ESem: To Harden Process Synchronization for Servers

Zhanbo Wang[1,2], Jiaxin Zhan[1,3], Xuhua Ding[4], Fengwei Zhang[3,1,*], Ning Hu[2]

[1] *Research Institute of Trustworthy Autonomous Systems, Southern University of Science and Technology, China*

[2] *Peng Cheng Laboratory, China*

[3] *Department of Computer Science and Engineering, Southern University of Science and Technology, China*

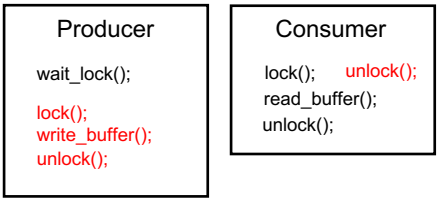
[4] *Singapore Management University, Singapore*

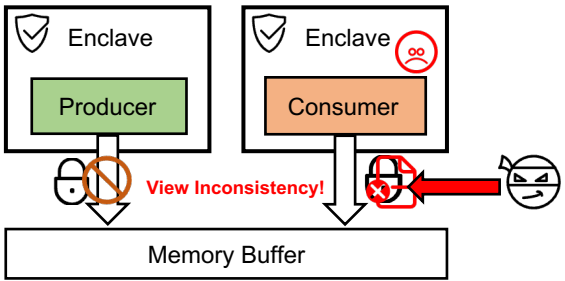
Motivation

- For the server, there is no trusted process synchronization mechanism (even TEE and TEE LibOS do not have it)
- This will lead to a situation where process synchronization is completely dependent on the legacy implementation provided by the kernel
- However, the legacy process synchronization may have security issues of view consistency because it is not protected

Motivation

- Attacks that break view consistency
  - The producer and consumer in the enclave have mutually exclusive access to the buffer
  - An attacker may release the lock before the consumer gets the data
  - The view seen by the consumer does not match the actual view, which may cause execution flow errors inside the enclave





✓ The lock ( ) is a process synchronization lock, not encryption

Motivation

- The problem of process synchronization has also been widely studied
  - e.g., TCLocks (OSDI'23), SynCord (OSDI'22), Trātr (Security'22), CLoF (SOSP '21), LockDoc (EuroSys '19), BRAVO (ATC'19), wPerf (OSDI '18), SyncPerf (EuroSys '17), SyncProf (ISSTA'16) ...

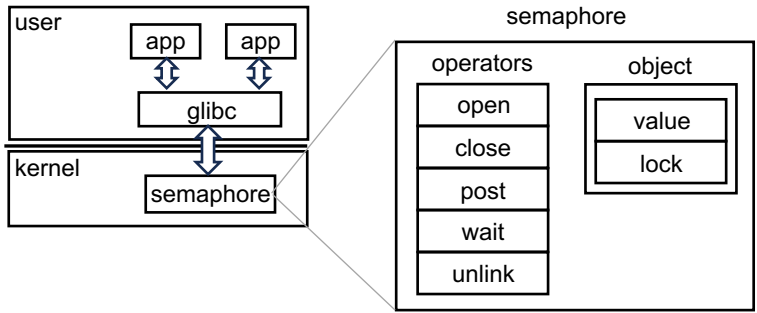
- In summary
  - The security of process synchronisation is fragile, important and worth studying

ESem -- Attack Model

- An attacker can manipulate synchronization objects to disrupt processes and induce errors in victim applications
- The secure communication channel between the application and ESem cannot be intercepted
- Denial-of-service attacks and protection of application code and data are out of scope

ESem -- Overview

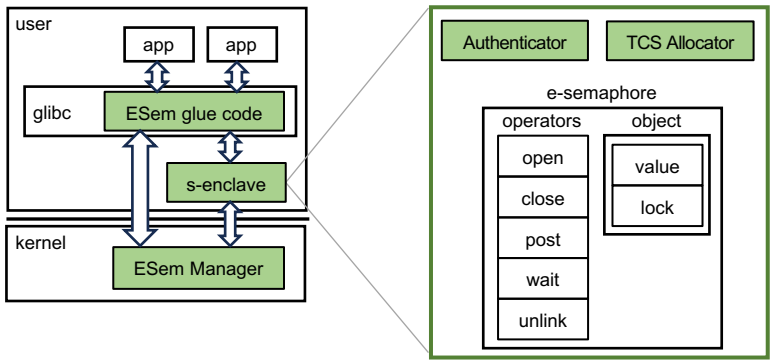
- How to protect process synchronization?
  - We propose a mechanism to use Intel SGX to harden process synchronization
  - We choose semaphore as the prototype because it is semantically rich and frequently used



