Ezhilchelvan P, Mitrani I. Evaluating the Probability of Malicious Co-residency in Public Clouds. *IEEE Transactions on Cloud Computing* 2015. DOI: 10.1109/TCC.2015.2451633

Copyright:

© 2015 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

DOI link to article:

<http://dx.doi.org/10.1109/TCC.2015.2451633>

Date deposited:

12/01/2016

Evaluating the Probability of Malicious Co-residency in Public Clouds

Paul Ezhilchelvan and Isi Mitrani

Abstract—We examine a system where servers can host several virtual machines in parallel and where some of the users are malicious. Arrivals and departures of both normal and malicious users are governed by random processes. The aim is to estimate the probability that a possible target will find itself sharing a server with an attacker. Two allocation policies for assigning virtual machines to servers are studied. In both cases, as well as attacks forming part of the arrival process, multiple simultaneous attacks are considered. Closed-form expressions for the desired estimates are obtained. Comparisons with simulations for purposes of validation are presented and the effect of increasing the number of available servers is illustrated.

Keywords: cloud security, co-location attacks, virtual machines, stochastic modeling, simulation.

I. INTRODUCTION

The Infrastructure-as-a-Service (IaaS) model enables enterprises to purchase computing resources on-the-fly and thus relieves them from overheads related to maintaining the required resources themselves. Cloud Infrastructure Providers (CIPs), such as Amazon EC2 and Microsoft Azure, respond to demands by multiplexing several customers' workloads on a single physical machine; typically, virtualization is used to encapsulate a customer's workload inside a virtual machine (VM). Thus, VMs belonging to, and controlled by, different customers can be hosted at the same server. Such co-resident VMs also share the host's physical resources such as CPU, cache, local disk and network interface. Virtualization offers advantages such as ease of deployment, economies of scale and lower cost per customer.

However, co-residency also introduces the possibility that VMs might interfere with each other's information or resources. Cloud Providers commonly employ hypervisors or virtual machine monitors to accomplish strong isolation between co-resident VMs. There have been several proposals for the effective design of such tools (e.g., see [13], [11], [4], [3]). Nevertheless, sharing of hardware components between VMs has repeatedly been shown to introduce vulnerabilities that can be exploited by cross-VM side-channel attacks to gain access to confidential information, or an enhanced performance at the victim's expense (see [12], [18], [2], [5], [10], [16]). The only sure way of avoiding interference is to statically partition the server resources between VMs (e.g., see [7], [15]), or to offer a single-tenant service. Such approaches are inefficient and/or expensive. Multi-tenancy provisioning is still widely offered.

Clearly, the possibility of security breaches due to malicious co-resident VMs is of a serious concern to both cloud

providers and cloud customers (see [17]). The aim of this paper is to quantify the extent to which that concern is justified. We formalize, and then answer, the following questions:

- i. In the normal course of events, with VMs being allocated and de-allocated as time passes according to some random processes, and assuming that a certain fraction of them are malicious, what is the probability that a target will find itself sharing a server with at least one attacker?
- ii. If an attacker is aware that a given target has just been admitted in the system, and immediately launches several simultaneous VMs, what is the probability that at least one of them will share a server with the target?

The answers to these questions depend, of course, on the manner in which VMs are allocated to servers. We examine two allocation policies: (a) *Random*, whereby a server is chosen with equal probability among the set of servers that are not fully occupied; (b) *Priority Blocks*, where the servers are grouped into blocks and prioritized; new VMs are allocated to a lower priority block only when all higher priority ones are full; within blocks, the random policy is used. This policy would be used if power saving is an objective of the Cloud provider. Note that when all blocks contain just one server, the Priority Blocks policy coincides with the well-known *First-Fit* policy.

We are interested only in the *possibility* of a co-resident attack. Whether the attack is successful or not is a question outside the scope of the present paper (but see the Conclusion for a possible extension). Thus, what we evaluate can be considered as an *upper bound* on the probability that the target will actually suffer damage as a result of an attack. Another feature of question (i) that is worth pointing out is that a malicious VM need not join the target's server *after* the target has started running. It could already be present at the server, by a combination of design and luck, when the target is admitted.

