

# 12. Risques applicatifs

7 octobre 2024

## Développement web il3

### Risques applicatifs des app web

HE-Arc (DGR) 2022

### 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**

### OWASP<sup>1</sup>

- Open Web Application Security Project

---

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

- Fondation pour améliorer la sécurité des webapps
- Fondée en 2004, internationale, sans but lucratif
- Référence principale dans le domaine
- Propose :
  - Top 10 (web et mobile<sup>2</sup>) : Méthode<sup>3</sup>, CVSS<sup>4</sup>, CWE<sup>5</sup>
  - Grande communauté d'experts
  - Formation, documentation et ressources
  - Outils d'audit, de tests et de formation

## Top 10<sup>6</sup> OWASP 2021 (fr<sup>7</sup> - historique<sup>8</sup>)

1. Contrôle d'accès défaillants
  2. Défaillances cryptographiques
  3. Injections
  4. Conception non sécurisée
  5. Mauvaise configuration de sécurité
  6. Composants vulnérables et obsolètes
  7. Identification & Authentification de mauvaise qualité
  8. Manque d'intégrité des données et du logiciel
  9. Carences des systèmes de contrôle et de journalisation
  10. Falsification de requêtes côté serveur
- Non exhaustif : ex. : risques liés à Node JS<sup>9</sup>

## Injection de code

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

---

<sup>2</sup><https://owasp.org/www-project-mobile-top-10/>

<sup>3</sup><https://owasp.org/Top10/#methodology>

<sup>4</sup><https://www.first.org/cvss/calculator/3.0>

<sup>5</sup>[https://cwe.mitre.org/top25/archive/2022/2022\\_cwe\\_top25.html](https://cwe.mitre.org/top25/archive/2022/2022_cwe_top25.html)

<sup>6</sup>[https://www.owasp.org/index.php/Category:OWASP\\_Top\\_Ten\\_Project](https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project)

