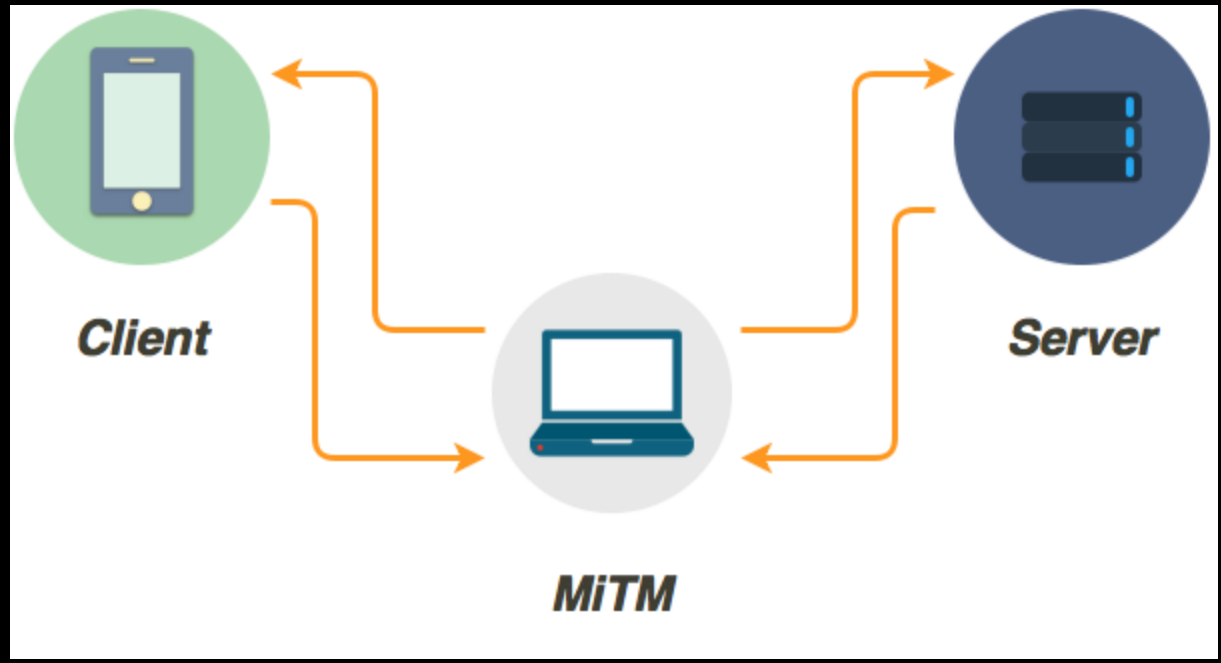
Android Mobile Pentest 101

Lecture 6 – SSL Pinning

Goal: Realize why you can’t perform dynamic analysis ☺

Description

- SSL Pinning is the process of associating a host with their expected X.509 certificate or public key. Once a certificate or public key is known or seen for a host, the certificate or public key is associated or 'pinned' to the host.
- Applications communicating over HTTPS and using Pinning makes it non-trivial to perform Man-In-The-Middle attack and grab the network traffic in clear text using the proxy tools.



Description

- We are going to discuss pinning
- The pinning strategy should be carefully designed as there are many trade-offs to consider
- There are three choices of pinning:

1. Certificate Pinning (We will mainly talk about this)
2. Public Key
3. Hashing

What To Pin -> Certificate

- Certificate is easiest to pin
- At runtime, you retrieve the website or server's certificate
- You compare the retrieved certificate with the certificate embedded within the application
- If the site/service rotates its certificate on a regular basis, then your application would need to be updated regularly