The fraction of malicious VMs represents the threat level that a possible target faces. This is a model parameter which can be varied, along with other parameters describing the loading conditions. For a given parameter set, we provide closed-form expressions for evaluating the desired probabilities.

The applications of our results have to do with the cost–security and/or cost–performance trade-offs faced by both the cloud providers and their customers. A provider may decide, at a higher cost, to increase security by investing in special protection software (e.g., see Steinburg and Kauer [14]), which may interfere with performance, or by increasing the number of servers (we show that the latter lowers the probability of malicious co-residency). The customer, on the other hand, may have the choice of using the standard multi-tenancy service, or paying extra for a single-tenancy one.

In order to make intelligent decisions, and to optimize expected benefit, both the provider and the users would need to know the probability of malicious co-residency. That probability can also be incorporated by the provider in any service-level agreement that focuses on security.

In a practical application, the solutions we provide would need to be combined with traffic monitoring software that would provide current estimates for the job arrival rate and average residence time. The problems of parameter estimation are not new and are quite well understood.

The tools employed in the analysis come from the fields of stochastic processes and queueing theory. While their use in performance engineering is well known, this is a novel application in the area of security.

To the best of our knowledge, the problem considered here has not been addressed before. Perhaps the closest work is by Azar et al [1], who consider a variant of question (ii) in the context of a static system (no processes of arrivals and departures) with a particular VM allocation policy. Other research has focused on demonstrating that a malicious co-residency can verifiably be accomplished (Ristenpart et al [12]), or has been concerned with techniques for detecting co-residency with specific targets (Zhang et al [19], Bates et al [2], Herzberg et al [6]). Despite action taken by the cloud providers since the publication of [12], the possibility of successful attack has not been eliminated.

Section 2 presents the model, analysis and results corresponding to the random allocation policy. It contains a subsection that deals with the case of multiple simultaneous attacks. Section 3 is concerned with the priority blocks policy. It also contains a subsection dealing with multiple simultaneous attacks. In each section, estimates relying on approximations are validated by comparing them with simulations. Section 4 contains the conclusions and suggestions for future work.

II. RANDOM ALLOCATION POLICY

In existing clouds servers are usually divided into disjoint sets, dedicated to different types of service. For example, in EC2, types are defined by combinations of VM resource requirements. Servers in a given set host only VMs of the corresponding type. Hence, an attacker attempting to co-locate with a target VM must make a request of the appropriate type. This means that each set of servers can be considered in isolation, independently of the others. Moreover, we assume that the servers in a given set can host the same maximum number of VMs. In EC2, that number is about 8 for VMs of type ‘small’.

With these observations in mind, we make the following assumptions.

A service set contains N identical servers, each of which can run a maximum of m parallel virtual machines. Thus, there can be at most $K = Nm$ user jobs running at any one time. Jobs arrive into the system in a Poisson stream at rate λ and have residence times that are i.i.d. random variables distributed exponentially with mean $1/\mu$. The offered load is therefore $\rho = \lambda/\mu$. Any incoming job that finds all virtual machines busy (i.e., K jobs already present) is lost.

When a job is admitted into the system, it is assigned to a server chosen at random (i.e., with equal probability) among the set of servers where there is at least one unoccupied virtual machine. Jobs that reside on virtual machines running on the same server are said to ‘share’ that server. It is assumed that the residence times do not depend on the number of jobs sharing a server.

Some of the jobs are malicious, and may be able to inflict damage on any job that shares a server with them. Such jobs will be referred to as ‘attackers’. An incoming job is an attacker with probability α , independently of past arrivals. A job is said to be placed ‘in jeopardy’ if (a) on arrival it is assigned to a server which contains at least one attacker, or (b) at least one attacker arrives and is assigned to the same server during the job’s residence in the system.

The problem is to estimate the steady-state probability, h , that a job admitted into the system will be placed in jeopardy.

