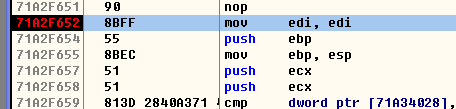
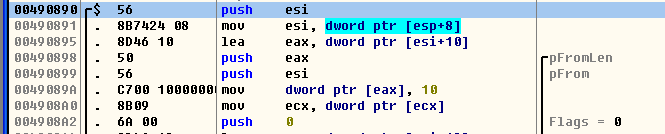
Send:

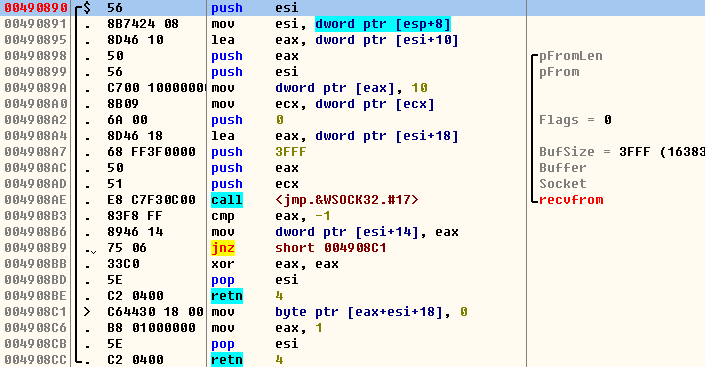


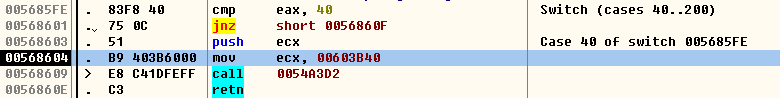


可能的解密入口：



收取数据：

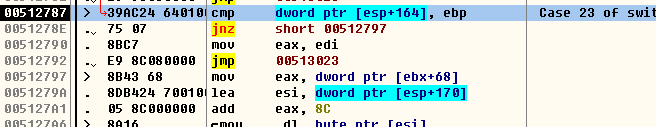




该代码可能是根据消息头进行群逻辑处理。

Mov ecx, 00603b40 该地址很有可能是个全局对象，而0054a3d2则是该对象的一个成员函数。代码执行到此处时，接收缓存的内容依然未改。[推测错误 ]

群的消息命令字是4194339（16进制400023），去掉高位，也就是23。在以下代码发现了处理分支：



代码走到以下位置时，esi指向的内存位置保存着接收的群消息：

00123644 38 38 41 45 31 44 44 34 36 36 46 44 00 00 00 00 88AE1DD466FD....

00123654 00 00 00 00 00 00 00 00 00 00 00 00 59 CF 3A 4D ............Y?M

00123664 23 00 40 00 21 00 00 00 00 00 00 00 5E 3B 83 A1 #.@.!.......^;儭

00123674 14 6D A4 D2 E3 D8 E8 AB B1 3A 5B BC F6 72 E2 EF mひ阖璜?[荐r怙

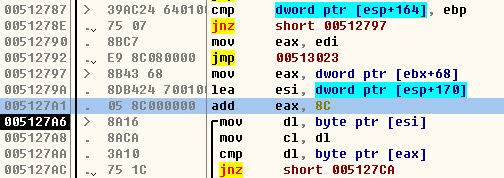
00123684 1D 52 51 32 60 3C AB E7 53 7A 67 3D 41 41 41 41 RQ2`<Szg=AAAA

即，123670位置保存着加密聊天内容。

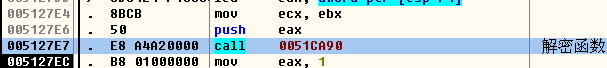
Eax中很可能保存的是密钥，add eax, 8c，内容为：

0124465C 30 30 32 36 32 32 43 42 39 46 37 39 002622CB9F79

很可能就是这12个字节。8c前的内容不清楚。



紧接着会调用一个函数：



0051ca90该函数则完成了将123670内存中加密内容解密的功能，得到如下的字符串：

00123670 62 62 62 00 53 47 4D 41 52 4B 23 33 35 39 63 39 bbb.SGMARK#359c9

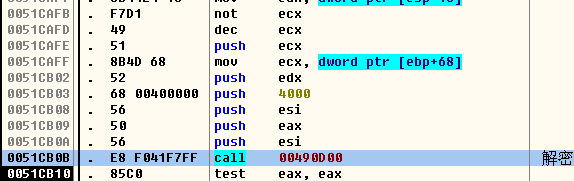
00123680 32 32 65 23 62 62 62 00 00 08 08 08 08 08 08 08 22e#bbb..

00123690 41 41 41 41 AAAA

Bbb就是聊天内容；359c922e是群的号码；

以上函数除了解密外，还会做一些UI处理，例如任务栏图标闪烁。这个函数非常长，需要进一步跟进，找到准确的解密位置。

进一步地，以下函数解密：



解出来的内容为：

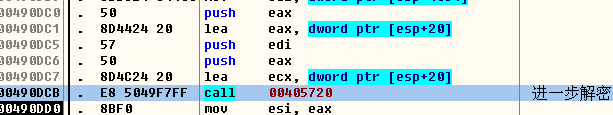
00123670 51 55 4E 4D 53 47 4D 41 52 4B 23 33 35 39 63 39 QUNMSGMARK#359c9

00123680 32 32 65 23 63 63 63 63 00 00 07 07 07 07 07 07 22e#cccc..

