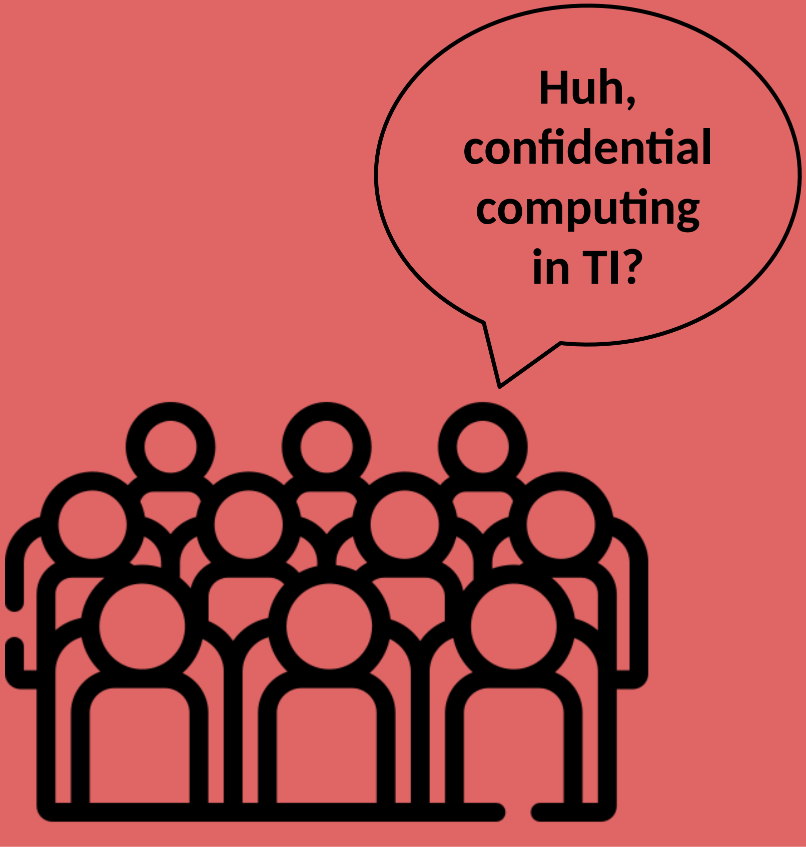
Securing Memory Isolation in Texas Instruments Microcontrollers

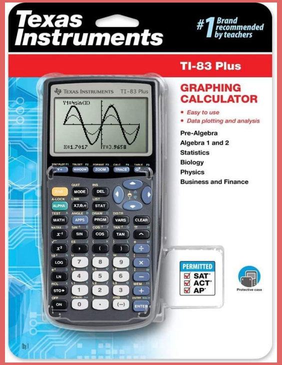
Marton Bognar

(based on work with Cas Magnus, Frank Piessens, Jo Van Bulck)

DistriNet, KU Leuven, Belgium

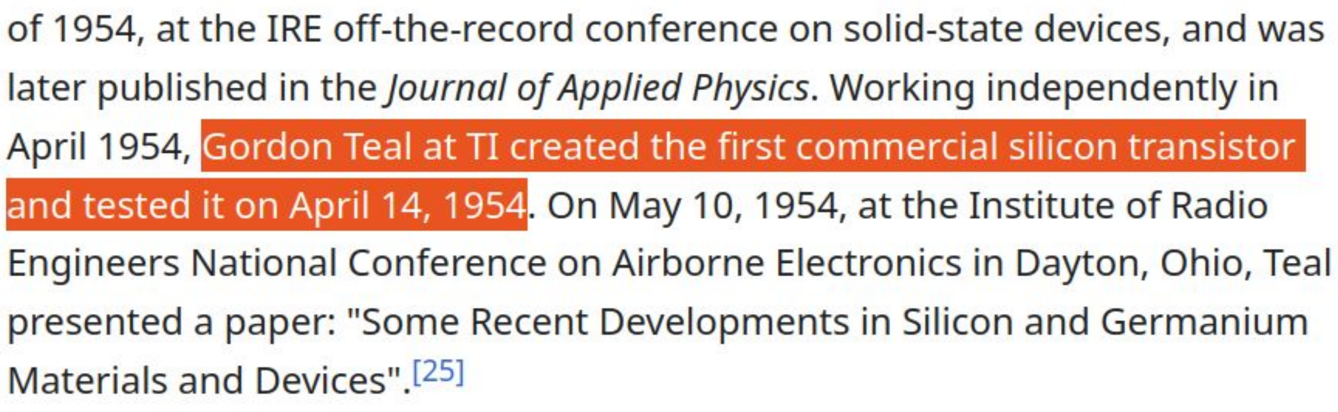


People from flaticon.com



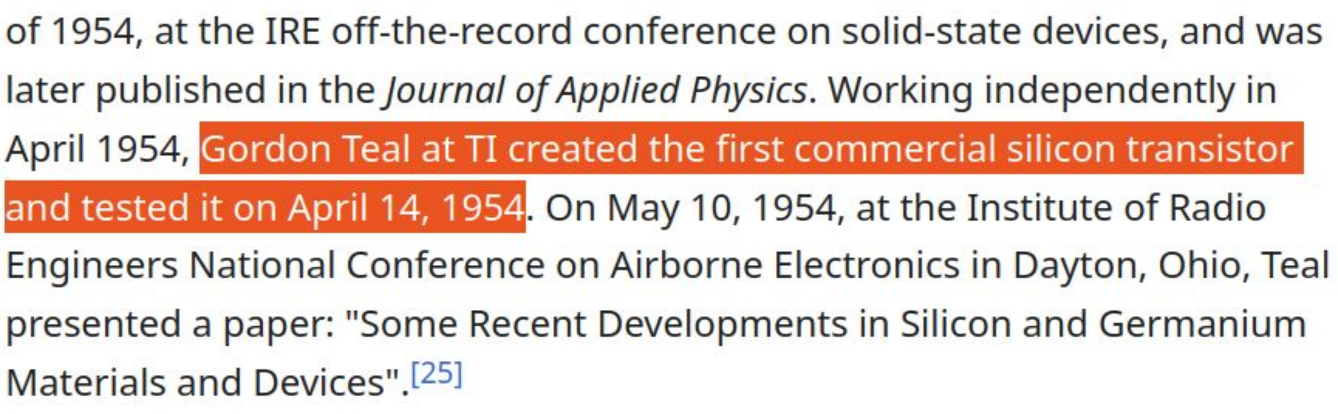
Huh, confidential computing in TI?

Texas Instruments



of 1954, at the IRE off-the-record conference on solid-state devices, and was later published in the *Journal of Applied Physics*. Working independently in April 1954, Gordon Teal at TI created the first commercial silicon transistor and tested it on April 14, 1954. On May 10, 1954, at the Institute of Radio Engineers National Conference on Airborne Electronics in Dayton, Ohio, Teal presented a paper: "Some Recent Developments in Silicon and Germanium Materials and Devices".[25]

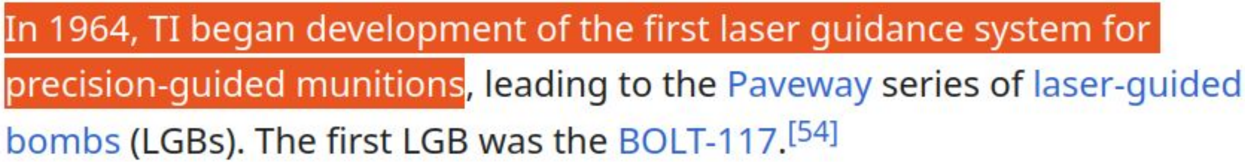
Texas Instruments



of 1954, at the IRE off-the-record conference on solid-state devices, and was later published in the *Journal of Applied Physics*. Working independently in April 1954, Gordon Teal at TI created the first commercial silicon transistor and tested it on April 14, 1954. On May 10, 1954, at the Institute of Radio Engineers National Conference on Airborne Electronics in Dayton, Ohio, Teal presented a paper: "Some Recent Developments in Silicon and Germanium Materials and Devices".[25]

Jack Kilby, an employee at TI, invented the integrated circuit in 1958.[26] Kilby recorded his initial ideas concerning the integrated circuit in July 1958, and successfully demonstrated the world's first working integrated circuit on September 12, 1958.[27] Six months later, Robert Noyce of Fairchild Semiconductor (who went on to co-found Intel) independently developed

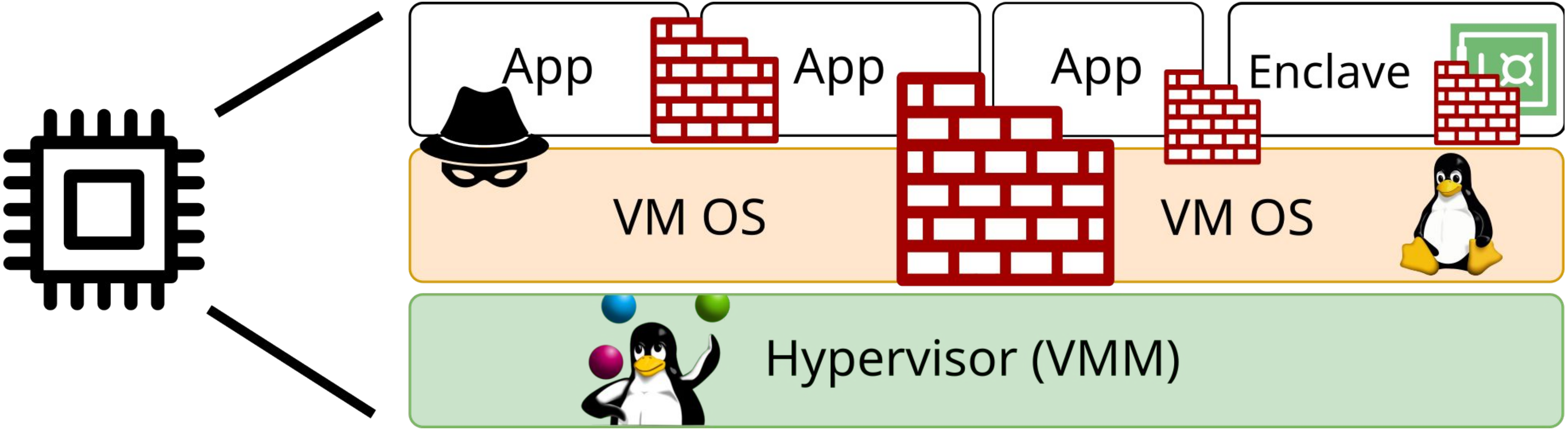
Texas Instruments



In 1964, TI began development of the first laser guidance system for precision-guided munitions, leading to the Paveway series of laser-guided bombs (LGBs). The first LGB was the BOLT-117.[54]

