Smartphone Forensics

Dr. Michael Spreitzenbarth

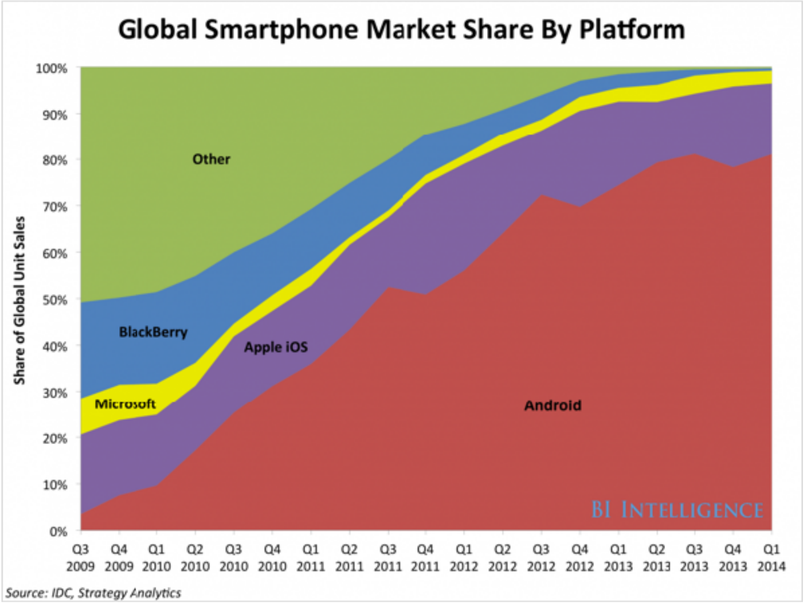
Success and Issues of the Android OS

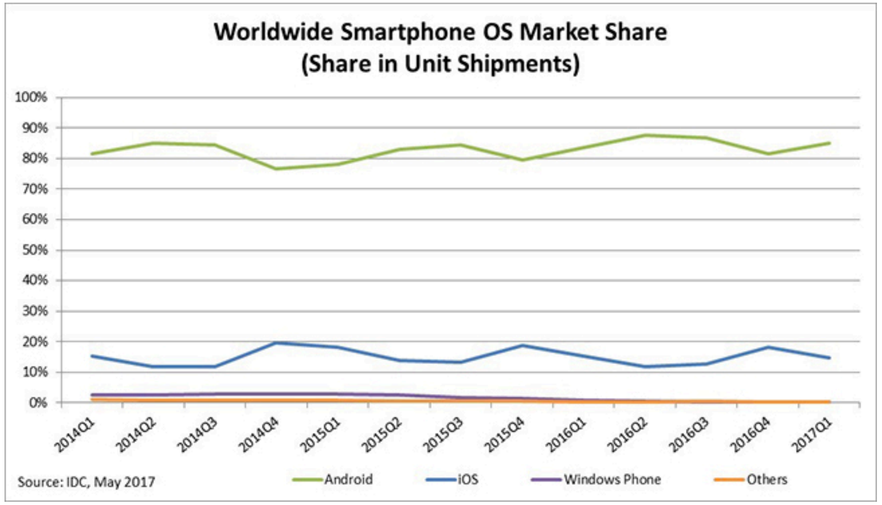
(Google) Android OS

- This OS is based on Linux
- It is mainly used on smartphones and tablets, but can also be found on TVs, kitchen appliances and also within cars (entertainment systems)

The Success of Android

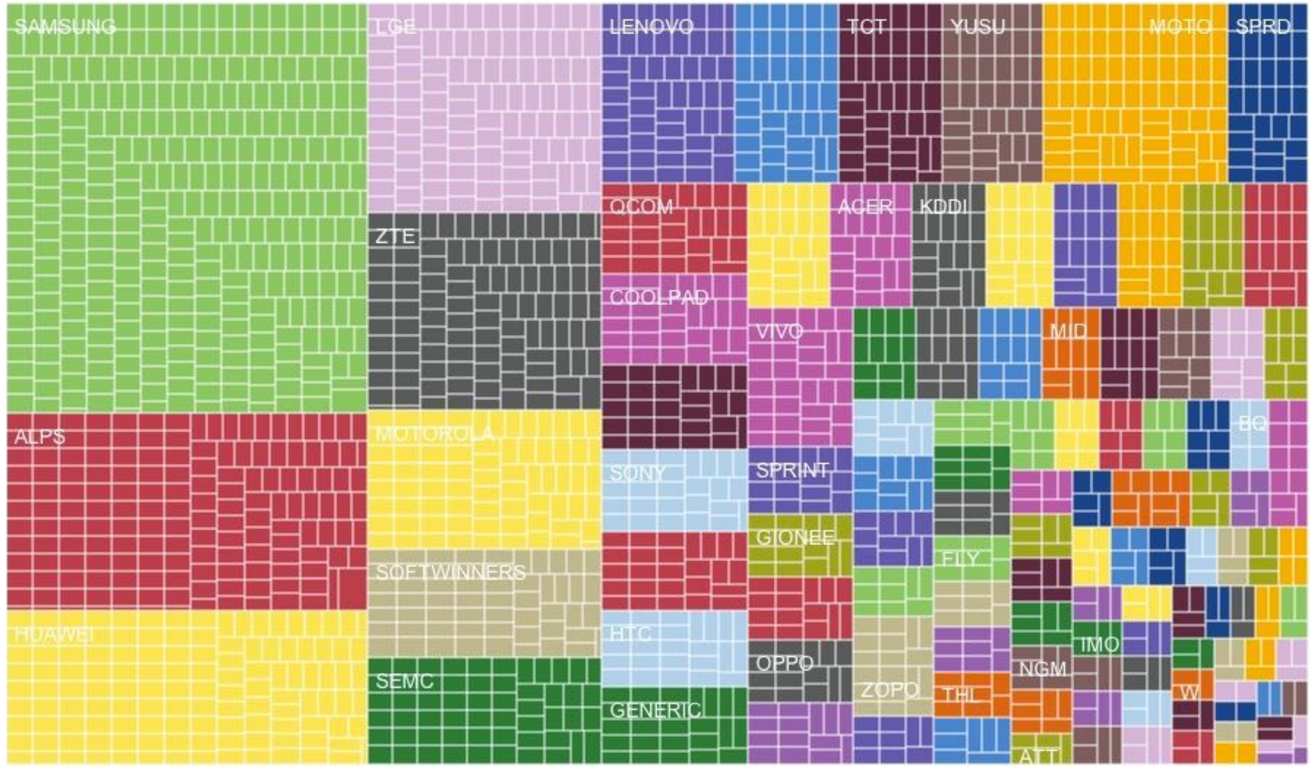
- The majority of the system is open-source
- Easy to develop apps because of Java
- Very few limitations and restrictions when it comes to publishing the apps
- Low costs for developers
- Devices are much cheaper than for iOS, BB10 or Windows Phone



