For quote and registration: email ron@thepscg.com or call +972-54-5529466 (Course #: LFD440)

Linux Kernel Debugging and Security

This course provides experienced programmers with a solid understanding of Linux kernel. debugging techniques and tools. This four day course includes extensive hands-on exercises and demonstrations designed to give you the necessary tools to develop and debug Linux kernel code.

Length: 4 Days
Type: Hands-On
Target Audience: Linux Kernel developers, security researchers, board designers, system administrators, Real-Time/embedded engineers making the transition to Linux, device drivers developers.

Prerequisites:

- Essential:
  - The attendees should proficient with the C language
  - The attendees should have basic knowledge kernel interfaces and methods
  - The attendees should have experience in building a kernel and kernel modules
- Recommended:
  - Working knowledge of Linux command line tools
  - Previous Device Driver or Embedded Linux experience
  - Previous training of *LFD420*, *Ron Munitz's Kernel Bootcamp*, or equivalent

Summary:
Linux Kernel Debugging and Security course provides experienced programmers with a solid understanding of the *Linux* kernel security features, as well as how to monitor, profile, trace and debug the kernel.

Outline:

1. Introduction
   - Objectives
   - Who You Are
   - The Linux Foundation
   - Linux Foundation Training
   - Certification Programs and Digital Badging
   - Linux Distributions
   - Platforms
   - Preparing Your System
   - Using and Downloading a Virtual Machine
   - Things change in Linux

- Documentation and Links
- Course Registration

2. Preliminaries
   - Procedures
   - Kernel Versions
   - Kernel Sources and Use of git
3. How to Work in OSS Projects **
   - Overview on How to Contribute Properly
   - Stay Close to Mainline for Security and Quality
   - Study and Understand the Project DNA
   - Figure Out What Itch You Want to Scratch
   - Identify Maintainers and Their Work Flows and Methods
   - Get Early Input and Work in the Open
   - Contribute Incremental Bits, Not Large Code Dumps
   - Leave Your Ego at the Door: Don't Be Thin-Skinned
   - Be Patient, Develop Long Term Relationships, Be Helpful
4. Kernel Features
   - Components of the Kernel
   - User-Space vs. Kernel-Space
   - What are System Calls?
   - Available System Calls
   - Scheduling Algorithms and Task Structures
   - Process Context
   - Labs
5. Monitoring and Debugging
   - Debuginfo Packages
   - Tracing and Profiling
   - sysctl
   - SysRq Key
   - oops Messages
   - Kernel Debuggers
   - debugfs
   - Labs
6. The proc Filesystem **
   - What is the proc Filesystem?
   - Creating and Removing Entries
   - Reading and Writing Entries
   - The seq_file Interface **
   - Labs
7. kprobes
   - kprobes
   - kretprobes
   - SystemTap **

- Labs

8. Ftrace
   - What is ftrace?
   - ftrace, trace-cmd and kernelshark
   - Available Tracers
   - Using ftrace
   - Files in the Tracing Directory
   - Tracing Options
   - Printing with trace_printk()
   - Trace Markers
   - Dumping the Buffer
   - trace-cmd
   - Labs
9. Perf
   - What is perf?
   - perf stat
   - perf list
   - perf record
   - perf report
   - perf annotate
   - perf top
   - Labs
10. Crash
    - Crash
    - Main Commands
    - Labs
11. Kernel Core Dumps
    - Generating Kernel Core Dumps
    - kexec
    - Setting Up Kernel Core Dumps
    - Labs
12. Virtualization**
    - What is Virtualization?
    - Rings of Virtualization
    - Hypervisors
13. QEMU
    - What is QEMU?
    - Emulated Architectures
    - Image Formats
    - Third Party Hypervisor Integration
14. Linux Kernel Debugging Tools
    - Linux Kernel (built-in) tools and helpers
    - kdb

- qemu+gdb
- kgdb: hardware+serial+gdb
- Labs

15. Embedded Linux**
    - Embedded and Real Time Operating Systems
    - Why Use Linux?
    - Making a Small Linux Environment
    - Real Time Linuxes
16. Notifiers**
    - What are Notifiers?
    - Data Structures
    - Callbacks and Notifications
    - Creating Notifier Chains
    - Labs
17. CPU Frequency Scaling**
    - What is Frequency and Voltage Scaling?
    - Notifiers
    - Drivers
    - Governors
    - Labs
18. Netlink Sockets**
    - What are netlink Sockets?
    - Opening a netlink Socket
    - netlink Messages
    - Labs
19. Introduction to Linux Kernel Security
    - Linux Kernel Security Basics
    - Discretionary Access Control (DAC)
    - POSIX ACLs
    - POSIX Capabilities
    - Namespaces
    - Linux Security Modules (LSM)
    - Netfilter
    - Cryptographic Methods
    - The Kernel Self Protection Project
20. Linux Security Modules (LSM)
    - What are Linux Security Modules?
    - LSM Basics
    - LSM Choices
    - How LSM Works
    - An LSM Example: Tomoyo
21. SELinux
    - SELinux

- SELinux Overview
- SELinux Modes
- SELinux Policies
- Context Utilities
- SELinux and Standard Command Line Tools
- SELinux Context Inheritance and Preservation**
- restorecon**
- semanage fcontext**
- Using SELinux Booleans**
- getsebool and setsebool**
- Troubleshooting Tools
- Labs

22. AppArmor
    - What is AppArmor?
    - Checking Status
    - Modes and Profiles
    - Profiles
    - Utilities
23. Netfilter
    - What is netfilter?
    - Netfilter Hooks
    - Netfilter Implementation
    - Hooking into Netfilter
    - Iptables
    - Labs
24. The Virtual File System
    - What is the Virtual File System?
    - Available Filesystems
    - Special Filesystems
    - The tmpfs Filesystem
    - The ext2/ext3 Filesystem
    - The ext4 Filesystem
    - The btrfs Filesystem
    - Common File Model
    - VFS System Calls
    - Files and Processes
    - Mounting Filesystems
25. Filesystems in User-Space (FUSE)**
    - What is FUSE?
    - Writing a Filesystem
    - Labs
26. Journaling Filesystems**
    - What are Journaling Filesystems?

- Available Journaling Filesystems
- Contrasting Features
- Labs

27. Closing and Evaluation Survey
    - Evaluation Survey

** These sections may be considered in part or in whole as optional. They contain either background reference material, specialized topics, or advanced subjects. The instructor may choose to cover or not cover them depending on classroom experience and time constraints.