12-2018

Co-location resistant virtual machine placement in cloud data centers

Amit AGARWAL
Birla Institute of Technology and Science (BITS)

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

Published in 2018 International Conference on Parallel and Distributed Systems (ICPADS), Article number 8644849, pp. 61-68.
https://doi.org/10.1109/PADSW.2018.8644849

Co-Location Resistant Virtual Machine Placement in Cloud Data Centers

Amit Agarwal
Department of Computer Science, Goa Campus
BITS, Pilani
Goa, India
f20140403@goa.bits-pilani.ac.in

Ta Nguyen Binh Duong
School of Computer Science and Engineering
Nanyang Technological University
Singapore
donta@ntu.edu.sg

***Abstract*—Due to increasing number of avenues for conducting cross-virtual machine (VM) side-channel attacks, the security of public IaaS cloud data centers 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. To reduce the probability of malicious co-location, we propose a novel VM placement algorithm called "Previously Co-Located Users First". We perform a theoretical and empirical analysis of our proposed algorithm to evaluate its resource 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.**

***Keywords*-data centers; cloud security; co-location attacks; virtual machine placement;**

I. Introduction

With the advent of cloud services, users can now request computing resource 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 consolidating and renting out such hardware to the users. In IaaS clouds, on-demand computing 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 [1]. These algorithms are usually designed with the aim of scheduling incoming VM requests on PMs in a way that maximizes 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 (eg. cores, memory etc) of the PM across multiple VMs which are running on that PM. At the same time, it is the task of the hypervisor to enforce a strong isolation between different VMs which are running on the same PM. This is to ensure that each VM's private data is inaccessible to other co-located VMs.

Although cloud computing service has proven 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 side-channels present in shared physical resources, e.g., Last Level Cache.

Ristenpart et al. [2] addressed the feasibility of such attacks in Amazon EC2 cloud services by demonstrating techniques to perform cloud cartography, co-location checks and side-channel data leakage. Zhang et al. [3] demonstrated the feasibility of side-channel attacks to extract private keys. In [4], the authors described an attack framework using Flush-Reload strategy to conduct side-channel attacks and steal sensitive data in PaaS clouds. Wu et al. [5] designed and implemented a high-bandwidth (over 700 bps) covert channel by leveraging the use of memory bus.

In light of such developments in side-channel attack vectors and newly discovered vulnerabilities such as Meltdown [6] and Spectre [7], it has become critical to design effective defense measures. It is important to note that an essential prerequisite for any of such attacks is physical co-location, i.e., a malicious user first needs to successfully co-locate his VM with the target'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 novel strategies to make placement decisions which not only optimize the utilization of cloud resources, but also guarantee some level of security against co-location based attacks. With this idea in mind, we make the following contributions in this paper:

- We describe a metric called "Co-Location Resistance" for quantifying the security of cloud against co-location based attacks which is a generalization(to account for multiple cloud users) of a metric earlier proposed [1].
- We design a new algorithm called "Previously Co-Located Users First" with dual-objective of maximizing cloud's co-location resistance and resource utilization.

- We theoretically analyze the expected co-location resistance that is provided by our algorithm.
- We perform an extensive empirical analysis of our proposed algorithm with other commonly used placement algorithms using the recently released trace of workloads from Microsoft Azure [8].

II. Background and Related Work

A. Reducing information leakage through existing side channels

The primary aim of the works in this category is to modify the infrastructure (hardware, network, OS, hypervisor, etc.) in a way which either eliminates or reduces the information leakage through side channels. Ideas [9] [10] [11] [12] have been proposed to redesign the CPU cache architecture in order to defend against specific side-channel attacks.

Li et al. [13] proposed a hypervisor-based defense system which aims to obfuscate the leaked timing information on IaaS clouds by using 3 replicas of each guest VM, and only permitting the observation of aggregate timing information of these VMs. Techniques such as reducing VM preemption frequency [14], modifying RDTSC instruction [15], and cryptographic key partitioning across multiple VMs [16] have also been proposed as a possible defense against cache-based side channel attacks.

However, most of the defense approaches in this category suffer from two major limitations: i) they require major changes to the existing cloud infrastructure including, but not limited to, hypervisor, guest OS and physical hardware, and ii) they do not ensure security against currently unknown side-channels.

