

Reverse Engineering with Kali APKTool



SEAS

Copyright Reserved © Toma Taylor Makoundou and Jeremy Case 2018



Plan

- I. Reverse Engineering Today
- II. Android Application Architecture
- III. Android “Under the Hood”
- IV. What is APKTool?
- V. Trojan Demo: Code Injection
- VI. Conclusion

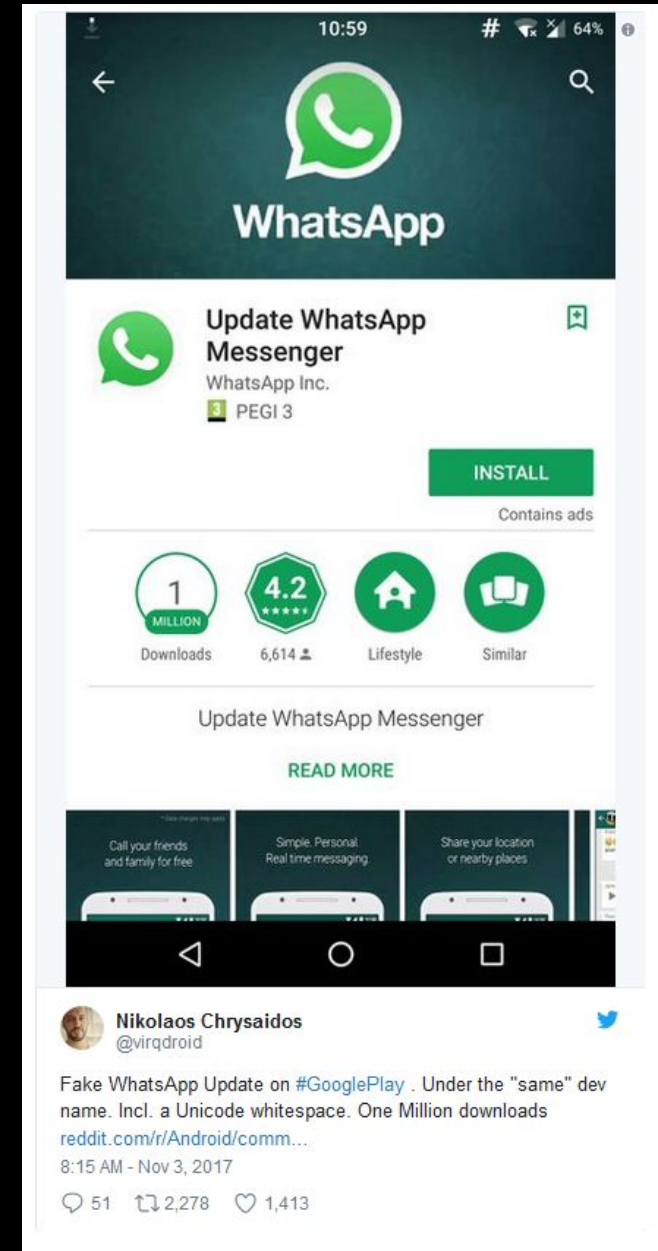


I. Reverse Engineering Today

Last November 2017, a **fake Update WhatsApp Messenger application** was posted on Google Play Store. Aftermath: **more than 1 million of users** downloaded the malicious application.

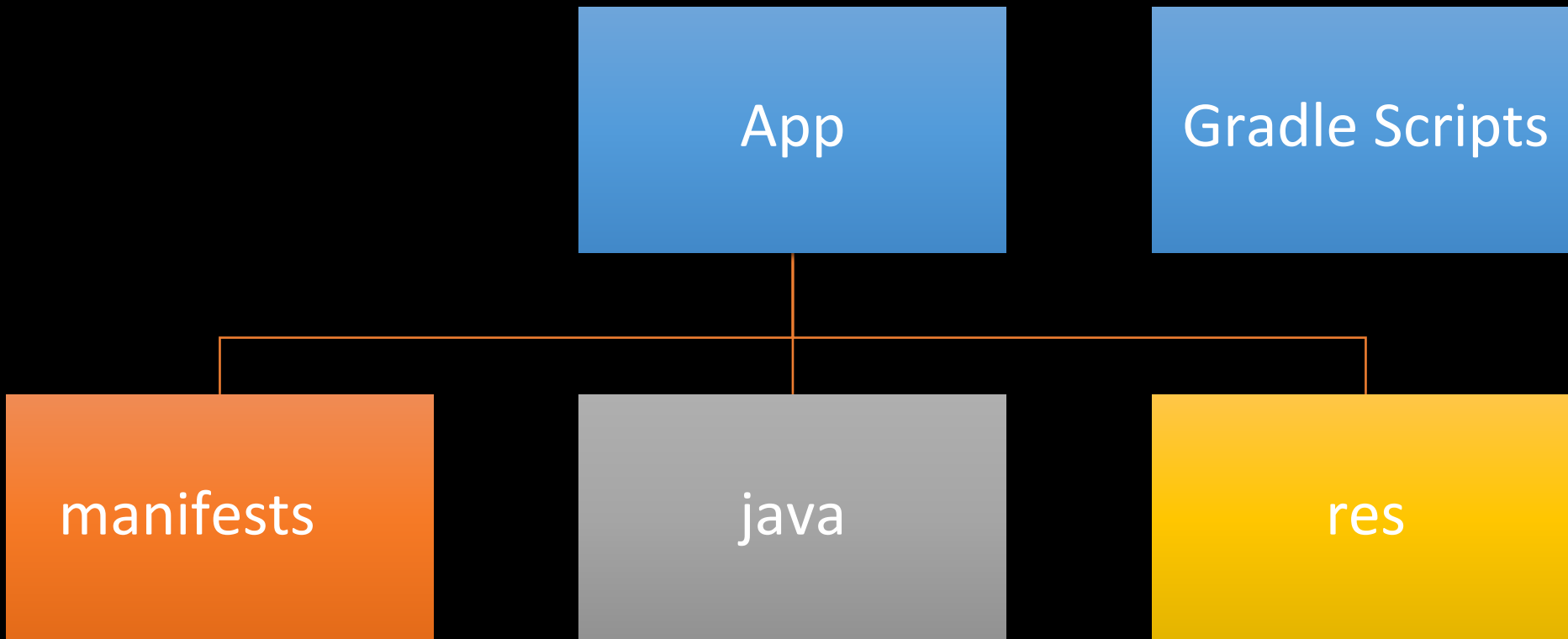
Attack Skeleton:

- **Reverse Engineering** was used to change the original code of WhatsApp
- The malicious app **forces the user to download** Cold Jewel Lines – A well-known Android Malware
- **Hidden components** in app launcher:
 - an empty *app_name* (in the *AndroidManifest.xml* file)
 - a transparent *ic_launcher.png* file, the application icon



