

# Fortress or Facade: Strengthening the Future of Confidential Computing

**Jo Van Bulck**

🏠 [DistriNet, KU Leuven, Belgium](#)    ✉️ [jo.vanbulck@cs.kuleuven.be](mailto:jo.vanbulck@cs.kuleuven.be)    🐦 [@jovanbulck](#)    🌐 [vanbulck.net](http://vanbulck.net)

DRADS, March 10, 2025

# The Big Picture: Protecting Private Data



**Data in transit**



**Data in use**



**Data at rest**

# The Big Picture: Protecting Private Data



**Data in transit**

- ✓ HTTPS etc.



**Data in use**



**Data at rest**

- ✓ Full disk encryption

# The Big Picture: Protecting Private Data



## Data in transit

- ✓ HTTPS etc.



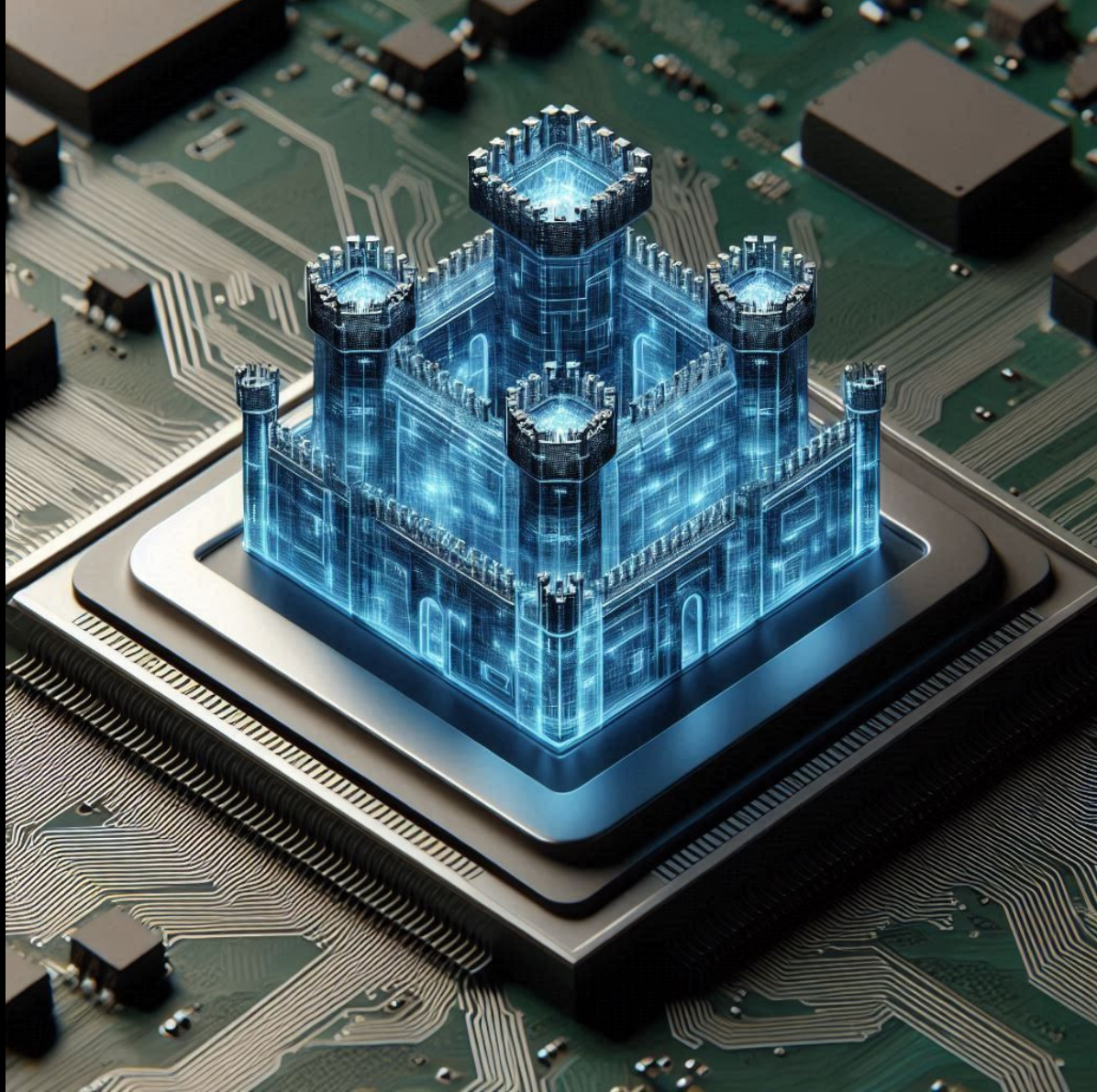
## Data in use

- ? Homomorphic encryption?
- ? Trusted Execution?  
= Confidential Computing  
= Hardware Enclaves

