Chair of IT Security
Department of Informatics
Technical University of Munich

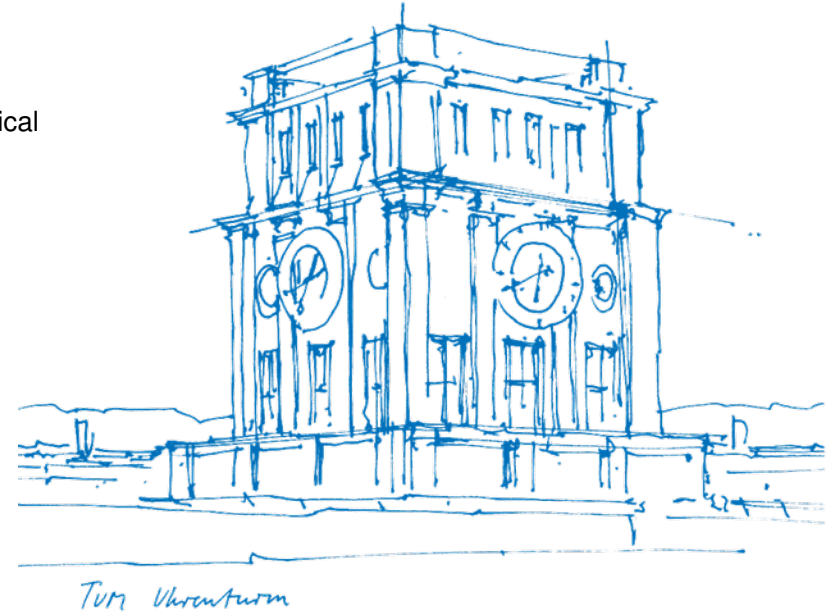
Low-Level Software Security

Preliminary Meeting - SS 2023 - Season II

Marius Momeu Manuel Andreas

Chair of IT Security - School of Computation, Information and Technology - Technical University of Munich (TUM)

January 26, 2023



Chair of IT Security
Department of Informatics
Technical University of Munich

Intro

Your tutors:

- Marius Momeu (momeu@sec.in.tum.de)

Chair of IT Security
Department of Informatics
Technical University of Munich

Intro

Your tutors:

- Marius Momeu (momeu@sec.in.tum.de)
- Manuel Andreas (andreas@sec.in.tum.de)

Objectives

This is a **practical-oriented lab** where you will practice systems development on prototypes that address trending topics in the area of *low-level software security*[1].

As such, your tutors will define tasks based on state-of-the-art research, which you will have to **design, implement, and evaluate a prototype** for.

You will then **describe your prototype and findings in a final written report**, and **present them in a final talk** at the end of the semester.

Goals in each topic may be **open-ended, exploratory, and agile** (i.e., reiterated and adjusted along the way as roadblocks arise).

[1] aka *systems software security*

Chair of IT Security
Department of Informatics
Technical University of Munich

Technical Content

Our topics aim to improve the security of *low-level* software, typically written in *memory unsafe* languages (C/C++/Assembly), in the context of well-known processor architectures (**ARM and x86 (Intel and AMD)**), and maybe even emerging ones (**RISC-V**). The following (non-exhaustive) list captures a number of broad areas we will pick topics from[2]:

- **Software hardening**
  - either software-based or using hardware extensions (such as *Intel VT-x/MPK/CET/HLAT* and *ARM PAC/MTE*)
  - to design code/data isolation, code/data debloating, control-/data-flow integrity schemes
  - for hardening OS kernels, containers, unikernels, μkernels
- **Software analysis**
  - via static/dynamic program analysis for generating and enforcing control-/data-flow policies
  - or via program testing (fuzzing or symbolic execution) for finding bugs
  - in closed- and open-source low-level software (OS kernels, device drivers, hypervisors)
- **Microarchitectural flaws**
  - and side-channels for leaking secrets, revealing stealthy monitors, etc.
- **CPU security extension design**
  - e.g., on RISC-V
- **Confidential computing in Trusted Execution Environments (TEEs)**
- **Remote (control-flow and data-flow) attestation**

[2] see more concrete examples on the lab's webpage description.

Hands-On Format

Throughout this lab you should expect to touch on (several) hands-on stuff, including but not limited to:

- Remotely operating servers or IoT devices via the command-line terminal (bash on Unix systems)
- System administration (e.g., spawning VMs, managing partitions, compiling and deploying kernels/unikernels)
- **Reading and coding in *C/C++/Assembly*** (*x86, ARM*), (maybe) *Rust*, and various scripting languages
- Understanding OS concepts, such as memory management (via paging or nested-paging[3]), interrupts, (bare-metal and emulated) device drivers, syscalls/hypercalls
- Using *LLVM*'s static analysis framework and *LLVM* binary lifters
- Examining various hardware extensions in architecture manuals (*Intel VT-x/MPK/CET/HLAT, ARM PAC/MTE, AMD-SEV-**)
- Understanding/working with software testing techniques: blackbox/whitebox/graybox fuzzing (coverage guidance, input mutation), state-of-the-art fuzzers (kAFL, syzkaller), symbolic/concolic execution, constraint solvers (z3)
- Computer architecture concepts (e.g., speculative execution, return stack buffers, caches, *TLBs*)
- Exploitation know-how: code-reuse attacks, data-oriented attacks, secret leaking via covert side-channels
- Compiling/building, dynamic or static linking, binary formats (mostly *ELF*)

[3] via *PTs* and *EPTs* on Intel's architecture

To be clear:

This is not an introductory lab in software security!

