FORENSICS
LAB SERIES

Lab 17: Android Logical Acquisition

Material in this Lab Aligns to the Following Certification Domains/Objectives	
Certified Cyber Forensics Professional (CCFP) Objectives	Computer Hacking Forensic Investigator (CHFI) Objectives
4: Digital Forensics	20: Mobile Forensics

Document Version: 2016-08-17

Contents

Introduction

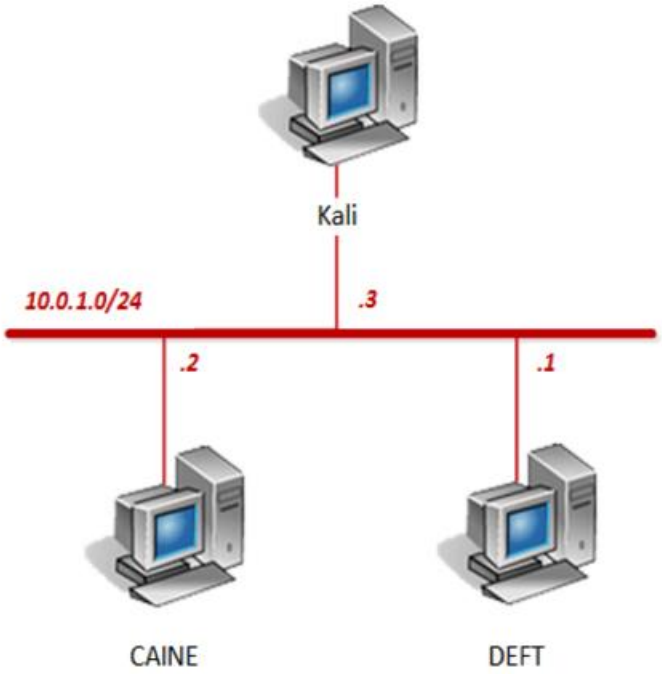
This lab will utilize the *Android-SDK* simulator in an effort to acquire a logical acquisition of the *Android* OS. The logical data will include contacts, call logs, SMS/MMS, application data, and system logs.

Objective

In this lab, you will be conducting forensic practices using various tools. You will be performing the following tasks:

1. Launching Android SDK
2. Pulling DB Information
3. Creating a Logical Acquisition
4. Parsing Information

Pod Topology



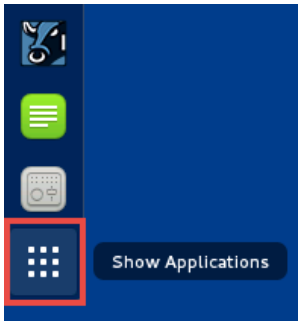
Lab Settings

The information in the table below will be needed in order to complete the lab. The task sections below provide details on the use of this information.

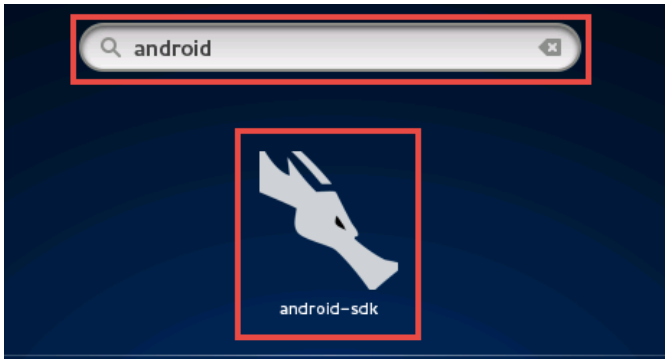
Virtual Machine	**IP Address**	**Account** (if needed)	**Password** (if needed)
DEFT	10.0.1.1	deft	password
CAINE	10.0.1.2	caine	
Kali	10.0.1.3	root	toor

1 Launching Android SDK

1. Click on the **Kali** graphic on the *topology page* to open the VM.
2. Login using `root` as the *username* and `toor` as the *password*.
3. Click on the **Show Applications** icon located in the left pane.

