Detecting VM Live Migration using a Hybrid External Approach

Sebastian Fiebig, Melanie Siebenhaar, Christian Gottron and Ralf Steinmetz
Multimedia Communications Lab (KOM), Technische Universität Darmstadt, Darmstadt, Germany

Keywords: VM Theft, Virtualization, VM Live Migration, Monitoring, Taxonomy.

Abstract: Cloud computing has become a paradigm of our time. It is not only a technical solution, but a business model to sell and rent computing power and servers. Virtual machines (VMs) are used to allow a dynamic and transparent server utilization, which is made possible by VM live migration. VM live migration allows to move VMs within and out of data centers while the VM is still running. Thus, resource usage becomes more efficient. However, VM live migration also provides an opportunity for new attack vectors, which can be used by malicious attackers. They can compromise hypervisors and afterwards steal VMs from data centers to gain control over resources. In the worst case scenario, the theft remains undetected by both system administrators and customers. In this paper, we present the first taxonomy of possible VM live migration detection approaches. There are two different monitoring approaches, i.e., internal or external monitoring, as well as different detection approaches, which correspond to the different approaches to detect migration. Moreover, we propose a hybrid external approach using delay measurement with ICMP ping and time-lag detection with the network time protocol (NTP) to detect VM live migration. We show that VM live migration can be detected by using a prototype of our hybrid external approach.

1 INTRODUCTION

Cloud computing has become a paradigm of our time. It is not a mere technical solution but also a business model. One of the key concepts of cloud computing is its dynamic resource provisioning. Through that, computing power can be co-located in data centers that can be shared by just one enterprise, but also be outsourced and used by different enterprises.

Cloud computing is based on the so-called virtualization. This feature allows dynamic resource provisioning and adaption with VMs. One of the features made possible by using virtualization is live migration. Live migration allows to move a VM between different servers using the underlying hypervisor, while the VM is running. Therefore, a seamless use of services is made possible. Because no or only barely measurable downtimes occur, it is in general hard to detect an ongoing VM live migration process. While VM live migration has many advantages, it also allows for completely new attack vectors. If the hypervisor is compromised by any means whatsoever, the live migration can be used to steal or copy VMs. Having no information from the hypervisor, a detection of a VM live migration is hardly possible without further monitoring. Thus, while on the one hand the invisible VM live migration has the advantage of ensuring a seamless VM utilization, on the other hand, it offers an opportunity for attackers to hide their actions. In this paper, we extend the work of (König and Steinmetz, 2011) and present the first taxonomy of VM live migration detection, giving two general distinct approaches and several detection approaches. We choose a hybrid external approach to detect VM live migration.

This paper is organized as follows: First, we discuss relevant literature in this field and give relevant background information. This includes virtualization and live migration as well as a description of a scenario of how VM theft can be realized. In Section 4, we show the detection taxonomy of VM live migration. Our hybrid external approach is discussed in Section 5. The corresponding prototypical implementation follows in Section 6. This prototype is an example of how to realize a monitoring system for a virtualized infrastructure. After that, our experimental results, which give rationales for preferring the hybrid approach, are presented and the applicability of our prototype is shown. Finally, we conclude and present suggestions for future improvements.

Fiebig S., Siebenhaar M., Gottron C. and Steinmetz R..
Detecting VM Live Migration using a Hybrid External Approach.
DOI: 10.5220/0004376904830488
In *Proceedings of the 3rd International Conference on Cloud Computing and Services Science* (CLOSER-2013), pages 483-488
ISBN: 978-989-8565-52-5

2 RELATED WORK

An overall summary of the different types of virtual migration exploitation is given in (Oberheide et al., 2008). Oberheide et al. describe three general attack vectors: Control Plane, Data Plane, and Migration Module. These differ in the way the migration process is compromised.

There are several publications covering the problem of malicious hypervisors as well as countermeasures to prevent attacks on them. (Xia et al., 2012) show the attack vector of rollback attacks, i.e., reestablishing a previous state of a VM without the user's awareness. This can lead to a state with open security holes. Xia et al. propose a so-called safe logging to prevent such rollback attacks on VMs. (Wang et al., 2010) describe a mechanism for securing the process of VM migration. The process is measured by using specific policies, e.g., allowed user roles that can migrate or allowed target hosts for the migration process. A general overview of cloud computing security is given by (Tsai et al., 2012).

