EnclaveFuzz: Finding Vulnerabilities in SGX Applications

Liheng Chen∗ , Zheming Li∗ , Zheyu Ma, Yuan Li, Baojian Chen, Chao Zhang†

* The first two authors contributed equally to this work.

† Corresponding author: chaoz@tsinghua.edu.cn

NDSS SYMPOSIUM/2024

Presented by Internet Society

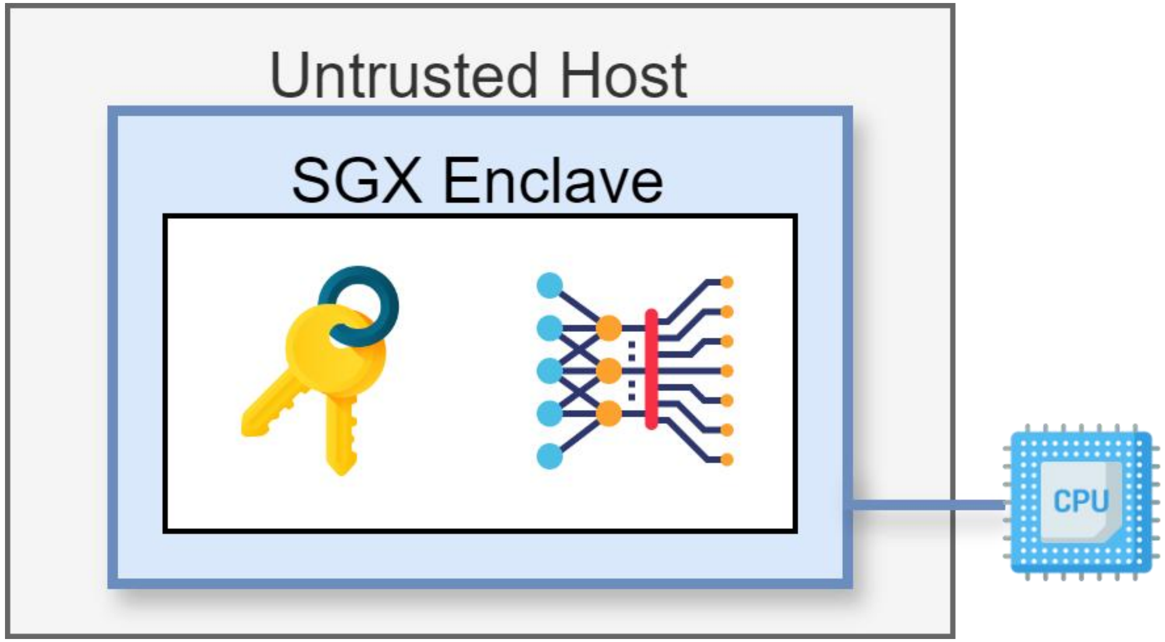
中国科学院大学
University of Chinese Academy of Sciences

中国科学院信息工程研究所
INSTITUTE OF INFORMATION ENGINEERING,CAS

#NDSSSymposium2024

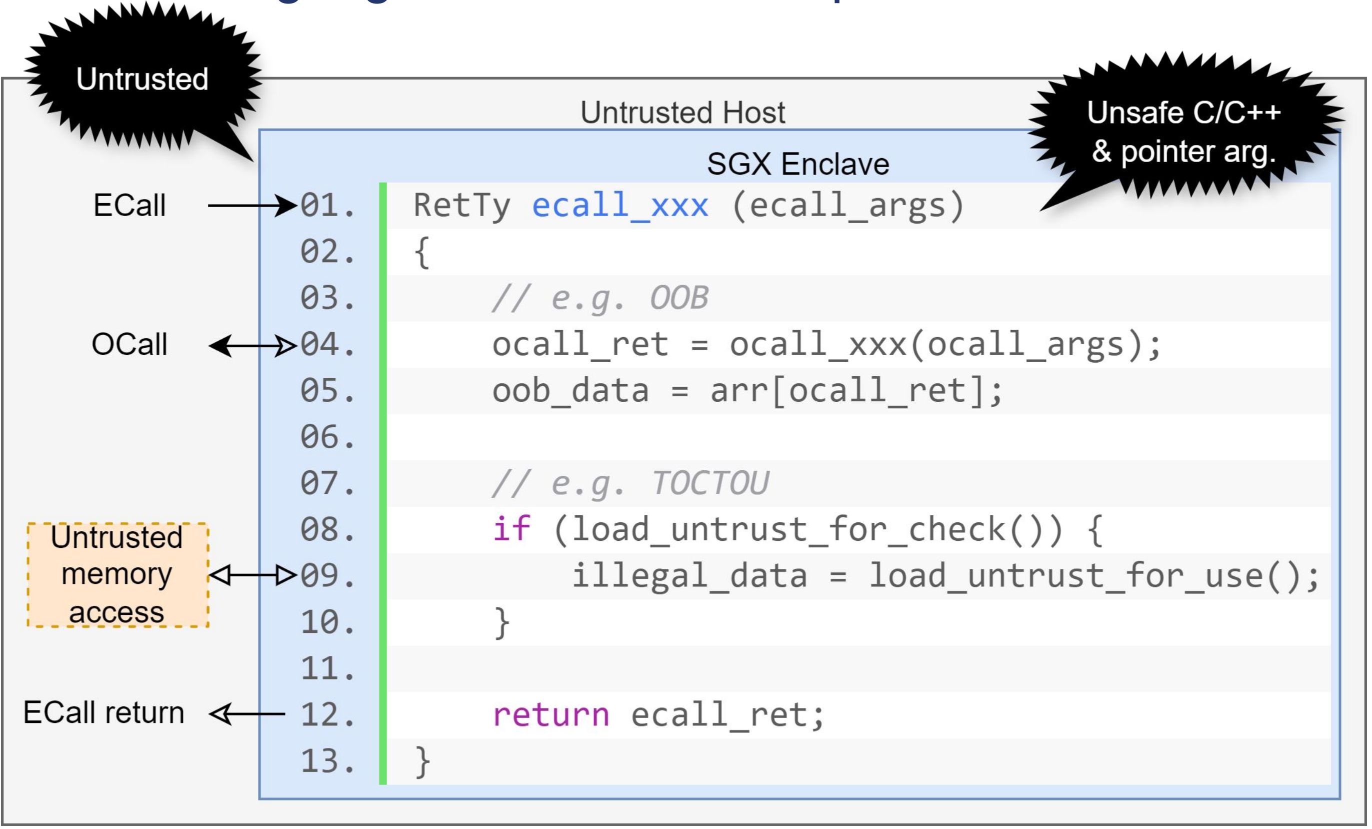
SGX Application

- Applications use hardware capabilities to defend against untrusted host.



Problem

- Applications are unaware of specific threat model.
- Memory unsafe language exacerbates the problem.



