# 背景

上次测试一个app的时候发现无法抓包利用xp框架解决后提示不是可信证书，无法进行下一步测试。

经过几个小时令人沮丧的故障排除后，我终于发现这个问题与Android的最新版本有关(API>=24)。在我更进一步之前，我所需要的所有信息都是在这些伟大的文章中找到的：

https://serializethoughts.com/2016/09/10/905/

https://android-developers.googleblog.com/2016/07/changes-to-trusted-certificate.html

从Nugat开始，Android改变了信任用户安装证书的默认行为。不再可能仅仅从sdCard安装BurpCA来开始拦截应用程序流量。除非另有规定，应用程序现在将只信任系统级CA。故障发生在“无形中”，并对我在BurpSuite中看到的所有警报负责。

有两种方法可以绕过这两条路，我会把它们都看一遍。

将Burp CA安装为设备上的系统级CA。...我推荐的最简单的解决方案，但确实需要一个根设备。还增加了不必设置锁定屏幕PIN的好处：)

修改清单并重新打包应用程序...工作稍微多一点，但不需要根特权。

注意：我在Windows 10机器上使用Burp Suite Pro完成了所有这些工作，并且使用的是Android7.1(API 25)GenymotionVM，但是这些步骤应该适用于任何设置。

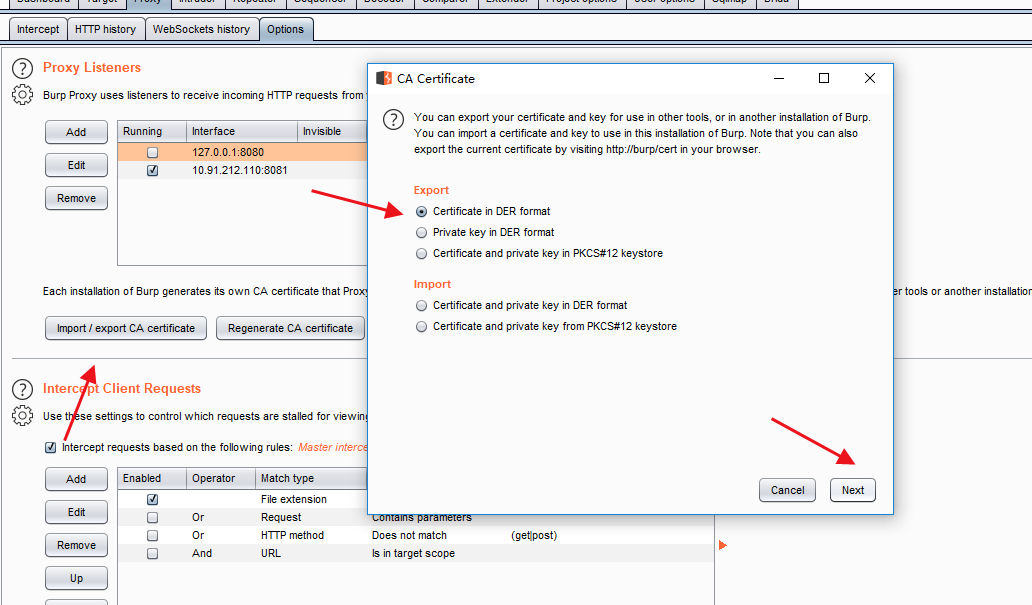
# 将Burp CA安装为系统级可信CA

由于安装用户证书的“传统”方法不再适用于Nugat和以上版本，因此对我来说，最简单的解决方案是将BurpCA安装到系统受信任的证书上。您可以看到与android设备捆绑在一起的所有系统ca，方法是设置->安全性->可信凭据以及查看系统CA。您将在浏览器包中看到类似的CA。

用于android的可信ca以一种特殊格式存储在/system/etc/security/cacerts...如果我们拥有根权限，则可以写入此位置并放入BurpCA(经过一些修改后)。

## 导出和转换Burp CA

第一步是以正确的格式获得BurpCA。使用Burp套件，以der格式导出CA证书。我把它保存下来cacert.der



Android希望证书采用PEM格式，并且文件名与subject\_hash\_old附加值.0.

使用openssl若要将der转换为PEM，则输出subject\_hash\_old并重命名该文件：

openssl x509 -inform DER -in cacert.der -out cacert.pem 改格式

openssl x509 -inform PEM -subject\_hash\_old -in cacert.pem |head -1 查看cacert.pem 的hash值

mv cacert.pem <hash>.0 重命名

## 将证书复制到设备

我们可以用adb若要复制证书，但必须将其复制到/system文件系统，我们必须将其重新装入为可写文件系统。作为根，这很容易用adb remount.

adb root

adb remount

adb push <cert>.0 /sdcard/

只是掉进一个壳里(adb shell)并将文件移动到/system/etc/security/cacerts并将其发送到644：

mv /sdcard/<cert>.0 /system/etc/security/cacerts/

chmod 644 /system/etc/security/cacerts/<cert>.0

最后，我们必须用以下两种方法来重新启动设备adb reboot或者关机重启。

设备重新启动后，浏览到设置->安全性->可信凭据应该将新的“Portswigger CA”显示为系统信任CA。

现在可以设置代理并开始用burp：)来记录任何和所有的应用程序流量。)