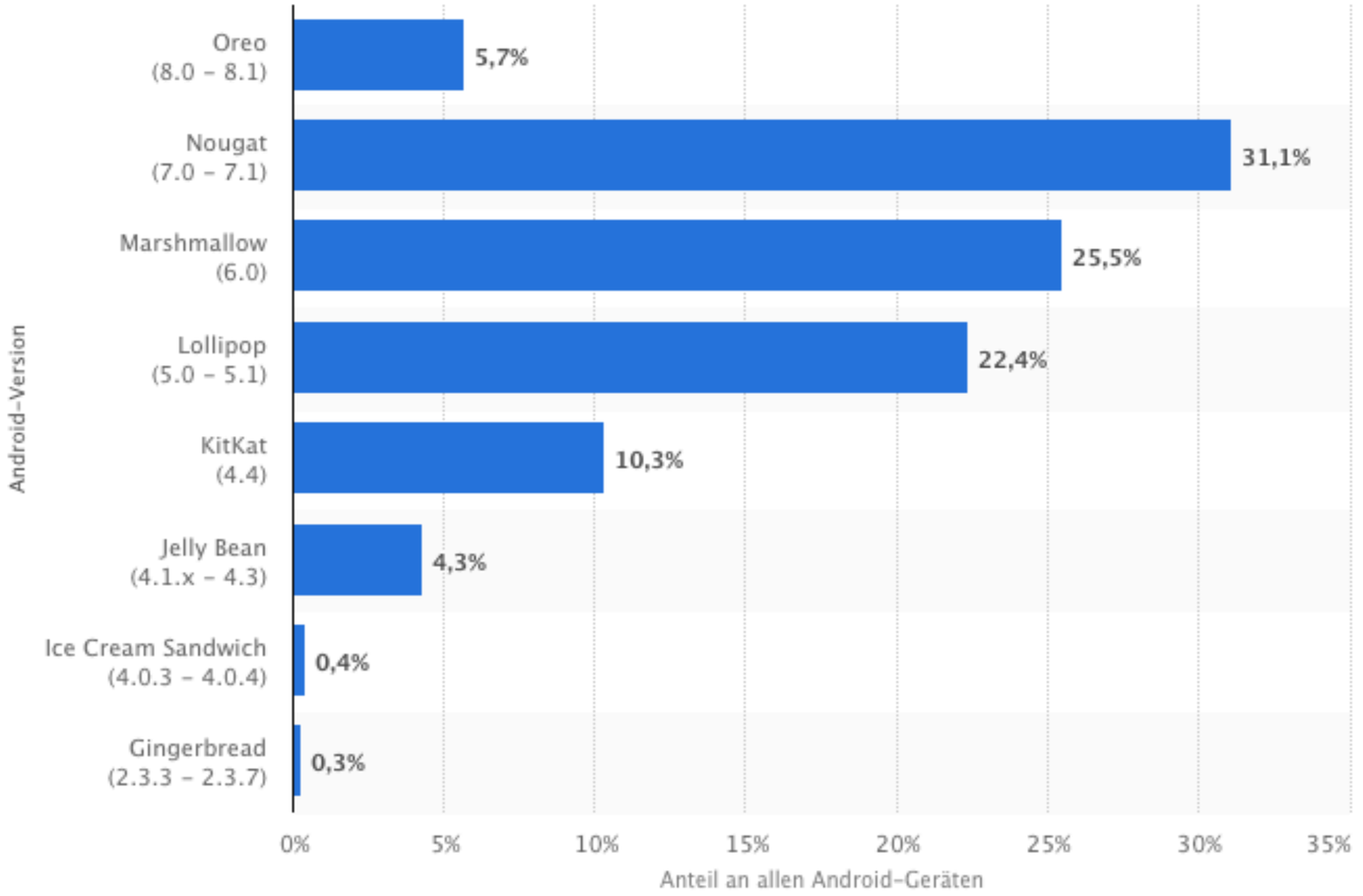


Fragmentation

- Fragmentation is the biggest issue for Android
- There are:
  - dozens of manufacturers
  - hundreds of providers
  - thousands of different devices
- …and nearly every device with a different version of the Android OS installed on it



Android Versions as of May 2018



Android OS - Security Features

Similarities to Linux

- The Android OS is using many features that are known from Linux and UNIX like systems:
    - User and group based permissions on file system level
    - SELinux: system rules that protect important calls and areas from unauthorized use or access

- Missing or unpropper SELinux rules are often one of the „entry doors“ to escalate privileges on the device.

Sandboxing

- Each app on an Android-based device is running in a separated and protected environment - called sandbox
- Access to a sandbox of another app is only possible by making use of the permission-system
- Those permissions have to be requested by the app developer and accepted by the user of the device
- Unfortunately, developers handle those permissions often in a sloppy way

Full Disk Encryption (FDE)

