

# EJERCICIOS ANÁLISIS Y GENERACIÓN DE INFORMES

## Prerrequisitos

Android  
Kali Linux

## Ejercicio 1 - MobSF

- Análisis estático y generación de informe de la aplicación que elijas utilizando MobSF
  - En primer lugar, conectamos el dispositivo Android a nuestra Kali

```
(root@kali)-[~/MobSF]
# adb connect 10.0.2.10:5555
connected to 10.0.2.10:5555
```

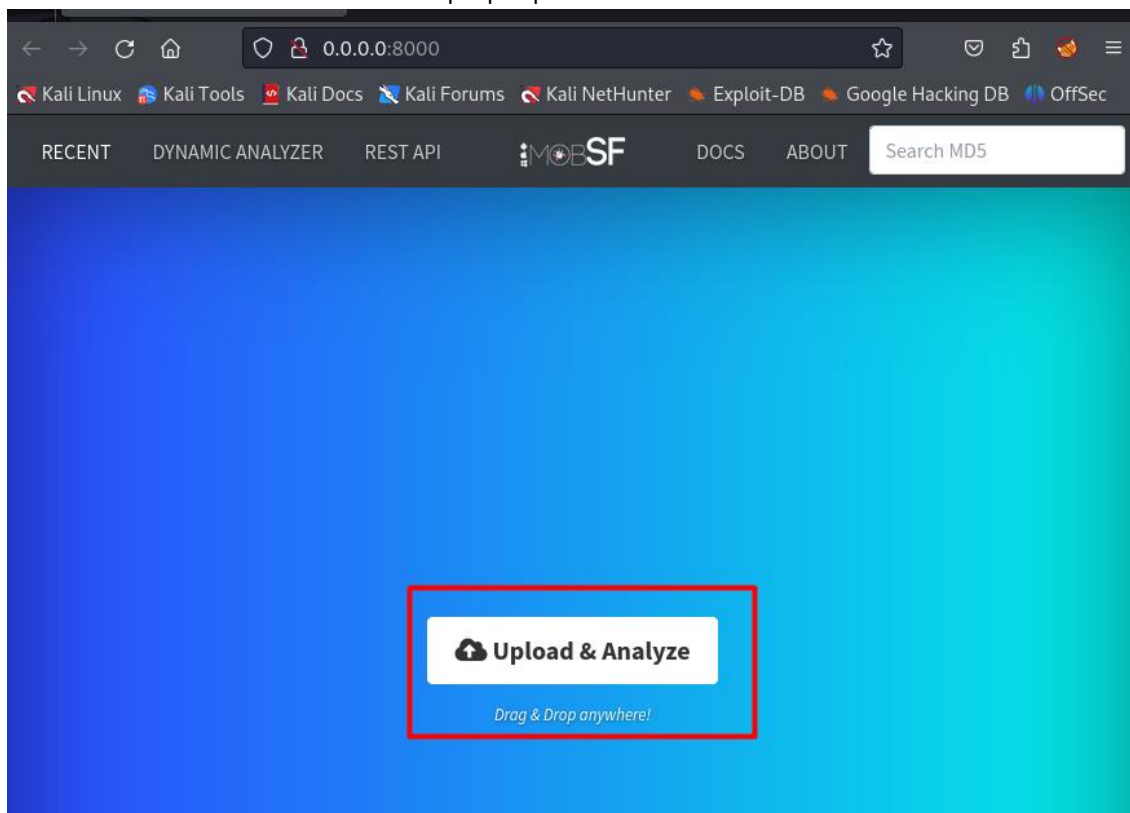
- 
- Luego confirmamos que esté conectada

```
(root@kali)-[~/Software/AplicacionesMóviles/MobSF1]
# adb devices
List of devices attached
10.0.2.10:5555 device
```

- 
- Una vez confirmado esto ponemos en marcha el Docker para poder acceder a MobSF

```
(root@kali)-[~/Software/AplicacionesMóviles]
# docker run -it --rm -p 8000:8000 opensecurity/mobile-security-framework-mobsf:l
atest
```

