

Anchored

Description

- A client asked me to check if I can intercept the https request and get the value of the secret parameter that is passed along with the user's email. The application is intended to run in a non-rooted device. Can you help me find a way to intercept this value in plain text.

Objective

- Decompile th APK file, and change the network security configuration to intercept the HTTPS request and get the flag.

Difficulty

- Medium

Flag

- HTB{UnTrUst3d_C3rT1f1C4T3s}

Release:

- [/release/Anchored.zip](#)
(09079c000742b01298c6c57232fb1d0f9a6c488374f2325b537d238c0d510b3a)

Prerequisites

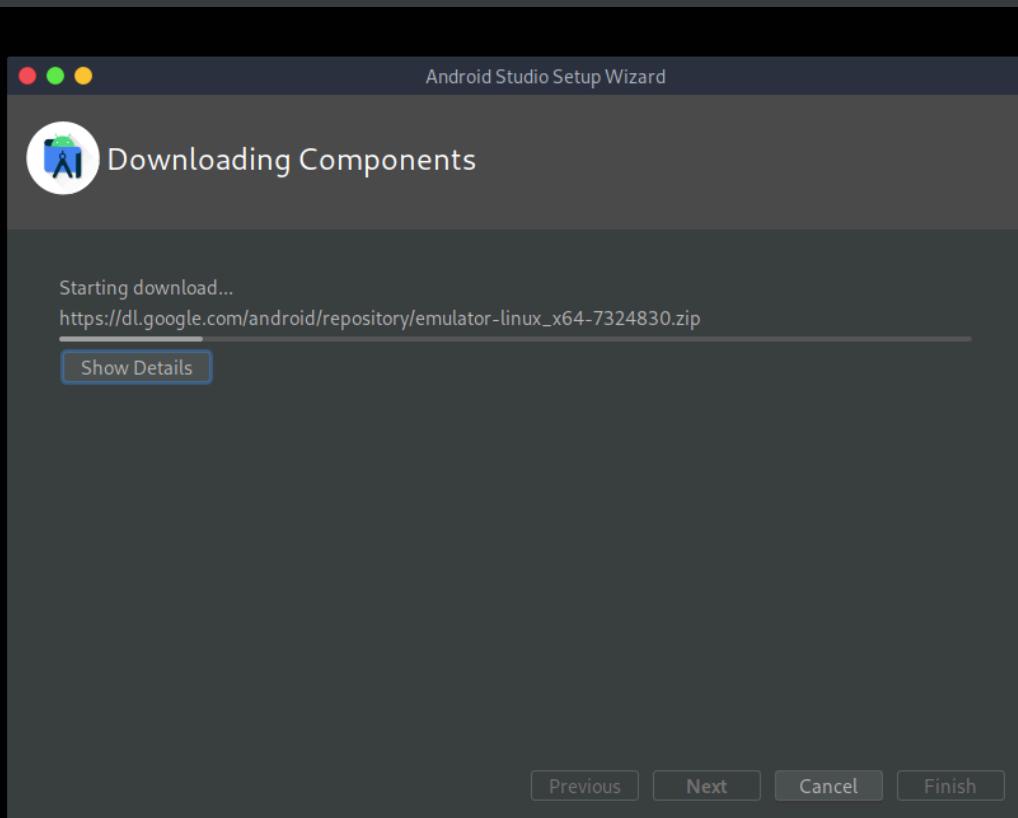
- A virtual Android image like [Android-x86](#) running on Virtual Box or VMware or another Android emulator, running with the Developer mode on.
- Alternatively, a real android device connected via USB, running with the Developer mode on.

Challenge:

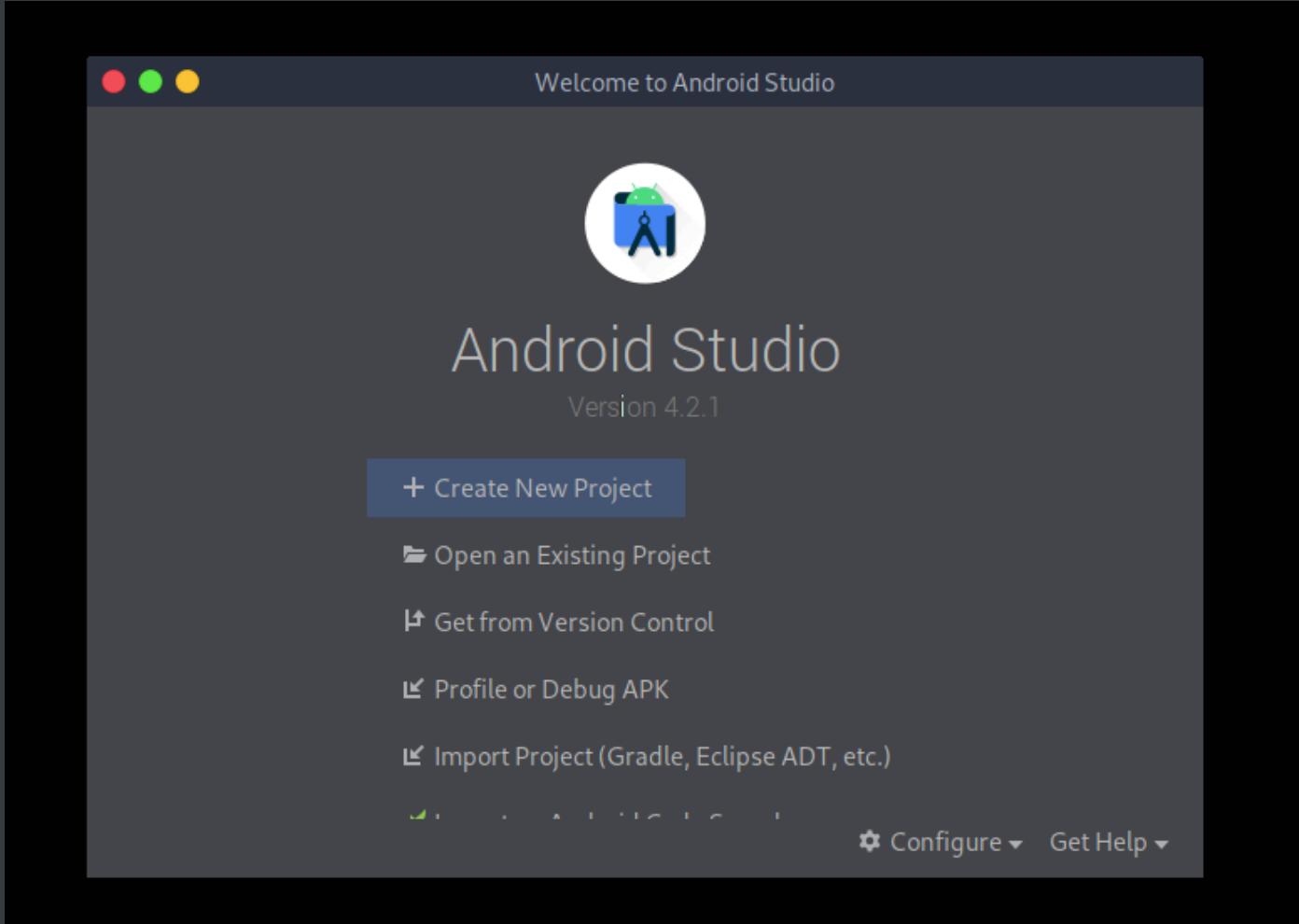
Unzipping the `Anchored.zip` file reveals the file `Anchored.apk`. In order to run the `Anchored.apk` file, we have to set up an Android emulator. To achieve this, we are going to use [Android Studio IDE](#).

```
wget https://redirector.gvt1.com/edgedl/android/studio/ide-zips/4.2.1.0/android-studio-ide-202.7351085-linux.tar.gz  
tar xvzf android-studio-ide-202.7351085-linux.tar.gz  
sh android-studio/bin/studio.sh
```

