Web application security in Python

CYDPyWeb3d | 3 days | On-site or online | Hands-on

Your Web application written in Python works as intended, so you are done, right? But did you consider feeding in incorrect values? 16Gbs of data? A null? An apostrophe? Negative numbers, or specifically -1 or -2^{31}? Because that's what the bad guys will do – and the list is far from complete.

Handling security needs a healthy level of paranoia, and this is what this course provides: a strong emotional engagement by lots of hands on labs and stories from real life, all to substantially improve code hygiene. Mistakes, consequences, and best practices are our blood, sweat and tears.

The curriculum goes through the common Web application security issues following the OWASP Top Ten but goes far beyond it both in coverage and the details.

All this is put in the context of Python, and extended by core programming issues, discussing security pitfalls of the programming language.

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

So that nothing unexpected happens.

Nothing.

Cyber security skills and drills

Audience

Python developers working on Web applications

Group size

12 participants

Outline

- Cyber security basics
- The OWASP Top Ten
- Common software security weaknesses
- JSON security
- Wrap up

Preparedness

General Python and Web development

29 labs

12 case studies

What you'll have learned

- Getting familiar with essential cyber security concepts
- 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
- Managing vulnerabilities in third party components
- Identify vulnerabilities and their consequences
- Learn the security best practices in Python
- Input validation approaches and principles

Table of contents

Day 1

- *Case study – Command injection via ping*
- Script injection
  - Server-side template injection (SSTI)
  - *Lab – Template injection*
- General protection best practices

A2 - Broken Authentication

- Authentication basics
- Multi-factor authentication
- Authentication weaknesses - spoofing
- Spoofing on the Web

Case study – PayPal 2FA bypass

- Password management
  - Inbound password management
    - Storing account passwords
    - Password in transit
    - *Lab – Is just hashing passwords enough?*
    - Dictionary attacks and brute forcing
    - Salting
    - Adaptive hash functions for password storage
    - Password policy
    - NIST authenticator requirements for memorized secrets
    - *Case study – The Ashley Madison data breach*
    - *The dictionary attack*
    - *The ultimate crack*
    - *Exploitation and the lessons learned*
    - Password database migration
    - (Mis)handling None passwords

Day 2

› The OWASP Top Ten

A2 - Broken Authentication

- Password management
  - Outbound password management
    - Hard coded passwords
    - Best practices
    - *Lab – Hardcoded password*
    - Protecting sensitive information in memory
    - Challenges in protecting memory

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

A3 - Sensitive Data Exposure

- Information exposure
- Exposure through extracted data and aggregation

Case study – Strava data exposure

- Error and exception handling principles

A4 - XML External Entities (XXE)

- DTD and the entities
- Entity expansion

Lab – Billion laughs attack

- External Entity Attack (XXE)
  - File inclusion with external entities
  - Server-Side Request Forgery with external entities

Lab – External entity attack

Case study – XXE vulnerability in SAP Store

  - Preventing XXE

Lab – Using non-vulnerable parsers

A5 - Broken Access Control

- Access control basics
- Failure to restrict URL access
- Confused deputy
  - Insecure direct object reference (IDOR)

Lab – Insecure Direct Object Reference

  - 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

A6 - Security Misconfiguration

- Configuration principles
- Python configuration best practices

- Configuring Flask

A7 - Cross-site Scripting (XSS)

- Cross-site scripting basics
- Cross-site scripting types
  - Persistent cross-site scripting
  - Reflected cross-site scripting
  - Client-side (DOM-based) cross-site scripting
  - *Lab – Stored XSS*
  - *Lab – Reflected XSS*
  - *Case study – XSS in Fortnite accounts*
- XSS protection best practices
  - Protection principles - escaping
  - XSS protection APIs in Python
  - XSS protection in Jinja2
  - *Lab – XSS fix / stored*
  - *Lab – XSS fix / reflected*
  - Additional protection layers
  - Client-side protection principles

A8 - Insecure Deserialization

- Serialization and deserialization challenges
- Deserializing untrusted streams
- Deserialization with pickle
- *Lab – Deserializing with Pickle*
- PyYAML deserialization challenges
- Deserialization best practices

Day 3

› The OWASP Top Ten

A9 - Using Components with Known Vulnerabilities

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

- *Lab – Importing JavaScript*
- *Case study – The British Airways data breach*
- Vulnerability management
  - Patch management
  - Vulnerability management
  - Vulnerability databases

A10 - Insufficient Logging & Monitoring

- Logging and monitoring principles
- Insufficient logging
- *Case study – Plaintext passwords at Facebook*
- Logging best practices
- Monitoring best practices

Web application security beyond the Top Ten

- Client-side security
- Same Origin Policy
  - *Lab – Same-origin policy demo*
  - Tabnabbing
  - *Lab – Reverse tabnabbing*
- Frame sandboxing
  - Cross-Frame Scripting (XFS) attack
  - *Lab - Clickjacking*
  - Clickjacking beyond hijacking a click
  - Clickjacking protection best practices
  - *Lab – Using CSP to prevent clickjacking*

› Common software security weaknesses

Input validation

- Input validation principles
  - Blacklists and whitelists
  - Data validation techniques
  - *Lab – Input validation*
  - What to validate – the attack surface
  - Where to validate – defense in depth
  - How to validate – validation vs transformations
  - Output sanitization
  - Encoding challenges
  - *Lab – Encoding challenges*
  - Validation with regex

- Regular expression denial of service (ReDoS)
- *Lab – Regular expression denial of service (ReDoS)*
- Dealing with ReDoS

- Files and streams
  - Path traversal
  - Path traversal-related examples
  - *Lab – Path traversal*
  - Additional challenges in Windows
  - Virtual resources
  - Path traversal best practices
  - Format string issues
- Unsafe native code
  - Native code dependence
  - *Lab – Unsafe native code*
  - Best practices for dealing with native code

› JSON security

JSON injection

Dangers of JSONP

JSON/JavaScript hijacking

Best practices

Case study – ReactJS vulnerability in HackerOne

› Wrap up

Secure coding principles

- Principles of robust programming by Matt Bishop
- Secure design principles of Saltzer and Schröder

And now what?

- Software security sources and further reading
- Python resources