Protection CSRF simple

En sécurité des systèmes d'information, le Cross-Site Request Forgery, abrégé CSRF ou XSRF, est un type de vulnérabilité des services d'authentification web.

CSRF est synonyme de falsification de requête intersite. C'est une sorte d'attaque dans laquelle un pirate vous oblige à exécuter une action contre un site Web sur lequel vous êtes actuellement connecté.

Principe de protection :

Génération coté serveur d'un « token » unique sauvegardé dans la session de l'utilisateur et inséré dans la page html afin que cette dernière renvoi ce token.

A la soumission du formulaire ou à toute requête on récupère le token et on le compare à celui sauvegardé en session.

1. Création du « token » et mis en SESSION

$token = bin2hex(random_bytes(32)) ; // utilisation de la transformation en hexadécimal de 32 octets aléatoires

ou

$token = md5(uniqid(mt_rand(), true)); // utilisation du hash md5 sur une valeur unique de 23 caractères donnée par uniqid

Mis en session du token

$_SESSION['nom_du_token'] = $token ;

2. Insertion du « token » dans la page html

Pour un formulaire il y a 2 possibilités :

Ajout d'un champ caché :

<input type="hidden" name="nom_du_token" value="la_valeur_du_token">

Ajout dans la querystring :

<form methode="post" action="page_de_soumission.php?non_du_token=valeur_du_token">

Pour un lien : exemple suppression d'un produit d'id 15

Supprimer

3. Validation coté serveur

Récupérer le token provenant du formulaire soumis :

$token = filter_input(INPUT_POST, 'token', FILTER_SANITIZE_STRING);

Tester s'il existe (filter_input renvoi false) et le comparer avec $_SESSION['nom_du_token']

```
if (!$token || $token !== $_SESSION['token']) {
   // retourner le code statut 405
   header($_SERVER['SERVER_PROTOCOL'] . ' 405 Method Not Allowed');
   exit;
} else {
   // process the form
}
```

Il est possible d'utiliser la fonction php qui renvoi un booléen :

hash_equals($token, $_SESSION['token']*) ;*

Pour Ajax

Interdire le « cross origin » : les requêtes inter domaines.

Mettre le « token » dans un champ caché ou dans une variable JavaScript qui sera transmise par la requête ajax.

Pour encore plus de protection : définir un TTL

Le Time To Live (« temps de vie » ou « durée de vie »), abrégé TTL, indique le temps pendant lequel une information doit être conservée, ou le temps pendant lequel une information doit être gardée en cache.

Ajouter une deuxième clé en session « nom_du_token_TTL » dont la valeur est le « timestamp » actuel :

Utiliser la fonction « time() » qui retourne l'heure courante, mesurée en secondes depuis le début de l'époque UNIX, (1er janvier 1970 00:00:00 GMT).

Lors de la validation du « token », tester si le « timestamp » sauvegardé en session augmenté d'un certain nombre de secondes (ex 1600) est inférieur au « timestamp » actuel.

Si ce n'est pas le cas cela veut dire que le délai est expiré.