Malware Memory Forensics (MMF) Workshop at ACSAC’14

Automating Introspection and Forensics Software Development via Binary Code Reuse

Zhiqiang Lin

Department of Computer Sciences
The University of Texas at Dallas

December 9th, 2014
Virtualization, Hypervisor, and the Cloud

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

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

Multiplexing, Isolation, Migration, ...
On Anti-Virus Software Evolution

Operating Systems

Linux Kernel

Virtualization Layer
On Anti-Virus Software Evolution
On Anti-Virus Software Evolution

Operating Systems

Linux Kernel

Virtualization Layer
On Anti-Virus Software Evolution

Operating Systems

Linux Kernel

Virtualization Layer
On Anti-Virus Software Evolution

Operating Systems

Linux Kernel

Virtualization Layer
Virtual Machine Introspection (VMI) [Garfinkel et al. NDSS’03]

Introspection

Virtualization Layer

Hardware Layer

Linux

Win-7

Product-VM

Product-VM

Using a trusted, dedicated virtualization layer program to monitor the running VMs:
- Intrusion Detection
- Malware Analysis
- Memory Forensics

Virtual Machine Introspection (VMI)

Hardware Layer

Virtualization Layer

Product-VM

Linux

Win-7

Introspection

[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

Virtualization Layer

Hardware Layer

Introspection

Linux
- Apache
- MySQL
- Product-VM

Win-7
- Microsoft Exchange
- Product-VM
The Semantic Gap in Out-of-VM ([Chen and Noble HotOS'01])

- **Background**
- **Related Work**
- **Binary Code Reuse**
- **Evaluation**
- **Summary**

---

**Introspection**

**Product-VM**

**Linux**

View exposed by Virtual Machine Monitor is at low-level. There is no abstraction and no APIs. Need to reconstruct the guest-OS abstraction.
The Semantic Gap in Out-of-VM ([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

**32-bit General-Purpose Registers**
- EAX
- EBX
- ECX
- EDX
- EBP
- ESP
- ESI
- EDI

**16-bit Segment Registers**
- CS
- SS
- DS
- ES
- FS
- GS

**Virtual Machine Monitor Layer**

```
00001800  eb 40 1b 02 63 74 00 f0 00 00 00 00 00 00 00 00 |.@..ct..........|
00001810  00 00 00 00 80 00 00 00 00 00 00 00 00 00 00 |..............|
00001820  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |..............|
00001830  10 76 16 cc 00 00 00 00 00 |................|
00001840  00 19 66 8c d0 50 b8 08 00 00 00 66 8e d0 53 8b |...f..P..f.S.|
00001850  d9 ff 2d 19 02 00 00 0f 20 c0 0f ba f0 ff 22 |".............."|
00001860  c0 eb 00 b9 80 00 00 00 0f 32 0f ba f0 08 0f 30 |........2......|
00001870  0f 20 e0 0f ba f0 05 0f 22 e0 60 9c 8b d3 01 ea |"............."|
00001880  04 89 a3 76 02 00 00 00 0f 01 83 80 02 00 00 0f 01 |.........v......|
00001890  8b 88 02 00 00 8b 8b 3c 00 00 00 0b c9 74 12 8b |........t......|
000018a0  b3 38 00 00 00 8b fb 81 c7 00 00 00 02 b9 f9 f3 |.8......0+...|
000018b0  a4 0f 01 9b 90 02 00 00 0f 01 93 68 02 00 00 66 |.........h..f|
000018c0  b8 10 00 66 8e d8 66 8e cc 66 8e d0 66 8e e0 66 |...f..f..f.f|
```

Kernel specific data structure definition

```
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];
...}
```
Example: Inspect pids of Guest Memory from VMM

Virtual Machine Monitor Layer

Kernel specific data structure definition
Kernel symbols (global variable)
Virtual to physical (V2P) translation

In Kernel 2.6.18

```c
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];
    ...
};
```

SIZE: 1408
Example: Inspect pids of Guest Memory from VMM

In Kernel 2.6.18

```c
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];
    ...
}
```

SIZE: 1408
Example: Inspect pids of Guest Memory from VMM

In Kernel 2.6.18

```c
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];
    ...
}
```

**SUMMARY**

- **Kernel specific data structure definition**
- **Kernel symbols (global variable)**
- **Virtual to physical (V2P) translation**
How to bridge the semantic gap
How to bridge the semantic gap

Guest-OS

What hypervisor observes

The Semantic Gap

What we want

CPU-Reg  Phy-MEM  DISK  H-Event  Instruction  ...

