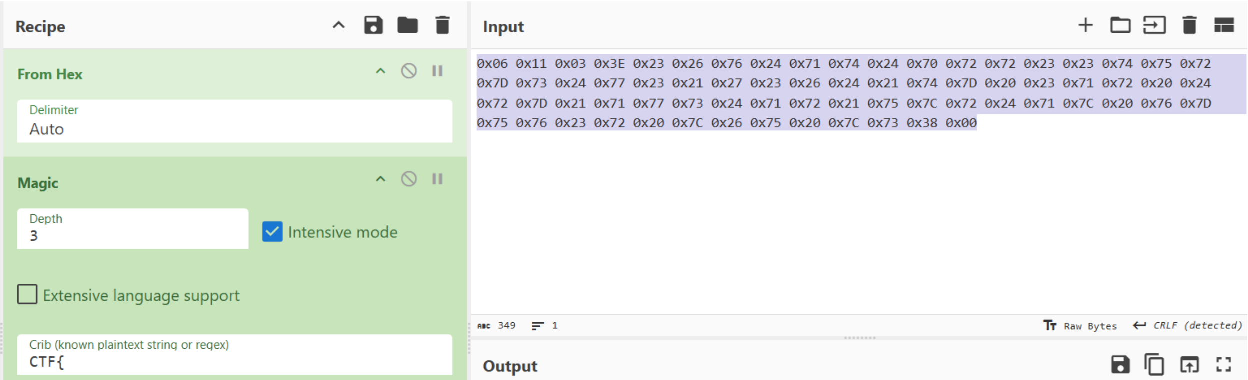
So, we got a password checker! And it mentions strlen(enc_flag)…

BUT HOW DO WE GET THE FLAG?

First, in IDA, I extracted all the enc_flag bytes (they can be found in unk_2008), and formatted them neatly with the help of chatgpt:

0x06 0x11 0x03 0x3E 0x23 0x26 0x76 0x24 0x71 0x74 0x24 0x70 0x72 0x72 0x23 0x23 0x74 0x75 0x72 0x7D 0x73 0x24 0x77 0x23 0x21 0x27 0x23 0x26 0x24 0x21 0x74 0x7D 0x20 0x23 0x71 0x72 0x20 0x24 0x72 0x7D 0x21 0x71 0x77 0x73 0x24 0x71 0x72 0x21 0x75 0x7C 0x72 0x24 0x71 0x7C 0x20 0x76 0x7D 0x75 0x76 0x23 0x72 0x20 0x7C 0x26 0x75 0x20 0x7C 0x73 0x38 0x00

So, this is the encrypted flag… dcode's cipher identifier didn't help with this or the decoded string, so let's try cyberchef's magic!



From Hex

Delimiter
Auto

Magic

Depth
3

☑ Intensive mode

□ Extensive language support

Crib (known plaintext string or regex)
CTF{

```
0x06 0x11 0x03 0x3E 0x23 0x26 0x76 0x24 0x71 0x74 0x24 0x70 0x72 0x72 0x23 0x23 0x74 0x75 0x72
0x7D 0x73 0x24 0x77 0x23 0x21 0x27 0x23 0x26 0x24 0x21 0x74 0x7D 0x20 0x23 0x71 0x72 0x20 0x24
0x72 0x7D 0x21 0x71 0x77 0x73 0x24 0x71 0x72 0x21 0x75 0x7C 0x72 0x24 0x71 0x7C 0x20 0x76 0x7D
0x75 0x76 0x23 0x72 0x20 0x7C 0x26 0x75 0x20 0x7C 0x73 0x38 0x00
```

abc 349 ☰ 1 Raw Bytes CRLF (detected)

Using this recipe CyberChef managed to decrypt the flag.

Note: if you were curious about the password, you need a password such that 69 ^ char ^ len = 0, meaning the same char needed to be used repeatedly. For example, AAAA would be a valid password because 69 ^ 4 ^ A = 0.

Made with love by: AndreiCat