Mitigating Processor Vulnerabilities by Restructuring the Kernel Address Space

Sebastian Eydam
About Me

- Computer-Science student at BTU Cottbus since 2015
- Cyberus Technology intern since 2017
- Cyberus Technology employee since 2022
About Cyberus

- German cyber security software company
- Founded in 2017
- Focus on secure virtualization and automated software testing

https://www.cyberus-technology.de/blog.html

https://github.com/cyberus-technology/hedron
Motivation

- processor-level vulnerabilities allow attackers in userspace to leak information from kernel address space
Motivation

- processor-level vulnerabilities allow attackers in userspace to leak information from kernel address space
- existing mitigations introduce costly instructions into performance critical parts of the kernel
Motivation

- processor-level vulnerabilities allow attackers in userspace to leak information from kernel address space
- existing mitigations introduce costly instructions into performance critical parts of the kernel
- investigate an alternative mitigation strategy on the kernel design level that ideally adds no runtime overhead and is CPU independent
Talking Points

Current Status
Proposed Mitigation
Case-Study: Hedron
Measurement
Efficacy
Conclusion
Current Status
Underlying Problem

Virtual address space of the attacker

<table>
<thead>
<tr>
<th>0</th>
<th>attacker</th>
<th>-1</th>
</tr>
</thead>
</table>

Virtual address space of the crypto-process

| 0 | crypto-process | -1 |

- attacker
- crypto-process
- OS management data
**Current Status**

Kernel Page-Table Isolation (KPTI)

---

**Virtual address space** of the attacker running in user-mode:

0  
| attacker |

---

**Virtual address space** of the attacker running in kernel-mode:

0  
| attacker |

---

-1  
| crypto-keys |

-1  
| OS management data |

---

**crypto-process**
Current Status
Existing Mitigations and Their Runtime-Overheads

- Kernel Page-Table Isolation: 5% - 30%
Current Status
Existing Mitigations and Their Runtime-Overheads

- Kernel Page-Table Isolation: 5% - 30%
- Speculative Load Hardening: 10% - 50%
Current Status
Existing Mitigations and Their Runtime-Overheads

- Kernel Page-Table Isolation: 5% - 30%
- Speculative Load Hardening: 10% - 50%
- Retpoline: up to 20%
Current Status
Existing Mitigations and Their Runtime-Overheads

- Kernel Page-Table Isolation: 5% - 30%
- Speculative Load Hardening: 10% - 50%
- Retpoline: up to 20%
- Disabling Speculative Execution: > 100%
Current Status
Existing Mitigations and Their Runtime-Overheads

- Kernel Page-Table Isolation: 5% - 30%
- Speculative Load Hardening: 10% - 50%
- Retpoline: up to 20%
- Disabling Speculative Execution: > 100%
- Indirect Branch Control: 20% - 50%
Proposed Mitigation

General Idea

Virtual address space of the attacker

0 - 1

Attacker

Management

Virtual address space of the crypto-process

0 - 1

Crypto-process

Management
Proposed Mitigation
Process-Local Data Selection

- Does it contain secrets?
  - yes
  - no

- Does it have to be shared?
  - yes
  - no

- Is it accessed during context switch?
  - yes
  - no

- Can the access be split?
  - yes
  - no

- Make it process-local.
- Do not make it process-local.
UTCB

Does it contain secrets?

Does it have to be shared?

Is it accessed during context switch?

Can the access be split?

- Does it contain secrets?
  - yes
  - Does it have to be shared?
    - yes
    - Is it accessed during context switch?
      - yes
      - Can the access be split?
        - yes
        - Make it process-local.
        - no
        - Do not make it process-local.
    - no
      - Is it accessed during context switch?
        - yes
        - Can the access be split?
          - yes
          - Make it process-local.
          - no
          - Do not make it process-local.
    - no
      - Can the access be split?
        - yes
        - Make it process-local.
        - no
        - Do not make it process-local.
  - no
    - Is it accessed during context switch?
      - yes
      - Can the access be split?
        - yes
        - Make it process-local.
        - no
        - Do not make it process-local.
    - no
      - Can the access be split?
        - yes
        - Make it process-local.
        - no
        - Do not make it process-local.
Case-Study: Hedron
Process-Local Data Selection

UTCB

Does it contain secrets?  **Yes**

Does it have to be shared?  

Is it accessed during context switch?  

Can the access be split?  

Does it contain secrets?  

Does it have to be shared?  

Is it accessed during context switch?  

Can the access be split?  

Make it process-local.  

Do not make it process-local.
Case-Study: Hedron
Process-Local Data Selection

**UTCB**

Does it contain secrets? Yes

Does it have to be shared? Yes

Is it accessed during context switch? Yes

Can the access be split?