- 
- Una vez hecho esto subimos el archivo .apk que queremos analizar

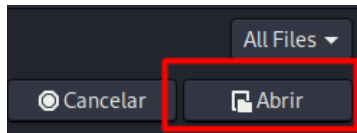


- 
- Seleccionamos el archivo

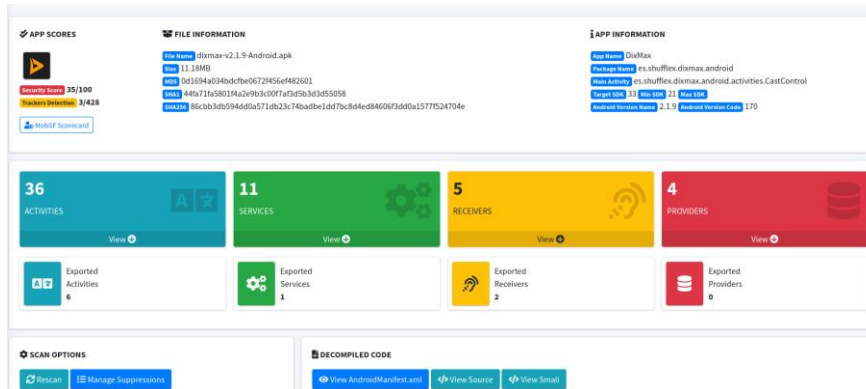
	Carpeta personal	Nombre	Tamaño	Tipo	Modificado
	Escritorio	dixmax-v2.1.9-Android.apk	11,7 MB	paquete de Android	13:00

-

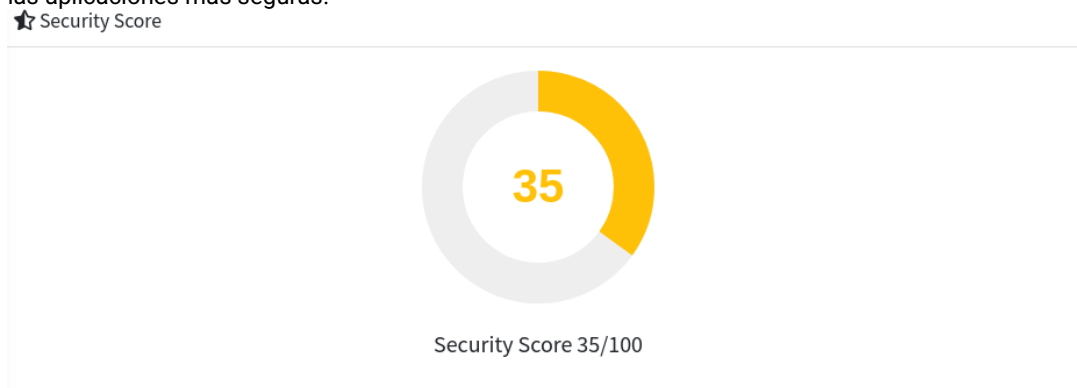
- Y lo abrimos



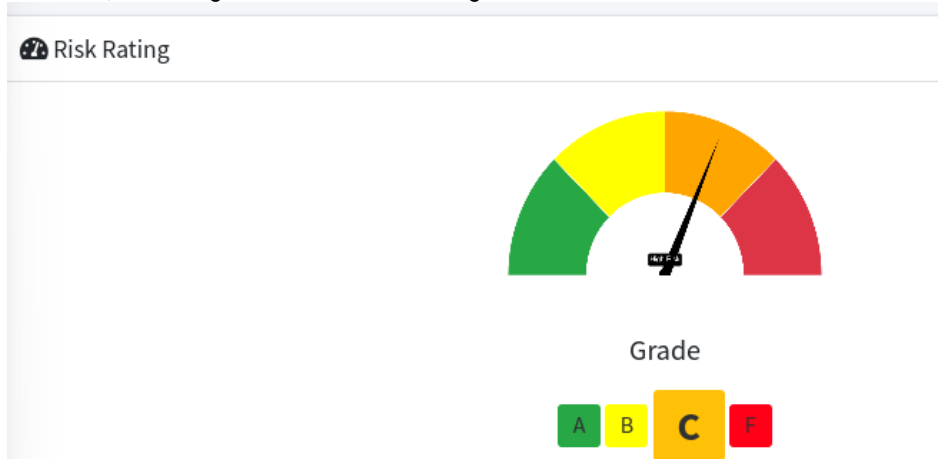
- 
- Como resultado tenemos lo siguiente



