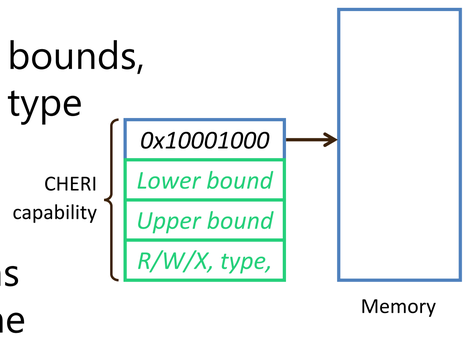
VeriCHERI: Exhaustive Formal Security Verification of CHERI at the RTL

Anna Lena Duque Antón[1], Johannes Müller[1], Philipp Schmitz[1], Tobias Jauch[1], Alex Wezel[1], Lucas Deutschmann[1], Mohammad R. Fadiheh[2], Dominik Stoffel[1] and Wolfgang Kunz[1]

[1] RPTU Kaiserslautern-Landau, Germany [2] Stanford University, USA

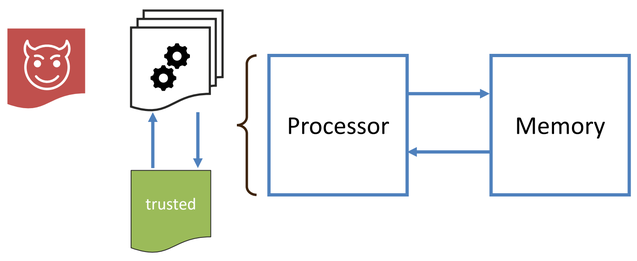
CHERI Protection

- Memory protection via capabilities
- Address pointers are enhanced with bounds, permissions, valid tag and an object type
- Legal memory accesses require valid and matching capabilities
- Security verification of CHERI designs is necessary, but creating trust for the entire system stack is challenging



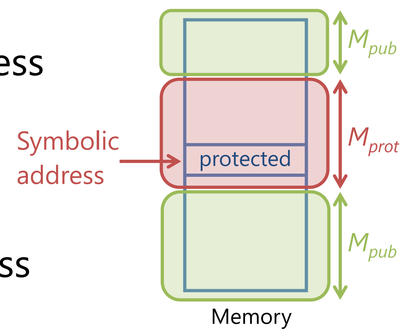
Attacker Model

- Capability-enhanced single-core processor executing mutually distrusting tasks
- A **trusted entity** securely manages context switches
- An **attacker task** tries to **break memory protection**



Formal Model

- In our model, two tasks only differ in the compartmentalization of the memory M into a set of accessible addresses (M_{pub}) and a set of protected addresses (M_{prot})
- Compartmentalization of M into M_{pub} and M_{prot} is enforced by CHERI capabilities



- We introduce a symbolic memory address that can be chosen freely by the solver
- Capabilities of an attacker task are fully symbolic, except for the fact that they deny access to the symbolic address
- Confidentiality 1-safety property:

$$\mathrm{AG}(\mathrm{cheri\_protected}(\mathrm{symbolic\_addr}) \rightarrow (\mathrm{read\_mem\_access} \rightarrow \mathrm{mem\_addr} \neq \mathrm{symbolic\_addr}))$$

- Integrity 1-safety property:

$$\mathrm{AG}(\mathrm{cheri\_protected}(\mathrm{symbolic\_addr}) \rightarrow (\mathrm{write\_mem\_access} \rightarrow \mathrm{mem\_addr} \neq \mathrm{symbolic\_addr}))$$

Verification Flow

Confidentiality Interval Property:

```
t: cheri_protected(symbolic_addr)
implies
t: !read_mem || mem_addr != symbolic_addr
```

Integrity Interval Property:

