hgame2025 WriteUp

Crypto

Ancient Recall

一个模拟塔罗牌占卜的程序，VALUE经过了250次Fortune_wheel得到final_Value，把Fortune_wheel逆向就好了。

```
def Reverse_wheel(FATE):
    FATEd = [
        (FATE[0] - FATE[1] + FATE[2] - FATE[3] + FATE[4]) // 2,
        (FATE[0] + FATE[1] - FATE[2] + FATE[3] - FATE[4]) // 2,
        (-FATE[0] + FATE[1] + FATE[2] - FATE[3] + FATE[4]) // 2,
        (FATE[0] - FATE[1] + FATE[2] + FATE[3] - FATE[4]) // 2,
        (-FATE[0] + FATE[1] - FATE[2] + FATE[3] + FATE[4]) // 2,
    ]
    return FATEd


final_Value = [
    25329519520662917748904983691141959172407947049182105205710670853114746750 19,
    25329519520662917748903276660741003578980230131054431788812947003815097952 70,
    25329519520662917748905544592872766049031303158592585441730683769670723357 30,
    25329519520662917748908653282415328853915101626115345140144091742842991390 15,
    25329519520662917748908306626081341560179463763099899341758339139211426093 34,
]

Value = final_Value
for i in range(250):
    Value = Reverse_wheel(Value)

initial_FATE = []
for i in Value:
    if i < 0:
        initial_FATE.append("re-" + tarot[(i ^ -1) % 78])
    else:
        initial_FATE.append(tarot[i % 78])

FLAG = ("hgame{" + "&".join(initial_FATE) + "}").replace(" ", "_")
print(FLAG)
# hgame{re-The_Moon&re-The_Sun&Judgement&re-Temperance&Six_of_Cups}
```

Intergalactic Bound

首先这是Twisted Hessian curves

```
a*x^3+y^3+1=dxy
```

方程代入P，Q两点联立消去d可以计算出a，再计算出d，这样参数就齐了。

然后用Pohlig_Hellman方法计算x即可

```
from Crypto.Util.number import *
from Crypto.Cipher import AES
import hashlib

p = 550990553680539486102767863 01
G = (196634467629629276330379267 40, 350744124309156560717770153 20)
Q = (268051376735366358258843301 80, 263768331126093094759511868 83)
x1, y1 = G
x2, y2 = Q

a = (
    (y1 * y2 * (y2**2 * x1 - y1**2 * x2) + (x1 * y1 - x2 * y2))
    * inverse(x1 * x2 * (x1**2 * y2 - y1 * x2**2), p)
    % p
)
d = (a * G[0] ** 3 + G[1] ** 3 + 1) % p * inverse(G[0] * G[1], p) % p

ciphertext =
b"k\xe8\xbe\x94\x9e\xfc\xe2\x9e\x97\xe5\xf3\x04'\x8f\xb2\x01T\x06\x88\x04\xeb3Jl\
xdd Pk$\x00:\xf5"

R.<x,y,z> = Zmod(p)[]
cubic = a * x^3 + y^3 + z^3 - d*x*y*z
E = EllipticCurve_from_cubic(cubic,morphism=True)
G = E(G)
Q = E(Q)
E_ord = G.order()

def Pohlig_Hellman(n,P,Q):
    factors, exponents = zip(*factor(n))
    primes = [factors[i] ^ exponents[i] for i in range(len(factors))]
    print(primes)
    dlogs = []
    for fac in primes:
        t = int(int(P.order()) // int(fac))
        dlog = discrete_log(t*Q,t*P,operation="+")
        dlogs += [dlog]
        print("factor: "+str(fac)+", Discrete Log: "+str(dlog)) #calculates
discrete logarithm for each prime order
    num2 = crt(dlogs,primes)
    return num2

x = Pohlig_Hellman(E_ord,G,Q)
print(f"x={x}")
# x=263317779882935292158320673 6

key = hashlib.sha256(str(x).encode()).digest()
cipher = AES.new(key, AES.MODE_ECB)
flag = cipher.decrypt(ciphertext)
print(f"flag={flag}")
# flag=b'hgame{NOth1ng_bu7_up_Up_UP!}\x04\x04\x04\x04'
```

