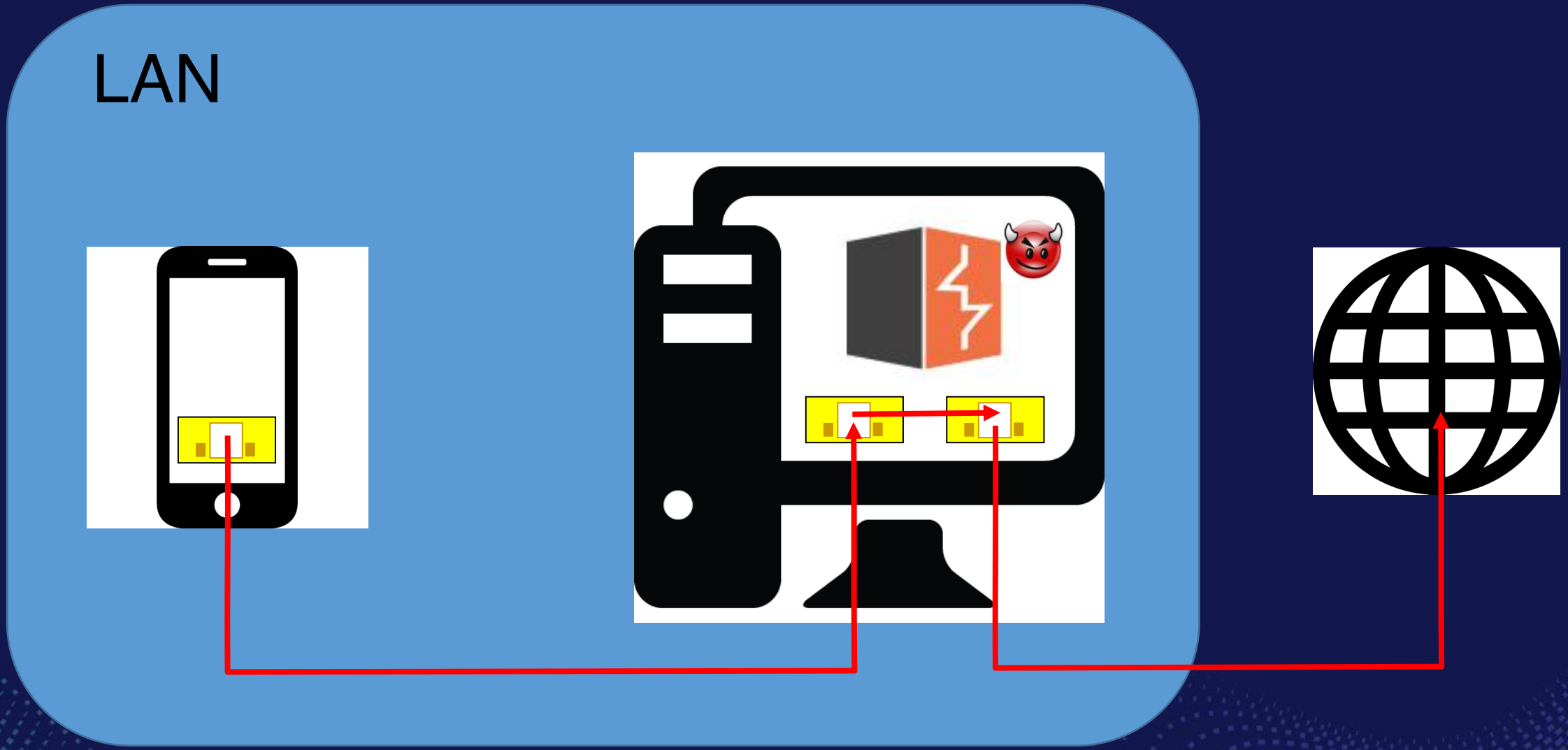
走进企业看安全

安卓APP通讯分析和抓包的几种常见模式

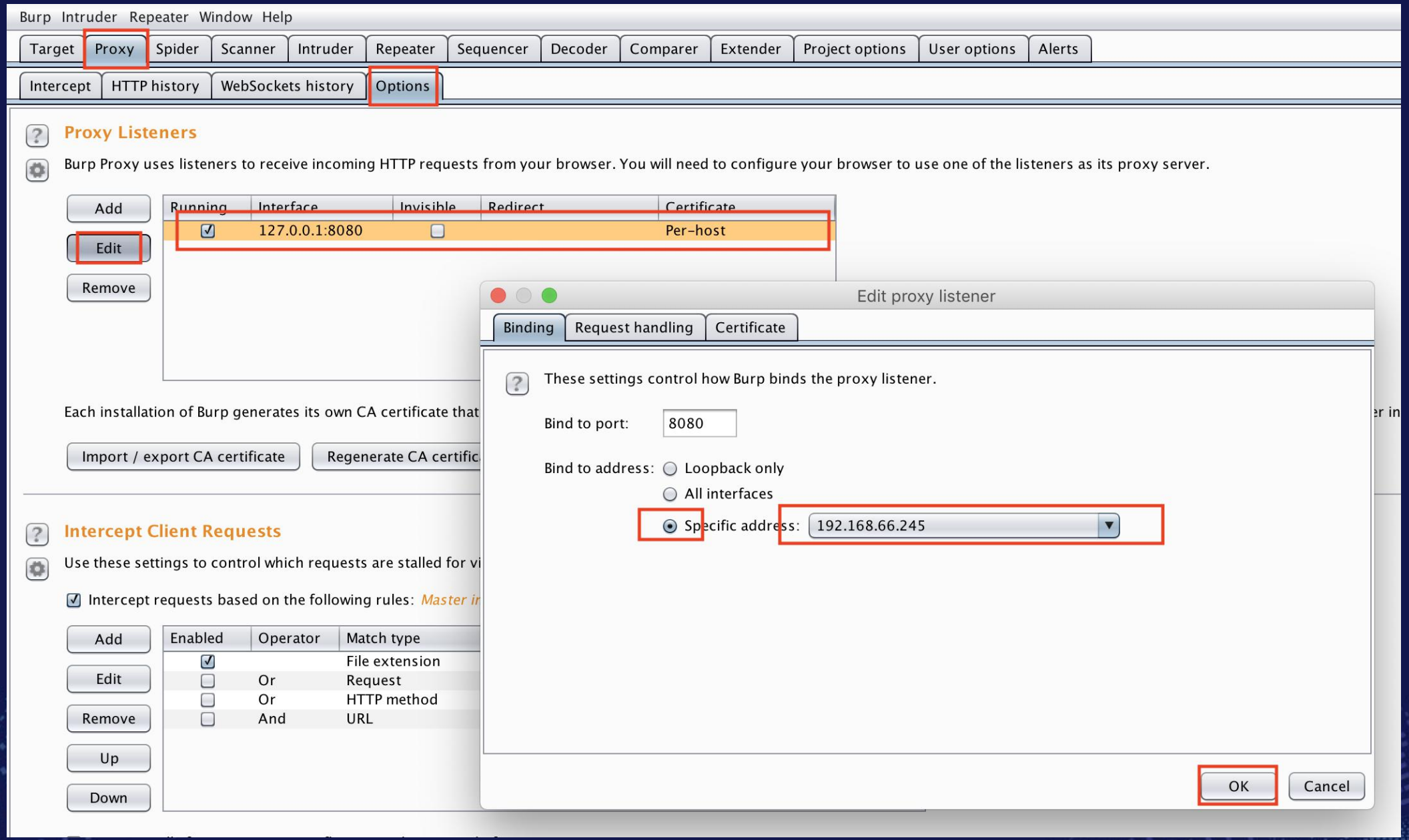
周信安　看雪科技

LAN


LAN

抓取HTTP包

1. 将电脑接入Wi-Fi，打开Burp Suite这一最经典的抓包工具。
2. 确保Burp监听内网端口。方法：点击Proxy->Options->Proxy Listeners下的第一个代理->Edit->点击Specific address前的圆点，并从复选框里选择内网地址（如192.168.66.245）->OK。