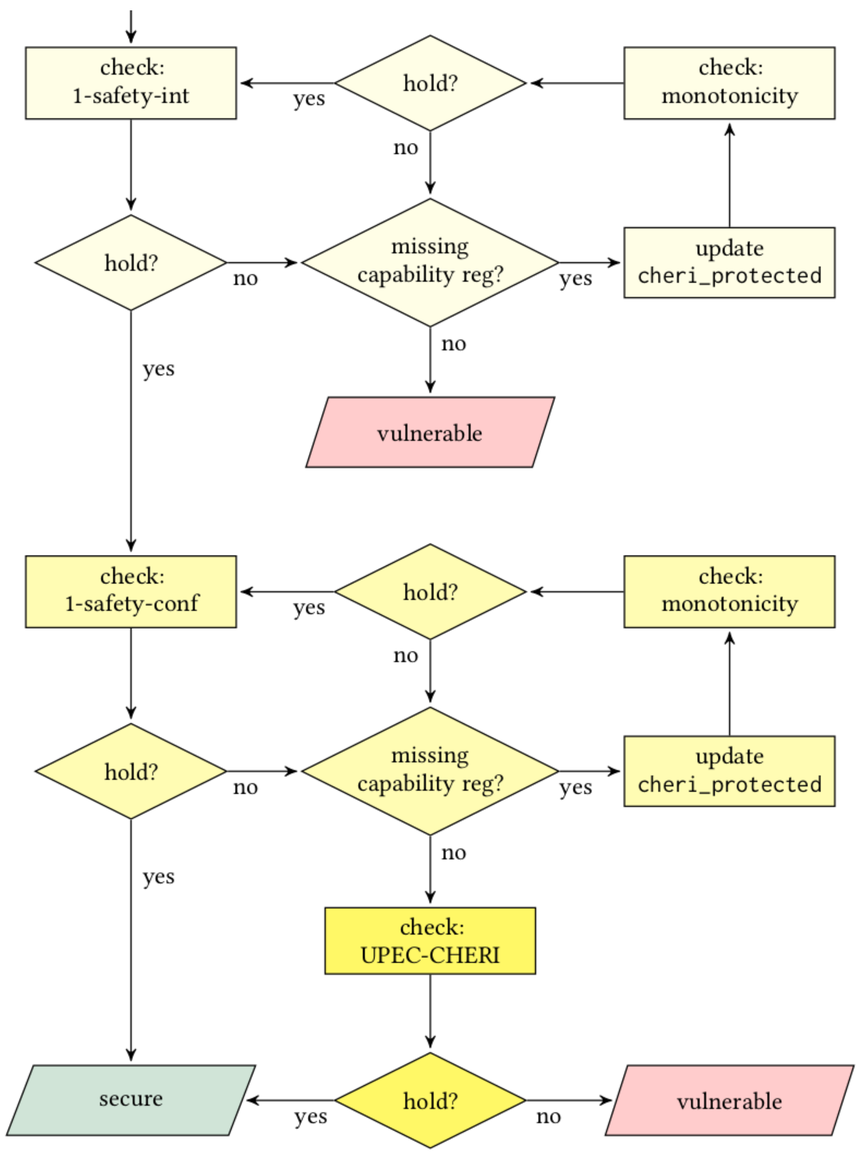
```
t: cheri_protected(symbolic_addr)
implies
t: !write_mem || mem_addr != symbolic_addr
```

Monotonicity Interval Property:

```
t: cheri_protected(symbolic_addr)
implies
t: cheri_protected(symbolic_addr)
```

UPEC-CHERI Interval Property:

```
t: cheri_protected(symbolic_addr)
t: $M_pub == $M'_pub && $P == $P'
implies
t + k: $P_arch == $P'_arch
```



Case Study on CHERIoT Ibex

- 32-bit RISC-V microcontroller implementing RV32IMCB and the CHERIoT ISA extension in a 2-stage pipeline

Property	Iteration	Result	Runtime	Memory	Description
1-safety-integrity	1	fail	< 1 min	4.3 GB	*Bug*: setup guide specification of protection enable pin
	2	fail	< 1 min	4.7 GB	*Bug*: capability stores across capability bounds
	3	hold	7 min	4.8 GB	-
Monotonicity	1-9	fail	≤ 1 min	4-5 GB	Missing capability register or pipeline buffer
	10	hold	15 min	6.2 GB	-
1-safety-confidentiality					
⟶ data	1	hold	7 min	7.3 GB	-
⟶ instructions	1	fail	< 1 min	4.8 GB	Instruction fetched from outside PCC bounds
UPEC-CHERI	1	fail	31 min	3.7 GB	*Side channel*: exception timing depends on fetched data
	2	hold	18 min	6.3 GB	-

- VeriCHERI detected a **Transient Execution Attack** vulnerability
  - Branch to address **outside of PCC bounds**
  - Illegal instruction fetch **raises an exception**
  - Exception is **delayed depending on two bits** of the fetched data
  - **Performance counter change** depending on the two bits

→ Measure the execution time to **probe two bits** for an **arbitrary protected address**

Conclusion

- VeriCHERI detected several security issues including a vulnerability to a Transient Execution Attack, which is not detectable by previous methods
- Formulating the security objectives as single-cycle interval properties allows us to introduce a scalable iterative verification flow
- The developed invariants are implemented as symbolic verification IPs which may be reused for other CHERI designs

RPTU Rheinland-Pfälzische Technische Universität Kaiserslautern Landau

RISC-V Summit Europe, Paris
May 12th – May 15th 2025