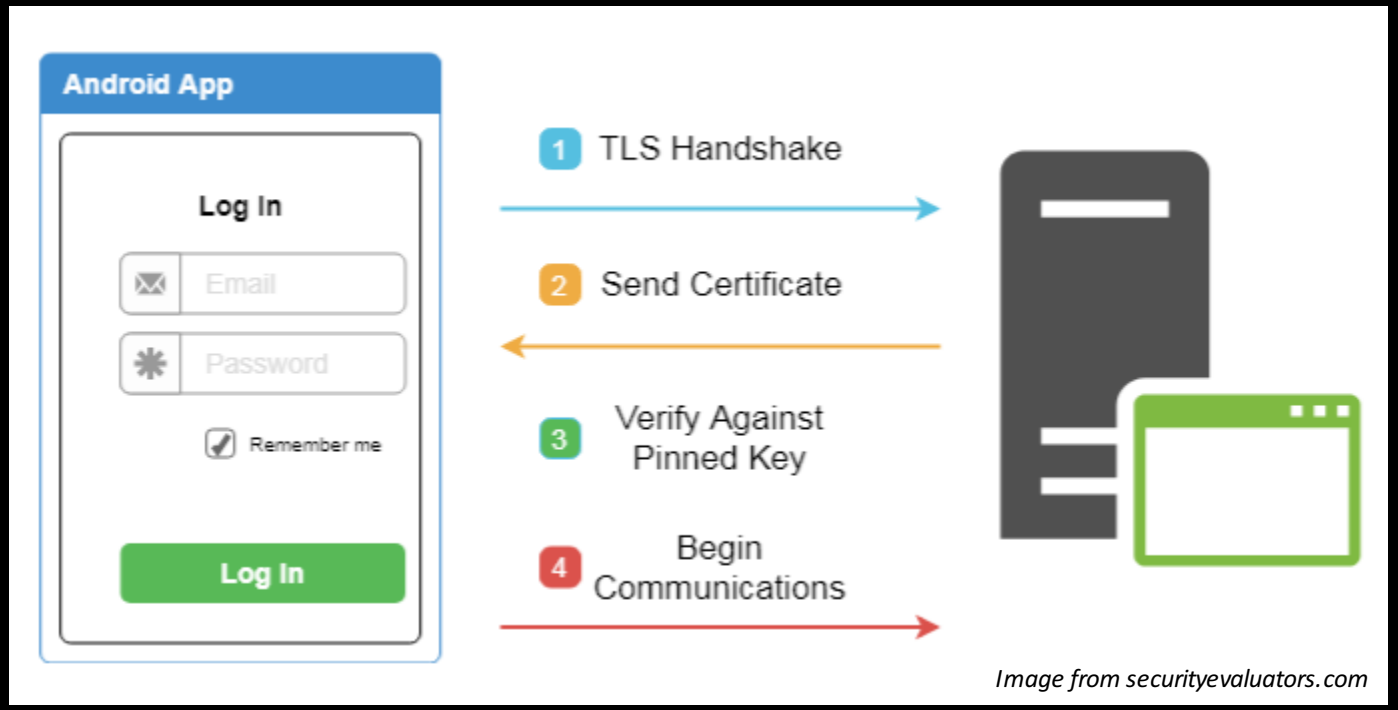
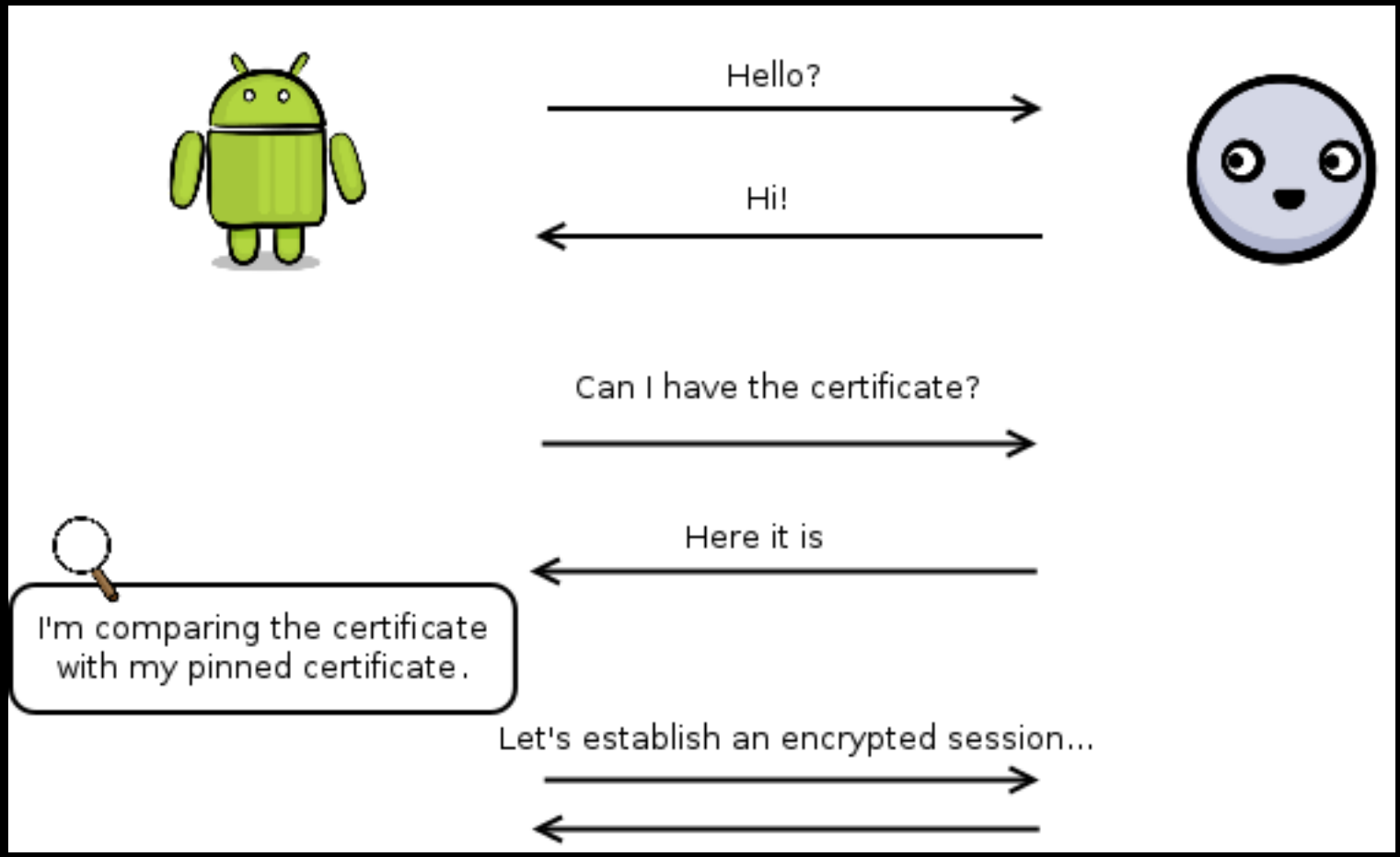


