

Virtualization Security

Fabio Martinelli, Daniele Sgandurra, Anna Vaccarelli

`anna.vaccarelli@iit.cnr.it`

11/11/2010

1 Virtualization

- Virtualization
- Virtualization-Based Security

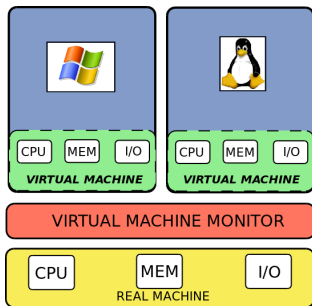
2 Current Solutions

- Application and Operating System Integrity
- Remote Attestation of Integrity
- Security of JVM in GRID systems

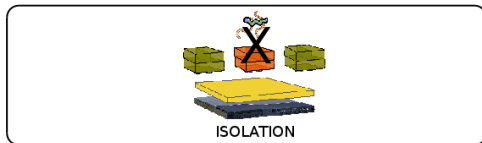
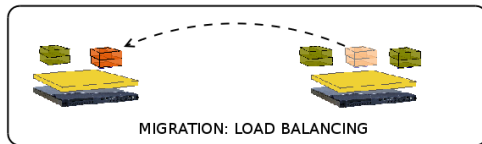
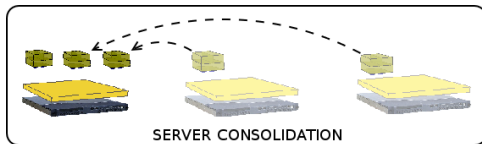
3 Future Trends

Virtualization

- **Virtual Machine Monitor (VMM)**: manages Virtual Machines (VMs), i.e. execution environments that emulate, at software, the behavior of the underlying physical machine.
- A real machine can **support several VMs**, each with a distinct OS.



Classic Virtualization Benefits



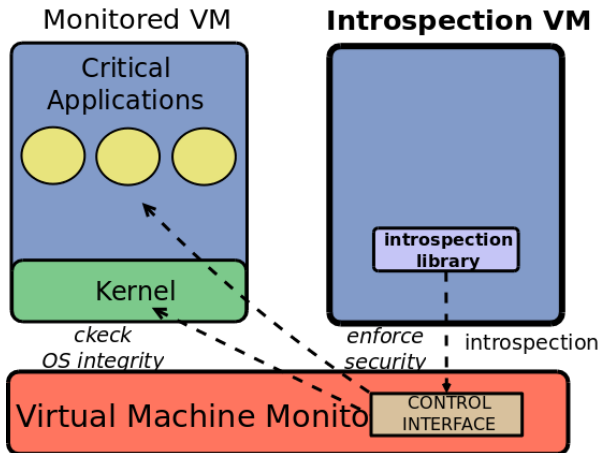
Virtualization-based Security

- It is important to notice that the most popular reasons that favor the adoption of virtualization are related to distinct considerations:
 - strong simplification in **system administration / management**;
 - **energy saving**.
- The ability of exploiting virtualization to increase the **overall security** is a further important advantage.
- As an example, a key advantage of virtualization is that its adoption may be **transparent** for the applications and the OSES:
 - checking the integrity of the software that a VM runs **without modifying this software or requiring some user intervention**;
 - the integrity of the overall status of the system hosted by the VM can be checked **without forcing the users to install further tools or to modify any software**.

Virtual Machine Introspection

- **Virtual Machine Introspection (VMI)** enables a privileged VM, or Introspection VM (I-VM), to retrieve critical data structures in the memory of a Monitored VM (Mon-VM) at the kernel or at the user-level.
- An I-VM can analyze the state of the processes/kernel on a Mon-VM at the hardware level, **without introducing additional units**.
- Introspection is applied at a **lower level** than the one an attacker can gain and it is **very complex to elude it**.

