



## SEC1 - Développement C/C++ de systèmes embarqués sécurisés

### Objectifs

- Introduction à la sécurité intégrée et aux normes industrielles, notamment ISO/SAE 21434, IEC 62443, NIST SP 800-53, Common Criteria et OWASP.
- Apprendre les bonnes pratiques de codage sécurisé pour les langages de programmation C/C++, y compris les meilleures pratiques pour la gestion de la mémoire, la validation des entrées et la gestion des erreurs.
- Introduction au langage de programmation RUST et à ses fonctionnalités de sécurité intégrées, telles que la sécurité de la mémoire et la sécurité des types.
- Apprendre les méthodologies de développement de logiciels sécurisés, notamment la modélisation des menaces, les principes de conception sécurisés et les normes de codage sécurisé.
- Introduire des techniques pour garantir la sécurité dans les systèmes intégrés, notamment les tests de sécurité, la provision de sécurité et les processus de démarrage sécurisés.
- Introduction à la cryptographie dans les systèmes intégrés.
- Le cours couvre la conception et la mise en Œuvre de l'architecture matérielle sécurisée des systèmes intégrés, notamment les processus de démarrage sécurisés et les protocoles de communication sécurisés.
- Apprendre la communication sécurisée dans les systèmes intégrés, y compris les protocoles de réseau, les protocoles de communication sécurisés et le transfert de données sécurisé.
- Obtenir un aperçu des problèmes de sécurité et des meilleures pratiques pour les dispositifs et les systèmes IoT (Internet des objets).

### Pré-requis

- Quelques notions de programmation sont souhaitables (quel que soit le langage)

### Environnement du cours

- Cours théorique
  - Support de cours au format PDF (en anglais)
  - Cours dispensé via le système de visioconférence Teams
  - Le formateur répond aux questions des stagiaires en direct pendant la formation et fournit une assistance technique et pédagogique
- Activités pratiques
  - Les activités pratiques représentent de 40% à 50% de la durée du cours
  - Elles permettent de valider ou compléter les connaissances acquises pendant le cours théorique.
  - Un PC Linux en ligne par stagiaire pour les activités pratiques
  - Exemples de code, exercices et solutions
  - Le formateur a accès aux PC en ligne des stagiaires pour l'assistance technique et pédagogique
  - Certains travaux pratiques peuvent être réalisés entre les sessions et sont vérifiés par le formateur lors de la session suivante
- Une machine virtuelle préconfigurée téléchargeable pour les activités pratiques après le cours
- Au début de chaque demi-journée une période est réservée à une interaction avec les stagiaires pour s'assurer que le cours répond à leurs attentes et l'adapter si nécessaire

### Audience visée

- Tout ingénieur ou technicien en systèmes embarqués possédant les prérequis ci-dessus

### Durée

- Totale : 18 heures
- 3 sessions de 6 heures chacune
- De 40% à 50% du temps de formation est consacré aux activités pratiques

- Certains travaux pratiques peuvent être réalisés entre les sessions et sont vérifiés par le formateur lors de la session suivante

## Plan du cours

### Première 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

## Deuxième 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

## Troisième 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