Extremely Vulnerable Android Labs(EVABS) APPS FULL WALKTHROUGH (WITH FLAG).

First of all, these walkthroughs were intentionally done fully in the Mobexler VM with the aim that this VM can cater to all of the tools needed in a single mobile tailored VM as part of assessing vulnerabilities and hacking into the mobile applications. EVABS was chosen as a target since there is no reliable writeup about this vulnerable app to be used as a reference and guide in order to complete all of the tasks.

As part of testing, here is the list of software used to complete this walkthrough:

- Mobexler VM
- Genymotion vers 3.1.1
- EVABS app vers 1.1 release 4
- Google Pixel 3 XL (as emulator)

Before begin, shoutout to the creator of this awesome EVABS, abhi-r3v0 that made us capable of testing and improve our competency by using his apps. Not to forget also the Enciphers team that hardworkingly made this magnificent mobile pentest tailored VM platform to be used in our daily pentesting.

In writing this walkthrough, I'll be laymen as I can and easy to follow accordingly thus can be referred by the beginner or someone that just encounters to do mobile penetration testing. In every part of the challenge level also I'll put some best practice, support links, alternative steps, and other related stuff on solving those challenges. Disclaim that every step I used may differ from the actual or best security practice as the goals of this EVABS app is getting the flag hidden inside the app. Hence, even the steps kinda messy and very much loophole, as long as the idea is there and easily understandable then it'll be very much helpful towards mastering the basics of mobile pentesting.

Initial steps.

Begin with opening the Genymotion that already setup the Google Pixel 3 XL android emulator, then Install EVABS in it. Follow the EVABS first time user registration accordingly then later you may use the app. While doing so, Mobexler should be opened at the very moment so then can proceed with doing the challenges.

In Mobexler, open the terminal and type *adb devices,* to see any connected devices. Since this is the first time setup so there is no device attached.

As my Genymotion IP is shown in the picture below, type *adb connect your.IP.address* to connect your mobile to the Mobexler adb. Next, type *adb devices* back to ensure your mobile is attached.

Using *frida-ps -Ua* to enumerate all of the running applications on your mobile. If an error like mine popup, simply use this command to fire up back the Frida server on your mobile.

Adb shell

Su

Cd /data/local/tmp

./frida-server &

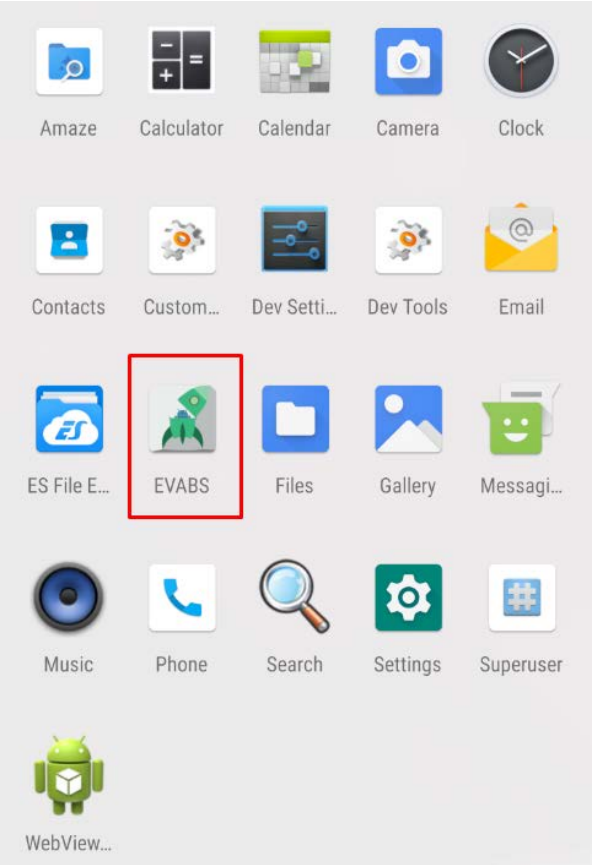
Or

Adb shell /data/loca/tmp/frida-server &

Note that this Frida server already pushed into the mobile in the first place. To doing so, kindly refer this link for Frida setup and installation. Here.

```
 Mobexler@Mobexler  ~  adb devices                                          1 ↵  25  08:00:03
List of devices attached

 Mobexler@Mobexler  ~  adb connect 192.168.28.101                            ✔  26  08:00:07
connected to 192.168.28.101:5555
 Mobexler@Mobexler  ~  adb devices                                           ✔  27  08:00:35
List of devices attached
192.168.28.101:5555     device

 Mobexler@Mobexler  ~  frida-ps -Ua                                          ✔  28  08:00:36
Failed to enumerate applications: unable to connect to remote frida-server: closed
 Mobexler@Mobexler  ~  adb shell                                             1 ↵  29  08:00:47
vbox86p:/ # su
:/ # cd /data/local/tmp
:/data/local/tmp # ls
frida-server re.frida.server
:/data/local/tmp # ./frida-server &
[1] 2085
:/data/local/tmp # exit
```

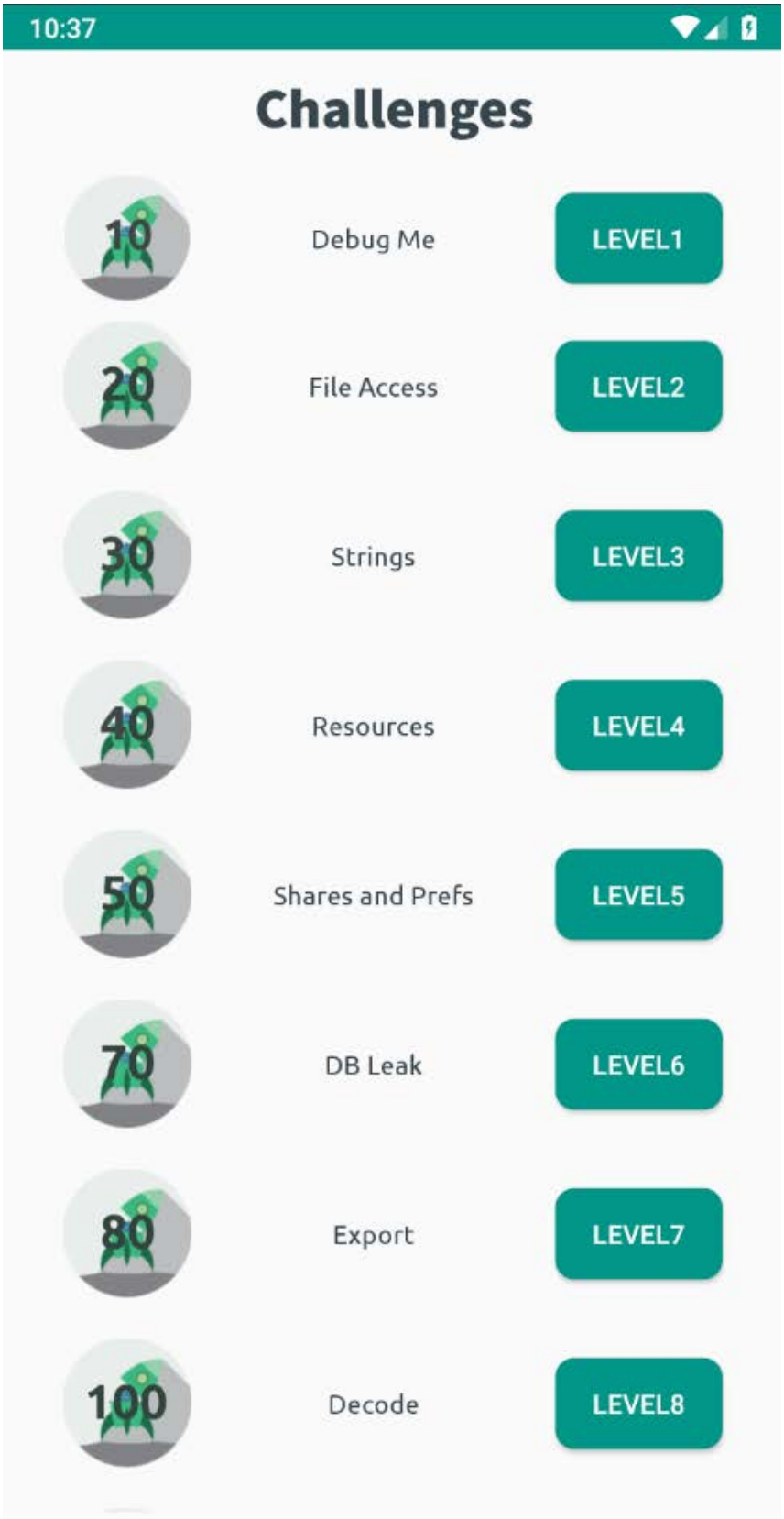
In your mobile, open the EVABS app back (if close after the first setup session) and let it run as it is.



Go back to the Mobexler terminal, type back *frida-ps -Ua* command to enumerate back all running applications in your mobile, and as you can see In the picture below, the EVABS app is listed with PID 2134. Your app PID may be different and for sure different with mine so take note on that.

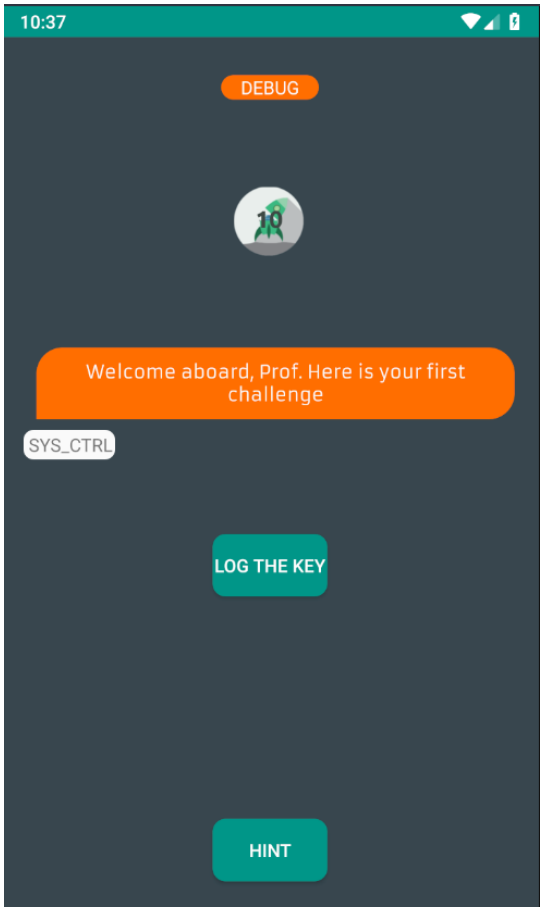
```
Mobexler@Mobexler  ~  frida-ps -Ua
 PID  Name                        Identifier
----  ------------------------    ------------------------------------
 706  Android Keyboard (AOSP)     com.android.inputmethod.latin
 920  Android Services Library    android.ext.services
 581  Android System              android
1415  Blocked Numbers Storage     com.android.providers.blockednumber
1558  Calendar Storage            com.android.providers.calendar
 581  Call Management             com.android.server.telecom
1544  Cell Broadcasts             com.android.cellbroadcastreceiver
1415  Contacts Storage            com.android.providers.contacts
2134  EVABS                       com.revo.evabs
1633  Email                       com.android.email
 581  Fused Location              com.android.location.fused
1854  Key Chain                   com.android.keychain
1680  Messaging                   com.android.messaging
 792  MmsService                  com.android.mms.service
1360  Phone                       com.android.dialer
 792  Phone Services              com.android.phone
 792  Phone and Messaging St…     com.android.providers.telephony
1211  Print Spooler               com.android.printspooler
1169  Quickstep                   com.android.launcher3
1116  SecureElementApplicati…     com.android.se
 802  Settings                    com.android.settings
 581  Settings Storage            com.android.providers.settings
 725  System UI                   com.android.systemui
1415  User Dictionary             com.android.providers.userdictionary
1185  com.android.smspush         com.android.smspush
1151  com.genymotion.genyd.…      com.genymotion.genyd
1093  com.genymotion.system…      com.genymotion.systempatcher
1132  com.genymotion.taskloc…     com.genymotion.tasklocker
Mobexler@Mobexler  ~  |
```

In EVABS, there are 12 challenges that need to be solved and each of the challenges marked with points indicates the difficulty of the level. However all those points are not cumulatively added as overall point and not show in any part of the app, maybe this the part dev need to fix or improve.



CHALLENGE LEVEL 1

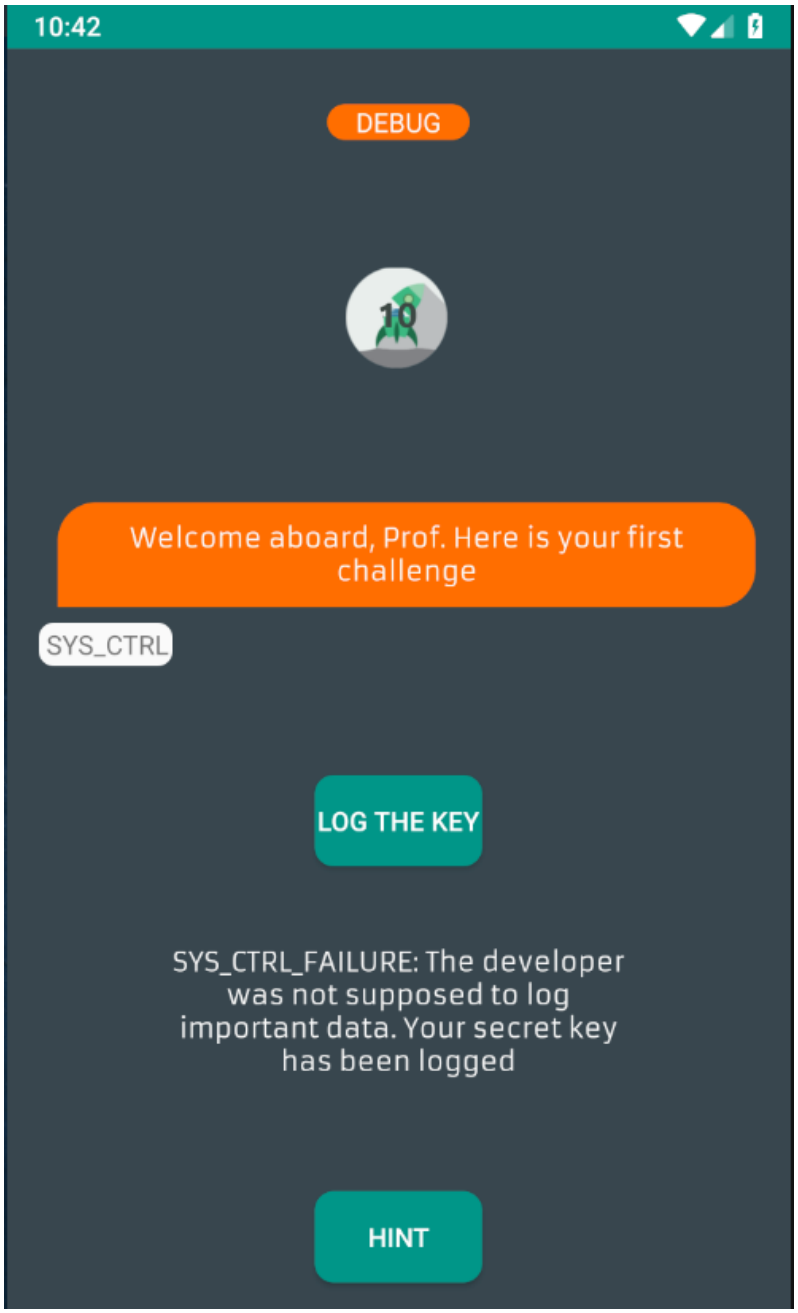
Start with the first challenge of the app, it starts with the most common vulnerability within a mobile app which is debugged log listing. You may press the Hint button for more challenge level details.



In the Mobexler terminal, using this command; *adb logcat --pid=pidnumber* to fire up debug log monitor specific to the EVABS app using its PID number queried using the previous command in the initial steps.

```
Mobexler@Mobexler  ~  adb logcat --pid=2134                                   INT(-2)  34  08:11:49
--------- beginning of main
09-21 22:35:45.926  2134  2134 I Zygote  : seccomp disabled by setenforce 0
09-21 22:35:45.932  2134  2134 I com.revo.evabs: Late-enabling -Xcheck:jni
09-21 22:35:45.967  2134  2134 W com.revo.evabs: Unexpected CPU variant for X86 using defaults: x86
09-21 22:35:46.060  2134  2134 I com.revo.evabs: The ClassLoaderContext is a special shared library.
09-21 22:35:46.133  2134  2134 D FirebaseApp: com.google.firebase.iid.FirebaseInstanceId is not linked. Skipping initialization.
09-21 22:35:46.135  2134  2134 D FirebaseApp: com.google.firebase.crash.FirebaseCrash is not linked. Skipping initialization.
09-21 22:35:46.141  2134  2134 D FirebaseApp: com.google.android.gms.measurement.AppMeasurement is not linked. Skipping initialization.
09-21 22:35:46.141  2134  2134 I FirebaseInitProvider: FirebaseApp initialization successful
09-21 22:35:46.236  2134  2163 D libEGL  : Emulator has host GPU support, qemu.gles is set to 1.
09-21 22:35:46.254  2134  2163 D vndksupport: Loading /vendor/lib/egl/libGLES_emulation.so from current namespace instead of sphal namespace
.
09-21 22:35:46.316  2134  2163 E libEGL  : load_driver(/vendor/lib/egl/libGLES_emulation.so): dlopen failed: library "/vendor/lib/egl/libGLE
S_emulation.so" not found
09-21 22:35:46.316  2134  2163 D vndksupport: Loading /vendor/lib/egl/libEGL_emulation.so from current namespace instead of sphal namespace.
09-21 22:35:46.351  2134  2163 D libEGL  : loaded /vendor/lib/egl/libEGL_emulation.so
09-21 22:35:46.354  2134  2163 D vndksupport: Loading /vendor/lib/egl/libGLESv1_CM_emulation.so from current namespace instead of sphal name
space.
09-21 22:35:46.403  2134  2163 D libEGL  : loaded /vendor/lib/egl/libGLESv1_CM_emulation.so
09-21 22:35:46.432  2134  2134 W com.revo.evabs: Accessing hidden method Landroid/view/View;->computeFitSystemWindows(Landroid/graphics/Rect
;Landroid/graphics/Rect;)Z (light greylist, reflection)
09-21 22:35:46.432  2134  2134 W com.revo.evabs: Accessing hidden method Landroid/view/ViewGroup;->makeOptionalFitsSystemWindows()V (light g
reylist, reflection)
09-21 22:35:46.450  2134  2163 D vndksupport: Loading /vendor/lib/egl/libGLESv2_emulation.so from current namespace instead of sphal namespa
ce.
09-21 22:35:46.451  2134  2163 D libEGL  : loaded /vendor/lib/egl/libGLESv2_emulation.so
09-21 22:35:46.483  2134  2134 W com.revo.evabs: Accessing hidden method Landroid/graphics/FontFamily;-><init>()V (light greylist, reflectio
n)
09-21 22:35:46.483  2134  2134 W com.revo.evabs: Accessing hidden method Landroid/graphics/FontFamily;->addFontFromAssetManager(Landroid/con
tent/res/AssetManager;Ljava/lang/String;IZIII[Landroid/graphics/fonts/FontVariationAxis;)Z (light greylist, reflection)
09-21 22:35:46.483  2134  2134 W com.revo.evabs: Accessing hidden method Landroid/graphics/FontFamily;->addFontFromBuffer(Ljava/nio/ByteBuff
er;I[Landroid/graphics/fonts/FontVariationAxis;II)Z (light greylist, reflection)
09-21 22:35:46.483  2134  2134 W com.revo.evabs: Accessing hidden method Landroid/graphics/FontFamily;->freeze()Z (light greylist, reflectio
n)
09-21 22:35:46.483  2134  2134 W com.revo.evabs: Accessing hidden method Landroid/graphics/FontFamily;->abortCreation()V (light greylist, re
```

