# Improving Cloud Survivability through Dependency based Virtual Machine Placement

Min Li[1], Yulong Zhang[1], Kun Bai[2], Wanyu Zang[1], Meng Yu[1], and Xubin He[3]
[1]*Computer Science, Virginia Commonwealth University, USA*
[2]*IBM T.J. Watson Research Center, USA*
[3]*Electrical and Computer Engineering, Virginia Commonwealth University, USA*
*{lim4, zhangy44, wzang, myu, xhe2}@vcu.edu, kunbai@us.ibm.com*

Keywords: Cloud computing: virtual machine placement: security: survivability

Abstract: Cloud computing is becoming more and more popular in computing infrastructure and it also introduces new security problems. For example, a physical server shared by many virtual machines can be taken over by an attacker if the virtual machine monitor is compromised through one of the virtual machines. Thus, collocating with vulnerable virtual machines, or "bad neighbours", on the same physical server introduces additional security risks. Moreover, the connections between virtual machines, such as the network connection between a web server and its back end database server, are natural paths of attacks. Therefore, both virtual machine placement and connections among virtual machines in the cloud have great impact over the overall security of cloud. In this paper, we quantify the security risks of cloud environments based on virtual machine vulnerabilities and placement schemes. Based on our security evaluation, we develop techniques to generate virtual machine placement that can minimize the security risks considering the connections among virtual machines. According to the experimental results, our approach can greatly improve the survivability of most virtual machines and the whole cloud. The computing costs and deployment costs of our techniques are also practical.

## 1 INTRODUCTION

Cloud computing is becoming predominant in computing infrastructure since it provides the flexibility and cost-effectiveness hardly found in traditional computing platforms. The key technique of cloud computing is resource sharing and dynamic resource allocation of the cloud. In an Infrastructure as a Service (IaaS) cloud like Amazon EC2, multiple virtual machines (VMs) share a physical server. Thus, the security of VMs is dependent not only on how secure the Operating System and applications they are running, but also the security of Virtual Machine Monitor (VMM, or hypervisor), running below the VMs.

There are many attacks developed against cloud environments. In this paper, we are interested in two types of attacks since they are related to how VMs are placed in a cloud. **Type I** attacks, such as (CVE-2007-4993, 2007; CVE-2007-5497, 2007), exploit the vulnerabilities of hypervisors, eg., Xen and KVM. Once succeed, attackers can compromise the physical server running the hypervisor and all VMs running above the hypervisor. Alternatively, **Type II** attacks compromise other VMs on the same physical server through mounting side channel attacks (Ristenpart et al., 2009; Hlavacs et al., 2011), instead of compromising the hypervisor.

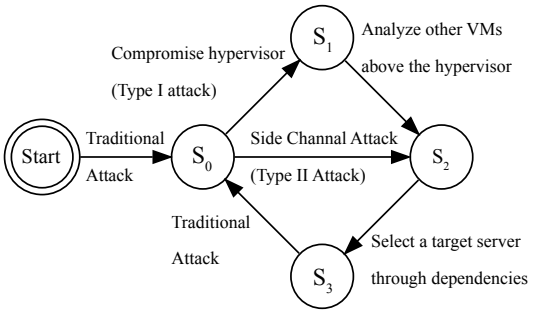



Figure 1: The State Transition Graph of Attacks. $S_0$: One VM compromised on a new server. $S_1$: Hypervisor compromised. $S_2$: Dependency information collected. $S_3$: New target server selected.

As shown in Figure 1, attackers can utilize both types of attacks to compromise as many VMs as possible in the cloud. In the first step, the attacker can compromise one guest VM (Dom U) or the management VM (Dom 0) through the vulnerabilities in operating system or services.Consequently, the attacker can

launch side channel attacks to collect information of other VMs on the same physical server. The information can also be collected using type I attacks, as shown in Figure 1. Because network connections, e.g., from a web server to a back end database server, may leak security information such as authentication information on the database server, the attacks can cause cascading effects, or *domino effects* in the cloud. Also, the connections among VMs become paths of attacks,

For example, the attack "Hey, you get out of my cloud" (HYG) (Ristenpart et al., 2009) is one kind of type II attack. The initial stage of the HYG attack is to locate the target VM. Once success is achieved, the attacker will try to launch a VM on the same physical server. It is a placement based attack and the success of the attack depends on the placement strategies of the cloud, or the configuration policy of the cloud.

