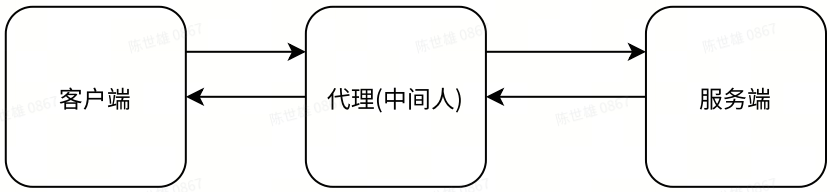
解锁抓包新姿势

原理简述

- 抓包通过代理实现（Http、Socks等）又称中间人代理。客户端的请求经过代理后到达服务端，然后服务端返回的数据又经过代理才到达客户端



- Http代理：作用于应用层上，只允许通过Http协议访问外部网站
- Socks代理：作用于会话层，只是简单地传递数据包，无需关心是什么应用协议，比Http代理速度快
- 单向证书校验：客户端内置了服务端证书，请求时客户端校验服务端证书是否合法
- 双向证书校验：在单向校验的基础上，增加了服务端校验客户端的证书合法性

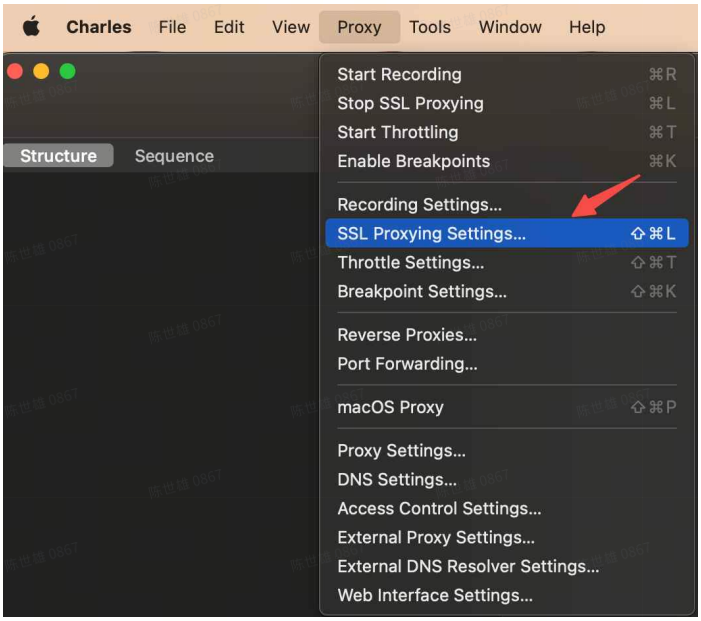
环境准备

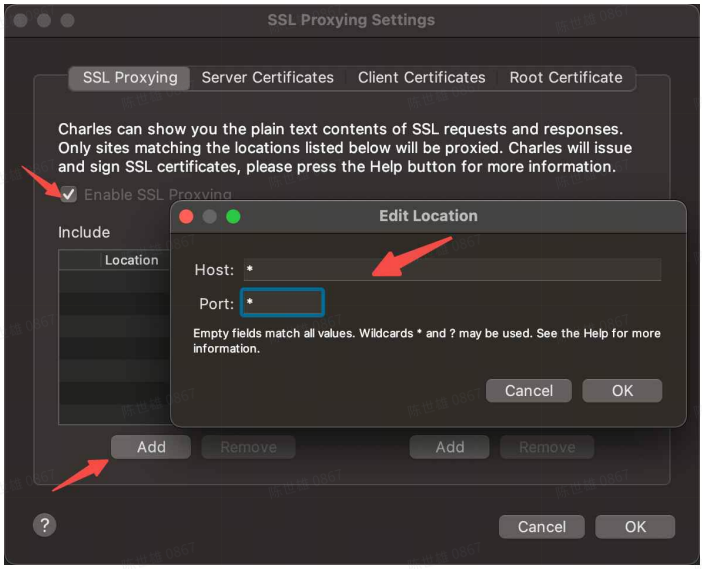
工具名称	工具简介	下载地址
Charles	PC端常用的抓包工具，支持分析 Https 协议。方便调试与服务器端的通讯协议	https://pan.baidu.com/s/X1NuN4MYsQ-SLDpdgJmfFdDQ?pwd=qn2r
VMOS Pro	移动端的虚拟机平台，可轻松开启root和xposed，实现“一机多系统”	
Xposed	APP的hook框架，可在不改包的情况下，修改APP逻辑	
TrustMeAlready	用来绕过证书锁定（SSL Pinning）的Xposed插件	
RootExplorer	高权限文件管理器，获取Root权限后可对系统文件进行操作	

