CT Advanced Computing Center (CACC) Security Seminar Series 2024

Speaker: Triet Huynh
Date: November 20, 2024
Time: 11:15 12:30pm
Location: ITE 336
Meeting Link : https://uconn-cmr.webex.com/uconn-cmr/j.php?MTID=m4d36ddf4d36edd727aa04331cdb7d7f6
Meeting number (access code): 2633 368 4613
Meeting password: MqmJMizs533

TRusT: A Compilation Framework for In-process Isolation to Protect Safe Rust against Untrusted Code

Rust was invented to help developers build highly safe systems. It comes with a variety of programming constructs that put emphasis on safety and control of memory layout. Rust enforces strict discipline about a type system and ownership model to enable compile-time checks of all spatial and temporal safety errors. Despite this advantage in security, the restrictions imposed by Rust's type system make it difficult or inefficient to express certain designs or computations. To ease or simplify their programming, developers thus often include untrusted code from unsafe Rust or external libraries written in other languages. Sadly, the programming practices embracing such untrusted code for flexibility or efficiency subvert the strong safety guarantees by safe Rust. This paper presents TRUST, a compilation framework which against untrusted code present in the program, provides trustworthy protection of safe Rust via in-process isolation. Its main strategy is allocating objects in an isolated memory region that is accessible to safe Rust but restricted from being written by the untrusted. To enforce this, TRUST employs software fault isolation and x86 protection keys. It can be applied directly to any Rust code without requiring manual changes. Our experiments reveal that TRUST is effective and efficient, incurring runtime overhead of only 7.55% and memory overhead of 13.30% on average when running 11 widely used crates in Rust.