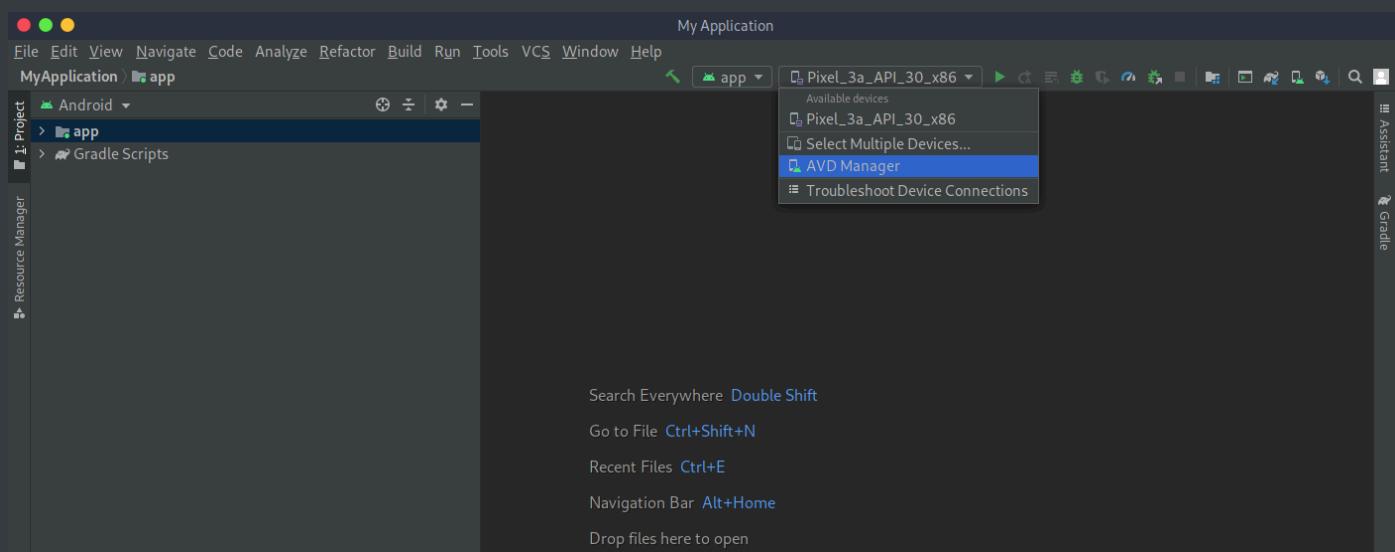
On the setup wizard we click `OK`, then we click on `Next`, and finally click on `Finish`. Next, we wait for the Android Studio to download the components.



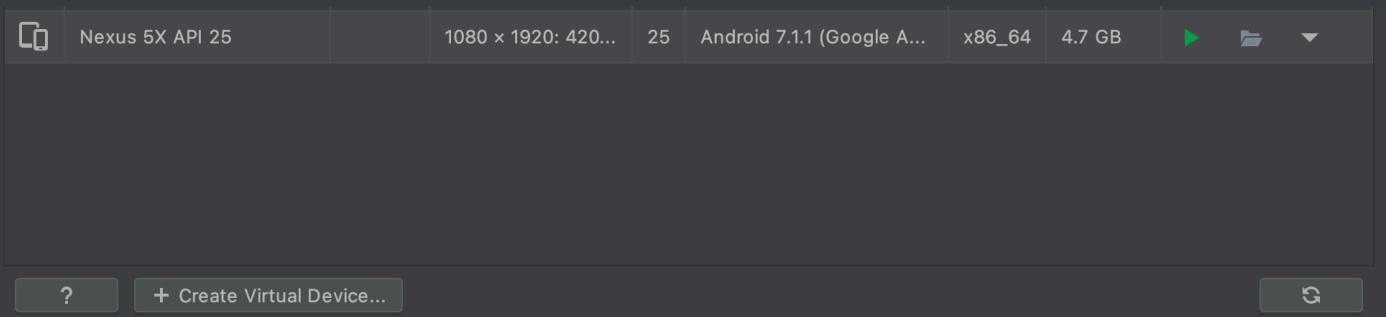
Once it's done, we click `Finish` once again.



Then we click `Next` and finally we click on `Finish`. Now that we have created a new project, we wait for some more files to get downloaded automatically from the IDE. When that's done, click on the top centre of the IDE and select `AVD Manager`.



On the AVD Manager menu, click on the `Create Virtual Device...` button to create a new device.



On the next window, we choose a device. In this example, we will be using a `Nexus 5X` device.

Choose a device definition

Category	Name	Play Store	Size	Resolution	Density
TV	Nexus One		3.7"	480x800	hdpi
Phone	Nexus 6P		5.7"	1440x2...	560dpi
Wear OS	Nexus 6		5.96"	1440x2...	560dpi
Tablet	Nexus 5X	▶	5.2"	1080x1...	420dpi
Automotive	Nexus 5	▶	4.95"	1080x1...	xhdpi
	Nexus 4		4.7"	768x1280	xhdpi
	Galaxy Nexus		4.65"	720x1280	xhdpi

New Hardware Profile Import Hardware Profiles Refresh Clone Device...

Nexus 5X

1080px
5.2"
1920px

Size: large
Ratio: long
Density: 420dpi

Once we have chose the device, we click `Next`. In the next window, we click on the `x86 Images` tab, and select an image that includes `Google Play`. That also means that the device wont be rooted, as it's also suggested from the description. In this example we are going to use a `x86_64` image with `API Level 29`.

Select a system image

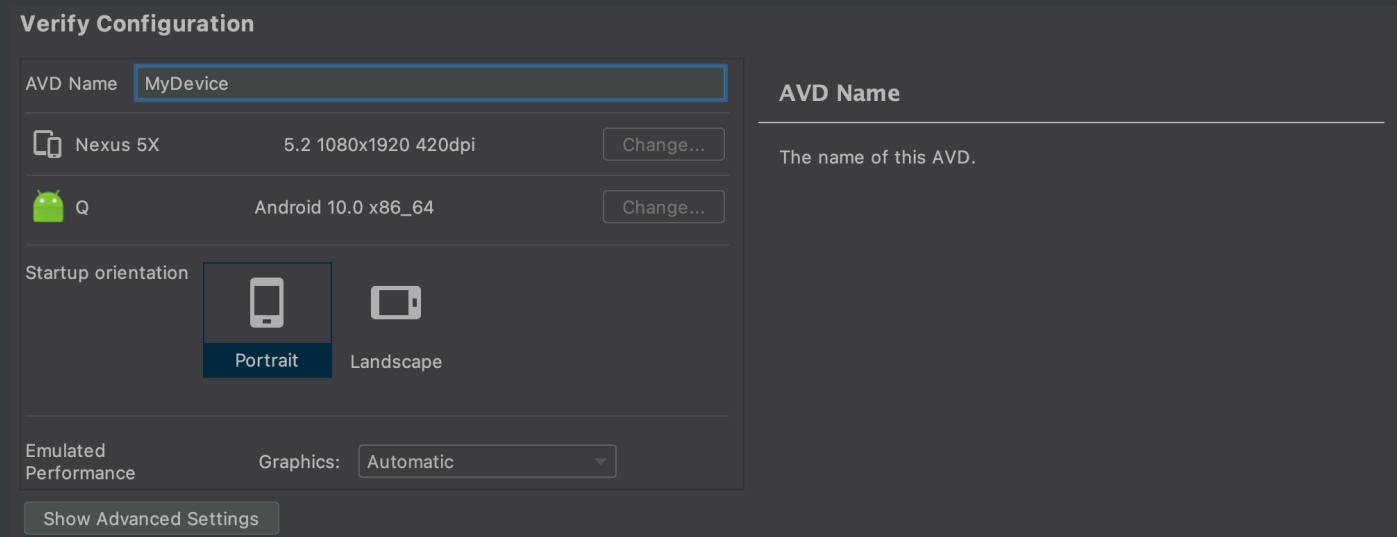
Recommended	x86 Images	Other Images	
Release Name	API Level ▾	ABI	Target
Q	29	x86_64	Android 10.0 (Google Play)
Q Download	29	x86_64	Android 10.0 (Google APIs)
Q Download	29	x86	Android 10.0 (Google APIs)
Q	29	x86_64	Android 10.0
Q Download	29	x86	Android 10.0
Pie Download	28	x86_64	Android 9.0 (Google Play)
Pie Download	28	x86_64	Android 9.0 (Google APIs)
Pie Download	28	x86	Android 9.0 (Google APIs)
Pie Download	28	x86	Android 9.0

