CHERITech'26
CONFERENCE

CHERI

12-13 Nov 2026
Munich, DE

CHERI-Zephyr and Compartmentalisation of Userspace Threads

Jennifer Jackson - University of Birmingham
Jesse Spielman - Durham University

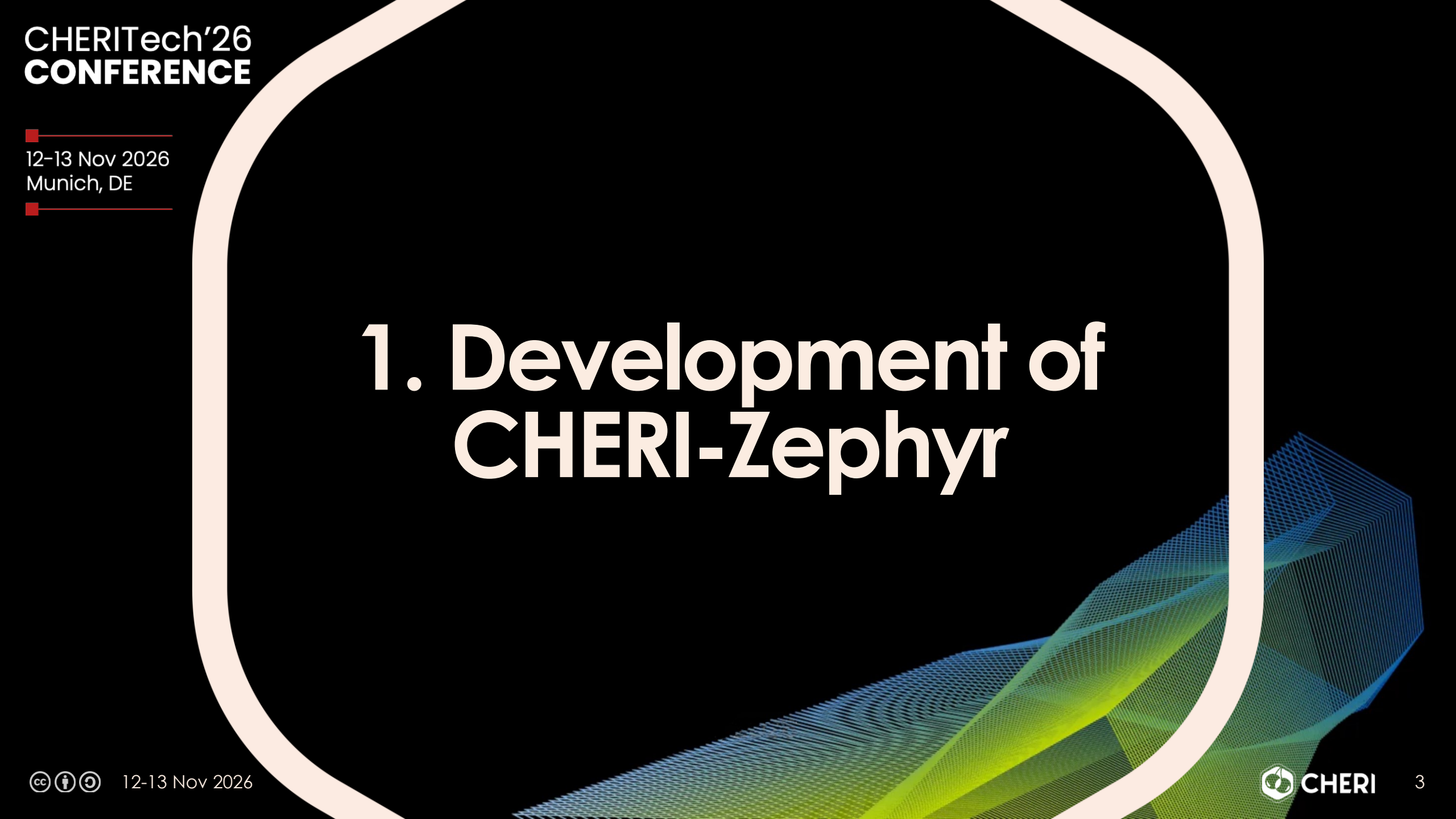
Located at semi | SEMICON EUROPA



Agenda

1. CHERI-Zephyr and it's development
2. Zephyr's userspace model and compartments
3. Proof-of-concept compartment case study

