

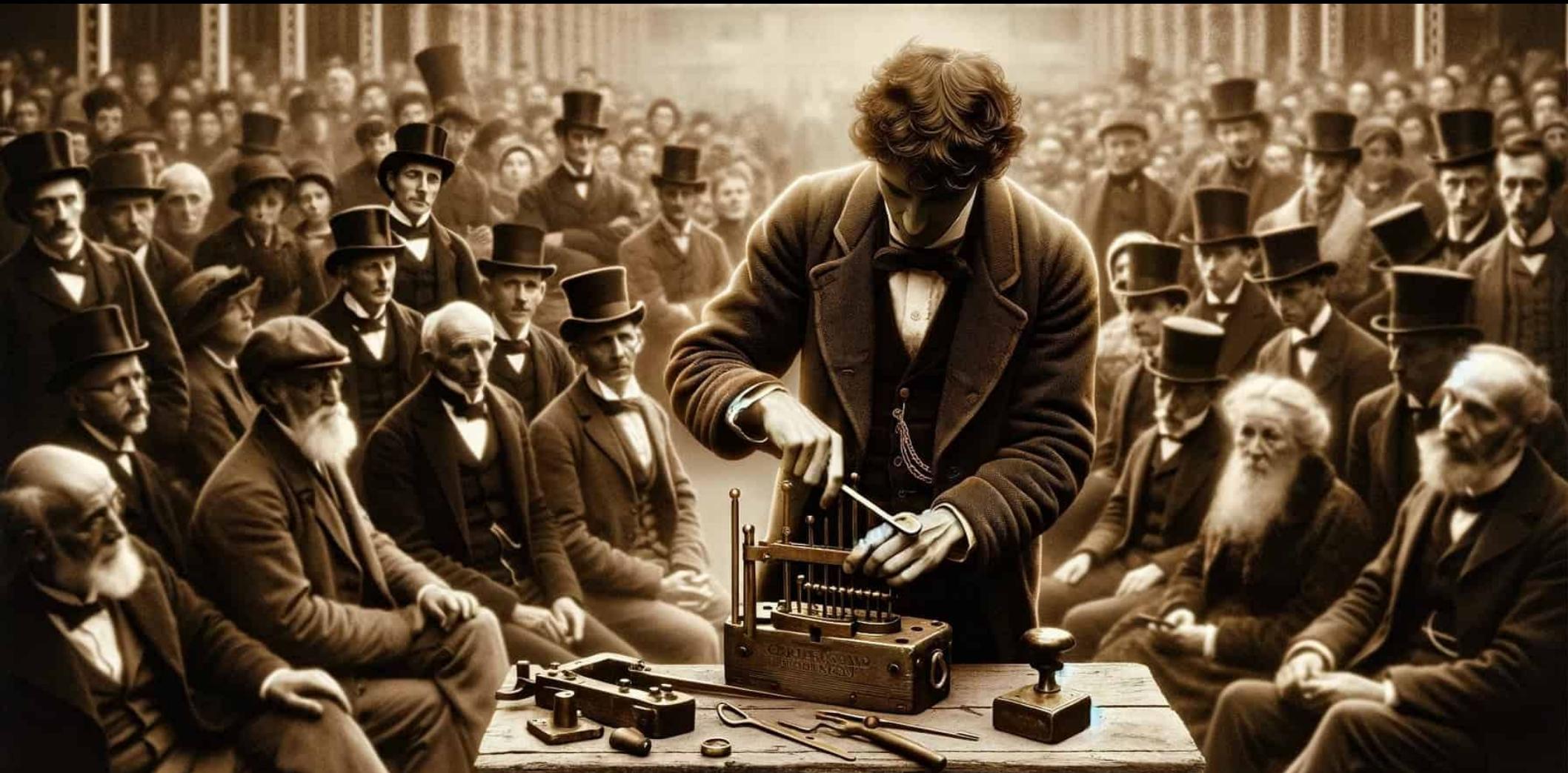
# Hack, Fix, Repeat: FOSS and the Future of Systems Security

**Jo Van Bulck**

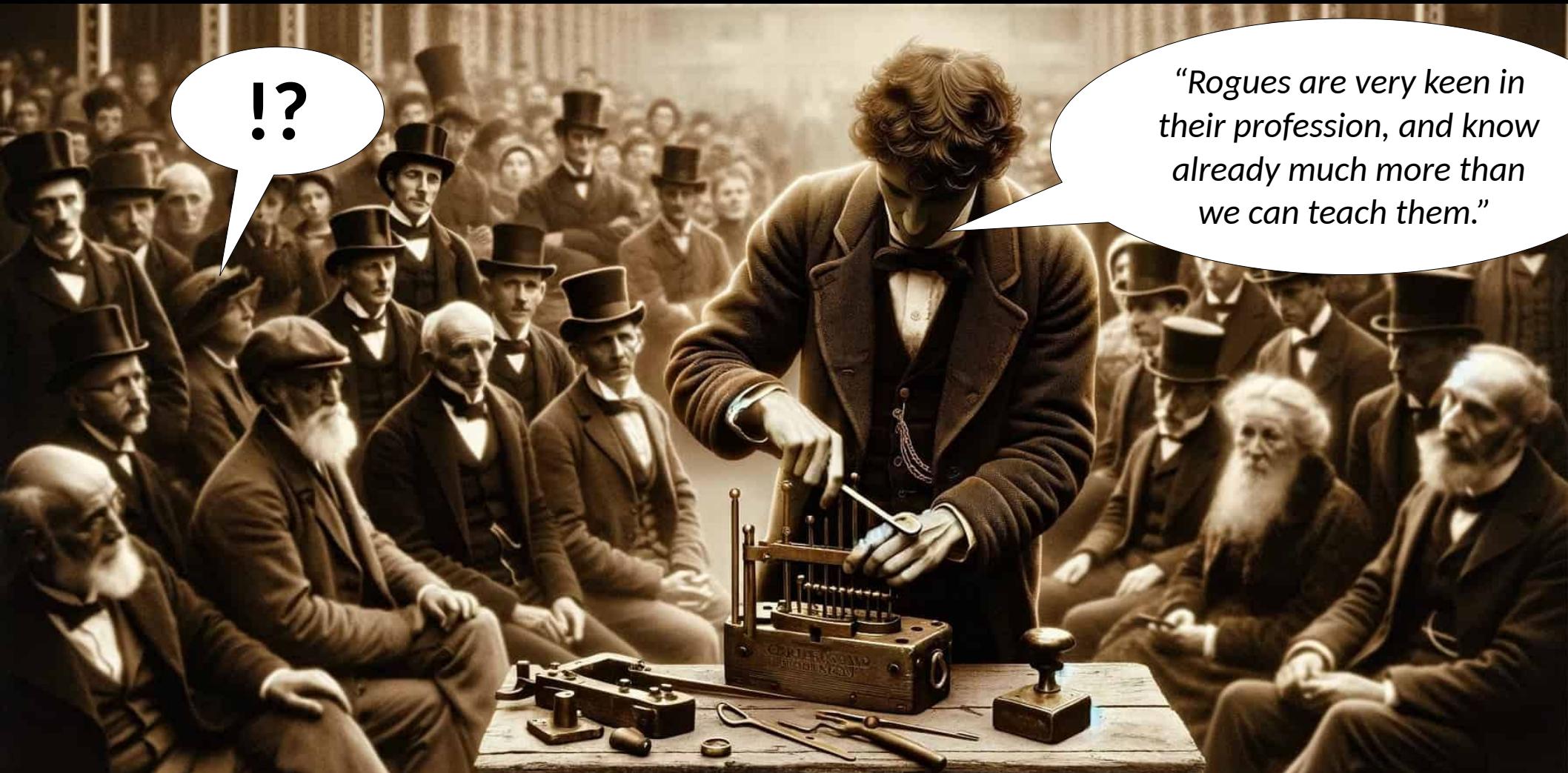
🏠 DistriNet, KU Leuven, Belgium 📩 jo.vanbulck@cs.kuleuven.be 🌐 vanbulck.net

Research Software Engineering Day, Dec 4, 2025

# The “Great Lock Controversy” at the Industrial Exhibition of 1851



# The “Great Lock Controversy” at the Industrial Exhibition of 1851



"Rogues are very keen in their profession, and know already much more than we can teach them."

# Kerckhoff's Principle: No Security through Obscurity

*“Il faut qu'il n'exige pas le secret, et qu'il puisse sans inconvénient tomber entre les mains de l'ennemi.”*

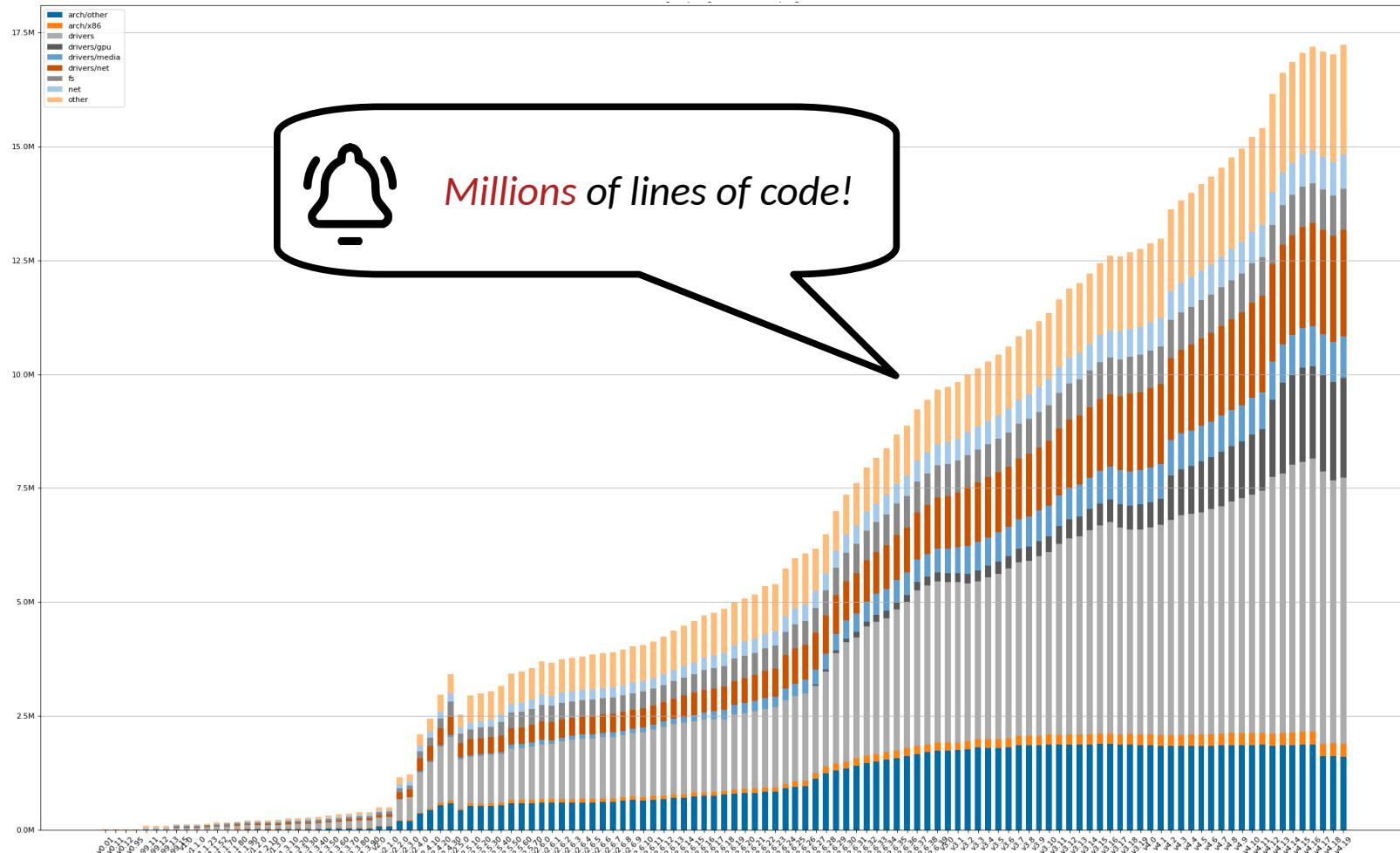
– Auguste Kerckhoff (1883)

