

Mountain climbing

转载

有人_295 于 2019-04-14 00:45:34 发布 1355 收藏

分类专栏: [Writeup](#) 文章标签: [Writeup](#)

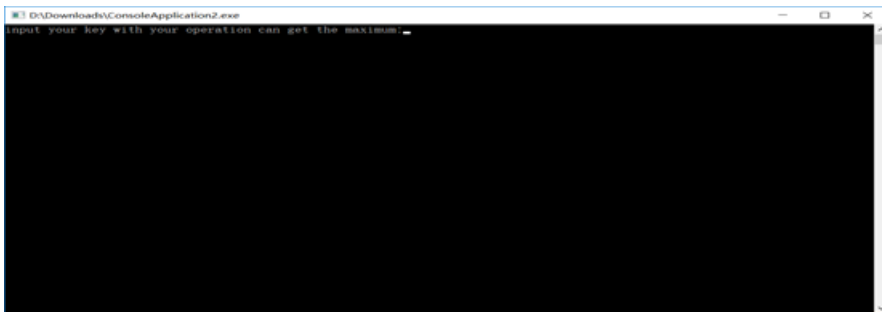


[Writeup](#) 专栏收录该内容

3 篇文章 0 订阅

订阅专栏

先是运行下程序，发现要输入一串东西。



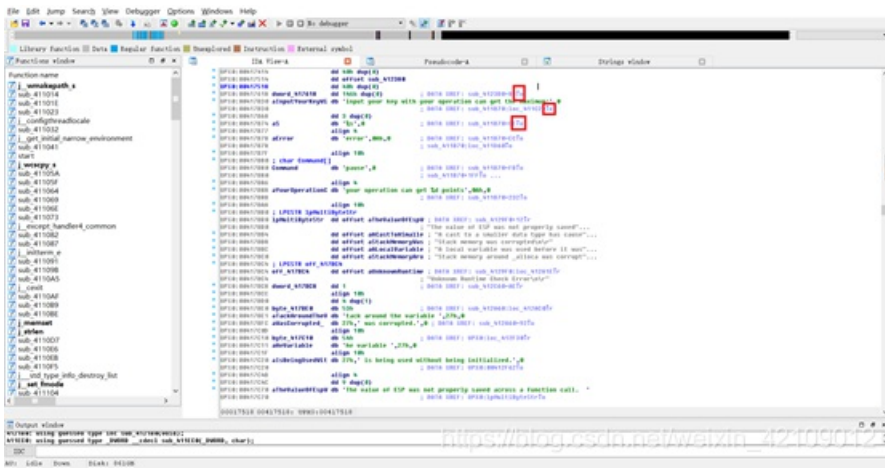
接着用查壳工具进行查壳，发现这个软件有个 UPX 的壳。



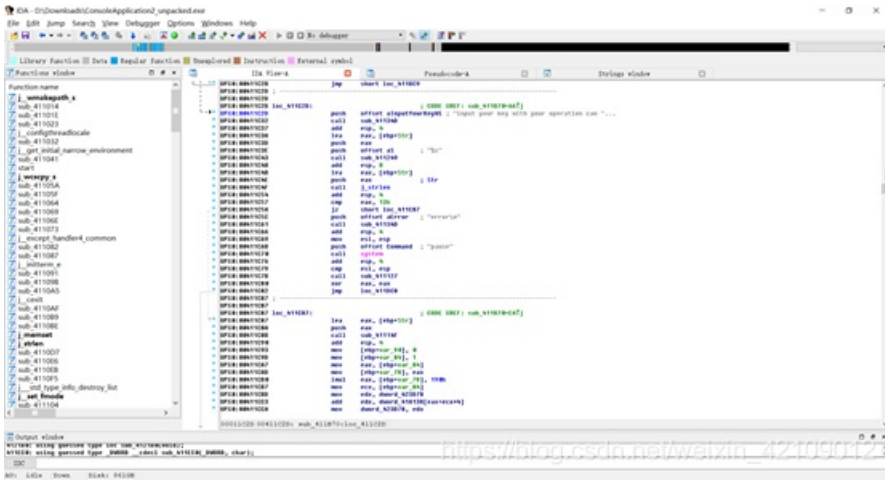
然后我用万能的脱壳工具进行脱壳，脱壳后得到能够使用IDA和OD打开的程序



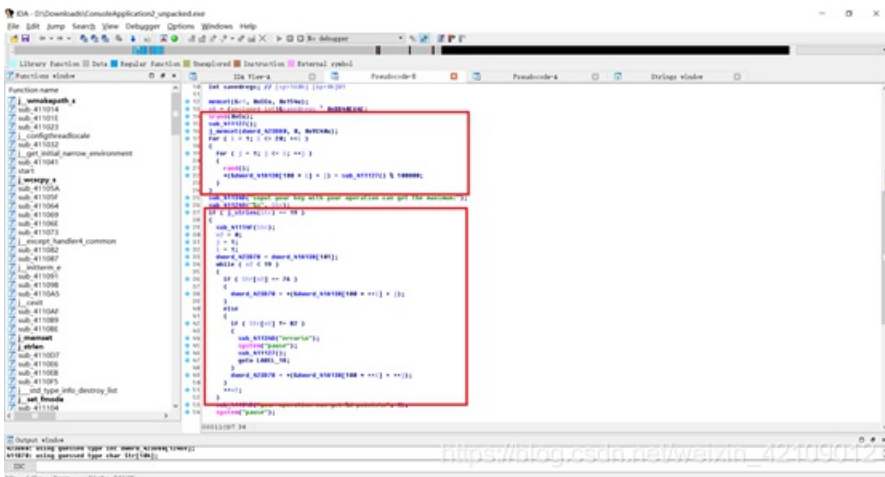
