Configuring Burp Suite With Android Nougat

 Posted on January 18, 2018 | ropnop

(//twitter.com/share?url=https%3a%2f%2fblog.ropnop.com%2fconfiguring-burp-suite-with-android-nougat%2f&text=Configuring%20Burp%20Suite%20With%20Android%20Nougat&via=ropnop)

(//reddit.com/submit?url=https%3a%2f%2fblog.ropnop.com%2fconfiguring-burp-suite-with-android-nougat%2f&title=Configuring%20Burp%20Suite%20With%20Android%20Nougat)

(https://news.ycombinator.com/submitlink?u=https%3a%2f%2fblog.ropnop.com%2fconfiguring-burp-suite-with-android-nougat%2f&t=Configuring%20Burp%20Suite%20With%20Android%20Nougat)

(//www.linkedin.com/shareArticle?url=https%3a%2f%2fblog.ropnop.com%2fconfiguring-burp-suite-with-android-nougat%2f&title=Configuring%20Burp%20Suite%20With%20Android%20Nougat)

Table of Contents

- Background
- Install Burp CA as a system-level trusted CA
- Mofiying and repackaging an app
- tl;dr cheatsheet

This last weekend I started testing a new Android app for fun, and ran into some trouble getting Burp Suite working properly. I burned a whole afternoon troubleshooting the issue, and decided to write up what I found out and two different ways I got it working.

Background

I've done quite a bit of Android testing in the past and my setup usually involves a Genymotion (https://www.genymotion.com/) VM or my old rooted Nexus Tablet. I run Burp Suite locally, install the User Cert as outlined in Portswigger's documentation (https://support.portswigger.net/customer/portal/articles/1841102-installing-burp-s-ca-certificate-in-an-android-device), configure a WiFi proxy and I'm off the races.

This particular app I wanted to test, however, required a minimum API level 24 (Android 7.0 - "Nougat") and suddenly it wasn't working. I followed the steps I always do but saw nothing but "connection reset" errors in Burp:
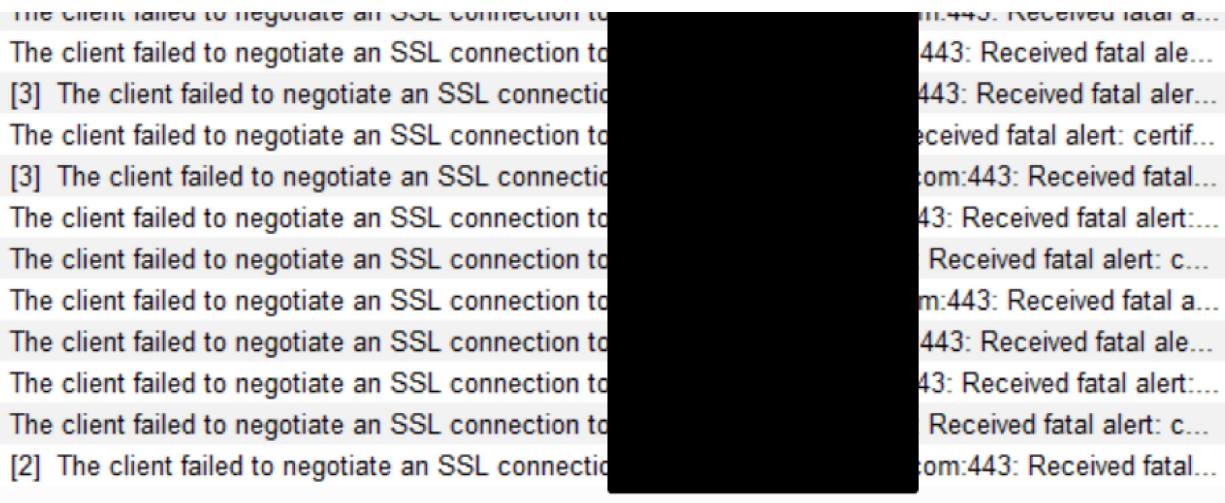
(/images/2018/01/burp_alerts.png)