# Linus's Law: Security through Open Source?

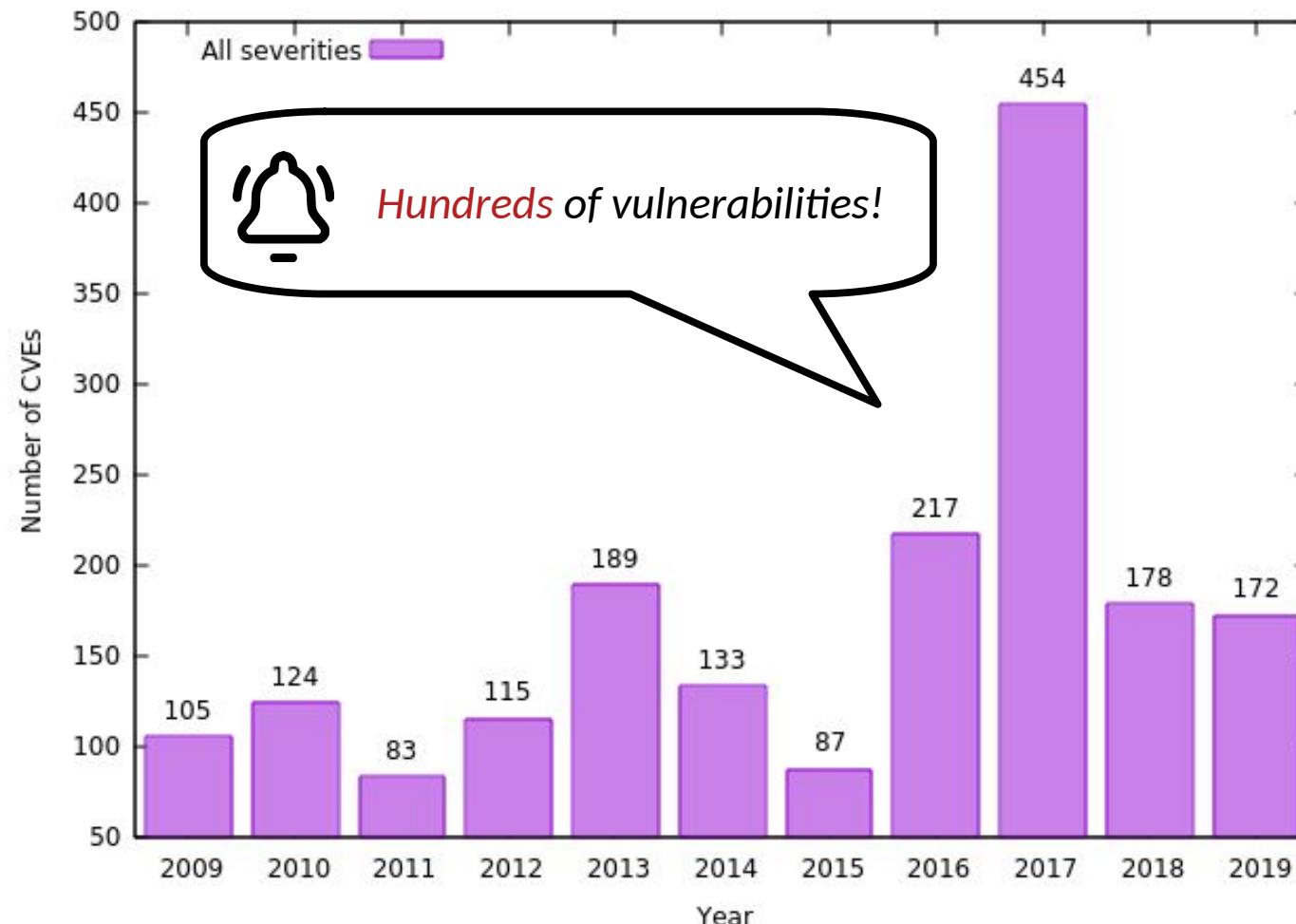
*“Given enough eyeballs, all bugs are shallow.”*

– Eric S. Raymond (1997)