Q

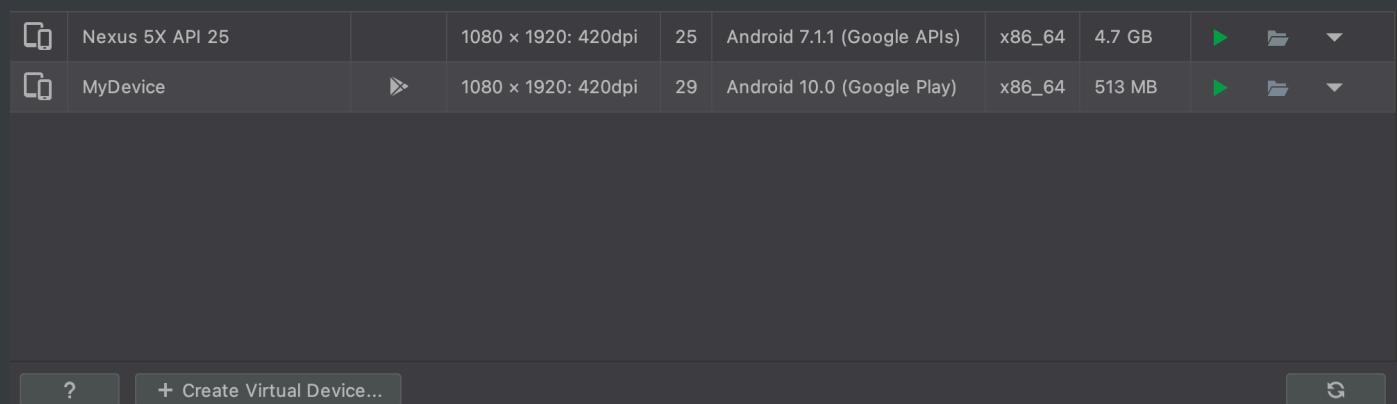
API Level
29
Android
10.0
Google Inc.
System Image
x86_64

Questions on API level?
See the [API level distribution chart](#)

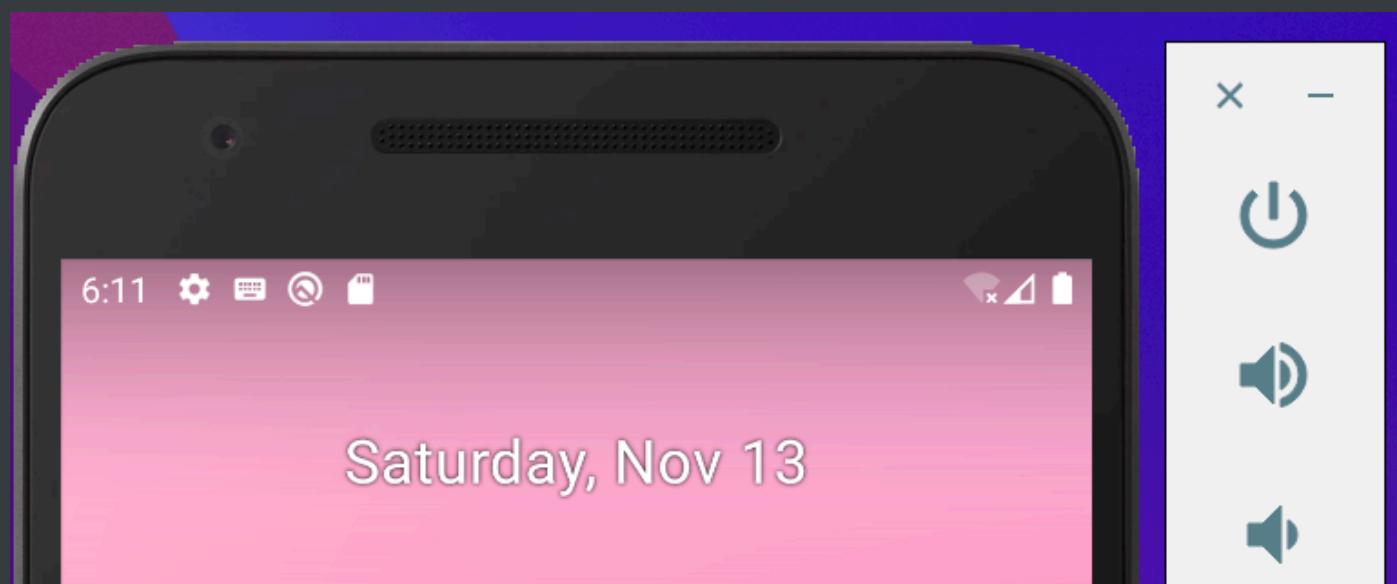
If it's not already downloaded, click on the `Download` link first. Then, select it and click `Next`. On the final window, name the device and click `Finish`.



Finally, we can click on the green `play` button to start the emulator.



Once the device is started, It should be looking like this.





Next step, is to install `adb` so we can communicate with it.

```
sudo apt install adb
```

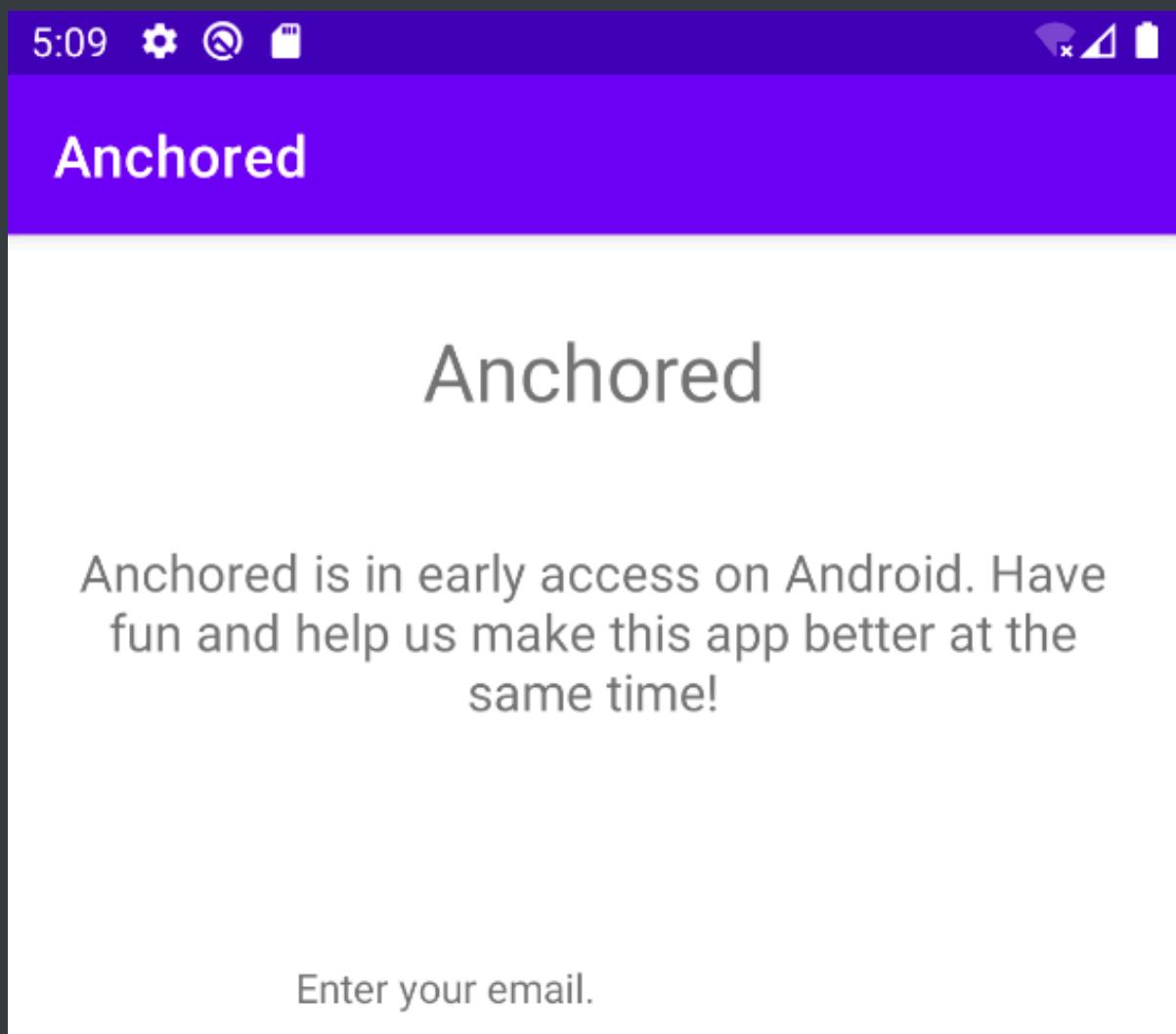
While the device is running, we can execute the following command to install the application on the device.