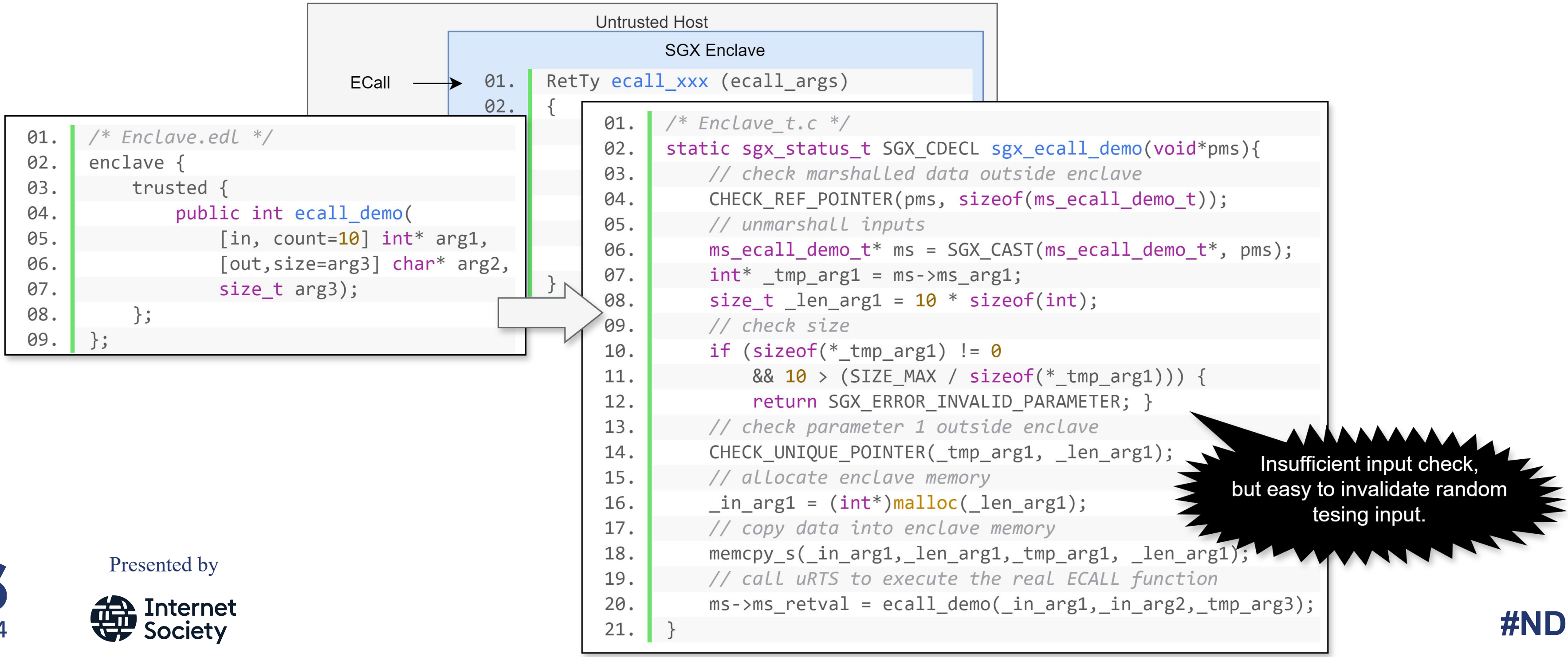
```
RetTy ecall_xxx (ecall_args)
{
    // e.g. OOB
    ocall_ret = ocall_xxx(ocall_args);
    oob_data = arr[ocall_ret];

    // e.g. TOCTOU
    if (load_untrust_for_check()) {
        illegal_data = load_untrust_for_use();
    }

    return ecall_ret;
}
```

Problem: Insufficient Input Check

- SGX EDL only performs the basic checking rules, enclave itself needs a deeper check.
- *But it is easy to invalidate fuzzing input.*



```
RetTy ecall_xxx (ecall_args)
{
}
```

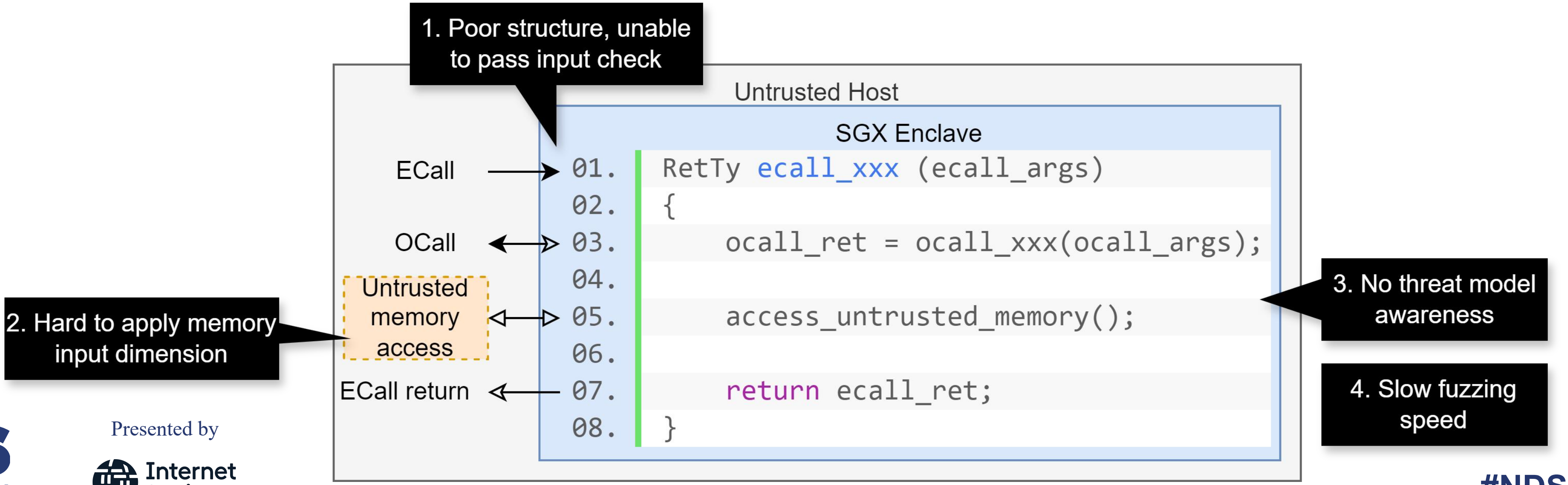
```
/* Enclave.edl */
enclave {
    trusted {
        public int ecall_demo(
            [in, count=10] int* arg1,
            [out,size=arg3] char* arg2,
            size_t arg3);
    };
};
```

```
/* Enclave_t.c */
static sgx_status_t SGX_CDECL sgx_ecall_demo(void*pms){
    // check marshalled data outside enclave
    CHECK_REF_POINTER(pms, sizeof(ms_ecall_demo_t));
    // unmarshall inputs
    ms_ecall_demo_t* ms = SGX_CAST(ms_ecall_demo_t*, pms);
    int* _tmp_arg1 = ms->ms_arg1;
    size_t _len_arg1 = 10 * sizeof(int);
    // check size
    if (sizeof(*_tmp_arg1) != 0
        && 10 > (SIZE_MAX / sizeof(*_tmp_arg1))) {
        return SGX_ERROR_INVALID_PARAMETER; }
    // check parameter 1 outside enclave
    CHECK_UNIQUE_POINTER(_tmp_arg1, _len_arg1);
    // allocate enclave memory
    _in_arg1 = (int*)malloc(_len_arg1);
    // copy data into enclave memory
    memcpy_s(_in_arg1,_len_arg1,_tmp_arg1, _len_arg1);
    // call uRTS to execute the real ECALL function
    ms->ms_retval = ecall_demo(_in_arg1,_in_arg2,_tmp_arg3);
}
```

Insufficient input check, but easy to invalidate random tesing input.