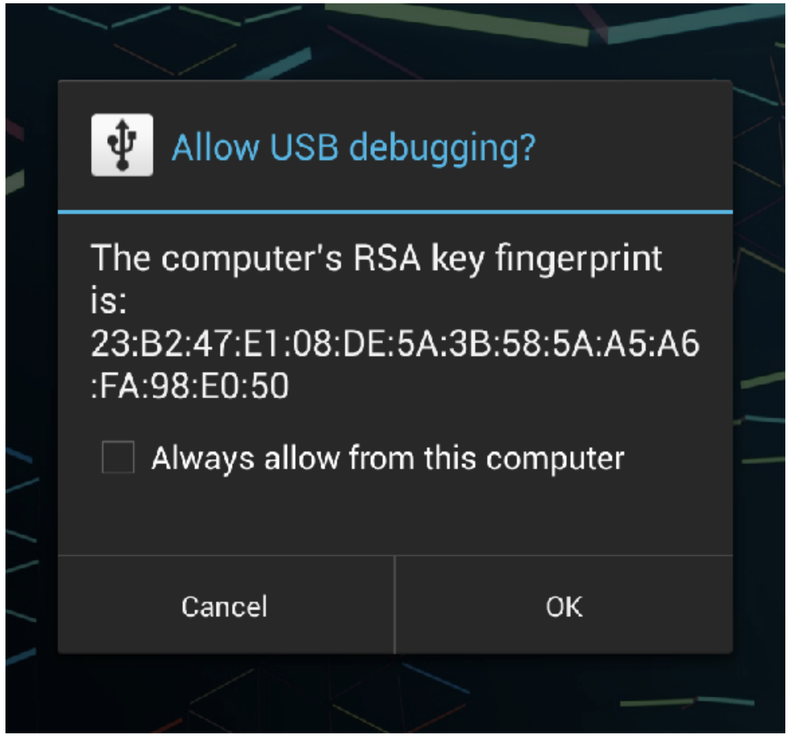
- Android started with Android 3.0 to enable the possibility to encrypt the device
- Android was using **dmcrypt** with **PBKDF2** until version 4.4
- With Android 4.4 they removed PBKDF2 and introduced **scrypt**, this should make Brute-Force-based attacks much harder
- Android 5.0 introduced the feature to store the keys on a secure hardware environment
- There are 3 ways to break the encryption:
    - Using hashcat to brute-force the password
    - Using the crypto footers of the partition to calculate the encryption key
    - Cold-Boot attacks to extract the key out of the RAM (only possible if the device is still up and running and had been unlocked since the last reboot)

Key Derivation Function (KDF)

- A KDF is more or less a function to derive a key with a specific length out of a user-based password
- PBKDF2 (Password-Based Key Derivation Function 2):
  - A cryptographic function combined with a fixed salt is used to generate the key.
  - The cryptographic function is SHA256
  - This function is executed several times (>100) to generate the final key.
- scrypt:
  - The function is very similar to PBKDF2 but is making heavy use of hardware-based operations which makes it very expensive to calculate those keys.

Secure USB Debugging

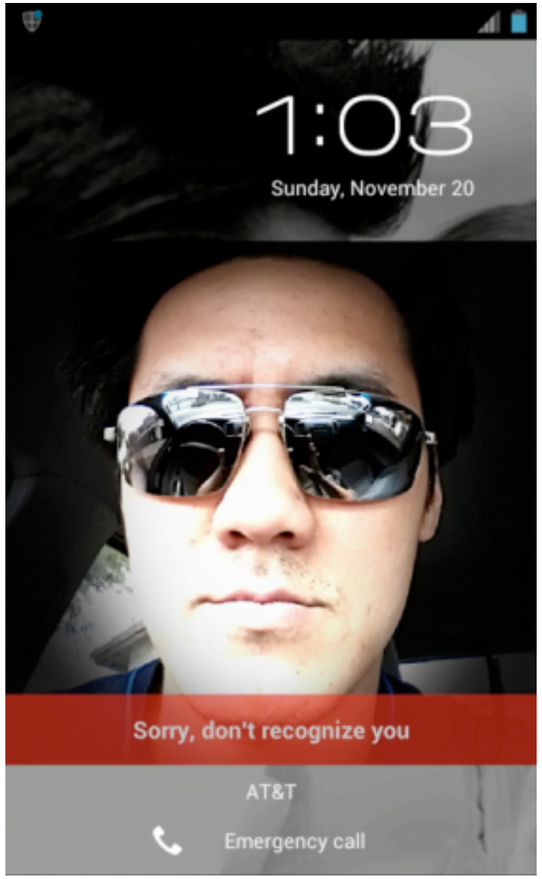
- With Android 4.2.2 Google introduces secure USB debugging
  - each host that tries to establish a connection to a device needs to be accepted by the user of the unlocked device
  - very similar to what you may know from SSH connections on Linux/Unix-based systems
  - no more silently connecting to an insecure device
  - protection of the debug interface



Screenlock

- Face Recognition
- Pattern Recognition
- PIN/Passwort
- SmartLock
- Fingerprint

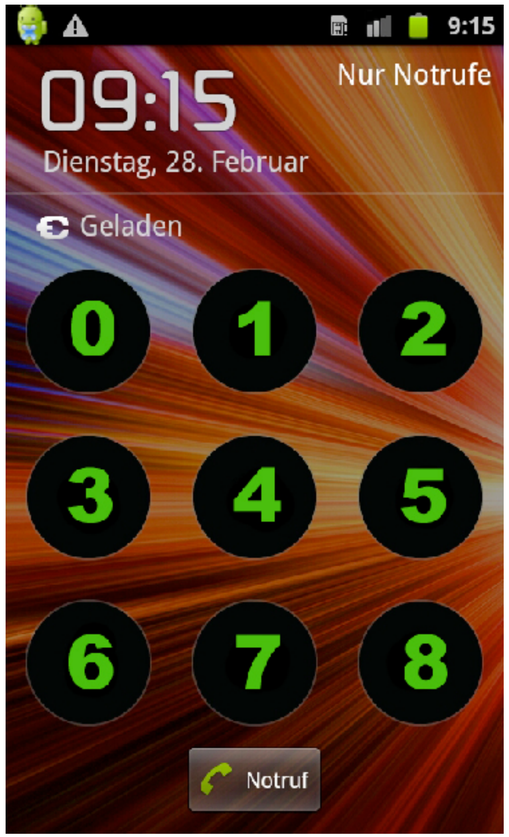
Face Recognition (1st Generation)



Face Recognition (1st Generation)

- no suitable protection of the device
- as an attacker/investigator you only need a picture of the owner
- just put the picture in front of the camera and play a bit with the angle and light to unlock the device