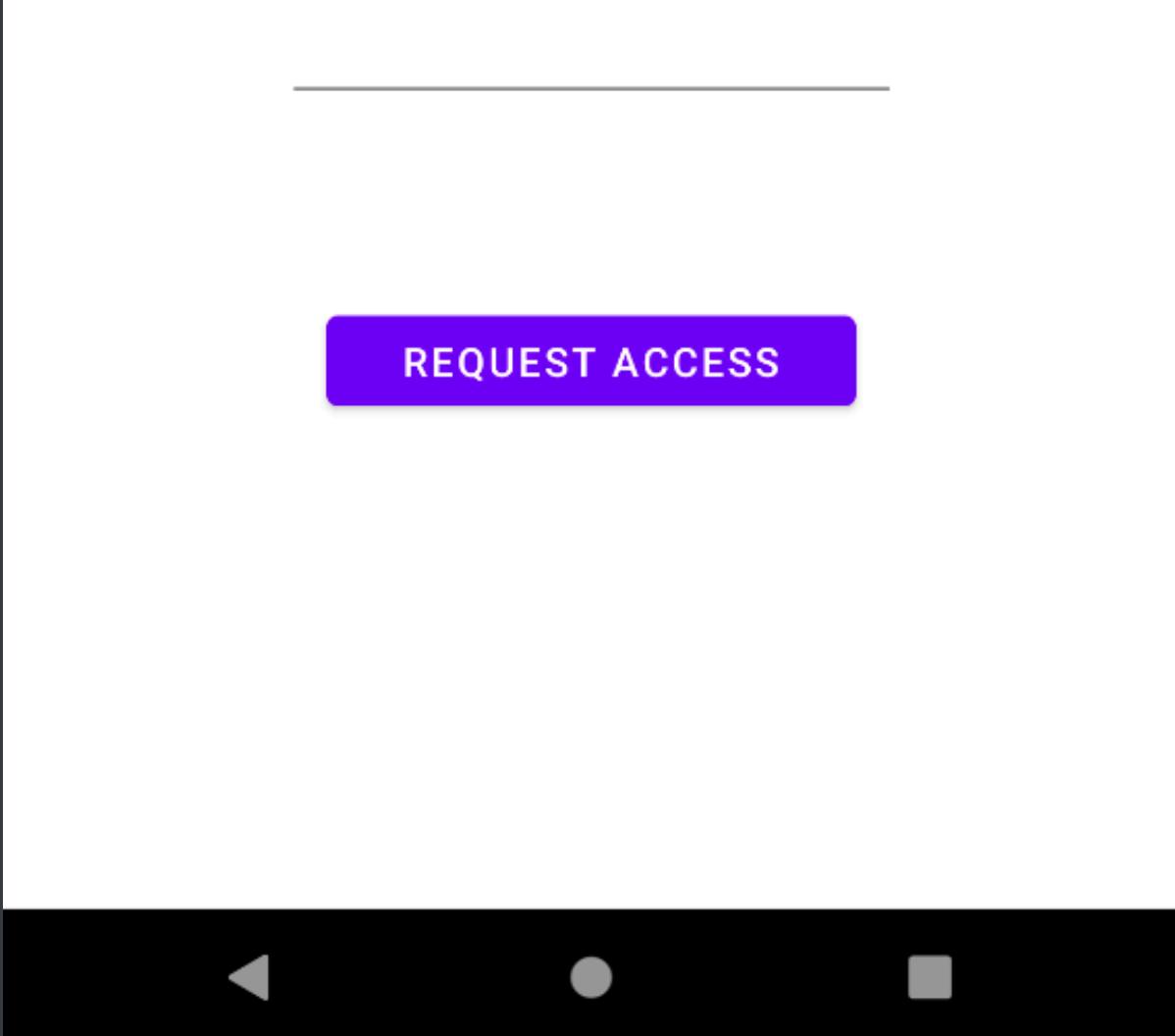
```
adb install Anchored.apk
```



adb install Anchored.apk
Performing Streamed Install
Success

We can now run the application on the Android device.





REQUEST ACCESS

This app lets us put our email address in order to get early access. According to the description, we need to find a way to intercept the value of a secret parameter that is past during the HTTPS request. To do so, we need to set up Burp Suite working with the Android emulator. First, we type the following on the host machine to get the ip address.

```
ifconfig
```



```
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
<SNIP>
    inet 192.168.1.17 netmask 0xffffffff broadcast 192.168.1.255
        nd6 options=201<PERFORMNUD,DAD>
        media: autoselect
        status: active
```

On the Burp Suit Proxy tab, go on Options and press the Add button under the Proxy Listeners section.

The screenshot shows the Burp Suite interface with the 'Proxy' tab selected. Under the 'Options' sub-tab, the 'Proxy Listeners' section is displayed. A table lists a single listener entry:

Running	Interface	Invisible	Redirect	Certificate	TLS Protocols
<input checked="" type="checkbox"/>	127.0.0.1:8080			Per-host	Default

Below the table, a note states: "Each installation of Burp generates its own CA certificate that Proxy listeners can use when negotiating TLS connections. You can import or regenerate it." Two buttons are present: "Import / export CA certificate" and "Regenerate CA certificate".

On the pop up window type the port 8090 and select the host's IP from the drop down menu.

The screenshot shows the 'Binding' tab of a configuration dialog box. The note says: "These settings control how Burp binds the proxy listener." The 'Bind to port:' field contains '8080'. The 'Bind to address:' section has three options: 'Loopback only' (radio button), 'All interfaces' (radio button), and 'Specific address:' (radio button, selected) followed by a dropdown menu containing '192.168.1.17'. At the bottom right are 'OK' and 'Cancel' buttons.

Click OK and make sure the new proxy listener is selected.

Dashboard Target **Proxy** Repeater Intruder Sequencer Decoder Comparer Logger Extender

Intercept HTTP history WebSockets history Options

Proxy Listeners

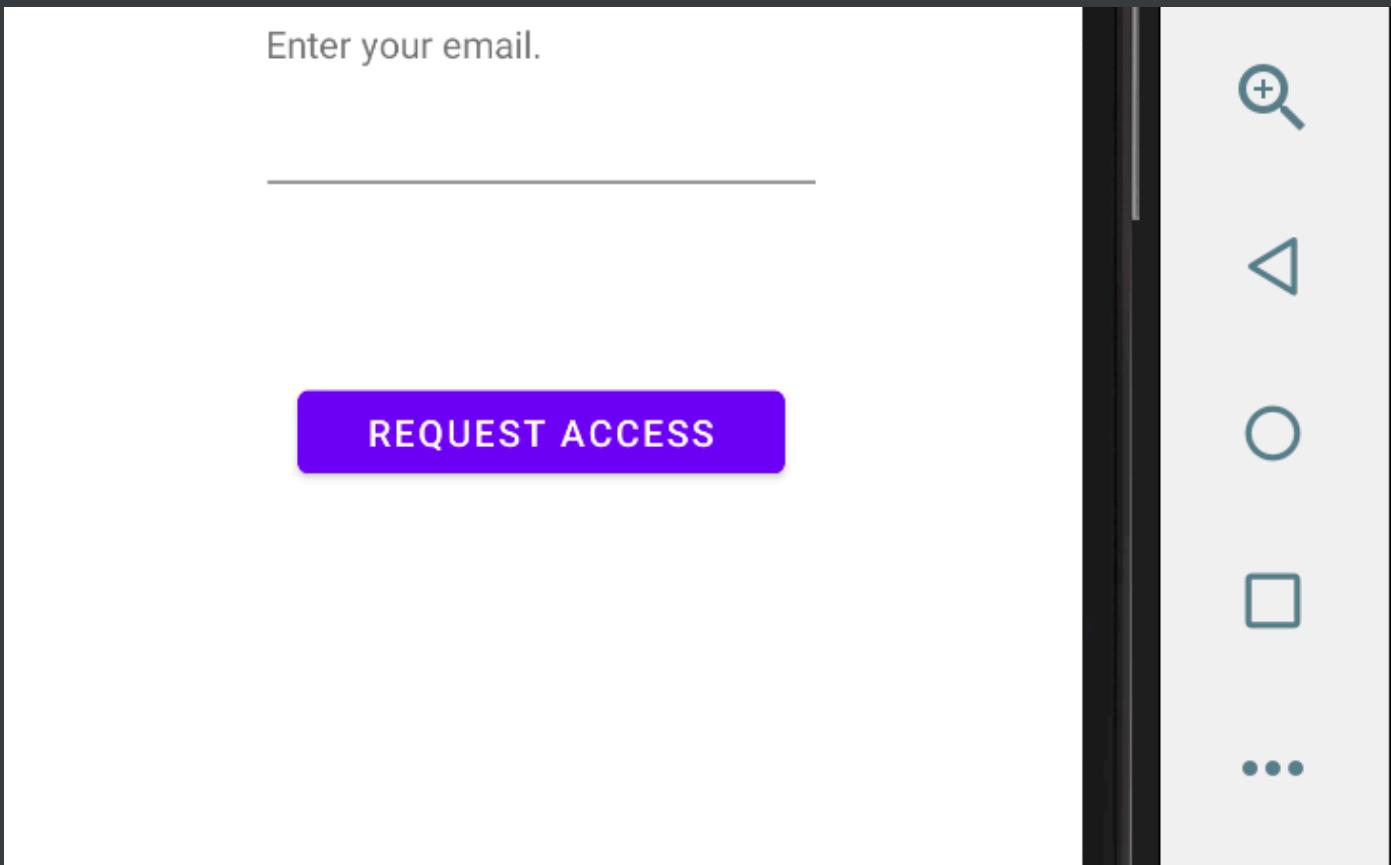
Burp Proxy uses listeners to receive incoming HTTP requests from your browser. You will need to configure your browser to use one of the listed listeners.

