BOYER CAROLE
TRÉCHEREL DIDIER

NET5039

VIRTUAL MACHINE ESCAPE

Table of Contents

Table of Figures

Introduction

Virtualization is known for a long time — since the 1960s — but is only experiencing a revival for the last decade. It has gone from a theoretic and educational tool to a really efficient solution for companies that want to save on both space and cost for their IT resources. Indeed, nowadays, most computers don't use all the resources they have. In a traditional configuration, an enterprise may have several computers: one for each users and services. This could make a lot of machines, and according to security policies, some services should be ran on individual computer, increasing the IT infrastructure even more. Using the virtualization concept, the enterprise is able to run several virtualized machines on a single computer, and, thereby, reduces the cost in hardware components and cooling systems. These virtualized machines, known as *virtual machines* (VMs) or *guest machines* are isolated from one another. Each one possesses its own operating system and configuration, and virtualization gives them the illusion that they are running singly and directly on a physical machine. However, they are sharing the hardware of the same computer, known as the *host machine*. The *hypervisor*, or *VM Monitor* (VMM) is the name of the layer between the host hardware and the guest machines. As it is working on the most privileged level, it allows the VMs to share the same resources (CPU, GPU, RAM, …) so they are able to access hardware resources and to run applications without interfering.

From a security point of view, virtualization improves security by running several services or applications on different virtual machines. Given the isolation feature provided, a compromised service cannot affect another that is not running in the same virtual machine. Moreover, in the eventuality that a service doesn't behave correctly, the snapshot capability of virtualization allows the administrator to quickly bring the VM to a previous estate enhancing the availability of the service. The administrator could also save a snapshot of an infected machine and proceed to do forensics analysis. However, a guest machine is still vulnerable to the same attacks that can affect an identical machine but running in a non-virtualized environment. As a further matter, the threat is even more important since the victim is sharing its hardware with other VMs: if a malicious code manages to break-out its VM, it could spread out to others or to the host. Thus, VMs should be isolated enough to not interact together and one's vulnerabilities should not affect neither the others nor the host machine.

The purpose of this document is to make a state of the art of the actual virtual machine escape techniques. Basically, VM escape describes the process of breaking-out of a virtual machine and interacting with the host, or another guest machine. There are 3 points that we will study in the present document:

- The concept of virtualization and description of virtual machines;
- Several attacks that allow to break-out from a virtual machine;
- Solutions for securing virtualization.

In an attempt to spell out these points, we will first describe what virtualization is. Trough the concept and its history, we will explain what are the security issues. This will bring us

to explain the different attacks that threaten VMs security and provide a way for breaking-out, depending of its architecture. Finally, we will conclude by showing secure solutions to this.

1. Virtualization

1.1. Definition

Virtualization is the concept of sharing one computer hardware to create the illusion of several machines that run simultaneously. As these machines do not have their own physical hardware, they are known as virtual machine (VM). In a nutshell, a virtual machine is a software-based emulation of a computer. All the virtual machines are sharing the resources of a single computer, called the host machine. They all have their own operating system and configuration, and are isolated from one another. They don't even know they are virtualized, following the concept of vertical isolation. This means that every application can only interact or interfere with another application whether it is running on the same virtual machine. The software emulating the VMs has to simulate the hardware and software resources, allowing the execution of programs in the same conditions as on a single computer. This component, known as the hypervisor is able to emulate several virtual machines simultaneously.

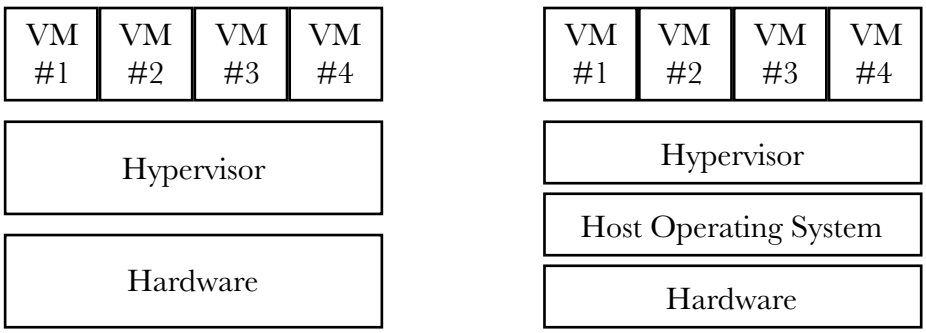

FIGURE 1 — TYPE 1 AND TYPE 2 HYPERVISORS

The hypervisor, or VM monitor, is the name of the underlying component that makes the virtualization possible. It processes to share the resources between the virtual machines. There's 2 most important types of hypervisor:

- **Type 1: *native*, or *bare metal***
  The type 1 hypervisor is installed above the physical hardware and can access it *directly*. It doesn't require a full OS in order to work. As it runs directly on the system hardware it can easily monitor the virtual machine access attempts.
  (e.g.: XEN, ORACLE VM, VMWARE ESX, KVM, …)

- **Type 2: *hosted***
  The type 2 hypervisor is an *additional* layer that is installed above the host running operating system. In this case, it is the host operating system that has direct access to the system hardware. The hypervisor relies on it to get access too, and then, monitor the virtual machines.
  (e.g.: QEMU, VIRTUALBOX, VMWARE SERVER, VIRTUAL PC, ...)

