Migrant Attack: A Multi-Resource DoS Attack on Cloud Virtual Machine Migration Schemes

Jia-Rung Yeh
Department of Computer Science
National Taiwan University
Taipei, Taiwan
Email: jiarung.yeh@gmail.com

Hsu-Chun Hsiao
Department of Computer Science
National Taiwan University
Taipei, Taiwan
Email: hchsiao@csie.ntu.edu.tw

Ai-Chun Pang
Department of Computer Science
National Taiwan University
Taipei, Taiwan
Email: acpang@csie.ntu.edu.tw

***Abstract*—Live virtual machine (VM) migration is the core technology in elastic cloud computing. With live VM migration, cloud providers can improve resource use and quality of service by adjusting the VM placement on demand. However, live migration is expensive because of high CPU usage and the negative effect on co-located VMs, and frequent live migration thus severely undermines the performance of the cloud. Although existing dynamic allocation schemes are designed to minimize the number of live migrations, this study demonstrated that a denial-of-service adversary can cause excessive live migrations by exploiting dynamic allocation. The attack, which we term *migrant attack*, deliberately varies the resource usages of a malicious VM to trigger live migration. A crucial feature of the migrant attack is that even if VMs on the same physical machine are perfectly isolated through virtualization, a malicious VM can still affect the availability of the co-located VMs. As proof of concept, we investigated two common VM allocation schemes: load balancing and consolidation. We evaluated the effectiveness of the attack by using both simulations and testbed experiments. We also discuss several potential countermeasures, such as enforcing another layer of isolation between malicious and harmless VMs in dynamic allocation schemes.**

I. Introduction

With the emergence of virtualization and cloud computing, users and developers can employ resources in a more effective and flexible manner than ever [1]. However, the success of cloud computing services attracts unwanted attention from malicious parties. A recent survey reported that 34% of cloud providers encountered 11-50 denial-of-service (DoS) attacks per month in 2014. While most of these attacks against clouds so far were conducted through traffic flood [2], concerns have been raised regarding whether any of the unique cloud properties expands the DoS attack surface.

One of the unique cloud properties is *live migration*: the virtualization technology allows cloud providers to move running virtual machines (VMs) among physical machines (PMs) on demand, thereby improving resource use and quality of service (QoS). In this paper, we consider two common dynamic allocation policies—load balancing and consolidation—and assume the allocator will continue to live migrate until resource use is in an acceptable state.

However, live migration is expensive. Because of their high resource usage, colocated VMs require more time to accomplish their workload in the original PM during migration [3]. Therefore, frequent live migration severely undermines cloud performance. Although existing dynamic allocation schemes that determine whether each VM requires migration are often designed to minimize the number of live migrations, in this paper, we show that an adversary can cause excessive live migration by exploiting dynamic allocation.

We present the *migrant attack*, a novel DoS attack that aims to destabilize a cloud's VM placement using a small number of compromised VMs. The adversary deliberately manipulates the resource usage of the compromised VMs to trigger live migration, thereby degrading the performance of co-located VMs and the cloud. The main challenge of designing such an attack is how to reduce the number of required compromised VMs, so that the attack can be both efficient and stealth. To address this challenge, the migrant attack builds on the following two key observations of dynamic allocation:

- Because the allocator monitors and schedules resources periodically, the allocation scheme is subject to a time of check to time of use (TOCTOU) problem. That is, the adversary can control the resource use of the compromised VMs such that the allocator is tricked to issue a migration request at the time the allocator checks, but by the time the allocator actually starts the migration process, the migration will further push the cloud away from a stable state.
- Because the allocator employs multiple dynamic allocation policies over multiple resources, the adversary can choose to invoke one that requires minimal work. In this study, we consider two allocation policies (i.e., load balancing and consolidation) and three resources (i.e., CPU, bandwidth, and memory), which renders six possible conditions to trigger live migration.

Based on these two observations, we demonstrated that VM allocators can be misled and therefore wrongly perform unnecessary migrations.

The main contributions of this study can be summarized as follows:

1) We identify new DoS vulnerabilities exploiting dynamic VM allocation.
2) We highlight the main challenge of exploiting such vulnerabilities in a practical manner and propose to leverage

978-1-5090-2285-4/16 $31.00 © 2016 IEEE
DOI 10.1109/AsiaJCIS.2016.14

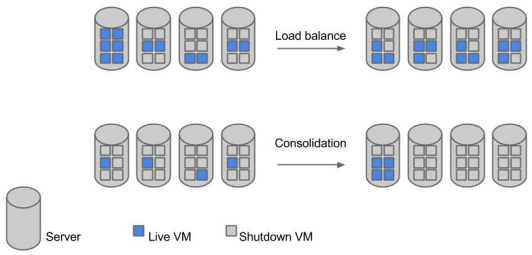

Fig. 1. Two basic strategies for tuning PM resource use.

the TOCTOU problem and target multiple resources and policies for improving efficiency and stealthiness of the attack.