Add	Running	Interface	Invisible	Redirect	Certificate	TLS Protocols
<input type="checkbox"/>	127.0.0.1:8080				Per-host	Default
<input checked="" type="checkbox"/>	192.168.1.17:8090				Per-host	Default

Each installation of Burp generates its own CA certificate that Proxy listeners can use when negotiating TLS connections. You can import or regenerate it.

[Import / export CA certificate](#) [Regenerate CA certificate](#)

Back to the Android emulator, click the three dots from the vertical menu near the device.



On the Extended Controls pop up window, click on settings. Then, uncheck the Use Android Studio HTTP proxy settings, and check the Manual proxy configuration. On the Host name field, add the host IP address, and on the Port number add 8090. Finally click APPLY.

Extended Controls - MyDevice:5554

The screenshot shows the 'Proxy' tab selected in the 'General' section of the Extended Controls app. The 'Manual proxy configuration' option is selected. The host name is set to '192.168.1.17' and the port number is '8090'. The 'Proxy authentication' section is collapsed. The 'Apply' button is visible at the bottom left, and the 'Proxy status' is shown as 'Success'.

- Location
- Displays
- Cellular
- Battery
- Camera
- Phone
- Directional pad
- Microphone
- Fingerprint
- Virtual sensors
- Bug report
- Snapshots
- Record and Playback
- Google Play
- Settings
- Help

General **Proxy** Advanced

Use Android Studio HTTP proxy settings

No proxy

Manual proxy configuration

Host name
192.168.1.17

Port number
8090

Proxy authentication

Login
username

Password
XXXX

Apply

Proxy status
Success

The Proxy status should say Success . If not, make sure that the Intercept is on button on the Intercept tab on Burp, is toggled on. Finally, fill in your email address on the app, and click the REQUEST ACCESS button. The request should now be intercepted on Burp.

Use Burp's embedded browser

There's no need to configure your proxy settings manually. Use Burp's embedded Chromium browser to start testing right away.

Open browser

Using Burp Proxy

If this is your first time using Burp, you might want to take a look at our guide to help you get the most out of your experience.

Burp Proxy options

Reference information about the different options you have for customizing Burp Proxy's behaviour.

As we can see, it failed to intercept the request and the tab `Dashboard` is now toggled on. Navigating to the `Dashboard` tab, we can see that the alert `The client failed to negotiate a TLS connection to anchored.com:4443: Received fatal alert: certificate_unknown` has been raised, and it's visible in the last line.

Event log

Filter: Critical, Error, Info, Debug

Search...

.....

Message ^

..... [11] The client failed to negotiate a TLS connection to mobilenetworkscoring-pa.googleapis.com:443: Received fatal alert: certificate_unknown

[2] Proxy service started on 192.168.1.18:8090

[2] The client failed to negotiate a TLS connection to play.googleapis.com:443: Received fatal alert: certificate_unknown

[2] The client failed to negotiate a TLS connection to www.google.com:443: Received fatal alert: certificate_unknown

[2] The client failed to negotiate a TLS connection to www.googleapis.com:443: Received fatal alert: certificate_unknown

[3] The client failed to negotiate a TLS connection to anchored.com:4443: Received fatal alert: certificate_unknown

This means that the connection is secure and we don't provide the specific certificate that is used by the app. Let's decompile the `Anchored.apk` file using `Apktool`, and see if we can get any useful information about the app.

```
java -jar apktool_2.5.0.jar d Anchored.apk
```



```
java -jar apktool_2.5.0.jar d Anchored.apk
I: Using Apktool 2.5.0 on Anchored.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: /home/bertolis/.local/share/apktool
/framework/1.apk
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Baksmaling classes.dex...
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...
```

Decompilation of the `Anchored.apk` file reveals the `./Anchored/AndroidManifest.xml`.

```
cat ./Anchored/AndroidManifest.xml
```



```
cat ./Anchored/AndroidManifest.xml
<?xml version="1.0" encoding="utf-8" standalone="no"?><manifest
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:compileSdkVersion="31" android:compileSdkVersionCodename="12"
    package="com.example.anchored" platformBuildVersionCode="31"
    platformBuildVersionName="12">
    <uses-permission android:name="android.permission.INTERNET"/>
    <application android:allowBackup="true"
        android:AppComponentFactory="androidx.core.app.CoreComponentFactory"
        android:icon="@mipmap/ic_launcher" android:label="@string/app_name"
        android:networkSecurityConfig="@xml/network_security_config"
        android:roundIcon="@mipmap/ic_launcher_round" android:supportsRtl="true"
        android:theme="@style/Theme.Anchored">
        <activity android:exported="true"
            android:name="com.example.anchored.MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category
                    android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
    </application>
</manifest>
```

AndroidManifest.xml seems also to include

android:networkSecurityConfig="@xml/network_security_config". Reading the content of this file reveals the following.

```
cat ./Anchored/res/xml/network_security_config.xml
```



```
cat ./Anchored/res/xml/network_security_config.xml
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <domain-config cleartextTrafficPermitted="false">
        <domain includeSubdomains="true">anchored.com</domain>
        <trust-anchors>
            <certificates src="@raw/certificate" />
        </trust-anchors>
    </domain-config>
</network-security-config>
```

As the `network_security_config.xml` file indicates, the app reads the ssl certificate for the domain `anchored.com`, from the file `@raw/certificate`. This path in the android project structure, is the path `project_name/res/raw/certificate`. We can check if the certificate is in this path by issuing the following command.

```
ls -l ./Anchored/res/raw/
```



```
ls -l Anchored/res/raw/
total 4
-rw-r--r-- 1 bertolis bertolis 1359 Nov 14 18:31 certificate.pem
```

In order to intercept the request and get the value of the parameter in plain text, we could issue a new certificate using Burp, add it in the user's `Trusted credentials` of the device, and finally include `<certificates src="user" />` in the `network_security_config.xml`. According to the offical documentation [Network security configuration](#), there is a way to allow the app to trust any certificate that is included in the user's `Trusted credentials` of the device.

