lnnlt

Android Reverse Shell
For Beginners

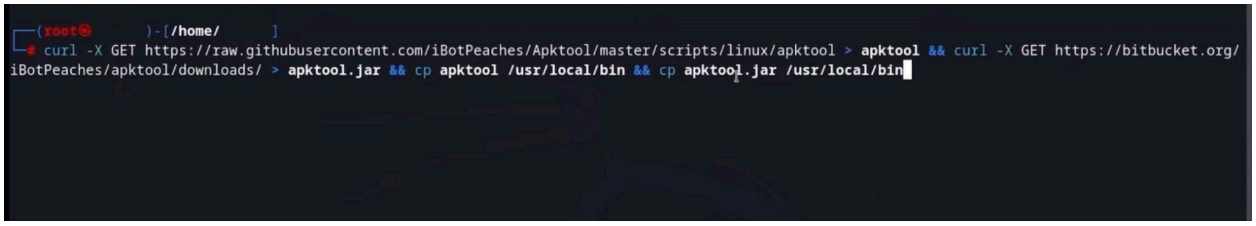
Introduction

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:

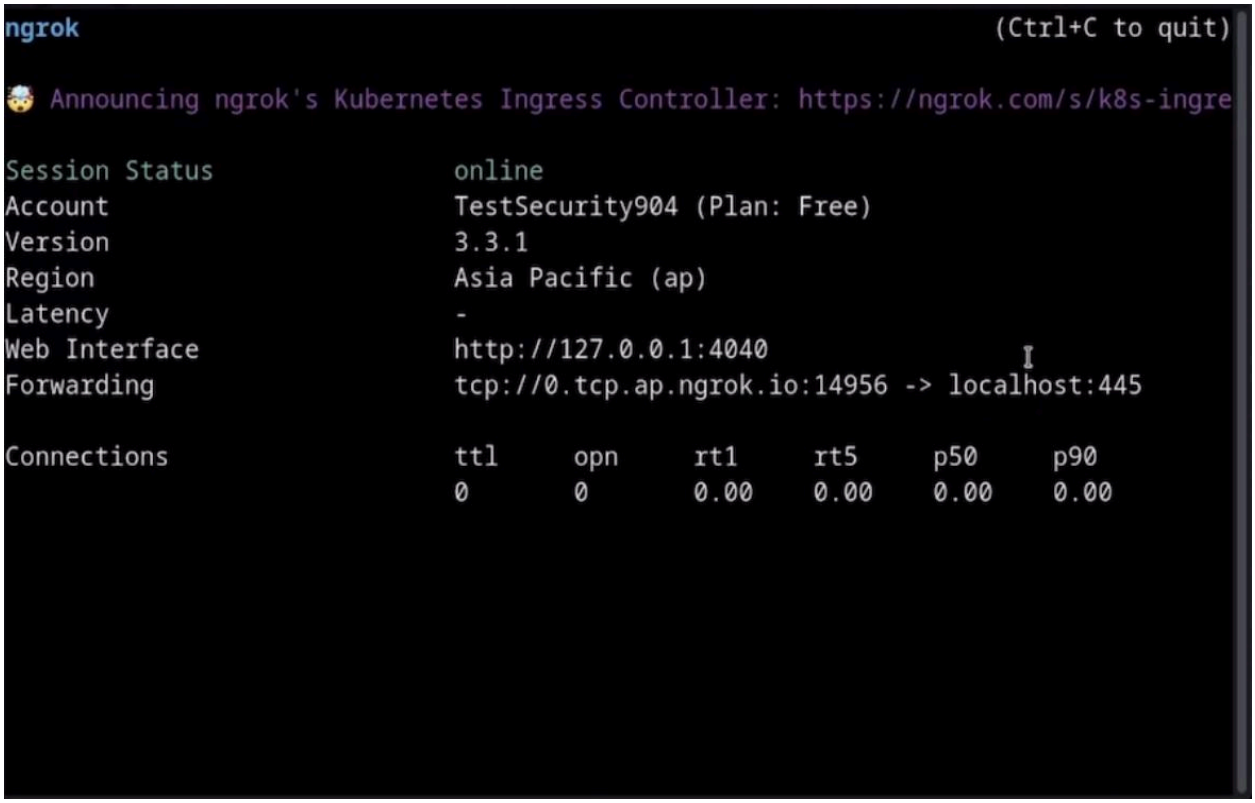
If encountering any issues with apktool, it is recommended to utilize the following command for resolution:



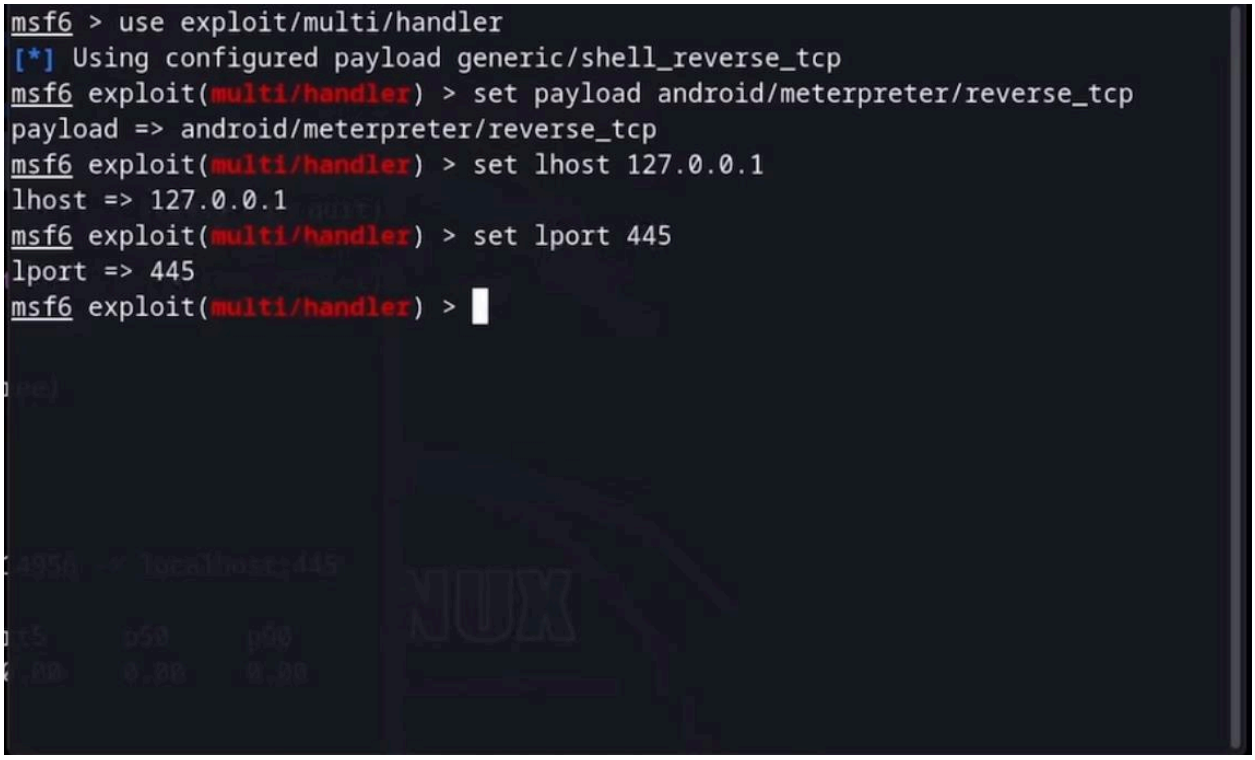
Chapter II

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:

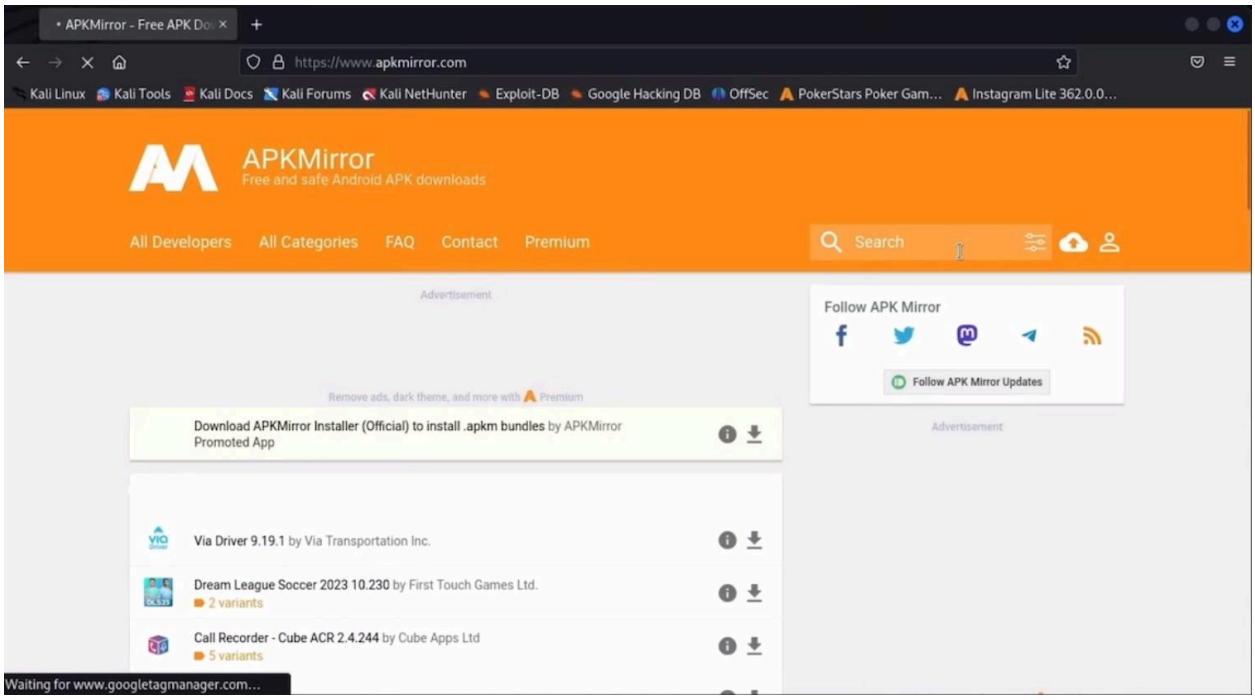


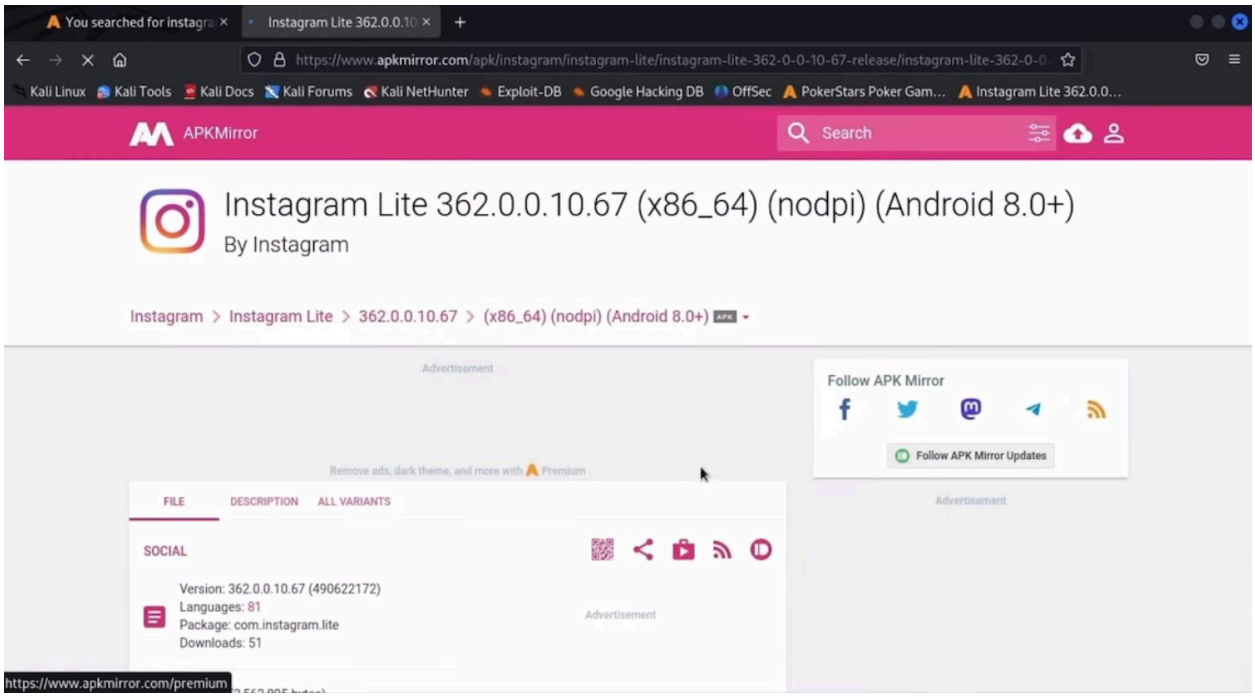
We proceed by establishing a listener using Metasploit:



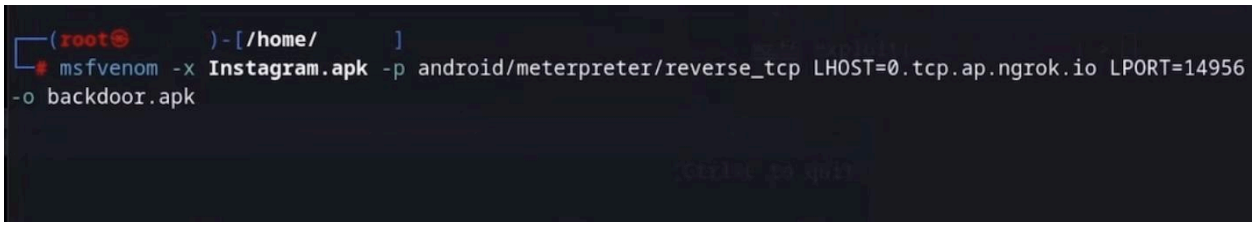
Chapter III

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:

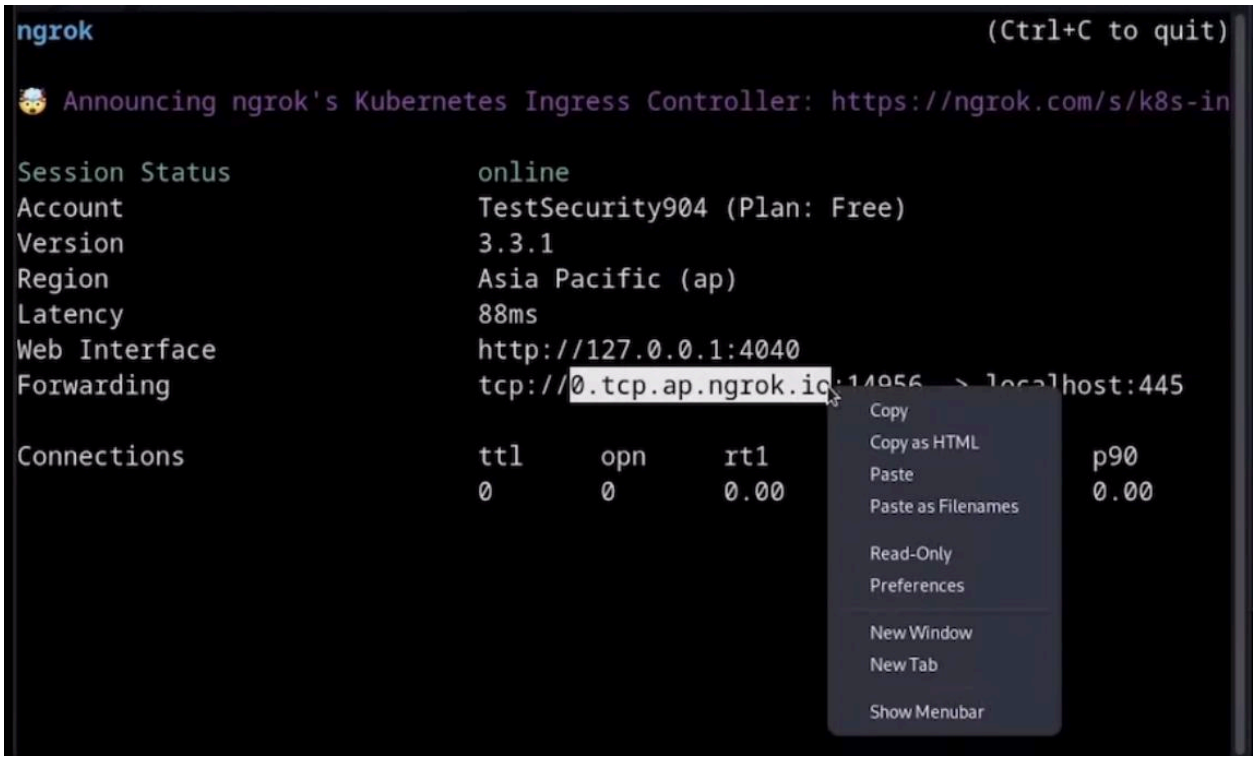




Using the following command, I will incorporate the malware into the .apk file:

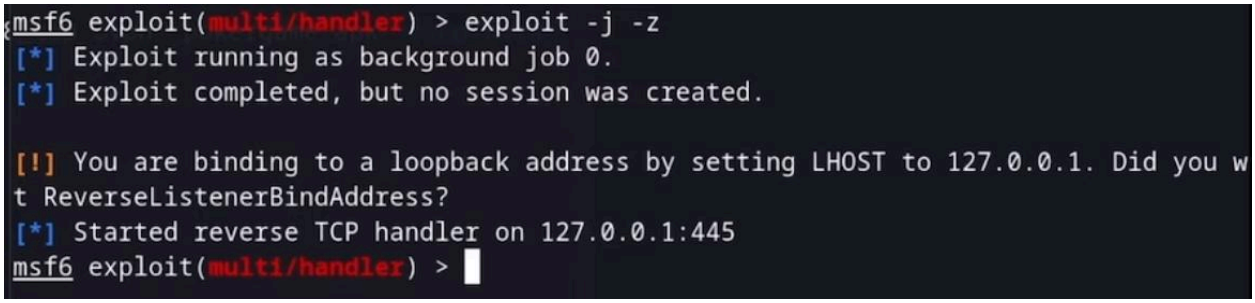


The values for LHOST and LPORT parameters correspond to those previously presented here:



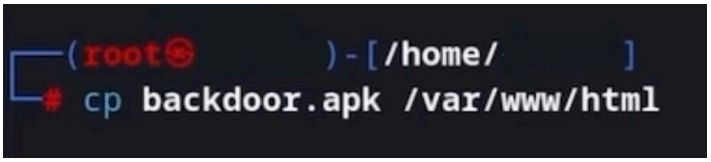
Upon executing the command, we will receive a backdoor.apk file compromising the authentic Instagram apk alongside the embedded malware.