Data-Type  Objects  Interrupts  Exceptions  K-events  Sys-call  Lib-call  ...

Background

Related Work

Binary Code Reuse

Evaluation

Summary
How to bridge the semantic gap

**Background**

**Related Work**

**Binary Code Reuse**

**Evaluation**

**Summary**

---

Guest-OS

- CPU-Reg
- Phy-MEM
- DISK
- H-Event
- Instruction
- ...

What hypervisor observes

- Data-structure
- Debug-Info
- Source-Code
- Binary-Code

With different constraints

**The Semantic Gap**

**Approaches**

- Manual
- Debugging-Tool
- Compiler-Assisted
- Binary Analysis

What we want

- Data-Type
- Objects
- Interrupts
- Exceptions
- K-events
- Sys-call
- Lib-call
- ...

---

**Guest-OS Assisted**
State-of-the-art in bridging the Semantic Gap

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."
In HotOS’01, Chen and Noble first raised the semantic gap problem in virtualization.
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 in bridging the Semantic Gap

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.
State-of-the-art in bridging the Semantic Gap

VMI
[Garfinkel et al, NDSS’03]

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

Copilot
[Petroni et al, Security’04]

In USENIX Security’04, Petroni et al. proposed Copilot, an introspection routine based on manually created code.
In USENIX Security’04, Petroni et al. proposed Copilot.

Introspection routine is based on manually created code.
State-of-the-art in bridging the Semantic Gap

- VMI [Garfinkel et al, NDSS’03]
- SBCFI [Petroni et al, CCS’07]
- The Semantic Gap [Chen et al, HotOS’01]
- Copilot [Petroni et al, Security’04]
State-of-the-art in bridging the Semantic Gap

- In CCS’07, Petroni et al. proposed SBCFI
- Introspection routine is based on customized kernel source code
In SP’08, Payne et al. proposed Lares. The Introspection routine is placed inside the guest OS with special help.
State-of-the-art in bridging the Semantic Gap

- VMI [Garfinkel et al, NDSS’03]
- SBCFI [Petroni et al, CCS’07]
- The Semantic Gap [Chen et al, HotOS’01]
- Copilot [Petroni et al, Security’04]
- Lares [Payne et al, SP’08]

- In SP’08, Payne et al. proposed Lares
- Introspection routine is placed inside the guest OS with special help
State-of-the-art in bridging the Semantic Gap

- **VMI** [Garfinkel et al, NDSS’03]
- **SBCFI** [Petroni et al, CCS’07]
- **Virtuoso** [Dolan-Gavitt et al., SP’11]
- **Lares** [Payne et al, SP’08]
- **Copilot** [Petroni et al, Security’04]
- **The Semantic Gap** [Chen et al, HotOS’01]

In SP’11, Dolan-Gavitt et al. proposed Virtuoso.

Introspection routine is based on the trained user level and kernel level code.
State-of-the-art in bridging the Semantic Gap

In SP’11, Dolan-Gavitt et al. proposed Virtuoso. Introspection routine is based on the trained user level and kernel level code.
State-of-the-art in bridging the Semantic Gap

- VMI [Garfinkel et al, NDSS’03]
- SBCFI [Petroni et al, CCS’07]
- VMST [Our solution, SP’12]

The Semantic Gap [Chen et al, HotOS’01]
Copilot [Petroni et al, Security’04]
Lares [Payne et al, SP’08]
Virtuoso [Dolan-Gavitt et al., SP’11]

In SP’12, we propose VM space traveling. Introspection routine is automatically generated from the native user level and kernel level code.
In SP’12, we propose VM space traveling.

