

Dark CTF 2020-Rev/c_maths-writeup

原创

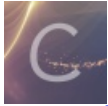
y4ung 于 2020-10-23 19:22:22 发布 158 收藏

分类专栏: [ctf](#) 文章标签: [ctf](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: https://blog.csdn.net/qq_35056292/article/details/109248210

版权



[ctf](#) 专栏收录该内容

35 篇文章 0 订阅

订阅专栏

1. 介绍

本题是dark ctf Reverse的第四题: [c_maths](#), 网址: <https://ctf.darkarmy.xyz/challs>

题目描述:

```
Clearly u know some C programming and a bit of maths right...? Note:Enclose the flag within darkCTF{} There are 3 parts to the flag.
nc cmaths.darkarmy.xyz 7001
```

题目中提到了 [There are 3 parts to the flag.](#), 有三个部分的flag。那么这道题应该将代码可以划分为三个部分来分析。

并且提示了 [nc cmaths.darkarmy.xyz 7001](#)。看来可能需要跟目标服务器进行交互才能拿flag。

2. 分析

```
$ file c_maths
c_maths: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux
-x86-64.so.2, BuildID[sha1]=d60c7a548fe6574179ed9b312304268dc221ceab, for GNU/Linux 3.2.0, not stripped
$ chmod +x c_maths
$ ./c_maths
123
```

输入了以后没反应啊。

2.1 静态分析

在main函数中, 可以看出用户需要输入三次。并且调用了system函数: [system\("cat small_chunk.txt"\);](#) 和 [system\("cat big_chunk.txt"\);](#); , 因此, 结合题目描述中提示的 [nc cmaths.darkarmy.xyz 7001](#), 题目的意思应该是让我们运行该nc命令以后, 输入三次, 然后服务器会从文件中返回flag的信息。

2.2 动态分析

2.2.1 用户的第一个输入

让我们继续看。字符串 [3h8t1e](#) 赋值给s, s经过变换, 然后赋值给v16, v16给s1, 接下来用户的输入必须要和s1相同才会继续往下运行。

因此, 第一个输入, 我们直接在本地debug, 把s1的值找到就行。

根据cutter中的地址和gdb调试时的地址差值，由本地cutter中的地址计算出gdb中第一个scanf前面的地址，打断点，动态调试。可以看出，用户的输入必须等于 `p1e8s3`。

```
37 int i; // [rsp+CCh] [rbp-14h]
38
39 setbuf(_bss_start, 0LL);
40 setbuf(stdin, 0LL);
41 setbuf(stderr, 0LL);
42 v3 = time(&timer);
43 srand(v3); // 根据时间设置随机数种子
44 time(&v19);
45 v4 = time(0LL);
46 srand(v4);
47 strcpy(s, "3h8t1e");
48 for ( i = 0; ; ++i )
49 {
50     v5 = i;
51     if ( v5 >= strlen(s) )
52         break;
53     if ( s[i] <= 96 || s[i] > 122 )
54     {
55         v16[i] = s[i];
56     }
57     else
58     {
59         v23 = s[i] + 11;
60         if ( v23 > 122 )
61         {
62             v22 = 122 - s[i];
63             v21 = v22 + 95;
64             v16[i] = v22 + 95;
65         }
66         else
67         {
68             v16[i] = v23;
69         }
70     }
71 }
72 for ( j = 0; v16[j]; ++j ) // 获取v16的长度
73 ;
74 v36 = j - 1;
75 for ( k = 0; k < j; ++k ) // v16[::-1]
76     s1[k] = v16[v36--];
77 s1[j] = 0;
78 __isoc99_scanf("%[^\n]*c", &s2);
79 v6 = strlen(s1);
80 if ( strncmp(s1, &s2, v6) ) // s1的值为p1e8s3
81     exit(1);
82
```

https://blog.csdn.net/qq_35056292

ok，现在用户的第一个输入找到了，我们执行程序，输入 `p1e8s3`，发现打印出一堆数字。

```
$ ./c_maths
p1e8s3
9801
1378
2999
1278
4140
7704
6351
1574
2549
```

2.2.2 用户的第二个输入

看一下IDA中的代码。由于这道题随机数是以当前时间（距离1970年那个时间的秒数）作为种子的，因此，生成的随机数是会变化的。在下图的第一个for循环里一共生成了9个随机数并打印出来。

直接往后面看，可以看到，用户的输入v12必须与v28相同。其中v28 <- v32 <- v8 <- v13, v29 <- v34 <- v7（随机数）。因此，为了得到v28的值，我们可以写个python脚本，将这部分的算法重写一遍，输入为生成的随机数，输出即为v28。

```

v13 = ""9527
4857
7110
6633
4008
2344
8538
1877
3069
"".split("\n")
v13 = [int(each) for each in v13[:-1]]
v34 = sum(v13)
v29 = v34 // 9

v32 = 0
for m in range(0, 9): # [0, 8]
    v8 = pow(v13[m]-v29, 2)
    v32 += v8

import math
v28 = int(math.sqrt(v32/9))
print("第二个输入:{}".format(v28))

```

```

v34 = 0;
for ( l = 0; l <= 8; ++l )
{
    v7 = randomfunc(100, 10000);
    v13[l] = v7;
    printf("%d\n", (unsigned int)v13[l]);
    v34 += v13[l];
}
v29 = v34 / 9;
v32 = 0;
for ( m = 0; m <= 8; ++m )
{
    v8 = pow((double)(v13[m] - v29), 2.0);
    v32 = (signed int)(v8 + (double)v32);
}
v28 = (signed int)sqrt((double)(v32 / 9));
__isoc99_scanf("%d", &v12);
if ( v28 != v12 )
{
    puts("Nothing for u here");
    exit(1);
}
system("cat small_chunk.txt");
putchar(10);

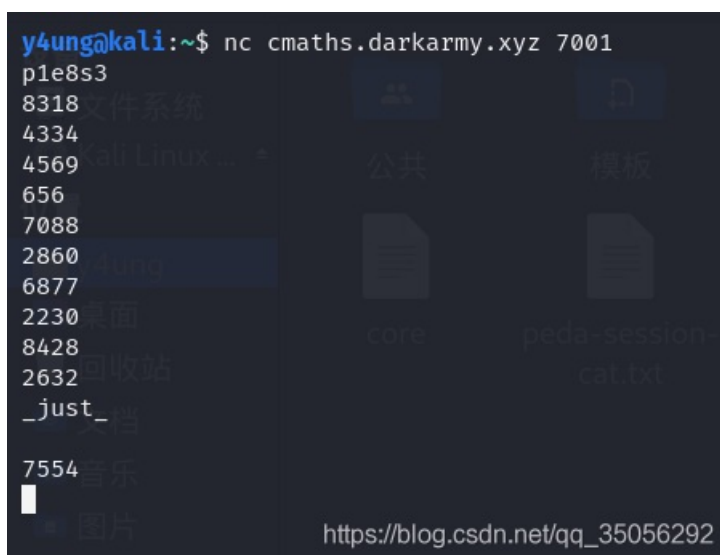
```

现在我们已经得到用户的第二个输入了。我们本地测试一下，发现打印出了 `cat: small_chunk.txt: 没有那个文件或目录` 和 `8644`。

```
$ ./c_maths p1e8s3
p1e8s3
9801
1378
2999
1278
4140
7704
6351
1574
2549
2902
cat: small_chunk.txt: 没有那个文件或目录

8644
```

没有small_chunk.txt应该是我们需要跟服务器去交互。让我们用题目描述里的命令去测试一下：发现small_chunk.txt的内容是 `_just_`，看来这是flag的第二部分，第一部分应该是 `p1e8s3`



```
y4ungakali:~$ nc cmaths.darkarmy.xyz 7001
p1e8s3
8318
4334
4569
656
7088
2860
6877
2230
8428
2632
_just_
7554
```