The above assumptions imply that the number of jobs in the system behaves like an $M/M/K/K$ queue, i.e. an Erlang loss system with K trunks and an offered load ρ . The probability that an incoming job is admitted is equal to $1 - B(\rho, K)$, where

$$B(x, n) = \frac{x^n}{n!} \left[\sum_{j=0}^n \frac{x^j}{j!} \right]^{-1} \quad (1)$$

is the Erlang-B function (also known as Erlang loss formula; see, for example, [9]).

Since all servers are treated equally, the steady-state rate γ at which jobs are admitted into any given server is equal to

$$\gamma = \frac{1}{N} \lambda [1 - B(\rho, K)] \quad (2)$$

Let q_j be the probability that, at the moment when a job is admitted into a server, it shares that server with j other jobs, $j = 0, 1, \dots, m-1$. These probabilities can be estimated by assuming that jobs are admitted into the server in a Poisson stream, with rate γ given by (2). In other words, during periods when there is at least one available virtual machine, a server is approximated by an $M/M/m-1/m-1$ queue with offered load $\sigma = \gamma/\mu$. This gives the expressions

$$q_j = \frac{\sigma^j}{j!} \left[\sum_{i=0}^{m-1} \frac{\sigma^i}{i!} \right]^{-1} \quad ; \quad j = 0, 1, \dots, m-1 \quad (3)$$

Consider now a target job which is admitted into a server and finds j other jobs sharing. If at least one of those jobs is an attacker, which occurs with probability $1 - (1 - \alpha)^j$, then the job is in jeopardy immediately. Otherwise, with probability $(1 - \alpha)^j$, the job is safe on arrival, but may be placed in jeopardy later. Its residence in the server may then be modelled by the Markov process whose state transition diagram is illustrated in figure 1.

The process is in state j , $j = 0, 1, \dots, m-1$, if there are j other jobs sharing the server. If $j < m-1$ and an attacker arrives, with rate $\alpha\gamma$, then the process enters an absorbing state, ‘hit’: the target is placed in jeopardy. If the target job departs, with rate μ , then another absorbing state, ‘miss’, is entered: the job escapes safely. The other transitions represent

arrivals and departures of ordinary jobs. State $m - 1$ can be left only as a result of a departure.

Fig. 1. Residence of a target job

Denote by h_j the probability that the process will be absorbed in state ‘hit’, given that it is currently in state j , i.e. the target job shares the server with j other jobs, none of which is an attacker ($j = 0, 1, \dots, m - 1$). If $j < m - 1$, then the next event to occur is one of the following:

- (a) arrival of an attacker, with probability

$$\frac{\alpha\gamma}{\gamma + (j+1)\mu}.$$

The target will then be in jeopardy.

- (b) arrival of a normal job, with probability

$$\frac{(1-\alpha)\gamma}{\gamma + (j+1)\mu}.$$

The process will then be in state $j + 1$.

- (c) departure of one of the other jobs, with probability

$$\frac{j\mu}{\gamma + (j+1)\mu}.$$

The process will then be in state $j - 1$.

- (d) departure of the target, with probability

$$\frac{\mu}{\gamma + (j+1)\mu}.$$

The process is absorbed in state ‘miss’.

If $j = m - 1$, all virtual machines are busy and no new jobs can be admitted. Then the only possible transitions are to state $m - 2$, with probability $(m - 1)/m$, or to a safe completion, with probability $1/m$.

Note that, dividing numerator and denominator by μ leads to expressions for the above probabilities that depend only on the offered load, $\sigma = \gamma/\mu$, rather than on γ and μ individually.

We can now write the following equations for the probabilities h_j . When $j < m - 1$ we have

$$h_j = \frac{\alpha\sigma}{\sigma + j + 1} + \frac{(1-\alpha)\sigma}{\sigma + j + 1}h_{j+1} + \frac{j}{\sigma + j + 1}h_{j-1}, \quad (4)$$

and when $j = m - 1$,

$$h_{m-1} = \frac{m-1}{m}h_{m-2}. \quad (5)$$

These equations allow h_j to be determined easily. The unconditional probability, h , that a target job will be placed in jeopardy, is then equal to

$$h = \sum_{j=0}^{m-1} [1 - (1-\alpha)^j + (1-\alpha)^j h_j] q_j, \quad (6)$$

with q_j given by (3).

