



## SEC1 - Developing C/C++ Secure Embedded Systems

### Objectives

- Introduction to embedded security and industry standards.
- Learn about secure coding practices for C/C++ programming languages.
- Learn about secure software development methodologies.
- Introduce techniques for ensuring security (Security Testing) in embedded systems
- Introduce cryptography in embedded system.
- Learn about secure communication in embedded systems.

### 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: 18 hours
- 3 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
  - Sophisticated Attacks
  - Processor Consolidation
- Security Policies
  - Perfect Security ?

- Embedded Security Challenges
- Confidentiality, Integrity and Availability
- Isolation
- Information Flow Control
- Physical Security Policies
- Security Threats
  - Summary of issues
  - Cyberattack exploits
- Legacy Systems
  - Updatability
  - Securing Legacy Systems
  - Project Requirements
  - Performance ?
- Security standards
- IoT recommended Security standards

## Secure C/C++ Code

- Secure C
  - Preprocessor and macros
  - Compilation, Declaration, definition, and initialization
  - Types
  - Pointers and arrays
  - Structure and unions
  - Expressions
  - Conditional and iterative structures
  - Functions
  - Memory Management
  - Error handling
  - Standard Libraries
- Secure C++
  - Declarations and Initialization
  - Expressions
  - Integers
  - Containers
  - Characters and Strings
  - Memory Management
  - Input Output
  - Exceptions and Error Handling
  - Object Oriented Programming
  - Concurrency
  - Miscellaneous

**Exercise:** Debugging memory problems

## Security in RUST (Optional)

- Development environment
- Libraries
- Language generalities
- Memory management
- Type system
- Foreign function interface (FFI)
- Recommendations

## Second Session

### Secure Software Development

- Threat modelling
  - Introduction to threat modeling
  - Example threat models
- Risk analysis
- Software Assurance Maturity Model (SAMM)
- Platform Security architecture (PSA)
- Frameworks and Standards
- Security Knowledge Framework and Certifications

### Ensuring security in Embedded Systems

- Introduction
- Security Testing
  - Penetration testing
  - Vulnerability scanning
  - Risk assessment
  - Static Analysis
  - Dynamic analysis
  - Protocol fuzzing
- Security provisioning
  - Security configuration management
  - Identity and access management
  - Incident response and management
  - Compliance and regulatory requirements
- Security Testing Tools overview

### Cryptography introduction

- Overview of cryptography
- Classic Cryptography
- Information assurance
- Symmetric encryption
- Asymmetric encryption
- Random number generation
- Integrity and authentication
- Access authentication
- Elliptic Curve cryptography
- Certificates and Public Key infrastructures
- Rules and recommendations

**Exercise:** Encryption/Decryption

**Exercise:** Private/Public Keys

**Exercise:** Authentication and Integrity on IoT Devices

## Third Session

### Secure Embedded System Hardware Architecture

- Crypto-Accelerator Overview
- ARM TrustZone
- Intel Software Guard eXtensions
- SoC Security overview
  - Memory Protection

- Trusted Boot and Firmware update overview
- Secure Elements
- Trusted Platform Module (TPM)
- Hardware Security Module (HSM)

**Exercise:** Secure boot

**Exercise:** ARM TrustZone application (secure/non secure)

## Overview of Secure Communication in embedded Systems

- Introduction
- Transport Layer Security (TLS)
- IPsec/IKE
- Network layer

## IoT security

- Secured IoT architecture
- IoT standard and recommendations
- Software development architecture and practices
- Cryptology
- Software security
- Hardware protection
- Network security
- Life cycle and support