The performance of VM live migration depends on a number of metrics. Two key parameters determining the speed of live migration are memory changes, namely the page dirty rate, and the network transfer speed, which both have a non-linear influence on the migration performance (Akoush et al., 2010).

(König and Steinmetz, 2011) show that ICMP ping is an appropriate mechanism to detect the live migration of VMs. The round-trip time of pings shows a higher average while migrating and at the end of the migration process packets might be lost. ICMP ping was also used as detection characteristic by (Nirschl, 2011). König et al. have discovered that the CPU load does not highly influence the ICMP ping round-trip times if it is made sure that as few memory changes as possible occur. This can be seen as corresponding to the fact that the VM live migration performance can be predicted without taking the CPU load into consideration (Akoush et al., 2010).

In previous publications, it has been shown that it is possible to use appropriate mechanisms to automatically detect changes in pings in order to detect a VM live migration (Gottron et al., 2012). Administrators can use this to monitor their server infrastructure and to apply countermeasures if needed, e.g., in case a malicious migration is happening.

To the best of our knowledge, in Section 4 the first taxonomy of different VM live migration detection approaches is given.

3 BACKGROUND

In this section, the VM live migration process is discussed and a simplified migration sequence for this paper is introduced. Additionally, an attacker scenario for VM theft is proposed to show how a malicious migration is realized.

The VM live migration process is described by (Clark et al., 2005) using six steps. For the understanding of this paper, the VM live migration can be divided in only three phases:

1. Memory Copy: Memory copy and mapping from source to target host while the VM runs on the source host. Detecting a migration in this phase is preferable, because only then the full migration can be prevented. Nevertheless, a detection is more difficult in this phase.
2. CPU Copy: Stopping the CPU and copy registers from source to target host. In this phase, the VM must be stopped, which causes detectable traces. These can be more easily tracked.
3. Switch: Removing the VM from the source host and starting the VM on the target host. After this phase, the VM runs on a new host. If the new host is not identical in software and hardware, these changes could be detected.

The three phases are the basis for the understanding of the detection taxonomy and experiments. Nevertheless, a detection of the live migration process even after phase 3 is important to prevent further damage.

In the following, a possible scenario for a VM theft is discussed. The steps for a successful attempt to steal a VM from a cloud infrastructure are as follows (see Figure 1 for a structure of the scenario):

1. Compromise the hypervisor migration module to gain access over the migration management.
2. Optional: Setup a corresponding subnet agent in the hypervisor network (as described in (Silvera et al., 2009)) to support migration across different subnets.
3. Migrate the VM to another host, that can be located in another subnet.

The attacker does not necessarily need to be an outsider, the cloud provider himself could be the attacker. He/she could try to optimize the resource utilization in his/her data center. In order to do so, he/she could move VMs out of overloaded European data centers to the USA without consent and knowledge of the SaaS (Software as a Service) provider and SaaS user. This becomes even worse when EU data protection directives must be obeyed. Given that, the cloud provider has the possibility to more or less hide all of

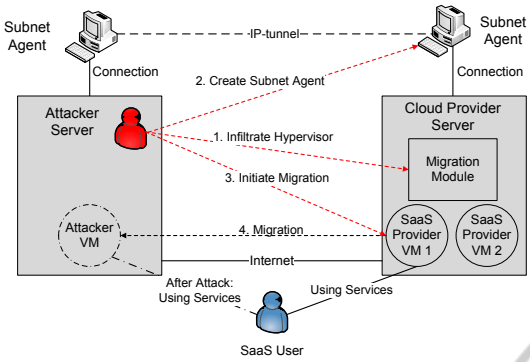

Figure 1: A scenario for VM theft to a different subnet (based on (Silvera et al., 2009)).

his/her actions. Compared to an attack coming from outside, this scenario is also easier to realize. With this slightly changed scenario, the hypervisor does not even need to be compromised. The cloud provider could simply present wrong information to the SaaS provider.

