2021强网杯ezmath

原创

Steins;G4te 于 2021-06-18 21:08:54 发布 193 收藏

强网杯ezmath wp学习复现

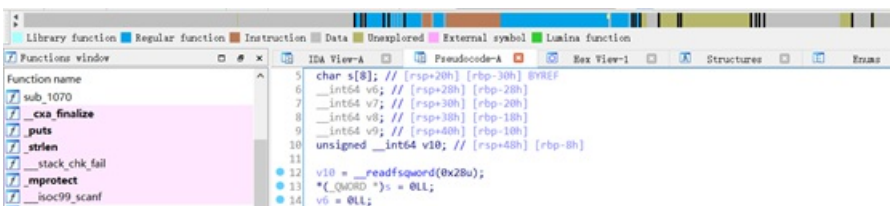
前言

博主是渣渣，通过写文章来分析与学习，最近参加强网杯，比较菜，打完之后学习复习一下，感谢来自https://www.yuque.com/fosusec/writeup/2021qwb#aldCs的wp，NU1L战队的wp，来自https://cdcq.github.io/2021/06/15/20210615a/的wp。

过程

拿到附件，放到die里面看下

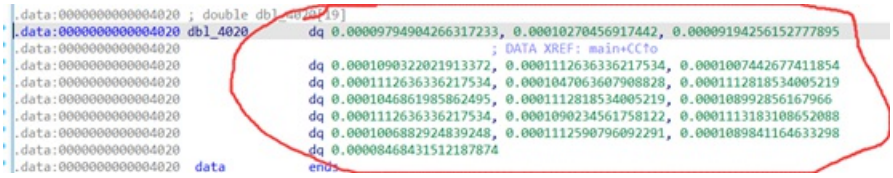
正常类型。无壳，64位，ELF文件
放到64位IDA中进行查看。



main函数如下

```
*(_QWORD *)s = 0LL;
v6 = 0LL;
v7 = 0LL;
v8 = 0LL;
v9 = 0LL;
__isoc99_scanf("%39s", s);
if ( strlen(s) == 38 )
{
  for ( i = 0; i <= 37; i += 2 )
  {
    if ( dbl_4020[i / 2] != sub_13F3(*(unsigned __int16 *)&s[i]) )
      goto LABEL_2;
  }
  puts("correct");
  result = 0LL;
}
else
{
LABEL_2:
  puts("wrong");
  result = 0LL;
}
```


大致了解到，flag字符串长38，要求对于flag偶数位索引（最后是36）经过一个函数运算得到值对应dbl_4020数组，都相同正确，相反则错误。



dbl_4020数组里面的值。这里搞不明白的同学后面可利用c代码。

```
double __fastcall sub_13F3(int a1)
{
  int i; // [rsp+8h] [rbp-Ch]
  double v3; // [rsp+Ch] [rbp-8h]

  v3 = unk_2010;
  for ( i = 8225; i < a1; ++i )
    v3 = 2.718281828459045 - (double)i * v3;
  return v3;
}
```

经过该函数运算。
可以看到v3的值我们不知道。

```
  v3 = unk_2010;
```

直接在IDA中使用快捷键R转换一下，发现不行。
可知是double类型，点击查看。

```
.rodata:000000000000200D                db    0
.rodata:000000000000200E                db    0
.rodata:000000000000200F                db    0
.rodata:0000000000002010 unk_2010       db 0CAh                 ; DA
.rodata:0000000000002010                                        ; .c
.rodata:0000000000002011                db 0C3h
.rodata:0000000000002012                db  42h ; B
.rodata:0000000000002013                db 0ADh
.rodata:0000000000002014                db  69h ; i
```

我们再次试试r转换一下，发现可行。
转换如下，回到咱们的函数。

```
.rodata:000000000000200E                db    0
.rodata:000000000000200F                db    0
.rodata:0000000000002010 qword_2010     dq 3FC9DE69AD42C3CAh    ;
.rodata:0000000000002010                                        ;
.rodata:0000000000002018 unk_2018       db    0                 ;
.rodata:0000000000002018                                        ;
.rodata:0000000000002019                db    0
.rodata:000000000000201A                db    0
.rodata:000000000000201B                db    0
```

