Co-location Resistant Strategy with Full Resources Optimization

Mouhebeddine Berrima
FSM, University of Monastir
LIP2, University of
Tunis-ElManar, Tunisia
berrima.mouheb@gmail.com

Aïcha Katajina Nasr
LIP2, University of
Tunis-ElManar, Tunisia
aichanassr@gmail.com

Narjes Ben Rajeb
INSAT, University of Carthage
LIP2, University of
Tunis-ElManar, Tunisia
narjes.benrajeb@gmail.com

ABSTRACT

In the public clouds, an adversary can co-locate his or her virtual machines (VMs) with others on the same physical servers to start an attack against the integrity, confidentiality or availability. The one important factor to decrease the likelihood of this co-location attack is the VMs placement strategy. However, a co-location resistant strategy will compromise the resources optimization of the cloud providers. The tradeoff between security and resources optimization introduces one of the most crucial challenges in the cloud security. In this work we propose a placement strategy allowing the decrease of co-location rate by compromising the VM startup time instead of the optimization of resources. We give a mathematical analysis to quantify the co-location resistance. The proposed strategy is evaluated against the abusing placement locality, where the attack and target VMs are launched simultaneously or within a short time window. Referring to EC2 placement strategy, the best co-location resistant strategy out of the existing public cloud providers strategies, our strategy decreases enormously the co-location attacks with a slight VM startup delay (relatively to the actual VM startup delay in the public cloud providers).

Keywords

Co-location attacks, abuse forcing strategy, parallel placement locality, virtual machines placement strategy.

1. INTRODUCTION

Cloud computing provides several benefits such as rapid and flexible deployment, reduced costs, scalability, pay-as-you-go economic model. The most important reason to reduce costs, for both user and cloud providers, is the use of multi-tenancy. Multi-tenancy is considered as one of the key characteristics of clouds and it applies to all three layers of a cloud: IaaS, PaaS and SaaS. At the infrastructure level, multi-tenancy enables sharing the same physical servers among different tenants, i.e. virtual machines of different users.

CCSW'16, October 28 2016, Vienna, Austria

DOI: http://dx.doi.org/10.1145/2996429.2996435

Multi-tenancy, however, introduces a security threats since a user cannot control which virtual machines are co-located on the same physical server. These threats arise primarily in public clouds in which the resources are shared by organizations that have potentially competing or conflicting interests, and thus, there is a motivation to unauthorized transfer of data and/or operations disruption.

While the public clouds enforce logical isolation among tenants, an adversary can break to start an attack, such as side-channel attack [13, 21], covert channel [18] and power attack [20], against the integrity, confidentiality and availability of other users co-located on the same hardware. To achieve the above attacks, the adversary needs to co-locate with at least one target VM. The co-location attack consists of two steps. In the first step, the adversary employs some strategies for launching (typically a large number of) VMs on the cloud. In the second step, each of these VMs attempts to perform co-location detection. The launch strategies of the first step exploits two placement locality observations [13]: (i) *Parallel placement locality*, in which the VMs of different users launched simultaneously or within a short time window are often co-located. (ii) *Sequential placement locality*, in which two VMs launched sequentially (the first terminated before launching the second) are often co-located. The straightforward way to prohibit the co-location attacks, is single-tenancy, i.e. the co-location is only allowed for the VMs of the same user, but this damages many benefits of the cloud. However, it is possible to decrease the likelihood of the co-location through the placement strategy of VMs, since it is one important factor that will influence the VMs placement.

The main objective of VMs placement strategies is to satisfy the constraints of the resources optimization policies, such as energy consumption minimization, network traffic minimization, migration time minimization, SLA violation minimization, etc [10, 12]. In the data centers, the important resource is the energy consumption (because of both its costs and its environment impact [8, 10]), which its efficiency mainly depends of the low utilization of the servers. The problem of allocating VMs to servers can be viewed as an instance of the well-known online bin packing problem. Many bin-packing heuristics have been adapted to the VM placement problem: First Fit [6], Best Fit [5], First Fit Decrasing [16, 17] etc.

The consolidation of the VMs in few servers increases the probability of the co-location attacks, since by exploiting the placement localities the adversary can predict the servers on which his VMs will be hosted. When the placement strategy

can be exploitable by the adversary, we say that the cloud suffers from a *placement vulnerability*. The trivial solution to mitigate these attacks consists in selecting uniformly at random the servers. The use of randomization disables the adversary to control the placement of his VMs on the servers, but it hugely hurts the optimization constraints of resources, and hence cannot be applied in the practice. The tradeoff between security and resources optimization is one of the difficult challenge in the academic and industrial research in the cloud security.

