Borrowed Capabilities: Flexibly Enforcing Revocation on a Capability Architecture

Thijs Vercammen (KU Leuven)

Thomas Van Strydonck (KU Leuven)

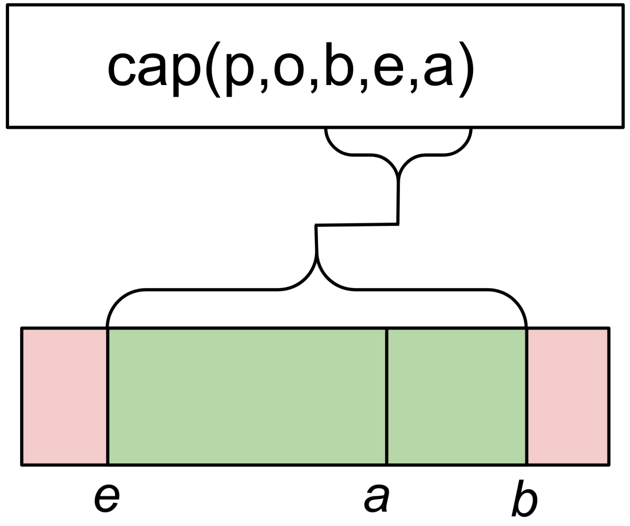
Dominique Devriese (Vrije Universiteit Brussel)

Introduction

- Memory vulnerabilities
    - Spatial
    - Temporal
- High-level countermeasures
    - Garbage collected programming languages
    - Strong type systems
- Low-level countermeasures

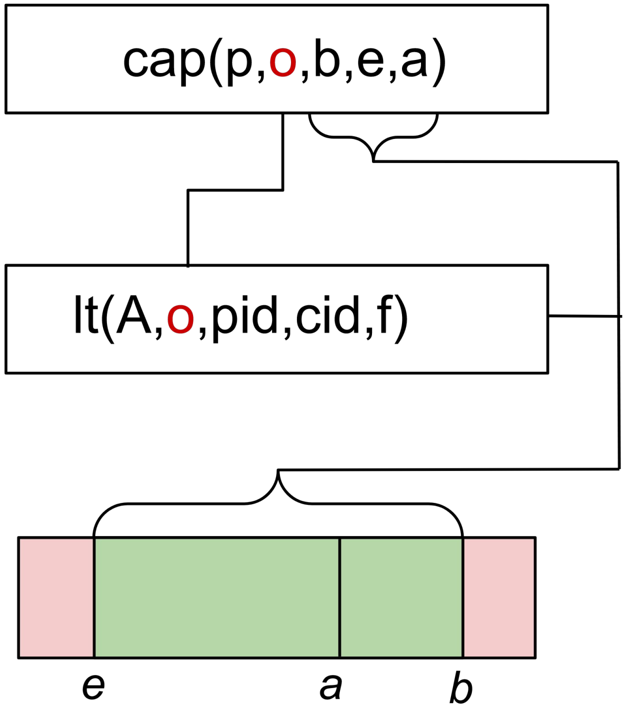
Introduction

- Hardware capabilities
  - Good spatial protection
  - No inherent temporal protection
    - Revocation
    - Borrowed capabilities



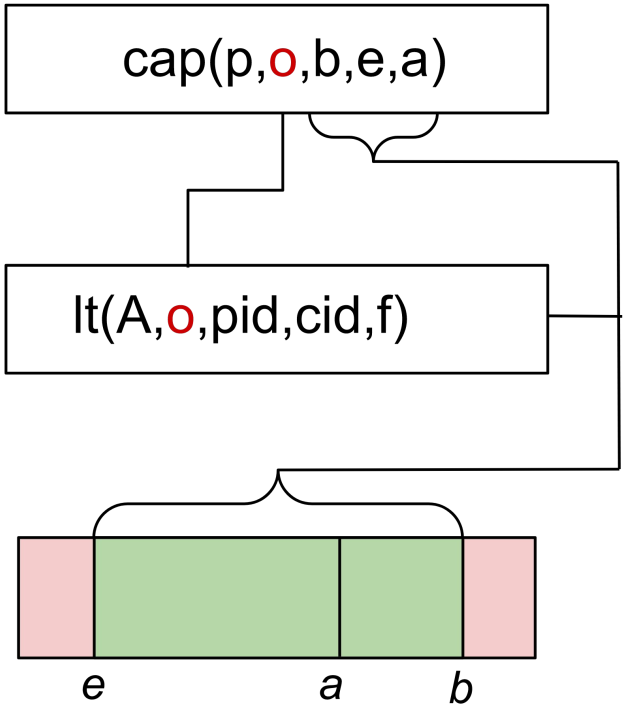
General Idea

- Goals
    - Caller revoke access from callee
    - Callee revoke access from caller
        - Mutation XOR Aliasing



General Idea

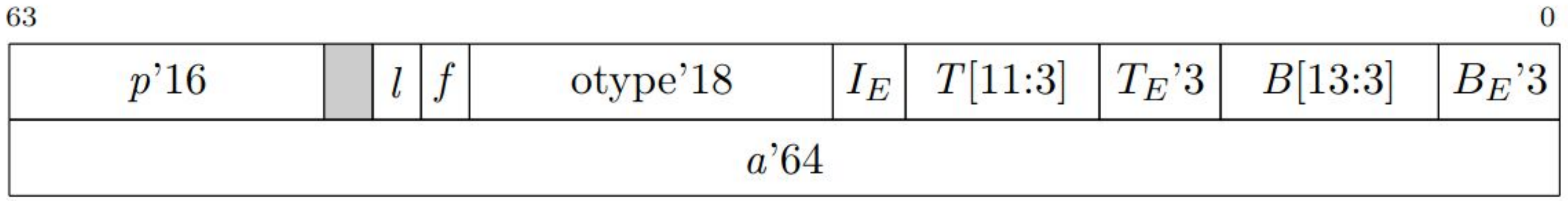
- Goals
  - Caller revoke access from callee
  - Callee revoke access from caller
    - Mutation XOR Aliasing
- Lifetime tokens as scopes



