bare-sgx: A Bare-Metal C Runtime for Intel SGX Development with Minimal Trust

Jo Van Bulck, Kobe Sauwens

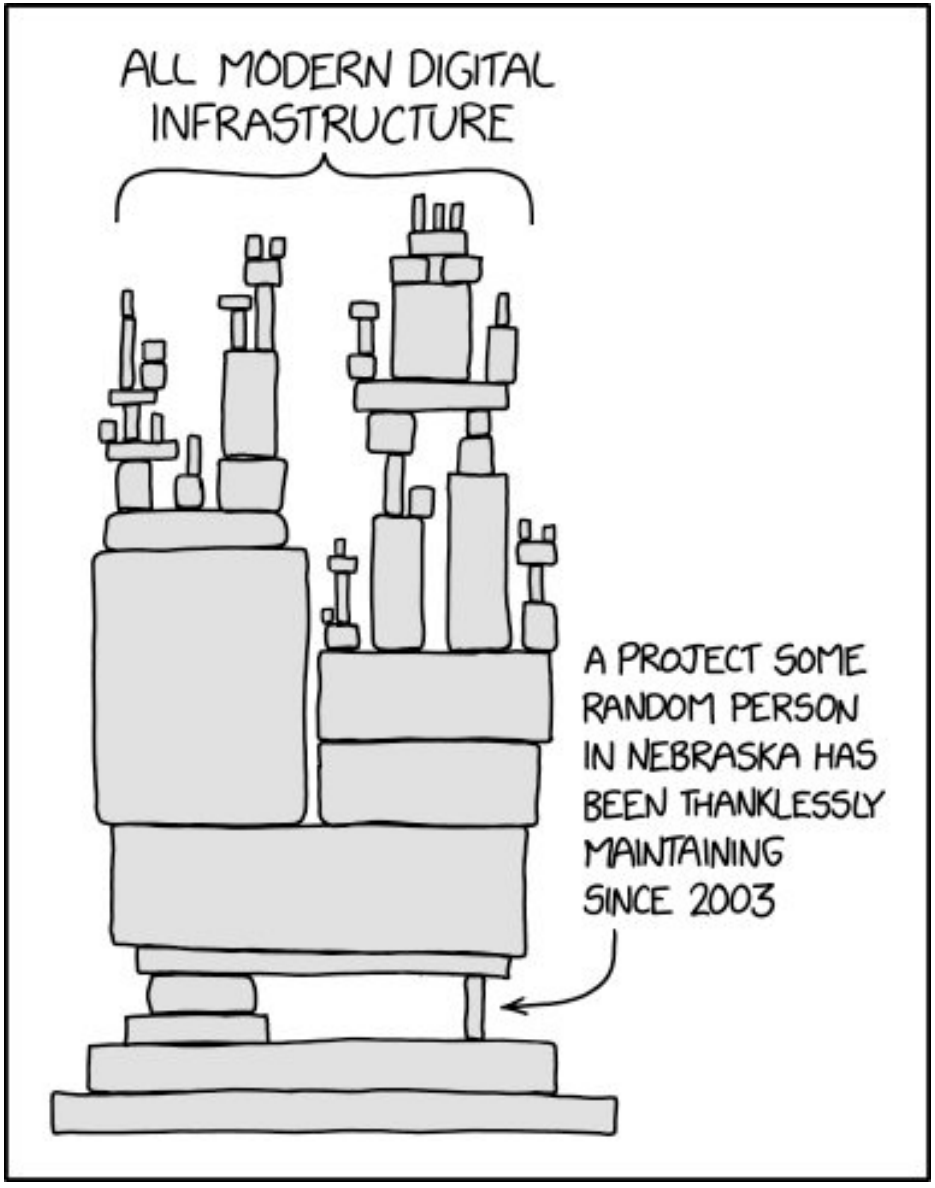
DistriNet, KU Leuven, Belgium jo.vanbulck@cs.kuleuven.be vanbulck.net

FOSDEM'26 Confidential Computing Devroom, Feb 1, 2026

DistriNet

KU LEUVEN

Trust?

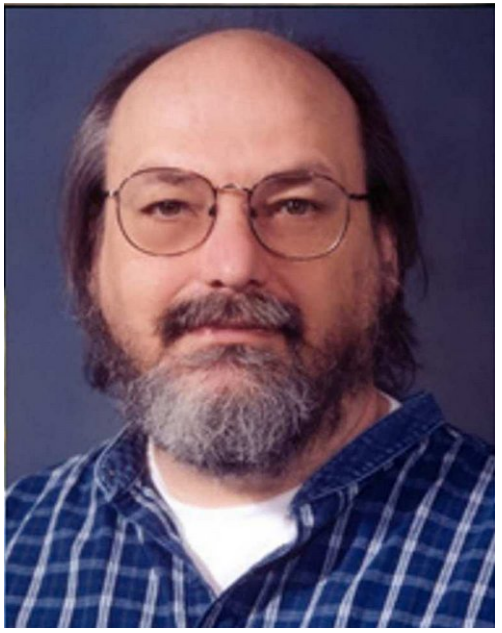


https://xkcd.com/2347/

Reflections on Trusting Trust...

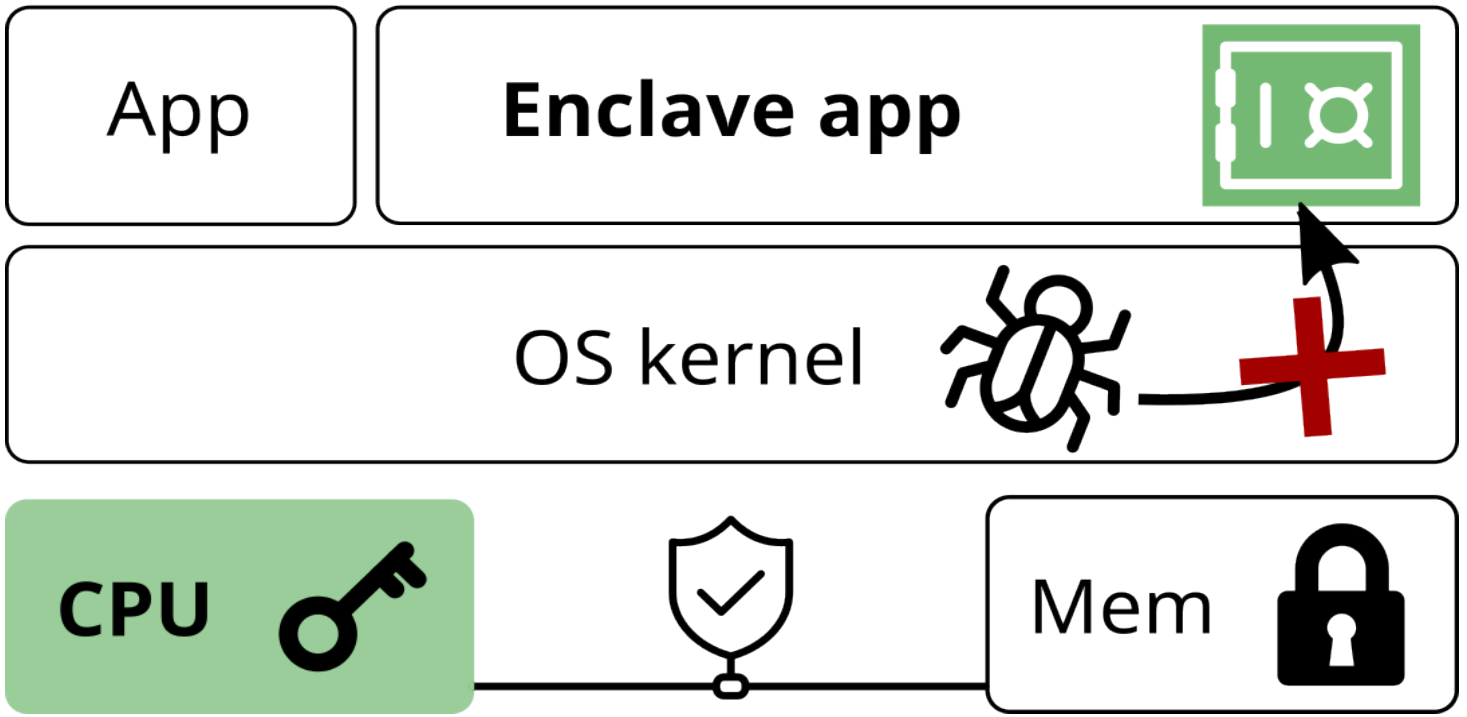
*"The moral is obvious. You can't trust code that you did not **totally create yourself**."*