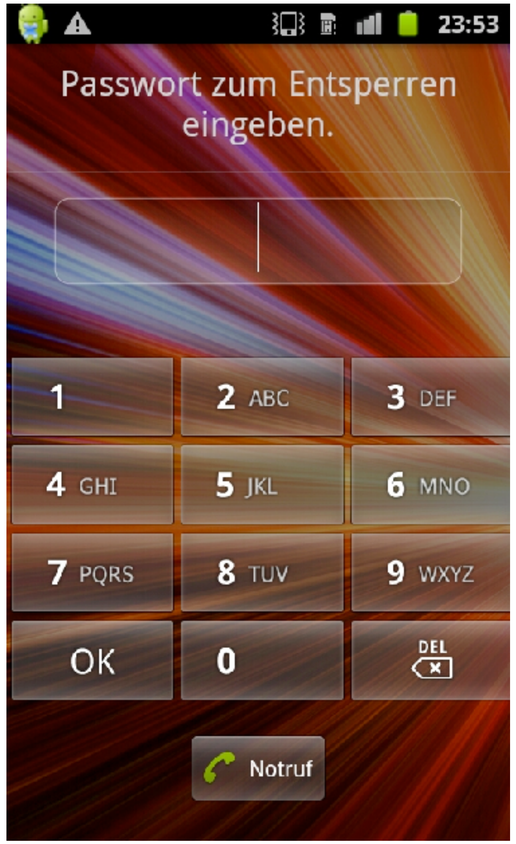
Pattern Recognition aka. Gesture-Lock

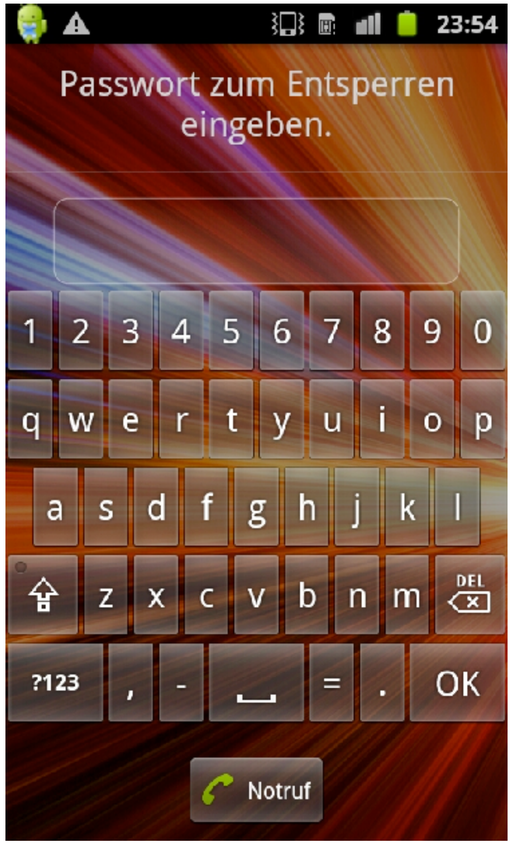


Pattern Recognition aka. Gesture-Lock

- Pattern is stored as numbers that are based on the location of the dots
- Pattern with a high complexity also ensures complex and long numbers
- Numbers are hashed with SHA1 but without a Salt
- Resulting hash is stored in /data/system/gesture.key
- Generation of the clear-text number is possible by using a pre-calculated rainbow table

PIN / Passwort





PIN / Passwort

- More or less the best protection of the device
- Long and complex PINs or Passwords generate the best protection
- PINs and Passworts are dealt identically by the Android OS
- PIN/Passwort is hashed by using SHA1, but this time a Salt is also being used
- Hash is stored at /data/system/password.key
- Starting with Android 4.2 the Salt can be found at /data/system/locksettings.db
- PIN/Passwort cracking can be performed with the help of hashcat
  (some devices can also make use of GPUs to fasten up the cracking)

Android Gatekeeper

- Since Google noticed, that it is still very easy to crack the screenlock which is based on Passwords/PINs or Patterns they introduced a new security feature in Android 6.0 called Gatekeeper.

- Gatekeeper uses the trusted element of the hardware to store encryption keys in a secure position where they cannot be extracted off the device.

- It is still possible to brute-force and crack the screenlock, but you now have to do it directly on the device.

- This process is much slower and less documented.

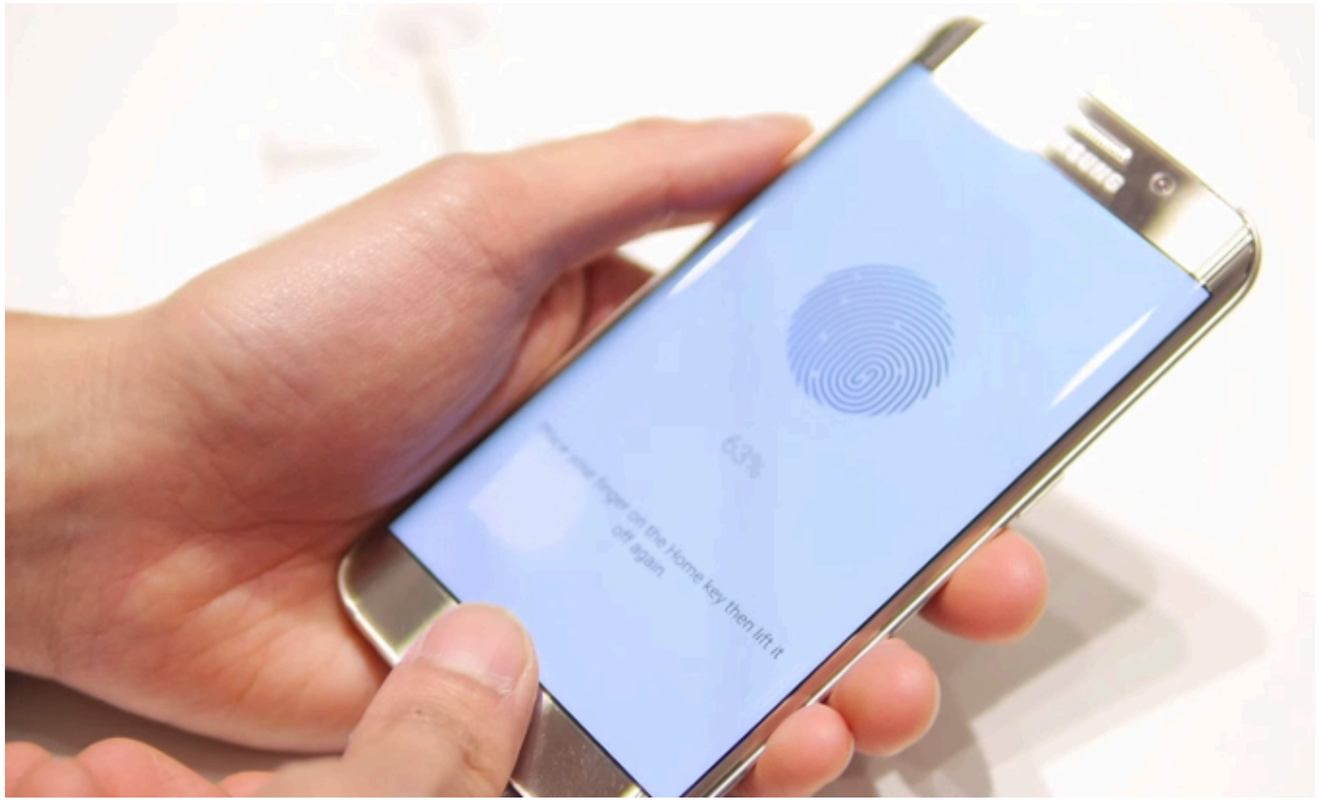
SmartLock

SmartLock

- With Android 5.0 the OS started to allow the unlock by using „smart“-devices like a watch, fitness tracker, ring or any other bluetooth-speaking device

