OWASP

@Maharshi Jinandra

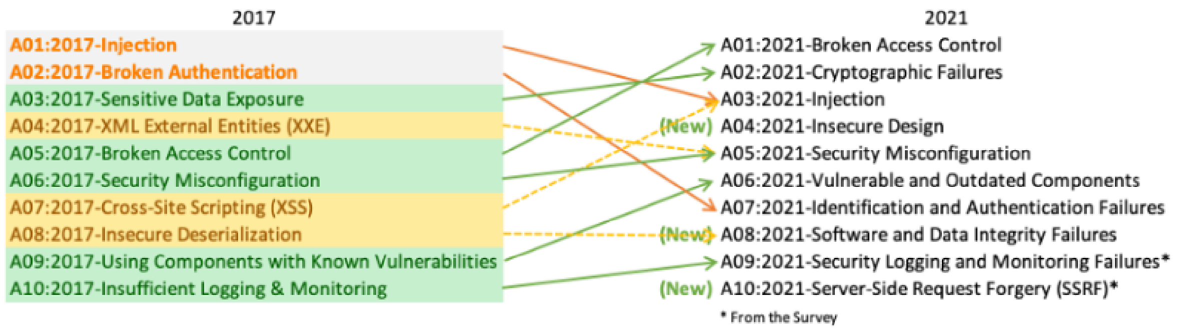
The **O**pen **W**eb **A**pplication **S**ecurity **P**roject is a non-profit organization founded in 2001, the goal of helping website owners and security experts protect web applications from cyber attacks. OWASP has 32,000 volunteers around the world who perform security assessments and research.

Among OWASP's key publications are the OWASP Top 10, the OWASP Software Assurance Maturity Model (SAMM), the OWASP Development Guide, the OWASP Testing Guide, and the OWASP Code Review Guide.

OWASP Top 10

A research project that offers rankings of and remediation advice for the top 10 most serious web application security dangers. The report is founded on an agreement between security experts from around the globe. The risks are graded according to the severity of the vulnerabilities, the frequency of isolated security defects, and the degree of the possible impacts.

- **Green arrows** are vulnerabilities that were promoted in importance
- **Orange arrows** are vulnerabilities that were demoted in importance
- **Yellow broken** line arrows are vulnerabilities removed and merged into other categories



2017
A01:2017-Injection
A02:2017-Broken Authentication
A03:2017-Sensitive Data Exposure
A04:2017-XML External Entities (XXE)
A05:2017-Broken Access Control
A06:2017-Security Misconfiguration
A07:2017-Cross-Site Scripting (XSS)
A08:2017-Insecure Deserialization
A09:2017-Using Components with Known Vulnerabilities
A10:2017-Insufficient Logging & Monitoring
2021
A01:2021-Broken Access Control
A02:2021-Cryptographic Failures
A03:2021-Injection
(New) A04:2021-Insecure Design
A05:2021-Security Misconfiguration
A06:2021-Vulnerable and Outdated Components
A07:2021-Identification and Authentication Failures
(New) A08:2021-Software and Data Integrity Failures
A09:2021-Security Logging and Monitoring Failures*
(New) A10:2021-Server-Side Request Forgery (SSRF)*
* From the Survey

A01:2021 – Broken Access Control

Means that attacker can gain access to user accounts and act as user or administrators, and that regular users can gain unintended privileged functions. Strong access mechanisms ensure that each role has clear and isolated privileges.

Mitigating Broken Access Control

- Deny access by default, expect for public resources
- Build strong access control mechanisms and reuse them across the applications
- Disable server directory listing and do not store sensitive data in root
- Rate limit API and controller access
- Validate JWT tokens after logout

A02:2021 — Cryptographic Failures

Previously known as Sensitive Data Exposure, covers the protection of data in transit and at rest. This includes passwords, credit card numbers, health records, personal information and other sensitive information.

It is especially important for organizations covered by standards like PIC Data Security Standards or data privacy regulations like EU General Protection Regulation

Mitigating Broken Access Control

- Identify sensitive data and apply appropriate security controls
- Don’t store sensitive data unless absolutely needed; discard sensitive data, use tokenization or truncation
- Encrypt all sensitive data at rest using strong encryption algorithms, protocols and keys
- Encrypt data in transit using secure protocols like TLS and HTTP HSTS (Strict-Transport-Security)
- Disable caching of sensitive data
- Store passwords using strong, salted hashing functions like Argon2, scrypt and bcrypt

A03:2021 — Injection

An injection vulnerability in a web application allows attackers to send hostile data to interpreter, causing the data to be compiled and executed on the server. A common form of injection is SQL injection.

