从零开始：攻防世界_REVERSE_进阶区writeup

原创

qq_52842965 于 2021-09-26 21:44:39 发布 1122 收藏

分类专栏： 攻防世界 文章标签： c语言

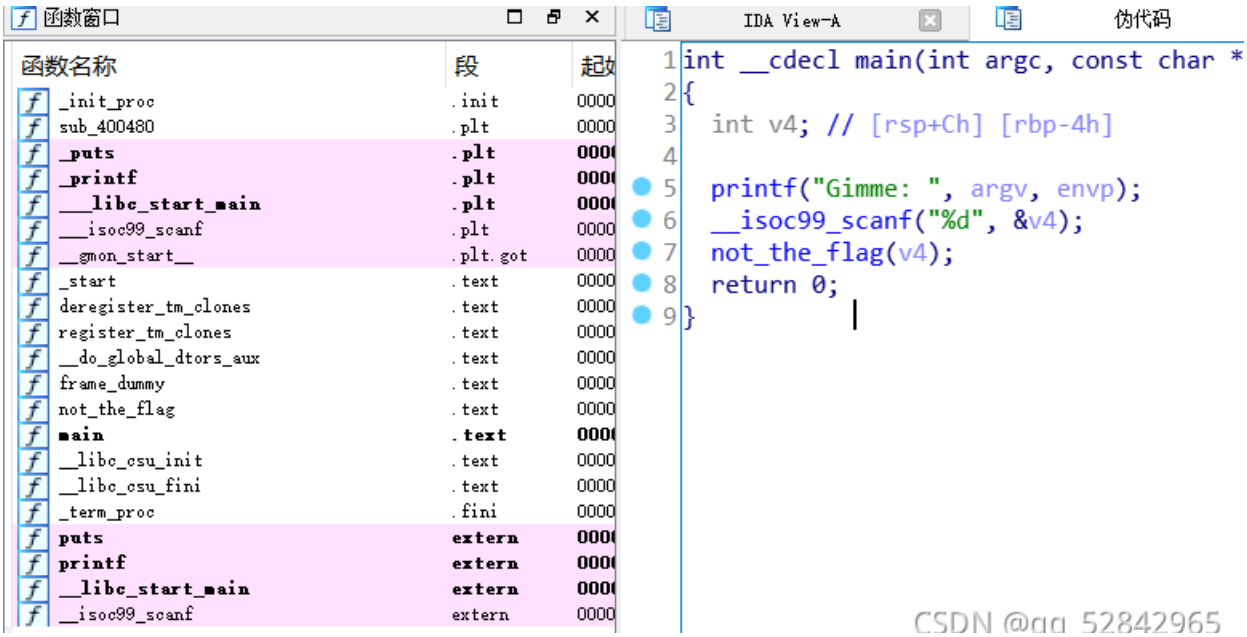
本文链接：https://blog.csdn.net/qq_52842965/article/details/120497535

1、answer_to_everything

把提供的附件下载出来用 PE 打开

提示是linux 的64位文件，也并无加壳，然后直接用ida64打开

打开后直接进入主函数，查看伪代码



进入主函数中，主函数的结构简单，直接查看 not_the_flag 函数

```
__int64 __fastcall not_the_flag(int a1)
{
  if ( a1 == '*' )
    puts("Cipher from Bill \nSubmit without any tags\n#kdudpeh");
  else
    puts("YOUSUCK");
  return 0LL;
}
```

其中有个 if 判断，输出的第一个内容表明flag没有任何特征
第二个内容，意思是你真差劲，即答案错误
我们注意到第一个文本后有一串特殊的字符串，可能为本题flag
kdudpeh
对于这一串字符，开始认为是本题flag，直接输入进去，错误，我们想起本题的题目描述

