**PC微信v3.5.0.46改动分析记录**

**(windows-3.5.0.46 version)**

**修订历史记录**

|  |  |  |  |
| --- | --- | --- | --- |
| 修改日期 | 版本 | 说明 | 作者 |
| 2022/02/14 | windows-3.5.0.46 |  | 胡智钦 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目录

[1 引言 4](#_Toc95751724)

# 引言

记录PC微信3.5.0.46版本的改动。

# Hook点改动

## CheckAppSingleton

3.5.0.46版本在WeChatWin.dll的导出方法StartWachat（RVA：wechatwin.dll:$CA7660），【注：Wachat这里没有打错】里面检查是否已经启动微信。里面直接调用CreateMutex创建互斥对象然后判断是否已经启动了微信，如果已经启动那么会把微信提到前台然后结束当前进程。

新版本使用的互斥对象的名字跟旧版本的还是一样，使用的是L”\_WeChat\_App\_Instance\_Identity\_Mutex\_Name”，但是多开微信的方式不能用之前的方法了，因为旧版本CheckAppSingleton是独立一个函数，新版本不是，如果在StartWachat的函数入口直接退出的话，那么不会走启动微信的流程。

修改方法如下：



【新版本多开微信方法】

### 特征与定位方法

特征使用旧版本的方法，L”\_WeChat\_App\_Instance\_Identity\_Mutex\_Name”的引用位置。然后从这个因为位置往后找cmp eax, B7指令，下一个字节就是修改目标。

字符串十六进制流：

{

0x5F, 0x00, 0x57, 0x00, 0x65, 0x00, 0x43, 0x00, 0x68, 0x00, 0x61, 0x00, 0x74, 0x00, 0x5F, 0x00,

0x41, 0x00, 0x70, 0x00, 0x70, 0x00, 0x5F, 0x00, 0x49, 0x00, 0x6E, 0x00, 0x73, 0x00, 0x74, 0x00,

0x61, 0x00, 0x6E, 0x00, 0x63, 0x00, 0x65, 0x00, 0x5F, 0x00, 0x49, 0x00, 0x64, 0x00, 0x65, 0x00,

0x6E, 0x00, 0x74, 0x00, 0x69, 0x00, 0x74, 0x00, 0x79, 0x00, 0x5F, 0x00, 0x4D, 0x00, 0x75, 0x00,

0x74, 0x00, 0x65, 0x00, 0x78, 0x00, 0x5F, 0x00, 0x4E, 0x00, 0x61, 0x00, 0x6D, 0x00, 0x65, 0x00,

0x00, 0x00

};

cmp eax, B7指令十六进制流：

{

0x3D, 0xB7, 0x00, 0x00, 0x00

};

### 修改方法

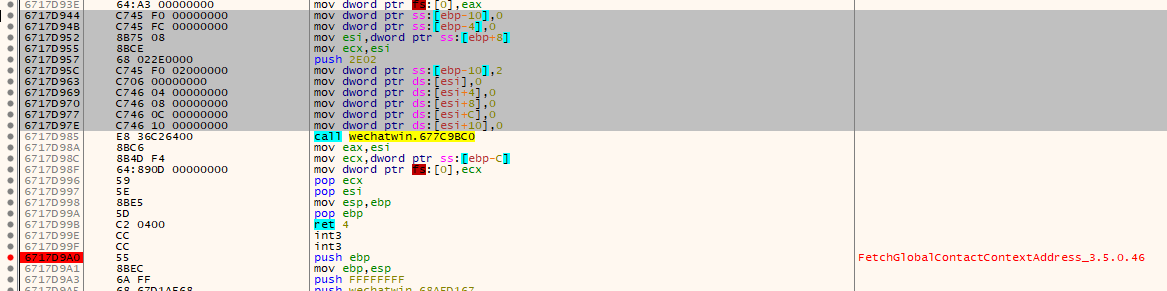
定位到修改位置后把jne指令改为jmp指令，即0x75修改位0xEB

### RVA

wechatwin.dll:$CA7818

## FetchGlobalContactContextAddress

### 特征



使用这个函数的前一个函数的特征来做定位

扫描十六进制流：

{

0xC7, 0x45, 0xF0, 0x00, 0x00, 0x00, 0x00, 0xC7, 0x45, 0xFC, 0x00, 0x00, 0x00, 0x00, 0x8B, 0x75,

