Les Vulnérabilités

Par Seïf-Eddin Bouguerouche

Sommaire

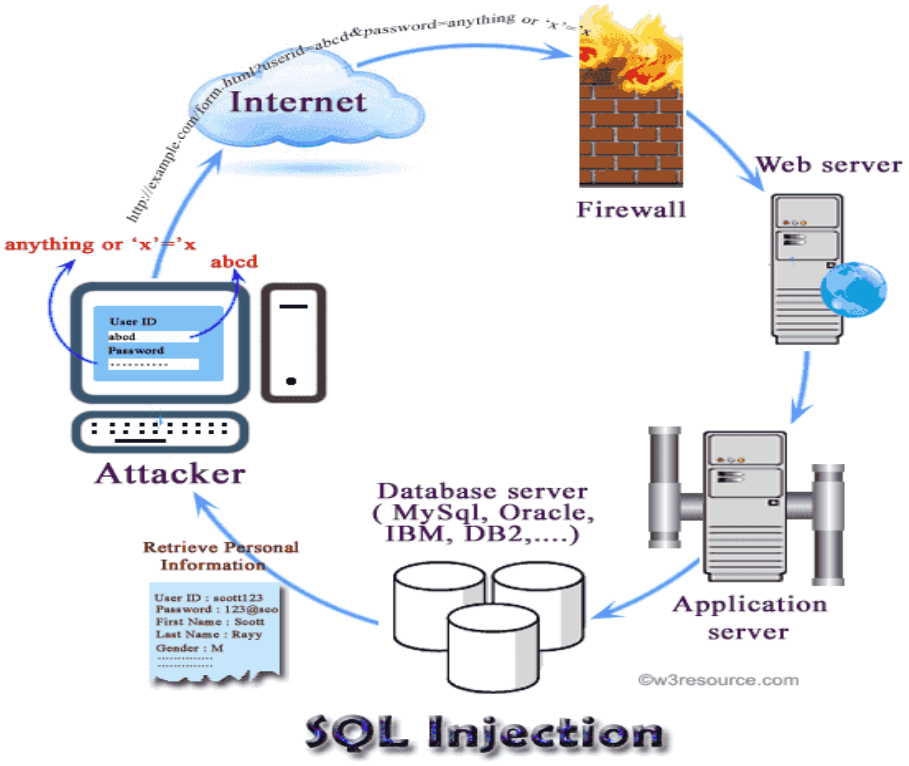
Introduction

Dans le domaine de la sécurité informatique, une vulnérabilité ou faille est une faiblesse dans un système informatique permettant à un attaquant de porter atteinte à l'intégrité de ce système, c'est-à-dire à son fonctionnement normal, à la confidentialité ou à l'intégrité des données qu'il contient.

Ces vulnérabilités sont la conséquence de faiblesses dans la conception, la mise en œuvre ou l'utilisation d'un composant matériel ou logiciel du système, mais il s'agit souvent d'anomalies logicielles liées à des erreurs de programmation ou à de mauvaises pratiques. Ces dysfonctionnements logiciels sont en général corrigés à mesure de leurs découvertes, mais l'utilisateur reste exposé à une éventuelle exploitation tant que le correctif (temporaire ou définitif) n'est pas publié et installé. C'est pourquoi il est important de maintenir les logiciels à jour avec les correctifs fournis par les éditeurs de logiciels. La procédure d'exploitation d'une vulnérabilité logicielle est appelée exploit.

Les vulnérabilités les plus courantes sont :

- La vulnérabilité « SQL Injection» : les hackers procèdent par la technique des injections SQL pour s'introduire discrètement sur le système de l'entreprise dans le but de s'approprier illégalement des informations confidentielles. Les pirates injectent un « malware » par le biais de codes JavaScript, dans le but de porter atteinte à l'entreprise.



