Seminar

Advanced Hands-on Training

Software Development Tools

TuCan-No: 20-00-0673-pr

Course Type: 4SWS / 6CPs

Workload: **~180hours**

Prof. Dr.-Ing. Mira Mezini

Process

- **By Friday, May 2nd, 9:00 AM :** Send an e-mail with your preferred topic (one only) and why you are the right person(s) for this topic to: naeumann@cs.tu-darmstadt.de
  To apply as a group, send **one** email with the names of all group members

  Subject: “[SDT] : Project Selection”
- **Tuesday, May 6th:** We inform you about your assigned topic via e-mail
  - schedule a meeting that week to discuss the topic and requirements in detail
- **Next:** Contact your supervisor to discuss details of your topic
- **During Hands-on Training:** Bi-weekly meetings with supervisor in an agile process
  - Discuss the current state and the next steps
- **Wednesday, August 6th:** Final submission of artifacts

GNN on IFDS Graphs

The IFDS framework is a foundational tool in static analysis that allows performing Taint Analysis over an “expoloded supergraph”, a graph representation of a program. Graph Neural Networks are neural networks that operate on graphs. Investigate the feasibility of applying GNNs on the IFDS graph representation.

Task: Build a research prototype of a GNN vulnerability classifier that operates on IFDS generated by a static analysis framework of your choice.

Suitable for: 2 - 4 people

Languages and Frameworks: Java, Python

Starting point:

- https://dl.acm.org/doi/fullHtml/10.1145/3575879.3575964

Contact: naeumann@cs.tu-darmstadt.de

LLM-Assisted Static Analysis

CodeQL is a static analysis framework that allows analyzing Codebases through “Queries”. CodeQL can be used to find vulnerabilities in code. We would like to explore the capability of LLMs in writing these queries, and evaluate the capability empirically.

Task: Create an LLM agent that writes CodeQL queries, and create an evaluation framework to count the precision and recall. Explore different approaches (zero-shot, multishot, feedback etc…).

Suitable for: 2 - 4 people

Languages and Frameworks: Python

Starting point:

- https://arxiv.org/html/2405.17238v2
- https://codeql.github.com/docs/codeql-overview/

Contact: naeumann@cs.tu-darmstadt.de

CrySL to CFG Compiler

CrySL is a domain-specific language that allows to specify usage patterns of APIs. It is primarily used to check if cryptographic libraries are used correctly or to generate code that uses them accordingly. To make it compatible with other tools and usable for future projects, we want to automatically translate CrySL rules to Lark EBNF grammars.

Task: Build a tool that parses CrySL rules and converts them to context-free grammars.

Suitable for: 1 - 2 people

Languages and Frameworks: <language of your choice>, Lark

Starting point:

- https://eclipse.dev/cognicrypt/documentation/crysl/
- https://github.com/CROSSINGTUD/Crypto-API-Rules
- https://lark-parser.readthedocs.io/en/stable/grammar.html

Contact: daniel.maninger@tu-darmstadt.de

```
93  SPEC java.security.MessageDigest
94
95  OBJECTS
96   java.lang.String algorithm;
97   byte[] input;
98   int offset;
99   int length;
100  byte[] hash;
101  ...
102
103 EVENTS
104  g1: getInstance(algorithm);
105  g2: getInstance(algorithm, _);
106  Gets := g1 | g2;
107  ...
108  Updates := ...;
109
110  d1: output = digest();
111  d2: output = digest(input);
112  d3: digest(hash, offset, length);
113  Digests := d1 | d2 | d3;
114
115  r: reset();
116
117 ORDER
118  Gets, (d2 | (Updates+, Digests)), (r, (d2 | (Updates+, Digests)))*
119
120 CONSTRAINTS
121  algorithm in {"SHA-256", "SHA-384", "SHA-512"};
122
123 ENSURES
124  digested[hash, ...];
125  digested[hash, input];
```

Example CrySL rule for `javax.crypto.KeyGenerator`

SynCode with Dynamic Rules

SynCode is a *constrained decoding* framework for large language models (LLMs). It ensures that the generated text conforms to a given (context-free) grammar. For example, you can use it to generate 100% syntactically correct Python code. Currently, the grammar is fixed during generation. We would like to make SynCode more flexible, so we can modify rules in the grammar on the fly.

Task: Modify SynCode to support dynamically activating and deactivating rules during generation.

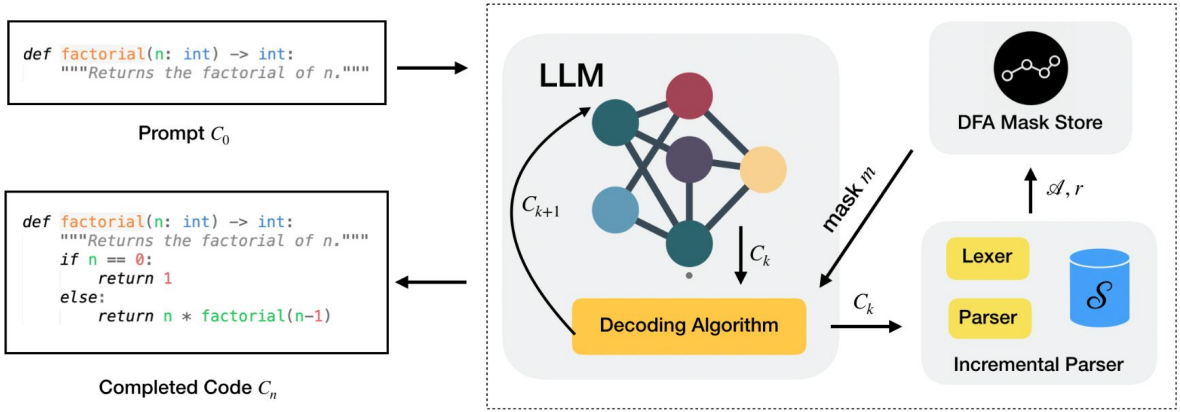
Suitable for: 2 - 4 people

Languages and Frameworks: Python

Starting point:

- https://arxiv.org/abs/2403.01632
- https://github.com/structuredllm/syncode

Contact: daniel.maninger@tu-darmstadt.de



The SynCode workflow

LLM-Based Developer Twin

Our goal is to create an AI teammate that reliably advances software projects with minimal human supervision. In this project, we develop Developer Twin, an intelligent tool designed to act as a software developer in GitHub-based engineering teams. Using large language models (LLMs), Developer Twin analyzes issues, implements the required code changes, ensures all tests pass, and commits updates to the repository.

Task: Build a research prototype of a Developer Twin that uses LLMs to analyze GitHub issues, implement code changes, pass tests, and commit updates to the repository.

Suitable for: 2 - 4 people

Languages and Frameworks: Python, LLM APIs

Starting point:

- https://github.com/features/copilot

Contact: amir.molzam@tu-darmstadt.de

Vulnerability Detection

The goal of this project is to create an open-source dataset of vulnerabilities (CWEs) in public GitHub repositories. It uses static analysis tools like CodeQL and dynamic analysis tools to identify security flaws. The vulnerabilities and corresponding code snippets are stored in a dataset.

Task: Build a research prototype that analyzes a public GitHub repository using a static analysis tool like CodeQL and a dynamic analysis tool to identify vulnerabilities, then stores the findings and relevant code snippets in an open-source dataset.

Suitable for: 2 - 4 people

Languages and Frameworks: Python, static analyzer, dynamic analyzer

Starting point:

- https://codeql.github.com/
- https://cwe.mitre.org/

Contact: amir.molzam@tu-darmstadt.de