Linux Kernel Debugging and Security (LFD440)

Modality: Virtual Classroom

Duration: 4 Days

SATV Value:

CLC:

NATU:

SUBSCRIPTION: No

About this course:

Learn the methods and internal infrastructure of the Linux kernel. This course focuses on the important tools used for debugging and monitoring the kernel, and how security features are implemented and controlled.

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.

The average salary of an Embedded Linux Developer is **$107,500** per year.

Audience:

- App Developers
- C/C++,C# developers
- Linux Developers

Prerequisite:

Before taking this course, you should:

- Be proficient in the C programming language.
- Be familiar with basic Linux (UNIX) utilities such as ls, grep and tar.
- Be comfortable using any of the available text editors (e.g. emacs, vi, etc.).
- Experience with any major Linux distribution is helpful but not strictly required.
- Have experience equivalent to having taken LFD420, the kernel internals course.

Course Outline:

Introduction

- Objectives
- Who You Are
- The Linux Foundation

- Linux Foundation Training
- Linux Distributions
- Platforms
- Preparing Your System
- Using and Downloading a Virtual Machine
- Things change in Linux
- Documentation and Links
- Course Registration

Preliminaries

- Procedures
- Kernel Versions
- Kernel Sources and Use of git

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

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

Monitoring and Debugging

- Debuginfo Packages
- Tracing and Profiling
- sysctl
- SysRq Key
- oops Messages
- Kernel Debuggers
- debugfs
- Labs

The proc Filesystem **

- What is the proc Filesystem?
- Creating and Removing Entries
- Reading and Writing Entries
- The seq_file Interface **
- Labs

kprobes

- kprobes
- kretprobes
- SystemTap **
- Labs

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

Perf

- What is perf?
- perf stat
- perf list
- perf record
- perf report
- perf annotate
- perf top
- Labs

Crash

- Crash
- Main Commands
- Labs

Kernel Core Dumps

- Generating Kernel Core Dumps
- kexec
- Setting Up Kernel Core Dumps
- Labs

Virtualization

- What is Virtualization?
- Rings of Virtualization
- Hypervisors

QEMU

- What is QEMU?
- Emulated Architectures
- Image Formats
- Third Party Hypervisor Integration

Linux Kernel Debugging Tools

- Linux Kernel (built-in) tools and helpers
- kdb
- qemu+gdb
- kgdb: hardware+serial+gdb
- Labs

Embedded Linux

- Embedded and Real Time Operating Systems
- Why Use Linux?
- Making a Small Linux Environment
- Real Time Linuxes

Notifiers

- What are Notifiers?
- Data Structures
- Callbacks and Notifications
- Creating Notifier Chains
- Labs

CPU Frequency Scaling

- What is Frequency and Voltage Scaling?
- Notifiers
- Drivers
- Governors
- Labs

Netlink Sockets**

- What are netlink Sockets?
- Opening a netlink Socket
- netlink Messages
- Labs

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

Linux Security Modules (LSM)

- What are Linux Security Modules?
- LSM Basics
- LSM Choices
- How LSM Works
- An LSM Example: Tomoyo

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

AppArmor

- What is AppArmor?
- Checking Status
- Modes and Profiles

- Profiles
- Utilities

Netfilter

- What is netfilter?
- Netfilter Hooks
- Netfilter Implementation
- Hooking into Netfilter
- Iptables
- Labs

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

Filesystems in User-Space (FUSE)**

- What is FUSE?
- Writing a Filesystem
- Labs

Journaling Filesystems**

- What are Journaling Filesystems?
- Available Journaling Filesystems
- Contrasting Features
- Labs

Closing and 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.**