



3. Ingeniería inversa de aplicaciones móviles Android

Ciberseguridad en Dispositivos móviles
DISCA – ETS de Ingeniería informática (UPV)

Indice

- ¿Qué es Android? Conceptos básicos
- Anatomía de una app Android sencilla
- Herramientas para el desarrollo y emulación de apps Android
- Reversing de apks
- Smali y parcheado de apks

Android Debug Bridge

- Comúnmente conocido como ADB
 - <https://developer.android.com/studio/command-line/adb>
 - Forma parte de las “platform-tools” que integra la SDK de Android
- Permite
 - Enviar órdenes a un dispositivo Android
 - Almacenar y recuperar ficheros del mismo
 - Abrir un Shell en el dispositivo
 - Lectura de información del dispositivo (memoria en uso, paquetes de aplicaciones instaladas, etc).

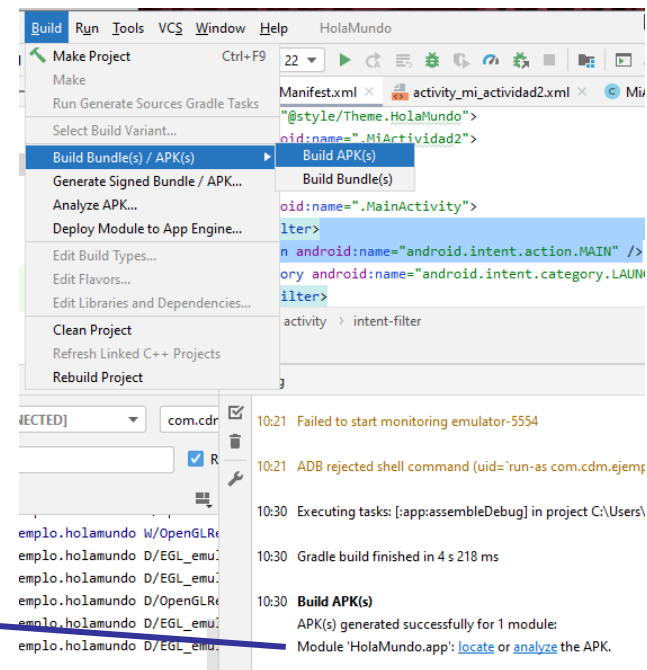
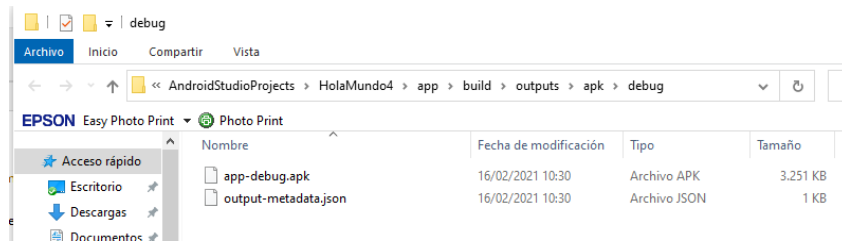
```
# Lists all devices
adb devices
#Result
List of devices attached
emulator-5554 attached
emulator-5555 attached
# Issue a command to a specific device
adb -s emulator-5554 shell
```


Indice

- ¿Qué es Android? Conceptos básicos
- Anatomía de una app Android sencilla
- Herramientas para el desarrollo y emulación de apps Android
- **Reversing de apks**
- Smali y parcheado de apks

Obtención del apk a instalar

- Los primero es saber qué deseamos instalar
 - Apk descargado de algún repositorio o proporcionado por alguien
 - Apk desarrollado por nosotros



