目录

移动端逆向：绕过抓包限制

- 最新版本：`0.8.0`
- 更新时间：`20250603`

简介

整理移动端逆向期间，关于如何绕过各种抓包的限制；包括绕过证书绑定和其他一些特殊处理；绕过证书绑定又包括iOS端和安卓端，比如iOS端的插件，比如SSL Kill Switch的ssl-kill-switch3和SSL Kill Switch 2，以及SSLBypass等插件等去实现绕过ssl证书绑定从而破解https；安卓端的Xposed模块，包括TrustMeAlready、JustTrustMe等等；其他特殊处理包括抖音的iOS版和Android版的一些特定处理等等；

源码+浏览+下载

本书的各种源码、在线浏览地址、多种格式文件下载如下：

HonKit源码

- crifan/mobile_re_capture_bypass_limit: 移动端逆向：绕过抓包限制

如何使用此HonKit源码去生成发布为电子书

详见：crifan/honkit_template: demo how to use crifan honkit template and demo

在线浏览

- 移动端逆向：绕过抓包限制 book.crifan.org
- 移动端逆向：绕过抓包限制 crifan.github.io

离线下载阅读

- 移动端逆向：绕过抓包限制 PDF
- 移动端逆向：绕过抓包限制 ePub
- 移动端逆向：绕过抓包限制 Mobi

版权和用途说明

鸣谢

感谢我的老婆**陈雪**的包容理解和悉心照料，才使得我 `crifan` 有更多精力去专注技术专研和整理归纳出这些电子书和技术教程，特此鸣谢。

其他

作者的其他电子书

本人 `crifan` 还写了其他 `150+` 本电子书教程，感兴趣可移步至：

crifan/crifan_ebook_readme: Crifan的电子书的使用说明

关于作者

关于作者更多介绍，详见：

关于CrifanLi李茂 – 在路上

绕过抓包限制概述

移动端（iOS、Android等）的app抓包期间，往往会遇到各种限制。

除了普通的，比如https的SSL证书安装等事情，可以通过安装抓包工具（Charles等）的SSL根证书之外（举例：移动端安装ssl证书 · app抓包利器：Charles），其他还些相对更加高级的，和抓包工具本身独立的相关内容，可以统称为：绕过抓包限制。

下面就来详细介绍一下，如何绕过这些抓包的限制。

绕过证书绑定

在移动端（iOS、Android等）的app的抓包期间，如果有些https请求，在之前抓包软件已设置了好各种证书和配置后，看到的：

- 要么是 `unknown`
    -
- 要么是：加密的乱码
- 要么是：报错无法抓包

而无法看到我们希望的明文数据，则：

最大可能是，对方用了https的： `SSL pinning`

什么是SSL pinning

- `SSL pinning` = `certificate pinning` = `证书绑定` = `SSL证书绑定`
    - 原理：
        - 内部加了ssl证书的校验，给合法的有效的证书，先计算出 `fingerprint` ，通过 `fingerprint` 指纹去匹配对比，才视为有效，否则拒绝访问
    - 效果：对方的app内部，只允许，承认其自己的，特定的证书
        - 导致此处Charles的证书不识别，不允许
        - 导致Charles无法解密看到https的明文数据

Android

信任证书问题

对于 `Android 7.0` = `API 24` 之后，做了些改动，使得系统安全性增加了，导致：

- `app` 默认不信任用户域的证书
  - -》之前把（ `Charles` 的）ssl证书，安装到 `受信任的凭据 -> 用户` 就没用了，因为不受信任了
  - 只信任（安装到）系统域的证书

导致无法抓包https，抓出来的https的请求，都是加了密的，无法看到原文了。

对此，总结出相关解决思路和方案：

- （努力想办法）让系统信任Charles的ssl证书
  - 作为app的开发者自己：改自己的app的配置，允许https抓包
    - 重要提醒：前提是得到或本身有app的源码
  - 把证书放到受系统信任的系统证书中去
    - 重要提示：前提是手机已root

下面详细介绍：

自己修改app去增加配置，允许https抓包

通过修改app的配置，使得允许https抓包

而修改app的配置，又分两种：

- 自己有app源码
  - 可以通过修改源码的方式去添加允许https抓包的配置
- 自己没源码，只有apk
  - 借助第三方工具修改apk，增加配置，允许https抓包

下面详细介绍如何操作：

通过修改app源码去增加配置允许https抓包

前提：

- 要么你自己是该app的开发者
  - 本身就有源码，就是app的拥有者
- 要么是你想要破解app的人
  - 本身就不可能有源码
    - 但是
      - 如果，有技术，有能力，有运气，破解得到app源码
        - 那理论上也可以使用此办法
      - 注意：实际情况下，**往往没机会破解出app源码**
        - 所以此办法不适用

如果具备修改app的源码，则具体操作过程是：

修改 `AndroidManifest.xml` ，增加如下配置：

```
<?xml version="1.0" encoding="utf-8"?>
  <manifest ... >
    <application          networkSecurityConfig="@xml/network_security_config"
                  ... >
    ...
    </application>
  </manifest>
```

在 `res` 目录下新建一个 `xml` 文件夹，再新建文件：

`res/xml/network_security_config.xml`

内容为：

手机中已正确安装Charles证书

```
<?xml version="1.0" encoding="utf-8"?>
  <network-security-config>
    <domain-config>
      <domain includeSubdomains="true">你要抓取的域名</domain>
      <trust-anchors>
        <certificates src="user"/>
      </trust-anchors>
  </domain-config>
</network-security-config>
```

其中：

- `<certificates src="user"/>`：信任用户自己安装的证书

手机中没安装Charles证书，但是已有Charles证书文件

```
<?xml version="1.0" encoding="utf-8"?>
    <network-security-config>
        <domain-config>
        <domain includeSubdomains="true">你要抓取的域名</domain>
        <trust-anchors>
        <certificates src="@raw/证书文件名"/>
        </trust-anchors>
    </domain-config>
</network-security-config>
```

然后再去：

- `res` 目录下新建一个 `raw` 文件夹
  - 将手机上安装的证书文件放入 `res/raw/` 目录下
    - 支持的证书格式：`pem`，`ca`

用工具修改apk增加配置允许https抓包

比如借助第三方工具：

levyitay/AddSecurityExceptionAndroid

去下载源码，再去：

```
cd AddSecurityExceptionAndroid
./addSecurityExceptions.sh ../xxx.apk
```

即可给apk增加允许https抓包的配置了，然后就可以继续用Charles抓包https了。

把证书放到系统信任区

- 背景和思路：既然只信任系统区的证书，那么可以想办法把Charles证书放到系统区，就可以被信任了，就可以https抓包了
  - 而系统信任的地方
    - 对应安卓的设置中的：`受信任的凭据 -> 系统`
    - 对应安卓系统目录：`/system/etc/security/cacerts/`
    - 对应的系统证书的名字有特定规则
      - 需要找到工具根据规则计算出名字后
      - 才能再去把证书放到系统区中
- 前提：手机已root
- 详细步骤
  - 计算证书名

    ```
    openssl x509 -subject_hash_old -in charles-ssl-proxying-certificate_saved.pem
    ```

    - 算出数值，比如 `3a1074b3`
  - 证书文件改名
    - 然后把原Charles证书 `charles-ssl-proxying-certificate_saved.pem` 改名为 `3a1074b3.0`
  - 放到系统分区
    - 放到 `/system/etc/security/cacerts/`
- 注意
  - 但是呢，现在多数手机都很难root了
    - 包括我之前的锤子M1L和很多常见品牌，比如小米、华为等，的最新手机
  - 如果真的可以root，那倒是容易此办法去解决ssl pinning的问题

Xposed插件

用其他工具绕开https校验实现https抓包：

- 确保了手机已root
  - Android：已root
    - 确保后续可以安装 `Xposed` 等工具
- 再去用可以绕开/禁止 `SSL pinning` 的插件
  - `Android`
    - 基于 `Xposed` 的JustTrustMe
      - 限制：
        - 只能支持 `Android 7.0` 之前的安卓
          - 超过 `Android 7.0` 就不工作了
    - 基于 `Cydia` 的Android-SSL-TrustKiller

即可绕开ssl的验证，抓包到https被解密变成明文的数据。

接下来详细，如何在已 `root` 的 `安卓` 中，基于 `Xposed` 框架和相关插件，去实现绕开https证书校验，实现抓包https得到明文数据。

准备好已root的安卓

对于想要获取已root的安卓，有两种方式：

- 对于安卓 `真机` 来说
  - 很久之前是很简单的事情
    - 随便买个安卓手机，都容易去 `root`
  - 现在
    - 大多数手机品牌（小米，华为等）新买到的都是安卓新版本
      - 比如： `Android 8.0` ， `Android 9.0` 之类的
    - 且都很难root
      - 都要向官方申请，要等很长时间（以月为单位）
        - 申请先解锁 `BL = BootLoader`
        - 然后才能root
      - 而且最后还未必通过
    - 结论就是：
      - 现在很难买到能root的新安卓手机了
  - 所以
    - 最终方案是：
      - 去淘宝买个二手的已root的安卓手机
        - 比如：400元左右的
          - 二手小米4
            - 型号： `MI 4 LTE-CU`
            - 安卓系统版本： `Android 4.4.4`
- 对于安卓 `模拟器` 来说

    - Mac中也有很多安卓模拟器
        - 目前测试能用的有
            - `夜神Nox` 安卓模拟器
                - 模拟的是：`Android 4.4.2`
            - `网易MuMu` 安卓模拟器
                - 模拟的是：`Android 6.0.1`
        - 其他不能用，不好用的有
            - Andy：安装后无法正常运行
            - 天天模拟器：没有我要的Mac版
            - Genymotion：收费的，还要麻烦的去破解，暂时懒得继续试
            - BlueStacks：只支持Win，不支持Mac，且也比较老旧

如前所述，已root的安卓，可以选用：

- 安卓真机
    - 二手小米4：`Android 4.4.4`
- 安卓模拟器
    - 夜神模拟器：`Android 4.4.2`
    - 网易MuMu：`Android 6.0.1`

中的任何一个。

此处以Mac中的 `夜神模拟器` 为例去解释。

- Mac版 夜神模拟器 Nox App Player

- 已经root了

- 模拟的是：Android 4.4.2

。

> **夜神模拟器**
>
> 关于夜神的更多解释详见：
>
> 好用的安卓模拟器：夜神Nox

安装Xposed框架

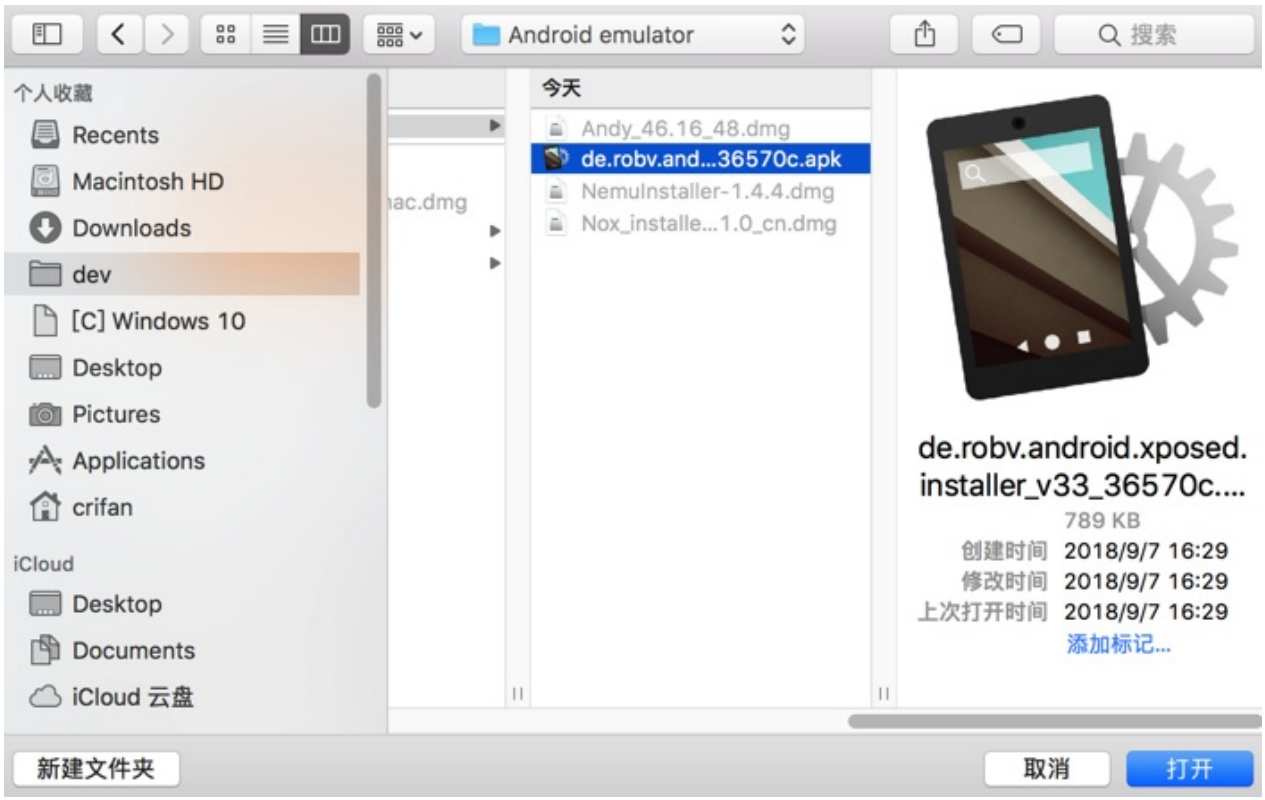
在已root的安卓中，安卓Xposed框架。

根据Xposed官网解释，Xposed的版本和安卓版本需要对应，否则无法正常安装和使用：

- `Android 4.0.3` ~ `Android 4.4`
  - 用 `v2.7`，`v2.6.1` 的 `Xposed installer`
    - 支持：此处基于 `Android 4.4.2` 的夜神安卓模拟器
- `Android 5` 以上
  - 用 `3.x` 版本的 `Xposed installer`
    - 比如：3.5.1

