OWASP Top 10, Secure Coding Fundamentals

CL-OAF | Virtual classroom | 2 days

Audience: Web developers
Preparedness: General Web development
Exercises: Hands-on

Writing web applications can be rather complex – reasons range from dealing with legacy technologies or underdocumented third-party components 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 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 cover typical Web vulnerabilities with a focus on how they affect web apps on the entire stack – from the base environment to modern AJAX and HTML5-based frontends. In addition, we discuss the security aspects of different platforms as well as typical programming mistakes you need to be aware of. 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.

Outline:

IT security and secure coding
Client-side security
XML security
Practical cryptography
Common coding errors and vulnerabilities
Denial of service

Participants attending this course will:

Understand basic concepts of security, IT security and secure coding

Learn about XML security

Learn client-side vulnerabilities and secure coding practices

Understand security concepts of Web services

Learn about JSON security

Learn about typical coding mistakes and how to avoid them

Learn about denial of service attacks and protections

Related courses:

- CL-OJF - OWASP Top 10, Java Secure Coding Fundamentals (Virtual classroom, 2 days)
- CL-OJA - OWASP Top 10, Java Secure Coding Follow Up (Virtual classroom, 1 days)
- CL-ONF - OWASP Top 10, C# Secure Coding Fundamentals (Virtual classroom, 2 days)
- CL-ONA - OWASP Top 10, C# Secure Coding Follow Up (Virtual classroom, 1 days)
- CL-WSC - Web application security (Onsite / Virtual classroom, 3 days)
- CL-WSM - Web application security master course (Onsite / Virtual classroom, 5 days)
- CL-WTS - Web application security testing (Onsite / Virtual classroom, 3 days)
- CL-JSM - Java and Web application security master course (Onsite / Virtual classroom, 5 days)
- CL-NSM - C# and Web application security master course (Onsite / Virtual classroom, 5 days)

Detailed table of contents

Day 1

Introduction

- About the workshop
- Let's meet
- What to expect

The story of Kevin Montes

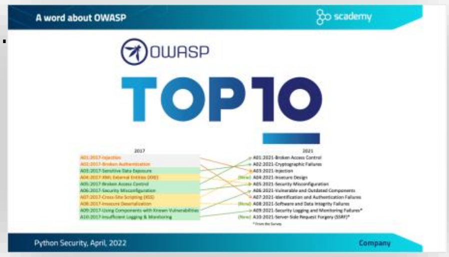
- How real-world security events shape our life
- What happened behind the scenes?
- Developers are the key to securing any application

Lab environment introduction

Team forming

Introduction to OWASP Top10

- Top10 in 2017 and 2021 ....
- How to use OWASP Top10



A word about OWASP

scademy

OWASP

TOP10

2017

A01:2017-Injection
A02:2017-Broken Authentication
A03:2017-Sensitive Data Exposure
A04:2017-XML External Entities (XXE)
A05:2017-Broken Access Control
A06:2017-Security Misconfiguration
A07:2017-Cross-Site Scripting (XSS)
A08:2017-Insecure Deserialization
A09:2017-Using Components with Known Vulnerabilities
A10:2017-Insufficient Logging & Monitoring

2021

A01:2021-Broken Access Control
A02:2021-Cryptographic Failures
A03:2021-Injection
(New) A04:2021-Insecure Design
A05:2021-Security Misconfiguration
A06:2021-Vulnerable and Outdated Components
A07:2021-Identification and Authentication Failures
(New) A08:2021-Software and Data Integrity Failures
A09:2021-Security Logging and Monitoring Failures*
(New) A10:2021-Server-Side Request Forgery (SSRF)*

* From the Survey

Python Security, April, 2022

Company

A01:2021 - Broken Access Control

- Definition
- Real-life examples
- Hands-on exercises
  - Local-file inclusion
  - Secret discovery by probing
  - Insecure Direct Object Reference (IDOR)
  - Cross-site Request Forgery
  - Session manipulation
- Discussion around business impact and countermeasures
  - Path validation and canonicalization
  - Access control patterns
  - CSRF tokens and SameSite cookies

