KU LEUVEN

DEPARTMENT OF COMPUTER SCIENCE

Unlimited Lives: Secure In-Process Rollback with Isolated Domains

Merve Turhan[1 2], Thomas Nyman[1], Christoph Baumann[1], Jan Tobias Muehlberg[2]
[1]Ericsson, [2]KU Leuven, Belgium
✉ merve.turhan@kuleuven.be

https://arxiv.org/pdf/2205.03205.pdf

How to improve the resilience of long-running software against run-time attacks?

We present secure in-process rollback, an approach for recovering vulnerable applications after an attack is detected

Motivation:

- Run-time attacks corrupt memory, alter behavior, or crash the application
- One malicious request can deny service to all clients
- Volatile application state is lost if the application crashes

Challenges:

- Conventional error handling not sufficient as attacks can alter behavior
- Data in memory may already be corrupt when an attack is detected
- Solution must be efficient, robust and be adaptable to new attacks

Our contribution:

- **Rollback mechanism that recovers the execution state of the program to a state in which allocated memory is free from corruption**

Requirements:

- We compartmentalize the application into distinct domains
- A memory defect within a domain must only affect that domain's memory, not the memory of other domains
- We leverage hardware-assisted software fault isolation (SFI) based on Protection Keys for Userspace in COTS processors (Intel, AMD)

Implementation: Library for Secure Domain RollBack (SDRoB)

Allows developer to create isolated domains within application

Analogy:

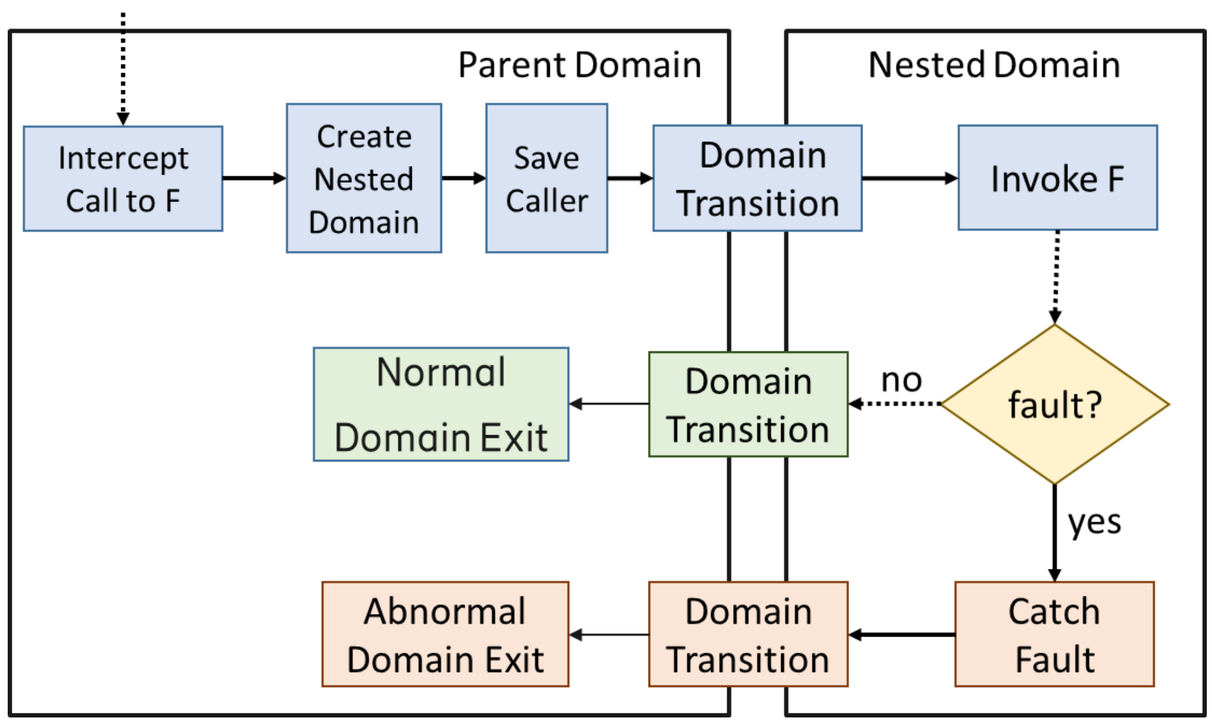
- Software exceptions that capture memory errors within isolated domain

What can be isolated within an application?

