

Intro to Advanced Web Application Penetration Testing

Alexis Ahmed

Offensive Security/Red Team Instructor @INE

Senior Pentester & Red Team Lead @HackerSploit

MAJOR TOPICS

- + Web Application Pentesting Methodology
- + Planning Web App Pentests
- + Pre-Engagement Phase
- + Web App Mapping & Crawling
- + Reconnaissance
- + Session Security



LEARNING OUTCOMES

- + Solid understanding of the web app pentesting methodology
- + Ability to plan and orchestrate a web app pentest
- + Ability to perform web app reconnaissance
- + Ability to map a web application through techniques like crawling
- + Solid understanding of session management & security

PREREQUISITES

- + Good understanding of the web & the HTTP protocol
- + Experience with web proxies like Burp or ZAP
- + Familiarity with Linux command line tools

LET'S GO!

The primary objective of this course is to introduce you to the process of planning, orchestrating and managing a professional web application security assessment/penetration test.

- + This course seeks to demonstrate how to professionally plan and orchestrate a web application penetration test.
- + You will also learn how to streamline your activities/actions and improve the efficiency of your pentests while maintaining technical accuracy and rigor.



Web App Pentesting Methodology

Web App Pentesting

- Web application penetration testing, is a comprehensive security assessment aimed at identifying vulnerabilities and weaknesses in web applications.
- It involves simulating real-world attacks to evaluate the application's security posture and provide recommendations for remediation.
- Using a methodology for web application penetration testing is vitally important as it provides a structured and systematic approach to conduct thorough security assessments.

Importance of Web App Pentesting Methodologies

- Consistency and Standardization - A methodology ensures that penetration tests are performed consistently across different web applications and projects. It provides standardized procedures, tools, and techniques, ensuring that all necessary areas of the application's security are thoroughly tested.
- Comprehensive Coverage - A well-defined methodology helps ensure comprehensive coverage of the web application's security. It guides testers to assess all critical components and functionalities, reducing the risk of overlooking crucial security flaws.
- Efficiency and Time Management - Following a methodology streamlines the testing process, making it more efficient and time-effective. Testers can prioritize tasks, focus on high-risk areas, and avoid wasting time on redundant activities.

Importance of Web App Pentesting Methodologies

- Thorough Identification of Vulnerabilities - A methodology encourages detailed testing and comprehensive assessments. It helps testers identify both common and rare vulnerabilities, improving the overall security posture of the web application.
- Risk Prioritization - A methodology allows testers to assign risk levels to identified vulnerabilities based on their potential impact on the application and business. This assists stakeholders in prioritizing and addressing critical issues first.
- Effective Reporting and Communication - A structured methodology encourages well-documented reports that are easy to understand for both technical and non-technical stakeholders. Clear communication of findings and recommendations is vital for remediation efforts.

Importance of Web App Pentesting Methodologies

- Industry Standards and Best Practices - Many methodologies are based on industry standards and best practices, such as those provided by organizations like OWASP and NIST. Following established guidelines ensures a comprehensive and relevant assessment.
- Legal and Ethical Compliance - A methodology helps ensure that penetration testing is conducted in an ethical and lawful manner, respecting the rules of engagement and gaining proper authorization from the target organization.
- Detection of Complex Vulnerabilities - Advanced security vulnerabilities often require a structured approach to be identified. A methodology enables testers to apply specialized techniques and tools to detect complex vulnerabilities that might be missed in ad-hoc testing.

Web App Pentesting Methodology

- Web application penetration testing methodology is a structured and systematic approach to conducting security assessments of web applications.
- It helps identify potential vulnerabilities and weaknesses, assesses the application's security posture, and provides actionable recommendations for remediation.
- While specific methodologies may vary, the following steps are commonly included in a web application penetration testing methodology:

Web App Pentesting Methodology

	Phase	Objectives
1	Pre-Engagement	<ul style="list-style-type: none">• Define the scope and objectives of the penetration test, including the target web application, URLs, and functionalities to be tested.• Obtain proper authorization and permission from the application owner to conduct the test.• Gather relevant information about the application, such as technologies used, user roles, and business-critical functionalities.
2	Information Gathering & Reconnaissance	<ul style="list-style-type: none">• Perform passive reconnaissance to gather publicly available information about the application and its infrastructure.• Enumerate subdomains, directories, and files to discover hidden or sensitive content.• Use tools like "Nmap" to identify open ports and services running on the web server.• Utilize "Google Dorks" to find indexed information, files, and directories on the target website.
3	Threat Modeling	<ul style="list-style-type: none">• Analyze the application's architecture and data flow to identify potential threats and attack vectors.• Build an attack surface model to understand how attackers can interact with the application.• Identify potential high-risk areas and prioritize testing efforts accordingly.
4	Vulnerability Scanning	<ul style="list-style-type: none">• Use automated web vulnerability scanners like "Burp Suite" or "OWASP ZAP" to identify common security flaws.• Verify and validate the scan results manually to eliminate false positives and false negatives.

