1-2019

Secure virtual machine placement in cloud data centers

Amit AGARWAL

Nguyen Binh Duong TA
Singapore Management University, donta@smu.edu.sg

Published in Future Generation Computer Systems, 2019 November, Volume 100, Pages 210-222
https://doi.org/10.1016/j.future.2019.05.005

Secure virtual machine placement in cloud data centers

Amit Agarwal [a,*], Ta Nguyen Binh Duong [b]

[a] *BITS Pilani, Goa, India*
[b] *Nanyang Technological University, Singapore*

HIGHLIGHTS

- New, realistic metrics defined for assessing secure VM placements.
- A new placement scheme having better security and resource-efficiency.
- User classification improves security in VM placement.

ARTICLE INFO

Article history:
Received 1 November 2018
Received in revised form 24 February 2019
Accepted 1 May 2019
Available online 21 May 2019

Keywords:
Data centers
Cloud security
Co-location attacks
Virtual machine placement

ABSTRACT

Due to an increasing number of avenues for conducting cross-VM side-channel attacks, the security of multi-tenant public IaaS cloud environments is a growing concern. These attacks allow an adversary to steal private information from a target user whose VM instance is co-located with that of the adversary. In this paper, we focus on secure VM placement algorithms which a cloud provider can use for the automatic enforcement of security against such co-location based attacks. To do so, we first establish a metric for evaluating and quantifying co-location security of multi-tenant public IaaS clouds, and then propose a novel VM placement algorithm called "Previously Co-Located Users First" which aims to reduce the probability of malicious VM co-location. Thereafter, we perform a theoretical and empirical analysis of our proposed algorithm to evaluate its efficiency and security. Our results, obtained using real-world cloud traces containing millions of VM requests and thousands of actual users, indicate that the proposed algorithm provides a significant increase in the cloud's co-location resistance with little compromise in resource utilization, compared to existing approaches. We also explore the potential for cloud providers to leverage passive cache monitoring techniques as an additional security measure in order to automatically improve the co-location resistance provided by general VM placement algorithms.

1. Introduction

With the advent of multi-tenant public IaaS cloud computing services, users can now request instances as per their specific requirements on demand. This type of service has benefits for both cloud users and cloud providers. For cloud users, it obviates the need for them to buy and maintain their own computing hardware. For cloud providers, it enables them to generate revenue by efficiently renting out such services to the users. In IaaS clouds, the service is provided in the form of Virtual Machines (VMs) which are instantiated by the cloud provider to run on one of many Physical Machines (PMs) that the cloud provider owns. The assignment of these VMs to PMs is done by the cloud provider using an appropriate placement algorithm. These algorithms are usually designed with the aim of scheduling the incoming VM requests on PMs in a way that maximizes the resource utilization of the data center. The task of creating VMs and their resource management is done using a hypervisor which runs on each PM. The hypervisor multiplexes the resources (e.g. cores, memory etc.) of a PM across multiple VMs which are running on that PM. At the same time, it is the task of the hypervisor to ensure that a strong isolation of shared resources is provided between different VMs which are running on the same PM so that each VM's private content is inaccessible to other co-located VMs.

Although these multi-tenant public cloud computing environments have proved to be extremely useful in the industry, it has its own caveats. The same strategy which allows a cloud provider to efficiently rent out services, by multiplexing the shared physical infrastructure among multiple cloud users, also becomes a breeding ground for a class of attacks known as co-location attacks. These attacks exploit the shared nature of public cloud infrastructure and enable a rogue VM to extract information from other benign VMs which are running on the same PM using

* Corresponding author.
E-mail addresses: f20140403@goa.bits-pilani.ac.in (A. Agarwal), donta@ntu.edu.sg (T.N.B. Duong).

side-channels present in shared cloud resources (e.g. Last Level Cache).

One of the most interesting works addressing such attacks in multi-tenant public clouds was by Ristenpart et al. [1]. In this work, the authors conducted a comprehensive empirical study on the feasibility of such attacks in Amazon EC2 cloud services. They demonstrated ways to (i) create a map of physical cloud infrastructure using network probes (ii) determine whether two VMs are co-located (resident on the same PM) (iii) launch VMs with the aim of co-locating them with specific target VMs (iv) extract data from co-located VMs using side-channel attacks. Zhang et al. [2] demonstrated the feasibility of side-channel attacks to extract critical data such as private keys. Wu et al. [3] designed and implemented a high-bandwidth (over 700 bps) covert channel by leveraging the use of memory bus.

In the light of such developments in side-channel attack vectors and newly discovered vulnerabilities such as Meltdown [4] and Spectre [5], it has become equally important to design new defense measures which the cloud providers can use to combat these attacks. It is important to note that an essential prerequisite for any such attack is physical co-location, i.e. a malicious user first needs to successfully co-locate his VM on the same PM as that of a benign user's VM. In a cloud environment, the assignment of VMs to PMs is solely controlled by the cloud provider using a placement strategy. The cloud provider can leverage such a strategy to make placement decision which not only optimizes the utilization of cloud resources but also guarantees some level of security against such co-location based attacks. Also, as some of the recent works have indicated [6–8], it might be possible for the cloud provider to detect side-channel attacks in some scenarios. This kind of detection capability can be leveraged by the cloud provider to gain prior knowledge about the class of cloud users (benign or malicious) based on their past activities and use that information for automatically reinforcing the security provided by placement algorithms. With this idea in mind, we make the following contributions in this paper:

- We describe a metric called "Co-Location Resistance" for evaluating and quantifying the security of multi-tenant public IaaS cloud against co-location based attacks. The metric is a generalization (to account for multiple cloud users) of a metric earlier proposed by Azar et al. [9].
- We describe "Secure VM Placement" (SVP) problem having dual-objective of maximizing co-location resistance and resource utilization of the cloud and then propose a new algorithm called "Previously Co-Located Users First" as a possible solution to the SVP problem. This algorithm can be utilized by cloud providers for the automatic enforcement of security in multi-tenant public IaaS clouds.
- We theoretically analyze the expected co-location resistance that is provided by our algorithm and compare it with the empirically obtained results.
- We perform an extensive empirical analysis of our proposed algorithm with other standard and secure placement algorithms using the recently released trace of workloads from Microsoft Azure [10].
- We explore the potential for cloud providers to leverage passive cache monitoring techniques and binary classifiers in order to improve the security provided by general VM placement algorithms.

The rest of the paper is organized in the following manner: Section 2 describes past works on combating co-location based attacks. In Section 3, we state our assumptions and threat model, define the evaluation metrics, and formulate the placement problem. We then describe our proposed VM placement algorithm and perform a theoretical analysis in Section 4. In Section 5, we present a comparative empirical analysis of our proposed strategy and other placement algorithms. Finally, in Section 6, we conclude the paper and provide some directions for future research work.

2. Background and related work

Recent works [11,12] have focused on increasing the power-efficiency and thereby reducing the operational cost of data centers. However, the problem of increasing resource efficiency while maintaining co-location security is still a major issue. A straightforward way to mitigate side-channel and other co-location based attacks is to allocate a dedicate PM to every cloud user. Although this would nullify the chances of co-location based attacks, it would also drastically affect the resource utilization of a cloud data center. Allocating a dedicated PM for every cloud user would result in a high number of idle cores on live PMs, which in turn would significantly increase the energy cost of the data center. Therefore, it is necessary to devise defense strategies which can combat co-location based attacks without significantly increasing the cost of running a data center.

In this section, we will review some of the defenses against co-location based attacks that have been proposed over the years and their respective pros and cons. Most of the ideas can be broadly classified into two main categories: (i) Reducing information leakage through existing side channels. (ii) Reducing the probability of co-location of attackers with cloud users. We discuss the proposed solutions pertaining to each of the two stated categories in the following subsections.

2.1. Reducing information leakage through existing side channels

The primary aim of the works falling in this category is to modify the architecture (Hardware, OS, Hypervisor etc.) in a way which either eliminates the side-channels or reduces the amount of information leakage through the side channels. In [13], the authors proposed to combat timing-based side-channels by modifying the RDTSC instruction which provides fine-grained timing information on Xen-virtualized x86 machines.

Wang et al. [14] proposed a new cache architecture which uses a security-aware cache replacement algorithm (SecRAND) to combat side-channel attacks. Liu et al. [15] designed a random fill cache architecture which, in addition to defending against contention-based attacks (e.g. Prime-Probe, Evict-Time), also provides security against reuse-based attacks (eg. Flush–Reload, Cache-Collision). Page [16] proposed the use of a configurable partitioned cache architecture which can defend against side-channel attacks by dynamically splitting the cache into protected regions.

Wang et al. [17] proposed a secure virtual network embedding scheme to combat information leakage through covert channels in a virtual-network environment. Li et al. [18] proposed a hypervisor-based defense system which aims to obfuscate the leaked timing information in IaaS clouds by using 3 replicas of each guest VM and only permitting the observation of aggregate timing information of these VMs. Varadarajan et al. [19] proposed to defend against cache-based side channel attacks by reducing the frequency of VM preemption which is controlled by the hypervisor scheduler.

Zhang et al. [20] proposed a system which can defend against side-channels by injecting noise into the cache and thereby obfuscating the information leakage through timing-based side-channels. Pattuk et al. [21] designed a system which prevents the compromise of cryptographic keys in shared cloud environments by partitioning the key across multiple VMs.

2.2. Reducing the probability of co-location of attackers with cloud users

The primary aim of the works falling in this category is to design ways to prevent or reduce the probability of co-location of a malicious user with a benign user. Establishing co-location is a major pre-requisite for conducting side-channel (and other co-location based) attacks. Therefore, a VM placement strategy which avoids (or reduces) the probability of malicious co-location would directly reduce the chances of a successful side-channel attack. There are two different ways by which the probability of co-location can be reduced: (i) by designing secure VM placement algorithms (ii) by designing secure live VM migration strategies.