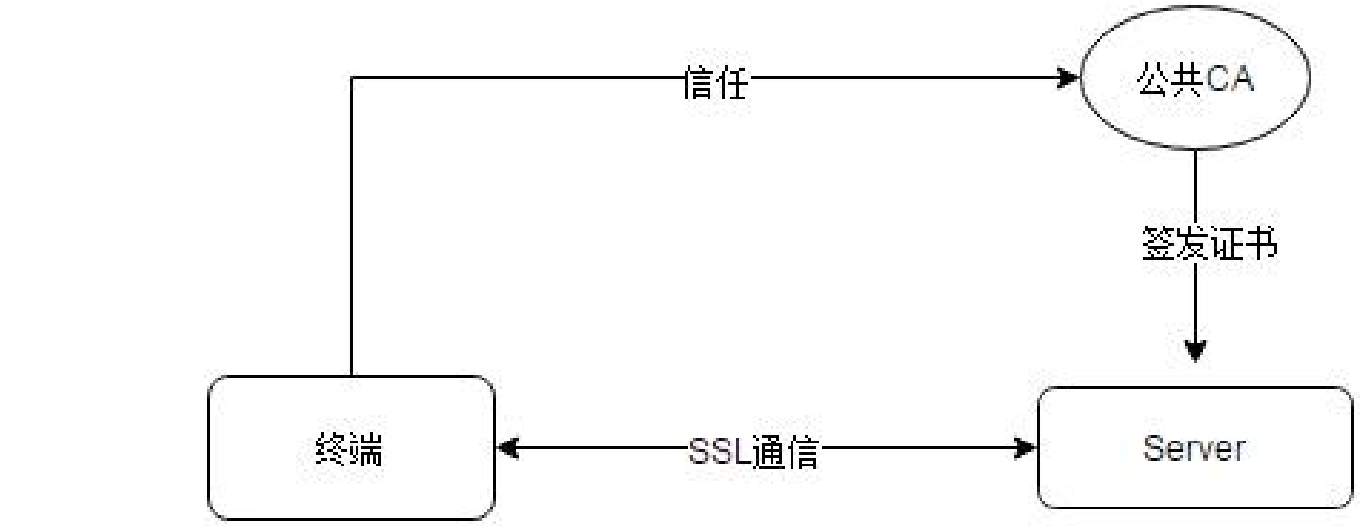

Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options Alerts
Intercept HTTP history WebSockets history Options
Proxy Listeners
Burp Proxy uses listeners to receive incoming HTTP requests from your browser. You will need to configure your browser to use one of the listeners as its proxy server.
Add
Edit
Remove
Running Interface Invisible Redirect Certificate
127.0.0.1:8080 Per-host
Each installation of Burp generates its own CA certificate that
Import / export CA certificate
Regenerate CA certific
Intercept Client Requests
Use these settings to control which requests are stalled for vi
Intercept requests based on the following rules: Master ir
Add
Edit
Remove
Up
Down
Enabled Operator Match type
File extension
Or Request
Or HTTP method
And URL
Edit proxy listener
Binding Request handling Certificate
These settings control how Burp binds the proxy listener.
Bind to port: 8080
Bind to address: Loopback only
All interfaces
Specific address: 192.168.66.245
OK
Cancel

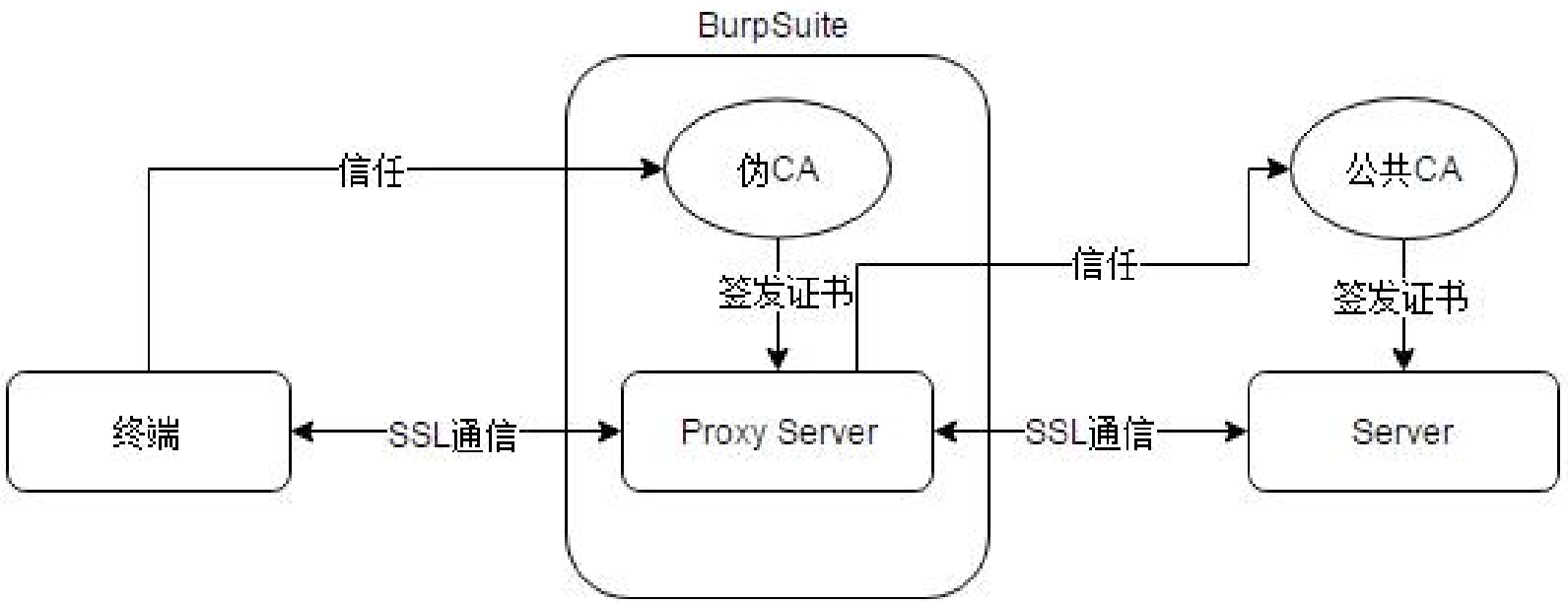
抓取HTTP包

3. 将安卓设备连接到同一个Wi-Fi中，并为安卓设备设置代理服务器地址。方法有两种：
    1. 在首次连接Wi-Fi的时候，在输入密码前点击高级选项，为代理选择手动，然后为代理服务器主机名和代理服务器端口填入Burp监听的内网地址和端口。根据经验，这种方法比方法2适用的安卓机型更多。
    2. 已经连上Wi-Fi后，长按Wi-Fi名称，点击修改网络，点击高级选项，为代理选择手动，然后为代理服务器主机名和代理服务器端口填入Burp监听的内网地址和端口。但是有部分机型会限制修改网络。

