CHAPTER

1

Introduction to Web Application Penetration Testing

In today's increasingly complex online landscape, it's essential to prioritize website security to safeguard personal information. With advancing technology, hackers are becoming more sophisticated in their endeavors to compromise security measures and access private data; for example, just take a look at the report "Top data breaches and cyber attacks in 2024" (`https://www.techradar.com/pro/top-data-breaches-and-cyber-attacks-in-2024`). One effective method of defense is ethical hacking, which involves testing website security by attempting to uncover vulnerabilities constructively. This proactive approach, including conducting red team exercises and continuous integration/continuous deployment (CI/CD) pipeline security assessments, enables companies and organizations to identify and address cybersecurity weaknesses before malicious actors exploit them.

Hacking web applications from an attacker's perspective allows for a more thorough and accurate evaluation of the application's real-world security as it uncovers vulnerabilities that are often missed by automated tools and standard security audits. By exploiting vulnerabilities as malicious hackers would, penetration testers gain a deeper understanding of an application's actual weaknesses and uncover issues that traditional methods often overlook. For example, automated vulnerability scanning can identify surface-level security flaws but may not reveal the complex exploit sequences that a skilled attacker

could utilize. This human-led, outside-in approach discovers more vulnerabilities and offers valuable insight into enhancing an application's defense against sophisticated cyberattacks. On the other hand, approaches focused solely on technical weaknesses or following best-practice guidelines often fail to replicate the tactics, techniques, and procedures of actual cybercriminal operations. For these reasons, web application penetration testing has become essential to robust security applications for every business.

Web application security contains a broad range of practices, such as static and dynamic application security testing (SAST/DAST) and software composition analysis (SCA), aimed at protecting web-based assets, including websites and their data, from threats such as hackers, malware, and misconfigured applications. Since web applications interact with users over the public Internet, they are vulnerable to security risks from threats such as hackers, malware, and misconfigured applications. Web application security is designed to protect the confidentiality, integrity, and availability of web-based assets like websites and their data.

To secure web applications, common measurements include the following:

- **Authentication and authorization:** Implement multifactor authentication (MFA) and role-based access control (RBAC).
- **Session management:** Use a strong session ID and securely manage it. Apply secure cookies with the HttpOnly and Secure flags.
- **Input validation:** Use whitelisting methods and regular expressions to clean and validate user inputs.
- **Output encoding:** Use encoding libraries such as OWASP Java Encoder.
- **Secure configuration:** Follow security hardening guidelines like CIS benchmarks.
- **Encryption:** Use Transport Layer Security (TLS) for data in transit and Advanced Encryption Standard (AES) for data at rest.

Web applications face various security threats, such as the following:

- **Injection vulnerabilities:** SQL injection (SQLi) and command injection
- **Authentication issues:** Brute-force attacks and credential stuffing
- **Session management:** Session hijacking and session fixation
- **Cross-site scripting:** Reflected, stored, and DOM-based XSS
- **Insecure direct object references (IDOR):** Unauthorized access to protected data
- **Security misconfiguration:** Unpatched software and exposed configuration files
- **Lack of transport layer protection:** Man-in-the-middle (MitM) attacks

To counter these threats, web application security solutions use strategies such as securing the development process, deploying web application firewalls, and performing regular security patching and audits. Penetration tests for web applications are important for staying ahead of evolving threats. They find problems before they can be misused, which helps lower the chances of security breaches, loss, and damage.

In this chapter, I'll discuss why web application security and penetration testing are important for all businesses. I'll start with an overview of the web penetration testing process and the techniques to use. Then, I'll discuss common web-based vulnerabilities and attacks that every penetration tester should know about.

The Importance of Web Application Security

The need to keep our online spaces safe affects every part of the Internet, not just websites. Protecting all online information equally is important, whether for websites or anything else online. Unfortunately, when hackers find a weak spot, it can cost companies a lot of money. This includes the money they have to spend to fix the problem, the money they lose because their services are down, and the trust they lose from their customers. For instance, the direct costs of remediation include repairing systems, hiring cybersecurity experts, and conducting thorough investigations. Additionally, companies face significant revenue losses during service downtimes as customers cannot access services. Moreover, the long-term impact on customer trust and brand reputation can be devastating. For example, the 2017 Equifax breach resulted in millions in fines, steep stock price drops, and irreparable damage to consumer confidence. Think about how bad it would be if the stock market went down for just an hour or someone got into a lot of customer credit card info. This shows why it's so important to keep online spaces secure. IBM's "Cost of a Data Breach Report 2023" discusses how expensive cyberattacks can be. You can access this report for free at `https://www.ibm.com/reports/data-breach` to see how much money these attacks can cost.

Businesses of all sizes now prioritize application security for several reasons. They employ security consultants, establish in-house security teams, and collaborate with third parties to assess and enhance their web application security. What was considered a luxury or limited to critical infrastructure is now standard practice for most organizations that depend on web applications.

The CIA Triad

As a web application security professional or penetration tester, it's crucial to understand how to measure the risk and impact of vulnerabilities and attacks. This

understanding helps assess the potential harm these security issues may cause a web application. It's important to know about the CIA triad, a fundamental information security principle.

The CIA triad is a necessary concept in information security, covering three essential principles as illustrated in Figure 1.1.

Keeping information confidential means making sure only authorized people can access it. This stops unauthorized access, sharing, or theft.

Integrity means keeping data accurate and consistent. It acts as a protection against any unauthorized changes, tampering, or corruption.

Availability confirms that authorized entities like users can access data and resources consistently without disruptions or service denials. These three pillars are crucial for securing information systems, emphasizing the importance of protecting sensitive data, maintaining its accuracy, and ensuring access for authorized individuals.

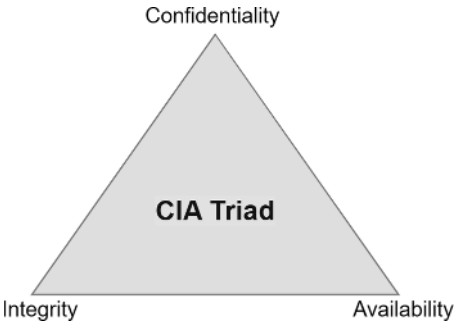

Figure 1.1: The CIA triad

Web apps use input validation, output encoding, and transaction security techniques to ensure data accuracy and prevent unauthorized modifications. Input validation filters out malicious data before processing, output encoding ensures safe data rendering, and transaction security maintains the integrity of sensitive transactions. If data is changed without authorization, it could lose its reliability and value.