Regardless of the implementation, virtual machines and their operating systems are typically unaware of which type of hypervisor is implemented, as they interact only with the hypervisor itself. There are pros and cons for both types of hypervisors. For example, the type 2 hypervisor can be seen as an advantage as it offers hardware driver compatibility. But on the other hand, there is all the security issues exposed by the host operating system — a type 2 hypervisor is as secure as the operating system it relies on.

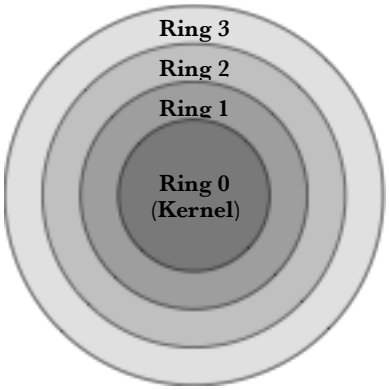

FIGURE 2 — PRIVILEGE RINGS FOR THE X86 PROCESSOR

To fully understand the hypervisor role and its security issues, it is important to understand the privilege levels for CPU. It is often called as *protection rings* and consists of several layers of privileges, from the most privileged — ring 0 — to the least. A software running on ring 0 has the total control of the hardware. Hypervisors runs on ring 0 and all the virtual machines run on a less privileged ring, thus ring 1. This result in a horizontal isolation.

1.2. History

In 1963, the MASSACHUSETTS INSTITUTE OF TECHNOLOGY (MIT) created the MAC project. At first it stood for Mathematics and Computation before being renamed MULTIPLE ACCESS COMPUTER. This project needed a multi-users system, but IBM turned down its demand, considering that the demand for time-sharing system wasn't big enough. GENERAL ELECTRIC, though, was interested in constructing a time-sharing computer, and was chosen by the MIT as their supplier. IBM realized its loss when BELL LABS needed a similar system too and began to do researches in this domain. Consequently, IBM designed the CP-40, a research precursor to the CP-67 and the CP/CMS family, to demonstrate its capability for supporting time-sharing users. Although time-sharing computers were based on the share of the system resources between the users (e.g., MULTICS), the particularity of CP-40 is that it ran multiple instances of users' operating system. Another example is MULTICS, an operating system, one of the result of the MAC project. MULTICS was later taken by BELL LABS and became UNIX in 1972. Although it's not an example of application virtualization, UNIX introduced great portability for applications.

This was one of the first step to software portability which evolved to software virtualization. In 1987, INSIGNIA SOLUTIONS developed SOFTPC, an emulator allowing to run DOS applications on UNIX environment. They improved their software emulator, and, in 1989, it

was able to run WINDOWS applications too — and not just DOS'. In 1998, VMWARE was created and released in 2001 two products: ESX SERVER, a type 1 hypervisor, and GSX SERVER a type 2 hypervisor.

1.3. Benefits and threats of security by virtualization

The CIA triad is one of the core principles for information security, and is based on 3 criterions:

- **Confidentiality**
  It refers to the ability to manage information access from legitimate users and prevent it from unauthorized ones. Based on data privacy, it can be described as a set of rules.

- **Integrity**
  It refers to trustworthiness of information, and ensures that information hasn't been modified whether by accident or by any malicious activity. It can be seen as reliability, and the proof that the information hasn't been sent by an impostor.

- **Availability**
  It refers to the availability of resources. The information system should be available when required. Indeed, an information system with poor availability can be as worse as no one at all.

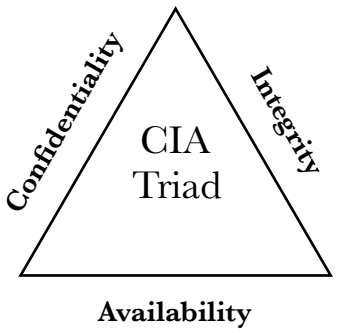

FIGURE 3 — THE CIA TRIAD

As it has been mentioned before, virtualization is a great solution for companies that want to reduce IT infrastructures cost and space and enhance security. From a security point of view, the most significant benefit of virtualization holds in both the isolation and the snapshot capability of virtual machines that can be used to provide higher security guarantees. Furthermore virtualization can also be very useful in many security search project (forensic analysis, patch testing, …).

In virtualization, the isolation is horizontal and vertical: each layer is separated from the others, and virtual machines are isolated from one another. Thus, a compromised or unstable machine won't affect the stability of the others. Moreover this means that security services can't be deactivated on all VMs: if a malware deactivates security services on an infected machine, these services will still be running on the others. The snapshot feature is a great asset too. VM snapshot is a copy of a virtual machine at a given point of time. This feature can preserve the stability of a virtual machine: if it becomes compromised, the administrator can easily get the system back to a previous state, before the supposed infection. Without virtualization, it would be much more difficult, and could possibly have terrible consequences for a company.

