Seguridad y Legalidad: Máster en el IoT (UCM)

Práctica 2

Attify Pentest: OKLOK

Prof.: Joaquín Recas

Dispositivo OKLOK

OKLOK

深圳市龙兄弟数码锁有限公司 **Tools**

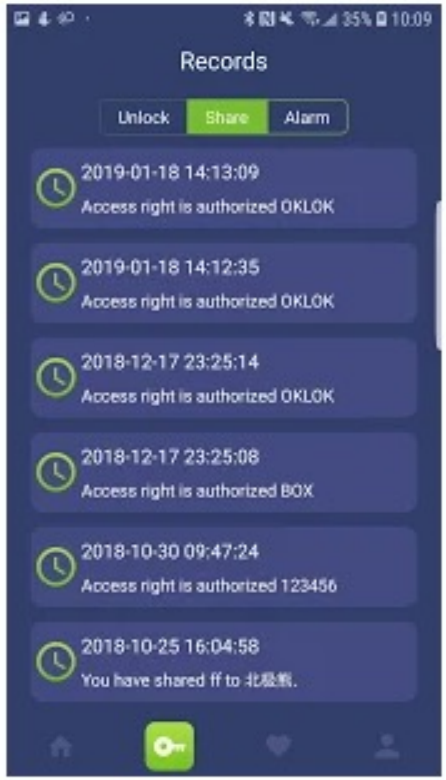
★★½☆☆ 186

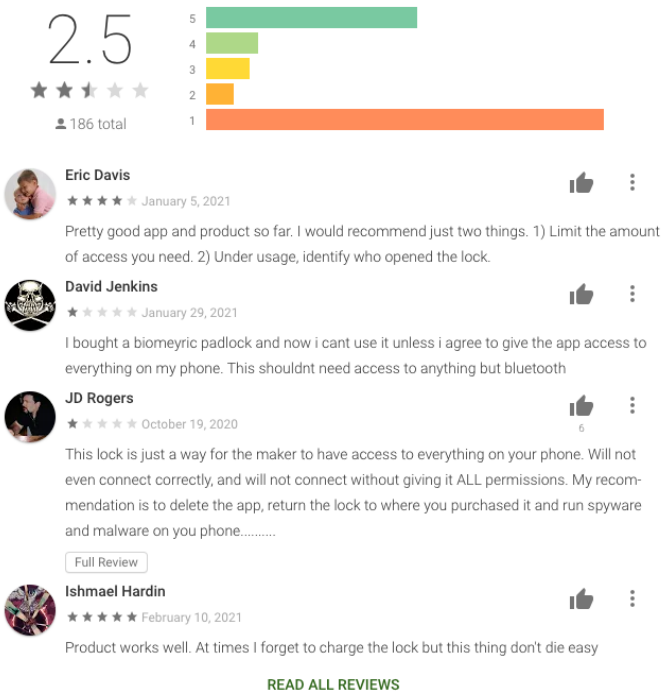
Everyone

This app is available for your device

You can share this with your family. Learn more about Family Library

Installed





OKLOK - This app has access to:

Device & app history
- retrieve running apps
- read your Web bookmarks and history

Identity
- find accounts on the device

Contacts
- find accounts on the device

Location
- approximate location (network-based)
- precise location (GPS and network-based)
- access extra location provider commands

Phone
- read phone status and identity

Photos/Media/Files
- read the contents of your USB storage
- modify or delete the contents of your USB storage

Storage
- read the contents of your USB storage
- modify or delete the contents of your USB storage

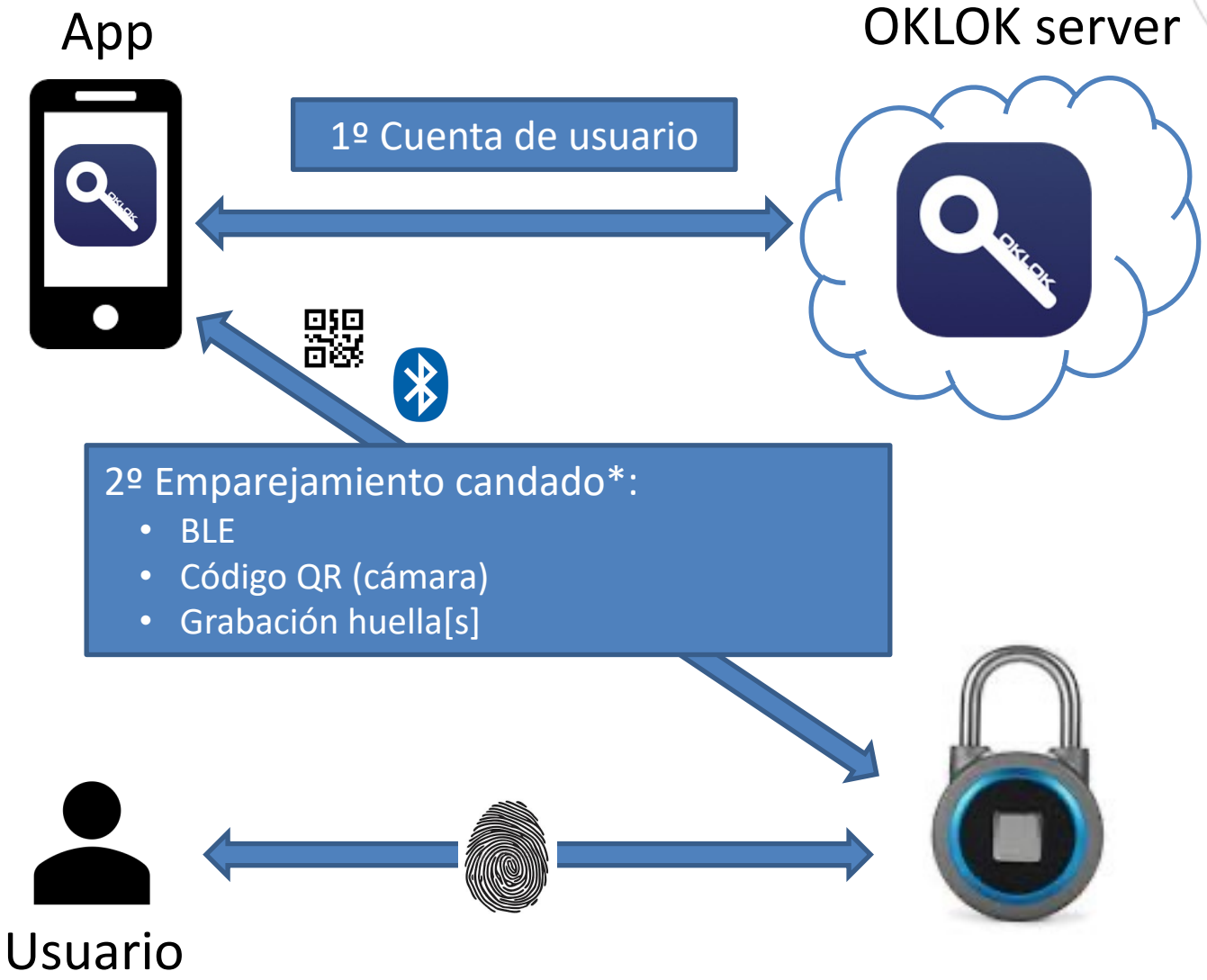
Camera
- take pictures and videos

Wi-Fi connection information
- view Wi-Fi connections

Device ID & call information
- read phone status and identity

Other
- download files without notification
- view network connections
- pair with Bluetooth devices
- access Bluetooth settings
- connect and disconnect from Wi-Fi
- full network access
- control Near Field Communication
- control vibration
- prevent device from sleeping

Funcionamiento

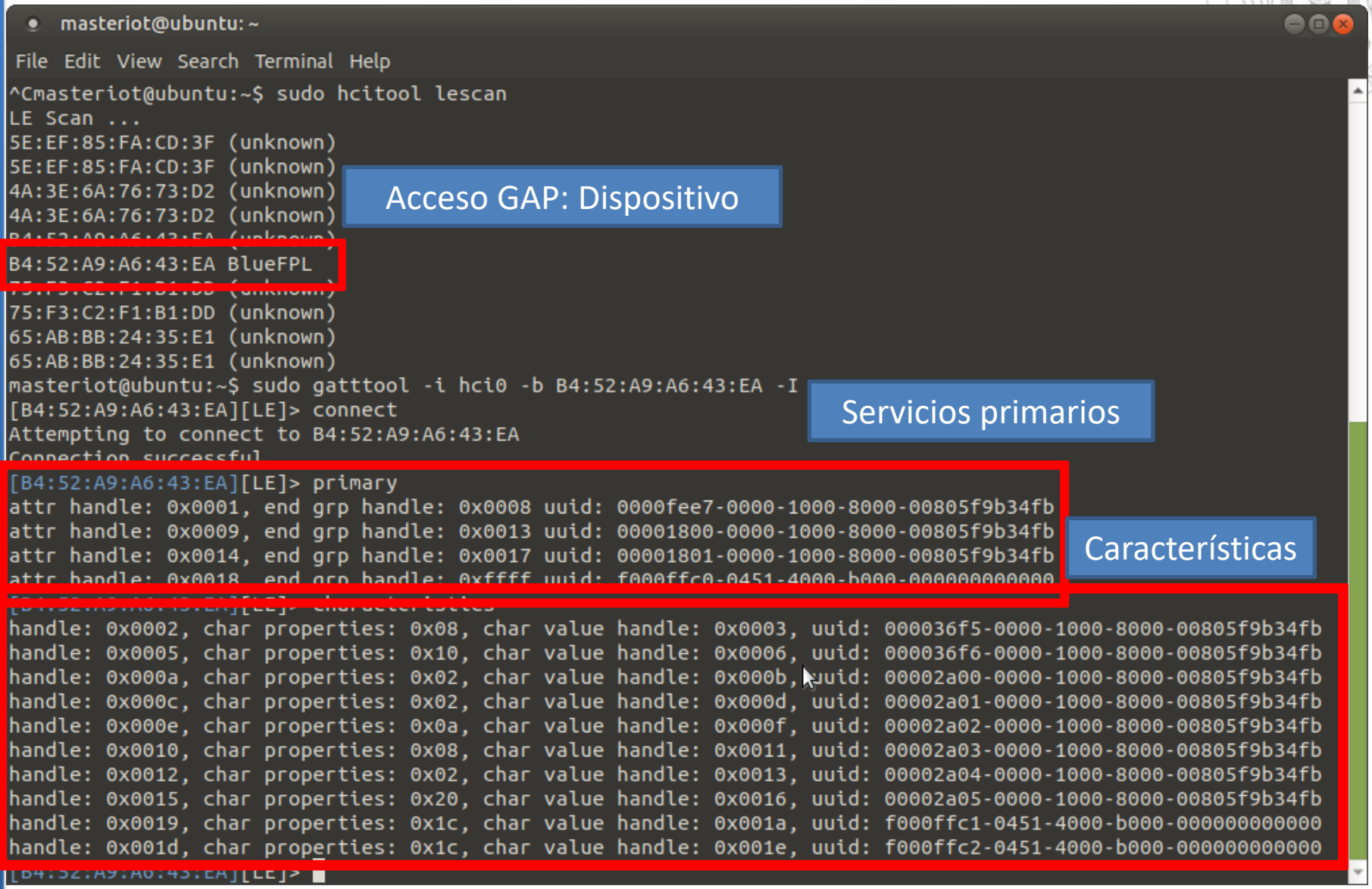


*El candado no puede estar enlazado con otra cuenta

OKLOK Pentest

- Averiguar qué mensaje BLE del teléfono desbloquea el candado y mandarlo usando otro dispositivo no autorizado
- Primeros pasos a seguir:
  1. Obtener la dirección física del candado
  2. Hacer una conexión con el dispositivo
  3. Ver la interfaz expuesta (servicios)
     I. Servicios
     II. Características
     III. Propiedades
     IV. Manejadores

BlueZ: hcitool & gatttool



Bettercap

- Bettercap: *the Swiss Army knife for WiFi, Bluetooth Low Energy, wireless HID hijacking and Ethernet networks reconnaissance and MITM attacks*.
- Permite ver manejadores y servicios de forma más *amigable:*

```
192.168.1.0/24 > 192.168.1.215 » ble.enum B4:52:A9:A6:43:EA
[02:28:47] [sys.log] [inf] ble.recon connecting to b4:52:a9:a6:43:ea ...
192.168.1.0/24 > 192.168.1.215 »
```

