ENFORCING CONTROL-FLOW INTEGRITY IN VIRTUALIZED ENVIRONMENTS ON ARM PLATFORMS

The 5th Italian Workshop on Embedded Systems
February 9, 2021

Gabriele Serra
Pietro Fara
Giorgiomaria Cicero
Alessandro Biondi

Sant'Anna
School of Advanced Studies – Pisa

MOTIVATION AND CONTEXT

Can we really trust our embedded devices?
Imagine if this will happen on our cars or trains.

iOS 14.4 and iPadOS 14.4

Released January 26, 2021

WebKit

Available for: iPhone 6s and later, iPad Air 2 and later, iPad mini 4 and later, and iPod touch (7th generation)

Impact: A remote attacker may be able to cause arbitrary code execution. Apple is aware of a report that this issue may have been actively exploited.

Description: A logic issue was addressed with improved restrictions.

CVE-2021-1871: an anonymous researcher

CVE-2021-1870: an anonymous researcher

MOTIVATION AND CONTEXT

Embedded systems

- OSes are written in C/C++
- Exposed to public access

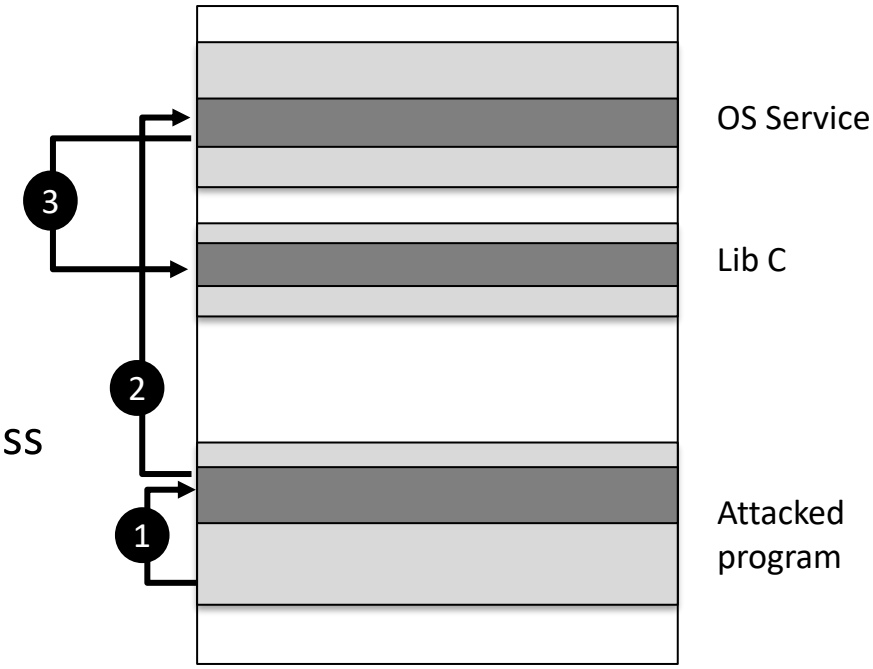
MOTIVATION AND CONTEXT

Embedded systems

- OSes are written in C/C++
- Exposed to public access

Susceptible to attacks

- **Code-Reuse-Attacks**
  - Re-use existent piece of code
  - I.e. flow deviated to gain root access
- Return-Oriented programming



MOTIVATION AND CONTEXT

Embedded systems

- OSes are written in C/C++
- Exposed to public access

Susceptible to attacks

- **Code-Reuse-Attacks**
  - Re-use existent piece of code
  - I.e. flow deviated to gain root access
- Return-Oriented programming

Mitigation technique

- Address space layout randomization (ASLR)
- **Integrity check of control flow (CFI)**

INTRODUCTION

CFI basic idea:

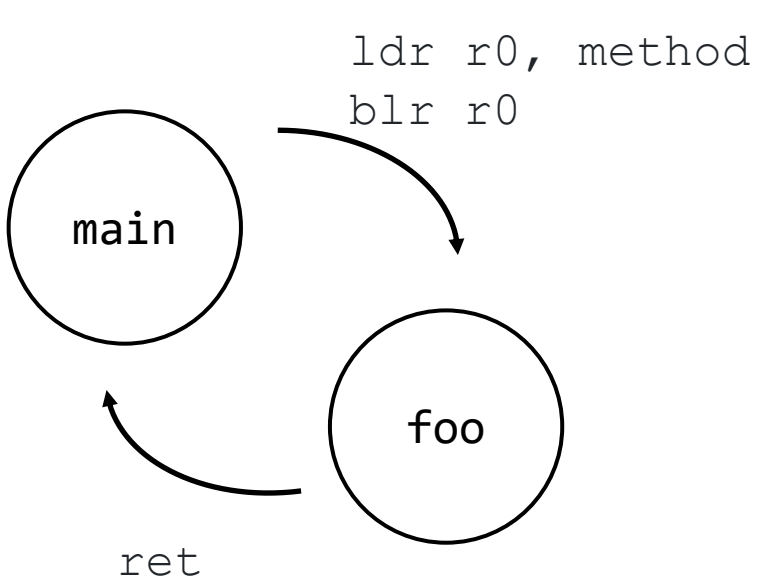
- build a Control Flow Graph (CFG) of the program
- **CFG** defines the legal execution

CFI basic idea:

- build a Control Flow Graph (CFG) of the program
- **CFG** defines the legal execution

```
void foo() { ... }

void main() {
        ...
        obj->method = foo;
        obj->method();
        ...
}
```



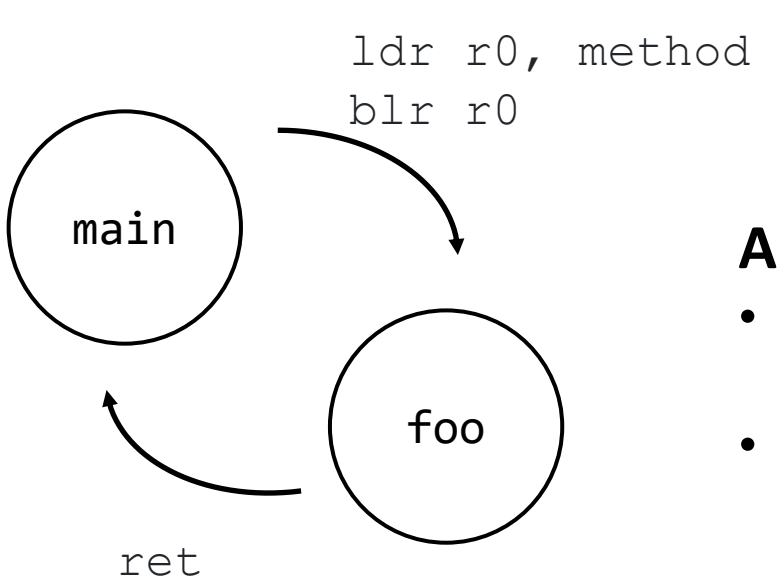
INTRODUCTION

CFI basic idea:

- build a Control Flow Graph (CFG) of the program
- **CFG** defines the legal execution

```
void foo() { ... }

void main() {
        ...
        obj->method = foo;
        obj->method();
        ...
}
```

