Lab 5 - Android forensics Lab

Table of Contents

Lab 5 - Android forensics Lab....1
5.0 Objectives....1
5.1 Methodology....1
5.1.0 Use the Android Emulator for forensics....2
Preparation of the AVD....2
Preparation of an user account and evidence....2
Notes about ADB....3
References....3
5.1.1 Logical data extraction with an in-house software agent....3
5.1.2 Logical file extraction....4
References....4
5.1.3 Physical data extraction....6
References....6
5.1.4 Evidence analysis....8
References....8
5.2 Reporting....10
5.2.1 Logical data extraction with an in-house software agent....10
5.2.2 Logical file extraction....10
5.2.3 Physical data extraction....10
5.2.4 Evidence analysis....11
5.3 General references for all Android....12
5.4 Lab feedback....12
Appendix 1 – apk installation....13
Appendix 2 – user data (SMS and GPS)....13
Appendix 3 – Droid Forensics....14
Appendix 4 – Problems with ADB....14
Appendix 5 – TWRP....15

5.0 Objectives

The main objective of this lab is to learn how more advanced forensic investigations can be done on Android smart phones. Specific objectives of the lab are:

1. Learn how to use the Android tool-chain for forensics.
2. Learn how to obtain logical and physical dumps from Android devices.
3. How to investigate evidence in Android and other smart phone devices.

5.1 Methodology

Notes about included chapters from books, papers and general preparations for the tasks

Högskolan Dalarna
Röda vägen 3
781 88 BORLÄNGE
Telefon: 023-77 80 00
Telefax: 023-77 80 50
URL: http://www.du.se/

- This lab needs the Android development tool chain. See the free chapter 3 from the book "Android Forensics and Mobile Security" for more info about installing and setup.
- Later on when dealing with imaging, logical and physical techniques read thru the free chapter 6 from the book "Android Forensics and Mobile Security".
- Another good paper to read about Android forensics is: SSDDFJ_V4_1_Lessard_Kessler.pdf.
- If you got an Android phone yourself you can use it for some of the tasks in this lab. Otherwise the Android emulator can be used for almost all of the tasks.
- If you find errors, new methods, utilities or other improvements please inform me!

5.1.0 Use the Android Emulator for forensics

Preparation of the AVD

- First create an AVD (Android Virtual Device) with a SD card of around 1,5 GB storage (it depends on the size of the emulators or phones data partition). **The SD card must be able to store a dd image of the data partition**. You can check the size of you data partition with the commands in task 5.2.3 a).

- You should have an AVD with API-19 (Android 4.4.x) or higher since from this version **hw.useext4 = yes** is set in the AVD:s **hardware-qemu.in**i file. You can find the hardwareqemu.ini file at: "C:\Users\<username>\.android\avd\<AVD-name.avd>" on a Windows computer.

- If you use an emulator you **must** use an **ARM emulator** for some of the tasks to work in the lab. Therefore when we run native programs they need to be cross-compiled (since we are running them on the ARM architecture).

- If we use an older emulator or set the **useext4** variable to "no" we are stuck with the YAFFS2 file system which make extraction and parsing a lot more difficult. Most, if not all phones and tablets use the ext4 file system since 2011.
  Some limited new devices have begun to use the F2FS (Flash-Friendly File System) initially developed by Samsung Electronics: http://en.wikipedia.org/wiki/F2FS.

Preparation of an user account and evidence

We are going to add/generate all possible kinds of user data to the AVD. See references for more help.

We should at least deal with the possible data that are present in the emulator as:

- Browser (Web)
- Calendar
- Camera
- Email
- Messaging (SMS/MMS)
- People (Contacts) and
- Telephone (Call Logs)

Högskolan Dalarna
Röda vägen 3
781 88 BORLÄNGE
Telefon: 023-77 80 00
Telefax: 023-77 80 50
URL: http://www.du.se/

from the Android device.

Optionally you can also install other third party apps and apps that require a Google account to gather evidence from, see appendix 1 – apk installation.

Now fill the AVD with data as contacts, photos, SMS, phone calls, web browser activity, calendar and e-mail content etc. It may be possible to capture Google apps data including Maps/Navigation content as well.
You can use the available apps in the system image and the functions in DDMS (SMS, calls etc.) for most of these tasks, see appendix 2 – user data.