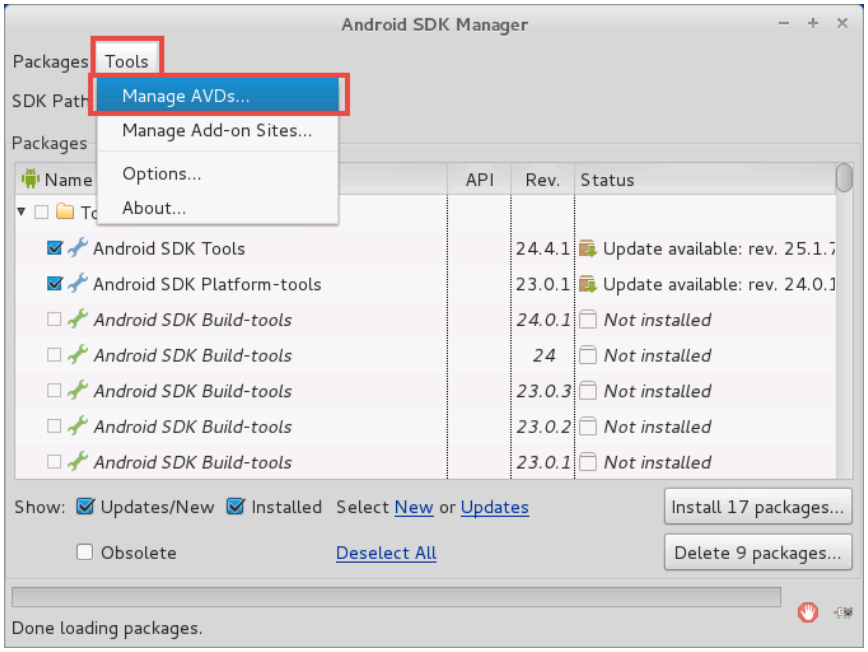


4. Type `android` in the search field located at the top. From the search results, click on the **android-sdk** icon to launch the *Android SDK* application.

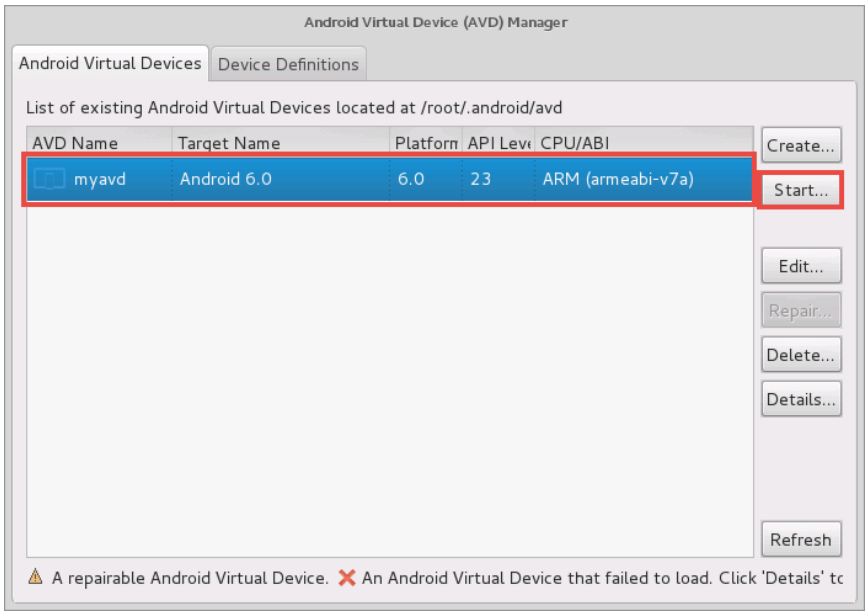


When the *Android SDK Manager* is launched, wait 1-2 minutes until the progress bar on the bottom is finished.

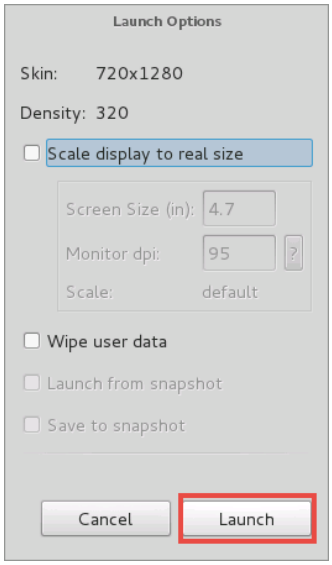
5. Using the *Android SDK Manager*, click on **Tools** and select **Manage AVDs**.