It is perhaps worth mentioning that equations (1), (2) and (3) remain valid when the residence times have a general distribution with mean $1/\mu$. Equations (4) and (5) require exponentially distributed residence times, but they can still be used as approximations when that assumption is violated.

In figure 2, the estimates provided by equation (6) are compared with those obtained from simulations. The system examined has 100 servers, each running 5 virtual machines, for a total capacity of 500 virtual machines. 10% of all jobs are attackers. The same interarrival and service time distributions that were assumed in the model were also used in the simulation. Thus, what the simulation tested was the extent to which the non-Poisson nature of arrivals *into a server* affects the jeopardy probability.

Each simulated point is obtained from a simulation run covering 200000 admitted jobs. The time unit is taken as the average residence time, that is $1/\mu = 1$. The jeopardy probability is plotted against the offered load (i.e. the job arrival rate). The latter varies between 50 and 400, making the system occupancy vary between 10% and 80%.

Note that, for all of the above parameters, the vast majority of the incoming jobs are admitted. Even in the most heavily loaded case, when $\rho = 400$ and 80% of the virtual machines are busy, the probability of rejecting a job, $B(400, 500)$, is on the order of 10^{-7} .

The figure shows that the Poisson approximation of arrivals into a server is very acceptable. The differences between the estimated and simulated jeopardy probabilities are on the order of 10% or less, over the whole range of offered loads.

It is intuitively clear that when servers are allocated at random, the more servers are available, the less likely it is that a target job will be placed in jeopardy. To quantify this remark, figure 3 shows the dependency of the probability h on the number of servers, N . The offered load is fixed at two levels, $\rho = 400$ and $\rho = 200$, while N varies from 100 to 500 servers. The number of virtual machines per server and the fraction of attackers are $m = 5$ and $\alpha = 0.1$, as before. Thus, the virtual machine occupancy varies from 80% to 16% when $\rho = 400$, and from 20% to 8% when $\rho = 200$. Both the model estimates and the simulated ones (based on 200000 jobs admitted for each N) are plotted.

The figure confirms our intuition that the jeopardy probability decreases with N . Moreover, it is clear that the higher the offered load, the longer it takes for that probability to approach 0.

Another observation that can be made in this context is that when the offered load is low, compared to the number

Fig. 2. Modelled and simulated probabilities: random allocation

$$N = 100, m = 5, \alpha = 0.1$$

Fig. 3. Increasing number of servers

$$m = 5, \alpha = 0.1$$

of servers, the model estimate of h tends to become more accurate. This effect, which manifests itself quite slowly, is due to the fact that incoming jobs are less frequently having to be redirected to other servers because their first choice is full. Hence, the admissions of jobs into any given server are more closely approximated by a Poisson process.

A. Isolated multiple attacks

Consider a situation where a particular target job has just entered the system and has been assigned to a server. An attacker has somehow learned that the job is running and decides to attack it immediately, by submitting simultaneously

ℓ independent requests for virtual machines. This attack is an isolated occurrence; it is not part of the arrival stream and is assumed not to affect the long-term offered load.

Denote by r_ℓ the probability that the target is placed in jeopardy as a result of the ℓ -fold attack, i.e. at least one of the attacker's ℓ jobs is assigned to the same server as the target. This probability can be estimated by arguing as follows.

Let v_t be the probability that exactly t of the ℓ attacking jobs are admitted into the system ($t = 0, 1, \dots, \ell$). For $t < \ell$, that is the Erlang probability that exactly $K - 1 - t$ virtual machines

were occupied just before the target job was admitted:

$$v_t = \frac{\rho^{K-1-t}}{(K-1-t)!} \left[\sum_{i=0}^K \frac{\rho^i}{i!} \right]^{-1} ; \quad t = 0, 1, \dots, \ell-1 . \quad (7)$$

Then v_ℓ , which is the probability that *at most* $K-1-\ell$ virtual machines were occupied, is given by

$$v_\ell = 1 - \sum_{t=0}^{\ell-1} v_t . \quad (8)$$

Think of the job assignment mechanism as first choosing a server at random; then, if all its virtual machines are busy, choose another server at random; etc. Using that mechanism, if t attacking jobs are admitted into the system, each of them is first directed to the target server with probability $1/N$. The probability that at least one of them is first directed to the target server is equal to $1 - (1 - 1/N)^t$.