Handles	Service > Characteristics	Properties	Data
0001 -> 0008	fee7		
0003	36f5	WRITE	
0006	36f6	NOTIFY	
0009 -> 0013	Generic Access (1800)		
000b	Device Name (2a00)	READ	
000d	Appearance (2a01)	READ	Unknown
000f	Peripheral Privacy Flag (2a02)	READ, WRITE	Privacy Disabled
0011	Reconnection Address (2a03)	WRITE	
0013	Peripheral Preferred Connection Parameters (2a04)	READ	Connection Interval: 80 -> 160 Slave Latency: 0 Connection Supervision Timeout Multiplier: 1000
0014 -> 0017	Generic Attribute (1801)		
0016	Service Changed (2a05)	INDICATE	
0018 -> ffff	f000ffc004514000b000000000000000		
001a	f000ffc104514000b000000000000000	WRITE, NOTIFY	
001e	f000ffc204514000b000000000000000	WRITE, NOTIFY	

No estándar

OKLOK Pentest

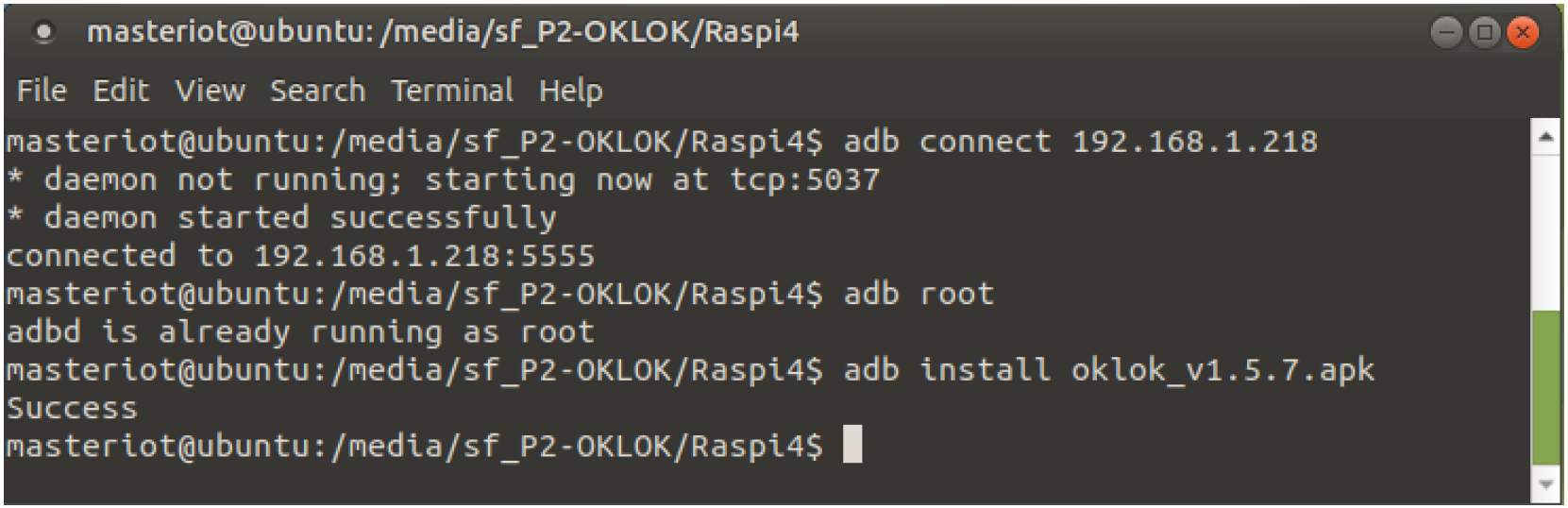
- Capturar y analizar el tráfico BLE:
  1. Usando HW/SW específico como ubertooth:
     - Open source wireless development platform suitable for Bluetooth experimentation.
  2. Usando el *Smartphone (rooted)*
  3. Usando máquina Virtual Androidx86
     - Acceso al dispositivo BT del Host
  4. Usando **Android en Raspberry Pi**
     1. Conectar teclado y ratón
     2. Hacer configuración inicial
     3. Poner en red con el PC
     4. Apagar ordenadamente (F5)

OKLOK Pentest

- Raspberry Pi Android 9 (LineageOS 16.0):
  - Soporte para BLE
  - Activar '*Developer options*':
    - Click 17 veces en: *Settings->About tablet->Build number*
    - *Dentro de Settings -> System -> Advanced -> Developer Options:*
      - Activar '*Bluetooth HCI snoop log*'
      - Habilitar acceso de *root* (RPi)
      - Activar *adb* por red (RPi)
      - Activar *Local Terminal*
  - Instalar App OKLOK
    - Play Store (no disponible en la RasPi)
    - Raspberry Pi mediante *abd* (Android Debug Bridge)

OKLOK Pentest

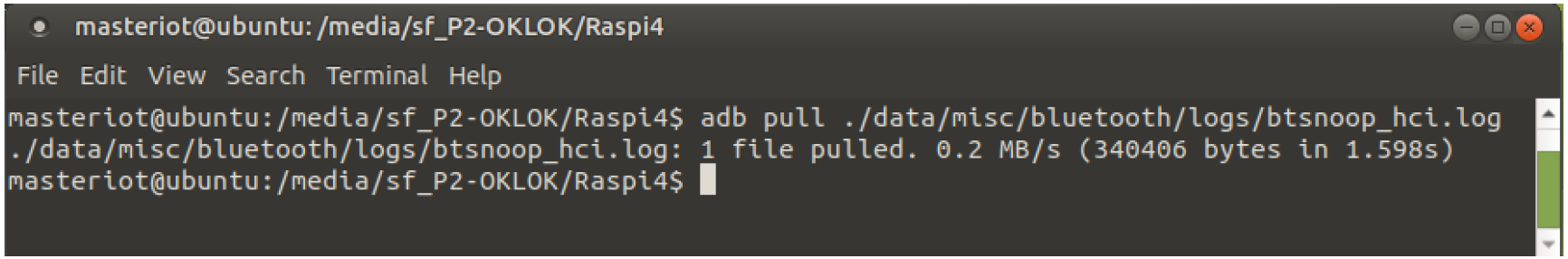
- Instalación mediante adb:



- Dar permiso a todo lo solicitado MUY IMPORTANTE
- Registrar cuenta, asociar candado, grabar huella.

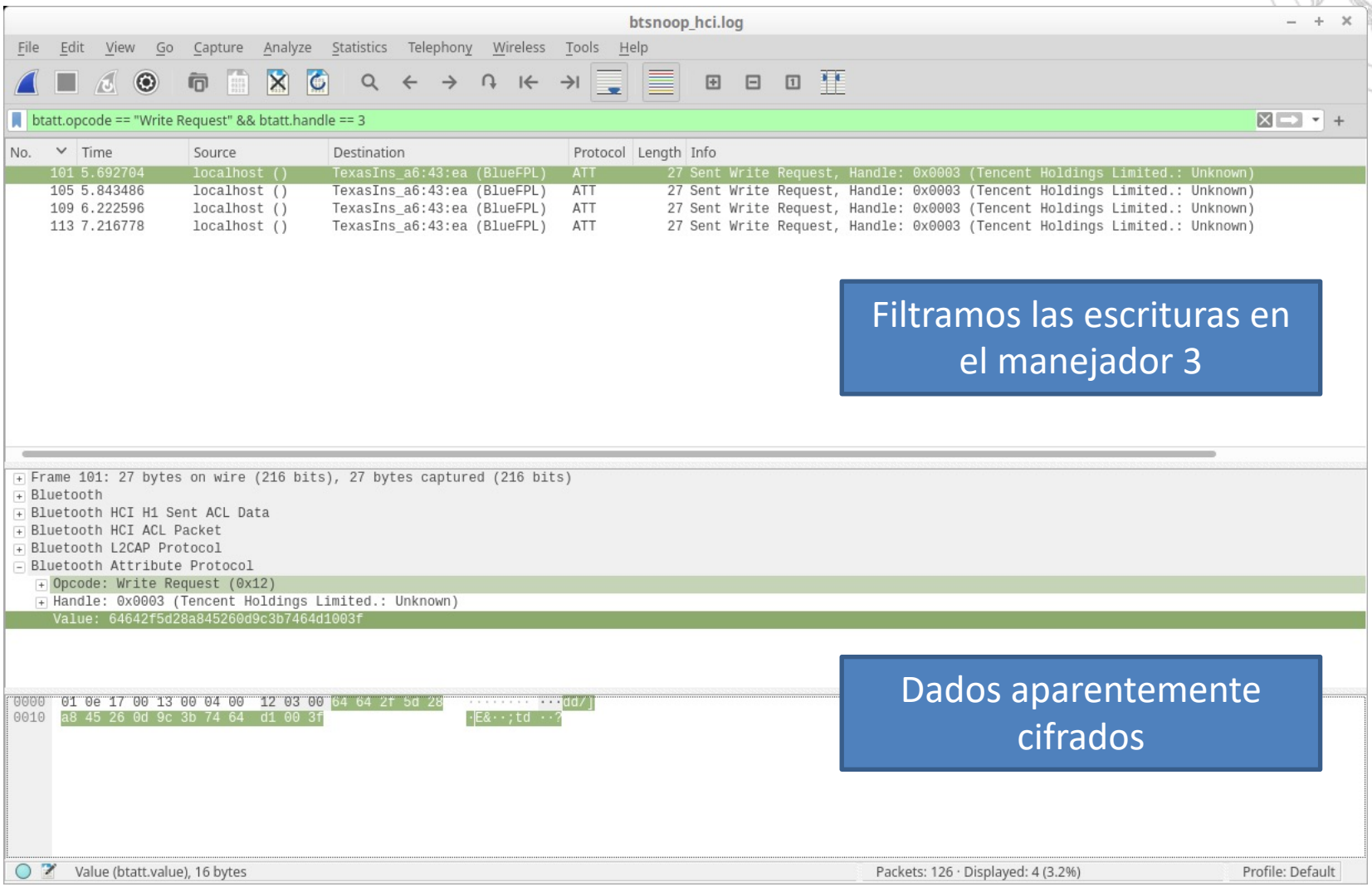
OKLOK Pentest

- Abrir el candado con la App para registrar HCI log
  - Tocar el sensor de huella para activar el candado.
- Abrir terminal linux en red con el dispositivo Android y obtener el fichero de *log* **BLE**:

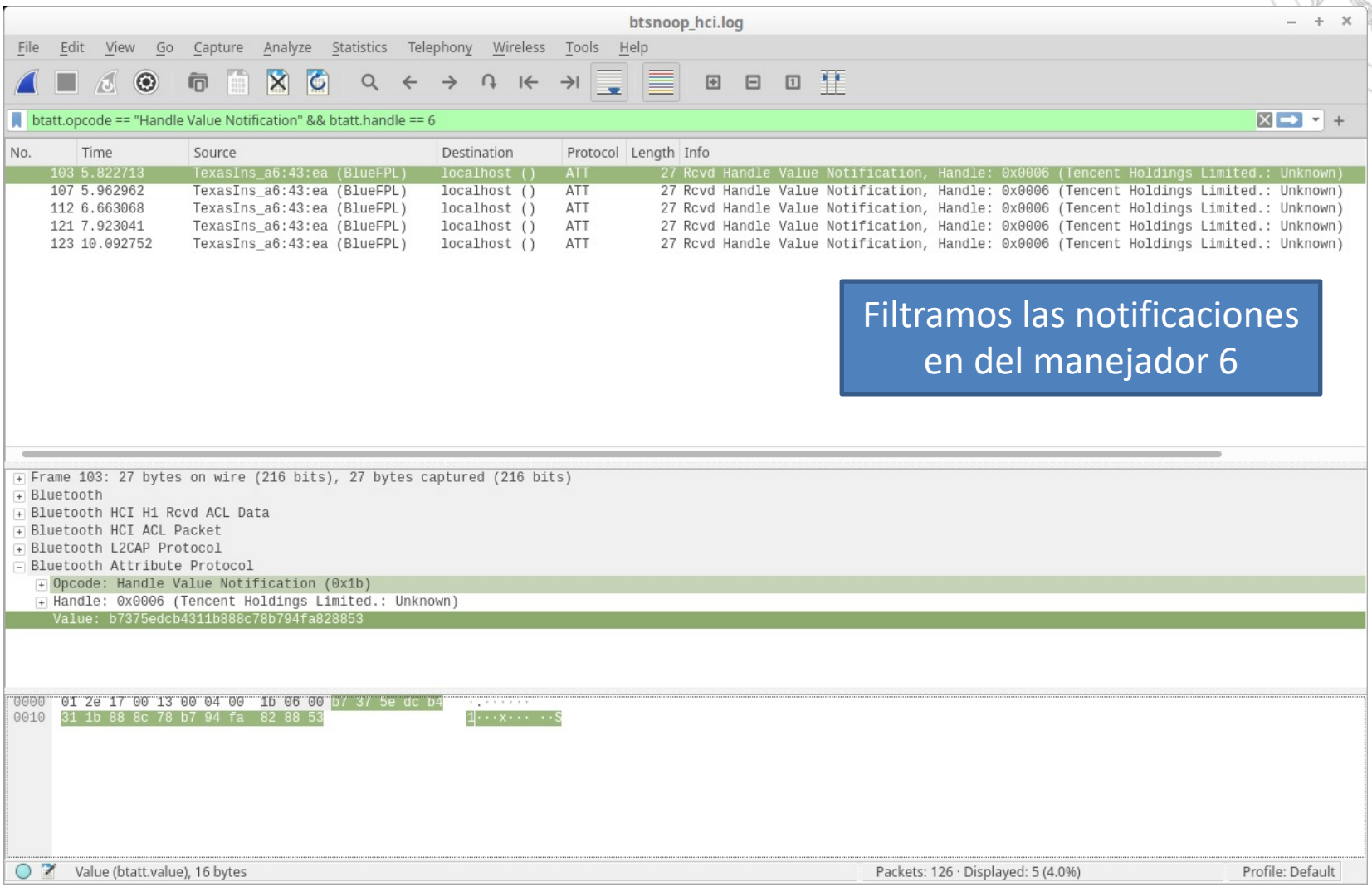


- Analizar el trafico usando

OKLOK Pentest



OKLOK Pentest



BLE Reply attack

Pese a que el candado recibe el paquete no se abre 😭 😭

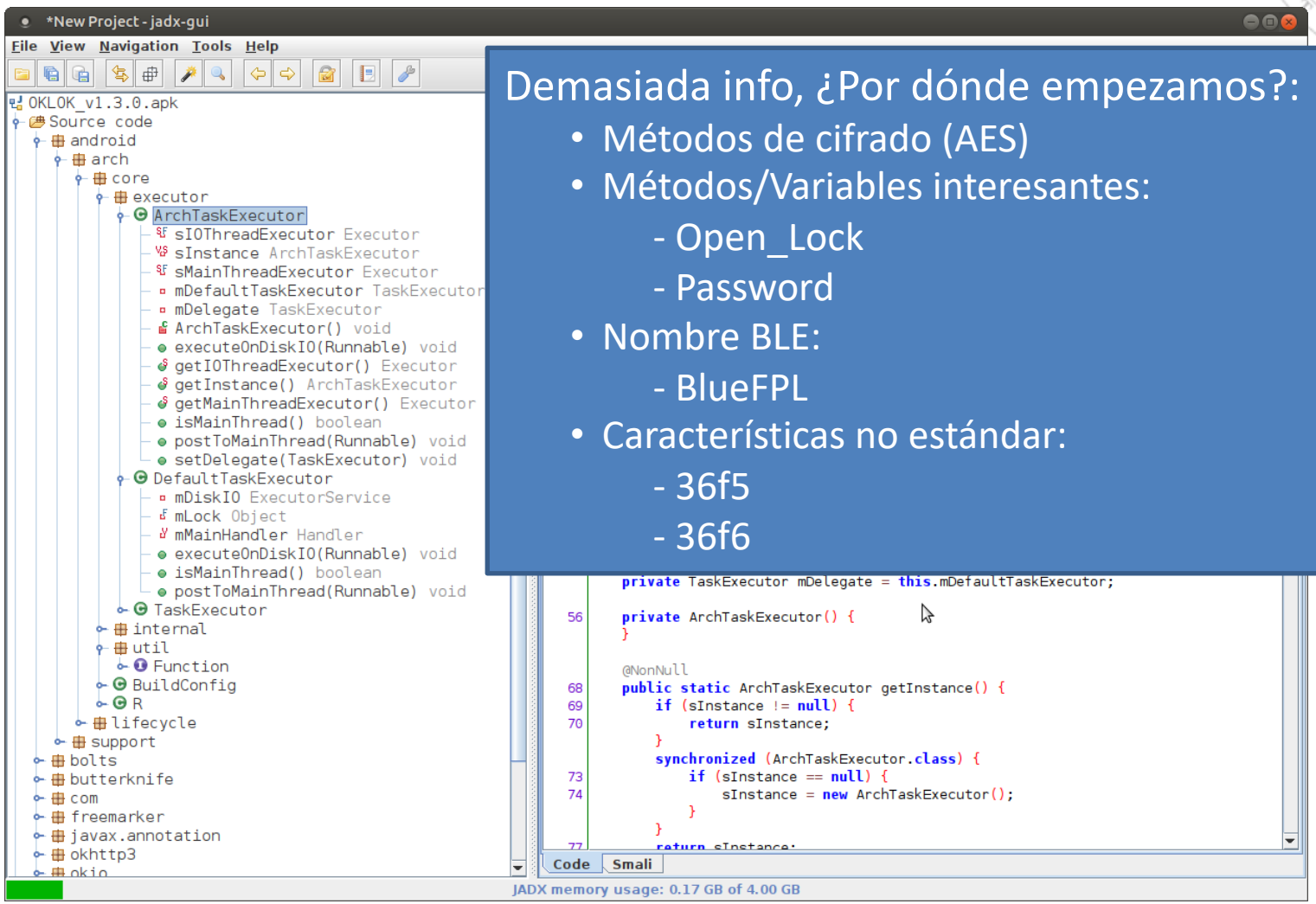
- Siguientes pasos:
  - Averiguar método de cifrado y cifra empleada
  - Averiguar protocolo de apertura

JADX

- Dex (Dalvik Executable) to Java decompiler
  - Command line and GUI tools for produce Java source code from Android **Dex** and **Apk** files
- Main features:
  - decompile Dalvik bytecode to java classes from **APK**, **dex**, **aar** and **zip** files
  - decode *AndroidManifest.xml* and other resources from *resources.arsc*
  - deobfuscator included
- jadx-gui features:
  - view **decompiled code** with highlighted syntax
  - jump to declaration
  - find usage
  - full text search

https://github.com/skylot/jadx

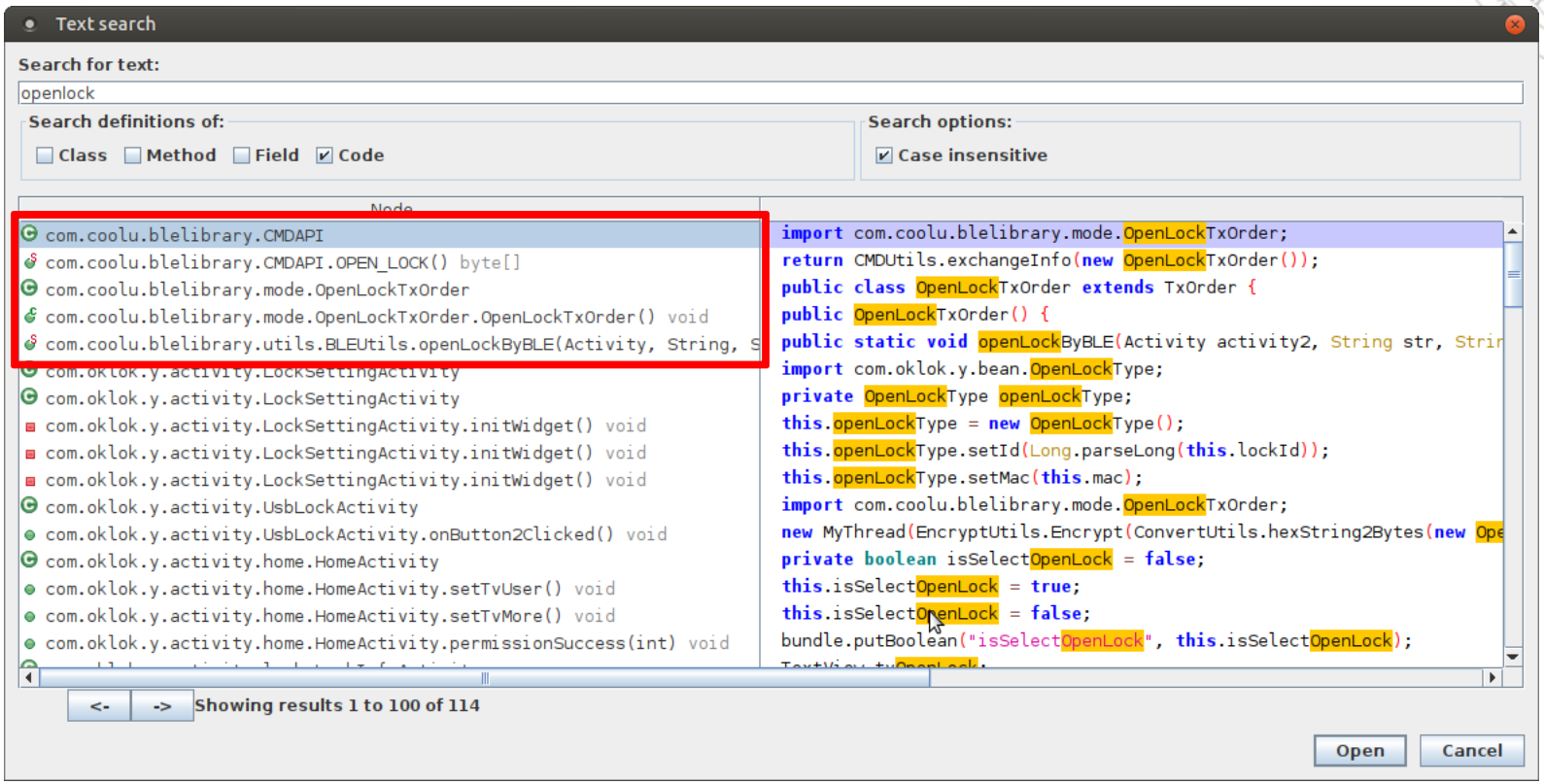
Jadx-gui: oklok_v1.3.0.apk



Demasiada info, ¿Por dónde empezamos?:

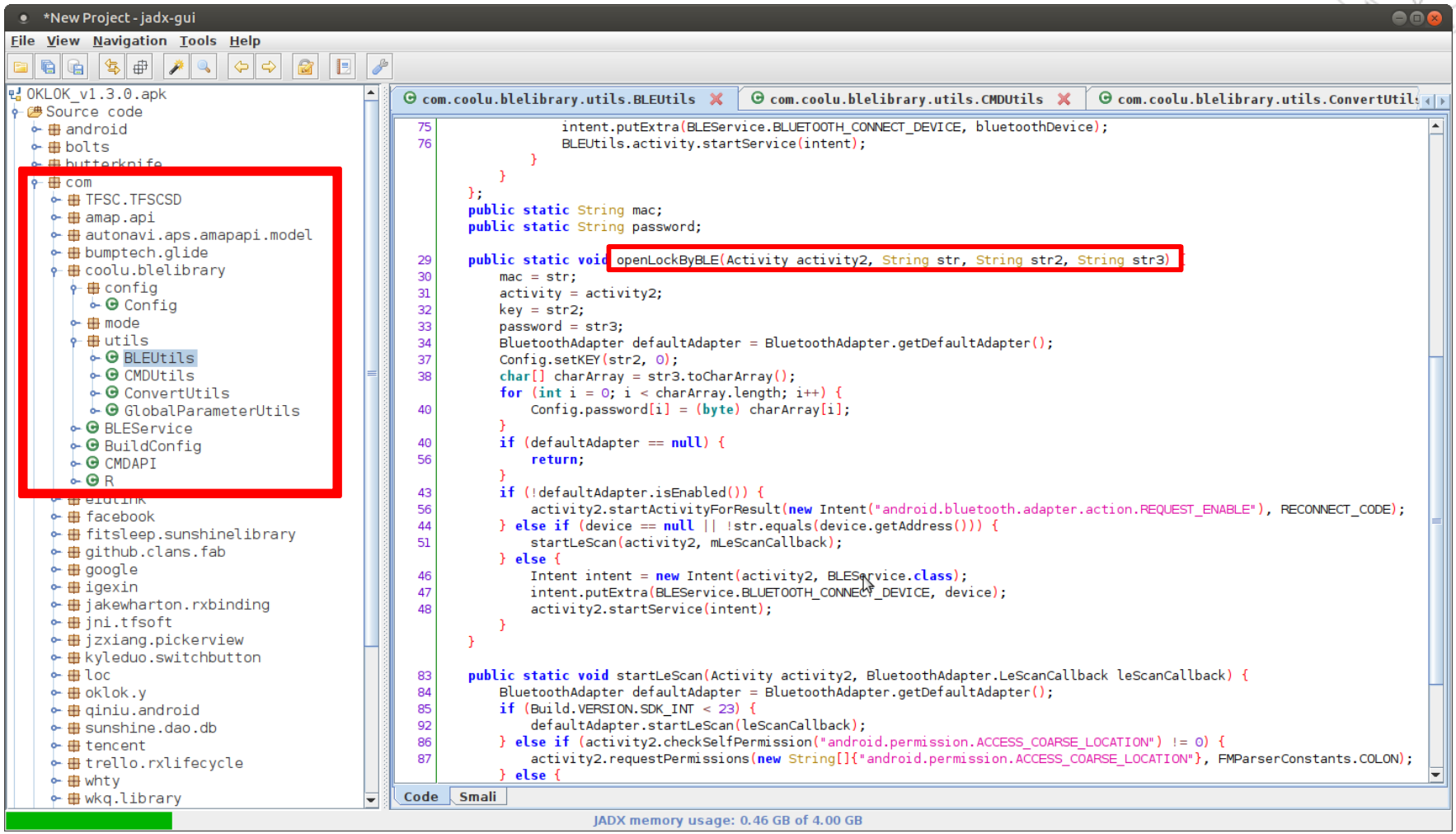
- Métodos de cifrado (AES)
- Métodos/Variables interesantes:
  - Open_Lock
  - Password
- Nombre BLE:
  - BlueFPL
- Características no estándar:
  - 36f5
  - 36f6