4 VM LIVE MIGRATION TAXONOMY

We now present the first taxonomy of VM live migration detection. There are two general monitoring approaches and four detection approaches that can be distinguished when trying to detect a live migration of VMs (see Figure 2):

- Internal Approach: When using an internal detection approach, a monitor inside of a VM is used to detect a malicious migration. These approaches include hypervisor detection, hardware detection, time-lag detection, and delay measurement. Internal approaches have the advantage of lower network usage and of not needing an extra monitoring server. They do not apply, however, if only a copy attack is performed to copy a consistent state of VMs leaving the original VM untouched.
  - Hypervisor Detection: Different types of hypervisors have different fingerprints (e.g., (Ferrie, 2006)) that can be used to detect a change or replacement of the hypervisor, e.g. another version.
  - Hardware Detection: Hardware benchmarks can be used to reveal a changed server configuration. They could for example reveal a changed CPU execution speed or link speed of the new physical host of the VM. This can be seen as the so-called VM footprint (Sonnek and Chandra, 2009). Two different footprints can be distinguished between the – dynamically changing, but location independent – amount of resources a VM wants to use (virtual footprint) and the amount of resources the VM actually uses on its physical host (physical footprint). Migrating a VM to a different host may lead to a changed physical footprint, which would then be detectable.
  - Time-lag Detection: Measure the NTP time to detect a lag in time. In phase 2 of the migration process, the VM is stopped for a certain amount of time and is then started on a new host again. Being stopped for a certain amount of time leads to a sudden time-lag, which can reveal a migration.
  - Delay Measurement: This is the original approach described in (König and Steinmetz, 2011) applying internal monitoring to defined network entities.
- External Approach: When using an external detection approach, mechanisms outside the supervised VMs are used to detect a malicious migration. The detection approaches are the same, but the specific realization differs. External approaches need an extra server to monitor VMs, but therefore not every VM needs to monitor itself.

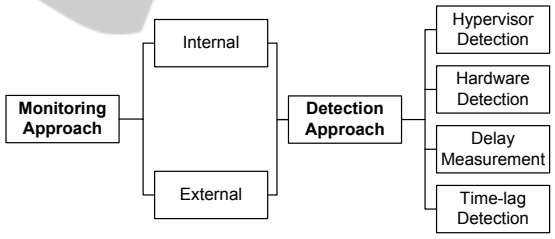

Figure 2: VM Live Migration Taxonomy that distinguishes between monitoring approach and detection approach. Former describes from where and latter how to monitor.

All these approaches allow either for a detection while live migration is in progress or only when the migration has been finished. Obviously, detection approaches that detect a change in the VM environment (e.g., hypervisor detection) are only able to detect migration after the last phase.

5 HYBRID EXTERNAL APPROACH

We use a combined approach of delay measurement with ICMP ping and time-lag measurement using NTP to detect a malicious VM live migration. Because the delay measurement using ICMP ping has

the disadvantage of needing a relative high monitoring frequency, the second approach is added. Thus, we can use a lower interval for the first approach and be nevertheless sure to detect a migration.

Based on the work of (König and Steinmetz, 2011) and similar to (Nirschl, 2011), we use the ICMP ping to detect the main characteristic features to assume a VM live migration. To do so, a heuristic based on the characteristic features of the ICMP ping round-trip times is used. These are increased average round-trip time, very high outliers mainly at the start of a migration and, additionally, unanswered pings in the second and third phase of migration.

As stated above, the main disadvantage is that the high monitoring interval can be counteracted with the time-lag measurement. However, in cases where a high monitoring interval is possible, the proposed heuristic is a possibility to abort the migration process before it has finished.

The NTP allows computers to synchronize their time with an external server. In (Broomhead et al., 2010), the problem that time does lag after a VM live migration as well as the possibility of using a clock mechanism called RADclock (Robust Absolute and Difference Clock) to prevent those time-lags, are discussed. For VM live migration detection, this time-lag is a valuable information to be able to detect a migration. After a migration, the time continues from the exact time as before the migration. This shift in time can be detected. This time converges to the proper time, as time jumps would not be appropriate for running programs that rely on a proper time progression. The slow converging of time allows a detection in a large time frame. This is even larger, the more time the second (CPU copy) and third phase (Switch) of the migration takes.

