

CHERIoT Audit

David Chisnall



A photograph of a stack of colorful wooden blocks. In the foreground, a green block is at the base, with a yellow block stacked on top of it. A red block is positioned to the right of the yellow block. The background is filled with a blurred arrangement of various colored blocks, including yellow, blue, and red, creating a sense of depth.

Memory safety is a building block

CHERIoT provides layered security

System

Fine-grained auditing

Rich policy enforcement

Compartments

Limited blast radius

Protected secrets

Foundations

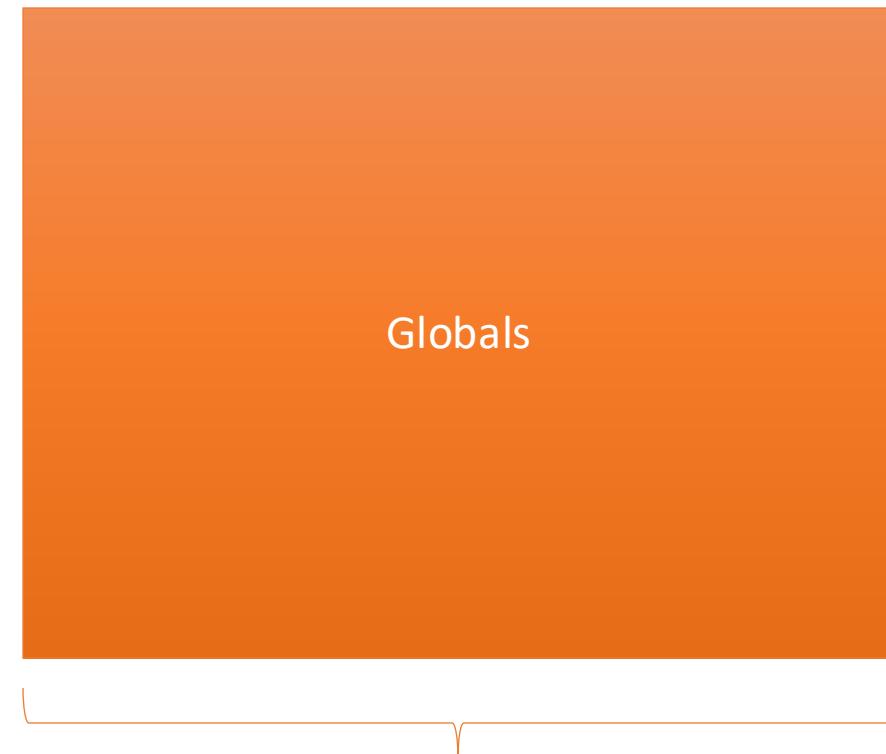
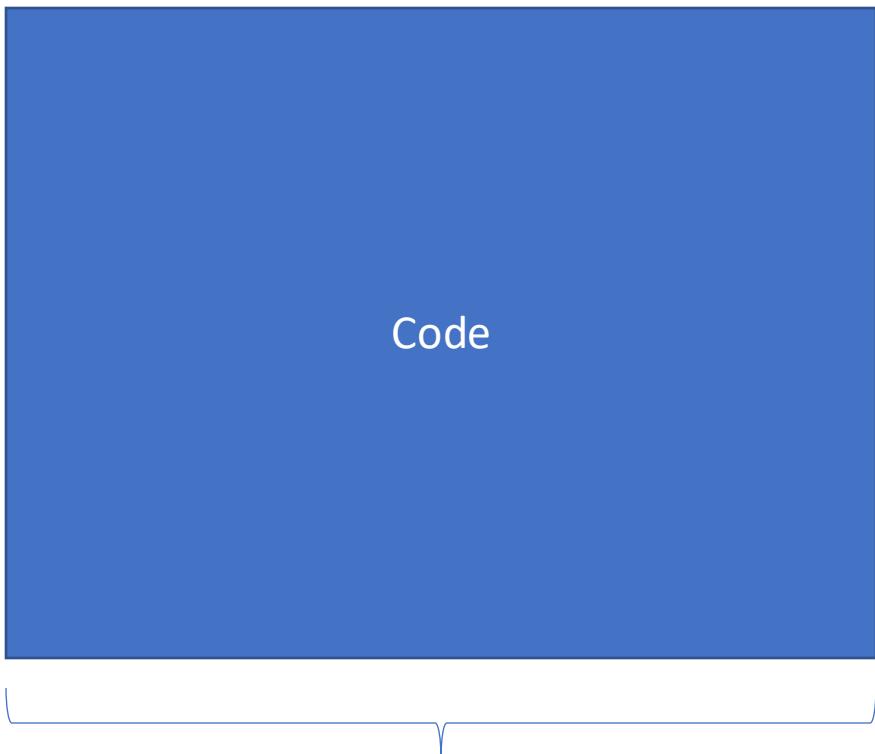
No use after free

