Week3-ChenMoFeiJin

RE

FAKE

描述

你能算出来吗?

密码：really

解题思路

用 IDA 打开 FAKE 文件，查看其 main 函数，发现一个用于校验的函数 `sub_401216`

```
__int64 __fastcall main(int a1, char **a2, char **a3)
{
  int v4[36]; // [rsp+0h] [rbp-D0h] BYREF
  char s[60]; // [rsp+90h] [rbp-40h] BYREF
  int i; // [rsp+CCh] [rbp-4h]

  puts("Give me your true flag:");
  __isoc99_scanf("%50s", s);
  if ( strlen(s) != 36 )
  {
    puts("Wrong length.");
    exit(0);
  }
  for ( i = 0; i <= 35; ++i )
    v4[i] = s[i];
  if ( (unsigned int)sub_401216(v4) == 1 )
    puts("Ohhhhhhhhhh!");
  else
    puts("Wrong flag. Keep looking!");
  return 0LL;
}
```

查看后发现是一段类似于矩阵乘法的运算，通过下面的代码解得 Flag 为 `hgame{@_FAKE_flag!-do_YOu_know_SMC?}`，是个假的Flag

```
A = numpy.mat([[29, -37, -35, -53, -19, -14, 11, 17, -48, 9, -99, 53, -20, 48, 90, 66, -49, -42, -67, 70, -33, -65, -70, -23, 89, 88, -56, 1, -50, 30, -7, 26, 63, -58, -68, 67],
               [-40, -20, 13, 70, 57, 30, -17, 41, -56, 90, 62, 15, -13, -63, 1, -57, -47, -92, 64, -79, 42, 90, 60, 57, 71, -41, 5, -91, 5, -5, 0, 56, -63, 47, 67, 10],
               [7, 49, -41, 59, -61, 24, 36, 2, -27, 53, -53, 93, 10, 28, -77, -93, 29, 12, 5, 77, 50, 81, 54, 34, 69, -87, 73, -99, 28, 58, -100, -5, -13, 77, 74, 88],
```

```
                [-2, -44, -61, -99, 51, 13, -61, -77, 23, 8, -52, 86, -96,
51, -92, 43, -77, 58, -100, -46, -46, 53, 31, -17, -9, 2, 5, 63, 8, -61, 94,
-17, 75, 81, -76, -52],
                [91, 64, 3, 59, -25, 96, 61, -44, 69, -69, 80, 40, 81, 17,
-29, 9, 9, 23, 32, 60, -62, -35, 78, -9, 99, -75, -69, -36, 68, 70, 69, 56,
-43, -81, 40, 37],
                [-3, 93, -89, 20, -64, -39, 90, -41, -20, 96, 48, 79, -61,
-86, -56, -56, 47, 61, 48, 54, 50, -63, -97, 51, 78, -34, -79, -86, -76, 66,
6, 32, 51, -13, -96, -36],
                [-13, -45, -25, 11, -36, 91, 64, 98, -46, 83, -23, -66, 17,
83, -33, 96, -74, -73, 31, 83, 77, -95, 37, -32, -27, 35, -65, -60, -74, -5,
-26, -9, 99, 76, 22, -44],
                [-1, 76, -84, -88, -36, -83, 64, -18, 82, 19, -79, -26, 84,
86, -51, 89, 55, 69, -11, 72, 70, 24, -36, 9, 42, 32, -93, 69, -8, 60, -99,
-57, -34, 78, 71, 9],
                [86, -60, -54, -2, -68, -51, 75, -58, -70, 94, 79, -20, 29,
82, 57, -80, 26, -31, 88, -59, 25, -6, 12, -18, -62, -57, 89, -82, 28, -16,
50, 94, 65, -86, -62, -66],
                [-63, 74, -11, 38, -88, 31, -46, 5, 72, 87, 30, 33, 13, -30,
31, -3, -59, -11, 83, 47, -91, 37, -75, 77, -7, -56, 14, -97, 86, 72, -24,
-7, -73, 48, -49, 53],
                [49, -27, 84, -52, -91, 63, -97, -100, -81, -14, -70, 70,
