EXTERIOR: Using A Dual-VM Based External Shell for Guest-OS Introspection, Configuration, and Recovery

Yangchun Fu, Zhiqiang Lin

Department of Computer Science
The University of Texas at Dallas

March 17th, 2013
Outline

1. Overview
2. Our Approach
3. Evaluations
4. Conclusion
Virtualization

Virtualization (i.e., hypervisor) [Popek and Goldberg, 1974] has pushed our computing paradigm from multi-tasking to multi-OS. Consolidation, Migration, Isolation ...
Virtualization (i.e., hypervisor) [Popek and Goldberg, 1974] has pushed our computing paradigm from multi-tasking to multi-OS.
Virtualization (i.e., hypervisor) [Popek and Goldberg, 1974] has pushed our computing paradigm from multi-tasking to multi-OS.

Consolidation, Migration, Isolation ...
Execution Mode
Execution Mode
Execution Mode
Virtual Machine Introspection (VMI) [Garfinkel et al. NDSS'03]
Virtual Machine Introspection (VMI) [Garfinkel et al. NDSS'03]

Using a trusted, dedicated virtualization layer program to monitor the running VMs
Virtual Machine Introspection (VMI) [Garfinkel et al. NDSS’03]

Using a trusted, dedicated virtualization layer program to monitor the running VMs

- Intrusion Detection
- Malware Analysis
- Memory Forensics
Virtual Machine Introspection (VMI)

Using a trusted, dedicated virtualization layer program to monitor the running VMs

- Intrusion Detection
- Malware Analysis
- Memory Forensics
Virtual Machine Introspection (VMI)

Using a **trusted, dedicated** virtualization layer program to **monitor** the running VMs

- Intrusion Detection
- Malware Analysis
- Memory Forensics

**EXTERIOR**

- **Execute** trusted utilities in SVM for **timely** Guest-OS introspe**ction**, (re)**configura**tion and **recovery**.
The Semantic Gap in VMI ([Chen and Noble HotOS'01])

- View exposed by Virtual Machine Monitor is at low-level
- There is no abstraction and no APIs
- Need to reconstruct the guest-OS abstraction
Outline

1. Overview
2. Our Approach
3. Evaluations
4. Conclusion
Using a Dual-VM Architecture
Using a Dual-VM Architecture

Secure VM (SVM)

User Space
Kernel Space

Guest VM (GVM)

User Space
Kernel Space

Virtual Machine Introspection
Virtual Machine Configuration
Intrusion Detection, Prevention (Recovery)
Using a Dual-VM Architecture

- **Virtual Machine Introspection**
- **Virtual Machine Configuration**
- **Intrusion Detection, Prevention (Recovery)**
Advantages

- **Isolation** (SVM and GVM are isolated)
- **Trustworthiness** (trust code is running in secure VM)
- **Automation** (no need to develop introspection utilities)
- **Security** (enabling malware analysis, forensics...)
- **Transparency** (programmers write native program in SVM)
Syscall trace of running `sysctl -w` to turn on the address space randomization in Linux kernel 2.6.32
Architecture Overview of EXTERIOR

Overview

- User Space
- Kernel Space
- Memory
- Global
- Heap
- Stack 1
- Stack 2
- Stack n
- Syscall 1
- Syscall 2
- Syscall n
- Process/IO/Memory/Security Management
- Interrupt/Exception Handler
- Other System Components and Drivers
- Kernel Syscall Context Identification
- Kernel Data Identification and Redirection
- Binary Translation Based Virtualization Layer
- GVM Memory Mapping and Address Resolution

Our Approach

- Secure VM (SVM)
- Guest VM (GVM)
- Xen/KVM/Vmware/VirtualBox/VirtualPC/HyperV/OpenVZ/QEMU

Evaluations

- C_user
- D_stack
- D_global
- D_heap
- P

Conclusion
The algorithms

1. Dynamic Binary Instrumentation
2. If SysCallExecContext(s):
3. If SysCallRedirectable(s):
4. RedirectableDataTracking(i);
5. For \( \alpha \) in MemoryAddress(i):
6. If DataRead(\( \alpha \)):
7. \( PA(\alpha) \leftarrow V2P(\alpha) \)
8. Load(\( PA(\alpha) \))
9. Else:
10. If Configuration:
11. Store(\( PA(\alpha) \))
12. Else (Introspection):
13. COW-Store(\( PA(\alpha) \))
The algorithms