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

The primary aim of the works in this category is to design ways to prevent or reduce the probability of co-location of a malicious user with a benign user, which is an essential prerequisite for conducting co-location based attacks. Azar et al. [1] proposed a formal model for analyzing secure placement algorithms, and designed a Random VM placement strategy to reduce co-location probability. Han et al. [17] proposed a game-theoretic based model for comparing the security of different VM placement policies against co-location attacks. In [18], 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.

In [19] [20], trust relationship among cloud users has been considered wherein each user is given the freedom to choose his own set of adversaries or trusted users; and this is taken into account while making VM placement decisions. VM migration based defenses [21] [22] have been proposed to restrict co-residency and limit the amount of information leakage due to side-channel attacks.

Compared to hardware-based defenses described in Section II-A, the above strategies might be more feasible to be adopted in cloud environments for two main reasons: i) unlike hardware based defenses, they do not require any changes to existing cloud infrastructure, and ii) they are likely to be more resilient against arbitrary and currently unknown side-channel attacks. However, they do have some shortcomings: i) modified placement algorithms [1] significantly affect the resource utilization of cloud data center, ii) migration-based defenses [21] [22] incur extra network cost and application overhead, and iii) the effect of many proposed strategies, e.g., [1], [23], and [24], have not been studied on large-scale, real-world cloud workloads like the Azure traces released in 2017 [8].

To address these issues, a co-location resistant placement algorithm called "Previously Co-Located Users First" with a dual-objective (maximizing resource utilization and co-location security) has been proposed in this paper. We evaluate the algorithm both theoretically and empirically. An extensive, comparative analysis of the proposed algorithm and existing placement algorithms have also been carried out with the Azure cloud workload traces [8].

III. Problem Formulation

A. Assumptions and Threat Model

Here we list down some of the assumptions with respect to the capabilities of the cloud provider (CP) :

- The CP has no prior knowledge about which users are malicious.
- 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 doesn't use any migration algorithm to relocate an already allocated VM.

Here we list down some of the assumptions with respect to the capabilities of a malicious user (MU) :

- Like all other cloud users, each MU is in sole control of his own workload. He is free to decide the timing, core and memory requirement of his VM requests.
- All MU have the ability to compromise a benign user by leaking information through arbitrary side-channels after a successful physical co-location with the benign user's VM.
- Multiple MU can coordinate among themselves in order to compromise a particular benign user.

B. Notations

Table I lists some of the notations to be used later in this paper:

Table I: Notations

Notation	Description
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 in P_i
$Users(P_i)$	Set of all users whose VMs are resident on P_i
$CoLocated(U_i)$	Set of all users which have been co-located with user U_i at least once
l_{V_i}	Life-time(in seconds) of V_i which is the difference between V_i's request-time and termination-time
l_{P_i}	Life-time(in seconds) of P_i which is the total time during which P_i hosts at least one VM

C. Performance Metrics

1) Core Utilization (CU): The standard metric used to evaluate online bin packing algorithms is the "number of bins used". In [1], the authors pointed out that "number of bins used" ("number of physical servers" in our problem) is not an accurate metric for the VM placement problem as it doesn't capture the actual amount of time for which the physical resources have been expended in an online VM placement scenario.

In [25], the authors showed that the energy consumption by the CPU cores in PMs exceeds all the other resources. Therefore, to evaluate the performance of different placement algorithms with respect to resource usage, we use Core Utilization which is defined as 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.

$$CU = \frac{\sum_{all\ V} V_i^c \times l_{Vi}}{\sum_{all\ P} P_i^c \times l_{Pi}} \quad (1)$$

Note that the CU value 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.

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. [1] introduced the idea of "Single CL-resistance" wherein the adversary is only interested in co-locating his VM with at least one of the target VMs. Generalizing that idea to a cloud environment having multiple benign and malicious users, 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 can also be interpreted as the probability of a randomly chosen benign cloud user to be SAFE.

