OPEN WEB APPLICATION SECURITY PROJECT

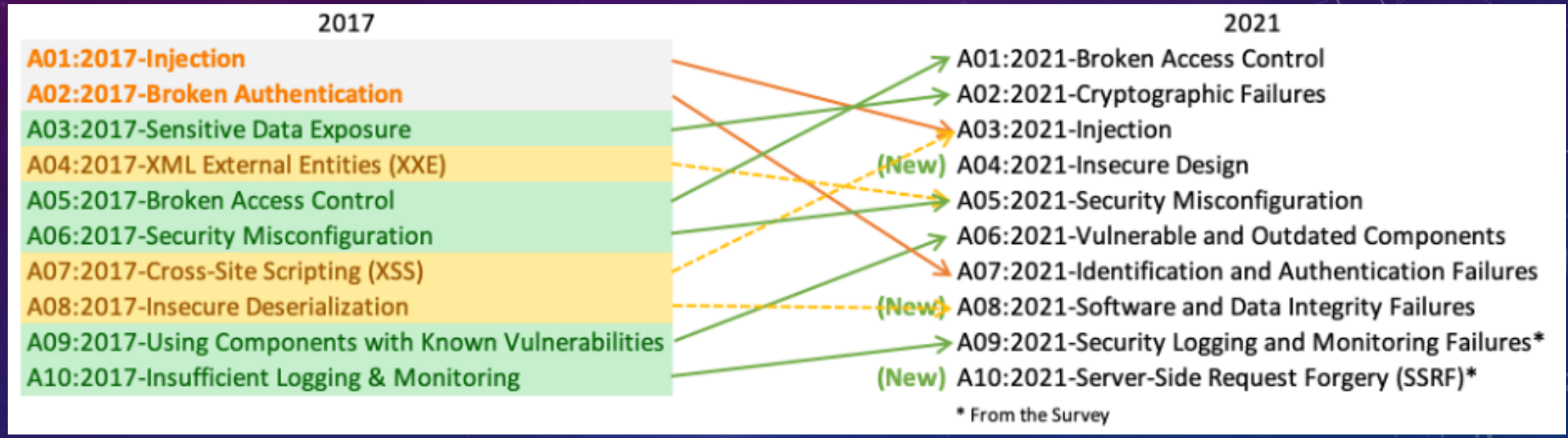
TEN OWASP

CARINE GUIOLET

État actuel du projet en septembre 2024 :

La publication du **OWASP Top 10:2025** au cours du premier semestre 2025.

Collecte de données (jusqu’à décembre 2024)


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A06:2021-Vulnerable and Outdated Components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

1. A01 :2021 - BROKEN ACCESS CONTROL

Description :

Un contrôle d’accès mal implémenté permet aux utilisateurs malveillants d’accéder à des ressources ou d’exécuter des actions non autorisées.

Exemples :

- Accès aux comptes d’autres utilisateurs en modifiant un identifiant dans l’URL.
- Suppression de données sans permissions adéquates.
- Élévation de privilèges non intentionnelle.

Mesures de protection :

- Mettre en œuvre un contrôle d’accès basé sur des rôles (RBAC).
- Bloquer l’accès aux ressources non autorisées côté serveur.
- Éviter de stocker des permissions sensibles côté client.

2. A02:2021 - CRYPTOGRAPHIC FAILURES

Description :
Les applications ne protègent pas suffisamment les données en transit ou au repos, conduisant à des fuites d'informations sensibles.

Exemples :

- Transmission de mots de passe ou de données sensibles en clair.
- Mauvaise gestion des clés de chiffrement.
- Utilisation d'algorithmes faibles comme MD5 ou SHA-1.

Mesures de protection :

- Utiliser TLS 1.2/1.3 pour sécuriser les communications.
- Stocker les mots de passe hachés avec bcrypt, Argon2 ou PBKDF2.
- Appliquer le chiffrement AES-256 pour les données sensibles.

3. A03:2021 - INJECTION

Description :
Une application est vulnérable aux injections (SQL, NoSQL, XSS) lorsque des données non vérifiées sont utilisées directement dans des requêtes ou des scripts.

