Embedded Systems Security for Software Engineers

About this training:

Is your Embedded System secured? Embedded devices have traditionally run in relative isolation and have therefore been protected from a wide range of security threats. Today's devices, however, are often connected to corporate networks, public clouds, or the Internet directly. Medical Devices, IoT Devices, Automotive Systems and Industrial Control System have been increasingly become target for attackers.

The design of security for Embedded Systems is a critical and difficult task!

Embedded Systems Security for software engineers is a 5-day training covering the know-how of building security and trust into Embedded Systems and devices.

Through lectures and practical hands-on labs, students are guided through the design and implementation of secured Embedded Systems, including preventing and mitigating embedded systems attacks using state of the art and recent innovation security technologies that are available today. All hands-on labs are done using NXP's LPC55S69 Secure Microcontroller.

Intended Audience:

This training is suited for beginners as well as experienced Embedded systems engineers without deep security background.

Duration: 5 Days (Theory and Hands-on)

Goals

1. Understand the need for embedded systems security
2. Understand the threat and vulnerability landscape of Embedded Systems
3. Understand the challenges of designing secure Embedded Systems
4. Learn the concepts of designing secure Embedded Systems
5. Master the fundamental building blocks of Embedded Security
6. Get familiar with Embedded Systems Security best practices

Extensive hands-on labs for all topics, demonstrating the attacks as well as the countermeasures within the secure flow to make your system robust and protected:

- Stack buffer overflow attack
- Defenses against Stack Buffer Overflow attacks
- Reverse engineering of a binary file to find stack buffer overflow vulnerability
- Cryptography in Practice (AES, SHA256, HMAC, RSA, TRNG)
- Using PUF to implement a secure Keys-store
- Hands on Arm's TrustZone

Attendees will use the LPC55S69-EVK development board (LPC55S69Xpresso) as well as:

- Virtual Machine with Pre-installed IDE and Tools
- A digital copy of all lecture slides
- A Lab Manual with instructions for all hands-on labs
- Source code starting points for the labs
- Datasheets and User's Manuals for all of the hardware and software tools

Perquisites:

- Basic C/C++ Programming
- Embedded systems programming basic knowledge
- Linux command line (nice to have)

System Requirements:

4 Cores CPU (Intel i5 or Ryzen 5 CPU minimum)

8GB RAM

20GB available disk space

Windows 10 64-bit

One available USB port

Table of Contents

Day 1 – Introduction to Embedded Systems Security

- Device tampering detection
- Security monitoring
- Secure OTA updates

Day 2 – Cryptography

- **Cryptography 101**
  - The goal of cryptography
  - Cryptography in Embedded Systems
  - Cipher algorithms leakage
  - Randomness
  - Use of encryption
  - Cryptographic Hash functions
  - SH2 and SHA3
  - Hash Based Message Authentication Codes (HMAC)
  - Symmetric key cryptography
  - Types of symmetric key algorithms
  - AES encryption/decryption and modes
  - Asymmetric key cryptography
  - Types of asymmetric key algorithms
  - Asymmetric vs symmetric cryptography
  - What is Key Derivation Function?
  - Types of KDF
  - Key Exchange Algorithms
  - The RSA algorithm
  - Elliptic Curve Diffie Helman
  - Digital signature
  - Digital certificates
  - Using certificates against MITM attacks
  - Encryption key management
  - Post Quantum Cryptography (PQC)
  - PQC algorithms
  - ❖ **Lab #4: Calculating SHA256**
  - ❖ **Lab #5: Calculating HMAC**
  - ❖ **Lab #6: AES-128 Encryption & Decryption**
  - ❖ **Lab #7: Key Derivation Function**
  - ❖ **Lab #8: Generating RSA Private & Public key**
  - ❖ **Lab #9: ECDH key exchange**

Day 3 – Embedded Systems Hardware Security

- **Introduction to Embedded Systems Hardware Security**
  - The hardware is secure myth
  - What is hardware security?
  - Can hardware be trusted?
  - Hardware security vs hardware trust
  - Hardware vulnerabilities
  - Hardware attack vectors
  - Hardware attack surface