# Context: Evolution of Linux Kernel Lines of Code

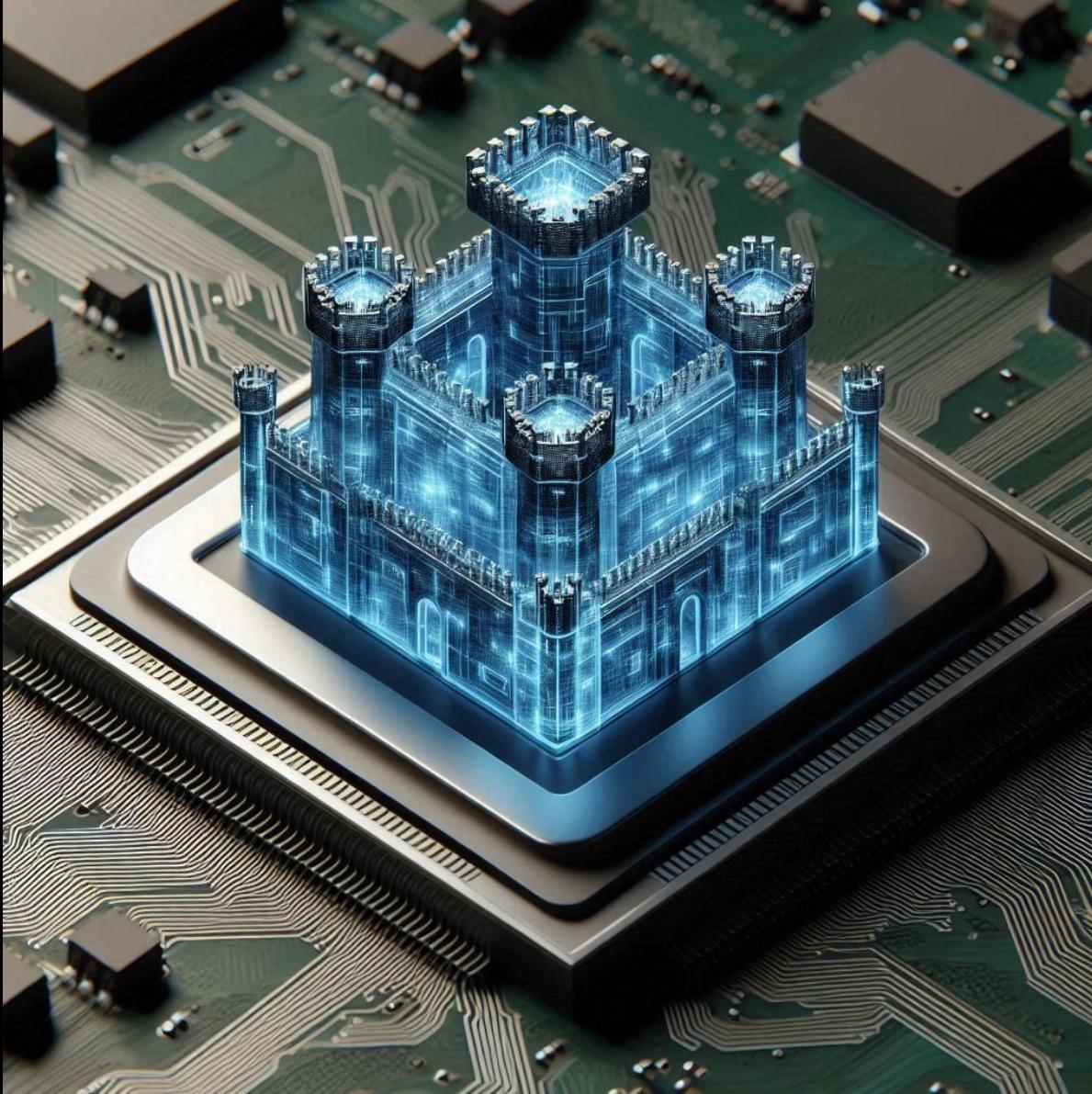


# Context: Evolution of Linux Kernel Vulnerabilities



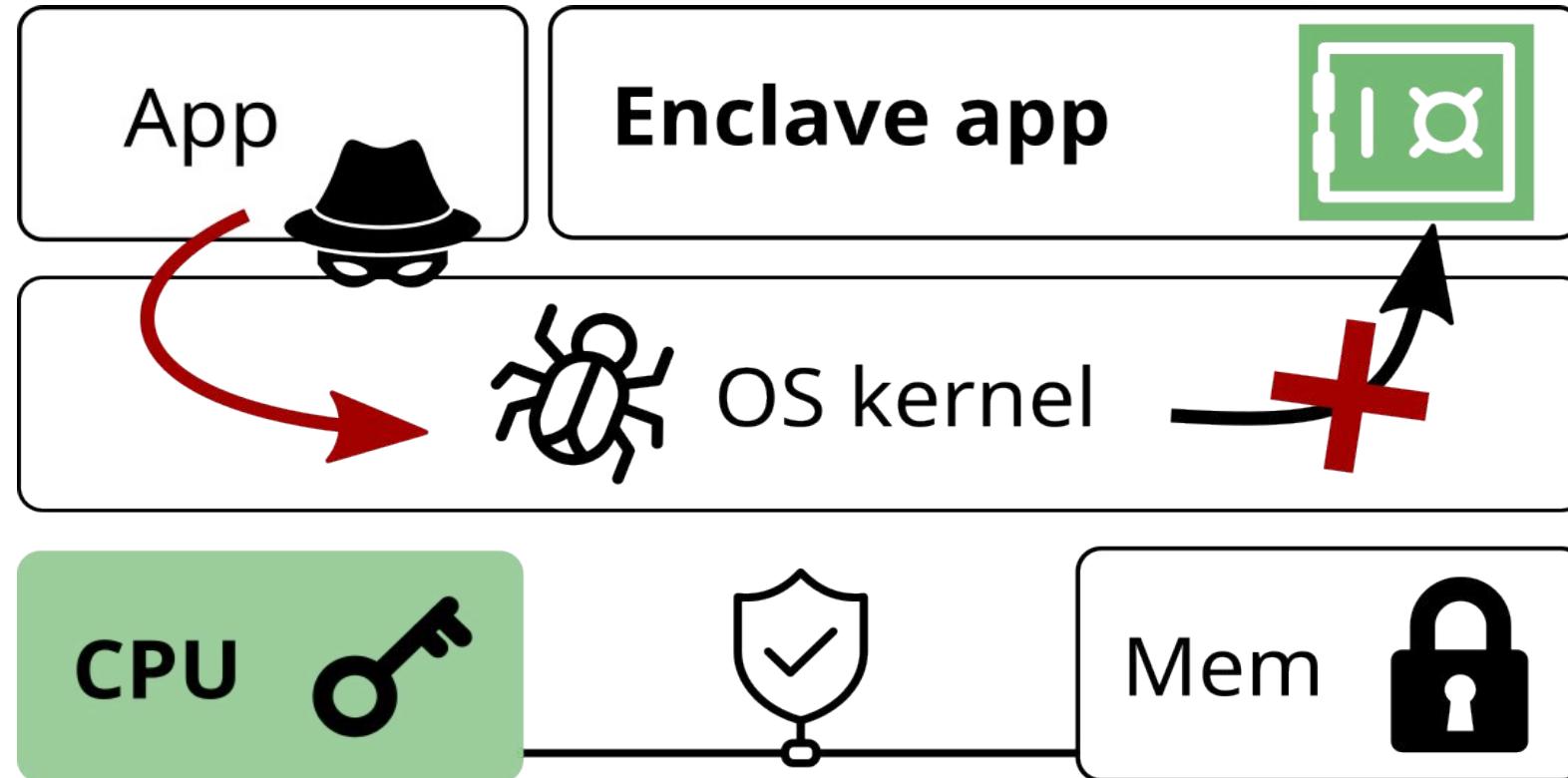
# Research Landscape

*Reducing attack surface with confidential computing*



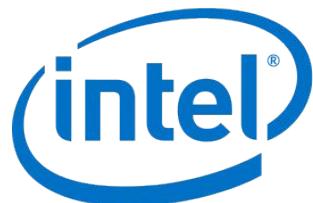
DALL·E 3

# Confidential Computing: Reducing Attack Surface



Trusted execution: Hardware-level **isolation** and attestation

# The Rise of Trusted Execution Environments (TEEs)

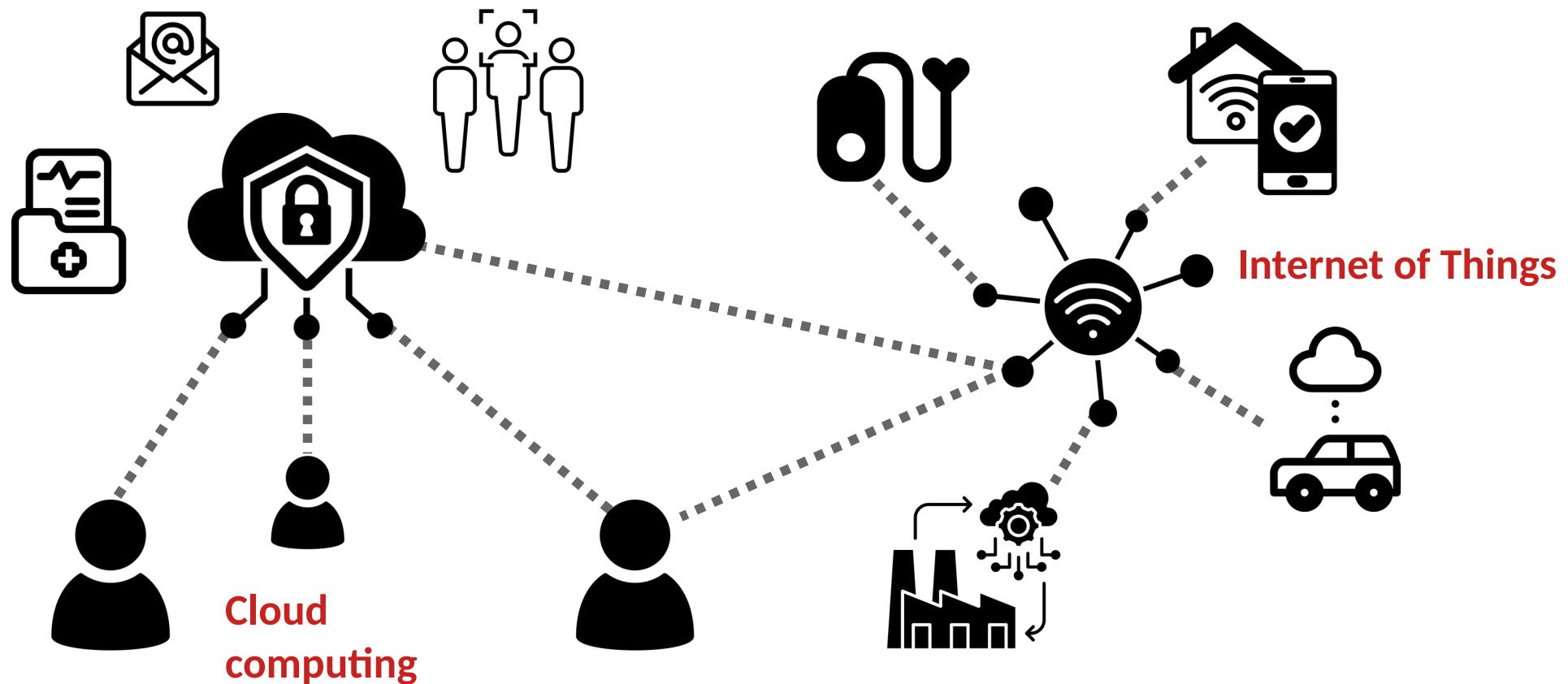


- 2004: ARM TrustZone
- 2015: **Intel Software Guard Extensions (SGX)**
- 2016: AMD Secure Encrypted Virtualization (SEV)
- 2018: IBM Protected Execution Facility (PEF)
- 2020: AMD SEV with Secure Nested Paging (SEV-SNP)
- 2022: Intel Trust Domain Extensions (TDX)
- 2023: ARM Confidential Compute Architecture (CCA)
- 2024: NVIDIA Confidential Computing

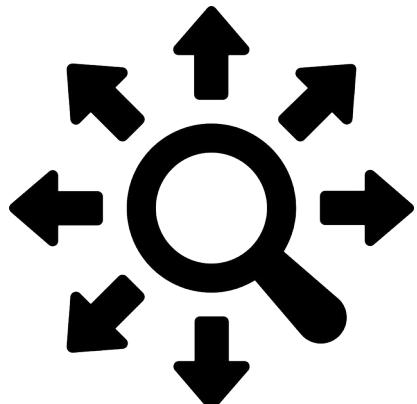


TEEs are here to stay...

# “Confidential Computing Today, Just Computing Tomorrow” \*

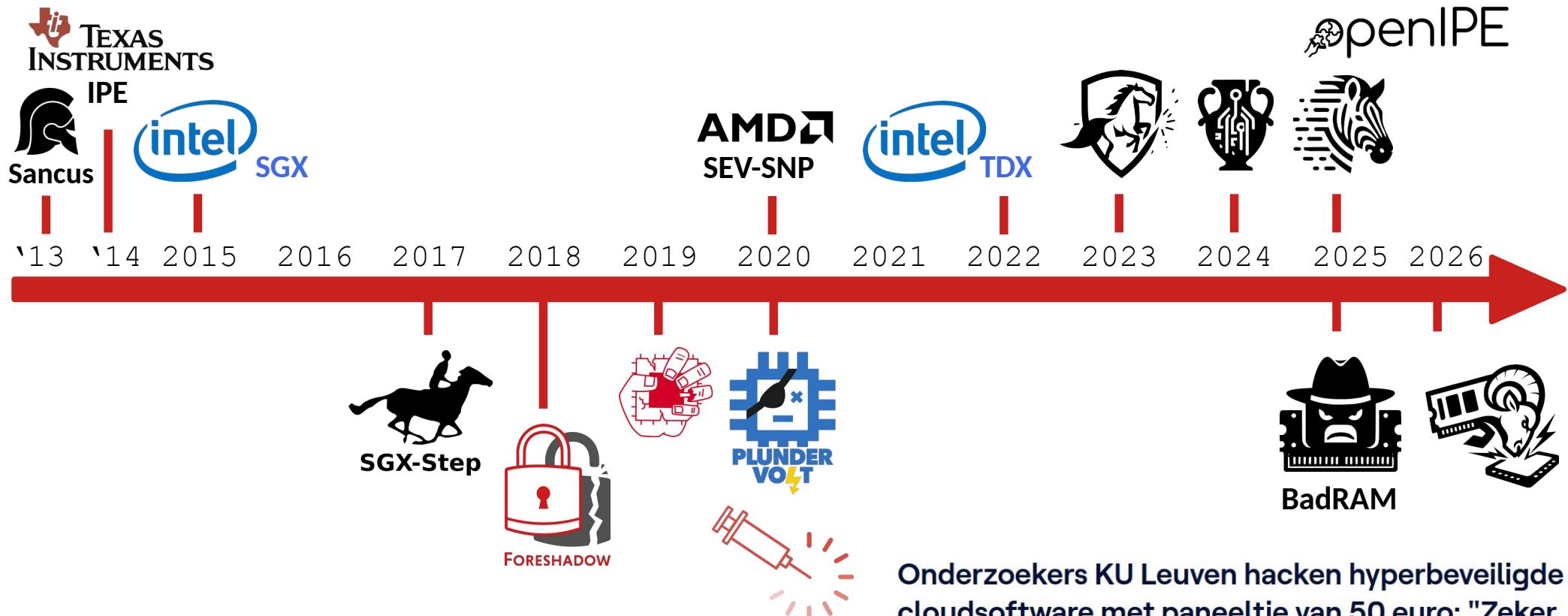


# Research Agenda: (Un)confidential Computing?



- **Offensive security analysis** of **closed-source** (large) commercial systems
  - *Critical analysis of vendor claims...*
- **Defensive prototypes** on **open-source** (small) research systems
  - *Next-generation innovations...*

