Towards ABI Unification for Intel SGX Enclave Shielding Runtimes

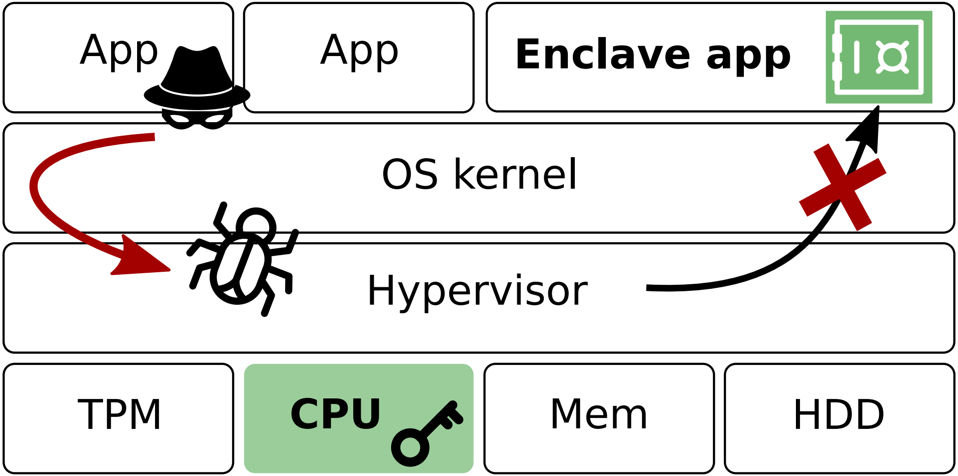
Jo Van Bulck, Fritz Alder, Frank Piessens

SILM'22 Workshop, Lightning talk, Genoa, Italy, June 6, 2022

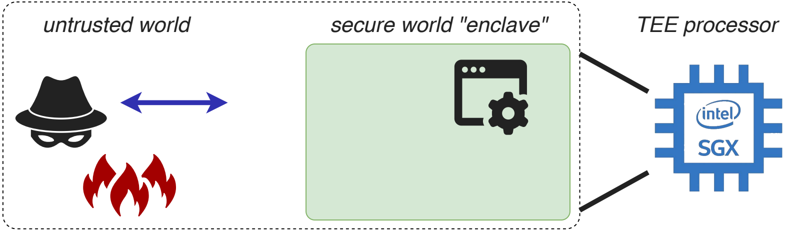
imec-DistriNet, KU Leuven jo.vanbulck@cs.kuleuven.be jovanbulck

DistriNet
KU LEUVEN

Intel SGX: Hardware-level isolation and attestation

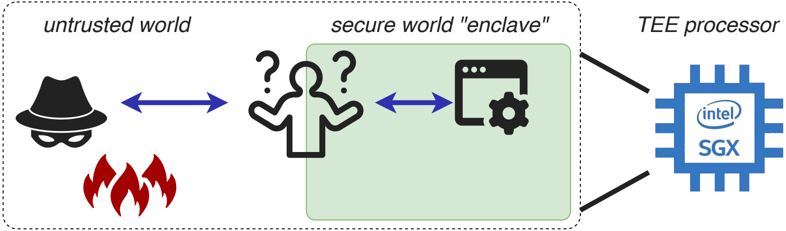


Why isolation is not enough: Enclave shielding runtimes



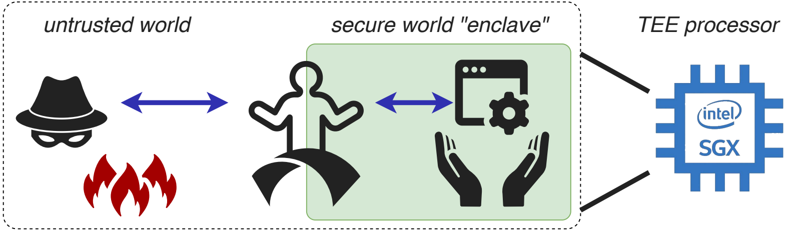
- TEE promise: enclave == “secure oasis” in a **hostile environment**

