OWASP Introduction

Table of Contents

[1 Overview 2](#_Toc96394162)

[1.1 What is OWASP? 3](#_Toc1553405176)

[1.2 OWASP’s Mission 3](#_Toc783616330)

[1.3 What OWASP Offers 3](#_Toc2057086474)

[1.4 Why OWASP Matters 3](#_Toc567791679)

[2 OWASP’s Top 10 4](#_Toc511877972)

[2.1 Broken Access Control 4](#_Toc1697148307)

[2.2 Cryptographic Failures 4](#_Toc1548642623)

[2.3 Injection 5](#_Toc976033916)

[2.4 Insecure Design 5](#_Toc1188481764)

[2.5 Security Misconfiguration 5](#_Toc1038550264)

[2.6 Vulnerable and Outdated Components 5](#_Toc1153714972)

[2.7 Identification and Authentication Failures 6](#_Toc1057512581)

[2.8 Software and Data Integrity Failures 6](#_Toc1181482813)

[2.9 Security Logging and Monitoring Failures 6](#_Toc173742754)

[2.10 Server-Side Request Forgery 6](#_Toc387196808)

[3 OWASP for Risk Management 7](#_Toc1183651481)

[3.1 Risk Identification 7](#_Toc400735530)

[3.2 Risk Assessment 7](#_Toc1741285651)

[3.3 Risk Mitigation 8](#_Toc591458582)

[3.4 Risk Monitoring and Control 8](#_Toc1390841461)

[3.5 Threat Modeling 8](#_Toc1325262562)

[4 Tools by OWASP 8](#_Toc1896360973)

[4.1 ZAP (Zed Attack Proxy) 8](#_Toc943690297)

[4.2 Dependency-Check 9](#_Toc1347266393)

[4.3 OWASP Dependency-Track 9](#_Toc2025540280)

[4.4 OWASP AppSec Pipeline 9](#_Toc977592841)

[4.5 OWASP Threat Dragon 9](#_Toc519495796)

[5 References 9](#_Toc2043764677)

# Overview

OWASP, the Open Web Application Security Project, is a community-driven organization focused on improving the security of software. It provides resources, tools, and guidelines to help organizations and developers create more secure web applications.

## What is OWASP?

OWASP is an open community dedicated to enabling organizations to conceive, develop, acquire, operate, and maintain applications that can be trusted. All their projects, tools, documents, forums, and chapters are free and open to anyone interested in improving application security. The OWASP Foundation was launched on December 1st, 2001, becoming incorporated as a United States non-profit charity on April 21, 2004.

## OWASP’s Mission

Their mission is clear: to make software security visible. By equipping the world with practical tools, methodologies, and knowledge, OWASP aims to level the playing field, creating a secure ecosystem for developers, users, and organizations alike.

## What OWASP Offers

* Top 10 Web Application Security Risks: A curated list of the most critical web application vulnerabilities, serving as a roadmap for developers to prioritize their security efforts.
* Cheat Sheet Series: Quick reference guides covering various security topics, providing bite-sized chunks of knowledge for busy developers.
* Testing Guide: A comprehensive blueprint for conducting security assessments, empowering organizations to identify and address vulnerabilities in their software.
* ZAP: An open-source web application scanner, automating the process of identifying vulnerabilities and providing actionable insights for developers.
* Chapters and Events: A global network of communities and conferences fostering collaboration, knowledge sharing, and continuous learning in the security space.

## Why OWASP Matters

In today's digital world, software security isn't optional, it's essential. OWASP empowers individuals and organizations to:

* Protect sensitive data: From financial information to personal details, OWASP helps shield valuable data from attackers.
* Build trust and confidence: Secure software fosters trust between users and developers, ensuring a positive user experience.
* Minimize risks and costs: Proactive security measures prevent costly data breaches and reputational damage.

# OWASP’s Top 10

The OWASP Top 10 is a standard awareness document for developers and web application security. It represents a broad consensus about the most critical security risks to web applications. Below is the list of top 10 which was released in 2021.