Reverse

SignUp

如图，程序将flag去掉hgame{后复制到Destination后用sub_140014D3检查

```
j___CheckForDebuggerJustMyCode(&unk_1400C70A3, argv, envp);
j_memset(Str1, 0, 0x40uLL);
sub_140002941("password:");
sub_14000180C("%44s", Str1);
if ( (unsigned int)sub_140002EC3() && (unsigned int)sub_1400034F9() )
{
  if ( !j_strncmp(Str1, "hgame{", 6uLL)
    && (j_strncpy(Destination, &Str1[6], 0x24uLL), (unsigned int)sub_1400014D3(Destination)) )
  {
    j_puts("right");
  }
  else
  {
    j_puts("wrong");
  }
}
```

用sub_1400016EF加密后与byte_1400BB880比较，相关参数有unk_1409BB2A0 unk_1499BB889。

```
__int64 __fastcall sub_140008820(__int64 a1, __int64 a2, __int64 a3)
{
  int i; // [rsp+24h] [rbp+4h]

  j___CheckForDebuggerJustMyCode(&unk_1400C70A3, a2, a3);
  sub_1400016EF(a1, &unk_1400BB2A0, &unk_1400BB880);
  for ( i = 0; i < 36; ++i )
  {
    if ( *(unsigned __int8 *)(a1 + i) != byte_1400BA000[i] )
      return 0LL;
  }
  return 1LL;
}
```

加密函数如图，已逆向（验证过了）

```
void encrypt(_DWORD *a1, __int64 a2, __int64 a3)
{
    unsigned int v4, v5, v6, v8;
    int v9, i;

    v6 = 0;
    for (v8 = 11; v8 > 0; v8--)
    {
        v6 += *(_DWORD *)(a3 + 4LL * (v8 % 4));
        v9 = (v6 >> 2) & 3;
        for (i = 0; i < 9; ++i)
        {
            v4 = a1[(i + 1) % 9];
            v5 = a1[(i + 8) % 9];
            a1[i] += ((v5 ^ *(_DWORD *)(a2 + 4LL * (v9 ^ i & 3))) + (v4 ^ v6)) ^
(((16 * v5) ^ (v4 >> 3)) + ((4 * v4) ^ (v5 >> 5)));
        }
    }
}
```

```
void decrypt(_DWORD *a1, __int64 a2, __int64 a3)
{
    unsigned int v4, v5, v6, v8;
    int v9, i;

    v6 = 0;
    for (v8 = 1; v8 < 12; v8++)
    {
        v6 -= *(_DWORD *)(a3 + 4LL * (v8 % 4));
        v9 = (v6 >> 2) & 3;
        for (i = 8; i >= 0; --i)
        {
            v4 = a1[(i + 1) % 9];
            v5 = a1[(i + 8) % 9];
            a1[i] -= ((v5 ^ *(_DWORD *)(a2 + 4LL * (v9 ^ i & 3))) + (v4 ^ v6)) ^
(((16 * v5) ^ (v4 >> 3)) + ((4 * v4) ^ (v5 >> 5)));
        }
    }
}
```

接下来就是unk_1409BB2A0 unk_1499BB889怎么找

追踪前者找到这个函数，居然用整个main的代码段作为数据加密，太疯狂了。

于是我把整个main代码段拿过来计算:

得到a2[4] = {2544000949, 3782569402, 2705540682, 1519331407}

```
__int64 __fastcall sub_140008670(__int64 a1, __int64 a2, __int64 a3)
{
  char *v4; // [rsp+48h] [rbp+28h]
  int i; // [rsp+64h] [rbp+44h]

  j___CheckForDebuggerJustMyCode(&unk_1400C70A3, a2, a3);
  v4 = (char *)j_j_j__malloc_base(0x10000uLL);
  j_memset(v4, 0, 0x10000uLL);
  j_memcpy(v4, main, 0x10000uLL);
  sub_1400011D6();
  for ( i = 0; i < 4; ++i )
    dword_1400BB2A0[i] = sub_140001AB4(&v4[0x4000 * i], 0x4000LL);
  return 1LL;
}
```