Web App Pentesting Methodology

	Phase	Objectives
5	Manual Testing & Exploitation	<ul style="list-style-type: none">• Perform manual testing to validate and exploit identified vulnerabilities in the application.• Test for input validation issues, authentication bypass, authorization flaws, and business logic vulnerabilities.• Attempt to exploit security flaws to demonstrate their impact and potential risk to the application.
6	Authentication & Authorization Testing	<ul style="list-style-type: none">• Test the application's authentication mechanisms to identify weaknesses in password policies, session management, and account lockout procedures.• Evaluate the application's access controls to ensure that unauthorized users cannot access sensitive functionalities or data.
7	Session Management Testing	<ul style="list-style-type: none">• Evaluate the application's session management mechanisms to prevent session fixation, session hijacking, and session-related attacks.• Check for session timeout settings and proper session token handling.
8	Information Disclosure	<ul style="list-style-type: none">• Review how the application handles sensitive information such as passwords, user data, and confidential files.• Test for information disclosure through error messages, server responses, or improper access controls.

Web App Pentesting Methodology

	Phase	Objectives
9	Business Logic Testing	<ul style="list-style-type: none">• Analyze the application's business logic to identify flaws that could lead to unauthorized access or data manipulation.• Test for order-related vulnerabilities, privilege escalation, and other business logic flaws.
10	Client-Side Testing	<ul style="list-style-type: none">• Evaluate the client-side code (HTML, JavaScript) for potential security vulnerabilities, such as DOM-based XSS.• Test for insecure client-side storage and sensitive data exposure.
11	Reporting & Remediation	<ul style="list-style-type: none">• Document and prioritize the identified security vulnerabilities and risks.• Provide a detailed report to developers and stakeholders, including recommendations for remediation.• Assist developers in fixing the identified security issues and retesting the application to ensure that the fixes were successful.
12	Post-Engagement	<ul style="list-style-type: none">• Conduct a post-engagement meeting to discuss the test results with stakeholders.• Provide security awareness training to the development team to promote secure coding practices.

Penetration Testing Methodologies

- There are several web application penetration testing methodologies that security professionals and organizations can follow to conduct comprehensive and structured assessments.
- Each methodology has its approach and focus areas, but they all share the common goal of identifying and mitigating security vulnerabilities in web applications.
- Here are some popular web application penetration testing methodologies:

Penetration Testing Execution Standard

- PTES is a complete penetration testing methodology that covers all aspects of security assessments, including web application testing.
- It provides a structured approach from pre-engagement through reporting and follow-up, making it suitable for comprehensive assessments.

penetration-testing-
execution-standard/**ptes**

The Penetration Testing Execution Standard (PTES)
Automation Framework



2

Contributors

4

Issues

17

Stars

8

Forks



OWASP Web Security Testing Guide (WSTG)

- The OWASP Web Security Testing Guide (WSTG) is a comprehensive resource provided by the Open Web Application Security Project (OWASP).
- It offers a structured methodology for performing web application security testing.





OWASP Top 10

OWASP Top 10

- The OWASP Top 10 is a regularly updated list of the most critical web application security risks.
- It is maintained by the Open Web Application Security Project (OWASP), a nonprofit organization focused on improving web application security.
- The OWASP Top 10 serves as a valuable guide for developers, web app pentesters, and organizations to understand and prioritize common security risks in web applications.

OWASP Top 10 Releases

- The OWASP Top 10 is a well-known list of the ten most critical web application security risks.
- It undergoes periodic updates to ensure it reflects the current threat landscape and the evolving security challenges faced by web applications.
- The first version of the OWASP Top 10 was released in 2003. It aimed to raise awareness about common web application security risks and help developers prioritize security efforts.
- The list included risks like Cross-Site Scripting (XSS), SQL Injection, and Session Management issues.
- Each release of the OWASP Top 10 builds upon the previous versions, improving its accuracy, relevance, and practicality.