Instalar/desinstalar un apk

- Se instalan apks, pero se desinstalan paquetes de aplicación

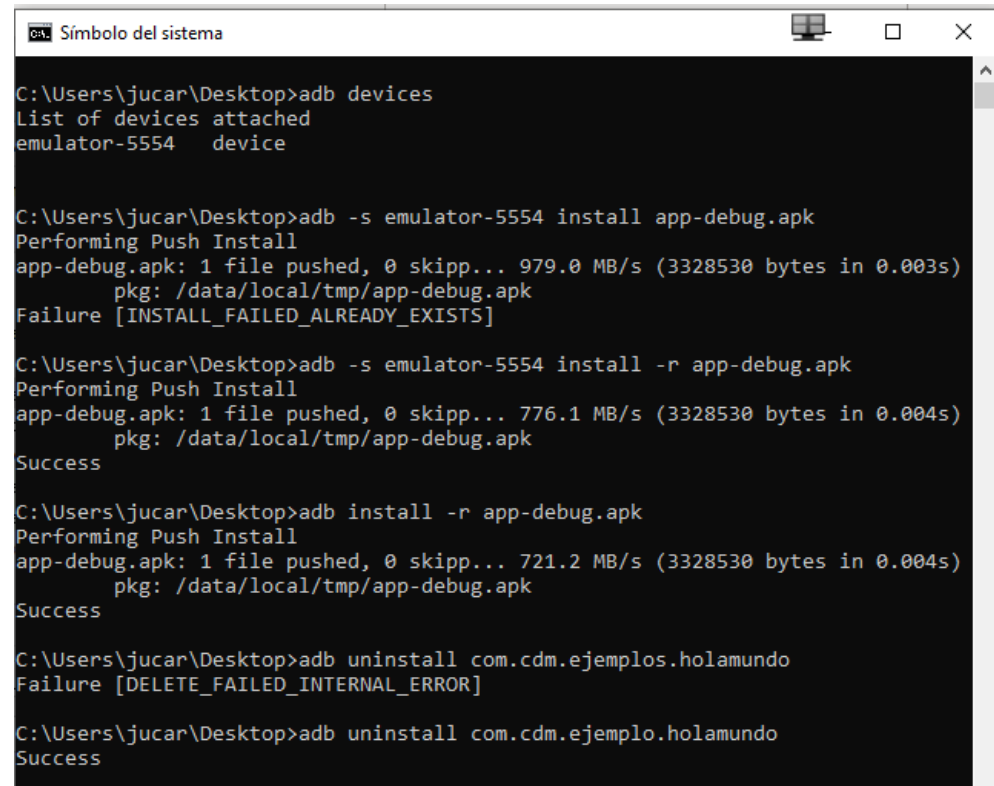
- **Instalación:**

`adb install`

- Opción `-s`
instala la app en la sdcard
- Opción `-r`
fuerza la instalación si la app ya existe

- **Desinstalación:**

`adb uninstall`



```
C:\Users\jucar\Desktop>adb devices
List of devices attached
emulator-5554    device

C:\Users\jucar\Desktop>adb -s emulator-5554 install app-debug.apk
Performing Push Install
app-debug.apk: 1 file pushed, 0 skipp... 979.0 MB/s (3328530 bytes in 0.003s)
pkg: /data/local/tmp/app-debug.apk
Failure [INSTALL_FAILED_ALREADY_EXISTS]

C:\Users\jucar\Desktop>adb -s emulator-5554 install -r app-debug.apk
Performing Push Install
app-debug.apk: 1 file pushed, 0 skipp... 776.1 MB/s (3328530 bytes in 0.004s)
pkg: /data/local/tmp/app-debug.apk
Success

C:\Users\jucar\Desktop>adb install -r app-debug.apk
Performing Push Install
app-debug.apk: 1 file pushed, 0 skipp... 721.2 MB/s (3328530 bytes in 0.004s)
pkg: /data/local/tmp/app-debug.apk
Success

C:\Users\jucar\Desktop>adb uninstall com.cdm.ejemplos.holamundo
Failure [DELETE_FAILED_INTERNAL_ERROR]

C:\Users\jucar\Desktop>adb uninstall com.cdm.ejemplo.holamundo
Success
```