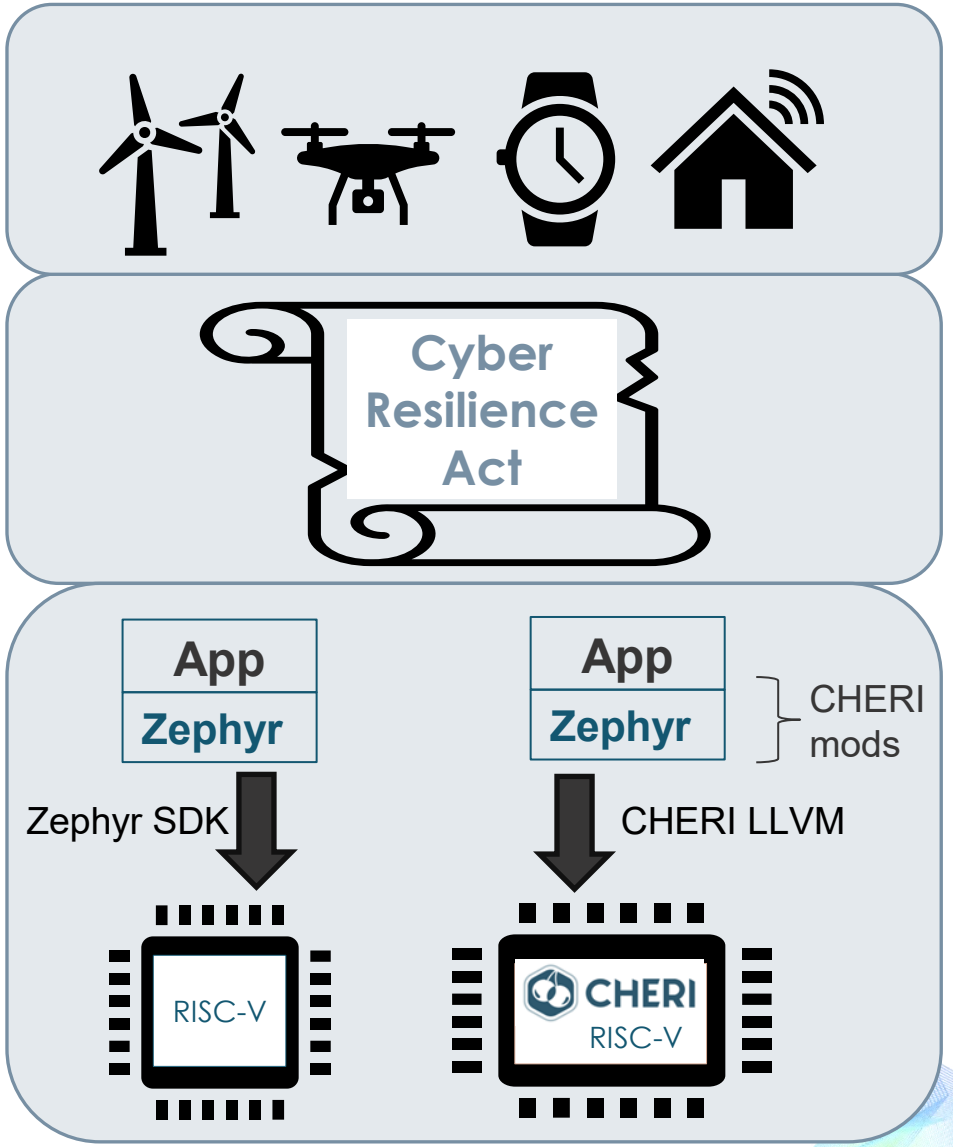
1. Development of CHERI-Zephyr



1.1 Why Support CHERI in Zephyr ?

- **Zephyr RTOS growing** rapidly in IoT and embedded systems.
- **Security** is critical for connected devices. Pressure from **legislation**.

- **CHERI prevents** up to **70%** of software vulnerabilities from being exploited.
- **CHERI** provides **memory safety** to **existing & future**, apps and products.
- **Existing C code** can be **re-compiled** for new CHERI architecture.



1.2 CHERI support

Early Prototype

- CHERI port from zephyr v3.5, **Basic hello_world**
- supported **Cambridge CHERI-RISCV 64-bit ISA**

```
-- west build: running target run
[0/1] To exit from QEMU enter:...+a, x'[QEMU] CPU: riscv64cheri
*** Booting Zephyr OS build zephyr-v3.5.0-1112-g6f49102e4680 ***
Hello World! qemu_riscv64cheri_purecap
```

V0.1.0

- Supported **Codasip CHERI-RISCV 64-bit 9.3 ISA**

https://github.com/CHERI-Alliance/CHERI-zephyr

CHERI-zephyr Public

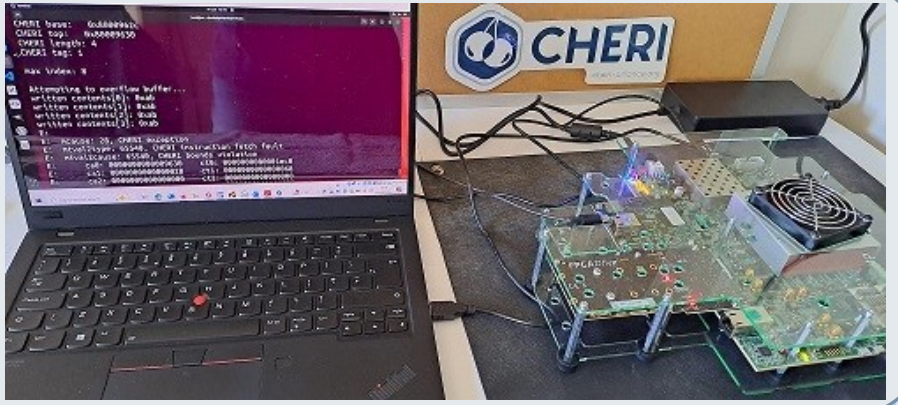
forked from zephyrproject-rtos/zephyr

V0.2.0

- Rebase to Zephyr v4.1

V0.3.0

- Added **support for 32-bit**, **memory allocators**
- CI, CHERI-specific testing, CHERI samples