发现值有了变化,这里再说明一下，看大神们的wp说这个值在进行计算的时候发生了变化，我不是很明白，＞へ＜如果有明白的大腿，可以一起交流下，共同进步.【1】

```
  int i; // [rsp+8h] [rbp-Ch]
  double v3; // [rsp+Ch] [rbp-8h]

  v3 = 0.2021;
  for ( i = 8225; i < a1; ++i )
    v3 = 2.718281828459045 - (double)i * v3;
  return v3;
}
```

在上述的文件中，可知flag的偶数索引位就是a1，即可知这个函数的意思是从i=8225到a1（字符串对应的int值（16进制的ASCII码）具体详解看总结）。
我们开始验证一波，Ctrl+F5得到c代码

```
43  _QWORD qword_2018 = 0LL; // idb
44  void *funcs_15C9 = &loc_11C0; // weak
45  __int64 (__fastcall *off_3D90)() = &sub_1180; // we
46  void *off_4008 = &off_4008; // idb
47  double dbl_4020[19] =
48  {
49    0.00009794904266317233,
50    0.00010270456917442,
51    0.00009194256152777895,
52    0.0001090322021913372,
53    0.0001112636336217534,
54    0.0001007442677411854,
55    0.0001112636336217534,
56    0.0001047063607908828,
57    0.0001112818534005219,
58    0.0001046861985862495,
59    0.0001112818534005219,
60    0.000108992856167966,
61    0.0001112636336217534,
62    0.0001090234561758122,
63    0.0001113183108652088,
64    0.0001006882924839248,
65    0.0001112590796092291,
66    0.0001089841164633298,
67    0.00008468431512187874
68  }; // idb
69  char byte_40B8; // weak
```


简单明了的数组存值

```
//----- (00000000000013F3) ----------------------------------
double __fastcall sub_13F3(int a1)
{
  int i; // [rsp+8h] [rbp-Ch]
  double v3; // [rsp+Ch] [rbp-8h]

  v3 = 0.2021;
  for ( i = ' !'; i < a1; ++i )
    v3 = 2.718281828459045 - (double)i * v3;
  return v3;
}

//----- (0000000000001449) ----------------------------------
```

主函数如下

```
//----- (0000000000001449) ----------------------------------------------------
__int64 __fastcall main(int a1, char **a2, char **a3)
{
  __int64 result; // rax
  int i; // [rsp+Ch] [rbp-44h]
  char s[8]; // [rsp+20h] [rbp-30h] BYREF
  __int64 v6; // [rsp+28h] [rbp-28h]
  __int64 v7; // [rsp+30h] [rbp-20h]
  __int64 v8; // [rsp+38h] [rbp-18h]
  __int64 v9; // [rsp+40h] [rbp-10h]
  unsigned __int64 v10; // [rsp+48h] [rbp-8h]

  v10 = __readfsqword(0x28u);
  *(_QWORD *)s = 0LL;
  v6 = 0LL;
  v7 = 0LL;
  v8 = 0LL;
  v9 = 0LL;
  __isoc99_scanf("%39s", s);
  if ( strlen(s) == 38 )
  {
    for ( i = 0; i <= 37; i += 2 )
    {
      if ( dbl_4020[i / 2] != sub_13F3(*(unsigned __int16 *)&s[i]) )
        goto LABEL_2;
    }
    puts("correct");
    result = 0LL;
  }
  else
  {
LABEL_2:
    puts("wrong");
    result = 0LL;
  }
  return result;
}
// 10D0: using guessed type __int64 __isoc99_scanf(const char *, ...);
```


我们先来拿第一个数来试一试，如果出现flag之类的东西，说明我们方向是对的。

数组第一个数字是0.00009794904266317233。

根据函数，编写一个python脚本，来进行验证。

根据函数，编写一个python脚本，来进行验证。

```
import time
v3=0.2021
i=8225#' !'
while v3!=0.00009794904266317233:
    v3=2.718281828459045 - i * v3
    i+=1
    print('我已经循环了%d次了 我是个废物'%(i-8225))
print(i)
```

