Web application security in Python

CELPyWeb | 1 year subscription | e-Learning | Online VM

The course provides a comprehensive exploration of secure coding principles and practices tailored specifically for Python developers. Starting off from the foundations of cybersecurity, you will understand the consequences of insecure code by examining threats through the lens of the CIA triad.

In the main part of the material, you will systematically walk through the various vulnerabilities outlined in the OWASP Top Ten. As you progress through the modules investigating the intricacies of authentication and authorization, through realizing the practical aspects of cryptography, to tackling injection attacks, you will gain a deep understanding of both theoretical concepts and practical skills for securing Python web applications. Further subjects include error handling, code quality or denial of service, as well as XML and JSON security, and security considerations of the Python platform.

These modules go beyond just the theory. Not only do they identify vulnerabilities, show their consequences, and detail the best practices, but - through hands-on labs and real-world case studies - they offer practical experience in identifying, exploiting, and mitigating these security risks within Python-based web applications.

So that you are prepared for the forces of the dark side.

So that nothing unexpected happens in your code.

Nothing.

Note: This course content is available as an e-learning subscription. We reserve a period of 3 months to digest the foundational material, after which we activate shorter learning units on a monthly basis. This gives secure coding efforts an initial boost, and builds up sustained readiness over time. These learning units are indicated in red in the table of contents below.

Cyber security skills and drills

Foundational material

46 LABS

21 CASE STUDIES

Monthly learning units

2-3 LABS

CASE STUDY

Audience

Python developers working on Web applications

Preparedness

General Python and Web development

Outline

- Cyber security basics
- The OWASP Top Ten
- A01 - Broken Access Control
- A02 - Cryptographic Failures
- Cryptography
- A03 - Injection
- A04 - Insecure Design
- A05 - Security Misconfiguration
- A06 - Vulnerable and Outdated Components
- A07 - Identification and Authentication Failures
- A08 - Software and Data Integrity Failures
- A09 - Security Logging and Monitoring Failures
- A10 - Server-Side Request Forgery
- Wrap up

Standards and references

OWASP, CWE and Fortify Taxonomy

What you'll have learned

- Getting familiar with essential cyber security concepts
- Identify Web application vulnerabilities and their consequences
- Learn the security best practices in C#
- Input validation approaches and principles
- Understanding Web application security issues
- Detailed analysis of the OWASP Top Ten elements
- Putting Web application security in the context of Python
- Going beyond the low hanging fruits

Table of contents

Horizontal authorization

- Authorization bypass through user-controlled keys

Case study – Authorization bypass on Facebook

Lab – Horizontal authorization

File upload

- Unrestricted file upload
- Good practices

Lab – Unrestricted file upload

Cross-site Request Forgery (CSRF)

Lab – Cross-site Request Forgery

Cross-site Request Forgery (CSRF) best practices

- CSRF best practices
- CSRF defense in depth

Lab – CSRF protection with tokens

› A02 - Cryptographic Failures

› Cryptography

Information exposure

- Exposure through extracted data and aggregation

Case study – Strava data exposure

- Leaking system information

Cryptography for developers

- Cryptography basics
- Cryptography in Python

PRNG

- Random number generation
- Pseudo random number generators (PRNGs)
- Cryptographically secure PRNGs
- Using virtual random streams

Case study – Equifax credit account freeze

PRNG in Python

- Weak PRNGs
- Using random numbers

Lab – Using random numbers in Python

Hashing

- Hashing basics
- Common hashing mistakes
- Hashing in Python

Lab – Hashing in Python

Encryption

- Confidentiality protection
- Symmetric encryption
- Block ciphers
- Modes of operation
- Modes of operation and IV – best practices

Encryption in Python

- Symmetric encryption in Python

Lab – Symmetric encryption in Python

Asymmetric encryption

- The RSA algorithm
- Using RSA – best practices
- RSA in Python
- Combining symmetric and asymmetric algorithms

› A03 - Injection

Injection problems

- Injection principles
- Injection attacks

SQL injection

- SQL injection basics

Lab – SQL injection

SQL injection attack techniques

- Attack techniques
- Content-based blind SQL injection
- Time-based blind SQL injection