Introspection routine is automatically generated from the native user level and kernel level code.
Approach Evolution

- **Debugger Assisted**
  - VMI
    - [Garfinkel et al, NDSS’03]
- **Compiler Assisted**
  - SBCFI
    - [Petroni et al, CCS’07]
  - VMST
    - [Our solution, SP’12]

- **Manual**
- **Guest Assisted**
- **Binary Assisted**

**Related Work**
Two Binary Code Reuse Based Approaches

Guest VM (GVM)

Memory

User Space

Kernel Space
Two Binary Code Reuse Based Approaches

Guest VM (GVM)

Memory

User Space

Kernel Space

Hypervisor
Two Binary Code Reuse Based Approaches

Using a trusted sibling VM to introspect the running guest VM.

1. Redirect kernel data [SP’12, VEE’13, NDSS’14]
2. Redirect system call execution [USENIX ATC’14]
Approach-I: Redirect Kernel Data [SP'12]

In-VM `getpid` Program

```c
#include <stdio.h>
#include <unistd.h>

int main()
{
    printf("pid=%d\n", getpid());
    return 0;
}
```

```
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)                = ?
```
Approach-I: Redirect Kernel Data [SP’12]

In-VM getpid Program

```
1 #include <stdio.h>
2 #include <unistd.h>
3
4 int main()
5 {
6     printf("pid=%d\n", getpid());
7     return 0;
8 }
```

Key Insight

- Reuse the execution context of a native process in SVM.
- When syscall `getpid` executed, redirects the data of interest from the guest VM.
Approach-I: Redirect Kernel Data [SP’12]
Approach-I: Redirect Kernel Data [SP’12]
Approach-I: Redirect Kernel Data [SP’12]

Secure VM (SVM)

Guest VM (GVM)

Hypervisor
Approach-I: Redirect Kernel Data [SP’12]

The diagram illustrates the Redirect Kernel Data approach, where the Memory of the Secure VM (SVM) and the Guest VM (GVM) is shown with the processes ps, netstat, kill, and P. The .data section contains binary code reused between the SVM and GVM, highlighted with the Hypervisor. The Hypervisor ensures secure execution by managing the binary code and processes.
Approach-I: Design & Implementation [SP’12]

Secure VM (SVM)
Approach-I: Design & Implementation [SP’12]
Approach-I: Design & Implementation [SP’12]

Memory

User Space

Outer Shell

Kernel Space

Kernel Code

D_{global}

D_{heap}

D_{stack1}

D_{stack2}

D_{stackn}

Kernel Syscall
Context Identification

Binary Translation Based Virtualization Layer

Kernel Data Identification and Redirection

GVM Memory Mapping and Address Resolution

Secure VM (SVM)
Approach-I: Design & Implementation [SP’12]

Memory

User Space

Kernel Space

D_{global}

D_{heap}

D_{stack1}  D_{stack2}  D_{stackn}

Kernel Code

Kernel Syscall

Context Identification

Binary Translation Based
Virtualization Layer

Kernel Data Identification
and Redirection

GVM Memory Mapping
and Address Resolution

Secure VM (SVM)

Xen/KVM/Vmware/VirtualBox/VirtualPC/HyperV/OpenVZ/QEMU

Guest VM (GVM)
Approach-II: Redirect System call Execution [ATC’14]

In-VM getpid Program

```c
1 #include <stdio.h>
2 #include <unistd.h>
3
4 int main()
5 {
6     printf("pid=%d\n", getpid());
7     return 0;
8 }
```

```
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) = ?
```

Key Insight
System call is the only interface to request OS service. Pushing the execution of `getpid` system call from SVM to GVM.
Approach-II: Redirect System call Execution [ATC’14]

In-VM getpid Program

```
1 #include <stdio.h>
2 #include <unistd.h>
3
4 int main()
5 {
6     printf("pid=%d\n", getpid());
7     return 0;
8 }
```

Key Insight

- System call is the only interface to request OS service.
- Pushing the execution of `getpid` system call from SVM to GVM.
Approach-II: Redirect System Call Execution [ATC’14]

