“Out-of-the-Box” Malware Defense

Xuxian Jiang

Department of Information and Software Engineering
George Mason University
December 12, 2007
Outline

- Motivations
- “Out-of-the-box” malware defense
  - New VMM mechanism: OBSERV
  - New capabilities enabled
    - Invisible system logging
    - Malware detection by view comparison
    - External run of COTS anti-malware software
- Future work
- Summary
Motivations

Internet malware remains a top threat

Malware: viruses, worms, rootkits, spyware, bots...

Virus writers get stealthy

Rootkit numbers rocketing up, McAfee says

The strange decline of computer worms
Why “Out-of-the-Box”? 

- State-of-the-art malware defense
  - Running anti-malware software inside the monitored system
    - Advantage: They can see everything (e.g., files, processes...) They may not see anything!
    - Disadvantages:
Why “Out-of-the-Box”?  

- Current approach fundamentally flawed
  - Malware running in the same system space with anti-malware software at the same privileged level
  - No clear winner in the arms race between them

- Solution: “Out-of-the-box” malware defense

![Diagram showing IE, Firefox, VirusScan, and Virtual Machine Monitor (VMM)]
The “Semantic-Gap” Challenge

- What we can observe:
  - Low-level states
    - Memory pages, disk blocks...
  - Low-level events
    - Privileged instructions,
    - Interrupts, I/O...

- What we want to observe:
  - High-level semantic states
    - Files, processes...
  - high-level semantic events
    - System calls, context switches...
Our Solution: OBSERV

OBSERV: “Out-of-the-Box” with SEMantically REconstructed View

→ A new mechanism missing in all current VMMs
New Capabilities

Capability I: Invisible system logging

Capability II: Malware detection by view comparison

Capability III: External run of COTS anti-malware software
OBSERV: Bridging the Semantic Gap

- **Step 1:** Procuring low-level VM states and events
  - Disk blocks, memory pages, registers...
  - Traps, interrupts...

- **Step 2:** Reconstructing high-level semantic view
  - Files, directories, processes, and kernel modules...
  - System calls, context switches...

**VM Introspection**

**Guest View Casting**
Step 1: VM Introspection

<table>
<thead>
<tr>
<th>Raw VMM Observations</th>
<th>Virtual Machines (VMs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>VM disk image</td>
<td>✓</td>
</tr>
<tr>
<td>VM physical memory</td>
<td>✓</td>
</tr>
<tr>
<td>VM hardware state (e.g., registers)</td>
<td>✓</td>
</tr>
<tr>
<td>VM-related low-level events (e.g., interrupts)</td>
<td>✓</td>
</tr>
</tbody>
</table>

VMware Academic Program
Step 2: Guest View Casting

Key observation: The guest OS provides all semantic “templates” of data structures and functions to reconstruct VM’s semantic view.
## Guest View Casting

<table>
<thead>
<tr>
<th>Raw VMM Observations</th>
<th>Casted Guest Functions &amp; Data Structures</th>
<th>Reconstructed Semantic View</th>
</tr>
</thead>
<tbody>
<tr>
<td>VM disk image</td>
<td>Device drivers, file system drivers</td>
<td></td>
</tr>
<tr>
<td>VM physical memory</td>
<td>Memory translation, task_struct, mm_struct</td>
<td></td>
</tr>
<tr>
<td>VM-related low-level events</td>
<td>Event semantics</td>
<td></td>
</tr>
<tr>
<td>(e.g., interrupts)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VM hardware state</td>
<td>CR3, MSR_SYSENTER_CS, MSR_SYSENTER_EIP/ESP</td>
<td>Event-specific arguments...</td>
</tr>
<tr>
<td>(e.g., registers)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Guest View Casting on Memory State
Guest Memory Addressing

- Traditional memory addressing
  - MMU translates VA to PA
  - OS image mapped to known PA
    - Linux: VA 0xc0000000 == PA 0x0
    - Windows: VA 0x80000000 == PA 0x0

- VM complicates the translation

  - Guest virtual -> guest physical
  - Emulated Address Translation
  - Reverse Address Translation
  - VM Introspection
  - Guest physical -> host physical

Kernel Symbols
Guest View Casting on System Calls

- System call instructions
  - `int 0x80; sysenter`

- System call convention
  - `EAX, EBX, ECX, EDX, ESI, EDI, EBP, ...`

1. `int 0x80, sysenter`
2. Trap Generation
3. Trap and Emulate
4. Emulate Instruction
5. Continue the Execution

Demo!
Related Work

* Virtual machine introspection (*Livewire*[Garfinkel03], *IntroVirt*[Joshi05], *HyperSpector*[Kourai05])
  - Focusing on targeted attacks for specialized IDSes

* Secure monitors (*CoPilot*[Petroni04], *Terra*[Garfinkel03], *sHype*[Sailer05], *SecVisor*[Perrig07])
  - Missing a basic mechanism similar to OBSERV
Outline

- **Motivations and research overview**
- "Out-of-the-box" malware defense
  - New VMM mechanism: OBSERV
  - New capabilities enabled
    - Invisible system logging
    - Malware detection by view comparison
    - External run of COTS anti-malware software
- Future work
- Summary
New Capability I: Invisible System Logging

- **Trusted logging:** an essential function for honeypots

- **Two current approaches**
  - **External** (e.g., tcpdump, ethereal, etc)
    - Only monitoring network traffic
  - **Internal** (e.g., sebek, syslog, etc)
    - Can be compromised!

<table>
<thead>
<tr>
<th></th>
<th>Tamper-Resistance</th>
<th>Deep Inspection</th>
</tr>
</thead>
<tbody>
<tr>
<td>External</td>
<td>✓</td>
<td>✗</td>
</tr>
<tr>
<td>Internal</td>
<td>✗</td>
<td>✓</td>
</tr>
</tbody>
</table>
Invisible System Logging

- **Sebek**: de-facto honeypot logging tool
- **Can be detected, disabled, or bypassed by NoSEBrEaK** [Holz+, BlackHat’04/Defcon 12]
Invisible System Logging

Demo Clip (2.5 minutes):

http://www.ise.gmu.edu/~xjiang/vmscope/sebek.swf

OBSERV-based logging

Sebek-based logging

[Holz+, Blackhat'04/Defcon 12]
New Capabilities II & III

**Capability I:** Invisible system logging

**Capability II:** Malware detection by view comparison

**Capability III:** External run of COTS anti-malware software
View Comparison on Volatile Memory State

Experiment setup

- **Guest VM:** Windows XP (SP2)
- **Host OS:** Scientific Linux 4.4
- **VMM:** VMware Server 1.0.1

**OBSERV view**

**“In-the-box” view**
View Comparison on **Persistent** Disk State

- **Experiment setup**

  - **Guest VM:** A Redhat 7.2-based honeypot
  - **Host OS:** Windows XP (SP2)
  - **VMM:** VMware Server 1.0.1

Symantec AntiVirus

**OBSERV view**

“**In-the-box**” view
External Run of COTS Anti-Malware Software

- **Experiment setup**
  - Both guest OS and host OS run Windows XP (SP2)
  - VMM: VMware Server 1.0.1

- **Running Symantec AntiVirus twice**
  - Inside
  - Outside

- Hacker Defender
- NTRootkit
Summary

- OBSERV enables “out-of-the-box” malware defense
  - Eliminating semantic gap
  - Enabling new malware defense capabilities
Thank you!

For more information:

Email:  xjiang@ise.gmu.edu
URL:  http://www.ise.gmu.edu/~xjiang