Notes about ADB

The Android Debug Bridge (ADB) is very useful during an investigation and you should really strive to be an ADB master! See Appendix 4 - Problems with ADB if you got problems.

References

- There are many ADB managers available on the internet if you don't like the console. Try a search for “android adb manager xda”. A couple of old programs that are able to manage the Android phone via the computer and ADB are: Droid Explorer: http://de.codeplex.com/ and QtADB: http://qtadb.wordpress.com/.

- If you need support with Android or some app you can find it here: https://support.google.com/.

5.1.1 Logical data extraction with an in-house software agent

When you have generated enough forensic user data you can install and run the attached Android program Droid Forensics (DroidForensics.apk) from the apk folder.

Note that the output will be placed in the [external memory]/droidforensics folder.

When the case screen (start activity) is visible you can fill in the case data. When this is finished you press the tabs button so you can begin capture data in the different tabs

More about Droid Forensics and problem solving in the appendix 3 – Droid Forensics.

When you have finished capture you can go back to the case screen (start activity) and press Menu so an OptionsMenu is shown containing:

- About,
- List Content Providers (can be saved as a HTML file to the SD card as well with a menu option when selected in this screen),
- Save MD5 sums and
- Zip case to file.

Högskolan Dalarna
Röda vägen 3
781 88 BORLÄNGE
Telefon: 023-77 80 00
Telefax: 023-77 80 50
URL: http://www.du.se/

The "Save MD5 sums" command writes a md5 sum for every generated evidence file into [datetime]-md5sums.txt. It should be run after capture of evidence and before the "Zip case to file" command which is the last command to be run.
After the "Zip case to file" command is executed the zip file with all the evidence text files can be pulled from the [external memory] root for further investigation on your computer. The path to the file on the emulator is usually: /storage/sdcard/<datetime>-case.zip.

Note that using a software agent in this way queering the Android phones content providers (accessible databases) is the only possible way to gather user data from a non-rooted Android device!

5.1.2 Logical file extraction

This part of the lab should ideally take place when you have used your AVD for some time generating information and possible deleted some user data and so on. **To perform this lab on a physical phone and actually get some valuable data, the phone must have rooted access via either apps, ADB or both.**

The most interesting information that are stored in Android is located in the /data/data/<app-name>/databases folder. As for example the web browser databases are located at: /data/data/com.android.browser/databases. The databases are usually in a SQLite binary file format http://www.sqlite.org/fileformat.html. The key-value XML format files in the shared_prefs folder or files in the files folder may also be of some interest.

If you start an ADB shell (or a device console apk app with root access) you can list all databases on the phone with:
 # ls -l /data/data/*/databases
If the output is long it is better to do it from your computer shell adding "adb shell" first in front of the command as:
 # adb shell ls -l /data/data/*/databases

You may also need to have BusyBox installed during the investigation. BusyBox can be installed in two ways. Either as a native cross-compiled binary by yourself or as an APK file that download, install and configure everything.

To actually be able to be root in the first place you need to find an exploitable vulnerability in the system in some way. For a small number of phones this is easy, but usually it is very hard to obtain root privileges.

Help for this task is found in Chapter 6 from Android Forensics and Mobile Security - Logical techniques.

References

Rooting
General and specific information about hacking and rooting Android devices:
http://wiki.cyanogenmod.org/ > Devices | Support | Learning Center.
If you select a device and then read the "See How to Build CyanogenMod for the ..." info you get the specific instructions for a device, otherwise it is found on the XDA developers forum.

Högskolan Dalarna
Röda vägen 3
781 88 BORLÄNGE
Telefon: 023-77 80 00
Telefax: 023-77 80 50
URL: http://www.du.se/

BusyBox
BusyBox is something that you install on your Android to give you some additional handy Linux/Unix based commands. You may need BusyBox installed because some commands are not available or missing in the phones system image. Here is some more info on what BusyBox is: http://www.busybox.net/about.html

I have included instructions for BusyBox installation in the BusyBox folder. In this lab it is probably enough to copy the busybox binary to a tmpfs filesystem and create symbolic links from the commands you want to use to the busybox binary, for example:

```
# ln -s ./busybox dd
```

Note that you need the correct architecture of the busybox binary, ARM or x86. More instructions are found in section 5.1.3.

