How to hide your VM from the big bad wolf? Co-location resistance vs. resource utilisation in VM placement strategies

Jens Lindemann
Universität Hamburg
Hamburg, Germany

ABSTRACT

VMs in cloud environments are at threat of attacks from VMs co-located on the same server, e. g. through side-channels. Reducing the ability of attackers to achieve co-location with specific VMs can alleviate the risk of targeted attacks. This paper presents the simulation framework VMPlaceSim, which allows to evaluate resource utilisation and resistance against co-location attacks of VM placement strategies. A new strategy based on the proportion of known users on servers is proposed and evaluated on real-world cloud workload data alongside existing strategies. The evaluation takes attacks into account in which malicious VMs are either launched in regular intervals or their launch is timed around the launch of target VMs. The results indicate that the new known-users strategy is significantly more resistant to co-location attacks than existing strategies aimed at optimising resource utilisation, while retaining a relatively high resource utilisation exceeding that of strategies aimed at thwarting co-location attacks.

CCS CONCEPTS

• **Computer systems organization** → **Cloud computing**; • **Security and privacy** → **Distributed systems security**.

KEYWORDS

cloud computing, VM placement, co-location resistance, security

ACM Reference Format:
Jens Lindemann. 2023. How to hide your VM from the big bad wolf? Co-location resistance vs. resource utilisation in VM placement strategies. In *The 18th International Conference on Availability, Reliability and Security (ARES 2023), August 29-September 1, 2023, Benevento, Italy.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3600160.3604983

1 INTRODUCTION

Public clouds allow customers flexible access to computing resources without having to invest in their own hardware. This is realised by cloud service providers renting out such resources to different users, e. g. virtual machines (VMs) in the case of infrastructure-as-a-service (IaaS) clouds.

When assigning VMs to physical servers, there are two competing interests: On the one hand, providers want to maximise the utilisation of their servers. This will save costs, as fewer servers need to be procured and operated. On the other hand, customers want their VMs to not be placed on the same server as those of malicious other users. This is because the sharing of resources by VMs on a server opens up possibilities for attacks, e. g. through security vulnerabilities in the hypervisor or side-channel attacks.

It is impossible to fulfil both goals perfectly at the same time, i. e. they form a trade-off. To achieve perfect *co-location resistance* as defined by Azar et al. [4], a provider would need to place only VMs of a single user on a server. This is incompatible with maximising resource utilisation: Consider a user with just one small VM. Unless the server is also very small, the provider would be left with a large amount of unused resources on the server hosting this VM. Therefore, different strategies aimed at retaining a relatively high resource utilisation while still achieving co-location resistance have been proposed. These strategies still leave room for optimisation, with some strategies falling short in terms of resource utilisation and thus being too costly to implement and others failing to provide a meaningful protection from co-location attacks.

The threat model of this paper assumes the attacker to be a customer of a public cloud service provider which is also used by other, benign users. The attacker's goal is to obtain co-location with VMs of other users. The attacker can launch VMs and is free to choose both the resource requirements of a VM and its start time. However, placement decisions are made by the provider, i. e. the attacker cannot choose which server a VM is placed on.

The provider, on the other hand, aims to protect its benign users from malicious users. More specifically, the provider's goal is to minimise the number of benign users whose VMs are placed on servers also hosting VMs of malicious users at the same time.

This paper makes the following key contributions:

(1) The open-source simulation framework VMPlaceSim is presented, which can be used to evaluate co-location resistance and resource utilisation achieved by VM placement strategies.
(2) The known-users (KU) placement strategy is proposed.
(3) The co-location resistance and resource utilisation of KU as well as existing placement strategies are evaluated experimentally. In contrast to previous work analysing such strategies, the evaluation covers attack scenarios where an attacker times VM launches around the launch of a target VM or spreads out VM launches equally over time.

The remainder of this paper is structured as follows: Section 2 discusses related work. Then, existing placement strategies as well as the known-users strategy are presented in Section 3. The simulation framework is presented in Section 4. Section 5 describes the evaluation, before Section 6 concludes the paper and gives an outlook on future work.

ARES 2023, August 29–September 1, 2023, Benevento, Italy
© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0772-8/23/08…$15.00
https://doi.org/10.1145/3600160.3604983

2 RELATED WORK

Obtaining co-location with a target VM may allow an attacker to launch attacks that would not otherwise be possible. Many such attacks are related to side-channels enabled by shared hardware. For example, through the Spectre vulnerability [20], data can be extracted from the host OS as well as co-located VMs. Cache-based side-channel attacks allow to extract cryptographic keys across VM boundaries [34]. Monitoring cache behaviour can also reveal the level of activity in co-located VMs [26]. A timing side-channel caused by memory deduplication can reveal the randomised address space layout [5] or which applications are running in another VM [22]. Foreshadow-VMM [28] allows reading data from the L1 cache from within a VM. Rowhammer attacks can even induce bit flips in read-only memory [19], including that of co-located VMs [30].

A further danger is posed by vulnerabilities in hypervisors, which could not only allow compromising the hypervisor or host OS, but also co-located VMs from there [25]. Finally, resource contention can affect performance across VMs [21], which can be exploited by inducing load, e. g. on memory [32, 33] or storage [8].

Achieving co-location with a target VM is realistic even in large public cloud environments [3, 27, 31]. While this effect has diminished significantly [31], Amazon EC2 was previously susceptible to an attacker timing their VM launches to coincide with that of a target VM [26].

To detect whether co-location with a target was achieved, an attacker could trigger load on the target VM and then measure the load on the server based on cache [26] or memory latency [27]. Cache-based side-channel attacks may also be used to detect co-location, e. g. by detecting the reception of pings [14].

Han et al. [16] describe a game-theoretic framework in which an attacker wishes to launch few VMs to achieve co-location with many target VMs, while the defender wants to place VMs so that the attacker's success is minimised while balancing the workload between servers and achieving a low power consumption.

Cortez et al. [11] study the workload of the Azure cloud and try to improve resource utilisation by predicting CPU usage of VMs and oversubscribing the host CPU resources. They also made two datasets of the Azure VM workload publicly available [9, 10].

CloudSim [7] is a simulation framework for cloud environments. It is relatively complex and supports simulating host and VM provisioning, application workloads as well as network traffic. It is not, however, geared towards evaluating co-location resistance. It also does not support directly loading the cloud workload dataset provided by Cortez et al. [11].

An alternate approach to the type of strategy analysed in this paper is to limit the amount of time they are co-located. This can be achieved by regularly migrating VMs between servers [24, 34]. While this reduces the time spent co-located with a malicious VM, this will increase the chance of encountering one.

3 PLACEMENT STRATEGIES

This section presents existing placement strategies (Sect. 3.1) as well as the proposed known-users (KU) strategy (Sect. 3.2). Implementations of the evaluated strategies are included with the simulation framework (cf. Sect. 4).

A placement strategy has the primary objective of assigning a new VM v_n of a user u_r to a server. There can be different side objectives, such as maximising co-location resistance or minimising the number of active servers, i. e. servers that are turned on and hosting VMs, in contrast to inactive servers, which can be turned off. This would help to reduce energy consumption and costs.

3.1 Existing strategies

Many different placement strategies already exist, some of which are evaluated in this paper. Of these, the PCUF, PSSF, LDBR, Azar and dedicated-instance strategies were designed with co-location resistance in mind. These provide what Masdari et al. [23] consider to be static VM placement, i. e. VMs will remain on a server throughout their lifetime. The remaining strategies are provided for comparison and are aimed at optimising either the resource utilisation or the balance of the workload between servers.

Agarwal and Duong [1] propose the **previously co-located users first (PCUF)** strategy. When an existing user u_r requests a new VM v_n, the strategy considers all active servers that fulfil two criteria: First, the server needs to have sufficient free capacity. Second, at least one of the VMs on the server must belong to a user who has previously had a VM co-located with one of u_r's VMs. Out of these servers, the one with the fewest free CPU cores is selected. If no server fulfils the criteria, an inactive server is activated instead.

If v_n is u_r's first VM, it is assigned to a random active server with sufficient capacity, if possible, or else to an inactive server.