A02:2021 - Cryptographic Failures

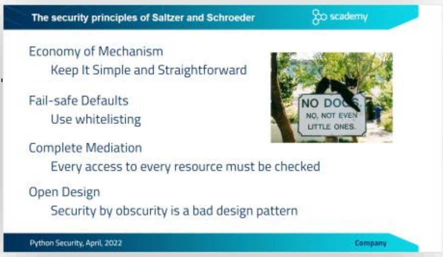
- Definition
- Real-life examples
- Hands-on exercises
  - Using deprecated cryptography
  - Weak encryption mechanisms
  - HMAC collision
  - Cryptographic bad practices
- Discussion around business impact and countermeasures
  - Rule #1 of implementing cryptography
  - Hash or message digest
  - Hash algorithms

A03:2021 – Injection

- Definition
- Real-life examples
- Hands-on exercises
  - SQL Injection
  - XSS
  - Template injection
  - XML injection
- Discussion around business impact and countermeasures
  - Prepared statements
  - Input validation
  - Filtering, sanitization

A04:2021 - Insecure design

- Definition...
- Real-life examples
- Hands-on exercises
  - Client-side authentication
  - Client-side include
  - Insecure file upload
- Discussion around business impact and countermeasures
  - Session handling best practices
  - Cookie security considerations
  - HTML5 security



The security principles of Saltzer and Schroeder

Economy of Mechanism
Keep It Simple and Straightforward

Fail-safe Defaults
Use whitelisting

Complete Mediation
Every access to every resource must be checked

Open Design
Security by obscurity is a bad design pattern

Python Security, April, 2022 Company

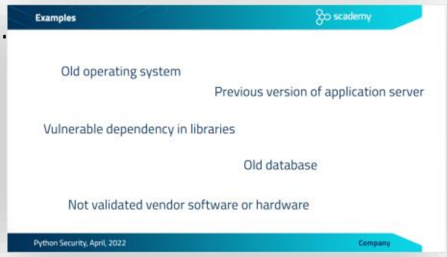
Day 2

A05:2021 - Security Misconfiguration

- Definition
- Real-life examples
- Hands-on exercises
  - XML external entity (XXE) – DoS
  - XML external entity – URL invocation
  - Path traversal
  - Security headers
  - Brute forcing passwords
- Discussion around business impact and countermeasures
  - XML parser options
  - Input validation
  - Content Security Policy
  - X-XSS-Protection
  - Password salting and stretching

A06:2021 - Vulnerable and Outdated

- Definition
- Real-life examples..
- Hands-on exercises
  - Check and fix 3rd party dependencies
  - Exploit vulnerable components
- Discussion around business impact and countermeasures
  - Keeping dependencies up to date
  - Updating dependencies immediately vs regularly



Examples

scademy

Old operating system

Previous version of application server

Vulnerable dependency in libraries

Old database

Not validated vendor software or hardware

Python Security, April, 2022

Company

A07:2021 - Identification and Authentication Failures

- Definition
- Real-life examples
- Hands-on exercises
  - Finding design flaws in source code
  - Session hijacking
- Discussion around business impact and countermeasures
  - Secure secret management

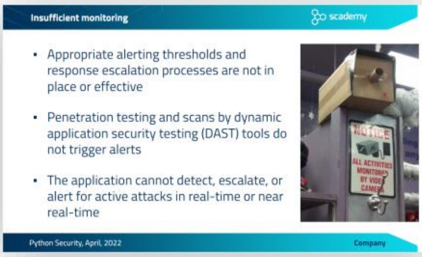
  - HTTPS
  - Using web frameworks for authentication
  - Identity verification

A08:2021 - Software and Data Integrity Failures

- Definition
- Real-life examples
- Hands-on exercises
  - Library from an untrusted source
- Discussion around business impact and countermeas ures
  - Using trusted sources
  - Integrity verification

A09:2021 - Security Logging and Monitoring Failures

- Definition
- Real-life examples
- Hands-on exercises
  - Logging sensitive information
  - Log injection
- Discussion around business impact and countermeasures
  - Choosing what to log
  - Input filtering and sanitization



Insufficient monitoring

- Appropriate alerting thresholds and response escalation processes are not in place or effective
- Penetration testing and scans by dynamic application security testing (DAST) tools do not trigger alerts
- The application cannot detect, escalate, or alert for active attacks in real-time or near real-time

Python Security, April, 2022

Company

A10:2021 - Server-Side Request Forgery

- Definition
- Real-life examples
- Hands-on exercises
  - Server-side request forgery
  - Remote file inclusion
- Discussion around business impact and countermeasures
  - Input validation
  - Domain allowlisting
  - URL schema enforcing
  - Authentication on every endpoint