Superuser APK
A superuser app is a guard program which will allow you to select which apps you allow to elevate their rights to run as root if your phone is rooted. If you do not install a superuser program that listens and catching these system calls, malware could perform root activities without your knowledge!
You do not need this program on the emulator.
The best superuser app is Chainfires SuperSU: https://play.google.com/store/apps/details?id=eu.chainfire.supersu

Nandroid dumps
Nandroid: http://www.makeuseof.com/tag/what-is-a-nandroid-backup-and-how-exactly-does-it-work/ only do logical capture (only files) dumps. This however needs that we boot up the phone on a recovery image with the nandroid tool included or install an app as: https://play.google.com/store/apps/details?id=com.h3r3t1c.onnandbup. It is not possible to do this in the emulator.
Many custom recovery images have nandroid built-in for backup reasons. If the file system in the phone is extX-based it usually creates *.tar.gz archive files, but if it is YAFFS2 it creates YAFFS image files.

Nandroid have opted for an incremental blob (Binary Large Object) way of storing the files. This means that a full backup is taken the first time and only changed files are backuped the second time and so on. Nandroid and other versions of backup tools may include incremental, compressed and encrypted backups as well.

To extract files from an YAFFS2 nandroid image we can use the program unyaffs. According to the extremely large manual (grin): “Unyaffs is a program to extract files from a yaffs file system image. Now it can only extract images created by mkyaffs2image.”

Logical file extraction with custom recovery and nandroid
If a suitable custom recovery is installed with nandroid like functionality a file based backup of the whole device is possible regardless of the file system on the device.

If the boot loader is unlocked (does not require a signed update package) it is in general possible to install a custom recovery, otherwise it must be exploited in some way.

Högskolan Dalarna
Röda vägen 3
781 88 BORLÄNGE
Telefon: 023-77 80 00
Telefax: 023-77 80 50
URL: http://www.du.se/

It is however not always necessary to root the phone to flash a custom recovery. In general one can say that if the flashing must be performed by the phone in a higher run level you need to have root privileges on the device.

If the flashing is performed by the host computer from download/fastboot mode or recovery mode it is not necessary. You can reach the two modes from ADB with:

```
# adb reboot bootloader
# adb reboot recovery
```

SSH access with SSHelper

With a tool as the included SSHelper one can use SSH against the device instead of ADB. In this way you can perform investigations over wireless easily.

5.1.3 Physical data extraction

The same prerequisites are basically needed for acquiring physical dumps as for the logical dumping of data. This applies to needed tools as well.

To perform this lab on a physical device and actually get some valuable data the device must have rooted access via either apps, ADB or both. To perform dumping of partitions follow the guidelines in the included Chapter 6 from Android Forensics and Mobile Security - AFPhysical technique.

If you put tools on the device you should use a tmpfs folder with rw (read write) access which is stored in RAM to avoid writing to NAND. Example how find a tmpfs mount point in an ADB shell (in this case we found /dev):

```
# mount
tmpfs /dev tmpfs rw,seclabel,nosuid,relatime,mode=755 0 0
```

References

Dumping partitions with dd (or other tool) - problems

- We should only use dd on eMMC NAND. By using dd on RAW NAND (if its even possible) we do not get any spare area/OOB data if it is a YAFFS2 file system.
- If the phone have no external SD card we can dump the image to we must find another solution. If we have root it is possible to install netcat on the phone and dump the image over the network to a computer with netcat in listen mode.
- For this to work we either need to cross-compile netcat with the NDK (Native Developer Kit) and put it on our phone, or use the netcat (nc) bundled with BusyBox (Android/ARM version), which is included in the labs Busybox folder.
- If you use a x86 emulator or x86 Android device you can use one of the x86 (ix86) binary packages here instead: http://www.busybox.net/downloads/binaries/latest/
- The netcat bundled in the Android emulator is a limited netcat version which cannot perform pipelining (http://en.wikipedia.org/wiki/Pipeline_%28Unix%29) of files.

Högskolan Dalarna
Röda vägen 3
781 88 BORLÄNGE
Telefon: 023-77 80 00
Telefax: 023-77 80 50
URL: http://www.du.se/

Hints to solve the task

- If we have no custom recovery but root we can copy the bundled busybox binary to a forensically safe storage place on the phone (tmpfs). Then we create symbolic links to the binary (we do not need any more commands than nc and dd), see 5.1.2 references. After this is done we can dump the user data partition to your computer.

- If we have access to a custom recovery as TWRP they usually have built-in access to busybox commands. ADB and IP works in recovery mode so we do not need to do any preparations at all in this case, more than get the system running.