Note that a disadvantage of co-locating VMs of users whose VMs have previously been placed together is that this will allow an attacker to reproducibly obtain co-location with such users. To avoid this, more variation would have to be introduced into VM placements, thereby exposing more different users to co-located VMs of malicious users. This also applies to other strategies relying on a similar principle (e. g. PSSF and KU).

The strategy of **Azar** et al. [4] keeps three lists for inactive ("empty"), active ("open") and full ("closed") cloud servers. Only a number of servers is initially made active. A new VM is assigned to a random active server. If less than half of one of the server's resources remains available after assigning a new VM, the server is considered full. An additional server is then made active.

Azar et al. do not discuss how their strategy handles an increase in the available capacity of a server after a VM shutdown. Agarwal and Duong [1, p. 217] therefore extend the strategy for their experiments: Whenever a VM is shut down, they check the available resources of a server and, if necessary, move the server to the list of active or, if it is now empty, inactive servers.

A problem with this is that the number of active servers may diverge from the configuration. Thus, the implementation for this paper incorporates an additional change: If the number of active servers would be too high after a VM has been shut down, empty active servers are shut down until the number of active servers matches the configuration, if possible. Also, newly emptied servers are only deactivated if enough servers would remain active.

The strategy makes the assumption that no VM may use more than half of a server's total capacity with respect to a resource. If there is a VM that breaks this assumption, the strategy may fail due to choosing a server with insufficient capacity. Therefore, a further modification was made: If one of the resource constraints of a VM exceeds half of the total capacity of a server, the active servers are filtered according to their free capacity. Inactive servers are then added to the list of candidates, so that the same number of candidate servers is always available for selection.

Han et al. [17] propose the **previously-selected-servers-first (PSSF)** placement strategy. The strategy divides the servers into equally sized groups. At first, only one group is active. All other servers remain inactive. The strategy first tries to fill up a group of servers before it activates another group.

If there is at least one server within the active group(s) with sufficient capacity for the new VM v_n of u_r, a random server which currently hosts or has previously hosted a VM of u_r is chosen. If no server fulfils these criteria, one of the servers currently hosting the lowest number of VMs within the first active group containing at least one server with sufficient free capacity is chosen. If no such server exists, the first inactive group is activated.

Unlike other strategies, PSSF operates based on active and inactive *groups* instead of servers. This leaves room for interpretation as to whether empty servers within an active group should be active. In this paper, such servers are considered to be inactive.

Xiao et al. [29] propose the **least danger based on reputation (LDBR)** strategy. It places VMs based on user reputations, which are to be determined using a detection mechanism for co-location attacks. They do not, however, propose such a detection mechanism.

LDBR aims to calculate the expected change in the number of VMs exposed to malicious users. Where n is the number of VMs on the server, p_i the probability that the owner of the ith VM on that server is malicious and p' the probability that the owner of the new VM is malicious, the expected change in the number of VMs exposed to malicious users is calculated as follows: $E(\Delta T) = n \prod_{i=1}^{n} p_i (1 - p') + (1 - \prod_{i=1}^{n} p_i) p'$. The new VM is allocated to the server with the minimum $E(\Delta T)$. For this paper, LDBR was modified slightly so that it prefers assigning a VM to an already active server. This encourages energy and cost savings.

The **best-fit** heuristic for online bin packing problems can also be used as a placement strategy. To implement this, a new VM v_n is assigned to one of the active servers with the fewest free cores, but enough capacity for v_n. If no active server has sufficient capacity, an inactive server is activated. The strategy is aimed at maximising the core utilisation, but does not directly optimise the use of other resources. VM ownership is not considered in placement decisions.

If the **worst-fit** heuristic is used as a placement strategy, it assigns a new VM v_n to one of the active servers with the most free cores. If no active server has sufficient free resources, an inactive server is activated. This strategy spreads out the core utilisation evenly between the active servers. Like best-fit, it makes VM placements without regarding VM ownership.

To apply the **first-fit** bin packing heuristic to VM placements, a fixed order of servers is defined. A new VM v_n is assigned to the first server in the fixed order that has sufficient free capacity. This strategy is geared towards optimising resource utilisation. VM ownership is not considered in placement decisions.

For the **next-fit** heuristic, a fixed order of servers is again defined. A new VM v_n is assigned to the server succeeding the one used to host the last created VM. If this server does not have sufficient capacity to host the new VM, it is skipped. At the end of the server list, the strategy reverts back to the first server. Again, VM ownership is not considered in placement decisions.

The **random** strategy assigns a newly requested VM to a random server. The implementation for this paper tries to first use an active server with sufficient capacity, before activating additional servers. Alternatively, all servers including inactive ones could be considered as candidates, spreading out the VMs even more. VM ownership is not considered in placement decisions.

The **dedicated-instance** strategy places only VMs of a single user on a server at a time. Amazon offers its customers the option to rent VMs placed in this manner [2]. This ensures that users never encounter a malicious VM. However, this also means that a server may host only one small VM if it is the user's only VM.

Hay et al. [18] propose to apply the **Chinese Wall** security policy [6] to the placement of VMs. They propose to assign customers to conflict of interest classes. Only VMs from at most one customer from each class may be placed onto a server. Other VM placement strategies that would allow for defining very specific constraints on the placement of VMs – and thus prevent specific pairs or types of VMs from being co-located – were proposed by Espling et al. [12] as well as Gaggero and Caviglione [13]. As the dataset used for evaluation does not contain information about conflicts of interest between users, these policies are not evaluated in this paper.

There are also much more complex strategies than those surveyed here. One example of these is **Protean** [15], which is used by Microsoft Azure. It covers multiple cloud zones and makes placement decisions on multiple layers.

3.2 Known-users (KU) strategy

The *known-users (KU)* strategy considers the proportion of users known on a server. Algorithm 1 shows s pseudo-code representation of the strategy. When a user u_r requests a new VM v_n, the strategy first considers all active servers with sufficient free resources. It then checks how many of the users whose VMs are currently hosted on the server have previously had VMs co-located with a VM of u_r and calculates the proportion of users for which this is the case.

The VM is then placed on the server on which u_r knows the highest proportion of users. If the maximum proportion is shared by multiple servers, one of these is chosen randomly.

Note that if there is a server that is already hosting a VM of u_r *and* that has sufficient capacity, this server will be chosen as u_r knows all other users on this server. Therefore, the strategy has a tendency of selecting servers previously selected to host VMs of a user. However, unlike PSSF, it considers the *users* currently on the server instead of the server itself. As attacks originate from users and not the server itself, this better estimates how dangerous a server is: On the one hand, a server may have hosted a VM of u_r previously, but the other VMs on the server may since have changed completely. On the other hand, a server may have never hosted a VM of u_r before, but the other VMs on the server may belong to users whom u_r has encountered before.

To implement this strategy, information about whether two users know each other, i. e. whether their VMs were previously co-located, needs to be stored. This can be achieved by storing one boolean value for each pair of users (i. e. $(n^2 - n)/2$ values for n users).

Algorithm 1: Known-user (KU) server selection strategy

```
Input: $S_a, S_i$ – sets of active and inactive servers
       $m_n, c_n, u_n$ – memory, CPU cores and owner of new VM
1  $S_c \leftarrow \emptyset$ ;   // Initialise set of candidate servers
2  $p_{max} \leftarrow 0$ ;       // Max. proportion of users known
3  forall $s \in S_a$ do
4      if $freememory(s) \geq m_n \wedge freecores(s) \geq c_n$ then
5          $T, K \leftarrow 0$ ;  // Sets of total and known users
6          forall $v \in s$ do // Iterate over VMs on server
7              $u_v \leftarrow v$.owner
8              $T \leftarrow T \cup \{u_v\}$
9              if $u_n.knows(u_v)$ then
10                 $K \leftarrow K \cup \{u_v\}$
11         $p_n \leftarrow |K| / |T|$
12         if $p_n > p_{max}$ then
13             $S_c \leftarrow \{s\}$
14             $p_{max} \leftarrow p_n$
15         else if $p_n = p_{max}$ then
16             $S_c \leftarrow S_c \cap \{s\}$
17 if $S_c = \emptyset$ then
18     $S_c = S_i$
Output: random $s \in S_c$
```

Alternatively, for each user, a list of known users can be stored, which avoids the need to store a value for each pair of users.

