Linux Kernel Programming Training

Duration: **5 Day**

BİLGİNÇ IT Academy
info@bilginc.com
bilginc.com

Overview

This five-day Linux security course provides a comprehensive introduction to the Linux Kernel, focusing on its architecture, core subsystems, and practical driver development techniques.

Participants gain both theoretical knowledge and hands-on experience by working directly with modern Linux kernel source code. Throughout the training, learners develop and extend a sample **character device driver**, enabling them to understand how kernel modules interact with the operating system.

The course also explores the evolution of the Linux Kernel across multiple releases, covering architectural improvements and features introduced up to **Linux kernel version 6.17**.

Key topics include:

- Kernel mode vs user mode execution
- Process scheduling mechanisms
- Memory management inside the kernel
- Device management and hardware abstraction
- Kernel security mechanisms

Participants will analyze real kernel code, trace execution paths, and follow best practices for safe and portable kernel module development.

By the end of the course, learners will be able to confidently navigate kernel sources, develop device drivers, and understand kernel behavior from both functional and security perspectives.

Prerequisites

To get the most value from this training, participants should have:

- Familiarity with user-mode programming concepts
- Basic understanding of POSIX standards and system calls
- Experience in C systems programming

Linux Kernel Programming Training

Duration: **5 Day**

BİLGİNÇ IT Academy
info@bilginc.com
bilginc.com

Who Should Attend

This course is designed for professionals such as:

- Linux device driver developers
- Cyber security professionals investigating malware or rootkits
- Software engineers interested in Linux Kernel internals
- Technical specialists working with low-level system integration or performance optimization

What You Will Learn

After completing the course, participants will be able to:

- Explain Linux Kernel architecture and subsystem roles
- Understand the difference between kernel mode and user mode execution
- Explore and analyze Linux Kernel source code
- Build and deploy a working Linux character device driver
- Understand process scheduling and kernel threading
- Explain how virtual memory works inside the kernel
- Trace and analyze system calls
- Use kernel debugging and tracing tools effectively
- Work with important kernel data structures such as task_struct
- Apply best practices for stable and secure kernel module development
- Understand Linux kernel security mechanisms and modules

Outline

Introduction to the Linux Kernel

- Linux architecture and design principles
- Kernel version evolution from 2.6 to 6.17
- User mode vs kernel mode execution
- Device handling and hardware abstraction

BİLGİNÇ IT Academy

info@bilginc.com

bilginc.com

- Task scheduling overview
- I/O scheduling concepts

Kernel Debugging and Tracing

- Kernel debugging techniques
- System call tracing using strace and ltrace
- Kernel tracing with tracefs
- Logging with dmesg and /proc/kmsg
- Exploring /proc and /sys virtual filesystems

Working with Kernel Sources

- Navigating the Linux kernel source tree
- Directory structure and subsystem layout
- Using Linux Cross Reference tools
- Kernel compilation and cross-compilation

Linux Kernel Boot Process

- Boot sequence from firmware to user space
- BIOS and EFI boot mechanisms
- Boot loaders and early kernel initialization
- Kernel startup routines and user-mode transition
- Platform considerations for Intel and ARM systems

Kernel Programming Fundamentals

- Best practices for kernel development
- Writing portable kernel modules
- Kernel data structures and linked lists
- Synchronization primitives such as spinlocks and semaphores
- Kernel messaging and notification chains

Kernel Subsystems

- System call mechanisms

BİLGİNÇ IT Academy

info@bilginc.com

bilginc.com

- Interrupt handling concepts
- Process and thread management
- Scheduling policies and run queues

Device Driver Development

- Fundamentals of character device drivers
- Major and minor device numbers
- Device registration
- udev integration and hotplug support
- sysfs interfaces for device management

Advanced Kernel Topics

- Kernel timekeeping and timers
- Wait queues and scheduling interactions
- Virtual memory architecture
- Dynamic memory allocation using kmalloc and kfree
- Slab and buddy allocators

Interrupt Handling

- Kernel interrupt handling
- IRQ management
- Writing safe interrupt handlers
- Bottom halves, tasklets, and work queues

Linux Networking Architecture

- Linux networking stack overview
- Network device structures
- Ethernet basics
- Netfilter and packet filtering

Linux Kernel Security

- Linux capabilities model

Linux Kernel Programming Training

Duration: **5 Day**

BİLGİNÇ
IT Academy
info@bilginc.com
bilginc.com

- Kernel security risks and attack surfaces
- Linux Security Modules (LSM)
- Overview of SELinux