Principled Symbolic Validation of Enclaves on Low-End Microcontrollers

Gert-Jan Goossens, *Jo Van Bulck*

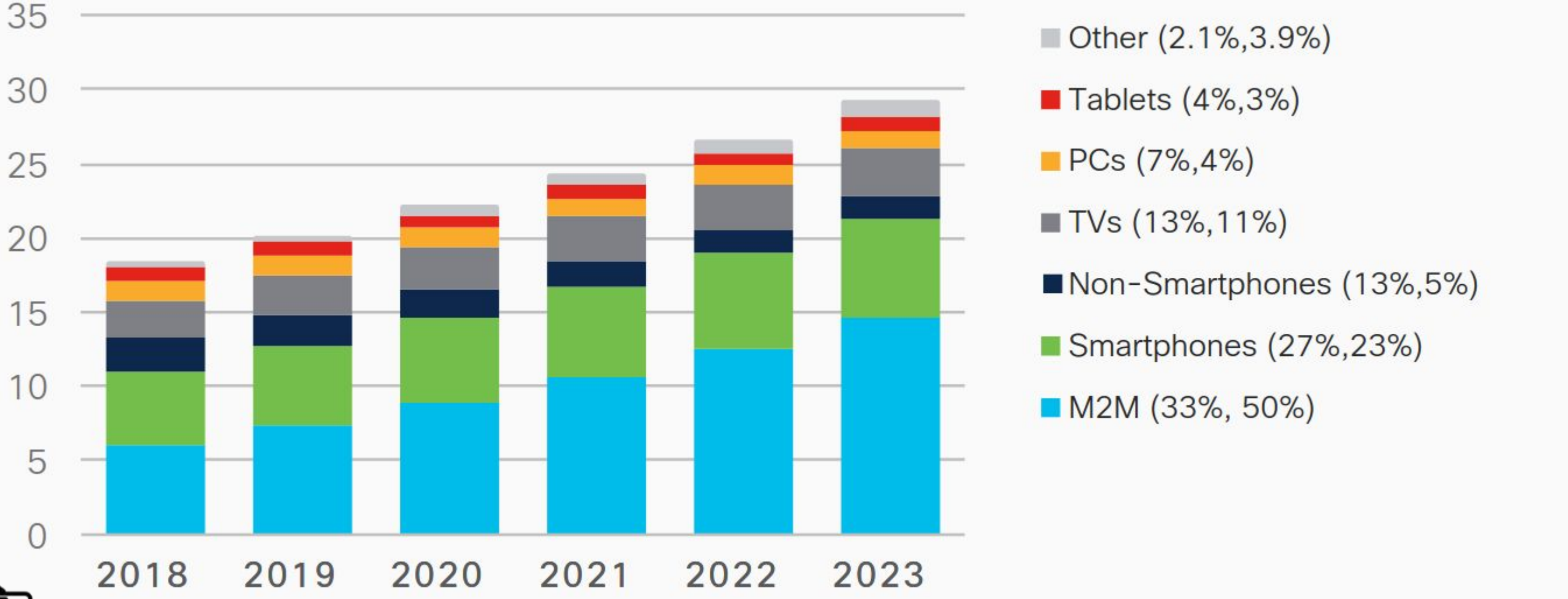
🏠 DistriNet, KU Leuven, Belgium ✉ jo.vanbulck@cs.kuleuven.be 🐦 @jovanbulck 🌐 vanbulck.net

8th Workshop on System Software for Trusted Execution (SysTEX) – July 2, 2025

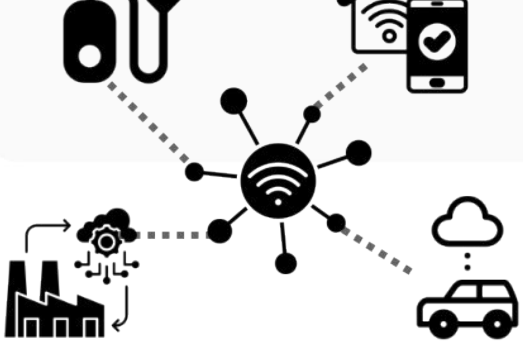
Context: Growth of the Internet of Things (IoT)

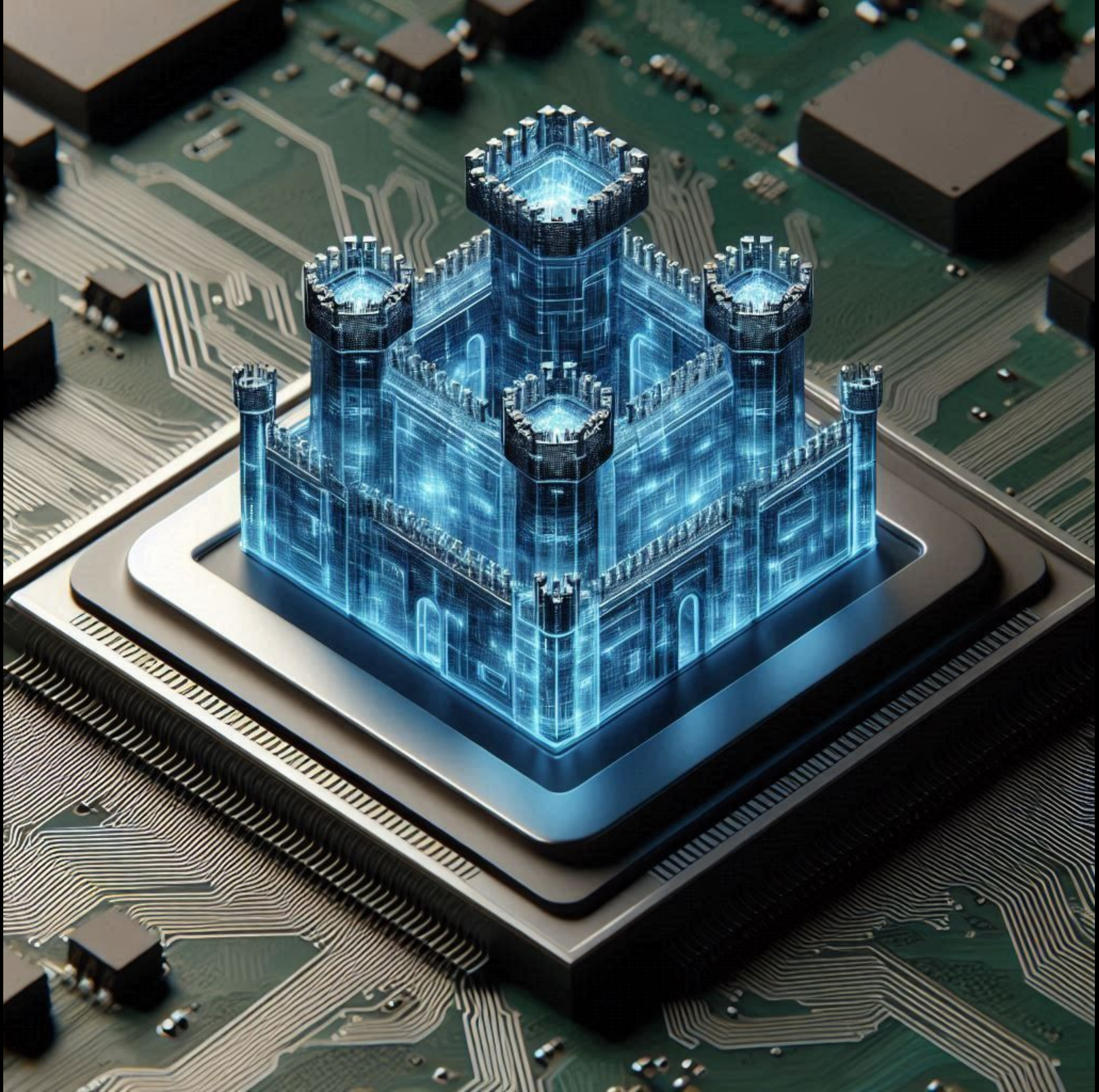
10% CAGR
2018–2023

Billions of Devices



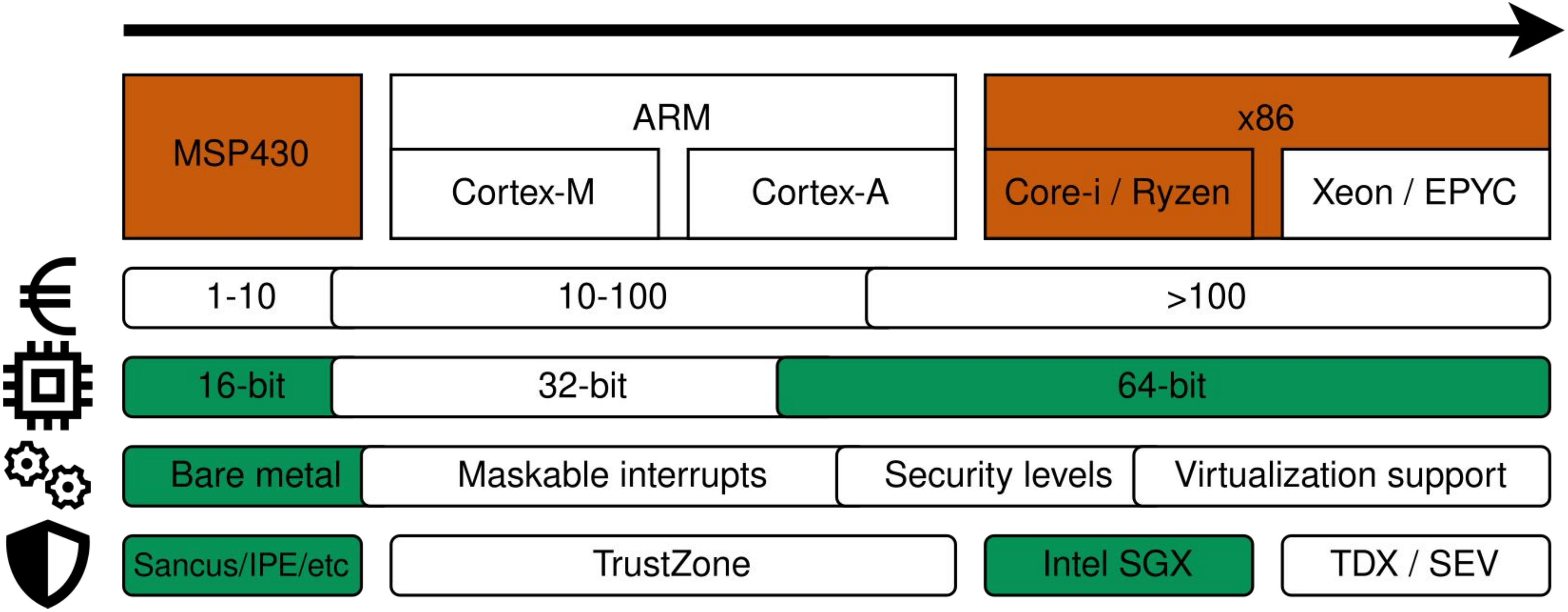
* Figures (n) refer to 2018, 2023 device share





