CYBERUS TECHNOLOGY

Mitigating Processor Vulnerabilities by Restructuring the Kernel Address Space

Sebastian Eydam

About Me

CYBERUS TECHNOLOGY

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

About Cyberus

CYBERUS TECHNOLOGY

- 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

Current Status

Proposed Mitigation

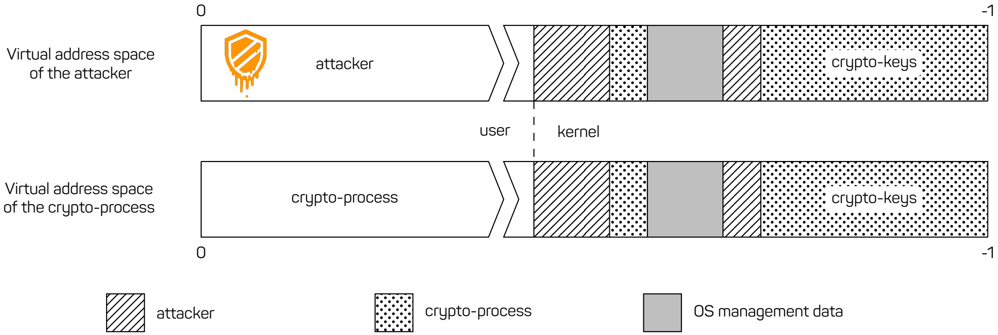
Case-Study: Hedron

Measurement

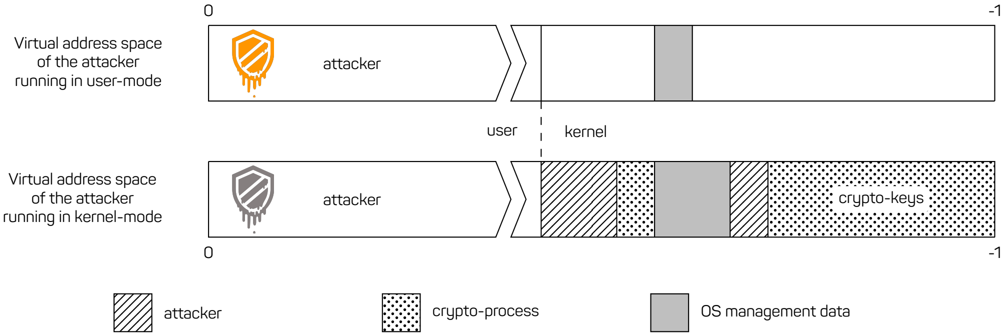
Efficacy

Conclusion

CYBERUS TECHNOLOGY



CYBERUS TECHNOLOGY



- Kernel Page-Table Isolation: 5% - 30%

- Kernel Page-Table Isolation: 5% - 30%
- Speculative Load Hardening: 10% - 50%

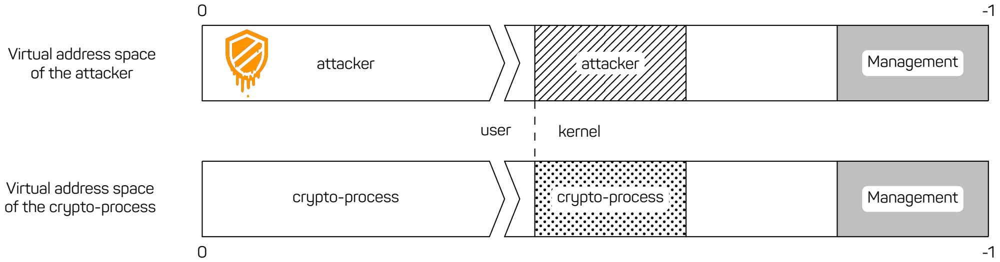
- Kernel Page-Table Isolation: 5% - 30%
- Speculative Load Hardening: 10% - 50%
- Retpoline: up to 20%

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

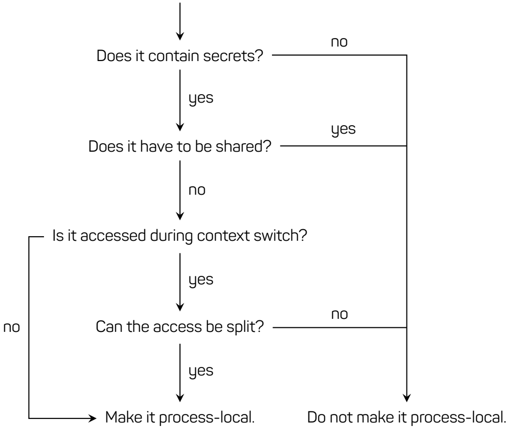
- Kernel Page-Table Isolation: 5% - 30%
- Speculative Load Hardening: 10% - 50%
- Retpoline: up to 20%
- Disabling Speculative Execution: > 100%
- Indirect Branch Control: 20% - 50%