Jadx-gui: oklok _v1.3.0.apk

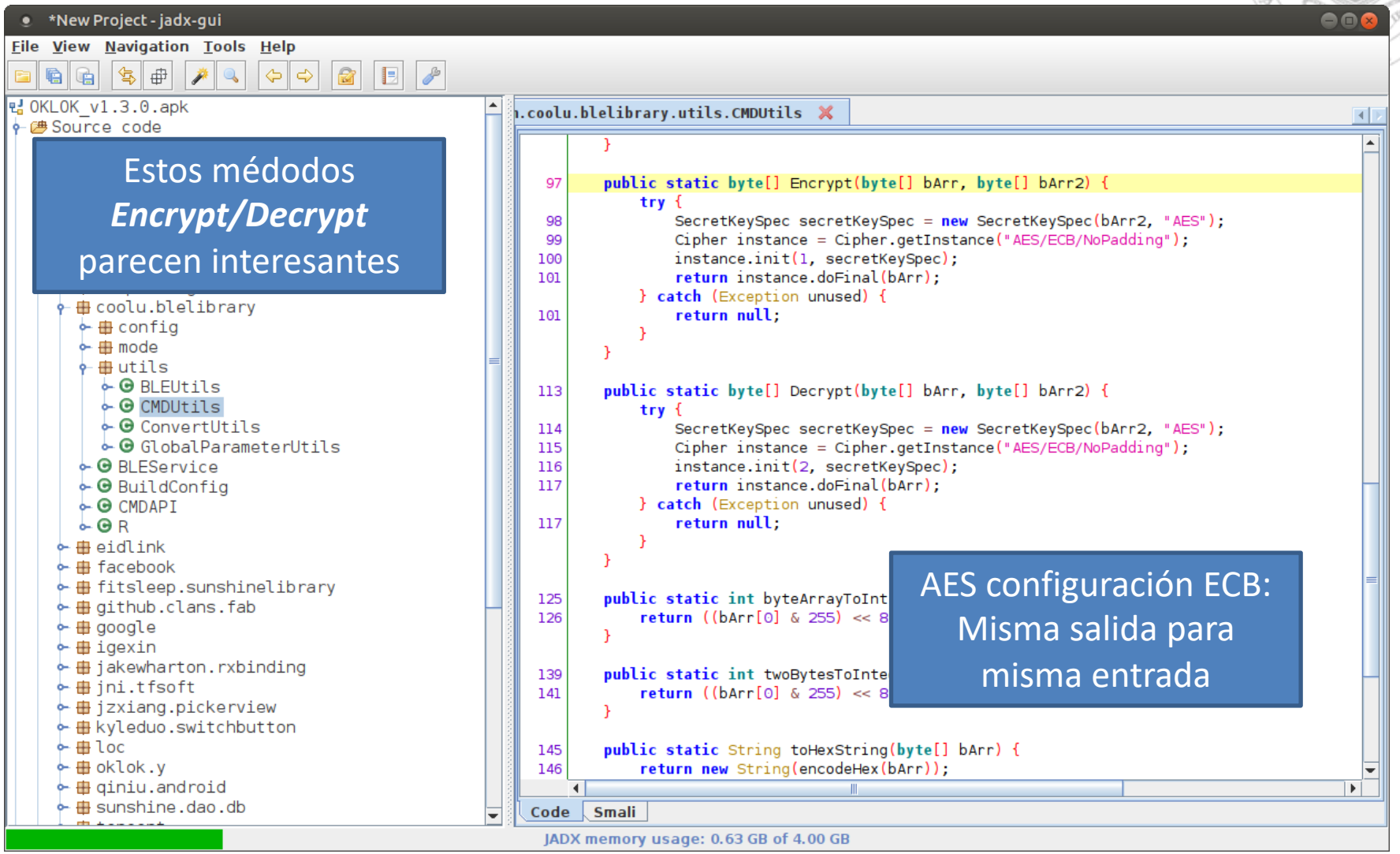


La clase *com.coolu.blelibrary* parece prometedora

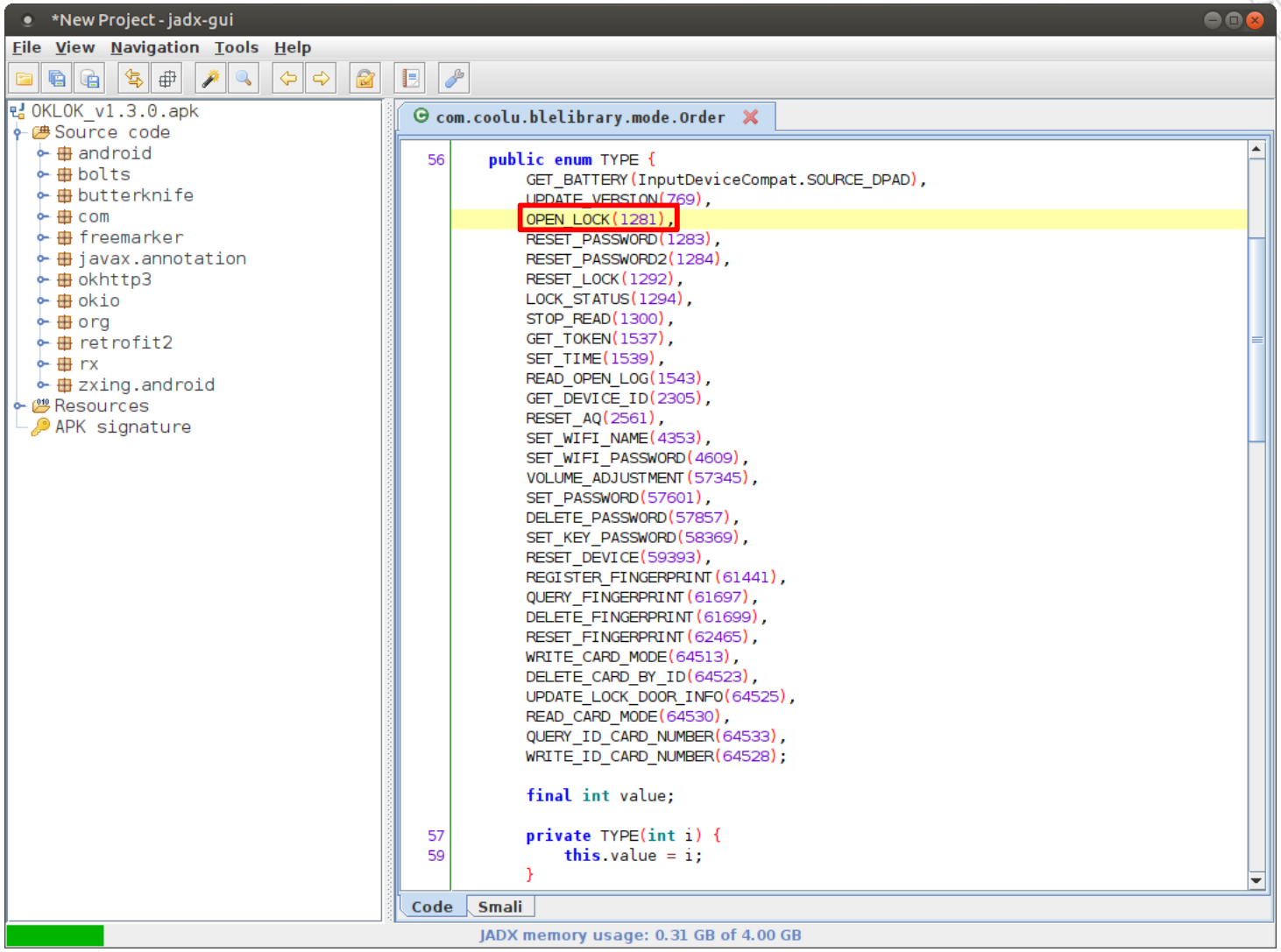
com.coolu.blelibrary



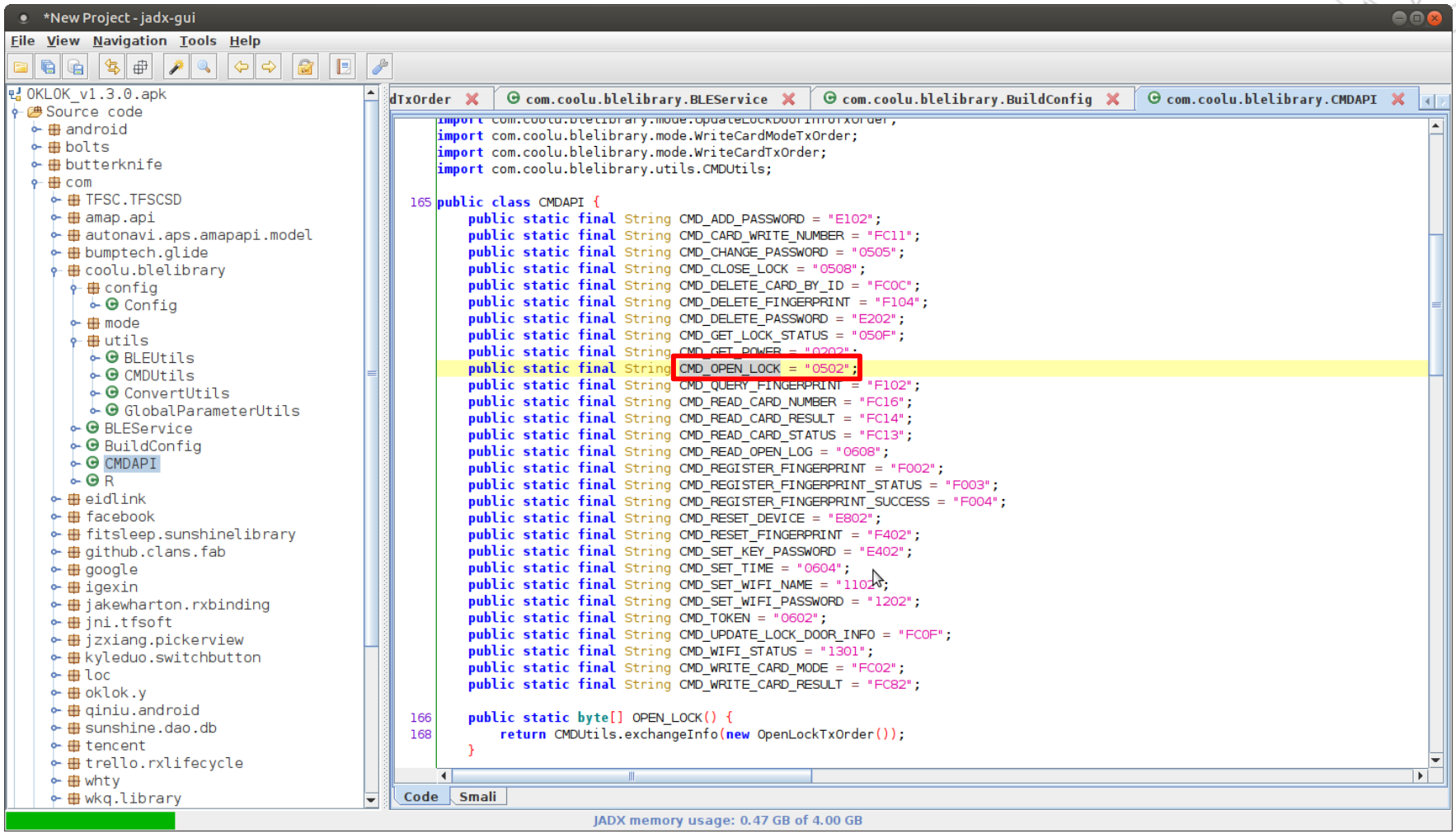
com.coolu.blelibrary



com.coolu.blelibrary



com.coolu.blelibrary



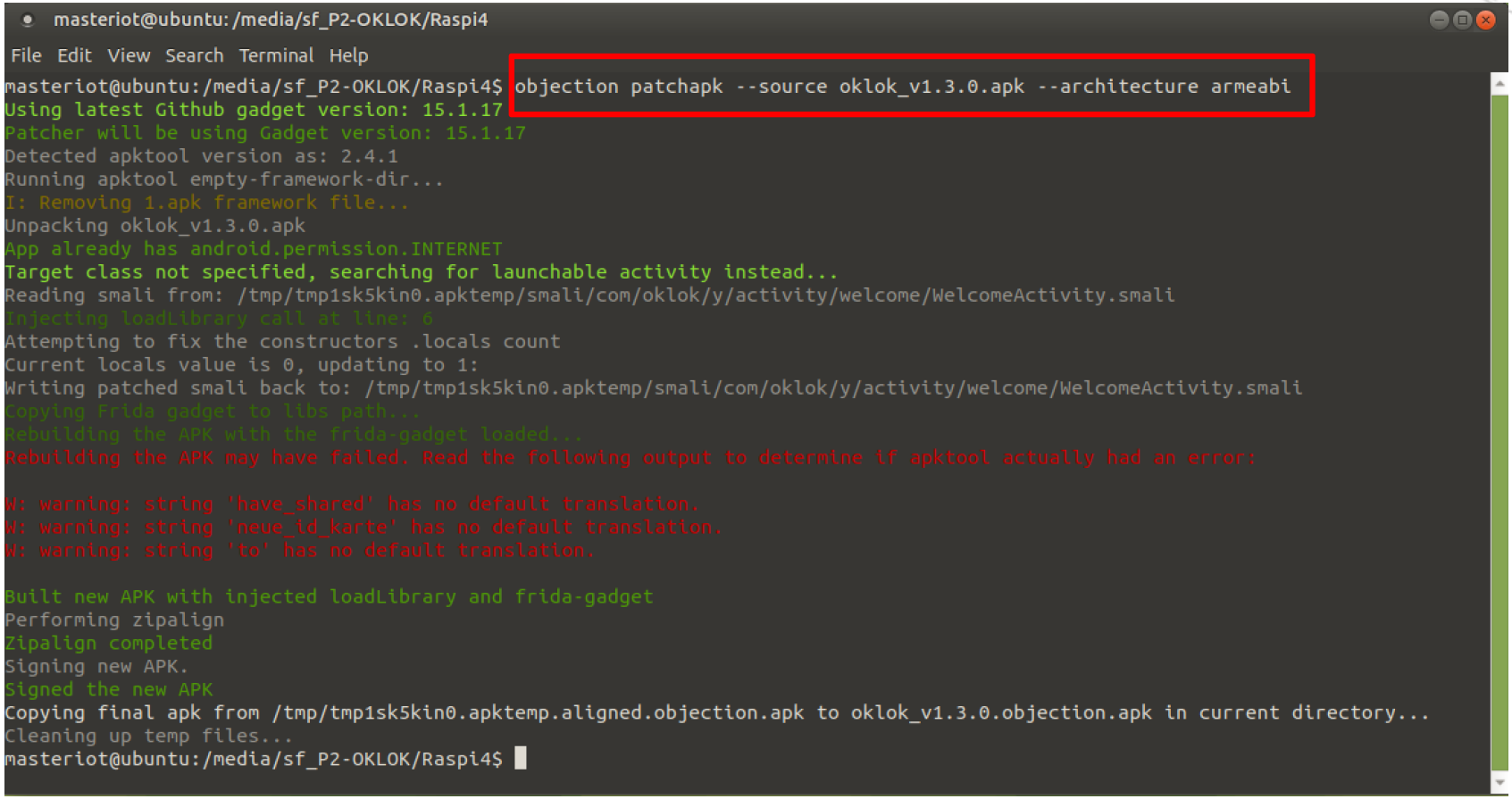
FRIDA

- Dynamic instrumentation toolkit for developers, reverse-engineers, and security researchers:
  - Scriptable: **Inject your own scripts** into black box processes.
  - Portable: Works on Windows, macOS, GNU/Linux, iOS, Android, and QNX.
  - Free: Frida is and will always be free software
- Objection:
  - Runtime mobile exploration toolkit, powered by Frida, built to help you assess the security posture of your mobile applications, without needing a jailbreak.

