April 15, 2022

11 min read

TOP 10 OWASP VULNERABILITIES: WHAT THEY'RE ALL ABOUT AND HOW TO DEAL WITH THEM

While most everybody in today's world is going online at least a dozen times a day, a lot of personal data is exposed and all sorts of internet threats increasingly put user info and hardware at risk. To protect users globally, specialized security and data safety regulations are introduced by local or international authorities and related communities like the Open Web Application Security Project or OWASP.

A non-profit organization governing application security worldwide, OWASP has been updating and introducing OWASP's Top 10 Security Vulnerabilities since the early 2000s. Over the years, it has provided security guidelines that enterprises and entrepreneurs alike take close heed of. And there are more than significant reasons for that—Nix specialists invite you to dive into the specifics of the project and see particular vulnerabilities and ways it promotes optimization.

About OWASP

OWASP has a centralized knowledge base dedicated to application development and operation safety measures and standards that is freely

Contributed to by tens of thousands of community members, the extensive knowledge base packed into the OWASP website is considered very reputable and is trusted by developers operating across industries worldwide.

OWASP's Top 10 Vulnerabilities List: the Latest Update and Current Contents

Top 10 OWASP vulnerabilities is sort of a community digest of the latest essential app development security threats, risks, and potential flaws and issues, and is regularly revised and updated to help developers keep up with the most relevant software protection approaches.

The list is subdivided into security flaw classes, each outlining not only vulnerability descriptions but also related best practices, references

The latest stretch of major updates took place on September 24, 2021—three new classes have been added to the list and four existing classes have been renamed, while some other categories have been revised and merged.

Thus, “Broken Access Controls” has been moved from the fifth place in the list up to the top class of potentially severe web app security hazards, now topping the list. The decision was based on the results of analysis indicating over 300,000 app security issue occurrences, the majority of which is related specifically to poor access control issues.

The description and remediation techniques outlined in the previously existing class titled “Sensitive Data Exposure” have been translated into more pinpointed chapters under “Cryptographic Failures.” As usual, the section is focused on data integrity and the prevention of info leaks of all sorts.

The “Injection” class has dropped to the third place in the list—over 95% of applications checked for some type of data injection issues demonstrated a peak occasion indicator of 20%. As such, cross-site scripting (XSS) has been outlined as a chapter in “Injection.”.

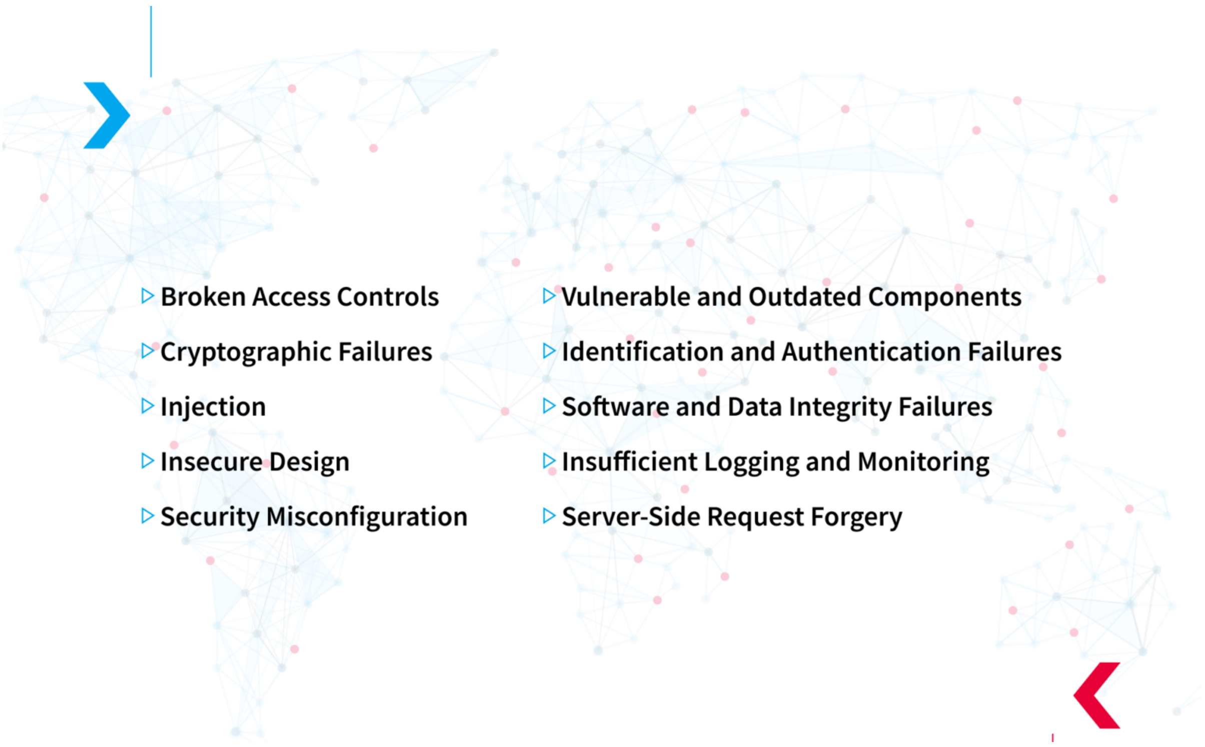
“Insecure Design” is a renewed class in the 2021 edition of OWASP Top 10 Security Vulnerabilities that focuses on various security threats spawned by software design flaws. The class outlines a number of design-dictated dangers, functional models and rules, and even software architecture-defining standards. The main goal is to help developers and entrepreneurs achieve consistent design and realize the importance of scalable architectures (an application must be susceptible to major changes and upgrades from the get-go; otherwise, you can’t really fix what’s been broken).