When making a placement decision, the strategy needs to look up for each user on each server that has sufficient resources available whether that user has previously been co-located with a VM of the new VM's owner. The complexity thus rises with the number of (1) servers with sufficient free resources available to host a VM and (2) different users present on the servers. The number of servers with sufficient free resources is likely to rise with the size of a cloud environment. The number of users that could be present on a server, on the other hand, depends on the relative size of VMs and servers, but should not vary depending on the cloud size. Therefore, a larger cloud environment makes placement decisions more complex, but only in proportion to the computational resources used by the VMs themselves.

4 SIMULATION FRAMEWORK

This paper is accompanied by the open-source framework VM-PlaceSim for simulating VM placement strategies and evaluating their performance, which is available on Github[1].

The framework includes implementations for all placement strategies evaluated in the paper, as described in Sect. 3. Each strategy is encapsulated in a separate Java class extending the interface PlacementStrategy. The framework is extensible: Additional strategies can be added by extending this interface.

The framework simulates VM allocations within a cloud based on the chosen placement strategy. It reports the results in the form of CSV files, which contain various statistics about resource utilisation, co-location resistance as well as the number of server starts and shutdowns. A detailed description of the metrics supported by the framework can be found in Section 4.1.

The framework processes information about the VMs to create throughout the simulation provided in form of a CSV file. The file format matches that of the publicly available Microsoft Azure datasets [9, 10]. For each VM, it specifies boot and shut-down time, CPU core count and memory footprint. Furthermore, the ownership of the VM is defined by a user ID. The CSV files of the Microsoft datasets also contain a VM ID, deployment ID, a VM category as well as information about CPU utilisation. However, these are not used in any of the placement strategies evaluated in this paper.

The framework allows many parameters of the simulation to be configured: First, the number of servers in the simulated cloud environment and their CPU core and memory capacity. Second, the number of servers initially active. Third, the start and end time, which allows the simulation to be restricted to a part of the period covered by the input data. Fourth, the statistics can be reported in desired intervals throughout the simulation. Fifth, the seed for the PRNG initialisation can be set to ensure comparability of results across strategies as well as reproducibility. Sixth, parameters specific to strategies can be set, e. g. the group size for PSSF.

Finally, the proportion(s) of malicious users can be set for experiments where these are to be chosen randomly from the input data. The framework can test multiple proportions in a single run for strategies whose placements are not based on whether a user is malicious. This is the case for all but the LDBR strategy, where the differences in the calculated user reputations will lead to different placements for different proportions of malicious users.

During a simulation, two events are processed for each VM: start and shutdown. Events are processed in the order of the time specified. If VMs are started and shut down at the same time, VM starts are processed first. In such cases, the time resolution of the input is insufficient to determine the real order of these events. Processing starts first prevents underestimating the resource requirements.

Both start and shutdown events are considered to be instantaneous, i. e. they have an immediate effect on resource consumption. This means that if a VM needs time to shut down before it can be decommissioned, the shutdown time in the input data should be set to the time of *completion* of the shutdown.

A similar assumption is made for the activation and deactivation of servers, which is also considered to be instantaneous. In reality, booting or shutting down a machine will take some time, during which the machine cannot host any VMs, but is still on and consumes energy. This implies that resource utilisation may be overestimated slightly.

4.1 Supported metrics

In the following, the metrics supported by the simulation framework are described. The framework will report the results of an experiment by writing the calculated metrics to CSV files. Where appropriate, these can also be calculated periodically throughout the experiment. This allows an insight into changes over time, such as the performance of a strategy improving or deteriorating.

One important aspect of the performance of a strategy is to what extent it exposes VMs of benign users to those of malicious users. To

[1]https://github.com/jl3/VMPlaceSim

this end, Agarwal and Duong [1, p. 214–215] propose to determine the proportion of users who were ever exposed to a malicious users. This means that a user is considered to be *safe* only if none of their VMs were ever co-located with any VM of any malicious user. In the following, this will be referred to as *user-based co-location resistance (UCLR)* – Agarwal and Duong [1] simply refer to this as co-location resistance (CLR). It is calculated as follows:

$$\text{UCLR} = \frac{\text{number of safe benign users}}{\text{total number of benign users}}$$

As an alternative to this, the simulation framework calculates the *VM-based co-location resistance (VMCLR)*. It considers whether individual VMs of benign users are safe. A VM is safe if it was never co-located with any VM owned by any malicious user. The VMCLR may better represent the security of a VM from co-location attacks where a user runs many VMs configured in a way that a successful attack on one VM does not easily allow the compromise of another (e. g. VMs are processing different data or are not sharing a cryptographic key at danger of a side-channel attack). This is in contrast to the UCLR, which assumes a user to be unsafe if just one of their VMs has been co-located with a malicious user. This may be more appropriate for a load-balancing scenario, where a user runs many identical VMs, so that an attack on any of these VMs would completely compromise the user's security. The VMCLR is calculated as follows:

$$\text{VMCLR} = \frac{\text{number of safe VMs owned by benign users}}{\text{total number of VMs owned by benign users}}$$

The CLR metrics can be determined for the full experiment or for sub-periods. When evaluating the CLR for sub-periods, the framework additionally supports calculating the NewVMCLR, i. e. the CLR only for VMs launched within a period. If the CLR of a placement strategy deteriorates over time, i. e. newly created VMs are exposed to more risk, the NewVMCLR will clearly show this. The NewVMCLR for a period P is calculated as follows:

$$\text{NewVMCLR} = \frac{\text{no. of safe VMs of benign users launched in } P}{\text{total no. of VMs of benign users launched in } P}$$

As mentioned in related work, providers may also try to limit the time that users or VMs are unsafe, e. g. by migrating them between servers. This may be appropriate if VMs are considered to only holds sensitive information for short periods of time or if attacks are assumed to take a long time. In this case, it may be useful to investigate the *time* that VMs of benign users are co-located with those of malicious users. To this end, the framework provides the *safe VM-time proportion (SVTP)*, which is calculated as follows for the set of all VMs V:

$$\text{SVTP} = \frac{\sum_{v \in V} \text{safeTime}(v)}{\sum_{v \in V}(\text{shutdownTime}(v) - \text{startTime}(v))}$$

The safe-time proportion can also be calculated on a per-user base. In this case, a user is deemed safe while none of their VMs is co-located with a malicious user. A user is deemed active when at least one of their VMs is running. Where U is the set of all users, the *safe user-time proportion (SUTP)* is calculated as follows:

$$\text{SUTP} = \frac{\sum_{u \in U} \text{safeTime}(u)}{\sum_{u \in U} \text{timeActive}(u)}$$

The framework also supports the metric of *coverage*, as proposed by Xiao et al. [29, p. 5–6]. They define this as the "rate of servers in danger", i. e. the proportion of servers that host a VM of a malicious user, thereby putting any VMs of benign users placed on this server at danger. It is calculated as follows:

$$\text{Coverage} = \frac{\text{number of servers with VMs of malicious users}}{\text{number of servers}}$$

Another important perspective is how efficiently the resources of servers are used by a placement strategy. To this end, the *core utilisation (CU)* can be calculated, which tells how many of the CPU cores of active servers are being used. A higher CU means that fewer servers need to be active. Where V is the set of all VMs and S the set of all servers, it is calculated as follows:

$$\text{CU} = \frac{\sum_{v \in V}(\text{cores}(v) * (\text{startTime}(v) - \text{shutdownTime}(v)))}{\sum_{s \in S}(\text{cores}(s) * \text{timeActive}(s))}$$

Statistics related to the activity of servers can also be generated: On the one hand, the number of server boots and shutdowns are provided. Note that while these numbers will be equal for a full simulation, they may differ for a sub-period. On the other hand, the average number of servers active is calculated both at the end of the simulation as well as for sub-periods.