Why isolation is not enough: Enclave shielding runtimes



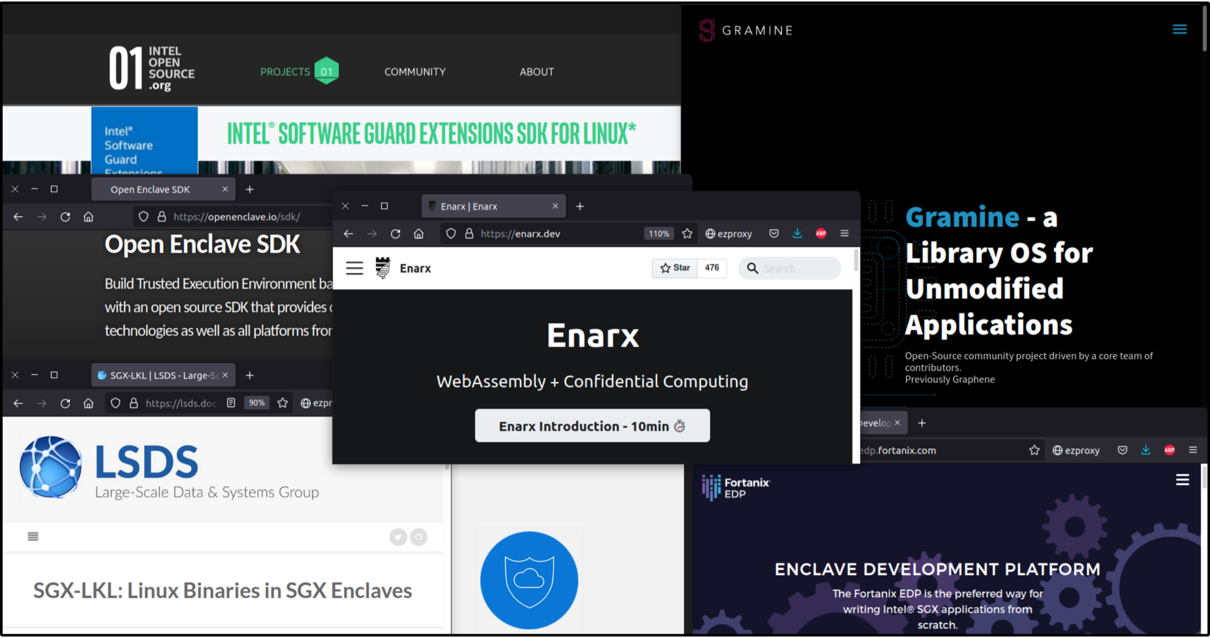
- TEE promise: enclave == "secure oasis" in a **hostile environment**
- . . . but application and compilers largely unaware of **isolation boundaries**

Why isolation is not enough: Enclave shielding runtimes



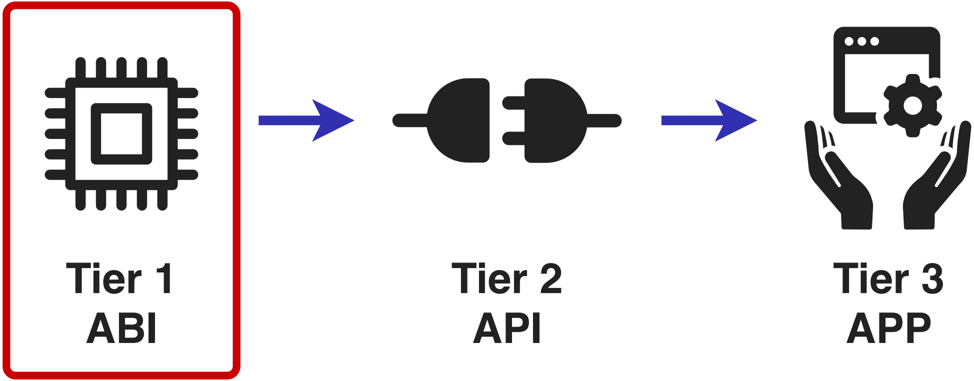
- TEE promise: enclave == "secure oasis" in a **hostile environment**
- ...but application and compilers largely unaware of **isolation boundaries**

> **Shielding runtime** == secure bridge on enclave entry/exit


01 INTEL OPEN SOURCE .org
PROJECTS 01
COMMUNITY
ABOUT
Intel® Software Guard Extensions
INTEL® SOFTWARE GUARD EXTENSIONS SDK FOR LINUX*
GRAMINE
Gramine - a Library OS for Unmodified Applications
Open-Source community project driven by a core team of contributors.
Previously Graphene
Open Enclave SDK
https://openenclave.io/sdk/
Open Enclave SDK
Build Trusted Execution Environment ba
with an open source SDK that provides
technologies as well as all platforms fro
Enarx | Enarx
https://enarx.dev
Enarx
Star 476
Search
Enarx
WebAssembly + Confidential Computing
Enarx Introduction - 10min
SGX-LKL | LSDS - Large-S
https://lsds.doc
LSDS
Large-Scale Data & Systems Group
SGX-LKL: Linux Binaries in SGX Enclaves
edp.fortanix.com
Fortanix EDP
ENCLAVE DEVELOPMENT PLATFORM
The Fortanix EDP is the preferred way for writing Intel® SGX applications from scratch.

