**WxBox - 产品规格说明书**

**(v0.1.2)**

**修订历史记录**

|  |  |  |  |
| --- | --- | --- | --- |
| 修改日期 | 版本  （指的是文档的版本号） | 说明 | 作者 |
| 2021-12-17 | v0.1.1 | 补充使用spdlog日志库 | 胡智钦 |
| 2021-12-29 | v0.1.2 | 补充使用TitanEngine辅助内存分析 | 胡智钦 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目录

[1 引言 5](#_Toc90441356)

[1.1 目的 5](#_Toc90441357)

[1.2 名词定义 5](#_Toc90441358)

[2 非功能性需求 5](#_Toc90441359)

[2.1 运行环境需求 5](#_Toc90441360)

[2.1.1 目标操作系统 5](#_Toc90441361)

[2.1.2 目标处理器 6](#_Toc90441362)

[2.1.3 目标微信客户端版本 6](#_Toc90441363)

[2.2 兼容不同版本的微信客户端 6](#_Toc90441364)

[3 功能性需求 6](#_Toc90441365)

[3.1 模块划分 6](#_Toc90441366)

[3.1.1 WxBox模块 6](#_Toc90441367)

[3.1.2 WxBot模块 6](#_Toc90441368)

[3.1.3 WxBox、WxBot与微信进程的交互模型 7](#_Toc90441369)

[3.2 WxBox模块功能需求 7](#_Toc90441370)

[3.2.1 搜索已启动的微信客户端进程 7](#_Toc90441371)

[3.2.2 获取微信客户端的版本 8](#_Toc90441372)

[3.2.3 检查指定微信客户端是否能够正常使用WxBot 8](#_Toc90441373)

[3.2.4 多开微信 8](#_Toc90441374)

[3.2.5 注入与卸载WxBot模块 8](#_Toc90441375)

[3.2.6 向WxBot模块请求数据 8](#_Toc90441376)

[3.2.7 接收WxBot主动发送的一些消息 9](#_Toc90441377)

[3.3 WxBot模块功能需求 9](#_Toc90441378)

[3.3.1 利用特征自动定位Hook点 9](#_Toc90441379)

[3.3.2 接管各个关键的微信Hook点 9](#_Toc90441380)

[3.3.3 调用微信客户端的功能 9](#_Toc90441381)

[3.3.4 与WxBox通信 10](#_Toc90441382)

[3.3.5 实现Plugin 11](#_Toc90441383)

[4 WxBox界面原型 12](#_Toc90441384)

[5 技术选型 13](#_Toc90441385)

[5.1 界面开发 13](#_Toc90441386)

[5.2 脚本引擎 13](#_Toc90441387)

[5.3 IPC 13](#_Toc90441388)

[5.4 指定定位、内存扫描与Hook 13](#_Toc90441389)

[6 开发周期 13](#_Toc90441390)

# 引言

WxBox项目的目的是要为“**PC微信客户端**”做一个“**微信机器人**”，该机器人主要的用处是监控传递到“**PC微信客户端**”的消息，然后“**微信机器人**”会识别消息的内容，做出响应，响应的结果可能是简单的回馈某些信息，也可能是执行安装到WxBox中的某个“**脚本**”。

## 目的

本产品规格说明书的目的是对WxBox项目的整体做一个简单的概述，描述机器人需要实现哪些功能、所组成的模块、整体架构、技术选型等内容。文档不会涉及如何具体的实现，详细的细节留到之后讨论过再补充一个文档或者做口头约定。

## 名词定义

|  |  |
| --- | --- |
| 名词 | 说明 |
| WxBox | 指主体程序，中文叫“微信盒子” |
| WxBot | 这是由WxBox向微信进程注入的一个模块，用以实现“微信机器人”的主要功能 |
| Plugins（脚本/扩展） | wxbot可以拥有脚本/扩展功能 |
| Hook点 | “微信机器人”需要Hook微信进程内某些函数的入口，这些入口位置我们称为Hook点 |
| 特征 | 这里的特征指的是Hook点的定位特征，用于对未知版本的微信做定位 |
| 自动定位 | 对于未知版本的微信，Hook点是未知的，需要用我们提前提取的特征来自动定位，并检测Hook点是否可用 |

# 非功能性需求

## 运行环境需求

### 目标操作系统

WxBox目标能兼容Windows 10之后的Windows操作系统，还有Mac OS。

### 目标处理器

只要兼容Intel指令集的就足够了。在Windows上微信客户端是32位，但是在Mac上微信客户端是32位的还是64位的进程还不知道，需要钟总到时用file命令看看，如果是64位的那么对项目在提取特征和利用特征会有点影响。

### 目标微信客户端版本

希望兼容不同版本的微信客户端，对于未知版本的我们使用特征自动定位，但对于已知的版本，我们用手动的方式收集并提供给WxBox。目前已知的版本是：Windows（3.4.5）还有Mac（3.2.2）。

