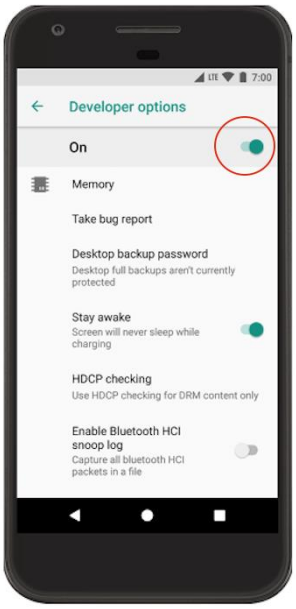
FRIDA

Credits: https://br.linkedin.com/in/atjunior
Frida Code: https://codeshare.frida.re/@sowdust/

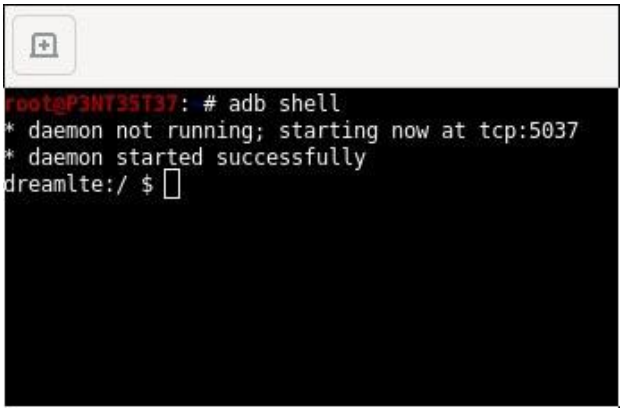
When i started making pentesting on android applications, i had problems on intercepting requests with burp suite and other proxies.
So i've started to search more about SSL-Pinning.
I made this tutorial using an Android Phone with ROOT and a Kali Linux Machine.
Let's start
First of all lets install ADB, Frida-Tools and Burp Suite on our machine.
To install Frida-Tools we will need pip installed. To install pip: Read

```
>>  apt-get install adb
>>  pip install frida-tools
```

After make this, we need to enable developer options on the phone:



With this option enabled, we conect our phone on the machine.
We go to our terminal, type ***adb shell*** Here is where we will make the conection with the phone, our phone will make an confirmation request, and we accept it.



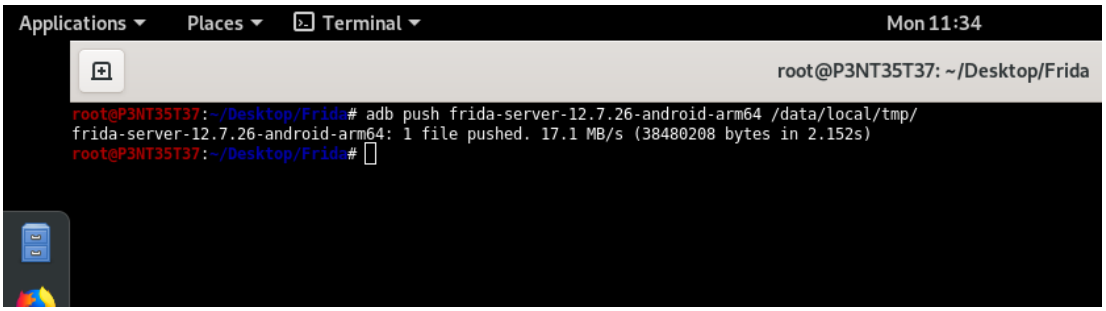
Now we need to know the architecture of our phone because we will need to send the frida binary to change it into a "Android Server". To get the binary: Github