3) We design a DoS attack (called the migrant attack) that can destabilize a cloud's VM placement using a small number of compromised VMs.
4) We evaluate the migrant attack using both testbed experiments and large-scale simulations.

This paper is structured as follows. Section II provides the cloud service background, allocation and migration, and OpenStack. Section III is divided into preparation and attacking parts. Section IV presents metrics related to the cost of attackers and damage, and Section V evaluates the experimental results. Section VI provides related works. Finally, Section VII presents the conclusion and future works.

II. Background

Cloud computing is an integration of virtualization and large-scale data center computing [1], which aims to use and manage resources more efficiently; it is designed to solve large-scale computation problems.

All these services must be deployed on a physical server for operation, indicating that the location where these services or VMs are launched influences the power consumption, and QoS. If a physical machine (PM) hosts only a small number of virtual machines (VMs), then we may reduce energy consumption by shutting down the PM and relocaiting the VMs to other PMs. The different distributions of VMs scattered on the PMs provide various types of feedback, leading us to adopt two basic strategies as below and Fig. 1.

1. Load balancing: A cloud transfers heavy loads from some PMs to other PMs with lighter loads.
2. Consolidation: A cloud collects the VMs in the PMs with low resource use into one PM to shut down the former PMs.

For these hybrid strategies, VM allocating methods determine how each VM is mapped to its PM to satisfy goals, such as optimal cost function, or some constraints that are limited by the physical bond of a PM's resources. Moreover, we focused on various methods of each allocator, such as the use of a PM that is required for it to be considered a high loading PM, to determine its threshold. These thresholds may be fixed or dynamic, depending on their designs.

After rescheduling these VMs, we must physically move the selected VMs among PMs. This type of movement is called migration. Currently, virtualization can be used to extract a system image from a running VM to network-attached storage, allowing migration to only manage the CPU state and memory to reduce network traffic and CPU loads.

Moreover, when migration is processed, its scheme determines when to stop the source VM and copy the final CPU state or memory data to the VM in the destination until the VM in the destination recovers to provide service. The period without service provided using VM is called downtime, and the time required for a complete migration to process is referred to as migration time. Various migration schemes can be classified into the following schemes according to the decisions reached at the time when service is halted: precopy, postcopy, and stop-and-copy migration schemes [4], and these schemes will give different downtime.

III. Attack

In this section, we describe the migrant attack and its process that exploits the weak isolation of resource use; the migrant attack is a challenge that must be addressed.

We divide this section into two parts to describe the migrant attack. In the first part of migrant attack, attackers use some batch tasks, which can be observed objectively to measure the PM's loads and the threshold of schemes from various allocators. In the second part, the attacker can use the observed threshold in the first part. With rented VMs, the interference by the attacker can mislead the allocator to falsely determine that PMs require consolidation or balance its loads. Therefore, excessive migration consumes more energy and increases service instability. We assume that the adversary can control a sufficient number of VMs for attacks. We will show in our evaluation that the required number of malicious VMs is reasonable. Because the adversary can pose as one or more regular customers to rent VMs from the cloud provider, the cloud provider cannot differentiate an attacker from a coalition of benign customers without further information.

A. Resource Use Leakage Exploitation

In this section, we differentiate PMs with different loadings by using the leakage of resource isolation. This is accomplished by examining the task completion time and whether the migration has happened. Our attack preparation involves six steps and two conditions for a loop to stop.

- Step 1: Scatter VMs on PMs, which belong to the target cloud.
- Step 2: Execute the batch tasks of each resource intensively (CPU, memory, up-link or down-link, and so on).
- Step 3: Record the time each task uses in set T.
- Step 4: Compute the variance and average of the time that each VM used.

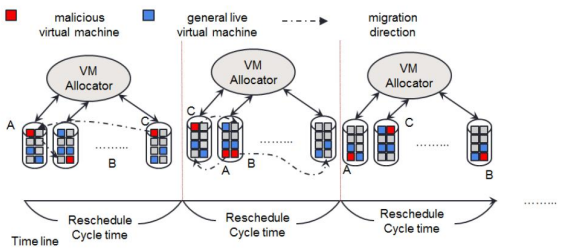

Fig. 2. Example of a migrant attack.

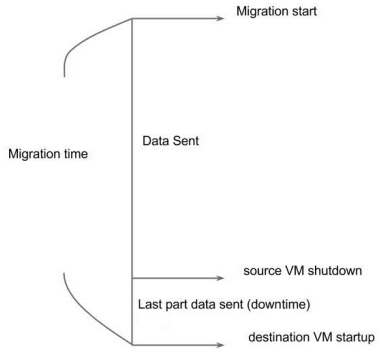

Fig. 3. Downtime locates the final part of migration.

- Step 5: Repeat Steps 2 and 4 in each scattered VM until either Condition 1 or 2 is satisfied.
- Step 6: Sort these VMs according to variance and average in decreasing order.

- Condition 1 The variance computed converged to some setting threshold.
- Condition 2 The threshold of the iteration amount.