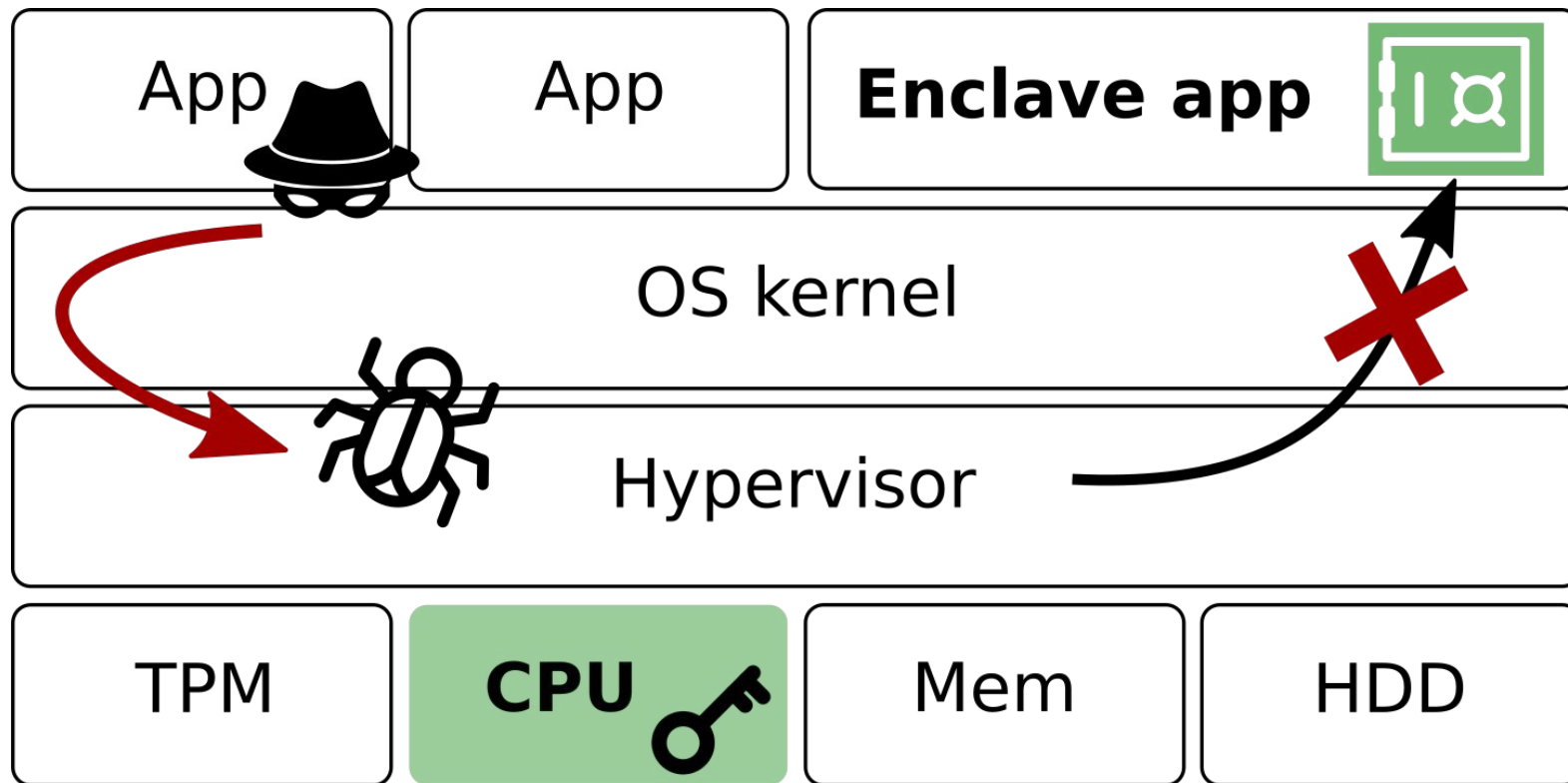


## Data at rest

- ✓ Full disk encryption

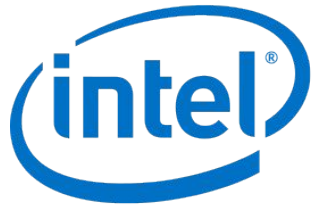


# Confidential Computing: Reducing Attack Surface



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

# The Rise of Trusted Execution Environments (TEEs)

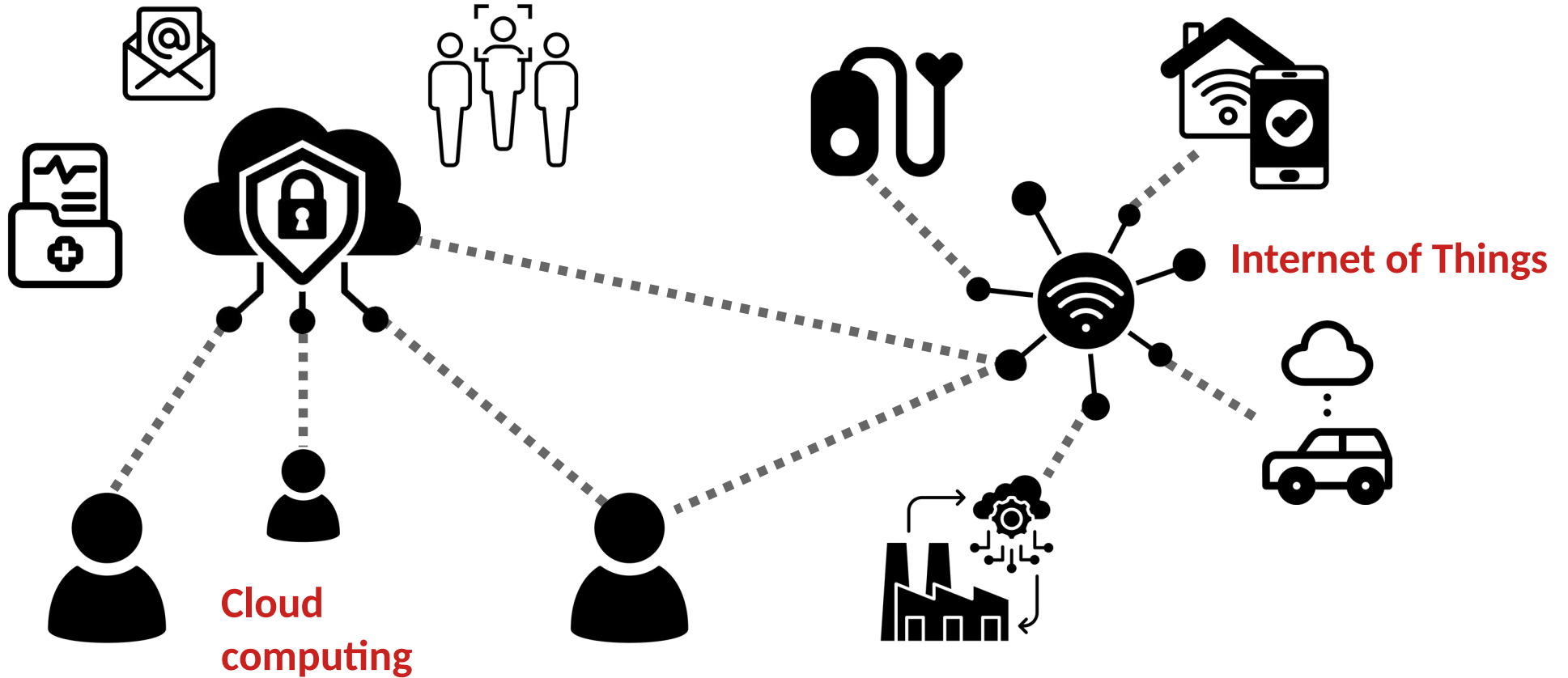


- 2004: ARM TrustZone
- 2015: **Intel Software Guard Extensions (SGX)**
- 2016: AMD Secure Encrypted Virtualization (SEV)
- 2018: IBM Protected Execution Facility (PEF)
- 2020: AMD SEV with Secure Nested Paging (SEV-SNP)
- 2022: Intel Trust Domain Extensions (TDX)
- 2023: ARM Confidential Compute Architecture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



TEEs are here to stay...

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





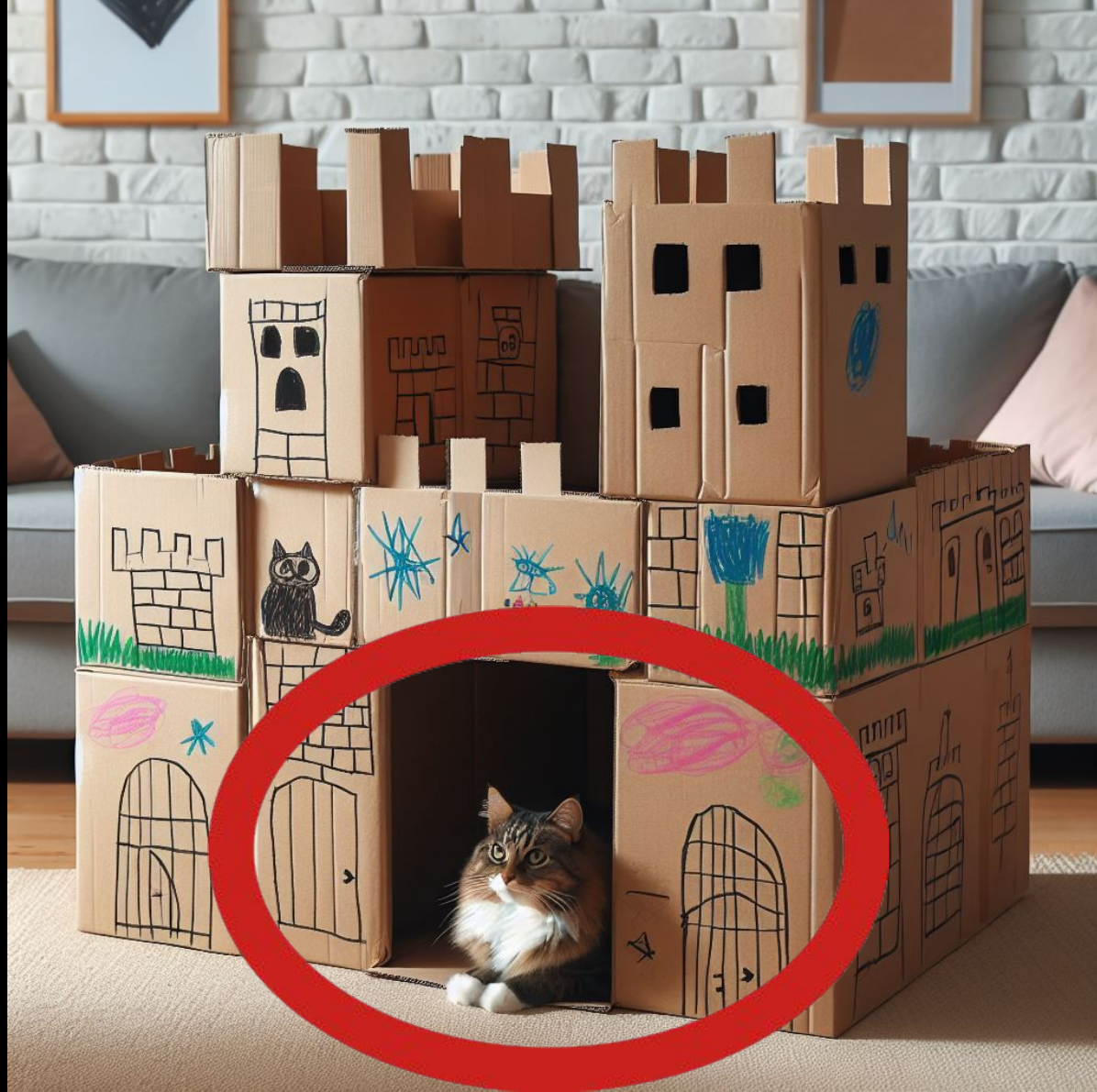
# Fortress or Facade?



