API security in Python

CYDPyApi3d | 3 days | On-site or online | Hands-on

Your application written in Python works as intended, so you are done, right? But do your APIs behave well for incorrect values? 16Gbs of data? A null? An apostrophe? Negative numbers, or specifically -1 or -2^31? Because these are the values the bad guys will feed in - and the list is far from complete.

The course provides a comprehensive walkthrough on the OWASP API Security Top Ten, equipping developers, security professionals, and architects with the knowledge to identify, mitigate, and prevent the most critical security risks in modern API-driven applications. Each of the ten risks - including Broken Object, Property and Function Level Authorization (BOLA, BOPLA and BFLA), Unrestricted Resource Consumption, Unsafe Consumption of APIs, and more - are discussed in detail with real-world examples, hands-on labs, and mitigation strategies. Topics are discussed in the context of classic APIs, rest APIs as well as GraphQL.

Beyond the top ten list, the course can also expand into further key security topics that are crucial for developers but often overlooked in API security, such as cryptography, integer overflows, and code quality.

Whether you are a beginner in API security or an experienced developer looking to sharpen your skills, this course offers valuable knowledge to build APIs that are not only functional and efficient but also secure and resilient.

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

So that nothing unexpected happens.

Nothing.

Skills and drills

23 LABS

19 CASE STUDIES

Audience

Python API developers

Group size

12 participants

Preparedness

General Python development

Outline

- Cyber security basics
- OWASP API Security Top Ten
- API1 - Broken Object Level Authorization
- API2 - Broken Authentication
- API3 - Broken Object Property Level Authorization
- API4 - Unrestricted Resource Consumption
- API5 - Broken Function Level Authorization
- API6 - Unrestricted Access to Sensitive Business Flows
- API7 - Server Side Request Forgery
- API8 - Security Misconfiguration
- API9 - Improper Inventory Management
- API10 - Unsafe Consumption of APIs
- Wrap up

Standards and references

CWE and Fortify Taxonomy

What you'll have learned

- Getting familiar with essential cyber security concepts
- Understanding API security issues
- Detailed analysis of the OWASP API Security Top Ten elements
- Putting API security in the context of Python
- Going beyond the low hanging fruits
- Managing vulnerabilities in third party components
- Input validation approaches and principles

Table of contents

Day 1

› **Cyber security basics**

What is security?

Threat and risk

Cyber security threat types – the CIA triad

Consequences of insecure software

› **OWASP API Security Top Ten**

OWASP API Security Top 10 2023

› **API1 - Broken Object Level Authorization**

Confused deputy

- Insecure direct object reference (IDOR)

Lab – Insecure Direct Object Reference

- Authorization bypass through user-controlled keys

Case study – Remote takeover of Nexx garage doors and alarms

Lab – Horizontal authorization

File upload

- Unrestricted file upload
- Good practices

Lab – Unrestricted file upload

› API2 – Broken Authentication

Authentication basics

Multi-factor authentication (MFA)

Case study – The InfinityGauntlet attack

Passwordless solutions

Time-based One Time Passwords (TOTP)

Authentication weaknesses

Spoofing on the Web

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
- *Lab – Using adaptive hash functions in Python*
- Using password cracking tools
- Password cracking in Windows
- Password change
- Password recovery issues
- Password recovery best practices
- *Lab – Password reset weakness*
- *Case study – Facebook account takeover via recovery code*
- *Case study – GitLab account takeover*
- Anti-automation

- Password policy
  - NIST authenticator requirements for memorized secrets
  - Password hardening
  - Using passphrases
  - Password database migration
  - (Mis)handling None passwords

Day 2

API3 - Broken Object Property Level Authorization

Information exposure

- Exposure through extracted data and aggregation
- *Case study – Strava data exposure*
- System information leakage
  - Leaking system information
- Information exposure best practices

Secrets management

- Hard coded passwords
- Best practices
- *Lab – Hardcoded password*
- Protecting sensitive information in memory
  - Challenges in protecting memory
  - *Case study – Microsoft secret key theft via dump files*

API4 - Unrestricted Resource Consumption

Denial of service

Flooding

Resource exhaustion

Sustained client engagement

Infinite loop

Economic Denial of Sustainability (EDoS)

Algorithmic complexity issues

- Regular expression denial of service (ReDoS)
  - *Lab – ReDoS*
  - Dealing with ReDoS
  - *Case study – ReDoS vulnerabilities in Python*

API5 - Broken Function Level Authorization

Authorization

- Access control basics
- Access control types
- Missing or improper authorization

- Failure to restrict URL access

Cross-site Request Forgery (CSRF)

Lab – Cross-site Request Forgery

- CSRF best practices
- CSRF defense in depth

Lab – CSRF protection with tokens

API6 – Unrestricted Access to Sensitive Business Flows

Security by design

- The STRIDE model of threats
- Secure design principles of Saltzer and Schroeder
  - Economy of mechanism
  - Fail-safe defaults
  - Complete mediation
  - Open design
  - Separation of privilege
  - Least privilege
  - Least common mechanism
  - Psychological acceptability

Logging and monitoring

- Logging and monitoring principles
- Insufficient logging

Case study – Plaintext passwords at Facebook

- Log forging
- Web log forging

Lab – Log forging

- Log forging – best practices
- Logging best practices
- Monitoring best practices

› API7 - Server Side Request Forgery

Server-side Request Forgery (SSRF)

Case study – SSRF in Ivanti Connect Secure

› API8 - Security Misconfiguration

Information exposure through error reporting

- Information leakage via error pages

Lab – Flask information leakage

Case study – Information leakage via errors in Apache Superset

Cookie security

- Cookie attributes

Same Origin Policy

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

Lab – Same-origin policy demo

Configuring XML parsers

- DTD and the entities
- Entity expansion
- External Entity Attack (XXE)
  - File inclusion with external entities
  - Server-Side Request Forgery with external entities
  - *Lab – External entity attack*
  - Preventing XXE
  - *Lab – Prohibiting DTD*
  - *Case study – XXE vulnerability in Ivanti products*

Day 3

API9 - Improper Inventory Management

Documentation blindspots

Dataflow blindspots

Using vulnerable components

Untrusted functionality import

Malicious packages in Python

Case study – The Polyfill.io supply chain attack

Vulnerability management

Lab – Finding vulnerabilities in third-party components

API10 - Unsafe Consumption of APIs

Input validation

- Input validation principles
- Denylists and allowlists
- *Case study – Denylist failure in urllib.parse.urlparse()*
- What to validate – the attack surface
- Where to validate – defense in depth
- When to validate – validation vs transformations
- Validation with regex
- Injection
  - Injection principles
  - Injection attacks
  - SQL injection
    - SQL injection basics
    - *Lab – SQL injection*
    - Attack techniques
    - Content-based blind SQL injection
    - Time-based blind SQL injection
    - SQL injection best practices
    - Input validation
    - Parameterized queries
    - *Lab – Using prepared statements*
    - *Case study – SQL injection against US airport security*
  - 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*
    - *Case study – Command injection in Ivanti security appliances*
- Open redirects and forwards
  - Open redirects and forwards – best practices
- Files and streams
  - Path traversal
  - *Lab – Path traversal*
  - Additional challenges in Windows
  - *Case study – File spoofing in WinRAR*
  - Path traversal best practices
  - *Lab – Path canonicalization*

› Wrap up

Secure coding principles

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

And now what?

- Software security sources and further reading
- Python resources