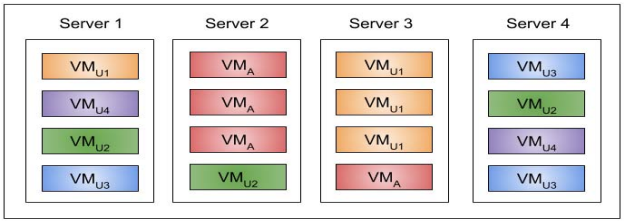

Figure 1: Allocation outcome of an arbitrary placement algorithm in which the malicious user (red) manages to co-locate his VMs with VMs belonging to benign users U_1 (orange) and U_2 (green). Users U_3 (blue) and U_4 (violet) are SAFE as none of their VMs have been co-located with the malicious user.

$$CLR = \frac{Total\ no.\ of\ benign\ users\ who\ are\ SAFE}{Total\ no.\ of\ benign\ users} \quad (2)$$

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 now illustrate an example to explain CLR metric. Assume a cloud environment consisting of 4 benign users(U_1, U_2, U_3, 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 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 $\frac{2}{4}$.

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

IV. Proposed Approach

A. Proposed Algorithm

We now propose a placement algorithm called "Previously Co-Located Users First" ($PCUF$) with the aim of maximizing CU and CLR of the cloud. The algorithm 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. A summary of the working of our algorithm is as follows:

- 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 user, we construct a list of $eligible\_pms$ by including all those PMs P_j from $live\_pms$ which 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. P_j has resident users which 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 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 not 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 user to get co-located with an adversary. While deallocating a VM V_i from PM P_j, we check whether P_j has any remaining VMs after deallocating V_i. If there are no remaining VMs on P_j, we remove P_j from $live\_pms$ and insert it into $empty\_pms$.

B. Theoretical Analysis

We will now describe a formula for $CLR_{pcuf}^{theoretical}$ which represents the expected fraction of benign users who will remain SAFE throughout the entire VM placement phase assuming $PCUF$ is used as the placement algorithm. We have considered a cloud environment consisting of N users out of which N_m users are malicious and N_b users are benign. To simplify our analysis, we have assumed 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. The Co-Location resistance of $PCUF$ has been derived using discrete probabilistic analysis[1] and is described by (3)

$$\begin{aligned} CLR_{pcuf}^{theoretical} &= \tfrac{1}{N} \\ &\times \textstyle\sum_{i=1}^{N} \left(\left(\tfrac{N_b - 1}{N-1} \right)^{min(i-1,\, k)} \right. \\ &\times \left. \textstyle\prod_{j=i+1}^{N} \left(1 - \tfrac{N_m}{N-1} \times \tfrac{min(j-1,\, k)}{j-1} \right) \right) \end{aligned} \tag{3}$$

Given N, P_m and k (avg. no. of users that a new user gets co-located with during his first VM allocation), we can estimate the CLR of $PCUF$ strategy using (3). 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 (3), we find that our estimated CLR is 67.43%. In Section VI-B, we evaluate the accuracy of our derived CLR by comparing it with empirical results.

V. 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. [8]. 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 II summarizes some statistics related to the dataset which we have used for our evaluation.

Table II: 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 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 III summarizes the statistics related to VM core and memory configuration.

[1]Due to space constraint, we have to omit the detailed analysis

Table III: 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

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 are mapped to benign 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 III-C. 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.

VI. Results and Analysis

A. Comparison of CU and CLR among different standard placement algorithms

We compare $PCUF$ with 3 most commonly used VM placement policies - Best-Fit (BF), Worst-Fit (WF) and Random Placement (RP). BF assigns an incoming VM to a live PM having the least remaining free cores, WF assigns it to a live PM having the most remaining free cores, RP assigns it to a live PM selected uniformly at random. 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. Fig. 2 and Fig. 3 depict the CLR and CU respectively, obtained by executing 4 different algorithms on the Azure workload with varying P_m. We make the following observations:

- Fig. 3 indicates that BF guarantees best CU whereas WF has the least CU among the 4 algorithms. $PCUF$ is less efficient (in terms of CU) than BF and RP by 14.25% and 5.68% respectively and performs better than WF by 2.55%. Note that CU value is independent of P_m because the algorithm has no knowledge about the category of any user.
- Fig. 2 indicates that $PCUF$ clearly outperforms all other algorithms in terms of CLR for all P_m. In fact, with more than 20% malicious users in the cloud, the CLR of BF, WF and RP is almost close to 0. This illustrates that standard VM placement algorithms provide almost negligible CLR compared to $PCUF$.
- CLR of all 4 algorithms decreases from 100 when $P_m = 0\%$ to approximately 0 as $P_m \to 100\%$. Increasing P_m increases the no. of malicious VMs which in turn increases the likelihood of a benign user's VM to get co-located with at least one malicious VM.