Azar et al. [9] proposed a formal model for the design and analysis of secure placement algorithms and designed a random VM placement strategy to reduce co-location probability. Han et al. [22] proposed a game theoretic based model for comparing the security of different VM placement policies against co-location attacks. In [23], they also proposed a new allocation policy called Previously-selected-servers-first (PSSF) which aims to reduce the probability of co-location by minimizing the spread of user's requested VMs.

Berrima et al. [24] proposed a placement algorithm which aims to decrease the co-location rate by compromising the VM startup time instead of resource optimization. Their strategy uses a mixing queue of a predefined capacity where VM requests are buffered. The actual placement begins only after the queue is full by selecting a random VM from the queue and allocating it to a physical server as per the optimization strategy.

Qiu et al. [25] proposed a Co-Residency-Resistant vm Deployment (CRRD) strategy based on custom defined threshold parameters. Their strategy takes into account these threshold parameters resulting in a policy of "first spread, later centralize and the more VMs you create the more concentrate". Afoulki et al. [26] proposed a secure placement policy by incorporating a trust relationship among cloud users. Each user is given the freedom to choose his own set of adversaries and this is taken into account while making placement decisions.

Zhang et al. [27] proposed an incentive compatible VM migration strategy based on the moving target defense philosophy to combat co-location based attacks. Moon et al. [28] proposed a cloud provider assisted VM migration strategy to restrict co-residency and limit the amount of information leakage due to side-channel attacks.

2.3. Analysis of past approaches

Table 1 provides a high-level summary of the pros, cons and possible improvements pertaining to past works. Most of the defenses described in Section 2.1 suffer from two major limitations which prevent them from being adopted in the current cloud architectures: (i) They require major changes to the existing cloud infrastructure including, but not limited to, hypervisor, guest OS and physical hardware. (ii) They do not ensure security against currently unknown side-channels.

Compared to hardware-based defenses described in Section 2.1, the strategies described in Section 2.2 might be more suitable and feasible to be adopted in cloud environments for two main reasons: (i) unlike hardware based defenses, they do not require major changes to existing cloud infrastructure, and (ii) they are likely to be more resilient against arbitrary and currently unknown side-channel attacks. However, these solutions also have some common shortcomings: (i) Modified placement algorithms significantly affect the resource utilization of the cloud data center. (ii) Migration-based defenses incur extra network cost to the cloud provider. (iii) Most of the proposed strategies have not been evaluated on real-world cloud workload dataset.

Also, it is worth noting that a central assumption in the earlier works is that the cloud provider has no prior knowledge regarding which users are malicious. We argue that this is rather a strong assumption and it is indeed possible, in some scenarios, to detect a malicious user based on his cache-based activities. Zhang et al. [6] proposed "CloudRadar" as a system to detect side-channel attacks by correlating the attacker's anomalous cache activities to a benign user's cryptographic application activity. They used signature-based detection schemes to identify a user's execution of some cryptographic application and anomaly-based detection schemes to identify the abnormal cache activities of a potential attacker. They also leveraged the use of hardware performance counters, which is present in modern CPUs, to passively collect and monitor the cache statistics of cloud users. Similarly, Chiappetta et al. [29] described ways to utilize the information from hardware performance counters coupled with techniques such as correlation-based approach, anomaly detection, and supervised learning to detect FLUSH+RELOAD based side-channel attacks in real-time.

In [7], the authors used several types of classifiers, including, but not limited to, Naive Bayes and Support Vector Machine. These probabilistic classifiers were trained on input features relevant for detecting side-channel attacks such as Branch misses, LLC misses, LLC references, Unhalted core cycles, number of instructions etc. and were later used to classify an unknown activity as either benign or malicious. In [8], the authors proposed a method to construct the security profile of VMs by combining different parameters such as Internal vulnerability score, Intrusion behavior score and Trust based score. The weighted average of these scores was used to construct a Security Profile Score (SPS). These mechanisms can be used by the cloud provider to proactively monitor the activities of tenants and determine whether a particular tenant is an attacker or not using a classification system.

To address the shortcomings of prior works, we relax our assumption to include the case where the cloud provider might have some information about the category of users. We also propose a co-location resistant placement algorithm called "Previously Co-Located Users First" with a dual-objective of maximizing both resource utilization and co-location security. An extensive comparative analysis of the proposed algorithm and existing placement algorithms has also been carried out using the Azure cloud workload traces [10]. We also study the effect of utilizing a classification system (for categorizing users as benign or malicious) on the performance of VM placement algorithms.

3. Problem formulation

3.1. Assumptions and threat model

The Cloud Provider uses a suitable placement algorithm to fulfill the incoming VM requests. Here we list down some of the assumptions with respect to the capabilities of Cloud Provider (*CP*):

- The *CP* may or may not have knowledge about the category of users.
- The *CP* has no information about the future VM requests and has to make placement decisions on VM requests as they arrive.
- The *CP* has sole control over the assignment decisions of VMs to PMs.
- The *CP* does not use any migration algorithm to re-locate an already allocated VM. In other words, once a VM has been assigned to a particular server, that VM is assumed to run on only that server until the user terminates it.

Table 1
Analysis of past approaches.

Category	Advantages	Limitations	Possible improvements
Architecture specific defenses, e.g., [13–21]	Effective in reducing information leakage through existing side channels by modifying OS, cache, hypervisor, network, etc.	Require major changes to the existing cloud infrastructure and do not ensure security against currently unknown side-channels.	Design of generalized techniques which could handle future side-channels while requiring minimum hardware/software modifications.
VM placement based defenses, e.g., [9,22–28]	Effective in reducing the probability of malicious co-location and are resilient against arbitrary side-channels.	Affect the resource utilization of the data center. Migration-based defenses incur extra network cost to the cloud provider.	Utilization of VM cache statistics to ensure greater isolation from attackers while ensuring high resource utilization.

The aim of malicious users is to compromise the VMs belonging to benign users by conducting side-channel (or other co-location based) attacks after a successful co-location. Here we list down some of the assumptions with respect to the capabilities of malicious users:

- Like all other cloud users, the malicious users are in sole control of their own workload. They are free to decide the timing, core-requirement and memory-requirement of their VM requests.
- The malicious users have the ability to compromise a benign user once their VM is co-located with that of a benign user by leaking information through arbitrary side-channels.
- Multiple malicious users can coordinate among themselves in order to compromise a particular benign user.

3.2. Notations

Here we list some of the notations to be used later in this paper:

- N: Total number of cloud users
- P_m: Percentage of cloud users that are malicious
- U_i: A cloud user i
- V_i: A virtual machine instance i
- V_i^c: Number of CPU cores occupied by V_i
- V_i^m: Amount of memory (in GB) occupied by V_i
- U_{V_i}: User who requested virtual machine instance i
- P_i: A physical machine instance i
- P_i^c: Number of CPU cores for P_i
- $Users(P_i)$: Set of all users whose VM instances are resident on P_i
- $CoLocated(U_i)$: Set of all users which have been co-located with a user U_i at least once
- l_{V_i}: Life-time of V_i which is the difference between the time (in seconds) when V_i was requested and the time when it was terminated
- l_{P_i}: Life-time of P_i which is the total time (in seconds) during which P_i hosts at least one VM instance

3.3. Performance metrics

3.3.1. Core utilization (CU)

The standard metric used to evaluate Online Bin Packing algorithms is the "Number of bins used". In [9], the authors explained why the "Number of bins used" ("Number of physical servers" in our problem) is not an accurate metric for the Online VM assignment problem. The reason, as pointed out by the authors, was that "Number of bins used" does not capture the actual amount of resources expended in an online VM assignment scenario. We now restate the example used by the authors to explain the specific reason.

Suppose there are two different VM placement algorithms — $\mathcal{P}_1$ and $\mathcal{P}_2$. $\mathcal{P}_1$ turns on one PM every hour for a total duration of n hours whereas $\mathcal{P}_2$ turns on n PMs in the first hour and then keeps those servers on for n hours. In such a scenario, both algorithms used the same number of physical servers to satisfy the incoming VM requests but clearly the first algorithm $\mathcal{P}_1$ is more efficient in terms of resource usage. Therefore, it is important to consider the amount of time for which physical resources are being used rather than just the quantity of the resources. Another reason why "Number of bins used" is not an accurate metric is that it does not account the fact that servers which were turned on at some time-instant to accommodate the incoming VM requests might be turned off once those VMs have been terminated. This is because, in "Online Bin Packing" problem, it is assumed that once an item has been packed in a bin, the item stays in that bin and cannot depart. However, this is clearly not true in a cloud environment.

In [30], the authors showed that the energy consumption by the CPU cores in the PMs exceeds all the other resources. Keeping this in mind, we argue that in an ideal scenario, all the CPU cores of live PMs would be allocated to some live VM. However, because of sub-optimal placement decisions made by the placement algorithm and the termination of live VMs by the cloud users, it is possible for some of the CPU cores in live PMs to remain idle. Therefore, to evaluate the performance of different placement algorithms with respect to resource usage, we use Core Utilization as a metric. Core Utilization is the ratio of the total number of cores used by each VM weighted by their respective lifetime to the sum of total cores of each PM weighted by their respective lifetime. Mathematically, we define the Core-Utilization CU as follows:

$$CU = \frac{\sum_{all\ V} V^C \times l_V}{\sum_{all\ P} P^C \times l_P} \tag{1}$$

Note that the CU value as indicated by (1) is a real value ranging between 0 and 1. CU is 0 when all the PM cores are idle and it becomes 1 when all the PM cores are being occupied by VMs. Therefore, our aim would be to maximize CU. In our results, we will indicate CU in the form of percentage.