Implementing authentication, authorization, and encryption in web apps assures that only users with proper authorization can access restricted data, keeping it confidential. Without adequate access controls, sensitive data in web apps are at risk of exposure.

Using secure configuration, patch management, and denial-of-service prevention, web apps can stay up and running and available for legitimate users. This is important because if web applications go offline, it can cost businesses millions of dollars per hour and harm their reputation.

When these CIA objectives are achieved, web applications can work safely and dependably, safeguarding the interests of businesses, customers, and users. The CIA triad offers a high-level structure for companies to assess the effectiveness of their web app security measures.

Proper input validation and output encoding are important for maintaining data integrity by filtering out malicious content that could alter data. However, it can be challenging to balance the CIA triad. For instance, increasing authentication for more robust confidentiality can affect availability, while implementing encryption for better integrity could create more user friction. Web application security needs to find the right balance.

Industry Needs

Web applications are complicated and involve many different technologies, platforms, and components. For instance, a modern web application may integrate with multiple application programming interfaces (APIs), utilize microservices architecture, and depend on third-party libraries, each raising unique security challenges. To perform security testing effectively, a deep understanding of these elements must pinpoint vulnerabilities across the entire system, from the client side to the server side and backend databases.

As web apps evolve, new vulnerabilities are frequently discovered. This requires security teams and specialists to continuously research, learn, and update their testing processes with the latest methods.

Detecting potential vulnerabilities in modern web applications requires automated testing tools and specialized knowledge. Specialists who understand how specific technologies or architectures operate are essential because many threats are associated with those tools and code.

The demand for web application security specialists is expected to increase due to constant attacks and emerging risks. As web apps remain complex and vulnerable, securing them will continue to require ongoing learning and adaptation.

Meeting regulations like General Data Protection Regulation (GDPR), Payment Card Industry Data Security Standard (PCI DSS), Health Insurance Portability and Accountability Act (HIPAA), and Network and Information Security Directive (NIS2) requires expertise in identifying sensitive data, assessing risks, and implementing necessary application controls. This demand is increasing the need for web app security professionals.

TIP **You can find more information about these standards here:**

```
https://gdpr.eu

https://listings.pcisecuritystandards.org/documents/PCI_DSS-
QRG-v3_2_1.pdf

https://www.hhs.gov/hipaa/index.html

https://digital-strategy.ec.europa.eu/en/policies/nis2-
directive
```

The field of web app security attracts experts from various backgrounds in development, quality assurance (QA) testing, IT security, and compliance.

Each contributes different skill sets required to secure today's complex web application ecosystems comprehensively.

There is a significant shortage of people skilled in cybersecurity and web application security. This high demand means that experts in this area can earn high salaries. The need for specialists in web application security is increasing due to several reasons: more aspects of our lives are moving online, providing more targets for hackers; cybercriminals are becoming more sophisticated in their methods; new rules and regulations are being introduced to protect data; and businesses are rapidly transitioning to digital platforms. This situation is clearly shown in the NIST infographic at `https://www.nist.gov/system/files/documents/2023/06/05/NICE%20FactSheet_Workforce%20Demand_Final_20211202.pdf`. The infographic shows why a career in cybersecurity, especially in web application security, is in high demand and fulfilling.

Overview of Web Application Penetration Testing

Penetration tests for web applications extend beyond automated tools. While these tools can identify common issues, they may overlook more intricate ones. Manual tests conducted by experienced professionals provide a more thorough analysis and reveal complex vulnerabilities that automated tools might miss, allowing for a comprehensive evaluation of the application's security.

In addition to identifying vulnerabilities, penetration tests are a proactive risk management measure. They facilitate the effective prioritization and allocation of resources to address issues. Pentest reports offer valuable insights into the potential impacts and likelihood of exploitation, which empower informed decisions regarding security spending and mitigation efforts. This approach focuses resources on areas with the highest potential for harm, making security more effective overall.

Based on Figure 1.2, the general architecture of a web application consists of a front end that users interact with, such as menus, and a backend that includes servers for handling requests and responses connected to a database. It also includes APIs for linking to third parties and other components of web applications. Each section has its vulnerabilities to specific types of attacks, which we will cover in this book.

Pentests also help create a security culture by increasing teams' awareness of secure coding, configuration, and practices. Integrating security into development proactively addresses security, identifies recurring issues, and fosters a security mindset among teams, as shown in Figure 1.3.

TIP **In this book, we will learn and practice web-based penetration testing, focusing on the security of live web applications deployed in production. It's important to note that this environment may sometimes be replicated in a controlled or developed environment. Our approach is to engage with live web applications, not the code!**