The average number of active servers is directly related to the CU: Given the same VM requests, a higher CU leads to a lower average number of active servers. Therefore, both metrics give an impression of the energy efficiency of a placement strategy.

Statistics about VM activity are reported: VM starts, VM shutdowns, the maximum number of VMs and the average number of VMs active. Note that these statistics relate to the dataset instead of the placement strategy, which does not have an influence on whether and when VMs are started or shut down.

Finally, statistics relating to the memory use of strategies that need to store additional information to consider in their placement decisions are reported: For KU and PCUF, the number of co-location relationships between user pairs is reported. For PSSF, the number of host-user relationship pairs is reported.

5 EVALUATION

The existing placement strategies as well as the newly proposed KU strategy were evaluated with regard to resource utilisation and co-location resistance. While the real-world Azure dataset was used to represent the VM deployments, no real-world data about VM deployments by attackers is available. Therefore, some users from the dataset were randomly declared to be malicious for a first experiment (Sect. 5.1). Additionally, isochronous and targeted attacks were evaluated. For this, artificially generated attacker deployments were generated that represent different possible behaviours of an attacker trying over a longer time to get their VMs co-located with other VMs. The Azure dataset workload was used to represent benign users. In an isochronous attack, attackers launch their VMs in regular intervals (Sect. 5.2). In a targeted attack, they time the launch of their VMs around the launch of a target VM (Sect. 5.3).

For the evaluation, a dataset published by Microsoft [10] was used, which contains information about VM deployments in the Azure cloud from 2019. It covers a period of 30 days and comprises 2 695 548 VMs of 6 687 distinct users.

This dataset is similar to the earlier 2017 dataset [9], days 11-20 of which were used by Agarwal and Duong [1] for their evaluation. For the sake of brevity, this paper will present results only for the newer 2019 dataset. However, the results of preliminary experiments indicate that while there is some difference in the absolute performance of placement strategies between the 2017 and 2019 datasets, a strategy that performs better on one of the datasets generally also performs better on the other dataset.

Like in Agarwal and Duong's experiments, servers were configured to have 32 cores and 224 GiB of memory each. The number of servers was set to the maximum number of VMs running simultaneously according to the dataset.

For the experiments, the start time was set to 1. The dataset contains a large number of VMs started at time 0, more than started at any other time. As some of these VMs may have been started before 0 with the dataset not specifying the order of their starts, all VMs with a specified start time of 0 are ignored.

For some placement strategies, specific parameters had to be set or default parameters had to be modified: To prevent next-fit from spreading out VMs over too many servers, the number of servers was lowered to 1.5 times the maximum number of concurrent servers needed by best-fit, averaged over all runs.

For LDBR, the p-values normally returned by the detection system for malicious users need to be simulated. As Xiao et al. [29] do not mention how they calculated these in their experiments, a PERT distribution with a mode of 0.9 and lambda of 3.0 (resulting in a mean of 0.74) was used to generate the p-values of malicious users. For benign users, the mode was set to 0.1.

For the Azar strategy, the minimum number of servers kept open for new VMs can be freely configured. For the evaluation, two configurations were tested: 5 % and 20 % of all servers. Similarly, two configurations of the PSSF strategy were tested: The number of servers per group was set to 5 % and 20 % of all servers.

5.1 Malicious users from dataset

For the first experiment, a proportion of users was randomly chosen to be malicious, similar to the experiments conducted by Agarwal and Duong [1]. This means that the VM launches and shutdowns represent real user behaviour, but it is unknown whether they belong to a real-world malicious user. For each configuration, 50 simulation runs were performed. The same 50 seeds were used to initialise the PRNG for different placement strategies, so that the same sets of users were chosen to be malicious for all strategies.

Table 1 shows the core utilisation (CU), the number of server boots (which is equal to the number of shutdowns) as well as the average and maximum number of servers used by different placement strategies. Furthermore, it shows the user- and VM-based co-location resistance (UCLR and VMCLR, cf. Sect. 4.1). The CLR values are based on 5 % of users being malicious. With the exception of LDBR, the resource efficiency metrics, however, are independent of the number of malicious users.

First-fit achieves the highest CU, slightly ahead of 'best'-fit. They are followed by KU, LDBR, PSSF-5 % and the random strategy. All other strategies achieve a CU of less than 0.8. Azar-5 %, Azar-20 %, PSSF-20 % and next-fit trail even dedicated-instance, i. e. they are inferior to just outright avoiding co-locating VMs of different users.

The average number of servers active is directly related to the CU: If the CU is high, fewer servers need to be active. The best-performing first-fit strategy needs 3 726 servers on average. Next-fit, on the other hand, needs more than 8 times as many (30 608).

The maximum number of servers needed during the experiments provides an insight into how much hardware the provider needs to have available. The results indicate that the ranking of the strategies is relatively similar to those for CU and the average number of active servers. Again, there is a wide range between the best- and worst-performing strategies: Best-fit needs 5 678 servers, while next-fit needs about 7.7 times as many (43 489).

LDBR performs the fewest server boots and shutdowns, closely followed by KU. Best-fit comes third. All other strategies perform significantly more boots and shutdowns, with Azar-5 % performing almost 90 times as many as LDBR.

Due to not placing VMs of different users on the same server, the dedicated-instance strategy always achieves a user-based CLR (UCLR) of 1. PSSF-20 % achieves a comparatively good UCLR. However, remember that its CU is inferior to dedicated-instance. PSSF-5 %, on the other hand, achieves a UCLR of just 0.1953. PCUF comes third. Azar performs poorly, with even Azar-20 % achieving a UCLR of just 0.1435. KU achieves a mid-field performance (0.4534). While there are other strategies with higher UCLRs, any further improvement can only be realised by accepting a significantly lower CU, resulting in a higher number of servers needed (e. g. +16.1 % for PCUF and +32.6 % for dedicated-instance on average) and thus higher costs. Best-fit and first-fit, which perform best in terms of CU, perform worst in terms of UCLR (<0.07).

For the VM-based CLR (VMCLR), dedicated-instance again achieves perfect results. The results indicate that the VMCLR is higher than the UCLR for all other strategies. PSSF-20 % comes into second place. Unlike for the UCLR, KU only comes ninth with just under a quarter of VMs being exposed to malicious users. Again, best-fit and first-fit perform worst.

Fig. 1 shows how the CU – and thus power and cost efficiency – changes over time for the different strategies. The value shown is the average CU up to a point in time.

After a short warm-up phase, the CU remains relatively stable for almost all strategies. For PSSF-5 %, however, the CU changes drastically: It increases from ~0.52 to ~0.83. Further drastic changes can be observed for the two configurations of the Azar strategy. While these improve markedly throughout the run of the experiment, their performance remains comparably poor. KU performs relatively well (CU ≈ 0.87).

Fig. 2 shows how the CLR for new VMs (NewVMCLR) develops over time given 5 % of malicious users. The value for each six-hour period corresponds to the VMCLR of VMs started during the period.

For most strategies, the NewVMCLR remains relatively stable over time. However, PSSF-5 % shows a drastic deterioration over time. It starts out with a performance very close to the optimum, but falls back to about 0.8. This indicates that this strategy will be more and more susceptible to co-location attacks in continuously operating cloud environments. While PSSF-20 % also shows a deterioration, it is less pronounced. The NewVMCLR of KU is in the mid-field, but remains relatively stable over time. It beats best-fit and first-fit, the only two strategies that achieve a higher CU.

