# 免越狱版 iOS 抢红包插件

Dec 26, 2016

CONTE
- 原理
- 获取砸壳
- 准备 dylil
- 检查依赖
- 将动态链制文件中
- 打包并重
- 安装
- 效果
- 小结
- 参考文章
- 赞赏

又到年末，微信红包又开始成为大家所关心的话题了，不管是公司年会，还是朋友聚会，似乎不发红包就没办法继续聊下去了。因此，值此新年来临之际，我对我的iOS 微信抢红包 tweak进行了一下改进。主要增加了插件开关，以及随机延迟功能，让你在新一轮红包大战中无往而不利。

但是，这毕竟是一个 tweak，只有少数有越狱机器的小伙伴才能使用这个插件，无疑门槛是太高了。到目前为止，已经有无数朋友在问到底有没有免越狱版本的插件了。

今天，我们就要讨论下如果制作免越狱版本的微信抢红包插件。

> 注意：本篇文章只讲我在自己摸索过程中的一点总结，并不深入讲解原理。我对 iOS 逆向也只是初窥门径，对很多原理也还未达到深入理解的程序，国内关于这方面的文章也是在少数，我也不希望我写出语焉不详的文章导致读者被我误导。同时，也不排除将来我研究得够深入的时候，再回过头来写关于原理的文章。

## § 原理

虽然这里并不深入讲解，但是最基本的原理我们还是要理解的，因为后面所做的工作，都是基于这个原理来进行开发的。越狱机器之所以能使用 tweak，主要是因为在越狱的时候，手机里就安装了 mobilesubstrate 这个库，这个库的作用就是能在程序运行的时候动态加载我们自己写的 dylib 动态运行库。而由于非越狱手机系统里面是没有这个库的，所以我们需要直接将这个库打包进 ipa 当中，使用它的 API 实现注入。

关于注入的原理，可以参考这篇文章：移动App入侵与逆向破解技术－iOS篇。

好了，这就是目前为止我们所需要了解的东西。下面就可以开始动手了。

## § 获取砸壳版本的微信 ipa

因为在 AppStore 上面下载得到的应用都是经过加密的，可以执行文件上已经被加过一层壳，所以我们需要先拿到砸过壳版本的微信应用。

有两种方法：

1. 直接在 PP 助手下载
2. 使用 Clutch 对越狱手机上应用进行砸壳

第一种方法没什么好说的，这里主要讲讲第二种方法。

首先，将 Cluth 仓库 clone 到本地：

**Bash**

```
$ git clone https://github.com/KJCracks/Clutch
$ cd Clutch
```

接着，使用 Xcode 进行构建，得到可执行文件：

**Bash**

```
$ xcodebuild -project Clutch.xcodeproj -configuration Release ARCHS="armv7 armv7s arm64" build
```

生成出来的可执行文件就在 Clutch 目录下，将其拷贝到手机上：

**Bash**

```
scp Clutch/clutch root@<your.device.ip>:/usr/bin/
```

之后，就可以使用这个工具来进行砸壳了。

先 ssh 到越狱手机上，然后列出当前安装的应用：

Bash

```
$ ssh root@<your.device.ip>
$ clutch -i

# Installed apps:
# 1:   WeChat <com.tencent.xin>
# 2:   DingTalk <com.laiwang.DingTalk>
# 3:   喜马拉雅FM（听书社区）电台有声小说相声英语 <com.gemd.iting>
```

可以得到 clutch 把相应的包名也显示出来了，这时我们就可以进行砸壳了：

Bash

```
$ clutch -d com.tencent.xin

# com.tencent.xin contains watchOS 2 compatible application. It's not possible to dump watch(
# Zipping WeChat.app
# Swapping architectures..
# ASLR slide: 0xb3000
# ...
# writing new checksum
# DONE: /private/var/mobile/Documents/Dumped/com.tencent.xin-iOS7.0-(Clutch-2.0.4).ipa
# Finished dumping com.tencent.xin in 76.9 seconds
```

可以看到，clutch 将砸过后的 ipa 文件放到了 `/private/var/mobile/Documents/Dumped/` 目录下。

我们将它改成一个比较简单的名字，然后拷回电脑上：

Bash

```
$ mv /private/var/mobile/Documents/Dumped/com.tencent.xin-iOS7.0-\(Clutch-2.0.4\).ipa /private
$ scp root@<your.device.ip>:/private/var/mobile/Documents/Dumped/wechat.ipa ~/Desktop
```

