


Paper On	Android Application Reversing Via Android Mobile
Author	Nieo
Email	Nieo@Hackermail.Com
Website	www.Uret.in
Date	10/12/2014 (dd/mm/yyyy)
Level	🌟🌟🌟🌟🌟
Language	English
Country	India 

Content

1. Introduction
2. Tools & Download
3. How to Use Tools
4. Tuts

Introduction

Understanding APK

What is Apk file?

Android application package file (APK) is the file format used to distribute and install application software and middleware onto Google's Android operating system. APK files are ZIP file formatted packages based on the JAR file format with .apk file extensions

What are the Contents you find inside Apk?

An APK file is an archive that usually contains the following directories:

- META-INF directory:
 - MANIFEST.MF: The Manifest File
 - CERT.RSA: The certificate of the application
 - CERT.SF: The list of resources and SHA-1 digest

Corresponding lines in the MANIFEST.MF file; for example:

```
Signature-Version: 1.0
Created-By: 1.0 (Android)
SHA1-Digest-Manifest: wxqnEAI0UA5nO5QJ8CGMwjKGGWE=
Name: res/layout/exchange_component_back_bottom.xml
SHA1-Digest: eACjMjESj7Zkf0cBFTZOnqWrt7w=
Name: res/drawable-hdpi/icon.png
SHA1-Digest: DGEqyIP8W0n0iV/ZzBx3MW0WGCA=
```

- `lib`: the directory containing the compiled code that is specific to a software layer of a processor, the directory is split into more directories within it:
 - `armeabi`: compiled code for all ARM based processors only
 - `armeabi-v7a`: compiled code for all ARMv7 and above based processors only
 - `x86`: compiled code for x86 processors only
 - `mips`: compiled code for MIPS processors only
- `classes.dex`: The classes compiled in the dex file format understandable by the Dalvik virtual machine
- `res`: The directory containing resources not compiled into `resources.arsc` (see below).
- `assets`: a directory containing applications assets, which can be retrieved by `AssetManager`.
- `AndroidManifest.xml`: An additional Android manifest file, describing the name, version, access rights, referenced library files for the application. This file may be in Android binary XML that can be converted into human-readable plain text XML with tools such as `AXMLPrinter2`, `android-apktool`, or `Androguard`
- `resources.arsc`: A file containing precompiled resources, such as binary XML for example.

Tools & Download

Name	Download
Apktool Mobile Ver-4.6	Click
APK Editor Ver.1.90	Click
Axel Ver.2.4	Click
920 Text Editor Ver.13.7.18	Click
aGrep Ver.2014.03	Click

How to Use Tool's

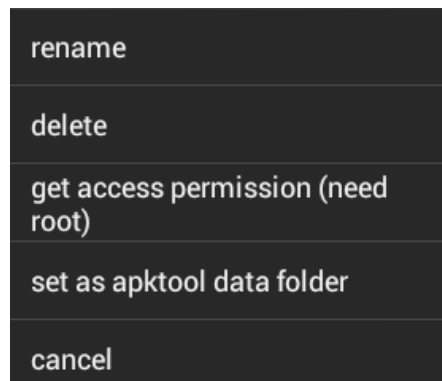
1. Apktool Mobile

What is Apk tool?

This tool is basically we use for Decompile & Recompile Apk file

How to Setup this Tool?

- Unzip Apktool4.6_armhf.zip to your SD Card and install Apktool.apk inside
- Once installation done start app then goto directory where Apktool4.6_armhf.zip Unzipped & rename folder to "Apktool"
- Now select that Unzipped Apktool folder and hold your finger until it pop-up following options



- Now select "Set as apktool data folder" option
- There are more option which you find inside setting option you can choose option which is more suitable for you

Option which I pref: -

- Vibration
- Notification
- Wrap output Message
- File Sort Rule
- Aapt - 4.4 (Android Asset Packaging Tool)
- Apk tool version – 2.0

How to Use this Tool?

For this you need to know what functions it offer like-

Decompile functions

- Decompile all (This will decompile Dex + Resources)
- Decompile Dex (This will decompile Dex in to Smali Codes)
- Decompile Resources (This will decompile layout ie AXML Binary format to XML)

Other functions

- Zipalign (This function align all files inside apk)
- Sign (This will sign apk after modification)
- Create odex
- Add/Extract/Delete (META-INF)
- Dex2Jar (This will convert Dex in to Java class file's)
- Jar2Dex (This will recompile Jar in to Dex file's)

- Install (This will help to install app)
- Cancel

Recompile functions

- Recompile (This will Re-Compile Source folder)

(Note:- Recompile function pop-up only when you click folder which end with “_src” for Eg:- apk_src i.e application decompile folder)

2. **APK Editor**

This tool have very good features like-

- Editing Text inside AXML file & Dex directly
- Replace moded file with original file inside APK directly
- Cloning APK so that you can use same application on single device
- Optimize & Sign

We are going to use this app as resource editor for editing text strings

3. **Axel**

This is very good app to open AXML Binary format in to human readable format or modify decompile XML format which we come across at the time of editing various application layout inside resource folder

4. **920 Text Editor**

We use this application to edit & save our Decompile Smali Codes

5. **aGrep**

We use this application to find the strings in Decompile Smali Codes folder

Tuts

What are the type of Restriction & Protection?