-89, -95, -49, 14, 81, 59, -96, -5, -74, 3, -85, 59, 28, 26, -77, -25, -66,
-94, 77, 24, -7, 20, -66, -24],
                [21, 27, 60, 5, -58, 30, 34, 2, 35, -1, 69, 42, 10, -19, 24,
83, 1, -2, 69, 70, 33, 22, -53, 55, 55, -69, -91, -31, 98, -63, -60, 60, 68,
84, 28, 53],
                [-96, -60, 33, -96, 32, -38, 5, 97, 50, -81, 76, 60, -68,
-92, 14, -61, 48, 34, 64, 38, 89, -96, -88, -59, 9, 98, -8, -97, 79, 90, 1,
15, -17, 54, 70, -80],
                [-50, -30, 73, -7, -2, -15, -35, -7, 53, 27, -82, 32, -92,
33, 71, -88, -45, 49, -16, -5, -38, 12, -98, -22, -29, 60, -28, -64, -98,
36, -57, 57, -58, 68, 79, -48],
                [-94, -5, -12, 12, -7, -56, 49, 17, 91, 95, 13, -22, 19, -23,
60, 91, 58, -78, -43, -85, 29, -31, -79, -91, -30, 54, 59, 71, -86, -36,
-20, 4, 18, -69, 6, -87],
                [97, 63, -62, 65, -87, -95, 80, 61, -13, -17, -94, -33, -93,
-78, -39, 52, 9, -20, -18, -11, -85, -89, 11, -44, -87, -80, -71, 56, -74,
-92, 37, 80, 64, -1, -76, -73],
                [-35, 0, -40, -19, 47, 31, 59, -78, -95, -9, 81, 33, -64, 20,
-40, 63, 75, -77, 31, -65, 54, 65, 65, 36, -61, -45, -4, -12, -48, 69, 35,
77, 76, 43, -85, -42],
                [-12, -51, 82, -72, 10, 95, 35, -41, 28, -37, -38, 99, 84,
76, 96, -73, -4, 55, -9, -35, 35, 22, 25, 67, 71, -71, -6, -92, -25, -13,
-63, 47, 33, -6, -93, -98],
                [54, 24, 78, -76, -49, -62, -51, 87, 93, -46, 45, -81, -30,
-72, -45, 56, 98, -34, -41, 83, 77, 64, 67, -17, 87, -50, -17, 79, -65, -52,
2, -36, 25, -17, 88, 37],
                [98, 30, -66, 90, -25, 3, 92, -29, 31, -66, 61, -68, 15, -74,
-36, -10, -75, 37, 24, -21, 41, 55, 62, -5, 29, -50, -70, -74, -54, -35,
-71, 75, 82, 8, -40, 34],
                [-94, 76, 12, -69, -2, -90, -71, -66, -75, 98, 66, 62, 3,
-34, -99, -13, 37, 28, -72, -11, 61, -65, 84, -24, -79, 64, 35, -32, 80,
-83, 17, 77, 19, -94, 75, 55],
                [-65, 21, -76, 24, -37, 20, -95, -31, 96, 47, 7, 43, -67,
-58, 81, 64, -94, 31, 65, 19, -48, -17, -23, 39, 80, 60, -34, -21, -83, -3,
-32, -20, -42, 68, -30, -10],
```

```
23              [-76, -12, 93, 1, -93, -90, 52, -52, -55, 55, -16, 67, -46,
    -15, 1, -91, -40, 48, -88, 56, 53, -80, -82, -82, -92, 77, -59, 14, -94, 17,
    -37, -41, 97, 37, 3, -84],
24              [-26, 53, 68, 58, -76, -55, -71, -16, 68, 46, -94, 84, 34,
    -85, -87, -91, -35, 5, 22, -89, 23, -71, 83, -25, -16, 22, -79, -74, 95,
    -12, 42, -52, 34, -91, -65, 41],
25              [10, -18, -1, -87, 35, 54, -76, 31, -53, -45, -59, -76, -97,