6. Select **myavd** from the middle pane and click **Start**.



7. In the *Launch Options* dialog window, leave the defaults set and click **Launch**.



8. Open a new terminal window by clicking on the **Terminal** icon.

9. Using the terminal, navigate to the **/usr/share/android-sdk/platform-tools/** directory by typing the command below followed by pressing the **Enter** key.

```
cd /usr/share/android-sdk/platform-tools
```

10. Enter the command below to connect to an Android emulator device using *Android Debug Bridge* (*adb*).

```
./adb devices
```

```
root@Kali2:/usr/share/android-sdk/platform-tools# ./adb devices
List of devices attached
* daemon not running. starting it now on port 5037 *
* daemon started successfully *
emulator-5554   offline

root@Kali2:/usr/share/android-sdk/platform-tools#
```

11. Initiate the same command once more.

```
./adb devices
```

```
root@Kali2:/usr/share/android-sdk/platform-tools# ./adb devices
List of devices attached
emulator-5554   device

root@Kali2:/usr/share/android-sdk/platform-tools#
```

12. Enter the command below to launch a Unix shell with the connected device.

```
./adb shell
```

```
root@Kali2:/usr/share/android-sdk/platform-tools# ./adb shell
root@generic:/ #
```

Notice the superuser status of being the *root* user.

2 Pulling DB Information

1. In this task, logical data for contacts, call logs, SMS-MMS, application data, and system logs and information are desired. Start with acquiring the SMS database. Exit the *adb* shell by typing the command below followed by pressing the **Enter** key.

```
exit
```

```
root@generic:/ # exit
root@Kali2:/usr/share/android-sdk/platform-tools#
```

2. Initiate a manual pull of the SMS database by entering the command below.

```
./adb pull /data/data/com.android.providers.telephony/databases/mmssms.db
```

```
root@Kali2:/usr/share/android-sdk/platform-tools# ./adb pull /data/data/com.andr
oid.providers.telephony/databases/mmssms.db
104 KB/s (102400 bytes in 0.958s)
root@Kali2:/usr/share/android-sdk/platform-tools#
```

3. List the files in the current directy.

```
ls -l
```

```
root@Kali2:/usr/share/android-sdk/platform-tools# ls -l
total 3460
-rwxr-xr-x 1 root root 1221540 Mar  9 13:22 adb
drwxr-xr-x 2 root root    4096 Mar  9 13:22 api
drwxr-xr-x 2 root root    4096 Mar 20 12:10 databases
-rwxr-xr-x 1 root root   58920 Mar  9 13:22 dmtracedump
-rwxr-xr-x 1 root root  211200 Mar  9 13:22 etc1tool
-rwxr-xr-x 1 root root  556700 Mar  9 13:22 fastboot
-rwxr-xr-x 1 root root   11427 Mar  9 13:22 hprof-conv
drwxr-xr-x 2 root root    4096 Mar  9 13:22 lib
-rw-r--r-- 1 root root  102400 Aug  4 10:20 mmssms.db
-rw-r--r-- 1 root root  220534 Mar  9 13:22 NOTICE.txt
drwxr-xr-x 2 root root    4096 Mar 20 12:10 shared_prefs
-rw-r--r-- 1 root root   16512 Mar  9 13:22 source.properties
-rwxr-xr-x 1 root root 1109318 Mar  9 13:22 sqlite3
drwxr-xr-x 3 root root    4096 Mar  9 13:22 systrace
root@Kali2:/usr/share/android-sdk/platform-tools#
```

Notice the *mmssms.db* database file was pulled from the *Android* system to the local Kali system.

4. Enter the command below to pull the complete directory with other databases.

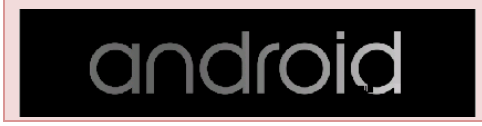
```
./adb pull /data/data/com.android.providers.telephony
```

```
root@Kali2:/usr/share/android-sdk/platform-tools# ./adb pull /data/data/com.andr
oid.providers.telephony
pull: building file list...
pull: /data/data/com.android.providers.telephony/databases/mmssms.db-journal ->
./databases/mmssms.db-journal
pull: /data/data/com.android.providers.telephony/databases/mmssms.db -> ./databa
ses/mmssms.db
pull: /data/data/com.android.providers.telephony/databases/telephony.db-journal
-> ./databases/telephony.db-journal
pull: /data/data/com.android.providers.telephony/databases/telephony.db -> ./dat
abases/telephony.db
pull: /data/data/com.android.providers.telephony/databases/HbpcdLookup.db-journa
l -> ./databases/HbpcdLookup.db-journal
pull: /data/data/com.android.providers.telephony/databases/HbpcdLookup.db -> ./d
atabases/HbpcdLookup.db
pull: /data/data/com.android.providers.telephony/shared_prefs/build-id.xml -> ./
shared_prefs/build-id.xml
pull: /data/data/com.android.providers.telephony/shared_prefs/preferred-apn.xml
-> ./shared_prefs/preferred-apn.xml
8 files pulled. 0 files skipped.
251 KB/s (218905 bytes in 0.850s)
root@Kali2:/usr/share/android-sdk/platform-tools#
```