After a few frustrating hours of troubleshooting, I finally figured out the issue lied with the latest versions of Android (API >= 24). Before I go any further, all the information I needed was found in these great write-ups:

- https://serializethoughts.com/2016/09/10/905/ (https://serializethoughts.com/2016/09/10/905/)
- https://android-developers.googleblog.com/2016/07/changes-to-trusted-certificate.html (https://android-developers.googleblog.com/2016/07/changes-to-trusted-certificate.html)

Starting with Nougat, Android changed the default behavior of trusting user installed certificates. It's no longer possible to just install the Burp CA from the sdcard to start intercepting app traffic. Unless otherwise specified, apps will now *only* trust system level CAs. The failure happens "invisibly" and is responsible for all the alerts I saw in Burp Suite.

There's two ways to bypass this, and I'll walk through them both.

- **Install the Burp CA as a system-level CA on the device**. My recommendation for the easiest solution, but does require a rooted device. Also added benefit of not having to set a lockscreen PIN :)
- **Modify the manifest and repackage the app**. Slightly more work, but doesn't require root privileges.

Note: I did all this with Burp Suite Pro on my Windows 10 machine and am using an Android 7.1 (API25) Genymotion VM, but the steps should be applicable to any setup.

Install Burp CA as a system-level trusted CA

Since the "traditional" way of installing a user certificate doesn't work anymore in Nougat and above, for me the easiest solution is to install the Burp CA to the system trusted certificates. You can see all the system CAs that are bundled with an Android device by going to *Settings -> Security -> Trusted Credentials* and viewing system CAs. You'll see the similar CAs you'd see in a browser bundle.

Trusted CAs for Android are stored in a special format in `/system/etc/security/cacerts`. If we have root privileges, it's possible to write to this location and drop in the Burp CA (after some modification).