抓取HTTPS包

1. 由于系统安全措施升级，从Android 7开始更加困难。
2. 因为原理是SSL中间人攻击，所以需要在Android上信任Burp的CA证书。
3. 有部分应用部署了SSL Pinning。


信任
公共CA
签发证书
终端
SSL通信
Server


BurpSuite
信任
伪CA
签发证书
信任
公共CA
签发证书
终端
SSL通信
Proxy Server
SSL通信
Server

Android 7以下系统抓取HTTPS包

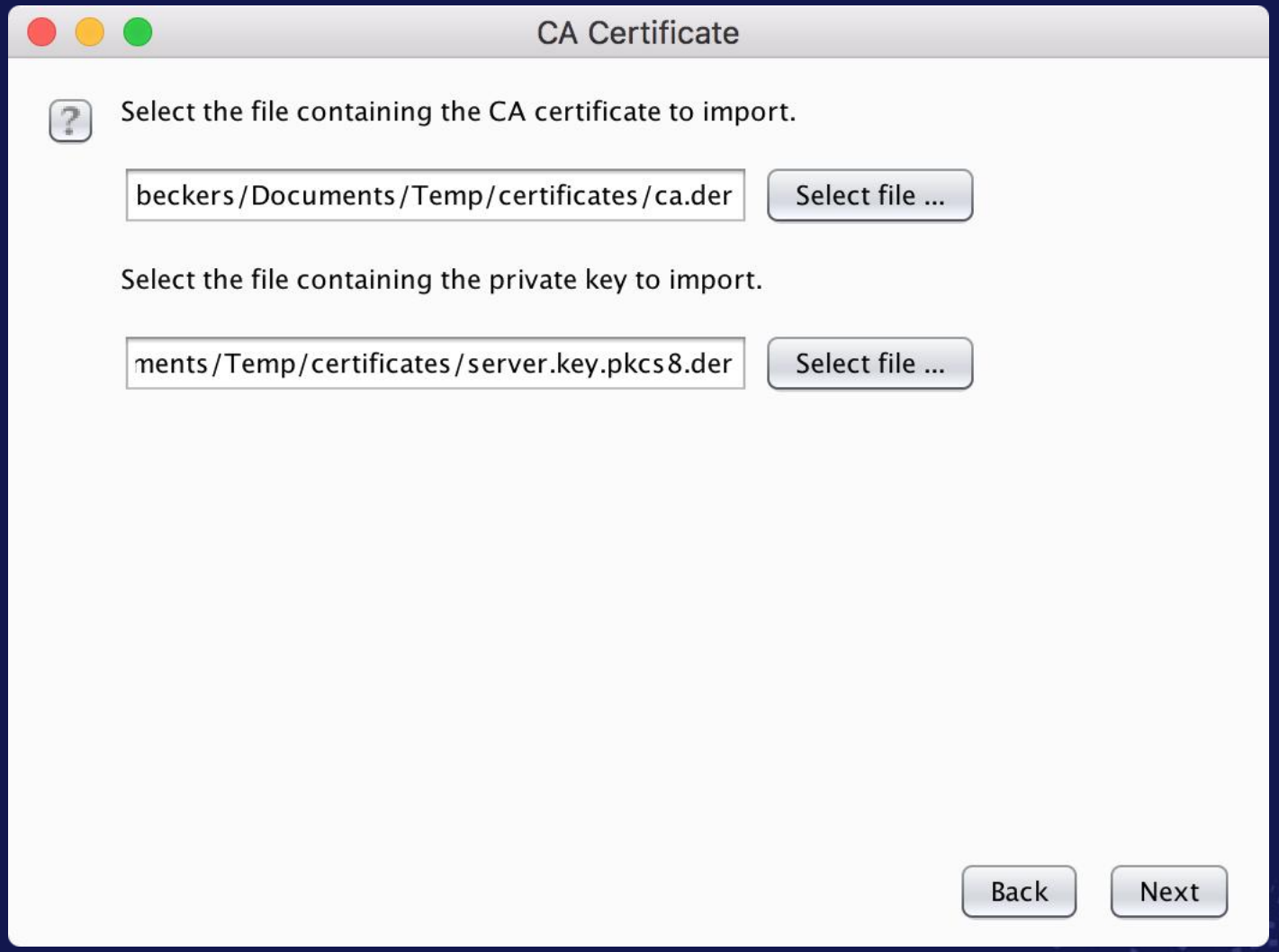
1. 完成“抓取HTTP包”中的所有步骤。
2. 在安卓中打开Chrome浏览器，并输入http://burp，点击网页右上角的CA Certificate，浏览器将下载Burp的CA证书并保存到系统的“下载”目录。
3. 使用文件浏览器（如ES文件浏览器）进入“下载”目录，将cacert.der改名为cacert.cer。
4. 进入安卓的设置->安全->从存储设备安装->进入“下载”目录->选择cacert.cer->为证书名称输入Burp->确定->如果没有设置锁屏密码，那么设置一个PIN码->成功。
5. 如果一切都顺利的话，现在在安卓中打开Chrome浏览器，并访问https://bbs.pediy.com，你将在Burp中看到通信流量。

