

Apktool & Jadx Araçları

Hazırlayan
Ümmü Derya Çelik

APKTOOL ARACI NEDİR ?

Apktool, tersine mühendislik işlemlerinde Android uygulamaların apk dosyalarını decompile ederek dalvik kodlarına, AndroidManifest.xml dosyalarına ulaşmamızı sağlayan araçtır.

Dalvik kodları, java kodlarını kullanarak yazdığımız uygulamanın Android işletim sisteminin bir katmanı olan Dalvik VM yani Dalvik sanal makinesinde çalışabilmesi için derlenmiş halidir.

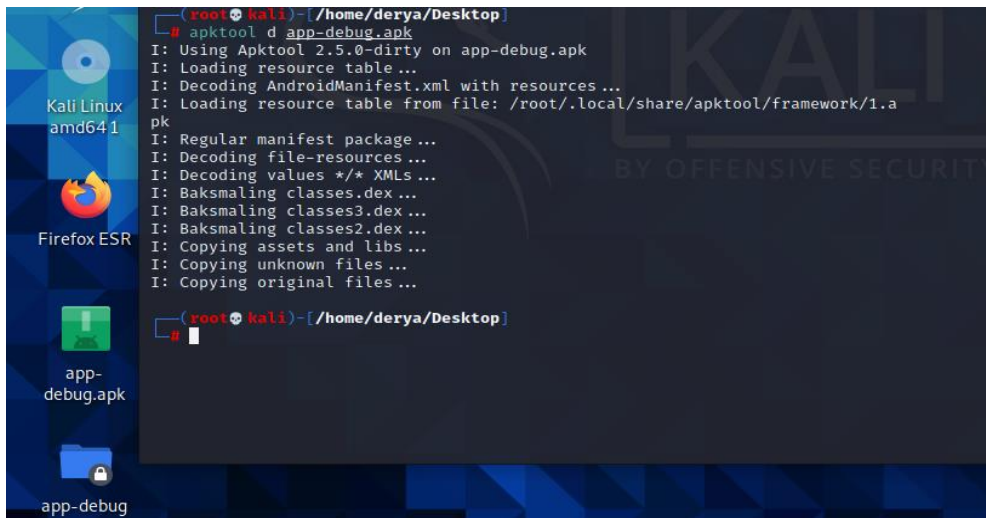
Apktool aracı ile kaynak kod olan java kodlarına birebir ulaşamamaktadır. Fakat Dalvik kodlarından da işe yarar bilgiler elde etmek mümkündür.

Apktool Aracı İle Apk Decompile ve Recompile Nasıl Yapılır?

1.Adım: “apt-get install” komutuyla apktool aracını Kali Linux sanal makinesine yüklemek ile başlıyoruz.

```
[sudo] password for derya:
(root@kali)~[/home/derya]
# apt-get install apktool
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
```

2.Adım: Aracın parametresi olan “d” parametresiyle apk dosyasını decompile ediyoruz. Şekilde de görüldüğü gibi app-debug adlı bir dosya oluştu.



