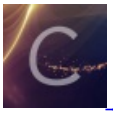
简单逆向题 实验吧之reversemeplz writeup

原创

Peanuts_CTF 于 2018-05-27 14:37:57 发布 1288 收藏
分类专栏： 逆向入门 文章标签： 逆向入门 实验吧 动态调试

本文链接：https://blog.csdn.net/w12315q/article/details/80468720


逆向入门 专栏收录该内容
4 篇文章 0 订阅
订阅专栏

reversemeplz 分值：20

来源： 2015poliCTF 难度：中 参与人数：3401人 Get Flag：73人 答题人数：106人 解题通过率：69%

Last month I was trying to simplify an algorithm.. and I found how to mess up a source really really bad. And then this challenge is born. Maybe is really simple or maybe is so hard that all of you will give up. Good luck!
(flag{xxx})

解题链接： http://ctf5.shiyanbar.com/re/challenge 通过

这道题目的链接： http://ctf5.shiyanbar.com/re/challenge 这是题目的下载地址

先用file命令看下文件的属性

```
chenminghang@bogon:~/Desktop$ file challenge.dms
challenge.dms: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamic
lly linked, interpreter /lib/ld-linux.so.2, for GNU/Linux 2.6.32, BuildID[sha1]
=da884e304160351da7785e93dc168eecafe770ed, stripped
```


这个程序是32位的，这样就可以拖进ida32位了

然后先f5一波

```
int __cdecl main(int a1)
{
  char dest; // [esp+0h] [ebp-118h]
  char s; // [esp+80h] [ebp-98h]
  int *v4; // [esp+10Ch] [ebp-Ch]

  v4 = &a1;
  gets(&s);
  if ( sub_8048801((unsigned __int8 *)&s) )
  {
    strcpy(&dest, "The flag is flag{");
    strcat(&dest, &s);
    strcat(&dest, "}");
    puts(&dest);
  }
  else
  {
    puts("Wrong input.");
  }
  return 0;
}
```


由此可知我们需要输入的数存在了s中，接着执行了sub_8048801这个函数

我们进入这个函数

```
_BOOL4 __cdecl sub_8048801(unsigned __int8 *a1)
{
  signed int v1; // edi
  int j; // esi
  char v3; // al
  int v4; // eax
  char v6; // dl
  char i; // al
  char v8; // [esp+3h] [ebp-59h]
  char v9; // [esp+4h] [ebp-58h]
  char v10; // [esp+5h] [ebp-57h]
  int v11; // [esp+10h] [ebp-4Ch]
  char v12; // [esp+14h] [ebp-48h]

  qmemcpy(&v12, &unk_8048980, 60u);
  v1 = 0;
  j = 0;
  do
  {
    if ( (char)a1[j] <= 96 )
      a1[j] = sub_8048519(a1[1] & 1);
    if ( (char)a1[j] > 122 )
      a1[j] = sub_8048519(a1[1] & 2);
    v3 = sub_8048519(a1[j]);
    *(&v9 + j) = v3;
    if ( (unsigned __int8)v3 > 204u && v3 != 207 )
      v1 = 1;
    ++j;
  }
  while ( j != 15 );
  v4 = 0;
  if ( v1 != 1 )
  {
    while ( 1 )
    {
      ++v4;
      if ( (unsigned __int8)*(&v9 + v4) - (unsigned __int8)*(&v8 + v4) != *(&v11 + v4) )
        break;
      if ( v4 == 14 )
      {
        if ( a1[15] )
        {
          v6 = v10;
          for ( i = v9; i != 204; i ^= v6-- )
            ;
          return 0;
        }
        if ( sub_8048519(0) )
          return 0;
        return (unsigned __int8)sub_8048519(*a1) == 98;
      }
    }
  }
  return 0;
}
```