Charles

SSL属性设置

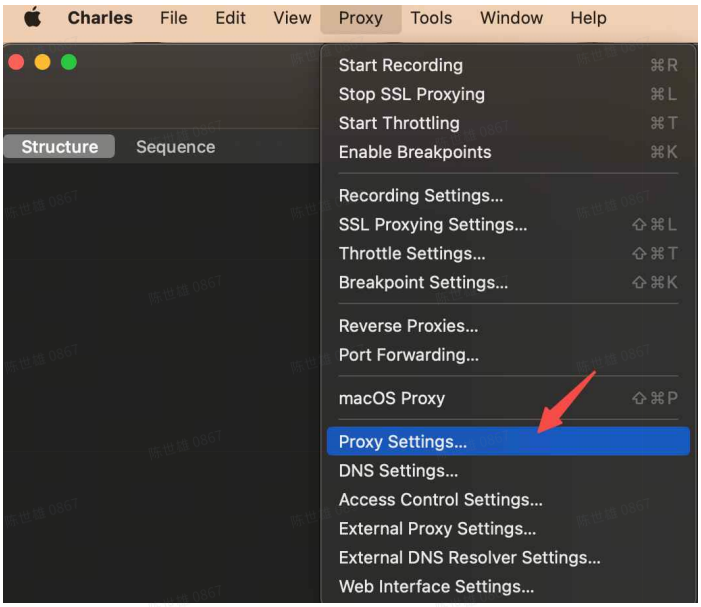
- 配置需要抓取https站点的匹配规则（*表示所有站点）

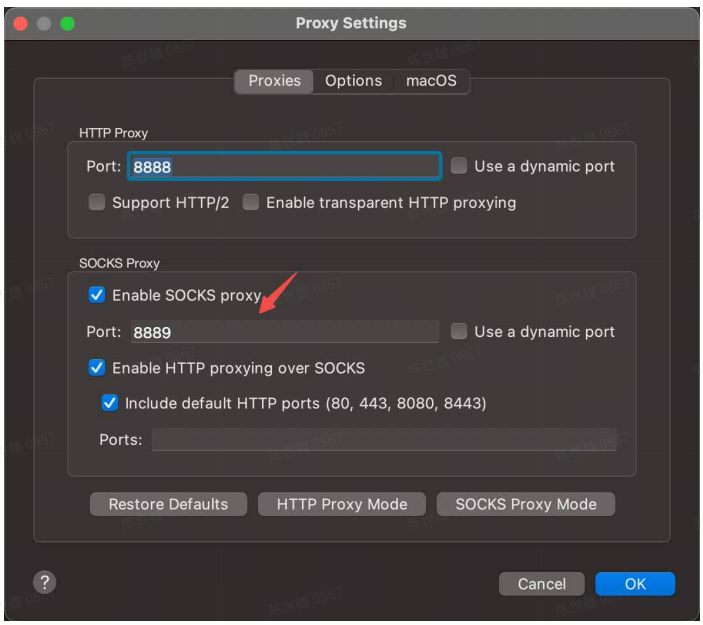




Socks代理设置

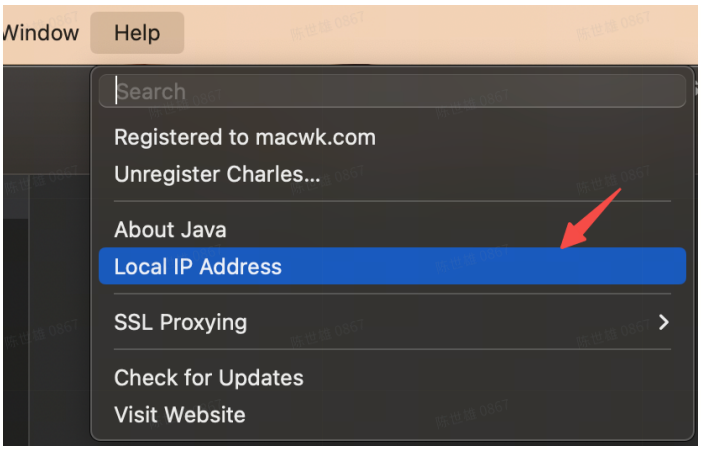
- 开启Socks代理，Port端口可自定义，Http Proxy可根据需要进行开启

