Certified C and C++ secure coding

Learn via: **Classroom / Virtual Classroom / Online**

Duration: **3 Gün**

https://bilginc.com/tr/egitim/certified-c-and-cplusplus-secure-coding-785-egitimi/

Overview

To put it bluntly, writing C/C++ code can be a minefield for reasons ranging from memory management or dealing with legacy code to sharp deadlines and code maintainability. Yet, beyond all that, what if we told you that attackers were trying to break into your code right now? How likely would they be to succeed?

This course will change the way you look at your C/C++ code. We'll teach you the common weaknesses and their consequences that can allow hackers to attack your system, and – more importantly – best practices you can apply to protect yourself. We give you a holistic view on C/C++ programming mistakes and their countermeasures from the machine code level to virtual functions and OS memory management. We present the entire course through live practical exercises to keep it engaging and fun.

Writing secure code will give you a distinct edge over your competitors. It is your choice to be ahead of the pack – take a step and be a game-changer in the fight against cybercrime.

Topics include:

- IT security and secure coding
- x86 machine code, memory layout and stack operations
- Buffer overflow
- Practical cryptography
- Security protocols
- XML security
- Common coding errors and vulnerabilities
- Principles of security and secure coding
- Knowledge sources

Prerequisites

General C/C++ development skills are required.

What You Will Learn

- Understand basic concepts of security, IT security and secure coding
- Realize the severe consequences of unsecure buffer handling
- Understand the architectural protection techniques and their weaknesses
- Have a practical understanding of cryptography
- Understand essential security protocols
- Learn about XML security
- Learn about typical coding mistakes and how to avoid them
- Be informed about recent vulnerabilities in various platforms, frameworks and libraries
- Get sources and further readings on secure coding practices

Note: This course comes with a number of easy-to-understand exercises providing real-time ethical hacking fun. By accomplishing these exercises with the support of the trainer, participants can analyze vulnerable code snippets and commit attacks against them in order to fully understand the root causes of certain security problems. All exercises are prepared in a plug-and-play manner by using a pre-set desktop virtual machine, which provides a uniform development environment.

Outline

Day 1

IT security and secure coding

- Nature of security
- What is risk?
- IT security vs. secure coding
- From vulnerabilities to botnets and cybercrime
- Nature of security flaws
- Reasons of difficulty
- From an infected computer to targeted attacks
- The Seven Pernicious Kingdoms
- OWASP Top Ten 2017

x86 machine code, memory layout and stack operations

- Intel 80x86 Processors – main registers
- Intel 80x86 Processors – most important instructions
- Intel 80x86 Processors – flags
- Intel 80x86 Processors – control instructions
- Intel 80x86 Processors – stack handling and flow control
- The memory address layout
- The function calling mechanism in C/C++ on x86
- Calling conventions
- The local variables and the stack frame
- Function calls – prologue and epilogue of a function
- Stack frame of nested calls
- Stack frame of recursive functions

Buffer overflow

- Stack overflow
- Buffer overflow on the stack
- Overwriting the return address
- Exercises – introduction
- Exercise BOFIntro
- Exercise BOFShellcode
- Exercise BOFShellcode – spot the bug!
- Protection against stack overflow
- Specific protection methods
- Protection methods at different layers
- The PreDeCo matrix of software security
- Stack overflow – Prevention (during development)
- Stack overflow – Detection (during execution)
- Fortify istrumentation (FORTIFY_SOURCE)
- Exercise BOFShellcode – Fortify
- Stack smashing protection
- Stack smashing protection variants
- Stack smashing protection in GCC
- Exercise BOFShellcode – Stack smashing protection
- Effects of stack smashing protection
- Bypassing stack smashing protection
- Overwriting arguments – Mitigation
- Address Space Layout Randomization (ASLR)
- Randomization with ASLR
- Using ASLR
- Circumventing ASLR: NOP sledding
- Non executable memory areas – the NX bit
- Access Control on memory segments
- The Never eXecute (NX) bit
- Exercise BOFShellcode – Enforcing NX memory segments

Day 2

Buffer overflow

- Return-to-libc attack – Circumventing the NX bit protection
- Circumventing memory execution protection
- Return-to-libc attack
- Exercise BOFShellcode – The Return-to-libc attack
- Return oriented programming (ROP)

- Exploiting with ROP
- ROP gadgets
- ROP gadget - Register fill with constants
- ROP gadget – Memory write
- Combining the ROP gadgets
- Real ROP attack scenarios
- ROP mitigation
- Mitigation techniques of ROP attack
- Heap overflow
- Memory allocation managed by a doubly-linked list
- Buffer overflow on the heap
- Steps of freeing and joining memory blocks
- Freeing allocated memory blocks
- Case study – Heartbleed
- TLS Heartbeat Extension
- Heartbleed – information leakage in OpenSSL
- Heartbleed – fix in v1.0.1g
- Protection against heap overflow