From these observations, we conclude that $PCUF$ provides much higher CLR compared to other standard placement algorithms with minimal compromise in CU.

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

We compare the $CLR_{pcuf}^{theoretical}$ given by (3) 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 users which we have not accounted for in our theoretical analysis.

C. Comparison of CU and CLR among different co-location resistant placement algorithms

We compare $PCUF$ with 3 other co-location resistant placement algorithms: i) Amazon's Dedicated Instance placement [26] which we will denote as DI, ii) CLR placement strategy proposed by Azar et al. [1] which we will denote as AZ, and iii) Previously Selected Servers First ($PSSF$) strategy proposed by Han et al. [18].

In DI strategy, two VMs that belong to different AWS accounts are never co-located on the same PM. Among the PMs owned by U_j, we consolidate U_j's VMs using a Best-Fit approach. AZ strategy maintains a fixed λ number of PMs as OPEN and every new incoming VM is randomly allocated to any one of these λ PMs. $PSSF$ strategy is parametrized using 2 variables - N^* and N_G. We ignore N^* in our evaluation as it is used for tuning workload balance which is not our objective in the current work.

Fig. 5 and Fig. 6 illustrate the CLR and CU respectively, obtained by executing different algorithms on the Azure workload with varying P_m. We have fixed the total number

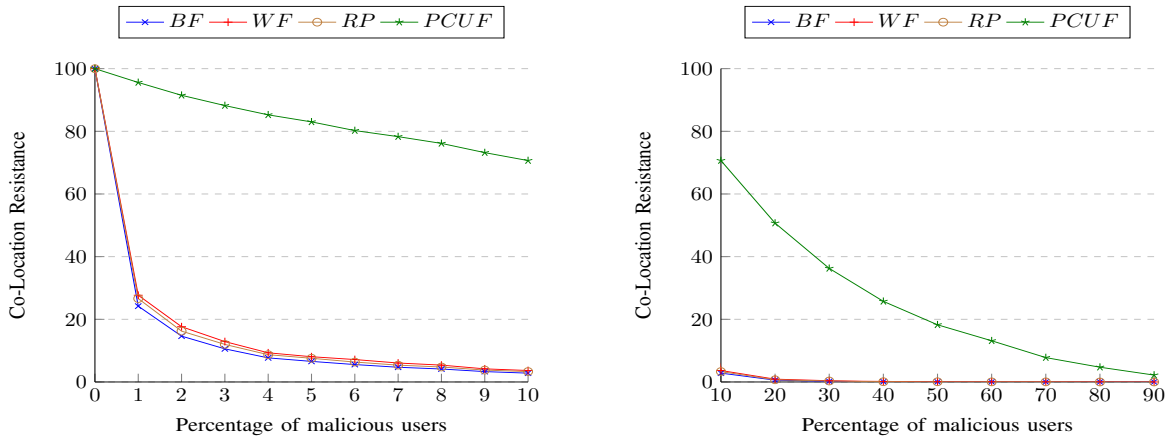

Figure 2: Co-Location Resistance (CLR) of $PCUF$ and other standard placement algorithms as a function of the percentage of malicious users (P_m). In (2a), P_m varies between 0 and 10 at increments of 1 whereas, in (2b), P_m varies between 10 and 90 at increments of 10. $PCUF$ consistently provides significantly higher CLR than other standard placement algorithms.