In this work, we consider the placement vulnerability problem in public clouds. We propose a placement strategy decreasing the probability of the co-location with full respect of the optimization constraints. Unlike the existing co-location resistant strategies [3, 4, 9] that randomly select the servers, here we use the randomization to select the VMs that will be assigned to the servers, while these latter are selected according to an optimization strategy. Hence, it will be difficult to predict which VMs will be co-located on the same physical server. For this to happen, we use a queue, that we call *mixing queue*, to place the incoming VM requests before the assignment to the servers. While the optimization constraints is respected, our strategy introduces a tradeoff between security and VM startup time. In our evaluation of the co-location resistance, we consider the abusing placement locality attack strategy, where the attack and target VMs are launched simultaneously or within a short time window. The evaluation is based on a mathematical analysis allowing us to quantify the co-location attacks. We argue that the delay incurred by the mixing queue, to ensure a 'certain level' of co-location resistance, is relatively not awkward according to the rate cloud usage of nowadays. Moreover, we compare our strategy with both EC2 and Y. Azar *et al* strategies. The comparison with EC2 strategy, the best co-location resistant strategy out of the existing public cloud providers strategies [15], allows us to conclude that our strategy can decrease enormously the co-location rate with a VM startup delay less than one minute.

The paper is organized as follows: in Section 2 we expose the works addressing the placement vulnerability problem, next in Section 3 we describe our placement strategy. Section 4 is devoted to the mathematical analysis. In Section 5 we evaluate the proposed strategy through the numerical results. Finally, in Section 6 we conclude and introduce some future works.

2. RELATED WORKS

Since the publication of T. Ristenpart *et al* [13] in 2008, in which the authors exposed the co-location attacks for the first time, the placement vulnerability problem became one of the main concerns in the cloud security. Even so, in the literature there are few works proposing co-location resistant placement strategies. The aim of such strategies is to assign VMs to physical servers so that it is difficult for an adversary to achieve a co-location.

In [3], the resistance is ensured by security policies expressed by the users, allowing them to choose their own adversaries to prohibit the co-location with their VMs. While the co-location is completely avoided, this strategy is not practice because the adversary can launch his VMs from many accounts, and thus it is impossible to identify the attack VMs. The strategy proposed in [9], improve the security by restricting the number of servers that one user can use. However, it is clear that this strategy damages the resources optimization, already the authors avoided the resources optimization problem.

The most interesting work is due to Y. Azar *et al* [4], since they consider the tradeoff between the co-location resistance and the resources optimization. Their algorithm extends the trivial solution by choosing randomly the physical server from a subset of all available servers. The cardinality of this subset expresses the compromise between the security and the resources optimization. Indeed, a large subset of servers improves the co-location resistance, but it damages the resources optimization, and vice versa.

The co-location analyzes performed on Amazon EC2 [15, 19], affirm that EC2 has mitigated its placement vulnerability during the last years : 7 or 8 co-residence with 20 attack VMs in 2008 [13], 1 co-residence with 160 attack VMs in 2012 [18], and 1 co-residence with more than 200 attack VMs in 2015 [19]. Despite EC2 does not announced its security strategy, the adjustment, as argued in [19], is evident because the expansion of available zones, physical servers per zone and the incoming VM requests gives more flexibility to place the VMs. However, Azure and GCE providers are more vulnerable to the co-location attacks [15].

3. THE PLACEMENT STRATEGY

Our strategy allows decreasing the probability of co-location attacks with full resources optimization, i.e. decreasing the number of active servers. The resources optimization is ensured by the selection of the servers according to an optimization strategy, while the co-location resistance is ensured by selecting uniformly at random the VMs from a queue with a predefined capacity. This queue is called the *mixing queue* because its role is to randomly mix the order of the VMs. It contains three sorts of VMs: attack, target, and other. Therefore, it is difficult to predict if a given VM will be co-located with the target VMs or the other VMs.

3.1 The algorithm

The pseudo-code of the algorithm implementing our strategy is described below. It is parameterized by the mixing queue capacity and the optimization strategy. Its input is a VM, and it outputs an assignment of VMs to the servers. We assume that there are enough servers to host all the VMs.

Algorithm 1: Secure-optimal placement algorithm

```
Input: VM
Output: assignment of VMs to the servers
Add VM to the mixing queue ;
if the mixing queue is not full then
    skip ;
else
    while the mixing queue is not empty do
        select a vm from the mixing queue uniformly at
        random ;
        place vm on a server according to
        OptimizationStrategy ;
```

The algorithm is launched whenever a VM request arrives. Before the placement of VMs on the servers, the algorithm inserts the incoming VM requests in the mixing queue. When the queue is filled, all the VMs are selected sequentially at random and placed on the servers according to the strategy conjointly used to optimize the resources.

