



5. Vulnerabilidades en apps Android: Análisis dinámico

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



Indice

- Introducción al análisis dinámico de aplicaciones móviles
- Supervisión de la ejecución de un apk utilizando un depurador
- Control de la ejecución de un apk con Frida
- Casos de uso



Análisis dinámico: concepto

- El análisis dinámico de una app consiste en analizar las propiedades de una aplicación mediante su ejecución, comprobando y verificando las acciones que ésta realiza
- Durante el análisis dinámico, se observan las siguientes propiedades de una aplicación:
 - Información almacenada en la memoria del proceso de la aplicación
 - Los ficheros generados por la misma
 - Su flujo de ejecución
 - Componentes exportados, es decir, disponibles a otras aplicaciones
 - Gestión de APIs sensibles del sistema
 - Las conexiones de red creadas

Análisis dinámico: Elementos analizables

- Para realizar el análisis dinámico, es necesario disponer de un dispositivo o emulador en el que ejecutar la aplicación
- Dadas las restricciones que imponen los sistemas operativos móviles, es posible que necesitemos utilizar un dispositivo con jailbreak o rooteado para poder analizar todos los elementos de la aplicación, sobre todo si no disponemos de su código fuente
- El análisis dinámico permite:
 - Verificar la existencia de vulnerabilidades identificadas durante el análisis estático de la aplicación, incluyendo:
 - Transmisión de datos de forma insegura a través de la red
 - Almacenamiento inseguro
 - Componentes de la aplicación expuestos
 - Posibles problemas de configuración de la aplicación
 - Vulnerabilidades existentes en el back-end de la aplicación
 - Fallos en la validación de datos de entrada y arquitectura de la aplicación.



Análisis dinámico: Metodología

- Durante la realización del análisis dinámico, es recomendable llevar un plan de los criterios de seguridad a revisar y las acciones que se han de llevar a cabo para la comprobación de cada criterio
- Para ello, se recomienda:
 - Listar los elementos específicos de una aplicación que vamos a analizar
 - Preparar el laboratorio de análisis para que todos los elementos analizados sean accesibles mediante las herramientas que tenemos instaladas
- Establecer la plantilla de un informe con la siguiente estructura:
 - Un resumen ejecutivo que describa los principales resultados del análisis
 - Un epígrafe por elemento analizable:
 - Dentro de cada epígrafe, una subsección con el elemento específico de la aplicación analizado para verificar el elemento en cuestión
 - Describir en cada subsección las operaciones realizadas y los resultados obtenidos.
- Preparar los ficheros binarios y el dispositivo para el análisis
- Realizar el análisis utilizando el informe como guía



Análisis dinámico: Preparación

- Para poder llevar a cabo todas las tareas que requiere el análisis dinámico y forense, es necesario que las aplicaciones permitan la realización de copias de seguridad (*backup*) y su depuración (*debugging*)
 - Las aplicaciones en producción no permiten realizar este tipo de actividades
 - Al tener acceso al binario, podemos modificarlo para activar estas opciones y poder realizar así un análisis dinámico y forense más completo
- Para ello debemos (ya lo sabemos hacer):
 - Desempaquetar el binario
 - Modificar en el manifiesto los parámetros requeridos

Cambiar: `<application android:allowBackup="false"`

A : `<application android:allowBackup="true" android:debuggable="true"`

- Reempaquetar el binario, firmarlo e instalarlo



Generamos un backup de la app

- Usamos la app DIVA como ejemplo
 - Previamente la hemos desensamblado y hemos modificado su manifiesto

```
c:\Users\jucar\Downloads\vault\backup>adb backup jakhar.aseem.diva
WARNING: adb backup is deprecated and may be removed in a future release
Now unlock your device and confirm the backup operation...

c:\Users\jucar\Downloads\vault\backup>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\backup

09/03/2021 14:36    <DIR>          .
09/03/2021 14:36    <DIR>          ..
09/03/2021 12:10      5.209.376 abe.jar
09/03/2021 14:36      2.890.101 backup.ab
              2 archivos       8.099.477 bytes
              2 dirs        53.398.831.104 bytes libres

c:\Users\jucar\Downloads\vault\backup>>>java -jar abe.jar unpack backup.ab backup.tar 1234
Calculated MK checksum (use UTF-8: true): 465073911DE7CE15B60BE97EF5A9B93F288323C1EE30DAF28107B6EA4617A99
0% 1% 2% 3% 4% 5% 6% 7% 8% 9% 10% 11% 12% 13% 14% 15% 16% 17% 18% 19% 20% 21% 22% 23% 24% 25% 26% 27% 28% 29% 30% 31% 32%
% 33% 34% 35% 36% 37% 38% 39% 40% 41% 42% 43% 44% 45% 46% 47% 48% 49% 50% 51% 52% 53% 54% 55% 56% 57% 58% 59% 60% 61% 62%
% 63% 64% 65% 66% 67% 68% 69% 70% 71% 72% 73% 74% 75% 76% 77% 78% 79% 80% 81% 82% 83% 84% 85% 86% 87% 88% 89% 90% 91% 92%
% 93% 94% 95% 96% 97% 98% 99% 100%
8272384 bytes written to backup.tar.

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

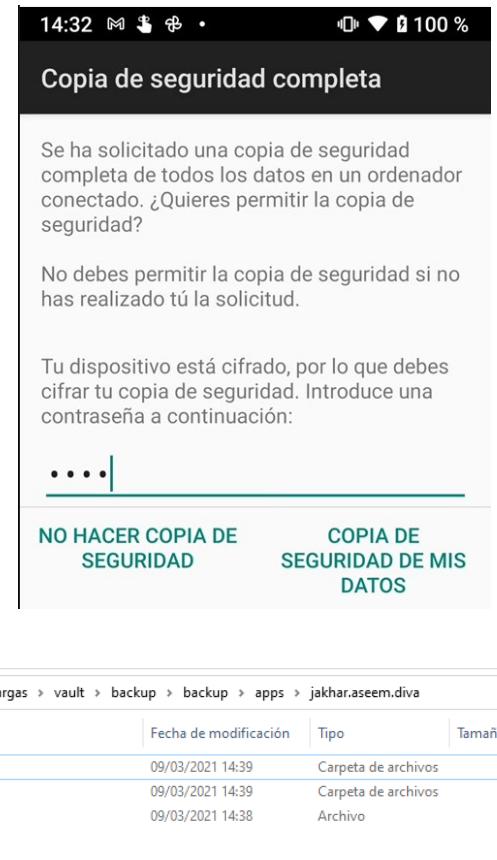
Directorio de c:\Users\jucar\Downloads\vault\backup

09/03/2021 14:38    <DIR>          .
09/03/2021 14:38    <DIR>          ..
09/03/2021 12:10      5.209.376 abe.jar
09/03/2021 14:36      2.890.101 backup.ab
09/03/2021 14:38      8.272.384 backup.tar
              3 archivos       16.371.861 bytes
              2 dirs        53.390.487.552 bytes libre
```

- Usamos la librería abe (Android Backup extractor), disponible en:

<https://github.com/nelenkov/android-backup-extractor>

- Referencia interesante: <https://infosecwriteups.com/extract-an-android-backup-file-96172efd4d86>





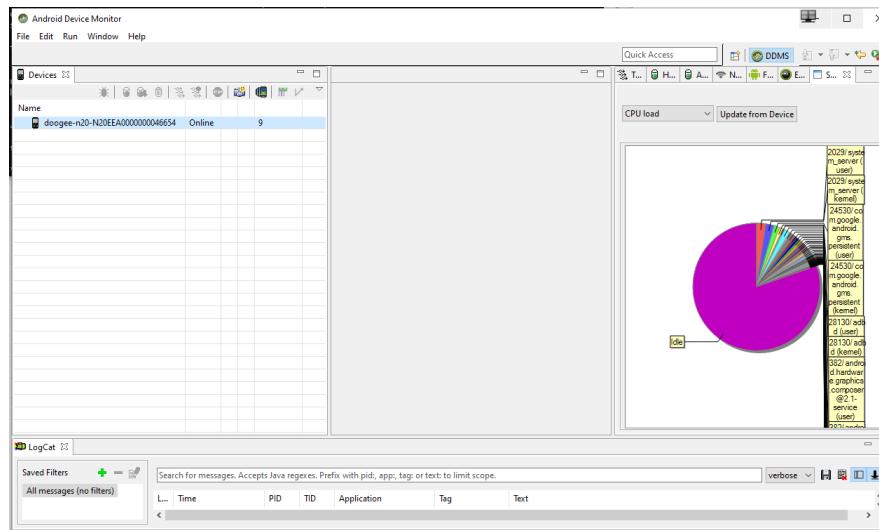
Indice

- Introducción al análisis dinámico de aplicaciones móviles
- **Supervisión de la ejecución de un apk utilizando un depurador**
- Control de la ejecución de un apk con Frida
- Casos de uso



Depurabilidad de apps Android

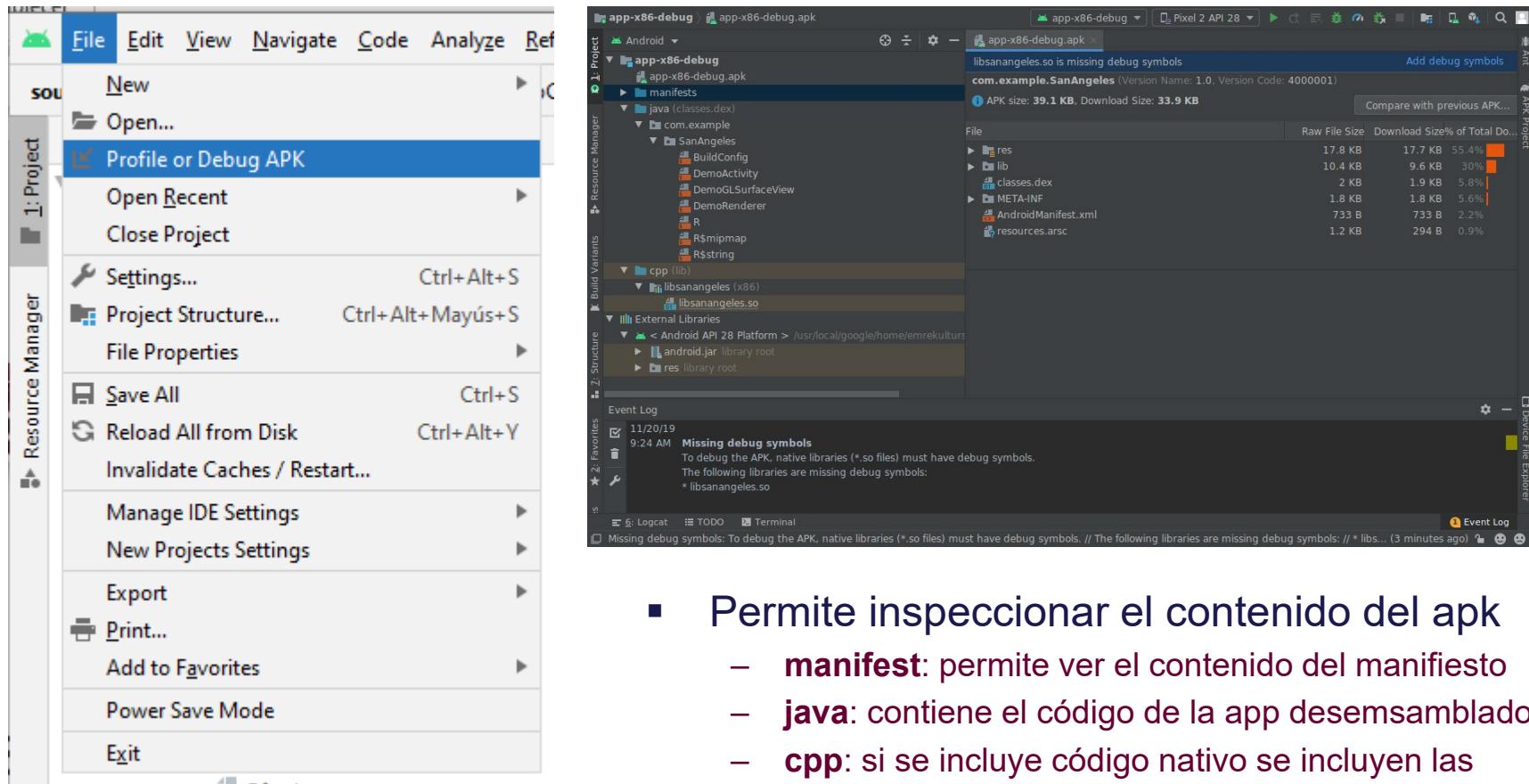
■ Uso del *Android Device Monitor (ADM)*



- Disponible en el directorio *Sdk > tools > monitor.bat*
- Característica obsoleta desde Android 3.1
 - <https://developer.android.com/studio/profile/monitor>



Depuración de apks con Android Studio



- Permite inspeccionar el contenido del apk
 - **manifest**: permite ver el contenido del manifiesto
 - **java**: contiene el código de la app desensamblado
 - **cpp**: si se incluye código nativo se incluyen las librerías incluidas en el APK (ficheros .so)
 - **External Libraries**: contiene la SDK de Android

<https://developer.android.com/studio/debug/apk-debugger>



Asociar código Kotlin/Java

Disassembled classes.dex file. To set up breakpoints for debugging, please attach Kotlin/Java source files.