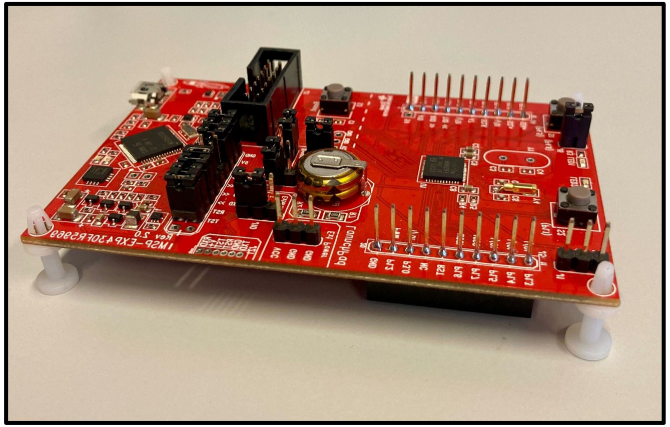
Part I: Security

Memory isolation: Conventional “high-end” systems



- Software **protection domains:** Processes, VMs, enclaves
- CPU support **memory isolation:** Virtual memory + privilege rings

Texas Instruments MSP430 microcontroller

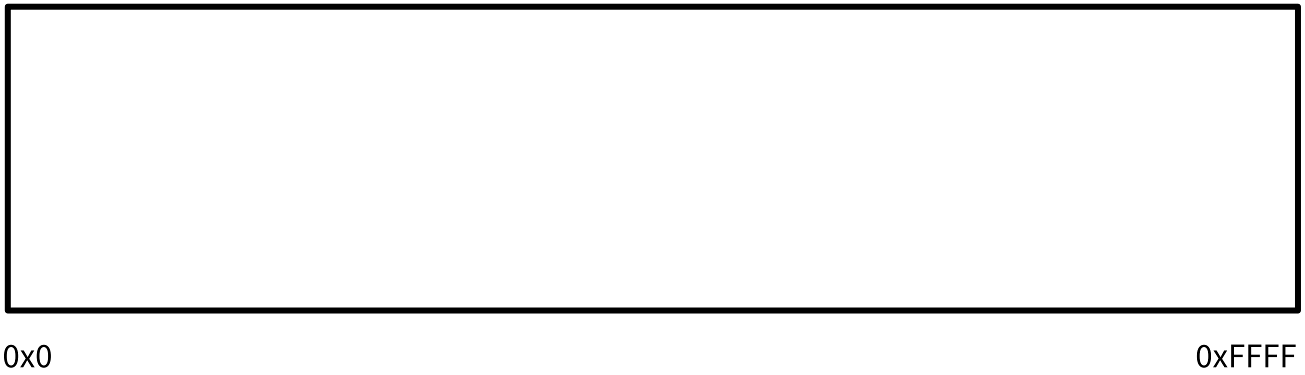


TI Embedded Security Portfolio –

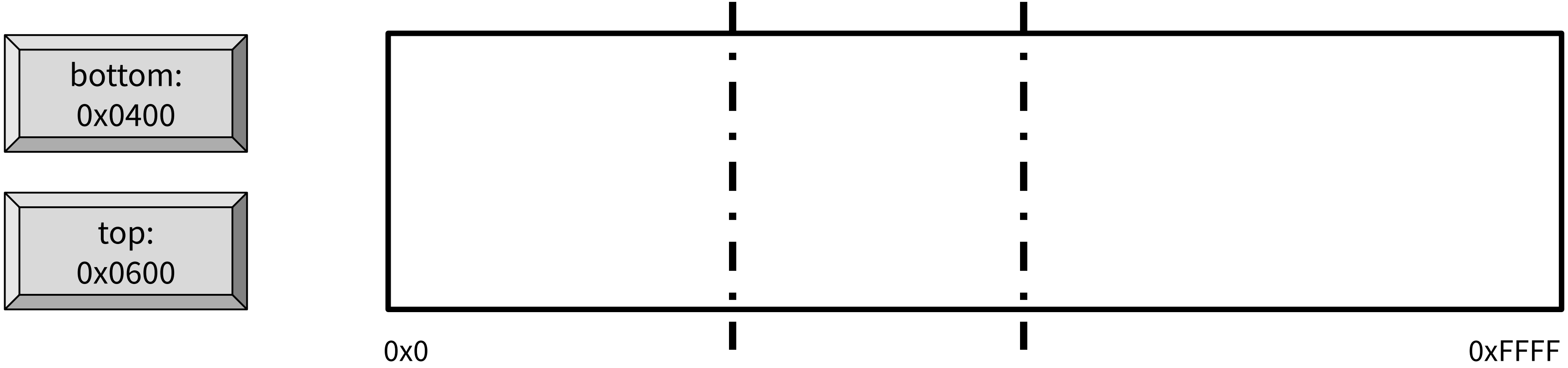
Security is hard, TI makes it easier

- Low-power microcontrollers
- FRAM edition (2014) with <u>security features:</u>
  - Physical tamper protection
  - Hardware AES cryptographic unit
  - Memory protection unit (MPU)
  - **Intellectual Property Encapsulation (IPE)**