Exemple de ce qu'est une injection SQL :

Considérons un site web dynamique (programmé en PHP dans cet exemple) qui dispose d'un système permettant aux utilisateurs possédant un nom d'utilisateur et un mot de passe valides de se connecter. Ce site utilise la requête SQL suivante pour identifier un utilisateur :

```
SELECT uid FROM Users WHERE name = '(nom)' AND password =
'(mot de passe hashé)';
```

L'utilisateur Dupont souhaite se connecter avec son mot de passe « truc » hashé en MD5. La requête suivante est exécutée :

```
SELECT uid FROM Users WHERE name = 'Dupont' AND password =
'45723a2af3788c4ff17f8d1114760e62';
```

Imaginons à présent que le script PHP exécutant cette requête ne vérifie pas les données entrantes pour garantir sa sécurité. Un hacker pourrait alors fournir les informations suivantes :

- Utilisateur : `Dupont';--`
- Mot de passe : n'importe lequel

La requête devient :

```
SELECT uid FROM Users WHERE name = 'Dupont';--' AND password =
'4e383a1918b432a9bb7702f086c56596e';
```

Les caractères `--` marquent le début d'un commentaire en SQL. La requête est donc équivalente à :

```
SELECT uid FROM Users WHERE name = 'Dupont';
```

Ainsi l'attaquant peut rajouter ce que bon lui semble en utilisant ce principe par exemple pour contourner le mot de passe avec la SQLI « ' OR 1=1 – »

La requête devient alors :

```
SELECT uid FROM Users WHERE name = 'Dupont' AND password = '' or 1 --';
```

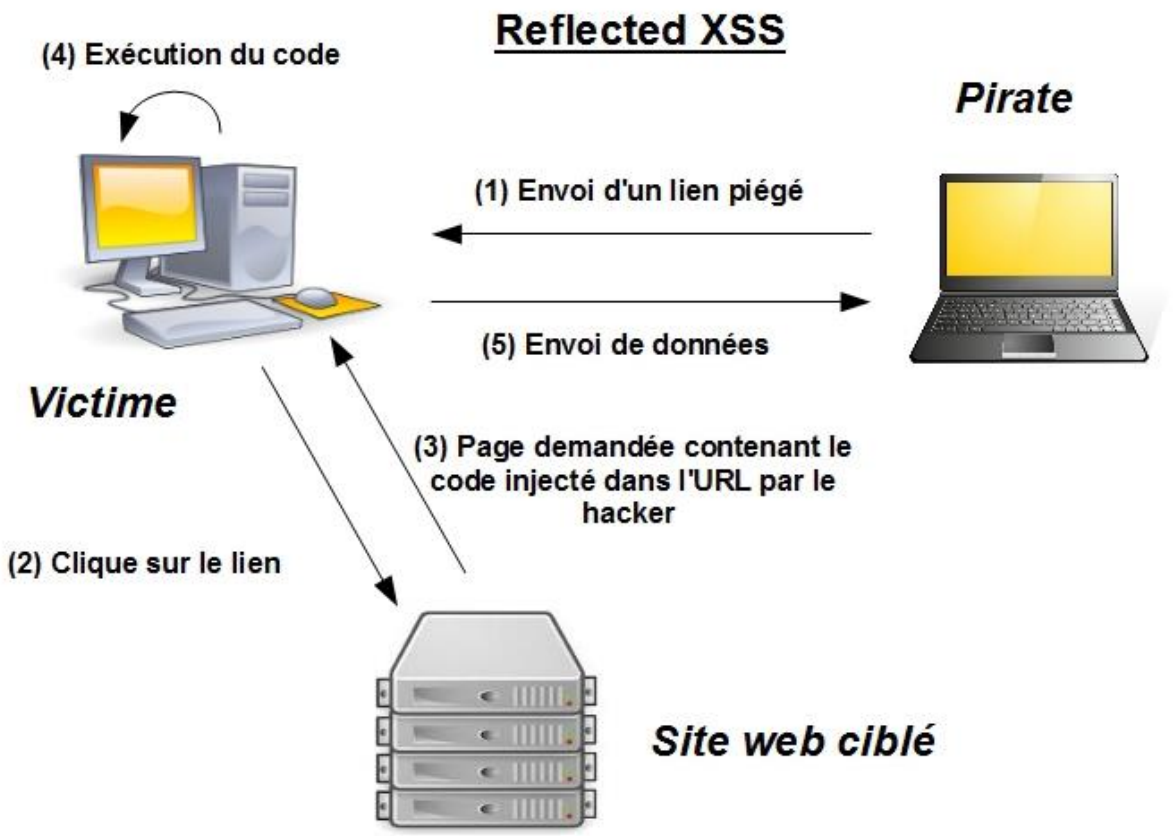
- La vulnérabilité « Cross Site Scripting» ou « XSS » : c'est le type d'attaque le plus fréquent sur les sites Internet. Les impacts pour ce type d'attaque sont multiples, pour ne pas citer que la redirection vers un site d'hameçonnage, le vol des sessions utilisateurs ou encore le vol de données sensibles.