Image from securityevaluators.com

What To Pin -> Certificate

- Or more human language:



What To Pin -> Public Key

- More flexible
- A little trickier due to the extra steps necessary to extract the public key from a certificate
- Its harder to work with keys since you must extract the key from the certificate
- As with a certificate, the program checks the extracted public key with its embedded copy of the public key

What To Pin -> Hash

- Allows you to anonymize a certificate or public key
- A digested certificate fingerprint is often available as a native API for many libraries, so its convenient to use
- An organization might want to supply a reserve identity in case the primary identity is compromised

Next slide is about where to pin our cert, key
(just more information, you can skip ☺)

Where To Pin -> Leaf certificate

- Guarantees with close to 100% certainty that this is your certificate even if Root CA was compromised
- If the certificate becomes invalid for some reason (either normal expiration or compromise) the app will be bricked until you can push an update out
- Allows self-signed certificates – which can be a good thing from an ease of maintenance perspective

Where To Pin -> Root certificate

- By pinning against the root certificate you are trusting the root certificate authority as well as any intermediaries they trust not to mis-issue certificates
- If CA gets compromised it’s game over
- Very important to maintain strong certificate validation

Where To Pin -> Intermediate certificate

- By pinning against an intermediate certificate you are trusting that the intermediate certificate authority to not mis-issue a certificate for your server(s)
- As long as you stick to the same certificate provider then any changes to your leaf certificates will work without having to update your app

More Information

- A common misconception about certificate pinning is that it prevents the user from viewing client-server communications
- OWASP's page on Certificate and Public Key Pinning[1] reads:

"You should pin anytime you want to be relatively certain of the remote host's identity or when operating in a hostile environment. Since one or both are almost always true, you should probably pin all the time."

