**PROOF OF CONCEPT : INJECTION SQL**

**Explication du problème :**

Tout site internet ayant besoin d’une connexion à une base de données via une interaction d’un utilisateur par un formulaire peut être victime d’une « **injection SQL »** si le site n’est pas protégé.

Une injection SQL consiste à modifier une requête SQL en cours par l’injection d’un morceau de requête non prévu, souvent par le biais d’un formulaire. Le pirate peut ainsi accéder à la base de données, mais aussi modifier le contenu et donc compromettre la sécurité du système.

Elle peut se faire lorsqu’un utilisateur malveillant entre des caractères spécifiques à une base de données MySQL, comme pour des tirets « -- », permettant de mettre en commentaire la suite de la requête SQL en cours et donc de la modifier.

**Comment résoudre ce problème :**

Afin de contrer une injection SQL, il y a plusieurs méthodes existantes, or nous allons nous concentrer sur une de ces méthodes qui est **« la validation des champs coté client »**.

Le **« client »** est une application comme un navigateur qui fonctionne localement sur l'ordinateur et qui envoie des requêtes sur le serveur quand nécessaire. Dans le domaine informatique, le terme **« côté client »** fait référence à la relation d'un client dans une architecture client-serveur.

Le **« serveur »** fait référence à des opérations qui sont effectuées par le serveur dans la communication entre client et serveur dans un réseau informatique. C’est ce qui est fait en arrière-plan, des opérations cachées à l’utilisateur.

La **« validation des champs »**, est le contrôle de ce que peut saisir l’utilisateur dans un formulaire/un champ, avant d’être affectée par la base de données.

La validation des champs coté client fonctionne grâce à des scripts (JavaScript) permettant de vérifier si la saisie des informations est autorisée ; dans notre cas, empêcher que l’utilisateur entre des tirets dans le formulaire.

**Voici le code de la page d’un formulaire de connexion coté client vulnérable à une injection SQL :**



**Voici le code de la page d’un formulaire de connexion coté client et le code des script qui ne sont pas vulnérable à une injection SQL :**





Pour remédiez au problème nous avons rajouté un fichier script fait en « **JavaScript** » puis nous avons aussi placer des « **id** » sur les 2 champs que nous voulons contrôlé : « **mon\_input** » et « **mon\_input2** ». Et pour finir sur le formulaire nous avons utilisé la fonction « **onkeypress** » et on la relié avec nos 2 scripts de contrôle qui fait que lorsque on utilise une touche il va exécuté le script en question. Nous avons fais 2 scripts différent pour qu’il dissocie les 2 champs.

Dans la partie script nous avons fait 2 fonctions qui sont les mêmes juste le champ en question est différent. Nous avons une 1er variable qui permet de chercher numéro de la touche exécuté, puis dans la 2e variable nous convertissons ce code en chaîne de caractère donc le caractère de la touche, puis nous avons la 3e variable qui permet de stocker l’élément possédant l’id en question et pour finir nous avons la 4e variable qui permet de connaître le caractère que nous voulons interdire. Et donc lorsqu’on fait le test de «**si la toucher qu’on utilise est en position -1 dans la variable caractère (donc qui n’existe pas) elle prend la valeur du champ et ajoute la touche en question** ».

Donc le résultat de ce script fait que si on tape la touche qui est autorisé elle s’écrit sinon elle ne s’écrit pas.

Lien du GitHub : <https://github.com/Retiks/dimpreProject>