    -50, 64, 52, -68, 95, 17, -19, 6, 25, 59, -24, 50, 25, 62, 89, -1, -60, -95,
    -98, 9, -83, 33, -74],
26              [-80, 88, 44, -50, -51, -68, -39, -54, -62, -55, 40, -55, 40,
    -29, -73, -34, 12, -79, -97, 99, -53, 77, -2, 76, 32, -27, -45, 73, 30, 52,
    -54, -44, 70, 98, 84, -42],
27              [55, -64, 16, 69, -60, 70, 95, 9, -28, 46, -62, 20, -39, 70,
    77, 48, 16, -17, -64, -76, 86, 63, -90, 36, 5, -27, -82, 30, -31, 3, 74,
    -78, -29, -83, 78, 37],
28              [7, -81, -71, -55, 81, -47, 92, 89, 64, -46, -60, -89, -63,
    58, -46, 89, 85, -82, 18, -7, -14, 97, 81, 39, -57, 91, -20, -29, 50, 33,
    -39, -44, -79, 97, -97, 10],
29              [-80, -5, -18, 84, -47, 91, -48, -6, 98, -8, -41, -70, -79,
    -63, -20, -22, -1, 49, 52, 64, 23, 69, -16, -4, -45, -89, 44, -4, 82, 67,
    88, 92, 88, 91, -6, -93],
30              [-29, 0, 72, 41, -51, 0, -72, 61, -35, 2, -29, 89, -94, -14,
    34, -57, 27, -73, -92, -1, 99, 99, -69, -2, -48, -39, -6, -8, 42, 3, 84,
    -92, -46, -61, -23, -28],
31              [-40, 44, 43, -77, 60, -95, -28, -2, 85, -37, 43, 47, 74, 28,
    88, -98, 78, -30, -84, 90, -29, 44, -49, 5, -51, 62, -21, 5, 49, 22, 58, 85,
    -66, 32, 75, 96],
32              [-66, 30, -68, -95, -85, -91, -92, -54, -91, -48, 85, -76,
    -85, -16, 68, 57, -15, -56, -88, 44, -89, -32, -88, 83, -25, -100, -92, -59,
    65, 4, -26, 50, 73, 46, 5, 91],
33              [31, 5, 62, -71, -33, -2, 77, -45, -62, 88, -98, 8, 16, -71,
    59, 46, 84, 4, -58, -41, -49, -19, 54, -23, -35, -55, -6, -21, -48, -96, 85,
    -96, 69, 51, -32, 49],
34              [-61, -69, -50, -29, -42, -100, -66, -28, 26, -93, 33, -88,
    88, 15, 13, 28, 99, 0, 25, -81, 74, -46, -45, 54, 66, 2, 83, -66, 53, -39,
    18, -30, -78, 89, 84, -21],
35              [46, -36, -96, -68, 10, 97, -50, 13, 59, -38, -11, -43, -7,
    31, 62, -34, -5, 0, 84, -48, 40, 0, -97, -83, 60, 14, 1, -7, 64, 44, 88, 10,
    -31, 14, 27, -55],
36              [-96, -91, 7, -88, -99, 26, -73, 2, -57, -40, 97, -30, 16,
    99, 58, 36, -72, 45, -28, 27, -32, 55, 6, 79, 57, -54, -45, -6, -59, 39,
    -90, 4, 90, -6, 58, -10]])
37  Y = numpy.mat([-874, 21163, 45615, -37017, 72092, -27809, 9604, 25498,
    -10472, 6560, -69431, 54106, -8292, -44677, -17772, -77151,11531, 4538,
    33735, -7107, -17028, -21641, -71317, -41387, -30463, -14435, 23472, 7913,
    23824, -13865, 50179,-75429, -18764, -20428, 11973, -23186]).T
38  X = A.I * Y
39  print("".join([chr(round(c)) for t in X.getA() for c in t]))
```