于是我用IDA打开，按F12查看字符串，点击开头的input开始查看，根据要输入的提示双击打开input字符串，汇编混杂，现在还不能使用反编译，点击那个箭头，跳转到全是汇编界面



然后点击Tab或F5，就可以得到反汇编



然后就是分析代码的时候了。

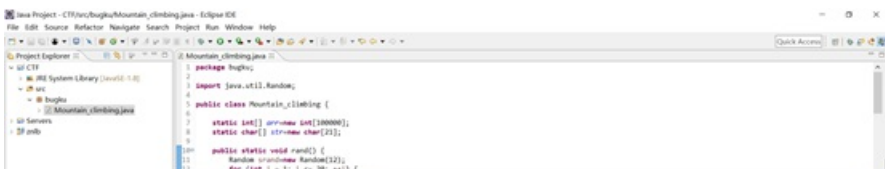


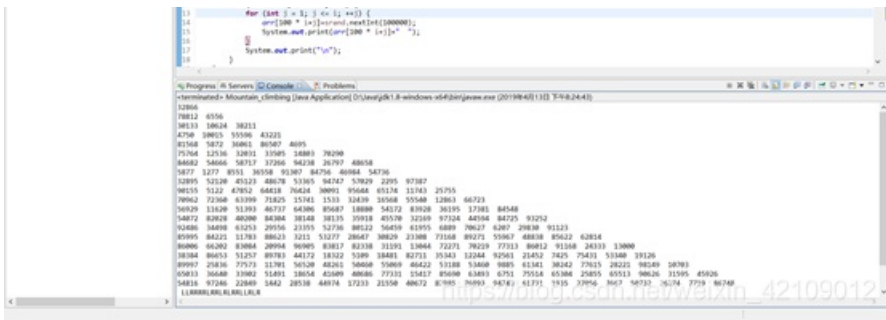
第一个红框表示生成随机数， **srand**代表随机数种子，**0xCu**表示**12**，生成一个**100000**以内的三角形随机数，一共**19**行。

第二个红框表示**76=L**，**82=R**，如果等于**76L**就直接向下移动一位，如果等于**82R**就向下和向右移动一位。

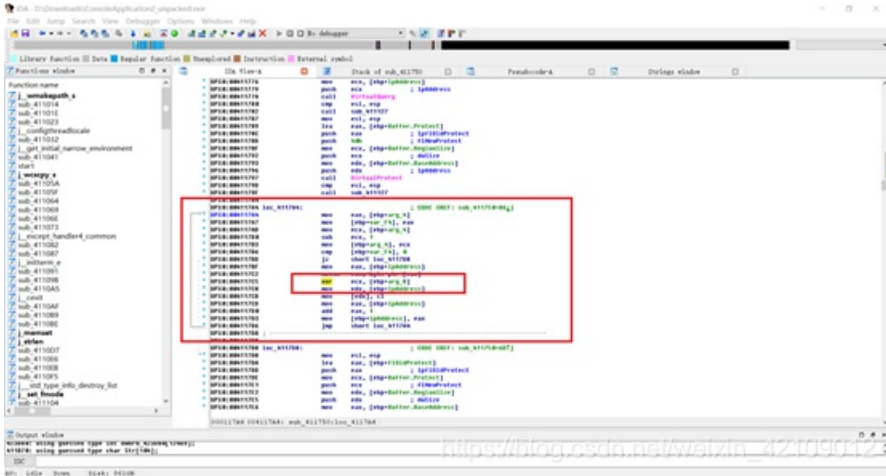
然后就进行反向编程，我用的是Java，编译后的结果如下，然后我就用结果去试，显而易见的错了，不可能这么简单的。。。