From the recorded time, we determine the threshold that allocator trigger the migration. This problem can be divided into two parts on the basis. Firstly, whether the attacker controlling the VM monitors the downtime or not. If we observe and measure the downtime, then we can directly confirm whether this VM is migrated. Otherwise, if malicious VMs don't monitor downtime, then we divide set T into various parts in the length that the same length of converged sequence in Condition 1. According to each parts, if the consecutive part's variance has changed, then migration might have occurred.

Therefore, we consider each threshold of resource use on the PM near the allocator that is preparing to migrate. According to this, we can determine whether each VM is an appropriate attack target.

B. Migrant Attack

We only select VMs with resources close to the threshold observed. The steps in the migrant attack are as follows:

- Step 1 Execute the measuring batch tasks, as provided in section III-A.
- Step 2 Determine the type of resource most appropriate to trigger a migration.
- Step 3 Execute the intensive task by selecting a resource for a period to trigger the migration.
- Step 4 Proceed to Step 1.

For example, in Fig. 2, instants A, B, and C are malicious VMs. If the location of A is CPU intensive, then it implements CPU batch tasks that cause the allocator to observe the location of A and migrates A to another PM to allow the new distribution of VMs to have superior resource use. In each round, malicious VMs consume different resources, like on CPU overloading PM executing the task with CPU intensive, according to the threshold which is easy to trigger a migration and we get from the part of leakage. When the allocator aims to reschedule the VM placement, these PMs are easy to select for executing load balancing. However, frequent or massive migration has a negative influence on the cloud; more details for migration cost are discussed in Section VI.

IV. METRICS

We consider that the migrant attack may influence a real cloud environment. Therefore, we aim to determine the number of malicious VMs and degrees of resource use that are sufficient to influence other users to enable these users to be aware of malicious behavior. To quantify the success of the migrant attack, we consider it from two perspectives: user sensitiveness and cloud cost, which allow us to define three metrics: downtime, migration time, and the degree of imbalance.

The downtime is a period during migration indicating that the migrated VM cannot serve its users. Various migration schemes have different downtime properties. Three types of migration schemes exist: precopy, postcopy, and stop-and-copy migration schemes. When a VM is migrated, a hypervisor copies this VM's state to its destination. However, the VM continues serving, and some memory states change, similar to copy processing. The hypervisor pauses the VM to copy the final memory state, and the three types of migration schemes provide various methods of determining the pause time point. The scheme used in this study is precopy migration, and it provides the downtime in the rearward part of migration.

Migration time is the summation of all migrations in the cloud. It is closely related to the reallocation frequency because a more frequent reallocation provides more migration time in the whole cloud environment.

The degree of imbalance is a metric that represents allocator performance, and we define it as the ratio of two resource use values. The first value is the difference between maximum(U_{Max}) and minimum(U_{min}) resource use in the same PM. And this maximum/minimum is refer to the different PMs' loads in the same time slot. For the second value, we computed the average(U_{Avg}) of the PMs by employing the resource use in the same time frame. These values indicate that resource use varies during the time frame. In a fixed moment, if this value is lower, then the entire system is more balanced or smooth.

$degree\ of\ imbalance = \frac{U_{Max} - U_{Min}}{U_{Avg}}$

The attacker's goal should be increased with the migration times, downtime, or degree of imbalance.

V. Evaluation

In this section, we present our experimental results. To describe the migrant attack in a real environment in a more detailed manner, we implemented emulation and simulation.

A. Emulation

For improved readability, we divided the experiment into four parts: testbed, exploitation, migration cost, and degree of imbalance. First, we introduce the testbed. Second, resource use exploitation emphasizes that various loads on the PM are differentiated using the VM performance. Third, we consider the cost of migration, which provides us with a direct description of the attacker and general user. Finally, we highlight the influence of the migrant attack on the allocator performance.

1) Testbed Model: We did not implement the migrant attack in a real environment because of legal and ethical concerns. Instead of actually renting VM to mount the migrant attack, we set up a small-scale but complete environment with a controller that behaves similarly to a specific allocator with policies for load balancing and consolidation, and we emulated the loads model on this cloud. In this section, we describe our environment and general loads model.

Our environment is small scale compared with a real cloud environment because establishing a large-scale environment to conduct a security experiment without disturbing other users is unrealistic. In addition, obtaining a small, independent, and meaningful trace for a data center or cloud data center is difficult. Without loss of generality, we emulated our CPU and network loads by using mathematical computation and http service to allow our VM to consume resources similar to a real environment, and our archiecture is on the Fig. 4.

We process a routine task called regular load in each VM that retrieves a job list to allow various tasks to continue executing every 2 s, and each list is completed approximately between 3 and 20 s. These tasks in the list are classified into three types of intensive jobs (CPU, upload, and download) for the purpose of emulating the general cloud load. And we constructed a central request dispatcher that determines and changes the load in every VM every 20 s. Each VM can obtain the information of use from the central request dispatcher.