— Ken Thompson, Turing Award Lecture 1984



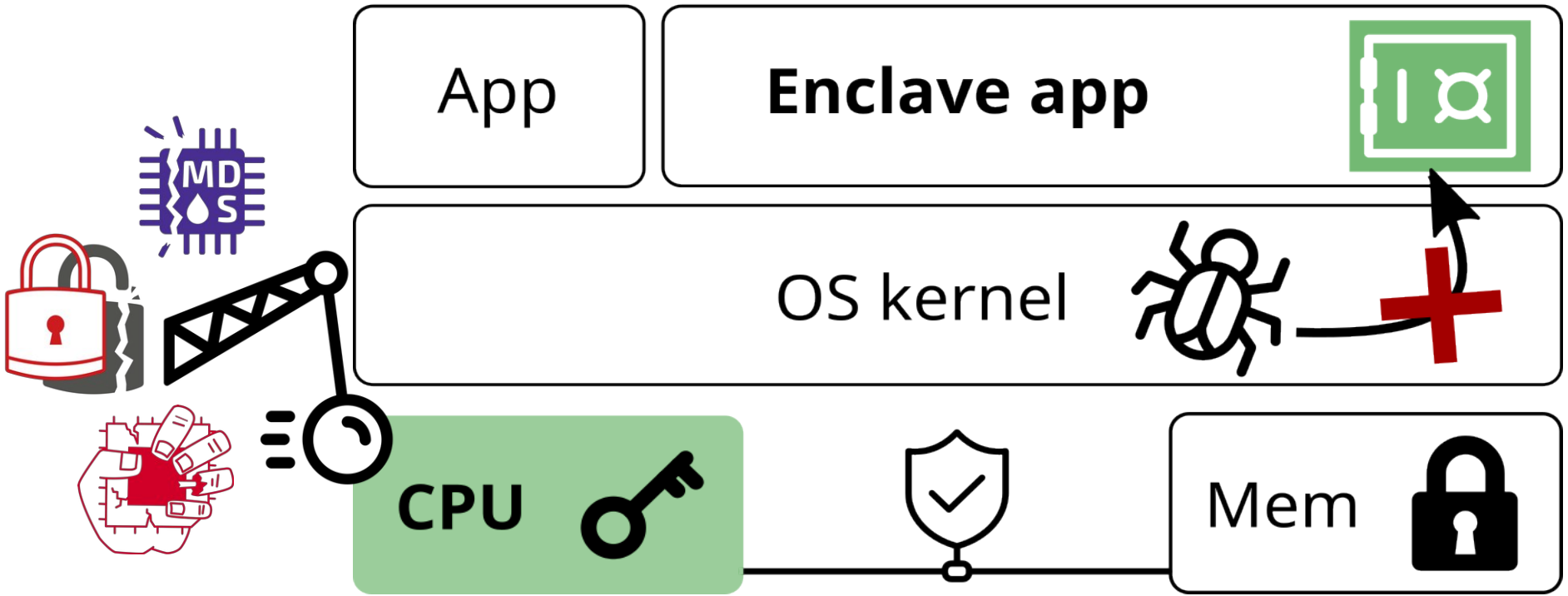
https://dl.acm.org/doi/abs/10.1145/358198.358210

Reflections on Trusting Trusted Execution Environments?



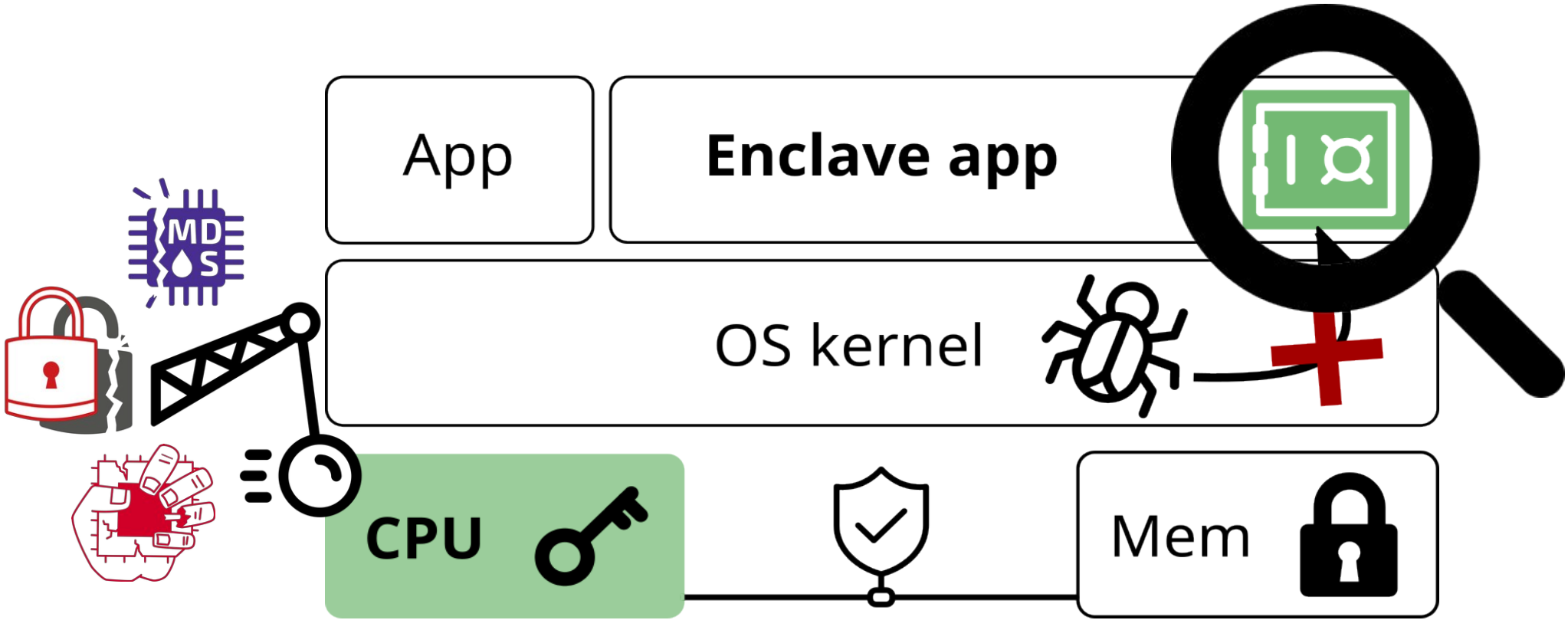
TEE promise: Hardware-level **isolation and attestation**

Reflections on Trusting Trusted Execution Environments?



