Destripando Pokémon Go by OWASP

Eduardo Sánchez

GOBIERNO DE ESPAÑA
MINISTERIO DE INDUSTRIA, ENERGÍA Y TURISMO

incibe_
TRABAJANDO POR LA CONFIANZA DIGITAL

CyberCamp.es

Ficha de este Pokemon

- Ingeniero Informático. Máster en Seguridad TIC
- Profesor de F.P. y Universidad UCLM y UNEX.
- Blog White Walkers of Hacking
- Colaborador del blog de Hacking Ético
- Cofundador de la comunidad Hack&Beers
- Socio fundador de ANPHACKET
- Organizador del congreso Qurtuba Security Congress
- Fundador de Security High School

eduSatoe

HACK &BEERS

Qurtuba Security Congress

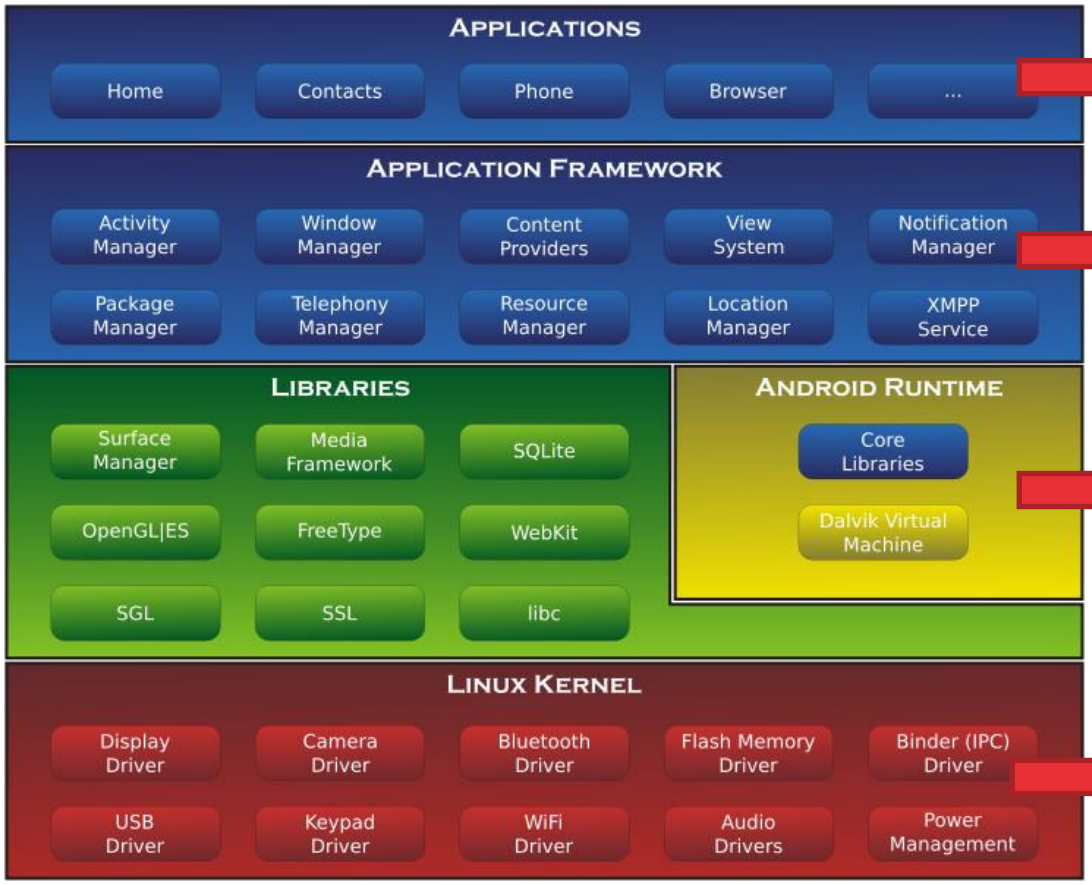
ANPHACKET

Security High School

Índice

- Introducción
  - Arquitectura
  - Modelo Seguridad
  - Tipos de Aplicaciones
  - OWASP Mobile Top 10 Risk
  - Metodología OWASP
- Obteniendo Información
  - Permisos
  - Fabricante
  - Conexiones
- Análisis Estático
  - Reversing
  - Hardcoded Password
- Análisis Dinámico
  - Bypassing Authentication
  - Content Provider Leakage
  - Insecure Storage
  - Client Side Injection
  - Logging
  - Network Analysis
- Herramientas Automatizadas
  - MobSF
  - DroidBox
- Resumen Metodología

Arquitectura Android



APPS DEL SISTEMA Y DEL USUARIO

MANEJA FUNCIONES BÁSICAS DEL TELÉFONO
- CONTENT PROVIDERS
- TELEPHONY MANAGER ...

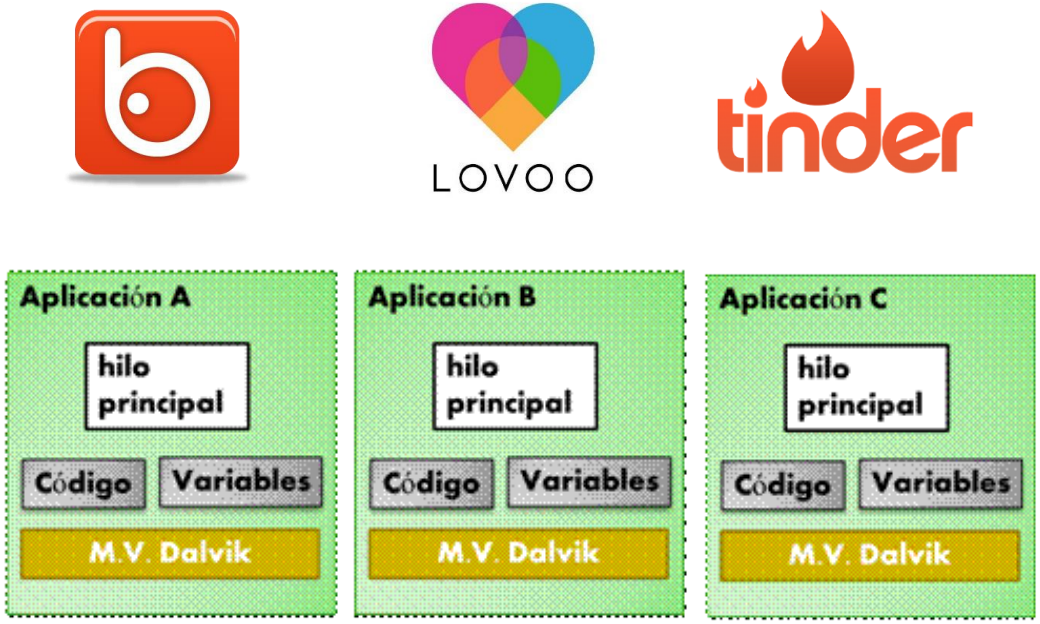
LIBRERÍAS OPEN-SOURCE
- SQLITE - MEDIA
- SSL - WEB BROWSER ...

Librerías JAVA con la DALVIK VM

Android 1.0 – linux KERNEL 2.6.25
...
Android 4.4 – linux KERNEL 3.10 → ART
Android 5.1.1 – linux KERNEL 3.16
Android 6.0.1 – linux KERNEL 3.18

Arquitectura Android

Dalvik Virtual Machine



- UNA **DVM** PARA CADA PROCESO.
- INDEPENDENCIA DE :
  - DATOS Y FICHEROS DE APLICACIÓN.
  - MEMORIA DE EJECUCIÓN.