00123690 41 41 41 41 AAAA

其中,cccc是聊天内容。

继续跟进，在00490d00函数中：



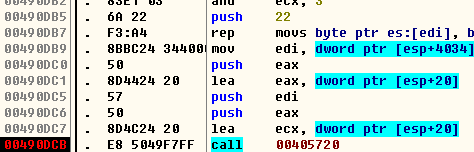
函数00405620中，只有一个call，两个call都包含大量数据操作指令，感觉上极有可能就是解密函数。

用CryptTool直接查看Feiq.exe，被扫描出使用了blowfish加密算法。

看看IP messager里的源代码（因为IP messager里也用了blowfish算法），blowfish.cpp/h里描述了blowfish的加解密接口：

DWORD Decrypt(const BYTE \*pInput, BYTE \*pOutput, DWORD lSize, int mode=BF\_CBC|BF\_PKCS5, \_int64 IV=0);

Mode参数默认为16进制22，在od里同样看到了压入参数22：



那么就可以据此推出：

Push 22 mode

Push eax lSize

Push edi output

Push eax input

同时注意this指针的传入，以及this对象的初始化，从blowfish.cpp的实现可以看出，需要在初始化时传入密钥之类的信息。

一次数据：

Size=20h

00123670 5E 3B 83 A1 14 6D A4 D2 E3 D8 E8 AB B1 3A 5B BC ^;儭mひ阖璜?[

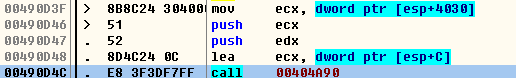
00123680 FD 2F 73 7D DA 75 A7 D5 D6 D0 7D 49 13 ED 84 DB ?s}趗д中}I韯

可以看出解密数据是直接取的群消息头之后的部分（可能会包含群消息之外的数据）。

[esp+20]处的内容和消息内容相同。

查看blowfish加解密类的使用，需要在初始化时传入密钥，这个是个关键点。对于这个类的使用，很大可能是将其作为一个临时对象构造，然后调用其解密函数。在以上分析的函数体内，必须在调用解密函数之前，构造了这个对象：

这个call非常可疑：



因为:

CBlowFish::CBlowFish (const BYTE \*key, int keybytes)

{

PArray = new DWORD [NPASS + 2];

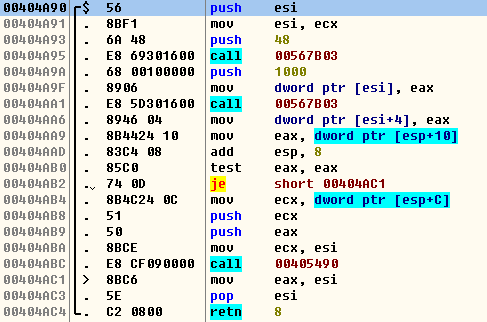
SBoxes = new DWORD [4][256];

if (key)

Initialize(key, keybytes);

}

该函数需要2个参数，而这里恰好push了两个参数，进入到00404A90函数处可见以下代码：



代码非常简单，和blowfish源码做对比：

CBlowFish::CBlowFish (const BYTE \*key, int keybytes)

{

PArray = new DWORD [NPASS + 2];

SBoxes = new DWORD [4][256];

if (key)

Initialize(key, keybytes);

}

其中NPASS为16，这里分配内存就是sizeof(DWORD)\*18，刚好等于48H！即可断定：上面那个可疑的call，就是构造blowfish对象，需要进一步跟踪其参数，即可获取密钥的来源。

调用404A90函数时，ecx就是key的长度，edx就是key。

在push处断点，断下来后ecx和edx值为：



Edx居然是发送方的网络地址（MAC或IP，目前未确认）。

BINGO！解密成功。

飞秋使用发送方的MAC地址作为解密的密钥，消息中携带了一个QUNMSGMARK字符串，然后是群号码，最后是发送内容。密钥的选取不当，导致可以轻易地监视到其他群的聊天内容。

自己写一个UDP组播接收程序，即可解出聊天内容。测试问题：

1. 发送带中文字符串时“测试中文AB”，recvfrom取出的数据末尾多了一个15，导致解密出错，但是有些中文又不出错（另外，自己程序里也有问题，用了strlen去测试接收到的数据长度，对于有些数据，中间可能会携带’\0’字符。）

修改了程序准确地处理了长度后，就算数据不是8的整数倍，也可以解密成功。此外，飞秋协议头最后一个#带了一个数字，经过几次数据跟踪，正是数据长度，而且是8的整数倍，解密数据长度完全可以使用该数据长度。

飞球退出后会广播2组消息，消息类型分别为0和600002H。

上线时会广播消息0、600003H、2000C9H（带群号码）、202（带群号码）。