Secure VM (SVM)

Guest VM (GVM)

Hypervisor
Approach-II: Redirect System call Execution [ATC’14]

Secure VM (SVM) vs. Guest VM (GVM)

- Memory
  - User Space
    - ps
    - netstat
    - kill
    - P
  - Kernel Space
    - .code
      - 1011001
      - 11110111
      - 10111001
      - 10111001
    - .data

- Memory
  - User Space
    - Hypervisor
    - Firefox
    - Skype
    - Virus
  - Kernel Space
    - .data
      - 11001
      - 10101010
      - 101110
      - 10101011
Approach-II: Redirect System call Execution [ATC’14]
Approach-II: Redirect System call Execution [ATC'14]

Secure VM (SVM)
- Memory
- User Space
- Kernel Space
  - ps
  - netstat
  - kill
  - P

Guest VM (GVM)
- Memory
- User Space
  - .code
    - 10111001
    - 11110111
    - 10111100
    - 10101011
- .data
  - 10111001

Hypervisor
- .code
  - 10111001
  - 11110111
  - 10111100
  - 10101011
Approach-II: Redirect System call Execution [ATC’14]

Secure VM (SVM)

- Memory
- ps
- netstat
- kill
- P

User Space
Kernel Space

.code
10111001
11110111
10111100
10101011

.data
10111001
11110111
10111100
10101011

Guest VM (GVM)

- Memory
- User Space
- Kernel Space

.code
10111001
11110111
10111100
10101011

.data
10111001
11110111
10111100
10101011

Hypervisor

.data
1110111
10111100
10101011
11010100
10011100
10101011

.code
10111001
11101001
10101000
10010100
10101011
Approach-II: Redirect System call Execution

Secure VM (SVM) vs Guest VM (GVM)

Memory
User Space
Kernel Space
.data
.code
Hypervisor

Memory
User Space
Kernel Space
.data
.code
Approach-II: Design & Implementation [ATC’14]

Kernel Space

User Space

Data

Control

Helper Process Creator

KVM

Kernel Space

Helper Process

Host OS

Guest VM (GVM)

Hardware Layer
Approach-II: Design & Implementation [ATC’14]
Approach-II: Design & Implementation [ATC’14]

User Space

Library Space

Kernel Space

Host OS

Guest VM (GVM)

Hardware Layer

Syscall Dispatcher

Syscall Data Exchanger

Helper Process Creator

Reverse Syscall Execution

Syscall Data Exchanger

KVM

Shared Memory
Approach-II: Design & Implementation [ATC’14]
Approach-II: Design & Implementation [ATC’14]
Approach-II: Design & Implementation [ATC’14]
I. Virtual Machine Introspection ([SP’12])

Normalized Performance Overhead

Benchmark Program

- ps
- lsmod
- ipcs
- uptime
- uname
- ifconfig
- arp
- date
- pidstat
- mpstat
- iostat
- vmstat
- netstat
- ugetpid

w/o VMI
w/ VMI
# II. Out-of-Box Attack Recovery, Repair

<table>
<thead>
<tr>
<th>Rootkit</th>
<th>Targeted Function Pointer</th>
<th>Repaired?</th>
</tr>
</thead>
<tbody>
<tr>
<td>adore-2.6</td>
<td>kernel global, heap object</td>
<td>x</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>syscall table</td>
<td>✓</td>
</tr>
</tbody>
</table>

Table: Rootkit Repairing with An Exterior [VEE’13] Tool.
III. Developing Out-of-VM Programs Natively

In-VM getpid Program

```c
#include <stdio.h>
#include <unistd.h>

int main() {
    printf("pid=%d\n", getpid());
    return 0;
}
```
IV. Writable VMI [VEE'13, ATC'14]

Advantages:
- Only install the management utilities at hypervisor layer.
- Automated, uniformed, and centralized management.
IV. Writable VMI [VEE’13, ATC’14]

Advantages
- Only install the management utilities at hypervisor layer.
- Automated, uniformed, and centralized management.
In-VM Management: Existing Approaches

