Made by: Georgi Zhizgov

Security Report

Table of contents

[1. What is OWASP? 2](#_Toc122616254)

[2. What is OWASP Top 10 security risks ranking for 2021? 2](#_Toc122616255)

[A01 Broken Access Control 2](#_Toc122616256)

[A02 Cryptographic Failures 2](#_Toc122616257)

[A03 Injection 2](#_Toc122616258)

[A04 Insecure Design 2](#_Toc122616259)

[A05 Security Misconfiguration 3](#_Toc122616260)

[A06 Vulnerable and Outdated Components 3](#_Toc122616261)

[A07 Identification and Authentication Failures 3](#_Toc122616262)

[A08 Software and Data Integrity Failures 3](#_Toc122616263)

[A09 Security Logging and Monitoring Failures 4](#_Toc122616264)

[A10 Server-Side Request Forgery (SSRF) 4](#_Toc122616265)

[3. Application analyzes 5](#_Toc122616266)

# What is OWASP?

The Open Web Application Security Project (OWASP) is a nonprofit foundation that works to improve the security of software. Through community-led open-source software projects, hundreds of local chapters worldwide, tens of thousands of members, and leading educational and training conferences, the OWASP Foundation is the source for developers and technologists to secure the web.

# What is OWASP Top 10 security risks ranking for 2021?

A01 Broken Access Control –  
Access control enforces policy such that users cannot act outside of their intended permissions. Failures typically lead to unauthorized information disclosure, modification, or destruction of all data or performing a business function outside the user's limits.

A02 Cryptographic Failures–  
The first thing is to determine the protection needs of data in transit and at rest. For example, passwords, credit card numbers, health records, personal information, and business secrets require extra protection, mainly if that data falls under privacy laws, e.g., EU's General Data Protection Regulation (GDPR), or regulations, e.g., financial data protection such as PCI Data Security Standard (PCI DSS).

A03 Injection–  
An application is vulnerable to injections when the user-supplied data is not validated, filtered, or sanitized by the application. Dynamic queries or non-parameterized calls without context-aware escaping are used directly in the interpreter. Hostile data is used within object-relational mapping (ORM) search parameters to extract additional, sensitive records. Hostile data is directly used or concatenated. The SQL or command contains the structure and malicious data in dynamic queries, commands, or stored procedures.

A04 Insecure Design–  
Insecure design is a broad category representing different weaknesses, expressed as missing or ineffective control design. An insecure design cannot be fixed by a perfect implementation as by definition, needed security controls were never created to defend against specific attacks. One of the factors that contribute to insecure design is the lack of business risk profiling inherent in the software or system being developed, and thus the failure to determine what level of security design is required.

A05 Security Misconfiguration–  
Without a concerted, repeatable application security configuration process, systems are at a higher risk. Such as:

* Missing appropriate security hardening across any part of the application stack or improperly configured permissions on cloud services.
* Unnecessary features are enabled or installed (e.g., unnecessary ports, services, pages, accounts, or privileges).
* Default accounts and their passwords are still enabled and unchanged.
* Error handling reveals stack traces or other overly informative error messages to users.
* For upgraded systems, the latest security features are disabled or not configured securely.
* The security settings in the application servers, application frameworks (e.g., Struts, Spring, ASP.NET), libraries, databases, etc., are not set to secure values.
* The server does not send security headers or directives, or they are not set to secure values.
* The software is out of date or vulnerable

A06 Vulnerable and Outdated Components–  
You are vulnerable if you do not know the versions of all components you use (both client-side and server-side). This includes components you directly use as well as nested dependencies. If the software is vulnerable, unsupported, or out of date. This includes the OS, web/application server, database management system (DBMS), applications, APIs and all components, runtime environments, and libraries. If you do not scan for vulnerabilities regularly and subscribe to security bulletins related to the components, you use. If you do not fix or upgrade the underlying platform, frameworks, and dependencies in a risk-based, timely fashion. If software developers do not test the compatibility of updated, upgraded, or patched libraries.

A07 Identification and Authentication Failures–  
Confirmation of the user's identity, authentication, and session management is critical to protect against authentication-related attacks. There may be authentication weaknesses such as the application permits brute force or other automated attacks. Permits default, weak, or well-known passwords, has missing or ineffective multi-factor authentication, exposes session identifier in the URL etc.

A08 Software and Data Integrity Failures–  
Software and data integrity failures relate to code and infrastructure that does not protect against integrity violations. An example of this is where an application relies upon plugins, libraries, or modules from untrusted sources, repositories, and content delivery networks (CDNs). An insecure CI/CD pipeline can introduce the potential for unauthorized access, malicious code, or system compromise.

A09 Security Logging and Monitoring Failures–  
This category is to help detect, escalate, and respond to active breaches. Without logging and monitoring, breaches cannot be detected. Insufficient logging, detection, monitoring, and active response occurs any time. So, you are vulnerable to information leakage by making logging and alerting events visible to a user or an attacker

A10 Server-Side Request Forgery (SSRF)–  
SSRF flaws occur whenever a web application is fetching a remote resource without validating the user-supplied URL. It allows an attacker to coerce the application to send a crafted request to an unexpected destination, even when protected by a firewall, VPN, or another type of network access control list (ACL).

# Application analyzes

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Likelihood** | **Impact** | **Risk** | **Actions possible** | **Planned** |
| *A01 Broken Access Control* | Very unlikely | Severe | Low | The authorization is set depending on users role in both front-end and back-end | YES |
| *A02 Cryptographic Failures* | Very unlikely | Severe | Low | The application crypths the confidentional information by generating JWT token | YES |
| *A03 Injection* | Moderate | Severe | Low | If the query is customized it may be a problem, but using the JPA it mainly | YES |
| *A04 Insecure Design* | High | Low | High | Currently the application in not foucosing on the designing | No |
| *A05 Security Misconfiguration* | unlikely | Low | Low | The application dosen't use any unnessesery ports and the security is handled by Spring framework | YES |
| *A06 Vulnerable and Outdated components* | Very unlikely | Moderate | Modareate | Currently the application frameworks up to date | No |
| *A07 Identification and Authentication Failures* | Very unlikely | Severe | Low | Applicaton uses Spring Authentication and Authorization | YES |
| *A08 Software and Data Integrity Failures* | Moderate | Moderate | Modareate | There is a CI/CD pipeline | YES |
| *A09 Security Logging and Monitoring Failures* | Very unlikely | Moderate | Low |  |  |
| *A10 Server-Side Request Forgery (SSRF)* | Very unlikely | Moderate | Low |  | No |