Proposed Mitigation
General Idea

CYBERUS TECHNOLOGY



CYBERUS TECHNOLOGY



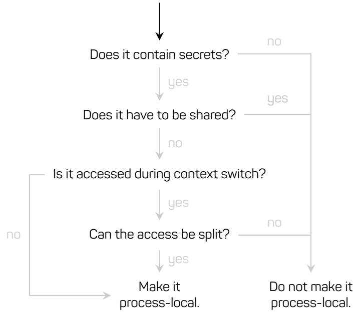
UTCB

Does it contain secrets?

Does it have to be shared?

Is it accessed during context switch?

Can the access be split?



CYBERUS TECHNOLOGY

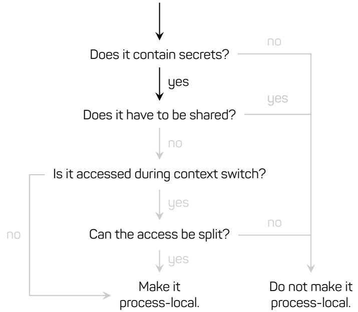
UTCB

Does it contain secrets? Yes

Does it have to be shared?

Is it accessed during context switch?

Can the access be split?



CYBERUS TECHNOLOGY

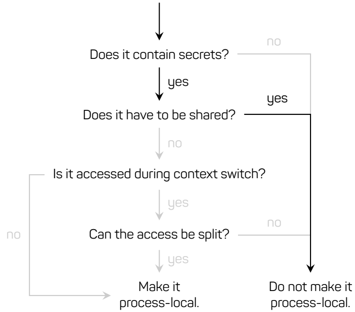
UTCB

Does it contain secrets? Yes

Does it have to be shared? Yes

Is it accessed during context switch?

Can the access be split?



CYBERUS TECHNOLOGY

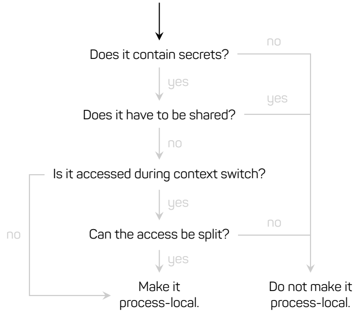
FPU-state

Does it contain secrets?

Does it have to be shared?

Is it accessed during context switch?

Can the access be split?



CYBERUS TECHNOLOGY

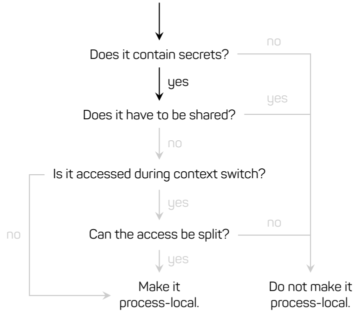
FPU-state

Does it contain secrets? Yes

Does it have to be shared?

Is it accessed during context switch?

Can the access be split?



CYBERUS TECHNOLOGY

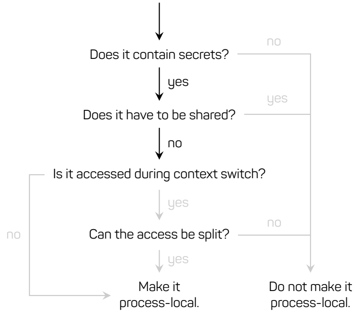
FPU-state

Does it contain secrets? Yes

Does it have to be shared? No

Is it accessed during context switch?

Can the access be split?



CYBERUS TECHNOLOGY

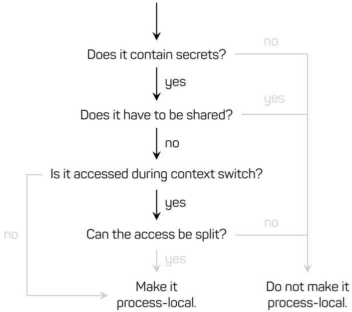
FPU-state

Does it contain secrets? Yes

Does it have to be shared? No

Is it accessed during context switch? Yes

Can the access be split?



