i春秋试验场 CTF夺旗赛（第三季）RE 整型数列

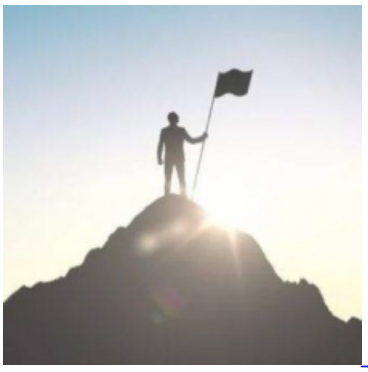
原创

A_dmins 于 2019-11-29 13:04:51 发布 325 收藏

分类专栏： CTF题 i春秋CTF 比赛CTF

本文链接：https://blog.csdn.net/qq_42967398/article/details/103308632

版权



CTF题 同时被 3 个专栏收录

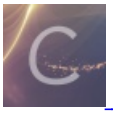
115 篇文章 11 订阅

订阅专栏

i春秋CTF

21 篇文章 1 订阅

订阅专栏



比赛CTF

25 篇文章 0 订阅

订阅专栏

i春秋试验场 CTF夺旗赛（第三季）RE 整型数列

emmmm，这次比赛比较简单，没啥好说的，，，

关键记录一下这个逆向题目，，，，，

ida直接打开进行静态分析：

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  __int64 v4; // [rsp+0h] [rbp-10h]
  signed int i; // [rsp+4h] [rbp-Ch]
  unsigned __int64 v6; // [rsp+8h] [rbp-8h]

  v6 = __readfsqword(0x28u);
  puts("This is a robot ,it can auto finish challenge, but maybe need some times...");
  LODWORD(v4) = 0;
  while ( (signed int)v4 <= 7 )
  {
    check();
    loadQuestion(v4);
    LODWORD(v4) = v4 + 1;
  }
  puts("OK, The Robot successfully finish it!");
  printf("And the flag is flag{", argv, v4, v6);
```

```
  for ( i = 0; i <= 6; ++i )
    printf("%d_", S[i]);
  printf("%d}\n", (unsigned int)dword_20203C);
  return 0;
}
```


看英语好像是说可以自动跑出来，需要一些时间，，，，
直接上kali进行运行，，，等了很久没出来，直接放弃
开始分析函数，进入check()，没有发现什么问题，进入loadQuestion()查看，发现这里是主要的函数：

```
unsigned __int64 __fastcall loadQuestion(unsigned int a1)
{
  int v1; // eax
  signed int v3; // [rsp+14h] [rbp-1Ch]
  signed int i; // [rsp+18h] [rbp-18h]
  signed int j; // [rsp+1Ch] [rbp-14h]
  __int64 v6; // [rsp+20h] [rbp-10h]
  unsigned __int64 v7; // [rsp+28h] [rbp-8h]

  v7 = __readfsqword(0x28u);
  printf("This is time %d, load questions...\n", a1);
  v6 = question(a1);
  v3 = 0;
  for ( i = 0; i <= 3; ++i )
  {
    for ( j = 0; j <= 99; ++j )
    {
      printf("testing number %d\n", (unsigned int)j);
      if ( func1(j) == v6 || func2(j) == v6 || func3(j) == v6 || func4(j) == v6 )
      {
        printf("\n==============>  Found number, is %d\n", (unsigned int)j);
        v3 = 1;
        v1 = key++;
        S[v1] = j;
        break;
      }
    }
    if ( v3 )
      break;
    printf("func%d is bad\n", (unsigned int)i);
  }
  if ( !v3 )
  {
    puts("system error, Exit!");
    exit(0);
  }
```


很显然嘛，有匹配，还有四个函数，进入查看貌似就是斐波拉契数组，，，，
这里我用了python进行转化，看得更清楚：

```python
def func1(x):
 if x <= 2:
  return x
 s = func1(x-1)
 return s + func1(x-2)

def func2(x):
 if x <= 3:
  return x
 s = func2(x-1)
 s1 = s + func2(x-2)
 return s1 + func2(x-3)

def func3(x):
 if x <= 4:
  return x
 s = func3(x-1)
 s1 = s + func3(x-2)
 s2 = s1 + func3(x-3)
 return s2 + func3(x-4)

def func4(x):
 if x <= 5:
  return x
 s = func4(x-1)
 s1 = s + func4(x-2)
 s2 = s1 + func4(x-3)
 s3 = s2 + func4(x-4)
 return s3 + func4(x-5)
```

四个函数都用了递归，查看一下v6等于的那个函数发现：

```
__int64 __fastcall question(unsigned int a1)
{
  printf("Questions%d! The %llu is who ?\n", a1, N[a1]);
  return N[a1];
}
```

