EnclaveFuzz: Finding Vulnerabilities in SGX Applications

Speaker: Liheng Chen

Supervisor: Chao Zhang

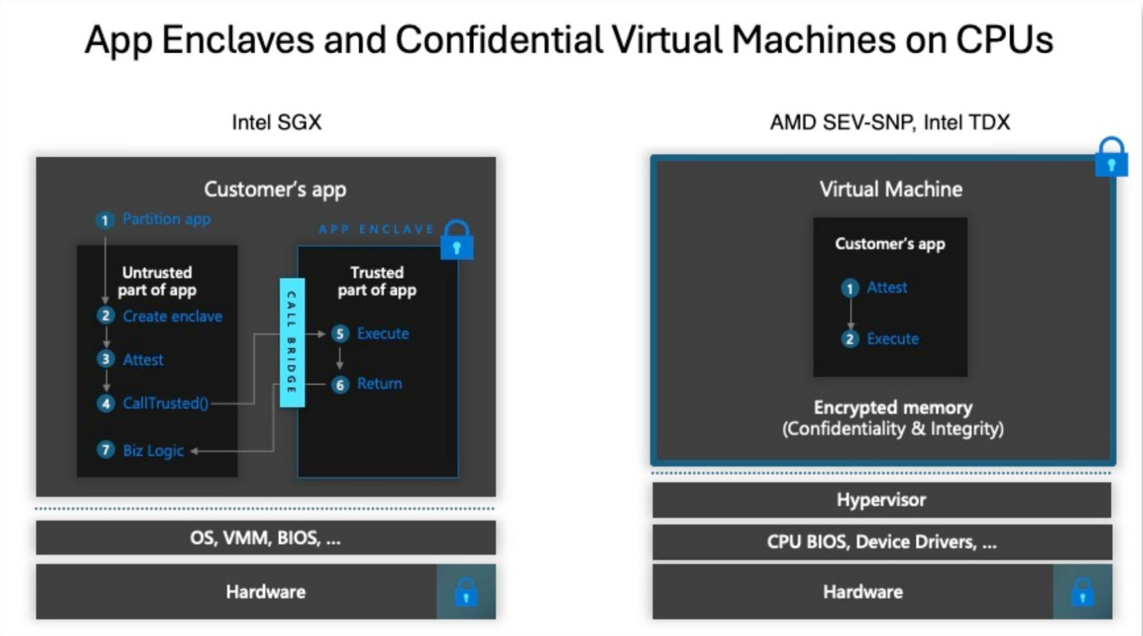
Email: 791960492@qq.com / leonechen9@gmail.com