The server will accept a new job if it is not full, i.e. if the target job shares it with fewer than $m-1$ other jobs. The probability of that event is $1 - q_{m-1} = 1 - B(\sigma, m-1)$, as in equation (3). Hence, the probability r_ℓ is given by

$$r_\ell = [1 - B(\sigma, m-1)] \sum_{t=0}^{\ell} \left[1 - \left(\frac{N-1}{N} \right)^t \right] v_t . \quad (9)$$

Note that the term $1 - B(\sigma, m-1)$ places an upper bound on the probability r_ℓ . No matter how many attacking jobs are submitted simultaneously, the jeopardy probability cannot exceed that value.

III. PRIORITY BLOCKS

Suppose now that the random assignment of jobs to available servers is replaced by a priority allocation. More precisely, the N servers are divided into S groups, numbered $1, 2, \dots, S$. Group i contains N_i servers, hence a total of $k_i = N_i m$ virtual machines. Servers in group i have priority over those in group $i+1$, for all $i = 1, 2, \dots, S-1$. In other words, an incoming job would be assigned to a server in group $i+1$ only if all virtual machines in groups $1, 2, \dots, i$ are occupied. Within each group, a server is chosen at random among those with available virtual machines.

Note that, in the extreme case when the number of groups is N and each group contains just one server, the above policy becomes what is known as the ‘First-Fit’ policy: the servers are numbered from 1 to N and each incoming job is allocated to the server with a lowest index that has an available virtual machine.

A possible reason for implementing such a priority allocation policy would be to minimize energy consumption. Groups of servers could be powered down when not needed. However, our interest is in finding out what effect, if any, this policy has on the probability h of a job being placed in jeopardy, given that it has been admitted into the system. The assumptions concerning job arrivals, fraction of attackers and residence in the system are as before.

Denote by K_i the total number of virtual machines in groups $1, 2, \dots, i$: $K_i = k_1 + k_2 + \dots + k_i$. The probability that all those machines are occupied is given by the Erlang-B function

in equation (1), $B(\rho, K_i)$, with $\rho = \lambda/\mu$ being the offered load.

Let p_i be the probability that an incoming job is admitted into a server in group i . For $i = 1$ that is the probability that there is an available virtual machine in group 1, while for $i > 1$ it is the probability that all machines in groups $1, 2, \dots, i-1$ are occupied, but there is an available one in group i . Hence,

$$p_1 = 1 - B(\rho, K_1) ;$$

$$p_i = B(\rho, K_{i-1}) - B(\rho, K_i) ; \quad i = 2, 3, \dots, S . \quad (10)$$

Since all servers within a group are treated equally, the steady-state rate at which jobs are admitted into a given server in group i is

$$\gamma_i = \frac{\lambda p_i}{N_i} ; \quad i = 1, 2, \dots, S . \quad (11)$$

Now one can apply the approximations (3), with offered load $\sigma_i = \gamma_i/\mu$, to estimate the probabilities, $q_{i,j}$, that a job admitted into a server in group i will find itself sharing that server with j other jobs. A similar argument to that leading to (4) and (5) provides expressions for the probabilities, $h_{i,j}$, that a job admitted into a server in group i and finding j other jobs there, none of which is an attacker, will be placed in jeopardy. Then, equation (6), with h_j and q_j replaced by $h_{i,j}$ and $q_{i,j}$ respectively, gives the probability, h_i , that a job admitted into a server in group i will be placed in jeopardy.