2.2.3 用户的第三个输入

但是第三个flag是啥呢。先分析下代码：

1. 用户的输入v10必须与v25相同
2. 并且解题的时间不能超过5s，这意味着这道题需要写脚本去跟目标服务器交互

```
v25 <- func3, v11 <- func2, v11, n <- v26 <- v27
```

v27为随机数，是for循环的边界：[100, v27-1]。在循环中：

1. 首先调用 `func1(n)`，赋值给v26。func1的作用是求n每一位数之和。

```

__int64 __fastcall func1(int a1)
{
    int v2; // [rsp+0h] [rbp-14h]
    unsigned int v3; // [rsp+10h] [rbp-4h]

    v2 = a1;
    v3 = 0;
    while ( v2 > 0 )
    {
        v3 += v2 % 10;
        v2 /= 10;
    }
    return v3;
}

```

2. 如果v26满足 `!(v26 & 1) && !(v26 % 3)`，则调用 `func2(&v11, n)`。v11初始时为0。

首先，创建了一个qword类型的数组v3，并分配了16字节。一个qword是8个字节，因此v3有两个元素。

接下来，将a2，也就是main函数中的n赋值给v3[0]。

1. 如果a1不为0的话，那么将a1的值赋值给v3[1]，然后将v3赋值给a1
2. 如果a1为0，那么将v3赋值给a1

```

_QWORD * __fastcall func2(_QWORD *a1, int a2)
{
    _QWORD *result; // rax
    _QWORD *v3; // [rsp+18h] [rbp-8h] v3是一个qword类型的数组，分配给v3 16字节
    // 一个qword为8字节，因此v3有两个元素

    v3 = malloc(0x10uLL); // 分配了16字节
    *(_DWORD *)v3 = a2; // dword为4字节
    if ( *a1 )
    {
        v3[1] = *a1;
        result = a1;
        *a1 = v3;
    }
    else
    {
        *a1 = v3;
        result = v3;
        v3[1] = 0LL;
    }
    return result;
}

```

在循环外，`v25 = func3(*(_QWORD *) (v11 + 8))`，也就是参数是v11[1]。跟进去func3看看。在函数中首先将a1赋值给v2，然后当v2不为空时，每次取v2[1]，当`!(v3 & 1)`时，`v4+=v2`。

看到这里我是有点懵逼的，a1就是v11嘛。不就是个一维数组吗？为什么在func3里要大费周章地用个循环去取v11[1]呢？

```

__int64 __fastcall func3(__int64 a1)
{
    __int64 v2; // [rsp+0h] [rbp-18h]
    char v3; // [rsp+10h] [rbp-8h]
    unsigned int v4; // [rsp+14h] [rbp-4h]

    v2 = a1;
    v4 = 0;
    v3 = 0;
    while ( v2 )
    {
        if ( !(v3 & 1) )
            v4 += *(_DWORD *)v2;
        v2 = *(_QWORD *)(v2 + 8);
        ++v3;
    }
    return v4;
}

```

```

v11 = 0LL;
v27 = randomfunc(5000, 10000);
printf("%d\n", v27);
v26 = 0;
for ( n = 100; n < (signed int)v27; ++n )
{
    v26 = func1(n);
    if ( !(v26 & 1) && !(v26 % 3) )
        func2(&v11, n);
}
v25 = func3(*(_QWORD *)(v11 + 8)); // 用户的第三个输入必须与v25相等，
// v11+8为v11[1]，因为qword为8字节
__isoc99_scanf("%d", &v10);
time(&v18);
v24 = (double)((signed int)v18 - (signed int)v19); // 解题时间必须在5s以内
if ( v25 != v10 || v24 >= 5.0 )
    puts("Nothing for u here");
else
    system("cat big_chunk.txt");
return 0;
}

```

https://blog.csdn.net/qq_35056292

决定用gdb调试看看，我关注的是func3，因此在这之前的部分我能跳过就跳过。

- 第一个输入是固定的，运行了以后直接复制进去即可。
 - 第二个输入由于涉及到生成的随机数，并且debug的时候我发现当你运行到printf后面的指令时，是看不到输出的随机数内容的。因此在运行到第二个scanf时，随便输入一个数字，然后在比较之前，先看一下要比较的内容：x/16xb \$rbp-0x3c 比如输出是 0b 25 00 00 ...，那么就设置rax的值为：set \$rax=0x250b。在这之后会调用system函数，一调用，gdb调试就结束了。所以在调用之前，将指令寄存器的值修改为system的下一条指令的地址：set \$rip=call system的下一条指令地址
- 下面为func2的一些调试过程：

```

$rbx : 0x2
$rcx : 0x6
$rdx : 0x69
$rsp : 0x00007fffffffdf0 → 0x0000038000000380
$rbp : 0x00007fffffff0c0 → 0x0000555555557f0 → <__libc_csu_init+0> push r15
$rsi : 0x69
$rdi : 0x00007fffffffdf0 → 0x0000000000000000 ←
$rip : 0x000055555555f8 → <main+995> call 0x5555555556c7 <func2> ←
$r8 : 0x0
$r9 : 0x5
$r10 : 0x00007fffffff7f4 → 0xf7d2ad1e00000000
$r11 : 0x246
$r12 : 0x000055555555130 → <_start+0> xor ebp, ebp
$r13 : 0x0
$r14 : 0x0
$r15 : 0x0
$eflags: [ZERO carry PARITY adjust sign trap INTERRUPT direction overflow resume virtualx86 identification]
$cs: 0x0033 $ss: 0x002b $ds: 0x0000 $es: 0x0000 $fs: 0x0000 $gs: 0x0000

0x00007fffffffdf0 | +0x0000: 0x0000038000000380 ← $rsp
0x00007fffffffdf8 | +0x0008: 0x0000038000000380
0x00007fffffffdf0 | +0x0010: 0x0000000000000000 ← $rax, $rdi
0x00007fffffffdf8 | +0x0018: 0x0000007b00000380
0x00007fffffff000 | +0x0020: 0x0000147600001f4c
0x00007fffffff008 | +0x0028: 0x000002ae000003f5
0x00007fffffff010 | +0x0030: 0x000015e1000021e7
0x00007fffffff018 | +0x0038: 0x00000fd0000069b

0x5555555555ec <main+983> lea rax, [rbp-0xd0]
0x5555555555f3 <main+990> mov esi, edx
0x5555555555f5 <main+992> mov rdi, rax
→ 0x5555555555f8 <main+995> call 0x5555555556c7 <func2>
└─ 0x5555555556c7 <func2+0> push rbp
0x5555555556c8 <func2+1> mov rbp, rsp
0x5555555556cb <func2+4> sub rsp, 0x20
0x5555555556cf <func2+8> mov QWORD PTR [rbp-0x18], rdi
0x5555555556d3 <func2+12> mov DWORD PTR [rbp-0x1c], esi
0x5555555556d6 <func2+15> mov edi, 0x10

func2 (
  $rdi = 0x00007fffffffdf0 → 0x0000000000000000,
  $rsi = 0x0000000000000069,
  $rdx = 0x0000000000000069
)

[#0] Id 1, Name: "c_maths", stopped 0x555555555f8 in main (), reason: SINGLE STEP

[#0] 0x555555555f8 → main()
[#1] 0x7ffff7cd0cca → __libc_start_main(main=0x555555555215 <main>, argc=0x1, argv=0x7fffffff1b8, init=<optimized out>)
[#2] 0x55555555515a → _start()

gef>