- With SSL Pinning, we cannot simply intercept request send to server from our device because the self-signed certificate generated by tools such as Burp won't have a valid trust chain, and if the certificate can't be verified as trusted, most mobile apps will terminate the connection instead of connecting over a potentially insecure channel. ☹

Detect Pinning

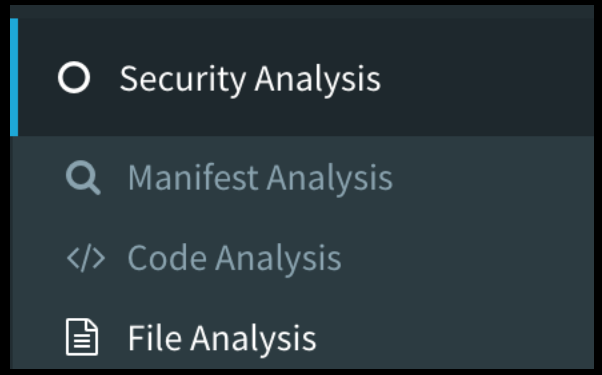
- To check if SSL Pinning is implemented or not, there are some ways

1. You will be able to intercept the first request and not other request (or can't intercept request)
2. Search for strings "Trusted"

- Use MobSF

(Left menu) Security Analysis tab -> File Analysis.

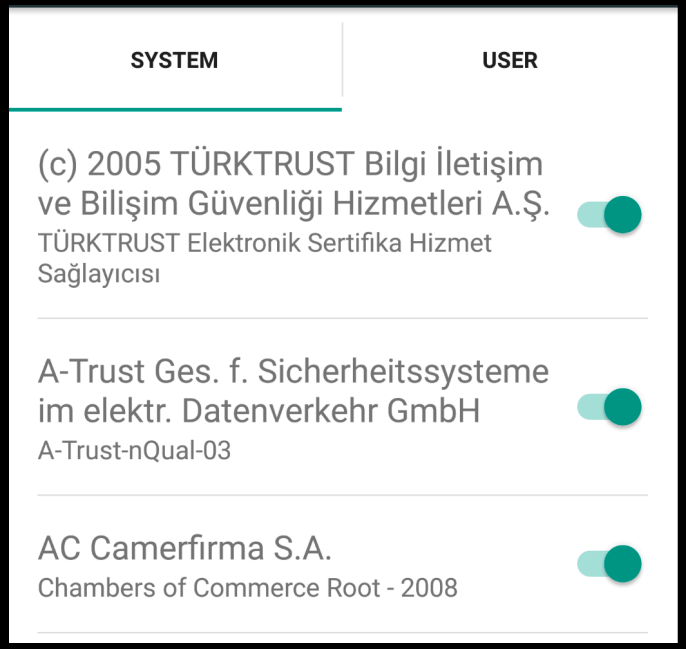
If you find "Certificate/Key Files Hard-coded inside the App" or something contains the word "Pinning" => Pinning



Bypass The Pinning

Method 1: Adding a Custom CA to the User Certificate Store

- By default, secure connections (using protocols like TLS and HTTPS) from all apps trust the pre-installed system CAs, and apps targeting Android 6.0 (API level 23) and lower also trust the user-added CA store by default