In security projects, virtualization can be very efficient in order to avoid damaging a machine. In malicious code research, researchers can analyze the impacts of code samples. Moreover, they can test it on several machines running different operating systems and/or configuration. Based on the isolation concept, the host machine won't be affected by the malware and the virtual machine can be easily destroyed or backed-up to a previous state. It can also be used as honeypot for attackers in order to detect any suspicious activity, allowing to foresee an eventual attack, or as a sandbox for the testing of new patches.

But virtual machines security has threats on several levels. An adversary could try to escape from a virtual machine to attack the hosting machine, or another virtual machine. Thus, the isolation concept of the virtual machine is very important. As the component interfacing the hardware and the virtual machines, the hypervisor is a critical point for security. Supposing that an attacker gain access to the hypervisor, and can control it, he will be able to control the entire system, including the virtual machines and the hardware's resources. A large internet service provider, Vaserv.com, was attacked by hackers using a zero-day vulnerability in the hypervisor of its virtualization application, HyperVM. The hackers had the ability to execute sensitive Unix commands, including 'rm -rf'. 100'000 websites were destroyed.

In this section, we have spell out what virtualization is, and how it can both improve and threaten security. As a matter of fact, we will focus in the following section on the different techniques an attacker could use to compromise a virtual environment. In order to do so, we will see techniques an attacker could use to detect the presence of an hypervisor and identify it. Then, we will explain how the attackers could disrupt the environment by performing a DoS attack or by escaping his virtual machine.

2. Attacks

A hypervisor, which is an additional architectural layer, can suffer from security vulnerabilities. These vulnerabilities allow attackers to escape from their walled environment to perform attacks against either another Virtual Machine, the hypervisor or the host operating system. So new attacks scenarios have emerged from virtualization.

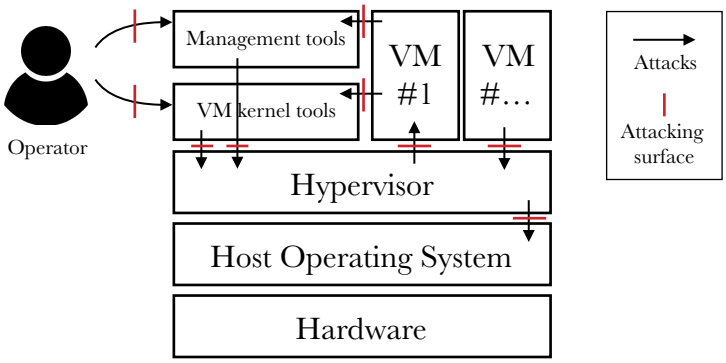

FIGURE 4 — ATTACK SURFACES FOR A TYPE 2 HYPERVISOR

The figure. 4 shows all the different attack surfaces one can find in a virtual environment using a type 2 hypervisor. The attacks are alike for a type 1 hypervisor. The attacks stemming from the hypervisor security vulnerabilities and affecting virtualized systems can be divided into three categories: detection of a hypervisor presence, denial of service and virtual machine escape.

2.1. Detection of a hypervisor presence

In theory, virtualization technologies (except paravirtualization) have to provide an identical copy of a real system to the guest. However, in practice, this is not totally true. There are several techniques that can help detect if there is a hypervisor running underneath the OS. Therefore, an attacker would be able to know whether his targeted system is virtualized and then act accordingly. We can take the example of virtual machines used to analyze malicious software. If the malicious code is able to detect that it is executed within a virtual machine, it can hide its impact by changing its behavior or it can simply refuse to run. That is an important issue in malware analysis.

2.1.1. Detection of a hypervisor presence

Detecting a hypervisor in a full-virtualized environment

There are two main categories of methods to detect the presence of a hypervisor in a full-virtualized environment.

The most commonly used methods rely on external timing. More precisely, they use the fact that executing certain instructions repeatedly will take more time within a hypervisor environment than without. For example, the time to access to data structures before and after triggering hypervisor events can be measured. However this timing method require a baseline comparison. As a matter of fact, the same measures have to be done on the same, known-to-be-safe system. That is to say, the same number of iterations of the same instruction has to be run on the same machine, but without the presence of the hypervisor.

There is another category of methods, discovered recently, that does not require preliminary baseline measurements. This new approach relies on a different kind of timing. It requires measuring and comparing the time for fetching information from memory before and after executing hypervisor-sensitive instructions. If they are different, there is a good chance that a hypervisor is lying underneath. An illustration of this approach can be described by using Translation Lookaside Buffer (TLBs). TLBs contain virtual addresses with the corresponding absolute addresses in physical memory that have been most recently used. This new technique starts with filling with known data the TLBs by accessing a series of present pages. Then, if a hypervisor is present, calling a hypervisor-sensitive instruction (such as CPUID, that does not affect memory) will lead to the cleaning of at least one portion of the TLBs. After the instruction is executed, we can access again to each of the pages that should be in the TLBs. By measuring this access time, we can see if it matches that of a new page or a cached page. If the latter case, there is a pretty good chance that a hypervisor is lying underneath.