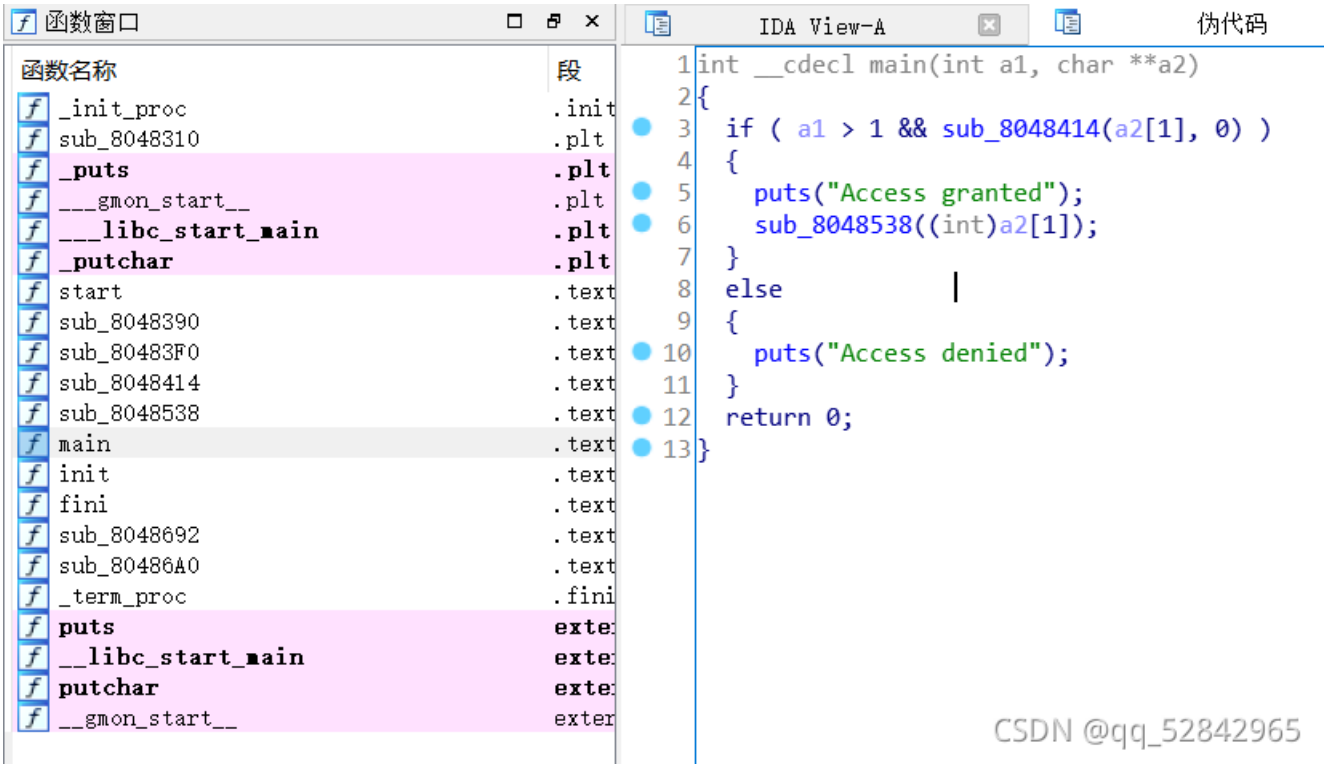
```
sha1 得到了一个神秘的二进制文件。寻找文件中的flag，解锁宇宙的秘密。注意：将得到的flag变为flag{XXX}形式提交。
```

注意中提示需要把得到的flag进行转变，猜测可能是需要进行某种加密，代码中并无进行何种加密的提示
题目描述中说到 sha1 这可能是种加密方式 百度后发现确实是一种加密方式，
采取在线加密后得到 80ee2a3fe31da904c596d993f7f1de4827c1450a
根据题目要求得到 flag为
flag{80ee2a3fe31da904c596d993f7f1de4827c1450a}

2、elrond32

先把提供的附件用PE打开，查看基本信息

为linux的32位ELF文件，直接ida32打开,查看主函数伪代码



基本函数只有sub_8048414和sub_8048538，以及if判断语句，若成立输出字样为准许进入
并执行函数，此时对题中唯一的两个函数进行分析
sub_8048538跟进查看代码

```
int __cdecl sub_8048538(int a1)
{
  int v2[33]; // [esp+18h] [ebp-A0h]
  int i; // [esp+9Ch] [ebp-1Ch]

  qmemcpy(v2, &unk_8048760, sizeof(v2));
  for ( i = 0; i <= 32; ++i )
    putchar(v2[i] ^ *(char *)(a1 + i % 8));
  return putchar(10);
}
```

代码中第一个函数qmemcpy作用简单来说就是把unk_8048760中的字符复制到v2中，
接着通过一个for循环把v2中的值以及a1中的相关数据进行异或运算后转换为字符输出，猜测输出的字符为flag，我们此时跟进

