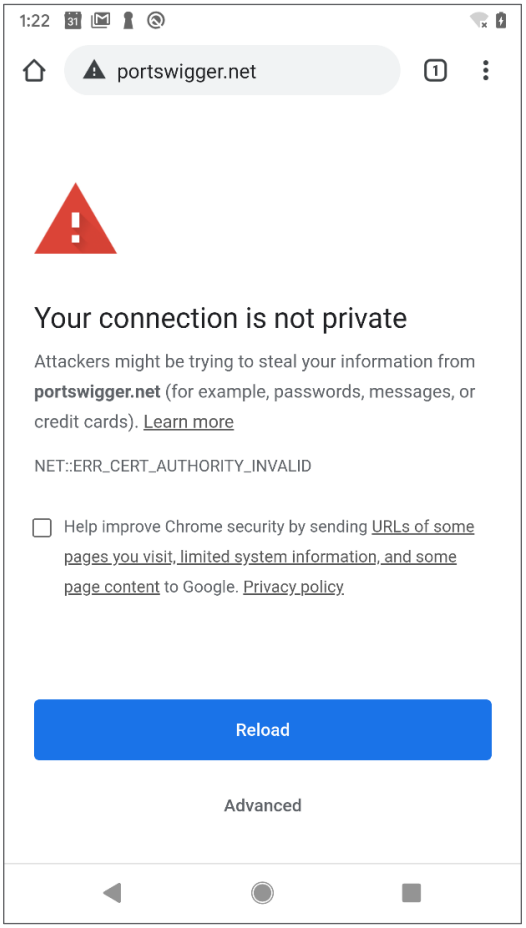
Android SSL Pinning Bypass: Android 7-10

The McAfee Advanced Threat Research team conducts security research with the aim of staying ahead of the evolving threat landscape to expose and reduce attack surfaces. This series of white papers discusses laboratory security research techniques that are generally known among the professional community of security researchers. The white papers are provided to elevate collaboration and security within the industry and are not to be used for unlawful purposes. Security researchers are responsible for lawfully obtaining equipment and for complying with contracts and licenses for their research.

This technique will work on Android version 7 (Nougat) up to the latest version which, at the time of writing, is Android 10. The Android operating system has made some modifications to the way user certificates are trusted, meaning they are no longer trusted as a root certificate authority. This means that you will be prompted with the following message if you try to use BurpSuite's proxy to inspect SSL traffic on both SSL pinned applications, as well as the unpinned counterparts:



This means that all applications will not trust the “user installed” certificate, preventing us from using it for our unpinning tasks.

While I am not going to go over how to root an Android phone, you can follow this tutorial for a pixel device on Android 10 (https://highonandroid.com/android-root/how-to-root-android-10-pixel-pixel-2-pixel-3-pixel-3a/). The process has changed in the last few Android releases. Android now has two separate partitions for everything, an “a” and a “b”. They have also made it so you have to patch the boot partition image for things like a third-party recovery. This is also the case for installing root, but since many of the files are overwritten in the patching process you must only choose one, either a third-party recovery or a root patched boot image. For this tutorial the root patch is required.

You will also need to setup Burp Suite for the network capture and SSL decrypting. A great tutorial can be found here (https://support.portswigger.net/customer/portal/articles/1841101-configuring-an-android-device-to-work-with-burp) and, if you need help installing the Burp Suite SSL Certificate on the Android device, you can follow these instructions (https://support.portswigger.net/customer/portal/articles/1841102-Mobile%20Set-up_Android%20Device%20-%20Installing%20CA%20Certificate.html).

Once you have the Burp Suite proxy up and running you can download the Burp Suite CA by navigating to http://<ip of proxy>/cert. This will download the CA certificate in a DER format. That will need to be converted to a PEM file. This can be done with the following:

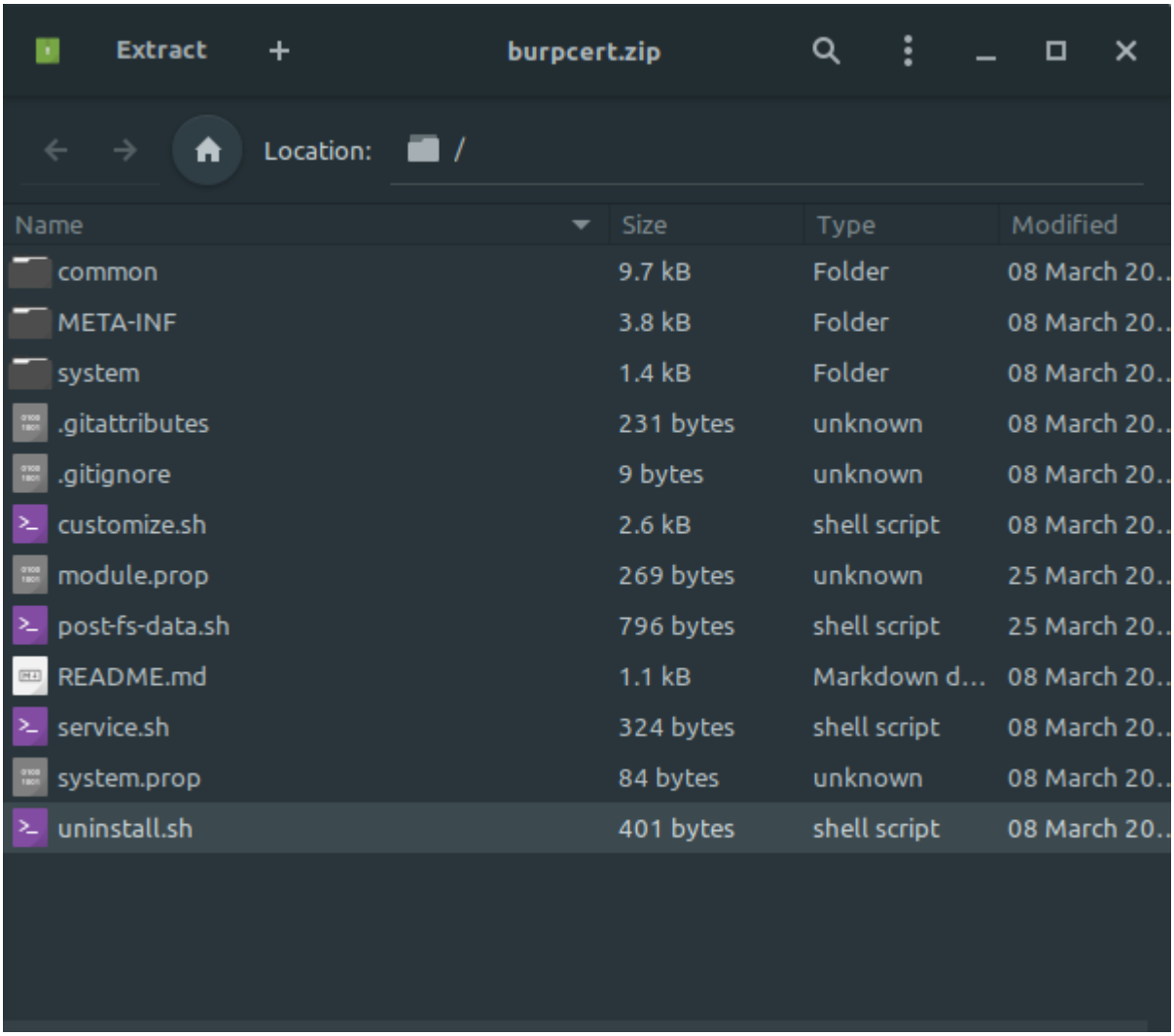
```
openssl x509 -inform der -in cacert.der -out burp.pem
name=$(openssl x509 -inform PEM -subject_hash_old -in burp.pem | head -1)
echo "$name"
cp burp.pem "$name.0"
```

Android uses the “subject_hash” as the filename for every CA; that is why we made a copy of the PEM file to a filename of a hash with a “.0” extension.

Now that we have a CA that Android can digest, we will need to place it within its system CA Cert folder. The problem now is that when Android boots it makes the “system” partition read only. This will prevent us from moving the cert to the correct directory. Even with root permissions we are not able to remount the “system” partition with “RW” permissions.

This is where Magisk modules come into play. Modules are able to modify the system partition before it has been booted up and converted to read-only. There is a module called “movecert” within the Magisk modules download section but it did not seem to work with the latest version (20.3) There is a pull request to fix it from a user named “fishso”, though I still could not get it to work based on the fact that the user certificates are not installed in the same place as these modules expect.