  Il en existe deux types :

  - XSS réfléchi (ou non permanent)

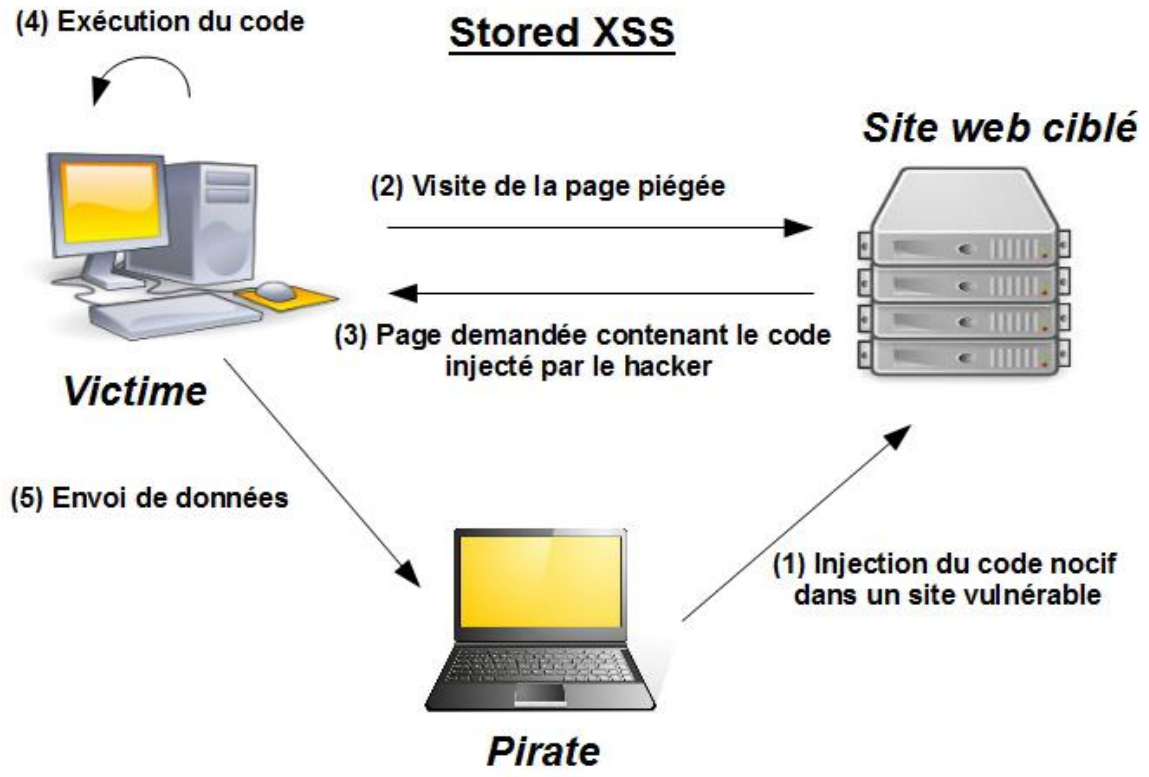
  Ce type de faille de sécurité, qui peut être qualifié de « non permanent », est de loin le plus commun.

