CHERITech'26
CONFERENCE

CHERI

12-13 Nov 2026
Munich, DE

Compartmentalizing a CANopen Stack with CHERIoT

Hugo Melder

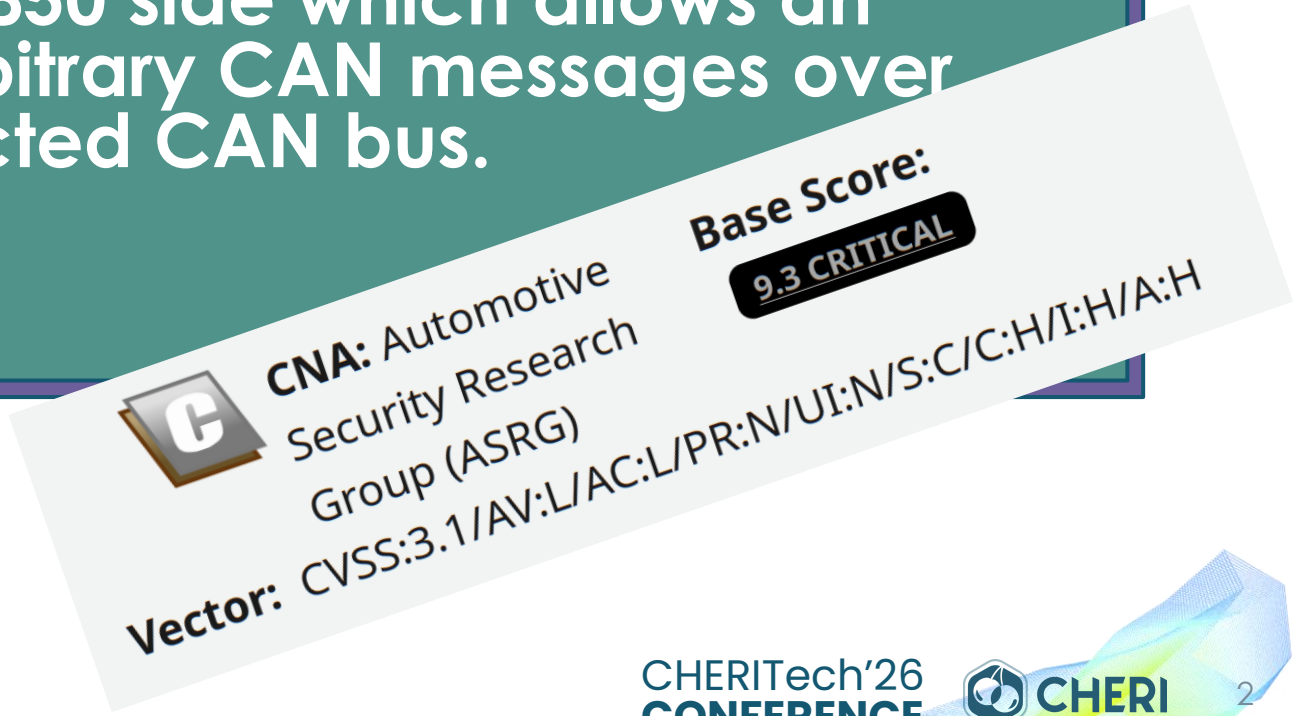
Technische Universität München

Fraunhofer AISEC

SCI

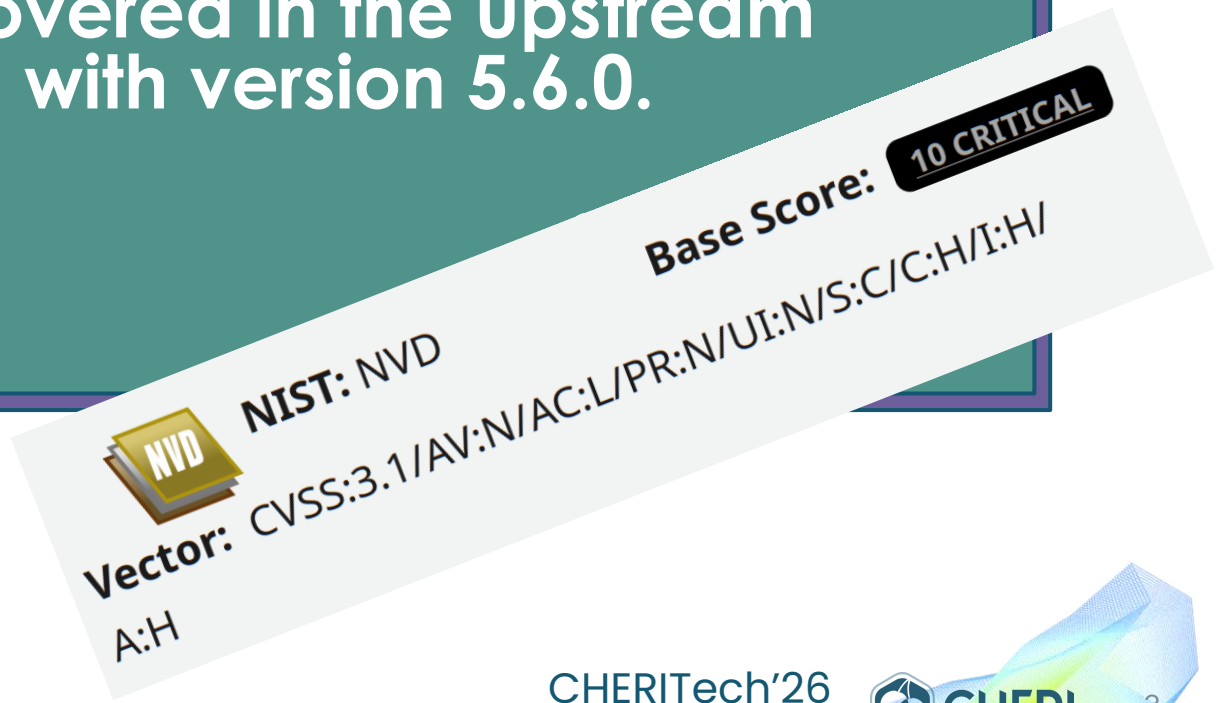
A Protocol Parser Vulnerability

> “There is a vulnerability during processing requests of this protocol on the V850 side which