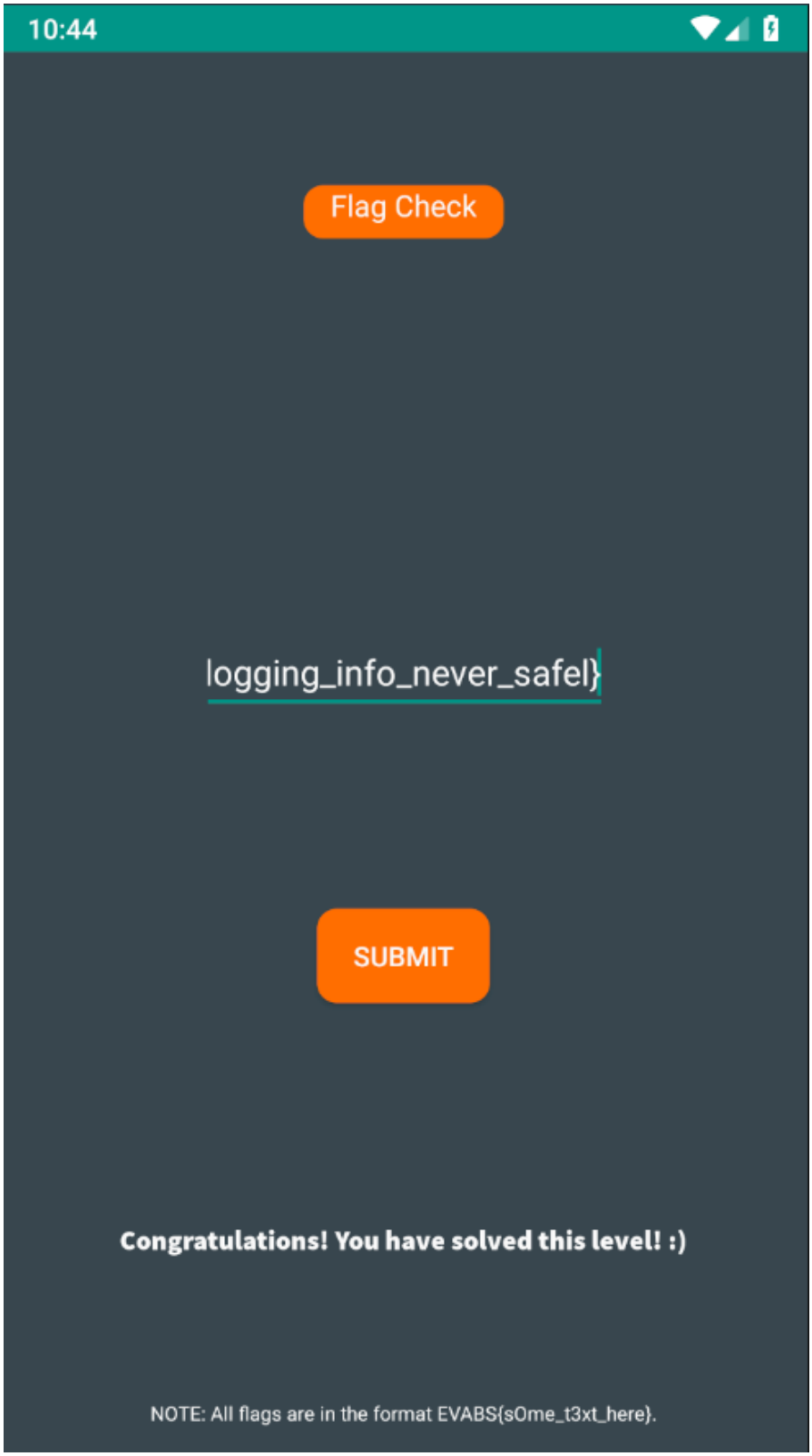
On the EVABS app, press the LOG THE KEY button as it mimicked the log input function in a real situation. By pressing that button, it will log the flag.



On the Mobexler terminal, as you pressed the button, it'll log the flag as debug log so that's our first flag!

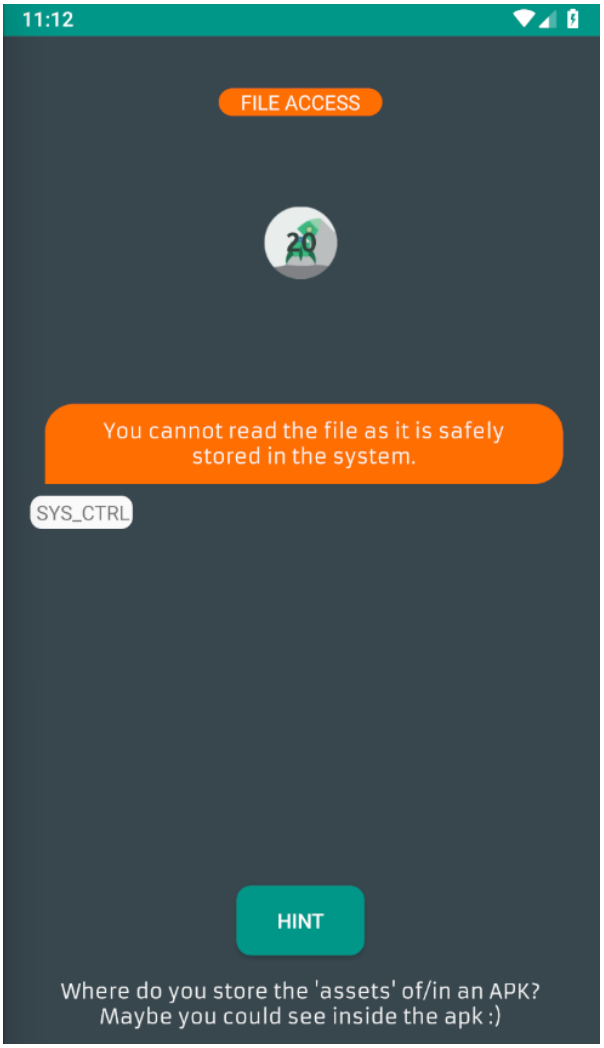
```
09-21 22:35:48.521  2134  2134 W ActivityThread: handleWindowVisibility: no activity for token android.os.BinderProxy@ff141b3
09-21 22:35:51.494  2134  2165 E eglCodecCommon: goldfish_dma_create_region: could not obtain fd to device! fd -1 errno=2
09-21 22:37:17.668  2134  2134 W ActivityThread: handleWindowVisibility: no activity for token android.os.BinderProxy@f6f4078
09-21 22:37:45.280  2134  2134 W ActivityThread: handleWindowVisibility: no activity for token android.os.BinderProxy@bb7162c
09-21 22:42:33.958  2134  2134 D ** SYS_CTRL **: : EVABS{logging_info_never_safel}
```

Copy the flag, open the EVABS FLAG CHECK section and paste it there like shown below. Then press the SUBMIT button to confirm your flag. This step applied on all challenges level as part of confirming the flag process so on further walkthrough part, I'll skip this.



CHALLENGE LEVEL 2

Next one, is File Access challenge as per stated on top of the pages. So, in common practice of developing an app, as a user can't read the app file as it's safely stored in the system.



On understanding this challenge, the hint stated that the assets of an app may be stored within the app file or APK.

```
Mobexler@Mobexler  ~  objection -g "com.revo.evabs" explore
Using USB device `Google Pixel 3 XL`
Agent injected and responds ok!

     _   _         _   _
 ___| |_|_|___ ___| |_|_|___ ___
| . | . | | -_|  _|  _| | . |   |
|___|___| |___|___|_| |_|___|_|_|
      |___|(object)inject(ion) v1.9.6

     Runtime Mobile Exploration
        by: @leonjza from @sensepost

[tab] for command suggestions
com.revo.evabs on (Android: 9) [usb] #
```

Entering directory within the APK can achieved by using Objection. One of the mobile exploration's tools made using Frida. Use *objection -g "app.package.name" explore*

```
[tab] for command suggestions
com.revo.evabs on (Android: 9) [usb] # ls
Type    Last Modified    Read    Write    Hidden    Size    Name
------  ---------------  ------  -------  --------  ------  ------

Readable: True  Writable: True
com.revo.evabs on (Android: 9) [usb] # cd ..
/data/user/0/com.revo.evabs
com.revo.evabs on (Android: 9) [usb] # ls
Type       Last Modified            Read    Write    Hidden    Size     Name
---------  -----------------------  ------  -------  --------  -------  ------------
Directory  2020-08-24 01:37:04 GMT  True    True     False     4.0 KiB  shared_prefs
Directory  2020-09-21 07:03:16 GMT  True    True     False     4.0 KiB  files
Directory  2020-08-24 01:36:34 GMT  True    False    False     4.0 KiB  lib
Directory  2020-08-24 01:36:34 GMT  True    True     False     4.0 KiB  code_cache
Directory  2020-08-24 01:36:34 GMT  True    True     False     4.0 KiB  cache

Readable: True  Writable: True
com.revo.evabs on (Android: 9) [usb] # env

Name                    Path
----------------------  ---------------------------------------------------------
cacheDirectory          /data/user/0/com.revo.evabs/cache
codeCacheDirectory      /data/user/0/com.revo.evabs/code_cache
externalCacheDirectory  /storage/emulated/0/Android/data/com.revo.evabs/cache
filesDirectory          /data/user/0/com.revo.evabs/files
obbDir                  /storage/emulated/0/Android/obb/com.revo.evabs
packageCodePath         /data/app/com.revo.evabs-QlKhXiqiffZ9ZVLfPab54g==/base.apk
com.revo.evabs on (Android: 9) [usb] # cd /data/app/com.revo.evabs-QlKhXiqiffZ9ZVLfPab54g==/
/data/app/com.revo.evabs-QlKhXiqiffZ9ZVLfPab54g==/
com.revo.evabs on (Android: 9) [usb] # ls
Type       Last Modified            Read    Write    Hidden    Size     Name
---------  -----------------------  ------  -------  --------  -------  --------
Directory  2020-09-08 06:25:54 GMT  False   False    False     4.0 KiB  oat
Directory  2020-08-24 01:36:34 GMT  True    False    False     4.0 KiB  lib
File       2020-08-24 01:36:34 GMT  True    False    False     7.9 MiB  base.apk

Readable: True  Writable: False
com.revo.evabs on (Android: 9) [usb] # file download base.apk base.apk
Downloading /data/app/com.revo.evabs-QlKhXiqiffZ9ZVLfPab54g==//base.apk to base.apk
Streaming file from device...
Writing bytes to destination...
Successfully downloaded /data/app/com.revo.evabs-QlKhXiqiffZ9ZVLfPab54g==//base.apk to base.apk
com.revo.evabs on (Android: 9) [usb] #
```

Once gained access within the app file, list any directory and files in it to locate the APK file. Use command *env* to list out environment of the app package. As you can see, I managed to find the APK file and download it using this command;

file download filename.apk filename.apk

Second name of the app used as the location where to save in the Mobexler path by using its current name.

```
Mobexler@Mobexler  ~  apktool d base.apk                                                    ✓ 65 09:06:34
I: Using Apktool 2.4.1 on base.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: /home/mobexler/.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...
Mobexler@Mobexler  ~  ls                                                                    ✓ 67 09:07:28
Android      base                  base.apk            Desktop  Documents  GNUstep  Music         Pictures  results  Templates  tool_update
AndroidZone  base16-xfce4-terminal BurpSuiteCommunity  dist     Downloads  iOSZone  node_modules  Public    SecZone  tools      Videos
Mobexler@Mobexler  ~  |                                                                      ✓ 68 09:07:40
```

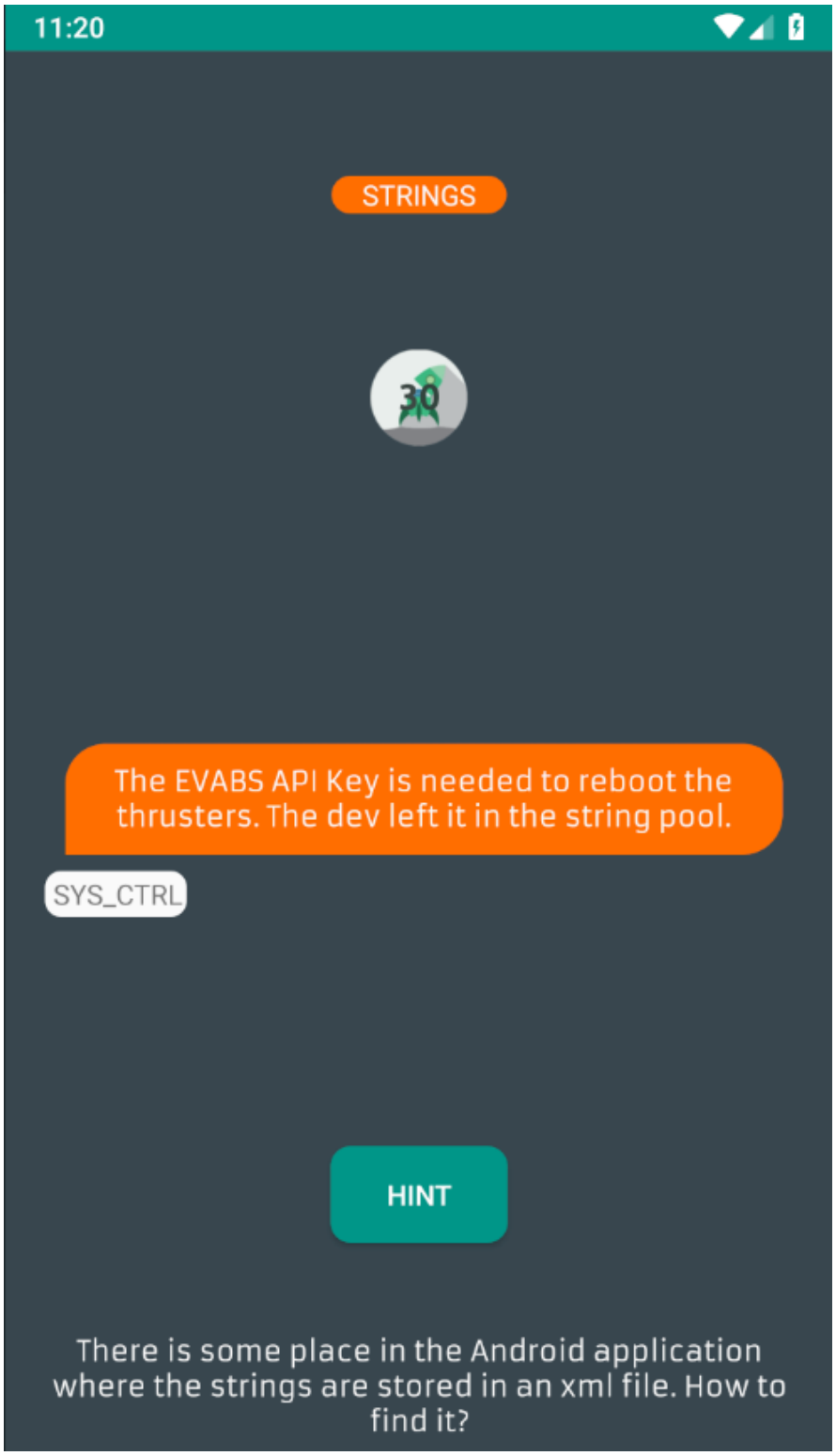
After downloaded, it obviously will save the file as APK format (.apk) because it is an app file. Using this command, *apktool -d filename.apk* to dissemble the app file into readable and accessible folder.

```
Mobexler@Mobexler  ~/base  ls
AndroidManifest.xml  apktool.yml  assets  lib  original  res  smali  unknown
Mobexler@Mobexler  ~/base  cd assets
Mobexler@Mobexler  ~/base/assets  ls
fonts  secrets
Mobexler@Mobexler  ~/base/assets  cat secrets
EVABS{fil3s_!n_ass3ts_ar3_eas!ly_hackabl3}
```

Remember what hint stated before? Try to look for an ‘assets’. In the app file folder just now, list all the contents until found the **assets** folder. In that folder, there is one text file named **secrets.** Within that file contained the flag that have been looking for.

CHALLENGE LEVEL 3

Challenge level 3 is about Strings whereby it can be any strings located in the app file. For this challenge, it needs us to find the API Key.



Based on the hint, typically this type of strings which is the one that critical and sensitive will be stored in the XML file.

```
Mobexler@Mobexler  ~  apktool d base.apk                                                    ✓ 65 09:06:34
I: Using Apktool 2.4.1 on base.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: /home/mobexler/.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...
Mobexler@Mobexler  ~  ls                                                                    ✓ 67 09:07:28
Android      base                  base.apk            Desktop  Documents  GNUstep  Music         Pictures  results  Templates  tool_update
AndroidZone  base16-xfce4-terminal BurpSuiteCommunity  dist     Downloads  iOSZone  node_modules  Public    SecZone  tools      Videos
Mobexler@Mobexler  ~  |                                                                     ✓ 68 09:07:40
```

Initially I already downloaded the app file from the app package then dissemble it into app folder, so may skip this step. If not, you may dissemble the app file by using this command, *apktool -d filename.apk*

```
Mobexler@Mobexler  ~  cd base                                                               ✓ 68 09:07:49
Mobexler@Mobexler  ~/base  ls                                                               ✓ 69 09:07:52
ndroidManifest.xml  apktool.yml  assets  lib  original  res  smali  unknown
Mobexler@Mobexler  ~/base  cd res                                                           ✓ 70 09:07:53
Mobexler@Mobexler  ~/base/res  ls                                                           ✓ 71 09:07:57
nim                        drawable-watch-v20  mipmap-xxxhdpi    values-en-rGB  values-in         values-ml      values-sk       values-v21
nimator-v21                drawable-xhdpi      raw               values-en-rIN  values-is         values-mn      values-sl       values-v22
nim-v21                    drawable-xxhdpi     values            values-es      values-it         values-mr      values-sq       values-v23
olor                       drawable-xxxhdpi    values-af         values-es-rUS  values-iw         values-ms      values-sr       values-v24
olor-v23                   font                values-am         values-et      values-ja         values-my      values-sv       values-v25
rawable                    layout              values-ar         values-eu      values-ka         values-nb      values-sw       values-v26
rawable-anydpi-v21         layout-sw600dp      values-az         values-fa      values-kk         values-ne      values-sw600dp  values-vi
rawable-hdpi               layout-v16          values-be         values-fi      values-km         values-night   values-ta       values-w820dp
rawable-ldpi               layout-v17          values-bg         values-fr      values-kn         values-nl      values-te       values-watch-v20
rawable-ldrtl-hdpi-v17     layout-v21          values-bn         values-fr-rCA  values-ko         values-pa      values-th       values-xlarge
rawable-ldrtl-mdpi-v17     layout-v22          values-bs         values-gl      values-ky         values-pl      values-tl       values-zh-rCN
rawable-ldrtl-xhdpi-v17    layout-v26          values-b+sr+Latn  values-gu      values-land       values-port    values-tr       values-zh-rHK
rawable-ldrtl-xxhdpi-v17   menu                values-ca         values-h720dp  values-large      values-pt      values-uk       values-zh-rTW
rawable-ldrtl-xxxhdpi-v17  mipmap-anydpi-v26   values-cs         values-hdpi    values-ldltr-v21  values-pt-rBR  values-ur       values-zu
rawable-mdpi               mipmap-hdpi         values-da         values-hi      values-lo         values-pt-rPT  values-uz
rawable-v21                mipmap-mdpi         values-de         values-hr      values-lt         values-ro      values-v16
rawable-v23                mipmap-xhdpi        values-el         values-hu      values-lv         values-ru      values-v17
rawable-v24                mipmap-xxhdpi       values-en-rAU     values-hy      values-mk         values-si      values-v18
Mobexler@Mobexler  ~/base/res  cd values                                                    ✓ 72 09:07:58
Mobexler@Mobexler  ~/base/res/values  ls                                                    ✓ 73 09:08:13
rrays.xml  attrs.xml  bools.xml  colors.xml  dimens.xml  drawables.xml  ids.xml  integers.xml  public.xml  strings.xml  styles.xml
Mobexler@Mobexler  ~/base/res/values  cat strings.xml                                       ✓ 74 09:08:14
```