3.3.2. Co-location resistance (CLR)

Currently, there is no standard metric to quantify the security of a cloud data center with respect to co-location based attacks. Azar et al. [9] introduced the idea of "Single CL-resistance" as a metric to quantify the resistance of a cloud to co-location based attacks. In "Single CL-resistance", the adversary is only interested in co-locating his VM with at least one of the target VMs. Therefore, if a malicious user is successful in co-locating his VMs with at least one VM of a benign user, that benign user is assumed to

be compromised. As stated in Section 3.1, we have assumed that multiple malicious users in the cloud can coordinate their efforts to compromise a benign user. Keeping this in mind, we define a user as SAFE if none of his VM instances is co-located with that of a malicious user throughout the VM placement process. We then define the Co-Location Resistance (*CLR*) of the cloud as the ratio of benign users who are SAFE to the total number of benign users.

$$CLR = \frac{Total\ number\ of\ benign\ users\ who\ are\ SAFE}{Total\ number\ of\ benign\ users} \quad (2)$$

The *CLR* indicated by (2) can also be interpreted as the probability of a randomly chosen benign cloud user to be SAFE. Also, note that the *CLR* value indicated by (2) is a real value ranging between 0 and 1. *CLR* is 0 when all the benign users are UNSAFE. On the other hand, if all the benign users are SAFE, the *CLR* becomes 1. Therefore, our aim would be to maximize the *CLR* value. In our results, we will indicate *CLR* in the form of percentage.

We will now illustrate an example to explain the *CLR* metric. Assume a cloud environment consisting of 4 benign users (U_1, U_2, U_3 and U_4) and one malicious user (U_A). Fig. 1 depicts the assignment of VMs (belonging to different users) after the placement phase is complete. We notice that the VMs belonging to the malicious user U_A have been allocated on Server-2 and Server-3. On Server-2, the VMs of U_A is co-located with a VM belonging to a benign user U_2. Similarly, on Server-3, the VM of U_A is co-located with 3 VMs belonging to a benign user U_1. Therefore, by our definition, all benign users except U_2 and U_1 are SAFE since none of their VMs have been co-located with that of U_A. Therefore, the *CLR* in this scenario as per (2) would be $\frac{2}{4}$.

3.4. Problem statement

A multi-tenant public IaaS cloud service has N users in total. The users belong to one of the two categories — benign or malicious. The benign users are the normal cloud users who request VM instances to utilize cloud resources for their computation. The intent of malicious users is to compromise the benign users who are using cloud services by using co-location based attacks. The cloud provider may or may not have knowledge about the category of users.

In such a cloud environment, VM requests of different resource requirement arrive one-by-one. Each incoming VM request V belongs to a particular cloud user and is characterized by a specific number of cores V^c and memory V^m requirement. The **Secure VM Placement** (SVP) problem is to assign these incoming VMs to the available PMs in a way which maximizes both *CU* and *CLR*.

Note that the SVP problem is similar to the well known Online Bin Packing problem (OBP) in that the items and bins in OBP problem are equivalent to VMs and PMs respectively in the SVP problem. The difference between OBP and SVP is that: OBP is a single-objective optimization problem where the objective is to minimize the number of bins used to pack the items. On the other hand, SVP is dual-objective: in addition to minimizing the amount of resources needed to satisfy the requirement of incoming VM requests, we also need to minimize the probability of malicious co-location.

3.5. Mathematical problem formulation

Given a set of s PMs: $P_1, P_2, \ldots, P_s$, each having a fixed core P_i^c and memory capacity P_i^m, and a list of t incoming VMs: $V_1, V_2, \ldots, V_t$, each belonging to a specific cloud user and characterized by a specific number of cores V_i^c and memory V_i^m requirement, find an assignment of each V_i to a P_j in a way which:

Maximizes Core-Utilization (*CU*) given by Eq. (1),

Maximizes Co-Location Resistance (*CLR*) given by Eq. (2),

subject to

PM core constraint given by Eq. (3)

$$\sum_{i=1}^{t} V_i^c \times x_{ij} \leq P_j^c\ ,\ \forall j \in \{1,\ 2,\ ....,\ s\} \quad (3)$$

PM memory constraint given by Eq. (4)

$$\sum_{i=1}^{t} V_i^m \times x_{ij} \leq P_j^m\ ,\ \forall j \in \{1,\ 2,\ ....,\ s\} \quad (4)$$

One VM to exactly one PM constraint given by Eq. (5)

$$\sum_{j=1}^{s} x_{ij} = 1\ ,\ \forall\ i\ \in \{1,\ 2,\ .....,\ t\} \quad (5)$$

where $x_{ij} = 1$ if V_i is assigned to P_j, otherwise $x_{ij} = 0$

4. Proposed approach

4.1. Proposed algorithm

We now propose a placement algorithm called "Previously Co-Located Users First" (*PCUF*) with the aim of maximizing Core-Utilization and Co-Location Resistance of the cloud. Algorithm 1 is invoked whenever a new VM request arrives. It takes as input a VM V_i, a list of live PMs *live_pms* and a list of empty PMs *empty_pms*. U_{curr} represents the cloud user who requested V_i. We now provide a brief description of the working of our algorithm.

Algorithm 1: Previously Co-Located Users First

```
Input  : V_i, live_pms, empty_pms
Output: Assignment of V_i to a PM
U_curr ← U_{V_i};
if U_curr is not a new cloud user then
    eligible_pms ← all P_j ∈ live_pms s.t. P_j has enough
    resources to host V_i and
    Users(P_j) \ U_curr ⊆ CoLocated(U_curr);
    if eligible_pms ≠ φ then
        P_k ← PM from eligible_pms having least number of
        free cores;
        Assign V_i to P_k;
    else
        P_k ← PM from empty_pms;
        Assign V_i to P_k;
        Remove P_k from empty_pms and insert it into
        live_pms;
    end
else
    eligible_pms ← all P_j ∈ live_pms s.t. P_j has enough
    resources to host V_i;
    if eligible_pms ≠ φ then
        Assign V_i to a random PM P_j ∈ eligible_pms;
    else
        P_k ← PM from empty_pms;
        Assign V_i to P_k;
        Remove P_k from empty_pms and insert it into
        live_pms;
    end
end
```

First we check whether the user U_{curr} is a new cloud user or not i.e. we check whether he has made any VM requests in the past or not. If he is not a new cloud user, we construct a list of *eligible_pms* by including all those PMs P_j from *live_pms* which

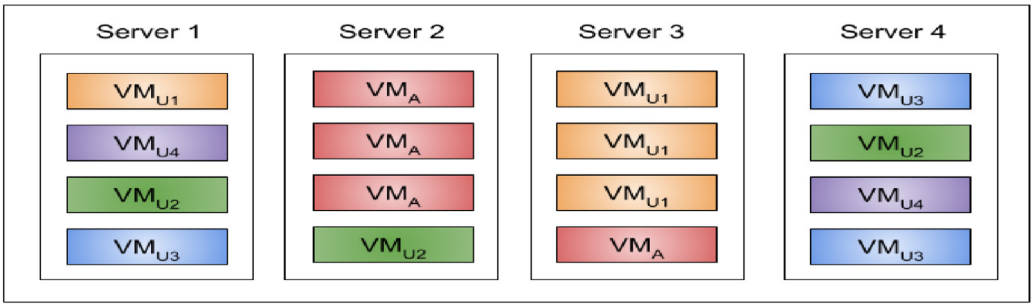

Fig. 1. Allocation outcome of an arbitrary placement algorithm in which the malicious user (red) manages to co-locate his VMs with VMs belonging to benign users U_1 (orange) and U_2 (green). Users U_3 (blue) and U_4 (violet) are SAFE as none of their VMs have been co-located with the malicious user. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

satisfy two conditions: (i) P_j has enough resources to host V_i, and (ii) $Users(P_j) \setminus U_{curr} \subseteq CoLocated(U_{curr})$ i.e. the users which are currently resident on P_j are a subset of users whom U_{curr} has already been co-located with. If this *eligible_pms* list is non-empty, we assign V_i to a PM belonging to *eligible_pms* which has least number of free cores. Otherwise, if *eligible_pms* is empty, we assign V_i to an empty PM P_k belonging to *empty_pms*.

On the other hand, if U_{curr} is a new cloud user and has not requested VM instances in the past, then we proceed as follows: we construct a list of *eligible_pms* by including all those PMs from *live_pms* which have enough resources to host V_i. If this *eligible_pms* list is non-empty, we assign V_i to a random PM belonging to *eligible_pms*. This random selection is done in order to make it difficult for a malicious user to get co-located with their intended target user. Otherwise, if *eligible_pms* is empty, we assign V_i to an empty PM P_k belonging to *empty_pms*.

In essence, the algorithm gives preference to those PMs which are currently hosting VMs belonging to the users who have already been co-located with U_{curr} in the past. The reason for choosing PMs in such a way is to limit the number of benign users that a malicious user can be co-located with. We believe that doing so would significantly affect the probability of a specific cloud user to get co-located with an adversary.

4.2. Theoretical analysis of the proposed algorithm

4.2.1. Time and space complexity

Since $Users(P_j) < P_j^c$ always (at least one core is assigned to a user's VM), a near-constant number of steps would be required to check whether a particular PM P_j is eligible or not. Therefore, the overall construction of *eligible_pms* list would require $O(|live\_pms|)$ number of steps. If the VM belongs to a new cloud user, the selection of a suitable PM from *eligible_pms* list requires an additional $O(1)$ steps (random selection), and $O(|eligible\_pms|)$ steps (selecting PM having least number of free cores) otherwise. As *eligible_pms* is always a subset of *live_pms*, the overall time complexity of Algorithm 1 for allocating a single VM is $O(|live\_pms|)$. During the allocation process, Algorithm 1 uses 3 lists — *live_pms*, *empty_pms*, and *eligible_pms*. Since *eligible_pms* is always a subset of *live_pms*, the space complexity for allocating a single VM is $O(|live\_pms| + |empty\_pms|)$.

