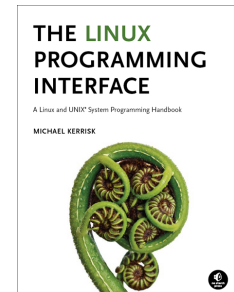


# Linux Security and Isolation APIs

Course code: M7D-SECISOL02

This course provides a deep understanding of the low-level Linux features—set-UID/set-GID programs, capabilities, namespaces, cgroups (control groups), and seccomp—used to implement privileged applications and build container, virtualization, and sandboxing technologies. Detailed presentations coupled with carefully designed practical exercises provide participants with the knowledge needed to understand, design, develop, and administer such applications. (The course does *not* cover administering container systems such as Docker and LXC, but provides participants with a good understanding of the underlying implementation and operation of such systems.)



## Audience and prerequisites

The primary audience comprises designers and programmers building privileged applications, container applications, and sandboxing applications. Systems administrators who manage such applications will also find the course of benefit.

Participants should have working knowledge of the fundamental system programming topics covered in the *Linux System Programming Essentials* (M7D-SPESS01) course. This includes file descriptors and file I/O, signals, and the process lifecycle (*fork()*, *exec()*, *wait()*, *exit()*).

Participants should have a good reading knowledge of the C programming language and some programming experience in a language such as C or Go. (Note, however, that most of the course exercises do not require writing programs.)

## Related courses

This course is also available as a number of smaller pieces:

- *Linux Capabilities and Namespaces*, M7D-CAPNS01
- *Linux Control Groups (Cgroups)*, M7D-CGROUPS02
- *Linux Secure Computing (Seccomp)*, M7D-SECCOMP01

## Course materials

- Course books (written by the trainer) that include all slides and exercises presented in the course
- An electronic copy of the trainer's book, *The Linux Programming Interface*
- Numerous example programs written by the course trainer

## Course duration and format

Four days, with around 40% of the course time devoted to practical sessions.

## Course inquiries and bookings

For inquiries about courses and consulting, you can contact us in the following ways:

- Email: [training@man7.org](mailto:training@man7.org)
- Phone: +49 (89) 2155 2990 (German landline)

## Prices, dates, and further details

For course prices, upcoming course dates, and further information about the course, please visit the course web page, <http://man7.org/training/secisol/>.

## About the trainer



Michael Kerrisk has a unique set of qualifications and experience that ensure that course participants receive training of a very high standard:

- He has been programming on UNIX systems since 1987 and began teaching UNIX system programming courses in 1989.
- He is the author of *The Linux Programming Interface*, a 1550-page book acclaimed as the definitive work on Linux system programming.

- He has been actively involved in Linux development, working with kernel developers on testing, review, and design of new Linux kernel–user-space APIs.
- Since 2000, he has been involved in the Linux *man-pages* project, which provides the manual pages documenting Linux system calls and C library APIs, and was the project maintainer from 2004 to 2021.

# Linux Security and Isolation APIs: course contents in detail

Topics marked with an asterisk (\*) may be covered, if time permits.

1. **Course Introduction**
2. **Security and Isolation APIs Overview (\*)**
  - Sandboxing
  - Containers
3. **Classical Privileged Programs**
  - A simple set-user-ID program
  - Saved set-user-ID and saved set-group-ID
  - Changing process credentials
  - A few guidelines for writing privileged programs
4. **Capabilities**
  - Process and file capabilities
  - Permitted and effective capabilities
  - Setting and viewing file capabilities
  - Capabilities-dumb and capabilities-aware applications
  - Text-form capabilities
5. **Capabilities and `execve()`**
  - Capabilities and `execve()`
  - The capability bounding set
  - Inheritable capabilities
  - Summary of process capability sets (so far)
  - Ambient capabilities
  - An alternative summary of process capability sets
  - Summary remarks
6. **Capabilities and UID 0**
  - Capabilities and UID transitions
  - Capabilities, UID 0, and `execve()`
  - Making a capabilities-only environment: `securebits (*)`
7. **Programming with capabilities (\*)**
  - Programming with capabilities
8. **Namespaces**
  - An example: UTS namespaces
  - Namespaces commands
  - Namespaces demonstration (UTS namespaces)
  - Namespace types and APIs
  - Namespaces, containers, and virtualization
9. **Mount Namespaces and Shared Subtrees**
  - Mount namespaces
    - Shared subtrees
    - Bind mounts
10. **PID Namespaces**
  - PID namespaces
11. **Other Namespaces**
  - IPC namespaces
  - Time namespaces
  - Cgroup namespaces
  - Network namespaces
12. **Namespaces APIs**
  - API Overview
  - Creating a child process in new namespaces: `clone()`
  - `/proc/PID/ns`
  - Entering a namespace: `setns()`
  - Creating a namespace: `unshare()`
  - PID namespaces idiosyncrasies
  - Namespace lifetime (\*)
13. **User Namespaces**
  - Overview of user namespaces
  - Creating and joining a user namespace
  - User namespaces: UID and GID mappings
  - Accessing files (and other objects with UIDs/GIDs)
  - User namespaces, `execve()`, and user ID 0
  - Security issues
  - Use cases
  - Combining user namespaces with other namespaces
14. **User Namespaces and Capabilities**
  - User namespaces and capabilities
  - What does it mean to be superuser in a namespace?
  - Discovering namespace relationships
  - File-related capabilities (\*)
  - User namespace “set-UID-root” programs (\*)
  - Namespaced file capabilities (\*)
15. **Mount Namespaces: Further Details (\*)**
  - Peer groups
  - Private mounts
  - Slave mounts
  - Unbindable mounts
  - Mounting a container filesystem
16. **Seccomp**
  - Seccomp filtering and BPF
  - The BPF virtual machine and BPF instructions
  - BPF filter return values
  - BPF programs
  - Checking the architecture
  - Productivity aids (*libseccomp* and other tools)
  - Applications and further information
  - Other filter return actions (\*)
17. **Seccomp: Further Details (\*)**
  - Caveats
  - Discovering the system calls made by a program
  - Further details on seccomp filters
  - Extended BPF (eBPF)
  - Further details on BPF programs
  - Recent seccomp features
  - Audit logging of filter actions
18. **Cgroups: Introduction**
  - Preamble
  - What are control groups?
  - An example: the `pids` controller
  - Creating and destroying cgroups
  - Populating a cgroup
  - Enabling and disabling controllers
19. **Cgroups: A Survey of the Controllers**
  - The `cpu`, `memory`, `freezer`, and `pids` controllers
  - Other controllers
20. **Cgroups: Advanced Features**
  - Cgroup namespaces
  - Release notification (`cgroup.events` file)
  - Delegation
21. **Cgroups: Thread Mode (\*)**
  - Overview of thread mode
  - Creating and using a threaded subtree
22. **Cgroups Version 1 (\*)**
  - Cgroups v1: hierarchies and controllers
  - Cgroups v1: populating a cgroup
  - Cgroups v1: release notification
  - Cgroups v1: delegation
  - Problems with cgroups v1; rationale for v2