (including the creation of virtual machines) be under the responsibility of an administrator account. Thus, on behalf of a big company (with several different user departments), the administrator must create several virtual machines and give roles to the users called upon to use them. The same is true on behalf of a service provider (different from a cloud service provider),



(a) Average *efficiency* under SPALAEC

(b) Average *coverage* under SPALAEC

(c) *workload balance* under SPALAEC

(d) *power consumption* under SPALAEC

FIGURE 6. **Average *efficiency, coverage, workload balance and power consumption* according to the number of virtual machines and the delay under different VM allocation policies that consider security, workload balance and power consumption.**

who, in turn, can create servers for different small and medium businesses. Thus, we note that the risks are that the users of different departments or small and medium-sized businesses who are competing by using servers from the same administrator account may attack each other. Indeed, our best friend can become our worst enemy. To remove this insecurity, our virtual machine allocation policies ensure that virtual machines from the same account should not be able to attack each other. In other words, for any virtual machine launched from an account named X, the virtual machines already hosted (including those coming from the account X) are all considered targets.

We chose to offer different algorithms. The reason is that we want to give a palette of choices to the cloud service providers. Hence, depending on the requirements of the applications to be hosted, they can choose only a secure allocation policy, whatever the performance constraints are, or choose an allocation that is secure with a good workload balance and/or good power consumption. Moreover, our allocation policies did not consider the migration of the virtual machines. That means once a virtual machine is hosted on a server, it will not move to another server. However, to avoid a server being overused, which may result in the violation of the service level agreement (SLA). We fixed a limit of virtual machines per server. We set the limit based on the total number of virtual machines per server rather than the number of users' virtual machines per server. The last case allows X users to have N machines per server. So, on a given server, we can find XN virtual machines, unlike in the first case, where the total number of virtual machines rarely exceeds N. The first case also favors the dispersion of a user's virtual machines. Hence, when a server goes down the possibility that the user loses all his virtual machines is limited.

In AWS Leadership Principles [24], they said that "they work vigorously to earn and keep customer trust." For us, that means that it is better to increase the workload balance to satisfy the users than to reduce the power consumption to help cloud service providers save money. For this reason, *SPALAEC* focuses on workload balance and power consumption while giving more priority to the workload balance.

However, any scientific work has limits. We observe that *SPALEC* provides better performance in terms of workload balance and power consumption. However, *SALEC-B* and *PSSF-BALANCED* beat *SPALEC* in terms of security. Obviously, we could reduce the efficiency and coverage by lifting certain constraints, assuming that only VMs belonging to different customers can attack each other. But as you know, we have a strong constraint that requires that VMs from the same account cannot attack each other. Even if VM_a and VM_b belong to client X, our algorithm must protect VM_a from damage caused by VM_b.

We note that considering any user as an attacker and avoiding attacks from the same user's account require significant resources for a good implementation of the solutions. Indeed, the allocation of a virtual machine needs several processes. This consequence is not negligible, given the large number of cloud computing users. Nonetheless, data centers have enormous computing capacities. Also, finding the best host is equivalent to solving an NP-hard combinatorial problem [8]. Therefore, we tried to find an optimal solution. The only limit we found regarding the efficiency and the coverage is that they need to be calculated for each user's VMs. However, the complexities of our algorithms are polynomials (as shown in the sub-section IV-E), which means that they are easier and faster to compute. On the other hand, we made an arbitrary choice on the difference in potential of loss (which

is considered high if it is equal to 4) between two virtual machines based on their investment in security. However, we wanted to show that we can label the security risks of a virtual machine based on the associated account. From this labeling, the provider will be able to define its own threshold of difference of potential of loss between virtual machines. In addition, we use a straightforward method by reducing the number of running servers, unlike energy-aware virtual machine allocation techniques (a) [25], [26], [27]. Moreover, our allocation policies do not consider the possibility of migrating a virtual machine from one server to another. However, it is a critical process for alleviating servers. It can also play an important role when a server can no longer run an application due to insufficient resources (b). Thus, (a), (b), and other simulation scenarios with many users may be the subjects of a study in our future work. Additionally, the simulation settings are the same as in previous works [14] and [15]. Nevertheless, we are yet to use 30% of the servers. In addition, 30% servers mean we have allocated all 125 VMS to 45 servers. So, indirectly, we remain efficient even with more VMs than servers. As part of future research, we will evaluate the algorithms from various aspects by using more virtual machines than hosts.