Conexión de un dispositivo

- Hay que habilitar la depuración por USB
 - Conexión por USB
(emulador o dispositivo real)
 - Conexión por WIFI (dispositivo real)
 - Conectar el dispositivo a la misma red que el PC
 - Conectar el dispositivo por USB al PC
 - Aceptar la conexión
 - Si no se acepta tendréis un mensaje de error
 - Definir el puerto de comunicación
 - `adb tcpip puerto`
 - Averiguar la ip del dispositivo
 - `adb shell ip addr show wlan0`
 - Definir la conexión con el dispositivo
 - `adb connect ip-dptvo:puerto`
 - Desconectar el dispositivo y aceptar en el mismo la petición de conexión que el PC emitirá regularmente
 - Una vez todos los pasos realizados podremos interactuar con el dispositivo como si este estuviera conectado en local
 - ¡¡ No olvidar volver al modo usb cuando sea necesario !!

```
Símbolo del sistema
C:\Users\jucar\Desktop>adb devices
List of devices attached
8ea5baef          unauthorized

C:\Users\jucar\Desktop>adb tcpip 5555
error: device unauthorized.
This adb server's $ADB_VENDOR_KEYS is not set
Try 'adb kill-server' if that seems wrong.
Otherwise check for a confirmation dialog on your device.

C:\Users\jucar\Desktop>adb tcpip 5555
restarting in TCP mode port: 5555

C:\Users\jucar\Desktop>adb shell ip addr show wlan0
32: wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1460 qdisc mq state UP group
default qlen 3000
    link/ether bc:2d:ef:8e:37:f7 brd ff:ff:ff:ff:ff:ff
    inet 192.168.18.63/24 brd 192.168.18.255 scope global wlan0
        valid_lft forever preferred_lft forever
    inet6 fe80::be2d:efff:fe8e:37f7/64 scope link
        valid_lft forever preferred_lft forever

C:\Users\jucar\Desktop>adb connect 192.168.18.63:5555
failed to authenticate to 192.168.18.63:5555

C:\Users\jucar\Desktop>adb devices
List of devices attached
192.168.18.63:5555    device

C:\Users\jucar\Desktop>adb install app-debug.apk
Performing Streamed Install
Success

C:\Users\jucar\Desktop>adb usb
restarting in USB mode
```

Órdenes adb shell (1/2)

- Entramos en el intérprete de comandos de Android, que proviene de LINUX
- Podemos utilizar órdenes como **ls, cd, rm, mkdir, touch, pwd, cp, mv**
 - ls: listado de archivos
 - cd: cambia de directorio
 - mkdir: crea un directorio
 - pwd: en qué directorio estoy?
 - cp: copia archivo o directorio
 - mv: mueve o cambia el nombre de archivo o directorio

Órdenes adb shell (2/2)

- Averiguar si existe conexión a internet
 - `ping www.upv.es` (Ctfl+C para parar)
- Ver parámetros de red
 - `ip -f inet addr show wlan0`
- Ver IP del dispositivo
 - `netcfg`
- ¿Qué IPs están conectadas al móvil?
 - `netstat`

Órdenes interesantes (1/2)

- Listado de paquetes instalado

```
adb shell pm list packages
```

```
adb shell pm list packages -f          (Paquetes y sus archivos)
```

- Averiguar qué directorio está instalado un paquete:

```
adb shell pm path com.android.phone
```

```
adb shell pm path com.twitter.android
```

- Borra los datos que se han creado con ese paquete

```
adb shell pm clear com.aplicacion.android
```

- Traer un archivo del móvil al ordenador

```
adb pull /sdcard/demo.mp4 C:\midirectorio
```

```
adb pull /sdcard/micancion.mp3 C:\midirectorio\miscanciones
```

- Depositar un archivo del ordenador en el móvil

```
adb push C:\aplicacion.apk /sdcard
```

```
adb push d:\test.apk /sdcard/canciones
```

Órdenes interesantes (2/2)

