VM Analysis and Hardware Trace Reconstruction

Suchakrapani Datt Sharma

Dec 12, 2016

École Polytechnique de Montréal
Laboratoire DORSAL
Agenda

Introduction

• Research Updates

New Investigations

• Intel PT
  • Advanced Analysis of VMs
• Trace Reconstruction Issues
  • Failed & Incorrect Decoding
  • Kernel Assisted Reconstruction

Upcoming

• Kernel Patches
Introduction

Research Focus: Hardware tracing on Intel and ARM for low overhead and high accuracy tracing and profiling

Research Updates

- Advanced VM analysis with hardware tracing
- FlowJIT: A robust hardware trace reconstruction technique
Hardware Tracing with PT

Configure and Enable PT → CPU → Intel PT Hardware → Intel PT Packets → Intel PT Software Decoder → Reconstructed Execution Flow → Binary → Runtime Data

Based on, Andi Kleen’s Presentation (TracingSummit 2015)
Intel PT

Hardware Trace Packets (Perf)

... Intel Processor Trace data: size 8544 bytes
00000000: 02 82 02 82 02 82 02 82 02 82 02 82 02 82 02 82 PSB
00000010: 00 00 00 00 00 00 PAD
00000016: 19 ba 39 4d 7b 89 5e 04 TSC 0x45e897b4d39ba
0000001e: 00 00 00 00 00 00 00 00 PAD
00000026: 02 73 57 64 00 1c 00 00 TMA CTC 0x6457 FC 0x1c
0000002e: 00 00 PAD
00000030: 02 03 27 00 CBR 0x27
00000034: 02 23 PSBEND
00000036: 59 8b MTC 0x8b
00000038: 59 8c MTC 0x8c

000000304: f8 TNT TTTTNN (6)
000000305: 06 00 00 TNT T (1)
000000308: 4d e0 3c 6d 9c TIP 0x9c6d3ce0
00000030d: 1c 00 00 TNT TTN (3)
000000310: 2d f0 3c TIP 0x3cf0
000000313: 06 TNT T (1)
000000314: 59 2e MTC 0x2e
000000316: 94 TNT NNTNTN (6)
000000317: a8 TNT NTNTNN (6)
000000318: a6 TNT TNTNTNN (6)
Intel PT

Timing

... Intel Processor Trace data: size 8544 bytes

- 00000000: 02 82 02 82 02 82 02 82 02 82 02 82 02 82 02 82 PSB

- 00000010: 00 00 00 00 00 00 PAD

- 00000016: 19 ba 39 4d 7b 89 5e 04 TSC 0x45e897b4d39ba

- 0000001e: 00 00 00 00 00 00 PAD

- 00000026: 02 73 57 64 00 1c 00 00 TMA CTC 0x6457 FC 0x1c

- 0000002e: 00 00 PAD

- 00000030: 02 03 27 00 CBR 0x27

- 00000034: 02 23 PSBEND

- 00000036: 59 8b MTC 0x8b

- 00000038: 59 8c MTC 0x8c

- 00000304: f8 TNT TTTTNN (6)

- 00000305: 06 00 00 TNT T (1)

- 00000308: 4d e0 3c 6d 9c TIP 0x9c6d3ce0

- 0000030d: 1c 00 00 TNT TTN (3)

- 00000310: 2d f0 3c TIP 0x3cf0

- 00000313: 06 TNT T (1)

- 00000314: 59 2e MTC 0x2e

- 00000316: 94 TNT NNTNTN (6)

- 00000317: a8 TNT NTNTNTN (6)

