2-Research Device & Emulator Setup

1- ADB (Android Debug Bridge)

tool used to interact or debug our target device or emulator

ADB Basics

```
adb devices (lists connected devices)
adb root (restarts adbd with root permissions)
adb start-server (starts the adb server)
adb kill-server (kills the adb server)
adb reboot (reboots the device)
adb devices -l (list of devices by product/model)
adb shell (starts the backround terminal)
exit (exits the background terminal)
adb help (list all commands)
adb -s <deviceName> <command> (redirect command to specific device)
adb -d <command> (directs command to only attached USB device)
adb -e <command> (directs command to only attached emulator)
```

Package Installation

```
adb shell install <apk> (install app)
adb shell install <path> (install app from phone path)
adb shell install -r <path> (install app from phone path)
adb shell uninstall <name> (remove the app)
```

Paths

```
/data/data/<package>/databases (app databases)
/data/data/<package>/shared_prefs/ (shared preferences)
/data/app (apk installed by user)
/system/app (pre-installed APK files)
/mmt/asec (encrypted apps) (App2SD)
/mmt/emmc (internal SD Card)
/mmt/adcard (external/Internal SD Card)
/mmt/adcard/external_sd (external SD Card)
adb shell ls (list directory contents)
adb shell ls -s (print size of each file)
adb shell ls -R (list subdirectories recursively)
```

File Operations

```
adb push <local> <remote> (copy file/dir to device)
adb pull <remote> <local> (copy file/dir from device)
run-as <package> cat <file> (access the private package files)
```

Phone Info

```
adb get-state (print device state)
adb get-serialno (get the serial number)
adb shell dumpsys iphonesybinfo (get the IMEI)
adb shell netstat (list TCP connectivity)
adb shell pwd (print current working directory)
adb shell dumpsys battery (battery status)
adb shell pm list features (list phone features)
adb shell service list (list all services)
adb shell dumpsys activity <package>/<activity> (activity info)
adb shell ps (print process status)
adb shell wm size (displays the current screen resolution)
dumpsys window windows | grep -E 'mCurrentFocus|mFocusedApp' (print current app's opened activity)
```

Package Info

```
adb shell list packages (list package names)
adb shell list packages -r (list package name + path to apks)
adb shell list packages -3 (list third party package names)
adb shell list packages -s (list only system packages)
adb shell list packages -u (list package names + uninstalled)
adb shell dumpsys package packages (list info on all apps)
adb shell dump <name> (list info on one package)
adb shell path <package> (path to the apk file)
```

Configure Settings Commands

```
adb shell dumpsys battery set level <n> (change the level from 0 to 100)
adb shell dumpsys battery set status<n> (change the level to unknown, charging, discharging, not charging or full)
adb shell dumpsys battery reset (reset the battery)
adb shell dumpsys battery set usb <n> (change the status of USB connection. ON or OFF)
adb shell wm size WxH (sets the resolution to WxH)
```

Device Related Commands

```
adb reboot-recovery (reboot device into recovery mode)
adb reboot fastboot (reboot device into recovery mode)
adb shell screencap -p "/path/to/screenshot.png" (capture screenshot)
adb shell screenrecord "/path/to/record.mp4" (record device screen)
adb backup -apk -all -f backup.ab (backup settings and apps)
adb backup -apk -shared -all -f backup.ab (backup settings, apps and shared storage)
adb backup -apk -nosystem -all -f backup.ab (backup only non-system apps)
adb restore backup.ab (restore a previous backup)
adb shell am start|startservice|broadcast <INTENT>[<COMPONENT>]
-a <ACTION> e.g. android.intent.action.VIEW
-c <CATEGORY> e.g. android.intent.category.LAUNCHER (start activity intent)
adb shell am start -a android.intent.action.VIEW -d URL (open URL)
adb shell am start -t image/* -a android.intent.action.VIEW (opens gallery)
```

Logs

```
adb logcat [options] [filter] [filter] (view device log)
adb bugreport (print bug reports)
```

Permissions

```
adb shell permissions groups (list permission groups definitions)
adb shell list permissions -g -r (list permissions details)
```