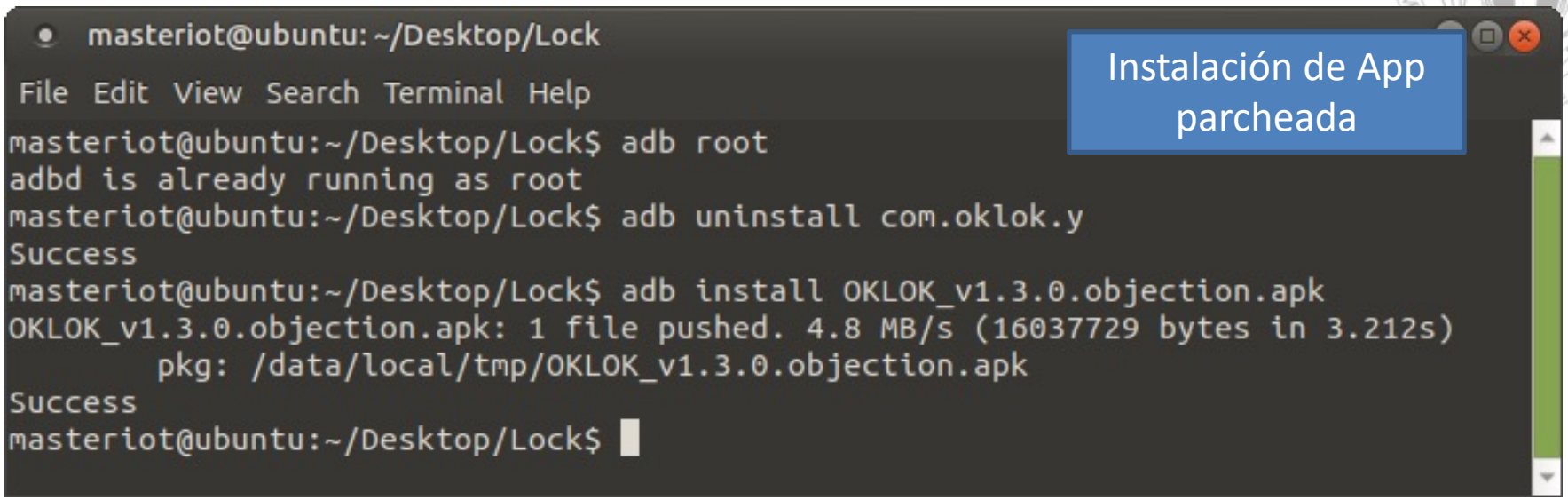
https://frida.re/

https://github.com/sensepost/objection

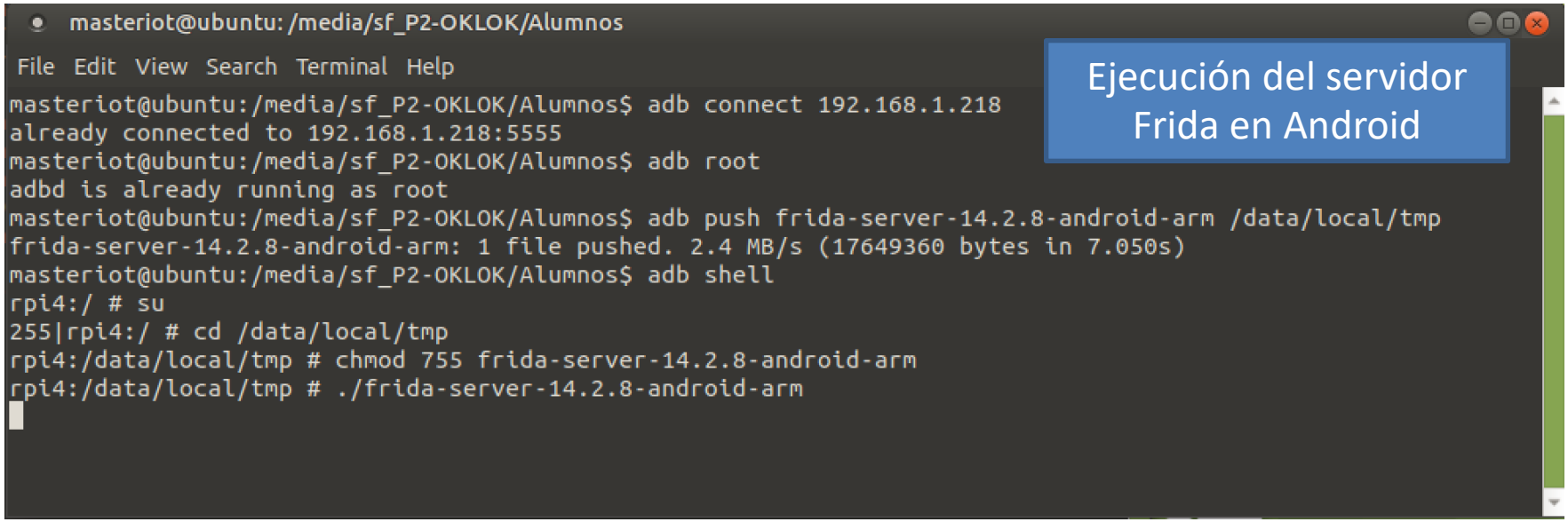
Instrumentalizar Apk v1.3.0



Instrumentalizar Apk

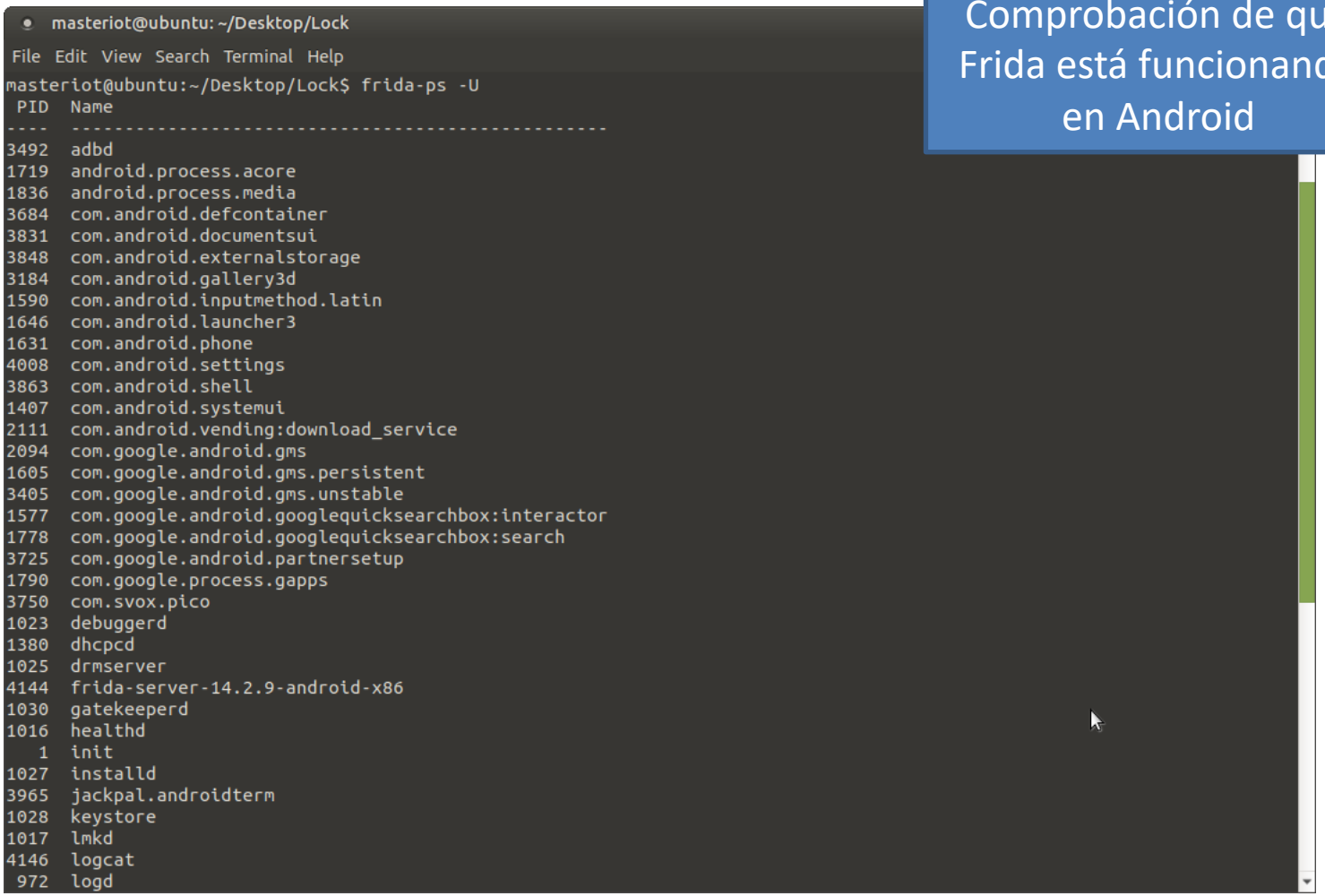


Instalación de App parcheada



Ejecución del servidor Frida en Android

Instrumentalizar Apk



Comprobación de que Frida está funcionando en Android

oklok-frida_new.js — UNREGISTERED

oklok-frida_new.js

```
1  //oklok1.js
2  Java.perform(function ()
3  {
4      var CMDUtils = Java.use('com.coolu.blelibrary.utils.CMDUtils');
5
6      var log_byte_array = function (arr) {
7          var result = "";
8          var buffer = Java.array('byte', arr);
9          for(var i = 0; i < buffer.length; ++i)  {
10             var hexb = (buffer[i] & 0xFF).toString(16);
11             if (hexb.length == 1) hexb = '0' + hexb;
12             result += hexb;
13         }
14         console.log(result);
15     };
16
17     CMDUtils.Encrypt.implementation = function (pt, key) {
18         console.log('[+] Inside Encrypt()  ======');
19         var ct = this.Encrypt(pt, key);
20         console.log('Pt:')
21         log_byte_array(pt)
22         console.log('key:')
23         log_byte_array(key);
24         return ct;
25     };
26
27     CMDUtils.Decrypt.implementation = function (ct, key) {
28         console.log('[+] Inside Decrypt()  ======');
29         var pt = this.Decrypt(ct, key);
30         console.log('Pt:')
31         log_byte_array(pt)
32         console.log('Key:')
33         log_byte_array(key);
34         return pt;
35     };
36 });
37
```

Line 1, Column 1 — Tab Size: 4 — JavaScript

Código sustitutorio de *Encrypt/Decrypt*

FRIDA

Ejecutamos la App parcheada en Android y enviamos el código a ejecutar

```
masteriot@ubuntu: ~/Desktop/Lock
File  Edit  View  Search  Terminal  Help
masteriot@ubuntu:~/Desktop/Lock$ frida -U -l oklok-frida_new.js com.oklok.y

     ____
    / _  |   Frida 14.2.8 - A world-class dynamic instrumentation toolkit
   | (_| |
    > _  |   Commands:
   /_/ |_|       help      -> Displays the help system
   . . . .       object?   -> Display information about 'object'
   . . . .       exit/quit -> Exit
   . . . .
   . . . .   More info at https://www.frida.re/docs/home/

[VirtualBox::com.oklok.y]-> [+] Inside Decrypt()  ======
Pt:
050d01000208c501020500000000000
Key:
034100624f0a29355c193f1a39192356
[+] Inside Encrypt()  ======
Pt:
060101014d130230471f24354b2c4f37
key:
034100624f0a29355c193f1a39192356
[+] Inside Encrypt()  ======
Pt:
0601010177345e347c743f3d5b27531f
key:
034100624f0a29355c193f1a39192356
[+] Inside Decrypt()  ======
Pt:
060207164ff647010205000000000000
Key:
034100624f0a29355c193f1a39192356
[+] Inside Decrypt()  ======
Pt:
060207164ff647010205000000000000
```

Cifra de 16B -> AES-128

Analizando la traza

```
key:  034100624f0a29355c193f1a39192356

[+] Inside Encrypt()  ======
Pt:  060101015705162b7c5b34162b4b4b2e    -->  Traza Nº 101
[+] Inside Encrypt()  ======
Pt:  0602077464a8bd01020500000000000     -->  Traza Nº 103
[+] Inside Decrypt()  ======
Pt:  06010101644c391e5b6a4f3237477a78    -->  Traza Nº 105
[+] Inside Decrypt()  ======
Pt:  0602077464a8bd01020500000000000     -->  Traza Nº 107
[+] Inside Encrypt()  ======
Pt:  020101017464a8bd2f22114d0c157e65    -->  Traza Nº 109
[+] Inside Decrypt()  ======
Pt:  0202015e64a8bd01020500000000000     -->  Traza Nº 112
[+] Inside Encrypt()  ======
Pt:  0501063030303030307464a8bd1f0375    -->  Traza Nº 113
[+] Inside Decrypt()  ======
Pt:  0502010064a8bd01020500000000000     -->  Traza Nº 121
[+] Inside Decrypt()  ======
Pt:  050d010064a8bd01020500000000000     -->  Traza Nº 123
```

Analizando la traza

```
key:  034100624f0a29355c193f1a39192356

[+] Inside Encrypt()  ======
Pt:  0601010                                    -->
[+] Inside Encrypt()  ======
Pt:  0602077464a8bd0102050000000000000          -->
[+] Inside Decrypt()  ======
Pt:  0601010                                    -->
[+] Inside Decrypt()  ======
Pt:  0602077464a8bd0102050000000000000          -->
[+] Inside Encrypt()  ======