The Algorithm

1: DynamicBinaryInstrumentation(i):
2:   if SysCallExecContext(s):
3:     if SysCallRedirectable(s):
4:       RedirectableDataTracking(i);
5:       for α in MemoryAddress(i):
6:         if DataRead(α):
7:           PA(α) ← V2P(α)
8:           Load(PA(α))
9:       else:
10:          if Configuration:
11:             Store(PA(α))
12:          else: //Introspection
13:             COW-Store(PA(α))
Mapping the GVM Memory Address

---

**Overview**

**Our Approach**

**Evaluations**

**Conclusion**

---

[Diagram showing the process of mapping GVM memory addresses]
## Effectiveness

<table>
<thead>
<tr>
<th>Category</th>
<th>Utility</th>
<th>Syntactics</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introspection</td>
<td>ps (1)</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>pstree (1)</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>lsmod (8)</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>dmesg (1)</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>vmstat (8)</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>netstat (8)</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>lsof (8)</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>uptime (1)</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>df (1)</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Configuration</td>
<td>sysctl (8)</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>route (8)</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>hostname (1)</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>chrt (1)</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>renice (1)</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Recovery</td>
<td>kill (1)</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>rmmod (8)</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
## Recovery

<table>
<thead>
<tr>
<th>Rootkit</th>
<th>Targeted Function Pointer</th>
<th>Succeed?</th>
</tr>
</thead>
<tbody>
<tr>
<td>adore-2.6</td>
<td>kernel global, heap object IDT table</td>
<td>✗</td>
</tr>
<tr>
<td>hookswrite</td>
<td>IDT table</td>
<td>✓</td>
</tr>
<tr>
<td>int3backdoor</td>
<td>IDT table</td>
<td>✓</td>
</tr>
<tr>
<td>kbdv3</td>
<td>syscall table</td>
<td>✓</td>
</tr>
<tr>
<td>kbeast-v1</td>
<td>syscall table, tcp4_seq_show</td>
<td>✓</td>
</tr>
<tr>
<td>mood-nt-2.3</td>
<td>syscall table</td>
<td>✓</td>
</tr>
<tr>
<td>override</td>
<td>syscall table</td>
<td>✓</td>
</tr>
<tr>
<td>phalanx-b6</td>
<td>syscall table, tcp4_seq_show</td>
<td>✓</td>
</tr>
<tr>
<td>rkit-1.01</td>
<td>syscall table</td>
<td>✓</td>
</tr>
<tr>
<td>rial</td>
<td>syscall table</td>
<td>✓</td>
</tr>
<tr>
<td>suckit-2</td>
<td>syscall table</td>
<td>✓</td>
</tr>
<tr>
<td>synapsys-0.4</td>
<td>IDT table</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>syscall table</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>syscall table</td>
<td>✓</td>
</tr>
</tbody>
</table>
### OS-Agnostic Testing