PACKAGES

List all packages

```
pm list packages
```

Uninstall

```
pm uninstall file.name
```

Install

```
pm install package.name
```

INTENTS

URI

```
am start -a android.intent.action.VIEW -d https://github.com
```

Mime Type and and Extra string

```
am start -a "android.intent.action.SEND" --es "android.intent.extra.TEXT"
"Hello World" -t "text/plain"
```

Activity

```
am start -n "your.application.packagename/path.to.the.Activity"
```

Activity with extras

```
am start -n "your.application.packagename/path.to.the.Activity" - e "key"
"data"
```

Service

```
am startservice -n "com.example.application/.BackgroundService"
```

Broadcast with Action

```
am broadcast -a "android.intent.action.PACKAGE_FIRST_LAUNCH" -d
"com.example.application"
```

SQLITE

```
cd data/data/<your-package-name>/databases/
```

```
sqlite3 <your-db-name>.db
```

Commands

TABLES

```
.tables
```

SCHEMA

```
.schema tablename
```

QUERY

```
SELECT * FROM tablename;
```

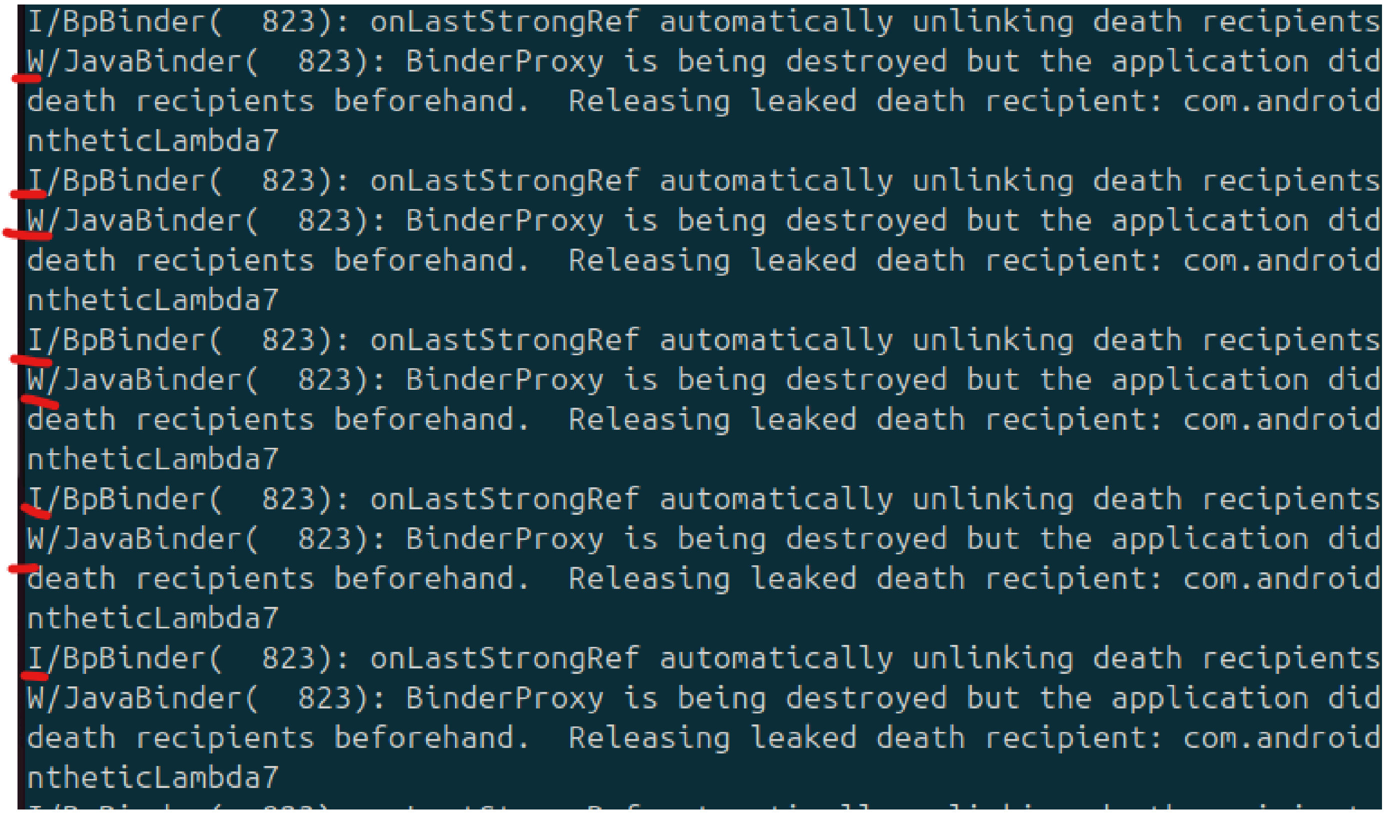
HELP

```
.help
```

adb logcat هنا دي ينستخدمها لعرض الرسائل و logs اللي بتتسجل اثنا تشغيل التطبيق

brief Show priority, tag, and PID of the process issuing the message. لو استخدمنا option ده

```
adb logcat -v brief
```



```
I/BpBinder(  823): onLastStrongRef automatically unlinking death recipients
W/JavaBinder(  823): BinderProxy is being destroyed but the application did