I modified their work to make a new module named “burpcert” that will just install the certificate authority you want and not move it from the user certs. For it to work on your device you will need to unzip the burpcert.zip file and replace the certificate within the “system/etc/security/cacerts” folder and then rezip the contents up. Make sure that you do not have any parent folders. The zip should look like the following:



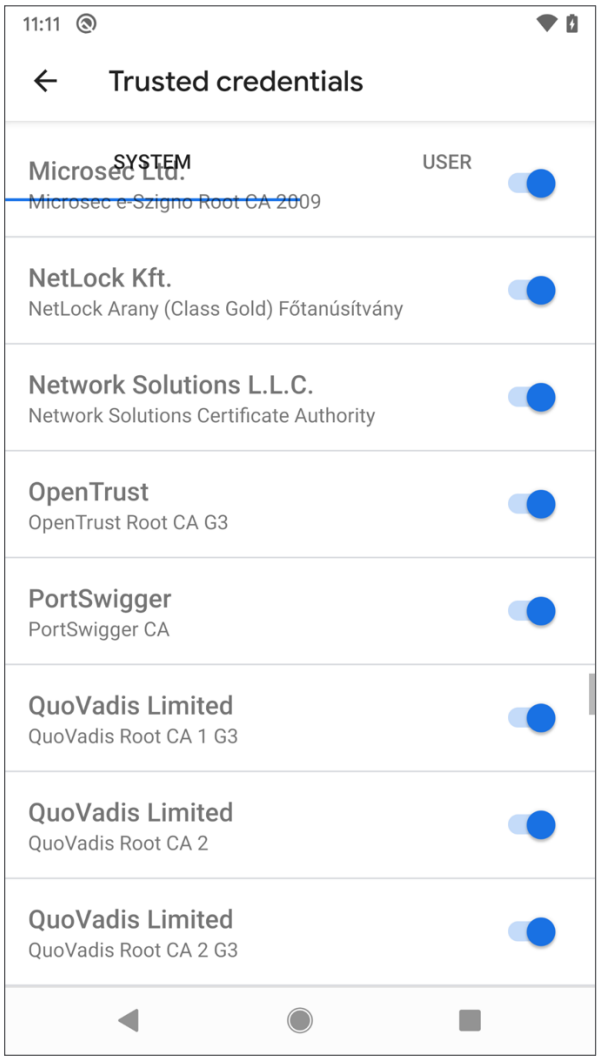
You will now need to install the module containing your custom CA through the Magisk Module Manager. The following video shows you how:

https://youtu.be/ULM9nYiQ9MA

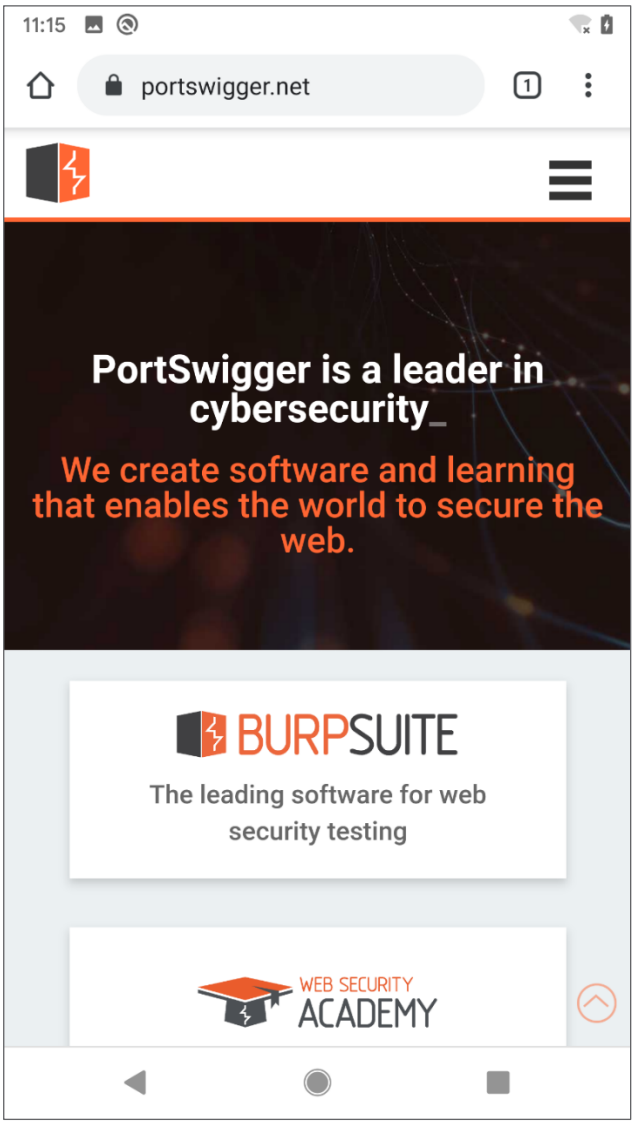
Once you have the rebooted the device you should be able to verify that the cert is properly installed by running the following command:

```
adb shell
ls /system/etc/security/cacerts/<your CA hash>.0
```

