oSEC1 - Secure C/C++ Development for Embedded Systems

Objectives

- Introduction to embedded security and industry standards, including ISO/SAE 21434, IEC 62443, NIST SP 800-53, Common Criteria, and OWASP.
- Learn about secure coding practices for C/C++ programming languages, including best practices for memory management, input validation, and error handling.
- Introduce the RUST programming language and its built-in security features, including memory safety and type safety.
- Learn about secure software development methodologies, including threat modeling, secure design principles, and secure coding standards.
- Introduce techniques for ensuring security in embedded systems, including security testing, security provisioning, and secure boot processes.
- Introduce cryptography in embedded system.
- The course covers the design and implementation of secure embedded system hardware architecture, including secure boot processes and secure communication protocols.
- Learn about secure communication in embedded systems, including network protocols, secure communication protocols, and secure data transfer.
- Get an overview of security issues and best practices for Internet of Things (IoT) devices and 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 answers 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.
  - Code examples, exercises and solutions
  - One Online Linux PC per trainee for the practical activities.
  - The trainer has access to trainees' Online PCs for technical and pedagogical assistance.
  - Eclipse environment and GCC compiler.
  - QEMU Emulated board or physical board connected to the online PC (depending on the course).
  - Some Labs may be completed between sessions and are checked by the trainer on the next session.
- Downloadable preconfigured virtual machine for post-course practical activities
- At the start of each session the trainer will interact with the trainees to ensure the course fits their expectations and correct if needed

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.

Evaluation modalities

- The prerequisites indicated above are assessed before the training by the technical supervision of the traineein his company, or by the trainee himself in the exceptional case of an individual trainee.
- Trainee progress is assessed in two different ways, depending on the course:
  - For courses lending themselves to practical exercises, the results of the exercises are checked by the trainer while, if necessary, helping trainees to carry them out by providing additional details.
  - Quizzes are offered at the end of sections that do not include practical exercises to verifythat the trainees have assimilated the points presented
- At the end of the training, each trainee receives a certificate attesting that they have successfully completed the course.
  - In the event of a problem, discovered during the course, due to a lack of prerequisites by the trainee a different or additional training is offered to them, generally to reinforce their prerequisites,in agreement with their company manager if applicable.

Plan

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

- 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
  - NIST SP 800-160: Developing Cyber-Resilient Systems
  - ISO/SAE 214341: Road vehicles & Cybersecurity engineering
  - ISO/IEC 15408: Security, cybersecurity and privacy protection
  - IEC 651508: Functional Safety of electrical/electronic/programmable electronic safety-related systems
  - UL 2900-2-2: Software cybersecurity for network-connectable products
- 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
    - Bluetooth
    - WiFi
    - 5G
    - NFC
    - RFID
    - SigFox

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

Renseignements pratiques

Inquiry : **18 hours**
Prochaines sessions : **from 28th to 30th of July, 2025 - Online EurAsia (9h-16h CET)**