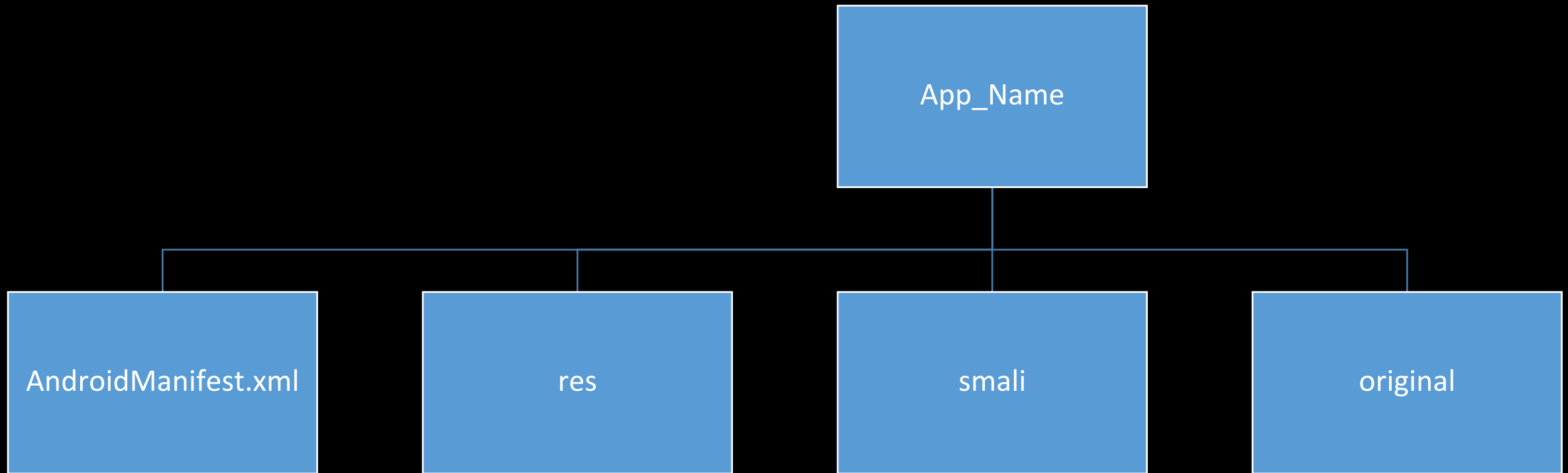
II. Android Application Architecture

- Original Architecture



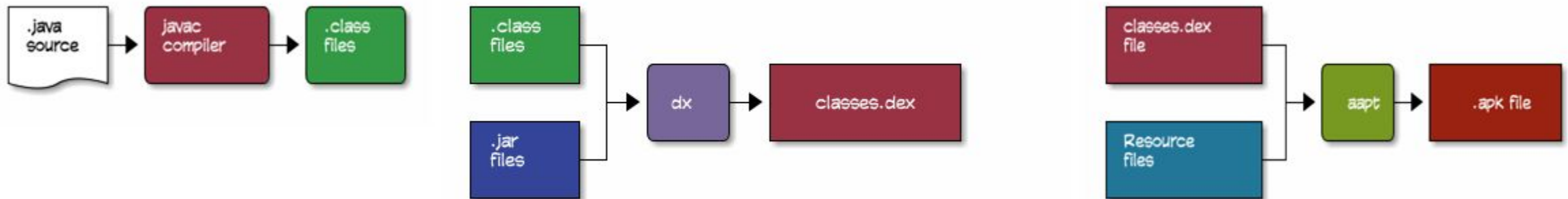
II. Android Application Architecture

- Disassembled Android Application Architecture from the APK



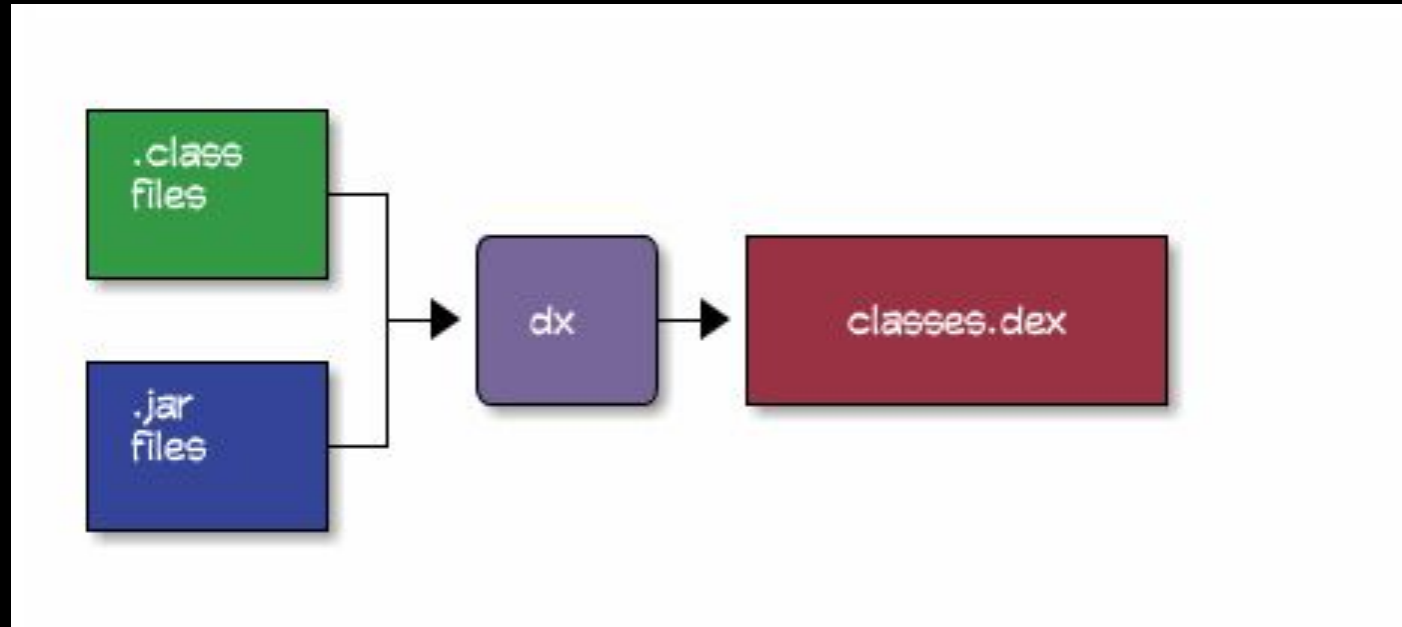
III. Android APK “Under the Hood”

In essence, Android apps written in Java, but are very different than traditional Java programs “under the hood”.