In this paper, we present an approach of virtual machine placement based on the security evaluation of VMs and the dependencies among them. Our technique not only increases the survivability of cloud but also is compatible with performance requirements. In security evaluation, we use Discrete Time Markov Chain (DTMC) to analyze the possibility of being compromised for each VM. In performance evaluation, we calculate both migration cost and the distance of placement. Based on performance and security analysis, we generate a new VM placement.

To the best of our knowledge, this paper is the first effort to develop the following mechanisms and techniques to enhance cloud security through changing cloud placement. The contributions of this paper are summarised as below:

- We propose a systematic approach to evaluating security risks of a cloud placement plan.
- Based on our security evaluation, we propose an innovative and practical approach to generate a safe placement plan.
- Our placement generation is compatible with performance requirements since it is also based on dependencies among VMs.
- The experimental results confirm that our placement plan can significantly improve the survivability of VMs in cloud.

In section 2, we review related work about VM placement and indicate our unique contributions. Section 3 defines the attack model and describes our goals of placement. In Section 4, we explain the architecture of our implementation, dependency based VM placement strategy, and other implementation details. In Section 5, we show how well our technique perform over real data.

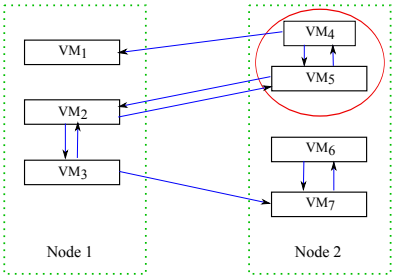


Figure 2: Cloud Placement Example.

## 2 RELATED WORK

A good VM placement plan can immensely improve the performance. For example, (Sindelar et al., 2011) demonstrated that memory share based placement can save the memory resources of a server. In their proposed placement scheme, VMs with the most shared pages are collocated in the same physical server.

To improve efficiency, (Yusoh and Tang, 2010) developed a generic algorithm to create a placement plan to reduce Estimated Total Execution Time (ETET). Work (Lucas Simarro et al., 2011) provided a scheduling model to optimize virtual cluster placement through cloud offers. The cloud prices and user demand have been considered in the model. The experimental results on the real data show that dynamic placement plan can bring more benefits on reducing users' costs than the fixed one.

Unfortunately, none of the above work considered the security issue. Our previous work (Zhang et al., 2012) proposed to periodically migrating VMs based on game theory, making it much harder for adversaries to locate the target VMs in terms of survivability measurement. However, our previous work did not discuss how to evaluate the security of a cloud placement and how to generate a placement plan to improve the cloud security. This paper proposes an innovative and effective placement strategy based on dependency relations among VMs.

## 3 SYSTEM OVERVIEW

In this section, we describe our basic assumptions and the goals of VM placement.

### 3.1 Characteristics

An example of virtual machine placement is shown in Figure 2. The most important component is Virtual Machine and Node. Each virtual machine runs different services and some of the VMs are dependent on others.

Node represents a physical machine which runs a few to many VMs, given the limit of hardware resource. In Figure 2, Node 1 has three VMs while Node 2 holds four VMs. Besides, Cloud Provider has necessary privileges to scan vulnerabilities of VMs and obtain information of network connections among VMs.

In addition, We assume the following about an attacker.

1. The attacker can exploit the vulnerability of a hypervisor or a VM.
2. The attacker follows the state transition graph in Figure 1 to compromise VMs step by step.
3. The attacker always chooses the easiest target to compromise in each step, in terms of the vulnerabilities in VMs and the attacker's skills.
4. The attacker has no global view of the cloud at the beginning of the attacks. However, the attacker may acquire more knowledge after compromising more nodes in the cloud.

Since the success of attacks highly depends on the placement strategy of cloud. Our purpose is to present systematic solution which reduces the security risks for both Type I and Type II attacks while not sacrificing performance.

