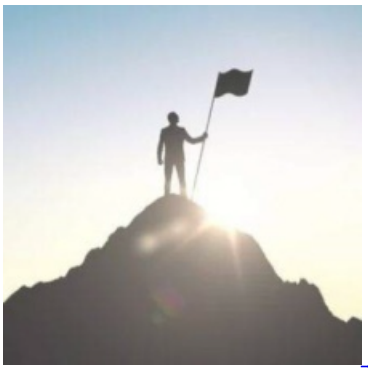
XCTF 进阶 RE notsequence

原创

A_dmins 于 2019-10-02 21:51:41 发布 191 收藏

分类专栏： CTF题 XCTF

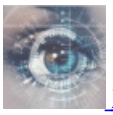
本文链接：https://blog.csdn.net/qq_42967398/article/details/101940013



CTF题 同时被 2 个专栏收录

115 篇文章 11 订阅

订阅专栏



XCTF

24 篇文章 0 订阅

订阅专栏

XCTF 进阶 RE notsequence

直接进入正题，下载文件，无壳，直接使用ida打开找到关键函数：

```
int __cdecl main()
{
  _DWORD *v0; // eax
  signed int v2; // [esp+14h] [ebp-Ch]
  _DWORD *v3; // [esp+1Ch] [ebp-4h]

  memset(&unk_8049BE0, 0, 0x4000u);
  puts("input raw_flag please:");
  v3 = &unk_8049BE0;
  do
  {
    v0 = v3;
    ++v3;
    scanf("%d", v0);
  }
  while ( *(v3 - 1) != 0 );
  v2 = sub_80486CD((int)&unk_8049BE0);
  if ( v2 == -1 )
  {
    printf("check1 not pass");
    system("pause");
  }
  if ( (unsigned __int8)sub_8048783((int)&unk_8049BE0, v2) ^ 1 )
  {
    printf("check2 not pass!");
    exit(0);
  }
  if ( v2 == 20 )
  {
    puts("Congratulations! fl4g is :\nRCTF{md5(/*what you input without space or \\n~*/)}");
    exit(0);
  }
  return 0;
}
```


程序过程很简单，有两个检查，一个if判断
首先查看第一个检查函数：

```
signed int __cdecl sub_80486CD(int a1)
{
  int j; // [esp+0h] [ebp-14h]
  int v3; // [esp+4h] [ebp-10h]
  int i; // [esp+8h] [ebp-Ch]
  int v5; // [esp+Ch] [ebp-8h]

  v5 = 0;
  for ( i = 0; i <= 1024 && *(_DWORD *)(4 * i + a1); i = v5 * (v5 + 1) / 2 )
  {
    v3 = 0;
    for ( j = 0; j <= v5; ++j )
      v3 += *(_DWORD *)(4 * (j + i) + a1);
    if ( 1 << v5 != v3 )
      return -1;
    ++v5;
  }
  return v5;
}
```


要求我们return v5，不能return -1
为了让我们的逻辑更加清楚，粗略写了一下C语言的代码：

```
int sub_80486CD(int a1){
 v5 = 0;
 for(int i = 0; i <= 1024 && a1[i]; i = v5 * (v5 + 1) / 2){
  v3 = 0;
  for(int j = 0;j <= v5; ++j)
   v3 += a1[i+j]
  if(1 << v5 != v3)
   return -1;
  ++v5;
 }
 return v5;
}
```

a1就是我们的输入，分析一下这个函数：
首先i=0,1,3,6,10,15,
v3表示从a1[i]开始加v5个的和：
a1[0]
a1[1] + a1[2]
a1[3] + a1[4] + a1[5]
a1[6] + a1[7] + a1[8] + a1[9]

v5左移一位等于v3即可！！！
可以列出几个例子：
a1[0] = 1
a1[1] + a1[2] = 2
a1[3] + a1[4] + a1[5] = 4
a1[6] + a1[7] + a1[8] + a1[9] = 8

