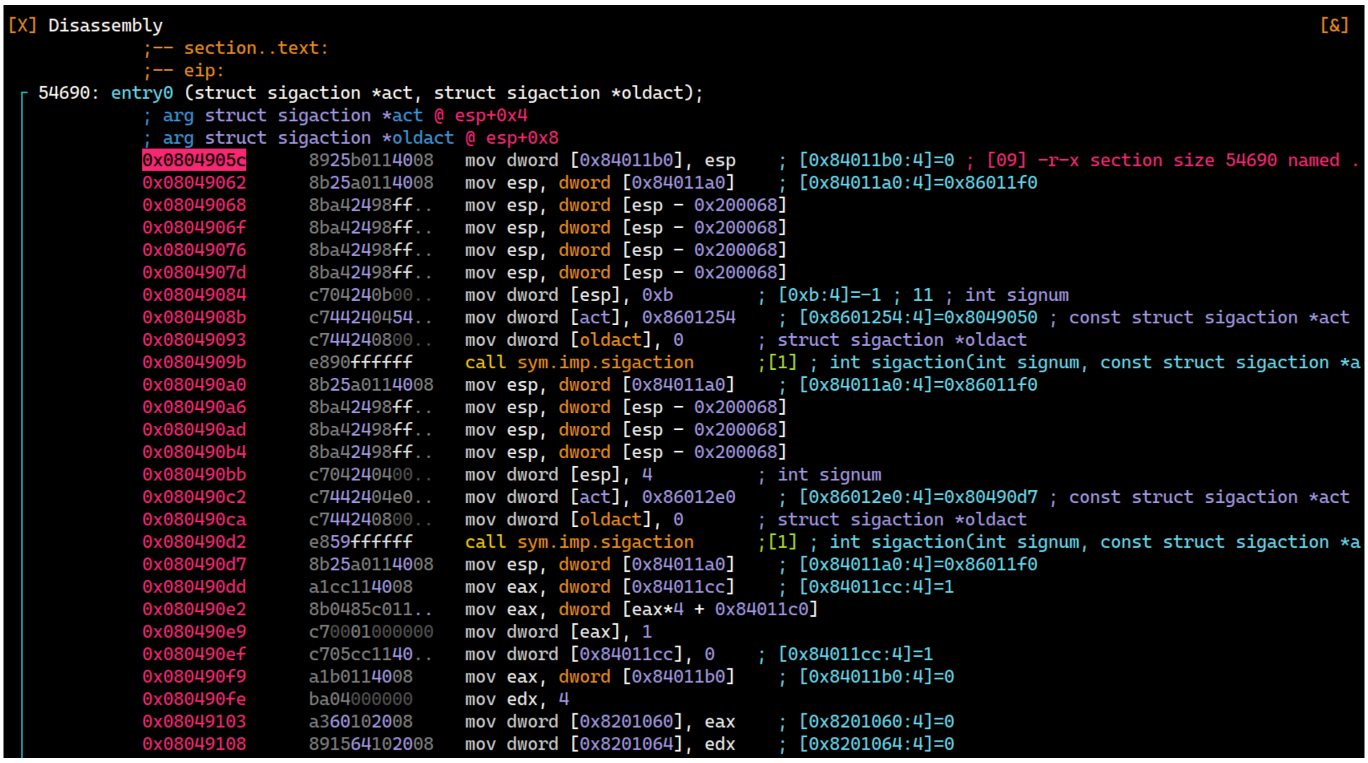
Walkthrough

This file contains the walkthrough for the challenge `Free Drinks`.