- Muestra un registro de los eventos que han ocurrido en el móvil (Ctrl+C para parar)
`adb logcat`
- Información del sistema
`adb shell dumphys` (toda la información)
`adb shell dumphys meminfo` (información de memoria)
`adb shell dumphys battery` (información de la batería)
- Obtener una captura de pantalla:
`adb shell screencap /sdcard/screen.png`
- Captura en video lo que se esté viendo en la pantalla:
`adb shell screenrecord /sdcard/demo.mp4`
- Información del dispositivo. Imei, número de serie, fecha de instalación, chip,... :
`adb shell getprop`
- Procesos en ejecución en el dispositivo
`adb shell ps`
- Uso del teléfono
`adb shell am start -a android.intent.action.CALL -d tel:123456789`
`adb shell am start -a android.intent.action.SENDTO -d sms:616964638
--es sms_body Hi --ez exit_on_sent true`

Interacción con apps

- Lanzar a ejecución una actividad

```
adb shell am start -n <package>/<activity>
```

- Simulate key strokes

```
adb shell input keyevent <code:4 -back, 3 -home>
```

- Dumpsys activity

```
adb shell dumpsys activity -h
```

- Puede ser interesante

```
adb shell dumpsys activity top
```

```
adb shell dumpsys activity package <package>
```

```
adb shell dumpsys service <package>/<service-name>
```


Envío de órdenes al emulador

- Tal y como se dice en <https://developer.android.com/studio/run/emulator-console>

- Inicialmente es necesario autenticarse

```
$ telnet localhost 5554
Trying ::1...
telnet: connect to address ::1: Connection refused
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Android Console: Authentication required
Android Console: type 'auth <auth_token>' to authenticate
Android Console: you can find your <auth_token> in
'/Users/me/.emulator_console_auth_token'
OK
auth 123456789ABCdefZ
Android Console: type 'help' for a list of commands
OK
help-verbose
```

- Una vez autenticado es posible controlar el emulador desde la línea de comandos

```
telnet localhost 5554
# set the power level
power status full
power status charging
# make a call to the device
gsm call 012041293123
# send a sms to the device
sms send 12345 Will be home soon
# set the geo location
geo fix 48 51
```

Nota: Cuidado si trabajamos en Windows porque Telnet no está activo por defecto

Indice

- Herramientas para el desarrollo y emulación de aplicaciones Android
- Depuración con ADB
- **Reversing de apks**
- Smali y parcheado de apks

AAPT

- Android Asset Packaging Tool
 - Permiter revisar, crear y actualizar ficheros APK
 - Facilita la compilación de recursos para convertirlos en activos binarios
 - Es la herramienta que utiliza el builder Android para producir las apks
 - Disponible dentro del directorio
Android\Sdk\build-tools

Uso de AAPT

- aapt usage
- aapt list chrome.apk
- aapt dump permissions chrome.apk
- aapt dump strings chrome.apk

Contenido de un apk

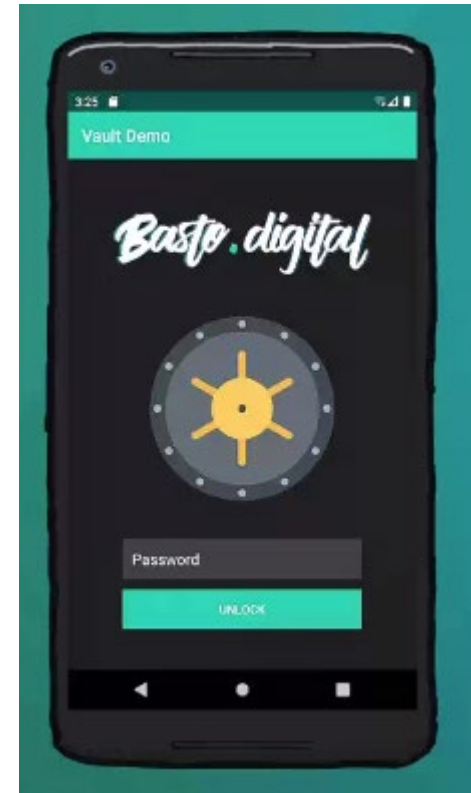
■ Propósito

- A menudo el código fuente de las apps contiene información sensible (passwords, tokens para el uso de APIs, credenciales de acceso a BBDDs, etc.)
- Analizar el código de la app nos permite comprender mejor su funcionalidad y buscar vulnerabilidades

