Exploring the Environmental Benefits of In-Process Isolation for Software Resilience

Merve Gülmez & Thomas Nyman & Christoph Baumann & Jan Tobias Mühlberg

Motivation

The urgent need for climate action requires a focus on sustainability in every field!

How can systems security be environmentally friendly?

Example: resiliency in cloud environments

Cloud elasticity and scalability makes it easy for developers to provision compute resource

→ Building resiliency is now achieved through redundancy, i.e., *over-provisioning* of compute resources

Over-provisioning is detrimental to the environment!

Can we limit need for redundancy and still build resilient systems?

→ Thus, contributing to environmental sustainability and sustainable security

Sustainable solutions against memory attacks

Memory-related errors in C and C++ are still one of the primary root causes of software vulnerabilities

Solution 1: Retrofit unsafe software with run-time defense techniques, e.g., such as stack canaries, control-flow integrity

✓ can detect attacks, ✘ but not recover application

IEEE DSN 2023

Secure Rewind and Discard with Isolated Domains

Solution 2: Perform new development in memory-safe languages, e.g, Rust

✓ more difficult to exploit by design

✘ must still interact with memory-unsafe code

✘ not complete guarantees of resilience against memory attacks

Ongoing-Work

Friend or Foe? Foreign Function Interface in Rust

Building in resiliency can contribute to sustainable security,
How we build in resiliency can contribute to environmental sustainability

Secure Rewind and Discard with Isolated Domains

We present secure rewind and discard, an approach for recovering vulnerable applications after an attack is detected

Requirements:

- We compartmentalize the application into distinct domains
  A memory defect within a domain must only affect that domain's memory

- Leveraging different pre-existing detection mechanisms
  such as stack canaries and domain violations

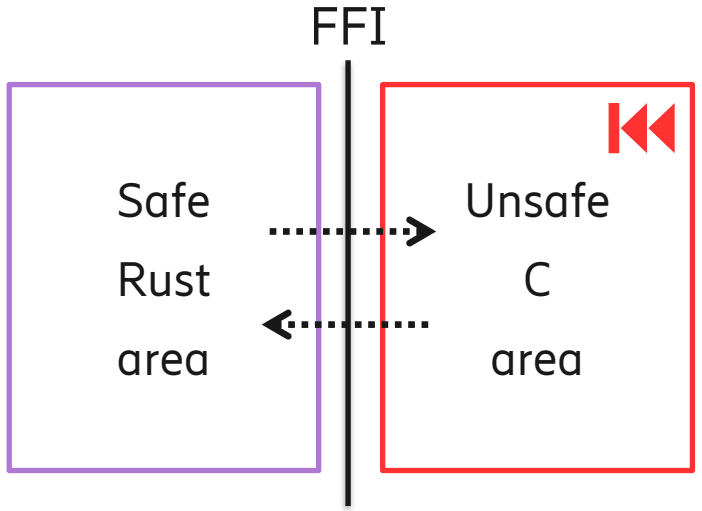
Secure Rewind and Discard with Isolated Domains for Foreign Function Interface

We implement secure rewind and discard for Rust application to maintain memory safety for unsafe Rust

Requirements:

- Providing heap and stack isolation between memory safe and unsafe area
- Rewinding capability in case of memory corruption of memory unsafe area

- Easy-Use-API to reduce the development effort!
  such as argument passing, return value



How to evaluate sustainability of software?

Classic evaluation metrics are necessary but not sufficient.

- Application confidentially, integrity, **availability**

To evaluate environmental sustainability also need to consider:

- Potential impact for implementation at massive scale,
  e.g., saving a few cycles in millions of software instances gives benefits of scale.

- Reposition security from a “cost” to providing additional benefit for system,
  e.g., turn detection capabilities into mechanism that allows system to recover early and efficiently.

Evaluating sustainability impact on protected system

Replication or diversification of software

- decrease the likelihood of memory-related attacks, increase software longevity
- over-provisioning hardware resources and is not environmentally friendly.

Our solution supports fast recovery time with only limited runtime overhead. SDRaD substantially reduces the time to recover from a fault.

- Memcached setup with a 10GB database,
  - a regular restart, takes about 2 minutes
    violate 99.999% availability if there were 3 faults per year
  - in-process rewinding takes only 3.5µs
    allowing for more than $9x10^7$ recoveries per year

In summary

- Secure Rewind and Discard with Isolated Domains
  Rewind & Discard: Improving Software Resilience using Isolated Domains (IEEE DSN 2023)

- Secure Rewind and Discard with Isolated Domains for FFI
  Friend or Foe Inside? Exploring In-Process Isolation to Maintain Memory Safety for Unsafe Rust (pre-print available)

- **Exploring the Environmental Benefits of In-Process Isolation for Software Resilience**

https://secure-rewind-and-discard.github.io/

Thank you

https://secure-rewind-and-discard.github.io/