Web application security masterclass in Python

CYDPyWeb5d | 5 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.

Skills and drills

40 LABS

20 CASE STUDIES

Audience

Python developers working on Web applications

Group size

12 participants

Preparedness

General Python and Web development

Outline

- Cyber security basics
- The OWASP Top Ten 2025
- Security testing
- Wrap up

Standards and references

OWASP, CWE and Fortify Taxonomy

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
- Understanding how cryptography supports security
- Learning how to use cryptographic APIs correctly in Python
- Input validation approaches and principles
- Getting familiar with security testing techniques and tools

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

› The OWASP Top Ten 2025

A01 - Broken Access Control

- Access control basics
- Failure to restrict URL access
- Confused deputy
  - Insecure direct object reference (IDOR)
  - Path traversal
  - *Lab – Insecure Direct Object Reference*
  - Path traversal best practices
  - 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*
- Open redirects and forwards
  - Open redirects and forwards – best practices
- Cross-site Request Forgery (CSRF)
  - *Lab – Cross-site Request Forgery*
  - CSRF best practices
  - CSRF defense in depth
  - *Lab – CSRF protection with tokens*
  - Insecure temporary file
- Server-side Request Forgery (SSRF)

Case study – SSRF in Ivanti Connect Secure

A02 - Security Misconfiguration

- Configuration principles
- Python configuration best practices
  - Configuring Flask
- Web security configuration issues
  - Content Security Policy
  - Fetch directives
  - Source allowlisting
  - Strict CSP: using nonces and hashes
  - Navigation and reporting directives
  - Document restrictions and sandboxing
  - CSP best practices
- Cookie security
  - Cookie attributes
- XML entities
  - 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 2

› The OWASP Top Ten 2025

A02 - Security Misconfiguration (continued)

- Python platform security
  - 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
- Secrets management

- Hard coded passwords
- Best practices

Lab – Hardcoded password

A03 - Software Supply Chain Failures

- Using vulnerable components
- Assessing the environment
- Hardening
- Untrusted functionality import
- Malicious packages in Python
- Supply chain security and the Software Bill of Materials (SBOM)
- SBOM examples

Case study – The Polyfill.io supply chain attack

- Vulnerability management
  - Patch management
  - Vulnerability management
  - Vulnerability databases
  - Vulnerability rating – CVSS
  - CVSS – Base Metric Group
  - CVSS – Supplemental Metric Group
  - CVSS – Threat Metric Group
  - CVSS – Environmental Metric Group

  Lab – Finding vulnerabilities in third-party components
  - Bug bounty programs
  - DevOps, the CI / CD build process and Software Composition Analysis
  - Dependency checking in Python

  Lab – Detecting vulnerable components
- Use of dangerous and obsolete language elements
  - Using dangerous language elements
  - Using obsolete language elements
  - Security aspects of monkey patching in Python
  - Dangers of compile(), exec() and eval()

A04 - Cryptographic Failures

- Cryptography for developers
  - Cryptography basics
  - Cryptography in Python
  - Elementary algorithms
    - Hashing
    - Hashing basics
    - Common hashing mistakes

- Hashing in Python
- *Lab – Hashing in Python*
- Random number generation
- Pseudo random number generators (PRNGs)
- Cryptographically secure PRNGs
- Using virtual random streams
- Weak PRNGs
- Using random numbers
- *Lab – Using random numbers in Python*
- *Case study – Weak randomness in OpenVPN Access Server*

- Confidentiality protection
  - Symmetric encryption
  - Block ciphers
  - Modes of operation
  - Modes of operation and IV – best practices
  - Symmetric encryption in Python
  - *Lab – Symmetric encryption in Python*
  - Asymmetric encryption
  - The RSA algorithm
  - Using RSA – best practices
  - RSA in Python
  - *Case study – RSA attacks: Bleichenbacher, ROBOT, and Marvin*
  - Combining symmetric and asymmetric algorithms
  - Key exchange and agreement
  - Key exchange
  - Diffie-Hellman key agreement algorithm
  - Key exchange pitfalls and best practices

Day 3

› The OWASP Top Ten 2025

A05 - Injection

- 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
  - Output sanitization
  - Encoding challenges

- Unicode challenges
- Validation with regex

- Unsafe native code
  - Native code dependence
  - *Lab – Unsafe native code*
  - Best practices for dealing with native code
- 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*
    - Database defense in depth
    - *Case study – SQL injection against US airport security*
    - SQL injection protection and ORM
- 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*
- Script injection
  - *Case study – Script injection in Kubernetes NGINX Ingress*
  - Server-side template injection (SSTI)
  - *Lab – Template injection*
- HTML injection - 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 to RCE in Teltonika routers*