6 PROTOTYPE

Our approaches have been prototypically implemented as a software demo. This prototype is able to detect if a VM is migrated and can give the user a notification about the likelihood of a migration process. To detect migration, the proposed hybrid external approach is used as described above in Section 5.

The system interface is built as follows: The user has the ability to specify his/her VM servers, i.e., host name or IP address. In addition, the monitoring interval can be adjusted. A migration process needs at least the time to transfer the whole memory content once. One minute might not be sufficient for VMs with only small memory size and high bandwidth connection within a data center. Results are either presented in a detailed chart using the LiveGraph API[1] or a simplified view with only a status of the migration likelihood.

7 EXPERIMENTS

We used two different testbeds to perform our experiments. The first series was carried out in the German-Lab (G-Lab)[2] testbed. The installed hypervisor was Proxmox 1.9[3] with four SUN Fire X4150 servers. The servers were connected by a Cisco 4500 L3 series switch having a bandwidth of 1 Gbit/s. On Server 2 a NFS server was installed with a 32-bit Ubuntu 12.04. For the experiments, a similar Ubuntu version using the KVM virtualization was also installed on the test VMs. The migration VM was migrated between server 2 and 3. The file system for the migrating VM was on the NFS server, as the migration did not involve the file system.

The second series was carried out in a heterogeneous testbed using two desktop computers with Proxmox installed. The first server was an Intel Core 2 Duo with 2.13 GHz, 2 GB RAM and a Gigabit network card. The second server was an Intel Core 2 Duo with 2.53 GHz, 4 GB RAM and a Gigabit network card. A third computer was used as monitor. All were connected with a Netgear switch with a bandwidth of 100 MBit/s.

There are several metrics that are interesting for VM live migration. The metrics can be derived from the parameters influencing the migration performance as given in (Akoush et al., 2010). These are page dirty rate, link speed, VM memory size, and migration overhead. Together with the results from (König and Steinmetz, 2011), a useful set of metrics, besides general hardware differences for the hypervisor servers are memory, CPU use, and network bandwidth and distance.

The network bandwidth was restricted using WANem[4] as proxy. To create CPU load, a bash script with an endless loop was used, so that the page dirty rate could be minimized.

On the first testbed, we could replicate the results given in (König and Steinmetz, 2011). That is, the ICMP ping to a VM gives a characteristic round-trip times pattern while migrating. Some of the more interesting test results were obtained using the second testbed with heterogeneous setup. This is also

[1] http://www.live-graph.org/
[2] http://www.german-lab.de
[3] http://pve.proxmox.com/wiki/Main_Page
[4] http://wanem.sourceforge.net/