Disadvantages:
- Scattered, distributed
- Install, update, and execute in each VM
In-VM Management: Existing Approaches

Hardware Layer

Virtualization Layer

Disadvantages
- Scattered, distributed
- Install, update, and execute in each VM
In-VM Management: Existing Approaches

Disadvantages
- Requiring the (admin) login password.
- Requiring install the management utilities in each VM.
In-VM Management: Existing Approaches

**Disadvantages**

- Requiring the (admin) login password.
- Requiring install the management utilities in each VM.
### Performance Impact: HyperShell [ATC’14]

<table>
<thead>
<tr>
<th>Process</th>
<th>$S$</th>
<th>$B(ms)$</th>
<th>$D(ms)$</th>
<th>$T(X)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ps</td>
<td>✓</td>
<td>1.33</td>
<td>5.42</td>
<td>4.08</td>
</tr>
<tr>
<td>pidstat</td>
<td>✓</td>
<td>1.95</td>
<td>7.56</td>
<td>3.88</td>
</tr>
<tr>
<td>nice</td>
<td>✓</td>
<td>0.07</td>
<td>0.11</td>
<td>1.57</td>
</tr>
<tr>
<td>getpid</td>
<td>✓</td>
<td>0.01</td>
<td>0.02</td>
<td>2.00</td>
</tr>
<tr>
<td>mpstat</td>
<td>✓</td>
<td>0.29</td>
<td>0.66</td>
<td>2.28</td>
</tr>
<tr>
<td>pstree</td>
<td>✓</td>
<td>0.69</td>
<td>6.03</td>
<td>8.74</td>
</tr>
<tr>
<td>chrt</td>
<td>✓</td>
<td>0.11</td>
<td>0.16</td>
<td>1.45</td>
</tr>
<tr>
<td>renice</td>
<td>✓</td>
<td>0.11</td>
<td>0.18</td>
<td>1.64</td>
</tr>
<tr>
<td>top</td>
<td>✓</td>
<td>504.92</td>
<td>510.85</td>
<td>1.01</td>
</tr>
<tr>
<td>nproc</td>
<td>✓</td>
<td>0.07</td>
<td>0.26</td>
<td>3.71</td>
</tr>
<tr>
<td>sleep</td>
<td>✓</td>
<td>1.27</td>
<td>1.28</td>
<td>1.01</td>
</tr>
<tr>
<td>pgrep</td>
<td>✓</td>
<td>0.89</td>
<td>4.72</td>
<td>5.30</td>
</tr>
<tr>
<td>pkill</td>
<td>✓</td>
<td>0.87</td>
<td>4.33</td>
<td>4.98</td>
</tr>
<tr>
<td>snice</td>
<td>✓</td>
<td>0.17</td>
<td>0.65</td>
<td>3.82</td>
</tr>
<tr>
<td>echo</td>
<td>✓</td>
<td>0.07</td>
<td>0.09</td>
<td>1.29</td>
</tr>
<tr>
<td>pwdx</td>
<td>✓</td>
<td>0.05</td>
<td>0.07</td>
<td>1.40</td>
</tr>
<tr>
<td>pmap</td>
<td>✓</td>
<td>0.16</td>
<td>0.36</td>
<td>2.25</td>
</tr>
<tr>
<td>kill</td>
<td>✓</td>
<td>0.01</td>
<td>0.04</td>
<td>4.00</td>
</tr>
<tr>
<td>killall</td>
<td>✓</td>
<td>0.62</td>
<td>3.03</td>
<td>4.89</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Memory</th>
<th>$S$</th>
<th>$B(ms)$</th>
<th>$D(ms)$</th>
<th>$T(X)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>free</td>
<td>✓</td>
<td>0.04</td>
<td>0.08</td>
<td>2.00</td>
</tr>
<tr>
<td>vmstat</td>
<td>✓</td>
<td>0.19</td>
<td>0.33</td>
<td>1.74</td>
</tr>
<tr>
<td>slabout</td>
<td>✓</td>
<td>0.22</td>
<td>0.36</td>
<td>1.64</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Modules</th>
<th>$S$</th>
<th>$B(ms)$</th>
<th>$D(ms)$</th>
<th>$T(X)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>rmmmd</td>
<td>✓</td>
<td>0.51</td>
<td>3.14</td>
<td>6.16</td>
</tr>
<tr>
<td>modinfo</td>
<td>✓</td>
<td>0.48</td>
<td>1.54</td>
<td>3.21</td>
</tr>
<tr>
<td>lsmod</td>
<td>✓</td>
<td>0.10</td>
<td>0.17</td>
<td>1.70</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Environment</th>
<th>$S$</th>
<th>$B(ms)$</th>
<th>$D(ms)$</th>
<th>$T(X)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>who</td>
<td>✓</td>
<td>0.14</td>
<td>0.72</td>
<td>5.14</td>
</tr>
<tr>
<td>env</td>
<td>✓</td>
<td>0.07</td>
<td>0.11</td>
<td>1.57</td>
</tr>
<tr>
<td>printenv</td>
<td>✓</td>
<td>0.07</td>
<td>0.1</td>
<td>1.43</td>
</tr>
<tr>
<td>whoami</td>
<td>✓</td>
<td>0.19</td>
<td>0.45</td>
<td>2.37</td>
</tr>
<tr>
<td>stty</td>
<td>✓</td>
<td>0.11</td>
<td>0.46</td>
<td>4.18</td>
</tr>
<tr>
<td>users</td>
<td>✓</td>
<td>0.09</td>
<td>0.53</td>
<td>5.89</td>
</tr>
<tr>
<td>uname</td>
<td>✓</td>
<td>0.09</td>
<td>0.11</td>
<td>1.22</td>
</tr>
<tr>
<td>id</td>
<td>✓</td>
<td>0.26</td>
<td>0.85</td>
<td>3.27</td>
</tr>
</tbody>
</table>