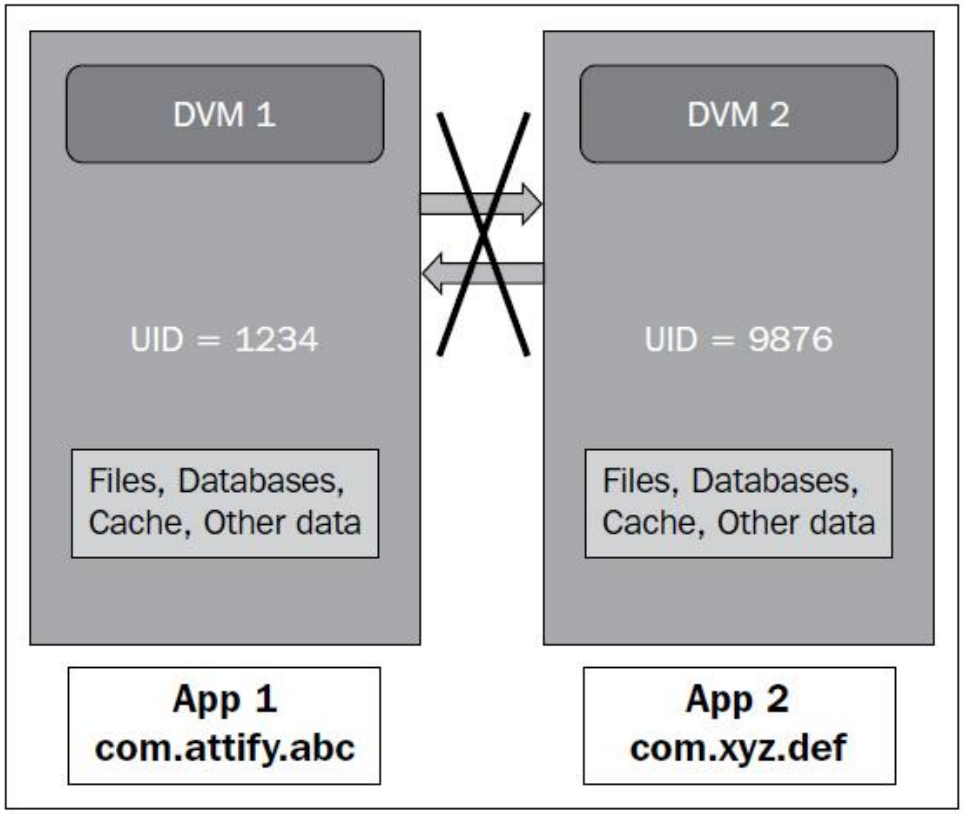
Arquitectura Android

ART – Android Runtime

- A partir de Android 5.x Lollipop → ART sustituye a DVM
- Dalvik compila cada vez que ejecutas.
- ART crea un archivo de compilación a la hora de instalar la App (instala más lento), de tal forma que ahorra batería porque no compila tanto (ahorro de CPU)
- Apps ocupan más en el dispositivo.
- ART mejoras de rendimiento → recolección de basura, App de depuración y perfiles.
- Compatibilidad → Uso de DVM bytecodes

Arquitectura Android

Sandboxing with Dalvik Virtual Machine



- RUTA /SYSTEM/ETC/PERMISSIONS/
- FICHERO PLATFORM.XML
- Permisos por Grupos
- UID único por App (ps)
- Varios UID por GID

Arquitectura Android

Directorios de Binarios y Aplicaciones

- **Binarios**

/system/bin/ y /system/xbin/

- **Aplicaciones**

/data/data/ → Instalación

/data/apk/ → Fichero .apk

- **Sistema**

/data/system/

- **Descargar versiones**

  → APK Pure

  → APK Downloader

Una **aplicación de Android** es un fichero con extensión **.APK**. En realidad, esto no es más que un fichero comprimido **.ZIP** renombrado, con una cierta estructura general.

Modelo de Seguridad

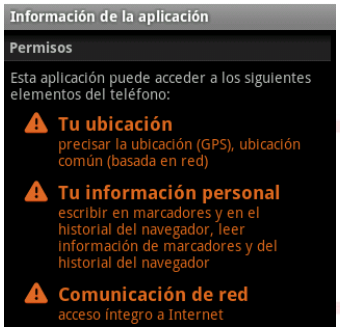
Cualquier **ACCIÓN** que realice una aplicación debe tener **PERMISO**

Todas las **APPS** están **FIRMADAS**

Se crea un **USUARIO** por cada **APP**

Modelo de Seguridad

ACEPTAMOS TODO



INSTALAMOS TODO

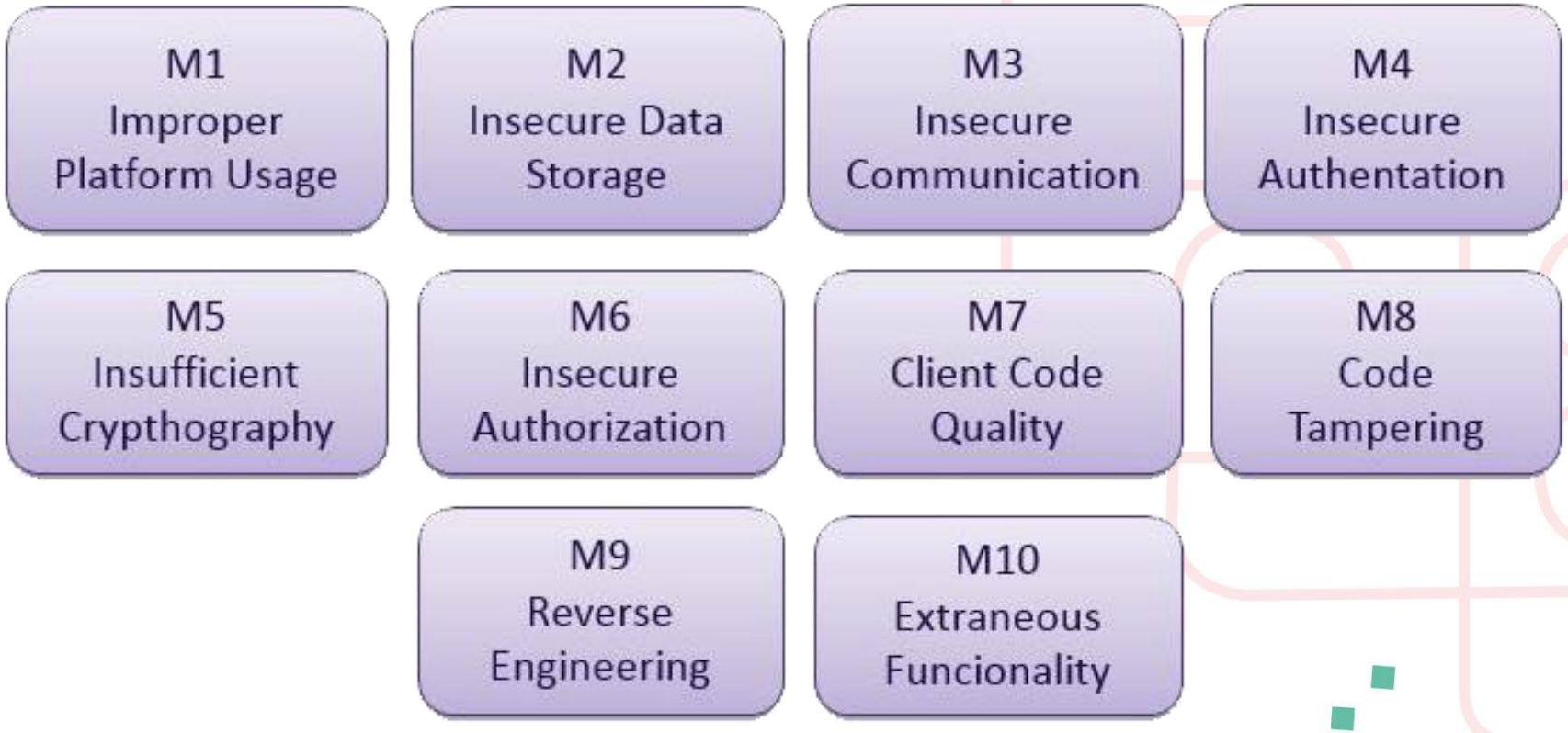
ROOT TIENE ACCESO A TODO

Tipos de Aplicaciones

Web – Nativas - Híbridas

- **Aplicaciones basadas en Web → Consultas al Exterior**
  - Conexión a un servicio Web.
  - BBDD online.
  - Poca información en el terminal.
- **Aplicaciones Nativas → Todo el Código Fuente en el terminal**
  - Todas las funcionalidades en el terminal.
  - BBDD local.
- **Aplicaciones Híbridas**
  - Funcionalidades en la App del terminal y conexión a un servicio Web.
  - BBDD con información repartida entre la Web y la BBDD local (copias).

OWASP Mobile Top 10 Risks 2016



OWASP Mobile CheckList 2016

CLIENT SIDE CHECKS

Sr.	Vulnerability Name	Applicable Platform	Compliant ? Yes/No/NA	Classification
1	Application is Vulnerable to Reverse Engineering Attack/Lack of Code	All		Static Checks
2	Account Lockout not Implemented	All		Dynamic Checks
3	Application is Vulnerable to XSS	All		Static + Dynamic Chec
4	Authentication bypassed	All		Dynamic Checks
5	Hard coded sensitive information in Application Code (including Crypt	All		Static Checks
6	Malicious File Upload	All		Dynamic Checks
7	Session Fixation	All		Dynamic Checks
8	Application does not Verify MSISDN	WAP		Unknown
9	Privilege Escalation	All		Dynamic Checks
10	SQL Injection	All		Static + Dynamic Chec
11	Attacker can bypass Second Level Authentication	All		Dynamic Checks
12	Application is vulnerable to LDAP Injection	All		Dynamic Checks
13	Application is vulnerable to OS Command Injection	All		Dynamic Checks
14	iOS snapshot/backgrounding Vulnerability	iOS		Dynamic Checks
15	Debug is set to TRUE	Android		Static Checks
16	Application makes use of Weak Cryptography	All		Static Checks
17	Cleartext information under SSL Tunnel	All		Dynamic Checks

OWASP Mobile Security Testing

Fases del proceso de Pentest

- **Information Gathering**
  - Info del fabricante y sus otras Apps. ¿Usa algún framework para desarrollo?
  - Funcionalidades de la App e interacciones con otras.
  - Protocolo/s utilizado/s y posibles fugas de información.
  - Recursos que utiliza (GPS, Camera, Contactos..)
- **Análisis Estático**
  - Ingeniería Inversa → donde podemos obtener: permisos, actividades principales, recursos, errores de configuración (modo debug), necesita permisos root, información hardcodeada (apikeys o credenciales...), puntos de entrada al sistema, autenticación...
- **Análisis Dinámico**
  - Análisis de vulnerabilidades → archivos creados por el sistema, datos sin cifrar e información sensible, análisis del log, esnifar tráfico de la App, fuzzing, chequear posible bypass, posibles SQLi, XSS, LFI...