Home Page: https://leonechen.github.io

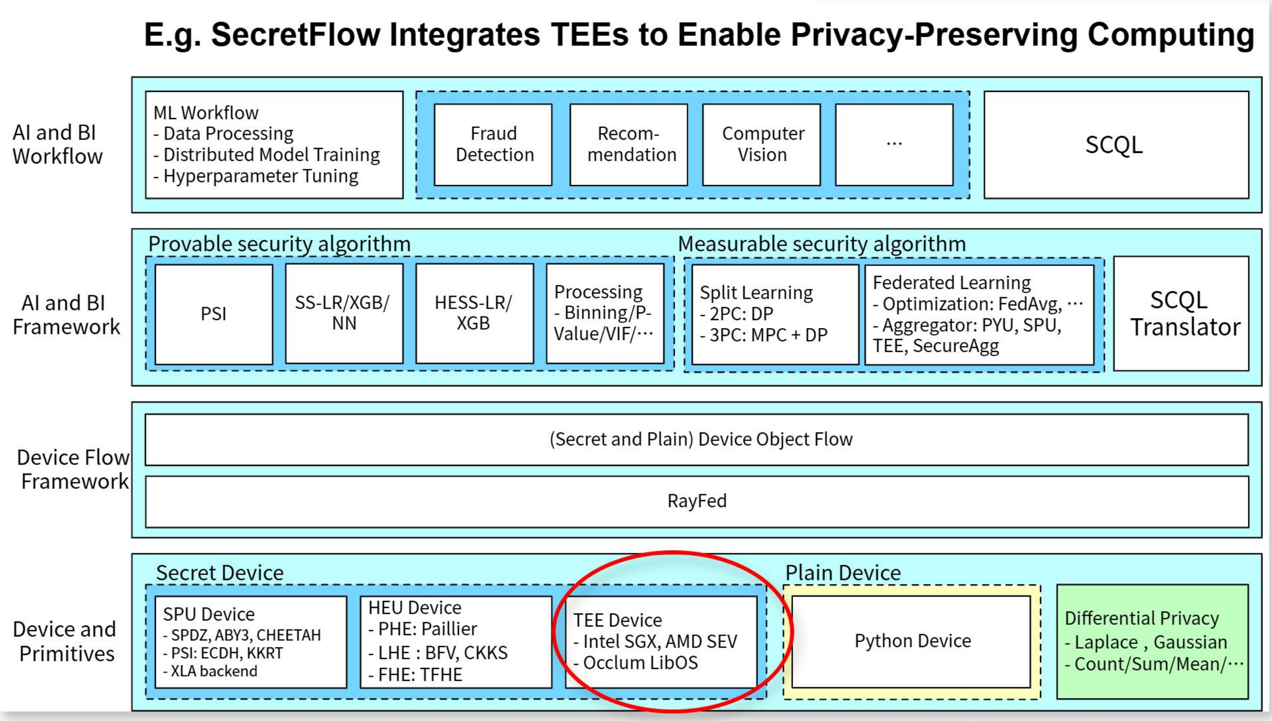
TEE and Its Applications

- **Trusted Execution Environment:** a segregated area of memory and CPU that's protected from the rest by using access control and encryption, e.g. enclaves and confidential VMs.
- **Application Scenario:** cloud computing, privacy-preserving computing...

× 蚂蚁集团 ANT GROUP
Security Evaluation



https://learn.microsoft.com/en-us/azure/confidential-computing/trusted-execution-environment



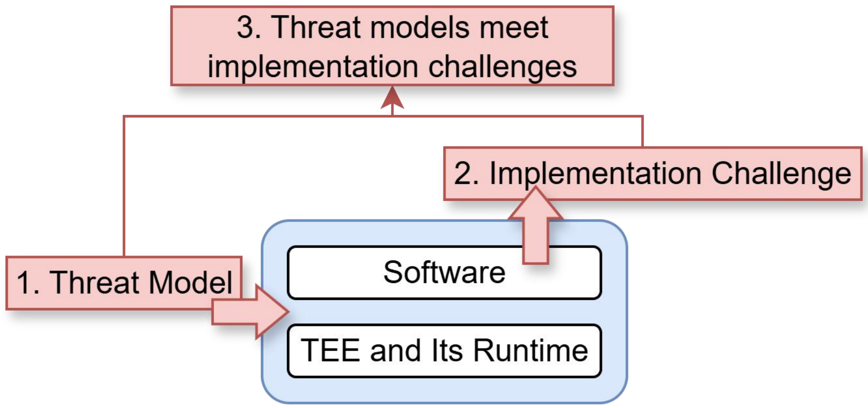
https://www.secretflow.org.cn/zh-CN/docs/secretflow/v1.12.0b0/

Why TEE ≠ Automatically Secure

- **Source of Threat:** Beyond the standard TEE threat model, TEE application security also faces implementation-level challenges.
  1. **Misaligned Trust Assumptions:** Developers misunderstand/overlook TEEs' threat models.
  2. **Implementation Challenge:** Implementation issues are difficult to avoid (especially in C/C++).
  3. **Threat Models Meet Implementation Challenge →** SGX-specific vulnerability.

