BUUCTF-reverse-五道较简单题

原创

feng_2016 于 2020-05-14 09:34:10 发布 1477 收藏

[ACTF新生赛2020]usualCrypt

base64变种，后面还有个大小写转换

[GWCTF 2019]xxor

比较简单，就是Tea加密变种
Tea加密可以到ctfwiki学习，解密原理是逆向求解
解密最好用c语言写，python写有点麻烦

[GUET-CTF2019]re

逻辑很简单，就是比对
注意v[16]和v[17]位置调换了
少个v[6]爆破即可

[V&N2020 公开赛]strangeCpp

关注不合理的夹杂数据
通过welcome字符串中不合理的夹杂数据定位到关键函数

```
__int64 sub_140013580()
{
  __int64 *v0; // rdi
  signed __int64 i; // rcx
  __int64 result; // rax
  __int64 v3; // [rsp+0h] [rbp-20h]
  int v4; // [rsp+24h] [rbp+4h]
  int j; // [rsp+44h] [rbp+24h]
  __int64 v6; // [rsp+128h] [rbp+108h]

  v0 = &v3;
  for ( i = 82i64; i; --i )
  {
    *(_DWORD *)v0 = 0xCCCCCCCC;
    v0 = (__int64 *)((char *)v0 + 4);
  }
  v6 = -2i64;
  sub_1400110AA((__int64)&unk_140027033);
  result = sub_140011384((unsigned int)dword_140021190);
  v4 = result;
  if ( (_DWORD)result == 607052314 && dword_140021190 <= 0xDE02EF )
  {
    for ( j = 0; j < 17; ++j )
    {
      putchar((unsigned __int8)(dword_140021190 ^ byte_140021008[j]));
      result = (unsigned int)(j + 1);
    }
  }
  return result;
}
```

result的返回值是v7 = v7 * 291
但是607052314不整除291
说明291 * v7溢出了
于是用c++写解密脚本
鉴于0xDE02EF比较小，可以爆破

```
#include<iostream>
using namespace std;
int main()
{
 unsigned int v7 = 0;
 for (int i = 0; i <= 0xDE02EF; i++)
 {
  v7 = (i << 8) ^ (i >> 12);
  if (v7 * 291 == 607052314)
  {
   cout << i << endl;
  }
 }
}
```

接出来123456，剩下就简单了

[2019红帽杯]easyRE

顺着程序逻辑得到tips：flag前四个字符是flag
后面是个骗人的base64千层饼
在base64数据的交叉引用后面找到奇怪的数据

```
.data:00000000006CC090 off_6CC090      dq offset aVm0wd2vhuxhtwg
.data:00000000006CC090                                         ; DATA XREF: sub_4009C6+31B↑r
.data:00000000006CC090                                         ; "Vm0wd2VHUXhTWGhpUm1SWVYwZDRWVll3Wkc5WFJ"...
.data:00000000006CC098                 align 20h
.data:00000000006CC0A0 ; char byte_6CC0A0[3]
.data:00000000006CC0A0 byte_6CC0A0     db 40h                  ; DATA XREF: sub_400D35+95↑r
.data:00000000006CC0A0                                         ; sub_400D35+C1↑r
.data:00000000006CC0A1                 db  35h ; 5
.data:00000000006CC0A2                 db  20h
.data:00000000006CC0A3 byte_6CC0A3     db 56h                  ; DATA XREF: sub_400D35+A6↑r
.data:00000000006CC0A4                 db  5Dh ; ]
.data:00000000006CC0A5                 db  18h
.data:00000000006CC0A6                 db  22h ; "
.data:00000000006CC0A7                 db  45h ; E
.data:00000000006CC0A8                 db  17h
.data:00000000006CC0A9                 db  2Fh ; /
.data:00000000006CC0AA                 db  24h ; $
.data:00000000006CC0AB                 db  6Eh ; n
.data:00000000006CC0AC                 db  62h ; b
.data:00000000006CC0AD                 db  3Ch ; <
.data:00000000006CC0AE                 db  27h ; '
.data:00000000006CC0AF                 db  54h ; T
.data:00000000006CC0B0                 db  48h ; H
.data:00000000006CC0B1                 db  6Ch ; l
```


跟进函数找到关键代码，利用前面的tips求解
总结一句就是关注夹杂数据