Table 1: Performance statistics of placement strategies for the full 2019 Azure dataset (strategy ranks in parentheses)

		Servers			5 % malicious	
Strategy	CU	Avg.	Max.	Boots	UCLR	VMCLR
Azar-5%	0.4828 (11)	7 167 (11)	10 583 (11)	948 521 (12)	0.1160 (9)	0.7550 (8)
Azar-20%	0.3056 (12)	11 322 (12)	13 542 (12)	436 907 (11)	0.1435 (8)	0.8234 (6)
Best-fit	0.9136 (2)	3 787 (2)	5 678 (1)	15 273 (3)	0.0681 (12)	0.5935 (12)
Ded. I.	0.6615 (9)	5 231 (9)	8 057 (9)	162 928 (9)	1.0000 (1)	1.0000 (1)
First-fit	0.9287 (1)	3 726 (1)	5 756 (2)	35 142 (5)	0.0649 (13)	0.5850 (13)
KU	0.8772 (3)	3 944 (3)	5 817 (3)	11 382 (2)	0.4534 (5)	0.7513 (9)
LDBR	0.8646 (4)	4 003 (4)	5 885 (4)	10 674 (1)	0.1444 (7)	0.7563 (7)
Next-fit	0.1130 (13)	30 608 (13)	43 489 (13)	2 211 258 (13)	0.5297 (4)	0.9896 (3)
PCUF	0.7554 (7)	4 580 (7)	6 916 (7)	59 832 (6)	0.8390 (3)	0.9062 (4)
PSSF-5%	0.8330 (5)	4 154 (5)	6 141 (5)	145 962 (8)	0.1953 (6)	0.8457 (5)
PSSF-20%	0.5613 (10)	6 164 (10)	8 780 (10)	294 399 (10)	0.9155 (2)	0.9897 (2)
Random	0.8115 (6)	4 264 (6)	6 493 (6)	20 802 (4)	0.0751 (11)	0.6323 (11)
Worst-fit	0.7387 (8)	4 684 (8)	7 153 (8)	79 196 (7)	0.0790 (10)	0.6335 (10)

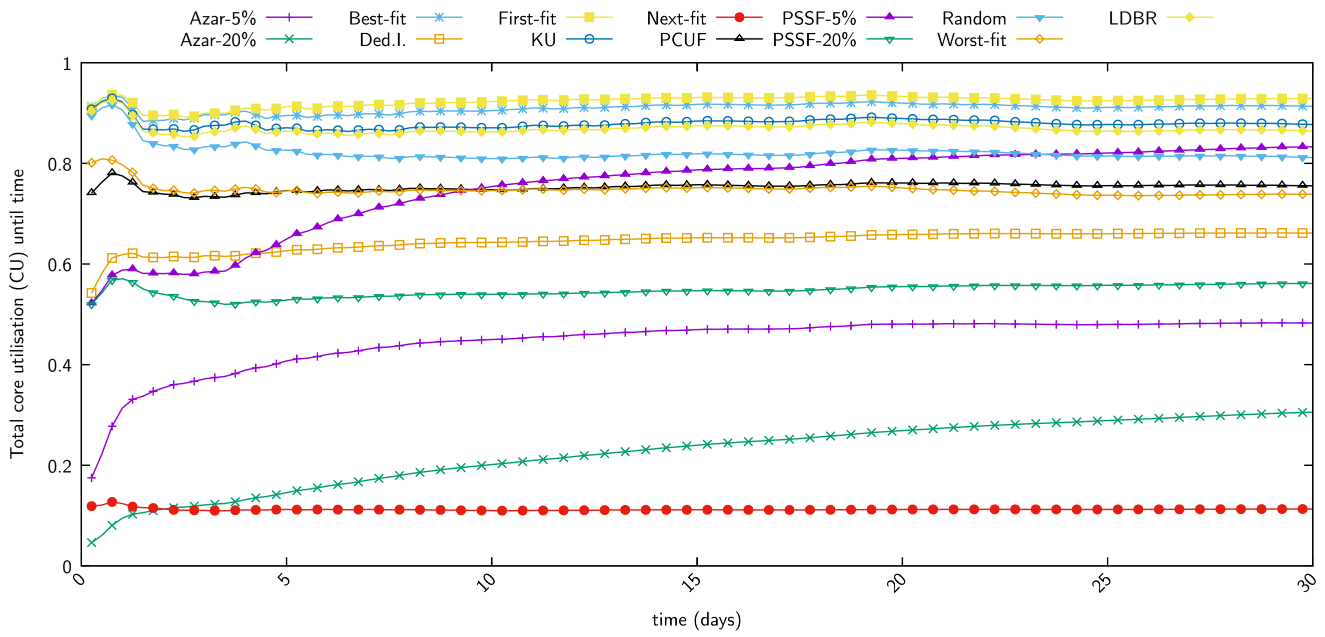

Figure 1: Changes in core utilisation over time

Fig. 3 shows the UCLR for proportions of malicious users between 1 % and 20 %. In line with the results of Agarwal and Duong [1], the results indicate that an increase in the proportion of malicious users leads to a reduction in UCLR. The only outlier is dedicated-instance, for which UCLR is always 1. While the UCLR of KU falls short of dedicated-instance, PSSF-20 %, PCUF and next-fit, it exceeds that of all other strategies.

Overall, the KU strategy achieves a very good CU, being beaten by just first-fit and best-fit. At the same time, it achieves a relatively good UCLR, far exceeding that of first-fit and best-fit. For all strategies that achieve a higher UCLR than KU, the CU falls short significantly. PCUF comes closest, but its CU is still lower by 0.1218, resulting in an increase in the average number of active servers and thus energy consumption of 16.1 %. A disadvantage, however, is that KU does not perform as well in terms of VMCLR: It is outperformed by LDBR and PSSF-5 %, whose CU is slightly lower, but still exceeds 0.83. Note, however, that these strategies performs significantly worse in terms of UCLR, making them less suitable for scenarios where an attack on any VM of a user would completely compromise their security. The algorithm with the highest CU beating KU in both UCLR and VMCLR is PCUF.

Some placement strategies rely on information about which servers a user's VMs have previously been hosted on or which users' VMs have previously been co-located on the same servers.

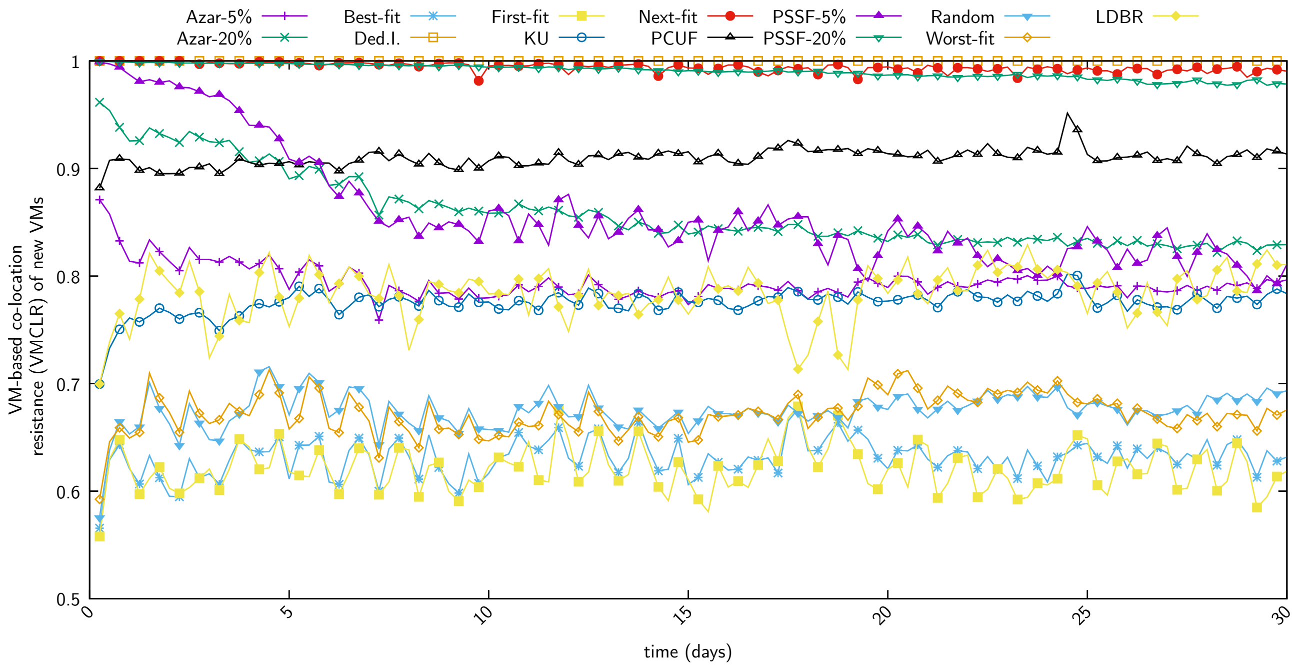

Figure 2: Changes of the co-location resistance for new VMs (NewVMCLR) over time with 5 % of malicious users