Mobile Security Testing by OWASP

OWASP Mobile Apps Checklist 2016

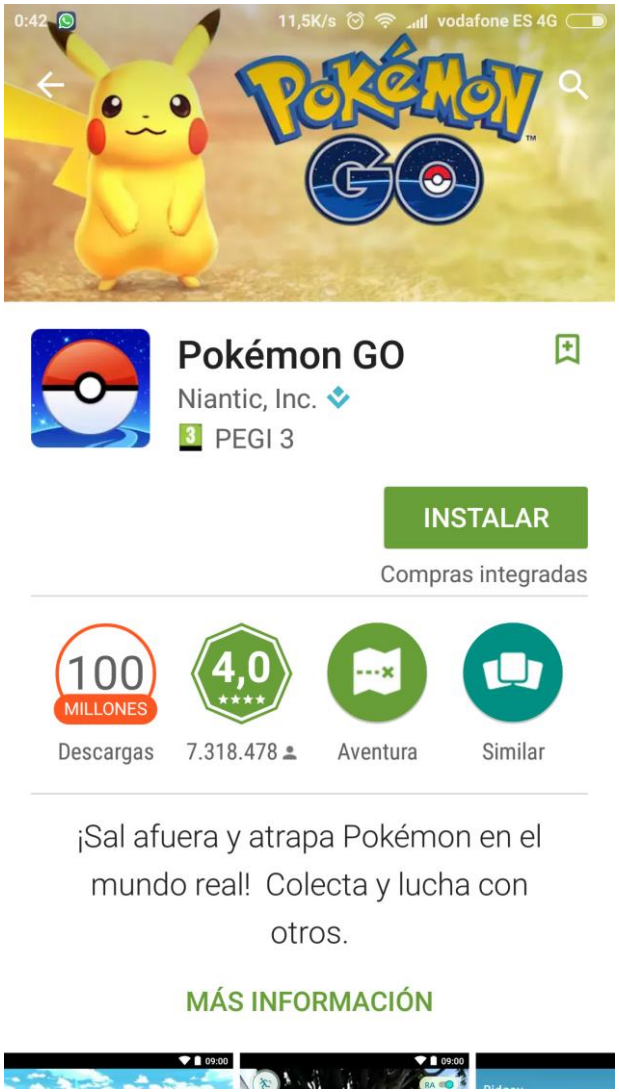
Estructura de las Apps

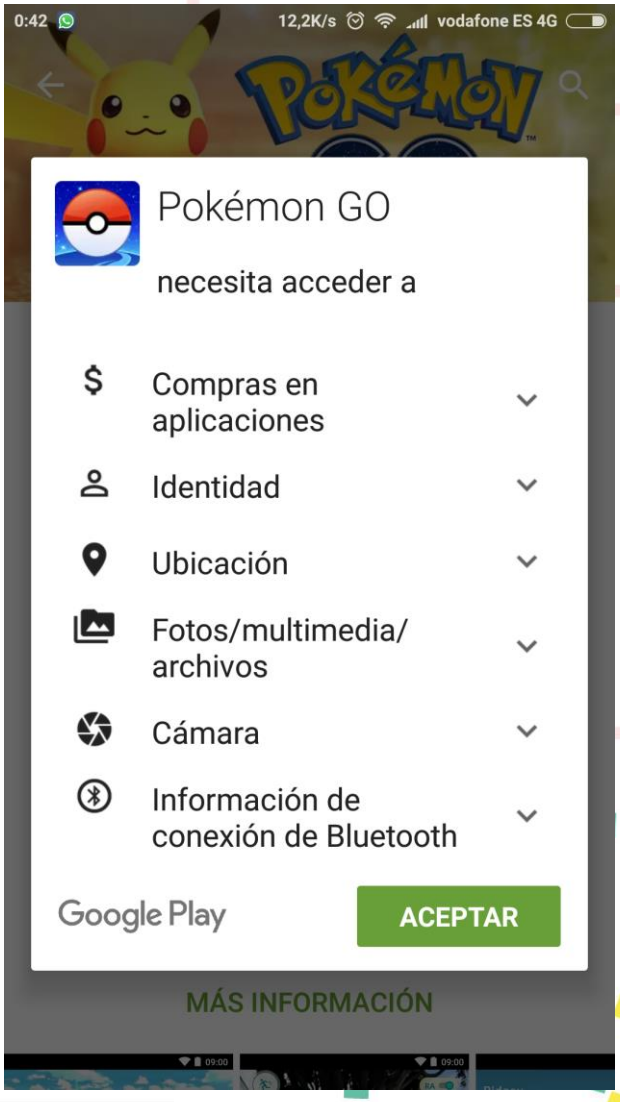
¿Dónde almacenan los datos?

- **Shared-Preferences**
  - Suelen guardarse booleanos en XML para temas de configuración.
  - Podemos encontrarnos sorpresas (user:pass)
- **Almacenamiento Interno**
  - En la memoria interna.
  - Protegido por los permisos MODE_PRIVATE.
- **Almacenamiento Externo**
  - SD Card externa → WORLD_READABLE.
  - MODE_WORLD_READABLE o MODE_WORLD_WRITABLE → Acceso libre a todos.
- **Bases de Datos SQLite**
  - Sólo acceso para el usuario de la aplicación.
- **En un servidor en la Red**