death recipients beforehand.  Releasing leaked death recipient: com.android
ntheticLambda7
I/BpBinder(  823): onLastStrongRef automatically unlinking death recipients
W/JavaBinder(  823): BinderProxy is being destroyed but the application did
death recipients beforehand.  Releasing leaked death recipient: com.android
ntheticLambda7
I/BpBinder(  823): onLastStrongRef automatically unlinking death recipients
W/JavaBinder(  823): BinderProxy is being destroyed but the application did
death recipients beforehand.  Releasing leaked death recipient: com.android
ntheticLambda7
I/BpBinder(  823): onLastStrongRef automatically unlinking death recipients
W/JavaBinder(  823): BinderProxy is being destroyed but the application did
death recipients beforehand.  Releasing leaked death recipient: com.android
ntheticLambda7
I/BpBinder(  823): onLastStrongRef automatically unlinking death recipients
W/JavaBinder(  823): BinderProxy is being destroyed but the application did
death recipients beforehand.  Releasing leaked death recipient: com.android
ntheticLambda7
```

هنا اه الحرف اللي في الاول ده بيعرف اه نوع الرسالة ودي الانواع

V Verbose (default for <tag>)

D Debug (default for '*')

I Info

W Warn

E Error

F Fatal

S Silent (suppress all output)

تاني حاجة بعد اول حرف بتعرفنا ايه هي نوع Service بعد كده رقم process id بعد كده uid

```
➜  module2 adb logcat -v brief "MainActivity:V *:S"
here give me all Verbose logs on MainActivity
```