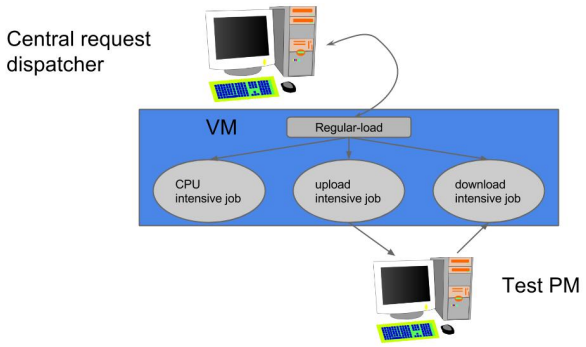

Fig. 4. Relation of VM, central request dispatcher and test PM.

According to this list, regular load determines the amount and the kind required for tasks in each round.

However, determining whether the PM required migration according to some static thresholds or particular methods in a realistic attack is difficult. The migrant attack can dramatically reduce the quality of the cloud if massive VMs are to be migrated. Therefore, we used a template of the allocator for load balancing and consolidation proposed by Xiao et al [5]. The allocator evaluates each PM by using skewness value and determines the PM suitable for load balancing or consolidation. The allocator selected some VMs from their own PMs needing load balancing or consolidation to migrate to the destination PM according to the value of skewness they proposed. We compare the quality of performance of load balancing among the migrant attack, a general scenario without any attacks, and a naive attack that consumes all the resources of the VM.

In emulation, we choose CPU, uplink bandwidth, and downlink bandwidth as our benchmark to determine whether PMs are overloading or needed a consolidation in utilization according to that the migration times of general scenario below 40 times in three hours. Since different applications make huge difference in these three kinds of resource on different VMs. We keep the average utilization of CPU between 15 to 75, the average transferring packet rate between 100 to 1500, and the average receiving packet rate between 150 to 2400.

We constructed our testbed by using physical servers with Intel Core(TM) i7-3770 3.40 GHz, 15-GB RAM, and 1-TB HD and used KVM as the hypervisor, whereas all the VMs have 1 VCPU and 512-MB RAM. In addition, we used the network tool ping to monitor the downtime. Because our allocator sent a migration request to the target PM, we pinged the migrated VM every 0.1 s to obtain the downtime by summing consecutive loss packages close to the trip time. We used two sets of thresholds to determine the requirement for load balancing. Finally, we installed and configured the latest OpenStack release called Icehouse on the operating system with Ubuntu 12.04TLS to better investigate conduct our experiment more efficiently [6].

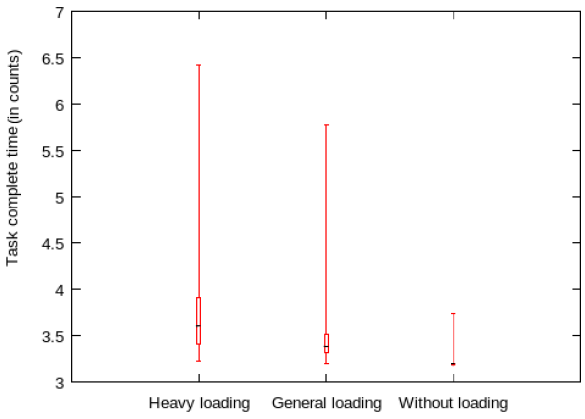

Fig. 5. Distribution of tasks using the time between different compute loads.

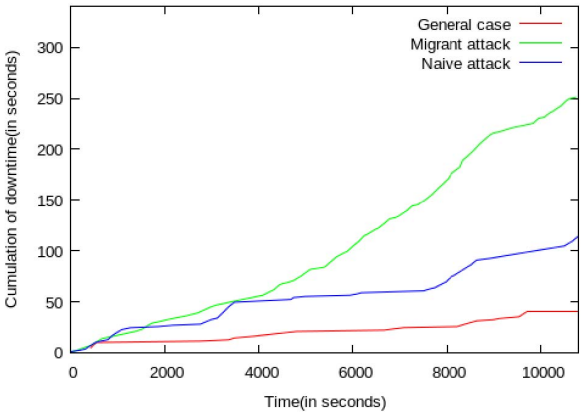

Fig. 6. Cumulation of downtime in different scenarios.

2) Leakage Test: The variance of VM can decide different conditions that determine the scale of PM. We selected three conditions for the PM with various loadings according to the number of VMs launched and workloads on the VMs. In the PM without loads, we launched only one VM to execute the leakage tests. In the PM with general loads and PM with heavy loads, we launched five VMs and selected one VM to execute the leakage test. Conversely, we placed general loads and heavy loads according to the testbed model. As displayed in Figs. 5, the variance is distinguishable among the three PMs. Moreover, the average and median numbers are features that help to distinguish the loading scales of the PM.