CYBERUS TECHNOLOGY

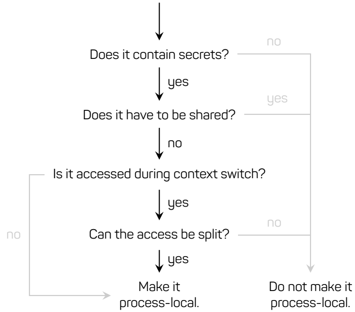
FPU-state

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

CYBERUS TECHNOLOGY

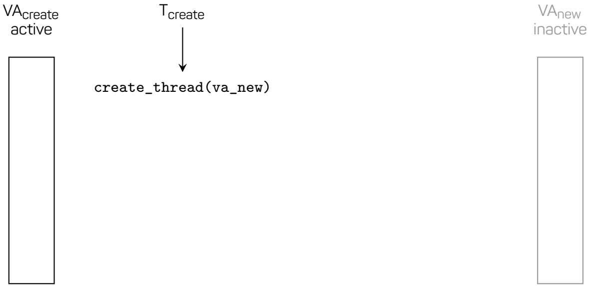
The prototype needed ...

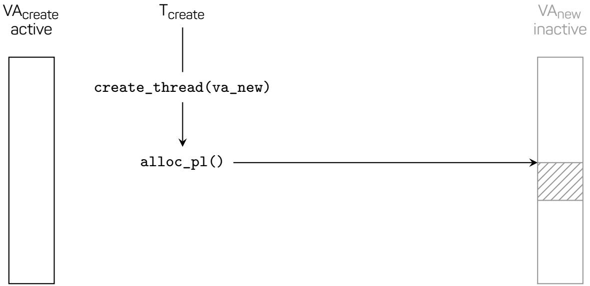
- a memory allocator for process-local memory
- slight modifications of the context switch
- a mechanism to initialize process-local memory

Case-Study: Hedron

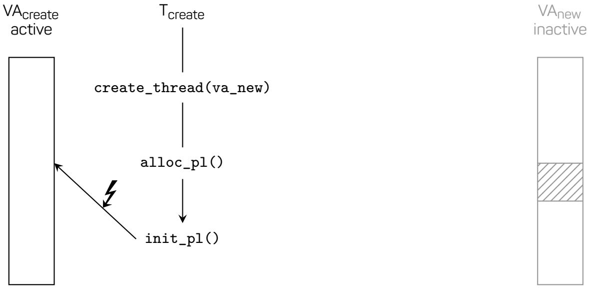
Initialization-Problem

CYBERUS TECHNOLOGY





VAcreate
active
Tcreate
create_thread(va_new)
alloc_pl()
VAnew
inactive



VAcreate
active
Tcreate
create_thread(va_new)
alloc_pl()
init_pl()
VAnew
inactive

