

Intel SGX interface vulnerabilities

and how to automatically detect them

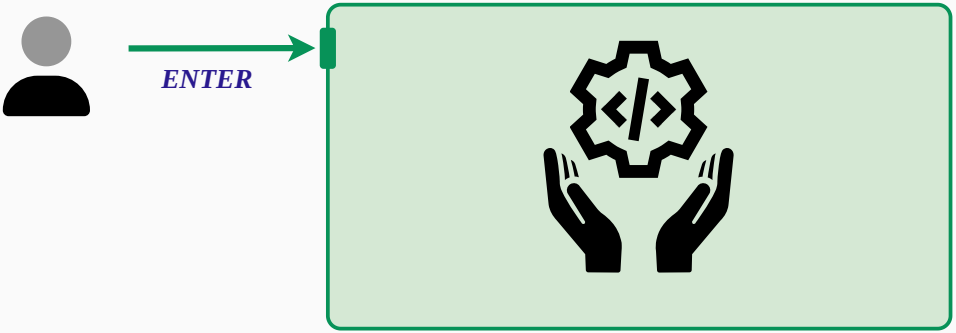
Fritz Alder

June 15, 2023

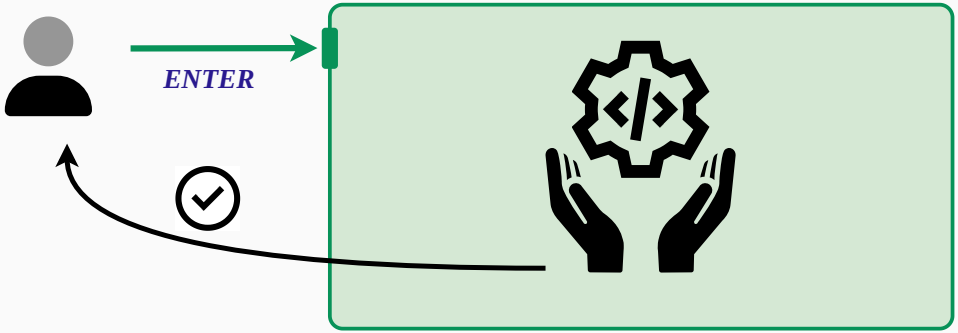
Workshop on Security, Privacy & Verifiable Computing for contemporary distributed systems

DistriNet KU LEUVEN

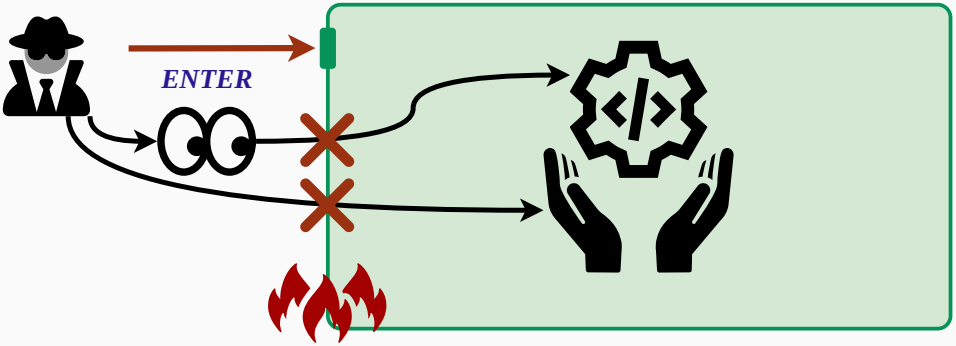
Trusted execution environment interfaces



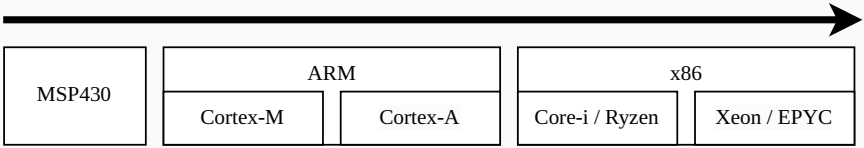
Trusted execution environment interfaces