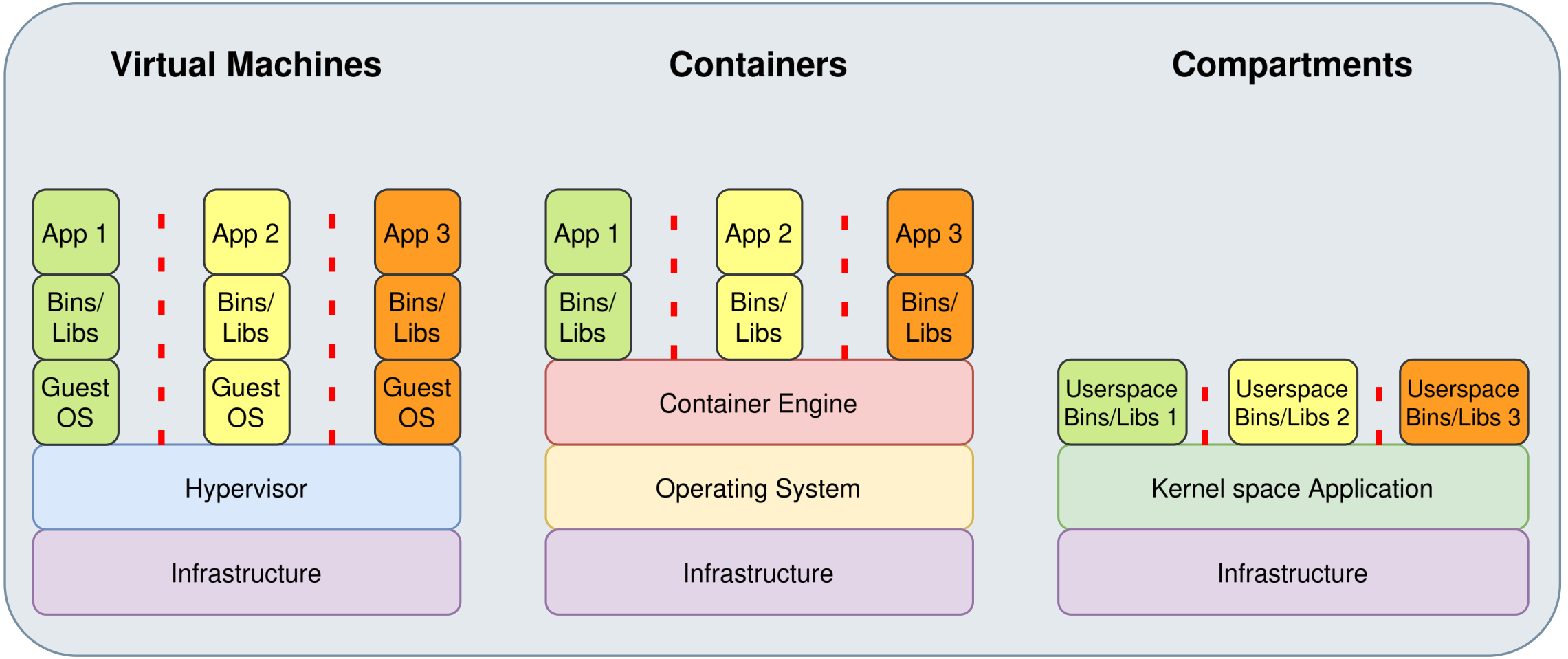


V0.4.0

- **Physical Hardware support**: Codasip Prime Kit
- System heap and **wider kernel functionality**