3) Measure the Damage During Downtime and Migration Times: Two scenarios of setting the VMs, 15 general VMs, and 13 general VMs with 2 malicious VMs are presented. Our schedule provides a new placement to migrate a VM every 2 min except during downtime. We compared the naive resource-consuming attack and migrant attack by measuring the migration times of each VM. The naive resource-consuming attack triggers slightly more migrations than the condition without any attacks does because the schedule efficiently manages various loads. The migrant attack can mislead the schedule to perform needless migrations to other VMs. In addition to migration times, we measured the downtime of migrated VMs in the migration process, observing the average downtime of each migration that increases slightly as the migration times increase.

On the user side, we recorded the migration times of different VMs. These malicious VMs were selected to migrate intermittently rather than frequently, and the malicious VMs implementing the migrant attack had more migration times than other VMs did, as displayed in Fig. 7 and Fig. 8. Moreover, the scenario with naive attacked VMs had slightly more migration times than the scenario without any attacks did(Fig. 6).

4) Measurement of the Damage Caused by Using the Imbalance Degree: Similar to the previous experiment, we recorded

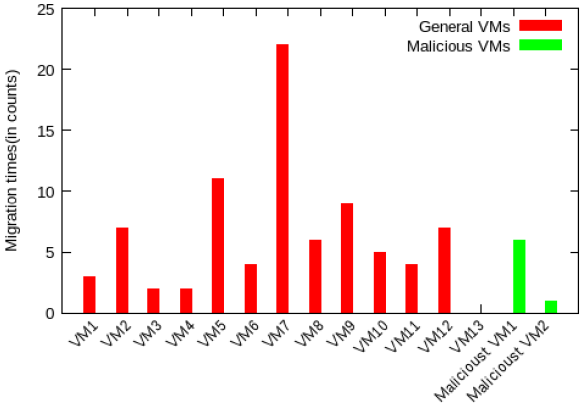

Fig. 7. Migration times of each VM where two malicious VMs implement the migrant attack.

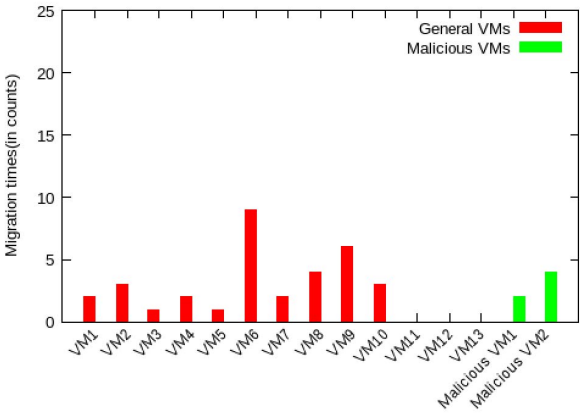

Fig. 8. Migration times of each VM where two malicious VMs implement the naive resource-consuming attack.

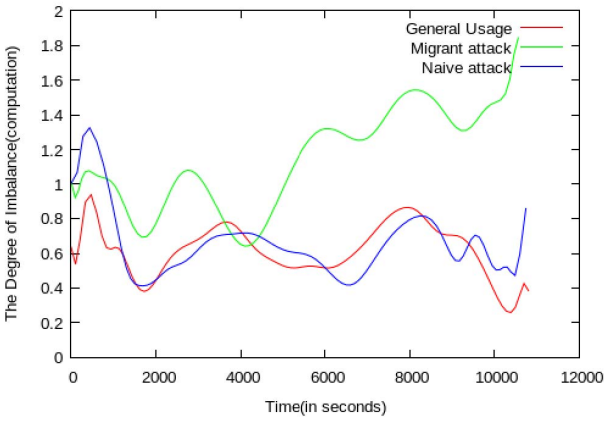

Fig. 9. Degree of imbalance between various PMs when computing the resource.

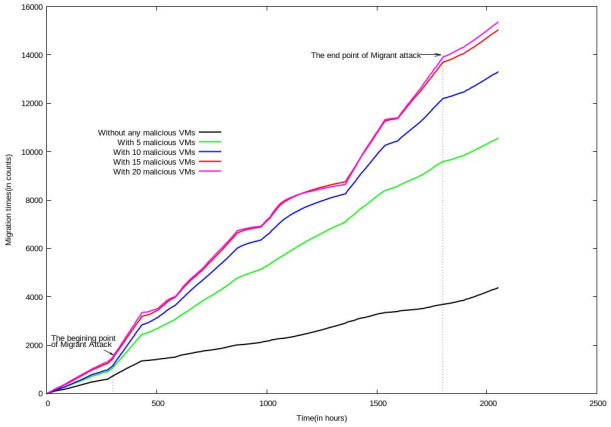

Fig. 10. Cumulation of average migration times in different scenarios.