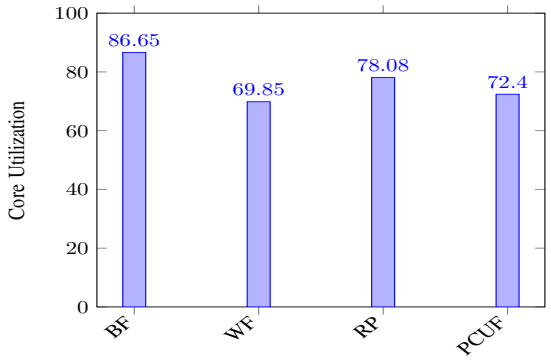

Figure 3: Core Utilization (CU) of $PCUF$ and different standard placement algorithms. In terms of CU, Best-Fit (BF) is most efficient whereas Worst-Fit (WF) is least efficient. The CU of $PCUF$ lies between that of WF and RP.

of PMs as 16,990 (same as the maximum number of simultaneously live VMs described in Table II). While evaluating AZ, we test for 4 different values of the parameter λ (5%, 10%, 15%, 20%) indicating the fraction of total PMs that are kept OPEN at all times. Similarly, while evaluating $PSSF$, we test for 4 different values of the parameter N_G (5%, 10%, 15%, 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. 5 indicates that as P_m increases, the CLR of all placement algorithms (except DI) decreases for the same reasons stated earlier in Section VI-A.
- Fig. 6 indicates that DI ranks third in terms of CU with a value of 61.9%. All placement algorithms, except $PCUF$ and $PSSF_{N_G=5\%}$, perform worse than DI in terms of CU. 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. 5 indicates, they might not be useful as their CU values are too low.
- Although CU of $PCUF$ is less than that of $PSSF_{N_G=5\%}$ by 1.69%, $PCUF$ consistently provides a significantly higher CLR compared to $PSSF_{N_G=5\%}$ for all values of P_m.

From these observations, we conclude that $PCUF$ achieves a better balance between CU and CLR compared to existing co-location resistant placement algorithms.

VII. Conclusion

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.

Although our approach efficiently handles the initial VM placement problem, open challenges such as defending against an adversary who manages to co-locate with the target user during initial VM placement still needs to be addressed. To effectively deal with such a scenario, it is necessary to incorporate live-migration algorithms. Another

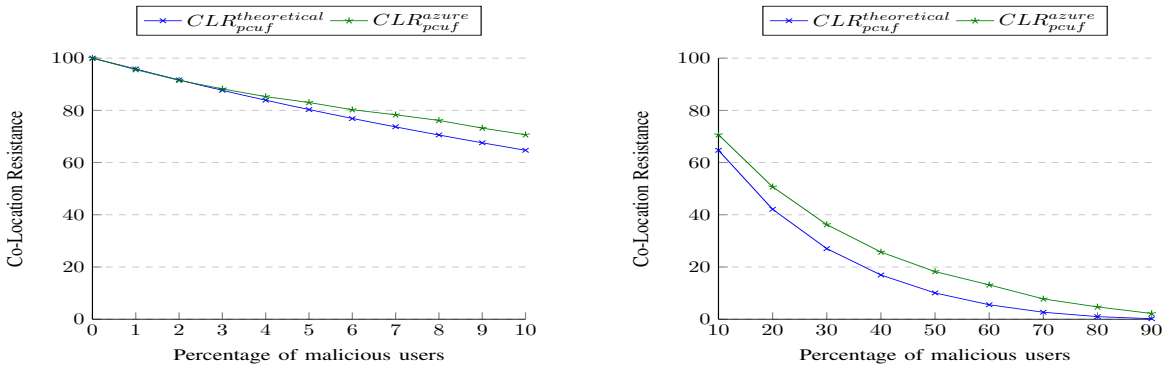

Figure 4: Theoretically derived CLR for $PCUF$ ($CLR^{theoretical}_{pcuf}$) and the empirical CLR obtained by executing $PCUF$ on Azure dataset (CLR^{azure}_{pcuf}) as a function of percentage of malicious users (P_m). In 4a, P_m varies between 0 and 10 at increments of 1 whereas, in 4b, P_m varies between 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.