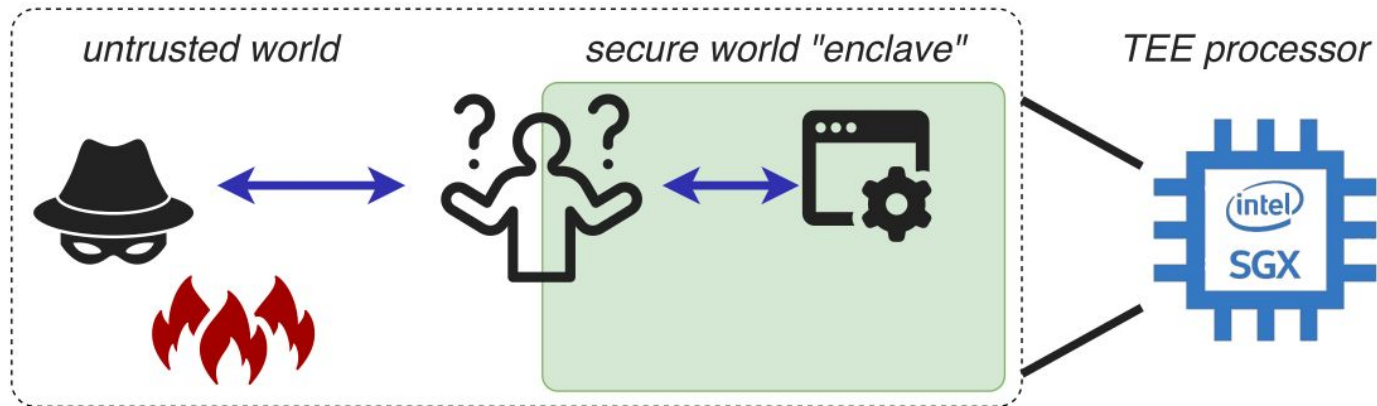


## Case Study #1: Interface?

---

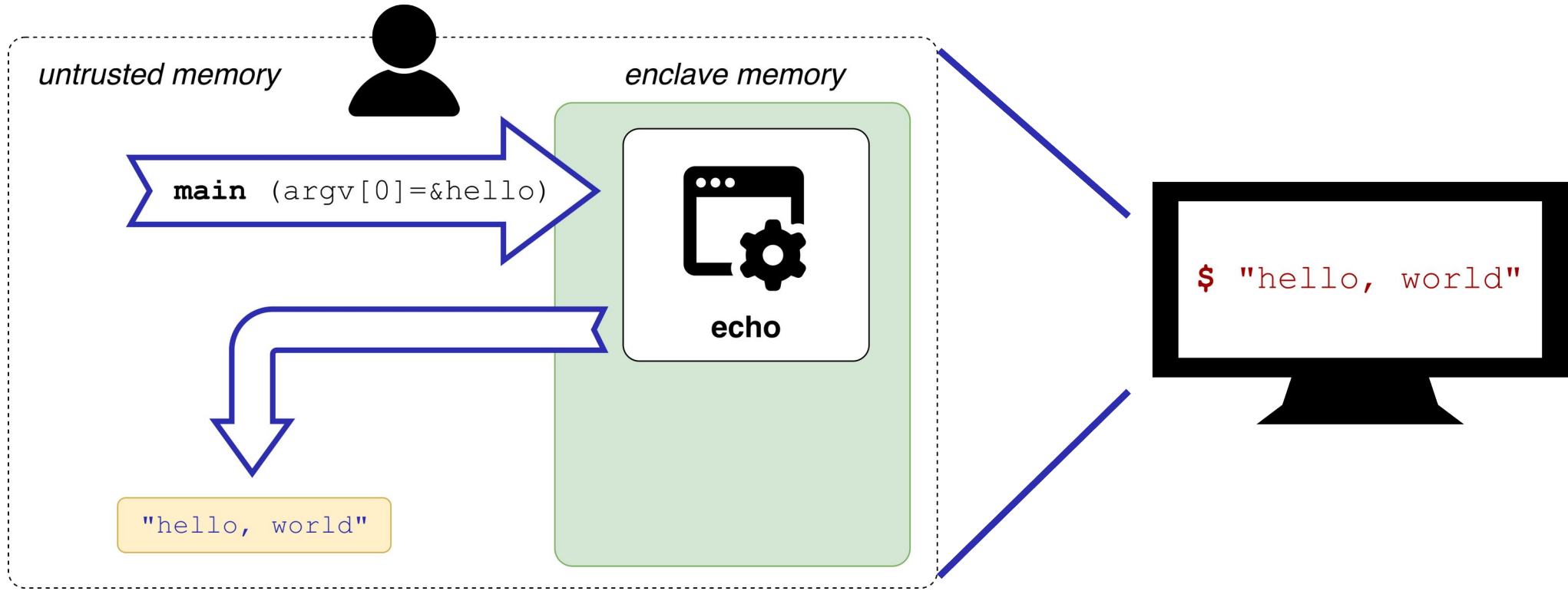


# Context: Writing “Secure” Enclave Software is Hard...

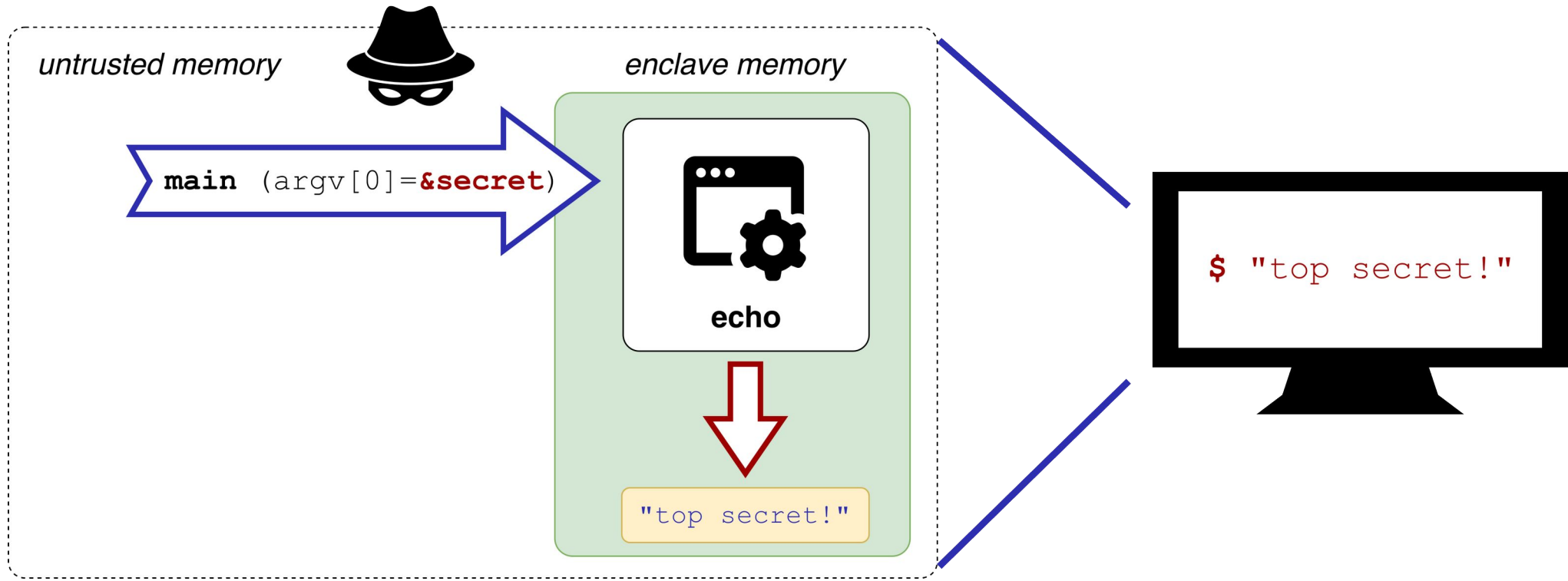


- **API level:** Sanitize pointer arguments in shared address space
- **ABI level:** Sanitize low-level CPU configuration registers
- **$\mu$ -arch level:** Spectre/LVI → `lfence`;  $\text{\AE}$ PIC/MMIO stale data → `verw`; cacheline GPU leak → `avoid dword0/1...`