这里又有一个函数sub_8048519

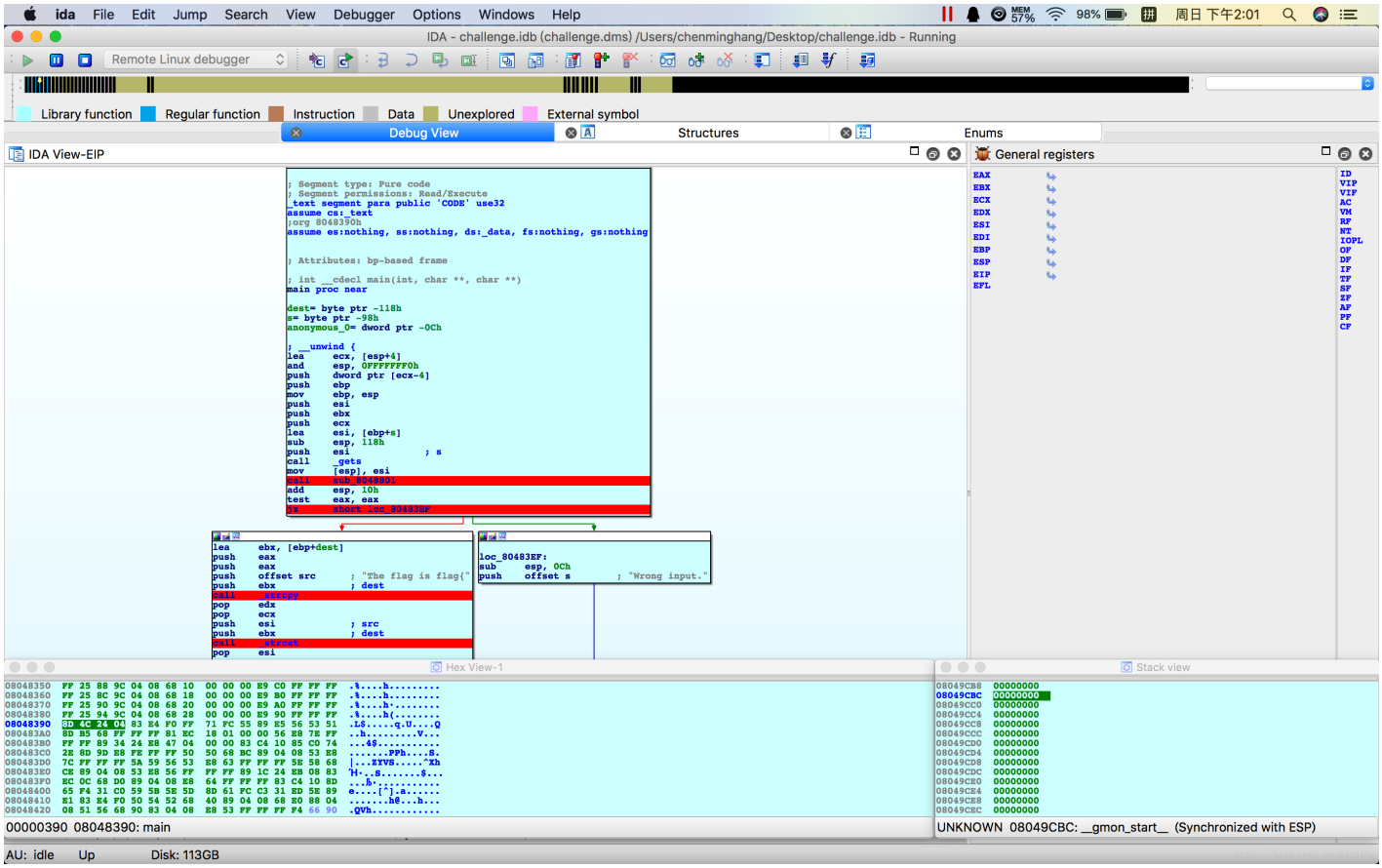
我们再点进去看下

```
v1 = sub_80484FB(a1);
v2 = 19;
if ( (v1 & 0x3F) != 38 )
  v2 = 0;
v3 = v2 | (v1 << 8) | 9 * ((v1 & 0x5F) == 86);
v4 = 71;
if ( (v1 & 0x77) != 116 )
  v4 = 0;
v5 = v4 | v3;
v6 = 84;
if ( (v1 & 0x3F) != 39 )
  v6 = 0;
v7 = v6 | v5;
v8 = 48;
if ( (v1 & 0x4F) != 4 )
  v8 = 0;
v9 = v1 & 0x1F;
v10 = 3 * ((v1 & 0x57) == 80) | 8 * (v9 == 1) | v7 | v8 | 2 * (v9 == 15) | 2 * ((v1 & 0x5B) == 83);
v11 = 114;
v52 = ~v1;
v12 = 8 * (v9 == 2) | 8 * (v9 == 11) | v10 | 2 * ((v1 & 0x57) == 66) | 8 * ((v1 & 0x2E) == 44);
if ( (v1 & 0x37) != 37 )
  v11 = 0;
v13 = v11 | v12;
v14 = 16;
v15 = 0;
if ( (v1 & 0x1C) == 8 )
  v15 = 16;
v16 = ((~v1 & 0x78u) < 1 ? 0x48 : 0) | v15 | v13;
v17 = 64;
if ( (v1 & 0x1D) != 16 )
  v17 = 0;
v18 = v17 | v16;
v19 = 0;
if ( (v1 & 0xF) == 11 )
  v19 = 16;
v20 = 4 * ((v1 & 0x55) == 64) | v19 | v18;
v21 = 72;
if ( (v1 & 0x4B) != 1 )
  v21 = 0;
v22 = v21 | v20;
v23 = 24;
if ( (v1 & 0x47) != 1 )
  v23 = 0;
v24 = v23 | v22;
v25 = 96;
if ( (v1 & 0x2B) != 34 )
  v25 = 0;
v26 = ((v52 & 0x55u) < 1 ? 0x48 : 0) | v25 | v24;
v27 = 0;
if ( (v1 & 0x31) == 16 )
  v27 = 16;
v28 = v27 | v26;
v29 = 0;
if ( (v1 & 0x55) == 81 )
  v29 = 68;
v30 = v29 | v28;
v31 = 0;
if ( (v1 & 0xE) == 8 )
  v31 = 32;
v32 = v31 | v30;
v33 = 97;
if ( (v1 & 0x59) != 72 )
  v33 = 0;
v34 = 81;
v35 = v33 | v32;
```

```
  v38 = 37;
  if ( (v1 & 0x47) != 66 )
    v38 = 0;
  v39 = v37 | v38 | 8 * ((v1 & 0x43) == 2);
  if ( (v1 & 0x46) != 2 )
    v14 = 0;
  v40 = v14 | v39;
  v41 = 80;
  if ( v36 != 3 )
    v41 = 0;
  v42 = v41 | v40;
  v43 = 70;
  if ( v36 != 1 )
    v43 = 0;
  v44 = v43 | v42;
  v45 = 40;
  if ( (v1 & 0x70) != 64 )
    v45 = 0;
  v46 = 0;
  v47 = 4 * ((v1 & 0x41) == 1) | ((v52 & 0xBu) < 1 ? 0x60 : 0) | v45 | v44;
  if ( (v1 & 0x48) == 64 )
    v46 = 32;
  v48 = v47 | v46;
  v49 = (v1 & 0x21) == 1;
  v50 = 0;
  if ( v49 )
    v50 = 68;
  return v48 | v50;
}
```