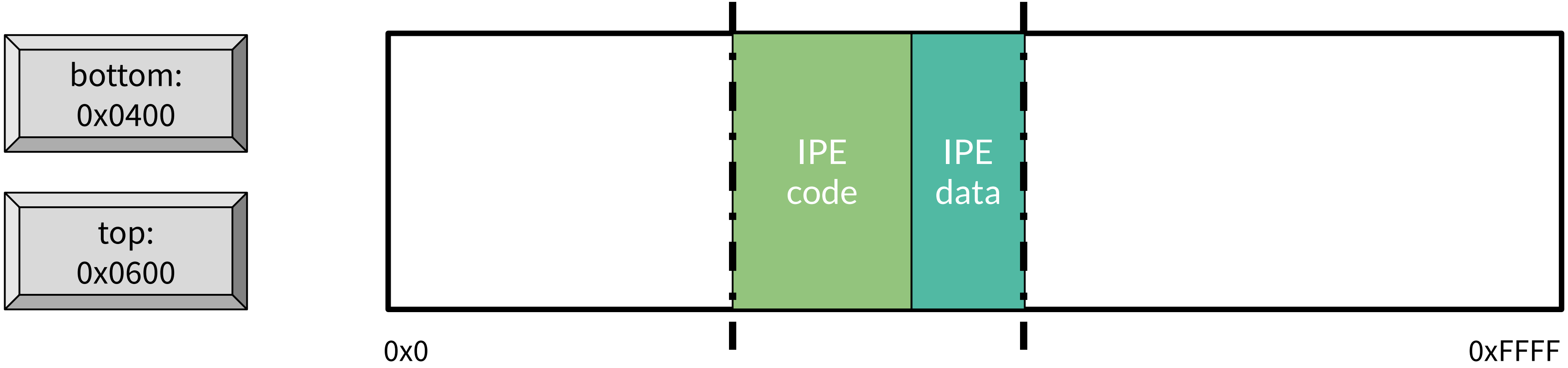
Intellectual Property Encapsulation



Intellectual Property Encapsulation



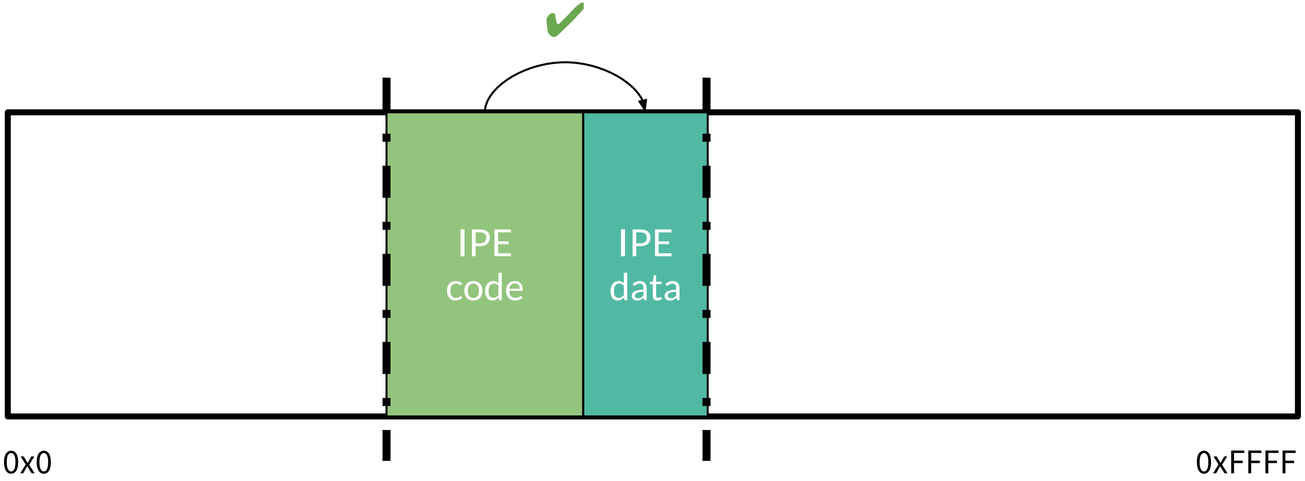
Intellectual Property Encapsulation



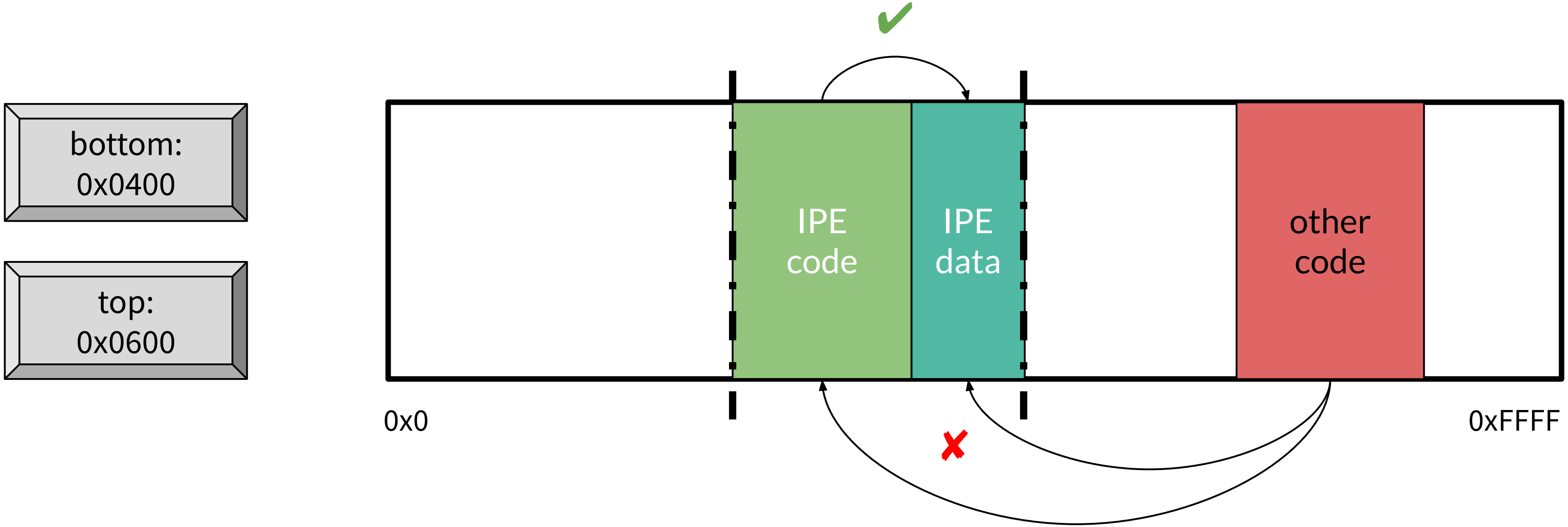
Intellectual Property Encapsulation

bottom:
0x0400

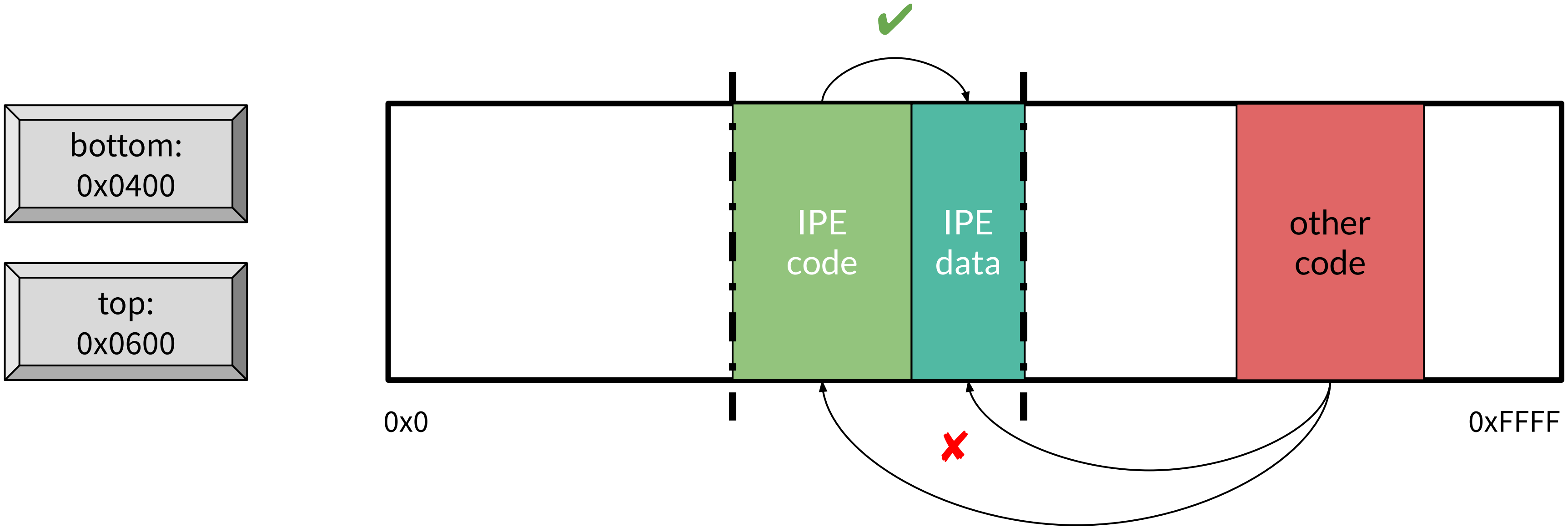
top:
0x0600



Intellectual Property Encapsulation



Intellectual Property Encapsulation

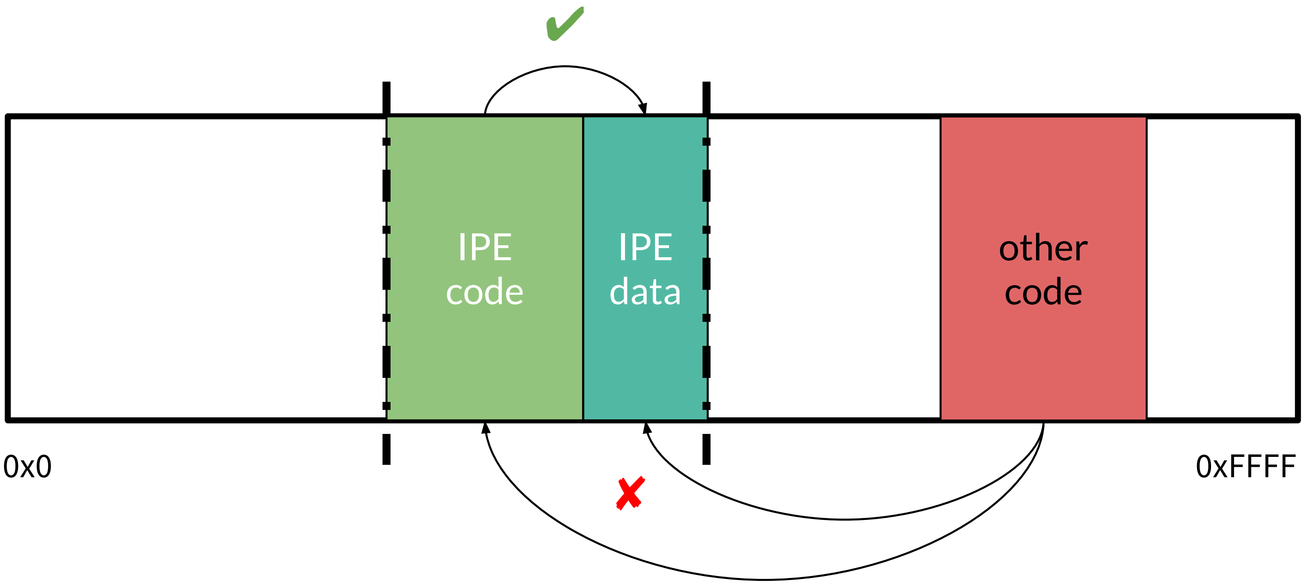


+ protection from JTAG debug port, direct memory access (DMA)

Intellectual Property Encapsulation

bottom:
0x0400

top:
0x0600

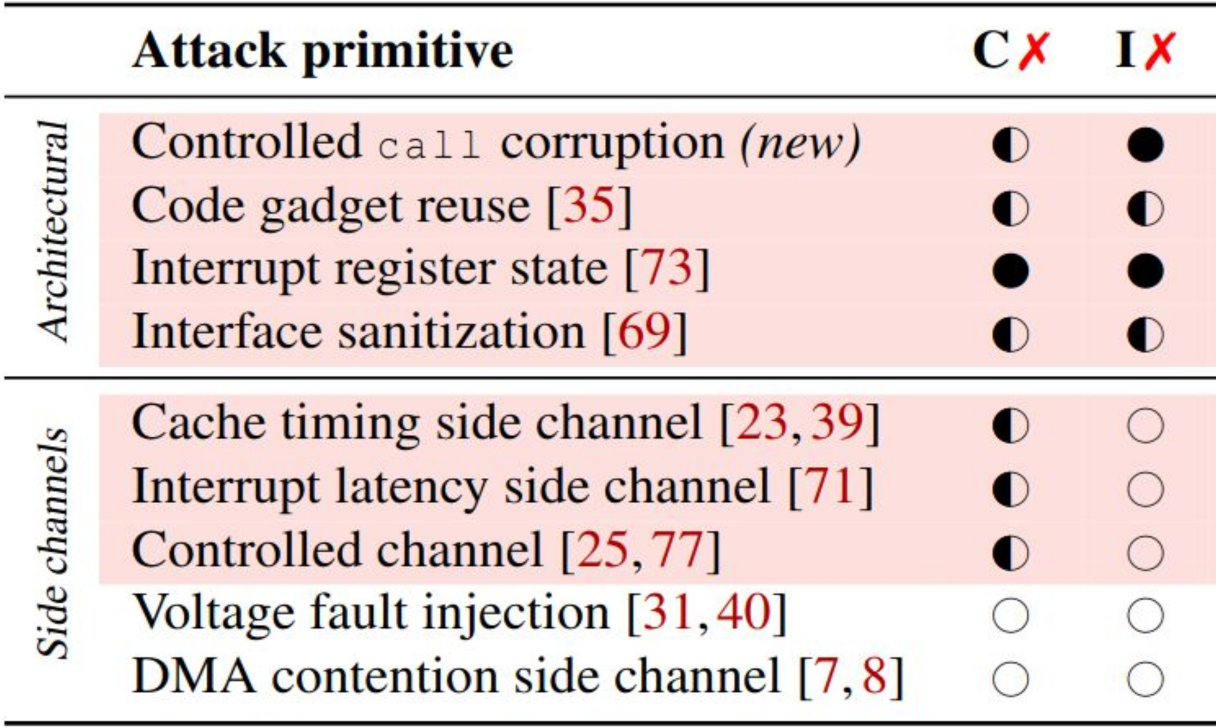


+ protection from JTAG debug port, direct memory access (DMA)