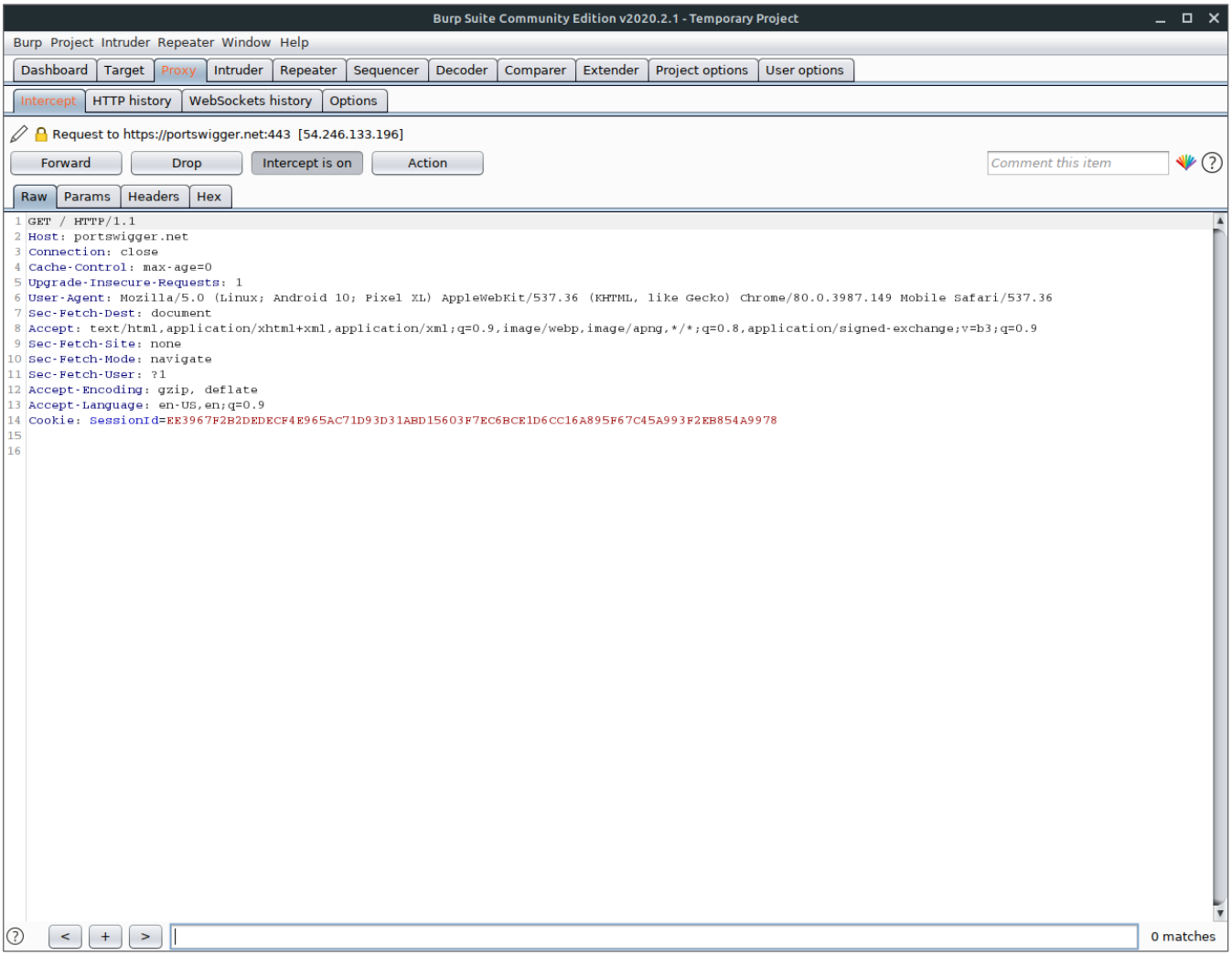
You should be able to see your certificate there, which means that you can also check that "Portswigger CA" exists in the "System" section of the "Trusted Credentials" list on your phone.



Retry a HTTPS site again using BurpProxy and you should no longer be warned that "Your connection is not private".



You should also see that your traffic is successfully being passed through BurpSuite decrypted.



Now that you have installed BurpSuite's CA Cert into your Android's system CA Cert directory you should be able to follow the same steps on my previous technique tutorial to fully unpin a certificate using Frida.

[Android SSL Pinning Bypass Android: 4-6]