allows an attacker […] to send arbitrary CAN messages over the connected CAN bus.



A Supply Chain Attack

> Malicious code was discovered in the upstream tarballs of xz, starting with version 5.6.0.



How can we mitigate this?

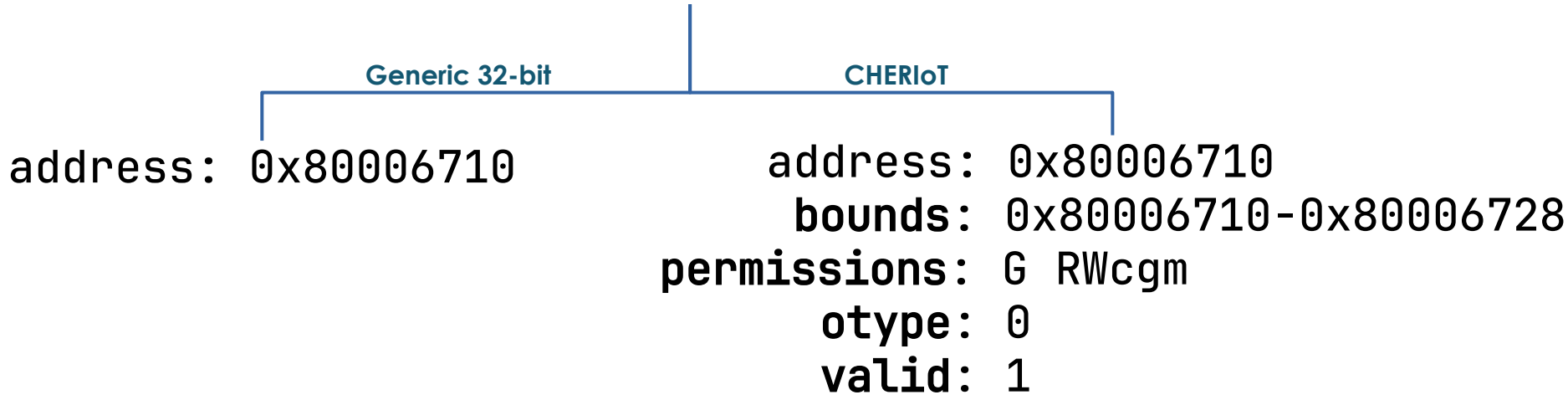
- **Isolation** and **safe sharing** between subsystems
- Memory Safety
- **Controlled degradation** and reset

