VMI-PL: A monitoring language for virtual platforms using virtual machine introspection

Florian Westphal ¹,²  Stefan Axelsson ¹  Christian Neuhaus ²
Andreas Polze ²

¹Blekinge Institute of Technology
Karlskrona, Sweden

²Hasso Plattner Institute
Potsdam, Germany

August 5, 2014
Outline

1. Introduction
   - Virtual Machine Introspection (VMI)
   - VMI Techniques

2. Virtual Machine Introspection Probe Language (VMI-PL)
   - Overview
   - Language Constructs
   - Demo

3. Evaluation
   - Measurement Scenarios
   - VMI-PL Performance
   - Comparison between VMI-PL and VMware VProbes

4. Conclusion
Virtual machine introspection (VMI) describes the method of monitoring and analyzing the state of a virtual machine from the hypervisor level.

— Jonas Pfoh [Pfoh et al., 2009]
Virtual Machine Introspection (VMI)

“Virtual machine introspection (VMI) describes the method of monitoring and analyzing the state of a virtual machine from the hypervisor level.”

— Jonas Pfoh [Pfoh et al., 2009]

First Described by Garfinkel and Rosenblum [Garfinkel et al., 2003]
Virtual Machine Introspection (VMI)

“Virtual machine introspection (VMI) describes the method of monitoring and analyzing the state of a virtual machine from the hypervisor level.”

— Jonas Pfoh [Pfoh et al., 2009]

- First Described by Garfinkel and Rosenblum [Garfinkel et al., 2003]
- Semantic gap problem [Chen et al., 2001]
VMI Techniques

Data-based techniques:
- Memory introspection
- Register introspection

Event-based techniques:
- Monitor writes to certain registers
- Monitor system and user defined interrupts

Stream-based techniques:
- Inspect network traffic
- Monitor keyboard input
VMI Techniques

Data-based techniques:
VMI Techniques

Data-based techniques:

- Memory introspection
VMI Techniques

Data-based techniques:

- Memory introspection
- Register introspection
VMI Techniques

Data-based techniques:
- Memory introspection
- Register introspection

Event-based techniques:
VMI Techniques

Data-based techniques:
- Memory introspection
- Register introspection

Event-based techniques:
- Monitor writes to certain registers
VMI Techniques

Data-based techniques:
- Memory introspection
- Register introspection

Event-based techniques:
- Monitor writes to certain registers
- Monitor system and user defined interrupts
VMI Techniques

Data-based techniques:
- Memory introspection
- Register introspection

Event-based techniques:
- Monitor writes to certain registers
- Monitor system and user defined interrupts

Stream-based techniques:
VMI Techniques

Data-based techniques:
- Memory introspection
- Register introspection

Event-based techniques:
- Monitor writes to certain registers
- Monitor system and user defined interrupts

Stream-based techniques:
- Inspect network traffic
VMI Techniques

Data-based techniques:
- Memory introspection
- Register introspection

Event-based techniques:
- Monitor writes to certain registers
- Monitor system and user defined interrupts

Stream-based techniques:
- Inspect network traffic
- Monitor keyboard input
Outline

1. Introduction
   - Virtual Machine Introspection (VMI)
   - VMI Techniques

2. Virtual Machine Introspection Probe Language (VMI-PL)
   - Overview
   - Language Constructs
   - Demo

3. Evaluation
   - Measurement Scenarios
   - VMI-PL Performance
   - Comparison between VMI-PL and VMware VProbes

4. Conclusion
VMI-PL:

Data probes
Event probes
Stream probes
Filters
Reconfiguration instructions
Configuration
VMI-PL:

- Data probes
- Event probes
- Stream probes
VMI-PL:
- Data probes
- Event probes
- Stream probes
- Filters
- Reconfiguration instructions
VMI-PL:

- Data probes
- Event probes
- Stream probes
- Filters
- Reconfiguration instructions
- Configuration
Configuration{
   <configuration-item>
 }
<event-probe>(<parameters>){
   <data-probe>(<parameters>)
   <filter>(<parameters>){
      <data-probe>(<parameters>)
   }
 }
<stream-probe>(<parameters>)
Data Probes:

- ReadMemory(<virtual address> <size>, [<file-path> | <stream-id>])
- WriteToMemoryAt(<register>, <offset>, <value>)

Event Probes:

- CRWrite(<number>)
- Syscall([<syscall number> | -])

Stream Probes:

- CaptureNetwork(<mac address>, [<file-path> | <stream-id>])
- CaptureKeyboardInput([<file-path> | <stream-id>])
Data Probes:

- `ReadMemory(<virtual address> <size>, [<file-path> | <stream-id>])`
Data Probes:

- ReadMemory(<virtual address> <size>, [<file-path> | <stream-id>])
- WriteToMemoryAt(<register>, <offset>, <value>)
Data Probes:

- `ReadMemory(<virtual address> <size>, [<file-path> | <stream-id>])`
- `WriteToMemoryAt(<register>, <offset>, <value>)`
- `ProcessList(<field>, [<file-path> | <stream-id>])`
Data Probes:
- ReadMemory(<virtual address> <size>, [<file-path> | <stream-id>])
- WriteToMemoryAt(<register>, <offset>, <value>)
- ProcessList(<field>, [<file-path> | <stream-id>])