Related Works

1. **Static:** TeeRex [SEC'20] and COIN attacks [ASPLOS'20] exploit symbolic execution but face state explosion in large-scale applications.
2. **Dynamic:** SGXFuzz [SEC'22] identifies input structures via page fault feedback, while FuzzSGX [EuroS&P'23] relies on host mutation, and they can only detect crashes or memory corruption.



```
RetTy ecall_xxx (ecall_args)
{
    ocall_ret = ocall_xxx(ocall_args);

    access_untrusted_memory();

    return ecall_ret;
}
```

1. Poor structure, unable to pass input check

2. Hard to apply memory input dimension

3. No threat model awareness

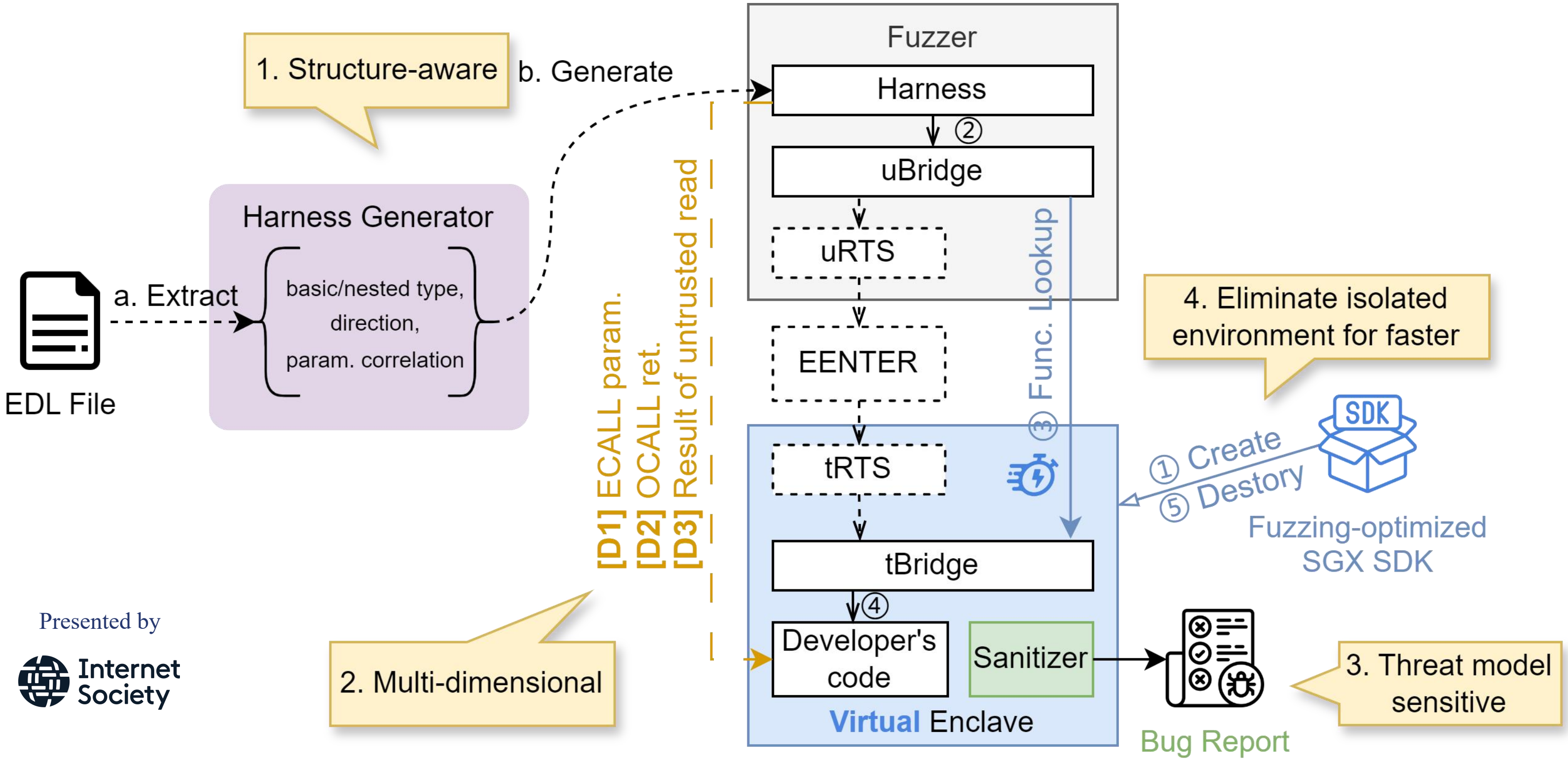
4. Slow fuzzing speed

EnclaveFuzz Overview

1. & 2. A multi-dimensional structure-aware fuzzing harness.

3. A sanitizer sensitive to the SGX threat model.

4. Faster SDK optimized for fuzzing by eliminating isolated environment.



1. Structure-aware

b. Generate

Fuzzer

Harness

②

uBridge

uRTS

EENTER

tRTS

tBridge

④

Developer's code

Sanitizer

Virtual Enclave

③ Func. Lookup

Harness Generator

basic/nested type, direction, param. correlation

a. Extract

EDL File

[D1] ECALL param.

[D2] OCALL ret.

[D3] Result of untrusted read

2. Multi-dimensional

4. Eliminate isolated environment for faster

① Create

⑤ Destory

SDK

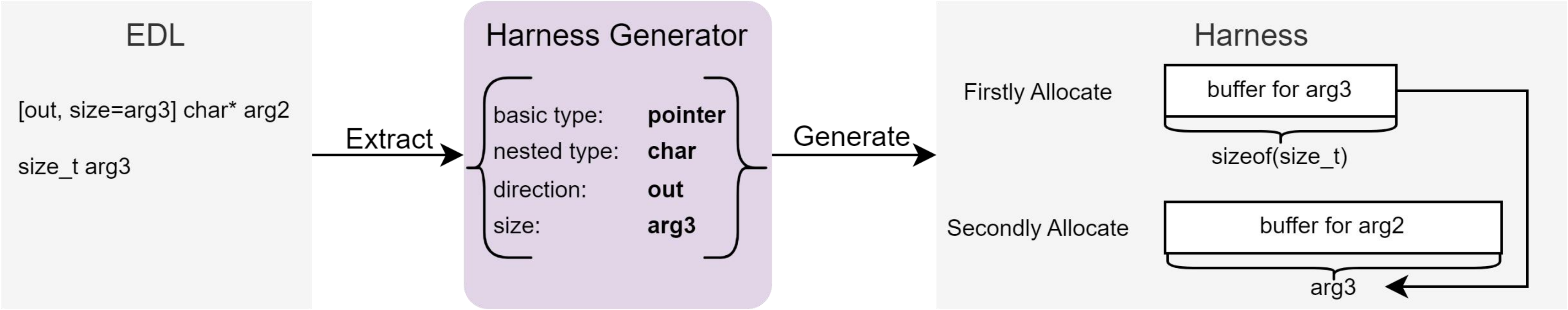
Fuzzing-optimized SGX SDK

Bug Report

3. Threat model sensitive

Structure-aware Fuzzing

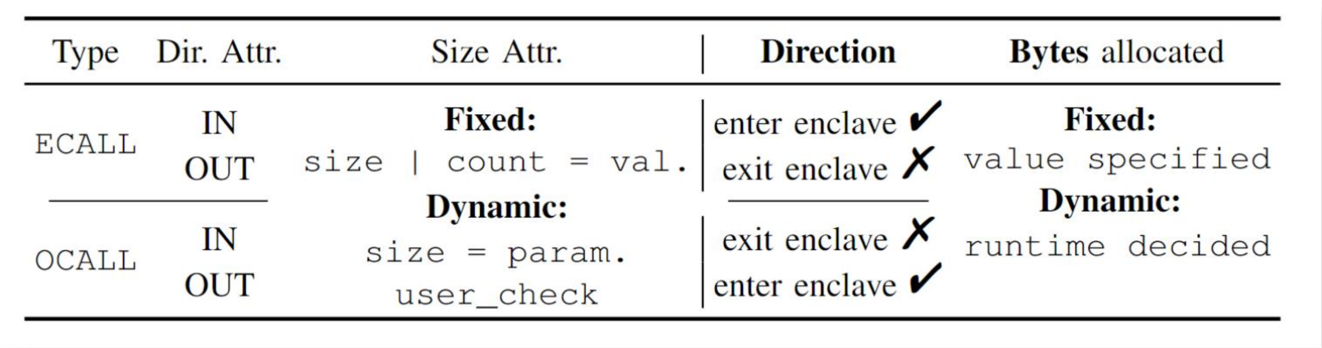
- Extracts information from EDL and generates a structure-aware fuzzing harness, to pass basic input check.