- No
- Yes

- Make it process-local.
- Do not make it process-local.
Case-Study: Hedron
Process-Local Data Selection

FPU-state

Does it contain secrets?

Does it have to be shared?

Is it accessed during context switch?

Can the access be split?

Does it contain secrets?

Does it have to be shared?

Is it accessed during context switch?

Can the access be split?

Make it process-local.

Do not make it process-local.
FPU-state

Does it contain secrets? Yes

Does it have to be shared? No

Is it accessed during context switch? Yes

Can the access be split? Yes

Make it process-local.

Does it contain secrets? No

Does it have to be shared? No

Is it accessed during context switch? No

Can the access be split? No

Do not make it process-local.

Is it accessed during context switch? Yes

Can the access be split? No

Make it process-local.

Is it accessed during context switch? No

Can the access be split? Yes

Make it process-local.
FPU-state

Does it contain secrets?  Yes

Does it have to be shared?  No

Is it accessed during context switch?  Yes

Can the access be split?  Yes

- Does it contain secrets?
  - Yes
  - Does it have to be shared?
    - Yes
    - Is it accessed during context switch?
      - Yes
      - Can the access be split?
        - Yes
        - Make it process-local.
      - No
      - Make it process-local.
    - No
    - Make it process-local.
  - No
  - Make it process-local.
Case-Study: Hedron
Process-Local Data Selection

FPU-state

Does it contain secrets?  Yes

Does it have to be shared?  No

Is it accessed during context switch?  Yes

Can the access be split?

Does it contain secrets?
  no
  yes

Does it have to be shared?
  yes
  no

Is it accessed during context switch?
  yes
  no

Can the access be split?
  yes
  no

Make it process-local.
Do not make it process-local.
Case-Study: Hedron
Process-Local Data Selection

FPU-state

Does it contain secrets? Yes

Does it have to be shared? No

Is it accessed during context switch? Yes

Can the access be split? Yes
The prototype needed ...

- a memory allocator for process-local memory
The prototype needed ...

- a memory allocator for process-local memory
- slight modifications of the context switch
The prototype needed...

- a memory allocator for process-local memory
- slight modifications of the context switch
- a mechanism to initialize process-local memory
Case-Study: Hedron
Initialization-Problem

\[ VA_{create} \rightarrow T_{create} \rightarrow create\_thread(va\_new) \rightarrow VA_{new} \]
Case-Study: Hedron
Initialization-Problem

T\text{create}

create\_thread(va\_new)

alloc\_pl()

VA_{create}
active

T_{create}

create\_thread(va\_new)

alloc\_pl()

VA_{new}
inactive
Case-Study: Hedron
Initialization-Problem

$\text{create}_\text{VA}$

$\text{create}_\text{active}$

$\text{create}_\text{thread}(\text{va}_\text{new})$

$\text{alloc}_\text{pl}()$

$\text{init}_\text{pl}()$

$\text{T}_\text{create}$

$\text{VA}_{\text{new}}$

$\text{inactive}$
focused on the context switch mechanism
focused on the context switch mechanism

microbenchmark
- focused on the context switch mechanism
- microbenchmark
- Linux kernel compile
• focused on the context switch mechanism
• microbenchmark
• Linux kernel compile
• Windows DiskSpd
Measurement
Microbenchmark

local_benchmark_loop
remote_benchmark_loop

loop [i<REPETITIONS/2]

semaphore_a.down()
context switch
semaphore_b.up()

semaphore_a.up()

semaphore_b.down()
<table>
<thead>
<tr>
<th>Cycles per context switch</th>
<th>Hedron (unmodified)</th>
<th>Hedron with mitigation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2294</td>
<td>2315</td>
</tr>
</tbody>
</table>
Measurement
Linux kernel compile - Results

unmodified
\( \sigma = 4.8s \)

mitigated
\( \sigma = 5.3s \)
Measurement
Windows DiskSpd - Results (128K seq read, threads: 1, queue depth: 32)

- Unmodified: $\sigma = 103$ MiB/s
- Mitigated: $\sigma = 74$ MiB/s

MiB/s
Measurement
Windows DiskSpd - Results (4k rnd read, threads: 16, queue depth: 32)

unmodified
\( \sigma = 4.2 \text{ MiB/s} \)

mitigated
\( \sigma = 4.0 \text{ MiB/s} \)
<table>
<thead>
<tr>
<th>Mitigation</th>
<th>Meltdown</th>
<th>Spectre v1</th>
<th>Spectre v2</th>
<th>unknown</th>
<th>zero cost</th>
<th>CPU independent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kernel Page Table Isolation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Disabling speculative execution</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Speculative Load Hardening</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Retpolines</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Indirect Branch Control</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Proposed Mitigation</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Efficacy of the Mitigation

