OWASP Top 10 Vulnerabilities

11 mins read

Cloud native applications, with their distributed architectures that comprise many third-party libraries and services, are an attractive target for hackers. The fact that 82% of all vulnerabilities are found in application code is not lost on attackers, who seek to use this vector to compromise the networks on which the application is deployed. Securing web applications, therefore, has become a business-critical requirement.

What is OWASP?

The Open Web Application Security Project (OWASP) is a non-profit global community that strives to promote application security across the web. A core OWASP principle is that their knowledge base is freely and easily accessible on their website. With its tens of thousands of members and hundreds of chapters, OWASP is considered highly credible, and developers have come to count on it for essential **web application security**, and **API security** guidance.

Every application developer, regardless of experience level, must make the effort to understand code security vulnerabilities in order to avoid frustrating and often costly **application security** failures.

What is the OWASP Top 10?

Every few years, OWASP revises and publishes its list of the top 10 web **application vulnerabilities**. The list includes not only the OWASP Top 10 threats but also the potential impact of each vulnerability and how to avoid them. The comprehensive list is compiled from a variety of expert sources such as security consultants, security vendors, and security teams from companies and organizations of all sizes. It is recognized as an essential guide

to web application security best practices.

OWASP has recently shared the 2021 OWASP Top 10 where there are three new categories, four categories with naming and scoping changes, and some consolidation within the Top 10.

The OWASP Top 10 is largely intended to raise awareness. However, since its debut in 2003, enterprises have used it as a de facto industry AppSec standard. If we look at the document closely, it specifically calls out the number of **CWE's** (Common Weakness Enumeration) attached with it.

The Latest List of OWASP Top 10 Vulnerabilities and Web Application

Security Risks

The **newest OWASP Top 10** list came out on September 24, 2021 at the OWASP 20th Anniversary. If you're familiar with the 2020 list, you'll notice a large shuffle in the 2021 OWASP Top 10, as **SQL injection** has been replaced at the top spot by Broken Access Control.

1. **Broken Access Control**
2. **Cryptographic Failures**
3. **Injection**
4. **Insecure Design**
5. **Security Misconfiguration**
6. **Vulnerable and Outdated Components**
7. **Identification and Authentication Failures**
8. **Software and Data Integrity Failures**
9. **Security Logging and Monitoring Failures**
10. **Server-Side Request Forgery**

OWASP Top 10 Vulnerabilities

In this section, we explore each of these OWASP Top 10 vulnerabilities to better understand their impact and how they can be avoided.

1. Broken Access Controls

Website security access controls should limit visitor access to only those pages or sections needed by that type of user. For example, administrators of an ecommerce site need to be able to add new links or add promotions. These functions should not be accessible for other

types of visitors.

Developers must be encouraged to internalize “security first” discipline to avoid pitfalls, such as content management systems (CMS) that generate all-access permission by default (up to and including admin-level access). Broken access control can give website visitors access to admin panels, servers, databases, and other business-critical applications. In fact, this OWASP Top 10 threat could even be used to redirect browsers to other targeted URLs.

Broken Access Controls Remediation

Broken access control vulnerability can be addressed in a number of ways:

- Adopt a least privileged approach so that each role is granted the lowest level of access required to perform its tasks.
- Delete accounts that are no longer needed or active.
- Audit activity on servers and websites so that you are aware of who is doing what (and when).
- If there are multiple access points, disable the ones that are not required at that moment.
- Keep servers lean by shutting down unnecessary services.

2. Cryptographic Failures

Data in transit and at rest — such as passwords, credit card numbers, health records, personal information, and business secrets — require extra protection due to the potential for cryptographic failures (sensitive data exposures). This is especially true if the data falls under any of the privacy laws such as GDPR, CCPA, and others. Is any data is sent in plain text? Are there any outdated or insecure cryptographic algorithms or protocols in use by default or in older code? Is it possible that default crypto keys are being utilized, that weak crypto keys are being generated and re-used, or that proper key management and rotation

are being overlooked? Is it possible to check crypto keys into source code repositories? Is encryption not enforced, and is the received data encrypted?

Cryptographic Failures Remediation

- On forms that collect data, turn off autocomplete.
- Reduce/minimize the size of the data surface area.
- Encrypt data while it is in transit and at rest.
- Use the most up-to-date encryption techniques.
- Disable caching on data-collecting forms.
- Use Strong adaptive and salted hashing functions when saving passwords.

3. Injection

Injection vulnerabilities can occur when a query or command is used to insert untrusted data into the interpreter via **SQL**, OS, NoSQL, or LDAP injection. The hostile data injected through this attack vector tricks the interpreter to make the application do something it was not designed for, such as generating unintended commands or accessing data without proper authentication.