DALL·E 3

TEE Computing Spectrum: “Low-End” vs. “High-End”



MSP430

ARM

Cortex-M

Cortex-A

x86

Core-i / Ryzen

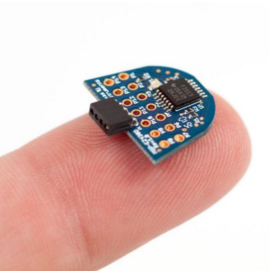
Xeon / EPYC

€ 1-10 | 10-100 | >100

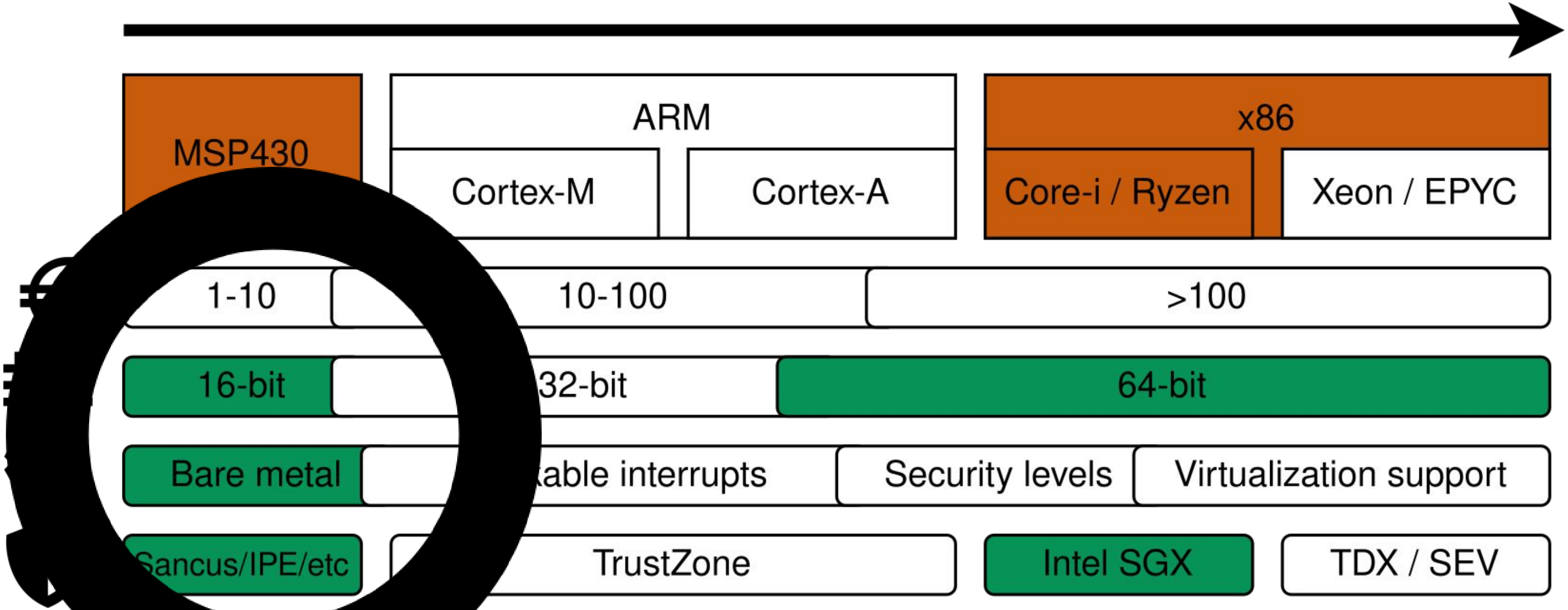
16-bit | 32-bit | 64-bit

Bare metal | Maskable interrupts | Security levels | Virtualization support

Sancus/IPE/etc | TrustZone | Intel SGX | TDX / SEV



TEE Computing Spectrum: “Low-End” vs. “High-End”



MSP430

ARM

Cortex-M

Cortex-A

x86

Core-i / Ryzen

Xeon / EPYC

1-10

10-100

>100

16-bit

32-bit

64-bit

Bare metal

able interrupts

Security levels

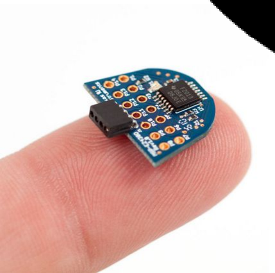
Virtualization support

Sancus/IPE/etc

TrustZone

Intel SGX

TDX / SEV



Heterogenous & underexplored!

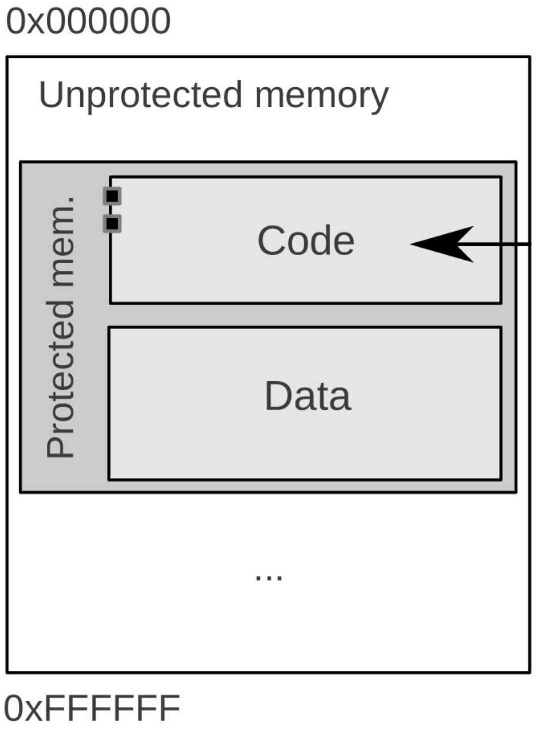
Sancus: Lightweight Trusted Computing for the IoT

Embedded enclaved execution:

- Isolation & attestation
- Save + clear CPU state on interrupt

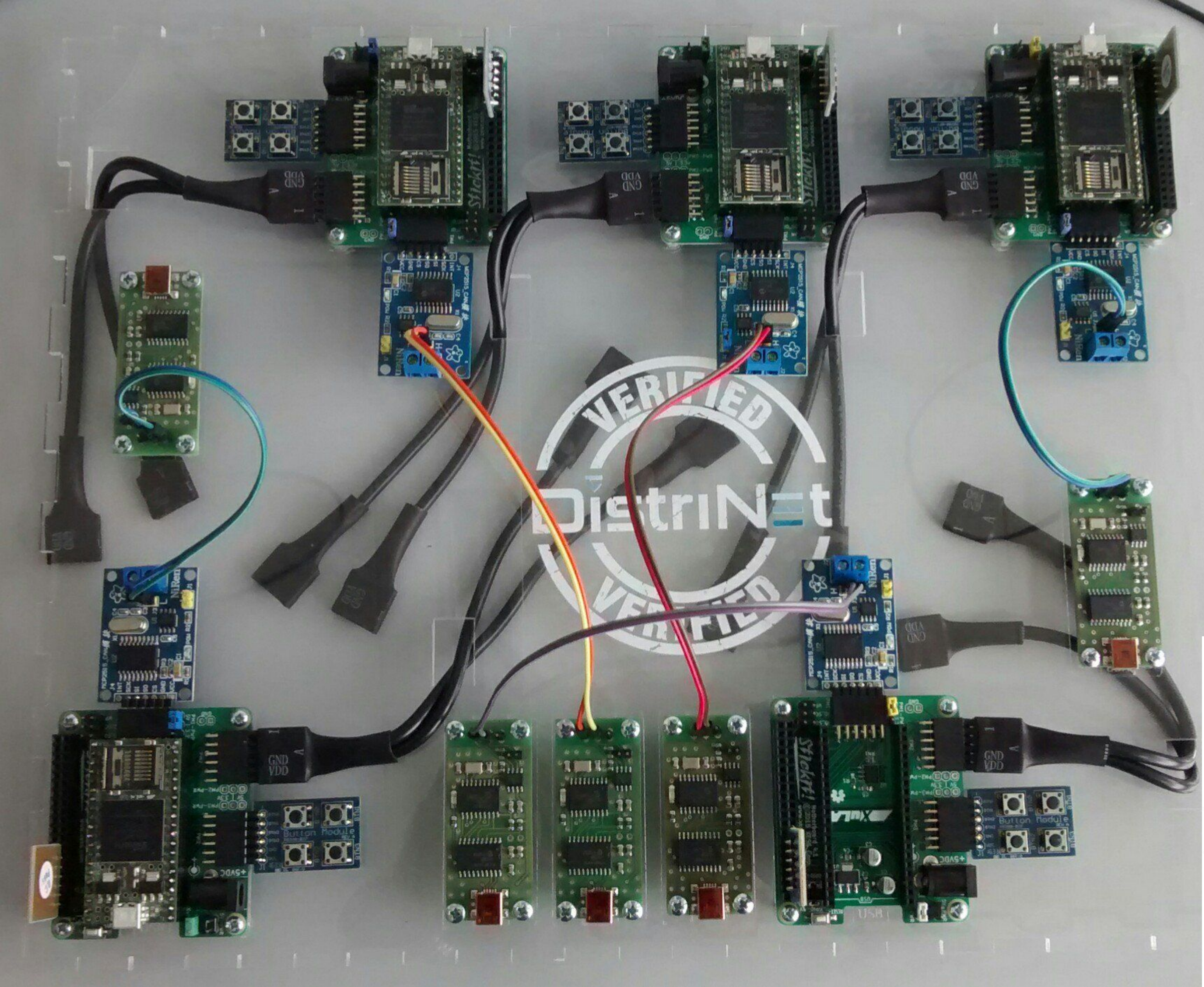
Small CPU (**16-bit openMSP430**):

