

12. Risques applicatifs

16 décembre 2025

Développement web il3

Risques applicatifs des app web

HE-Arc (DGR) 2025

Risque

- Faille ou bug permettant d'altérer le fonctionnement
- Un attaquant pourra :
 - Modifier le fonctionnement
 - Accéder ou modifier les données
- Présence possible à tous les niveaux d'un système
 - Application
 - Serveur et Client
 - OS
 - SGBD, ...
- Responsabilité des développeurs :
 - OS, serveurs, langages : patches rapidement disponibles
 - nos applications : **c'est nous qui en sommes responsables**

OWASP¹

- Open Worldwide Application Security Project
- Fondation pour améliorer la sécurité des webapps
- Fondée en 2004, internationale, sans but lucratif
- Référence principale dans le domaine

1. <https://owasp.org/>

- Propose :
 - Top 10 (web, mobile ², API ³, LLM ⁴, OT ⁵) tous les 4 ans : Méthode ⁶, CVSS ⁷, CWE ⁸
 - Grande communauté d'experts
 - Formation, documentation et ressources
 - Outils d'audit, de tests et de formation (ex : Juice Shop ⁹)
 - Cheat Sheets ¹⁰ (yc pour CICD, Ajax, Laravel, Django,...;)

Top 10 ¹¹ OWASP 2025 (fr ¹² - historique ¹³)

1. Contrôle d'accès défaillants
 2. Mauvaise configuration de sécurité
 3. Vulnérabilités des dépendances
 4. Défaillances cryptographiques
 5. Injections
 6. Conception non sécurisée
 7. Authentification de mauvaise qualité
 8. Manque d'intégrité des données et du logiciel
 9. Carences des systèmes d'alerte et de journalisation
 10. Mauvaise gestion des conditions exceptionnelles
- Non exhaustif : ex. : risques liés à Node JS ¹⁴

Injection de code

- Données mal validées : possibilité d'exécuter du code
- Passées par requêtes :
 - formulaires
 - URL
 - ...
- Type de code injectable : TOUS !

-
2. <https://owasp.org/www-project-mobile-top-10/>
 3. <https://owasp.org/www-project-api-security/>
 4. <https://genai.owasp.org/resource/owasp-top-10-for-llm-applications-2025/>
 5. <https://ot.owasp.org/>
 6. <https://owasp.org/Top10/#methodology>
 7. <https://www.first.org/cvss/calculator/3.0>
 8. https://cwe.mitre.org/top25/archive/2022/2022_cwe_top25.html
 9. <https://owasp.org/www-project-juice-shop/>
 10. <https://cheatsheetseries.owasp.org/>
 11. https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project
 12. <https://owasp.org/Top10/fr/>
 13. <https://www.hahwul.com/cullinan/history-of-owasp-top-10/>
 14. https://cheatsheetseries.owasp.org/cheatsheets/NPM_Security_Cheat_Sheet.html

- HTML
- SQL
- Javascript
- ...

Injections SQL

- Modifier les requêtes envoyées au SGBD
- Obtention d'un résultat non prévu par le développeur
- Deviner la structure du code pour l'exploiter
- SQL est puissant : UNION, INTO DUMPFILE, ...

Exemples ¹⁵

```
SELECT titre, num FROM livres WHERE num=2 UNION  
SELECT login, password FROM user INTO DUMPFILE 'www/exploit.txt'
```

Eviter les injections SQL

- N'accepter que des caractères valides
- A défaut, neutraliser les caractères dangereux
- Utiliser les entités HTML
- Vérifications strictes dans le code
- Eviter les noms prévisibles pour une appli critique

Cross Site Scripting (XSS)

- Injection de code (html et script)

15. https://fr.wikipedia.org/wiki/Injection_SQL



- Exécution par le navigateur du client

A High Level View of a typical XSS Attack

Cross Site Scripting (XSS)

- Enjeux : tout ce qui est possible en JS
 - Redirection
 - Lecture de cookies (session, ...)
 - Envoi d'info à un autre serveur
 - Modification du contenu de la page
 - ...
- Souvent utilisé pour transmettre le cookie de session

```

```

3 types de XSS

- Reflected XSS
 - Affichage d'une partie de la requête (recherche, erreur, ...)
- Stored XSS
 - Stockage dans la BDD et affichage (= exécution) par plusieurs clients
- DOM based XSS

- Exécutée lors de la modification du DOM (Exemple ¹⁶)

Cross Site Request Forgery (CSRF - Sea Surf)

- **Principe :**
 - Faire réaliser à quelqu'un une action à son insu, avec ses propres infos d'authentification (credentials)
 - Envoi par mail ou post forum de liens ou images
 - Les URL correspondent à actions (vote, suppression, ...)