Obteniendo Información

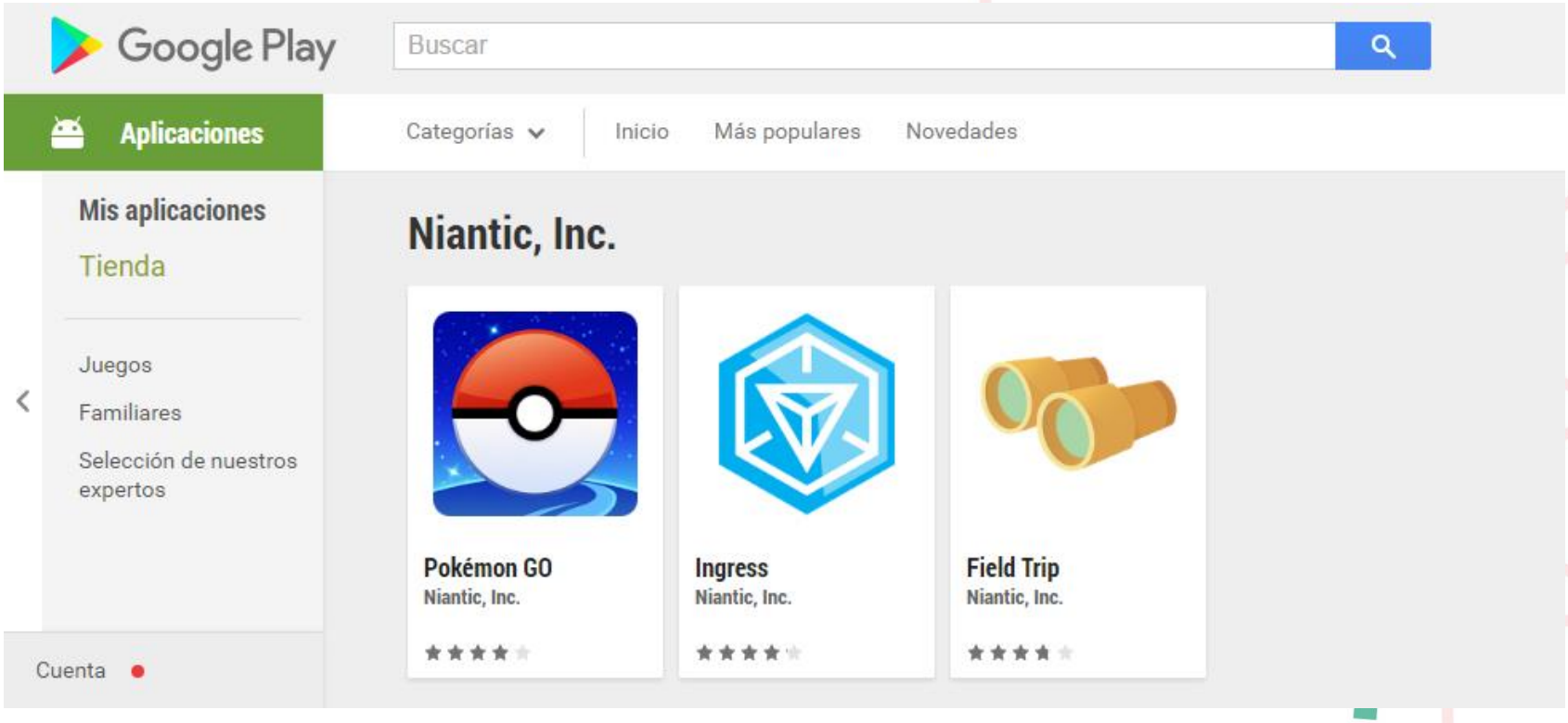
Permisos





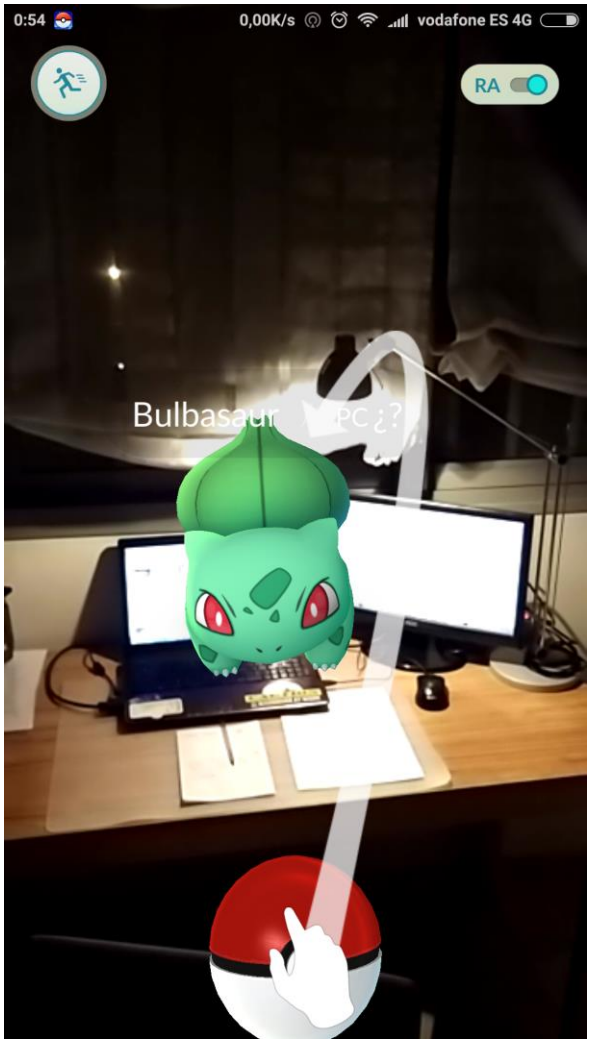
Obteniendo Información

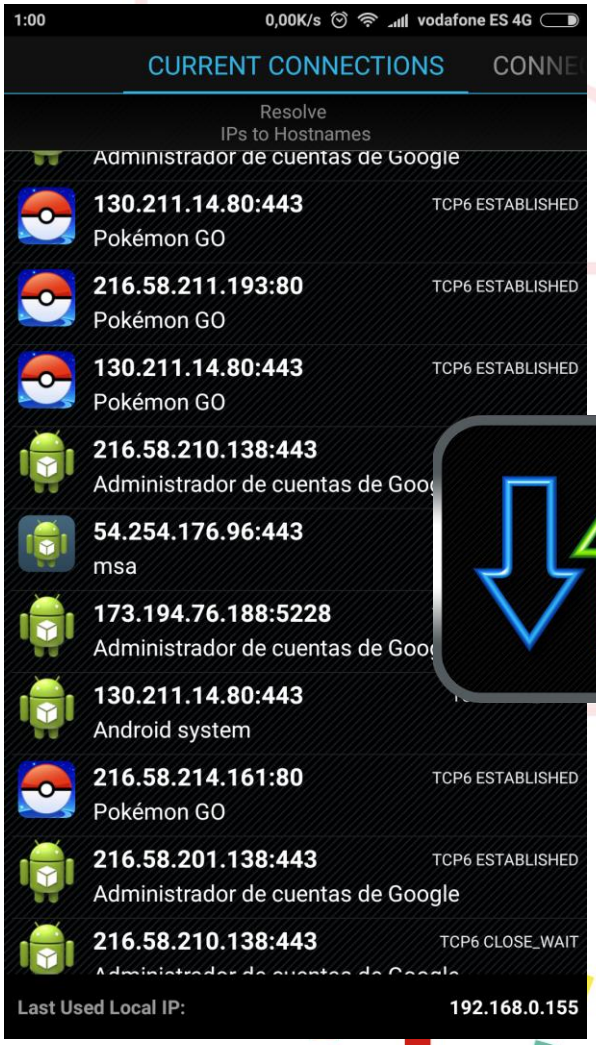
Fabricante



Obteniendo Información

Conexiones





Análisis Estático

Reversing

- **APKTool**

```
andro@l4b:~/Desktop/CyberCamp$ apktool d Pokémon\ GO v0.47.1 apkpure.com.apk
I: Using Apktool 2.0.3 on Pokémon GO_v0.47.1_apkpure.com.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: /home/andro/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...
andro@l4b:~/Desktop/CyberCamp$
```

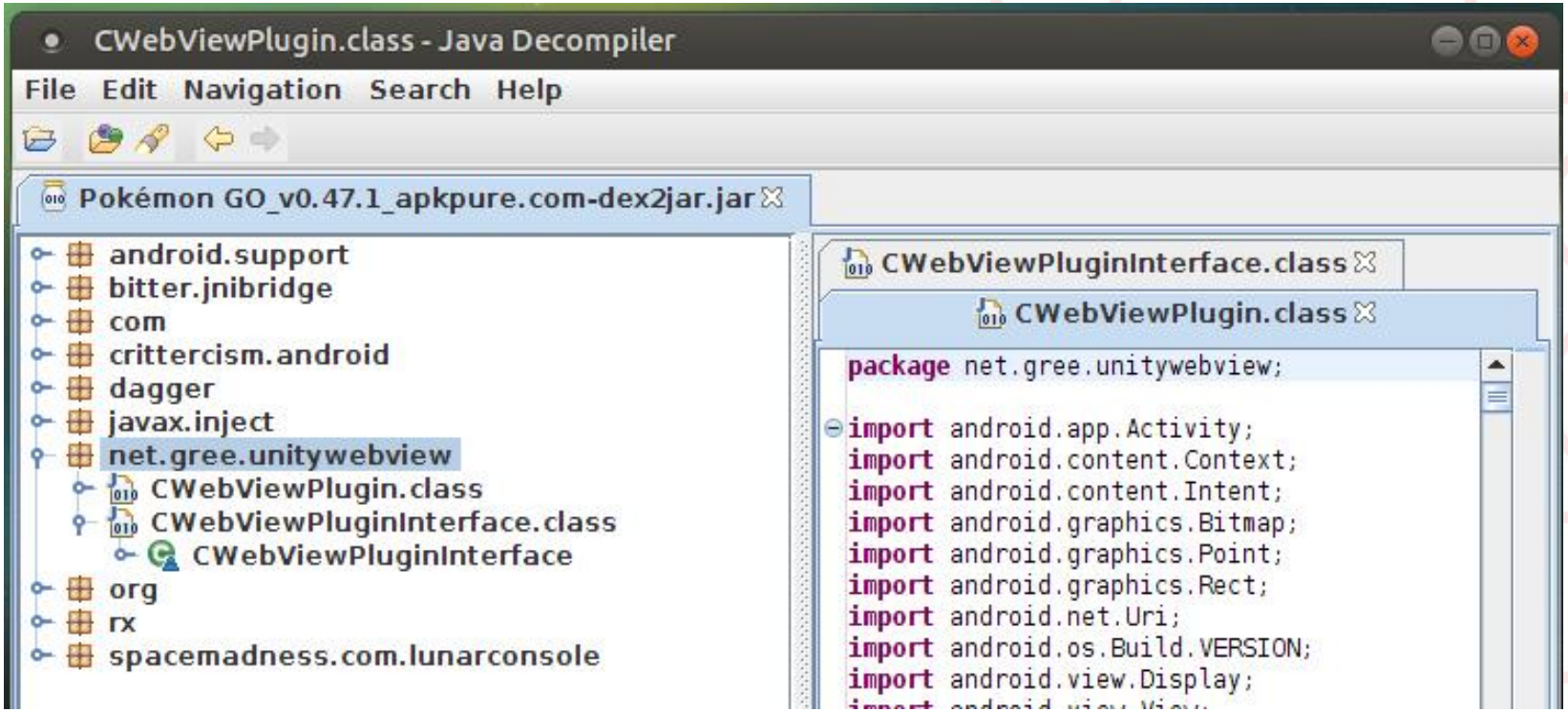
- **DEX2JAR**

```
andro@l4b:~/tools/dex2jar$ sudo ./d2j-dex2jar.sh /home/andro/Desktop/CyberCamp/P
okémon\ GO_v0.47.1_apkpure.com.apk
[sudo] password for andro:
dex2jar /home/andro/Desktop/CyberCamp/Pokémon GO_v0.47.1_apkpure.com.apk -> ./Po
kémon GO_v0.47.1_apkpure.com-dex2jar.jar
andro@l4b:~/tools/dex2jar$ ls P*
Pokémon GO_v0.47.1_apkpure.com-dex2jar.jar
andro@l4b:~/tools/dex2jar$
```