## [**Broken Access Control**](https://owasp.org/Top10/A01_2021-Broken_Access_Control/)

Access Control refers to a system that controls access to information or functionality. Broken access controls allow attackers to bypass authorization and perform tasks as though they were privileged users such as administrators. Access controls can be secured by ensuring that a web application uses authorization tokens\* and sets tight controls on them.

## [**Cryptographic Failures**](https://owasp.org/Top10/A02_2021-Cryptographic_Failures/)

If web applications don’t protect sensitive data such as financial information and passwords, attackers can gain access to that data and sell or utilize it for nefarious purposes. One popular method for stealing sensitive information is using an [on-path attack](https://www.cloudflare.com/learning/security/threats/on-path-attack/). Data exposure risk can be minimized by [encrypting](https://www.cloudflare.com/learning/ssl/what-is-encryption/) all sensitive data as well as disabling the [caching](https://www.cloudflare.com/learning/cdn/what-is-caching/) of any sensitive information.

## [**Injection**](https://owasp.org/Top10/A03_2021-Injection/)

Injection attacks happen when untrusted data is sent to a code interpreter through a form input or some other data submission to a web application. For example, an attacker could enter SQL database code into a form that expects a plaintext username. If that form input is not properly secured, this will result in that SQL code being executed. This is known as an [SQL injection attack](https://www.cloudflare.com/learning/security/threats/sql-injection/).Injection attacks can be prevented by validating and/or sanitizing user-submitted data.

## [**Insecure Design**](https://owasp.org/Top10/A04_2021-Insecure_Design/)

Insecure design isn't a single bug, but a shaky foundation – fundamental security flaws baked into an application's architecture. Think of missing walls, unlocked doors, and exposed data. This leaves it vulnerable to data breaches, hijacking, and lost trust, demanding security woven into every thread of development, not just bolted on later.

## [**Security Misconfiguration**](https://owasp.org/Top10/A05_2021-Security_Misconfiguration/)

Security misconfiguration is the most common vulnerability on the list and is often the result of using default configurations or displaying excessively verbose errors. For instance, an application could show a user overly descriptive errors which may reveal vulnerabilities in the application. This can be mitigated by removing any unused features in the code and ensuring that error messages are more general.

## [**Vulnerable and Outdated Components**](https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components/)

Web developers use components such as third-party libraries and packages in their web applications. Some attackers look for vulnerabilities in these components which they can then use to orchestrate attacks. Component developers often offer security patches. The web developers must update the components to minimize the risks.

## [**Identification and Authentication Failures**](https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/)

Vulnerabilities in authentication (login) systems can give attackers access to user accounts and even the ability to compromise an entire system using an admin account. Some strategies to mitigate authentication vulnerabilities require [two-factor authentication (2FA)](https://www.cloudflare.com/learning/access-management/what-is-two-factor-authentication/) as well as limiting or delaying repeated login attempts using [rate limiting](https://www.cloudflare.com/rate-limiting/).

## [**Software and Data Integrity Failures**](https://owasp.org/Top10/A08_2021-Software_and_Data_Integrity_Failures/)

Malicious attacks or coding errors can corrupt code and data, leading to inaccurate information, compromised systems, and broken functionality. From financial miscalculations to medical blunders, the consequences can be severe. To avoid this, we need to have secure software, validate input, data integrity checks and diligent system monitoring.

## [**Security Logging and Monitoring Failures**](https://owasp.org/Top10/A09_2021-Security_Logging_and_Monitoring_Failures/)

Many web applications are not taking enough steps to detect data breaches. The average discovery time for a breach is around 200 days (about 6 and a half months) after it has happened. This gives attackers a lot of time to cause damage before there is any response. OWASP recommends that web developers should implement logging and monitoring as well as incident response plans to ensure that they are made aware of attacks on their applications.