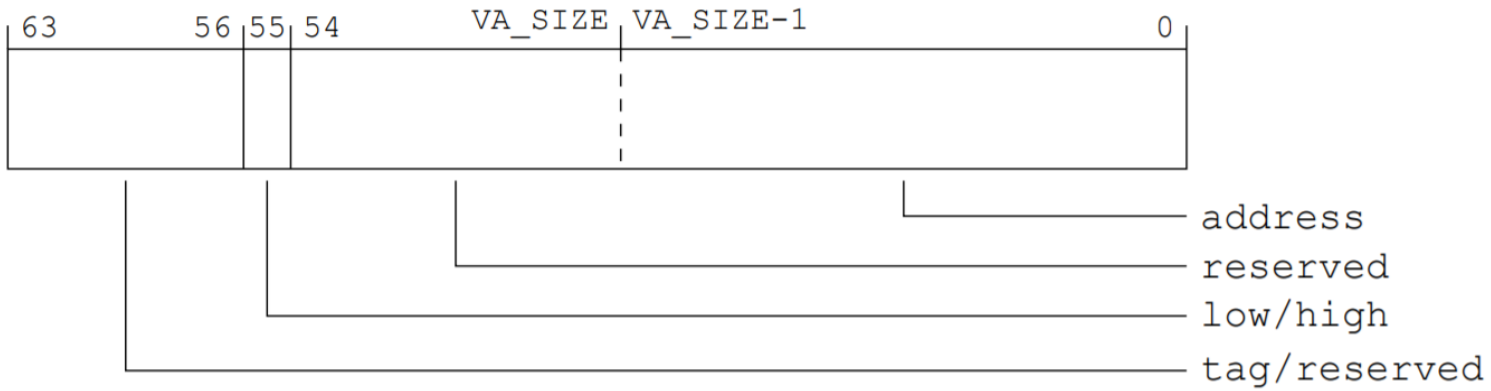


ARM introduced hw supports:

- Branch Targets Identification (**BTI**)
  - Forward branch protection
- Pointer Authentication Code (**PAC**)
  - Backward branch protection

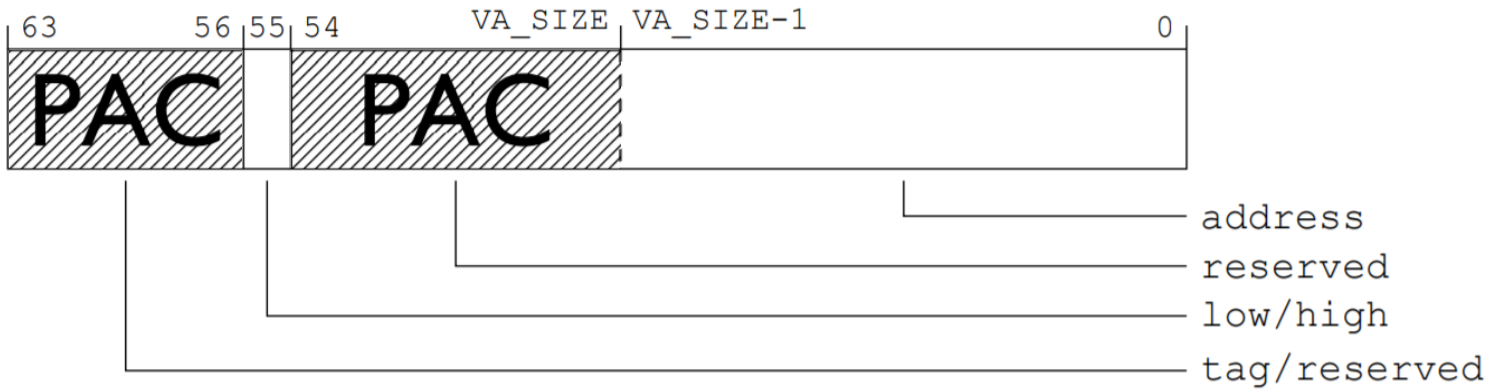
Pointers in AArch64:

- Address represented on `[0:VA_SIZE]`
- Typically `VA_SIZE = 48`
- Empty `[VA_SIZE:54]` and `[56:63]`



AArch64 Pointer Authentication Codes (PAC):

- Hardware-based CFI
- Leverages empty space on 64-bit virtual addresses
- Append a Message Authentication Code (MAC)



BACKGROUND

Introduced two insns:

- PAC
- AUTH

PAC Creation takes:

- A pointer
- A 64-bit context
- A 128-bit secret key

PAC algorithm ‘H’ can be:

- QARMA
- Implementation defined

Introduced two insns:

- PAC
- AUTH

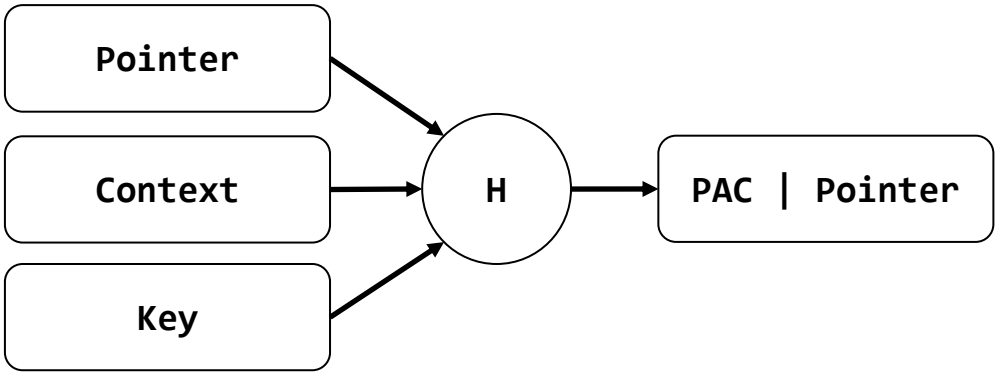
PAC Creation takes:

- A pointer
- A 64-bit context
- A 128-bit secret key

PAC algorithm ‘H’ can be:

- QARMA
- Implementation defined

PAC:



Introduced two insns:

- PAC
- AUTH

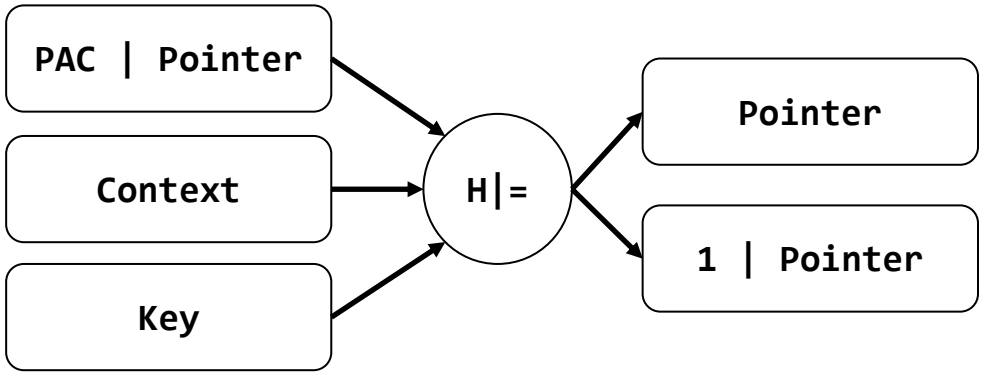
PAC Creation takes:

- A pointer
- A 64-bit context
- A 128-bit secret key

PAC algorithm ‘H’ can be:

- QARMA
- Implementation defined

AUTH:



Pointer authentication ISSUES

- Weakness against signing gadget

Pointer authentication ISSUES

- Weakness against signing gadget
- Weakness against kernel attackers
  - Cross EL/Key forgeries
  - Key memory leak

Pointer authentication ISSUES

- Weakness against signing gadget
- Weakness against kernel attackers
  - Cross EL/Key forgeries
  - Key memory leak
- Attack cannot be detected
  - Reported to ARM by Cicero et al in 2019
  - Will be fixed with FPAC in ARM v8.6

Pointer authentication ISSUES

- Weakness against signing gadget
- Weakness against kernel attackers
  - Cross EL/Key forgeries
  - Key memory leak
- Attack cannot be detected
  - Reported to ARM by Cicero et al in 2019
  - Will be fixed with FPAC in ARM v8.6
- **Available only on ARM ^v8.3**
- **Currently no COTS SoC available**

