FAILLE WEB

Verrou par verrou et l'un après l'autre, aucun système ne résistera !

JUSTAL KEVIN

2014-2015

Justal Kevin - justal.kevin@gmail.com

Table des matières

1 Fonctions PHP

1.1 Logique humaine

1.1.1 L'ancrage du "str replace" ou le mauvais filtre

Le filtre est un classique du WEB. Toutes les chaines qui entrent doivent être analysé pour empecher l'utilisateur d'entrer des choses à des fins malicieuses. Même ce principe de base qui consiste à simplement éliminé une chaine dans une chaine peut avec une simple erreur humaine permettre de faire un peu tout et n'importe quoi. Prenons l'exemple d'un simple formulaire :

```
<form method="POST" action="">
    <input type="text" name="secret"><br>
    <input type="submit" name="send" value="Envoyer">
</form>
```

Il s'agit d'un simple champ texte que j'envoie par une méthode POST sur la même page que ce bout de code. Si vous ne comprenez pas,ce n'est pas bien grave, ceci ne sert qu'à mettre un contexte. Maintenant imaginons que nous souhaitons récupéré la variable et filtrer tout Javascript :

```
$var=str_replace("<script>","",$_POST["secret"]);
```

J'ai retrouvé ce bout de code sur plusieurs site et ceci m'a légèrement fait sourire. Comme on peux le voir sur cette ligne ci-dessus, nous remplaçons toutes les balises <script> par une chaine vide. Il y a pourtant ici 2 erreurs flagrantes.
La première demande de connaitre exactement ce que fait la balise str_replace. Dans le HTML, il est possible d'utiliser des balises écrites en minuscule ou en majuscule. Or, str_replace respecte la case, il m'est donc possible de rentrer une balise <SCRIPT> sans que le filtre ne s'alarme. La deuxième est simple d'ordre logique, que se passe-t-il si j'envoie ceci via mon script PHP :

```
<sc<script>ript>alert("HAHA")</sc<script>ript>
```

Dans cette chaine, si nous remplaçons script par une chaine vide nous obtenons :

```
<script>alert("HAHA")<script>
```

On réussit ainsi à bypasser le filtre de manière assez simple.

1.1.2 Solutions

Pourquoi ne pas simplement utiliser les fonctions PHP : htmlentities() et htmlspecialchars().

2 Directory transversal - Attaque sur le htaccess

2.1 Explications

2.1.1 Qu'est ce que la technologie htaccess ?

Les fichiers .htaccess sont des fichiers de configuration de Apache. Ils permettent de sécurisé via un mot de passe et un identifiant l'accés à une zone du serveur. Ils sont localisés et ne peuvent affecter que les répertoire où ils résident. La particularité d'une telle fonctionnalité apporte deux avantages. D'une part, on peux déléguer la gestion d'une partie du site sans donner le droit de gérer le serveur lui-même. D'autre part, les modifications sont prises en compte sans qu'il soit nécessaire de redémarrer le serveur HTML.

2.1.2 Les directives htaccess

Un fichier htaccess prend la forme suivante :

```
AuthUserFile /var/www/.htpasswd
AuthName "Visiteur, vous pénétrez dans une section réservée aux membres, veuillez vous identifier"
AuthType Basic
require Admin
```

La première directive, **AuthUserFile**, est le lien entre le htaccess et le htpasswd. Cette simple directive indique simplement où se situe le fichier htpasswd. Le chemin inscrit ici est généralement le chemin d'accès absolue mais il est possible de trouver aussi un chemin relative mais cela reste tout de même relativement rare.

La directive **AuthName** permet de spécifier un titre à la fenêtre de connexion.

La directive **AuthType** indique le type d'authentification. Il n'existe que deux types possibles : Basic ou Digest. Le premier type indique simplement que le mot de passe lors de l'authentification sera transmise en clair du client au serveur. C'est pourquoi cette méthode n'est pas à utiliser pour un transfert de donnée sensible. Le type Digest est un sois-disant type améliorant la sécurité du transfert, cependant de nombreuses failles existent ici. Ce qui rend ce type inutile car plus lourd à mettre en place et pas vraiment sécurisé.

La directive **requiere** spécifie simplement qui est autorisé à accéder à cette partie du site. On ira donc chercher dans le fichier htpasswd l'utilisateur Admin pour comparer le mot de passe.

2.1.3 Les directives htpasswd

Un fichier htpasswd prend la forme suivante :

```
admin1 :$apr1$Ikl22aeJ$w1uWlBGlbatPnETT2XGx..
admin2 :$apr1$yJnQGpTi$WF5eCC/8lKsgBKY7fvag60
```

Un fichier httpasswd prend toujours la forme ci-dessus. Ce fichier lie un utilisateur à un password crypté via un algorithme comme SHA, DES, MD5...

2.2 La navigation transversale ou Directory transversal

Pour expliqué la faille, je prendrais un exemple. Le site w3challs.com dispose d'un exemple sur cette faille du système. Avant même de commencer l'expérimentation, il faut encore un peu d'explication pour comprendre la faille. Cette faille réside dans le PHP du site et en particulier dans la balise include.

```
$template = 'red.php';
if (isset($_COOKIE['TEMPLATE']))
    $template = $_COOKIE['TEMPLATE'];
include ("/home/users/phpguru/templates/" . $template);
```

Ici, le fait que dans l'include, on ne vérifie pas que le résultat attendu soit une page .html ou .php, on peux alors imaginer de modifié la variable $template. Il y a plusieurs manières de procédé qui dépendent de la manière dont est implémenté le code du site que l'on souhaite attaquer : Par l'URL, Par la requète HTML...

