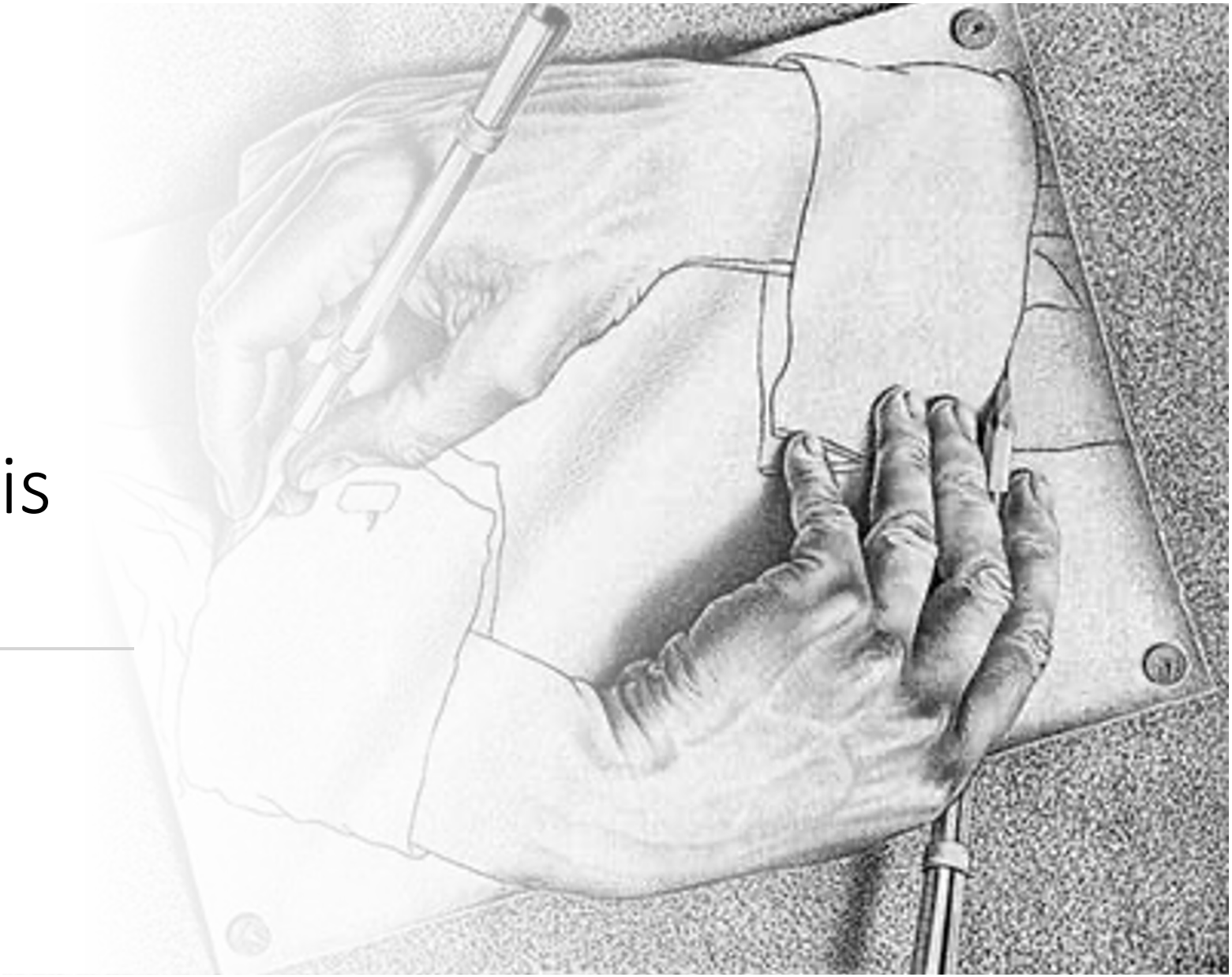
UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