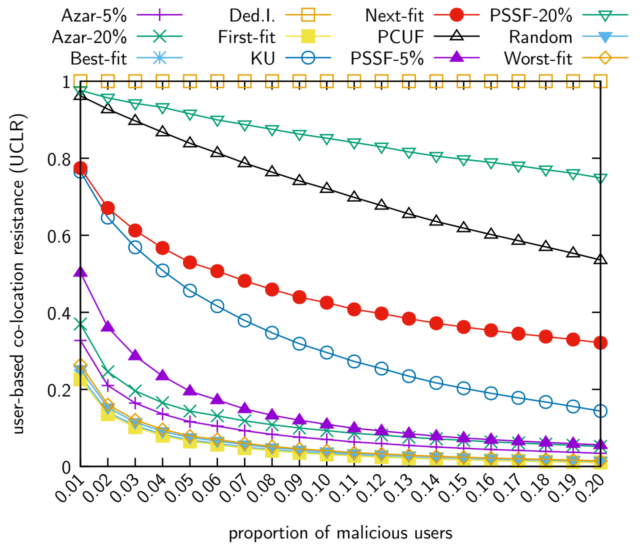

Figure 3: User-based co-location resistance depending on the proportion of malicious users

This means that additional data needs to be stored by the system making the placement decisions. For PSSF-5 %, 85 787 user-server relations had to be stored on average per run. An average of 147 684 user-user relations had to be stored per run for KU. For PCUF, 14 134 user-user relations had to be stored. For LDBR, a reputation value has to be stored for each user, in addition to any data that is needed for its calculation.

The results indicate that the amount of additional data which needs to be stored per customer is negligible compared to the memory size of a VM. The amount of data that has to be stored in total is also small enough to be handled on a central server, even for clouds having more than the 6 687 users in the dataset used here.

While this was not a problem for the 30-day run of the evaluation, the amount of data may rise over time due to fluctuations in the user base. To counter this, providers could delete placement data relating to users that were inactive for some time. While such users would be handled like new users if they became active again, this should not make co-location attacks significantly more likely if providers only delete placement data after a sufficiently long inactivity period.

The results indicate that some strategies are clearly inferior to others. Next-fit, Azar and PSSF-20 % achieve a lower CU than even the dedicated-instance strategy, which always achieves perfect CLR. In the configuration tested, LDBR is similar in terms of CU and VMCLR to KU, but performs significantly worse in terms of UCLR. Finally, the random and worst-fit strategies are beaten in CU, UCLR and VMCLR by multiple other strategies, including KU.

Generally, there is not a single optimum placement strategy, though. Any strategy is outperformed by others in either core utilisation or co-location resistance. Cloud service providers must make a trade-off between optimising resource utilisation and thus minimising energy consumption, on the one hand, and protecting users from being co-located with malicious users, on the other hand.

5.2 Isochronous attacks

For isochronous attacks, an attacker does not target specific users or VMs, but aims to be co-located with as many users or VMs as possible. The attacker launches a fixed number of VMs in fixed intervals. Three different isochronous attacker types were tested, which launched 1 VM every 60 minutes (**iso1**), 10 VMs every 600 minutes (**iso2**) or 5 VMs every 60 minutes (**iso3**). After launching a VM, it was left running for 10 minutes.

For each attacker type, two scenarios were considered: First, an attacker who uses the same account for launching all of their VMs (constant-ID scenario). On the other hand, an attacker who uses a new account for every attack interval (dynamic-ID scenario). Note that providers can make dynamic-ID attacks hard to perform by thoroughly authenticating a user's identity: If the same person tries to sign up more than once, the second sign-up can either be denied or be treated as if it had the same user ID as the first.

20 runs were performed for each configuration. The same PRNG seeds were used for each placement strategy.

Table 2 shows the UCLR of different placement strategies under isochronous attacks. For strategies not considering the user ID, the UCLR is identical for the constant-ID and dynamic-ID scenarios. For strategies that consider the user ID, the UCLR is lower for dynamic-ID attacks. This trend is especially strong for PCUF, which achieves a UCLR of 1 for constant-ID attacks, but is very susceptible to dynamic-ID attacks. Similar drops in performance of varying extent can be observed for PSSF-5 %, KU and LDBR.

For constant-ID attacks, there is no significant difference in the UCLR between the iso1 and iso2 scenarios. However, the UCLR achieved by first-fit, PSSF-5 % and worst-fit decreases, i. e. they are more susceptible to multiple attack VMs being launched simultaneously than to the VM launches being spread out over time. Under dynamic-ID attacks, the UCLR tends to be higher in the iso2 scenario for strategies considering user IDs. This means that these strategies (KU, PCUF, PSSF) are less vulnerable to attackers launching VMs in large bursts (with one user ID used per burst) instead of individually over time. LDBR, however, is instead more vulnerable to VMs being launched in large bursts.

In the iso3 scenario, the UCLR decreases for most strategies compared to iso1, i. e. launching more VMs per burst increases the success rate of an attack. However, KU, PCUF, dedicated-instance and PSSF-20 % are mostly resistant to increasing the number of VMs per burst.

KU outperforms all strategies exceeding its CU for isochronous constant-ID attacks. Only dedicated-instance, PCUF and PSSF-20 % beat it in UCLR, but achieve a much lower CU. In the dynamic-ID scenario, KU is outperformed by many other strategies, including the resource-efficient first-fit. Other strategies considering the user ID in their placement decisions face the same challenge, though.

5.3 Targeted attacks

In a *targeted* attack, an attacker targets a specific VM of a benign user. For this attack, we assume the attacker to have knowledge about when a target VM is launched. This may either be due to them being able to obtain this information, e. g. through a publicly available performance dashboard, or because they can actively cause a VM to be launched, e. g. by accessing a publicly available service which launches additional VMs to process requests under load. For the experiments, a random target VM was chosen every 300 minutes, resulting in a total of 8 638 targets per run. VMs were then launched in one or five bursts at or around the launch time of the targeted VM. Four targeted attacker types were tested, which launched five bursts of ten VMs each every minute starting at the launch time of the targeted VM (**tgt1**), one burst of 50 VMs one minute after the launch of the target (**tgt2**), five bursts of ten VMs each every minute starting two minutes before the launch of the target (**tgt3**) or five bursts of two VMs each every minute starting at the launch time of the target (**tgt4**). Attack VMs were shut down after ten minutes. Again, two scenarios were tested: an identical user ID throughout a run (constant-ID scenario) or a new user ID for every target (dynamic-ID scenario). 20 runs with different sets of malicious VMs were performed for each experiment configuration. The same 20 sets of malicious VMs were used for all strategies.

Table 3 shows the average number of target VMs seen by an attacker per experiment run. Again, there is no difference in results between constant-ID and dynamic-ID attacks for strategies not considering the ownership of VMs. The results indicate that dedicated-instance and next-fit completely prevent malicious VMs from being co-located with their targets. PSSF-20 % and PCUF also allow only few successful attacks in most scenarios. PCUF is, however, vulnerable to a constant-ID attacker launching VMs slightly *before* the target VM (scenario tgt3). The remaining strategies considering user IDs (KU, PSSF-5 %) perform similar for constant-ID attacks to those not considering user IDs. However, KU, PCUF and PSSF are actually more effective at preventing dynamic-ID attacks.

With the exception of dedicated-instance, LDBR, next-fit and worst-fit, launching all attack VMs in a single burst a minute after the target (tgt2) slightly increases the chance of an attacker achieving co-location. For almost all strategies, beginning to launch the first VMs two minutes *before* the target (tgt3) decreases the chances of a successful attack. The chance of a constant-ID attacker succeeding increases by more than 600 times for PCUF, though. Launching fewer VMs (tgt4) significantly decreases the risk of a successful attack in almost all cases. However, the risk increases more than 100-fold for a constant-ID attack on PCUF.

