FH AACHEN
UNIVERSITY OF APPLIED SCIENCES

Extrahierung und Analyse von Android RAM-Abbildern

Simon Broenner

Lehrgebiet Datennetze, IT-Sicherheit und IT-Forensik

IT Forensik
FH Aachen

Aufgabenstellung

Praxisprojekt & Ba.-Arbeit:

- Untersuchung der Möglichkeiten zu:
  - Extrahierung
  - Analyse
- Reproduzierbare Vorgehensweise zur Extrahierung erarbeiten
- Einbau in VOLIX II Volatility-Frontend (Analyse-Teil)

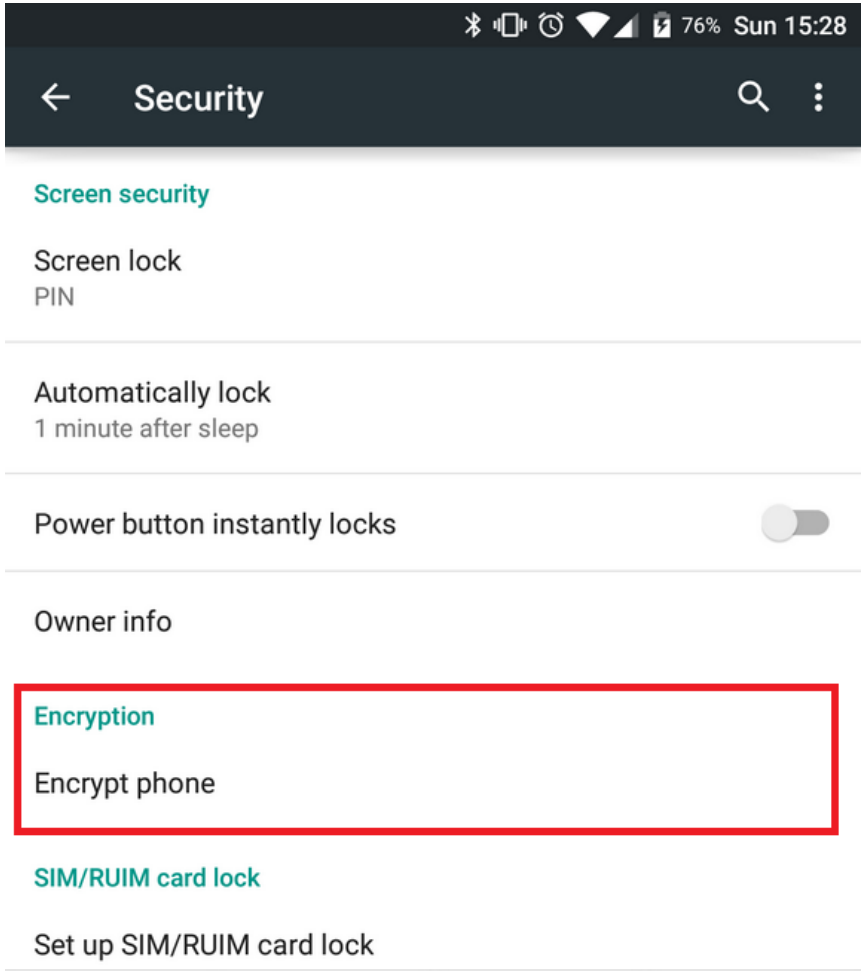
Zukünftig:

- Extrahierung vereinfachen, automatisieren

Was bringt Android RAM-Analyse?

