Security Testing Python Web Applications

Category: Networking & Security | **Vendor:** Various

Duration: 24.00 hours (3 days) **19.5 CPD Hours** **Rating:** ⭐ 4.6 (5,878 reviews)

Course Information

Delivery Format: Instructor Led - Online

Course Overview

Your Web application written in Python is tested functionally, 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 -231? Because that’s what the bad guys will do – and the list is far from complete.

Testing for security needs a remarkable software security expertise and 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.

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.A special focus is given to finding all discussed issues during testing, and an overview is provided on security testing methodology, techniques and tools.

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

So that nothing unexpected happens.

Nothing.

About This Course

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

Who Should Attend

Python developers and testers working on Web applications

Prerequisites & Entry Requirements

General Prerequisites:

General Python and Web development, testing and QA

Learning Outcomes

Upon successful completion of this course, participants will be able to:

- 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
- Understanding security testing methodology and approaches
- Getting familiar with common security testing techniques and tools
- Managing vulnerabilities in third party components
- Identify vulnerabilities and their consequences
- Learn the security best practices in Python
- Input validation approaches and principles

Detailed Course Outline

DAY 1

Cyber security basics

- What is security?
- Threat and risk
- Cyber security threat types
- Consequences of insecure software

- Constraints and the market
- The dark side

The OWASP Top Ten

- OWASP Top 10 – 2017
- A1 – 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
- Case study – Hacking Fortnite accounts
- Testing for SQL injection
- Code injection

- Code injection via input()
- OS command injection

- Lab – Command injection
- Case study – Shellshock
- Lab – Shellshock
- Case study – Command injection via ping
- Testing for command injection
- Script injection

- Server-side template injection (SSTI)
- Lab – Template injection
- A2 – Broken Authentication

- Authentication basics
- Multi-factor authentication

- Authentication weaknesses – spoofing

- Spoofing on the Web

- Testing for weak authentication

- Case study – PayPal 2FA bypass

- User interface best practices

- Lab – On-line password brute forcing

- 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

- Password length

- Password hardening

- Using passphrases

- Case study – The Ashley Madison data breach

- The dictionary attack

- The ultimate crack

- Exploitation and the lessons learned

- (Mis)handling None passwords

- Testing for password management issues

DAY 2

Security testing

- Security testing vs functional testing
- Manual and automated methods
- Security testing methodology

- Security testing – goals and methodologies
- Overview of security testing processes
- Identifying and rating assets

- Preparation
- Identifying assets
- Identifying the attack surface
- Assigning security requirements
- Lab – Identifying and rating assets
- Threat modeling

- SDL threat modeling
- Mapping STRIDE to DFD
- DFD example
- Attack trees
- Attack tree example
- Lab – Crafting an attack tree
- Misuse cases
- Misuse case examples
- Risk analysis
- Lab – Risk analysis
- Security testing approaches

- Reporting, recommendations, and review

The OWASP Top Ten

- A3 – Sensitive Data Exposure

- Information exposure
- Exposure through extracted data and aggregation
- Case study – Strava data exposure
- Error and exception handling principles
- Information exposure through error reporting
- Information leakage via error pages
- Lab – Flask information leakage
- 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
- A5 – Broken Access Control

- Access control basics
- Failure to restrict URL access
- Testing for authorization issues
- 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
- Testing for confused deputy weaknesses
- File upload

- Unrestricted file upload
- Good practices
- Lab – Unrestricted file upload
- Testing for file upload vulnerabilities
- A6 – Security Misconfiguration

- Configuration principles
- Configuration management
- Python configuration best practices

- Configuring Flask
- Testing for misconfiguration issues
- 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
- Additional protection layers
- Testing for XSS