- There will **not** be ready-made solutions for you to find on $SEARCHENGINE, $FORUM, ...
- You **will** have to dive deep into complex code bases / technologies
- Things **may not** work out as expected

However:

- This is an **excellent opportunity** to get familiar with state-of-the-art security research
- You gain **valuable practical skills** in working with sophisticated technology (likely not taught in the curricula)
- If mutual interest exists: you get the chance to participate in any **scientific publication** that may emerge out of this work

Process

Three prototype development phases:

1. Designing
2. Implementation
3. Evaluation

Four presentation meetings:

1. Research Expose
2. System Design
3. Status update and issues (bilateral)
4. Final talk

Two report deliverables:

1. Intermediate draft
2. Final version

Grading

Graded deliverables:

- Design / Prototype / Evaluation
- Final presentation
- Final report

Mandatory ungraded deliverables:

- Intermediate presentations
- Regular status updates on your prototype

Optional ungraded deliverables:

- Draft slides for the final presentation (to get our feedback on)
- Intermediate report draft

	80 %	Design / Prototype / Experiments
	10 %	Final Talk (Presentation and Q&A)
	10 %	Final Report (Content and Structure)
Σ	100 %	Final Grade

Disclaimer: The grading scheme above might suffer slight modifications.

Deliverables' Format

Prototype:

- source code (documented and cleaned-up)
- reproducible evaluation experiments
- guideline (`README`) for compiling, installing, and evaluating

Presentation:

- TUM presentation template[4]
- custom templates can be used as well
- 16:9 aspect ratio

Generally, we encourage you to use LaTeX for writing.

Report/Writeup:

- complete description and findings of the prototype
- informal language and style (no scientific writing constraints), max. 10 pages
- we will provide a template

[4] https://sharelatex.tum.de/templates/tum-templates/tum-presentation-v2.0.0

Chair of IT Security
Department of Informatics
Technical University of Munich

Logistics

When? Irregularly ($\approx$ 4 mandatory meetings), on Tuesdays, at 10:00h (exact dates & time TBA)

Where? On-site in our meeting room: 01.08.033

Language: English

Course of study: both Master's and Bachelor's students

Capacity: 16 students (8 teams, 2 students / team)

Registration: Via the matching system

Course Resources

Moodle[5] **page** for announcements, for submitting deliverables, and for uploading lecture slides.

Gitlab[6] **repositories** on LRZ's git server where you can keep your prototype's source code.

ARM/Intel/AMD machines for prototyping / running experiments, depending on your topic.

Matrix[7] for instant messaging with team partners and tutors.

Your tutors for brainstorming and addressing issues.

[5] https://www.moodle.tum.de/
[6] https://gitlab.lrz.de/
[7] https://matrix.tum.de/

Qualification Challenge

To test your readiness for tackling our topics, we propose a qualification challenge that will book you a seat in the lab upon completion:

1. Clone and compile the latest Linux kernel
2. Generate a root file system with `debootstrap`[8]
3. Spawn a QEMU/KVM virtual machine that boots your freshly compiled kernel
4. Write a loadable kernel module (LKM) and load it in your VM
5. Finally, modify your LKM to print "Hello World" to the kernel ring buffer upon initialization

If this is too technical for you, and/or you're not interested in diving deep into these low-level concepts then this course is probably not for you!

Deliverable:

- A brief writeup of how you have solved the challenge
- Any scripts or source code you may have written to solve the challenge, e.g. the LKM source code, a script to boot QEMU with your kernel etc. combined in a *single* tar file.

[8] https://wiki.debian.org/Debootstrap

Chair of IT Security
Department of Informatics
Technical University of Munich

Qualification Challenge (contd.)

Still Interested?

Write us an E-Mail to momeu@sec.in.tum.de and andreas@sec.in.tum.de
In your E-Mail, please use the subject: *Matching - LLSS - SS 2023*

We will consider the following for your approval to our practical course:

1. **Mandatory**: Succesful completion and documentation of the qualification challenge.
2. **Optional but nice**: Mention successful completion of any of the following courses:
   - Binary Exploitation, Rootkit Praktikum
   - Systems Hardening, Software Security Analysis, Trusted Execution Environment, Reverse Engineering
   - IT Security, Secure Mobile Systems
   - Computer Architecture, Operating Systems
   - Any other course/thesis/project related to this security domain

Deadline: 19.02.2023, EoD

Chair of IT Security
Department of Informatics
Technical University of Munich

Q&A

Marius Momeu
momeu@sec.in.tum.de

Manuel Andreas
andreas@sec.in.tum.de