# **OWASP Top Ten**

(Open web application Security Project)

The open web application security project is a non-profit organization founded in 2002, with the goal of helping website owners and security experts protect web applications from cyber attack.

## What is OWASP Top 10:

* Top 10 is a regularly-updated report outlining security concerns for web application security, focusing on the 10 most critical risks. OWASP refers to the top 10 as an ‘awareness’ and they recommend that all companies incorporate the report into their processes in order to minimize security risks.

## Owasp Top 10 2017:

A01: Injection

A02: Broken authentication

A03: Sensitive data exposure

A04: XML External Entities

A05: Broken Access Control

A06: Security Misconfiguration

A07: Cross-site Scripting(XSS)

A08: Insecure Deserialization

A09: Using Component with known vulnerabilities

A10: Insufficient logging & Monitoring.

### A07 What is cross site scripting:(#7)

Is a web based vulnerability that allows an attacker to run scripts, using it an attacker can modify, manipulate data.

eg.DOM based, Reflected XSS, cookie-based-XSS

### A01 What is injection:(#1)

Is a web based vulnerability generally happens when an user try to perform forceful input i.e injection : ‘ , “

Eg. SQL injection,LDAP injection.

### 

### A02 What is Broken authentication:(#2)

Caused by implementation of poor authentication @ session management functions.

Is a broken auth scenarios a user can steal session id, password keys, tokens, & juicy information about any user

Eg. password breach, credential leakage.

## OWASP TOP 10 2021

1. Broken Access control. (#5 in 2017)
2. Cryptographic failures (#3 in 2017)(Sensitive Data exposure)
3. Injection (#1 in 2017)
4. Insecure design (New Risk in 2021)
5. Security misconfiguration (#6 in 2017)
6. Vulnerable and Outdated component (#9 in 2017)(Using Component with vulnerabilities)
7. Identification and authentication (#2 in 2017)(Broken Authentication)
8. Software and data integrity failures (new)
9. Security logging and monitoring failures. (#10 in 2017)(Insufficient logging & Monitoring)
10. server -side request forgery(SSRF) (new)

### #A01 - Broken Access control

It is highest risk in 2021.

* Ineffective access control
* Non-user access user account
* Non-admin user accesses admin account.

How to prevent

* By default all permission denied.
* Strong access control mechanism.

### #A02 - Cryptographic failures

* Revealing sensitive information

Credit card no., password etc.

* Ineffective crypto method/algorithm

How to prevent

* Don't store sensitive info unnecessarily
* Make sure to encrypt all such data
* Do not use legacy protocols to transfer such information
* Always use authenticated encryption

### #A03 - Injection

* Command inside/ instead of data
* User supplied data is validated / filtered
* Eg. SQL.

Name contains drop table command

Select \* from TABLE where id=123 or 1=1

How to prevent

* Validate data.

### #A04 - Insecure design

* Missing or ineffective control design
* Not source of other 9 risks
* Lack of business risk profiling

How to prevent

* Secure SDLC(following best practices).
* Library of secure designing

### #A05 - Security Misconfiguration

* Unnecessary features are enabled or installed
* Default accounts and their passwords are still enabled and unchanged

How to prevent

* A minimal platform without any unnecessary features, components, documentation, and samples
* Remove or do not install unused features and frameworks

### #A06 - Vulnerable & Out dated components

* the software is vulnerable, unsupported, or out of date.
* If you do not secure the components’ configurations

How to prevent

* Remove unused dependencies, unnecessary features, components, files, and documentation
* Only obtain components from official sources over secure links.

### #A07 - Identification & Authentication

* Permits default, weak, or well-known passwords, such as "Password1" or "admin/admin"
* Has missing or ineffective multi-factor authentication.

How to prevent

* Implement Multi Factor authentication

### #A08 - Software & Data Integrity Failures

* plugins, libraries, or modules from untrusted sources, repositories

How to prevent

* Use digital signatures to verify the software
* Ensure Dependencies & Libraries

### #A09 - Security logging & monitoring Failures

* unclear log messages
* Logs are only stored locally.

How to prevent

* Ensure All login

### #A10 - Server Side Request Forgery

* Bad Request

How to prevent

* Validate All the requests.

# CWE/SANS Top 25

The CWE Top 25 is the work of the Homeland Security Systems Engineering and Development Institute, sponsored by CISA and operated by MITRE.

* SANS Stands For SysAdmin, Audit, Network & Security.
* CWE Stands For Common Weakness Enumeration

The Common Weakness Enumeration is a category system for hardware and software weaknesses and vulnerabilities.

