逆向分析网络协议 iOS 篇

2016-03-09 txx 糖炒小虾

前言

上一篇文章介绍了 Android 通过逆向＋动态断点的方式获取签名规则的手法。看到有童鞋留言说想看 iOS 该怎么做。那我就分享一下我的做法给大家。

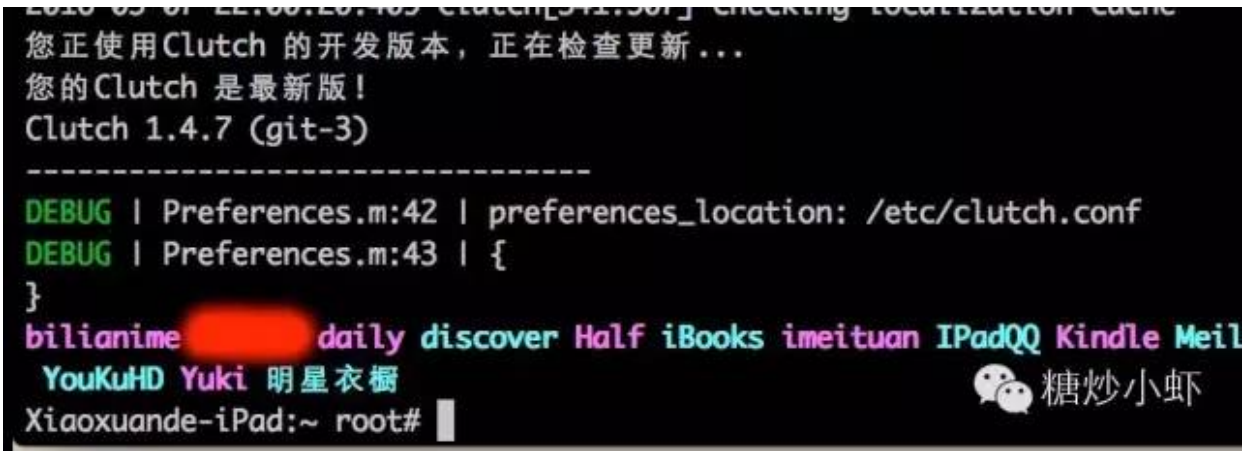
砸壳

我们知道，从 AppStore 下载的 ipa 是有加密的，没法乱搞。所以逆向的第一步就要砸壳。

砸壳说的很高深，但实际上很简单，毕竟前人已经帮我们做好了，`Clutch` 非常好用。

首先我们需要一台越狱机，之后可以找 `Cydia` 的源装 `Clutch`，也可以直接在 `https://github.com/KJCracks/Clutch` 上下载可执行文件后 `scp` 到手机上安装。