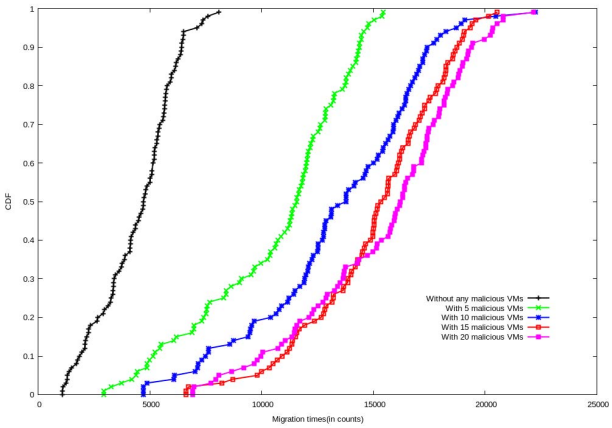

Fig. 11. CDF of migration times after each simulation.

the use of resources to compute the degree of imbalance. Moreover, we transformed our raw data into Bezier curves to show the trends for better readability. Since the Bezier curve is a smooth curve used to model smooth curves from the data difficult to read that can be scaled indefinitely. The degree of imbalance for the scenario without malicious VMs is less than that for the scenarios with malicious VMs(Fig. 9). However, some general scenarios are close to or higher than the attacked scenario for the following two reasons: (1) because the use of such a resource is decreasing, the decreasing condition may not be persistent, and consolidation or load balancing is not conducted in real time and (2) when a particular resource is subjected to the migrant attack, the other resource is not influenced by the migrant attack.

B. Simulation

We simulated another experiment to demonstrate the influence on the large-scale condition, and implement the restriction of migration to mitigate the influence of frequently migrations.

1) Environment: Wolski et al. presented a IaaS dataset, which records the VMs' used cores, start, and stop time in a company with 50000 100000 employees [7]; 24 cores are present in each node. For the original frequent migration, for which our use simulation was employed, we assumed that 20 nodes are present instead of 13 nodes. We simulated three types of resource use because determining the precise information regarding each type of resource use of every VM is difficult. We randomly generated resource use in N(40,3.5), which distributes approximately 30%-50% VMs for initialization; we increased 20% in N on average for the intensive resource used. For a real-time and long-tail effect, we changed each use value by a step, which is uniformly random from -3 to +3 units after every second.

In particular, for a higher number of nodes and VMs than the amount of emulation, we set the reallocation frequency to 10 min once which reference from the consolidation work [8]. Similarly, according to the behavior of migration times without any attacks, we determine whether PMs are not overloading or needed a consolidation between 20 to 80 in the first resource, and 20 to 75 in the second and third resource. In our simulation, almost 9000 VMs were initiated and distributed in 2000 h. Each malicious VM assumes four cores.

2) Result: We processed each scenario 100 times for 0, 5, 10, 15, and 20 malicious VMs, and all malicious VMs mount migrant attacks from the 300th hour to the 1800th hour. According to Fig. 10 and Fig. 11, the trend of each scenario with migrant attack stops, and the incerase of migration times then slow. Otherwise, the migration times increase slowly during some segments of the migrant attack because of the few VM requests (Fig. 12). According to Fig. 11, when more malicious VMs are present, each scenario on average has more migration times. One trend is that more malicious VMs have a wider range of migration time distribution. This indicates a positive relationship between the stability of migration times and the malicious VMs.

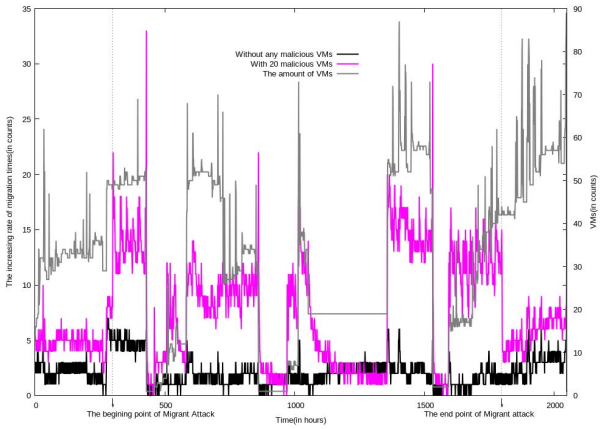

Fig. 12. Comparison of migration times and VM initiated in each hour in the cloud.

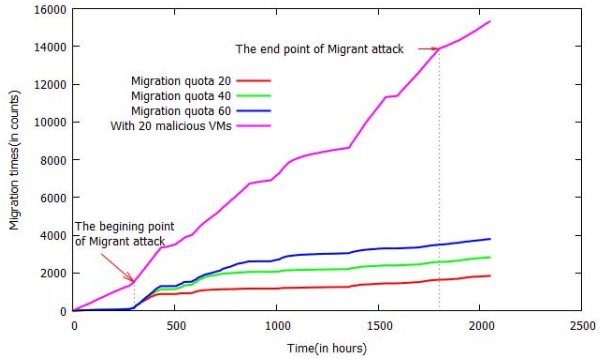

Fig. 13. The comparison of different migration quota settings.

3) Mitigation: We set a mitigation scheme that we divide all PMs into two groups: Restraining group and Loosing group. Loosing group is a group without scheduling of QoS and resource utilization by gathering some VM with using up their migration quota, and Restraining group is the other group with regular scheduling as before. Then, we determine different thresholds of quota by 20, 40, and 60. For comparison, we test this mitigation scheme with the scenario has 20 malicious VMs, and repeat the simulation ten times in each scenario. From Fig. 13, we can see that migration quota can effectively mitigate the burst of migration times.

