Incentive Compatible Moving Target Defense against VM-Colocation Attacks in Clouds

Yulong Zhang[1], Min Li[1], Kun Bai[2], Meng Yu[1], and Wanyu Zang[1]

[1] Computer Science Department, Virginia Commonwealth University
401 W. Main Street, Richmond, VA 23284, USA
{zhangy44, lim4, myu, wzang}@vcu.edu
[2] IBM T.J. Watson Research Center
19 Skyline Dr., Hawthorne, NY 10532, USA
kunbai@us.ibm.com

Abstract. Cloud computing has changed how services are provided and supported through the computing infrastructure. However, recent work [11] reveals that virtual machine (VM) colocation based side-channel attack can leak users' privacy. Techniques have been developed against side-channel attacks. Some of them like NoHype remove the hypervisor layer, which suggests radically changes of the current cloud architecture. While some other techniques may require new processor design that is not immediately available to the cloud providers.

In this paper, we propose to construct an incentive-compatible moving-target-defense by periodically migrating VMs, making it much harder for adversaries to locate the target VMs. We developed theories about whether the migration of VMs is worthy and how the optimal migration interval can be determined. To the best of our knowledge, our work is the first effort to develop a formal and quantified model to guide the migration strategy of clouds to improve security. Our analysis shows that our placement based defense can significantly improve the security level of the cloud with acceptable costs.

Keywords: Cloud computing, Moving target defence, VM migration, VCG mechanism

1 Introduction

Cloud computing [3] is becoming a major trend in computing services with its inspiring features of elastic "data anywhere" and "computing anywhere". Generally, there are three types of cloud computing services: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS). Among them the IaaS is the most fundamental one, where a cloud user owns a virtual machine (VM) and purchases virtual power to execute as needed, just like running a virtual server. A typical example of public IaaS is the Amazon Elastic Computing (EC2) Services [1]. Although IaaS offers cost-efficiency and ease-of-use to cloud users, there are significant security concerns that need to be addressed when considering moving critical applications and sensitive data to the clouds. Recent work [11] reveals the problem of side-channel based attacks through virtual machine colocation, showing that the adversaries can map the internal VM-placement of the cloud and mount cross-VM side-channel attacks by placing malicious VMs on the victim's physical server.

Many software level approaches [10] and hardware modification methods [13, 7] have been developed against the side-channel attacks. Unfortunately, the software approach cannot perfectly cover the new advances in attack models, and the hardware approach, modifying the cloud physical platform, is far from practical for commercial clouds. Using commercial off-the-shelf (COTS) components, the *Shamir's secret sharing* approach [12] can make the attack much harder. We can split our secret D into k pieces and store them into k VMs [3]. Using Shamir's secret sharing, D can be easily constructed from all k pieces but knowledge less than k pieces reveals no information about D. Note that the secret sharing might be controlled by either a single party or multiple parties [2]. Thus, the attacker will be forced to use *brute-force-attacks* to achieve colocation with multiple target VMs, according to [11]. Now, the defense problems becomes how to protect the k pieces from being all captured by the attacker.

The methods to securely live-migrate VMs [4, 8] can make it much harder for adversaries to locate the target VMs [6]. As shown in Figure 1, the secret, "helloworld", is divided into two pieces and held by two VMs, initially as VM_0 and VM_2. An attacker can launch VMs colocated with the benign VMs with some probabilities. For example, if VM_3 is controlled by an attacker, the attacker will be able to extract partial secret, "hello", out of VM_0. Although the splitting of "helloworld" into "hello" and "world" can prevent the attack from reconstructing the secret from VM_3, we should at the same time guarantee that VM_5 can never be taken over by the same attacker. The fact that the attack would simultaneously attempt to launch lots of VMs to enumerate the colocations with both VM_0 and VM_2 motivates the migration of the VMs, e.g. moving VM_0 to VM_1. Unfortunately, there is currently no formal model to guide the dynamic migration strategy for clouds. Similarly, quantifying the benefits of dynamism still remains an open problem.