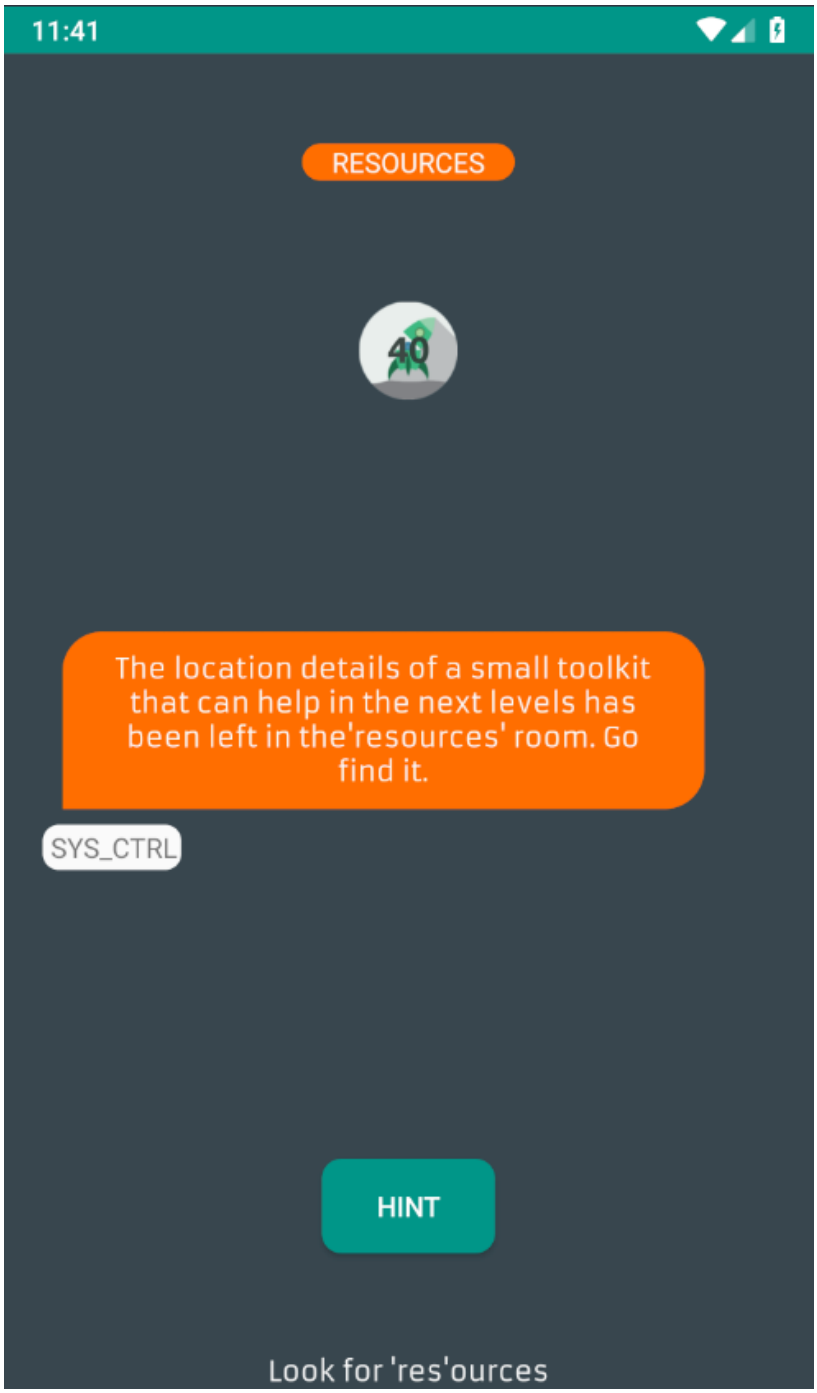
Well, there are lot of folders and files within a single app folder so to know fast where usually XML file stored, you may Google-Fu to know where to the exact directory. I did the same and for this case, it located in values folder from res directory.

```
    <string name="section_format">Hello World from section: %1$d</string>
    <string name="status_bar_notification_info_overflow">999+</string>
    <string name="the_evabs_api_key">EVABS{saf3ly_st0red_in_Strings?}</string>
    <string name="title_activity_home">Home</string>
    <string name="title_activity_launch">Launch</string>
    <string name="title_activity_login">Sign in</string>
    <string name="title_activity_splash">Splash</string>
    <string name="title_activity_test">Test</string>
</resources>
Mobexler@Mobexler  ~/base/res/values  |
```

As per hint stated, the API Key stored in the XML file type plus with this challenge all about strings. So, I found the file name **strings.xml** which contain API Key that will lead to the flag for this challenge.

CHALLENGE LEVEL 4

This challenge is about Resources, whereby it is a location for any resources used will be put together in a cumulated folder.



Since this challenge about locating resources, then our hint that may lead to the flag can be found in the ‘resources’ room. Even the hint stated, look for ‘res’ources. Why is ist ‘res’ources? In common mobile app development, there always be one folder named **res** as location of all resources used.

```
Mobexler@Mobexler  ~  cd base/res                                                                  ✔ 113 09:22:53
Mobexler@Mobexler  ~/base/res  ls                                                                  ✔ 114 09:23:08
anim                    drawable-watch-v20  mipmap-xxxhdpi    values-en-rGB  values-in        values-ml      values-sk       values-v21
animator-v21            drawable-xhdpi      raw               values-en-rIN  values-is        values-mn      values-sl       values-v22
anim-v21                drawable-xxhdpi     values            values-es      values-it        values-mr      values-sq       values-v23
color                   drawable-xxxhdpi    values-af         values-es-rUS  values-iw        values-ms      values-sr       values-v24
color-v23               font                values-am         values-et      values-ja        values-my      values-sv       values-v25
drawable                layout              values-ar         values-eu      values-ka        values-nb      values-sw       values-v26
drawable-anydpi-v21     layout-sw600dp      values-az         values-fa      values-kk        values-ne      values-sw600dp  values-vi
drawable-hdpi           layout-v16          values-be         values-fi      values-km        values-night   values-ta       values-w820dp
drawable-ldpi           layout-v17          values-bg         values-fr      values-kn        values-nl      values-te       values-watch-v20
drawable-ldrtl-hdpi-v17 layout-v21          values-bn         values-fr-rCA  values-ko        values-pa      values-th       values-xlarge
drawable-ldrtl-mdpi-v17 layout-v22          values-bs         values-gl      values-ky        values-pl      values-tl       values-zh-rCN
drawable-ldrtl-xhdpi-v17 layout-v26         values-b+sr+Latn  values-gu      values-land      values-port    values-tr       values-zh-rHK
drawable-ldrtl-xxhdpi-v17 menu              values-ca         values-h720dp  values-large     values-pt      values-uk       values-zh-rTW
drawable-ldrtl-xxxhdpi-v17 mipmap-anydpi-v26 values-cs        values-hdpi    values-ldltr-v21 values-pt-rBR  values-ur       values-zu
drawable-mdpi           mipmap-hdpi         values-da         values-hi      values-lo        values-pt-rPT  values-uz
drawable-v21            mipmap-mdpi         values-de         values-hr      values-lt        values-ro      values-v16
drawable-v23            mipmap-xhdpi        values-el         values-hu      values-lv        values-ru      values-v17
drawable-v24            mipmap-xxhdpi       values-en-rAU     values-hy      values-mk        values-si      values-v18
Mobexler@Mobexler  ~/base/res  grep -Rw ~/base/res/ -e 'Toolkit' -e 'toolkit'                      ✔ 115 09:23:17

/home/mobexler/base/res/layout-v16/activity_res_raw.xml:            <TextView android:textColor="@color/colorWhite" android:id="@id/textViewstrings" and
roid:background="@drawable/chatbox" android:layout_width="wrap_content" android:layout_height="wrap_content" android:layout_marginLeft="10.0dip" android
:layout_marginRight="10.0dip" android:text="The location details of a small toolkit that can help in the next levels has been left in the'resources' roo
m. Go find it." android:fontFamily="@font/armata" />
/home/mobexler/base/res/raw/link.txt:# This Toolkit will help you fix EVABS
/home/mobexler/base/res/layout-v17/activity_res_raw.xml:            <TextView android:textColor="@color/colorWhite" android:id="@id/textViewstrings" and
roid:background="@drawable/chatbox" android:layout_width="wrap_content" android:layout_height="wrap_content" android:layout_marginLeft="10.0dip" android
:layout_marginRight="10.0dip" android:text="The location details of a small toolkit that can help in the next levels has been left in the'resources' roo
m. Go find it." android:fontFamily="@font/armata" android:textAlignment="center" />
/home/mobexler/base/res/layout/activity_res_raw.xml:            <TextView android:textColor="@color/colorWhite" android:id="@id/textViewstrings" android
:background="@drawable/chatbox" android:layout_width="wrap_content" android:layout_height="wrap_content" android:layout_marginLeft="10.0dip" android:lay
out_marginRight="10.0dip" android:text="The location details of a small toolkit that can help in the next levels has been left in the'resources' room. G
o find it." />
Mobexler@Mobexler  ~/base/res                                                                      ✔ 116 09:23:23
```

By using the same app folder that has been dissembled before, list all the directory inside of the **res** folder. As portrayed, there are bunch of folders only within the **res** folder, so I decided to use the command follows to get some extra hunches;

Grep -Rw /folder/location/ -e 'Word' -e 'Search'

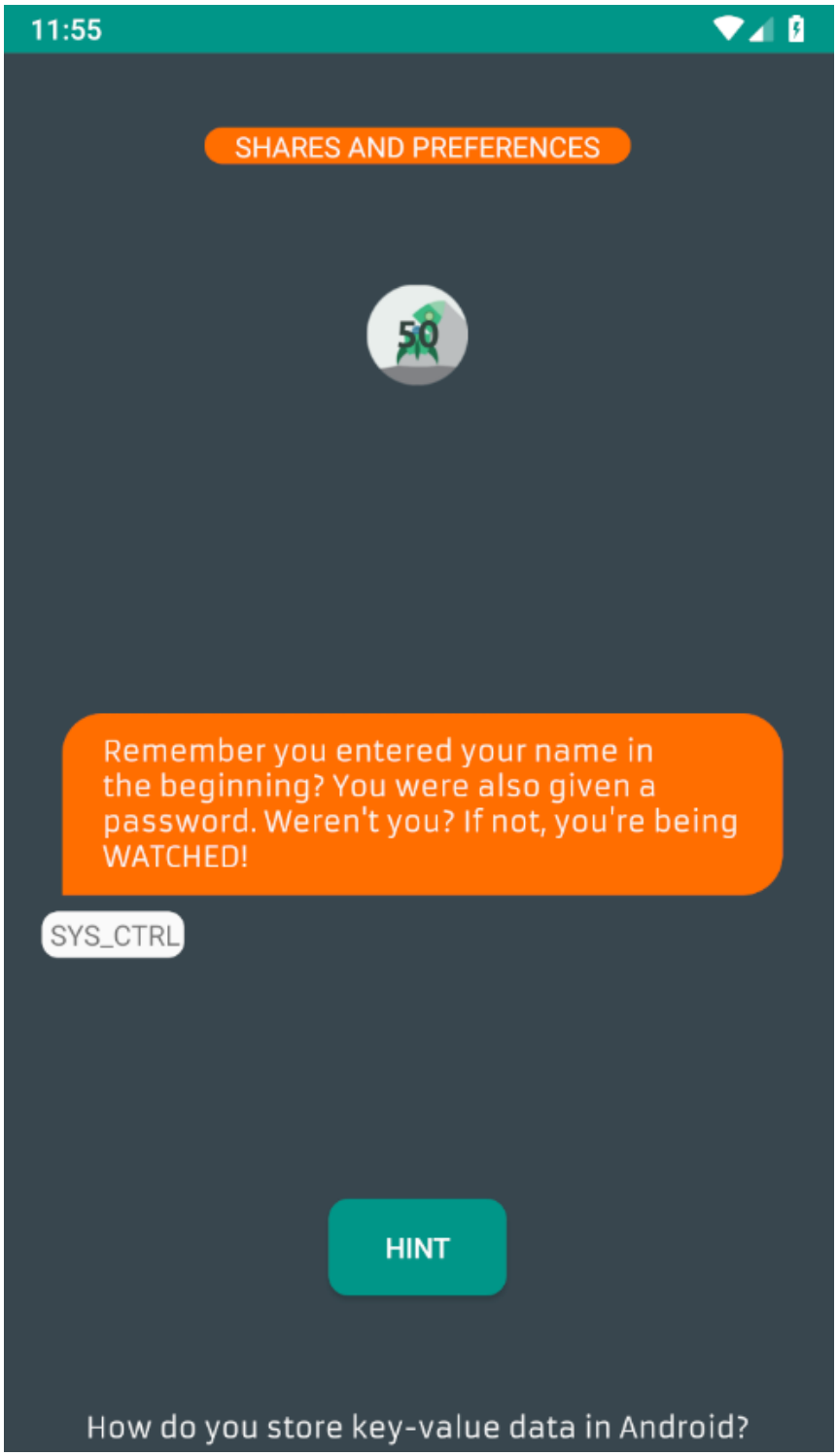
Because based on this challenge description, it mentioned something related to the toolkit. Then, it clued me this text file named link.txt contained text shown in above picture.

```
Mobexler@Mobexler  ~/base/res  cd raw/
Mobexler@Mobexler  ~/base/res/raw  ls
banner.png  law.mp3  link.txt
Mobexler@Mobexler  ~/base/res/raw  cat link.txt
www.github.com/abhi-r3v0/Adhrit

# This Toolkit will help you fix EVABS

EVABS{th!s_plac3_is_n0t_as_s3cur3_as_it_l00ks}
Mobexler@Mobexler  ~/base/res/raw  |
```

Without hesitation, I then went to the directory of that text file and cat the content. It lead me to the right location as that text file contain the flag as well.

CHALLENGE LEVEL 5

Challenge level 5 focused on Shares and Preferences of particular mobile app packages. Description of this challenge stated that this challenge kinda related to password storage whereby it is really you key in the password or not.



While hint trigger you how certain critical and sensitive data stored in in typical android app development. In this case, how password stored.

```
Mobexler@Mobexler  ~  objection -g "com.revo.evabs" explore
Using USB device `Google Pixel 3 XL`
Agent injected and responds ok!

     _   _         _   _
 ___| |_|_|___ ___| |_|_|___ ___
| . | . | | -_|  _|  _| | . |   |
|___|___| |___|___|_| |_|___|_|_|
      |___|(object)inject(ion) v1.9.6

     Runtime Mobile Exploration
        by: @leonjza from @sensepost

[tab] for command suggestions
com.revo.evabs on (Android: 9) [usb] #
```

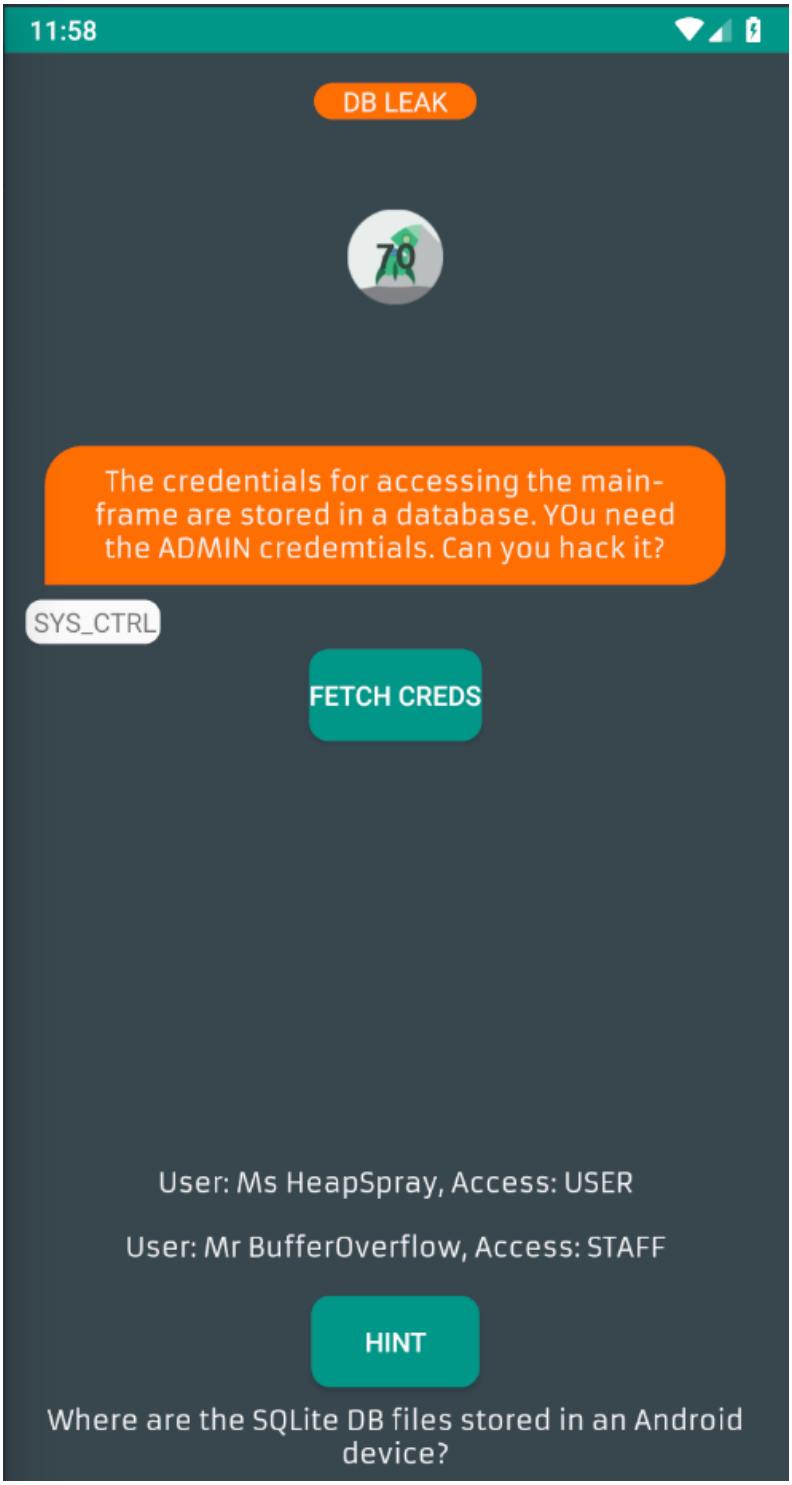
As usual, fire up back objection by using command *objection -g 'app.package.name' explore* to start exploration and search of shared & preferences folder.

```
com.revo.evabs on (Android: 9) [usb] # ls
Type        Last Modified            Read    Write    Hidden    Size     Name
---------   ----------------------   ------  -------  --------  -------  ------------
Directory   2020-08-24 01:37:04 GMT  True    True     False     4.0 KiB  shared_prefs
Directory   2020-09-21 07:03:16 GMT  True    True     False     4.0 KiB  files
Directory   2020-08-24 01:36:34 GMT  True    False    False     4.0 KiB  lib
Directory   2020-08-24 01:36:34 GMT  True    True     False     4.0 KiB  code_cache
Directory   2020-08-24 01:36:34 GMT  True    True     False     4.0 KiB  cache

Readable: True  Writable: True
com.revo.evabs on (Android: 9) [usb] # cd shared_prefs
/data/user/0/com.revo.evabs/shared_prefs
com.revo.evabs on (Android: 9) [usb] # ls
Type    Last Modified            Read    Write    Hidden    Size     Name
------  ----------------------   ------  -------  --------  -------  --------------
File    2020-08-24 01:36:39 GMT  True    True     False     118.0 B  PREFERENCE.xml
File    2020-08-24 01:37:04 GMT  True    True     False     191.0 B  DETAILS.xml

Readable: True  Writable: True
com.revo.evabs on (Android: 9) [usb] # file cat PREFERENCE.xml
Downloading /data/user/0/com.revo.evabs/shared_prefs/PREFERENCE.xml to /tmp/tmpxsdovrur.file
Streaming file from device...
Writing bytes to destination...
Successfully downloaded /data/user/0/com.revo.evabs/shared_prefs/PREFERENCE.xml to /tmp/tmpxsdovrur.file
====
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <boolean name="isFirstRun" value="false" />
</map>
====
com.revo.evabs on (Android: 9) [usb] # file cat DETAILS.xml
Downloading /data/user/0/com.revo.evabs/shared_prefs/DETAILS.xml to /tmp/tmpea92gufv.file
Streaming file from device...
Writing bytes to destination...
Successfully downloaded /data/user/0/com.revo.evabs/shared_prefs/DETAILS.xml to /tmp/tmpea92gufv.file
====
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <string name="password">EVABS{shar3d_pr3fs_c0uld_be_c0mpromiz3ds}</string>
    <string name="username">Prof</string>
</map>
====
com.revo.evabs on (Android: 9) [usb] #
```

Once in the app file, there is a folder named **shared_prefs**, and inside it there are two file that may contain precious data. After reading both files using command, *file cat filename.xml* then for second XML file, it contains the flag that I am looking for.

CHALLENGE LEVEL 6

Ok here now is a database related challenge. For this challenge it wants us to hack the ADMIN credentials that stored in the database within the app file.