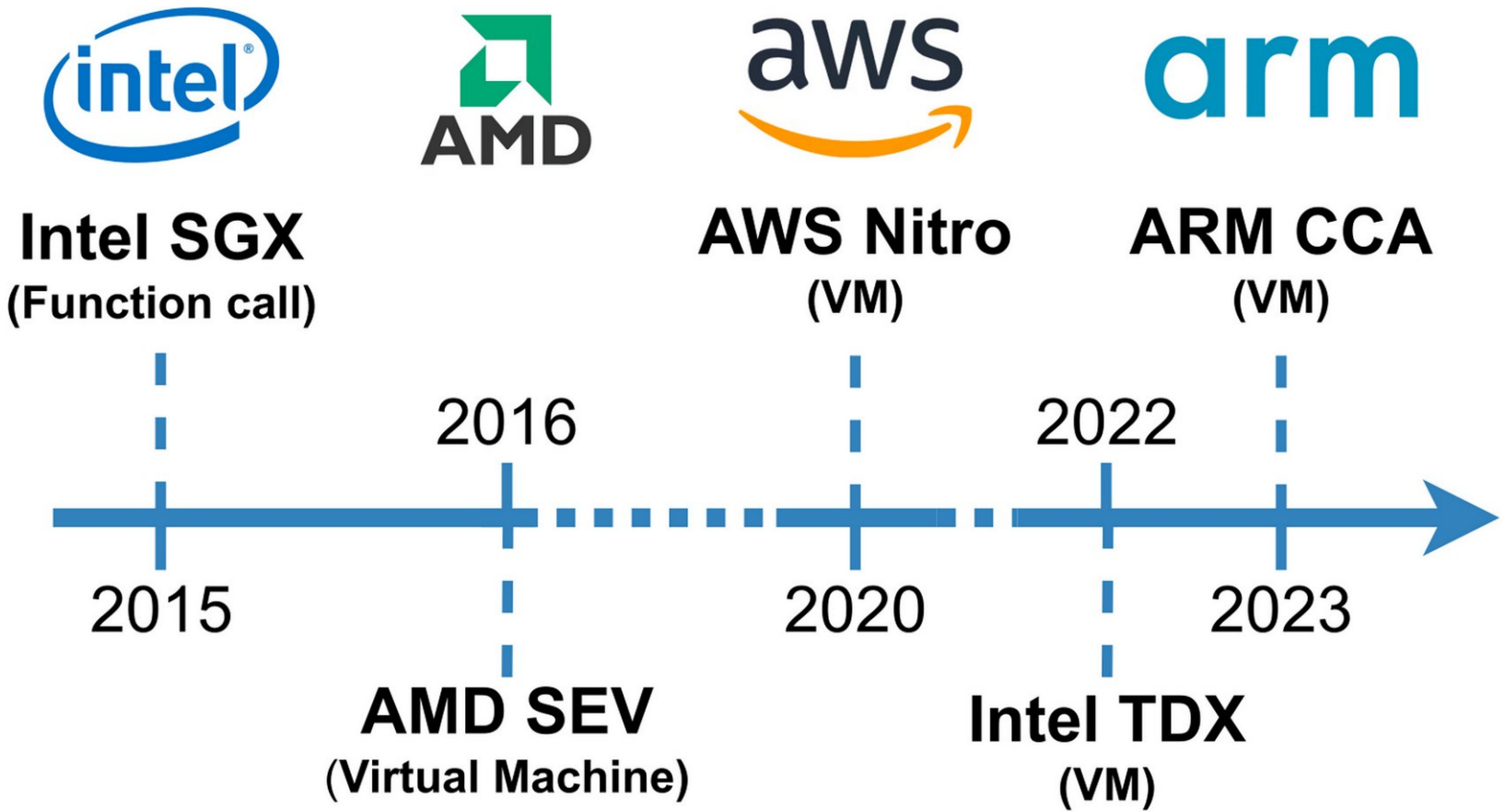
CPU vulnerabilities: Microarchitectural reality (not today)

Reflections on Trusting Trusted Execution Environments?



Software attack surface: Enclave trusted computing base(!)

TEE Evolution: Towards Coarse-Grained Lift and Shift



Intel SGX Promise: Minimal Trusted Computing Base...

intel

Overview | About Intel | News & Events | Financial Info | Stock Info | Filings & Reports | Board & Governance | ESG

Overview | Press Releases | IR Calendar | Annual Stockholders' Meeting | Investor Meeting | Email Alerts | Presentations

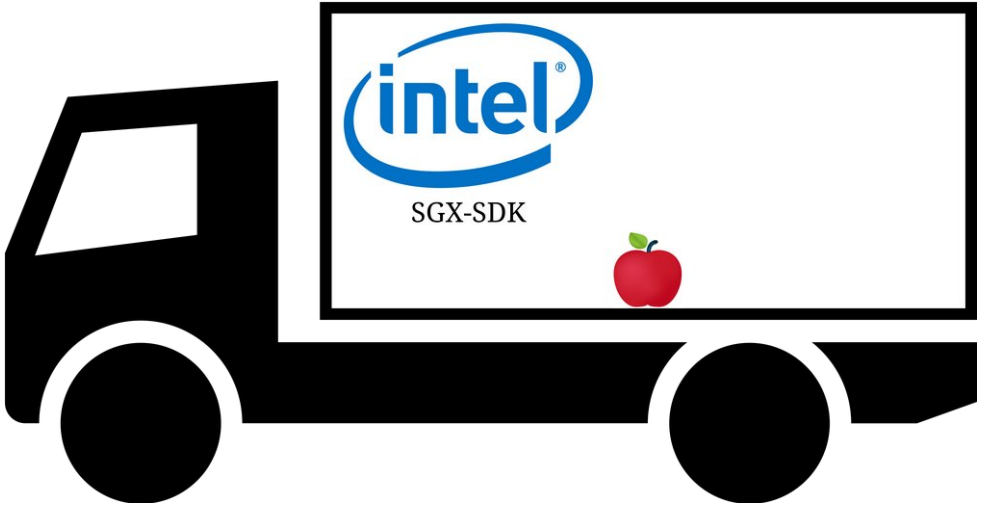
Data Protection across the Compute Stack

Technologies such as disk- and network-traffic encryption protect data in storage and during transmission, but data can be vulnerable to interception and tampering while in use in memory. "Confidential computing" is a rapidly emerging usage category that protects data while it is in use in a Trusted Execution Environment (TEE). Intel SGX is the most researched, updated and battle-tested TEE for data center confidential computing, with the smallest attack surface within the system. It enables application isolation in private memory regions, called enclaves, to help protect up to 1 terabyte of code and data while in use.

https://www.intc.com/news-events/press-releases/detail/1423/intel-xeon-scalable-platform-built-for-most-sensitive

Intel SGX Reality: Open-Source Enclave SDK Ecosystem

```
jo@aeolus:~/sgx-step/sdk/intel-sdk/linux-sgx/sdk$ echo ; sloccount . 2>&1 | grep "Source Lines of Code"

Total Physical Source Lines of Code (SLOC)                = 222,681

jo@aeolus:~/sgx-step/sdk/intel-sdk/linux-sgx/sdk$ cd -
/home/jo/sgx-step/sdk/oe/openenclave
jo@aeolus:~/sgx-step/sdk/oe/openenclave$ echo ; sloccount . 2>&1 | grep "Source Lines of Code" ; echo

Total Physical Source Lines of Code (SLOC)                = 199,412
```



Manual code review finds 35 vulnerabilities in 8 enclave SDKs

All issues have been privately reported and patches are available.

Written by **Catalin Cimpanu,** Contributor

Nov. 12, 2019 at 10:00 a.m. PT

https://archive.fosdem.org/2020/schedule/event/tale/

https://www.zdnet.com/article/manual-code-review-finds-35-vulnerabilities-in-8-enclave-sdks/

ZDNET tomorrow belongs to those who embrace it today

Home / Tech / Security

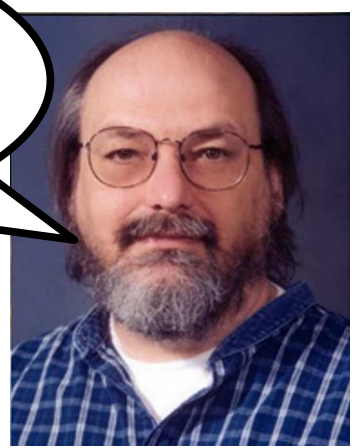
Manual code review finds 35 vulnerabilities in 8 enclave SDKs

All issues have been privately reported

Written by **Catalin Cimpanu,** Contributor

Nov. 12, 2019 at 10:00 a.m. PT

You can't trust code that you did not totally create yourself...



https://archive.fosdem.org/2020/schedule/event/tale/

https://www.zdnet.com/article/manual-code-review-finds-35-vulnerabilities-in-8-enclave-sdks/

ZDNET tomorrow belongs to those who embrace it today

Home / Tech / See

Man[illegible]5 vulnerabilities in 8 enclave S[illegible]

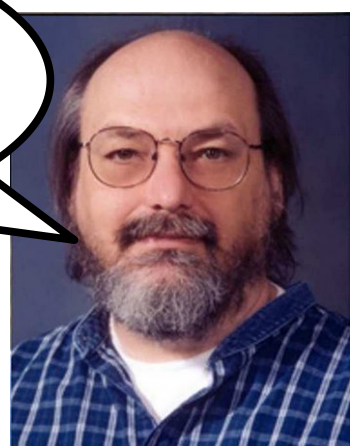
All issues have b[illegible]n privately reported

[illegible]ten by **Catalin Cimpanu,** Contributor

[illegible]12, 2019 at 10:00 a.m. PT

Can we build enclaves we totally create ourselves?