- 00000318: a6 TNT NTNNTT (6)
### Conditional Branches

... Intel Processor Trace data: size 8544 bytes

<table>
<thead>
<tr>
<th>Address</th>
<th>Trace Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000:</td>
<td>02 82 02 82 02 82 02 82 02 82 02 82 02 82 02 82 02 82</td>
<td>PSB</td>
</tr>
<tr>
<td>00000010:</td>
<td>00 00 00 00 00 00 00</td>
<td>PAD</td>
</tr>
<tr>
<td>00000016:</td>
<td>19 ba 39 4d 7b 89 5e 04</td>
<td>TSC 0x45e897b4d39ba</td>
</tr>
<tr>
<td>0000001e:</td>
<td>00 00 00 00 00 00 00</td>
<td>PAD</td>
</tr>
<tr>
<td>00000026:</td>
<td>02 73 57 64 00 1c 00 00</td>
<td>TMA CTC 0x6457 FC 0x1c</td>
</tr>
<tr>
<td>0000002e:</td>
<td>00 00</td>
<td>PAD</td>
</tr>
<tr>
<td>00000030:</td>
<td>02 03 27 00</td>
<td>CBR 0x27</td>
</tr>
<tr>
<td>00000034:</td>
<td>02 23</td>
<td>PSBEND</td>
</tr>
<tr>
<td>00000036:</td>
<td>59 8b</td>
<td>MTC 0x8b</td>
</tr>
<tr>
<td>00000038:</td>
<td>59 8c</td>
<td>MTC 0x8c</td>
</tr>
<tr>
<td>0000003a:</td>
<td>f8</td>
<td>TNT TTTTNN (6)</td>
</tr>
<tr>
<td>0000003b:</td>
<td>06 00 00</td>
<td>TNT T (1)</td>
</tr>
<tr>
<td>0000003c:</td>
<td>4d e0 3c 6d 9c</td>
<td>TIP 0x9c6d3ce0</td>
</tr>
<tr>
<td>0000003d:</td>
<td>1c 00 00</td>
<td>TNT TTN (3)</td>
</tr>
<tr>
<td>0000003e:</td>
<td>2d f0 3c</td>
<td>TIP 0x3cf0</td>
</tr>
<tr>
<td>0000003f:</td>
<td>06</td>
<td>TNT T (1)</td>
</tr>
<tr>
<td>00000040:</td>
<td>59 2e</td>
<td>MTC 0x2e</td>
</tr>
<tr>
<td>00000041:</td>
<td>94</td>
<td>TNT NNTNTN (6)</td>
</tr>
<tr>
<td>00000042:</td>
<td>a8</td>
<td>TNT NTNTNN (6)</td>
</tr>
<tr>
<td>00000043:</td>
<td>a6</td>
<td>TNT NTNNNT (6)</td>
</tr>
</tbody>
</table>
## Indirect Branches

... Intel Processor Trace data: size 8544 bytes

<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000</td>
<td>02 82 02 82 02 82 02 82 02 82 02 82 02 82 02 82 02 82 PSB</td>
</tr>
<tr>
<td>00000010</td>
<td>00 00 00 00 00 00 00 00 PAD</td>
</tr>
<tr>
<td>00000016</td>
<td>19 ba 39 4d 7b 89 5e 04 TSC 0x45e897b4d39ba PAD</td>
</tr>
<tr>
<td>0000001e</td>
<td>00 00 00 00 00 00 00 00 TMA CTC 0x6457 FC 0x1c PAD</td>
</tr>
<tr>
<td>00000026</td>
<td>02 73 57 64 00 1c 00 00</td>
</tr>
<tr>
<td>0000002e</td>
<td>00 00 CBR 0x27</td>
</tr>
<tr>
<td>00000030</td>
<td>02 03 27 00 PSBEND</td>
</tr>
<tr>
<td>00000034</td>
<td>02 23 MTC 0x8b</td>
</tr>
<tr>
<td>00000036</td>
<td>59 8b MTC 0x8c</td>
</tr>
<tr>
<td>00000038</td>
<td>59 8c</td>
</tr>
<tr>
<td>00000304</td>
<td>f8 TNT TTTTNN (6)</td>
</tr>
<tr>
<td>00000305</td>
<td>06 00 00 TNT T (1)</td>
</tr>
<tr>
<td>00000308</td>
<td>4d e0 3c 6d 9c TIP 0x9c6d3ce0</td>
</tr>
<tr>
<td>0000030d</td>
<td>1c 00 00 TNT TTN (3)</td>
</tr>
<tr>
<td>00000310</td>
<td>2d f0 3c TIP 0x3cf0</td>
</tr>
<tr>
<td>00000313</td>
<td>06 TNT T (1)</td>
</tr>
<tr>
<td>00000314</td>
<td>59 2e MTC 0x2e</td>
</tr>
<tr>
<td>00000316</td>
<td>94 TTTNTTN (6)</td>
</tr>
<tr>
<td>00000317</td>
<td>a8 TNT NTNTTN (6)</td>
</tr>
<tr>
<td>00000318</td>
<td>a6 TNT NTNNTT (6)</td>
</tr>
</tbody>
</table>
VM Analysis

