# **OWASP WSTG v4.2 Penetration Testing Checklist**

**Source:** OWASP Web Security Testing Guide v4.2

This checklist is designed for tracking tasks during a web application security assessment. The format is suitable for direct use in Microsoft Word or other document editors.

### **1. Information Gathering**

* [ ] **Search Engine Discovery:** Conduct reconnaissance for information leakage.
* [ ] **Fingerprint Web Server:** Identify web server type and version.
* [ ] **Review Metafiles:** Check robots.txt, sitemap.xml, etc., for sensitive information.
* [ ] **Enumerate Applications:** Identify all applications on the server.
* [ ] **Review Webpage Content:** Analyze comments, metadata, and scripts for leaked information.
* [ ] **Identify Application Entry Points:** Map all user-controllable inputs.
* [ ] **Map Execution Paths:** Understand the application's flow and logic.
* [ ] **Fingerprint Web Application Framework:** Identify the framework (e.g., React, Angular, Django).
* [ ] **Fingerprint Web Application:** Identify the application itself.
* [ ] **Map Application Architecture:** Diagram the application's components and data flow.

### **2. Configuration & Deployment Management**

* [ ] **Test Infrastructure Configuration:** Check for misconfigurations in the network and hosting environment.
* [ ] **Test Application Platform Configuration:** Check for platform-specific vulnerabilities.
* [ ] **Test File Extension Handling:** Determine how the server handles unexpected file extensions.
* [ ] **Review Backup Files:** Search for old, backup, or unreferenced files containing sensitive data.
* [ ] **Enumerate Admin Interfaces:** Discover administrative portals.
* [ ] **Test HTTP Methods:** Test for dangerous or unnecessary HTTP methods (e.g., PUT, DELETE).
* [ ] **Test HSTS:** Verify HTTP Strict Transport Security is correctly implemented.
* [ ] **Test RIA Cross-Domain Policy:** Check policies for technologies like Flash or Silverlight.
* [ ] **Test File Permissions:** Look for overly permissive file and directory permissions.
* [ ] **Test for Subdomain Takeover:** Check for dangling DNS records pointing to third-party services.
* [ ] **Test Cloud Storage:** Check for publicly accessible or misconfigured cloud buckets (S3, etc.).

### **3. Identity Management**

* [ ] **Test Role Definitions:** Analyze roles and permissions for potential privilege escalation.
* [ ] **Test User Registration Process:** Look for flaws in the account creation process.
* [ ] **Test Account Provisioning:** Review how accounts are created and managed.
* [ ] **Test for Account Enumeration:** Check if valid usernames can be discovered.
* [ ] **Test for Weak Username Policy:** Check for lack of restrictions on username format.

### **4. Authentication**

* [ ] **Test Credentials Over Encrypted Channel:** Ensure all credentials are sent over HTTPS.
* [ ] **Test for Default Credentials:** Check for default usernames and passwords.
* [ ] **Test Lockout Mechanism:** Attempt to brute-force credentials and test account lockout.
* [ ] **Test for Bypassing Authentication:** Attempt to access authenticated pages without logging in.
* [ ] **Test "Remember Me" Functionality:** Check for vulnerabilities in persistent login features.
* [ ] **Test Browser Cache Weaknesses:** Look for sensitive data in the browser's cache.
* [ ] **Test for Weak Password Policy:** Verify the application enforces strong passwords.
* [ ] **Test for Weak Security Questions:** Check if security questions are easy to guess.
* [ ] **Test Password Reset Functionality:** Look for flaws in the password reset and change process.
* [ ] **Test for Weaker Auth in Alternative Channels:** Check if other channels (e.g., mobile) have weaker authentication.

### **5. Authorization**

* [ ] **Test for Directory Traversal / File Inclusion:** Attempt to access files outside the web root.
* [ ] **Test for Bypassing Authorization:** Attempt to access functions as an unauthorized user.
* [ ] **Test for Privilege Escalation:** Attempt to gain higher privileges than intended.
* [ ] **Test for Insecure Direct Object References (IDOR):** Attempt to access other users' data by changing ID parameters.