[Attach Kotlin/Java Sources...](#)

- Si utilizamos un decompilador, como jadx o dex2jar+jd-gui, podremos obtener un código fuente aproximado par las clases del proyecto y asociárselo
 - No es necesario hacerlo clase por clase, basta con proporcionar el raíz del proyecto con el código
 - Podemos usar jadx y luego pasarle el directorio con los fuentes
 - El entorno sustituye los smali existentes por los java que encuentra

vault.apk-decompiled

Inicio Compartir Vista

Este equipo > Descargas > vault.apk-decompiled

Nombre	Fecha de modificación	Tipo	Tamaño
build	19/02/2021 10:39	Carpeta de archivos	
dist	19/02/2021 12:49	Carpeta de archivos	
original	19/02/2021 10:38	Carpeta de archivos	
res	19/02/2021 10:38	Carpeta de archivos	
smali	19/02/2021 10:38	Carpeta de archivos	
sources	19/02/2021 10:38	Carpeta de archivos	
AndroidManifest	19/02/2021 10:38	Documento XML	2 KB
apktool.yml	19/02/2021 10:38	Archivo YML	3 KB



vault.apk>MainActivity.java

```
package digital.basto.vault.ui.view;

import ...

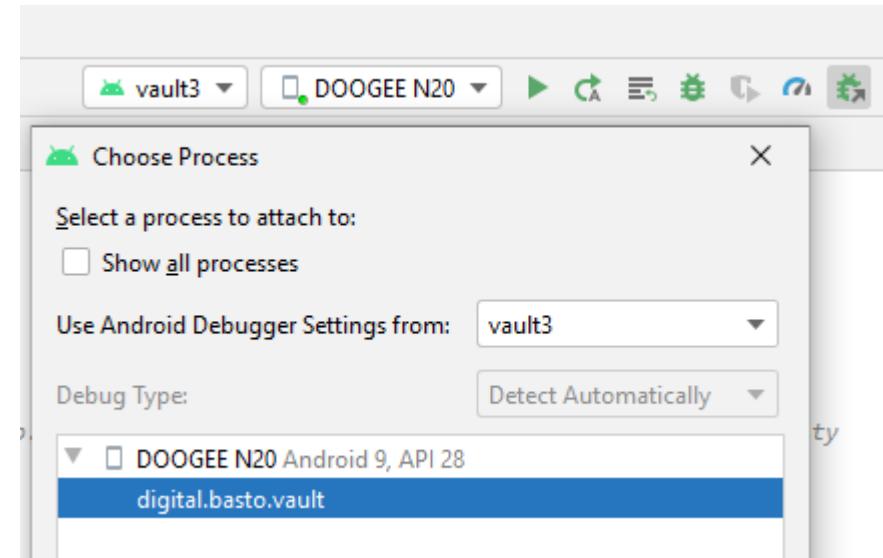
public class MainActivity extends AppCompatActivity {
    /* access modifiers changed from: protected */
    @Override // android.core.app.ComponentActivity
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        setSupportActionBar(findViewById(R.id.toolbar));
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        TextView balanceLabel = (TextView) findViewById(R.id.balanceText);
        Bundle extras = getIntent();
        if (extras != null) {
            balanceLabel.setText(extras.getString("vaultBalanceString"));
        }
    }
}
```

- Esto nos permite depurar la app “casi” tal y como lo haríamos con un proyecto convencional



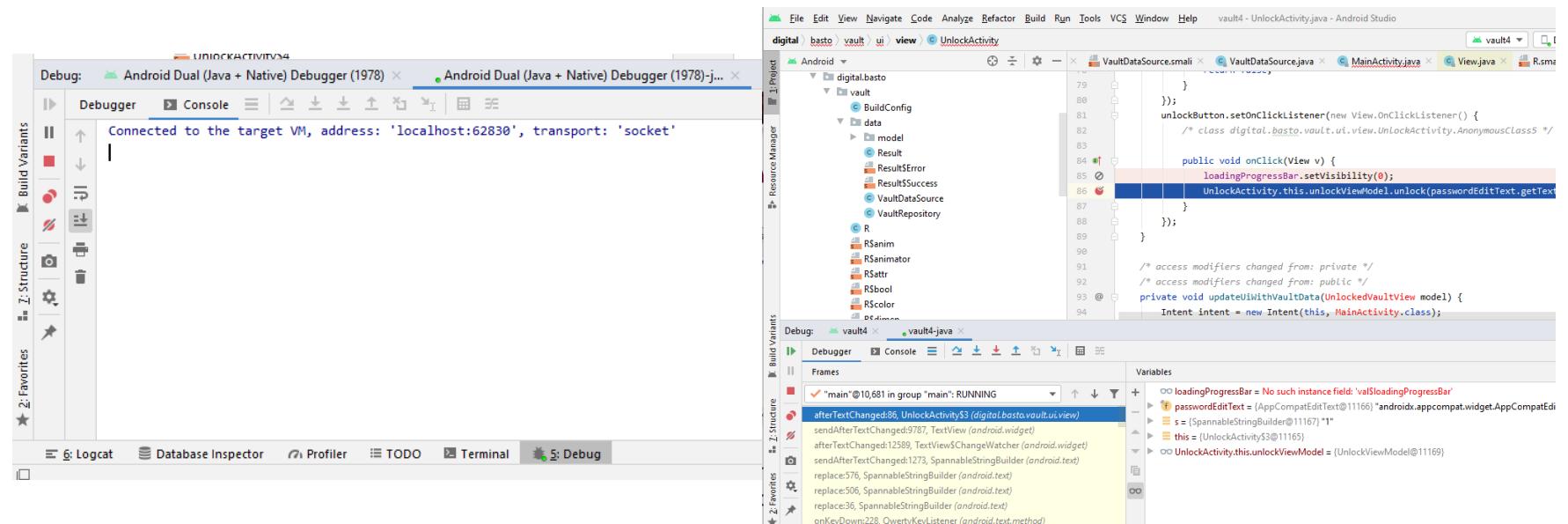
Depuración con Android Studio

- Mucho información disponible
Fuente oficial: <https://developer.android.com/studio/debug>
- Si en lugar de un emulador depuramos en un dispositivo real debemos asegurarnos de que el dispositivo esté en modo depuración
- Tras inspeccionar el código fuente (smali o java) y posicionar los breakpoints que deseemos iniciamos la depuración utilizando la herramienta de depuración 
 - Si la app ya está en ejecución, en lugar de detenerla para iniciar la depuración podemos asociar el depurador a una app que ya esté en ejecución
 - Obviamente para ello la app debe ser depurable:
 - flag android:debuggable="true" en su manifiesto





Ventana de depuración



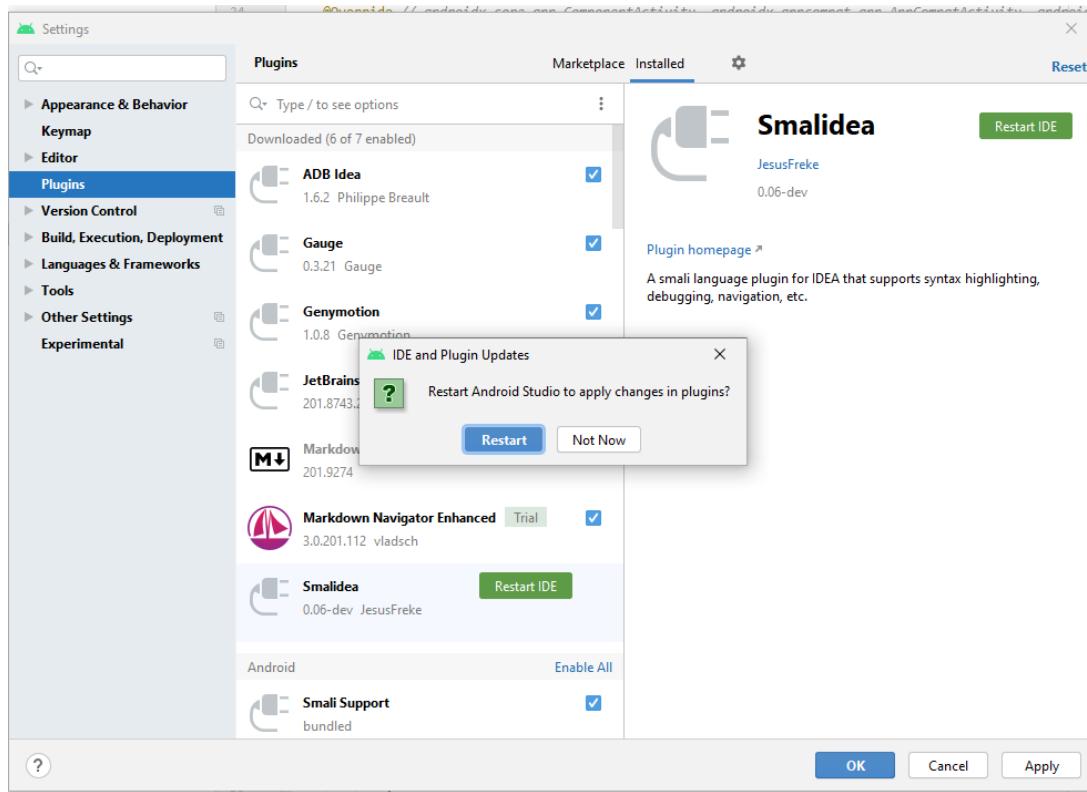
- Limitación: Es necesario importar el código Java para poder depurar, pero la depuración es inexacta
 - Relación entre código java y smali mal gestionada
- Solución:
 - Uso de smalidea para que la depuración directa sobre código smali sea posible
 - Utilizar los fuentes generados por jadx (o similar) como guía de alto nivel



Smalidea

■ Plugin de Android Studio

- Última versión disponible en: <https://bitbucket.org/JesusFreke/smalidea/downloads/>
- Proyecto Github disponible en: <https://github.com/JesusFreke/smalidea>



Activable/Desactivable
bajo demanda a través
de los settings del proyecto



Necesidad de reiniciar
Android Studio para que
los cambios se apliquen



Información de la ejecución de una app

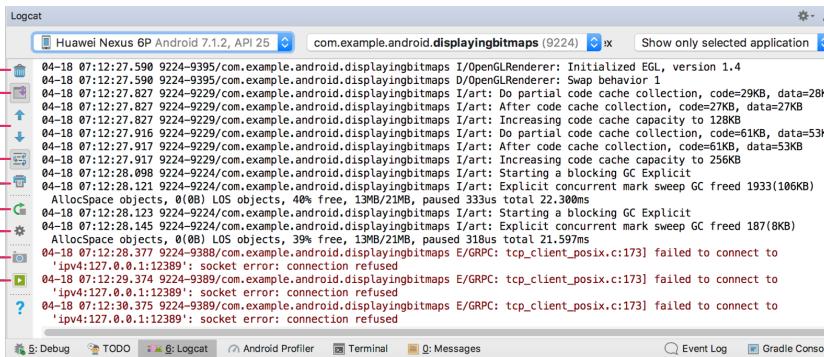
Logcat

Más información en:

<https://developer.android.com/studio/debug/am-logcat>

The Logcat toolbar provides the following buttons:

- 1 **Clear logcat**: Click to clear the visible log.
- 2 **Scroll to the end**: Click to jump to the bottom of the log and see the latest log messages. If you then click a line in the log, the view pauses scrolling at that point.
- 3 **Up the stack trace** **Down the stack trace**: Click to navigate up and down the stack traces in the log, selecting the subsequent filenames (and viewing the corresponding line numbers in the editor) that appear in the printed exceptions. This is the same behavior as when you click on a filename in the log.
- 4 **Use soft wraps**: Click to enable line wrapping and prevent horizontal scrolling (though any unbreakable strings will still require horizontal scrolling).
- 5 **Print**: Click to print the logcat messages. After selecting your print preferences in the dialog that appears, you can also choose to save to a PDF.
- 6 **Restart**: Click to clear the log and restart logcat. Unlike the **Clear logcat** button, this recovers and displays previous log messages, so is most useful if Logcat becomes unresponsive and you don't want to lose your log messages.
- 7 **Logcat header**: Click to open the **Configure Logcat Header** dialog, where you can customize the appearance of each logcat message, such as whether to show the date and time.
- 8 **Screen capture**: Click to [capture a screenshot](#).
- 9 **Screen record**: Click to [record a video](#) of the device (for a maximum of 3 minutes).