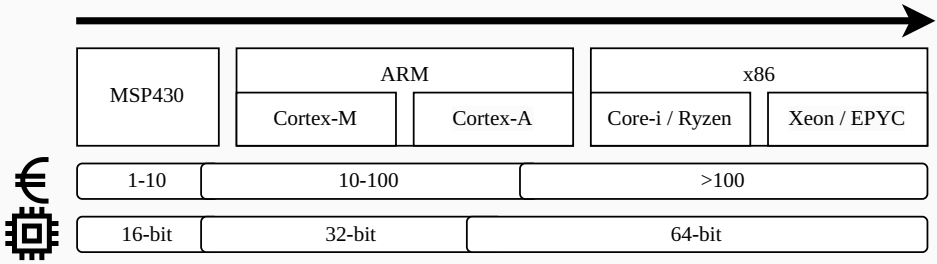
Trusted execution environment interfaces



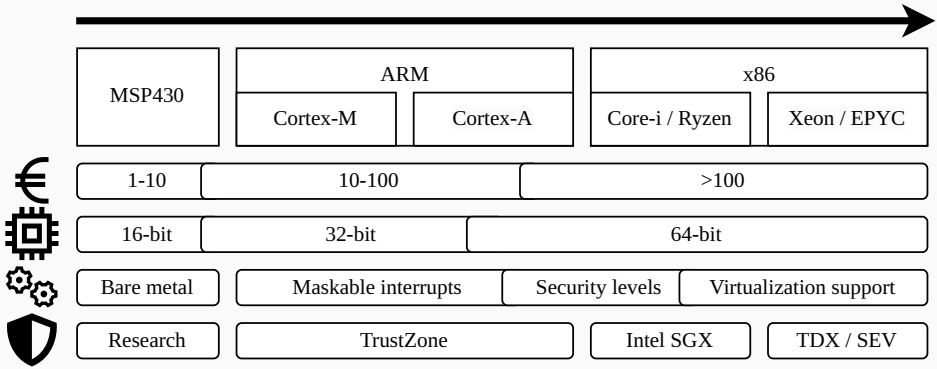
Computing spectrum



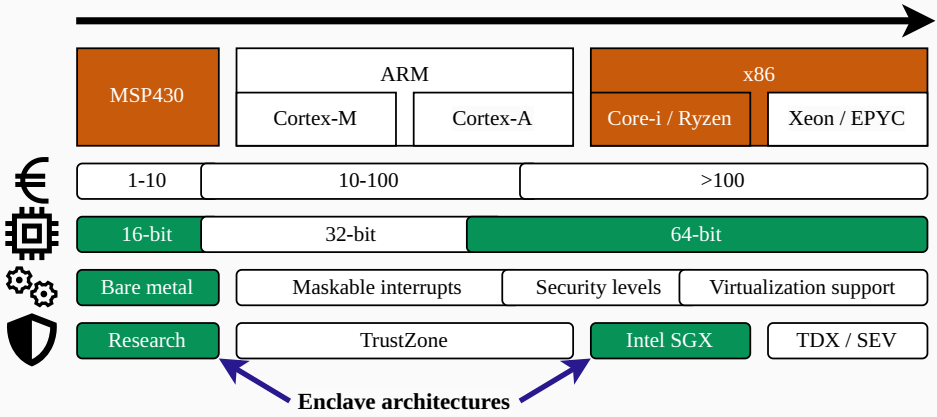
Computing spectrum



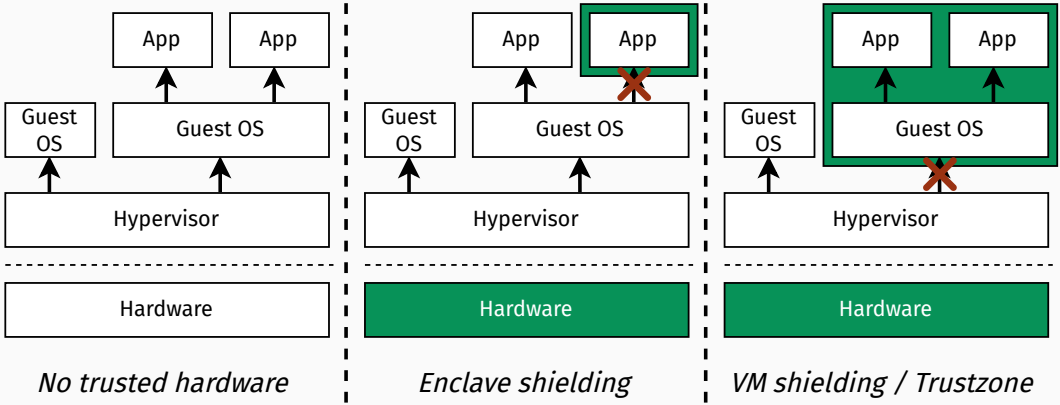
Computing spectrum



Computing spectrum



Trusted execution environment types



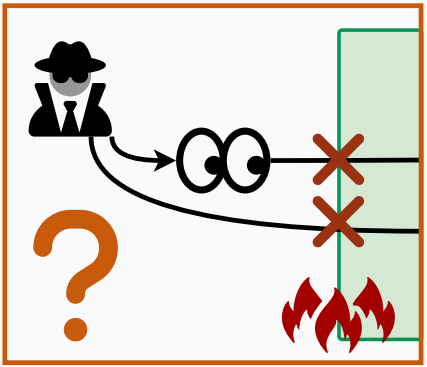
Excurse: Intel SGX attack research @ DistriNet

Van Bulck et al. “Telling Your Secrets Without Page Faults: Stealthy Page Table-Based Attacks on Enclaved Execution”, USENIX Security Symposium, 2017.

