C/C++ Secure Coding

5-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

C and C++ Secure Coding
5-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 shown in this course, you can produce high-quality code that stands up to attack.

The course covers major security principles in C/C++ and software vulnerabilities caused be unsecure coding. 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 applications using security methods and techniques for the C/C++ language.

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

Introduction to Secure Coding

- What is application security?
- SDLC – Secure Development Lifecycle
- Cyber security attacks
- Secure design
- Secure coding principles
- Security testing
- Attackers motivation
- The defender's approach

Buffer Overflows and Code Injections

- Stack Overflows attacks
- Heap overflows attacks
- Array indexing attacks
- Format strings attacks
- Unsafe API's
- Safer API's
- Stack guards
- Compiler checks
- Better ways to manipulate strings and buffers
- **Hands-on labs**

Integer Overflows

- Int / 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
- **Hands-on labs**

Safe API

- Dangerous and banned APIs
- Real-World Risks
- Using safe API's
- The 'n' Functions
- Detecting Dangerous APIs
- Alternatives
- StrSafe
- **Hands-on labs**

Input Validation

- What is considered Input?
- Common Errors
- Black List VS. White List Validation
- Attack scenario: Canonicalization
- String Manipulation and Comparison
- Data Type Conversion
- Regular Expressions
- Validation practices
- **Hands-on labs**

Network Security

- Introduction to Networking
- Network attacks
- Insecure Services
- Application Layer Threats and attacks
- Traffic Sniffing
- Traffic Manipulation
- Man-in-the-Middle
- Avoiding Server Socket Hijacking
- Firewall Friendly Application
- **Hands-on labs**

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**

Protecting Data at Rest

- Goals of cryptography – CIA
- Introduction to cryptography
- Symmetric encryption
- ECB vs. CBC
- Message integrity
- Hash vs. HMAC
- Secure password storage
- Salt
- **Hands-on labs**

Protecting Data at Motion

- Asymmetric Encryption
- RSA
- ECC
- The SSL protocol
- Digital signature
- PKI
- Digital certificates
- Certificate authority
- Challenge-response
- Authentication
- **Hands-on labs**

Key Management

- Key generation
- Key derivation
- Key storage
- TPM
- Key exchange (DH)
- **Hands-on labs**

Authentication & Authorization

- Authentication scenarios
- Common mistakes
- Attack scenario: Brute-force
- Authentication protocols
- Attack scenario: weak passwords
- CAPTCHA
- Authorization attacks
- Authorization models
- Access Control List (ACL)
- Role Based Access Control (RBAC)
- Attack scenario: Exposed functionality via anonymous authentication
- **Hands-on labs**

Secure File Handling

- Directory Traversal attacks
- File canonicalization attacks
- Creating files with correct ACLs
- Ensure files are closed when no longer needed
- Insecure usage of shared directories
- **Hands-on labs**

Thread safety

- Concurrency & Race conditions
- Mutual Exclusion
- Deadlock
- Time of Check/Time of Use (TOCTOU)
- Files as Locks
- Symbolic link attacks
- Temporary files
- Handling the race window
- Controlling race objects
- Using atomic operations

Application Denial of Service vulnerabilities

- Application / OS crash
- CPU starvation
- Memory starvation
- File system starvation
- Resource starvation
- Triggering high network bandwidth
- User-level DoS
- Exploiting a specific vulnerability to cause DoS
- **Hands-on labs**

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
- The least privilege principle
- The defense in depth principle
- The segmentation principle
- Avoiding hard coded secrets
- Use Static Code Tools
- Integrating security into the development lifecycle

Logging & Error Handling

- Process uncaught and unexpected exceptions
- Prevent sensitive information disclosure via errors
- Declare new exception classes for security
- Events you should and should not log
- Log integration with exception management
- **Hands-on labs**

Anti-Reversing

- Eliminate “symbolic info”
- Code Obfuscation and Code Encryption
- Anti-debugging tricks
- Code Checksums
- Confusing a Disassembler
- Inlining and Outlining sensitive code
- Interleaving Code
- Existing tools
- **Hands-on labs**