

# Cybersécurité

## C1 - La protection des applications web

# Sommaire

- 1) Mise en contexte
- 2) Présentation de OWASP
- 3) Projet TOP 10
- 4) Classement des vulnérabilités
- 5) Broken Access Control
- 6) Injection SQL
- 7) Exemple d'une injection SQL





## Mise en contexte

Nous avons entamé la nouvelle ère du monde informatique depuis quelques années avec l'arrivée du Cloud et des crypto-monnaies ou encore avec la multiplication des applications smartphone et des réseaux sociaux.

Les rapports et les médias montrent que le nombre d'attaques ou de bugs informatiques ne cessent d'accroître.

Comment se protéger contre ces attaques ?



# Présentation de OWASP

**OWASP** (Open Web Application Security project)

→ Communauté travaillant sur la sécurité des applications web

Elle a pour but de **publier des recommandations** de sécurisation des sites web et **propose des outils** permettant de tester la sécurité des applications web.

Elle s'adresse aux développeurs, architectes, chefs de projets, managers...



# Projet TOP 10

## TOP 10 de l'OWASP

→ projet visant à lister les 10 risques de sécurité et vulnérabilités les plus critiques affectant les applications Web.

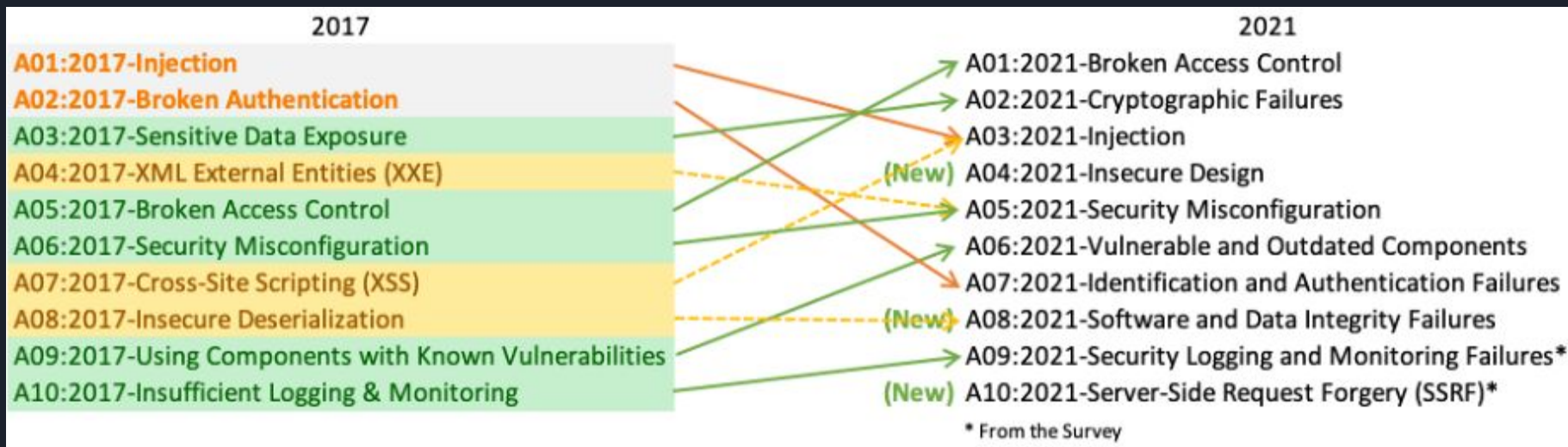
Ce classement est mis à jour et:

- a pour but d'informer sur l'existence de ces vulnérabilités
- et de fournir des guides simplifiés sur les bonnes pratiques pour s'en prémunir.

# Evolution des classement (2013 à 2017)

OWASP Top 10 - 2013	→	OWASP Top 10 - 2017
A1 – Injection	→	A1:2017-Injection
A2 – Broken Authentication and Session Management	→	A2:2017-Broken Authentication
A3 – Cross-Site Scripting (XSS)	↘	A3:2017-Sensitive Data Exposure
A4 – Insecure Direct Object References [Merged+A7]	U	A4:2017-XML External Entities (XXE) [NEW]
A5 – Security Misconfiguration	↘	A5:2017-Broken Access Control [Merged]
A6 – Sensitive Data Exposure	↗	A6:2017-Security Misconfiguration
A7 – Missing Function Level Access Contr [Merged+A4]	U	A7:2017-Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	⊗	A8:2017-Insecure Deserialization [NEW, Community]
A9 – Using Components with Known Vulnerabilities	→	A9:2017-Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	⊗	A10:2017-Insufficient Logging&Monitoring [NEW,Comm.]