Structure Generation Policy

- Analyzes parameters and handles data directions based on EDL attributes.
- Dynamically provide a random size first and then the buffer when processing *user_check*.

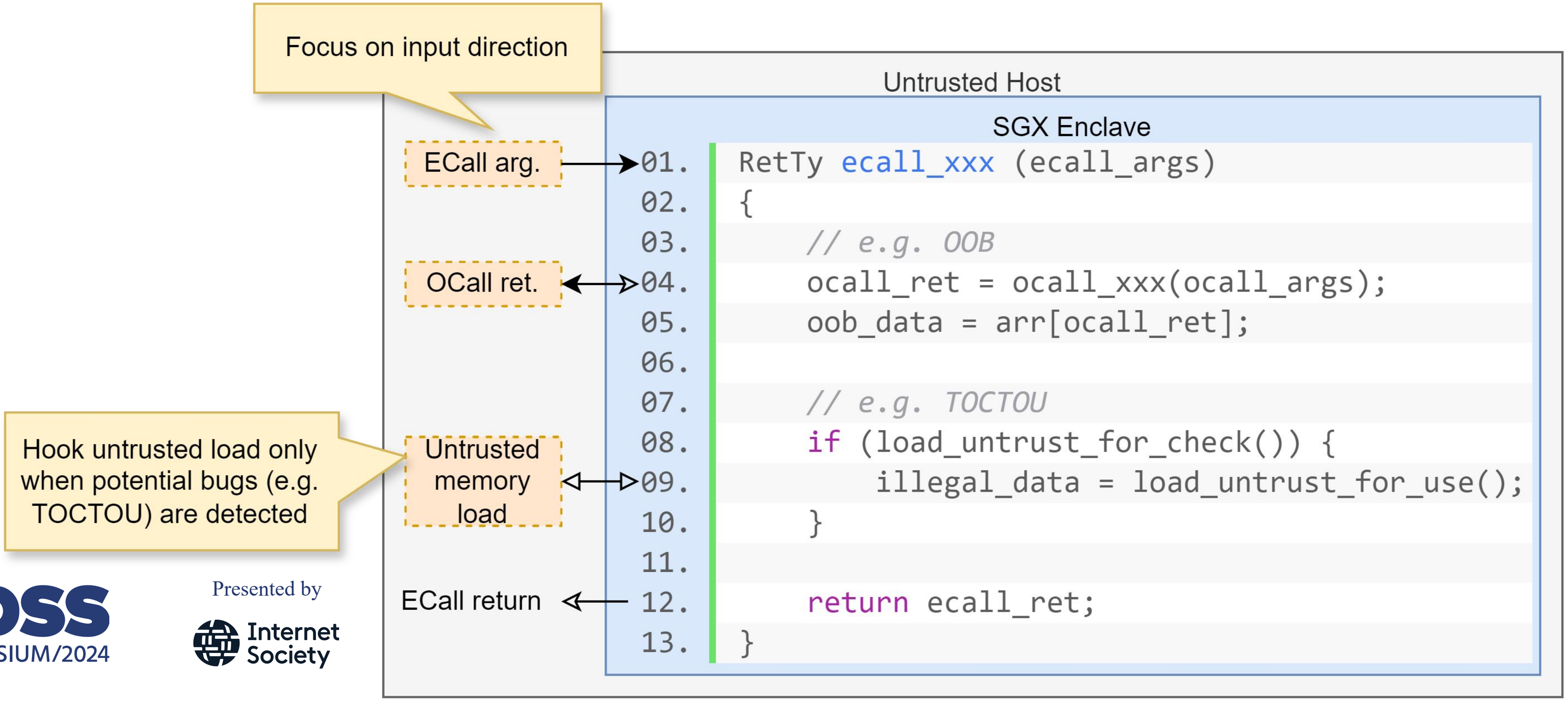
See paper for more details.



Type	Dir. Attr.	Size Attr.	**Direction**	**Bytes allocated**
ECALL	IN OUT	**Fixed:** size | count = val. **Dynamic:** size = param. user_check	enter enclave ✓ exit enclave ✗	**Fixed:** value specified **Dynamic:** runtime decided
OCALL	IN OUT		exit enclave ✗ enter enclave ✓	

Multi-dimensional Fuzzing

- Only untrusted input is provided.
- Besides E/OCall, hooks untrusted memory load only when potential bugs are detected to avoid huge overhead.



Focus on input direction

Hook untrusted load only when potential bugs (e.g. TOCTOU) are detected

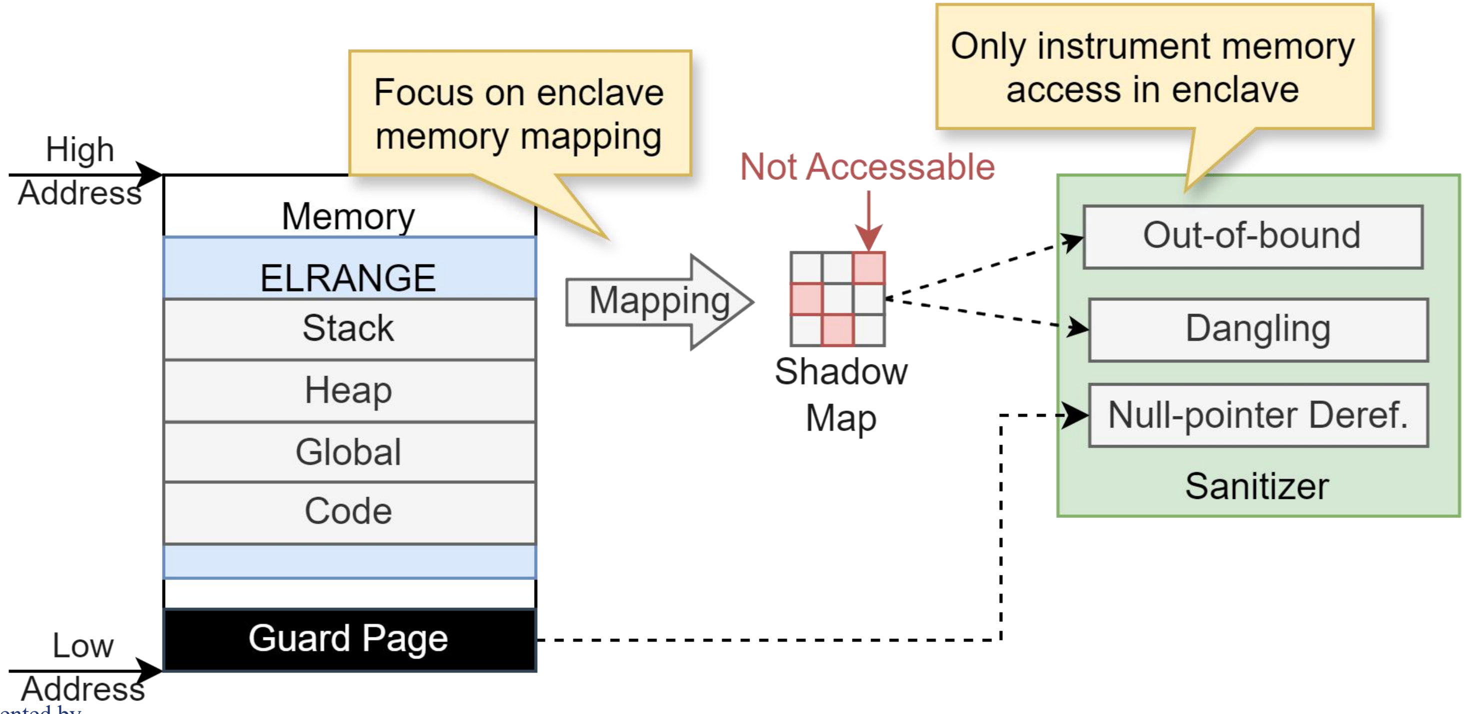
```
RetTy ecall_xxx (ecall_args)
{
    // e.g. OOB
    ocall_ret = ocall_xxx(ocall_args);
    oob_data = arr[ocall_ret];

    // e.g. TOCTOU
    if (load_untrust_for_check()) {
        illegal_data = load_untrust_for_use();
    }

    return ecall_ret;
}
```

