Cloud Application Security in Python for AWS

Category: Networking & Security | **Vendor:** Amazon Web Services

Duration: 40.00 hours (5 days) **32.5 CPD Hours** **Rating:** ⭐ 4.6 (5,878 reviews)

Course Information

Delivery Format: Instructor Led - Online

Course Overview

Your cloud 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 -231? Because that's what the bad guys will do – and the list is far from complete.

The cloud has become a critical aspect of online services. No matter which model you're using (SaaS, PaaS, IaaS), part of your service is now operated by someone else. This may look like a net positive, but it also greatly expands the attack surface and brings in several new risks that may not be obvious. Have you configured all security settings correctly? Are you prepared for supply chain attacks? How can you protect the confidentiality of user data in the cloud? And most importantly: can the bad guys use your exposure to their advantage?

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 and the AWS cloud platform.

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
- Cloud infrastructure security
- API security
- XML security
- JSON security
- Denial of service
- Cryptography for developers
- Wrap up

Who Should Attend

Python developers working on Web applications and AWS

Prerequisites & Entry Requirements

General Prerequisites:

General Python and Web development

Learning Outcomes

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

- Getting familiar with essential cyber security concepts
- Understand cloud security specialties
- 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
- Learn to deal with cloud infrastructure security
- Identify vulnerabilities and their consequences
- Learn the security best practices in Python
- Input validation approaches and principles
- Understanding how cryptography can support application security
- Learning how to use cryptographic APIs correctly in Python

Detailed Course Outline

Day 1

Cyber security basics

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

- Cloud infrastructure basics

- Cloud architectures and security

- The Cloud Cube Model

- Attack surface in the cloud

- Cloud data security

- Data confidentiality and integrity in the cloud

- Data privacy in the cloud

- Compliance considerations

- Cloud deployment security

- Hardening cloud deployments

- Security of jump boxes

- Serverless computing and security

- Cloud security standards and best practices

- SOC compliance

- CSA controls

- Other standards

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
- NoSQL injection

- NoSQL injection specialties
- NoSQL injection in MongoDB
- NoSQL injection in DynamoDB
- SQL injection best practices

- Input validation
- Parameterized queries
- Lab – SQL injection best practices
- Additional considerations
- Case study – Hacking Fortnite accounts
- SQL injection protection and ORM
- Parameter manipulation

- CRLF injection

- Log forging
- Lab – Log forging
- Log forging – best practices
- HTTP response splitting
- 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 via ping
- Script injection

- Server-side template injection (SSTI)
- Lab – Template injection

Day 2

The OWASP Top Ten

- A2 - Broken Authentication

- Authentication

- Authentication basics
- Multi-factor authentication
- Multi-factor authentication best practices
- Authentication weaknesses - spoofing
- Spoofing on the Web
- Case study – PayPal 2FA bypass
- User interface best practices
- Lab – On-line password brute forcing
- Single sign-on (SSO)

- Single sign-on concept
- OAuth2

- OAuth2 basics
- OAuth2 in practice
- Best practices
- Configuration best practices

- Case study – Stealing SSO tokens from Epic Games accounts
- SAML

- SAML basics
- SAML profiles
- SAML security
- 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 hardening
- Using passphrases
- 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
- 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
- Why do we protect session IDs – Session hijacking
- Session fixation
- Session invalidation
- Session ID best practices
- Session handling in Flask
- Cross-site Request Forgery (CSRF)

- Lab – Cross-site Request Forgery
- CSRF best practices
- CSRF defense in depth
- Lab – CSRF protection with tokens
- Cookie security

- Cookie security best practices
- Cookie attributes
- A3 - Sensitive Data Exposure

- Information exposure
- Exposure through extracted data and aggregation
- Case study – Strava data exposure
- System information leakage

- Leaking system information
- Information exposure best practices
- 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

Day 3

The OWASP Top Ten

- A5 - Broken Access Control

- Access control basics

- Failure to restrict URL access

- Confused deputy

- Insecure direct object reference (IDOR)

- Lab – Insecure Direct Object Reference

- Case study – Authorization bypass on Facebook

- Authorization bypass through user-controlled keys

- Lab – Horizontal authorization

- File upload

- Unrestricted file upload

- Good practices

- Lab – Unrestricted file upload

- 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
- Integrity – deserializing untrusted streams
- Deserialization with pickle
- Lab – Deserializing with Pickle
- PyYAML deserialization challenges
- Integrity – deserialization best practices
- 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
- 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
- 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
- Frame sandboxing

- Cross-Frame Scripting (XFS) attack
- Lab - Clickjacking

- Clickjacking beyond hijacking a click
- Clickjacking protection best practices
- Lab – Using CSP to prevent clickjacking

Day 4

Cloud infrastructure security

- Container security

- Container security concerns
- Containerization, virtualization, and security
- Attack surface of container technologies
- Container security tools
- Docker security