Our Topic

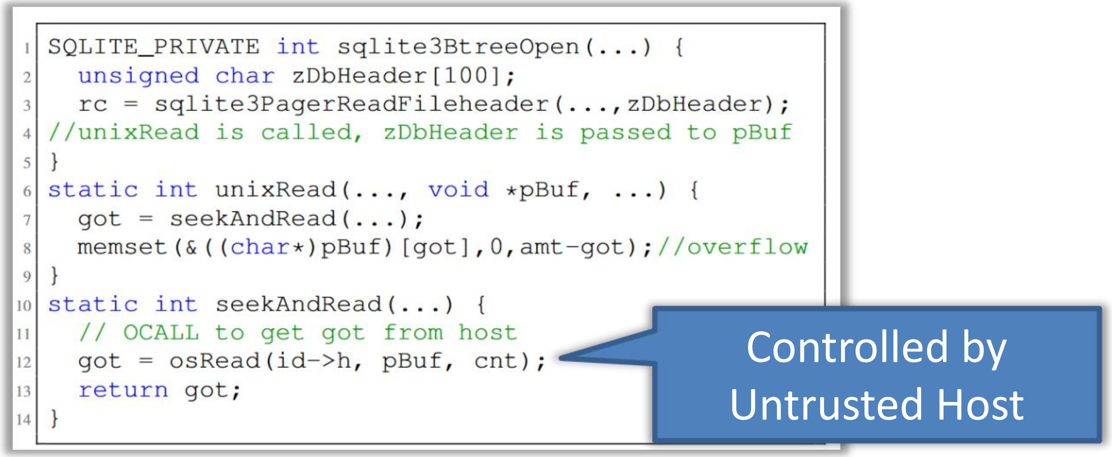
- (Side channels are out of our scope.)

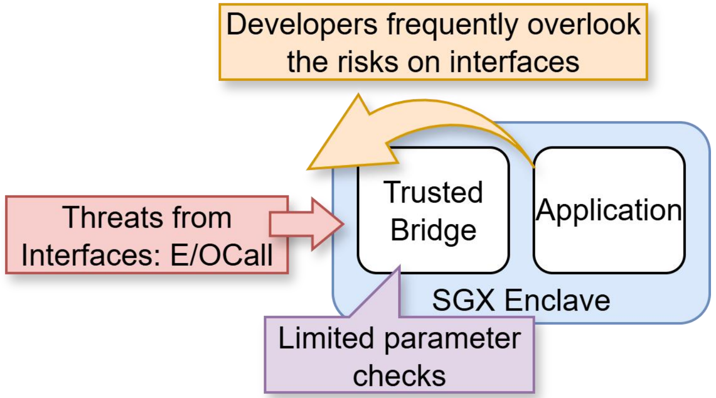


Source of Threat

SGX-Specific Vulnerability

- **Risk:** SGX developers may overlook threat model, leaving issues such as stack overflow unchecked (left fig.), then we are able to hijack control flow etc.
- **Summary:** With ELRANGE-targeted shadow map, we discovered 162 vulnerabilities in 14 SGX applications, highlighting the threat model is often overlooked.
- **Lessons:**
  1. Cross-boundary data/pointer should be carefully handled.
  2. Memory safe language (e.g. Rust) is important.





Test Case Prepare

- **Insight:**
  1. Test cases derived via #PF inference (e.g. SGXFuzz) rarely pass TBridge's simple checks. However, many SGX apps are open-source, EDL can be used to craft inputs that avoid rejection.
  2. Data read from untrusted memory is also untrusted. We can instrument (at LLVM IR) to intercept untrusted memory accesses for testing.

