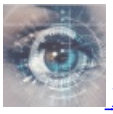
XCTF logmein

原创

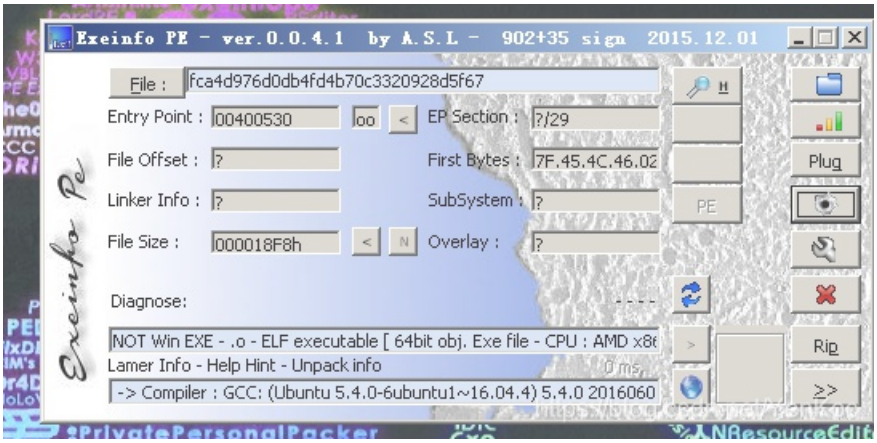
YenKoc 于 2020-01-13 20:54:21 发布 563 收藏

分类专栏： XCTF

本文链接：https://blog.csdn.net/YenKoc/article/details/103963881

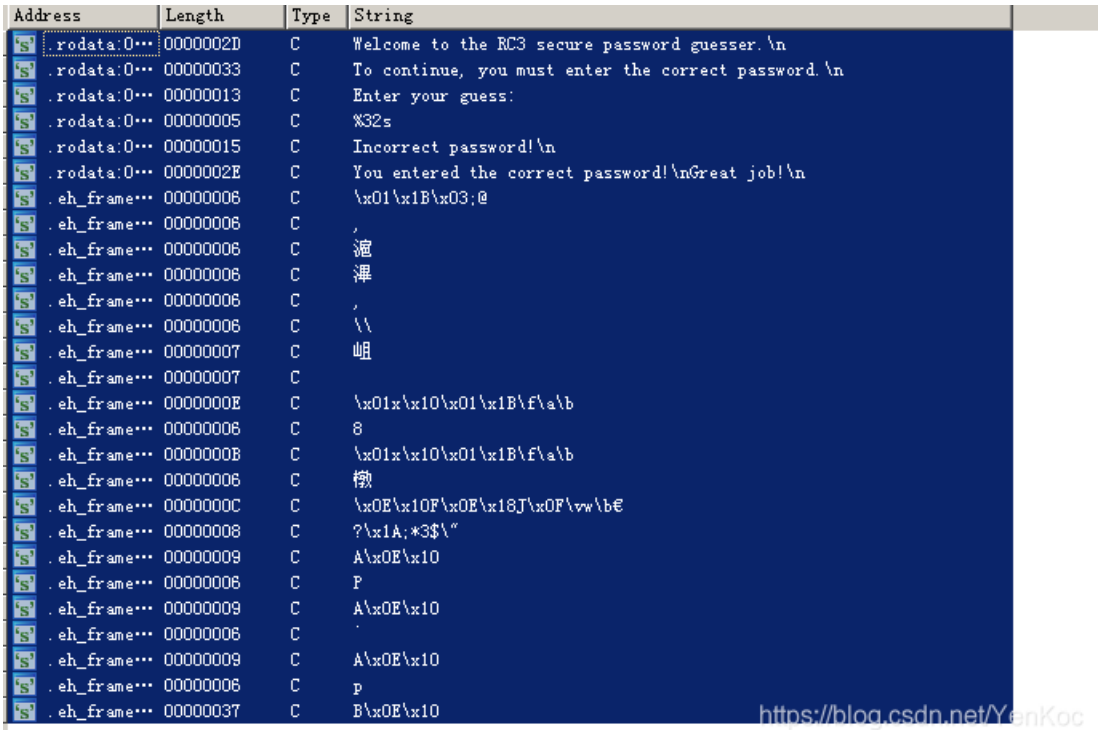


一.查壳



发现是64位的Linux文件（ELF可以看出是linux的文件）

二.拖入**ida64**，静态分析



```
ext:0000000000400630                push    rbp
ext:0000000000400631                mov     rbp, rsp
ext:0000000000400634                sub     rsp, 90h
ext:000000000040063B                mov     rdi, offset format ; "Welcome to the RC3 secure password gues"...
ext:0000000000400645                mov     [rbp+var_4], 0
ext:000000000040064C                mov     rax, ds:qword_4008B0
ext:0000000000400654                mov     qword ptr [rbp+var_20], rax
ext:0000000000400658                mov     rax, ds:qword_4008B8
ext:0000000000400660                mov     [rbp+var_18], rax
ext:0000000000400664                mov     cx, ds:word_4008C0
ext:000000000040066C                mov     [rbp+var_10], cx
ext:0000000000400670                mov     rax, ds:qword_4008D0
ext:0000000000400678                mov     [rbp+var_28], rax
ext:000000000040067C                mov     [rbp+var_2C], 7
ext:0000000000400683                mov     al, 0
ext:0000000000400685                call    _printf
ext:000000000040068A                mov     rdi, offset aToContinueYouM ; "To continue, you must enter the correct"...
ext:0000000000400694                mov     [rbp+var_5C], eax
ext:0000000000400697                mov     al, 0
ext:0000000000400699                call    _printf
ext:000000000040069E                mov     rdi, offset aEnterYourGuess ; "Enter your guess: "
ext:00000000004006A8                mov     [rbp+var_60], eax
ext:00000000004006AB                mov     al, 0
ext:00000000004006AD                call    _printf
ext:00000000004006B2                mov     rdi, offset a32s ; "%32s"
ext:00000000004006BC                lea     rsi, [rbp+s]