下面介绍在夜神模拟器中安装Xposed的详细步骤：

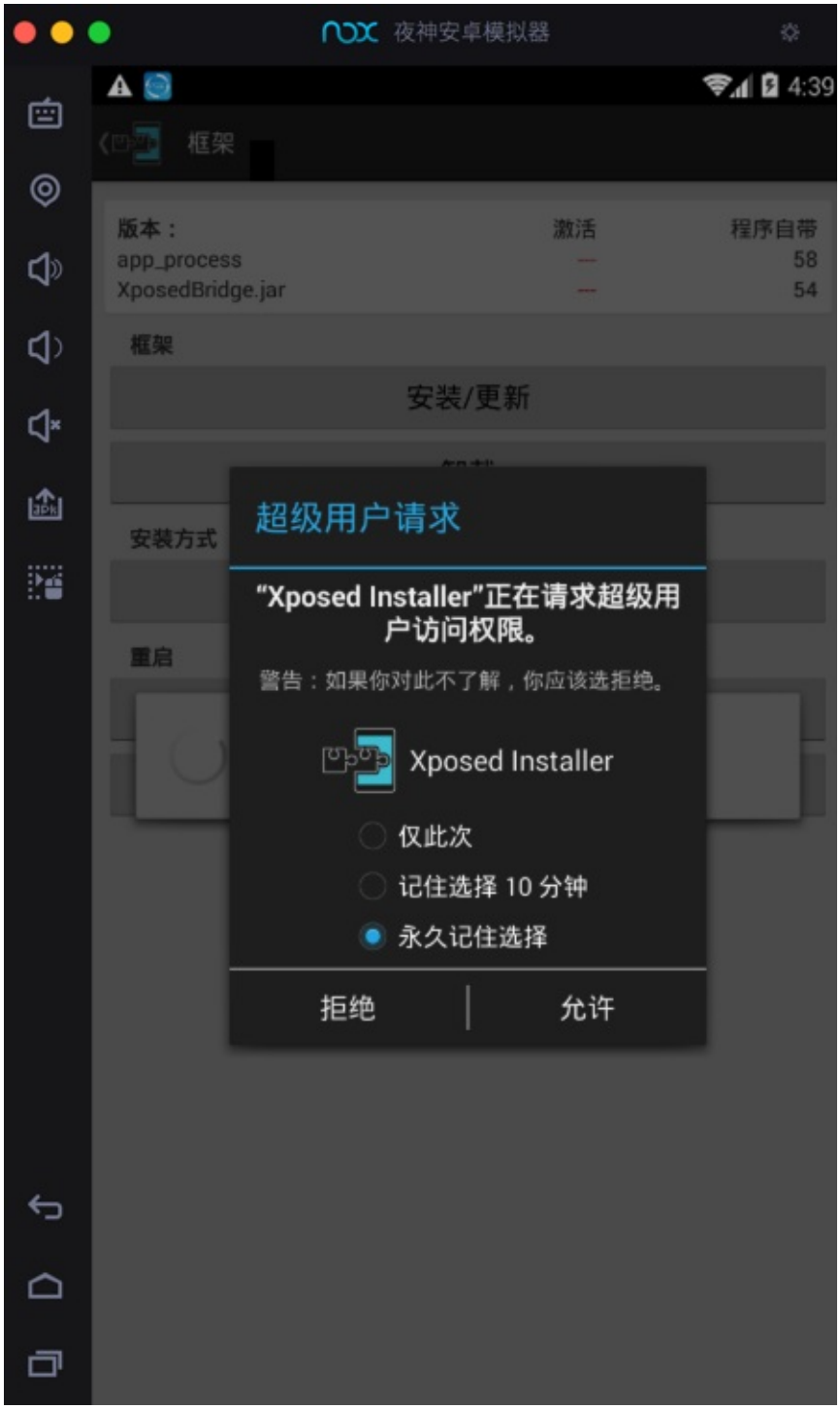
下载 `v2.7` 的xposed installer的de.robv.android.xposed.installer_v33_36570c.apk：



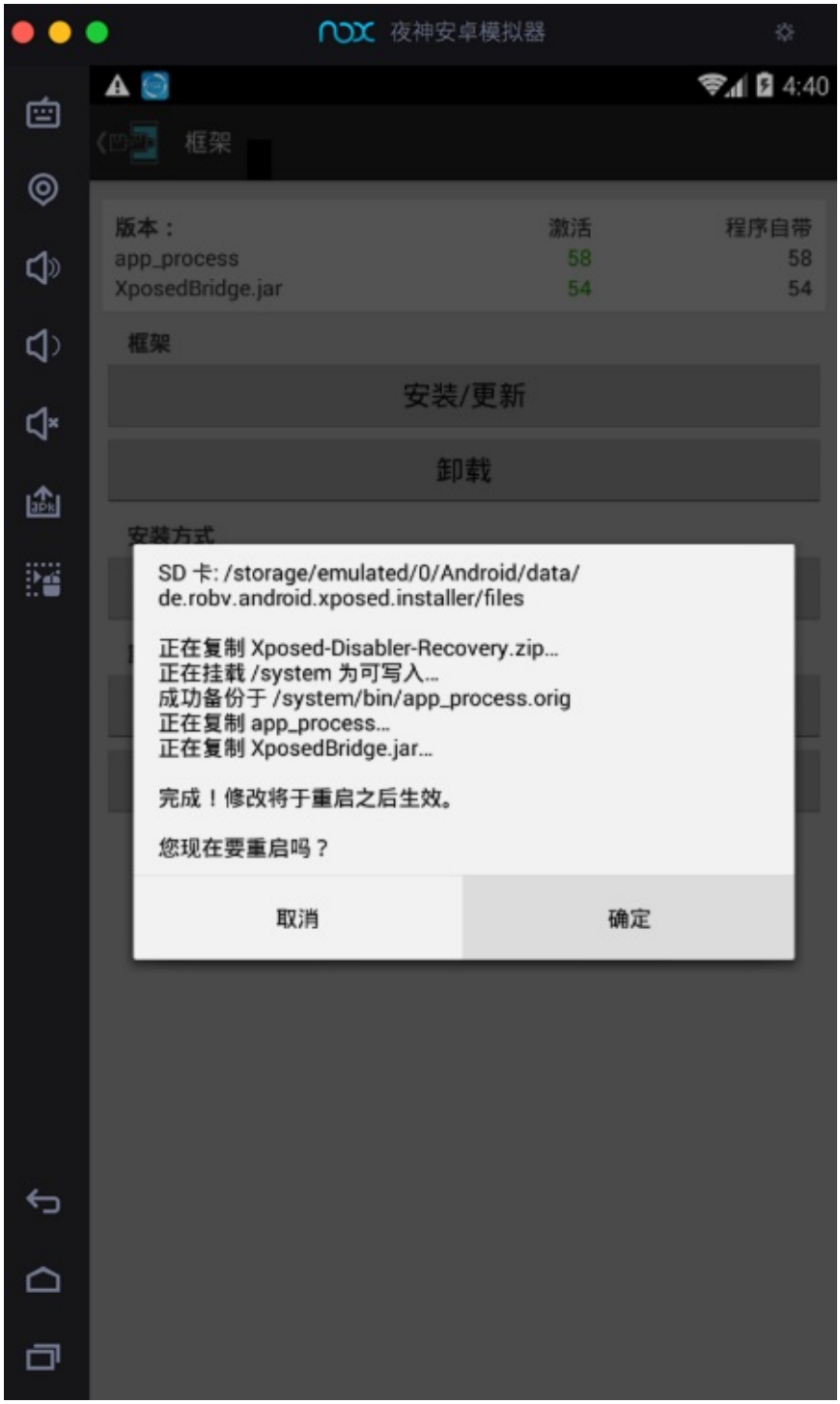
然后拖动到夜神模拟器中，即可自动安装，安装完毕后，可以在桌面上看到：

接着点击进入 `Xposed Installer` ，再去安装 `Xposed框架` 到安卓系统中：

点击 `安装/更新` ，在弹框中对于 超级用户请求，设置 `永久记住选择` ：



然后会去安装到系统中，再点击 `确定` 去重启：



重启后，看到Xposed框架中显示：

```
app_process 激活 58 程序自带 58
XposedBridge.jar  激活 54 程序自带 54
```