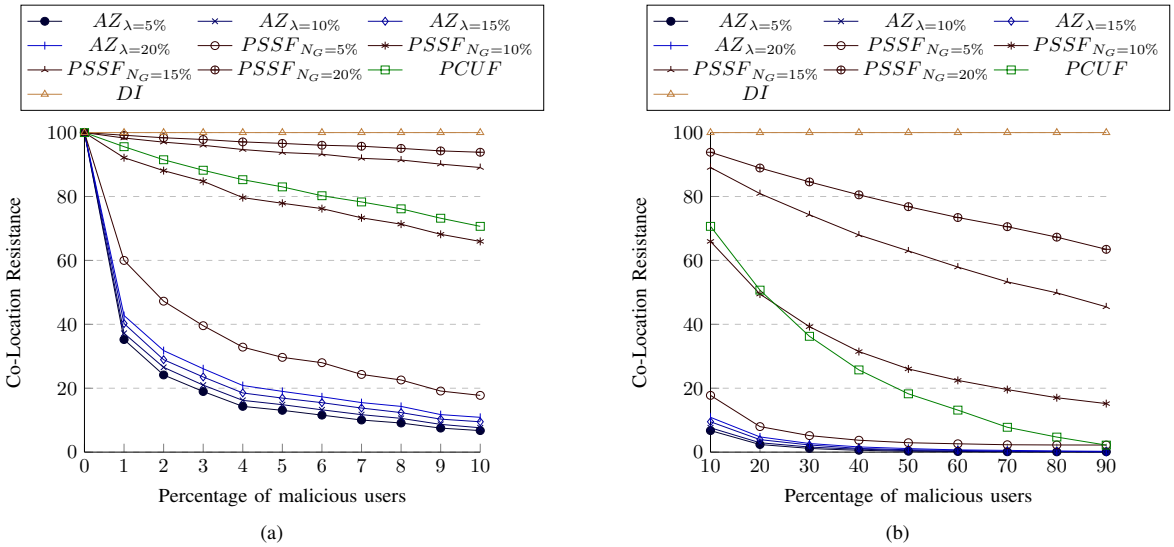

Figure 5: Co-Location Resistance (CLR) of different co-location resistant placement algorithms - $PCUF$, DI, AZ (4 distinct λ values) and $PSSF$ (4 distict N_g values) as a function of the percentage of malicious users (P_m). In (5a), we vary P_m between 0 and 10 at increments of 1 whereas, in (5b), we vary the P_m between 10 and 90 at increments of 10.

interesting research direction would be to design classifiers to categorize users (as benign or malicious) based on their cache statistics. These classifiers can then be used in conjunction with placement algorithms to provide a higher degree of isolation from malicious users.

ACKNOWLEDGMENT

The research has been supported via the Academic Research Fund (AcRF) Tier 1 Grant RG121/15.

REFERENCES

[1] Y. Azar, S. Kamara, I. Menache, M. Raykova, and B. Shepard, "Co-location-resistant clouds," in *Proceedings of the 6th Edition of the ACM Workshop on Cloud Computing Security*, pp. 9–20, ACM, 2014.

[2] 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 *Proceedings of the 16th ACM conference on Computer and communications security*, pp. 199–212, ACM, 2009.

[3] Y. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Cross-vm side channels and their use to extract private keys," in

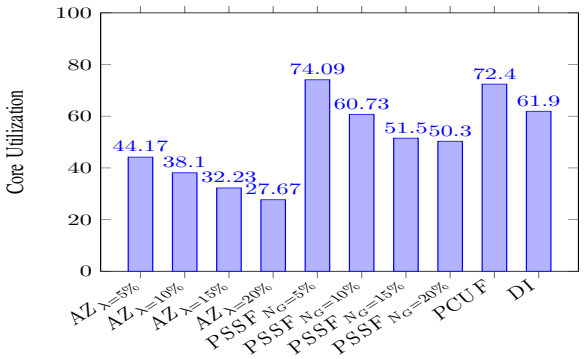

Figure 6: Core Utilization (CU) of $PCUF$ and other co-location resistant placement algorithms. $PCUF$ provides a significantly higher CU compared to all other placement algorithms and is second only to $PSSF_{N_G=5\%}$

Proceedings of the 2012 ACM conference on Computer and communications security, pp. 305–316, ACM, 2012.

[4] Y. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Cross-tenant side-channel attacks in paas clouds," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pp. 990–1003, ACM, 2014.

[5] Z. Wu, Z. Xu, and H. Wang, "Whispers in the hyper-space: High-speed covert channel attacks in the cloud.," in *USENIX Security symposium*, pp. 159–173, 2012.

