PHP201 – DEFENDING PHP

Course Learning Objectives

Defending PHP prepares developers to harden PHP applications and defend against attacks. In ten modules, this course covers handling requests, validating input data, preventing parameter manipulation, avoiding unchecked redirects, controlling application access, managing sessions, preventing cross-site request forgeries, impeding SQL injections, restricting low-level commands and file access, securing data storage, and preventing cross-site scripting attacks.

Description

This course has been developed for PHP developers and web application architects who want to defend against common security vulnerabilities found in PHP applications and have completed OWASP Top 10 as a prerequisite.

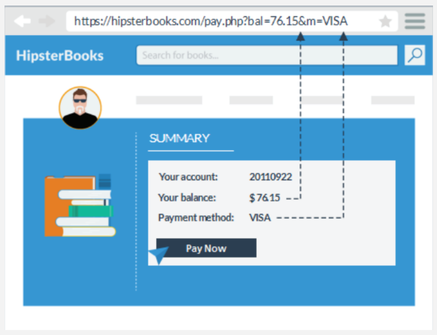
Audience

PHP Developers

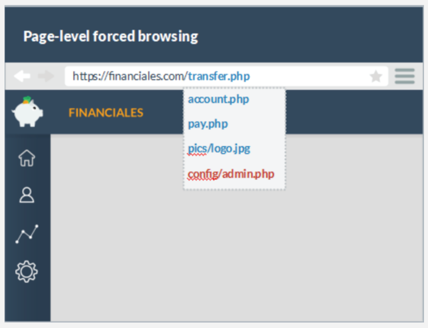
Web App Architects

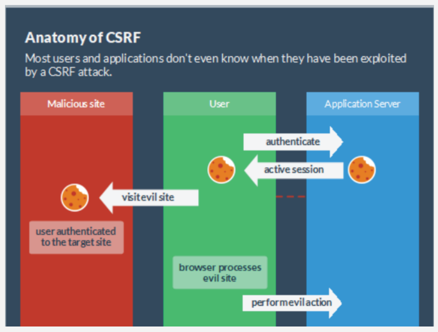
Time Required

Tailored learning - 105 minutes total



Your account:	20110922
Your balance:	$ 76.15
Payment method:	VISA


Page-level forced browsing
https://financiales.com/transfer.php
FINANCIALES
account.php
pay.php
pics/logo.jpg
config/admin.php


Anatomy of CSRF
Most users and applications don't even know when they have been exploited by a CSRF attack.
Malicious site
User
Application Server
authenticate
active session
visit evil site
user authenticated to the target site
browser processes evil site
perform evil action

PHP201 – DEFENDING PHP

Course Outline

1. Handling Requests

- Untrusted incoming requests
- Handling requests defense
- Creating a validation library
- Checking the HTTP method
- Checking the Content Type
- Enforcing checks
- Input validation
- Validating integers
- Best practices for input validation
- Validating email addresses
- Validating complex data
- Maximize centralization
- Getting string parameters
- Getting text parameters
- Getting integer parameters

2. Parameter Manipulation

- Vulnerability of parameter manipulation
- Parameter data in an app
- Application logic
- Defense: Limiting Parameter Exposure
- Matching parameters against approved values
- Coding example
- Coding example using PHP 8
- Validating parameters using regular expressions
- Preg_match() example
- Regex denial of service (ReDos)
- Validating integers

3. Unchecked redirects

- Lack of URL validation
- Redirecting to the login page
- HTTP header directs overview
- Avoid exposing the redirect location
- Indirectly mapping endpoints
- Validating a redirect URL
- Validating a redirect URL - Code
- The header() function

4. Access Control

- Page-level forced browsing
- Defense against Page-level level access control
- Action-level authorization
- Defense of action-level access control
- Object-level authorization vulnerability
- Object-level authorization defense
- Forced browsing summary
- Defining an access control matrix
- Translating matrix to database
- Access control summary
- Tracking user-specific authorization data
- Checking requested permissions
- Checking user authorization
- Implementing object-level authorization

5. Session Management

- Guessable session identifiers
- Defense against guessable session identifiers
- Session hijacking and defense
- Session fixation
- Renewing the session identifier
- Configuring PHP’s cookie properties
- Built-in session management mechanism
- Starting and ending a session
- Garbage collection
- PHP API: session activity timeouts
- PHP API: programmatic timeout

6. Cross-site Request Forgery

- Anatomy of CSRF
- Where does CSRF matter?
- Anti-CSRF tokens
- Anti-CSRF token steps
- Generating the random token
- Creating and assigning token to form
- Checking that the token matches
- Shorter session timeouts
- Re-authentication

Security Compass

PHP201 – DEFENDING PHP

Course Outline

7. SQL Injection

- Retrieving information
- What can go wrong?
- The UNION SELECT attack
- Parametrized queries
- Binding variables with PDO
- Using named placeholders with PDO
- Binding variables with MySQLi
- The limits of parametrization

8. Insecure Commands and Files

- Command injection vulnerability
- Application attack
- Defense against command injection
- Best practices for defending against command injection
- Path traversal manipulation
- Defense against path traversal manipulation
- Best practices for defending against path traversal
- Insecure file upload
- Securing file uploads
- Storing uploaded files
- Serving uploaded files
- Serving uploaded files through a proxy server
- Best practices for defending against insecure file upload

9. Insecure Storage

- Insecure storage of sensitive data
- Third-party authentication solution
- Storing passwords
- Password storage in PHP
- Verifying passwords against a hash
- Cryptography in PHP
- Data encryption in PHP
- Data decryption in PHP

10. Cross-site Scripting (XSS)

- Attacker input
- XSS in login
- Repeating username
- Non-sanitized input
- Relying on input validation
- Defense best practices
- Encoding for HTML context
- XSS defenses: PHP library
- Encoding for HTML attribute
- Encoding for URLs
- More advanced encoding
- Output encoding practices
- Sanitization
- Sanitization example
- Using a sanitization library

SecurityCompass