III. Dalvik Virtual Machine

- conversion Java classes into mobile-optimized .dex files.
- conversion from mobile-optimized .dex files. into Java classes is more tedious due to the loss of information when going from JVM to Dalvik.



III. Smali Code (Assembly-like .dex code)

- .dex code is binary bytecode for the Dalvik virtual machine.
- **DISADVANTAGE:** .dex code is not readable for humans.
- **SOLUTION:** use reverse-engineering tools, such as APKTool, to convert .dex code into Smali, an Assembly-like language.

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000660 48 65 6C 6C 6F 2C 20 77 6F 72 6C 64 2E 36 34 30 Hello, world.640
00000670 30 39 30 39 31 36 35 30 30 30 31 39 36 32 33 0D 090916500019623.
00000680 0A 3A 31 30 31 45 35 30 30 30 39 30 39 33 36 35 .:101E5000909365
00000690 30 30 38 30 39 33 36 34 30 30 32 46 35 46 33 46 00809364002F5F3F
000006A0 34 46 38 30 39 31 36 36 30 31 45 46 0D 0A 3A 31 4F80916601EF...1
000006B0 54 68 69 73 20 69 73 20 61 20 68 65 78 61 64 65 This is a hexade
000006C0 63 69 6D 61 6C 20 74 75 74 6F 72 69 61 6C 21 46 cimal tutorial!F
000006D0 38 39 34 45 31 39 39 33 36 0D 0A 3A 31 30 31 45 894E19936...101E
000006E0 37 30 30 30 0D 01 02 03 04 05 06 07 08 09 0A 0B 7000.....
000006F0 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B .....
00000700 1C 1D 1E 1F 20 21 22 23 24 25 26 27 28 29 2A 2B .... !"#%&'()*+
00000710 2C 2D 2E 2F 30 31 32 33 34 35 36 37 38 39 3A 3B ,-./0123456789:;
00000720 3C 3D 3E 3F 40 41 42 43 44 45 46 47 48 49 4A 4B <=>?@ABCDEFGHIJK
00000730 4C 4D 4E 4F 50 51 52 53 54 55 56 57 58 59 5A 5B LMNOPQRSTUVWXYZ{
00000740 5C 5D 5E 5F 60 61 62 63 64 65 66 67 68 69 6A 6B }^_`abcdefghijklmnopqrstuvwxyz
00000750 6C 6D 6E 6F 70 71 72 73 74 75 76 77 78 79 7A 7B lmnopqrstuvwxyz{
00000760 7C 7D 7E 7F 80 81 82 83 84 85 86 87 88 89 8A 8B }~.C.,[~!@!$%
00000770 8C 8D 8E 8F 90 91 92 93 94 95 96 97 98 99 9A 9B 012...~!@!$%
00000780 9C 9D 9E 9F 0A 01 A2 A3 A4 A5 A6 A7 A8 A9 AA AB e.ZY`!C!E!V!@!#
00000790 AC AD AE AF 0B 01 02 03 04 05 06 07 08 09 0A 0B ~.0!~!~!~!~!~!~!
000007A0 BC BD BE BF 0C 01 C2 C3 C4 C5 C6 C7 C8 C9 CA CB 00%AAAAAAACEEEE
000007B0 CC CD CE CF 0D 01 D2 D3 D4 D5 D6 D7 D8 D9 DA DB IIII000000000000
000007C0 DC DD DE DF 0E 01 E2 E3 E4 E5 E6 E7 E8 E9 EA EB 0YPSAAAAAAAcceeee
000007D0 EC ED EE EF 0F 01 F2 F3 F4 F5 F6 F7 F8 F9 FA FB IIII000000000000
000007E0 FC FD FE FF 03 39 43 0D 0A 3A 31 30 31 45 44 30 0Ypy39C...:101ED0
000007F0 30 30 35 37 30 30 45 38 39 35 33 32 39 36 30 32 005700E895329602
```