With respect to the newly proposed KU strategy, the results indicate a reversal of the trend observed for isochronous attacks: KU is outperformed by other strategies including the resource-efficient first-fit for constant-ID attacks. It performs respectably under dynamic-ID attacks, though, and is only outperformed by strategies achieving a lower CU.

6 CONCLUSION AND FUTURE WORK

This paper presents the open-source simulation framework VM-PlaceSim for the evaluation of VM placement strategies. The simulation results offer an insight into the resource utilisation and co-location resistance of placement strategies. The framework is extensible: Additional placement strategies can easily be added. Information about the VMs to be simulated can be supplied in CSV format. Loading the publicly available Microsoft Azure datasets [9, 10] containing real-world VM deployment data is supported.

Additionally, the paper presents the known-users (KU) placement strategy. This was evaluated alongside existing strategies with respect to effective resource utilisation and co-location resistance. VM-based co-location resistance is introduced as an additional

Table 2: User-based CLR of different placement strategies for isochronous attackers (strategy ranks in parentheses)

	Constant user ID			Dynamic user ID		
Strategy	iso1	iso2	iso3	iso1	iso2	iso3
Azar-5%	0.2620 (11)	0.2631 (11)	0.1603 (11)	0.2620 (9)	0.2631 (11)	0.1603 (11)
Azar-20%	0.2929 (10)	0.2910 (9)	0.1791 (10)	0.2929 (7)	0.2910 (9)	0.1791 (9)
Best-fit	0.2027 (13)	0.2019 (13)	0.1276 (12)	0.2027 (12)	0.2019 (13)	0.1276 (12)
Ded. I.	1.0000 (1)	1.0000 (1)	1.0000 (1)	1.0000 (1)	1.0000 (1)	1.0000 (1)
First-fit	0.6886 (8)	0.5168 (8)	0.5683 (7)	0.6886 (4)	0.5168 (4)	0.5683 (3)
KU	0.9731 (4)	0.9611 (4)	0.9650 (4)	0.1870 (13)	0.3707 (7)	0.1731 (10)
LDBR	0.8497 (5)	0.7415 (5)	0.7651 (5)	0.6203 (5)	0.4854 (6)	0.4572 (5)
Next-fit	0.7485 (7)	0.7396 (6)	0.5282 (8)	0.7485 (3)	0.7396 (3)	0.5282 (4)
PCUF	1.0000 (1)	1.0000 (1)	1.0000 (1)	0.2147 (10)	0.4918 (5)	0.2146 (8)
PSSF-5%	0.7826 (6)	0.6192 (7)	0.6585 (6)	0.2711 (8)	0.2950 (8)	0.2257 (7)
PSSF-20%	0.9975 (3)	0.9983 (3)	0.9975 (3)	1.0000 (1)	1.0000 (1)	1.0000 (1)
Random	0.2110 (12)	0.2102 (12)	0.1244 (13)	0.2110 (11)	0.2102 (12)	0.1244 (13)
Worst-fit	0.3888 (9)	0.2889 (10)	0.2463 (9)	0.3888 (6)	0.2889 (10)	0.2463 (6)

Table 3: Average number of target VMs seen by attackers targeting 8 638 VMs per run (strategy ranks in parentheses)

	Constant user ID				Dynamic user ID			
Strategy	tgt1	tgt2	tgt3	tgt4	tgt1	tgt2	tgt3	tgt4
Azar-5%	65.6 (6)	71.0 (6)	36.4 (5)	12.6 (5)	65.6 (8)	71.0 (9)	36.4 (9)	12.6 (9)
Azar-20%	32.6 (5)	36.6 (5)	16.8 (4)	6.7 (4)	32.6 (6)	36.6 (7)	16.8 (6)	6.7 (6)
Best-fit	150.3 (12)	170.6 (13)	82.2 (11)	34.0 (11)	150.3 (12)	170.6 (13)	82.2 (12)	34.0 (12)
Ded. I.	0.0 (1)	0.0 (1)	0.0 (1)	0.0 (1)	0.0 (1)	0.0 (1)	0.0 (1)	0.0 (1)
First-fit	105.6 (8)	116.0 (8)	60.6 (7)	28.9 (9)	105.6 (10)	116.0 (10)	60.6 (10)	28.9 (11)
KU	133.5 (10)	136.7 (10)	92.3 (12)	29.7 (10)	23.4 (5)	24.8 (5)	10.7 (5)	4.6 (5)
LDBR	141.3 (11)	139.1 (11)	69.3 (10)	36.4 (12)	70.2 (9)	70.9 (8)	23.2 (7)	12.5 (8)
Next-fit	0.0 (1)	0.0 (1)	0.0 (1)	0.0 (1)	0.0 (1)	0.0 (1)	0.0 (1)	0.0 (1)
PCUF	0.1 (3)	1.3 (3)	62.4 (9)	13.9 (6)	3.6 (4)	4.2 (4)	4.1 (4)	2.9 (4)
PSSF-5%	95.0 (7)	106.1 (7)	47.4 (6)	20.4 (7)	33.3 (7)	34.0 (6)	24.5 (8)	10.9 (7)
PSSF-20%	2.1 (4)	2.5 (4)	0.0 (1)	0.0 (1)	0.1 (3)	0.0 (1)	0.3 (3)	0.3 (3)
Random	111.8 (9)	124.0 (9)	61.4 (8)	23.4 (8)	111.8 (11)	124.0 (11)	61.4 (11)	23.4 (10)
Worst-fit	179.2 (13)	153.1 (12)	103.2 (13)	39.3 (13)	179.2 (13)	153.1 (12)	103.2 (13)	39.3 (13)

metric to evaluate the security of a placement strategy against co-location attacks and is used in the evaluation.

The results indicate that some placement strategies perform better than certain others with respect to both criteria, such as the dedicated-instance strategy compared to the previously-selected-servers-first (PSSF) strategy with groups of 20 % of total servers. While other strategies, such as previously-co-located-users-first (PCUF), are even more resistant against co-location attacks, the known-users (KU) strategy achieves a much higher co-location resistance than any other placement strategy exceeding its resource utilisation. Overall, there is not a single optimum placement strategy: resource utilisation and co-location resistance form a trade-off.

Strategies also differ in their ability to counter specific attacker behaviours. If an attacker always using the same user ID, strategies considering the user ID in placement decisions are relatively good at preventing attacks, even if these take place over a long time. If an attacker constantly changes accounts and is not prevented from this by the provider, these strategies lose their advantage over those not considering the user ID, though. If attackers are able to time the launch of their VMs around the launch of a target VM, this trend reverses: Strategies considering the user ID in their decisions are actually more vulnerable to attacks with a constant user ID.

Potential future work includes extending the framework to support simulating complex multi-zone cloud environments and evaluating placement strategies for these. More complex placement strategies could also be examined. CPU oversubscription could also be considered, e. g. by basing placement decisions on a prediction of future CPU usage of VMs, as proposed by Cortez et al. [11].

REFERENCES

[1] Amit Agarwal and Ta Nguyen Binh Duong. 2019. Secure virtual machine placement in cloud data centers. *Future Gener. Comput. Syst.* 100 (2019), 210–222.

[2] Amazon AWS. 2021. Amazon EC2 Dedicated Instances. https://aws.amazon.com/ec2/pricing/dedicated-instances/ (visited 2023-03-16).

[3] Ahmed Osama Fathy Atya, Zhiyun Qian, Srikanth V. Krishnamurthy, Thomas La Porta, Patrick D. McDaniel, and Lisa M. Marvel. 2019. Catch Me if You Can: A Closer Look at Malicious Co-Residency on the Cloud. *IEEE/ACM Trans. Netw.* 27, 2 (2019), 560–576.

[4] Yossi Azar, Seny Kamara, Ishai Menache, Mariana Raykova, and F. Bruce Shepherd. 2014. Co-Location-Resistant Clouds. In *ACM Cloud Computing Security Workshop, CCSW*. ACM, 9–20.