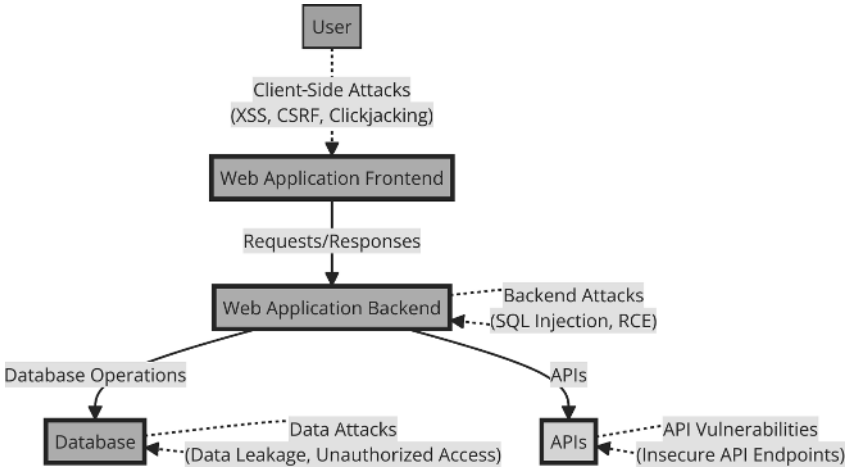

Figure 1.2: A web application architecture and related attack surfaces

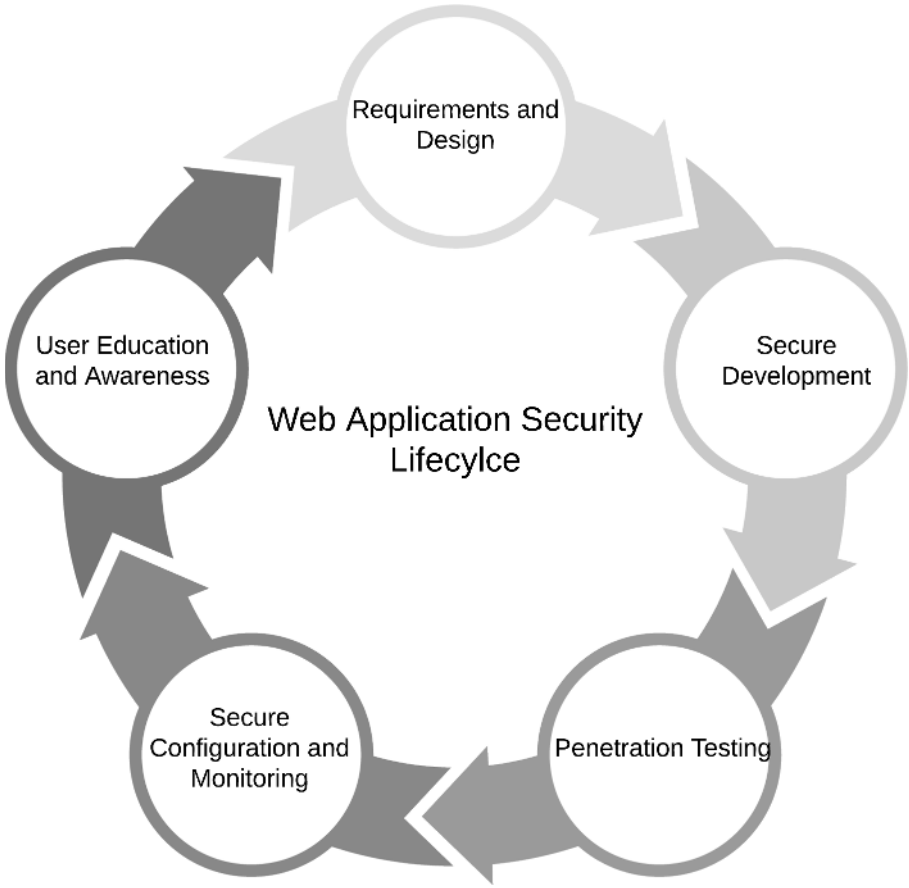

Figure 1.3: The lifecycle highlights that penetration testing is just one component of web app security

The Penetration Testing Process

A successful web application penetration test involves several stages, as shown in Figure 1.4. Some of these stages are not purely technical. The testing process begins with an important phase called scoping and reconnaissance. Though not technical, this phase is crucial for the entire test's success. It's about preparing and setting everything up for what's to come. During this stage, the tester takes time to understand the client's needs, identifies the main areas to focus on, and determines their goals for the test. It's not just about knowing how to break into systems or find vulnerabilities; it also involves planning ahead, organizing the work, and ensuring that the most critical tasks receive the highest priority.

Figure 1.4: The penetration testing process

During this initial phase, it's important to define the objectives and scope of the test clearly. This involves understanding the client's needs, identifying potential risks, and deciding on specific goals. A clear plan at this stage helps the penetration tester focus their efforts and resources efficiently. Another vital aspect is organizing the test. This means coordinating with everyone involved, scheduling activities, and allocating resources effectively. This organization ensures that the test runs smoothly and everyone involved understands what's happening.

In this phase, it's important to prioritize goals. The tester evaluates which vulnerabilities or security weaknesses could have the most significant impact. They then focus their attention and resources on these areas. This approach ensures efforts are concentrated where they can make the most difference by first addressing the most critical security threats.

After completing these basic steps, the tester is ready to advance to the more technical phases of the process. They should follow the structure outlined in the figure, moving from scanning to gaining access, then to maintaining access, and finally to analysis and report. Every step in the process builds on the work done during the scoping and reconnaissance phase. It highlights the importance of initial planning and organization in conducting a thorough and effective penetration test.

Scoping and Reconnaissance

The initial stages of any penetration test are scoping and reconnaissance. Proper scoping is important as it determines the test's boundaries, limitations, and

objectives, ensuring an effective and targeted analysis. During scoping, discussions with the client help to understand their key requirements, priorities, and risk tolerance. Based on these inputs, the penetration tester will define what systems, applications, and data will be included or excluded from the test. Clear scoping also establishes expectations and lays the foundation for a thorough and actionable test report.

Gathering intelligence about the target systems and networks, known as *reconnaissance*, comes after scoping. This involves collecting information passively and actively from sources like company websites, social media, search engines, vulnerability databases, and subpoenaed documents. The penetration tester will map the network architecture, identify critical systems and applications, and determine employees and contractors. This reconnaissance provides valuable insights to assess exploitation techniques and prioritize test efforts within legal and ethical boundaries.

After completing basic scoping and reconnaissance, the penetration tester can create a test plan that details specific weaknesses and vulnerabilities to target. Based on the intelligence gathered, test cases are designed to systematically evaluate security controls and identify exploitable vulnerabilities. Technical tools are used to perform targeted scans, intercept web traffic, and attempt privilege escalation. This will be covered separately later in this book.

The outcomes of scoping, reconnaissance, test planning, and initial testing form the basis for the rest of the penetration test. Any extra systems found or data exposed during active testing can be added to the original scope. Well-planned scoping and reconnaissance establish the foundation for a thorough and insightful penetration test that identifies actual security weaknesses and provides practical remediation suggestions.