→ Program-counter-based access control

Part II: Insecurity

IPE attack primitives – a familiar story



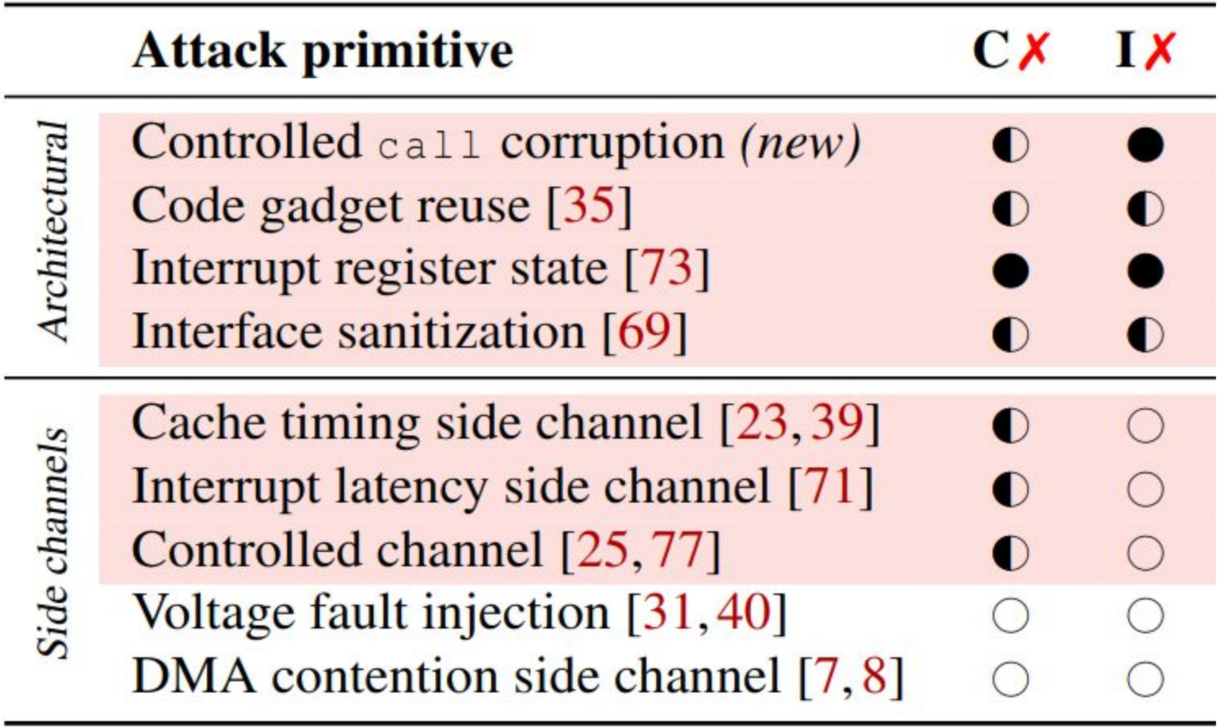
	Attack primitive	C ✗	I ✗
Architectural	Controlled `call` corruption *(new)*	◐	●
	Code gadget reuse [35]	◐	◐
	Interrupt register state [73]	●	●
	Interface sanitization [69]	◐	◐
Side channels	Cache timing side channel [23,39]	◐	○
	Interrupt latency side channel [71]	◐	○
	Controlled channel [25,77]	◐	○
	Voltage fault injection [31,40]	○	○
	DMA contention side channel [7,8]	○	○

Breaking confidentiality (C ✗) and integrity (I ✗) of code or data indirectly (◐) or directly (●).
Tested on multiple different MSP430 CPUs.

🗋 Bognar et al. "Intellectual Property Exposure: Subverting and Securing Intellectual Property Encapsulation in Texas Instruments Microcontrollers", USENIX 2024.

IPE attack primitives – a familiar story

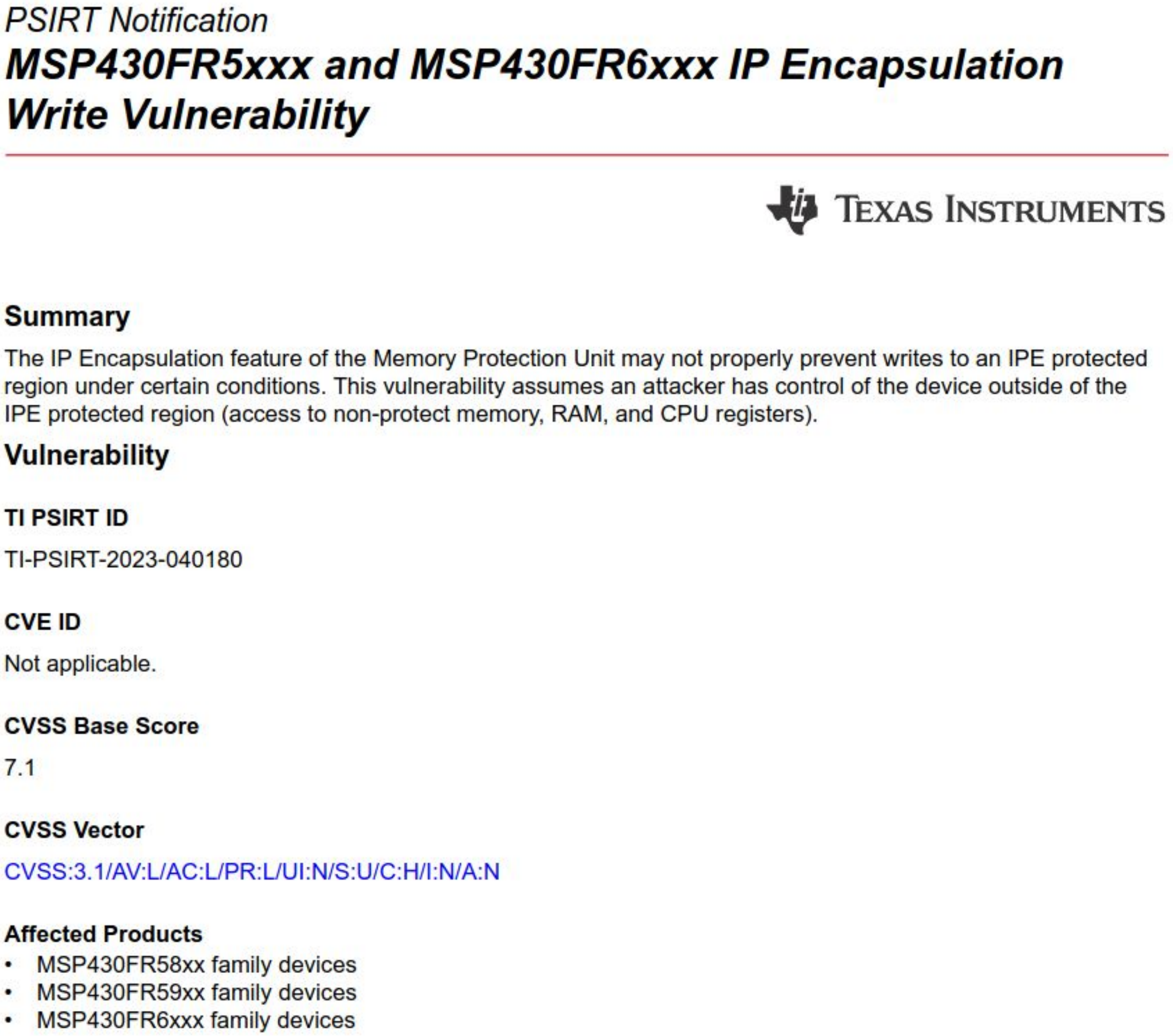
Software-based



	Attack primitive	C ✗	I ✗
Architectural	Controlled `call` corruption *(new)*	◐	●
	Code gadget reuse [35]	◐	◐
	Interrupt register state [73]	●	●
	Interface sanitization [69]	◐	◐
Side channels	Cache timing side channel [23, 39]	◐	○
	Interrupt latency side channel [71]	◐	○
	Controlled channel [25, 77]	◐	○
	Voltage fault injection [31, 40]	○	○
	DMA contention side channel [7, 8]	○	○

Breaking confidentiality (C ✗) and integrity (I ✗) of code or data indirectly (◐) or directly (●).
Tested on multiple different MSP430 CPUs.

Bognar et al. "Intellectual Property Exposure: Subverting and Securing Intellectual Property Encapsulation in Texas Instruments Microcontrollers", USENIX 2024.



PSIRT Notification

MSP430FR5xxx and MSP430FR6xxx IP Encapsulation Write Vulnerability

TEXAS INSTRUMENTS

Summary

The IP Encapsulation feature of the Memory Protection Unit may not properly prevent writes to an IPE protected region under certain conditions. This vulnerability assumes an attacker has control of the device outside of the IPE protected region (access to non-protect memory, RAM, and CPU registers).

Vulnerability

TI PSIRT ID

TI-PSIRT-2023-040180

CVE ID

Not applicable.