阿巴，发现不是那么回事，都100w次了都没出来
于是进行调试一下

```
import time
v3=0.2021
i=8225#' !'
while v3!=0.00009794904266317233:
    v3=2.718281828459045 - i * v3
    i+=1
    print('我已经循环了%d次了 我是个废物'%(i-8225))
    print(v3)
    time.sleep(0.1)
print(i)
```

```
-3.491234285328703e+34
我已经循环了10次了 我是个废物
2.874682310539654e+38
我已经循环了11次了 我是个废物
-2.3673008827294053e+42
我已经循环了12次了 我是个废物
1.949709007015938e+46
我已经循环了13次了 我是个废物
-1.6059753090790283e+50
我已经循环了14次了 我是个废物
1.3230024596193035e+54
我已经循环了15次了 我是个废物
-1.0900217264803441e+58
我已经循环了16次了 我是个废物
8.981779026198036e+61
我已经循环了17次了 我是个废物
```

渐渐的发现，数越来越大，一正一负，以至于到了大概80左右都inf了

```
1.4692893841397714e+289
我已经循环了75次了 我是个废物
-1.2193632598975963e+293
我已经循环了76次了 我是个废物
1.0120715057150049e+297
我已经循环了77次了 我是个废物
-8.401205568940256e+300
我已经循环了78次了 我是个废物
6.9746808633342e+304
我已经循环了79次了 我是个废物
-inf
我已经循环了80次了 我是个废物
inf
我已经循环了81次了 我是个废物
-inf
我已经循环了82次了 我是个废物
```

思路不对，应该是小数他在计算机中损失精度的问题。进行调整

思路1

没有办法，我们看签到题的格式是flag{}形式
思考前两个字应该是fl。

```
In [51]: hex(ord('f'))
Out[51]: '0x66'

In [52]: hex(ord('l'))
Out[52]: '0x6c'
```

为0x666c

```
53]: '666c'.unhex()
53]: b'fl'
```

```
54]: int(0x666c)
54]: 26220
```

```
In [7]: '6c66'.unhex()
Out[7]: b'lf'
```

```
In [5]: int(0x6c66)
Out[5]: 27750
```

我们知道有e的参与，于是我们大胆尝试

```
In [9]: 2.718281828459045/26620
Out[9]: 0.00010211426853715422

In [10]: 2.718281828459045/27750
Out[10]: 9.795610192645207e-05
```

```
res = [0.00009794904266317233, 0.00010270456917442, 0.00009194256152777895,
0.00010903220219133 72, 0.0001112636336217534, 0.0001007442677411854,
0.0001112636336217534, 0.0001047063607908828, 0.0001112818534005219,
 0.0001046861985862495, 0.0001112818534005219, 0.000108992856167966,
0.0001112636336217534, 0.0001090234561758122, 0.0001113183108652088,
0.0001006882924839248, 0.0001112590796092291, 0.0001089841164633298,
0.00008468431512187874]
flag = b''
```

我们发现，与flag有关系，有戏，可能是因为是小数浮点关系，所以我们大胆尝试反推

```
[14]: 2.718281828459045/0.00009794904266317233
t[14]: 27751.99996396786
```

第一个数的整数部分是27751，与咱们猜的27750差1
相同规律我们在试试第二个，如果没啥问题会出现ag什么的

```
In [15]: 2.718281828459045/0.00010270456917442
Out[15]: 26466.999962218535
```

```
In [18]: hex(26465)
Out[18]: '0x6761'

In [19]: '6761'.unhex()
Out[19]: b'ga'
```

没问题，说明我们的思路对着呢，简单进行处理
flag += hex(int(math.e/i)-1)[2:].unhex()[::-1]
没问题，得到flag

```
----------------
b'flag{saam_dim_gei_lei_jam_caa_sin_laa}'
```

脚本如下

```
import math
res = [0.00009794904266317233, 0.00010270456917442, 0.00009194256152777895,
0.00010903220219133 72, 0.00011126363362175 34, 0.00010074426774118 54,
0.00011126363362175 34, 0.00010470636079088 28, 0.00011128185340052 19,
 0.00010468619858624 95, 0.00011128185340052 19, 0.00010899285616796 6,
0.00011126363362175 34, 0.00010902345617581 22, 0.00011131831086520 88,
0.00010068829248392 48, 0.00011125907960922 91, 0.00010898411646332 98,
0.00008468431512187874]
flag = b''
for i in res:
 flag += hex(int(math.e/i)-1)[2:].unhex()[::-1]
print(flag)
print(len(flag))
```