■ Ejemplo: Vault.apk

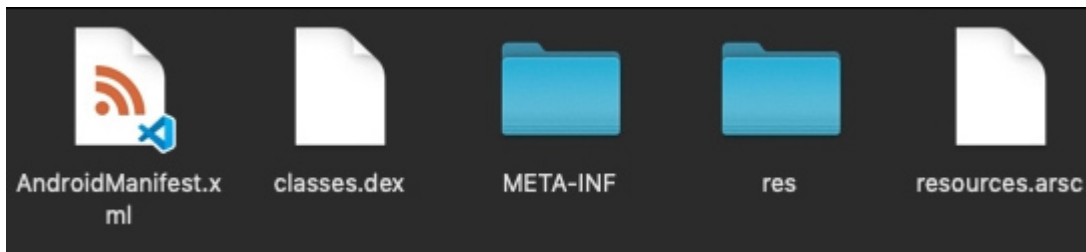
- <https://basto.digital/download/vault.apk>

```
unzip -d vault-unzip vault.apk
```

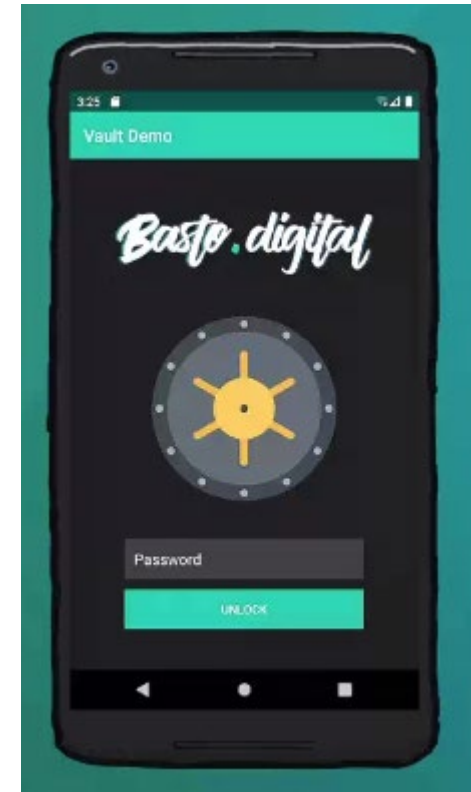


Contenido de un apk

```
unzip -d vault-unzip vault.apk
```



```
AndroidManifest.xml
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3     package="com.baste.digital.vault"
4     android:versionCode="1"
5     android:versionName="1.0"
6     android:installLocation="auto"
7     android:allowBackup="true"
8     android:fullBackupContent="true"
9     android:usesCleartextTraffic="true"
10     android:theme="@style/AppTheme"
11 />
```