Any application that accepts parameters as input can be susceptible to injection attacks. The level of the threat is highly correlated with the thoroughness of the application's input validation measures.

Injection Remediation

Injection attacks can be prevented by any combination of the following approaches:

- Segregate commands from data to avoid exposure to attacks that replace data with

unintended command execution.

- Code SQL queries with parameters rather than structuring the command from user input content only. These are called parameterized queries or prepared statements.
- Eliminate the interpreter altogether through the use of a safe API.
- Implement positive server-side validation as well as an intrusion detection system that spots suspicious client-side behaviors.

4. Insecure Design

Insecure design is a wide term that encompasses a variety of flaws and is defined as "missing or poor control design." Threat modeling, secure design patterns, and reference architectures are among the new categories for 2021, with a demand for increasing the usage of threat modeling, safe design patterns, and reference architectures. As a community, we must move beyond "shift left" coding to pre-code tasks that are important to the Secure by Design principles.

Insecure Design Remediation

- To help analyze and build security and privacy-related measures, establish and use a safe development lifecycle with AppSec professionals.
- Create and use a library of secure design patterns or components that are ready to use.
- Use threat modeling for crucial authentication, access control, business logic, and key flows.
- User stories should include security language and controls.
- Integrate plausibility checks into your application at each level (from frontend to backend).
- To ensure that all important flows are resistant to the threat model, write unit and integration tests. Make a list of use-cases and misuse-cases for each tier of your app.
- Depending on the exposure and protection requirements, divide tier tiers on the system

and network layers.

- Limit user and service resource consumption.

5. Security Misconfiguration

Gartner estimates that **up to 95%** of cloud breaches are the result of human errors. Security setting misconfigurations are one of the prime drivers of that statistic, with OWASP noting that, of the top ten, this vulnerability is the most common. There are many types of misconfiguration that expose the company to cybersecurity risk, including:

- Accepting default settings that are insecure
- Overly accessible cloud storage resources
- Incomplete configurations
- Misconfigured HTTP headers
- Verbose error messages that contain sensitive information

Security Misconfiguration Remediation

Security misconfigurations can strike almost anywhere in the environment, including network-attached devices, databases, web and application servers, and containers. The following practices can help maintain a well-configured environment:

- Use templates to deploy development, test, and production environments that are preconfigured to meet the organization's security policies.
- Leverage segmented application architectures that minimize the risk from an insecurely configured element; maintain a library of properly configured container images.
- Deploy minimal platforms and remove unused features and services.
- Continuously monitor cloud resources, applications, and servers for security misconfigurations and remediate detected issues in real time, using automated

workflows wherever possible.

6. Vulnerable and Outdated Components

Modern distributed web applications often incorporate open source components such as libraries and frameworks. Any component with a known vulnerability becomes a weak link that can impact the security of the entire application.

Although the use of open source components with known vulnerabilities ranks low in terms of security problem severity, it is #1 when ranking the OWASP Top 10 by how often a vulnerability was the **root cause of an actual data breach**.

Vulnerable and Outdated Components Remediation

The most effective defense is continuous **scanning of all code components** for known vulnerabilities and deploying a patch or other remedy as quickly as possible when a vulnerability is detected. There are a number of best practices that enhance the effectiveness of this line of defense:

- All components integrated into the company's frameworks should be under configuration management.
- The scanner must be able to automatically discover all the components to be monitored.
- Scanning should be conducted against a comprehensive **vulnerability database** that is enriched with **threat intelligence** data.
- The patch management workflows for identifying, testing, and deploying the right patch should be as automated as possible in order to reduce to a minimum the operational risk associated with patching.

7. Identification and Authentication Failures

When applications incorrectly execute functions related to session management or user authentication, intruders may be able to compromise passwords, security keys, or session tokens and permanently or temporarily assume the identities and permissions of other users. This vulnerability poses a grave threat to the security of the application and the resources it accesses and can also severely compromise other assets connected to the same network.

Authentication Remediation

The key OWASP best practice recommendations to mitigate broken authentication vulnerabilities are:

- Implement **multi-factor authentication**.
- Do not deploy with default credentials, especially for users with admin privileges.
- Enforce strong passwords.
- Carefully monitor failed login attempts.
- Use a secure session manager that generates random, time-limited session IDs. Never include session IDs in URLs.