In a preliminary investigation of the above algorithm, it appears that the co-location only depends on the number of target, attack and other VMs, as well as the servers capacity, but we do not see how the mixing queue capacity effects the co-location resistance. In fact, since the cloud is shared with many other users (a very large number of users), and in a given time interval the number of VMs is proportional to the number of the corresponding user accounts, we can conclude that the increase of the mixing queue capacity promotes the increasing number of the other VMs. Therefore, the greater the mixing queue capacity, the greater the number of the other VMs in the mixing queue (relatively to the target and attack VMs).

3.2 Co-location resistance Versus VM startup time

Nonetheless, the increasing of mixing queue capacity delays the VM startup. This means that our strategy introduces a tradeoff between the security and the VM startup time. The crucial question, to balance this tradeoff, is which value to choose for the mixing queue capacity ? The answer to this question relies on the VM requests rate, i.e. the number of launched VMs per time unit, because the greater the incoming VM requests rate, the shortest the VM startup delay. Unfortunately, we are not aware of cloud providers announcing their VM requests rate statistics. However, recently in 2015 [19], the authors exploited their co-location analysis on EC2 to investigate the business scale and the dynamic environment. Following their measurements, they observed that the number of live VMs is larger than 650K. Concerning the dynamism, what more interests us, they detected that there are more than 15K VMs that are changed (shutdown, newly booted or re-located) every 20 minutes, which indicates that EC2 is very dynamic with tens of VMs booted and shutdown every second. These observations, help us to approximate a realistic incoming VM requests rate.

The Amazon EC2 is the dominant public cloud provider [2], thus obviously it supports the higher VM requests rate. However, we want to evaluate our strategy with respect to almost all providers, even those having a low rate. Therefore, we consider a low rate of VM requests ranging between 1K and 3K per 10 minutes. Note that for the few next years, we expect that the strong growth of the public cloud usage (usage evolution from less than 5K in November 2006 to 50K in September 2009 [1], and to 650K in 2015 [19] in Amazon EC2) will mitigate the VM startup delay incurred by the mixing queue.

4. THEORETICAL ANALYSIS

In our analysis, we focus on the co-location resistance, since the evaluation of the resources optimization relies on the optimization strategy [5, 6, 16, 17]. We point out that the optimization strategy must be online to emulate the randomness of the incoming VM requests.

4.1 Estimation of the attacked VMs

To analyze and quantify the co-location resistance, we define a metric as "the average number of the attacked VMs". An attacked VM is a target VM co-located with at least one attack VM.

Assumptions. We consider a system composed of enough physical servers to host all the VMs. Without loss of generality, we consider the one-dimensional problem, and we assume that the servers have the same resources capacity, and each VM requires one resource unit. Moreover, we assume that the mixing queue capacity is multiple of the server capacity.

To calculate the average number of all attacked VMs, we start by determining the average number of the attacked VMs on one server. In the remaining of the paper, we denote by p the server capacity.

LEMMA 4.1. *Consider a mixing queue with a capacity of M, containing M_a attack VMs, M_t target VMs and M_o other VMs, such that $M = M_a + M_t + M_o$. According the Algorithm 1, the average number of the attacked VMs on the k-th server is:*

$$\mathcal{N}^k(M_a, M_t, M_o) = \frac{p.M_t^k}{M^k}.\left(1 - \frac{\binom{M_t^k+M_o^k-1}{p-1}}{\binom{M^k-1}{p-1}}\right)$$

with $M^k = M_a^k + M_t^k + M_o^k$ and $M_i^k = M_i - (k-1).p.\frac{M_i}{M}$ for $i \in \{a, t, o\}$.

PROOF. We consider the first server, i.e. $M_i^1 = M_i$ for $i \in \{a, t, o\}$, and let X be the random variable representing the number of the target VMs co-located with at least one attack VM. We have

$$\begin{aligned} Pr[X = i] &= Pr[\{M_t = i\} \cap \{M_a \geq 1\}] \\ &= \sum_{j=1}^{p-i} \frac{\binom{M_t}{i}.\binom{M_a}{j}.\binom{M_o}{p-i-j}}{\binom{M}{p}} \end{aligned}$$