```
org Smali java ByteCode
0 0x0 const/16 v0, [0+ 4096], [4096]
1 0x1 new-array v1, v0, [type@ 200 [0]
2 0x2 const/4 v5, [0+ 0], [0]
3 0x3 if-gez v5, [0+ 3]
4 0x4 return-void
5 0x5 get-object v0, v1, [field@ 27 Lcom/android/root/adfroot;:Ljava/io/FileOutputStream; v0$in]
6 0x6 invoke-virtual v0, v4, [method@ 1132 Ljava/io/FileOutputStream; :()I read]
7 0x7 move-result v5
8 0x8 new-instance v7, [type@ 228 Ljava/lang/String;]
9 0x9 const/4 v0, [0+ 0], [0]
10 0xA invoke-direct v7, v0, v5, [method@ 1195 Ljava/lang/String; :()I v-into]
11 0xB const-string v0, [string@ 150 Forked]
12 0xC invoke-virtual v7, v0, [method@ 1198 Ljava/lang/String; :()Ljava/lang/CharSequence; Z contains]
13 0xD move-result v0
14 0xE if-gez v0, [0+ 90]
15 0xF new-instance v3, [type@ 15 Landroid/content/Intent;]
16 0x10 get-object v0, v11, [field@ 28 Lcom/android/root/adfroot;:Lcom/android/root/adfroot; this$0]
17 0x11 invoke-static v0, [method@ 205 Lcom/android/root/adfroot; :()Landroid/content/Context; access$0]
18 0x12 move-result-object v0
19 0x13 const-class v5, [type@ 107 Lcom/android/root/AlarmReceiver;]
20 0x14 invoke-direct v3, v0, v0, [method@ 52 Landroid/content/Intent; :()Landroid/content/Context; Ljava/lang/Class; V-into]
21 0x15 const-string v0, [string@ 1390 start]
22 0x16 const/4 v0, [0+ 1], [1]
23 0x17 invoke-virtual v3, v0, v0, [method@ 85 Landroid/content/Intent; :()Ljava/lang/String; Z Landroid/content/Intent; putExtra]
24 0x18 get-object v0, v11, [field@ 28 Lcom/android/root/adfroot;:Lcom/android/root/adfroot; this$0]
25 0x19 invoke-static v0, [method@ 205 Lcom/android/root/adfroot; :()Landroid/content/Context; access$0]
26 0x1A move-result-object v0
27 0x1B const/4 v0, [0+ 0], [0]
28 0x1C const/4 v20, [0+ 0], [0]
29 0x1D invoke-static v0, v9, v9, v10, [method@ 36 Landroid/app/PendingIntent; :()Landroid/content/Context; I Landroid/app/PendingIntent; getRaw]
30 0x1E move-result-object v0
31 0x1F get-object v0, v11, [field@ 28 Lcom/android/root/adfroot;:Lcom/android/root/adfroot; this$0]
32 0x20 invoke-static v0, [method@ 205 Lcom/android/root/adfroot; :()Landroid/content/Context; access$0]
33 0x21 move-result-object v0
34 0x22 const-string v0, [string@ 811 alarm]
35 0x23 invoke-virtual v0, v9, [method@ 40 Landroid/content/Context; :()Ljava/lang/String; Ljava/lang/Object; getSystemService]
36 0x24 move-result-object v0
37 0x25 check-cast v0, [type@ 7 Landroid/os/AlarmManager;]
```


IV. What is the APKTool?

*According to the APKTool page, the APKtool is “a tool for reverse engineering 3rd party, closed, binary Android apps. It can **decode resources** to nearly original form and **rebuild them** after making some modifications. It also makes working with an app easier because of the project like file structure and automation of some repetitive tasks like building apk, etc.”.*

The APKTool includes multiple features:

- Disassembling resources to nearly original form (including resources.arsc, classes.dex, 9.png. and XMLs)
- Rebuilding decoded resources back to binary APK/JAR
- Organizing and handling APKs that depend on framework resources
- Smali Debugging (Removed in 2.1.0 in favor of IdeaSmali)
- Helping with repetitive tasks

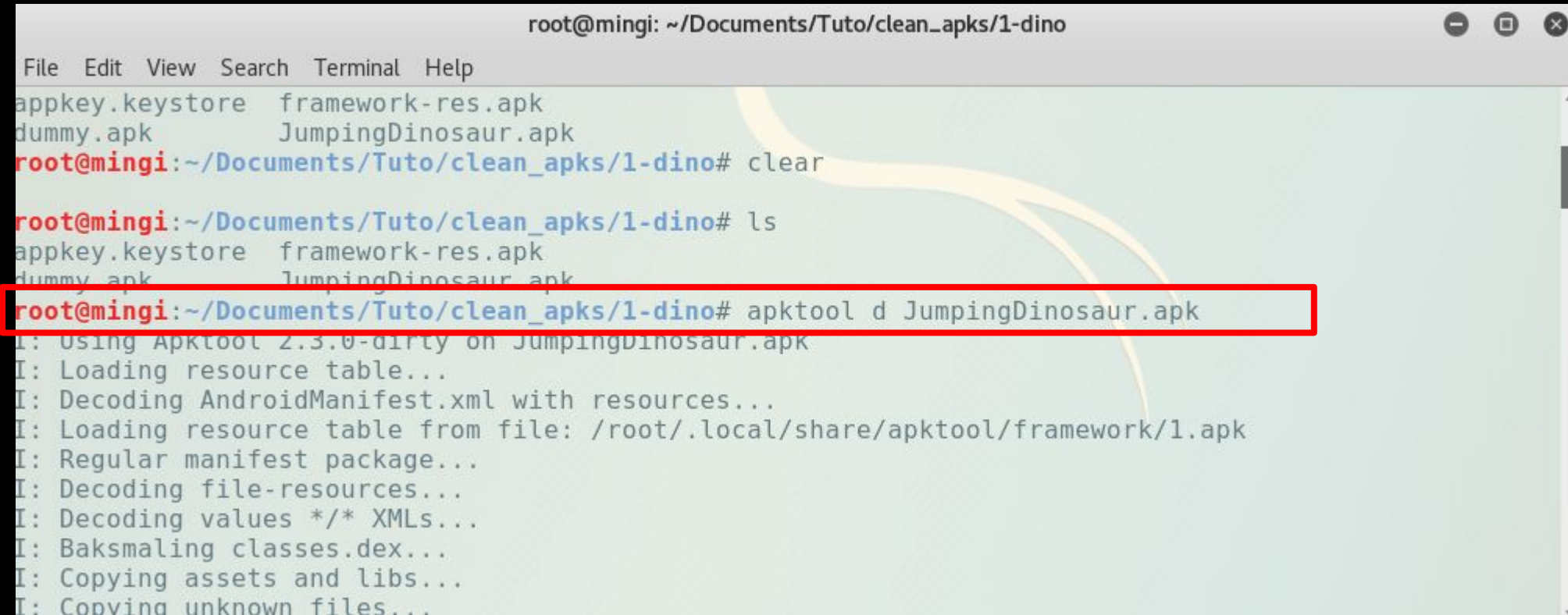
V. Demo: Code Injection



/!\ WARNING /!\

It is not recommended to perform this demo
on your personal working Android device.

V. Demo: Code Injection

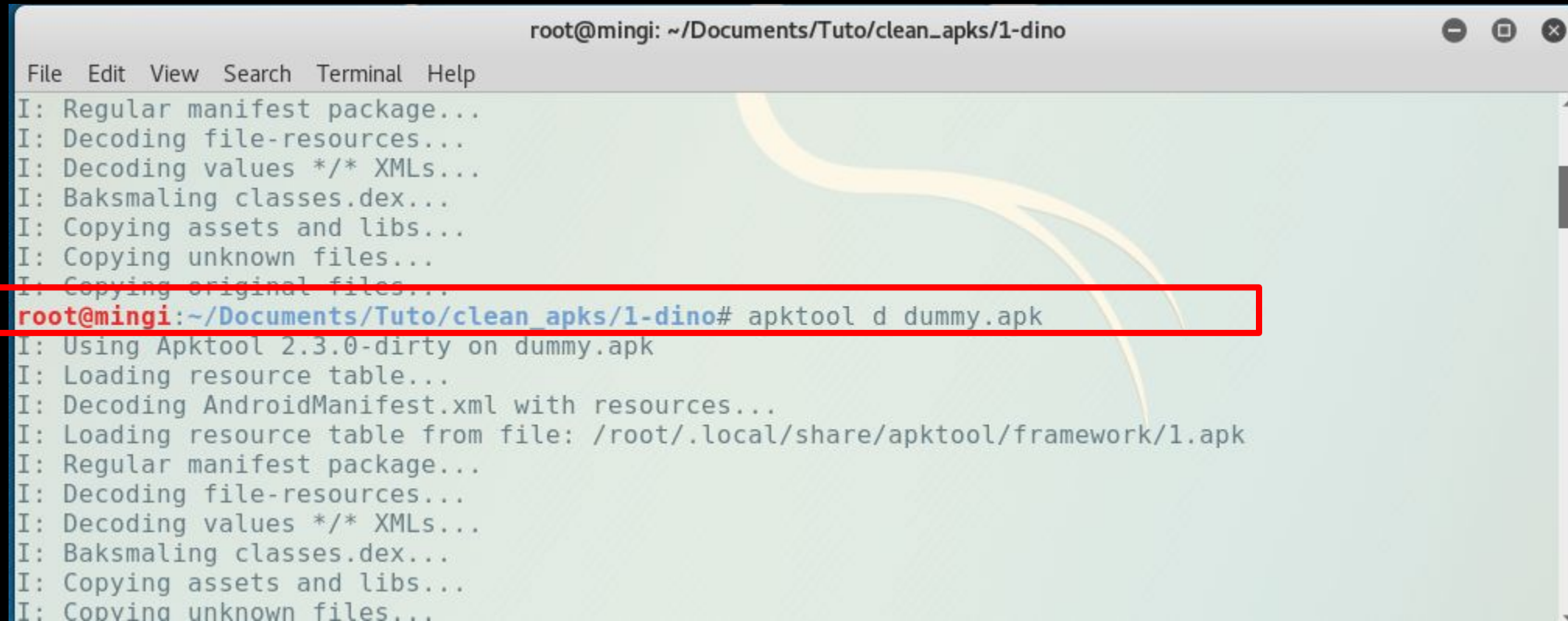


A terminal window titled "root@mingi: ~/Documents/Tuto/clean_apks/1-dino" with a menu bar (File, Edit, View, Search, Terminal, Help). The window shows a list of files: appkey.keystore, framework-res.apk, dummy.apk, and JumpingDinosaur.apk. The user enters the command "clear", then "ls", and finally "apktool d JumpingDinosaur.apk", which is highlighted with a red rectangle. The output of the command shows the decompilation process using Apktool 2.3.0-dirty.