- Advertisement Banners (Google or ad-mob)
- Nag & Msg Box Pop-Up (In Trial or Free App)
- Locked Features (In the form of In-App Purchase)
- Google or Amazon License (In the form of License Verification Library-LVL)

Before we start with Tut's Download all Target Files from here :- [Click](#)

Tut -1

Removing of Advertisement Banners from application

TARGET	Dual File Manager
METHOD	Ad Removing by Editing Layout
DIFFICULTY	(X) Newbies () Intermediate () Advanced () Master

Step-1

Start Apktool & Select “Decompile Resources” Option

Step-2

Once Decompile Process finish go to “res” a resources folder which you found inside decompile folder of app

Eg:-Dual File Manager_src

Step-3

Now search for Layout Folder & open every XML file with the help of "Axel" app to find String called "Banner"

(Hint :- In most of the case Banner called through "main.xml" or "Activity_main.xml")

Step-4

Finding & Editing XML Codes with "Axel" app

As I say in this app also we find Banner related XML code in "main.xml"

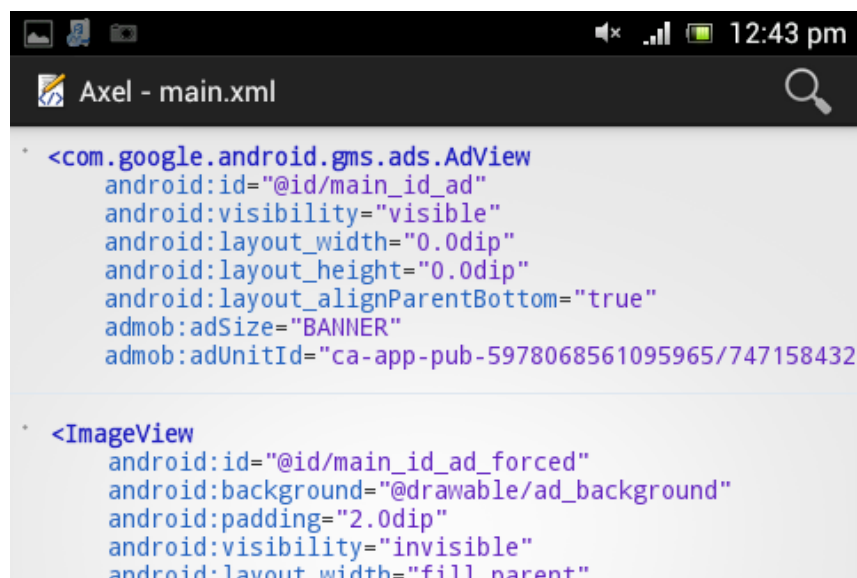
Original Code: -

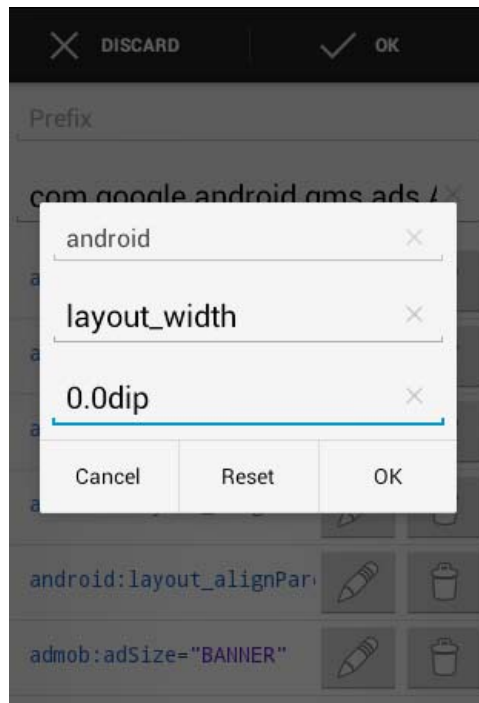
```
<com.google.android.gms.ads.AdView
android:id="@id/main_id_ad"
android:visibility="visible"
android:layout_width="fill_parent"-----> Set this to "0dip"
android:layout_height="wrap_content"-----> Set this to "0dip"
android:layout_alignParentBottom="true"
admob:adSize="BANNER"
admob:adUnitId="ca-app-pub-5978068561095965/7471584327" />
```

Modify Code: -

```
<com.google.android.gms.ads.AdView
android:id="@id/main_id_ad"
android:visibility="visible"
android:layout_width="0dip"
android:layout_height="0dip"
android:layout_alignParentBottom="true"
admob:adSize="BANNER"
admob:adUnitId="ca-app-pub-5978068561095965/7471584327" />
```

So just tap this code after opening "main.xml" file with "Axel app" then edit relevant field as marked with RED in above & Save xml





Step-5

Recompile Source folder with APK Tool

Step-6

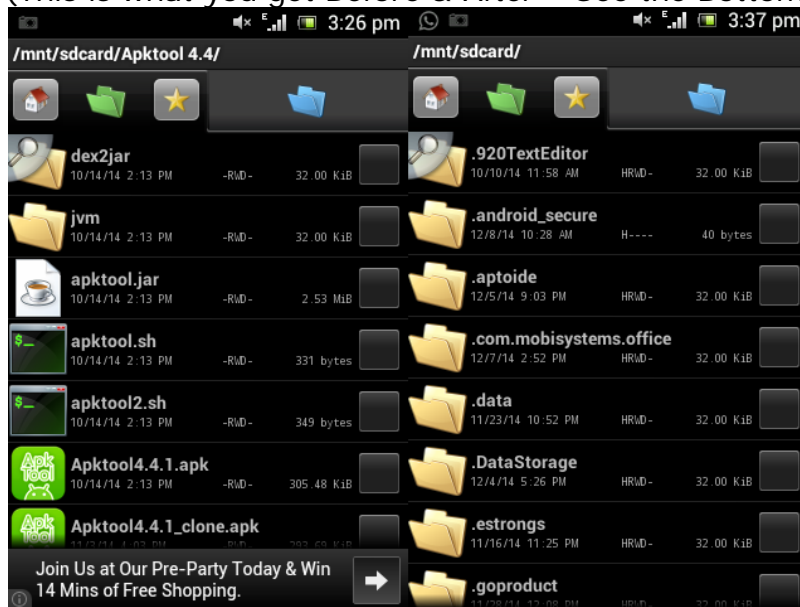
Now Zipalign apk with APK Tool

Step-7

Sign apk with APK Tool

(I pref "APK editor" app for Zipalign & Signing Bcz its fast. To do this you have to select "Optimize & Sign" Option after step-5)

(This is what you get Before & After – See the Bottom)



Important things for Reference Only

What is dip?

Density-independent Pixels - an abstract unit that is based on the physical density of the screen.

Related example: -

- px (Pixels)
- in (inches)
- mm (Millimeters)
- pt (Points)
- sp (Scale Independent Pixels)

What are the Method through which we can remove Banner ad's?