HINT **The first and most important stage of any successful penetration test is reconnaissance, which provides the depth of information needed to identify genuine vulnerabilities. Proper scoping determines the boundaries and objectives of the test, but without accurate and thorough reconnaissance, the rest of the test will be limited and less insightful.**

Scanning

Penetration testing is a process of uncovering hidden information on a target system. This involves using various automated tools and techniques, such as port scanning to detect open ports, service scanning to identify running services, and network scanning to map the network layout. Additionally, vulnerability scanning is used to pinpoint security flaws, while host scanning is used to inventory network-connected devices. SSL/TLS scanning is used to identify encryption issues.

Port scanning checks for open ports that allow external connections, which helps penetration testers find vulnerabilities. Service scanning determines active services, revealing potential attack points. Network scanning creates a map of the network structure, providing a layout of the target environment. Vulnerability scanning looks explicitly for security holes that could be exploited. Network-connected devices are logged when scanning hosts, giving penetration testers a complete view of the target. SSL/TLS scanning looks for incorrect encryption setups and identifies issues with security protocols designed to protect data.

By using these techniques, penetration testers are highly likely to discover vulnerabilities that may jeopardize the security and integrity of the target system. The scanning phase establishes the foundation for subsequent exploitation by thoroughly understanding the target and its potential attack points. In general, scanning plays a crucial role in the penetration testing process.

Gaining Access

After the initial scanning, the next phase is gaining access, which is an exciting part of a penetration test. During the scanning phase, penetration testers try to exploit those weaknesses and gain access to the target system when vulnerabilities are found. This may involve accessing an open port, cracking weak login credentials, hijacking a session, or exploiting a software flaw. Gaining initial access can be pretty challenging as it requires navigating through layers of security controls.

An open or vulnerable port frequently serves as the entry point for unauthorized access. Ports that are open to the Internet enable incoming connections, and if not adequately secured, they can provide penetration testers with an initial point of access. Subsequently, testers will attempt to gain entry into login portals and try to guess or crack account passwords through brute-force methods. If they are successful, they can access user accounts and potentially the entire system with escalated privileges.

DEFINITION **Gaining initial access or a foothold into a target system is an important first step in penetration testing. Establishing a foothold provides a starting point for penetration testers to investigate the network further and enhance access.**

At times, more technical approaches are necessary to gain access. For instance, exploiting weaknesses in web software or a server to run harmful code, acquire higher privileges, and gain control. Other advanced tactics include intercepting and taking over user sessions and exploiting vulnerabilities in enterprise services like remote access or administration protocols.

Once inside, penetration testers typically have broad access to resources!

Maintaining Access

After initially gaining access to a target system, penetration testers often aim to maintain that access and establish a foothold to conduct further testing. Like real attackers, testers work to cover their tracks and ensure they can regain access even if their initial method is discovered or blocked.

Security testers may deploy backdoors, rootkits, or other forms of malicious software to establish covert access. Moreover, they may exploit authorized tools and applications that blend in with regular system operations. This could require creating accounts with elevated permissions, extracting and decoding password hashes, or misusing remote administration tools. Additionally, testers may infiltrate less critical systems within the network that are not precisely monitored, utilizing them as initial access points to progress further within the environment.

DEFINITION **The concept of pivoting needs to leverage a previously compromised system or network to access other systems or networks within the targeted environment. This approach enables the tester to broaden the scope of their assessment by traversing interconnected systems, thereby facilitating a more comprehensive penetration test.**

Maintaining access over time enables penetration testers to replicate a committed attacker's actions. It allows them to thoroughly examine the target's network, identify additional vulnerabilities, and gain higher privileges when necessary to uncover weaknesses across multiple layers. This stage can expose flaws in an organization's incident response and threat-hunting capabilities, highlighting areas that real attackers could exploit to operate without detection for extended periods. By exercising persistence within controls and limitations, penetration testers offer valuable insights to strengthen defenses and minimize the risks of covert, long-term security breaches.

DEFINITION **Threat hunting represents a proactive cybersecurity approach that actively explores potential threats within an organization's network. This process involves using advanced techniques to detect and mitigate these threats before they inflict harm.**

Analysis and Report

After penetration testers have gained and maintained access to a target system, the next step is to analyze their findings and compile a comprehensive report thoroughly. This requires documenting all identified vulnerabilities, tested attack vectors, exploited weaknesses, and any access or privileges obtained.

It also involves mapping the scope of impact, such as compromised systems, exposed data, and potential business risks.

The analysis process demands detailed consideration of the importance of comprehensive security arising from the test results. Pentesters must adopt the perspective of potential attackers to evaluate the realistic extent of damage that could be inflicted through the attained access level. Also, they need to assess how easy it is for attackers to find and use weaknesses and look for ways attackers could gain more access or move through the network.

The final report presents the testers' findings and recommendations clearly and actionably. It outlines the discovered vulnerabilities, the ones that were attempted but not successfully exploited, and other significant findings. The report also includes and assigns risk ratings using the Common Vulnerability Scoring System (CVSS) and suggests solutions for remediation. The report may model potential attack scenarios for high-risk vulnerabilities, showing how an attacker could inflict severe damage if the problems are not addressed.

The analysis and reporting phase includes converting raw test data into actionable intelligence that organizations can leverage to fortify their defensive measures. An extensive, well-communicated report is pivotal in enabling stakeholders to comprehend the actual risks they confront and confirm the resources required for efficacious remediation. The main aim is to give context and urgency to the findings, which will help companies prioritize the most effective security improvements.

Detailed information and examples are provided in Appendices B and C.

Methodologies

Web application penetration testing methodologies are structured frameworks that outline the steps and procedures involved in executing thorough and effective penetration tests on web applications. These methodologies have a systematic approach, comprehensive reconnaissance, vulnerability scanning, exploitation, and complete reporting, ensuring a detailed assessment of the application's security posture. This book will align with the OWASP Top 10 as a foundational framework for further discussion and analysis.

OWASP Top 10

OWASP provides two main web application penetration testing approaches: the Top 10 and the Testing Guide. The OWASP Top 10 (`https://owasp.org/Top10`) has a prioritized list of the most critical web application security risks, including injection, broken authentication, and sensitive data exposure, while the OWASP Testing Guide is a detailed methodology for assessing each vulnerability category, including information gathering, configuration management

testing, and business logic testing. Together, they form a comprehensive framework for penetration testers.