This challenge requires the participants to analyze the program and find out how to bypass the authentication and have the program to decrypt the flag. The original logic for this challenge is relatively simple, but is made far more sophisticated by compiling the program with movfuscator. The program looks like this in a disassembler:



```
[X] Disassembly                                                                                        [&]
            ;-- section..text:
            ;-- eip:
┌ 54690: entry0 (struct sigaction *act, struct sigaction *oldact);
│           ; arg struct sigaction *act @ esp+0x4
│           ; arg struct sigaction *oldact @ esp+0x8
│           0x0804905c      8925b0114008   mov dword [0x84011b0], esp    ; [0x84011b0:4]=0 ; [09] -r-x section size 54690 named .
│           0x08049062      8b25a0114008   mov esp, dword [0x84011a0]    ; [0x84011a0:4]=0x86011f0
│           0x08049068      8ba42498ff..   mov esp, dword [esp - 0x200068]
│           0x0804906f      8ba42498ff..   mov esp, dword [esp - 0x200068]
│           0x08049076      8ba42498ff..   mov esp, dword [esp - 0x200068]
│           0x0804907d      8ba42498ff..   mov esp, dword [esp - 0x200068]
│           0x08049084      c704240b00..   mov dword [esp], 0xb         ; [0xb:4]=-1 ; 11 ; int signum
│           0x0804908b      c744240454..   mov dword [act], 0x8601254    ; [0x8601254:4]=0x8049050 ; const struct sigaction *act
│           0x08049093      c744240800..   mov dword [oldact], 0        ; struct sigaction *oldact
│           0x0804909b      e890ffffff     call sym.imp.sigaction       ;[1] ; int sigaction(int signum, const struct sigaction *a
│           0x080490a0      8b25a0114008   mov esp, dword [0x84011a0]    ; [0x84011a0:4]=0x86011f0
│           0x080490a6      8ba42498ff..   mov esp, dword [esp - 0x200068]
│           0x080490ad      8ba42498ff..   mov esp, dword [esp - 0x200068]
│           0x080490b4      8ba42498ff..   mov esp, dword [esp - 0x200068]
│           0x080490bb      c704240400..   mov dword [esp], 4           ; int signum
│           0x080490c2      c7442404e0..   mov dword [act], 0x86012e0    ; [0x86012e0:4]=0x80490d7 ; const struct sigaction *act
│           0x080490ca      c744240800..   mov dword [oldact], 0        ; struct sigaction *oldact
│           0x080490d2      e859ffffff     call sym.imp.sigaction       ;[1] ; int sigaction(int signum, const struct sigaction *a
│           0x080490d7      8b25a0114008   mov esp, dword [0x84011a0]    ; [0x84011a0:4]=0x86011f0
│           0x080490dd      a1cc114008     mov eax, dword [0x84011cc]    ; [0x84011cc:4]=1
│           0x080490e2      8b0485c011..   mov eax, dword [eax*4 + 0x84011c0]
│           0x080490e9      c70001000000   mov dword [eax], 1
│           0x080490ef      c705cc1140..   mov dword [0x84011cc], 0     ; [0x84011cc:4]=1
│           0x080490f9      a1b0114008     mov eax, dword [0x84011b0]    ; [0x84011b0:4]=0
│           0x080490fe      ba04000000     mov edx, 4
│           0x08049103      a360102008     mov dword [0x8201060], eax    ; [0x8201060:4]=0
│           0x08049108      891564102008   mov dword [0x8201064], edx    ; [0x8201064:4]=0
```

Similar to past movfuscator challenges as seen in other CTF events, participants will need to use tools like demovfuscator and employ other static and dynamic analysis techniques to find out how this program works and where to patch to bypass the authentication so the program could proceed to decrypt the flag. The authentication algorithm can be expressed as:

```
username_hash = djb2(username)
password_hash = djb2(password)
valid ⇔ (username_hash = 967993393) ∧ (password_hash = 2248063266)
```

Basically, it checks if the djb2 hash, with a non-standard of 7741, of the username and password matches the set of preset values. The following credentials were used to generate the preset numbers:

- Username: `Lom Tevasseur`
- Password: `Ihopewegetmoresponsorsnextyear`

While it is possible for the participants to brute-force a set of credentials that will produce the same djb2 hashes, it is far too time-consuming. This challenge is purposefully made easier by ensuring that the flag decryption process does not require the correct username and password to be provided. In other words, the participants can simply patch the corresponding assembly of the following if-statement to bypass the authentication and make the program decrypt the flag:

```
if (username_hash == expected_username_hash && password_hash ==
expected_password_hash) {
```

The flag, after decryption, will be `flag{who_DOeSNT_LiKE_a_FREE_NegRonI}`.