表示Xposed框架已激活，可以继续使用了。

> Xposed框架
>
> 关于Xposed框架的更多解释详见：
>
> 安卓逆向调试：XPosed框架

用JustTrustMe破解ssl pinning

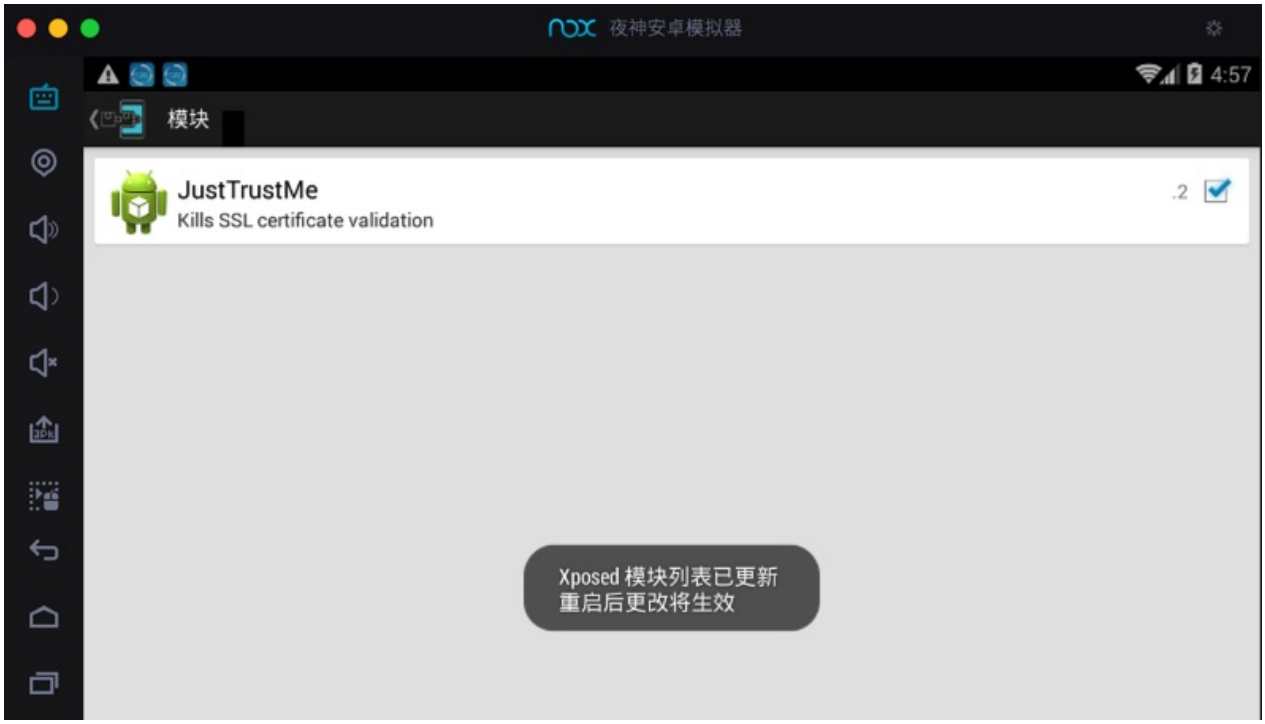
再去下载和安装 `JustTrustMe` ：

去JustTrustMe的GitHub的release下载JustTrustMe.apk

像安装普通安卓apk一样，拖动进去，即可把 `JustTrustMe.apk` 安装到夜神模拟器中。

注意：此 `JustTrustMe` 没有app界面，所以安卓后，也没有 `打开` 之类的操作。而只能是，去Xposed中才能看到和激活此应用。

然后去Xposed中找到并勾选以激活 `JustTrustMe` ：

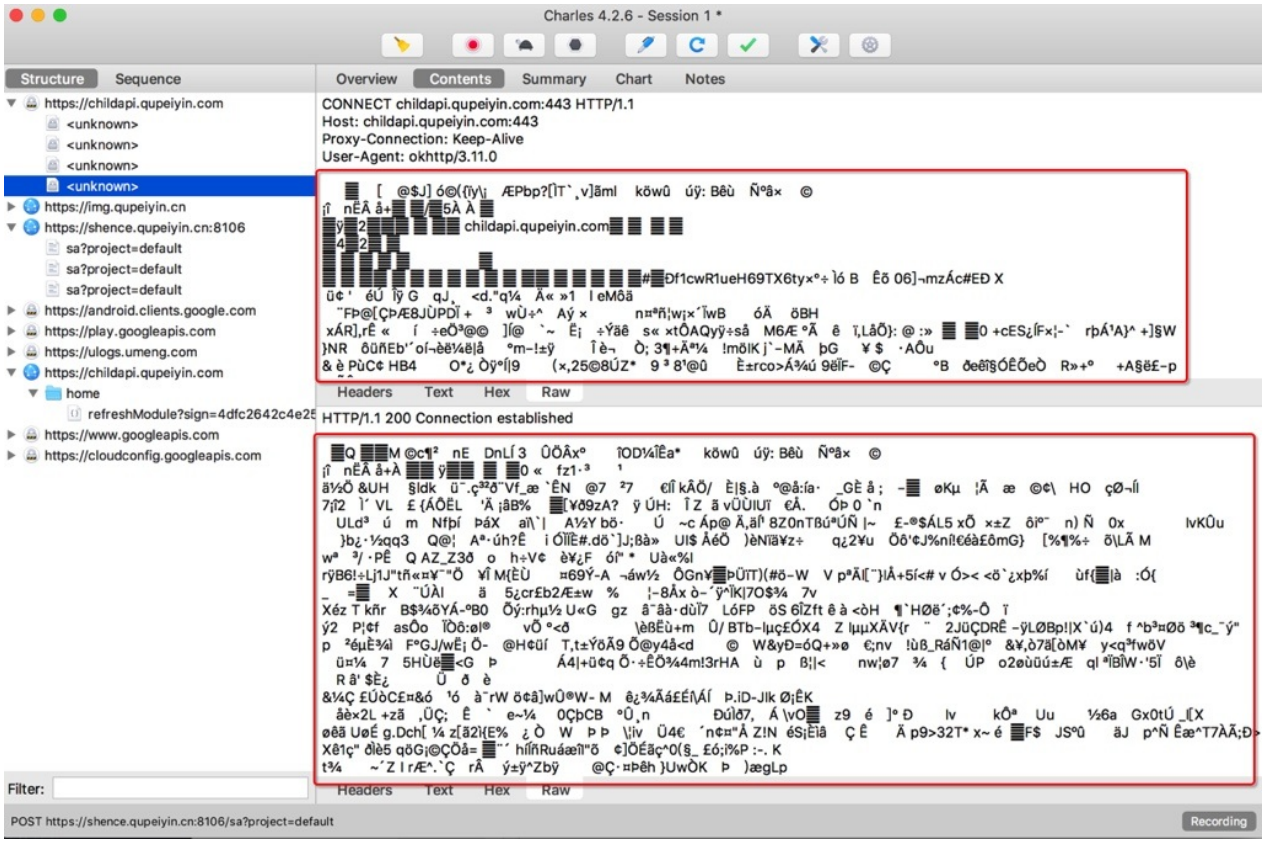


然后记得重启Xposed：

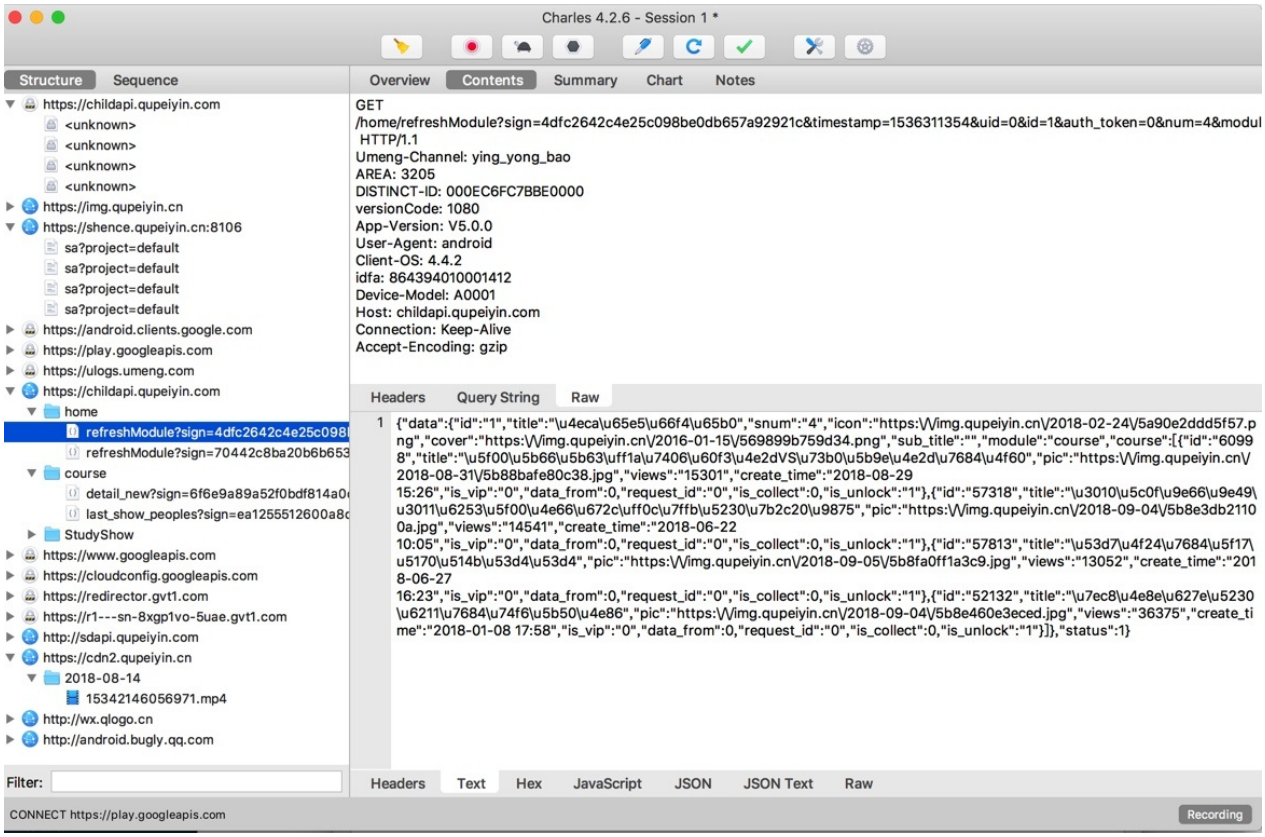
Charles抓包可以看到https明文

然后再去用Charles抓包：

对于之前没有启动https的，抓包https看到的都是 `unknown` 的请求和数据是加密的乱码：



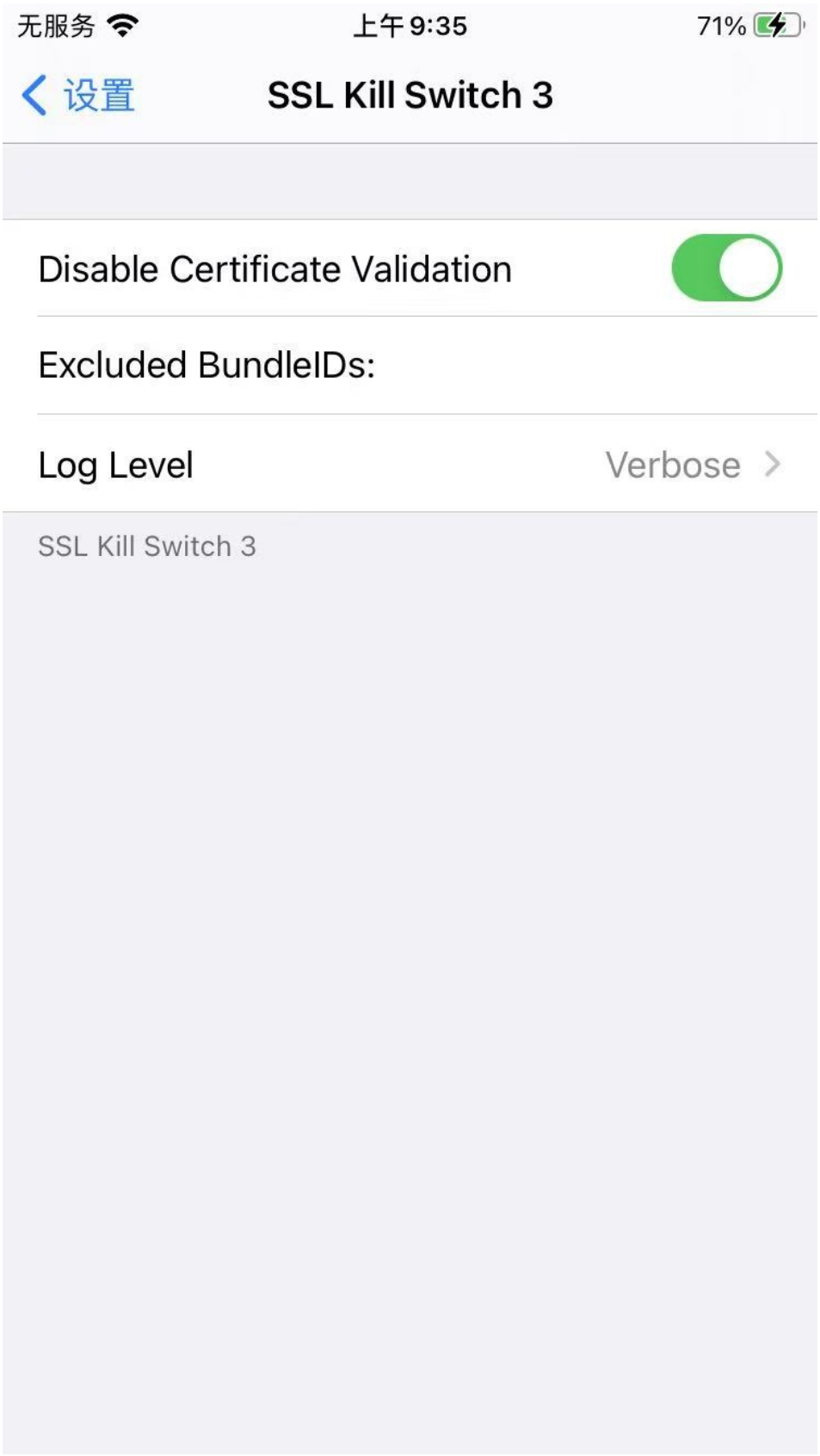
现在，即可绕开app的https的证书校验，从而可以看到明文数据了：



TrustMeAlready

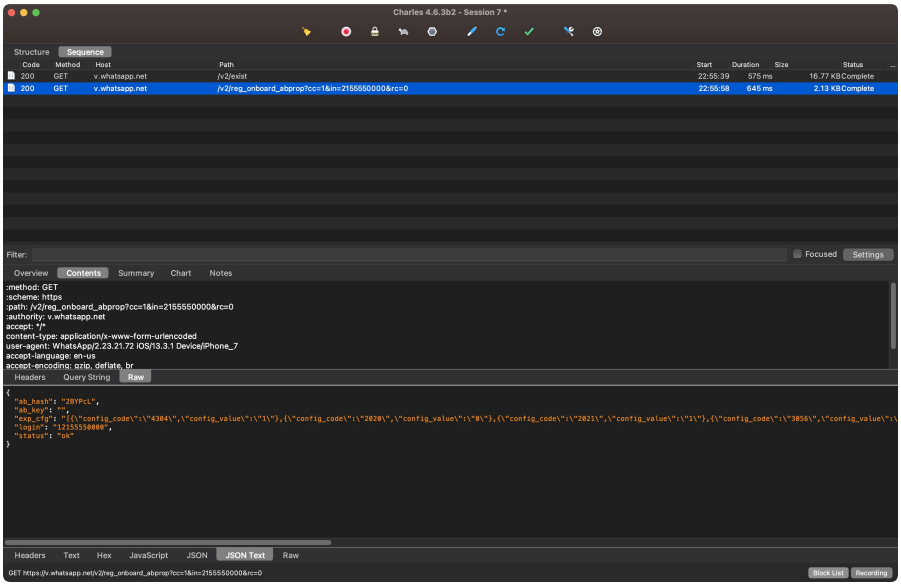
iOS端

- 前提
  - iOS：已越狱
    - 确保后续能安装 `Cydia` / `Sileo` 等包管理器
- 绕过SSL pinning的工具
  - 插件=tweak=越狱插件
    - 最好用的是：`NyaMisty/ssl-kill-switch3`
      - Repo源地址
        - https://repo.misty.moe/apt/
      - 下载地址
        - https://github.com/NyaMisty/ssl-kill-switch3
          - https://github.com/NyaMisty/ssl-kill-switch3/releases
            - v1.5.1
              - **rootfull**
                - moe.misty.sslkillswitch3_1.5.1+rootful_iphoneos-arm.deb
              - roothide
                - moe.misty.sslkillswitch3_1.5.1+roothide_iphoneos-arm64e.deb
              - rootless
                - moe.misty.sslkillswitch3_1.5.1+rootless_iphoneos-arm64.deb
      - 注意：安装后要开启才能使用
        - `设置` -> `SSL Kill Switch 3` ->勾选=开启：`Disable Certificate Validation`

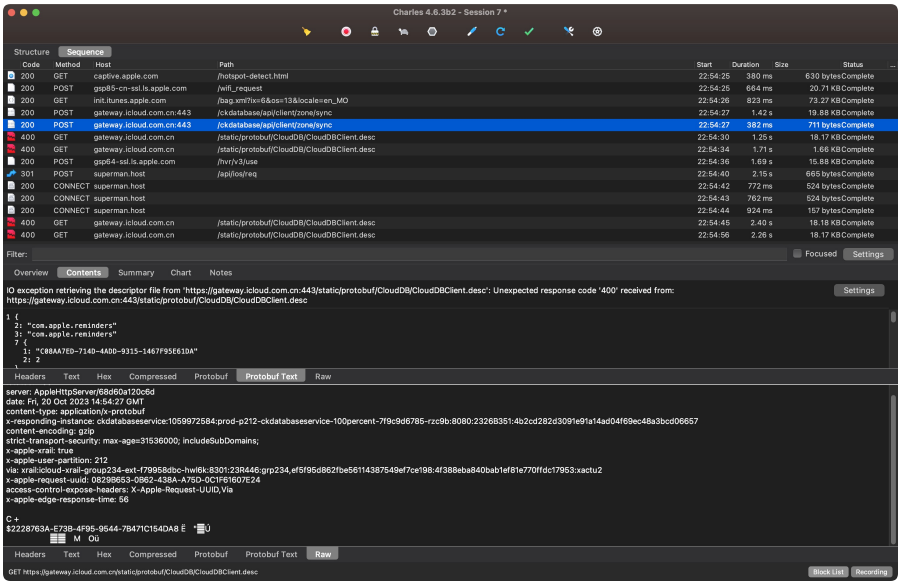


- 插件详情
  - rootfull

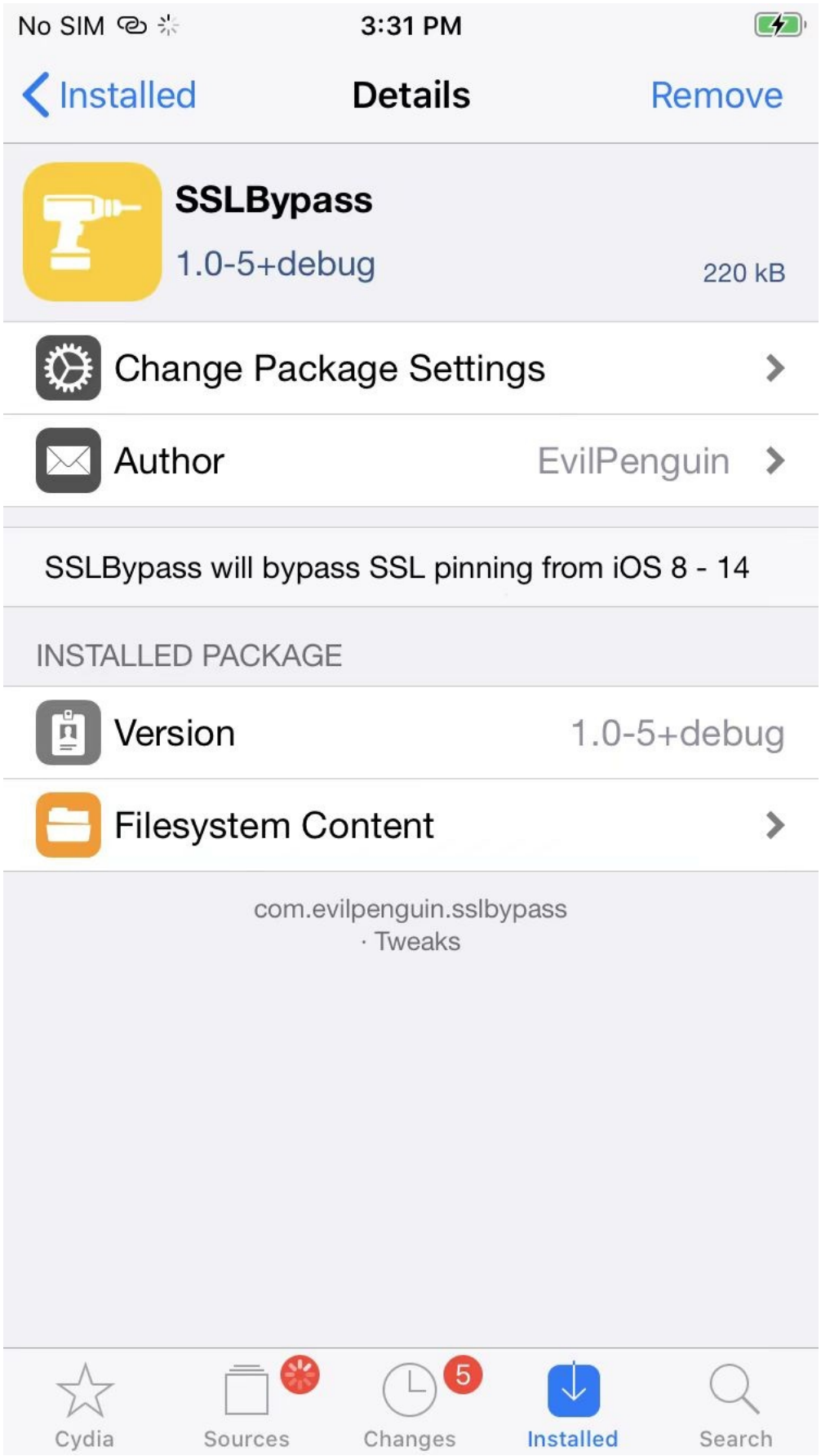
- 效果
  - WhatsApp的https请求可以看到明文
    - https://v.whatsapp.net