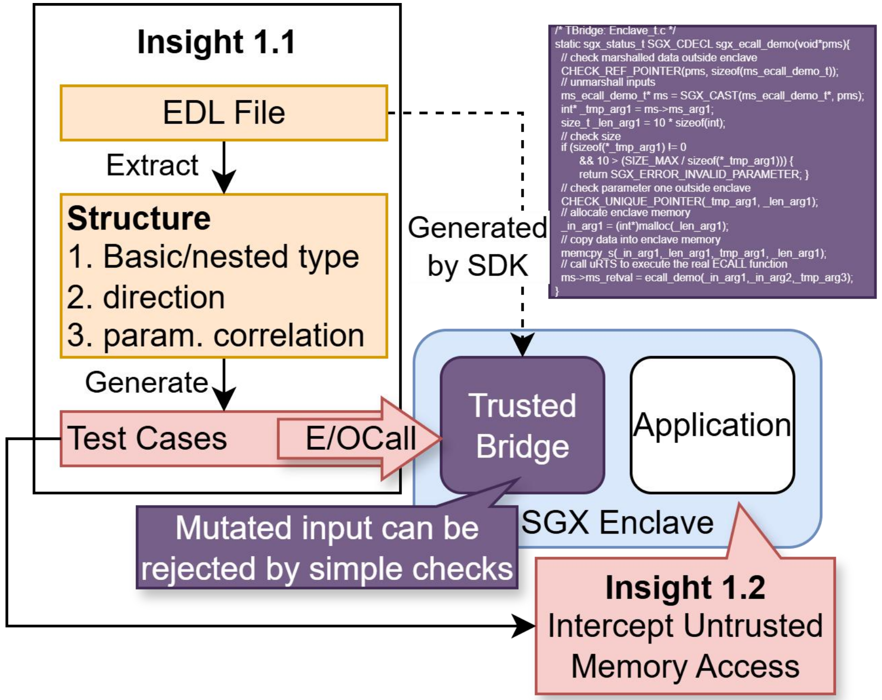


TABLE IV: Experiment Results

Enclave Name	Code Coverage[1] Enclave Cov. SGXFuzz	Enclave Cov. EnclaveFuzz	Interesting Cov. SGXFuzz	Interesting Cov. EnclaveFuzz	Effectiveness SGXFuzz	Effectiveness EnclaveFuzz	Input Validity SGXFuzz	Input Validity EnclaveFuzz
intel-sgx-ssl	0.75%	18.04%	0.02%	18.39%	1.66%	99.66%	0%	100%
AE LE	3.85%	11.67%	14.29%	32.08%	1.98%	15.25%	26.89%	100%
AE PCE	4.10%	13.94%	22.53%	45.34%	3.49%	15.30%	17.48%	100%
AE PVE	2.36%	8.63%	10.05%	16.95%	6.32%	22.62%	33.15%	100%
AE QE	2.64%	3.20%	13.23%	6.68%	3.60%	16.13%	5.52%	100%
SGX_SQLite	2.39%	6.78%	1.45%	7.20%	26.64%	99.96%	30.39%	100%
TaLoS	5.86%	9.78%	4.66%	10.00%	36.56%	99.58%	53.50%	100%
mbedtls-SGX	6.54%	30.64%	8.16%	32.64%	53.68%	99.66%	21.23%	100%
wolfssl	3.64%	42.44%	0.38%	45.00%	7.72%	99.78%	38.27%	99.99%
sgx-walle	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	30.06%	99.99%
sgx-dnet	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	69.15%	100%
plinius	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	68.41%	100%
sgxwalle	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	20.74%	100%
BiORAM-S	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	48.43%	82.95%
bolos-encla	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	40.10%	84.09%
ehsm	3.69%	16.91%	3.81%	15.00%	76.97%	81.60%	0%	91.79%
sgx-reencrypt	8.60%	33.31%	14.92%	31.26%	20.26%	28.26%	84.38%	100.00%
SGXCryptoFile	5.85%	17.62%	15.04%	80.56%	4.15%	5.88%	0%	100.00%
trusted-function-frame	2.53%	1.97%	2.13%	1.53%	75.64%	75.22%	0%	100.00%
wasm-micro-runtime	3.95%	1.67%	2.08%	0.94%	32.64%	46.04%	78.04%	100.00%
average	4.57%	16.53%	6.83%	23.54%	19.26%	49.21%	33.29%	97.94%

[1] For SGXFuzz, the coverage is gathered by the Ghidra script shipped with kAFL fuzzer.
For EnclaveFuzz, the coverage is collected by clang's source-based code coverage feature [50].