CHERIoT

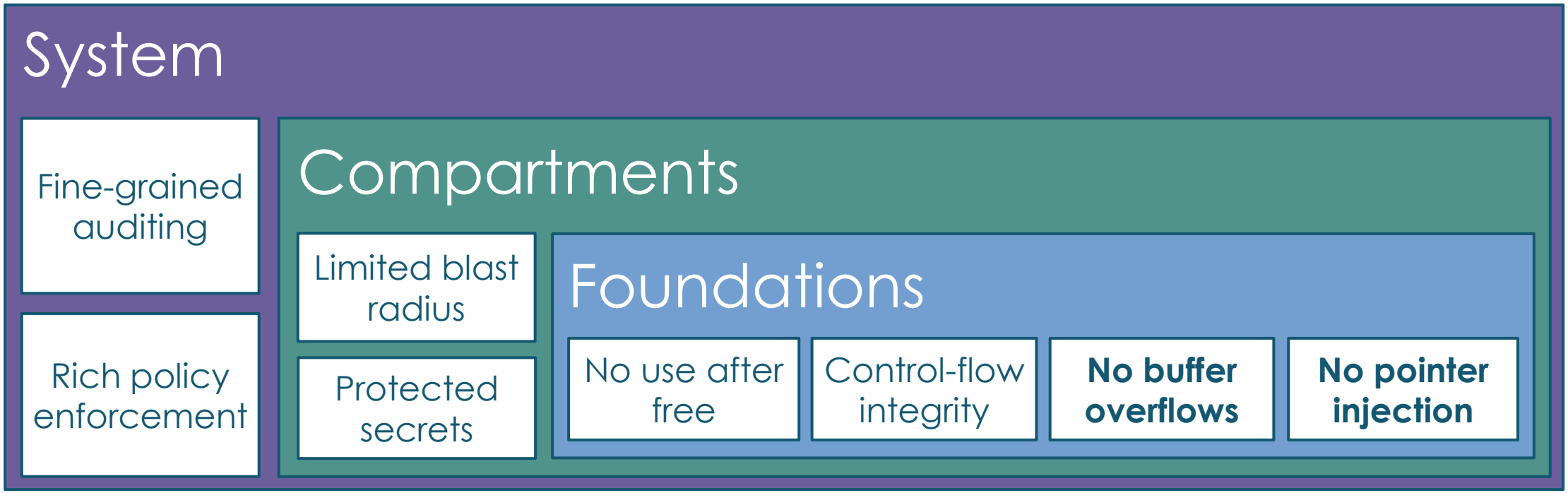


CHERIoT is a capability-based Architecture

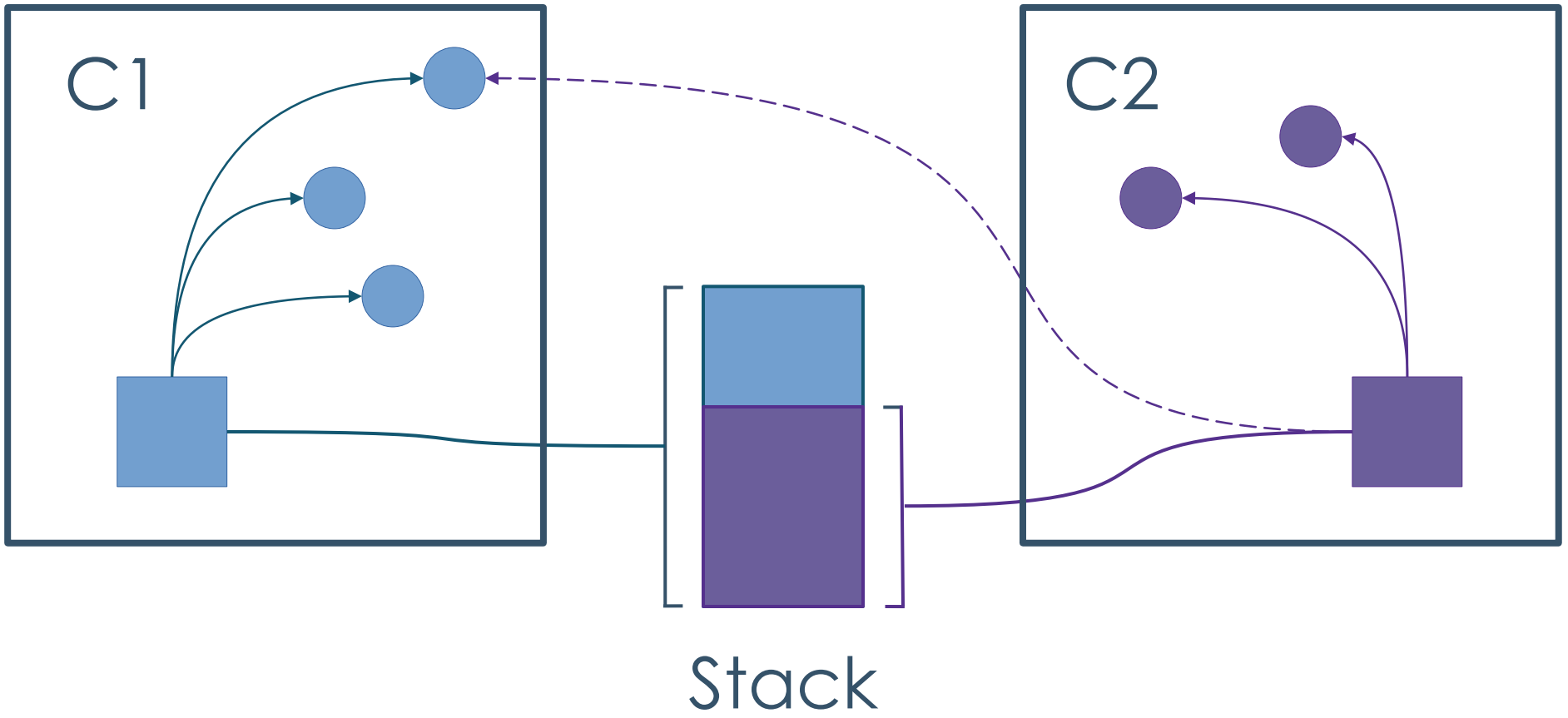