Pt:  0201010                                    -->
[+] Inside Decrypt()  ======
Pt:  0202015e64a8bd0102050000000000000          -->  Traza Nº 112
[+] Inside Encrypt()  ======
Pt:  0501063                                    -->  Traza Nº 113
[+] Inside Decrypt()  ======
Pt:  0502010064a8bd0102050000000000000          -->  Traza Nº 121
[+] Inside Decrypt()  ======
Pt:  050d010064a8bd0102050000000000000          -->  Traza Nº 123
```

CMDAPI -> CMD_TOKEN

CMDAPI -> CMD_TOKEN

CMDAPI -> CMD_GET_POWER

CMDAPI -> CMD_OPEN_LOCK 😏

```
import com.coolu.blelibrary.mode.WriteCardIdOrder;
import com.coolu.blelibrary.utils.CMDUtils;

public class CMDAPI {
    public static final String CMD_ADD_PASSWORD = "E102";
    public static final String CMD_CARD_WRITE_NUMBER = "FC11";
    public static final String CMD_CHANGE_PASSWORD = "0505";
    public static final String CMD_CLOSE_LOCK = "0508";
    public static final String CMD_DELETE_CARD_BY_ID = "FC0C";
    public static final String CMD_DELETE_FINGERPRINT = "F104";
    public static final String CMD_DELETE_PASSWORD = "E202";
    public static final String CMD_GET_LOCK_STATUS = "050F";
    public static final String CMD_GET_POWER = "0202";
    public static final String CMD_OPEN_LOCK = "0502";
    public static final String CMD_QUERY_FINGERPRINT = "F102";
    public static final String CMD_READ_CARD_NUMBER = "FC16";
    public static final String CMD_READ_CARD_RESULT = "FC14";
    public static final String CMD_READ_CARD_STATUS = "FC13";
    public static final String CMD_READ_OPEN_LOG = "0608";
    public static final String CMD_REGISTER_FINGERPRINT = "F002";
    public static final String CMD_REGISTER_FINGERPRINT_STATUS = "F003";
    public static final String CMD_REGISTER_FINGERPRINT_SUCCESS = "F004";
    public static final String CMD_RESET_DEVICE = "E802";
    public static final String CMD_RESET_FINGERPRINT = "F402";
    public static final String CMD_SET_KEY_PASSWORD = "E402";
    public static final String CMD_SET_TIME = "0604";
    public static final String CMD_SET_WIFI_NAME = "1102";
    public static final String CMD_SET_WIFI_PASSWORD = "1202";
    public static final String CMD_TOKEN = "0602";
    public static final String CMD_UPDATE_LOCK_DOOR_INFO = "FC0F";
    public static final String CMD_WIFI_STATUS = "1301";
    public static final String CMD_WRITE_CARD_MODE = "FC02";
    public static final String CMD_WRITE_CARD_RESULT = "FC82";
```

Analizando la traza

```
key:  034100624f0a29355c193f1a39192356

[+] Inside Encrypt()  ======
Pt:  060101015705162b7c5b34162b4b4b2e    -->  Traza Nº 101
[+] Inside Encrypt()  ======
Pt:  0602077464a8bd01020500000000000     -->  Traza Nº 103
[+] Inside Decrypt()  ======
Pt:  06010101644c391e5b6a4f3237477a78    -->  Traza Nº 105
[+] Inside Decrypt()  ======
Pt:  0602077464a8bd01020500000000000     -->  Traza Nº 107
[+] Inside Encrypt()  ======
Pt:  020101017464a8bd2f22114d0c157e65    -->  Traza Nº 109
[+] Inside Decrypt()  ======
Pt:  0202015e64a8bd01020500000000000     -->  Traza Nº 112
[+] Inside Encrypt()  ======
Pt:  0501063030303030307464a8bd1f0375    -->  Traza Nº 113
[+] Inside Decrypt()  ======
Pt:  0502010064a8bd01020500000000000     -->  Traza Nº 121
[+] Inside Decrypt()  ======
Pt:  050d010064a8bd01020500000000000     -->  Traza Nº 123
```

Analizando la traza

```
key:  03410624f0a29355c193f1a39192356

[+] Inside Encrypt()  ======
Pt:  0601010157051 62b7c5b34162b4b4b2e    -->  T
[+] Inside Encrypt()  ======
Pt:  0602077                              -->  T
[+] Inside Decrypt()  ======
Pt:  0601010164 4c391e5b6a4f3237477a78    -->  T
[+] Inside Decrypt()
Pt:  0602077                              -->  T
[+] Inside E
Pt:  0201010174 64a8bd2f22114d0c157e65    -->  T
[+] Inside Decrypt()  ======
Pt:  0202015                              -->  T
[+] Inside Encrypt()  ======
Pt:  0501063030303030307464a8bd1f0375     -->  Traza Nº 113
[+] Inside Decrypt()  ======
Pt:  0502010064a8bd0102050000000000000    -->  Traza Nº 121
[+] Inside Decrypt()  ======
Pt:  050d010064a8bd0102050000000000000    -->  Traza Nº 123
```