Team Win Recovery Project (TWRP)

It is possible to run a custom recovery with the Android emulator with a specially crafted recovery image from Team Win Recovery Project (TWRP): http://teamw.in/project/twrp2/169. It is possible with any API level together with a specially crafted goldfish_2.6_kernel as long it is an **ARM** AVD. The binaries to make it work is attached in the TWRP.custom-recovery folder. TWRP have busybox built-in from start.

See appendix 5 for my SDK/AVD configuration. Note that "Use Host GPU" is off! I did not get the newest TWRP image and kernel to work with my system.

Start the AVD you configured and named to "TWRP" from the folder where you have put the recovery image together with the kernel and a batch file that looks something like this in Windows:

```
REM http://stackoverflow.com/questions/5034076/what-does-dp0-mean-and-how-does-it-work
cd /d %~dp0
C:\android-sdk-windows\tools\emulator -avd TWRP -ramdisk twrp-2.8.0.1-twrp.img -kernel goldfish_2.6_kernel
pause
```

Just for information the Android kernel history for physical devices looks something like this:

```
Android Version    |API Level  |Linux Kernel in Android Open Source Project (AOSP)
-------------------------------------------------
1.5   Cupcake      |3          |2.6.27
1.6   Donut        |4          |2.6.29
2.0/1 Eclair       |5-7        |2.6.29
2.2.x Froyo        |8          |2.6.32
2.3.x Gingerbread  |9, 10      |2.6.35
3.x.x Honeycomb    |11-13      |2.6.36
4.0.x Ice Cream San|14, 15     |3.0.1
4.1.x Jelly Bean   |16         |3.0.31
4.2.x Jelly Bean   |17         |3.4.0
4.3.x Jelly Bean   |18         |3.4.39
4.4.x Kitkat       |19         |3.4.x
5.0.x Lollipop     |21         |3.4.x
```


Högskolan Dalarna
Röda vägen 3
781 88 BORLÄNGE

Telefon: 023-77 80 00
Telefax: 023-77 80 50
URL: http://www.du.se/

Dumping with MTD utils
We should only use nanddump on RAW NAND since by using nanddump we also get the spare area/OOB data if it is a YAFFS2 file system.

If you got time and are interested you can try it out. MTD utils is found on [server]\embedded_forensics\Android-tools\MTD-Utils for Android (including nanddump and nandwrite). There is probably newer releases available on the internet.

The previous mentioned TWRP custom recovery image have nanddump included under /sbin as well.

5.1.4 Evidence analysis

Consult the relevant guides and documentation for how to do the investigation, as for example SSDDFJ_V4_1_Lessard_Kessler.pdf. If you find out about any other non-described method or tool please inform me about this in the report.

If you have worked against a later emulator (>= API-19) or a newer Android telephone you should have a couple of physical ext4 dd images (at least userdata and perhaps cache) and a directory tree with files from a logical extraction.

References

Find deleted records in SQLite databases
As earlier mentioned it is in the SQLite databases one will find the majority of data: https://www.sqlite.org/datatype3.html that could be of interest in an investigation. SQLite databases or any database for that matter works just as the Windows registry or as files on a hard drive. If something is deleted it still is there until it is overwritten with new data or the admin has performed the SQLite vacuum command on the database.

You can try to delete some of the data you filled the phone with as calls, SMS, contacts etc. and try to "recover" (show that it is there) this data from the dumped database.

From the SQLite FAQ at: http://www.sqlite.org/faq.html#q20
"(20) I accidentally deleted some important information from my SQLite database. How can I recover it?
If you have a backup copy of your database file, recover the information from your backup.

If you do not have a backup, recovery is very difficult. You might be able to find partial string data in a binary dump of the raw database file. Recovering numeric data might also be possible given special tools, though to our knowledge no such tools exist.

SQLite is sometimes compiled with the SQLITE_SECURE_DELETE option which overwrites all deleted content with zeros. If that is the case then recovery is clearly impossible.

Recovery is also impossible if you have run VACUUM since the data was deleted. If SQLITE_SECURE_DELETE is not used and VACUUM has not been run, then some of the

Högskolan Dalarna
Röda vägen 3
781 88 BORLÄNGE
Telefon: 023-77 80 00
Telefax: 023-77 80 50
URL: http://www.du.se/

deleted content might still be in the database file, in areas marked for reuse. But, again, there exist no procedures or tools that we know of to help you recover that data."