```
char *p = malloc(23);
```



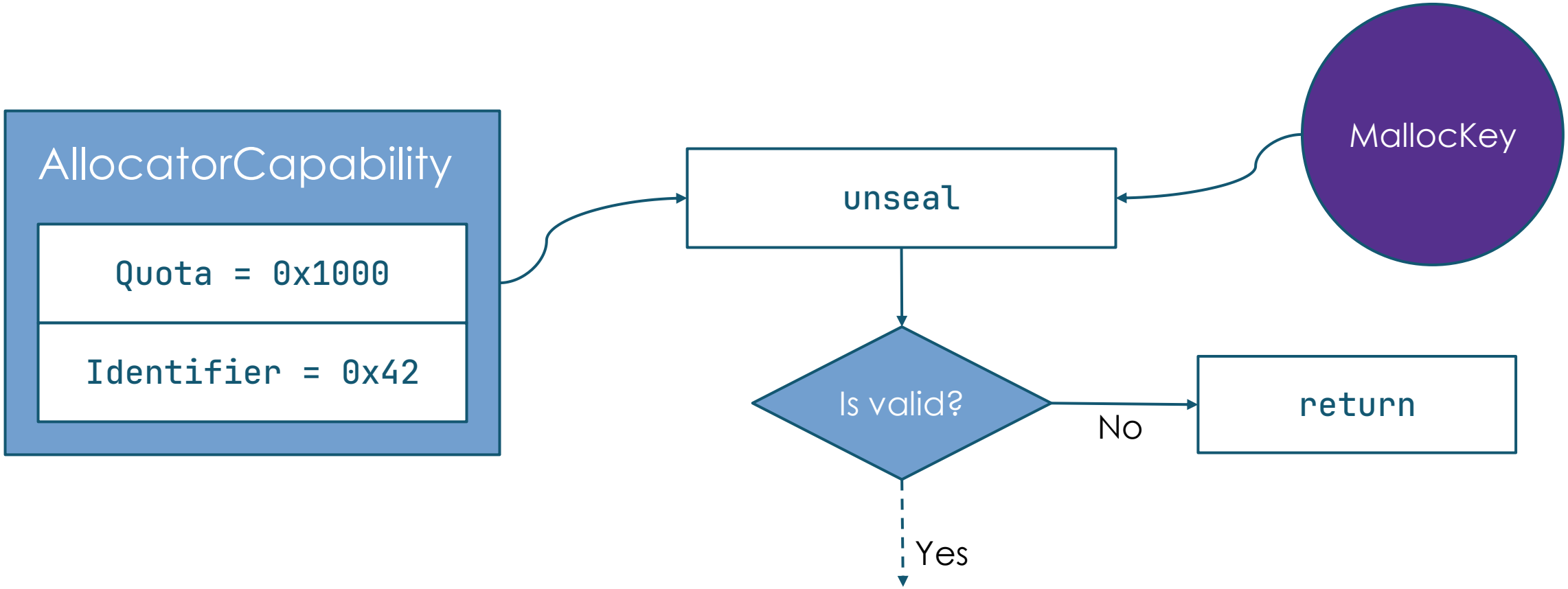
CHERIoT provides a Layered Security Model