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

Van Bulck et al. “Foreshadow: Extracting the Keys to the Intel SGX Kingdom with Transient Out-of-Order Execution”, USENIX Security Symposium, 2018.

Van Bulck et al. “Nemesis: Studying Microarchitectural Timing Leaks in Rudimentary CPU Interrupt Logic ”, CCS, 2018.



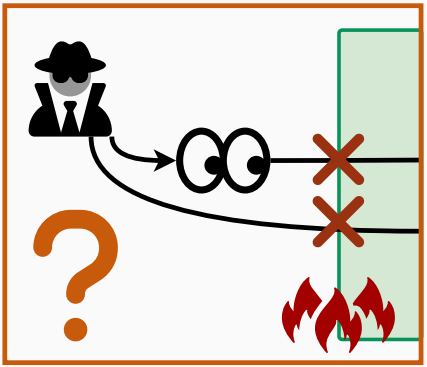
Excurse: Intel SGX attack research @ DistriNet

Canella et al. “Fallout: Leaking Data on Meltdown-resistant CPUs”, CCS, 2019.

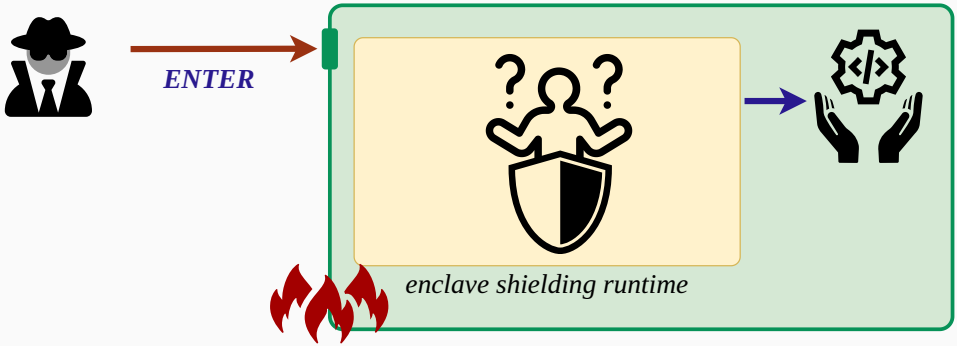
Schwarz et al. “ZombieLoad: Cross-Privilege-Boundary Data Sampling”, CCS, 2019.