The average number of the attacked VM is given by the expectation of X, then we have (below, some equalities use Vandermonde formula)

$$\begin{aligned} \mathcal{N}(M_a, M_t, M_o) &= \sum_{i=1}^{p-1} i.Pr[X = i] \\ &= \frac{1}{\binom{M}{p}}.\sum_{i=1}^{p-1} i.\binom{M_t}{i}.\sum_{j=1}^{p-i}\binom{M_a}{j}.\binom{M_o}{p-i-j} \\ &= \frac{1}{\binom{M}{p}}.\sum_{i=1}^{p-1} i.\binom{M_t}{i}.\left(\sum_{j=0}^{p-i}\binom{M_a}{j}.\binom{M_o}{p-i-j} - \binom{M_a}{0}.\binom{M_o}{p-i}\right) \\ &= \frac{1}{\binom{M}{p}}.\sum_{i=1}^{p-1} i.\binom{M_t}{i}.\left(\binom{M_a+M_o}{p-i} - \binom{M_o}{p-i}\right) \\ &= \frac{1}{\binom{M}{p}}.\left(\sum_{i=1}^{p-1} i.\binom{M_t}{i}.\binom{M_a+M_o}{p-i} - \sum_{i=1}^{p-1} i.\binom{M_t}{i}.\binom{M_o}{p-i}\right) \\ &= \frac{1}{\binom{M}{p}}.\left(\sum_{i=1}^{p-1} M_t.\binom{M_t-1}{i-1}.\binom{M_a+M_o}{p-i} - \sum_{i=1}^{p-1} M_t.\binom{M_t-1}{i-1}.\binom{M_o}{p-i}\right) \\ &= \frac{M_t}{\binom{M}{p}}.\left(\binom{M-1}{p-1} - \binom{M_o+M_t-1}{p-1}\right) \end{aligned}$$

$$= M_t.\left(\frac{p}{M} - \frac{\binom{M_o+M_t-1}{p-1}}{\binom{M}{p}}\right)$$
$$= \frac{p.M_t}{M}.\left(1 - \frac{\binom{M_t+M_o-1}{p-1}}{\binom{M-1}{p-1}}\right)$$

If we consider the k-th server, then we must remove the VMs that were placed on the (k-1) previous servers. Let Y_k be the random variable representing the number of the attack VMs placed on the k-th server. Y_k follows a binomial distribution $\mathcal{B}(i; p, \frac{M_a}{M})$, thus the average number of the attack VMs placed on the k-th server is $\mathbb{E}(Y_k) = p.\frac{M_a}{M}$. We show by induction on k that

$$M_a^k = M_a - (k-1).p.\frac{M_a}{M}$$

Base case: we have $k = 1$, since we consider the first server, then we get that $M_a^1 = M_a$ and we conclude.
Inductive step: assume that $k > 1$, so we have $M_a^k = M_a^{k-1} - p.\frac{M_a}{M}$, by induction hypothesis we get that

$$M_a^k = M_a - (k-2).p.\frac{M_a}{M} - p.\frac{M_a}{M}$$
$$= M_a - (k-1).p.\frac{M_a}{M}$$

Similarly, we can conclude for M_t^k and M_o^k. □

Based on the above result, we can deduce the average number of the attacked VMs on all the used servers.

Corollary 4.2. *Consider a mixing queue with a capacity of M, containing M_a attack VMs, M_t target VMs and M_o other VMs, such that $M = M_a + M_t + M_o$. The average number of the attacked VMs is*

$$\widetilde{\mathcal{N}}(M_a, M_t, M_o) = \sum_{k=1}^{\frac{M}{p}} \mathcal{N}^k(M_a^k, M_t^k, M_o^k)$$

with $M_i^k = M_i - (k-1).p.\frac{M_i}{M}$ for $i \in \{a, t, o\}$.

Next, we approximate the average number of the all attacked VMs. The approximation is based on the fact that the co-location probabilities on all the servers are approximately equals, because we always keep the same proportions of VMs. The practical results give a closed approximation. The aim of this approximation is to understand the weighting effect of M_a, M_t, M_o and p on the co-location attacks.

Proposition 4.3. *Consider a mixing queue with a capacity of M, containing M_a attack VMs, M_t target VMs and M_o other VMs, such that $M = M_a + M_t + M_o$. We have,*

$$\widetilde{\mathcal{N}}(M_a, M_t, M_o) \approx M_t.\left(1 - \left(\frac{M_t + M_o}{M}\right)^{p-1}\right)$$

As mentioned previously, the increase of the mixing queue capacity promotes the increasing number of the other VMs. However, in a given time interval, the number of the other VMs can be influenced by the adversary, by launching a very large number of VMs. Therefore, to be more accuracy in our evaluation, we must take into account the adversary strategy.

