

OWASP Secure Coding Practices – EXAM CHEAT SHEET

(Based on OWASP Secure Coding Practices – Quick Reference Guide v2.x)



What is OWASP?

OWASP (Open Web Application Security Project) is an open-source organization that provides standards, tools, and guidelines to improve web application security.

Exam One-liner:

OWASP provides secure coding guidelines to protect web applications from common security vulnerabilities.



Goal of Secure Coding (WHY OWASP?)

- Prevent hacking and data breaches
 - Protect user data
 - Reduce cost of fixing security bugs later
 - Build secure software from the start
-



CIA TRIAD (VERY IMPORTANT)

1. Confidentiality

- Data visible only to authorized users
- Example: passwords, CNIC, bank details

2. Integrity

- Data should not be modified without permission
- Example: marks, balances

3. Availability

- System should be accessible when needed
- Example: website should not crash

Mnemonic: CIA

Risk Concept (EXAM FAVORITE)

Risk = Threat + Vulnerability + Impact

- Threat: attacker/hacker
- Vulnerability: weakness in system
- Impact: damage or loss

Car Example: Thief (threat) + unlocked car (vulnerability) = stolen items (impact)



Client-Side Controls Are NOT Secure

- JavaScript validation can be bypassed
- Hidden fields are not safe
- Always validate on **SERVER SIDE**



OWASP SECURE CODING PRACTICES



Input Validation

Meaning: Verify all user input before processing

- ✓ Validate on server side
- ✓ Identify trusted vs untrusted input
- ✓ Validate type, length, range
- ✓ Use whitelist (allowed characters)
- ✓ Reject invalid input

Protects Against: SQL Injection, XSS, Path Traversal

Exam Line: Input validation ensures only safe and expected data is processed.



Output Encoding

Meaning: Make output safe before sending to user

- ✓ Encode untrusted data
- ✓ Context-based encoding (HTML, JS, URL, SQL)
- ✓ Prevent script execution

Protects Against: Cross Site Scripting (XSS)

Exam Line: Output encoding prevents malicious code execution in browsers.

Authentication & Password Management

- ✓ Authentication for all protected pages
- ✓ Strong password policy
- ✓ Store passwords as **hashed & salted**
- ✓ Same error message for login failure
- ✓ Account lock after failed attempts
- ✓ Use MFA for sensitive systems

- ✗ No plain text passwords
- ✗ No MD5

Exam Line: Authentication verifies user identity securely.

Session Management

- ✓ Secure, random session IDs
- ✓ Session timeout
- ✓ Logout destroys session
- ✓ No session IDs in URLs
- ✓ Use Secure & HttpOnly cookies

Protects Against: Session Hijacking, CSRF

Exam Line: Session management ensures secure user sessions.

Access Control

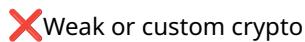
- ✓ Role-based access (Admin ≠ User)
- ✓ Check authorization on every request
- ✓ Deny by default
- ✓ Protect URLs, functions, data

Protects Against: Unauthorized access

Exam Line: Access control restricts users to authorized resources only.

Cryptographic Practices

- ✓ Strong encryption algorithms
- ✓ Secure key management
- ✓ Cryptographically secure random numbers
- ✓ Protect master secrets



Weak or custom crypto

Exam Line: Cryptography protects sensitive data using encryption.

Error Handling & Logging

- ✓ Generic error messages
- ✓ No stack traces
- ✓ Log security events
- ✓ Logs on server side
- ✓ No sensitive data in logs

Exam Line: Proper error handling prevents information disclosure.

Data Protection

- ✓ Least privilege principle
- ✓ Encrypt sensitive stored data
- ✓ Disable browser caching
- ✓ No sensitive data in URLs
- ✓ Remove sensitive comments

Exam Line: Data protection ensures secure storage of sensitive information.

Communication Security

- ✓ Use HTTPS / TLS
- ✓ Valid certificates
- ✓ No fallback to HTTP
- ✓ Encrypt sensitive transmissions

Exam Line: Communication security protects data during transmission.

System Configuration

- ✓ Patch systems regularly
- ✓ Remove unused services
- ✓ Disable directory listing
- ✓ Least privilege for services
- ✓ Hide server information

Exam Line: Secure configuration reduces attack surface.



Database Security

- ✓ Prepared statements (parameterized queries)
- ✓ Least privilege DB accounts
- ✓ Encrypted connection strings
- ✓ Remove default DB passwords

Protects Against: SQL Injection

Exam Line: Database security prevents unauthorized database access.



File Management

- ✓ Restrict file uploads
- ✓ Validate file type (not only extension)
- ✓ No executable uploads
- ✓ Store files outside web root
- ✓ Virus scanning

Exam Line: File management prevents malicious file uploads.



Memory Management

- ✓ Avoid buffer overflow
- ✓ Proper memory allocation
- ✓ Free memory correctly
- ✓ Avoid unsafe functions

Exam Line: Memory management prevents buffer overflow vulnerabilities.



General Coding Practices

- ✓ Use trusted libraries
- ✓ Avoid dynamic code execution
- ✓ Secure updates
- ✓ Protect shared resources
- ✓ Review third-party code

ONE-SHOT REVISION LINE

OWASP Secure Coding Practices help developers prevent common web vulnerabilities such as SQL Injection, XSS, broken authentication, and insecure session management.

 **EXAM TIP**

- Learn **headings + 1 line explanation**
 - CIA Triad + Risk formula = guaranteed marks
 - Write examples if possible
-

You can print this as **final exam notes**.