```
root@mingi: ~/Documents/Tuto/clean_apks/1-dino
File Edit View Search Terminal Help
appkey.keystore framework-res.apk
dummy.apk      JumpingDinosaur.apk
root@mingi:~/Documents/Tuto/clean_apks/1-dino# clear
root@mingi:~/Documents/Tuto/clean_apks/1-dino# ls
appkey.keystore framework-res.apk
dummy.apk      JumpingDinosaur.apk
root@mingi:~/Documents/Tuto/clean_apks/1-dino# apktool d JumpingDinosaur.apk
I: Using Apktool 2.3.0-dirty on JumpingDinosaur.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...
```

apktool d JumpingDinosaur.apk

V. Demo: Code Injection



```
root@mingi: ~/Documents/Tuto/clean_apks/1-dino
File Edit View Search Terminal Help
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@mingi:~/Documents/Tuto/clean_apks/1-dino# apktool d dummy.apk
I: Using Apktool 2.3.0-dirty on dummy.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...
```

apktool d dummy.apk

Inject Malicious Code

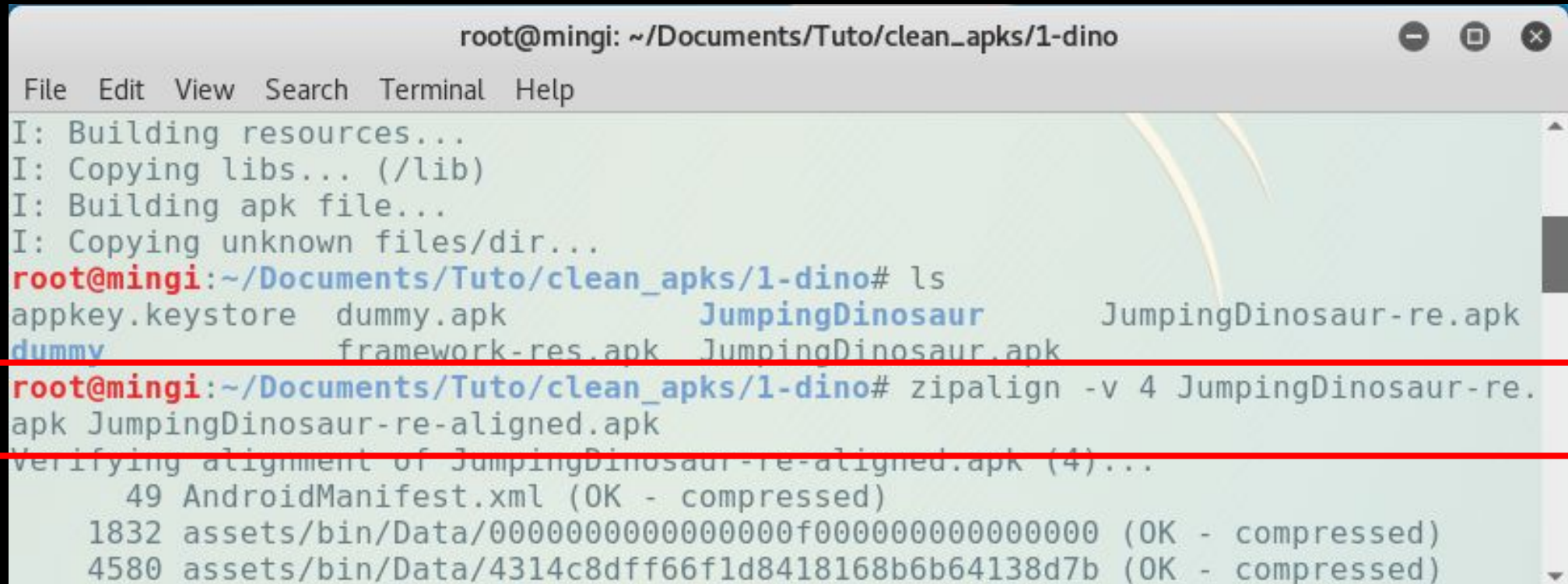
N.B: Please refer to the handout.

V. Demo: Code Injection

