Low-Level Software Security
Preliminary Meeting - WS 2022/23 - Season I

Marius Momeu   Manuel Andreas

Chair of IT Security, Department of Informatics, Technical University of Munich (TUM)

July 19, 2022
Intro

Your tutors:

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

Your tutors:

• Marius Momeu (momeu@sec.in.tum.de)
• Manuel Andreas (manuel.andreas@tum.de)
Objectives

This lab will be a playground where you can build practical prototypes, which aim to solve limitations of 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\(^2\).

Alongside that, you will describe your prototype and findings in a final written report, and present them in a final talk at the end of the semester.

\(^1\) aka systems software security
\(^2\) you are welcome to propose topics of your own
Technical Content

We are generally researching ways to improve the security of low-level software running on most popular processor architectures: ARM and x86 (Intel and AMD). As such, the following list captures some high-level areas we will pick topics from:

- **Software Hardening**
  - 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, 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)

- **Confidential computing in Trusted Execution Environments (TEEs)**
  - such as ARM TrustZone, Intel SGX/MKTME/TXT, AMD-SEV-*
  - both security guarantees and vulnerabilities

- **Remote (control-flow and data-flow) attestation**

- **Microarchitectural flaws** and side-channels for leaking secrets, revealing stealthy monitors, etc.

---

3see 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\(^4\)), 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)

Disclaimer: this is not an introductory lab in software security!

\(^4\)via PTs and EPTs on Intel’s architecture
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
- Intermediate report draft

**Optional ungraded** deliverables:
- Draft slides for the final presentation (to get our feedback on)

<table>
<thead>
<tr>
<th>Weight</th>
<th>Component</th>
</tr>
</thead>
<tbody>
<tr>
<td>80 %</td>
<td>Design / Prototype / Experiments</td>
</tr>
<tr>
<td>10 %</td>
<td>Final Talk (Presentation and Q&amp;A)</td>
</tr>
<tr>
<td>10 %</td>
<td>Final Report (Content and Structure)</td>
</tr>
<tr>
<td></td>
<td><strong>Σ 100 %</strong> Final Grade</td>
</tr>
</tbody>
</table>

**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

Report/Writeup:

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

Presentation:

• TUM presentation template
• custom templates can be used as well
• 16:9 aspect ratio

Generally, we encourage you to use \LaTeX for writing.

[5] https://latex.tum.de/templates/608c2650db4bc7007f58c931
Logistics

**When?** irregularly (≈ 4 mandatory meetings), on Tuesdays, at 10:00h (exact dates & time TBA)

**Where?** Most likely onsite, unless the uni enforces online teaching

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

**Language:** English

**Course of study:** both Master’s and Bachelor’s students

**Registration:** via the matching system
Seminar Resources

**Moodle**⁶ page for announcements, for submitting deliverables, and for uploading lecture slides.

**Gitlab**⁷ 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**⁸ for instant messaging with team partners and tutors.

**Your tutors** for brainstorming and addressing issues.

---

⁶ [https://www.moodle.tum.de/](https://www.moodle.tum.de/)
⁷ [https://gitlab.lrz.de/](https://gitlab.lrz.de/)
⁸ [https://matrix.tum.de/](https://matrix.tum.de/)
Matching Prioritization

Each of the following will grant you a higher prioritization in the matching, with different weights. However, they are optional:

1 **Highest weight:** 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

2 **Medium weight:** letter of motivation of maximum two pages describing up to 3 concepts/technologies from the slide *[Technical Content]*. The description should touch on the questions *what do you know about these concepts?* and *why do you find them important for security?*

3 **Lowest weight:** presence in today’s premeeting

Send 1 and 2 via email to momeu@sec.in.tum.de and manuel.andreas@tum.de
In your email, please use the subject: *Matching - LLSS - WS 2022/23*

**Deadline:** Sunday, 31st of July, EoD
Bonus!

We’re planning to organize guest lectures and invite some of our friends from the infosec industry and academia.
Questions?

Marius Momeu
momeu@sec.in.tum.de
@MariusMomeu

Manuel Andreas
manuel.andreas@tum.de