Uso de Breakpoints

Punto de ejecución actual

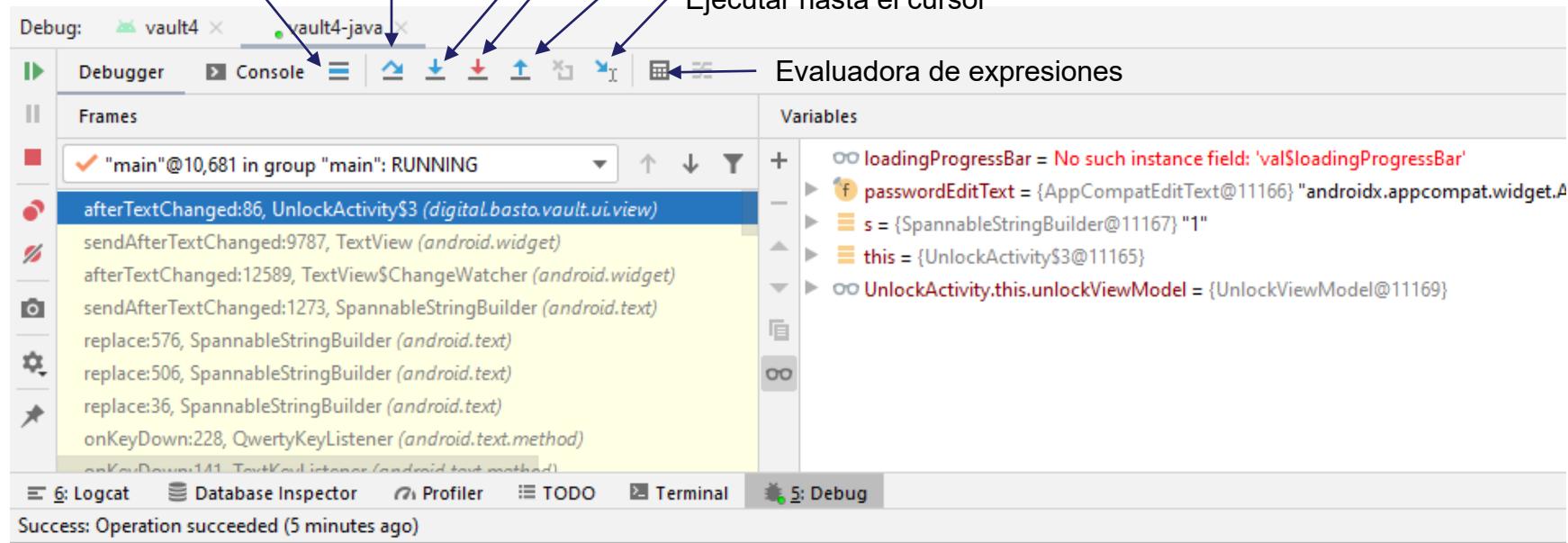
realizar llamada
(con inspección)

realizar llamada
(sin inspección)

Forzar inspección

Salir de llamada

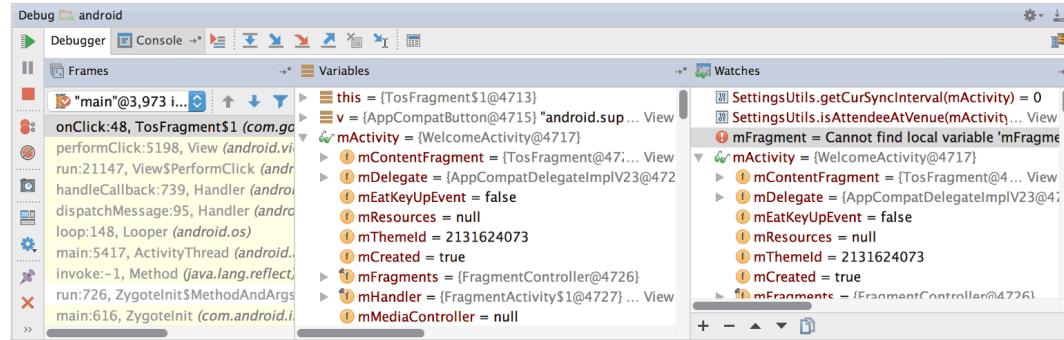
Ejecutar hasta el cursor



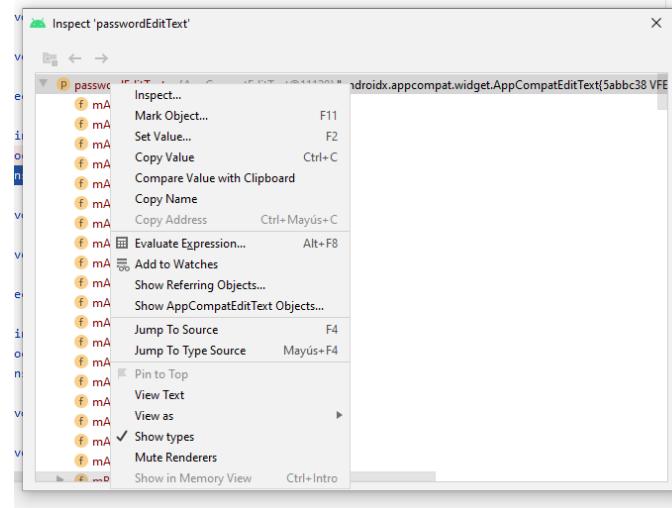


Inspección de variables

■ Monitorización de variables (Ventana Wathches)



– Añadir/eliminar puntos de observación (watchpoints)





Ejemplo: vault.apk

The screenshot shows the Android Studio interface during a debug session of the `vault4` project. The top navigation bar includes File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, VCS, Window, Help, and the current file `vault4 - VaultDataSource.smali`. The toolbar has icons for running, stopping, and navigating.

The Project tool window on the left lists Java classes: `Result$Success`, `VaultDataSource` (selected), `VaultRepository`, `R`, `R$anim`, and `R$animator`.

The Disassembled classes.dex file view shows assembly code:

```
Disassembled classes.dex file. To set up breakpoints for debugging, please attach Kotlin/Java source files.  
    invoke-virtual {v0, p1}, Ljava/lang/String;::equals(Ljava/lang/Object;)Z  
    move-result v0
```

The Debug tool window shows the current stack trace:

- main@10,681 in group "main": RUNNING
- unlock:15, VaultDataSource (digital.basto.vault.data)
- unlock:40, VaultRepository (digital.basto.vault.data)
- unlock:33, UnlockViewModel (digital.basto.vault.ui.view)
- onClick:103, UnlockActivity\$5 (digital.basto.vault.ui.view)
- performClick:6603, View (android.view)
- performClickInternal:6576, View (android.view)
- access\$3100:780, View (android.view)
- run:26090, View\$PerformClick (android.view)
- handleCallback:873, Handler (android.os)
- dispatchMessage:99, Handler (android.os)
- loop:193, Looper (android.os)
- main:6711, ActivityThread (android.app)
- invoke:-1, Method (java.lang.reflect)
- run:493, RuntimeInit\$MethodAndArgsCaller (com.android.internal.os)
- main:911, ZygoteInit (com.android.internal.os)

The Variables tool window shows local variables for the selected frame:

- `password = "1234"`
- `this = (VaultDataSource@11389)`
- `shadow$_klass_ = {Class@11063} "class digital.basto.vault.data.VaultDataSource" ... Navigate`
- `shadow$_monitor_ = 0`
- `vaultCombination = "Subscribe!"`



Depuración de apks con Android Studio + Smalidea

- Muy interesante para realizar una traza de ejecución de una app partiendo de su apk
 - + Permite inspeccionar las variables de ejecución de la app para recuperar información y monitorizar su ejecución
 - + Facilita la detección de los puntos de ejecución críticos y/o interesantes
 - No permite la modificación de los valores de las variables monitorizadas
 - Difícil de automatizar



¿Cómo afrontar las siguientes situaciones con lo que sabemos?

- Certificate pinning: Consiste en conseguir las funciones que verifican un determinado certificado en una conexión segura lo hagan aunque el certificado no sea de confianza
- Root check bypass: evitar que una app pueda determinar si su proceso se ejecuta con permisos de root
- PIN bypass: evitar el PIN de una app en un método de autenticación

Solución

- Podemos analizar el código e inferir lo que podría ocurrir
- Podemos ejecutar el código y monitorizarlo para comprender su comportamiento mejor y verificar lo inferido durante el análisis estático
- Podemos modificar su código smali, los recursos de su apk o su manifiesto de acuerdo a lo que deseemos
 - Reempaquetamos
 - Firmamos e instalamos
- ¿podemos modificar el comportamiento de una app dinámicamente, es decir, mientras ésta se ejecuta?



Indice

- Introducción al análisis dinámico de aplicaciones móviles
- Supervisión de la ejecución de un apk utilizando un depurador
- **Control de la ejecución de un apk con Frida**
- Casos de uso

Frida

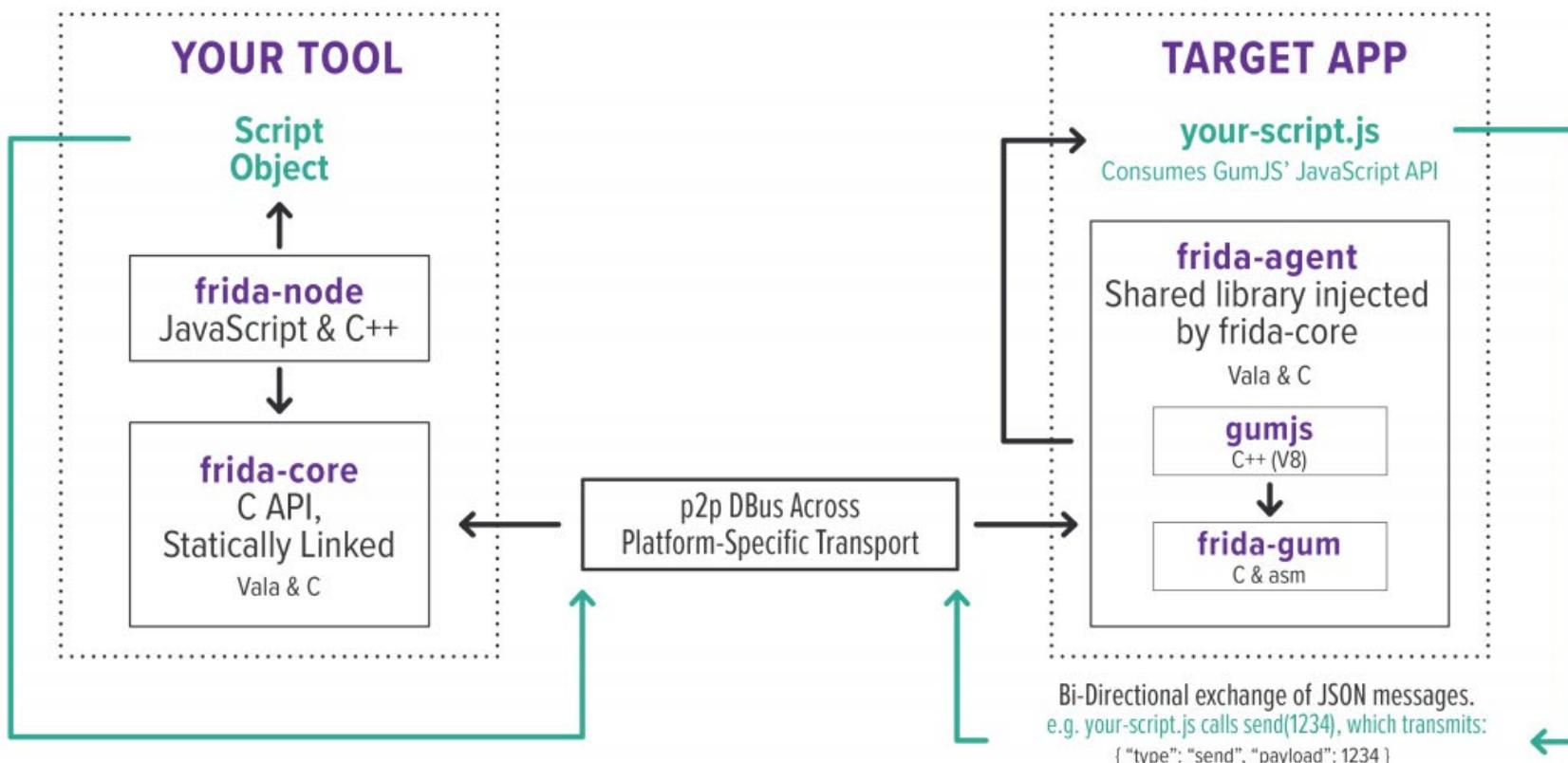
- Frida es un toolkit de la instrumentación dinámica de código
 - Permite injectar scripts en procesos que son vistos como cajas negras (su código no se necesita)
 - Es gratuito y funciona en Windows, macOS, GNU/Linux, iOS, Android, y QNX.
- Las últimas noticias relacionadas con Frida pueden encontrarse en <https://frida.re/news>
- La última versión de la herramienta está disponible para descarga en <https://github.com/frida/frida/releases>

¿En qué consiste la instrumentación dinámica de código?