8. Software and Data Integrity Failures

Code and infrastructure that do not guard against integrity violations are referred to as software and data integrity failures. A program that uses plugins, libraries, or modules from untrusted sources, repositories, or content delivery networks (CDNs) is an example of this. Unauthorized access, malicious code, or system compromise can all be risks of an unsecured CI/CD pipeline. Finally, many programs now have auto-update capabilities that allow updates to be obtained without necessary integrity checks and applied to previously trusted applications. Attackers could potentially distribute and run their own updates across all systems with this functionality.

Software and Data Integrity Failures Remediation

- Use digital signatures, or other similar measures, to ensure that the program or data is genuine and has not been tampered with.
- To reduce the risk of harmful code or configuration being introduced into your development pipeline, make sure there is a review procedure in place for code and configuration modifications.
- Ascertain that libraries and dependencies, such as npm or Maven, use trusted repositories. Consider hosting an internal, approved known-good repository if you have a higher risk profile.
- To protect the integrity of the code going through the build and deploy processes, make sure your CI/CD pipeline includes adequate segregation, configuration, and access control.
- Ensure that unsigned or unencrypted serialised data is not delivered to untrustworthy clients without some kind of integrity check or digital signature to detect alteration or replay.

9. Insufficient Logging and Monitoring

Studies indicate that the time from **attack to detection can take up to 200 days**, and often longer. This window gives cyber thieves plenty of time to tamper with servers, corrupt databases, steal confidential information, and plant malicious code.

Remediation

Implement readily available logging and audit software to quickly detect suspicious activities and unauthorized access attempts. Even if a detected attack has failed, logging and monitoring provide invaluable tools for analyzing the source and vector of the attack and learning how security policies and controls can be hardened to prevent intrusions.

10. Server-Side Request Forgery (SSRF)

Server-side request forgery (also termed as SSRF) is a web security flaw that allows an attacker to force a server-side application to send HTTP requests to any domain the attacker chooses.

When a web application fetches a remote resource without validating the user-supplied URL, an SSRF fault occurs. Even if the program is secured by a firewall, VPN, or another sort of network access control list, an attacker can force it to send a forged request to an unexpected location.

Remediation

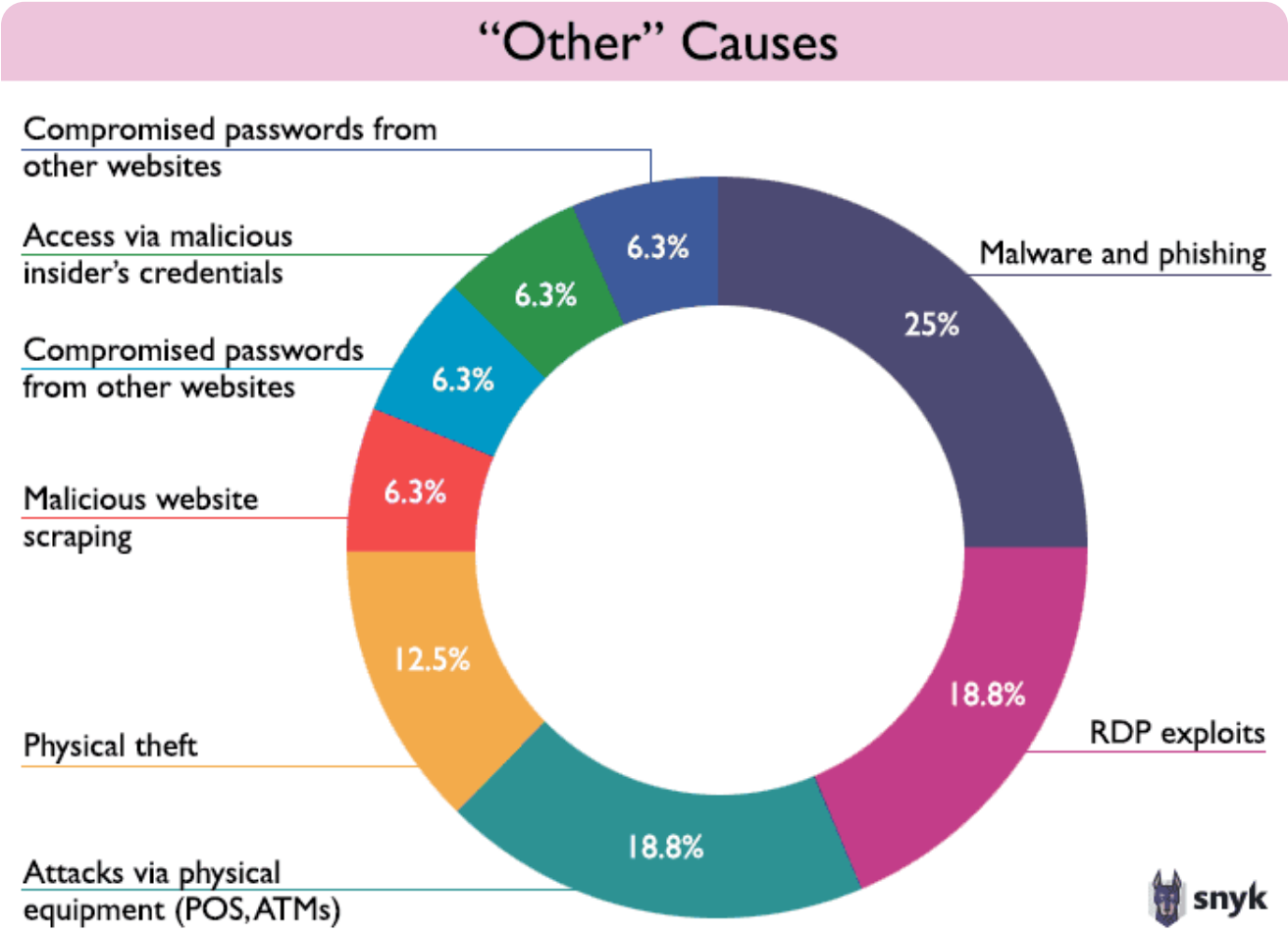
- Implement input validation.
- Use Regular Expressions (RegEx).
- Only accept the intended IP address format (IPv4 or IPv6).
- To compare against the allow list, use the method/output library's value as the IP address.
- Validate incoming Domain Names.
- Review the OWASP **Cheat Sheet Series**