- Apple的相关请求也可以看到明文
  - https://gateway.icloud.com.cn

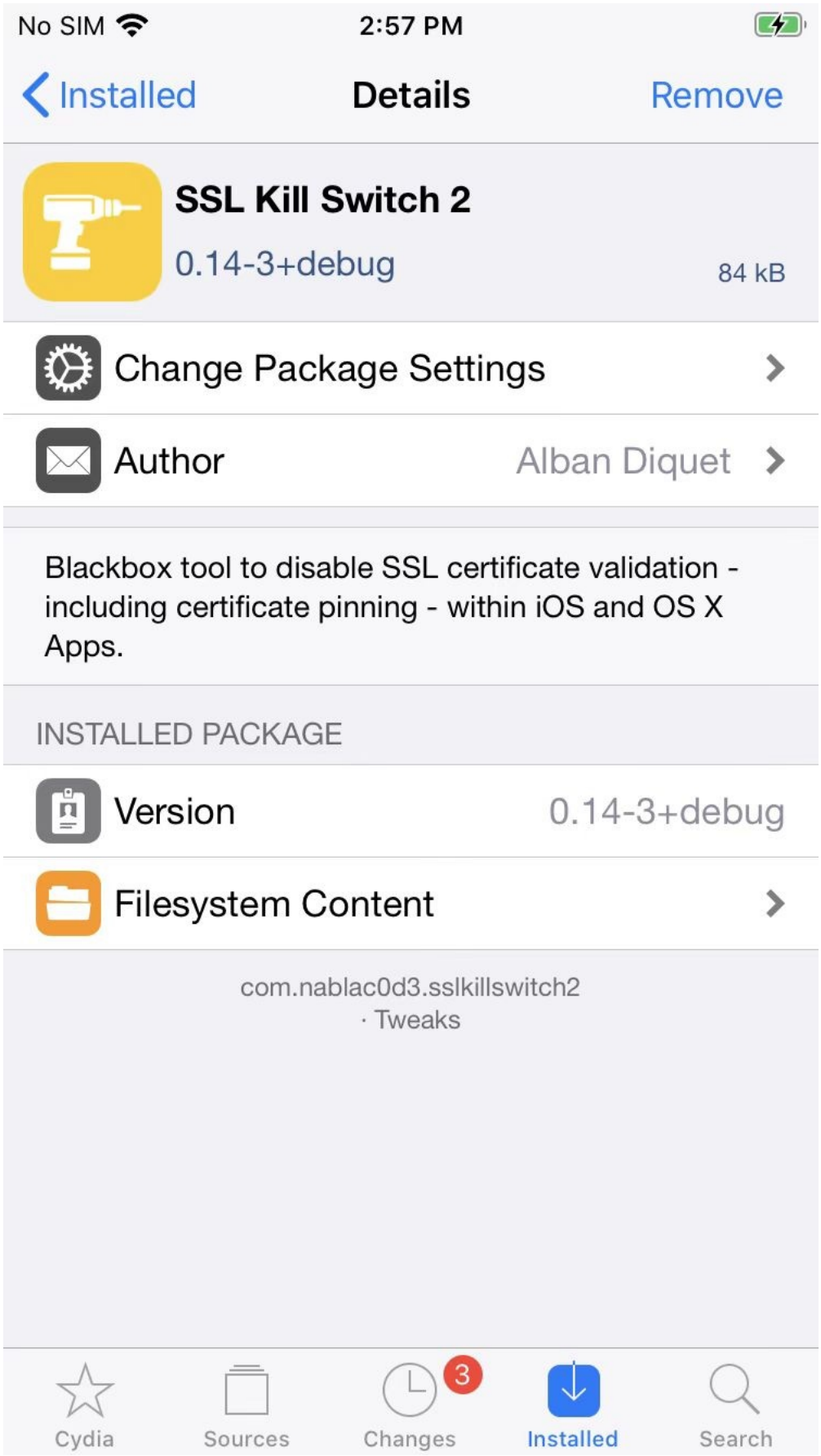


- 其次是：`evilpenguin/SSLBypass`
  - 下载地址
    - https://github.com/evilpenguin/SSLBypass
      - https://github.com/evilpenguin/SSLBypass/blob/main/packages/com.evilpenguin.sslbypass_1.0-5%2Bdebug_iphoneos-arm.deb
  - 插件详情



- 再次是：`nabla-c0d3/ssl-kill-switch2`
  - 下载地址
    - https://github.com/nabla-c0d3/ssl-kill-switch2

        - https://github.com/nabla-c0d3/ssl-kill-switch2/releases
            - https://github.com/nabla-c0d3/ssl-kill-switch2/releases/download/0.14/com.nablac0d3.sslkillswitch2_0.14.deb
    - 插件详情



- 注：
  - 旧版本：iOS SSL Kill Switch
- 其他

        - SSL Kill Switch 2 (iOS 13)
    - `Frida` 的js
        - 绕过证书校验
            - Frida CodeShare Project: iOS SSL Bypass

插件

SSL Kill Switch

记得是有多个版本：

- `SSL Kill Switch 2`
- `ssl-kill-switch3`

抓包举例

SSL Kill Switch 2 (iOS 13)

Apple账号

经测试，Apple账号登录过程中的https请求：

- 除了特殊的，特定的：
  - https://gsa.apple.com
- 之外，其他普通的（包括带账号绑定的https请求），是可以抓包的，能看到明文的
  - 比如：
    - https://setup.icloud.com
    - https://bag.itunes.apple.com

具体步骤：

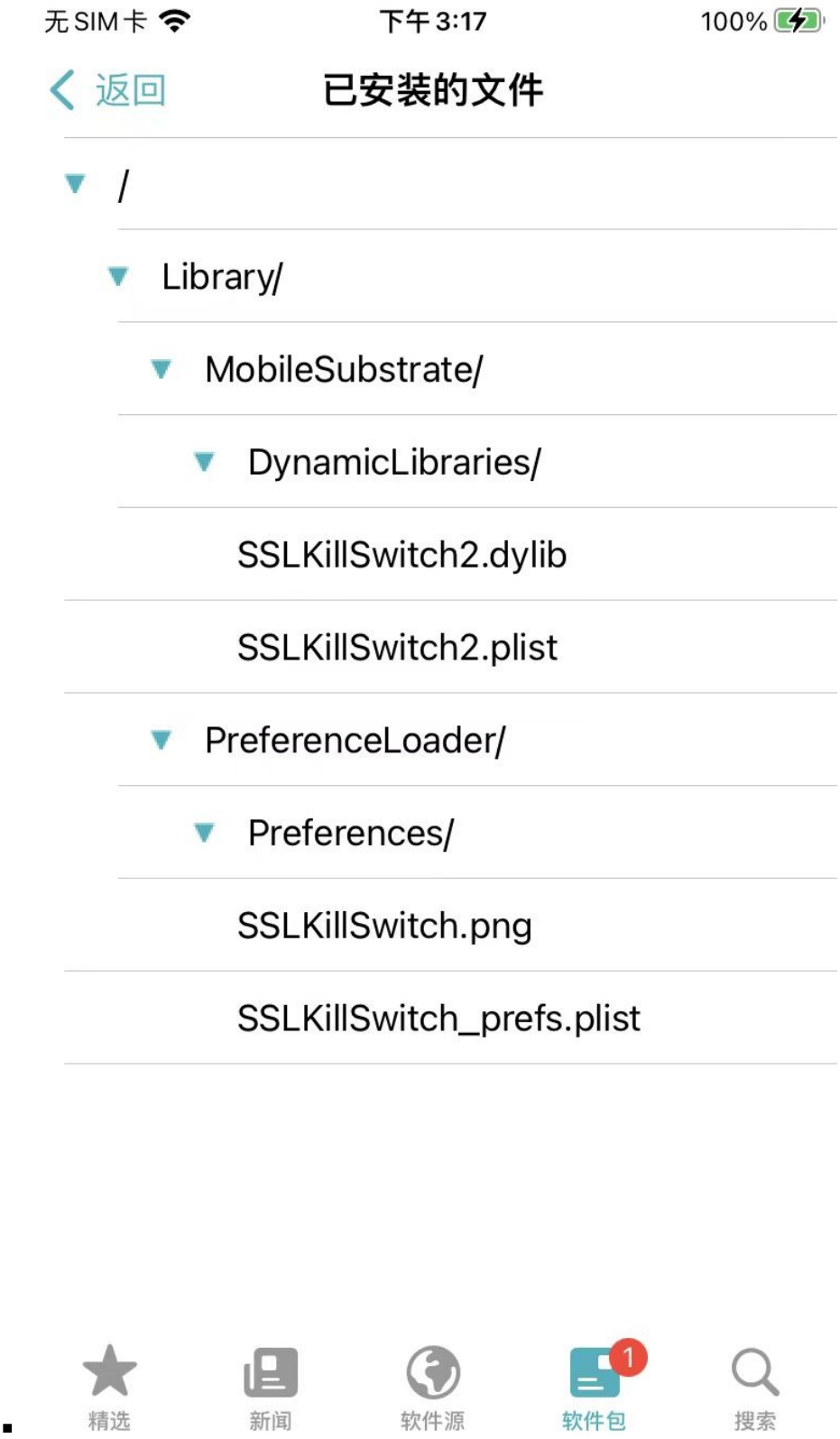
- 手机
  - iPhone8: `iOS 15.1` 、 `palera1n` 的 `rootful` 越狱
- Sileo中安装插件
  - `julioverne` 的 `SSL Kill Switch 2 (iOS 13)`
    - 源地址：https://julioverne.github.io


Sileo - 粘贴自： 微信
编辑
软件源
所有软件包
查看已经添加的软件源的所有软件包
84
添加 1 个软件源
添加以下从剪贴板中找到的 URL：
https://julioverne.github.io/
添加 1 个软件源
手动输入
取消
palera1n strap
https://strap.palera.in/
66
5 软件源
精选
新闻
软件源
软件包
搜索

- 插件安装后效果

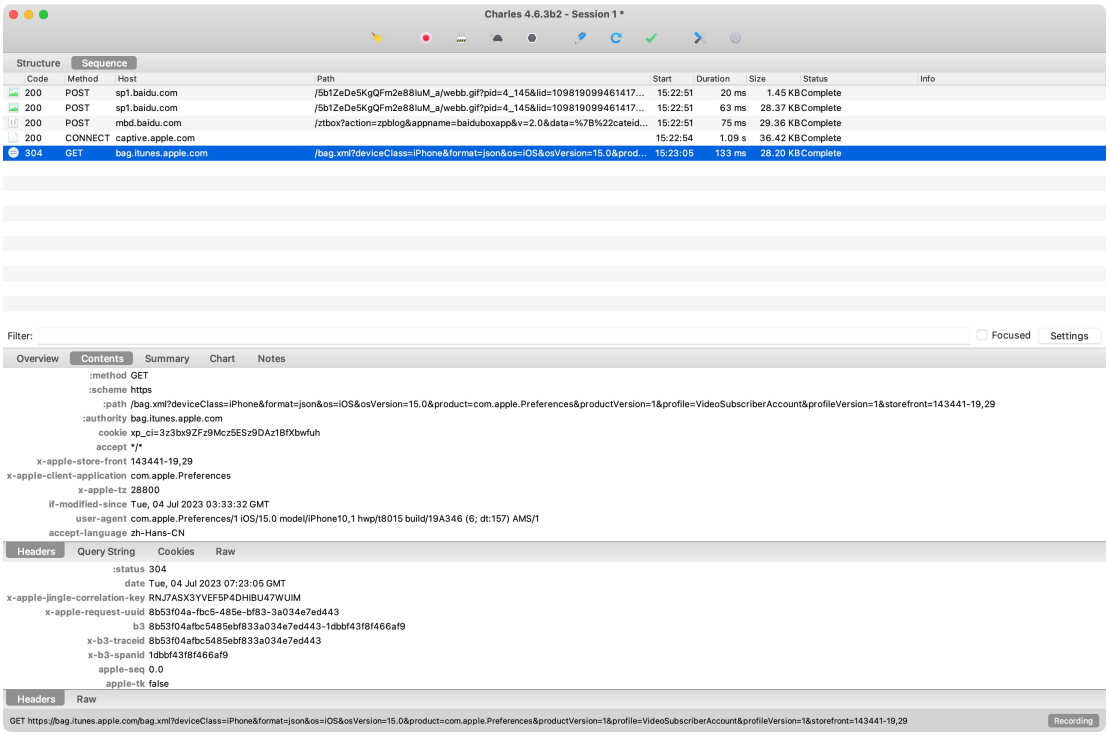
- `com.julioverne.sslkillswitch2` (0.14c)