Análisis Estático

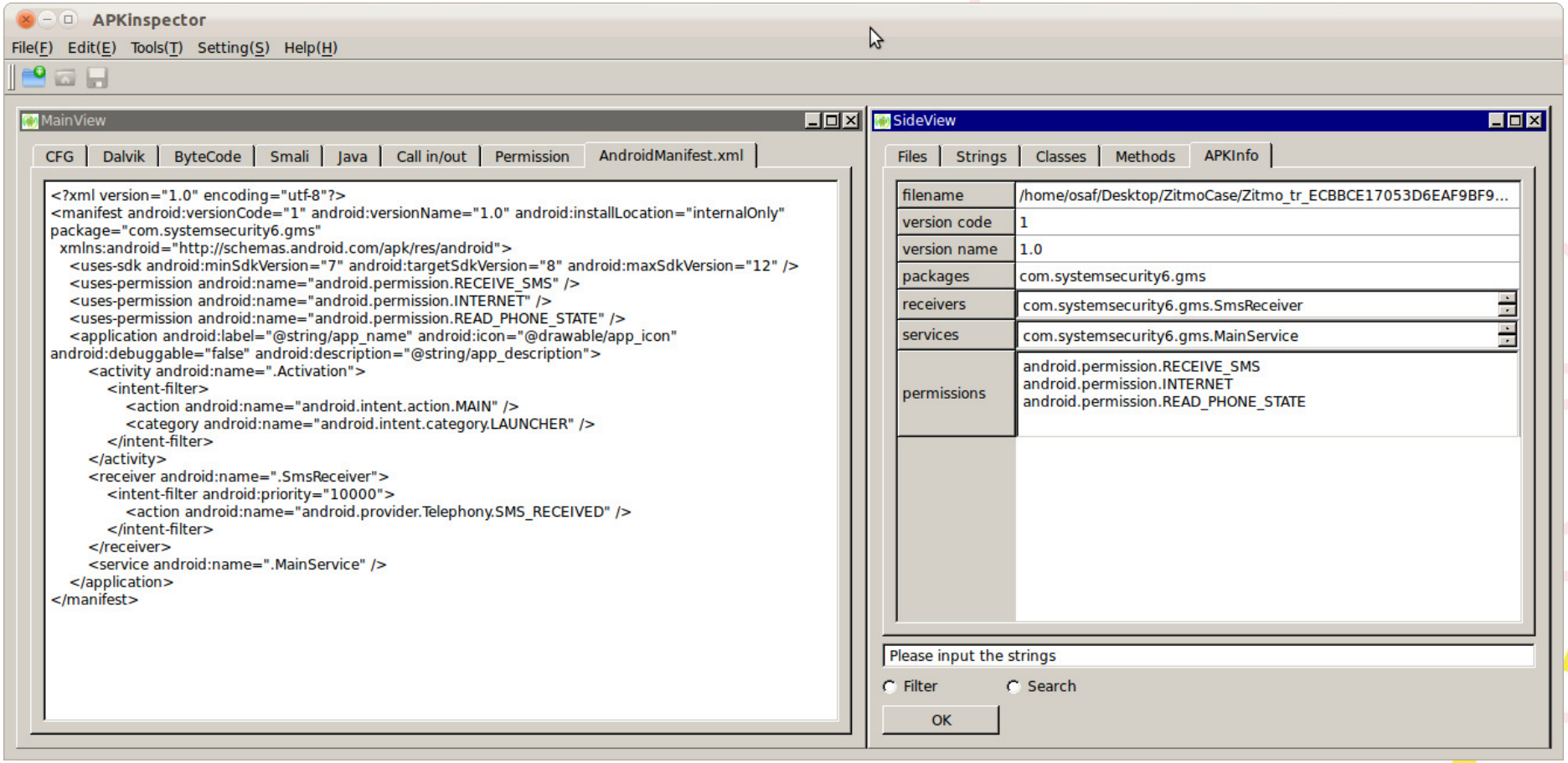
Reversing

- **JD-GUI**



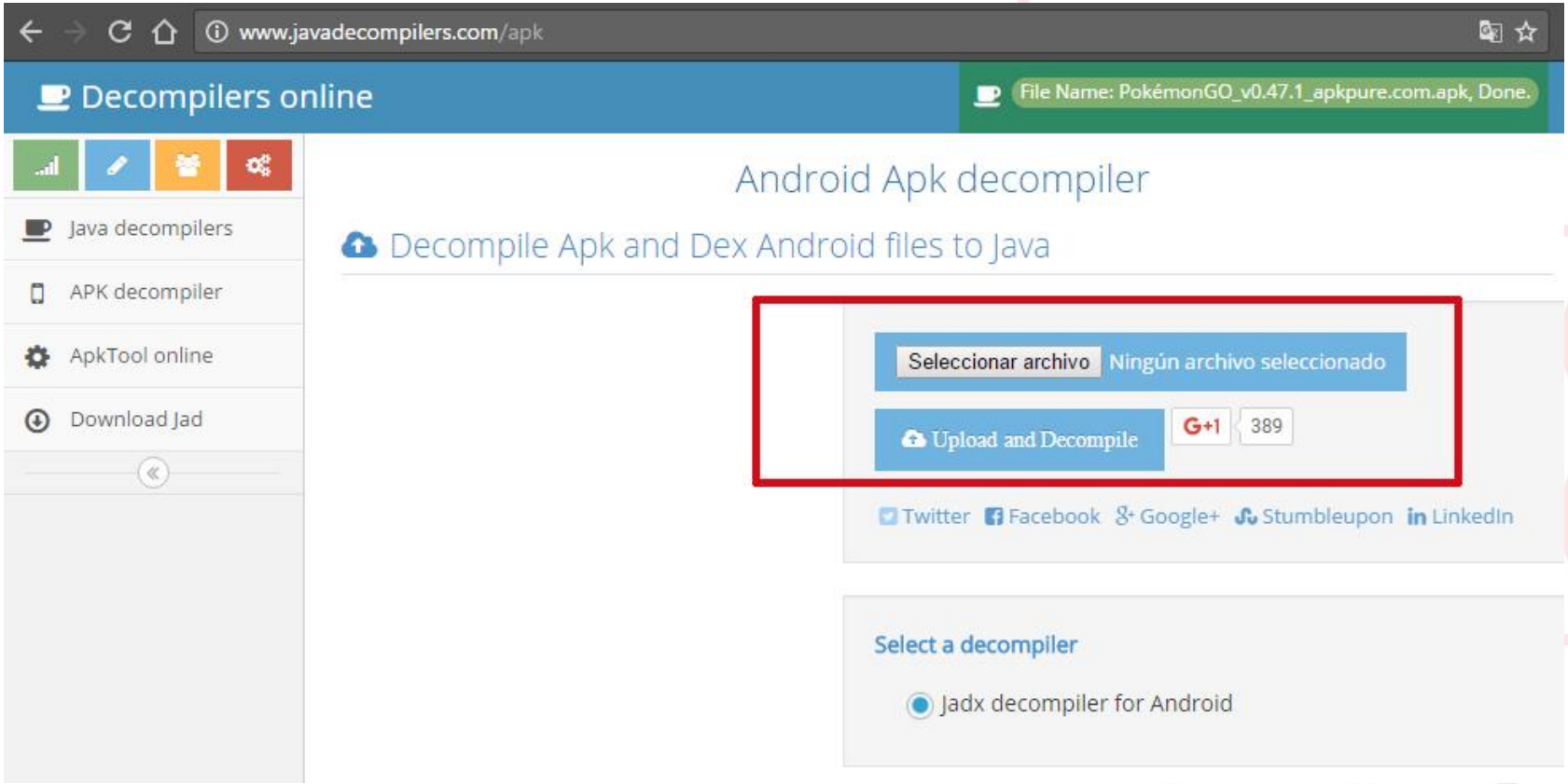
Análisis Estático

APK Inspector



Análisis Estático

Herramientas online



Análisis Estático

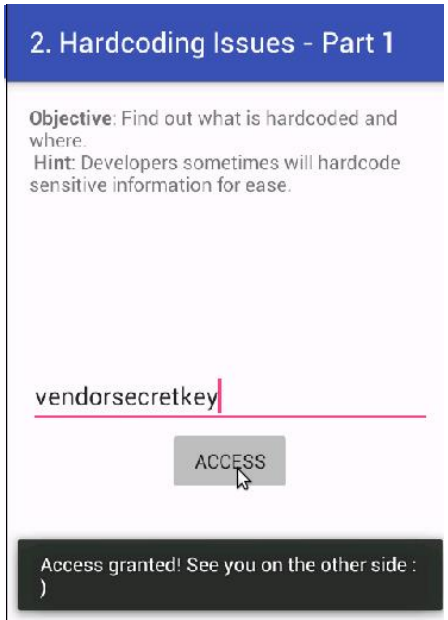
AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:"http://schemas.android.com/apk/res/android" android:versionCode="2016090901" android:versionName="0.37.0" android:installLocation="aut
o" package="com.nianticlabs.pokemongo" platformBuildVersionCode="24" platformBuildVersionName="7.0">
    <supports-screens android:anyDensity="true" android:smallScreens="true" android:normalScreens="true" android:largeScreens="true" android:xlargeScree
ns="true" />
    <uses-permission android:name="com.android.vending.BILLING" />
    <uses-permission android:name="android.permission.VIBRATE" />
    <uses-permission android:name="android.permission.BLUETOOTH" />
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <application android:theme="@style/UnityThemeSelector" android:label="@string/app_name" android:icon="@drawable/app_icon" android:debuggable="false"
 android:banner="@drawable/app_banner" android:isGame="false">
        <activity android:label="@string/app_name" android:name="com.unity3d.player.UnityPlayerNativeActivity" android:launchMode="singleTask" android:s
creenOrientation="portrait" android:configChanges="mcc|mnc|locale|touchscreen|keyboard|keyboardHidden|navigation|orientation|screenLayout|uiMode|screenS
ize|smallestScreenSize|fontScale">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
            <meta-data android:name="unityplayer.UnityActivity" android:value="true" />
            <meta-data android:name="unityplayer.ForwardNativeEventsToDalvik" android:value="true" />
        </activity>
        <activity android:theme="@*android:style/Theme.Black.NoTitleBar" android:name="com.google.nianticproject.platform.AccountsActivity" android:scre
enOrientation="portrait" />
        <service android:name="com.upsight.android.analytics.internal.DispatcherService" />
```

Análisis Estático

Unintended Data Leackage → Hardcoded Passwords

- **Buscar en el código fuente claves o API-keys.**