```

https://blog.csdn.net/qq_35056292

rbp-0x18为arg1，也就是&v11，地址为0x00007fffffffdf0；rbp-0x1c为arg2，也就是n

```

[ Legend: Modified register | Code | Heap | Stack | String ]

$rax : 0x00007fffffffdf0 → 0x0000000000000000
$rbx : 0x2
$rcx : 0x6
$rdx : 0x69
$rsp : 0x00007fffffffdfb0 → 0x000000000000000e ← $rsp
$rbp : 0x00007fffffffdfd0 → 0x00007fffffff0c0 → 0x0000555555557f0 → <__libc_csu_init+0> push r15
$rsi : 0x69
$rdi : 0x00007fffffffdf0 → 0x0000000000000000
$rip : 0x0000555555556cf → <func2+8> mov QWORD PTR [rbp-0x18], rdi
$r8 : 0x0
$r9 : 0x5
$r10 : 0x00007fffffff7f4 → 0xf7d2ad1e00000000
$r11 : 0x246
$r12 : 0x000055555555130 → <_start+0> xor ebp, ebp
$r13 : 0x0
$r14 : 0x0
$r15 : 0x0
$eflags: [zero carry parity adjust sign trap INTERRUPT direction overflow resume virtualx86 identification]
$cs: 0x0033 $ss: 0x002b $ds: 0x0000 $es: 0x0000 $fs: 0x0000 $gs: 0x0000

0x00007fffffffdfb0 | +0x0000: 0x000000000000000e ← $rsp
0x00007fffffffdfb8 | +0x0008: 0x00000000555556b4
0x00007fffffffdfc0 | +0x0010: 0x00007fffffff0c0 → 0x0000555555557f0 → <__libc_csu_init+0> push r15
0x00007fffffffdfc8 | +0x0018: 0x0000000600000001
0x00007fffffffdfd0 | +0x0020: 0x00007fffffff0c0 → 0x0000555555557f0 → <__libc_csu_init+0> push r15 ← $rbp
0x00007fffffffdfd8 | +0x0028: 0x0000555555555fd → <main+1000> add DWORD PTR [rbp-0x34], 0x1
0x00007fffffffdfde0 | +0x0030: 0x0000038000000380
0x00007fffffffdfde8 | +0x0038: 0x0000038000000380

0x5555555556c7 <func2+0> push rbp
0x5555555556c8 <func2+1> mov rbp, rsp
0x5555555556cb <func2+4> sub rsp, 0x20
→ 0x5555555556cf <func2+8> mov QWORD PTR [rbp-0x18], rdi
0x5555555556d3 <func2+12> mov DWORD PTR [rbp-0x1c], esi
0x5555555556d6 <func2+15> mov edi, 0x10
0x5555555556db <func2+20> call 0x5555555550d0 <malloc@plt>
0x5555555556e0 <func2+25> mov QWORD PTR [rbp-0x8], rax
0x5555555556e4 <func2+29> mov rax, QWORD PTR [rbp-0x8]

[#0] Id 1, Name: "c_maths", stopped 0x5555555556cf in func2 (), reason: SINGLE STEP

[#0] 0x5555555556cf → func2()
[#1] 0x5555555555fd → main()
[#2] 0x7ffff7cd0cca → __libc_start_main(main=0x555555555215 <main>, argc=0x1, argv=0x7fffffff1b8, init=<optimized o
[#3] 0x55555555515a → _start()

gef>

```

https://blog.csdn.net/qq_35056292

rbp-0x8 为v3的首地址，地址为0x00005555555592a0


```

[ Legend: Modified register | Code | Heap | Stack | String ]

$rax : 0x00005555555592a0 → 0x0000000000000069 ("i?")
$rbx : 0x2
$rcx : 0x00005555555592b0 → 0x0000000000000000
$rdx : 0x69
$rspx : 0x00007fffffffdfb0 → 0x000000690000000e
$rbp : 0x00007fffffffdf0 → 0x00007fffffff0c0 → 0x0000555555557f0 → <__libc_csu_init+0> push r15
$rsi : 0x10
$rdi : 0x00005555555592a0 → 0x0000000000000069 ("i?")
$rip : 0x00005555555556ed → <func2+38> mov rax, QWORD PTR [rbp-0x18]
$r8 : 0x00005555555592a0 → 0x0000000000000069 ("i?")
$r9 : 0x00007fffffff7e68be0 → 0x00005555555592b0 → 0x0000000000000000
$r10 : 0x2b0
$r11 : 0x20
$r12 : 0x0000555555555130 → <_start+0> xor ebp, ebp
$r13 : 0x0
$r14 : 0x0
$r15 : 0x0
eflags: [ZERO carry PARITY adjust sign trap INTERRUPT direction overflow resume virtualx86 identification]
cs: 0x0033 $ss: 0x002b $ds: 0x0000 $es: 0x0000 $fs: 0x0000 $gs: 0x0000

0x00007fffffffdfb0|+0x0000: 0x000000690000000e ← $rsp
0x00007fffffffdfb8|+0x0008: 0x00007fffffffdf0 → 0x0000000000000000
0x00007fffffffdfc0|+0x0010: 0x00007fffffff0c0 → 0x0000555555557f0 → <__libc_csu_init+0> push r15
0x00007fffffffdfc8|+0x0018: 0x00005555555592a0 → 0x0000000000000069 ("i?")
0x00007fffffffdfd0|+0x0020: 0x00007fffffff0c0 → 0x0000555555557f0 → <__libc_csu_init+0> push r15 ← $rbp
0x00007fffffffdfd8|+0x0028: 0x0000555555555fd → <main+1000> add DWORD PTR [rbp-0x34], 0x1
0x00007fffffffdfde0|+0x0030: 0x0000038000000380
0x00007fffffffdfde8|+0x0038: 0x0000038000000380

0x5555555556e4 <func2+29> mov rax, QWORD PTR [rbp-0x8]
0x5555555556e8 <func2+33> mov edx, DWORD PTR [rbp-0x1c]
0x5555555556eb <func2+36> mov DWORD PTR [rax], edx
→ 0x5555555556ed <func2+38> mov rax, QWORD PTR [rbp-0x18]
0x5555555556f1 <func2+42> mov rax, QWORD PTR [rax]
0x5555555556f4 <func2+45> test rax, rax
0x5555555556f7 <func2+48> jne 0x55555555712 <func2+75>
0x5555555556f9 <func2+50> mov rax, QWORD PTR [rbp-0x18]
0x5555555556fd <func2+54> mov rdx, QWORD PTR [rbp-0x8]

[#0] Id 1, Name: "c_maths", stopped 0x5555555556ed in func2 (), reason: SINGLE STEP

[#0] 0x5555555556ed → func2()
[#1] 0x5555555556fd → main()
[#2] 0x7ffff7cd0cca → __libc_start_main(main=0x55555555215 <main>, argc=0x1, argv=0x7fffffff1b8, init=<optimized c
[#3] 0x55555555515a → _start()

gef> |

```

rbp-0x1c为arg2, 赋值给edx
 将edx赋值给[rax], rax保存的地址,
 对该地址赋值

由于一开始v11为空, 因此进入else块。用v3对v11赋值:

```

[ Legend: Modified register | Code | Heap | Stack | String ]

$rax : 0x00007fffffffdf0 → 0x0000000000000000
$rbx : 0x2
$rcx : 0x00005555555592b0 → 0x0000000000000000
$rdx : 0x00005555555592a0 → 0x0000000000000069 ("i?")
$rsp : 0x00007fffffffdfb0 → 0x000000690000000e
$rbp : 0x00007fffffffdfd0 → 0x00007fffffffec0 → 0x0000555555557f0 → <__libc_csu_init+0> push r15
$rsi : 0x10
$rdi : 0x00005555555592a0 → 0x0000000000000069 ("i?")
$rip : 0x000055555555701 → <func2+58> mov QWORD PTR [rax], rdx
$r8 : 0x00005555555592a0 → 0x0000000000000069 ("i?")
$r9 : 0x00007ffff7e68be0 → 0x00005555555592b0 → 0x0000000000000000
$r10 : 0x2b0
$r11 : 0x20
$r12 : 0x000055555555130 → <_start+0> xor ebp, ebp
$r13 : 0x0
$r14 : 0x0
$r15 : 0x0
$eflags: [ZERO carry PARITY adjust sign trap INTERRUPT direction overflow resume virtualx86 identification]
$cs: 0x0033 $ss: 0x002b $ds: 0x0000 $es: 0x0000 $fs: 0x0000 $gs: 0x0000

0x00007fffffffdfb0|+0x0000: 0x000000690000000e ← $rsp
0x00007fffffffdfb8|+0x0008: 0x00007fffffffdf0 → 0x0000000000000000
0x00007fffffffdfc0|+0x0010: 0x00007fffffffec0 → 0x0000555555557f0 → <__libc_csu_init+0> push r15
0x00007fffffffdfc8|+0x0018: 0x00005555555592a0 → 0x0000000000000069 ("i?")
0x00007fffffffdfd0|+0x0020: 0x00007fffffffec0 → 0x0000555555557f0 → <__libc_csu_init+0> push r15 ← $rbp
0x00007fffffffdfd8|+0x0028: 0x0000555555555fd → <main+1000> add DWORD PTR [rbp-0x34], 0x1
0x00007fffffffdfde0|+0x0030: 0x0000038000000380
0x00007fffffffdfde8|+0x0038: 0x0000038000000380

0x5555555556f6 <func2+47> shl BYTE PTR [rbp+0x19], 0x48
0x5555555556fa <func2+51> mov eax, DWORD PTR [rbp-0x18] arg1, 即&v11
0x5555555556fd <func2+54> mov rdx, QWORD PTR [rbp-0x8] rbp-0x8为v3的首地址
→ 0x55555555701 <func2+58> mov QWORD PTR [rax], rdx
0x55555555704 <func2+61> mov rax, QWORD PTR [rbp-0x8]
0x55555555708 <func2+65> mov QWORD PTR [rax+0x8], 0x0
0x55555555710 <func2+73> jmp 0x5555555572c <func2+101>
0x55555555712 <func2+75> mov rax, QWORD PTR [rbp-0x18]
0x55555555716 <func2+79> mov rdx, QWORD PTR [rax]

[#0] Id 1, Name: "c_maths", stopped 0x55555555701 in func2 (), reason: SINGLE STEP

[#0] 0x55555555701 → func2()
[#1] 0x555555555fd → main()
[#2] 0x7ffff7cd0cca → __libc_start_main(main=0x55555555215 <main>, argc=0x1, argv=0x7fffffff1b8, init=<optimized out>)
[#3] 0x5555555515a → _start()

gef>
https://blog.csdn.net/qq_35056292