✓ The application uses the legacy semaphore in the kernel by calling the glibc interface

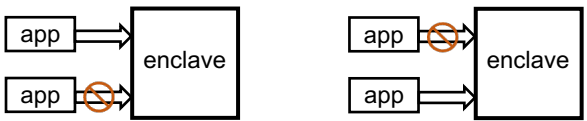
ESem -- Overview

- In the ESem architecture, includes s-enclave, ESem glue code and ESem Manager. The s-enclave stores semaphore object and operations. It also contains Authenticator and TCS Allocator



ESem -- Overview

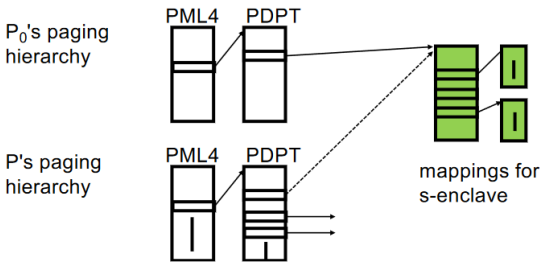
- Main challenge
  - ➢ Securing process synchronization stems from the conflict between intro-process isolation and inter-process sharing. No existing hardware-assisted isolation technique has the built-in support for both needs

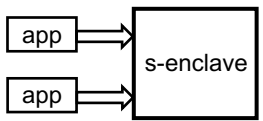


✓ Enclave only serves one process at a time

ESem -- Enclave Management

- To address that main challenge, we propose the enclave roaming mechanism
  - ➢ The ESem manager copies the PDPT page entry for the s-enclave to the PDPT page for the process. All processes accessing the e-semaphore share the same enclave mapping. When the process closes its e-semaphore, the ESem manager removes the mapping from its PDPT page entry





✓ s-enclave can serve multiple processes at a time through enclave-roaming

ESem -- Access Control

- Since the s-enclave is shared across multiple processes, a rogue process may invoke the enclave to operate semaphores not allocated to it

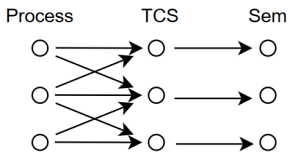
- ESem relies on the Authenticator inside the s-enclave to check the request. It maintains a semaphore metadata table to ensure that only authenticated processes can access the e-semaphore
  - ➢ During e-semaphore creation, if the owner process has an application enclave, it can share the key securely with the s-enclave. And by modifying the semaphore metadata table, the access rights of the e-semaphore are granted to the process. Otherwise, it is authenticated only by pid

Semaphore	PID	Key	TCS
x	30000	0xFE...23	...
y	30001	-	...

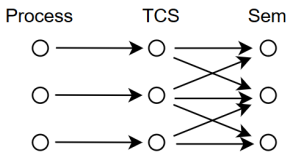
ESem -- Thread Management

Since context switching is required for external processes to enter the enclave, a mechanism is needed to manage the correspondence between external processes and internal threads. The Authenticator also checks the TCS that the process will use

- Thread-Semaphore Binding
  - Lockless Access
- Thread-Process Binding
  - Atomic Exchange with Spin Lock
  - Supports SGX Switchless Mode



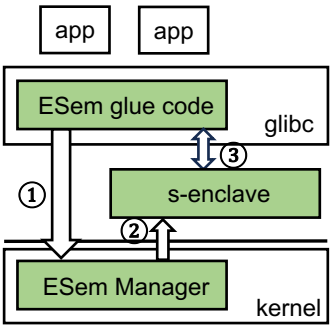
TCS-Semaphore binding



TCS-Process binding

ESem -- Workflow

- The application calls the ESem Manager in the kernel by calling the interface in glibc
- The ESem Manager is responsible for initializing the s-enclave and mapping the s-enclave to the process page table so that the s-semphore can be opened
- After the application opens the s-semphore, it can call the P/V operation in the s-enclave through interfaces in glibc



