

Static analysis

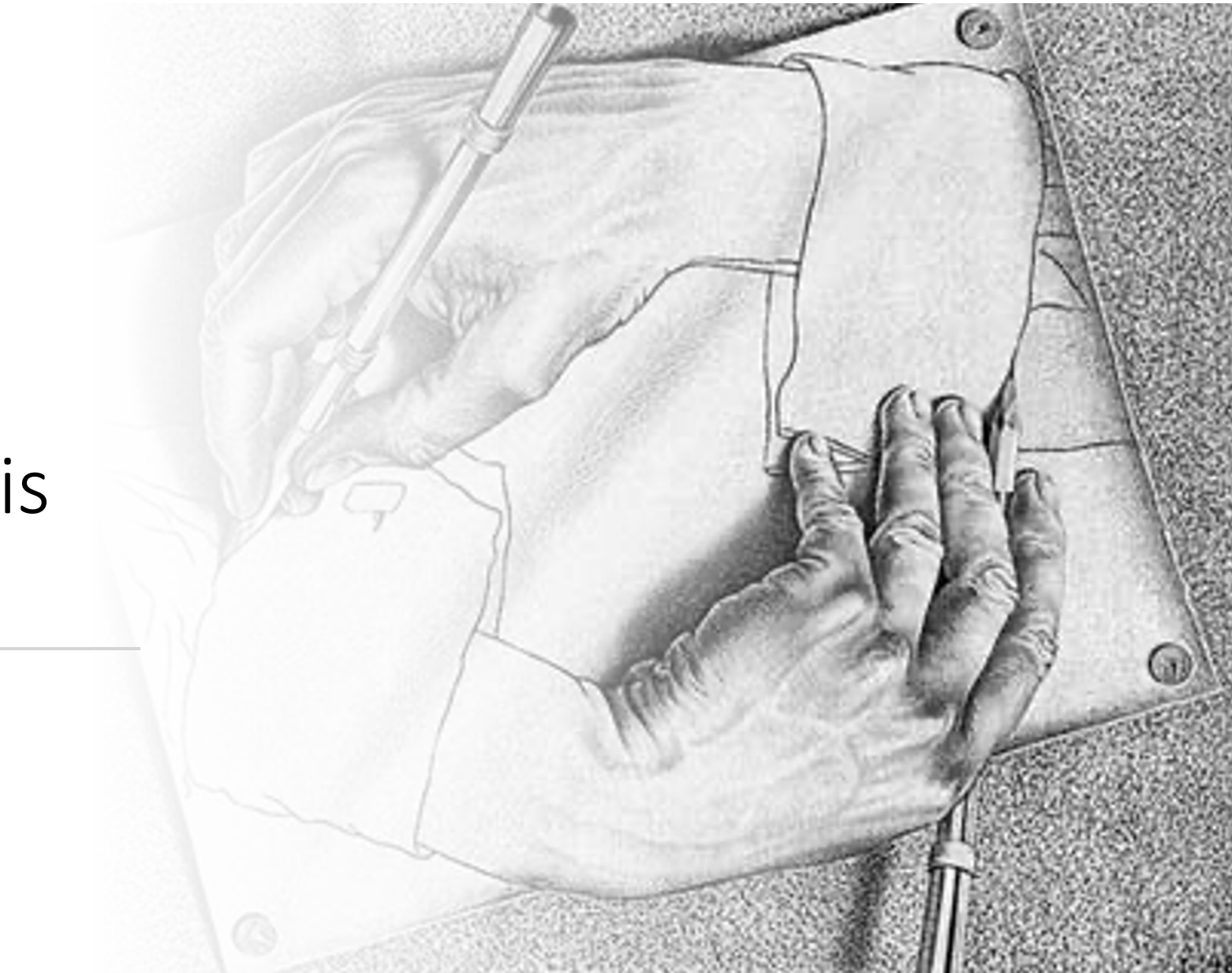
Practical lab

Software Security

a.a. 2022/2023

Laurea Magistrale in Ing. Informatica

Roberto Natella



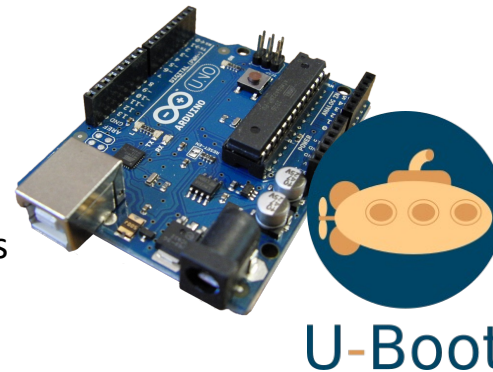
Lab Static Analysis

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

NFS
Server



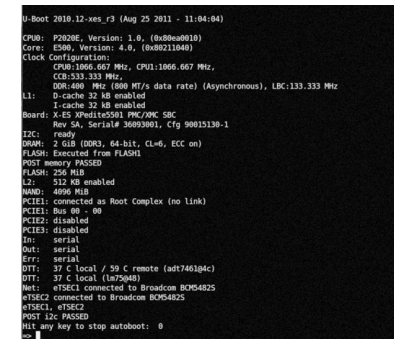
Signed boot resources



U-Boot

```
U-Boot 2018.12-xen_3 (Aug 25 2011 - 11:04:04)
CPU0: P0202E, Version: 1.0, (0x00e00010)
Core: E500, Version: 4.0, (0x0211040)
Clock Configuration:
CPU0:1066.667 MHz, CPU1:1066.667 MHz,
CPU2:333.333 MHz,
CPU3:400 MHz (500 MT/s data rate) (Asynchronous), LDC:133.333 MHz
L1:
D-cache 32 kB enabled
I-cache 32 kB enabled
Board: X-ES Xpedite5001 PWC/PMC SRC