接着通过 for循环把v2中的值以及a1中的相关数据进行异或运算后转换为字符输出，猜测输出的字符为flag，我们此时跟进unk_8048760 去查找v2的内容

```
.rodata:08048760 unk_8048760     db  0Fh
.rodata:08048761                 db    0
.rodata:08048762                 db    0
.rodata:08048763                 db    0
.rodata:08048764                 db  1Fh
.rodata:08048765                 db    0
.rodata:08048766                 db    0
.rodata:08048767                 db    0
.rodata:08048768                 db    4
.rodata:08048769                 db    0
.rodata:0804876A                 db    0
.rodata:0804876B                 db    0
.rodata:0804876C                 db    9
.rodata:0804876D                 db    0
.rodata:0804876E                 db    0
.rodata:0804876F                 db    0
.rodata:08048770                 db  1Ch
.rodata:08048771                 db    0
.rodata:08048772                 db    0
.rodata:08048773                 db    0
.rodata:08048774                 db  12h
.rodata:08048775                 db    0
.rodata:08048776                 db    0
.rodata:08048777                 db    0
.rodata:08048778                 db  42h ; B
.rodata:08048779                 db    0
.rodata:0804877A                 db    0
.rodata:0804877B                 db    0
.rodata:0804877C                 db    9
.rodata:0804877D                 db    0
.rodata:0804877E                 db    0
.rodata:0804877F                 db    0
.rodata:08048780                 db  0Eh
```


这是提取数据得到v2的数组

```
.rodata:080487C8                 db  55h ; U
.rodata:080487C9                 db    0
.rodata:080487CA                 db    0
.rodata:080487CB                 db    0
.rodata:080487CC                 db    2
.rodata:080487CD                 db    0
.rodata:080487CE                 db    0
.rodata:080487CF                 db    0
.rodata:080487D0                 db  0Ch
.rodata:080487D1                 db    0
.rodata:080487D2                 db    0
.rodata:080487D3                 db    0
.rodata:080487D4                 db    8
.rodata:080487D5                 db    0
.rodata:080487D6                 db    0
.rodata:080487D7                 db    0
.rodata:080487D8                 db  41h ; A
.rodata:080487D9                 db    0
.rodata:080487DA                 db    0
.rodata:080487DB                 db    0
.rodata:080487DC                 db  0Ah
.rodata:080487DD                 db    0
.rodata:080487DE                 db    0
.rodata:080487DF                 db    0
.rodata:080487E0                 db  14h
.rodata:080487E1                 db    0
.rodata:080487E2                 db    0
```

导出数据

导出为

- hex 字符串(未空格)
- hex 字符串(间隔)
- 字符串文字
- C无符号字符数组(十六进制)
- C无符号字符数组(十进制)
- C变量初始化
- 原始字节

保存数据到剪贴板

预览

```
{
   15,    0,    0,    0,   31,    0,    0,    0,    4,    0,
    0,    0,    9,    0,    0,    0,   28,    0,    0,    0,
   18,    0,    0,    0,   66,    0,    0,    0,    9,    0,
    0,    0,   12,    0,    0,    0,   68,    0,    0,    0,
   13,    0,    0,    0,    7,    0,    0,    0,    9,    0,
    0,    0,    6,    0,    0,    0,   45,    0,    0,    0,
   55,    0,    0,    0,   89,    0,    0,    0,   30,    0,
    0,    0,    0,    0,    0,    0,   89,    0,    0,    0,
   15,    0,    0,    0,    8,    0,    0,    0,   28,    0,
    0,    0,   35,    0,    0,    0,   54,    0,    0,    0,
```

行:1 列:1

这时得到v2的值，只要再得到a1的值，就可以得到flag

我们根据sub_8048538函数的参数知道 a1 来自于传入的参数 (int)a2[1]，a2[1]又是sub_8048414函数的参数根进去分析这个函数，传入的参数a2=0，还有一个数组a1

```
signed int __cdecl sub_8048414(_BYTE *a1, int a2)
{
  signed int result; // eax

  switch ( a2 )
  {
    case 0:
      if ( *a1 == 105 )
        goto LABEL_19;
      result = 0;
      break;
    case 1:
      if ( *a1 == 101 )
        goto LABEL_19;
      result = 0;
      break;
    case 3:
      if ( *a1 == 110 )
        goto LABEL_19;
      result = 0;
      break;
    case 4:
      if ( *a1 == 100 )
        goto LABEL_19;
      result = 0;
      break;
    case 5:
      if ( *a1 == 97 )
        goto LABEL_19;
      result = 0;
      break;
    case 6:
      if ( *a1 == 103 )
        goto LABEL_19;
      result = 0;
      break;
    case 7:
      if ( *a1 == 115 )
        goto LABEL_19;
      result = 0;
      break;
    case 9:
      if ( *a1 == 114 )
LABEL_19:
        result = sub_8048414(a1 + 1, 7 * (a2 + 1) % 11);
      else
        result = 0;
      break;
    default:
      result = 1;
      break;
  }
  return result;
}
```