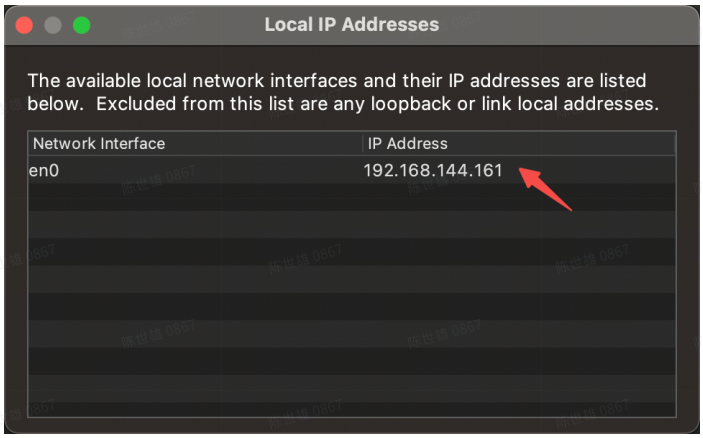




查看本机IP地址：

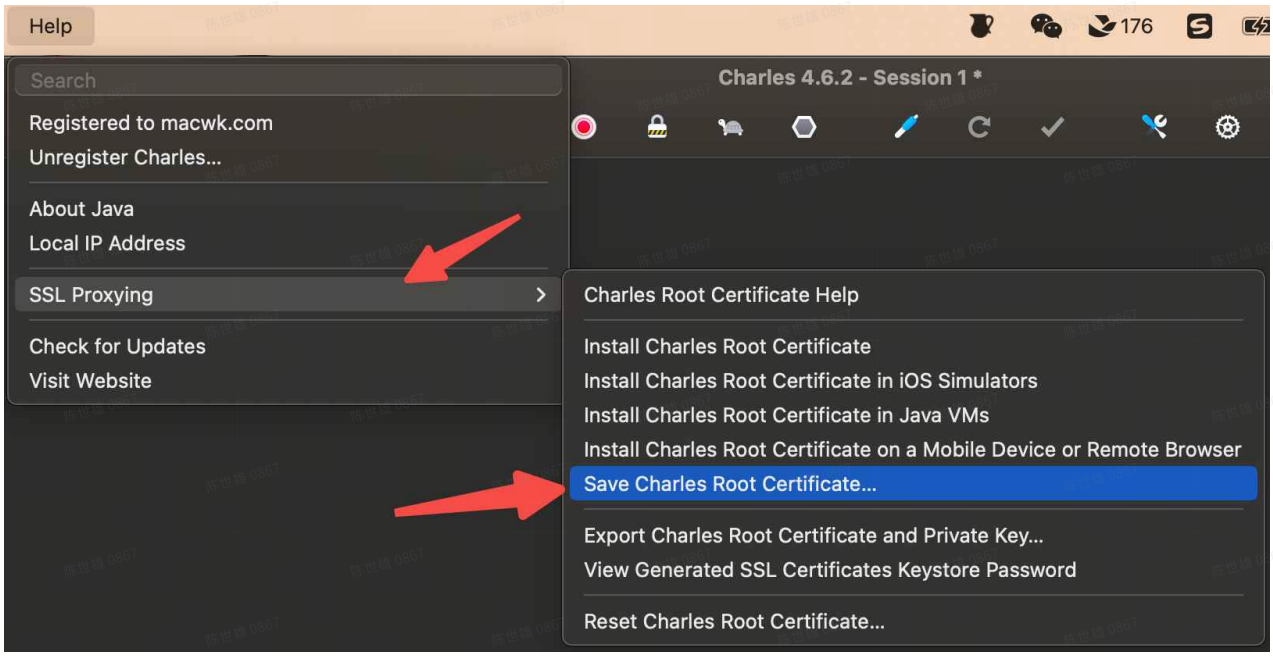
- 后面设置代理的时候会用到IP地址





导出根证书

- 后面抓包需要用到证书

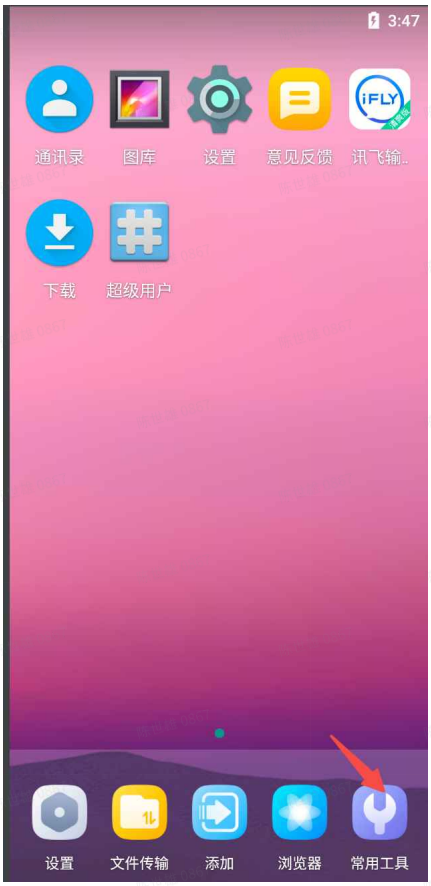


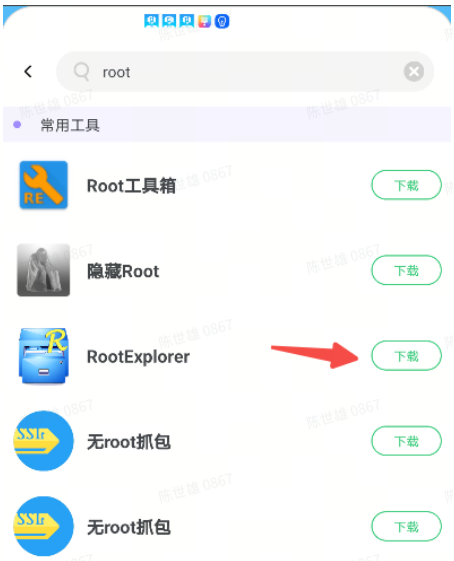
VMOS Pro

添加虚拟机

开启root权限

安装RootExplore





设置Socks5代理：

- 填入Charles查看的本机IP地址和Socks代理端口

常见抓包问题

问题	原因	解决

<table>
<tr><td rowspan="3">已安装用户证书仍无法抓包（无指定证书校验的情况）</td><td rowspan="3">Android 7.0开始使用了更严格的网络安全机制，APP默认不信任用户证书</td><td>使用Android 7.0以下手机</td><td>无限制</td></tr>
<tr><td rowspan="2">≥ Android 7.0版本手机</td><td>• 自己开发的APP：可通过改配置信任用户证书</td></tr>
<tr><td>• 第三方APP：可安装系统证书（需Root权限）</td></tr>
<tr><td>单向证书检验无法抓包</td><td>APP内置了仅被接受的服务器证书，而不接受其它任何证书，通过比对证书是否一致，来确认连接的合法性</td><td colspan="2">使用TrustMeAlready插件可绕过</td></tr>
<tr><td>双向证书校验无法抓包</td><td>除了客户端的校验，服务端也对客户端证书合法性进行校验</td><td colspan="2">• TrustMeAlready 插件
• 反编译获取内置客户端证书和密码，导入抓包工具（Charles）</td></tr>
</table>

案例一：他趣（无指定证书校验）

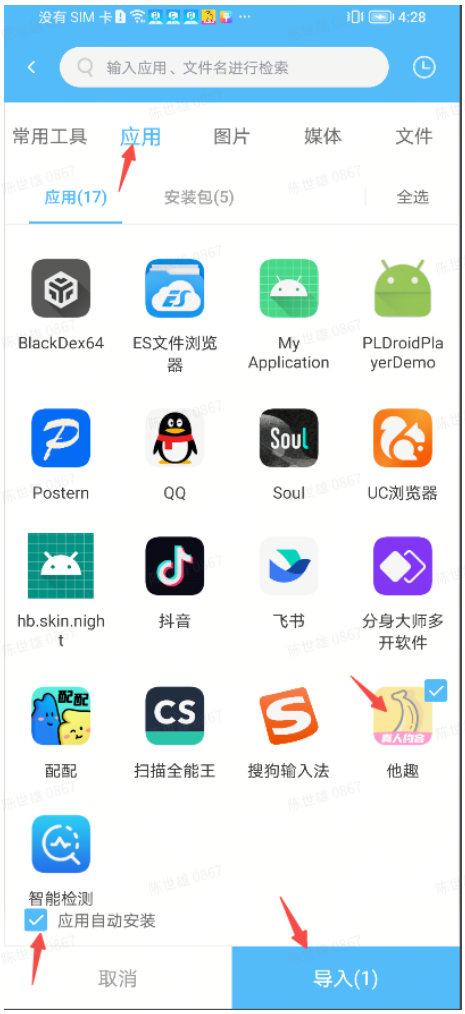
方式一：安装抓包插件（仅限他趣）

自行开发的插件包，他趣APP检测设备如果安装了该插件，则信任用户安装的任何证书

方式二：安装系统证书（通用）

安装他趣APP

- 先在手机上安装app，然后在虚拟机中导入安装



生成手机端证书

- 查看Charles导出的pem根证书的hash值

```
1 openssl x509 -inform PEM -subject_hash_old -in charles-ssl-proxying-
  certificate.pem
```