The OWASP Top 10 highlights the 10 most important web application vulnerabilities. It is a guideline for organizations to identify and fix issues like injection flaws, cross-site scripting, and broken authentication. Addressing the risks outlined in the OWASP Top 10 can significantly strengthen an application's overall security, providing a robust defense against the most prevalent and dangerous threats.

DEFINITION **The Open Web Application Security Project (OWASP) is an open-source community of professionals collaborating to create standards, tools, and projects that help experts build secure applications.**

OWASP Web Security Testing Guide

The OWASP Web Security Testing Guide (`https://owasp.org/www-project-web-security-testing-guide`) presents comprehensive methodologies for assessing each vulnerability category within the Top 10. It contains techniques for gathering information, conducting configuration management testing, assessing authentication systems for bypass potential, and testing business logic. Following the Testing Guide protocols guarantees a careful evaluation of each area posing a Top 10 risk.

The OWASP Top 10 and Testing Guide work together to give structure and flexibility to web application assessments. The Top 10 lists the most critical vulnerabilities to focus on first, while the Testing Guide outlines the methods needed to identify those issues. This helps penetration testers optimize their efforts and measure how well an application defends against the most common risks that attackers exploit.

NOTE **This book will use OWASP as our main framework. OWASP is designed for web application penetration testing and is considered the standard for web application security best practices.**

Open-Source Security Testing Methodology Manual (OSSTMM)

The OSSTMM (`https://www.isecom.org/OSSTMM.3.pdf`) is a framework for penetration testing created by ISECOM. Its purpose is to thoroughly evaluate the security of a network's broadcast domain, which includes all devices that can communicate with each other in a network.

What makes the OSSTMM different is its broad focus. It doesn't just search for weaknesses in software or networks. It also looks at how practical staff training is, considers the impact of human behavior on security, and examines physical security measures. This means the OSSTMM covers everything from how well employees can defend against attacks to potential security risks posed by people to the security of the physical premises.

By addressing these areas, the OSSTMM offers a complete view of an organization's security situation, ensuring that all potential threats, whether digital, human, or physical, are considered.

The Penetration Testing Execution Standard (PTES)

PTES (`http://www.pentest-standard.org/index.php/Main_Page`) is another widely recognized web application penetration testing methodology. PTES follows a comprehensive approach that includes information gathering, vulnerability scanning, exploitation, and reporting. It highlights the importance of planning, scoping, and documenting the testing process. By following PTES, penetration testers can ensure that they cover all the necessary steps and provide a thorough assessment of the web application's security. PTES offers a well-defined framework that helps maintain consistency and ensures that no critical areas are overlooked during testing.

TIP **PTES is a highly adaptable framework used in various domains, such as network, system, wireless, and others. It is not specifically designed only for web application penetration testing.**

Tools and Techniques

The web application penetration testing process requires security analysts to use different tools and methodologies to identify vulnerabilities. These tools contain both free and open-source options as well as commercial products. Additionally, manual methods such as code reviews and inspections hold significance. It is imperative to recognize the perpetual relevance of the human element. It's important to carefully examine all results and outputs, regardless of the tools used. There is a chance of experiencing false positives, where the tool must provide accurate information. Human discernment and astute analysis assume pivotal roles in the comprehensive evaluation and interpretation of results, ensuring precise decision-making.

Many free and open-source tools are available to help with web application penetration testing. Both command-line and graphical user interface (GUI)–based

tools can handle tasks such as intercepting web traffic, conducting fuzz testing, and automating SQL injection attacks. These tools can quickly and efficiently identify issues on a large scale, making it difficult to find them manually. Open-source tools offer a variety of capabilities at no cost. However, they may lack the advanced functionality of commercial products.

TIP **You can find more information and a list of free and open-source testing tools here:** `https://owasp.org/www-project-web-security-testing-guide/v41/6-Appendix/A-Testing_Tools_Resource`.

Commercial web application penetration testing tools offer enhanced features for a fee. These tools typically have advanced capabilities such as tailored vulnerability modeling, asset identification and mapping, automated report generation, and exploit development. Important considerations should be reflected when selecting a tool. Manual methods will continue to be essential for comprehensive testing. In the interim, a blend of tools, methodologies, and expertise is imperative to guarantee comprehensive penetration testing of stylish, detailed web applications.

A web proxy is a commonly used toolset in web application penetration testing. It is an intermediary between the tester and the web application, allowing them to intercept and modify requests and responses. This enables testers to analyze the traffic, manipulate inputs, and identify security vulnerabilities. Web proxies are crucial in identifying issues such as insecure transmission of sensitive data, insufficient input validation, and weak authentication mechanisms.

NOTE **In the upcoming chapters and scenarios, we will use proxy tools extensively. In the following chapter, I will demonstrate how to configure web proxies.**

Fuzzing tools are required for web application penetration testing. They generate a substantial volume of random or malformed inputs to produce unexpected behavior and expose vulnerabilities within the application. The practice of fuzzing is instrumental in identifying buffer overflows, input validation flaws, and other security weaknesses that may not be easily noticeable through traditional testing methodologies. Fuzzing tools are great at finding complex problems and providing helpful insights for further analysis and solutions.

DEFINITION **Fuzzing is like testing software by putting in weird or random data to see if it has any problems. The goal is to find security issues and weaknesses by giving the system different inputs to see what happens.**

Web application penetration testing uses different methodologies in addition to tools and techniques. These methodologies are white-box, gray-box, and

black-box testing (see Figure 1.5). White-box testing grants you complete access to the application's internal architecture, source code, and structural intricacies. This methodology helps comprehensive testing and in-depth analysis, thus enabling the thorough identification of vulnerabilities. Gray-box testing provides partial knowledge of the application, such as restricted access to the website or specific system details. This methodology balances white-box and black-box testing, affording a realistic assessment of the application's security posture. On the other hand, black-box testing emulates an external hacker without knowledge of the application. You solely rely on publicly available information to disclose potential vulnerabilities exploitable by real attackers.