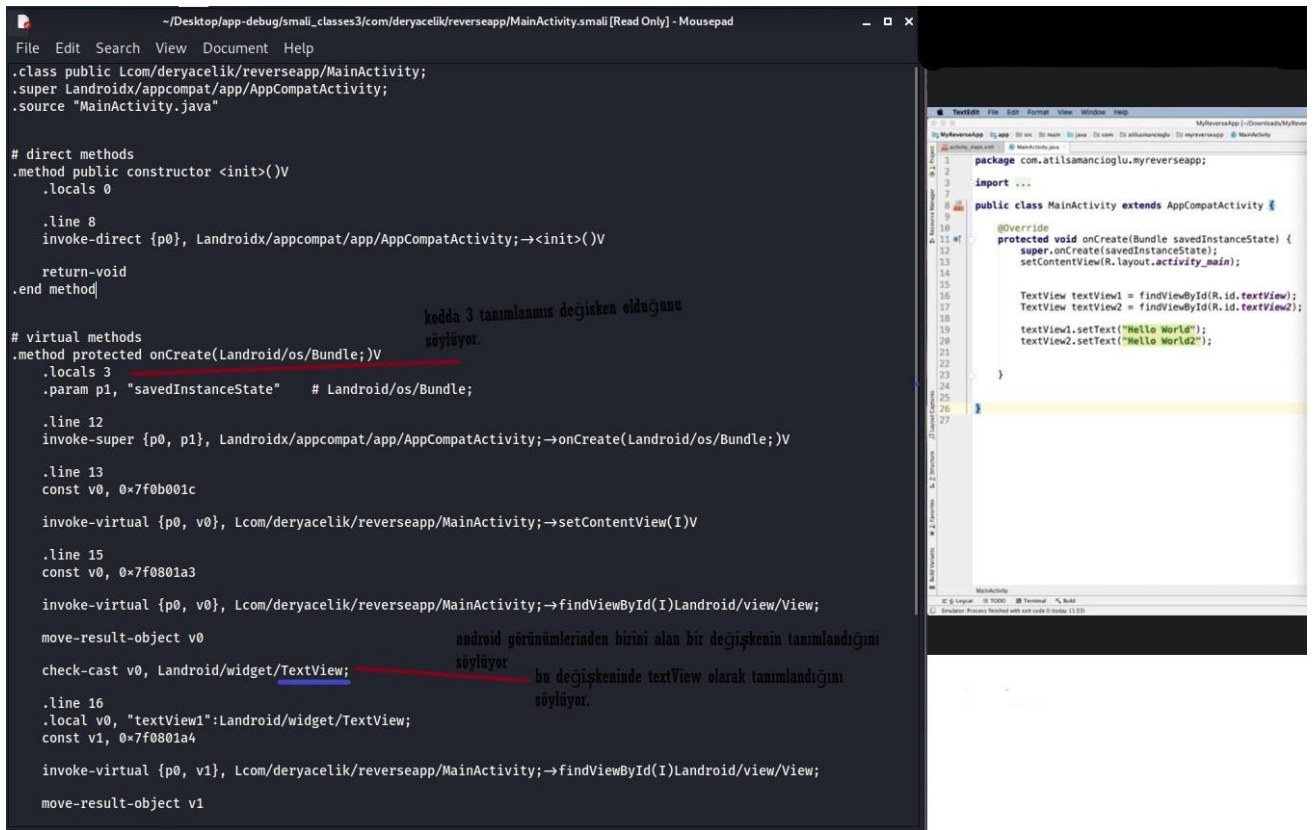
3.Adım: app-debug dosyasında dalvik kodlarını elde ettiğimize göre incelemeye geçebiliriz.

Manifest dosyasına bakarak başlayabiliriz.

```
<?xml version="1.0" encoding="utf-8" standalone="no"?><manifest xmlns:android="http://schemas.android.com/apk/res/android" android:compileSdkVersion="31" android:compileSdkVersionCodename="12" package="com.deryacelik.reverseapp" platformBu
<application android:allowBackup="true" android:appComponentFactory="androidx.core.app.CoreComponentFactory" android:debuggable="true" android:extractNativeLibs="false" android:icon="@mipmap/ic_launcher" android:label="@string/app_name
<activity android:exported="true" android:name="com.deryacelik.reverseapp.MainActivity">
<intent-filter>
<action android:name="android.intent.action.MAIN">
<category android:name="android.intent.category.LAUNCHER">
</intent-filter>
</activity>
</application>
</manifest>
```

Uygulamanın çalıştığı android sürümü, uygulama adı, uygulama sahibi gibi bilgileri görüntüleyebilmekteyiz.

İkinci olarak MainActivity.smali dosyasından uygulamanın kodlarına dalvik kodlarına dönüşmüş şekilde erişebiliriz.



```
File Edit Search View Document Help
.class public Lcom/deryacelik/reverseapp/MainActivity;
.super Landroidx/appcompat/app/AppCompatActivity;
.source "MainActivity.java"

# direct methods
.method public constructor <init>()V
.locals 0

.line 8
invoke-direct {p0}, Landroidx/appcompat/app/AppCompatActivity;-><init>()V
return-void
.end method

# virtual methods
.method protected onCreate(Landroid/os/Bundle;)V
.locals 3
.param p1, "savedInstanceState" # Landroid/os/Bundle;

.line 12
invoke-super {p0, p1}, Landroidx/appcompat/app/AppCompatActivity;->onCreate(Landroid/os/Bundle;)V

.line 13
const v0, 0x7f0b001c

invoke-virtual {p0, v0}, Lcom/deryacelik/reverseapp/MainActivity;->setContentView(I)V

.line 15
const v0, 0x7f0801a3

invoke-virtual {p0, v0}, Lcom/deryacelik/reverseapp/MainActivity;->findViewById(I)Landroid/view/View;
move-result-object v0

check-cast v0, Landroid/widget/TextView;

.line 16
.local v0, "textView1":Landroid/widget/TextView;
const v1, 0x7f0801a4

invoke-virtual {p0, v1}, Lcom/deryacelik/reverseapp/MainActivity;->findViewById(I)Landroid/view/View;
move-result-object v1
```

kodda 3 tanımlanmış değişken olduğu söyler.