<sup>7</sup><https://owasp.org/Top10/fr/>

<sup>8</sup><https://www.hahwul.com/cullinan/history-of-owasp-top-10/>

<sup>9</sup>[https://cheatsheetseries.owasp.org/cheatsheets/NPM\\_Security\\_Cheat\\_Sheet.html](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<sup>10</sup>

```
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)

---

<sup>10</sup>[https://fr.wikipedia.org/wiki/Injection\\_SQL](https://fr.wikipedia.org/wiki/Injection_SQL)



A High Level View of a typical XSS Attack

- Exécution par le navigateur du client

## 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<sup>11</sup>)

## 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<sup>12</sup> (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<sup>13</sup>, Homoglyphes<sup>14</sup>, Unicode Spoofing<sup>15</sup>)

## Risques non liés à l'application

- IoT : souvent mal sécurisé (shodan.io<sup>16</sup>)
- DoS
- Spoofing (IP, DNS, ARP)

<sup>11</sup>[https://www.owasp.org/index.php/DOM\\_Based\\_XSS](https://www.owasp.org/index.php/DOM_Based_XSS)

<sup>12</sup><https://www.owasp.org/index.php/CSRF>

<sup>13</sup><https://www.xudongz.com/blog/2017/idn-phishing/>

<sup>14</sup>[https://github.com/codebox/homoglyph/blob/master/raw\\_data/char\\_codes.txt](https://github.com/codebox/homoglyph/blob/master/raw_data/char_codes.txt)

<sup>15</sup><https://onlineunicodetools.com/spoof-unicode-text>

<sup>16</sup><https://www.shodan.io/>



- 2017 : NIST 800-63-3<sup>20</sup> suivi par la NCSC<sup>21</sup>
  - 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
- Plusieurs tentatives pour s’en affranchir :
  - Microsoft<sup>22</sup>, passwordless<sup>23</sup> authentication
  - 2022 : Passkeys : JS API WebAuthN<sup>24</sup> + CTAP/U2F<sup>25</sup>

## Passkeys<sup>26</sup>

- Paire de clés asymétriques au lieu d’un mot de passe
- Initiative de l’alliance FIDO<sup>27</sup>
- 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<sup>28</sup>



Figure 2: Architecture



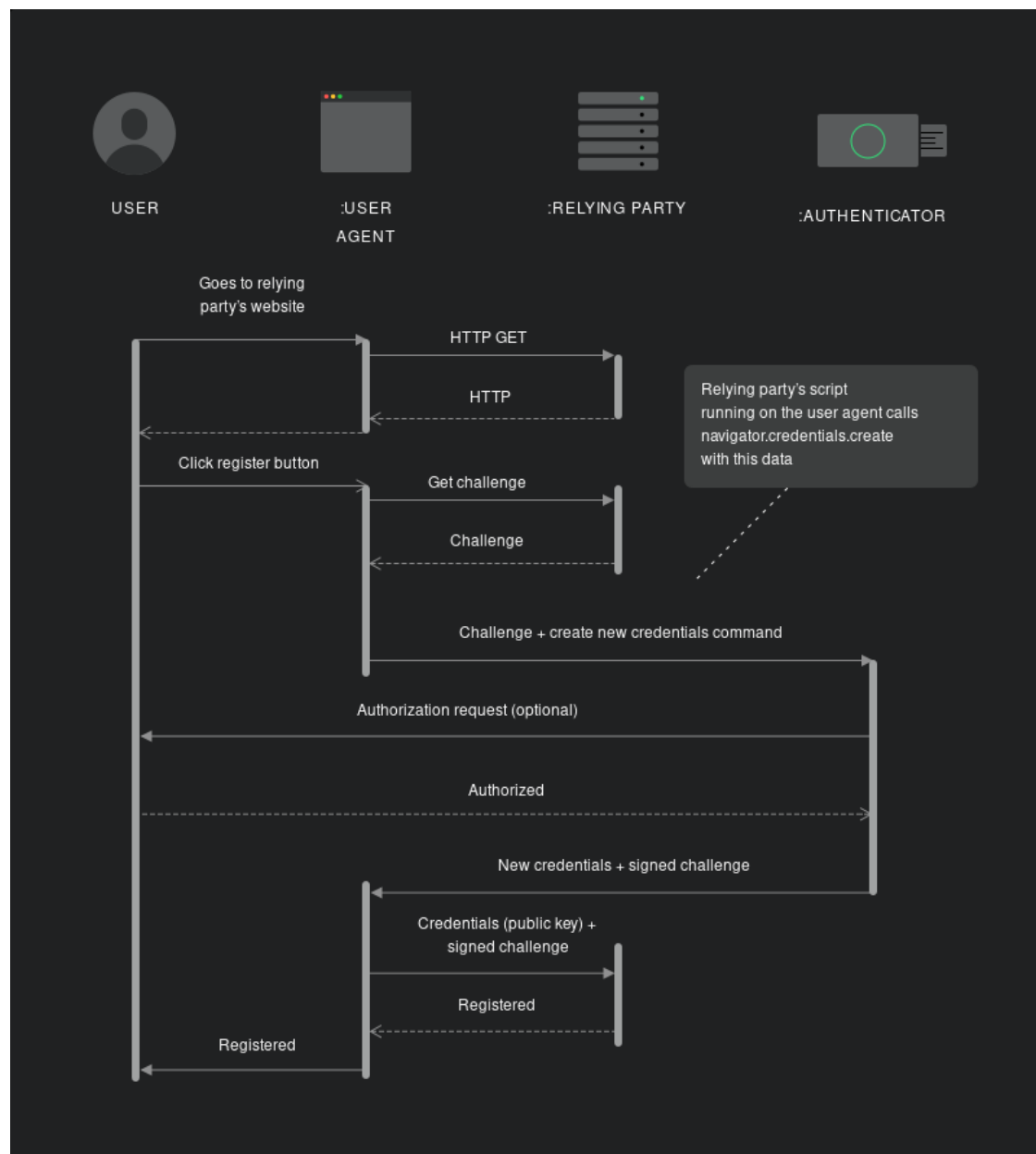


Figure 3: Reg

## Passkeys : Acteurs<sup>29</sup>

## Passkeys : Enregistrement<sup>30</sup>

## Passkeys : Authentification<sup>31</sup>

### Collecte d'information

- Toute information est bonne pour l'attaquant
  - Messages d'erreur
  - 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) : Honeypots<sup>32</sup>

### 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) :

<sup>20</sup><https://nakedsecurity.sophos.com/2016/08/18/nists-new-password-rules-what-you-need-to-know/>

<sup>21</sup><https://www.ncsc.gov.uk/guidance/password-guidance-simplifying-your-approach>

<sup>22</sup><https://www.microsoft.com/security/blog/2021/09/15/the-passwordless-future-is-here-for-your-microsoft-account/>

<sup>23</sup><https://hacks.mozilla.org/2014/10/passwordless-authentication-secure-simple-and-fast-to-deploy/>

<sup>24</sup><https://en.wikipedia.org/wiki/WebAuthn>

<sup>25</sup><https://u2f-key.tech/fr/>

<sup>26</sup><https://medium.com/webauthnworks/introduction-to-webauthn-api-5fd1fb46c285>

<sup>27</sup><https://fidoalliance.org/members/>

<sup>28</sup><https://webauthn.guide/>

<sup>29</sup><https://auth0.com/blog/introduction-to-web-authentication/>

<sup>30</sup><https://www.freecodecamp.org/news/intro-to-webauthn/>

<sup>31</sup><https://www.freecodecamp.org/news/intro-to-webauthn/>

<sup>32</sup><https://hackertarget.com/cowrie-honeypot-analysis-24hrs/>



Figure 4: Auth

- Session, IP, user agent, proxy, ...
- Utiliser un framework
  - ces bonnes pratiques sont déjà implémentées
- Suites et logiciels de test

## Références

- Référence
  - OWASP<sup>33</sup>, webinar fr 2016<sup>34</sup>
  - WebAuthn : w3c<sup>35</sup>, MDN<sup>36</sup>
- Exemples, explications
  - Présentation XSS et CSRF<sup>37</sup> en français
  - Protection CSRF<sup>38</sup> en français
- Utilitaires, tutos, exercices
  - Web Goat<sup>39</sup>
  - Insecure Labs<sup>40</sup>
  - Google-Gruyere<sup>41</sup>

## Sources

---

<sup>33</sup>[https://www.owasp.org/index.php/Main\\_Page](https://www.owasp.org/index.php/Main_Page)

<sup>34</sup><https://www.youtube.com/watch?v=pHI2zitLph8>

<sup>35</sup><https://www.w3.org/TR/webauthn/>

<sup>36</sup>[https://developer.mozilla.org/en-US/docs/Web/API/Web\\_Authentication\\_API](https://developer.mozilla.org/en-US/docs/Web/API/Web_Authentication_API)

<sup>37</sup>[https://www.journaledunet.com/developpeur/tutoriel/php/031030php\\_nexen-xss1.shtml](https://www.journaledunet.com/developpeur/tutoriel/php/031030php_nexen-xss1.shtml)

<sup>38</sup><https://www.apprendre-php.com/tutoriels/tutoriel-39-introduction-aux-cross-site-request-forgeries-ou-sea-surf.html>

<sup>39</sup><https://www.owasp.org/index.php/Webgoat>

<sup>40</sup><https://www.insecurelabs.org/task>

<sup>41</sup><https://google-gruyere.appspot.com/>