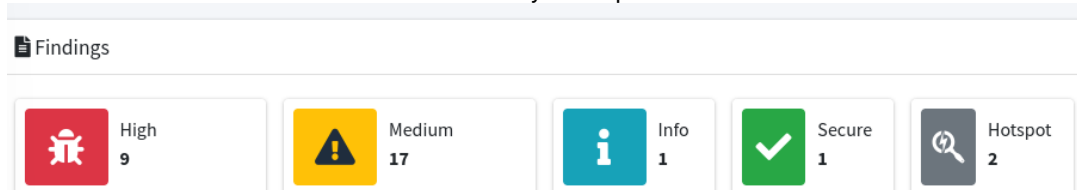
- Explica toda la información estática generada por MobSF en el informe
  - Podemos ver que la aplicación tiene un 35 sobre 100 en seguridad, por tanto no es de las aplicaciones más seguras.



- 
- De hecho, tiene un grado C en cuanto a riesgo



- 
- Encontramos 9 vulnerabilidades altas 17 medias y 2 hotspot a tener en cuenta



- - **High:** de los 9 obtenidos 8 se encuentran en el manifest y uno en la propia network, concretamente que la configuración base está configurada de forma insegura para permitir el tráfico de texto claro a todos los dominios.

- **Medium:** de los 17 obtenidos 1 se halla en el certificado, otro se refiere a la network, 4 de ellos son del manifiesto, la gran mayoría son de código ya que son 9, 1 de trackers y 1 de secrets, concretamente a que esta aplicación podría contener secretos codificados
- Hay servidores repartidos por todo el mundo y se concentra en china

🌐 SERVER LOCATIONS



DOMAIN	↑↓	COUNTRY/REGION	↑↓
config.unityads.unitychina.cn		<b>IP:</b> 119.167.231.221 <b>Country:</b> China <b>Region:</b> Shandong <b>City:</b> Qingdao	

## Ejercicio 2 - MobSF y Android


- Análisis dinámico y generación de informe de la aplicación que elijas utilizando MobSF

- Instalamos la app en el teléfono, para ello realizamos el siguiente comando

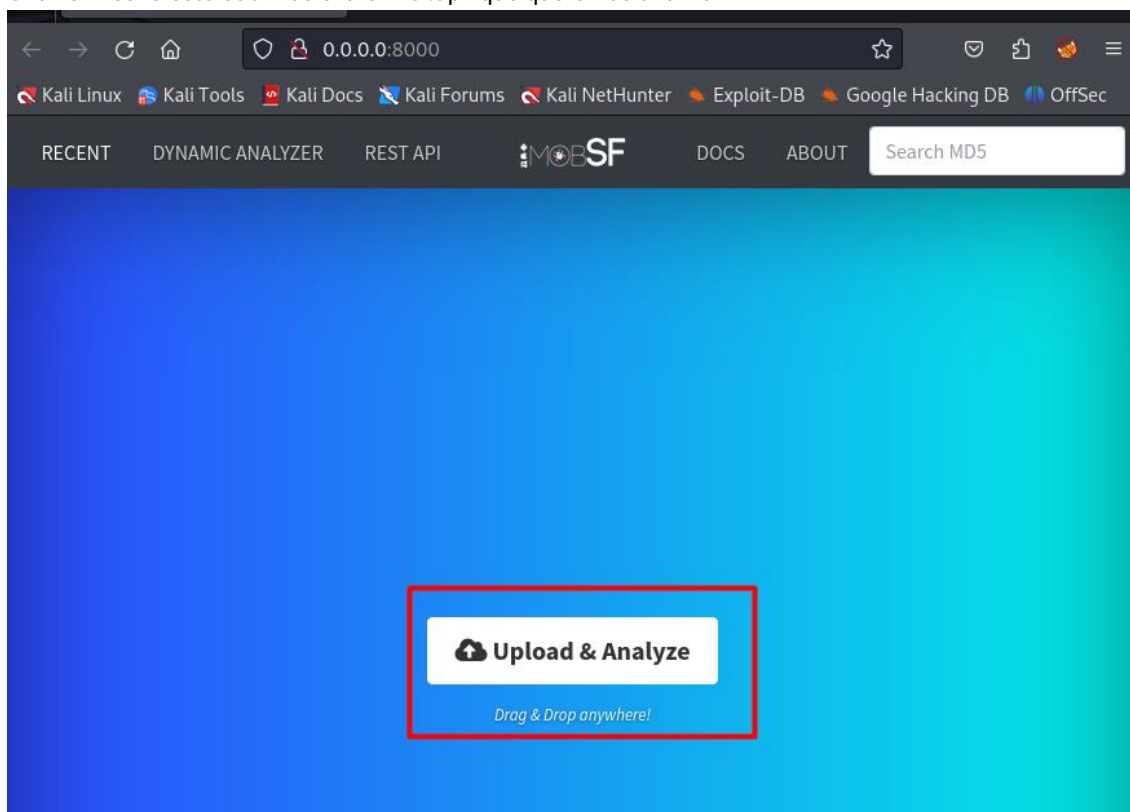
```
(root@kali)-[/home/kali/Escritorio]
# adb install /home/kali/Escritorio/dixmax-v2.1.9-Android.apk
Performing Streamed Install
Success
```