“Security Misconfiguration” went up from 6th to 5th place in the list. Across 100% of apps checked for various misconfigurations, the security fault occurrence coefficient made up about 5%. The relevance of this class been merged with the updated class.

“Using Components with Known Vulnerabilities” is now known as “Vulnerable and Outdated Components”—the only class to not have any Common Vulnerability and Exposures (CVE) connected to the underlying CWEs.

Then, there is “Identification and Authentication Failures” (formerly known as “Broken Authentication” and now containing CWEs that are more focused on identity verification errors), “Software and Data Integrity Failures” (introduced in the 2021 OWASP list edition and merging former “Insecure Deserialization” class), “Security Logging and Monitoring Failures” (renamed from “Insufficient Logging and Monitoring”), and “Server-Side Request Forgery” (an interesting class with the lowest issue occurrence rates yet eagerly pushed by the community).

OWASP Top 10 Security Vulnerabilities Right Now


Broken Access Controls
Cryptographic Failures
Injection
Insecure Design
Security Misconfiguration
Vulnerable and Outdated Components
Identification and Authentication Failures
Software and Data Integrity Failures
Insufficient Logging and Monitoring
Server-Side Request Forgery

Top 10 Vulnerabilities are highly valued by the global community of developers and market players operating across industries and niches. It is a real "app security Bible" for security engineers of all ranks. Let's see what the list currently packs to set sturdy global app protection standards.

#1 Broken Access Controls

It is important to set your user roles straight and provide managed access to content intended for respective types of users. For instance, web app admins must have a higher level of access to backend areas of the application, admin dashboards, debugging tools, and such. Accordingly, regular users don't need this and must only get limited frontend access to the user-focused functionality.

Such permissions must be closely governed. Employment of content management systems (CMS) helps a lot while you may tackle broken access controls issues by:

- Setting up user roles, access permissions, and privileges
- Automating session time-outs and cleaning up inactive user accounts
- Monitoring and auditing activity on servers and in web apps
- Managing access points and shutting down unused ones
- Managing services and disabling unused ones