Murdock et al. “Plundervolt: Software-Based Fault Injection Attacks Against Intel SGX”, S&P, 2020.

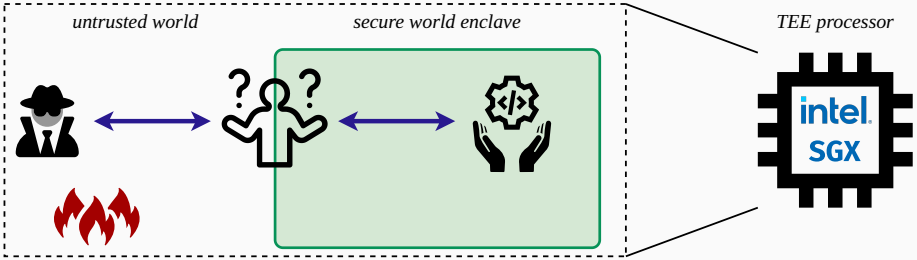
Van Bulck et al. “LVI: Hijacking Transient Execution through Microarchitectural Load Value Injection ”, S&P, 2020.



TEE interface sanitization



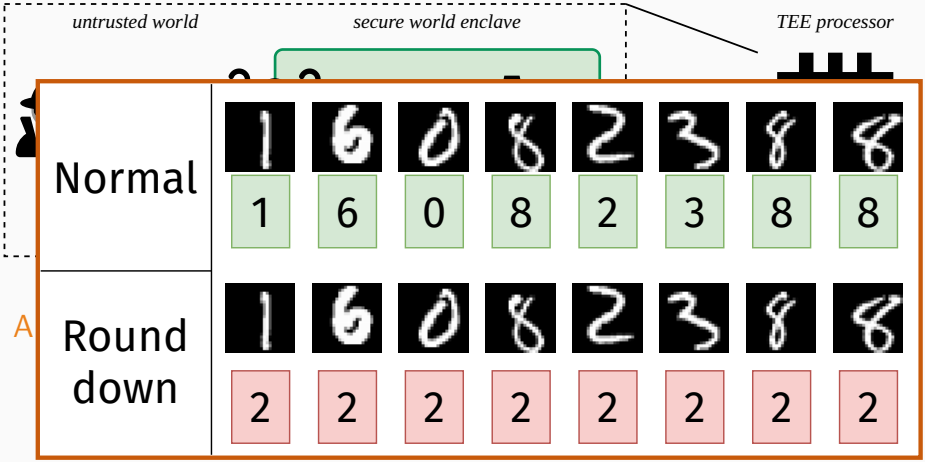
Context: Writing “secure” enclave software is hard . . .



- ABI level: Sanitize low-level CPU configuration registers

Alder et al. “Faulty Point Unit: ABI Poisoning Attacks on Intel SGX”, ACSAC 2020.
Van Bulck et al. “A Case for Unified ABI Shielding in Intel SGX Runtimes”, SysTEX 2022.

Context: Writing “secure” enclave software is hard . . .

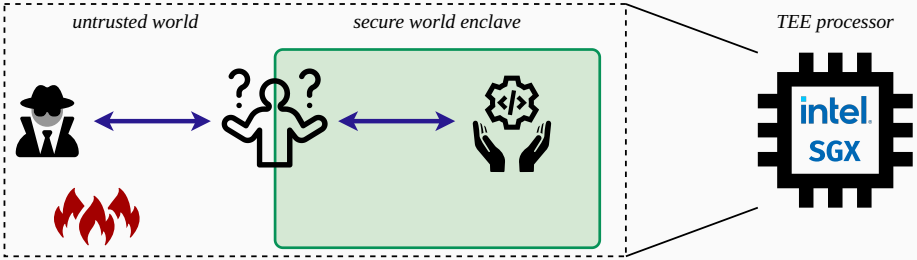


- A

Alder et al. “Faulty Point Unit: ABI Poisoning Attacks on Intel SGX”, ACSAC 2020.