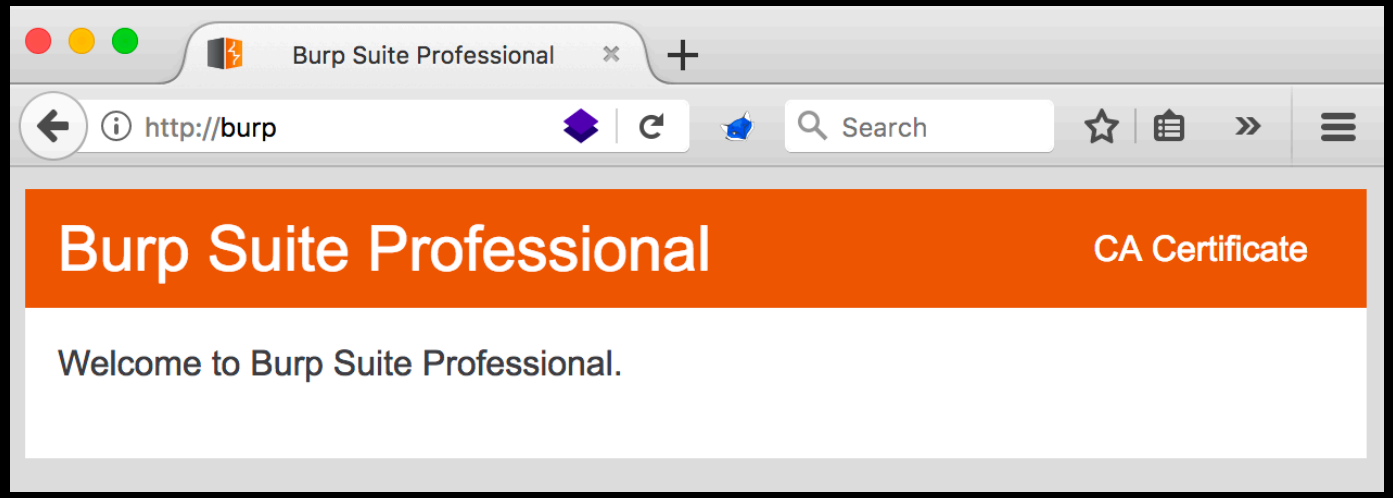


- If we have valid trusted certificate, we pass

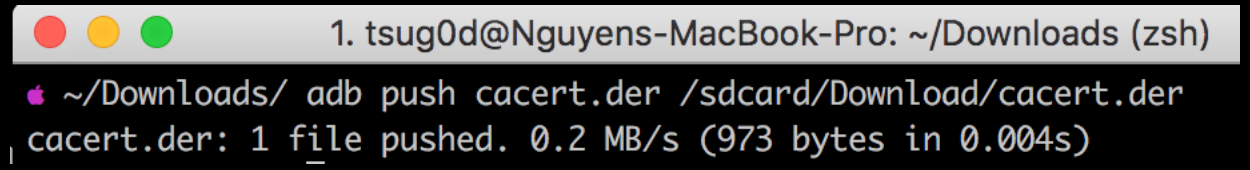
⇒ Let add our custom CA

Bypass The Pinning

- First we get the burp certificate (set the browser proxy via burp) by accessing http://burp



- Then push it into our virtual device:

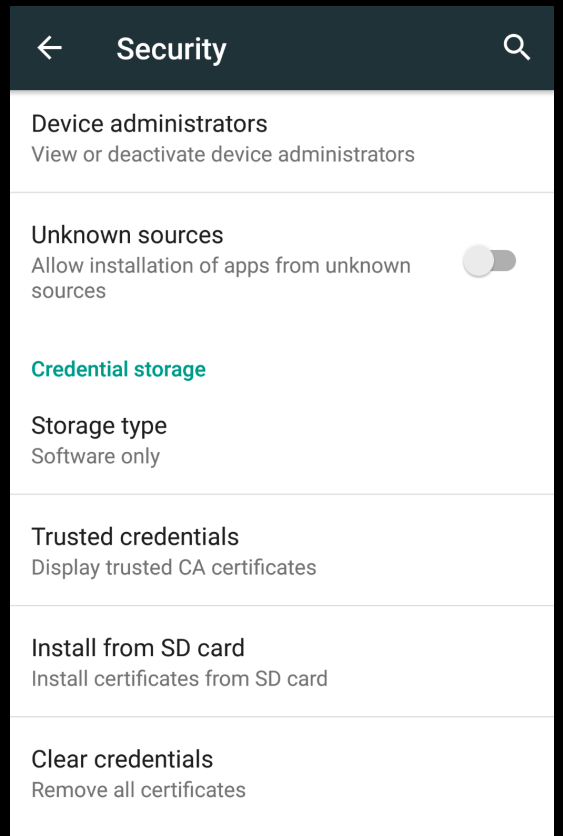


Bypass The Pinning

- Android cert doesn't understand .der extension, so we have to change it to .cer