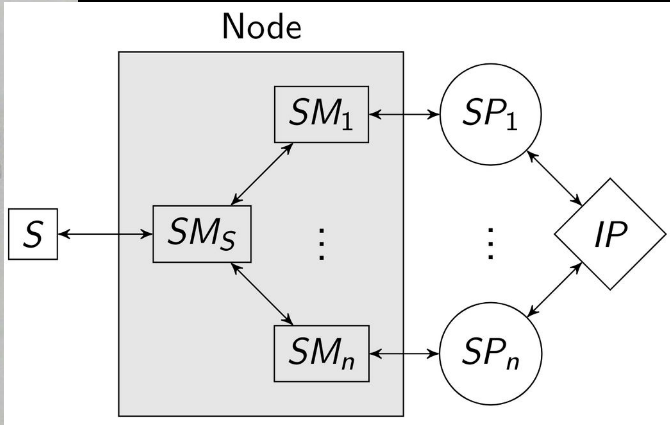
- Area: ≤ 2 kLUTs
- Deterministic execution: no pipeline/cache/MMU/…
- Research vehicle for rapid prototyping of attacks & mitigations

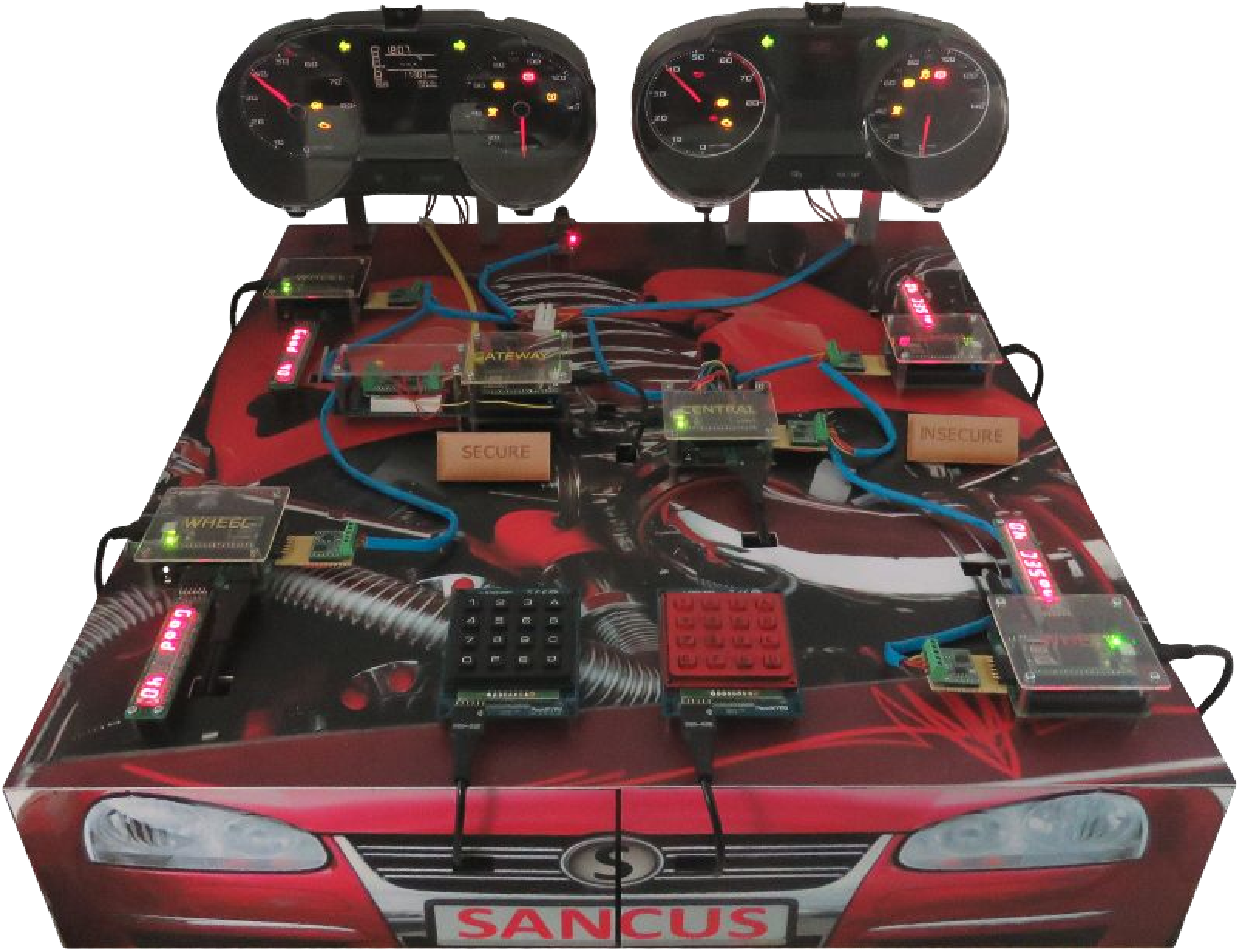


https://github.com/sancus-tee
https://downloads.distrinet-research.be/software/sancus

Noorman et al. Sancus 2.0: A Low-Cost Security Architecture for IoT devices. TOPS, 2017.

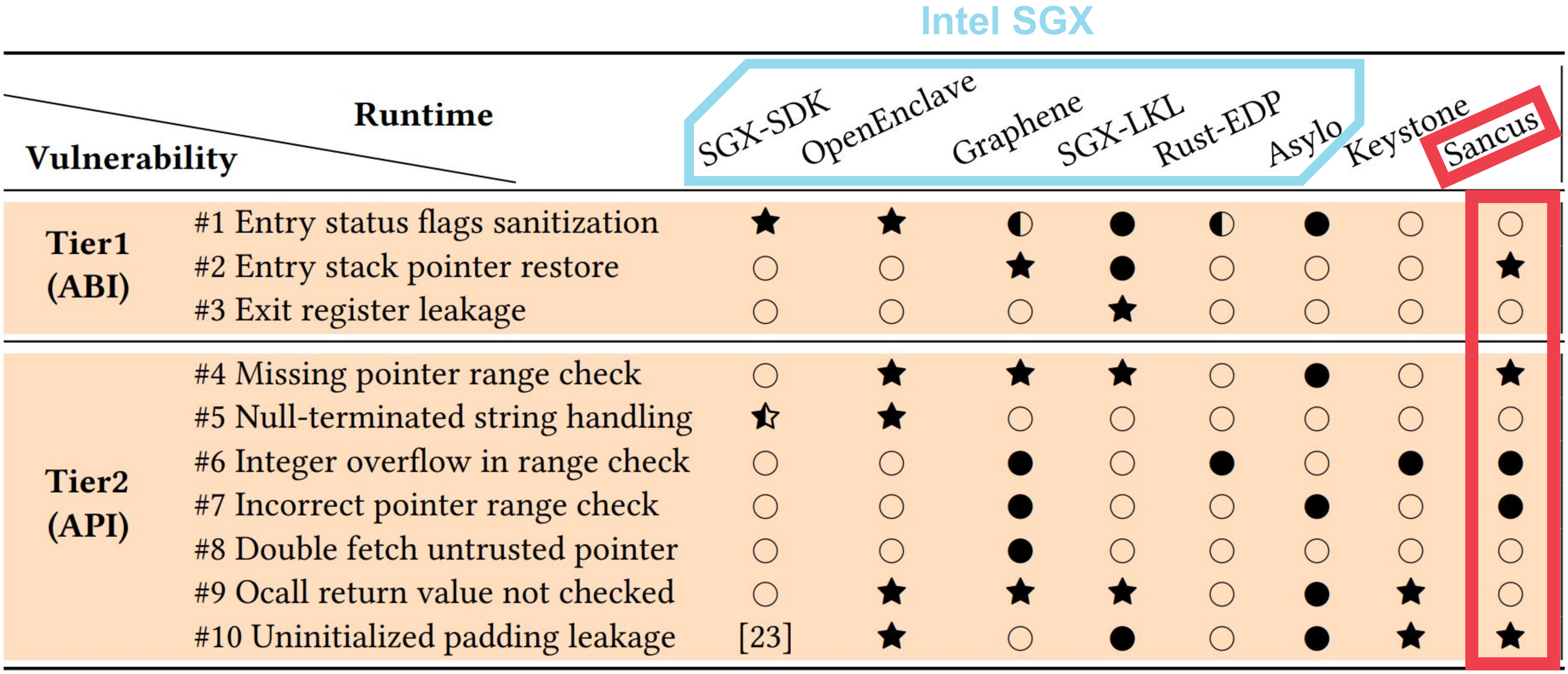

VERIFIED
DistriNet
VERIFIED
GND
VDD
Stickit!
Button Module


Node
SM_1
SP_1
S
SM_S
⋮
⋮
IP
SM_n
SP_n



Van Bulck et al., “VulCAN: Efficient Component Authentication and Software Isolation for Automotive Control Networks”, ACSAC 2017.

Challenge: Writing “Secure” Enclave Software is Hard...

