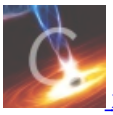
xctf BABYRE

原创

lcer. 于 2021-03-09 17:02:55 发布 59 收藏

分类专栏： xctf-wp 文章标签： wp

本文链接：https://blog.csdn.net/weixin_43655690/article/details/114589698



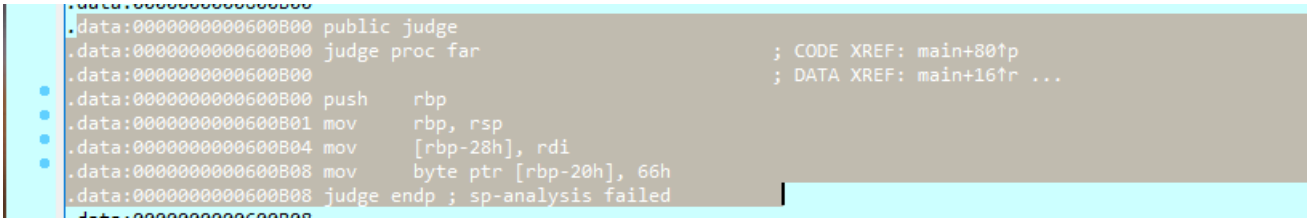
elf文件IDA打开后，程序很简单

```
1  int __cdecl main(int argc, const char **argv, const char **envp)
2  {
3    char s; // [rsp+0h] [rbp-20h]
4    int v5; // [rsp+18h] [rbp-8h]
5    int i; // [rsp+1Ch] [rbp-4h]
6
7    for ( i = 0; i <= 181; ++i )
8    {
9      envp = (const char **)(*((unsigned __int8 *)judge + i) ^ 0xCu);
10     *((_BYTE *)judge + i) ^= 0xCu;
11   }
12   printf("Please input flag:", argv, envp);
13   __isoc99_scanf("%20s", &s);
14   v5 = strlen(&s);
15   if ( v5 == 14 && (unsigned int)judge((__int64)&s) )
16     puts("Right!");
17   else
18     puts("Wrong!");
19   return 0;
20 }
```

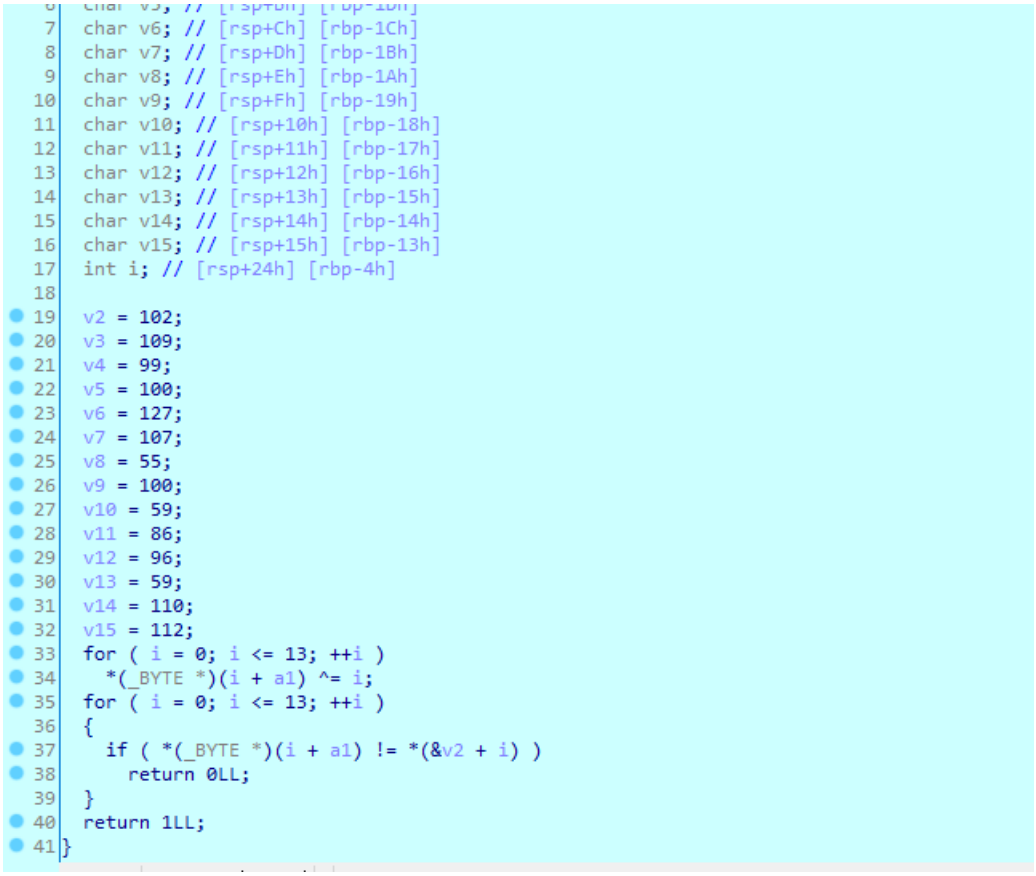
要求输入14个字符，然后进入judge中进行相应操作，那么judge是关键函数，可是我们不能反编译这个函数，可能是对函数进行了加密，我们发现7-11行对judge进行了一系列操作，应该是解密过程，我们直接说解决的办法吧

这里应该有两种方式可以解决，由于IDA python运用的还不是很熟练,这里只说一种方式吧。

我们再15行这里设置断点（也就是在运行judge之前设置断点），这时judge已经解密，我们我们需要做的就是重新定义这里。具体如下



在调试过程中（注意不能直接静态进行，因为一定要经过前面7-11行代码的解密），首先找到并选中judge这段，然后按d，将这段代码变为数据，然后按c在变为代码，然后右键选择create function(或者按p)，然后就可以f5反编译了



```
  char v6; // [rsp+Ch] [rbp-1Ch]
  char v7; // [rsp+Dh] [rbp-1Bh]
  char v8; // [rsp+Eh] [rbp-1Ah]
  char v9; // [rsp+Fh] [rbp-19h]
  char v10; // [rsp+10h] [rbp-18h]
  char v11; // [rsp+11h] [rbp-17h]
  char v12; // [rsp+12h] [rbp-16h]
  char v13; // [rsp+13h] [rbp-15h]
  char v14; // [rsp+14h] [rbp-14h]
  char v15; // [rsp+15h] [rbp-13h]
  int i; // [rsp+24h] [rbp-4h]

  v2 = 102;
  v3 = 109;
  v4 = 99;
  v5 = 100;
  v6 = 127;
  v7 = 107;
  v8 = 55;
  v9 = 100;
  v10 = 59;
  v11 = 86;
  v12 = 96;
  v13 = 59;
  v14 = 110;
  v15 = 112;
  for ( i = 0; i <= 13; ++i )
    *(_BYTE *)(i + a1) ^= i;
  for ( i = 0; i <= 13; ++i )
  {
    if ( *(_BYTE *)(i + a1) != *(&v2 + i) )
      return 0LL;
  }
  return 1LL;
}
```

得到这个之后便可以写脚本了

```
v2 = [102, 109, 99, 100, 127, 107, 55, 100, 59, 86, 96, 59, 110, 112]
flag = ''
for i in range(len(v2)):
    flag += chr(v2[i]^i)
print(flag)
```

得到flag{n1c3_j0b}