## [**Server-Side Request Forgery**](https://owasp.org/Top10/A10_2021-Server-Side_Request_Forgery_%28SSRF%29/)

Server-Side Request Forgery (SSRF) is a crafty attack where an attacker manipulates a web application into making unauthorized requests to other servers. Think of it as a Trojan horse hidden in a URL, tricking the application to fetch sensitive data or perform actions the attacker desires. This can expose internal systems, steal sensitive information, or even hijack crucial services. To stay safe, validate user-supplied URLs and restrict access to internal resources, so your trusty messenger only delivers requests where they belong.

# OWASP for Risk Management

OWASP, while primarily focused on providing guidance and tools for web application security, is inherently related to risk management within the realm of cybersecurity. Here's how OWASP intersects with risk management

## Risk Identification

OWASP's resources, such as the OWASP Top 10, provide a comprehensive list of common vulnerabilities and security risks in web applications. These resources help in identifying potential risks that applications might face. Understanding these risks is the first step in managing them effectively.

## Risk Assessment

OWASP tools like ZAP and Dependency-Check aid in assessing and evaluating risks by scanning applications for vulnerabilities. These assessments provide insights into the potential impact and likelihood of various security issues.

## Risk Mitigation

OWASP's recommendations, best practices, and tools assist in mitigating identified risks. By following secure coding practices, applying fixes for vulnerabilities, and using secure development frameworks, organizations can reduce the likelihood and impact of security threats.

## **Risk Monitoring and Control**

OWASP tools that offer continuous monitoring capabilities, like Dependency-Track and AppSec Pipeline, help in monitoring risks associated with third-party dependencies and in maintaining control over the security posture of applications. This continuous monitoring enables organizations to identify and address new risks as they arise.

## **Threat Modeling**

OWASP's tools like Threat Dragon facilitate threat modeling, which is an essential part of risk management. Threat modeling helps in proactively identifying potential threats and vulnerabilities during the design phase of an application, allowing for preemptive risk management strategies.

# Tools by OWASP

OWASP offers various tools designed to assist in different aspects of web application security testing and enhancement. Here are some prominent ones and how you can use them to bolster your web application's security:

## ZAP (Zed Attack Proxy)

ZAP is a widely used open-source security testing tool. It helps in finding security vulnerabilities during the development and testing phases. You can use ZAP to intercept and modify HTTP/HTTPS requests, perform automated scanning for common vulnerabilities like XSS (Cross-Site Scripting), SQL Injection, and more.

**How to use:** Start by configuring ZAP as a proxy for your web application. Then, perform various attacks and scans through ZAP to identify vulnerabilities. Analyze the results and apply fixes to enhance security.

## Dependency-Check

This tool helps identify known vulnerabilities in project dependencies, such as libraries and frameworks.

**How to use:** Integrate Dependency-Check into your build process or development pipeline. It scans your dependencies for known vulnerabilities and provides a report highlighting issues that need attention. Update or replace vulnerable dependencies accordingly.

## OWASP Dependency-Track

This is a platform that helps manage and monitor the use of third-party components in applications.

**How to use:** Use Dependency-Track to track the usage of third-party components, monitor their vulnerabilities, and receive alerts for new vulnerabilities. It aids in making informed decisions about which components to use or update.

## OWASP AppSec Pipeline

This tool is designed to create an automated security pipeline for web applications, integrating security checks into your CI/CD (Continuous Integration/Continuous Deployment) process.

**How to use:** Configure the AppSec Pipeline to automate security checks, including scanning for vulnerabilities, code analysis, and compliance testing, at various stages of your development pipeline.

## OWASP Threat Dragon

This is a threat modeling tool that helps in identifying threats and mitigating risks during the design phase of your application.

**How to use:** Use Threat Dragon to create visual threat models that represent your application's architecture and potential risks. It assists in understanding and addressing security concerns early in the development lifecycle.

# References

1. <https://owasp.org/>
2. <https://en.wikipedia.org/wiki/OWASP>
3. <https://www.cloudflare.com/learning/security/threats/owasp-top-10/>
4. <https://owasp.org/www-project-top-ten/>