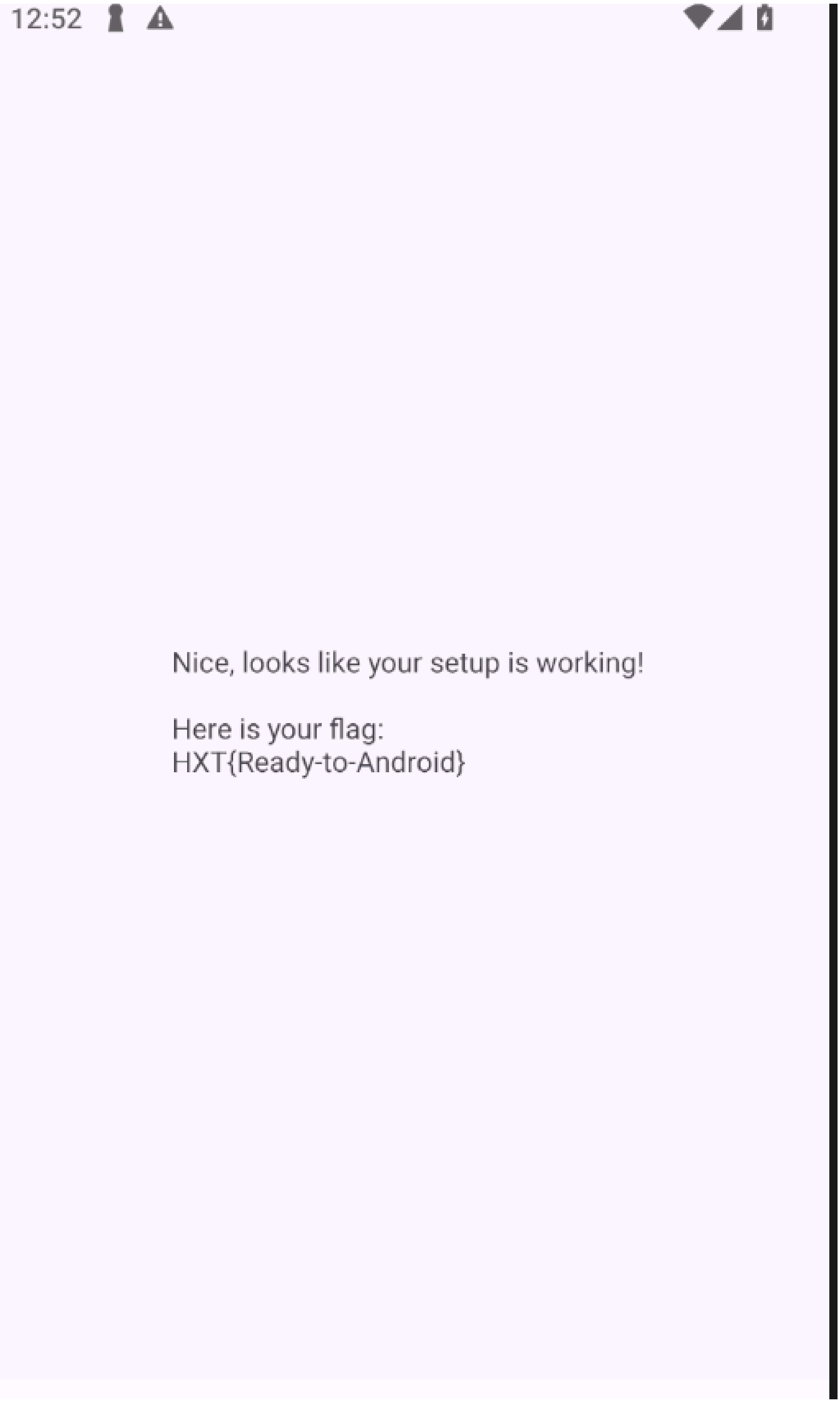
Solving Chellange

1- install app on emulator by using adb install

```
➜  module2 adb install adb_test_application.apk
Performing Streamed Install
Success
```