Experimenting with **Compartmentalisation**

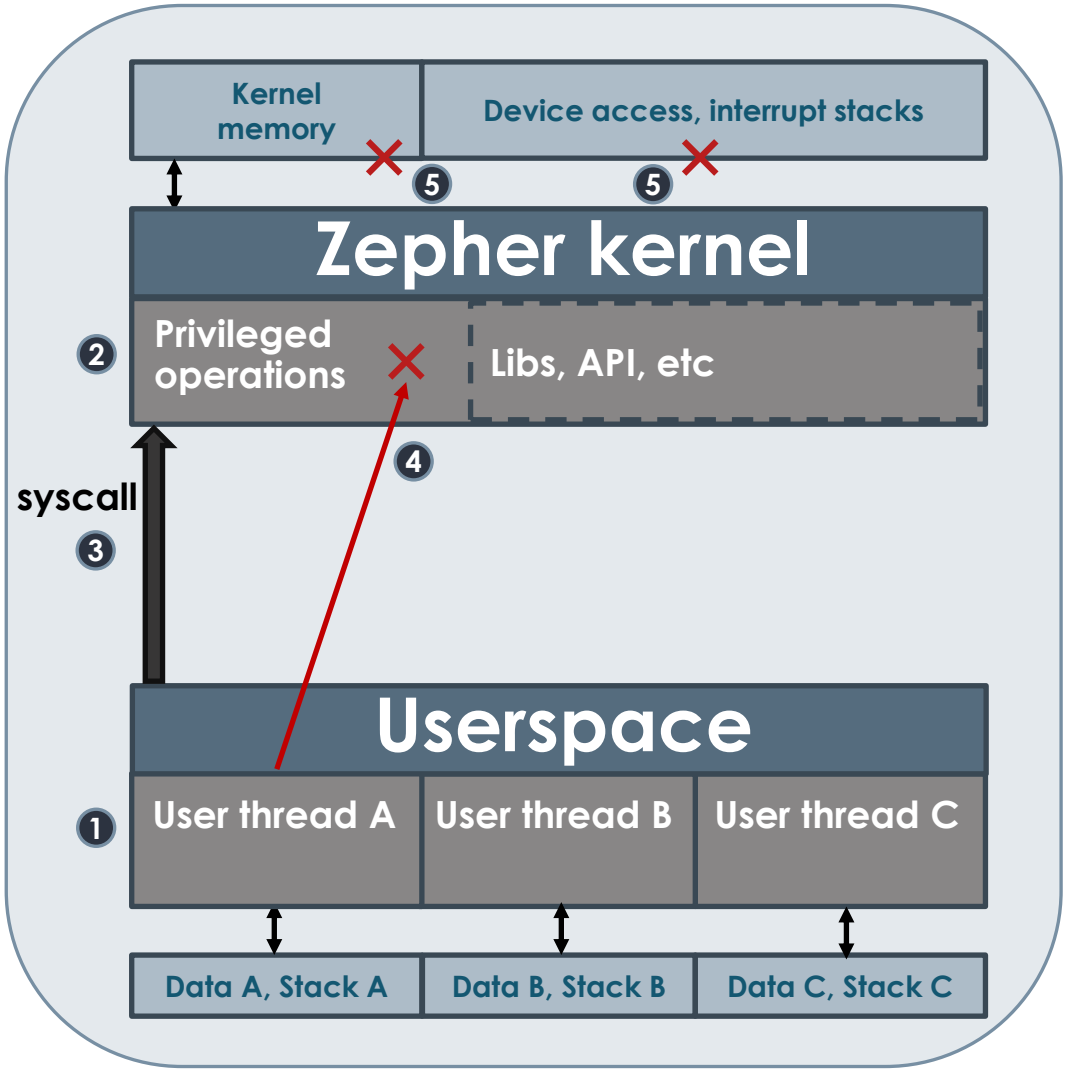
1.3 Compartmentalisation



2. Userspace & Compartments

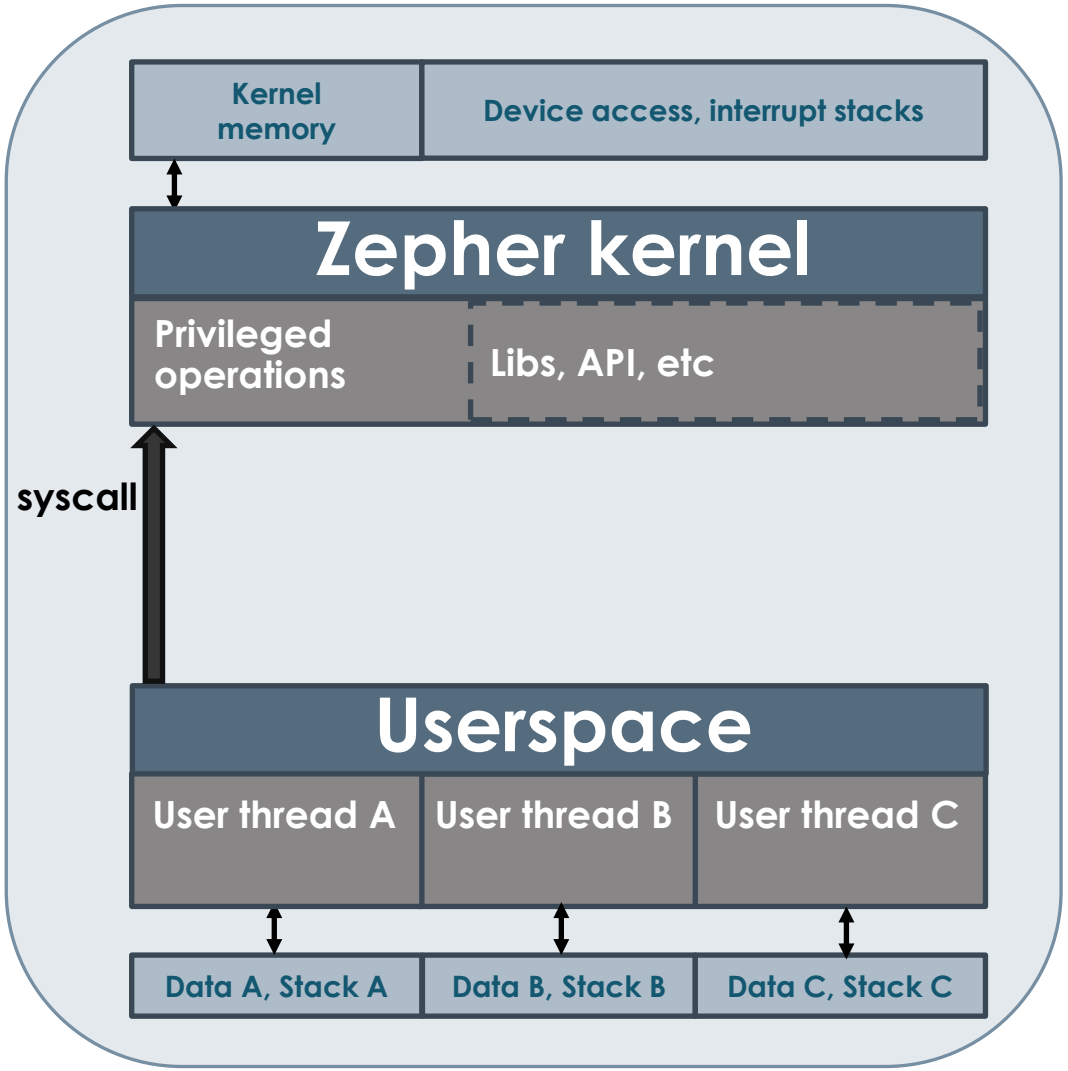


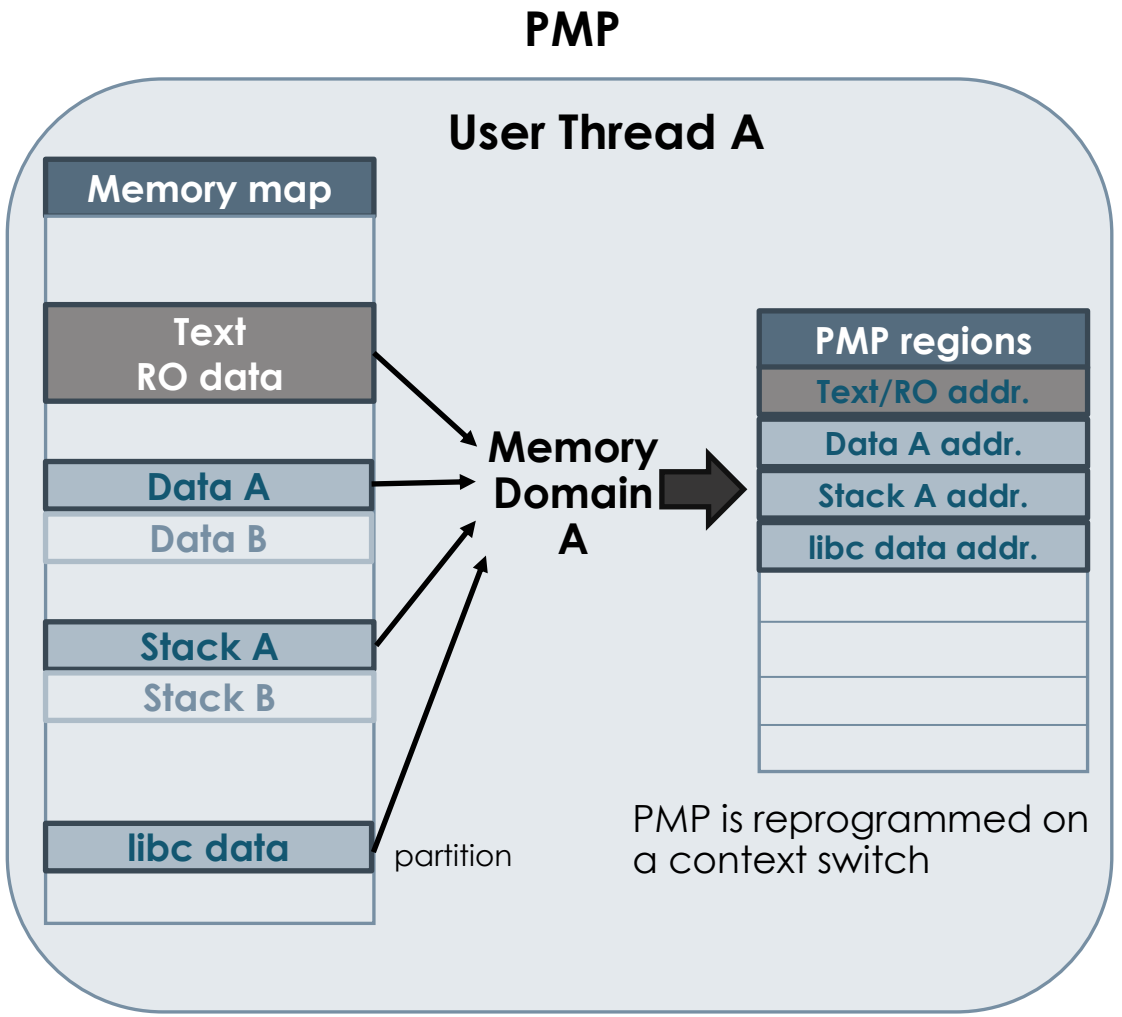
2.1 Zephyr's userspace thread model



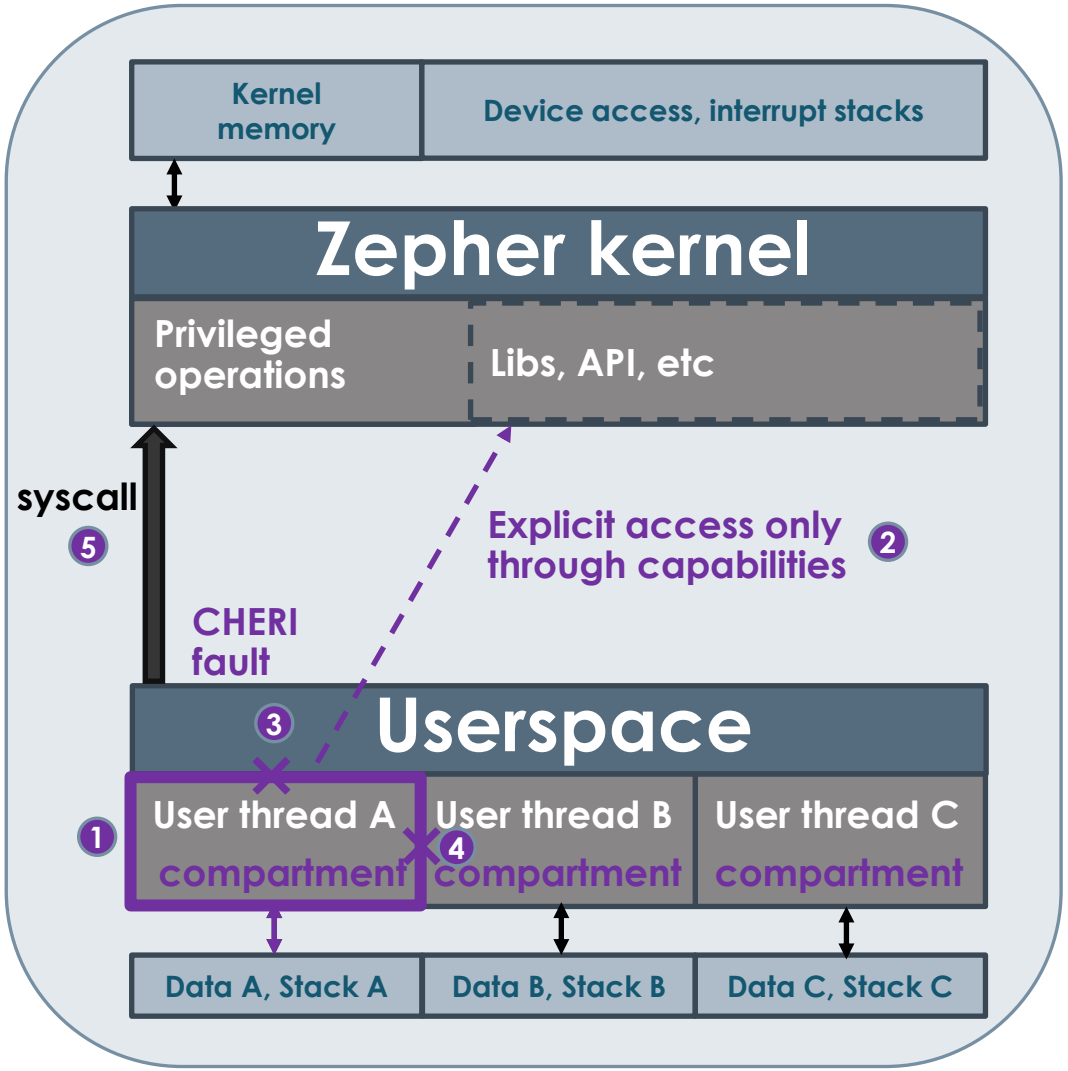
1. **User threads** run with **reduced privileges**
2. **Kernel** controls thread **scheduling** and **context switching**
3. Thread must invoke a **system call** when it needs a kernel service. Cannot manipulate scheduler internals.
4. Thread cannot directly access privileged registers.
5. With **memory access control, (PMP RISCV).** Thread Cannot directly access kernel memory / most hardware peripherals. (PMP fault)