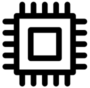
Towards unified shielding in Intel SGX runtimes?

- We celebrate application and programming language diversity
- . . . but unification of shared insights at the binary level (ABI)!



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

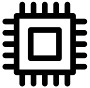
Tier1: Establishing a trustworthy enclave ABI



$\rightsquigarrow$ Attacker controls CPU register contents on enclave entry/exit

$\leftrightarrow$ Compiler expects well-behaved **calling convention** (e.g., stack)

Tier1: Establishing a trustworthy enclave ABI

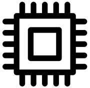


$\leadsto$ Attacker controls CPU register contents on enclave entry/exit

$\leftrightarrow$ Compiler expects well-behaved **calling convention** (e.g., stack)

$\Longrightarrow$ Need to **initialize CPU registers** on entry and **scrub** before exit!

Tier1: Establishing a trustworthy enclave ABI



$\rightsquigarrow$ Attacker controls CPU register contents on enclave entry/exit

$\leftrightarrow$ Compiler expects well-behaved **calling convention** (e.g., stack)

$\Longrightarrow$ Need to **initialize CPU registers** on entry and **scrub** before exit!

Non-trivial for x86 ISA $\rightarrow$ attacks! (CCS'19, ACSAC'20, CCS'21)

A Case for Unified ABI Shielding in Intel SGX Runtimes

Jo Van Bulck, Fritz Alder, Frank Piessens
imec-DistriNet, KU Leuven, Belgium

ABSTRACT

With hardware support for trusted execution, most notably Intel SGX, becoming widely available, recent years have seen the emergence of numerous *shielding runtimes* to transparently protect enclave applications in hostile environments. While, at the application level, a wide range of languages and development paradigms are supported by diverse runtimes, shielding responsibilities at the lowest level of the application binary interface (ABI) remain strikingly similar. Particularly, the ABI dictates that certain CPU registers need to be cleansed and initialized via a small, hand-written assembly stub upon every enclave context switch.

This paper and call for action analyzes the ABI sanitization layers of 8 open-source SGX shielding runtimes from industry and academia, categorizes historic vulnerabilities therein, and identifies cross-cutting tendencies and insights. We conclude that there is *no* technical reason for maintaining separate, often notoriously complex and vulnerable ABI code bases. Moving forward, we outline challenges and opportunities for a single, unified ABI sanitization layer that complies with best practices from software engineering and can be scrutinized and integrated across SGX runtimes.

ACM Reference Format:
Jo Van Bulck, Fritz Alder, Frank Piessens. 2022. A Case for Unified ABI Shielding in Intel SGX Runtimes. In *Proceedings of the 5th Workshop on System Software for Trusted Execution (SysTEX '22 Workshop)*. ACM, New York, NY, USA, 3 pages.

on every enclave context switch. Next, a secondary stage, written in a higher-level language, may sanitize application programming interface (API) state, such as pointer arguments. It is worth noting that low-level ABI shielding responsibilities are relatively contained and language-agnostic, whereas sanitizing program-visible API state is typically more complex and may be highly dependant on the specific runtime and supported programming model.

Table 1: Overview of the Intel SGX ABI vulnerability landscape. The top rows compare ABI sanitization layers in terms of total lines of code (as measured on January 20, 2022; using `cloc`) and lines changed since original release (as reported by `git`; following renamed/moved files). The third row distinguishes (aspired) production runtimes from research prototypes. The bottom rows list which runtimes were found to be vulnerable to (●), not vulnerable to (○), or not analyzed by (–) prior attack studies.