- By editing Layout
- By editing code in dex

Here is List of Known Ads to identify them easily –

- .gstatic.com
- .admob.com
- .analytics.localytics.com
- .flurry.com
- .greystripe.com
- inmobi.com
- admax.nexage.com
- ads.mdotm.com
- my.mobfox.com
- .plus1.wapstart.ru
- .madnet.ru
- .mp.mydas.mobi
- millennialmedia.com
- .g.doubleclick.net
- .appsdt.com
- ad.leadboltads.net
- run.admost.com
- mobile.admost.com

Now before we start with our second Tut let's go through few basic Dalvik Op-Codes which help us in to understand our decompile smali code's & reversing of our target's

Table: -

CODE		EXPLANATION
nop		No operation
const/4	vx,lit4	Puts the 4 bit constant into vx
const/16	vx,lit16	Puts the 16 bit constant into vx
const-string	vx,string_id	Puts reference to a string constant identified by string id into vx.
Goto		Unconditional jump by short offset
If-eqz	vx,target	Jumps to target if vx=0 vx is an integer value
If-nez	vx,target	Checks vx and jumps if vx is nonzero.
If-lez	vx,target	Checks vx and jumps if vx<=0
xor-int	vx, vy, vz	Calculates vy XOR vz and puts the result into vx.
add-int	vx,vy,vz	Calculates vy+vz and puts the result into vx.
sub-int	vx,vy,vz	Calculates vy-vz and puts the result into vx.
mul-int	vx, vy, vz	Multiplies vz with vy and puts the result int vx.
div-int	vx,vy,vz	Divides vy with vz and puts the result into vx.
return-void		Return without a return value

NOTE: - vx,vv,vy,vz are integer vale to explain the use of code

I just cover here only important code that we are going to use while reversing & understanding the target for more reference of Dalvik Op-Code you can go to here: -

http://pallergabor.uw.hu/androidblog/dalvik_opcodes.html

Tut -2

Removing Nag & Msg Box Pop-Up

TARGET	Medical Calculators
METHOD	Reversing Smali Codes
DIFFICULTY	() Newbies (X) Intermediate () Advanced () Master

Step-1

Note down & understand all restriction

- Skip Verification Button visible only for 7day's
- 7 day's Trial End Pop-Ups

So let's begin with removing first restriction ie after expiry of 7 day's "**Skip Login Verification**" Button will be removed from Screen so that we can't access app offline until we login



Step-2

Start with decompile apk & once it's done then go to smali\Pedcall\Calculator folder

Now we need to find the code which controls our Skip Login Verification Button visibility in such manner that after expiry of 7 day's it will be removed from screen

But the question is how we can find that code in a bunch of smali files?

Let's apply some logic, we know that this button appears on screen with login option & there we see some login-related smali. So just open & search for "Skip" in one after another login-related smali. Alternatively, you can use "aGrep" app for this, which we discussed next here.

And their You will find following code in "login.smali"

Code View: -

```
.line 413
.local v11, btn_skip:Landroid/widget/Button; --(See this is our Skip Verification button)
if-nez v23, :cond_0 --- (A condition which control visibility of our Button)

.line 414
const/16 v28, 0x8

move/from16 v0, v28

invoke-virtual {v11, v0}, Landroid/widget/Button; ->setVisibility(I)V--(Here it remove our Button)

.line 417
:cond_0
new-instance v28, LPedcall/Calculator/login$6;

move-object/from16 v0, v28

move-object/from16 v1, p0

invoke-direct {v0, v1}, LPedcall/Calculator/login$6; -><init>(LPedcall/Calculator/login;)V

move-object/from16 v0, v28

invoke-virtual {v11, v0}, Landroid/widget/Button; -
>setOnClickListener(Landroid/view/View$OnClickListener;)V--(On Click Button action listener)
```

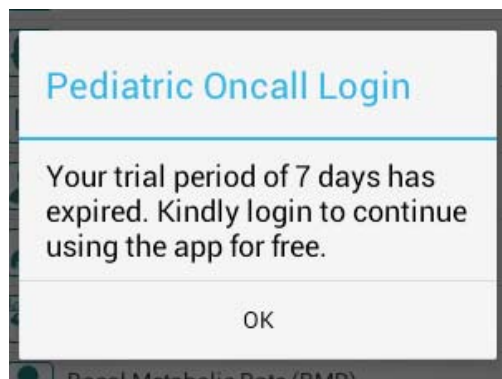
So we just need to change -

if-nez v23, :cond_0 to goto :cond_0

in order to make our Skip Verification button always visible even after expiry of 7 day'

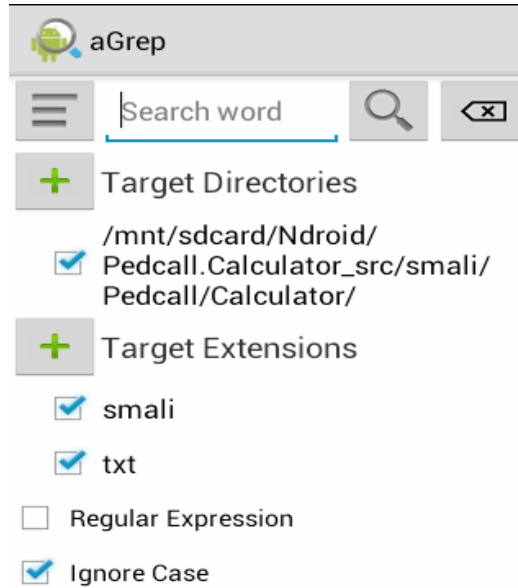
Step-3

Now we are going to remove 7 day's trial expired Popup's

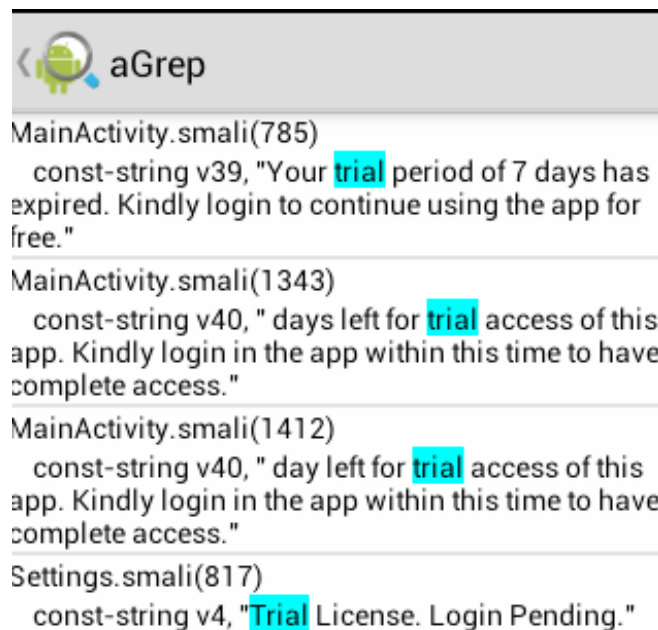


So let's search for the smali file which contain sting "Your trial period of 7 days has expired. Kindly login to continue using the app for free."

