Space Traveling across VM
Automatically Bridging the Semantic-Gap in Virtual Machine Introspection via Online Kernel Data Redirection

Yangchun Fu, and Zhiqiang Lin

Department of Computer Sciences
The University of Texas at Dallas

May 23rd, 2012
1. Background and The Problem
2. State-of-the-Art
3. Our Approach: Data Space Traveling
4. Conclusion
Cloud Runs Virtual Machines (VM)

- **Windows XP**
  - Product-VM

- **Linux**
  - Product-VM
  - Files
  - MySQL

- **Win-7**
  - Product-VM
  - Microsoft Exchange

**Virtualization Layer**

**Hardware Layer**

**Consolidation, Multiplexing, Migration, Isolation, Encapsulation, Interposition, Security, Reliability, Dependability**...

**VMI** [Garfinkel and Rosenblum, NDSS’03]
Virtual Machine Introspection (VMI) [Garfinkel and Rosenblum]

Using a trusted, isolated, dedicated VM to monitor other VMs

- Intrusion Detection
- Malware Analysis
- Memory Forensics

Semantic Gap Problem
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
Example: Inspect pids of Guest Memory from VMM

In Kernel 2.6.18

struct task_struct {
  ...
  [188] pid_t pid;
  [192] pid_t tgid;
  ...
  [356] uid_t uid;
  [360] uid_t euid;
  [364] uid_t suid;
  [368] uid_t fsuid;
  [372] gid_t gid;
  [376] gid_t egid;
  [380] gid_t sgid;
  [384] gid_t fsgid;
  ...
  [428] char comm[16];
  ...
};
The Semantic Gap
[Chen et al, HotOS’01]

- In HotOS’01, Chen and Noble first raised the semantic gap problem in virtualization

“Services in the VM operate below the abstractions provided by the guest OS ... This can make it difficult to provide services.”
State-of-the-art

- VMI
  [Garfinkel et al, NDSS’03]

- The Semantic Gap
  [Chen et al, HotOS’01]

- In NDSS’03, Garfinkel et al. first proposed VMI, demonstrated for IDS
- Introspection routine is based on crash utility
In CCS’07, Jiang et al. proposed VMwatcher.

Introspection routine is based on manually created code.
In CCS’07, Petroni et al. proposed SBCFI.

Introspection routine is based on customized kernel source code.
In SP’11, Dolan-Gavitt et al. proposed Virtuoso.

- Introspection routine is based on the trained user level and kernel level code.
In SP’12, we propose VM Space Traveler (VMST).

Introspection routine is automatically generated from the native user level and kernel level code.
Key Idea

Data can be transferred
- In Internet, data is transferred though network packet

Insight
An inspection program $P(\mu, k)$ is often composed of static binary code $P$, runtime dynamic user-level data $\mu$ (including user-level stack, heap, and global variables), and inspected kernel data $k$.

- Transfer kernel space data $k$ from one machine to the other
  - `mov eax, [0x1c0eff08]`
Principles

\[ P'(\mu, k) = P(\mu, k'), \text{ where} \]
- \( P' \) is the new introspection program
- \( P \) is the old inspection program
- \( \mu \) is the user level data
- \( k \) is the kernel data being inspected
- \( k' \) is from other machine

Outcome

We reuse legacy binary code of \( P \) to automatically generate new program \( P' \)
How?

strace of a getpid program

1 execve("./getpid",..) = 0
2 brk(0) = 0x83b8000
3 access("/etc/ld.so.nohwcap",..) = -1
23 getpid() = 13849
26 write(1, "pid=13849\n", 10) = 10
27 exit_group(0) = ?

Three Key Components

- Syscall execution context identification
- Redirectable data identification
- Kernel data redirection
I. Syscall Execution Context Identification

One intuitive approach

**Hard-code** all the starting and ending PC of
- Interrupt
- Exception
- Context switch

Our OS-agnostic solution

- Instrument VMM interrupt/exception handler to capture the starting and ending point of interrupt/exception
- Disable the context switch by disabling the timer
II. Redirectable Data Identification

Challenges

- Identify kernel stack data (kernel control flow related)
- Differentiate kernel stack, heap, and global variable
- Differentiate kernel code and data

Our solution: a variant of dynamic data flow analysis

- Identify the kernel global and kernel heap (derived from kernel global), and redirect their memory access
- Alternatively, identify only the stack variable (derived from esp), and no redirection for them.
III. Kernel Data Redirection

The Algorithm

1: DynamicInstInstrument(i):
2:   if SysExecContext(s):
3:     if SysRedirect(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 NotDirty(α):
11:       CopyOnWritePage(α)
12:       UpdatePageEntryInSTLB(α)
13:    PA(α) ← V2P(α)
14:    Store(PA(α))
Architecture

Kernel
Data
Kernel
Code
Applications

Common Utilities

Syscall Execution Context Identification
Redirectable Data Identification
Kernel Data Redirection

Kernel

Secure-VM

VM-Space Traveler

Introspection

Applications

Kernel Data

Kernel Code

Product-VM
### Automatic VMI Tool Generation

<table>
<thead>
<tr>
<th>Utilities w/ options</th>
<th>Description</th>
<th>Syntax? (diff)</th>
<th>Semantics? (Manual)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ps -A</td>
<td>Reports a snapshot of all processes</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>lsmod</td>
<td>Shows the status of modules</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>lsof -c p</td>
<td>Lists opened files by a process p</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>ipcs</td>
<td>Displays IPC facility status</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>netstat -s</td>
<td>Displays network statistics</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>uptime</td>
<td>Reports how long the system running</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>ifconfig</td>
<td>Reports network interface parameters</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>uname -a</td>
<td>Displays system information</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>arp</td>
<td>Displays ARP tables</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>free</td>
<td>Displays amount of free memory</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>date</td>
<td>Print the system date and time</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>pidstat</td>
<td>Reports statistics for Linux tasks</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>mpstat</td>
<td>Reports CPU related statistics</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>iostat</td>
<td>Displays I/O statistics</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>vmstat</td>
<td>Displays VM statistics</td>
<td>✗</td>
<td>✓</td>
</tr>
</tbody>
</table>
Performance Overhead