4.2.2. Expected Co-Location Resistance provided by PCUF algorithm

In this section, we will derive an expression for the expected number of cloud users who will be SAFE at the end of the entire VM placement phase assuming *PCUF* is used as the placement algorithm. Consider a cloud data center consisting of N users out of which N_m users are malicious and N_b users are benign. Let π represent an arbitrary permutation sequence of these N cloud users. The sequence of users in π represents the sequence in which those cloud users make their first VM request to the cloud provider. Without loss of generality, we will denote the ith user in π as U_i. For e.g., if $\pi = U_1U_2$, it indicates that the timing of first VM request of user labeled U_1 precedes the timing of first VM request made by the user labeled U_2. We will associate a random variable X_i with each user U_i such that:

$$X_i = \begin{cases} 1 & \text{if } U_i \text{ is benign and SAFE} \\ 0 & \text{otherwise} \end{cases}$$

The expected number of benign users who will be SAFE is calculated by summing up the probability that $X_i = 1$ over all N cloud users and is indicated by (6).

$$E\ [\ Users\ who\ are\ benign\ and\ SAFE\ ]\ =\ \sum_{i=1}^{N} P(X_i\ =\ 1) \tag{6}$$

A user U_i can become co-located with another user U_j in 2 possible ways:

- If $j < i$ i.e. user U_j is in-front of U_i in the sequence π, then U_i can become co-located with U_j when U_i requests his first VM instance. If U_i does not co-locate with the user U_j during U_i's first VM allocation, it is guaranteed by our algorithm that future VM requests from U_i also will not be co-located with U_j.
- If $j > i$ i.e. user U_j is behind U_i in the sequence π, the U_i can become co-located with U_j when U_j requests his first VM instance. If U_j does not co-locate with the user U_i during U_j's first VM allocation, it is guaranteed by our algorithm that future VM requests from U_j also will not be co-located with U_i.

Having made the above observations, we can now calculate the probability that a user U_i is benign and does not get co-located with any malicious user throughout the allocation phase using (7).

$$\begin{aligned} P(X_i\ =\ 1)\ =\ & P(U_i\ is\ benign) \\ & \times\ P(U_i\ is\ not\ colocated\ with\ any\ malicious\ user \\ & \quad U_j |\ j\ <\ i\ and\ U_i\ is\ benign) \\ & \times\ P\ (No\ malicious\ user\ U_j\ gets\ colocated\ with \\ & \quad U_i;\ |\ j\ >\ i\ and\ U_i\ is\ benign) \end{aligned} \tag{7}$$

We will now calculate the probability of each of the three events (on the R.H.S) involved in (7).

- The probability that U_i is a benign user is simply the ratio of the number of benign users to the total number of cloud users and is indicated by (8).

$$P(U_i\ is\ benign)\ =\ \frac{N_b}{N} \tag{8}$$

- We will now calculate the probability that U_i does not get co-located with a malicious user U_j who in front of U_i ($j < i$) in the sequence π. To do so, we make the following observation: whenever user U_i makes his first VM request, the VM is either assigned to a random PM or a new PM depending on whether enough resources are available. To simplify our analysis, we will assume that a random incoming new user would get co-located with k other users on an average when his first VM is instantiated on a PM. Therefore, the probability of U_i not co-locating with any malicious user would be equal to the probability of that all k selected users out of $(i-1)$ preceding users are benign. The probability of this event happening is indicated by (9)

$$P(U_i \text{ is not colocated with any malicious user } U_j | \, j < i \text{ and } U_i \text{ is benign}) = \left(\frac{N_b - 1}{N-1}\right)^{min(i-1,\, k)} \tag{9}$$

- We will now calculate the probability that no malicious user U_j, who is behind U_i ($j > i$) in the sequence π, gets co-located with U_i. To do so, we will first state the probability for a user U_j who is malicious, to get co-located with U_i in (10).

$$P(U_j \text{ is malicious and } U_j \text{ gets colocated with } U_i; \, | \, j > i \text{ and } U_i \text{ is benign}) = \begin{cases} \frac{N_m}{N-1} \; ; \; if \; (j-1) \leq k \\ \frac{N_m}{N-1} \times \frac{\binom{j-2}{k-1}}{\binom{j-1}{k}} \; ; \; if \; (j-1) \geq k \end{cases} = \left(\frac{N_m}{N-1} \times \frac{min(j-1,\, k)}{j-1}\right) \tag{10}$$

Now we can calculate the probability that no malicious user U_j ($j > i$) gets co-located with benign user U_i in the following way:

$$P(\text{No malicious user } U_j \text{ gets colocated with } U_i; \, | \, j > i \text{ and } U_i \text{ is benign}) = \prod_{j=i+1}^{n} \left(1 - P(U_j \text{ is malicious and } U_j \text{ gets colocated with } U_i; \, | \, j > i \text{ and } U_i \text{ is benign})\right) = \prod_{j=i+1}^{N} \left(1 - \frac{N_m}{N-1} \times \frac{min(j-1,\, k)}{j-1}\right) \tag{11}$$

By plugging (8), (9) and (11) into (7), we derive the probability that a user U_i is benign and SAFE:

$$P(X_i = 1) = \frac{N_b}{N} \times \left(\frac{N_b - 1}{N-1}\right)^{min(i-1,\, k)} \times \prod_{j=i+1}^{N} \left(1 - \frac{N_m}{N-1} \times \frac{min(j-1,\, k)}{j-1}\right) \tag{12}$$

Finally, by plugging (12) into (6), the expected number of benign users who will be SAFE can be summarized using (13):

$$E[\, Users \text{ who are benign and SAFE }] = \sum_{i=1}^{N} \left(\frac{N_b}{N} \times \left(\frac{N_b - 1}{N-1}\right)^{min(i-1,\, k)} \times \prod_{j=i+1}^{N} \left(1 - \frac{N_m}{N-1} \times \frac{min(j-1,\, k)}{j-1}\right)\right) \tag{13}$$

Table 2
Azure Workload Statistics.

Characteristics	Count
Workload duration	10 consecutive days
Total number of VMs	619846
Total number of subscriptions	1884
Maximum number of running VMs at any time instant	16990
Average lifetime of VMs	4.08 hrs

The Co-Location Resistance, as defined in Section 3.3.2, can now be easily derived by dividing the expected number of benign users who are SAFE to the total number of benign users. We call this value as $CLR_{pcuf}^{theoretical}$ which represents the expected fraction of benign cloud users who will remain SAFE throughout the VM placement phase (assuming *PCUF* is used as the placement algorithm):

$$CLR_{pcuf}^{theoretical} = \frac{1}{N} \times \sum_{i=1}^{N} \left(\left(\frac{N_b - 1}{N-1}\right)^{min(i-1,\, k)} \times \prod_{j=i+1}^{N} \left(1 - \frac{N_m}{N-1} \times \frac{min(j-1,\, k)}{j-1}\right)\right) \tag{14}$$

Given the total number of cloud users (N), the percentage of malicious users (P_m) and the average number of users that a user gets co-located with during his first VM allocation (k), we can estimate the *CLR* of our proposed *PCUF* strategy using (14). For example, consider a cloud data center with $N = 100$, $P_m = 10\%$ and $k = 2$. Using N and P_m, we find $N_m = \frac{P_m}{100} \times N = 10$ and $N_b = N - N_m = 90$. By plugging N, N_m, N_b and k into (14), we find that our estimated *CLR* is 67.43%. In Section 5.4, we evaluate the accuracy of our derived *CLR* by comparing it with empirical results.

5. Results and analysis

5.1. Evaluation methodology

We have conducted our experiments on the Microsoft Azure VM workload dataset that has been made public in 2017 by Cortez et al. [10]. The dataset contains 2,013,767 VMs over a period of 30 consecutive days. For our evaluation, we use a subset of this dataset by including all the VMs that were created and terminated between the 11th and 20th day of the 30 day period. The results for other time intervals of the dataset are similar and have not been presented in this paper due to space limitations. Table 2 summarizes some statistics related to the dataset which we have used for our evaluation.

The main features in the workload that are relevant for evaluating VM placement algorithms include VM id, Subscription id, VM Start time, VM Stop time, VM core count and VM memory (in GB). We map each subscription id to a unique cloud user in our evaluation. In our experiments, it has been assumed that all PMs have the same core and memory size — 32 cores and 224 GB respectively, which is equal to twice the configuration of largest VM present in the dataset. Table 3 summarizes the statistics related to VM core and memory configuration.

Unfortunately, the workload dataset has no information about which subscriptions were used for conducting co-location based attacks. Therefore, for our evaluation purpose, we randomly map P_m% of the subscriptions to malicious users, while the rest of the subscriptions are mapped to benign cloud users. After this, we form a sequence of VM start and stop events, sort the events as per their timestamp and then invoke VM placement algorithm for each event sequentially. We repeat this for all the events and then finally compute the *CU* and *CLR* of the cloud as per the equations described in Section 3.3. We repeat each experiment 20 times so that we can have a different set of malicious users for each experiment. We then compute the average *CU* and *CLR* over all 20 experiments and use this average value as our final estimate.

Table 3
Azure VM Instance Types.