To begin with, press the FETCH CREDS button to load in the admin credentials which is in this case is the STAFF access privilege. In order to know what's next since there's no entry point for SQLi, the hint stated that to find where the SQLite DB files stored. That's enough clue for me to further on this challenge.

```
Mobexler@Mobexler  ~  objection -g "com.revo.evabs" explore
Using USB device `Google Pixel 3 XL`
Agent injected and responds ok!

     _   _         _   _
 ___| |_|_|___ ___| |_|_|___ ___
| . | . | | -_|  _|  _| | . |   |
|___|___| |___|___|_| |_|___|_|_|
      |___|(object)inject(ion) v1.9.6

     Runtime Mobile Exploration
        by: @leonjza from @sensepost

[tab] for command suggestions
com.revo.evabs on (Android: 9) [usb] # ls
Type    Last Modified    Read    Write    Hidden    Size    Name
------  ---------------  ------  -------  --------  ------  ------

Readable: True  Writable: True
com.revo.evabs on (Android: 9) [usb] # cd ..
/data/user/0/com.revo.evabs
com.revo.evabs on (Android: 9) [usb] # ls
Type        Last Modified            Read    Write    Hidden    Size     Name
---------   -----------------------  ------  -------  --------  -------  ------------
Directory   2020-08-24 01:37:04 GMT  True    True     False     4.0 KiB  shared_prefs
Directory   2020-09-21 07:03:16 GMT  True    True     False     4.0 KiB  files
Directory   2020-08-24 01:36:34 GMT  True    False    False     4.0 KiB  lib
Directory   2020-09-22 03:57:14 GMT  True    True     False     4.0 KiB  databases
Directory   2020-08-24 01:36:34 GMT  True    True     False     4.0 KiB  code_cache
Directory   2020-08-24 01:36:34 GMT  True    True     False     4.0 KiB  cache
```

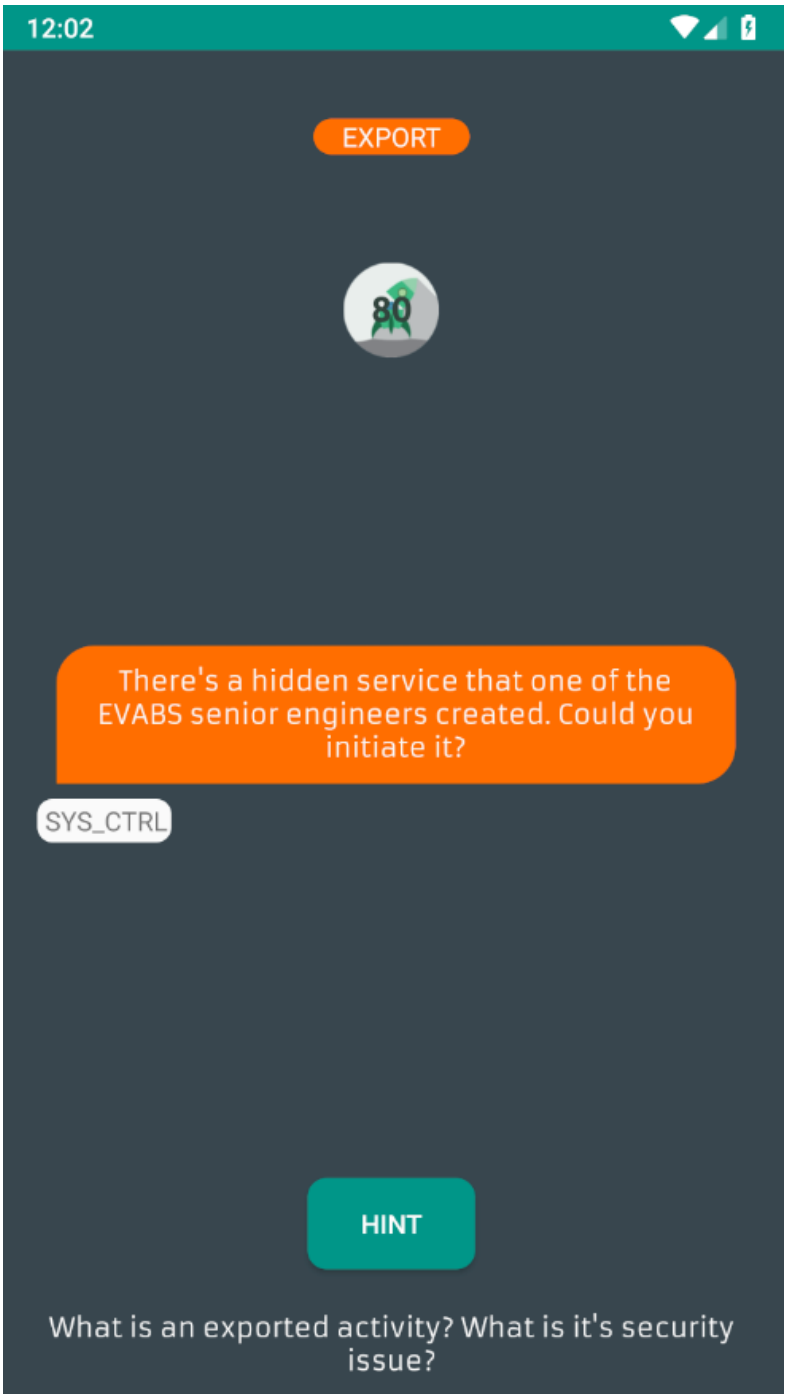
As usual, command *objection -g "app.package.name" explore* to dive in the app file. After listed out directory, you can see that there's a directory named **databases.**

```
com.revo.evabs on (Android: 9) [usb] # cd databases
/data/user/0/com.revo.evabs/databases
com.revo.evabs on (Android: 9) [usb] # ls
Type    Last Modified            Read    Write    Hidden    Size      Name
------  -----------------------  ------  -------  --------  --------  -------------------
File    2020-09-22 03:57:14 GMT  True    True     False     32.0 KiB  MAINFRAME_ACCESS-shm
File    2020-09-22 03:57:14 GMT  True    True     False     4.0 KiB   MAINFRAME_ACCESS
File    2020-09-22 03:57:14 GMT  True    True     False     40.3 KiB  MAINFRAME_ACCESS-wal

Readable: True  Writable: True
com.revo.evabs on (Android: 9) [usb] # file cat MAINFRAME_ACCESS
Downloading /data/user/0/com.revo.evabs/databases/MAINFRAME_ACCESS to /tmp/tmpal6h94wl.file
Streaming file from device...
Writing bytes to destination...
Successfully downloaded /data/user/0/com.revo.evabs/databases/MAINFRAME_ACCESS to /tmp/tmpal6h94wl.file
====
SQLite format 3@  [illegible]
[illegible]===
com.revo.evabs on (Android: 9) [usb] # file cat MAINFRAME_ACCESS-shm
Downloading /data/user/0/com.revo.evabs/databases/MAINFRAME_ACCESS-shm to /tmp/tmpznk2h2kw.file
Streaming file from device...
Writing bytes to destination...
Successfully downloaded /data/user/0/com.revo.evabs/databases/MAINFRAME_ACCESS-shm to /tmp/tmpznk2h2kw.file
====
[illegible]
====
com.revo.evabs on (Android: 9) [usb] # file cat MAINFRAME_ACCESS-wal
Downloading /data/user/0/com.revo.evabs/databases/MAINFRAME_ACCESS-wal to /tmp/tmplnh0bm5q.file
Streaming file from device...
Writing bytes to destination...
Successfully downloaded /data/user/0/com.revo.evabs/databases/MAINFRAME_ACCESS-wal to /tmp/tmplnh0bm5q.file
====
7[illegible]n4>s[illegible],[illegible]SQLite format 3@  [illegible]
[illegible]-[illegible]tableandroid_metadataandroid_metadata[illegible]CREATE TABLE android_metadata (locale TEXT)[illegible]?[illegible]`
[illegible]E^TS
[illegible]en_US[illegible]
[illegible]SQLite format 3@  [illegible]
[illegible]OOV[illegible]
        tableCREDSCREDS[illegible]CREATE TABLE_CREDS(admin VARCHAR, pass VARCHAR, access VARCHAR)W[illegible]-[illegible]tableandroid_me
id_metadata (locale TEXT)[illegible]49[q][illegible]h{
[illegible]
0~!(
[illegible]r.l33tEVABS{sqlite_is_not_safe}EADMIN[illegible]
[illegible]r BufferOverflow0xNotSecureSQLite_STAFF*[illegible]r.l33tEVABS{sqlite_is_not_safe}EADMIN[illegible]b[illegible][}
```

In the **databases** directory, there are 3 files which all related to mainframe access mentioned earlier in the front page of this challenges. Using command, *file cat filename.file* on all files in that folder to read the content then I found that on file **MAINFRAME_ACCESS.wal** was the right databases content. Plus, the flag for this challenge also can be found in this file.

CHALLENGE LEVEL 7

Comes to mediocre part for this lab here and afterward, not so hard but tricky and lengthy. This challenge is about export function intentionally made for debugging during development or purposely created. Furthermore, it wants us to initiate that function thru app. Let's do it!



Based on the hint given, its mentioned about common vulnerability regarding exported activity in a mobile app. Frankly speaking this was my first time counter this issue so then I take some time to understand it before proceed[1].

[1] https://appsec-labs.com/portal/hacking-android-apps-through-exposed-components/

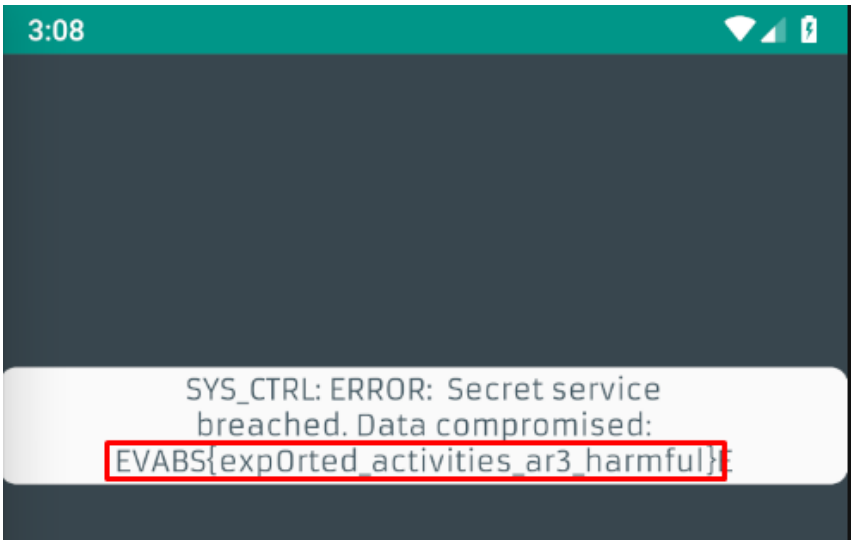
```
 Mobexler@Mobexler  ~  ls
Android      base16-xfce4-terminal  Desktop    Downloads  Music         Public   Templates    Videos
AndroidZone  base.apk               dist       GNUstep    node_modules  results  tools
base         BurpSuiteCommunity     Documents  iOSZone    Pictures      SecZone  tool_update
 Mobexler@Mobexler  ~  cd base
 Mobexler@Mobexler  ~/base  ls
AndroidManifest.xml  apktool.yml  assets  lib  original  res  smali  unknown
 Mobexler@Mobexler  ~/base  cat AndroidManifest.xml
<?xml version="1.0" encoding="utf-8" standalone="no"?><manifest xmlns:android="http://schemas.android.com/apk/res/
.evabs">
    <uses-permission android:name="android.permission.INTERNET"/>
    <application android:allowBackup="false" android:debuggable="true" android:icon="@mipmap/ic_launcher" android:
ndroid:roundIcon="@mipmap/ic_launcher_round" android:supportsRtl="true" android:theme="@style/AppTheme">
        <activity android:exported="true" android:name="com.revo.evabs.ExportedActivity"/>
        <activity android:name="com.revo.evabs.Frida1"/>
        <activity android:name="com.revo.evabs.FileRead"/>
        <activity android:name="com.revo.evabs.DebugMe"/>
        <activity android:name="com.revo.evabs.Welcome"/>
        <activity android:name="com.revo.evabs.ChallengeList" android:parentActivityName="com.revo.evabs.Launch">
            <meta-data android:name="android.support.PARENT_ACTIVITY" android:value=".Launch"/>
        </activity>
        <activity android:name="com.revo.evabs.ExportedInfo"/>
        <activity android:name="com.revo.evabs.SmaliInject"/>
        <activity android:name="com.revo.evabs.StringsSecrets"/>
        <activity android:name="com.revo.evabs.SharedBreach"/>
        <activity android:name="com.revo.evabs.Decode"/>
        <activity android:name="com.revo.evabs.BadComm"/>
        <activity android:name="com.revo.evabs.DBLeak"/>
        <activity android:name="com.revo.evabs.CustomAccess">
            <intent-filter>
                <action android:name="com.revo.evabs.action.SENSOR_KEY"/>
                <category android:name="android.intent.category.DEFAULT"/>
                <data android:mimeType="text/plain"/>
            </intent-filter>
        </activity>
        <activity android:name="com.revo.evabs.Res_raw"/>
        <meta-data android:name="preloaded_fonts" android:resource="@array/preloaded_fonts"/>
        <activity android:name="com.revo.evabs.Welcome0"/>
        <activity android:name="com.revo.evabs.Splash">
```

This exported activity issues commonly found in the **AndroiManifest.xml** file and I started locating it using the app folder extracted previously. After some time understand this vulnerability, the exported function must be set to *true* in order to initiate the activity declared within.

```
 Mobexler@Mobexler  ~  adb shell am start -n com.revo.evabs/com.revo.evabs.ExportedActivity
Starting: Intent { cmp=com.revo.evabs/.ExportedActivity }
 Mobexler@Mobexler  ~
```

Since it found to be already set to *true* (because yeah this lab is a vulnerable lab isn't) then I used command as follows to execute the exported activity outside of the app environment;

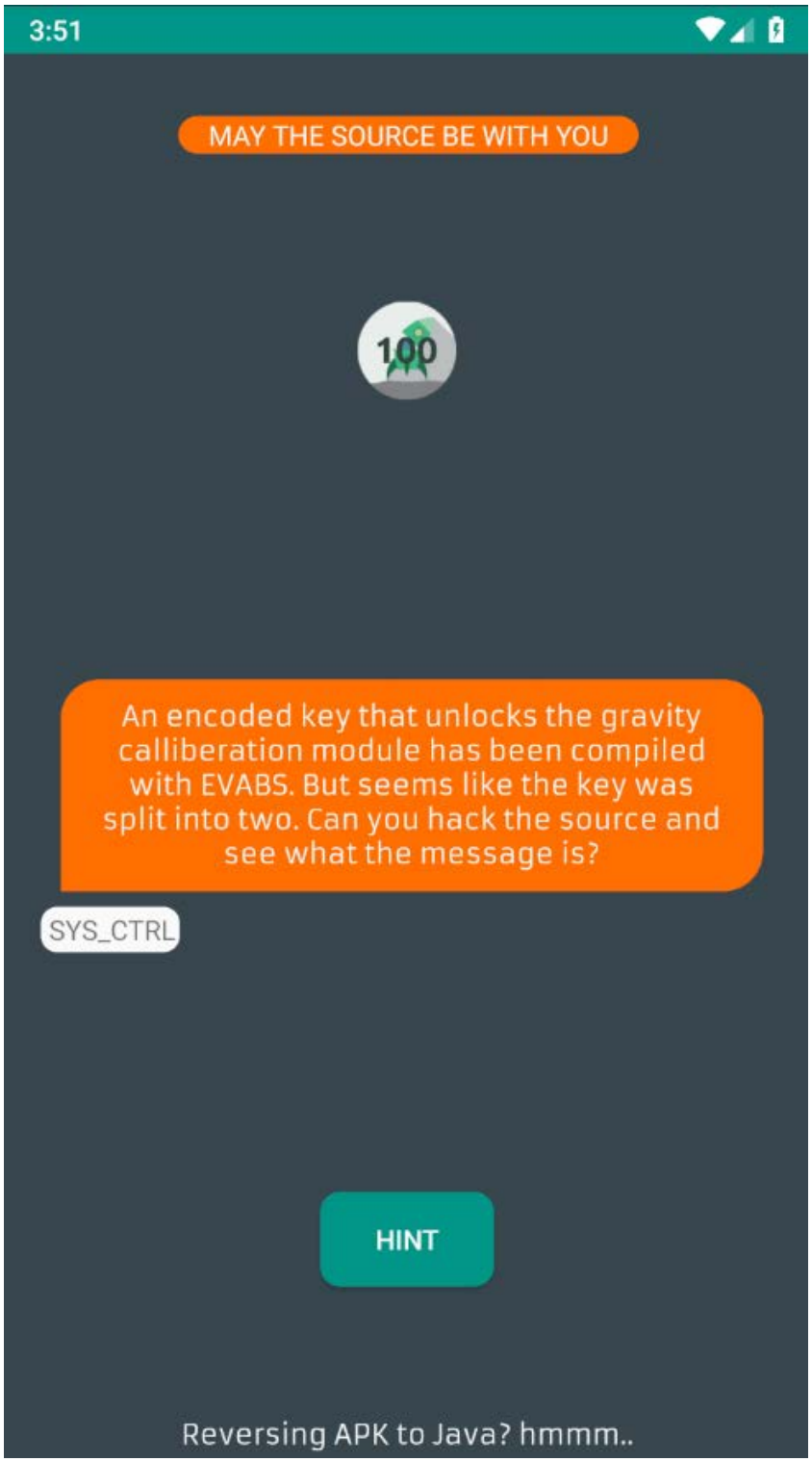
adb shell am start -n app.package.name/app.package.name.ActivityName



Once the command executed, the error message above popup in the app together with the flag.

CHALLENGE LEVEL 8

Next, the reversing part. I read the word source on the title, I already know this gonna be hella draggy since I'm not quite master on doing static especially reversing. Ok, this challenge requires us to hack and find a key that have been split into two part in the source code of this app.



As it built based on Java and the key mentioned stored in the source code then what else to do rather than reverse the APK file.

```
 Mobexler@Mobexler  ~  d2j-dex2jar.sh base.apk
./base-dex2jar.jar exists, use --force to overwrite
 Mobexler@Mobexler  ~  d2j-dex2jar.sh base.apk --force
dex2jar base.apk -> ./base-dex2jar.jar
 Mobexler@Mobexler  ~  ls
Android                 base.apk            dist        hs_err_pid10933.log  Pictures   Templates
AndroidZone             base-dex2jar.jar    Documents   iOSZone              Public     tools
base                    BurpSuiteCommunity  Downloads   Music                results    tool_update
base16-xfce4-terminal   Desktop             GNUstep     node_modules         SecZone    Videos
```

By using extracted APK file used on previous challenge, I fired this command to dissemble the APK file into Java file;

d2-dex2jar.sh file.apk

Noted: --*force* tag used because I already did before, and this tag will overwrite the previous one.

