Chair of Decentralized Systems Engineering
Department of Informatics

LLVM-assisted Program Analysis for High-performance Security Enforcement

General information	**Advisor**	Dr. Myoung Jin Nam
	Email	Myoungjin.nam “at” tum email address
	Date	25.01.2022
Type	Master / Bachelor / Guided Research	
Description	Before developing security defenses against known attacks, it is essential to ensure integrity and safety of systems by detecting and preventing errors in the first place during the development stage and practical use. Unfortunately prevention of system vulnerabilities still suffers from heavy overhead (dynamic analysis) and false positives (static analysis). One of the major contributors to the run-time overhead (slowdown) is an increase in executed instructions to check errors at runtime, along with cache misses, increased memory bandwidth, etc. For example, run-time checks are required basically at every memory access to detect buffer overflow – one of the software vulnerabilities that attackers exploit most frequently. Performing extra operations to check and retrieving supplementary metadata significantly slows down a program. Resolving overhead is still a key challenge in the memory safety area despite a large amount of research for decades.	

Chair of Decentralized Systems Engineering
Department of Informatics

	This research aims at achieving both high performance and strong security enforcement i.e. preventing memory safety violations (buffer overflows, use-after-free, double free, etc) at fine granularity (byte or object-level granularity, rather than page or file-level) while minimising run-time overhead. One way to reduce performance degradation is static analysis, more specifically for this research, customised compiler optimisation. Majority of software-based memory safety solutions are based on instrumentation - compiler-assisted instrumentation or binary rewriting. The research is focused on LLVM-assisted program analysis and transformation to remove run-time checks as many as possible; to reduce run-time overhead; and to get close to lightweight and strong system protection.
Keywords	Security, Integrity, Memory Safety, LLVM, Compiler Optimisation, Static Analysis, Dynamic Analysis, High Performance, Buffer Overflows
Goals	**Concrete outcomes** 1. Implement compiler optimiser using LLVM 2. Evaluate and demonstrate performance improvement in run-time verification tools. **Bonus points** 3. Profiling and finding performance hot spots 4. Extension to detect other kinds of security vulnerabilities
Prerequisites	**Compulsory** ● Familiar with C and C++ ● Taken courses on Compiler covering LLVM **Preferred** ● Knowledge of static analysis and compiler-assisted dynamic analysis ● Familiar with debuggers (GDB, Valgrind, lldb ..) ● Familiar with performance analyzing tools (Perf or PCM)
References	1. Dynamic Analysis vs. Static Analysis 2. Memory safety - Wikipedia 3. SoK: Eternal War in Memory 4. FRAMER: A Tagged-Pointer Capability System with Memory Safety Applications

Application process

Please send an email to the advisor including the following:

- Email subject: “Thesis application (DSE)”
- CV
- A copy of your transcript(s)
- A **motivation statement**, please include samples of your work that you are proud of (e.g., major projects, open-source contributions, Github page, etc.) and/or writing samples (e.g., your technical blog, project reports, etc.)