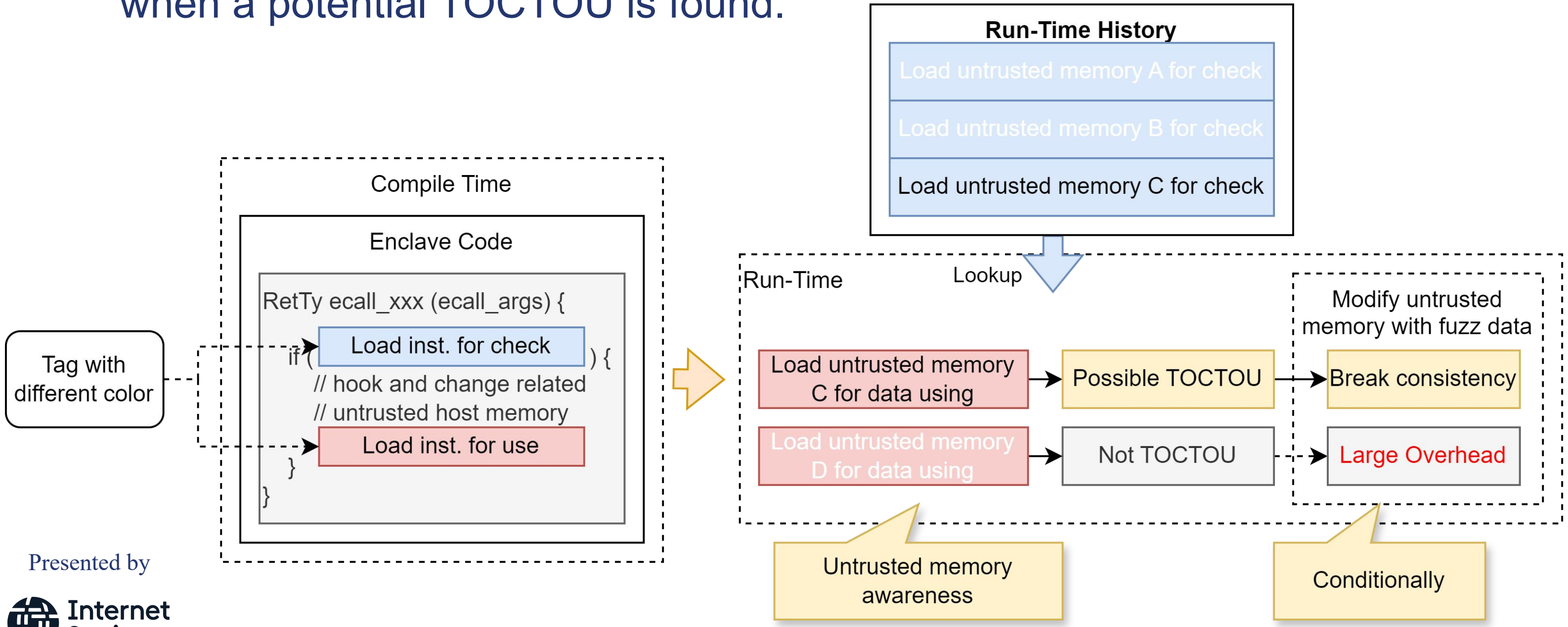
Vulnerability Detection

1. Miagrated ASan for memory corruption detection, but focus on enclave trusted memory.



Vulnerability Detection

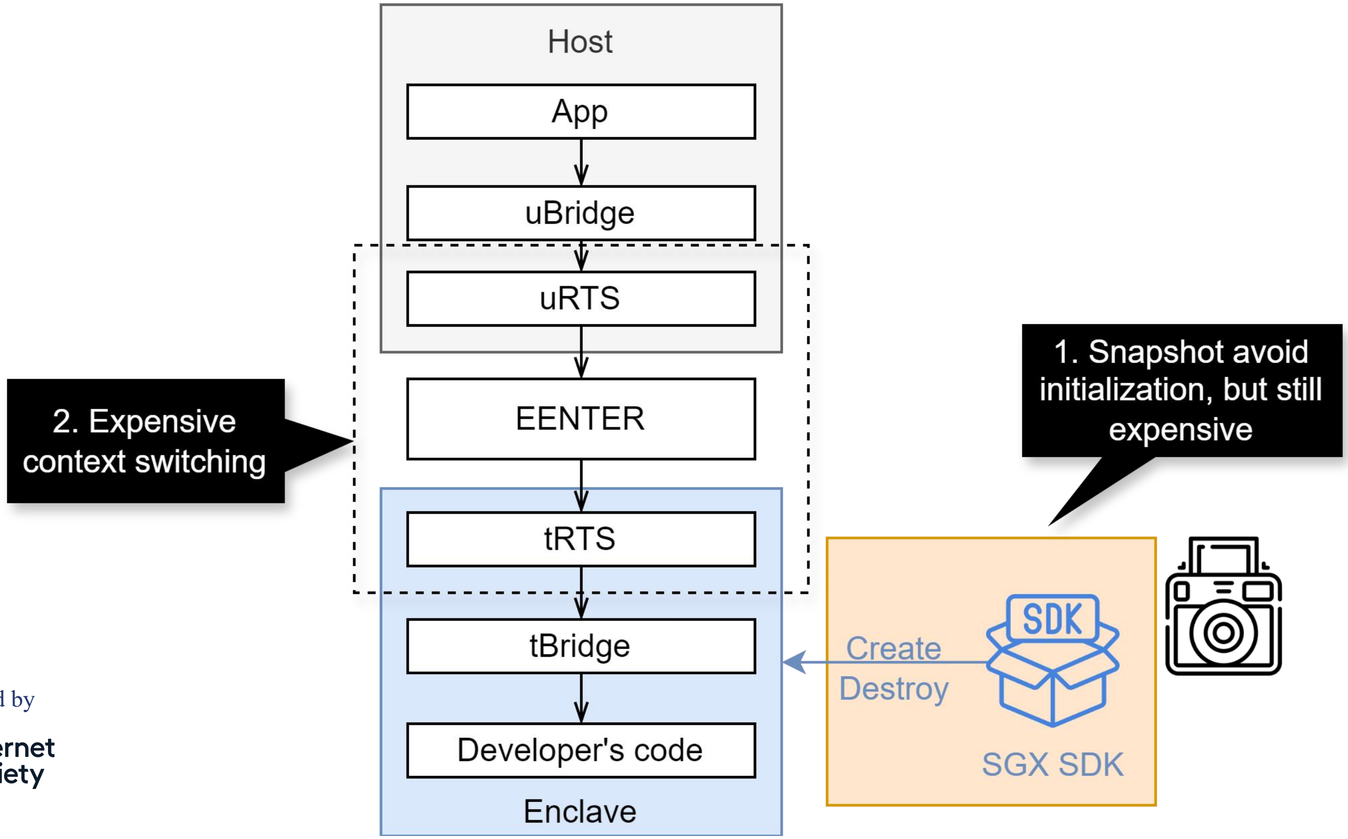
2. TOCTOU detection
   a) Different tags are given to *load* instructions during compilation.
   b) Hook *load* at runtime to detect possible TOCTOU with untrusted memory awareness.
   c) To avoid huge overhead, only modify untrusted memory to break consistency when a potential TOCTOU is found.