SQL injection best practices

- Input validation
- Parameterized queries

Lab – Using prepared statements

SQL injection additional considerations

- Database defense in depth

Case study – Hacking Fortnite accounts

Code injection - OS command injection

- Code injection
- Code injection via input()
- OS command injection

Lab – Command injection

OS command injection best practices

- Avoiding command injection with the right APIs

Lab – Command injection best practices

Case study – Shellshock

Lab - Shellshock

HTML injection - Cross-site scripting (XSS)

- Cross-site scripting basics
- Persistent cross-site scripting
- Reflected cross-site scripting
- Client-side (DOM-based) cross-site scripting

XSS attacks

Lab – Stored XSS

Lab – Reflected XSS

Case study – XSS in Fortnite accounts

XSS best practices 1

- Protection principles - escaping
- XSS protection APIs in Python

Lab – XSS fix / stored

XSS best practices 2

- XSS protection in Jinja2

Lab – XSS fix / reflected

- Additional protection layers – defense in depth

Template injection (Unit 6)

- Script injection
- Server-side template injection (SSTI)

Lab – Template injection

Case study – Template injection in Shopify leading to RCE

- Client-side template injection (CSTI) in Angular
- Client-side template injection (CSTI) in React

Input validation principles 1 (Unit 1)

- Input validation principles
- Denylists and allowlists
- Data validation techniques

Lab – Input validation

Input validation principles 2 (Unit 1)

- What to validate – the attack surface
- Where to validate – defense in depth
- When to validate – validation vs transformations

Case study – Missing input validation in Upserve

Input validation principles 3 (Unit 1)

- Output sanitization
- Encoding challenges
- Unicode challenges

Lab – Encoding challenges

Lab – Dealing with Unicode homoglyph attacks

- Validation with regex

Integer handling problems (Unit 2)

- Representing signed numbers
- Integer visualization
- Integers in Python
- Integer overflow
- Integer overflows in ctypes and numpy

Lab – Integer problems in Python

Other numeric problems (Unit 2)

- Working with floating-point numbers

Path traversal and file validation (Unit 3)

- Path traversal

Lab – Path traversal

- Path traversal-related examples
- Additional challenges in Windows
- Virtual resources
- Path traversal best practices

Lab – Path canonicalization

Native code (CFFI) issues (Unit 4)

- Native code dependence

Lab – Unsafe native code

- Best practices for dealing with native code

A04 - Insecure Design

Insecure design

- The STRIDE model of threats
- Secure design principles of Saltzer and Schroeder

Insecure design - Saltzer and Schroeder 1

- Economy of mechanism
- Fail-safe defaults
- Complete mediation
- Open design

Insecure design - Saltzer and Schroeder 2

- Separation of privilege
- Least privilege
- Least common mechanism
- Psychological acceptability

Client-side security (Unit 5)

- Same Origin Policy
- Simple request
- Preflight request
- Cross-Origin Resource Sharing (CORS)

Clickjacking

- Frame sandboxing
- Cross-Frame Scripting (XFS) attacks

Lab - Clickjacking

- Clickjacking beyond hijacking a click

Anti-clickjacking best practices

- Clickjacking protection best practices

Lab – Using CSP to prevent clickjacking

A05 – Security Misconfiguration

Misconfiguration and XML parsing

- Configuration principles
- XML Entities
- DTD and the entities
- Entity expansion

XML External Entity (XXE)

- File inclusion with external entities
- Server-Side Request Forgery with external entities

Lab – External entity attack

Case study – XXE vulnerability in SAP Store

XXE best practices

- Preventing XXE

Lab – Prohibiting DTD

A06 – Vulnerable and Outdated Components

Vulnerable components and dependencies

- Using vulnerable components
- Assessing the environment
- Hardening
- Untrusted functionality import
- Malicious packages in Python

Vulnerability management

- Patch management
- Vulnerability management
- Vulnerability databases

Lab – Finding vulnerabilities in third-party components

A07 – Identification and Authentication Failures

Authentication

- Authentication basics
- Multi-factor authentication (MFA)
- Authentication weaknesses

Case study – PayPal 2FA bypass

Session security

- Session management essentials
- Why do we protect session IDs – Session hijacking
- Session fixation
- Session ID best practices

Password management

- Storing account passwords
- Password in transit

Lab – Is just hashing passwords enough?