Pointer authentication ISSUES

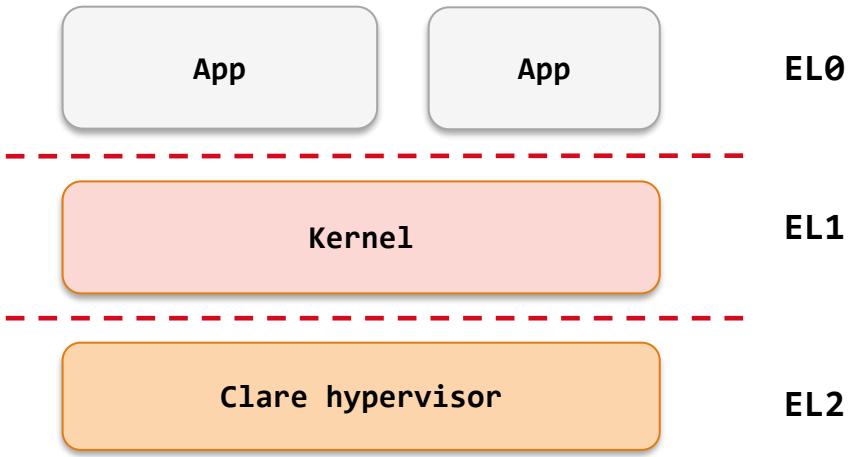
- Weakness against signing gadget
- Weakness against kernel attackers
  - Cross EL/Key forgeries
  - Key memory leak
- Attack cannot be detected
  - Reported to ARM by Cicero et al in 2019
  - Will be fixed with FPAC in ARM v8.6
- Available only on ARM ^v8.3
- Currently no COTS SoC available

Leverage on virtualization to counteract these issues!