函数中是一个switch选择，其中加入了简单的递归

```
result = sub_8048414(a1 + 1, 7 * (a2 + 1) % 11);
```

我们分析一下switch语句，这个语句是以a2的值为参考，其中case有01345679和default组成，我们判断a2可能有8个值，并且与a1的值对应，只要得到a2的值，就可以知道a1的值，根据a2的递归运算，得到a2的值

```
#include<stdio.h>
int main()
{
 int a2=0;
 while(1)
 {
  printf("%d",a2);
  a2=7*(a2+1)%11;
  if(a2==2||a2==8)
  {
   break;
  }
 }
 return 0;
}
```

a2的值依次为07136594，则a1数组可以手动比对出来，然后得到数组a1的值，就可以得到flag

```
 int a1[]={105,115,101,110,103,97,114,100};
```

编写脚本输出flag

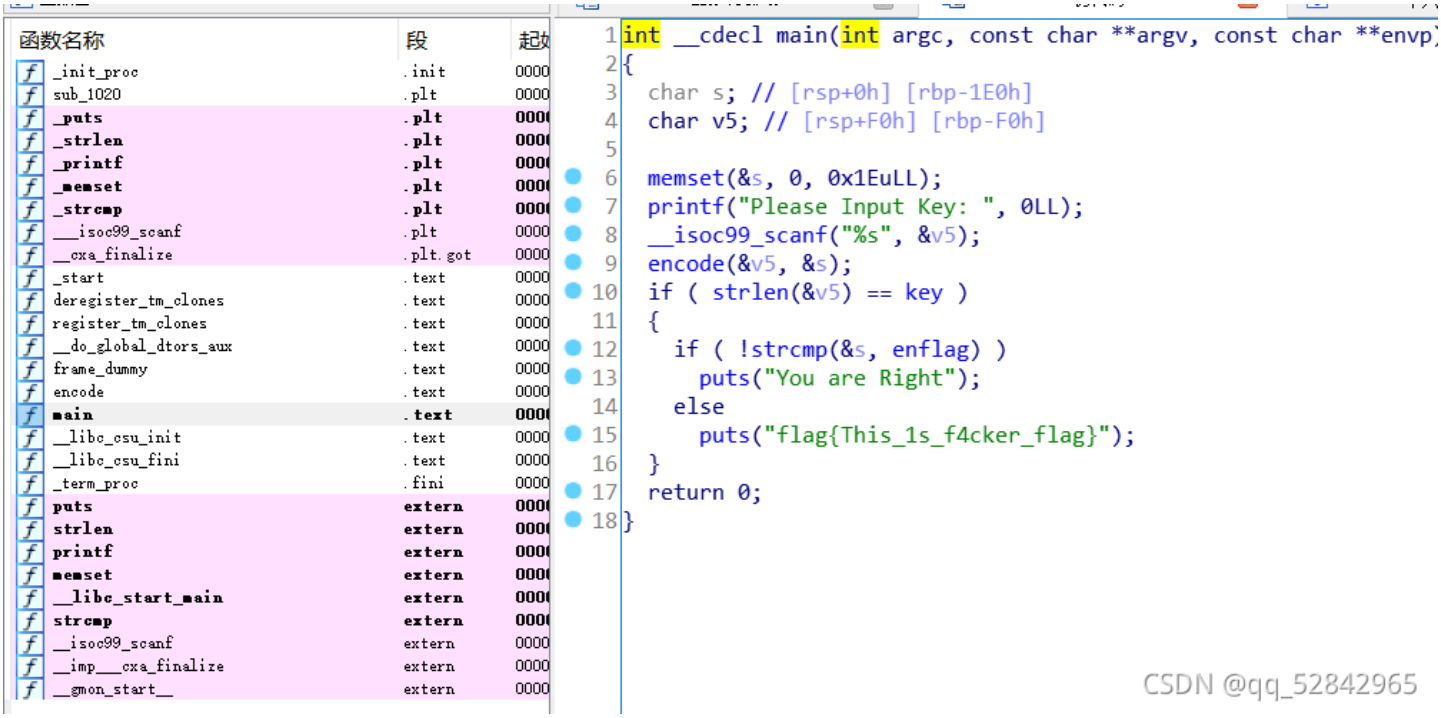
```
key=[105,115,101,110,103,97,114,100]
v2=[0x0F,0x1F,0x04,0x09,0x1C,0x12,0x42,0x09,0x0C,0x44,0x0D,0x07,0x09,0x06,0x2D,0x37,0x59,0x1E,0x00,0x59,0x0F,0x0
8,0x1C,0x23,0x36,0x07,0x55,0x02,0x0C,0x08,0x41,0x0A,0x14]
a=''
for i in range(33):
    a+=chr(v2[i]^key[(i%8)])
print(a)
```