0x0601=1537 -> GET_TOKEN

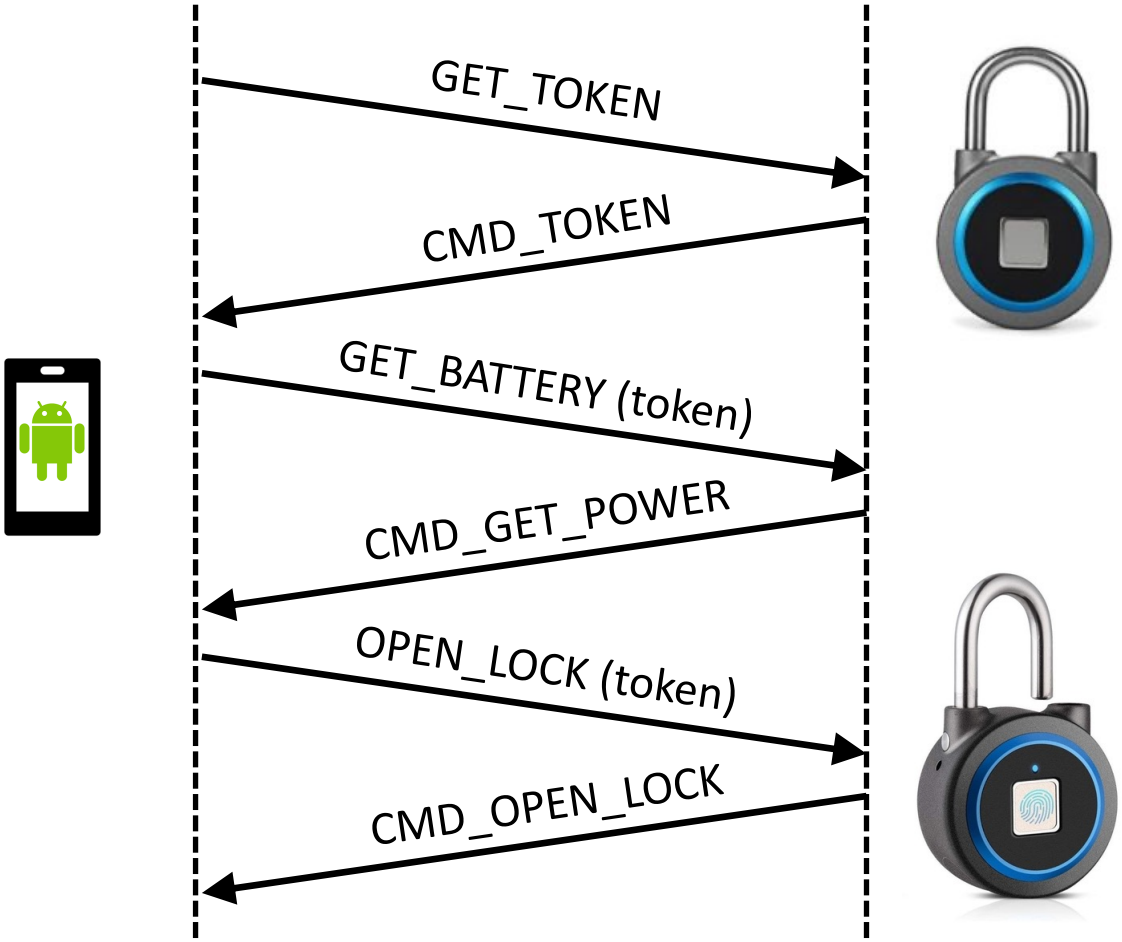
0x0601=1537 -> GET_TOKEN

0x0201=513 -> GET_BATTERY (SOURCE_DPAD)

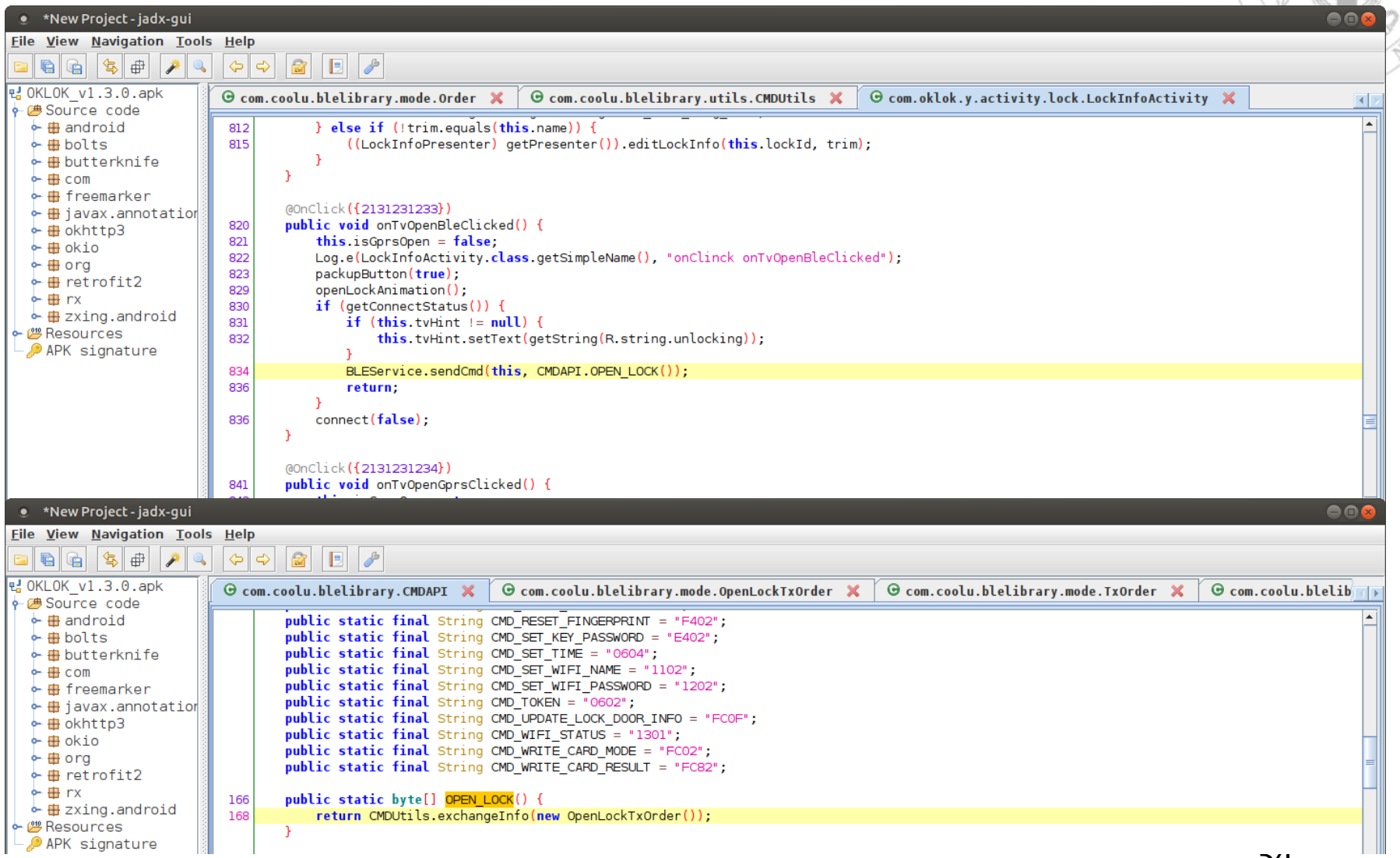
0x0501=1281 -> OPEN_LOCK

```
public enum TYPE {
    GET_BATTERY(InputDeviceCompat.SOURCE_DPAD),
    UPDATE_VERSION(769),
    OPEN_LOCK(1281),
    RESET_PASSWORD(1283),
    RESET_PASSWORD2(1284),
    RESET_LOCK(1292),
    LOCK_STATUS(1294),
    STOP_READ(1300),
    GET_TOKEN(1537),
    SET_TIME(1539),
    READ_OPEN_LOG(1543),
    GET_DEVICE_ID(2305),
    RESET_AQ(2561),
    SET_WIFI_NAME(4353),
    SET_WIFI_PASSWORD(4609),
    VOLUME_ADJUSTMENT(57345),
    SET_PASSWORD(57601),
    DELETE_PASSWORD(57857),
    SET_KEY_PASSWORD(58369),
    RESET_DEVICE(59393),
    REGISTER_FINGERPRINT(61441),
    QUERY_FINGERPRINT(61697),
    DELETE_FINGERPRINT(61699),
    RESET_FINGERPRINT(62465),
    WRITE_CARD_MODE(64513),
    DELETE_CARD_BY_ID(64523),
    UPDATE_LOCK_DOOR_INFO(64525),
    READ_CARD_MODE(64530),
    QUERY_ID_CARD_NUMBER(64533),
    WRITE_ID_CARD_NUMBER(64528);
```

