



### Sécurisez vos ressources images



Comme nous l'avons vu en cours, la protection d'accès à vos ressources est la première barrière pour éviter à tout utilisateur un peu trop curieux d'aspirer de fouiller l'arborescence de votre site.

Habituellement, la bonne pratique consiste à mettre dans un répertoire « **public** » toutes les ressources auxquelles le navigateur aura accès (*script js notamment*) et à restreindre tout le reste.

#### TO DO (1) :



- A l'aide d'un ou plusieurs fichiers **.htaccess**, **verrouillez** tous les accès directs à vos images (et dossiers)



### XSS Persistant



Pour bien se protéger – et surtout protéger votre site - des différentes attaques par injection de code, il est important de comprendre comment elles fonctionnent.

#### TO DO (2) :



- Commencez par réaliser une attaque **XSS persistante** en utilisant les entrées BDD de votre setlist (comme vu en cours). Cette injection affichera secrètement votre **SID** de session si vous êtes connecté (*vous ferez un **console.log** de tout ça en gros*)



#### TO DO (3) :



- Puis vous effectuerez les corrections nécessaires pour empêcher ce genre d'attaques à l'avenir.



#### Tips

- Aujourd'hui la plupart des navigateurs web sont capables de détecter lorsque des morceaux de script proviennent de **requêtes GET** ou **POST**. Cela ne pourra empêcher une attaque XSS Persistante car le code qui provient de la BDD sera exécuté ; mais vous n'en verrez pas l'effet tout de suite.  
Afin de passer outre cette protection et de voir immédiatement l'effet de votre attaque, je vous invite à rechercher comment **désactiver temporairement l'auditeur XSS** sur votre navigateur (**XSS auditor**).

### Injection SQL



Afin de simuler une injection SQL, vous devrez probablement abandonner (temporairement du moins) les bonnes pratiques d'utilisation de PDO et utiliser des requêtes non préparées.

#### TO DO (4) :



- Réalisez une **injection SQL** qui va vous permettre de vous connecter à votre site (pour modifier la setlist) **sans connaître le mot de passe admin**.



#### TO DO (5) :



- Vous corrigerez ensuite le code, de manière à **prévenir et empêcher cette injection SQL**.



### Un peu de Hash, un peu de Sel ...



Nous avons vu en cours que le principe de **Hachage / Salage des mots de passes** dans une BDD et probablement la 1<sup>ère</sup> barrière efficace contre les attaques de types « *Dictionnaire* », « *Brut Force* » ou « *Rainbow Tables* ».

#### TO DO (6) :



- Appliquez « simplement » le « **Hachage / Salage** » sur votre table utilisateurs.



httpSSSSSSSS



: Pour les Warriors !

L'utilisation **d'HTTPS** et l'encryptage de toutes les requêtes http qui en découle et sans nul doute la protection la plus efficace contre les attaques du type « *Man In the Middle* ».

Cela nécessite un certificat (la clé publique) approuvé par un organisme d'autorité supérieur et vous n'aurez pas le loisir d'obtenir ce genre de certificat pour notre TP.

Heureusement pour vous, afin de vérifier le comportement du protocole https, on peut utiliser ce que l'on appelle des certificats « auto-signés ».

### TO DO (7) :



- En cherchant un peu, trouvez et implémentez la procédure permettant d'utiliser **l'https** sur votre **XAMPP** pour le host « **localhost** ».



### TO DO (8) :



- Téléchargez / Installez **WIRESHARK** et utilisez-le pour vérifier que vos échanges soient bien cryptés.
- Comparez-les avec des échanges **http non sécurisés**.