得到flag为flag{s0me7hing_S0me7hinG_t0lki3n}

3、666

这是按照习惯性的用PE打开下载出来的附件

提示为 linux 64位的无加壳文件，直接用ida64打开
进入后直接查看主函数的伪代码



我们对主函数代码进行分析

```
memset(&s, 0, 0x1EuLL);
printf("Please Input Key: ", 0LL);
__isoc99_scanf("%s", &v5);
encode(&v5, &s);
if ( strlen(&v5) == key )
{
  if ( !strcmp(&s, enflag) )
    puts("You are Right");
  else
    puts("flag{This_1s_f4cker_flag}");
}
```

从第三行看起，输入字符串给 v5 ，下一行 encode加密 将得到的结果保存在 s
接着是 if 条件判断 v5的长度是否等于 key ，这时我们并不知道 key 的具体值，我们跟踪 key 查看 key 值为十六进制12h，即十进制18

```
                public key
key             dd 12h
```

此时得知我们输入的v5的长度为18
接着查看下一个if判断

```
if ( !strcmp(&s, enflag) )
    puts("You are Right");
  else
    puts("flag{This_1s_f4cker_flag}");
```

如果加密后的 s 和 enflag 相等，则证明输入的v5正确，即为正确的flag
跟进enflag查看其内容

```
enflag          db 'izwhroz""w"v.K".Ni',0
```

值为 izwhroz"“w"v.K”.Ni 这一列字符串，我们现在已经得到加密后的结果，只要根据加密方式，编写脚本就可以得到flag
此时跟进encode，查看加密方式

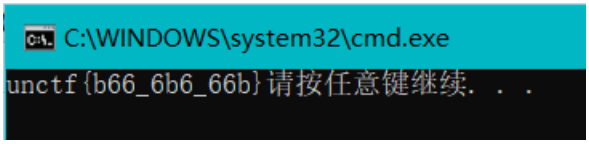
```
int __fastcall encode(const char *a1, __int64 a2)
{
  char v3[32]; // [rsp+10h] [rbp-70h]
  char v4[32]; // [rsp+30h] [rbp-50h]
  char v5[40]; // [rsp+50h] [rbp-30h]
  int v6; // [rsp+78h] [rbp-8h]
  int i; // [rsp+7Ch] [rbp-4h]

  i = 0;
  v6 = 0;
  if ( strlen(a1) != key )
    return puts("Your Length is Wrong");
  for ( i = 0; i < key; i += 3 )
  {
    v5[i] = key ^ (a1[i] + 6);
    v4[i + 1] = (a1[i + 1] - 6) ^ key;
    v3[i + 2] = a1[i + 2] ^ 6 ^ key;
    *(_BYTE *)(a2 + i) = v5[i];
    *(_BYTE *)(a2 + i + 1LL) = v4[i + 1];
    *(_BYTE *)(a2 + i + 2LL) = v3[i + 2];
  }
  return a2;
}
```

具体分析后发现v3,v4,v5 实际为函数中新创建的中间数组，具体加密方式为 for 循环中的代码
分析后实际是对a1中的每个数字进行异或运算和加减后得到加密后的字符再填入a2中，我们直接根据代码编写脚本
根据函数传参，a1就是s ，a2就是enflag ，将enflag转换为十六进制写入数组

```
#include<stdio.h>
int main()
{
 int a2[18]={0x69 ,0x7A ,0x77, 0x68, 0x72 ,0x6F, 0x7A, 0x22, 0x22 ,0x77 ,0x22,0x76 ,0x2E ,0x4B ,0x22, 0x2E ,0x4E
 ,0x69 };
 for(int i=0;i<18;i+=3)
 {
  a2[i]=(a2[i]^18)-6;
  a2[i+1]=(a2[i+1]^18)+6;
  a2[i+2]=(a2[i+2]^18^6);
 }
 for(int i=0;i<18;i++)
 {
  printf("%c",a2[i]);
 }
 return 0;
}
```

得到flag为



unctf{b66_6b6_66b}