# A Decade of Vetting Confidential Computing @DistriNet



Hackers raken in hart van Intel-processor  
De Standaard

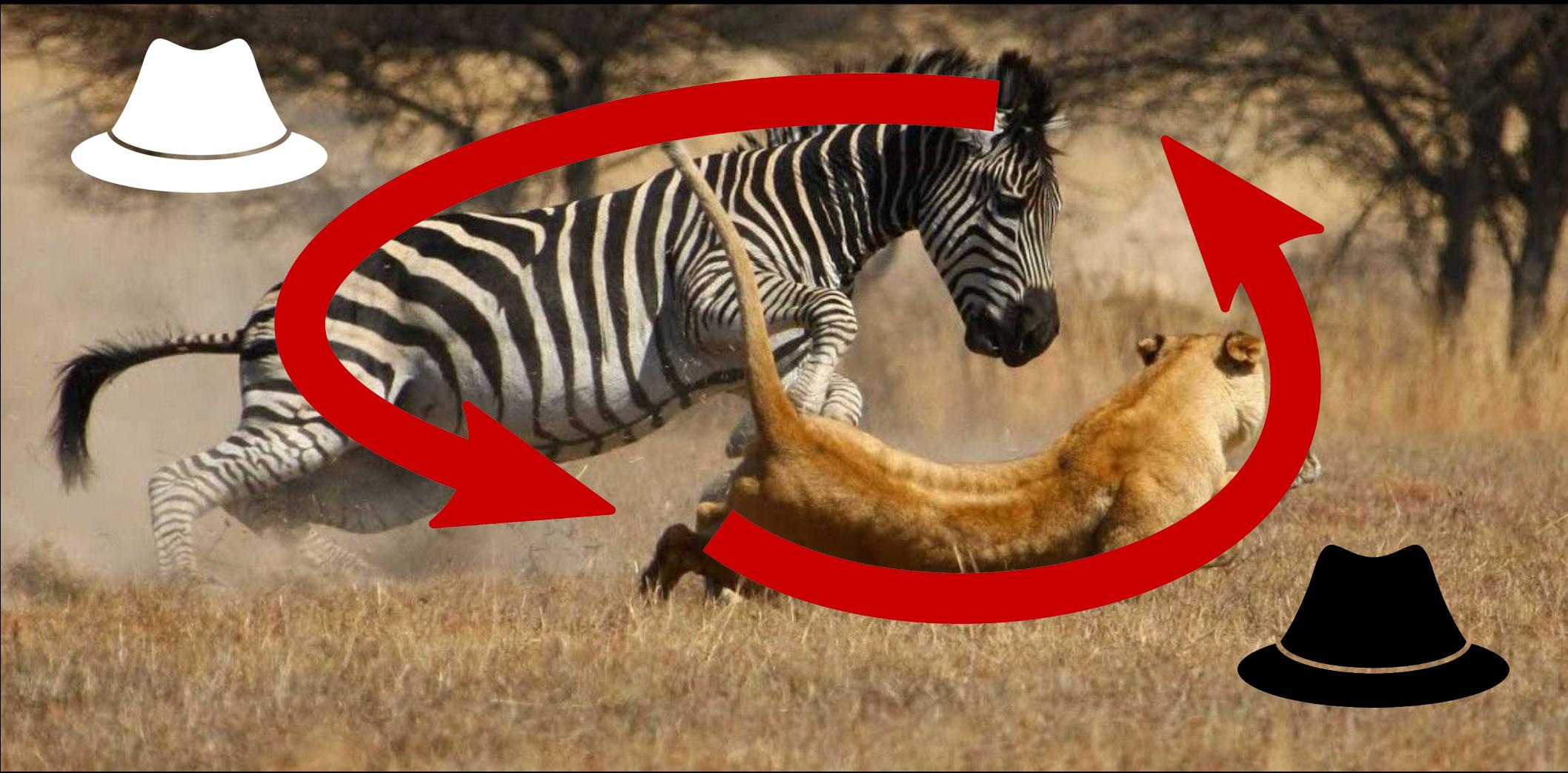
Onderzoekers KU Leuven hacken hyperbeveiligde  
cloudsoftware met paneeltje van 50 euro: "Zeker



# Scientific Understanding Driven by Attacker-Defender Race...



# Scientific Understanding Driven by Attacker-Defender Race...

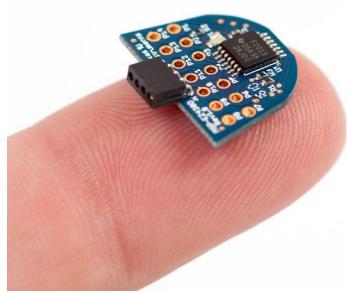




# Highlight #1: Sancus

*Do's and don'ts for long-lived open-source research prototypes*

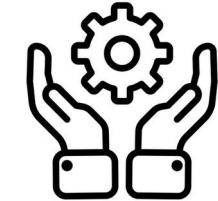
# Sancus: Lightweight Trusted Computing for the IoT



+



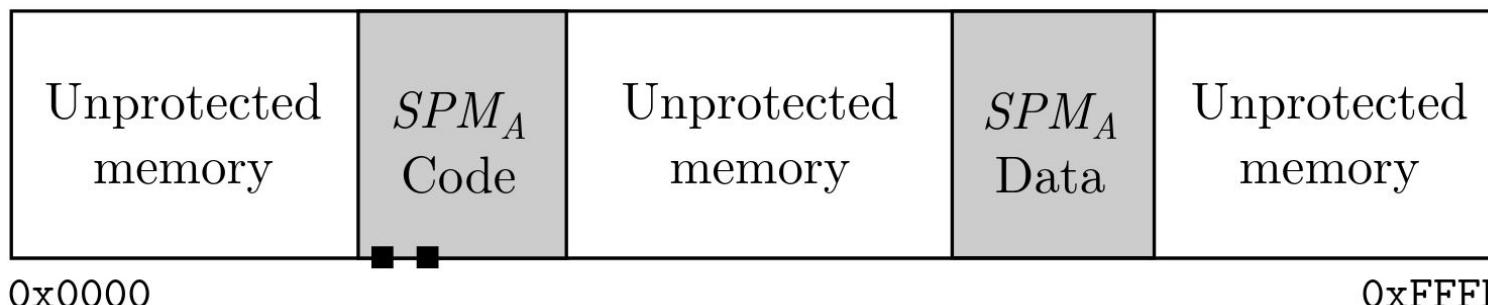
+

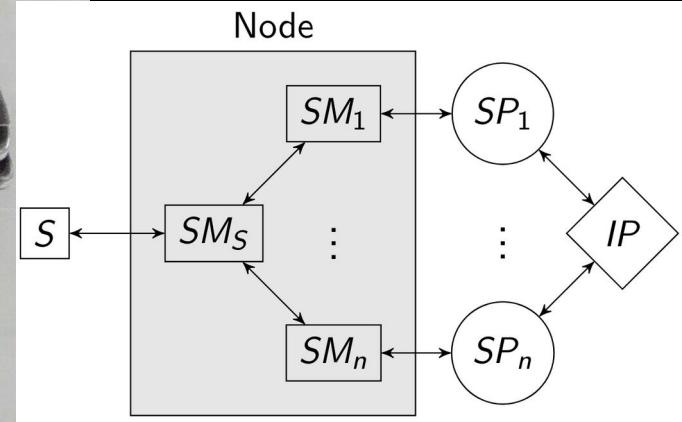
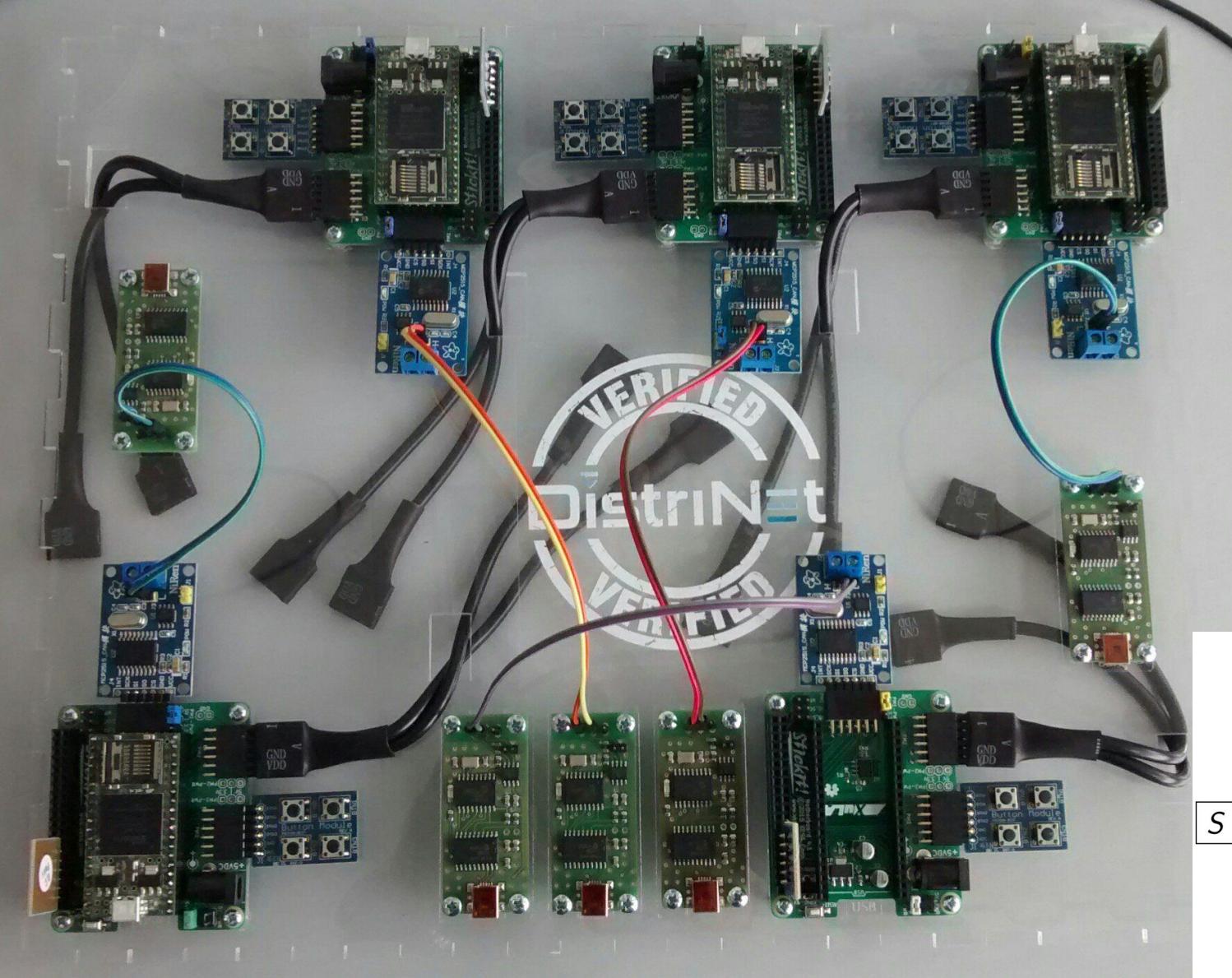