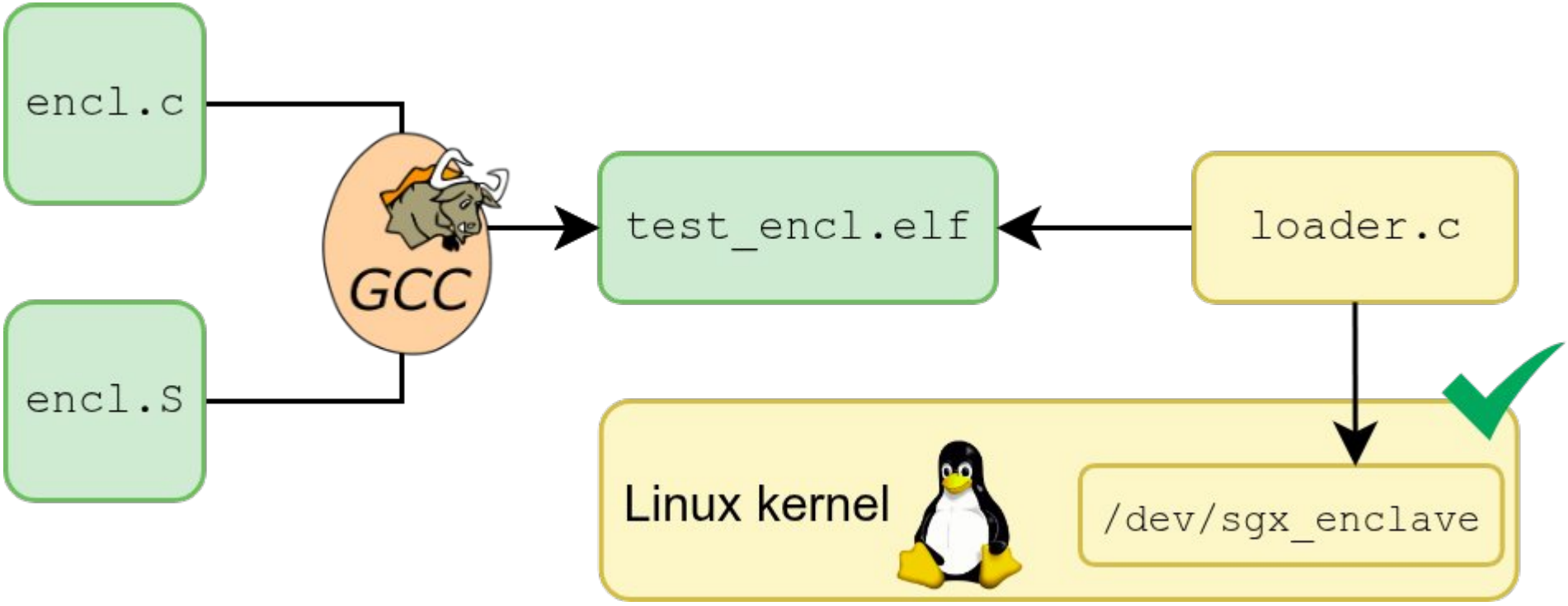
You can't trust code that you did not totally create yourself...



https://archive.fosdem.org/2020/schedule/event/tale/

https://www.zdnet.com/article/manual-code-review-finds-35-vulnerabilities-in-8-enclave-sdks/

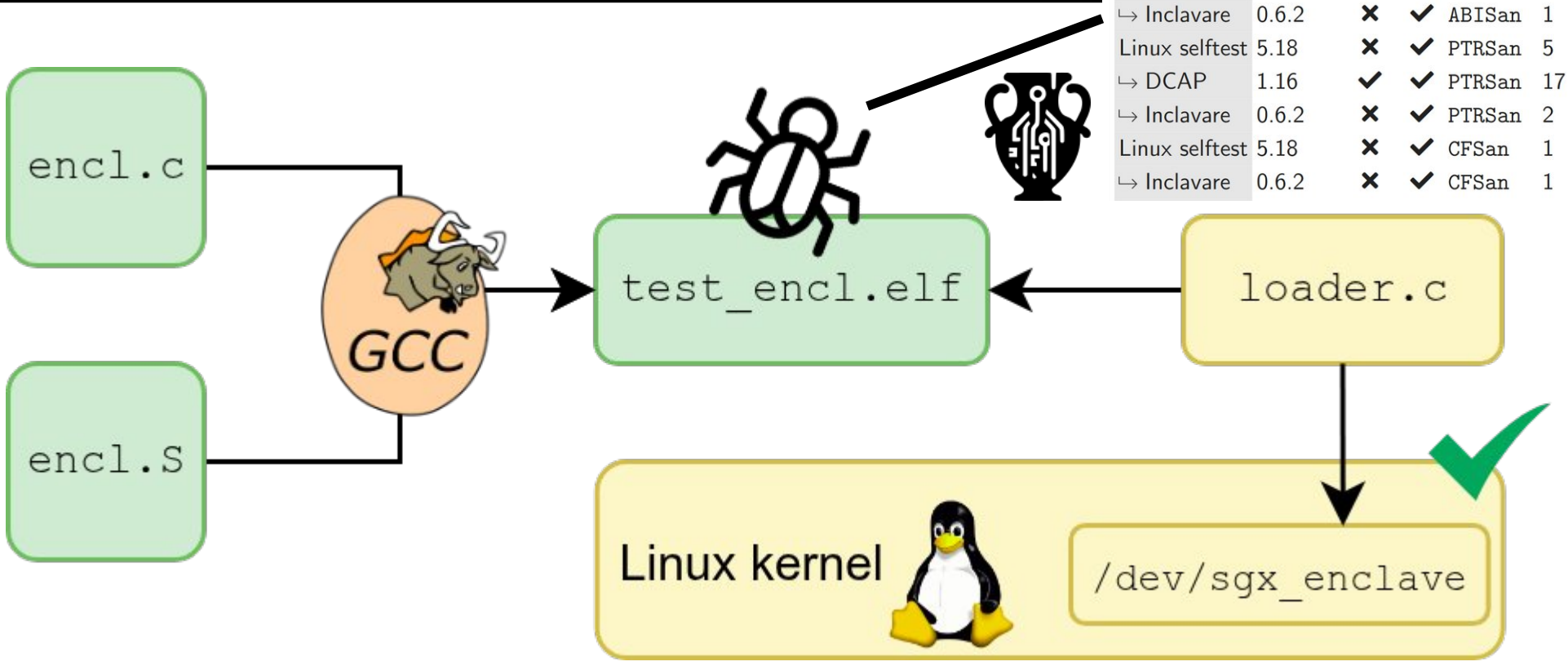
Starting Point: Linux `selftests/sgx`



https://github.com/torvalds/linux/tree/master/tools/testing/selftests/sgx

Starting Point: Linux `selftests/sgx`

Runtime	Version	Prod	Src	Plugin	Instances
Linux selftest	5.18	✗	✓	ABISan	1
↪ DCAP	1.16	✓	✓	ABISan	1
↪ Inclavare	0.6.2	✗	✓	ABISan	1
Linux selftest	5.18	✗	✓	PTRSan	5
↪ DCAP	1.16	✓	✓	PTRSan	17
↪ Inclavare	0.6.2	✗	✓	PTRSan	2
Linux selftest	5.18	✗	✓	CFSan	1
↪ Inclavare	0.6.2	✗	✓	CFSan	1



https://vanbulck.net/files/oakland24-pandora.pdf

Starting Point: Linux `selftests/sgx`

From	Dave Hansen <dave.hansen@intel.com>
To	Jo Van Bulck, jarkko@kernel.org, linux-sgx@vger.kernel.org, linux-kernel@vger.kernel.org
Cc	dave.hansen@linux.intel.com
Subject	**Re: [PATCH v2 0/4] selftests/sgx: Harden test enclave**

On 7/20/23 15:16, Jo Van Bulck wrote:

> While I understand that the bare-metal Intel SGX selftest enclave is certainly not intended as a full-featured independent production runtime, it has been noted on this mailing list before that "people are likely to copy this code for their own enclaves" and that it provides a "great starting point if you want to do things from scratch" [1].

I wholeheartedly agree with the desire to spin up enclaves without the overhead or complexity of the SDK. I think I'm the one that asked for this test enclave in the first place. There *IS* a gap here. Those who care about SGX would be wise to close this gap in _some_ way.

https://lore.kernel.org/all/da0cfb1e-e347-f7f2-ac72-aec0ee0d867d@intel.com/

Minimal SGX Enclave Development on Bare-Metal Linux Platforms

This repository provides a minimal, fully customizable framework for developing Intel SGX enclaves directly on bare-metal Linux, without relying on bloated external SDKs. It offers a clean, low-level starting point for building minimalist enclaves in assembly or C, interfacing directly with the upstream Linux SGX driver.