Introspection VM



Advantages of Virtual Machine Introspection

- **Full visibility** of the system running inside the Mon-VM: the I-VM can access every Mon-VM component, such as the main memory or the processor's registers.
- **More robustness**: the I-VM is isolated from the Mon-VM.
- **Transparency**: the security checks can be implemented without modifying the software on the Mon-VM and they are almost invisible.

An Architecture that Exploits Introspection

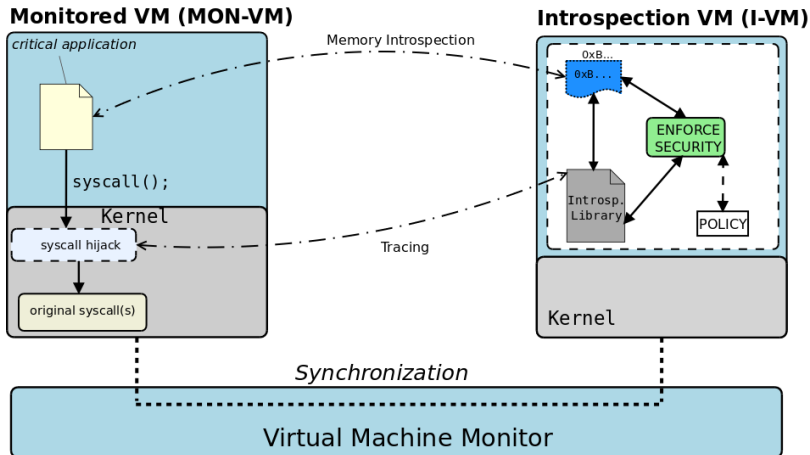
- The run-time architecture consists of two virtual machines:
 - 1 the **monitored VM (Mon-VM)**, i.e. the VM executing the critical applications;
 - 2 the **introspection VM (I-VM)**, i.e. the VM monitoring those applications through virtual machine introspection.
- The I-VM **can access each component of the Mon-VM** to inspect its running state both to:
 - **check the security of the critical applications;**
 - **protect the kernel integrity.**

Checking the Security of Critical Applications

The I-VM checks the security of the critical applications:

- it verifies that the **system call** that an application wants to issue belongs to the set of system calls that the policy admits;
- it checks that the **current trace of system calls** issued by the applications belongs to the policy;
- it checks whether some **security properties** coupled with the current system-call holds.

Example: Checking the Security of Critical Applications



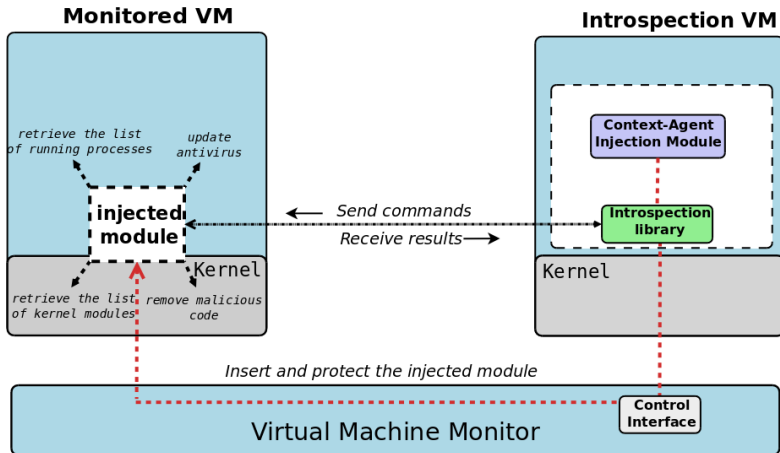
Protecting the Operating System Integrity

- Periodically, the I-VM checks the **kernel integrity**.
- Examples:
 - **detecting kernel code modifications;**
 - **running processes checker;**
 - **open files checker;**
 - **loaded modules Authenticator;**
 - **promiscuous mode checker;**
 - **anti-spoofing.**

