



Protecting supply chains with CHERI

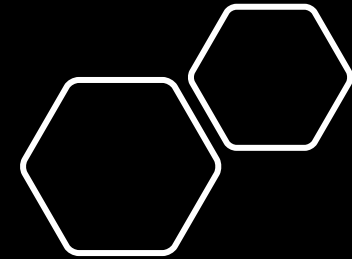
David Chisnall
SCI Semiconductor

NIGHTMARE SUPPLY CHAIN ATTACK SCENARIO —

What we know about the xz Utils backdoor that almost infected the world

Malicious updates made to a ubiquitous tool were a few weeks away from going mainstream.

DAN GOODIN - 4/1/2024, 7:55 AM



What (nearly) went wrong with liblzma?

ld-linux.so

ssh
GOT

libsystemd.so
GOT

liblzma.so
GOT

What (nearly) went wrong with liblzma?

ld-linux.so

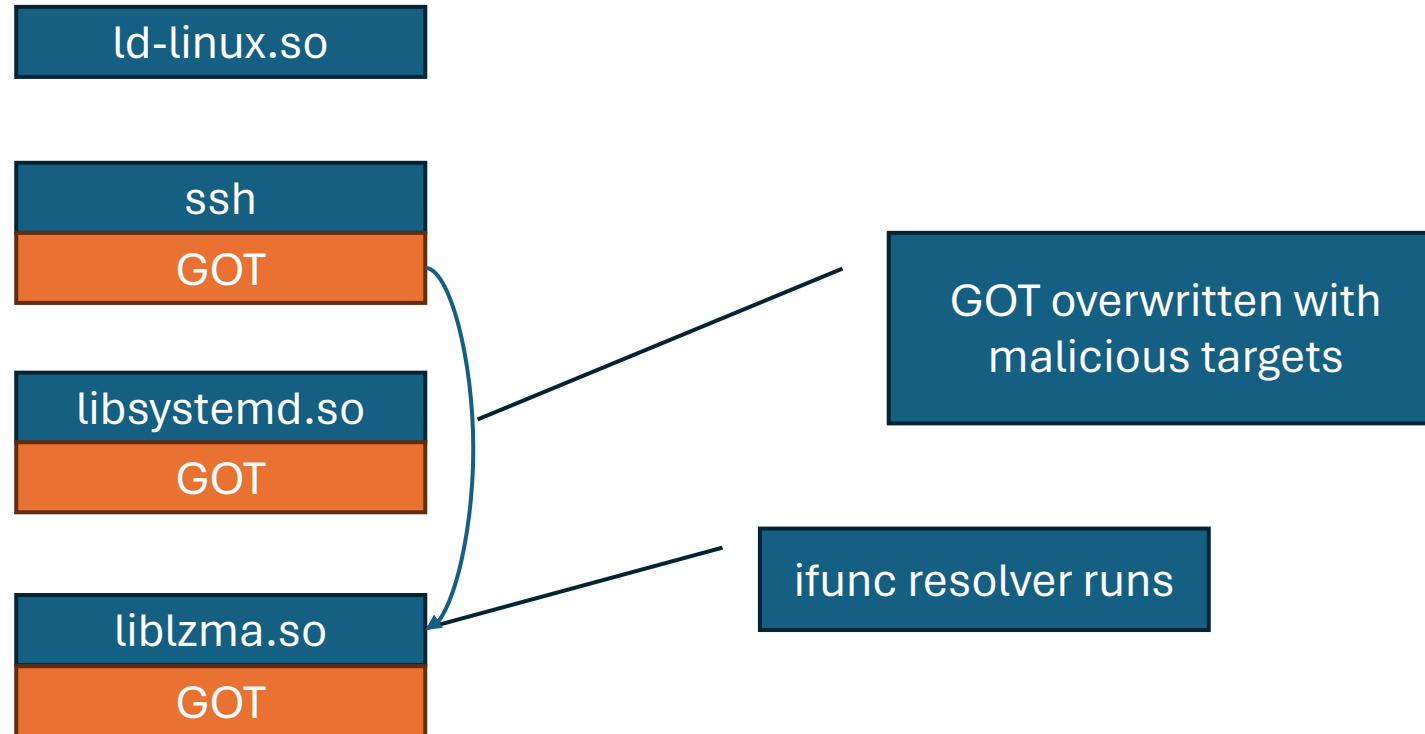
ssh
GOT

libsystemd.so
GOT

liblzma.so
GOT

Read-only GOTs protect cross-library control flow

What (nearly) went wrong with liblzma?