Conclusion 1: Greatly improve input validity (3x) and coverage (4x).

Vulnerability Detection

- **Insight:**
  - Memory corruption can be detected by instrumenting memory accesses (e.g., with AddressSanitizer), but the instrumentation must be adapted to the enclave memory layout (ELRANGE).

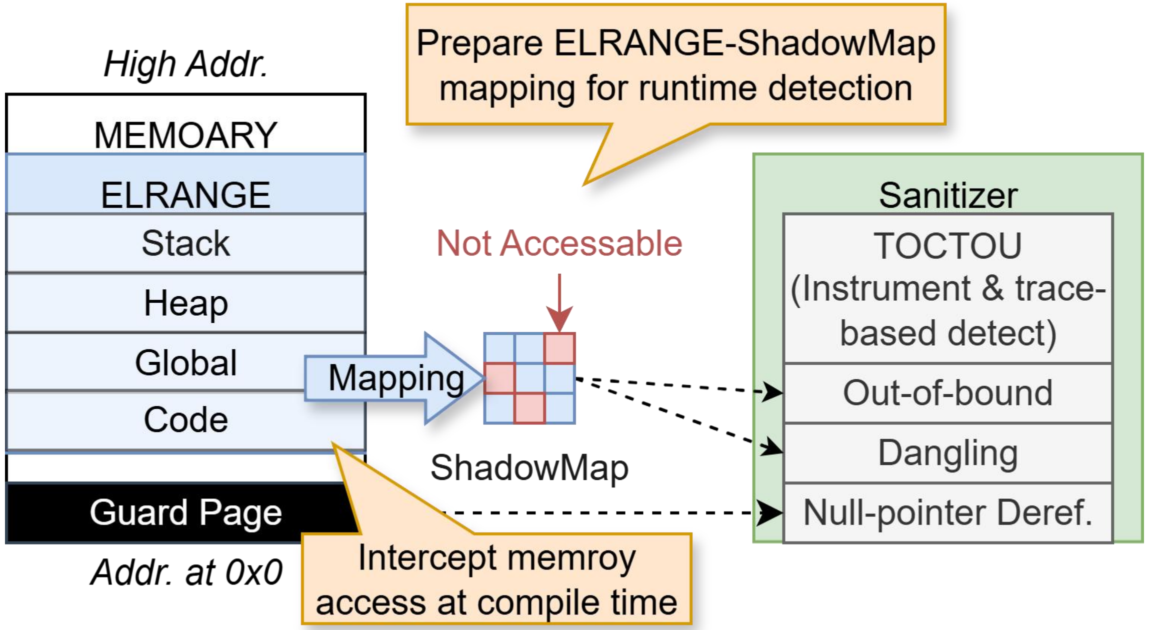


TABLE III: Vulnerabilities and Bugs found by EnclaveFuzz

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-runtime	4	
	BiORAM-SGX	1	
Use After Free	intel-sg		
	SGX_S		
	mbedtls		
TOCTOU	TaL		
	wasm-micr		
Stack Overflow	SGX_S		
	ehs		
	BiORAN		
	SGXCry		
Heap Overflow	sgx-w		
	TaL		
	sgxw		
	ehs		
	wasm-micr		
Int Overflow	TaL		
	sgx-c		
	plin		
Arbitrarily Read/Write/Execute	trusted-functio		
	wasm-micr		
Unchecked Size	trusted-function-framework	1	1
Total	14 Apps		162