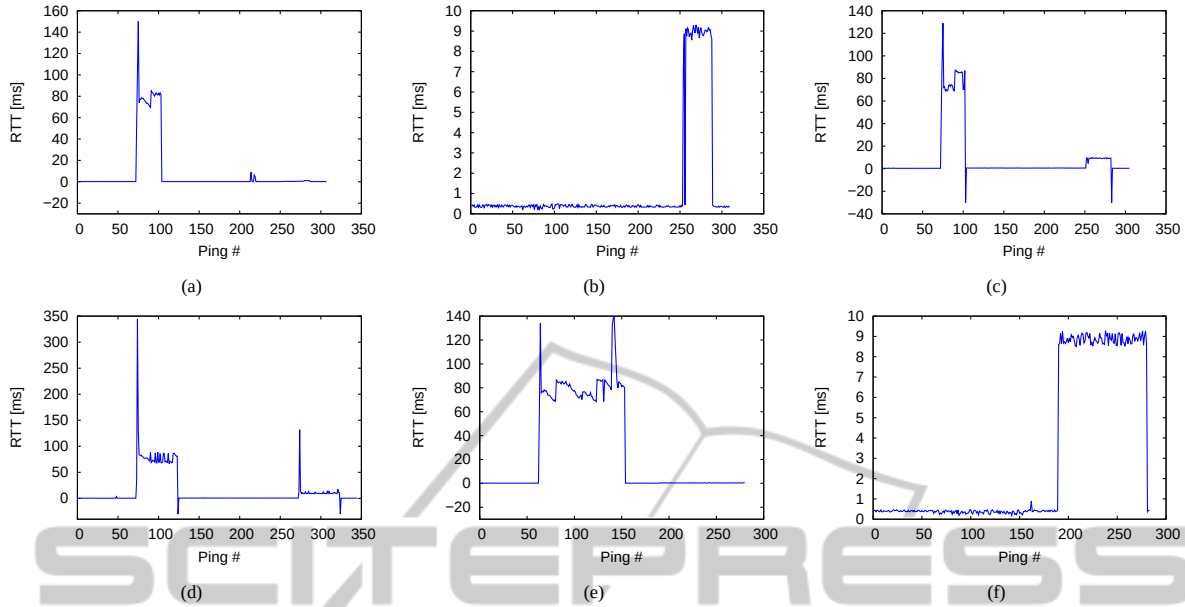

Figure 3: Measurements in the heterogeneous network topology. Measuring migration from first to second server and vice versa with ICMP ping every second. Negative values mark unanswered pings, not round-trip times. (a) Ping to first server. (b) Ping to second server. (c) Ping to VM. (d) CPU load 100 percent on VM, ping to VM. SCP transfer measurements in a similar way, copy from first to second server and vice versa. (e) Ping to first server. (f) Ping to second server.

more realistic in case of an attacker stealing VMs into his/her own subnet. The most interesting result measurements are shown in Figure 3. The first three Figures 3(a)–3(c) show the ICMP ping measurement for both servers involved in the migration and the VM itself. The migration is first performed from server 1 to server 2 and afterwards from server 2 to server 1. As can be seen, not only the measurement for the VM shows a characteristic trace, but also the hypervisor servers do so. The characteristics are significantly more pronounced when pinging the less powerful server (in terms of CPU power). In addition, the characteristics are almost only measurable for the server initiating the migration process, independent from the computing power.

We used the first testbed described and measured the occurring time course. As can be seen in Figure 4, the time-lags after a migration are over 100 ms. That time corresponds to phase 2 of the migration process that largely depends on the speed at which the last remaining VM resources are copied. The time converges after a synchronizing event with an NTP server. A low monitoring interval is possible, because the time to converge takes more than several minutes and only after an NTP update occurs.

In addition, we performed a simulation approach for the VM live migration. To do so, we used the second testbed and copied files using SCP (Secure CoPy). The characteristics while pinging the first and the second server are depicted in 3(e) and 3(f) for a

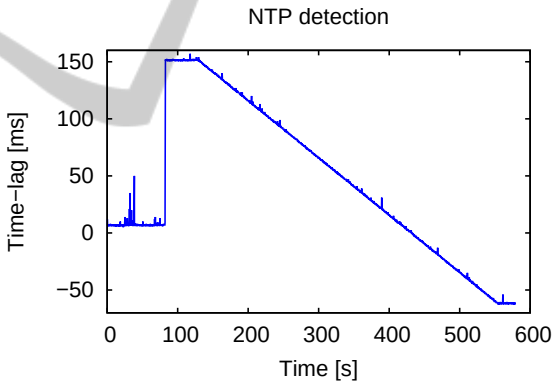

Figure 4: Measuring a time-lag when migrating a VM. This induces a time-lag of more than 100 ms.

file with a size of 2 GB. The SCP copy shows similar characteristics as a migration process. When compared to Figure 3(a) and 3(b), the round-trip times and general pattern are similar. Outliers at the beginning and an increase in average ping round-trip time can be seen. Only the unanswered pings are not present, as could be expected. This supports the need for a hybrid approach.

We used the prototype to test VM live migration detection in both testbeds. As expected, several early detections using the ICMP ping approach occurred. Using a high ping interval to detect unanswered pings as well as time-lag detection we were able to detect a migration in all of our simulations.

To sum up, the experiments produced the fol-

lowing interesting results: The ping measurement as shown by (König and Steinmetz, 2011) have not only characteristic patterns for the VM itself, but also for the hypervisor servers. Not only a migration, but also a copy process of a file, as performed with SCP, leaves a characteristic trace, and so will other network activities. Only the unanswered pings are unique in combination with the other two characteristics. If a high monitoring interval is not possible, our approach can detect a migration process after its execution.