CYBERUS TECHNOLOGY

- focused on the context switch mechanism

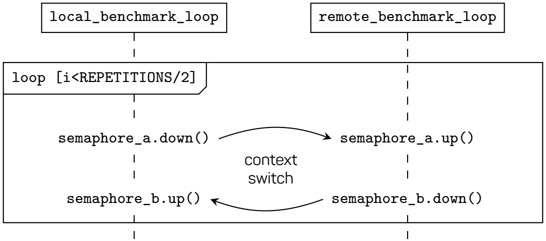
CYBERUS TECHNOLOGY

- 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

CYBERUS TECHNOLOGY

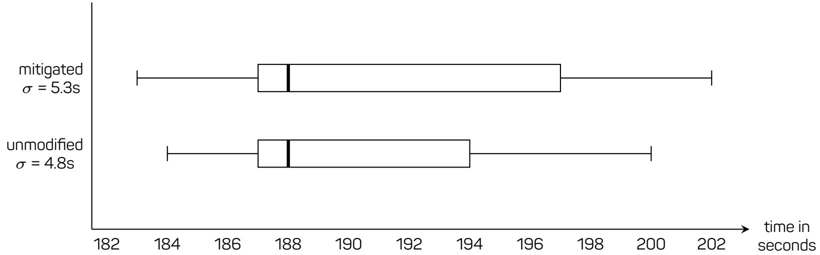


CYBERUS TECHNOLOGY

	Hedron (unmodified)	Hedron with mitigation
Cycles per context switch	2294	2315

Measurement

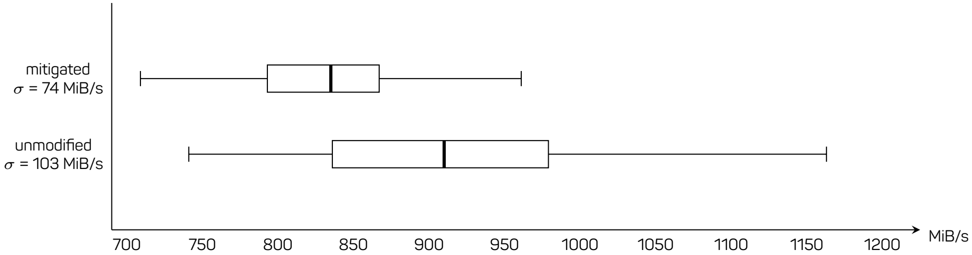
Linux kernel compile - Results



Measurement

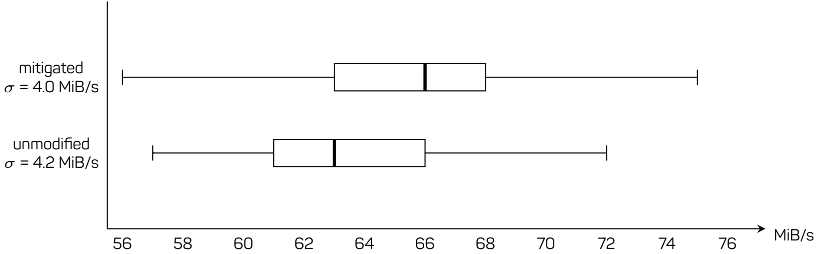
Windows DiskSpd - Results (128K seq read, threads: 1, queue depth: 32)

CYBERUS TECHNOLOGY



Measurement

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



Efficacy of the Mitigation

CYBERUS TECHNOLOGY

Mitigation	Meltdown	Spectre v1	Spectre v2	unknown	zero cost	CPU independent
Kernel Page Table Isolation						
Disabling speculative execution						
Speculative Load Hardening						
Retpolines						
Indirect Branch Control						
Proposed Mitigation						

Efficacy of the Mitigation

CYBERUS TECHNOLOGY

Mitigation	Meltdown	Spectre v1	Spectre v2	unknown	zero cost	CPU independent
Kernel Page Table Isolation	✓					✓
Disabling speculative execution						
Speculative Load Hardening						
Retpolines						
Indirect Branch Control						
Proposed Mitigation						

Efficacy of the Mitigation

CYBERUS TECHNOLOGY

Mitigation	Meltdown	Spectre v1	Spectre v2	unknown	zero cost	CPU independent
Kernel Page Table Isolation	✓					✓
Disabling speculative execution		●	●			
Speculative Load Hardening						
Retpolines						
Indirect Branch Control						
Proposed Mitigation						

Efficacy of the Mitigation

CYBERUS TECHNOLOGY

Mitigation	Meltdown	Spectre v1	Spectre v2	unknown	zero cost	CPU independent
Kernel Page Table Isolation	✓					✓
Disabling speculative execution		●	●			
Speculative Load Hardening		●				
Retpolines						
Indirect Branch Control						
Proposed Mitigation						

Efficacy of the Mitigation

CYBERUS TECHNOLOGY

Mitigation	Meltdown	Spectre v1	Spectre v2	unknown	zero cost	CPU independent
Kernel Page Table Isolation	✓					✓
Disabling speculative execution		●	●			
Speculative Load Hardening		●				
Retpolines			●			
Indirect Branch Control						
Proposed Mitigation						

Efficacy of the Mitigation

CYBERUS TECHNOLOGY

Mitigation	Meltdown	Spectre v1	Spectre v2	unknown	zero cost	CPU independent
Kernel Page Table Isolation	✓					✓
Disabling speculative execution		●	●			
Speculative Load Hardening		●				
Retpolines			●			
Indirect Branch Control			●			
Proposed Mitigation						

Efficacy of the Mitigation

CYBERUS TECHNOLOGY

Mitigation	Meltdown	Spectre v1	Spectre v2	unknown	zero cost	CPU independent
Kernel Page Table Isolation	✓					✓
Disabling speculative execution		●	●			
Speculative Load Hardening		●				
Retpolines			●			
Indirect Branch Control			●			
Proposed Mitigation	✓	○	○	✓	✓	✓

Conclusion

CYBERUS TECHNOLOGY

- investigate an alternative mitigation for side-channel attacks

Conclusion

CYBERUS TECHNOLOGY

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

Conclusion

CYBERUS TECHNOLOGY

- 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

CYBERUS TECHNOLOGY

- 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

Conclusion

CYBERUS TECHNOLOGY

- 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

Conclusion

CYBERUS TECHNOLOGY

- 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

Conclusion

CYBERUS TECHNOLOGY

- 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`