SQLite specific references

- As you see there is no undelete or known recovery method for the deleted data in the SQLite databases. The closest work I found is the Firefox 3 History Recovery paper at: [server]\embedded_forensics\docs\Firefox 3 History Recovery.
- A former student did a thesis on this subject. I have attached his report in the docs\sqlite folder: Examensrapport Sebastian Zankl.pdf
- Various SQLite viewer tools including undelete at: [server]\embedded_forensics\common-tools\SQLite-tools
- SQLite Database File Format: http://www.sqlite.org/fileformat.html
- SQLite Datatypes:https://www.sqlite.org/datatype3.html
- SQL As Understood By SQLite: http://www.sqlite.org/lang.html
- SQLite analysis articles at: http://forensicsfromthesausagefactory.blogspot.com/2011/07/sqlite-overflow-pages-and-other-loose.html
- Researching SQLite records: https://mobileforensics.wordpress.com/2012/04/19/researching-sqlite-records/ or http://www.garykessler.net/software/index.html
- Finding and Reverse Engineering Deleted SMS Messages: http://az4n6.blogspot.se/2013/02/finding-and-reverse-engineering-deleted_1865.html
- Creating a Perl script to retrieve Android SMS: http://www.cheeky4n6monkey.blogspot.se/2013/02/creating-perl-script-to-retrieve.html
- Android SQLite & deleted record recovery: http://stackoverflow.com/questions/10566501/android-sqlite-deleted-record-recovery
- Google search "sqlite forensic android" etc.

Analysis of a YAFFS2 image

If we have to analyze an YAFFS2 image in some form we have two choices. Either try to mount the image or view the image in a viewer. To mount the image it must have the spare or OOB (Out-of-band) data preserved. It may be possible to mount an YAFFS2 nanddump image forensically in GNU/Linux with YAFFS2 support or examine the image in FTK. I have not yet been able to view a decoded YAFFS2 image myself in FTK or Mobile Phone Examiner Plus (MPE+) although support is available according to the documentation. Probably the forensic image must contain a vendor specific header?

Other tools that might work is:
https://code.google.com/p/yaffey/
https://code.google.com/p/yaffs2utils/
and so on, check with Google search.

Otherwise we remove the spare or OOB (Out-of-band) data. We must know the page size and the OOB size for this. Viaforensics have scripts for this and in lab 4 a very simple script (sa-remover.py) was bundled as well. Then we can try to carve out structures with carving capable tools as Photorec - http://www.cgsecurity.org/wiki/PhotoRec etc.

Högskolan Dalarna
Röda vägen 3
781 88 BORLÄNGE
Telefon: 023-77 80 00
Telefax: 023-77 80 50
URL: http://www.du.se/

If you like to test this methods but do not have a suitable image you can try with the DFRWS images on [server]\embedded_forensics\DFRWS.org\2011 - Android forensics. The 7z archives have no SD card image (it takes very long time to extract the tgz files with a SD card included).

5.2 Reporting

5.2.1 Logical data extraction with an in-house software agent

Hand in the generated files with evidence data from the forensic program (software agent) Droid Forensics. Please report problems if found.

5.2.2 Logical file extraction

You should at least have extracted the databases etc. that are used for your selected applications in 5.1.x and appendix 1. Please report problems if found. You will examine the databases later in task 5.2.4 with suitable tools.

a) Try or examine the

- Android Forensics Toolkit-AFT - aft.sh which need an ADB or app shell to execute. You can modify the script and build in some intelligent function for the logical dumping of /data/*. Remember to set permissions (chmod). See the readme.txt file for more info.
- And the Android Forensics Toolkit - aft.py which works the same as above script but from your computer console via ADB.

If you compare the scripts to this plain command from your computer shell:

```
# adb pull /data localFolderName
```

What is the best solution in your opinion? Is there any meaning to have a script doing this task?

b) It is possible to extract the location cache files and historically view where a device have been located a specific date time. The files are named cache.cell and cache.wifi and is located in /data/data/com.google.android.location/files on the Android device.
In the scripts/Android-locdump folder you have more information about how to perform this investigation.
Note that the location cache files are not present on the emulator and in Android versions from 4.x and newer.
Parse the location cache for wifi and cell if you have an older Android phone. Otherwise use the attached /scripts/android-locdump script on the files in the sample-test folder.

