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) supplemented by a printed version for face-to-face courses.
  - Online courses are dispensed using the Teams video-conferencing system.
  - The trainer answers trainees' questions during the training and provide technical and pedagogical assistance.
- Practical activities
  - Practical activities represent from 40% to 50% of course duration.
  - Code examples, exercises and solutions
  - For remote trainings:
    - One Online Linux PC per trainee for the practical activities.
    - The trainer has access to trainees' Online PCs for technical and pedagogical assistance.
    - 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.
  - For face-to-face trainings:
    - One PC (Linux ou Windows) for the practical activities with, if appropriate, a target board.
    - One PC for two trainees when there are more than 6 trainees.
  - For onsite trainings:
    - An installation and test manual is provided to allow preinstallation of the needed software.
    - The trainer come with target boards if needed during the practical activities (and bring them back at the end of the course).
- 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: 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.

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
  - 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

Renseignements pratiques

Inquiry : 30 hours