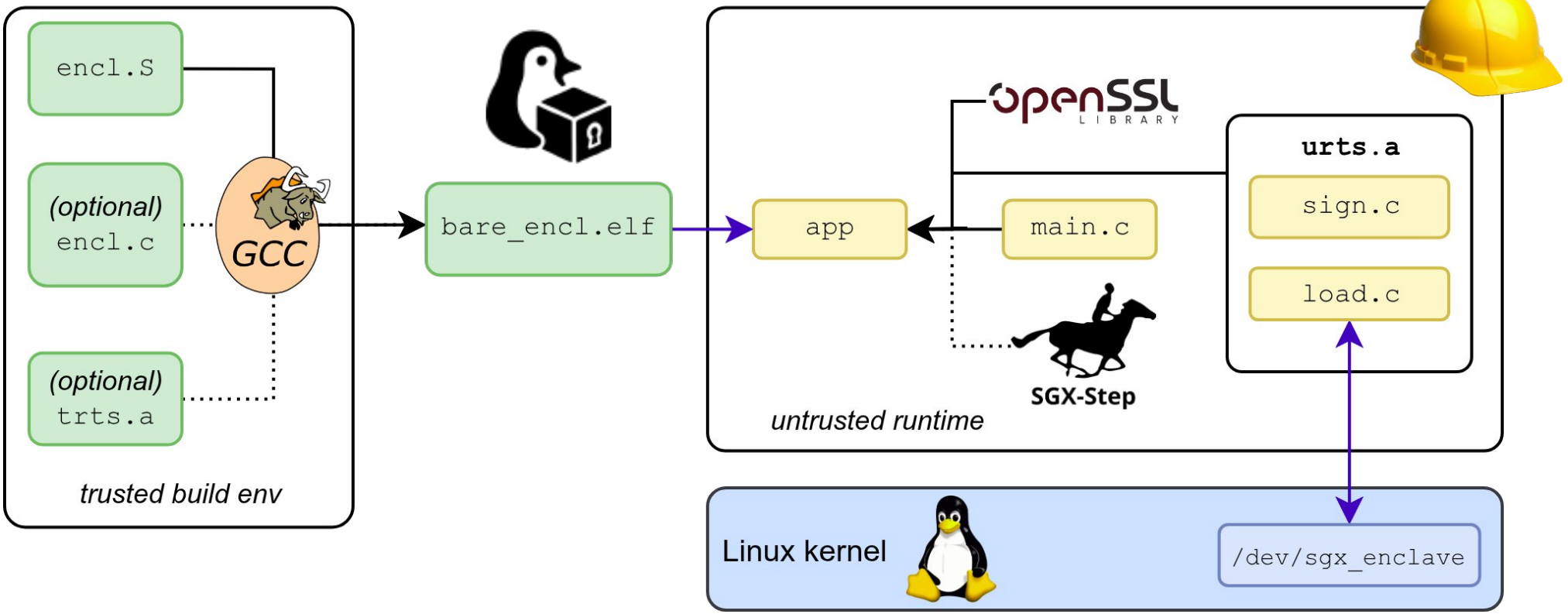
By interacting directly with the SGX driver in the Linux kernel, `bare-sgx` removes the complexity and overhead of existing SGX SDKs and library OSs. The result is extremely small enclaves, often just a few pages, tailored to a specific purpose and excluding *all* other unnecessary code and features. Therefore, `bare-sgx` provides a truly minimal trusted computing base while avoiding fragile dependencies that could hinder portability or long-term reproducibility.

License. `bare-sgx` is free software, licensed under GPLv2. The initial code was forked from the selftests/sgx test enclave in the Linux kernel repository, following a discussion on the kernel mailing list.

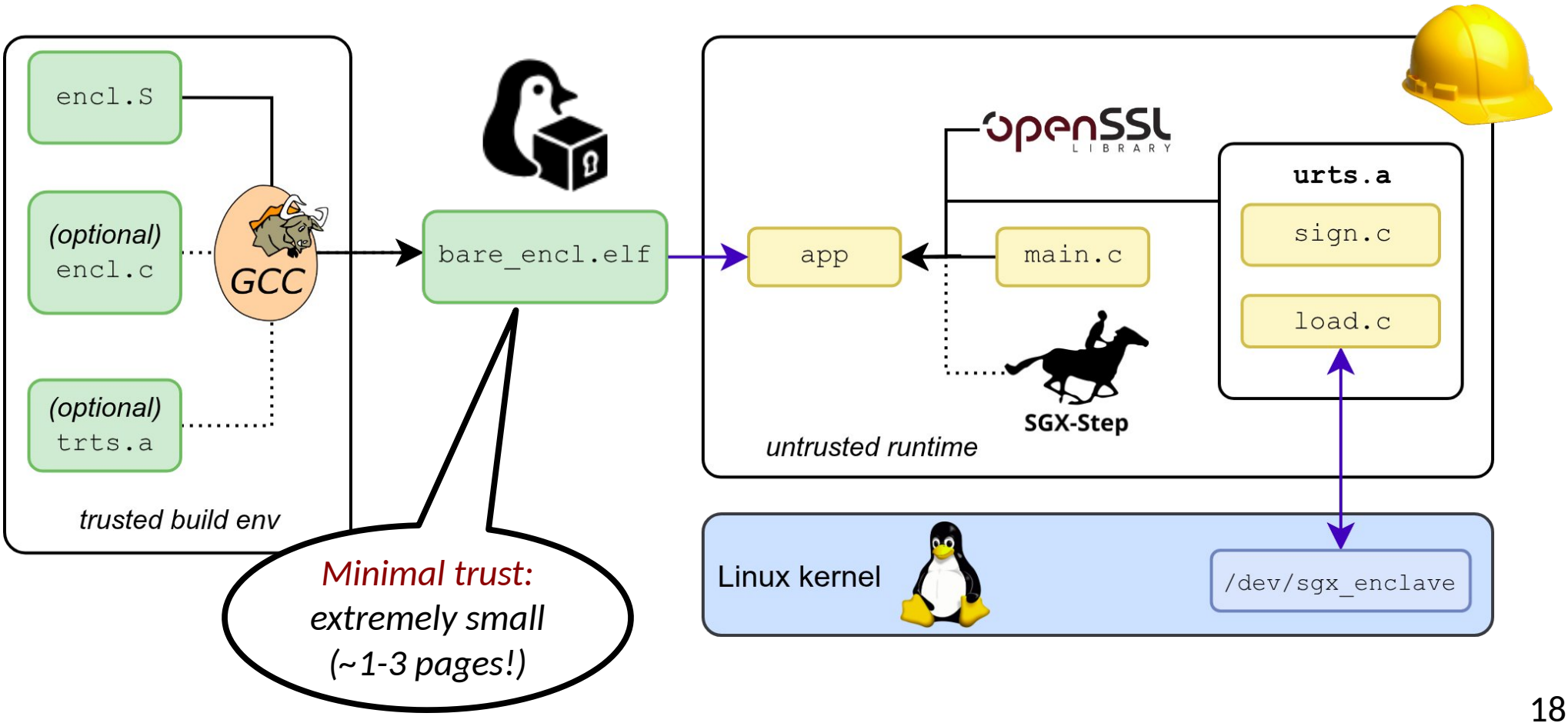
https://github.com/jovanbulck/bare-sgx

bare-sgx: Untrusted Runtime Features

Rapid prototyping +
long-term packaging
(buildroot)



bare-sgx: Untrusted Runtime Features

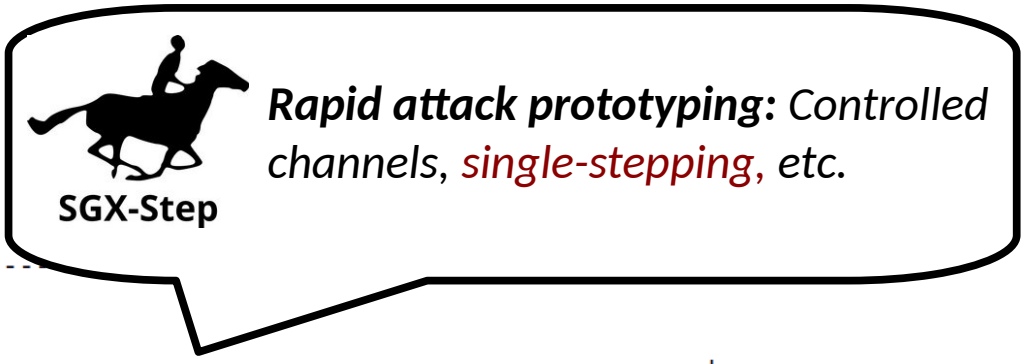


Mini buildroot VM: *Stable long-term packaging of artifacts & PoCs without bloated/fragile SDK dependencies*