- Try to find this device and bring it near the Android device you are trying to unlock

Fingerprint

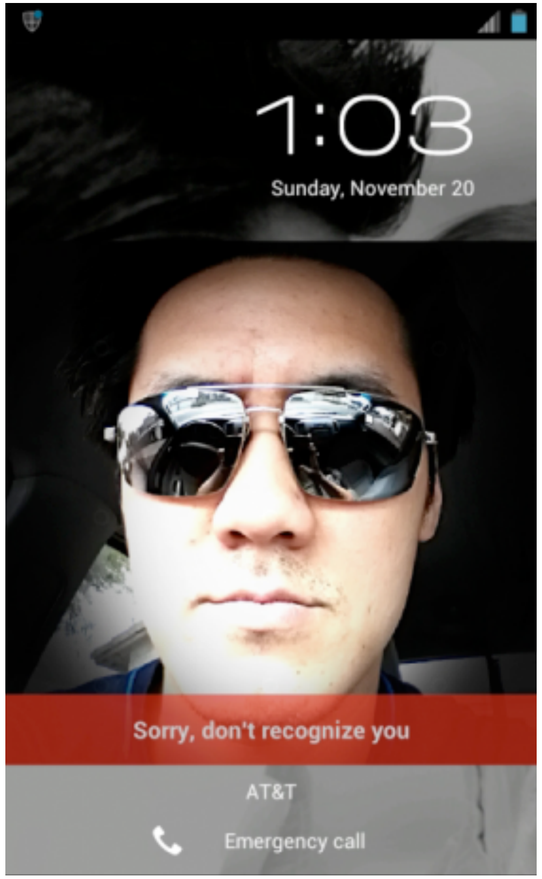


Fingerprint

- After iOS, Android also introduced the feature to unlock the device by using your fingerprint
- Most of the Android-based devices have unlimited tries to identify the fingerprint
- The needed quality of the copied fingerprint has to be only of medium conformity (60% is often enough)
- Many Android-based devices even allow to unlock them by using the fingerprint after the device has been rebooted

Fingerprint

Face Recognition (2nd Generation)

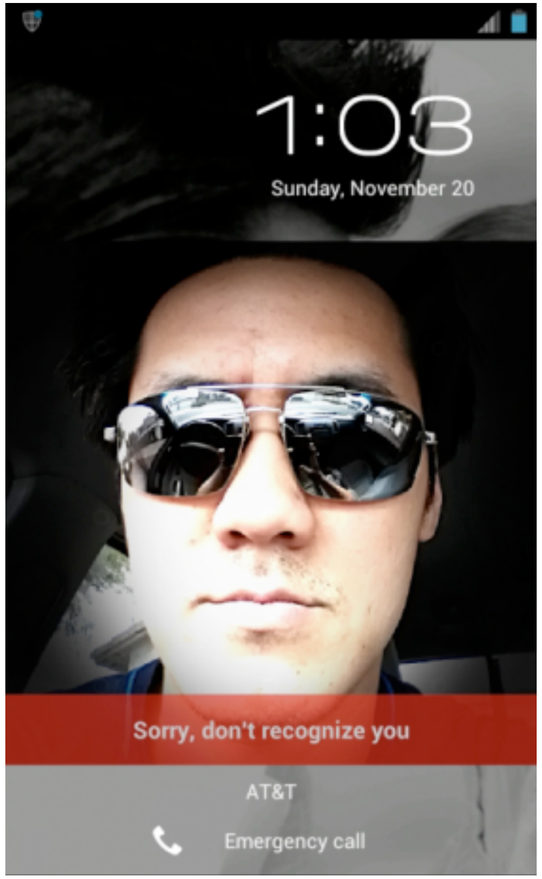


Face Recognition (2nd Generation)

- protection of the device is getting stronger
- the device is now using a picture of the iris only
- as an attacker/investigator you now need a picture of the iris of the owner (may be a bit harder to get because you need to be closer or use a better camera)
- just put a contact lens on the picture in front of the camera and play a bit with the angle to unlock the device

Face Recognition (2nd Generation)

Face Recognition (3rd Generation)



Face Recognition (3rd Generation)

- protection of the device is getting stronger
- the device is now using a picture based on the IR color spectrum
- as an attacker/investigator you now need a picture of the owner that shows only the IR color spectrum of the face
- just put the picture in front of the camera and play a bit with the angle and light to unlock the device

Android OS - File System

File System

- Until Android 2.3.3 **YAFFS2** was used as file system on Android-based devices
  - Documentation and real Implementation differ a lot
  - nearly no tool support
  - hard to analyze on a file system level
- Starting with Android 2.3.3 **Ext3** and **Ext4** are used
  - very good documentation
  - great tool support (e.g., FTK, Autopsy and Sleuthkit)
  - Garbage-Collection and wearleveling are active and very aggressiv
    (hard to find something that has been deleted weeks before the analysis)

Android Device Analysis - Overview

All I want to be is ROOT

- User- and group-based permission model on file system layer
- SELinux rules to protect sensitive areas of the file system

=> gain root privileges

All I want to be is ROOT

- Samsung, HTC and other manufacturers sometimes make use of open bootloaders that allow to install custom bootloaders or to boot the device from memory.

- For a large amount of devices you will be able to find exploits the make use of known vulnerabilities to escalate privileges, but be careful especially if the device is making use of KNOX or Android4Work.

- Regardless if you are using an exploit or a custom bootloader, always proof the origin and the trustworthiness.

Where to find Evidence?