[6] M. Lipp, M. Schwarz, D. Gruss, T. Prescher, W. Haas, S. Mangard, P. Kocher, D. Genkin, Y. Yarom, and M. Hamburg, "Meltdown," *ArXiv e-prints*, Jan. 2018.

[7] P. Kocher, D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher, M. Schwarz, and Y. Yarom, "Spectre Attacks: Exploiting Speculative Execution," *ArXiv e-prints*, Jan. 2018.

[8] E. Cortez, A. Bonde, A. Muzio, M. Russinovich, M. Fontoura, and 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*, pp. 153–167, ACM, 2017.

[9] Z. Wang and R. B. Lee, "A novel cache architecture with enhanced performance and security," in *Proceedings of the 41st annual IEEE/ACM International Symposium on Microarchitecture*, pp. 83–93, IEEE Computer Society, 2008.

[10] F. Liu and R. B. Lee, "Random fill cache architecture," in *Microarchitecture (MICRO), 2014 47th Annual IEEE/ACM International Symposium on*, pp. 203–215, IEEE, 2014.

[11] D. Page, "Partitioned cache architecture as a side-channel defence mechanism.," *IACR Cryptology ePrint archive*, vol. 2005, no. 280, 2005.

[12] Y. Zhang and 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*, pp. 827–838, ACM, 2013.

[13] P. Li, D. Gao, and M. K. Reiter, "Stopwatch: a cloud architecture for timing channel mitigation," *ACM Transactions on Information and System Security (TISSEC)*, vol. 17, no. 2, p. 8, 2014.

[14] V. Varadarajan, T. Ristenpart, and M. M. Swift, "Scheduler-based defenses against cross-vm side-channels.," in *USENIX Security Symposium*, pp. 687–702, 2014.

[15] B. C. Vattikonda, S. Das, and H. Shacham, "Eliminating fine grained timers in xen," in *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, pp. 41–46, ACM, 2011.

[16] E. Pattuk, M. Kantarcioglu, Z. Lin, and H. Ulusoy, "Preventing cryptographic key leakage in cloud virtual machines.," in *USENIX Security Symposium*, pp. 703–718, 2014.

[17] Y. Han, T. Alpcan, J. Chan, and C. Leckie, "Security games for virtual machine allocation in cloud computing," in *International Conference on Decision and Game Theory for Security*, pp. 99–118, Springer, 2013.

[18] Y. Han, J. Chan, T. Alpcan, and C. Leckie, "Using virtual machine allocation policies to defend against co-resident attacks in cloud computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 14, no. 1, pp. 95–108, 2017.

[19] Z. Afoulki, A. Bousquet, and J. Rouzaud-Cornabas, "A security-aware scheduler for virtual machines on iaas clouds." http://www.univ-orleans.fr/lifo/prodsci/rapports/RR/RR2011/RR-2011-08.pdf, 2011.

[20] V. Natu and T. N. B. Duong, "Secure virtual machine placement in infrastructure cloud services," in *10th IEEE Conference on Service-Oriented Computing and Applications*, pp. 26–33, 2017.

[21] Y. Zhang, M. Li, K. Bai, M. Yu, and W. Zang, "Incentive compatible moving target defense against vm-colocation attacks in clouds," in *IFIP International Information Security Conference*, pp. 388–399, Springer, 2012.

[22] S.-J. Moon, V. Sekar, and 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*, pp. 1595–1606, ACM, 2015.

[23] M. Berrima, A. K. Nasr, and N. Ben Rajeb, "Co-location resistant strategy with full resources optimization," in *Proceedings of the 2016 ACM on Cloud Computing Security Workshop*, pp. 3–10, ACM, 2016.

[24] Y. Qiu, Q. Shen, Y. Luo, C. Li, and Z. Wu, "A secure virtual machine deployment strategy to reduce co-residency in cloud," in *Trustcom/BigDataSE/ICESS, 2017 IEEE*, pp. 347–354, IEEE, 2017.

[25] D. Kusic, J. O. Kephart, J. E. Hanson, N. Kandasamy, and G. Jiang, "Power and performance management of virtualized computing environments via lookahead control," *Cluster computing*, vol. 12, no. 1, pp. 1–15, 2009.

[26] "AWS EC2 Dedicated Instance." https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/dedicated-instance.html.