Static Analysis Practical Lab



Software Security

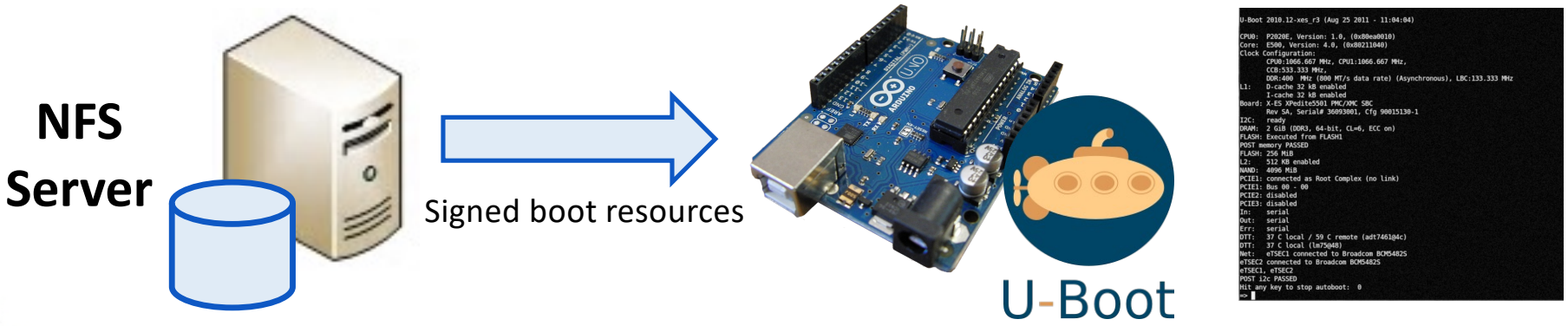
a.a. 2023/2024

Laurea Magistrale in Ing. Informatica

Roberto Natella

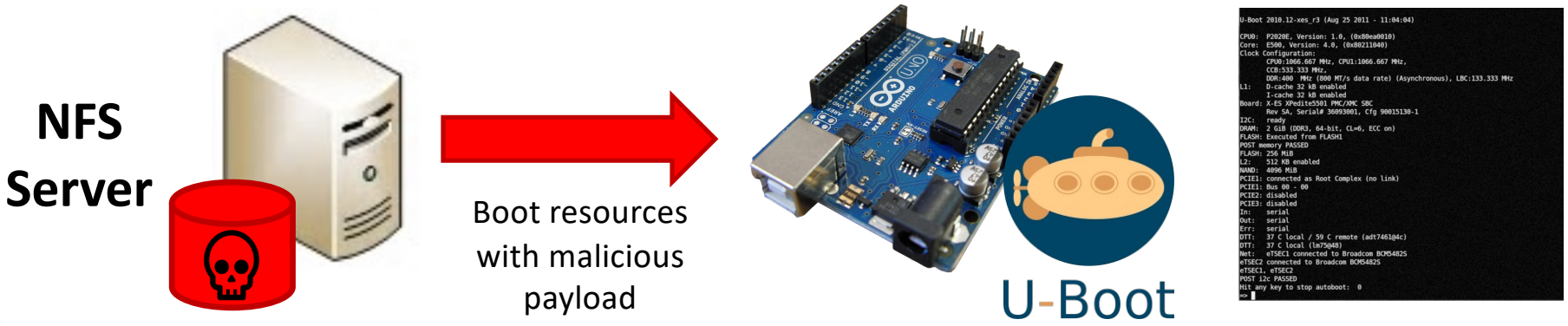
Lab Static Analysis

- U-Boot
  - Bootloader for embedded devices
  - Fetches boot resources (e.g., the Linux kernel) from the network
  - Verifies digital signature



Lab Static Analysis

- Goal: Find **Remote-Code-Execution (RCE)** vulnerabilities in U-Boot
- Attacker can take control of U-Boot before verified boot