Detecting a hypervisor in an emulated environment

There are several methods to detect the presence of emulators such as VMware Workstation or Virtual PC.

- First, we can look for artifacts in processes, the file system or the registry.
- Another method is to seek VM-specific hardware devices. As a matter of fact, virtual machine emulators usually use VM-specific hardware, such as hard disks whose device names are constant or network cards whose MAC addresses are within a predefined range.
- A last technique is to search for VM-specific processor instructions and features. For example we can try to identify particular guest-to-host communication channels.

2.1.2. Identifying the hypervisor

Previously we have seen methods to detect the presence of a hypervisor. Nevertheless, it can be really interesting for an attacker to be able to identify with precision which hypervisor is used underneath. There are techniques that an attacker can perform to do so. We will focus here on identifying emulators (such as VMware Workstation). The idea is to use specific instructions that are not handled by some hypervisors the same way as they are on real system. These instructions will trigger hypervisor-specific exceptions or, on the contrary will lead to the absence of exception whereas some would have been raised on a real system. In this paper we will present one technique of identifying a specific hypervisor, which is called RedPill.

RedPill

RedPill is a method that permits the detection of the hypervisor by using almost one CPU instruction. It has been discovered by Joanna Rutkowska in 2004. Here is the code written in C used in this method of detection:

```
int swallow_redpill() {
    unsigned char m[2+4], rpill[] = "\x0f\x01\x0d\x00\x00\x00\x00\xc3";
    *((unsigned*)&rpill[3]) = (unsigned)m;
    ((void(*)())&rpill)();
    return (m[5]>0xd0) ? 1 : 0;
}
```

Figure 5 — The RedPill's CPU instruction

The most important part of this code is the SIDT instruction (encoded as `0F010D[addr]`). This instruction is an instruction in assembly language that moves in a memory location the content of the IDTR (Interrupt Descriptor Table Register). The IDTR contains the physical memory address of the IDT (Interrupt Descriptor Table). This SIDT instruction is really

interesting and useful because it can be executed in non-privileged mode (ring 3) but it returns the content of the sensitive register. As there are at least two OS running (the host and the guest OS), the hypervisor has to relocate the guest IDTR in a safe place. The process will get the relocated address of the IDT table. JOANNA RUTKOWSKA observed that this relocated address was different on VMWARE WORKSTATION 4 and on VIRTUAL PC 2004, both running on WINDOWS XP host OS. Therefore, this method can be used to tell the difference between distinct emulators.

If an hypervisor can be exactly identified, an attacker will be able to use the distinctive vulnerabilities of this particular hypervisor to adapt his attack scenario. Besides, with these methods, the creation of viruses targeting only the systems where a particular type of hypervisor is running on is now possible.

We can notice that the methods described previously can identify popular hypervisors that usually do not try to hide themselves.

2.2. Denial of Service

Bad configuration or design flaws within the hypervisor can be the source of Denial of Service (DOS) attacks. A DOS attack occurs when one virtual machine uses all the computing capacities of the real host. This extreme resource consumption can prevent the other virtual machines sharing the same host machine to run correctly. Besides, it can cause the hypervisor to exit. In this section, we will focus on type 1 hypervisors. We will present the vulnerabilities that exist in popular open-source hypervisors (XEN and KVM) and in closed-source hypervisor VMWARE that can lead to a Denial of Service attack.

2.2.1. Vulnerabilities in Type 1 open-source hypervisors

Here, we focus on the two most popular open-source hypervisors XEN and KVM. We can notice that Denial of Service is, by far, the most frequent risk on XEN and KVM hypervisors.

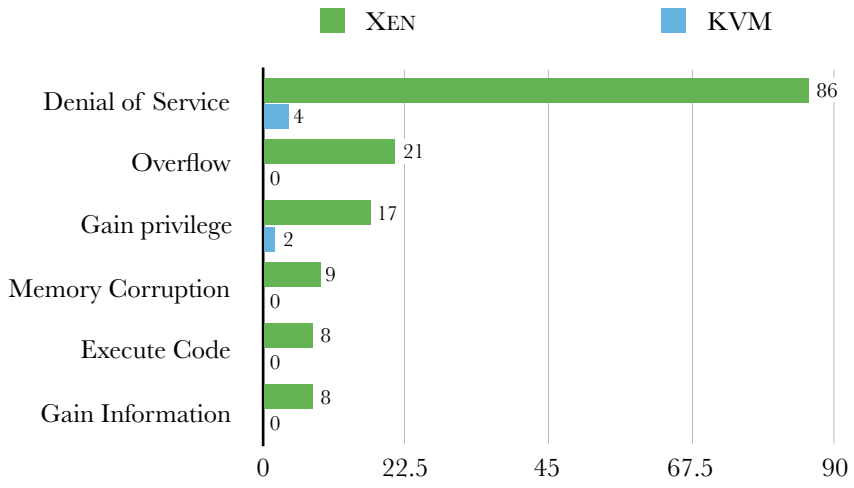

FIGURE 6 — XEN AND KVM VULNERABILITIES

XEN

A Denial of Service attack due to the presence of vulnerability in a XEN hypervisor can affect its availability partially or completely, lead to its crash, or worst, the crash of the host machine. In all cases, every virtual machine sharing the same hypervisor will not be able to run properly. The first known vulnerability that can cause DOS was found only in 2009. Then the number of vulnerabilities found increased between 2012 and 2013. However, it has fallen between 2013 and 2014.