[5] Antonio Barresi, Kaveh Razavi, Mathias Payer, and Thomas R. Gross. 2015. CAIN: Silently Breaking ASLR in the Cloud. In *USENIX Workshop on Offensive Technologies, WOOT*. USENIX Association.

[6] D. F. C. Brewer and Michael J. Nash. 1989. The Chinese Wall Security Policy. In *IEEE Symposium on Security and Privacy,*. IEEE Computer Society, 206–214.

[7] Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, César A. F. De Rose, and Rajkumar Buyya. 2011. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw. Pract. Exp.* 41, 1 (2011), 23–50.

[8] Ron Chi-Lung Chiang, Sundaresan Rajasekaran, Nan Zhang, and H. Howie Huang. 2015. Swiper: Exploiting Virtual Machine Vulnerability in Third-Party Clouds with Competition for I/O Resources. *IEEE Trans. Parallel Distributed Syst.* 26, 6 (2015), 1732–1742.

[9] Eli Cortez. 2017. AzurePublicDatasetV1. https://github.com/Azure/AzurePublicDataset/blob/79bca52b02b87d64e332de5533d417981abb3f90/AzurePublicDatasetV1.md (visited 2023-03-16).

[10] Eli Cortez. 2019. AzurePublicDatasetV2. https://github.com/Azure/AzurePublicDataset/blob/79bca52b02b87d64e332de5533d417981abb3f90/AzurePublicDatasetV2.md (visited 2023-03-16).

[11] Eli Cortez, Anand Bonde, Alexandre Muzio, Mark Russinovich, Marcus Fontoura, and Ricardo Bianchini. 2017. Resource Central: Understanding and Predicting Workloads for Improved Resource Management in Large Cloud Platforms. In *Symposium on Operating Systems Principles, SOSP*. ACM, 153–167.

[12] Daniel Espling, Lars Larsson, Wubin Li, Johan Tordsson, and Erik Elmroth. 2016. Modeling and Placement of Cloud Services with Internal Structure. *IEEE Trans. Cloud Comput.* 4, 4 (2016), 429–439.

[13] Mauro Gaggero and Luca Caviglione. 2019. Model Predictive Control for Energy-Efficient, Quality-Aware, and Secure Virtual Machine Placement. *IEEE Trans Autom. Sci. Eng.* 16, 1 (2019), 420–432. https://doi.org/10.1109/TASE.2018.2826723

[14] Berk Gülmezoglu, Thomas Eisenbarth, and Berk Sunar. 2017. Cache-Based Application Detection in the Cloud Using Machine Learning. In *ACM ASIA Conference on Computer and Communications Security, ASIACCS*. ACM, 288–300.

[15] Ori Hadary, Luke Marshall, Ishai Menache, Abhisek Pan, Esaias E. Greeff, David Dion, Star Dorminey, Shailesh Joshi, Yang Chen, Mark Russinovich, and Thomas Moscibroda. 2020. Protean: VM Allocation Service at Scale. In *USENIX Symposium on Operating Systems Design and Implementation, OSDI*. USENIX Association, 845–861.

[16] Yi Han, Tansu Alpcan, Jeffrey Chan, and Christopher Leckie. 2013. Security Games for Virtual Machine Allocation in Cloud Computing. In *Conference Decision and Game Theory for Security, GameSec*. Springer, 99–118.

[17] Yi Han, Jeffrey Chan, Tansu Alpcan, and Christopher Leckie. 2017. Using Virtual Machine Allocation Policies to Defend against Co-Resident Attacks in Cloud Computing. *IEEE Trans. Dependable Secur. Comput.* 14, 1 (2017), 95–108.

[18] Brian Hay, Kara L. Nance, and Matt Bishop. 2011. Storm Clouds Rising: Security Challenges for IaaS Cloud Computing. In *Hawaii International Conference on System Sciences, HICSS*. IEEE Computer Society.

[19] Yoongu Kim, Ross Daly, Jeremie S. Kim, Chris Fallin, Ji-Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu. 2014. Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors. In *ACM/IEEE International Symposium on Computer Architecture, ISCA*. IEEE Computer Society, 361–372.

[20] Paul Kocher, Jann Horn, Anders Fogh, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, and Yuval Yarom. 2019. Spectre Attacks: Exploiting Speculative Execution. In *IEEE Symposium on Security and Privacy*. IEEE, 1–19.

[21] Younggyun Koh, Rob C. Knauerhase, Paul Brett, Mic Bowman, Zhihua Wen, and Calton Pu. 2007. An Analysis of Performance Interference Effects in Virtual Environments. In *IEEE International Symposium on Performance Analysis of Systems and Software, ISPASS*. IEEE Computer Society, 200–209.

[22] Jens Lindemann and Mathias Fischer. 2018. A memory-deduplication side-channel attack to detect applications in co-resident virtual machines. In *ACM Symposium on Applied Computing, SAC*. ACM, 183–192.

[23] Mohammad Masdari, Sayyid Shahab Nabavi, and Vafa Ahmadi. 2016. An overview of virtual machine placement schemes in cloud computing. *J. Netw. Comput. Appl.* 66 (2016), 106–127.

[24] Soo-Jin Moon, Vyas Sekar, and Michael K. Reiter. 2015. Nomad: Mitigating Arbitrary Cloud Side Channels via Provider-Assisted Migration. In *ACM Conference on Computer and Communications Security, CCS*. ACM, 1595–1606.

[25] Diego Perez-Botero, Jakub Szefer, and Ruby B. Lee. 2013. Characterizing hypervisor vulnerabilities in cloud computing servers. In *International Workshop on Security in Cloud Computing, SCC*. ACM, 3–10.

[26] Thomas Ristenpart, Eran Tromer, Hovav Shacham, and Stefan Savage. 2009. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In *ACM Conference on Computer and Communications Security, CCS*. ACM, 199–212.

[27] Venkatanathan Varadarajan, Yinqian Zhang, Thomas Ristenpart, and Michael M. Swift. 2015. A Placement Vulnerability Study in Multi-Tenant Public Clouds. In *USENIX Security Symposium*. USENIX Association, 913–928.

[28] Ofir Weisse, Jo Van Bulck, Marina Minkin, Daniel Genkin, Baris Kasikci, Frank Piessens, Mark Silberstein, Raoul Strackx, Thomas F Wenisch, and Yuval Yarom. 2018. *Foreshadow-NG: Breaking the virtual memory abstraction with transient out-of-order execution.* Technical Report. KU Leuven. https://lirias.kuleuven.be/2089352?limo=0 (visited 2023-03-16).

[29] Yiming Xiao, Liang Liu, Zuchao Ma, Zijie Wang, and Weizhi Meng. 2021. Defending co-resident attack using reputation-based virtual machine deployment policy in cloud computing. *Trans. Emerg. Telecommun. Technol.* 32, 9 (2021).

[30] Yuan Xiao, Xiaokuan Zhang, Yinqian Zhang, and Radu Teodorescu. 2016. One Bit Flips, One Cloud Flops: Cross-VM Row Hammer Attacks and Privilege Escalation. In *USENIX Security Symposium*. USENIX Association, 19–35.

[31] Zhang Xu, Haining Wang, and Zhenyu Wu. 2015. A Measurement Study on Co-residence Threat inside the Cloud. In *USENIX Security Symposium*. USENIX Association, 929–944.

[32] Shungeng Zhang, Huasong Shan, Qingyang Wang, Jianshu Liu, Qiben Yan, and Jinpeng Wei. 2019. Tail Amplification in n-Tier Systems: A Study of Transient Cross-Resource Contention Attacks. In *IEEE International Conference on Distributed Computing Systems, ICDCS*. IEEE, 1527–1538.

[33] Tianwei Zhang, Yinqian Zhang, and Ruby B. Lee. 2017. DoS Attacks on Your Memory in Cloud. In *ACM ASIA Conference on Computer and Communications Security, AsiaCCS*, Ramesh Karri, Ozgur Sinanoglu, Ahmad-Reza Sadeghi, and Xun Yi (Eds.). ACM, 253–265.

[34] Yinqian Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. 2012. Cross-VM side channels and their use to extract private keys. In *ACM Conference on Computer and Communications Security, CCS*. ACM, 305–316.