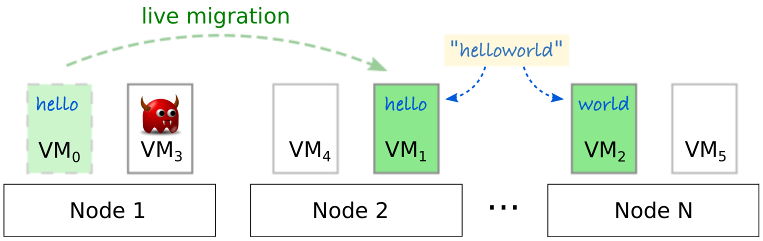


Fig. 1: Shamir's secret sharing in cloud environment and the moving target defense through VM-migration.

Intuitively, the cloud provider can offload the choice to cloud users, letting them migrating at free will. Or the cloud provider can migrate VMs according to the security demand - those VMs with higher security demand of dynamism have higher chances to be migrated.

[3] It is not necessary to have k identical VMs. We can have one service VM and $k-1$ secret holding VMs that can be very small and cost little.

At first glance, this is reasonable. However, as long as some of the VMs sharing the secret are migrated, other VMs can take the free ride of the achieved dynamism. Since any rational cloud user will try to maximize their benefit[4] , he/she could report a lower preference of dynamism than the actual one, pining its hope on other VMs honest movements to reach the security goal. So in this case, the game equilibrium is that no one would truthfully report the security valuation and migrate.

Our goal and contributions: To solve the aforementioned problems, the goals of this paper are (1) to model the migration benefit and cost, (2) to provide instructional criteria for the migration strategy of the cloud provider, and (3) to motivate the cloud users to migrate in a incentive-compatible way. In order to address these issues, we develop a migration strategy based on the Vickrey-Clarke-Groves(VCG) mechanism[9] in game theories, which can maximize the social welfare given the individuals are all "selfish".

The contributions of this paper are as follows. To the best of our knowledge, (1) this is the first work to address VM-colocation based attacks in clouds using moving target defense; (2) our work is the first effort to apply VCG game and realize incentive compatible migration in cloud; (3) we offer two criteria about how to make the strategy acceptable by rational cloud users, and how to determine the optimal time interval of migrations, (4) our analytical evaluation shows that our defense approach is practical for commercial clouds.

This paper is organized as follows. In Section 2, we review some of the fundamental concepts of game theory, especially the VCG mechanism. In Section 3, we present our proposed method. In Section 4, we evaluate the security and practicability of our scheme. We conclude the work in Section 5.

2 Preliminaries

2.1 Assumptions and Notations

We have the following assumptions: (1) the cloud controller and the migration process are all secure; (2) for simplicity, each migration of a VM will result in a constant cost in terms of service interruption and consume the same amount of resources; (3) the cloud provider has enough CPU, network bandwidth, and other resources to perform arbitrary migration; (4) the cloud provider has sufficient resources as the reward, e.g., extra memory or CPUs, to motivate the players to migrate, which will be further discussed later; and (5) a node's destination of migration is randomly chosen, as long as the cloud has free space. In reality, cloud provider would take performance and efficacy into consideration during the migration. For example, VMs with frequent connections should be placed close to each other. In this paper, for simplicity, we only measure security in the goal function, but a commercial cloud can freely modify the goal function as needed, where our game model remains the same.

The incentive problem of cloud migration can be modelled as a game. The specific game discussed here is a finite player, single round, simultaneous action, incomplete information game, with payoffs depending on the final result of players' actions. In the cloud migration

[4] Note that even if all the VMs sharing the secret are controlled by one user, because different VMs may have different purposes, each VM should be treated as an independent party with individual interest.

problem, the game players, $P_1, P_2, ..., P_n$, are n VMs sharing a secret. The cloud provider (game mediator) is denoted as P_t, who doesn't participate in the game but operates the game. Players have actions which they can perform at designated times in the game, and as a result they receive payoffs. The actions of the players are denoted as $X = (x_1, x_2, ..., x_n)$, and specially $X_{-i} = (x_1, x_2, ..., x_{i-1}, x_{i+1}, ..., x_n)$. The players have different pieces of information, on which the payoffs may depend; each player has his/her individual strategy to maximize his or her payoff. Each player $P_i \in \{P_1, P_2, ..., P_n\}$ can decide whether to accept the migration request from P_t or not, so $x_i = \{accept, refuse\}$. For those who agree to migrate, the direct payment is $\bar{p_i}$, which measures the cost of downtime due to live migration. Here we assume the cost of downtime is a constant value for each P_i. As a result of migration, all the players get benefited. We measure the benefit as $v_i(X)$, where the valuation function v_i quantifies P_i's real security demand of the overall dynamism. However, under the assumption of incomplete information, P_i can decide to report a fake valuation function v'_i at will. Finally, the utility function of each player is $u_i = v_i - p_i$.