2- open app

we get first flag HXT{Ready-to-Android}

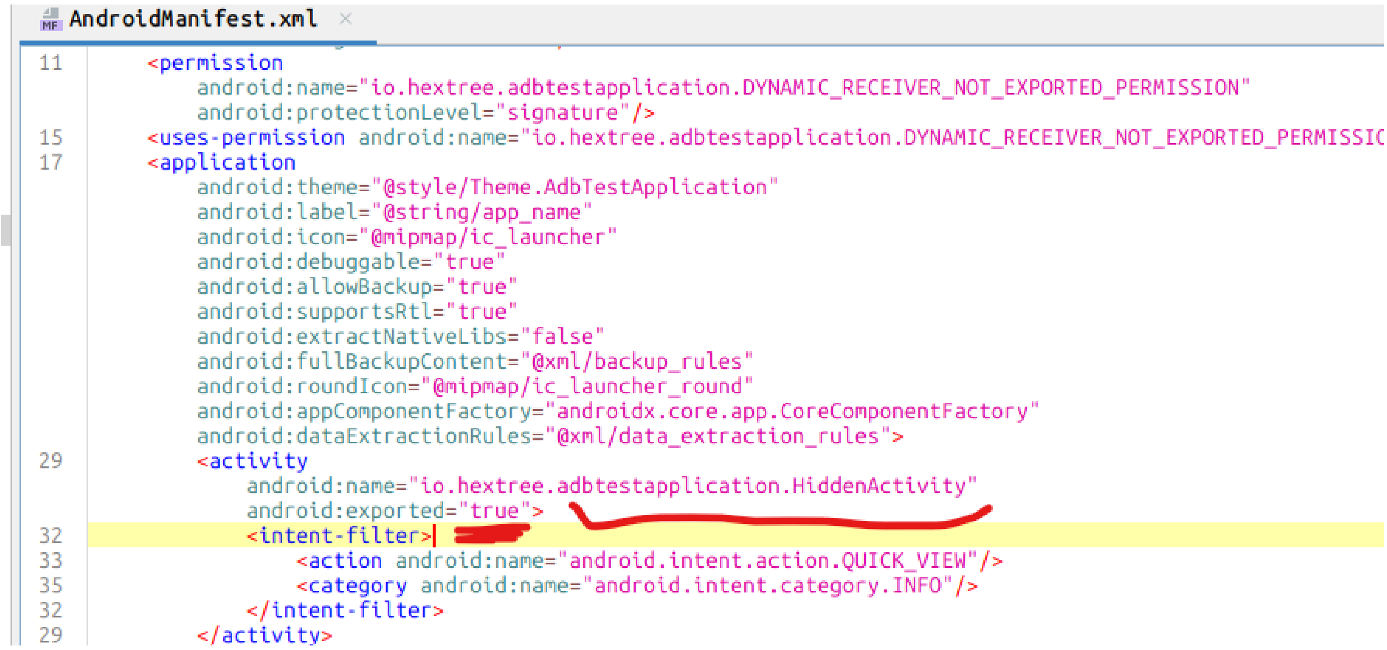


12:52

Nice, looks like your setup is working!

Here is your flag:
HXT{Ready-to-Android}

3- open app with jadx-gui to show AndroidMeniFast.xml file to find hidden activities