长度也是对应的。

思路2

Math数学数学，我们再进行思考，在这里，循环本质仍是数学

我们发现了一个彩蛋函数。

```
1  double __fastcall sub_1301(double a1)
2  {
3    int i; // [rsp+Ch] [rbp-1Ch]
4    double v3; // [rsp+10h] [rbp-18h]
5    double v4; // [rsp+18h] [rbp-10h]
6    double v5; // [rsp+20h] [rbp-8h]
7
8    v3 = 1.0;
9    v4 = 1.0;
10   v5 = 1.0;
11   for ( i = 1; i <= 8225; ++i )
12   {
13     v3 = v3 * a1;
14     v5 = (double)i * v5;
15     v4 = v3 / v5 + v4;
16   }
17   return v3 * v4;
18 }
```


这个函数感觉是那么回事，应该是暗示。
进行分析是泰勒公式

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots, \quad -\infty < x < \infty$$

是8225阶麦克劳林公式，最后return 大概是$a1^{8225}e^{a1}$
对应着x^ne^x.正如https://cdcq.github.io/2021/06/15/20210615a/的wp所说。
但是函数2，3我都没有找到，如果有大神找到的话，请求帮助，互相进步【2】.
这里我对他的wp做一些解释。

$$\lim_{n\to\infty} n\int_0^1 x^n e^x dx = \lim_{n\to\infty} n\int_0^1 x^n \sum_{i=0}^{\infty}\frac{x^i}{i!}dx$$
$$= \lim_{n\to\infty} n\int_0^1 \sum_{i=0}^{\infty}\frac{x^{i+n}}{i!}dx = \lim_{n\to\infty} n\sum_{i=0}^{\infty}\frac{\int_0^1 x^{i+n}dx}{i!}$$
$$= \lim_{n\to\infty} n\sum_{i=0}^{\infty}\frac{1}{(i+n+1)i!} = \lim_{n\to\infty}\sum_{i=0}^{\infty}\frac{1}{(\frac{i+1}{n}+1)i!}$$
$$= \sum_{i=0}^{\infty}\frac{1}{i!} = e$$

$$\lim_{n\to\infty} n\int_0^1 x^n e^x dx = \lim_{n\to\infty} n\int_0^1 x^n \sum_{i=0}^{\infty}\frac{x^i}{i!}dx$$
$$= \lim_{n\to\infty} n\int_0^1 \sum_{i=0}^{\infty}\frac{x^{i+n}}{i!}dx = \lim_{n\to\infty} n\sum_{i=0}^{\infty}\frac{\int_0^1 x^{i+n}dx}{i!}$$
$$= \lim_{n\to\infty} n\sum_{i=0}^{\infty}\frac{1}{(i+n+1)i!} = \lim_{n\to\infty}\sum_{i=0}^{\infty}\frac{1}{(\frac{i+1}{n}+1)i!}$$
$$= \sum_{i=0}^{\infty}\frac{1}{i!} = e$$

这里是麦克劳林公式

1+1/2！+1/3！+……=e^x

$$\lim_{n\to\infty} n\int_0^1 x^n e^x dx = \lim_{n\to\infty} n\int_0^1 x^n \sum_{i=0}^{\infty}\frac{x^i}{i!}dx$$
$$= \lim_{n\to\infty} n\int_0^1 \sum_{i=0}^{\infty}\frac{x^{i+n}}{i!}dx = \lim_{n\to\infty} n\sum_{i=0}^{\infty}\frac{\int_0^1 x^{i+n}dx}{i!}$$
$$= \lim_{n\to\infty} n\sum_{i=0}^{\infty}\frac{1}{(i+n+1)i!} = \lim_{n\to\infty}\sum_{i=0}^{\infty}\frac{1}{(\frac{i+1}{n}+1)i!}$$
$$= \sum_{i=0}^{\infty}\frac{1}{i!} = e$$

这里是前者的积分=1/（i+n+1）*x^{i+n+1}从1到0计算.

总结

如有不对，请多指正，共同进步。

1、高数基础再次巩固，一些公式，在比赛中了解加深印象，增加知识。

2、存入的字符是倒着存的，小端存储，而且是十六进制。hex（），unhex()，数据在内存中的形式，一个字的话是0x6161->aa，而转换为数字的话就是int（0x6161）所对应的值，代表着数组索引对应的数字值。

3、一些脑洞，要勇敢的去试，结合flag{}特点与数学特点。