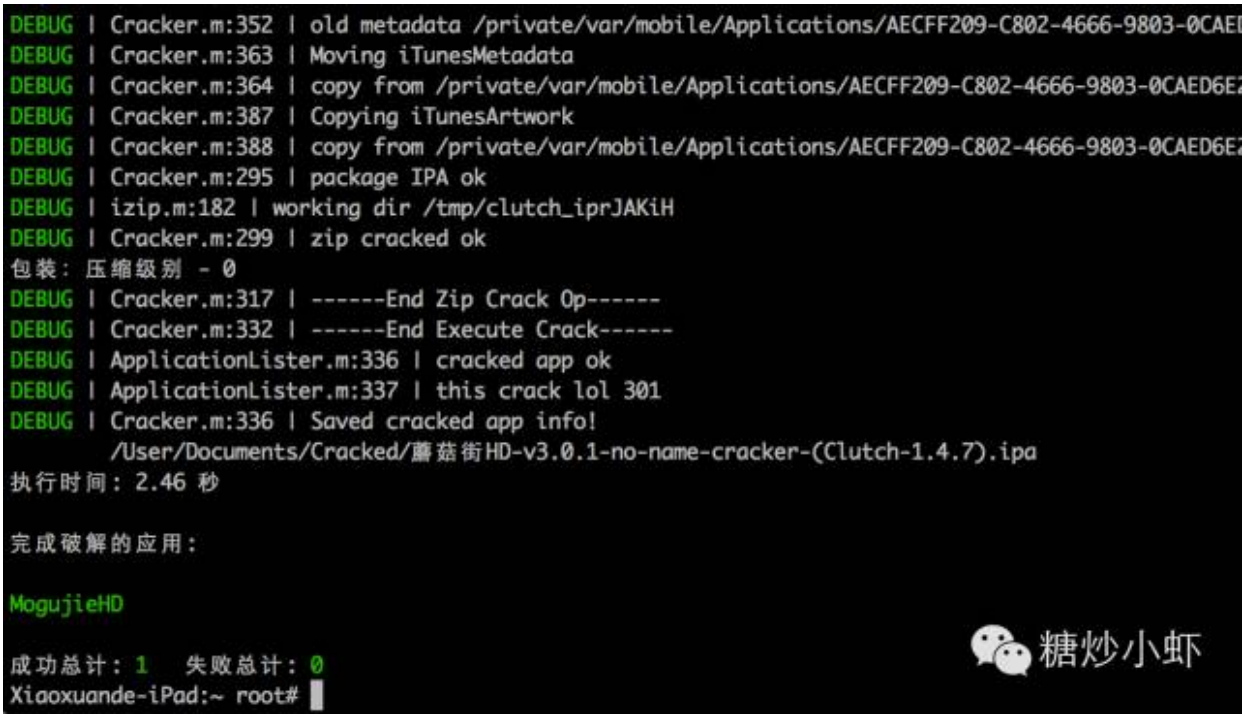
通过 ssh 手机或者用 MobileTerminal，执行 Clutch。就可以看到本地安装的各种 APP 们。



假设我们今天要分析 XApp, 所以就

```
Clutch XApp
```

如果成功的话，就看到这样的类似物。（由于不想露出来我到底逆向了谁，这里就用蘑菇街代替吧，Limboy 大大别打我）



根据日志中提示的路径，可用 `ForkLift` 之类的工具把他下到本地。

ida 逆向分析

逆向分析一个相对成熟的项目，找到突破点是很重要的。

我们根据抓包得到的签名格式上看，那个 `V2.0` + md5，非常显眼。于是 V2.0 就成了我们重点关注对象。

在 IDA 中 `Shift+F12`，打开 Strings，搜索 2.0

V2.0%@ 明显是 `stringWithFormat` 的参数，跟进去看一下。发现一个叫做

```
[BaseEngine RunRequest:path:useSSL:completionHandler:errorHandler:finish
```

引用了他，这个类名看起来非常像我们找的地方。打开这个类，F5 一下。

可以看到，`v95` 就是 `md5` 字符串了。从上文得知 `v95` 是从 `md5DecodingString` 这个方法算出来的。于是我们只需要在 `md5DecodingString` 这里 Hook 一下，就能知道原文是什么了。

这里吐槽一句，为什么是 md5 的方法叫做 Decoding 而不是 Encoding…

在方法列表里搜一下，会发现 `md5DecodingString` 是 `NSString-Extension.h` 这个 `Category` 的一个方法。

于是我们仅需在此处下一个断点，就能得到原文了。

LLDB Remote Debug

上面是静态分析，下面讲的就是动态调试了。

Debug Server 配置

debugServer 顾名思义，这个是用来做远程调试服务器的，而这个文件其实大伙儿已经有了。

我们有两个途径拿到他：

1. 用 `scp` 从 iDevice `/Developer/usr/bin/debugserver` 目录下获得
2. 从 `/Applications/Xcode.app/Contents/Developer/Platforms/iPhoneOS.platform/ DeviceSupport/7.0.3\ \(11B508\)/DeveloperDiskImage.dmg` 获得，中间的那个 `DeviceSupport` 可以根据具体路径替换。

拿到它之后并不能直接使用，因为默认是没有权限的。

我们要给他 `task_for_pid` 的权限，书写 `entitlements.plist` 如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.app
<plist version="1.0">
<dict>
	<key>com.apple.springboard.debugapplications</key>
	<true/>
	<key>run-unsigned-code</key>
	<true/>
	<key>get-task-allow</key>
	<true/>
	<key>task_for_pid-allow</key>
	<true/>
</dict>
</plist>
```

签名：`codesign -s - --entitlements entitlements.plist -f debugserver`

之后，把这个传回 iDevice `/usr/bin/debugserver`。

lldb 挂载

在要挂载的进程存在的时候，我们在 ssh 中执行