2.2 Incentive Compatible Game

While our ultimate goal is to design a migration strategy to fulfil some security requirements, the problem is that in real-world cloud environment most of the cloud users are not willing to migrate their VMs unless it is necessary, because the migration will result in service interruption. So the main purpose of our work is to design an incentive compatible mechanism to mitigate this problem. Following Nisan's work [9], the terms "mechanism" and "incentive compatible" are defined as :

Definition 1 *(Mechanism) [9] Given a set of n players, and a set of outcomes, A, let V_i be the set of possible valuation functions of the form $v_i(a)$ which player i could have for an outcome $a \in A$. A mechanism is a function $f: V_1 \times V_2 \times ... \times V_n \to A$. Given the valuations claimed by the players, f selects an outcome, and n payment functions, $p_1, p_2, ..., p_n$, where $p_i : V_1 \times V_2 \times ... \times V_n \to R$.*

Definition 2 *(Incentive compatible) [9] If, for every player i, every $v_1 \in V_1, v_2 \in V_2, ..., v_n \in V_n$, and every $v'_i \in V_i$, where $a = f(v_i, v_{-i})$ and $a' = f(v'_i, v_{-i})$, then $v_i(a) - p_i(v_i, v_{-i}) \geq v_i(a') - p_i(v'_i, v_{-i})$, then the mechanism is incentive compatible.*

Specially, among those incentive compatible mechanisms, the Vickrey-Clarke-Groves (VCG) mechanism is the mostly used one. The VCG mechanism generally seeks to maximize the social welfare of all players in one game, where the social welfare is calculated as $\sum_{i=1}^{n} v_i$. So the goal function of VCG is $argmax_{a \in A} \sum_{i=1}^{n} v_i$. The VCG mechanism and the rule to design VCG mechanisms [9] are defined as:

Definition 3 *(Vickrey-Clarke-Groves mechanism) [9] A mechanism, consisting of payment functions $p_1, p_2, ..., p_n$ and a function f, for a game with outcome set A, is a Vickrey-Clarke-Groves mechanism iff $f(v_1, v_2, ..., v_n) = argmax_{a \in A} \sum v_i(a)$ (f maximizes the social welfare) and for some functions $h_1, h_2, ..., h_n$, where $h_i : V_{-i} \to R$ (h_i does not depend on v_i), $\forall v_i \in V, p_i(v_i) = h(v_{-i}) - \sum_{j \neq i} v_j$.*

Definition 4 *(Clarke pivot rule) [9] The choice $h_i(v_{-i}) = max_{b \in A} \sum_{j \neq i} v_i(b)$ is called the Clarke pivot payment. Under this rule the payment of player i is $p_i(v_1, v_2, ..., v_n) = max_b \sum_{j \neq i} v_i(b) - \sum_{j \neq i} v_i(a)$, where $a = f(v_1, v_2, ..., v_n)$.*

3 VM-migration Based Moving Target Defence

3.1 The Optimal Number of Moving VMs

At first glance, it appears that maximum dynamism can be achieved if all VMs migrate, but in this section we will disprove it and find the optimal number of moving VMs. As denoted in Section 2, each player has the unique valuation function $v_i(X)$, where $X = (x_1, x_2, ..., x_n)$ is the vector of actions of all players. Intuitively, v_i has a positive correlation with the randomness or diversity of the VM placement. Suppose there are m physical nodes as the available candidates, and $\gamma$of them are randomly chosen as the migration destination, of which the capacities (the maximum number of extra VMs one node can host using its current free space) are $C = c_1, c_2, ..., c_\gamma$. If k out of n VMs are randomly selected to be migrated ($k < \sum_{i=1}^{\gamma} c_i$), the number of possible placements, $N(m, \gamma, n, k, C)$ can be derived using the *Balls In Bins* analysis.