Event Probes:
- CRWrite(<number>)
Data Probes:
- `ReadMemory(<virtual address> <size>, [<file-path> |
  <stream-id>])`
- `WriteToMemoryAt(<register>, <offset>, <value>)`
- `ProcessList(<field>, [<file-path> | <stream-id>])`

Event Probes:
- `CRWrite(<number>)`
- `Syscall([<syscall number> | -])`
Data Probes:
- `ReadMemory(<virtual address> <size>, [<file-path> | <stream-id>])`
- `WriteToMemoryAt(<register>, <offset>, <value>)`
- `ProcessList(<field>, [<file-path> | <stream-id>])`

Event Probes:
- `CRWrite(<number>)`
- `Syscall([<syscall number> | -])`

Stream Probes:
- `CaptureNetwork(<mac address>, [<file-path> | <stream-id>])`
Data Probes:
- `ReadMemory(<virtual address> <size>, [<file-path> | <stream-id>])`
- `WriteToMemoryAt(<register>, <offset>, <value>)`
- `ProcessList(<field>, [<file-path> | <stream-id>])`

Event Probes:
- `CRWrite(<number>)`
- `Syscall([<syscall number> | -])`

Stream Probes:
- `CaptureNetwork(<mac address>, [<file-path> | <stream-id>])`
- `CaptureKeyboardInput([<file-path> | <stream-id>])`
Filters:
- RegisterHasValue(<register>, <value>)
Filters:

- RegisterHasValue(<register>, <value>)

Reconfiguration Instructions:

- Pause
Filters:
- RegisterHasValue(<register>, <value>)

Reconfiguration Instructions:
- Pause
- Reconfigure(<event probe id>)
Configuration{
    ProcessListHead: 0xc17cdfe0
    TasksOffset: 440
    PIDOffset: 520
    ProcessNameOffset: 744
    MMStructOffset: 468
    ExeFileOffset: 444
    DEntryOffset: 12
    ParentOffset: 16
    DNameOffset: 28
    PGDOffset: 40
}
CRWrite(3){
    ReadRegister(CR3, #demo)
}
ExecuteAt(0xc104f060){
    ProcessList(PID, NAME, PATH, PGDP, #demo)
    ReadRegister(CR3, #demo)
}
CaptureNetwork(00:16:35:AF:94:4B, #demo)
Outline

1. Introduction
   - Virtual Machine Introspection (VMI)
   - VMI Techniques

2. Virtual Machine Introspection Probe Language (VMI-PL)
   - Overview
   - Language Constructs
   - Demo

3. Evaluation
   - Measurement Scenarios
   - VMI-PL Performance
   - Comparison between VMI-PL and VMware VProbes

4. Conclusion
Evaluation - Measurement Scenarios

- Without monitoring support
Evaluation - Measurement Scenarios

- Without monitoring support
- Monitoring support enabled (VMI-PL, VProbes)
Evaluation - Measurement Scenarios

- Without monitoring support
- Monitoring support enabled (*VMI-PL*, *VProbes*)
- Process life cycle monitoring (*LC*)
Evaluation - Measurement Scenarios

- Without monitoring support
- Monitoring support enabled (VMI-PL, VProbes)
- Process life cycle monitoring (LC)
- Process execution monitoring (Exec)
UnixBench Benchmark
VMI-PL Apache Build Slowdown

Slowdown in %

VMI-PL
LC
Exec
UnixBench Benchmark

Performance in %

- VMI-PL
- VProbes

VMI-PL/ VProbes
LC
Exec

F. Westphal (BTH, HPI)
1. Introduction
   - Virtual Machine Introspection (VMI)
   - VMI Techniques

2. Virtual Machine Introspection Probe Language (VMI-PL)
   - Overview
   - Language Constructs
   - Demo

3. Evaluation
   - Measurement Scenarios
   - VMI-PL Performance
   - Comparison between VMI-PL and VMware VProbes

4. Conclusion
Conclusion

- Classification scheme for VMI techniques
Conclusion

- Classification scheme for VMI techniques
- Simple description language for VMI
Conclusion

- Classification scheme for VMI techniques
- Simple description language for VMI
  - Support for data, event, and stream-based techniques
Conclusion

- Classification scheme for VMI techniques
- Simple description language for VMI
  - Support for data, event, and stream-based techniques
  - Support for VM state manipulation

Prototype for KVM
Available at
https://github.com/FlorianWestphal/VMI-PL

F. Westphal (BTH, HPI)
VMI-PL - DFRWS USA 2014
August 5, 2014
Conclusion

- Classification scheme for VMI techniques
- Simple description language for VMI
  - Support for data, event, and stream-based techniques
  - Support for VM state manipulation
  - Support for retrieving operating system level information

Prototype for KVM
Available at
https://github.com/FlorianWestphal/VMI-PL
Conclusion

- Classification scheme for VMI techniques
- Simple description language for VMI
  - Support for data, event, and stream-based techniques
  - Support for VM state manipulation
  - Support for retrieving operating system level information
- Prototype for KVM
  - Available at https://github.com/FlorianWestphal/VMI-PL
References

J. Pfoh, Ch. Schneider, C. Eckert (2009)
A Formal Model for Virtual Machine Introspection.

A Virtual Machine Introspection Based Architecture for Intrusion Detection.

P.M. Chen, B.D. Noble (2001)
When virtual is better than real [operating system relocation to virtual machines].