

Very Easy C Rev - 2

26 October 2025 17:13

CATEGORY: Cryptography



STEP BY STEP SOLUTION

The way the secret was encrypted is given so we just need to come up with another code to decode the secret.

This is the C program I used:

```
int main() {
    unsigned int newkey = 0xdeadbeaf;

    unsigned char data[] = {
        0x4c, 0xb9, 0x42, 0x04, 0x71, 0x00, 0x3c, 0xe6, 0x7f, 0x46, 0x9a, 0x2a, 0x41, 0x98, 0x29,
        0x9d, 0x34, 0xcf, 0x71, 0x20, 0x22, 0x2a, 0xe7, 0x48, 0x10, 0x76, 0x6e, 0x35, 0x35, 0xe7, 0x21,
        0x93, 0x78, 0xfd, 0x00
    };

    char enc5 = 'I'^((newkey) & 0xFF)^data[0];
```

```

for (int i = 0; i < 34; i++) {
    data[i] = data[i] ^ enc5 ^ (newkey & 0xFF);
    newkey = (newkey >> 1) | (newkey << 31);
}

for (int i = 0; i < 34; i++) {
    printf("%c", data[i]);
}
}

```

And it gives us the flag directly. (IDC{1_l1k3_70_5l33p_bu7_d33db34f})

A small note on why it works:

In the encryption algorithm each character at ith position in the secret is xor-ed with:

1. The last 8 bits of 0xdeadbeaf rotated right i times (rotating right implies 100 --> 010 --> 001 --> 100 and so on)
2. The 6th character after step 1 was performed.

This also makes the sixth character 0 and unretrievable....

BUT we do know that our flag is going to start with the letter 'I'

So we can retrieve the 6th encoded character.

Now that we have the missing piece we can simply undo the two xors and thus read the flag.

FLAG: IDC{1_l1k3_70_5l33p_bu7_d33db34f}