The source of CA certificates. Each certificate can be one of the following:

- a raw resource ID pointing to a file containing X.509 certificates. Certificates must be encoded in DER or PEM format. In the case of PEM certificates, the file *must not* contain extra non-PEM data such as comments.
- "system" for the pre-installed system CA certificates
- "user" for user-added CA certificates

The following snippet indicates that we can achieve this by including the line

```
<certificates src="user" />
```

 into the `network_security_config.xml` file.

```
<base-config cleartextTrafficPermitted="true">
    <trust-anchors>
        <certificates src="system" />
        <certificates src="user" />
    </trust-anchors>
</base-config>
```

The `network_security_config.xml` should look like this.



```
cat Anchored/res/xml/network_security_config.xml
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <domain-config cleartextTrafficPermitted="false">
        <domain includeSubdomains="true">anchored.com</domain>
        <trust-anchors>
            <certificates src="user" />
            <certificates src="@raw/certificate" />
        </trust-anchors>
    </domain-config>
</network-security-config>
```

Then, we recompile the application by issuing the following command.

```
java -jar apktool_2.5.0.jar b Anchored
```



```
java -jar apktool_2.5.0.jar b Anchored
I: Using Apktool 2.5.0
I: Checking whether sources has changed...
I: Smaling smali folder into classes.dex...
I: Checking whether resources has changed...
I: Building resources...
I: Copying libs... (/lib)
I: Building apk file...
I: Copying unknown files/dir...
I: Built apk...
```

Before we run this on the Android device, we need to self sign it, otherwise it's not going to be executed. Issue the following commands in order to sign the file. Put a random password, first and last name, and press enter on the other fields as shown in the picture below. On the last field, type yes .

```
cd Anchored/dist/
keytool -genkey -keystore john.keystore -validity 1000 -alias john
```



```
keytool -genkey -keystore john.keystore -validity 1000 -alias john
Enter keystore password:
Re-enter new password:

Warning:
No -keyalg option. The default key algorithm (DSA) is a legacy algorithm
and is no longer recommended. In a subsequent release of the JDK, the
default will be removed and the -keyalg option must be specified.

What is your first and last name?
[Unknown]: john doe
What is the name of your organizational unit?
[Unknown]:
What is the name of your organization?
[Unknown]:
What is the name of your City or Locality?
[Unknown]:
What is the name of your State or Province?
[Unknown]:
What is the two-letter country code for this unit?
[Unknown]:
Is CN=john doe, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=Unknown
correct?
[no]: yes

Generating 2,048 bit DSA key pair and self-signed certificate
(SHA256withDSA) with a validity of 1,000 days
    for: CN=john doe, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown,
C=Unknown
```

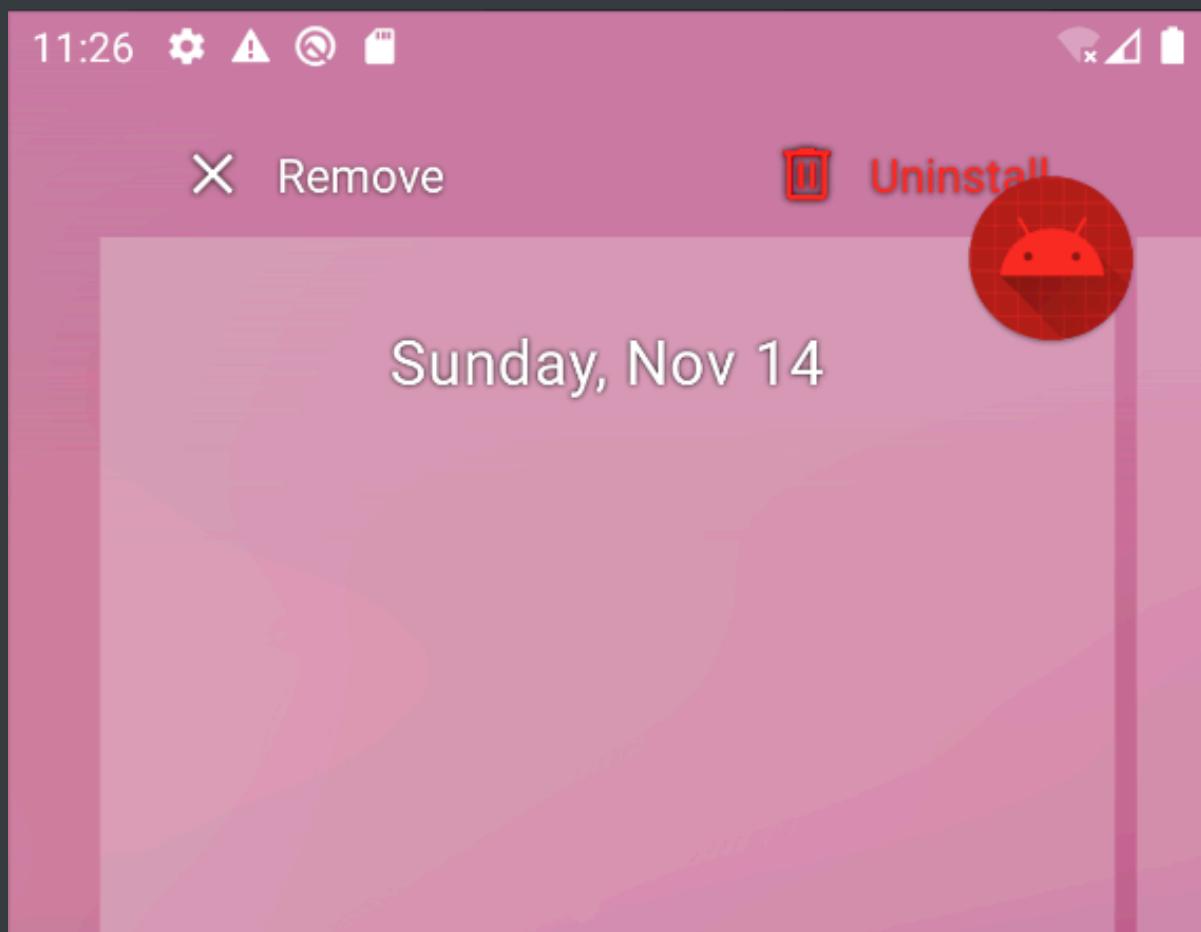
The key has been generated successfully. Let's now use it to sign the APK file.

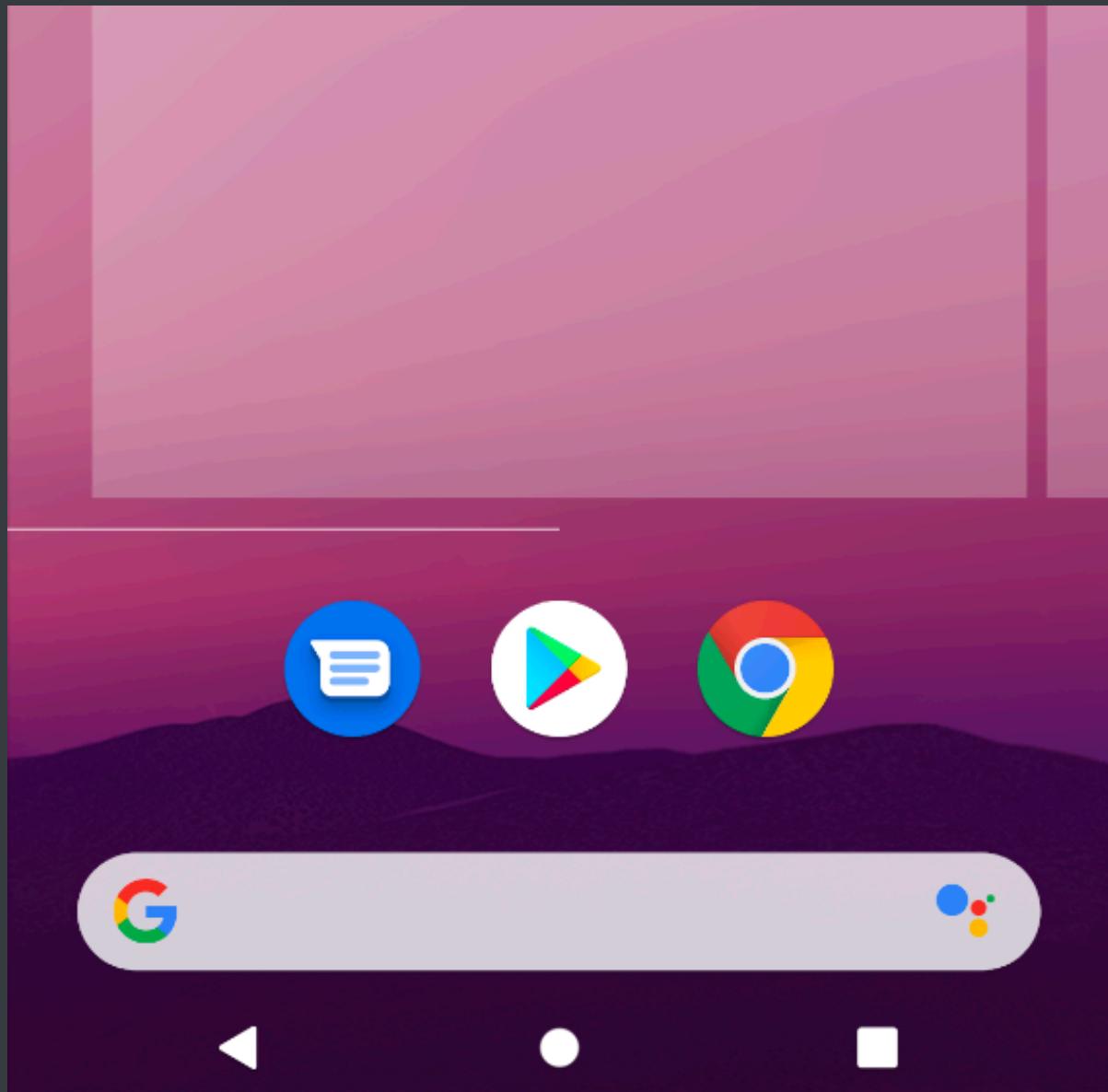
```
jarsigner -keystore john.keystore -verbose Anchored.apk john
```



```
jarsigner -keystore john.keystore -verbose Anchored.apk john  
Enter Passphrase for keystore:  
adding: META-INF/MANIFEST.MF  
adding: META-INF/JOHN.SF  
adding: META-INF/JOHN.DSA  
signing: classes.dex  
signing: resources.arsc  
signing: AndroidManifest.xml  
  
<SNIP>  
jar signed.  
  
Warning:  
The signer's certificate is self-signed.
```