```

第一次进入func2以后，v11的内容为0x64

```

$rbx : 0x2
$rcx : 0x6
$rdx : 0x72
$rsp : 0x00007fffffffdf0 → 0x0000038000000380
$rbp : 0x00007fffffff0c0 → 0x0000555555557f0 → <_libc_csu_init+0> push r15
$rsi : 0x72
$rdi : 0x00007fffffffdf0 → 0x00005555555592a0 → 0x0000000000000069 ("i?")
$rip : 0x0000555555555f8 → <main+995> call 0x5555555556c7 <func2>
$r8 : 0x00005555555592a0 → 0x0000000000000069 ("i?")
$r9 : 0x00007ffff7e68be0 → 0x00005555555592b0 → 0x0000000000000000
$r10 : 0x2b0
$r11 : 0x20
$r12 : 0x000055555555130 → <_start+0> xor ebp, ebp
$r13 : 0x0
$r14 : 0x0
$r15 : 0x0
$eflags: [ZERO carry PARITY adjust sign trap INTERRUPT direction overflow resume virtualx86 identification]
$cs: 0x0033 $ss: 0x002b $ds: 0x0000 $es: 0x0000 $fs: 0x0000 $gs: 0x0000

0x00007fffffffdf0 | +0x0000: 0x0000038000000380 ← $rsp
0x00007fffffffdf8 | +0x0008: 0x0000038000000380
0x00007fffffffdf0 | +0x0010: 0x00005555555592a0 → 0x0000000000000069 ("i?") ← $rax, $rdi
0x00007fffffffdf8 | +0x0018: 0x0000007b00000380
0x00007fffffff000 | +0x0020: 0x0000147600001f4c
0x00007fffffff008 | +0x0028: 0x000002ae000003f5
0x00007fffffff010 | +0x0030: 0x000015e1000021e7
0x00007fffffff018 | +0x0038: 0x00000fd0000069b

0x5555555555ec <main+983> lea rax, [rbp-0xd0]
0x5555555555f3 <main+990> mov esi, edx
0x5555555555f5 <main+992> mov rdi, rax
→ 0x5555555555f8 <main+995> call 0x5555555556c7 <func2>
↳ 0x5555555556c7 <func2+0> push rbp
0x5555555556c8 <func2+1> mov rbp, rsp
0x5555555556cb <func2+4> sub rsp, 0x20
0x5555555556cf <func2+8> mov QWORD PTR [rbp-0x18], rdi
0x5555555556d3 <func2+12> mov DWORD PTR [rbp-0x1c], esi
0x5555555556d6 <func2+15> mov edi, 0x10

func2 (
  $rdi = 0x00007fffffffdf0 → 0x00005555555592a0 → 0x0000000000000069 ("i?"),
  $rsi = 0x0000000000000072,
  $rdx = 0x0000000000000072
)

[#0] Id 1, Name: "c_maths", stopped 0x555555555f8 in main (), reason: SINGLE STEP

[#0] 0x555555555f8 → main()
[#1] 0x7ffff7cd0cca → __libc_start_main(main=0x55555555215 <main>, argc=0x1, argv=0x7fffffe1b8, init=<optim
[#2] 0x5555555515a → _start()

gef> |

```

第二次进入func2，为v3重新分配一块内存空间，地址为0x00005555555592c0，保存在 rbp-0x8中

```

[ Legend: Modified register | Code | Heap | Stack | String ]

$rax : 0x00005555555592c0 → 0x0000000000000000 ←
$rbx : 0x2
$rcx : 0x00005555555592d0 → 0x0000000000000000
$rdx : 0x21
$rsp : 0x00007fffffffdfb0 → 0x000000720000000e
$rbp : 0x00007fffffffdfd0 → 0x00007fffffff0c0 → 0x0000555555557f0 → <__libc_csu_init+0> push r15
$rsi : 0x10
$rdi : 0x00005555555592c0 → 0x0000000000000000
$rip : 0x00005555555556e0 → <func2+25> mov QWORD PTR [rbp-0x8], rax
$r8 : 0x00005555555592c0 → 0x0000000000000000
$r9 : 0x00007ffff7e68be0 → 0x00005555555592d0 → 0x0000000000000000
$r10 : 0x2b0
$r11 : 0x20
$r12 : 0x000055555555130 → <_start+0> xor ebp, ebp
$r13 : 0x0
$r14 : 0x0
$r15 : 0x0
$eflags: [ZERO carry PARITY adjust sign trap INTERRUPT direction overflow resume virtualx86 identification]
$cs: 0x0033 $ss: 0x002b $ds: 0x0000 $es: 0x0000 $fs: 0x0000 $gs: 0x0000

0x00007fffffffdfb0|+0x0000: 0x000000720000000e ← $rsp
0x00007fffffffdfb8|+0x0008: 0x00007fffffffdf0 → 0x00005555555592a0 → 0x00000000000000069 ("i?")
0x00007fffffffdfc0|+0x0010: 0x00007fffffff0c0 → 0x0000555555557f0 → <__libc_csu_init+0> push r15
0x00007fffffffdfc8|+0x0018: 0x0000000600000001
0x00007fffffffdfd0|+0x0020: 0x00007fffffff0c0 → 0x0000555555557f0 → <__libc_csu_init+0> push r15 ← $rbp
0x00007fffffffdfd8|+0x0028: 0x0000555555555fd → <main+1000> add DWORD PTR [rbp-0x34], 0x1
0x00007fffffffdfde0|+0x0030: 0x0000038000000380
0x00007fffffffdfde8|+0x0038: 0x0000038000000380

0x5555555556d3 <func2+12> mov DWORD PTR [rbp-0x1c], esi
0x5555555556d6 <func2+15> mov edi, 0x10
0x5555555556db <func2+20> call 0x5555555550d0 <malloc@plt>
→ 0x5555555556e0 <func2+25> mov QWORD PTR [rbp-0x8], rax ← 重新为v3分配内存空间
0x5555555556e4 <func2+29> mov rax, QWORD PTR [rbp-0x8]
0x5555555556e8 <func2+33> mov edx, DWORD PTR [rbp-0x1c]
0x5555555556eb <func2+36> mov DWORD PTR [rax], edx
0x5555555556ed <func2+38> mov rax, QWORD PTR [rbp-0x18]
0x5555555556f1 <func2+42> mov rax, QWORD PTR [rax]

[#0] Id 1, Name: "c_maths", stopped 0x5555555556e0 in func2 (), reason: SINGLE STEP

[#0] 0x5555555556e0 → func2()
[#1] 0x5555555555fd → main()
[#2] 0x7ffff7cd0cca → __libc_start_main(main=0x555555555215 <main>, argc=0x1, argv=0x7fffffff1b8, init=<optimized out>)
[#3] 0x55555555515a → _start()

gef>