接下来进入第二个函数：

```
1  signed int __cdecl sub_8048783(int a1, signed int a2)
2  {
3    int v3; // [esp+10h] [ebp-10h]
4    int v4; // [esp+14h] [ebp-Ch]
5    signed int i; // [esp+18h] [ebp-8h]
6    int v6; // [esp+1Ch] [ebp-4h]
7
8    v6 = 0;
9    for ( i = 1; i < a2; ++i )
10   {
11     v4 = 0;
12     v3 = i - 1;
13     if ( !*(_DWORD *)(4 * i + a1) )
14       return 0;
15     while ( a2 - 1 > v3 )
16     {
17       v4 += *(_DWORD *)(4 * (v3 * (v3 + 1) / 2 + v6) + a1);
18       ++v3;
19     }
20     if ( *(_DWORD *)(4 * (v3 * (v3 + 1) / 2 + i) + a1) != v4 )
21       return 0;
22     ++v6;
23   }
24   return 1;
25 }
```


这里也粗略写了一下C语言的代码：

```
int sub_8048783(int a1 , int a2){
 v6 = 0;
 for(int i = 1;i < a2; ++i ){
  v4 = 0;
  v3 = i - 1;
  if( !a1[i] )
   return 0;
  while(a2 - 1 > v3){
   v4 += a1[v3 * (v3 + 1) / 2 + v6];
   v3++;
  }
  if(a1[v3 * (v3 + 1) / 2 + i] != v4)
   return 0;
  ++v6;
 }
 return 1;
}
```

题目的要求是让我们返回1，也就是说要我们把这个函数执行完成，这里存在两个参数
第一个自然是我们的输入，第二个类似于上面举例子的层数
由于这里的a2没变，而且后面的if条件需要我们将这个类似的层数等于20

```
if ( v2 == 20 )
{
  puts("Congratulations! fl4g is :\nRCTF{md5(/*what you input without space or \\n~*/)}");
  exit(0);
}
```

所以我们现在就可以认为a2=20，接下来继续分析这个函数
首先a1里面的每个数都不能为0，由于程序有点复杂
我们可以将下标打印出来看一下结果，编写python：

```
v6 = v3 = 1   # 因为v3未变化前 v6=v3
a2 = 20    # 表示层数
i = 1
while a2 - 1 > v3:
 print(int(v3 * (v3 + 1) / 2 + v6))
 v3 += 1

print(v3 * (v3 + 1) / 2 + i)
```

由于20太大了，所以我就调小了层数，一点点尝试，得到结果：
a1[2] = a1[0] = 1
a1[4] = a1[0] + a1[1] = 2
a1[5] = a1[2] = 1
a1[7] = a1[0] + a1[1] + a1[3] = 3
a1[8] = a1[2] + a1[4] = 3
a1[11] = a1[0] + a1[1] + a1[3] + a1[6] = 4
a1[12] = a1[2] + a1[4] + a1[7] = 6
a1[13] = a1[5] + a1[8] = 4

再结合上一个函数得出的结论，可以推出前几层的数：
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
…… …… ……

这里就很显然了，学过C语言的应该都敲过，杨辉三角嘛~~
所以这里应该是让我们输入一个20层的杨辉三角！！
然后根据最后答案的提示：

```
puts("Congratulations! fl4g is :\nRCTF{md5(/*what you input without space or \\n~*/)}");
```

