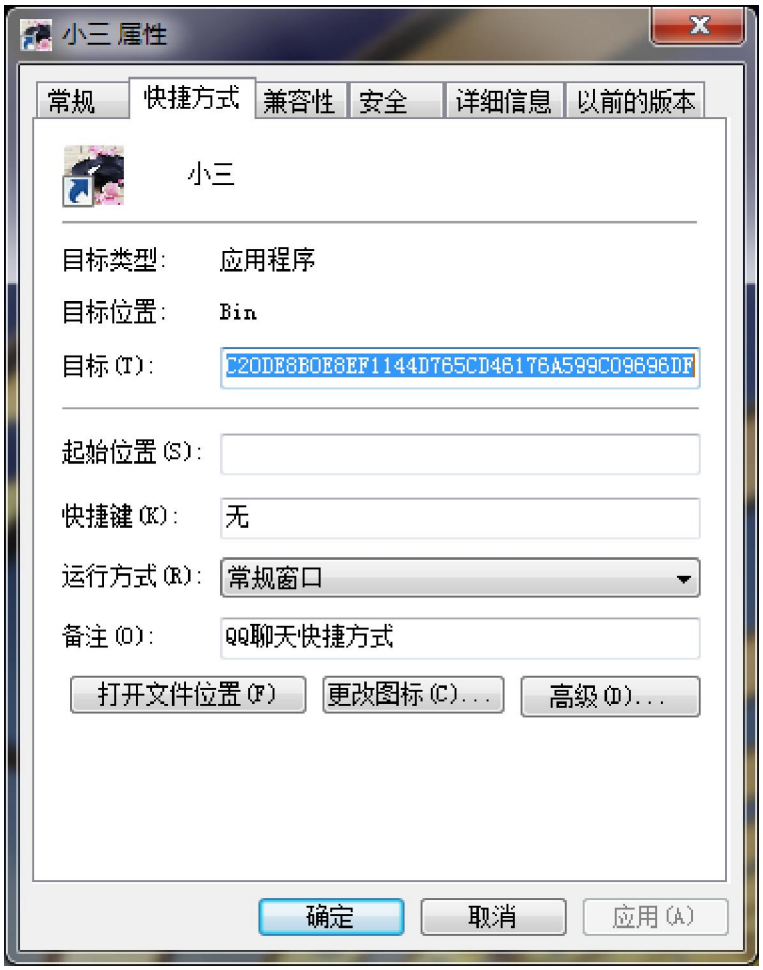
QQScLauncher 逆向分析

逆向版本：QQ2011 正式版（2425）
逆向目的：只是感兴趣，没有任何其他意图
版权声明：本文档由**代码疯子**整理，在保留本文档版权声明和原始出处的前提下欢迎转载！
需要工具：IDA Pro、Ollydbg
正文内容：

在 QQ 最新版（QQ2011 正式版）中加入了不少新功能，如手写、视频群聊、语音输入等，另外还有一个功能是可以把好友拖放到桌面上，点击就可以直接与之进行聊天，处于好奇，本人对他进行了一下逆向，于是便有本文。

先去准备两个 QQ 小号（842342202、2390318912，不是必须的，这里显示，所以用小号）

先随便把一个好友拖放到桌面上，会产生一个快捷方式，查看这个快捷方式的属性，得到目标内容：




```
"C:\Program Files\Tencent\QQ\Bin\QQScLauncher.exe" /uin:842342202 /quicklunch:
1D0BEC54CBEE33C09ED6F8089A7528E12EA3F4892C20DE8B0E8EF1144D765CD46176A599C09696DF
```

可以看到，这里是通过一个叫做 QQScLauncher.exe 的程序来启动的，uin 参数指定自己的 QQ 号码，

quicklaunch 肯定就是用于鉴别好友用的了。

用 PEiD 对 QQScLauncher.exe 进行查壳，结果什么也没查到，估计是我的 PEiD 数据库太久了，用 File Format Identifier 查出来是 Visual C++ 2005 Release -> Microsoft。其实是什么不要紧，拿起 IDA 逆一逆就没什么了。