Exemple ¹⁷ (SOP, CORS)

Phishing

- Site sosie d'un site officiel :
 1. L'utilisateur saisit ses données...
 2. ... l'attaquant les récupère...
 3. ... et les utilise sur le site officiel
- Difficile à contrer pour le développeur
- L'utilisateur doit être prudent
- Bien lire les URLs et le GUI du navigateur pas toujours suffisant
- Ne pas utiliser de lien dont on n'est pas sûr de la source (Homograph Attack ¹⁸, Homoglyphes ¹⁹, Unicode Spoofing ²⁰)

Risques non liés à l'application

- IoT : souvent mal sécurisé (shodan.io ²¹)
- DoS
- Spoofing (IP, DNS, ARP)
- Buffer Overflows (surtout en C)
- Trojans, backdoors
- Usurpation de mots de passe : dictionnaire, force brute
- **SOCIAL ENGINEERING !!!**

16. https://www.owasp.org/index.php/DOM-Based_XSS

17. <https://www.owasp.org/index.php/CSRF>

18. <https://www.xudongz.com/blog/2017/idn-phishing/>

19. https://github.com/codebox/homoglyph/blob/master/raw_data/chars.txt

20. <https://onlineunicodetools.com/spoof-unicode-text>

21. <https://www.shodan.io/>

Authentification

- **Identification** : annoncer qui on est
 - **Authentification** : prouver qu'on est la personne qu'on prétend être :
 1. Avec quelque chose que l'on **sait** (PIN, mot de passe)
 2. Avec quelque chose que l'on **possède** (téléphone, token, ...)
 3. Avec quelque chose que l'on **est** (biométrie)
 - La sécurité augmente si on combine ces facteurs
 - Important de prendre en compte l'utilisabilité

Top 500 passwords cloud



FIGURE 1 – top 500 passwords cloud

Mots de passe

- 30% of users have a password from the top 10'000 (source [22](#))
 - Our passwords habits revealed [23](#)
 - xkcd's password strength [24](#)
 - 2017 : NIST 800-63-3 [25](#) suivi par la NCSC [26](#)
 - Mots de passe longs plutôt qu'avec des caractères spéciaux
 - Ne forcer le changement qu'en cas de nécessité
 - Autoriser et accompagner l'utilisation de password managers
 - Utiliser la 2FA

22. <https://mojoauth.com/blog/why-are-businesses-still-using-passwords/>

23. <https://visual.ly/our-password-habits-revealed>

24. <https://xkcd.com/936/>

25. <https://pages.nist.gov/800-63-3/>

26. <https://www.ncsc.gov.uk/guidance/password-guidance-simplifying-your-approach>

- Plusieurs tentatives pour s'en affranchir :
 - Microsoft ²⁷, passwordless ²⁸ authentication
 - 2022 : Passkeys : JS API WebAuthN ²⁹ + CTAP/U2F ³⁰

Passkeys ³¹

- Paire de clés asymétriques au lieu d'un mot de passe
- Initiative de l'alliance FIDO ³²
- Fin 2022 : intégrée à Android, iOS, win11 et MacOS
- Résolution de challenges : pas d'info sensible sur le réseau
- 3 acteurs :
 - User Agent : Humain / Navigateur
 - Relying Party : Serveur (service auquel on veut s'authentifier)
 - Authenticator : Clef USB / Smartphone / OS + biométrie
- Communication :
 - User Agent <=> Authenticator : CTAP / U2F
 - User Agent <=> Relying Party : API JS WebAuthn ³³
- Disponible sur Switch Edu-ID : **Testez !**

Passkeys : Acteurs ³⁴

Passkeys : Enregistrement ³⁵

Passkeys : Authentification ³⁶

Collecte d'information

- Toute information est bonne pour l'attaquant
 - Messages d'erreur

27. <https://www.microsoft.com/security/blog/2021/09/15/the-passwordless-future-is-here-for-your-microsoft-account/>

28. <https://hacks.mozilla.org/2014/10/passwordless-authentication-secure-simple-and-fast-to-deploy/>

