ACTF2020 writeup

原创

DrumWashingMachine 于 2020-06-20 00:21:55 发布 394 收藏 1

分类专栏： CTF 文章标签： 密码学 反编译 信息安全

本文链接：https://blog.csdn.net/qq_46365538/article/details/106866373

ACTF2020 writeup

Pwm

- 不会

Reverse

> 题目名称 easyalgorithm
>
> FLAG: ACTF{Oolong_milk_tea}
>
> 大概就先把文件拖进IDA里面反编译一下，看反编译出来的c代码大概就知道是首先判断输入是否为ACTF{}，然后将花括弧中的字符串赋给dest，然后通过4006A6和400792两个函数对dest进行加密，然后将得到的密文和v3后的连续地址上的asc码进行比较

```
unsigned __int64 sub_4008B3()
{
  signed int i; // [rsp+Ch] [rbp-554h]
  int v2; // [rsp+10h] [rbp-550h]
  char v3; // [rsp+20h] [rbp-540h]
  char v4; // [rsp+21h] [rbp-53Fh]
  char v5; // [rsp+22h] [rbp-53Eh]
  char v6; // [rsp+23h] [rbp-53Dh]
  char v7; // [rsp+24h] [rbp-53Ch]
  char v8; // [rsp+25h] [rbp-53Bh]
  char v9; // [rsp+26h] [rbp-53Ah]
  char v10; // [rsp+27h] [rbp-539h]
  char v11; // [rsp+28h] [rbp-538h]
  char v12; // [rsp+29h] [rbp-537h]
  char v13; // [rsp+2Ah] [rbp-536h]
  char v14; // [rsp+2Bh] [rbp-535h]
  char v15; // [rsp+2Ch] [rbp-534h]
  char v16; // [rsp+2Dh] [rbp-533h]
  char v17; // [rsp+2Eh] [rbp-532h]
  char v18; // [rsp+30h] [rbp-530h]
  char v19; // [rsp+31h] [rbp-52Fh]
  char v20; // [rsp+32h] [rbp-52Eh]
  char v21; // [rsp+33h] [rbp-52Dh]
  char v22; // [rsp+34h] [rbp-52Ch]
```

```
_BYTE v23[3]; // [rsp+35h] [rbp-52Bh]
char v24; // [rsp+44h] [rbp-51Ch]
char s; // [rsp+50h] [rbp-510h]
char dest[1032]; // [rsp+150h] [rbp-410h]
unsigned __int64 v27; // [rsp+558h] [rbp-8h]

v27 = __readfsqword(0x28u);
v2 = 1179927361;
v3 = -60;
v4 = -59;
v5 = -119;
```

000008B3 sub_4008B3:26 (4008B3)

IDA… | Pse… | Hex… | Str… | Enums | Imp… | Exp…

```
34  v4 = -59;
35  v5 = -119;
36  v6 = -118;
37  v7 = -52;
38  v8 = -100;
39  v9 = 24;
40  v10 = 68;
41  v11 = 8;
42  v12 = 10;
43  v13 = 106;
44  v14 = 41;
45  v15 = -46;
46  v16 = -28;
47  v17 = 46;
48  memset(&s, 0, 0x100uLL);
49  memset(&v18, 0, 0x15uLL);
50  memset(dest, 0, 0x400uLL);
51  printf("Please input:", 0LL);
52  __isoc99_scanf((__int64)"%s", (__int64)&v18);
53  if ( v18 == 65 && v19 == 67 && v20 == 84 && v21 == 70 && v22 == 123 && v24
54  {
55    memcpy(dest, v23, 0xFuLL);
56    sub_4006A6((__int64)&s, (__int64)&v2, 4);
57    sub_400792((__int64)&s, (__int64)dest, 0xFuLL);
58    for ( i = 0; i <= 14; ++i )
59    {
60      if ( dest[i] != *(&v3 + i) )
61      {
62        printf("Error!");
63        return __readfsqword(0x28u) ^ v27;
64      }
65    }
66    printf("Great!", dest);
67  }
68  return __readfsqword(0x28u) ^ v27;
```

000008DE sub_4008B3:34 (4008DE)