We can defeat Type I attacks because our mechanism will change the VM placement strategy after a specific time. Hence, if an adversary plan to compromise a specified VM through compromising hypervisor and the whole node. The VM can survive if it can be migrated to other node before an attacker compromise the node. In addition, we can also resist Type II attacks because we try to assign dependent VMs in the same node. In this situation, it is difficult for the adversary to compromise other nodes. Hence, we increase the survivability of VMs on other node. In this work, we try to both **reduce the number of compromised VMs** and **increase the survivability of services**.

To verify the improvement of survivability of service, we define the survivability of service. Service is accomplished by a set of *connected VMs*, which are defined as VMs have data transmission.Next, we provide the definition of ***compromised possibility of VM*** that the possibility of being compromised for a given VM at specific associate attack step. According to the definition, we can give a theorem about how to evaluate the survivability of service.

**Theorem 1 (Survivability of a Service).** *Given a service $S_i$ (VM chain) including some related $S_i = \{VM_a, VM_b, ..., VM_n\}$ and node set $N = \{N_1, N_2, ..., N_m\}$, If the survivability in specific attack step for the Nodes which hold the VM belong to $S_i$ is $\{PN_1, PN_2, ..., PN_m\}$, Then survivability (PS) for service $S_i$ is below:*

$$PS_i = \prod_{j=1}^{m} PN_j. \quad (1)$$

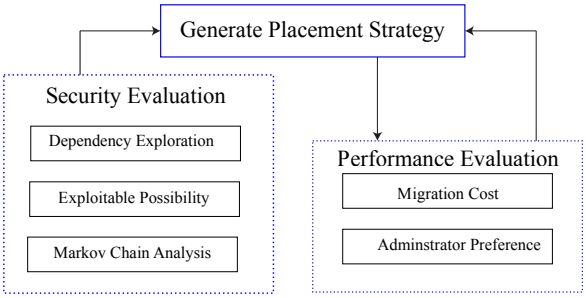


Figure 3: Architecture.

Now we change the question from how to evaluate a service to the one how to evaluate a node. Then, calculating the node survivability in Equation 1 becomes the critical problem. To solve the problem, we define the survivability of a node.

**Theorem 2 (Survivability of a Node).** *Given a node N and a set of VMs $= \{VM_a, VM_b, ...VM_m\}$ which locate at node N, and the compromised probability for these VMs are $\{P_a, P_b, ..., P_m\}$, the survivability (PN) for Node N is below:*

$$PN_N = \prod_{j=1}^{m} (1 - P_j) \quad (2)$$

Because we assume if an adversary takes over a VM, the physical node will highly possibly be compromised. In other words, survivability possibility of a physical node is possibility that all owned VMs survive in this attack, so we can obtain equation 2. Similarly, survivability possibility of service is possibility that all VMs which constitute this service can survive in this attack. In addition, because we assume if the node is compromised, all VMs on this node can't work, the survivability possibility of VM is equal to survivability possibility of physical node, so we can acquire Equation 1.

# 4 IMPLEMENTATION

The structure of our implementation is shown in Figure 3. In order to improve the overall security level while not sacrificing performance, our design includes three components: security evaluation, strategy generation, and performance evaluation. Periodically, the cloud provider changes VM placement to defend against

placement based Attacks. First, a dependency exploration mechanism identifies the service dependency relations among VMs. In our design, we identify the dependency relations through network connections. Since the operating system maintains the information about network port and IP address, we collect the network information through the operating system. In the following, we score each VM's security level according to the National Vulnerability Database (NVD). Afterwards, we map the vulnerabilities to the possibility of compromise and leverage Discrete Time Markov Chain Analysis (DTMC) to predict the possibility of successful attacks in each attack step. Finally, we design an algorithm to create the placement plan. Integrated with migration cost and migration time analysis, we can conclude the final placement solution.

## 4.1 Security Evaluation

The Security Evaluation component consists of exploitable possibility assessment and Markov chain analysis. First, we explore dependency relations among all VMs and construct a graph based on the dependencies. Second, we scan each VM against the National Vulnerability Database to generate a estimate value in terms of the VM's security level. Third, we use Markov chain to predict the possibility of being attacked for each VM.