The ifunc is not the problem

Any library can
change GOT
permissions

Any library can
tamper with
any other data

What happens when a supply-chain attacker compromises your program?

Game Over



Supply chain security requires
boundaries around reused code



CHERI Compartmentalization

Mitigating Unknown Vulnerabilities



What is a compartment?

The diagram shows a large white rectangle representing a compartment. Inside this rectangle, there are two smaller, solid-colored rectangles. The left one is blue and contains the word 'Code'. The right one is orange and contains the word 'Data'. The entire compartment is outlined with a white border.

Code

Data

Isolation is easy, sharing is hard

CHERI is designed to enable safe
sharing!



Compartments interact only via capabilities

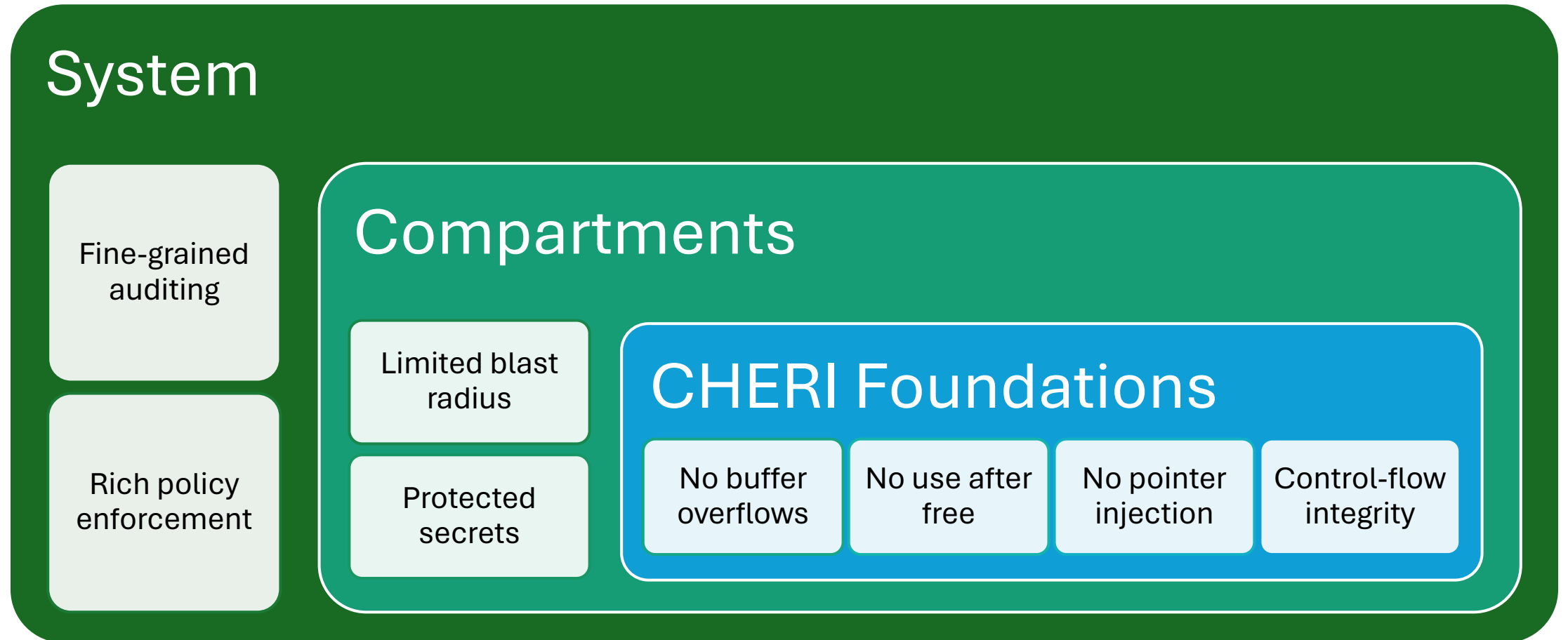


Call other compartments via call gates

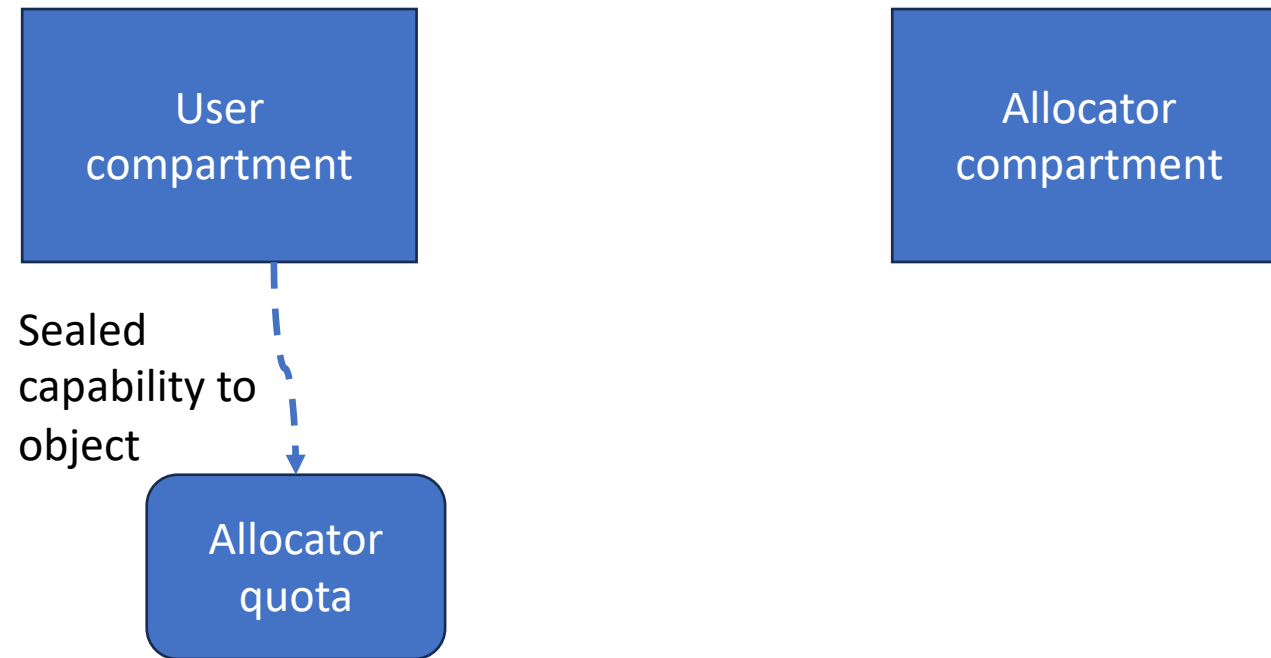


Access shared resources (e.g. MMIO regions) only via memory capabilities

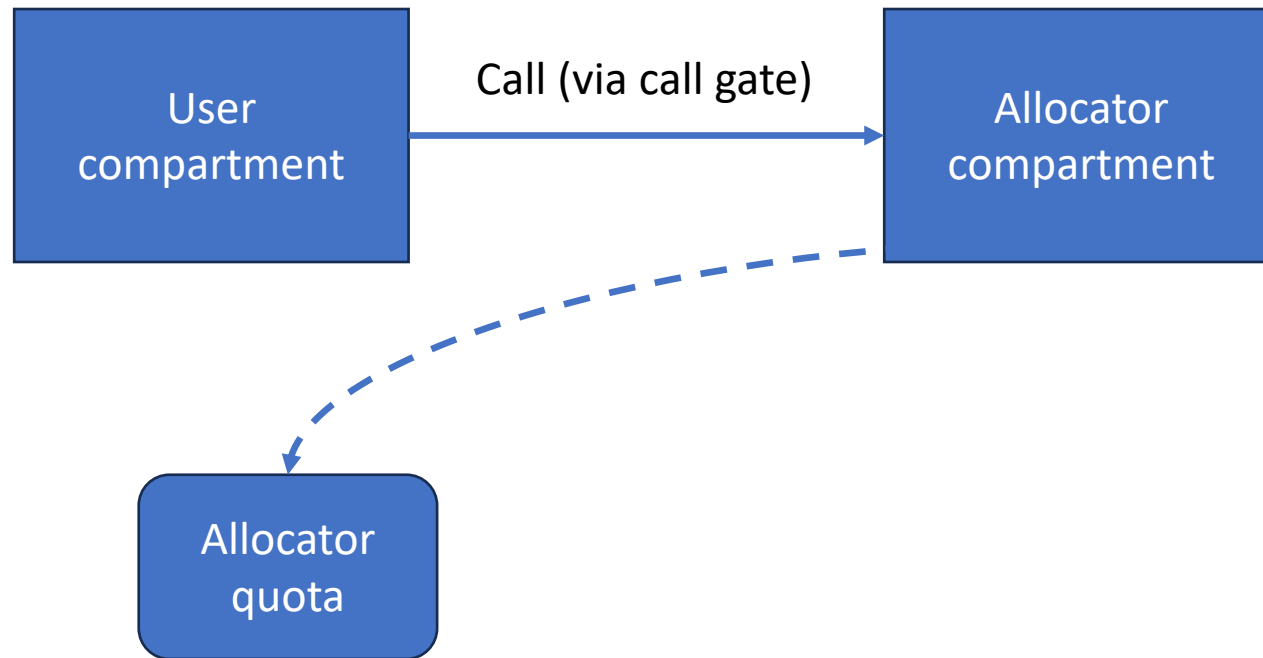
CHERIIoT provides layered security



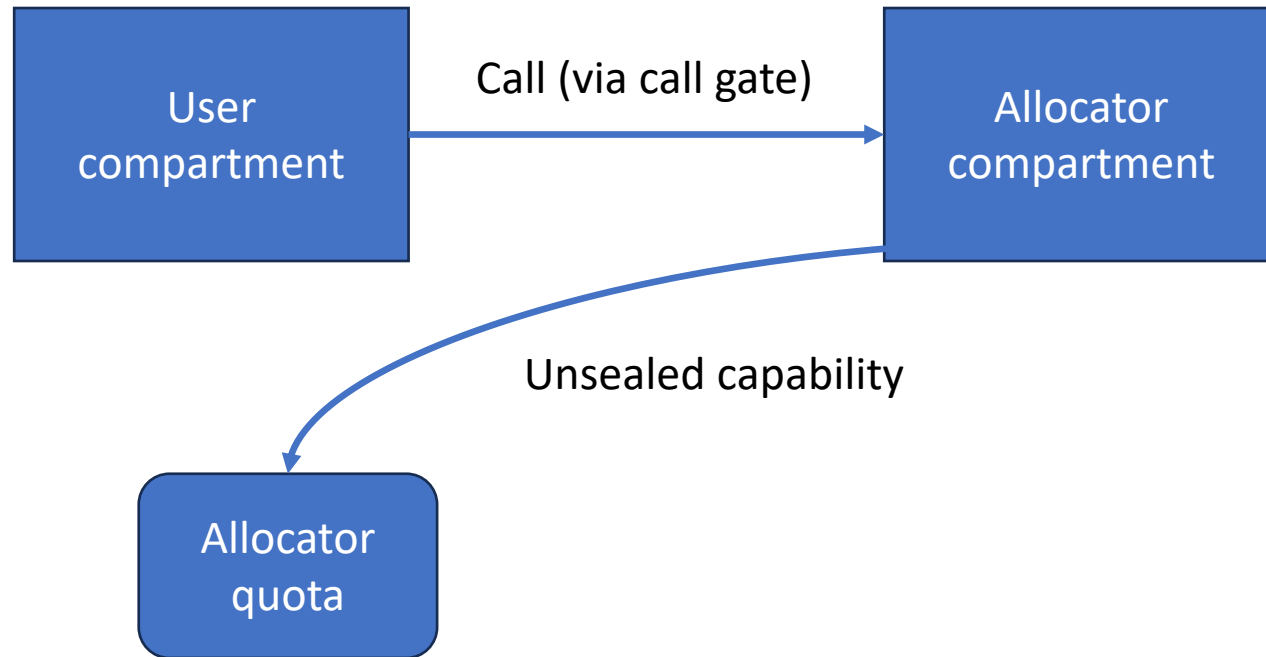
Sealing enables software capabilities