		SGX-SDK*	OE**	EDP	Gramine	Enarx	GoTEE	SGX-LKL	OpenSGX
Metrics	LoC ABI stub	301	277	248	427	169	239	103	49
	LoC changed	243	589	187	1,840	844	65	47	0
	Production?	✓	✓	✓	✓	✓	✗	✗	✗
Vulnerabilities	Entry flags [17]	●	●	●	●	–	–	●	–
	Entry stack [17]	○	○	○	●	–	–	●	–
	Exit registers [17]	○	○	○	○	–	–	●	–
	Entry FPU [1]	●	●	●	○	○	●	●	–
	Exception stack [3]	●	●	○	○	●	–	●	–

* Derived runtimes include Apache Teaclave [15, 18], VeraCruz [2], and Google Asylo [9].
** Derived runtimes include EdgelessRT [4], and recent versions of SGX-LKL "OE edition".

Summary: Intel SGX ABI vulnerability landscape

	SGX-SDK	OE	EDP	Gramine	Enarx	GoTEE	SGX-LKL	OpenSGX
Entry flags [4]	●	●	●	●	–	–	●	–
Entry stack [4]	○	○	○	●	–	–	●	–
Exit registers [4]	○	○	○	○	–	–	●	–
Entry FPU [1]	●	●	●	○	○	●	●	–
Exception stack [2]	●	●	○	○	●	–	●	–

Relatively understood, but special care for **stack pointer + status register + FPU**

Summary: Intel SGX ABI vulnerability landscape

	SGX-SDK	OE	EDP	Gramine	Enarx	GoTEE	SGX-LKL	OpenSGX
Entry flags [4]	●	●	●	●	–	–	●	–
Entry stack [4]	○	○	○	●	–	–	●	–
Exit registers [4]	○	○	○	○	–	–	●	–
Entry FPU [1]	●	●	●	○	○	●	●	–
Exception stack [2]	●	●	○	○	●	–	●	–
Production?	✓	✓	✓	✓	✓	✗	✗	✗

🔔 (Aspired) **production-quality** runtimes vs. research prototypes

KEEP CALM

AND

SHOW ME THE NUMBERS

Summary: Intel SGX ABI shielding layer metrics

	SGX-SDK	OE	EDP	Gramine	Enarx	GoTEE	SGX-LKL	OpenSGX
LoC ABI stub	**301**	**277**	**248**	**427**	**169**	**239**	**103**	**49**
LoC changed	243	589	187	1,840	844	65	47	0
Production?	✓	✓	✓	✓	✓	✗	✗	✗
Entry flags [4]	●	●	●	●	–	–	●	–
Entry stack [4]	○	○	○	●	–	–	●	–
Exit registers [4]	○	○	○	○	–	–	●	–
Entry FPU [1]	●	●	●	○	○	●	●	–
Exception stack [2]	●	●	○	○	●	–	●	–

Size: Non-trivial: > *100s lines of hand-written, vulnerable asm code*

Summary: Intel SGX ABI shielding layer metrics

	SGX-SDK	OE	EDP	Gramine	Enarx	GoTEE	SGX-LKL	OpenSGX
LoC ABI stub	301	277	248	427	169	239	103	49
LoC changed	**243**	**589**	**187**	**1,840**	**844**	**65**	**47**	**0**
Production?	✓	✓	✓	✓	✓	✗	✗	✗
Entry flags [4]	●	●	●	●	–	–	●	–
Entry stack [4]	○	○	○	●	–	–	●	–
Exit registers [4]	○	○	○	○	–	–	●	–
Entry FPU [1]	●	●	●	○	○	●	●	–
Exception stack [2]	●	●	○	○	●	–	●	–

🔔 **History:** Maintaining ABI code is an *ongoing* and *living* effort!

Summary: Intel SGX ABI patch timelines

	SGX-SDK	OE	EDP	Gramine	Enarx
Initial commit	°24/06/16	°29/08/17	°07/12/18	°20/06/16	°20/02/20
Direction flag [4]	17/10/19	09/10/19	**07/12/18**	01/05/19	20/03/20
Alignment-check flag [4]	12/11/19	09/10/19	21/10/19 10/02/20	19/11/19	★17/02/22
FPU extended state [1]	16/01/20	**09/10/19** 14/07/20	10/02/20 19/06/20	17/10/19	29/05/20
Exception stack [2]	13/07/21	13/07/21	N/A	**01/04/19** 31/01/20	22/10/21

Security: Already known, not communicated, open gap

Summary: Intel SGX ABI patch timelines

	SGX-SDK	OE	EDP	Gramine	Enarx
Initial commit	° 24/06/16	° 29/08/17	° 07/12/18	° 20/06/16	° 20/02/20
Direction flag [4]	**17/10/19**	**09/10/19**	07/12/18	01/05/19	20/03/20
Alignment-check flag [4]	**12/11/19**	**09/10/19**	**21/10/19** 10/02/20	**19/11/19**	★ 17/02/22
FPU extended state [1]	**16/01/20**	09/10/19 **14/07/20**	**10/02/20** **19/06/20**	17/10/19	29/05/20
Exception stack [2]	**13/07/21**	**13/07/21**	N/A	01/04/19 31/01/20	**22/10/21**

Deepened understanding: Importance of academic research!