```

接下来将a2赋值给v3

```

[ Legend: Modified register | Code | Heap | Stack | String ]

$rax : 0x00005555555592c0 → 0x0000000000000072 ("r"? )
$rbx : 0x2
$rcx : 0x00005555555592d0 → 0x0000000000000000
$rdx : 0x72
$rsp : 0x00007fffffffdfb0 → 0x000000720000000e
$rbp : 0x00007fffffffdf0 → 0x00007fffffff0c0 → 0x0000555555557f0 → <__libc_csu_init+0> push r15
$rsi : 0x10
$rdi : 0x00005555555592c0 → 0x0000000000000072 ("r"? )
$rip : 0x00005555555556ed → <func2+38> mov rax, QWORD PTR [rbp-0x18]
$r8 : 0x00005555555592c0 → 0x0000000000000072 ("r"? )
$r9 : 0x00007ffff7e68be0 → 0x00005555555592d0 → 0x0000000000000000
$r10 : 0x2b0
$r11 : 0x20
$r12 : 0x0000555555555130 → <_start+0> xor ebp, ebp
$r13 : 0x0
$r14 : 0x0
$r15 : 0x0
eflags: [ZERO carry PARITY adjust sign trap INTERRUPT direction overflow resume virtualx86 identification]
cs: 0x0033 $ss: 0x002b $ds: 0x0000 $es: 0x0000 $fs: 0x0000 $gs: 0x0000

0x00007fffffffdfb0|+0x0000: 0x000000720000000e ← $rsp
0x00007fffffffdfb8|+0x0008: 0x00007fffffffdf0 → 0x00005555555592a0 → 0x0000000000000069 ("i"? )
0x00007fffffffdfc0|+0x0010: 0x00007fffffff0c0 → 0x0000555555557f0 → <__libc_csu_init+0> push r15
0x00007fffffffdfc8|+0x0018: 0x00005555555592c0 → 0x0000000000000072 ("r"? )
0x00007fffffffdfd0|+0x0020: 0x00007fffffff0c0 → 0x0000555555557f0 → <__libc_csu_init+0> push r15 ← $rbp
0x00007fffffffdfd8|+0x0028: 0x0000555555555fd → <main+1000> add DWORD PTR [rbp-0x34], 0x1
0x00007fffffffdfde0|+0x0030: 0x0000038000000380
0x00007fffffffdfde8|+0x0038: 0x0000038000000380

0x5555555556e4 <func2+29> mov rax, QWORD PTR [rbp-0x8]
0x5555555556e8 <func2+33> mov edx, DWORD PTR [rbp-0x1c]
0x5555555556eb <func2+36> mov DWORD PTR [rax], edx
→ 0x5555555556ed <func2+38> mov rax, QWORD PTR [rbp-0x18]
0x5555555556f1 <func2+42> mov rax, QWORD PTR [rax]
0x5555555556f4 <func2+45> test rax, rax
0x5555555556f7 <func2+48> jne 0x555555555712 <func2+75>
0x5555555556f9 <func2+50> mov rax, QWORD PTR [rbp-0x18]
0x5555555556fd <func2+54> mov rdx, QWORD PTR [rbp-0x8]

[#0] Id 1, Name: "c_maths", stopped 0x5555555556ed in func2 (), reason: SINGLE STEP

[#0] 0x5555555556ed → func2()
[#1] 0x5555555555fd → main()
[#2] 0x7ffff7cd0cca → __libc_start_main(main=0x555555555215 <main>, argc=0x1, argv=0x7fffffff1b8, init=<optimized out>)
[#3] 0x55555555515a → _start()

gef>
https://blog.csdn.net/qq_35056292

```

然后判断 `if(*a1)`

```

[ Legend: Modified register | Code | Heap | Stack | String ]
$rax : 0x00005555555592a0 → 0x0000000000000069 ("i?")
$rbx : 0x2
$rcx : 0x00005555555592d0 → 0x0000000000000000
$rdx : 0x72
$rsp : 0x00007fffffffdfb0 → 0x000000720000000e
$rbp : 0x00007fffffffdf0 → 0x00007fffffff0c0 → 0x0000555555557f0 → <__libc_csu_init+0> push r15
$rsi : 0x10
$rdi : 0x00005555555592c0 → 0x0000000000000072 ("r?")
$rip : 0x00005555555556f4 → <func2+45> test rax, rax
$r8 : 0x00005555555592c0 → 0x0000000000000072 ("r?")
$r9 : 0x00007ffff7e68be0 → 0x00005555555592d0 → 0x0000000000000000
$r10 : 0x2b0
$r11 : 0x20
$r12 : 0x0000555555555130 → <_start+0> xor ebp, ebp
$r13 : 0x0
$r14 : 0x0
$r15 : 0x0
$eflags: [ZERO carry PARITY adjust sign trap INTERRUPT direction overflow resume virtualx86 identification]
$cs: 0x0033 $ss: 0x002b $ds: 0x0000 $es: 0x0000 $fs: 0x0000 $gs: 0x0000

0x00007fffffffdfb0|+0x0000: 0x000000720000000e ← $rsp
0x00007fffffffdfb8|+0x0008: 0x00007fffffffdf0 → 0x00005555555592a0 → 0x0000000000000069 ("i?")
0x00007fffffffdfc0|+0x0010: 0x00007fffffff0c0 → 0x0000555555557f0 → <__libc_csu_init+0> push r15
0x00007fffffffdfc8|+0x0018: 0x00005555555592c0 → 0x0000000000000072 ("r?")
0x00007fffffffdfd0|+0x0020: 0x00007fffffff0c0 → 0x0000555555557f0 → <__libc_csu_init+0> push r15 ← $rbp
0x00007fffffffdfd8|+0x0028: 0x0000555555555fd → <main+1000> add DWORD PTR [rbp-0x34], 0x1
0x00007fffffffdf0|+0x0030: 0x0000038000000380
0x00007fffffffdf0|+0x0038: 0x0000038000000380

0x5555555556eb <func2+36> mov     DWORD PTR [rax], edx
0x5555555556ed <func2+38> mov     rax, QWORD PTR [rbp-0x18]
0x5555555556f1 <func2+42> mov     rax, QWORD PTR [rax]
→ 0x5555555556f4 <func2+45> test    rax, rax
0x5555555556f7 <func2+48> jne     0x555555555712 <func2+75>
0x5555555556f9 <func2+50> mov     rax, QWORD PTR [rbp-0x18]
0x5555555556fd <func2+54> mov     rdx, QWORD PTR [rbp-0x8]
0x555555555701 <func2+58> mov     QWORD PTR [rax], rdx
0x555555555704 <func2+61> mov     rax, QWORD PTR [rbp-0x8]

[#0] Id 1, Name: "c_maths", stopped 0x5555555556f4 in func2 (), reason: SINGLE STEP

[#0] 0x5555555556f4 → func2()
[#1] 0x5555555556fd → main()
[#2] 0x7ffff7cd0cca → __libc_start_main(main=0x555555555215 <main>, argc=0x1, argv=0x7fffffff1b8, init=<optimized out>,
[#3] 0x55555555515a → _start()

gef>

```

判断a1是否为空

重点来了，不要眨眼！

```

mov     rax, QWORD PTR [rbp-0x18] ; rbp-0x18保存的是a1，也就是v11的地址赋值给rax
mov     rdx, QWORD PTR [rax] ; 将rax寄存器中保存的地址赋值给rdx，其中这个地址指向的是前一次进入func2时v3的地址