```
public void access(View paramView)
{
  if (((EditText)findViewById(2131492987)).getText().toString().equals("vendorsecretkey"))
  {
    Toast.makeText(this, "Access granted! See you on the other side :)", 0).show();
    return;
  }
  Toast.makeText(this, "Access denied! See you in hell :D", 0).show();
}
```

```
protected static final int NO_OPTIONS;
String SHAPass = null;
Button btn;
EditText et;
String password = "5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8";
```

```
We found 1 hashes! [Timer: 108 m/s] Please find them below...
5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8
5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8 SHA1 : password
```

Análisis Dinámico

Bypassing Authentication

- En **AndroidManifest.xml** vemos las **Activity** que hay con la propiedad → **exported="true"**

```
<?xml version="1.0" encoding="utf-8"?>
<manifest android:versionCode="1" android:versionName="1.0" package="com.isi.testapp"
  xmlns:android="http://schemas.android.com/apk/res/android">
    <uses-sdk android:minSdkVersion="8" android:targetSdkVersion="18" />
    <application android:theme="@style/AppTheme" android:label="@string/app_name" android:icon="@drawable/
ic_launcher" android:debuggable="true" android:allowBackup="true">
        <activity android:label="@string/app_name" android:name="com.isi.testapp.MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name="com.isi.testapp.Welcome" android:exported="true" />
    </application>
</manifest>
```


```
tamer@ubuntu ~/Desktop/tallerMorteruelo2016/03BypassAuth $ adb shell
root@android:/ # am start -n com.isi.testapp/.Welcome
Starting: Intent { cmp=com.isi.testapp/.Welcome }
root@android:/ #
```

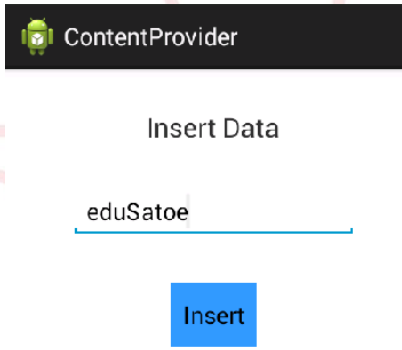
Private Area

Análisis Dinámico

Content Provider Leakage

- En **AndroidManifest.xml** vemos las **Content Provider** que hay con la propiedad → **exported="true"**

```
<?xml version="1.0" encoding="utf-8"?>
<manifest android:versionCode="1" android:versionName="1.0" package="com.isi.contentprovider"
  xmlns:android="http://schemas.android.com/apk/res/android">
    <uses-sdk android:minSdkVersion="8" android:targetSdkVersion="18" />
    <application android:theme="@style/AppTheme" android:label="@string/app_name" android:icon="@drawable/
ic_launcher" android:debuggable="true" android:allowBackup="true">
        <activity android:label="@string/app_name" android:name="com.isi.contentprovider.MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <provider android:name=".MyProvider" android:exported="true"
android:authorities="com.isi.contentprovider.MyProvider" />
    </application>
</manifest>
```



- Encontrar URI en código Smali → **grep -R content:// .**

```
tamer@ubuntu ~/Desktop/tallerMorteruelоCon2016 $ adb shell content query --uri content://com.i
si.contentprovider.MyProvider/udetails
Row: 0 id=3, name=beersForAll
Row: 1 id=1, name=eduSatoe
Row: 2 id=2, name=hackandbeers
tamer@ubuntu ~/Desktop/tallerMorteruelоCon2016 $
```

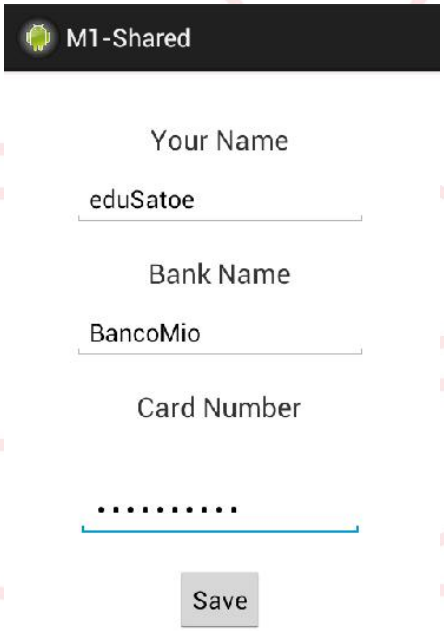
Análisis Dinámico

Insecure Storage

- **Shared Preferences**

GoFitPrefs.xml

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="PASSWORD">morterueloCon.n37</string>
    <string name="idCenter">GF02</string>
    <string name="USER">edu.android-pruebas@gmail.com</string>
</map>
```



- **Files → userinfo.xml**

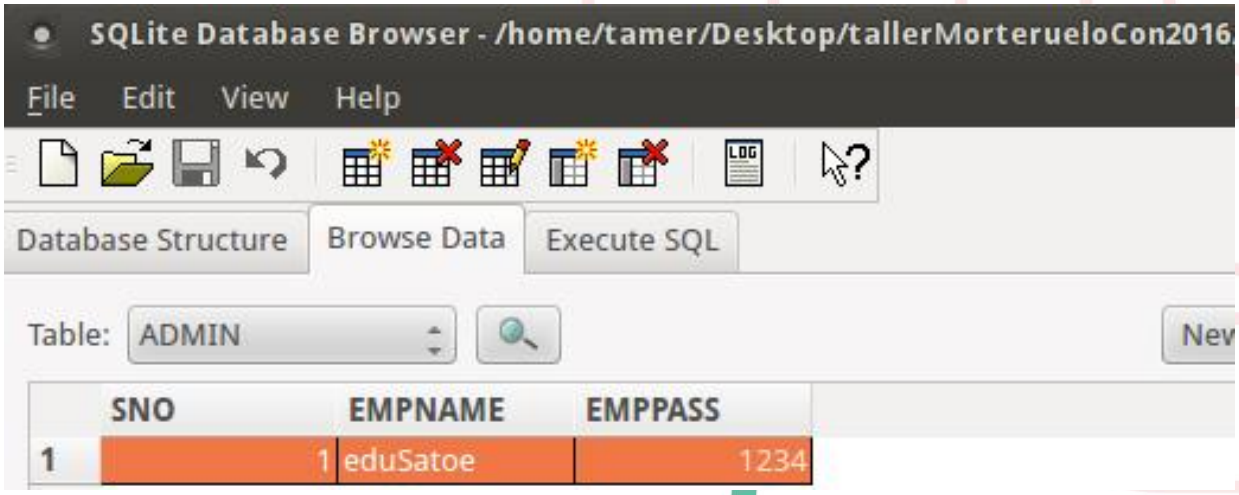
  grep -i password userinfo.xml

Análisis Dinámico

Insecure Storage

- SQLite Databases

- Encontrar información sensible almacenada de manera insegura.