Implementation

- Platform
  - Intel i7-9700k CPU
  - 16 Gigabytes RAM
  - Linux Ubuntu 22.04
  - Intel SGX SDK 2.18
  - GLibC 2.36

- ESem Component Codebase Sizes

Module	Line of Code
Enclave	1350
Lib-C	452
Kernel Module	1020

Performance Evaluation

- Micro Evaluation
  - Most operations complete in 15 microseconds

Operation	Esem (µs)	Legacy (µs)	Difference (µs)
Open	17.02	9.87	7.15
Close	12.45	10.73	1.72
Post	7.02	5.3	1.72
Wait	7.39	6.2	1.19
Post - lockless	6.78	0.014	6.77
Wait - lockless	6.23	0.012	6.22
Unlink	10.05	9.03	1.02

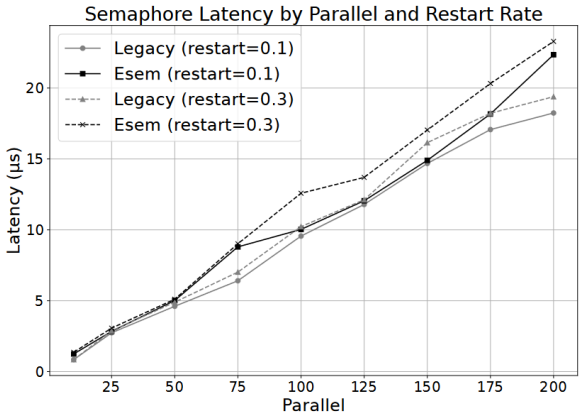
Performance Evaluation

- Macro Evaluation
  - PTS-NG Semaphore Benchmark

Metric	Legacy	Esem	Esem - Switchless
PTS – NG (Op/s)	11,488,876	9,533,037	9,125,001
Difference (Op/s, %)	- -	-1,955,839 -17.02%	-2,363,875 -20.58%

Performance Evaluation

- Macro Evaluation
  - LMBench Semaphore Benchmark



✓ ESem does not result in a substantial slowdown in the entire workflow

Performance Evaluation

- Real-World Application Workload
  - All relative differences are below 3% compared to legacy

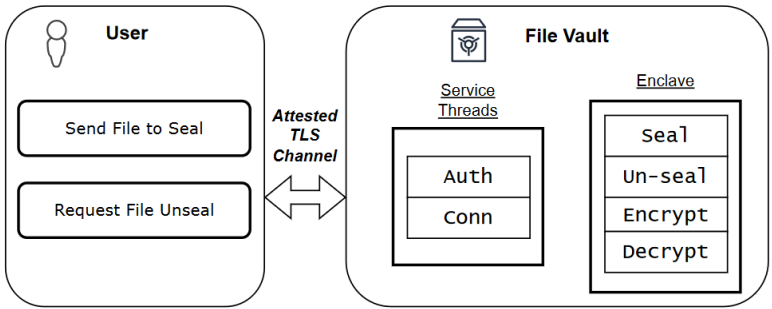
Application	Legacy	Esem	Difference
PostgreSQL (tps)	1,088.49	1,112.79	+24.30 (+2.23%)
Redis (rps)	62,847.97	61,097.82	-1,750.15 (-2.79%)
Apache (tpr)	9,208.09	9,427.83	+219.74 (+2.39%)

Application of Esem -- File Vault

- Why File Vault?
  - Confidential file data is protected by SGX
  - Synchronization of concurrent operations is not protected