## 兼容不同版本的微信客户端

由于微信的机器人，需要向微信进程注入指令来接管某些操作的入口或者主动去调用微信某些API，为此PC微信客户端在内存中的一些数据结构还有Hook点必须得精确，否则会导致功能不正常，严重的话会导致微信进程直接崩溃。为了能让这个微信机器人能支持不同的操作系统，还有不同版本的微信客户端，需要对微信的一些操作调用入口提取特征，用以在微信客户端版本更新时能够快速定位Hook点，或者让WxBox直接尝试使用这些特征去自动定位Hook点，关于这些特征，在另外一个文档有详细说明。

Windows平台的更新周期大概是两到三个月，而Mac平台大概是三到四个月，这个可以看[微信更新日志](https://weixin.qq.com/cgi-bin/readtemplate?lang=zh_CN&t=weixin_faq_list&pass_ticket=723JQxfx6liHClP9cfdRLZ0VVEMmdzNqdVME5QpQuY%2FzqQkLGFBabnpE3nHb6D76)。

# 功能性需求

## 模块划分

WxBox整个项目会划分出两个模块：WxBox和WxBot。其中WxBox定义为“**微信盒子**”，而WxBot定义为“**微信机器人**”。

### WxBox模块

这个模块即“**微信盒子**”，是项目的主体程序，这是一个窗体程序，在Windows上以exe的形式发布。主要负责将WxBot模块注入进目标的微信进程上，或者把WxBot模块从目标微信进程上卸载掉。另外还要通过WxBox来多开微信、获取微信的联系人信息等操作。

### WxBot模块

WxBox项目的主要功能是依赖这个WxBot模块来完成，在Windows和Mac OS上分别以dll和so模块的形式发布。WxBot模块用来实现“**微信机器人**”，接管微信进程上一些关键的Hook点，实现WxBox的Plugin引擎等功能。另外WxBot模块是不依赖WxBox的，也就是说当WxBox把WxBot模块注入到目标微信进程后，它是可以脱离WxBox使用的，这时候把WxBox给关闭掉，也不影响WxBot的使用。

它有两种注入模式：直接定位注入模式和自动定位注入模式。

#### 直接定位注入模式

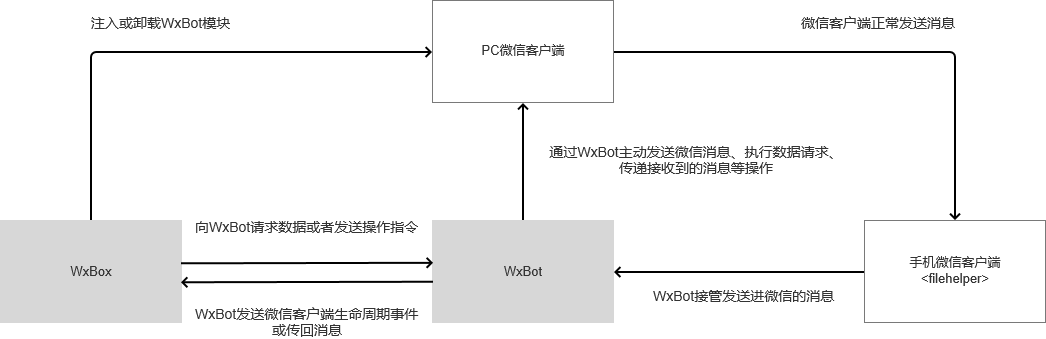
这是对已知的微信客户端版本使用的注入模式，对于已知版本的微信，所有Hook点是清楚的，不需要再去利用特征来定位Hook点。这个模式导致功能异常还有导致微信进程崩溃的可能性更低。

#### 自动定位注入模式

这是对未知的微信客户端版本使用的注入模式，对于未知版本的微信，所有Hook点是需要利用特征去尝试定位，如果定位失败那么也就是说注入WxBot模块会失败。另外由于所有自动定位寻找到的Hook点状况都是未知的，所以很有可能导致功能异常还有微信进程的崩溃。

### WxBox、WxBot与微信进程的交互模型

这里主要以手机微信给“文件传输助手”发送消息的模型为例子来描述。



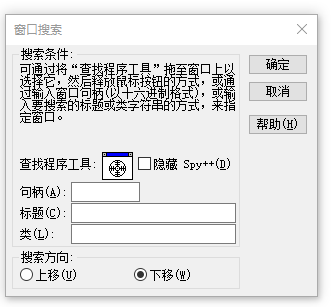
【WxBox、WxBot、PC微信客户端以及手机微信客户端的交互模型】

## WxBox模块功能需求

### 搜索已启动的微信客户端进程

通过两种方式来搜索已启动的微信客户端进程。

1. 扫描系统上所有打开的进程，搜索微信的主进程；
2. 使用类似Visual Studio Spyxx的窗口搜索功能，根据选中的窗口来获取所在的进程来判断是否为微信的主进程，这个功能在spyxx上如下图：