- La idea consiste en insertar código en un binario durante su ejecución con el objetivo de entender mejor su comportamiento
- Esto permite (entre otras cosas) ...
 - trazar las APIs que utiliza una app
 - construir una herramienta que evalúe sus prestaciones
 - manipular las llamadas a funciones (*function hooking*)
- Principales características
 - Programable: es posible inyectar scripts propios en programas que son manipulados como cajas negras (su código no está disponible)
 - Inmediatez: editar, guardar y ver los resultados al instante, sin compilaciones o reinicios
 - Portable: puede inyectarse en procesos de ejecución de Windows, MacOS, GNU/Linux, iOS, Android, y QNX. Permite además la interacción con Node.js, Python, Swift, .Net, Qt/Qml y C
 - Gratuita: ha sido desarrollada como software libre para su propia retroalimentación
 - Confiable: tiene el respaldo de importantes empresas, y una suite de pruebas completa



Arquitectura de la herramienta





Prerrequisitos

- Prerrequisitos
 - Un emulador (AVD o Genymotion) o dispositivos con privilegios de administrador (rooted o jailbroken), aunque FRIDA puede funcionar también (de manera más limitada) sobre dispositivos sin dichos privilegios
 - Necesita estos permisos para poder manipular el ptrace del dispositivo bajo estudio e introducir sobre el mismo el gadget
 - Si no los tienen también funcionará, veremos cómo más adelante
 - Adb
 - Última versión de frida-server para Android: <https://github.com/frida/frida/releases>
 - Python 3.x (altamente recomendado)



Modos de funcionamiento

- Inyección de código (injected)
 - Uso de frida-server, frida-core + gumjs
- Empotrar el código en la app (embedded)
 - Uso de frida-gadget
- Precargado de código (preloaded)
 - Uso de frida-gadget, código a ejecutar cargado desde una librería compartida
- Más información en <https://frida.re/docs/modes>



Instalación

- Es muy sencillo
 - Proceso descrito en <https://frida.re/docs/installation>
 - Aunque se aconseja utilizar Python 3.x también funciona sobre Python 2.7 o superior
- Instalación en el dispositivo/emulador con el que trabajemos (modo de funcionamiento *injected*)
 - Proceso descrito en <https://frida.re/docs/android>
 - Descargar frida-server (<https://github.com/frida/frida/releases>)
 - Dispositivo Android real: [frida-server-14.2.13-android-arm/arm64.xz](#)
 - Emulador Android: [frida-server-14.2.13-android-x86.xz](#)

```
c:\Users\jucar\Downloads\frida-server>adb push frida-server /data/local/tmp
frida-server: 1 file pushed, 0 skipped. 56.5 MB/s (42925716 bytes in 0.724s)

c:\Users\jucar\Downloads\frida-server>adb shell "chmod 755 /data/local/tmp/frida-server"

c:\Users\jucar\Downloads\frida-server>adb shell
root@vbox86p:/ # cd /data/local/tmp
local/      lost+found/
root@vbox86p:/ # cd /data/local/tmp/
root@vbox86p:/data/local/tmp # ./frida-server
```



Encontrar dispositivos y apps

```
1 #To list the available devices for frida
2 frida-ls-devices
3
4 # Connect Frida to an iPad over USB and list running processes
5 $ frida-ps -U
6
7 # List running applications
8 $ frida-ps -Ua
9
10 # List installed applications
11 $ frida-ps -Uai
12
13 # Connect Frida to the specific device
14 $ frida-ps -D 0216027d1d6d3a03
```



Lanzamiento de Frida

```
1 #Hooking before starting the app
2 frida -U --no-pause -l hookNative.js -f com.erev0s.jniapp
3
4 #Basic frida hooking
5 frida -U com.erev0s.jniapp -l hookNative.js
```

- Opciones
 - U: verificar la conexión por USB con los dispositivos
 - l: script a ejecutar
 - f: paquete con el que se trabaja

Frida Code Editor

```
1 Java.perform(function() {
2   // Código JavaScript Aquí
3 });
4
```



Indice

- Introducción al análisis dinámico de aplicaciones móviles
- Supervisión de la ejecución de un apk utilizando un depurador
- Control de la ejecución de un apk con Frida
- **Casos de uso**



Ejemplos sencillos

- Comenzaremos con la app AndroidLab.apk
 - <https://ironhackers.es/AndroidLab.apk>

The screenshot shows a mobile application interface. On the left, there's a vertical navigation menu with items: Level 1- Call me!, Level 2- AlwaysTrue, Level 3- CreateMe, and Level 4- Sniff. Below the menu, the text "Free for personal use" is visible. In the center, there's a code editor window titled "AndroidManifest.xml" displaying the following XML code:

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    android:compileSdkVersion="29"
    android:compileSdkVersionCodename="10"
    package="com.ironhackers.androidlab"
    platformBuildVersionCode="29"
    platformBuildVersionName="1.0">
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <application android:allowBackup="true"
        android:appComponentFactory="androidx.core.app.CoreComponentFactory"
        android:debuggable="true"
        android:icon="@drawable/ironpollo"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:label="@string/title_activity_sniff" android:name="com.ironhackers.androidlab.Sniff" android:theme="@style/AppTheme.NoActionBar"/>
        <activity android:label="@string/title_activity_createme" android:name="com.ironhackers.androidlab.CreateMe" android:theme="@style/AppTheme.NoActionBar"/>
        <activity android:label="@string/title_activity_alwaystrue" android:name="com.ironhackers.androidlab.AlwaysTrue" android:theme="@style/AppTheme.NoActionBar"/>
        <activity android:label="@string/title_activity_callme" android:name="com.ironhackers.androidlab.CallMe" android:theme="@style/AppTheme.NoActionBar"/>
        <activity android:label="@string/app_name" android:name="com.ironhackers.androidlab.MainActivity" android:theme="@style/AppTheme.NoActionBar">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
    </application>
</manifest>
```

On the right side of the screen, there's a dock with three icons: Amaze, AndroidLab, and API Demos.



AndroidLab: Level 1

Callme

Nothing to show

Free for personal use

The screenshot shows an IDE interface with two tabs: 'Callme.java' and 'AndroLab-Level1.js'. The Java code in 'Callme.java' defines a class 'Callme' that extends 'AppCompatActivity'. It contains an onCreate method and a private void 'call_me_win' method. The JavaScript code in 'AndroLab-Level1.js' uses Java.perform to inject code into the 'Callme' activity. It logs 'onCreate' execution, calls 'call_me_win', and then logs 'call_me_win' execution.

```
1 package com.ironhackers.androidlab;
2
3 import android.os.Bundle;
4 import android.widget.TextView;
5 import androidx.appcompat.app.AppCompatActivity;
6 import androidx.appcompat.widget.Toolbar;
7
8 public class Callme extends AppCompatActivity {
9     private TextView msg;
10
11     /* access modifiers changed from: protected */
12     @Override // androidx.activity.ComponentActivity, androidx.core.ap
13     public void onCreate(Bundle savedInstanceState) {
14         super.onCreate(savedInstanceState);
15         setContentView(R.layout.activity_callme);
16         setSupportActionBar(findViewById(R.id.toolbar));
17         this.msg = (TextView) findViewById(R.id.flag);
18     }
19
20     private void call_me_win() {
21         this.msg.setText(R.string.flag1);
22     }
23 }
24 
```

```
1
2
3 Java.perform(function () {
4
5     var callmeActivity = Java.use("com.ironhackers.androidlab.Callme");
6
7     callmeActivity.onCreate.implementation = function(bundle) {
8         console.log('');
9         console.log("> Ejecutando onCreate(...)");
10        this.onCreate(bundle);
11        console.log("> Invocando call:me_win()");
12        this.call_me_win();
13    };
14
15 });
16 
```

Callme

c4ll_m3_fl4g

Free for personal use

AndroidLab: Level 1

■ Alternativa

```
1  Java.perform(function () {  
2      Java.choose('com.ironhackers.androidlab.Callme', {  
3          onMatch: function(instance) {  
4              Java.scheduleOnMainThread(function () {  
5                  instance.call_me_win();  
6              });  
7          },  
8          onComplete: function() {}  
9      });  
10  });
```



AndroidLab: Level 2

```
Alwaytrue.java
```

```
1 package com.ironhackers.androidlab;
2
3 import android.os.Bundle;
4 import android.widget.TextView;
5 import androidx.appcompat.app.AppCompatActivity;
6 import androidx.appcompat.widget.Toolbar;
7
8 public class Alwaytrue extends AppCompatActivity {
9     /* access modifiers changed from: protected */
10    @Override // androidx.activity.ComponentActivity, androidx.core.app.Comp
11    public void onCreate(Bundle savedInstanceState) {
12        super.onCreate(savedInstanceState);
13        setContentView(R.layout.activity_alwaytrue);
14        setSupportActionBar(findViewById(R.id.toolbar));
15        if (impossible_check()) {
16            ((TextView) findViewById(R.id.msg)).setText(R.string.flag2);
17        }
18    }
19
20    private boolean impossible_check() {
21        return false;
22    }
23}
24
```

```
Java.perform(function () {
    var atClass = Java.use("com.ironhackers.androidlab.Alwaytrue");

    atClass.impossible_check.implementation = function() {
        console.log('');
        console.log("> Ejecutando impossible_check()");
        return true;
    };
});
```

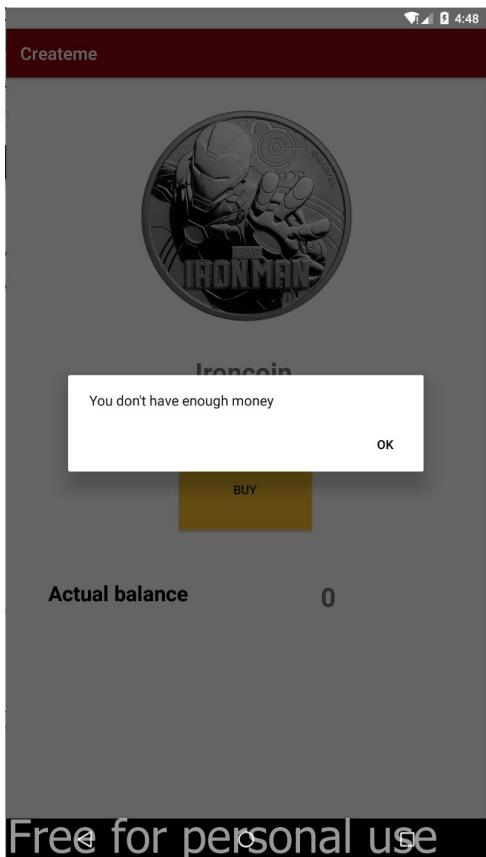
Alwaytrue

4lw4ystru3_f14g

Free for personal use



AndroidLab: Level 3



```

1 package com.ironhackers.androidlab;
2
3 import android.app.AlertDialog;
4 import android.content.DialogInterface;
5 import android.os.Bundle;
6 import android.view.View;
7 import android.widget.Button;
8 import android.widget.TextView;
9 import androidx.appcompat.app.AppCompatActivity;
10 import androidx.appcompat.widget.Toolbar;
11
12 public class Createme extends AppCompatActivity {
13     private Button button;
14     private Person person;
15
16     /* access modifiers changed from: protected */
17     @Override // androidx.activity.ComponentActivity, androidx.core.app.ComponentActivity
18     public void onCreate(Bundle savedInstanceState) {
19         super.onCreate(savedInstanceState);
20         setContentView(R.layout.activity_createme);
21         setSupportActionBar((Toolbar) findViewById(R.id.toolbar));
22         this.person = new Person("Iron", 35, null);
23         TextView balance = (TextView) findViewById(R.id.balance);
24         this.button = (Button) findViewById(R.id.button);
25         this.button.setOnClickListener(new View.OnClickListener() {
26             /* class com.ironhackers.androidlab.Createme$AnonymousClass1 */
27
28             public void onClick(View view) {
29                 if (Createme.this.person.getWallet() == null || Createme.this.person.getWallet().getMoney() < 100) {
30                     AlertDialog.Builder builder1 = new AlertDialog.Builder(Createme.this);
31                     builder1.setMessage("You don't have enough money");
32                     builder1.setCancelable(true);
33                     builder1.setPositiveButton("OK", new DialogInterface.OnClickListener() {
34                         /* class com.ironhackers.androidlab.Createme$AnonymousClass1$AnonymousClass1 */
35
36                         public void onClick(DialogInterface dialog, int id) {
37                             dialog.cancel();
38                         }
39                     });
39                     builder1.create().show();
40                     return;
41                 }
42                 AlertDialog.Builder builder12 = new AlertDialog.Builder(Createme.this);
43                 builder12.setMessage("Congrats, the flag is " + Createme.this.getString(R.string.flag3));
44                 builder12.setCancelable(true);
45                 builder12.setPositiveButton("OK", new DialogInterface.OnClickListener() {
46                     /* class com.ironhackers.androidlab.Createme$AnonymousClass1$AnonymousClass2 */
47
48                     public void onClick(DialogInterface dialog, int id) {
49                         dialog.cancel();
50                     }
51                 });
52                 builder12.create().show();
53             }
54         });
55     }
56     if (this.person.getWallet() != null) {
57         balance.setText(this.person.getWallet().getMoney() + BuildConfig.FLAVOR);
58         return;
59     }
59     balance.setText("0");
60 }
61
62 }

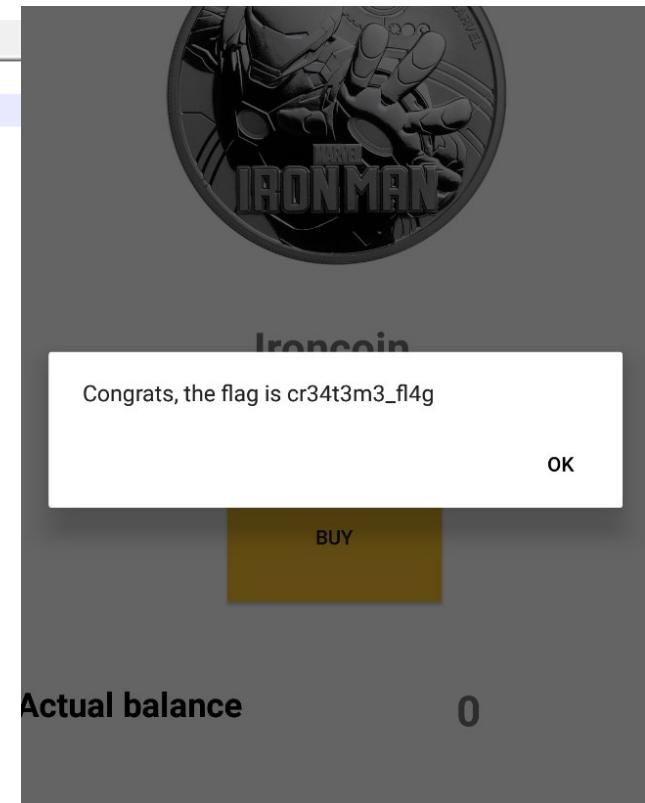
```



AndroidLab: Level 3

- Estrategia → Crear una cartera con suficiente dinero como para crear una alerta con el flag buscado

```
1
2 Java.perform(function() {
3     var wallet=Java.use("com.ironhackers.androidlab.Wallet");
4     Java.choose('com.ironhackers.androidlab.Createme', {
5         onMatch: function(instance) {
6             instance.person.value.setWallet(wallet.$new(100));
7         },
8         onComplete: function() {
9     }
10 });
11 })
```





AndroidLab: Level 3

- ¿Y cómo actualizar la interfaz de la app?