Mine: ***Frida-server-12.7.26-android-arm64***

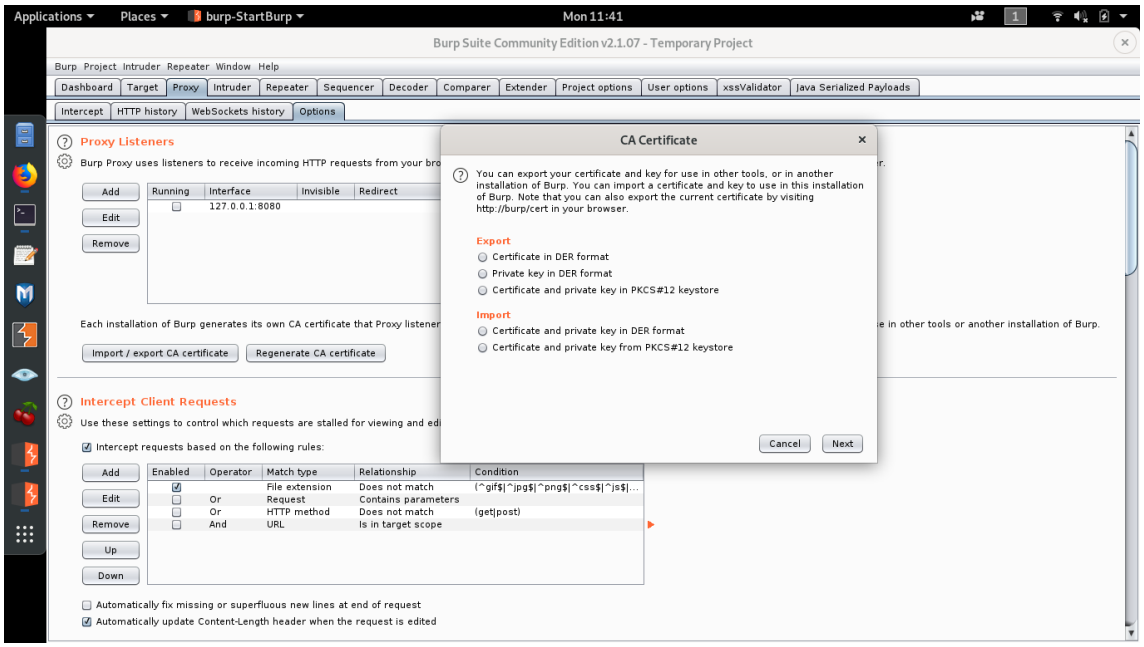
We need to send the frida binary using adb:

```
>> adb push frida-server-12.7.26-android-arm64 /data/local/tmp/
```

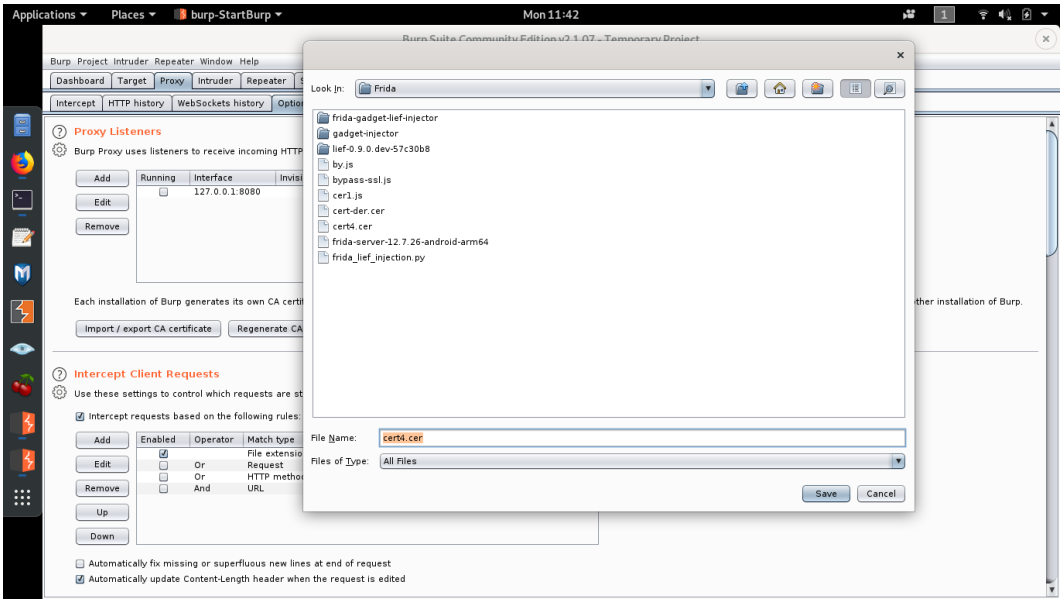
/data/local/temp/ → is the directory of androd to copy