- 已安装文件

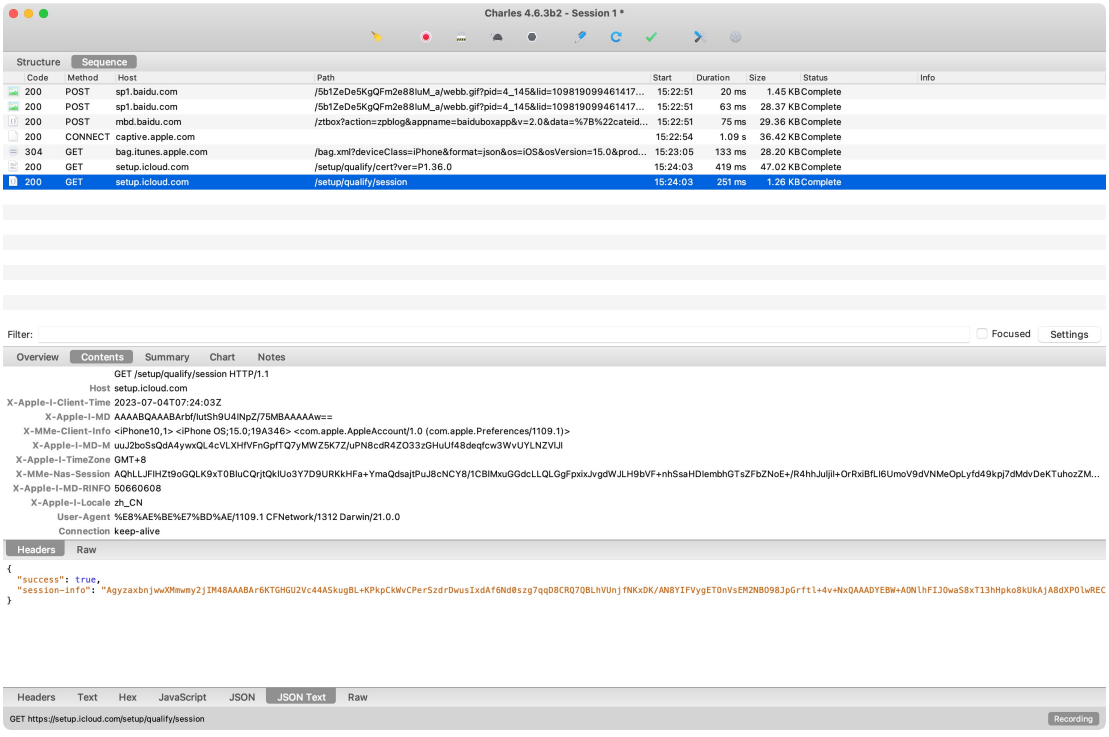


https抓包效果：

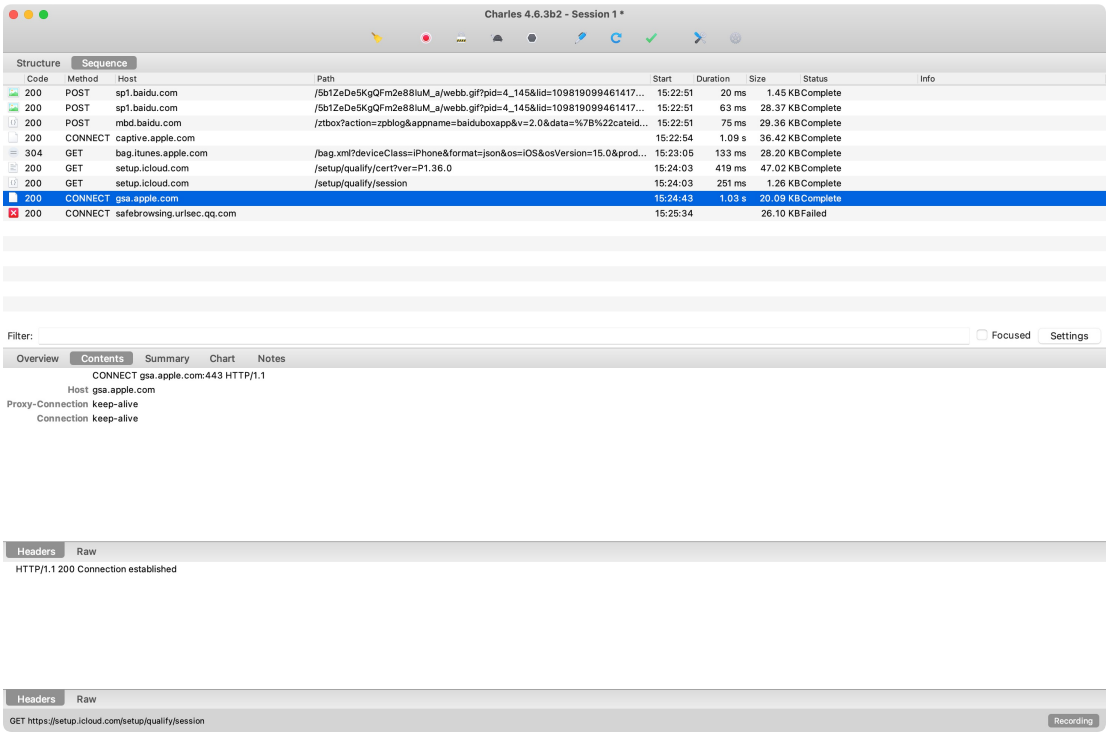
- 能抓包明文的
  - https://bag.itunes.apple.com/bag.xml



- https://setup.icloud.com/setup/qulify/session



- 无法抓包的
  - https://gsa.apple.com




```
https://gsa.apple.com
200
CONNECT
gsa.apple.com

Tue Jul 04 15:24:43 CST 2023
1031
20568
Complete
```

SSLBypass

其他特殊处理

对于其他更加复杂的特殊情况，一般是：

- 特定的app，内部加了额外的反扒方面的技术手段
  - 解决方案：往往要涉及到，app的破解和逆向，找到关键点函数，才能针对性的去绕过处理
    - 注：关于app的逆向破解，详见
      - Android逆向开发
      - iOS逆向开发

抖音

Android版

Android版=安卓版抖音，对于网络抓包做了限制，导致的现象是：

- 当已设置好WiFi的代理，再去抓包抖音，会出现：`网络错误。当前无网络，请检查后重试`
  。


16:08
直播
经验
同城
关注
商城
推荐
网络错误
当前无网络，请检查后重试
刷新
首页
朋友
消息
我

- mitmdump等抓包工具抓包时报错：`Client TLS handshake failed`

```
[16:07:30.334][192.168.1.20:40974] server connect polaris.zijieapi.com:443 (223
.111.245.248:443)
[16:07:30.381][192.168.1.20:40974] Client TLS handshake failed. The client does
 not trust the proxy's certificate for polaris.zijieapi.com (OpenSSL Error([('S
SL routines', '', 'ssl/tls alert certificate unknown')]))
[16:07:30.381][192.168.1.20:40974] client disconnect
[16:07:30.381][192.168.1.20:40974] server disconnect polaris.zijieapi.com:443 (
223.111.245.248:443)
[16:07:33.674][192.168.1.20:40984] client connect
[16:07:33.693][192.168.1.20:40984] server connect api3-normal-c.amemv.com:443 ([
2409:8c20:aa51:2e:3::3de]:443)
[16:07:33.734][192.168.1.20:40984] Client TLS handshake failed. The client does
 not trust the proxy's certificate for api3-normal-c.amemv.com (OpenSSL Error([
('SSL routines', '', 'ssl/tls alert certificate unknown')]))
...
[16:07:42.408][192.168.1.20:37932] server disconnect i.snssdk.com:443 ([2409:8c
20:5223:104:3::3fd]:443)
[16:07:44.161][192.168.1.20:37946] client connect
[16:07:44.195][192.168.1.20:37946] server connect log.snssdk.com:443 ([2409:8c2
0:9c73:103:3::9]:443)
[16:07:44.321][192.168.1.20:37946] Client TLS handshake failed. The client does
 not trust the proxy's certificate for log.snssdk.com (OpenSSL Error([('SSL rou
tines', '', 'ssl/tls alert certificate unknown')]))
[16:07:44.324][192.168.1.20:37946] client disconnect
[16:07:44.326][192.168.1.20:37946] server disconnect log.snssdk.com:443 ([2409:
8c20:9c73:103:3::9]:443)
[16:07:44.378][192.168.1.20:37962] client connect
[16:07:44.403][192.168.1.20:37962] server connect log.snssdk.com:443 ([2409:8c2
0:9c73:103:3::8]:443)
[16:07:44.451][192.168.1.20:37962] Client TLS handshake failed. The client does
 not trust the proxy's certificate for log.snssdk.com (OpenSSL Error([('SSL rou
tines', '', 'ssl/tls alert certificate unknown')]))
```

```
[16:07:44.453][192.168.1.20:37962] client disconnect
[16:07:44.454][192.168.1.20:37962] server disconnect log.snssdk.com:443 ([2409:
8c20:9c73:103:3::8]:443)
[16:07:47.372][192.168.1.20:41272] client connect
[16:07:47.382][192.168.1.20:41272] server connect i.snssdk.com:443 ([2409:8c20:
5223:104:3::3fe]:443)
[16:07:47.412][192.168.1.20:41272] Client TLS handshake failed. The client does
 not trust the proxy's certificate for i.snssdk.com (OpenSSL Error([('SSL routi
nes', '', 'ssl/tls alert certificate unknown')]))
[16:07:47.413][192.168.1.20:41272] client disconnect
[16:07:47.413][192.168.1.20:41272] server disconnect i.snssdk.com:443 ([2409:8c
20:5223:104:3::3fe]:443)
```

主要是用的另外一个手段：

改用另外一个网络库（Google开发的）： `Cronet`

而其默认不允许https抓包，而想要绕过限制，可以去hook特定的一些函数，从而实现，绕过抓包限制，可以正常（https）抓包。

而实现绕过安卓版抖音的抓包限制的：

- 核心思路，就一种
  - hook函数 `SSL_CTX_set_custom_verify` ，使其参数 `mode` 从 `1` 改为 `0`
    - 目的：实现忽略SSL协议验证，从而允许https抓包

不过具体实现做法=方式，有2种：

- hook代码的方式
  - 用Frida找到并去hook函数 `SSL_CTX_set_custom_verify` ，使其参数 `mode` 从 `1` 改为 `0`
- 修改并替换so库的方式
  - 已知上述几个特定函数是属于 `libsscronet.so` ，所以可以去修改此so库文件，并替换掉该库文件

下面详细解释：

Frida去hook代码的方式

Frida的hook代码=函数：

- `hookNative_SSL_CTX_set_custom_verify`

```
function hookNative_SSL_CTX_set_custom_verify(){
  var origFuncPtr_verify = Module.findExportByName("libsscronet.so", "SSL_CTX_set_custo
m_verify");
  console.log("origFuncPtr_verify=" + origFuncPtr_verify)
  // https://commondatastorage.googleapis.com/chromium-boringssl-docs/ssl.h.html#SSL_VE
RIFY_NONE
  // OPENSSL_EXPORT void SSL_CTX_set_custom_verify(SSL_CTX *ctx, int mode, enum ssl_ver
ify_result_t (*callback)(SSL *ssl, uint8_t *out_alert));
  var func_verify = new NativeFunction(origFuncPtr_verify, 'pointer', ['pointer', 'int',
 'pointer']);
  console.log("func_verify=" + func_verify)
  var newFuncPtr_verify = new NativeCallback(
    function (ctx, mode, callback) {
      console.log("SSL_CTX_set_custom_verify calling: ctx=" + ctx + ", mode=" + mode +
", callback=" + callback)
      // #define SSL_VERIFY_NONE 0x00
      // #define SSL_VERIFY_PEER 0x01
      // #define SSL_VERIFY_FAIL_IF_NO_PEER_CERT 0x02
      // #define SSL_VERIFY_PEER_IF_NO_OBC 0x04
      const SSL_VERIFY_NONE = 0
      return func_verify(ctx, SSL_VERIFY_NONE, callback)
      // return func_verify(ctx, mode, callback)
    },
    'pointer',
    ['pointer', 'int', 'pointer']
  )
  console.log("newFuncPtr_verify=" + newFuncPtr_verify)
  Interceptor.replace(origFuncPtr_verify, newFuncPtr_verify)
}


function afterLibLoaded_libsscronet(libraryName) {
  console.log("libraryName=" + libraryName)
  hookNative_SSL_CTX_set_custom_verify()
}

function hookNative_libsscronet(){
  // let libsscronet_so_name = "libsscronet.so"
  // console.log("libsscronet_so_name=" + libsscronet_so_name)
  // FridaAndroidUtil.waitForLibLoading(libsscronet_so_name, afterLibLoaded_libsscronet)

  FridaAndroidUtil.hookAfterLibLoaded("libsscronet.so", afterLibLoaded_libsscronet)
}

function hookDouyin_Native(){
  hookNative_libsscronet()
}
```

```
function hookAndroid() {
  if( Java.available){
    console.error("Java is not available")
    return
  }

  console.log("Java is available")
  console.log("Java.androidVersion=" + Java.androidVersion)

  Java.perform(function () {
    hookDouyin_Native()

    console.log("-------------------- Begin Hook --------------------")
  })

}

setImmediate(hookAndroid)
```

相关代码：

```
class FridaUtil {

  // Frida pointer to C string
  static ptrToCStr(curPtr){
    var curCStr = curPtr.readCString()
    // console.log("curCStr=" + curCStr)
    return curCStr
  }

}

class FridaAndroidUtil {

...

  static waitForLibLoading(libraryName, callback_afterLibLoaded){
    console.log("libraryName=" + libraryName + ", callback_afterLibLoaded=" + callback_
afterLibLoaded)
    var android_dlopen_ext = Module.getExportByName(null, 'android_dlopen_ext')
    console.log("android_dlopen_ext=" + android_dlopen_ext)
    if (null == android_dlopen_ext) {
      return
    }

    Interceptor.attach(android_dlopen_ext, {
      onEnter: function (args) {
        // android_dlopen_ext(const char *_Nullable __filename, int __flags, const andr
oid_dlextinfo *_Nullable __info)

        // console.log("args=" + args)
        var filenamePtr = args[0]
        var libFullPath = FridaUtil.ptrToCStr(filenamePtr)
```