CVSS Base Score

7.1

CVSS Vector

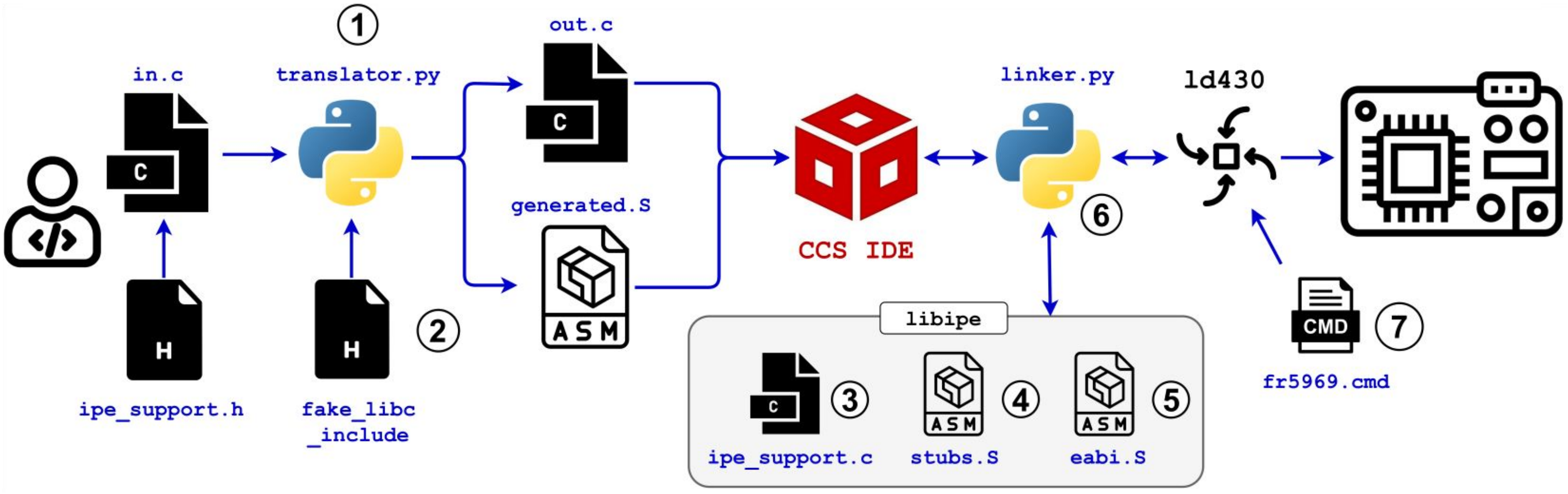
CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N

Affected Products

- MSP430FR58xx family devices
- MSP430FR59xx family devices
- MSP430FR6xxx family devices

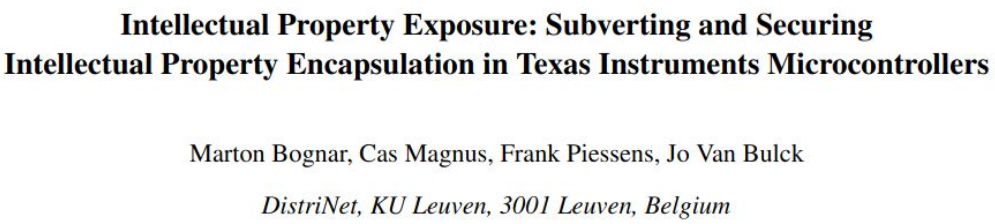
Part III: The ugly

Software mitigation: MPU to the rescue



- Re-purpose MPU to prevent architectural leakage
- Weaker attacker model → trust reset handler + JTAG

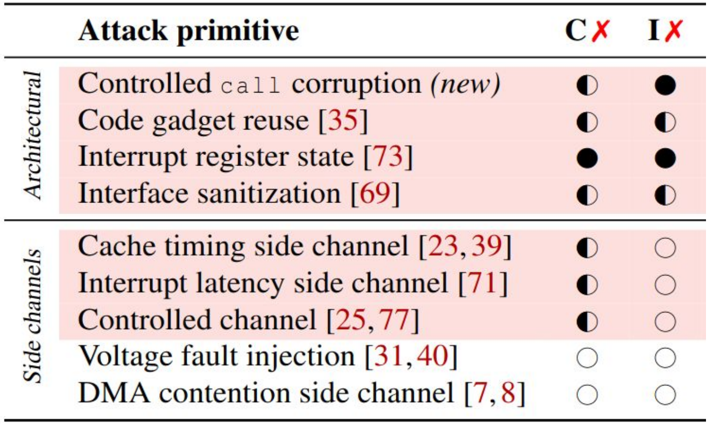
Code, paper: attacks and defenses on TI IPE



Intellectual Property Exposure: Subverting and Securing Intellectual Property Encapsulation in Texas Instruments Microcontrollers

Marton Bognar, Cas Magnus, Frank Piessens, Jo Van Bulck

DistriNet, KU Leuven, 3001 Leuven, Belgium

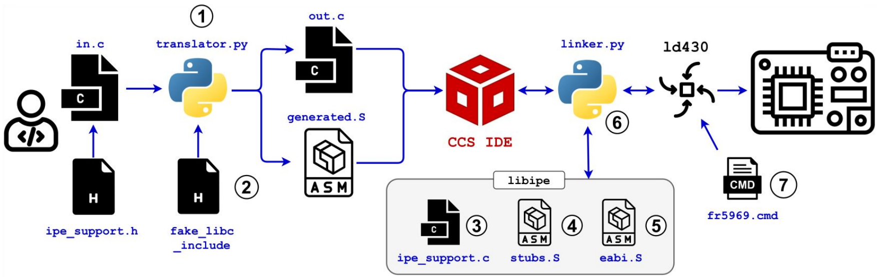


	Attack primitive	C ✗	I ✗
Architectural	Controlled `call` corruption *(new)*	◐	●
	Code gadget reuse [35]	◐	◐
	Interrupt register state [73]	●	●
	Interface sanitization [69]	◐	◐
Side channels	Cache timing side channel [23, 39]	◐	○
	Interrupt latency side channel [71]	◐	○
	Controlled channel [25, 77]	◐	○
	Voltage fault injection [31, 40]	○	○
	DMA contention side channel [7, 8]	○	○

ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

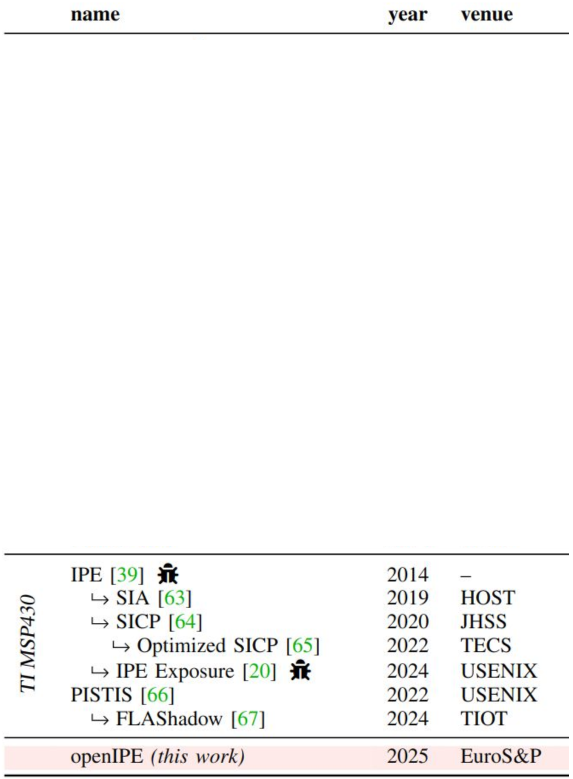


https://github.com/martonbognar/ipe-exposure

Part IV: The clean design

Research trends

- **TI MSP430** difficult to do research on:
  - Closed-source hardware and firmware
  - No white-box simulator

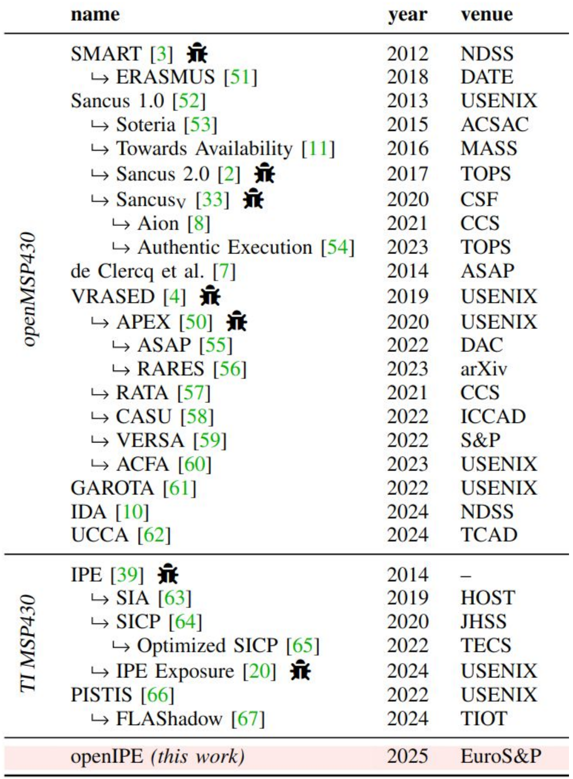


	name	year	venue
TI MSP430	IPE [39] 🐞	2014	–
	↪ SIA [63]	2019	HOST
	↪ SICP [64]	2020	JHSS
	↪ Optimized SICP [65]	2022	TECS
	↪ IPE Exposure [20] 🐞	2024	USENIX
	PISTIS [66]	2022	USENIX
	↪ FLAShadow [67]	2024	TIOT
	openIPE *(this work)*	2025	EuroS&P

Research trends

- **TI MSP430** difficult to do research on:
  - Closed-source hardware and firmware
  - No white-box simulator

- **openMSP430:** popular in research
  - Many systems (re-)implement isolation features
  - No compatibility with each other or industry standards
  - Limited applicability to real-world devices