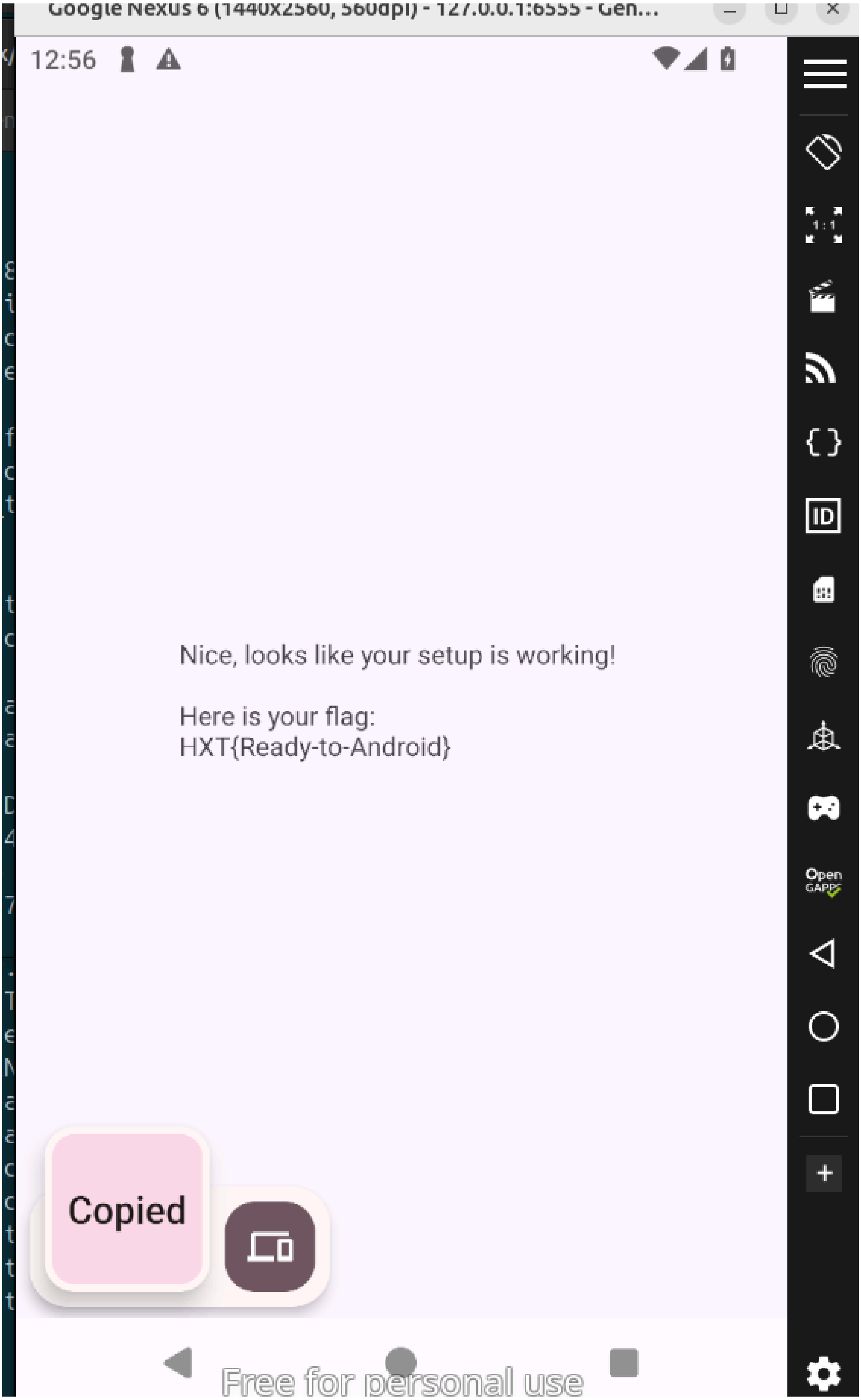


AndroidManifest.xml

```
    <permission
        android:name="io.hextree.adbtestapplication.DYNAMIC_RECEIVER_NOT_EXPORTED_PERMISSION"
        android:protectionLevel="signature"/>
    <uses-permission android:name="io.hextree.adbtestapplication.DYNAMIC_RECEIVER_NOT_EXPORTED_PERMISSIC
    <application
        android:theme="@style/Theme.AdbTestApplication"
        android:label="@string/app_name"
        android:icon="@mipmap/ic_launcher"
        android:debuggable="true"
        android:allowBackup="true"
        android:supportsRtl="true"
        android:extractNativeLibs="false"
        android:fullBackupContent="@xml/backup_rules"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:appComponentFactory="androidx.core.app.CoreComponentFactory"
        android:dataExtractionRules="@xml/data_extraction_rules">
        <activity
            android:name="io.hextree.adbtestapplication.HiddenActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.QUICK_VIEW"/>
                <category android:name="android.intent.category.INFO"/>
            </intent-filter>
        </activity>
```

هو هنا معمول له exported يعني هو بيسمح ان activity or app يستخدمه علشان كده هنفتح الصفحة دي واحنا في MainActivity

before



Google Nexus 6 (1440x2560, 560dpi) - 127.0.0.1:6555 - Gen...

12:56

Nice, looks like your setup is working!