Android 7及以上系统抓取HTTPS包（使用Magisk插件）

1. 完成“抓取HTTP包”中的所有步骤。
2. 打开命令行，切换到一个合适的工作目录，运行mkdir certificates && cd certificates。
3. 安装openssl。Ubuntu系统可以运行sudo apt-get install openssl。macOS系统建议先安装brew，再运行brew install openssl。
4. Ubuntu系统运行cp /usr/lib/ssl/openssl.cnf ./。macOS系统运行cp /usr/local/etc/openssl/openssl.cnf ./。
5. 运行openssl req -x509 -days 730 -nodes -newkey rsa:2048 -outform der -keyout server.key -out ca.der -extensions v3_ca -config openssl.cnf。为Country Name输入CN，再回车两次，再输入Burp四次。

Android 7及以上系统抓取HTTPS包（使用Magisk插件）

6. 执行openssl rsa -in server.key -inform pem -out server.key.der -outform der。
7. 执行openssl pkcs8 -topk8 -in server.key.der -inform der -out server.key.pkcs8.der -outform der -nocrypt。
8. 确认你的certificates文件夹里存在ca.der和server.key.pkcs8.der这两个文件。
9. 打开Burp->Proxy->Options->Import / export CA certificate->Import->Certificate and private key in DER format->为第一个框选择ca.der，为第二个框选择server.key.pkcs8.der->Next->Close。


CA Certificate
Select the file containing the CA certificate to import.
beckers/Documents/Temp/certificates/ca.der
Select file ...
Select the file containing the private key to import.
ments/Temp/certificates/server.key.pkcs8.der
Select file ...
Back
Next

Android 7及以上系统抓取HTTPS包（使用Magisk插件）

10.重新启动Burp，并按照“抓取HTTP包”中所述步骤确保Burp监听内网端口。

11.在安卓中打开Chrome浏览器，并输入http://burp，点击网页右上角的CA Certificate，浏览器将下载Burp的CA证书并保存到系统的“下载”目录。

12.使用文件浏览器（如大名鼎鼎的ES文件浏览器）进入“下载”目录，将cacert.der改名为cacert.cer。

13.进入安卓的设置->安全性和位置信息->加密与凭据->从存储设备安装->进入“下载”目录->选择cacert.cer->为证书名称输入Burp->确定->如果没有设置锁屏密码，那么设置一个PIN码->成功。

Android 7及以上系统抓取HTTPS包（使用Magisk插件）

14.接下来我们需要通过一款Magisk插件来在系统启动时自动将Burp证书加入系统证书目录中。从https://github.com/NVISO-BE/MagiskTrustUserCerts/releases下载最新版本的AlwaysTrustUserCerts.zip，并在电脑上执行adb push ./AlwaysTrustUserCerts.zip /sdcard/，将其传送到安卓手机上。

15.安卓手机上进入Magisk Manager->点击左上角按钮->点击模块->点击+->选择AlwaysTrustUserCerts.zip->重启。

16.如果一切都顺利的话，现在在安卓中打开Chrome浏览器，并访问https://bbs.pediy.com，你将在Burp中看到通信流量。

使用Packet Capture在受限的环境下抓包（原理）

1. 有时候，由于种种限制，我们不能在手机上安装Magisk，而手机又正好是Android 7及以上系统。
2. Packet Capture利用Android提供的API设立一个虚拟的VPN服务。打开VPN后，流量就会流过这个VPN，此时Packet Capture进行中间人攻击，这样就能解密SSL流量。
3. 没有root过的安卓也可以。

使用Packet Capture在受限的环境下抓包（步骤）

1. 下载Packet Capture，安装到手机上。
2. 打开APP，点击Get Started->Continue->Install Certificate，按提示安装证书。
3. 来到主界面。右上角共有两个绿色的三角按钮，点击第二个按钮，点击允许->确定，将开始抓包。
4. 在安卓中打开Chrome浏览器，并访问https://bbs.pediy.com。页面载入完成后切换到Packet Capture应用，点击第一个条目，这时你将看到解密后的SSL报文。
5. 返回主界面。点击右上角的停止按钮，将停止抓包。
6. 如果你想针对一个应用抓包，点击第一个绿色的三角按钮即可。

使用Wireshark+tcpdump实时抓取非HTTP/HTTPS流量

有些安卓应用，如网游，会采用自定义协议与服务器通信。为了实时抓取这些流量，需要在安卓上启动tcpdump（需要root权限），并利用管道将流量传给电脑上运行的Wireshark。这样就能实时地在电脑上观察安卓的流量。