4.2 Analysis against abusing placement locality

In [13], the authors introduced two strategies to achieve co-location: (i) brute-forcing placement and (ii) abusing locality placement. (i) In brute-forcing placement strategy, that is effective for a large set of targets, the adversary launches his attack VMs over some long period. The attack has been performed with 1,785 attack VMs over 18 days, where the number of VMs at each launch is at most 20, and the set of the target VMs is of 1,686. (ii) Abusing placement locality is more effective since the adversary can achieve a high rate of co-location with a small set of target. This attack exploits parallel placement locality which means if two VMs are launched almost at the same time, then they are often placed on the same server. Thus, the adversary can achieve a high success rate of co-location if he launches his VMs simultaneously with or soon after the launch of the target VMs. The experiments in [13] have been performed with 20 attack VMs, launched 5 minutes after the launch of the set target of either 1, 10, or 20 VMs. Next, we assess the co-location resistance against abusing placement locality (APL) strategy, because it is the most efficient.
Adversary ability. We assume that the adversary can launch VMs as many as he needs simultaneously with the launch of the target. Note that this assumption is stronger (i.e promotes the attacks) than assuming that the attack VMs are launched soon after the target launch, because APL strategy exploits parallel placement locality.

Next, we show that even under this adversary ability, it is possible to decrease the co-location rate. Indeed, at the time of launching the target VMs, the mixing queue probably contains other VM requests inserted before those of the target. The worst case is when the target launches its VMs when the mixing queue is empty. Hence, the greater the mixing queue capacity, the lower the probability of such worst case. The next result gives an estimation of the average number of the attacked VMs against APL strategy.

Theorem 4.4. *Consider N_t target VMs and N_a attack VMs launched following the APL strategy. The average number of the attacked VMs by executing the Algorithm 1 parameterized by a mixing queue capacity of M is:*

$$\widetilde{\mathcal{N}}(min(N_a, (M - (N_t + N_o)), N_t, N_o)$$

avec $N_o = \lfloor \frac{M+1}{2} \rfloor - 1$.

Proof. The mixing queue filling is cyclic, i.e. the mixing queue is empty again whenever it is filled. At a target VMs launch, it is probably that some cases are occupied by other VM requests. Since there is not any condition on the arrival time of target VMs, the probability that the target VM requests will be inserted from the i-th case of the mixing queue (with $i = 1..M$) is $\frac{1}{M}$.
Let X be the random variable representing the case number from which the target VMs will be inserted. X satisfies uniform distribution over $\{1, 2 \ldots, M\}$, because the probabilities are equiprobable. Thus, the expected case is that of number $\lfloor \frac{M+1}{2} \rfloor$ (we consider the lower integer part because it increases the probability of co-location).
It follows that, the expected number of the other VMs in the mixing queue at the target VMs launch is $\lfloor \frac{M+1}{2} \rfloor - 1$. After the insertion of the target VM requests, it remains $(M - (N_t + \lfloor \frac{M+1}{2} \rfloor - 1))$ free cases, thus the adversary can

insert in the mixing queue the minimum between N_a and the remaining cases. □

In addition to the number of the attacked VMs, we can derive form the above theorem the number of the attack VMs needed to co-locate with a particular set of target VMs. Indeed, both metrics are equivalent in the sense that they allow us to evaluate and compare the effectiveness of the co-location strategies.

The best adversarial strategy is when the other VMs are evicted from the mixing queue by launching VMs soon before and after the target. In practice, launching VMs soon after the target VMs is realistic using some techniques [13]. However, the launch of the attack VMs soon before the target VMs is very difficult unless the adversary knows in advance the instant launch. This can be easily avoided if the target keeps secret the instant launch. One can even assume further that the adversary can predict the period in which the target VMs will be launched. In such case the adversary must launch a massive amount of VMs to fill up the mixing queue throughout the expected period, and consequently our strategy remains effective. Moreover, we speculate that the ability to insert sequentially a very large number of VM requests in the mixing queue is very stronger. Indeed, since the cloud is shared between a very large number of users, it is difficult that one monopolizes the cloud for a some long period, even from many accounts. This really needs a strong analysis to quantify the adversary ability.

5. NUMERICAL RESULTS AND EVALUATION AGAINST APL STRATEGY

In this section, we evaluate our strategy through the numerical results of the theoretical analysis against APL strategy. Moreover, we compare our strategy with both EC2 and Y.Azar *et al* strategies.