Sealing enables software capabilities



Sealing enables software capabilities



CHERIoT linker reports describe contents and interactions

- Code and data hashes
- Exported functions
- Imported functions
- MMIO regions
- Sealed objects
- Thread stack sizes
- Thread entry points

```
{
  "compartments": {
    "Firewall": {
      "code": {
        "inputs": [
          {
            "file": "build/cheriot/cheriot/release/Firewall.compartment",
            "section_name": ".text",
            "sha256": "b69e004de8cbaee30f71f5f4d929f57ed0e21401f4130ee31e108b40c93b2688",
            "size": 4850
          },
          {
            "file": "build/cheriot/cheriot/release/Firewall.compartment",
            "section_name": ".init_array",
            "sha256": "e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855",
            "size": 0
          }
        ],
        "name": ".Firewall_code",
        "output": {
          "sha256": "eb6f4833e07c93b357411bc5328e8681c5e5ee2ba65e14f9d2894c978e195407"
        }
      },
      "exports": [
        {
          "export_symbol": "__export_Firewall__Z21ethernet_driver_startv",
          "exported": true,
          "interrupt_status": "enabled",
          "kind": "Function",
          "register_arguments": 0,
          "start_offset": 208
        }
      ]
    }
  }
}
```

A man with a beard and short dark hair, wearing a blue t-shirt, is sitting at a desk with a laptop. He has a frustrated or exasperated expression, with his eyebrows furrowed and his mouth slightly open. His hands are raised in the air, palms facing forward, in a gesture of surrender or frustration. The background is a blurred indoor setting, possibly a home office or living room, with shelves and books visible. The overall tone is one of annoyance or frustration.

JSON is not for humans

Rego policy language

- Part of the OpenPolicyAgent project
- Mostly declarative policy language
- Consumes JSON, produces JSON
- Supports composable modules

The screenshot shows the Open Policy Agent documentation website. The page title is "Policy Language". The left sidebar contains a "CORE DOCS" menu with items like "Introduction", "Philosophy", "Policy Language" (highlighted), "What is Rego?", "Why use Rego?", "Learning Rego", "The Basics", "Scalar Values", "Strings", "Composite Values", "Variables", "References", "Comprehensions", "Rules", "Negation", "Universal Quantification (FOR ALL)", "Modules", "Future Keywords", "Some Keyword", "Every Keyword", "With Keyword", "Default Keyword", "Else Keyword", "Operators", "Built-in Functions", "Example Data", "Metadata", and "Schema". The main content area has a search bar, a "Playground" button, and social media icons. The "Policy Language" section includes a sub-section "What is Rego?" which explains that Rego is inspired by Datalog and extends it to support structured document models like JSON. It also defines Rego queries as assertions on data stored in OPA. A blue callout box states: "The examples in this section try to represent the best practices. As such, they make use of keywords that will become standard keywords in OPA v1.0, but have been introduced gradually. See the docs on future keywords for more information." Below this is the "Why use Rego?" section, which notes that Rego is easy to read and write, provides powerful support for nested documents, and is declarative, allowing authors to focus on return values rather than execution details. It also mentions that OPA optimizes queries for performance.