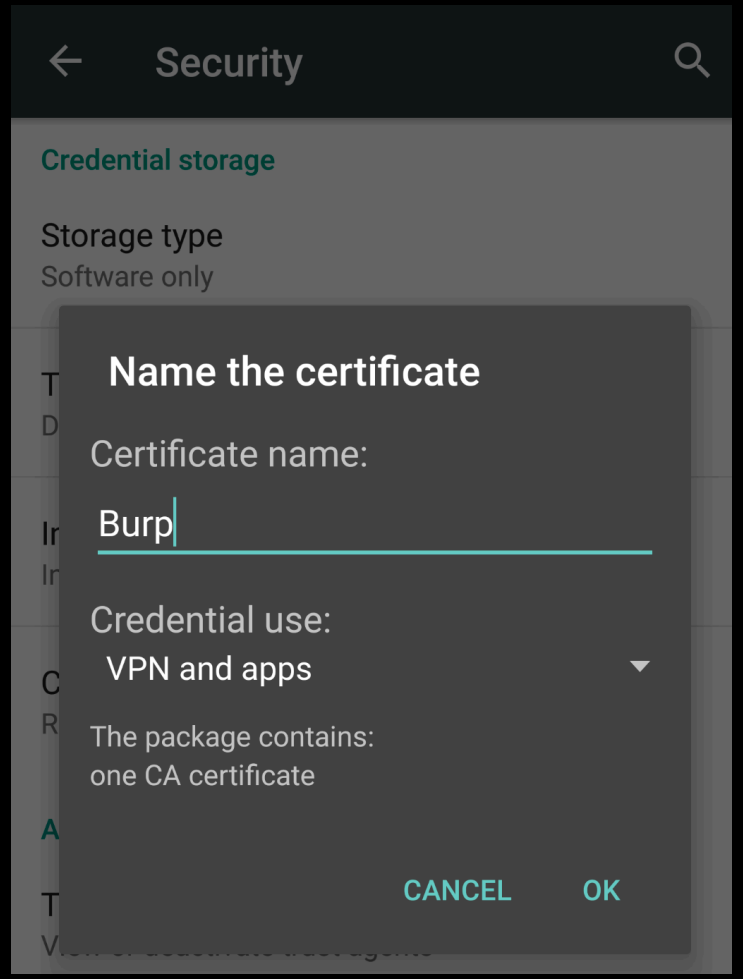
```
1. adb shell (adb)
 ~/Downloads/ adb shell
root@vbox86p:/ # cd sdcard/Download/
root@vbox86p:/sdcard/Download # mv cacert.der cacert.cer
```

