Rootkit Programming

Premeeting

Manuel Andreas

Chair of IT-Security (I20)
Prof. Dr. Claudia Eckert
Technical University of Munich

January 30, 2025

What is a Rootkit?

> A rootkit is a collection of computer software, typically malicious, designed to enable access to a computer or an area of its software that is not otherwise allowed (for example, to an unauthorized user) and often masks its existence or the existence of other software.
> — Wikipedia

Course Contents

In this course you will create your **own rootkit** (aka your own piece of malware) with the following features:

- Escalate privileges to root
- Hide files on disk
- Hide processes
- Hide network connections
- ...

Your rootkit will take the form of:

- Userspace Rootkit
- Linux Kernel Module (LKM)
- Hypervisor
- ...

Further, we will focus on the **detection of rootkits** using

- Virtual Machine Introspection (VMI) / Memory Forensics

Teaching Goals

- How the kernel, the loader and the libc interact with each other to execute a program
- Details about the Linux kernel boot process (e.g. initramfs)
- Linux kernel hacking
  - How to create your **own kernel module**
  - How the Linux kernel tracing system works
  - Getting familiar with **fundamental linux subsystems**
- How modern hypervisors can **interact** with and **inspect** its running VMs

Prerequisites

We **do not** have formal requirements for students who want to join the course.

However, we **strongly recommend** being familiar with the following:

- How to write **a C program** and how **pointers** work
- What a **Syscall** is
- Basic knowledge about IT Security (IN0042) and how an operating system works in general (as taught in IN0009)

Having seen or worked with **assembly** is a plus!

Organizational Matters

- The course has 16 slots
- We will meet once a week
- You will work with a partner in teams of two

Organizational Matters

- The course has **16 slots**
- We will meet once a week
- You will work with a partner in teams of **two**
- Phase I:
  - **Weekly** exercises requiring you to implement new rootkit / detection mechanisms
  - Solutions are presented & discussed the following week

Organizational Matters

- The course has **16 slots**
- We will meet once a week
- You will work with a partner in teams of **two**
- Phase I:
  - **Weekly** exercises requiring you to implement new rootkit / detection mechanisms
  - Solutions are presented & discussed the following week
- Phase II:
  - **Project** phase
  - Come up with your own hiding or detection technique
  - Final presentation on your concept

Registration

Awesome! How can I join?

- **No** letter of motivation
- Instead, solve a **small qualification task**
  - Create a driver for our custom virtio device[1] (in the form of a Linux Kernel Module) in order to retrieve a secret value (**flag**).
  - Download the challenge & submit your flag at `https://courses.sec.in.tum.de/rootkit`
  - Due at **19.02.2024 23:59** (end of matching period)
  - **FCFS** based on your hand-in time
- Nonetheless, do not forget to **register** yourself in the **matching system**!

[1] The device is completely made-up in QEMU

Qualification Challenge Hints

- First steps:
  1. Download appropriate Linux kernel sources (**v6.11.11**)[2].
  2. Place our provided kernel configuration (**config-6.11.11**) into the kernel source tree and rename it to **.config**.
  3. Build the kernel: `make all`
  4. You can now start building your own Linux kernel module.
- For testing your module: insert it into our remote VM and debug via **printk**. For a more sophisticated setup, you may build the patched QEMU and setup your own local VM.
- If you simply want to reproduce our remote setup, we ship an appropriate **Dockerfile**.
- To figure out what type of data you must sent to our device we supply source code as a QEMU **patch**
  - Relevant parts implemented in `virtio-flagbrah.c` and `virtio_ids.h`
- Our challenge runs on an x86-64 CPU with **KVM** enabled.

[2] `https://cdn.kernel.org/pub/linux/kernel/v6.x/linux-6.11.11.tar.xz`

Qualification Challenge - Useful Resources

- General Linux kernel modules info:
  `https://linux-kernel-labs.github.io/refs/heads/master/labs/kernel_modules.html`
- **VirtIO LKM driver template**:
  `https://docs.kernel.org/driver-api/virtio/writing_virtio_drivers.html`
- Make sure to use correct memory for data transfers:
  `https://docs.kernel.org/core-api/dma-api-howto.html#what-memory-is-dma-able`

- Contact me at andreas@sec.in.tum.de

- Contact me at andreas@sec.in.tum.de

Questions?