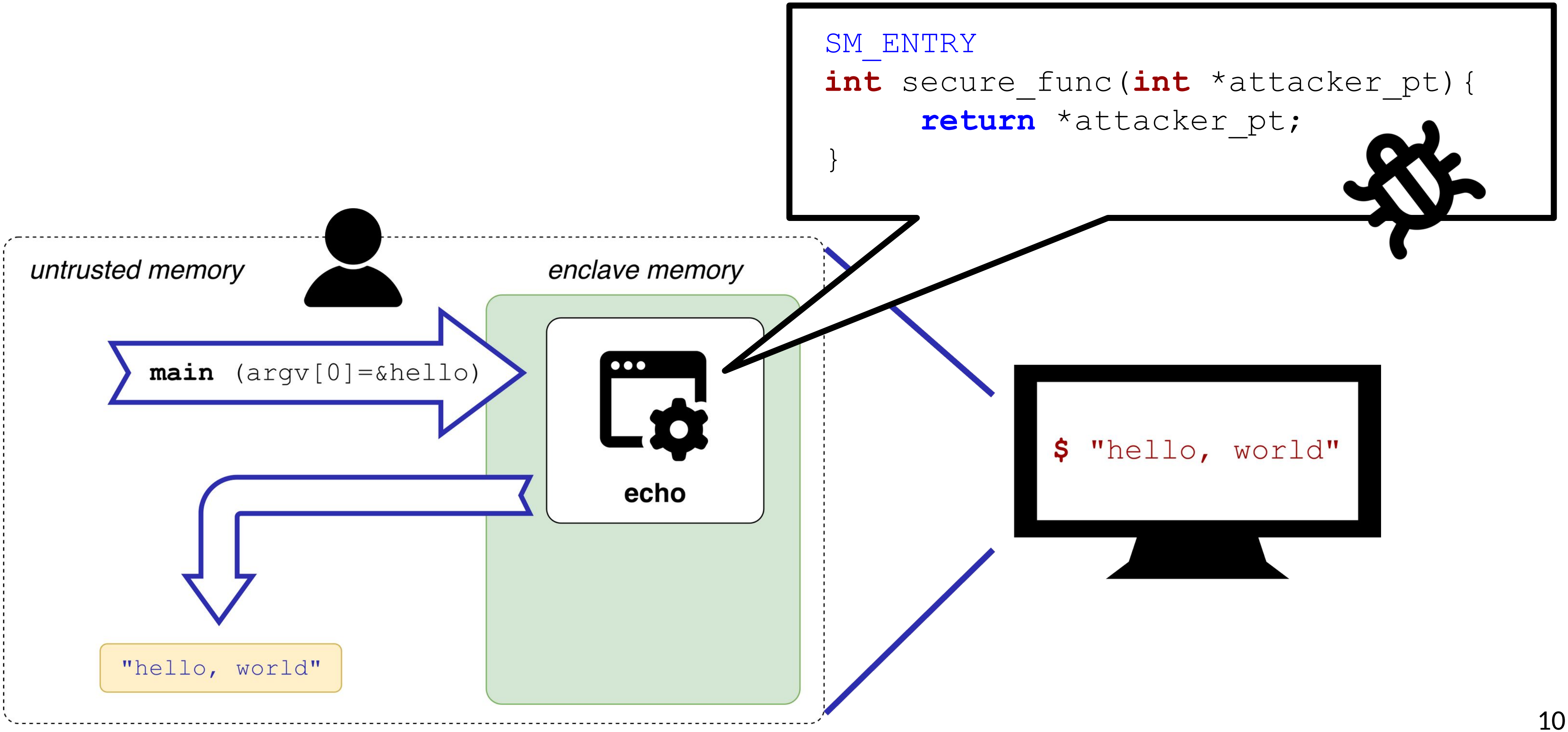


Intel SGX

Vulnerability	Runtime	SGX-SDK	OpenEnclave	Graphene	SGX-LKL	Rust-EDP	Asylo	Keystone	Sancus
Tier1 (ABI)	#1 Entry status flags sanitization	★	★	◐	●	◐	●	○	○
	#2 Entry stack pointer restore	○	○	★	●	○	○	○	★
	#3 Exit register leakage	○	○	○	★	○	○	○	○
Tier2 (API)	#4 Missing pointer range check	○	★	★	★	○	●	○	★
	#5 Null-terminated string handling	⯪	★	○	○	○	○	○	○
	#6 Integer overflow in range check	○	○	●	○	●	○	●	●
	#7 Incorrect pointer range check	○	○	●	○	○	●	○	●
	#8 Double fetch untrusted pointer	○	○	●	○	○	○	○	○
	#9 Ocall return value not checked	○	★	★	★	○	●	★	○
	#10 Uninitialized padding leakage	[23]	★	○	●	○	●	★	★

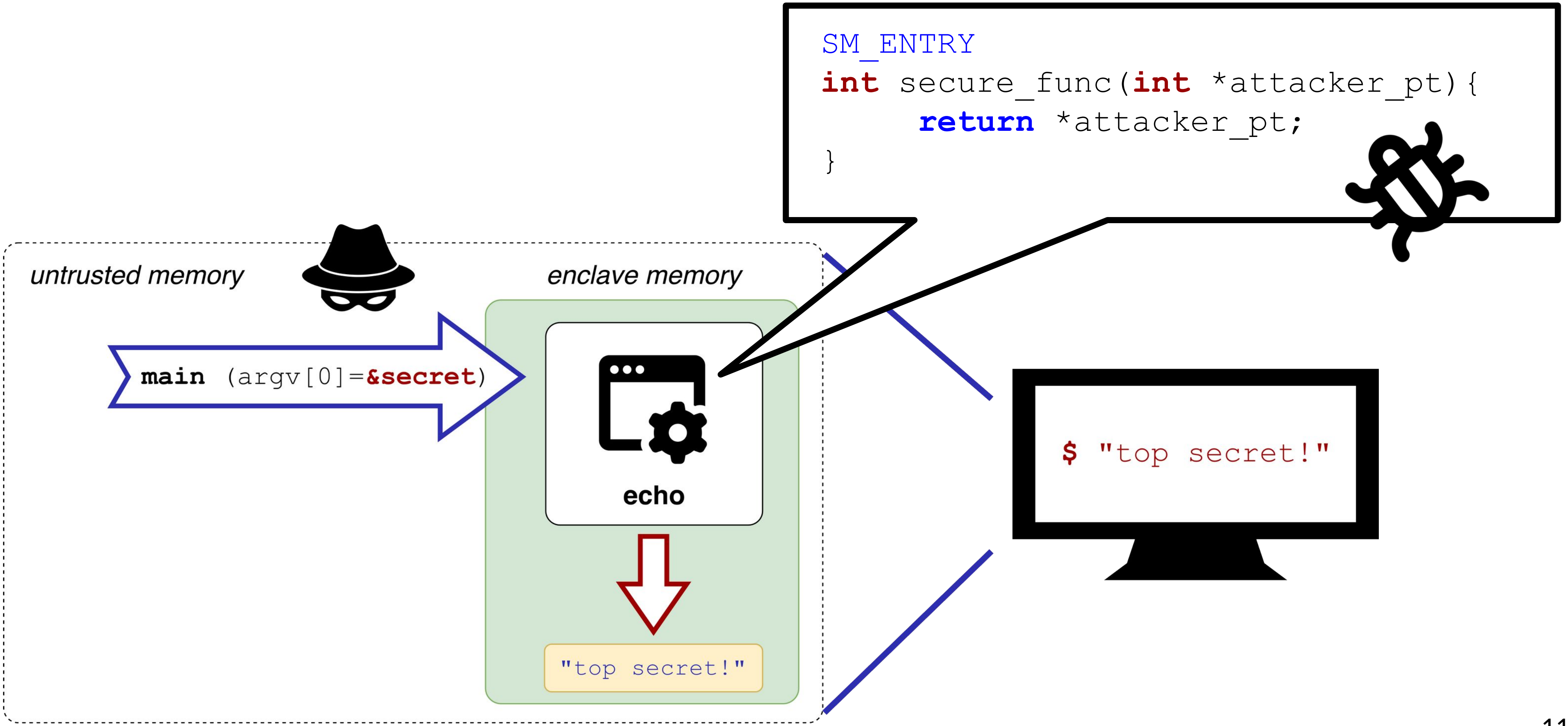
Van Bulck et al., “A Tale of Two Worlds: Assessing the Vulnerability of Enclave Shielding Runtimes”, CCS 2019.

Example: Confused-Deputy Pointer Attacks



```
SM_ENTRY
int secure_func(int *attacker_pt){
      return *attacker_pt;
}
```

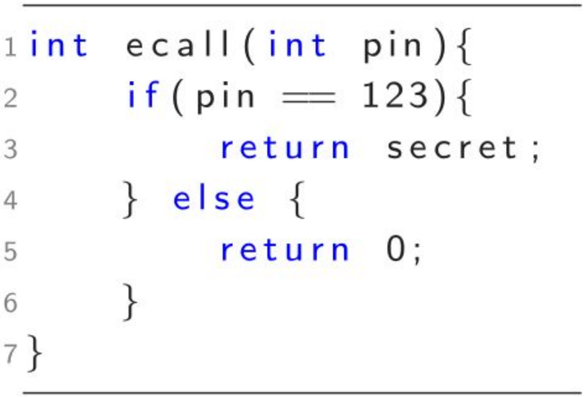
Example: Confused-Deputy Pointer Attacks



```
SM_ENTRY
int secure_func(int *attacker_pt){
    return *attacker_pt;
}
```

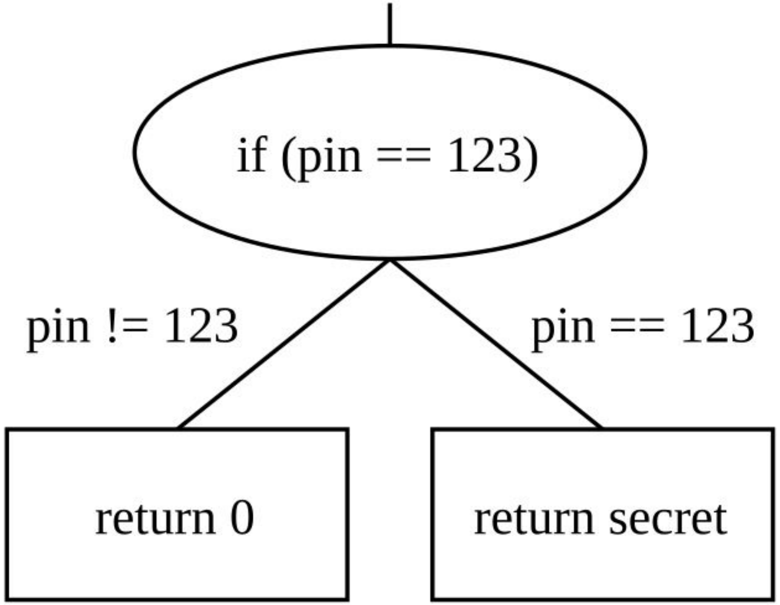
time to validate

Principled Software Validation: Symbolic Execution



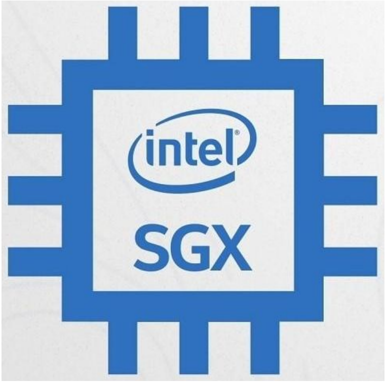
```
int ecall(int pin){
    if(pin == 123){
        return secret;
    } else {
        return 0;
    }
}
```