```
root@mingi: ~/Documents/Tuto/clean_apks/1-dino
File Edit View Search Terminal Help
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...
root@mingi:~/Documents/Tuto/clean_apks/1-dino# ls
appkey.keystore  dummy.apk          JumpingDinosaur
dummy           framework-res.apk  JumpingDinosaur.apk
root@mingi:~/Documents/Tuto/clean_apks/1-dino# cd JumpingDinosaur/
root@mingi:~/Documents/Tuto/clean_apks/1-dino/JumpingDinosaur# gedit AndroidManifest.xml
root@mingi:~/Documents/Tuto/clean_apks/1-dino/JumpingDinosaur# cd ..
root@mingi:~/Documents/Tuto/clean_apks/1-dino# ls
appkey.keystore  dummy.apk          JumpingDinosaur
dummy           framework-res.apk  JumpingDinosaur.apk
root@mingi:~/Documents/Tuto/clean_apks/1-dino# apktool b JumpingDinosaur -o JumpingDinosaur-re.apk
I: Using Apktool 2.3.0 dirty
I: Checking whether sources has changed...
I: Smaling smali folder into classes.dex...
I: Checking whether resources has changed...
I: Building resources...
```

apktool b JumpingDinosaur -o JumpingDinosaur-re.apk

V. Demo: Code Injection



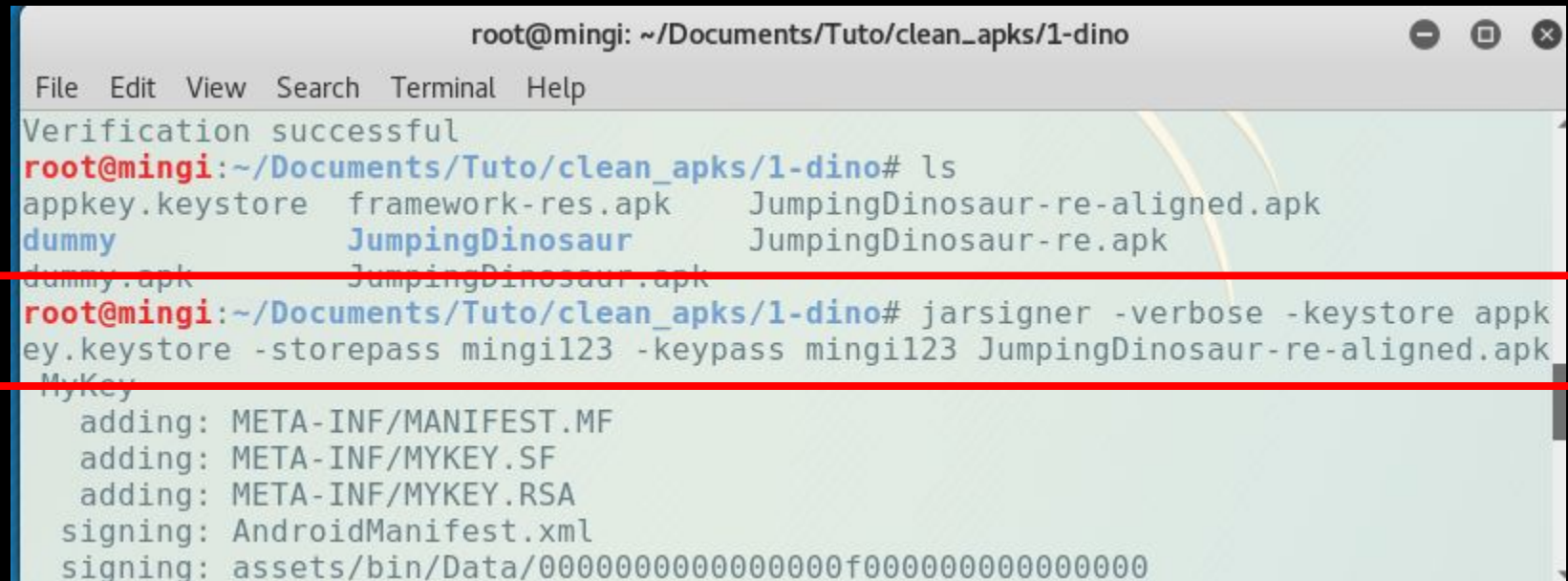
The screenshot shows a terminal window titled "root@mingi: ~/Documents/Tuto/clean_apks/1-dino". The terminal output shows the following steps:

```
File Edit View Search Terminal Help
I: Building resources...
I: Copying libs... (/lib)
I: Building apk file...
I: Copying unknown files/dir...
root@mingi:~/Documents/Tuto/clean_apks/1-dino# ls
appkey.keystore  dummy.apk          JumpingDinosaur    JumpingDinosaur-re.apk
dummy           framework-res.apk  JumpingDinosaur.apk
root@mingi:~/Documents/Tuto/clean_apks/1-dino# zipalign -v 4 JumpingDinosaur-re.apk JumpingDinosaur-re-aligned.apk
Verifying alignment of JumpingDinosaur-re-aligned.apk (4)...
  49 AndroidManifest.xml (OK - compressed)
1832 assets/bin/Data/000000000000000000f000000000000000 (OK - compressed)
4580 assets/bin/Data/4314c8dff66f1d8418168b6b64138d7b (OK - compressed)
```

A red rectangular box highlights the command `zipalign -v 4 JumpingDinosaur-re.apk JumpingDinosaur-re-aligned.apk` and its output.

`zipalign -v 4 JumpingDinosaur-re.apk JumpingDinosaur-re-aligned.apk`

V. Demo: Code Injection



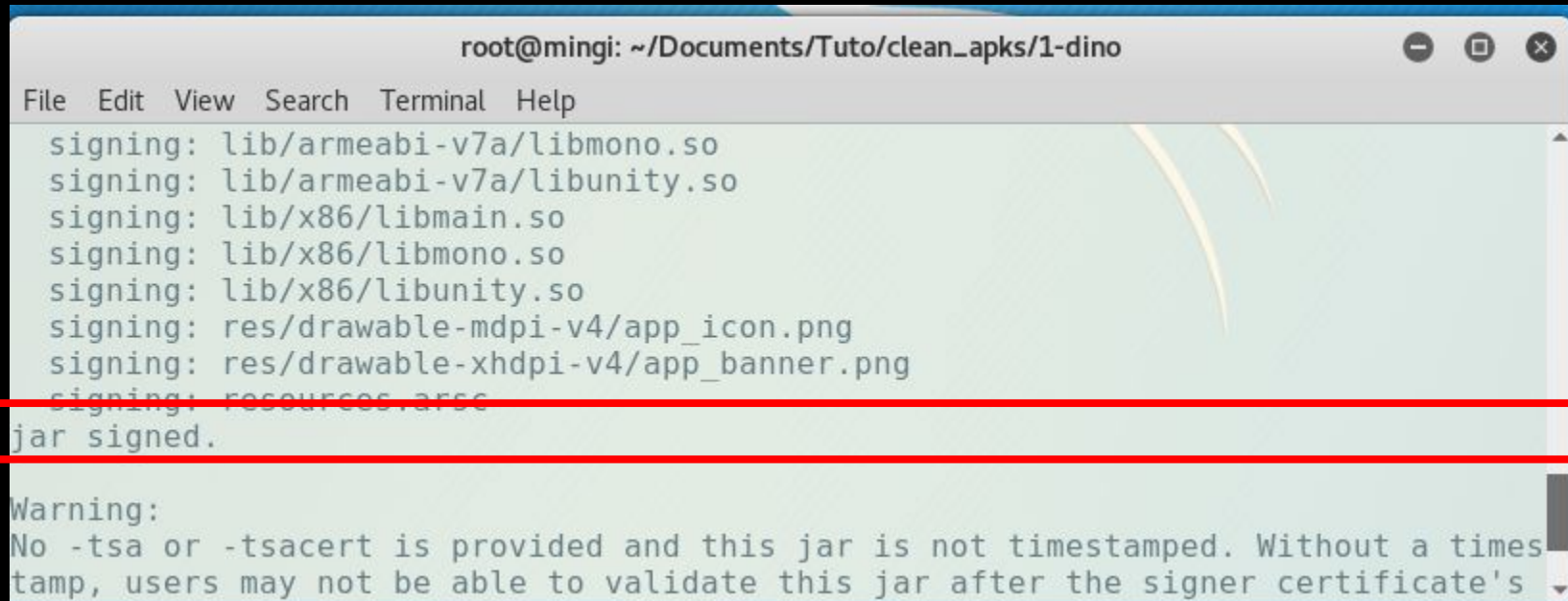
A terminal window titled "root@mingi: ~/Documents/Tuto/clean_apks/1-dino" showing the following commands and output:

```
Verification successful
root@mingi:~/Documents/Tuto/clean_apks/1-dino# ls
appkey.keystore  framework-res.apk  JumpingDinosaur-re-aligned.apk
dummy           JumpingDinosaur    JumpingDinosaur-re.apk
dummy.apk       JumpingDinosaur.apk
root@mingi:~/Documents/Tuto/clean_apks/1-dino# jarsigner -verbose -keystore appkey.keystore -storepass mingi123 -keypass mingi123 JumpingDinosaur-re-aligned.apk MyKey
adding: META-INF/MANIFEST.MF
adding: META-INF/MYKEY.SF
adding: META-INF/MYKEY.RSA
signing: AndroidManifest.xml
signing: assets/bin/Data/00000000000000000f00000000000000
```

The command line for the `jarsigner` command is highlighted with a red rectangle.

```
jarsigner -verbose -keystore appkey.keystore -storepass mingi123  
-keypass mingi123 JumpingDinosaur-re-aligned.apk MyKey
```

V. Demo: Code Injection

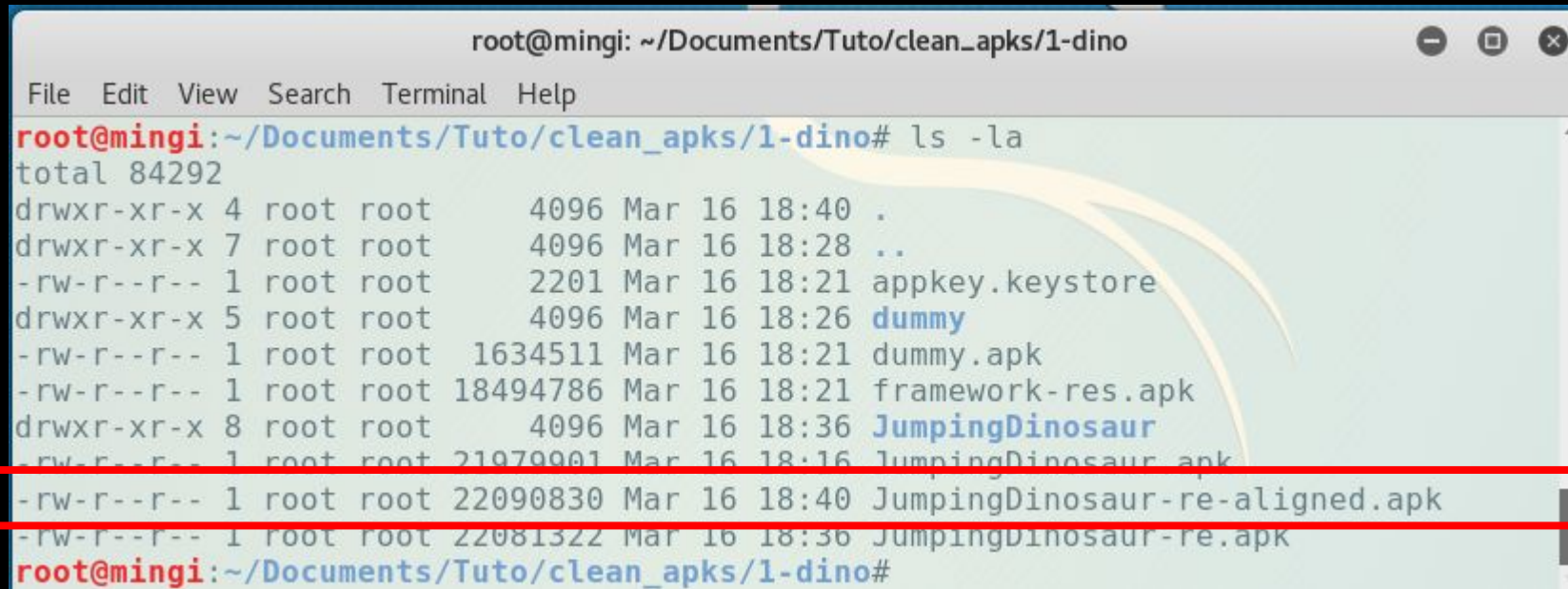
A terminal window titled 'root@mingi: ~/Documents/Tuto/clean_apks/1-dino' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal output shows the signing of various files: 'lib/armeabi-v7a/libmono.so', 'lib/armeabi-v7a/libunity.so', 'lib/x86/libmain.so', 'lib/x86/libmono.so', 'lib/x86/libunity.so', 'res/drawable-mdpi-v4/app_icon.png', and 'res/drawable-xhdpi-v4/app_banner.png'. The line 'signing: resources.arsc' is highlighted with a red box, and the final output is 'jar signed.'. Below this, a warning message is displayed: 'Warning: No -tsa or -tsacert is provided and this jar is not timestamped. Without a times tamp, users may not be able to validate this jar after the signer certificate's'.

```
root@mingi: ~/Documents/Tuto/clean_apks/1-dino
File Edit View Search Terminal Help
signing: lib/armeabi-v7a/libmono.so
signing: lib/armeabi-v7a/libunity.so
signing: lib/x86/libmain.so
signing: lib/x86/libmono.so
signing: lib/x86/libunity.so
signing: res/drawable-mdpi-v4/app_icon.png
signing: res/drawable-xhdpi-v4/app_banner.png
signing: resources.arsc
jar signed.