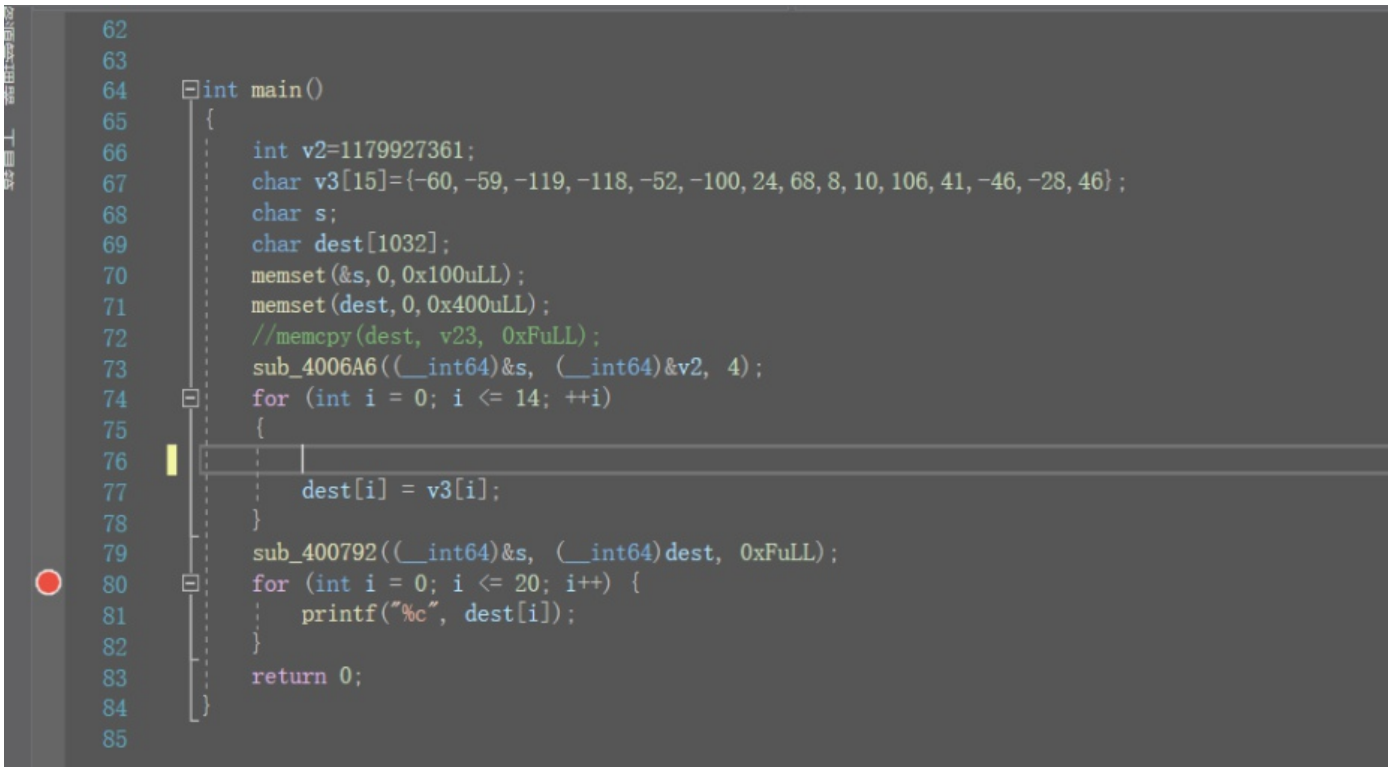
然后进入792函数

```
unsigned __int64 __fastcall sub_400792(__int64 a1, __int64 a2, unsigned __int64 a3)
{
  unsigned int v3; // eax
  char v4; // ST27_1
```

```
unsigned __int64 result; // rax
signed int v6; // [rsp+28h] [rbp-10h]
int i; // [rsp+2Ch] [rbp-Ch]
unsigned __int64 v8; // [rsp+30h] [rbp-8h]

v8 = 0LL;
v6 = 0;
for ( i = 0; ; *(_BYTE *)(a2 + v8++) ^= *(_BYTE *)((unsigned __int8)(*(_BYTE *)(v6 + a1) + *(_BYTE *)(i + a1)) + a1) )
{
  result = v8;
  if ( v8 >= a3 )
    break;
  v6 = (unsigned __int8)(((unsigned int)((v6 + 1) >> 31) >> 24) + v6 + 1) - ((unsigned int)((v6 + 1) >> 31) >> 24);
  v3 = (unsigned int)((i + *(unsigned __int8 *)(v6 + a1)) >> 31) >> 24;
  i = (unsigned __int8)(v3 + i + *(_BYTE *)(v6 + a1)) - v3;
  v4 = *(_BYTE *)(v6 + a1);
  *(_BYTE *)(a1 + v6) = *(_BYTE *)(i + a1);
  *(_BYTE *)(a1 + i) = v4;
}
return result;
}
```