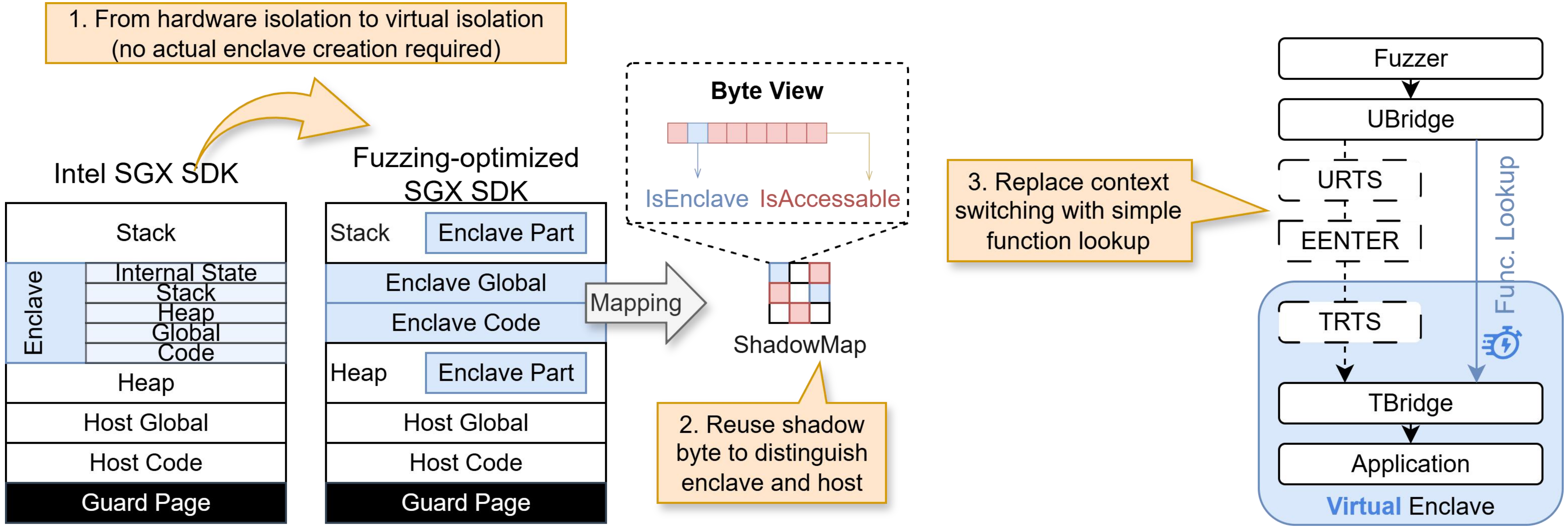
Conclusion 2: Total 162 vulnerabilities, mostly Null-Pointer Dereference & TOCTOU. SGX threat model is overlooked.

```
SQLITE_PRIVATE int sqlite3BtreeOpen(...) {
  unsigned char zDbHeader[100];
  rc = sqlite3PagerReadFileheader(...,zDbHeader);
//unixRead is called, zDbHeader is passed to pBuf
}
static int unixRead(..., void *pBuf, ...) {
  got = seekAndRead(...);
  memset(&((char*)pBuf)[got],0,amt-got);//overflow
}
static int seekAndRead(...) {
  // OCALL to get got from host
  got = osRead(id->h, pBuf, cnt);
  return got;
}
```

Listing 3: Stack Overflow in SGX_SQLite

Fuzzing Speed

- **Insight:**
  - The enclave creation and context switching are very time-consuming. Aside from TBridge, the remaining can be optimized away for fuzzing.



Fuzzing Speed

- **Insight:**
  - The enclave creation and context switching are very time-consuming. Aside from TBridge, the remaining can be optimized away for fuzzing.