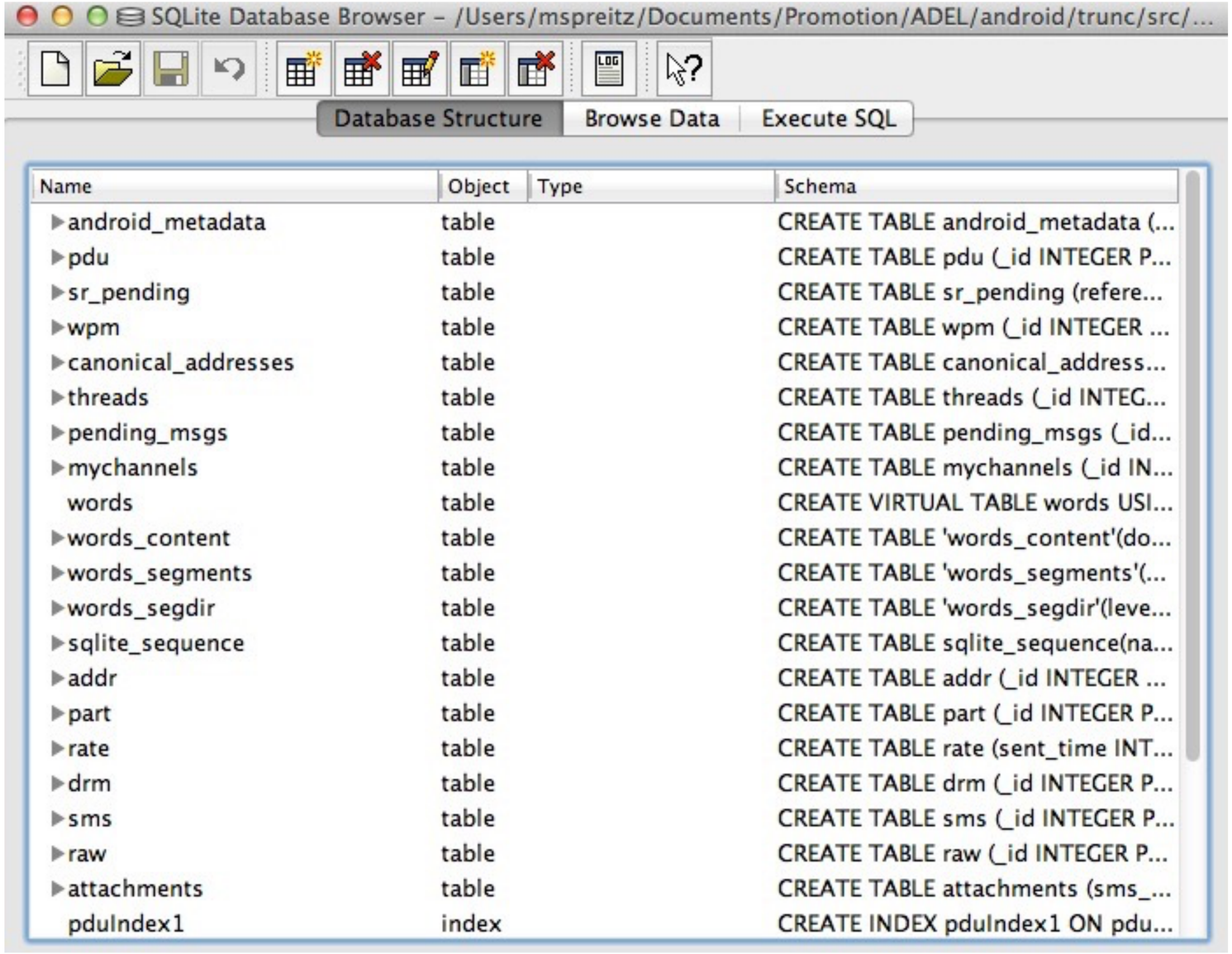
- Each app is making use of at least one local database
- Location of the installed apps can be found at:
    - /data/data/....
    - /sdcard/data/data/....
- Documents, Pictures and Videos can often be found at:
    - /sdcard/....
- The sandbox concept of apps is also true for the memory (RAM)

How does the Evidence look like?

- Pictures are stored as JPEG files
    - look for meta data (EXIF)

- Storage of the apps is often done by using local databases (SQLite DB files)
    - for normal use: SQLite Database Browser
    - for professional use: Sanderson Forensic Toolkit for SQLite

SMS Messages



SMS Messages

	_id	thread_id	address	persor	date	protocol	read
1	1	1	3Alerts		946884213134	0	
2	2	1	3Alerts		946884245833	0	
3	3	1	3Alerts		1312572952989	0	
4	4	2	Rate Advice		1312572963616	0	
5	5	3	SWISSCOM		1312742754973	57	
6	6	4	Swisscom		1312742856733	57	
7	7	5	801		1312742978556	57	
8	8	5	801		1312800462737	57	
9	9	6	123		1312881260226	57	
10							
11							

	body	service_center	locke
1	Please keep your phone on while we connect you to Three. It'll take abo	+447782000801	
2	We've nearly finished. Now we just need you to turn your phone off and	+447782000801	
3	If you now need to, top-up £15 & buy All in One 15 Add-on to get 300	+447782000801	
4	From 3: In EU it's max of 36.6p/min to call,11.5p/min to receive call,1C	+447782000801	
5	Tarife (CHF) in Europa: Anrufe innerhalb Europa und in die Schweiz: 0.8!	+41794999000	
6	Damit Sie Ihre NATEL-Dienste optimal verwenden können, erhalten Sie	+41794999000	
7	Unser Geschenk: Sie erhalten ein NATEL easy Datenpaket 100 MB für die	+41794999000	
8	Aufgeladener Betrag: CHF 30.00 Neues Guthaben: CHF 38.71 Sie könne	+41794999000	
9	Sie haben noch 18 MB vom NATEL easy Datenpaket zur Verfügung. Sie k	+41794999000	
10	Sie haben noch 8 MB vom NATEL easy Datenpaket zur Verfügung. Sie kö	+41794999000	
11	Ihr NATEL easy Datenpaket DATA100MB ist abgelaufen oder aufgebrauc	+41794999000	

Pictures (JPEG)

- Have a look at the EXIF meta data within each file
- This meta data contains:
    - Location of the device (GPS) while the picture was taken
    - Device information
    - Timestamp
    - etc.