2.2 Zephyr's userspace thread model





2.3 To CHERI Compartments

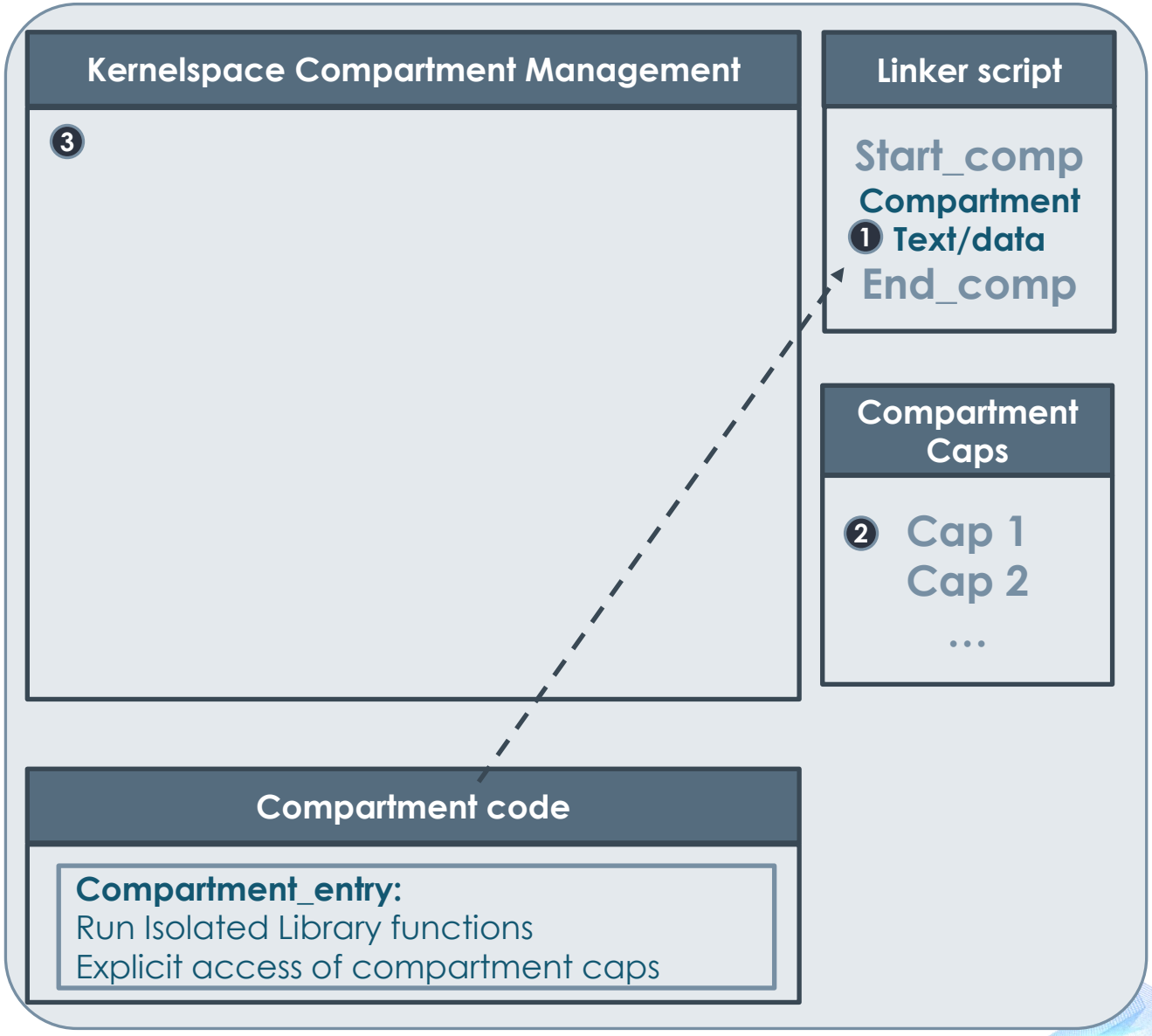


- **With CHERI: - replace PMP**
  - **Same security guarantees as PMP** but is enforced directly through CHERI capabilities.
  - **No fixed limit** on number of protection **regions**.
  - Enables complex compartment layouts **without** the **power scaling penalties** of additional PMP regions.
  - **No PMP programming switching overheads**.

1. Compartment boundaries are the unit of isolation.
2. Thus compartments can be limited to own code, own data and explicitly authorised APIs and entry points.
3. protection enforced at boundary, when attempting to access unauthorised code or objects. (**CHERI fault**)
4. Prevents arbitrary jumps into another compartment's internal code.
5. Syscalls still used for privileged operations.

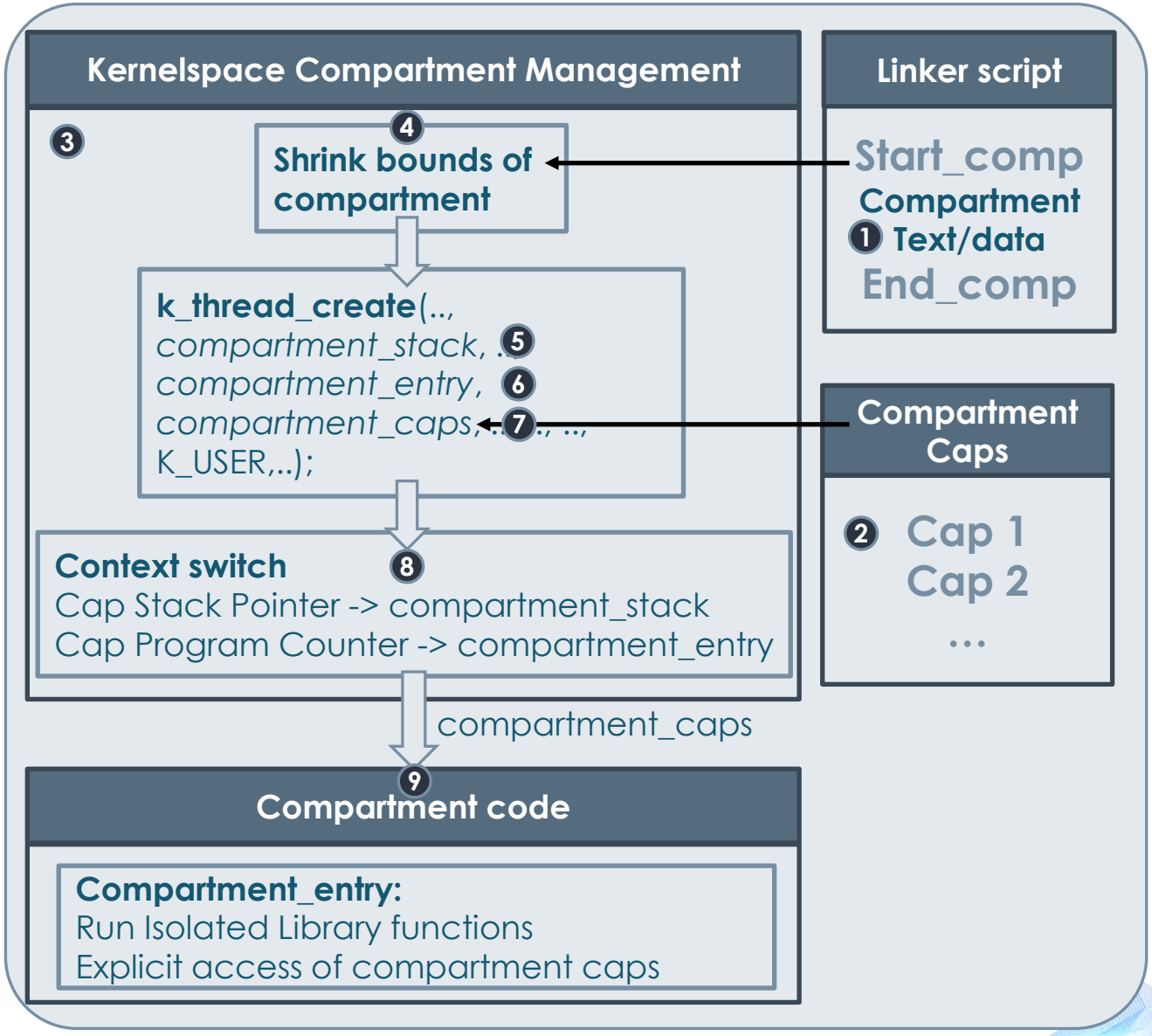
2.4 Proof of Concept

1. **linker script** to place the compartment code into its own section in memory.
2. Structure containing **allowed capabilities**.
3. Set up of the compartment is done in the kernel