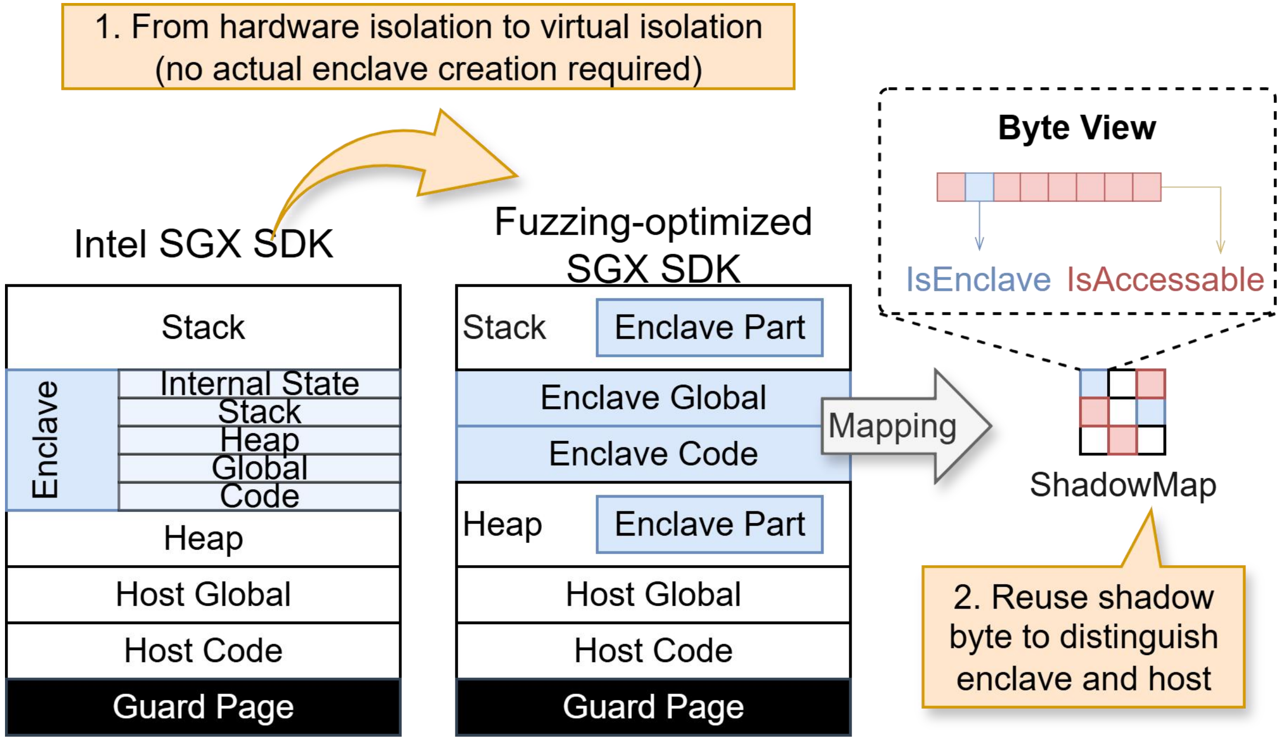


TABLE VI: Ablation Study: fuzzing-optimized SDK

Enclave Name	EnclaveFuzz-SIM	EnclaveFuzz-HW	EnclaveFuzz (Opt.SDK)
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×

TABLE VII: Optimized SDK Performance

Item	Opt.SDK	Sim w.	HW w.	Sim.	HW.
1K Create	0.14s	1.89s	28.27s	1.50s	25.63s

Conclusion 3: Much faster (6.91x) than Intel SGX SDK's simulation mode (2.67x) and hardware mode (1x, baseline). Enclave creation is 200× faster than in hardware mode.

TEE Security = Architecture × Semantics × Implementation

EPILOGUE

TEEs are designed to enhance application security, but improper use can instead undermine it, leading to privacy breaches, financial losses, regulatory violations, and ultimately, a loss of user trust.

Thanks!

Speaker: Liheng Chen

Supervisor: Chao Zhang

Email: 791960492@qq.com / leonechen9@gmail.com

Home Page: https://leonechen.github.io