Statische & dynamische Code Analyse

Dennis Lühring aus Pfinztal, selbstständiger Softwareentwickler C/C++(…)

dl.soluz@gmx.net

Ein (ultra) schneller Tool Überblick

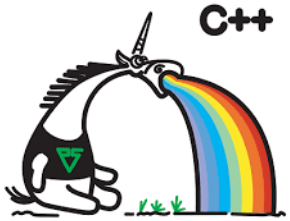
Statische Analyse

- Es gibt verschiedene freie und kommerzielle Tools
- Der Quelltext wird nach logischen Fehlern und schlechten Praktiken durchsucht – Memoryleaks, Out-of-Bounds, Null-Ptr-Access
- Lokale oder globale Suche – bei den Tools hier eher lokal
- Die Großen wie Coverity, KlocWork, CodeSonar, sonarsource machen globale Suchen kosten aber auch erheblich mehr
- Es kann Falschmeldungen geben
- Erstkontakt kann zu Warnungs-Explosion führen
- Es gibt nicht das EINE Wundertool
- (PC)Lint ist böse aber die Welt ist besser geworden

cppcheck

http://cppcheck.sourceforge.net/

- Von Daniel Marjamäki (und anderen)
- Linux, Windows
- Frei/Open Source
- Eigener Parser / „einfache“ Mustererkennung
- Arbeitet nur auf Quelltext (hat Probleme mit Syntaxfehler usw. - lieber sauber kompiliert)
- Tests: z.B. falsche STL-Verwendung, hpp/cpp Interface Unterschiede, Puffer-Über/Unterläufe, Memoryleaks, ...
- GUI oder auf Kommandozeile
- XML/CSV-Reports
- Sehr einfacher Kontakt z.B. im IRC-Chat
- Plugins für viele IDEs oder Integrations-Server wie Jenkins
- Meldungen: Fehler, Warnung, Style, Portierungs- und Performanzprobleme



PVS-Studio

https://www.viva64.com/en/pvs-studio/

- Hauptprodukt von Viva64 (russische Firma)
- Windows, (Linux)
- Kommerziell, Trial, Freie-Lizenz mit Code-Kommentar bei Open Source
- Basiert auf Clang und hat auch dessen Diagnosen im Bauch
- Projekt muss kompilierbar sein
- Tests: viel mehr als cppcheck, x64 Probleme erkennen
- Wird in großen Projekten verwendet z.B. Unreal Engine
- Integration in VStudio und frei stehend, QtCreator/CLion
- Meldungen: 3 Stufen - Fehler, Warnungen, Info
- Preis: 1-2k EUR

VStudio /analyze

https://msdn.microsoft.com/de-de/library/ms182025.aspx

- Windows (Linux-Builds?)
- Kommerziell oder frei in den Community Editions
- Fest in VStudio integriert, CL-Kommandozeile
- Viele Test und in Qualität zwischen PVS und cppcheck
- Integration in Microsoft Entwicklungs-Welt

Clang-Scan/Tidy

http://clang.llvm.org/extra/clang-tidy/
https://clang-analyzer.llvm.org/

- Linux (, Windows)
- Frei/Open Source
- Starke Kontrollfluss-Analyse (kleine Bruder von Coverity)
- z.B. „beim x. Durchlauf kommt hier ein nullptr raus“
- Sehr viele Tests
- z.B. „C++ Core Guidelines“ und Boost-Usage-Checks
- Leicht erweiterbares Framework für eigene Test
- Frontends für QtCreator, CLion, ...

Dynamische Analyse

- Manche Fehlerarten lassen sich schwer mit statischer Analyse finden z.B. Thread/Memory Probleme
- Es ist einfacher den Fehler beim Auftreten zu erkennen
- Die Tools arbeiten mit „Dynamic Binary Instrumentation“ oder mit Instrumentierung schon zur Kompilezeit
- Um die Fehler zu erkennen braucht man Tests
- Dynamische Analysetools haben (hatten) einen schlechten Ruf
- Es hat sich viel in den letzten Jahren geändert

Valgrind

http://valgrind.org/

- Linux, Android, … kein Windows Port
- Bisher unter Linux das Primärtool
- Erkennt Speicher und Thread-Probleme
- Arbeitet auf fertigen Binaries
- Dynamic Binary Instrumentation
- Sehr langsam (20-40x langsamer)
- Falschmeldungen können vorkommen
- Für CLion gibts ein Plugin

Intel Inspector

https://software.intel.com/en-us/intel-inspector-xe

- Linux/Windows
- Kommerziell, Trial
- Findet Memory und Thread Probleme
- Dynamic Binary Instrumentation oder auch zur Kompilezeit
- Integration ins VStudio
- Falschmeldungen können vorkommen
- Sehr langsam (20-160x langsamer)
- Preis: ~2-3k EUR

Sanitizers

https://github.com/google/sanitizers

- Wird vom Team um Konstantin (Kostya) Serebryany bei Google entwickelt
- Integrativer Bestandteil von GCC (ab 4.8) und Clang (ab 3.3)
- Instrumentiert durch den Kompiler
- Clang bekommt zuerst neue Features
- Nur für 64-Bit – Linux ist Primärplatform
- Falschmeldungen werden stark vermieden
- Wird in großen Projekten wie Chrome, Qt, Firefox verwendet
- Sehr schnell - verlangsamen das Programm nur um Faktor 2
- Definitiv ein Gamechanger

Sanitizer-Varianten

AdressSanitizer/LeakSanitizer

- Findet Speicherprobleme: Memoryleaks, Use-After-Free, Use-After-Dtor, Double-Free
- Anregungen und Ideen werden schnell umsetzt
- By Design sollte es keine Falschmeldungen geben - Programmende beim 1. Fehler
- Es gibt Blacklist für LeakSanitizer

MemorySanitizer

- Erkennt Nutzung von nicht initialisiertem Speicher
- Librarys müssen mit MSan kompiliert sein

ThreadSanitizer

- Findet Data Races

UndefinedBehaviorSanitizer

Fertig!

- Fragen?
- Andere Tools aus diesem Bereich/Preisklasse?
- Interesse an einem vollständigen Talk mit mehr Details und Themen? (Live-Finding, Fuzzing, Hardening,…)