# 移动和重新打包应用程序

注：此方法是笔者未经实验，只是一种思路，未必可以

如果您没有根或者不想修改系统受信任的证书，可以将BurpCA安装为用户证书，然后修改要MITM的特定APK。

从Nougat开始，应用程序默认会忽略用户安装的证书。从以下几个方面可以看出这一点：logcat启动应用程序时输出：

No network security config

如果没有网络安全配置，应用程序将只信任系统CA并将不会遵守用户已安装的Burp证书。

为了解决这个问题，它包括：

拆卸APK

添加新的xml资源来定义网络安全配置文件

修改AndroidManifest.xml

重新打包和自签名apk

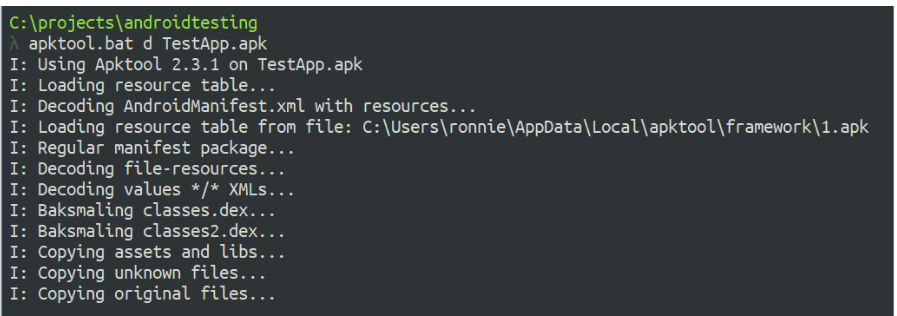
拆卸和修改APK

从使用开始APKTool拆卸APK

## 拆卸和修改APK

从使用开始APKTool拆卸APK

apktool d TestApp.apk



接下来，通过创建文件添加一个新的网络安全配置。network\_security\_config.xml在res/xml目录：

vim TestApp\res\xml\network\_security\_config.xml

配置需要明确声明信任用户证书是可以接受的。全部内容应是：

<network-security-config>

<base-config>

<trust-anchors>

<!-- Trust preinstalled CAs -->

<certificates src="system" />

<!-- Additionally trust user added CAs -->

<certificates src="user" />

</trust-anchors>

</base-config>

</network-security-config>

最后，我们必须在AndroidManifest.xml...在<application>标记，添加android:networkSecurityConfig属性指向新的XML文件：

## 重新组装和标志

最后，现在必须重建和签署APK才能安装。使用apktool b，则将在dist/目录：

<application android:allowBackup="true"

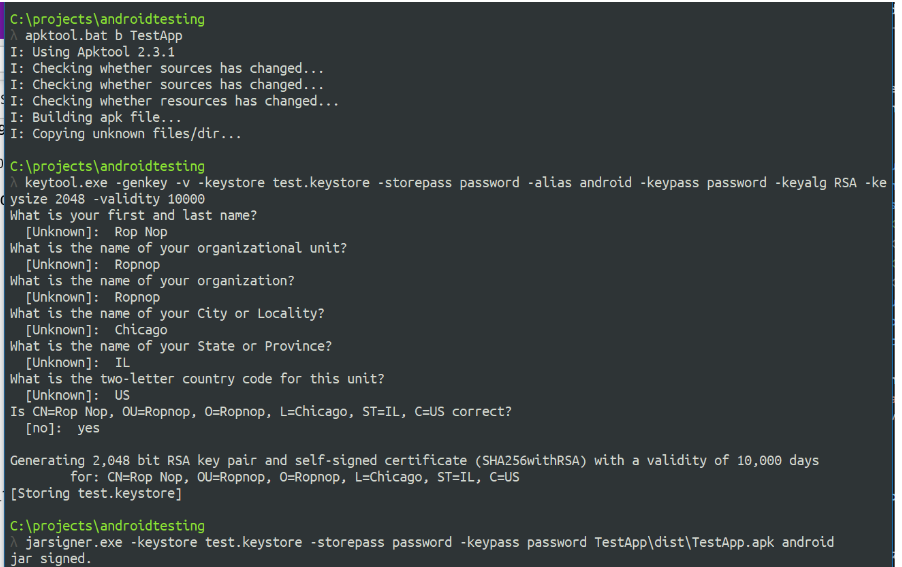
android:networkSecurityConfig="@xml/network\_security\_config" ...etc...>

apktool b TestApp

若要自我签署应用程序，请使用keytool若要创建新的密钥存储库和密钥，请执行以下操作jarsigner签署新的APK：

keytool -genkey -v -keystore test.keystore -storepass password -alias android -keypass password -keyalg RSA -keysize 2048 -validity 10000

jarsigner.exe -verbose -keystore test.keystore -storepass password -keypass password TestApp\dist\TestApp.apk android



最后，使用adb:

adb install TestApp\dist\TestApp.apk

现在，当我们启动应用程序时，logcat输出将指示正在使用一个新的网络安全配置：



随着BurpCA安装到用户证书，我们现在可以MITM应用程序流量！

顺带说下之前是怎么解决抓包问题的安装xp框架后在安装JustTrustMe和SSLKILLER模块。

JustTrustMe是Github上的一个开源工程，他是一个Xposed模块，用来禁止SSL证书验证

# 参考链接：

1. <https://blog.ropnop.com/configuring-burp-suite-with-android-nougat/>

# 附录

1.burp证书



1. xp框架



1. 模块