VM cores	VM memory (in GB)	VM count
1	0.75	12119
	1.75	280069
	2	221
2	3.5	163702
	4	80
	14	61
	16	6
4	7	69627
	8	11
	28	98
	32	10
8	14	67137
	16	6
	56	26609
	64	5
16	112	85

5.2. Explanation of different standard and secure placement algorithms used in the analysis

In our analysis, we compare the performance of our algorithm with 3 most commonly used standard VM placement policies: (i) Best-Fit (*BF*), (ii) Worst-Fit (*WF*) and (iii) Random Placement (*RP*), along with 3 co-location resistant VM placement policies: (i) Amazon's Dedicated Instance placement [31] which we will denote as *DI*, (ii) CLR placement strategy proposed by Azar et al. [9] which we will denote as *AZ*, and (iii) Previously Selected Servers First (*PSSF*) strategy proposed by Han et al. [23]. In the following subsections, we will provide a brief summary of each of the above-mentioned placement strategies.

5.2.1. Standard placement algorithms

In the Best-Fit policy, an incoming VM request is assigned to a live PM having the least remaining free cores. Contrary to this, the Worst-Fit policy assigns an incoming VM to a live PM having the most number of remaining free cores. Random Placement policy assigns an incoming VM to a PM selected uniformly at random from the list of live PMs. In all these policies, if no live PM has sufficient resources to host the incoming VM, a new empty PM is started for hosting that VM.

5.2.2. Amazon's Dedicated Instance strategy

In Amazon's Dedicated Instance strategy, two VMs that belong to different AWS accounts are never co-located on the same PM. Therefore, for allocating a VM V_i requested by user U_j, we only consider those PMs which host only U_j's VMs. Among those PMs, we consolidate U_j's VMs using a Best-Fit approach. If there are no candidate PMs for allocation, a new PM is started for accommodating V_i.

5.2.3. Azar's placement strategy

Azar et al. [9] proposed a random placement strategy wherein all the PMs in the data center are dynamically labeled either OPEN (already host some VMs and can receive more VMs), CLOSED (cannot receive more VMs) or EMPTY (do not host any VMs). At every instant, exactly λ PMs are kept OPEN, with λ being a predefined parameter of the algorithm. When a new VM request arrives, it is assigned to a PM P_j randomly selected from λ OPEN PMs. If P_j cannot accommodate any more VMs, it is labeled as CLOSED and a PM labeled EMPTY is re-labeled as OPEN to keep exactly λ PMs OPEN. However, the authors have not mentioned any procedure for the de-allocation of VMs. Therefore, we perform the deallocation procedure for their algorithm in the following way: when a VM V_i is deallocated from PM P_j, we first check whether P_j becomes empty. If P_j becomes empty after deallocation, we label P_j as EMPTY, otherwise, we label it as OPEN. Note that this deallocation procedure might result in more than λ OPEN PMs.

5.2.4. Han's placement strategy

Han et al. [23] proposed a VM placement policy called *PSSF* which aims to satisfy 3 requirements: security, power consumption and workload balance. The *PSSF* algorithm is parametrized using 2 variables — N^* and N_G. N^* determines the maximum number of VMs of a user which can be allocated to the same PM, which in turn helps in tuning the workload balance. Since we are not dealing with the objective of workload balance in this paper, therefore we ignore the parameter N^* in our evaluation. In their approach, all the PMs in the data center is divided into groups of size N_G and every PM dynamically maintains a list of cloud users that it has ever hosted. When a user U_i requests a new VM, those PMs which have already hosted or are currently hosting U_i's VMs are considered first for assignment. If no such PM exists, then remaining PMs are considered with the priority being given to PMs which have a lower group index and more number of resources left.

5.3. Comparative analysis of different placement algorithms assuming no information about the category of users is available

Figs. 2 and 3 illustrate the *CLR* and *CU* respectively, obtained by executing different standard and co-location resistant placement algorithms on the Azure workload with varying percentage of malicious users (P_m). We have fixed the total number of PMs in the data center as 16,990 which is the same as the maximum number of simultaneously live VMs described in Table 2. While evaluating the *AZ* strategy, we test for 4 different values of the parameter λ (5%, 10%, 15% and 20%) indicating the fraction of total PMs that are kept OPEN at all times. Similarly, while evaluating the *PSSF* strategy, we test for 4 different values of the parameter N_G (5%, 10%, 15% and 20%) indicating the group size in terms of the fraction of total PMs in the data center. We make the following important observations in our analysis:

- Fig. 3 indicates the following relationship among the various placement algorithms in terms of Core Utilization:

  $$BF > RP > PSSF_{N_g=5\%} > PCUF > WF > DI > PSSF_{N_g=10\%} > PSSF_{N_g=15\%} > PSSF_{N_g=20\%} > AZ_{\lambda=5\%} > AZ_{\lambda=10\%} > AZ_{\lambda=15\%} > AZ_{\lambda=20\%}$$

  BF guarantees better *CU* than all other placement algorithms whereas $AZ_{\lambda=20\%}$ performs worst. Our proposed algorithm *PCUF* ranks fourth among all algorithms and is less efficient (in terms of *CU*) than *BF*, *RP* and $PSSF_{N_g=5\%}$ by 14.25%, 5.68% and 1.69% respectively. We note that all standard placement algorithms — BF, WF and RP perform better than DI. Among the co-location resistant placement algorithms, all algorithms except $PSSF_{N_g=5\%}$ and *PCUF* are worse off than DI. We also note that the Core-Utilization value is independent of the percentage of malicious users for all placement algorithms because the algorithm has no knowledge about the category of each user.
- Fig. 2 indicates that as P_m increases, the *CLR* of all algorithms (except *DI*) decrease. By increasing P_m, the number of malicious VMs in the cloud increases which in turn increases the likelihood of a benign user's VM to get co-located with at least one malicious VM. In fact, with more than 20% malicious users in the cloud, the *CLR* of *BF*, *WF* and *RP* is

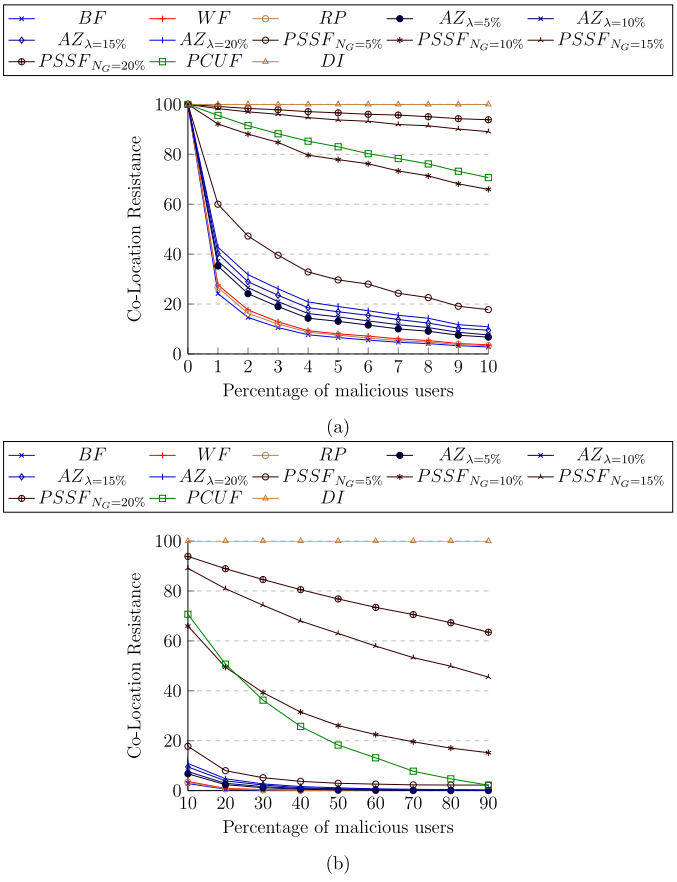

Fig. 2. Variation of Co-Location Resistance (*CLR*) of different placement algorithms with the change in percentage of malicious users (P_m). In 2(a), P_m varies between 0 and 10 at increments of 1 whereas, in 2(b), P_m varies between 10 and 90 at increments of 10.

almost close to 0. This illustrates that standard VM placement algorithms provide almost negligible co-location resistance compared to our proposed algorithm. The same figure also indicates the following relationship among the various placement algorithms in terms of Co-Location Resistance:

When percentage of malicious users $< 20\%$

$$DI > PSSF_{N_g=20\%} > PSSF_{N_g=15\%} > PCUF > PSSF_{N_g=10\%} > PSSF_{N_g=5\%} > AZ_{\lambda=20\%} > AZ_{\lambda=15\%} > AZ_{\lambda=10\%} > AZ_{\lambda=5\%} > WF > RP > BF$$

When percentage of malicious users $\geq 20\%$

$$DI > PSSF_{N_g=20\%} > PSSF_{N_g=15\%} > PSSF_{N_g=10\%} > PCUF > PSSF_{N_g=5\%} > AZ_{\lambda=20\%} > AZ_{\lambda=15\%} > AZ_{\lambda=10\%} > AZ_{\lambda=5\%} > WF > RP > BF$$

- We note that any approach which provides lower *CU* than *DI* is impractical simply because the *CLR* provided by *DI* is always 100%. Therefore, although $PSSF_{N_G=10\%}$(for $P_m > 20\%$), $PSSF_{N_G=15\%}$ and $PSSF_{N_G=20\%}$ could have better *CLR* than *PCUF* as Fig. 2 indicates, they might not be useful as their *CU* values are too low. Having said that, we note that *PCUF* clearly outperforms *BF*, *WF*, *RP* and *AZ* in terms of *CLR* for all values of P_m by a huge margin. Therefore, the only competitor to *PCUF* algorithm is $PSSF_{N_G=5\%}$. Among these two, the *CU* of *PCUF* is less than that of $PSSF_{N_G=5\%}$ by 1.69%. However, we note that *PCUF* consistently provides a significantly higher *CLR* compared to $PSSF_{N_G=5\%}$ for all values of P_m.