```
.data:0000000000202040 N                dq 3736710778780434371, 14787220707439219840, 10155230078262535612
.data:0000000000202040                                          ; DATA XREF: question+27↑o
.data:0000000000202040                                          ; question+55↑o
.data:0000000000202040                  dq 6284562052556236687, 6463547837513153369, 4771310769738115795
.data:0000000000202040                  dq 1443656559118621430 7, 4379970031970910517
```

八个数字，还是大数，貌似逻辑已经清楚了，，，，，
就是看这八个数在这几个函数生成的结果（可以看成数组）中的第多少个（位），，，
显然，如果我们直接运行程序等待的话，那么我们可能会等一年，，，
毕竟第四个函数调用了多层递归，，，，
我们可以转换思路，直接先初始化，把数组给存起来，这样就快多了，，，，
利用python编写脚本：

```python
a = [3736710778780434371, 14787220707439219840,
 10155230078262535612,6284562052556236687,
 6463547837513153369, 4771310769738115795,
 14436565591186214307, 4379970031970910517]
```

```
a1 = [0] * 100
a1[0] = 1
a1[1] = 2
for i in range(2,100):
 a1[i] = a1[i-1] + a1[i-2]

print(a1)

a2 = [0] * 100
a2[0] = 1
a2[1] = 2
a2[2] = 3
for i in range(3,100):
 a2[i] = a2[i-1] + a2[i-2] + a2[i-3]

print(a2)

a3 = [0] * 100
a3[0] = 1
a3[1] = 2
a3[2] = 3
a3[3] = 4
for i in range(4,100):
 a3[i] = a3[i-1] + a3[i-2] + a3[i-3] +a3[i-4]

print(a3)

a4 = [0] * 100
a4[0] = 1
a4[1] = 2
a4[2] = 3
a4[3] = 4
a4[4] = 5
for i in range(5,100):
 a4[i] = a4[i-1] + a4[i-2] + a4[i-3] +a4[i-4] + a4[i-5]

print(a4)


for i in range(0,8):
 for k in range(0,100):
  if(a[i] == a1[k]):
   print(k + 1)
   break

for i in range(0,8):
 for k in range(0,100):
  if(a[i] == a2[k]):
   print(k + 1)
   break

for i in range(0,8):
 for k in range(0,100):
  if(a[i] == a3[k]):
   print(k + 1)
   break

for i in range(0,8):
 for k in range(0,100):
```

```
  if(a[i] == a4[k]):
   print(k + 1)
   break
```

运行无果，，，，
经过分析得知，python与C语言不一样，C语言最大的数为2^64，然而python却没有这个限制，，，，
直接编写C语言脚本进行解题，，，，，
最后解题脚本（C语言）：

```
#include<stdio.h>
using namespace std;

__int64 a1[100],a2[100],a3[100],a4[100];

int main(){
    __int64 a[8] = {3736710778780434371, 14787220707439219840,10155230078262535612,6284562052556236687,6463547837513153369, 4771310769738115795,14436565591186214307, 4379970031970910517};

    a1[0] = 1;
    a1[1] = 2;
    for(int i=2;i<=99;i++)
        a1[i] = a1[i-1] + a1[i-2];

    a2[0] = 1;
    a2[1] = 2;
    a2[2] = 3;
    for(int i=3;i<=99;i++)
        a2[i] = a2[i-1] + a2[i-2] + a2[i-3];

    a3[0] = 1;
    a3[1] = 2;
    a3[2] = 3;
    a3[3] = 4;
    for(int i=4;i<=99;i++)
        a3[i] = a3[i-1] + a3[i-2] + a3[i-3] +a3[i-4];

    a4[0] = 1;
    a4[1] = 2;
    a4[2] = 3;
    a4[3] = 4;
    a4[4] = 5;
    for(int i=5;i<=99;i++)
        a4[i] = a4[i-1] + a4[i-2] + a4[i-3] +a4[i-4] + a4[i-5];

    printf("flag{");
    for(int i=0;i<8;i++)
        for(int k=10;k<=99;k++)
            if(a[i] == a1[k]){
                printf("%d_",k+1);
                break;
            }
    for(int i=0;i<8;i++)
        for(int k=10;k<=99;k++)
            if(a[i] == a2[k]){
                printf("%d_",k+1);
                break;
            }
    for(int i=0;i<8;i++)
        for(int k=10;k<=99;k++)
```

```
            if(a[i] == a3[k]){
                printf("%d_",k+1);
                break;
            }
    int f = 1;
    for(int i=0;i<8;i++)
        for(int k=10;k<=99;k++)
            if(a[i] == a4[k] && f == 1){
                printf("%d_",k+1);
                f = 0;
                break;
            }else if(a[i] == a4[k]){
                printf("%d",k+1);
                break;
            }
    printf("}\n");
    return 0;
}
```

秒出结果：

```
flag{99_95_86_93_98_86_99_91}

Process returned 0 (0x0)   execution time : 1.953 s
Press any key to continue.
```

这题还要结合C语言的特性去搞，，，，
有点难顶~~