<table>
<thead>
<tr>
<th>Mitigation</th>
<th>Meltdown</th>
<th>Spectre v1</th>
<th>Spectre v2</th>
<th>unknown</th>
<th>zero cost</th>
<th>CPU independent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kernel Page Table Isolation</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Disabling speculative execution</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Speculative Load Hardening</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Retpolines</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Indirect Branch Control</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Proposed Mitigation</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Efficacy of the Mitigation

<table>
<thead>
<tr>
<th>Mitigation</th>
<th>Meltdown</th>
<th>Spectre v1</th>
<th>Spectre v2</th>
<th>unknown</th>
<th>zero cost</th>
<th>CPU independent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kernel Page Table Isolation</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Disabling speculative execution</td>
<td></td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Speculative Load Hardening</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Retpolines</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Indirect Branch Control</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Proposed Mitigation</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Efficacy of the Mitigation

<table>
<thead>
<tr>
<th>Mitigation</th>
<th>Meltdown</th>
<th>Spectre v1</th>
<th>Spectre v2</th>
<th>unknown</th>
<th>zero cost</th>
<th>CPU independent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kernel Page Table Isolation</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Disabling speculative execution</td>
<td></td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Speculative Load Hardening</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Retpolines</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Indirect Branch Control</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Proposed Mitigation</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Efficacy of the Mitigation

<table>
<thead>
<tr>
<th>Mitigation</th>
<th>Meltdown</th>
<th>Spectre v1</th>
<th>Spectre v2</th>
<th>unknown</th>
<th>zero cost</th>
<th>CPU independent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kernel Page Table Isolation</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Disabling speculative execution</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Speculative Load Hardening</td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Retpolines</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Indirect Branch Control</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Proposed Mitigation</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Efficacy of the Mitigation

<table>
<thead>
<tr>
<th>Mitigation</th>
<th>Meltdown</th>
<th>Spectre v1</th>
<th>Spectre v2</th>
<th>unknown</th>
<th>zero cost</th>
<th>CPU independent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kernel Page Table Isolation</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Disabling speculative execution</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Speculative Load Hardening</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Retpolines</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Indirect Branch Control</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Proposed Mitigation</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Efficacy of the Mitigation

<table>
<thead>
<tr>
<th>Mitigation</th>
<th>Meltdown</th>
<th>Spectre v1</th>
<th>Spectre v2</th>
<th>unknown</th>
<th>zero cost</th>
<th>CPU independent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kernel Page Table Isolation</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
</tr>
<tr>
<td>Disabling speculative execution</td>
<td>⚫</td>
<td>⚫</td>
<td>⚫</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Speculative Load Hardening</td>
<td></td>
<td>⚫</td>
<td>⚫</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Retpolines</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Indirect Branch Control</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td><strong>Proposed Mitigation</strong></td>
<td>✔</td>
<td></td>
<td></td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>
Conclusion

- investigate an alternative mitigation for side-channel attacks
Conclusion

- investigate an alternative mitigation for side-channel attacks
  - created a process-local memory region with distinct content for different process
Conclusion

- investigate an alternative mitigation for side-channel attacks
  - created a process-local memory region with distinct content for different process
  - switching the process-local memory is done as part of the context switch
Conclusion

- investigate an alternative mitigation for side-channel attacks
  - created a process-local memory region with distinct content for different processes
  - switching the process-local memory is done as part of the context switch

- this mitigation is zero-cost and CPU independent
Conclusion

• investigate an alternative mitigation for side-channel attacks
  • created a process-local memory region with distinct content for different processes
  • switching the process-local memory is done as part of the context switch

• this mitigation is zero-cost and CPU independent

• implemented a prototype as a proof of concept

https://github.com/amphi/hedron/tree/new-mitigation-prototype
Conclusion

- investigate an alternative mitigation for side-channel attacks
  - created a process-local memory region with distinct content for different processes
  - switching the process-local memory is done as part of the context switch

- this mitigation is zero-cost and CPU independent

- implemented a prototype as a proof of concept

- measurements show no overhead

https://github.com/amphi/hedron/tree/new-mitigation-prototype
Conclusion

- investigate an alternative mitigation for side-channel attacks
  - created a process-local memory region with distinct content for different process
  - switching the process-local memory is done as part of the context switch

- this mitigation is zero-cost and CPU independent

- implemented a prototype as a proof of concept

- measurements show no overhead

- https://github.com/amphi/hedron/tree/new-mitigation-prototype