What are the other Non-OWASP vulnerabilities?

OWASP states very clearly in their methodology that the Top 10 list is, by definition, only a subset of important security issues and organizations should be aware of additional security risks.

You should maintain awareness of other new vulnerabilities discovered in the wild such as

the **Log4Shell Vulnerability**, which was disclosed in December 2021.



Other non-OWASP Vuln chart

OWASP Vulnerabilities: FAQs

What Is an OWASP Vulnerability?

OWASP vulnerabilities are security weaknesses or problems published by the Open Web Application Security Project. Issues contributed by businesses, organizations, and security professionals are ranked by the severity of the security risk they pose to web applications.

What Are the Top 10 OWASP Vulnerabilities?

OWASP's top ten list is compiled and published every three to four years, highlighting the most critical security vulnerabilities. Additionally, the list includes examples of the

weaknesses, how they can be exploited by attackers, and suggested methods that reduce or eliminate application exposure.

What Is the number 1 Application Security Risk Reported by OWASP?

Injection is the number 1 flaw reported by OWASP. Injection can send untrusted data through SQL or other paths such as LDAP, allowing the interpreter to access unauthorized data or execute commands not intended by the application.

How Can OWASP Top 10 Vulnerabilities Be Tested?

OWASP provides an in-depth testing guide that offers test cases for a multitude of test scenarios. Many development teams have adopted a more automated solution by utilizing software to scan code for vulnerabilities with automated warnings and consistent application of best practices.

How Should the OWASP Top 10 Be Used?

OWASP's top 10 list offers a tool for developers and security teams to evaluate development practices and provide thought related to website application security. While it is by no means all-inclusive of web application vulnerabilities, it provides a benchmark that promotes visibility of security considerations.

UP NEXT

SQL Injection (SQLi) Attack

SQL injection is one of the most common methods of extracting unauthorized data from commercial websites. Learn how SQLi works, and how to prevent it.

Keep reading →

Cybersecurity Exploits

RELATED ARTICLES

OWASP API Security Top 10 Risks

What Are Security Misconfigurations and How Can You Prevent Them?

Security Vulnerability: types and remediation

What is CVE? Keeping Track of Vulnerabilities with CVE Vulnerability

Database
Securing Front-end Attack Surfaces

Sour Mint - The case of malicious advertisement SDK affecting thousands of mobile apps

OWASP Top 10 Vulnerabilities

SQL Injection (SQLi) Attack

Man-in-the-Middle (MITM) Attack

Malicious Code Explained

Cross-Site Scripting (XSS)

Cross Site Request Forgery (CSRF)

Snyk CLI

Company

About

Customers

Careers

Events

Snyk Impact

Snyk for government

Press kit

Security & trust

Legal terms

Privacy

For California residents: Do not sell my personal information

Connect

Book a live demo

Contact us

Support

Report a new vuln

Security

JavaScript security

Container Security

Kubernetes Security

Application Security

Open Source Security

Cloud Security

Secure SDLC

Cloud Native Security

Secure coding

Python Code Examples

JavaScript Code Examples

Code Checker