**OpenMSP430 CPU extensions  
for isolation + attestation**

**LLVM compiler pass**

**Support software  
“operating system”**



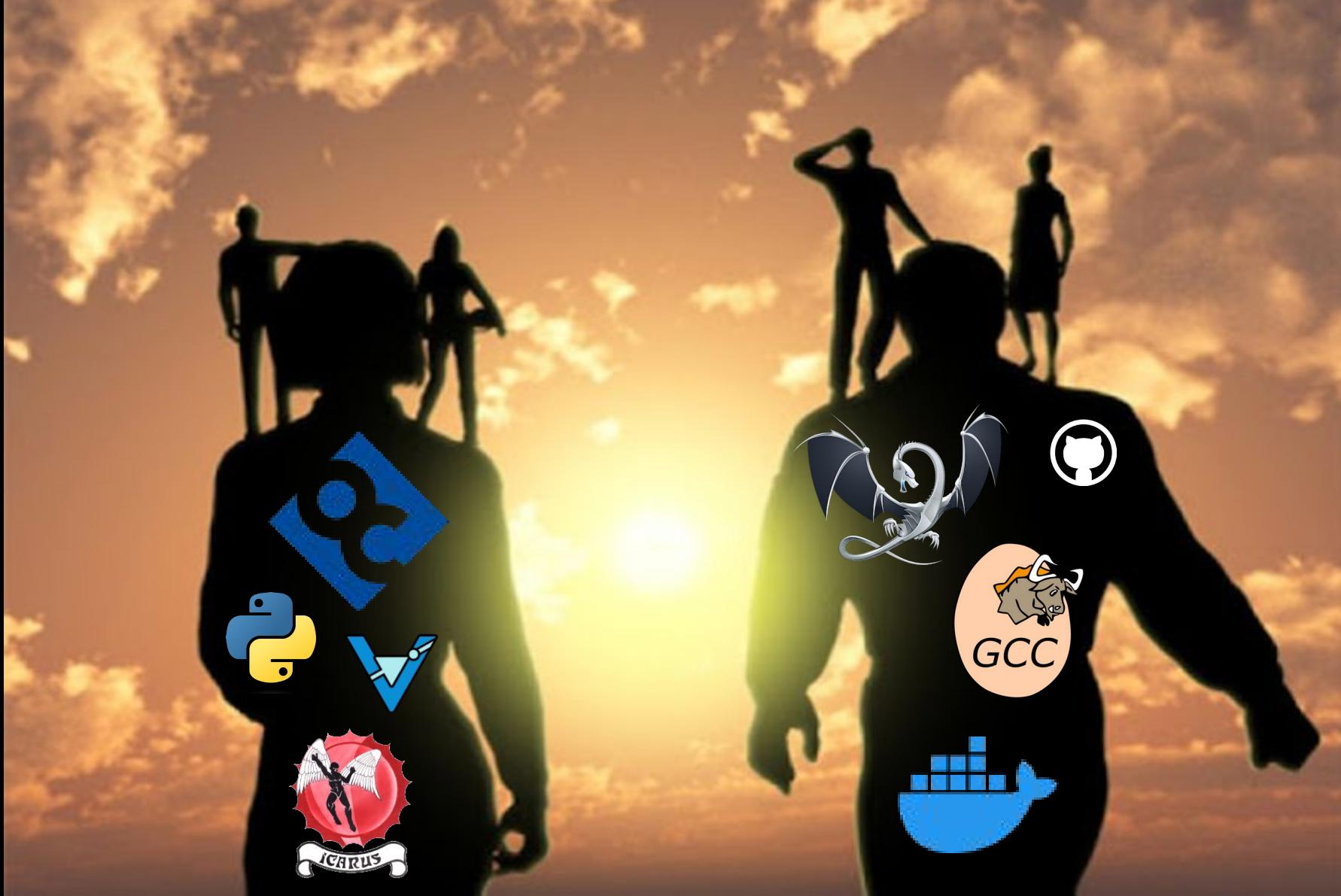




## Key #1: Open-Source Ecosystem

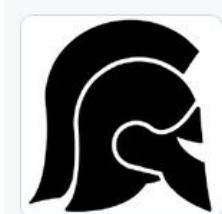
---





# Sancus: Open-Source Artifacts for Reproducible Science

- No commercialization/patents; FOSS licenses
- Limit **dependencies**: e.g., LLVM <> GCC
- **Upstream** eagerly: Avoid dead forks...
  - 2012-2017: Public **tarballs** + private git
  - 2017: Move to public **GitHub** organization



## Sancus

A Lightweight Trusted Execution Environment for Secure IoT Devices

3 followers

imec-DistriNet, KU Leuven, Belgium

<https://distrinet.cs.kuleuven.be/software/>

Overview

Repositories 16

Projects

Packages

Teams

People 8

*"The most important book about technology today,  
with implications that go far beyond programming."*  
— Gary Kawasaki

Revised & Expanded

## THE CATHEDRAL & THE BAZAAR

MUSINGS ON LINUX AND OPEN SOURCE  
BY AN ACCIDENTAL REVOLUTIONARY



ERIC S. RAYMOND

WITH A FOREWORD BY BOB YOUNG, CHAIRMAN & CEO OF RED HAT, INC.