<table>
<thead>
<tr>
<th>Linux Distribution</th>
<th>Kernel Version</th>
<th>Release Date</th>
<th>Transparent?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Debian 4.0</td>
<td>2.6.26</td>
<td>2007-04-06</td>
<td>✓</td>
</tr>
<tr>
<td>Debian 5.0</td>
<td>2.6.28</td>
<td>2009-02-12</td>
<td>✓</td>
</tr>
<tr>
<td>Debian 6.0</td>
<td>2.6.32</td>
<td>2010-01-22</td>
<td>✓</td>
</tr>
<tr>
<td>Fedora-8</td>
<td>2.6.23</td>
<td>2007-11-08</td>
<td>✓</td>
</tr>
<tr>
<td>Fedora-10</td>
<td>2.6.27</td>
<td>2008-11-25</td>
<td>✓</td>
</tr>
<tr>
<td>Fedora-12</td>
<td>2.6.31</td>
<td>2009-11-17</td>
<td>✓</td>
</tr>
<tr>
<td>Fedora-14</td>
<td>2.6.35</td>
<td>2010-11-02</td>
<td>✓</td>
</tr>
<tr>
<td>Fedora-16</td>
<td>3.1.0</td>
<td>2011-11-08</td>
<td>✓</td>
</tr>
<tr>
<td>OpenSUSE-10.3</td>
<td>2.6.22</td>
<td>2007-10-04</td>
<td>✓</td>
</tr>
<tr>
<td>OpenSUSE-11.0</td>
<td>2.6.25</td>
<td>2008-06-19</td>
<td>✓</td>
</tr>
<tr>
<td>OpenSUSE-11.1</td>
<td>2.6.27</td>
<td>2008-12-18</td>
<td>✓</td>
</tr>
<tr>
<td>OpenSUSE-11.2</td>
<td>2.6.31</td>
<td>2009-11-12</td>
<td>✓</td>
</tr>
<tr>
<td>OpenSUSE-11.3</td>
<td>2.6.34</td>
<td>2010-07-15</td>
<td>✓</td>
</tr>
<tr>
<td>OpenSUSE-12.1</td>
<td>3.1.0</td>
<td>2011-11-16</td>
<td>✓</td>
</tr>
<tr>
<td>Ubuntu-8.04</td>
<td>2.6.24</td>
<td>2008-04-24</td>
<td>✓</td>
</tr>
<tr>
<td>Ubuntu-8.10</td>
<td>2.6.27</td>
<td>2008-10-30</td>
<td>✓</td>
</tr>
<tr>
<td>Ubuntu-9.04</td>
<td>2.6.28</td>
<td>2009-04-23</td>
<td>✓</td>
</tr>
<tr>
<td>Ubuntu-9.10</td>
<td>2.6.31</td>
<td>2009-10-29</td>
<td>✓</td>
</tr>
<tr>
<td>Ubuntu-10.04</td>
<td>2.6.32</td>
<td>2010-04-29</td>
<td>✓</td>
</tr>
<tr>
<td>Ubuntu-10.10</td>
<td>2.6.35</td>
<td>2010-10-10</td>
<td>✓</td>
</tr>
<tr>
<td>Ubuntu-11.04</td>
<td>2.6.38</td>
<td>2011-04-28</td>
<td>✓</td>
</tr>
<tr>
<td>Ubuntu-11.10</td>
<td>3.0.4</td>
<td>2011-10-13</td>
<td>✓</td>
</tr>
</tbody>
</table>
Limitations and Future Work

Limitations

- Can handle kernel ASLR
- Need an identical trusted kernel
- Need to stop the guest VM

Future Work

- Derandomize the kernel address space
- Port to Windows OS
Limitations and Future Work

Limitations
- Can handle kernel ASLR
- Need an identical trusted kernel
- Need to stop the guest VM

Future Work
- Derandomize the kernel address space
- Port to Windows OS
Conclusion

EXTERIOR is a novel dual-VM based external shell for trusted, native, out-of-VM program execution.
Conclusion

EXTERIOR is a novel dual-VM based external shell for trusted, native, out-of-VM program execution.

It can be used for (automatic) introspection, (re)configuration of the guest-OS state (in the cloud), and can perform a timely response such as recovery from a kernel malware intrusion.
EXTERIOR is a novel dual-VM based external shell for trusted, native, out-of-VM program execution.

It can be used for (automatic) introspection, (re)configuration of the guest-OS state (in the cloud), and can perform a timely response such as recovery from a kernel malware intrusion.

EXTERIOR has demonstrated a new program execution model on top of virtualization.
Conclusion

- EXTERIOR is a novel dual-VM based external shell for trusted, native, out-of-VM program execution.

- It can be used for (automatic) introspection, (re)configuration of the guest-OS state (in the cloud), and can perform a timely response such as recovery from a kernel malware intrusion.

- EXTERIOR has demonstrated a new program execution model on top of virtualization.

- (We believe) It will open new opportunities for system administration and security.
Thank you!

Secure VM (SVM)

User Space
Kernel Space

ps netstat kill p

User Space
Kernel Space

Guest VM (GVM)

apache mysql firefox

C_{user} D_{stack}
D_{global} P D_{heap}
Thank you!

Secure VM (SVM)

User Space
Kernel Space

ps netstat kill

Guest VM (GVM)

User Space
Kernel Space

apache mysql firefox

P

Contact us via. {yangchun.fu,zhiqiang.lin}@utdallas.edu for any questions