5.1 Effect of the mixing queue capacity

Our strategy is parameterized with the mixing queue capacity which effects both the co-location and the VM startup delay. The latter is also correlated with the rate of the incoming VM requests, the greater the VM requests rate, the shortest the VM startup delay. As argued in Section 3, we consider an incoming VM rate ranging between 1K and 3k per 10 minutes. Table 1 gives some samples of startup delay incurred by the buffering of the VMs requests in the mixing queue. We do not consider any other parameter can affect the startup delay.

Table 1: VM Startup delay (in minutes) incurred by the mixing queue

Rate \ M	1K	5K	10K	15K	20K
1K	10	50	100	150	200
3K	3.33	16.66	33.33	50	66.66

In [11], a study across Amazon EC2, Azure and Rackspace shows that the average VM startup time ranges between 1.5 minutes and 13 minutes according to the type, location and operating system. Thus, we think that increasing the startup time by few minutes to improve security is not awkward. On the other hand, one may ask if the mass launch of the buffered VMs in the mixing queue affect the VM startup time in the clouds ? We believe that there is not a significant affect, since the study in [11] shows that the VM startup time is unlikely affected by the business of open hours in which there are massive VM requests.

Based on the main result of the previous section (Theorem 4.4), we expose the percentage of the attacked VMs for a configuration with 5 target VMs and 100 attack VMs. As the number of VMs per physical servers depends of the type of the VMs (for example on EC2, there are no more 32 micro VMs, and no more 8 small VMs per physical server [19]), and varies from one cloud provider to another, we consider two values for server capacity, 10 and 20, that are closed to the commonly used capacities. In the next table, the variables P and M represent the server and the mixing queue capacities respectively.

Table 2: Percentage of the attacked VMs with 5 target VMs and 100 attack VMs

P \ M	1K	5K	10K	15K	20K
10	61.45%	16.64%	8.65%	5.84%	4.41%
20	86.78%	31.93%	17.40%	11.49%	9.09%

Following the above numerical results, we observe that if we consider an incoming VM requests rate of 3K per 10 minutes, and we tolerate a VM startup delay of around 198 seconds, we get a percentage of about 61%. Recall that, if we consider a server capacity smaller than 10, we obtain a smaller percentage of the attacked VMs. However, a low percentage of 4.41% requires a high startup delay of about 66 minutes. To sum up, even if we consider a low incoming VM requests rate, our strategy decreases the percentage of the attacked VMs up to the half with a low startup delay of few minutes.

In reality, multi-tenancy inhibits a full co-location resistance, and for this reason, the aim of the co-location resistant placement strategies is to reduce as much as possible the co-location rate. To better understand the effectiveness of the co-location resistance, and always based on Theorem 4.4, we quantify the effort needed by the adversary to achieve an attack.

Next, we estimate the number of the attack VMs needed to achieve co-location with a particular target set. This means that we consider the *complete* co-location attacks, where the adversary is successful if he co-locates with each target VM. We give more ability to the adversary by assuming that all the attack VMs can be inserted in the current mixing queue containing those of the target. In Table 3 we consider a target set of one VM, while in Table 4 we consider a target set of 10 VMs. From the numerical results of these tables, we remark that the increase of the number of the attack VMs, needed to achieve complete co-location, is linear with the increase of the mixing queue capacity. For example, if we consider the configuration with one target VM and a server capacity of 10 resource units (Table 3), each increase of the mixing queue capacity by 100 requires the launch of around 64 additional VMs to achieve a complete co-location.

5.2 Comparison with EC2 strategy

In [19], an analysis has been done when the target and attack VMs are launched simultaneously. One of the measurements states that achieving co-location with a particular target of type m1.small needs launching around 400 attack

Table 3: Attack VMs needed to co-locate with one target VM

P \ M	100	200	300	400	500	600	700	800	900	1K
10	64	131	197	264	331	398	464	531	598	665
20	37	78	118	159	200	240	281	321	362	403

Table 4: Attack VMs needed to co-locate with 10 target VMs

P \ M	100	200	300	400	500	600	700	800	900	1K
10	71	145	219	294	368	442	516	591	665	739
20	43	90	138	185	233	280	328	375	422	470

VMs (the average over 15 measurements), which will take 20 rounds with each round launching 20 attack VMs. To compare with our algorithm, we consider the best case (for the co-location resistance) of the 15 measurements, in which the adversary needs launching around 580 attack VMs. This will take 29 rounds, thus we get that the probability of the co-location with 20 attack VMs and a target VM is $\frac{1}{29}$. By setting the server capacity at 30 (note that in [13] no more than eight m1.small VMs were ever observed to be simultaneously co-resident), and the mixing queue capacity at 250, our algorithm gives us a probability less than the half of $\frac{1}{29}$. If we assume that the incoming VM requests rate is around 3K per 10 minutes, the VM startup delay is less than one minute.