### **6. Session Management**

* [ ] **Test Session Management Schema:** Analyze the overall session management implementation.
* [ ] **Test Cookie Attributes:** Check for Secure, HttpOnly, and SameSite flags on session cookies.
* [ ] **Test for Session Fixation:** Attempt to force a user to use a known session ID.
* [ ] **Test for Exposed Session Variables:** Look for session tokens in URLs or logs.
* [ ] **Test for Cross-Site Request Forgery (CSRF):** Check if state-changing actions are protected from CSRF.
* [ ] **Test Logout Functionality:** Ensure logging out properly invalidates the session.
* [ ] **Test Session Timeout:** Verify that sessions expire after a period of inactivity.
* [ ] **Test for Session Puzzling:** Test for logic flaws in state management within a session.
* [ ] **Test for Session Hijacking:** Attempt to steal and use a valid user's session.

### **7. Input Validation**

* [ ] **Test for Reflected Cross-Site Scripting (XSS)**
* [ ] **Test for Stored Cross-Site Scripting (XSS)**
* [ ] **Test for DOM-Based Cross-Site Scripting (XSS)**
* [ ] **Test for HTTP Verb Tampering**
* [ ] **Test for HTTP Parameter Pollution**
* [ ] **Test for SQL Injection** (Oracle, MySQL, SQL Server, PostgreSQL, MS Access, NoSQL, ORM)
* [ ] **Test for LDAP Injection**
* [ ] **Test for XML Injection (XXE)**
* [ ] **Test for Server-Side Includes (SSI) Injection**
* [ ] **Test for XPath Injection**
* [ ] **Test for IMAP/SMTP Injection**
* [ ] **Test for Code Injection** (including LFI/RFI)
* [ ] **Test for OS Command Injection**
* [ ] **Test for Format String Injection**
* [ ] **Test for HTTP Splitting/Smuggling**
* [ ] **Test for Host Header Injection**
* [ ] **Test for Server-Side Template Injection (SSTI)**
* [ ] **Test for Server-Side Request Forgery (SSRF)**

### **8. Error Handling**

* [ ] **Test for Improper Error Handling:** Trigger errors to check for information leakage.
* [ ] **Test for Stack Traces:** Check if detailed debug information and stack traces are exposed to users.

### **9. Weak Cryptography**

* [ ] **Test for Weak Transport Layer Security (TLS):** Check for outdated protocols and weak cipher suites.
* [ ] **Test for Padding Oracle Vulnerabilities**
* [ ] **Test for Sensitive Information Sent Unencrypted**
* [ ] **Test for Weak Encryption Algorithms:** Check for weak or custom cryptographic implementations.

### **10. Business Logic**

* [ ] **Test Business Logic Data Validation:** Look for flaws in server-side validation against business rules.
* [ ] **Test Ability to Forge Requests:** Manipulate requests to exploit business logic.
* [ ] **Test Integrity Checks:** Test for weaknesses in data integrity validation.
* [ ] **Test for Process Timing Flaws:** Exploit timing differences to infer information.
* [ ] **Test Function Use Limits:** Attempt to bypass limits on how many times a function can be used.
* [ ] **Test for Workflow Circumvention:** Attempt to skip or repeat steps in a defined business process.
* [ ] **Test Defenses Against Application Misuse:** Identify ways to use application features for unintended purposes.
* [ ] **Test Upload of Unexpected File Types**
* [ ] **Test Upload of Malicious Files**

### **11. Client-Side Testing**

* [ ] **Test for DOM-Based XSS**
* [ ] **Test for JavaScript Execution Flaws**
* [ ] **Test for HTML Injection**
* [ ] **Test for Client-side URL Redirects**
* [ ] **Test for CSS Injection**
* [ ] **Test for Client-side Resource Manipulation**
* [ ] **Test Cross-Origin Resource Sharing (CORS) Policy**
* [ ] **Test for Cross-Site Flashing**
* [ ] **Test for Clickjacking**
* [ ] **Test WebSockets Security**
* [ ] **Test Web Messaging Security (postMessage)**
* [ ] **Test Browser Storage Security** (Local Storage, Session Storage)
* [ ] **Test for Cross-Site Script Inclusion**

### **12. API Testing**

* [ ] **Test GraphQL API:** Check for introspection queries, batching attacks, and other GraphQL-specific vulnerabilities.