oSEC9 - Advanced Embedded Linux Security

Objectives

- Learn the basics of embedded Linux security
- Understand Linux threat model
- Discover the features in Linux kernel to harden security
- Understand Linux Security Modules
- Learn how sandboxing can harden your system’s security

Labs are conducted on QEMU ARM-based board

Prerequisite

- C Language knowledge (see for example our L2 training course)
- Secured Embedded Linux Platform Build (see for example our D11 training course)
- You may be interested also by the SEC8 Secured Embedded Linux Platform Build course
- You may be interested also by the SEC1 Secure Development for Embedded System course
- You may be interested also by the SEC2 Advanced Embedded Systems Security course

Equipment

- Training manuals and software exercises
- One Linux PC for two trainees
- One target platform for two trainees

Duration

- Total: 3 days
- From 40% to 50% of training time is devoted to practical activities

Course Environment

- Theoretical course
  - PDF course material (in English) supplemented by a printed version.
  - The trainer answers trainees' questions during the training and provide technical and pedagogical assistance.
- At the start of each session the trainer will interact with the trainees to ensure the course fits their expectations and correct if needed

Target Audience

- Any embedded systems engineer or technician with the above prerequisites.

Course Outline

First Day

Defining the threat model for embedded Linux

- Potential security risks to an embedded system
- Threat model for embedded Linux
  - Identifying Assets and Threats
  - Understanding Attack Vectors
  - Identifying Security Weaknesses and Risks
  - Analyzing Threats and Evaluating Impact
  - Countermeasures and Threat Mitigation
- Reducing Attack Surface
- Common Linux Vulnerabilities
- Vulnerable Linux tools
- Check for known vulnerabilities

Basic security features in Linux

- User and Group Management
- File Permissions and Ownership
  - Restrict access to sensitive information
  - Limit public access to system files
- Adjusting Systems Services
- Input Validation and Improper Input Handling
  - Overview of Input Validation and Its Importance
  - Input Validation Techniques
  - Preventing and Mitigating Input-Related Attacks
- Stack buffer overflow
  - Understanding the impact and techniques for mitigating
  - Enabling stack protection mechanisms in the Linux kernel
  - Address Space Layout Randomization (ASLR)
  - Preventing Stack-based Attacks through code review
- Privilege Escalation
  - Privilege Escalation Attack Vectors
  - Horizontal and Vertical Privilege Escalation
  - Exploiting SUID executables
  - Escalating privileges through misconfigured services
  - Multi-User Escalation
  - Buffer overflow attacks
  - Mitigating privilege escalation attacks
  - Best practices for preventing privilege escalation

Network Hardening

- Network Security Overview
- Securing SSH
- Encrypting network traffic
- Using SSL/TLS certificates
- Virtual Private Network (VPN)
- Wireless Network Security
- Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS)
- Firewall on Linux

    - Types of firewall available in Linux
    - Configuring firewall using iptables, firewalld or nftables

Second Day

Advanced security features in Linux

- Restricting System Calls in Linux
    - Introduction to system call restrictions
    - Understanding the purpose and benefits of restricting system calls
    - How to use seccomp to restrict system calls in Linux
    - Analyzing the impact of system call restrictions on application functionality
    - Seccomp limitations
    - Best practices for creating a system call whitelist
    - Systemd system call filtering
- Enhancing Security with Capabilities
    - Overview of capabilities in Linux
    - Understanding the significance of privilege separation in Linux
    - The different types of capabilities
    - Capability Commands
    - File System Capabilities
    - Implementing file system capabilities
    - Protecting SUID executables
    - Enhancing the security of Daemons
    - Setting default capabilities for newly created processes
    - Case studies and real-world examples

Hardening the Linux Kernel

- Methods to harden the Linux Kernel
- Custom kernel configuration
- Kernel hardening options
- Kernel Self-Protection
- Disabling unnecessary services
- Limiting the available memory resources

Linux Security Modules (LSMs)

- Introduction to Linux Security Modules (LSMs)
    - Overview of LSMs and their purpose
    - Types of LSMs available in Linux
    - Understanding the Linux security model
- Access permissions
    - Discretionary Access Control (DAC)
    - Mandatory Access Control (MAC)
- Overview of the concepts, goals and principles of MAC security models
- MAC Models
- Implementation of MAC
    - access control lists (ACLs)
    - role-based access control (RBAC)
    - label-based access control (LBAC)
    - Managing MAC in a Multi-user Environment
    - DAC vs MAC

Security Enhanced Linux (SELinux)

- Overview of SELinux and its purpose

- Enable SELinux
- Architecture and Components
- SELinux Contexts and Labels
- Benefits of using SELinux
- SELinux policies
  - Understanding SELinux Policies
  - Creating and managing SELinux policies
  - SELinux policy structure and language
- Enforcing, Permissive, and Disabled Modes
- User, Role, and Type Components
- Defining Custom Domain Types
- SELinux Boolean Values
- SELinux Auditing and Logging
- Troubleshooting SELinux
- Advanced SELinux Configuration
  - Managing SELinux Port Contexts
  - Configuring SELinux for systemd Services
  - Managing SELinux for Containers

Third Day

Other Linux Security Modules

- AppArmor
  - Overview of AppArmor features and capabilities
  - Implementing AppArmor in Linux
  - Creating and managing AppArmor profiles
  - Understanding and using AppArmor rules
  - AppArmor vs SELinux: Choosing the right solution for your needs
- Simple Mandatory Access Control for Linux (SMACK)
  - Overview of SMACK and its purpose
  - Characteristics and features of SMACK
  - Configuration and Implementation of SMACK
  - Customizing SMACK policies
  - Combining SMACK with other security features
  - SMACK's strengths and weaknesses
- TOMOYO
  - Overview of TOMOYO and its purpose
  - The difference between TOMOYO and other LSMs
  - TOMOYO policies
- Yama
  - Explanation of Yama and its role in Linux security
  - Architecture of Yama and its interaction with other LSMs
  - Customizing the different rules and policies of Yama
- SafeSetID
  - Importance of SafeSetID in enhancing security in Linux
  - Setting up SafeSetID rules and policies
  - Limitations and Challenges

Validating Kernel Modules

- Overview of key components involved in Module Signing
- Key Concepts of Module Signing
- Types of Module Signing Methods
  - Discussion of the pros and cons of each method
- Module Signing steps
- Verifying the Signature of a Loaded Module

- Preventing malicious modules with LoadPin
- Steps for integrating LoadPin into the Linux environment

Application signing in Linux

- Signing packages for package managers
- Gnu Privacy Guard (GnuPG)
- Integrity Measurement Architecture (IMA)
- The Extended Verification Module (EVM)
- evmctl tool

Sandboxing

- Overview of Sandboxing and its Importance
- Understanding the Concept of Isolation and Resource Control
- Control Groups (cgroups)
- Chroot and its Security Benefits
- Containerization with LXC (Linux Containers)
  - Securing Application and Daemons with LXC
- Docker and its Security Features
- Exploring Namespaces in Linux
- Firejail overview

Testing, Logging and Auditing

- Scanning the Linux system
  - Scanning for known malware
- Linux auditing and monitoring tools
- Reviewing Logs for Suspicious Activity
- Retention policies and archiving logs
- Keeping logs secure and protected against tampering or deletion