Finally, the unconditional steady-state probability, h , that a job admitted into the system will be placed in jeopardy, is estimated as

$$h = \frac{1}{1 - B(\rho, K_S)} \sum_{i=1}^S h_i p_i . \quad (12)$$

The denominator in the right-hand side of (12) is the probability that an incoming job is admitted.

Figure 4 illustrates the modelled and simulated probabilities h for the same parameters as in figure 2, except that the 100 servers are divided into four groups of 25 servers each, and a priority assignment policy is used.

The jeopardy probabilities now tend to be higher than in the random assignment case, because the offered load tends to be distributed among fewer servers. However, the agreement between modelled and simulated values is again on the order of 10% or less.

A notable feature of this figure is that, in both model and simulation, the probability h is not a monotone function of the offered load. It seems that the reason for that non-monotonicity lies in the complex interplay between three trends: on one hand, a higher offered load implies a higher arrival rate of attackers. On the other hand, a higher offered load causes more servers to be used. Also, a job which shares a server with $m-1$ other jobs is protected from attack (because new arrivals are not admitted), and that happens more frequently in the present system.

As with the random assignment of jobs to servers, it is reasonable to ask what happens when the number of servers increases. The answer is less clear-cut in the present case. In figure 5, the number of groups is kept fixed at $S = 4$, while the number of servers per group increases from 25 to 125.

Fig. 4. Modelled and simulated probabilities: priority blocks
 $(N_1, N_2, N_3, N_4) = (25, 25, 25, 25)$, $m = 5$, $\alpha = 0.1$

Fig. 5. Increasing number of servers
 $(N_1, N_2, N_3, N_4) = (N/4, N/4, N/4, N/4)$, $m = 5$, $\alpha = 0.1$

The jeopardy probability is evaluated for two different offered loads, $\rho = 400$ and $\rho = 200$.

There is a general tendency for the jeopardy probability to decrease, but that decrease is not always monotone. In particular, when the offered load is 400, increasing the number of servers per group from 75 to 100 causes a slight increase in the value of h . This is due to the way the servers are used: when there are 375 virtual machines per group, the first group is not enough to cope with the load and the second group is frequently brought into play; when there are 500 virtual machines per group, the first group can cope and the second group is hardly used; fewer servers are active, on the average,

and their utilization is a little higher.

When the offered load is 200 and the group size becomes 50 (i.e., 250 virtual machines), the first group alone can cope with the load. Further increases in its size lead to lower jeopardy probability as in the case of random assignment.

If the total number of servers is increased, not by increasing the number of servers per group, but by keeping that number fixed and increasing the number of groups, then a different behaviour would be observed. After a certain point, the jeopardy probability would remain constant. Once there are enough groups so that their servers can cope with the offered load, adding more lower priority groups would not change anything

very much since those servers would stay largely unused. This remark applies, in particular, to the First-Fit allocation policy.

A. Isolated multiple attacks

In the context of priority server allocation, one can again imagine an isolated attack on a specific target by ℓ simultaneous submissions. To estimate the probability of placing the target in jeopardy, we adapt the argument in subsection II-A to the model with groups of servers and priorities.

Suppose that the target has joined a server in group i , having found all servers in groups $1, 2, \dots, i-1$ fully occupied. This happens with probability p_i , given by equation (10). Since the attackers are submitted at the same time, they are also directed to servers of priority i or lower. Let $v_{i,t}$ be the probability that exactly t of the ℓ attacking jobs are admitted into group i ($t = 0, 1, \dots, \ell$). These probabilities are given by expressions similar to equations (7) and (8):

$$v_{i,t} = \frac{\rho_i^{K_i-1-t}}{(K_i-1-t)!} \left[\sum_{j=0}^{K_i} \frac{\rho_i^j}{j!} \right]^{-1} ; \quad t = 0, 1, \dots, \ell-1, \quad (13)$$

$$v_{i,\ell} = 1 - \sum_{t=0}^{\ell-1} v_{i,t}, \quad (14)$$

where $\rho_i = \lambda p_i$ and $K_i = N_i m$ are the offered load and the number of virtual machines in group i , respectively.