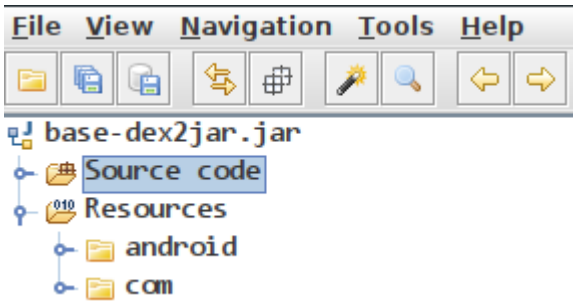
Furthermore, what made me eager on using Mobexler is because all tools related like this one was already installed so no worries. Kudos to the Mobexler team.

```
 Mobexler@Mobexler  ~  jadx-gui                                                    ✓  183  13:02:07
INFO  - output directory: base-dex2jar
INFO  - loading ...
INFO  - converting to dex: base-dex2jar.jar ...
WARN  - Code restructure failed: missing block: B:10:0x0026, code lost:
    r0 = getWindow();
 in method: android.support.v7.app.AppCompatActivity.performMenuItemShortcut(int, android.view.KeyEvent):boolean, dex: b
ase-dex2jar.jar
INFO  - JADX INFO: finally extract failed in android.support.v4.app.FragmentActivity.requestPermissionsFromFragment(andr
oid.support.v4.app.Fragment, java.lang.String[], int):void
WARN  - Code restructure failed: missing block: B:2:0x0006, code lost:
    r0 = r1.mMenuView;
 in method: android.support.v7.widget.Toolbar.canShowOverflowMenu():boolean, dex: base-dex2jar.jar
INFO  - JADX INFO: finally extract failed in com.revo.evabs.flagcheck.checktheflag.doInBackground(java.lang.String[]):ja
va.lang.String
```

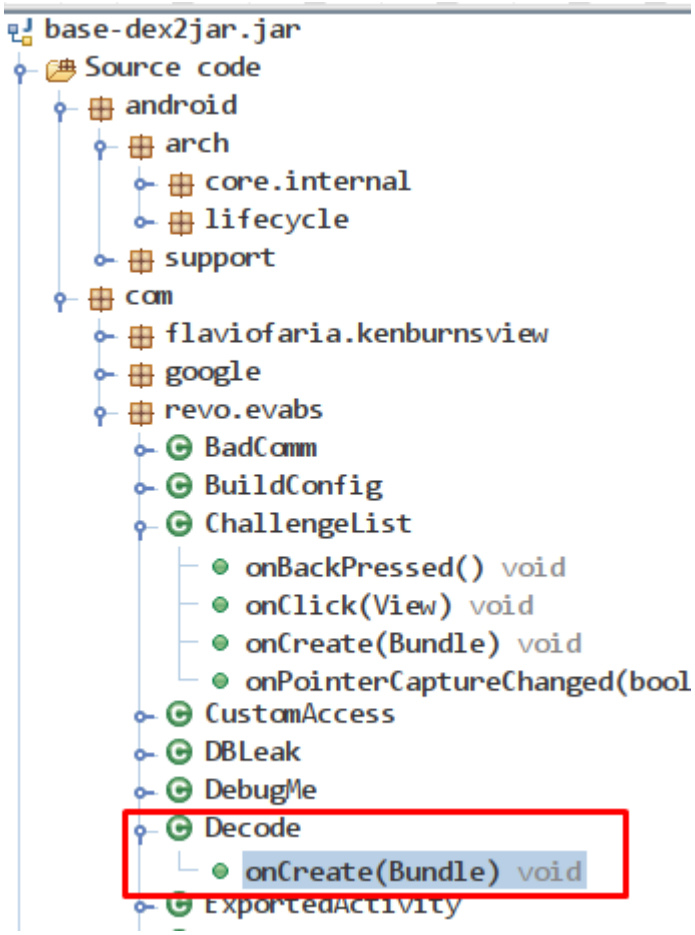
Then, to read the Java/Jar file I used this Java to Dex gui tool which is Jadx;

Jadx-gui

GUI popup



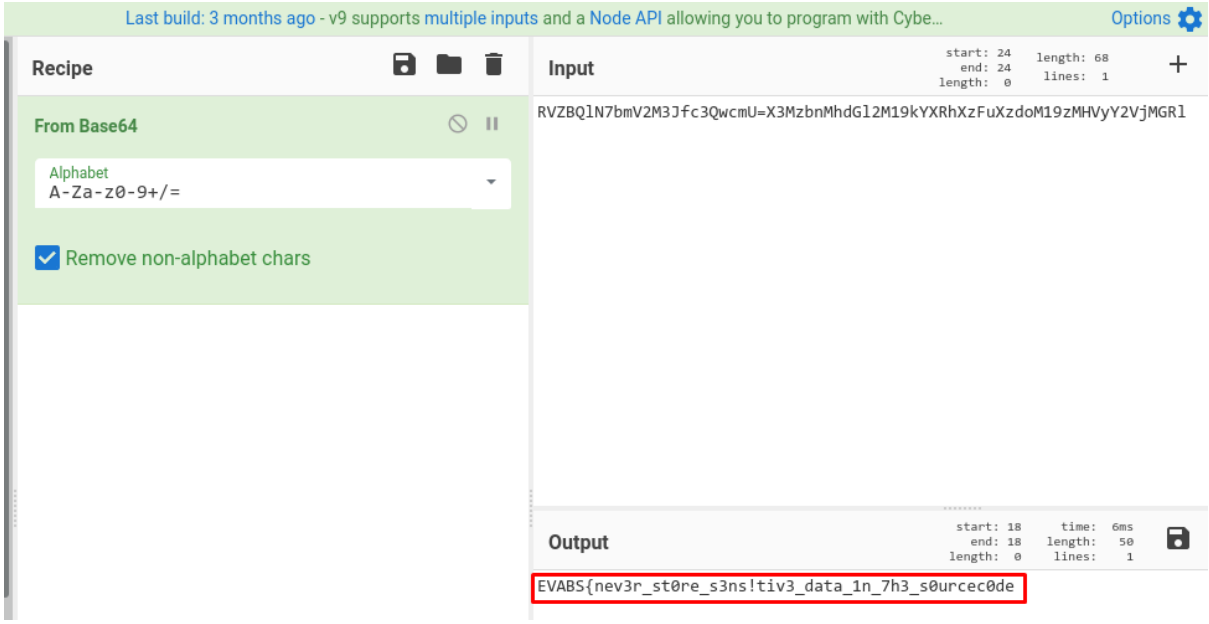
Within that Jadx gui, navigate to the Java file decompiled before then clicked on **Source code** tab as task for this challenge are about reversing its source code.



After some time reading all those classes and functions declared, then I stumbled on this code named **Decode** where it contained some weird string.

```
com.revo.evabs.Splash    com.revo.evabs.SmaliInject    com.revo.evabs.Decode
1  package com.revo.evabs;
2
3  import android.os.Bundle;
4  import android.support.v7.app.AppCompatActivity;
5  import android.view.View;
6  import android.widget.Button;
7  import android.widget.TextView;
8
9  public class Decode extends AppCompatActivity {
10     /* access modifiers changed from: protected */
11     public void onCreate(Bundle bundle) {
12         super.onCreate(bundle);
13         setContentView((int) R.layout.activity_decode);
14         "RVZBQlN7bmV2M3Jfc3QwcmU=" + "X3MzbnMhdGl2M19kYXRh" + "XzFuXzdoM19zMHVyY2VjMGRl";
15         final TextView textView = (TextView) findViewById(R.id.textviewhintgc);
16         ((Button) findViewById(R.id.buttonhintdecode)).setOnClickListener(new View.OnClickListener() {
17             public void onClick(View view) {
18                 textView.setText("Reversing APK to Java? hmmm..");
19             }
20         });
21     }
22 }
```

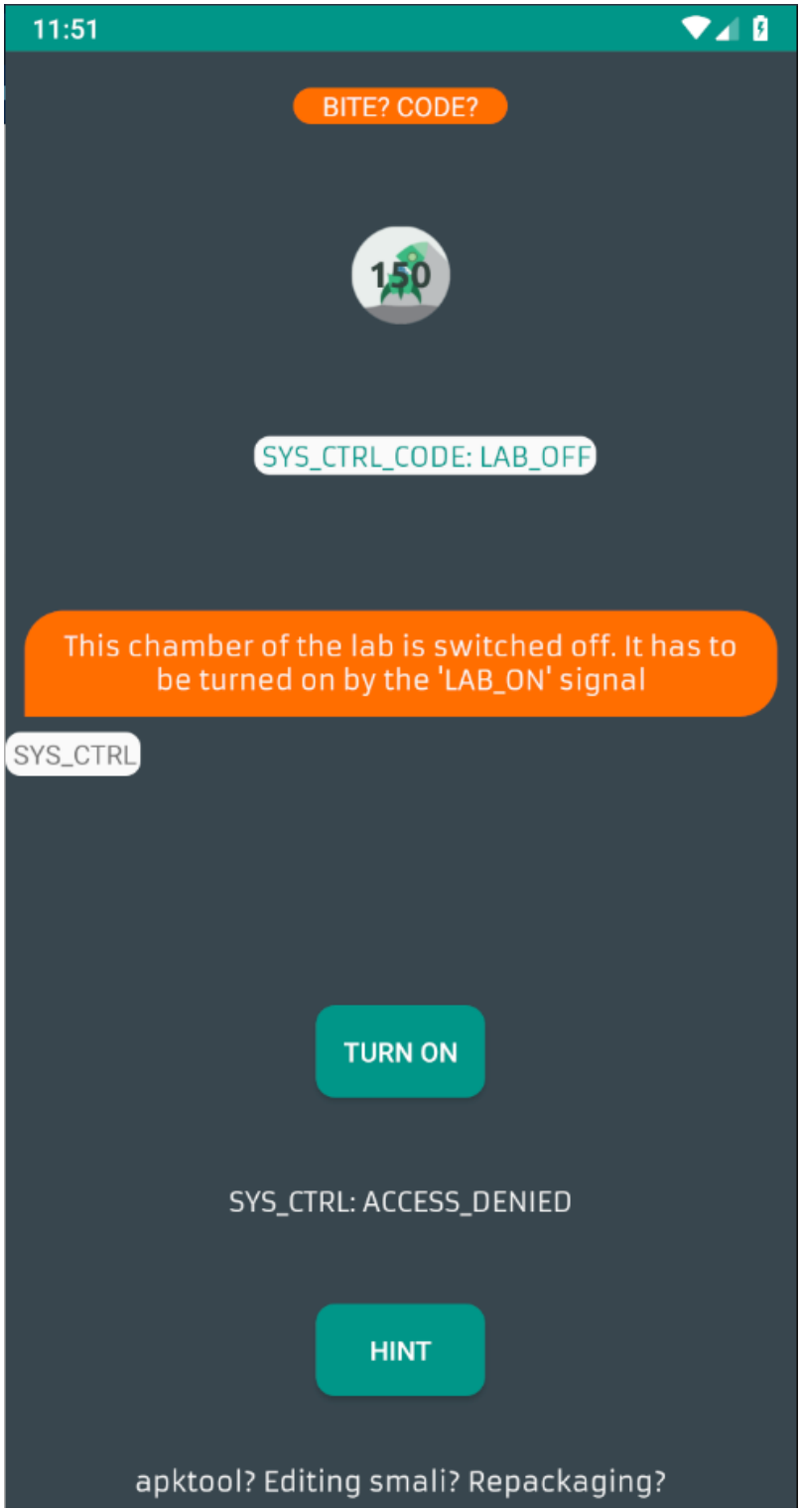
In that code, there's a line that split into two just like what mentioned at the challenge page. Well its basically split into three part but whatever, gonna finish this challenge anyway lol.



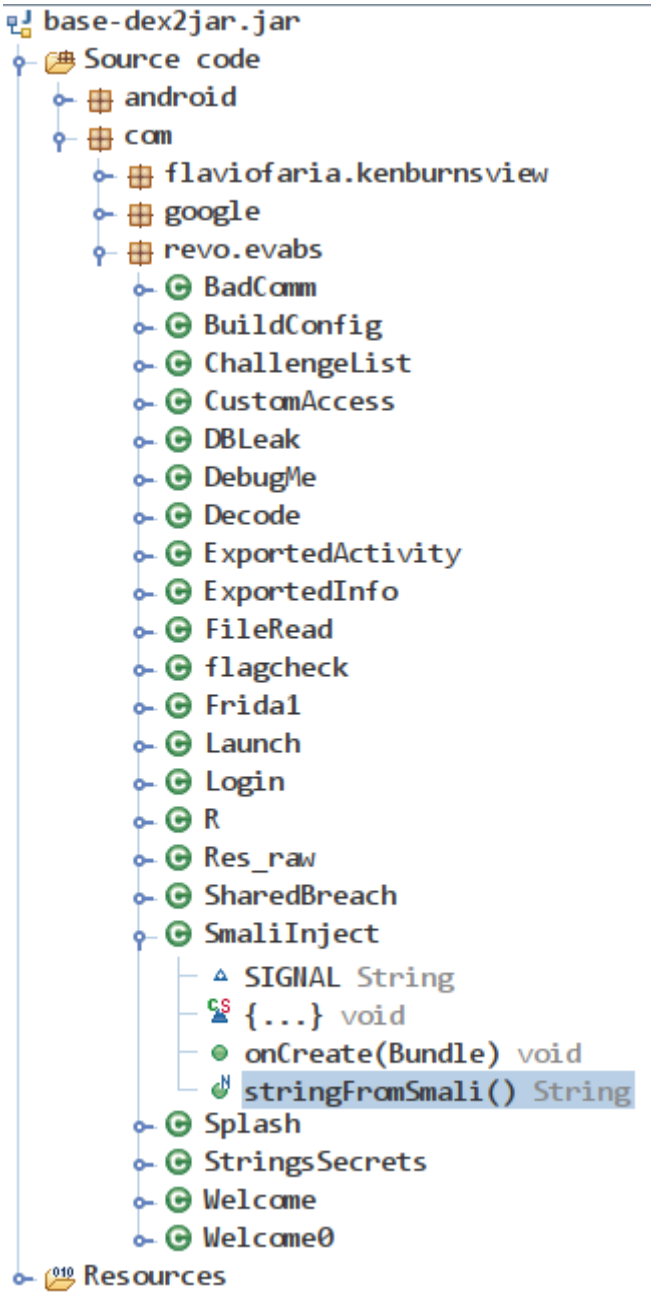
Then I copied that string to translate it. I used CyberChef to translate it to readable form and it turned out the flag for this challenge. One good thing about this website, is you can use Magic recipe to predict what encode string entered before you use the right recipe like my case using Base64 recipe.

CHALLENGE LEVEL 9

Another source code related challenge. Why? It stated code there! This challenge has some sort of control feature there and it need us to switch back on so then can enter the lab chamber.



This challenge worked by pressing the button TURN ON and theoretically it will grant me an access to the lab chamber, but it turned out ACCESS_DENIED for me. Any idea? Yeah need to edit this app smali file.



So, remember previously I used this Jdax gui? Yea gonna use back the same tool and same folder(to make thing much easier and faster). After that, since this challenge about altering smali file then I go straight to the smali classes named **SmaliInject** and look deeper into **stringFromSmali()** function.

com.revo.evabs.SmaliInject

```java
1  package com.revo.evabs;
2
3  import android.os.Bundle;
4  import android.support.v7.app.AppCompatActivity;
5  import android.view.View;
6  import android.widget.Button;
7  import android.widget.TextView;
8
9  public class SmaliInject extends AppCompatActivity {
10     String SIGNAL = "LAB_OFF";
11
12     static {
13         System.loadLibrary("native-lib");
14     }
15
16     /* access modifiers changed from: protected */
17     public void onCreate(Bundle bundle) {
18         super.onCreate(bundle);
19         setContentView((int) R.layout.activity_smali_inject);
20         final TextView textView = (TextView) findViewById(R.id.textViewlabstatus);
21         final TextView textView2 = (TextView) findViewById(R.id.textViewsmalihint);
22         final TextView textView3 = (TextView) findViewById(R.id.textViewlaboff);
23         final TextView textView4 = (TextView) findViewById(R.id.textViewflag);
24         ((Button) findViewById(R.id.buttonhintsmali)).setOnClickListener(new View.OnClickListener() {
25             public void onClick(View view) {
26                 textView2.setText("apktool? Editing smali? Repackaging?");
27             }
28         });
29         ((Button) findViewById(R.id.buttonlabon)).setOnClickListener(new View.OnClickListener() {
30             public void onClick(View view) {
31                 String stringFromSmali = SmaliInject.this.stringFromSmali();
32                 if (SmaliInject.this.SIGNAL.equals("LAB_ON")) {
33                     textView3.setText("SYS_CTRL_CODE: LAB_ON");
34                     textView.setText("SYS_CTRL: ACCESS_GRANTED. LAB UNLOCKED");
35                     TextView textView = textView4;
36                     textView.setText("EVABS{" + stringFromSmali + "}");
37                     return;
38                 }
39                 textView3.setText("SYS_CTRL_CODE: LAB_OFF");
40                 textView.setText("SYS_CTRL: ACCESS_DENIED");
41             }
42         });
43     }
44
45     public native String stringFromSmali();
46 }
```

In that classes code, confirmed it is a smali part that needed to edit because refer to 2nd red box there's a if condition which will trigger the LAB_ON signal as per declared at the 1st redbox class. Here's the logic for this challenge, once the **String SIGNAL** class changed to **LAB_ON,** then the statement will trigger **textView.setText** statement at the 3rd red box highlighted. Thus, I assumed will get me the flag for this challenge.

```
19
20 .method public constructor <init>()V
21     .registers 2
22
23     invoke-direct {p0}, Landroid/support/v7/app/AppCompatActivity;-><init>()V
24
25     const-string v0, "LAB_OFF"
26
27     iput-object v0, p0, Lcom/revo/evabs/SmaliInject;->SIGNAL:Ljava/lang/String;
28
29     return-void
30 .end method
31
32
33 # virtual methods
34 .method protected onCreate(Landroid/os/Bundle;)V
35     .registers 9
36
37     invoke-super {p0, p1}, Landroid/support/v7/app/AppCompatActivity;->onCreate(Landroid/os/Bundle;)V
38
39     const v0, 0x7f0c002a
40
41     invoke-virtual {p0, v0}, Lcom/revo/evabs/SmaliInject;->setContentView(I)V
42
43     const v0, 0x7f0a0036
44
45     invoke-virtual {p0, v0}, Lcom/revo/evabs/SmaliInject;->findViewById(I)Landroid/view/View;
46
47     move-result-object v0
48
49     check-cast v0, Landroid/widget/Button;
50
51     const v1, 0x7f0a0035
52
53     invoke-virtual {p0, v1}, Lcom/revo/evabs/SmaliInject;->findViewById(I)Landroid/view/View;
54
55     move-result-object v1
56
57     check-cast v1, Landroid/widget/Button;
58
59     const v2, 0x7f0a0130
60
61     invoke-virtual {p0, v2}, Lcom/revo/evabs/SmaliInject;->findViewById(I)Landroid/view/View;
62
63     move-result-object v2
64
65     check-cast v2, Landroid/widget/TextView;
66
```

Code | Smali

Since I understand how's the code work then what I did is, at the below of the windows frame, simply click the SMALI tab beside CODE tab to read its smali code then find the string that need to be changed in order to make the codes work the way it should be. In this case, the string needed to edit is one that I highlighted.

```
Mobexler@Mobexler  ~/base/smali/com/revo/evabs  ls                                                    ✓ 242 15:23:58
'BadComm$1.smali'            'ExportedInfo$1.smali'          'R$anim.smali'       R.smali
'BadComm$2.smali'             ExportedInfo.smali             'R$array.smali'      'R$string.smali'
'BadComm$AsyncLogin.smali'   'FileRead$1.smali'              'R$attr.smali'       'R$styleable.smali'
 BadComm.smali                FileRead.smali                 'R$bool.smali'       'R$style.smali'
 BuildConfig.smali           'flagcheck$1.smali'             'R$color.smali'      'SharedBreach$1.smali'
 ChallengeList.smali         'flagcheck$checktheflag.smali'  'R$dimen.smali'       SharedBreach.smali
'CustomAccess$1.smali'        flagcheck.smali                'R$drawable.smali'   'SmaliInject$1.smali'
'CustomAccess$2.smali'       'Frida1$1.smali'                'Res_raw$1.smali'    'SmaliInject$2.smali'
 CustomAccess.smali           Frida1.smali                   Res_raw.smali        SmaliInject.smali
'DBLeak$1.smali'             'Launch$1.smali'                'R$font.smali'       'Splash$1.smali'
'DBLeak$2.smali'             'Launch$2.smali'                'R$id.smali'         Splash.smali
 DBLeak.smali                 Launch.smali                   'R$integer.smali'    'StringsSecrets$1.smali'
 DebugMe.smali               'Login$1.smali'                 'R$layout.smali'      StringsSecrets.smali
'Decode$1.smali'             'Login$2.smali'                 'R$menu.smali'       'Welcome0$1.smali'
 Decode.smali                 Login.smali                    'R$mipmap.smali'      Welcome0.smali
 ExportedActivity.smali      'R$animator.smali'              'R$raw.smali'         Welcome.smali
Mobexler@Mobexler  ~/base/smali/com/revo/evabs  nano SmaliInject.smali                                  ✓ 274 09:25:44
Mobexler@Mobexler  ~/base/smali/com/revo/evabs                                                          ✓ 276 09:33:24
```

In the app folder, go to */base/smali/com/revo/evabs/* to locate the smali file which is file named **SmaliInject.smali.** Using this command to open and edit the file, *nano filename.file*

```
  GNU nano 2.9.3                                        SmaliInject.smali

.class public Lcom/revo/evabs/SmaliInject;
.super Landroid/support/v7/app/AppCompatActivity;
.source "SmaliInject.java"


# instance fields
.field SIGNAL:Ljava/lang/String;


# direct methods
.method static constructor <clinit>()V
    .locals 1

    .line 56
    const-string v0, "native-lib"

    invoke-static {v0}, Ljava/lang/System;->loadLibrary(Ljava/lang/String;)V

    .line 57
    return-void
.end method

.method public constructor <init>()V
    .locals 1

    .line 11
    invoke-direct {p0}, Landroid/support/v7/app/AppCompatActivity;-><init>()V

    .line 13
    const-string v0, "LAB_ON"

    iput-object v0, p0, Lcom/revo/evabs/SmaliInject;->SIGNAL:Ljava/lang/String;
```

Edit the string from LAB_OFF to LAB_ON as highlighted above.

```
Mobexler@Mobexler  ~  apktool b base -o baseV1.apk                                                     ✓ 278 09:35:46
I: Using Apktool 2.4.1
I: Checking whether sources has changed...
I: Smaling smali folder into classes.dex...
I: Checking whether resources has changed...
I: Building apk file...
```