Run-Time History
Load untrusted memory A for check
Load untrusted memory B for check
Load untrusted memory C for check
Compile Time
Enclave Code
RetTy ecall_xxx (ecall_args) {
if (
Load inst. for check
) {
// hook and change related
// untrusted host memory
Load inst. for use
}
}
Tag with different color
Run-Time
Lookup
Modify untrusted memory with fuzz data
Load untrusted memory C for data using
Possible TOCTOU
Break consistency
Load untrusted memory D for data using
Not TOCTOU
Large Overhead
Untrusted memory awareness
Conditionally

Slow fuzzing speed of related works

- Snapshot mode is still slow compared to persistent mode.
- Context switching is also expensive.

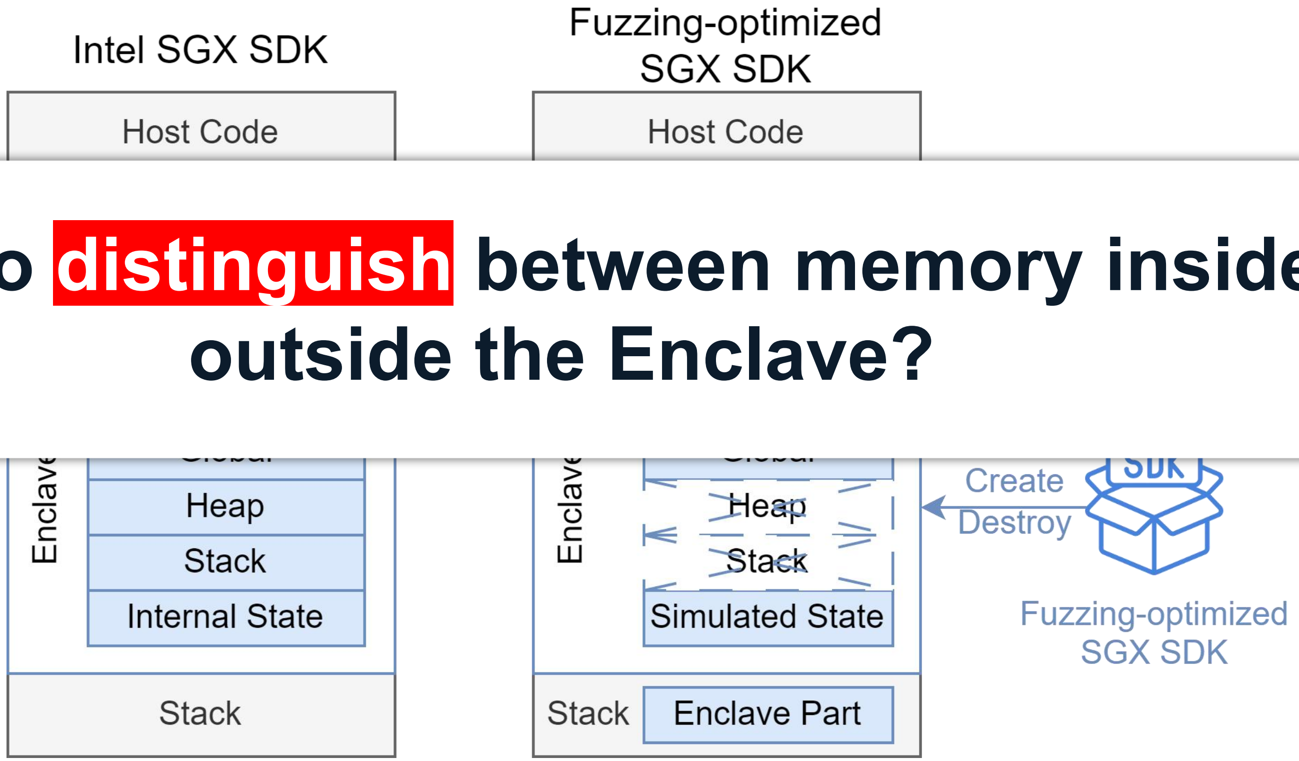


2. Expensive context switching

1. Snapshot avoid initialization, but still expensive

Eliminate Isolated Environment

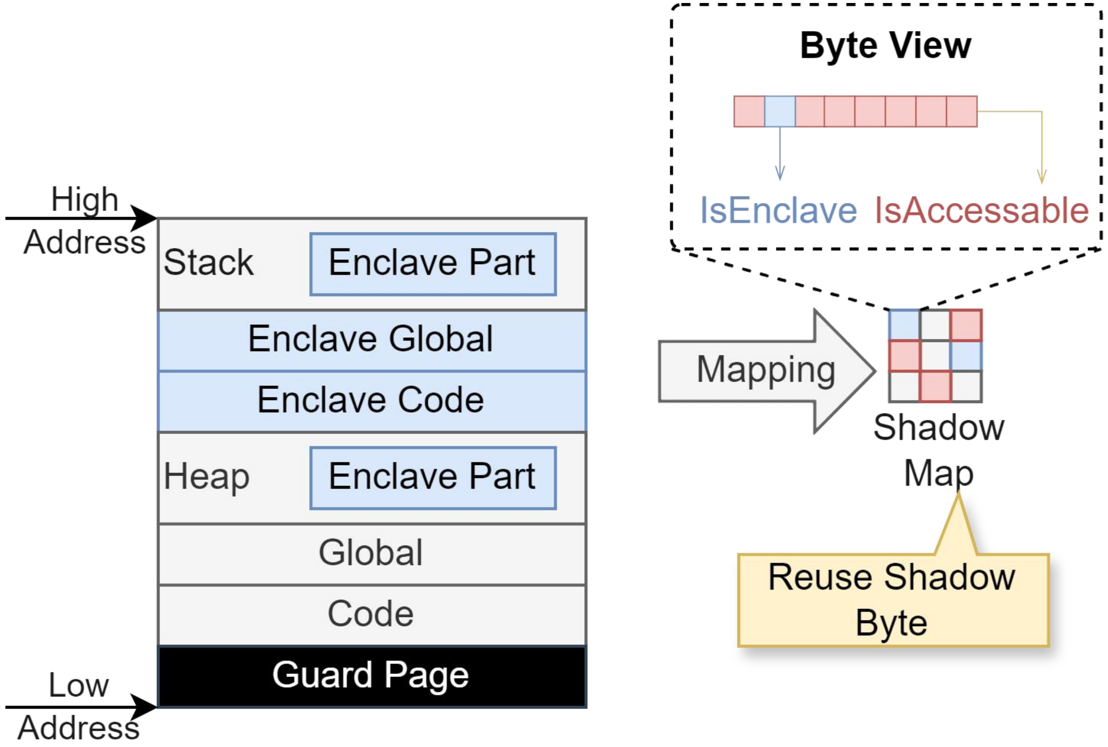
1. Remove independent memory to reduce creation/destruction overhead.



But how to distinguish between memory inside and outside the Enclave?