5.3 Comparison with Y. Azar et al strategy

Roughly speaking, the strategy of Y. Azar *et al* assigns an incoming VM to a server belonging to a subset of the all servers. These open servers, are devoted to receive the new VMs, if one of the open servers can not host a new VMs, then is closed and a new server is opened. The algorithm keeps always λ servers open, with λ is a predefined parameter of the algorithm. Note that in their analysis, the authors do not consider the adversary strategy.

In order to compare the both strategies against APL strategy, we use the "probability of at least one co-location" as a comparison yardstick. Moreover, to consider the worst case for the co-location resistance, we assume that the N_t target VMs are placed on N_t physical servers in the both strategies. This case benefits the adversary to achieve the co-location attacks. In the Y. Azar strategy, the probability of at least one co-location is:

$$1 - \left(1 - \frac{N_t}{\lambda}\right)^{N_a}$$

In our strategy, we use the First Fit[1] algorithm to optimize the number of the used servers, and we proceed to calculate the probability as follows: we place the N_t target VMs on the first N_t servers, then we calculate the probability of co-locating the other VMs with those of the target, thus we can deduce the probability of at least one attacked VMs:

$$1 - \prod_{k=0}^{N_t-1} \left(\frac{\binom{N_o - k.(p-1)}{p-1}}{\binom{N_a+N_o}{p-1}} \right)$$

[1]We choose the First Fit as an example because it provides an interesting competitive ratio for the one-dimensional problem in the online setting [7].

Concerning the comparison of the resources optimization, we consider the one-dimensional problem. In our strategy the competitive ratio depends on the optimization strategy. The competitive ratio of an online algorithm A is the worst-case ratio between A and the offline performance of the optimal algorithm [14]. So, the smaller the competitive ratio, the better performance of the online algorithms A. For the First Fit strategy, it is known that the competitive ratio for the one-dimensional bin packing is equals to 1.7 [7]. So, if we consider the cost function introduced in [4], defining the cost as the sum over time of the number of servers that are on at each time step, we can show (by proceeding as the proof Theorem 5.1 in [4]) that the competitive ratio is equals to 1.7 + 2 (for one-dimensional problem). This competitive ratio is very smaller than $\lambda + 2$ that is the competitive ratio of Y. Azar *et al* strategy.

The next table exposes the performance overhead imposed by the both strategies to achieve the same probability of at least one co-location with 5 target VMs. The performance is described by the competitive ratio and the startup delay incurred only by the placement strategy (Competitive ratio:Startup delay). The startup delay depends on the VM requests rate, thus given a mixing queue capacity M and a VM requests rate per one minute r, the startup delay in minutes is given by the quotient $\frac{M}{r}$.

Table 5: Probability of at least one co-location with 5 target VMs and 100 attack VMs

Strategy \ Co-location	40%	4.9%	0.5%
Azar et al strategy	102 : 0	1002 : 0	10002 : 0
Our strategy	3.7 : $\frac{910}{r}$	3.7 : $\frac{9K}{r}$	3.7 : $\frac{90K}{r}$

With r denotes the VM requests rate per one minute.

Based on Table 5, we derive that our strategy is significantly more effective in terms of resources optimization, but is less effective in terms of startup delay. We believe that for a high VM requests rate our strategy is more interesting when considering a certain low co-location probability. For example, with a VM requests rate of 3K/minute, we can achieve a co-location probability of 4.9% with a low startup delay of 3 minutes, and a competitive ratio of 3.7 that is very smaller than the competitive ratio provided by Azar *et al* strategy. A startup delay of 3 minutes can be tolerated by the users running no real-time services. In summary, a reliable comparison must take into account the security,

the resources optimization and the startup time. Following their preferences, the users can choose between security and startup time, but for the cloud providers the resources optimization is crucial.

6. CONCLUSION AND FURTHER WORK

In this work we proposed a VM placement strategy mitigating the co-location attacks with full resources optimization. Our strategy introduces a tradeoff between security and VM startup delay. The proposed strategy is evaluated against the parallel placement locality, in which the target and attack VMs are launched simultaneously or within a short time. For a low rate of the incoming VM requests, our evaluation shows that the proposed strategy allows to decrease the likelihood of co-location with startup delay of few minutes. Moreover, the comparison with EC2 strategy, affirms that our strategy allows a better co-location resistance with a VM startup less than one minute.