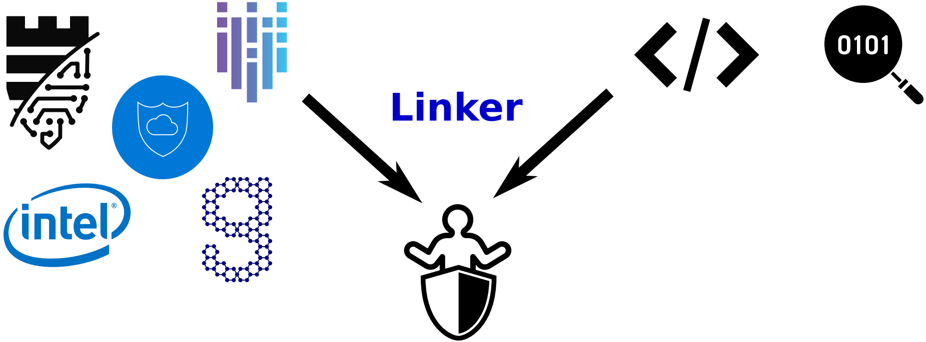
Summary: Intel SGX ABI patch timelines

	SGX-SDK	OE	EDP	Gramine	Enarx
Initial commit	°24/06/16	°29/08/17	°07/12/18	°20/06/16	°20/02/20
Direction flag [4]	17/10/19	09/10/19	07/12/18	01/05/19	20/03/20
Alignment-check flag [4]	12/11/19	09/10/19	21/10/19 10/02/20	19/11/19	★ **17/02/22**
FPU extended state [1]	16/01/20	09/10/19 14/07/20	10/02/20 19/06/20	17/10/19	29/05/20
Exception stack [2]	13/07/21	13/07/21	N/A	01/04/19 31/01/20	22/10/21

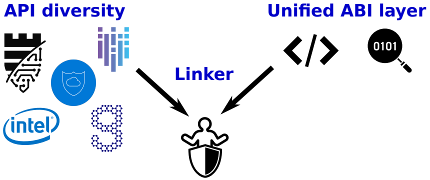
Systematization: Revealed *missing patch*, fixed in Enarx v0.2.1

Towards unified ABI shielding for Intel SGX runtimes?

API diversity **Unified ABI layer**



Towards unified ABI shielding for Intel SGX runtimes?



Thank you! Food for thought?

Challenges and opportunities of a joined enclave ABI? — Does diversity benefit security? — Lessons from OS kernel development? — Towards a unified enclave API calling convention? — Towards a standardized enclave ELF binary format? — Open-source SGX ecosystem "wildgrowth"?

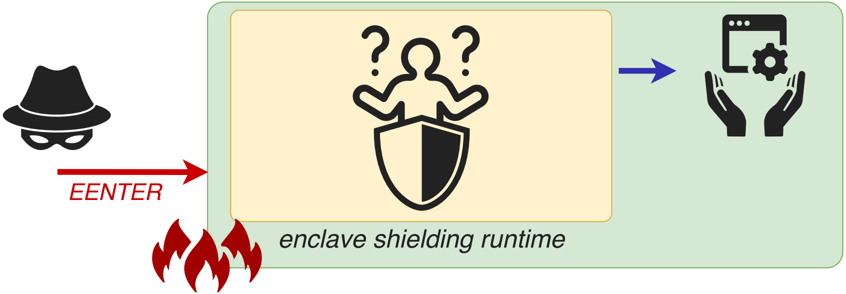
References i

- Fritz Alder, Jo Van Bulck, David Oswald, and Frank Piessens.
  Faulty point unit: ABI poisoning attacks on Intel SGX.
  In *36th Annual Computer Security Applications Conference (ACSAC)*, pages 415–427, December 2020.
- Jinhua Cui, Jason Zhijingcheng Yu, Shweta Shinde, Prateek Saxena, and Zhiping Cai.
  SmashEx: smashing SGX enclaves using exceptions.
  In *28th ACM Conference on Computer and Communications Security (CCS)*, page 779–793, 2021.