https://angr.io/



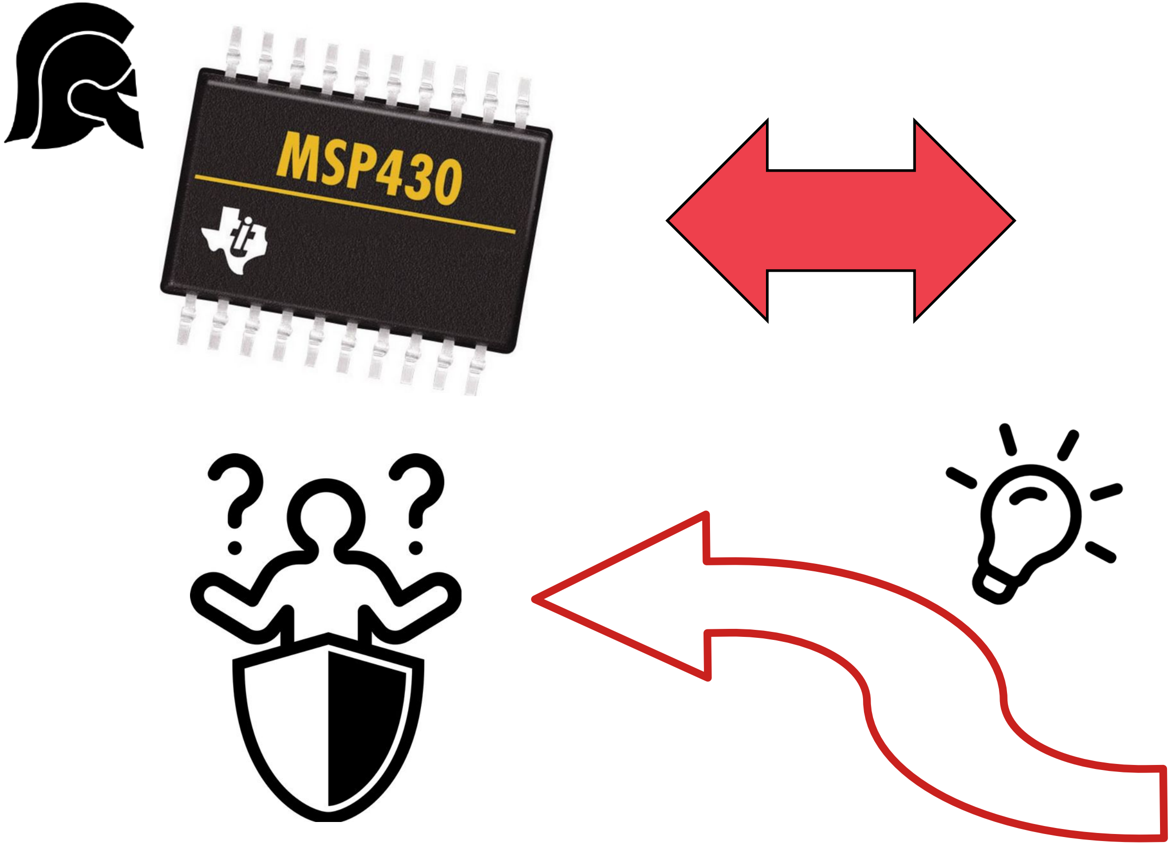
- Symbolic execution uses a constraint solver
- Execution works on instruction-level, i.e., as close to the binary as possible

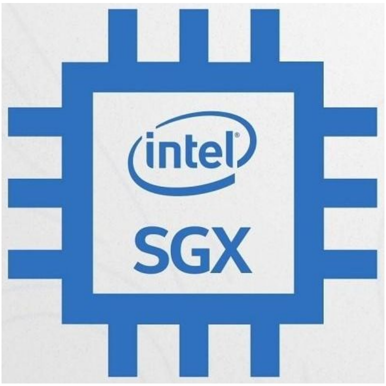
Research Gap: Symbolic Enclave Validation Tools



- TeeRex [USENIX’20]
- Coin [ASPLOS’20]
- Guardian [CCSW’21]
- SymGX [CCS’23]
- **Pandora** [S&P’24]

Research Gap: Symbolic Enclave Validation Tools

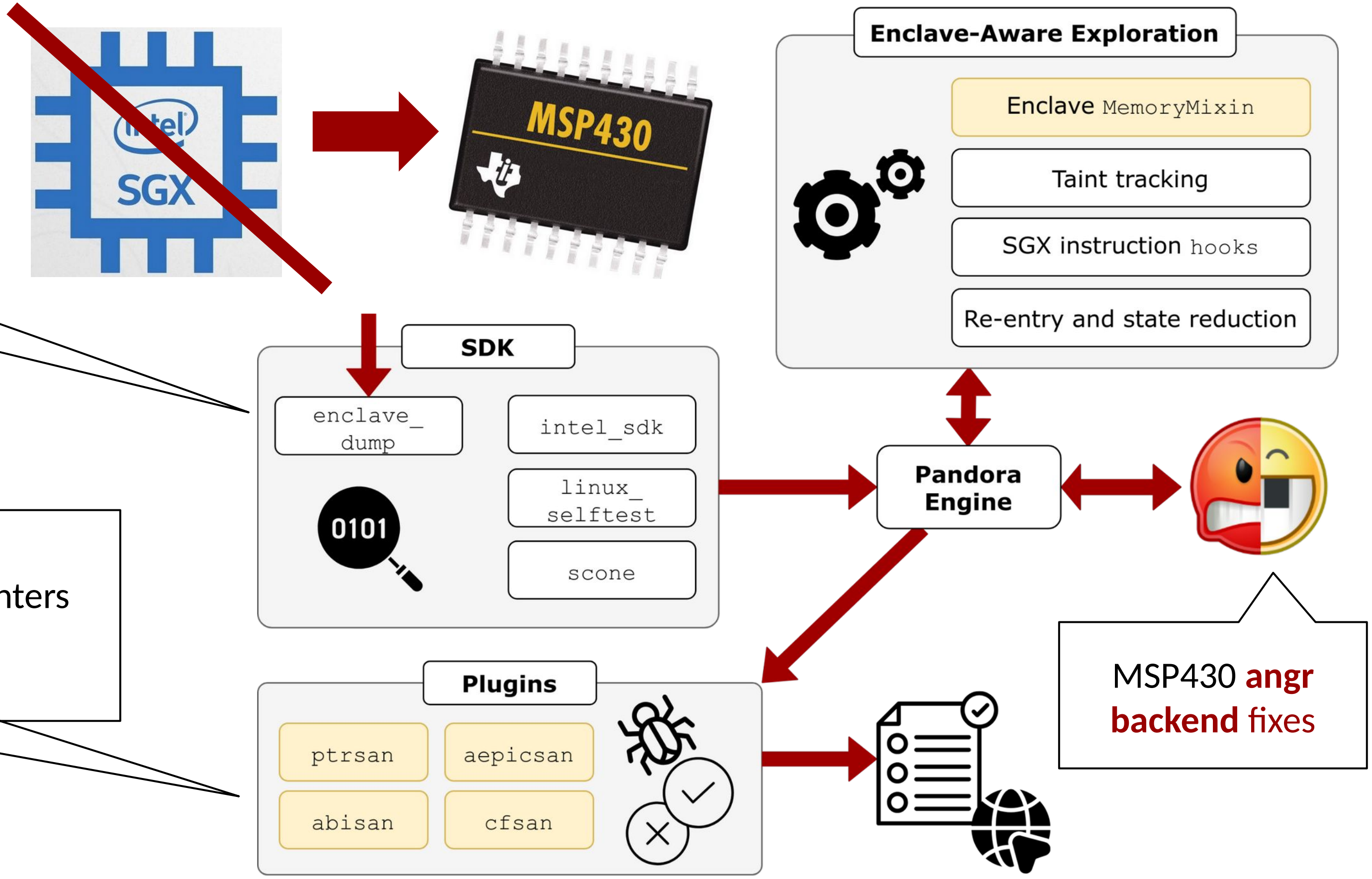




- TeeRex [USENIX’20]
- Coin [ASPLOS’20]
- Guardian [CCSW’21]
- SymGX [CCS’23]
- **Pandora** [S&P’24]

