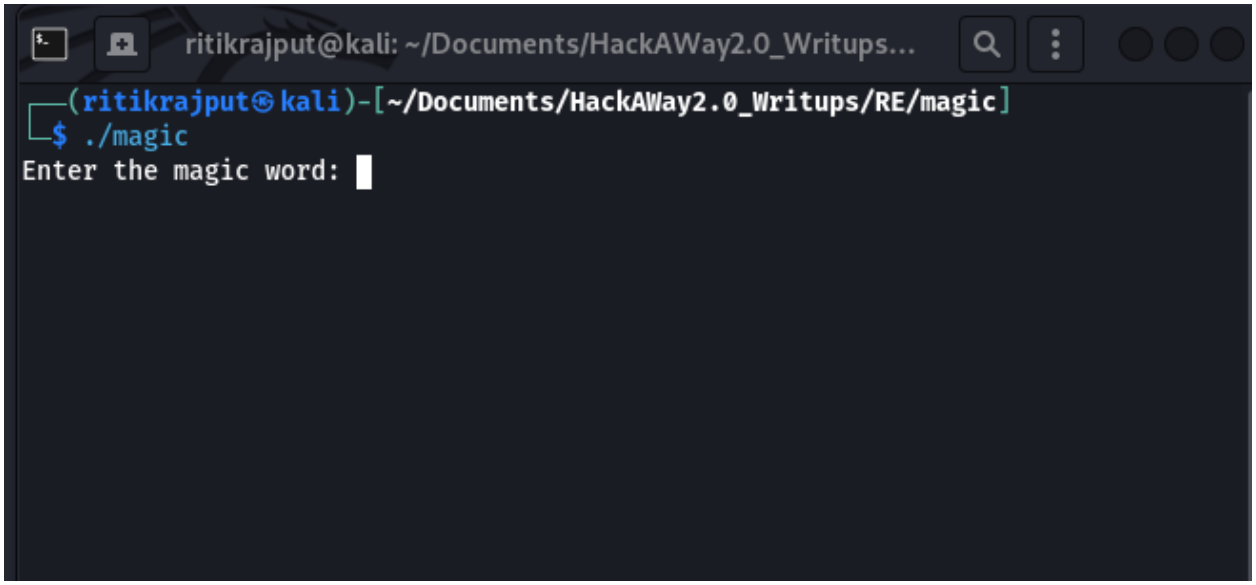
Reverse Engineering: Find the magic word to get the flag.

Overview:
A binary file name magic we have to enter the correct word to get the flag.



4.1 Load the binary

Load the binary into the Ghidra

4.2 Find relevant strings

- Open the **Defined Strings** view (right pane in your screenshots). Look for readable strings such as prompts: `"Enter the magic word:"` or garbled strings like `` `fcva`vpavg `` and `PPFYhK#ALD"...` that look encrypted/obfuscated.

```
00101258 48 8d 85        LEA        RAX=>flag,[RBP + -0x90]
         70 ff ff ff
0010125f 89 ce           MOV        ESI,ECX
00101261 48 89 c7        MOV        RDI,RAX
00101264 e8 10 ff        CALL       xor                                    void xor(char * c
         ff ff
                    magic.c:32 (20)
00101269 48 8d 05        LEA        RAX,[s_Enter_the_magic_word:_00102008]   = "Enter the magi
         98 0d 00 00
00101270 48 89 c7        MOV        RDI=>s_Enter_the_magic_word:_00102008,RAX  = "Enter the magi
00101273 b8 00 00        MOV        EAX,0x0
         00 00
00101278 e8 d3 fd        CALL       <EXTERNAL>::printf                     int printf(char *
         ff ff
                    magic.c:33 (24)
0010127d 48 8b 15        MOV        RDX,qword ptr [stdin]
         bc 2d 00 00
00101284 48 8d 45 a0     LEA        RAX=>input,[RBP + -0x60]
00101288 be 40 00        MOV        ESI,0x40
```

```
Defined Strings - 97 items
Filter:
Decompile: main - (magic)
/* WARNING: Unknown calling convention */

int main(void)

{
  int iVar1;
  char *pcVar2;
  size_t sVar3;
  char flag [25];
  char secret [12];
  char input [64];
  size_t n;
  int key;
  int flag_len;
  int secret_len;

  builtin_strncpy(secret,"`fcva`vpavg",0xc);
  builtin_strncpy(flag,"PPFYhK#ALD\"$[LA L\"@LU#]n",0x19);
  xor(secret,0x13,0xb);
  xor(flag,0x13,0x18);
  printf("Enter the magic word: ");
  pcVar2 = fgets(input,0x40,stdin);
  if (pcVar2 == (char *)0x0) {
    iVar1 = 1;
  }
  else {
    sVar3 = strlen(input);
    if ((sVar3 != 0) && (secret[sVar3 + 0xb] == '\n')) {
      secret[sVar3 + 0xb] = '\0';
    }
    iVar1 = strcmp(input,secret);
    if (iVar1 == 0) {
      printf("Correct! Here is your flag: %s\n",flag);
    }
    else {
      puts("Wrong input. Try again!");
    }
    iVar1 = 0;
  }
  return iVar1;
}
00101269   main   LEA RAX,[0x102008]
```