Export and convert the Burp CA The first step is to get the Burp CA in the right format. Using Burp Suite, export the CA Certificate in DER format. I saved it as `cacert.der`
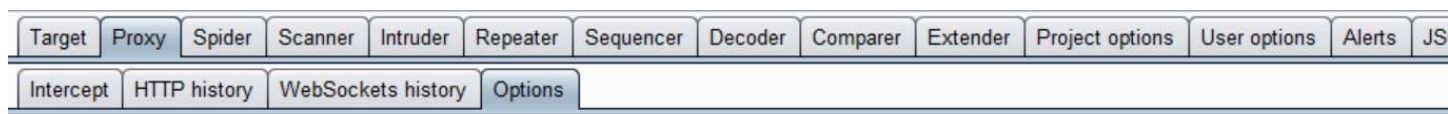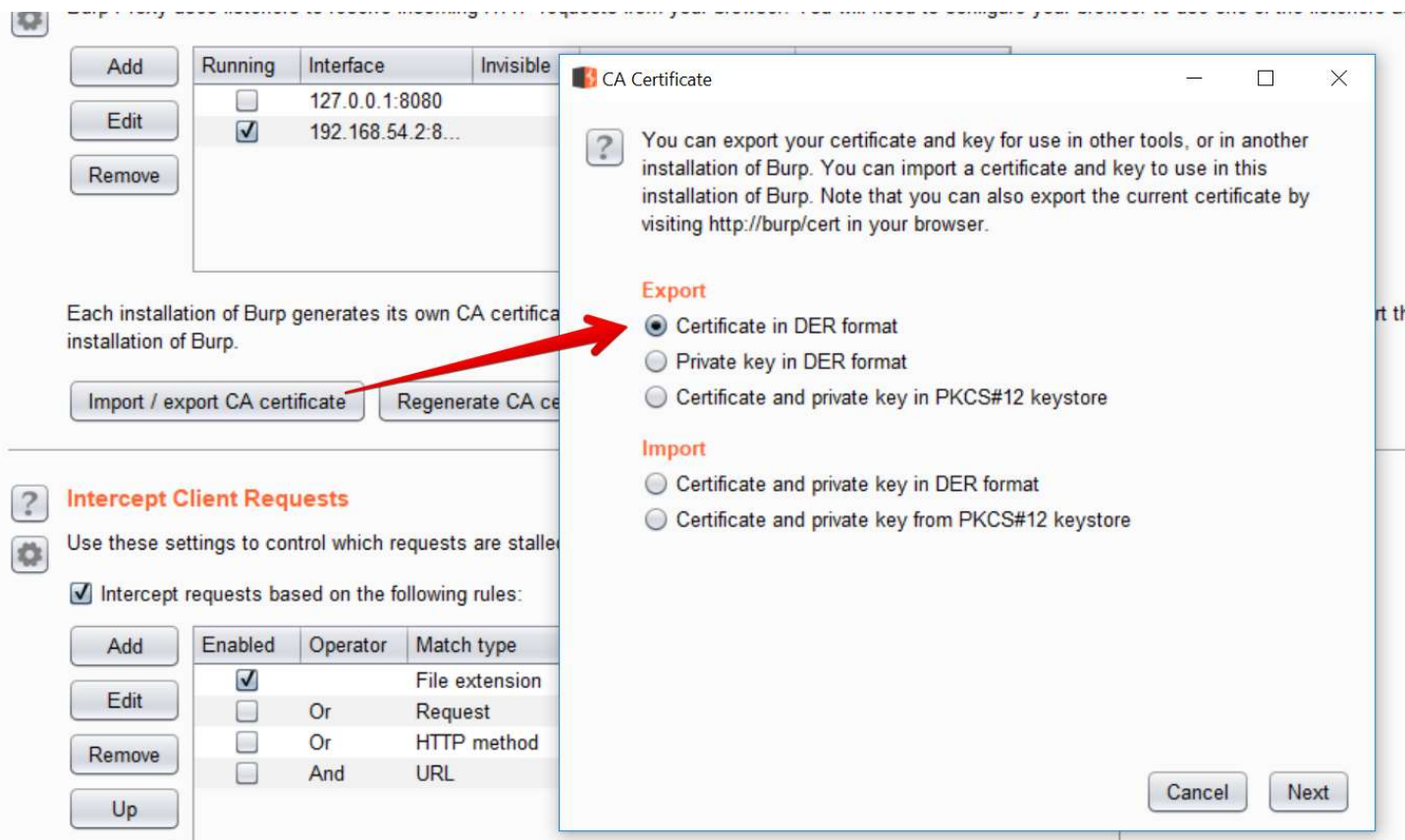
(/images/2018/01/export_burp_ca.png)

Android wants the certificate to be in PEM format, and to have the filename equal to the `subject_hash_old` value appended with `.0`.

Note: *if you are using OpenSSL <1.0, it's actually just the* `subject_hash` *, not the "old" one*

Use `openssl` to convert DER to PEM, then output the `subject_hash_old` and rename the file:

```
1    openssl x509 -inform DER -in cacert.der -out cacert.pem
2    openssl x509 -inform PEM -subject_hash_old -in cacert.pem |head -1
3    mv cacert.pem <hash>.0
```

For example, with my certificate:

(/images/2018/01/openssl_convert.png)

Copy the certificate to the device We can use `adb` to copy the certificate over, but since it has to be copied to the `/system` filesystem, we have to remount it as writable. As root, this is easy with `adb remount`.

```
1    adb root
2    adb remount
3    adb push <cert>.0 /sdcard/
```

The just drop into a shell (`adb shell`) and move the file to `/system/etc/security/cacerts` and chmod it to 644:

```
1    mv /sdcard/<cert>.0 /system/etc/security/cacerts/
2    chmod 644 /system/etc/security/cacerts/<cert>.0
```

Lastly, we have to full reboot the device with either `adb reboot` or a power cycle.

```
C:\projects\androidtesting
λ adb root

C:\projects\androidtesting
λ adb remount
remount succeeded

C:\projects\androidtesting
λ adb push 9a5ba575.0 /sdcard/
9a5ba575.0: 1 file pushed. 0.2 MB/s (1375 bytes in 0.005s)

C:\projects\androidtesting
λ adb shell
vbox86p:/ # mv /sdcard/9a5ba575.0 /system/etc/security/cacerts/
vbox86p:/ # chmod 644 /system/etc/security/cacerts/9a5ba575.0
vbox86p:/ # reboot
```