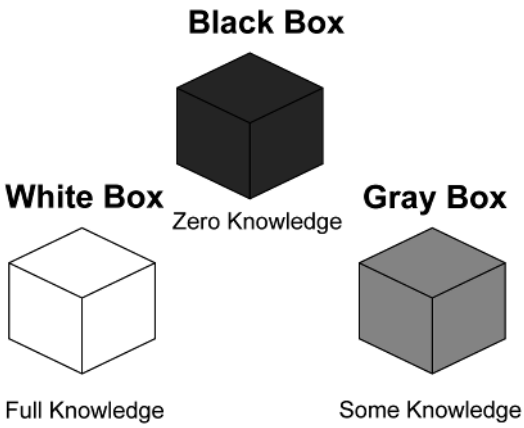

Figure 1.5: Different pentest approaches divided by the pentester's knowledge about the target

Different testing approaches use various tools and techniques. In white-box testing, manual code review, thorough scanning with vulnerability scanners, and extensive manual testing are expected. Gray-box testing may involve a combination of automated scanning tools, manual testing, and limited access to the application's internals. Black-box testing relies on automated scanners, web proxies for traffic analysis, and manual testing techniques that simulate an attacker's perspective. (It can also be manual testing only. So, automated testing is not mandatory.)

Reporting

I'd like to emphasize the significance of reporting. Reporting shouldn't just be about finding vulnerabilities but should also include actionable recommendations for fixing them. These recommendations should be practical and customized to the web application's specific vulnerabilities and context. They should

offer clear guidance on how to address the identified security weaknesses and enhance the application's overall security.

NOTE **You will learn how to write an impactful web application penetration testing report in Appendix C.**

Effective stakeholder communication is indispensable throughout the reporting process. Penetration testers should be able to articulate the findings clearly and comprehensively, eschewing technical terminology where possible. Furthermore, it is important to actively engage with the management level and address any questions or concerns regarding the findings or recommendations.

Providing practical, achievable, and industry-standard actionable recommendations to enhance application security is very important. These recommendations should encompass specific steps or measures aimed at mitigating identified vulnerabilities, such as patch applications, adoption of secure coding practices, or enhancement of access controls.

Moreover, reporting should be directed only toward technical stakeholders. It should also be understandable for nontechnical stakeholders, like senior management or business owners. This will help them grasp the risks and make well-informed decisions regarding the application's security and any necessary remediation efforts.

Reporting should be timely. The report should be delivered promptly after the web application penetration testing is completed. This ensures that stakeholders receive the findings and recommendations promptly, allowing them to take immediate action to address the identified vulnerabilities.

Regular checking and retesting can confirm that the recommended fixes have been implemented properly. This shows that the fixes are working and ensures that the web application's security is improving.

Clear and brief reporting is also important in web application penetration testing. This is vital for effectively communicating findings, documenting vulnerabilities, assigning risk ratings, and providing actionable recommendations. It helps effective communication with stakeholders and managers, enhances the understanding of the security posture, and supports decision-making processes to improve application security. In this book, I will demonstrate in detail and provide practical examples to help you master the art of clear and straightforward reporting in web application penetration testing.

Types of Web Application Vulnerabilities

This section will reference the OWASP Top 10 to explore various web application vulnerabilities. The goal is to analyze the standard security flaws found in web applications as outlined by OWASP. By using the OWASP Top 10, we can

comprehensively understand these vulnerabilities and their potential impact on web application security.

The following topics are related to OWASP's Top 10 categories, each representing a high-risk issue that threatens web applications. We will delve into these extensively to understand their nature and impact.

The OWASP Top 10 changes regularly to cover new threats. Focusing on its flaws helps teams prioritize fixing vulnerabilities and allocating resources for better security. The list acts as a security benchmark, letting organizations align their strategies with industry standards.

Figure 1.6 shows the OWASP's Top 10 most exploited weaknesses. Understanding and addressing these vulnerabilities can significantly reduce risk and protect applications and users from attacks. Developers and security teams must stay current with the OWASP Top 10 and enforce effective measures to mitigate these issues.

OWASP Top 10:2021	
No.	**Vulnerability**
A1	Broken Access Control
A2	Cryptographic Failures
A3	Injection
A4	Insecure Design
A5	Security Misconfiguration
A6	Vulnerable and Outdated Components
A7	Identification and Authentication Failures
A8	Software and Data Integrity Failures
A9	Security Logging and Monitoring Failures
A10	Server-Side Request Forgery

Figure 1.6: The OWASP Top 10 vulnerabilities

Broken Access Control

According to OWASP, broken access control occurs when access control and authentication functions in an application are not implemented correctly. This includes issues such as missing or improperly implemented access control checks, weak session management, and insufficient permission enforcement. This allows attackers to bypass intended authorization, leading to unauthorized access, a significant factor in many attacks. Attackers exploit weak spots or misconfigured access control methods, potentially leading to severe consequences. To prevent this, applications must enforce robust authorization and validation at all access points. The most common issues involve missing access control

checks for specific functions, concealing unauthorized actions within authorized actions, and neglecting authentication weaknesses. Default passwords, backdoors, and easily guessable usernames can also lead to broken access control. To defend against this, all authorization procedures, default settings, forgotten access points, and authentication methods should be carefully reviewed for security. Using role-based access control, privilege separation, and the principle of least privilege can be helpful in reducing the impact of broken access control vulnerabilities.

Remember to ensure proper access control security by validating all input, authorizing all functions, encrypting credentials, limiting access attempts, and logging access to sensitive functions. Developers should pay attention to tasks like data sanitization and input validation, as attackers could exploit vulnerabilities to gain unauthorized access. Before deployment, it's important to carefully review the design and test the security of access control logic. This helps identify and address potential issues.

Cryptographic Failures

OWASP states cryptographic failures occur when applications and APIs do not effectively safeguard sensitive data through encryption, hashing, and key management. Inadequate cryptography can allow attackers to intercept, alter, or exploit confidential information, such as passwords or e-payment card details. Examples of cryptographic failures include using deprecated algorithms (e.g., MD5, SHA-1), incorrect encryption modes (e.g., ECB instead of CBC), improper initialization vectors (IVs), insufficient key management (e.g., hard-coded keys, weak key generation), and prioritizing performance over security by not using sufficient key lengths. To solve these problems, developers should use up-to-date algorithms, pick the proper modes and key lengths, set up vectors correctly, create and store keys securely, and have cryptography experts conduct thorough security assessments.