Example: Inserting Software to Protect OS

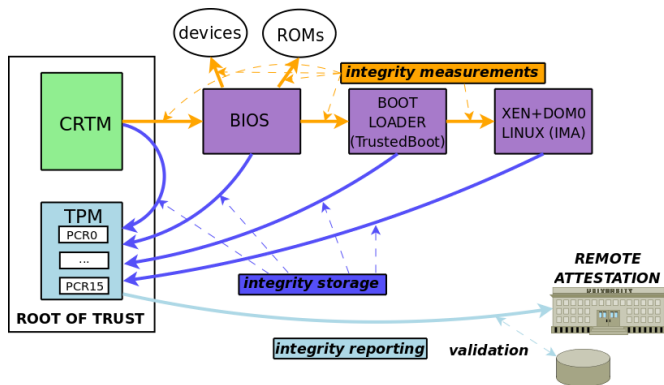
- The I-VM can **transparently inject** a new module inside a Mon-VM to apply security checks:
 - **antivirus updates;**
 - **OS patches;**
 - **remove malicious software;**
 - **check the running processes;**
 - **etc.**
- The I-VM also **protects the module** from modifications by any software running in the Mon-VM .

Example: Inserting Software to Protect OS



Standard Boot-Time Integrity Checks

Current solutions: **static checks** at boot-time that exploits the TPM to verify the integrity of the application binaries that have been loaded in the OS.



An Integrity Measurement System for Virtual Machines

- Our view: **node integrity** is a precondition of any overlay security policy and it should be attested not only when a node joins an overlay but also as long as the node belongs to the overlay (**dynamically**).
- Virtual machine introspection can implement an **integrity measurement system** on a cloud architecture to **continuously attest the integrity** of cloud nodes by applying alternative policies.
- To protect the integrity of a node by defining both a **start-up attestation** and a **continuous monitoring** that are applied when a node joins the overlay and as long as the node belongs to the overlay.

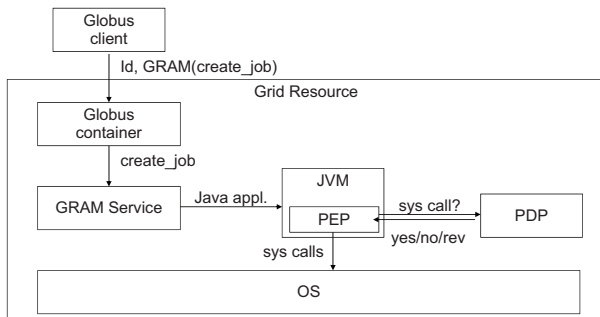
An Integrity Measurement System for Virtual Machines

- Each node of the overlay runs two VMs:
 - the Mon-VM;
 - the I-VM, which enforces the security policy (Assurance VM).
- The I-VMs cooperate **to apply measurements on behalf of the overlay** by accessing the live state of the Mon-VM.
- Anytime an I-VM detects a loss of integrity, **it kills the Mon-VM** on its node and informs the other I-VMs.
- Trust in the measurements requires the correct configuration of both the I-VM and the underlying VMM and it is guaranteed by measurements and controls of a **Trusted Platform Module (TPM)**.

Java Virtual Machine (JVM)

- Similar concepts holds of Virtual machines for programmings languages as Java
 - The JVM is monitored by a **policy** (depending on several factors) evaluated by a Policy Decision Point (PDP).
 - The JVM could be also embedded in a virtualization environment for computations as a **GRID** system.

Architecture



Virtualization-Based Secure Isolation

- **Isolate** different areas on a computer for doing activities that require **different levels of security**.
- **Protecting web surfers from downloading malware and having sensitive data stolen:**
 - e.g., if a user wants to open or execute files downloaded from the Internet **the system copies the file to a "shadow" VM** that executes the file. Any malware that might get downloaded stay within that VM and cannot infect the rest of the computer;
 - e.g., if a user accesses a site using https, the operation is done by a **high assurance VM** that provides a higher level of security.
- The isolating of the activities is all **invisible to the end user**.

