

Hands-on Training

# Software Development Tools

## Summer Semester 22

TuCan-No: 20-00-0673-pr  
Course Type: 4SWS / 6 CPs  
Workload: ~**180hours**

Prof. Dr.-Ing. Mira Mezini

# Process

- **Today:** Send an e-mail with your three preferred topics and why you are the right person for these topics to: **leonid.glanz@tu-darmstadt.de**
- **Tomorrow:** Assignment of topics
- **Next:** Contact your supervisor to discuss details of your topic
- **During Practicum:** Bi-weekly meetings with supervisor in an agile process
  - Discuss the current state and the next steps
- **End of September:** Final submission of artifacts

# Secure/Parallel Code Execution

In app security reviewers analyze certain behaviors of the app and may execute dangerous code. In order to support them we need a secure environment for the execution.

- **Task:** Establish a secure JVM environment to execute such code.
- **Languages & Frameworks:**  **Scala**
- **Needed Experience:** Java, other things can be learned as you work
- **Suitable for:** 1 – 2 people
- **Contact:** [leonid.glanz@tu-darmstadt.de](mailto:leonid.glanz@tu-darmstadt.de)

# Resolving Useless Usages of Dynamic Loading

During a static analysis, dynamically loaded code cannot be analyzed and so important information could be missed.

- **Task:** Identify all useless usages of dynamic loading and replace it with static code
- **Languages & Frameworks:**  **Scala**  **OPAL**
- **Needed Experience:** A bit Scala, the rest can be learned as you work
- **Suitable for:** 1 – 2 people
- **Contact:** [leonid.glanz@tu-darmstadt.de](mailto:leonid.glanz@tu-darmstadt.de)

# Anti-Debugging Detection

Malware authors can use anti-debugging techniques to evade dynamic analysis, which then cannot detect malware hidden in this manner.

- **Task:** Identify anti-debugging code and replace it with direct code execution calls.
- **Languages & Frameworks:**  **Scala**  **OPAL**
- **Needed Experience:** A bit Scala, the rest can be learned as you work
- **Suitable for:** 1 – 2 people
- **Contact:** [leonid.glanz@tu-darmstadt.de](mailto:leonid.glanz@tu-darmstadt.de)



# Visualization of Security Information

In order to get an overview of an app many small analysis need to be executed and their results show to the user.

- **Task:** Implement small analyses and visualize their results on a website
- **Languages & Frameworks:** OPAL, Bash, Vue.js/React ...
- **Needed Experience:** Everything can be learned as you work
- **Suitable for:** 1 - 2 people
- **Contact:** [leonid.glanz@tu-darmstadt.de](mailto:leonid.glanz@tu-darmstadt.de)



# Data Flow to Critical Points

During an app security review, the flow of sensitive data is examined to identify potential leaks.

- **Task:** Identify all points that read/write external resources to track potential data leakages
- **Languages & Frameworks:**  **Scala**  **OPAL**
- **Needed Experience:** A bit Scala, the rest can be learned as you work
- **Suitable for:** 1 – 2 people
- **Contact:** [florian.breitfelder@tu-darmstadt.de](mailto:florian.breitfelder@tu-darmstadt.de)

# Field Source Detection



Various analyses focus on the data flow between methods, however, if fields contain important information, these are missed.

- **Task:** Identify usages of specific fields and report, which instructions use these fields.
- **Languages & Frameworks:**  **Scala**  **OPAL**
- **Needed Experience:** A bit Scala, the rest can be learned as you work
- **Suitable for:** 1 person
- **Contact:** [florian.breitfelder@tu-darmstadt.de](mailto:florian.breitfelder@tu-darmstadt.de)



# Visualising Data Flows

After analysis of the sensitive data flows these flows need to be shown to the analyst.

- **Task:** Visualize all data flows to show the dangers of data leakages in Android Apps
- **Languages & Frameworks:**  **Scala**  **OPAL** **D3.js**
- **Needed Experience:** A bit Scala, the rest can be learned as you work
- **Suitable for:** 1 - 2 people
- **Contact:** [florian.breitfelder@tu-darmstadt.de](mailto:florian.breitfelder@tu-darmstadt.de)

# Taint Analysis for WebAssembly/JS

The analysis should track tainted data between WebAssembly and JavaScript

- **Task:** Write a Taint Analysis for WebAssembly from and to JavaScript based on our WebAssembly framework WasmA in Combination with TAJIS
- **Languages & Frameworks:** Go, Java, WasmA, TAJIS
- **Needed Experience:** Go, Java and experience with analysis frameworks, (Taint Analysis)  
(Can also be learned during the work)
- **Suitable for:** 1 person
- **Contact:** [roth@cs.tu-darmstadt.de](mailto:roth@cs.tu-darmstadt.de)

# Taint Analysis for SQL

The analysis should determine whether tainted data is written into a database and read from a database.

- **Task:** Write a Taint Analysis for SQL that checks a Java program for sql statements and determines whether tainted data is written into and possibly again returned from a database
- **Languages & Frameworks:** Java, SQL
- **Needed Experience:** Java, SQL, (Taint Analysis)
- **Suitable for:** 1 person
- **Contact:** [roth@cs.tu-darmstadt.de](mailto:roth@cs.tu-darmstadt.de)

# Taint Analysis for Java/Javascript

The analysis should track information flows between Java/Javascript

- **Task:** Write an analysis in OPAL, that determines whether JavaScript is called and then hand it over to TAJs and the other way around
- **Languages & Frameworks:** Java, JavaScript, Opal, TAJs
- **Needed Experience:** Java, JavaScript, (Taint Analysis)
- **Suitable for:** 1 person
- **Contact:** [roth@cs.tu-darmstadt.de](mailto:roth@cs.tu-darmstadt.de)

# GitHub Data Set Crawler

The crawler should crawl Java projects from GitHub with different filters, e.g., stars, commits, activity, then it should build the projects, and collect information of API usages, e.g., JCA.

- **Task:** Write a crawler that collects open-source Java projects based on different filters. The crawler is extended with a functionality to automatically build the projects. Further, it should collect information about API usages.
- **Languages & Frameworks:** Java/Scala/Go/Python, Docker, opt. Bash
- **Needed Experience:** Experience with one of the languages, the rest will be learned as you go
- **Suitable for:** 1-2 people
- **Contact:** [wickert@cs.tu-darmstadt.de](mailto:wickert@cs.tu-darmstadt.de)

# Benchmark Evaluation Suite

The benchmark evaluation suite should be able to execute different crypto API misuse detection tools on existing source code and binaries. Furthermore, the results should be parsed and unified across the different detection tools.

- **Task:** Implement a benchmark evaluation suite that can execute different crypto API misuse detection tools such as CogniCrypt and CryptoGuard on provided source code and binaries. Write a parser that unifies the different output formats from the detection tools.
- **Languages & Frameworks:** Java/Scala/Go/Python, Docker, opt. Bash
- **Needed Experience:** Experience with one of the languages, the rest will be learned as you go
- **Suitable for:** 1-2 people
- **Contact:** [wickert@cs.tu-darmstadt.de](mailto:wickert@cs.tu-darmstadt.de)

# Positions & Theses

If you are interested in **HiWi Positions** or **Bachelor- or Master theses** contact:

[leonid.glanz@tu-darmstadt.de](mailto:leonid.glanz@tu-darmstadt.de)