Les failles de sécurité informatique

En informatique quand il y a un problème il faut d’abord regarder ce qu’il y a entre la chaise et le clavier. C’est là que se trouve la première faille de sécurité.

Et c'est souvent un développeur mal formé ou pire un développeur négligeant.

Pour éviter que ce soit vous : Il faut s’informer (la veille informatique), se former, combattre la procrastination, être rigoureux, se mettre à la place du hacker. Être paranoïaque et sur le sujet de la sécurité ce n’est pas un défaut.

**L’injection SQL**,

Comme le nom le laisse entendre c’est une attaque dirigée vers la base de données. Le malveillant profite d’une faiblesse (manque de filtre) de votre code pour placer le sien et faire ce qu’il veut ; récupérer des informations, détruire la bdd, voir diffuser des maliciels.

Les injections sont facilement évitables mais sont encore près de la moitié des attaques subies par de nombreuses sociétés petites ou grandes. Il y a plusieurs sortes d’injections SQL mais les plus courantes sont sur les variables avec des chaînes de caractères et pour les variables numériques.

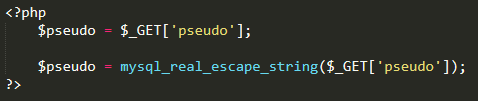
Pour l’éviter

Pour les variables avec des chaînes de caractères.

On sécurise avec mysql\_real\_escape\_string().

Cette fonction ajoute un antislash à ces caractères

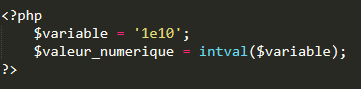




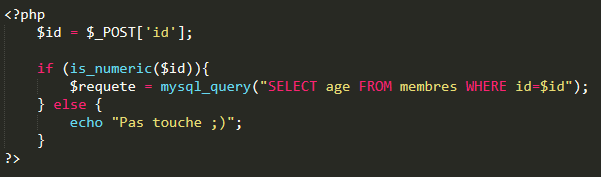
Pour les variables numériques

Il faut changer la variable pour qu’elle ne contienne que des nombres avec intval().

Vérifier si la variable contient vraiment un chiffre



Faire appel à la fonction is\_numeric() dans une condition if



**La faille XSS (cross-site scripting),**

C’est aussi une injection de contenu malveillant.

L’attaquant peut utiliser tous les langages pris en charge par le navigateur.

Il peut rediriger vers un autre site (hameçonnage) ou récupérer les cookies (vol de session).

Vol d’identité

Rendre la page quasi illisible voire totalement.

Pour la détecter on essaye de faire passer un script dans un formulaire.



Si une boite de dialogue s’affiche, alors il y a un problème.

Il y a deux sortes de failles XSS :

Les non-persistants

Les persistants

Les deux sortes se divisent en deux groupes :

* les vulnérabilités côté serveur elles sont dites traditionnelles.
* les vulnérabilités côté client qui sont basés sur le DOM.

On parle aussi de faille non permanente (la plus répandue).

Si les données ne sont pas vérifiées elles pourront servir à injecter du code malveillant.

La faille XSS stocké (permanente)

Plus puissante que la non permanente elle se produit quand les données renvoyées par le serveur contiennent des caractères spéciaux qui n’ont pas été encodés.

Les basées sur le DOM

?

Pour l’éviter

En PHP :

* utiliser la fonction htmlspecialchars ()​ qui filtre les '<' et '>' (cf. ci-dessus) ;
* utiliser la fonction htmlentities ()​ qui est identique à htmlspecialchars()​ sauf qu'elle filtre tous les caractères équivalents au codage HTML ou JavaScript.

**La faille CSRF (cross-site request forgery),**

C’est amener un utilisateur authentifié à utiliser une requête HTTP falsifiée qui pointe sur une action interne du site sans que ce dernier ne s’en rende compte.

Pour l’éviter

* Demander des confirmations à l'utilisateur pour les actions critiques, au risque d'alourdir l'enchaînement des formulaires.
* 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 [jetons](https://fr.wikipedia.org/wiki/Jeton_d%27authentification) de validité (ou *Token*) dans les formulaires. Ce système d'autorisation est basé sur la création d'un token via le chiffrement d'un identifiant utilisateur, un [nonce](https://fr.wikipedia.org/wiki/Nonce_(cryptographie)) et un horodatage. Le serveur doit vérifier la correspondance du jeton envoyé en recalculant cette valeur et en la comparant avec celle reçue[1](https://fr.wikipedia.org/wiki/Cross-site_request_forgery#cite_note-1).
* Éviter d'utiliser des requêtes HTTP GET pour effectuer des actions : cette technique va naturellement éliminer des attaques simples basées sur les images, mais laissera passer les attaques fondées sur [JavaScript](https://fr.wikipedia.org/wiki/JavaScript), lesquelles sont capables très simplement de lancer des requêtes HTTP POST.
* Effectuer une vérification du [référent](https://fr.wikipedia.org/wiki/R%C3%A9f%C3%A9rent_(informatique)) dans les pages sensibles : connaître la provenance du client permet de sécuriser ce genre d'attaques. Ceci consiste à bloquer la requête du client si la valeur de son référent est différente de la page d'où il doit théoriquement provenir.

**L'attaque par dictionnaire / par force brute**

C’est deux formes d’attaque qui vont souvent ensemble.

L’attaque par force brute c’est en gros essayer toutes les combinaisons possible pour cracker un mot de passe, avec les nouveaux ordinateurs les mots de passes pas très compliqués et courts ne résistent pas longtemps. Pour les autres on rajoute un dictionnaire, dans celui-ci il y a différents types de mots de passe possible qui vont servir à l’attaque.

Pour l’éviter

Il faut des mots de passe robuste, long avec majuscules, des caractères spéciaux, des chiffres

Randomiser le mp (générer un mp aléatoire)

Limiter les temps de connexions

Augmenter le coût par tentative (hash et captcha)

Renouveler ses mp.

Le salage.

**La faille upload**

Permet d’uploader un fichier avec une extension non autorisée dans lequel on injecte du code (en PHP). Souvent présente dans les scripts d’upload d’images.

Le vilain pourra prendre le contrôle de tout (applications et serveur).

C’est la faille la plus dangereuse.

Pour l’éviter

Le fameux **Never trust user input**

Vérifier la configuration d ‘Apache

Ne pas placer le .htaccess dans le répertoire d’upload

Ne pas permettre l’écrasement de fichier

Générer un nom aléatoire pour le fichier uploadé et enregistrer le nom dans une bdd.

Ne pas permettre de voir l’index of du répertoire d’upload.

Assigner les bonnes permissions au répertoire.

Vérifier le mime-type avec getimagesize () et l’extension du fichier.