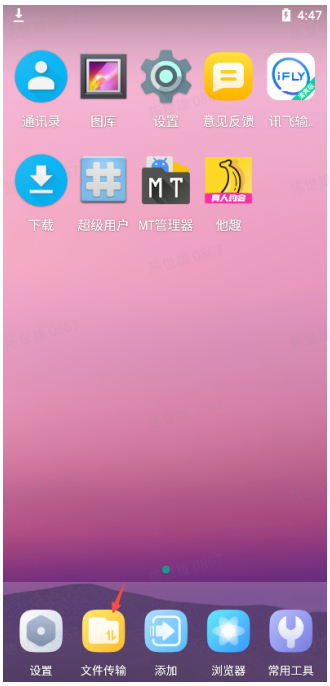
```
yms@ymsdeMBP Desktop % openssl x509 -inform PEM -subject_hash_old -in charles-ssl-proxying-certificate.pem
244a5ad8
-----BEGIN CERTIFICATE-----
MIIFVDCCBDygAwIBAgIGAYIj7LbxMA0GCSqGSIb3DQEBCwUAMIGuMT8wPQYDVQQD
DDZDaGFybGVzIFByb3h5IENBICgyMiBKdWwgMjAyMiwgeW1zZGVNYWNCb29rLVBy
by5sb2NhbCkxJTAjBgNVBAsMHGh0dHBzOi8vY2hhcmxlc3Byb3h5LmNvbS9zc2wx
ETAPBgNVBAoMCFhLNzIgTHRkMREwDwYDVQQHDAhBdWNrbGFuZDERMA8GA1UECAwI
QXVja2xhbmQxCzAJBgNVBAYTAk5aMB4XDTIyMDcyMTAzMjEzMloXDTIzMDcyMTAz
MjEzMlowga4xPzA9BgNVBAMMNkNoYXJsZXMgUHJveHkgQ0EgKDIyIEp1bCAyMDIy
LCB5bXNkZU1hY0Jvb2stUHJvLmxvY2FsKTElMCMGA1UECwwcaHR0cHM6Ly9jaGFy
bGVzcHJveHkuY29tL3NzbDERMA8GA1UECgwIWEs3MiBMdGQxETAPBgNVBAcMCEF1
Y2tsYW5kMREwDwYDVQQIDAhBdWNrbGFuZDELMAkGA1UEBhMCTlowggEiMA0GCSqG
SIb3DQEBAQUAA4IBDwAwggEKAoIBAQCC510TXxhCuO3KdF8jIN9UL7A9PiLVHQ1X
265Ie0RPXqNykE5Bxu79Zd1GT3Uwyru7rqlhyTIzRwHcCU3bxtZY19nBJwUgUIMp
xkmyurnB63fqkvABnl3I5R2h9wGFSXpLyQUlX1Z5cNK6yDdDLaPZrSBd7WJE8Kns
zj6X8vt2rWn0UaZ+4RcJkCPE76krW++syHcLO6Py2yBFKEm0SYC2j5nxRaTfjiMN
ljvK+77sDuJ5xqe0y5eaKSvBDk7IBjN3K39wJcm3VrMvj+3c9CDYP4G/NXGEgyMa
Q9gDIwrA0u583F05Aw7usAmtpgauA+/fdWrxPi7+uk5I0YkvguErAgMBAAGjggF0
MIIBcDAPBgNVHRMBAf8EBTADAQH/MIIBLAYJYIZIAYb4QgENBIIBHROCARlUaGlz
IFJvb3QgY2VydGlmaWNhdGUgd2FzIGdlbmVyYXRlZCBieSBDaGFybGVzIFByb3h5
IGZvciBTU0wgUHJveHlpbmcuIElmIHRoaXMgY2VydGlmaWNhdGUgaXMgcGFydCBv
ZiBhIGNlcnRpZmljYXRlIGNoYWluLCB0aGlzIG1lYW5zIHRoYXQgeW91J3JlIGJy
b3dzaW5nIHRocm91Z2ggQ2hhcmxlcyBQcm94eSB3aXRoIFNTTCBQcm94eWluZyBl
bmFibGVkIGZvciB0aGlzIHdlYnNpdGUuIFBsZWFzZSBzZWUgaHR0cDovL2NoYXJs
ZXNwcm94eS5jb20vc3NsIGZvciBtb3JlIGluZm9ybWF0aW9uLjAOBgNVHQ8BAf8E
BAMCAgQwHQYDVR0OBBYEFDO0j8NltU+DLKlArQ++j+qQyUHoMA0GCSqGSIb3DQEB
CwUAA4IBAQBjIjVCMGP3Q1WFGDHPixxWGV/rkNAEw/+3tZZ6hR/Mj84XDlNUAdB4
dvTglPK5GG9qip2Uplb/SheOfL3SqbMdhIkapw0Rd0bBA1hbZqToexByPraGUhEv
RxjLt6veMNueAWouW2J1Rfahu0hodVqf7o9x4n75gOsg39fyoWDn3MJp5P1SYV6W
RcEuH66h7JYS6jfIeNpv5q/LL+oaxP+L49s8ze+I9Qjdlxab0NJDgCJiDlfS0HQ/
4/zGuU5Cu1xUWOuxj08ndfaNugLjpmq29DIFtoKhD2KonbtC2Um8JZbeOzZr1APo
1jrE7fW6QquZhl70BXcyPqcOntBOOiTq
-----END CERTIFICATE-----
```