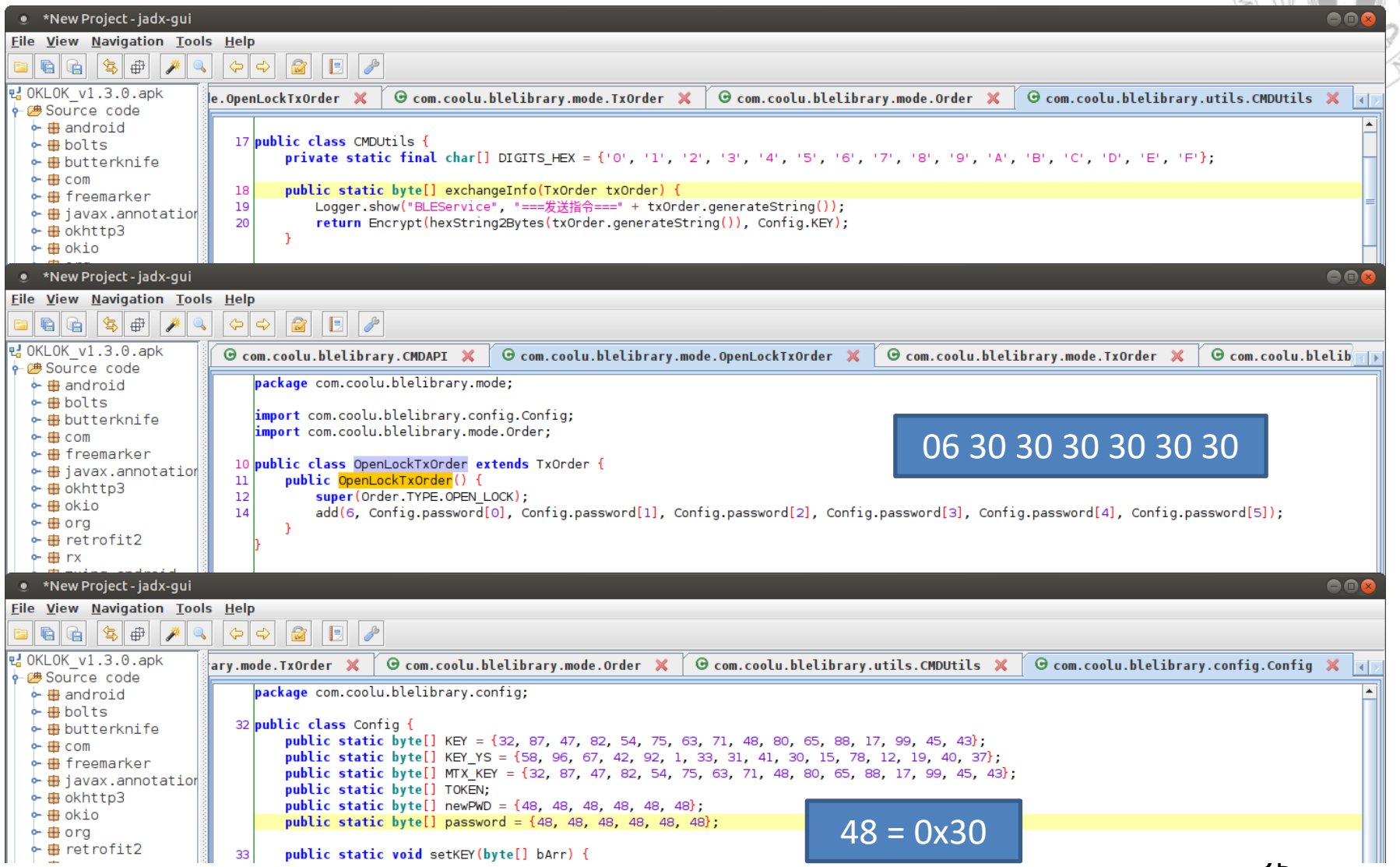
Proceso de desbloqueo



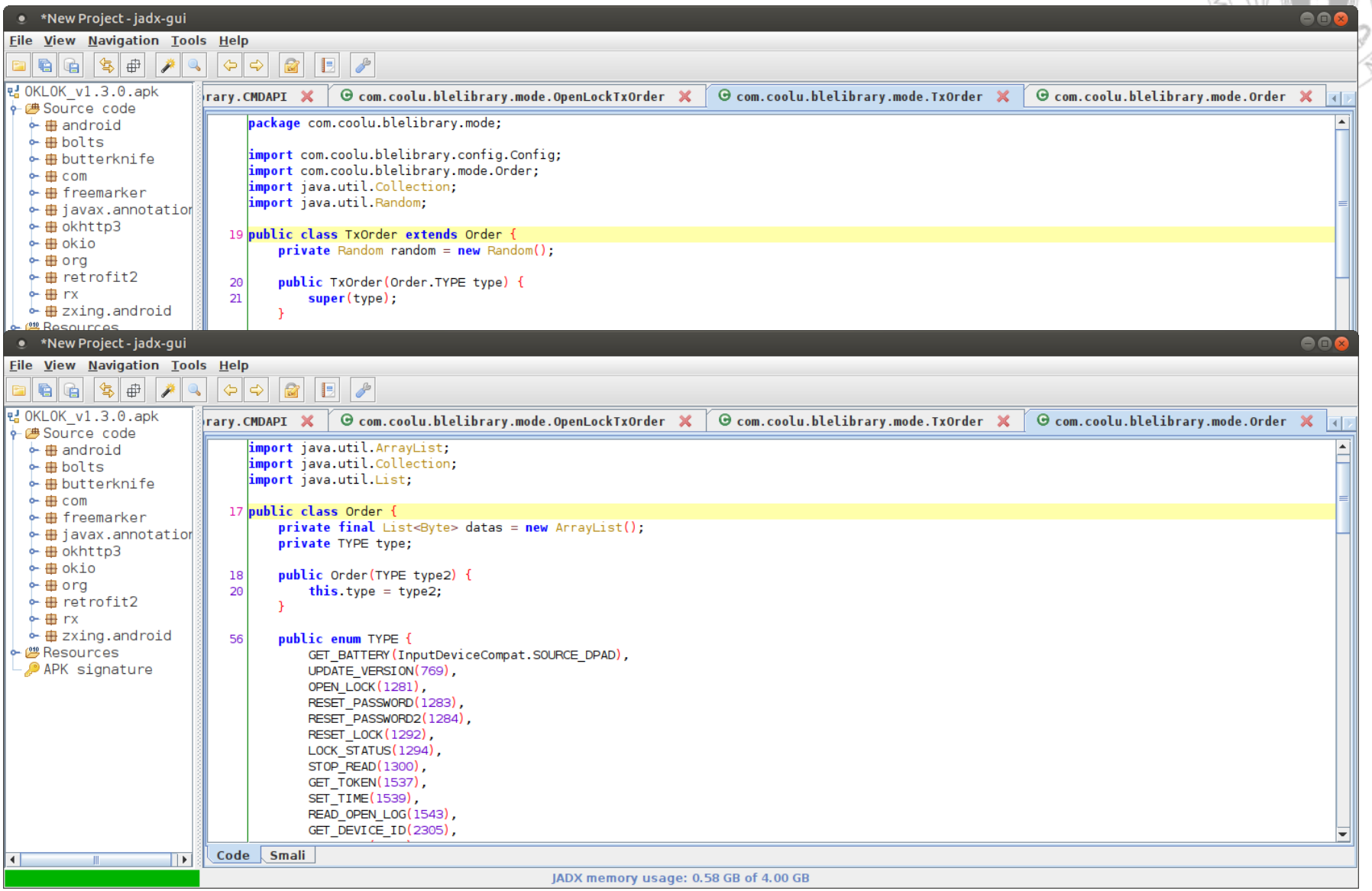
Mensaje de desbloqueo



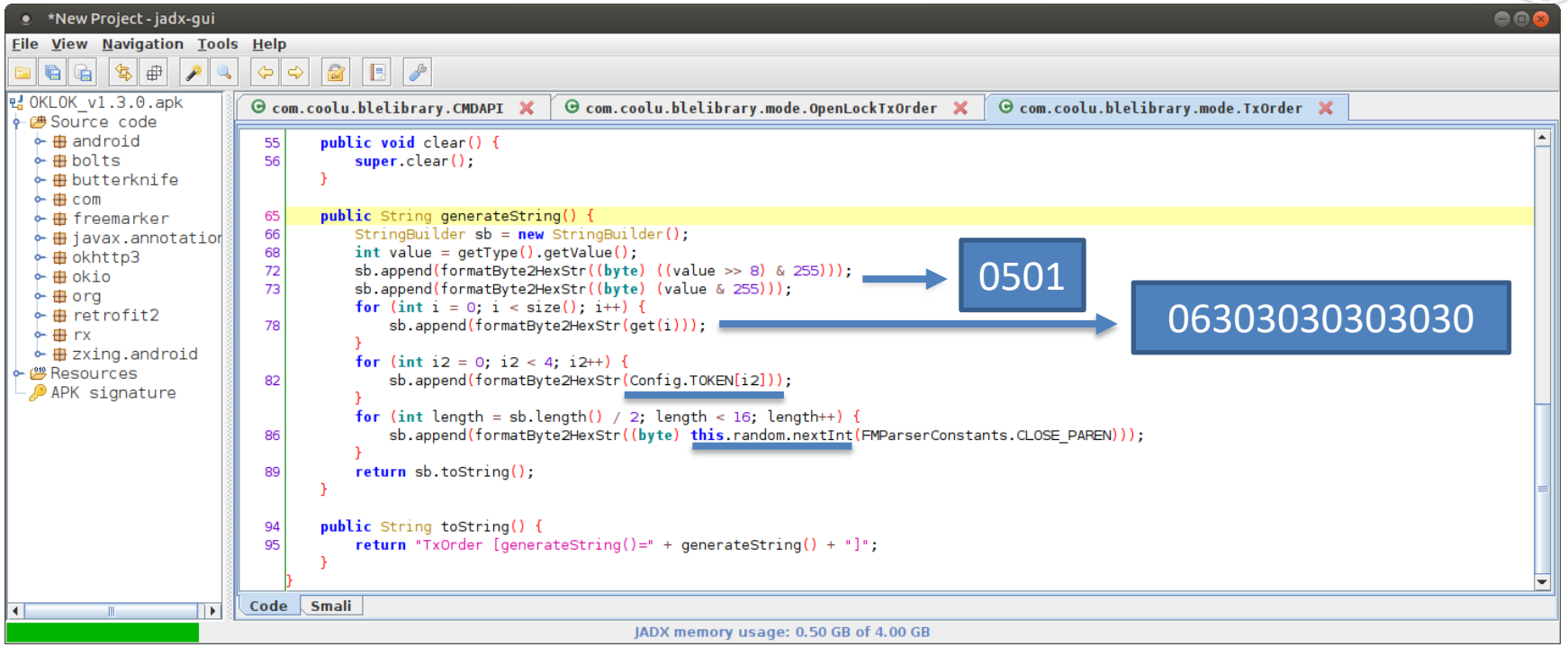
Mensaje de desbloqueo



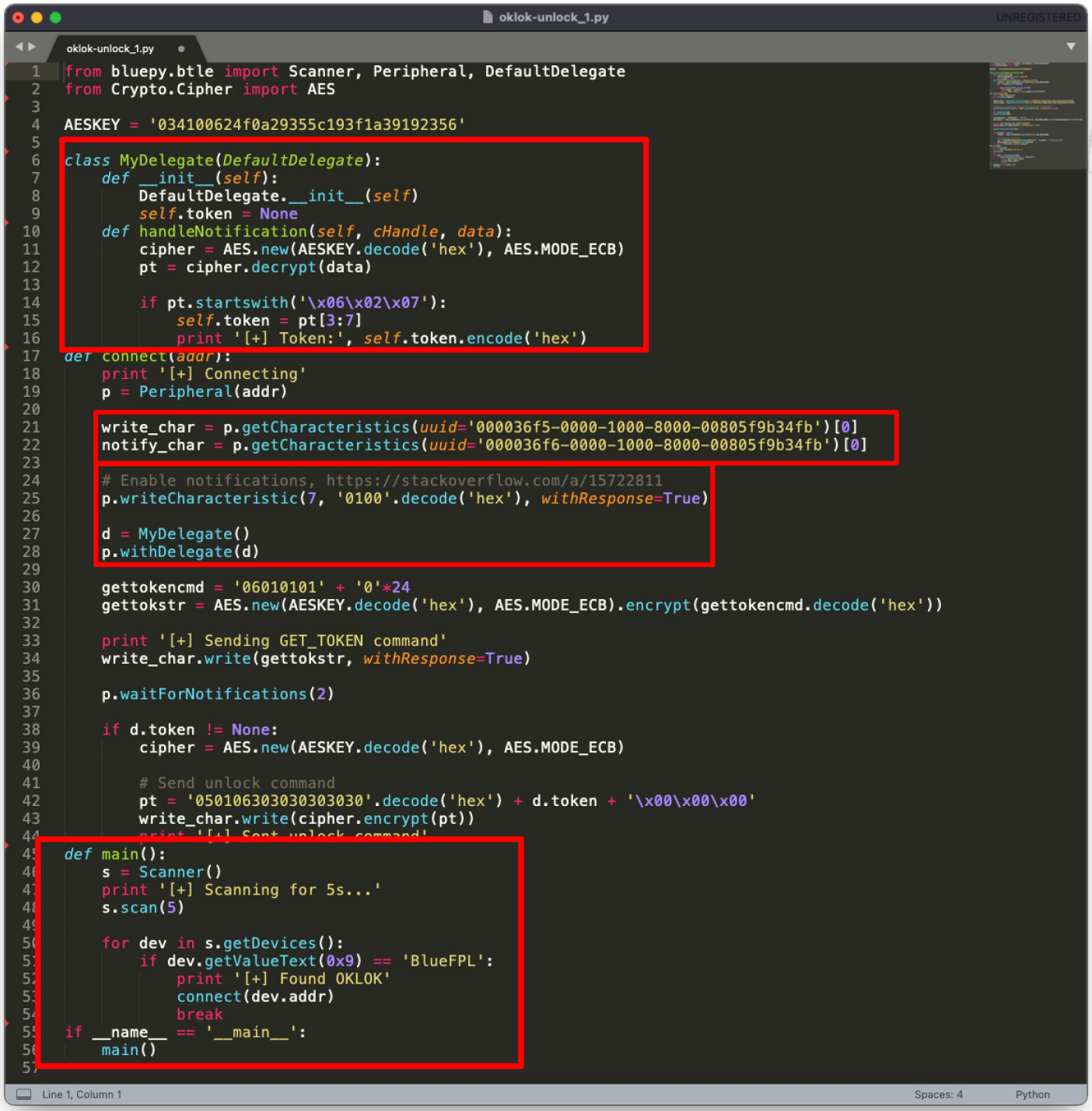
Mensaje de desbloqueo



Mensaje de desbloqueo



0501 06303030303030 [TOKEN] [RAND]


oklok-unlock_1.py
UNREGISTERED
oklok-unlock_1.py
from bluepy.btle import Scanner, Peripheral, DefaultDelegate
from Crypto.Cipher import AES
AESKEY = '034100624f0a29355c193f1a39192356'
class MyDelegate(DefaultDelegate):
def __init__(self):
DefaultDelegate.__init__(self)
self.token = None
def handleNotification(self, cHandle, data):
cipher = AES.new(AESKEY.decode('hex'), AES.MODE_ECB)
pt = cipher.decrypt(data)
if pt.startswith('\x06\x02\x07'):
self.token = pt[3:7]
print '[+] Token:', self.token.encode('hex')
def connect(addr):
print '[+] Connecting'
p = Peripheral(addr)
write_char = p.getCharacteristics(uuid='000036f5-0000-1000-8000-00805f9b34fb')[0]
notify_char = p.getCharacteristics(uuid='000036f6-0000-1000-8000-00805f9b34fb')[0]
# Enable notifications, https://stackoverflow.com/a/15722811
p.writeCharacteristic(7, '0100'.decode('hex'), withResponse=True)
d = MyDelegate()
p.withDelegate(d)
gettokencmd = '06010101' + '0'*24
gettokstr = AES.new(AESKEY.decode('hex'), AES.MODE_ECB).encrypt(gettokencmd.decode('hex'))
print '[+] Sending GET_TOKEN command'
write_char.write(gettokstr, withResponse=True)
p.waitForNotifications(2)
if d.token != None:
cipher = AES.new(AESKEY.decode('hex'), AES.MODE_ECB)
# Send unlock command
pt = '050106303030303030'.decode('hex') + d.token + '\x00\x00\x00'
write_char.write(cipher.encrypt(pt))
def main():
s = Scanner()
print '[+] Scanning for 5s...'
s.scan(5)
for dev in s.getDevices():
if dev.getValueText(0x9) == 'BlueFPL':
print '[+] Found OKLOK'
connect(dev.addr)
break
if __name__ == '__main__':
main()
Line 1, Column 1
Spaces: 4
Python

Tareas para el laboratorio

1. Obtener la dirección física del candado
2. Registrar y analizar el tráfico BLE entre la App y el candado
3. Intentar un *reply-attack* mediante BLE
4. Analizar la App usando *jadx*
5. Instrumentar el código de la App usando *Frida* y *Objection* para averiguar la clave
6. Entender el mecanismo de apertura
7. Abrir el candado usando un PC
8. Proponer mejoras de seguridad