5.2.3 Physical data extraction

a) Log in to your running AVD or physical phone with ADB, then issue the commands below and report the output.

1. **List the available partitions** - `# cat /proc/partitions`
2. **List the available Memory Technology Devices** - `# cat /proc/mtd`

Högskolan Dalarna
Röda vägen 3
781 88 BORLÄNGE
Telefon: 023-77 80 00
Telefax: 023-77 80 50
URL: http://www.du.se/

3. **Disk free and file system block size** - `# df`
4. **Check the file system mount points** - `# mount`
5. Can you find **any other** informative command to run which gives you details about the storage system?

b) Dump the user data partition if possible to the sdcard with dd (can take some time), what command did you use?
Note! For optimal performance of dd it can be beneficial to use the correct cluster/block size (for example: bs=4096).

c) Pull the partition dump file from the device (can take some considerable time) and load or mount it into a forensic tool which understands the ext4 file system as FTK or FTK Imager for example. Show the folder tree with a screenshot or a tree command/view.

d) To fix the problem with no external SD card to write your dump to you have to use both of the two methods described below:

1. A more simple method only possible if we have IP access in some way. Either via the emulator or ADB (IP-address to the host computer is 10.0.2.2) or Wi-Fi. Report the steps you took to make it work with netcat (commands used).

2. And a little harder method using ADB port forwarding. Report the steps you took to make ADB port forwarding work (commands used).
   Unfortunately there are some errors in the ADB port forward method presented by the Android Forensics and Mobile Security book on page 282. You have to figure out the correct commands.
   Note it is extremely slow to capture the NAND with port forwarding! So test with a small partition or cancel rather quick to check if it works at all.

5.2.4 Evidence analysis

Since this task also is going to be an examination project in some form. Do not spend countless hours examining it (unless you plan to do it as examination project). In the examination project I expect students to have done a deeper analysis!

- Perform an investigation on your AVD (or your own Android phone) with the evidence you got. If you work against your own phone you can filter out or anonymise sensitive personal data in the report.
- Use the SQLite database programs found in references to gather existing data. Remember to also check the XML files in the shared_prefs, files in the files folder and possible other apps database folders for other interesting application you can find.
- You can use any forensically sound tool for your examination, remember FTK have support for ext4.
- Try to extract deleted data out of the SQLite databases if possible (there is no straightforward tool for this). See the references for more information about suitable tools for this. Some examples the strings command, any hex-editor or special tools as the included SQLite Parser.v2.3 or SQLite-Deleted-Records-Parser tools.
- Are there any other evidence of great forensic value in the images you have captured which I have missed to mention?

Högskolan Dalarna
Röda vägen 3
781 88 BORLÄNGE
Telefon: 023-77 80 00
Telefax: 023-77 80 50
URL: http://www.du.se/

- Make a small well written report of your evidence findings where you also documented the steps you took for the investigation and tools used.

At least 4-5 Android applications (SMS, call logs, browser, contacts etc.) which you populated with data and maybe erased some data from should be covered in your analysis.

5.3 General references for all Android

- Many of the tools mentioned and not mentioned are found at: [server]\embedded_forensics\Android-tools. Note that newer versions may exist!
- DFRWS 2011 Forensics Challenge: http://www.dfrws.org/2011/challenge/index.shtml or [server]\embedded_forensics\DFRWS.org\2011 - Android forensics
- Various Android guidelines at: [server]\embedded_forensics\docs\Android
- Also check out the sub folders of: [server]\embedded_forensics\viaforensics.com
- YAFFS (Yet Another Flash File System): http://en.wikipedia.org/wiki/YAFFS

5.4 Lab feedback

a) Was this a relevant and appropriate lab and what about length etc?
b) What corrections and/or improvements do you suggest for this lab?

Högskolan Dalarna
Röda vägen 3
781 88 BORLÄNGE
Telefon: 023-77 80 00
Telefax: 023-77 80 50
URL: http://www.du.se/

Appendix 1 – apk installation

If you not got a Google account yet you should consider getting one now since your user data is easily shared with Google cloud services in that case. In order to get this working on the emulator you need to install Gmail, Calendar, Hangouts, Google+ or some other Google application. Se instructions below.

In order to have Maps/Navigation you need Google Play Services:
http://developer.android.com/google/play-services/index.html support in the emulator.