Pictures (JPEG)

Basic Image Information

Camera:	Apple iPhone 4
Lens:	3.9 mm Digital Zoom: 2.347432024×
Exposure:	Auto exposure, Program AE, 1/17 sec, f/2.8, ISO 80
Flash:	Off, Did not fire
Date:	**May 22, 2011** 2:50:02PM (6 months, 7 days, 7 hours, 7 mi ahead of GMT)
Location:	Map via encoded GPS coordinat (also see the Google Maps pane Timezone guess from earthtools
File:	**1,936 × 2,592** JPEG (**5.0** m 1,345,625 bytes (1.3 megabytes)

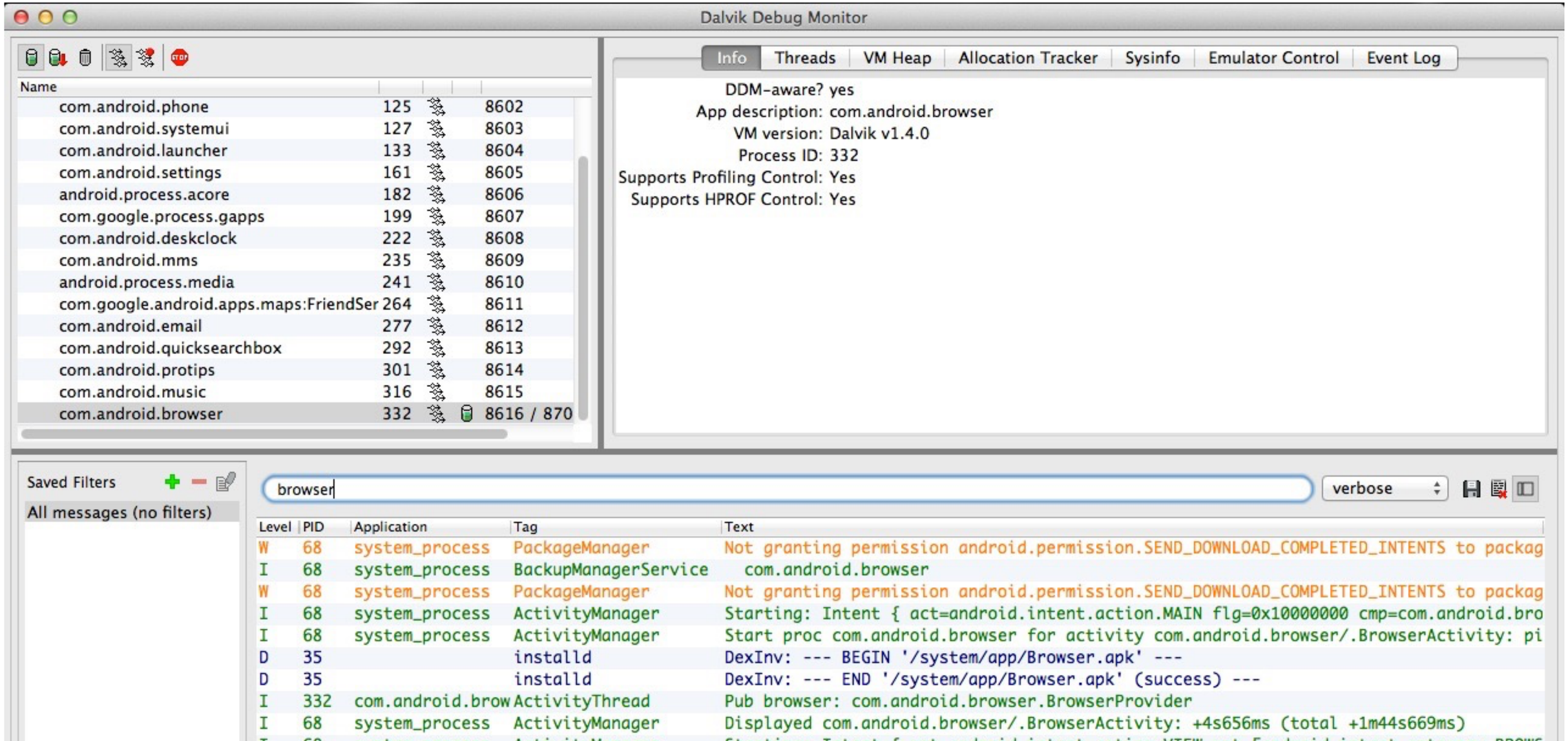
GPS-encoded location: 49° 30' 04"N, 10° 58' 07"E
Map center: 49° 30' 04"N, 10° 58' 07"E
Display area: 970 m × 349 m
Distance between: 0 m

Click on map to measure distance from GPS-encoded location

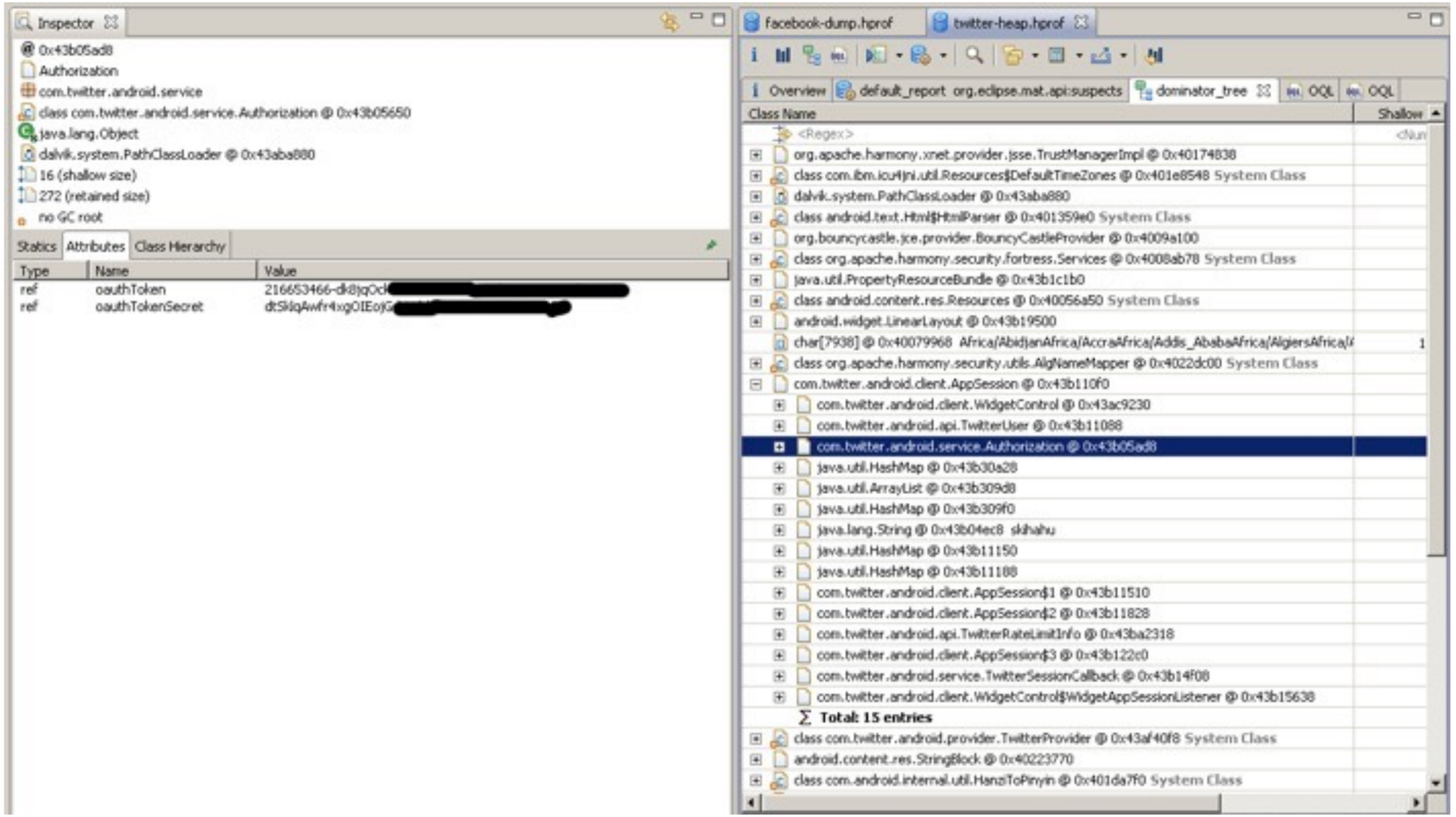
Cached Data in Memory (RAM)