(/images/2018/01/copy_cert.png)

After the device reboots, browsing to *Settings -> Security -> Trusted Credentials* should show the new "Portswigger CA" as a system trusted CA.
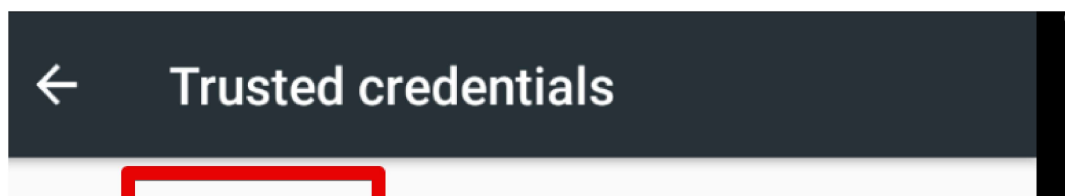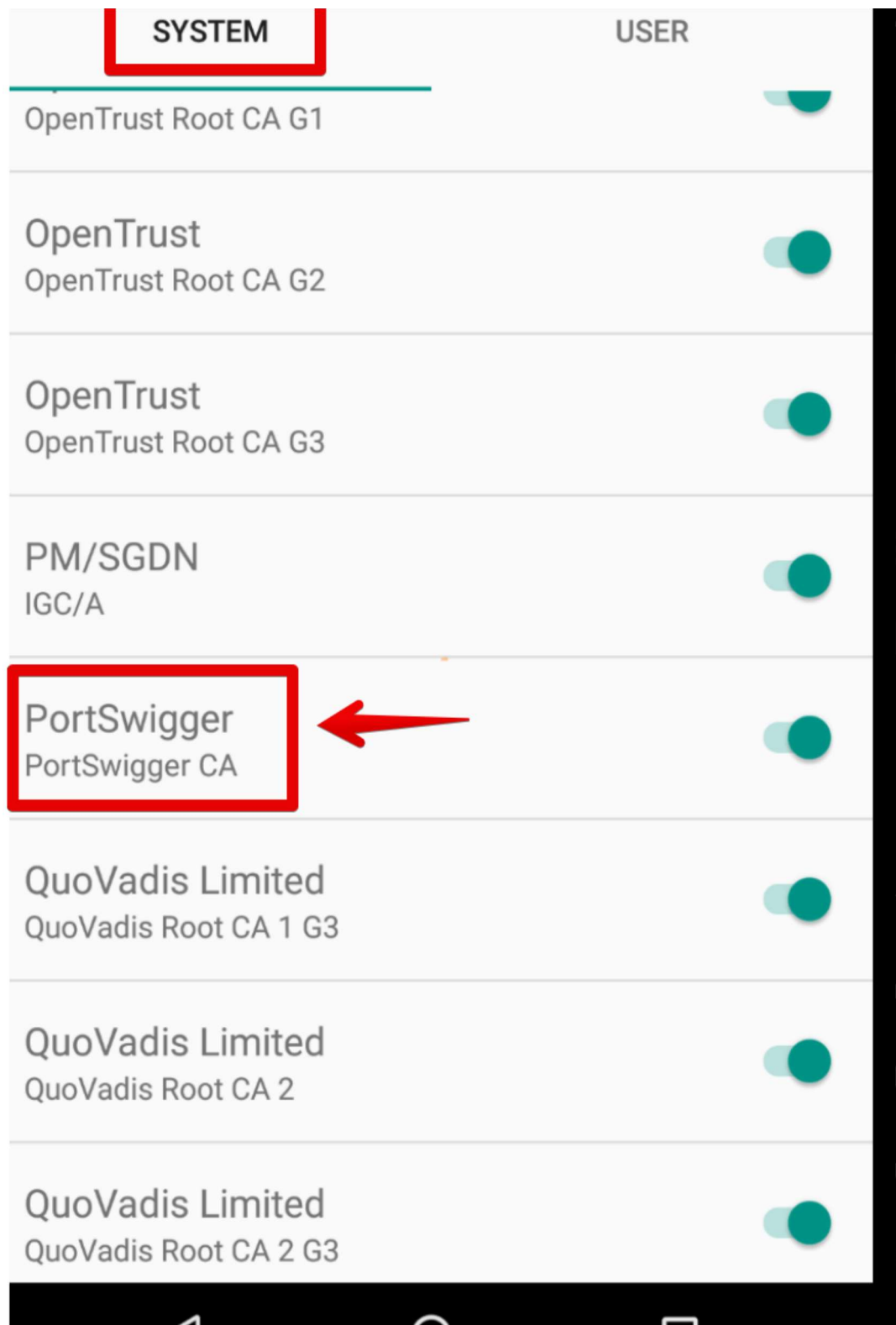
(/images/2018/01/portswigger_system_ca.png)