这里，我把它拷到了电脑桌面上。之后就可以进行下一步操作了。

## § 准备 dylib 动态链接库

这步就很简单了，直接到我的 WeChatRedEnvelop 上把源码 clone 下来，然后执行 `make` 命令，就能拿到 dylib 文件了。

Bash

```
$ git clone https://github.com/buginux/WeChatRedEnvelop.git
$ cd WeChatRedEnvelop
$ make

# > Making all for tweak WeChatRedEnvelop…
# ==> Preprocessing Tweak.xm…
# ==> Compiling Tweak.xm (armv7)…
# ==> Compiling XGPayingViewController.m (armv7)…
# ...
# ==> Signing WeChatRedEnvelop…

$ cp .theos/obj/debug/WeChatRedEnvelop.dylib ~/Desktop # 注意是 .theos 目录，这是个隐藏目录
```

将生成的 dylib 文件拷贝到桌面，跟刚刚砸过壳的微信应用放到一个目录层级。

## § 检查依赖项

因为这个代码是我自己写的，所以我知道这个 dylib 除了 mobilesustrate 外没有依赖其它的库。但是如果我们拿到的是别人的 dylib，就需要先进行一下依赖项检查，以确保之后我们将所有的依赖库都打包进 ipa 当中。

使用 macOS 自带的 otool 工具就可以进行依赖项检查：

Bash

```
$ otool -L WeChatRedEnvelop.dylib
WeChatRedEnvelop.dylib (architecture armv7):
        /Library/MobileSubstrate/DynamicLibraries/WeChatRedEnvelop.dylib (compatibility vers
        /usr/lib/libobjc.A.dylib (compatibility version 1.0.0, current version 228.0.0)
        /System/Library/Frameworks/Foundation.framework/Foundation (compatibility version 300
        /System/Library/Frameworks/CoreFoundation.framework/CoreFoundation (compatibility ver
        /System/Library/Frameworks/UIKit.framework/UIKit (compatibility version 1.0.0, curre
        /usr/lib/libsubstrate.dylib (compatibility version 0.0.0, current version 0.0.0)
        /usr/lib/libc++.1.dylib (compatibility version 1.0.0, current version 307.4.0)
        /usr/lib/libSystem.B.dylib (compatibility version 1.0.0, current version 1238.0.0)
WeChatRedEnvelop.dylib (architecture arm64):
        /Library/MobileSubstrate/DynamicLibraries/WeChatRedEnvelop.dylib (compatibility vers
        /usr/lib/libobjc.A.dylib (compatibility version 1.0.0, current version 228.0.0)
        /System/Library/Frameworks/Foundation.framework/Foundation (compatibility version 300
        /System/Library/Frameworks/CoreFoundation.framework/CoreFoundation (compatibility ver
        /System/Library/Frameworks/UIKit.framework/UIKit (compatibility version 1.0.0, curre
        /usr/lib/libsubstrate.dylib (compatibility version 0.0.0, current version 0.0.0)
        /usr/lib/libc++.1.dylib (compatibility version 1.0.0, current version 307.4.0)
        /usr/lib/libSystem.B.dylib (compatibility version 1.0.0, current version 1238.0.0)
```

可以看到除了 substrate 库，其它依赖的都是系统自带的库。我们将 libsubstrate.dylib 拷出，使用 `install_name_tool` 命令修改动态库的路径，指向 app 二进制文件的同级目录。

> 如果你的系统中不是 /usr/lib/libsubstrate.dylib 而是 /Library/Frameworks/CydiaSubstrate.framework/CydiaSubstrate 的话，解决方法可以参考 Github 上的 issue。