This work only considers attacks between users sharing the same hypervisor. However, other types of attacks have not been considered and are closely related to this work. For example, with Rowhammer exploits, malicious code is executed on a vulnerable system to compromise the machine's services (web browsers, cloud services). The resulting analysis is that the attacker needs a privilege that gives him the right to execute code in this type of exploit. In addition, the code exploits only compromise the victim's virtual machine's services (other network machines are spared). Therefore, other works like in [28] try to change the context of the attack via the network. Indeed, since the virtual machines are connected by a network then launching an attack on other machines of the network can be acquired. So the paper [28] shows that from a remote machine, a malicious user can trigger and exploit Rowhammer bit flips directly by only sending network packets. Our solution does not take into account this type of attack, the reason is that we focus on the security of the allocation policies and not on the security of the subnets. However, this type of attack can be considered in our future work. Also, our algorithms use the amount of security investment during allocation. So, this amount of security investment does not vary over time. However, as future works, we can address the latter by defining a dynamic game where the attacker and legitimate users can change strategies (such as the investment in security) at any time. We cannot tackle all existing attacks at the same time. Indeed, the vulnerabilities of virtual machines can be explored in different ways via the network (Flooding Attacks (DDoS), Metadata Spoofing Attacks, Rowhammer attacks over RDMA-enabled networks, etc.), hosts (cross VM side-channel attacks, VM creation attacks, VM scheduler based attacks, VM migration, and rollback attacks, VM Hopping, VM Escape, etc.), applications (Malware injection, Steganography attacks, Web services & Protocol based attacks, etc.) and information security policy (Contracts and Electronic Discovery, Laws and Regulations, Audit Assurance, Information leakage, Vendor Lock-in, Identity Management, etc.) [7].

VII. CONCLUSION

In this paper, we develop the first secure and performant solution against the interdependency attack between cloud users sharing the same hypervisor. It focuses on minimizing security metrics while considering power consumption and workload balance. This approach considers all legitimate users as attackers who attempt to hack the host's hypervisor and gain unauthorized privileges on the VMs it contains. Specifically, we define a secure allocation policy that maximizes workload balance (*SALAEC-B*) and a secure and performant allocation policy that simultaneously optimizes security, workload balance, and power consumption (*SPALAEC*). We also show that these solutions are optimal with polynomial complexities synonymous with "feasible" and "efficient". In addition, results from the simulation show that *SALAEC-B* is secure and balanced, and it performs better than its counterpart in the related work, *PSSF-Balanced* [14]. Finally, *SPALAEC* is also secure against the interdependency attack while being efficient regarding workload balance and power consumption. Furthermore, our VM allocation policies prevent the negative impact that can be caused by the failure of one of the servers, unlike in *PSSF-LEAST* [15] and *PSSF-Balanced* [14]. Our allocation policies do not consider the possibility of migrating a virtual machine from one server to another. We propose an energy-aware approach with virtual machine migration as future work to deal with the high energy consumption in cloud computing and service level agreements (SLAs) violations. In addition, during allocation, our algorithms use the amount of security investment, which does not change over time (i). Our future work may investigate (i) and other simulation scenarios with many users.

REFERENCES

[1] T. Velte, A. Velte, and R. Elsenpeter, *Cloud Computing, A Practical Approach*, 1st ed. New York, NY, USA: McGraw-Hill, 2009.

[2] A. Jasti, P. Shah, R. Nagaraj, and R. Pendse, "Security in multi-tenancy cloud," in *Proc. 44th Annu. 2010 IEEE Int. Carnahan Conf. Secur. Technol.*, Oct. 2010, pp. 35–41.

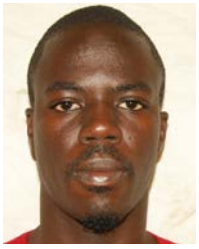
[3] C. A. Kamhoua, L. Kwiat, K. A. Kwiat, J. S. Park, M. Zhao, and M. Rodriguez, "Game theoretic modeling of security and interdependency in a public cloud," in *Proc. IEEE 7th Int. Conf. Cloud Comput.*, Jun. 2014, pp. 514–521.