Control-flow integrity

No buffer overflows

No pointer injection

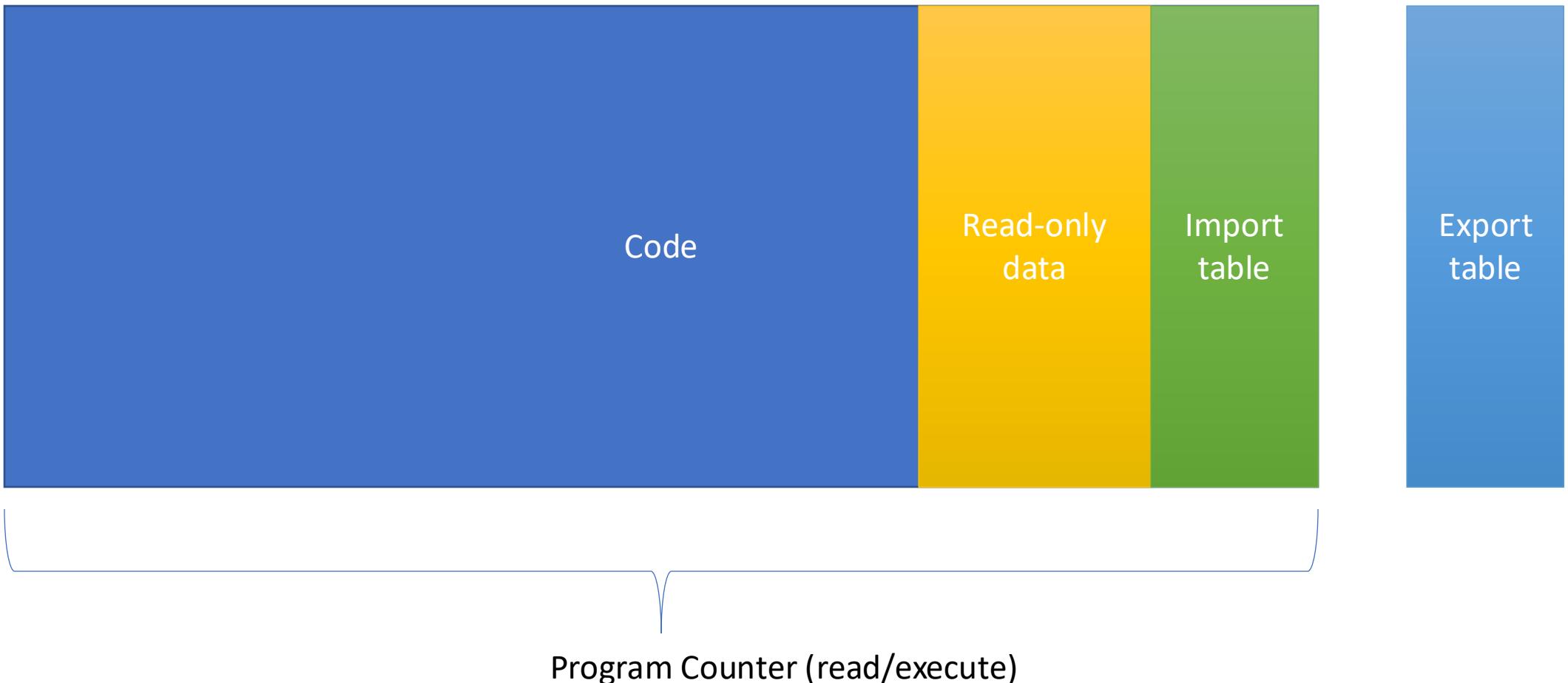
Compartments are code and data



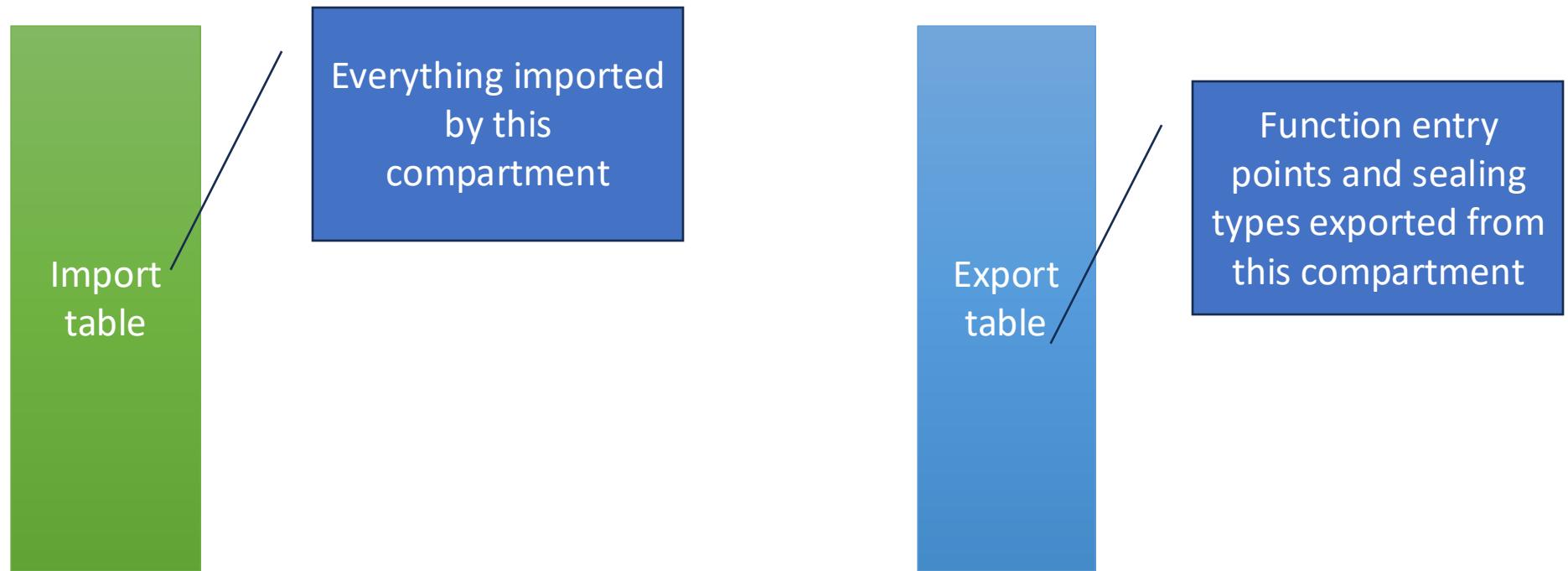
Program Counter (read/execute)

Global Pointer (read/write/global)

Compartments are code and data and exports



Compartment graphs are imports and exports



There are
many kinds of
import!

Static sealed objects

Shared objects

Functions

- From shared libraries or compartments
- With interrupts enabled or disabled

Sealing keys

The linker generates detailed JSON for exports

```
{  
  "export_symbol":  
  "__export.sealing_type.alloc.MallocKey",  
  "exported": true,  
  "kind": "SealingKey"  
},
```

The linker generates detailed JSON for exports

```
{  
  "export_symbol":  
  "t",  
  "export_alloc__Z20heap_quota_remainingP10S0bjStruct",  
  "exported": true,  
  "interrupt_status": "enabled",  
  "kind": "Function",  
  "register_arguments": 1,  
  "start_offset": 192  
},
```