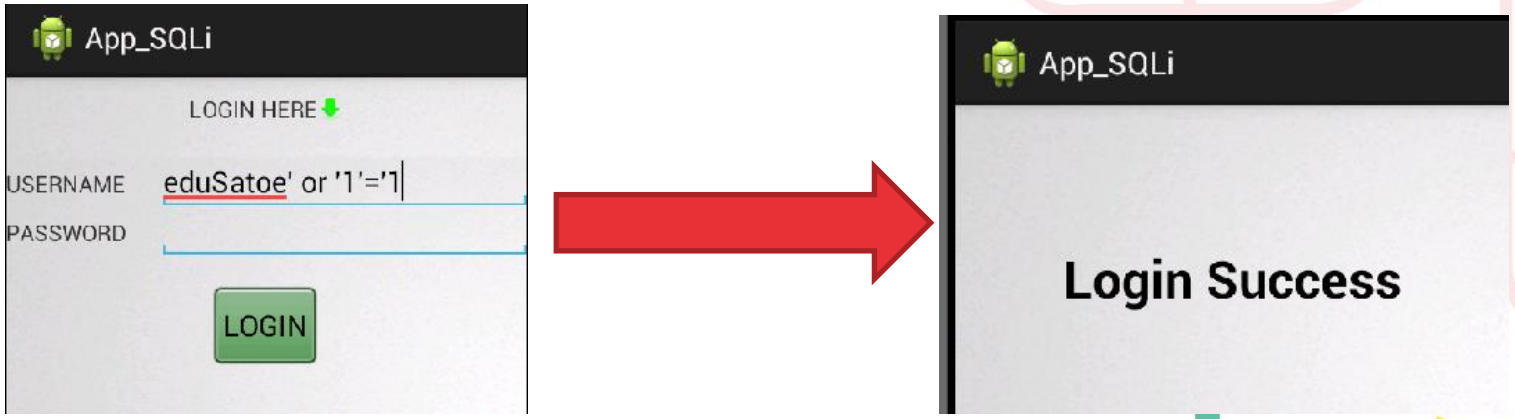


Análisis Dinámico

Client Side Injection → SQL Injection

- **Encontrar posibles parámetros de entrada vulnerables.**

```
public boolean login(String paramString1, String paramString2)
{
  String str = "select * from ADMIN where EMPNAME = '" + paramString1 + "' and EMPPASS = '" + paramString2 + "'";
  if (this.db.rawQuery(str, null).getCount() != 0);
  for (int i = 1; ; i = 0)
    return i;
}
```



Análisis Dinámico

Unintended Data Leackage → Logging

- Comprobamos AndroidManifest.xml para ver si la App tiene la característica → debuggable="true"

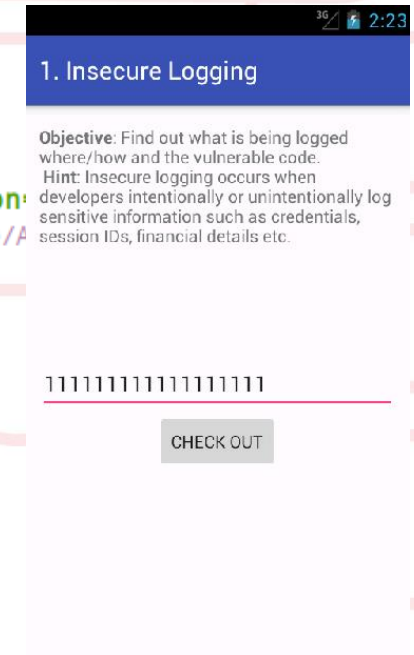
- Lanzamos logcat con script pidcat

```
<application android:allowBackup="true" android:debuggable="true" android:icon
android:label="@string/app_name" android:supportsRtl="true" android:theme="@style/A
```

```
tamer@ubuntu /Arsenal/binary $ python pidcat.py jakhar.aseem.diva
           dalvikvm  D  Not late-enabling CheckJNI (already on)

                        Process jakhar.aseem.diva created for activity jakh
                        ar.aseem.diva/.MainActivity
                        PID: 624   UID: 10056   GIDs: {1015, 1028, 3003}

        diva-log  E  Error while processing transaction with credit card:
                     1111111111111111
```



Análisis Dinámico

Network Analysis

Modos

- **Emulator**

Menu > Settings > Wireless & networks > Mobile Networks > Access Point Names

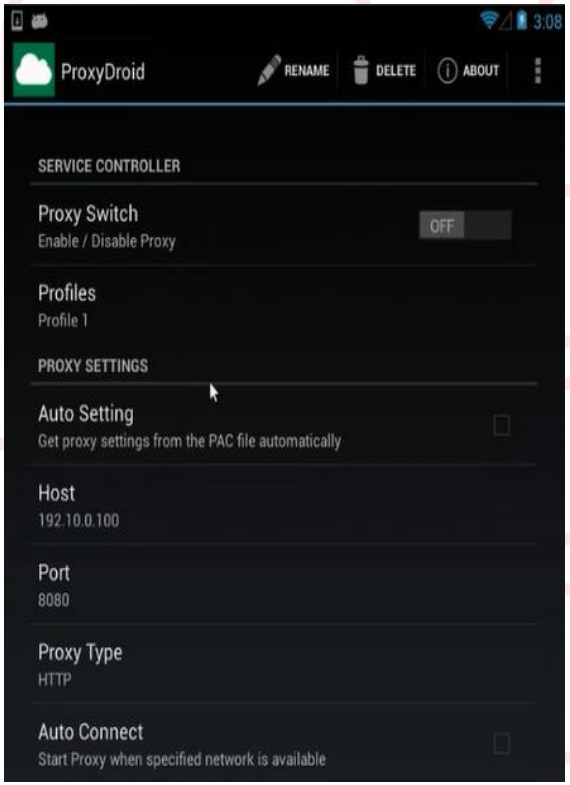
emulator –avd lab hhtp-proxy http://ip:port

- **Device**

ProxyDroid

- **Certificate**

Settings > Security



Análisis Dinámico

Proxy → Burp Suite

```
tamer@ubuntu ~/Desktop/tallerMorterueloCon2016/02NetworkAnalysis $ emulator -avd atavd -force-32bit
-dpi-device 160 -scale 125dpi -http-proxy http://localhost:8080 &
```

forward | drop | intercept is on | action

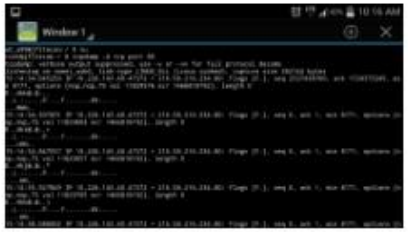
raw | params | headers | hex

```
GET
/ws/goFitServices_v2.ashx?servicio=autenticacion_usuario_mail&idCentro=GF02&mail=edu.android.pruebas@gmail.com&password=morterueloCon.n37&codigo=7CBB5419EDC033E3C50ABC23527103E7
HTTP/1.1
User-Agent: Dalvik/1.6.0 (Linux; U; Android 4.1.2; sdk Build/MASTER)
Host: reservas.go-fit.es
Connection: Keep-Alive
Accept-Encoding: gzip
```

mail=edu.android.pruebas@gmail.com&password=morterueloCon.n37

Análisis Dinámico

TCPdump for ARM



Android TCPDUMP

Home Downloads Compiling Uses About Contact

Welcome to Android *tcpdump*

This is the official site for the *tcpdump* binary for Android devices. Precompiled and ready to go! Always the latest releases! If you want to compile your own? Go to our Compile Section for instructions.

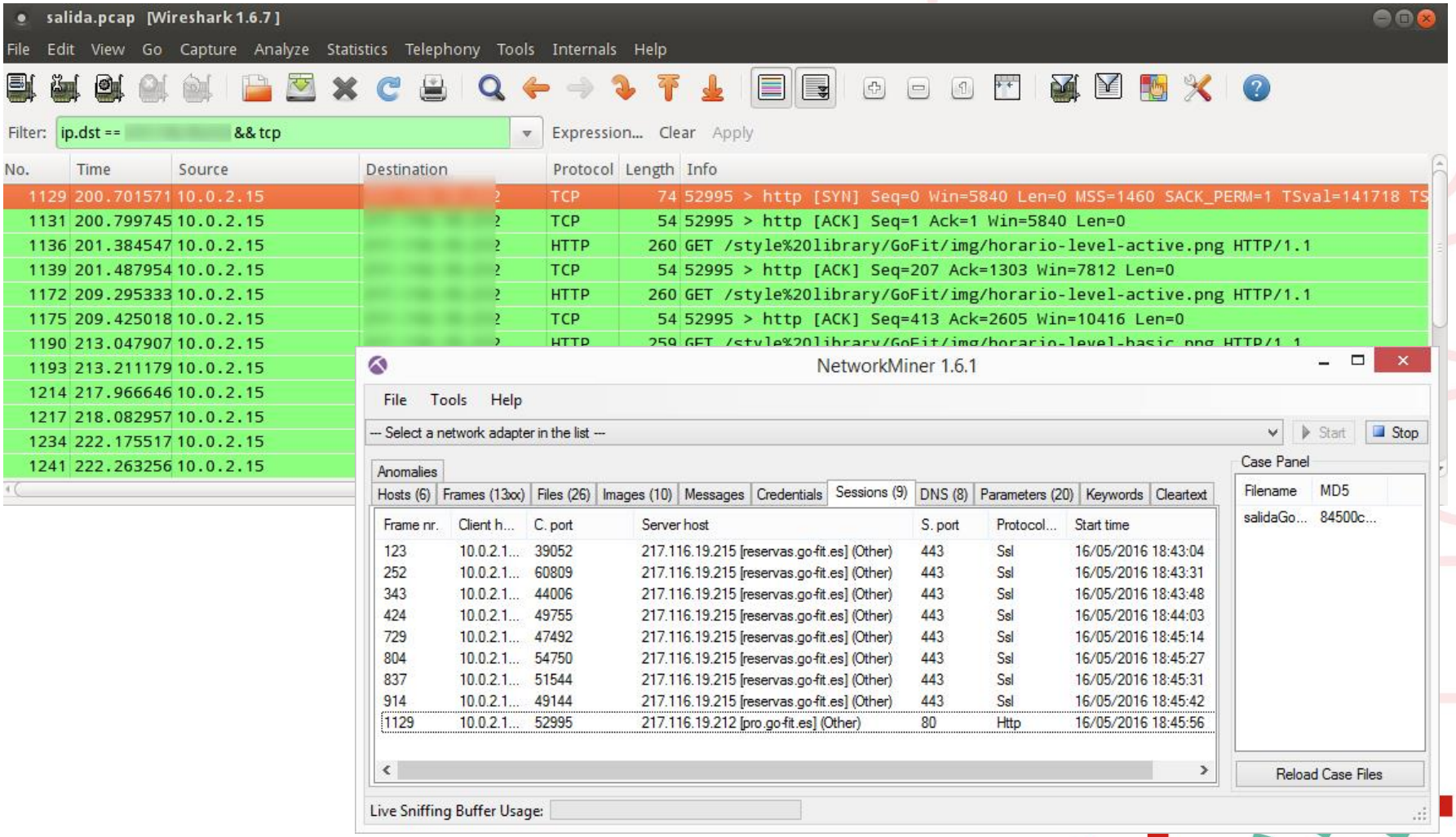
Android *tcpdump* is a command line packet capture utility. It can capture packets from your Wifi connection, Cellular connections, and any other network connections you may have. It has been compiled from the source code located at http://www.tcpdump.org. If you use linux's or unix's *tcpdump* application, then you will be

```
tamer@ubuntu ~/Desktop/SVT $ adb push tcpdump /data/local/tmp/
1650 KB/s (1893728 bytes in 1.120s)
tamer@ubuntu ~/Desktop/SVT $ adb shell
root@android:/ # cd /data/local/tmp/
root@android:/data/local/tmp # tcpdump -v -s 0 -w salida.pcap
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 65535 byte
s
Got 189
```

```
tamer@ubuntu ~/Desktop/SVT $ adb pull /data/local/tmp/salida.pcap
954 KB/s (290200 bytes in 0.297s)
```