Virtualization and Hardware-Based Security

- Tighter integration among **virtualization** and **hardware**.
- Multicore and VMs: **assign each VM to a dedicated CPU**.
- **Peripherals** will also support **trusted-computing** authentication and integrity goals by exploiting TPMs and attestation capabilities:
 - peripherals **enforcing system-wide access control and information flow security policies** in a verifiable manner.
- Hardware support for **cloud computing**:
 - low-level **monitoring** and **metering** (CPU cycle or storage bandwidth) to support business-level requirements with minimum overhead;
 - resource control to **enforce resource usage limits** (against DOS);
 - reliable sanitization of frequently reused resources for **fast and efficient provisioning of workloads**.

Compliance, Update and Patch Management

- **Compliance:**
 - how to enforce compliance where users can **spawn multiple VMs with distinct environments**.
- **Update and patch management:**
 - **VM sprawl:** potential for large number of applications and OSs deployed.
 - **How to update VMs** that aren't currently spawned.

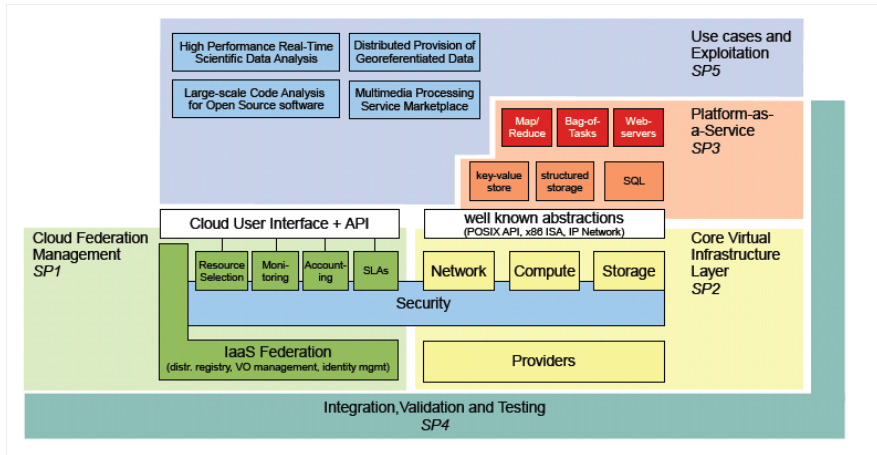
Secure Clouds management

- Adding **controls on data sharing** between VMs to improve isolation.
- Continuously **monitoring isolation mechanisms** and protecting integrity.
- **Automating security management** to account for increasing dynamics of “Cloud Computing”.
- **Secure Migration of VMs** among Clouds.
- Advantages:
 - **simplifies security management**;
 - reduces the risk of security exposures through consistent, **policy-driven enforcement**;
 - leverage virtualization through **centralized security services**.

CONTRAIL project

- **New EU research project** on elastic computing
- main goal to create a secure and scalable elastic computing paradigm including:
 - Integration of tools for Platform as a Service (PaaS) with Infrastructure as a Service (IaaS).
 - Securely managing Clouds

CONTRAIL project architecture



Some References



[1] Transparent Process Monitoring in a Virtual Environment.

Daniele Sgandurra, Fabrizio Baiardi, Dario Maggiari and Francesco Tamberi
In Electronic Notes in Theoretical Computer Science, volume 236, pp. 85-100, 2009.



[2] Attestation of Integrity of Overlay Networks.

Fabrizio Baiardi and Daniele Sgandurra.
Journal of System Architecture, Special Issue on Security and Dependability Assurance of Software Architectures, 2010



[3] Policy Driven Virtual Machine Monitor for Protected Grids.

Fabrizio Baiardi, Laura Ricci, Paolo Mori, Anna Vaccarelli.
HPDC 2006: 313-316



[4] Runtime monitoring for next generation Java ME platform.

Gabriele Costa, Fabio Martinelli, Paolo Mori, Christian Schaefer, Thomas Walter
Computers & Security 29(1): 74-87 (2010)



[5] Measuring the Semantic Integrity of a Process Self.

D. Sgandurra.
Ph.D. Thesis Univ. Pisa (2010)