发现对密文主要进行加密的就是选中的那一条语句，且发现异或运算的逆运算为异或运算所以将v3后面的先赋给dest然后然dest执行一遍792函数即可



```
int main()
{
    int v2=1179927361;
    char v3[15]={-60, -59, -119, -118, -52, -100, 24, 68, 8, 10, 106, 41, -46, -28, 46};
    char s;
    char dest[1032];
    memset(&s, 0, 0x100uLL);
    memset(dest, 0, 0x400uLL);
    //memcpy(dest, v23, 0xFuLL);
    sub_4006A6((__int64)&s, (__int64)&v2, 4);
    for (int i = 0; i <= 14; ++i)
    {

        dest[i] = v3[i];
    }
    sub_400792((__int64)&s, (__int64)dest, 0xFuLL);
    for (int i = 0; i <= 20; i++) {
        printf("%c", dest[i]);
    }
    return 0;
}
```

Crypto

题目名称：Column Permutation Cipher

FLAG:忘了

好像是矩阵转置吧，大概就把字符串先读入然后依次枚举j，输出str[i%j]=0 1 2 3 4…j;

```
#include<bits/stdc++.h>
using namespace std;
const int maxn=1e5+10;
char s[maxn];
```

```
int main()
{
    gets(s);
    int t=5;
    for(;t<51;t++){
        int tt=0;
        printf("%d\n",t);
        for(int t1=0;t1<=t;t1++){
            for(int i=0;i<strlen(s);i++){
                if(i%t==t1){
                    printf("%c",s[i]);
                }
            }
        }
        puts("");puts("");
    }
    return 0;
}
```

题目名称：我的密码本

FLAG：忘了

emmmm大概替换密码，本来想用字频分析的，但是中间几个连续的太明显了，一看就是I have a dream,接着把每个符号换成对应的字母就行了

题目名称：[bomb or boom](http://actf2020.csuaurora.org/challenges#bomb or boom)

FLAG：也忘了

emmm5个密码只用破译四个就能拿到最后密码，明显是门限方案，结合题目名称，应该是bloom门限

密码1：培根密码

密码2：盲文

密码3：音符

密码4：放进浏览器f12即可

密码5：beautifulfxxk

```
from Crypto.Util.number import long_to_bytes
#from Cryptodome.Util.number import *
a1 =2891369521230520600
d1 =5539166121540472709
a2 =5485400237604727152
d2 =9993590208169240051
a3 =10305113992248275270
d3 =23524210813213316809
a4 =63558232650391605454
d4 =134070550878039878083

dd = d1*d2*d3*d4
t1 = pow(dd//d1,d1-2,d1)
assert(t1*d2*d3*d4%d1 == 1)
t2 = pow(dd//d2,d2-2,d2)
assert(t2*d1*d3*d4%d2 == 1)
t3 = pow(dd//d3,d3-2,d3)
```

```
assert(t3*d2*d1*d4%d3 == 1)
t4 = pow(dd//d4,d4-2,d4)
assert(t4*d1*d2*d3%d4 == 1)
s = a1*t1*d2*d3*d4+a2*t2*d1*d3*d4+a3*t3*d1*d2*d4+a4*t4*d1*d2*d3
p = 80804238007977405688648566160504278593148666302626415149704905628622876270862865768337953835725801963142
685182510812938072115996355782396318303927020705623120652014080032809421180400984242061592520733710243483947
230962631945045134540159517488288781666622635328316972979183761952842010806304748313326215619695085380586052
550443025074501971925005072999275628549710915357400946408857
s %= dd
print(s)
s %= p
print(long_to_bytes(s))
s1=long_to_bytes(s)
string = str(s1, 'utf-8')
```

题目名称：[naive encryption](http://actf2020.csuaurora.org/challenges#naive encryption)

FLAG：也忘了

脚本没了QAQ，大概就是一个很简单的加密，本来应该用逆元来干的，脑抽了，后来索性就把1-1000赋给一个数组然后把这个数组送进脚本加密，最后根据密文从这个数组中搜索相同的值，那个值的数组下标就是对应的明文

[外链图片转存失败,源站可能有防盗链机制,建议将图片保存下来直接上传(img-3QjQUPBe-1595158567300)(D:\桌面\TIM图片20200605231507.png)]