[4] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds," in *Proc. 16th ACM Conf. Comput. Commun. Secur.*, Nov. 2009, pp. 199–212.

[5] Y. Yarom and N. Benger, "Recovering openssl ecdsa nonces using the flush+reload cache side-channel attack," *IACR Cryptol. ePrint Arch.*, vol. 2014, p. 140, Jul. 2014.

[6] Y. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Cross-VM side channels and their use to extract private keys," in *Proc. ACM Conf. Comput. Commun. Secur.*, Oct. 2012, pp. 305–316.

[7] B. O. Sane, I. Niang, and D. Fall, "A review of virtualization, hypervisor and VM allocation security: Threats, vulnerabilities, and countermeasures," in *Proc. Int. Conf. Comput. Sci. Comput. Intell. (CSCI)*, Dec. 2018, pp. 1317–1322.
[8] J. Xu and J. A. B. Fortes, "Multi-objective virtual machine placement in virtualized data center environments," in *Proc. IEEE/ACM Int. Conf. Green Comput. Commun. Int. Conf. Cyber, Phys. Social Comput.*, Dec. 2010, pp. 179–188.
[9] J. F. Frenzel, "Genetic algorithms," *IEEE Potentials*, vol. 12, no. 3, pp. 21–24, Oct. 1993.
[10] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers," *Concurrency Comput., Pract. Exper.*, vol. 24, no. 13, pp. 137–1420, Sep. 2012.
[11] L. Kwiat, C. A. Kamhoua, K. A. Kwiat, J. Tang, and A. P. Martin, "Security-aware virtual machine allocation in the cloud: A game theoretic approach," in *Proc. 8th IEEE Int. Conf. Cloud Comput.*, Jul. 2015, pp. 556–563.
[12] Y. Han, T. Alpcan, J. Chan, and C. Leckie, "Security games for virtual machine allocation in cloud computing," in *Lecture Notes in Computer Science*. Cham, Switzerland: Springer, 2013, pp. 99–118.
[13] S. B. Ousmane, B. C. S. Mbacke, and N. Ibrahima, "A game theoretic approach for virtual machine allocation security in cloud computing," in *Proc. 2nd Int. Conf. Netw., Inf. Syst. Secur.*, Mar. 2019, p. 47.
[14] Y. Han, J. Chan, T. Alpcan, and C. Leckie, "Using virtual machine allocation policies to defend against co-resident attacks in cloud computing," *IEEE Trans. Dependable Secur. Comput.*, vol. 14, no. 1, pp. 95–108, Jan. 2017.
[15] Y. Han, J. Chan, T. Alpcan, and C. Leckie, "Virtual machine allocation policies against co-resident attacks in cloud computing," in *Proc. IEEE Int. Conf. Commun. (ICC)*. IEEE, 2014, pp. 786–792.
[16] B. O. Sane, M. Ba, D. Fall, S. Kashihara, Y. Taenaka, I. Niang, and Y. Kadobayashi, "Solving the interdependency problem: A secure virtual machine allocation method relying on the attacker's efficiency and coverage," in *Proc. 20th IEEE/ACM Int. Symp. Cluster, Cloud Internet Comput.*, May 2020, pp. 440–449.
[17] P. Graubner, M. Schmidt, and B. Freisleben, "Energy-efficient management of virtual machines in eucalyptus," in *Proc. IEEE 4th Int. Conf. Cloud Comput.*, Jul. 2011, pp. 243–250.
[18] H. Zhao and W. Chenyu, "A dynamic dispatching method of resource based on particle swarm optimization for cloud computing environment," in *Proc. 10th Web Inf. Syst. Appl. Conf.*, Nov. 2013, pp. 351–354.
[19] W. Lin, J. Z. Wang, C. Liang, and D. Qi, "A threshold-based dynamic resource allocation scheme for cloud computing," *Proc. Eng.*, vol. 23, pp. 695–703, Oct. 2011.
[20] C. A. Kamhoua, N. Pissinou, and K. Makki, "Game theoretic modeling and evolution of trust in autonomous multi-hop networks: Application to network security and privacy," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2011, pp. 1–6.
[21] S. Jin, J. Ahn, S. Cha, and J. Huh, "Architectural support for secure virtualization under a vulnerable hypervisor," in *Proc. 44th Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Dec. 2011, pp. 272–283.
[22] J. Szefer, E. Keller, R. B. Lee, and J. Rexford, "Eliminating the hypervisor attack surface for a more secure cloud," in *Proc. 18th ACM Conf. Comput. Commun. Secur.*, NY, NY, USA, Oct. 2011, pp. 401–412.
[23] H. Jia, X. Liu, X. Di, H. Qi, L. Cong, J. Li, and H. Yang, "Security strategy for virtual machine allocation in cloud computing," *Proc. Comput. Sci.*, vol. 147, pp. 140–144, Sep. 2019.
[24] (2020). *Amazon's Leadership Principles*. [Online]. Available: https://aws.amazon.com/careers/culture/
[25] R. Yadav, W. Zhang, K. Li, C. Liu, M. Shafiq, and N. K. Karn, "An adaptive heuristic for managing energy consumption and overloaded hosts in a cloud data center," *Wireless Netw.*, vol. 26, no. 3, pp. 1905–1919, Apr. 2020.
[26] R. Yadav and W. Zhang, "MeReg: Managing energy-SLA tradeoff for green mobile cloud computing," *Wireless Commun. Mobile Comput.*, vol. 2017, pp. 1–11, Jun. 2017.
[27] R. Yadav, W. Zhang, H. Chen, and T. Guo, "MuMs: Energy-aware VM selection scheme for cloud data center," in *Proc. 28th Int. Workshop Database Expert Syst. Appl. (DEXA)*, Aug. 2017, pp. 132–136.
[28] A. Tatar, R. K. Konoth, E. Athanasopoulos, C. Giuffrida, H. Bos, and K. Razavi, "Throwhammer: Rowhammer attacks over the network and defenses," in *Proc. Annu. Tech. Conf.*, Jul. 2018, pp. 213–226.