- until Android 2.3
    - kill -10 <pid>
- starting with Android 2.3
    - Dalvik Debug Monitor Server (DDMS)
      (only possible for apps that have debugging enabled)

Cached Data in Memory (RAM)



Cached Data in Memory (RAM)



Android Device Analysis - Preservation

adb Backup

- The adb backup feature is available since Android 4.0
- This feature allows to backup apps and their data to a remote computer or the Google cloud
- This feature is only available for us (the investigator) if:
  - adb itself is enabled
  - the screen is unlocked already
  - the app that we want to backup doesn’t have this feature disabled within the Android manifest

adb Backup

- The generated backup file is often encrypted
- Therefore you need to use an additional tool to gain access to the clear text of the backup files:
  - Android Backup Extractor by Nikolay Elenkov
  - https://github.com/nelenkov/android-backup-extractor

Autopsy

- Starting with Autopsy 4.0 it is possible to analyze Android file system images
- With this feature you can create „one-to-one“ dumps of the file system partitions with dd and import them into Autopsy

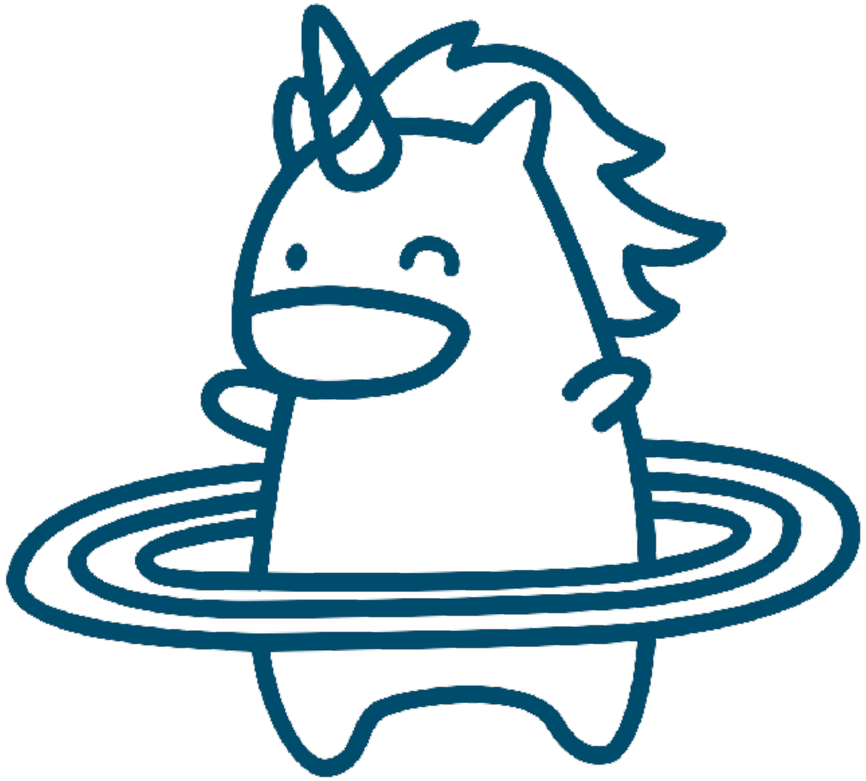
- How to:
  - Boot the Android device by using a alternate Recovery Image
  - Use the mount command to get a list of partitions
    (the important ones normally are data and cache)
  - Create a „one-to-one“ copy of those partitions by using dd

adb

- Manual investigation and preservation using the command line interface adb

- Important files:
  - Browser History and Bookmarks: /data/data/com.android.browser/browser.db
  - Contacts and Call-Lists: /data/data/com.android.providers.contacts/contact2.db
  - Media: /data/data/com.android.providers.media/external.db
  - SMS/MMS: /data/data/com.android.providers.telephony/mmssms.db
  - Installed Apps:
    - /data/system/packages.list
    - /data/system/packages.xml
    - /data/system/dmappmgr.db (nur Samsung)

Let's start with the exercises now!



Exercise 1

- Breaking the pattern-based screenlock

Exercise 2

- Breaking the PIN-based screenlock

Exercise 3

- Having a look at pictures and their Exif data

Exercise 4

- Having a look at a logical copy of the Google-Maps application

Exercise 5

- Using Autopsy to analyze a physical dump of a device

Exercise 6

- Breaking the Full Disk Encryption (FDE) of Android 4

Recap

- Different security features of Android on file system level
- Different ways to protect the device against unauthorized users and how to circumvent those
- Why it is important to get root
- Where to find the interesting data (evidence)?
- How does the evidence look like?
- 3 ways to perform preservation and start to analyze the evidence

See you next week!