https://securitylab.github.com/research/uboot-rce-nfs-vulnerability/

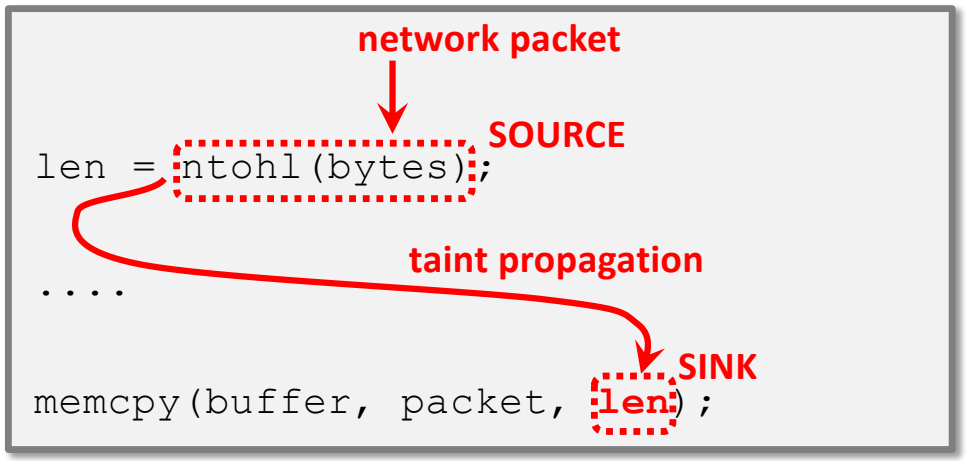
Lab Static Analysis

UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

- Your goal is to write a query to track down **unsafe calls to memcpy()**

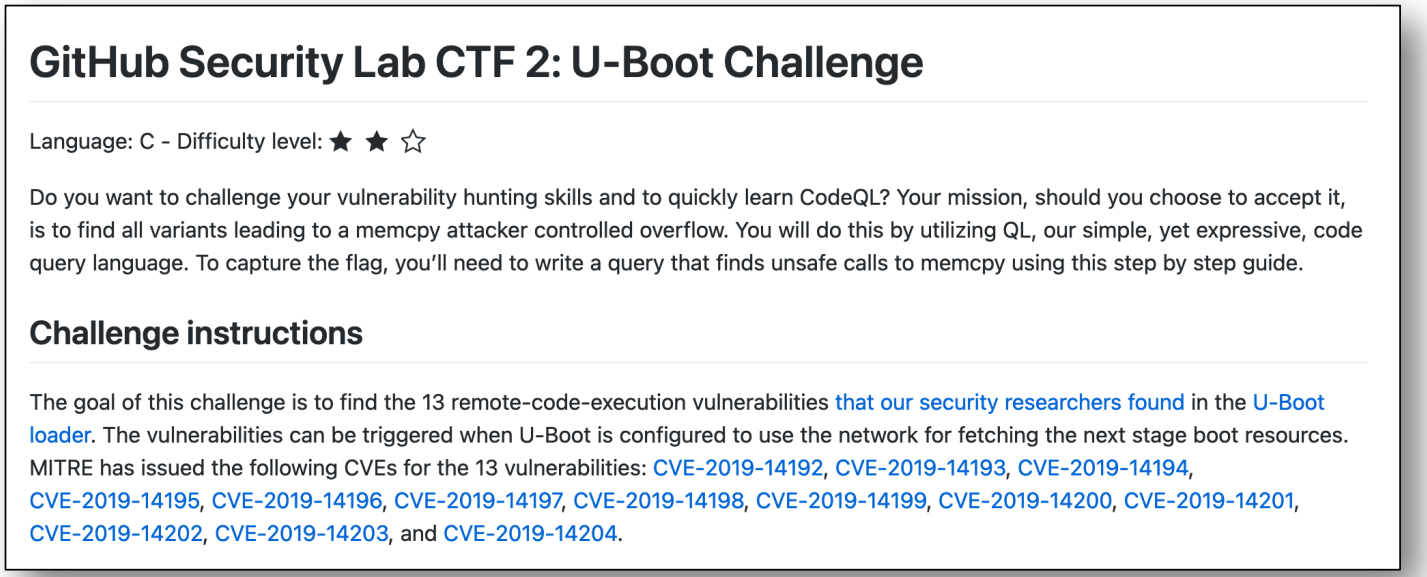
- U-Boot contains **hundreds of function calls** that
  - **read data** from the network **(source)**
  - **pass the data** to **memcpy() as "size" parameter (sink)**
  - … often without validation before use

Lab Static Analysis



ntohs() (network to host short) and ***ntohl()*** (network to host long) convert data from network ordering to the **host's native byte ordering**

Lab Static Analysis



GitHub Security Lab CTF 2: U-Boot Challenge

Language: C - Difficulty level: ★ ★ ☆

Do you want to challenge your vulnerability hunting skills and to quickly learn CodeQL? Your mission, should you choose to accept it, is to find all variants leading to a memcpy attacker controlled overflow. You will do this by utilizing QL, our simple, yet expressive, code query language. To capture the flag, you'll need to write a query that finds unsafe calls to memcpy using this step by step guide.

