OWASP

•

OWASP basically stands for the **Open Web Application Security Project**, it is a non-profit global online community consisting of tens of thousands of members and hundreds of chapters that produces articles, documentation, tools, and technologies in the field of web application security.

Every three to four years, OWASP revises and publishes its list of the top 10 web application vulnerabilities. This list not only contains the most common top 10 vulnerabilities but also contain the potential impact of each vulnerability and how to avoid them. The OWASP Top 10 Web Application Security Risks was most recently updated in 2017 and it basically provides guidance to developers and security professionals on the most critical vulnerabilities that are most commonly found in web applications, and are also easy to exploit. OWASP's top 10 is considered as an essential guide to web application security best practices.

The top 10 OWASP vulnerabilities in 2020 are:
1. Injection
2. Broken Authentication
3. Sensitive Data Exposure
4. XML External Entities (XXE)
5. Broken Access control
6. Security misconfigurations
7. Cross-Site Scripting (XSS)
8. Insecure Deserialization
9. Using Components with known vulnerabilities
10. Insufficient logging and monitoring.

1. Injection

Injection vulnerabilities occur when an attacker uses a query or command to insert untrusted data into the interpreter via SQL, OS, NoSQL, or LDAP injection. The data that is injected through this attack vector makes the application do something it is not designed for. Not all applications are vulnerable to this attack, only the applications that accept parameters as input are vulnerable to injection attacks.

Injection attacks can be prevented by
- Using safer API which avoids the use of the interpreter
- Using parameterized queries when coding
- Segregating commands from data to avoid exposure to attacks

2. Broken Authentication

Broken Authentication is a vulnerability that allows an attacker to use manual or automatic methods to try to gain control over any account they want in a system. In worse conditions, they could also gain complete control over the system. This vulnerability is also more dangerous because websites with broken authentication vulnerabilities are very common on the web. Broken authentication normally occurs when applications incorrectly execute functions related to session management allowing intruders to compromise passwords, security keys, or session tokens.

Broken authentication attacks can be prevented by

- Implementing multi-factor authentication
- Protecting user credentials
- Sending passwords over encrypted connections

3. Sensitive Data Exposure

This vulnerability is one of the most widespread vulnerabilities on the OWASP list and it occurs when applications and APIs don't properly protect sensitive data such as financial data, social security numbers, usernames, and passwords, or health information, and this enables attackers to gain access to such information and commit fraud or steal identities.

Sensitive data exposure attacks can be prevented by

- Using the secure URL's
- Using strong and unique passwords
- Encrypting all sensitive information that does need to be stored

4. XML External Entities (XXE)

This vulnerability occurs for web applications that parse XML input. It happens when poorly configured XML processors evaluate external entity references within the XML documents and send sensitive data to an unauthorized external entity, i.e., a storage unit such as a hard drive. By default, most XML parsers are vulnerable to XXE attacks.

XXE attacks can be prevented by

- Using less complex data formats such as JSON
- Keeping XML processors and libraries upgraded
- Using SAST tools

5. Broken Access Controls

This vulnerability occurs when there is broken access to resources, it means there are some improperly configured missing restrictions on authenticated users which allows them to access unauthorized functionality or data like access to others accounts, confidential documents, etc. For this attack, attackers take the help of session management and try to access data from the unexpired session tokens, which gives them access to many valid IDs and passwords.

Broken access control attacks can be prevented by

- Deleting accounts that are no longer needed or are not active
- Shutting down unnecessary services to reduce the burden on servers

- Using penetration testing

6. Security Misconfiguration

It is estimated that up to 95% of cloud breaches are the result of human errors and this fact leads us to the next vulnerability called security misconfiguration. This vulnerability refers to the improper implementation of security intended to keep application data safe. As we know that developer's work is basically to work on the functionality of websites and not on security and this flaw allows hackers to keep track of the configuration of the security and find new possible ways to enter websites. The most common reason for this vulnerability is not patching or upgrading systems, frameworks, and components.

Security misconfiguration attacks can be prevented by

- Using Dynamic application security testing (DAST)
- Disabling the use of default passwords
- Keeping an eye on cloud resources, applications, and servers

7. Cross-Site Scripting (XSS)

This is also a widespread vulnerability that almost affects 53% of all web applications. XSS vulnerability allows a hacker to inject malicious client-side scripts into a website and then use the web application as an attack vector to hijack user sessions, or redirecting the victim to malicious websites.

Cross-site scripting attacks can be prevented by

- Using appropriate response headers
- Filtering the input and encoding the output
- Using the content security policy
- Applying a zero-trust approach to user input

8. Insecure Deserialization

Insecure Deserialization vulnerability allows an attacker to remotely execute code in the application, tamper or delete serialized (written to disk) objects, conduct injection attacks, replay attacks, and elevate privileges. This attack is also known as untrusted Deserialization. It is a serious application security issue that affects most of the modern systems.

Insecure Deserialization attacks can be prevented by

- Implementing digital signatures
- Using penetration testing
- Isolating the code that deserializes and running it in low privilege environments to prevent unauthorized actions

9. Using Components with known vulnerabilities

Nowadays there are many open-source and freely available software components (libraries, frameworks) that are available to developers and if there occurs any component which has got a known vulnerability in it then it becomes a weak link that can impact the security of the entire application. It also occurs because developers frequently don’t know which open source and third-party components are present in their applications and this makes it difficult for developers to update components when new vulnerabilities are discovered in their current versions.

This attack can be prevented by

- Removing all unnecessary dependencies
- Using virtual patching
- Using components only from official and verified sources

10. Insufficient Logging and Monitoring

It is estimated that the time from attack to detection can take up to 200 days, and often longer. In the meantime, attackers can tamper with servers, corrupt databases, and steal confidential information. Insufficient logging and ineffective integration of the security systems allow attackers to pivot to other systems and maintain persistent threats.

Insufficient logging and monitoring attacks can be prevented by

- Implementing logging and audit software
- Establishing an effective monitoring system
- Thinking like an attacker and use a pen testing approach