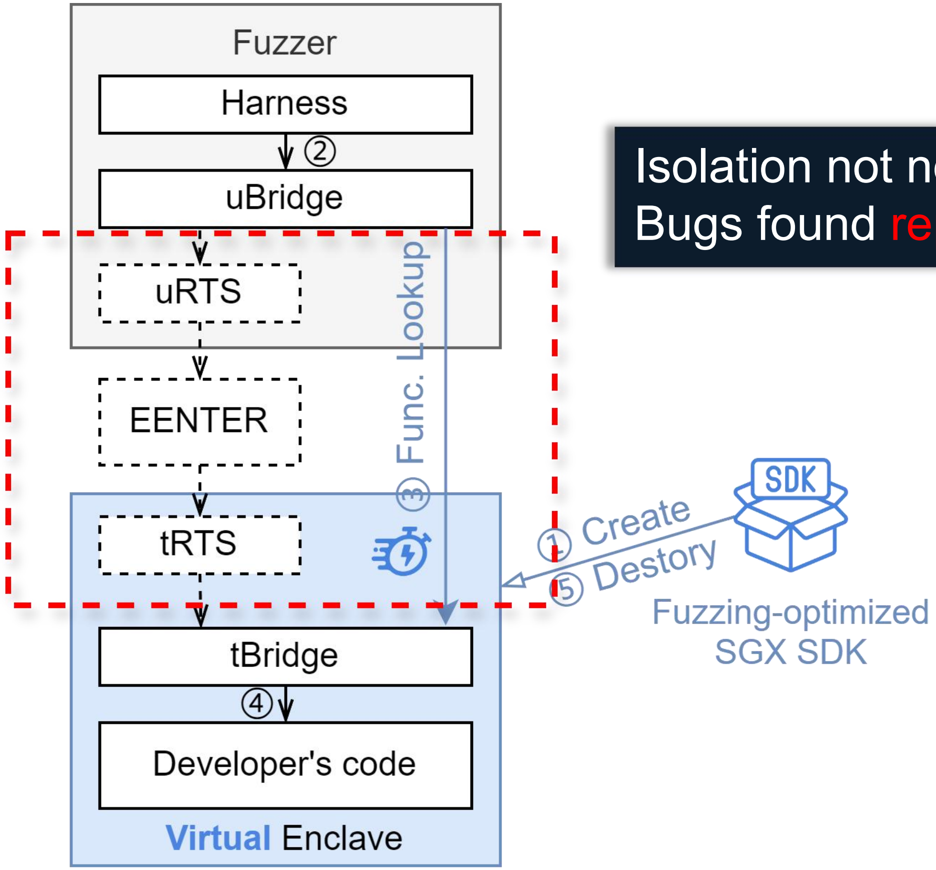
Eliminate Isolated Environment

1. Remove independent memory to reduce creation/destruction overhead.
2. Reuse shadow byte to distinguish memory inside and outside the Enclave, i.e. enclave memory awareness.



Eliminate Isolated Environment

3. Replace expensive context switches with function table lookups

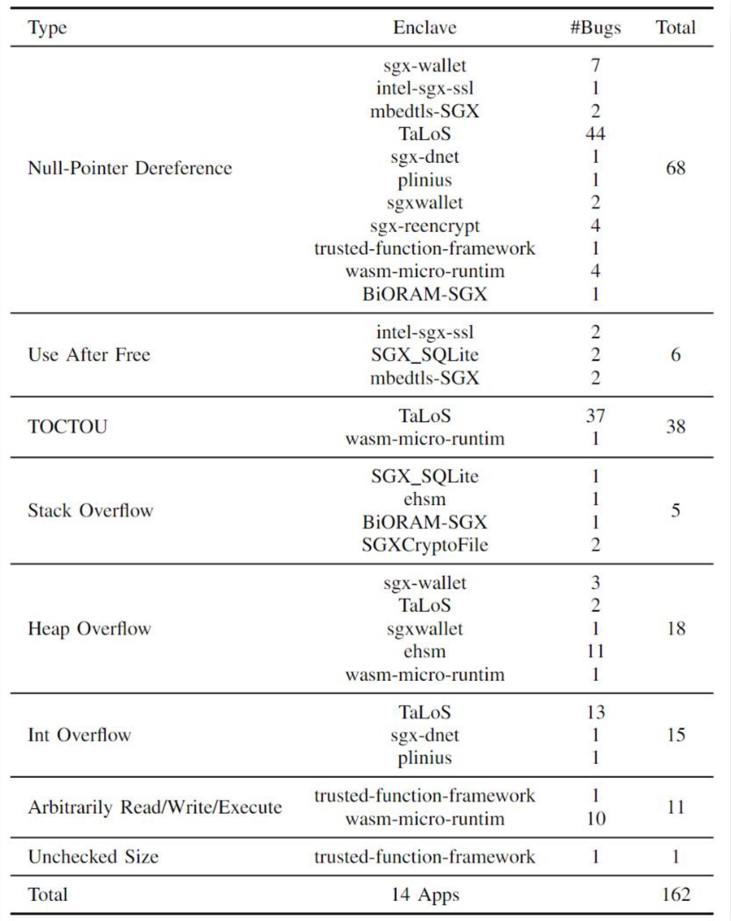


Isolation not necessary for fuzzing.
Bugs found reproduced in HW mode.

Bugs Found

- EnclaveFuzz found 162 bugs in 14/20 real-world open source enclaves.

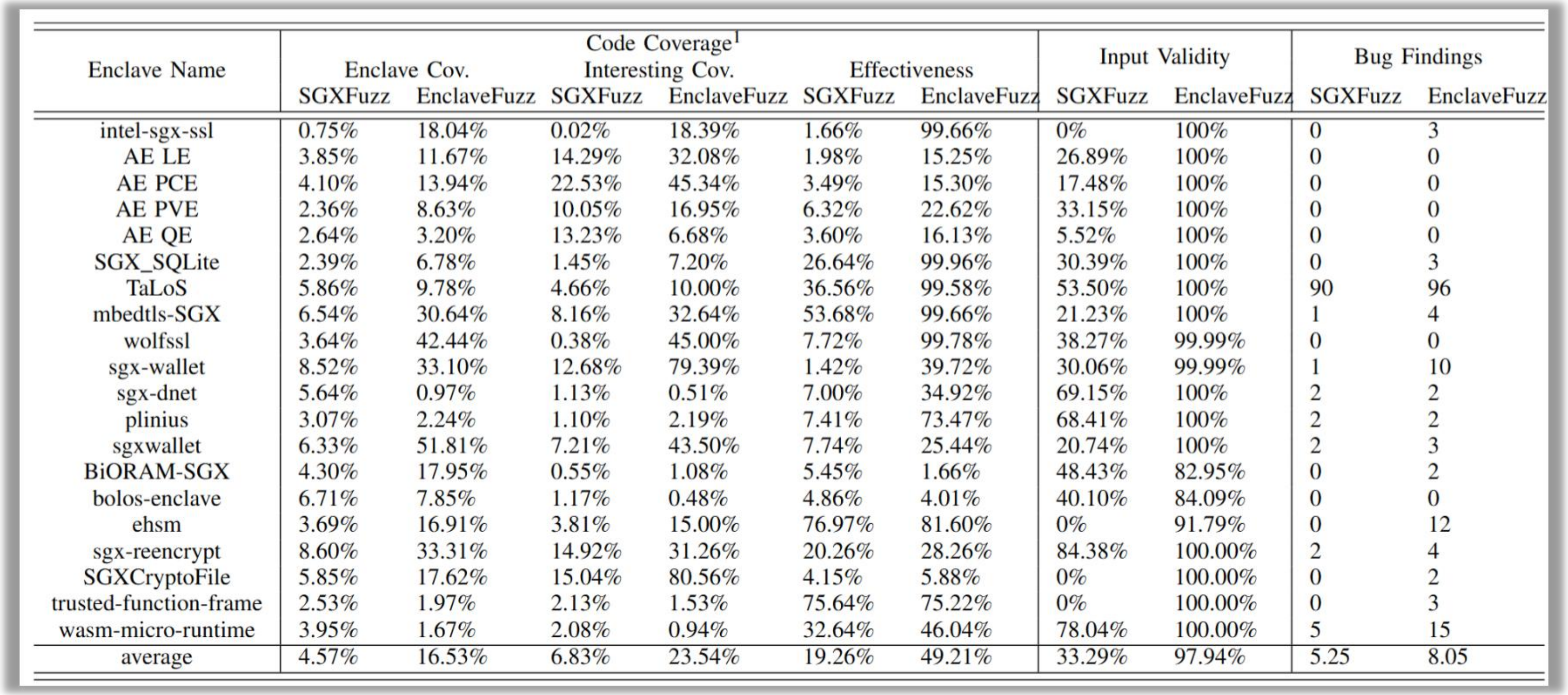
Mostly Null-Pointer Dereference & TOCTOU. SGX threat model was overlooked, especially cross-boundary pointers.