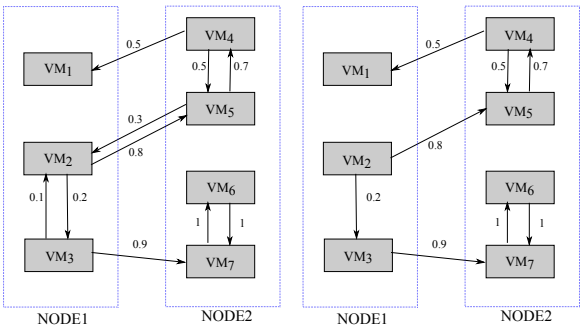
### 4.1.1 Dependency Exploration

The dependency relations among VMs is the basis of security evaluation. There are already many research on how to discover dependency relations between VMs. For example, LWT (Apte et al., 2010) identifies the cross-domain dependencies by CPU utilization. The authors claim that there will be the same spike in the CPU utilization of dependent VMs. Hence how to identify the dependency of VMs is out of the scope of our work. In this paper, we simply use network topological structure information like IP address and network port numbers generated by `netstat` to identify dependency relations. After all dependency relations are obtained, we can construct the VM Dependency Graphs like the one shown in Figure 2.

### 4.1.2 Exploitable Vulnerability

In order to quantify the exploitable vulnerability, we use the Common Vulnerability Scoring System(CVSS) (CVSS, 2012). CVSS includes three metrics group: base, temporal, and environment. Each of the metrics represents different characters of vulnerabilities.

Now we have the quantified vulnerability for each VM. We need to map the quantified vulnerability to the possibility of compromise for each connected VM. Here we use a linear mapping function. Given a VM $VM_a$ and the vulnerability score of VMs connected with $VM_a$ is $\{V_1, V_2, ..., V_n\}$, the possibility of compromise for $VM_a$, denoted by $P_a$, is given by Equation 3.




(a) Step One. (b) Step Two.

Figure 4: An example based on Markov Chain Analysis.

$$P_a = \frac{V_a}{\sum_{k=1}^{n} V_k} \tag{3}$$

The linear mapping Function 3 is very simple, however, more complex mapping functions can be used and the following discussions will remain the same.

### 4.1.3 Markov Chain Analysis

We run Markov Chain analysis over an *Attack Dependency Graph* which is defined as follows.

**Definition 1.** ***Attack Dependency Graph****: Given the Dependency Attack Graph $G\langle V, E\rangle$ of an attack and two virtual machines $v_1$ and $v_2$, where $V$ is the set of virtual machine, and $E$ is the set of transitions among attack, if there exists an edge $e \in E$ from $v_1$ to $v_2$, then $v_2$ is* attack dependent *on $v_1$, denoted by $v_1 \rightarrow v_2$. Besides, the possibility of transition is determined by Equation 3*

All virtual machines in the cloud and attack dependencies among them can be represented by one or a set of attack dependency graph(s) (ADG). Besides if we associate a probability to each edge, an ADG can be modelled by a Discrete Time Markov Chain (Sahner et al., 1997). Given $n$ nodes in the ADG, the initial probability distribution on each node (that a particular node is compromised) is $\underline{\pi}(0) = (1, \underbrace{0, 0, \ldots, 0}_{n-1})$. After the $k^{\text{th}}$ step, the probability that the attacker can reach other nodes in the ADG can be calculated by Equation 4.

$$\underline{\pi}(n) = \underline{\pi}(0)\mathbb{P}^n \tag{4}$$

where $\mathbb{P}$ is the state-transition probability matrix of DTMC and $\mathbb{P} = \underbrace{\mathbb{P} \cdot \mathbb{P} \cdots \mathbb{P}}_{n}$. $\mathbb{P}$ is given by $\{a_{ij}\}$, where $a_{ij}$ is the probability associated to edge $v_i \rightarrow v_j$.

The initial $\mathbb{P}$ can be determined by attacker's first choice. If attacker can compromise $v_j$ from $v_i$ based on compromised possibility which has been scored by CVSS and mapped by equation 3, then we assign $a_{ij}$ as this score. In order to eliminate loop in ADG, we will remove the compromised node and its edges. Here, to simplify our work, we just convert the example (Figure 2) to ADG format and assume that $\frac{1}{m}$ as the probability to each transition to the successor if $v_i$ has m successors. However, it's not hard to assign the initial $\mathbb{P}$ using CVSS (CVSS, 2012).