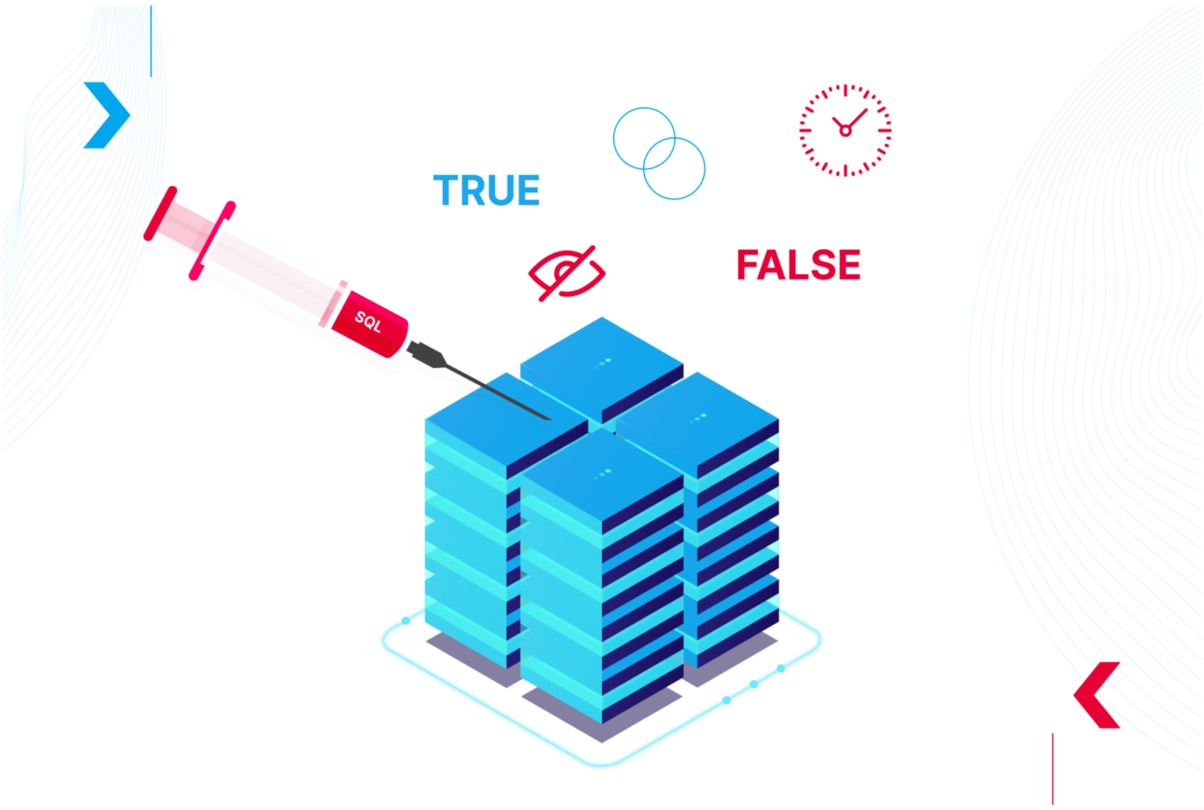
#2 Cryptographic Failures

All sorts of sensitive data (including payment credentials, enterprise secrets, personal data, passwords, health records, etc.) stored in the company or private databases need an extra level of protection from unauthorized access attempts and exposure. This is why cryptographic encryption, and more.

Here's how OWASP recommends tackling cryptographic failures:

- Disabling autocomplete and caching functions in forms that gather data
- Minimizing data surface area
- Encrypting data in both idle and transfer states
- Employing the latest encryption models

#3 Injection



SQL injection is a common web app threat where a cybercriminal (or any other party with malicious intents) uses a command or query in order to inject some faulty data into the app's interpreter. This allows them to take certain control over the app and make it execute specific functions, opening more ways to undermine performance and steal data. And the

What you can do to fend off such an issue and create an efficient line of defense against injections includes:

- Coding prepared statements/parameterized queries (parameter-powered SQL queries)
- Using a safe API in order to eliminate faulty interpreter operation
- Introducing intrusion detection algorithms and positive server-side validation functionality

#4 Insecure Design

This class covers a vast area of possible design flaws and errors. The updated list highlights secure reference architectures, design patterns, and threat modeling as separate categories in this class. A major design and coding paradigm shift is currently promoted by the global community, outlining the importance of handling pre-code tasks in order to achieve all-around consistent designs.