# API Vulnerabilities: Confused-Deputy Attacks



# API Vulnerabilities: Confused-Deputy Attacks







Home / Tech / Security

# 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



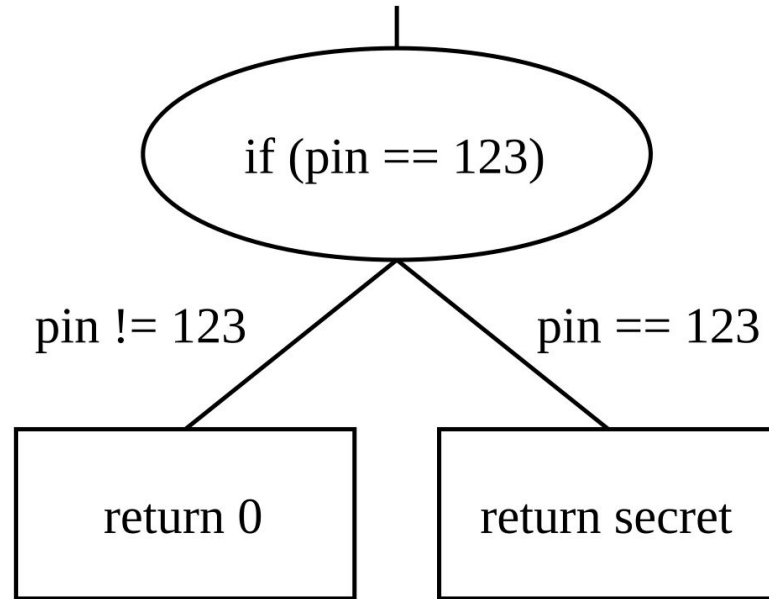


# Pandora: Principled Symbolic Validation of Intel SGX Enclaves

```
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

# Pandora: Principled Symbolic Validation of Intel SGX Enclaves

Runtime	Version	Prod	Src	Plugin	Instances
EnclaveOS	3.28	✓	✗ <sup>†</sup>	ABISan	1
EnclaveOS	3.28	✓	✗ <sup>†</sup>	PTRSan	15
EnclaveOS	3.28	✓	✗ <sup>†</sup>	ÆPICSan	33
EnclaveOS	3.28	✓	✗ <sup>†</sup>	CFSan	2
GoTEE	b35f	✗	✓	PTRSan	31
GoTEE	b35f	✗	✓	ÆPICSan	18
GoTEE	b35f	✗	✓	CFSan	1
Gramine	1.4	✓	✓	ABISan	1
Intel SDK	2.15.1	✓	✓	PTRSan	2
Intel SDK	2.19	✓	✓	ÆPICSan	22
↳ Occlum	0.29.4	✓	✓	ÆPICSan	11
Open Enclave	0.19.0	✓	✓	ABISan	2
Rust EDP	1.71	✓	✓	ABISan	1

Runtime	Version	Prod	Src	Plugin	Instances
Linux selftest	5.18	✗	✓	ABISan	1
↳ DCAP	1.16	✓	✓	ABISan	1
↳ Inclave	0.6.2	✗	✓	ABISan	1
Linux selftest	5.18	✗	✓	PTRSan	5
↳ DCAP	1.16	✓	✓	PTRSan	17
↳ Inclave	0.6.2	✗	✓	PTRSan	2
Linux selftest	5.18	✗	✓	CFSan	1
↳ Inclave	0.6.2	✗	✓	CFSan	1
SCONE	5.7 / 5.8	✓	✗	ABISan	2 / 1
SCONE	5.7 / 5.8	✓	✗	PTRSan	10 / 3
SCONE	5.7 / 5.8	✓	✗	ÆPICSan	11 / 3
SCONE	5.8	✓	✗	CFSan	1