8 CONCLUSIONS

In this paper, we presented the first taxonomy of VM live migration detection, showing different approaches that were categorized in two general groups. Only the delay measurement approach allows a detection during the first migration phase, i.e., during the copying of memory. This is the only way to prevent the migration process. Nevertheless, a detection after migration has finished is also very valuable in order to prevent further damage.

We proposed a hybrid external approach that comprises an ICMP ping detection and a time-lag detection using NTP. Our approach has been tested using a prototype on different testbed configurations to show the suitability of our hybrid external detection approach. This applies especially to the detection of migrations when using a low monitoring interval and having certainty in cases where ICMP ping characteristics have another origin than a VM live migration.

Further issues have to be addressed in future work. The mobile IP scenario has to be thoroughly tested and attacker scenarios have to be evaluated against the prototype. Especially, the specifics of different server types, e.g., video servers or webshop servers, could be used to create even more improved monitoring approaches. In specific terms, that means that mechanisms providing a correct classification for migration cases and non-migration cases even while using low monitoring intervals, are needed. In this field, tests with machine learning have been performed, but their applicability is very restricted. Finally, experiments in real cloud environments, e.g., Amazon EC2, have to be conducted. In this paper, we focused on the scenario of VM theft; however, the case of copying VM data without actually migrating the original VM leaves even less possibilities for detection. Therefore, more advanced detection approaches which also address this aspect must be developed.

REFERENCES

Akoush, S., Sohan, R., Rice, A., Moore, A., and Hopper, A. (2010). Predicting the Performance of Virtual Machine Migration. In *IEEE International Symposium on Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS'10)*, pages 37–46.

Broomhead, T., Cremean, L., Ridoux, J., and Veitch, D. (2010). Virtualize Everything but Time. In *Proceedings of the 9th Conference on Operating Systems Design and Implementation (OSDI'10)*, pages 1–6.

Clark, C., Fraser, K., Hand, S., Hansen, J. G., Jul, E., Limpach, C., Pratt, I., and Warfield, A. (2005). Live Migration of Virtual Machines. In *Proceedings of the 2nd Symposium on Networked Systems Design and Implementation (NSDI'05)*, pages 273–286.

Ferrie, P. (2006). Attacks on Virtual Machine Emulators. Symantec Security Response.

Gottron, C., Fiebig, S., König, A., Reinhardt, A., and Steinmetz, R. (2012). Visualizing the Migration Process of Virtual Machines. In *Proceedings of the 12th Euroview*.

König, A. and Steinmetz, R. (2011). Detecting Migration of Virtual Machines. In *Proceedings of the 11th Euroview*.

Nirschl, J. (2011). Virtualized guest live migration profiling and detection. Graduate Theses and Dissertations. Paper 12055.

Oberheide, J., Cooke, E., and Jahanian, F. (2008). Empirical Exploitation of Live Virtual Machine Migration. BlackHat DC convention.

Silvera, E., Sharaby, G., Lorenz, D., and Shapira, I. (2009). IP Mobility to Support Live Migration of Virtual Machines Across Subnets. In *Proceedings of SYSTOR 2009: The Israeli Experimental Systems Conference (SYSTOR'09)*, pages 13:1–13:10.

Sonnek, J. and Chandra, A. (2009). Virtual Putty: Reshaping the Physical Footprint of Virtual Machines. In *Proceedings of the 2009 conference on Hot topics in cloud computing (HotCloud'09)*.

Tsai, H.-Y., Siebenhaar, M., Miede, A., Huang, Y., and Steinmetz, R. (2012). Threat as a Service?: Virtualization's Impact on Cloud Security. *IT Professional*, 14(1):32–37.

Wang, W., Zhang, Y., Lin, B., Wu, X., and Miao, K. (2010). Secured and Reliable VM Migration in Personal Cloud. In *2nd International Conference on Computer Engineering and Technology (ICCET'10)*, pages 705–709.

Xia, Y., Liu, Y., Chen, H., and Zang, B. (2012). Defending against VM Rollback Attack. In *IEEE/IFIP 42nd International Conference on Dependable Systems and Networks Workshops (DSN-W'12)*, pages 1–5.