## CWE Top 25 List

### #1 CWE-787 Out-of-bounds Write.

The software writes data past the end, or before the beginning, of the intended buffer.

this can result in corruption of data, a crash, or code execution.

### #2 CWE-79 Improper neutralization of input during web page generation (Cross-site-scripting)

The software does not neutralize or incorrectly neutralizes user-controllable input before it is placed in output that is used as a web page that is served to other users.

### #3 CWE-89 Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')

The software constructs all or part of an SQL command using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the intended SQL command when it is sent to a downstream component.

* SQL injection is the placement of malicious code in SQL statements, via web page input.

### #4 CWE-20 Improper Input Validation

The product receives input or data, but it does not validate or incorrectly validates that the input has the properties that are required to process the data safely and correctly.

### #5 CWE-125 Out-of-bounds Read

The software reads data past the end, or before the beginning, of the intended buffer.

### #6 CWE-78 Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')

The software constructs all or part of an OS command using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the intended OS command when it is sent to a downstream component.

### #7 CWE-416 Use After Free

Referencing memory after it has been freed can cause a program to crash, use unexpected values, or execute code.

### #8 CWE-22 Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')

The software uses external input to construct a pathname that is intended to identify a file or directory that is located underneath a restricted parent directory, but the software does not properly neutralize special elements within the pathname that can cause the pathname to resolve to a location that is outside of the restricted directory.

### #9 CWE-353 Cross-site request forgery

The software uses external input to construct a pathname that is intended to identify a file or directory that is located underneath a restricted parent directory, but the software does not properly neutralize special elements within the pathname that can cause the pathname to resolve to a location that is outside of the restricted directory.

### #10 CWE-434 Unrestricted Upload of File with Dangerous Type

The software allows the attacker to upload or transfer files of dangerous types that can be automatically processed within the product's environment.

### #11 CWE-476 NULL Pointer Dereference

The software allows the attacker to upload or transfer files of dangerous types that can be automatically processed within the product's environment.

### #12 CWE-502 Deserialization of Untrusted Data

The application deserializes untrusted data without sufficiently verifying that the resulting data will be valid.

### #13 CWE-190 Integer Overflow or Wraparound

The software performs a calculation that can produce an integer overflow or wraparound, when the logic assumes that the resulting value will always be larger than the original value. This can introduce other weaknesses when the calculation is used for resource management or execution control.

### #14 CWE-287 Improper Authentication

Failure to verify a user's identity results in improper authentication. This can allow an attacker to acquire privileges to access sensitive data in your application.

### 

### #15 CWE-798 Use of Hard-coded Credentials

Hardcoded Passwords, also often referred to as Embedded Credentials, are plain text passwords or other secrets in source code. Password hardcoding refers to the practice of embedding plain text (non-encrypted) passwords and other secrets (SSH Keys, DevOps secrets, etc.)

### 

### #16 CWE-862 Missing Authorization

The software does not perform an authorization check when an actor attempts to access a resource or perform an action

### 

### #17 CWE-77 Improper Neutralization of Special Elements used in a Command ('Command Injection')

Data enters the application from an untrusted source.

### 

### #18 CWE-306 Missing Authentication for Critical Function

The product does not perform any authentication for functionality that requires a provable user identity or consumes a significant amount of resources.

### 

### #19 CWE-119 Improper Restriction of Operations within the Bounds of a Memory Buffer

Improper Restriction of Operations within the Bounds of a Memory Buffer vulnerability exists that could cause a crash of the Control Expert software when an incorrect project file is opened

### 

### #20 CWE-276 Incorrect Default Permissions

Incorrect Default Permissions weakness describes a case where software sets insecure permissions to objects on a system

### 

### #21 CWE-918 Server-Side Request Forgery (SSRF)

SSRF vulnerabilities occur when an attacker has full or partial control of the request sent by the web application

### 

### #22 CWE-362 Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')

A race condition vulnerability typically occurs when your application has access to the same shared data and attempts to change variables within it simultaneously.

### 

### #23 CWE-400 Uncontrolled Resource Consumption

The software does not properly control the allocation and maintenance of a limited resource, thereby enabling an actor to influence the amount of resources consumed, eventually leading to the exhaustion of available resources.

### 

### #23 CWE-611 Improper Restriction of XML External Entity Reference

The software processes an XML document that can contain XML entities with URIs that resolve to documents outside of the intended sphere of control, causing the product to embed incorrect documents into its output.

### 

### #25 CWE-94 Improper Control of Generation of Code ('Code Injection')

The software constructs all or part of a code segment using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the syntax or behavior of the intended code segment.