To do this we just fire up "aGrep" app & search for it



Result of Our Search you get in to this window



So now we know where we are going to find this string ie. inside "MainActivity.smali"

Code View: -

```
:goto_1
  invoke-virtual {v14, v15}, Ljava/util/Date; ->before(Ljava/util/Date;)Z

  move-result v38

  if-eqz v38, :cond_6-----(Conditional Jump just before Trial Msg so set this to goto :cond_6)

  .line 192
  new-instance v38, Landroid/app/AlertDialog$Builder; --(If we don't jump then we land to Msg Box)

  move-object/from16 v0, v38

  move-object/from16 v1, p0

  invoke-direct {v0, v1}, Landroid/app/AlertDialog$Builder; -><init>(Landroid/content/Context;)V

  .line 193
  const-string v39, "Pediatric Oncall Login" ---(Caption Text of PopUp Msg)

  invoke-virtual/range {v38 .. v39}, Landroid/app/AlertDialog$Builder; -
  >setTitle(Ljava/lang/CharSequence;)Landroid/app/AlertDialog$Builder;

  move-result-object v38

  .line 194
  const-string v39, "Your trial period of 7 days has expired. Kindly login to continue using the app for free."
```

Our conditional jump will land us at

Code View: -

```
:cond_6
  invoke-virtual/range {p0 .. p0}, LPedcall/Calculator/MainActivity; -
  >getApplicationContext()Landroid/content/Context;

  move-result-object v4

  check-cast v4, LPedcall/Calculator/AppAnalytics;

  .line 206
  .local v4, appState:LPedcall/Calculator/AppAnalytics;
  invoke-virtual {v4}, LPedcall/Calculator/AppAnalytics; ->getwarn()Z --(Uhh! Warning)

  move-result v37

  .line 208
  .local v37, warn:Z ----(One more warning Msg goging to start if we continue in this code )
  if-nez v37, :cond_0 --(So we required to Jump one more time so set condition to goto :goto_2)
```

Now why goto_2 & not cond_0 ?

Because If you see above code in continuation then you will going to find that there is two more condition (cond_7 & cond_8) between day's & time calculation and both of this condition used to generate alert dialog for Number of Trial Day's left.

Condition-7 Code View which lead us to Trial Msg

```
:cond_7
new-instance v38, Landroid/app/AlertDialog$Builder;

move-object/from16 v0, v38

move-object/from16 v1, p0

invoke-direct {v0, v1}, Landroid/app/AlertDialog$Builder; -> <init>(Landroid/content/Context;)V

.line 224
const-string v39, "Pediatric Oncall Login"

invoke-virtual/range {v38 .. v39}, Landroid/app/AlertDialog$Builder; -
>setTitle(Ljava/lang/CharSequence;)Landroid/app/AlertDialog$Builder;

move-result-object v38

.line 225
new-instance v39, Ljava/lang/StringBuilder;

invoke-static/range {v31 .. v31}, Ljava/lang/String; ->valueOf(I)Ljava/lang/String;

move-result-object v40

invoke-direct/range {v39 .. v40}, Ljava/lang/StringBuilder; -> <init>(Ljava/lang/String;)V

const-string v40, " days left for trial access of this app. Kindly login in the app within this time to
have complete access." --- (Trial Day's remaining Msg)
```

Condition-8 Code View which lead us to Trial Msg

```
:cond_8
new-instance v38, Landroid/app/AlertDialog$Builder;

move-object/from16 v0, v38

move-object/from16 v1, p0

invoke-direct {v0, v1}, Landroid/app/AlertDialog$Builder; -> <init>(Landroid/content/Context;)V

.line 233
const-string v39, "Pediatric Oncall Login"

invoke-virtual/range {v38 .. v39}, Landroid/app/AlertDialog$Builder; -
>setTitle(Ljava/lang/CharSequence;)Landroid/app/AlertDialog$Builder;

move-result-object v38

.line 234
new-instance v39, Ljava/lang/StringBuilder;

invoke-static/range {v16 .. v17}, Ljava/lang/String; ->valueOf(J)Ljava/lang/String;

move-result-object v40

invoke-direct/range {v39 .. v40}, Ljava/lang/StringBuilder; -> <init>(Ljava/lang/String;)V

const-string v40, " day left for trial access of this app. Kindly login in the app within this time to
have complete access." --- (Trial Day's remaining Msg)
```


Since we don't want any more popup we don't use this two condition but end method of both this condition is important for us ie both of this condition lead us to goto_2

```
:goto_6
    const/16 v38, 0x1

    move/from16 v0, v38

    invoke-virtual {v4, v0}, LPedcall/Calculator/AppAnaylitics; ->setWarn(Z)V

    goto/16 :goto_2 -----(This is where cond_7 & cond_8 lead us)
```

So we just ignore cond_0 and go with the flow of code & here we land & after this point everything is going good without any annoying popup

```
:goto_2
    sget v38, Landroid/os/Build$VERSION; ->SDK_INT:I

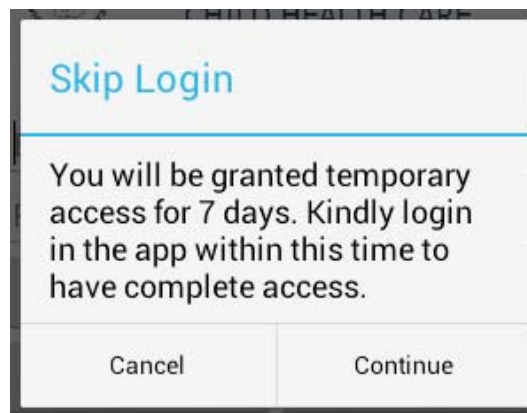
    const/16 v39, 0xb

    move/from16 v0, v38

    move/from16 v1, v39

    if-lt v0, v1, :cond_b
```