```
uncrackable1.js uncrackable2.js AndroLab-Level1.js AndroidLab-Level2.js AndroidLab-Level3.js AndroidLab-Level3b.js
1
2 Java.perform(function() {
3     var wallet=Java.use("com.ironhackers.androidlab.Wallet");
4     Java.choose('com.ironhackers.androidlab.Createme', {
5         onMatch: function(instance) {
6             instance.person.value.setWallet(wallet.$new(100));
7
8             var textview = Java.use('android.widget.TextView');
9             var balObj = instance.findViewById(2131230785); //0x7f080041
10            var balTV = Java.cast(balObj, textview);
11            var string = Java.use('java.lang.String');
12            balTV.setText(string.$new('100'));
13
14        },
15        onComplete: function() {
16        }
17    });
18 });
19 });

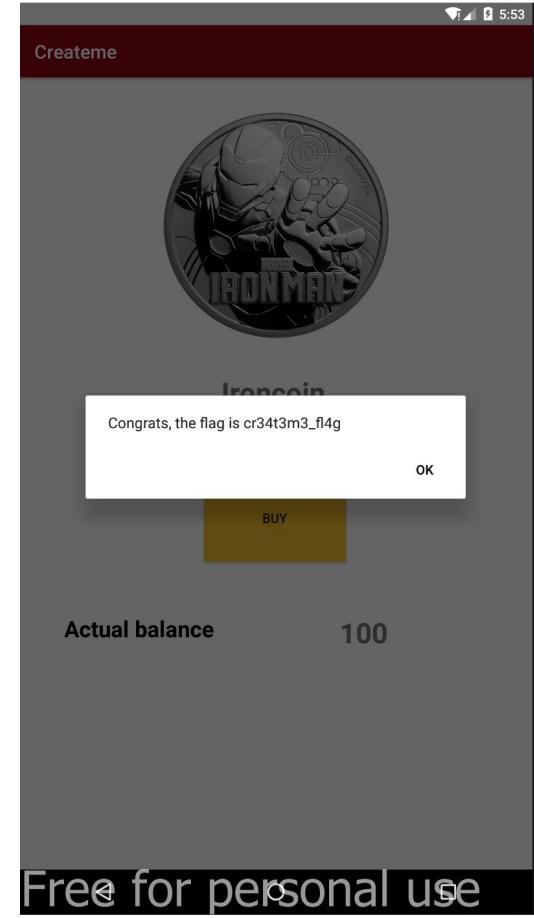
c:\Users\jucar\Downloads\frida-server\scripts>frida -U -l AndroidLab-Level3.js com.ironhackers.androidlab
Frida 14.2.13 - A world-class dynamic instrumentation toolkit
Commands:
  help      -> Displays the help system
  object?   -> Display information about 'object'
  exit/quit -> Exit
  More info at https://www.frida.re/docs/home/
Error: android.view.ViewRootImpl$CalledFromWrongThreadException: Only the original thread that created a view hierarchy can touch its views.
        at <anonymous> (frida/node_modules/frida-java-bridge/lib/env.js:124)
        at value (frida/node_modules/frida-java-bridge/lib/class-factory.js:1058)
        at e (frida/node_modules/frida-java-bridge/lib/class-factory.js:580)
        at apply (native)
        at value (frida/node_modules/frida-java-bridge/lib/class-factory.js:963)
        at e (frida/node_modules/frida-java-bridge/lib/class-factory.js:547)
        at onMatch (/AndroidLab-Level3.js:12)
        at _chooseObjectsArtLegacy (frida/node_modules/frida-java-bridge/lib/class-factory.js:317)
        at <anonymous> (frida/node_modules/frida-java-bridge/lib/class-factory.js:245)
        at it (frida/node_modules/frida-java-bridge/lib/android.js:545)
[Google Nexus 5:<com.ironhackers.androidlab>]
```



AndroidLab: Level 3

- Actualizar la interfaz en el hilo principal de la app

```
uncrackable1.js uncrackable2.js AndroLab-Level1.js AndroidLab-Level2.js AndroidLab-Level3.js AndroidLab-Level3b.js
1
2 Java.perform(function(){
3     var wallet=Java.use("com.ironhackers.androidlab.Wallet");
4     Java.choose('com.ironhackers.androidlab.Createme', {
5         onMatch: function(instance) {
6             Java.scheduleOnMainThread(function () {
7                 instance.person.value.setWallet(wallet.$new(100));
8
9                 var textview = Java.use('android.widget.TextView');
10                var balObj = instance.findViewById(2131230785); //0x7f080041
11                var balTV = Java.cast(balObj, textview);
12                var string = Java.use('java.lang.String');
13                balTV.setText(string.$new('100'));
14            });
15        },
16        onComplete: function() {
17        }
18    });
19 });
20});
```



```
c:\Users\jucar\Downloads\frida-server\scripts>frida -U -l AndroidLab-Level3b.js com.ironhackers.androidlab
Frida 14.2.13 - 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/
[Google Nexus 5::com.ironhackers.androidlab]-> %reload
[Google Nexus 5::com.ironhackers.androidlab]->
```



AndroidLab: Level 4

Sniff

Enter the flag!

Nooo, you failed!

OK

CHECK

Free for personal use

```
| (C) | Frida 14.2.13 - A world-class dynamic instrument
| / \ | Commands:
| / \ |   help      -> Displays the help system
| . . . |   object?   -> Display information about 'object'
| . . . |   exit/quit -> Exit
| . . . |   More info at https://www.frida.re/docs/home/
Google Nexus 5::com.ironhackers.androidlab]-> %reload
Google Nexus 5::com.ironhackers.androidlab]-> exit
thank you for using Frida!
:Users\jucar\Downloads\frida-server\scripts>frida -U -l And
| (C) | Frida 14.2.13 - A world-class dynamic instrument
| / \ | Commands:
| / \ |   help      -> Displays the help system
| . . . |   object?   -> Display information about 'object'
| . . . |   exit/quit -> Exit
| . . . |   More info at https://www.frida.re/docs/home/
Google Nexus 5::com.ironhackers.androidlab]->
El flag buscado es: sniff_f14g
El flag buscado es: sniff_f14g
```

Enter the flag!

Correct, you got it!!

OK

CHECK

```
1
2
3 Java.perform(function() {
4
5     var sniffObj = Java.use("com.ironhackers.androidlab.Sniff");
6
7     sniffObj.generateFlag.implementation = function(str1, str2) {
8         console.log('');
9         console.log("> El flag buscado es: " + str2);
10        this.generateFlag(str2, str2);
11    };
12
13});
```



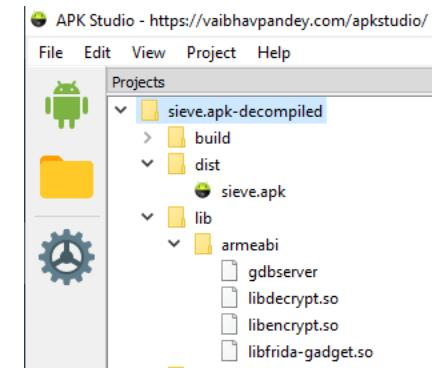
Trabajo con dispositivos no ruteados

- Usaremos la app Sieve a título de ejemplo
 - Recordar que esta app utiliza librerías compiladas para ARM, con lo que requiere del uso de un dispositivo real o un emulador corriendo un ABI ARM
- Si el dispositivo utilizado no está rooteado, el uso de Frida requiere de cierta preparación previa de la app bajo estudio
 1. Bajar la librería frida-gadget.so desde <https://github.com/frida/frida/releases>
 2. Desensamblar el apk de la app bajo estudio y copiar la librería en su directorio *lib*
 3. Incorporar la llamada en la app para utilizar la librería

```
System.loadLibrary("frida-gadget")
```

Al trabajar con código desensamblado hay que añadir el siguiente código al smali de la Actividad principal de la app → Hay que hacerlo en su constructor (al principio, tras el .locals)

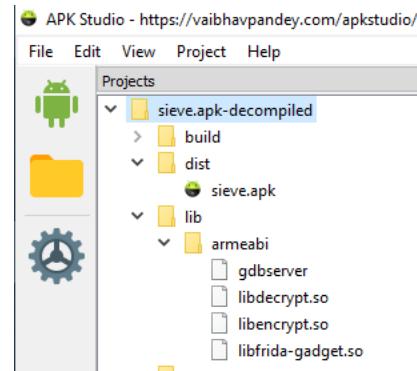
```
    ...
45  # direct methods
46  .method public constructor <init>()V
47      .locals 3
48
49      const-string v2, "frida-gadget"
50      invoke-static {v2}, Ljava/lang/System;->loadLibrary(Ljava/lang/String;)V
51
52      .prologue
53      const/4 v1, 0x0
```





Trabajo con dispositivos no ruteados

- Usaremos la app Sieve a título de ejemplo
 - Recordar que esta app utiliza librerías compiladas para ARM, con lo que requiere del uso de un dispositivo real o un emulador corriendo un ABI ARM
- Si el dispositivo utilizado no está rooteado, el uso de Frida requiere de cierta preparación previa de la app bajo estudio
 1. Bajar la librería frida-gadget.so desde <https://github.com/frida/frida/releases>
 2. Desensamblar el apk de la app bajo estudio y copiar la librería en su directorio *lib*





Trabajo con dispositivos no ruteados

- Tras haber incorporado el gadget de Frida al directorio lib de la app debemos incorporar la llamada en la app para utilizar la librería (recordar que trabajamos con código desensamblado)
 - Incorporar al constructor de la Actividad principal de la app la llamada a la librería
 - Hay que incluir la llamada justo después del .locals del constructor
 - El número de registros que .locals refleja se incrementa en 1 y se adapta el índice de la variable vX a añadir al valor resultante del incremento. P.ej: si tenemos .locals 3 → trabajaremos con v2

```
45 # direct methods
46 .method public constructor <init>()V
47     .locals 3
48
49     const-string v2, "frida-gadget"
50     invoke-static {v2}, Ljava/lang/System;->loadLibrary(Ljava/lang/String;)V
51
52     .prologue
53     const/4 v1, 0x0
```

System.loadLibrary("frida-gadget")

- Añadir al manifiesto permisos de INTERNET para la app

```
<uses-permission android:name="android.permission.INTERNET"/>
```

- Reensamblar, firmar, alinear e instalar el apk resultante



Trabajo con dispositivos no ruteados

- Lanzar a ejecución la app en el dispositivo
- Lanzar en nuestra máquina frida de la siguiente manera

```
C:\Users\jucar\Downloads\frida-server\scripts>frida -U gadget -l Sieve.js com.mwr.example.sieve
```

```
/ [ ] Frida 14.2.13 - 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/
[N20::gadget]->
```



Obtención de PIN por fuerza bruta

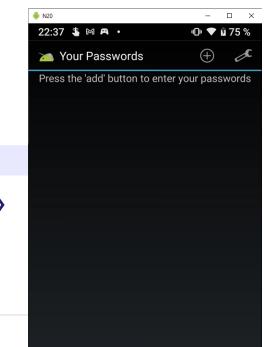
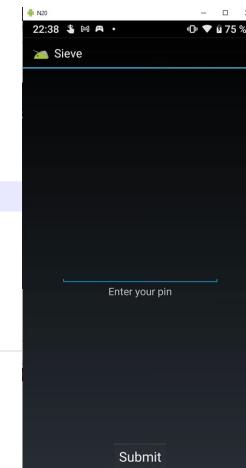
Símbolo del sistema - frida -U gadget -l Sieve.js com.mwr.example.sieve

```
/ Frida 14.2.13 - 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/
[N20::gadget]-> message: {'type': 'send', 'payload': '1000: '} data: None
message: {'type': 'send', 'payload': '1001: '} data: None
message: {'type': 'send', 'payload': '1002: '} data: None
message: {'type': 'send', 'payload': '1003: '} data: None
message: {'type': 'send', 'payload': '1004: '} data: None
message: {'type': 'send', 'payload': '1005: '} data: None
message: {'type': 'send', 'payload': '1006: '} data: None
message: {'type': 'send', 'payload': '1007: '} data: None
message: {'type': 'send', 'payload': '1008: '} data: None
message: {'type': 'send', 'payload': '1009: '} data: None
message: {'type': 'send', 'payload': '1010: '} data: None
    console.log('Done:');
};

var ShortLoginActivity = Java.use('com.mwr.example.sieve.ShortLoginActivity');
ShortLoginActivity.submit.implementation = function(v) {
    var service=this.serviceConnection.value
    for(var i=1000; i<1300; i++)
        {
            service.checkPin(i+"");
            send(i + ": ");
            Log.v("frida-sieve", "Probando con: "+i);
        }
    Log.v("frida-sieve", "submit");
};
```

Símbolo del sistema - adb logcat -s "frida-sieve:V"