Notice all the databases for the telephony directory have been pulled.

5. Before moving on to the next step, change focus to the Android simulator and very whether the operating system has booted yet.

If you see the text "*android*" on the screen, that means the *Android* simulator is still loading. Wait an additional 5 minutes for the *Android OS* to boot.



3 Creating a Logical Acquisition

1. Once the *Android* simulator is fully booted, initiate a backup of the *Android* system that doesn't require root access but does require physical access to the phone or this case, virtual access. Type the *adb* command below to start the backup process followed by pressing the **Enter** key.

```
./adb backup shared -all
```

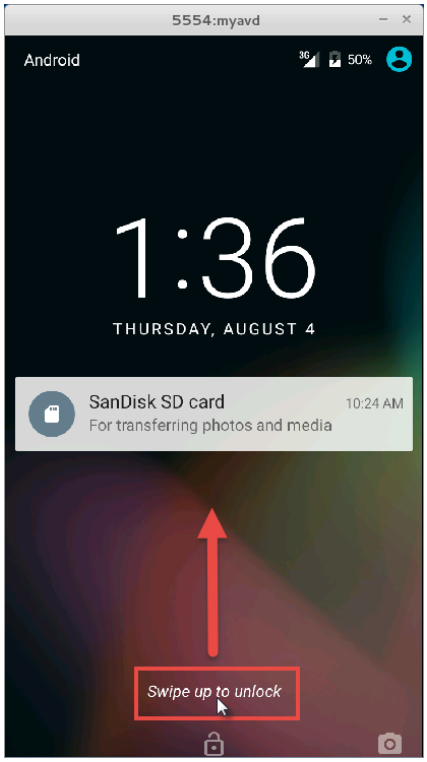
```
root@Kali2:/usr/share/android-sdk/platform-tools# ./adb backup -shared -all
Now unlock your device and confirm the backup operation.
```

When prompted to unlock the device, do not press the *Enter* key again, instead proceed to the next step.

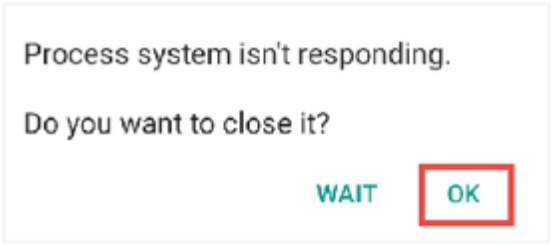
Command Breakdown:

-shared = chooses whether or not to back up data from shared storage and the SD card
-all = include all applications for which backups are enabled

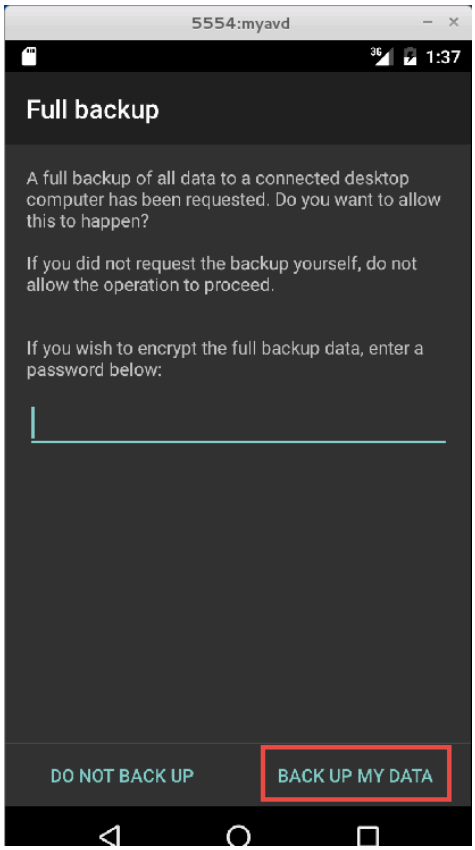
2. Change focus to the **Android** simulator window and unlock the device. To unlock the device, simply click-and-hold on the bottom portion of the screen (where it says *Charging*) and move the mouse in an upwards movement.