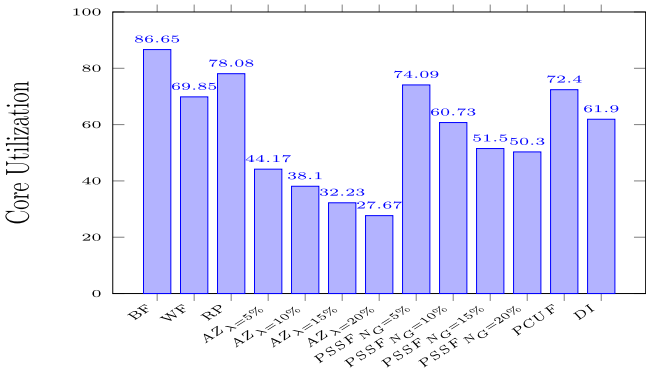

Fig. 3. Comparison of Core Utilization (*CU*) among *PCUF* and different standard and co-location resistant placement algorithms.

From these observations, we conclude that our proposed *PCUF* algorithm provides much higher co-location resistance compared to standard placement algorithms (*BF*, *WF*, *RP*) with minimal compromise in core utilization. Also, *PCUF* achieves a better balance between *CU* and *CLR* compared to existing secure placement algorithms (*DI*, *PSSF*, *AZ*).

5.4. Comparison of $CLR_{pcuf}^{theoretical}$ and CLR_{pcuf}^{azure}

We will now compare the $CLR_{pcuf}^{theoretical}$ given by (14) with the actual *CLR* obtained by executing *PCUF* on Azure dataset which we call CLR_{pcuf}^{azure}. By executing *PCUF* on the Azure dataset, we observed that the average number of users with whom a new incoming user gets co-located for the first time is approximately 2.23. We use this value as our k while calculating $CLR_{pcuf}^{theoretical}$. We set $N = 1884$ as per the number of users in our Azure dataset.

Fig. 4 shows that our derived $CLR_{pcuf}^{theoretical}$ provides a decent approximation to CLR_{pcuf}^{azure}. Specifically, we observe that CLR_{pcuf}^{azure} is strictly greater than our derived $CLR_{pcuf}^{theoretical}$ and the maximum difference between CLR_{pcuf}^{azure} and $CLR_{pcuf}^{theoretical}$ is 9.21% at $P_m = 30\%$. The average difference of our derived $CLR_{pcuf}^{theoretical}$ from CLR_{pcuf}^{azure} is 5.9%. A possible reason for this difference might be attributed to VM de-allocation requests from cloud users which we have not accounted for in our theoretical analysis.

5.5. Effect of utilizing user classification information on the performance of placement algorithms

In Section 3.1, we assumed that the cloud provider may or may not have knowledge about the category of users. This section pertains to the empirical analysis of different placement algorithms for the former case. In Section 2.3, we have already described past works that demonstrate specific techniques using which a cloud provider can gain knowledge about the category of users. To model the "cloud provider's knowledge about the category of users" in an abstract way, we proceed as follows:

Each cloud user will be associated with 2 attributes — Actual Class Label (*ACL*) and Predicted Class Label (*PCL*). Each of these class labels can either be $\mathcal{M}$ (indicating malicious) or $\mathcal{B}$ (indicating benign). The *ACL* of each malicious user is $\mathcal{M}$ and each benign user is $\mathcal{B}$. The cloud provider will have access to a Binary Classifier $\mathcal{C}_{p,q}$ which can predict whether a user U is malicious or not. Note that the cloud provider will have access only to the predicted class labels (PCL_U) of each cloud user and will have no knowledge about the user's actual class labels (ACL_U).

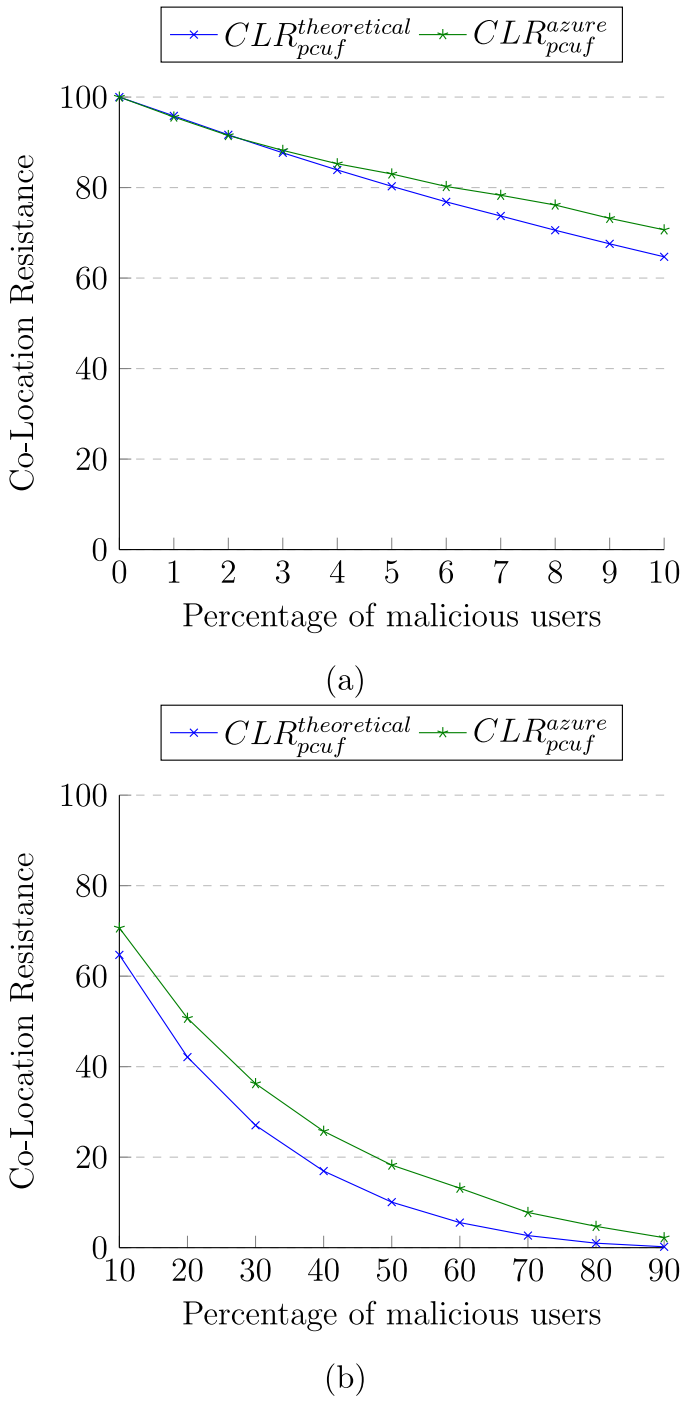

Fig. 4. Comparison between the theoretically derived *CLR* for *PCUF* ($CLR^{theoretical}_{pcuf}$) and the empirical *CLR* obtained by executing *PCUF* algorithm on Azure dataset (CLR^{azure}_{pcuf}) for different percentages of malicious users (P_m). In 4(a), P_m varies between 0 and 10 at increments of 1 whereas, in 4(b), P_m varies between 10 and 90 at increments of 10. $CLR^{theoretical}_{pcuf}$ well approximates CLR^{azure}_{pcuf} for smaller values of P_m but the difference increases for larger values of P_m.

Also, we assume that the classifier $\mathcal{C}_{p,q}$ has a True Positive rate of p and False Positive Rate of q which remains constant throughout the entire VM allocation phase. The True Positive rate of $\mathcal{C}_{p,q}$ is the ratio of the number of users who were predicted malicious and are actually malicious (users having both $PCL_U = \mathcal{M}$ and $ACL_U = \mathcal{M}$) to the total number of actual malicious users. The False Positive Rate of $\mathcal{C}_{p,q}$ is the ratio of the number of users who were predicted malicious but are actually benign (users having $PCL_U = \mathcal{M}$ and $ACL_U = \mathcal{B}$) to the total number of actual benign users. The predictions generated by such a classifier can be utilized by the cloud provider's placement algorithm to make better placement decisions.

As stated in Section 5.1, we randomly map P_m% of the subscriptions to malicious users, while the rest of the subscriptions are mapped to benign cloud users. Let N_m and N_b denote the number of malicious and benign users respectively after such mapping has been done. The Actual Class Label (ACL_U) of each malicious and benign user is fixed as $\mathcal{M}$ and $\mathcal{B}$ respectively. To generate the Predicted Class Label of each user (PCL_U), we simulate the prediction results of a hypothetical Binary Classifier $\mathcal{C}_{p,q}$ using the following technique:

- Among all the N_m malicious users, we assign $PCL_U = \mathcal{M}$ to $p \times 100\%$ randomly selected users and the rest $(1-p) \times 100\%$ users are assigned $PCL_U = \mathcal{B}$. This is done considering the fact that the True Positive Rate of the classifier is p.
- Among all the N_b benign users, we assign $PCL_U = \mathcal{M}$ to $q \times 100\%$ randomly selected users and the rest $(1-q) \times 100\%$ users are assigned $PCL_U = \mathcal{B}$. This is done considering the fact that the False Positive Rate of the classifier is q.

Also, in our analysis, we have only considered those Binary Classifiers for which the True Positive rate is not less than the False Positive rate i.e, $p >= q$. As stated in [32], any classifier that has a True Positive rate which is less than its False Positive rate can be negated (by negating its classification decision on every instance) to produce a corresponding classifier which has a True Positive rate greater than its False Positive rate.