The new APK file has been successfully signed. Now we can try to run the edited application on our Android device. First we have to uninstall the previous application from the device, because the sign is different and it will throw an error. From the menu, tap and hold the application and drag it over the right corner that says **Uninstall**.





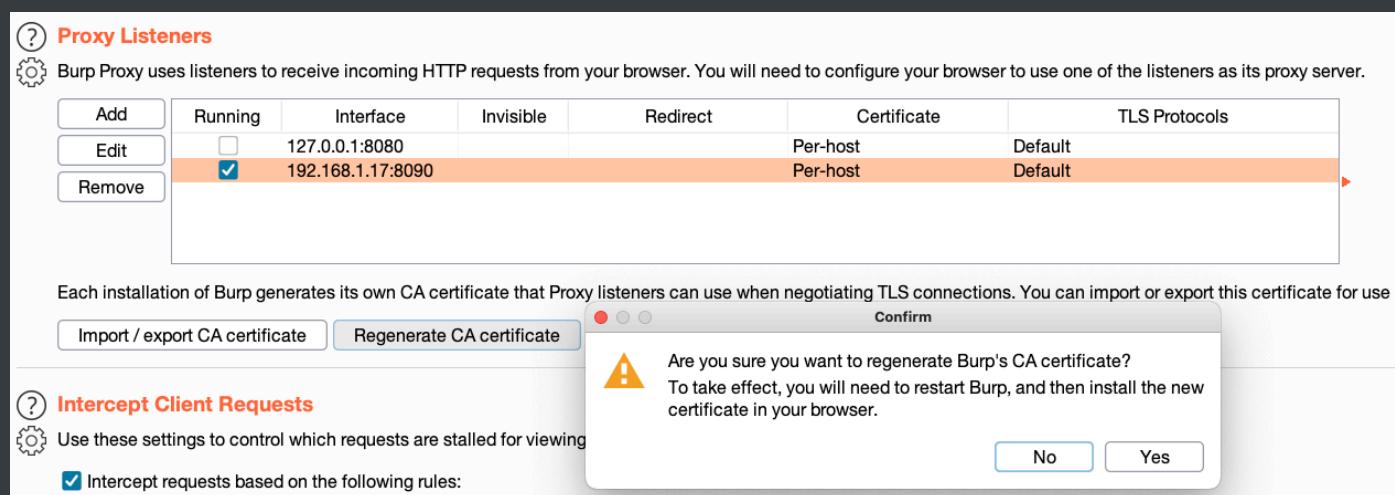
Then, install the new APK file.

```
adb install Anchored/dist/Anchored.apk
```

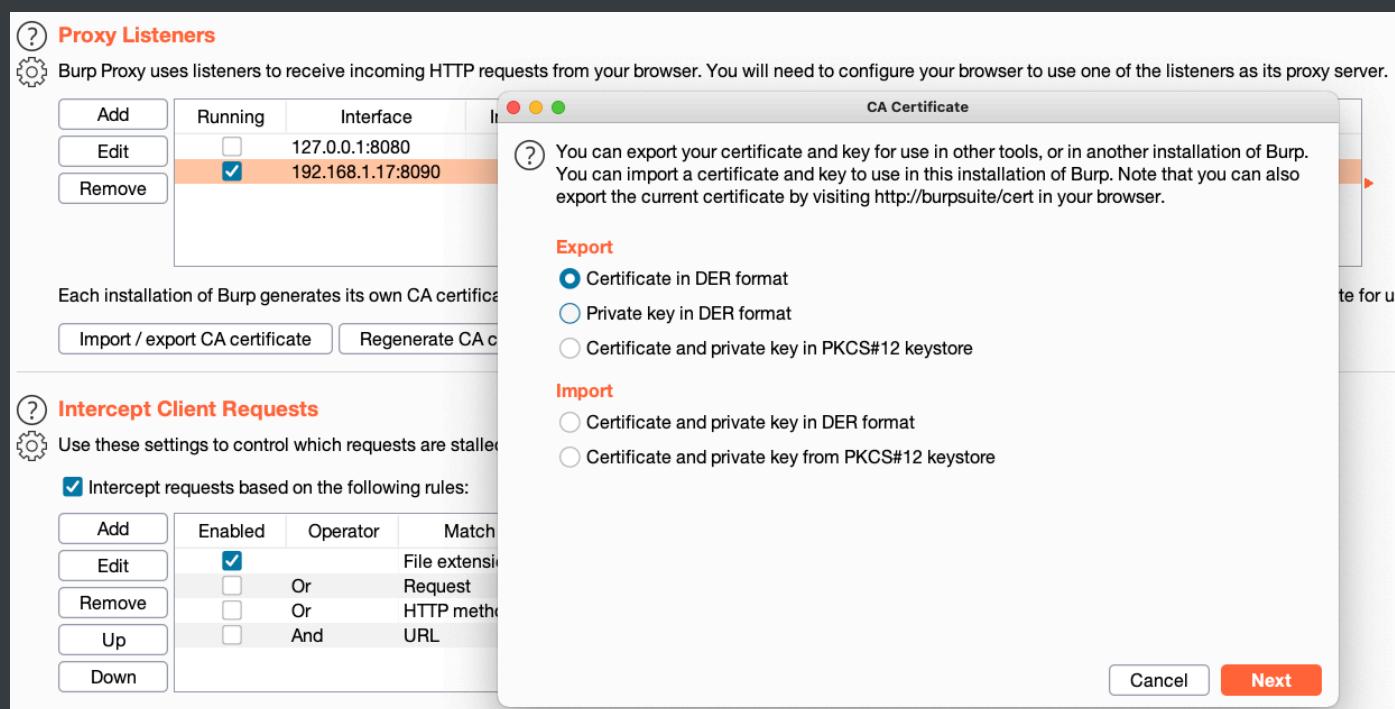


```
adb install Anchored.apk  
Performing Streamed Install  
Success
```

Let's now proceed on issueing a new certificate with Burp, and install it on the user's Trusted credentials of the device. On the Proxy -> Options tab on Burp, we click Regenerate CA Certificate .



On the confirmation dialog, press Yes . Then, we click on Import / export CA certificate . On export section on the popup window, select Certificate in DER format and click Next .



Next, click on select file .

 Choose a file to export the CA certificate.

Select file ...

[Back](#)

[Next](#)

Name it `cert-der.crt` and save it to `Downloads` directory.

File Name:	<input type="text" value="cert-der.crt"/>
Files of Type:	<input type="text" value="All Files"/>
<input type="button" value="Save"/> <input type="button" value="Cancel"/>	

Then click on `Next` once again and then `Close`.

 Choose a file to export the CA certificate.

Select file ...BackNext

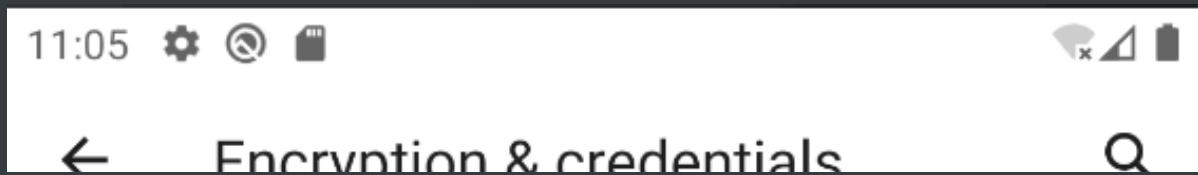
To upload the certificate to the emulator, type the following.

```
adb push cert-der.crt /sdcard/Downloads
```



```
adb push cert-der.crt /sdcard/Download/  
cert-der.crt: 1 file pushed, 0 skipped. 0.4 MB/s (939 bytes in 0.002s)
```

Once the certificate is pushed, we also need to install it to the device. Navigate to Settings -> Security -> Encryption & credentials , and tap on the Install from SD card .