3. If prompted with *Process system isn't responding*, click **OK** to continue.



4. Once the *Android* device is unlocked, notice a *Full backup* window appear. Click **Back Up My Data**.



A notification will appear on-screen signaling that the backup has started. Wait 1 minute for the backup to finish.

5. Change focus to the **terminal**.

6. Enter the command below to list the files in the current directory. Notice the *backup.ab* file has been created.

```
ls -l
```

```
root@Kali2:/usr/share/android-sdk/platform-tools# ls -l
total 3508
-rwxr-xr-x 1 root root 1221540 Mar  9 13:22 adb
drwxr-xr-x 2 root root    4096 Mar  9 13:22 api
-rw-r----- 1 root root   22606 Aug  4 14:30 backup.ab
drwxr-xr-x 2 root root    4096 Aug  4 14:12 databases
-rwxr-xr-x 1 root root   58920 Mar  9 13:22 dmtracedump
-rwxr-xr-x 1 root root  211200 Mar  9 13:22 etc1tool
-rwxr-xr-x 1 root root  556700 Mar  9 13:22 fastboot
-rwxr-xr-x 1 root root   11427 Mar  9 13:22 hprof-conv
drwxr-xr-x 2 root root    4096 Mar  9 13:22 lib
-rw-r--r-- 1 root root  102400 Aug  4 14:11 mmssms.db
-rw-r--r-- 1 root root  220534 Mar  9 13:22 NOTICE.txt
-rw-r----- 1 root root   22606 Aug  4 14:27 -shared
drwxr-xr-x 2 root root    4096 Aug  4 14:12 shared_prefs
-rw-r--r-- 1 root root   16512 Mar  9 13:22 source.properties
-rwxr-xr-x 1 root root 1109318 Mar  9 13:22 sqlite3
drwxr-xr-x 3 root root    4096 Mar  9 13:22 systrace
root@Kali2:/usr/share/android-sdk/platform-tools#
```

7. The backup of the data has been captured. Create a copy of the *backup.ab* file to be used with a parser program by entering the command below.

```
cp backup.ab /root/Downloads/android-backup-extractor-20151102-bin/
```

```
root@Kali2:/usr/share/android-sdk/platform-tools# cp backup.ab /root/Downloads/a
ndroid-backup-extractor-20151102-bin/
root@Kali2:/usr/share/android-sdk/platform-tools#
```


4 Parsing Information

1. Change to the directory of the parser tool by entering the command below.

```
cd /root/Downloads/android-backup-extractor-20151102-bin
```

```
root@Kali2:/usr/share/android-sdk/platform-tools# cd /root/Downloads/android-bac
kup-extractor-20151102-bin
root@Kali2:~/Downloads/android-backup-extractor-20151102-bin#
```

2. List the files in the current directory and confirm that the *backup.ab* file is present.

```
ls -l
```

```
root@Kali2:~/Downloads/android-backup-extractor-20151102-bin# ls -l
total 6868
-rw-r--r-- 1 root root 6167026 Nov  2  2015 abe.jar
-rw-r--r-- 1 root root    8337 Oct 13  2015 adb-split-extraction.sh
-rw-r--r-- 1 root root    6786 Oct 13  2015 adb-split-no-extraction.sh
-rw-r----- 1 root root   22606 Aug  4 14:34 backup.ab
drwxr-xr-x 4 root root    4096 Nov  2  2015 Doc
-rw-r--r-- 1 root root     583 Dec 31  2012 LICENSE.TXT
drwxr-xr-x 2 root root    4096 Nov  2  2015 perl
-rw-r--r-- 1 root root   15984 Nov  2  2015 README.TXT
drwxr-xr-x 2 root root    4096 Nov  2  2015 star-1.5.2-i686-pc-cygwin
drwxr-xr-x 2 root root    4096 Nov  2  2015 star-1.5.3-i686-pc-cygwin
drwxr-xr-x 2 root root    4096 Nov  2  2015 star-ubuntu-lucid
-rw-r--r-- 1 root root  773441 Oct 20  2015 tar-bin-split.jar
-rw-r--r-- 1 root root       9 Nov  2  2015 VERSION.TXT
root@Kali2:~/Downloads/android-backup-extractor-20151102-bin#
```