Demo: OWASP Top 10



Web Application Security Testing

Web Application Security Testing

- Web application security testing is the process of evaluating and assessing the security aspects of web applications to identify vulnerabilities, weaknesses, and potential security risks.
- It involves conducting various tests and assessments to ensure that web applications are resistant to security threats and can effectively protect sensitive data and functionalities from unauthorized access or malicious activities.
- The primary goal of web application security testing is to uncover security flaws before they are exploited by attackers.
- By identifying and addressing vulnerabilities, organizations can enhance the overall security posture of their web applications, reduce the risk of data breaches and unauthorized access, and protect their users and sensitive information.

Web Application Security Testing Types

- Web application security testing typically involves a combination of automated scanning tools and manual testing techniques.
- Some common types of security testing conducted on web applications include:
 - Vulnerability Scanning: Using automated tools to scan the web application for known vulnerabilities, such as SQL injection, Cross-Site Scripting (XSS), insecure configurations, and outdated software versions.
 - Penetration Testing: Simulating real-world attacks to assess the application's defenses and identify potential security weaknesses. This involves ethical hacking to gain insights into how an attacker might exploit vulnerabilities.
 - Code Review and Static Analysis: Manual examination of the application's source code to identify coding flaws, security misconfigurations, and potential security risks.

Web Application Security Testing Types

- Authentication and Authorization Testing: Evaluating the effectiveness of authentication mechanisms and access control features to ensure that only authorized users have appropriate access levels.
- Input Validation and Output Encoding Testing: Assessing how the application handles user inputs to prevent common security vulnerabilities like XSS and SQL injection.
- Session Management Testing: Verifying how the application manages user sessions and related tokens to prevent session-related attacks.
- API Security Testing: Assessing the security of APIs (Application Programming Interfaces) used by the web application for data exchange and integration with other systems.

Web Application Penetration Testing

- Web application pentesting, is a subset of web application security testing that specifically involves attempting to exploit identified vulnerabilities.
- It is a simulated attack on the web application conducted by skilled security professionals known as pentesters, bug bounty hunters or ethical hackers.
- The process involves a systematic and controlled approach to assess the application's security by attempting to exploit known vulnerabilities.

Web App Pentesting vs Web App Security Testing

- Key differences between web app security testing and web app pentesting:
 - **Scope:** Web application security testing covers a broader range of assessments, including static and dynamic analysis, while web application pentesting focuses on actively exploiting vulnerabilities.
 - **Objective:** The primary goal of security testing is to identify weaknesses, whereas pentesting aims to validate vulnerabilities and assess the organization's ability to detect and respond to attacks.
 - **Methodology:** Security testing includes both manual and automated techniques, while pentesting is predominantly a manual process, involving the use of various tools and techniques for exploitation.
 - **Exploitation:** Security testing does not involve exploitation of vulnerabilities, while pentesting does, albeit in a controlled and authorized manner.

Web App Pentesting vs Web App Security Testing

Aspect	Web App Security Testing	Web App Pentesting
Objective	Identify vulnerabilities and weaknesses in the web application without actively exploiting them.	Actively attempt to exploit identified vulnerabilities and assess the organization's response to attacks.
Focus	Broader in scope, includes both manual and automated testing techniques.	Specific to identifying vulnerabilities and exploiting them, mainly a manual process.
Methodology	Various types of assessments, such as SAST, DAST, IAST, SCA, etc.	Manual testing using tools and techniques to simulate real-world attacks.
Exploitation	Does not involve exploitation of vulnerabilities.	Involves controlled exploitation to validate vulnerabilities.
Impact	Non-intrusive; primarily focused on identifying issues.	Can be intrusive, may cause application disruption during testing.
Reporting	Identifies vulnerabilities and provides remediation recommendations.	Documents successful exploits, identifies weaknesses, and recommends remediation measures.
Testing Approach	May include automation for vulnerability scanning.	Primarily manual, using manual testing techniques and tools.
Goal	Enhance overall security posture of the web application.	Validate the effectiveness of existing security controls and incident response capabilities.



Planning Web Application Pentests With WSTG

OWASP Web Security Testing Guide

- The OWASP Web Security Testing Guide (WSTG) is a comprehensive and community-driven resource provided by the Open Web Application Security Project (OWASP).
- The guide aims to help security professionals, developers, and organizations conduct effective web application security assessments by providing a structured and systematic approach to testing web applications for security vulnerabilities.
- It serves as a practical and hands-on reference for planning, executing, and reporting on web application security testing activities.