  Il apparaît lorsque des données fournies par un client web sont utilisées telles quelles par les scripts du serveur pour produire une page de résultats. Si les données non vérifiées sont incluses dans la page de résultat sans encodage des entités HTML, elles pourront être utilisées pour injecter du code dans la page dynamique reçue par le navigateur client.

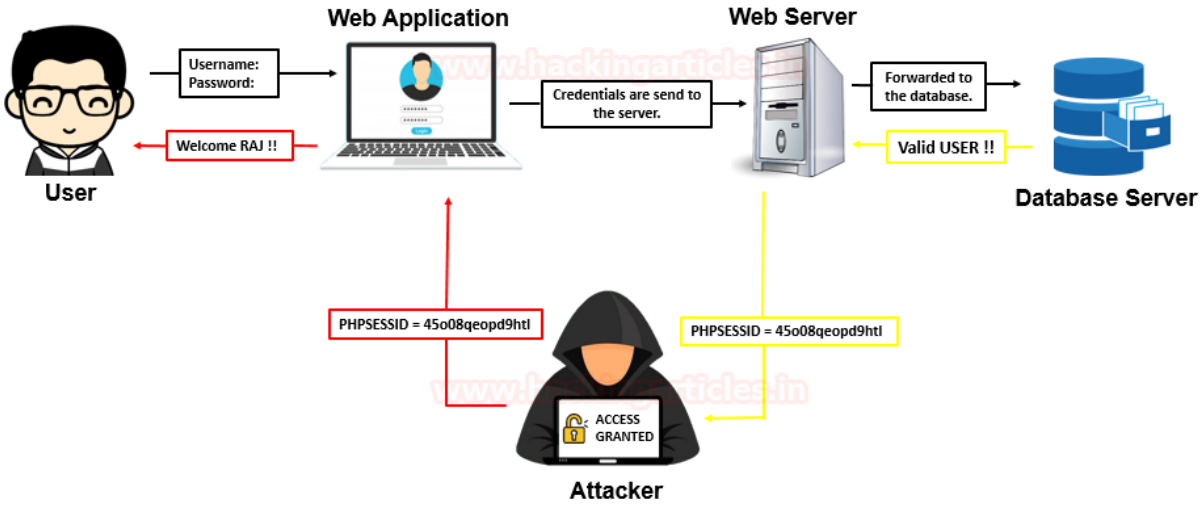


- XSS stocké (ou permanent)

Ce type de vulnérabilité, aussi appelé faille permanente ou du second ordre permet des attaques puissantes. Elle se produit quand les données fournies par un utilisateur sont stockées sur un serveur (dans une base de données, des fichiers, ou autre), et ensuite ré-affichées sans que les caractères spéciaux HTML aient été encodés.



- La vulnérabilité « Broken Authentification and Session Management » : cette vulnérabilité permet aux hackers d'usurper l'identité d'un utilisateur dont l'identifiant a été volé et utilisé à son insu. Ce type de cas arrive surtout pour les entreprises avec des réseaux dont les requêtes HTTP ne seraient pas en SSL.