这么长的一个函数。。。一般情况下不会让我们分析的吧，毕竟这只是个简单的题目，那我们就用动态分析的方法看一看

用ida远程调试一下



这个是调试的界面，我把断点下在了主函数的call位置进入call的函数

```
_BOOL4 __cdecl sub_8048801(unsigned __int8 *a1)
{
  signed int v1; // edi
  int j; // esi
  char v3; // al
  int v4; // eax
  char v6; // dl
  char i; // al
  char v8; // [esp+3h] [ebp-59h]
  char v9; // [esp+4h] [ebp-58h]
  char v10; // [esp+5h] [ebp-57h]
  int v11; // [esp+10h] [ebp-4Ch]
  char v12; // [esp+14h] [ebp-48h]

  qmemcpy(&v12, &unk_8048980, 60u);
  v1 = 0;
  j = 0;
  do
  {
    if ( (char)a1[j] <= 96 )
      a1[j] = sub_8048519(a1[1] & 1);
    if ( (char)a1[j] > 122 )
      a1[j] = sub_8048519(a1[1] & 2);
    v3 = sub_8048519(a1[j]);
    *(&v9 + j) = v3;
    if ( (unsigned __int8)v3 > 204u && v3 != 207 )
      v1 = 1;
    ++j;
  }
  while ( j != 15 );
  v4 = 0;
  if ( v1 != 1 )
  {
    while ( 1 )
    {
      ++v4;
      if ( (unsigned __int8)*(&v9 + v4) - (unsigned __int8)*(&v8 + v4) != *(&v11 + v4) )
        break;
      if ( v4 == 14 )
      {
        if ( a1[15] )
        {
          v6 = v10;
          for ( i = v9; i != 204; i ^= v6-- )
            ;
          return 0;
        }
        if ( sub_8048519(0) )
          return 0;
        return (unsigned __int8)sub_8048519(*a1) == 98;
      }
    }
  }
  return 0;
}
```

Hex View-1

看见j = 15 那我们就输入十五个字母试试先试试abcdefghigklmn

```
[14] Accepting connection from 10.211.55.2...
abcdefghigklmn
```


