逆向期末

DES

可以看到这里使用了DES加密，其秘钥为44 45 33 5F 45 6E 31 43，具体证据如下：
秘钥生成：0x00405140处找到了PC-1置换的表，39 31 29 21 19 11 09 01 3A 32 2A 22 1A 12 0A 02 3B 33 2B 23 1B 13 0B 03 3C 34 2C 24 3F 37 2F 27 1F 17 0F 07 3E 36 2E 26 1E 16 0E 06 3D 35 2D 25 1D 15 0D 05 1C 14 0C 04；0x00405180处找到了PC-2置换的表，0E 11 0B 18 01 05 03 1C 0F 06 15 0A 17 13 0C 04 1A 08 10 07 1B 14 0D 02 29 34 1F 25 2F 37 1E 28 33 2D 21 30 2C 31 27 38 22 35 2E 2A 32 24 1D 20
加密过程：0x00405060处找到了IP置换的表，3A 32 2A 22 1A 12 0A 02 3C 34 2C 24 1C 14 0C 04 3E 36 2E 26 1E 16 0E 06 40 38 30 28 20 18 10 08 39 31 29 21 19 11 09 01 3B 33 2B 23 1B 13 0B 03 3D 35 2D 25 1D 15 0D 05 3F 37 2F 27 1F 17 0F 07；0x004050A0处找到了PC-2置换的表，28 08 30 10 38 18 40 20 27 07 2F 0F 37 17 3F 1F 26 06 2E 0E 36 16 3E 1E 25 05 2D 0D 35 15 3D 1D 24 04 2C 0C 34 14 3C 1C 23 03 2B 0B 33 13 3B 1B 22 02 2A 0A 32 12 3A 1A 21 01 29 09 31 11 39 19；在0x004050E0处找到了E扩展的表，20 01 02 03 04 05 04 05 06 07 08 09 08 09 0A 0B 0C 0D 0C 0D 0E 0F 10 11 10 11 12 13 14 15 14 15 16 17 18 19 18 19 1A 1B 1C 1D 1C 1D 1E 1F 20 01；在0x00405120处找到了P盒置换的表，10 07 14 15 1D 0C 1C 11 01 0F 17 1A 05 12 1F 0A 02 08 18 0E 20 1B 03 09 13 0D 1E 06 16 0B 04 19；在0x004051C0处找到了一堆S盒代换的表。
我们将密文和秘钥放到cyberchef工具箱中进行解密，得到明文：

RC4

可以看到这里使用了一个RC4加密，其秘钥为52 43 34 6B 65 79，具体证据如下：
初始化：循环了256次去初始化了一个sbox，然后使用秘钥交换操作对sbox进行打乱，其伪代码类似：

```
for i from 0 to 255
    S[i] = i
j = 0
for i from 0 to 255
    j = (j + S[i] + key[i mod key_length]) mod 256
    swap(S[i], S[j])
```

秘钥生成：使用两个索引变量i和j来动态更新sbox并输出，其伪代码类似：

```
i = 0
j = 0
while generating output:
    i = (i + 1) mod 256
    j = (j + S[i]) mod 256
    swap(S[i], S[j])
    output S[(S[i] + S[j]) mod 256]
```

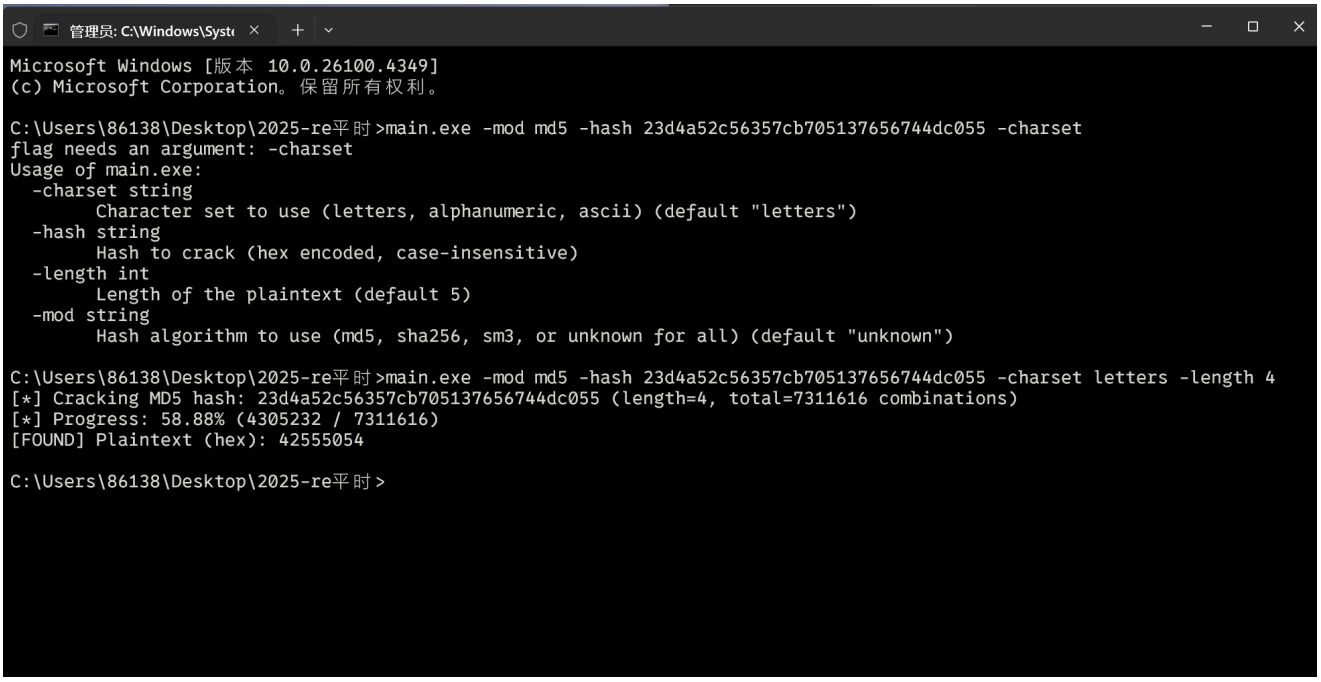
由于RC4是流密码，输入输出长度相同且加解密一致，因此可以通过patch的方法，将输入位置的字节直接替换为输出位置的字节，那么输出位置的字节则为原本的明文，得到明文：