The linker generates detailed JSON for imports

```
{  
  "kind": "SharedObject",  
  "length": 4,  
  "permits_load": true,  
  "permits_load mutable": true,  
  "permits_load_store_capabilities": true,  
  "permits_store": true,  
  "shared_object": "allocator_epoch",  
  "start": 2147604608  
}
```

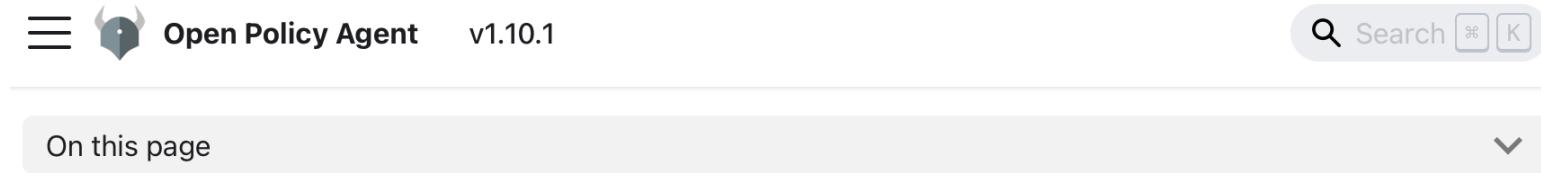
The linker generates detailed JSON for imports

```
{  
  "kind": "MMIO",  
  "length": 4096,  
  "permits_load": true,  
  "permits_load mutable": false,  
  "permits_load_store_capabilities": false,  
  "permits_store": true,  
  "start": 2197815296  
}
```

The linker generates detailed JSON for imports

```
{  
  "contents": "00040000 00000000 00000000 00000000 00000000  
00000000",  
  "kind": "SealedObject",  
  "sealing_type": {  
    "compartment": "alloc",  
    "key": "MalloclKey",  
    "provided_by":  
"build/cheriot/cheriot/release/cheriot.allocator.compartment"  
,  
    "symbol": "__export.sealing_type.alloc.MalloclKey"  
  }  
}
```

CHERIoT Audit uses Rego



The screenshot shows the OPA documentation interface. At the top, there is a header with a menu icon, the OPA logo (a stylized bull icon), the text "Open Policy Agent", and the version "v1.10.1". To the right of the header is a search bar with a magnifying glass icon, the word "Search", and keyboard shortcut keys. Below the header is a navigation bar with a dropdown menu labeled "On this page" and a downward arrow icon. The main content area features a large, bold title "Policy Language".

Policy Language

OPA is purpose built for reasoning about information represented in structured documents. The data that your service and its users publish can be inspected and transformed using OPA's native query language Rego.

What is Rego?

Rego was inspired by [Datalog](#), which is a well understood, decades old query language. Rego extends Datalog to support structured document models such as JSON.

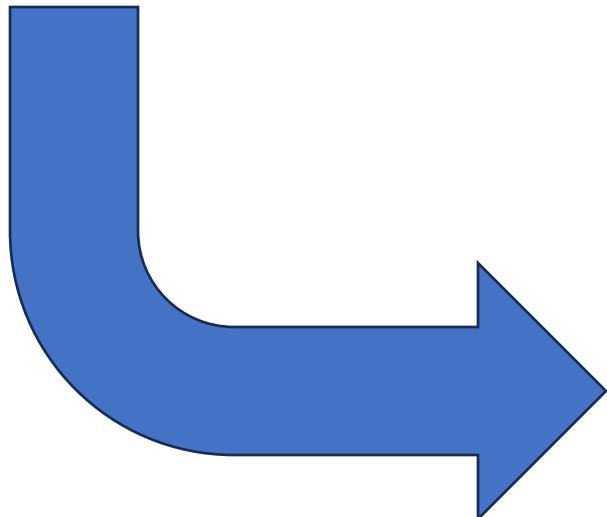
Rego queries are assertions on data stored in OPA. These queries can be used to define policies that enumerate instances of data that violate the expected state of the system.