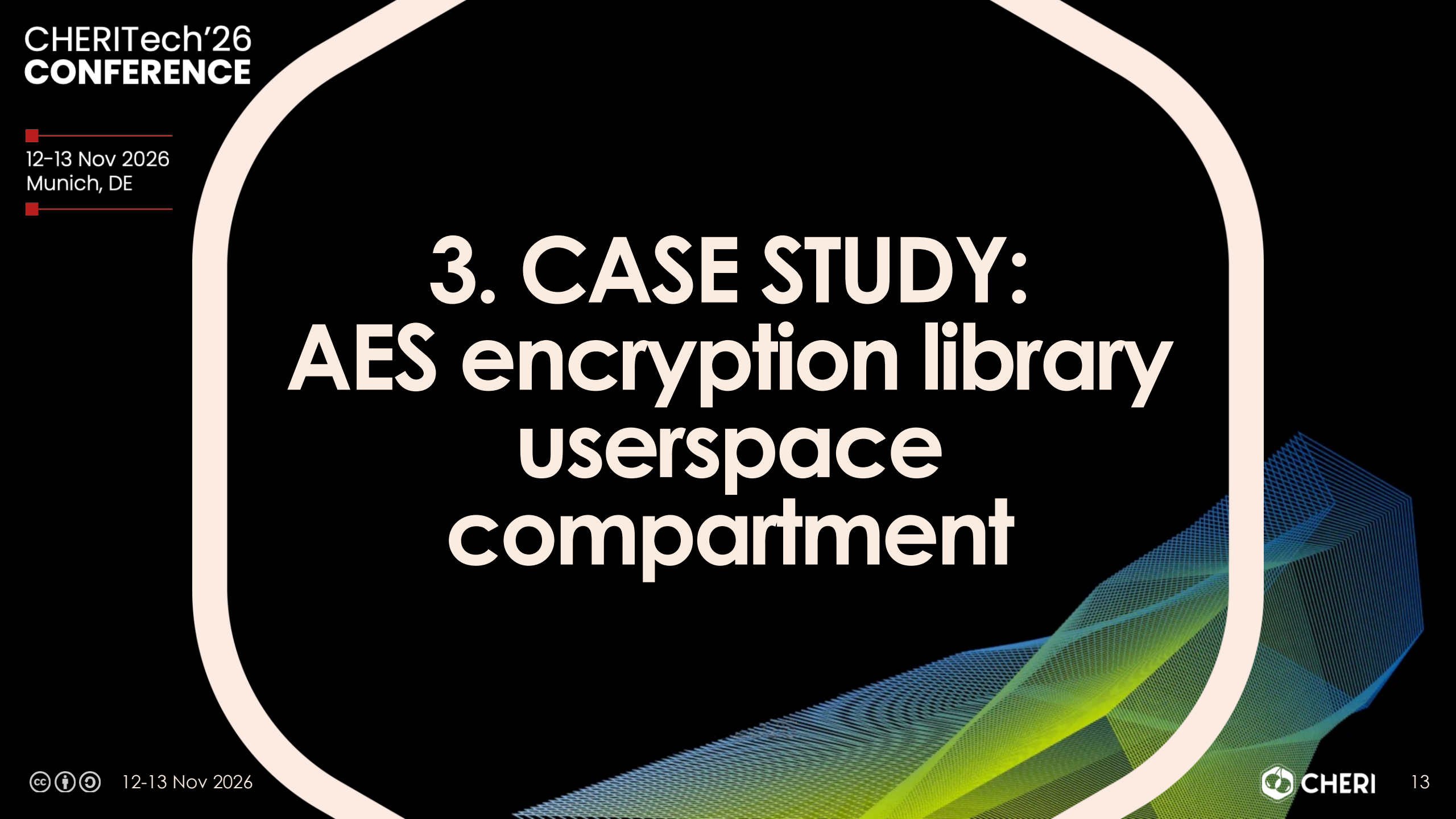


2.5 Proof of Concept

4. **Bounds** of the **compartment** is **shrunk** by the kernel using linker script *start* and *end* points.
5. Compartment **stack** is CHERI **bounded** and placed at compile time.
6. Compartment **entry function** passed to k_thread_create.
7. **Capabilities** granting **explicit access** used by the compartment is defined.
8. On **Context Switch Stack Pointer** and **Program Counter** are CHERI **bounded**.
9. **Compartment code** running isolated library functions.