Encryption & credentials

ENCRYPTION

Encrypt phone

Encrypted

CREDENTIAL STORAGE

Storage type

Hardware-backed

Trusted credentials

Display trusted CA certificates

User credentials

View and modify stored credentials

Install from SD card

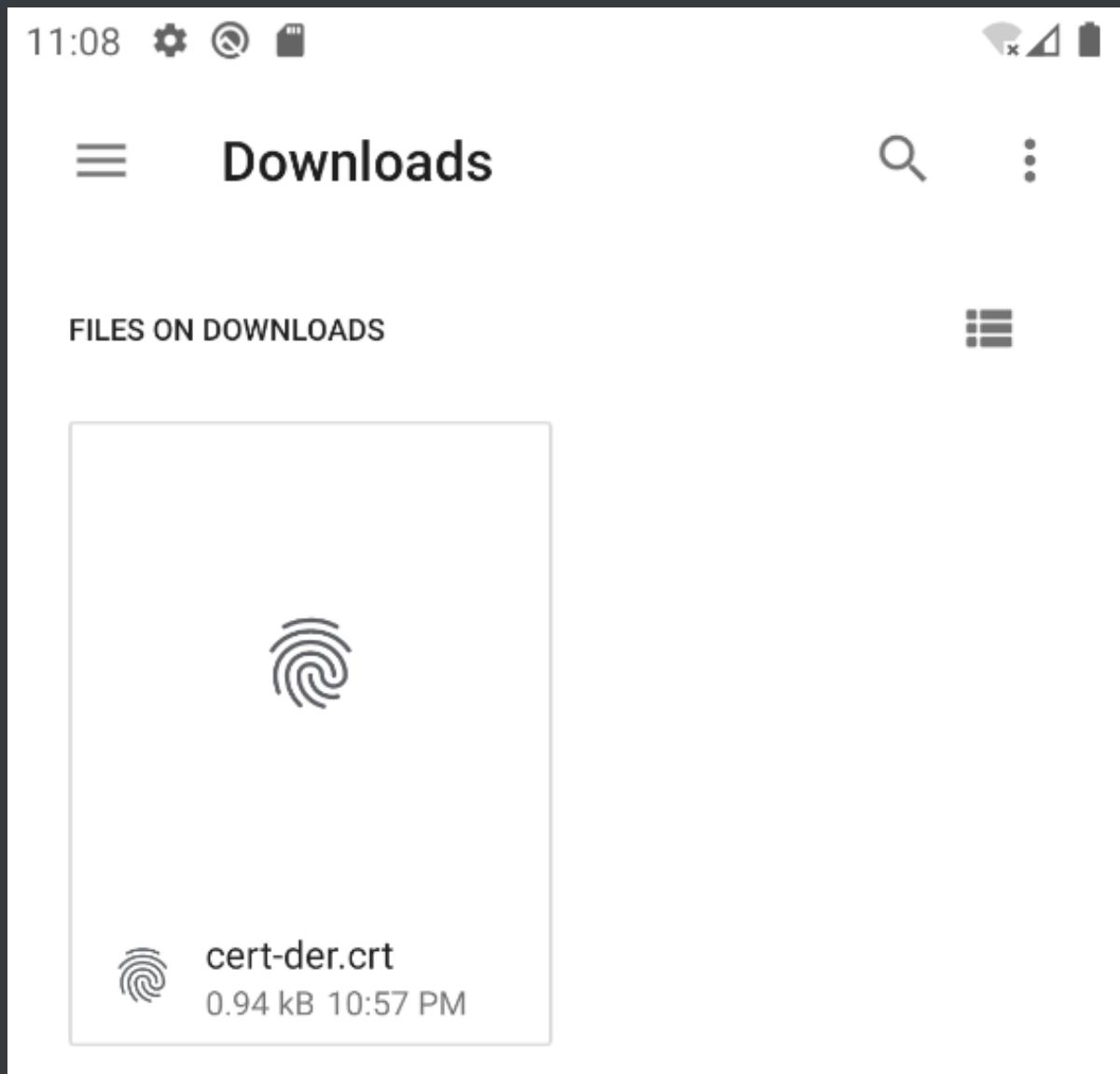
Install certificates from SD card

Clear credentials

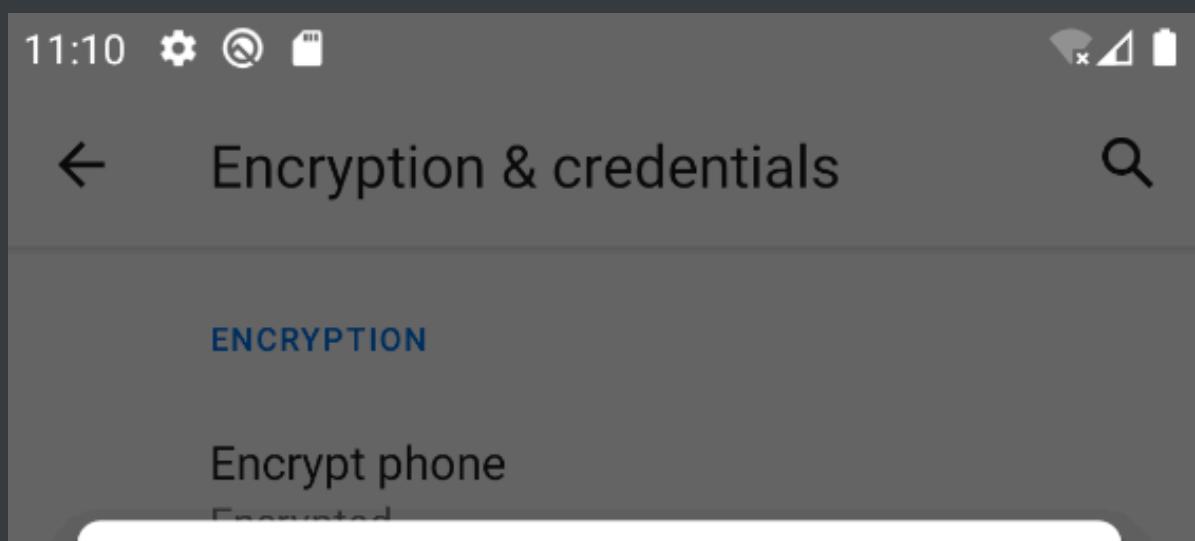
Remove all certificates



Once we tap on it, we navigate to `Downloads` and tap on the certificate that we upload in the previous step.



Tap on it, give the certificate a name and tap `OK`.



Name the certificate

Certificate name:

anchored

Credential use:

VPN and apps

Note: The issuer of this certificate may inspect all traffic to and from the device.

The package contains:

one CA certificate

Cancel

OK

Clear credentials

Remove all certificates



Let's now run the app and try to intercept the request one more time. Make sure that the Intercept is on button on the Intercept tab on Burp is toggled on, and tap the REQUEST ACCESS button once again on the app. Back on the Burp, we can see that the value UnTrUst3d_C3rT1f1C4T3s of the parameter msg is captured in plain text.

The screenshot shows the OWASP ZAP proxy interface. The 'Proxy' tab is selected. Under the 'Intercept' tab, a message is displayed: "Request to https://anchored.com:4443 [24.205.141.134]". Below this are buttons for 'Forward', 'Drop', 'Intercept is on' (which is highlighted in blue), 'Action', and 'Open Browser'. At the bottom, there are tabs for 'Pretty' (selected), 'Raw', 'Hex', '\n', and a three-dot menu.

```
1 POST /test.php HTTP/1.1
2 Content-Type: application/x-www-form-urlencoded
3 User-Agent: Dalvik/2.1.0 (Linux; U; Android 10; Android SDK built for x86_64 Build/QSR1.190920.001)
4 Host: anchored.com:4443
5 Connection: close
6 Accept-Encoding: gzip, deflate
7 Content-Length: 49
8
9 msg=UnTrUst3d_C3rT1f1C4T3s&mail=test%40gmail.com&
```

The flag is HTB{UnTrUst3d_C3rT1f1C4T3s} .