CHERIoT-Audit consumes JSON with Rego



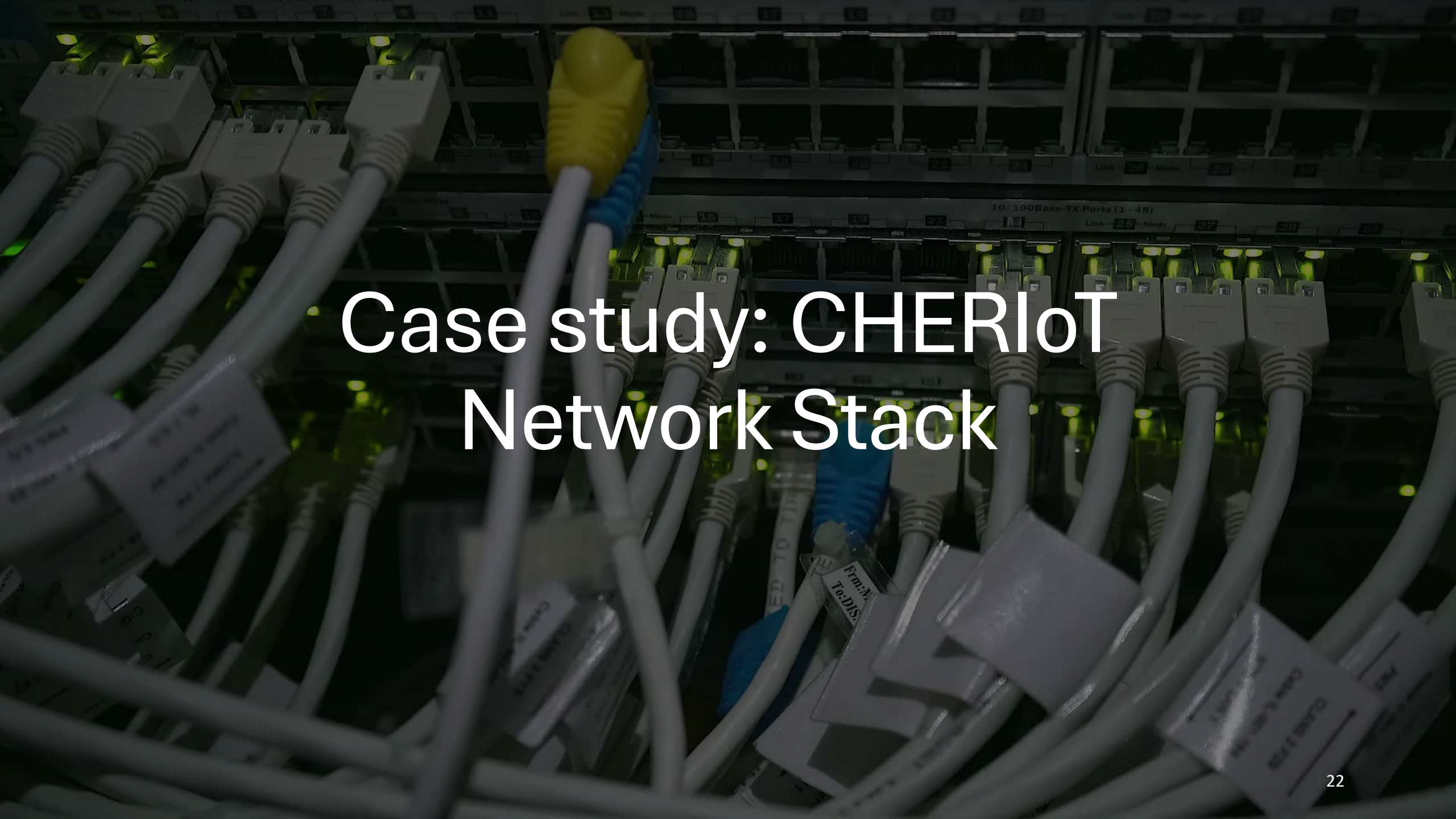
Firmware integrators write policies



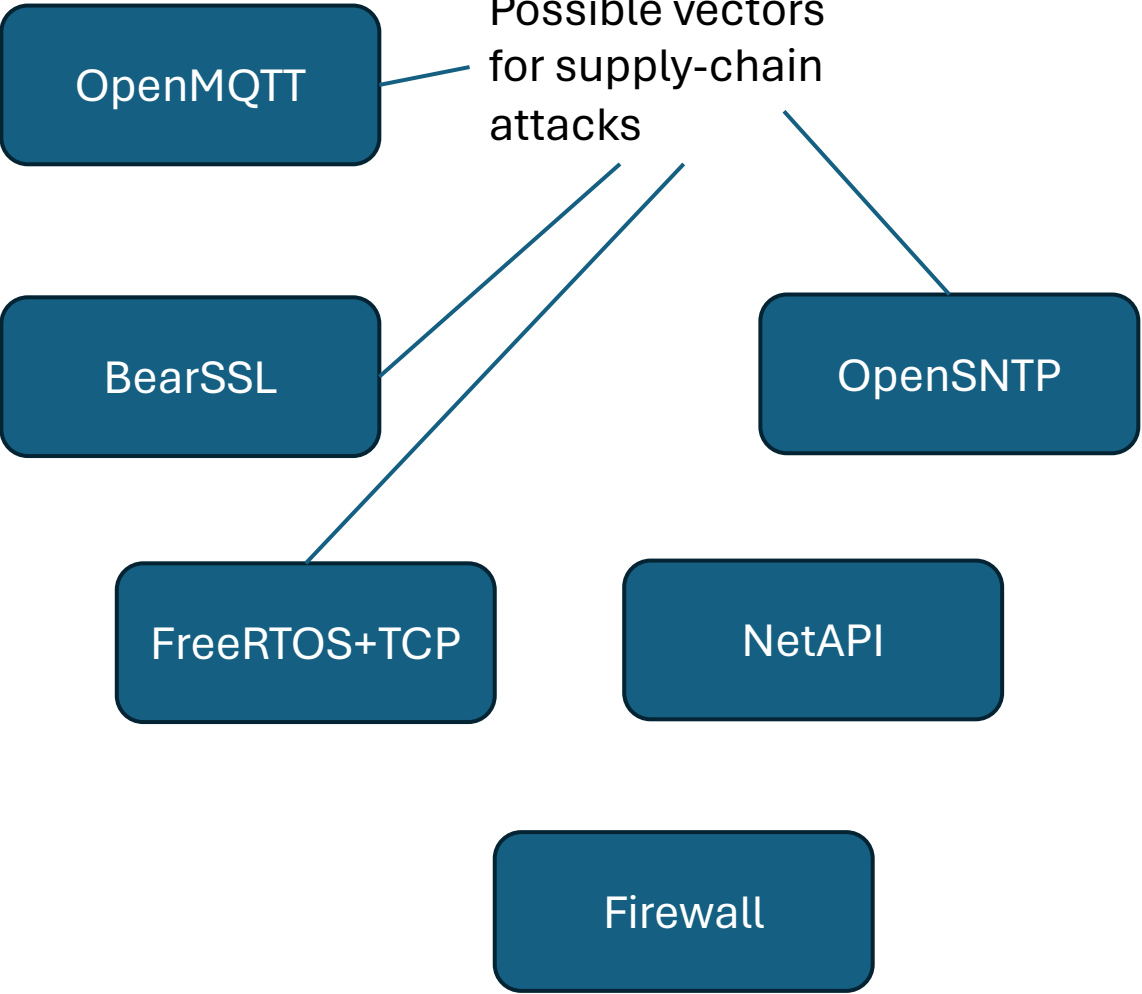
`cheriot-audit` checks them



Can also inspect compartment status



Case study: CHERIoT Network Stack



No compartment except the
firewall may access the ethernet
device directly

```
data.compartment.mmio_allow_list(ethernetDevice, {"Firewall"})
```


The TCP/IP compartment's
incoming frame API is exposed
only to the Firewall compartment

```
data.compartment.compartment_call_allow_list("TCPIP",  
"ethernet_receive_frame.*", {"Firewall"})
```

data.network_stack.all_connection_capabilities

What compartments can connect where?

```
[ {  
  "capability": {  
    "connection_type": "UDP",  
    "host": "pool.ntp.org",  
    "port": 123  
  },  
  "owner": "SNTP"  
}, {  
  "capability": {  
    "connection_type": "TCP",  
    "host": "cheriot.demo",  
    "port": 8883  
  },  
  "owner": "mqtt_demo"  
} ]
```

What happens
when a
supply-chain
attacker
compromises
the TCP/IP
stack?

They cannot call user code

They cannot allocate more memory
than their quota

They cannot control the firewall

They can tamper with packets to and
from the network

They can (currently) lie about DNS
responses

Summary

See <https://cheriot.org> for more information!



Memory safety is just the start



CHERI memory safety is a building block for compartmentalisation



Sealing is essential for rich abstractions



Compartmentalisation is a key part of supply-chain security