A.L.S.E.
http://www.alse-fr.com

Formation Pratique
Embedded System Security for C/C++ Developers

DOULOS

Présentation Les microcontrôleurs embarqués sont désormais universellement utilisés dans la plupart des systèmes électroniques : des simples capteurs aux composants automobiles les plus sophistiqués en passant par les applications grand public.
Les problèmes de sécurité et vulnérabilités affectant les grands réseaux d'ordinateurs sous Windows ou Linux (incluant Linux embarqué) sont bien documentés et pris en compte, mais les petits systèmes à base de micro-contrôleurs le sont beaucoup moins. Pourtant, ces systèmes sont de plus en plus communicants, parfois sur des réseaux à très large échelle (comme Internet), ce qui conduit à des **vulnérabilités très importantes et largement mésestimées par leurs concepteurs**.

Objectifs Cette formation sensibilise de façon pratique et concrète aux vulnérabilités et problèmes de sécurités qui affectent les systèmes à base de micro-contrôleurs, et elle enseigne les techniques qui permet de les protéger. Comme la plupart de ces systèmes sont programmés en langage C/C++ , une partie du cours est dédiée aux méthodes de codage qui limitent les failles de sécurité. Les solutions alternatives de protection par des méthodes matérielles sont également passées en revue pour sécuriser le boot, les communications, et la mise à jour des applications embarquées.

Audience Cette formation est destinée aux ingénieurs Hardware, Software et System-On-Chip qui souhaitent prendre en compte les problèmes de sécurité affectant les systèmes embarqués. ***Important*** : ce cours ne traite pas particulièrement de la sécurité sous Linux embarqué car un autre cours (de 5 jours) est consacré à ce sujet à part entière.

Exercices Les exercices pratiques utilisent un kit spécifique qui permet la mise en œuvre de fonctions hardware et software de protection. Ces exercices occupent environ 50 % du temps de la formation, ils sont indispensables pour un réel apprentissage et ils sont des éléments clés du succès de ces formations.
Ils assurent également un contrôle continu de l'acquisition des compétences.

Instructeurs Les Instructeurs certifiés Doulos sont aussi et surtout des **Experts.** Ils savent partager leur savoir-faire avec passion et sont particulièrement appréciés des participants.

Supports de cours La qualité des supports de cours Doulos est réputée. Ils constituent avec les cahiers d'exercices de véritables ouvrages de référence utilisables à la suite de la formation.

Pré-requis La participation à cette formation demande une connaissance préalable du langage C (voire C++) et des architectures de systèmes embarqués. En particulier, un niveau de base de familiarité avec les fonctions, les variables, les types de données, les opérateurs et les instructions. D'autres formations Doulos sont disponibles sur ces sujets.

Durée **Trois (3) jours** soit **21 heures effectives** de cours ; Typiquement 9h00 → 17h30 avec une heure de pause pour déjeuner. Il est également possible, à distance

Matériel La formation à distance est dispensée via un outil spécialement conçu, qui permet une interaction entre les stagiaires et le formateur via vidéo, messages instantanés et partages d'écrans. Les exercices à distance sont effectués sur une ferme de kits télé-contrôlés pour une expérience optimale.

Il suffit de disposer d'un PC Windows ou Linux avec un écran confortable (ou deux), ainsi qu'un accès stable à Internet avec une bande passante correcte et si possible un micro+casque.

Conditions Un environnement calme et non perturbé est indispensable.

Objectifs pédagogiques

- Identifier les principales menaces et vulnérabilités des systèmes embarqués
- Comment utiliser les standards d'encryption pour protéger les données en transit et les données statiques.
- Gestion des clés et des certificats d'authentification
- Sécuriser les communications avec TLS
- Écrire du code C plus sûr.
- Standards de codage et outils d'analyse statique pour identifier les vulnérabilités du code C.
- Utiliser une Méthodologie de Développement et un environnement sécurisés.
- Les ressources Hardware dédiées à la sécurité des Systèmes embarqués
- Approches pour tester la sécurité des applications embarquées.

Embedded System Security for C/C++ Developers

Programme

Jour 1

Introduction à la Sécurité

• Pourquoi cette prise en compte est indispensable
• Quelles sont les vulnérabilités
• Le Cycle de développement du Logiciel Sécurisé

Écrire du code C/C++ sécurisé – Partie I - Vulnérabilités Mémoire et Attaques

• Safe use of pointers
• Memory allocation and corruption
• Buffer overflow
• Return Oriented Programming

Écrire du code C/C++ sécurisé – Partie II - Vulnérabilités & contre-mesures

• String and format functions
• Integer security
• Concurrency
• File I/O
Exercice pratique : Memory Overflow-based attacks

Secure Software Development Lifecycle

• Software design goals and threats
• Threat modelling
Exercice pratique : Creating a Threat Model

Jour 2

Cryptographie

• Encryption and Decryption
• Hashes
• Block encryption
• Block Cipher Modes
• AES
• Streaming Ciphers
• ChaCha20+Poly1305
• AEAD
Exercice pratique : Message encryption/decryption

La Cryptographies en Action

• Key management
• Signing
• Certificate and Certificate Agencies
• Pre-shared secrets
Exercice pratique : Installing and using certificates

Transport Layer Security

• Secure communications
• Random Number Generators
• Authentication
• IoT Protocols
• MQTT • DTLS • HTTPS • TLS Implementation
• Wireless LAN Security and Threats
Exercice pratique : Sending secure messages

Jour 3

Les Règles pour un Codage Sécurisé

• Common Criteria
• CWE and CVE
• The Role of Coding Standards • CERT C and MISRA-C
• Dynamic and Static Analysis
Exercice pratique : Use of static analysis tools

Secure Embedded System Software Architecture

• Secure software architecture goals
• Traditional guiding principles
• Side channel & timing attacks
• Least privilege, trust and secure processes
• Arm Platform Security Architecture (PSA)
Exercice pratique : Side-channel timing attack

Secure Embedded System Hardware Architecture

• Security in hardware • Crypto-Accelerator Overview
• Arm TrustZone • Secure boot and update
• Hardware options for security

Les Tests de Sécurité

• Unit tests and frameworks • Tools
• Penetration Testing • Protocol Fuzzing
• Disassembly • Simple Power Analysis (SPA)
• Differential Power Analysis (DPA)