## Key #2: Modular Base Design

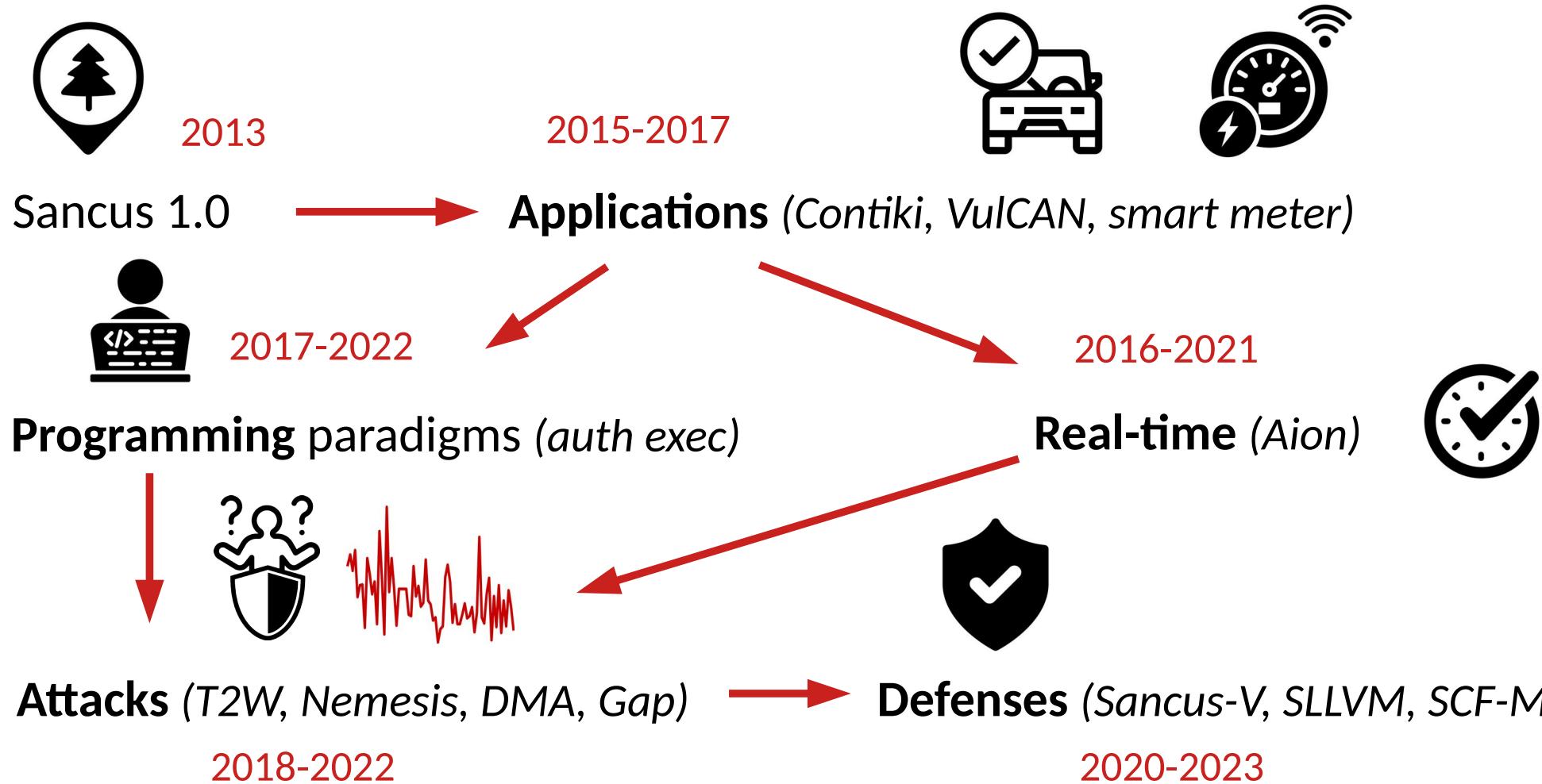
---



**PIVOT!**

**PIVOT!**

# Sancus is Dead, Long Live Sancus!

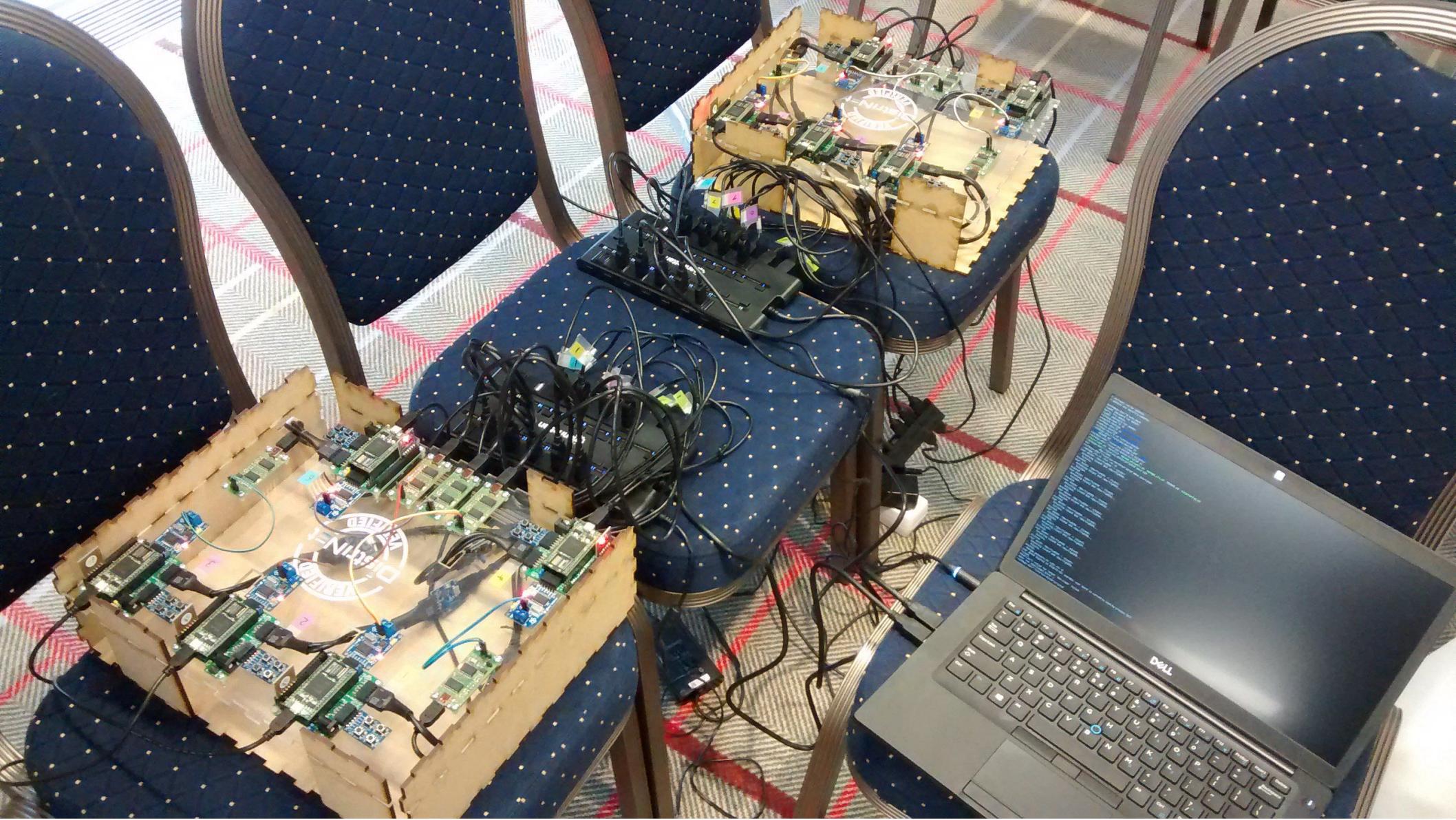




## Key #3: Build Usable Systems

---





# Sancus: Building Usable Systems

- Large **engineering effort** >> minimal publication effort
- Simulators and test frameworks
- **Continuous integration**
- Tutorial [DSN'18] → VulCAN [ACSAC'20]

Docker passing Run Sancus examples passing

*"I'm happy to say that the evaluation worked flawlessly - great job!"*

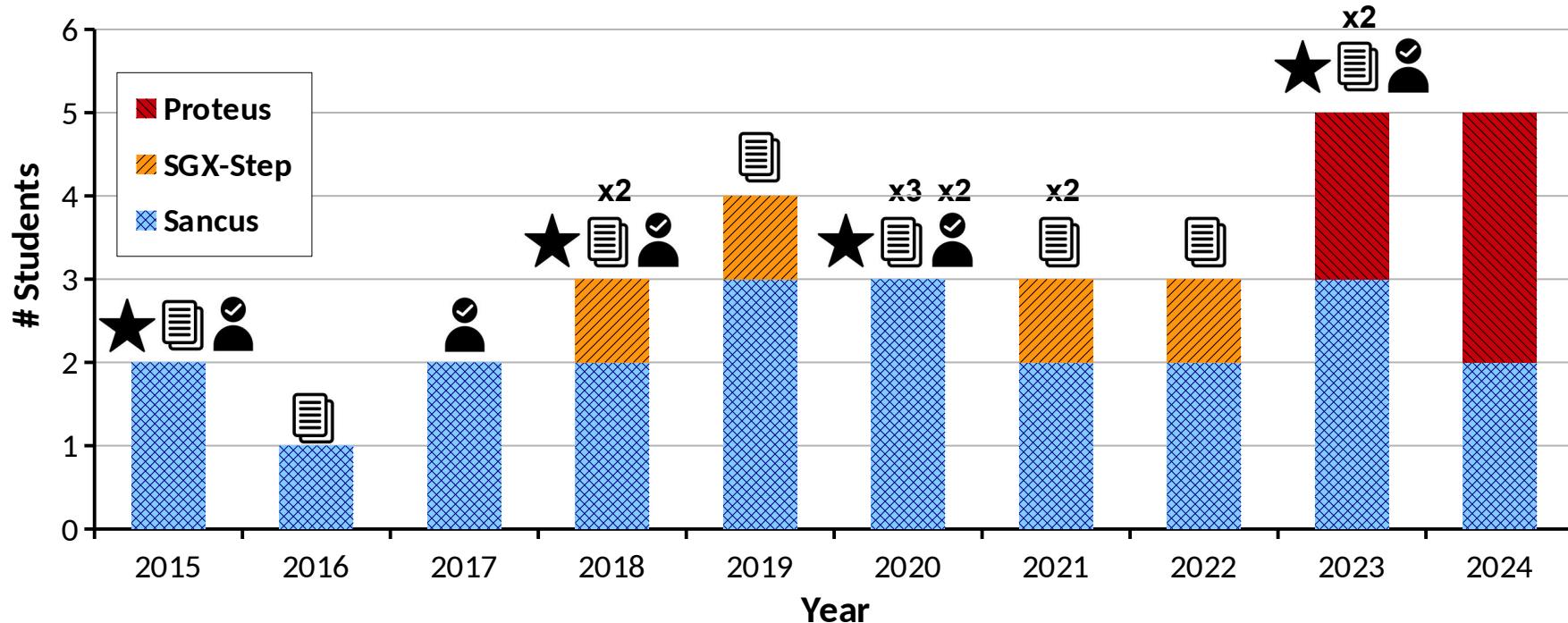




## Key #4: Impact through Education

---

# Maximizing Impact through Education



- Continuous **master thesis** involvement
- Understanding: Putting **theory into practice**  
→ e.g., *MIT xv6 OS, KUL Proteus RISC-V core*



## Key #4: Science Communication

---

# Sancus: Lightweight and Open-Source Trusted Computing for the IoT

[View on GitHub](#)[Watch a demo](#)[Explore Research](#)

**“** We do have problems with security, ones that need to be dealt with, not only with changes to software toolchains but also to the underlying hardware.

—Rik Farrow *USENIX ;login:*



## SOFTWARE ISOLATION

Outside software cannot read or write a protected module's runtime state. A module can only be called through one of its designated entry points.



## SECURE COMMUNICATION

Sancus safeguards the authenticity, integrity, and freshness of all traffic between a protected module and its remote provider.



## LIGHTWEIGHT CRYPTOGRAPHY

A minimalist cryptographic hardware unit enables low-overhead symmetric key derivation, authenticated encryption, and hashing.



## SECURE I/O

Secure driver modules have exclusive ownership over memory-mapped I/O peripheral devices, and can implement software-defined access control policies.



## SOFTWARE ATTESTATION

Remote or local parties can verify at runtime that a particular software module has been isolated on a specific node without having been tampered with.



## BACKWARDS COMPATIBILITY

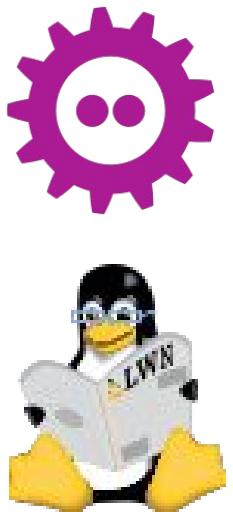
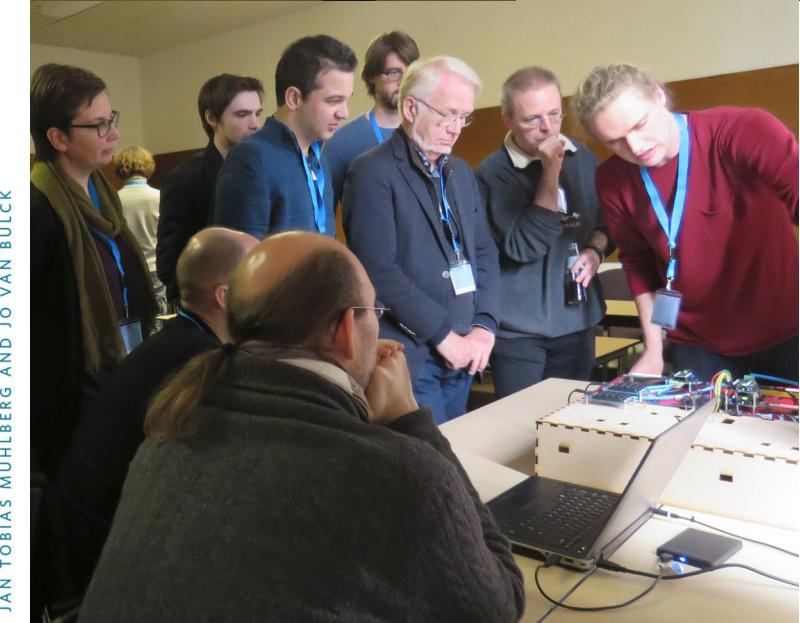
Legacy applications continue to function as expected; critical components can be migrated gradually into Sancus-protected modules.

# AUTOMOTIVE COMPUTING



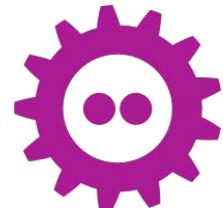
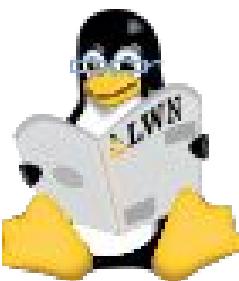
## Reflections on Post-Meltdown Trusted Computing A Case for Open Security Processors

JAN TOBIAS MÜHLBERG AND JO VAN BULCK



*“A project based on open-source building blocks  
and free-software ethos [...] should be lauded  
and considered by anyone [...]”*