Van Bulck et al. “A Case for Unified ABI Shielding in Intel SGX Runtimes”, SysTEX 2022.

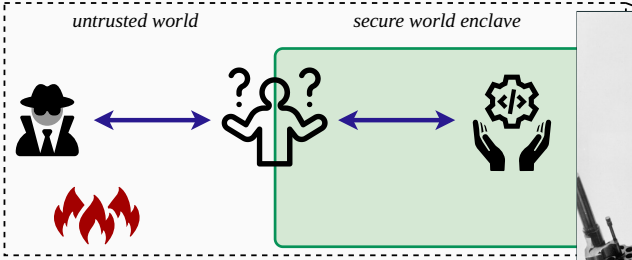
Context: Writing “secure” enclave software is hard . . .



- ABI level: Sanitize low-level CPU configuration registers
- API level: Sanitize pointer arguments in shared address space

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

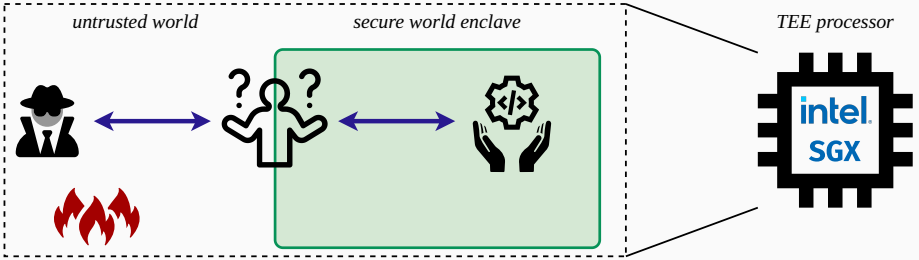
Context: Writing “secure” enclave software is hard . . .



- ABI level: Sanitize low-level CPU configura
- API level: Sanitize pointer arguments in sha

Van Bulck et al. “A Tale of Two Worlds: Assessing the Vulnerability o

Context: Writing “secure” enclave software is hard . . .

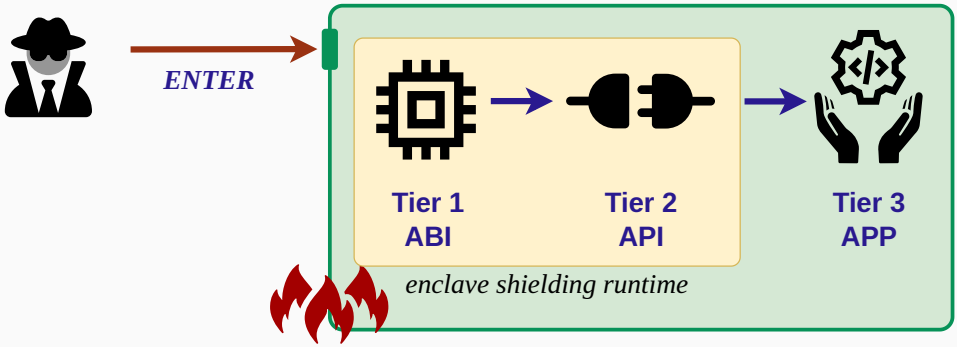


- ABI level: Sanitize low-level CPU configuration registers
- API level: Sanitize pointer arguments in shared address space
- μ-arch level: Spectre/LVI → `lfence`; ÆPIC/MMIO stale data → `verw`; cacheline GPU leak → avoid `dword0/1`...

https://www.intel.com/content/www/us/en/security-center/advisory/intel-sa-00615.html