ONE OF THE FAMOUS CRYPTOGRAPHIC FAILURE VULNERABILITIES

Heartbleed (CVE-2014-0160): This bug in OpenSSL exposed sensitive information such as usernames, passwords, and private keys from the server's memory. It impacted hundreds of thousands of sites. See `https://heartbleed.com`.

DEFINITION **Common Vulnerabilities and Exposure (CVE) is a global database of publicly disclosed cybersecurity vulnerabilities. This list is developed and maintained by the MITRE Corporation (`mitre.org`). Top IT vendors like Microsoft, Oracle, Cisco, and IBM act as CVE Numbering Authorities (CNAs). A CVE ID is unique and**

assigned to a specific vulnerability and security advisory. A CVE ID consists of a year and a unique number, e.g., "CVE-Year-Number." Remember that software or products can have multiple CVEs, each pointing to a particular cybersecurity vulnerability. Since you will encounter CVEs in different chapters of this book, I highly recommend learning more about them by exploring the following sites:

```
https://cve.mitre.org
https://www.cve.org
https://csrc.nist.gov/glossary/term/common_vulnerabilities_
and_exposures
```

Injection

When user-supplied data is sent to an interpreter as part of a command or query, it is considered an injection. Attackers exploit poor input validation to inject malicious code or commands into the input data, which the interpreter then executes. Several common types of injection vulnerabilities include SQL injection, OS command injection, LDAP injection, XML injection, and format string injection. These vulnerabilities can be exploited to execute arbitrary commands, retrieve sensitive data, and manipulate application behavior.

To prevent injection, developers should validate and sanitize all input, use parameterized queries, avoid interpreters, and conduct proper security checks. User input should be treated as untrusted by default, and APIs should be designed only to accept specified data types. Data validation should occur at every point that accepts inputs and whenever those inputs are passed along an execution flow.

In the following few chapters, you'll learn about different types of injections, focusing on the important concept of injecting into databases, mainly SQL injection.

HINT **Injection attacks can occur wherever user input is accepted, regardless of the technology or environment. This book will focus on important injections in web applications, including SQL, SAML messages, and XML injections.**

Insecure Design

Insecure design refers to flaws in application design that make the app vulnerable to attacks. Typical insecure designs lack security requirements during the planning phase, nonadherence to a threat modeling approach as STRIDE or DREAD, and secure coding best practices like OWASP ASVS. Failing to include these steps can cause problems like not checking if functions work, giving too many execution privileges, not setting up trusted boundaries, and exposing sensitive data. Mitigating insecure design commences with the early

establishment of security requirements during the design phase. Developers must understand the acceptable use cases, potential threats, assets, and risks associated with the application. A comprehensive threat modeling exercise can disclose security vulnerabilities inherent in the design.

DEFINITION **Threat modeling helps you identify risks, potential vulnerabilities, and attack vectors to implement security measures throughout an application's life-cycle. Learn more about the threat modeling process at `https://owasp.org/www-community/Threat_Modeling_Process`.**

When implementing a program, developers need to follow secure coding guidelines. They use code reviews and security tests to find design flaws developers might have missed. Using secure frameworks and libraries that enforce security by design can reduce the chances of insecure designs. Security standards such as OWASP ASVS and the prohibition of risky functions provide checklists to see if the design has included security. To create a secure application that can fight against attacks, it is crucial to properly plan, model, implement, and verify the design.

DEFINITION **The OWASP Application Security Verification Standard (ASVS) is a set of controls for creating and developing secure applications. It includes a list of security requirements that applications must meet to be considered secure. Following ASVS helps you create apps that follow the best security practices. See `https://owasp.org/www-project-application-security-verification-standard`.**

DEFINITION **Static Application Security Testing (SAST) analyzes an application's source code or compiled versions to identify security vulnerabilities. This process helps prevent and resolve coding errors and insecure programming early.**

Security Misconfiguration

One common security issue is misconfiguration in an application's stack or server. The OWASP Top 10 suggests several areas of misconfiguration: missing proper security measures at any part of the application stack or incorrectly set permissions on cloud services, default settings left unchanged, unpatched software, and verbose error messages revealing too much information. This shows that security settings are not locked down and configured in software components like web servers, application servers, frameworks, libraries, and databases. This could expose sensitive data, allowing attackers to exploit insecure configurations.

Disabling unnecessary features can reduce an application's attack surface. Many applications come with default features that are enabled but never used. These may include unnecessary ports, services, pages, accounts, and privileges. Keeping these default features enabled can expose vulnerabilities that attackers may exploit.

Applications and services often come with default accounts and passwords that are not changed during setup. This can put them at risk of unauthorized access. To prevent this, changing all default accounts and passwords during setup is important. Error handling should provide necessary information to users without revealing too much detail that attackers could exploit using generic error messages and logging detailed errors on the server side. Some applications display detailed error messages that can give attackers insight into the application's internal workings, which they can then use to create targeted attacks.

HINT **Periodic audits and system hardening based on security standards and best practices mitigate the risk of security misconfiguration vulnerabilities.**

Vulnerable and Outdated Components

Using outdated or vulnerable software components poses significant risks to application security, as these components may contain publicly known exploits that attackers can easily leverage. Organizations often don't know all the versions of the components they use, both on the user's and server's sides, including the parts those components need to work. This makes it hard to check for problems and security updates.

If the software is not up-to-date or has security weaknesses, it puts the entire application at risk. This includes the operating system, web servers, application servers, databases, applications, APIs, libraries, and runtime environments. Regularly checking for weaknesses and signing up for security updates for all the used components is important but often overlooked.

It's important to promptly fix and upgrade the platform, frameworks, and dependencies. However, these tasks are often delayed and may occur only monthly or quarterly, leaving organizations vulnerable to security risks for extended periods. Additionally, software developers must test updated libraries for compatibility to avoid breaking application functionality.