If t of the attackers are admitted into group i , then the probability that at least one of them is directed to the same server as the target is $1 - [(N_i - 1)/N_i]^t$. That attacker will be admitted into the server if the target shares it with fewer than $m-1$ other jobs. The probability of that event is $1 - q_{i,m-1} = 1 - B(\sigma_i, m-1)$, with $\sigma_i = \lambda p_i / (N_i \mu)$, as in equation (3).

Hence, the probability $r_{i,\ell}$, that the target will be placed in jeopardy as a result of an ℓ -fold attack, given that it has joined group i , is given by

$$r_{i,\ell} = [1 - B(\sigma_i, m-1)] \sum_{t=0}^{\ell} \left[1 - \left(\frac{N_i - 1}{N_i} \right)^t \right] v_{i,t}. \quad (15)$$

The unconditional probability, h_ℓ , of achieving jeopardy with ℓ simultaneous attacks, is obtained as in (12):

$$h_\ell = \frac{1}{1 - B(\rho, K_S)} \sum_{i=1}^S r_{i,\ell} p_i. \quad (16)$$

Again, the factors $1 - B(\sigma_i, m-1)$ place an upper bound on the jeopardy probability.

IV. CONCLUSION

We have addressed the problem of assessing the likelihood of malicious co-residency in public clouds. Closed-form expressions have been obtained which allow the desired probabilities to be evaluated simply and efficiently. On the basis of the experiments conducted, the accuracy of the estimates appears to be acceptable. However, more extensive simulations, and possibly real life experiments, would be required in order to see whether that accuracy remains acceptable when either the

arrivals are not Poisson, or the VM residence times are not distributed exponentially, or both.

One may be interested in the probability, h' that a target suffers actual damage as a result of malicious co-residency. Assuming that the success probability, β , for a co-resident attack is known, is independent of the length of time the target remains in the system, and neglecting the possibility of more than one attack on the same target, h' can be approximated by βh . If that last possibility cannot be neglected, then a more complicated analysis would be needed in order to determine, not just the probability of at least one malicious co-residency, but also the *distribution* of the number of malicious co-residencies. The question would also arise whether the success of one attack is independent of the success or failure of another.

There are other algorithms for allocating VMs to servers that might be considered (see [8] et al). These include *least-full first*, *most-full first* and *next-fit* (whereby the list of servers is traversed cyclically). The last of these is likely to have a similar behaviour to the random policy, but the other two would be non-trivial to analyze since they introduce rather complex state-dependencies.

