eBPF Kernel Programming Learning Plan

1. Week 1-2: Mastering C and Linux System Programming

Week 1: Mastering C Programming Basics

Day 1-2: Refresh C Language Basics

- Data Types and Variables
- Control Flow: conditionals (if, switch), loops (for, while)
- Functions: pass arguments by value/reference, recursion
- Pointers and Memory: malloc(), free(), pointer arithmetic

Day 3-4: Focus on Advanced C Concepts

- Structures and Unions
- Memory Management and Pointers
- Dynamic Data Structures: linked lists

Day 5-7: Low-Level and System-Specific C

- File I/O: fopen(), fread(), fwrite()
- Bitwise Operations: &, |, ^, <<, >>
- Preprocessor Directives: #define, #include

Week 2: Linux System Programming Essentials

Day 1-2: Introduction to System Calls

- System Calls: open(), read(), write(), close()
- Error Handling: errno, perror(), strerror()

- File Descriptors: dup(), dup2()

Day 3-4: Process Management

- Process Creation: fork(), exec()

- Process Synchronization: wait(), waitpid()

- Signals: signal(), sigaction()

Day 5-6: File I/O in Linux

- Advanced File Operations: stat(), fstat(), lstat()

- Directory Handling: opendir(), readdir(), closedir()

- File Permissions and Locking: chmod(), chown(), fcntl()

Day 7: Introduction to Sockets

- Socket Basics: socket(), bind(), listen(), accept()

- Basic TCP and UDP programming: send(), recv()

2. Week 3-4: eBPF Basics and Initial Programs

Week 3: Introduction to eBPF

Day 1-2: Overview of eBPF

- What is eBPF? Use cases in tracing, networking, and security.

- Understand eBPF programs, hooks, and maps.

- Explore basic eBPF helpers like bpf_trace_printk().

Day 3-4: Working with eBPF Maps

- eBPF maps as key-value stores in the kernel.

- Map types: arrays, hashmaps, and per-CPU arrays.

- Accessing maps from user-space and kernel-space.

Day 5-7: Loading and Verifying eBPF Programs

- Using bpftool to load eBPF programs into the kernel.

- eBPF verifier: safety and security checks.

- Practice writing and loading simple eBPF programs.

Week 4: Writing Practical eBPF Programs

Day 1-2: eBPF Tracepoints and kprobes

- Tracing function calls using tracepoints and kprobes.

- Practice tracing system calls, network functions, etc.

- Collecting data using eBPF maps.

Day 3-4: User-Space Interaction with eBPF

- Interacting with eBPF programs from user-space (using libbpf).
- Reading eBPF map data in user-space.
- Basic usage of bcc (BPF Compiler Collection).

Day 5-7: XDP - eXpress Data Path

- Introduction to XDP: high-performance packet processing.
- Writing XDP programs to drop or filter network packets.
- Practice deploying XDP programs.

3. Week 5-6: Advanced eBPF Programming

Week 5: Advanced eBPF Techniques

Day 1-3: Working with BPF Type Format (BTF)

- Introduction to BTF: Debugging information for eBPF programs.
- Working with BTF-enabled kernels.
- Using BTF for enhanced program portability.

Day 4-5: Perf Events and Perf Buffer

- Using perf events for performance monitoring.
- Capturing and analyzing performance data using eBPF.
- Working with the perf buffer for efficient data transfer.

Day 6-7: eBPF and Network Traffic Control (TC)

- Introduction to Traffic Control (TC) with eBPF.
- Writing TC programs to manipulate network traffic.
- Deploying TC programs for monitoring and shaping traffic.

Week 6: Building eBPF Applications

Day 1-2: Seccomp and Security Filters

- Using eBPF with seccomp to filter system calls.
- Writing security filters for process isolation and sandboxing.

Day 3-4: CO-RE (Compile Once, Run Everywhere)

- Introduction to CO-RE: Making eBPF programs portable.

- Using libbpf and CO-RE to write kernel-version-independent programs.

Day 5-7: Final Project

- Build a complete eBPF-based monitoring tool.

- Deploy the tool to capture real-time data (e.g., system calls, network traffic).