For selecting γ ones out of the m nodes, and selecting k ones out of the n VMs, the numbers of possible schemes are given respectively:

$$S(m, \gamma) = \binom{m}{\gamma}, T(n, k) = \binom{n}{k} \tag{1}$$

The generating function for placing k distinguishable VMs (balls) in γ distinguishable nodes (bins) is

$$GF = \prod_{i=1}^{\gamma} \sum_{j=0}^{c_i} \frac{x^j}{j!} \tag{2}$$

where the coefficient of $\frac{x^k}{k!}$, denoted as $\theta(\gamma, k, C)$, is the number of the total number of possible placements.

Specially, if $c_1 = c_2 = ... = c_\gamma = \bar{c}$, Equation 2 reduces to be

$$GF = (1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + ... + \frac{x^{\bar{c}}}{\bar{c}!})^\gamma \tag{3}$$

Finally, the total number possible VM-placements is

$$N(m, \gamma, n, k, C) = S(m, \gamma)T(n, k)\theta(\gamma, k, C) \tag{4}$$

Given the m, γ, n and C, we can use Equation 4 to find k_{opt}, which is the optimal number of moving VMs corresponding to the largest N. To make it tangible, an example curve illustrating the $k - N$ relationship is shown in Figure 2. Since $S(m, \gamma)$ does not contribute too much in the magnitude of N, we simply set $\gamma = m$ (so $S(m, \gamma) = 1$); for demonstration, we set the total VMs as $n = 25$ and the capacity of each destination node as $\bar{c} = 4$. From the curve we can learn that:

1. It is not true that a larger k is better, which means that the intuitive strategy of migrating VMs as many as possible is incorrect.
2. A larger γ, namely more destination nodes, significantly increases the number of possible placement schemes.

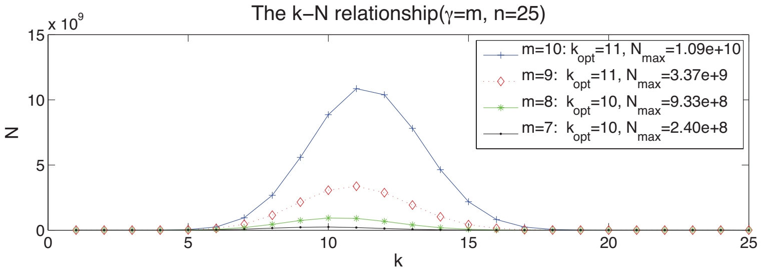


Fig. 2: The k-N relationship when n=25, $\bar{c} = 4$, and m=γ varies from 10 to 7.

3. Compared to γ, N is even more sensitive to k. N can reach the order of 10^9 when k is still in the order of 10^1, which implies the huge potential of VM migration as a moving target defense strategy.

On the one hand, the optimal k is given by

$$k_{opt} = argmax_k N(m, \gamma, n, k, C) \tag{5}$$

and in reality, only cloud provider controls the complete information of m, γ, n, k, C, so it is the cloud provider's responsibility to calculate the k_{opt}.

On the other hand, if we assign 1 to *accept* and 0 to *refuse*, we have the actual number of moving VMs, k_{real}, as:

$$k_{real} = \sum_{i=1}^{n} x_i \tag{6}$$

In economics, the valuation function should reflect the security preference of the cloud users. Since this is not the main focus of this paper, we simply model the preference as a linear function of the system diversity:

$$v_i(X)|_{m,\gamma,n,C} = \lambda_i N(m, \gamma, n, \sum_{i=1}^{n} x_i, C) + \eta_i \tag{7}$$

where λ_i and η_i are determined by the ith cloud user's preference.

Once the cloud provider decides the security level, and thus the value of k, the rest is to motivate the randomly chosen k VMs, denoted as Ω, to get migrated. To satisfy the Clarke pivot rule defined in Definition 4, we design the payment function for each $P_i \in \Omega$ as:

$$p_i(X) = \sum_{j \neq i} v_j(X_{-i}) - \sum_{j \neq i} v_j(X) + \bar{p}_i \tag{8}$$

where $\bar{p}_i$ is the constant cost due to service interruption as assumed. We will further introduce the determination of $\bar{p}_i$ in Section 3.2.

