UNIVERSITÄT BONN

Secure Software Engineering

Exercise Sheet 5, Winterterm 25/26

Discussion Week: Tue 08.12.25 to Fri 12.12.25

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 - Defensive Coding

Answer the following sub-tasks:

a) How is *defensive coding* different from *risk analysis* and *secure design*?

b) What are the *defensive coding principle*?

Ex. 2 - Complexity

Explain the following statement: "Complexity is the enemy of security" (– Gary McGraw)

Ex. 3 - Control flow graph

Look at the mov.c file from FFmpeg repository. a) Draw a control flow graph and calculate the cyclomatic complexity for the `mov_build_index` function (which had at ĩo CVEs in the past). b) How can this function be simplified?

You are allowed to use tools to automate this. Most LLMs will produce inaccurate data for call graph analysis and cyclomatic complexity, though.

Ex. 4 - Vulnerability of the Day

In the lecture, we talked about *Server-Side Request Forgery*. Research an example of this that 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.)

Ex. 5 - Log4Shell (optional)

Familiarize yourself with the Log4Shell vulnerability and attempt to understand its components. Note: Run this lab on an Ubuntu VM.

Setup

1. Git clone `https://github.com/kozmer/log4j-shell-poc`
2. Build and run the vulnerable web app:
   `docker build -t log4j-shell-poc .  && docker run -network host log4j-shell-poc`
3. Download JDK 1.8.0_20 from Oracle's archive (account required) and extract it into the repo directory (needed to compile the exploit payload)

4. Run the exploit: `python3 poc.py -userip 127.0.0.1 -webport 8000 -lport 9001`

Questions

1. How does the vulnerability in Log4j work in terms of JNDI?
2. Why does the attacker need their own LDAP server?
3. Perform a remote code execution by executing any application on the web server.
4. How would you mitigate this vulnerability if the patch wasn't available?