<table>
<thead>
<tr>
<th>Benchmark Program</th>
<th>w/o VMI</th>
<th>w/ VMI</th>
</tr>
</thead>
<tbody>
<tr>
<td>ps</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>lsmod</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>ipcs</td>
<td>20%</td>
<td>20%</td>
</tr>
<tr>
<td>uptime</td>
<td>40%</td>
<td>40%</td>
</tr>
<tr>
<td>uname</td>
<td>60%</td>
<td>60%</td>
</tr>
<tr>
<td>ifconfig</td>
<td>80%</td>
<td>80%</td>
</tr>
<tr>
<td>arp</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>date</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>pidstat</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>mpstat</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>iostat</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>vmstat</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>netstat</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>ugetpid</td>
<td>0%</td>
<td>0%</td>
</tr>
</tbody>
</table>
## OS-Agnostic Testing

<table>
<thead>
<tr>
<th>Linux Distribution</th>
<th>Kernel Version</th>
<th>Release Date</th>
<th>OS-agnostic?</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Redhat-9</td>
<td>2.4.20-31</td>
<td>11/28/2002</td>
<td>✗</td>
<td>53</td>
</tr>
<tr>
<td></td>
<td>2.6.18-1.2798.fc6</td>
<td>10/14/2006</td>
<td>✗</td>
<td>53</td>
</tr>
<tr>
<td></td>
<td>2.6.38.6-26.rc1.fc15</td>
<td>05/09/2011</td>
<td>✓</td>
<td>0</td>
</tr>
<tr>
<td>Fedora-6</td>
<td>2.6.34-12-default</td>
<td>09/13/2010</td>
<td>✓</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>2.6.35</td>
<td>08/10/2010</td>
<td>✓</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>2.6.37.1-1.2-default</td>
<td>02/17/2011</td>
<td>✓</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>2.6.39.4</td>
<td>08/03/2011</td>
<td>✓</td>
<td>0</td>
</tr>
<tr>
<td>OpenSUSE-11.3</td>
<td>2.6.35</td>
<td>08/07/2004</td>
<td>✗</td>
<td>53</td>
</tr>
<tr>
<td>OpenSUSE-11.4</td>
<td>2.6.32-5</td>
<td>01/22/2010</td>
<td>✓</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>2.6.32-rc8</td>
<td>02/09/2010</td>
<td>✓</td>
<td>0</td>
</tr>
<tr>
<td>Debian 3.0</td>
<td>2.4.27-3</td>
<td>08/07/2004</td>
<td>✗</td>
<td>53</td>
</tr>
<tr>
<td>Debian 4.0</td>
<td>2.6.18-6</td>
<td>12/17/2006</td>
<td>✗</td>
<td>53</td>
</tr>
<tr>
<td></td>
<td>2.6.32-5</td>
<td>01/22/2010</td>
<td>✓</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>2.6.32-rc8</td>
<td>02/09/2010</td>
<td>✓</td>
<td>0</td>
</tr>
<tr>
<td>Ubuntu-4.10</td>
<td>2.6.8.1-3</td>
<td>08/14/2004</td>
<td>✗</td>
<td>53</td>
</tr>
<tr>
<td>Ubuntu-5.10</td>
<td>2.6.12-9</td>
<td>08/29/2005</td>
<td>✗</td>
<td>53</td>
</tr>
<tr>
<td></td>
<td>2.6.32-27</td>
<td>12/09/2010</td>
<td>✓</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>2.6.33</td>
<td>03/15/2010</td>
<td>✓</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>2.6.34</td>
<td>07/05/2010</td>
<td>✓</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>2.6.36</td>
<td>11/22/2010</td>
<td>✓</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>2.6.37.6</td>
<td>03/27/2010</td>
<td>✓</td>
<td>0</td>
</tr>
<tr>
<td>Ubuntu-10.04</td>
<td>2.6.38-8-generic</td>
<td>06/03/2011</td>
<td>✓</td>
<td>0</td>
</tr>
<tr>
<td>Ubuntu-11.04</td>
<td>3.0.0-12-generic</td>
<td>08/05/2011</td>
<td>✓</td>
<td>0</td>
</tr>
<tr>
<td>Ubuntu-11.10</td>
<td>3.2.0-12-generic</td>
<td>08/05/2011</td>
<td>✓</td>
<td>0</td>
</tr>
</tbody>
</table>
Limitations and Future Work

Limitations

- Need an identical trusted kernel
- Not entirely transparent to arbitrary OS kernels (relies on syscall knowledge)
- Non-blocking system call
- Does not inspect any disk data, memory swapped to disk

Future Work

- Kernel version inference in cloud VM
- Porting to Windows OS
- Addressing the non-blocking issue
VMST has **automatically bridged the semantic gap**, and **automatically generated the introspection tools** by reusing the legacy code (no training involved).

- It also enables **native VMI tool development**.

- (We hope) Cloud/VM/OS Providers, and AV-Software Vendors, could benefit from our techniques (for **VMI and memory forensics**).
Thank You

VM-Space Traveler

zhiqiang.lin@utdallas.edu