- Now we go to Settings -> Security -> Install from SD card



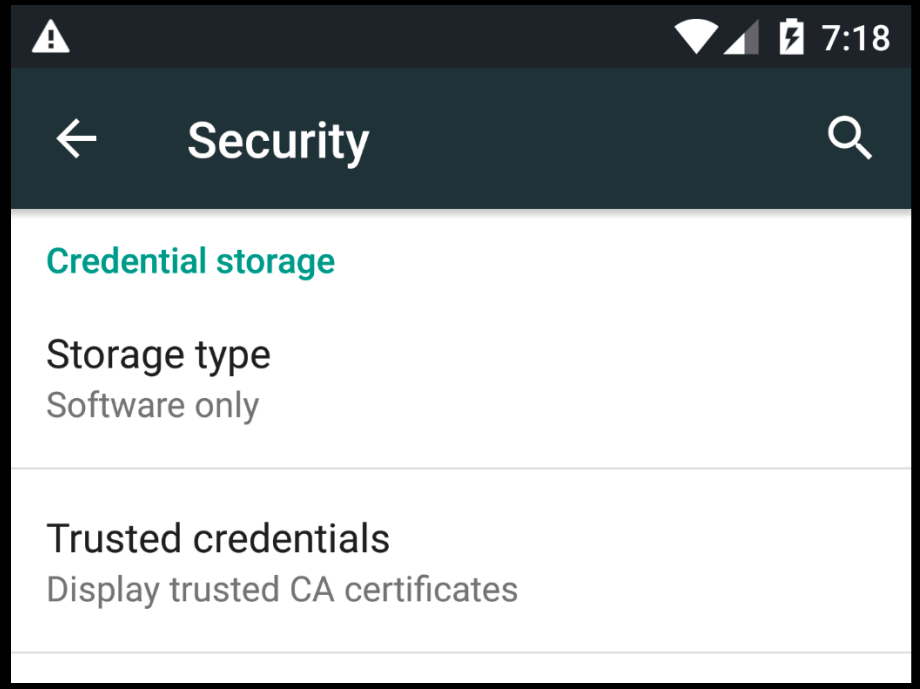
Bypass The Pinning

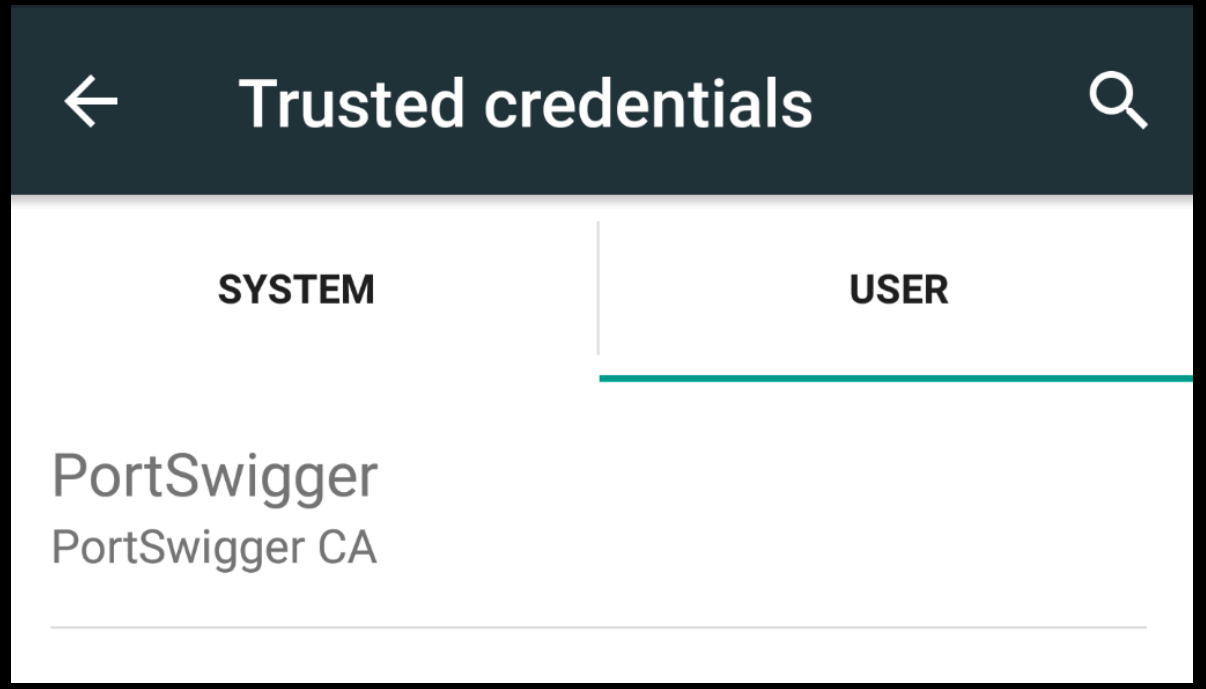
- Come to /sdcard/Download/ folder, we see our pushed cert, click it
- Then fill like below and click OK



Bypass The Pinning

- Now go to Trusted credentials and check if our cert is installed





- So, ssl request is proxied via burpsuite using PortSwigger CA, which is valid cert installed, we pass

Bypass The Pinning

- Sometime, the developer force us using android 7 to avoid this trick. It's defined in AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.pinning_app.app"  platformBuildVersionCode="25"
platformBuildVersionName="7.0">
```

- So we decompile it using apktool, change to:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.pinning_app.app"  platformBuildVersionCode="23"
platformBuildVersionName="6.0">
```

- Then recompile it, if the application is only validating on valid certificate, this technique should allow you to establish a successful MITM condition.

Bypass The Pinning

Method 2: Overwrite Packaged CA Certificate with Custom CA Certificate

- Sometime when decompile the apk, you will find some CA in the directory

- Just Overwrite the 'CustomCA.cer' certificate with our custom CA should allow us to trick the application into accepting our certificate.

Bypass The Pinning

Method 3: Patch the app

- Yes, patch the app, decompile the apk, find CertPinning smali code, change it ☺
- Some real case study:

1. Bypassing OkHTTP3 Certificate Pinning

https://blog.securityevaluators.com/bypassing-okhttp3-certificate-pinning-c68a872ca9c8

2. Bypassing certificate pinning/hardcoded ssl certificate/certificate pinning

https://www.youtube.com/watch?v=uEndLXB4tfA

3. Facebook Bypassing Certificate Pinning

https://blog.dewhurstsecurity.com/2015/11/10/mobile-security-certificate-pining.html

Bypass The Pinning

Method 4: Hook

- Frida time, talk in another lecture ☺