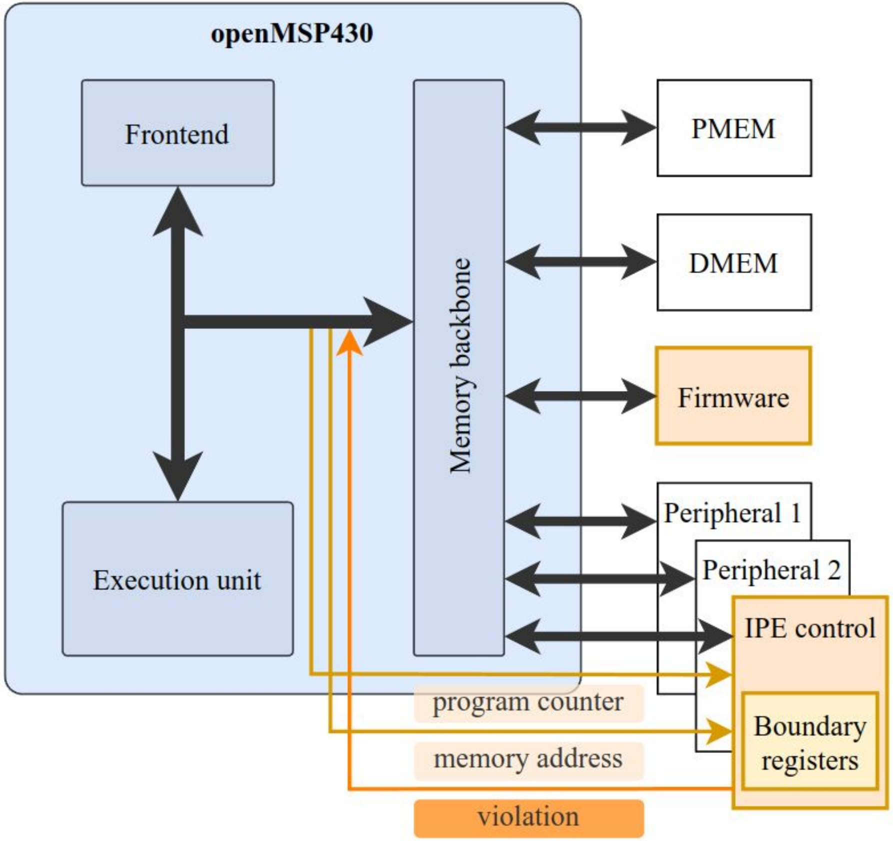


	name	year	venue
openMSP430	SMART [3] 🐞	2012	NDSS
	↪ ERASMUS [51]	2018	DATE
	Sancus 1.0 [52]	2013	USENIX
	↪ Soteria [53]	2015	ACSAC
	↪ Towards Availability [11]	2016	MASS
	↪ Sancus 2.0 [2] 🐞	2017	TOPS
	↪ Sancus_V [33] 🐞	2020	CSF
	↪ Aion [8]	2021	CCS
	↪ Authentic Execution [54]	2023	TOPS
	de Clercq et al. [7]	2014	ASAP
	VRASED [4] 🐞	2019	USENIX
	↪ APEX [50] 🐞	2020	USENIX
	↪ ASAP [55]	2022	DAC
	↪ RARES [56]	2023	arXiv
	↪ RATA [57]	2021	CCS
	↪ CASU [58]	2022	ICCAD
	↪ VERSA [59]	2022	S&P
	↪ ACFA [60]	2023	USENIX
	GAROTA [61]	2022	USENIX
	IDA [10]	2024	NDSS
	UCCA [62]	2024	TCAD
TI MSP430	IPE [39] 🐞	2014	–
	↪ SIA [63]	2019	HOST
	↪ SICP [64]	2020	JHSS
	↪ Optimized SICP [65]	2022	TECS
	↪ IPE Exposure [20] 🐞	2024	USENIX
	PISTIS [66]	2022	USENIX
	↪ FLAShadow [67]	2024	TIOT
	openIPE *(this work)*	2025	EuroS&P

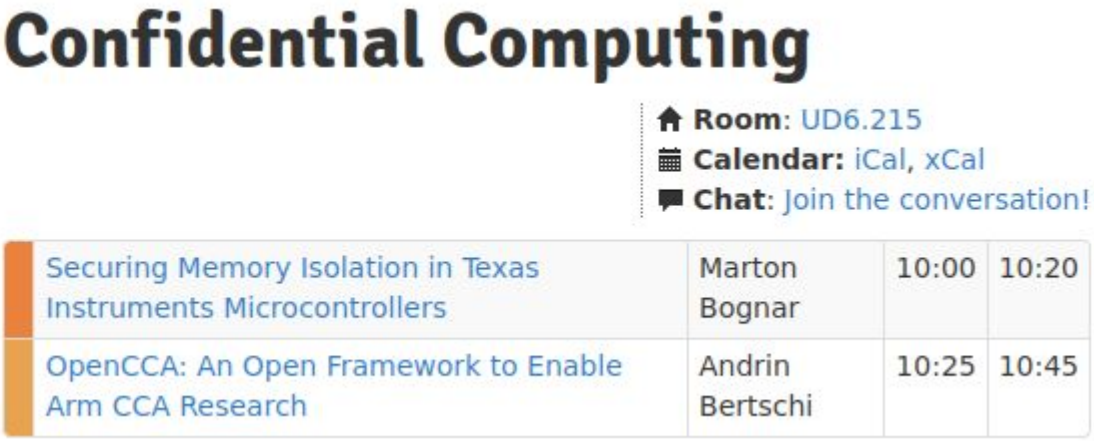
Our proposal: openIPE

- **Flexible isolation** primitive
  - Based on the IPE specification
  - With protected firmware
  - But freely configurable!
- Includes proposed **hardware fixes** for IPE

openIPE



Our proposal: openIPE



Confidential Computing

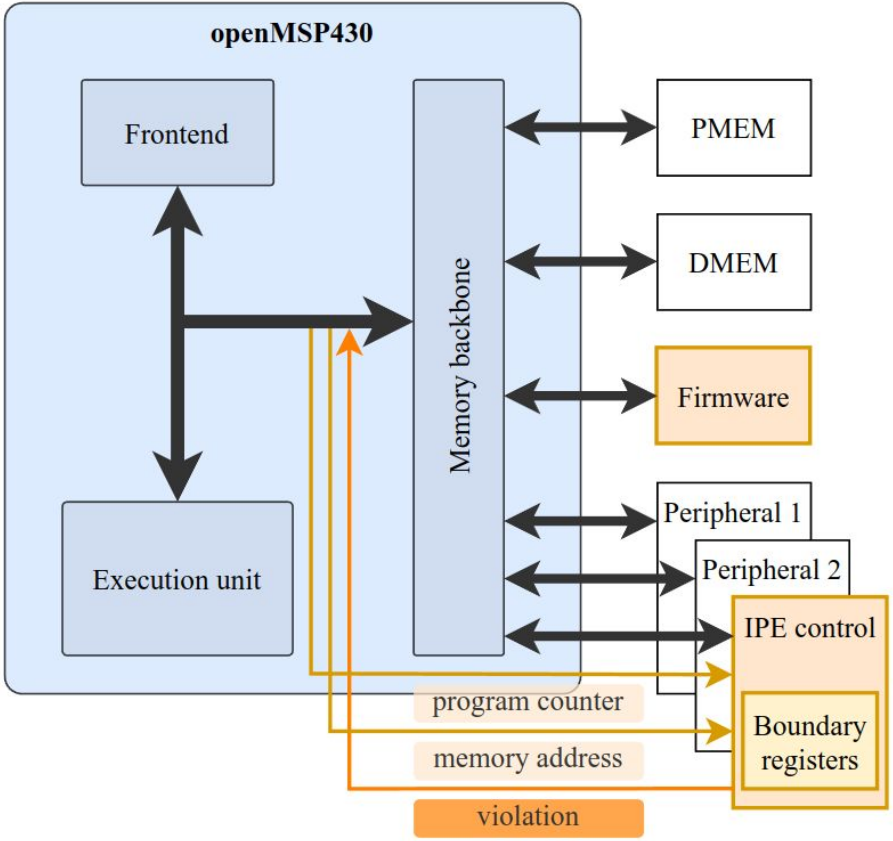
- **Room**: UD6.215
- **Calendar:** iCal, xCal
- **Chat**: Join the conversation!

Securing Memory Isolation in Texas Instruments Microcontrollers	Marton Bognar	10:00	10:20
OpenCCA: An Open Framework to Enable Arm CCA Research	Andrin Bertschi	10:25	10:45

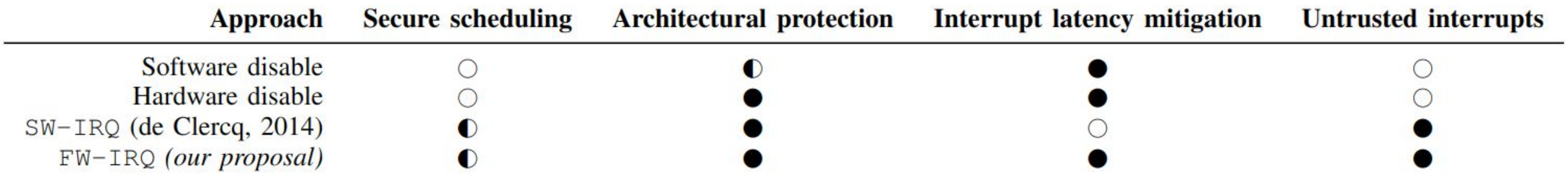
Our proposal: openIPE

- **Flexible isolation** primitive
  - Based on the IPE specification
  - With protected firmware
  - But freely configurable!
- Includes proposed **hardware fixes** for IPE