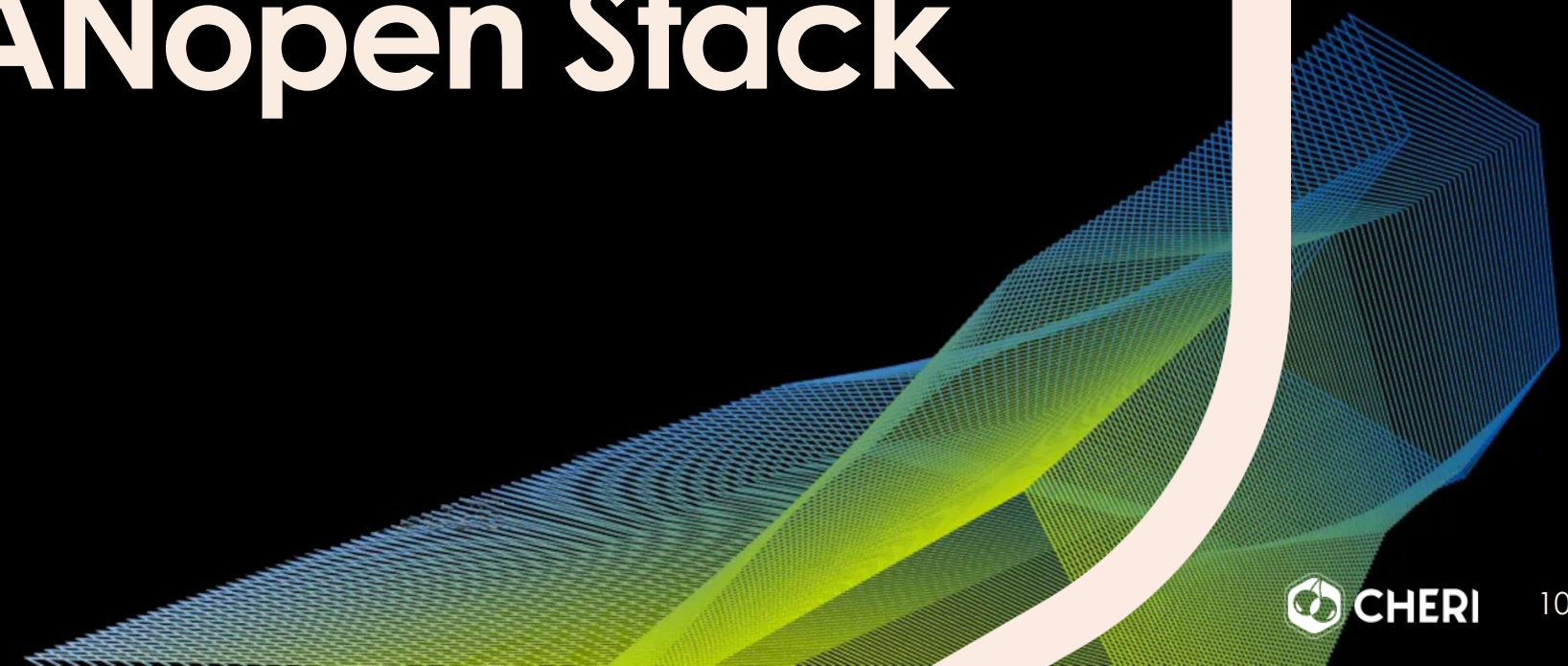
Compartments are Code and Data