Here is your flag:
HXT{Ready-to-Android}

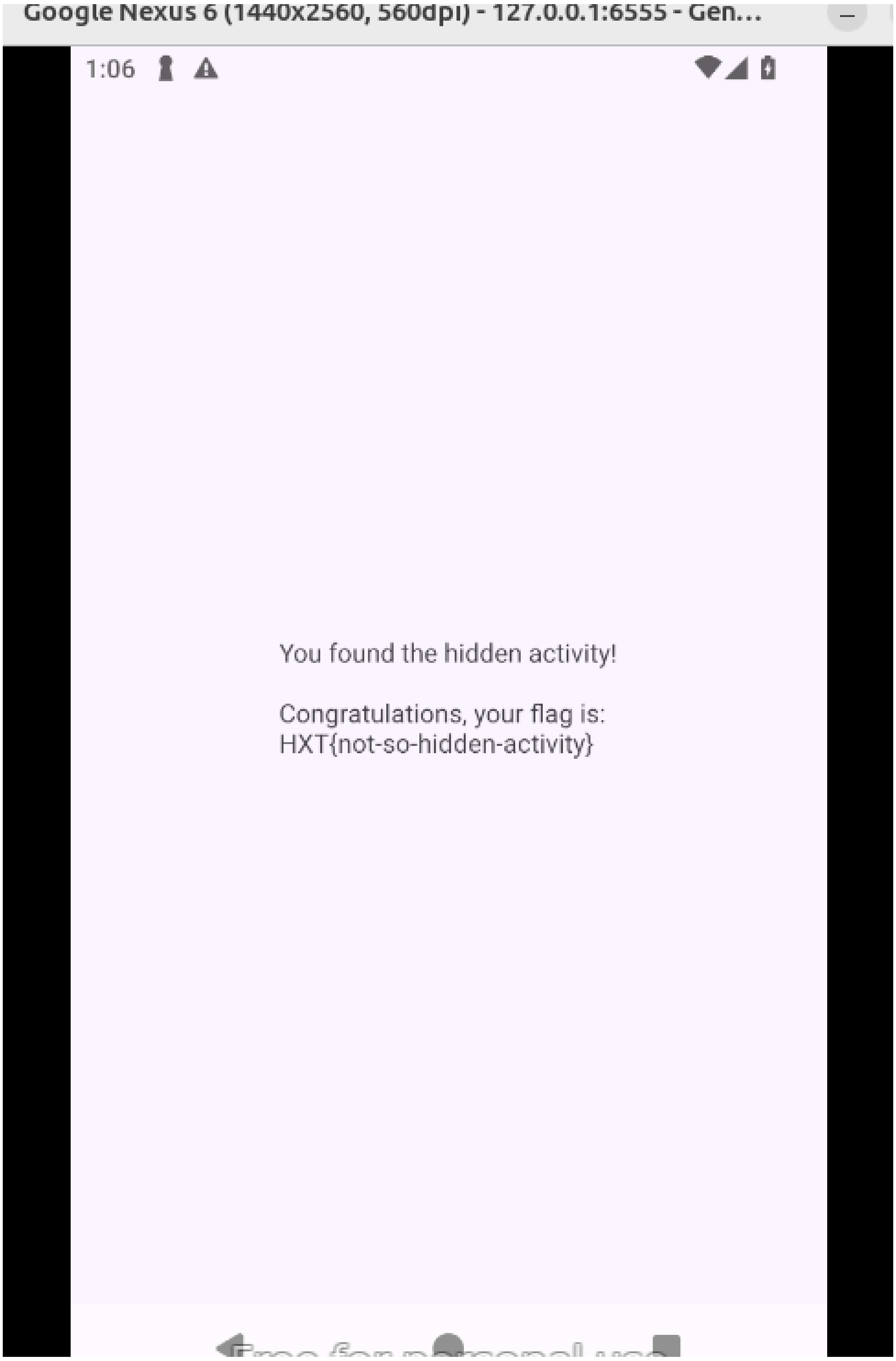
start hidden activity

```
vbox86p:/ $ am start-activity -n
io.hextree.adbtestapplication/io.hextree.adbtestapplication.HiddenActivity
```

```
Starting: Intent { cmp=io.hextree.adbtestapplication/.HiddenActivity }
vbox86p:/ $
```

after

flag is HXT{not-so-hidden-activity}


Google Nexus 6 (1440x2560, 560dpi) - 127.0.0.1:6555 - Gen...
1:06

You found the hidden activity!

Congratulations, your flag is:
HXT{not-so-hidden-activity}

4- get the last flag in logcat

flag is HXT{log-all-the-cats}

```
➜  module2 adb logcat "MainActivity:V *:S"
```

```
--------- beginning of main
05-09 13:28:14.775  4661  4661 V MainActivity: Congratulations, you found the log! Your flag is: HXT{log-all-the-cats}
```