Exemples :

- Injection SQL (' OR '1'='1) pour accéder à des bases de données.
- Injection de commandes système (; rm -rf /) via des entrées utilisateur.
- Exécution de scripts malveillants via XSS.

Mesures de protection :

- Utiliser des requêtes préparées pour SQL.
- Échapper les entrées utilisateur avant affichage.
- Appliquer une validation stricte des entrées.

4. A04:2021 - INSECURE DESIGN

Description :
Les erreurs de conception de l’application introduisent des failles exploitables par les attaquants.

Exemples :

- Absence d’un mécanisme de validation des entrées utilisateur.
- Manque de vérification des permissions dans les API.
- Autorisation implicite de certaines actions sensibles.

Mesures de protection :

- Appliquer la méthode "Security by Design".
- Effectuer des revues de code et des audits de sécurité.
- Définir des exigences de sécurité dès la conception.

5. A05:2021 - SECURITY MISCONFIGURATION

Description :
Des configurations faibles ou incorrectes rendent l'application vulnérable.

Exemples :

- Exposition des messages d'erreur détaillés en production.
- Utilisation de mots de passe par défaut.
- Services inutiles activés sur un serveur.

Mesures de protection :

- Désactiver les fonctionnalités non utilisées.
- Réduire les messages d'erreur affichés aux utilisateurs.
- Automatiser la gestion des configurations sécurisées.

6. A06:2021 - VULNERABLE AND OUTDATED COMPONENTS

Description :
L’utilisation de bibliothèques ou de frameworks obsolètes peut introduire des failles de sécurité.

Exemples :

- Utilisation d’une version ancienne de Log4j vulnérable à l’exécution de code à distance.
- Dépendances non mises à jour avec des vulnérabilités connues.

Mesures de protection :

- Mettre à jour régulièrement les bibliothèques et frameworks.
- Utiliser des outils de scan des dépendances comme OWASP Dependency-Check.
- Supprimer les composants inutilisés.

7. A07:2021 - IDENTIFICATION AND AUTHENTICATION FAILURES

Description :
Les failles dans la gestion des sessions et de l'authentification permettent aux attaquants d'usurper des identités.

Exemples :

- Absence de vérification de la force des mots de passe.
- Sessions utilisateur sans expiration.
- Manque de protection contre les attaques par force brute.

Mesures de protection :

- Appliquer l'authentification multifactorielle (MFA).
- Implémenter des limites de tentatives de connexion.
- Utiliser des sessions sécurisées avec expiration.

8. A08:2021 - SOFTWARE AND DATA INTEGRITY FAILURES

Description :
L’application ne garantit pas l’intégrité des mises à jour logicielles ou des données échangées.

Exemples :

- Mise à jour logicielle téléchargée depuis une source non vérifiée.
- Manipulation des données via des attaques de type "man-in-the-middle".

Mesures de protection :

- Signer numériquement les mises à jour et les packages.
- Utiliser HTTPS et vérifier les certificats.
- Appliquer le contrôle d’intégrité des fichiers.

9. A09:2021 - SECURITY LOGGING AND MONITORING FAILURES

Description :
Le manque de journalisation et de surveillance empêche la détection rapide des attaques.

Exemples :

- Absence de logs pour les tentatives de connexion échouées.
- Journaux insuffisants pour identifier les attaques en cours.
- Manque d’alertes sur les comportements suspects.

Mesures de protection :

- Activer la journalisation des événements de sécurité.
- Stocker les logs de manière sécurisée et centralisée.
- Configurer des alertes pour les actions suspectes.

10. A10:2021 - SERVER-SIDE REQUEST FORGERY

Description :
Une attaque SSRF se produit lorsque l'application permet à un attaquant d'envoyer des requêtes à des ressources internes via une URL manipulée.

Exemples :

- Accès aux métadonnées d'un serveur cloud (http://169.254.169.254/latest/meta-data).
- Exfiltration de données internes via des requêtes HTTP.

Mesures de protection :

- Filtrer les entrées utilisateur pour éviter l'injection d'URL malveillantes.
- Restreindre l'accès aux ressources internes via des règles de pare-feu.
- Désactiver les protocoles non nécessaires (file://, ftp://).