Type	Enclave	#Bugs	Total
Null-Pointer Dereference	sgx-wallet	7	68
	intel-sgx-ssl	1	
	mbedtls-SGX	2	
	TaLoS	44	
	sgx-dnet	1	
	plinius	1	
	sgxwallet	2	
	sgx-reencrypt	4	
	trusted-function-framework	1	
	wasm-micro-runtim	4	
	BiORAM-SGX	1	
Use After Free	intel-sgx-ssl	2	6
	SGX_SQLite	2	
	mbedtls-SGX	2	
TOCTOU	TaLoS	37	38
	wasm-micro-runtim	1	
Stack Overflow	SGX_SQLite	1	5
	ehsm	1	
	BiORAM-SGX	1	
	SGXCryptoFile	2	
Heap Overflow	sgx-wallet	3	18
	TaLoS	2	
	sgxwallet	1	
	ehsm	11	
	wasm-micro-runtim	1	
Int Overflow	TaLoS	13	15
	sgx-dnet	1	
	plinius	1	
Arbitrarily Read/Write/Execute	trusted-function-framework	1	11
	wasm-micro-runtim	10	
Unchecked Size	trusted-function-framework	1	1
Total	14 Apps		162

Compare with SOTA

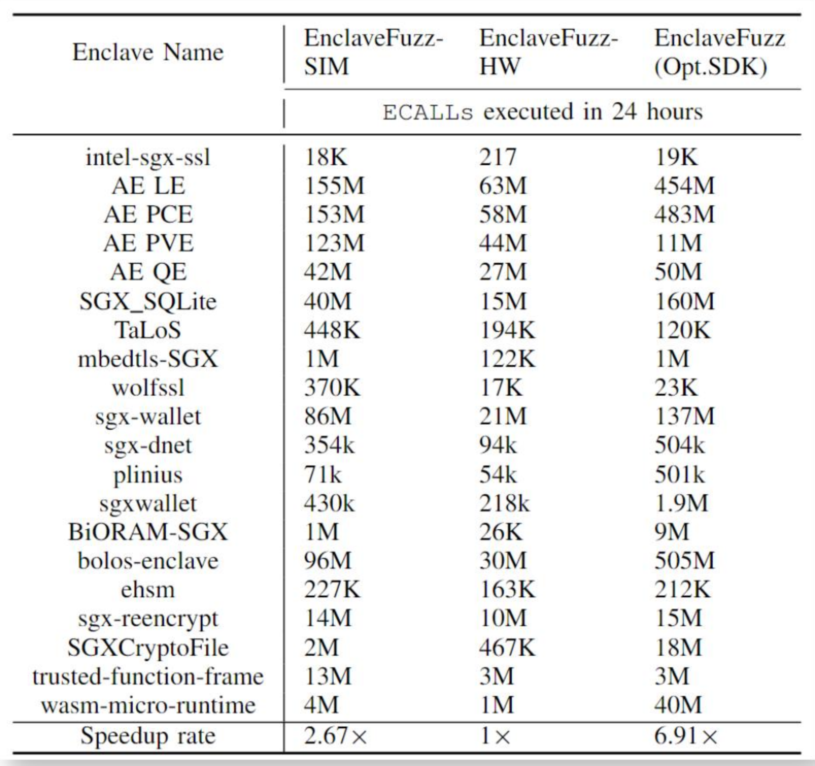
1. Effectiveness of structured test cases is nearly 3 times that of SOTA, and the coverage is nearly 4 times.



Enclave Name	Code Coverage[1]						Input Validity		Bug Findings	
	Enclave Cov.		Interesting Cov.		Effectiveness					
	SGXFuzz	EnclaveFuzz	SGXFuzz	EnclaveFuzz	SGXFuzz	EnclaveFuzz	SGXFuzz	EnclaveFuzz	SGXFuzz	EnclaveFuzz
intel-sgx-ssl	0.75%	18.04%	0.02%	18.39%	1.66%	99.66%	0%	100%	0	3
AE LE	3.85%	11.67%	14.29%	32.08%	1.98%	15.25%	26.89%	100%	0	0
AE PCE	4.10%	13.94%	22.53%	45.34%	3.49%	15.30%	17.48%	100%	0	0
AE PVE	2.36%	8.63%	10.05%	16.95%	6.32%	22.62%	33.15%	100%	0	0
AE QE	2.64%	3.20%	13.23%	6.68%	3.60%	16.13%	5.52%	100%	0	0
SGX_SQLite	2.39%	6.78%	1.45%	7.20%	26.64%	99.96%	30.39%	100%	0	3
TaLoS	5.86%	9.78%	4.66%	10.00%	36.56%	99.58%	53.50%	100%	90	96
mbedtls-SGX	6.54%	30.64%	8.16%	32.64%	53.68%	99.66%	21.23%	100%	1	4
wolfssl	3.64%	42.44%	0.38%	45.00%	7.72%	99.78%	38.27%	99.99%	0	0
sgx-wallet	8.52%	33.10%	12.68%	79.39%	1.42%	39.72%	30.06%	99.99%	1	10
sgx-dnet	5.64%	0.97%	1.13%	0.51%	7.00%	34.92%	69.15%	100%	2	2
plinius	3.07%	2.24%	1.10%	2.19%	7.41%	73.47%	68.41%	100%	2	2
sgxwallet	6.33%	51.81%	7.21%	43.50%	7.74%	25.44%	20.74%	100%	2	3
BiORAM-SGX	4.30%	17.95%	0.55%	1.08%	5.45%	1.66%	48.43%	82.95%	0	2
bolos-enclave	6.71%	7.85%	1.17%	0.48%	4.86%	4.01%	40.10%	84.09%	0	0
ehsm	3.69%	16.91%	3.81%	15.00%	76.97%	81.60%	0%	91.79%	0	12
sgx-reencrypt	8.60%	33.31%	14.92%	31.26%	20.26%	28.26%	84.38%	100.00%	2	4
SGXCryptoFile	5.85%	17.62%	15.04%	80.56%	4.15%	5.88%	0%	100.00%	0	2
trusted-function-frame	2.53%	1.97%	2.13%	1.53%	75.64%	75.22%	0%	100.00%	0	3
wasm-micro-runtime	3.95%	1.67%	2.08%	0.94%	32.64%	46.04%	78.04%	100.00%	5	15
average	4.57%	16.53%	6.83%	23.54%	19.26%	49.21%	33.29%	97.94%	5.25	8.05

Compare with SOTA

2. Improve the test speed of real-world applications by nearly 7 times.



Enclave Name	EnclaveFuzz-SIM	EnclaveFuzz-HW	EnclaveFuzz (Opt.SDK)
	ECALLs executed in 24 hours		
intel-sgx-ssl	18K	217	19K
AE LE	155M	63M	454M
AE PCE	153M	58M	483M
AE PVE	123M	44M	11M
AE QE	42M	27M	50M
SGX_SQLite	40M	15M	160M
TaLoS	448K	194K	120K
mbedtls-SGX	1M	122K	1M
wolfssl	370K	17K	23K
sgx-wallet	86M	21M	137M
sgx-dnet	354k	94k	504k
plinius	71k	54k	501k
sgxwallet	430k	218k	1.9M
BiORAM-SGX	1M	26K	9M
bolos-enclave	96M	30M	505M
ehsm	227K	163K	212K
sgx-reencrypt	14M	10M	15M
SGXCryptoFile	2M	467K	18M
trusted-function-frame	13M	3M	3M
wasm-micro-runtime	4M	1M	40M
Speedup rate	2.67×	1×	6.91×

See paper for more ablation studies.

Takeaway

- EnclaveFuzz is a multi-dimensional, structure-aware fuzzer for SGX applications, featuring a threat-model-aware sanitizer and an SDK that eliminates the isolated environment.

https://github.com/vul337/EnclaveFuzz

LAB https://netsec.ccert.edu.cn/vul337