In Figure 2, we assume that the first compromised VM should be $VM_2$, so $\underline{\pi}(0) = \{0\ 1\ 0\ 0\ 0\ 0\ 0\}$. Hence we may obtain the ADG for step one and step two in Figure 4. Finally attack possibility for 6 attack step from above assumption.

$$\begin{pmatrix} \underline{\pi}(0) \\ \underline{\pi}(1) \\ \underline{\pi}(2) \\ \underline{\pi}(3) \\ \underline{\pi}(4) \\ \underline{\pi}(5) \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.2 & 0 & 0 & 0.8 & 0 \\ 0 & 0.26 & 0 & 0.56 & 0 & 0 & 0.18 \\ 0.56 & 0 & 0.26 & 0 & 0 & 0.18 & 0 \\ 0 & 0.026 & 0 & 0 & 0 & 0 & 0.414 \\ 0 & 0 & 0.026 & 0 & 0 & 0.414 & 0 \end{pmatrix} \tag{5}$$

where $\underline{\pi}(k), 0 \leq k \leq 5$ is the probability distribution in step $k$. In the above example, according to $\underline{\pi}(4)$, in the $4^{\text{th}}$ step, the probability that $VM_2$ is compromised will be $2.6\%$ and the probability that $VM_7$ is compromised will be $41.4\%$.

## 4.2 Placement Generation

From the above discussion, we have gained the possibility of being attacked for each VM. Therefore, we design an algorithm to generate placement plans. The core part of the algorithm is to separate VMs with high risks from VMs with low risks. The algorithm tries to assign VM with connections to same node.

In the first part of the algorithm, we separate VMs from others. The VMs with high risks, called *dangerous VMs* are identified by DTMC analysis. In DTMC analysis, at a specific step, if the probability of being compromised of a VM is larger than zero, the VM has high security risk in this step. A VM compromised in earlier steps is considered at higher risk level than the ones compromised at later steps. To satisfy our goal functions, we sort VM set in descending order of attack possibility. Then, we assign each node a VM from the first one of VM set. Then we choose the node with minimal attack possibility to hold the rest of VMs. If the current node is full, we choose next node until VM set is empty. To minimize the number of migratedVM, our placement plan will consider the previous plan. For example, if one VM belongs to the physical node without dangerous VMs in preceding placement and for new placement plan, this VM is still put into a node without dangerous VMs. Then we will not migrate this VM because migration this kind of VM will not increase VM security but decrease the migration performance.

## 4.3 Performance Evaluation

In this section, we discuss the overhead of our placement mechanism. We will look at the following types of costs: the computing costs of placement, the *migration time* of a VM, and the total number of migration needed to achieve a new placement, or *migration path*.

### 4.3.1 Computing costs

The computing costs include the costs of DTMC and the algorithm. The algorithm complexity is polynomial. The cost of DTMC, denoted by $C_{DTMC}$, is defined as the following.

**Definition 2 (Cost of DTMC).** *Given a series of K Attack Steps and the cost for step i is $PDTMC_i$, the performance cost for this series of attack steps is*

$$C_{DTMC} = \sum_{i=1}^{K} PDTMC_i \tag{6}$$

According to our experiments, the costs of matrix multiplication for each attack step in DTMC is *polynomial time complexity* is $PDTMC_i = i^3$. Our experiment result of DTMC shows calculation complexity for a cloud with 2048 VMs in term of 7 steps, $C_{DTMC} = \sum_{i=1}^{7} PDTMC_i = 38.36s$. In general, regarding with a cloud with N VMs in term of M steps, the total calculation for DTMC is $\sum_{i=1}^{M} PDTMC_i^N$.

### 4.3.2 Migration Time and Migration Path

When migrating a VM, the VM is usually shut off first, hence, migration time is one of the most significant factor we should consider in order to improve the system performance. Figure 5 presents a migration delay of Web server on our platform.

The cost on migration path happens when a new placement plan is deployed. The cost may differ if the VMs are migrated in a different order because immigrated VM should wait until the target node has enough space. In addition, we should choose suspended VMs to migrate first because migrating suspended VMs will not cause performance loss. Therefore, we should try to

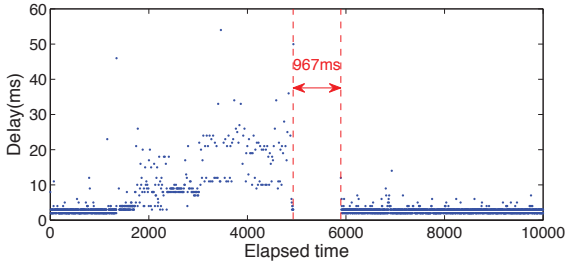



Figure 5: Migration impact on response delay of web server. As illustrated in the graph, the web service downtime due to migration is 967ms.