Challenge instructions

The goal of this challenge is to find the 13 remote-code-execution vulnerabilities that our security researchers found in the U-Boot loader. The vulnerabilities can be triggered when U-Boot is configured to use the network for fetching the next stage boot resources. MITRE has issued the following CVEs for the 13 vulnerabilities: CVE-2019-14192, CVE-2019-14193, CVE-2019-14194, CVE-2019-14195, CVE-2019-14196, CVE-2019-14197, CVE-2019-14198, CVE-2019-14199, CVE-2019-14200, CVE-2019-14201, CVE-2019-14202, CVE-2019-14203, and CVE-2019-14204.

https://securitylab.github.com/ctf/uboot/

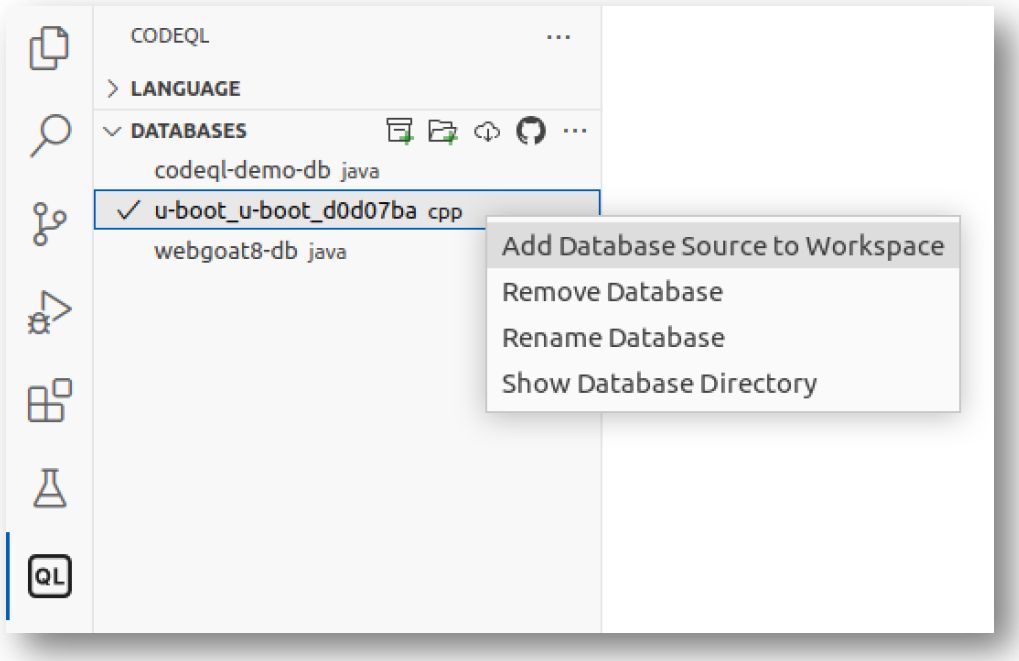
Setup

- Download **CodeQL DB** from:
  https://github.com/github/securitylab/releases/download/u-boot-codeql-database/u-boot_u-boot_cpp-srcVersion_d0d07ba86afc8074d79e436b1ba4478fa0f0c1b5-dist_odasa-2019-07-25-linux64.zip