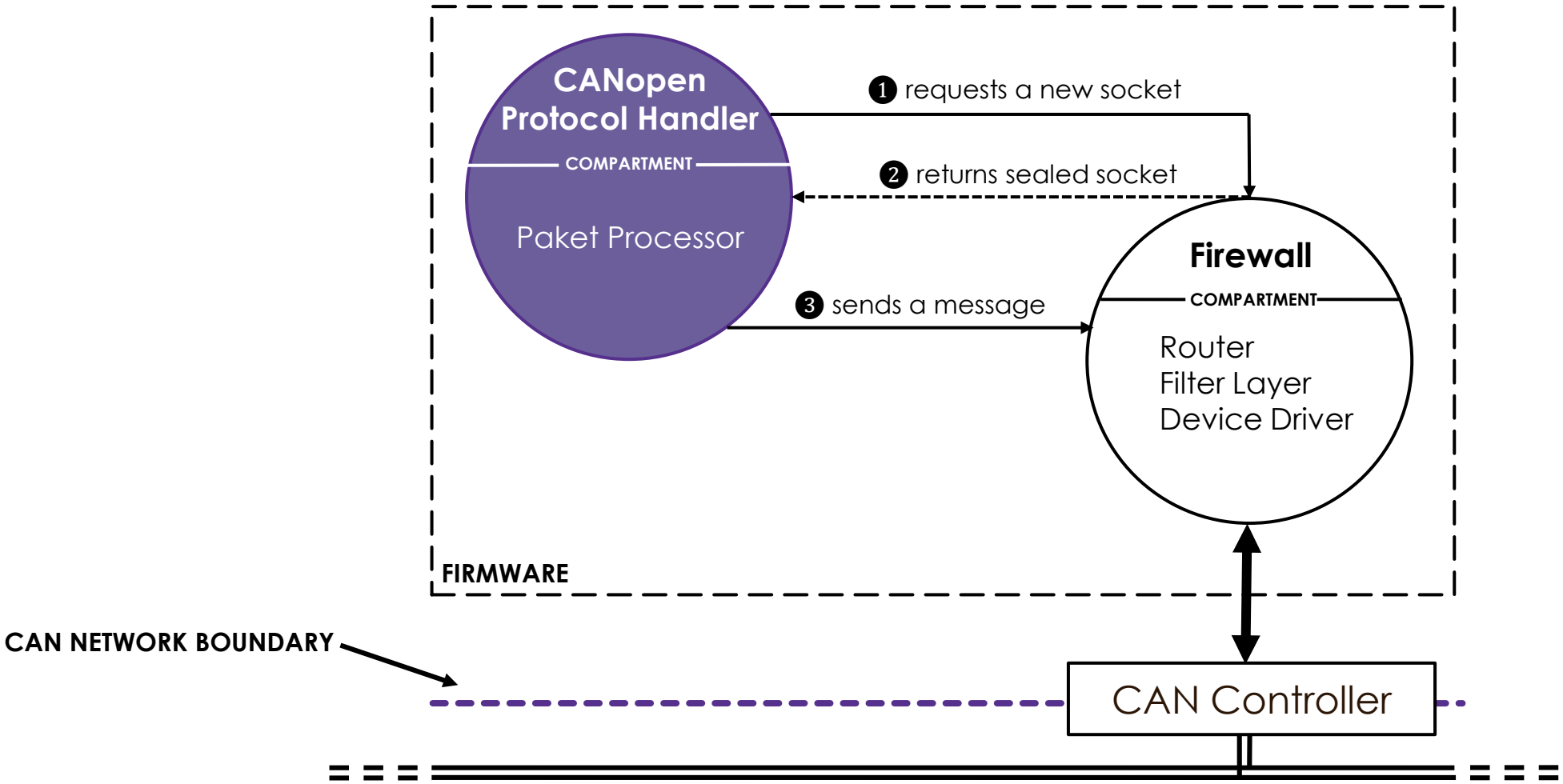
Pushing CHERI up the Stack: Software Capabilities