下面使用 IDA 对其进行简单的分析，可以看到一个获取命令行参数**/uin** 和**/quicklunch** 的过程：

```
.text:00401000 sub     esp, 270h
.text:00401006 mov     eax, dword_403000
.text:0040100B xor     eax, esp
.text:0040100D mov     [esp+270h+var_4], eax
.text:00401014 mov     eax, ds:__argc          ;命令行参数个数 argc的地址
.text:00401019 push    ebx
.text:0040101A push    ebp
.text:0040101B push    esi
.text:0040101C push    edi
.text:0040101D xor     edi, edi                ; edi清零
.text:0040101F xor     ebp, ebp                ; ebp清零
.text:00401021 cmp     [eax], edi              ;看看是不是正常启动（三个参数）
.text:00401023 mov     [esp+280h+var_270], edi  ;存放 QQ号的位置
.text:00401027 mov     [esp+280h+var_26C], edi  ; 0/1暗示 QQ号是否找到
.text:0040102B mov     [esp+280h+var_268], edi  ; 0/1暗示 quicklunch是否找到
.text:0040102F jle     loc_40112A              ;不是正常启动则跳转
.text:00401035 mov     ebx, ds:wcsncmp         ;存储 wcsncmp函数的地址，后面会经常用到
.text:0040103B jmp     short loc_401040
.text:0040103B ; ---------------------------------------------------------------------------
.text:0040103D align 10h
.text:00401040
.text:00401040 loc_401040:                     ; CODE XREF: wWinMain(x,x,x,x)+3Bj
.text:00401040                                 ; wWinMain(x,x,x,x)+93j
.text:00401040 mov     ecx, ds:__wargv         ;命令行参数的地址
.text:00401046 mov     edx, [ecx]              ;取得第一个字符串的地址
.text:00401048 mov     esi, [edx+edi*4]        ;取得第一个字符串的字符
.text:00401048                                 ;第一次将取到 C:\Program Files\Tencent\QQ\Bin\QQScLauncher.exe
.text:00401048                                 ;
.text:0040104B push    5                       ; MaxCount
.text:0040104D push    offset Str2             ; "/uin:"
.text:00401052 push    esi                     ; Str1
.text:00401053 call    ebx ;wcsncmp            ;调用 wcsncmp比较字符串,看前五个字符是否是 /uin:
.text:00401055 add     esp, 0Ch                ;平衡 wcsncmp堆栈
.text:00401058 test    eax, eax                ;找到时返回 0, 否则返回非 0值
.text:0040105A jnz     short loc_40106D        ;没有找到 "/uin:"时跳转
.text:0040105C add     esi, 0Ah                ;0xA=10,跳过前十个字节,长度正好是 /uin:的 Unicode
长度
```

```
.text:0040105F mov     [esp+280h+var_270], esi ;把 QQ号码存下来
.text:00401063 mov     [esp+280h+var_26C], 1 ;表示 QQ号码已经找到
.text:0040106B jmp     short loc_401089
.text:0040106D ; ---------------------------------------------------------------------------
.text:0040106D
.text:0040106D loc_40106D:                     ; CODE XREF: wWinMain(x,x,x,x)+5Aj
.text:0040106D push    0Ch                     ; MaxCount
.text:0040106F push    offset aQuicklunch      ; "/quicklunch:"
.text:00401074 push    esi                     ; Str1
.text:00401075 call    ebx ; wcsncmp           ;调用 wcsncmp 比较字符串，看前 12 个字符是否是
/quicklunch:
.text:00401075                                 ;看到这我石化了 ,quicklunch? 快餐么？
.text:00401077 add     esp, 0Ch
.text:0040107A test    eax, eax                ;看是否找到 quicklunch
.text:0040107C jnz     short loc_401089        ;没有找到则跳转
.text:0040107E lea     ebp, [esi+18h]          ; quicklunch值的地址由 ebp保存
.text:00401081 mov     [esp+280h+var_268], 1 ;表示 quicklunch已经找到
.text:00401089
.text:00401089 loc_401089:                     ; CODE XREF: wWinMain(x,x,x,x)+6Bj
.text:00401089                                 ; wWinMain(x,x,x,x)+7Cj
.text:00401089 mov     eax, ds:__argc          ;命令行参数个数 argc的地址
.text:0040108E add     edi, 1                  ;准备取得下一个命令行参数
.text:00401091 cmp     edi, [eax]              ;看看命令行参数是否已经取完
.text:00401093 jl      short loc_401040        ;没有取完则调回去继续取
.text:00401095 xor     ebx, ebx
```

对于上面的代码中蓝色的行，我们是如何知道他的意义的呢？看下面的几行代码就知道了：

```
.text:00401095 xor     ebx, ebx                ; ebx寄存器清零
.text:00401097 cmp     [esp+280h+var_268], ebx ;从这里的比较可以知道这几个字段的意义
.text:0040109B jz      loc_40112A
.text:004010A1 cmp     [esp+280h+var_26C], ebx
.text:004010A5 jz      loc_40112A
```