29. <https://en.wikipedia.org/wiki/WebAuthn>

30. <https://proton.me/blog/fr/universal-2nd-factor-u2f>

31. <https://medium.com/webauthnworks/introduction-to-webauthn-api-5fd1fb46c285>

32. <https://fidoalliance.org/members/>

33. <https://webauthn.guide/>

34. <https://auth0.com/blog/introduction-to-web-authentication/>

35. <https://www.freecodecamp.org/news/intro-to-webauthn/>

36. <https://www.freecodecamp.org/news/intro-to-webauthn/>



FIGURE 2 – Architecture



FIGURE 3 – Reg

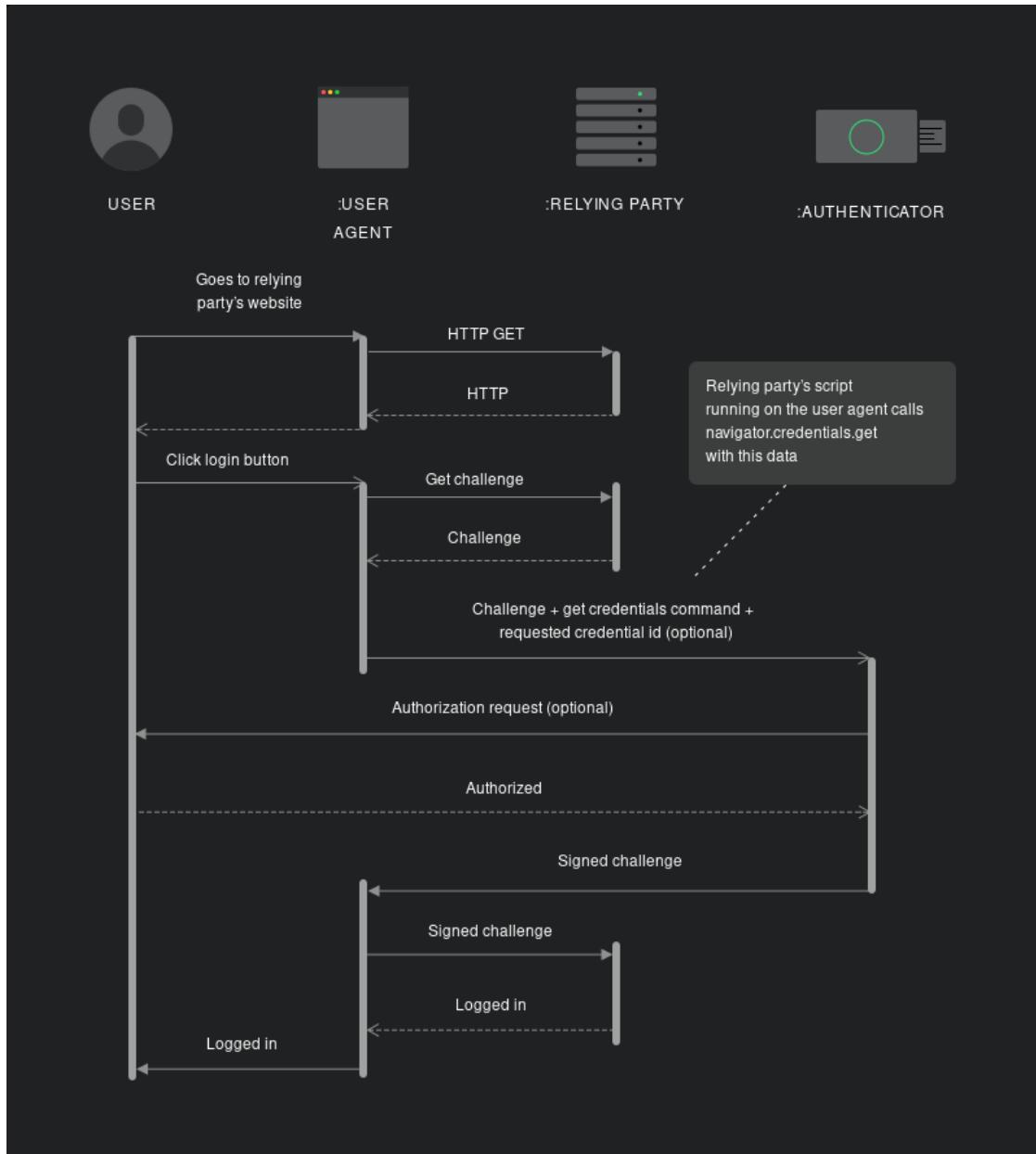


FIGURE 4 – Auth

- Configuration OS serveur
- Configuration serveurs (http, sql, php, ...)
- Identifiants et commentaires dans sources -au cas où-
- SOCIAL ENGINEERING !
- Le développeur doit laisser filter un minimum d'info !
- Utilisée aussi par les “white hats” (ethical hackers) :
 - Cowrie Honeypot ³⁷ (visualisation des attaques en 24h)
 - Autres cartes ³⁸ de menaces et attaques

Bonnes pratiques

- Configuration stricte du serveur
- Valider toutes les entrées (formulaires, requêtes HTTP)
- Filtrage/encodage de toutes les entrées en entités HTML
- Ne jamais afficher directement une saisie de formulaire
 - Ni aucune donnée transmise par HTTP avant de l'avoir filtrée !
- Tester ses formulaires avec des expressions à risques
- Contrôler le maximum de paramètres (même si redondant) :
 - Session, IP, user agent, proxy, ...
- Suites et logiciels de test
- Utiliser un framework
 - ces bonnes pratiques sont déjà implémentées

Laravel, Django et le top 10 OWASP

OWASP 21	Laravel	Django
A01 Accès	Role based AC	Décorateur @login_required, django.contrib.auth make_password(), gestion SECRET_KEY
A02 Crypto	Passwords : Bcrypt, EncryptCookies, Crypt	ORM Django, RawSQL() avec placeholders pour requêtes brutes
A03 Injection	ORM Eloquent, protection injection SQL	Architecture secure by default, check --deploy
A04 Conception	Starter Kits (Breeze, Fortify, Jetstream)	DEBUG = False obligatoire en production, middleware de sécurité activé par défaut