Now just last popup remain which we are going to edit & not remove



We repeat the process of searching the string & we find above strings inside "login\$6.smali"

```
# virtual methods
.method public onClick(Landroid/view/View;)V
    .locals 3
    .parameter "v"

    .prologue
    .line 423
    new-instance v0, Landroid/app/AlertDialog$Builder;

    iget-object v1, p0, LPedcall/Calculator/login$6; -> this$0: LPedcall/Calculator/login;

    invoke-direct {v0, v1}, Landroid/app/AlertDialog$Builder; -> <init>(Landroid/content/Context;)V

    .line 426
    .local v0, alertDialog: Landroid/app/AlertDialog$Builder;
    const-string v1, "Skip Login"

    invoke-virtual {v0, v1}, Landroid/app/AlertDialog$Builder; -
> setTitle(Ljava/lang/CharSequence;) Landroid/app/AlertDialog$Builder;

    .line 429
    const-string v1, "You will be granted temporary access for 7 days. Kindly login in the app within this
time to have complete access." -----(Just Edit this to Unlimited License Version)

    invoke-virtual {v0, v1}, Landroid/app/AlertDialog$Builder; -
> setMessage(Ljava/lang/CharSequence;) Landroid/app/AlertDialog$Builder;

    .line 434
    const-string v1, "Continue"
```

Setp-4

Now all done & go for Recompile, Zipalign & Sign the apk.

Tut -3

Unlocking Locked Features

TARGET	Sensor Alarm
METHOD	Reversing Smali Codes & Editing Resources
DIFFICULTY	() Newbies () Intermediate (X) Advanced () Master

Step-1

Note Down Locked features & Nag Msg Popup at the time of click

Locked Features: -

- Camera
- Temperature
- Phone Number –SMS
- Phone Number –Call

Nag Msg: -



Step-2

Start Apktool & Select "Decompile All" Option

Step-3

Once Decompile Process finish go to following folder
Sensor Alarm Free_src\smali\com\oceanicsoftware\sensoralarm_free

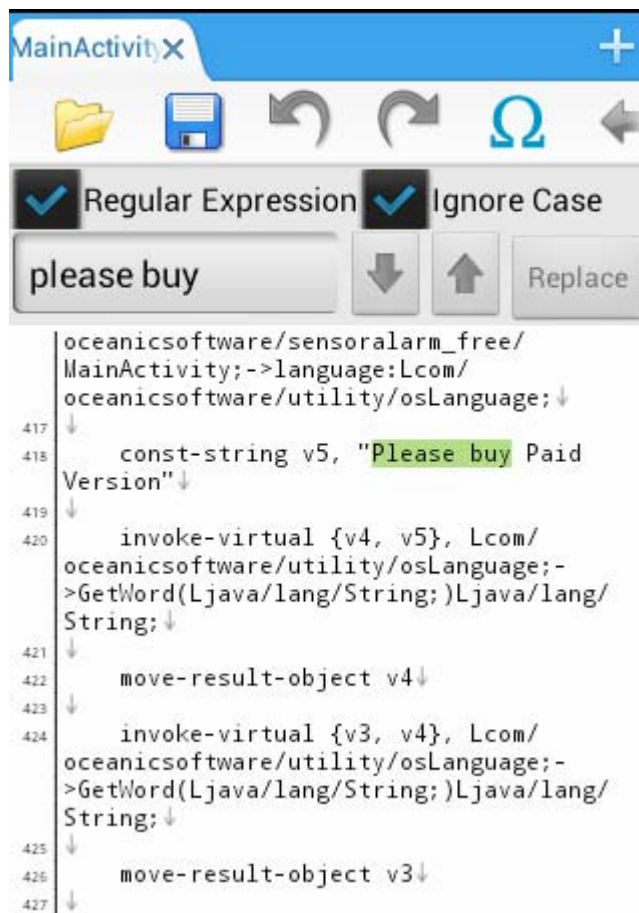
This is main folder where we can find our all relevant codes to reverse

Now search for the string "Please buy Paid Version" with the help of "aGrep" app

You find this string inside "MainActivity.smali"

Step-4

Now go to the first set of Alert msg code in "MainActivity.smali". To open, find & Edit this we use our next app called "920 Text Editor"



Code View:-

```
.line 540 ----- (Here all it's Start)
new-instance v3, Landroid/app/AlertDialog$Builder; ---- (Method Create Alert Dialog)

    invoke-direct {v3, p0}, Landroid/app/AlertDialog$Builder; -
><init>(Landroid/content/Context;)V

    invoke-virtual {v3}, Landroid/app/AlertDialog$Builder; -
>create()Landroid/app/AlertDialog;

    move-result-object v2

    .line 541
    .local v2, paidAlert:Landroid/app/AlertDialog;
    sget-object v3, Lcom/oceanicsoftware/sensoralarm_free/MainActivity; -
>language:Lcom/oceanicsoftware/utility/osLanguage;

    const-string v4, "Warning" ←----- (Our Title String)

    invoke-virtual {v3, v4}, Lcom/oceanicsoftware/utility/osLanguage; -
>GetWord(Ljava/lang/String;)Ljava/lang/String;

    move-result-object v3

    invoke-virtual {v2, v3}, Landroid/app/AlertDialog; -
>setTitle(Ljava/lang/CharSequence;)V ←----- (Title Set here)

    .line 542
    sget-object v3, Lcom/oceanicsoftware/sensoralarm_free/MainActivity; -
>language:Lcom/oceanicsoftware/utility/osLanguage;

    sget-object v4, Lcom/oceanicsoftware/sensoralarm_free/MainActivity; -
>language:Lcom/oceanicsoftware/utility/osLanguage;

    const-string v5, "Please buy Paid Version" ----- (The Bad Boy Msg)

    invoke-virtual {v4, v5}, Lcom/oceanicsoftware/utility/osLanguage; -
>GetWord(Ljava/lang/String;)Ljava/lang/String;

    move-result-object v4

    invoke-virtual {v3, v4}, Lcom/oceanicsoftware/utility/osLanguage; -
>GetWord(Ljava/lang/String;)Ljava/lang/String;

    move-result-object v3

    invoke-virtual {v2, v3}, Landroid/app/AlertDialog; -
>setMessage(Ljava/lang/CharSequence;)V

    .line 543
    sget-object v3, Lcom/oceanicsoftware/sensoralarm_free/MainActivity; -
>language:Lcom/oceanicsoftware/utility/osLanguage;

    const-string v4, "Buy"
```