Taking into account the additional information regarding user's category which is available, we also make a slight modification to the placement algorithms which, conventionally, do not utilize that information at all. We use $\mathcal{A}_1$ to denote an arbitrary placement algorithm which does not utilize user classification information. Examples of $\mathcal{A}_1$ include *BF*, *WF*, *RP*, *PCUF*, *PSSF*, *AZ*. We design an algorithm $\mathcal{A}_2$ which accounts for user classification information in the following way: At every instant, $\mathcal{A}_2$ maintains 2 group of PMs $- G_{\mathcal{M}}$ and $G_{\mathcal{B}}$. $G_{\mathcal{M}}$ and $G_{\mathcal{B}}$ contain those PMs which currently host VMs belonging to users whose $PCL = \mathcal{M}$ and $PCL = \mathcal{B}$ respectively. If the incoming VM request V belongs to a user U who has been predicted as malicious by $\mathcal{C}_{p,q}$ i.e, $PCL_U = \mathcal{M}$, then we invoke $\mathcal{A}_1$ with $live\_pms = G_{\mathcal{M}}$ otherwise we invoke $\mathcal{A}_1$ with $live\_pms = G_{\mathcal{B}}$.

Using the technique described above, we can modify an arbitrary algorithm designed based on the assumption that no user classification information is available into a corresponding algorithm based on the relaxed version of the assumption. From now on, we will refer to BF_2, WF_2, RP_2, $PCUF_2$, $PSSF_2$ and AZ_2 as the modified versions of algorithms described in Section 5.2. We study the effect of different classifier parameters such as "True Positive rate" (p) and "False Positive rate" (q) on the performance of these placement algorithms. For doing so, we fix $P_m = 10\%$ and vary the True Positive (p) and False Positive (q) rate of the Binary Classifier $\mathcal{C}_{p,q}$. Also for $PSSF_2$ and AZ_2 algorithm, we fix $N_g = 5\%$ and $\lambda = 5\%$ respectively. After carrying out multiple experiments, we made some important observations:

- Tables 4–9 indicate the *CU* values obtained by executing BF_2, WF_2, RP_2, $PCUF_2$, $PSSF_2$ and AZ_2 respectively for different values of p and q. We observe that Core-Utilization of the BF_2, WF_2, RP_2 and $PCUF_2$ is almost unaffected by p and q. Also, we observe that BF_2 provides best *CU* compared to other algorithms whereas AZ_2 provides the worst *CU* for all values of p and q. $PCUF_2$ ranks third in terms of *CU*, lagging behind BF_2 and RP_2 by approximately 13% and 5% respectively.

Table 4
CU for BF_2.

$p:q$	0	0.2	0.4	0.6	0.8	1.0
1.0	85.92	85.57	85.57	85.57	85.7	86.65
0.8	85.99	85.65	85.47	85.55	85.66	–
0.6	86.14	85.6	85.73	85.58	–	–
0.4	86.15	85.82	85.58	–	–	–
0.2	86.23	85.69	–	–	–	–
0.0	86.65	–	–	–	–	–

Table 5
CU for WF_2.

$p:q$	0	0.2	0.4	0.6	0.8	1.0
1.0	69.0	68.2	68.62	68.73	68.46	69.85
0.8	69.14	68.72	68.59	68.31	68.8	–
0.6	69.28	68.6	68.39	68.61	–	–
0.4	69.54	68.73	68.52	–	–	–
0.2	69.7	68.47	–	–	–	–
0.0	69.85	–	–	–	–	–

Table 6
CU for RP_2.

$p:q$	0	0.2	0.4	0.6	0.8	1.0
1.0	77.16	76.46	76.35	76.59	76.91	78.08
0.8	77.31	76.47	76.51	76.42	76.87	–
0.6	77.53	76.67	76.62	76.38	–	–
0.4	77.73	76.77	76.29	–	–	–
0.2	77.83	76.68	–	–	–	–
0.0	78.08	–	–	–	–	–

Table 7
CU for $PCUF_2$.

$p:q$	0	0.2	0.4	0.6	0.8	1.0
1.0	72.47	72.4	72.47	72.43	72.35	72.5
0.8	72.48	72.41	72.53	72.41	72.5	–
0.6	72.57	72.44	72.58	72.46	–	–
0.4	72.5	72.45	72.51	–	–	–
0.2	72.46	72.45	–	–	–	–
0.0	72.59	–	–	–	–	–

Table 8
CU for $PSSF_2$.

$p:q$	0	0.2	0.4	0.6	0.8	1.0
1.0	71.29	65.96	61.68	63.46	67.97	74.08
0.8	71.89	66.69	62.02	62.72	67.31	–
0.6	72.49	67.36	61.93	62.29	–	–
0.4	72.94	67.69	62.3	–	–	–
0.2	73.51	67.69	–	–	–	–
0.0	74.09	–	–	–	–	–

Table 9
CU for $AZAR_2$.

$p:q$	0	0.2	0.4	0.6	0.8	1.0
1.0	41.0	37.52	37.67	37.56	38.7	44.16
0.8	41.5	37.56	37.66	37.51	37.62	–
0.6	42.28	37.64	37.6	37.54	–	–
0.4	42.83	38.0	37.61	–	–	–
0.2	43.3	37.92	–	–	–	–
0.0	44.17	–	–	–	–	–

Table 10
CLR for BF_2.

$p:q$	0	0.2	0.4	0.6	0.8	1.0
1.0	100.0	80.05	60.29	41.02	21.72	2.82
0.8	15.25	10.61	6.57	4.2	3.13	–
0.6	7.64	5.03	3.75	3.18	–	–
0.4	5.34	3.68	3.23	–	–	–
0.2	3.68	3.2	–	–	–	–
0.0	2.82	–	–	–	–	–

Table 11
CLR for WF_2.

$p:q$	0	0.2	0.4	0.6	0.8	1.0
1.0	100.0	80.07	60.54	41.26	22.3	3.66
0.8	18.44	12.44	8.0	5.51	3.99	–
0.6	9.46	6.52	4.72	4.16	–	–
0.4	6.59	4.71	4.21	–	–	–
0.2	4.62	4.01	–	–	–	–
0.0	3.66	–	–	–	–	–

Table 12
CLR for RP_2.

$p:q$	0	0.2	0.4	0.6	0.8	1.0
1.0	100.0	80.12	60.41	41.07	22.06	3.15
0.8	17.49	11.97	7.95	5.11	3.83	–
0.6	8.74	5.79	4.55	3.82	–	–
0.4	5.83	4.47	4.0	–	–	–
0.2	4.42	3.73	–	–	–	–
0.0	3.32	–	–	–	–	–

Table 13
CLR for $PCUF_2$.

$p:q$	0	0.2	0.4	0.6	0.8	1.0
1.0	100.0	86.29	80.14	75.65	72.9	70.78
0.8	91.47	79.67	74.62	71.73	70.58	–
0.6	85.42	74.44	71.26	70.24	–	–
0.4	79.36	71.56	70.91	–	–	–
0.2	74.45	70.77	–	–	–	–
0.0	70.66	–	–	–	–	–

- Contrary to *CU*, the True Positive (p) and False Positive (q) rate of the classifier have a significant influence on the Co-Location Resistance of the placement algorithms. Tables 10–15 indicate the *CLR* values obtained by executing BF_2, WF_2, RP_2, $PCUF_2$, $PSSF_2$ and AZ_2 respectively for different values of p and q. The rows of table indicate different values of p whereas the columns indicate different values of q. When $p = 1$ and $q = 0$, the *CLR* of all algorithms is exactly 100%. This is because a binary classifier with $p = 1$ and $q = 0$ is equivalent to an ideal classifier which correctly classifies the categories of all the cloud users. In that case, all algorithms will place the VMs belonging to malicious users in PMs belonging to the group $G_{\mathcal{M}}$ and the VMs belonging to benign users in PMs which belong to the group $G_{\mathcal{B}}$. Thus,

Table 14
CLR for $PSSF_2$.

$p:q$	0	0.2	0.4	0.6	0.8	1.0
1.0	100.0	95.7	84.02	58.16	37.87	17.46
0.8	52.86	65.37	70.09	59.1	41.33	–
0.6	34.57	49.04	62.39	60.27	–	–
0.4	27.06	42.84	59.48	–	–	–
0.2	21.83	40.59	–	–	–	–
0.0	17.63	–	–	–	–	–

Table 15
CLR for $AZAR_2$.

$p:q$	0	0.2	0.4	0.6	0.8	1.0
1.0	100.0	80.46	61.21	42.75	24.75	6.74
0.8	25.84	19.09	14.14	10.75	8.75	–
0.6	14.81	11.54	9.39	8.97	–	–
0.4	11.27	9.3	8.74	–	–	–
0.2	8.45	8.63	–	–	–	–
0.0	6.74	–	–	–	–	–

there is no inter-mixing of VMs belonging to malicious and benign users.

- We also observe that as q increases from 0 to 1, the *CLR* decreases. This trend is similar for all algorithms; reason being that as q increases, more number of benign users are classified as malicious by classifier $\mathcal{C}_{p,q}$ and are therefore allocated PMs which belong to the group $G_{\mathcal{M}}$. This, in turn, leads to a higher probability of malicious co-location for those benign users who have been misclassified as malicious due to high False Positive rate of the classifier $\mathcal{C}_{p,q}$. Similarly, we observe that as p decreases from 1 to 0, the *CLR* of all algorithms decrease. This happens because a decrease in p leads to more number of malicious users being misclassified as benign. This, in turn, leads to a higher number of malicious users being allocated PMs belonging to the group $G_{\mathcal{B}}$, thereby increasing the probability of malicious co-location.
- Also, it is worth noting that $PCUF_2$ ensures better *CLR* than all other algorithms for almost all values of p and q. Except for the case when $p = 1, q = 0.2$ and $p = 1, q = 0.4$ where *PSSF* performs better than *PCUF*, the general relationship with respect to *CLR* is as follows:

  $PCUF > PSSF > AZ > WF > RP > BF$

  This shows that our proposed *PCUF* algorithm is effective in terms of *CU* and *CLR*, compared to other placement algorithms, even when user classification information is taken into account.