Now it's possible to set up the proxy and start intecepting any and all app traffic with Burp :)

Mofiying and repackaging an app

If you don't have root or don't want to modify the system trusted certificates, you can install the Burp CA as a user cert and then modify the specific APK you want to MitM.

Starting with Nougat, apps will ignore user-installed certificates by default. This is evident by looking at `logcat` output when launching the app:
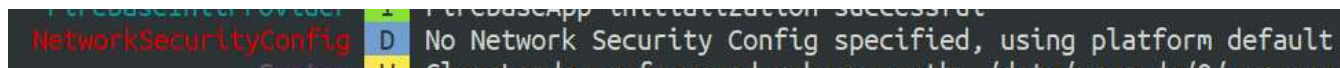
(/images/2018/01/no_network_config.png)

Without a network security config, the app will *only* trust system CAs and will not honor the user installed Burp certificate.

To get around this, it involves:

- Disassembling the APK
- Adding a new XML resource to define a network security profile
- Modifying AndroidManifest.xml
- Repackaging and self-signing the APK

Disassemble and modify the APK Start by using apktool (https://ibotpeaches.github.io/Apktool/) to disassemble the APK

```
1   apktool d TestApp.apk
```


(/images/2018/01/apktool_d.png)

Next, add a new network security config by creating the file `network_security_config.xml` in the `res/xml` directory:

```
1   vim TestApp\res\xml\network_security_config.xml
```

The config needs to explicitly state that trusting user certs is acceptable. The entire contents should be:

```
1    <network-security-config>
2        <base-config>
3            <trust-anchors>
4                <!-- Trust preinstalled CAs -->
5                <certificates src="system" />
6                <!-- Additionally trust user added CAs -->
7                <certificates src="user" />
8            </trust-anchors>
9        </base-config>
10   </network-security-config>
```

Finally, we have to define the network security config in `AndroidManifest.xml`. In the `<application>` tag, add the `android:networkSecurityConfig` attribute pointing to the new XML file:

```
1    <application android:allowBackup="true" android:networkSecurityConfig="@xml/network_security_config" ...etc...>
```