### Related Work

Evaluation

<table>
<thead>
<tr>
<th>System Util</th>
<th>$S$</th>
<th>$B(ms)$</th>
<th>$D(ms)$</th>
<th>$T(X)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>✓</td>
<td>0.11</td>
<td>0.12</td>
<td>1.09</td>
</tr>
<tr>
<td>w</td>
<td>✓</td>
<td>0.95</td>
<td>6.62</td>
<td>6.97</td>
</tr>
<tr>
<td>hostname</td>
<td>✓</td>
<td>0.04</td>
<td>0.06</td>
<td>1.50</td>
</tr>
<tr>
<td>groups</td>
<td>✓</td>
<td>0.21</td>
<td>0.62</td>
<td>2.95</td>
</tr>
<tr>
<td>hostid</td>
<td>✓</td>
<td>0.16</td>
<td>0.56</td>
<td>3.50</td>
</tr>
<tr>
<td>locale</td>
<td>✓</td>
<td>0.09</td>
<td>0.17</td>
<td>1.89</td>
</tr>
<tr>
<td>getconf</td>
<td>✓</td>
<td>0.09</td>
<td>0.34</td>
<td>3.78</td>
</tr>
<tr>
<td>link</td>
<td>✓</td>
<td>0.07</td>
<td>0.13</td>
<td>1.86</td>
</tr>
<tr>
<td>uptime</td>
<td>✓</td>
<td>0.07</td>
<td>0.47</td>
<td>6.71</td>
</tr>
<tr>
<td>syclt</td>
<td>✓</td>
<td>8.5</td>
<td>42.72</td>
<td>5.03</td>
</tr>
<tr>
<td>arch</td>
<td>✓</td>
<td>0.07</td>
<td>0.11</td>
<td>1.57</td>
</tr>
<tr>
<td>dmesg</td>
<td>✓</td>
<td>0.38</td>
<td>0.51</td>
<td>1.34</td>
</tr>
<tr>
<td>lscpu</td>
<td>✓</td>
<td>0.26</td>
<td>1.21</td>
<td>4.65</td>
</tr>
<tr>
<td>mcookie</td>
<td>✓</td>
<td>0.29</td>
<td>0.49</td>
<td>1.69</td>
</tr>
<tr>
<td>blkid</td>
<td>✓</td>
<td>0.14</td>
<td>0.61</td>
<td>4.36</td>
</tr>
<tr>
<td>badblocks</td>
<td>✓</td>
<td>0.35</td>
<td>0.44</td>
<td>1.26</td>
</tr>
<tr>
<td>lspci</td>
<td>✓</td>
<td>31.40</td>
<td>36.52</td>
<td>1.16</td>
</tr>
<tr>
<td>iostat</td>
<td>✓</td>
<td>0.45</td>
<td>1.04</td>
<td>2.31</td>
</tr>
<tr>
<td>du</td>
<td>✓</td>
<td>0.11</td>
<td>0.53</td>
<td>4.82</td>
</tr>
<tr>
<td>df</td>
<td>✓</td>
<td>0.16</td>
<td>0.35</td>
<td>2.19</td>
</tr>
<tr>
<td>blkid</td>
<td>✓</td>
<td>8.07</td>
<td>6.53</td>
<td>0.81</td>
</tr>
<tr>
<td>getcap</td>
<td>✓</td>
<td>0.04</td>