Now we need to generate the Certificate of BurpSuite and send it to the same directory of frida binary file.
On burp: We go on tab proxy > Tab options > Export > Certificate in DER format

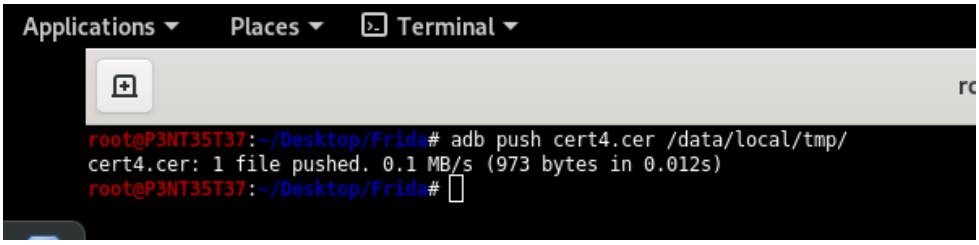


Now we need to change certificate extension from .der to .cer
I've created an certificate called cert4.cer, we can save it any place to send it to android phone.



Now we send it to the same directory of frida server on Android

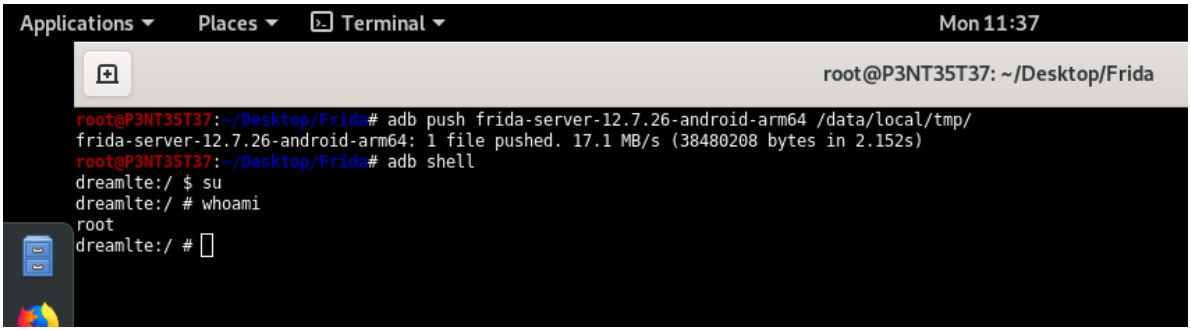
```
>> adb push cert4.cer /data/local/tmp/
```



In terminal we'll run the binary we sent. So:

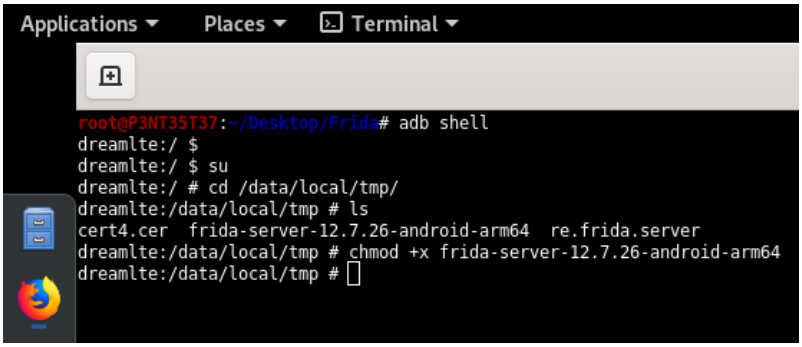
```
>> adb shell
>> su
```

Su → to root privileges



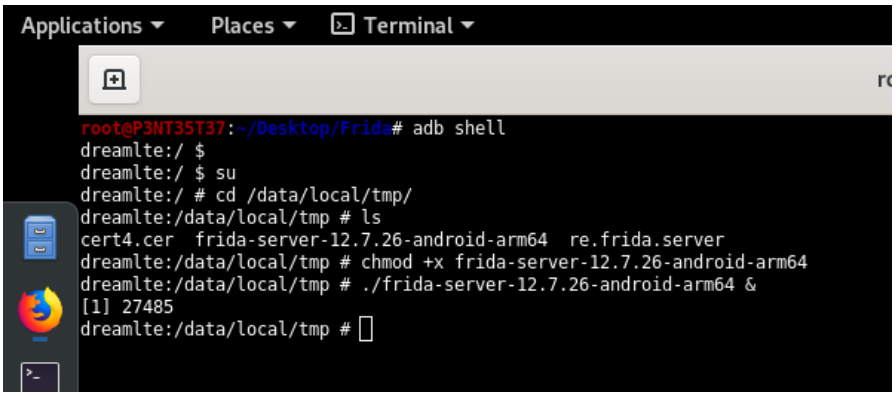
Now we need to change the privileges of binary execution

```
>> chmod +x frida-server-12.7.26-android-arm64
```



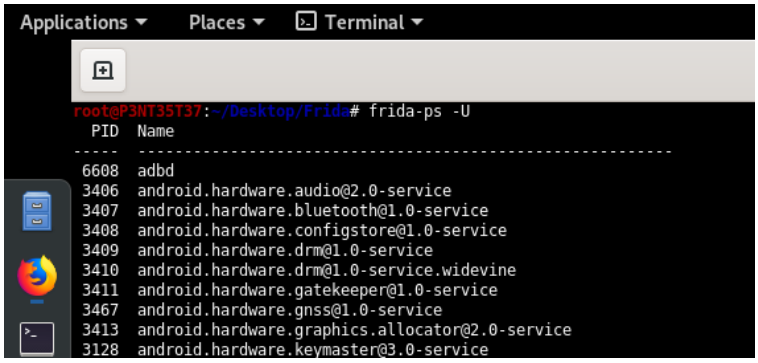
Lets run it:

```
>> ./frida-server-12.7.26-android-arm64 &
```



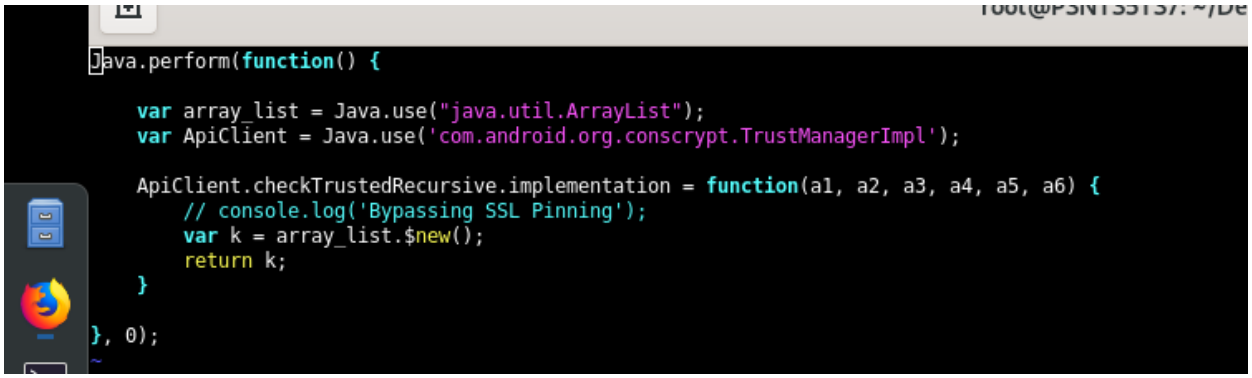
Now we need to see if everything is ok to start
So we try to list android processes. Type:

```
>> frida-ps -U
```



So we have frida and frida server running. Lets configure our proxy on android and burp.
Create an .js to by-pass ssl-pinning

```
*/
Java.perform(function() {
var array_list = Java.use("java.util.ArrayList");
var ApiClient = Java.use('com.android.org.conscrypt.TrustManagerImpl');
ApiClient.checkTrustedRecursive.implementation = function(a1, a2, a3, a4, a5, a6) {
// console.log('Bypassing SSL Pinning');
var k = array_list.$new();
return k;
}
}, 0);
```



