Rootkit Programming

Fabian Franzen, Clemens Jonischkeit

Technische Universität München
Chair for IT-Security
Munich, Germany

29.01.2019

Overview

Goal: Create your own custom Linux rootkit[1] in course!

You will learn:

1. What a rootkit is
2. **Linux kernel principles and LKM programming**
3. How rootkits work from a practical perspective
4. How to detect and analyse rootkits
    - by digging a bit into Virtual Machine Introspection (VMI)

You will be working:

- in teams of two or on your own

[1]We will rely on Linux Kernel Modules (LKM)

Rootkits

What is a rootkit?
A *kit* (i. e., group of programs or functions) that allows an attacker to maintain *root* access.

What specific roles does a rootkit have?

1. provides a backdoor or way back into the system
2. make the admin belive that no backdoor is present
   - hides files, connections, etc.
3. overtime the term has been perverted and there are often additional elements implemented into a rootkit

Curriculum

Your rootkit will target **Debian 9** and its **4.9 kernel** (on a 64bit machine!)

- system call hooking
- file hiding
- process hiding
- module hiding
- socket hiding
- privilege escalation
- keylogging
- foundations of VMI

Modus Operandi & Requirements

- There will be weekly programming assignments.

To participate you must have...

- a programming background in C
  - the kernel is written in C
  - all assignments will be done in C
- root access to a machine[2] running Linux
- basic knowledge how operating systems work

[2]with x86-64 architecture and VM-x extensions

Time & Place

every **Tuesday 14:00 - 16:00 in MI 01.05.013**[3]

[3]but of course not during the semester break

Qualification Task

- Please solve this small **qualification task**
  - Set up a VM using QEMU for this course with Debian 9
  - Write a kernel module[4] that prints a process list on module load including these process properties:
    - PID (in root namespace)
    - PID (in its own namespace)
    - Comm (process name)
    - ID of PID-Namespace
    - ID of User-Namespace
    - ID of Network-Namespace
- Latest, until Wed, 13th February 2019 23:59 to franzen@sec.in.tum.de!
- Registration via **Matching System** neccessary!

[4] for the standard debian kernel

Tipps

- Use `unshare` (the shell tool) to create a testing namespaces
- You can also test using `docker`
- Namespace-IDs of processes can be seen in `/proc/<pid>/ns/`

Literature

- LXR Free Electrons[5] (source code browser)
- The Linux Kernel Module Programming Guide[6]
- Love, Robert. *Linux Kernel Development, Third Edition* (2010)[7]

[5] https://elixir.bootlin.com/linux/v4.9.133/source
[6] http://tldp.org/LDP/lkmpg/2.6/html/index.html
[7] http://proquest.safaribooksonline.com.eaccess.ub.tum.de/9780768696974

Questions?