New SA, Serial# 36993001, Cfg 90013130-1
I2C: ready
DRAM: 2 GiB (DDR3, 64-bit, CL=6, ECC on)
FLASH: Executed from FLASH
POST memory PASSED
FLASH: 256 MiB
L2: 512 kB enabled
NAND: 4096 MiB
PCI0: connected as Root Complex (no Link)
PCI1: Bus 00 - 00
PCI2: disabled
PCI3: disabled
In: serial
Out: serial
Err: serial
DTT: 37 C local / 38 C remote (ad7f6184c)
DTT: 37 C local (in75040)
Net: e7SEC1 connected to Broadcom BCM4425
e7SEC2 connected to Broadcom BCM4425
e7SEC1, e7SEC2
POST I2C PASSED
Hit any key to stop autoboot: 0
=>
```

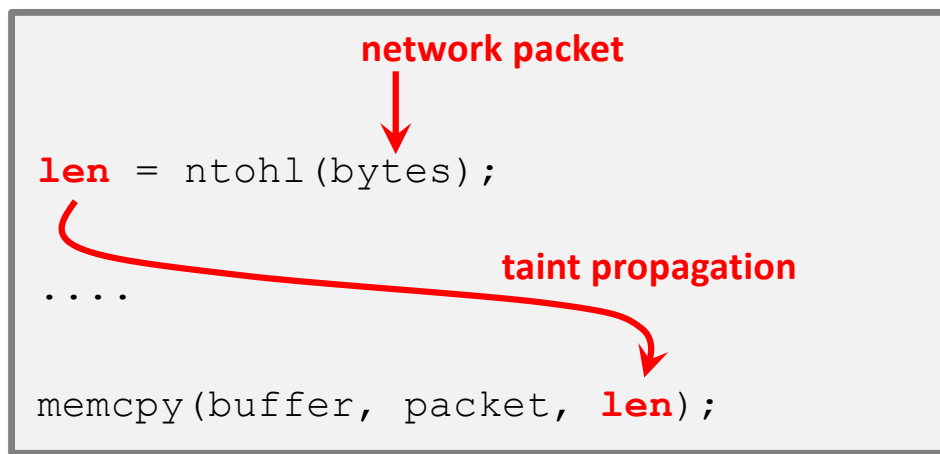
- Find **Remote-Code-Execution (RCE)** vulnerabilities in U-Boot
- Attacker can
 - take control of U-Boot before verified boot
 - tamper boot resources