Practical cryptography

- Cryptosystems
- Elements of a cryptosystem
- Symmetric-key cryptography
- Providing confidentiality with symmetric cryptography
- Symmetric encryption algorithms
- Block ciphers – modes of operation
- Other cryptographic algorithms
- Hash or message digest
- Hash algorithms
- SHAttered
- Message Authentication Code (MAC)
- Providing integrity and authenticity with a symmetric key
- Random numbers and cryptography
- Cryptographically-strong PRNGs
- Hardware-based TRNGs
- Asymmetric (public-key) cryptography
- Providing confidentiality with public-key encryption
- Rule of thumb – possession of private key
- The RSA algorithm
- Introduction to RSA algorithm
- Encrypting with RSA
- Combining symmetric and asymmetric algorithms
- Digital signing with RSA
- Public Key Infrastructure (PKI)
- Man-in-the-Middle (MitM) attack
- Digital certificates against MitM attack
- Certificate Authorities in Public Key Infrastructure
- X.509 digital certificate

Security protocols

- Secure network protocols
- Specific vs. general solutions
- SSL/TLS protocols
- Security services
- SSL/TLS handshake

XML security

- Introduction
- XML parsing
- XML injection
- (Ab)using CDATA to store XSS payload in XML
- Exercise – XML injection
- Abusing XML Entity
- XML Entity introduction

- XML bomb
- Exercise – XML bomb
- XML external entity attack (XXE) – resource inclusion
- XML external entity attack – URL invocation
- XML external entity attack – parameter entities
- Exercise – XXE attack
- Preventing entity-related attacks
- Case study – XXE in Google Toolbar

Common coding errors and vulnerabilities

- Improper error and exception handling
- Typical problems with error and exception handling
- Empty catch block
- Overly broad catch
- Exercise ErrorHandling – spot the bug!
- Exercise – Error handling
- Case study – '#iamroot' authentication bypass in macOS
- Authentication process in macOS (High Sierra)
- Incorrect error handling in opendirectoryd
- The #iamroot vulnerability (CVE-2017-13872)
- Time and state problems
- Time and state related problems
- Serialization errors (TOCTTOU)
- Attacks with symbolic links
- Exercise TOCTTOU
- Code quality problems
- Dangers arising from poor code quality
- Poor code quality – spot the bug!
- Unreleased resources
- Type mismatch – Spot the bug!
- Exercise TypeMismatch
- Memory allocation problems
- Smart pointers
- Zero length allocation
- Double free
- Mixing delete and delete[]
- Use after free
- Use after free – Instance of a class
- Spot the bug
- Use after free – Dangling pointers

Day 3

Common coding errors and vulnerabilities

- Input validation
- Input validation concepts
- Integer problems
- Representation of negative integers
- Integer ranges
- Integer overflow
- Integer problems in C/C++
- The integer promotion rule in C/C++
- Arithmetic overflow – spot the bug!
- Exercise IntOverflow
- What is the value of abs(INT_MIN)?
- Signedness bug – spot the bug!
- Integer truncation – spot the bug!
- Integer problem – best practices
- Case study – Android Stagefright
- Injection flaws
- SQL Injection exercise
- Typical SQL Injection attack methods
- Blind and time-based SQL injection
- SQL Injection protection methods
- Command injection
- Command injection exercise – starting Netcat

- Printf format string bug
- Printf format strings
- Printf format string bug – exploitation
- Exercise Printf
- Printf format string exploit – overwriting the return address
- Mitigation of printf format string problem
- Some other input validation problems
- Array indexing – spot the bug!
- Off-by-one and other null termination errors
- The Unicode bug
- Path traversal vulnerability
- Path traversal mitigation
- Log forging
- Some other typical problems with log files
- Case study - Shellshock
- Shellshock – basics of using functions in bash
- Shellshock – vulnerability in bash
- Exercise - Shellshock
- Shellshock fix and counterattacks
- Exercise – Command override with environment variables
- Improper use of security features
- Typical problems related to the use of security features
- Insecure randomness
- Weak PRNGs in C and C++
- Stronger PRNGs in C
- Password management
- Exercise – Weakness of hashed passwords
- Password management and storage
- Brute forcing
- Special purpose hash algorithms for password storage
- Argon2 and PBKDF2 implementations in C/C++
- bcrypt and scrypt implementations in C/C++
- Case study – the Ashley Madison data breach
- Typical mistakes in password management
- Exercise – Hard coded passwords
- Insufficient anti-automation
- Captcha
- Captcha weaknesses

Principles of security and secure coding

- Matt Bishop's principles of robust programming
- The security principles of Saltzer and Schroeder

Knowledge sources

- Secure coding sources – a starter kit
- Vulnerability databases
- Recommended books – C/C++