Password storage

- Dictionary attacks and brute forcing
- Salting
- Adaptive hash functions for password storage

Lab – Using adaptive hash functions in Python

Password policy

- NIST authenticator requirements for memorized secrets

Password storage - a case study

Case study – The Ashley Madison data breach
The dictionary attack
The ultimate crack
Exploitation and the lessons learned

Additional password management challenges

- Password database migration
- (Mis)handling None passwords

Password auditing (Unit 7)

- Using password cracking tools
- Password cracking in Windows

Lab – Password audit with John the Ripper

- On-line password brute forcing

Lab – On-line password brute forcing

- Password recovery issues
- Password recovery best practices

Lab – Password reset weakness

Outbound password management (Unit 7)

- Hard coded passwords

- Best practices

Lab – Hardcoded password

Case study – Hard-coded Telnet password in TOTOLINK T8

Protecting secrets in memory (Unit 7)

- Challenges in protecting memory

A08 - Software and Data Integrity Failures

Cryptography

Integrity protection and MAC

- Integrity protection
- Message Authentication Code (MAC)
- Calculating HMAC in Python

Lab – Calculating MAC in Python

Digital signatures

- Digital signature
- Digital signature with RSA
- ECC basics
- Digital signature with ECC

Lab – Digital signature with ECDSA in Python

Subresource integrity

- Importing JavaScript

Lab – Importing JavaScript

Case study – The British Airways data breach

Insecure deserialization

- Serialization and deserialization challenges
- Integrity – deserializing untrusted streams
- Deserialization with pickle

Lab – Deserializing with Pickle

- PyYAML deserialization challenges
- Integrity – deserialization best practices

A09 - Security Logging and Monitoring Failures

Logging

- Logging and monitoring principles
- Insufficient logging

Case study – Plaintext passwords at Facebook

- Logging best practices

Log forging (Unit 6)

- CRLF injection
- Log forging

Lab – Log forging

- Log forging – best practices

Monitoring (Unit 6)

- Monitoring best practices
- Firewalls and Web Application Firewalls (WAF)
- Intrusion detection and prevention

Case study – The Marriott Starwood data breach

A10 - Server-Side Request Forgery

SSRF

- Server-side Request Forgery (SSRF)

Case study – SSRF and the Capital One breach

Time and state (Unit 4)

Introduction

- Time and state

Race conditions

- Time of check to time of usage – TOCTTOU
- TOCTTOU attacks in practice

Lab - TOCTTOU

- Insecure temporary file
- Thread safety and the Global Interpreter Lock (GIL)
- Avoiding race conditions in Python

Case study: TOCTTOU in Calamares (CVE-2019-13178)

Locking and deadlocks

- Mutual exclusion and locking
- Deadlocks
- Synchronization and thread safety

› Error handling (Unit 8)

Principles

- Error and exception handling principles
- Information exposure through error reporting
- Information leakage via error pages

Lab – Flask information leakage

- Returning a misleading status code

Exception handling

- In the except block. And now what?
- Empty except block

Lab – Exception handling mess

› Denial of service (Unit 8)

- Flooding
- Resource exhaustion

Sustained client engagement

- Infinite loop

Case study – DoS against Tesla GUI via malicious web page

- Economic Denial of Sustainability (EDoS)

Amplification

- Some amplification examples

Algorithmic complexity issues

- Regular expression denial of service (ReDoS)

Lab – ReDoS

- Dealing with ReDoS
- Hash table collision
- How do hash tables work?
- Hash collision against hash tables

› JSON security (Unit 9)

- JSON validation
- JSON injection
- Dangers of JSONP
- JSON/JavaScript hijacking
- Best practices

Case study – ReactJS vulnerability in HackerOne

› XML security (Unit 9)

- XML validation
- XML injection
- XPath injection
- Blind XPath injection
- XSLT injection
- XQuery injection
- XML signature
- XML signature wrapping

Case study – signature wrapping in single sign-on solutions

- XML encryption
- XML encryption wrapping

› Python platform security (Unit 5)

- The Python ecosystem and its attack surface
- Python bytecode and security
- Security features offered by Python
- PEP 578 and audit hooks
- The difficulties of sandboxing untrusted code
- Sandboxing Python

› Wrap up

Software security principles

- Principles of robust programming by Matt Bishop

Sources and further readings

- Software security sources and further reading
- Python resources