4.3 Inspect the decompiled `main`

- In the decompiled code you will see:
  - `builtin_strncpy(secret, "<garbled>", 0xc);`

- `builtin_strncpy(flag, "<garbled>", 0x19);`
- then `xor(secret, 0x13, 0xb);`
- and `xor(flag, 0x13, 0x18);`
- a `printf("Enter the magic word: ");` followed by `fgets(...)`, `strcmp(input, secret)` and `printf` of flag if matched.

Defined Strings - 97 items

Filter:

Decompile: main - (magic)

```
/* WARNING: Unknown calling convention */

int main(void)

{
  int iVar1;
  char *pcVar2;
  size_t sVar3;
  char flag [25];
  char secret [12];
  char input [64];
  size_t n;
  int key;
  int flag_len;
  int secret_len;

  builtin_strncpy(secret,"`fcva`vpavg",0xc);
  builtin_strncpy(flag,"PPFYhK#ALD\"$[LA L\"@LU#]n",0x19);
  xor(secret,0x13,0xb);
  xor(flag,0x13,0x18);
  printf("Enter the magic word: ");
  pcVar2 = fgets(input,0x40,stdin);
  if (pcVar2 == (char *)0x0) {
    iVar1 = 1;
  }
  else {
    sVar3 = strlen(input);
    if ((sVar3 != 0) && (secret[sVar3 + 0xb] == '\n')) {
      secret[sVar3 + 0xb] = '\0';
    }
    iVar1 = strcmp(input,secret);
    if (iVar1 == 0) {
      printf("Correct! Here is your flag: %s\n",flag);
    }
    else {
      puts("Wrong input. Try again!");
    }
    iVar1 = 0;
  }
  return iVar1;
}
```

This tells us the binary copies two encoded strings into memory, XORs them in place with key `0x13` for given lengths, then compares expected secret with user input. So the secret and flag are present but XOR-obfuscated with a single-byte key.

4.5 Extract the encoded text and key

Copy the XOR string and make a python script to decode it

```
RE > magic > script.py > ...
# Decoding the secret and flag by XORing with 0x13
enc_secret = "`fcva`vpavg"
enc_flag = 'PPFYhK#ALD"$[LA L"@LU#]n'
key = 0x13

dec_secret = ''.join(chr(ord(c) ^ key) for c in enc_secret)
dec_flag = ''.join(chr(ord(c) ^ key) for c in enc_flag)

print("Decoded secret (magic word):", dec_secret)
print("Decoded flag:", dec_flag)
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

/usr/local/bin/python3.11 /home/ritikrajput/Documents/HackAWay2.0_Writups/RE/magic/script.py
┌──(ritikrajput㉿kali)-[~/Documents/HackAWay2.0_Writups]
└─$ /usr/local/bin/python3.11 /home/ritikrajput/Documents/HackAWay2.0_Writups/RE/magic/script.py
Decoded secret (magic word): supersecret
Decoded flag: CCUJ{X0R_W17H_R3_1S_F0N}
```

Reveal secret code as well as flag

You can also use the secret code to test it in binary file

```
ritikrajput@kali: ~/Documents/HackAWay2.0_Writups...
┌──(ritikrajput㉿kali)-[~/Documents/HackAWay2.0_Writups/RE/magic]
└─$ ./magic
Enter the magic word: supersecret
Correct! Here is your flag: CCUJ{X0R_W17H_R3_1S_F0N}

┌──(ritikrajput㉿kali)-[~/Documents/HackAWay2.0_Writups/RE/magic]
└─$
```

Flag: CCUJ{X0R_W17H_R3_1S_F0N}