Preventing Injection Attacks

- Use a safe API which avoids the use of the interpreter entirely
- Use positive or whitelist server side input validations
- Escape special characters
- Use LIMIT and other SQL control within queries to prevent mass disclosure of records in case of SQL injection

A04:2021 — Insecure Design

This category of weakness that originate from missing or ineffective security controls. Some applications are built without security in mind. Others do have a secure design, but have implementation flaws that can lead to exploitable vulnerabilities.

By definition, an insecure design cannot be fixed by proper implementation or configuration. This is because it is lacking basic security controls that can effectively protect against important threats.

Preventing insecure design

- Establish a secure SDLC
- Leverage application security practices from early stages of software development
- Create a library of secure design patterns, and use it to build new applications
- Leverage threat modeling to design critical features like authentication and access control
- Integrate security concerns and controls into all user stories

A05:2021 — Security Misconfiguration

Lack of security hardening across the application stack. This can include improper configuration of cloud service permissions, enabling or installing features that are not required, and default admin accounts or passwords. This now also includes XML External Entities (XXE), previously a separate OWASP category.

Preventing security misconfiguration

- Establish a hardening process for applications, which is fast and east to deploy
- Configure development, OA, and production identically (with different credentials)
- All systems should have a minimal setup without unnecessary features and components
- Configurations should be regularly updated, applying patches and security advisories
- Establish an automated process to verify secure configurations in all environments

A06:2021 — Vulnerable and Outdated Components

Previously known as “Using Components with known vulnerabilities”, includes vulnerabilities resulting from unsupported or outdated software.

Anyone who builds or uses an application without knowing its internal components, their versions, and whether they are updated, is exposed to this category of vulnerabilities.

Preventing

- Remove unused dependencies, features, components, and files from application
- Maintain an inventory of components and their versions, both on the client side and sever side, using software composition analysis tools
- Continuously scan libraries and their dependencies for vulnerable components
- Only use components from official sources, and prefer signed packages
- Urgently remediate vulnerabilities, remove affected components, or apply a virtual patch

A07:2021 — Identification and Authentication Failures

Previously known as “Broken Authentication”, this category now also includes security problems related to user identities. Confirming and verifying user identities, and establishing secure session management, is critical to protect against many types of exploits and attacks.

Mitigating

- Implement multi-factor authentication
- Do not deploy systems with default cerdentials
- Check for a list of the top 10,000 worst passwords
- Harden all authentication-related processes like registration and credential recovery
- Limit or delay failed login attempts

A08:2021 — Software and Data Integrity Failures

This involves code and infrastructure that are vulnerable to integrity violations. This includes software updates, modification of sensitive data, and CI/CD pipelines changes performed without validation. An insecure CI/CD pipeline can lead to unauthorized access, introduced of malware, and other severe vulnerabilities.

This is a global concern around applications with automatic updates. In several cases, attackers broke into the supply chain and created their own malicious updates. Thousands of organizations were compromised by downloading updates and applying these malicious updates to previously trusted applications, without integrity validation.

Preventing

- Use digital signatures or similar mechanisms to verify software or data is from the expected source and has not been altered
- Ensure libraries and dependencies, such as npm or maven, are pulling from trusted repositories
- Establish a review process for code and configuration changes
- Ensure that your CI/CD pipelines has proper configuration and access controls

A09:2021 — Security Logging and Monitoring Failures

Previously named “Insufficient Logging and Monitoring”, involves weakness in an application’s ability to detect security risks and respond to them. Breaches cannot be detected without logging and monitoring. Failures in this category affect visibility, and forensics

Preventing

- Ensure Login access control, and server-side input validation is logged
- Ensure logs contain enough context to identify suspicious behavior and enable in-depth forensic analysis
- Ensure logs are in a format compatible with log management solutions
- Take measures to prevent attackers from tampering with log data

A10:2021 — Server Side Request Forgery

SSRF vulnerability occurs when a web application pulls data from a remote resource based on a user-specified URL, without validating the URL. Even severs protected by firewall, VPN, or network access control list can be vulnerable to this attack, if they accept unvalidated URLs as user inputs.

Preventing

- Avoid accepting URLs in client inputs, and if absolutely necessary, sanitize inputs
- Isolate any remote resource access functionality in a separate network to reduce impact
- Use “deny by default” firewall policies to block unwanted internet traffic
- Use a positive allow list with URL scheme, port and destination
- Disable HTTP redirections
- Never return raw responses to clients