- Jo Van Bulck, Fritz Alder, and Frank Piessens.
  A case for unified ABI shielding in Intel SGX runtimes.
  In *5th Workshop on System Software for Trusted Execution (SysTEX)*. ACM, March 2022.
- Jo Van Bulck, David Oswald, Eduard Marin, Abdulla Aldoseri, Flavio D. Garcia, and Frank Piessens.
  A tale of two worlds: Assessing the vulnerability of enclave shielding runtimes.
  In *26th ACM Conference on Computer and Communications Security (CCS)*, pages 1741–1758, November 2019.

The big picture: Enclave shielding responsibilities

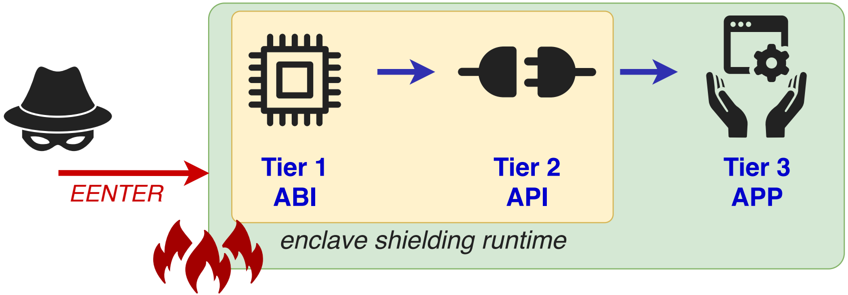
Key questions: how to securely bootstrap from the untrusted world to the enclaved application binary (and back)? Which sanitizations to apply?



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

The big picture: Enclave shielding responsibilities

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



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

ABI vs. API sanitization responsibilities

Application Binary Interface

- Expectations by compiler
- Low-level CPU state (registers)
- Hand-written assembly stub

Application Programming Interface

- Expectations by application writer
- High-level program state (pointers)
- Automated abstractions (e.g., `edger8r` DSL, EDP type system)

ABI vs. API sanitization responsibilities

Application Binary Interface

- Expectations by compiler
- Low-level CPU state (registers)
- Hand-written assembly stub

Application Programming Interface

- Expectations by application writer
- High-level program state (pointers)
- Automated abstractions (e.g., `edger8r` DSL, EDP type system)

(Needlessly) **duplicated effort** across runtimes!

Depending on specific runtime and programming model…


REC
SHELDON COOPER
presents
FUN WITH FLAGS

x86 string instructions: Direction Flag (DF) operation

- x86 rep string instructions to speed up streamed memory operations

```c
/* memset(buf, 0x0, 100) */
for (int i=0; i < 100; i++)
  buf[i] = 0x0;
```

→

```asm
lea rdi, buf
mov al, 0x0
mov ecx, 100
rep stos [rdi], al
```

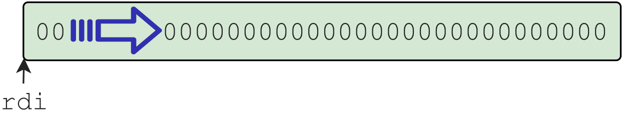
x86 string instructions: Direction Flag (DF) operation

- x86 rep string instructions to speed up streamed memory operations
- Default operate **left-to-right**

```c
/* memset(buf, 0x0, 100) */
for (int i=0; i < 100; i++)
  buf[i] = 0x0;
```

→

```asm
lea rdi, buf
mov al, 0x0
mov ecx, 100
rep stos [rdi], al
```

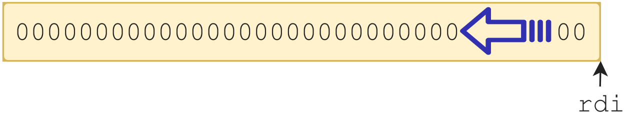


x86 string instructions: Direction Flag (DF) operation

- x86 rep string instructions to speed up streamed memory operations
- Default operate **left-to-right**, unless software sets *RFLAGS.DF=1*

```c
/* memset(buf, 0x0, 100) */
for (int i=0; i < 100; i++)
  buf[i] = 0x0;
```

```asm
lea rdi, buf+100
mov al, 0x0
mov ecx, 100
std ; set direction flag
rep stos [rdi], al
```