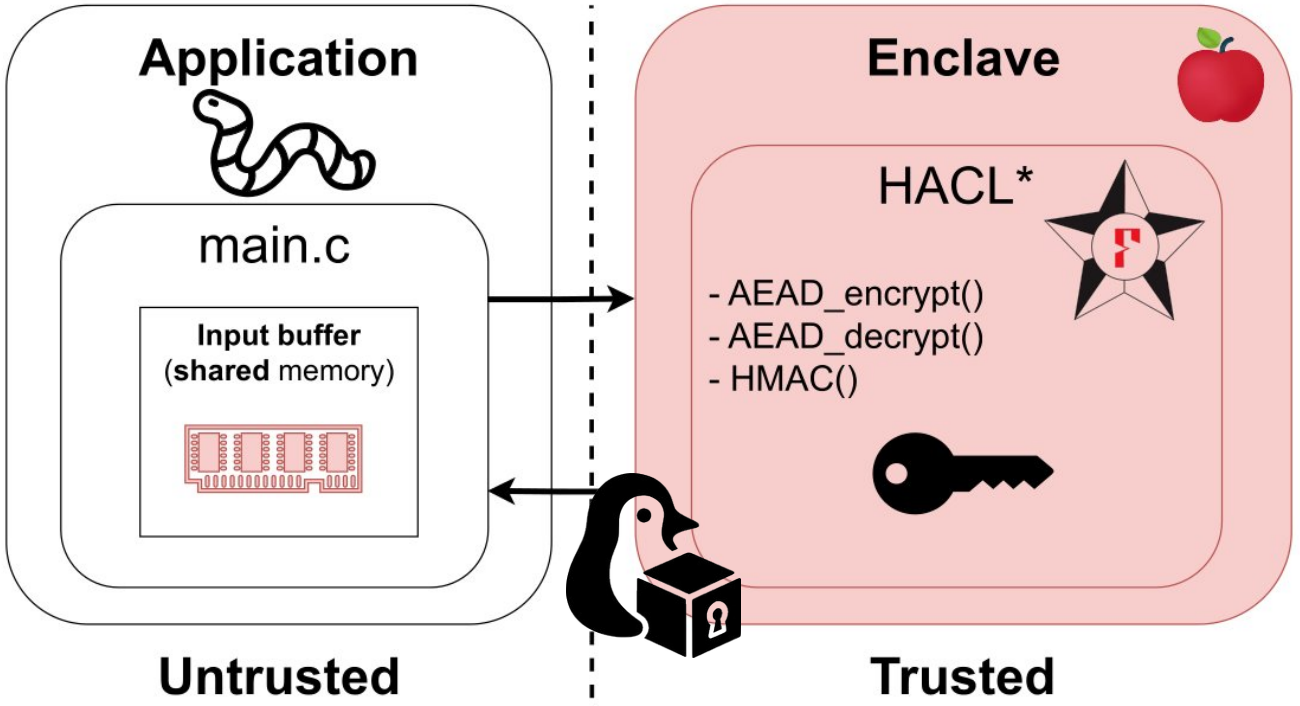
```
    __
   / /_  ____ _________        _________ __  __
  / __ \/ __ `/ ___/ _ \______/ ___/ __ `/ |/_/
 / /_/ / /_/ / /  /  __/_____(__  ) /_/ />  <
/_.___/\__,_/_/   \___/     /____/\__, /_/|_|
                                 /____/
```

```
Welcome to minimal Linux!
    L_ Login as user root (no password required)
    L_ Run bare-sgx programs (compiled on host) as cd /host/ecall_asm && ./app
    L_ Exit qemu with CTRL-A followed by X
buildroot login: root
# cd /host/ecall_asm/ && ./app
[main.c] loaded enclave at 0x7f253b378000
[main.c] reading enclave memory..
        L mem at 0x7f253b378000 is ffffffffffffff
[main.c] calling enclave TCS..
        L enclave returned deadbeefcafebabe
#
```

```
jo@aeolus:~/sgx-step/app/baresgx$ sudo ./app

------------------------------------------------------------------------------
[main.c] loading baresgx enclave
------------------------------------------------------------------------------

==== Victim Enclave ====
[enclave.c] tcs at 7fb30ebf0000; aep at 5571f12ee85a
    Driver: /dev/sgx_enclave
    Base:   0x7fb30ebf0000
    Limit:  0x7fb30ebf5000
    Size:   20480
    Exec:   1 pages
    TCS:    0x7fb30ebf0000
    SSA:    0x7fb30ebf1f48
    AEP:    0x5571f12ee85a
    EDBGRD: debug
[main.c] dry run
        L enclave returned deadbeefcafebabe

------------------------------------------------------
[main.c] configuring attacker runtime
------------------------------------------------------

[main.c] entry page at 0x7fb30ebf1000
+-----------------------------------------------------------------------------------------------+
| XD | PK | IGN | RSVD | PHYS ADRS      | IGN | G | PAT | D | A | PCD | PWT | U/S | R/W | P |
| 0  | x  | x   | 0    | 0x0040787b1000 | x   | x | x   | 0 | 1 | x   | x   | 1   | 0   | 1 |
+-----------------------------------------------------------------------------------------------+
[main.c] APIC timer IRQ handler seems to be working

------------------------------------------------------------------------------
[main.c] single-stepping baresgx enclave
------------------------------------------------------------------------------
```



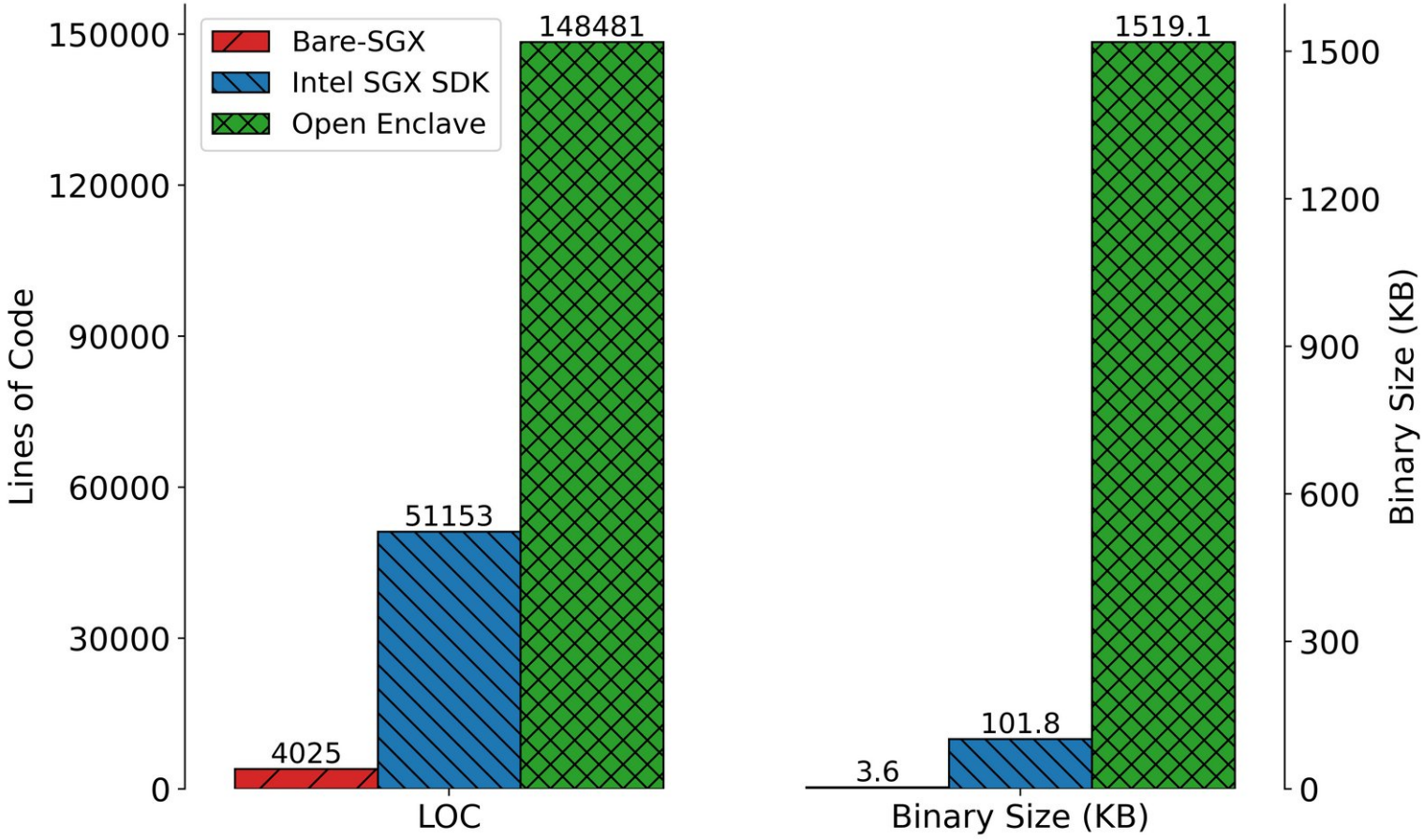
Case Study: High-Assurance, Formally Verified Crypto (HACL*)