# Evolution des classement (2017 à 2021)



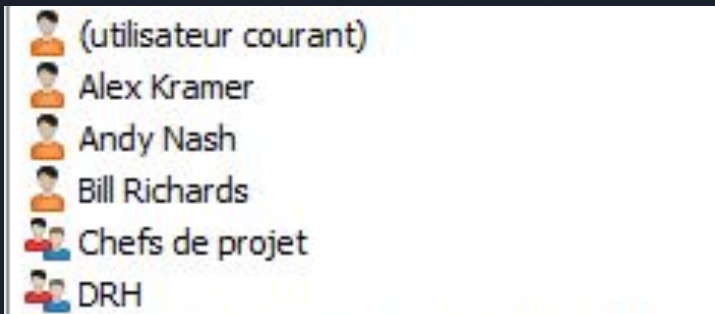
# Analyse du classement 2021





# Broken Access Control

Le contrôle d'accès applique une stratégie pour que les utilisateurs ne puissent pas agir en dehors de leurs autorisations prévues. On place les utilisateurs dans des groupes où la gestion de profil est possible.



## Broken Access Control

Les motivations des attaquants sont généralement la divulgation d'informations, la modification ou la destruction de données ou encore l'exécution d'une fonction quelconque qui n'était pas attribuée à l'utilisateur.



Andy Nash



**Admin**



# Broken Access Control

Les vulnérabilités courantes du contrôle d'accès incluent :

- Contournement des restrictions en modifiant l'URL, l'état interne ou la page HTML de l'application, ou en utilisant un outil d'attaque d'API personnalisé (Application Programming Interface).
- Permettre à la clé primaire d'être remplacée par l'enregistrement d'un autre utilisateur, autorisant l'affichage ou la modification du compte de quelqu'un d'autre.
- Élévation de privilège en manipulant les métadonnées
- Forcer la navigation vers des pages authentifiées ou privilégiées



## Broken Access Control: Exemple de faille

Un attaquant force la navigation vers les URL cibles, logiquement les droits “admin” sont requis pour accéder à la page d'administration.

<https://example.com/app/getappInfo>

[https://example.com/app/admin\\_getappInfo](https://example.com/app/admin_getappInfo)

Si un user non authentifié peut accéder à la page *getappInfo*:  
c'est un défaut.

Si un non-admin peut accéder à la page *admin\_getappInfo*:  
c'est une faille.

# Injection SQL

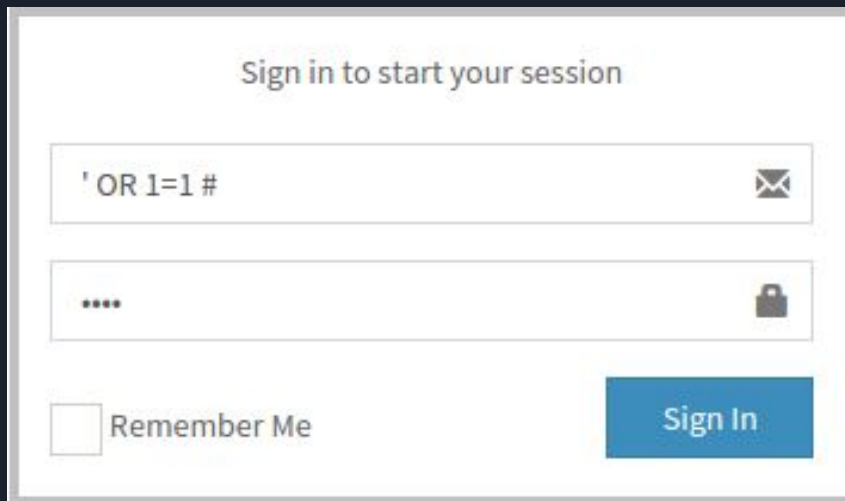
Type de cyberattaque dans lequel l'attaquant insère son propre code dans un site web ou une application qui utilise le langage SQL pour manipuler ses informations dans la base de données.

→ le pirate peut ainsi accéder aux informations potentiellement sensibles.



## Injection SQL: Exemple

L'attaquant ajoute le code suivant à la requête, comme la condition "1=1" est toujours vraie, la requête s'exécute.



A screenshot of a web application's login page. The page has a white background with a light gray border. At the top, the text "Sign in to start your session" is centered. Below this, there are two input fields. The first field contains the text "' OR 1=1 #" and has an envelope icon on the right. The second field contains four dots "...." and has a padlock icon on the right. Below these fields, there is a checkbox labeled "Remember Me" and a blue button labeled "Sign In".



## Quelques solutions

Les faiblesses du contrôle d'accès sont courantes en raison du manque de détection automatisée et du manque de tests fonctionnels efficaces par les développeurs d'applications.

- Refuser par défaut l'accès aux fonctionnalités/applications.
- Le test manuel est le meilleur moyen de détecter le contrôle d'accès manquant ou inefficace (exemple: effectuer des audit régulièrement).
- On peut ajouter aussi des logiciels de scan de vulnérabilité des applications (VST) ou des outils de test de sécurité des applications statiques (SAST).