<td>0.08</td>
<td>2.00</td>
</tr>
<tr>
<td>lssof</td>
<td>✓</td>
<td>3.31</td>
<td>6.12</td>
<td>1.85</td>
</tr>
<tr>
<td>pwr</td>
<td>✓</td>
<td>0.07</td>
<td>0.11</td>
<td>1.57</td>
</tr>
<tr>
<td>chgrp</td>
<td>✓</td>
<td>0.19</td>
<td>0.47</td>
<td>2.47</td>
</tr>
<tr>
<td>chmod</td>
<td>✓</td>
<td>0.07</td>
<td>0.14</td>
<td>2.00</td>
</tr>
<tr>
<td>chown</td>
<td>✓</td>
<td>0.19</td>
<td>0.47</td>
<td>2.47</td>
</tr>
<tr>
<td>cp</td>
<td>✓</td>
<td>0.11</td>
<td>0.27</td>
<td>2.45</td>
</tr>
<tr>
<td>uniq</td>
<td>✓</td>
<td>0.09</td>
<td>0.35</td>
<td>3.89</td>
</tr>
<tr>
<td>file</td>
<td>✓</td>
<td>0.87</td>
<td>1.72</td>
<td>1.98</td>
</tr>
<tr>
<td>find</td>
<td>✓</td>
<td>0.20</td>
<td>0.58</td>
<td>2.90</td>
</tr>
<tr>
<td>grep</td>
<td>✓</td>
<td>0.35</td>
<td>2.14</td>
<td>6.11</td>
</tr>
<tr>
<td>ln</td>
<td>✓</td>
<td>0.08</td>
<td>0.14</td>
<td>1.75</td>
</tr>
<tr>
<td>ls</td>
<td>✓</td>
<td>0.14</td>
<td>0.27</td>
<td>1.93</td>
</tr>
<tr>
<td>ifconfig</td>
<td>✓</td>
<td>0.32</td>
<td>1.15</td>
<td>3.59</td>
</tr>
<tr>
<td>ip</td>
<td>✓</td>
<td>0.10</td>
<td>0.20</td>
<td>2.00</td>
</tr>
<tr>
<td>route</td>
<td>✓</td>
<td>138.65</td>
<td>150.32</td>
<td>1.08</td>
</tr>
<tr>
<td>ipmaddr</td>
<td>✓</td>
<td>0.13</td>
<td>0.34</td>
<td>2.62</td>
</tr>
<tr>
<td>iptunnel</td>
<td>✓</td>
<td>0.09</td>
<td>0.29</td>
<td>3.22</td>
</tr>
<tr>
<td>nameif</td>
<td>✓</td>
<td>0.10</td>
<td>0.21</td>
<td>2.10</td>
</tr>
<tr>
<td>netstat</td>
<td>✓</td>
<td>0.25</td>
<td>0.37</td>
<td>1.48</td>
</tr>
<tr>
<td>arp</td>
<td>✓</td>
<td>0.14</td>
<td>0.24</td>
<td>1.71</td>
</tr>
<tr>
<td>ping</td>
<td>✓</td>
<td>15.02</td>
<td>18.2</td>
<td>1.21</td>
</tr>
</tbody>
</table>

| Avg.        |     | 7.27    | 8.45    | 2.73   |
Disk Introspection: FDE disk virus scanning [ATC’14]
Disk Introspection: FDE disk virus scanning [ATC’14]