- XSS protection best practices
  - Protection principles - escaping
  - XSS protection APIs in Python
  - XSS protection in Jinja2
  - *Lab – XSS fix / stored*
  - *Lab – XSS fix / reflected*
  - Client-side protection principles
  - Additional protection layers – defense in depth
  - *Case study – XSS vulnerabilities in DrayTek Vigor routers*

Day 4

› The OWASP Top Ten 2025

A06 - Insecure 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
- Client-side security
  - Same Origin Policy
    - Simple request
    - Preflight request
    - Cross-Origin Resource Sharing (CORS)
    - *Lab – Same-origin policy demo*
  - Frame sandboxing
    - Cross-Frame Scripting (XFS) attacks
    - *Lab – Clickjacking*
    - Clickjacking beyond hijacking a click
    - Clickjacking protection best practices
    - *Lab – Using CSP to prevent clickjacking*

A07 - Authentication Failures

- 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
- User interface best practices
- *Lab – On-line password brute forcing*
- *Lab – Rate limiting to prevent online bruteforcing*

- Session handling
  - Session management essentials
  - Why do we protect session IDs – Session hijacking
  - Session fixation
  - Session invalidation
  - Session ID best practices
  - Session handling in Flask
- 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*
- Password policy
  - NIST authenticator requirements for memorized secrets
  - Password hardening
  - Using passphrases
  - Password database migration
  - (Mis)handling None passwords

A08 - Software and Data Integrity Failures

- Subresource integrity
  - Importing JavaScript
  - *Lab – Importing JavaScript*
  - *Case study – The British Airways data breach*

Day 5

› The OWASP Top Ten 2025

A08 - Software and Data Integrity Failures (continued)

- Integrity protection
  - Authenticity and non-repudiation
  - Message Authentication Code (MAC)
    - Calculating HMAC in Python
    - *Lab – Calculating MAC in Python*
  - Digital signature
    - Elliptic Curve Cryptography
    - ECC basics
    - Digital signature with ECC
    - Digital signature in Python
    - *Lab – Digital signature with ECDSA in Python*
  - Authenticated encryption
    - Authenticated encryption modes of operation
    - Authenticated encryption modes of operation: CCM
    - Authenticated encryption modes of operation: GCM
- Public Key Infrastructure (PKI)
  - Key management challenges
  - Certificates
    - Certificates and PKI
    - Chain of trust
    - X.509 certificates
    - PKI actors and procedures
    - Inappropriate certificate validation
    - PGP – Web of Trust
    - Certificate pinning
    - Certificate revocation
- Insecure deserialization
  - Serialization and deserialization challenges
  - Integrity – deserializing untrusted streams
  - Deserialization with pickle
  - *Lab – Deserializing with Pickle*
  - *Case study – The security of the machine learning supply chain*
  - *Case study – The first wave of supply chain attacks: RCE via pickle (2022)*
  - *Case study – Compromising the Hugging Face Hub repository*
  - Integrity – deserialization best practices

A09 - Logging and Alerting Failures

- Logging and monitoring principles
- Insufficient logging

Case study – Plaintext passwords at Facebook

- Logging best practices
- Monitoring best practices

A10 - Mishandling of Exceptional Conditions

- Error and exception handling principles
- Error handling
  - Returning a misleading status code
  - Information exposure through error reporting
    - Information leakage via error pages

    Lab – Flask information leakage

    Case study – Information leakage via errors in Apache Superset
- Exception handling
  - In the except block. And now what?
  - Empty except block

  Lab – Exception handling mess
- Control flow
  - Incorrect block delimitation
  - Dead code
  - Using if-then-else and switch defensively

Web application security beyond the Top Ten

- X01 - Lack of Application Resilience
  - Denial of service
  - Flooding
  - Resource exhaustion
  - Sustained client engagement
  - Infinite loop
  - Economic Denial of Sustainability (EDoS)
  - Amplification
    - Some amplification examples
  - Algorithmic complexity issues
    - Regular expression denial of service (ReDoS)

    Lab – ReDoS

    - Dealing with ReDoS

    Case study – ReDoS vulnerabilities in Python

    - Hash table collision
    - How do hash tables work?
    - Hash collision against hash tables

› Security testing

Security testing techniques and tools

- Code analysis
  - Static Application Security Testing (SAST)
  - *Lab – Using static analysis tools*
- Dynamic analysis
  - Security testing at runtime
  - Penetration testing
  - Stress testing
  - Dynamic analysis tools
    - Dynamic Application Security Testing (DAST)
    - Web vulnerability scanners
    - *Lab – Using web vulnerability scanners*
  - Fuzzing

› Wrap up

Secure coding principles

- Principles of robust programming by Matt Bishop

And now what?

- Software security sources and further reading
- Python resources