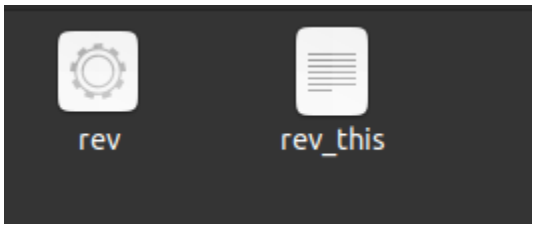
Mark Seliternikov

picoCTF - reverse_cipher [300 points]

לאתגר זה מקבלים 2 קבצים, אחד קובץ טקסט והשני קובץ בינארי.



כאשר בודקים את התוכן של קובץ הטקסט, ניתן לראות שיש בו את הדגל, אך הוא לא עובר את הבדיקה של האתגר (הדגל לא נכון).
דבר זה אומר שהדגל מוצפן.

```
mark@mark-ubuntu:~/Downloads$ cat rev_this
picoCTF{w1{1wq85jc=2i0<}mark@mark-ubuntu:~/Downloads$
```

הקובץ הבינארי הוא קובץ ELF שהוא Shared Object (כלומר הוא לא executable).

```
mark@mark-ubuntu:~/Downloads$ file rev
rev: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/l
d-linux-x86-64.so.2, for GNU/Linux 3.2.0, BuildID[sha1]=523d51973c11197605c76f84d4afb0fe9e59338c, not
 stripped
```

כדי להבין מה עושה הקובץ יש צורך ב-reverse engineering.
יש כמה דרכים לעשות זאת, החלטתי להשתמש ב-ghidra כמו באתגר קודם שפתרתי (picoCTF גם המליצו להשתמש בתוכנה לאתגר זה).

ככה נראית התוכנה לאחר שימוש ב-ghidra.

```c
void main(void)

{
  size_t sVar1;
  char local_58 [23];
  char local_41;
  int local_2c;
  FILE *local_28;
  FILE *local_20;
  uint local_14;
  int local_10;
  char local_9;

  local_20 = fopen("flag.txt","r");
  local_28 = fopen("rev_this","a");
  if (local_20 == (FILE *)0x0) {
    puts("No flag found, please make sure this is run on the server");
  }
  if (local_28 == (FILE *)0x0) {
    puts("please run this on the server");
  }
  sVar1 = fread(local_58,0x18,1,local_20);
  local_2c = (int)sVar1;
  if ((int)sVar1 < 1) {
                    /* WARNING: Subroutine does not return */
    exit(0);
  }
  local_10 = 0;
  while (local_10 < 8) {
    local_9 = local_58[local_10];
    fputc((int)local_9,local_28);
    local_10 = local_10 + 1;
  }
  local_14 = 8;
  while ((int)local_14 < 0x17) {
    if ((local_14 & 1) == 0) {
      local_9 = local_58[(int)local_14] + '\x05';
    }
    else {
      local_9 = local_58[(int)local_14] + -2;
    }
    fputc((int)local_9,local_28);
    local_14 = local_14 + 1;
  }
  local_9 = local_41;
  fputc((int)local_41,local_28);
  fclose(local_28);
  fclose(local_20);
  return;
}
```

```
local_10 = 0;
while (local_10 < 8) {
  local_9 = local_58[local_10];
  fputc((int)local_9,local_28);
  local_10 = local_10 + 1;
}
```

ניתן לראות כי ה-8 התווים הראשונים נכתבים איך שהם ללא שינוי.
הגיוני כי הדגל המוצפן מתחיל ב-picoCTF{.

```
if ((local_14 & 1) == 0) {
  local_9 = local_58[(int)local_14] + '\x05';
}
```

לאחר מכן יש לולאה שמטפלת בשאר הדגל (חוץ מהתיו האחרון).
ניתן לראות כי כל פעם שה-local_14, שהוא מתפקד כמו 'i' בלולאת for. הוא & 1 == 0. אז התוכנה מעלה את ערך ה-ASCII ב-5.

& - זה בעצם bitwise and, האופרטור בודק 2 ערכים ביט מול ביט ואם שני הביטים הם 1 אז הביט של התוצאה באותו סידור יהיה גם 1, אחרת יהיה 0.

```
else {
  local_9 = local_58[(int)local_14] + -2;
}
```

אחרת התוכנה פשוט מחסירה 2 מערך ה-ASCII.

לאחר מכן התוכנה כותבת את התיו לקובץ הטקסט לפי ה-if וה-else.

```
flag = "picoCTF{w1{1wq85jc=2i0<}"

for i in range(8):
    print(flag[i], end="")

for i in range(8,23):

    char = ord(flag[i])

    if i & 1 == 0:
        print(chr(char - 5), end="")

    else:
        print(chr(char + 2), end="")

print(flag[23])
```

אין סיבה לפתור את זה ידנית אז כתבתי סקריפט קצר בפייתון שיכתוב לי הדגל הלא מוצפן.

כל מה שהסקריפט עושה זה פעולה הפוכה של התכון הכתוב בתוך ה-{ }.

```
mark@mark-ubuntu:~/Downloads$ python3 solver.py
picoCTF{r3v3rs37ee84d27}
mark@mark-ubuntu:~/Downloads$
```

הנה הדגל :)