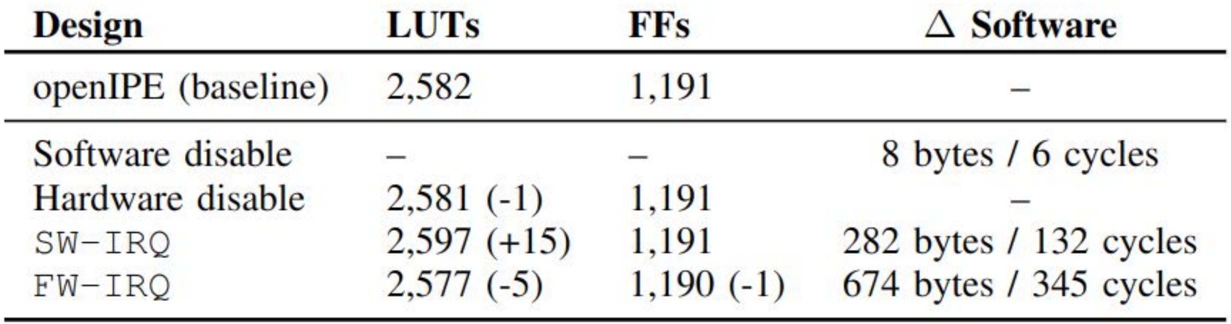
openIPE



Case study: Secure interrupt handling



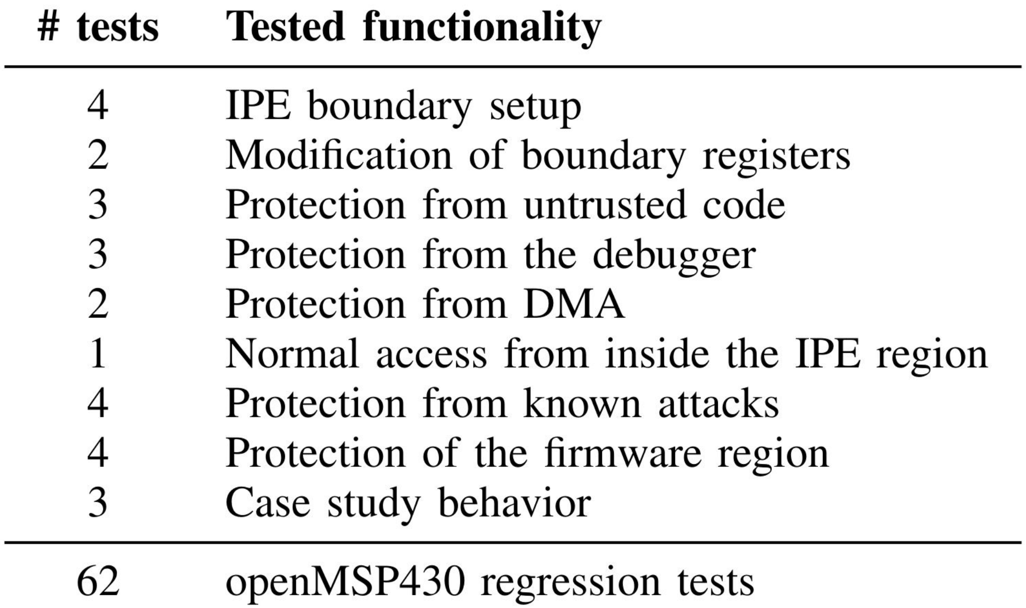
Approach	Secure scheduling	Architectural protection	Interrupt latency mitigation	Untrusted interrupts
Software disable	○	◐	●	○
Hardware disable	○	●	●	○
`SW-IRQ` (de Clercq, 2014)	◐	●	○	●
`FW-IRQ` *(our proposal)*	◐	●	●	●



Design	LUTs	FFs	Δ Software
openIPE (baseline)	2,582	1,191	–
Software disable	–	–	8 bytes / 6 cycles
Hardware disable	2,581 (-1)	1,191	–
`SW-IRQ`	2,597 (+15)	1,191	282 bytes / 132 cycles
`FW-IRQ`	2,577 (-5)	1,190 (-1)	674 bytes / 345 cycles

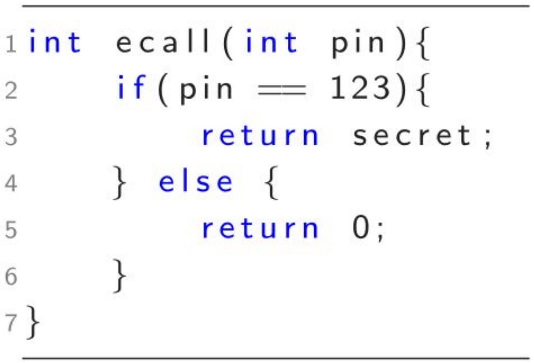
Hardware security validation: Unit tests

- Functional and security tests
- Backwards compatibility for (future) extensions



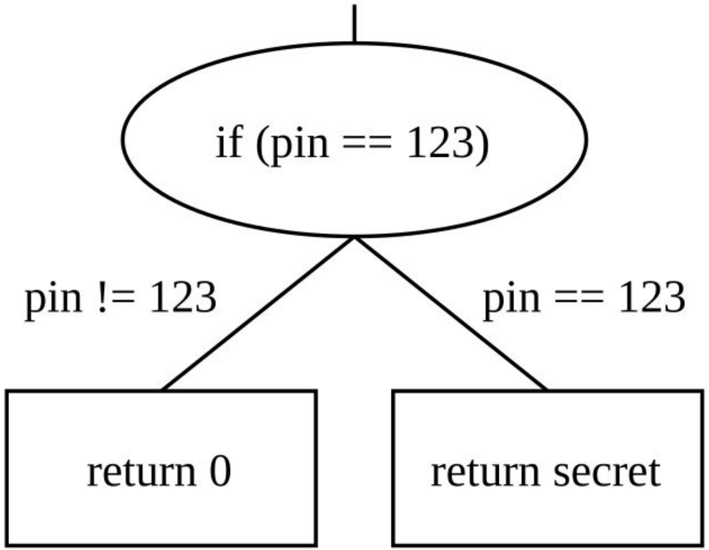
# tests	Tested functionality
4	IPE boundary setup
2	Modification of boundary registers
3	Protection from untrusted code
3	Protection from the debugger
2	Protection from DMA
1	Normal access from inside the IPE region
4	Protection from known attacks
4	Protection of the firmware region
3	Case study behavior
62	openMSP430 regression tests

Software security validation: Symbolic execution



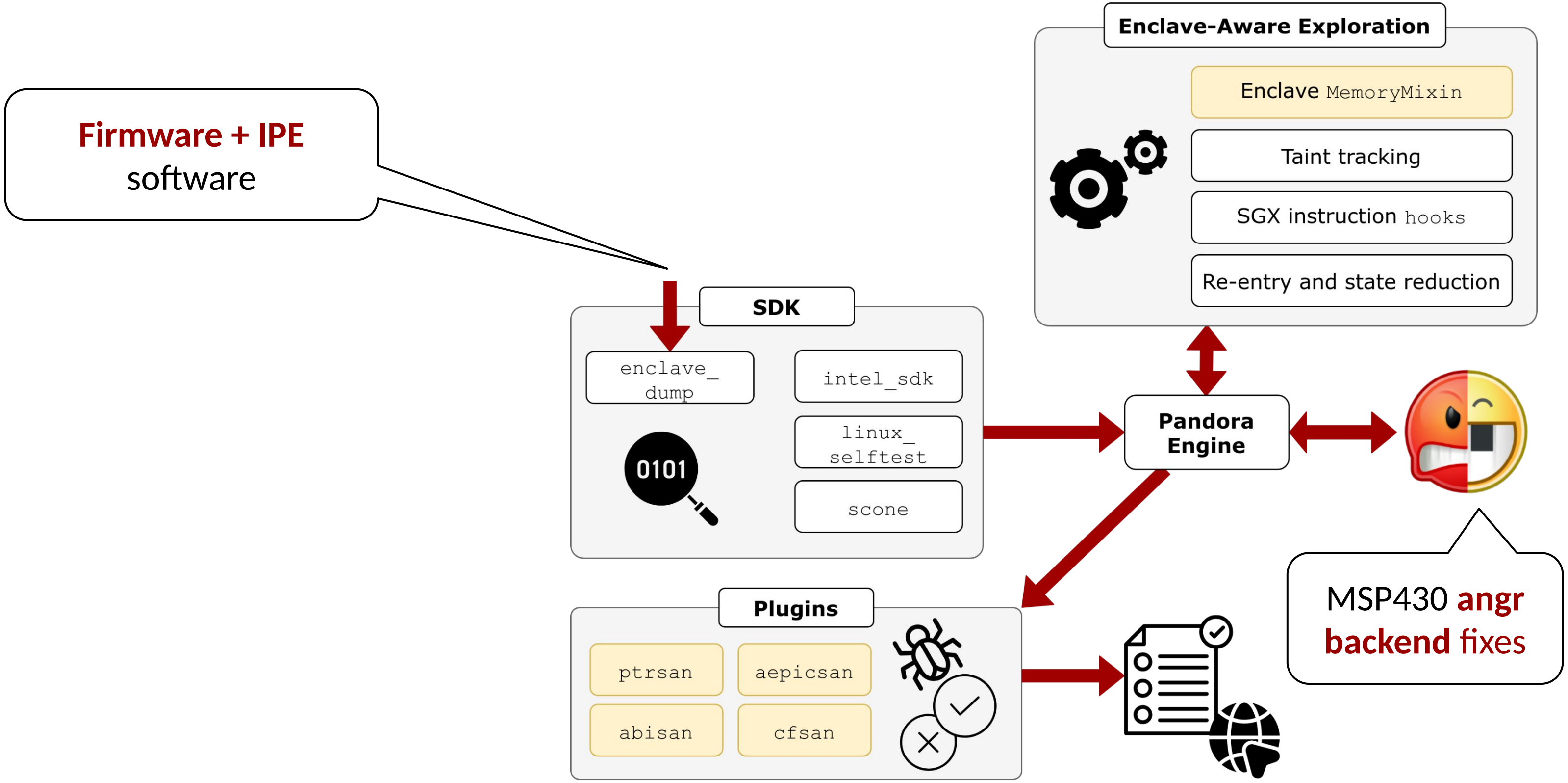
```
int ecall(int pin){
    if(pin == 123){
        return secret;
    } else {
        return 0;
    }
}
```

https://angr.io/



- Symbolic execution uses a constraint solver
- Execution works on instruction-level, i.e., as close to the binary as possible

Principled symbolic IPE enclave validation



Firmware + IPE software

Enclave-Aware Exploration

- Enclave `MemoryMixin`
- Taint tracking
- SGX instruction `hooks`
- Re-entry and state reduction