搜索 SMC 得到解决方案：在 IDA 中稍加寻找可以发现一个函数 `sub_40699B`，正是这个函数对 `sub_401216` 进行加密

```
__int64 sub_40699B()
{
  __int64 result; // rax
  unsigned int i; // [rsp+Ch] [rbp-4h]

  mprotect(&dword_400000, 0x10000uLL, 7);
  for ( i = 0; ; ++i )
  {
    result = i;
    if ( i > 0x43E )
      break;
    *((_BYTE *)sub_401216 + (int)i) ^= byte_409080[i];
  }
  return result;
}
```

构造 IDA 脚本，执行后再次查看 `sub_401216`，发现新的矩阵运算

```
#include <idc.idc>
static main() {
    auto addr = 0x401216;
    auto i = 0;
    for (; i <= 0x43E; i++)
        PatchByte(addr + i, Byte(addr + i) ^ Byte(0x409080 + i));
}
```

构造下面代码

```
A = numpy.mat([[104, 103, 97, 109, 101, 123],
               [64, 95, 70, 65, 75, 69],
               [95, 102, 108, 97, 103, 33],
               [45, 100, 111, 95, 89, 48],
               [117, 95, 107, 111, 110, 119],
               [95, 83, 77, 67, 63, 125]])
Y = numpy.mat([[55030, 61095, 60151, 57247, 56780, 55726],
               [46642, 52931, 53580, 50437, 50062, 44186],
               [44909, 46490, 46024, 44347, 43850, 44368],
               [54990, 61884, 61202, 58139, 57730, 54964],
               [48849, 51026, 49629, 48219, 47904, 50823],
               [46596, 50517, 48421, 46143, 46102, 46744]])
X = Y * A.I
print("".join([chr(round(c)) for t in X.getA() for c in t]))
```

最终得到 Flag，`hgame{E@sy_Se1f-Modifying_COoodee33}`

Crypto

LikiPrime

描述

Wow! RSA!

解题思路

注：由于 `n` `e` `c` 为服务器随机提供的值，故题解中不表面具体数值，只提供解题思路。

```
def get_prime(secret):
    prime = 1
    for _ in range(secret):
        prime = prime << 1
    return prime - 1
```

观察代码可得 p 和 q 均为 2^k-1 的形式，故可通过下述代码直接枚举出可能的组合，并求出 Flag

```
p, q = 0, 0
for i in range(2, int(math.log2(n))//2):
    p = (1 << i) - 1
    if n % p == 0: # 判断 p 是满足条件
        q = n // p # 得到 q
        d = libnum.invmod(e, (p - 1) * (q - 1)) # 求 e 的逆元
        print(libnum.n2s(pow(c, d, n))) # 得到答案
```

最终得到 Flag，`hgame{Mers3nne~Pr!Me^re4l1y_s0+5O-li7tle!}`

HappyNewYear!!

描述

Liki 的朋友们在新年的时候给她发送了新年祝福
好家伙，一看就是群发的，有几个朋友发送的内容还是相同的！

解题思路

注：由于 `n` `e` `c` 为服务器随机提供的值，故题解中不表面具体数值，只提供解题思路。

有题意可得，明文中有重复的内容，且加密指数 e 比较低，结合一些关键字即可找到一种名为 `RSA低加密指数广播攻击` 的方法，其基本思想是由于信息 c 相同，e 相同，所以 c^e 相同，不同次的加密改变的 n 的，得到了不同的 $c^e \mod n$ 的值，故可通过中国剩余定理来得到可能的 c^e 的值，在对得到的值开 e ，次方即可得到明文。下面是代码实现。

```
for i in range(len(n)): # n e c 均为数组且其中数值一一对应
    for j in range(i + 1, len(n)): # 为避免重复走后一半即可
        x = libnum.solve_crt((c[i], c[j]), (n[i], n[j])) # 由于不知道哪些是配对的，所以两两枚举
        s = libnum.n2s(int(libnum.nroot(x, e[0]))) # 得到解密后的文字
        if s[0] >= 32: # 若解密后的字符串是正常的则输出
            print(s)
```