Análisis Dinámico

Wireshark + NetworkMiner



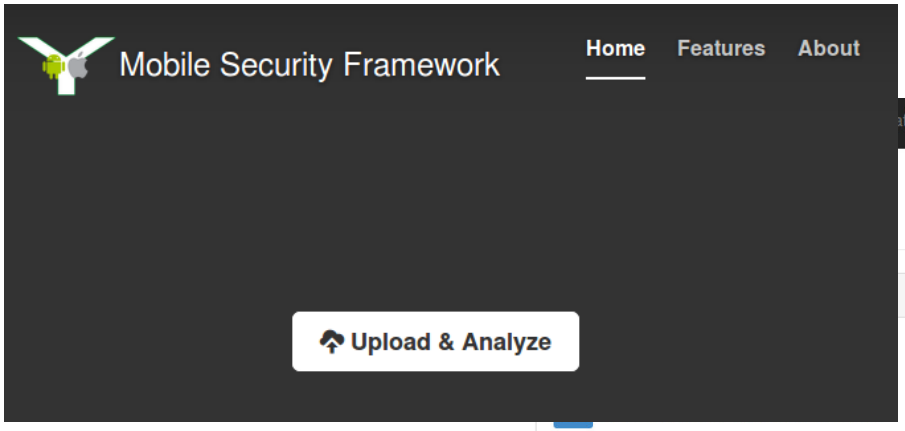
Herramientas Automatizadas

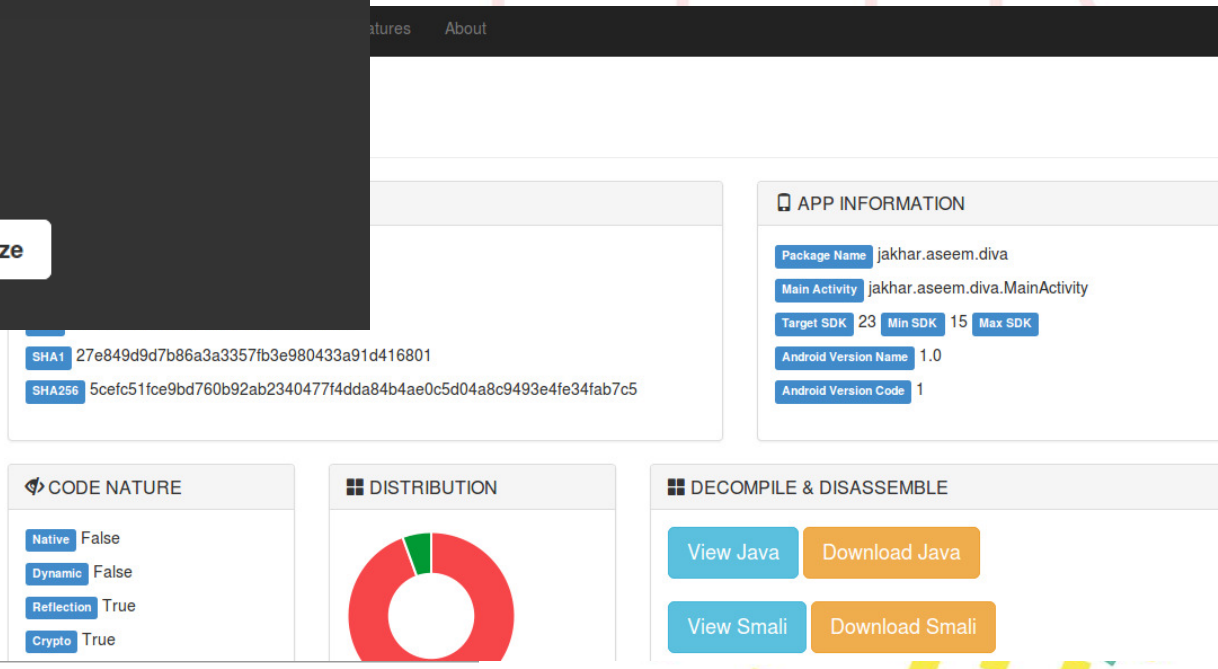
MobSF

- **Iniciamos server.**

```
andro@l4b:~/tools/MobSF$ python manage.py runserver
```

- **Accedemos a localhost:8000**





Herramientas Automatizadas

DroidBox

- **Arrancando la aplicación**

./startmenu.sh nombreAVD

./droidbox.sh APK segundos

- **¿Qué tenemos que mirar?**
  - **Hashes para analizar APK**
  - **Datos entrada/salida**
  - **Operaciones lectura/escritura**
  - **Servicios iniciados**
  - **Fugas de información**
  - **Etc**

Resumen Metodología

Tres fases

- **Recogida de información**

1. **Fabricante en Google Play y ver otras posibles Apps del mismo.**
2. **Instalar la App y manejar todas las posibles Actividades.**
3. **Identificar los posibles puntos de entrada.**
4. **Recursos que utiliza o necesita (GPS, Internet...)**
5. **Posible framework de desarrollo.**
6. **Intentar localizar versiones anteriores (apkpure).**
7. **Tipo de App: Nativa, Web o Híbrida.**
8. **Identificar posibles intenciones a las que llama (tlf, otras Apps...)**
9. **TCPdump activo para una primera fase de análisis.**

NOTA: Tener la check-list delante para ir rellenando toda la info.

Resumen Metodología

Tres fases

- **Análisis Estático**

1. **Llevar a cabo el proceso con APKtool o similares para obtener AndroidManisfest.xml**
2. **Buscar información de la App para ver si es depurable, si tiene actividades, content provider, etc exported=true.**
3. **Obtener el nombre del paquete y del launcher (actividad principal).**
4. **Identificar servicios que pueden correr en paralelo.**
5. **Permisos de la aplicación (¿alguno que no corresponda con el uso de la misma?). ¿Necesita Root? ¿Por qué?**
6. **Versión de la API y posible metadatos sobre el framework que emplea. Lo cual puede llevarnos a información sobre el sistema donde se encuentre puede ser vulnerable.**
7. **Información hardcodeada: API-Keys, Secret-keys, password, etc.**
8. **¿Cómo almacena los datos? (imprescindible hacer un buen testing en la fase 1).**
9. **Otras fallos como posible detección de SQLi u otras vulnearbilidades vista en el código fuente.**
10. **Identificación de socket de conexión en código fuente.**
11. **Búsqueda de cadenas URIS o string con información valiosa.**

NOTA: Tener la check-list delante para ir rellenando toda la info.

Resumen Metodología

Tres fases

- **Análisis Dinámico**

1. **Análisis de puntos de entrada con TCPdump y con Proxy Inverso.**
2. **Monitoreo del log del sistema (pidcat) → logcat.x (E I D W)**
3. **Información que se almacena en el terminal después de interactuar con la aplicación (databases, shared_preferences).**
4. **Análisis dinámico y posibles llamadas a intenciones, servicios u otros recursos (DroidBox).**

NOTA: Tener la check-list delante para ir rellenando toda la info.


cyber
camp
CYBERSECURITY EVENT
2016
Gracias por
su atención
GOBIERNO
DE ESPAÑA
MINISTERIO
DE INDUSTRIA, ENERGÍA
Y TURISMO
incibe_
2006-2016
TRABAJANDO POR
LA CONFIANZA DIGITAL