追踪后者找到这个函数，是个反调试程序，正常运行下Dr0~Dr3应该都是0，推定a3[4] = {0, 0, 0, 0}

```
__int64 __fastcall sub_14000F730(__int64 a1, __int64 a2, __int64 a3)
{
  __int64 result; // rax
  HANDLE CurrentThread; // rax
  LPCONTEXT lpContext; // [rsp+28h] [rbp+8h]

  j___CheckForDebuggerJustMyCode(&unk_1400C70A3, a2, a3);
  lpContext = (LPCONTEXT)VirtualAlloc(0LL, 0x4D0uLL, 0x1000u, 4u);
  if ( lpContext )
  {
    sub_1400034FE(lpContext, 1232LL);
    lpContext->ContextFlags = 1048592;
    CurrentThread = GetCurrentThread();
    if ( GetThreadContext(CurrentThread, lpContext) )
    {
      qword_1400BB880[0] = lpContext->Dr0;
      qword_1400BB880[1] = lpContext->Dr1;
      qword_1400BB880[2] = lpContext->Dr2;
      qword_1400BB880[3] = lpContext->Dr3;
      if ( qword_1400BB880[0] || qword_1400BB880[1] || qword_1400BB880[2] || (result = 24LL, qword_1400BB880[3]) )
      {
        j_puts("Debug error.");
        j_exit(0);
      }
    }
    else
    {
      return 0LL;
    }
  }
  else
  {
    j_puts("VirtualAlloc failed.");
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <malloc.h>

#define _BYTE unsigned char
#define _DWORD unsigned int

void encrypt(_DWORD *a1, __int64 a2, __int64 a3)
{
    unsigned int v4, v5, v6, v8;
    int v9, i;

    v6 = 0;
    for (v8 = 11; v8 > 0; v8--)
    {
        v6 += *(_DWORD *)(a3 + 4LL * (v8 % 4));
        v9 = (v6 >> 2) & 3;
        for (i = 0; i < 9; ++i)
        {
            v4 = a1[(i + 1) % 9];
            v5 = a1[(i + 8) % 9];
            a1[i] += ((v5 ^ *(_DWORD *)(a2 + 4LL * (v9 ^ i & 3))) + (v4 ^ v6)) ^
(((16 * v5) ^ (v4 >> 3)) + ((4 * v4) ^ (v5 >> 5)));
        }
    }
}

void decrypt(_DWORD *a1, __int64 a2, __int64 a3)
{
    unsigned int v4, v5, v6, v8;
    int v9, i;

    v6 = 0;
    for (v8 = 1; v8 < 12; v8++)
    {
```

```
        v6 -= *(_DWORD *)(a3 + 4LL * (v8 % 4));
        v9 = (v6 >> 2) & 3;
        for (i = 8; i >= 0; --i)
        {
            v4 = a1[(i + 1) % 9];
            v5 = a1[(i + 8) % 9];
            a1[i] -= ((v5 ^ *(_DWORD *)(a2 + 4LL * (v9 ^ i & 3))) + (v4 ^ v6)) ^
(((16 * v5) ^ (v4 >> 3)) + ((4 * v4) ^ (v5 >> 5)));
        }
    }
}

int main()
{
    char s[] = {
        0x23, 0xEA, 0x50, 0x30, 0x00, 0x4C, 0x51, 0x47,
        0xEE, 0x9C, 0x76, 0x2B, 0xD5, 0xE6, 0x94, 0x17,
        0xED, 0x2B, 0xE4, 0xB3, 0xCB, 0x36, 0xD5, 0x61,
        0xC0, 0xC2, 0xA0, 0x7C, 0xFE, 0x67, 0xD7, 0x5E,
        0xAF, 0xE0, 0x79, 0xC5};

    _DWORD a2[4] = {2544000949, 3782569402, 2705540682, 1519331407};
    _DWORD a3[4] = {0, 0, 0, 0};

    decrypt((_DWORD *)s, a2, a3);
    // 3fe4722c-1dbf-43b7-8659-c1c4a0e42e4d
    // hgame{3fe4722c-1dbf-43b7-8659-c1c4a0e42e4d}
    for (int i = 0; i < 36; i++)
    {
        printf("%c", s[i]);
    }
    printf("\n");

    return 0;
}
```