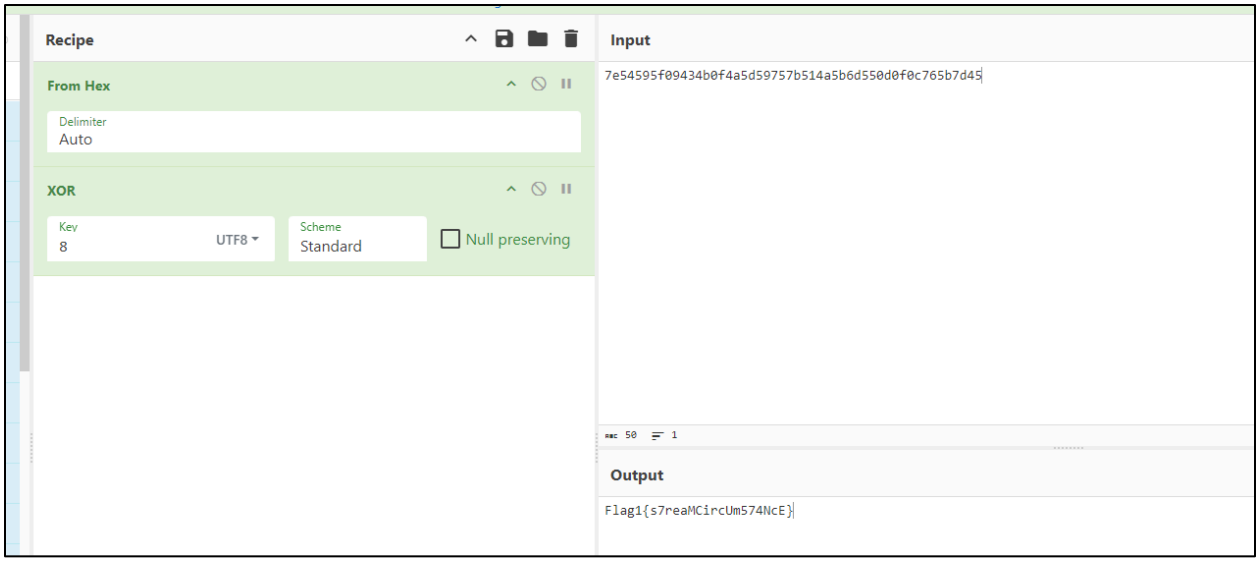
This challenge is very straightforward as it's made to showcase the similarities in reversing different architectures, all three files have the flag hardcoded and xored with a key so we will go through all three and get the flags.

First the power.exe we open it with ida and find the flag and the key 8

```
char key[2]; // [rsp+2Eh] [rbp-C2h] BYREF
char correct_output[64]; // [rsp+30h] [rbp-C0h] BYREF
char encrypted_input[64]; // [rsp+70h] [rbp-80h] BYREF
char input[56]; // [rsp+B0h] [rbp-40h] BYREF
const char *hexString; // [rsp+E8h] [rbp-8h]

_main(argc, argv, envp);
hexString = "7e54595f09434b0f4a5d59757b514a5b6d550d0f0c765b7d45";
hexToString("7e54595f09434b0f4a5d59757b514a5b6d550d0f0c765b7d45", correct_output);
strcpy(key, "8");
printf("Enter the correct passphrase: ");
scanf("%49s", input);
v3 = strlen(input);
xor_encrypt(input, encrypted_input, key, v3);
v4 = strlen(input);
if ( !memcmp(encrypted_input, correct_output, v4) )
{
  printf("Correct! The flag is: ");
  printf(input);
}
else
{
  puts("Wrong passphrase!");
}
return 0;
```

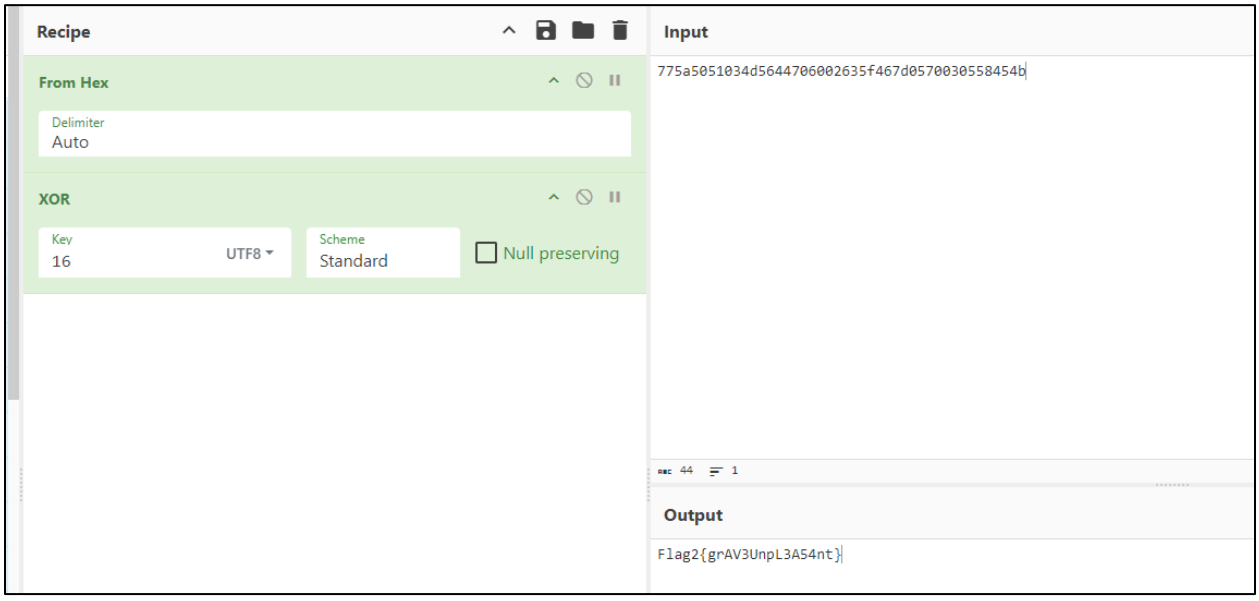
Using cyber chef the first flag is Flag1{s7reaMCircUm574NcE}