**Bash**

```
$ scp root@<your.device.ip>:/usr/lib/libsubstrate.dylib ~/Desktop
$ install_name_tool -change /usr/lib/libsubstrate.dylib @loader_path/libsubstrate.dylib WeCh
$ otool -L WeChatRedEnvelop.dylib
WeChatRedEnvelop.dylib (architecture armv7):
        /Library/MobileSubstrate/DynamicLibraries/WeChatRedEnvelop.dylib (compatibility vers
        /usr/lib/libobjc.A.dylib (compatibility version 1.0.0, current version 228.0.0)
        /System/Library/Frameworks/Foundation.framework/Foundation (compatibility version 300
        /System/Library/Frameworks/CoreFoundation.framework/CoreFoundation (compatibility ver
        /System/Library/Frameworks/UIKit.framework/UIKit (compatibility version 1.0.0, curre
        @loader_path/libsubstrate.dylib (compatibility version 0.0.0, current version 0.0.0)
        /usr/lib/libc++.1.dylib (compatibility version 1.0.0, current version 307.4.0)
        /usr/lib/libSystem.B.dylib (compatibility version 1.0.0, current version 1238.0.0)
WeChatRedEnvelop.dylib (architecture arm64):
        /Library/MobileSubstrate/DynamicLibraries/WeChatRedEnvelop.dylib (compatibility vers
        /usr/lib/libobjc.A.dylib (compatibility version 1.0.0, current version 228.0.0)
        /System/Library/Frameworks/Foundation.framework/Foundation (compatibility version 300
        /System/Library/Frameworks/CoreFoundation.framework/CoreFoundation (compatibility ver
        /System/Library/Frameworks/UIKit.framework/UIKit (compatibility version 1.0.0, curre
        @loader_path/libsubstrate.dylib (compatibility version 0.0.0, current version 0.0.0)
        /usr/lib/libc++.1.dylib (compatibility version 1.0.0, current version 307.4.0)
        /usr/lib/libSystem.B.dylib (compatibility version 1.0.0, current version 1238.0.0)
```

可以看到 libsubstrate.dylib 的路径已经变更了。

## § 将动态链接库注入二进制文件中

接下来，就需要将我们的库注入到微信的二进制文件中，可以使用开源的 optool 工具。

编译安装 optool 工具：

**Bash**

```
# 因为 optool 添加了 submodule，因为需要使用 --recuresive 选项，将子模块全部 clone 下来
$ git clone --recursive https://github.com/alexzielenski/optool.git
$ cd optool
$ xcodebuild -project optool.xcodeproj -configuration Release ARCHS="x86_64" build
```

> 如果碰到类似 “error: There is no SDK with the name or path ‘/path/to/optool/macosx10.9’” 的错误，请使用 Xcode 打开工程，在 Build Setting 中选择正确的 SDK 版本。

将砸壳过的 ipa 文件解压，然后将 libsubstrate.dylib 与 WeChatRedEnvelop.dylib 拷贝到解压后的 WeChat.app 目录下。

```bash
$ cd ~/Desktop
$ unzip wechat.ipa -d wechat
$ cp libsubstrate.dylib WeChatRedEnvelop.dylib wechat/Payload/WeChat.app
```

使用 optool 把 WeChatRedEnvelop.dylib 注入到二进制文件中：

```
$ /path/to/optool install -c load -p "@executable_path/WeChatRedEnvelop.dylib" -t wechat/Payload
```

> 在开始打包之前，请先将 WeChat.app 里面的 Watch 目录删除，这个目录是跟 Watch 有关的，如果不删除的话，会导致后继的安装步骤出问题。出现 `A WatchKit app within this app is not a valid bundle` 的错误。

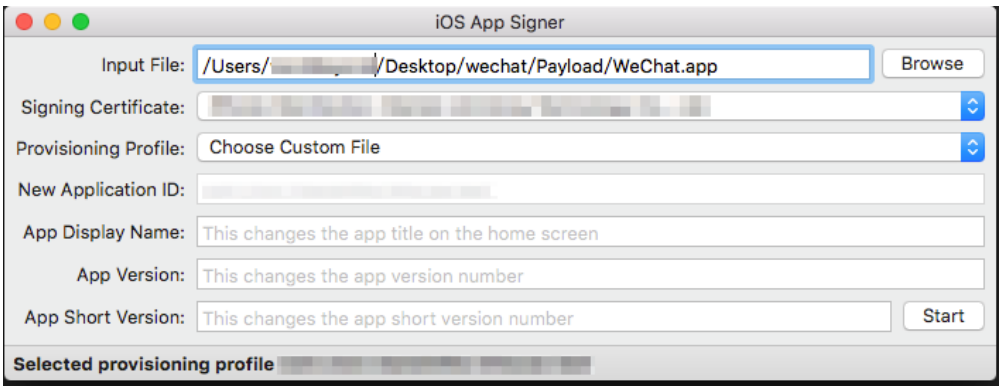
## § 打包并重签名

打包 ipa 与重签名可以直接使用图形化工具 ios-app-signer 来完成。

这个工具可以自动加载出本机的证书以及 Provisioning Profile 文件，使用起来十分方便，当然也可以手动选择证书文件。

> 如果是使用个人开发者证书，需要先将设备的 UUID 加到 Provisioning Profile 中。

我这里是直接从一个 archive 过的应用中提取 embedded.mobileprovision 文件，并在 Provisioning Profile 一栏中选择 Choose Custom File 使用这个文件。