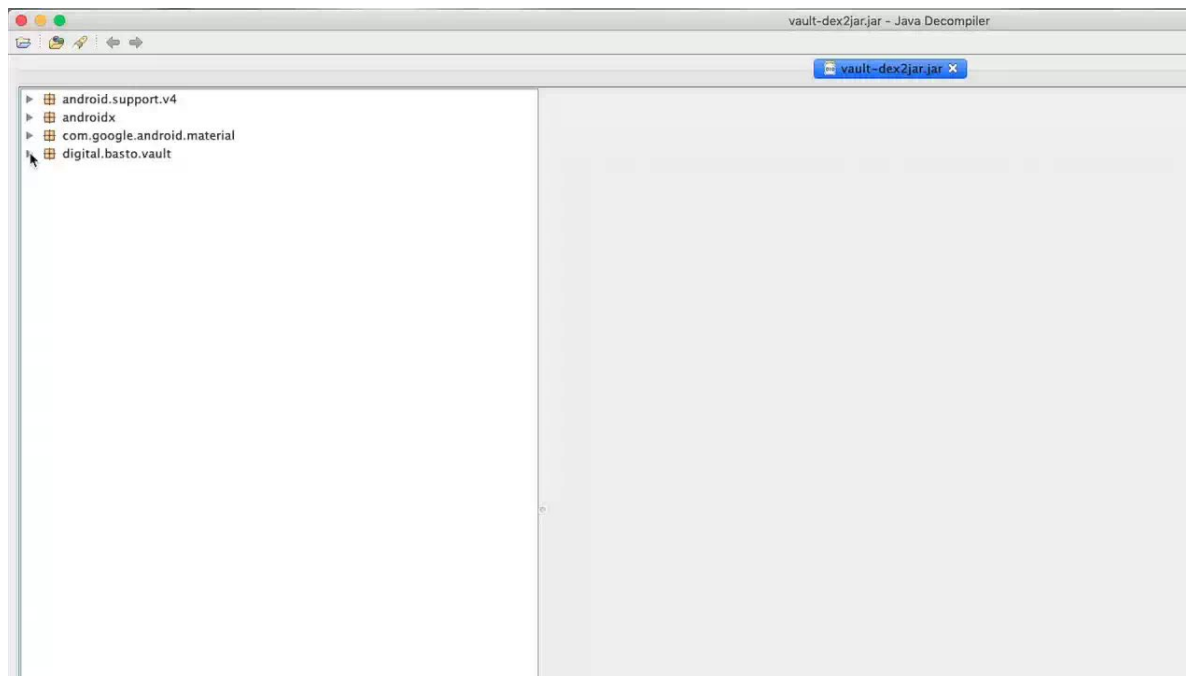
El fichero classes.dex contiene el Código de la app en formato binario (formato Dalvik Executable, o DEX)

Desensamblar vs. Decompilar apps

- Desensamblar es transformar el código binario (DEX) en código máquina de bajo nivel que ya puede ser leído por un humano (smali)
 - El código smali puede ser modificado
 - Es posible reensamblar la app con posterioridad
- Decompilar consiste en transformar el código binario (dex) o de bajo nivel (smali) en código de alto nivel (normalmente Java)
 - Similar al original, pero rara vez idéntico
 - La calidad del código Java obtenido dependerá del decompilador utilizado

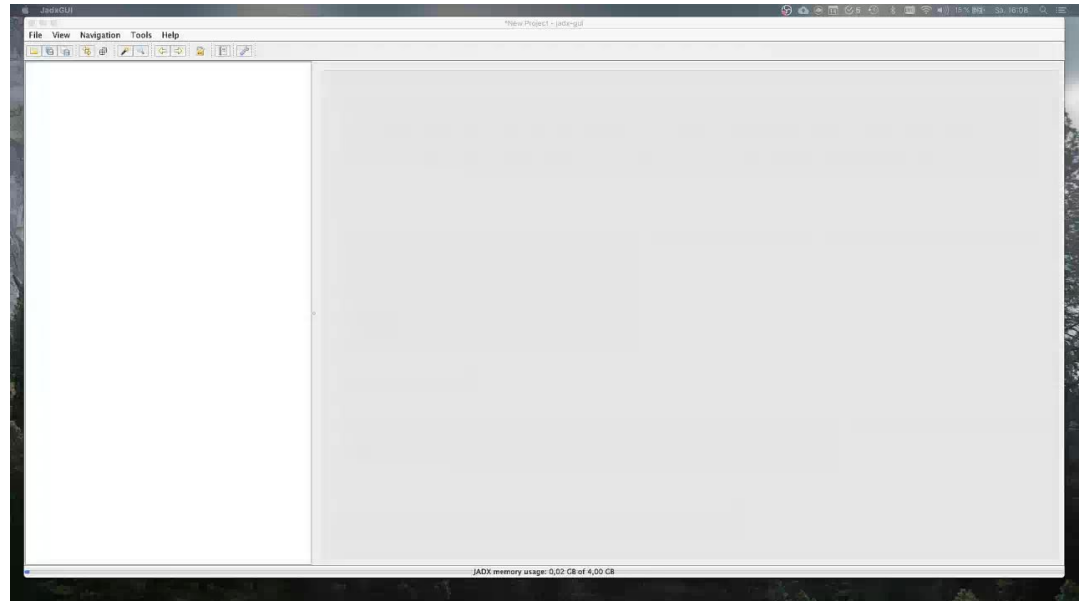
DEX → JAR → JAVA

- Primero, uso de dex2jar
 - <https://github.com/pxb1988/dex2jar>
- A continuación, utilizar jd-gui (decompilador java)
 - <https://java-decompiler.github.io>