0x08, 0x8B, 0xCE, 0x68, 0x02, 0x2E, 0x00, 0x00, 0xC7, 0x45, 0xF0, 0x02, 0x00, 0x00, 0x00, 0xC7,

0x06, 0x00, 0x00, 0x00, 0x00, 0xC7, 0x46, 0x04, 0x00, 0x00, 0x00, 0x00, 0xC7, 0x46, 0x08, 0x00,

0x00, 0x00, 0x00, 0xC7, 0x46, 0x0C, 0x00, 0x00, 0x00, 0x00, 0xC7, 0x46, 0x10, 0x00, 0x00, 0x00,

0x00

};

定位十六进制流：

{

0xCC, 0xCC, 0x55, 0x8B, 0xEC

};

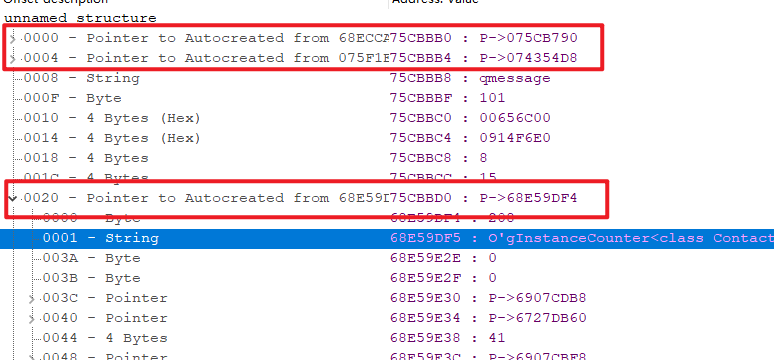
偏移：2字节

### RVA

wechatwin.dll:$8D9A0

### 变化

获取到ContactContext后，联系人信息的头节点的首地址的偏移变为了0x4C，另外联系人信息的节点结构也发生了变化：



整体缩减了8字节，但是从0x20位置后的实际信息的结构跟之前是一样的

## InitWeChatContactItem

新版本的特征没有变化，用3.4.5.27版本的特征能够定位到

### RVA

wechatwin.dll:$632420

## DeinitWeChatContactItem

特征发生了变化，但是还是在InitWeChatContactItem函数的下一个位置

### 特征

定位到InitWeChatContactItem方法后，向后定位第一个下面执行序列的位置

int 3

int 3

push esi

move esi, ecx

指令十六进制流：

{

0xCC, 0xCC, 0x56, 0x8B, 0xF1

};

偏移：2字节

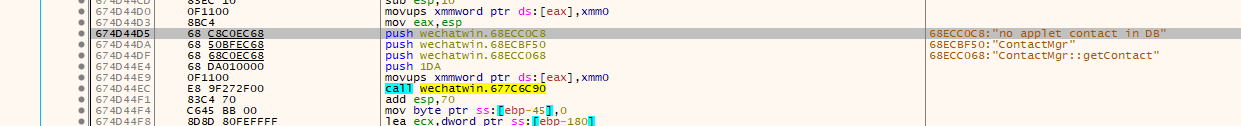
### RVA

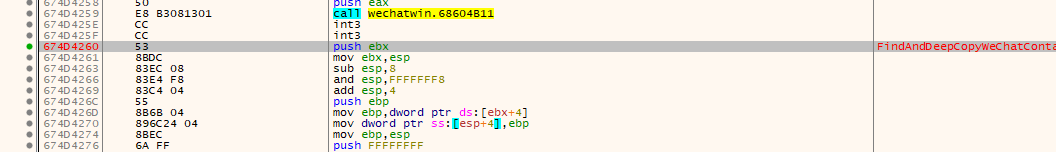
wechatwin.dll:$632F20

## FindAndDeepCopyWeChatContactItemWrapper

### 特征

有一个新字符串“no applet contact in DB”可用来做特征，找到这个字符串的引用位置后，往回定位到函数入口地址





字符串十六进制流：

{

0x6E, 0x6F, 0x20, 0x61, 0x70, 0x70, 0x6C, 0x65, 0x74, 0x20, 0x63, 0x6F, 0x6E, 0x74, 0x61, 0x63,