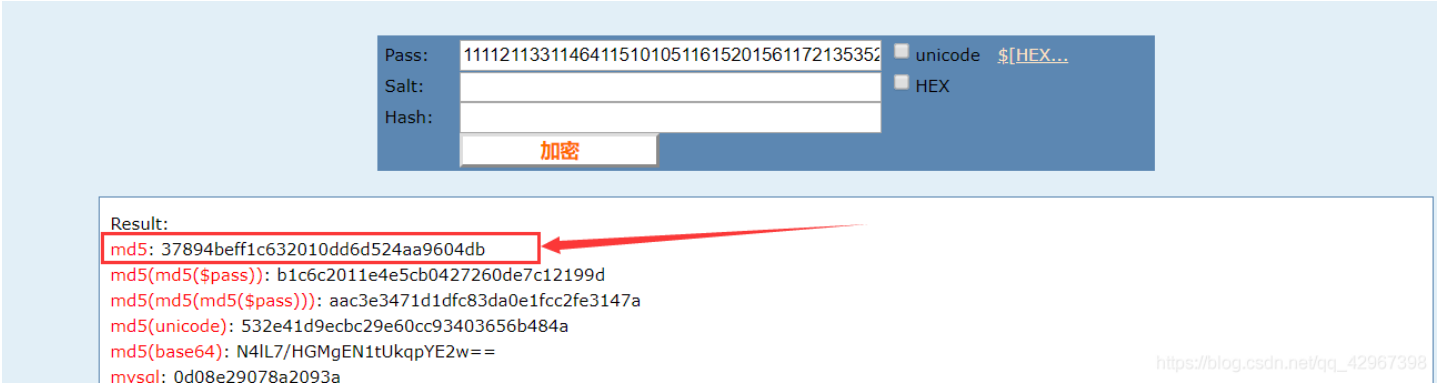
去掉空格和换行进行MD5值加密，裹上RCTF{}提交即可
接下来就是编写python脚本了，附上解题脚本：

```
a = []

for i in range(0,20):
 b = []
 for j in range(0,i + 1):
  b.append(1)
 a.append(b)

print(a)
for i in range(1,20):
 for j in range(1,i):
  a[i][j] = a[i - 1][j] + a[i - 1][j - 1]

print(a)

flag = ""
for i in range(0,len(a)):
 for j in range(0,len(a[i])):
  flag += str(a[i][j])

print(flag)
```

运行结果：

```
python 1.py
[[1], [1, 1], [1, 1, 1], [1, 1, 1, 1], [1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]]
[[1], [1, 1], [1, 2, 1], [1, 3, 3, 1], [1, 4, 6, 4, 1], [1, 5, 10, 10, 5, 1], [1, 6, 15, 20, 15, 6, 1], [1, 7, 21, 35, 35, 21, 7, 1], [1, 8, 28, 56, 70, 56, 28, 8, 1], [1, 9, 36, 84, 126, 126, 84, 36, 9, 1], [1, 10, 45, 120, 210, 252, 210, 120, 45, 10, 1], [1, 11, 55, 165, 330, 462, 462, 330, 165, 55, 11, 1], [1, 12, 66, 220, 495, 792, 924, 792, 495, 220, 66, 12, 1], [1, 13, 78, 286, 715, 1287, 1716, 1716, 1287, 715, 286, 78, 13, 1], [1, 14, 91, 364, 1001, 2002, 3003, 3432, 3003, 2002, 1001, 364, 91, 14, 1], [1, 15, 105, 455, 1365, 3003, 5005, 6435, 6435, 5005, 3003, 1365, 455, 105, 15, 1], [1, 16, 120, 560, 1820, 4368, 8008, 11440, 12870, 11440, 8008, 4368, 1820, 560, 120, 16, 1], [1, 17, 136, 680, 2380, 6188, 12376, 19448, 24310, 24310, 19448, 12376, 6188, 2380, 680, 136, 17, 1], [1, 18, 153, 816, 3060, 8568, 18564, 31824, 43758, 48620, 43758, 31824, 18564, 8568, 3060, 816, 153, 18, 1], [1, 19, 171, 969, 3876, 11628, 27132, 50388, 75582, 92378, 92378, 75582, 50388, 27132, 11628, 3876, 969, 171, 19, 1]]
11112113311464115101051161520156117213535217118285670562881193684126126843691110451202102522101204510111155165330462423
30165551111126622049579292479249522066121113782867151287171617161287715286781311149136410012002300334323003200210013649114111510545513653003500564356435500530031365455105151116120560182043688008114401287011440800843681820560120161117136680238061881237619448243102431019448123766188238068013617111815381630608568185643182443758486204375831824185648568306081615318111917196938761162827132503887558292378923787558250388271321162838769691711910
https://blog.csdn.net/qq_42967398
```

md5加密得到：



得到flag：RCTF{37894beff1c632010dd6d524aa9604db}