- Subroutine in main application, e.g., C function that processes unsanitized input
- Libraries with possible memory vulnerabilities
- Library that handles sensitive information, e.g., cryptographic keys in OpenSSL

What errors can be detected?

- Stack overflows, heap overflow, domain violations
- Leverage different pre-existing detection mechanisms e.g., stack canaries, CFI, ASLR and more

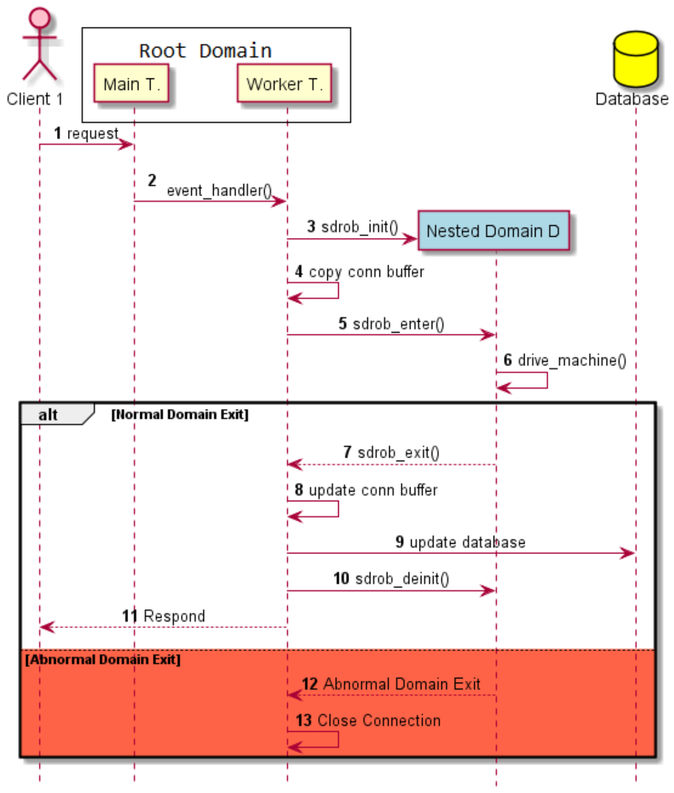


Example Case Study: Memcached

- Memcached is a multithreaded memory object caching system with no persistent storage.
- A malicious request that crashes any thread causes cache database to be lost
- SDRoB can make Memcached resilient against malicious requests and recover without affecting benign clients

Changes in Memcached:

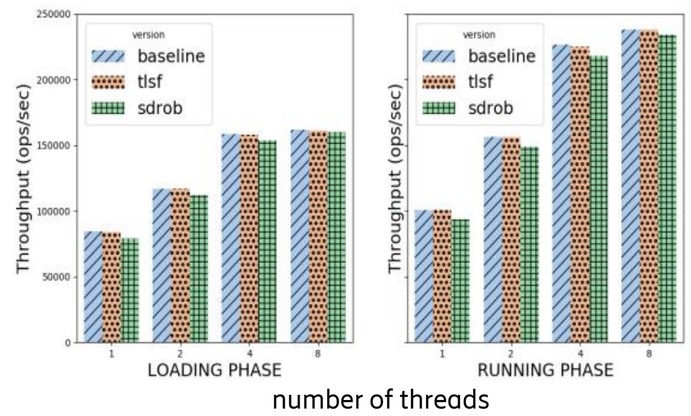
- Each client event is run in separate nested domain
- If any event corrupted, connection is closed, the nested domain rollback to prior good state



Rollback performance:

Abnormal Domain Exits Latency	Memcached Container Restart Time
3.46 µs ±0.9µs	400000µs ± 19000µs

Performance impact of the isolation mechanism:



Reasonable performance overhead : 3.0% – 7.3%

KU Leuven
Department of Computer Science
Celestijnenlaan 200A box 2402
3001 Leuven, Belgium

5G HOSTS DistriNet ERICSSON

The character illustration © Semanur Gulmez