```

```

[ Legend: Modified register | Code | Heap | Stack | String ]

$rax : 0x00007fffffffdf0 → 0x00005555555592a0 → 0x0000000000000069 ("i?")
$rbx : 0x2
$rcx : 0x00005555555592d0 → 0x0000000000000000
$rdx : 0x00005555555592a0 → 0x0000000000000069 ("i?")
$rspx : 0x00007fffffffdf0 → 0x000000720000000e
$rbp : 0x00007fffffffdf0 → 0x00007fffffff0c0 → 0x0000555555557f0 → <__libc_csu_init+0> push r15
$rsi : 0x10
$rdi : 0x00005555555592c0 → 0x0000000000000072 ("r?")
$rip : 0x000055555555719 → <func2+82> mov rax, QWORD PTR [rbp-0x8]
$r8 : 0x00005555555592c0 → 0x0000000000000072 ("r?")
$r9 : 0x00007fffffff7e68be0 → 0x00005555555592d0 → 0x0000000000000000
$r10 : 0x2b0
$r11 : 0x20
$r12 : 0x0000555555555130 → <_start+0> xor ebp, ebp
$r13 : 0x0
$r14 : 0x0
$r15 : 0x0
$eflags: [zero carry PARITY adjust sign trap INTERRUPT direction overflow resume virtualx86 identification]
$cs: 0x0033 $ss: 0x002b $ds: 0x0000 $es: 0x0000 $fs: 0x0000 $gs: 0x0000

0x00007fffffffdfb0|+0x0000: 0x000000720000000e ← $rsp
0x00007fffffffdfb8|+0x0008: 0x00007fffffffdf0 → 0x00005555555592a0 → 0x0000000000000069 ("i?")
0x00007fffffffdfc0|+0x0010: 0x00007fffffff0c0 → 0x0000555555557f0 → <__libc_csu_init+0> push r15
0x00007fffffffdfc8|+0x0018: 0x00005555555592c0 → 0x0000000000000072 ("r?")
0x00007fffffffdfd0|+0x0020: 0x00007fffffff0c0 → 0x0000555555557f0 → <__libc_csu_init+0> push r15 ← $rbp
0x00007fffffffdfd8|+0x0028: 0x0000555555555fd → <main+1000> add DWORD PTR [rbp-0x34], 0x1
0x00007fffffffdfde0|+0x0030: 0x0000038000000380
0x00007fffffffdfde8|+0x0038: 0x0000038000000380

0x55555555710 <func2+73> jmp 0x5555555572c <func2+101>
0x55555555712 <func2+75> mov rax, QWORD PTR [rbp-0x18]
0x55555555716 <func2+79> mov rdx, QWORD PTR [rax]
→ 0x55555555719 <func2+82> mov rax, QWORD PTR [rbp-0x8]
0x5555555571d <func2+86> mov QWORD PTR [rax+0x8], rdx
0x55555555721 <func2+90> mov rax, QWORD PTR [rbp-0x18]
0x55555555725 <func2+94> mov rdx, QWORD PTR [rbp-0x8]
0x55555555729 <func2+98> mov QWORD PTR [rax], rdx
0x5555555572c <func2+101> nop

[#0] Id 1, Name: "c_maths", stopped 0x55555555719 in func2 (), reason: SINGLE STEP

[#0] 0x55555555719 → func2()
[#1] 0x555555555fd → main()
[#2] 0x7ffff7cd0cca → __libc_start_main(main=0x55555555215 <main>, argc=0x1, argv=0x7fffffff1b8, init=<optimized out>)
[#3] 0x5555555515a → _start()

gef>

```

```

mov rax, QWORD PTR [rbp-0x8] ; rbp-0x8保存的是v3的地址，将其赋值给rax寄存器
mov QWORD PTR [rax+0x8], rdx ; rdx保存的是v11的地址所指向的地址

```

```

$rax : 0x0000555555592c0 → 0x0000000000000072 ("r"?
$rbx : 0x2
$rcx : 0x0000555555592d0 → 0x0000000000000000
$rdx : 0x0000555555592a0 → 0x0000000000000069 ("i"?
$rsp : 0x00007fffffffdfb0 → 0x000000720000000e
$rbp : 0x00007fffffffdfd0 → 0x00007fffffff0c0 → 0x0000555555592f0 → <__libc_csu_init+0> push r15
$rsi : 0x10
$rdi : 0x0000555555592c0 → 0x0000000000000072 ("r"?
$rip : 0x0000555555592721 → <func2+90> mov rax, QWORD PTR [rbp-0x18]
$r8 : 0x0000555555592c0 → 0x0000000000000072 ("r"?
$r9 : 0x00007fffffff7e68be0 → 0x0000555555592d0 → 0x0000000000000000
$r10 : 0x2b0
$r11 : 0x20
$r12 : 0x0000555555592130 → <_start+0> xor ebp, ebp
$r13 : 0x0
$r14 : 0x0
$r15 : 0x0
$eflags: [zero carry PARITY adjust sign trap INTERRUPT direction overflow resume virtualx86 identification]
$cs: 0x0033 $ss: 0x002b $ds: 0x0000 $es: 0x0000 $fs: 0x0000 $gs: 0x0000

0x00007fffffffdfb0 | +0x0000: 0x000000720000000e ← $rsp
0x00007fffffffdfb8 | +0x0008: 0x00007fffffffdf0 → 0x0000555555592a0 → 0x0000000000000069 ("i"?
0x00007fffffffdfc0 | +0x0010: 0x00007fffffff0c0 → 0x0000555555592f0 → <__libc_csu_init+0> push r15
0x00007fffffffdfc8 | +0x0018: 0x0000555555592c0 → 0x0000000000000072 ("r"?
0x00007fffffffdfd0 | +0x0020: 0x00007fffffff0c0 → 0x0000555555592f0 → <__libc_csu_init+0> push r15 ← $rbp
0x00007fffffffdfd8 | +0x0028: 0x0000555555592fd → <main+1000> add DWORD PTR [rbp-0x34], 0x1
0x00007fffffffdfde | +0x0030: 0x0000038000000380
0x00007fffffffdfde8 | +0x0038: 0x0000038000000380

0x555555555714 <func2+77> rex.RB call 0x55559d65e262
0x55555555571a <func2+83> mov eax, DWORD PTR [rbp-0x8]
0x55555555571d <func2+86> mov QWORD PTR [rax+0x8], rdx
→ 0x555555555721 <func2+90> mov rax, QWORD PTR [rbp-0x18]
0x555555555725 <func2+94> mov rdx, QWORD PTR [rbp-0x8]
0x555555555729 <func2+98> mov QWORD PTR [rax], rdx
0x55555555572c <func2+101> nop
0x55555555572d <func2+102> leave
0x55555555572e <func2+103> ret

[#0] Id 1, Name: "c_maths", stopped 0x555555555721 in func2 (), reason: SINGLE STEP

[#0] 0x555555555721 → func2()
[#1] 0x555555555fd → main()
[#2] 0x7ffff7cd0cca → __libc_start_main(main=0x555555555215 <main>, argc=0x1, argv=0x7fffffff1b8, init=<optimized out>)
[#3] 0x55555555515a → _start()

gef> x/16xb $rax+0x8
0x55555555592c8: 0xa0 0x92 0x55 0x55 0x55 0x55 0x00 0x00
0x55555555592d0: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
gef>