OWASP WSTG Checklist

- The OWASP Web Security Testing Checklist is an spreadsheet based checklist that can be used to help you track the status of completed and pending test cases.
- This checklist is based on OWASP Web Security Testing Guide and includes a comprehensive penetration testing methodology/framework that web app pentesters can implement in their pentests or security assessments.
- It also provides a set of detailed and granular web app security tests that outline the various techniques that can be used to test most common web application misconfigurations, flaws or and vulnerabilities.
- Moreover, the checklist also contains the OWASP Risk Assessment Calculator and the Summary Findings template.



Demo: OWASP Web Security Testing Guide (WSTG) Checklist



Pre-Engagement Phase

Pre-Engagement Phase

- The pre-engagement phase of web application penetration testing is a crucial step that lays the foundation for a successful and well-planned security assessment.
- It involves preliminary preparations, understanding project requirements, and obtaining necessary authorizations before initiating the actual testing.
- During the pre-engagement phase, the penetration tester and the client must discuss and agree upon a number of legal and technical details pertinent to the execution and outcomes of the security assessment.

Pre-Engagement Phase

- This can be one or more documents with the objective to define the following:

Objectives &
Scope of the
engagement

Timeline &
milestones

Liabilities &
responsibility

Authorized
actions

Expectations
and
deliverables

Statement of
work

Pre-Engagement Steps

- Understanding Project Objectives:
 - Clearly define the objectives and goals of the penetration test.
 - Understand what the stakeholders aim to achieve through the testing process.
- Scope Definition:
 - Identify the scope of the penetration test, including the specific web applications, URLs, and functionalities to be tested.
 - Define the scope boundaries and limitations, such as which systems or networks are out-of-scope for testing.
- Authorization and Legal Requirements:
 - Obtain proper authorization from the organization's management or application owners to conduct the penetration test.
 - Ensure that the testing activities comply with any legal or regulatory requirements, and that all relevant permissions are secured.

Pre-Engagement Steps

- Rules of Engagement (RoE):
 - Establish a set of Rules of Engagement that outline the specific rules, constraints, and guidelines for the testing process.
 - Include details about the testing schedule, testing hours, communication channels, and escalation procedures.
- Communication and Coordination:
 - Establish clear communication channels with key stakeholders, including IT personnel, development teams, and management.
 - Coordinate with relevant personnel to ensure minimal disruption to the production environment during testing.

Pre-Engagement Steps

- Contract and Non-Disclosure Agreements:
 - Sign necessary contracts and non-disclosure agreements (NDAs) with the organization to protect sensitive information and ensure confidentiality.
- Scoping Meeting:
 - Conduct a scoping meeting with key stakeholders to discuss the testing objectives, scope, and any specific concerns or constraints.
 - Use this meeting to clarify expectations and ensure everyone is aligned with the testing approach.
- Preparation of Tools and Resources:
 - Ensure that the testing team has all the required tools, licenses, and resources needed for the assessment.
 - Set up a secure testing environment and any necessary virtual machines for testing.

Pre-Engagement Steps

- Risk Assessment and Acceptance:
 - Perform a risk assessment to understand the potential impact of the penetration test on the web application and the organization.
 - Obtain management's acceptance of any risks associated with the testing process.
- Engagement Kick-off:
 - Officially kick-off the penetration test, confirming the start date and timeline with the organization's stakeholders.
 - Share the RoE and any other relevant details with the testing team.



Defining The Pentest Scope



Demo: OWASP ZAP Context Scope



Defining The Scope - Website Screenshots With EyeWitness



Practical Demo



Website Reconnaissance

What are we looking for

- + IP addresses
- + Directories hidden from search engines
- + Names
- + Email addresses
- + Phone Numbers
- + Physical Addresses
- + Web technologies being used



Practical Demo



Harvesting Profiles & Contact Information



Practical Demo



Finding Information in Data Breaches



Practical Demo



Web App Technology Fingerprinting



Practical Demo



Google Dorks



Practical Demo



Mapping Tools - HTTRack



Practical Demo



Passive Crawling With Burp Suite



Lab Demo: Passive Crawling With Burp Suite



Automated Crawling - Web App Scanning



Lab Demo: Web App Scanning With OWASP ZAP

Session IDs & Cookies

Session IDs & Cookies

- In the context of web application penetration testing, understanding session IDs and cookies is crucial, as these components play a significant role in user authentication and session management.

Session IDs

- Session IDs (Session Identifiers) are unique tokens or strings generated by web applications to identify and track user sessions. They are essential for maintaining stateful communication between the client (user's browser) and the server.
- Session IDs are typically used to associate requests from a user with their session data stored on the server.
- For example, suppose you're conducting a penetration test on an e-commerce website. After a user logs in, the server generates a session ID (e.g., "Session12345") and associates it with the user's session.
- This session ID is then sent to the user's browser as a cookie.

