# Hypervisor assisted malware detection (analysis)

**Gábor Pék**
Laboratory of Cryptography and System Security (CrySyS)
Budapest University of Technology and Economics
www.crysys.hu

# Outline

- Basics of hardware assisted virtualization
- Conventional malware detection
  - Signature based
  - Heuristics
- Malware analysis in ring -1
- Performance as a No.1 bottleneck
- Malware detection in ring -1
- Statistics as a prospering tool

# Basics of hardware assisted virtualization

- Introduced in 2006 (Intel VT-x and AMD-V)
- Same theoretical basics, differences only in the implementation
- A new CPU privilegue level (ring -1) for the hypervisor
- Hypervisor mediates between the hardware and (virtualized) operating systems.
  - Main tasks: management of resources, virtualizing processors and states
  - Examples: XenServer, Hyper-V Server, Blue Pill
- CPU can operate in root and non-root mode
  - Root mode - CPU executes the hypervisor
  - Non-root mode – CPU executes the guest operating systems

# More details about hardware ass. virtualization

- Saving guest states
  - VMCB  - Virtuali Machine Control Block
  - VMCS – Virtual Machine Control Structure
- VM Exits : switching between root and non-root mode
  - Exceptions
  - Faults
  - Privileged instruction execution
  - Invalid opcode execution
- VM Exits can be intercepted and handled by the hypervisor
  - Basics of malware analysis and detection

# Conventional malware detection

- Current AVs run at the same priv. level as malware do (ring 0, ring 3)
- Signature based - searching for static patterns
  - E.g., Byte sequences, header information
  - Difficult to handle new variants
  - Requires large databases
  - Low false positive rate
- Heuristics – searching for behaviour based patterns
  - E.g., Memory pages with specific information
  - New variants can be detected easily
  - Higher false positive rate

# Malware analysis in ring -1

- Many research papers about hypervisor assisted malware analysis
  - Ether
  - Pyrenée
  - MAVMM
  - Patagonix, etc…
- Using a properiatory or existing virtualization software
  - Xen
  - KVM
  - TVMM
  - All of them requires a reboot – is it a problem?
- Performance problems…

# Perfomance problems

- Guests being analysed are unusable
- Cause: No native support for decreasing the semantic gap between the hypervisor and the operating system
  - E.g., No CPU support for system call tracing → heavy use of syscalls by user-space malware
- Current solution for syscall tracing uses page faults
  - Problems: Excessive number of page faults by design
  - Filtering is not a efficient solution

# Goals

- Goals
  - Detecting unknown parasitic malware in real time
  - Transparency
  - Low performance overhead
  - Using heuristics (many advantages)
  - False positives?
- Solution: Stastics or Machine Learning
  - Separating benign and malicious syscall sequences
  - Creating stastics from benign and infected systems
  - We can use hypothesis theory to differentiate them

# Challenges

- Decreasing the semantic gap
  - Reconstructing operating system level structures in the hypervisor: process names, process features, memory protections, etc
- Solving performance issues (new syscall method)
- Automatically record benign and a lot of (>1000) malicious traces
- A system should be build to
  - Fetch malicious samples and execute them
  - Save and record syscall logs from the hypervisor (not trivial)
  - Revert the system
  - Start everything again

# Performance issues

- Important to show that there is no significant performance degradation
- Create samples (performance counters) from native and monitored systems
- Sampling is relatively time consuming
- Questions:
  - What is the lowest number of sample elements required to draw conclusion?
  - Is there a significant difference between performances?
  - Compare 3 methods (unmonitored, old syscall methods, new approach)

# Thank you