Year	2008	2009	2010	2011	2012	2013	2014
# of vulnerabilities	0	1	0	0	31	30	24

Here are some examples of XEN vulnerabilities that can cause major damage whether to the hypervisor or the host itself. This is a non-exhaustive list.

- A vulnerability related to the grant table version switch list was discovered in 2012 (CVE-2012-5510). When the hypervisor frees the status pages of a guest by downgrading its grant table version, the freeing is not complete. As a matter of fact, the pages are not removed from the domain tracking list. This vulnerability can bring about list corruption and can eventually cause the hypervisor to exit. All XEN versions since 4.0 are vulnerable. However, Version 3.4 and earlier are not.

- Another vulnerability was found in 2012 in relation with the Transcend Memory (TMEM) (CVE-2012-6031). The TMEM allows the kernel to use memory that it cannot enumerate or directly address. Local guest OS users can use a specific function in the TMEM (`do_tmem_get` function) in order to cause a denial of service leading to the host crash.

- A vulnerability related to the network backend driver (network driver in the host side) was revealed in 2014 (CVE-2014-2580). When using certain version of LINUX, the driver can enable local guest administrators to perform a DOS attack thanks to a malformed packet. This can be responsible for the host crash.

- Another vulnerability was found in 2014 on XEN hypervisor that can make a DOS attack happen (CVE-2011-1166). When local guests using paravirtualization specify user mode execution without specifying user-mode pagetables, they can prompt the host to crash. We can mention that the complexity of this attack is low.

KVM

Until now, only four vulnerabilities causing DOS on KVM hypervisors have been revealed. Three of them were found in 2010 and one in 2012. We can give the example of the vulnerability discovered in 2012 (CVE-2011-4622). It was found that local host users could perform a DOS attack by starting a timer. As a matter of fact, a certain function in KVM (`create_pit_timer` function) does not run correctly when requests are interrupted by Programmable Interval Timer (PIT) without interrupt controller being available, resulting in a NULL pointer dereference.

2.2.2. Vulnerabilities in type 1 closed-source hypervisor VMWare ESX

There are very few known vulnerabilities in VMWare ESX hypervisor that can trigger a Denial of Service attack. We will present two of them that affect in an important way the running of the hypervisor or the host machine.

- A vulnerability related to the VMWare Descheduled Time Accounting driver was discovered in 2009 (CVE-2009-1805). When the Desceduled Time Accounting Service is not running, virtual machines users on Windows are able to perform a Denial of Service attack that can lead to the complete disruption of the service. This attack is possible on VMware ESX 3.0.2, 3.0.3 and 3.5. However this attack is quite difficult to perform.

- A vulnerability related to the Virtual Ethernet Module (VEM) was discovered in 2011 (CVE-2011-0355). The Cisco Nexus 1000V VEM used in VMware ESX 4.0 and 4.1 has an issue with dropped packets that can allow virtual machines users to perform a DoS attack on the hypervisor by sending a packet over an access vEthernet port. This can cause the total crash of the ESX host operating system. Unlike the previous one, the complexity of this attack is low.

As we have seen in this section, vulnerabilities causing Denial of Service have been discovered very recently and are more present in open-source hypervisors than in closed-source hypervisors. However in both cases, Denial of Service is the most spread attack on hypervisors nowadays.

2.3. Virtual Machine Escape

Virtual Machine Escape (VME) is the most complex, challenging and dangerous category of attack. It is the most serious threat to virtualization security. This type of attacks exploits vulnerabilities in the source code or the design of the hypervisor. During a VME attack, the attacker runs code inside a VM that allows it to break out and interact directly with the hypervisor. This could give the attacker access to the host operating system and all other virtual machines running on that host. Eventually, the attacker would be able to take completely over the whole system. For example, a virtual machine escape can allow the attacker to gain access to the memory of the host system or of the other guest systems and then, read, write or execute its content. Another example is that the attacker would be able to perform any administrative task such as virtual machine creation, deletion resource, quota modification and more. He could also edit the privileges assigned to its specific virtual machine. Thus, we can see why VME is the most dangerous threat that can exist in a virtual environment. However, only few vulnerabilities providing the basis for this attack have been reported so far. In this section we will, once again, tackle first vulnerabilities in open-source hypervisors such as Xen and KVM and then in closed-source hypervisors such as VMware.