- Una vez hecho esto corremos la app en dinámico

```
(root@kali)-[~/Software/AplicacionesMóviles]
# docker run -it --rm -p 8000:8000 -p 1337:1337 -e MOBSF_ANALYZER_IDENTIFIER=10.0
.2.10:5555 opensecurity/mobile-security-framework-mobsf:latest
[INFO] 25/Oct/2023 11:51:25 -
```



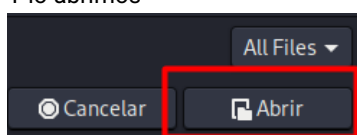
- Una vez hecho esto subimos el archivo .apk que queremos analizar



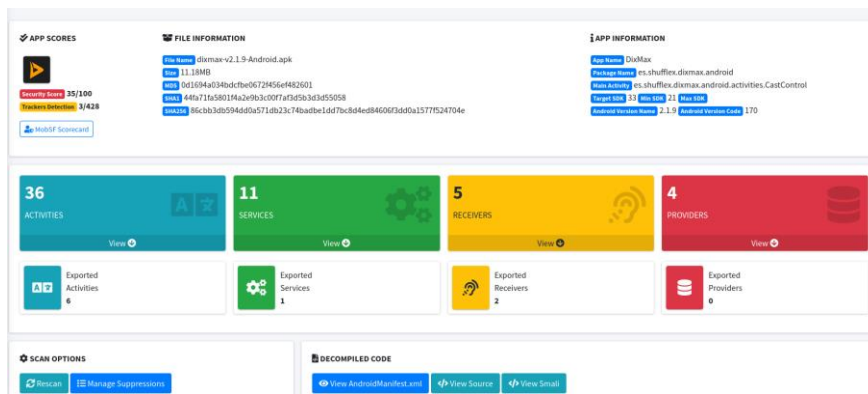
- Seleccionamos el archivo

Carpeta personal		Nombre	Tamaño	Tipo	Modificado
Escritorio		dixmax-v2.1.9-Android.apk	11,7 MB	paquete de Android	13:00