。





然后我就不知道怎么做，于是百度了一下，发现原来还要进行一步操作，可是我照着网上的IDA做就是不能找到想要的东西。
 。。。以下第一幅图是我的，没找打xor的0x4运算，而别人的有。。。。
 都找得到00411750即开始标志，push，不过我这个xor看不懂。



接着我又换一个软件OD用，使用网上的过程，还是不行。。。。
 都可以找到0041114F，然后跟进就始终找不到00411987 xor ecx 0x4。。。。。



```

00411950 8B45 BC mov eax,duword ptr ss:[ebp-0x44]
00411957 83E8 01 add eax,esi
00411962 8945 BC mov duword ptr ss:[ebp-0x44],eax
00411965 8370 BC 13 cmp duword ptr ss:[ebp-0x44],0x13
00411969 7D 29 jlt short ConsoleA.00411994
0041196B 8B45 BC mov eax,duword ptr ss:[ebp-0x44]
0041196E 25 01000000 and eax,0x00000001
00411973 79 05 jlt short ConsoleA.0041197A
00411975 48 dec eax
00411976 83C8 FE or eax,-0x2
00411979 40 inc eax
0041197A 85C0 test eax,eax
0041197C 74 14 jz short ConsoleA.00411992
0041197E 8B45 08 mov eax,duword ptr ss:[ebp+0x8]
00411981 8B45 BC add eax,duword ptr ss:[ebp-0x44]
00411984 0F8E 08 movsx ecx,byte ptr ds:[eax]
00411987 83F1 04 xor ecx,0x4
0041198A 8B55 08 mov edx,duword ptr ss:[ebp+0x8]
0041198D 8355 BC add edx,duword ptr ss:[ebp-0x44]
00411990 8B00 mov byte ptr ds:[edx],cl
00411992 EB C8 jlt short ConsoleA.0041199C
00411994 40 inc eax
00411995 48 dec eax
00411996 6A 04 push 0x4
00411998 8B45 BC mov eax,duword ptr ss:[ebp-0x44]

```

最后整不出来，就思考了一下思路，即在偶数位置需要与4进行异或xor运算

最后在整理一下思路

- 1、生成随机数，srand代表随机数种子，0xCu表示12，生成一个100000以内的三角形随机数，一共19行。
- 2、76=L，82=R，如果等于76L就直接向下移动一位，如果等于82R就向下和向右移动一位。反编译过来就是下一行左边大得到L，右边大得到R。
- 3、偶数位置需要与4进行异或xor运算

然后使用Java逆向编程出来即可

```

import java.util.Random;

public class Mountain_climbing {
    //全局变量
    static int[] arr=new int[100000];
    static char[] str=new char[21];
    //生成随机数
    public static void rand() {
        Random srand=new Random(12);
        for (int i = 1; i <= 20; ++i) {
            for (int j = 1; j <= i; ++j) {
                arr[100 * i+j]=srand.nextInt(100000);
                System.out.print(arr[100 * i+j]+" ");
            }
            System.out.print("\n");
        }
    }
    //进行位移L或R,
    public static void DFS() {
        int i=1;
        int j=1;
        int max=0;
        while(true) {
            int ii=i+1;
            int jj=j+1;
            if(arr[100 * ii+j]>arr[100 * ii + jj]) {
                str[i] = 'L';
                i++;
                max += arr[100 * i + j];
            }
            else {
                str[i] = 'R';
                i++;
                j++;
                max += arr[100 * i + j];
            }
        }
    }
}

```

```

    if (i == 20){
        str[i] = '\0';
        break;
    }
}
System.out.println(str);
System.out.println(max);
}
//改变偶数位置异或运算
public static void change()
{
    int i;
    for (i = 1; i <= 19; i++)
    {
        if (i % 2==0)
        {
            str[i] ^= 4;
        }
    }
    System.out.println(str);
}
//主函数
public static void main(String[] args) {
    rand();
    DFS();
    change();
}
}