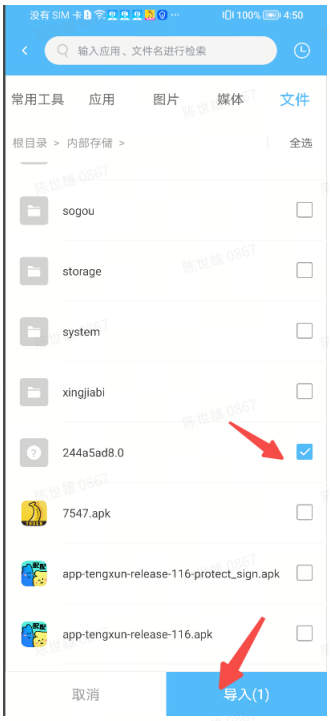
- pem证书重命名——8位数的hash值对证书进行重命名(注意文件后缀改为 0)

```
1 mv charles-ssl-proxying-certificate.pem 244a5ad8.0
```

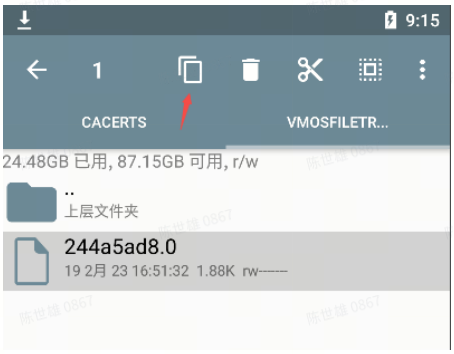
安装系统证书

- 将重命名后的证书 244a5ad8.0 拷贝到手机 sdcard 根目录
- 将证书244a5ad8.0 导入到虚拟机



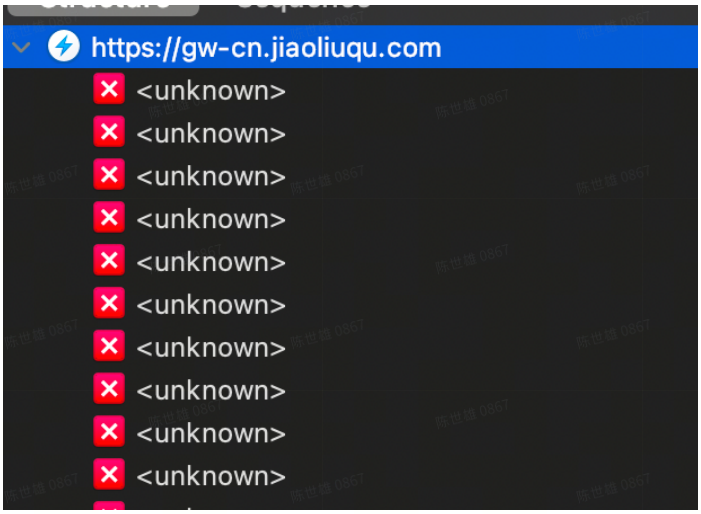


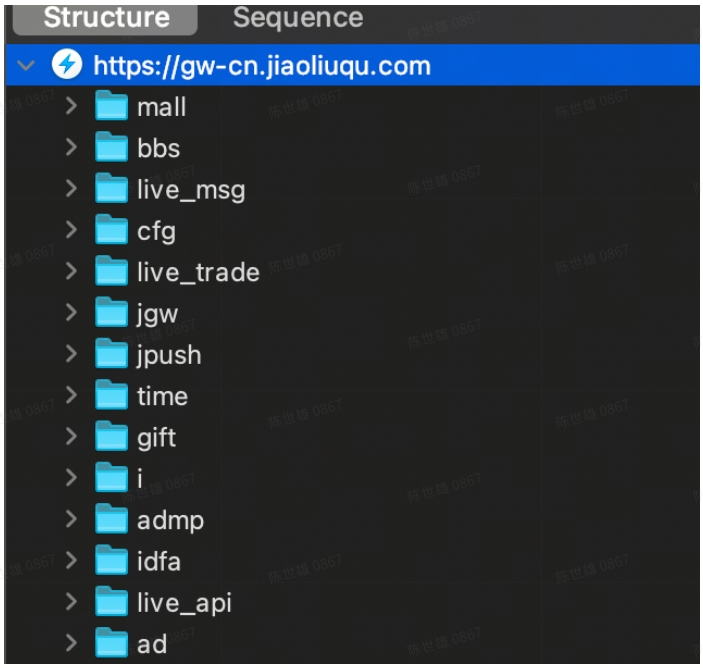
- 打开虚拟机中的MT管理器（需授root权限），把/storage/emulated/0/VMOSfiletransferstation/244a5ad8.0 证书移动到目录 /system/etc/security/cacerts