使用Wireshark+tcpdump实时抓取非HTTP/HTTPS流量（步骤）

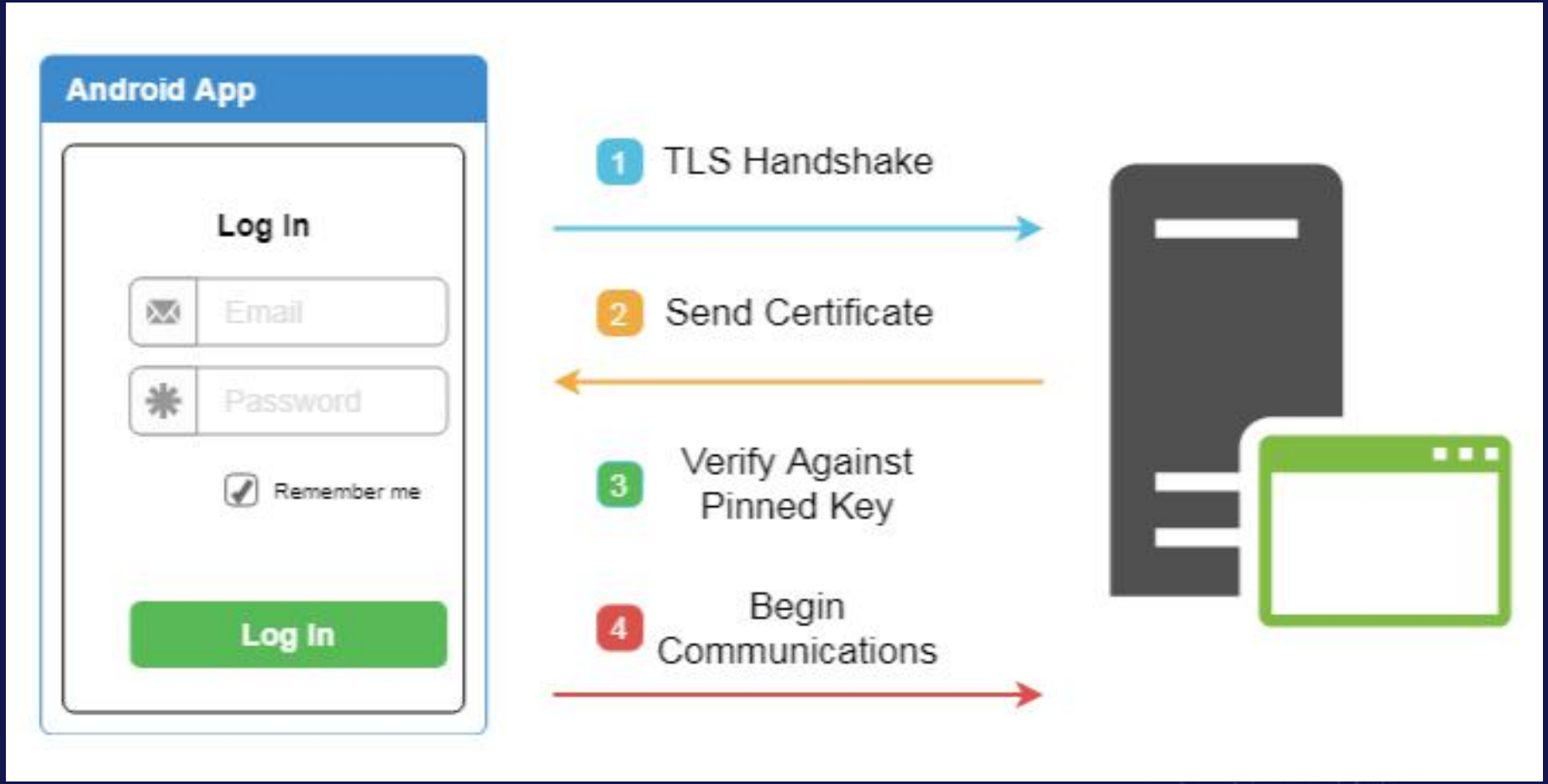
1. 下载Wireshark macOS版并安装。
2. 下载Android Studio并安装。确保在macOS的终端能运行adb。
3. 下载适用于Android的tcpdump，将手机连接到macOS，在macOS的终端运行adb push /Users/zhouxinan/Downloads/tcpdump /data/local/tmp，注意把tcpdump的路径修改成你自己的路径。

使用Wireshark+tcpdump实时抓取非HTTP/HTTPS流量（步骤）

4. 确保你的手机已经root，且能够以root的身份执行命令。步骤：adb shell->su->如果看见#，则进入下一步。
5. 以root身份执行chmod 777 /data/local/tmp/tcpdump，然后执行/data/local/tmp/tcpdump，如果看到提示文字，则进入下一步。
6. 退出adb。在macOS的终端执行adb shell "su -c /data/local/tmp/tcpdump -i any -U -w - 2>/dev/null" | wireshark -k -S -i -，如果一切都顺利的话，现在将会弹出Wireshark并显示Android上的所有流量。

使用Frida绕过SSL Pinning

1. 什么是SSL Pinning?
2. Frida分为运行在安卓上的frida-server和运行在电脑上的frida程序两部分，可以动态注入代码到安卓。
3. 所以我们可以动态改变APP的行为，让它信任我们的证书。
4. 需要root权限，但不需要安装Magisk。


Android App
Log In
Email
Password
Remember me
Log In
1 TLS Handshake
2 Send Certificate
3 Verify Against Pinned Key
4 Begin Communications