You can start design-reinforcing practices right now by:

- Introducing a library of high-security, ready-to-use components or design patterns
- Employing threat modeling in terms of access control, business logic management, authentication, and other aspects susceptible to security risks
- Integrating frontend-to-backend plausibility checks
- Creating and using integration and unit tests to achieve threat-resistant flows of data
- Limiting resource consumption rates for both services and users

Human factor error is a major common reason for data breaches (especially in cloud environments). And misconfiguration of certain security settings is the most common protection breach occurrence mentioned by OWASP. Whether insecure default settings are used, written faults are made in sensitive descriptions, or configurations are left incomplete, you may tackle this class of issues by:

- Preconfiguring development, testing, and production environments —based on this, environment deployment templates can be created and used
- Segmenting architectures and configuring solutions brick-by-brick
- Removing unnecessary services and features as well as deploying minimal platforms
- Scrupulously monitoring servers, software resources, and apps as a whole for efficient misconfiguration detection

#6 Vulnerable and Outdated Components

Open-source frameworks and libraries, as well as other publicly available components and elements, have known vulnerabilities and yet they are very commonly used by developers and entrepreneurs. Even a single open-source component that doesn’t fall under a reinforced security measure of some sort can undermine the whole application. Such components are not, however, among the top threats on a global scale, but are still important.

In order to reinforce this particular layer of security, you may try:

- Introducing configuration management for each and every component and integration application flaws via timely patching

#7 Identification and Authentication Failures

Faulty user authentication or error-prone session management may pose some of the nastiest application sabotage and corruption threats. Cybercriminals may easily get their hands on sensitive, enterprise-defining data by indulging in an authentication failure of some sort and making a successful attempt at unauthorized access.

OWASP's best practices and recommendations in this matter include the following:

- Enabling multi-factor authentication
- Restraining from default credentials (admins should get specialized access rights, as mentioned above)
- Thoroughly tracking failed logging in attempts
- Promoting strong, complex passwords
- Employing session time-outs with the help of a reliable session manager

#8 Software and Data Integrity Failures

This class of failures covers a range of potential flaws that may appear throughout all parts of code architecture and app infrastructure. And those flaws are commonly provoked by either an insufficiently secure CI/CD pipeline (which opens possibilities for unauthorized access and harmful coding), the introduction of components from unconfirmed sources (e.g., plugins or modules from untrusted CDNs or repositories), or auto-update functions (that deliver updates that were approved by the previous use but didn't go through up-to-date verification).

In order to tackle all of the above issues, OWASP recommends you try:

- Introducing a necessary code and configuration reviewing procedure for all related changes and updates
- Employing digital signatures and keys to further reinforce software access integrity
- Using only trusted component repositories and sources
- Thoroughly segregating, managing, and configuring the CI/CD pipeline

#9 Insufficient Logging and Monitoring

Supplementing the above-mentioned "Broken Access Controls" and "Identification and Authentication Failures" classes, this one emphasizes the importance of sturdy user access protection for the safety of one's servers, databases, code architectures, and sensitive credentials

For this, OWASP outlines specialized practices which include:

- Accelerating suspicious activity detection by using solutions for logging and user routine checks
- Analyzing vectors and sources of successful or potential attacks through scrupulous monitoring of actions
- Reinforcing intrusion-resistant algorithms and hardening logging and user session security policies

#10 Server-Side Request Forgery

Last but not least, server-side request forgery (SSRF) is a malicious cyber criminal's trick where hackers manage to make a server-side app send HTTP requests anywhere they wish (i.e., to any specified domain). This flaw occurs when the user-specified URL isn't validated by a web application in the process of fetching a remote resource. Forged requests may be redirected even if a firewall or VPN protection is in place.

What you can do to eliminate the threat, though, includes but not limits to:

- Enabling thorough validation of input data
- Validating input domain names
- Restricting access to IPv4 and IPv6 (only intended IP address formats)
- Using RegEx (Regular Expressions)
- Regularly taking a good look through OWASP's SSRF cheat sheet

Bottom Line

date practices and guidelines. And if you are having trouble finding your way around all of these tech recommendations, contact experienced specialists for a pre-consultation. In either case, OWASP standards are there to help you achieve the most secure, consistent, and long-term lasting results. So try not taking them for granted.

Recommended articles for Software Engineering

PhoneGap End of Development Announcement – What Are Your Options Now?

With support for PhoneGap ending in October, developers and businesses need to find an alternative for their app development needs.

BLOG SOFTWARE ENGINEERING