android görünümünden birini alan bir değişkenin tanımlandığını söyler.

bu değişkeninde textView olarak tanımlandığını söyler.

Decompile ettiğimiz uygulamanın gerçek kodları ve dalvik kodlarını karşılaştırarak hangi bilgileri elde edeceğimize bakalım.

Görselde de belirtildiği gibi 3 tanımlanan değişkenin olduğunu, textView adında iki tane android görünümü değişkeninin tanımlandığını ve metin içeriğine erişebiliriz.

>> “b” parametresiyle decompile edilmiş uygulama dosyası recompile edilerek yeni apk uzantılı dosya haline getirilebilir.

```
(root@kali)~[~/Desktop]
# apktool b app-debug -o newapk.apk
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
I: Using Apktool 2.5.0-dirty
I: Checking whether sources has changed ...
I: Smaling smali folder into classes.dex ...
I: Checking whether sources has changed ...
I: Smaling smali_classes3 folder into classes3.dex ...
I: Checking whether sources has changed ...
I: Smaling smali_classes2 folder into classes2.dex ...
I: Checking whether resources has changed ...
I: Building resources ...
W: aapt: brut.common.BrutException: brut.common.BrutException: Could not extract resource: /prebuilt/linux/aapt_64 (defaulting to $PATH binary)
W: res/drawable/$avd_hide_password__0.xml: Invalid file name: must contain only [a-z0
```

⋮ Kod manipüle işlemine basit bir örnek verecek olursak bu metin içerikleri değiştirilip kaydedilip recompile edilerek apk dosyası halinde değiştirilmiş hali dağıtılmaya devam edilebilir.

```
invoke-virtual {p0, v1}, Lcom/deryacelik/reverseapp/MainActivity;→findViewById(I)Landroid/view/View;
move-result-object v1

check-cast v1, Landroid/widget/TextView;

.line 18
.local v1, "textView2":Landroid/widget/TextView;
const-string v2, "hello"

invoke-virtual {v0, v2}, Landroid/widget/TextView;→setText(Ljava/lang/CharSequence;)V

.line 19
const-string v2, "hello2"

invoke-virtual {v1, v2}, Landroid/widget/TextView;→setText(Ljava/lang/CharSequence;)V

.line 20
return-void
.end method
```

Görselde de görüldüğü üzere metin içeriğine eriştik.

```
invoke-virtual {p0, v1}, Lcom/deryacelik/reverseapp/MainActivity;→findViewById(I)Landroid/view/View;
move-result-object v1

check-cast v1, Landroid/widget/TextView;

.line 18
.local v1, "textView2":Landroid/widget/TextView;
const-string v2, "hello dalvik"

invoke-virtual {v0, v2}, Landroid/widget/TextView;→setText(Ljava/lang/CharSequence;)V

.line 19
const-string v2, "hello2"
```

Metin içeriği “Hello Dalvik” olarak değiştirildi.

JADX ARACI NEDİR ?

Jadx, Apktool gibi decompile ve recompile işlemi yapabilen fakat Apktool' dan farklı olarak dalvik kodlarında kalmayıp java kaynak kodlarına ulaşabilme imkanı sunan bir araçtır.

Jadx Aracı ile Decompile İşlemi Nasıl Yapılır ?

