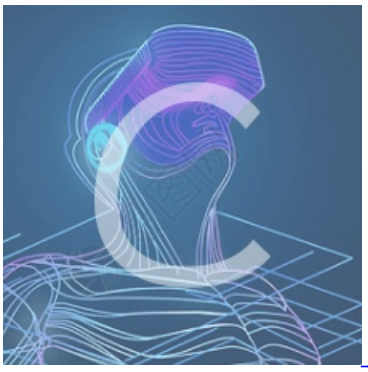
BUUCTF_luck_guy

原创

ZYen12138 于 2020-10-22 12:53:47 发布 290 收藏

分类专栏： # BUUCTF CTF

本文链接：https://blog.csdn.net/weixin_46009088/article/details/109219551



BUUCTF 同时被 2 个专栏收录

17 篇文章 2 订阅

订阅专栏

CTF

17 篇文章 0 订阅

订阅专栏

BUUCTF_luck_guy

照例用IDA打开，找到main函数，如下：

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  int v4; // [rsp+14h] [rbp-Ch]
  unsigned __int64 v5; // [rsp+18h] [rbp-8h]

  v5 = __readfsqword(0x28u);
  welcome();
  puts("_________________");
  puts("try to patch me and find flag");
  v4 = 0;
  puts("please input a lucky number");
  __isoc99_scanf("%d", &v4);
  patch_me(v4);
  puts("OK,see you again");
  return 0;
}
```


就patch_me函数看起来有点用.

```
1  int __fastcall patch_me(int a1)
2  {
3    int result; // eax
4
5    if ( a1 % 2 == 1 )
6      result = puts("just finished");
7    else
8      result = get_flag();
9    return result;
10 }
```

get_flag接着点进去.

```
unsigned __int64 get_flag()
{
  unsigned int v0; // eax
  char v1; // al
  signed int i; // [rsp+4h] [rbp-3Ch]
  signed int j; // [rsp+8h] [rbp-38h]
  __int64 s; // [rsp+10h] [rbp-30h]
  char v6; // [rsp+18h] [rbp-28h]
  unsigned __int64 v7; // [rsp+38h] [rbp-8h]

  v7 = __readfsqword(0x28u);
  v0 = time(0LL);
  srand(v0);
  for ( i = 0; i <= 4; ++i )
  {
    switch ( rand() % 200 )
    {
      case 1:
        puts("OK, it's flag:");
        memset(&s, 0, 0x28uLL);
        strcat((char *)&s, f1);
        strcat((char *)&s, &f2);
        printf("%s", &s);
        break;
      case 2:
        printf("Solar not like you");
        break;
      case 3:
        printf("Solar want a girlfriend");
        break;
      case 4:
        v6 = 0;
        s = ' fo`guci';
        strcat(&f2, (const char *)&s);
        break;
      case 5:
        for ( j = 0; j <= 7; ++j )
        {
          if ( j % 2 == 1 )
            v1 = *(&f2 + j) - 2;
          else
            v1 = *(&f2 + j) - 1;
          *(&f2 + j) = v1;
        }
        break;
      default:
        puts("emmm,you can't find flag 23333");
        break;
    }
  }
  return __readfsqword(0x28u) ^ v7;
}
```

前面都是初始化随机数的种子,重要的部分在后面的switch!

```
for ( i = 0; i <= 4; ++i )
{
  switch ( rand() % 200 )
  {
    case 1:
      puts("OK, it's flag:");
      memset(&s, 0, 0x28uLL);
      strcat((char *)&s, f1);
      strcat((char *)&s, &f2);
      printf("%s", &s);
      break;
    case 2:
      printf("Solar not like you");
      break;
    case 3:
      printf("Solar want a girlfriend");
      break;
    case 4:
      v6 = 0;
      s = '■fo`guci';
      strcat(&f2, (const char *)&s);
      break;
    case 5:
      for ( j = 0; j <= 7; ++j )
      {
        if ( j % 2 == 1 )
          v1 = *(&f2 + j) - 2;
        else
          v1 = *(&f2 + j) - 1;
        *(&f2 + j) = v1;
      }
      break;
    default:
      puts("emmm,you can't find flag 23333");
      break;
  }
}
```


我们看到它很佛系的把 switch 的对象设置为在一个0~199的随机数字，我们不管它，直接去看 case 的内容，除了 2 和 3 都是输出以外其他的应该都有用，看到 case1 先是输出 “OK, it’s flag:”，接下来把 s 清零，在接下来就是两个拼接，我们点进 f1 看第一个拼接的字符串，如下：

```
47 58 59 7B 64 6F+f1 db 'GXY{do_not_',0          ; DATA XREF: get_flag+9E↑o
```

在看到第二个拼接，点击 f2 去，如图：

```
f2 db ?                                ; DATA XREF: get_flag+AF↑o
```

应该是还没初始化，我们往下看，发现 case4 对 f2 进行了赋值，再看 case5 对 f2 进行了修改，至此，我们已经很明确的知道他的顺序是case4 -> case5 -> case1，直接将 case5 转化为 python 脚本输出，如下：

```python
s = 'icug`of ' #千万要注意IDA的字符是反向存储的！！！
str = list(s) #将s进行分割转化为元组
flag = 'GXY{do_not_'
for j in range(len(str)):
    if  j % 2 == 1:
        flag += chr(ord(str[j]) - 2)
    else:
        flag += chr(ord(str[j]) - 1)

print(flag)
```

运行得到结果！

```
GXY{do_not_hate_me}
```