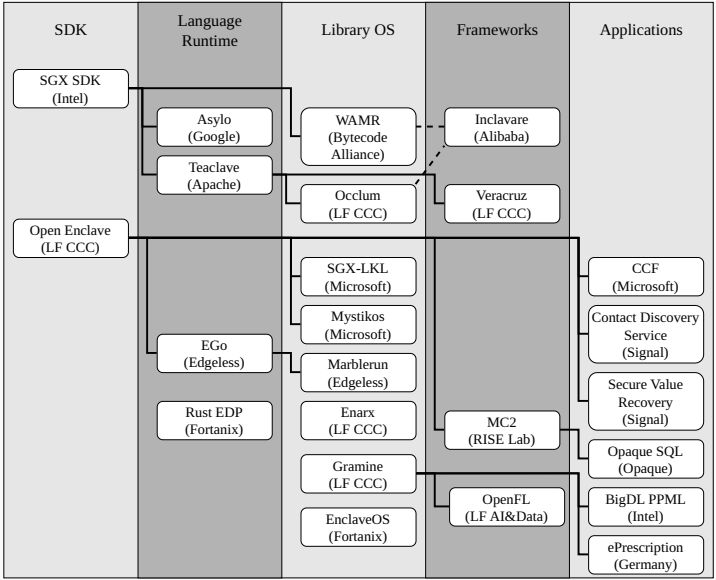
https://www.intel.com/content/www/us/en/security-center/advisory/intel-sa-00219.html

TEE interface sanitization

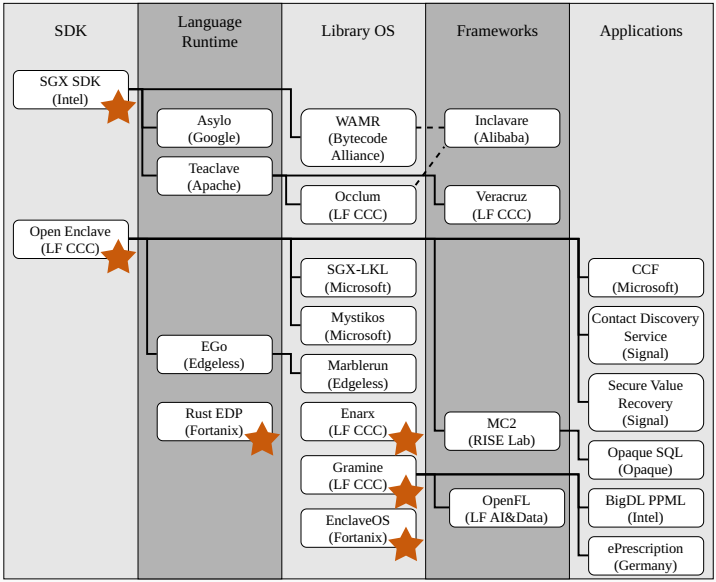


Key insight: split sanitization responsibilities across the ABI and API tiers: *machine state* vs. higher-level *programming language interface*

Intel SGX Ecosystem



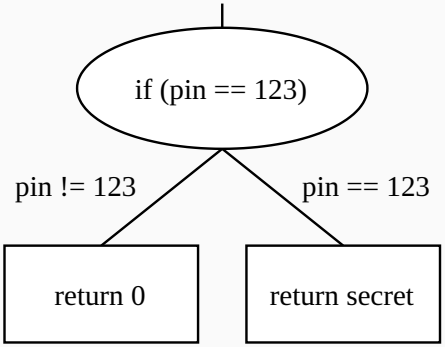
Intel SGX Ecosystem



Symbolic execution and angr

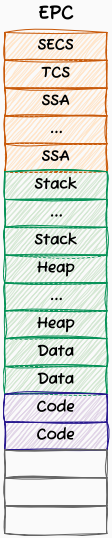
```
1 int ecall(int pin){
2     if(pin == 123){
3         return secret;
4     } else {
5         return 0;
6     }
7 }
```

https://angr.io/



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

SGX binaries are (not) normal binaries



Loading enclaves is different than loading normal `ELF` binaries!

↔ No uniform **enclave** binary format!

- Untrusted runtime loader parses ELF binary embedded metadata to create enclave image with TCS, SSA, Stack, Heap, etc.
- MRENCLAVE independent of load address → partial relocation in enclave

↔ No `syscalls`; untrusted interaction through `enclu` (ecall/ocall/...)