The second part of Equation 8, $-\sum_{j \neq i} v_j(X)$, is actually a positive reward from cloud provider to the ith VM. It can be any form with monetary value equalling to it, like extra CPU scheduling credit, bonus VM life, extra network bandwidth and so on, depending on the need of P_i. The first part, $\sum_{j \neq i} v_j(X_{-i})$, is the monetary charge of P_i. If P_i chooses *accept*, $\bar{p}_i$ is the actual migration cost; if P_i chooses *refuse*, $\bar{p}_i$ is in the form of monetary charge. Given that everyone else accepts migration, if P_i decides to migrate, we have $k_{real} = k_{opt}$; if P_i refuses to do so, $k_{real} = k_{opt} - 1$.

Theorem 1 *(The criterion for the migration decision) The mechanism with the valuation function shown in Equation 7 and the payment function shown in Equation 8 is individually rational if and only if*

$$\sum_{i=1}^{n} \lambda_i (N(m, \gamma, n, k_{opt}, C) - N(m, \gamma, n, k_{opt} - 1, C)) > \bar{p}_i$$

Proof. To prove that the mechanism is individually rational, we should show that the mechanism has a utility of at least zero for each of the players. For those selected to be migrated:

$$\begin{aligned} u_i(X) = v_i(X) - p_i(X) &= \sum_{i=1}^{n} v_i(X) - \sum_{j \neq i}^{n} v_j(X_{-i}) - \bar{p}_i \\ &= \sum_{i=1}^{n} \lambda_i (N(m, \gamma, n, k_{opt}, C) - N(m, \gamma, n, k_{opt} - 1, C)) - \bar{p}_i \end{aligned} \tag{9}$$

Only when the value of the above equation is larger than 0, P_i can achieve individual rationality. As a matter of fact, Theorem 1 provides a criterion for cloud provider to decide whether a migration is cost-efficient.

Theorem 2 *(Incentive Compatibility) The mechanism with the above valuation and payment functions is incentive compatible and thus motivates the players to truthfully reveal their security preference.*

Proof. To get to the proof of incentive compatibility, we must show that for any given i, X_{-i} and the P_i's choice $x_i = accept$ or $x_i = refuse$:

$$u_i(X = accept \cup X_{-i}) \geq u_i(X' = refuse \cup X_{-i}) \tag{10}$$

The utility of P_i for X is given by

$$u_i(X) = v_i(X) - p_i(X) = \sum_{i=1}^{n} v_i(X) - \sum_{j \neq i}^{n} v_j(X_{-i}) - \bar{p}_i \tag{11}$$

Likewise,

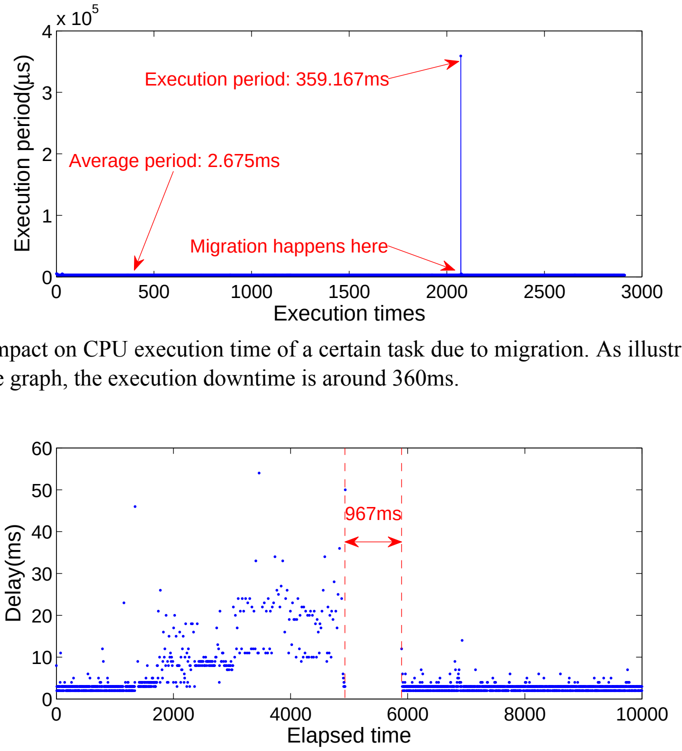
$$u_i(X') = \sum_{i=1}^{n} v_i(X') - \sum_{j \neq i}^{n} v_j(X_{-i}) - \bar{p}_i \tag{12}$$

So we have

$$\begin{aligned} u_i(X) - u_i(X') &= \sum_{i=1}^{n} v_i(X) - \sum_{i=1}^{n} v_i(X') \\ &= \sum_{i=1}^{n} \lambda_i(N(m, \gamma, n, k_{opt}, C) - N(m, \gamma, n, k_{opt} - 1, C)) \end{aligned} \tag{13}$$

According to Equation 5, the value of Equation 13 is always non-negative. Therefore, the mechanism is incentive compatible.