```

mov rax, QWORD PTR [rbp-0x18] ; rbp-0x18保存的是a1，也就是v11的地址赋值给rax，也就是对函数的返回值进行赋值
 mov rdx, QWORD PTR [rbp-0x8] ; 将v3的地址赋值给rdx
 mov QWORD PTR [rax],rdx ; 将v3的地址赋值给rax保存的地址所指向的地址0x0000555555592c0，也就是0x00007fffffffdf0（v11）指向0x0000555555592c0


```

[ Legend: Modified register | Code | Heap | Stack | String ]
$rax : 0x00007fffffffdf0 → 0x0000555555592c0 → 0x0000000000000072 ("r"? ←
$rbx : 0x2
$rcx : 0x0000555555592d0 → 0x0000000000000000
$rdx : 0x0000555555592c0 → 0x0000000000000072 ("r"?
$rsp : 0x00007fffffffdfb0 → 0x000000720000000e
$rbp : 0x00007fffffffdf0 → 0x00007fffffff0c0 → 0x0000555555557f0 → <__libc_csu_init+0> push r15
$rsi : 0x10
$rdi : 0x0000555555592c0 → 0x0000000000000072 ("r"?
$rip : 0x00005555555572c → <func2+101> nop
$r8 : 0x0000555555592c0 → 0x0000000000000072 ("r"?
$r9 : 0x00007ffff7e68be0 → 0x0000555555592d0 → 0x0000000000000000
$r10 : 0x2b0
$r11 : 0x20
$r12 : 0x000055555555130 → <_start+0> xor ebp, ebp
$r13 : 0x0
$r14 : 0x0
$r15 : 0x0
$eflags: [zero carry PARITY adjust sign trap INTERRUPT direction overflow resume virtualx86 identification]
$cs: 0x0033 $ss: 0x002b $ds: 0x0000 $es: 0x0000 $fs: 0x0000 $gs: 0x0000

0x00007fffffffdfb0 | +0x0000: 0x000000720000000e ← $rsp
0x00007fffffffdfb8 | +0x0008: 0x00007fffffffdf0 → 0x0000555555592c0 → 0x0000000000000072 ("r"?
0x00007fffffffdfc0 | +0x0010: 0x00007fffffff0c0 → 0x0000555555557f0 → <__libc_csu_init+0> push r15
0x00007fffffffdfc8 | +0x0018: 0x0000555555592c0 → 0x0000000000000072 ("r"?
0x00007fffffffdfd0 | +0x0020: 0x00007fffffff0c0 → 0x0000555555557f0 → <__libc_csu_init+0> push r15 ← $rbp
0x00007fffffffdfd8 | +0x0028: 0x000055555555fd → <main+1000> add DWORD PTR [rbp-0x34], 0x1
0x00007fffffffdfde0 | +0x0030: 0x0000038000000380
0x00007fffffffdfde8 | +0x0038: 0x0000038000000380

0x55555555720 <func2+89> or BYTE PTR [rax-0x75], cl
0x55555555723 <func2+92> rex.RB call 0x55554daae271
0x55555555729 <func2+98> mov QWORD PTR [rax], rdx
→ 0x5555555572c <func2+101> nop
0x5555555572d <func2+102> leave
0x5555555572e <func2+103> ret
0x5555555572f <func1+0> push rbp
0x55555555730 <func1+1> mov rbp, rsp
0x55555555733 <func1+4> mov DWORD PTR [rbp-0x14], edi

[#0] Id 1, Name: "c_maths", stopped 0x5555555572c in func2 (), reason: SINGLE STEP

[#0] 0x5555555572c → func2()
[#1] 0x555555555fd → main()
[#2] 0x7ffff7cd0cca → __libc_start_main(main=0x55555555215 <main>, argc=0x1, argv=0x7fffffff1b8, init=<optimized
[#3] 0x5555555515a → _start()

gef>

```

最终, v11[1]的值为9483

```

[ Legend: Modified register | Code | Heap | Stack | String ]
$rax : 0x000055555555660 → 0x000000000000250b ("%?")
$rbx : 0x7
$rcx : 0x16 9483
$rdx : 0x15
$rsp : 0x00007fffffffdf0 → 0x0000038000000380
$rbp : 0x00007fffffff0c0 → 0x0000555555557f0 → <__libc_csu_init+0> push r15
$rsi : 0x10
$rdi : 0x2512
$rip : 0x000055555555614 → <main+1023> mov rdi, rax
$r8 : 0x0000555555556680 → 0x0000000000002511
$r9 : 0x00007ffff7e68be0 → 0x0000555555556690 → 0x0000000000000000
$r10 : 0x2b0
$r11 : 0x20
$r12 : 0x000055555555130 → <_start+0> xor ebp, ebp
$r13 : 0x0
$r14 : 0x0
$r15 : 0x0
$eflags: [ZERO carry PARITY adjust sign trap INTERRUPT direction overflow resume virtualx86 identification]
$cs: 0x0033 $ss: 0x002b $ds: 0x0000 $es: 0x0000 $fs: 0x0000 $gs: 0x0000

0x00007fffffffdf0 | +0x0000: 0x0000038000000380 ← $rsp
0x00007fffffffdf8 | +0x0008: 0x0000038000000380
0x00007fffffffdf0 | +0x0010: 0x0000555555556680 → 0x0000000000002511
0x00007fffffffdf8 | +0x0018: 0x0000007b00000380
0x00007fffffff000 | +0x0020: 0x0000147600001f4c
0x00007fffffff008 | +0x0028: 0x000002ae000003f5
0x00007fffffff010 | +0x0030: 0x000015e1000021e7
0x00007fffffff018 | +0x0038: 0x00000fd0000069b

0x55555555607 <main+1010> jl 0x5555555555a5 <main+912>
0x55555555609 <main+1012> mov rax, QWORD PTR [rbp-0xd0]
0x55555555610 <main+1019> mov rax, QWORD PTR [rax+0x8]
→ 0x55555555614 <main+1023> mov rdi, rax
0x55555555617 <main+1026> call 0x5555555579b <func3>
0x5555555561c <main+1031> mov DWORD PTR [rbp-0x48], eax
0x5555555561f <main+1034> lea rax, [rbp-0xd4]
0x55555555626 <main+1041> mov rsi, rax
0x55555555629 <main+1044> lea rdi, [rip+0x9e5] # 0x555555556015

[#0] Id 1, Name: "c_maths", stopped 0x55555555614 in main (), reason: BREAKPOINT

[#0] 0x55555555614 → main()
[#1] 0x7ffff7cd0cca → __libc_start_main(main=0x55555555215 <main>, argc=0x1, argv=0x7fffffff1b8, init=<optimized>
[#2] 0x5555555515a → _start()

gef>
https://blog.csdn.net/qq_35056292

```

- 在func3 调用之前打个断点，然后输入 `continue`，直接执行到调用func3的地方。在执行到调用func3时，输入 `si`，进入到func3中。

```

mov QWORD PTR [rbp-0x18], rdi ; rdi为v11[1] 的地址所指向的地址: 0x000055555555660 -> 0x250b, 将其赋值给rbp-0x18
...
cmp QWORD PTR [rbp-0x18], 0x0 ; 判断rbp-0x18保存的地址0x000055555555660 里的值是否为0

```

```

[ Legend: Modified register | Code | Heap | Stack | String ]

$eax : 0x250b
$rbx : 0x7
$rcx : 0x16
$rdx : 0x15
$rspx : 0x00007fffffffdf0 → 0x00007fffffff0c0 → 0x0000555555557f0 → <__libc_csu_init+0> push r15
$rbp : 0x00007fffffffdf0 → 0x00007fffffff0c0 → 0x0000555555557f0 → <__libc_csu_init+0> push r15
$rsi : 0x10
$rdi : 0x000055555555660 → 0x000000000000250b ("%?")
$rip : 0x0000555555557c3 → <func3+40> add DWORD PTR [rbp-0x4], eax
$r8 : 0x000055555555680 → 0x0000000000002511
$r9 : 0x00007ffff7e68be0 → 0x000055555555690 → 0x0000000000000000
$r10 : 0x2b0
$r11 : 0x20
$r12 : 0x000055555555130 → <_start+0> xor ebp, ebp
$r13 : 0x0
$r14 : 0x0
$r15 : 0x0
eflags: [ZERO carry PARITY adjust sign trap INTERRUPT direction overflow resume virtualx86 identification]
cs: 0x0033 $ss: 0x002b $ds: 0x0000 $es: 0x0000 $fs: 0x0000 $gs: 0x0000

0x00007fffffffdf0 | +0x0000: 0x00007fffffff0c0 → 0x0000555555557f0 → <__libc_csu_init+0> push r15 ← $rsp, $rbp
0x00007fffffffdf8 | +0x0008: 0x00005555555561c → <main+1031> mov DWORD PTR [rbp-0x48], eax
0x00007fffffffdf0 | +0x0010: 0x0000038000000380
0x00007fffffffdf8 | +0x0018: 0x0000038000000380
0x00007fffffffdf0 | +0x0020: 0x000055555555680 → 0x0000000000002511
0x00007fffffffdf8 | +0x0028: 0x0000007b00000380
0x00007fffffff000 | +0x0030: 0x0000147600001f4c
0x00007fffffff008 | +0x0038: 0x000002ae000003f5

0x555555557b8 <func3+29> add DWORD PTR [rbp+0x480975c0], eax
0x555555557be <func3+35> mov eax, DWORD PTR [rbp-0x18]
0x555555557c1 <func3+38> mov eax, DWORD PTR [rax]
→ 0x555555557c3 <func3+40> add DWORD PTR [rbp-0x4], eax
0x555555557c6 <func3+43> mov rax, QWORD PTR [rbp-0x18]
0x555555557ca <func3+47> mov rax, QWORD PTR [rax+0x8]
0x555555557ce <func3+51> mov QWORD PTR [rbp-0x18], rax
0x555555557d2 <func3+55> add DWORD PTR [rbp-0x8], 0x1
0x555555557d6 <func3+59> cmp QWORD PTR [rbp-0x18], 0x0

[#0] Id 1, Name: "c_maths", stopped 0x555555557c3 in func3 (), reason: SINGLE STEP

[#0] 0x555555557c3 → func3()
[#1] 0x5555555561c → main()
[#2] 0x7ffff7cd0cca → __libc_start_main(main=0x55555555215 <main>, argc=0x1, argv=0x7fffffff1b8, init=<optimized out>, fini=
[#3] 0x5555555515a → _start()

gef>