Next up is courage.out and we put it on ida and find the key is 16

```
4   size_t v4; // rax
5   char v6[67]; // [rsp+Dh] [rbp-C3h] BYREF
6   char s1[64]; // [rsp+50h] [rbp-80h] BYREF
7   char s[56]; // [rsp+90h] [rbp-40h] BYREF
8   const char *v9; // [rsp+C8h] [rbp-8h]
9
10  v9 = "775a5051034d5644706002635f467d0570030558454b";
11  hexToString("775a5051034d5644706002635f467d0570030558454b", &v6[3]);
12  strcpy(v6, "16");
13  printf("Enter the correct passphrase: ");
14  __isoc99_scanf("%49s", s);
15  v3 = strlen(s);
16  xor_encrypt(s, s1, v6, v3);
17  v4 = strlen(s);
18  if ( !memcmp(s1, &v6[3], v4) )
19  {
20    printf("Correct! The flag is: ");
21    printf(s);
22  }
23  else
24  {
25    puts("Wrong passphrase!");
26  }
27  return 0;
28 }
```

The second flag is Flag2{grAV3UnpL3A54nt}

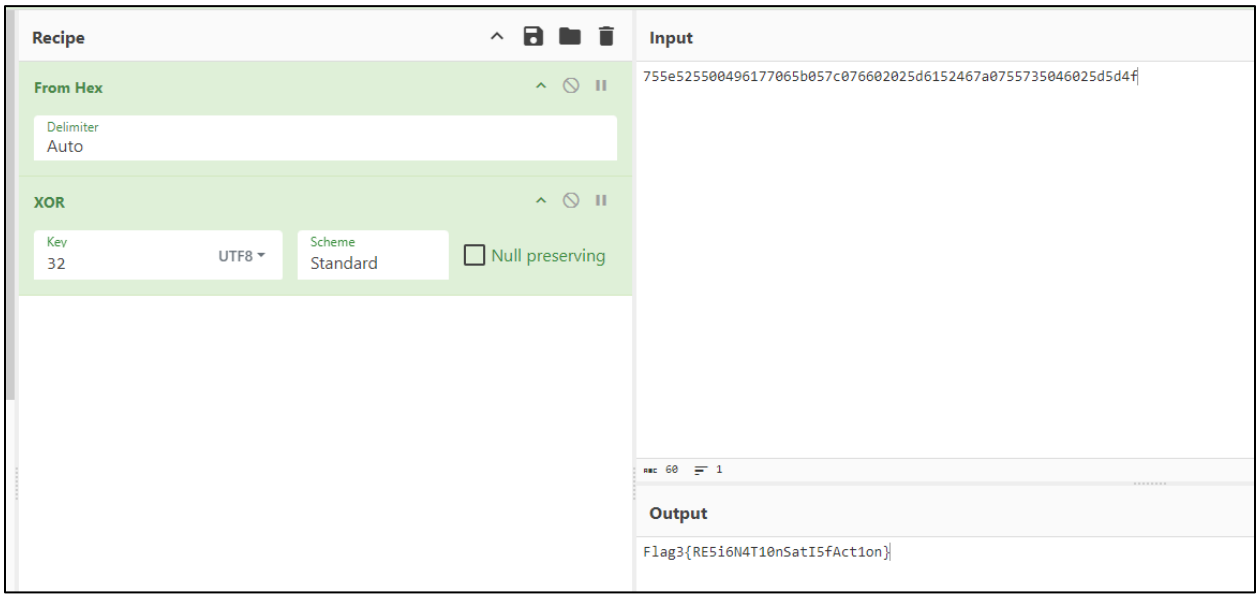


The final executable is wisdom, and we find the key is 32

```
int v10; // [xsp+4Ch] [xbp-B4h]
_BYTE v11[50]; // [xsp+52h] [xbp-AEh] BYREF
_BYTE v12[50]; // [xsp+84h] [xbp-7Ch] BYREF
char v13[50]; // [xsp+B6h] [xbp-4Ah] BYREF

v10 = 0;
v9 = argc;
v8 = argv;
context = objc_autoreleasePoolPush();
v7 = "755e525500496177065b057c076602025d6152467a0755735046025d5d4f";
hexToString("755e525500496177065b057c076602025d6152467a0755735046025d5d4f");
strcpy(v6, "32");
NSLog(&CFSTR("Enter the correct passphrase: ").isa);
scanf("%49s", v13);
strlen(v13);
xor_encrypt(v13);
v3 = strlen(v13);
if ( !memcmp(v12, v11, v3) )
  NSLog(&CFSTR("Correct! The flag is: %s").isa, v13);
else
  NSLog(&CFSTR("Wrong passphrase!").isa);
objc_autoreleasePoolPop(context);
return 0;
}
```

And the final flag is Flag3{RE5i6N4T10nSatI5fAct1on}



We follow the description and combine the flags, and we have the final flag as Intigriti{s7reaMCircUm574NcEgrAV3UnpL3A54ntRE5i6N4T10nSatI5fAct1on}