6. Conclusion and Directions for future work

In this paper, a co-location resistant VM placement algorithm called "Previously Co-Located Users First" has been described. We used Core Utilization and Co-Location Resistance metrics for quantifying the resource efficiency and security of VM placement algorithms. We performed an extensive theoretical and empirical analysis of our proposed algorithm using a large, real-world cloud trace. Our results indicate that the proposed algorithm can achieve much higher co-location resistance with little compromise in Core Utilization compared to existing standard and co-location resistant placement algorithms. In addition to this, we also performed a comparative study of different algorithmic solutions taking into account the case where the cloud provider has access to the predicted category of users. Specifically, our analysis shows that the Co-Location Resistance of placement algorithms is directly proportional to the True Positive rate and inversely proportional to the False Positive rate of the underlying classifier. We also observe that our proposed algorithm consistently achieves higher Co-Location Resistance compared to other placement algorithms in both scenarios.

Although our approach efficiently handles the initial VM placement problem, there are some open questions which need to be addressed. For instance, how should we defend against an adversary who manages to co-locate with the target user during initial VM placement. We believe that in order to effectively deal with such a scenario, it is necessary to incorporate live-migration algorithms. The migration algorithm should be triggered when the system notices a significant change in a user's cache behavior, which is likely to happen when the adversary conducts a side-channel attack. The challenge would then be to isolate either the adversary or the victim within the timeframe of the attack. Our future work would focus on the design and analysis of such secure live migration algorithms.

Another possible research direction would be to design classifiers which can be used to classify users (as benign or malicious) based on their cache statistics. These classifiers can then be used in conjunction with placement algorithms for SVP problem to provide a higher degree of isolation from malicious cloud users. It would also be interesting to design and analyze new placement algorithms for SVP problem which can provide better Co-Location Resistance by accounting for parameters like True Positive and False Positive rate of the underlying classifier.

Acknowledgment

This research has been supported via the Academic Research Fund (AcRF) Tier 1 Grant RG135/18.

Conflict of interest statement

None.

Declaration of competing interest

The authors declared that they had no conflicts of interest with respect to their authorship or the publication of this article.

References

[1] T. Ristenpart, E. Tromer, H. Shacham, S. Savage, Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds, in: Proceedings of the 16th ACM Conference on Computer and Communications Security, ACM, 2009, pp. 199–212.

[2] Y. Zhang, A. Juels, M.K. Reiter, T. Ristenpart, Cross-VM side channels and their use to extract private keys, in: Proceedings of the 2012 ACM Conference on Computer and Communications Security, ACM, 2012, pp. 305–316.

[3] Z. Wu, Z. Xu, H. Wang, Whispers in the hyper-space: high-speed covert channel attacks in the Cloud, in: USENIX Security Symposium, 2012, pp. 159–173.

[4] M. Lipp, M. Schwarz, D. Gruss, T. Prescher, W. Haas, S. Mangard, P. Kocher, D. Genkin, Y. Yarom, M. Hamburg, Meltdown, 2018, arXiv preprint arXiv:1801.01207.

[5] P. Kocher, D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher, M. Schwarz, Y. Yarom, Spectre attacks: Exploiting speculative execution, 2018, arXiv preprint arXiv:1801.01203.

[6] T. Zhang, Y. Zhang, R.B. Lee, Cloudradar: A real-time side-channel attack detection system in clouds, in: International Symposium on Research in Attacks, Intrusions, and Defenses, Springer, 2016, pp. 118–140.

[7] M. Alam, S. Bhattacharya, D. Mukhopadhyay, S. Bhattacharya, Performance counters to rescue: A machine learning based safeguard against micro-architectural side-channel-attacks.

[8] F. Ahamed, S. Shahrestani, B. Javadi, S. Garg, Developing security profile for virtual machines to ensure secured consolidation: conceptual model, in: Proceedings of the 13th Australasian Symposium on Parallel and Distributed Computing (AusPDC 2015), Held in Parramatta, Sydney, Australia, 27-30 January, 2015.
[9] Y. Azar, S. Kamara, I. Menache, M. Raykova, B. Shepard, Co-location-resistant clouds, in: Proceedings of the 6th Edition of the ACM Workshop on Cloud Computing Security, ACM, 2014, pp. 9–20.
[10] E. Cortez, A. Bonde, A. Muzio, M. Russinovich, M. Fontoura, R. Bianchini, Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms, in: Proceedings of the 26th Symposium on Operating Systems Principles, ACM, 2017, pp. 153–167.
[11] Z. Guo, Z. Duan, Y. Xu, H.J. Chao, JET: Electricity cost-aware dynamic workload management in geographically distributed datacenters, Comput. Commun. 50 (2014) 162–174.
[12] Z. Guo, S. Hui, Y. Xu, H.J. Chao, Dynamic flow scheduling for power-efficient data center networks, in: Quality of Service (IWQoS), 2016 IEEE/ACM 24th International Symposium on, IEEE, 2016, pp. 1–10.
[13] B.C. Vattikonda, S. Das, H. Shacham, Eliminating fine grained timers in xen, in: Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop, ACM, 2011, pp. 41–46.
[14] Z. Wang, R.B. Lee, A novel cache architecture with enhanced performance and security, in: Proceedings of the 41st Annual IEEE/ACM International Symposium on Microarchitecture, IEEE Computer Society, 2008, pp. 83–93.
[15] F. Liu, R.B. Lee, Random fill cache architecture, in: Microarchitecture (MICRO), 2014 47th Annual IEEE/ACM International Symposium on, IEEE, 2014, pp. 203–215.
[16] D. Page, Partitioned Cache architecture as a side-channel defence mechanism, IACR cryptology eprint archive, 2005, 280.
[17] Z. Wang, J. Wu, Z. Guo, G. Cheng, H. Hu, Secure virtual network embedding to mitigate the risk of covert channel attacks, in: Computer Communications Workshops (INFOCOM WKSHPS), 2016 IEEE Conference on, IEEE, 2016, pp. 144–145.
[18] P. Li, D. Gao, M.K. Reiter, Stopwatch: a cloud architecture for timing channel mitigation, ACM Trans. Inf. Syst. Secur. (TISSEC) 17 (2) (2014) 8.
[19] V. Varadarajan, T. Ristenpart, M.M. Swift, Scheduler-based Defenses against Cross-VM Side-channels, in: USENIX Security Symposium, 2014, pp. 687–702.
[20] Y. Zhang, M.K. Reiter, Düppel: retrofitting commodity operating systems to mitigate cache side channels in the cloud, in: Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, ACM, 2013, pp. 827–838.
[21] E. Pattuk, M. Kantarcioglu, Z. Lin, H. Ulusoy, Preventing cryptographic key leakage in cloud virtual machines, in: USENIX Security Symposium, 2014, pp. 703–718.
[22] Y. Han, T. Alpcan, J. Chan, C. Leckie, Security games for virtual machine allocation in cloud computing, in: International Conference on Decision and Game Theory for Security, Springer, 2013, pp. 99–118.
[23] Y. Han, J. Chan, T. Alpcan, C. Leckie, Using virtual machine allocation policies to defend against co-resident attacks in cloud computing, IEEE Trans. Dependable Secure Comput. 14 (1) (2017) 95–108.
[24] M. Berrima, A.K. Nasr, N. Ben Rajeb, Co-location resistant strategy with full resources optimization, in: Proceedings of the 2016 ACM on Cloud Computing Security Workshop, ACM, 2016, pp. 3–10.
[25] Y. Qiu, Q. Shen, Y. Luo, C. Li, Z. Wu, A secure virtual machine deployment strategy to reduce co-residency in cloud, in: Trustcom/BigDataSE/ICESS, 2017 IEEE, IEEE, 2017, pp. 347–354.
[26] Z. Afoulki, A. Bousquet, J. Rouzaud-Cornabas, A security-aware scheduler for virtual machines on iaas clouds, 2011, http://www.univ-orleans.fr/lifo/prodsci/rapports/RR/RR2011/RR-2011-08.pdf.
[27] Y. Zhang, M. Li, K. Bai, M. Yu, W. Zang, Incentive compatible moving target defense against vm-colocation attacks in clouds, in: IFIP International Information Security Conference, Springer, 2012, pp. 388–399.
[28] S.-J. Moon, V. Sekar, M.K. Reiter, Nomad: Mitigating arbitrary cloud side channels via provider-assisted migration, in: Proceedings of the 22nd Acm Sigsac Conference on Computer and Communications Security, ACM, 2015, pp. 1595–1606.
[29] M. Chiappetta, E. Savas, C. Yilmaz, Real time detection of cache-based side-channel attacks using hardware performance counters, Appl. Soft Comput. 49 (2016) 1162–1174.
[30] D. Kusic, J.O. Kephart, J.E. Hanson, N. Kandasamy, G. Jiang, Power and performance management of virtualized computing environments via lookahead control, Cluster Comput. 12 (1) (2009) 1–15.
[31] AWS EC2 Dedicated Instance, https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/dedicated-instance.html.
[32] T. Fawcett, An introduction to ROC analysis, Pattern Recognit. Lett. 27 (8) (2006) 861–874.

Mr. Amit Agarwal is a Computer Science graduate from BITS Pilani K.K. Birla Goa Campus, India. He is currently working as a software engineer at PayPal. His main areas of research include Computer Security, Artificial Intelligence, Data Structures and Algorithms.

Dr. Ta Nguyen Binh Duong is currently a regular faculty (Lecturer) in the School of Computer Science and Engineering (SCSE), Nanyang Technological University (NTU), Singapore. He obtained his Ph.D. in Computer Science from NTU Singapore. Previously, he was a Research Scientist with A*STAR Institute of High Performance Computing, and a Research Fellow with SCSE, NTU and University College Cork, Ireland. His main areas of expertise include distributed computing, machine learning, distributed simulations, and computer networking.