```

➤ a1的地址指向的地址赋值给eax, 再从该地址处取值得到0x250b

执行完 `add dword [var_4h], eax` , v4的值如下:

```

[ Legend: Modified register | Code | Heap | Stack | String ]
$rax : 0x250b
$rbx : 0x7
$rcx : 0x16
$rdx : 0x15
$rsp : 0x00007fffffffdf0 → 0x00007fffffffef0c0 → 0x0000555555557f0 → <_libc_csu_init+0> push r15
$rbp : 0x00007fffffffdf0 → 0x00007fffffffef0c0 → 0x0000555555557f0 → <_libc_csu_init+0> push r15
$rsi : 0x10
$rdi : 0x0000555555556660 → 0x000000000000250b ("%?")
$rip : 0x0000555555557c6 → <func3+43> mov rax, QWORD PTR [rbp-0x18]
$r8 : 0x0000555555556680 → 0x0000000000002511
$r9 : 0x00007ffff7e68be0 → 0x0000555555556690 → 0x0000000000000000
$r10 : 0x2b0
$r11 : 0x20
$r12 : 0x000055555555130 → <_start+0> xor ebp, ebp
$r13 : 0x0
$r14 : 0x0
$r15 : 0x0
$eflags: [zero carry parity adjust sign trap INTERRUPT direction overflow resume virtualx86 identification]
$cs: 0x0033 $ss: 0x002b $ds: 0x0000 $es: 0x0000 $fs: 0x0000 $gs: 0x0000

0x00007fffffffdf0 | 0x0000: 0x00007fffffffef0c0 → 0x0000555555557f0 → <_libc_csu_init+0> push r15 ← $rsp, $rbp
0x00007fffffffdf8 | +0x0008: 0x000055555555661c → <main+1031> mov DWORD PTR [rbp-0x48], eax
0x00007fffffffdf0 | +0x0010: 0x0000038000000380
0x00007fffffffdf8 | +0x0018: 0x0000038000000380
0x00007fffffffdf0 | +0x0020: 0x0000555555556680 → 0x0000000000002511
0x00007fffffffdf8 | +0x0028: 0x0000007b00000380
0x00007fffffffef0 | +0x0030: 0x0000147600001f4c
0x00007fffffffef8 | +0x0038: 0x000002ae000003f5

0x555555557bc <func3+33> or DWORD PTR [rax-0x75], ecx
0x555555557bf <func3+36> rex.RB call 0x55559a565850
0x555555557c5 <func3+42> cld
→ 0x555555557c6 <func3+43> mov rax, QWORD PTR [rbp-0x18]
0x555555557ca <func3+47> mov rax, QWORD PTR [rax+0x8]
0x555555557ce <func3+51> mov QWORD PTR [rbp-0x18], rax
0x555555557d2 <func3+55> add DWORD PTR [rbp-0x8], 0x1
0x555555557d6 <func3+59> cmp QWORD PTR [rbp-0x18], 0x0
0x555555557db <func3+64> jne 0x555555557b3 <func3+24>

[#0] Id 1, Name: "c_maths", stopped 0x555555557c6 in func3 (), reason: SINGLE STEP

[#0] 0x555555557c6 → func3()
[#1] 0x55555555661c → main()
[#2] 0x7ffff7cd0cca → __libc_start_main(main=0x55555555215 <main>, argc=0x1, argv=0x7fffffffef1b8, init=<optimized out>,
[#3] 0x5555555515a → _start()

gef> x/16xb $rbp-0x4
0x7fffffffdfcc: 0x0b 0x25 0x00 0x00 0xc0 0xe0 0xff 0xff
0x7fffffffdfd4: 0xff 0x7f 0x00 0x00 0x1c 0x56 0x55 0x55
gef>

```

https://blog.csdn.net/qq_35056292

然后继续取v11的值，因为v11是个qword类型，因此汇编中一次是加了8，最终从v11取的值是9474。

也就是说，每次取的都是v11中下标为1的数去计算。因此，可以把v11当成是一个二维数组，每个数组保存的都是func2计算后的结果。每一维的[1]元素在func3中被取出计算。

3. pwntools 脚本

根据前面的分析结果，写出pwntools的脚本与目标服务器进行交互：

```

# -*- coding:utf-8 -*-
from pwn import *

# 第一个输入
host = 'cmaths.darkarmy.xyz'
port = 7001
print("[i] connecting to {}:{}".format(host, port))
conn = remote(host, port)

print("[>] p1e8s3")
conn.sendline("p1e8s3")
flag_part1 = "p1e8s3"

```

```

# 第二个输入
v13 = []
for i in range(0, 9):
    response = conn.recvline().strip()
    v13.append(int(response))
    print("[<] {}".format(response))

v34 = sum(v13)
v29 = v34 // 9

v32 = 0
for m in range(0, 9): # [0, 8]
    v8 = pow(v13[m]-v29, 2)
    v32 += v8

import math
v28 = int(math.sqrt(v32/9))
print("[>] {}".format(v28))
conn.sendline(str(v28))

flag_part2 = conn.recvline().strip()
print("[<] {}".format(flag_part2))
_blank = conn.recvline().strip()
print("[<] {}".format(_blank))
v27 = conn.recvline().strip()
print("[<] {}".format(v27))
v27 = int(v27)

# 第三个输入
v11 = []
v26 = 0

# === func1 ~ func3 ===
def func1(a1):
    v2 = a1
    v3 = 0

    while v2 > 0:
        v3 += (v2 % 10)
        v2 = v2 // 10

    return v3

def func2(a1, a2):
    # 需要修改a1的值, 因此返回a1, 而不是伪代码中的result (返回了未被使用)
    v3 = [0, 0]
    v3[0] = a2

    if a1:
        v3[1] = a1[0]
        a1 = v3
    else:
        a1 = v3
        v3[1] = 0

    return a1

```

```

def func3(a1):
    v4 = 0
    v3 = 0
    for v2 in a1:
        if not (v3 & 1):
            v4 += v2[1]
            v3 += 1

    return v4

# =====

v11_list = []

for n in range(100, v27):
    v26 = func1(n)
    if (not (v26 & 1)) and (not (v26 % 3)):
        v11 = func2(v11, n)
        v11_list.append(v11)

v25 = func3(v11_list)
print("[> {}".format(v25))
conn.sendline(str(v25))
flag_part3 = conn.recvline().strip()
print("< {}".format(flag_part3))

```

最终运行脚本，可以得到flag:

```

$ python test.py
[i] connecting to cmaths.darkarmy.xyz:7001
[+] Opening connection to cmaths.darkarmy.xyz on port 7001: Done
[> p1e8s3
[< 7258
[< 5931
[< 2721
[< 4994
[< 980
[< 2193
[< 1126
[< 5552
[< 932
[> 2294
[< _just_
[<
[< 7658
[> 2438529
[< give_me_the_flag
[*] Closed connection to cmaths.darkarmy.xyz port 7001

```

最终flag为: `darkCTF{p1e8s3_just_give_me_the_flag}`



[创作打卡挑战赛](#) >

[赢取流量/现金/CSDN周边激励大奖](#)