可以看到，这里判断 quicklunch 和 QQ 号是否找到，没找到就跳转；跳转到的地址和上面命令行参数个数不是 3 是一样的（绿色的代码行）。

接下来，程序生成一个特定的字符串，格式为 "qqexchangewnd_shortcut_prefix_842342202"，后面那几位就是 QQ号码了。然后通过 FindWindow来查找窗口，调用代码如下：

```
004010E4    PUSH ECX          ; /Title = "qqexchangewnd_shortcut_prefix_842342202"
004010E5    PUSH 00402198     ;|5b3838f5-0c81-46d9-a4c0-6ea28ca3e942
004010EA    CALL DWORD PTR DS:[<&USER32.FindWindowW>]       ; \FindWindowW
```

如果 FindWindow返回非零值，则接下来的代码逻辑是计算出 quicklunch的长度，代码如下：

```
.text:004010F4 mov     ecx, ebp                ;让 ecx指向 quicklunch
```

```
.text:004010F6 mov     [esp+280h+ProcessInformation], ebx
.text:004010FA lea     esi, [ecx+2]          ; esi略过 quicklunch第一个字符
.text:004010FD lea     ecx, [ecx+0]
.text:00401100
.text:00401100 loc_401100:                   ; CODE XREF: wWinMain(x,x,x,x)+109j
.text:00401100 mov     dx, [ecx]             ;找到 ECX中第一个 '\0'
.text:00401100                               ;这里其实就是求字符串长度吧
.text:00401103 add     ecx, 2
.text:00401106 cmp     dx, bx
.text:00401109 jnz     short loc_401100
.text:0040110B sub     ecx, esi              ;计算出字符串的字节数
.text:0040110D sar     ecx, 1                ;因为是 Unicode, 所以字节数除以 2
```

然后，程序会调用 SendMessage 发送 WM_COPYDATA 消息，本人对这个消息不太熟，所以先看 MSDN 的解释：

程序通过发送一个 WM_COPYDATA 消息来把数据传送给另一个应用程序。

语法：

```
lResult = SendMessage(      // returns LRESULT in lResult
    (HWND) hWndControl,     // handle to destination control
    (UINT) WM_COPYDATA,     // message ID
    (WPARAM) wParam,        // = (WPARAM) () wParam;
    (LPARAM) lParam         // = (LPARAM) () lParam;
    );
```

参数：

wParam 为传送数据的窗口的句柄，lParam 是一个指向 COPYDATASTRUCT 结构的指针。

COPYDATASTRUCT 的结构如下：

```
typedef struct tagCOPYDATASTRUCT {
    ULONG_PTR dwData;   // 附加数据
    DWORD cbData;
    PVOID lpData;
} COPYDATASTRUCT, *PCOPYDATASTRUCT;
```

返回值：

如果接收方应用程序处理了这个消息，那么返回 TRUE，否则返回 FALSE。

逆向所得代码：

```
.text:00401117 push    ecx                   ; lParam(指向 COPYDATASTRUCT结构的指针)
.text:00401118 push    ebx                   ; wParam(这里是 0)
.text:00401119 push    4Ah                   ; Msg (就是 WM_COPYDATA)
.text:0040111B push    eax                   ;hWnd(接收方窗口句柄，上面 FindWindow的返回值)
.text:0040111C mov     [esp+290h+hObject], edx ; 填充 COPYDATASTRUCT.cbData
.text:00401120 mov     [esp+290h+var_25C], ebp ; 填充 COPYDATASTRUCT.lpData
.text:00401124 call    ds:SendMessageW       ; 调用 SendMessage
```

可以看到上面 SendMessage 时 wParam 用了 0，这里本来应该是发送方的窗口句柄值（***我猜*** *wParam*

的字段是为了通知接收方消息是从哪里传来的，这里接收方不会反过来和发送方通信，所以填0 也不影响。有兴趣自己测试吧，这里懒得验证了）。

之后，QQ 就可以弹出聊天窗口了（这里的逻辑就由 QQ 负责了）。

如果 QQ 没有开启，那么上面的 FindWindow 调用就返回 NULL，那么将进入另一个分支。来到如下代码：

```
.text:0040114A loc_40114A:                            ; CODE XREF: wWinMain(x,x,x,x)+F2j
.text:0040114A call    GetQQInstallPath               ; 如果 QQ没有开启，则跳转到这里
```