Principled Symbolic ~~Intel SGX~~ Sancus Enclave Validation

Sancus **enclave loader**



- **PtrSan:** Confused-deputy pointers
- **CfSan:** Control-flow hijacking

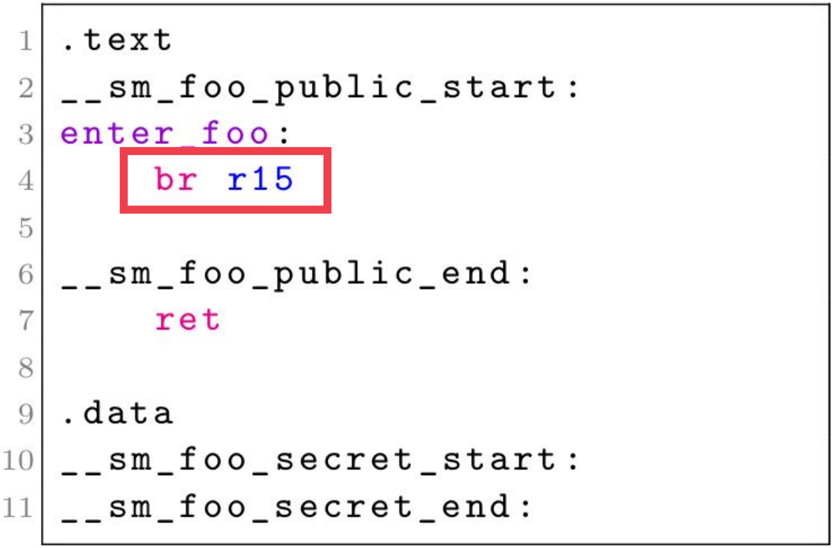
MSP430 **angr backend** fixes

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

Evaluation #1: Unit Test Framework

CfSan

→ 21 assembly testcases



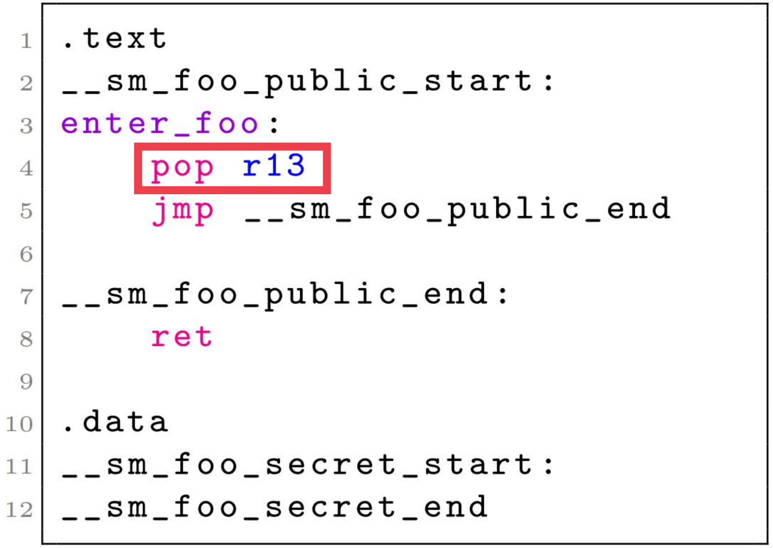
```
.text
__sm_foo_public_start:
enter_foo:
    br r15

__sm_foo_public_end:
    ret

.data
__sm_foo_secret_start:
__sm_foo_secret_end:
```

PtrSan

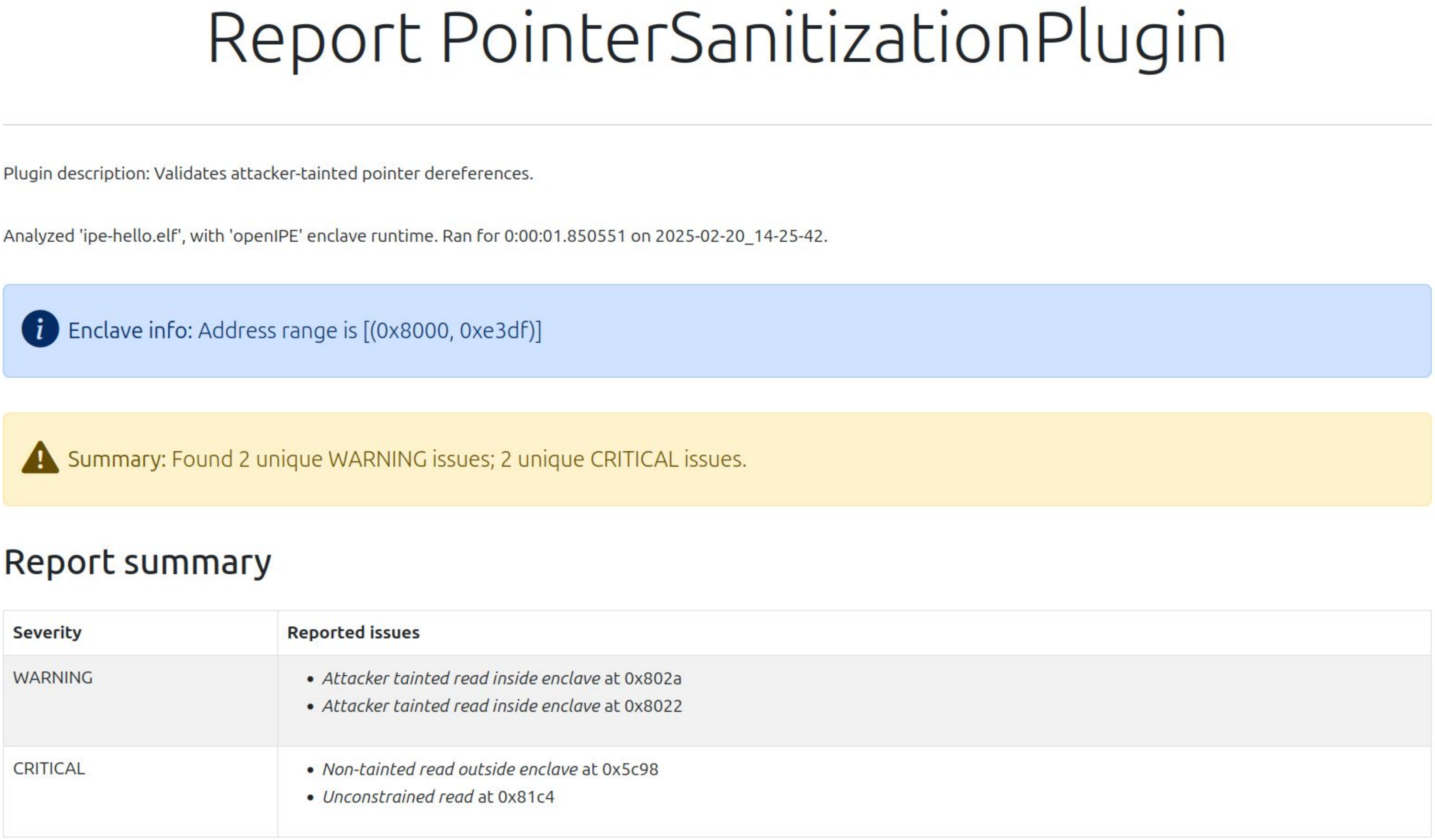
→ 15 assembly testcases



```
.text
__sm_foo_public_start:
enter_foo:
    pop r13
    jmp __sm_foo_public_end

__sm_foo_public_end:
    ret

.data
__sm_foo_secret_start:
__sm_foo_secret_end
```



Report PointerSanitizationPlugin

Plugin description: Validates attacker-tainted pointer dereferences.

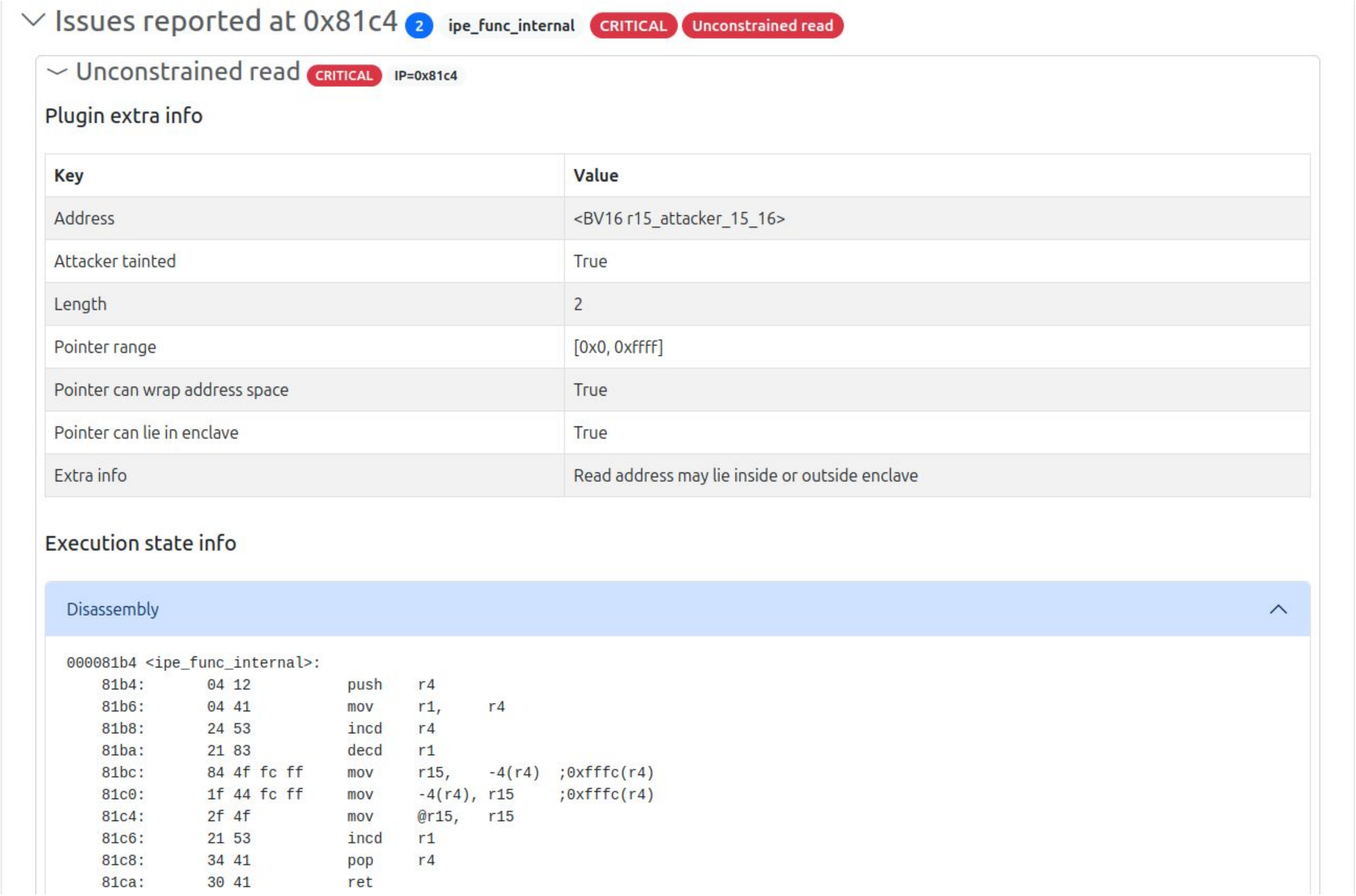
Analyzed 'ipe-hello.elf', with 'openIPE' enclave runtime. Ran for 0:00:01.850551 on 2025-02-20_14-25-42.

Enclave info: Address range is [(0x8000, 0xe3df)]

Summary: Found 2 unique WARNING issues; 2 unique CRITICAL issues.