3. Convert the *backup.ab* file to a *.tar* file so that it can be extracted. Enter the command below.

```
java -jar abe.jar unpack backup.ab backup.tar
```

```
root@Kali2:~/Downloads/android-backup-extractor-20151102-bin# java -jar abe.jar
unpack backup.ab backup.tar
root@Kali2:~/Downloads/android-backup-extractor-20151102-bin#
```


4. View the contents of the *backup.tar* file by entering the command below.

```
tar -tvf backup.tar
```

```
root@Kali2:~/Downloads/android-backup-extractor-20151102-bin# tar -tvf backup.tar
-rw------- 1000/1000      2414 1969-12-31 18:00 apps/android/_manifest
-rw------- 1000/1000        99 2016-08-04 14:19 apps/android/r/wallpaper_info.xml
-rw------- 1000/1000      2426 1969-12-31 18:00 apps/com.android.browser/_manifest
-rw------- 1000/1000      2430 1969-12-31 18:00 apps/com.android.calculator2/_manifest
-rw------- 1000/1000      2427 1969-12-31 18:00 apps/com.android.calendar/_manifest
-rw-rw---- 10019/10019     135 2016-08-04 14:21 apps/com.android.calendar/sp/calendar_alerts.xml
-rw-rw---- 10019/10019     130 2016-03-09 14:07 apps/com.android.calendar/sp/_has_set_default_values.xml
-rw-rw---- 10019/10019     721 2016-03-09 14:07 apps/com.android.calendar/sp/com.android.calendar_preferences.xm
l
-rw------- 1000/1000      2424 1969-12-31 18:00 apps/com.android.camera/_manifest
-rw------- 1000/1000      2437 1969-12-31 18:00 apps/com.android.captiveportallogin/_manifest
-rw------- 1000/1000      2427 1969-12-31 18:00 apps/com.android.contacts/_manifest
-rw------- 1000/1000      2431 1969-12-31 18:00 apps/com.android.customlocale2/_manifest
-rw------- 1000/1000      2429 1969-12-31 18:00 apps/com.android.deskclock/_manifest
-rw------- 1000/1000       398 1969-12-31 18:00 apps/com.android.deskclock/_meta
-rw-rw---- 10023/10023   28672 2016-03-09 14:07 apps/com.android.deskclock/db/alarms.db
-rw------- 10023/10023    8720 2016-03-09 14:07 apps/com.android.deskclock/db/alarms.db-journal
-rw-rw---- 10023/10023     230 2016-08-04 14:21 apps/com.android.deskclock/sp/com.android.deskclock_preferences.
xml
-rw------- 1000/1000      2429 1969-12-31 18:00 apps/com.android.development/_manifest
```

5. Change to the **/usr/share/android-sdk/platform-tools** directory by entering the command below.

```
cd /usr/share/android-sdk/platform-tools
```

```
root@Kali2:~/Downloads/android-backup-extractor-20151102-bin# cd /usr/share/andr
oid-sdk/platform-tools
root@Kali2:/usr/share/android-sdk/platform-tools#
```

6. Generate a simple list of services that were running on the Android device at the time of backing up by entering the command below.

```
./adb shell service list
```

```
root@Kali2:/usr/share/android-sdk/platform-tools# ./adb shell service list
Found 99 services:
0       carrier_config: [com.android.internal.telephony.ICarrierConfigLoader]
1       phone: [com.android.internal.telephony.ITelephony]
2       isms: [com.android.internal.telephony.ISms]
3       iphonesubinfo: [com.android.internal.telephony.IPhoneSubInfo]
4       simphonebook: [com.android.internal.telephony.IIccPhoneBook]
5       isub: [com.android.internal.telephony.ISub]
6       telecom: [com.android.internal.telecom.ITelecomService]
7       imms: [com.android.internal.telephony.IMms]
8       media_projection: [android.media.projection.IMediaProjectionManager]
9       launcherapps: [android.content.pm.ILauncherApps]
10      fingerprint: [android.hardware.fingerprint.IFingerprintService]
11      trust: [android.app.trust.ITrustManager]
12      media_router: [android.media.IMediaRouterService]
13      media_session: [android.media.session.ISessionManager]
14      restrictions: [android.content.IRestrictionsManager]
15      print: [android.print.IPrintManager]
```