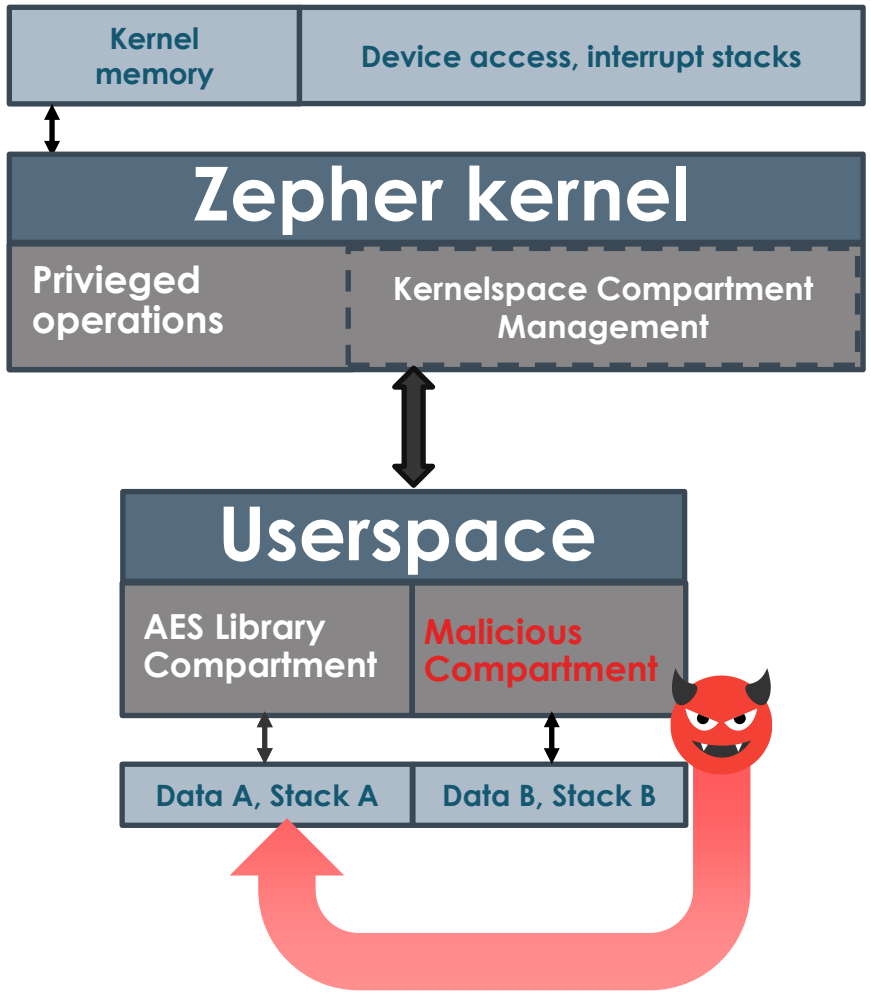


3. CASE STUDY: AES encryption library userspace compartment



3.1 Case study Overview

- Consider a simple **IoT device** with AES-based functionality.
- **Isolate AES functionality** from kernelspace compartment manager (which runs with full privileges) or other compartments.
- A second, **malicious compartment** attempts exfiltration.



Demo #1

Demo 1/2

Exfiltration from full-bounds compartments is possible

```
(.venv) user@34f019110bd0:~/zephyrproject/zephyr/demo$ ./run_demo.sh demo1.elf
```

Demo #2

Demo 2/2

CHERI prevents exfiltration from reduced-bounds compartments

```
(.venv) user@34f019110bd0:~/zephyrproject/zephyr/demo$ ./run_demo.sh demo2.elf
```

3.2 Summary & Next Steps

- Compartmentalisation is possible with CHERI using the Zephyr architecture and abstractions.
- Important to ensure bounds are restricted to gain the benefits of CHERI's memory protection.
- **Next Step**: Improve practicality by automating compartment creation and invocation (e.g. linker script).
- **Next Step**: Sidestep global captable to minimise library alterations.

CHERITech'26 CONFERENCE

12–13 Nov 2026
Munich, DE

Located at semi | SEMICON EUROPA

CHERI

THANK YOU

Contact: contact@cheri-alliance.org

Web: www.cheri-alliance.org

CHERI™ and the CHERI logo are protected trademarks. Use of these marks is subject to the CHERI Alliance Trademark Policy – https://cheri-alliance.org/legal-and-policies/trademark/

This work © 2026 by CHERI Alliance is licensed under CC BY-SA 4.0 (Creative Commons Attribution-ShareAlike 4.0 International) – https://creativecommons.org/licenses/by-sa/4.0/

Static Video Demo Output 1

```
*** Booting Zephyr OS build 4e87a3487411 ***

>> Demo 1/2: Exfiltration from full-bounds compartments is possible

>> Full PCC (code) Bounds: 0x80000000-0x80035b00 (len: 219904) [r-x]

>> AES Compartment:
>> code bounds:  0x80000000-0x80035b00 (len: 219904) [r-x]
>> stack bounds: 0x8001c5b0-0x8001d5b0 (len:   4096) [rw-]

>> Starting AES compartment...
>> Secret key location: 0x8001d4e0
>> Secret Key:          2b 7e 15 16 28 ae d2 a6 ab f7 15 88 09 cf 4f 3c
>> Encryption result:   3ad77bb40d7a3660a89ecaf32466ef97
>> Ground truth:        3ad77bb40d7a3660a89ecaf32466ef97
>> Encryption validated!

>> Attack Compartment:
>> code bounds:   0x80000000-0x80035b00 (len: 219904) [r-x]
>> stack bounds:  0x8001d5b0-0x8001e5b0 (len:   4096) [rw-]

>> Starting attack compartment...
>> Attempting to access secret key...
>> Exfiltrated Key: 2b 7e 15 16 28 ae d2 a6 ab f7 15 88 09 cf 4f 3c !
```

Static Video Demo Output 2

```
>> Demo 2/2: CHERI prevents exfiltration from reduced-bounds compartments

>> Full PCC (code) Bounds: 0x80000000-0x80035e00 (len: 220672) [r-x]

>> AES Compartment:
>> code bounds: *before* reduction: 0x80000000-0x80035e00 (len: 220672) [r-x]
>> code bounds: *after* reduction:  0x80021000-0x80021f00 (len:   3840) [r-x]
>> stack bounds:                    0x8001c690-0x8001d690 (len:   4096) [rw-]

>> Starting AES compartment...
>> Secret key location: 0x8001d5c0
>> Secret Key:          2b 7e 15 16 28 ae d2 a6 ab f7 15 88 09 cf 4f 3c
>> Encryption result:   3ad77bb40d7a3660a89ecaf32466ef97
>> Ground truth:        3ad77bb40d7a3660a89ecaf32466ef97
>> Encryption validated!

>> Attack Compartment:
>> code bounds: *before* reduction: 0x80000000-0x80035e00 (len: 220672) [r-x]
>> code bounds: *after* reduction:  0x80022000-0x8002201c (len:     28) [r-x]
>> stack bounds:                    0x8001d690-0x8001e690 (len:   4096) [rw-]

>> Starting attack compartment...
>> Attempting to access secret key...
E:  mcause: 28, CHERI exception
E:  mtval2type: 65536, CHERI instruction fetch fault
E:  mtval2cause: 65536, CHERI tag violation
E: >>> ZEPHYR FATAL ERROR 0: CPU exception on CPU 0
E: Current thread: 0x8001a620 (unknown)

>> k_sys_fatal_error_handler: reason=0 esf=0x8001dd50
>> CPU CHERI hardware exception!

>> Test Complete
```