MD5

可以看到这里使用了md5哈希算法，哈希值为23d4a52c56357cb705137656744dc055，具体证据如下：
首先md5的哈希值为16字节，和上面的哈希值长度所匹配；其次可以看到md5的初始IV生成算法如下：

```
    v3 = shift(D);
    v4 = shift(C);
    v5 = shift(B);
    v6 = shift(A);
    sprintf(out, "%08x%08x%08x%08x", v6, v5, v4, v3);
```

由于函数的输入为4位大小写字母，因此可以使用脚本去进行爆破：



解码后得到为BUPT

SHA256

可以看到这里使用了sha256哈希算法，哈希值为45c6c64ffd31a750bcb2e150519f1963e5424bafd380637cff33075c25d35334，具体证据如下：
首先sha256的哈希值为32字节，和上面的哈希值长度匹配；其次看到了sha256的初始IV如下：

```
    ctx->datalen = 0;
    ctx->bitlen = 0LL;
    ctx->state[0] = 1779033703;
    ctx->state[1] = -1150833019;
    ctx->state[2] = 1013904242;
    ctx->state[3] = -1521486534;
    ctx->state[4] = 1359893119;
    ctx->state[5] = -1694144372;
    ctx->state[6] = 528734635;
    ctx->state[7] = 1541459225;
```

由于函数的输入为4位大小写字母，因此可以使用脚本去进行爆破：

```
C:\Users\86138\Desktop\2025-re平时>main.exe -mod md5 -hash 23d4a52c56357cb705137656744dc055 -charset letters -length 4
[*] Cracking MD5 hash: 23d4a52c56357cb705137656744dc055 (length=4, total=7311616 combinations)
[*] Progress: 58.88% (4305232 / 7311616)
[FOUND] Plaintext (hex): 42555054

C:\Users\86138\Desktop\2025-re平时>main.exe -mod sha256 -hash 45c6c64ffd31a750bcb2e150519f1963e5424bafd380637cff33075c25
d35334 -charset letters -length 4
[*] Cracking SHA-256 hash: 45c6c64ffd31a750bcb2e150519f1963e5424bafd380637cff33075c25d35334 (length=4, total=7311616 com
binations)
[*] Progress: 65.81% (4811473 / 7311616)
[FOUND] Plaintext (hex): 42555054
```

解码后得到为BUPT

EXCEPTION

可以看到这里发生了除零异常，由于在IDA中能够直接看到try/exception函数，因此可以直接在IDA中去对异常进行patch修改。
首先修改try代码段结尾中JMP的地址，使其直接跳转到异常处理的代码段：
然后将发生除零异常处的指令都nop掉，使其能够顺利的通过：
F5重新进行反编译，此时能够看到正常的伪代码：

IAT Hook

可以看到这里对WriteFile函数进行了一个IAT Hook操作，具体证据如下：
这里使用GetModuleHandleA(0)来获取了模块的基地址，然后进行了如下类似操作：

```
v4 = *(_DWORD *)(*(_DWORD *)(a1 + 60) + a1 + 128);
v5 = (_DWORD *)(a1 + v4);
```

这是经典的寻找导入地址表的操作，因此可以判断hook方式为IAT Hook。
其中，WriteFile的地址条目被替换为自定义的函数sub_401000，我们跟进来看一下逻辑。

API Hook

可以看到这里对WriteFile函数进行了一个API Hook操作，具体证据如下：
这里使用了VirtualProtect修改了WriteFile函数的前5个字节，使其跳转到自定义的函数_Z11MyWriteFilePvPKvmPmP11_OVERLAPPED，我们跟进这个函数，发现其中实现了自定义的脱钩函数，也验证了这一说法。
接下来分析_Z11MyWriteFilePvPKvmPmP11_OVERLAPPED函数的逻辑

Flower

可以看到这里使用了花指令，导致反汇编算法在线性扫描的时候无法正确的识别。其中，在如下地址处存在xor_jz型的jmp花指令：
0x401836
0x401960
我们将其nop掉，然后Apply Patches Into File，重新打开文件，发现可以正确的反汇编。