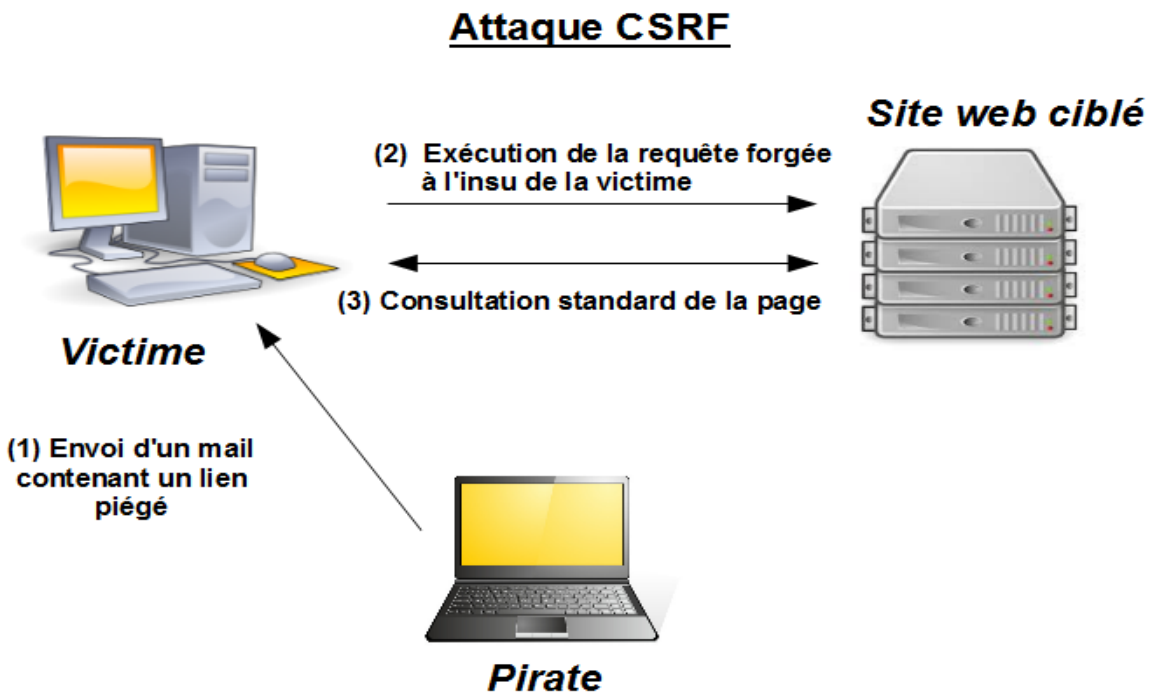
Voici un exemple de Broken Authentification and Session Management :

Ou l'attaquant récupère l'ID de session dand l'URL d'un site de reservation

```
http://example.com/sale/saleitems;jsessionid=2P0OC2JSNDLPSKHCJUN2JV?dest=Hawaii
```

- La vulnérabilité « Cross Site Request Forgery» : L'objet de l'attaque CSRF est de forcer des utilisateurs à exécuter une ou plusieurs requêtes non désirées sur un site donné, forgées par un utilisateur malintentionné. La victime choisie aura les privilèges nécessaires à l'exécution de la requête, voire une session encore active.

  Forger une requête revient à créer / falsifier une requête HTTP (par le biais d'une URL ou d'un formulaire principalement) pointant sur une action précise interne au site et néfaste pour la victime.



Example avec le cas d'un service bancaire :

Imaginez un bouton sur votre site de banque en ligne permettant d'effectuer un virement sur un compte extérieur ayant pour URL :

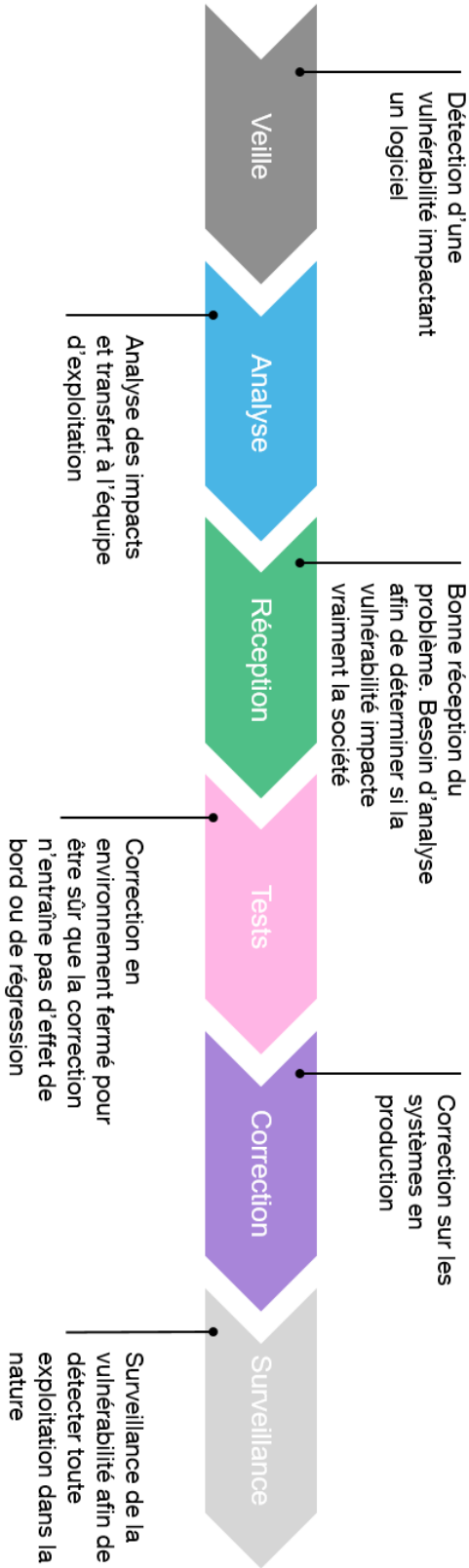
http://www.mabanque.com/transaction.php?dest=ID12345&somme=2000