3.2 The Constant Cost of Migration



(a) Impact on CPU execution time of a certain task due to migration. As illustrated in the graph, the execution downtime is around 360ms.

(b) Migration impact on response delay of web server. As illustrated in the graph, the web service downtime due to migration is 967ms.

Fig. 3: The influence of migration to VM computation and web service.

In Equation 8 we introduced the $\bar{p}_i$, which is the constant cost of service interruption each time P_i is migrated. According to [8], $\bar{p}_i$ is determined by the initial memory size of a VM and the applications' memory access pattern. For demonstration, we illustrate the downtime due to migration in Figure 3. The platform specifications are listed in Table 1, and the VM that we migrate is of 1 vcpu, 1 GB memory and 4 GB disk. To test the influence to the cpu execution, we keep the VM executing "`for(i=0;i<999999;i++); gettimeofday(&time,NULL);`" and recording the execution period. As shown in 3a, the migration will bring in an execution delay of around 350ms. To test the influence to the network service, we keep measuring the VM's response delay of `GET` requests. As shown in 3b, there will be a 967ms downtime to the web server. P_t and each P_i can estimate the $\bar{p}_i$ respectively, according to the service downtime.

Table 1: Platform specifications of migration downtime experiment.

Hardware (source and destination nodes):	**VMM:**
CPU: Intel Xeon x5650 2.66GHz × 2	Xen version: 4.0.1-21326-02-0.5
RAM: 8GB DDR3 1333MHz × 3	Dom0: openSUSE 11.3 x86_64
Ethernet: Broadcom NetXtreme II BCM5709	Dom0 kernel: 2.6.34.7-0.7-xen

3.3 Defense Timeline

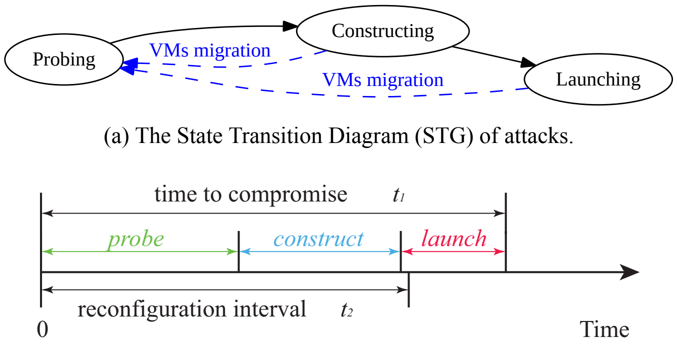
Although the migration can confuse the attackers and force them to restart the attack procedures, with time going by the attackers can still figure out the new VM placement as well. To solve the problem, we have to keep migrating the VMs, and how to determine the optimal time interval becomes a new problem. Following the model in [5], the State Transition Diagram (STG) of attacks is shown in Figure 4a. More specifically, the attackers have to firstly probe the placement of the VMs, for example, by continually creating and stopping their probing VMs and testing the colocation with target VMs; then, it takes some time to construct the attacks; and there is still a period before the attackers successfully gathering all the information. Any distortion of the VM placement, for example, by migrating any of the VMs to other places, will reset the attack to the initial state.

As shown in Figure 4b, suppose the total time needed for a successful attack is t_1, the time interval between any two migrations t_2 should satisfy $t_2 < t_1$, which offers the cloud provider the criterion to determine migration intervals. If $t_2 \geq t_1$, there is a highly chance for the attackers to fully extract the secret. In reality the constructing time and the launching time of the attacks could be treated as constants, while the probing time has a linear relationship with the N, which explodes with the increasing of γ as shown in Figure 5d.

4 Evaluation

4.1 Security

We can never guarantee that our defense strategy or algorithm is agnostic to the adversaries. So when we design a moving target defense scheme, we must ensure that the adversaries are impossible or difficult to recover our actual internal infrastructures even if the strategy



(a) The State Transition Diagram (STG) of attacks.