VI. RELATED WORKS

The impact of a migrant attack greatly depends on the allocator and migration cost. Therefore, we divide this section as follows: VM allocation, migration cost, and other cloud DoS attacks.

A. VM Allocation

In this scenario, an allocator may easily exhibit superior performance or resource use but is also an easy target of the migrant attack. According to the allocation policy, determining the threshold is crucial to ascertaining whether use should involve load balancing or consolidation. Therefore, we classified previous methods according to the fixed and unfixed thresholds.

Most studies have used a fixed threshold. After Beloglazov et al. reported an allocation problem related to energy use and QoS [9], a multiobjective method was developed. Using this method, efficient use with lower power consumption was obtained, and this problem was transformed into different convex, genetic, and max-min problems [10], [11], [12]. Moreover, Belabed et al. proposed a model with various topologies of a data center to present a reasonable topology for the cloud data center [**?**].

Other studies have reported only two methods with unfixed thresholds used for determining whether a PM is overloaded. One of the methods is a shadow-routing-based dynamic algorithm proposed by Guo et al. They formulated the problem as a max-min problem that dynamically changes the distribution of VM by using constraints [13]. The other method is the Markov chain model proposed by Beloglazov et al. They solved the condition with unknown and nonstationary workloads by using a sliding window. The Markov chain model provides a sequence of states that distributes converged VMs [14].

Otherwise, we have found some allocators begin to notice the interference between co-locate VMs. Corradi et al. implement a consolidation allocator with Openstack and discuss the interference between VMs do influence the allocator performance [8]. Otherwise, Caron et al. proposed the model concern the interference between VMs in QoS which they define them as isolation requirements [15].

B. Migration Cost

We aimed to determine the extent of damage caused by the migrant attack in a real system. Therefore, we evaluated some topics related to migration cost that highlighted the positive relationship between the performance of workloads on the VMs and the migration times [16]. In addition, the relationship between power consumption and time of migration is positive [3]. Therefore, as our migrant attack begins to increase migration times, the various costs of migration increase.

C. Other DoS Attack

The attacked target is increasingly close to the core network, even cloud provider. Alarifi et al. design a class of DoS attacks to IaaS clouds similar as us. Differently, their attack cheats the protocol by building on some existed wave of resource usage, and our attack skilly choose the weakly or close to overloading resource to attack [17]. A coremelt attack uses subverted machines between a link to saturate that link [18]. Similarly, crossfire attacks target a group of network apparatuses belonging to a network area to degrade or block the area's network availability [19]. Other attacks focus on the saturation of a link in the cloud environment [20], [21].

Furthermore, some attacks focus on the weakness of the protocol and design of defense schemes to send traffic that forces low use of the scheme or apparatuses or failure to

provide service [22], [23]. Moreover, some attacks target the hypervisor. Zhou et al. proposed the exploitation of virtual CPU allocation, in which malicious VMs extract more CPU cycles than other VMs do, and they implemented this attack in a public cloud [24]. Varadarajan et al. used a similar concept called resource-freeing attack, in which the attack using more resource to squeezing others' resource using [25]. Duncan et al. proposed an attack scenario in which a malicious insider may use a higher level of authorization to cause damage during migration [26].

VII. Future Work and Conclusion

The naive idea of defense and mitigation can be applied to complex allocators and limit the use of resources. The concept of a migrant attack can be applied to other types of dynamic allocation. Moreover, a migrant attack can be used to coordinate with the particular victim VM or service. It may result in a chain migration.

The VM allocation scheme is crucial for ensuring elasticity in the cloud data center. Lose elasticity will increase latency of each request and incur additional cost because of high power consumption and decreased revenues. A conventional DoS attack on a cloud focuses on eluding the defense scheme or targets a bottleneck in the service provider. This study demonstrated that a malicious VMs can attack an allocation scheme and cause damage to users and clouds. We hope that this paper motivates more detailed investigations into the migrant attack and eventually provides a solution for it.

Acknowledgment

This work was supported in part by the Ministry of Science and Technology under Grant 103-2221-E-002-142-MY3 and Grant 104-3115-E-002-005, and in part by the Institute for Information Industry (III) under Grant 105-FS-C06.

References

[1] Bhaskar Prasad Rimal, Eunmi Choi, and Ian Lumb. A taxonomy, survey, and issues of cloud computing ecosystems. In *Cloud Computing*, pages 21–46. Springer, 2010.

[2] D Anstee, A Cockburn, and G Sockrider. Worldwide infrastructure security report. Technical report, Burlington, MA, USA, 2014.

[3] Waltenegus Dargie. Estimation of the cost of vm migration. In *Computer Communication and Networks (ICCCN), 2014 23rd International Conference on*, pages 1–8. IEEE, 2014.