- Docker and security
- Docker security features
- Common Docker security mistakes
- Docker security best practices
- Hardening Docker
- Lab – Static analysis of Docker image
- Kubernetes security

- The Kubernetes architecture and security
- Common Kubernetes security mistakes
- Securing Kubernetes hosts
- Best practices for Kubernetes access control
- Building secure Kubernetes images
- Secure deployment of Kubernetes containers
- Protecting Kubernetes deployments at runtime

- Lab – Scanning a Kubernetes image for vulnerabilities

- AWS security

- Security considerations

- AWS and security

- AWS security features

- The AWS shared responsibility model

- AWS cloud compliance

- AWS hardening

- Security tools for AWS

- Identity and access management (IAM)

- Identity and access management in AWS

- Access tokens

- Groups, roles and credentials

- Data security

- Data security in AWS

- Policies

- Storing cryptographic keys

- Protecting data at rest

- Protecting data in transit

- Detection and monitoring

- Utilizing AWS monitoring for security

- Protecting logs

- The AWS Security Hub

API security

- 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
- When to validate – validation vs transformations
- Output sanitization
- Encoding challenges
- Unicode challenges
- Lab – Encoding challenges
- Validation with regex
- Integer handling problems

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

- Setting manipulation
- Manipulating critical state data
- Resource manipulation
- Open redirects and forwards

- Case study – Unvalidated redirect at Epic Games
- Open redirects and forwards – best practices
- Files and streams

- Path traversal

- Path traversal-related examples

- Additional challenges in Windows

- Virtual resources

- Path traversal best practices

- Format string issues

XML security

- XML validation

- XML injection

- XPath injection

- Blind XPath injection

JSON security

- JSON validation

- JSON injection

- Dangers of JSONP

- JSON/JavaScript hijacking

- Best practices

- Case study – ReactJS vulnerability in HackerOne

Day 5

Denial of service

- Denial of Service

- Flooding
- Resource exhaustion
- Sustained client engagement
- Infinite loop
- Economic Denial of Sustainability (EDoS)
- Algorithm complexity issues

- Regular expression denial of service (ReDoS)

- Lab – ReDoS in Python
- Dealing with ReDoS

Cryptography for developers

- Cryptography basics
- Cryptography in Python
- Elementary algorithms

- Random number generation

- Pseudo random number generators (PRNGs)
- Cryptographically strong PRNGs
- Using virtual random streams
- Weak and strong PRNGs
- Using random numbers in Python
- Lab – Using random numbers in Python
- Case study – Equifax credit account freeze
- Hashing

- Hashing basics
- Common hashing mistakes

- Hashing in Python
- Lab – Hashing in Python
- Confidentiality protection

- Symmetric encryption

- Block ciphers
- Modes of operation
- Modes of operation and IV – best practices
- Symmetric encryption in Python
- Lab – Symmetric encryption in in Python
- Asymmetric encryption
- Combining symmetric and asymmetric algorithms
- Integrity protection

- Message Authentication Code (MAC)

- MAC in Python
- Lab – Calculating MAC in Python
- Digital signature

- Digital signature in Python
- Lab – Digital signature with ECDSA in Python
- Public Key Infrastructure (PKI)

- Some further key management challenges
- Certificates

- Chain of trust
- Certificate management – best practices
- Transport security

- Transport security weaknesses

- The TLS protocol

- TLS basics

- TLS features (changes in v1.3)

- The handshake in a nutshell (v1.3)

- TLS best practices

- TLS authentication best practices

- Lab – Using a secure socket in Python

- HTTP Strict Transport Security (HSTS)

- Lab – Setting HSTS in Python

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

Skills You Will Learn:

- Getting familiar with essential cyber security concepts
- Understand cloud security specialties
- 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
- Learn to deal with cloud infrastructure security
- Identify vulnerabilities and their consequences
- Learn the security best practices in Python
- Input validation approaches and principles
- Understanding how cryptography can support application security
- Learning how to use cryptographic APIs correctly in Python

Frequently Asked Questions

Q: What delivery options are available for Cloud Application Security in Python for AWS?

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 5-day Cloud Application Security in Python for AWS course provides up to 32.5 CPD hours of structured learning. CPD certificates can be provided upon request.

Q: What is the duration of the Cloud Application Security in Python for AWS training?

The training takes place over 5 day(s), with each day lasting approximately 40.00 hours including breaks for lunch and refreshments.

Q: Do you provide corporate training for Cloud Application Security in Python for AWS?

Yes, we provide corporate training, dedicated training, and closed classes for Cloud Application Security in Python for AWS. 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: Cloud Application Security in Python for AWS, Cloud Application Security, CASEC-PAWS

Q: Why choose Nexus Human for Cloud Application Security in Python for AWS?

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 Cloud Application Security in Python for AWS 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.