Most of the Google programs can be found if you search for Gapps etc.
Examples:

- http://opengapps.org/
- https://www.apkmirror.com/

Unzip the Gapps file and pick the program you want (usually /system/app/ or /system/priv-app/ in the zip file). Select the same package as Android version as you got in the emulator. Then install them via the ADB install command. See below for reference.

Installation can be done in a couple of ways.
Either use: “adb.exe install <file.apk>” to install the application. If you got a physical phone attached via USB use: “adb -d install <application filename>.apk”

Reference:
adb install [-l] [-r] [-s] <file> - push this package file to the device and install it
('-l' means forward-lock the app)
('-r' means reinstall the app, keeping its data)
('-s' means install on SD card instead of internal storage)

adb uninstall [-k] <package> - remove this app package from the device
('-k' means keep the data and cache directories)

Or install the .apk from Android with a file manager as ES File Explorer.
To install via one of the file manager programs you put the .apk file on the SD card and navigate to that place and then just click on the file answering the questions popping up.
To find one of the file managers apk programs, search for them.

Appendix 2 – user data (SMS and GPS)

To send a SMS or make telephone calls in Eclipse:

4. Switch to DDMS view.
5. Navigate to the Emulator Control tab.
6. Then mark an emulator who is going to receive the SMS or call in the Devices tab and put in the Incoming number (which is the calling/sending number).
7. Then press Call or text a SMS and press Send.

Note that the emulator have its own number as well which usually is 5554 for the first running emulator. It is located up to the left in the emulator window.

To send GPS events to the emulator in Eclipse:

Högskolan Dalarna
Röda vägen 3
781 88 BORLÄNGE
Telefon: 023-77 80 00
Telefax: 023-77 80 50
URL: http://www.du.se/

1. Switch to DDMS view and mark an emulator in the Devices tab.
2. Find the Location Controls in the Emulator Control tab.
3. Click the GPX tab and click Load GPX.
4. Locate and select the GPX file.
5. Mark the file and click Play to begin sending coordinates to the Emulator.

In the forensic folder there is a suitable GPX file, but you can use your own as well.

With DDMS you can browse certain parts of the phones file system. You can for example access the SD card. You can pull and push files from/to the phones memory. You can also delete files and create folders with the 4 buttons located just above the text Info in the File Explorer tab in DDMS.

To add many contacts at once you can export contacts from another resource (program, gadget, webmail etc.) in vCard (.vcf) format and push the generated file to the AVD:s SD card. Then start the Contacts program in Android and import the contacts.
Gmail can for example export contacts in this format. This is not needed if you have installed Gmail in the emulator since it automatically will sync with Google cloud services in that case.

Appendix 3 – Droid Forensics

If you do the lab against your physical phone you probably have more than enough with data. For the emulator to work you have to use the apps a little to have some data to collect.

Droid Forensics compability/bug problems
If Droid Forensics hangs, long press the Home or Recents button (depending on Android version) and kill the program by swiping the app out of RAM.
Alternatively by entering the Settings > Applications > Manage Applications. Find Droid Forensics in the list and press it, then press the Force Stop button.
Next time you run Droid Forensics check one or two boxes at a time when capturing evidence to find the function call that makes the program hang.

The ones who can hang or take very long time is:

- Phone Status tab (Phone)
  - Console cmds (date, netstat, ps, ls, …), works on emulator but not on my current Android phone version. Perhaps need root permissions in order to run on device?
- Phone Evidence tab (PIM)
  - If your emulator is slow or you got many contacts the ContactsContract.*.* may take some time to complete
  - downloads/download, this is a permission problem – Force closes the program
- Apps Evidence
  - The only one who may work is: Google Talk(IM), all other functions in this tab are not implemented yet. Do not use them.

Appendix 4 – Problems with ADB

If the emulator or device is offline when entering the adb commands you can be forced to kill ADB and start over.