Warning:
No -tsa or -tsacert is provided and this jar is not timestamped. Without a times
tamp, users may not be able to validate this jar after the signer certificate's
```

jar signed

V. Demo: Code Injection



A terminal window titled 'root@mingi: ~/Documents/Tuto/clean_apks/1-dino' displays the output of the 'ls -la' command. The output lists several files, including 'appkey.keystore', 'dummy', 'dummy.apk', 'framework-res.apk', 'JumpingDinosaur', 'JumpingDinosaur.apk', 'JumpingDinosaur-re-aligned.apk', and 'JumpingDinosaur-re.apk'. A red rectangular box highlights the line for 'JumpingDinosaur-re-aligned.apk'. A yellow curved arrow points from the 'JumpingDinosaur' directory entry to the highlighted file.

```
root@mingi: ~/Documents/Tuto/clean_apks/1-dino
File Edit View Search Terminal Help
root@mingi:~/Documents/Tuto/clean_apks/1-dino# ls -la
total 84292
drwxr-xr-x 4 root root    4096 Mar 16 18:40 .
drwxr-xr-x 7 root root    4096 Mar 16 18:28 ..
-rw-r--r-- 1 root root    2201 Mar 16 18:21 appkey.keystore
drwxr-xr-x 5 root root    4096 Mar 16 18:26 dummy
-rw-r--r-- 1 root root 1634511 Mar 16 18:21 dummy.apk
-rw-r--r-- 1 root root 18494786 Mar 16 18:21 framework-res.apk
drwxr-xr-x 8 root root    4096 Mar 16 18:36 JumpingDinosaur
-rw-r--r-- 1 root root 21979901 Mar 16 18:16 JumpingDinosaur.apk
-rw-r--r-- 1 root root 22090830 Mar 16 18:40 JumpingDinosaur-re-aligned.apk
-rw-r--r-- 1 root root 22081322 Mar 16 18:36 JumpingDinosaur-re.apk
root@mingi:~/Documents/Tuto/clean_apks/1-dino#
```

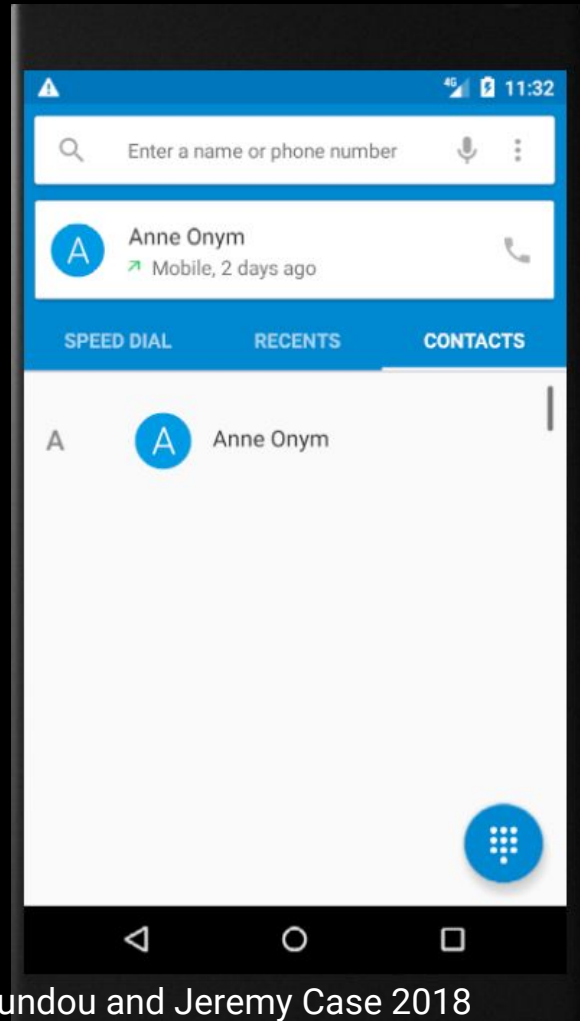
JumpingDinosaur-re-aligned is the **malicious app!**

V. Demo: Code Injection

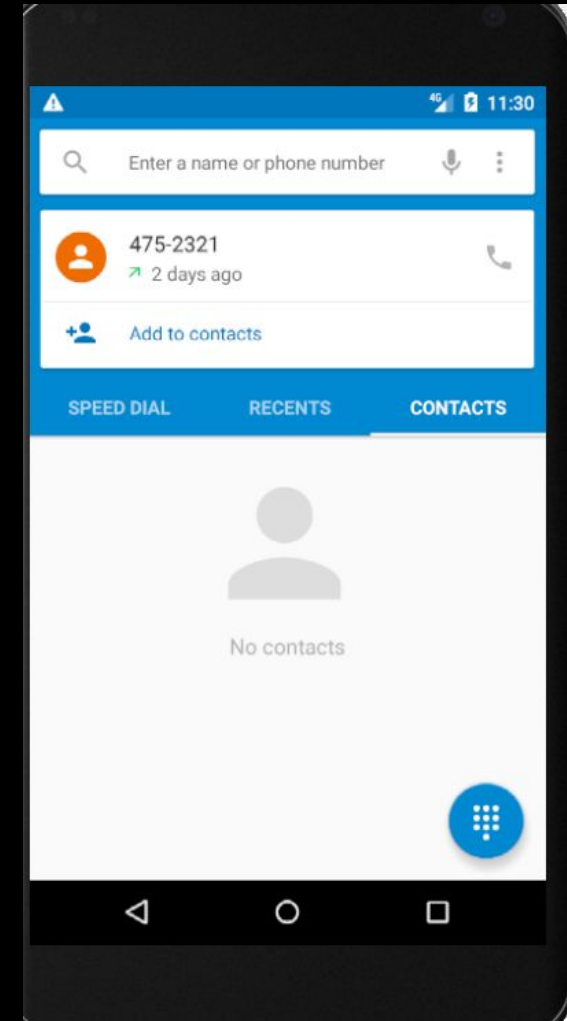
Start the malicious app



Check your contacts



Restart your phone.
Your contacts are gone.



V. Demo: Code Injection

How do you protect an app from code injection ?

Some suggestions:

- 1) Sign the application before submission to the Google Play Store
- 2) Keep your keys secured
- 3) Attach the app to a hashcode to ensure Integrity

VI. Conclusion

As ethical hackers, the onus is on us to **abide** by the law, **inform** clients of possible entry points and **mitigate** risks.

Therefore, Apktool is a useful tool for discovering the vulnerabilities of an Android application in order to preserve its confidentiality, integrity and availability.

References

- <https://developer.android.com/studio/publish/app-signing.html#signing-manually>
- <https://ibotpeaches.github.io/Apktool/documentation/>
- Source:<https://www.xda-developers.com/fake-whatsapp-updater-malware-google-play-store/>

THANK YOU FOR LISTENING!



ANY QUESTIONS?



SEAS

Copyright Reserved © Toma Taylor Makoundou and Jeremy Case 2018