# Report PointerSanitizationPlugin

Plugin description: Validates attacker-tainted pointer dereferences.

Analyzed 'pandora\_selftest\_enclave\_sanitization3.elf', with 'Linux selftest enclave' enclave runtime. Ran for 0:00:12.758955 on 2023-08-03\_19-16-58.

 Enclave info: Address range is [0x0, 0xbfff]

 Summary: Found 1 unique WARNING issue; 2 unique CRITICAL issues.

## Report summary


Severity	Reported issues
WARNING	<ul style="list-style-type: none"><li>• <i>Attacker tainted read inside enclave at 0x2476</i></li></ul>
CRITICAL	<ul style="list-style-type: none"><li>• <i>Unconstrained read at 0x22c3</i></li><li>• <i>Unconstrained read at 0x20be</i></li></ul>

Unconstrained read **CRITICAL** RIP=0x22c3

Plugin extra info


Key	Value
Address	<BV64 0x3000 + ((attacker_mem_66_32{UNINITIALIZED} .. 0x1) << 0x3)>
Attacker tainted	True
Length	8
Pointer range	[0x3008, 0xffffffff800003008]
Pointer can wrap address space	False
Pointer can lie in enclave	True
Extra info	Read address may lie inside or outside enclave

Execution state info

Disassembly 

CPU registers 

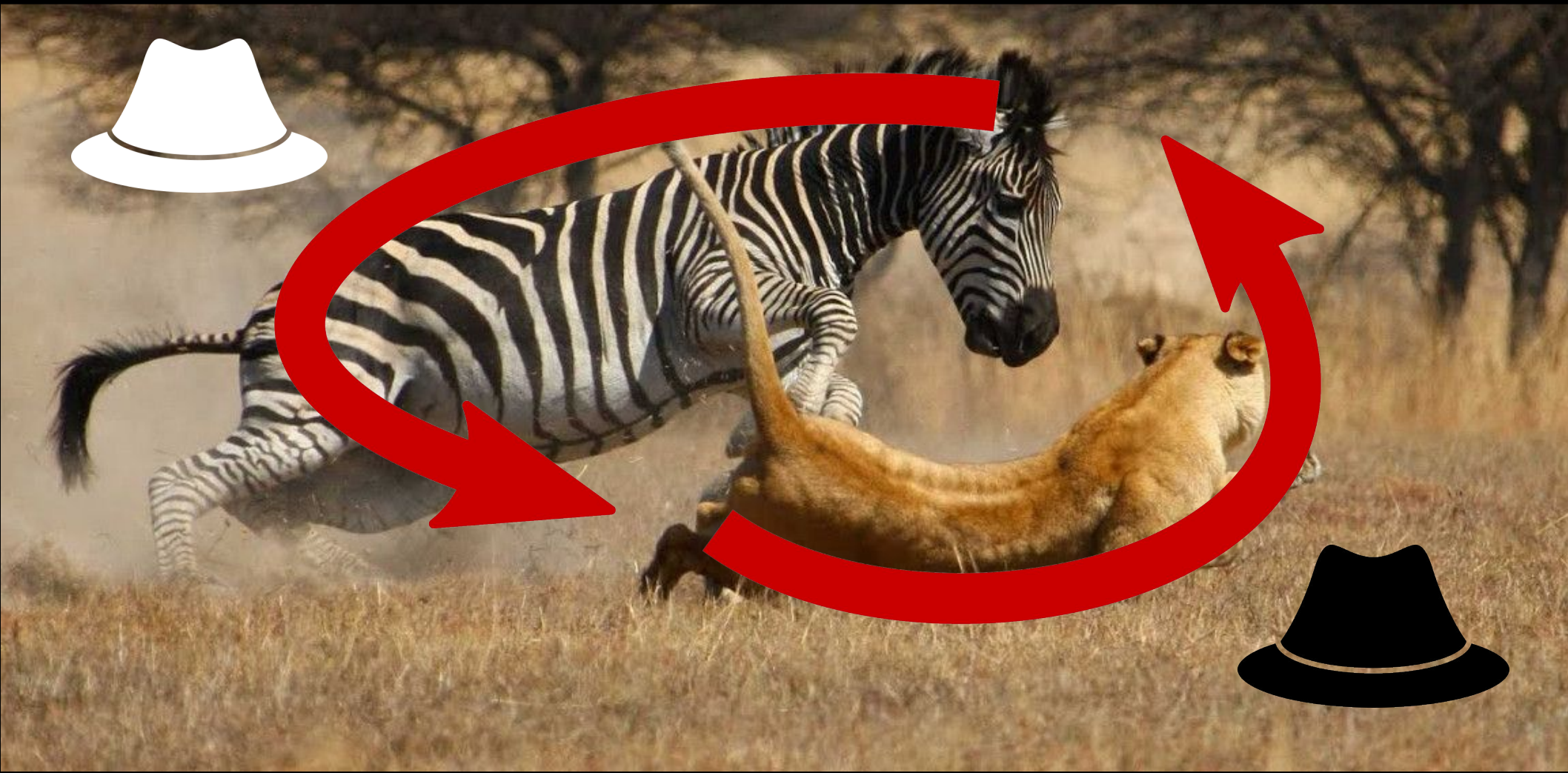
Backtrace

Basic block trace (most recent first) 

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



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



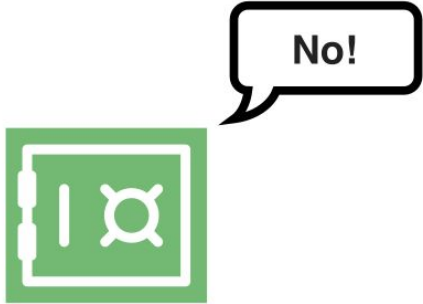