2.3.1. Vulnerabilities in the open-source hypervisor XEN

As far, the most notable virtual machine escape performed on XEN hypervisors was released in 2012 by a French information security company called VUPEN SECURITY. This attack exploits a critical memory corruption vulnerability reported the same year on XEN hypervisors (CVE-2012-0217). This flaw is based on an issue affecting the system call functionality in XEN in some specific situation involving non canonical addresses. It concerns XEN 4.1.2 and earlier when running on an INTEL processor. This vulnerability can allow a local guest attacker to escape from his closed virtual environment and execute code on the host system with the highest privileges. The attacker would then have free access to hardware and would be able to handle unprivileged domains.

XEN developers have patched the vulnerability since June 2012. Now, XEN version 4.1.3 and later do not have that critical flaw anymore. MICROSOFT, RED HAT, ORACLE and other virtualization vendors affected by this vulnerability have also fixed their respective products.

2.3.2. Vulnerabilities in the closed-source hypervisor VMWARE

In this section we will present three notable research works that have been able to reveal vulnerabilities leading to virtual machine escape with VMWARE hypervisors.

The first vulnerability that can be responsible for VM escape was reported in 2007 by IDEFENSE. The issue is related to Shared Folders functionality (shared folders allow easier exchange of data between an OS running on the virtual machine and the host OS) in VMWARE WORKSTATION. As WORKSTATION does not interpret filenames properly, an attacker would be able to write files from inside a guest machine to the host machine with the privileges of the user who is running VMWARE WORKSTATION on the host.

In the SANSFIRE conference in 2007 TOM LISTON and ED SKOUDIS, two security researchers presented several tools that could be used to perform a VM escape also on VMWARE WORKSTATION. These tools are:

- VMCHAT: It is a chat software that allows guests to communicate with each other, or a guest to communicate with the host using the VMWARE hypervisor communication channel. This program does not require any special code to be installed.
- VMCAT: This tool can be used to tunnel a command shell between hosts and guests.
- VM DRAG-N-SPLOIT: This tool allows a user to send a command shell from the host to the guest.
- VMFTP: This tool is a working exploit for the IDEFENCE Shared Folders vulnerability presented previously. A user on any guest machine, with any level of privilege, would be able to read and write to the host as long as Shared Folders is enabled and at least one folder is shared with the guest virtual machine.

In 2008, researchers working at a company called CORE SECURITY released vulnerability in certain versions of VMWARE WORKSTATION, ACE, and PLAYER. Using this flaw, a local guest attacker would be able to exploit the VMWARE Shared Folders functionality to read or write in any area of the host machine lying underneath.

The most notable discovery about VM escape was disclosed in 2009. KOSTYA KORTCHINSKY, a researcher at IMMUNITY presented a tool called CLOUDBURST that can be used to perform a VM escape attack on VMWARE hypervisors. CLOUDBURST exploits a flaw on VMWARE discovered the same year. This vulnerability is related to a virtual machine display driver buffer overflow. Using the CLOUDBURST tool, an attacker could execute code from within a guest machine on the underlying host. This is the most serious escape scenario known so far.

Virtual Machine Escape attacks are really rare and much more difficult and complex to perform compared to detection or Denial of Service presented previously. And as we have seen, this type of attack is only at the stage of research today. However it is the most critical and dangerous threat in a virtual environment. In conclusion, in order to ensure security in a virtual environment, an administrator should add an extra level of security. By doing so, he will prevent his system from being attacked. Thereby, In the following section, we will present secure solutions for virtualized environment.

3. Protections

There is several ways to prevent VM escape. In this section we will see solutions to avoid this. As the hypervisor seems to be the main channel for attacks, we will first see whether we can find secure hypervisors. Then, we will present solutions to enhance security.

3.1. Secure Hypervisors

3.1.1. TERRA

TERRA is an architecture for virtualization that provides stronger security. TERRA is based the hypervisor named TRUSTED VIRTUAL MONITOR MACHINE (TVMM). TVMM has a lot of additional features and strong security measures.

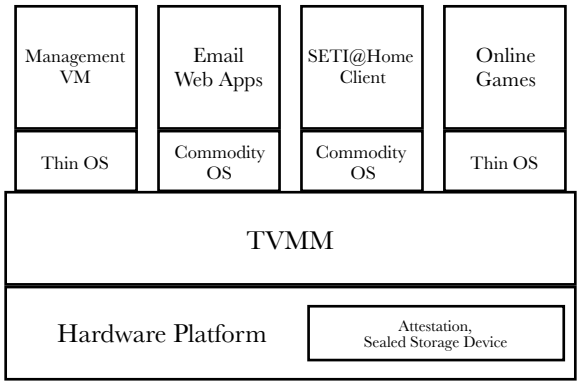

FIGURE 7 — TERRA ARCHITECTURE

TERRA allows several VM to run on the physical machine, and acts like a trusted party for the authentication of remote users. It has two levels of abstraction: OPENBOX VM and CLOSEDBOX VM. The OPENBOX VM runs as an ordinary VM whereas the CLOSEDBOX VM cannot be neither inspected nor modified except by the owner. The mains benefits of TERRA are:

- **Root secure**
  Nobody (even the owner) can break the isolation and basic privacy given by a CLOSEDBOX VM.