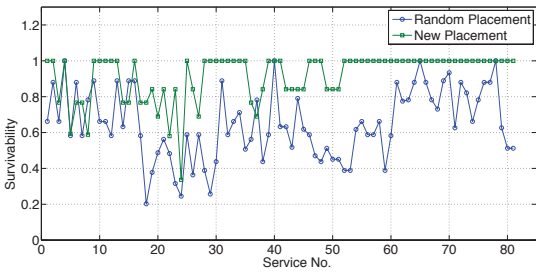



Figure 6: Comparison of Survivability.

choose a migration path with minimum costs. Calculation of the optimal migration path is out of the scope of this paper due to limit of space.

## 5 EXPERIMENTAL RESULTS

We apply our placement algorithm to the data set to generate placement plans. The data set includes 81 VMs on 10 node. The capacity for 10 nodes are 20,15,10,10,10,5,5,5,5,5. Based on the data set, we generated a random placement plan and optimize the placement using our algorithm. We compared the new placement plan with the random one to investigate the improvement of security levels.

According to our experimental results shown in Figure 6, 91.3% services obtained improved survivability. The maximum survivability enhancement is 74.28% and the average improvement of survivability possibility is 27.15%. Our results also show reduced number of compromised VMs.

## ACKNOWLEDGEMENT

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

## REFERENCES

Apte, R., Hu, L., Schwan, K., and Ghosh, A. (2010). Look who's talking: discovering dependencies between virtual machines using cpu utilization. In *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*, HotCloud'10, pages 17–17, Berkeley, CA, USA. USENIX Association.

CVE-2007-4993 (2007). Cve-2007-4993: Xen guest root can escape to domain 0 through pygrub. `http://cve.mitre.org/cgibin/cvename.cgi?name=CVE-2007-4993,2007.`

CVE-2007-5497 (2007). Cve-2007-5497: Vulnerability in xenserver could result in privilege escalation and arbitrary code execution. `http://support.citrix.com/article/CTX118766,2007.`

CVSS (2012). Common vulnerability scoring system. `http://www.first.org/cvss/cvss-guide`.

Hlavacs, H., Treutner, T., Gelas, J., Lefevre, L., and Orgerie, A. (2011). Energy consumption side-channel attack at virtual machines in a cloud. In *Dependable, Autonomic and Secure Computing (DASC), 2011 IEEE Ninth International Conference on*, pages 605 –612.

Lucas Simarro, J., Moreno-Vozmediano, R., Montero, R., and Llorente, I. (2011). Dynamic placement of virtual machines for cost optimization in multi-cloud environments. In *High Performance Computing and Simulation (HPCS), 2011 International Conference on*, pages 1 –7.

Ristenpart, T., Tromer, E., Shacham, H., and Savage, S. (2009). Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In *Proceedings of the 16th ACM conference on Computer and communications security*, CCS '09, pages 199–212, New York, NY, USA. ACM.

Sahner, R., Trivedi, K., and Puliafito, A. (1997). Performance and reliability analysis of computer systems (an example-based approach using the sharpe software. *Reliability, IEEE Transactions on*, 46(3):441.

Sindelar, M., Sitaraman, R. K., and Shenoy, P. (2011). Sharing-aware algorithms for virtual machine colocation. In *Proceedings of the 23rd ACM symposium on Parallelism in algorithms and architectures*, SPAA '11, pages 367–378, New York, NY, USA. ACM.

Yusoh, Z. and Tang, M. (2010). A penalty-based genetic algorithm for the composite saas placement problem in the cloud. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1 –8.

Zhang, Y., Li, M. L., Bai, K., Yu, M., Zang, W., and He, X. (4-6 June 2012). Incentive compatible moving target defense against vm-colocation attacks in clouds. In *IFIP International Information Security and Privacy Conference 2012*.