Embedded/Connected Device Secure Coding

4-Day Course

Syllabus

AppSec Labs Ltd.

info@appsec-labs.com | https://appsec-labs.com | 23 HaTa'as St., Kfar Saba 44641 Israel | T: +972-9-7485005 | F: +972-9-7730595

Embedded/Connected Device Secure Coding
4-Day Course

Course description

Secure Programming is the last line of defense against attacks targeted toward our systems. This course shows you how to identify security flaws & implement security countermeasures in different areas of the software development lifecycle and apply these skills to improve the overall quality of the products and applications. Using sound programming techniques and best practices, in addition to understanding and practicing the capabilities of the potential hacker, as shown in this course, you can produce high-quality code that stands up to attacks.

The course covers major security principles in C/C++, software, and hardware vulnerabilities caused be unsecure coding and practices. The course also cover common tools and techniques used against embedded/connected devices. The objectives of the course are to acquaint students with security concepts and terminology, and to provide them with a solid foundation for developing software using the best practices in C/C++. By course completion, students should be proficient in secure programming and have learnt the basics of security analysis and design. Students should then be able to develop, design and maintain a secure product, using security methods and techniques for the C/C++ language and in the IoT world.

Target audience

Members of the software development team:

- C / C++ Developers
- Designers & Architects

Prerequisites

Before attending this course, students should be familiar with:

- C/C++ language
- Background in memory management
- Background in OS mechanisms

Course topics

Day 1

Information gathering

- Device Unpacking
- PCB Analysis
- Data Sheets
- FCC

Serial and Connections

- Identifying Inputs
- Bus Pirate
- GoodFET
- Getting Console/Shell

Debug

- Using Logic Analyzer
- Signal Monitoring
- Digital Decoding
- JTAG Overview
- Jtagulator
- OpenOCD
- JTAG Debugging

Availability

- Application/OS crash
- CPU/Memory starvation
- Resource starvation
- Triggering high network bandwidth
- User level DOS
- Concurrency & Race conditions
- Race window/objects
- Mutual Exclusion
- Deadlock
- Time of Check/Time of Use (TOCTOU)
- Files as Locks and temporary files
- Symbolic link attacks
- Using atomic operations

- **Hands-on Labs**

Day 2

Buffer Overflows and Code Injections

- Stack Overflows attacks
- Heap overflows attacks
- Array indexing attacks
- Format strings attacks
- Secure vs. Insecure API's
- Stack guards
- Compiler checks
- Better ways to manipulate strings and buffers

Integer Overflows

- Integer / Double overflows
- Integer conversion rules
- Signed and unsigned problems
- Safe integer usage
- Enforcing limits on integer values
- Preventing lost or misinterpreted data due to conversion
- Using secure integer libraries

Safe API

- Banned APIs
- Real-World Risks
- Using safe API's
- The 'n' Functions
- Detecting Dangerous APIs
- Alternatives
- StrSafe

Secure Memory Usage

- Secure memory handling
- Erasing Data
- Secure pointer usage
- Memory Dumps
- Use smart pointers for resource management
- Ensure pointer arithmetic
- Avoid null pointer dereferencing
- Ensure sensitive data is not paged to disk

- **Hands-on Labs**

Day 3

Network Security

- Network attacks
- Insecure Services
- Application Layer Threats and attacks
- Traffic sniffing
- Traffic manipulations
- Man-in-the-Middle
- Avoiding Server Socket Hijacking
- Firewall Friendly application

Cryptography

- Introduction to cryptography
- Symmetric encryption
- Asymmetric encryption
- ECC
- Transport Level encryption
- Storage Level encryption
- The problem with Crypto in IoT
- Lightweight cryptography

Secure Coding Tips

- Prefer Streams to C-Style Input and Output
- Avoid defining macros
- Do not ignore values returned by functions or methods
- Secure defaults and initializations
- Security principles
- Hardcoded secrets
- Static Code Tools

Integrating security into the development lifecycle

- **Hands-on Labs**

Day 4

Flash & Chip Manipulations

- Using a Device Programmer
- Connecting using SPI and I2C
- In-circuit Connection
- Pulling off the Chip
- Dumping the Content of a Chip
- Patching Flash Content
- Uploading a Modified Binary to Chip

Firmware Analysis

- Getting the Device Firmware
- binwalk
- Reversing the Binary with IDA
- Patching
- Bypassing Limitations
- Uploading Modified Firmware

Anti-Reversing

- Eliminate “symbolic info”
- Obfuscate the program
- Code Encryption
- Use anti-debugger tricks
- Code Checksums
- Confusing a Disassembler
- Inlining and Outlining sensitive code
- Interleaving Code

- **Hands-on Labs**