DEX → JAVA

- Alternativa 1: JADX



- <https://github.com/skylot/jadx>

- Alternativa 2: Smali2Java

- <https://github.com/AlexeySoshin/smali2java>

- Uso interno de la herramienta apktool

Apktool

■ Uso de apktool

— <https://ibotpeaches.github.io/Apktool>

```
C:\Users\jucar\Downloads>apktool d vault.apk
I: Using Apktool 2.5.0 on vault.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: C:\Users\jucar\AppData\Local\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...
```

Simbolo del sistema

```
C:\Users\jucar\Downloads\vault>dir
El volumen de la unidad C no tiene etiqueta.
El número de serie del volumen es: D0B5-19E1

Directorio de C:\Users\jucar\Downloads\vault

19/02/2021  07:48    <DIR>          .
19/02/2021  07:48    <DIR>          ..
19/02/2021  07:48                1.557 AndroidManifest.xml
19/02/2021  07:48                2.317 apktool.yml
19/02/2021  07:48    <DIR>          original
19/02/2021  07:48    <DIR>          res
19/02/2021  07:48    <DIR>          smali
                2 archivos              3.874 bytes
                5 dirs 77.540.388.864 bytes libres
```

```
C:\Users\jucar\Downloads\vault>
```

Reensamblado

- (Re)generación del apk

```
C:\Users\jucar\Downloads>apktool b -o new-vault.apk vault
I: Using Apktool 2.5.0
I: Checking whether sources has changed...
I: Smaling smali folder into classes.dex...
I: Checking whether resources has changed...
I: Building resources...
I: Building apk file...
I: Copying unknown files/dir...
I: Built apk...
```