Où le paramètre « dest » est l'identifiant du destinataire de la transaction et « somme », le montant de la transaction. Admettons ensuite que Mme Lambda soit allée voir ses relevés bancaires et que sa session soit toujours active. Imaginons ensuite qu'un utilisateur malveillant, connaissant la victime, ait décidé de lui envoyer un mail (en usurpant l'adresse e-mail d'un de ses proches).

L'e-mail contiendrait un lien vers cette action et avec pour destinataire son identifiant de compte. Si Mme Lambda clique sur le lien, alors la transaction sera effectuée contre son gré.

Si cette attaque est couplée d'une attaque XSS (en utilisant par exemple une balise img) sur le site bancaire, alors chaque utilisateur connecté sur le site sera victime de ce détournement de fonds.

Les étapes de découvertes d'une vulnérabilité



Solution face aux vulnérabilités :

Concernant l'injection SQL il y a plusieurs solutions comme :

- Les requêtes paramétré ainsi c'est le SGBD qui se charge d'échapper les caractères selon le type des paramètres.

- Utiliser des comptes utilisateurs SQL à accès limité (en lecture-seule) quand cela est possible.

- utiliser des expression régulière afin de rejeter tout caractère suspicieux entré par l'utilisateur ou utiliser les fonctions que propose php comme htmlspecialchars() et trim().

Ensuite pour les failles XSS il y a comme solution :

- utiliser la fonction htmlspecialchars() qui filtre les '<' et '>'

- Retraiter systématiquement le code HTML produit par l'application avant l'envoi au navigateur

Les solutions pour contrer les Broken Authentification and Session Management sont :

- L'authentification à plusieurs facteurs afin d'empêcher les attaques pas force brute, le bourrage d'identifiant ou des attaques par dictionnaires

- Des vérifications de mot de passe afin qu'ils ne soient pas considérés comme faible ou appartenant à une liste de mot de passe facile à trouvé

Pour se prémunir des attaques Cross Site Request Forgery :

- Éviter d'utiliser des requêtes HTTP GET pour effectuer des actions : cette technique va naturellement éliminer des attaques simples basées sur les images

- Demander des confirmations à l'utilisateur pour les actions critiques

- Demander une confirmation de l'ancien mot de passe à l'utilisateur pour changer celui-ci ou changer l'adresse mail du compte

- Utiliser des Tokens dans les formulaires est basé sur la création du token via le chiffrement d'un identifiant utilisateur

Sources :

- vuldb.com

- https://nvd.nist.gov/

- https://owasp.org/

- https://hdivsecurity.com/

-https://developer.mozilla.org/fr/docs/Glossary/Cross-site_scripting

- https://www.kaspersky.fr/resource-center/definitions/what-is-a-cross-site-scripting-attack

- https://blog.clever-age.com/fr/2014/02/10/owasp-xss-cross-site-scripting/

- https://www.cisa.gov/uscert/ncas/alerts

- https://cxsecurity.com/