A05 Config	APP_DEBUG = FALSE, permissions fichiers (775/664)	

37. <https://hackertarget.com/cowrie-honeypot-analysis-24hrs/>

38. <https://www.google.com/search?q=ipviking>

OWASP 21	Laravel	Django
A06 Dépendances	Enlightn Security Checker (scan dépendances)	tiers : safety, Bandit, ... pour dépendances
A07 Auth	Sanctum et Passport auth. API, rate limiting	Validateurs passwords, django-axes contre brute-force
A08 Intégrité	mass assignment protection : \$fillable et \$guarded	ModelForm avec Meta.fields (approche liste blanche)
A09 Logs	logs intégré, monitoring en temps réel	Logging configurable, django-axes, ...
A10 SSRF	Validation stricte des URLs	Validation des URLs externes, sécurité en-têtes HTTP
CSRF Protection	VerifyCsrfToken, @csrf pour forms, token automatique pour AJAX via Axios	CsrfViewMiddleware, {% csrf_token %} pour forms
XSS Protection	Échappement automatique via {{ }} dans Blade	Échappement automatique dans templates, éviter mark_safe()

(Généré par perplexity.ai)

Références

- Référence
 - OWASP ³⁹, webinar fr 2016 ⁴⁰
 - WebAuthn : w3c ⁴¹, MDN ⁴²
- Exemples, explications
 - Présentation XSS et CSRF ⁴³ en français
 - Protection CSRF ⁴⁴ en français
- Utilitaires, tutos, exercices
 - Juice Shop ⁴⁵
 - Web Goat ⁴⁶
 - Google-Gruyere ⁴⁷
- Passkeys developer Cheat Sheet ⁴⁸

39. <https://www.owasp.org/>

40. <https://www.youtube.com/watch?v=pHI2zitLph8>

41. <https://www.w3.org/TR/webauthn/>

42. https://developer.mozilla.org/en-US/docs/Web/API/Web.Authentication_API

43. <https://www.journaldunet.com/solutions/dsi/1209139-comment-eviter-les-failles-cross-site-scripting-xss/>

44. <https://www.apprendre-php.com/tutoriels/tutoriel-39-introduction-aux-cross-site-request-forgeries-ou-sea-surf.html>

45. <https://owasp.org/www-project-juice-shop/>

46. <https://www.owasp.org/index.php/Webgoat>

47. <https://google-gruyere.appspot.com/>

48. <https://www.corbado.com/blog/passkeys-cheat-sheet>

Sources