Report summary

Severity	**Reported issues**
WARNING	• *Attacker tainted read inside enclave* at 0x802a • *Attacker tainted read inside enclave* at 0x8022
CRITICAL	• *Non-tainted read outside enclave* at 0x5c98 • *Unconstrained read* at 0x81c4



Issues reported at 0x81c4 2 ipe_func_internal CRITICAL Unconstrained read

Unconstrained read CRITICAL IP=0x81c4

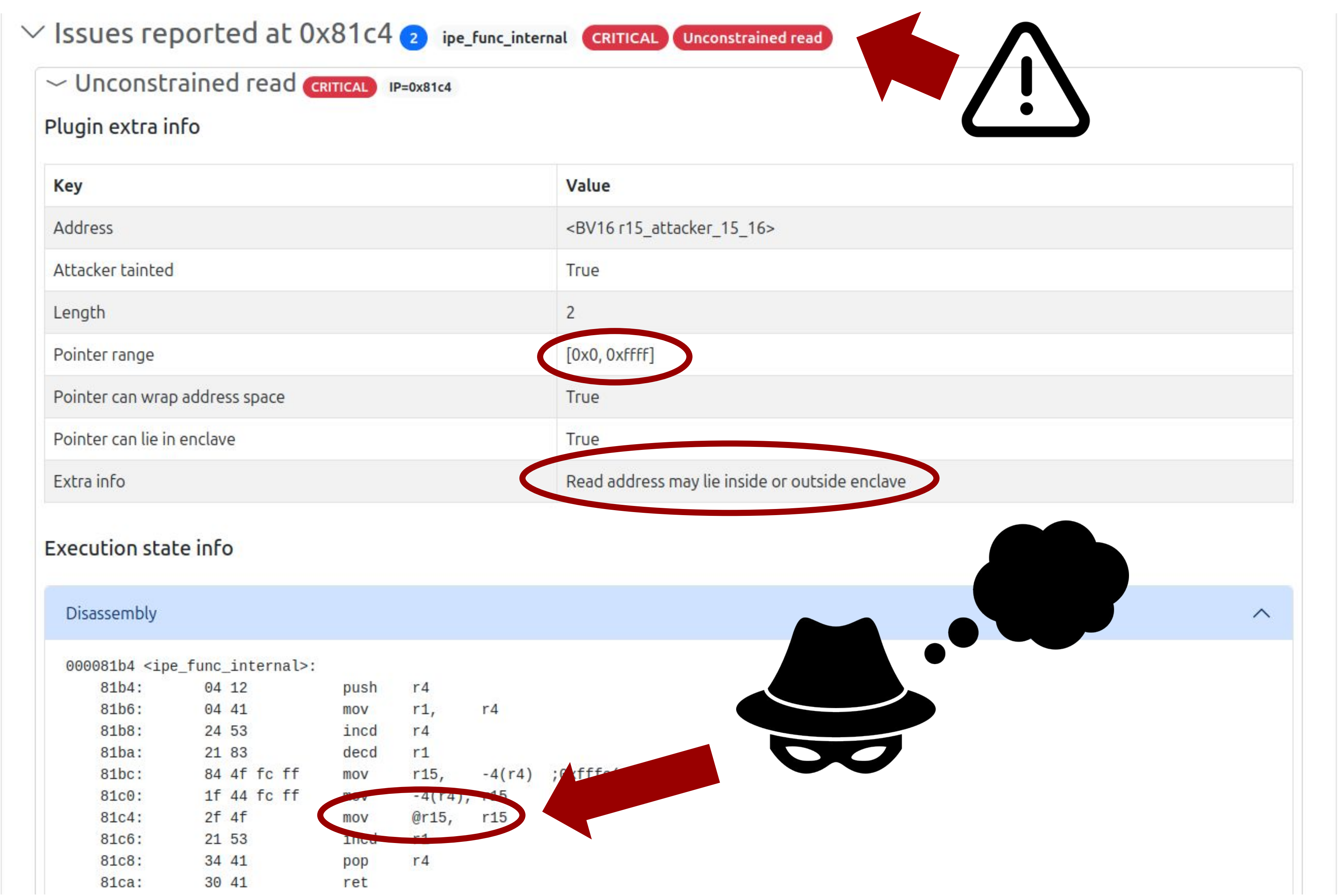
Plugin extra info

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(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

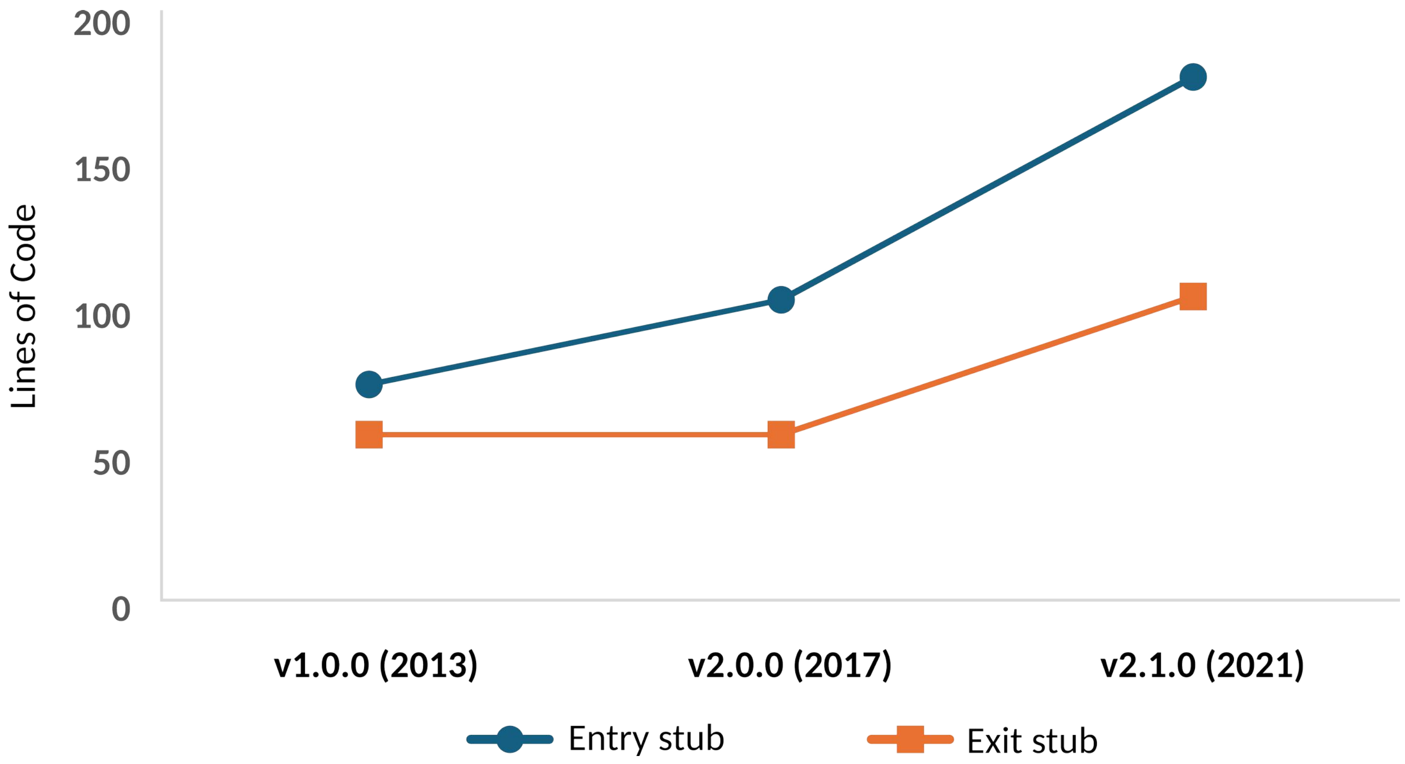
Key	Value
Address	<BV16 r15_attacker_15_16>
Attacker tainted	True
Length	2
Pointer range	[0x0, 0xffff]
Pointer can wrap address space	True
Pointer can lie in enclave	True
Extra info	Read address may lie inside or outside enclave