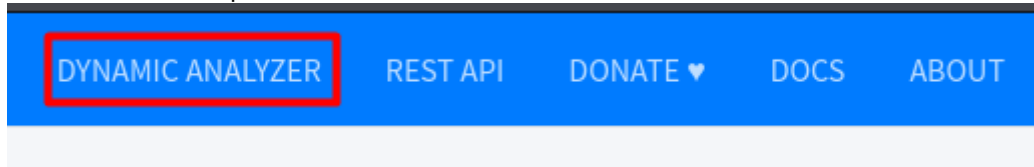
- Y lo abrimos



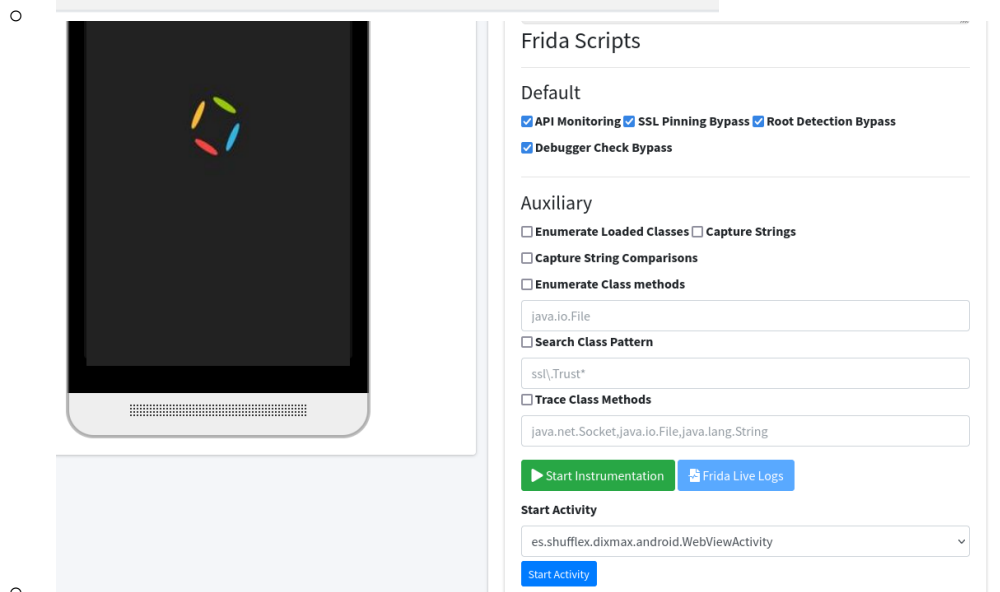
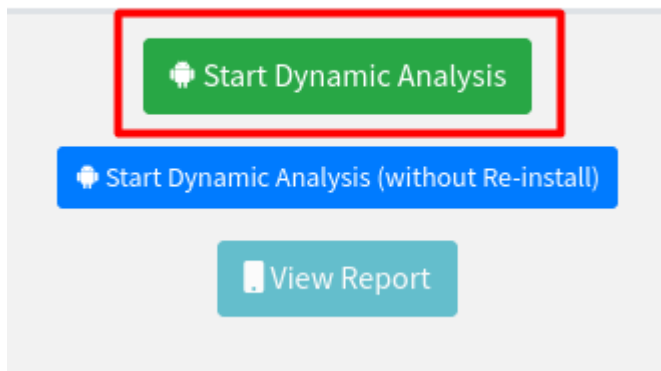
- Como resultado tenemos lo siguiente



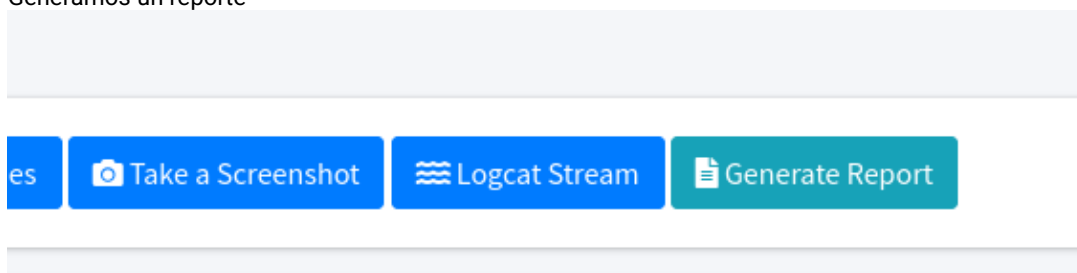
- Una vez dentro de la pantalla accedemos a DYNAMIC ANALYZER



- Tras esto empezamos el análisis dinámico

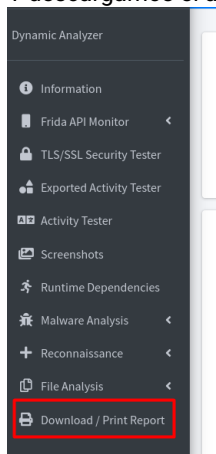


- Generamos un reporte



-

- Y descargamos el archivo en pdf



- El archivo pdf luce así

dynamic Analysis [http://0.0.0.0:8000/dynamic\\_report/b90cd09e1a34c23b703bdfid6525e645](http://0.0.0.0:8000/dynamic_report/b90cd09e1a34c23b703bdfid6525e645)