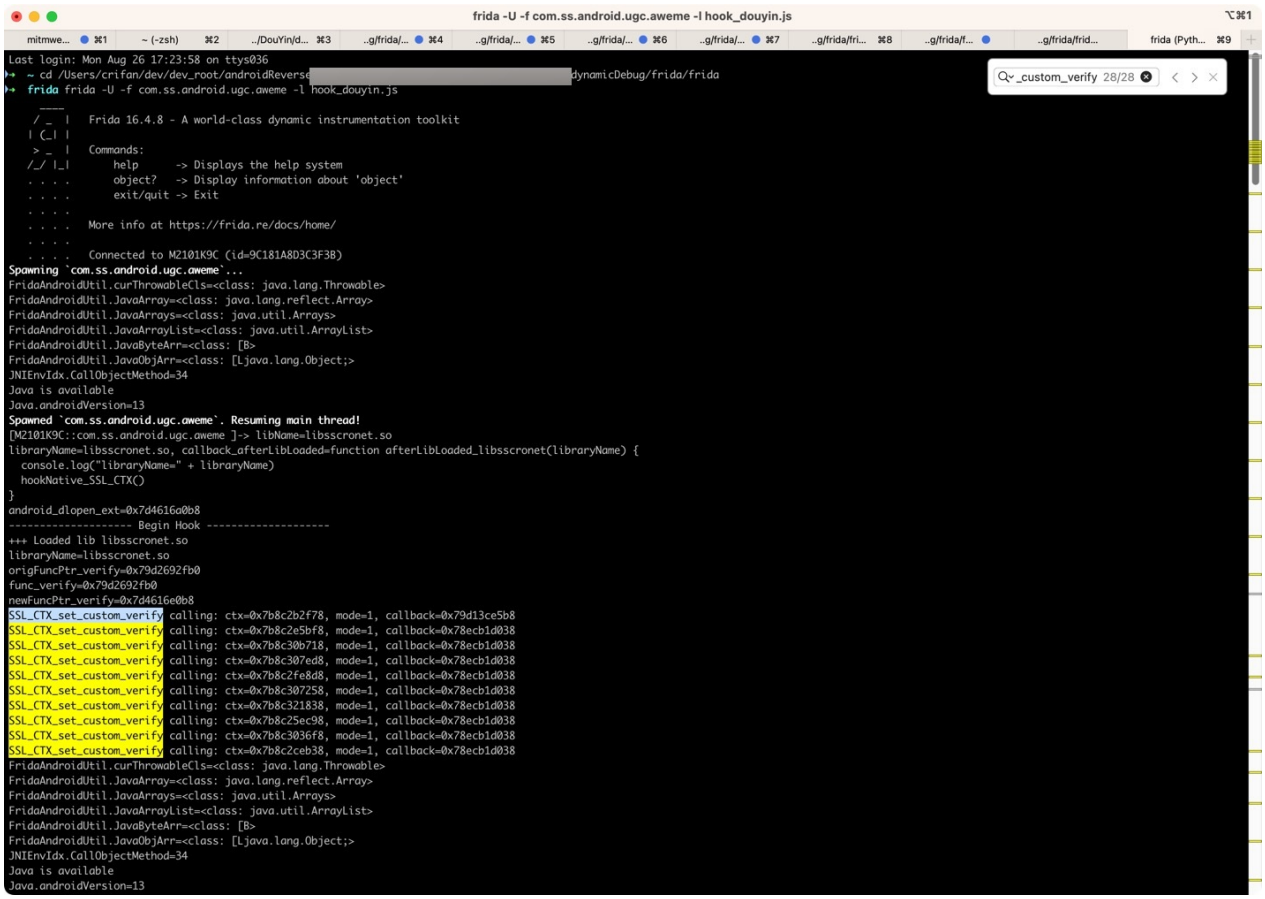
```
        // console.log("libFullPath=" + libFullPath)
        // var flags = args[1]
        // var info = args[2]
        // console.log("android_dlopen_ext: [+] libFullPath=" + libFullPath + ", flags=
" + flags + ", info=" + info)
        // if(libraryName === libFullPath){
        if(libFullPath.includes(libraryName)){
          console.log("+++ Loaded lib " + libraryName)
          this.isLibLoaded = true
        }
      },

      onLeave: function () {
        if (this.isLibLoaded) {
          callback_afterLibLoaded(libraryName)

          this.isLibLoaded = false
        }
      }
    })

  }

  static hookAfterLibLoaded(libName, callback_afterLibLoaded){
    console.log("libName=" + libName)
    FridaAndroidUtil.waitForLibLoading(libName, callback_afterLibLoaded)
  }

}
```

最新（`FridaAndroidUtil`等）工具类函数代码，详见：

https://github.com/crifan/JsFridaUtil/blob/main/frida/FridaAndroidUtil.js

hook输出效果：

```
------------------- Begin Hook -------------------
+++ Loaded lib libsscronet.so
libraryName libsscronet.so
origFuncPtr_verify 0x79d2692fb0
func_verify 0x79d2692fb0
newFuncPtr_verify 0x7d4616e0b8
SSL_CTX_set_custom_verify calling: ctx 0x7b8c2b2f78, mode 1, callback 0x79d13ce5b8
SSL_CTX_set_custom_verify calling: ctx 0x7b8c2e5bf8, mode 1, callback 0x78ecb1d038
SSL_CTX_set_custom_verify calling: ctx 0x7b8c30b718, mode 1, callback 0x78ecb1d038
SSL_CTX_set_custom_verify calling: ctx 0x7b8c307ed8, mode 1, callback 0x78ecb1d038
SSL_CTX_set_custom_verify calling: ctx 0x7b8c2fe8d8, mode 1, callback 0x78ecb1d038
SSL_CTX_set_custom_verify calling: ctx 0x7b8c307258, mode 1, callback 0x78ecb1d038
SSL_CTX_set_custom_verify calling: ctx 0x7b8c321838, mode 1, callback 0x78ecb1d038
SSL_CTX_set_custom_verify calling: ctx 0x7b8c25ec98, mode 1, callback 0x78ecb1d038
SSL_CTX_set_custom_verify calling: ctx 0x7b8c3036f8, mode 1, callback 0x78ecb1d038
SSL_CTX_set_custom_verify calling: ctx 0x7b8c2ceb38, mode 1, callback 0x78ecb1d038
```



效果：

抖音可以正常访问网络了：


10:49
直播 经验 同城 关注 商城 推荐
你说有没有一种可能
观看30秒领取1000金币
直播
sisley
恭喜你获得
+13335金币
65
567
2
167
16
104
31
@活跃的探索精灵（阿清）
#惊不惊喜意不意外 #抖音极速版全民涨金币 #抖音极速版如何稳定开出150000金币 #抖音极速版领金币技巧 #这谁顶得住啊
开宝箱
66
99
相关搜索 · 抖音极速版逛街领金币10000
首页
朋友
消息
我

修改并替换libsscronet.so库文件的方式

此处，要去给抖音的 `libsscronet.so` 去patch打补丁 == 修改并替换 `libsscronet.so` 的so库文件的方式：

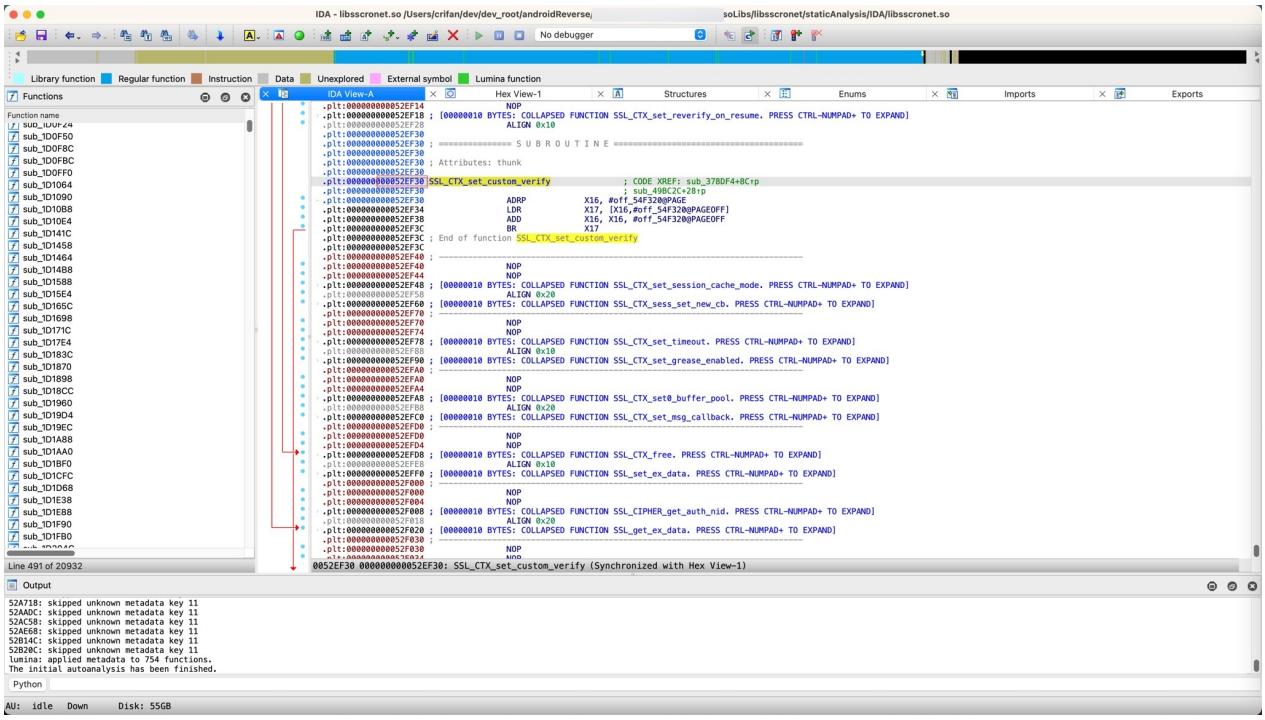
实现绕过证书校验，实现反反抓包的效果：

修改libsscronet.so的so库文件

要patch的点=具体位置：

先是从函数 `SSL_CTX_set_custom_verify` 的调用的地方：

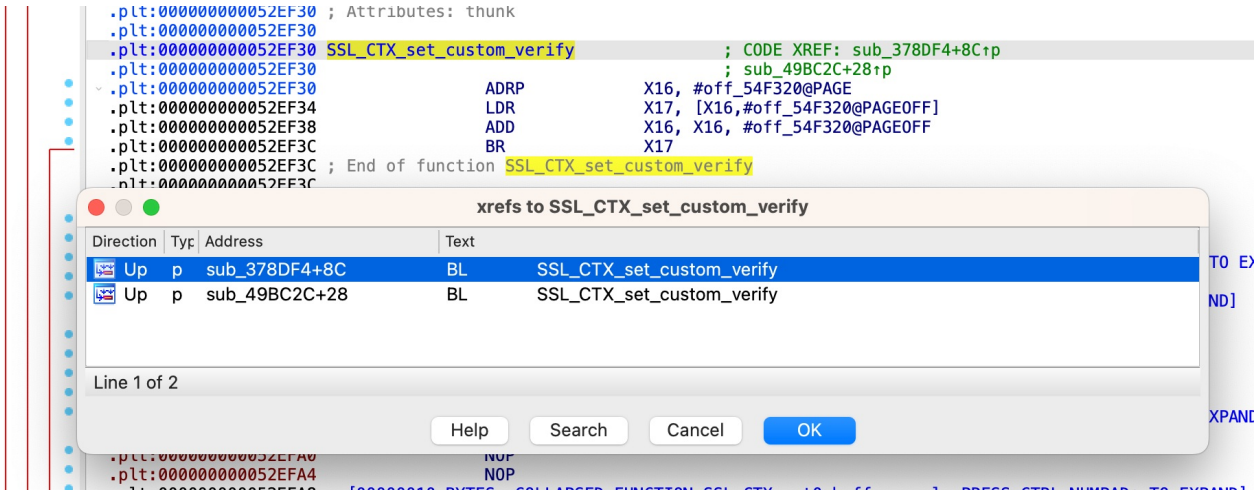
通过IDA，可以找到函数 `SSL_CTX_set_custom_verify` 的实际二进制内偏移地址是： `0x52EF30`



通过：

`SSL_CTX_set_custom_verify` == `0x52EF30`

找到2处引用：

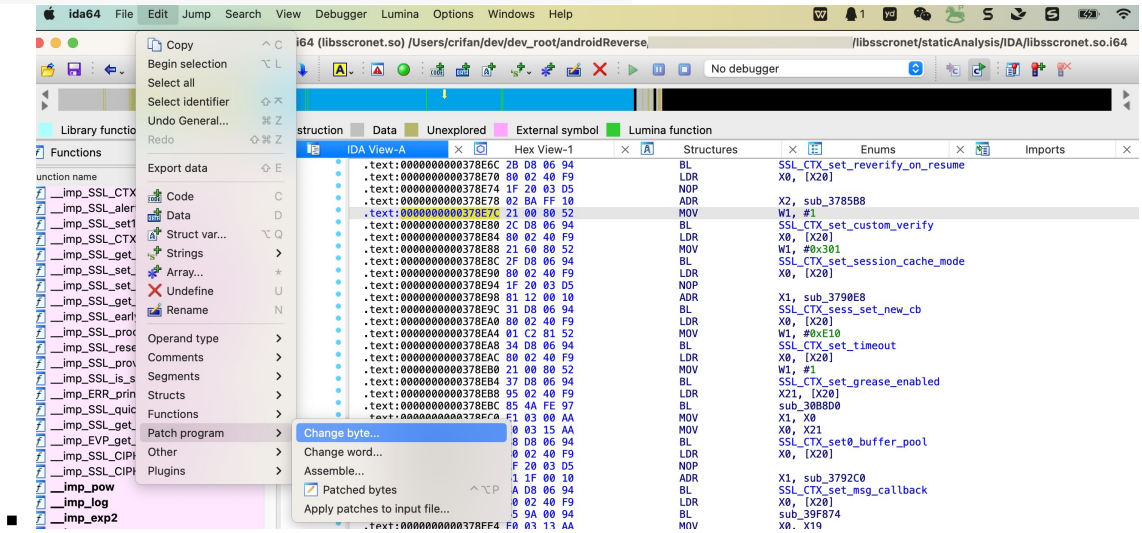


->

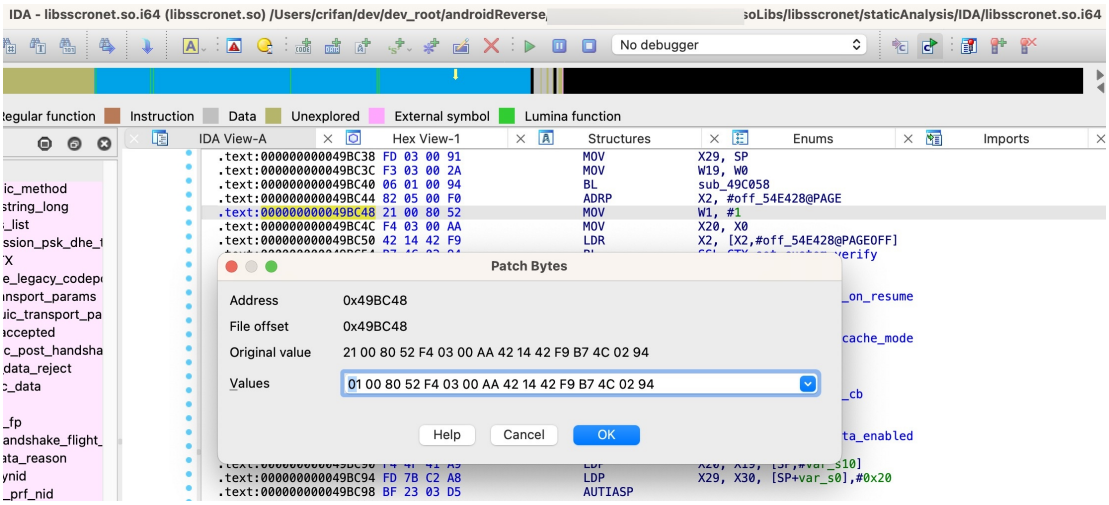
- `sub_378DF4 + 8C`
  - `.text:0000000000378E7C MOV W1, #1`
- `sub_49BC2C + 28`
  - `.text:000000000049BC48 MOV W1, #1`