- Resource consumption and process analysis
  - **PTParse**¹: Extract PT data from Perf
  - **VMPT**²: Format PT data as XML *bundles*
- *Bundle* contains PIP (CR3 value / NR bit), VMCS and TSC packets in XML
  - VMCS Base Register → Associated VM
  - CR3 → Process
  - NR → VM Entry/Exit
- Decoder + TraceCompass View

¹ https://github.com/tuxology/dorsal/tree/master/ptparse
² https://github.com/tuxology/vmpt
Intel PT

Perf data → PTParse → VMPT →

.bundle>
  .PIP> 792ec000 </PIP>
  .NR> 1 </NR>
  .VMCS> 7eb71000 </VMCS>
  .TSC> 2342353646 </TSC>
</bundle>
There's a glitch in the matrix though..
Intel PT

Limitations

Configure and Enable PT

CPU

Intel PT Hardware

Intel PT Packets

Runtime Data

Intel PT Software Decoder

Reconstructed Execution Flow

Binary

Based on, Andi Kleen's Presentation (TracingSummit 2015)
Limitations

Intel PT Software Decoder

CPU
Intel PT Hardware

Binary

Reconstructed Execution Flow
Limitations - JIT Code

Intel PT Software Decoder

- Trace Packets
  - TNT - T
  - TNT - N

- Static Code
  - jnz
  - add
  - nop
  - jz

CPU

Intel PT Hardware

Binary

Reconstructed Execution Flow
Limitations - JIT Code

- Trace Packets:
  - TNT - T
  - TNT - N

- Static Code:
  - jnz
  - add
  - nop
  - jz

- Reconstructed Execution Flow

- Binary

- CPU
  - Intel PT Hardware

- Intel PT Software Decoder
Limitations - JIT Code

Intel PT

CPU

Intel PT Hardware

Intel PT Software Decoder

Trace Packets

TNT - T
TNT - N

Static Code

jnz
add
nop
jz

Runtime Generated Code

Reconstructed Execution Flow

Binary
Limitations - JIT Code

Intel PT Software Decoder

- Trace Packets
  - TNT - T
  - TNT - N

- Static Code
  - jnz
  - add
  - nop
  - jz

- Runtime Generated Code

Reconstructed Execution Flow

CPU
- Intel PT Hardware

Binary
Intel PT

Limitations - JIT Code

CPU

Intel PT Hardware

Intel PT Software Decoder

Trace Packets
TNT - T
TNT - N

Static Code
jnz
add
nop
jz

Runtime Generated Code

Reconstructed Execution Flow

Binary
Limitations - JIT Code

Intel PT Software Decoder

- Trace Packets
  - TNT - T
  - TNT - N

- Static Code
  - jnz
  - add
  - nop
  - jz

- Runtime Generated Code

- Binary
- ??