DAY 3

The OWASP Top Ten

- A8 – Insecure Deserialization

- Serialization and deserialization challenges
- Deserializing untrusted streams
- Deserialization with pickle
- Lab – Deserializing with Pickle
- PyYAML deserialization challenges
- Testing for insecure deserialization
- 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
- Bug bounty programs
- Vulnerability databases
- Vulnerability rating – CVSS
- DevOps, the build process and CI / CD
- Dependency checking in Python
- Lab – Detecting vulnerable components
- A10 – Insufficient Logging & Monitoring

- Logging and monitoring principles
- Insufficient logging
- Plaintext passwords at Facebook

- Firewalls and Web Application Firewalls (WAF)
- Intrusion detection and prevention
- Case study – The Marriott Starwood data breach
- Web application security beyond the Top Ten

- Client-side security
- Lab – Client-side security
- Tabnabbing
- Lab – Reverse tabnabbing
- Frame sandboxing

- Cross-Frame Scripting (XFS) attack
- Lab – Clickjacking
- Clickjacking beyond hijacking a click
- Some further best practices

- HTML5 security best practices
- CSS security best practices
- Ajax security best practices

Security testing

- Security testing techniques and tools

- Code analysis

- Security aspects of code review
- 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
- SQL injection tools
- Lab – Using SQL injection tools
- Proxy servers
- Fuzzing

Common software security weaknesses

- Input validation

- Input validation principles

- Lab – Input validation
- 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

- Path traversal best practices

- Testing for path traversal

- Format string issues

- Unsafe native code

- Native code dependence

- Lab – Unsafe native code

- Best practices for dealing with native code

Wrap up

- And now what?

- Software security sources and further reading

- Python resources

- Security testing resources

Skills You Will Learn:

- 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
- Understanding security testing methodology and approaches
- Getting familiar with common security testing techniques and tools
- Managing vulnerabilities in third party components
- Identify vulnerabilities and their consequences
- Learn the security best practices in Python
- Input validation approaches and principles

Frequently Asked Questions

Q: What delivery options are available for Security Testing Python Web Applications?

We offer multiple delivery formats:

- Live Instructor-Led Classroom Online (Virtual/Live Online)
- Traditional Instructor-Led Classroom Training (ILT)
- On-site delivery at your offices anywhere in United Kingdom
- Private dedicated courses customized for your team

Q: How many CPD hours does this course provide?

The 3-day Security Testing Python Web Applications course provides up to 19.5 CPD hours of structured learning. CPD certificates can be provided upon request.

Q: What is the duration of the Security Testing Python Web Applications training?

The training takes place over 3 day(s), with each day lasting approximately 24.00 hours including breaks for lunch and refreshments.

Q: Do you provide corporate training for Security Testing Python Web Applications?

Yes, we provide corporate training, dedicated training, and closed classes for Security Testing Python Web Applications. Training can take place anywhere in United Kingdom including London, Manchester, Birmingham, Edinburgh, or live online allowing teams from across United Kingdom or internationally to attend.

Q: What related terms do people search for?

Popular related searches include: Security Testing Python Web Applications, Security Testing, SECT-PYWA

Q: Why choose Nexus Human for Security Testing Python Web Applications?

Nexus Human is recognized as one of the leading training providers. Our trainers have won multiple awards including:

- Small Firms Best Trainer Award
- National Training Partner of the Year (Ireland) - Multiple Years
- Global Top 30 Instructor Awards (2012, 2019, 2021)
- Tech Excellence Award Nominations
- Learning Performance Institute (LPI) External Training Provider Sponsor 2024

Q: Are there any discount codes available?

Yes! Use discount code **PENPAL5** when booking your Security Testing Python Web Applications training. Please note that only one discount code can be used per booking and cannot be combined with other special offers.

Nexus Human

Professional Training & Development

Email: info@nexushuman.com

Website: www.nexushuman.com

Phone: +353 1 889 8700 (Ireland) | +44 20 8142 8961 (UK)

© 2026 Nexus Human. All rights reserved. This brochure was generated on 19/10/2026.