1. Encrypted by dm-crypt
2. 101,415 files
3. 1336.09 megabytes in size

Clamav successfully detect two viruses!!
Disk Introspection: FDE disk virus scanning [ATC'14]

Clamav successfully detect two viruses!!

1. Encrypted by dm-crypt
2. 101,415 files
3. 1336.09 megabytes in size
## Comparison with the most related work

<table>
<thead>
<tr>
<th>Systems</th>
<th>Execution Context Reuse</th>
<th>wo/ Dual-VM Architecture</th>
<th>wo/ Identical Kernel</th>
<th>wo/ Trust to Guest Kernel</th>
<th>High Code Coverage</th>
<th>Fully Automated</th>
<th>Memory Introspection</th>
<th>Disk Introspection</th>
<th>Guest Management</th>
<th>Process Monitoring</th>
</tr>
</thead>
<tbody>
<tr>
<td>VIRTUOSO</td>
<td>X</td>
<td>✓</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>VMST</td>
<td>✓</td>
<td>X</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>EXTERIOR</td>
<td>✓</td>
<td>X</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>PROCESSIMPLANTING</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>PROCESSOUTGRAFTING</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>✓</td>
</tr>
<tr>
<td>GEARS</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>GEARSPACK</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>HYPERSHELL</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Virtualization (Hypervisor) Layer Applications

- Virtual Machine Introspection
- Virtual Machine (Re)Configuration, Repair
- Automated Out-of-VM Management
Two Approaches to bridging the semantic gap

Secure VM (SVM)

Kernel Space

ps  netstat  kill  p

User Space

Guest VM (GVM)

Kernel Space

apache  mysql  firefox

User Space

Redirect kernel data [SP’12, VEE’13, NDSS’14] → Fine-grained, slower performance

Redirect system call execution [USENIX ATC’14] → More practical, fast performance
Two Approaches to bridging the semantic gap

Reusing (legacy) binary code with a trusted sibling VM to introspect the running guest VM.

1. Redirect kernel data [SP’12, VEE’13, NDSS’14] → Fine-grained, slower performance
For Cloud Developers: No Gap, Everything is Native

In-VM getpid Program

```c
#include <stdio.h>
#include <unistd.h>

int main()
{
    printf("pid=%d\n", getpid());
    return 0;
}
```
## References

<table>
<thead>
<tr>
<th></th>
<th>Reference</th>
<th>Conference/Book</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>VM Space Traveling: Automatically Bridging the Semantic Gap in Virtual Machine Introspection via Online Kernel Data Redirection</td>
<td><em>IEEE Symposium on Security and Privacy [SP’12]</em></td>
</tr>
<tr>
<td>2</td>
<td>EXTERIOR: Using A Dual-VM Enabled External Shell for Guest-OS Introspection, Configuration, and Recovery</td>
<td><em>ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments [VEE’13]</em></td>
</tr>
<tr>
<td>3</td>
<td>Hybrid-Bridge: Efficiently Bridging the Semantic-Gap in Virtual Machine Introspection via Decoupled Execution and Training Memoization</td>
<td><em>Network and Distributed System Symposium [NDSS’14]</em></td>
</tr>
<tr>
<td>4</td>
<td>HyperShell: A Practical Hypervisor Layer Guest OS Shell for Automated In-VM Management</td>
<td><em>USENIX Annual Technical Conference [ATC’14]</em></td>
</tr>
</tbody>
</table>

http://www.utdallas.edu/~zhiqiang.lin/s3.html