Reconstructed Execution Flow
Limitations - JIT Code

Intel PT Software Decoder

Trace Packets

TNT - T
TNT - N

Static Code

jnz
add
nop
jz

Runtime Generated Code

Binary

Reconstructed Execution Flow
Limitations - Self-modifying Code

Intel PT Software Decoder

Trace Packets

- TNT - T
- TNT - N

Static Code

- jnz
- add
- nop
- jz

Reconstructed Execution Flow
Limitations - Self-modifying Code

- CPU
  - Intel PT Hardware
- Intel PT Software Decoder
  - Trace Packets
    - TNT - T
    - TNT - N
  - Static Code
    - jnz
    - add
    - jmp
    - jz
- Binary
- Reconstructed Execution Flow
Limitations - Self-modifying Code

**Intel PT Software Decoder**

- **Trace Packets**
  - TNT - T
  - TNT - N

- **Static Code**
  - jnz
  - add
  - jmp
  - jz

**Reconstructed Execution Flow**

**CPU**
- Intel PT Hardware
Limitations - Self-modifying Code

- Intel PT Software Decoder
  - Trace Packets
    - TNT - T
    - TNT - N
  - Static Code
    - jnz
    - add
    - nop
    - jz

- Reconstructed Execution Flow

- Binary
FlowJIT
FlowJIT

Trace Reconstruction

- JIT code (such as eBPF) allocates memory for code-cache
- We define dynamic Code Sections ($CS_r$) - pages corresponding to code-cache executing in process
FlowJIT

Technique

- In-kernel tracking of runtime generated/modified code pages
  - Compilers use `malloc()` and `mprotect()`
  - FlowJIT intercepts and modifies exec bits
- Synthetic page faults at execution
  - Intercept tracked pages and re-flip exec bits
  - Record IP, Timestamp, complete page data as a FlowJIT event and copy to disk.
- Events indexed and queried by IP. At decode time, query by failed decoding IP
FlowJIT

**Technique**

- **Target Process**
  - Runtime Code

- **Page Access Control**
  - Tracked Pages
    - NX
    - NX

- **PF Handler**
  - X
  - X

- **ioctl()**

- **Trace Decoder**
  - Query
  - Userspace
  - Kernel

- **ID**
  - Timestamp
  - Instruction Pointer
  - Runtime Code

- **FlowJIT Events**
FlowJIT

Use case: eBPF JIT Code

FlowJIT

Image at IP

FlowJIT

Query IP

Hardware Trace Packets

? (T_r)

I_r

Process Code

eBPF Code

CS_p

CS_r

N

TNT

T

CFG(CS_r)

Flow(CS_r)
Experiments

**Number of Page Faults with increasing JIT compiled code sites**

- eBPF (Enabled)
- eBPF (Disabled)
- Baseline (Enabled)
- Baseline (Disabled)

**Time Overhead with increasing JIT compiled code sites**

- FlowJIT Enabled
- FlowJIT Disabled
Experiments

\[ T = T(\text{Tracking Initiation}) + T(\text{Access Change}) + T(\text{Page Fault}) \]

20K executions of JIT compiled sites
**Upcoming**

**FlowJIT Patch**

- An initial version of kernel patch against v4.7 available
  - [https://github.com/tuxology/flowjit](https://github.com/tuxology/flowjit)
- Enhance patch and work on Perf
  - Probably link FlowJIT events to PT data in Perf's aux buffer?

**Future Work**

- Extend self-modifying code to provide better code security and application robustness
- Extend FlowJIT to support ARM through Perf
Outcomes

- *Low Overhead Hardware-Assisted Virtual Machine Analysis and Profiling*, IEEE CCSNA'16 Globecom Workshops
- *Hardware Trace Reconstruction of Runtime Compiled Code* [Submitted]
- *Hardware-Assisted Instruction Profiling and Latency Detection*, Journal of Engineering, IET
“Education never ends, Watson. It is a series of lessons, with the greatest for the last.”

~ Arthur Conan Doyle

Questions?

suchakrapani.sharma@polymtl.ca