Using outdated components is a common issue that expands an application's attack surface and risks the exploitation of known vulnerabilities. Companies must have better visibility into all components in use, regularly scan for vulnerabilities, and promptly upgrade and patch all software.

HINT **Attackers often target web systems and infrastructure due to vulnerable or outdated components. Addressing these security weaknesses is really important, as they are a common entry point for many attacks.**

Identification and Authentication Failures

One common mistake with security is having weak controls for usernames and passwords. This lets automated attacks happen, like when the bad guys have a list of usernames and passwords and try them all. It also enables them to keep trying different passwords until they get in. To stop this, the controls should be more innovative, and passwords should be solid and different for everyone. They should also have ways to get your account back that no one can guess. And use password hashing algorithms like bcrypt or Argon2 to securely store passwords.

Some top identification and authentication failure vulnerabilities include the following:

- Allowing default or weak passwords makes it easy for attackers to gain unauthorized access.
- Failure to detect and block repeated login attempts enables brute-force attacks.
- Not using multifactor authentication since relying only on usernames and passwords is not secure. Use MFA whenever possible.
- Exposing session IDs, such as including them in URLs or storing them unencrypted, can lead to session hijacking attacks.
- Failing to invalidate sessions and not revoking session IDs after logout or inactivity allows unauthorized access.

To fix these issues, use strong passwords, encrypt session IDs, and require manual reviews for actions like password resets. Also, block any suspicious bot activity.

Fixing common identification and authentication weaknesses makes it harder for attackers to carry out automated attacks and strengthens access controls. The right combination of technical and procedural controls is essential.

Software and Data Integrity Failures

Failing to ensure the integrity of software code and data can lead to several vulnerabilities, such as unauthorized code execution, data manipulation, and supply chain attacks. Many applications rely on plugins, libraries, or modules from untrusted sources that lack integrity controls. They can introduce vulnerabilities if continuous integration and delivery pipelines are not correctly secured. Attackers could potentially upload malicious code updates that are distributed to all users.

Additionally, auto-update functionality in applications often lacks robust integrity verification of updates before applying them, allowing attackers to download and run unauthorized code on installations.

Data objects stored in a way that an attacker can modify are at risk, as attackers can inject malicious code that gets executed. This is known as insecure serialization.

In order to address these issues, applications need to verify the integrity of all third-party components and code to make sure they have not been tampered with. Secure CI/CD pipelines must include threat modeling, automated vulnerability scanning, static code analysis (SAST), dynamic testing (DAST), and regular penetration testing to guarantee the integrity of software releases. Auto-updates should verify the integrity and authenticity of updates through cryptographic validation.

DEFINITION **CI/CD automates code integration, testing, and deployment in software development to ensure quick and frequent updates to production environments.**

To prevent the mentioned threats, make sure to design serialized data structures with integrity and validate them. Using secure software development best practices and a defense-in-depth approach can help reduce software and data integrity failures.

TIP **As a real-life scenario for this kind of vulnerability, I can mention a massive cyberattack against the SolarWinds update procedure.** See `https://orangematter.solarwinds.com/2021/05/07/an-investigative-update-of-the-cyberattack`.

Security Logging and Monitoring Failures

Detecting attacks and breaches relies on proper logging and monitoring of security events. However, many applications do not effectively implement this. Common issues include failing to log important events such as logins, logouts, failed logins, and high-value transactions. Additionally, the clarity and completeness of log messages are often inadequate or missing warnings and errors.

Many applications also fail to monitor their logs for suspicious activity and often only store logs locally instead of in a centralized system. This decentralized storage makes it difficult to analyze and correlate logs.

Moreover, appropriate alerting thresholds and escalation processes for detected threats are often lacking. During pentests and dynamic application security testing, incidents typically do not trigger alerts.

DEFINITION **Dynamic Application Security Testing (DAST) is a way to test live applications for security issues. It finds weaknesses and potential exploits in real time so that they can be fixed to make the application more secure.**

As a result, applications often struggle to detect active attacks in real time or near real time, allowing attackers to go unnoticed for extended periods.

Security logging and monitoring failures leave organizations vulnerable to undetected threats, significantly affecting their ability to respond swiftly and effectively. Real-time monitoring and alerting are necessary for identifying and mitigating potential security incidents as they occur. Ensuring proper logging of auditable events, clear log messages, centralized log storage, real-time log monitoring, effective alerting thresholds, and well-defined response processes to detect and mitigate attacks is essential. Without these controls, applications remain vulnerable.

Server-Side Request Forgery

Server-side request forgery (SSRF) occurs when a web application retrieves remote resources based on user-supplied URLs without properly checking and validating the request. Attackers can use this to send a request to an unexpected location, circumventing network access controls.

Many web applications provide features that involve fetching URLs, increasing the chance of SSRF flaws. An attacker exploiting SSRF can send requests the application was not intended to send, potentially accessing internal services that should be restricted.

Due to complex cloud architectures and microservices, the impact of SSRF is becoming more severe. Attackers may be able to access internal services, databases, and APIs that should be firewalled. This could lead to data exposure, account takeovers, and other compromises.

As a developer or an IT personnel, you play a crucial role in preventing SSRF. Several variants of SSRF exist, each with its own unique risks. A blind SSRF attack merely verifies if a request succeeds without returning data. An internal SSRF targets internal services instead of external ones. An XML SSRF occurs when XML data is returned from the forged request and parsed. To effectively prevent SSRF, it's essential that applications validate all external URLs before making any requests.

SSRF vulnerabilities allow you to exploit functions that retrieve remote resources, potentially sending requests to restricted locations. To defend against this threat, it's important to validate URLs, use whitelisting, and sanitize input properly. Chapter 7 will explore SSRF attacks, different techniques, and how to prevent them to ensure a complete understanding.

Key Takeaways

- Testing a web application's security from an attacker's perspective is more effective than relying on standard methods.

- To effectively test web applications, you must use the proper methods, tools, and techniques to find vulnerabilities.
- A successful web application test usually has five main stages: scoping and reconnaissance, scanning, gaining and maintaining access, analysis, and reporting. The most important stage is reconnaissance.
- The OWASP Top 10 categorizes the most common web application vulnerabilities into 10 categories, making it a key reference point.