Rego can inspect firmware

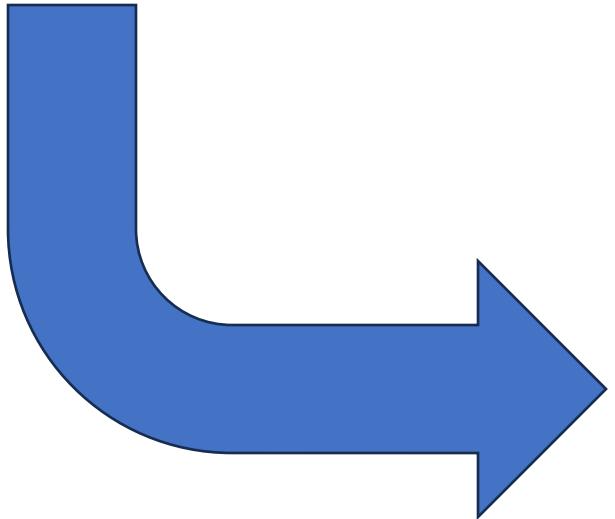
How much memory are all compartments allowed to allocate, between them?

```
sum([
  data.rtos.decode_allocator_capability(c).quota |
  c = input.compartments[_].imports[_] ;
  data.rtos.is_allocator_capability(c) ])
```



1078272

data.network_stack.all_connection_capabilities

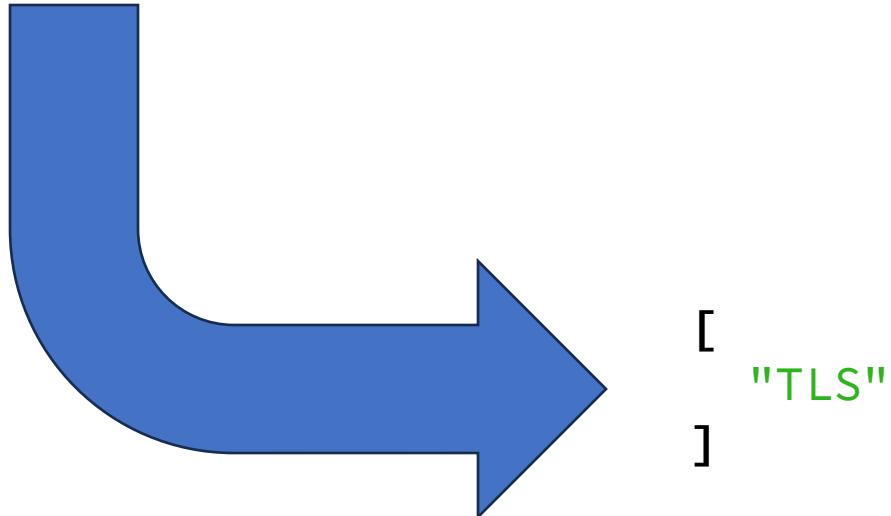


```
[  
  {  
    "capability": {  
      "connection_type": "UDP",  
      "host": "pool.ntp.org",  
      "port": 123  
    },  
    "owner": "SNTP"  
  },  
  {  
    "capability": {  
      "connection_type": "TCP",  
      "host": "example.com",  
      "port": 443  
    },  
    "owner": "https_example"  
  }]
```

Which compartments can connect to which servers?

Which compartments can send TCP data?

```
data.compartment.compartments_calling_export_matching("TCPIP",  
"network_socket_send\(.*\")")
```





Rego can enforce policies

Only the TLS stack may send TCP data

```
data.compartment.compartment_call_allow_list(  
    "TCPIP",  
    `network_socket_send\(.*\`,  
    {"TLS"})
```

Only the scheduler may access the CLINT

```
data.compartment.mmio_allow_list(  
    "clint",  
    {"scheduler"})
```

Only the SNTP compartment has write access to the published time result

```
data.compartment  
  .shared_object_writeable_allow_list(  
    "sntp_time_at_last_sync", {"SNTP"})
```

CHERIoT Audit enables fearless code reuse through end-to-end security design

ABI designed to be easy to inspect

Rich communications lowered to linker-visible information

Information exported from the linker

Policies expressed over the exported structure