REFERENCES

- [1] Y. Azar, S. Kamara, I. Menache, M. Raykova and B. Shepherd, "Co-Location-Resistant Clouds", Procs 6th ACM Workshop on Cloud Computing Security (CCSW'14), Scottsdale, AZ, 2014.
- [2] A. Bates, B. Mood, J. Pletcher, H. Pruse, M. Valafar and K. Butler, "Detecting Co-residency with Active Traffic Analysis Techniques", Procs ACM Workshop on Cloud Computing Security (CCSW'12), Raleigh, NC, pp. 1-12, 2012.
- [3] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum and D. Boneh, "Terra: A Virtual Machine-based Platform for Trusted Computing", Procs 19th ACM Symposium on Operating Systems Principles (SOSP'03), Bolton Landing, NY, pp. 193-206, 2003.
- [4] D. Gupta, L. Cherkasova, R. Gardner and A. Vahdat, "Enforcing Performance Isolation Across Virtual Machines in Xen", Procs ACM/IFIP/USENIX International Conference on Middleware (Middleware'06), Melbourne, Australia, pp. 342-362, 2006.
- [5] T.G. Handel and M.T. Sandford, II, "Hiding Data in the OSI Network Model", Procs 1st International Workshop on Information Hiding, pp. 23-38, 1996.
- [6] A. Herzberg, H. Shulman, J. Ullrich and E. Weippl, "Cloudoscopy: Services Discovery and Topology Mapping", Procs ACM Workshop on Cloud Computing Security (CCSW'13), Berlin, pp. 113-122, 2013.
- [7] E. Keller, J. Szefer, J. Rexford and R.B. Lee, "NoHype: Virtualized Cloud Infrastructure Without the Virtualization", Procs 37th International Symposium on Computer Architecture (ISCA'10), Saint-Malo, France, pp. 350-361, 2010.
- [8] K. Mills, J. Filliben and C. Dabrowski, "Comparing VM-Placement Algorithms for On-Demand Clouds", Procs 3rd International Conference on Cloud Computing Technology and Science (CLOUDCOM'11), Washington, DC, pp. 91-98, 2011.
- [9] I. Mittrani, *Probabilistic Modelling*, Cambridge University Press, 1998.
- [10] K. Okamura and Y. Oyama, "Load-based Covert Channels Between Xen Virtual Machines", Procs ACM Symposium on Applied Computing (SAC'10), Sierre, Switzerland, pp. 173-180, 2010.
- [11] H. Raj, R. Nathuji, A. Singh and P. England, "Resource Management for Isolation Enhanced Cloud Services", Procs ACM Workshop on Cloud Computing Security (CCSW'09), Chicago, IL, pp. 77-84, 2009.
- [12] T. Ristenpart, E. Tromer, H. Shacham and S. Savage, "Hey, You, Get off of My Cloud: Exploring Information Leakage in Third-party Compute Clouds", Procs 16th ACM Conference on Computer and Communications Security (CCS'09), Chicago, IL, pp. 199-212, 2009.
- [13] A. Shieh, S. Kandula, A. Greenberg and C. Kim, "Seawall: Performance Isolation for Cloud Datacenter Networks", Procs 2nd USENIX Conference on Hot Topics in Cloud Computing (HotCloud'10), Boston, MA, pp. 1-1, 2010.

- [14] U. Steinberg and B. Kauer, "NOVA: A Microhypervisor-based Secure Virtualization Architecture", Procs. 5th European Conference on Computer Systems (EuroSys '10), pp. 209-222, Paris, 2010.
- [15] J. Szefer, E. Keller, R.B. Lee, and J. Rexford, "Eliminating the Hypervisor Attack Surface for a More Secure Cloud", Procs 18th ACM Conference on Computer and Communications Security (CCS '11), Chicago, IL, pp. 401-412, 2011.
- [16] V. Varadarajan, T. Kooburat, B. Farley, T. Ristenpart and M.M. Swift, "Resource-freeing Attacks: Improve Your Cloud Performance (at Your Neighbor's Expense)", Procs ACM Conference on Computer and Communications Security (CCS'12), Raleigh, NC, pp. 281-292, 2012.
- [17] W. Wongthai, F. Rocha and A. van Moorsel, "Logging Solutions to Mitigate Risks Associated with Threats in Infrastructure as a Service Cloud", Procs International Conference on Cloud Computing and Big Data (CloudCom-Asia), pp. 163-170, 2013.
- [18] Y. Xu, M. Bailey, F. Jahanian, K. Joshi, M. Hiltunen and R. Schlichting, "An Exploration of L2 Cache Covert Channels in Virtualized Environments", Procs 3rd ACM Workshop on Cloud Computing Security (CCSW'11), Chicago, IL, pp. 29-40, 2011.
- [19] Y. Zhang, A. Juels, A. Oprea and M.K. Reiter, "HomeAlone: Co-residency Detection in the Cloud via Side-Channel Analysis", Procs IEEE Symposium on Security and Privacy, pp. 313-328, 2011.

Biographies

Paul Devadoss Ezhilchelvan received Ph.D. degree in computer science in 1989 from the University of Newcastle, U.K. He joined the School of Computing Science in 1983 where he is currently a Reader. His research interests are in Distributed algorithms, Consensus and multicast protocols, wireless ad-hoc networks, performance evaluation and stochastic modelling.

Isi Mitrani is Emeritus Professor at the University of Newcastle, where he has spent most of his working career. He studied Mathematics at the Universities of Sofia and Moscow, and Operations Research at the Technion, Haifa. His research interests are in the areas of probabilistic modelling, performance evaluation and optimization.