EÖTVÖS LORÁND TUDOMÁNYEGYETEM
INFORMATIKAI KAR

SZAKDOLGOZAT TÉMABEJELENTŐ

Hallgató adatai:
Név: Bahrami Benedek Attila
Neptun kód: C66EPC

Képzési adatok:
Szak: programtervező informatikus, alapképzés (BA/BSc/BProf)
Tagozat : Nappali

Belső témavezetővel rendelkezem

Témavezető neve: *Richárd Szalay*
munkahelyének neve, tanszéke: ***ELTE-IK, Programozási nyelvek és Fordítóprogramok Tanszék***
munkahelyének címe: ***1117, Budapest, Pázmány Péter sétány 1/C.***
beosztás és iskolai végzettsége: ***PhD student; Computer science MSc***

A szakdolgozat címe: Enhancing pattern matching-based static analysis of C-family software projects with project-level knowledge

A szakdolgozat témája:
(A témavezetővel konzultálva adja meg 1/2 - 1 oldal terjedelemben szakdolgozat témájának leírását)

Static analysis is a method to analyse the source code of software projects without performing a real execution of the application.
It is widely used in industry to find bugs and code smells during development, to aid in the prevention of bad code that misbehaves in production.
Among various methods, the most important techniques are the ones that are based on pattern matching on a syntactic representation of the software project.
Unfortunately, for programming languages in the C family, such as C++, the concept of "separate translation" causes issues for static analysis.
As most static analysers are built upon compilers, and in C++, each compiler only sees the local information in the source file it is to compile or analyse (as opposed to a more project-level knowledge), crucial details might be hidden, which lowers the accuracy of the analysis.

Clang-Tidy is a static analysis rule collection that is built upon the LLVM Compiler Infrastructure's C-family compiler, Clang.
It performs pattern matching on Clang's "Abstract Syntax Tree" (AST) representation, and generating diagnostics based on which analysis modules -- called "checks" -- the user turns on.
Unfortunately, the current versions of Clang-Tidy checks only access what is visible to the compiler, which is a local information.
Several classes of security issues and bad coding patterns might be diagnosed if the implemented checks would be capable of creating per-compilation knowledge, and reusing the full knowledge about the project during diagnosis.

The work of the thesis is to enhance Clang-Tidy on the infrastructure level to support multi-pass analyses in a generic manner, by utilizing the ideas similar to that of MapReduce.
This is achieved by allowing individual checks to store check-specific data on a thread-safe location.
A subsequent execution of the analysis will be able to do the pattern matching fine-tuned with the data stored in the previous step also available.
To prove the usability of the solution, a new safety and security related check, currently not provided by Clang-Tidy, will be developed utilizing the new infrastructure created in this work.
In the end, the results of the thesis will allow the international community behind LLVM to develop and make available a wider potential of checks.

Budapest, 2021. 11. 23.