Now we know where our Bad Boy Msg created & how it will be called so now next we do is go to the start of whole process which start at here

Code View: -

```
# virtual methods
.method CreateAlarm(Ljava/lang/String;)V
    .locals 12
    .parameter "message"

    .prologue
    const/4 v11, 0x5

    const/4 v10, 0x2

    const/4 v9, 0x0

    const/4 v8, 0x1

    .line 532
    sget-boolean v3, Lcom/oceanicsoftware/sensoralarm_free/MainActivity; -
    >FREE_APP_FLAG:Z ----- (Here it set the flag via Boolean)

    if-eqz v3, :cond_1 ----- (Condition if v3 equal to 0 then jump to cond_1)

    .line 534
    iget v3, p0, Lcom/oceanicsoftware/sensoralarm_free/MainActivity; -
    >freeAlarmSize:I

    add-int/lit8 v3, v3, 0x1

    iput v3, p0, Lcom/oceanicsoftware/sensoralarm_free/MainActivity; -
    >freeAlarmSize:I

    .line 536
    iget v3, p0, Lcom/oceanicsoftware/sensoralarm_free/MainActivity; -
    >freeAlarmSize:I

    iget v4, p0, Lcom/oceanicsoftware/sensoralarm_free/MainActivity; -
    >FREE_EVENT_SIZE:I

    if-le v3, v4, :cond_1

    .line 538
    iput v9, p0, Lcom/oceanicsoftware/sensoralarm_free/MainActivity; -
    >freeAlarmSize:I

    .line 540 ----- (Point where popup Start in the form of Alert)
    new-instance v3, Landroid/app/AlertDialog$Builder;

    invoke-direct {v3, p0}, Landroid/app/AlertDialog$Builder; -
    ><init>(Landroid/content/Context;)V

    invoke-virtual {v3}, Landroid/app/AlertDialog$Builder; -
    >create()Landroid/app/AlertDialog;

    move-result-object v2
```

Step-5

In order to avoid Nag Msg we need to break the free version code routine which we can do with the help of reversing conditional Jump code which we find at ".line 532"

Original Code: -

```
# virtual methods
.method CreateAlarm(Ljava/lang/String;)V
    .locals 12
    .parameter "message"

    .prologue
    const/4 v11, 0x5

    const/4 v10, 0x2

    const/4 v9, 0x0

    const/4 v8, 0x1

    .line 532
    sget-boolean v3, Lcom/oceanicsoftware/sensoralarm_free/MainActivity; -
    >FREE_APP_FLAG:Z

    if-eqz v3, :cond_1 ----- (Now set "if-eqz" To "goto" )

    .line 534
    iget v3, p0, Lcom/oceanicsoftware/sensoralarm_free/MainActivity; ->freeAlarmSize: I
```

Modify Code: -

```
# virtual methods
.method CreateAlarm(Ljava/lang/String;)V
    .locals 12
    .parameter "message"

    .prologue
    const/4 v11, 0x5

    const/4 v10, 0x2

    const/4 v9, 0x0

    const/4 v8, 0x1

    .line 532
    sget-boolean v3, Lcom/oceanicsoftware/sensoralarm_free/MainActivity; -
    >FREE_APP_FLAG:Z

    goto, :cond_1 (Hence from Conditional Jump "if-eqz" To Unconditional jump "Goto" )

    .line 534
    iget v3, p0, Lcom/oceanicsoftware/sensoralarm_free/MainActivity; ->freeAlarmSize: I
```


Step-6

Now you have to repeat Step-6 three more time in order to fix three more nag msg in similar file at

```
.line 1139
  sget-boolean v1, Lcom/oceanicsoftware/sensoralarm_free/MainActivity; ->FREE_APP_FLAG:Z

  if-eqz v1, :cond_6 ---- ( Change it To goto, :cond_6)
```

```
.line 703
  :cond_0
  sget-boolean v6, Lcom/oceanicsoftware/sensoralarm_free/MainActivity; ->FREE_APP_FLAG:Z

  if-eqz v6, :cond_2 ---- ( Change it To goto, :cond_2)
```

```
.line 954
  .local v0, btnSignal:Landroid/widget/ImageButton;
  sget-boolean v3, Lcom/oceanicsoftware/sensoralarm_free/MainActivity; ->FREE_APP_FLAG:Z

  if-eqz v3, :cond_1 ---- ( Change it To goto, :cond_1)
```

Now save & go to next .smali file's & continue our search for same sting & we find one more msg in "MainActivity\$11\$1.smali" file at

```
.line 897
  .local v1, inSecond:I
  :try_start_0
  sget-boolean v3, Lcom/oceanicsoftware/sensoralarm_free/MainActivity; ->FREE_APP_FLAG:Z

  if-eqz v3, :cond_0 ---- ( Change it To goto, :cond_0)
```

Aanh! Hell there is One more msg to go inside "MainActivity\$17.smali"

```
.line 1227
  sget-boolean v1, Lcom/oceanicsoftware/sensoralarm_free/MainActivity; ->FREE_APP_FLAG:Z

  if-eqz v1, :cond_2 ---- ( Change it To goto, :cond_2)
```

If you dig in little more inside code ie condition were we set our jump, then you will notice that we are lucky because all our jumps to the condition's are working codes for the relevant button action. It means no dead or junk code & this app is not Demo version.

How to know whether trial or free app is having only Demo Codes?