  Note: the file is already available in our VM

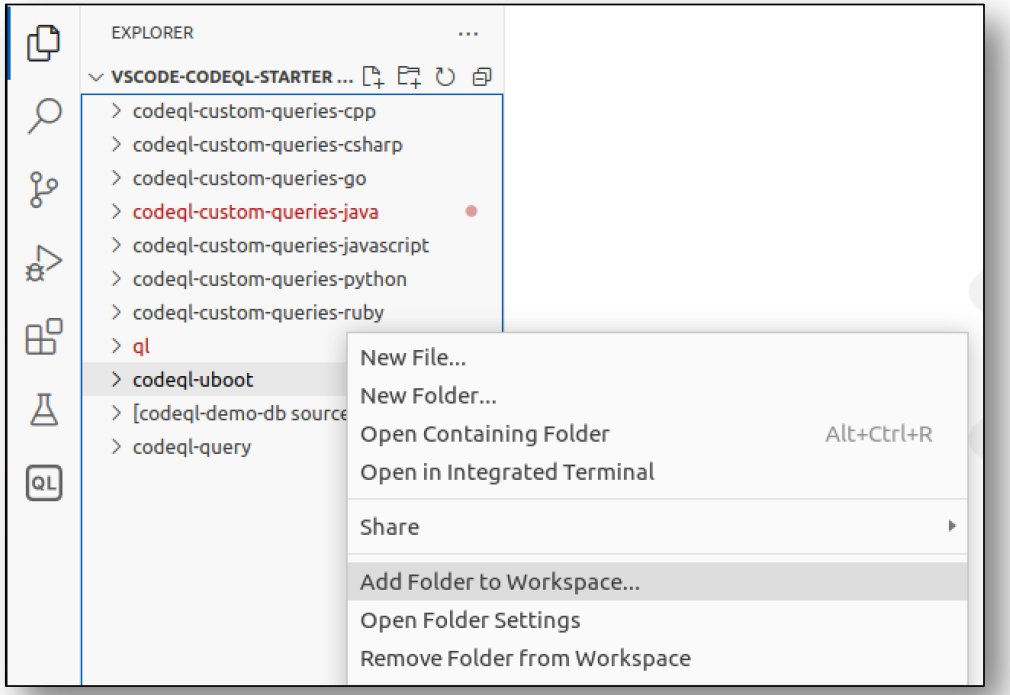
- **Import** into VS Code

- (Optional, to explore the source code of u-boot) Add "**Database Source**" to workspace



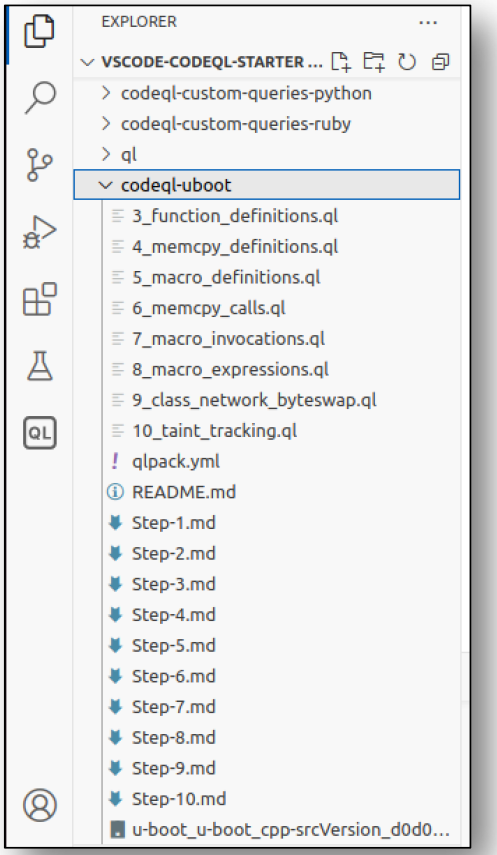
Setup

- Open the **CodeQL workspace** in VSCode
- Right-click in the Explorer area, "**Add Folder to Workspace**"
- Select **static-analysis/codeql-uboot** from the repository of the class



Setup

- Follow instructions in **Step-1.md**, **Step-2.md**, etc.
- Write your queries in the **.ql files**



Instructions are also available at:
https://github.com/rnatella/software-security/tree/main/static-analysis/codeql-uboot

Roadmap

1. Find all functions named memcpy
2. Find all ntoh* macros
3. Find all the calls to memcpy
4. Find all the invocations of ntoh* macros
5. Find the expressions that correspond to macro invocations
6. Write your own NetworkByteSwap class
7. Write a taint tracking query

- Extra task: how to check for sanitization?

More labs – Static Analysis

- GitHub Security Lab CTF 4: CodeQL and Chill - The Java Edition https://securitylab.github.com/ctf/codeql-and-chill/
- CodeQL for JavaScript: Unsafe jQuery Plugin https://lab.github.com/githubtraining/codeql-for-javascript:-unsafe-jquery-plugin

More labs – CI/CD

- Hello GitHub Actions
  https://github.com/skills/hello-github-actions
- Test with Actions
  https://github.com/skills/test-with-actions
- Securing your workflows
  https://github.com/skills/secure-repository-supply-chain
- Secure code game
  https://github.com/skills/secure-code-game