## Case Study #2: Temporal Resolution?

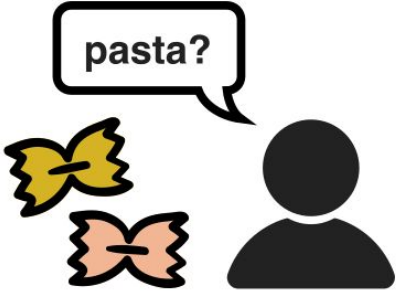
---

# Case Study: Comparing a Secret Password

p a s s w o r d

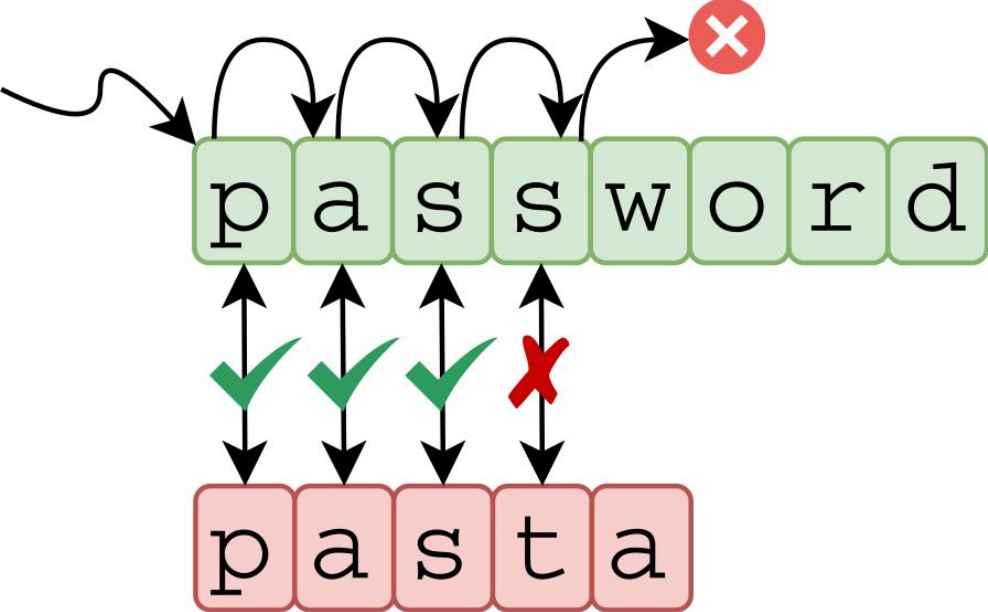


p a s t a

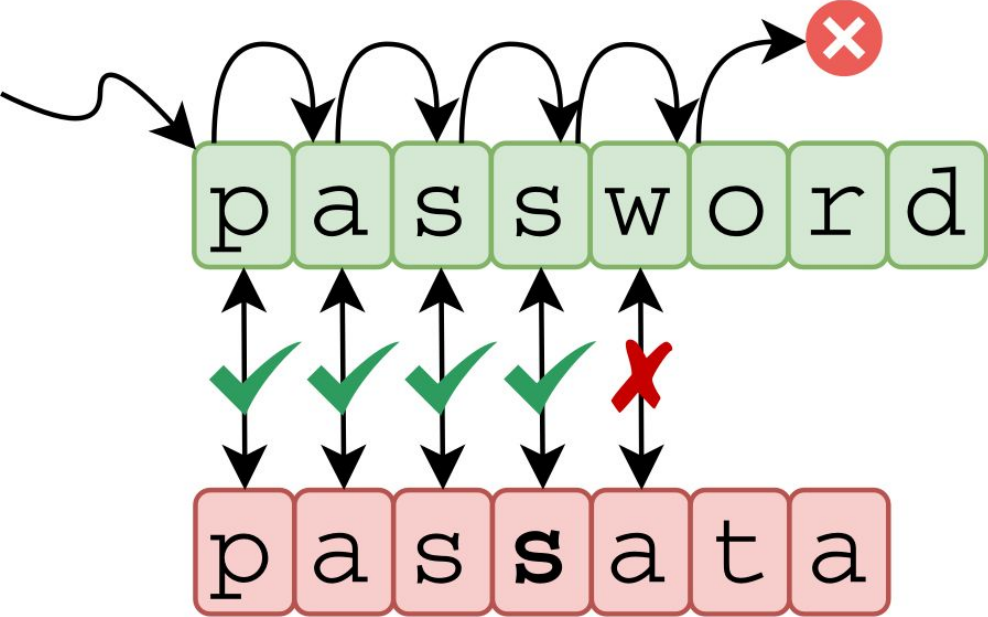




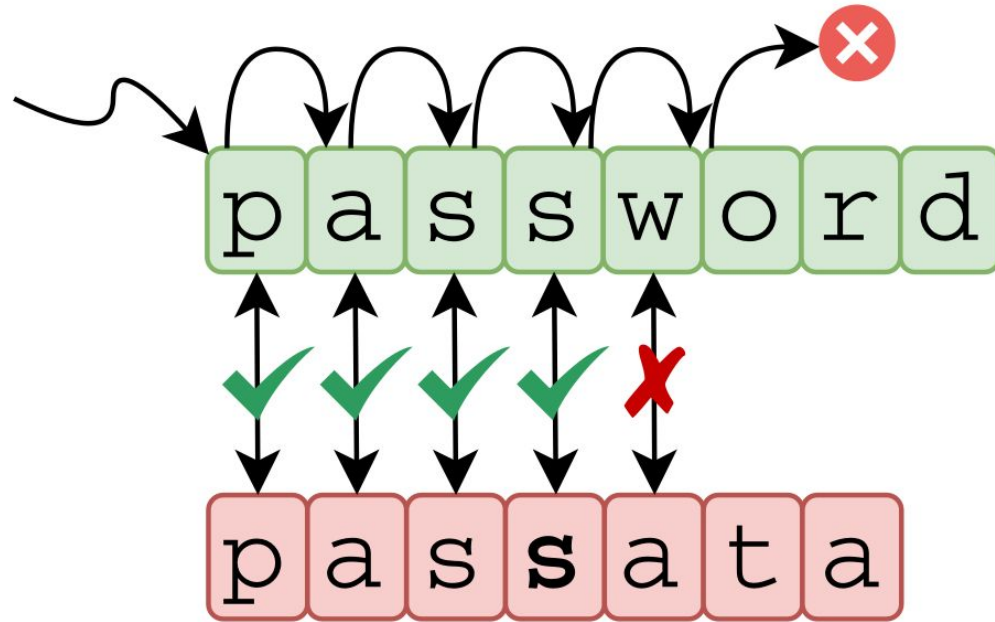
# Case Study: Comparing a Secret Password



# Case Study: Comparing a Secret Password



# Case Study: Comparing a Secret Password

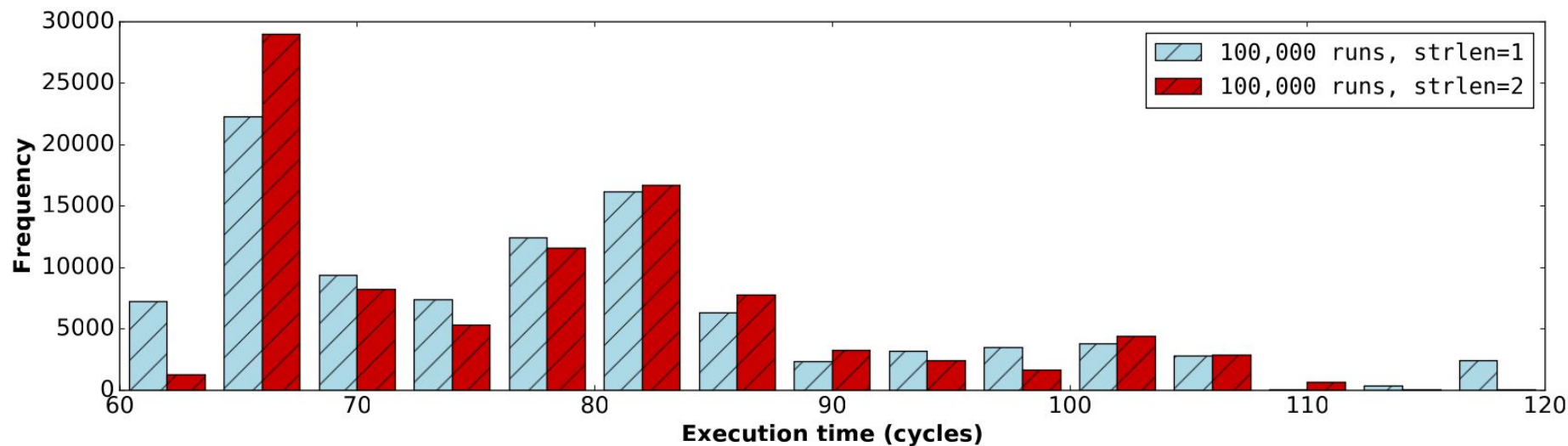


Overall **execution time** reveals correctness of individual password bytes!

# Building the Side-Channel Oracle with Execution Timing?



**Too noisy:** modern x86 processors are lightning fast. . .



# Challenge: Side-Channel Sampling Rate



Slow  
shutter speed

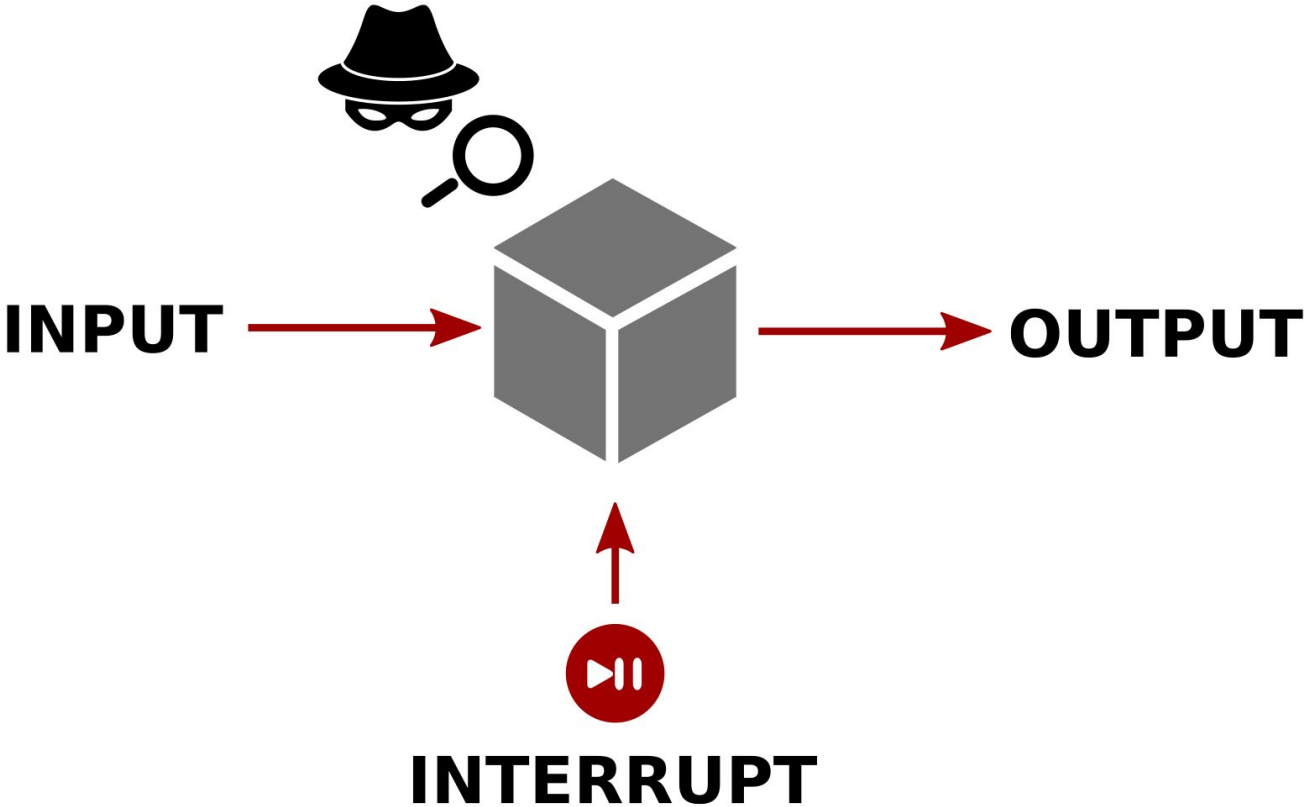


Medium  
shutter speed

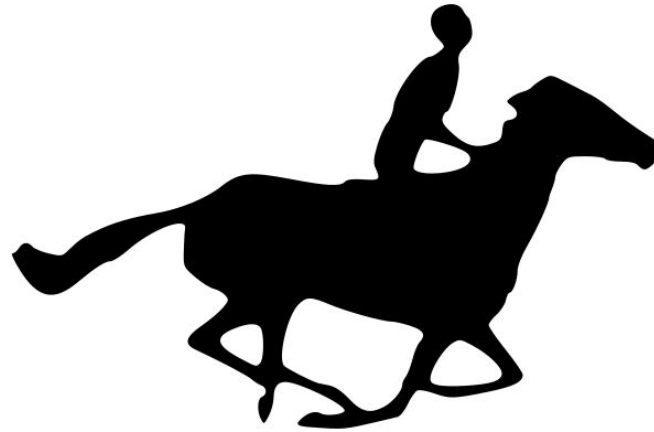


Fast  
shutter speed

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



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



## SGX-Step



**ACSAC 2023  
Cybersecurity Artifacts  
Impact Award**

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

 Watch

22

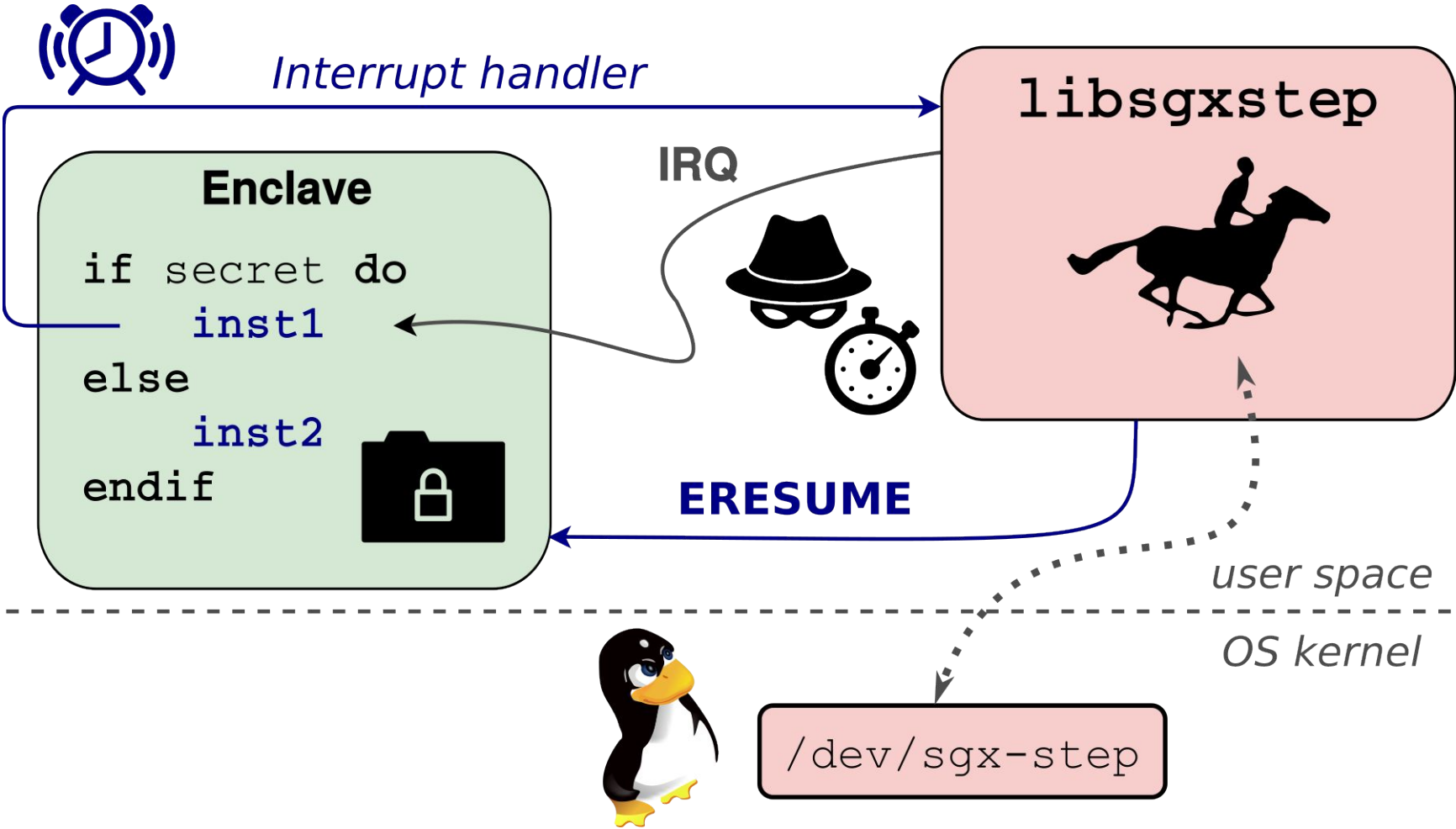
 Star

245

 Fork

52

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

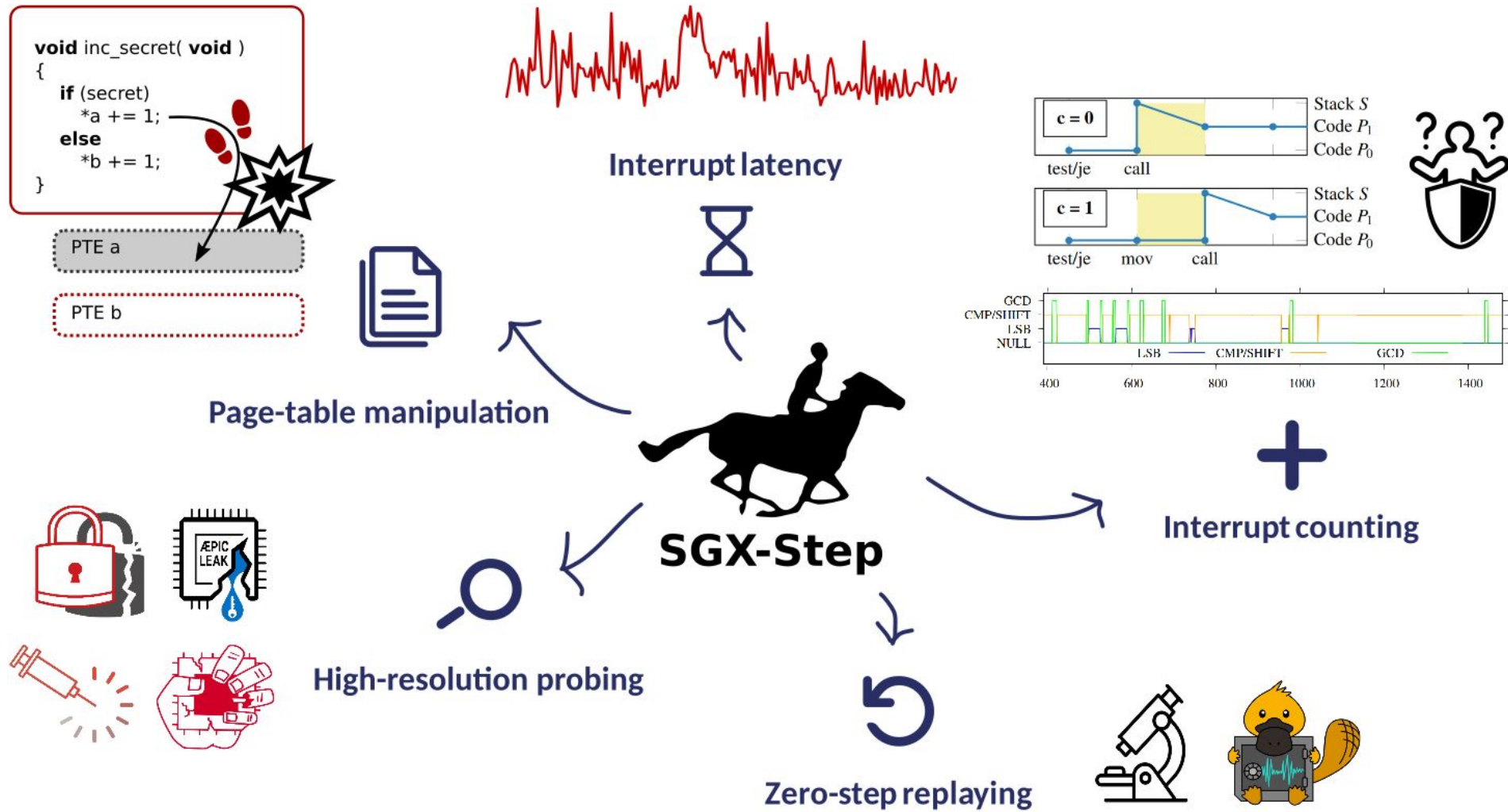




# SGX-Step Demo: Single-Stepping Password Comparison