- 重启虚拟机（无需重启手机）

设置系统证书的抓包前后对比结果





总结

- 系统证书为通用方案，可解决大部分第三方不信任用户证书的抓包问题

案例二：饿了么V9.1.14（单向证书校验）

单向证书校验相关逻辑

- 通过反编译查看，饿了么对部分接口做了单向证书检验

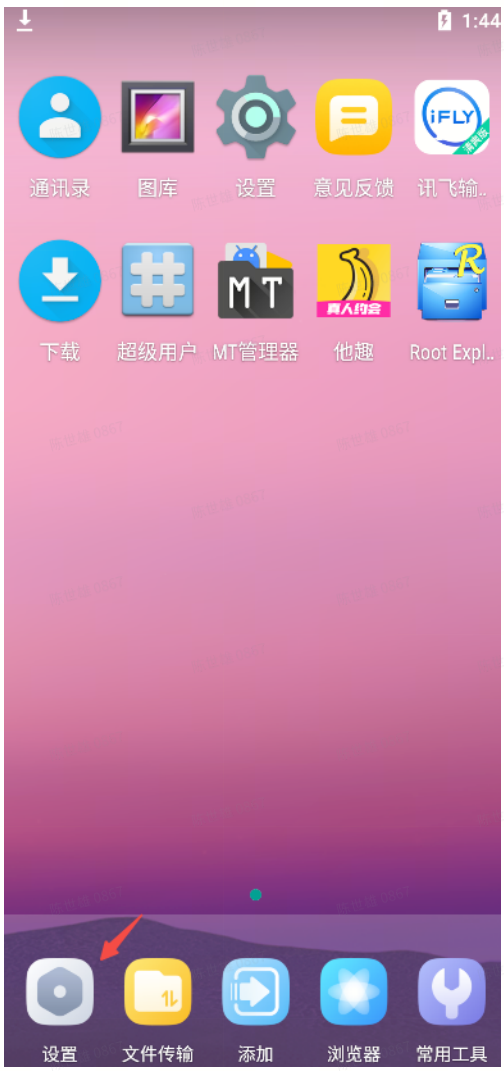
```
1  //第一个入参inputstream是服务端的证书文件流 GeoTrust_Global_CA_All.pem
2  private static SSLContext b(InputStream arg6, boolean arg7) {
3      Collection v1 =
   CertificateFactory.getInstance("X.509").generateCertificates(arg6);
4      if(v1.isEmpty()) {
5          throw new IllegalArgumentException("expected non-empty set of trusted
   certificates");
6      }
7     //证书对应秘钥 "password"
8      char[] v3 = "password".toCharArray();
9      KeyStore v4 = ag.a(v3);
10     Iterator v5 = v1.iterator();
11     int v1_1;
12     for(v1_1 = 0; v5.hasNext(); ++v1_1) {
13         Object v0_1 = v5.next();
14         v4.setCertificateEntry(Integer.toString(v1_1), ((Certificate)v0_1));
15     }
16
17     KeyManagerFactory v0_2 =
   KeyManagerFactory.getInstance(KeyManagerFactory.getDefaultAlgorithm());
18     v0_2.init(v4, v3);
```

```
    // 单向校验
    TrustManagerFactory v1_2 =
TrustManagerFactory.getInstance(TrustManagerFactory.getDefaultAlgorithm());
    v1_2.init(v4);
    SSLContext v2 = SSLContext.getInstance("TLS");
    if(arg7) {
        v2.init(v0_2.getKeyManagers(), new TrustManager[]{new c()}, new
SecureRandom());
        return v2;
    }

    v2.init(v0_2.getKeyManagers(), v1_2.getTrustManagers(), new
SecureRandom());
    return v2;
    }
```