使用Frida绕过SSL Pinning（步骤）

1. 配置adb能在命令行运行。
2. 在电脑上安装python3与Frida。
3. 下载frida-server，上传到安卓的/data/local/tmp目录，并赋予+x的执行权限。
4. 让Burp监听内网端口。
5. 让Burp导出crt证书。方法：点击Proxy->Options->Import / export CA certificate->Certificate in DER format->选择一个地方保存为burp.crt->Next->Close。复制burp.crt的完整路径，接下来命令行运行adb push <burp.crt的完整路径> /data/local/tmp/cert-der.crt，完成。

使用Frida绕过SSL Pinning（步骤）

6. 在平板上关闭SELinux（平板每次关机或重启，都要重复一次这个操作）
7. 让frida-server启动并保持在后台运行（务必用root用户启动，不加任何参数，否则有bug）

```
> adb shell
$ su
$ /data/local/tmp/frida-server
```

使用Frida绕过SSL Pinning（步骤）

8. 另开一个终端运行 frida -U -l unpinning.js -f com.zhouxinan.test --no-pause

unpinning.js为frida脚本、com.zhouxinan.test为目标APP包名。 现在请在Burp中观察流量。

脚本地址:
https://codeshare.frida.re/@pcipolloni/universal-android-ssl-pinning-bypass-with-frida/

APP通讯分析

1. 抓包使得我们可以看见应用层的流量。（即HTTP报文）
2. 但是很多APP与API之间的通讯是加密的。甚至还有签名保护。
3. 所以我们需要进行逆向，还原加密、签名所用到的算法和密钥。

Request

Raw | Params | Headers | Hex | Decrypted

```
POST /assistant-pad/api/parentGroup/verifyToken/getBindAccountList HTTP/1.1
requestType: ENCODE
apkVersionCode: 2090100
apkPackageName: [redacted]
timestamp: [redacted]
machineId: [redacted]
deviceOSVersion: 2074855
padDeviceModel: [redacted]
accountId: [redacted]
Accept-Charset: utf-8
deviceModel: Nexus 5
appVer: 2.9.1.0
padMachineId: [redacted]
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
User-Agent: Dalvik/2.1.0 (Linux; U; Android 5.1.1; Nexus 5 Build/LMY48I)
Host: [redacted]
Connection: close
Content-Length: 504

toy6xflEZfN17YVycfiNAx/GBjq5GyxA42ip4pLBCxhg4ZQdqYBK1f9SciybYZXoLZIO1Rx5BAmRX
5tkESJtR7vEJTHMog9xFQFlgZZU2m+CkaBhy3I7pcgatoJg2eymCYV9blCz13DCBL+lxve8OF4Ku
vaBrgmQUS6HfcIT3kIyXOZfLxtXeeY/NBrOt4ZhqwTYCfxiMkV/iHan2Rdc628tmANPDDeghsLzW
GVdXoXCBRhKB7rsvraOR+z6BQbM9s/9JbV2Xdz3/lhLOVc9iL7Z1swpxD9Qqjobm71ds+eNf37Jd
2YkoWRbBayDMGX03YGqvB024suVOkiWaIisCWWN/kqHn+uu+sunAToNDMIuugkSf/dIAJhU3Cl8Y
Qae7Cx2/z2MoNGQiG/iWOItLkEIPzQXRolpiduGVyIiAXoZzMOlzUTGPqteVGV/8SmyDDuDx8Iz+
7zu43lHuGVN69L9t2pZY0+v0Yh0x5CGf5EIK/ZqOQUnIA==
```

