CheckCBox: Automated and Zero Cost Spatial Memory Safety
Arun Kumar, Aravind Machiry
Purdue University

The Never Ending Trend of Spatial Safety Violations

Spatial Safety Violations still are the Major class of vulnerabilities in Low-level system software.

Safe languages
Safe by design: Prevents memory corruption vulnerabilities.

Retrofitting Techniques
Address Sanitizer (ASan)
ASAN and SoftBound CETS
High Performance Overhead
No Backward Compatibility and needs runtime changes

What about Legacy code? Not feasible to rewrite.

Checked C
Fast
Backward compatible

Pointers annotated with Checked C types are guaranteed to not have any spatial violations

Can We Automatically Convert C to Checked C?

3C
Complete Automated Conversion is not Feasible
Some regions of code will be still unchecked

CheckCBox

CheckCBox: High level Idea

Challenges
- Automatically generating marshalling layers - Interaction between checked/unchecked/tainted types
- Handling Callbacks from “unchecked” region to “checked” region

Progress
- We were able to successfully encapsulate unchecked regions using RLBox and create required marshalling stubs
- Working on formalizing Checked C semantics with RLBox
- [On going] Working on automated encapsulation of unchecked regions into RLBox

Are you Curious?
- Open Source: https://github.com/purs3lab/CheckC-Box
- Contact:
  - Arun Kumar (bhattacharjee@purdue.edu)
  - PurS3 Lab: https://purs3lab.github.io/