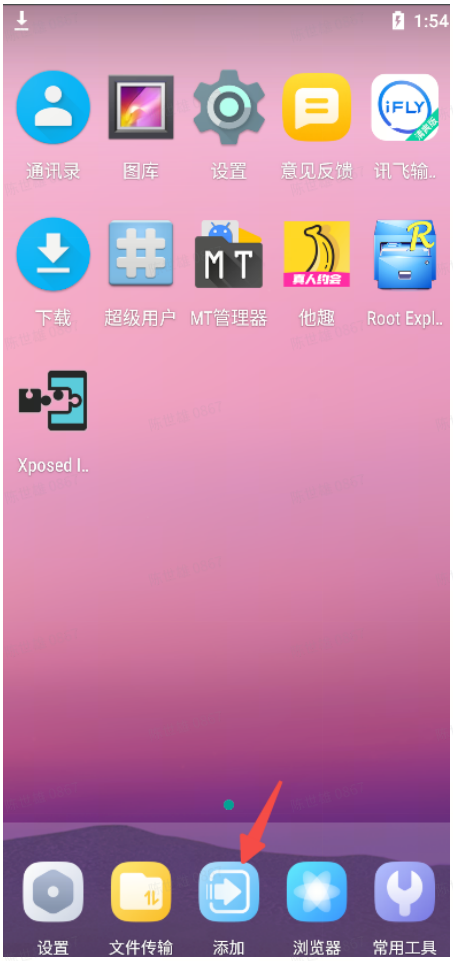
开启xposed框架

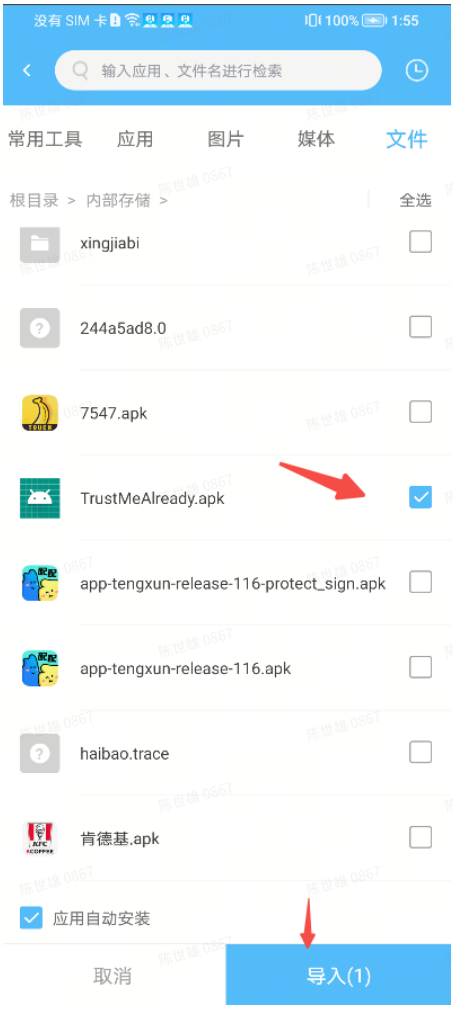
- 开启xposed框架后，需要重启虚拟机（无需重启手机）



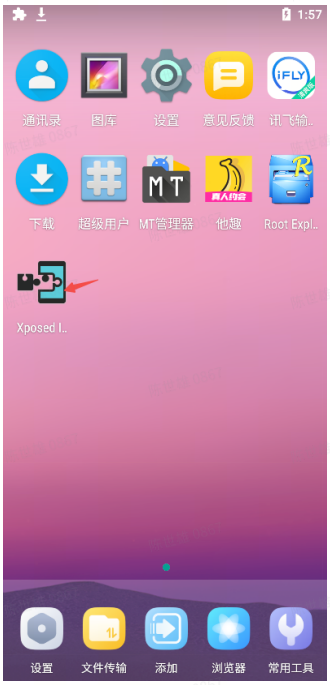
TrustMeAlready 安装&激活

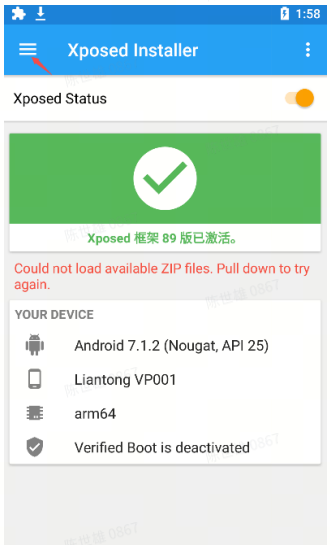
- TrustMeAlready.apk拷贝到手机sdcard根目录
- 虚拟机中导入并安装