#### INFORMATION

Raw Logs

#### BASE64

Search:

CLASS	METHOD
android.util.Base64	encode
	Arguments: [[91, 68, 69, 70, 65, 85, 76, 84, 93], 0, 9, 11]
	Result: 87,48,82,70,82,107,70,86,84,70,82,100
	Return Value: 87488270821077086847082100
	Called From: android.util.Base64.encode(Base64.java:494)

- Explica toda la información dinámica generada por MobSF en el informe
  - En el reporte podemos encontrar información interesante acerca de esta aplicación uno de estas es la localización de los servidores que parece ser que están en California

#### SERVER LOCATIONS



- Al no haber interactuado con la aplicación durante el análisis no se encuentran datos que podría haber sido interesante de conocer como el TLS security

Search:

TESTS	RESULT
Cleartext Traffic Test	Not Tested
TLS Misconfiguration Test	Not Tested
TLS Pinning/Certificate Transparency Bypass Test	Not Tested
TLS Pinning/Certificate Transparency Test	Not Tested

Showing 1 to 4 of 4 entries

Previous 1 Next

- En el siguiente cuadro especifica lo ya dicho en relación a la localización de los servidores utilizados para la detección de malware

#### DOMAIN MALWARE CHECK

Search:

DOMAIN	STATUS	GEOLOCATION
connectivitycheck.gstatic.com	good	<b>IP:</b> 142.250.200.131 <b>Country:</b> United States of America <b>Region:</b> California <b>City:</b> Mountain View <b>Latitude:</b> 37.405991 <b>Longitude:</b> -122.078514 <b>View:</b> <a href="#">Google Map</a>
firebase-settings.crashlytics.com	good	<b>IP:</b> 142.250.201.67 <b>Country:</b> United States of America <b>Region:</b> California <b>City:</b> Mountain View <b>Latitude:</b> 37.405991 <b>Longitude:</b> -122.078514 <b>View:</b> <a href="#">Google Map</a>

- Además de todo esto se encontraron varios URL's y una hilera de caracteres que la app reconoció como un correo

#### URLS

<https://firebaseinstallations.googleapis.com/v1/projects/api-6221998778630643160-114593/installations>  
[http://connectivitycheck.gstatic.com/generate\\_204](http://connectivitycheck.gstatic.com/generate_204)  
[https://play.googleapis.com/play/log?format=raw&proto\\_v2=true](https://play.googleapis.com/play/log?format=raw&proto_v2=true)  
[https://firebase-settings.crashlytics.com/spi/v2/platforms/android/gmp/1:288834995628:android:6dc5794d0850f14f/settings?instance=3a506c9435bfae4202ecda8f564574ca621fb67a&build\\_version=322&display\\_version=2.5.0&source=1](https://firebase-settings.crashlytics.com/spi/v2/platforms/android/gmp/1:288834995628:android:6dc5794d0850f14f/settings?instance=3a506c9435bfae4202ecda8f564574ca621fb67a&build_version=322&display_version=2.5.0&source=1)  
[https://www.google.com/generate\\_204](https://www.google.com/generate_204)

#### EMAILS

234113032@23.41

- Hay información de los archivos en base64

#### BASE64

Search:

CLASS	METHOD
android.util.Base64	<b>encode</b>  <i>Arguments:</i> [[91, 68, 69, 70, 65, 85, 76, 84, 93], 0, 9, 11]  <i>Result:</i> 87,48,82,70,82,107,70,86,84,70,82,100  <i>Return Value:</i> 87488270821077086847082100  <i>Called From:</i> android.util.Base64.encode(Base64.java:494)

-

○ Además de información del dispositivo

🔍 DEVICE DATA

Search:

10/25/23,

analysishttp://0.0.0.0:8000/dynamic\_report/b90cd09e1a34c23b703bdfd6525

CLASS	↑↓METHOD↑↓
android.content.ContentResolver	<div>query</div> <div>Arguments: [&lt;instance: android.net.Uri, \$className: android.net.Uri\$HierarchicalUri&gt;', None, None, None]</div> <div>Result: [object Object]</div> <div>Called From: android.content.ContentResolver.query(ContentResolver.java:752)</div>
android.content.ContentResolver	<div>query</div>

○