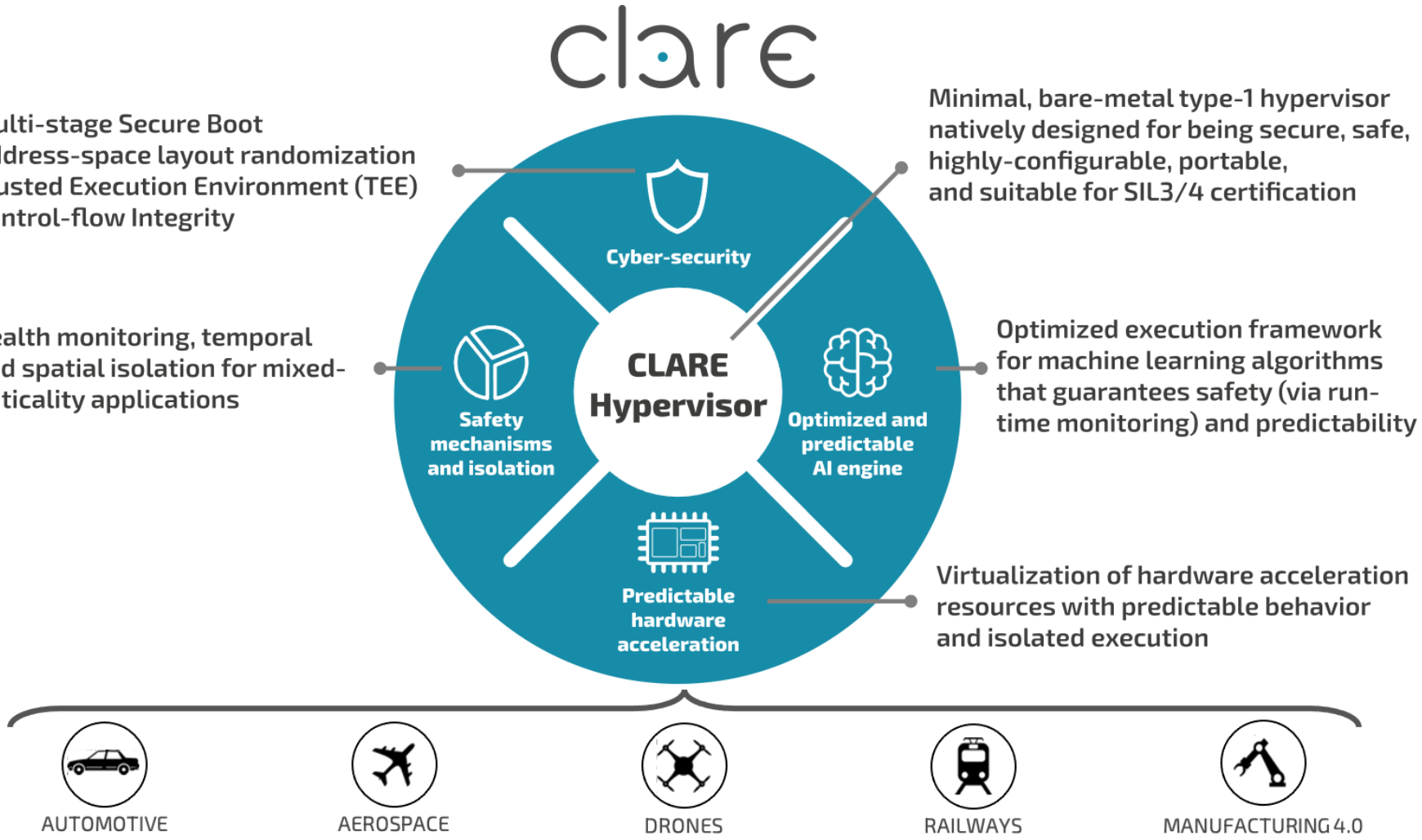
PROPOSED APPROACH

Leverage on CLARE hypervisor to:

1. Improve key management
2. Provide PA to all AArch64 SoC



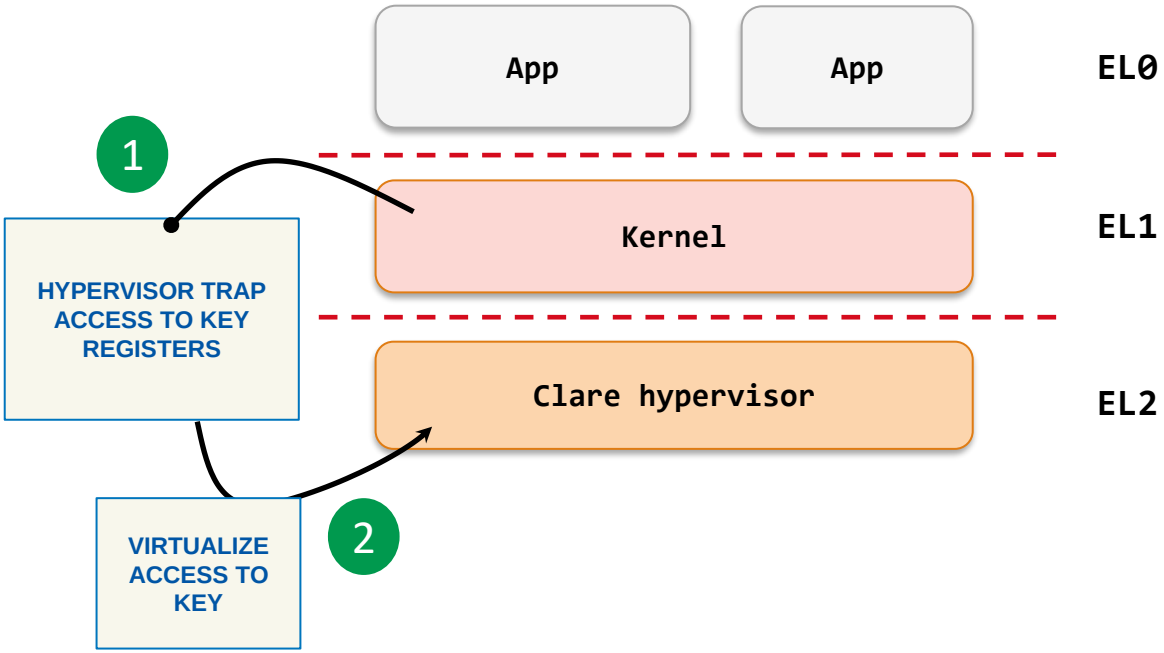
CLARE IN A NUTSHELL



Check it out @ clare.santannapisa.it

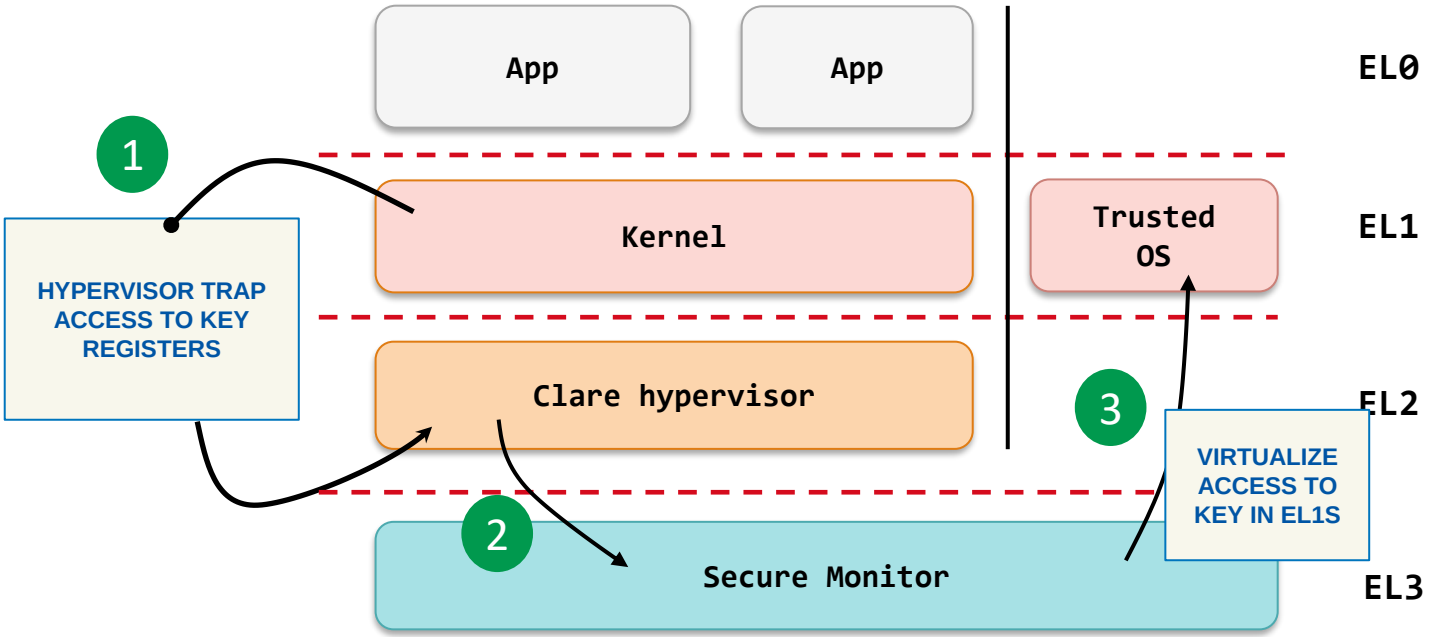
Leverage on CLARE hypervisor to:

1. Improve key management
2. Provide PA to all AArch64 SoC



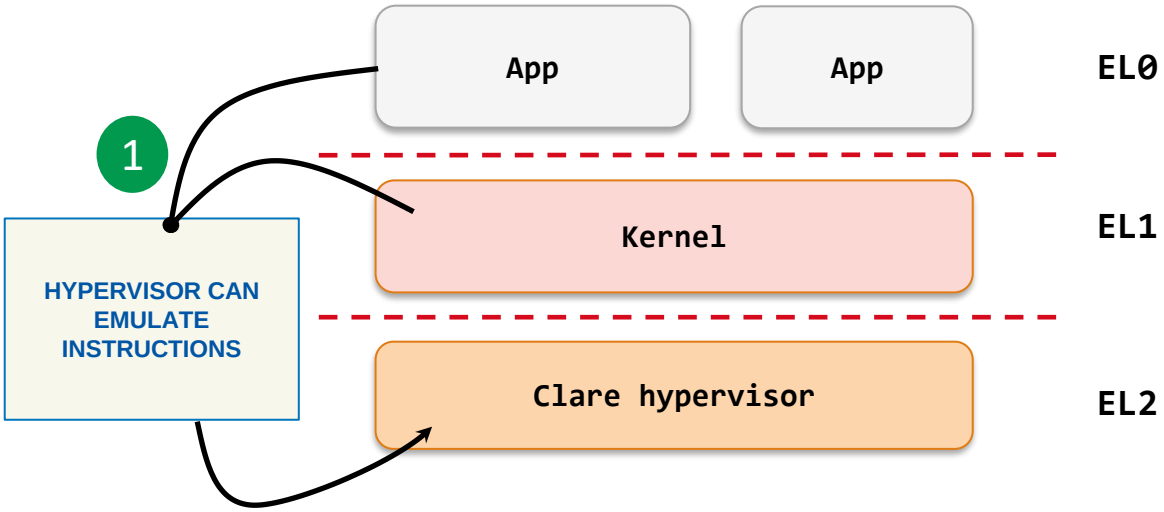
Leverage on CLARE hypervisor to:

1. Improve key management
2. Provide PA to all AArch64 SoC



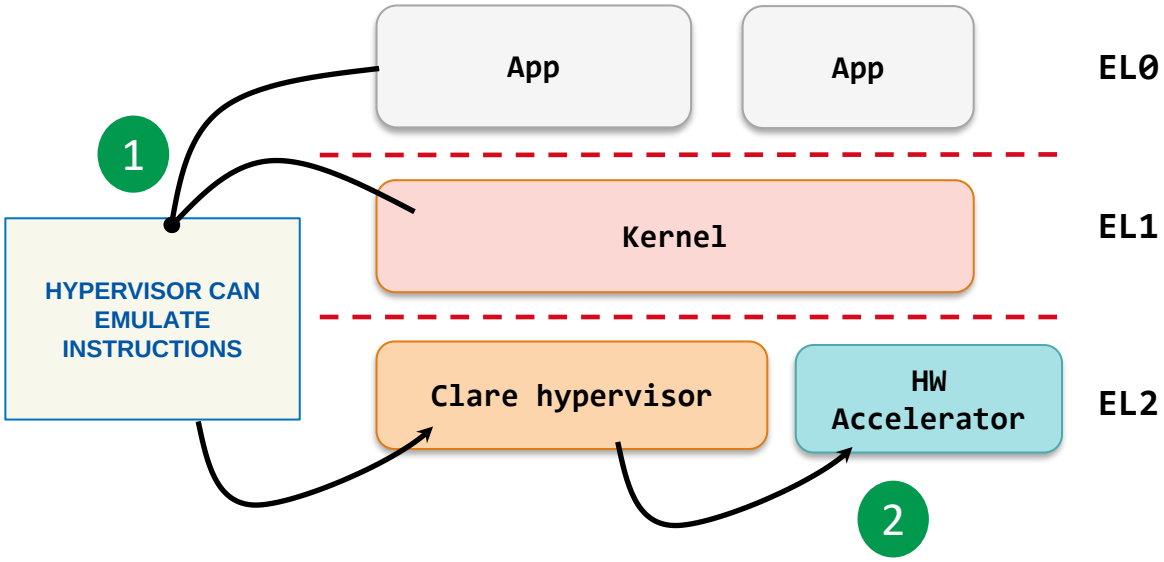
Leverage on CLARE hypervisor to:

1. Improve key management
2. Provide PA to all AArch64 SoC



Leverage on CLARE hypervisor to:

1. Improve key management
2. Provide PA to all AArch64 SoC
    1. PA can be HW accelerated
    2. **Clare** can detect attacks



CONCLUSIONS & FUTURE WORKS

Advances in embedded system connectivity and technologies need to be followed by corresponding advancements in associated security protection!

Future directions

- Measure aggregate overhead when protecting 1 domain out of N
- Apply PAC in selective way (only “more at risk” processes)
- Simplify access to PAC features
- Fully integrate the stack for production in **CLARE** hypervisor

THANK YOU

QUESTIONS?

gabriele.serra@santannapisa.it
gabrieleserra.ml