UNIVERSITÄT BONN

Secure Software Engineering

Exercise Sheet 6, Winterterm 25/26

Discussion Week: Tue 16.12.25 to Fri 19.12.25

We highly encourage you to do the assignments yourselves, as all the exercises are relevant for the exam. It is also recommended to visit the tutorial, as there will be a discussion about the exercises, not just the solutions. For the *Vulnerability of the Day* exercises, we recommend looking into the code snippets, if you find any, while doing research. Keep in mind, the entirety of the lecture is relevant for the exam. If a topic or some details are not specifically talked about or mentioned in the exercises, that does **not** mean that it will not be part of the exam!

If you have any questions, please do not hesitate to ask your tutor Aura, Kati or Lukas. (Hint: the mails are embedded in the names.)

Good luck and have fun! :)

Ex. 1 - Code Quality and Security Practices

Answer the following sub-tasks:

a) Explain what a *code smell* is and why it matters for security. Give two examples and their corresponding refactorings.

b) Describe the Test-Driven Development (TDD) cycle. Is there a security benefit of writing tests before implementation?

c) Why is using `printf(str)` instead of `printf("%s", str)` a vulnerability? How can we exploit it?

Ex. 2 - Static Analysis

a) Explain the difference between false positives and false negatives. Which one is more dangerous from a security perspective, and why?

b) What are the four core elements of taint analysis in SAST tools? Briefly describe each.

c) At which phases of the software development lifecycle can SAST be applied? Give the pros and cons per phase.

d) What's the difference between a linter and full blown static analysis tool?

Ex. 3 Alternative 1 - Static Analysis with Semgrep

Use Semgrep to analyze the Log4j library source code (vulnerable version) and understand the benefits and limitations of static analysis tools. Before doing the task, install semgrep as CLI and VSCode plugin and clone the vulnerable Log4j version < 2.14.1 from github.

Tasks:

a) Run Semgrep with the Java security ruleset on the Log4j source code. What are your observations? Explain the potential bugs.

b) Examine the vulnerable file directly: `JndiLookup.java`

Identify the dangerous part. Does Semgrep flag this as a vulnerability? Why or why not?

c) Scan WebGoat. Are Semgrep's findings legit?

Ex. 3 Alternative 2 - Static Analysis with clang (Hard and Slow)

Clone FFmpeg and checkout a version with a known vulnerability, e.g. n5.1 containing CVE-2022-2566

Tasks:

a) Scan the vulnerable file with: `clang -analyze libavformat/mov.c -I. -I./libavutil`. Explain the command. Does the analyzer find anything? Explain.

b) Now use `scan-build configure && make -j$(nproc)` from clang to scan the entire project. How many issues did it find? Discuss your observations and results.

Ex. 4 - Vulnerability of the Day

In the lecture you have talked about *uncontrolled format string* and *OS command injection*. Research an example of this which was not discussed and explain what happened, how it happened and how it was dealt with. In addition to that, explain which of the CIA properties were affected. (You can look up a CVE, in case you cannot find an attack on a company or something similar.)