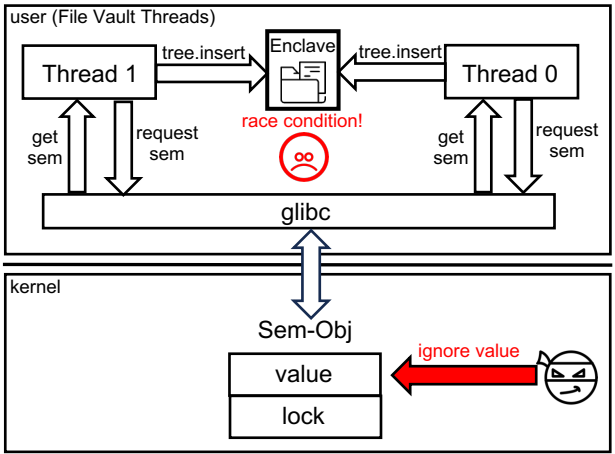
Application of Esem -- File Vault

- Composition
  - Service Threads: The File Vault application requires user authentication in the Service Thread. Once started, the application will establish a TLS connection with the user
  - File Vault Enclave: Encrypt and decrypt user files. The user needs to pass the sealed key along with the file to the SGX enclave



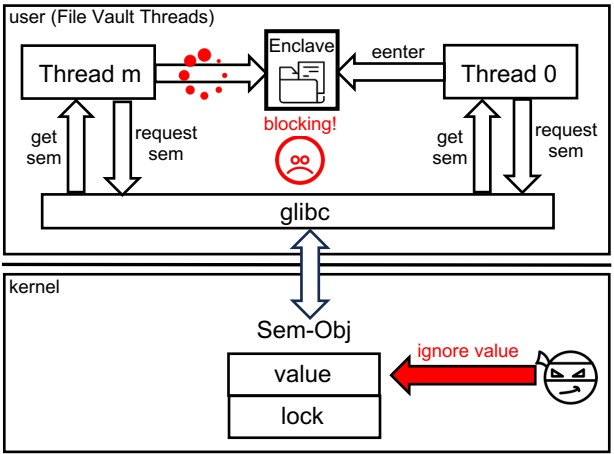
Application of Esem -- File Vault

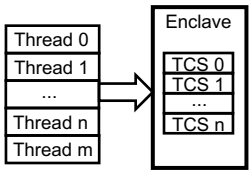
- Attacks on Synchronization
  - File tree race: The attack causes corruption of the file tree structure



Application of Esem -- File Vault

- Attacks on Synchronization
  - Service thread blocking: The attack causes the semaphore that manages service threads to appear perpetually busy, effectively hogging the system, leading to reduced performance

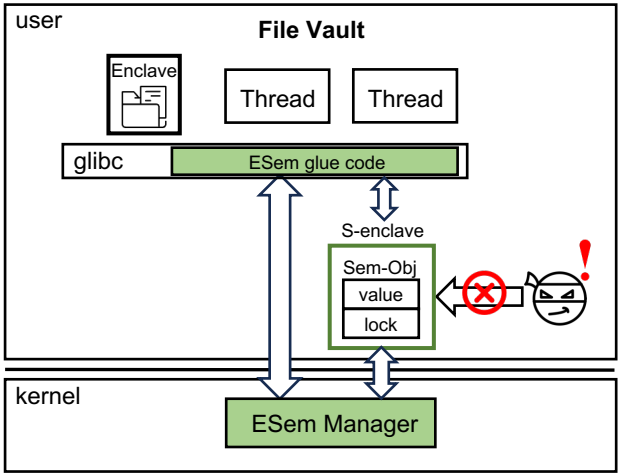




✓ Since the sem value is invalid, m external threads accessing n threads inside the enclave (m>>n) will cause blocking

Application of Esem -- File Vault

- ESem Hardened Version
  - Ensuring the consistency and reliability of services in the face of synchronization attacks



Conclusion

- ESem protects process synchronization from kernel privilege attacks through hardware-assisted isolation technology
  - Balanced security and performance
  - POSIX APIs compliant
  - Suitable for real-world applications
- Future Work
  - Exploring authorized synchronization mechanisms, protection of shared resources within enclaves, and comparison with other TEE-based solutions to further enhance ESem security and applicability

Thanks for listening!

Q & A

zhanjx@mail.sustech.edu.cn