Using Hardware-Assisted Virtualization to Protect Application Address Space Inside Untrusted Environment

Denis Silakov
Institute for System Programming, RAS
silakov@ispras.ru

May 12, 2011
Hardware-Assisted Virtualization

Non-root mode

- Ring 3: Applications
- Ring 2: Not used
- Ring 1: Not used
- Ring 0: OS

Root mode

- Virtual machine monitor (hypervisor)

Classical x86 privilege levels («protection rings»)

Hardware-assisted virtualization – new privilege level
Virtualization and Security

Hypervisor is just a program. It can perform different tasks, not only virtual machine management.

Hypervisor has higher privileges than OS.

Hypervisor can be made much more smaller than OS ⇒ (potentially) less vulnerable.

⇒ Some security functions can be delegated to hypervisor.
Hypervisor-Based Protection Systems

- Encrypt process memory (*Overshadow*)
- Launch different processes in different VMs (*Qubes OS, Terra*)
- Provide additional services for Malware detection software like antivirus, IDS, etc. (*VMware VMsafe*)
Our Goal

Develop a protection system with the following properties:

- No modifications of applications or OS are required
- Memory and files of trusted process are protected from unallowed modifications attempts only. The process should still be able to interact with other programs

OS is considered to be (potentially) compromised, so the system should protect processes from the OS itself.
System Architecture

OS
- Interception of necessary system calls
- Security module
- Analysis of system calls and prevention of forbidden actions

Hypervisor

Trusted process

Untrusted process
Controlling Consistency of a Trusted Process

- Protect files on storage media
- Protect address space of a running process
Checking Consistency of Application Files

Checksum-based algorithm

- Calculate checksum (SHA-1) for every memory page of application file inside trusted environment
- At startup, hypervisor checks the sums to ensure consistency of loaded/launched files
Protecting Control Flow

Single-core CPU $\Rightarrow$ only one process is running at any moment of time

OS

Trusted process

Context switch

Untrusted process

Context switch

Trusted process

Hypervisor

Save process state and context

Verify that state of trusted process was not modified
Protecting Address Space

Based on *Nested Page Tables* (used to translate memory addresses inside VM to physical ones). A separate set of NPT for every process is maintained by hypervisor.

For every process, hypervisor sets corresponding access permissions for different physical pages

- for pages used by a trusted process, only that process is granted with write permissions
- if a page is not used by any trusted process, hypervisor doesn’t perform any specific actions
- OS is allowed to read all pages, but cannot modify memory of trusted processes
Types of Protected Applications

- Statically linked programs
- Dynamically linked applications and programs loading libraries at runtime (using `dlopen()` functionality); pre-loaded libraries are also handled
- Multithread applications
Assumptions

- System dynamic loader is trusted
- All libraries loaded by trusted process are also trusted (work in progress – allow to load untrusted libraries)
- Applications don’t use lazy binding (can be achieved by setting LD_BIND_NOW variable)
Implementation

- Hypervisor based on KVM (Kernel Virtual Machine)
- Supported hardware – AMD processors with hardware-assisted virtualization
- Supported OSes – 32bit Linux with kernel 2.6.31 or higher
Attack Detection Power

*Trusted process – SSH (network connection with remote host)*

Emulated attacks:

- Modification of application executables and libraries (including those loaded using `dlopen()`)
- Injection of malicious code in the process address space:
  - load a library using `LD_PRELOAD`
  - modify process memory using `ptrace()`-based techniques (gdb debugger, PreZ code injector)

All attack attempts were successfully detected by the protection system.
Performance

- Apache – Flood load test (≈10 requests per second)
- SCP – copy large file (4GB) using SSH
Questions?

Denis Silakov
Institute for System Programming, RAS
silakov@ispras.ru