Cookies

- Cookies are small pieces of data (usually text) that a web server sends to the user's browser, which stores them locally.
- Cookies serve various purposes, such as session management, user tracking, and personalization. In the context of session management, session cookies are commonly used to store the session ID, allowing the server to recognize and maintain the user's session.
- For example, during a penetration test, you discover that the website uses cookies for session management. When a user logs in, the server sends a cookie named "sessionID" with the value "Session12345" to the user's browser. On subsequent requests, the browser includes this cookie, allowing the server to identify and associate the user's requests with their session.

Session Hijacking & Session Fixation

Session Hijacking

- Session hijacking, also known as session theft, is a security attack where an attacker illegitimately takes over a user's active session on a web application.
- In this type of attack, the attacker gains unauthorized access to the user's session token or identifier, allowing them to impersonate the victim and perform actions on their behalf.
- Session hijacking is a severe security threat because it can lead to unauthorized access to user accounts, sensitive data, and potential misuse of the hijacked session.

Session Hijacking - Token Acquisition

- Session Prediction: Predicting or guessing the session token, especially if it's predictable or lacks sufficient randomness.
- Session Sniffing: Intercepting the session token as it's transmitted over an unsecured network, such as an open Wi-Fi hotspot.
- Cross-Site Scripting (XSS): Exploiting a vulnerability in the web application to inject malicious JavaScript into a victim's browser, which can steal the session token.

Session Hijacking - Impersonation

- Once the attacker has the session token, they can impersonate the victim by presenting this token during requests to the web application.
- The application, unaware of the hijacking, treats the attacker as the authenticated user.

Session Hijacking - Impact

- Data Theft: Access and steal the victim's sensitive data, such as personal information, financial details, or confidential documents.
- Account Takeover: Change the victim's account settings, passwords, or email addresses, effectively locking the victim out of their account.
- Malicious Transactions: Conduct unauthorized transactions, make purchases, or manipulate the victim's data.
- Data Manipulation: Modify or delete the victim's data or settings.

Session Fixation

- Session fixation is a web application security attack where an attacker sets or fixes a user's session identifier (session token) to a known value of the attacker's choice.
- Subsequently, the attacker tricks the victim into using this fixed session identifier to log in, thereby granting the attacker unauthorized access to the victim's session.

Session Fixation - Token Acquisition

- The attacker obtains a session token issued by the target web application. This can be done in several ways, such as:
- Predicting or guessing the session token: Some web applications generate session tokens that are easy to predict or lack sufficient randomness.
- Intercepting the session token: If the application doesn't use secure channels (e.g., HTTPS) to transmit session tokens, an attacker may intercept them as they travel over an insecure network, such as an open Wi-Fi hotspot.

Session Fixation - Impersonation

- With a session token in hand, the attacker sets or fixes the victim's session token to a known value that the attacker controls. This value could be one generated by the attacker or an existing valid session token.
- The attacker lures the victim into using the fixed session token to log in to the web application. This can be accomplished through various means:
 - Sending the victim a link that includes the fixed session token.
 - Manipulating the victim into clicking on a specially crafted URL.
 - Social engineering tactics to convince the victim to log in under specific circumstances.

Session Fixation - Hijacking

- Once the victim logs in with the fixed session token, the attacker can now hijack the victim's session.
- The web application recognizes the attacker as the legitimate user since the session token matches what is expected.

Advanced Web Application Penetration Testing - Summary

Key Concepts - Recap

- + Web Application Pentesting Methodology
- + Planning Web App Pentests
- + Pre-Engagement Phase
- + Web App Mapping & Crawling
- + Reconnaissance
- + Session Security



Learning Outcomes Recap

- + Solid understanding of the web app pentesting methodology
- + Ability to plan and orchestrate a web app pentest
- + Ability to perform web app reconnaissance
- + Ability to map a web application through techniques like crawling
- + Solid understanding of session management & security

Next Steps

- + Additional Resources:
 - + OWASP Web Security Testing Guide
 - + OWASP Penetration Testing Checklist: https://owasp.org/www-project-web-security-testing-guide/assets/archive/OWASP_Web_Application_Penetration_Checklist_v1_1.pdf
- + Additional Tools: Caido - Web Security Auditing Toolkit: <https://caido.io/>



**Thank
You!**

EXPERTS AT MAKING YOU AN EXPERT