```
jo@breuer:~/sgx-step-demo$ sudo ./app █
```

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



# 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<sup>®</sup> 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 e
- SSA.GPRSGX.AEXNOTIFY: Enclave-writable byte that allows enclave software to dynamically enable/disable AEX notifications.

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



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

→ shipped in millions of devices ≥ 4th Gen Xeon CPU

## 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.



Code 1

in [intel/linux-sgx](#)

Filter

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

```
48 * Description:
49 *   The file provides mitigations for SGX-Step
50 */
71 * Function:
   constant_time_apply_sgxstep_mitigation_and_continue_execution
72 *   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**



## Case Study #3: Spatial Resolution?

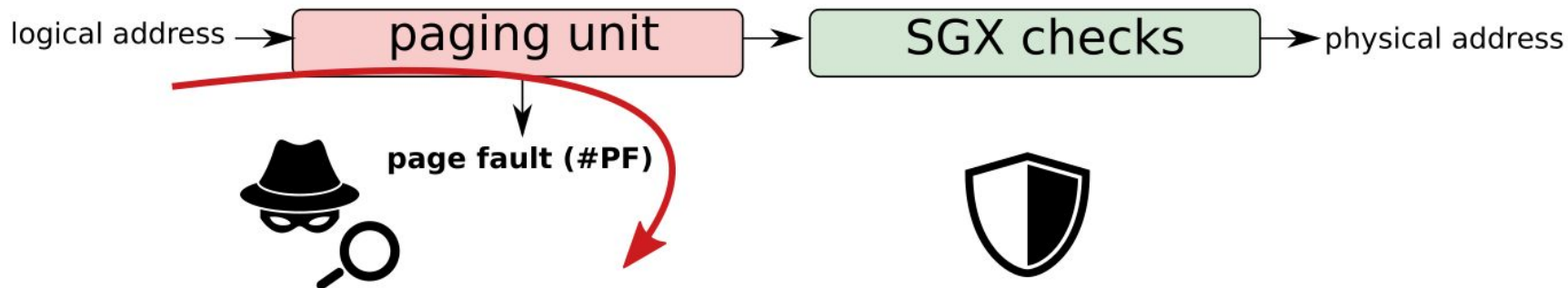
---

# Idea: Page Faults as a Side Channel



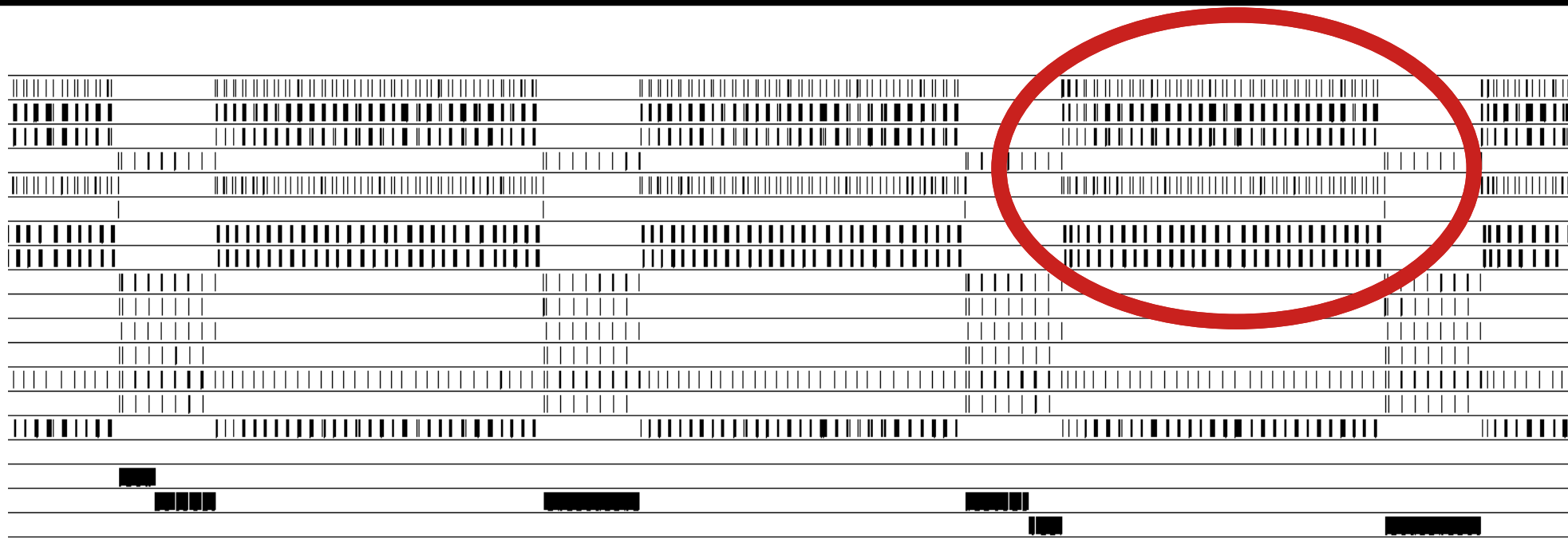
**SGX machinery** protects against direct address remapping attacks

# Idea: Page Faults as a Side Channel



... but untrusted address translation may **fault(!)**

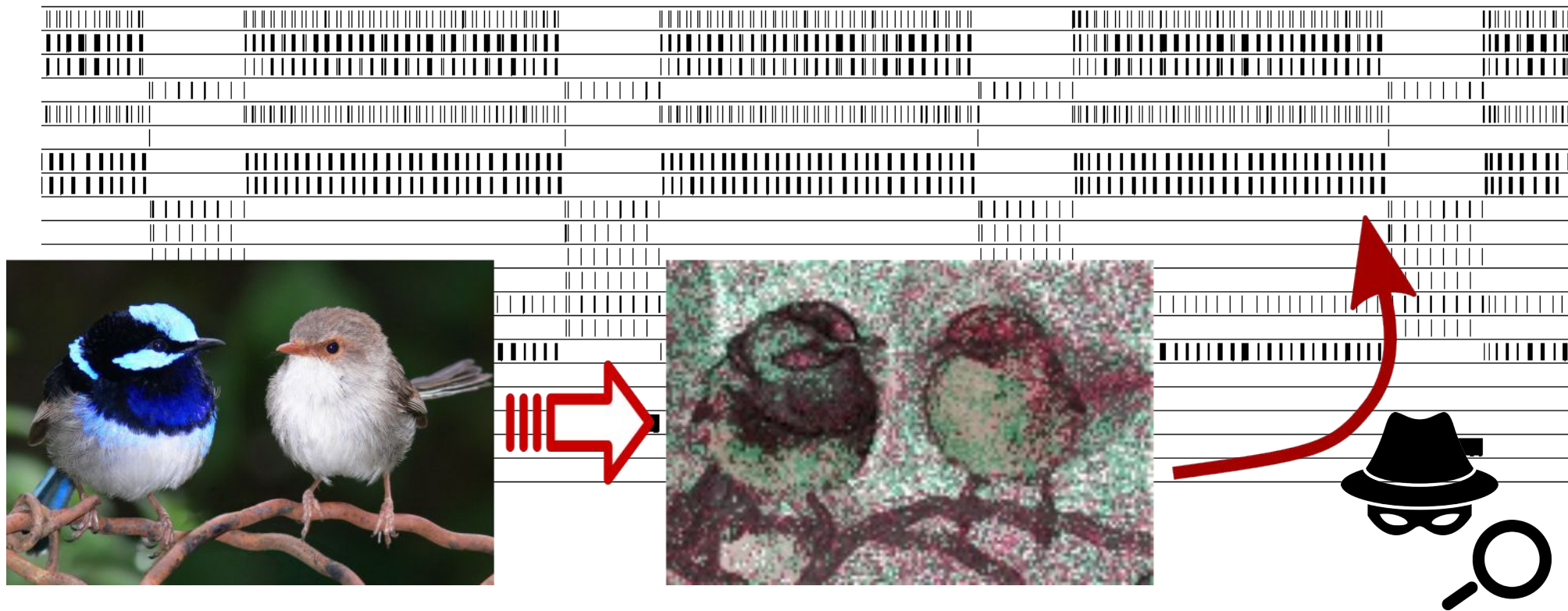
# Spatial Resolution: Page-Granular Memory Access Traces