- **Hardware Security Primitives**
  - Hardware RoT typical components
  - What is hardware security perimeter?
  - PUF (Physically Unclonable Function) definition
  - Applications of PUF & classes
  - Using PUF for key generation
  - Using PUF for device authentication
  - SRAM-PUF
  - Hardware based TRNG
  - TRNG vs PRNG
  - Common TRNG architectures
  - Secure RTC
  - Power domain isolation
  - Monotonic counter
  - HSM (Hardware Security Module) typical architecture
  - When HSM is useful?
  - HSM embedded system use cases
  - TMP (Trusted Platform Module) typical architecture
  - Types of TPM
  - When TPM is useful?
  - Mitigation against TPM MITM attacks
  - HSM vs TPM
  - Limitation of HSM/TPM
  - Secure element
  - Secure element vs secure enclave vs TPM
  - Beyond TPM
  - Trusted Execution Environment (TEE)
  - Choosing the best solution
  - Secure JTAG controller
  - Debug authentication
  - Secure JTAG on PCB
  - Detecting tampers external to the system

- ❖ **Lab #10: SRAM-PUF key management – Under the Hood**
- ❖ **Lab #11: Hardware Random Generator**

- **Memory Protection**
  - Sources & types of attacks on memory
  - Known attacks on memories
  - Memory protection of MCU/MPU/FPGA
  - Confidentiality and integrity protections
  - Memory access pattern protections
  - Security of non-volatile memories

Day 4 – Embedded Systems Software Security

- **Embedded Software Vulnerabilities**
  - Embedded Systems software security introduction
  - Common vulnerabilities and exposures
  - Top 10 Embedded Systems programming errors

- **Common Attacks Against Embedded Software**
  - Application & OS based attacks
  - Network based attacks
  - Attacks on Firmware software
  - Firmware attack surface
  - Firmware attack vectors
  - Firmware attack techniques

- **Attacks Against the Core**
  - Attacks against the memory
  - Memory attack modes
  - Attacks against CPUs
  - Meltdown & Spectre
  - Denial of Service attack
  - Attacks on IPC
  - Attacks on time management
  - Attacks on process management
  - Attacks on scheduling

  - ❖ **Lab #12: Stack Buffer Overflow**
  - ❖ **Lab #13: Reverse Engineering 2**
  - ❖ **Lab #14: Stack Canaries**
  - ❖ **Lab #15: Vulnerable Serial**

Day 5 – Embedded Systems Software Security

- **Defensive Architectures**
  - The need for secure processor architecture
  - Traditional processor architectures
  - Secure processor architectures
  - Trusted Execution Environment (TEE)
  - TEE use cases
  - TEE terminology
  - How TEE is implemented?
  - Hardware support for TEE
  - ARM's Trustzone
  - TEE implementations
  - Limitation of TEE

  - ❖ **Lab #16: TrustZone on Armv8-M**
  - ❖ **Lab #17: Vulnerable TrustZone**

- **Application Security**
  - Access control introduction
  - Access control models
  - Benefits of using access control in Embedded Systems
  - Software integrity
  - The boot process
  - Threats to system's boot integrity
  - Bootloader authentication process
  - Boot chain of trust
  - FIT image authentication in Embedded Linux
  - Secure boot technologies
  - Secure boot + Access control
  - Secure boot with TPM process
  - Measured boot vs secured boot
  - Measured boot with TPM
  - Secure Over the Air update (OTA) process
  - Putting it all together
  - Application sandboxing
  - Virtualization & Hypervisors
  - Containerization

- **Securing Data in Motion**
  - Securing data in motion introduction
  - Choosing the network layer for security
  - Relying on data-link protection only

- IPsec protocol
- SSL/TLS
- IPSec vs SSL
- Intrusion detection & security monitoring
- Embedded firewalls

- **Securing Data at Rest**
  - Securing data at rest introduction
  - Threats to data encryption solutions
  - Software IP protection
  - Full Disk Encryption
  - Block level encryption
  - Filesystem encryption
  - Application-level encryption
  - Secure information & data logging
  - Secure key storage

  - ❖ **Lab #18: AES-GCM**

- **Embedded Systems Security Best Practices**
  - Secure by design
  - Defense in depth
  - Domain separation
  - Attack surface reduction
  - Least privilege
  - Least sharing
  - Mediated access
  - Protective defaults
  - Anomaly detection
  - Distributed privilege
  - Zero-Trust architecture
  - Minimal trusted elements
  - Least persistence
  - Protective failure
  - Continuous protection
  - Redundancy
  - Input validation
  - Security by obscurity and it’s issues
  - KISS
  - Usable security
  - Security standards for Embedded Systems
  - Coding standards and guidelines