(b) Attack and defense lifetime.

Fig. 4: The timeline of attacks and moving target defense (VM migration).

is not secret. In the mechanism designed here, we assume that the forms of the valuation function and the deliberate payment function are known to the public. However, in reality the individual security preference is kept as a privacy of each cloud user, thus the adversaries have no way to get λ_i and η_i. Even if some of these parameter-pairs are exposed, as long as the adversaries cannot obtain the information of all the VMs in the cloud, our strategy based on the overall social welfare is always secure. Furthermore, during the generation of the migration scheme, γ are randomly chosen and the capacity $C = \{c_1, c_2, ..., c_n\}$ is a secret of cloud provider only, thus according to Equation 5 the final migration scheme is only known to cloud provider.

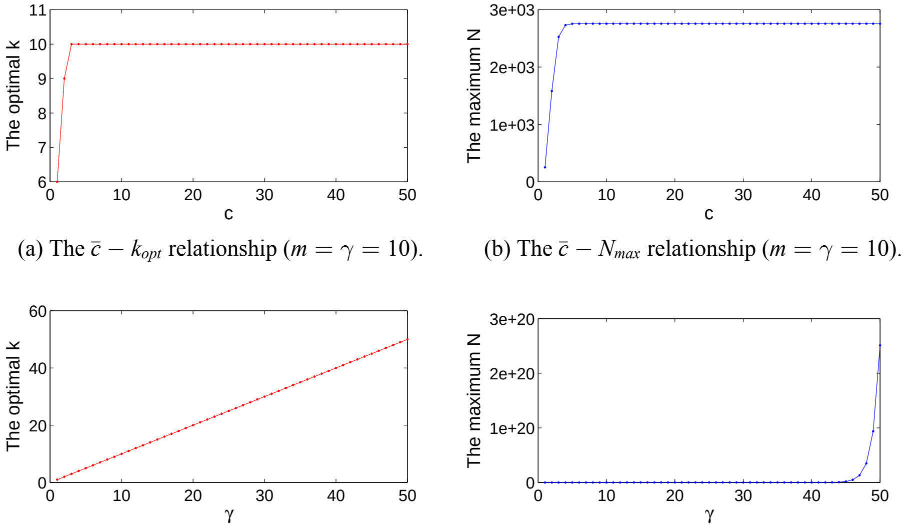
Another indicator of one moving target defense scheme's capability is the number of the total target space. As demonstrated later in Figure 5d, N can increase rapidly with a small increase of γ. With the exploding number of possible VM placements, the adversaries are extremely difficult to enumerate all the possibilities.

We note that there might be information leakage during VM migration. But since the design of migration methods is not within the scope of this paper, we prospect more secured VM migration mechanisms.

4.2 Practicability

As shown in Figure 5, compared with $\bar{c}$, γ contributes more on the value of both k_{opt} and N_{max}. Empirically, $k_{opt} \approx \gamma$; theoretically, for any $c_1, c_2, x \in \mathbb{N}\ and\ c_1 < c_2$ we have $\frac{x^{c_1}}{c_1!} > \frac{x^{c_2}}{c_2!}$ and $lim_{c \to \infty} \frac{x^c}{c!} = 0$, so according to Equation 3, γ has a significantly larger influence on the value of k_{opt}. Consequently, for the real-world cloud providers, it is pretty easy to determine the optimal strategy by assigning $k_{opt} = \gamma$ without complex algorithms, which maks this approach scalable.

Another potential concern is that whether the cloud users are willing to split a secret into multiple VMs, which is the precondition of our VM migration based defense. As shown in Table 2, the small instance has a price of $1/4$ of the price of the large one, and offers $1/4$ of the compute units, $1/4.4$ of the memory and $1/5.3$ of the storage. Therefore splitting a secret from one large VM into four samll VMs won't largely increase the cost. Compare with the risk of being attacked through covert channels, the cloud users will be willing to endure the cost due to transferring computation from few large VMs to numerous small VMs.



(a) The $\bar{c} - k_{opt}$ relationship ($m = \gamma = 10$).

(b) The $\bar{c} - N_{max}$ relationship ($m = \gamma = 10$).

(c) The $\gamma - k_{opt}$ relationship ($\bar{c} = 4$).