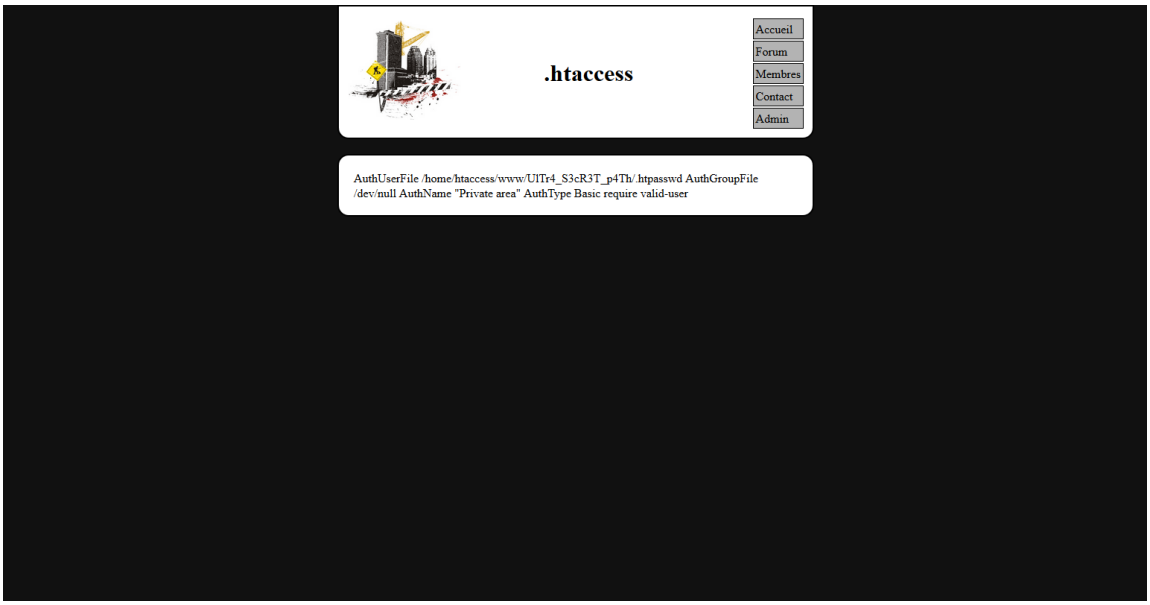
Dans le cas ci-dessus, on utilise $_COOKIE, on en retient donc que la page ou la destination vers où pointe $template a été enregistré sur l'ordinateur de l'utilisateur. Il est donc possible de modifier la requète avant de l'envoyer au serveur.

Imaginons alors que la variable template soit "../../../.htaccess", on remonte alors les repertoires jusqu'au root. Si le systeme de la machine est Linux, il existe alors forcement un repertoire etc/passwd. Maintenant, sur les serveur en ligne, les developpeurs posent généralement ces dossiers dans des repertoires comme admin/.htaccess ou encore pass/.htaccess. Il suffit de faire preuve d'un peu d'imagination pour trouver où pourrait se trouver le fichier.

2.3 Exploitation

Sur w3chall.com, on trouve une page avec cette faille. La première chose à faire est donc de chercher le fichier .htaccess. En forcant, on trouve que le fichier assez rapidement. Dans la barre d'adresse, il suffit de finir l'adresse par :

```
/?page=../admin/.htaccess
```



Bien entendu, avant d'arriver à cela, j'ai tapé plusieurs autres chemins comme ./admin/.htaccess ou encore .htaccess. Une fois ici, on remarque la générosité du système qui nous donne l'emplacement exacte du fichier htpasswd. Il suffit alors de s'y rendre :

```
?page=../UlTr4_S3cR3T_p4Th/.htpasswd
```

Et voila qu'apparaisent sous vos yeux les passwords et logins qui se trouvent dans le fichier htpasswd. Ils sont bien entendu crypté mais avec l'utilisation d'un logiciel tiers comme John the Ripper, la reconstitution du password d'origine n'est qu'une question de temps.

2.4 Variante

La première correstion apporté par les développeurs furent d'ajouter l'extension du fichier à la fin de l'include. Ce qui donnait un lien finissant toujours par .html ou .php. Il devient alors théoriquement impossible de rentrer quelques choses finissant par aucune extension comme nous l'avons fait jusqu'à maintenant. Erreur ! Il est possible de terminer une chaine à l'endroit où l'on souhaite en ajoutant le charatère de fin de chaine : le null ou encore %00. Ce qui dans notre cas donnerais :

```
/ ?page=../admin/.htaccess%00.html
```

Cependant le serveur ne lira cette chaine que jusqu'au charatere null, le reste sera ignoré.

2.5 Solution

Pour se prémunir d'une telle attaque, pourquoi ne pas simplement escape tout les ../ ou %2e%2e/ (si encodé) lors des navigations.