You can find this by two basic way: -

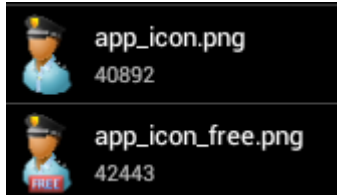
- There is no conditional jump in such case before start of Nag Msg code
- If it has conditional jump then its land over Error Msg i.e for no action & return void

Step-7

Now lets modify some of resources & give Pro version look to this app

Start with changing Launcher icon & this we can do simply by: -

Go to res\drawable-mdpi folder & inter change the name of "app_icon_free.png" & "app_icon.png"



(Note: - This will not be the case in all app many Trial & Free app not include full version component)

Now we going to change the name of app from "Sensor Alarm Free" to "Sensor Alarm" & this we can do by :-

Go to res\values then open "strings.xml" & edit as follows

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="app_name">Sensor Alarm Free</string> --- (Change this to Sensor Alarm)
  <string name="hello_world">Hello world!</string>
  <string name="menu_settings">Settings</string>
  <string name="title_activity_main">MainActivity</string>
  <string name="hello">Hello World, SensorAlarmActivity!</string>
</resources>
```

Step-8

Now all done & go for Recompile, Zipalign & Sign the apk.

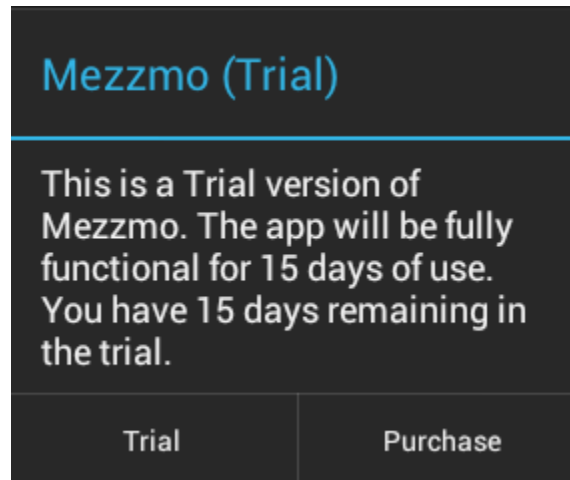
Tut -4

Reversing Google or Amazon License

TARGET	Mezzmo
METHOD	Reversing Smali Codes & Editing Resources
DIFFICULTY	() Newbies () Intermediate () Advanced (X) Master

Step-1

First we identify what are the restriction's this app has & we find that at start it Pop-Up "Trail Nag" like below



But we are not dealing with this app in similar manner as we do before. A reason of doing this is reside in List of Permission this app required which you find inside "AndroidManifest.xml" :-

```
</application>
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="com.android.vending.CHECK_LICENSE" />--(This is IMP)
</manifest>
```

So as you see app need permission for checking License it means that this trial version has code relating to License verification & if app find license on your Android phone it will run in Register mode & not in trial.

Now we are going to learn how to take advantage of target own restriction & this we can do only by having core understanding of various aspect of target.

Step-2

How & where do we find such License code?

First How

By looking forward to few basic calling method used while coding like

-isRegisteredVersion()

-isTrialVersion()

-isLicensed()

Now Where

You can look for this inside

-Service Manager.smali

-License.smali

-Main Activity.smali (In most of the case name of App itself)

So by following above method we find our required file inside "smali\conceiva\mezzmo" folder in "Mezzmo.smali" (As I say with the name of app)

You can use "aGrep" app if you don't want to do this manually

Step-3

Now let's search for relevant code applying How we find code & here what we have when we try searching for code start with "isTrialVersion()"

```
.method public licenseCheck(Landroid/content/Context;)V
    .locals 17
    .parameter "context"

    .prologue
    .line 5951
    invoke-virtual/range {p0 .. p0}, Lconceiva/mezzmo/Mezzmo; -
>getApplication()Landroid/app/Application;

    move-result-object v12

    check-cast v12, Lconceiva/mezzmo/MezzmoApplication;

    invoke-virtual {v12}, Lconceiva/mezzmo/MezzmoApplication; ->isTrialVersion()Z –
(Main Code Start)

    move-result v12

    if-eqz v12, :cond_2 -----(Humm! Condition let's Dig inn)
```


Ok..So now we find where our code begin let's dig in little to understand what if we jump to Condition 2 & what if we not.

First we go through without jumping the code & we find following code in continuation to above

```
.line 5954
  invoke-virtual/range {p0 .. p1}, Lconceiva/mezzmo/Mezzmo; -
>getInstallDate(Landroid/content/Context;)J

  move-result-wide v10

  .line 5955
  .local v10, installDate:J -----(This will get our app install date)
  new-instance v4, Ljava/util/Date;

  invoke-direct {v4}, Ljava/util/Date; -><init>()V

  .line 5956
  .local v4, date:Ljava/util/Date;
  invoke-virtual {v4}, Ljava/util/Date; ->getTime()J -----(This will get Time)
```

```
. move-result-wide v2

  .line 5957
  .local v2, currentDate:J ----- (This will put Current Date in to Comparison Calculation)
  sub-long v7, v2, v10---(sub-long subtract vale of v10-Install Date, v2-Current Date & put result in
to v7)

  .line 5958
  .local v7, diff:J --(Difference Btw Installed & Current Date Store Here to move into further
calculation)
  long-to-int v12, v7

  div-int/lit16 v12, v12, 0x3e8

  invoke-static {v12}, Lconceiva/mezzmo/Utils; ->splitToComponentTimes(I)[I

  move-result-object v1

  .line 5960
  .local v1, ct:[I
  const/4 v12, 0x0

  aget v12, v1, v12

  const/16 v13, 0x168

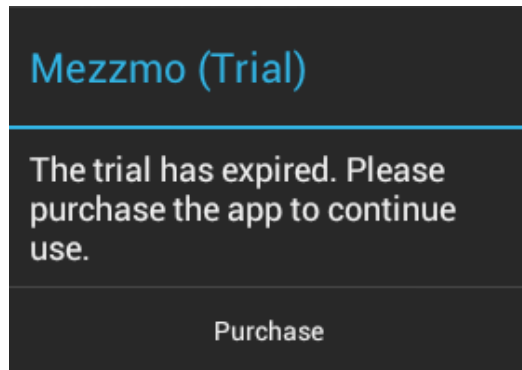
  if-lt v12, v13, :cond_1 ----- (Conditional Jump to success if we have left trial day's > 0)
```

So Now we have two Conditional Jump's :-

Cond_2 ---- (Yet to know where it land's)

Cond_1 ---- (This will Jump to Continue Trail Day's)

If we do not jump Cond_1 we get Trial Expired Msg like follow



Code View: -