Once done editing, It need to recompiled back in order to make it able to run normally as standard android app. Run this command, *Apktool b appfolder -o editedapp.apk* to compile back into APK file.

```
 Mobexler@Mobexler  ~  keytool -genkey -v -keystore evabs.keystore -keyalg RSA -keysize 2048 -validity 10000
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

Generating 2,048 bit RSA key pair and self-signed certificate (SHA256withRSA) with a validity of 10,000 days
        for: CN=Unknown, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=Unknown
[Storing evabs.keystore]
 Mobexler@Mobexler  ~  ls                                                                          ✔  277  09:35:42
Android      base16-xfce4-terminal  BurpSuiteCommunity  Documents       GNUstep  node_modules  results    tools
AndroidZone  base.apk               Desktop             Downloads       iOSZone  Pictures      SecZone    tool_update
base         base-dex2jar.jar       dist                evabs.keystore  Music    Public        Templates  Videos
 Mobexler@Mobexler  ~                                                                              ✔  278  09:35:46
```

Done compiling, proceed to sign the APK file. This signing process are essential to make it able to install and run normal on device(or emulator). In order to sign, I need a key first then be able to sign. So, creating an APK file key can use command as follows;

Keytool -genkey -v -keystore keyname.keystore -keyalg RSA -keysize 2048 -validity 1000

And proceed the process as requested whereby fill in all that information. You can skip by pressing Enter for each question ya know. Later, it will save as per declared on the command like the picture above.

```
 Mobexler@Mobexler  ~  jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore evabs.keystore baseV1.apk myke
y
Enter Passphrase for keystore:
 updating: META-INF/MYKEY.SF
 updating: META-INF/MYKEY.RSA
  signing: classes.dex
  signing: AndroidManifest.xml
```

The key already made then proceed with signing. The signing command using Jarsigner as follows;

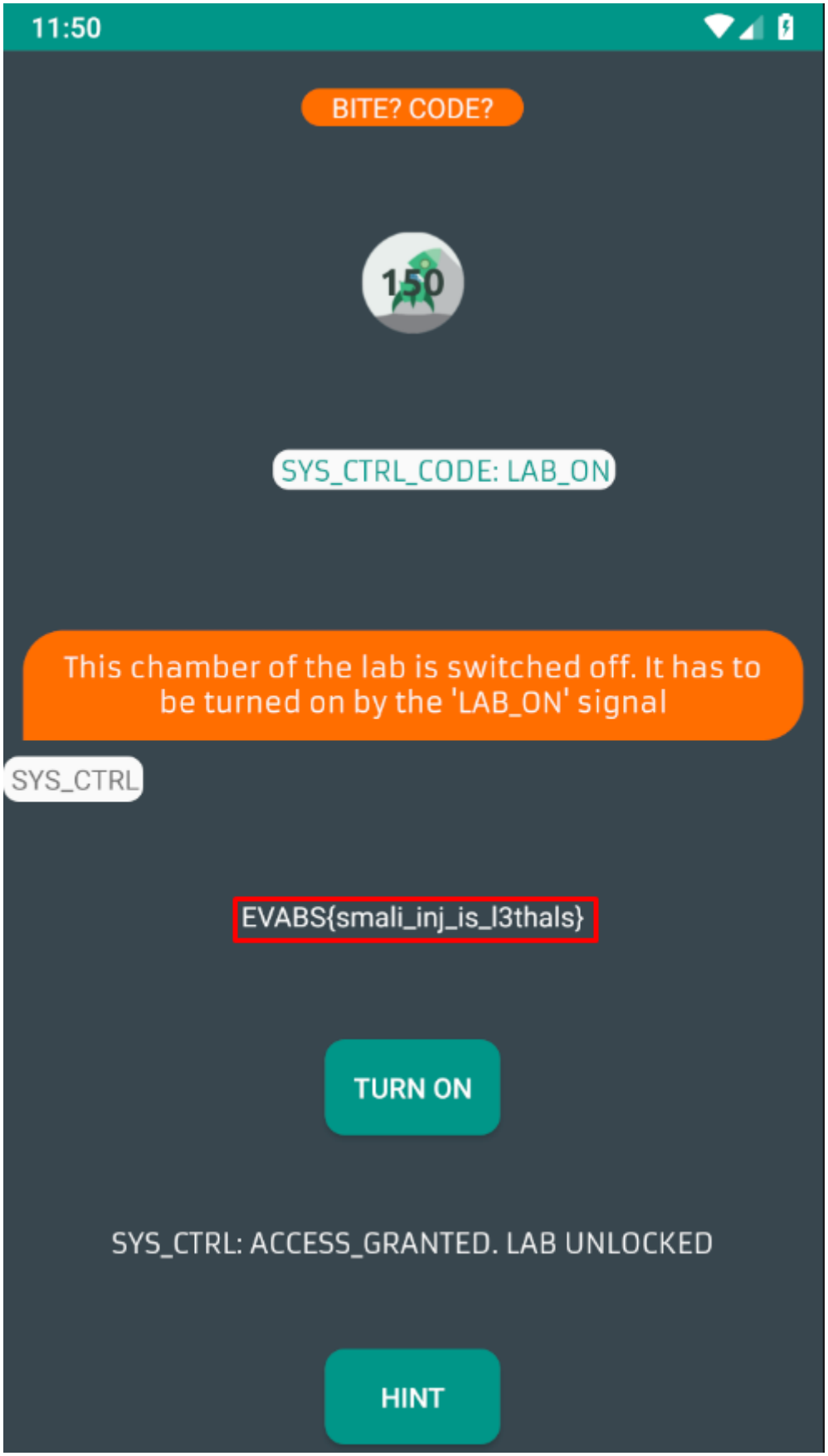
Jarsigner -verbose -signalg SHA1withRSA -digestalg SHA1 -keystore keyname.keystore apkfile.apk mykey

```
>>> Signer
    X.509, CN=Unknown, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=Unknown
    [trusted certificate]

jar signed.

Warning:
The signer's certificate is self-signed.
 Mobexler@Mobexler  ~                                                                              ✔  280  09:39:49
 Mobexler@Mobexler  ~                                                                              ✔  280  09:41:07
 Mobexler@Mobexler  ~  adb -s 192.168.28.101:5555 install '/home/mobexler/baseV1.apk'              ✔  280  09:41:07
```

Smali code edited, APK signed, then proceed to installation! You can manually install the APK file but for me, I just used adb command as follows;

Adb -s your.device.IP install 'apk/file/path/file.apk'

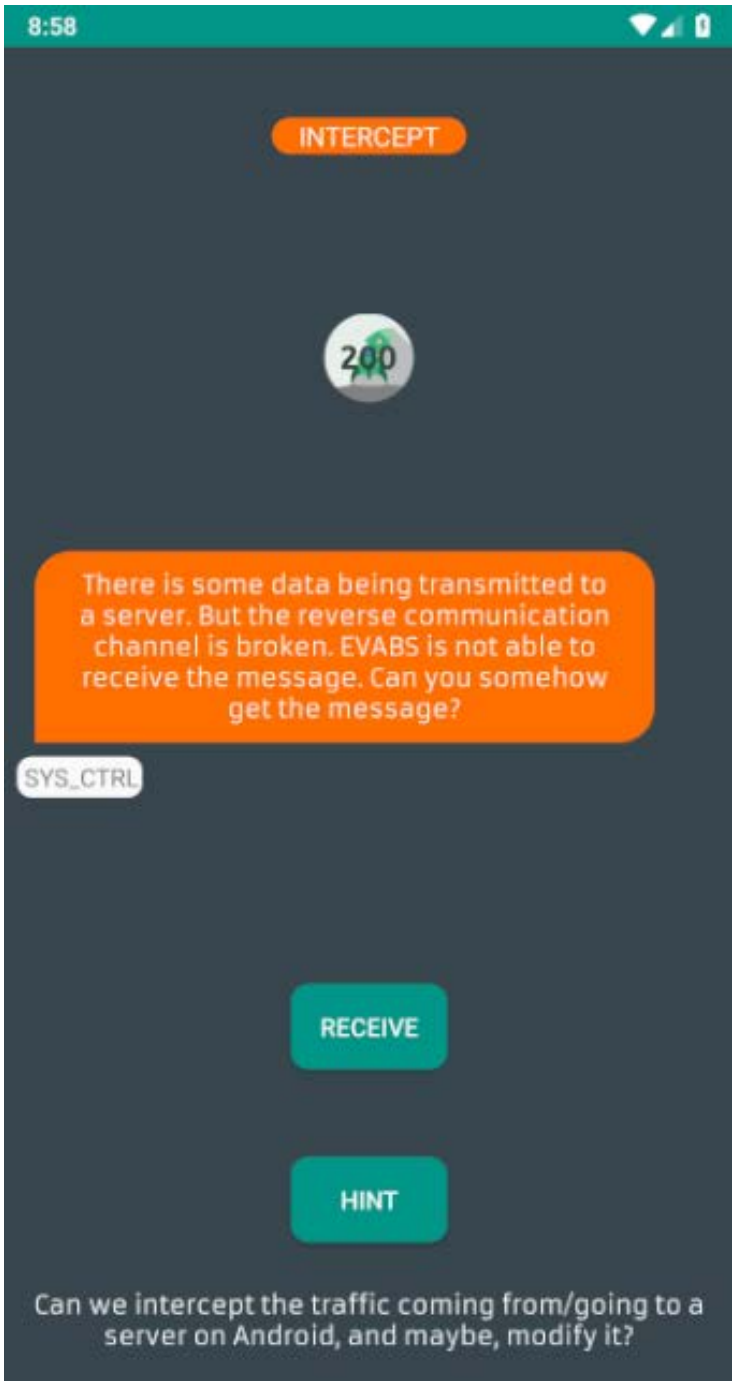
It will directly push the APK into mobile device and install afterward.



APK file successfully installed and open back the app as usual then press TURN ON button back and voila! ACCESS_GRANTED. LAB UNLOCKED. Lab chamber unlocked and got the flag.

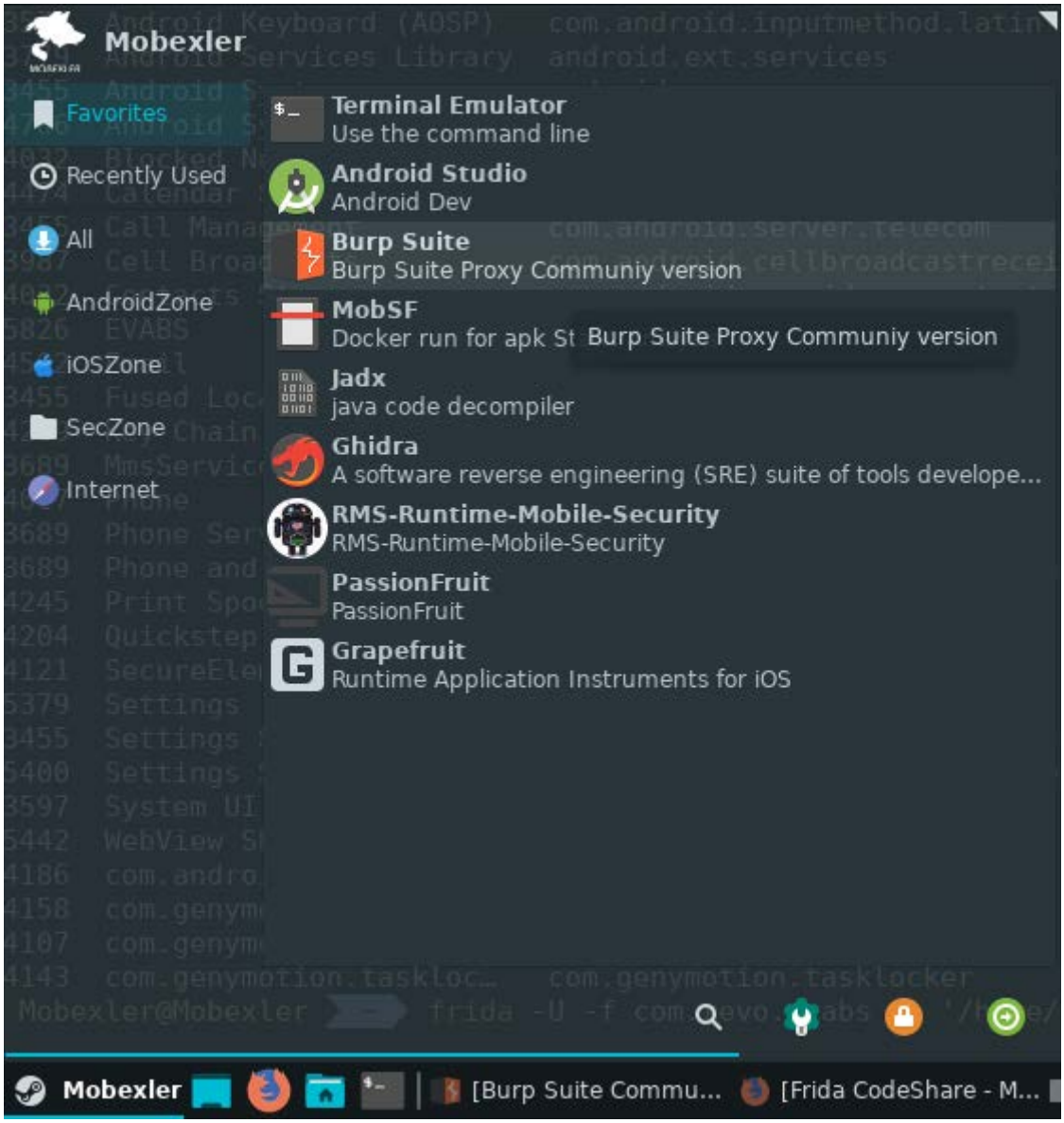
CHALLENGE LEVEL 10

10th challenge of this lab is about intercepting a network linked between the app and its backend. As explained within this challenge, a data being transmitted, and it need to be intercepted because of the communication channel used was broken. So let's do some interception!



What is this challenge all about was already well explained, yet the hint more directly focuses on what to do specifically for this challenge. Intercept traffic and modify. Simple.

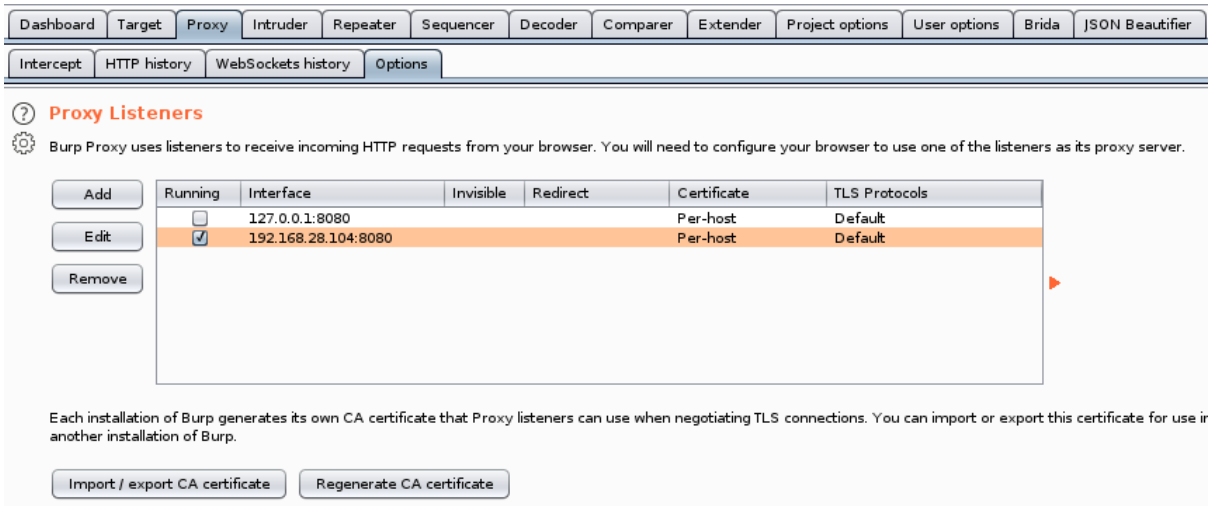
How this challenge work basically like normal client server interaction which is any node must send or receive a transmitted data thus by clicking the RECEIVE button, a data will be sent from the server and receive here. I must intercept it!