加上 `opcode` 后显示为：

- `sub_378DF4`
  - `.text:0000000000378E7C 21 00 80 52 MOV W1, #1`



- `sub_49BC2C`
  - `.text:000000000049BC48 21 00 80 52 MOV W1, #1`



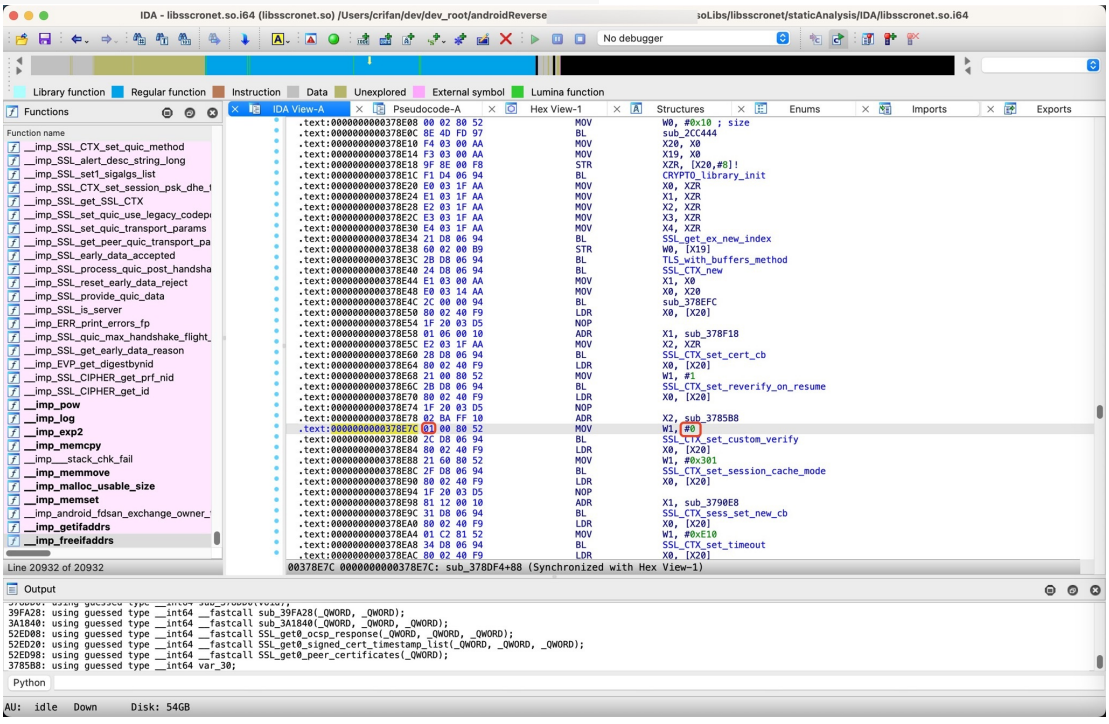
根据之前的hook思路，就是：

让传递给 `SSL_CTX_set_custom_verify` 的参数中 `mode` 参数的值从 `1` 改为 `0` ，而 `mode` 参数值，是保存在 `W1` 中的，所以就是去：

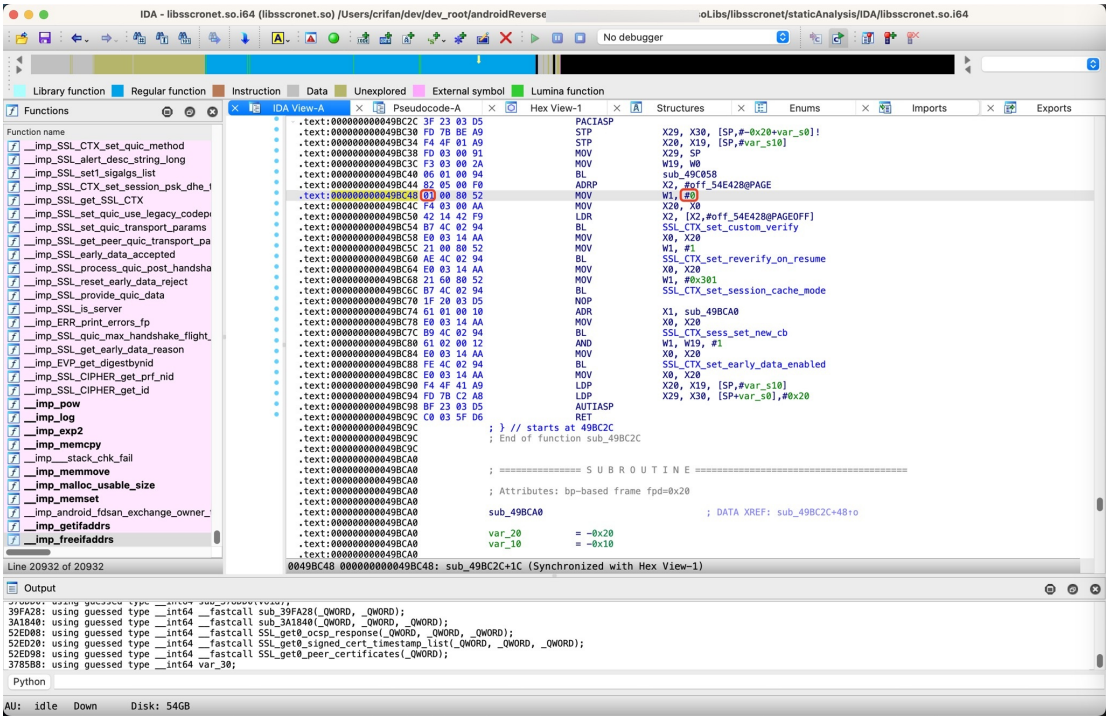
把 `W1` 的值(从之前的 `1`)改为 `0`

分别去改为：

- `sub_378DF4`
  - `.text:0000000000378E7C 01 00 80 52 MOV W1, #0`



- `sub_49BC2C`
  - `.text:000000000049BC48 01 00 80 52 MOV W1, #0`



替换libsscronet.so的so库文件

本身手动即可去替换修改后的so库文件，到安卓手机中已安装的app的相关路径下面的旧的so库文件

不过为了简化，目前已写成搅脚本自动化操作：

`/Users/crifan/dev/dev_root/androidReverse/xxx/Douyin/dy297/Douyin/dy297/libsscronet_so/autoReplaceSo/autoReplaceSo.sh`

```
#!/bin/bash
# Function: Script to auto replace Douyin libsscronet.so
# Author: Crifan Li
# Update: 20240923

# SEPERATOR="--------------------"
SEPERATOR="======================"

function log() {
  echo "${SEPERATOR} $1 ${SEPERATOR}"
}

LIBSSCRONET_SO="libsscronet.so"
echo "LIBSSCRONET_SO=${LIBSSCRONET_SO}"

SO_NEW_PATH_PC="/Users/crifan/dev/dev_root/androidReverse/xxx/Douyin/dy297/Douyin/dy297/libsscronet_so/input/libsscronet_yyy.so"
# SO_NEW_PATH_PC="/Users/crifan/dev/dev_root/androidReverse/xxx/Douyin/dy297/Douyin/dy297/libsscronet_so/input/libsscronet_crifan.so"
# SO_NEW_PATH_PC="/Users/crifan/dev/dev_root/androidReverse/xxx/Douyin/dy313/input/libsscronet/libsscronet_yyy.so"
echo "SO_NEW_PATH_PC=${SO_NEW_PATH_PC}"
SO_TEMP_DOWNLOAD_FOLDER="/sdcard/Download/"
echo "SO_TEMP_DOWNLOAD_FOLDER=${SO_TEMP_DOWNLOAD_FOLDER}"
```

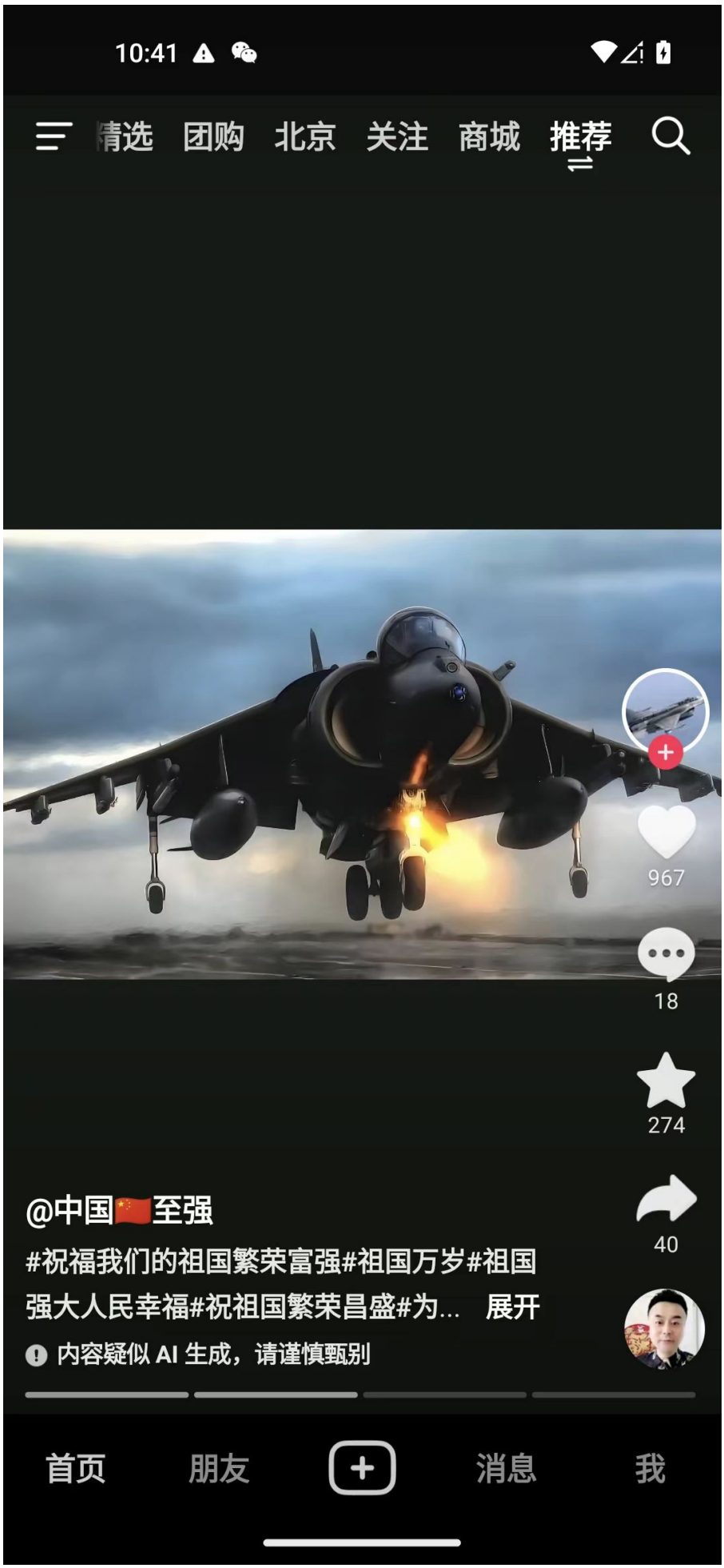
```
SO_TEMP_DOWNLOAD_PATH="${SO_TEMP_DOWNLOAD_FOLDER}/${LIBSSCRONET_SO}"
echo "SO_TEMP_DOWNLOAD_PATH=${SO_TEMP_DOWNLOAD_PATH}"

# echo "${SEPERATOR} Download ${LIBSSCRONET_SO} into android ${SEPERATOR}"
log "Download ${LIBSSCRONET_SO} into android"
adb push ${SO_NEW_PATH_PC} ${SO_TEMP_DOWNLOAD_FOLDER}/${LIBSSCRONET_SO}

# echo "${SEPERATOR} Find ${LIBSSCRONET_SO} location ${SEPERATOR}"
log "Find ${LIBSSCRONET_SO} location"
# LIBSSCRONET_SO_ROOT_FOLDER="/data/app"
LIBSSCRONET_SO_ROOT_FOLDER="/data/data" # /data/data/com.ss.android.ugc.aweme/app_libra
rian/29.7.0.6746496378/libsscronet.so
SO_FILE_ANDROID=$(adb shell find ${LIBSSCRONET_SO_ROOT_FOLDER} -name ${LIBSSCRONET_SO})
echo "SO_FILE_ANDROID=${SO_FILE_ANDROID}"

# echo "${SEPERATOR} Extract ${LIBSSCRONET_SO} folder path ${SEPERATOR}"
log "Extract ${LIBSSCRONET_SO} folder path"
# SO_FOLDER_ANDROID=${SO_FILE_ANDROID#libsscronet}
# SO_FOLDER_ANDROID=${SO_FILE_ANDROID##libsscronet}
# SO_FOLDER_ANDROID=${${SO_FILE_ANDROID}#libsscronet.so}
SO_PATH_LEN=${#SO_FILE_ANDROID}
echo "SO_PATH_LEN=${SO_PATH_LEN}"
SO_FILE_LEN=${#LIBSSCRONET_SO}
echo "SO_FILE_LEN=${SO_FILE_LEN}"
SO_FOFER_LEN=$((${SO_PATH_LEN}-${SO_FILE_LEN}))
echo "SO_FOFER_LEN=${SO_FOFER_LEN}"
# SO_FOLDER_ANDROID=${SO_FILE_ANDROID##libsscronet}
# SO_FOLDER_ANDROID=${SO_FILE_ANDROID%libsscronet}
# SO_FOLDER_ANDROID=${SO_FILE_ANDROID:96}
# SO_FOLDER_ANDROID=${SO_FILE_ANDROID:0:97}
SO_FOLDER_ANDROID=${SO_FILE_ANDROID:0:${SO_FOFER_LEN}}
echo "SO_FOLDER_ANDROID=${SO_FOLDER_ANDROID}"

# echo "${SEPERATOR} Into adb shell ${SEPERATOR}"
log "Into adb shell"

adb shell <<EOF
uname -a
ls -lh ${SO_TEMP_DOWNLOAD_FOLDER} | grep ${LIBSSCRONET_SO}
# cd /data/app
# pwd
# find . -name libsscronet.so
# SO_FILE_ANDROID=$(find . -name libsscronet.so)
# echo "SO_FILE_ANDROID=${SO_FILE_ANDROID}"
cd ${SO_FOLDER_ANDROID}
# echo "${SEPERATOR} Into ${SO_FOLDER_ANDROID} ${SEPERATOR}"
log "Into ${SO_FOLDER_ANDROID}"
pwd
# echo "${SEPERATOR} Remove ${LIBSSCRONET_SO} ${SEPERATOR}"
log "Remove ${LIBSSCRONET_SO}"
ls -lh | grep ${LIBSSCRONET_SO}
rm -f ${LIBSSCRONET_SO}
ls -lh | grep ${LIBSSCRONET_SO}
# echo "${SEPERATOR} Move ${LIBSSCRONET_SO} to here ${SEPERATOR}"
log "Move ${LIBSSCRONET_SO} to here"
mv ${SO_TEMP_DOWNLOAD_PATH} ${SO_FOLDER_ANDROID}
```

```
ls -lh | grep ${LIBSSCRONET_SO}
# echo "${SEPERATOR} Change permission for ${LIBSSCRONET_SO} ${SEPERATOR}"
log "Change permission for ${LIBSSCRONET_SO}"
chmod 755 ${LIBSSCRONET_SO}
chown system ${LIBSSCRONET_SO}
chgrp system ${LIBSSCRONET_SO}
ls -lh | grep ${LIBSSCRONET_SO}
EOF
```