– Mischa Spiegelmock, LWN.net, 2018

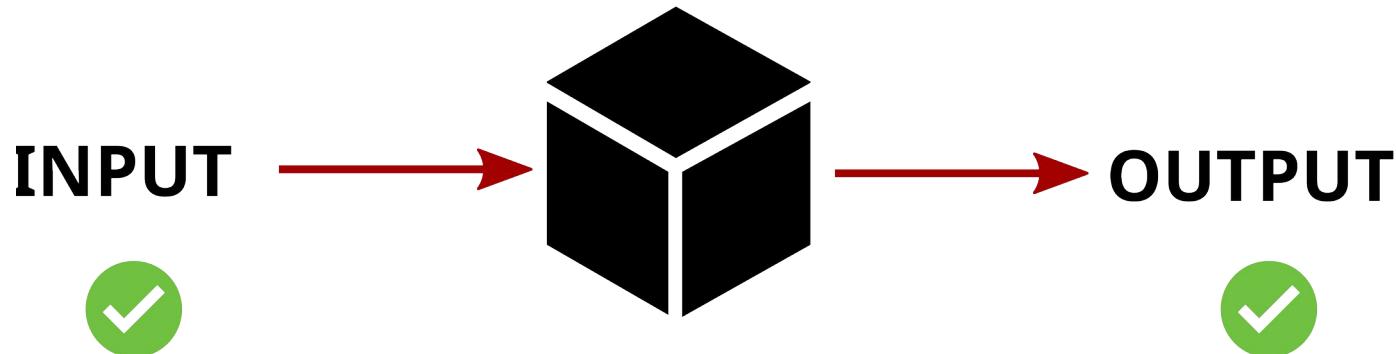




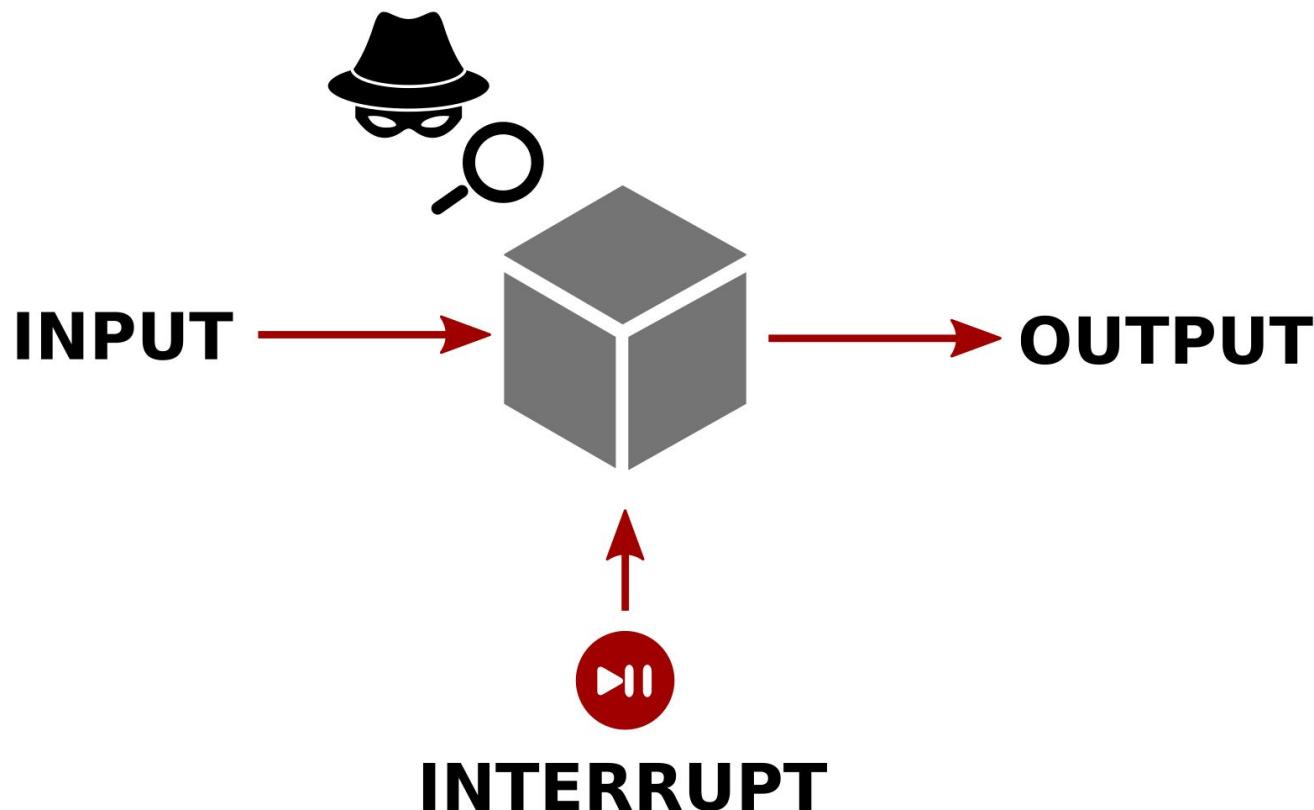
## Highlight #2: SGX-Step

*From open-source research to real-world impact*

# SGX-Step: Executing Enclaves one Instruction at a Time



# SGX-Step: Executing Enclaves one Instruction at a Time



# SGX-Step: Open-Source Release



Used in > 50 academic  
papers worldwide



ACSAC 2023  
Cybersecurity Artifacts  
Impact Award

## SGX-Step



<https://github.com/jovanbulck/sgx-step>

Watch

22

Star

245

Fork

52



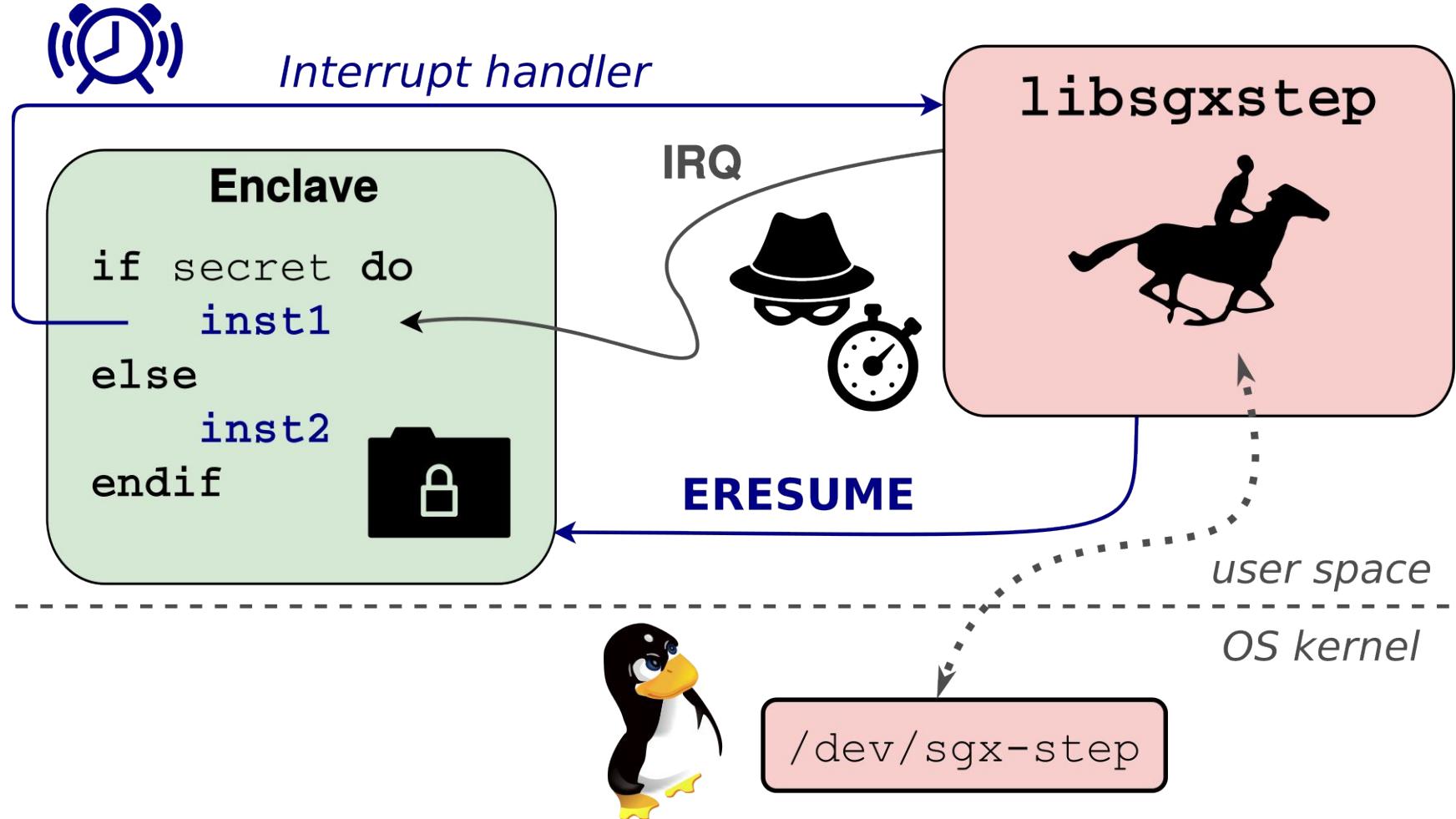
Van Bulck et al., "SGX-Step: A Practical Attack Framework for Precise Enclave Execution Control", SysTEX 2017.



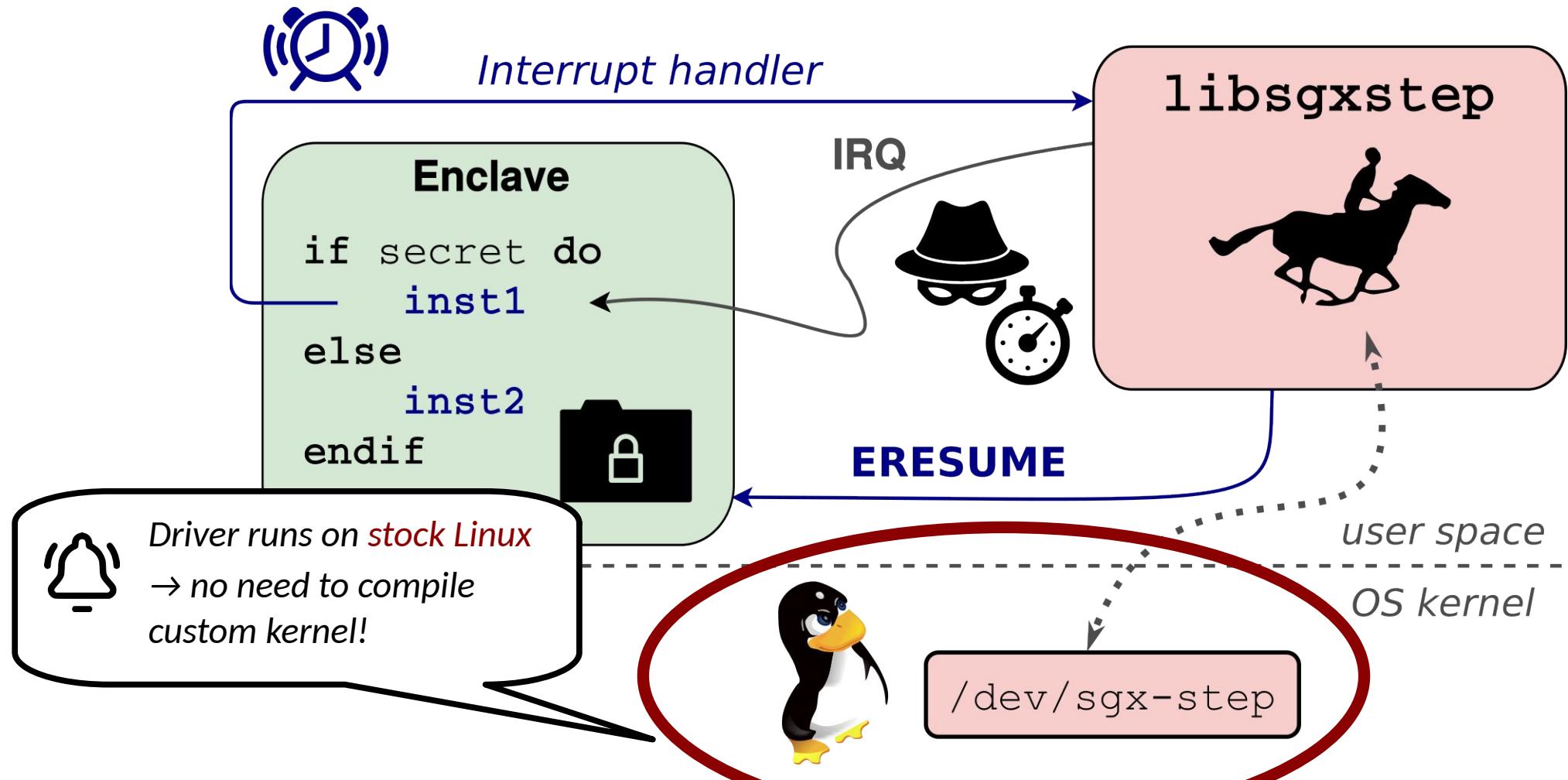
## **Key #1: Accessible Library Design**