(d) The $\gamma - N_{max}$ relationship ($\bar{c} = 4$).

Fig. 5: The influence to k_{opt} and corresponding N_{max} by adjusting $\bar{c}$ or γ in Equation 5.

Table 2: Amazon EC2 Linux/UNIX instances pricing and specifications (US East, up to January 2012) [1] .

Instance types	Price	Memory	Compute Units	Storage	API name
Small	\$0.085 per hour	1.7 GB	1	160 GB	m1.small
Large	\$0.34 per hour	7.5 GB	4	850 GB	m1.large

5 Discussion and Conclusion

Our work can be extended in different avenues. First, only security goal is considered here for simplicity, but the valuation and payment functions can be easily adjusted to include the performance considerations. Second, so far we have only considered the non-cooperative game. Actually, there might be some connections between VMs on cloud, forming a cooperative game which is a competition between coalitions of players rather than between individual players. In the future, we will develop the mechanism in regard to the coordination game. Furthermore, in the next step, we will extend the mechanism to suite into the asymmetric information game, because in reality some VMs are offering services to other VMs (thus knowing some statistics of the latter), and some VMs may even be controlled by attackers with some knowledge of other VMs.

To summarize, we proposed an incentive compatible moving target defense of cloud VM-colocation attacks, based on the VCG mechanism. When the migration is acceptable by rational users and how to determine the time interval are discussed. Our analysis shows that this defense strategy is practical to be applied to commercial clouds.

Acknowledgments. This work was supported in part by NSF Grants CNS-1100221 and CNS-0905153.

References

1. Amazon web services. `http://aws.amazon.com`
2. Workshop on Secret Sharing and Cloud Computing. MEXT, Institute of Mathematics for Industry, Kyushu University, Japan (2011)
3. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R.H., Konwinski, A., Lee, G., Patterson, D.A., Rabkin, A., Stoica, I., Zaharia, M.: Above the clouds: A berkeley view of cloud computing. Tech. Rep. UCB/EECS-2009-28, EECS Department, University of California, Berkeley (Feb 2009)
4. Danev, B., Masti, R.J., Karame, G.O., Capkun, S.: Enabling secure vm-vtpm migration in private clouds. In: Proceedings of the 27th Annual Computer Security Applications Conference. pp. 187–196. ACSAC '11, ACM, New York, NY, USA (2011)
5. Evans, D., Nguyen-Tuong, A., Knight, J.: Effectiveness of moving target defenses. In: Jajodia, S., Ghosh, A.K., Swarup, V., Wang, C., Wang, X.S. (eds.) Moving Target Defense, Advances in Information Security, vol. 54, pp. 29–48. Springer New York (2011)
6. Kant, K.: Configuration management security in data center environments. In: Jajodia, S., Ghosh, A.K., Swarup, V., Wang, C., Wang, X.S. (eds.) Moving Target Defense, Advances in Information Security, vol. 54, pp. 161–181. Springer New York (2011)
7. Keller, E., Szefer, J., Rexford, J., Lee, R.B.: Nohype: virtualized cloud infrastructure without the virtualization. SIGARCH Comput. Archit. News 38, 350–361 (June 2010)
8. Liu, H., Xu, C.Z., Jin, H., Gong, J., Liao, X.: Performance and energy modeling for live migration of virtual machines. In: Proceedings of the 20th international symposium on High performance distributed computing. pp. 171–182. HPDC '11, ACM, New York, NY, USA (2011)
9. Nisan, N., Roughgarden, T., Tardos, E., Vazirani, V.: Introduction to Mechanism Design (for Computer Scientists). Cambridge University Press (2007)
10. Page, D.: Defending against cache-based side-channel attacks. Information Security Technical Report 8(1), 30 – 44 (2003)
11. Ristenpart, T., Tromer, E., Shacham, H., Savage, S.: Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In: Proceedings of the 16th ACM conference on Computer and communications security. pp. 199–212. CCS '09, ACM, New York, NY, USA (2009)
12. Shamir, A.: How to share a secret. Commun. ACM 22, 612–613 (November 1979)
13. Wang, Z., Lee, R.B.: New cache designs for thwarting software cache-based side channel attacks. SIGARCH Comput. Archit. News 35, 494–505 (Jun 2007)