【spyxx上的窗口搜索功能】

### 获取微信客户端的版本

这里分两种情况，一种是未启动的微信，一种是已经启动的微信进程。方法都一样，在Windows操作系统上，方法是找到WeChat.exe所在的路径然后去获取WeChatWin.dll的文件版本，在Mac OS上估计差不多。

### 检查指定微信客户端是否能够正常使用WxBot

当微信客户端的版本是我们的已知微信版本时这个是能确定可以使用的。对于未知的微信客户端版本，尝试用“自动定位注入模式”来把WxBot模块注入进去，然后等待反馈。

### 多开微信

正常情况下，PC上的微信客户端是只能有一个实例存在的，因为程序内左右检查。这个功能使用WxBox来启动一个微信进程，在微信启动时需要让WxBox把微信进程给Attach住，在Windows上，需要等微信进程把WeChatWin.dll加载进内存后，WxBox暂停微信进程的运行，然后把微信多开的限制给解除，然后再取消Attach。就像是给微信进程挂在了调试器上修改了内存，然后继续运行起来。

### 注入与卸载WxBot模块

不管是已知还是未知的微信客户端版本，都需要用WxBox把WxBot模块给注入到目标微信进程上，WxBot自己会根据WxBox传过来的参数完成初始化。对于包含的参数，例如有plugins目录的绝对路径，等等。

### 向WxBot模块请求数据

当WxBot正常运作的时候，WxBox可以通过IPC调用来向WxBot请求其挂载的微信的一些数据或者执行一些操作。例如：向WxBot请求其登录的微信的所有联系人的信息。

### 接收WxBot主动发送的一些消息

这个主要包括WxBot监视下的微信的生命周期。

## WxBot模块功能需求

### 利用特征自动定位Hook点

对于未知版本的微信因为Hook点是未知的，所以需要利用特征去做定位，这个过程和结果未必会成功，甚至WxBot不能知道尝试的结果是成功还是失败。

### 接管各个关键的微信Hook点

WxBot需要Hook的位置主要包括了，微信的生命周期还有接收消息的位置。

#### 需要接管的微信生命周期

对于这些消息WxBot会主动发送给WxBox（如果WxBox还存活的话）。

1. 微信账户登录成功；
2. 微信账户退出登录成功；
3. 微信进程正常退出。

#### 接收消息

当PC微信客户端接收到消息之后，WxBot需要第一时间Hook住，并且需要识别消息的类型、消息的发送者、消息的文本、对于群聊还需要知道发送消息的成员是谁。

### 调用微信客户端的功能

主要是利用已知或者自动定位找到的一些微信API的入口来实现的。

#### 检查微信是否为登录状态

WxBot的一些功能需要在微信正常登录后才能使用，所以这个功能是必不可少的。

#### 获取微信账户的Profile个人信息

获取登录账户的个人Profile信息，包括：1）wxid；2）微信号。其它信息现在先不管，因为没啥必要。

#### 获取所有联系人的信息

获取所有联系人的wxid、微信号、昵称、备注等信息。这里的所有联系人其实还包含了群聊的，它们在微信内存中是放在一起存储的。

#### 获取所有群聊联系方式

独立获取所有群聊的联系方式，主要用来获取群聊的wxid还有群聊名。

#### 根据wxid获取对应的联系人的信息

根据wxid来获取对应联系人的微信号、昵称、备注等信息。

#### 主动发送消息

主动给账户中的联系人发送消息。

##### 给单个对象发送消息

能发送的消息类型有：1）文本消息；2）图片；3）文件。

##### 在群聊中发送消息

能发送的消息类型有：1）文本消息；2）图片；3）文件；4）@群聊中的成员。

### 与WxBox通信

能够实现WxBox之间的双向通信，WxBot有时候需要给WxBox主动发送一些有关生命周期的事件消息。而WxBox有时候会主动向WxBot请求数据或者请求执行操作。