Högskolan Dalarna
Röda vägen 3
781 88 BORLÄNGE
Telefon: 023-77 80 00
Telefax: 023-77 80 50
URL: http://www.du.se/

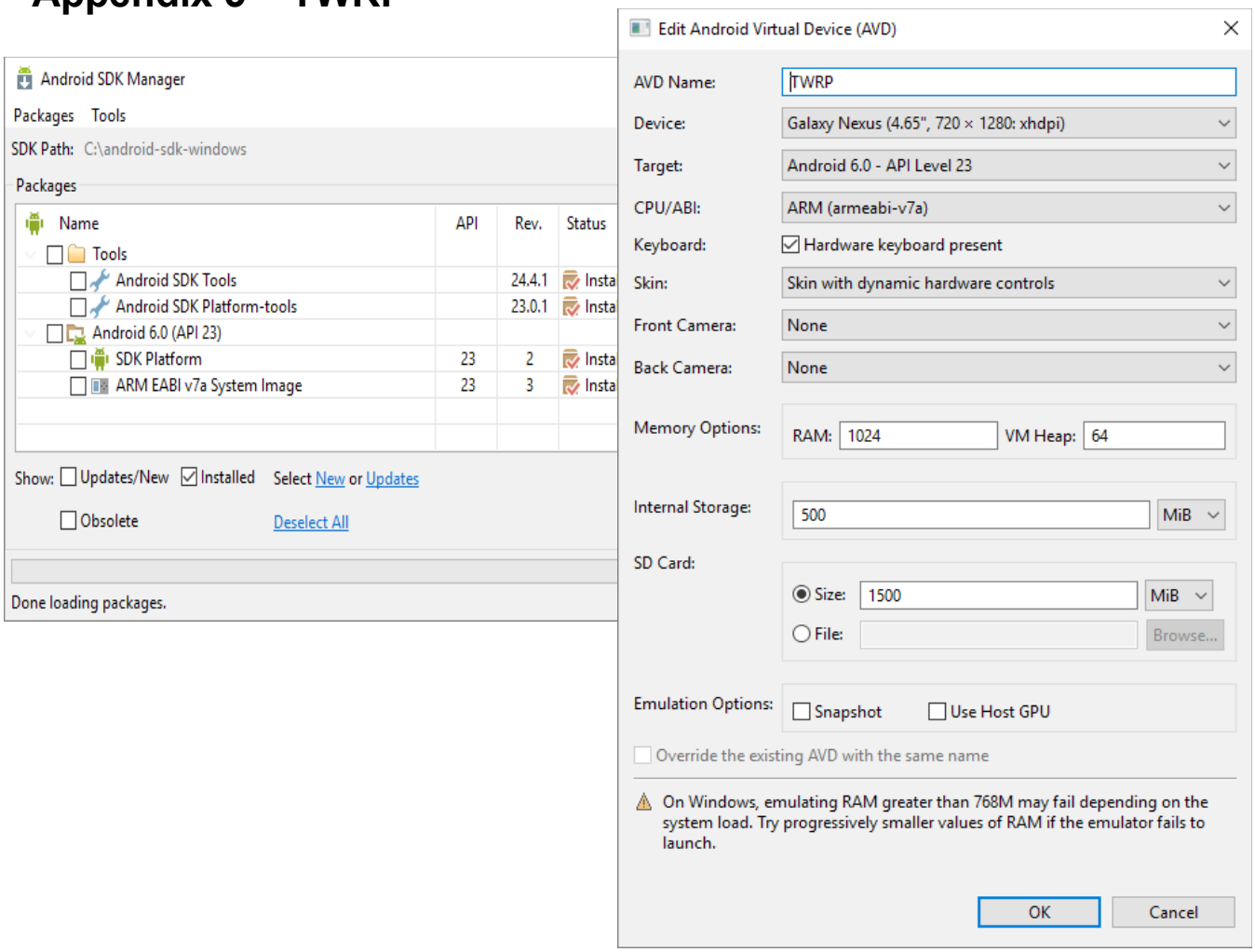
If you got connection problems as:
D:\hjo\kurs\lab>adb shell
adb server is out of date. killing...
ADB server didn't ACK
* failed to start daemon *
error:

D:\hjo\kurs\lab>adb shell
adb server is out of date. killing...
* daemon started successfully *
** daemon still not runningerror: cannot connect to daemon

Check here: http://forums.androidcentral.com/android-sideload-wonder-machine/74171-daemon-not-running-adb-server-didnt-ack.html

Basically you have to wait until the connections on port 5037 times out, check with netstat -ano or tcpview. Just be cool and grab a cup of coffee and don't issue any more ADB command :-)

Appendix 5 – TWRP



Högskolan Dalarna
Röda vägen 3
781 88 BORLÄNGE
Telefon: 023-77 80 00
Telefax: 023-77 80 50
URL: http://www.du.se/