Detecting Migration of Virtual Machines

André König, Ralf Steinmetz
Multimedia Communications Lab (KOM), TU Darmstadt, Germany
{andre.koenig, ralf.steinmetz}@kom.tu-darmstadt.de

Motivation

Hardware virtualization, i.e., introducing an additional layer of abstraction (Hypervisor) between hardware and software is the basis for current applications such as cloud computing, energy efficient operation of communication networks, and high performance computing. Hypervisors like Proxmox [1] enable running multiple virtual machines in parallel, on the same physical machine. By decoupling hardware and software, virtual machines may be migrated online (during operation) between physical machines. From a security perspective, virtualization and migration of virtual machines offer new ways for realizing security mechanisms but also new attack vectors [2]–[4]. Besides attacks on the level of the hypervisor, in particular in the context of migration of virtual machines, attacks on the level of the communication network must be taken into consideration. Here, a precondition for both attacks and security mechanisms is detecting migration processes. Attack mechanisms may use this information to, e.g., launch denial of service attacks. Security mechanisms can decide whether an ongoing migration process is legitimate or, e.g., whether an unauthorized image of a virtual machine is created. For both, in addition to the detection of migration processes by the (possibly compromised) migrating hypervisor, a detection performed outside the migrating hypervisor is helpful.

The focus of our work is set on the foundations of detecting migration processes of virtual machines, outside the migrating hypervisor. We present first selected experiments performed with the German-Lab testbed [5]. Results from different scenarios of an online migration of a Linux machine by the Proxmox Hypervisor show that the roundtrip time of ICMP packets is a promising metric for detecting migration processes. In future steps, we will validate the results in real-world cloud-based systems.

Related Work

Related work on security in virtualized computing environments dates back to 1976. In [6], Attanasio et al. present a security analysis of the IBM VM/370 hypervisor. Gold et al. develop extensions for securing VM/370 in [7]. A survey on attack vectors for contemporary hypervisors is presented by Ferrie in [8].

Hypervisor-based malware (virtual machine-based root kits) such as SubVirt [9] uses virtualization techniques to insert a malicious hypervisor between hardware and operating system, e.g., to prepare further attacks. Opposed to this are approaches such as GuardHype [10], that, acting as 'hypervisor for hypervisors', control access of a hypervisor to the physical hardware.

Approaches for determining whether a system is running on a virtual machine, such as the one described by Quist et al. in [11], are, in general, based on analyzing interrupt or memory tables. These differ depending on whether an operating system is running on a virtual machine or directly on the physical hardware. This way, detecting a migration of a system to a virtual machine is possible ex-post, if it was running directly on physical hardware before. A migration of an already virtualized machine can not be detected.

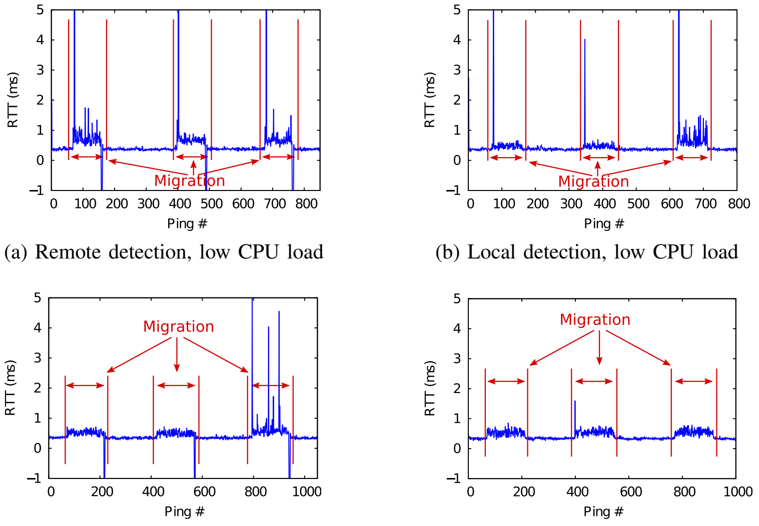
Regarding the security of migration processes, Oberheide et al. categorize possible attacks and demonstrate the vulnerability at the example of a man-in-the-middle attack in [12]. To the best of our knowledge, directly related work on the remote of ongoing migration processes outside the migrating hypervisor does not exist.

Detecting Migration Processes

We conducted our experiments in the German-Lab testbed. We set up a Proxmox 1.8 cluster on 4 identically configured SUN Fire X4150 servers. The servers are connected by a Cisco 4500 L3 series switch at a bandwidth of 1 Gb/s. On Server 1, a 32-bit Ubuntu 11.04 virtual machine was installed as NFS server. During the experiments, a second virtual machine running Ubuntu was migrated between Servers 2 and 3. The virtual machine was configured with 1 CPU core, 512 MB main memory, 8 GB hard disk located on the NFS server, and an rtl8139-based network interface with direct (TAP) access to the network. The migration process was initiated manually via the Proxmox web interface. Before, during, and after the migration process, the roundtrip time of ICMP packets between the migrated virtual machine and a third, identically configured virtual machine installed on Server 4 was measured. For this, ICMP packets with a size of 64 byte were sent at an interval of 0.1 seconds.

