VirtFlow: Guest Independent Execution Flow Analysis Across Virtualized Environment

Hani Nemati

Dec 12, 2016

Polytechnique Montréal
Laboratoire DORSAL
Agenda

Motivation

• Why analyzing nested VM is important?

Investigations

• Nested Virtual Machine Process State Transition
• Any-Level VM Detection Algorithm
• Nested VM State Detection (NSD) Algorithm
• Performance Analysis and Evaluation
  • Execution Flow Analysis
  • CPU Cap and CPU Overcommitment Problem
  • Overhead of Virtualization Layer for different Types of Workload
  • Overhead of Virtualization for different Operating System

Conclusion and in-progress
What is Nested Virtual Machine?

- Root (L0) - Code of Host Hypervisor
- Non-Root (L1) - Code of VM Hypervisor
- Non-Root (L2) - Code of Last Level VM
Motivation

Why analyzing Nested VM is important?

- Compatibility issues (e.g., Windows 7)
- Security Concern (e.g., McAfee)
- Software Scaling (e.g., SaaS)
- Continuous Integration (CI)
Investigations

What is our goal in this project?????

To find Execution Flow of Virtual Machine
Investigations

How does it work internally? (Host Tracing)

L2

L1

L0

pCPU

PUSH CR3 TO GUEST
HYPervisor candida te
LIST

KVM NESTED_VMEXIT

VM RESUME
VM LUNCH

HLT

HLT
Investigations

What if we only have access to L1? (VM (L1) Tracing)

Tracing L1 - Wrong Result !!!!!!!

Tracing L0
Investigations

Level Detection Result:

- Calculating first 10000 prime numbers:
Investigations

Preemption in different levels:

Two Nested VMs are preemting

Nested VM is being preempted by a process inside L1
Investigations

Nested Virtual Machine Process State Transition
Investigations

N-Level Nested Virtual Machine Process State Transition
Investigations

Two Nested VMs are preemption each other inside a VM
Investigations

A VM preempts a Nested VM
Investigations

Two Nested VMs and One VM are preempting each other
Investigations

Overhead of Virtualization layers for different types of Workload

![Bar chart showing overhead of virtualization layers for different types of workload. The chart compares different levels (L0, L1, L2) of overhead for Nested CPU-Intensive, Nested I/O-Intensive, Non-Nested CPU-Intensive, and Non-Nested I/O-Intensive workloads.]
Investigations

Average wake up latency for Nested VM and Normal VM
# Investigations

Execution time for Fibo program on different level of virtualization and different OS

<table>
<thead>
<tr>
<th>Experiment</th>
<th>$T_{L_0}$ (ms)</th>
<th>$T_{L_1}$ (ms)</th>
<th>$T_{L_2}$ (ms)</th>
<th>$U$ (%)</th>
<th>$O$ (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nested-Linux</td>
<td>18.779</td>
<td>4.728</td>
<td>1539.45</td>
<td>98.5</td>
<td>23.507</td>
</tr>
<tr>
<td>Nested-Windows</td>
<td>439.864</td>
<td>283.582</td>
<td>1653.62</td>
<td>69.5</td>
<td>723.446</td>
</tr>
<tr>
<td>VM-Linux</td>
<td>5.623</td>
<td>1512.18</td>
<td>-</td>
<td>99.6</td>
<td>5.623</td>
</tr>
<tr>
<td>VM-Windows</td>
<td>216.362</td>
<td>1569.1</td>
<td>-</td>
<td>88.1</td>
<td>216.362</td>
</tr>
<tr>
<td>Host</td>
<td>1508.75</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>-</td>
</tr>
</tbody>
</table>
Demo
What do you need to test this project?

- Access to **Host** only
- Run **LTTng** on Host with my new added tracepoint (**vcpu_enter_guest**)
- Clone **TraceCompass** from my github (**virtFlow**)
  - [https://github.com/Nemati](https://github.com/Nemati)
- Open Resource View of TraceCompass
One More Thing ...
VM PT
Intel Processor Trace
VM PT Packets

- **PIP** (Paging Information Packet): CR3 change
  - NR bit (VM Entry, VM Exit)
- **VMCS** (Unique base pointer per vCPU): Successful VMPTRLD
- **Timing**
  - TSC: Time Stamp Counter
  - MTC: Mini Time Counter
Using Intel PT

- **CPU**
  - PT
  - Trace Control

- **Kernel**
  - Trace Record

- **Userspace**

- **Trace Control and Translation Module**
  - Trace Extractor
    - Decoding
    - vmpt.xml (Intermediate Format)

- **Trace Analysis**
  - HAVAna
  - Events

- **VM Visualization**
  - TraceCompass
Algorithm 1 HAVAna Algorithm

1: procedure HAVAna(Input: Event Packets \( P_e[i] \) from IF Output:
   
   Updated SHT)
2: \( SE[i] = parseXML(P_e[i]) \)
3: if \( (SE[i].name == VMCS) \) then
4:     Modify Status attribute of \( SE[i].base \) as \( VMM \)
5: else if \( (SE[i].name == PIP) \) then
6:     if \( (SE[i].NR == 1) \) then
7:         Query Status attribute of current running base
8:         Modify Status attribute as \( VM \)
9:         Modify VMM Status as \( IDLE \)
10:        Modify Status attribute of \( SE[i].cr3 \) as \( VM \)
11: else if (Query Status attribute of \( SE[i].base == VMM \)) then
12:        Query Status attribute of current running base
13:        Modify \( base \) Status as \( IDLE \)
14:        Modify VM Status as \( IDLE \)
15:        Modify VMM Status as \( IDLE \)
16: else
17:        Query Status attribute of current running base
18:        Modify \( base \) Status as \( VMM \)
19:        Modify Status attribute of \( SE[i].cr3 \) as \( VMM \)
20:        Modify VM Status as \( IDLE \)
21: end if
22: end if
23: end procedure
VM Analysis

- 4 vCPUs are pinned to one pCPU
- 4 Threaded application to calculate Prime numbers
VM Analysis

- 4 vCPUs are pinned to one pCPU
- MD5 hashing with 3 workers
Conclusion and in-progress

Inferences

- Any-Level VM Detection Algorithm (ADA)
- Nested VM State Detection (NSD) algorithm
- Calculate overhead of virtualization layers for different types of workload
- Analysis wake up latency for VM and Nested VM
- Analysis behavior of KVM for different OS

Going Further

- Wait Analyzing of process inside VM and Nested VM
Outcome of this project


Questions?

Hani.nemati@polymtl.ca
https://github.com/Nemati