Execution state info

Disassembly

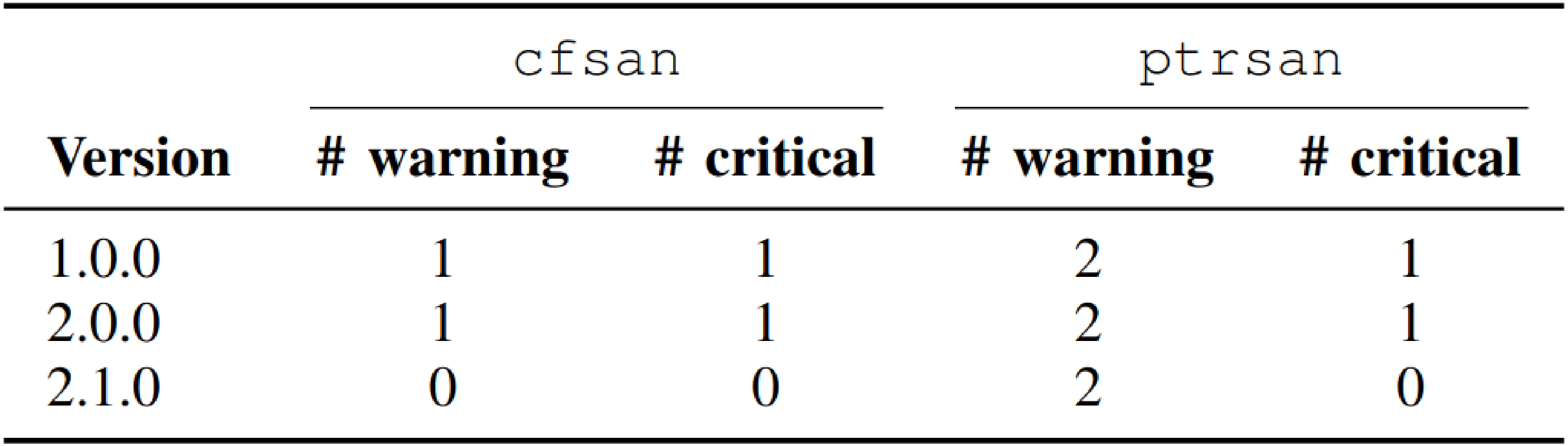
```
000081b4 <ipe_func_internal>:
    81b4:       04 12           push    r4
    81b6:       04 41           mov     r1,     r4
    81b8:       24 53           incd    r4
    81ba:       21 83           decd    r1
    81bc:       84 4f fc ff     mov     r15,    -4(r4)  ;0xfffc(
    81c0:       1f 44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

Evaluation #2: Sancus Trusted Runtime



Complexity: v1 (2013) << v2 (2017) << v2.1 (2021)

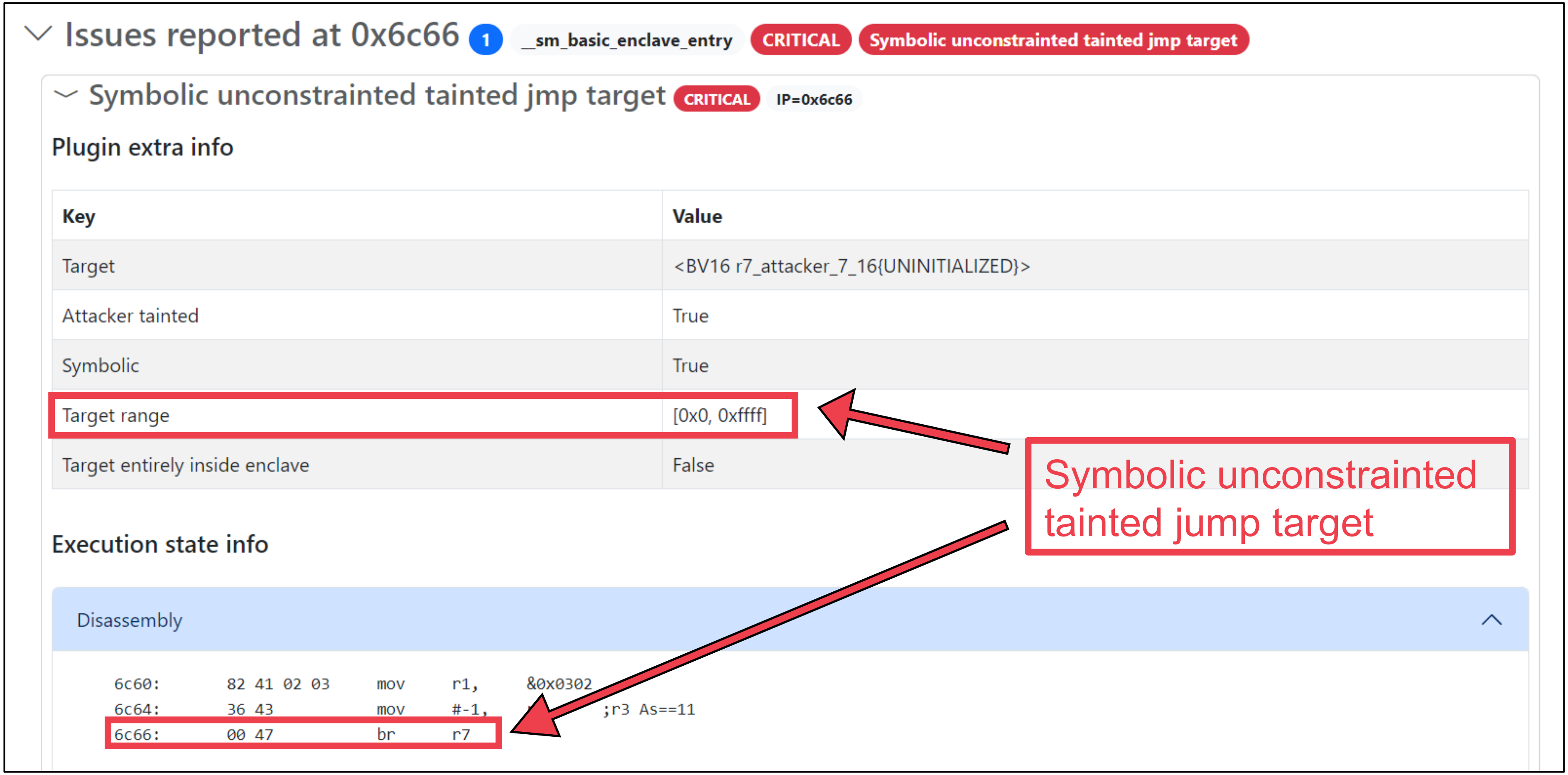
Evaluation #2: Sancus Trusted Runtime



	cfsan		ptrsan	
Version	# warning	# critical	# warning	# critical
1.0.0	1	1	2	1
2.0.0	1	1	2	1
2.1.0	0	0	2	0

Complexity: v1 (2013) << v2 (2017) << v2.1 (2021)

Example `CfSan`: Control-Flow Hijacking (<v2.1)



Issues reported at 0x6c66 1 __sm_basic_enclave_entry CRITICAL Symbolic unconstrainted tainted jmp target

Symbolic unconstrainted tainted jmp target CRITICAL IP=0x6c66

Plugin extra info

Key	Value
Target	<BV16 r7_attacker_7_16{UNINITIALIZED}>
Attacker tainted	True
Symbolic	True
Target range	[0x0, 0xffff]
Target entirely inside enclave	False

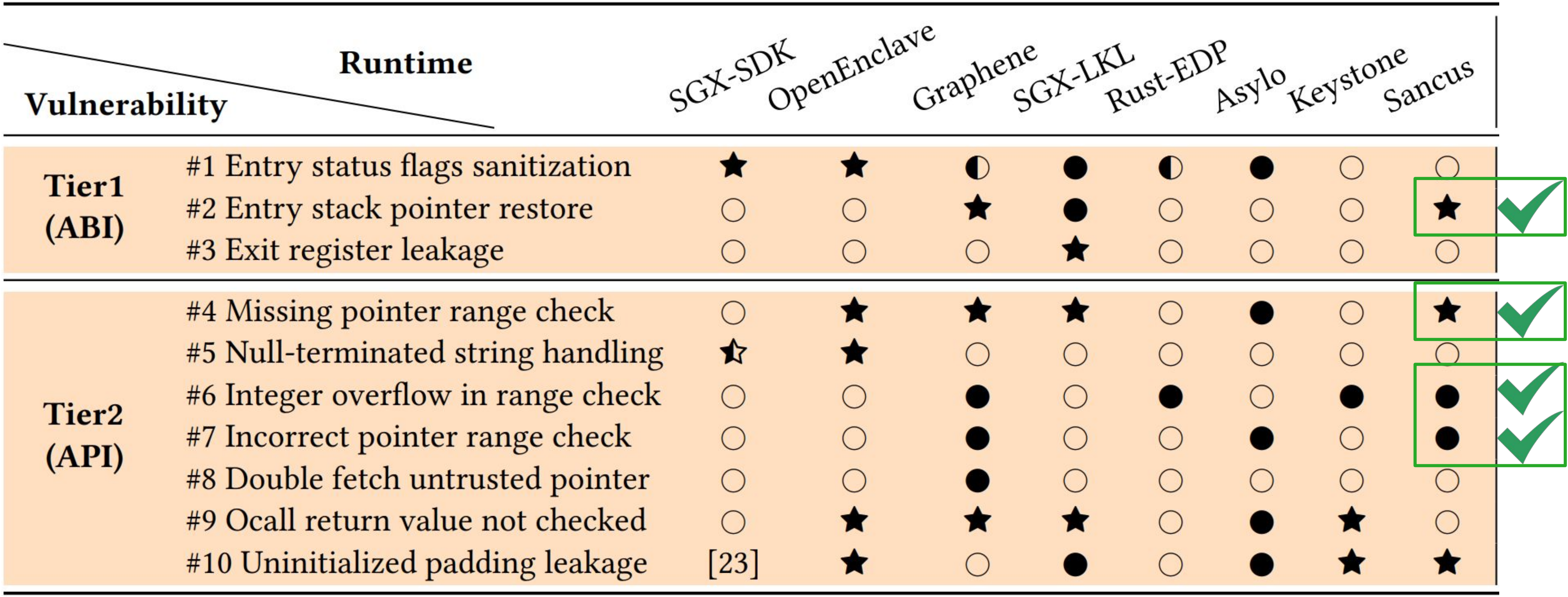
Symbolic unconstrainted tainted jump target

Execution state info

Disassembly

```
6c60:       82 41 02 03     mov     r1,     &0x0302
6c64:       36 43           mov     #-1,            ;r3 As==11
6c66:       00 47           br      r7
```

Evaluation #3: Sancus Applications and Libraries



	Runtime Vulnerability	SGX-SDK	OpenEnclave	Graphene	SGX-LKL	Rust-EDP	Asylo	Keystone	Sancus	
Tier1 (ABI)	#1 Entry status flags sanitization	★	★	◐	●	◐	●	○	○	
	#2 Entry stack pointer restore	○	○	★	●	○	○	○	★	✓
	#3 Exit register leakage	○	○	○	★	○	○	○	○	
Tier2 (API)	#4 Missing pointer range check	○	★	★	★	○	●	○	★	✓
	#5 Null-terminated string handling	⯪	★	○	○	○	○	○	○	
	#6 Integer overflow in range check	○	○	●	○	●	○	●	●	✓
	#7 Incorrect pointer range check	○	○	●	○	○	●	○	●	✓
	#8 Double fetch untrusted pointer	○	○	●	○	○	○	○	○	
	#9 Ocall return value not checked	○	★	★	★	○	●	★	○	
	#10 Uninitialized padding leakage	[23]	★	○	●	○	●	★	★	

🗋 Van Bulck et al., "A Tale of Two Worlds: Assessing the Vulnerability of Enclave Shielding Runtimes", CCS 2019.

Conclusions and Take-Away

- **TEE-agnostic:** Symbolic hardware abstraction layer

  → *Intel SGX + MSP430 Sancus + (open)IPE*

- **Extensible:** Vulnerability validation via plugins

  → `PtrSan` + `CfSan` + ...

- **Evaluation:** Effective reproduction + unit tests

  → *CI/CD: Unit tests + trusted runtime/applications*

Thank you! Questions?

`github.com/pandora-tee`

</> SysTEX'25 Artifact Evaluated Available

</> SysTEX'25 Artifact Evaluated Functional

</> SysTEX'25 Artifact Evaluated Reusable

Sancus compilation passing

Sancus validation passing