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

Lab Static Analysis

- Your goal is to write a query to track down **unsafe calls to memcpy**
- U-Boot contains **hundreds of calls** to
 - libc functions/macros that **read data from the network (source)**
 - memcpy to **copy the data to memory (sink)**
 - input data is **passed to memcpy() as "size"**, but often not validated before use



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/>



Roadmap

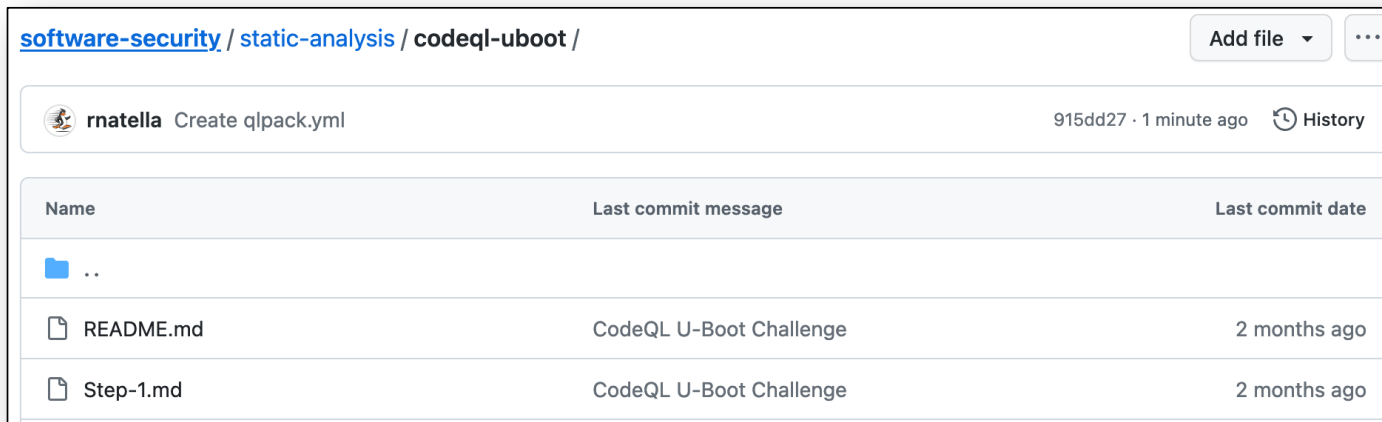
1. Find all functions named memcpy
2. Find all ntohs* macros
3. Find all the calls to memcpy
4. Find all the invocations of ntohs* macros
5. Find the expressions that correspond to macro invocations
6. Write your own NetworkByteSwap class
7. Write a taint tracking query

- Extra task: how to check for sanitization?



Roadmap

- Clone/pull from <https://github.com/rnatella/software-security>
- Go to subfolder **static-analysis/codeql-uboot**
- Follow instructions in **Step-1.md**, **Step-2.md**, etc.
- You can download the 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
- Run your queries on VSCode in your computer



The screenshot shows a GitHub repository interface. At the top, the breadcrumb path is 'software-security / static-analysis / codeql-uboot /'. Below this, a commit by 'rnatella' is shown with the message 'Create qlpack.yml' and a timestamp of '915dd27 · 1 minute ago'. A table below lists the repository's contents:

Name	Last commit message	Last commit date
..		
README.md	CodeQL U-Boot Challenge	2 months ago
Step-1.md	CodeQL U-Boot Challenge	2 months ago



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>