```
.line 5961
    const/4 v12, 0x0

    aget v12, v1, v12

    div-int/lit8 v5, v12, 0x18

    .line 5962
    .local v5, daysused:I
    new-instance v9, Landroid/app/AlertDialog$Builder; ---- (Nag Msg Dialog Create
Here)

    move-object/from16 v0, p1

    invoke-direct {v9, v0}, Landroid/app/AlertDialog$Builder; -
> <init> (Landroid/content/Context;)V
    .line 5963
    .local v9, dlgAlert:Landroid/app/AlertDialog$Builder;
    sget v12, Lconceiva/mezzmo/R$string; -> trial_expired:I --- (This will move Trial
Expired Strings)
```

Where we land if we jump to Cond_1

Code View: -

```
:cond_1
    sget-boolean v12, Lconceiva/mezzmo/Mezzmo;->mTrialShown:Z --- (Trial Version flag)

    if-nez v12, :cond_0

    .line 5988
    const/4 v12, 0x0

    aget v12, v1, v12

    div-int/lit8 v5, v12, 0x18

    .line 5989
    .restart local v5      #daysused:I ----- (Trail Day's Remain Popup Start here)
    new-instance v9, Landroid/app/AlertDialog$Builder;

    move-object/from16 v0, p1

    invoke-direct {v9, v0}, Landroid/app/AlertDialog$Builder;-
    ><init>(Landroid/content/Context;)V

    .line 5990
    .restart local v9      #dlgAlert:Landroid/app/AlertDialog$Builder;
    new-instance v12, Ljava/lang/StringBuilder;

    sget v13, Lconceiva/mezzmo/R$string;->trial_active:I ----- (This will activate Trial Version)
```

So it's clear from above if we have left number of day's trial >0 we jump to Cond_1 & then our Trial Version will activated.

Hence we find one method to continue our app even after expiry of trial day's by jumping to Cond_1.

But still this will not all we want, Our purpose is not to defeat trail period but to defeat License check & gain full version control.

So Let's go further with our first finding ie where do Jump lands to Cond_2 & there we land in to lala land with this jump..i.e In License verification Routine

Code View: -

```
:cond_2
    invoke-virtual/range {p0 .. p0}, Lconceiva/mezzmo/Mezzmo; -
>getApplication()Landroid/app/Application;

    move-result-object v12

    check-cast v12, Lconceiva/mezzmo/MezzmoApplication;

    invoke-virtual {v12}, Lconceiva/mezzmo/MezzmoApplication; ->isAmazonVersion()Z
    (Here it's Checking for Amazon Version of Mezzmo ie download from Amazon Play Store
    Because this app will offer in two different play store ie "Google & Amazon")

    move-result v12

    if-nez v12, :cond_0----- (One more IMP Conditional Jump)

    .line 6022
    new-instance v12, Lconceiva/mezzmo/Mezzmo$MyLicenseCheckerCallback;
    (Call for License Check)

    const/4 v13, 0x0

    move-object/from16 v0, p0

    invoke-direct {v12, v0, v13}, Lconceiva/mezzmo/Mezzmo$MyLicenseCheckerCallback; -
><init>(Lconceiva/mezzmo/Mezzmo; Lconceiva/mezzmo/Mezzmo$MyLicenseCheckerCallback;)
    V

    move-object/from16 v0, p0

    iput-object v12, v0, Lconceiva/mezzmo/Mezzmo; -
>mLicenseCheckerCallback:Lcom/google/android/vending/licensing/LicenseCheckerCallback; --
    (Google Vending Standard License check method)
```

Now lets check what if we jump to Cond_0

Code View: -

```
:cond_0
    iget-object v8, p0, Lconceiva/mezzmo/Mezzmo; -
>loadingLayout:Landroid/widget/RelativeLayout; ---- (It's Layout format ie Main Screen When
App Start)

    invoke-virtual {v8, v12}, Landroid/widget/RelativeLayout; ->setVisibility(I)V

    .line 546
    :goto_0
    return-void
```

Yipee! We get the Starting point where we can jump without going through License verification & this is what we can call taking advantage of application own restriction by way of finding loop.

So just set jump (in code mention & discussed above)

- if-eqz v12, :cond_2 change it to goto :cond_2
- if-nez v12, :cond_0 change it to goto :cond_0

Done.

Now Recompile, Zipalign & Sign APK.

Step-4

Now only name is remain to change ie removing of that "Trial" from app name. But I am not going to tell you how it's your home work & do it by your own. (Hint- Fire up your APK Editor & Look in "resources.arsc" to change it)

End Note's

Hop you find something valuable in this paper.

I just teach you how to stand & walk in Android Reversing Scenes now it's up to you how far you learn & convert your small steps it in to running.

Disclaimer

The document published here are to be considered public and freely distributable, provided they cite the source. All document on these pages were written exclusively for research purposes, none of this analysis was done for commercial purposes or behind some type of compensation. I have published documents of a purely theoretical analysis of the structure of a program in any case the software has actually been disassembled or altered. Any correspondence between this published documents and instructions of the software under analysis is to be considered purely coincidental. All documents are sent anonymously and automatically published, the rights of these works belong exclusively to the signatory of the document (ie. me), and in no case shall the operator of this site, or the server that hosts can be held responsible for the content here, plus the site manager is not able to trace the identity of the sender of the documents. All documents and files on this site do not show any kind of warranty, so it is recommended to all readers or execution, the staff assumes no responsibility for the misuse of such documents and / or files it is fair to add that any reference to events or things people have to be considered purely coincidental.

We also recall that the Reverse Engineering is a technological tool of great power and importance, without it would not be possible to create anti-virus, detect malicious functions have not been reported in a program for public use. It would be impossible to discover, in the absence of a secure system for inspecting the integrity, if the "this" program is really what the user chose to install and run, nor would it be possible to continue the development of those programs (or I 'use of those devices) considered obsolete and no longer supported by official sources.