ext:00000000004006C0                mov     [rbp+var_64], eax
ext:00000000004006C3                mov     al, 0
```



```
1  void __fastcall __noreturn main(__int64 a1, char **a2, char **a3)
2  {
3    size_t v3; // rsi@1
4    int i; // [sp+3Ch] [bp-54h]@3
5    char s[36]; // [sp+40h] [bp-50h]@1
6    int v6; // [sp+64h] [bp-2Ch]@1
7    __int64 v7; // [sp+68h] [bp-28h]@1
8    char v8[8]; // [sp+70h] [bp-20h]@1
9    int v9; // [sp+8Ch] [bp-4h]@1
10
11   v9 = 0;
12   strcpy(v8, ":\"AL_RT^L*.?+6/46");
13   v7 = 28537194573619560LL;
14   v6 = 7;
15   printf("Welcome to the RC3 secure password guesser.\n", a2, a3);
16   printf("To continue, you must enter the correct password.\n");
17   printf("Enter your guess: ");
18   __isoc99_scanf("%32s", s);
19   v3 = strlen(s);
20   if ( v3 < strlen(v8) )
21     sub_4007C0();
22   for ( i = 0; i < strlen(s); ++i )
23   {
24     if ( i >= strlen(v8) )
25       sub_4007C0();
26     if ( s[i] != (*(&v7 + i % v6) ^ v8[i]) )
27       sub_4007C0();
28   }
29   sub_4007F0();
30 }
```



```
strcpy(v8, ":\"AL_RT^L*.?+6/46");
v7 = 28537194573619560LL;
```

注意这里两个坑：

1.strcpy是复制字符串的意思，前面定义的**v8**数组只有**8**个，但是后面的字符串是超过**8**个的，所以有可能这个**v8**定义是反编译错误的（算了一遍，确实是错的，不需要管这个数组，直接用原字符串就好了）

2._int 64 v7

```
11   v9 = 0;
12   strcpy(v8, ":\"AL_RT^L*.?+6/46");
13   v7 = 'ebmarah';
14   v6 = 7;
15   printf("Welcome to the RC3 secure password guesser.\n", a2, a3);
```

```
printf("To continue, you must enter the correct password.\n");
printf("Enter your guess: ");
__isoc99_scanf("%32s", s);
v3 = strlen(s);
if ( v3 < strlen(v8) )
  sub_4007C0();
for ( i = 0; i < strlen(s); ++i )
{
  if ( i >= strlen(v8) )
    sub_4007C0();
  if ( s[i] != (char)(*((_BYTE *)&v7 + i % v6) ^ v8[i]) )
    sub_4007C0();
}
sub_4007E0();
```


先将v7转换成字符串，这步个人理解是BYTE是字节型的指针，刚好对应的是一个字符，同时这个字符需要倒过来，因为是小端存储。

三。算法分析，写出脚本

```
v7="harambe"
v6=7
v8=":\"AL_RT^L*.?+6/46"
str=""
for i in range(0,len(v8)):
    str+=chr(ord((v7[i%7]))^ord(v8[i]))
print(str)
```

得到flag

```
RC3-2016-XORISGUD
```