题目名称：[naive rsa](http://actf2020.csuaurora.org/challenges#naive rsa)

FLAG：也忘了（梅 开 六 度

大概就是一个知道n，e和p%q的rsa加密，数字都比较大，就通过两个方程来联立

n=p*q

p=i*q+r r=p%q

大概联立之后是一个一元二次方程，求根公式得到Δ表达式，然后枚举i，求i=？Δ开方为整数

计算i的脚本

```
import math
from Crypto.Util.number import *
import random
import gmpy2
#from flag import FLAG
from hashlib import sha512
# k=1004566198076519599602210506731394352706714405801310206329305430708682949563948700333927208488048733764810655777562050736705748648879470498998723741270
# N=617551474948504941337305307137757553094532163416400907262288565420847153730009500576696377857741919958172124604052378548332400164704746797721481405003094358713376385202739966008687081558013899017520442554287644168739023335616912949583960335826502791123976212645405630550392946705302484936623679824847569620 7

k=166437827376467256148149729255164233694132787599408922364771799801391783230342242836164503041151544482141980377309613006088294955552338574867558385322 3
N=575409410485601592096315531552969450375296870137032371395416724690686355802265691870874938944755403817816868965210223953610000373235920940084167551443302298042738929832814849351887330529045059585413100483032634509865109120092400651755934690365300943781461723302773438884722939492153162456567789430720138029 9
enc=266136474080728785411018107173926631772846249012746699567650691009361033826084120305006962600513119828684348258650345526957655513836554173352462873353490593132963651078460394495900025848215857842151100198742277932597155272200315592857945615830054237586557716482948717739413654372472441736547676542476098466 9
```

```
def main():
    # w1=gmpy2.mpz(1)
    # w2=gmpy2.mpz(1)
    # print(w1==w2)

    for i in range(1,N):
        tmp=pow(k,2)+4*i*N
        t2=gmpy2.mpz(tmp)
        t1=gmpy2.isqrt(tmp)
        t1=gmpy2.mpz(t1)
        if(t1*t1==t2):
            print(i)
            print(t1*t1)
            print(t2)
            tmp1=(-k+t1)
            print(tmp1%(2*i)==0)
            tmp1=tmp1//(2*i)
            q=tmp1
            print(q)
            print("\n")
            p=N//q
            print(p)
            print(p*q==N)
            break


if __name__ == "__main__":
    main()
```

破解明文的脚本

```
import math
from Crypto.Util.number import *
import random
#from flag import FLAG
from hashlib import sha512
import gmpy2

#enc=619131807053341983712996592129731844946129596361804867448023818077266884366071907777148396343131913424753531469667253075830996004711559531521437680038739168021990771840146116736026028722046579674483003435680909352182226754031166260584553606550138319245292347367443791048564768901364057882741366609043947257
#p%q=166437827376467256148149729255164233694132787599408922364771799801391783230342242836164503041151544482141980377309613006088294955552338574867558385322
3
N=57540941048560159209631553155296945037529687013703237139541672469068635580226569187087493894475540381781686896521022395361000037323592094008416755144330229804273892983281484935188733052904505958541310048303263450986510912009240065175593469036530094378146172330277343888472293949215316245656778943072013802
99
enc=26613647408072878541101810717392663177284624901274669956765069100936103382608412030500696260051311982868434825865034552695765551383655417335246287335349059313296365107846039449590002584821585784215110019874227793259715527220031559285794561583005423758655771648294871773941365437247244173654767654247609846
69
e=65537
#N=61755147494850494133730530713775755309453216341640090726228856542084715373000950057669637785774191995817212460405237854833240016470474679772148140500309435871337638520273996600868708155801389901752044255428764416873902333561691294958396033582650279112397621264540563055039294670530248493662367982484756962
07
#cmp='actf'
q=32009136331594067007397116196763568288303595330843359187929170568996095306744796380232613472868608500486216665703022217129964309206167923027920290358479


p=17976411625878628866213850065307738994271245158271052147551634527069325255636814170955679808040260386244508466000186411415022824808207872142183570473642376581
```

```
846609018641414592282480829787314218357947364376581

def main():
    phi_n = (p-1)*(q-1)
    d = gmpy2.invert(e,phi_n)
    c=enc
    m=pow(c,d,N)
    print(m)
    m=int(m)
    # a=input()
    print(long_to_bytes(m))


if __name__ == "__main__":
    main()
```

Mise

- 题目名称：签到
- 进入公众号，即可获得宝贵的flag一枚（狗头保命