STEVENS
INSTITUTE OF TECHNOLOGY
1870

Ph.D. DISSERTATION DEFENSE

Candidate:	Yuchen Zhang
Degree:	Doctor of Philosophy
School/Department:	Charles V. Schaefer, Jr. School of Engineering and Science / Computer Science
Date:	Monday, April 17th, 2023
Time/Location:	11:00 a.m. / https://stevens.zoom.us/j/5868723006
Title:	Balancing the Security and Performance of Modern Programming Languages
Chairperson:	Dr. Georgios Portokalidis, Department of Computer Science, Stevens Institute of Technology
Committee Members:	Dr. Jun Xu, School of Computing, The University of Utah Dr. Shucheng Yu, Department of Electrical and Computer Engineering, Stevens Institute of Technology Dr. Eric Koskinen, Department of Computer Science, Stevens Institute of Technology

ABSTRACT

Ensuring the balance between security and performance is a critical aspect of modern programming languages. While security checks are essential in safeguarding systems from threats and vulnerabilities, performance considerations are crucial for achieving optimal program execution speed and efficiency. It is worth noting that many programming languages (i.e., C programming) prioritize efficiency over security, often because adopting protections incurs performance overhead that exceeds practical acceptance.

C and C++ have gained widespread usage in system programming due to their ability to low-level hardware control and high efficiency. However, these languages lack safety features, leading to security vulnerabilities. To address this issue, researchers have proposed adding checks like AddressSanitizer (ASan) to C/C++ codes. Nevertheless, the deployment of ASan can incur significant run-time overhead. My dissertation research tackles this issue through the development of ASan--, a tool comprising a series of optimizations to reduce sanitizer checks, thereby enhancing the efficiency of ASan. This tool also retains the good properties of ASan, including its capability, scalability, and usability.

Another direction to address the trade-off between safety and efficiency is through the redesign of programming languages. Rust has emerged as a viable alternative to C/C++ due to its capability of improving security by offloading several run-time validation checks to the compilation process. However, the performance overhead of the residual checks in Rust has not been extensively studied, and it is unclear how it compares to C programming. My research addresses this gap by conducting an empirical study on Rust run-time checks and comparing the performance with C. Our results reveal that Rust can provide enhanced security without significantly sacrificing performance to the same extent as C programming.

Despite the inherent features of safety and performance, Rust's safety mechanisms are not infallible. It comes with strict rules that forbid low-level controls (e.g., dereferencing raw pointers). For flexibility, Rust allows circumventing those rules through unsafe codes, which can undermine the language's safety guarantees. My research addresses this issue by conducting another study on Rust unsafe codes, evaluating the necessity of different scenarios of unsafe code usage, and developing a tool that can detect unnecessary unsafe code, suggest safe alternatives, and automatically convert the unsafe code to its safe equivalent to further enhance the security of the Rust software ecosystem.