7. Generate a more detailed list of services with the help of a tool called *Dumpsys* by entering the command below.

```
./adb shell dumpsys activity services
```

```
root@Kali2:/usr/share/android-sdk/platform-tools# ./adb shell dumpsys activity s
ervices
ACTIVITY MANAGER SERVICES (dumpsys activity services)
  Last ANR service:
ServiceRecord{f8ac26a u0 com.android.systemui/.SystemUIService}
    intent={cmp=com.android.systemui/.SystemUIService}
    packageName=com.android.systemui
    processName=com.android.systemui
    baseDir=/system/priv-app/SystemUI/SystemUI.apk
    dataDir=/data/user/0/com.android.systemui
    app=ProcessRecord{c4d95a5 660:com.android.systemui/u0a13}
    createTime=-1m25s103ms startingBgTimeout=--
    lastActivity=-35s376ms restartTime=-35s376ms createdFromFg=false
    startRequested=true delayedStop=false stopIfKilled=false callStart=false las
tStartId=1
    executeNesting=1 executeFg=false executingStart=-35s375ms
    restartCount=0 restartDelay=-- nextRestartTime=-36s502ms crashCount=0
    Delivered Starts:
    #0 id=1 dur=-1m20s546ms dc=1
      intent=Intent { cmp=com.android.systemui/.SystemUIService }
```

8. Identify the user's last login on the *Android* by issuing the command below.

```
./adb shell dumpsys user
```

```
root@Kali2:/usr/share/android-sdk/platform-tools# ./adb shell dumpsys user
Users:
  UserInfo{0:Owner:13} serialNo=0
    Created: <unknown>
    Last logged in: +1h27m25s643ms ago
root@Kali2:/usr/share/android-sdk/platform-tools#
```

Note that last login timestamps will differ upon the time when the *Android* simulator was booted.

9. Check the *WiFi* connections the *Android* has made by entering the command below.

```
./adb shell dumpsys wifi
```

```
root@Kali2:/usr/share/android-sdk/platform-tools# ./adb shell dumpsys wifi
Wi-Fi is disabled
Stay-awake conditions: 1
mMulticastEnabled 0
mMulticastDisabled 0
mInIdleMode false
mScanPending false
WifiController:
 total records=3
 rec[0]: time=08-04 14:18:13.308 processed=DefaultState org=ApStaDisabledState d
est=<null> what=155652(0x26004)
 rec[1]: time=08-04 14:18:13.708 processed=DefaultState org=ApStaDisabledState d
est=<null> what=155652(0x26004)
 rec[2]: time=08-04 14:20:36.654 processed=DefaultState org=ApStaDisabledState d
est=<null> what=155660(0x2600c)
curState=ApStaDisabledState
mScreenOff false
mDeviceIdle false
```

Notice that *WiFi* has been disabled.

10. Check what applications were using battery power on the *Android*. Enter the command below.

```
./adb shell dumpsys batterystats
```

```
root@Kali2:/usr/share/android-sdk/platform-tools# ./adb shell dumpsys batterysta
ts
Battery History (0% used, 2140 used of 256KB, 7 strings using 298):
                    0 (9) RESET:TIME: 2016-03-09-14-02-20
                    0 (2) 050 status=charging health=good plug=ac temp=0 volt=0
+running +plugged +charging
          +1m01s288ms (2) 050 +screen
          +1m01s288ms (3) 050 +wake_lock=-1:"screen" brightness=bright
          +1m01s508ms (2) 050 brightness=medium
          +2m06s084ms (2) 050 +sensor
          +3m08s666ms (2) 050 +user=0:"0"
          +3m08s667ms (3) 050 +audio +userfg=0:"0"
          +3m09s339ms (2) 050 -audio
          +3m10s781ms (3) 050 +audio +top=u0a43:"com.android.sdksetup"
          +3m11s925ms (2) 050 -audio
          +3m12s899ms (2) 050 +audio
          +3m13s199ms (2) 050 -audio
          +3m20s573ms (3) 050 +audio -top=u0a43:"com.android.sdksetup"
          +3m22s359ms (2) 050 -audio
          +3m26s134ms (2) 050 +audio
          +3m27s243ms (2) 050 -audio
          +3m30s606ms (2) 050 +top=u0a7:"com.android.launcher3"
```

11. Close all **PC Viewers** and end the reservation to complete the lab.