之前某次的运行效果：

```
➜  libsscronet_so autoReplaceSo/autoReplaceSo.sh
LIBSSCRONET_SO libsscronet.so
SO_NEW_PATH_PC /Users/crifan/dev/dev_root/androidReverse/xxx/DouYin/dy297/libsscronet_so/input/libsscronet_crifan.so
SO_TEMP_DOWNLOAD_FOLDER /sdcard/Download/
SO_TEMP_DOWNLOAD_PATH /sdcard/Download//libsscronet.so
====================== Download libsscronet.so into android ======================
/Users/crifan/dev/dev_root/androidReverse/xxx/...: 1 file pushed, 0 skipped. 97.4 MB/s (5567544 bytes in 0.055s)
====================== Find libsscronet.so location ======================
SO_FILE_ANDROID /data/app/~~cJq-AFnKyZwZZqpRJY9hWA==/com.ss.android.ugc.aweme-1LCghzY2HKvNbI0WF3vccQ==/lib/arm64/libsscronet.so
====================== Extract libsscronet.so folder path ======================
SO_PATH_LEN 111
SO_FILE_LEN 14
SO_FOFER_LEN 97
SO_FOLDER_ANDROID /data/app/~~cJq-AFnKyZwZZqpRJY9hWA==/com.ss.android.ugc.aweme-1LCghzY2HKvNbI0WF3vccQ==/lib/arm64/
====================== Into adb shell ======================
Linux localhost 4.19.269 #1 SMP PREEMPT Sat Jul 27 21:05:42 CST 2024 aarch64 Toybox
-rw-rw---- 1 u0_a216 media_rw 5.3M 2024-08-28 09:41 libsscronet.so
/data/app/~~cJq-AFnKyZwZZqpRJY9hWA==/com.ss.android.ugc.aweme-1LCghzY2HKvNbI0WF3vccQ==/lib/arm64
-rwxr-xr-x 1 system system 5.3M 2024-04-22 22:06 libsscronet.so
-rw-rw---- 1 u0_a216 media_rw 5.3M 2024-08-28 09:41 libsscronet.so
-rwxr-xr-x 1 system system 5.3M 2024-08-28 09:41 libsscronet.so
```

替换so后，重启抖音，抖音即可正常访问网络：


10:41
精选
团购
北京
关注
商城
推荐
967
18
274
40
@中国🇨🇳至强
#祝福我们的祖国繁荣富强#祖国万岁#祖国强大人民幸福#祝祖国繁荣昌盛#为... 展开
内容疑似 AI 生成，请谨慎甄别
首页
朋友
消息
我

正常抓包了。

附录

如何确定要修改的值的逻辑 = 应该修改二进制指令为何值?

此处是要修改函数 `SSL_CTX_set_custom_verify` 的参数，此处是第二个参数= `mode` 参数=存放在 `W1` 中的值：从 `1` 变成 `0`

```
MOV             W1, #1
```

改为：

```
MOV             W1, #0
```

如何把要改的指令生成对应的Opcode操作码?

而已知要去把：

```
MOV             W1, #0
```

的改动写入到so库文件中，但是对应的opcode操作码的值是什么，此时并不知道

所以可以借助于

指令和二进制opcode互转 · 最流行汇编语言：ARM

中的：

Online ARM to HEX Converter (armconverter.com)

去生成此处的，ARM64的二级制的汇编指令码

从而才能搞懂：

原始的字节码 -》 对应的汇编代码含义

以及，要修改成什么汇编代码 -》 所对应的新的字节码是多少

举例：

之前的汇编代码是：

```
MOV             W1, #1
```

对应着：

ARM64的字节码是：

```
21008052
```

而要改为：

```
MOV             W1, #0
```

对应的ARM64字节码是：

```
01008052
```

》所以我们最终要改的数据就是：

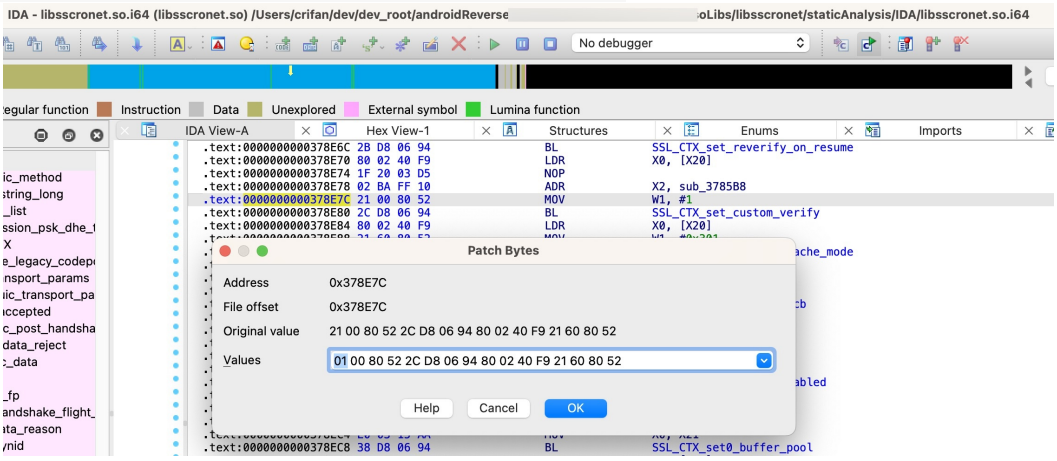
- 从：`21 00 80 52`
- 改为：`01 00 80 52`

-》对应到IDA中就是

- `0x378E7C`
  - `Origin value`
    - `21 00 80 52 2C D8 06 94 80 02 40 F9 21 60 80 52`
      - 注：后续有更多的二进制字节码，不用管，保持原始值即可
        - `2C D8 06 94 80 02 40 F9 21 60 80 52`
          - 且后续注意到：是后续其他3行汇编代码对应的汇编指令

```
.text:0000000000378E80 2C D8 06 94          BL        SSL_CTX_set_
custom_verify
.text:0000000000378E84 80 02 40 F9          LDR       X0, [X20]
.text:0000000000378E88 21 60 80 52          MOV       W1, #0x301
```

  - `Values`
    - `01 00 80 52 2C D8 06 94 80 02 40 F9 21 60 80 52`



IDA中如何修改二进制=打补丁=patch

- 概述
  - IDA->Edit-》Patch program-》Change byte-》修改数据-》OK
  - IDA->Edit-》Patch program-》Apply patches to input file-》OK
- 详解

- 打补丁 · 逆向利器：IDA

iOS版抖音

绕过iOS版抖音的证书绑定校验

比如iOS版 `抖音` 内部就做了证书校验

想要逆向，绕过抖音的证书绑定校验话，可以通过写hook代码绕过证书校验，即可实现抓包 `HTTPS` 看到明文数据。

具体步骤：

之前已给此处iPhone7搭建过Charles抓包环境：

- 已安装过Charles的根证书： `charles-ssl-proxying-certificate.cer`
  - 且已信任过

此处只需要去：

把Charles的SSL根证书放入iPhone中对应的的位置，比如：

`/Library/PreferenceLoader/Preferences/charles/charles-ssl-proxying-certificate.cer`

然后再去写hook代码：

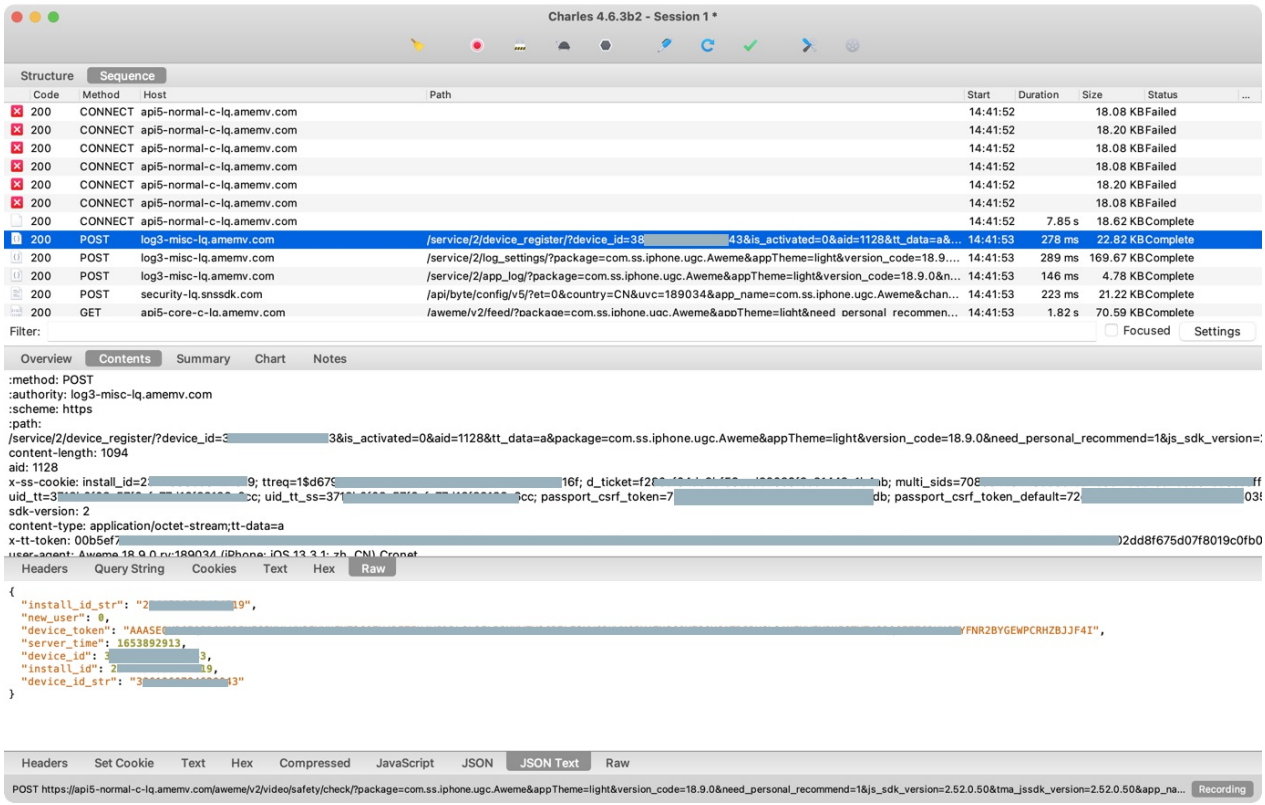
```
/*============================================================================
 Hook: Charles cert -> bypass https capture
============================================================================*/

// to fix Charles capture https show Unknown issue
// https://bbs.pediy.com/thread-270700.htm
// https://iosre.com/t/topic/20202/46

#define CHARLES_CERT_FILE @"/Library/PreferenceLoader/Preferences/charles/charles-ssl-proxying-certificate.cer"

bool cfgHookEnable_aweme = true;

%hook TTNetworkManagerChromium

- (NSArray *)ServerCertificate {
    iosLogDebug("%s", "");
    NSArray* serverCertList = %orig();
    iosLogDebug("serverCertList=%{public}@", serverCertList);

    ////    NSString *serverCertListStr = [NSArray nsStrListToStr:serverCertList isSortList:FALSE isAddIndexPrefix:TRUE];
//    NSString *serverCertListStr = [CrifanLibHookiOS nsStrListToStr:serverCertList isSortList:FALSE isAddIndexPrefix:TRUE];
//    iosLogInfo("serverCertListStr=%{public}@", serverCertListStr);
//
////    return nil;
//    return serverCertList;
```

```
    NSMutableArray *newCertList = [NSMutableArray arrayWithArray:serverCertList];
    iosLogDebug("newCertList=%{public}@, newCertList.count=%{public}lu", newCertList, [
newCertList count]);

    if (cfgHookEnable_aweme) {
        NSString *certResourcePath = CHARLES_CERT_FILE;
        iosLogDebug("certResourcePath=%{public}@", certResourcePath);

        NSFileManager *defaultManager = [NSFileManager defaultManager];
        BOOL isExistedCert = [defaultManager fileExistsAtPath:certResourcePath];
        iosLogInfo("isExistedCert=%s", boolToStr(isExistedCert));
        if (isExistedCert) {
            NSData *certP12Data = [NSData dataWithContentsOfFile:certResourcePath];
            iosLogDebug("certP12Data=%{public}@", certP12Data);
            [newCertList addObject:certP12Data];
            iosLogDebug("newCertList=%{public}@, newCertList.count=%{public}lu", newCer
tList, [newCertList count]);
        }
    }

    NSMutableArray *retNewCertList = [newCertList copy];
    iosLogDebug("retNewCertList=%{public}@", retNewCertList);
    return retNewCertList;
}

%end
```

再去Charles做好基本的代理配置：

- Charles中开启代理
- iPhone中使用Mac中的Charles代理

即可顺利实现iPhone7中抖音的https的抓包，看到https的明文了：



- 注：此处代码只对当时的某些版本，比如：抖音 `17.8` 、 `18.9` ，是有效的
  - -》 目前最新版本抖音，此代码已无效。
    - -》 但是也有人说，最新版本也是有效的。待各位自己确认核实。

附录

下面列出相关参考资料。

参考资料

- 【已解决】绕过抖音反抓包网络错误：Frida去hook校验函数SSL_CTX_set_custom_verify
- 【已解决】绕过抖音反抓包网络错误：给libsscronet.so二进制打补丁patch
- 【记录】IDA中给libsscronet.so打补丁patch program修改二进制
- 【记录】给抖音douyin297.apk替换库文件libsscronet.so
- 【记录】给改机小米的Pixel5中的抖音：替换库文件libsscronet.so
- 【已解决】安卓adb push导入文件报错：adb error failed to copy remote couldn't create file Permission denied
- 【已解决】安卓手机中寻找已安装抖音的data数据的目录
- 【已解决】mitmdump抓包改机小米后的Pixel5中抖音失败且抖音提示：网络错误，当前无网络，请检查后重试
- 【已解决】用脚本自动化处理替换apk中so库文件libsscronet.so的过程
- 【记录】抖音so库libsscronet.so：静态分析
- 【记录】给抖音v29.7.0去替换libsscronet.so库
- 【记录】给抖音31.3.0的apk替换libsscronet.so使得可以访问网络和抓包
- 【记录】抖音的反抓包相关网络错误：libsscronet.so库和SSL_CTX_set_custom_verify
- 【记录】对比libsscronet.so二进制打补丁的区别：别人修改的和我们修改的
- 【记录】研究疑似libmetasec_ml.so调用到libsscronet.so函数：Cronet_Engine_SetOpaque
-
- SSL证书绑定 · iOS安全与防护 (crifan.org)
- tiktok加密参数分析_tiktik iid-CSDN博客
- android逆向奇技淫巧二十九：x音AES使用分析 - 第七子007 - 博客园
- Disable SSL pinning using Frida (github.com)
-