Web application security for PCI DSS

Customized for PCI DSS requirement 6.3 compliance

CYDWeb_PCIDSS | 1+2 days | On-site or online | Hands-on

Your application written in any programming language 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.

PCI DSS is a mandatory security standard for all companies developing or working with systems that handle credit cards. It does not only require following the secure coding guidelines out there, but also requires developers to train themselves on the latest best practices. But ticking the box annually is not enough.

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 Java, and extended by core programming issues, discussing security pitfalls of the Java language.

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

So that nothing unexpected happens.

Nothing.

Note: This course is customized for PCI DSS requirement 6.3 compliance, concerning both the content and the delivery structure.

The course covers essential secure coding skills that are a must for all developers working with cardholder data and brings in a number of case studies from the financial sector.

Aligned to the compliance requirements, the delivery of the training days can be done separately, breaking the course into two separate events that can span across year boundaries, aligned to your long-term compliance plans.

Important: delivery of the 1-day initial plenary session can only be organized bundled (having at least two sessions batched).

Cyber security skills and drills


26 LABS 15 CASE STUDIES

Audience

Managers and developers working on Web applications in finance

Group size

Plenary: 30, Classes: 12 participants

Preparedness

None for plenary, general Web development for secure coding

Outline

- Cyber security basics
- The OWASP Top Ten 2021
- Security testing
- Wrap up

Standards and references

OWASP, CWE and Fortify Taxonomy

What you'll have learned

- Getting familiar with essential cyber security concepts
- Learning about security specialties of the finance sector
- Having essential understanding of PCI DSS requirements
- Managing vulnerabilities in third party components
- Understanding Web application security issues
- Detailed analysis of the OWASP Top Ten elements
- Putting Web application security in the context of any programming language
- Going beyond the low hanging fruits
- Understanding how cryptography supports security
- Getting familiar with security testing techniques and tools

Table of contents

- Least common mechanism
- Psychological acceptability

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

A05 - Security Misconfiguration

- Configuration principles
- Server misconfiguration
- 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*
      - *Case study – XXE vulnerability in SAP Store*
    - *Lab – Prohibiting DTD*

A06 - Vulnerable and Outdated Components

- Using vulnerable components
- *Case study – The Equifax data breach*
- Assessing the environment
- Hardening
- Untrusted functionality import
- Vulnerability management
  - Patch management
  - Vulnerability management
  - Vulnerability databases
  - DevOps, the build process and CI / CD

A09 - Security Logging and Monitoring Failures

- Logging and monitoring principles

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

Day 2 - Web application security day

› The OWASP Top Ten 2021

A01 - 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*
- Cross-site Request Forgery (CSRF)
  - *Lab – Cross-site Request Forgery*
  - CSRF best practices
  - *Lab – CSRF protection with tokens*

A02 - Cryptographic Failures

- Information exposure
  - Exposure through extracted data and aggregation
  - *Case study – Strava data exposure*
  - System information leakage
    - Leaking system information
  - Information exposure best practices
- Cryptography for developers
  - Cryptography basics
  - Elementary algorithms
    - Random number generation
    - Pseudo random number generators (PRNGs)
    - Cryptographically strong PRNGs

- Using virtual random streams
- *Lab – Using random numbers*
  - *Case study – Equifax credit account freeze*
- Confidentiality protection
  - Symmetric encryption
  - Block ciphers
  - Modes of operation
  - Modes of operation and IV – best practices
  - *Lab – Symmetric encryption*
  - Asymmetric encryption
  - Combining symmetric and asymmetric algorithms

› The OWASP Top Ten 2021

A03 - 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 – Hacking Fortnite accounts*
- Code injection
  - OS command injection
    - OS command injection best practices
      - *Case study – Shellshock*
    - *Lab - Shellshock*

Day 3 - Web application security day

› The OWASP Top Ten 2021

A03 - 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 in Fortnite accounts*
- XSS protection best practices
  - Protection principles - escaping
  - *Lab – XSS fix / stored*
  - *Lab – XSS fix / reflected*
  - Additional protection layers – defense in depth

› The OWASP Top Ten 2021

A07 - Identification and Authentication Failures

- Authentication
  - Authentication basics
  - Multi-factor authentication
  - Time-based One Time Passwords (TOTP)
  - Authentication weaknesses
  - *Case study – Equifax Argentina*
  - Spoofing on the Web
  - *Case study – PayPal 2FA bypass*
  - User interface best practices
  - *Case study – Information disclosure in Simple Banking for Android*
  - *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 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 null passwords

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

A08 - Software and Data Integrity Failures

- Subresource integrity
  - Importing JavaScript
  - *Lab – Importing JavaScript*
  - *Case study – The British Airways data breach*
- Insecure deserialization
  - Serialization and deserialization challenges
  - Integrity – deserializing untrusted streams
  - Integrity – deserialization best practices
  - Property Oriented Programming (POP)
    - Creating a POP payload
    - *Lab – Creating a POP payload*
    - *Lab – Using the POP payload*
    - Summary – POP best practices

› Security testing

Security testing techniques and tools

- Code analysis
  - Static Application Security Testing (SAST)
- 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*
  - Fuzzing

› Wrap up

Secure coding principles

- Principles of robust programming by Matt Bishop

And now what?

- Software security sources and further reading