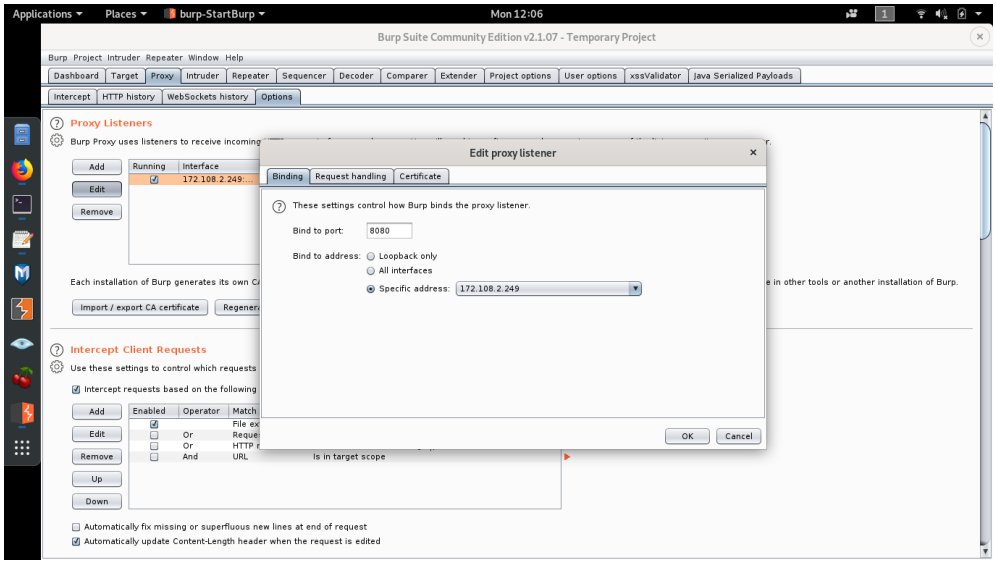
```
Java.perform(function() {

    var array_list = Java.use("java.util.ArrayList");
    var ApiClient = Java.use('com.android.org.conscrypt.TrustManagerImpl');

    ApiClient.checkTrustedRecursive.implementation = function(a1, a2, a3, a4, a5, a6) {
        // console.log('Bypassing SSL Pinning');
        var k = array_list.$new();
        return k;
    }

}, 0);
~
```

Recaping: we have burp, adb, pip, frida-tools, frida-server and burp certificate (.cer)

Lets configure it

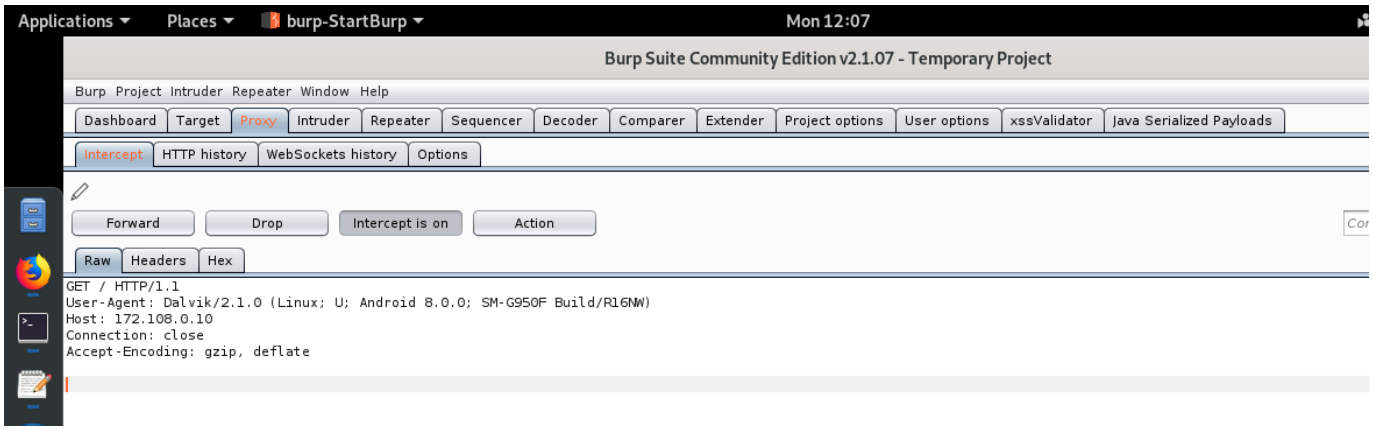
On android we go to connections > network management and we put our burp proxy:

Mine: **IP: 172.108.2.249 port: 8080**



Android and Burp gotta be in the same network

On burp we can see if the requests have been intercepted

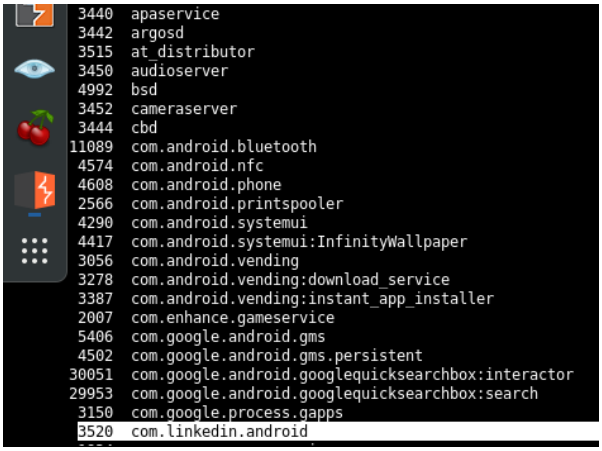


Now the communication is working, we will try to by-pass Linkedin SSL because we are not intercepting application requests. So we need to list the processes again and select the app we will try to find bugs.

Again:

```
>> frida-ps -U
```

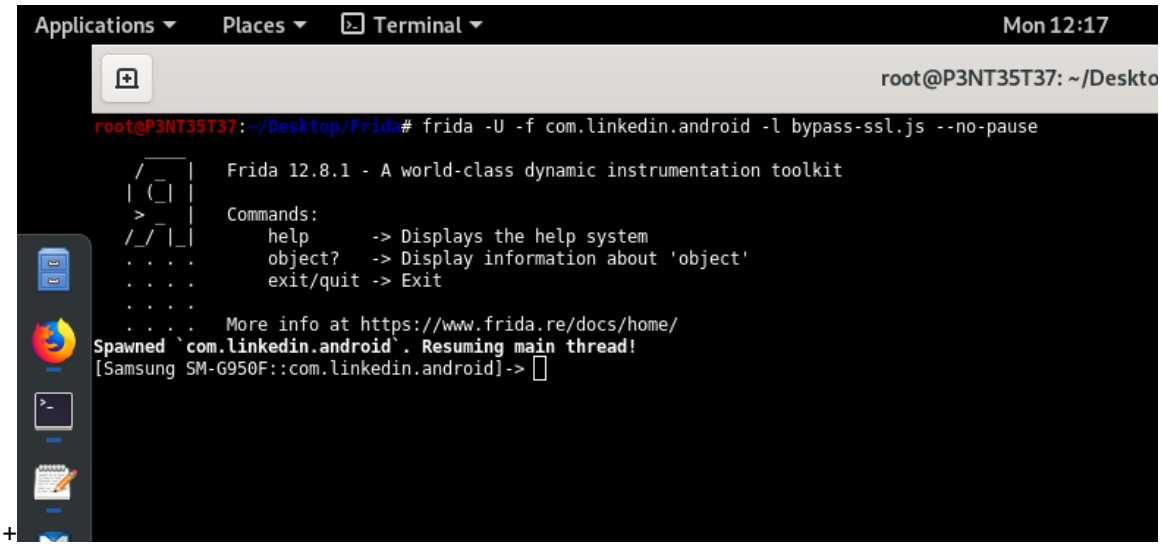
Now we locate Linkedin pid