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

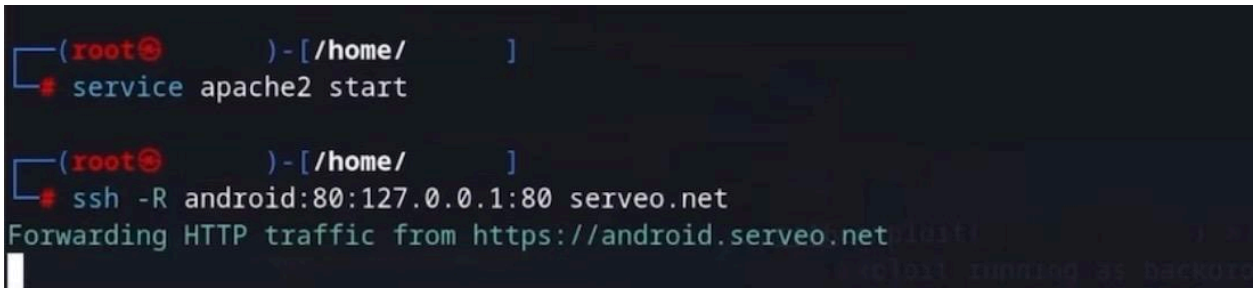


Additionally, it is imperative to establish a dedicated website for facilitating the download process for our target. The HTML file, available on my GitHub repository under the name “index.html”, serves as the cornerstone of this endeavor.

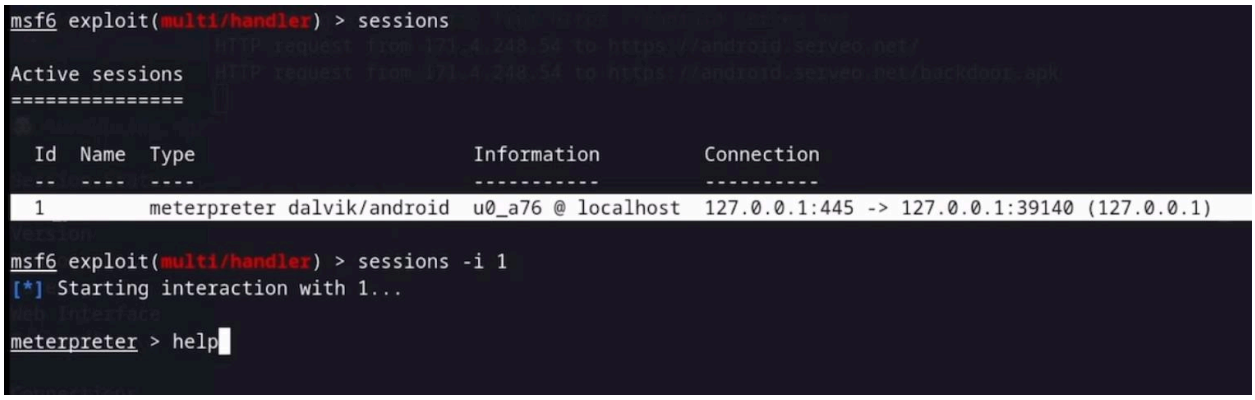
Subsequently, we must ensure the integration of the “backdoor.apk” file into our HTML directory to finalize the implementation process:



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



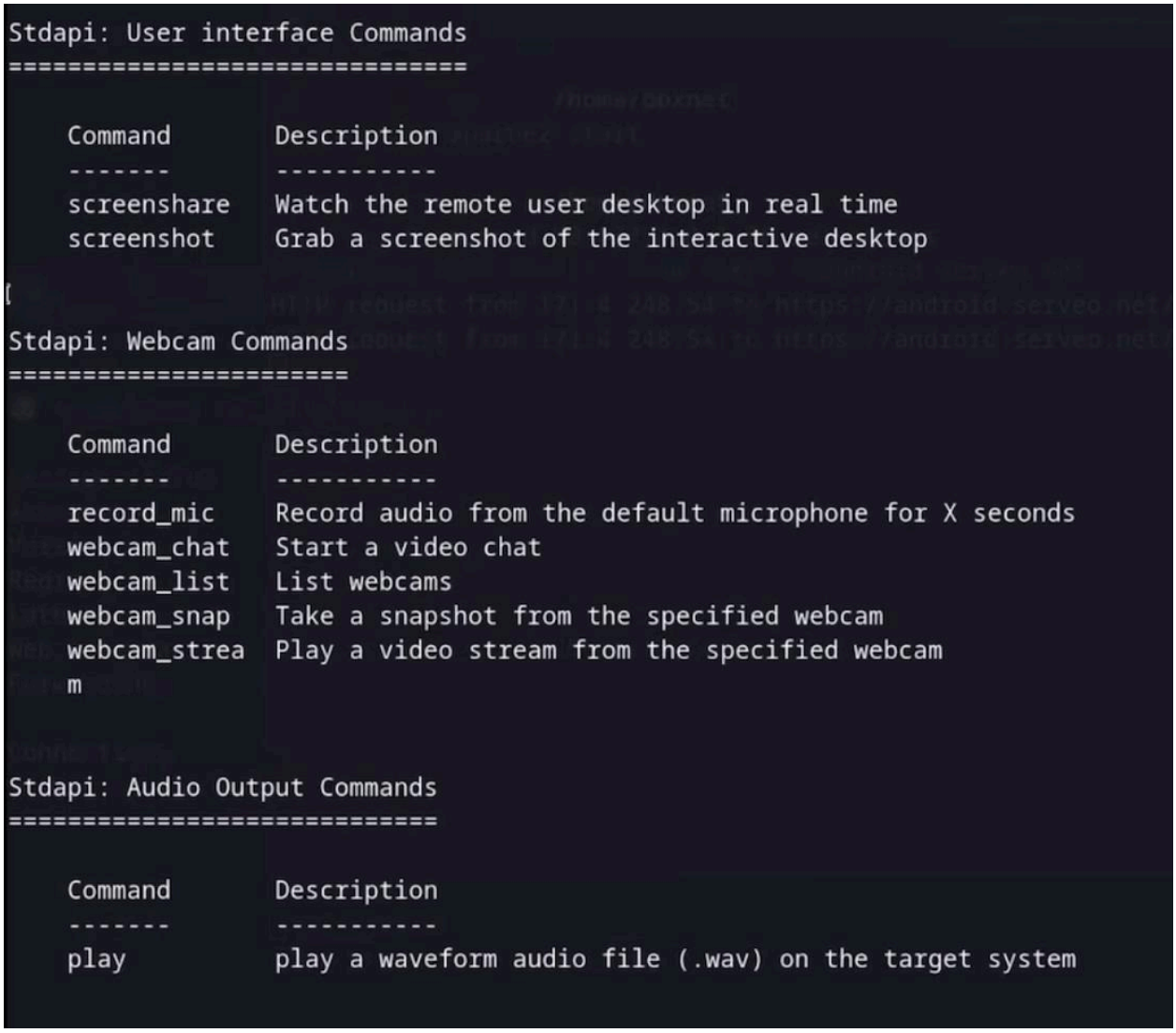
If the victim installs the .apk file, we will be presented with the following outcomes on our listener:



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:




activity_sta Start an Android activity from a Uri string
rt
check_root Check if device is rooted
dump_calllog Get call log
dump_contact Get contacts list
s
dump_sms Get sms messages
geolocate Get current lat-long using geolocation
hide_app_ico Hide the app icon from the launcher
n
interval_col Manage interval collection capabilities
lect
send_sms Sends SMS from target session
set_audio_mo Set Ringer Mode
de
sqlite_query Query a SQLite database from storage
wakelock Enable/Disable Wakelock
wlan_geoloca Get current lat-long using WLAN information
te
Application Controller Commands
Command Description
app_install Request to install apk file
app_list List installed apps in the device
app_run Start Main Activty for package name
app_uninstal Request to uninstall application
l
meterpreter >