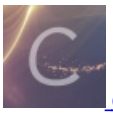
攻防世界Reverse进阶区-666-writeup

原创

y4ung 于 2020-11-13 16:38:22 发布 322 收藏 2

分类专栏： ctf 文章标签： ctf

本文链接：https://blog.csdn.net/qq_35056292/article/details/109677786



1. 介绍

本题是xctf攻防世界中Reverse的进阶区的题666

题目来源： 2019_UNCTF

2. 分析

```
$ file 666
666: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x8
6-64.so.2, for GNU/Linux 3.2.0, BuildID[sha1]=eaed290e5801a19d53e8597ac1a0499ec8bef798, not stripped
$ chmod +x 666
$ ./666
Please Input Key: 123
Your Length is Wrong
```

扔进IDA里看一下。main函数中，用户输入保存到v5。然后调用encode(&v5,&s)函数。

在判断中，首先比较用户输入的长度是否等于key，也就是18。然后比较了s和enflag是否相等。s应该是刚刚调用encode函数后得到的。enflag的值为`izwhroz""w"v.K".Ni`

其中，key的值为18，enflag的值为`izwhroz""w"v.K".Ni`：

```
.data:0000000000004060                 public enflag
.data:0000000000004060 ; char enflag[]
.data:0000000000004060 enflag          db 'izwhroz""w"v.K".Ni',0
.data:0000000000004060                                         ; DATA XREF: main+8F↑o
.data:0000000000004073                 align 20h
.data:0000000000004080                 public key
```

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char s; // [rsp+0h] [rbp-1E0h]
  char v5; // [rsp+F0h] [rbp-F0h]

  memset(&s, 0, 0x1EuLL);
  printf("Please Input Key: ", 0LL);
  __isoc99_scanf("%s", &v5);
  encode(&v5, &s);
  if ( strlen(&v5) == key )
  {
    if ( !strcmp(&s, enflag) )
      puts("You are Right");
    else
      puts("flag{This_1s_f4cker_flag}");
  }
  return 0;
}
```

现在来看看encode函数。
a1也就是main函数里用户的输入v5，a2是main函数里的s，是最后要比较的字符串。
首先呢，检查了一下用户输入的长度，必须为key。
然后在一个for循环中，每次取用户输入的三个字符，分别做相关的异或运算，再分别赋值给a2的对应的位置。

```
int __fastcall encode(const char *a1, __int64 a2)
{
  char v3[32]; // [rsp+10h] [rbp-70h]
  char v4[32]; // [rsp+30h] [rbp-50h]
  char v5[40]; // [rsp+50h] [rbp-30h]
  int v6; // [rsp+78h] [rbp-8h]
  int i; // [rsp+7Ch] [rbp-4h]

  i = 0;
  v6 = 0;
  if ( strlen(a1) != key )
    return puts("Your Length is Wrong");
  for ( i = 0; i < key; i += 3 )
  {
    v5[i] = key ^ (a1[i] + 6);
    v4[i + 1] = (a1[i + 1] - 6) ^ key;
    v3[i + 2] = a1[i + 2] ^ 6 ^ key;
    *(_BYTE *)(a2 + i) = v5[i];
    *(_BYTE *)(a2 + i + 1LL) = v4[i + 1];
    *(_BYTE *)(a2 + i + 2LL) = v3[i + 2];
  }
  return a2;
}
```

于是，我们可以写个脚本把用户的正确输入给算出来。最终得到flag为：`unctf{b66_6b6_66b}`

> 需要注意异或运算的优先级！

```
s = 'izwhroz""w"v.K".Ni'
key = 18
v5 = []

for i, each in enumerate(s):
    ascii_number = ord(each)
    if i % 3 == 0:
        v5.append((ascii_number ^ key) - 6)  # 异或的优先级比较低，需要用括号括起来
    elif i % 3 == 1:
        v5.append((ascii_number ^ key) + 6)
    elif i % 3 == 2:
        v5.append(ascii_number ^ key ^ 6)

v5_str = ""

for each in v5:
    v5_str += chr(each)

print(v5_str)
```

3. 总结

异或的运算优先级小于+和-，在写脚本的时候需要用小括号括起来。