UNIVERSITATEA
DIN
CRAIOVA

Open4Tech Guest Lecture

Memory Safety:
Static vs Dynamic Analysis

Victor Ciura
Principal Engineer, CAPHYON
www.caphyon.com

Duration: **120 min**
Difficulty: **intermediate**
Format: **interactive** (lecture + examples to be analyzed together with the students)
Media: **slides**, **live code demos**
Prerequisites: **C++, knowledge about the program stack & heap, memory allocators**

Keywords

Clang, llvm, asan, msvc, visual studio, dynamic analysis, address sanitizer, memory, CVE, Azure, fuzzing

Abstract

Clang-tidy is the go-to assistant for most C/C++ programmers looking to improve their code, whether to modernize it or to find hidden bugs with its built-in checks. Static analysis is great, but you also get tons of false positives.

Dynamic/runtime analysis, on the other hand, can catch more classes of memory vulnerabilities, but comes with its own costs.

Let's see how AddressSanitizer works behind the scenes (compiler and ASAN runtime) and analyze the instrumentation impact, both in perf and memory footprint. We'll examine a handful of examples diagnosed by ASAN and see how easy it is to read memory snapshots to pinpoint the failure.

Outline

- Intro
- Static analysis vs dynamic analysis
- Test Units as they relate to dynamic analysis and fuzzing
- Fuzzing techniques and tools
- Common CVEs
- What are sanitizers? (ecosystem, types of sanitizers, what can they diagnose)
- Memory safety
- Address Sanitizer
- Types of memory safety issues ASAN can detect (examples)
- Instrumentation perf impact
- Instrumentation memory footprint
- ASan: behind the scenes
- ASan in Visual Studio 2019
- VS projects: setup, workflows, runtime linkage (EXEs, DLLs)
- static vs dynamic link (ASan runtime)
- ASan Snapshots in IDE: call-stack, memory window, ASan breakpoint
- ASan + Fuzzing in Azure cloud
- Azure MSRD service: fuzzing jobs
- Takeaway
- Closing thoughts

Links

https://www.youtube.com/watch?v=yJLyANPHNaA
https://clang.llvm.org/extra/clang-tidy/checks/list.html
www.clangpowertools.com
https://docs.microsoft.com/en-us/cpp/code-quality/code-analysis-for-cpp-corecheck
https://aka.ms/cpp/cfg-llvm
https://www.youtube.com/watch?v=0EsqxGgYOQU
https://github.com/google/sanitizers
https://github.com/google/sanitizers/wiki/AddressSanitizer
https://github.com/llvm/llvm-project/tree/master/compiler-rt
https://devblogs.microsoft.com/cppblog/addresssanitizer-asan-for-windows-with-msvc/
https://devblogs.microsoft.com/cppblog/asan-for-windows-x64-and-debug-build-support/
https://www.microsoft.com/security/blog/2020/09/15/microsoft-onefuzz-framework-open-source-developer-tool-fix-bugs/