这里的函数名字是我自己改的，看代码逻辑可以知道他是用来获取 QQ 安装目录的：

```
.text:004012E0 GetQQInstallPath proc near             ; CODE XREF: wWinMain(x,x,x,x):loc_40114Ap
.text:004012E0 push    esi                            ; quicklunch
.text:004012E1 push    20Ah                           ; unsigned int
.text:004012E6 call    ??2@YAPAXI@Z                   ; operator new(0x020A)分配空间
.text:004012EB add     esp, 4
.text:004012EE mov     esi, eax
.text:004012F0 push    104h                           ; nSize
.text:004012F5 push    esi                            ; lpFilename
.text:004012F6 push    0                              ; hModule
.text:004012F8 mov     word ptr [esi], 0
.text:004012FD call    ds:GetModuleFileNameW          ; 得到当前进程的路径
.text:00401303 push    5Ch                            ; 字符 '\'
.text:00401305 push    esi                            ; Str
.text:00401306 call    ds:wcsrchr                     ; wcsrchr返回字符在字符串中最后一次出现的位置
.text:0040130C add     esp, 8
.text:0040130F test    eax, eax
.text:00401311 jz      short loc_401338               ; 没有找到 '\'时跳转
.text:00401313 add     eax, 2                         ; eax指向下一个字符
.text:00401316 mov     ecx, eax
.text:00401318 sub     ecx, esi                       ; 前面一段字符的字节数
.text:0040131A push    offset aQq_exe                 ; "QQ.exe"
.text:0040131F sar     ecx, 1                         ; 算术右移一位，得到长度
.text:00401321 mov     edx, 104h
.text:00401326 push    offset aS                      ; "%s"
.text:0040132B sub     edx, ecx
.text:0040132D push    edx                            ; SizeInWords
.text:0040132E push    eax                            ; Dst（替换 QQScLauncher.exe）
.text:0040132F call    ds:swprintf_s                  ; 调用 swprintf_s
.text:00401335 add     esp, 10h
.text:00401338
.text:00401338 loc_401338:                            ; CODE XREF: GetQQInstallPath+31j
.text:00401338 mov     eax, esi
.text:0040133A pop     esi
.text:0040133B retn                                   ; 返回
.text:0040133B GetQQInstallPath endp
```

上面先通过 new 来分配一段空间，然后调用 GetModuleFileName 来获取当前进程的路径，也就是：

```
C:\Program Files\Tencent\QQ\Bin\QQScLauncher.exe
```

然后通过 wcsrchr 来查找'\'字符，从而 eax 指向 **\QQScLauncher.exe**，eax 加上 2（一个 Unicode 字符长度），于是 eax 指向 **QQScLauncher.exe**，后面调用 swprintf_s 时第一个参数直接填入 eax 即可。

随后程序会调用一个函数来生成一些 CreateProcess 需要的一些参数，这个过程和前面分析的过程差不多，所以不再赘述。之后就是通过 CreateProcess 来创建一个 QQ.exe 进程了。

```
012829C8  |ModuleFileName = "C:\Program Files\Tencent\QQ\Bin\QQ.exe"
01282BE0  |CommandLine = "/uin:842342202
/quicklunch:1D0BEC54CBEE33C09ED6F8089A7528E12EA3F4892C20DE8B0E8EF1144D765CD46176A599C09696D
F"
00000000  |pProcessSecurity = NULL
00000000  |pThreadSecurity = NULL
00000000  |InheritHandles = FALSE
00000000  |CreationFlags = 0
00000000  |pEnvironment = NULL
00000000  |CurrentDir = NULL
0012FCA4  |pStartupInfo = 0012FCA4
0012FC94  \pProcessInfo = 0012FC94
```

可以看到这里把 QQ 号和好友的识别符作为命令行参数传给了 QQ.exe。如果启动 QQ.exe 失败则会继续尝试使用 CreateProcess 来启动 QQ，直到启动成功，之后程序的逻辑也就结束了。

分析到这里，可以看出 QQScLauncher.exe 的内部逻辑已经很清晰了，这个程序没有经过特殊处理，很适合像我这样的新手分析，有兴趣的话还可以自己写一个 QQScLauncher.exe。

有意思的是命令行参数竟然写成了 **quicklunch**……

至于 QQ 是如何通过那一串 16 进制来找到 QQ 好友的，暂时没有找到地方，有兴趣的朋友可以试着看一下。