[4] William Voorsluys, James Broberg, Srikumar Venugopal, and Rajkumar Buyya. Cost of virtual machine live migration in clouds: A performance evaluation. In *Cloud Computing*, pages 254–265. Springer, 2009.

[5] Zhen Xiao, Weijia Song, and Qi Chen. Dynamic resource allocation using virtual machines for cloud computing environment. volume 24, pages 1107–1117. IEEE, 2013.

[6] Openstack. In *https://www.openstack.org/software/icehouse/*, 2014.

[7] Richard Wolski and John Brevik. Using parametric models to represent private cloud workloads. *Services Computing, IEEE Transactions on*, 7(4):714–725, 2014.

[8] Antonio Corradi, Mario Fanelli, and Luca Foschini. Vm consolidation: A real case based on openstack cloud. *Future Generation Computer Systems*, 32:118–127, 2014.

[9] Anton Beloglazov, Jemal Abawajy, and Rajkumar Buyya. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future generation computer systems*, 28(5):755–768, 2012.

[10] Thuan Duong-Ba, Thinh Nguyen, Bella Bose, and Tuan Tran. Joint virtual machine placement and migration scheme for datacenters. In *Global Communications Conference (GLOBECOM), 2014 IEEE*, pages 2320–2325. IEEE, 2014.

[11] Ting Yang, Young Choon Lee, and Albert Y Zomaya. Energy-efficient data center networks planning with virtual machine placement and traffic configuration. In *Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on*, pages 284–291. IEEE, 2014.

[12] Nguyen Trung Hieu, Marco Di Francesco, and Antti Yla-Jaaski. A virtual machine placement algorithm for balanced resource utilization in cloud data centers. In *Cloud Computing (CLOUD), 2014 IEEE 7th International Conference on*, pages 474–481. IEEE, 2014.

[13] Yang Guo, Alexander L Stolyar, and Anwar Walid. Shadow-routing based dynamic algorithms for virtual machine placement in a network cloud. In *INFOCOM, 2013 Proceedings IEEE*, pages 620–628. IEEE, 2013.

[14] Anton Beloglazov and Rajkumar Buyya. Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints. *Parallel and Distributed Systems, IEEE Transactions on*, 24(7):1366–1379, 2013.

[15] Eddy Caron and Jonathan Rouzaud Cornabas. Improving users' isolation in iaas: Virtual machine placement with security constraints. In *Cloud Computing (CLOUD), 2014 IEEE 7th International Conference on*, pages 64–71. IEEE, 2014.

[16] Kateryna Rybina, Abhinandan Patni, and Alexander Schill. Analysing the migration time of live migration of multiple virtual machines. In *4th International Conference on Cloud Computing and Services Science (CLOSER 2014)*, 2014.

[17] Suaad Alarifi and Stephen D Wolthusen. Robust coordination of cloud-internal denial of service attacks. In *Cloud and Green Computing (CGC), 2013 Third International Conference on*, pages 135–142. IEEE, 2013.

[18] Ahren Studer and Adrian Perrig. The coremelt attack. In *Computer Security–ESORICS 2009*, pages 37–52. Springer, 2009.

[19] Min Suk Kang, Soo Bum Lee, and Virgil D Gligor. The crossfire attack. In *Security and Privacy (SP), 2013 IEEE Symposium on*, pages 127–141. IEEE, 2013.

[20] Huan Liu. A new form of dos attack in a cloud and its avoidance mechanism. In *Proceedings of the 2010 ACM workshop on Cloud computing security workshop*, pages 65–76. ACM, 2010.

[21] Yichuan Wang, Jianfeng Ma, Di Lu, Xiang Lu, and Liumei Zhang. From high-availability to collapse: quantitative analysis of "cloud-droplet-freezing" attack threats to virtual machine migration in cloud computing. *Cluster Computing*, 17(4):1369–1381, 2014.

[22] Massimo Ficco and Massimiliano Rak. Stealthy denial of service strategy in cloud computing. *Cloud Computing, IEEE Transactions on*, 3(1):80–94, 2015.

[23] Zhenqian Feng, Bing Bai, Baokang Zhao, and Jinshu Su. Shrew attack in cloud data center networks. In *Mobile Ad-hoc and Sensor Networks (MSN), 2011 Seventh International Conference on*, pages 441–445. IEEE, 2011.

[24] Fangfei Zhou, Manish Goel, Peter Desnoyers, and Ravi Sundaram. Scheduler vulnerabilities and coordinated attacks in cloud computing. In *2011 10th IEEE International Symposium on Network Computing and Applications (NCA)*, pages 123–130. IEEE, 2011.

[25] Venkatanathan Varadarajan, Thawan Kooburat, Benjamin Farley, Thomas Ristenpart, and Michael M Swift. Resource-freeing attacks: improve your cloud performance (at your neighbor's expense). In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 281–292. ACM, 2012.

[26] Adrian Duncan, Sadie Creese, Michael Goldsmith, and Jamie S Quinton. Cloud computing: Insider attacks on virtual machines during migration. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2013 12th IEEE International Conference on*, pages 493–500. IEEE, 2013.