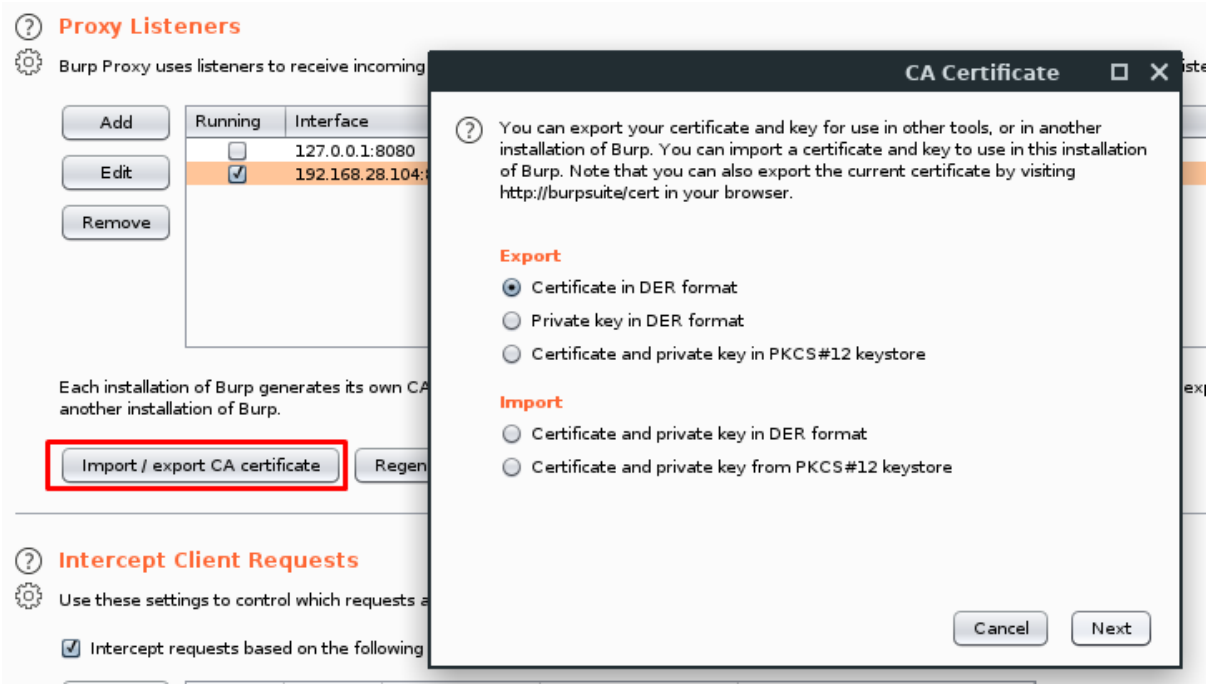
Intercepting a network must use a proxy platform and the best proxy platform for sure is no other, the Burp Suite! As it open, follow and read next to proceed using it.

```
Mobexler@Mobexler  ~  ifconfig
enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.28.104  netmask 255.255.255.0  broadcast 192.168.28.255
        inet6 fe80::839a:bcf9:f262:b5f8  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:7b:ef:68  txqueuelen 1000  (Ethernet)
        RX packets 28959  bytes 2340170 (2.3 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 36989  bytes 49843963 (49.8 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Once burp opened, in your terminal command *ifconfig* to get your machine IP and in my case, Mobexler machine IP.



Then, head up to the **Proxy** tab and click **Options** tab. In the **Proxy Listeners** section, **Add** and key in your machine IP with port. Usually for interception will be using 8080.

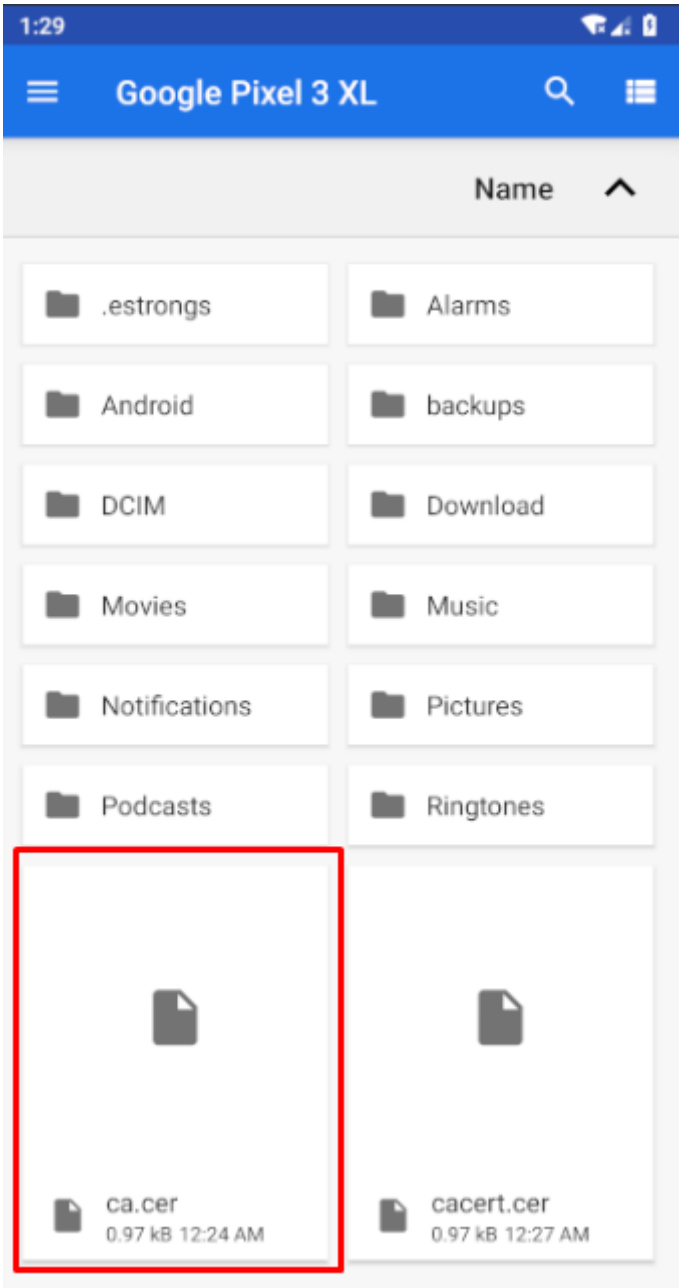


Done setup, click button marked red as picture above then tick on **Certificate in DER format.** Click **Next.**

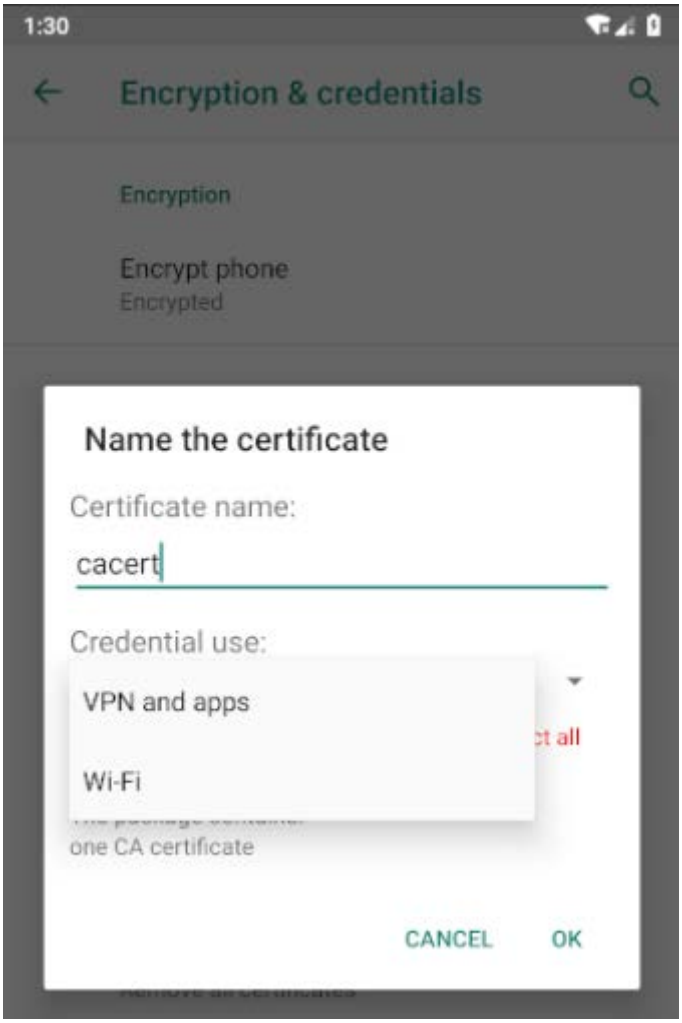
```
Mobexler@Mobexler  ~  adb push '/home/mobexler/Desktop/ca.cer' 'sdcard/'        ✓ 315 10:34:48
```

Once the cert exported, transfer it to the mobile device by using the command follows;

Adb push "/dir/of/the/ca.cert" "sdcard/"



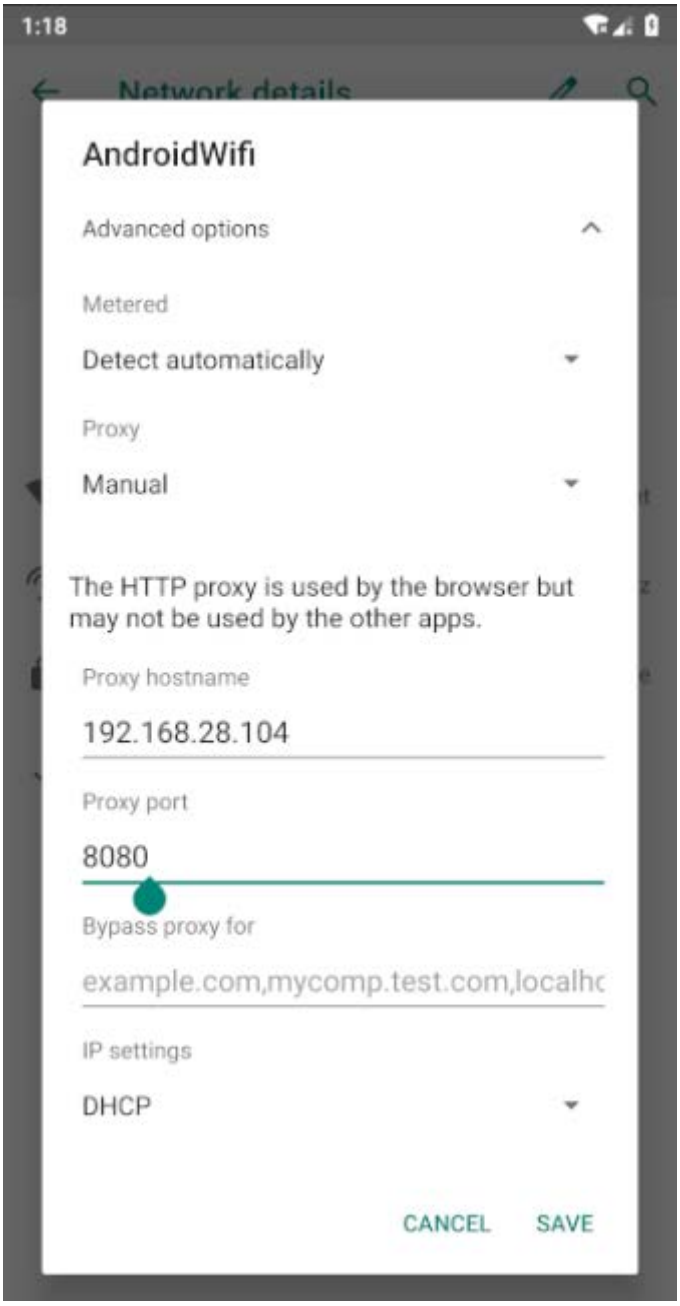
In your mobile device, you'll find it in the sdcard directory as declared in the previous command.



Ok for this step, I did the long way(coz you know it's a first time discovering a solution) thus I'll just share the shortcut way in completing this. You may click the pushed ca.cert then it'll be popup as the picture above.

You can name any name but for me to not confused, I just named it cacert and need to do it twice. Yes twice, one for **VPN and apps** and **Wi-Fi.**

You may confirm its installation by navigating to Settings > Security & Location > Encryption & Credential > User Credentials. This path for Google Pixel 3 XL device.



Now, open your connected wifi setting and click **Advanced Setting** to set to burp proxy settings. Starts with change **Proxy** to Manual, **proxy hostname** to your machine IP and port declared as per set in the burp. Click **SAVE.**

13:04:56 11 Oct 2020	Proxy	The client failed to negotiate an SSL connection to www.google.com:443: Received fatal alert: certificate_unknown
13:04:10 11 Oct 2020	Proxy	The client failed to negotiate an SSL connection to evabsflag.000webhostapp.com:443: Received fatal alert: certificate_unknown
13:00:33 11 Oct 2020	Proxy	Proxy service started on *:8082
13:00:09 11 Oct 2020	Suite	You appear to be using a 32-bit JVM. Please note that some Burp features are not fully supported on 32-bit systems.
13:00:08 11 Oct 2020	Proxy	Proxy service started on 127.0.0.1:8080

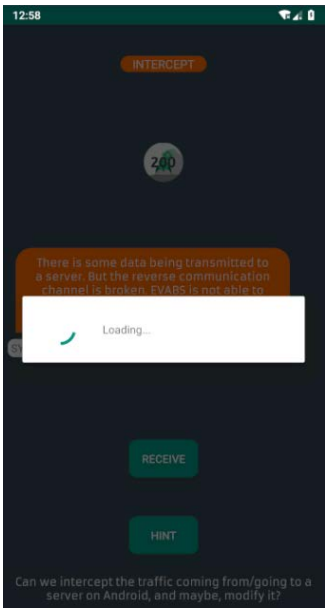
As it saved and connected to the proxy, above alert will triggered indicating that proxy is connected but can't be intercepted yet. At this part, can read there that the issue is about the client failed to negotiate an SSL connection and to overcome this, need to modify the SSL a bit.

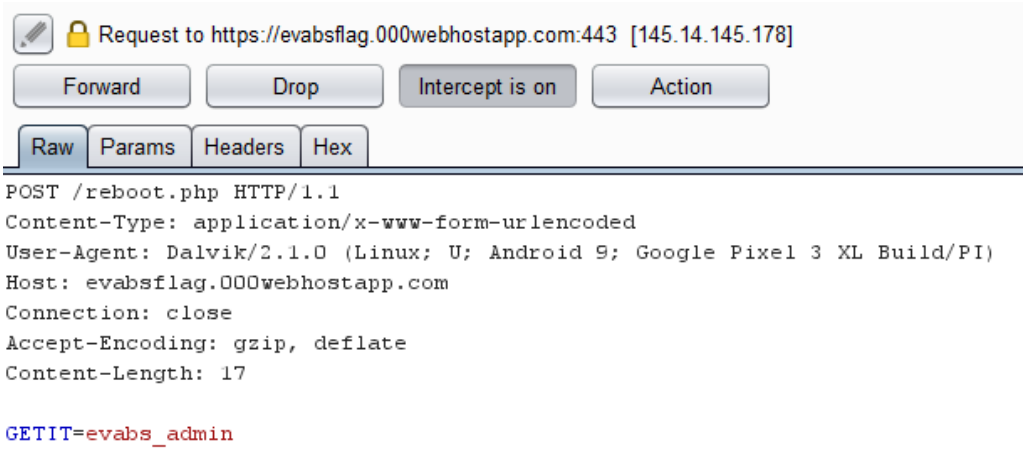
```
Mobexler@Mobexler  ~  frida -U -f com.revo.evabs -l '/home/mobexler/Desktop/frida-ssl.js' --no-paus
     ____
    / _  |   Frida 12.10.4 - A world-class dynamic instrumentation toolkit
   | (_| |
    > _  |   Commands:
   /_/ |_|       help      -> Displays the help system
   . . . .       object?   -> Display information about 'object'
   . . . .       exit/quit -> Exit
   . . . .
   . . . .   More info at https://www.frida.re/docs/home/
Spawned `com.revo.evabs`. Resuming main thread!
[Google Pixel 3 XL::com.revo.evabs]-> |
```

As modifying an SSL quite bit time consumed, you may download the modified one[2] then can easily used it to bypass the SSL certificate. Fire up below command to bypass!

frida -U -f app.package.name -l '/downloaded/path/of/modified/ssl.js' –no-paus

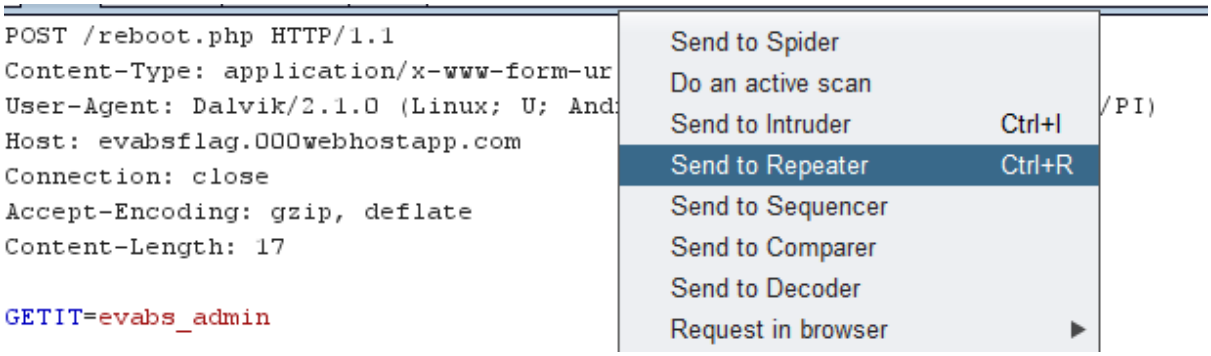
No error then you good to go.



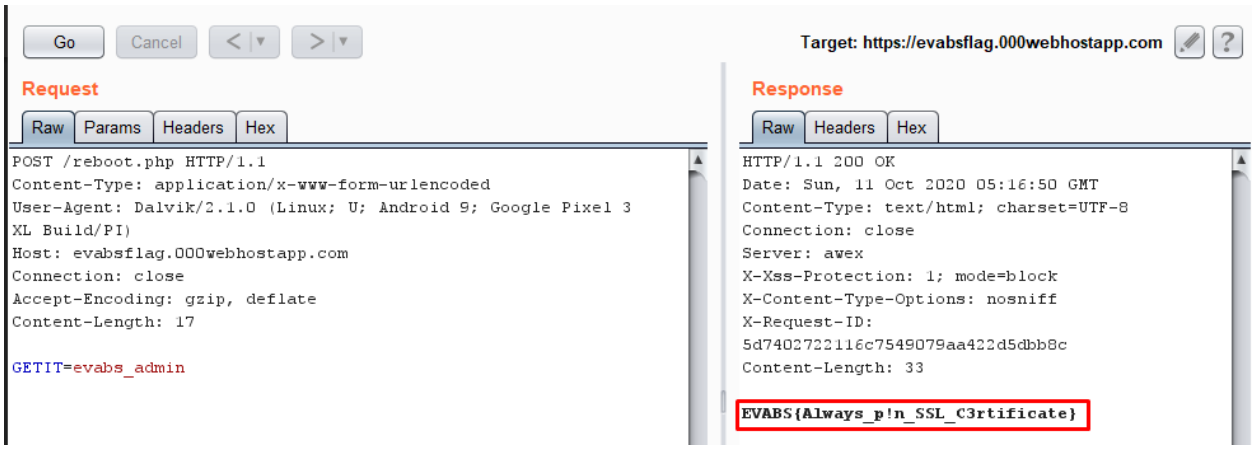


Back to the Burp, at the **Proxy** tab click the **intercept is on** to initiate the interception. While, at the app on the device, click that RECEIVE button then the apps start loading and it'll load to the Burp.

[2] https://codeshare.frida.re/@sowdust/universal-android-ssl-pinning-bypass-2/



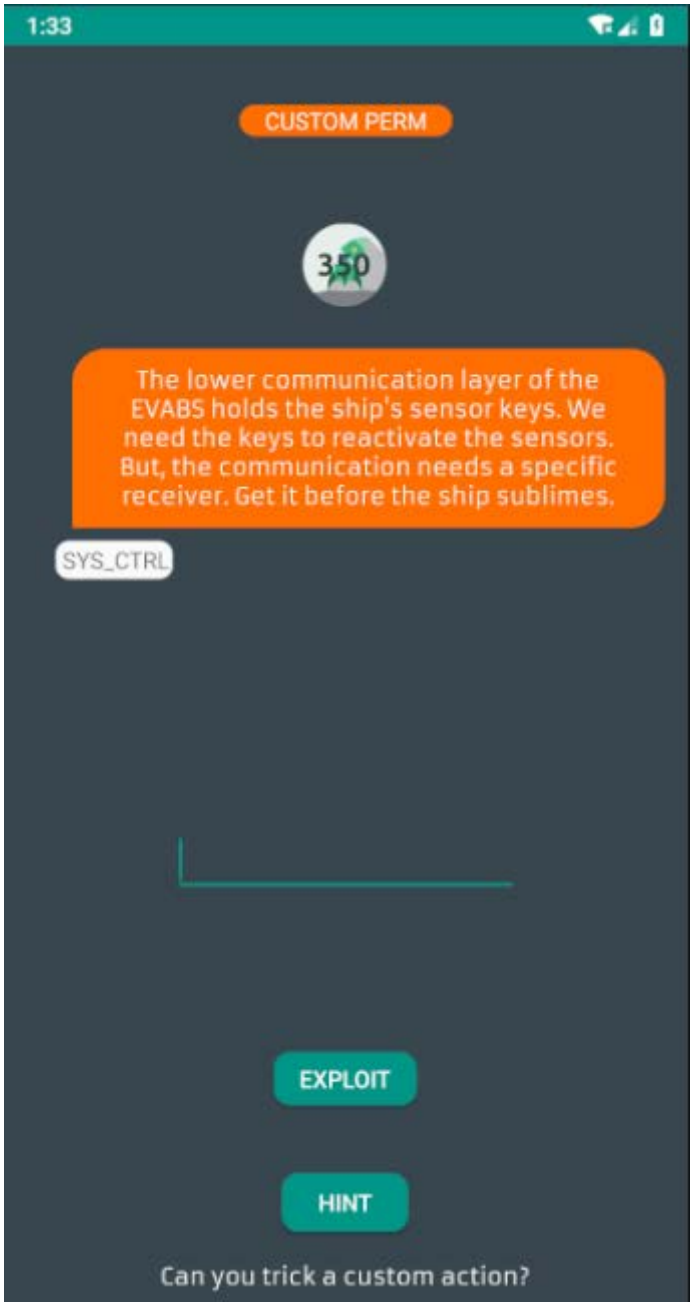
As it loads in the Burp, Right Click the request and click **Send to Repeater** to repeat the interception while receive the response from the server.



When it sent to the repeater, go to **Repeater** tab and you'll see the response from the app. Click **Go** button there then you can see the response back from the server. The response comes together with the flag. Interception done!

CHALLENGE LEVEL 11

This challenge regarding bypass the application permission. The description for this challenge aint much helping in term of understanding this challenge but hint button always there ya know. Based on hint stated, we need to do custom action to trick its permission. You may refer this link[3] to understand more about it.



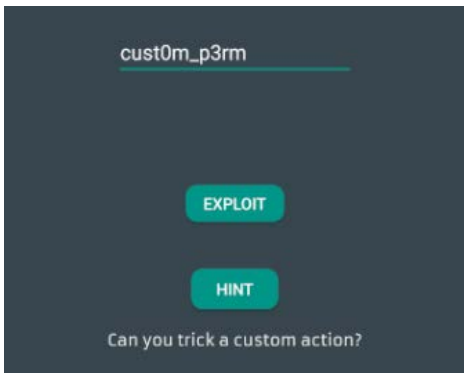
Ok as we can see, it asking to insert some string or so called sensor key to exploit the system while not specified what type of strings or key needed.

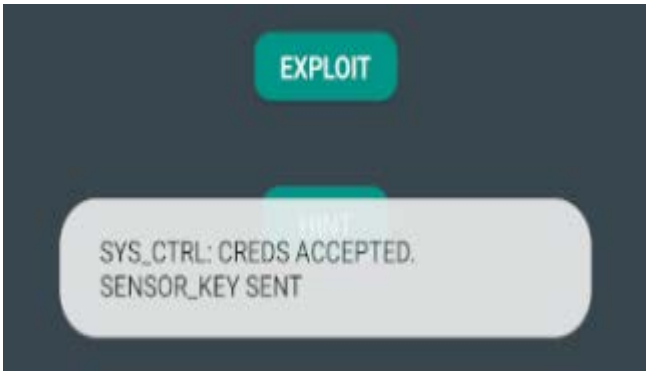
[3] https://developer.android.com/guide/topics/permissions/defining

Thus, I tried insert a single character and press EXPLOIT button to test then it popup error message saying that wrong cred sensors key locked. Meaning its required specific string in order to unlock the sensor key.