```

如果你只看了前面，恭喜你只得到思路，因为TMD-----Java和C语言随机数种子一样，但生成的随机数不一样。。。。。

拿去试还是不得行，经过长时间的比对，发现没有错误。。。。。。。

最后发现我用Java和别人用C语言编的随机生成数不一样，可是Java中应该是这样写的，最后才发觉两个语言的随机算法不一样。。。。。。。。。。。

于是我最后使用C语言编程得到最后答案，即flag

```

#include <stdio.h>
#include <stdlib.h>

int main()
{
    srand(0xCu); // 随机数种子
    int arr[2020]; // 存放随机数的数组
    printf("76 = %c 82 = %c 76^4 = %c 82^4 = %c\n", 76, 82, 76 ^ 4, 82 ^ 4);
    for (int i = 0; i < 20; i++) // 总长度19
    {
        printf("%2d ", i);
        for (int j = 0; j <= i; j++)
        {
            arr[100 * i + j] = rand() % 100000; // 依次生成随机数
            printf("%5d ", arr[100 * i + j]);
        }
        printf("\n");
    }
    int all = 0; // 最终总和
    all += arr[100 * 0 + 0]; // 第一行第一个赋值给all
    char str[21];
    int i = 0, j = 0;
}

```

```

while (1)
{
printf("%d ", i);
int i_ = i + 1;
int j_ = j + 1;
if (arr[100 * i_ + j] > arr[100 * i_ + j_]) // 正下方的一个数大于其右边的数
{
str[i] = 'L';
if (i % 2 == 1)
{
str[i] ^= 4;
}
i++;
all += arr[100 * i + j];
}
else
{
str[i] = 'R';
if (i % 2 == 1)
{
str[i] ^= 4;
}
i++;
j++;
all += arr[100 * i + j];
}
if (i == 19)
{
str[i] = '\0';
break;
}
}
printf("\n 输入为: %s\n", str);
printf("maxnum = %d\n", all);

system("pause");
return 0;
}

```

The screenshot shows a Windows command prompt window titled "D:\学习文件\visual c++\CF\Debug\CF.exe". The program outputs a 10x19 grid of numbers. The numbers in the grid are:

16	1	82	*	R	70	4	*	R	82	4	*	V								
0	77																			
1	5408	6232																		
2	29032	1508	26150																	
3	12947	29926	11981	22371																
4	4078	29659	4665	2229	24699															
5	27370	3081	18012	25965	2064	26890														
6	21064	5225	11777	29853	2956	22459	3341													
7	21337	14755	5689	24855	4173	32304	292	3384												
8	15312	12952	1868	10888	15081	13463	32652	3409	28353											
9	26131	14588	12455	26295	25163	26049	8283	27502	15148	4945										
10	26170	1833	5196	9794	26694	2831	11993	2839	9970	27428	6884									
11	4616	30265	3752	32051	10445	9249	8092	28884	26285	8838	10784	6547								
12	7905	8373	19377	18302	27928	13669	25828	30502	28754	32537	2843	5401	10227							
13	22871	20993	8538	10009	6581	22716	12808	4653	24593	21533	9407	6840	30369	2330						
14	3	38084	22286	19327	13114	18100	15644	21728	17392	8396	27467	2002	3830	12684	1420					
15	29631	21820	9994	8319	10918	7978	24806	30607	17659	8764	3258	20719	6639	23336	23786	11048				
16	3544	31948	22	1591	644	25981	26918	31716	16427	15551	28157	7107	27297	24418	24384	32438	22224			
17	12282	12601	13235	21606	2516	13995	27889	16331	25292	28096	31589	28728	16697	4720	16655	22288	2291	20143		
18	16325	24537	16778	17119	18198	28537	11813	1490	21034	1978	6451	2174	24812	28772	5283	6429	15484	29353	5942	
19	7299	6961	32019	24731	29103	17887	17338	26840	13216	8789	12474	24299	19818	18218	14564	31409	5256	31930	26804	9736

Below the grid, the program outputs: "输入为: RVRVRHLVRLVRLVRLVRL" and "maxnum = 444740". A red box highlights the input string and the maxnum value.

https://blog.csdn.net/weixin_42109012

flag为: zscf{RVRVRHLVRLVRLVRLVRL}