SDK

- `enclave_dump`
- `intel_sdk`
- `linux_selftest`
- `scone`

0101

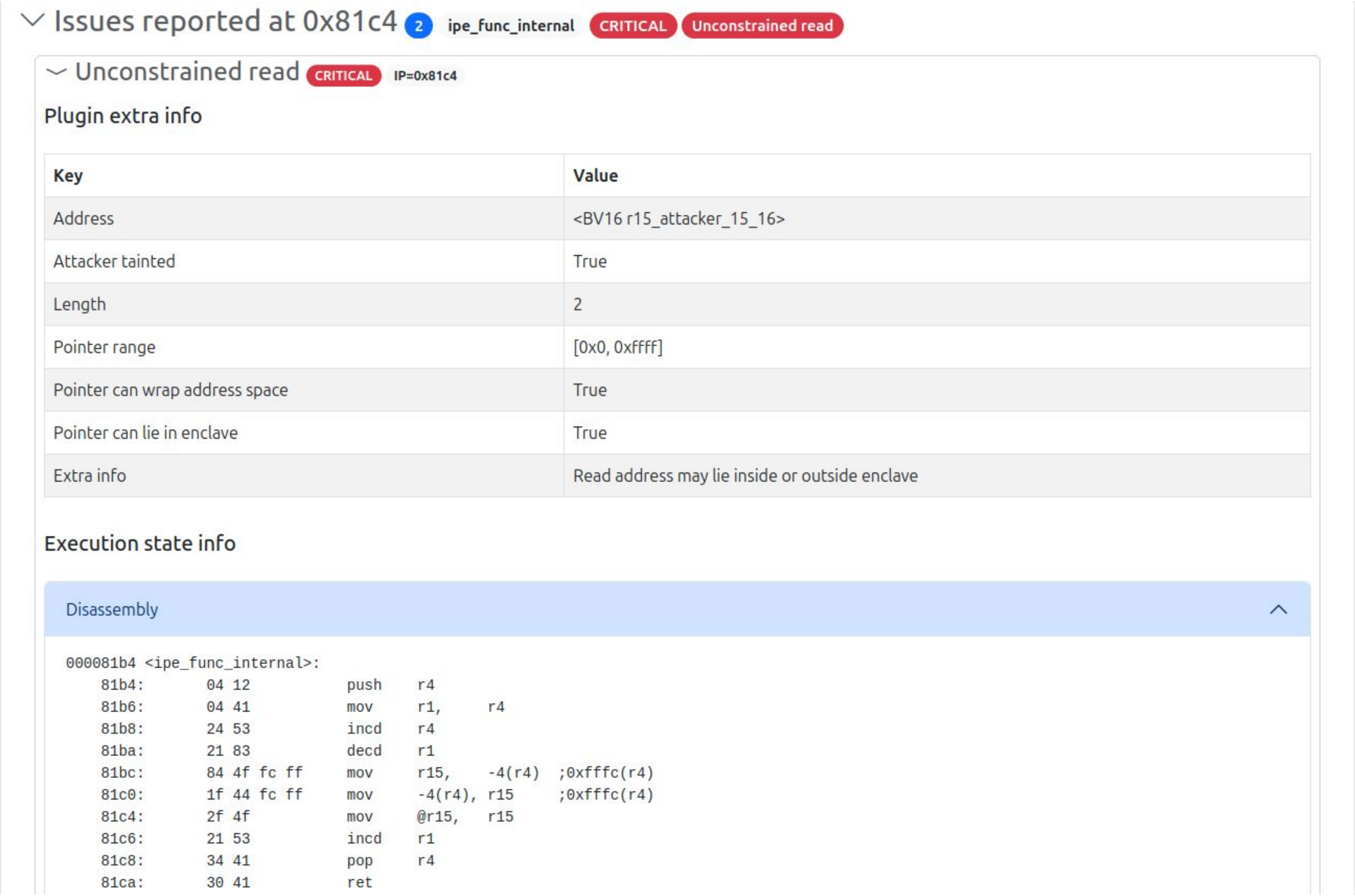
Pandora Engine

Plugins

- `ptrsan`
- `aepicsan`
- `abisan`
- `cfsan`

MSP430 **angr backend** fixes

Alder et al. “Pandora: Principled Symbolic Validation of Intel SGX Enclave Runtimes”, S&P 2024.



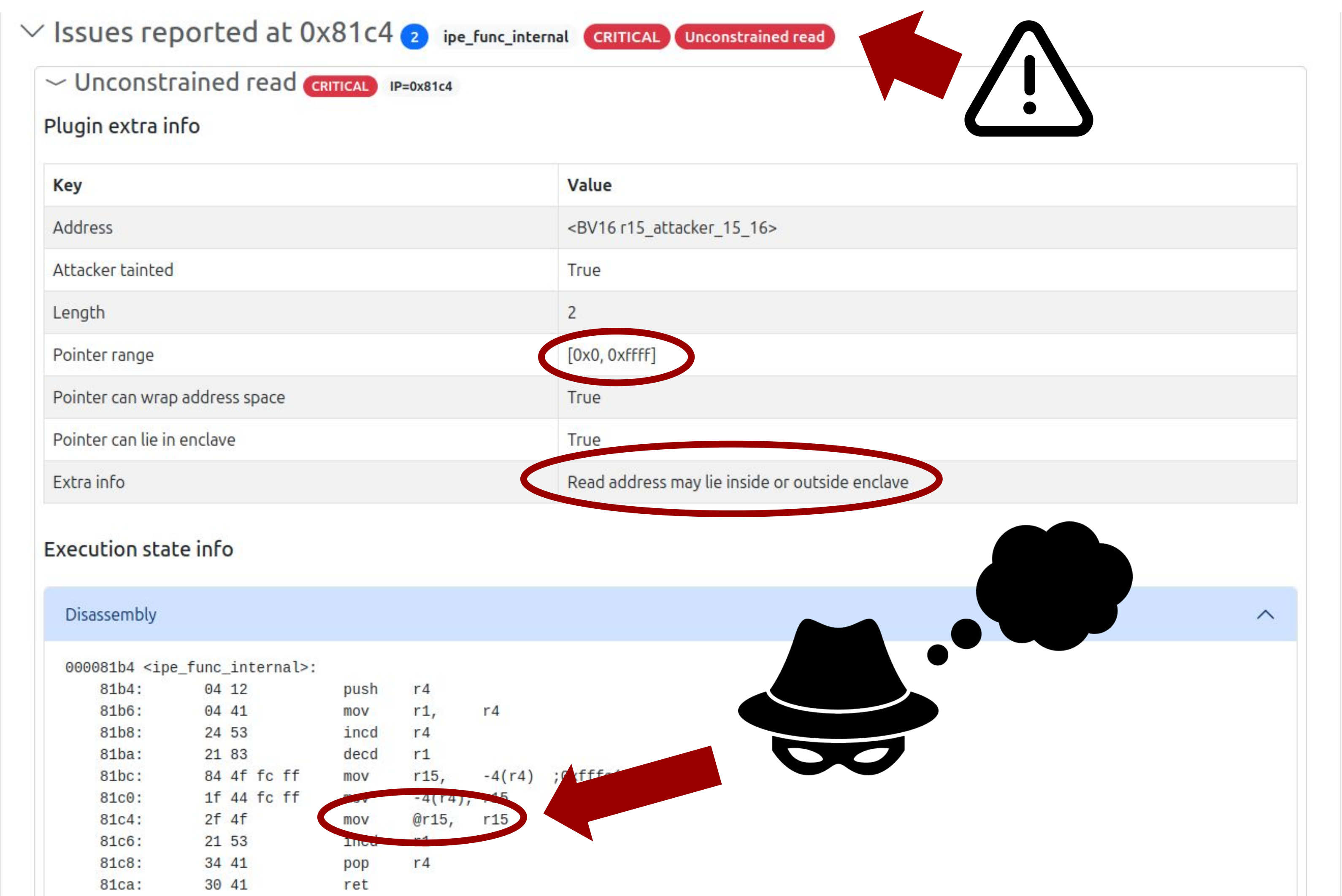
Unconstrained read

Plugin extra info

Key	Value
Address	<BV16 r15_attacker_15_16>
Attacker tainted	True
Length	2
Pointer range	[0x0, 0xffff]
Pointer can wrap address space	True
Pointer can lie in enclave	True
Extra info	Read address may lie inside or outside enclave

Execution state info

```
000081b4 <ipe_func_internal>:
    81b4:       04 12           push    r4
    81b6:       04 41           mov     r1,     r4
    81b8:       24 53           incd    r4
    81ba:       21 83           decd    r1
    81bc:       84 4f fc ff     mov     r15,    -4(r4)  ;0xfffc(r4)
    81c0:       1f 44 fc ff     mov     -4(r4), r15     ;0xfffc(r4)
    81c4:       2f 4f           mov     @r15,   r15
    81c6:       21 53           incd    r1
    81c8:       34 41           pop     r4
    81ca:       30 41           ret
```



Issues reported at 0x81c4

Unconstrained read

Plugin extra info

Key	Value
Address	<BV16 r15_attacker_15_16>
Attacker tainted	True
Length	2
Pointer range	[0x0, 0xffff]
Pointer can wrap address space	True
Pointer can lie in enclave	True
Extra info	Read address may lie inside or outside enclave

Execution state info

Disassembly

```
000081b4 <ipe_func_internal>:
    81b4:       04 12           push    r4
    81b6:       04 41           mov     r1,     r4
    81b8:       24 53           incd    r4
    81ba:       21 83           decd    r1
    81bc:       84 4f fc ff     mov     r15,    -4(r4)  ;0xfffc(
    81c0:       1f 44 fc ff     mov     -4(r4), r15
    81c4:       2f 4f           mov     @r15,   r15
    81c6:       21 53           incd    r1
    81c8:       34 41           pop     r4
    81ca:       30 41           ret
```

Our goals with openIPE

- **Research**
  - Unified memory isolation implementation
  - Rapid prototyping of security features
  - Thorough testing

- **Teaching**
  - Relatively simple architecture
  - Wide range of concepts

Icons: flaticon.com

Conclusions and outlook

openIPE

- **IPE Exposure:** First security analysis of Texas Instruments IPE
  - Novel vulnerability: *controlled call corruption* + known primitives
  - Software-only mitigation via MPU
- **openIPE:** Open-source extensible memory isolation
  - Hardware + firmware + software co-design
  - Unit tests and symbolic execution

https://github.com/martonbognar/ipe-exposure https://github.com/martonbognar/openipe

Academic publications and contact: https://mici.hu