然后我们继续运行下程序

先解释下，在第一个循环当中其实最主要的就是对v9 进行一个赋值然后进到下一个函数，为了让其return 1 我们就要让其满足最后一个return 条件

sub_8048519 == 98

现在程序大概的思路清晰了，我们来解决sub_8048519这个函数到底是干什么的问题

查看v9 中的值和我们的输入进行对比

```
[stack]:FFF30420 db  6Eh ; n
[stack]:FFF30421 db  6Fh ; o
[stack]:FFF30422 db  70h ; p
[stack]:FFF30423 db  71h ; q
[stack]:FFF30424 db  72h ; r
[stack]:FFF30425 db  73h ; s
[stack]:FFF30426 db  74h ; t
[stack]:FFF30427 db  75h ; u
[stack]:FFF30428 db  76h ; v
[stack]:FFF30429 db  74h ; t
[stack]:FFF3042A db  78h ; x
[stack]:FFF3042B db  79h ; y
[stack]:FFF3042C db  7Ah ; z
[stack]:FFF3042D db  61h ; a
```

这和我们的输入一进行对比我们就能发现一一对应的关系，当然这过程当中我也试过所有可打印的ASCii码值最后发现了这个规律，读者可以尝试下，毕竟这比一字一句的去看那个长到爆炸的函数好，当然那句

```
if ( (unsigned __int8)v3 > 204u && v3 != 207 )
     v1 = 1;
```

可以发现这个v3转换是有范围的不然是进入不到下一个函数的，这个ida原来打开应该是-49 因为这个v3是个char类型所以有溢出

接下来我们既发现了 a —m 对应的是 n—z

那么这个函数就解决了

接下来进入下一个循环的

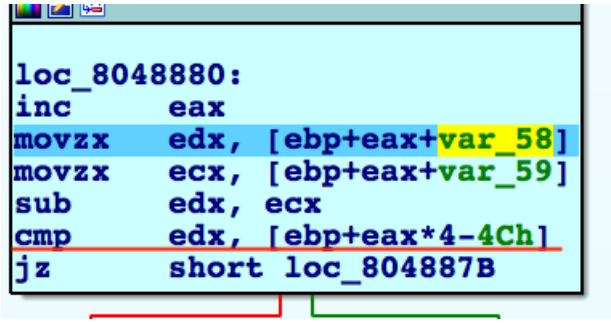
```
if ( (unsigned __int8)*(&v9 + v4) - (unsigned __int8)*(&v8 + v4) != *(&v11 + v4)
break;
```

可以发现这句话是一个关键因为为了保证我们的循环能够顺利运行到我们要的`return 1` 的位置那么我们就需要是`if` 语句中的一只为 `0`

那么我们再动态调试一波，看一下`v8` 的地址刚好就在`v9` 上面一个也就是说是

```
v9[I] - v9[I - 1] = v11[I]
```

那么`v11` 又是什么尼 我们来看下汇编



这个`ebp +eax*4-4ch` 也就是在这个地址的数每`4`位取一个值作为`v11` 还要考虑到`v11`为`int8` 范围是`-128～127`

```
OFFh
OFFh
OFFh
OFFh
 11h
   0
   0
   0
OF5h
OFFh
OFFh
OFFh
   3
   0
   0
   0
OF8h
OFFh
OFFh
OFFh
   5
   0
   0
   0
 OEh
   0
   0
   0
OFDh
OFFh
OFFh
OFFh
   1
   0
   0
   0
   6
   0
   0
   0
OF5h
OFFh
OFFh
OFFh
   6
   0
```


第一个数为FFh 因为int 为8位 所以 FF应为-1 二下面第四个是11h为17 依次类推，我这里还没有截取完全读者可以自己去截取下

接着v11数就出来了，而且我们知道了第一个数就是 'o'因为最后一个判断条件

接下来贴上代码

```
# /usr/bin/python3.6
# -*-coding:utf-8-*-

__Author__= 'Gtozju'

v11 = [-1, 17, -11,  3, -8, 5, 14, -3, 1, 6, -11, 6, -8, -10]


map = {'n':'a', 'o':'b', 'p':'c', 'q':'d', 'r':'e', 's':'f', 't':'g', 'u':'h', 'v':'i', 'w':'j', 'x':'k', '

key = ord('b')
flag = 'o'
for i in range(14):
    key += v11[i]
    flag += map[chr(key)]

print('flag{' + flag + '}')
```

这道题目就解决啦，总结下动态调试找规律是这个题目的简单方法，当然也有复杂一些一步步逆出来