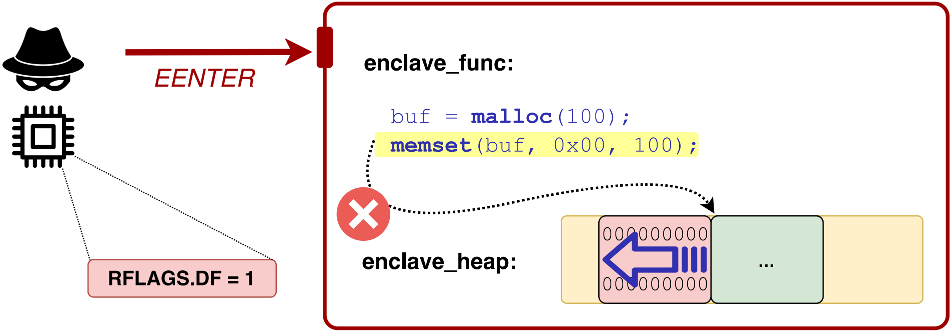
SGX-DF: Inverting enclaved string memory operations

x86 System-V ABI

[8] The direction flag `DF` in the `%rFLAGS` register must be clear (set to "forward" direction) on function entry and return. Other user flags have no specified role in the standard calling sequence and are *not* preserved across calls.

SGX-DF: Inverting enclaved string memory operations

⚠ Enclave heap **memory corruption:** right-to-left. . .



Summary:

A potential security vulnerability in Intel SGX SDK may allow for information disclosure, escalation of privilege or denial of service. Intel is releasing software updates to mitigate this potential vulnerability. This potential vulnerability is present in all SGX enclaves built with the affected SGX SDK versions.

Vulnerability Details:

CVEID: CVE-2019-14566

Description: Insufficient input validation in Intel(R) SGX SDK versions shown below may allow an authenticated user to enable information disclosure, escalation of privilege or denial of service via local access.

CVSS Base Score: 7.8 (High)

CVSS Vector: CVSS:3.1/AV:L/AC:H/PR:L/UI:N/S:C/C:H/I:H/A:H

CVEID: CVE-2019-14565

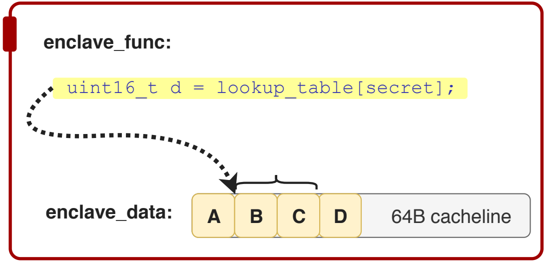
Description: Insufficient initialization in Intel(R) SGX SDK versions shown below may allow an authenticated user to enable information disclosure, escalation of privilege or denial of service via local access.

CVSS Base Score: 7.0 (High)

CVSS Vector: CVSS:3.1/AV:L/AC:H/PR:L/UI:N/S:C/C:L/I:L/A:H

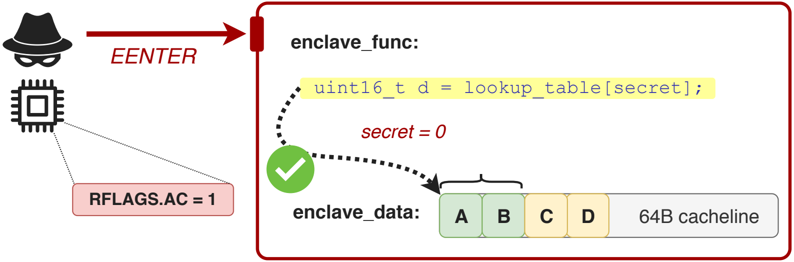
SGX-AC: Building an intra-cacheline side-channel

There's more! Alignment Check (AC) flag enables **exceptions for unaligned data accesses** → *intra-cacheline side-channel* ☺



SGX-AC: Building an intra-cacheline side-channel

Enter enclave with *RFLAGS.AC=1* and secret index=0
→ well-aligned data access: **no exception**



SGX-AC: Building an intra-cacheline side-channel

Enter enclave with *RFLAGS.AC=1* and secret index=1
→ unaligned data access: **alignment-check exception. . .**