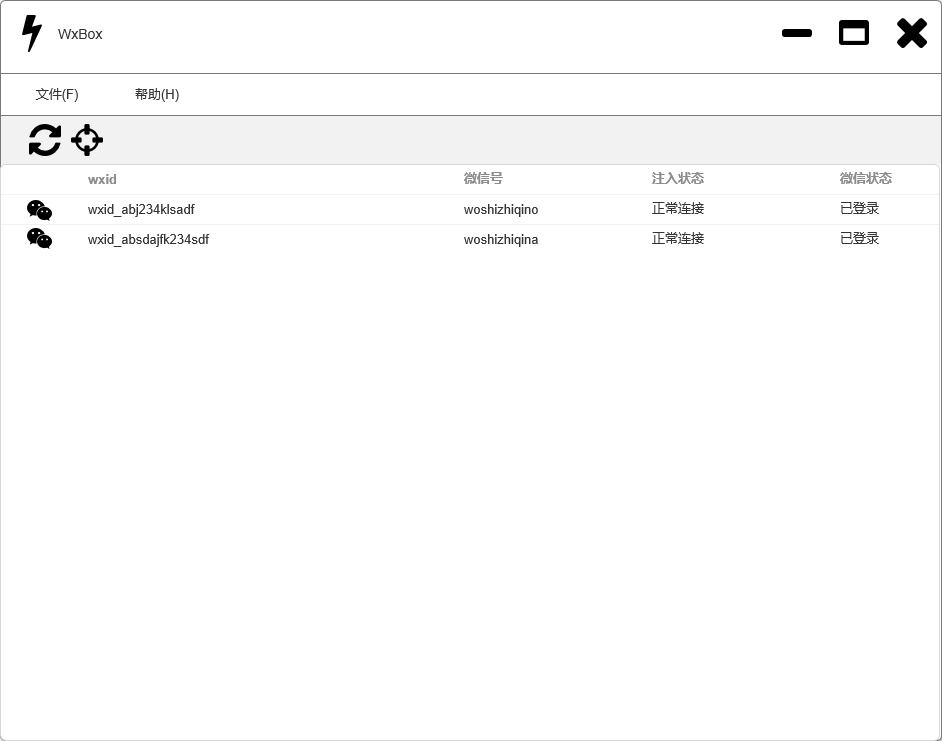
### 实现Plugin

这个功能是“微信机器人”的核心功能，每个Plugin都是以lua脚本的形式存在的。在WxBox的安装目录下，会有一个plugins目录，里面放着lua脚本，每个lua脚本对应着一个Plugin。对于每个WxBot都能相互隔离的调用不同的Plugin，互不影响。

一个Plugin的调用方式的例子是：我用手机微信给“文件传输助手”发送一条“wxbotplugin://shutdown\_my\_computer”文本消息，那么WxBot识别到这条消息后，会去plugins目录寻找是否有一个shutdown\_my\_computer.lua的扩展，如果有的话就会加载这个脚本来执行。

Plugin具体要怎么实现，要实现到什么程度现在还未定，留到沟通过后再决定。

# WxBox界面原型



【WxBox界面原型】

# 技术选型

## 界面开发

可选择Qt或者.NET（用mono来实现跨平台）

## 脚本引擎

lua

## IPC

IPC库选用gRPC，仓库地址：[grpc/grpc](https://github.com/grpc/grpc)。

传输数据格式选用protobuf，仓库地址：[protocolbuffers/protobuf](https://github.com/protocolbuffers/protobuf)。

## 指令定位、内存扫描与Hook

选用frida-gum API。可看看这两个文档：[frida官方文档](https://frida.re/docs/home/)、[frida C API文档](https://frida.re/docs/c-api/)。

frida仓库地址：[frida/frida](https://github.com/frida/frida)。

frida-gum仓库地址：[frida/frida-gum](https://github.com/frida/frida-gum)。

另外程序的动态分析使用TitanEngine来辅助，TitanEngine仓库地址：[x64dbg/TitanEngine](https://github.com/x64dbg/TitanEngine)。它的依赖方式是，把它编译好的lib、dll打包做成wrap包。

另外这里补充下Hook的原理。Hook的原理简单来说是这样的，我们需要Hook的东西是可运行的指令，指令所在的内存段一般是只读保护的，不可修改。要Hook某个函数的话，常见做法是向目标的进程注入我们自己的代码，在代码中修改需要Hook指令所在的内存的访问属性为可写，然后把原先的指令拷贝一份备份起来，然后修改它，让它直接跳转到我们代码的入口，我们的代码执行完之后，再执行原先备份的那一条指令，执行完之后再跳转到原先Hook指令的下一条指令继续执行。由于Windows和Mac中实现Hook需要调用的系统API是不一样，为了用相同的接口实现，所以我们选用frida这个第三方的库。

## 构建工具

使用meson+ninja组合。meson跟ninja的关系类比tarball的话，meson就是configure，ninja就是make。

meson仓库地址：[mesonbuild/meson](https://github.com/mesonbuild/meson)。

meson官方文档：[mesonbuild](https://mesonbuild.com/)。

## 日志库

选用spdlog，仓库地址：[gabime/spdlog](https://github.com/gabime/spdlog)。

# 开发周期

这个项目是一个休闲项目，周期不定，有空的时候就写，没空的时候放着也没关系。

have a good time！！！！！！！！

^\_^