An interesting observation was pointed out in [13, 19], stating that the time lag between the adversary and the target launches does not have significant effect on the co-location. This means that for a long lag, the attacks exploit the sequential placement locality. As mentioned in [15], the sequential locality is influenced by the churn rate where some co-located other VMs with the target were terminated. Therefore, a strong analysis of the churn is needed to evaluate our strategy against sequential locality, and consequently against brute-forcing placement strategy. Moreover, the above observation leads to the following questions: for a short lag time, how we can distinguish if an attack takes advantage of the parallel or sequential locality ? From which lag time, an attack takes advantage of the sequential locality ? The answers to these questions, will help us to better understand how proceed in the analysis.

In another direction, it would be interesting to combine our strategy with that proposed by Y. Azar *et al* [4], in such case we will get more flexibility because the tradeoff will be between security, resources optimization and VM startup time.

References

[1] Amazon usage estimates. http://www.rightscale.com/blog/cloud-industry-insights/amazon-usage-estimates. Accessed: 2016-07-23.

[2] Rightscale 2015 state of the cloud report. http://www.rightscale.com/blog/cloud-industry-insights/cloud-computing-trends-2015-state-cloud-survey. Accessed: 2016-07-23.

[3] Z. Afoulki and J. Rouzaud-Cornabas. A security aware scheduler for virtual machines on Iaas clouds. Technical report, University of Orleans, 2011.

[4] 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*, CCSW '14, pages 9–20. ACM, 2014.

[5] A. Beloglazov and R. Buyya. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. *Concurr. Comput. : Pract. Exper.*, 24(13):1397–1420, 2012.

[6] N. Bobroff, A. Kochut, and K. Beaty. Dynamic placement of virtual machines for managing sla violations. In *Proceedings of 10th IFIP/IEEE International Symposium on Integrated Network Management*.

[7] G. Dósa and J. Sgall. First Fit bin packing: A tight analysis. In *30th International Symposium on Theoretical Aspects of Computer Science*, STACS 2013, pages 538–549. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2013.

[8] Digital Pwer Group. The cloud begins with coal Ű big data, big networks, big infrastructure, and big power. 2013.

[9] Y. Han, J. chan, T. Alpcan, and C. Leckie. Virtual machine allocation policies against co-resident attacks in cloud computing. In *Proceedings of the 2014 IEEE International Conference on Communications*, ICC'14, pages 786 – 792. IEEE, 2014.

[10] Z.Á. Mann. Allocation of virtual machines in cloud data centers—a survey of problem models and optimization algorithms. *ACM Comput. Surv.*, 48(1):11:1–11:34, 2015.

[11] M. Mao and M. Humphrey. A performance study on the vm startup time in the cloud. In *Proceedings of the 2012 IEEE Fifth International Conference on Cloud Computing*, CLOUD '12, pages 423–430. IEEE Computer Society, 2012.

[12] F.L. Pires and B. Barán. Virtual machine placement literature review. *CoRR*, 2015.

[13] 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*, CCS '09, pages 199–212. ACM, 2009.

[14] Daniel D. Sleator and Robert E. Tarjan. Amortized efficiency of list update and paging rules. *Commun. ACM*, 28(2):202–208, 1985.

[15] V. Varadarajan, Y. Zhang, T. Ristenpart, and M. Swift. A placement vulnerability study in multi-tenant public clouds. In *Proceedings of the 24th USENIX Conference on Security Symposium*, SEC'15, pages 913–928. USENIX Association, 2015.

[16] A. Verma, P. Ahuja, and A. Neogi. pmapper: Power and migration cost aware application placement in virtualized systems. In *Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware*, Middleware '08, pages 243–264. Springer-Verlag New York, Inc., 2008.

[17] A. Verma, G. Dasgupta, T.K. Nayak, P. De, and R. Kothari. Server workload analysis for power minimization using consolidation. In *Proceedings of the 2009 Conference on USENIX Annual Technical Conference*, USENIX'09, pages 28–28. USENIX Association, 2009.

[18] Z. Wu, Z. Xu, and H. Wang. Whispers in the hyper-space: High-speed covert channel attacks in the cloud. In *Proceedings of the 21st USENIX Conference on Security Symposium*, Security'12, pages 9–9. USENIX Association, 2012.

[19] Z. Xu, H. Wang, and Z. Wu. A measurement study on co-residence threat inside the cloud. In *Proceedings of the 24th USENIX Conference on Security Symposium*, SEC'15, pages 929–944. USENIX Association, 2015.

[20] Z. Xu, H. Wang, Z. Xu, and X. Wang. Power attack: An increasing threat to data centers. In *Proceedings of 21st Annual Network and Distributed System Security Symposium*, NDSS'14, 2014.

[21] Y. Zhang, A. Jules, 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*, CCS '14, pages 990–1003. ACM, 2014.