The challenge provides an associated binary, which is also hosted on a remote server which exposes a port. Connection is established with the remote server through netcat as usual

Reversing

Static

A straightforward **objdump** provides three main insights:

1. A **scanf** call
2. A bunch of “movzx” (attached below)
3. A **strcmp,** followed by a **fopen** call, the functionality of which can be inferred when interacting with the server

Intuitively, the program seems to take some input, perform a bunch of deterministic byte swaps (notice **BYTE PTR**), compare it with “some string”, perform a **fopen** (likely the flag), and dump the content to **stdout**.

```
12b9:   e9 ed 00 00 00          jmp    13ab <main+0x162>
12be:   0f b6 45 e2             movzx  eax,BYTE PTR [rbp-0x1e]
12c2:   88 45 d6                mov    BYTE PTR [rbp-0x2a],al
12c5:   0f b6 45 e8             movzx  eax,BYTE PTR [rbp-0x18]
12c9:   88 45 e2                mov    BYTE PTR [rbp-0x1e],al
12cc:   0f b6 45 d6             movzx  eax,BYTE PTR [rbp-0x2a]
12d0:   88 45 e8                mov    BYTE PTR [rbp-0x18],al
12d3:   0f b6 45 e4             movzx  eax,BYTE PTR [rbp-0x1c]
12d7:   88 45 d6                mov    BYTE PTR [rbp-0x2a],al
12da:   0f b6 45 ea             movzx  eax,BYTE PTR [rbp-0x16]
12de:   88 45 e4                mov    BYTE PTR [rbp-0x1c],al
12e1:   0f b6 45 d6             movzx  eax,BYTE PTR [rbp-0x2a]
12e5:   88 45 ea                mov    BYTE PTR [rbp-0x16],al
12e8:   0f b6 45 e3             movzx  eax,BYTE PTR [rbp-0x1d]
12ec:   88 45 d6                mov    BYTE PTR [rbp-0x2a],al
12ef:   0f b6 45 eb             movzx  eax,BYTE PTR [rbp-0x15]
12f3:   88 45 e3                mov    BYTE PTR [rbp-0x1d],al
12f6:   0f b6 45 d6             movzx  eax,BYTE PTR [rbp-0x2a]
12fa:   88 45 eb                mov    BYTE PTR [rbp-0x15],al
12fd:   0f b6 45 e6             movzx  eax,BYTE PTR [rbp-0x1a]
1301:   88 45 d6                mov    BYTE PTR [rbp-0x2a],al
1304:   0f b6 45 e9             movzx  eax,BYTE PTR [rbp-0x17]
1308:   88 45 e6                mov    BYTE PTR [rbp-0x1a],al
130b:   0f b6 45 d6             movzx  eax,BYTE PTR [rbp-0x2a]
130f:   88 45 e9                mov    BYTE PTR [rbp-0x17],al
1312:   0f b6 45 e4             movzx  eax,BYTE PTR [rbp-0x1c]
1316:   88 45 d6                mov    BYTE PTR [rbp-0x2a],al
1319:   0f b6 45 e8             movzx  eax,BYTE PTR [rbp-0x18]
131d:   88 45 e4                mov    BYTE PTR [rbp-0x1c],al
1320:   0f b6 45 d6             movzx  eax,BYTE PTR [rbp-0x2a]
1324:   88 45 e8                mov    BYTE PTR [rbp-0x18],al
1327:   48 b8 66 43 54 35 39    movabs rax,0x5357383935544366
```

A simple **strings** then leaks the string that is compared as **fCT598WSH**.

Dynamic

The same analysis as above can also be performed through GDB, which will make it easier to construct/reverse the mangling logic.

Mangling Logic

Either way, the following mangling logic comes up:

1. Input a string
2. Swap 1st and 7th characters
3. Swap 3rd and 9th characters
4. Swap 2nd and 10th characters
5. Swap 5th and 8th characters
6. Swap 3rd and 7th characters
7. Compare against **fCT598WSoU** and dump the flag