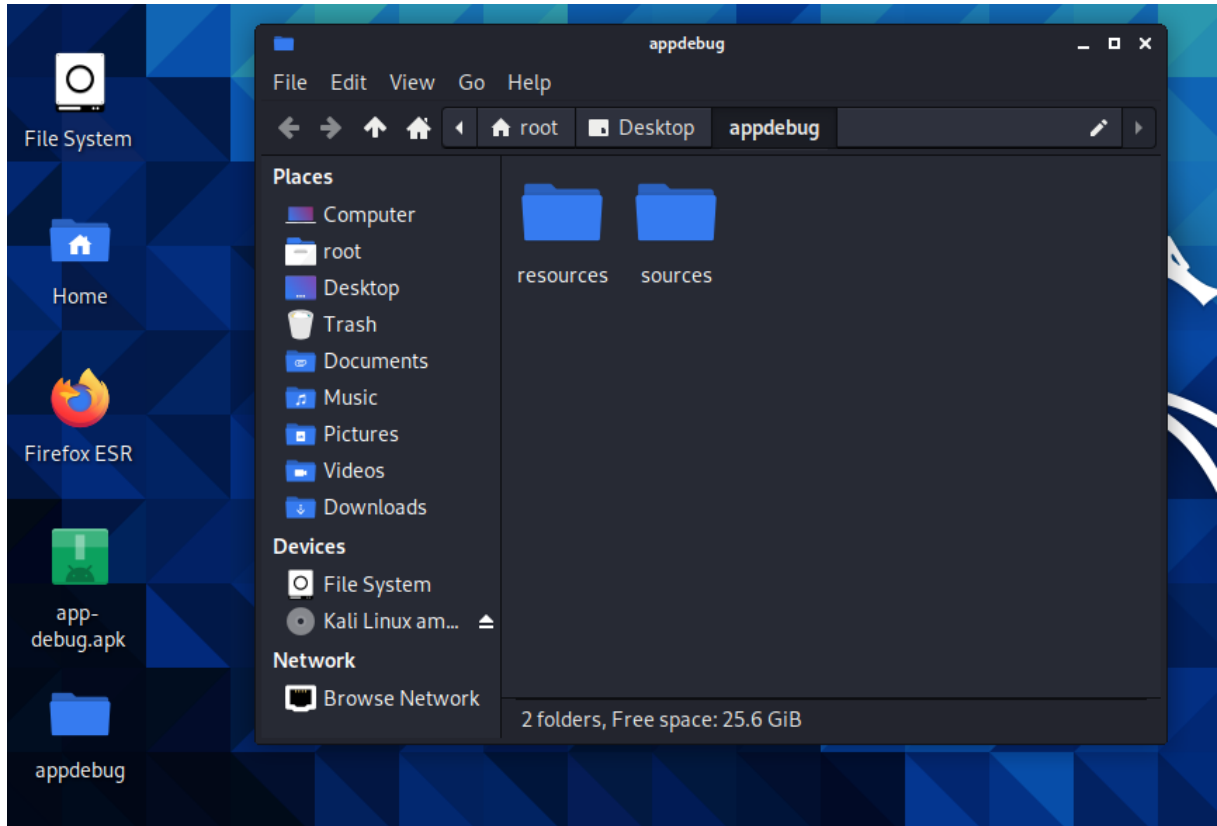
1.Adım: Kali Linux makinemize “apt-get install” komutuyla Jadx aracını yüklüyoruz.

```
(root@kali)-[/home/derya]
# apt-get install jadx
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  jadx
0 upgraded, 1 newly installed, 0 to remove and 1193 not upgraded.
Need to get 15.2 MB of archives.
After this operation, 17.1 MB of additional disk space will be used.
Get:1 http://kali.download/kali kali-rolling/main amd64 jadx all 1.2.0-0kali1
[15.2 MB]
Fetched 15.2 MB in 2s (6,105 kB/s)
Selecting previously unselected package jadx.
(Reading database ... 267597 files and directories currently installed.)
Preparing to unpack .../jadx_1.2.0-0kali1_all.deb ...
Unpacking jadx (1.2.0-0kali1) ...
Setting up jadx (1.2.0-0kali1) ...
Processing triggers for kali-menu (2021.1.4) ...
```

2.Adım: Apktool aracında da olduğu gibi “d” parametresiyle apk dosyamızı decompile ediyoruz.

```
(root@kali)-[~/Desktop]
# jadx -d appdebug app-debug.apk
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
INFO - loading ...
INFO - processing ...
INFO - done
```

3.Adım: Adını bizim belirlediğimiz decompile edilmiş “appdebug” dosyasını incelemeye geçebiliriz.



```
appdebug >> sources >> com >> deryacelik >> reverseapp >> MainActivity.java
```

Belirtilen yolu takip ettikten sonra MainActivity yani kaynak kodların bulunduğu java kodlarına erişmiş bulunmaktayız.

