SEC12 - Comprehensive Secure Systems Programming

Objectives

- Learn how to verify programs are in a secure state on startup and when calling out to other program
- Become familiar with MISRA C guidelines for the use of the C language in critical systems
- Lean ways to use C/C++ safely in critical systems
- Learn how to interpret the output of the MISRA C 2012 checking tool
- how to manipulate files and directories in a secure manner
- Discover how to protect your programs from malicious user input
- How to secure communication with TLS
- Embedded system hardware features for security
- Secure Software Development methodology and framework
- Secure System Software Consideration
- Apprehend the context and the use of Hypervisors and System Virtualization
- Discover Security checks and Tools

Prerequisites

- Some programming concepts are desirable (whatever language)

Course environment

- Theoretical course
  - PDF course material (in English)
  - Course dispensed using the Teams video-conferencing system
  - The trainer to answer trainees' questions during the training and provide technical and pedagogical assistance through the Teams video-conferencing system
- Practical activities
  - Practical activities represent from 40% to 50% of course duration
  - One Online Linux PC per trainee for the practical activities
  - The trainer has access to trainees' Online PCs for technical and pedagogical assistance
- Downloadable preconfigured virtual machine for post-course practical activities

Duration

- Total: 30 hours
- 5 sessions, 6 hours each
- From 40% to 50% of training time is devoted to practical activities
- Some Labs may be completed between sessions and are checked by the trainer on the next session

Target Audience

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

Course Outline

First Session

Introduction to embedded security

- Embedded Security Trends
  - Embedded Systems Complexity
  - Network connectivity
  - Reliance on Embedded Systems for Critical Infrastructure
  - Processor consolidation
- Security policies
  - Perfect Security
  - Confidentiality, Integrity, and Availability
  - Isolation
  - Information Flow Control
  - Physical Security Policies
  - Application-Specific Policies
- Security Threats

Writing Secure C/C++ Code

- Safe use of pointers
- Memory allocation and corruption
- Buffer overflow
- Return Oriented Programming
- Core embedded Operating system Security Requirements
- String and format functions
- Integer security
- Concurrency
- File I/O

Exercise: Memory Overflow Attacks

Second Session

Secure Coding

- Coding Standards
- Case Study: MISRA C:2012 and MISRA C++:2008
- Embedded C++
- Complexity Control
- Static Source Code Analysis
- Creating a Tailored Organizational Embedded Coding Standard
- Dynamic Code Analysis

Exercise: Use of static analysis tools

Cryptography Overview

- Cryptographic Modes
- Block Ciphers
- Authenticated Encryption
- Public Key Cryptography
- Key Agreement
- Public Key Authentication

- Elliptic Curve Cryptography
- Cryptographic Hashes
- Message Authentication Codes
- Random Number Generation
- Key Management for Embedded Systems

Exercise: Memory Overflow Attacks

Third Session

Transport Layer Security

- Secure communications
- Authentication
- IoT Protocols
- MQTT
- DTLS
- HTTPS
- CoAP
- TLS Implementation
- Wireless LAN Security and Threats

Exercise: Installing and using certificates
Exercise: Sending secure messages with TLS

Secure Embedded System Software Architecture

- Secure software architecture goals
- Least privilege, trust and secure processes
- Arm Platform Security Architecture (PSA)

Secure Embedded System Hardware Architecture

- Crypto-Accelerator Overview
- Arm TrustZone
- Secure boot and update
- Hardware options for security

Fourth Session

System Software Consideration

- The Operating System
- Multiple Independent Levels of Security
  - Information Flow
  - Data Isolation
  - Damage Limitation
  - Periods Processing
  - Tamper Proof
  - Evaluable
- Core embedded Operating system Security Requirements
  - Memory Protection
  - Virtual Memory
- Guard Pages
- Location obfuscation
  - Fault Recovery
  - Impact of Determinism
  - Secure Scheduling
- Hypervisors and System Virtualization

    - Introduction to System Virtualization
    - Applications of System Virtualization
    - Environment Sandboxing
    - Virtual Security Appliances
- Hypervisor Architectures
- Paravirtualization
- Leveraging Hardware Assists for Virtualization
    - ARM TrustZone
- Hypervisor Security
- I/O Virtualization
- Remote Management
- Assuring Integrity of the TCB
    - Trusted Hardware and Supply Chain
    - Secure Boot
    - Static versus Dynamic Root of Trust
    - Remote Attestation

Exercise: Memory Protection (MPU)
Exercise: ARM TrustZone
Exercise: Secure Boot

Fifth Session

Data Protection Protocols for Embedded Systems

- Data-in-Motion Protocols
    - Generalized Model
    - Choosing the Network Layer for Security
    - Ethernet Security Protocols
    - IPsec versus SSL
    - IPsec
    - SSL/TLS
    - Embedded VPN Clients
    - DTLS
    - SSH
    - Custom Network Security Protocols
    - Secure Multimedia Protocols
    - Broadcast Security
- Data-at-Rest Protocols
    - Choosing the Storage Layer for Security
    - Symmetric Encryption Algorithm Selection
    - Managing the Storage Encryption Key

Testing for Security

- Basic Testing Methods
    - White-Box Testing
    - Black-Box Testing
    - Grey-Box Testing
- Fuzz-Testing