```
c:\Program Files\ScreenCPY\scrcpy-win64-v1.17>adb logcat -s "frida-sieve:V"
----- beginning of system
----- beginning of crash
----- beginning of main
03-11 22:35:21.026 12867 12906 V frida-sieve: JS activo
03-11 22:35:24.024 12867 12867 V frida-sieve: Probando con: 1000
03-11 22:35:37.898 12867 12906 V frida-sieve: JS activo
03-11 22:35:38.712 12867 12906 V frida-sieve: JS activo
03-11 22:35:50.362 12867 12906 V frida-sieve: JS activo
03-11 22:36:08.969 12867 12867 V frida-sieve: Probando con: 1000
03-11 22:36:08.970 12867 12867 V frida-sieve: Probando con: 1001
03-11 22:36:08.971 12867 12867 V frida-sieve: Probando con: 1002
03-11 22:36:08.972 12867 12867 V frida-sieve: Probando con: 1003
03-11 22:36:08.973 12867 12867 V frida-sieve: Probando con: 1004
03-11 22:36:08.974 12867 12867 V frida-sieve: Probando con: 1005
03-11 22:36:08.974 12867 12867 V frida-sieve: Probando con: 1006
03-11 22:36:08.975 12867 12867 V frida-sieve: Probando con: 1007
03-11 22:36:08.976 12867 12867 V frida-sieve: Probando con: 1008
03-11 22:36:08.977 12867 12867 V frida-sieve: Probando con: 1009
03-11 22:36:08.979 12867 12867 V frida-sieve: Probando con: 1010
```





Casos de estudio más complejos

- OWASP MSTG Uncrackable Apps

- Disponibles en

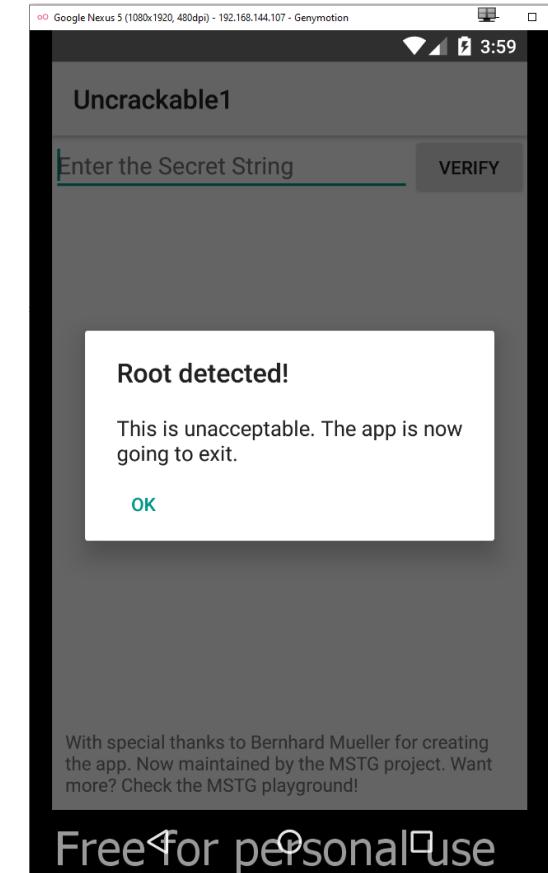
<https://github.com/OWASP/owasp-mstg>

- https://github.com/OWASP/owasp-mstg/blob/master/Crackmes/Android/Level_01
 - https://github.com/OWASP/owasp-mstg/blob/master/Crackmes/Android/Level_02



UncrackableApp1

- Objetivo: ¿Cuál es la clave secreta?
- Obstáculos
 - Detección root → Bypass root
 - Detección depuración → Bypass Debugging
- Estrategia
 - Análisis estático
 - Uso de un depurador
 - Uso de Frida





Root bypassing

```
Projects
  UnCrackable-Level1.apk-decompiled
    build
    dist
      UnCrackable-Level1.apk
    original
    res
    smali
    sources
      owasp
      sg
        vantagepoint
          a
            a.java
            b.java
            c.java
      uncrackable1
    AndroidManifest.xml
    apktool.yml

  AndroidManifest.xml
  a.java
  a.smali
  c.java

1 package sg.vantagepoint.a;
2
3 import android.os.Build;
4 import java.io.File;
5
6 public class a {
7     public static boolean a() {
8         for (String str : System.getenv("PATH").split(":")) {
9             if (new File(str, "su").exists()) {
10                 return true;
11             }
12         }
13         return false;
14     }
15
16     public static boolean b() {
17         String str = Build.TAGS;
18         return str != null && str.contains("test-keys");
19     }
20
21     public static boolean c() {
22         for (String str : new String[]{"system/app/Superuser.apk", "/system/"}){
23             if (new File(str).exists()){
24                 return true;
25             }
26         }
27         return false;
28     }
29 }
30

57 .method public static b()Z
58     .locals 2
59
60     sget-object v0, Landroid/os/Build;-->TAGS:Ljava/lang/String;
61
62     if-eqz v0, :cond_0
63
64     const-string v1, "test-keys"
65
66     invoke-virtual {v0, v1}, Ljava/lang/String;-->contains(Ljava/lang/CharSequence;
67
68     move-result v0
69
70     if-eqz v0, :cond_0
71
72     const/4 v0, 0x0
73
74     return v0
75
76     :cond_0
77     const/4 v0, 0x0
78
79     return v0
80 .end method
81

82 .method public static c()Z
83     .locals 7
84
85     const-string v0, "/system/app/Superuser.apk"
86
87     const-string v1, "/system/xbin/daemonsu"
88
89     const-string v2, "/system/etc/init.d/99SuperSUDaemon"
90
91     const-string v3, "/system/bin/.ext/.su"
92
93     const-string v4, "/system/etc/.has_su_daemon"
94
95     const-string v5, "/system/etc/.installed_su_daemon"
96
97     const-string v6, "/dev/com.koushikdutta.superuser.daemon/"
98
99     filled-new-array/range {v0 .. v6}, [Ljava/lang/String;
100
101     move-result-object v0
102
103     array-length v1, v0
104
105     const/4 v2, 0x0
106
107     const/4 v3, 0x0
108
109     :goto_0
110     if-ge v3, v1, :cond_1
111
112     aget-object v4, v0, v3
113
114     new-instance v5, Ljava/io/File;
115
116     invoke-direct {v5, v4}, Ljava/io/File;--><init>(Ljava/lang/String;)V
117
118     invoke-virtual {v5}, Ljava/io/File;-->exists()Z
119
120     move-result v4
121
122     if-eqz v4, :cond_0
123
124     const/4 v0, 0x1
125
126     return v2
127
128     :cond_0
129     add-int/lit8 v3, v3, 0x1
130
131     goto :goto_0
132
133     :cond_1
134     return v2
135 .end method
136
```

Análisis estático + Tampering

```
1 package sg.vantagepoint.uncrackable;
2
3 import android.util.Base64;
4 import android.util.Log;
5
6 public class a {
7     public static boolean a(String str) {
8         byte[] bArr;
9         byte[] bArr2 = new byte[0];
10        try {
11            bArr = sg.vantagepoint.a.a.a(b("8d127684cbc37c17616d806cf50473cc"), Base64.decode("5UJiFctbmgbDoLXmpLl2mkno8HT4Lv8dlat8FxR2GOc=", 0));
12        } catch (Exception e) {
13            Log.d("CodeCheck", "AES error:" + e.getMessage());
14            bArr = bArr2;
15        }
16        return str.equals(new String(bArr));
17    }
18}
```