Trusted wrapper code: Generated with `edger8r` + mini `libc`/heap (FreeRTOS)

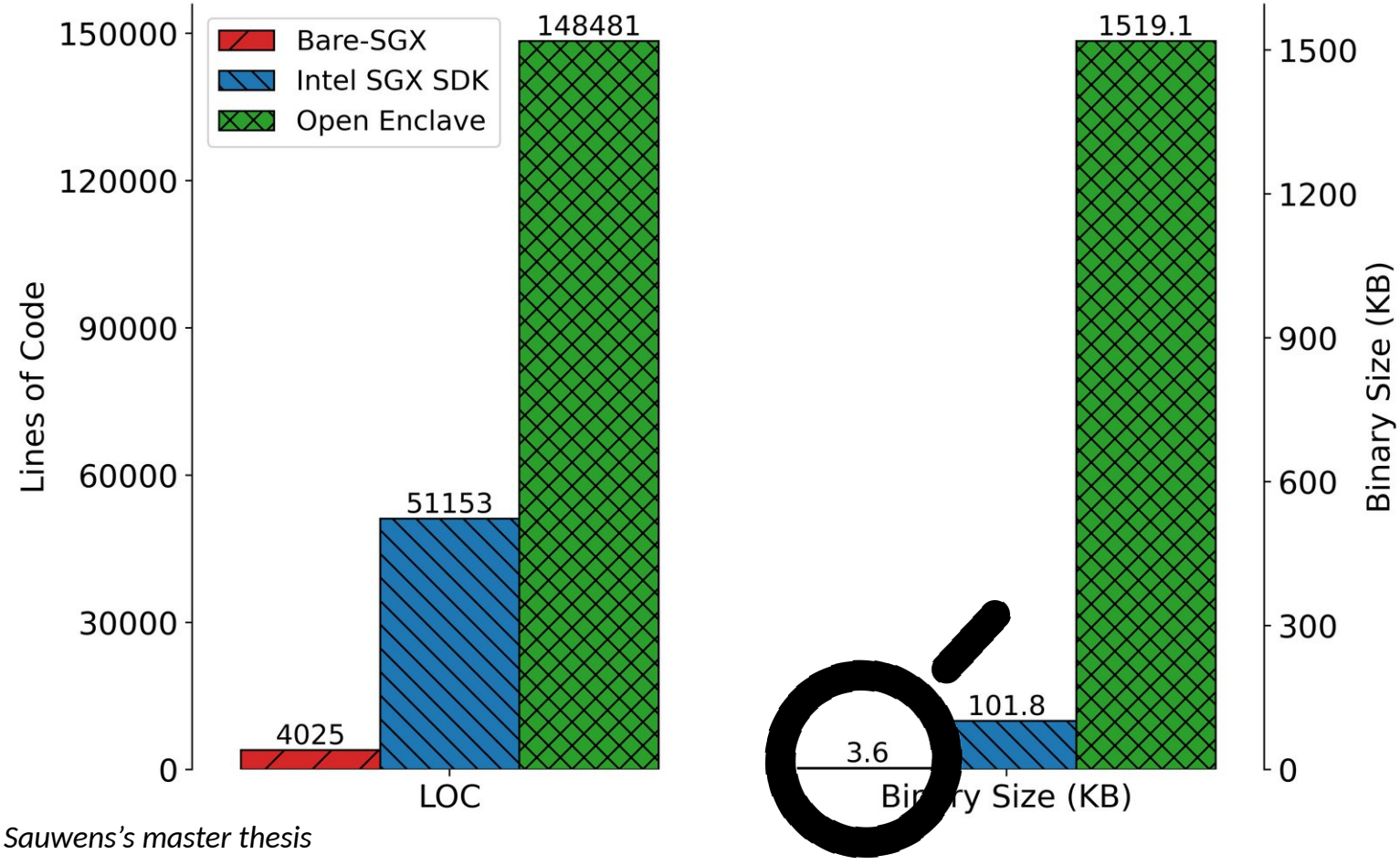
Results from Kobe Sauwens's master thesis

TCB Size Evaluation: Lines of Code and Binary Size



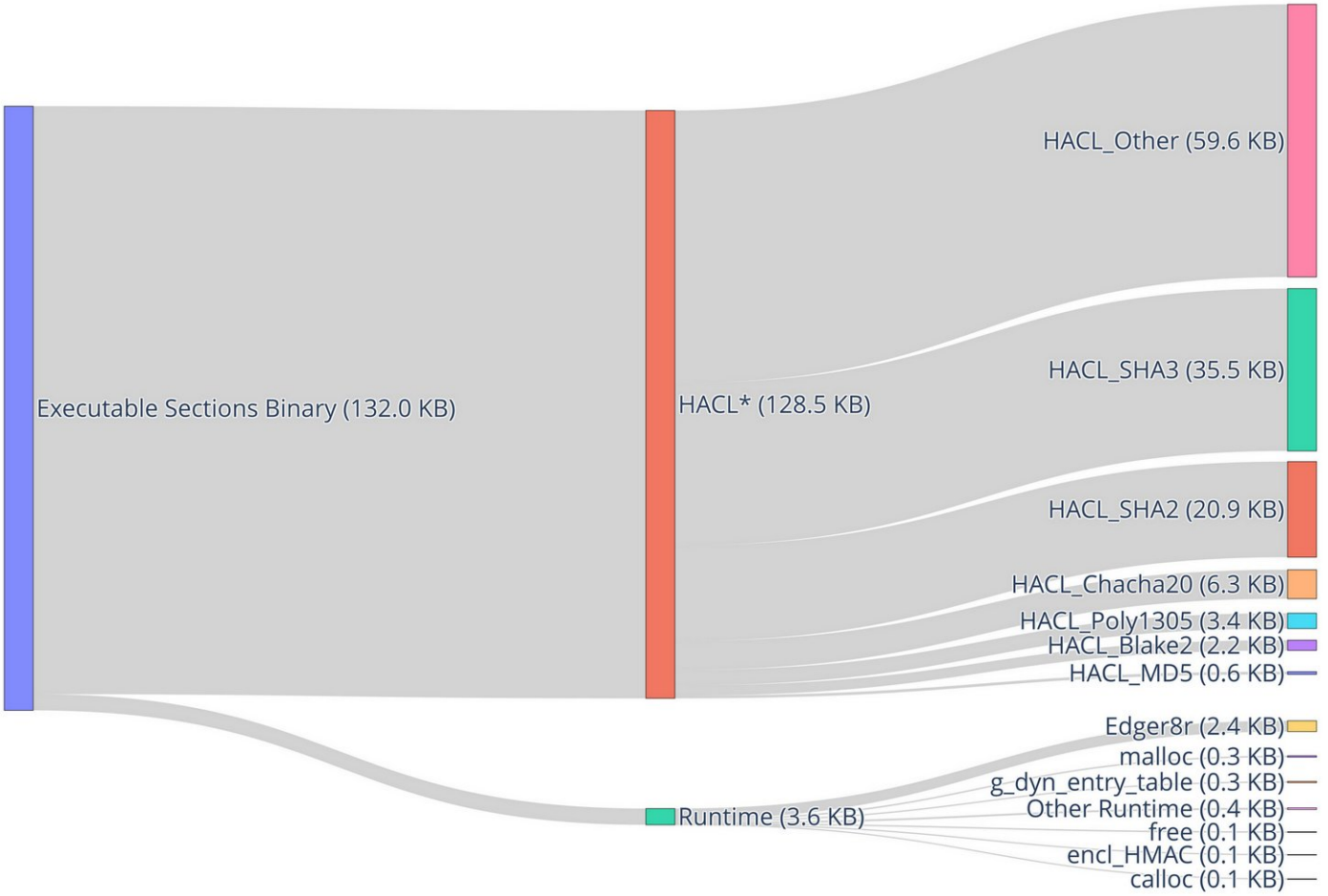
Results from Kobe Sauwens's master thesis