0x74, 0x20, 0x69, 0x6E, 0x20, 0x44, 0x42, 0x00

};

定位十六进制流：

{

0xCC, 0xCC, 0x53, 0x8B, 0xDC, 0x83, 0xEC

};

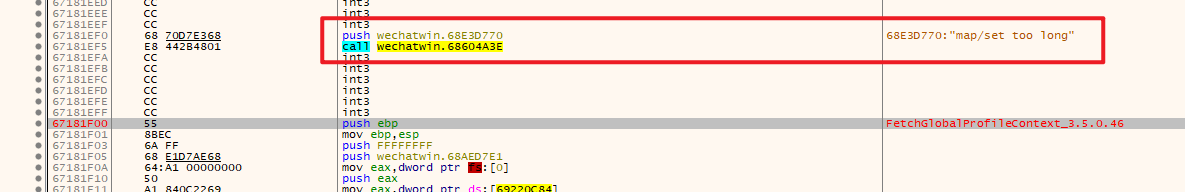
偏移：2

### RVA

wechatwin.dll:$3E4260

## FetchGlobalProfileContext

### 特征



使用”map/set too long”字符串的引用来做定位，但是要注意，需要在这个字符串的字节模式前加上一个0，因为wechatwin.dll还有一个字符串”unordered\_map/set too long”，避免扫描错误，所以在前面加上0。

字符串十六进制流：

{

0x00, 0x6D, 0x61, 0x70, 0x2F, 0x73, 0x65, 0x74, 0x20, 0x74, 0x6F, 0x6F, 0x20, 0x6C, 0x6F, 0x6E, 0x67, 0x00

};

（**warning：因为前面加上了0x00，所以在定位引用位置的时候需要加上1的偏移**）

定位十六进制流：

{

0xCC, 0xCC, 0xCC, 0x55, 0x8B, 0xEC

};

偏移：3

### RVA

wechatwin.dll:$91F00

### 变化

取得ProfileContext之后，wxid、微信号等这些信息的偏移位置都发生了变化

## HandleRawMessages

特征没有变化

### RVA

wechatwin.dll:$4C414D

## HandleReceivedMessages

使用3.4.5.27版本的特征能够定位到一个位置，但是出现了偏差，定位到了HandleReceivedMessages的前一个函数，需要调整特征

### 特征

第一步跟旧版本的一样，定位"NetSceneSync id addMsgListToDB count = %d"字符串的引用位置：

字符串十六进制流：

{

0x4E, 0x65, 0x74, 0x53, 0x63, 0x65, 0x6E, 0x65, 0x53, 0x79, 0x6E, 0x63, 0x20, 0x69, 0x64, 0x20, 0x61, 0x64, 0x64, 0x4D, 0x73, 0x67, 0x4C, 0x69, 0x73, 0x74, 0x54, 0x6F, 0x44, 0x42, 0x20, 0x63, 0x6F, 0x75, 0x6E, 0x74, 0x20, 0x3D, 0x20, 0x25, 0x64, 0x00

};

第二步，定位到这个字符串的函数入口（改动在这一步）

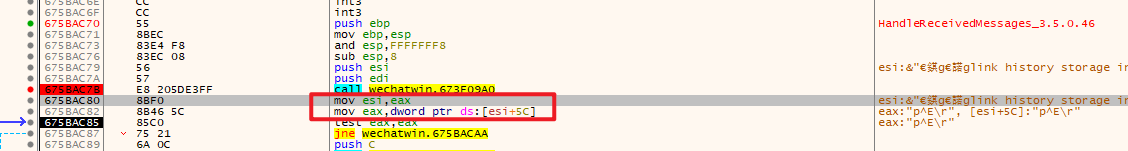
指令十六进制流：

{

0xCC, 0xCC, 0xCC, 0x55, 0x8B, 0xEC

};

第三步，向下定位提取消息结构体地址信息的位置



指令十六进制流：

{

0x8B, 0xF0

};