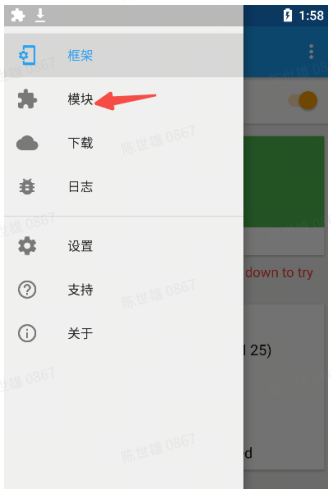


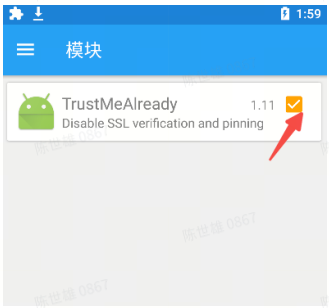


- 激活TrustMeAlready模块（**勾选后需要重启虚拟机**）

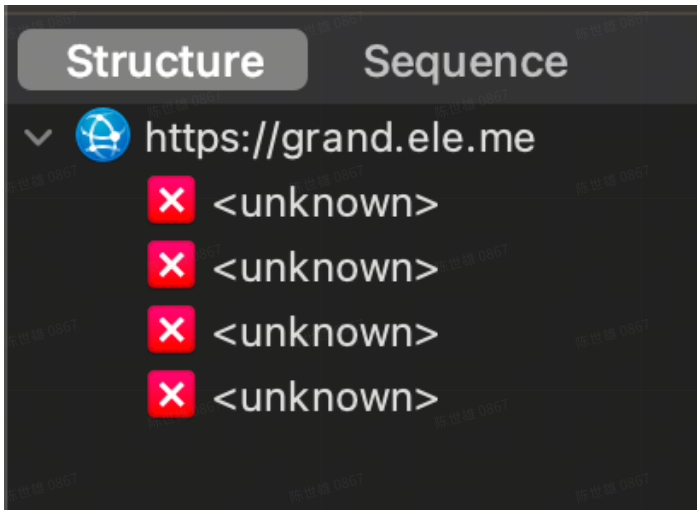








TrustMeAlready抓包前后对比结果



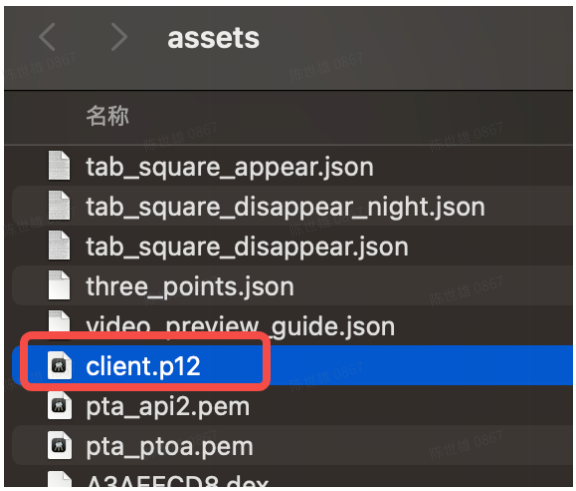
总结

- TrustMeAlready通过hook单向证书校验相关逻辑，去掉了证书校验，使单向校验功能失效

案例三：Soul V3.34.1（双向证书校验）

获取证书文件和秘钥

- 通过反编译获取证书文件：证书文件一般存放在assert或raw目录下，搜索文件后缀.p12或.pem

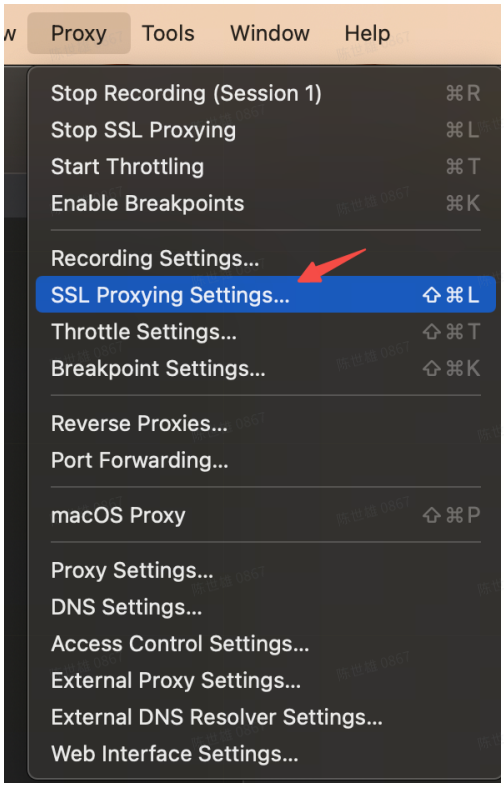


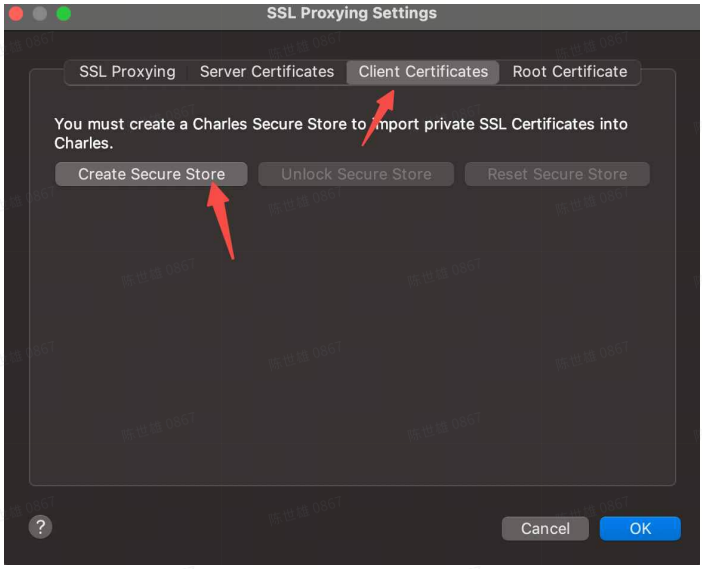
- 通过hook获取秘钥： **}%2R+\OSsjpP!w%X**

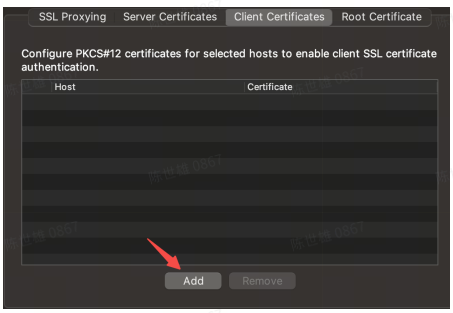
```
// 获取秘钥的关键代码
XposedHelpers.findAndHookMethod(
                "java.security.KeyStore",
                rawClassLoader,
                "load",
                InputStream.class,
                char[].class,
                new XC_MethodHook() {
                    @Override
```

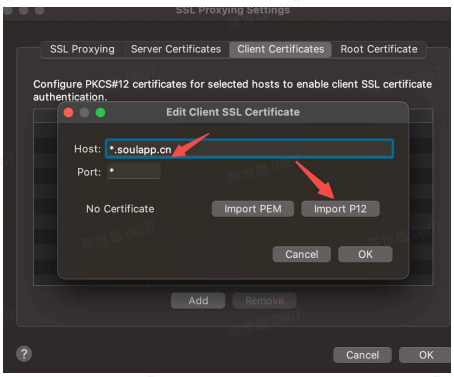
```
                protected void beforeHookedMethod(MethodHookParam param)
    throws Throwable {
                    char[] psw = (char[]) param.args[1];
                    XposedBridge.log("KeyStorettest psw = " + new
    String(psw));
                }
            }
        );
```

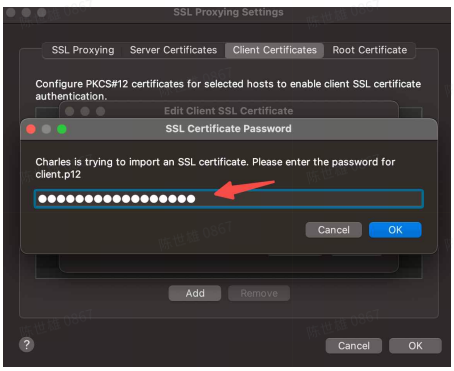
- 将client.p12证书导入charles，并输入密钥



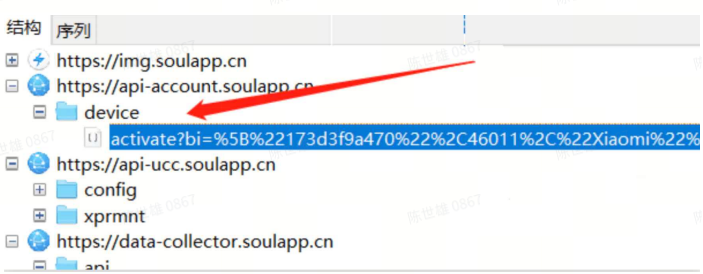


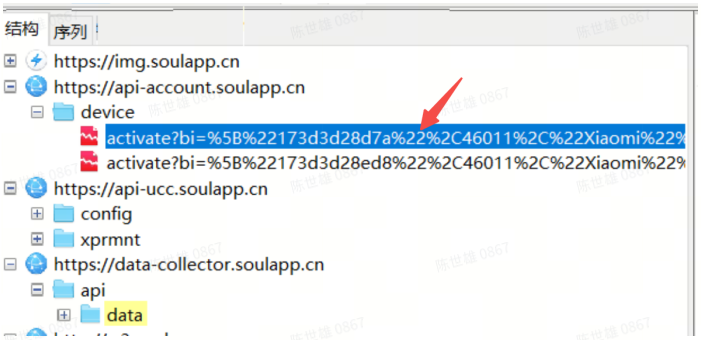






双向认证的抓包前后对比结果





总结

双向认证抓包难度相关较大，需要反编译获取证书和秘钥，如遇app加固，反编译门槛更高

数据安全建议

- 对报文进行加密处理，避免直接明文传输
- 对App进行加固，提高安全门槛，防止反编译