```
61 :goto_0
62
63 new-instance v1, Ljava/lang/String;
64 invoke-direct {v1, v0}, Ljava/lang/String;:-><init>([B)V
65 #codigo_añadido [INICIO] =>
66
67 const-string v2, "CodeCheck"
68
69 new-instance v3, Ljava/lang/StringBuilder;
70
71 invoke-direct {v3}, Ljava/lang/StringBuilder;:-><init>()V
72
73 const-string v4, "Secret string: "
74
75 invoke-virtual {v3, v4}, Ljava/lang/StringBuilder;:->append(Ljava/lang/String;)Ljava/lang/StringBuilder;
76
77 invoke-virtual {v3, v1}, Ljava/lang/StringBuilder;:->append(Ljava/lang/String;)Ljava/lang/StringBuilder;
78
79 invoke-virtual {v3}, Ljava/lang/StringBuilder;:->toString()Ljava/lang/String;
80
81 move-result-object v0
82
83 invoke-static {v2, v0}, Landroid/util/Log;:->d(Ljava/lang/String;Ljava/lang/String;)I
84
85 #<= codigo_añadido [FIN]
86
87
88 invoke-virtual {p0, v1}, Ljava/lang/String;:->equals(Ljava/lang/Object;)Z
89
90 move-result p0
91
92
93 return p0
94
95 .end method
```

- Actuamos a nivel Smali para conocer el valor de bArr



Análisis estático + Tampering

■ Resultado

The screenshot displays two windows. On the left, a terminal window titled 'Seleccionar Símbolo del sistema - adb logcat' shows logcat output for an Android application. The log includes various system messages and OpenGL renderer initialization details. On the right, an Android application titled 'Uncrackable1' is shown running on a 'Google Nexus 5 (1080x1920, 480dpi) - 192.168.144.107 - Genymotion' emulator. The app interface includes a text input field containing 'I want to believe', a 'VERIFY' button, and a success message 'Success!' with a 'OK' button. Below the app is a virtual keyboard. A portion of the Java code for the application is visible at the bottom left.

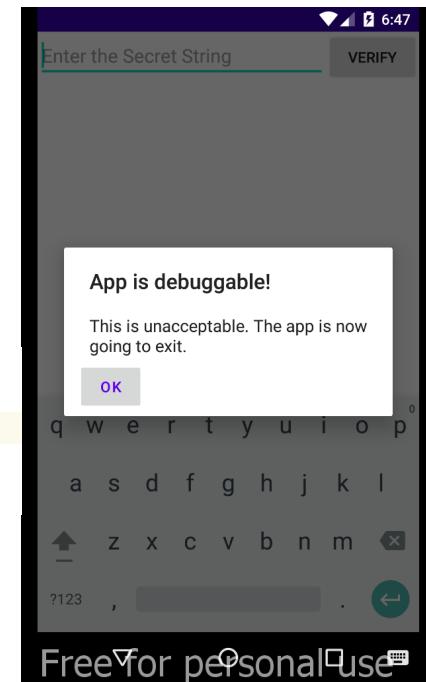
```
03-10 05:39:12.194 7781 7781 E libprocessgroup: failed to make and chown /acct/uid_0/proc/7781/libprocessgroup
03-10 05:39:12.194 7781 7781 W Zygote : createProcessGroup failed, kernel missing?
03-10 05:39:12.311 7781 7781 W System : ClassLoader referenced unknown path: /data
03-10 05:39:12.360 7781 7804 D OpenGLRenderer: Use EGL_SWAP_BEHAVIOR_PRESERVED: true
03-10 05:39:12.388 7781 7781 D : static HostConnection* HostConnection::create()
03-10 05:39:12.393 7781 7781 D : HostConnection::get() New Host Connection
03-10 05:39:12.416 7781 7781 D : HostComposition ext GL_OES_EGL_image_external
03-10 05:39:12.416 7781 7781 W : Process pipe failed
03-10 05:39:12.500 7781 7804 D libEGL : loaded /system/lib/egl/libEGL_emulation.so
03-10 05:39:12.500 7781 7804 D libEGL : loaded /system/lib/egl/libGLESv1_CM_emulation.so
03-10 05:39:12.504 7781 7804 D libEGL : loaded /system/lib/egl/libGLESv2_emulation.so
03-10 05:39:12.508 7781 7804 D : HostConnection::get() New Host Connection
03-10 05:39:12.509 7781 7804 D : HostComposition ext GL_OES_EGL_image_external
03-10 05:39:12.511 7781 7804 I OpenGLRenderer: Initialized EGL, version 1.4
03-10 05:39:12.512 7781 7804 W OpenGLRenderer: Failed to choose config with EGL_SMA
03-10 05:39:12.544 7781 7804 D EGL_emulation: eglCreateContext: 0xeea944e0: maj 3 m
03-10 05:39:12.547 1624 1693 E Surface : getSlotFromBufferLocked: unknown buffer: 0
03-10 05:39:12.761 628 649 I ActivityManager: Displayed owasp.mstg.uncrackable1/s
03-10 05:39:12.978 628 649 E eglCodecCommon: goldfish dma create_region: could no
03-10 05:39:15.347 7781 7781 D CodeCheck: Secret string: want to believe
03-10 05:39:15.394 287 679 D AudioFlinger: mixer(0xf4300000) throttle end: thrott
03-10 05:40:23.197 7781 7804 E Surface : getSlotFromBufferLocked: unknown buffer: 0
03-10 05:40:23.200 628 1138 W InputMethodManagerService: Window already focused, i
ken = android.os.BinderProxy@1a6cae
03-10 05:40:24.898 1034 I LatinIME: Starting input. Cursor position = 0,0
03-10 05:40:26.672 7781 7781 D CodeCheck: Secret string:I want to believe
03-10 05:40:26.716 287 679 D AudioFlinger: mixer(0xf4300000) throttle end: thrott
87     #<= codigo_añadido [FIN]
88
89
90     invoke-virtual {p0, v1}, Ljava/lang/String;
91
92     move-result p0
93
94     return p0
95 .end method
96
97 .method public static b(Ljava/lang/String;)V
98     .locals 7
99
100    invoke-virtual {p0}, Ljava/lang/String;
101
102    move-result v0
103
```



Solución utilizando el depurador

- En lugar de utilizar el código Smali, crear un proyecto vacío en AndroidStudio y copiar el manifiesto, los layouts de la app y el código decompilado de la misma
- Root Bypassing → retornar false en los métodos de la clase sg.vantagepoint.a.c
- Luego ejecutar en modo depuración:
 - Hay que actuar sobre la clase sg.vantagepoint.a.b

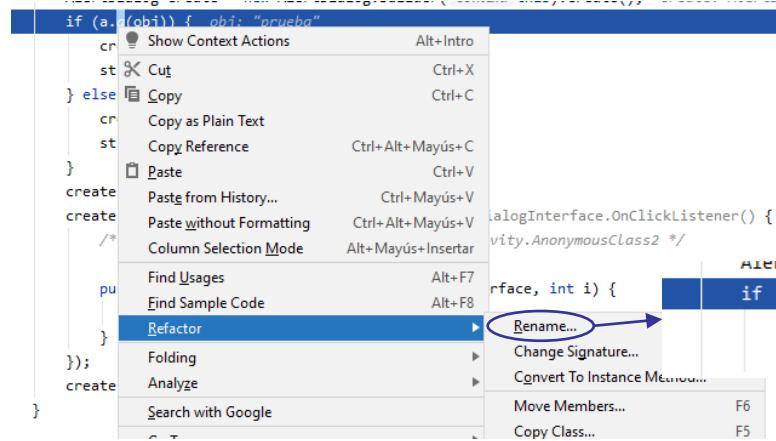
```
public class b {  
    public static boolean a(Context context) {  
        return false; //|(context.getApplicationContext().getApplicationInfo().flags & 2) != 0;  
    }  
}
```



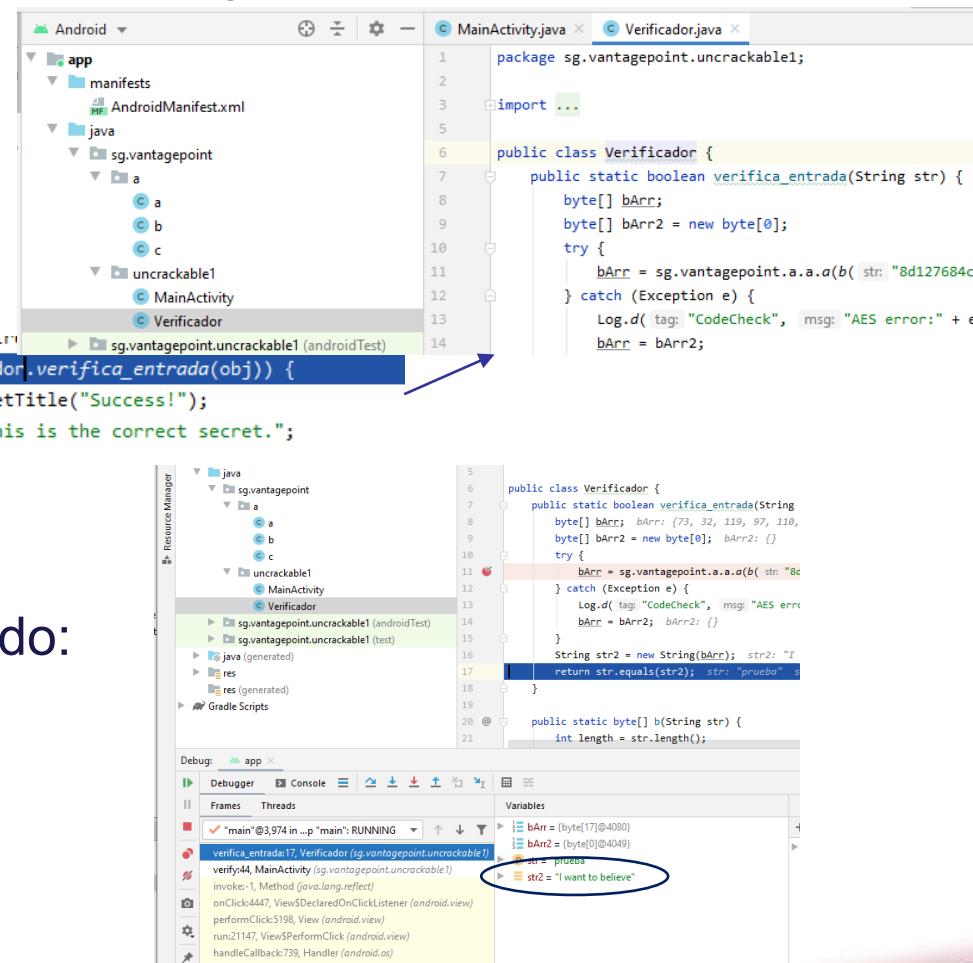


Solución utilizando el depurador

- Podemos combatir la ofuscación de código mediante el renombrado de clases y métodos:



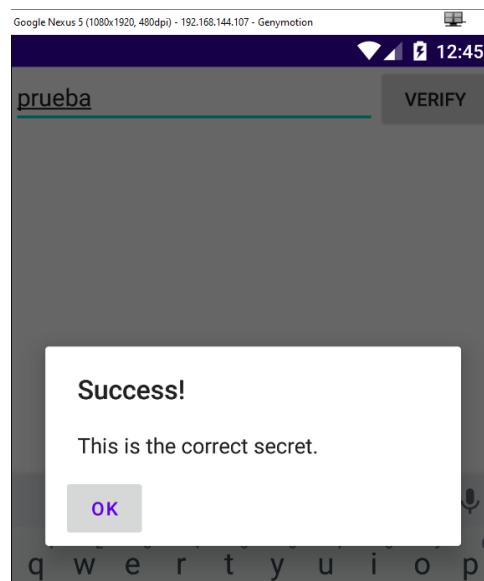
- Colocando un punto de ruptura (breakpoint) en el lugar adecuado:





Uso de Frida

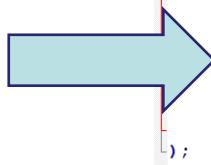
```
C:\ Símbolo del sistema - frida -U -l uncrackable1.js -f sg.vantagepoint.uncrackable1
C:\Users\jucar\Downloads\frida-server\scripts>frida -U -l uncrackable1.js -f sg.vantagepoint.un
Frida 14.2.13 - A world-class dynamic instrumentation toolkit
Commands:
  help      -> Displays the help system
  object?   -> Display information about 'object'
  exit/quit -> Exit
  ...
  ...
  More info at https://www.frida.re/docs/home/
Spawning `sg.vantagepoint.uncrackable1`. Use %resume to let the main thread start executing!
[Google Nexus 5:sg.vantagepoint.uncrackable1]> %resume
[Google Nexus 5:sg.vantagepoint.uncrackable1]>
*****
** CDM: Script para capturar el código secreto de la app UNCRACKABLE1 **
*****
> Tu código [prueba] es ahora el código correcto
> El código secreto que buscas es: [I want to believe]
```



```
uncrackable1.js uncrackable1b.js
1 Java.perform()
2
3   function () {
4     console.log('');
5     console.log('*****');
6     console.log('** CDM: Script para capturar el código secreto de la app UNCRACKABLE1 **');
7     console.log('*****');
8
9   function bufferToString(buf) {
10    var buffer = Java.array('byte', buf);
11    var result = "";
12    for(var i = 0; i < buffer.length; ++i){
13      result += String.fromCharCode(buffer[i]);
14    }
15    return result;
16  }
17
18  var clase = Java.use("sg.vantagepoint.a.a");
19
20  clase.a.implementation = function(ba1, ba2) {
21    const retval = this.a(ba1, ba2);
22    console.log('');
23    console.log("> El código secreto que buscas es: [" + bufferToString(retval)+"]");
24    return retval;
25  };
26
27
28  var verificador = Java.use("sg.vantagepoint.uncrackable1.Verificador");
29  verificador.verifica_entrada.implementation = function(str1) {
30    console.log('');
31    console.log("> Tu código [" + str1+"] es ahora el código correcto");
32    const retval = this.verifica_entrada(str1);
33    return true;
34  };
35
36
37
38 );
```

Mejora del script

```
Java.perform(  
    function () {  
        console.log('');  
        console.log('***** CDM: Script para capturar el código secreto de la app UNCRACKEABLE1 *****');  
        console.log('*****');  
  
        function bufferToString(buf) {  
            var buffer = Java.array('byte', buf);  
            var result = "";  
            for(var i = 0; i < buffer.length; ++i){  
                result += String.fromCharCode(buffer[i]);  
            }  
            return result;  
        };  
  
        var clase = Java.use("sg.vantagepoint.a.a");  
  
        clase.a.implementation = function(ba1, ba2) {  
            const retval = this.a(ba1, ba2);  
            console.log('');  
            console.log("> El código secreto que buscas es: [" + bufferToString(retval)+"]");  
            return retval;  
        };  
  
        var verificador = Java.use("sg.vantagepoint.uncrackable1.Verificador");  
        verificador.verifica_entrada.implementation = function(str1) {  
            console.log('');  
            console.log("> Tu código [" + str1+"] es ahora el código correcto");  
            const retval = this.verifica_entrada(str1);  
            return true;  
        };  
  
        var actividadPpal = Java.use("sg.vantagepoint.uncrackable1.MainActivity");  
        actividadPpal.onResume.implementation = function() {  
            this.onResume();  
            verificador.verifica_entrada("entrada_dummy");  
        };  
    };  
);
```





Caso de estudio Uncrackable2

- Solventar la detección de root y depuración utilizando Frida
 - Estrategia: evitar que la llamada a exit termine con la ejecución de la app

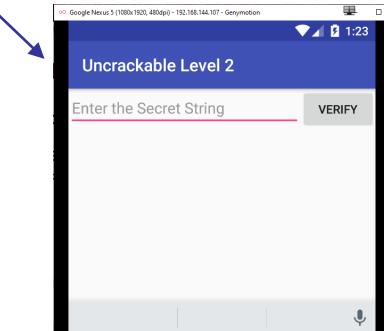
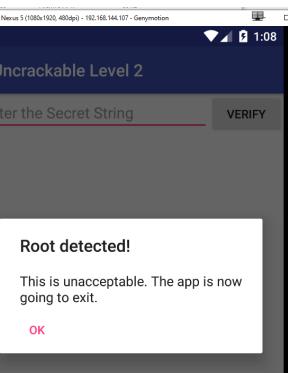
```
private void a(String str) {
    AlertDialog create = new AlertDialog.Builder(this).create();
    create.setTitle(str);
    create.setMessage("This is unacceptable. The app is now going to exit.");
    create.setButton(-3, "OK", new DialogInterface.OnClickListener() {
        /* class sg.vantagepoint.uncrackable2.MainActivity$AnonymousClass1 */

        public void onClick(DialogInterface dialogInterface, int i) {
            System.exit(0);
        }
    });
    create.setCancelable(false);
    create.show();
}
```



```
uncrakable1.js | uncrakable2.js
1 // owasp.mstg.uncrackable2
2 // $ frida -U -f owasp.mstg.uncrackable2 -l uncrackable2.js
3
4 Java.perform(
5
6     function () {
7         console.log('!');
8         console.log('******');
9         console.log('** CDM: Script para capturar el código secreto de la app UNCRACKABLE2 **');
10        console.log('******');
11
12        var system = Java.use('java.lang.System');
13        system.exit.implementation = function (arr1) {
14            console.log("");
15            console.log(">> Inhibo la llamada a System.exit, con lo que la app no terminará");
16        };
17
18    });
19
20
21
22 );
```

```
C:\Users\jucar\Downloads\frida-server\scripts>frida -U -l uncrackable2.js -f owasp.mstg.uncrackable2
|_ /_ \_| Frida 14.2.13 - 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/
|_ /_ \_| Spawning `owasp.mstg.uncrackable2`. Use %resume to let the main thread start executing!
|_ /_ \_| [Google Nexus 5::owasp.mstg.uncrackable2] -> %resume
|_ /_ \_| [Google Nexus 5::owasp.mstg.uncrackable2] ->
*****
** CDM: Script para capturar el código secreto de la app UNCRACKABLE2 **
*****
>> Inhibo la llamada a System.exit, con lo que la app no terminará
```





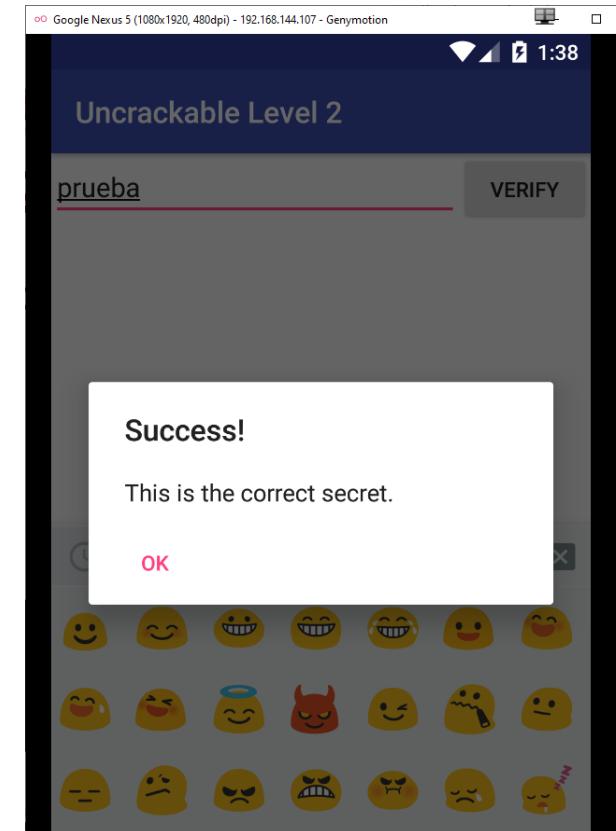
Uncrackable2: string secreto