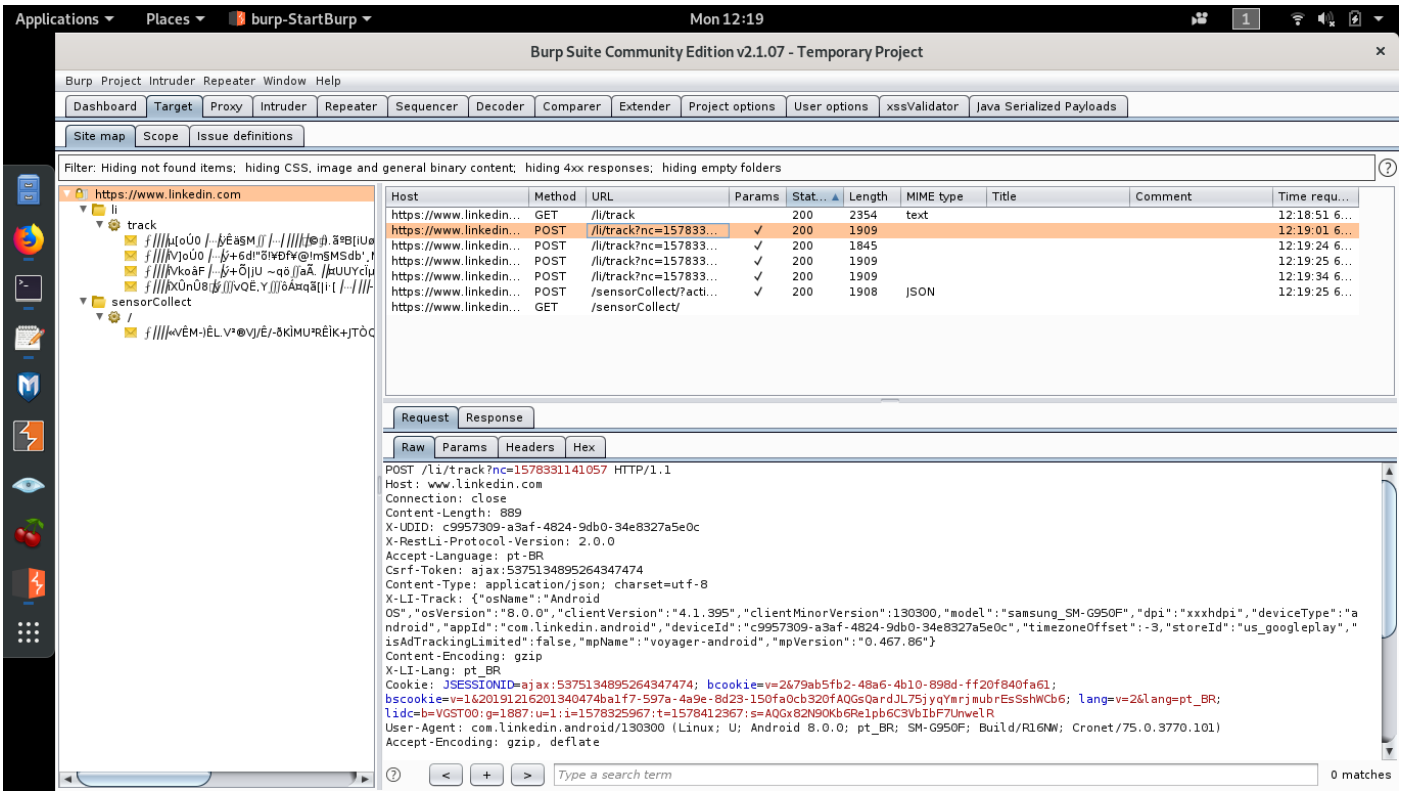
So we access the .js directory. Mine called: ***by-pass-ssl.js***

```
>> frida -U -f com.linkedin.android -l bypass-ssl.js --no-pause
```

Note: Android will close and open automatically the process of Linkedin. At this time we bypassed it



Now on burp we just need to see the requests of HTTPS that have been intercepted by Linkedin app.



Credits: https://br.linkedin.com/in/atjunior
Frida Code: https://codeshare.frida.re/@sowdust/