- **Attestation**
  Any application that runs in a CLOSEDBOX VM can be cryptographically identified by a remote user. Therefore he can trust the application and its behavior.

- **Trusted path**
  It establishes a trusted path between the VM and the remote user. This means that the users know which VM he's interacting with, and it ensures that the VM is interacting with an human user. This trusted path ensures the integrity of the information.

3.1.2. sHYPE

The SECURE HYPERVISOR (sHYPE) is an hypervisor architecture developed by IBM. It was initially developed for rHYPE, an open-source hypervisor. sHYPE tries to control the information flows between the virtual machines. More precisely, it focuses only on the explicit information flows. sHYPE acts like a *reference monitor* and verifies the authorization queries. This means it uses the MANDATORY ACCESS CONTROL (MAC) and if some subject wants to use a designed object, the reference monitor will either permit or deny the access attempt. The ACCESS CONTROL MODULE (ACM) is the component that makes the decision according to the Security Policy. So the policy enforcement and the policy management are separated.

It is important to know that a subject is defined as a virtual machine, and the object as a virtual resource. Thus, sHYPE does not check which program wants to use the resource.

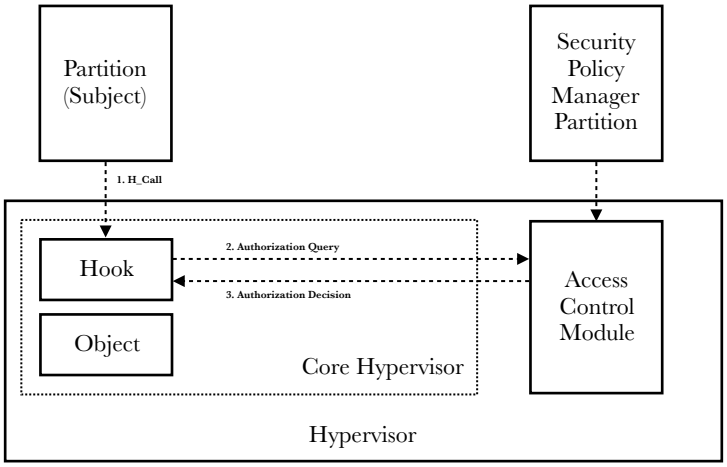

FIGURE 8 — sHYPE ARCHITECTURE

3.1.3. VAX VMM

VAX VMM is one of the first hypervisor with secure concern. This project started in 1981 and has been given the top-level A1 security rating by the NATIONAL COMPUTER SECURITY CENTER (NCSC) — it is the most secure level according to the TRUSTED COMPUTER SYSTEM EVALUATION CRITERIAL published by the NCSC in 1985. VAX VMM is based on the VAX architecture developed by DIGITAL EQUIPMENT CORPORATION in the 1970s.

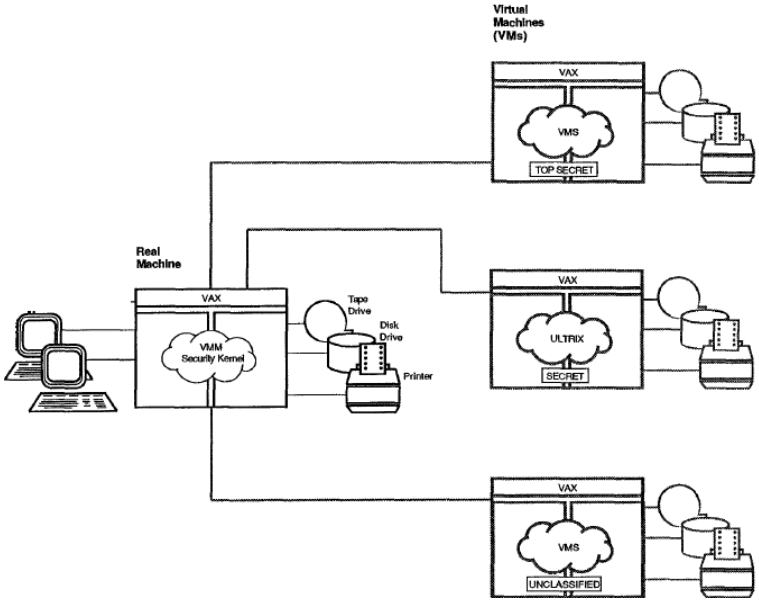

FIGURE 9 — VAX VMM ARCHITECTURE

The VAX SECURITY KERNEL runs simultaneously several VM on a single VAX physical system, enforcing isolation and the controlled share of sensitive data. The VAX VMM makes the difference between subjects (users or VMs) and objects (device, volumes, or security kernel files). Each one of these is given a secrecy class and a integrity class which will determine if the subject can access to the object.