BERNARD OUSMANE SANE (Member, IEEE) received the M.E. degree in data transmission and information security and the Ph.D. degree in computer science from the University Cheikh Anta Diop of Dakar, Senegal, in 2022. He was a Special Research Student with Nara Institute of Science and Technology (NAIST). He is currently a Project Assistant Professor with Keio University, Japan. His research interests include cloud computing security, game theory security, quantum error correction, quantum internet, and quantum cryptography.

MANDICOU BA received the Ph.D. degree in computer science from the University of Reims Champagne-Ardenne, France, in 2014. He is currently an Assistant Professor with Ecole Supérieure Polytechnique, University Cheikh Anta Diop of Dakar, Senegal. His research interests include network security, the IoT security, cloud computing security, self-stabilization, clustering, ad hoc and wireless sensor networks, deep learning, and IA.

DOUDOU FALL received the M.E. degree in data transmission and information security from the University Cheikh Anta Diop of Dakar, Senegal, in 2009, and the M.E. and Ph.D. degrees in information science from Nara Institute of Science and Technology (NAIST), Japan, in 2012 and 2015, respectively. He is currently an Assistant Professor with the Division of Information Science, NAIST. His research interests include cloud computing security, the IoT security, blockchain security, vulnerability, and security risk analysis.

YUZO TAENAKA (Member, IEEE) received the D.E. degree in information science from Nara Institute of Science and Technology (NAIST), Japan, in 2010. He was an Assistant Professor with The University of Tokyo, Japan, and has been an Associate Professor with the Laboratory for Cyber Resilience, NAIST, since April 2018. His research interests include information networks, cybersecurity, distributed systems, and software-defined technology.

IBRAHIMA NIANG received the Ph.D. degree in computer science from the University of Paris V Descartes, in 2002. He obtained the rank of a Full Professor with the University Cheikh Anta Diop of Dakar, Senegal, in 2018. His research interests include quality of service (QoS) management, mobility and optimization of networks and systems, cloud-edge computing systems, and the Internet of Things.

YOUKI KADOBAYASHI (Member, IEEE) received the Ph.D. degree in computer science from Osaka University, Japan, in 1997. He is currently a Professor with the Laboratory for Cyber Resilience, Nara Institute of Science and Technology, Japan. His research interests include cybersecurity, web security, and distributed systems. He is a member of ACM and the IEEE Communications Society.