RISC-V: The Free and Open
Instruction Set Architectu

CHERI Capabilities

- CHERI-RISC-V
- Sail ISA description language
  - Can generate an emulator
- CHERI-LLVM



Linear Capabilities

- Capabilities that can not be copied
- Holder has guarantee no copies exist
  - Exclusive access
- Not trivial to implement in ISA

Lifetime Tokens

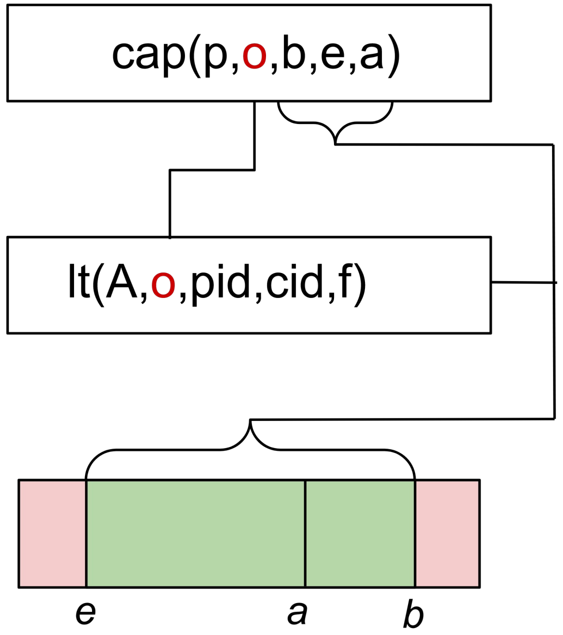
- Alive
  - Can be used to
    - Borrow
    - Dereference borrowed capabilities
  - Linear
- Dead
  - After kill operation
  - Proof of lifetime’s end
  - Lose linearity

$$\begin{tikzcd}
\boxed{\phantom{\text{lt(A, o, \_, \_, \_)}}} \arrow[r, "\text{CCreateToken}"] & \boxed{\text{lt(A, o, \_, \_, \_)}}
\end{tikzcd}$$

$$\begin{tikzcd}
\boxed{\text{lt(A, o, \_, \_, \_)}} \arrow[r, "\text{CKillToken}"] & \boxed{\text{lt(D, o, \_, \_, \_)}}
\end{tikzcd}$$

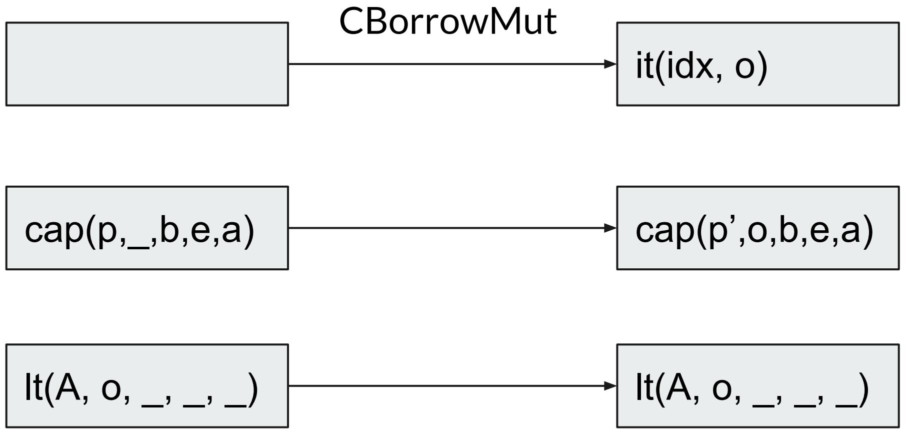
Borrowing

- Two operations
    - Immutable
    - Mutable
- Requires lifetime token



Borrowing

- Locks original capability away
  - Borrow table
- Borrowed capability in its place
- Index tokens
  - Retrieve original capability



Extra Features

- Lifetime fractions
  - Multiple living lifetime tokens
- Lifetime hierarchies
  - Sublifetimes

Example Program

```
li              x1 0x5
sw.cap          x1 0(c2)      #let mut
                              #x = 5;

CCreateToken    c31 c0        #{
CBorrowMut      c4 c2 c31     #y = &mut x;
li              x1 0x6
sw.cap          x1 0(c2)      #*y = 6;

CMove           c30 c31
CCreateToken    c31 c30       #{
CBorrowImmut    c5 c2 c31     #z = &y;
lw.cap          x7 0(c2)      #temp = 6;
CKillToken      c31 c31       #}

CRetrieveIndex c5 c5 c31
CUnlockToken    c30 c31
CKillToken      c30 c30       #}
CRetrieveIndex c4 c4 c30
```

Contributions

- Concept and design of borrowed capabilities
- Implementation in Sail
  - Linear capabilities
  - Borrowed capabilities
- LLVM extension for borrowed capabilities

Conclusion

- Revocation on capability machines
- Practical open questions
    - Hardware implementation
        - Borrow table
        - Some instructions
    - 2^{17} unique lifetimes

Questions?