For now, we focus on examining whether ICMP packets should be sent from or to the machine that is being migrated, i.e., on whether the detection should be performed locally, by the machine being migrated itself, or remotely. We further evaluate the effects of CPU load on the detectability of a migration process.

a) Remote detection, low CPU load: Figure 1a shows the results for the experiment in which we sent ICMP packets from the virtual machine on Server 4 to the virtual machine being migrated between Servers 1 and 2. The CPU load on the machine being migrated was not increased artificially. In this



(a) Remote detection, low CPU load

(b) Local detection, low CPU load

(c) Remote detection, high CPU load

(d) Local detection, high CPU load

Fig. 1. ICMP roundtrip times during online migration processes of a Ubuntu 11.04 virtual machine in Proxmox

setup, we observe an increased roundtrip time for the whole duration of the migration process. Additionally, we observe a peak roundtrip time at the beginning and packet loss at the end of the migration process. For reasons of presentation, the peak is not represented fully and packet loss is represented as a negative roundtrip time in Figure 1a. The packet loss at the end of the migration phase is caused by the virtual machine's CPU being stopped while its registers are transferred to the target machine. Explaining the peak roundtrip time at the beginning of the migration process is part of our future work.

b) Local detection, low CPU load: The results for the experiment in which we sent ICMP packets from the virtual machine being migrated between Servers 1 and 2 to the virtual machine on Server 4 are shown in Figure 1b. Again, the CPU load on the machine being migrated was not increased artificially. As for the previous setup, we observe an increased roundtrip time during the whole migration process with a peak at the beginning. In contrast to the previous setup, we did not observe packet loss at the end of the migration process. This is because the CPU operation of the machine being migrated and, thus, sending ICMP packets is stopped while the CPU operation is transferred to the target machine. Therefore, the packet loss that is characteristic for the end of the migration process when monitored remotely does not occur.

c) Remote detection, high CPU load: Figure 1d shows the effect of a high CPU load on the roundtrip time of ICMP packets sent to the machine being migrated, i.e., on a remotely monitored migration process. The CPU load was increased by piping /dev/urandom to an MD5 message digest generator. Again, we observe an increased roundtrip time during the migration process and the characteristic packet loss at the end of the migration process. However, the increased CPU load affects the peak of the roundtrip time that we observed in the previous experiments without artificially increased CPU load. Scrutinizing the particular reason for this effect is part of our future work.

d) Local detection, high CPU load: The roundtrip time of ICMP packets sent from the machine being migrated, i.e., for a locally monitored migration process in combination with an artificially increased CPU load of the machine being migrated is shown in Figure 1d. Although we still observe an increased roundtrip time during the migration process, we neither observe the peak at the beginning, nor the packet loss at the end of the migration process. From this, we conclude that the effects of both local vs. remote monitoring of the migration process and CPU load are characteristic and reproducible. Altogether, monitoring the roundtrip time seems to be a promising approach towards detecting migration processes of virtual machines. In our future work, we will identify further factors that affect the detectability of migration processes and validate our results in real-world cloud-based systems.

References

[1] Proxmox Wiki, *http://pve.proxmox.com/wiki/Main_Page*, 2011.

[2] T. Garfinkel et al., "When Virtual is Harder than Real: Security Challenges in Virtual Machine Based Computing Environments," *Proc. of HotOS '05*.

[3] M. Price, "The Paradox of Security in Virtual Environments," *IEEE Computer Magazine*, vol. 41, pp. 22 – 28, 2008.

[4] D. Hyde, "A Survey on the Security of Virtual Machines," Dept. of Comp. Science, Washington Univ. in St. Louis, Tech. Rep., 2009.

[5] G-Lab Project Homepage, *http://www.german-lab.de*, 2011.

[6] C. R. Attanasio et al., "Penetrating an operating system: a study of VM/370 integrity," *IBM Systems Journal*, vol. 15, pp. 102 – 116, 1976.

[7] B. D. Gold et al., "A security retrofit of VM/370," *Proc. of National Computer Conference*, 1979.

[8] P. Ferrie, "Attacks on Virtual Machine Emulators," *Proc. of AVAR '06*.

[9] S. T. King et al., "SubVirt: Implementing malware with virtual machines," *Proc. of S&P '06)*.

[10] M. Carbone et al., "Taming Virtualization," *IEEE Journal on Security and Privacy*, vol. 6, pp. 65 – 67, 2008.

[11] D. Quist et al., "Further Down the VM Spiral - Detection of full and partial emulation for IA-32 virtual machines," http://www.offensivecomputing.net/dc14/furthur_down_the_vm_spiral.pdf.

[12] J. Oberheide et al., "Empirical Exploitation of Live Virtual Machine Migration," *Proc. of Black Hat DC*, 2008.