3.2. Other securing solutions

3.2.1. HYPERWALL

Using a secure hypervisor is not the only solution to improve security of virtualized environment. HYPERWALL is another architecture designed to enhance security of an untrusted hypervisor. The hypervisor can freely manages the hardware resources while the VMs are running. The CONFIDENTIALITY AND INTEGRITY PROTECTION (CIP) is a element that protects the virtual machine memory according to the security policy. Thus, a compromised virtual machine would be unable to affect the others, and malware would not spread among them. HYPERWALL is based on several mechanisms:

- **CIP Tables**
  HYPERWALL stores the CIP tables in the DRAM. The access to that section of the physical memory by any software is denied by the memory controller. Although this reduces the amount of DRAM disposable for the system, it avoid to add a physical hardware component to store the CIP tables for the processor ship. The figure shows the host physical memory.

- **Protected memory regions**
  All the rest of the DRAM isn't freely available for the VMs. Indeed, the CIP tables protect the region used by the VMs.

- **Cryptographic keys**
  The isolation enforced by the CIP tables are improved by the use of cryptography. This allows the users to trust HYPERWALL. If the CPU secret key is burned into the ship, both the keys KHASH and KENC are stored inside the protected sections of the memory, and generated again on each boot-up cycle.

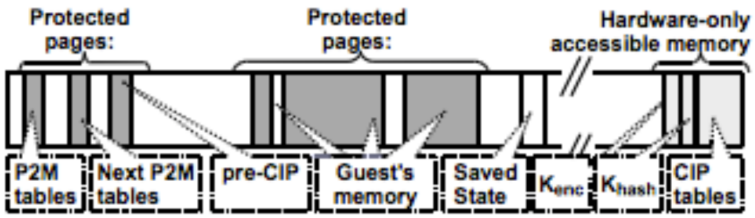

FIGURE 10 — HOST PHYSICAL MEMORY. THE CIP TABLES ARE STORED IN THE LIGHT GRAY SECTION, THE PROTECTED REGION ARE IN DARK GRAY.

3.2.2. HYPERSAFE

HYPERSAFE is an isolation technique which can improve hypervisor security by avoiding the introduction of malicious code inside the hypervisor. To do so, HYPERSAFE won't prevent any exploitable software bug in the hypervisor code source, but will guarantee its integrity throughout its runtime.

HYPERSAFE doesn't require any hardware support and can easily be implemented in common hypervisors. A principal feature of HYPERSAFE is to protect both the hypervisor and itself. This is important, because another kind of component that protects an hypervisor could be compromised, and thereby, the hypervisor would be exposed to common attacks.

HYPERSAFE has two key techniques:

- **Non-by-passable memory lockdown**
  Once a memory page is locked down, it cannot be modified until it has been unlocked. The unlocking logic will deny any attempt to modify a locked memory page, whether

for legitimate hypervisor code, or malicious code. This technique ensure hypervisor integrity.

- **Restricted pointer indexing**
  This technique use the previous one to control data. As the control data can impact the control flow of the hypervisor execution, it has to be secured.

Conclusion

Virtualization is a dazzling solution for companies who want to avoid wasting the unused resources, by reusing it for other systems. Reducing the cost and the space needed, it can also improve security in some ways. But virtualization isn't just recent invention, it's the results of decades of computer researches meeting the demands of scientists before being used by companies. In this paper, we first introduced the concept of virtualization, and the different types of hypervisors. To ensure security, it relies on the principe of both horizontal and vertical isolations. However, it comes with other threats that must be taken into consideration, otherwise the issues might just be postponed, or, at worse, be increased if a company puts to much trust and reliance on it. Indeed, the isolation isn't total, and it is possible for an attacker to know whether the operating system is being virtualized (time measuring, specific hardware devices or processor instructions, …) or not, and which hypervisor it is, by analyzing its behavior.

Knowing that the system is being virtualized is one thing, but it is another to do something with this information. Thus, we showed that the isolation isn't perfect and described two kinds of disrupting attacks that can be performed: the Denial of Service (DoS) and the Escaping of Virtual Machine (VME). DoS attacks can be very destructive. Some companies rely on the availability of their services to work. A DoS attack on them would result in important losses for the company and could lead to its closure. However, as we have mentioned before, VME can be even more destructive. Indeed, without any securing protection, the VME can allow an attacker to bypass security protections, such as access control by escaping his virtualized system to send malicious code in another system — which may contain sensitive data.

Therefore, these two examples of attacks show that virtualized environment must be secured. We introduced two kinds of secure protections for virtualization. One of its weakest component is the hypervisor itself that can allow attackers to manipulate all the virtualized systems. This is why one of the solution is to use secure hypervisor. VAX VMM is the most secured of these, as it has been certified with the A1 level a by the NCSC. It enforces strict rules even in its development process. sHYPE is also a very secured solution that control explicit information flows. It reduces the number of channels the information can use. TERRA is more flexible, yet it is less secure as doesn't use the MAC.

Using a specific hypervisor isn't the only solution available. HYPERWALL is a special architecture that protects the memories of the virtual machines, and HYPERSAFE avoid the introduction of malicious code inside the hypervisor.

Throughout this paper, we described virtual machine escape, and gave various solutions to reduce the possibility of doing so. At last but not the least, the best secure solution will still be ineffective if the system isn't properly configured. A secure solution incorrectly set is as bad as none at all.