```
debugserver *:8888 -a XApp
```

这时就会发现 XApp，已经被断点了，一切都静止了。

在本地命令行中执行:

```
> lldb
> platform select remote-ios
> process connect connect://iOSIP:8888
```

过一会儿就会看到

```
* thread #1: tid = 0x17f0, 0x0000000196d95ca0 libsystem_kernel.dylib`mac
```

表示我们 lldb 已经挂载成功了

Address 断点

正常的 App Store 项目中，是没有符号的。所以我们常用的符号断点是不行的，但是我们可以给函数的内存地址下断点。

Class-dump

我们先要知道目标函数的地址是什么。这个就需要 `Class-Dump` 了。

```
Class-Dump XApp -H -A -S -o headers/
```

这句话翻译一下，把 XApp dump出头文件，并标记 IMP 的地址，方法排序输出到 headers 文件夹。

我们在 headers 中就可以找到刚才说的 NSString-Extension.h。

```
#import "NSString.h"
@interface NSString (Extension) //节选
```

```
- (id)md5DecodingString;    // IMP=0x000000010024b99c
- (id)md5StringFor16;    // IMP=0x000000010024b9b8
- (id)objectFromJSONString;    // IMP=0x000000010024d078

@end
```

于是 IMP 的地址就是: `0x000000010024b99c`

获取 ASLR 偏移量

> ASLR（Address space layout randomization）是一种针对缓冲区溢出的安全保护技术，通过对堆、栈、共享库映射等线性区布局的随机化，通过增加攻击者预测目的地址的难度，防止攻击者直接定位攻击代码位置，达到阻止溢出攻击的目的。据研究表明ASLR可以有效的降低缓冲区溢出攻击的成功率，如今Linux、FreeBSD、Windows等主流操作系统都已采用了该技术。

我们通过 image 方法获取偏移量

```
image list -o -f
```

得到

```
[  0] 0x00000000000f8000 /var/mobile/Applications/27C2DBB3-34A3-4B71-91F
```

其中，0x00000000000f8000 就是我们想要的偏移量。

断点

终于跑到了最后一步，根据我们拿到的地址下断点:

```
br s -a "0x00000000000f8000+0x000000010024b99c"
```

然后提示:

```
Breakpoint 1: where = XApp`___lldb_unnamed_function11956$$XApp, address
```

表示断点下完了，我们执行 `c` 或 `continue` 让 `app` 跑起来，触发一个网络请求，就会发现断点停下来的。

```
* thread #1: tid = 0x17f0, 0x000000010034399c XApp`___lldb_unnamed_funct
    frame #0: 0x000000010034399c XApp`___lldb_unnamed_function11956$$XAp
XApp`___lldb_unnamed_function11956$$XApp:
->  0x10034399c <+0>:  mov    x8, x0
    0x1003439a0 <+4>:  adrp   x9, 1160
    0x1003439a4 <+8>:  ldr    x0, [x9, #3368]
    0x1003439a8 <+12>: adrp   x9, 1152
```

由于 OC 方法 $arg1 就是 self，这里的 self 是 NSString，所以仅需

```
po $arg1
```

就得到了 MD5 之前的字符串:

```
/api/recommended/getapp_id=1002client_info={
  "channel" : "0",
  "os" : "iOS",
  "bssid" : "90:72:40:f:e5:a9",
  "model" : "iPad4,4",
  "ssid" : "TieTie Tech 5GHz",
  "start_time" : "1457361686666",
  "version" : "7.1.2",
  "resume_time" : "1457361686666"
}curidentity=0expectId=1148692page=2pageSize=15req_time=1457362529031sor
```

断点变 Log

Xcode 中的 Action 断点，在命令行下也是可以玩的:

```
breakpoint command add 1

> po $arg1
```

```
> c
> DONE
```

这样，我们每次出发网络请求就不用手动打印并继续了。

微信扫一扫
关注该公众号