Reassemble and Sign Finally, the APK must now be rebuilt and signed in order to be installed. Using `apktool b`, a new build will be created in the `dist/` directory:
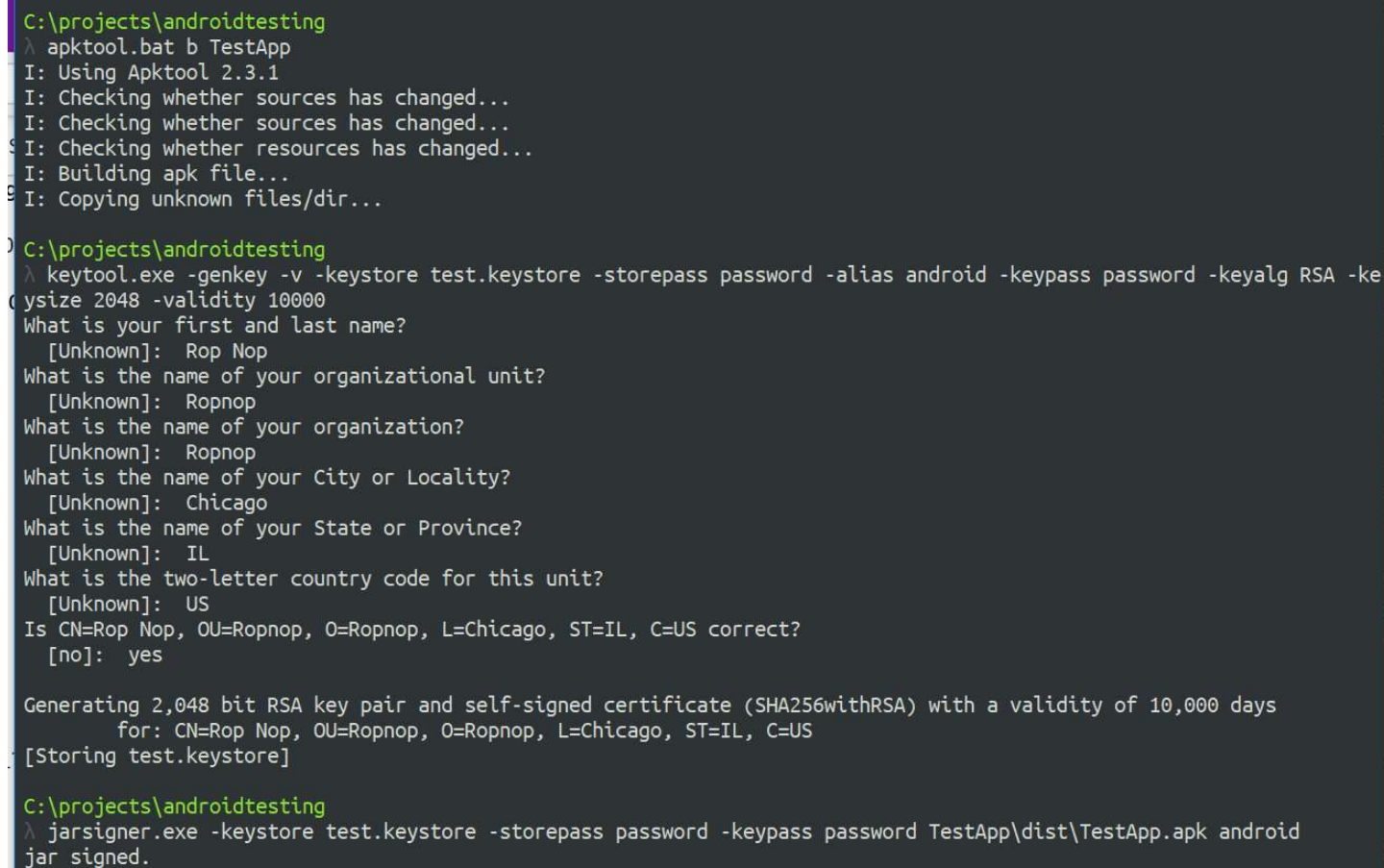
```
1   apktool b TestApp
```

To self-sign the app, use `keytool` to create a new keystore and key, then `jarsigner` to sign the new APK:

```
1    keytool -genkey -v -keystore test.keystore -storepass password -alias android -keypass password -keyalg RSA -keysize 2048 -validity 10000
2
3
4    jarsigner.exe -verbose -keystore test.keystore -storepass password -keypass password TestApp\dist\TestApp.apk android
```


(/images/2018/01/build_and_sign.png)

Lastly, install the new APK with `adb`:

```
1    adb install TestApp\dist\TestApp.apk
```

Now, when we start the application, the logcat output will indicate a new network security config is being used:

(/images/2018/01/logcat_new_config.png)

With the Burp CA installed to user certificates, we can now MitM the application traffic!