```
Projects
UnCrackable-Level2.apk-decomp...
lib
    arm64-v8a
    armeabi-v7a
x86
    libfoo.so
    x86_64
original
res
smali
sources
```

```
CodeCheck.java
1 package sg.vantagepoint.uncrackable2;
2
3 public class CodeCheck {
4     private native boolean bar(byte[] bArr);
5
6     public boolean a(String str) {
7         return bar(str.getBytes());
8     }
9 }
10
```

```
script
Frida
var codeCheck = Java.use('sg.vantagepoint.uncrackable2.CodeCheck');
codeCheck.a.implementation = function (arr1) {
    console.log("");
    console.log(">> [CodeCheck.a] Verificando el string ["+arr1+"]");
    const retVal = this.a(arr1);
    console.log(">> Pero pasando del resultado (devuelvo true siempre)");
    return true;
};
```

```
*****
** CDM: Script para capturar el código secreto de la app UNCRACKABLE2 **
*****
[Google Nexus 5::owasp.mstg.uncrackable2]->
>> [CodeCheck.a] Verificando el string [prueba]
>> Pero pasando del resultado (devuelvo true siempre)
```



Free for personal use

Uncrackable2: ¿y el string secreto?

- El problema es que el string se gestiona en una librería de la que no disponemos del código fuente, pero sabemos que es la librería **libfoo.so**
- **Solución:** decompilar la librería y estudiar su estructura
 - Localizarla en el directorio donde hayamos decompilado la app
 - Cualquier decompilador nos valdría → usaremos **radare2** por ser una herramienta multiplataforma, aunque se prioriza Linux
 - Documentación: <https://radare.gitbooks.io/radare2book/content>
 - Descarga:
 - Windows: <https://rada.re/r/>
 - » Instalación: <https://medium.com/@jacob16682/debugging-using-radare2-and-windows-5e58677bf943>
 - Linux: <https://rada.re/n/radare2.html>
 - » Instalación: <https://rada.re/n/radare2.html>

Análisis de libfoo.so con radare2

```
C:\Users\jucar\Downloads\UnCreakableApps\Level2\UnCrackable-Level2.apk-decompiled\lib\x86\r2 -A libfoo.s  
[x] Analyze all flags starting with sym. and entry0 (aa)  
[x] Analyze function calls (aac)  
[x] Analyze len bytes of instructions for references (aar)  
[x] Check for objc references  
[x] Check for vtables  
[x] Type matching analysis for all functions (aft)  
[x] Propagate noreturn information  
[x] Use -AA or aaaa to perform additional experimental analysis.  
- - - - -  
[0x000000600]> afl  
0x000000600 1 35 entry0  
0x000000550 1 6 sym.imp.__cxa_finalize  
0x000000f30 1 40 entry.init0  
0x000000580 1 6 sym.imp.waitpid  
0x000000560 1 6 sym.imp.__cxa_atexit  
0x000000570 1 6 sym.imp.__stack_chk_fail  
0x0000005b0 1 6 sym.imp.fork  
0x0000005d0 1 6 sym.imp.getppid  
0x0000005a0 1 6 sym.imp._exit  
0x0000005e0 1 6 sym.imp.ptrace  
0x0000005f0 1 6 sym.imp.strncmp  
0x0000005c0 1 6 sym.imp.pthread_create  
0x000000590 1 6 sym.imp.pthread_exit  
0x000000640 2 21 -> 6 entry.fini0  
0x000000659 3 22 fcn.00000659  
0x000000679 1 38 fcn.00000679  
0x0000006cf 3 81 fcn.000006cf  
0x0000006a9 1 12 fcn.000006a9  
0x000000720 10 209 fcn.00000720  
0x000000f60 8 199 sym.Java_sg_vantagepoint_unbreakable2_CodeCheck_bar  
[0x000000600]> s sym.Java_sg_vantagepoint_unbreakable2_CodeCheck_bar
```

libfoo.so

- Diagrama de flujo

```
[0x000000f60]> pd
/ 199: sym.Java_sg_vantagepoint_uncrackable2_CodeCheck_bar (int32
    ; var int32_t var_ch @ ebp-0xc
    ; arg int32_t arg_8h @ ebp+0x8
    ; arg int32_t arg_10h @ ebp+0x10
[0x000000f60]> VV
```

```
mov dword [var_14h], eax
cmp byte [ebx + 0x40], 1
jne 0x1007
f t
|
|
0xf8b [oe]
mov edi, dword [arg_8h]
; 'Than'
; [0x6e616854:4]=-1
mov dword [esp], 0x6e616854
; 'ks f'
; [0x6620736b:4]=-1
mov dword [var_28h], 0x6620736b
; 'or a'
; [0x6120726f:4]=-1
mov dword [var_24h], 0x6120726f
; 'll t'
; [0x74206c6c:4]=-1
mov dword [var_20h], 0x74206c6c
; 'he'
; [0x6568:2]=0xffff
mov word [var_1ch], 0x6568
; ' fis'
; [0x73696620:4]=-1
mov dword [var_1ah], 0x73696620
; 'h'
; [0x60d0:a] 0x014
; "D!"
mov word
mov eax,
sub esp,
push 0
push dword [arg_10h]
push edi
call dword [eax + 0x2e0];[oc]
add esp, 0x10
mov esi, eax
mov eax, dword [edi]
sub esp, 8
push dword [arg_10h]
push edi
call dword [eax + 0x2e0];[od]
add esp, 0x10
cmp eax, 0x17
jne 0x1007
f t
|
|
0xff0 [og]
sub esp, 4
lea eax, [var_2ch]
; size_t n
push 0x17
; const char *s2
push eax
; const char *s1
push esi
; int strcmp(const char *s1, const char *s2, size_t n)
call sym.imp.strncmp;[of]
add esp, 0x10
test eax, eax
je 0x101e
t f
|
|
```

Thanks for all the fishD!

n = 0x17 = 23



libfoo.so

```
function attachToStrncmp() {
    Interceptor.attach(Module.getExportByName('libfoo.so', 'strcmp'), {
        // strcmp recibe 3 argumentos, str1, str2, max número de caracteres a comparar
        onEnter: function (args) {
            var numC = args[2].toInt32();
            if (numC != 23){
                return; //si no son 23 caractéres los comparados no hacemos caso a la comparación
            }
            var str1 = args[0].readUtf8String();
            var str2 = args[1].readUtf8String();
            console.log(">> [attachToStrncmp] strcmp ejecutado\n"
                       + " - str1: " + str1 + "\n"
                       + " - str2: " + str2 + "\n"
                       + " - numC: " + numC + "\n");
        },
    });
}

var actividadPpal = Java.use("sg.vantagepoint.uncrackable2.MainActivity");
actividadPpal.onResume.implementation = function() {
    console.log(">> [MainActivity.onResume] Allá vamos");
    this.onResume();
    attachToStrncmp();
};

>> [attachToStrncmp] strcmp ejecutado
- arg0: settings_system_version
- arg1: settings_system_version
- arg2: 23

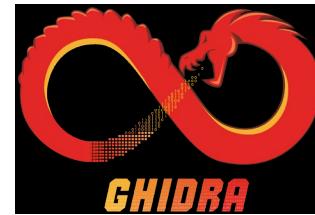
>> [CodeCheck.a] Verificando el string [01234567890123456789123]
>> [attachToStrncmp] strcmp ejecutado
- arg0: 01234567890123456789123able2_Co
- arg1: Thanks for all the fish
- arg2: 23

>> Pero pasando del resultado (devuelvo true siempre)
>> [attachToStrncmp] strcmp ejecutado
- arg0: settings_system_version
- arg1: settings_secure_version
- arg2: 23
```



libfoo.so

- Desensamblado de la función CodeCheck::bar ofrecido por Ghidra
 - La Agencia de Seguridad Nacional del Gobierno de los Estados Unidos liberó hace poco todo el código fuente de una de sus herramientas más poderosas
 - Su nombre es Ghidra, un framework de ingeniería inversa para software.
 - Objetivo: “traducir todo el código que transmite un software a un ordenador a una estructura que puedan entender un humano”
 - <https://github.com/NationalSecurityAgency/ghidra>
 - <https://ghidra-sre.org/>



```
C:\Decompile: Java_sg_vantagepoint_uncrackable2_CodeCheck_bar - (libfoo.so)
13 undefined4 local_24;
14 undefined2 local_20;
15 undefined4 local_1e;
16 undefined2 local_1a;
17 int local_18;
18
19 local_18 = *(int *) (in_GS_OFFSET + 0x14);
20 if (DAT_00014008 == 'x01') {
21     local_30 = 0xe6e616854;
22     local_2c = 0x6620736b;
23     local_28 = 0x6120726f;
24     local_24 = 0x74206c6c;
25     local_20 = 0x6568;
26     local_1e = 0x73696620;
27     local_1a = 0x68;
28     __s1 = (char *) (**(code **) (*param_1 + 0x2e0)) (param_1, param_3, 0);
29     iVar1 = (**(code **) (*param_1 + 0x2ac)) (param_1, param_3);
30     if (iVar1 == 0x17) {
31         iVar1 = strncmp (__s1, (char *) &local_30, 0x17);
32         if (iVar1 == 0) {
33             uVar2 = 1;
34             goto LAB_00011009;
35         }
36     }
37 }
38 uVar2 = 0;
39 LAB_00011009:
40 if (* (int *) (in_GS_OFFSET + 0x14) == local_18) {
41     return uVar2;
42 }
43 /* WARNING: Subroutine does not return */
44 _stack_chk_fail();
45 }
```



libfoo.so

■ Automatizar la extracción del resultado

```
var actividadPpal = Java.use("sg.vantagepoint.uncrackable2.MainActivity");
actividadPpal.onResume.implementation = function() {
    console.log(">> [MainActivity.onResume] Allá vamos");
    this.onResume();
    attachToStrncmp();
}

var entrada23Caracteres = '01234567890123456789123';
Java.choose("sg.vantagepoint.uncrackable2.CodeCheck", {
    onMatch : function(instance) {
        instance.a(Java.use("java.lang.String").$new(entrada23Caracteres));
        return "stop";
    },
    onComplete:function() {}
});
Interceptor.detachAll();
};
```

```
*****
** CDM: Script para capturar el código secreto de la app UNCRACKEABLE2 **
*****
>> [MainActivity.onResume] Allá vamos

>> [CodeCheck.a] Verificando el string [01234567890123456789123]
>> [attachToStrncmp] strcmp ejecutado
  - str1: 01234567890123456789123able2_Co
  - str2: Thanks for all the fish
  - numC: 23
```



Repositorio de scripts Frida

- <https://codeshare.frida.re/>
- <https://github.com/t0thkr1s/frida>
- <https://github.com/as0ler/frida-scripts>
- <https://github.com/dweinstein/awesome-frida>
- ...



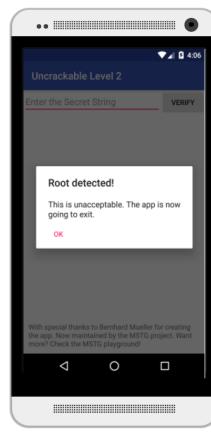
MobSF: Análisis dinámico



RECENT SCANS STATIC ANALYZER DYNAMIC ANALYZER API DOCS ABOUT Search MDS

Dynamic Analyzer - owasp.mstg.uncrackable2

Stop Screen Remove MobSF RootCA Start Exported Activity Tester Start Activity Tester Take a Screenshot Logcat Stream Generate Report



Dynamic Analyzer Errors

Setting up Dynamic Analysis environment
Running HTTPS intercepting proxy
Invoking MobSF agents
Environment is ready for user assisted dynamic analysis.
Navigate through all the flows of the app manually.
Screen streaming started

Frida Scripts

Default

API Monitoring SSL Pinning Bypass Root Detection Bypass
 Debugger Check Bypass

Auxiliary

Enumerate Loaded Classes Capture Strings Capture String Comparisons
 Enumerate Class methods
 java.io.File
 Search Class Patterns
 ssl.Trust*
 Trace Class Methods
 java.net.Socket,java.io.File,java.lang.String

Start Instrumentation Frida Live Logs

Frida Code Editor

```
1 Java.perform(function() {  
2   // Use send() for logging  
3 });  
4
```

Available Scripts (Use CTRL to choose multiple)

aes_key
bypass_flag_secure
default
file_trace

Shell Access

Thu Mar 11 2021 10:04:32 GMT+0100 (hora estándar de Europa central)
Enter "help" for more information.
[root@android] #

MobSF Android Runtime
Android instance: 192.168.144.107:5555

Dynamic Analyzer Supports

- Genymotion Android VM version 4.1 to 10.0 (x86, upto API 29)
- Android Studio Emulator AVD version 5.0 to 9.0 (arm, arm64, x86, and x86_64 upto API 28)
- We recommend using Android 7.0 and above.

For Android versions less than 5, we use Xposed Framework and therefore, you must MobSFy the Android Runtime prior to Dynamic Analysis.
Other versions use Frida and are automatically MobSFyed on the go.

HTTPS Traffic Interception
For Android version 4.1 - 4.3, set Android VM proxy as <Host IP>:1337
For Android versions 4.4 - 10, global proxy settings are automatically applied.
For more information, please refer Documentation.

Apps Available for Dynamic Analysis

APP	FILE NAME	PACKAGE	ACTION
Uncrackable Level 2 - 1.0	UnCrackable-Level2.apk	owasp.mstg.uncrackable2	<input type="button" value="Start Dynamic Analysis"/> <input type="button" value="View Report"/>

Showing 1 to 1 of 1 entries



5. Vulnerabilidades en apps Android: Análisis dinámico

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