
Deepa Srinivasan, Zhi Wang, Xuxian Jiang, Dongyan Xu*
North Carolina State University, Purdue University*
Malware Infection Trend

New malware samples collected by McAfee Labs, by month*

*Figure source: McAfee Threats Report: Second Quarter 2011, McAfee Labs
Anti-Malware Isolation

- Traditional anti-malware tools are not well-isolated
- Virtual Machine (VM) introspection
  - Isolate tool by placing it outside a VM
  - Analyze states and events externally
Anti-Malware Isolation

- Traditional anti-malware tools are not well-isolated
- Virtual Machine (VM) introspection
  - Isolate tool by placing it outside a VM
  - Analyze states and events externally
Out-of-VM Solutions

- Livewire (Garfinkel et al., NDSS ‘03)
- XenAccess (Payne et al., ACSAC ‘07)
- VMScope (Jiang et al., RAID ‘07)
- Lares (Payne et al., Oakland ‘08)
- ...

Semantic Gap in Introspection

- **What we want to observe**
  - High-level states and events (e.g. system calls, processes)

- **What we can observe**
  - Low-level states and events (e.g. raw memory, interrupts)
Addressing the Semantic Gap

- Guest view casting
  - VMWatcher (Jiang et al., CCS ‘07)

- Automatic generation of introspection-based tools
  - Virtuoso (Dolan-Gavitt et al., Oakland ‘11)

- Issues
  - Sensitive to internal OS changes or updates
  - Incompatible with existing anti-malware tools
Our Goal

Support existing in-host process monitors out-of-VM without semantic gap!
In-host strace
In-host Monitors

- Process-level monitoring
  - System calls
  - Library calls
  - Instruction execution traces
Process Out-Grafting

- **Isolation**
  - Monitor protection from malware

- **Compatibility**
  - Natural support for fine-grained user-mode process monitoring tools (*strace, ltrace, ...*)

- **Efficiency**
  - No significant performance overhead due to isolation
Rest of This Talk

- Motivation
- **System Design**
- Implementation & Evaluation
- Related Work
- Conclusion
Assumptions

- Trusted hypervisor
- Untrusted monitored VM

- Non-goals
  - OS kernel-level monitoring
  - Stealthy monitoring
Key Techniques

Technique I: On-Demand Grafting

Technique II: Mode-sensitive Split Execution
Key Techniques

Technique I: On-Demand Grafting

Production VM

Security VM

Hypervisor

User Kernel

User Kernel

Monitor
On-Demand Grafting
On-Demand Grafting

- Relocate suspect process to security VM
- Enable efficient, native inspection
  - Eliminate hypervisor intervention
  - Support existing process monitoring tools
- Initiate out-grafting as needed
- Restore process after monitoring

When? What? How?
When to Out-Graft?

- Process in user-mode
- Process in kernel-mode
  - Hypervisor notified when user-mode execution resumes
What to Out-Graft?

- Architecture-specific resources
- No OS kernel-specific resources
  - Main root cause in semantic gap
  - Continued execution of out-grafted process
How to Out-Graft?

Production VM

Stub protected by hypervisor

Security VM

Suspect Process

Monitor

Hypervisor

User Kernel

Kernel

User

VCPU Register State Page Mappings

Physical Memory Frames

Helper Module
Mode-Sensitive Split Execution
Mode-Sensitive Split Execution

- All user-mode execution occurs in security VM
- All kernel-mode execution occurs in production VM
  - Out-grafted process considers itself running in production VM
System Call Redirection

- Smooth, continued execution of out-grafted process
- Monitor isolation

Hypervisor (e.g., KVM)
Page Fault Forwarding

- Consistent process memory mapping between VMs
- No semantic knowledge in security VM (e.g. memory-mapped file)
Page Fault Forwarding

- Consistent process memory mapping between VMs
- No semantic knowledge in security VM (e.g. memory-mapped file)
Rest of This Talk

- Motivation
- System Design
- **Implementation & Evaluation**
- Related Work
- Conclusion
Implementation

- Hypervisor: KVM (2.6.36.1)
  - Out-grafting support: +1309 SLOC
  - Extended page tables support (Intel Core i7)
- Host OS: Ubuntu 10.04 (kernel 2.6.28)
- Guest OS: Ubuntu 9.04, Fedora 10
Security Analysis

- Monitor isolation and effectiveness
  - System call forwarding
  - Security VM protected from kernel attacks
Security Analysis

- **Stub protection**
  - Hypervisor-protected memory and page tables
Security Analysis

- Out-grafting detection
  - Strong policy for random, arbitrary out-grafting
Case Studies

- *thttpd* as test process for out-grafting
  - Both disk and network usage
- Support existing in-host tools, out-of-VM
  - strace, ltrace, gdb
  - OmniUnpack *(Martignoni et al., ACSAC ‘07)*
Case Studies

- `httpd` as test process for out-grafting
  - Both disk and network usage
- Support existing in-host tools, out-of-VM
  - `strace`, `ltrace`, `gdb`
  - [OmniUnpack](#) (Martignoni et al., ACSAC ‘07)
strace

In-VM strace

Out-of-VM strace
Code Unpacking Detection

- **OmniUnpack** *(Martignoni *et al.*, ACSAC ‘07)*
  - Track page writes and executions
  - Detect unpacking when executing a previously-written page

- **Faithful reproduction of algorithm in Linux**
  - Kernel module in security VM

- NX bit support only in security VM kernel
- Localized overhead in out-of-VM monitoring
- Thorough test of page fault forwarding
Performance

- Dell T1500, Intel Core i7, 4 cores, 2.6 GHz, 4 GB RAM
- VM configuration
  - Production VM: 1 VCPU, 2047 MB RAM
  - Security VM: 1 VCPU, 1 GB RAM
Performance

- Inter-VM system call (*getpid*): ~11 µs
- Process state identification: ~250 µs
- Slowdown to out-grafted process
  - File-copy time: 35.42%
  - `thttpd` throughput: 7.38%
Performance

Production VM Slowdown with a Contending Process Out-grafted
Rest of This Talk

- Motivation
- System Design
- Implementation & Evaluation
- Related Work
- Conclusion
Related Work

- **VM introspection**
  - Livewire (Garfinkel et al., NDSS ‘03), XenAccess (Payne et al., ACSAC ‘07), VMScope (Jiang et al., RAID ‘07), Lares (Payne et al., Oakland ‘08), VMWatcher (Jiang et al., CCS ‘07), Virtuoso (Dolan-Gavitt et al., Oakland ‘11)

- **Efficient, isolated monitoring**
  - SIM (Sharif et al., CCS ’09)

- **Process migration**
  - BLCR (Smith, UCB-TR ‘08), Zap (Osman et al., OSDI ‘02), Migration survey (Nuttall et al., ACM OS Review ‘94)

- **System call forwarding**
  - Application protection (Ta-Min et al., OSDI ‘06), Monitoring fidelity (Martignoni et al., ICISS ‘09)

- **Sandboxing, isolation techniques**
  - Ostia (Garfinkel et al., NDSS ‘04), Janus (Goldberg et al., Security ‘96)

Technique I: On-Demand Grafting

Technique II: Mode-sensitive Split Execution
Thank you!

Questions?