- Alineado

```
zipalign.exe -p 4 new-vault.apk new-vault.alineado.apk
```

- Si no existe, generación de un almacén de firmas

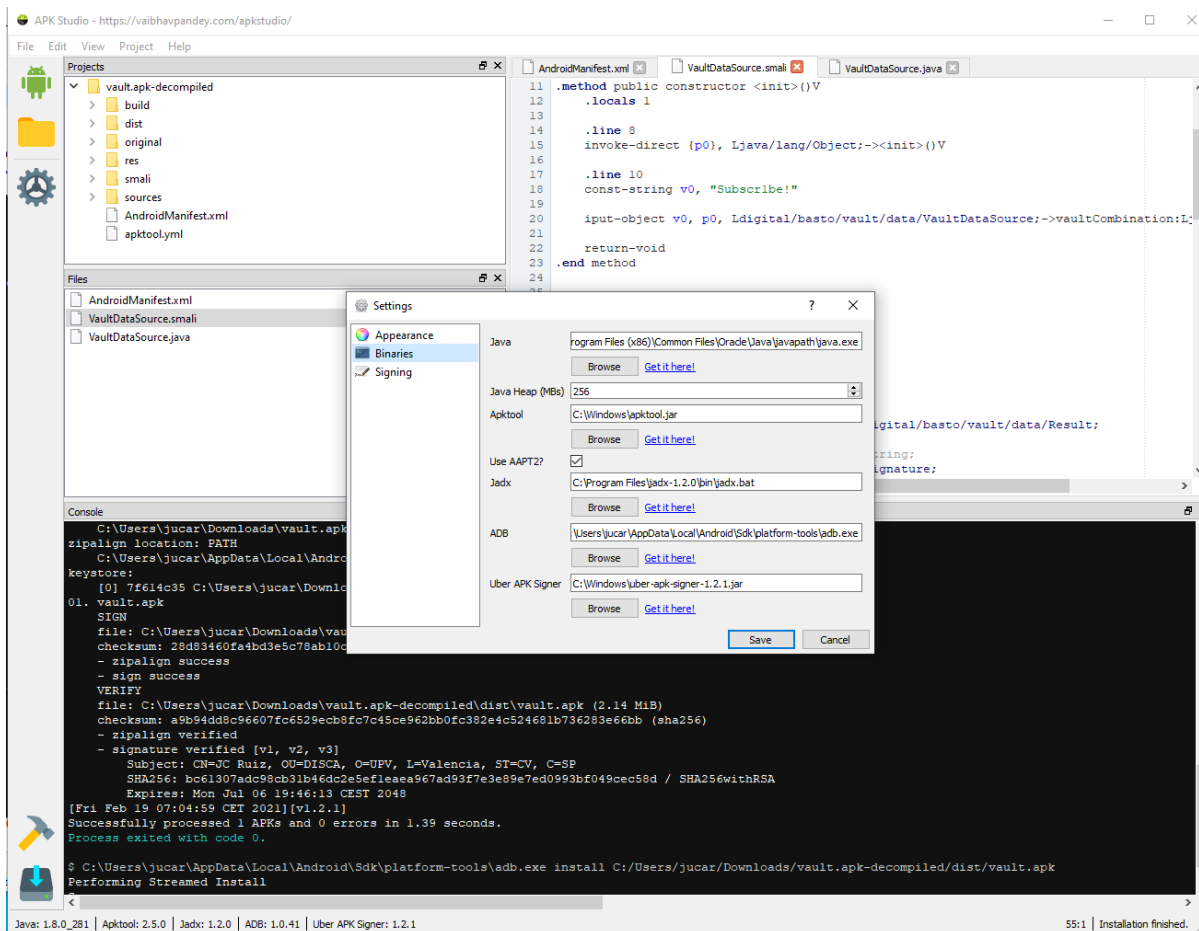
```
keytool -genkey -v -keystore nombrellave.keystore -alias usuario -keyalg RSA -keysize 2048 -validity 10000
```

- Firma del apk

```
apksigner.bat sign --ks Store2.jks --out app-alineada-firmada.apk app-alineada.apk
```

APK Studio

- <https://github.com/vaibhavpandeyvpz/apkstudio>



Indice

- ¿Qué es Android? Conceptos básicos
- Anatomía de una app Android sencilla
- Herramientas para el desarrollo y emulación de apps Android
- Reversing de apks
- **Smali y parcheado de apks**

Código Smali

- Lo trabajaremos en prácticas

```
@Override
public void onClick(View v) {

    startActivity(new Intent(getApplicationContext(), LoginActivity.class));

}
```

```
37 # virtual methods
38 .method public onClick(Landroid/view/View;)V
39     .locals 3
40
41     .line 26
42     iget-object p1, p0, Les/upv/cdm/jcruizg/holamundo/MainActivity$1;->this$0:Les/upv/cdm/jcruizg/holamundo/MainActivity;
43
44     new-instance v0, Landroid/content/Intent;
45
46     invoke-virtual {p1}, Les/upv/cdm/jcruizg/holamundo/MainActivity;->getApplicationContext()Landroid/content/Context;
47     move-result-object v1
48     const-class v2, Les/upv/cdm/jcruizg/holamundo/LoginActivity;
49     invoke-direct {v0, v1, v2}, Landroid/content/Intent;-><init>(Landroid/content/Context;Ljava/lang/Class;)V
50     invoke-virtual {p1, v0}, Les/upv/cdm/jcruizg/holamundo/MainActivity;->startActivity(Landroid/content/Intent;)V
51
52     return-void
53 .end method
```

<https://github.com/JesusFreke/smali>



3. Ingeniería inversa de aplicaciones móviles Android

Ciberseguridad en Dispositivos móviles
DISCA – ETS de Ingeniería informática (UPV)