```
com.revo.evabs.CustomAccess    com.revo.evabs.DebugMe    com.revo.evabs.Expo

10 import android.widget.Toast;
11
12 public class CustomAccess extends AppCompatActivity {
13     public final String EVABS_SENSOR_KEY = "com.revo.evabs.action.SENSOR_KEY";
14
15     static {
16         System.loadLibrary("native-lib");
17     }
18
19     /* access modifiers changed from: private */
20     public void GetSensorKey() {
21         if ("cust0m_p3rm".equals(((EditText) findViewById(C0474R.C0476id.editTextcustomacc
22             Toast.makeText(this, "SYS_CTRL: CREDS ACCEPTED. SENSOR_KEY SENT", 1).show();
23             Intent intent = new Intent("com.revo.evabs.action.SENSOR_KEY");
24             intent.putExtra("android.intent.extra.TEXT", "EVABS{" + stringFromJNI() + "}")
25             intent.setType("text/plain");
26             startActivity(intent);
27             return;
28         }
29         Toast.makeText(this, "SYS_CTRL: WRONG_CREDS. SENSOR_KEY LOCKED", 1).show();
30     }
31
32     /* access modifiers changed from: protected */
33     public void onCreate(Bundle bundle) {
34         super.onCreate(bundle);
35         setContentView((int) C0474R.layout.activity_custom_access);
36         ((Button) findViewById(C0474R.C0476id.buttoncustomaccess)).setOnClickListener(new
37             public void onClick(View view) {
38                 CustomAccess.this.GetSensorKey();
```

First thing in my mind where to find the key is within the code itself. Fire up *jadx-gui* and open the application source code. Under class named **com.revo.evabs.CustomAccess** at **GetSensorKey** parameter can see there its specific string defined. The correct string stated is **Cust0m_p3rm**. Easy.





So, based on my understanding what the code does is after entering the correct input the flag will be passed to intent **com.revo.evabs.action.SENSOR_KEY** using the **putExtra()** function. After I input founded string, it only popup message the creds accepted sensor key sent but nothing happened. The input was correct, but the code doesn't do like what it should be.

```
/* access modifiers changed from: private */
public void GetSensorKey() {
    if ("cust0m_p3rm".equals(((EditText) findViewById(C0474R.C0476id.editTextcustomaccess)).getT
        Toast.makeText(this, "SYS_CTRL: CREDS ACCEPTED. SENSOR_KEY SENT", 1).show();
        Intent intent = new Intent("com.revo.evabs.action.SENSOR_KEY");
        intent.putExtra("android.intent.extra.TEXT", "EVABS{" + stringFromJNI() + "}");
        intent.setType("text/plain");
        startActivity(intent);
        return;
    }
    Toast.makeText(this, "SYS_CTRL: WRONG_CREDS. SENSOR_KEY LOCKED", 1).show();
}
```

I read the code back and see nothing wrong with it hence I decided to hook the function using frida. The idea of hooking comes to my mind since that's the only easy way for me to try. The idea is to hook and capture the **putExtra()** function in the main code for it to print the flag.

When hooking using Frida, I have to specify the object (function) that I want to change and the process (application) will execute this function. Frida will replace the content of the function it wants to become a new one. That way, when the application runs this function, it will work the way we want it to.

For this script, putExtra() function was used because it has many copies depending on the parameter type defined, so I have to use overload() to indicate the correct putExtra() function that takes the argument of 2 strings. Initial try was quite unsuccessful as there are many parse errors when using solely javascript code.

```python
import frida
import sys

def onMessage(message, data):
    print(message)

package = "com.revo.evabs"

jscode = """
Java.perform(function () {
    send("[-] Starting hooks android.content.Intent.putExtra");
    var intent = Java.use("android.content.Intent");
    intent.putExtra.overload("java.lang.String", "java.lang.String").implementation =
    function(var_1, var_2) {
        send("[+] Flag: " + var_2);
    };

});
"""
#device = frida.get_device_manager().add_remote_device('192.168.28.101:5555')
process = frida.get_usb_device().attach(package)
#process = frida.get_device_manager().add_remote_device('192.168.28.101:5555').attach(package)
script = process.create_script(jscode)
script.on("message", onMessage)
print("[*] Hooking", package)
```

Frankly speaking, I did several codes including java and javascript to hook the function as usual. Credit to my friend haha. End up using python code wrap the hooking javascript script inside as shown on the picture above.

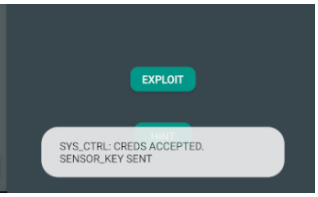
```
 Mobexler@Mobexler  ~  frida-ls-devices                                  1 ↵  919  13:15:37
Id                  Type    Name
------------------  ------  ----------------
local               local   Local System
192.168.28.101:5555 usb     Google Pixel 3 XL
socket              remote  Local Socket
 Mobexler@Mobexler  ~  cat s.py                                          ✔  920  13:15:40
import frida
import sys

def onMessage(message, data):
    print(message)

package = "com.revo.evabs"

jscode = """
Java.perform(function () {
    send("[-] Starting hooks android.content.Intent.putExtra");
    var intent = Java.use("android.content.Intent");
    intent.putExtra.overload("java.lang.String", "java.lang.String").implementation = function(var_1, var_2
) {
        send("[+] Flag: " + var_2);
    };
});
"""
#device = frida.get_device_manager().add_remote_device('192.168.28.101:5555')
process = frida.get_usb_device().attach(package)
#process = frida.get_device_manager().add_remote_device('192.168.28.101:5555').attach(package)
script = process.create_script(jscode)
script.on("message", onMessage)
print("[*] Hooking", package)
script.load()
sys.stdin.read()
```

Moving forward, the code must be made with correct device function process as the device attached using usb even actually only virtual Google Pixel 3 XL device.

```
Mobexler@Mobexler  ~  python3 s.py                                  ✔ 921 13:15:54
[*] Hooking com.revo.evabs
{'type': 'send', 'payload': '[-] Starting hooks android.content.Intent.putExtra'}
```

Since the code done, run the code by keying *python filename.py* and it will start hooking the application.

```
Mobexler@Mobexler  ~  python3 s.py                                  ✔ 924 13:19:04
[*] Hooking com.revo.evabs
{'type': 'send', 'payload': '[-] Starting hooks android.content.Intent.putExtra'}
{'type': 'send', 'payload': '[+] Flag: EVABS{always_ver1fy_packag3sa}'}
{'type': 'error', 'description': 'Error: Implementation for putExtra expected return value compatible with
android.content.Intent', 'stack': 'Error: Implementation for putExtra expected return value compatible with
 android.content.Intent\n    at je (frida/node_modules/frida-java-bridge/lib/class-factory.js:634)\n    at
frida/node_modules/frida-java-bridge/lib/class-factory.js:616', 'fileName': 'frida/node_modules/frida-java-
bridge/lib/class-factory.js', 'lineNumber': 634, 'columnNumber': 1}
                                                                                    2020-12-07
```



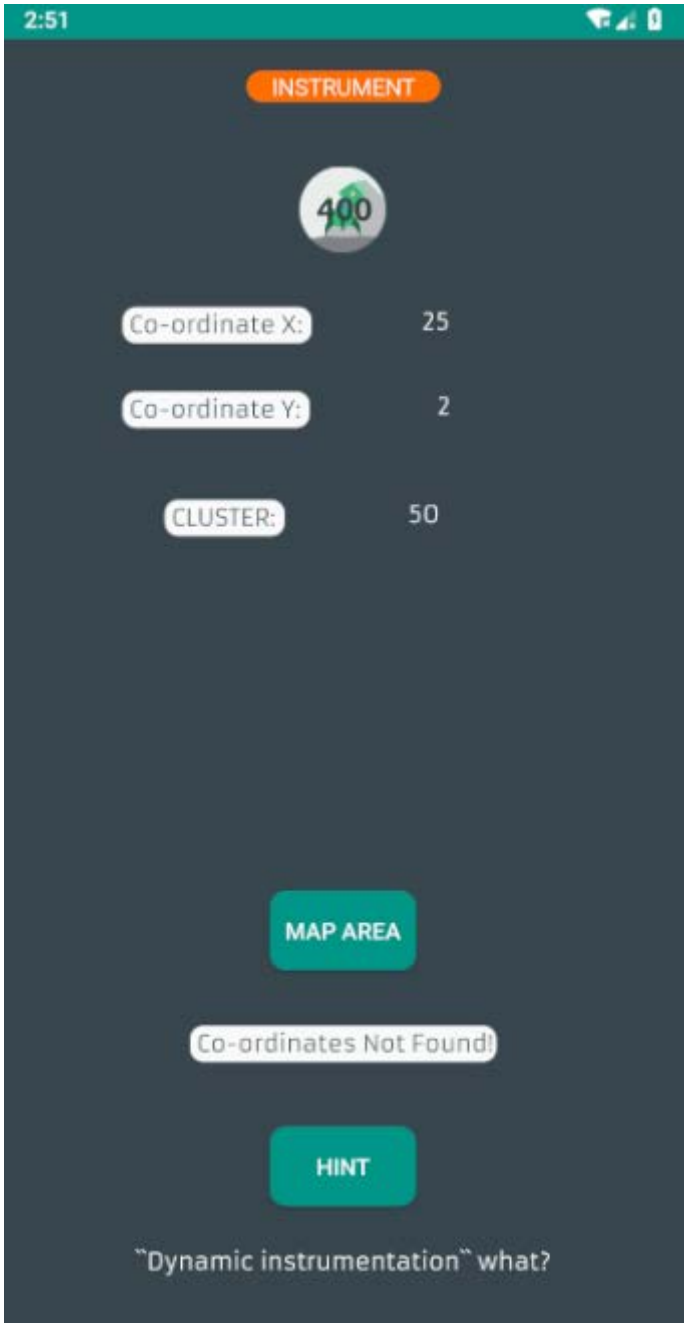
As the code already hooked the app, input back the correct string found earlier which is **cust0m_p3rm** in the devices and the flag popped up!

Disclaimer, the flag shown in the picture above has slight error. After several alteration on the flag upon submitting it, the approved flag is **EVABS{always_verify_packag3s}.**

CHALLENGE LEVEL 12

Last challenge!! Ah finally the final challenge for this application. This last challenge named with Instrument and first question come out to my mind is it a dynamic instrumentation or what. In Android, from my understanding there are Instrumentation which is some implementation style of base class while the other one is dynamic instrumentation whereby it's a testing process regarding analyzing and modifying of android binary application at the runtime.

Then, when I hit Hint button, it come out with stated sentence above. Which is pretty much I'm sure that this challenge gonna be some injection and bypassing apps runtime. As we understood what this challenge all about then we'll proceed with implementing the hint given.



Let's play around a bit the challenge. Whenever I press the **MAP AREA** button, those co-ordinates will automatically inserted and **Co-ordinates Not Found** message appeared too. Meaning that those co-ordinates need to be modified in order to get the correct co-ordinates.

Those co-ordinates seem to be constant as I did try hit the map button several times and turned out it still entered the same value together with the messages. Since this challenge about dynamic instrumentation, I also tried to look up on the debug log to deeply understand how this co-ordiantes work however pretty much nothing helpful there. So move out to do some source code review as usual.

com.revo.evabs.Frida1

```java
13      /* renamed from: a */
14      int f45a = 25;
15
16      /* renamed from: b */
17      int f46b = 2;
18
19      /* renamed from: x */
20      int f47x;
21
22      static {
23          System.loadLibrary("native-lib");
24      }
25
26      public void onClick(View view) {
27          TextView textView = (TextView) findViewById(C0474R.C0476id.result);
28          ((TextView) findViewById(C0474R.C0476id.valuea)).setText(String.valueOf(this.f45a));
29          ((TextView) findViewById(C0474R.C0476id.valueb)).setText(String.valueOf(this.f46b));
30          this.f47x = this.f45a * this.f46b;
31          int nextInt = new Random().nextInt(70);
32          ((TextView) findViewById(C0474R.C0476id.xres)).setText(String.valueOf(this.f47x));
33          if (this.f47x > nextInt + 150) {
34              textView.setText("VIBRAN IS RESDY TO FLY! YOU ARE GOING HOME!");
35              Log.d("CONGRATZ!", stringFromJNI());
36              return;
37          }
38          textView.setText("Co-ordinates Not Found!");
39      }
40
41      /* access modifiers changed from: protected */
```

Above is the code snippet for the challenge. Fortunately, it aint that hard to understand since just defined fix value and simple arithmetic.

```java
/* renamed from: a */
int f45a = 25;

/* renamed from: b */
int f46b = 2;

/* renamed from: x */
int f47x;

static {
    System.loadLibrary("native-lib");
}

public void onClick(View view) {
    TextView textView = (TextView) findViewById(C0474R.C0476id.result);
    ((TextView) findViewById(C0474R.C0476id.valuea)).setText(String.valueOf(this.f45a));
    ((TextView) findViewById(C0474R.C0476id.valueb)).setText(String.valueOf(this.f46b));
    this.f47x = this.f45a * this.f46b;
    int nextInt = new Random().nextInt(70);
    ((TextView) findViewById(C0474R.C0476id.xres)).setText(String.valueOf(this.f47x));
    if (this.f47x > nextInt + 150) {
        textView.setText("VIBRAN IS RESDY TO FLY! YOU ARE GOING HOME!");
        Log.d("CONGRATZ!", stringFromJNI());
        return;
    }
    textView.setText("Co-ordinates Not Found!");
}

/* access modifiers changed from: protected */
```

From the image above, we can see that all those integers defined with specific value. Then, it'll multiplied each other to come out with a new integer. Afterward, new integer defined randomly within ranged of 70. With the new random integer, it'll used to trigger the flag.

Third highlighted code where I hooked in order to modify the logic of the arithmetic used. As you can see below, I return the value of new random integer with negative value so then the logic will true then popup the flag stored.

```
  GNU nano 2.9.3                                        s1.py

import frida
import sys

def onMessage(message, data):
    print(message)

package = "com.revo.evabs"

jscode = """
Java.perform(function () {
    send("[-] Starting hooks java.util.Random.nextInt");
    var random = Java.use("java.util.Random");
    random.nextInt.overload("int").implementation = function(var_1) {
        return -150;
    };

});
"""

process = frida.get_usb_device().attach(package)
script = process.create_script(jscode)
script.on("message", onMessage)
print("[*] Hooking", package)
script.load()
sys.stdin.read()
                                                [ Read 25 lines ]
^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify
^X Exit        ^R Read File   ^\ Replace     ^U Uncut Text  ^T To Linter
```

Well, since previous challenge I used python for hooking then I'll proceed for this one too. Kinda new for me too yet glad to know that. Can refer this post[4] for better understanding. Typically, hooking using Frida will code using Javascript directly.

```
Mobexler@Mobexler  ~  nano s1.py                                  1 ↵  993  21:47:24
Mobexler@Mobexler  ~  python3 s1.py                                ✓  995  21:48:51
[*] Hooking com.revo.evabs
{'type': 'send', 'payload': '[-] Starting hooks java.util.Random.nextInt'}
```

Done with the script needed for hooking, moving forward with running it. Since it's a python code then it'll need to run using python command similar to previous challenge.

Whenever the code executed, it will wait the function declared trigger by the application. So ,in order to see the result, it need to analyze thru debug log as the script only modify the value.

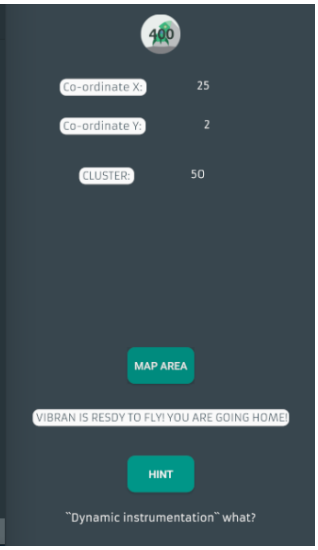
[4] https://erev0s.com/blog/frida-code-snippets-for-android/

```
1588  Download Manager          com.android.providers.downloads
2223  EVABS                     com.revo.evabs
1903  Email                     com.android.email
 640  Fused Location            com.android.location.fused
1588  Media Storage             com.android.providers.media
1946  Messaging                 com.android.messaging
 993  MmsService                com.android.mms.service
1971  One Time Init            com.android.onetimeinitializer
1995  Package installer         com.android.packageinstaller
1383  Phone                     com.android.dialer
 993  Phone Services            com.android.phone
 993  Phone and Messaging St…   com.android.providers.telephony
1622  Quickstep                 com.android.launcher3
2071  Search                    com.android.quicksearchbox
1545  SecureElementApplicati…   com.android.se
1001  Settings                  com.android.settings
 640  Settings Storage          com.android.providers.settings
2035  Superuser                 com.genymotion.superuser
2016  System Tracing             com.android.traceur
1195  System UI                 com.android.systemui
1930  Work profile setup        com.android.managedprovisioning
1643  com.android.smspush       com.android.smspush
1578  com.genymotion.genyd.…    com.genymotion.genyd
1530  com.genymotion.system…    com.genymotion.systempatcher
1562  com.genymotion.taskloc…   com.genymotion.tasklocker
Mobexler@Mobexler  ~  adb logcat --pid=2223
```

To do so, find the EVABS application ID and run the command;

Adb logcat - -pid=XXXX

```
adb shell                    adb logcat --pid=2223                python3 s1.py
12-31 11:12:22.518  2223  2252 D OpenGLRenderer: Swap behavior 1
12-31 11:12:22.519  2223  2252 W OpenGLRenderer: Failed to choose config with EGL_SWAP_BEHAVIOR_PRESERVED,
retrying without...
12-31 11:12:22.519  2223  2252 D OpenGLRenderer: Swap behavior 0
12-31 11:12:22.556  2223  2252 D EGL_emulation: eglCreateContext: 0xebccc740: maj 3 min 1 rcv 4
12-31 11:12:22.641  2223  2252 D vndksupport: Loading /vendor/lib/hw/android.hardware.graphics.mapper@2.0-i
mpl.so from current namespace instead of sphal namespace.
12-31 11:12:22.642  2223  2252 D vndksupport: Loading /vendor/lib/hw/gralloc.vbox86.so from current namespa
ce instead of sphal namespace.
12-31 11:12:22.643  2223  2252 D         : createUnique: call
12-31 11:12:22.643  2223  2252 D         : HostConnection::get() New Host Connection established 0xe7571b60
, tid 2252
12-31 11:12:22.650  2223  2252 D         : HostComposition ext GL_OES_EGL_image_external_essl3 GL_OES_verte
x_array_object ANDROID_EMU_gles_max_version_3_1
12-31 11:12:22.651  2223  2252 E eglCodecCommon: goldfish_dma_create_region: could not obtain fd to device!
 fd -1 errno=2
12-31 11:12:24.482  2223  2223 W ActivityThread: handleWindowVisibility: no activity for token android.os.B
inderProxy@df81b58
12-31 11:12:30.229  2223  2223 W ActivityThread: handleWindowVisibility: no activity for token android.os.B
inderProxy@a11ef60
12-31 11:12:33.676  2223  2223 W ActivityThread: handleWindowVisibility: no activity for token android.os.B
inderProxy@8297294
12-31 11:13:03.126  2223  2223 W ActivityThread: handleWindowVisibility: no activity for token android.os.B
inderProxy@95f848c
12-31 11:13:07.861  2223  2223 W ActivityThread: handleWindowVisibility: no activity for token android.os.B
inderProxy@5195249
12-31 11:16:40.093  2223  2308 W com.revo.evabs: Accessing hidden method Landroid/os/Process;->setArgV0(Lja
va/lang/String;)V (light greylist, JNI)
12-31 11:16:51.607  2223  2223 D CONGRATZ!: EVABS{a_dynamic_h00k}E
```



As the script and debug log executed, press the **MAP AREA** button back to trigger the process together with the hooking. Once the function hooked triggered, the flag will appear in the debug log. Done the last challenge and it's a wrap!

In conclusion, as all challenge done and all flag catches, it proved that Mobexler VM capable on solely used it to do penetration testing and assessments. Even though EVABS itself does not cater all of mobile common vulnerability and OWASP listed risk, it still gets the job done. For Mobexler itself, it loaded with many other tools that may not fully utilized on conducting testing for these purposes. It's a powerful and reliable VM on using it as a platform to do pentest also other assessments. It even has its own checklist for Android and iOS application testing. Feel free to explore this great experience.

Additional references may refer here:

1. Mobexler Checklist[5]
2. Hooking with Frida[6]
3. OWASP mobile guide[7]

Tags: #Mobile #Pentest #Assessment #Mobexler #EVABS #Extreme #Vulnerable #Android #Application #CTF

Contact:-

Github: github.com/Ap0k4L1p5

Email: prof.apokalips@protonmail.com

[5] https://mobexler.com/checklist.htm#android
[6] https://book.hacktricks.xyz/mobile-apps-pentesting/android-app-pentesting/frida-tutorial/frida-tutorial-2
[7] https://mobile-security.gitbook.io/mobile-security-testing-guide/android-testing-guide/0x05h-testing-platform-interaction