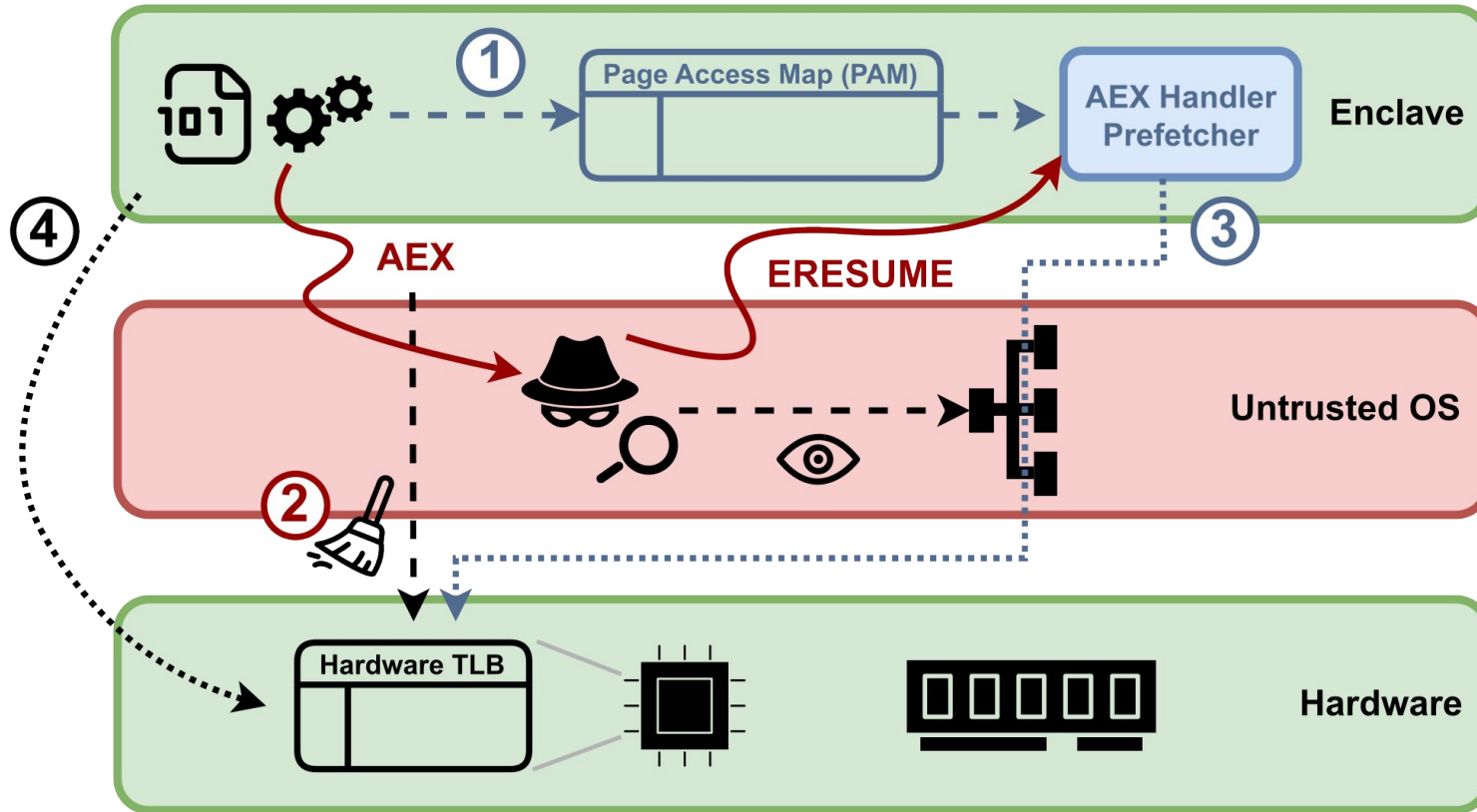
Detailed trace of (coarse-grained) **code and data accesses over time...**



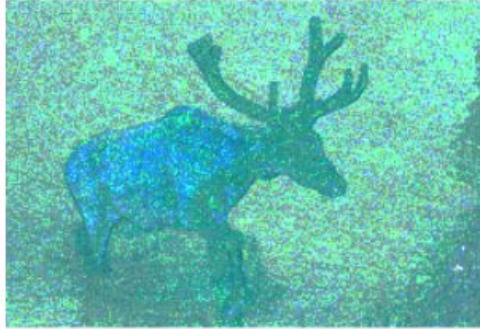
# Spatial Resolution: Page-Granular Memory Access Traces



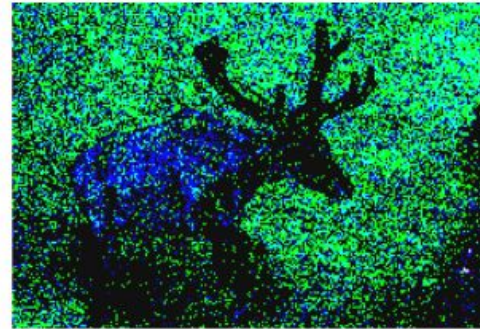
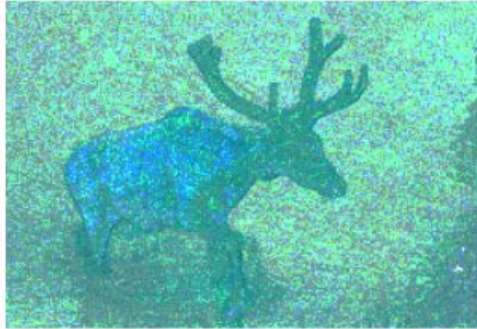
# TLBlur: Self-Monitoring and Restoring Enclave Page Accesses



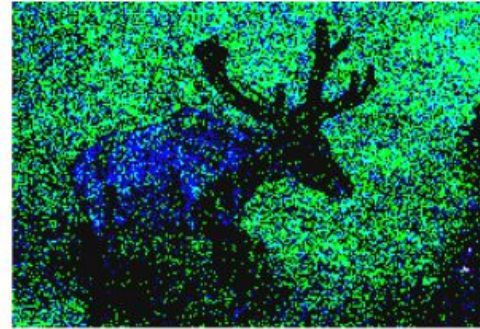
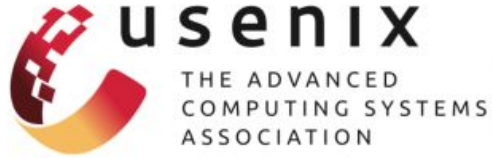
# TLBlur: Compiler-Assisted Leakage Reduction in Practice



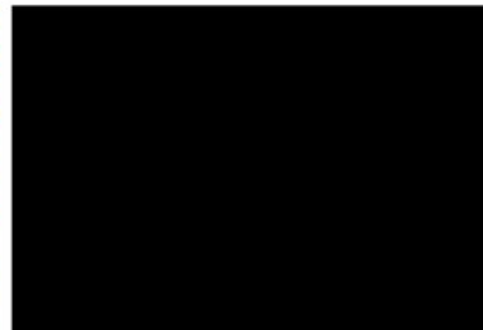
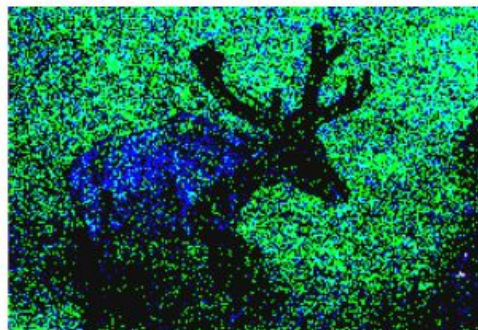
# TLBlur: Compiler-Assisted Leakage Reduction in Practice



# TLBlur: Compiler-Assisted Leakage Reduction in Practice



# TLBlur: Compiler-Assisted Leakage Reduction in Practice



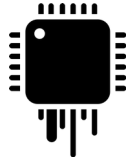
Automated “blurring” of page-access traces in space and time



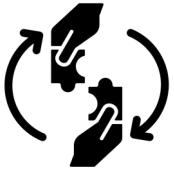
# Conclusions and Take-Away



New era of **confidential computing** for the cloud and IoT



... but current architectures are **not perfect!**



Scientific understanding driven by **attacker-defender race**



*Thank you!*