TCB Size Evaluation: Lines of Code and Binary Size



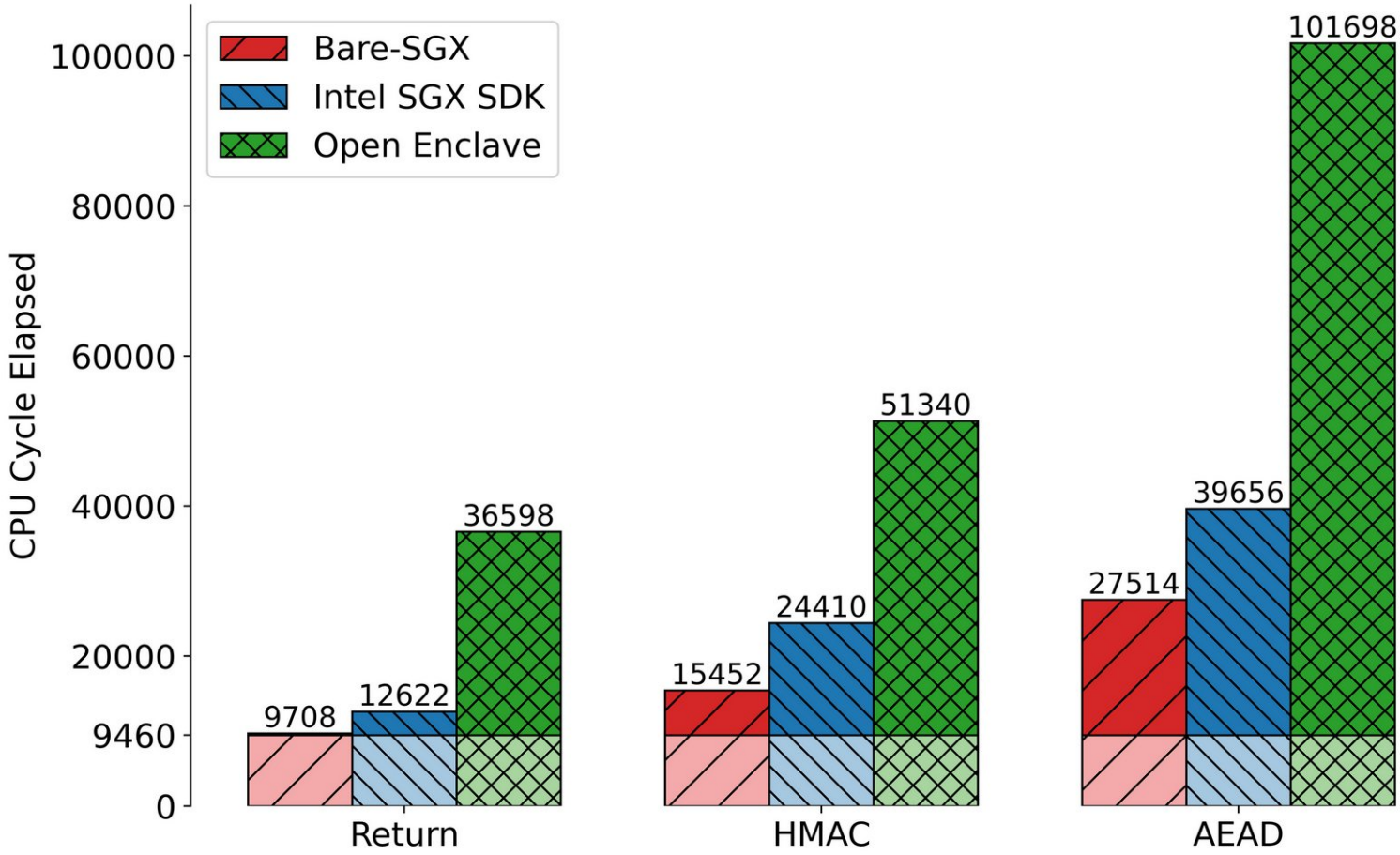
Results from Kobe Sauwens's master thesis

Binary Size Analysis: `bare-sgx` + HACL*



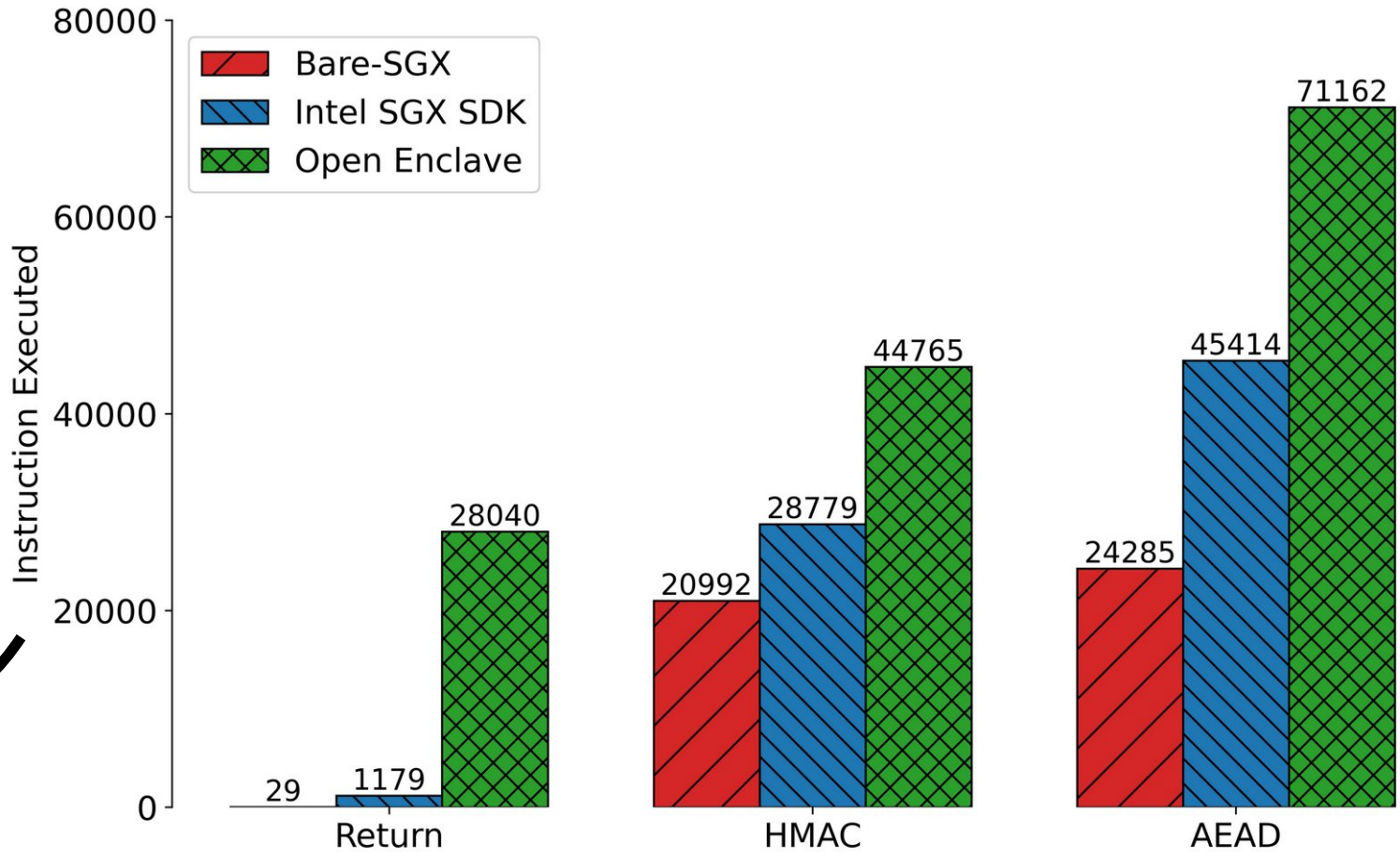
Results from Kobe Sauwens's master thesis

Performance Evaluation: CPU Cycles



Results from Kobe Sauwens's master thesis

Performance Evaluation: Number of Executed Instructions



Results from Kobe Sauwens's master thesis

Conclusions and Take-Away

https://github.com/jovanbulck/bare-sgx

Reality: Intel SGX SDK ecosystem bloated and vulnerable

bare-sgx: Truly minimal-trust, specialized enclave development

Use cases: Formal verification; long-term packaging; testing; …

Thank you! Questions?

https://distrinet.cs.kuleuven.be/jobs/