---

# SGX-Step: Extensible Linux Kernel Framework



# SGX-Step: Extensible Linux Kernel Framework





## Key #2: Reusable Primitives

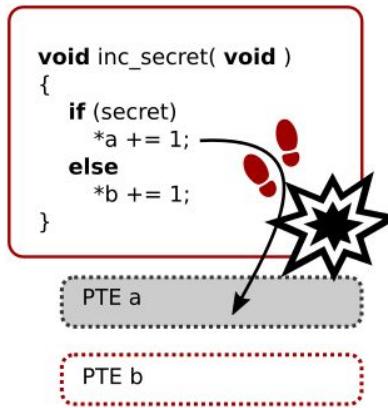
---

# SGX-Step: Enabling a New Line of High-Resolution Attacks

Yr	Venue	Paper	Step	Use Case	Drv
'15	S&P	Ctrl channel	~ Page	Probe (page fault)	✓
'16	ESORICS	AsyncShock	~ Page	Exploit (mem safety)	-
'17	CHES	CacheZoom	X >1	Probe (L1 cache)	✓
'17	ATC	Hahnel et al.	X 0 - >1	Probe (L1 cache)	✓
'17	USENIX	BranchShadow	X 5 - 50	Probe (BPU)	X
'17	USENIX	Stealthy PTE	~ Page	Probe (page table)	✓
'17	USENIX	DarkROP	~ Page	Exploit (mem safety)	✓
'17	SysTEX	SGX-Step	✓ 0 - 1	Framework	✓
'18	ESSoS	Off-limits	✓ 0 - 1	Probe (segmentation)	✓
'18	AsiaCCS	Single-trace RSA	~ Page	Probe (page fault)	✓
'18	USENIX	Foreshadow	✓ 0 - 1	Probe (transient exec)	✓
'18	EuroS&P	SgxPectre	~ Page	Exploit (transient)	✓
'18	CHES	CacheQuote	X >1	Probe (L1 cache)	✓
'18	ICCD	SGXlinger	X >1	Probe (IRQ latency)	X
'18	CCS	Nemesis	✓ 1	Probe (IRQ latency)	✓
'19	USENIX	Spoiler	✓ 1	Probe (IRQ latency)	✓
'19	CCS	ZombieLoad	✓ 0 - 1	Probe (transient exec)	✓
'19	CCS	Fallout	-	Probe (transient exec)	✓
'19	CCS	Tale of 2 worlds	✓ 1	Exploit (mem safety)	✓
'19	ISCA	MicroScope	~ 0 - Page	Framework	X
'20	CHES	Bluethunder	✓ 1	Probe (BPU)	✓
'20	USENIX	Big troubles	~ Page	Probe (page fault)	✓
'20	S&P	Plundervolt	-	Exploit (undervolt)	✓
'20	CHES	Viral primitive	✓ 1	Probe (IRQ count)	✓
'20	USENIX	CopyCat	✓ 1	Probe (IRQ count)	✓
'20	S&P	LVI	✓ 1	Exploit (transient)	✓

Yr	Venue	Paper	Step	Use Case	Drv
'20	CHES	A to Z	~ Page	Probe (page fault)	✓
'20	CCS	Déjà Vu NSS	~ Page	Probe (page fault)	✓
'20	MICRO	PTHammer	-	Probe (page walk)	✓
'21	USENIX	Frontal	✓ 1	Probe (IRQ latency)	✓
'21	S&P	CrossTalk	✓ 1	Probe (transient exec)	✓
'21	CHES	Online template	✓ 1	Probe (IRQ count)	✓
'21	NDSS	SpeechMiner	-	Framework	✓
'21	S&P	Platypus	✓ 0 - 1	Probe (voltage)	✓
'21	DIMVA	Aion	✓ 1	Probe (cache)	✓
'21	CCS	SmashEx	✓ 1	Exploit (mem safety)	✓
'21	CCS	Util::Lookup	✓ 1	Probe (L3 cache)	✓
'22	USENIX	Rapid prototyping	✓ 1	Framework	✓
'22	CT-RSA	Kalyna expansion	✓ 1	Probe (L3 cache)	✓
'22	SEED	Enclzyer	-	Framework	✓
'22	NordSec	Self-monitoring	~ Page	Defense (detect)	✓
'22	AutoSec	Robotic vehicles	✓ 1 - >1	Exploit (timestamp)	✓
'22	ACSAC	MoLE	✓ 1	Defense (randomize)	✓
'22	USENIX	AEPIC	✓ 1	Probe (I/O device)	✓
'22	arXiv	Confidential code	✓ 1	Probe (IRQ latency)	✓
'23	ComSec	FaultMorse	~ Page	Probe (page fault)	✓
'23	CHES	HQC timing	✓ 1	Probe (L3 cache)	✓
'23	ISCA	Belong to us	✓ 1	Probe (BPU)	✓
'23	USENIX	BunnyHop	✓ 1	Probe (BPU)	✓
'23	USENIX	DownFall	✓ 0 - 1	Probe (transient exec)	✓
'23	USENIX	AEX-Notify	✓ 1	Defense (prefetch)	✓

# SGX-Step: A Versatile Open-Source Attack Framework



Page-table manipulation



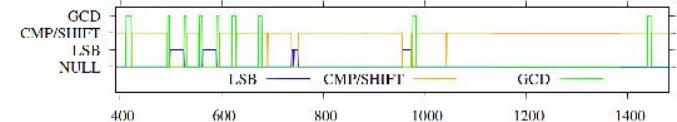
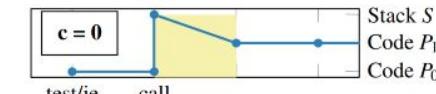
Interrupt latency



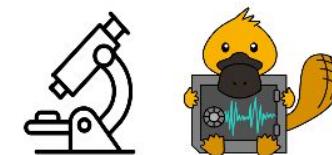
SGX-Step



High-resolution probing



Interrupt counting



Zero-step replaying

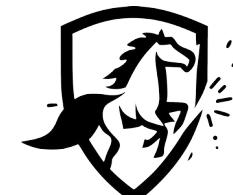
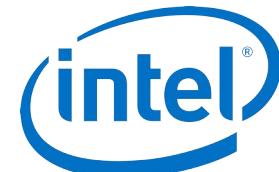


## Key #3: Engage with Industry

---

# Mitigating SGX-Step: Hardware-Software Co-Design

- 2017: **SGX-Step** academic paper and open-source release  
→ *de-facto standard attack framework; used in > 45 international papers*
- 2018-2020: High-impact attacks **Foreshadow**, **ZombieLoad**, **LVI**, etc.  
→ *coverage in (inter)national press; widespread patches in processors and operating systems*
- 2023: **AEX-Notify** hardware extension for software-based mitigations  
→ *collaboration with Intel Research; shipped in millions of Intel CPUs worldwide*
- 2023: ACSAC'23 Cybersecurity Artifacts Impact Award
- 2025: **TLBlur** improved compiler mitigation



## CHAPTER 8

ASYNCHRONOUS ENCLAVE EXIT NOTIFY AND THE EDECCSSA USER LEAF FUNCTION

---

## 8.1 INTRODUCTION

Asynchronous Enclave Exit Notify (AEX-Notify) is an extension to Intel® SGX that allows Intel SGX enclaves to be notified after an asynchronous enclave exit (AEX) has occurred. EDECCSSA is a new Intel SGX user leaf function (ENCLU[EDECCSSA]) that can facilitate AEX notification handling, as well as software exception handling. This chapter provides information about changes to the Intel SGX architecture that support AEX-Notify and ENCLU[EDECCSSA].

The following list summarizes the a details are provided in Section 8.3)

- SECS.ATTRIBUTES.AEXNOTIFY
- TCS.FLAGS.AEXNOTIFY: This en
- SSA.GPRSGX.AEXNOTIFY: Enclave-writable byte that allows enclave software to dynamically enable/disable AEX notifications.



**SGX-Step led to new x86 processor instructions!**

→ shipped in millions of devices  $\geq$  4th Gen Xeon CPU

An AEX notification is delivered by ENCLU[ERESUME] when the following conditions are met:

## Intel AEX Notify Support Prepped For Linux To Help Enhance SGX Enclave Security

Written by Michael Larabel in Intel on 6 November 2022 at 06:01 AM EST. [5 Comments](#)



Future Intel CPUs and some existing processors via a microcode update will support a new feature called the Asynchronous EXit (AEX) notification mechanism to help with Software Guard Extensions (SGX) enclave security. Patches for the Linux kernel are pending for implementing this Intel AEX Notify support with capable processors.

Intel's Asynchronous EXit (AEX) notification mechanism lets SGX enclaves run a handler after an AEX event. Those handlers can be used for things like mitigating SGX-Step as an attack framework for precise enclave execution control.



intel sdk/trts/linux/trts\_mitigation.S

```
48     * Description:  
49     *      The file provides mitigations for SGX-Step  
50     */  
51     * Function:  
52     *      constant_time_apply_sgxstep_mitigation_and_continue_execution  
53     *      Mitigate SGX-Step and return to the point at which the  
most recent  
73     *      interrupt/exception occurred.
```



SGX-Step led to changes in major OSs and enclave SDKs

# Conclusion: 7 Magic Ingredients

- 1) Open-source ecosystem
- 2) Modular base design
- 3) Impact through education
- 4) Science communication
- 5) Accessible library design
- 6) Reusable primitives
- 7) Engage with industry