除了第二步外，第一和第三步跟旧版本的一样

### RVA

wechatwin.dll:$4CAC80

## WXSendTextMessage

使用旧版本的特征能够定位到

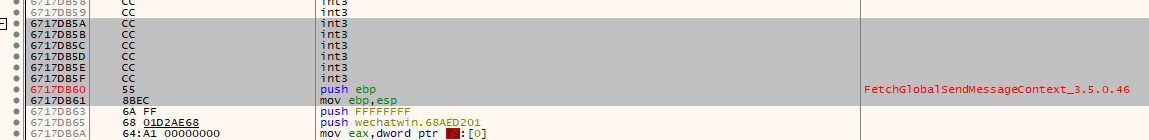
### RVA

wechatwin.dll:$49BC80

## FetchGlobalSendMessageContext

### 特征

它刚好是FetchGlobalContactContextAddress\_3.5.0.46函数的下一个函数，利用FetchGlobalContactContextAddress\_3.5.0.46的特征来做定位：



扫描十六进制流：

{

0xC7, 0x45, 0xF0, 0x00, 0x00, 0x00, 0x00, 0xC7, 0x45, 0xFC, 0x00, 0x00, 0x00, 0x00, 0x8B, 0x75,

0x08, 0x8B, 0xCE, 0x68, 0x02, 0x2E, 0x00, 0x00, 0xC7, 0x45, 0xF0, 0x02, 0x00, 0x00, 0x00, 0xC7,

0x06, 0x00, 0x00, 0x00, 0x00, 0xC7, 0x46, 0x04, 0x00, 0x00, 0x00, 0x00, 0xC7, 0x46, 0x08, 0x00,

0x00, 0x00, 0x00, 0xC7, 0x46, 0x0C, 0x00, 0x00, 0x00, 0x00, 0xC7, 0x46, 0x10, 0x00, 0x00, 0x00,

0x00

};

定位十六进制流：

{

0xCC, 0xCC, 0xCC, 0xCC, 0xCC, 0xCC, 0x55, 0x8B, 0xEC

};

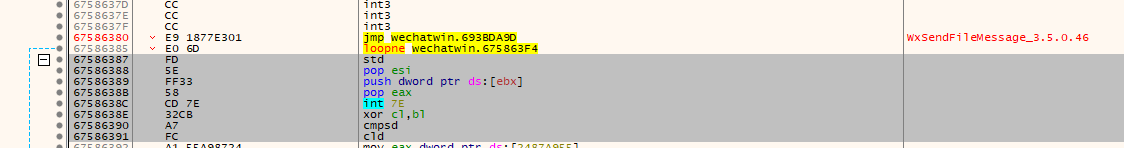
偏移：6字节

### RVA

wechatwin.dll:$8DB60

## WXSendFileMessage

### 特征



使用这一段作为特征然后定位到这个函数的入口位置

扫描十六进制流：

{

0xFD, 0x5E, 0xFF, 0x33, 0x58, 0xCD, 0x7E, 0x32, 0xCB, 0xA7, 0xFC

};

向前定位十六进制流：

{

0xCC, 0xCC, 0xCC, 0xE9

};

偏移：3字节

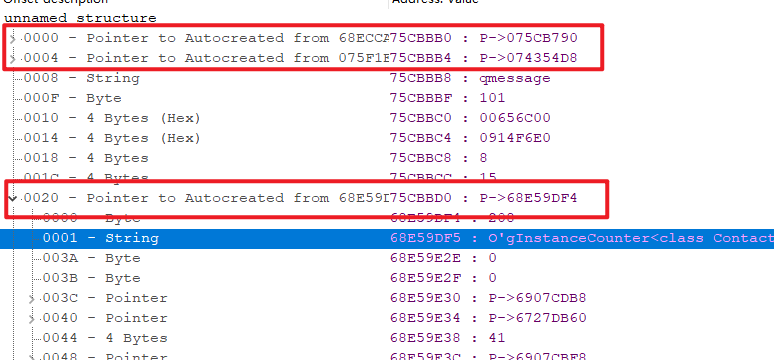
### RVA

wechatwin.dll:$496380

# 数据结构变化

## Contact

获取到ContactContext后，联系人信息的头节点的首地址的偏移变为了0x4C，另外联系人信息的节点结构也发生了变化：



整体缩减了8字节，但是从0x20位置后的实际信息的结构跟之前是一样的

## Profile

取得ProfileContext之后，wxid、微信号等这些信息的偏移位置都发生了变化，具体还没有做记录。