Compartmentalized CANopen Stack



Separating Concerns: Routing and Processing



Capabilities authorize Communication

```
Socket __cheriot_compartment("firewall")
  socket_create(TimeoutArgument         timeout,
                AllocatorCapability     heapCapability,
                CommunicationCapability canCapability);

int __cheriot_compartment("firewall")
  socket_close(TimeoutArgument     timeout,
               AllocatorCapability heapCapability,
               Socket              socket);

int __cheriot_compartment("firewall")
  socket_send(TimeoutArgument timeout, Socket socket, struct can_frame *frame);

int __cheriot_compartment("firewall")
  socket_receive(TimeoutArgument timeout, Socket socket, struct can_frame *frame);
```

Which Compartments can create Sockets ?

Query: `data.canopen_stack.all_communication_capabilities`


```
[
  {
    "capability": {
    "cob_nodes": [ 1, 42, 2 ],
    "cob_function": 384,
    "communication_type": "transmit",
    "receive_queue_count": 0
    },
  "owner": "TPDO1"
  }
]
```

Evaluation



Guarantees on a Hardware Level

Memory

- Violation of spatial or temporal memory safety triggers fault
- Compromised compartment has limited access

Software-defined Capabilities for Access Control

Authorization

- Well-defined interfaces between compartments
- Static software-defined capabilities

Operations are bounded by propagating Timeouts

Timeout objects...

- track elapsed and remaining time
- are passed down the call chain
- make it easier to reason about **Worst-Case Execution Time**

The Firewall reboots itself

- Plant memory corruption vulnerability into firewall
- Can be exploited by rogue compartment or remotely

```
// Send a frame with invalid length 0xF
struct can_frame frame = {.can_id = 0b1000000001, .can_dlc = 0xF, .data = {}};
int status = socket_send(timeout, socket, &frame);
TEST_EQUAL(status, -EAGAIN, "compartment reboot during socket_send");
```

Latency of the Stack

- Cost of compartmentalization is negligible
- Dominated by queueing operations
- Predictable execution times with low jitter

The right abstractions for security often end up being the same abstractions as for performance and usability.

CC BY SA

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/