tl;dr cheatsheet

Install System CA

```
1    # Convert DER to PEM
2    openssl x509 -inform DER -in cacert.der -out cacert.pem
3
4    # Get subject_hash_old (or subject_hash if OpenSSL < 1.0)
5    openssl x509 -inform PEM -subject_hash_old -in cacert.pem |head -1
6
7    # Rename cacert.pem to <hash>.0
```

```
 8    mv cacert.pem 9a5ba575.0
 9
10    # Remount and copy cert to device
11    adb root
12    adb remount
13    adb push 9a5ba575.0 /sdcard/
14    adb shell
15    vbox86p:/ # mv /sdcard/9a5ba575.0 /system/etc/security/cacerts/
16    vbox86p:/ # chmod 644 /system/etc/security/cacerts/9a5ba575.0
17    vbox86p:/ # reboot
```

Modify APK

```
 1    apktool d TestApp.apk
 2    vim TestApp\res\xml\network_security_config.xml
 3    #Content:
 4            <network-security-config>
 5            <base-config>
 6            <trust-anchors>
 7            <!-- Trust preinstalled CAs -->
 8            <certificates src="system" />
 9            <!-- Additionally trust user added CAs -->
10            <certificates src="user" />
11            </trust-anchors>
12            </base-config>
13            </network-security-config>
14
15    vim TestApp\AndroidManifest.xml
16    # Add to <application > tag:
17            android:networkSecurityConfig="@xml/network_security_config"
18
19    # Rebuild and self-sign
20    keytool -genkey -v -keystore test.keystore -storepass password -alias android -keypass password -keyalg RSA -keysize 2048 -validity 10000
21
22    apktool b TestApp
23
24    jarsigner -keystore test.keystore -storepass password -keypass password TestApp\dist\TestApp.apk android
25
26    # Install new APK
27    adb install TestApp\dist\TestApp.apk
28
29    # Install Burp CA to User Certs
30    mv cacert.der cacert.cer
31    adb push burpca.cer /mnt/sdcard
32    Settings -> Security -> Install from SD Card
```

Hope this helps! -ropnop

Tags: #android (https://blog.ropnop.com/tags/android/)   #burp (https://blog.ropnop.com/tags/burp/)   #nougat (https://blog.ropnop.com/tags/nougat/)

🐦
(//twitter.com/share?url=https%3a%2f%2fblog.ropnop.com%2fconfiguring-burp-suite-with-android-nougat%2f&text=Configuring%20Burp%20Suite%20With%20Android%20Nougat&via=ropnop)

🔴
(//reddit.com/submit?url=https%3a%2f%2fblog.ropnop.com%2fconfiguring-burp-suite-with-android-nougat%2f&title=Configuring%20Burp%20Suite%20With%20Android%20Nougat)

Y
(https://news.ycombinator.com/submitlink?u=https%3a%2f%2fblog.ropnop.com%2fconfiguring-burp-suite-with-android-nougat%2f&t=Configuring%20Burp%20Suite%20With%20Android%20Nougat)

in
(//www.linkedin.com/shareArticle?url=https%3a%2f%2fblog.ropnop.com%2fconfiguring-burp-suite-with-android-nougat%2f&title=Configuring%20Burp%20Suite%20With%20Android%20Nougat)

See also

- Proxying and Intercepting CLI Tools (/proxying-cli-tools/)
- Sans Holiday Hack 2016 - Writeup (/sans-holiday-hack-2016-writeup/)
- Installing Drozer on OS X El Capitan (/installing-drozer-on-os-x-el-capitan/)

← **PREVIOUS POST (HTTPS://BLOG.ROPNOP.COM/SANS-HOLIDAY-HACK-2017-WRITEUP/)**

NEXT POST → (HTTPS://BLOG.ROPNOP.COM/EXTRACTING-SSH-PRIVATE-KEYS-FROM-WINDOWS-10-SSH-AGENT/)

SHOW 16 COMMENTS

(mailto:rflathers@gmail.com)   (https://github.com/ropnop)   (https://twitter.com/ropnop)   ()

ropnop (blog.ropnop.com) • © 2020 • ropnop blog (https://blog.ropnop.com)

Hugo v0.63.2 (https://gohugo.io) powered • Modified theme based on Beautiful Hugo (https://github.com/halogenica/beautifulhugo)