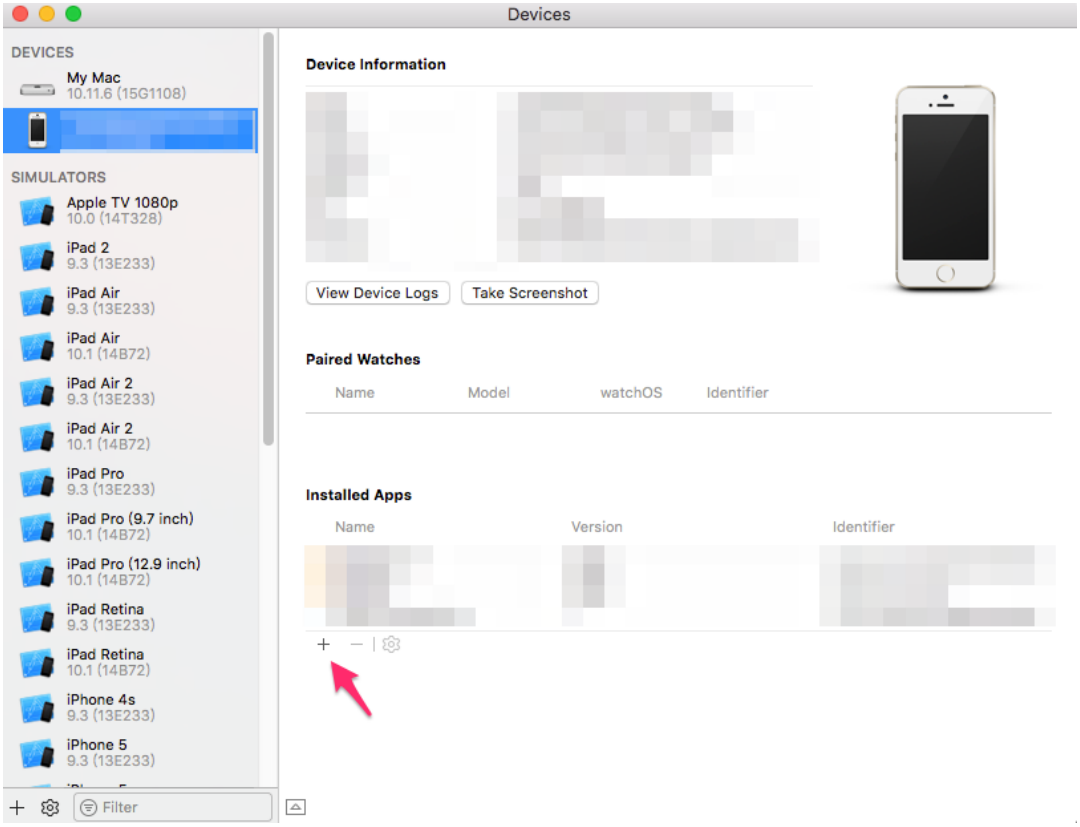

点击 start 后，指定保存路径，iOS App Signer 就会帮你搞定所有事情。

## § 安装

在 iOS App Signer 完成打包与重签名后，我们就可以进行安装了。

安装有两种方法，一种是使用 iTool 工具安装 ipa 文件。

另外一种是针对有 XCode 的同学，在菜单 - Window - Devices 中打开设置窗口。点击 Installed Apps 栏下面的 + 号就可以选择 ipa 文件进行安装了。




## § 效果

重签名后的应用由于与原应用的 Bundle id 是不同的，所以可以同时安装两个应用。这也就是淘宝上所谓的微信多开的原理。


## § 小结

我们最终得到了一个可以安装在免越狱设备上的微信 ipa 文件，但是这些步骤对于有些来说，可能还是太过复杂了。他们需要的只是重签名过的 ipa 文件，但是很可惜，由于我是使用的个人证书打包的，所以无法直接将 ipa 文件发出来。

如果有哪位好心的小伙伴可以贡献一个企业证书的话，我就可以利用这个企业证书打个包，然后直接进行应用分发，这也许就是最方便快捷的方法了。

## § 参考文章

- 移动App入侵与逆向破解技术－iOS篇5
- DingTalkNoJailTweak

## § 赞赏

如果本篇文章对你有帮助，可以进行小额赞助，鼓励作者写出更好的文章。