最终得到下面两段话，观察到两段话的末尾拼起来是个 Flag

```
Hello Liki4:

I am afraid that there are too many blessings on the 30th night, you will not see my greetings,
I am afraid that the firecrackers in the first grade are too noisy, you will not hear my blessings,

@ind3r~YOu^9ot=i7}
```

```
I am afraid the dishes in the second grade are too fragrant, you will not reply my text messages,
so I won't give you New Year greetings this year, I hope you don't know how to praise, good night.

hgame{!f+yOu-pl4y_rem
```

最终得到 Flag，`hgame{!f+yOu-pl4y_rem@ind3r~YOu^9ot=i7}`

$x^g \equiv b \pmod{p}$

$a^{\phi(m)} \equiv 1 \pmod{m}$

MISC

A R K

描述

星藏点雪 月隐晦明 以夕为墨 泼雪作屏
补充说明：明日方舟是一款塔防游戏，可以将可部署单位放置在场地中。并且具有自律功能，可以记录部署的操作。
翻译：没用 没用 出题人用可部署单位画了个东西 背景是白色的
本题目所有解题操作均只用流量，与网址无关

解题思路

用 Wireshark 打开 `.pcapng` 文件，发现很多链接是经过 TLS 加密的，往下翻翻发现一个通过 FTP 协议下载的文件 `ssl.log`

下载后导入 Wireshark 发现有些 TLS 被解密了，向下翻发现一个名为 `getBattleReplay` 的请求，其返回值为一段 Base64 编码

解密后发现是一个损坏的 `zip` 文件，通过下面的代码修复文件头

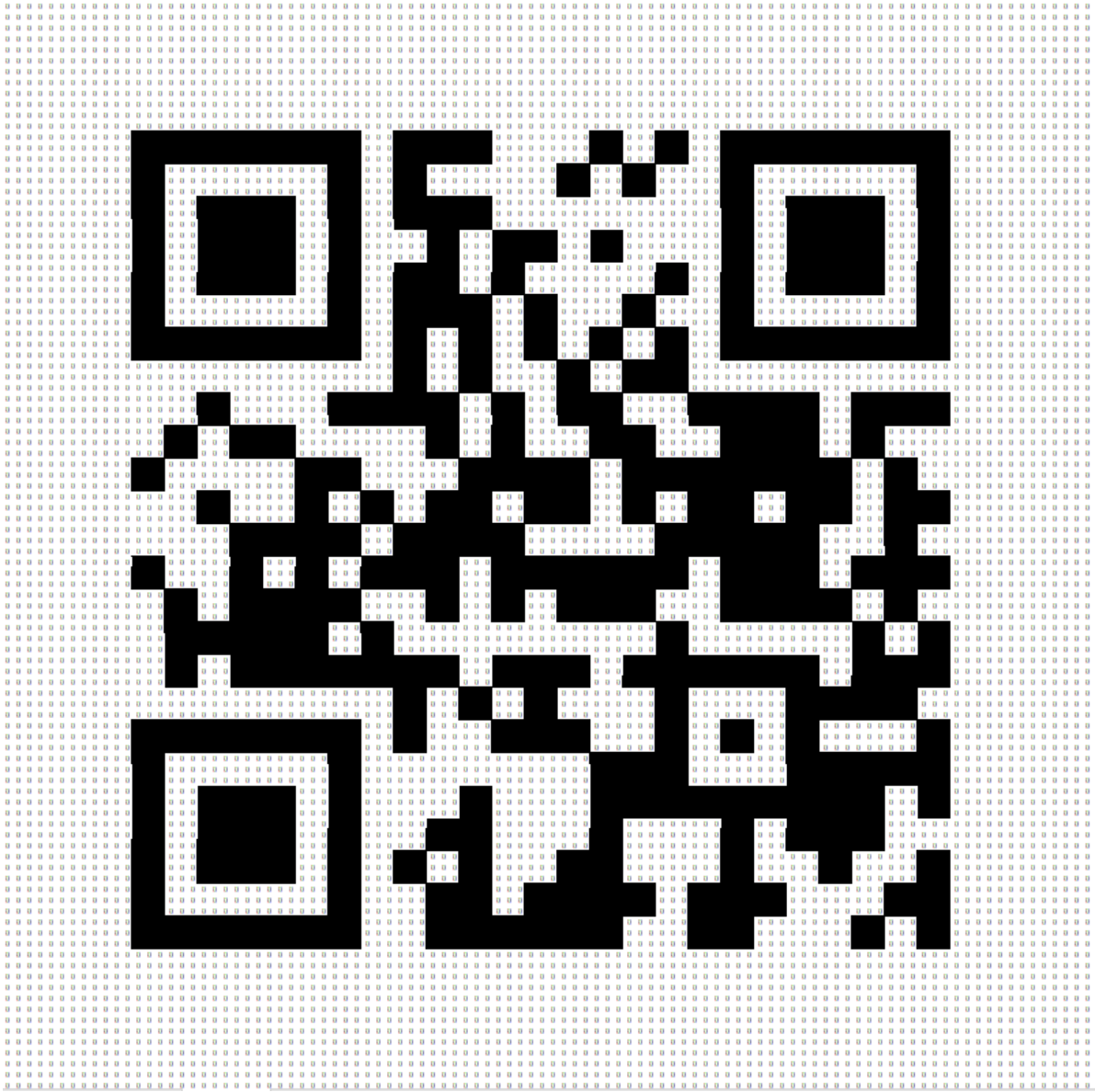
```python
import base64

s = ... # s 为得到的 Base64 字符串
open("file.zip", mode = 'wb').write((lambda ss: b'PK\x03\x04' + ss[4:])(base64.b64decode(s)))
```

打开压缩包得到一个名为 `default_entry` 的文件，文件结构如下

```
{
    "campaignOnlyVersion": ...,
    "timestamp": ...,
```

```
    "journal": {
        "metadata": {
            "standardPlayTime": ...,
            "gameResult": ...,
            "saveTime": ...,
            "remainingCost": ...,
            "remainingLifePoint": ...,
            "killedEnemiesCnt": ...,
            "missedEnemiesCnt": ...,
            "levelId": ...,
            "stageId": ...,
            "validKilledEnemiesCnt": ...
        },
        "squad": [
            {
                "charInstId": ...,
                "skinId": ...,
                "tmplId": ...,
                "skillId": ...,
                "skillIndex": ...,
                "skillLvl": ...,
                "level": ...,
                "phase": ...,
                "potentialRank": ...,
                "favorBattlePhase": ...,
                "isAssistChar": ...
            }
        ],
        "logs": [
            {
                "timestamp": ...,
                "signiture": {
                    "uniqueId": ...,
                    "charId": "..."
                },
                "op": ...,
                "direction": ...,
                "pos": {
                    "row": ...,
                    "col": ...
                }
            },
        ],
        "randomSeed": ...,
        "runeList": ...
    }
}
```

发现 `logs` 下有大量数据，每条数据都有个 pos 属性，包含了行列的坐标，绘制张白底的图片将对应坐标涂黑即可得到一个二维码



最终得到 Flag，`hgame{Did_y0u_ge7_Dusk?}`