APP通讯分析（步骤）

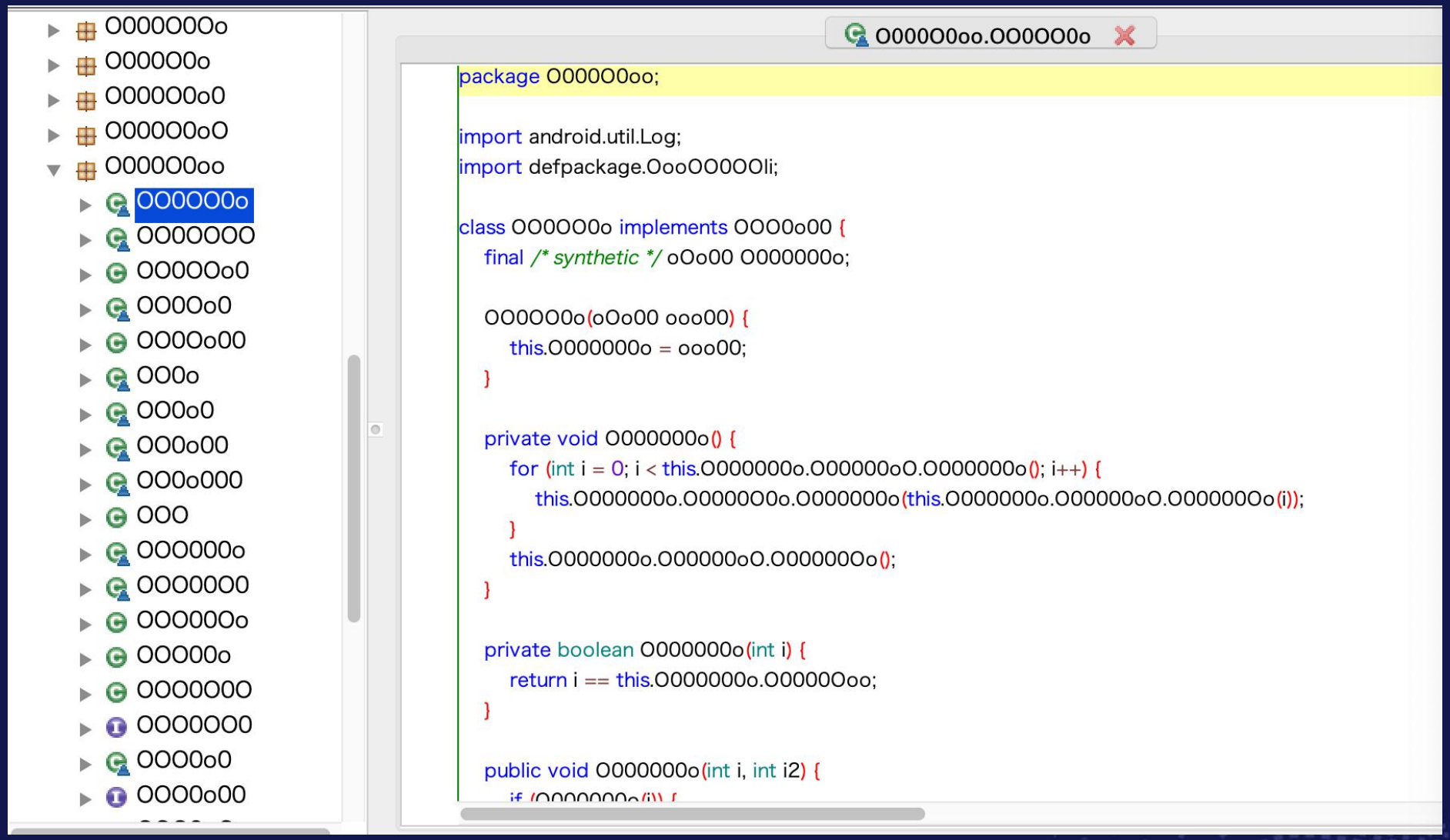
1. 如果有壳的话，先脱壳。
2. 搜索感兴趣的字符串，如API名字。
3. 阅读反编译出来的源码，尝试移植到IDE中。
4. 尝试用移植出来的代码解密通讯协议的报文。有可能还需要加密回去并签名。

一例通讯协议

1. 首先将参数列表按照参数名排序，然后用##连接，形成a=1##b=2这样的参数字符串。
2. 将API的完整URL、参数字符串、“盐”值（固定字符串）连接起来计算MD5值。
3. 将MD5值附加到参数字符串前，用&连接，并将参数字符串中的##替换为&，最终形成如sign=xxxxx&a=1&b=2这样的参数字符串。
4. 将参数字符串用固定密钥和固定算法进行加密。

展望

1. APP加壳、加固
2. 混淆
3. 自定义加解密、签名算法
4. 反调试
5. root检测


OOOOOOo
OOOOOo
OOOOOo0
OOOOOoO
OOOOOoo
OOOOOo
OOOOOOO
OOOOOo0
OOOOo0
OOOOo00
OOOo
OOOo0
OOOo00
OOOo000
OOO
OOOOOOo
OOOOOO0
OOOOOOo
OOOOOo
OOOOOOO
OOOOOO0
OOOOo0
OOOOo00
OOOOOOo.OOOOOOo
package OOOOOOoo;
import android.util.Log;
import defpackage.OooOOOOOli;
class OOOOOOo implements OOOOo00 {
final /* synthetic */ oOo00 O0000000o;
OOOOOOo(oOo00 ooo00) {
this.O0000000o = ooo00;
}
private void O0000000o() {
for (int i = 0; i < this.O0000000o.OOOOOOoO.O0000000o(); i++) {
this.O0000000o.O0000OOo.O0000000o(this.O0000000o.OOOOOOoO.OOOOOOOo(i));
}
this.O0000000o.OOOOOOoO.OOOOOOOo();
}
private boolean O0000000o(int i) {
return i == this.O0000000o.OOOOOOoo;
}
public void O0000000o(int i, int i2) {

参考文献

1. https://github.com/NVISO-BE/MagiskTrustUserCerts
2. https://blog.nviso.be/2017/12/22/intercepting-https-traffic-from-apps-on-android-7-using-magisk-burp/
3. https://blog.nviso.be/2018/01/31/using-a-custom-root-ca-with-burp-for-inspecting-android-n-traffic/
4. https://blog.wirelessmoves.com/2017/02/adb-and-tcpdump-on-android-for-live-wireshark-tracing.html
5. https://techblog.mediaservice.net/2017/07/universal-android-ssl-pinning-bypass-with-frida/