Herausforderungen bei Android:

- Vollverschlüsselung (Full Disk Encryption) ab 4.x eingebaut
- Locker/Vault Apps
  - Einfach/schnell programmierbar
  - Leicht erhältlich
  - Android segmentiert App-Daten
  - Verschlüsselung schnell implementierbar
- Malware-Analyse/-Bekämpfung
  - Wenig Tools
  - Android Application Sandbox



Vollverschlüsselung aktivieren bei Android 5.x

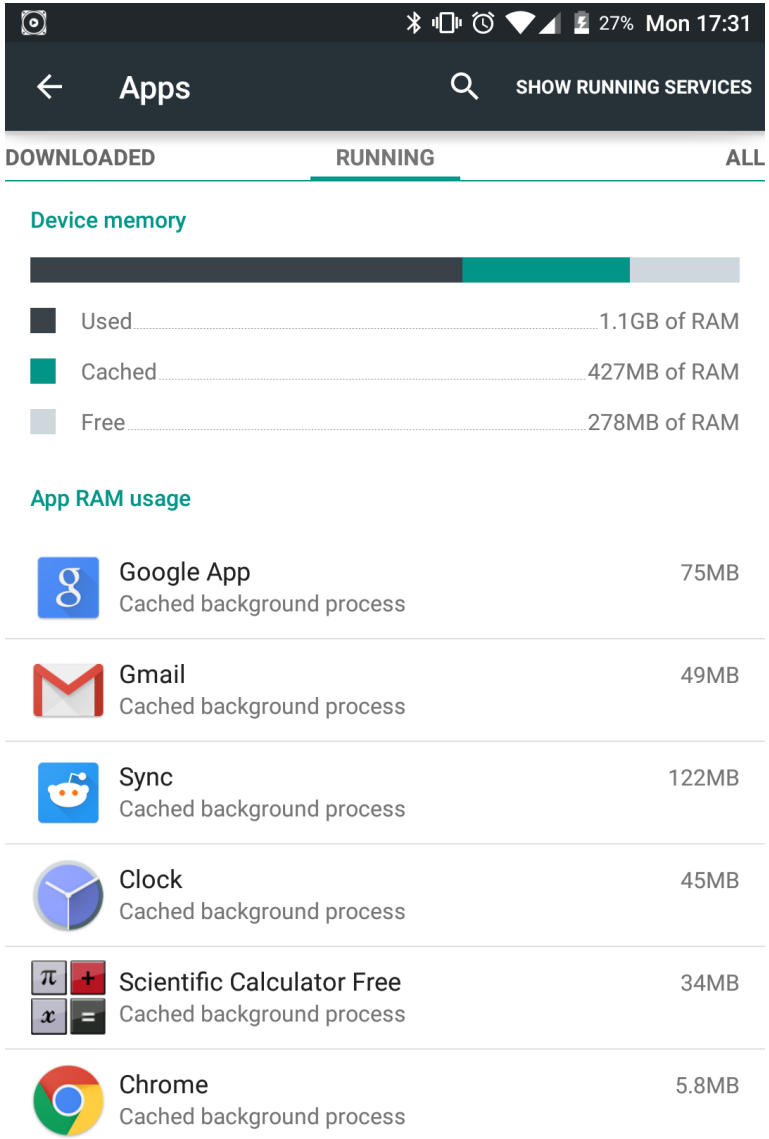
Locker-Apps aus dem Play Store

Was bringt Android RAM-Analyse?

Vorteile der RAM-Analyse:

- RAM unverschlüsselt

  => Auch bei FDE verwertbar
- Bekannte Struktur
  - aus Kernel-Datenstrukturen (vtypes)
  - aus Symbole (aus System.map)
- Enthält Infos über
  - FDE
  - Locker-Apps
  - Laufende Malware



Herausforderungen bei Android

Nachteile der RAM-Analyse:

- Extrahierung aufwändig
- Extrahierung nicht immer möglich
- Analyse aufwändig

Funktionierende Methoden:

- LiME (Linux Memory Extractor) Kernelmodul
- Cold-Boot-Attack, z.B. FROST

FROST: Forensic Recovery of Scrambled Telephones

LiME Kernelmodul

Einige Nachteile:

- Benötigt Root-Rechte
- Benötigt Konsolen-Zugriff (Terminal-Emulator, ADB o.Ä.)
- Kernelmodul muss angepasst werden
  - Kernel-Quellcode wird benötigt
  - Cross-Compile des Kernelmoduls mit Toolchain aus Android NDK

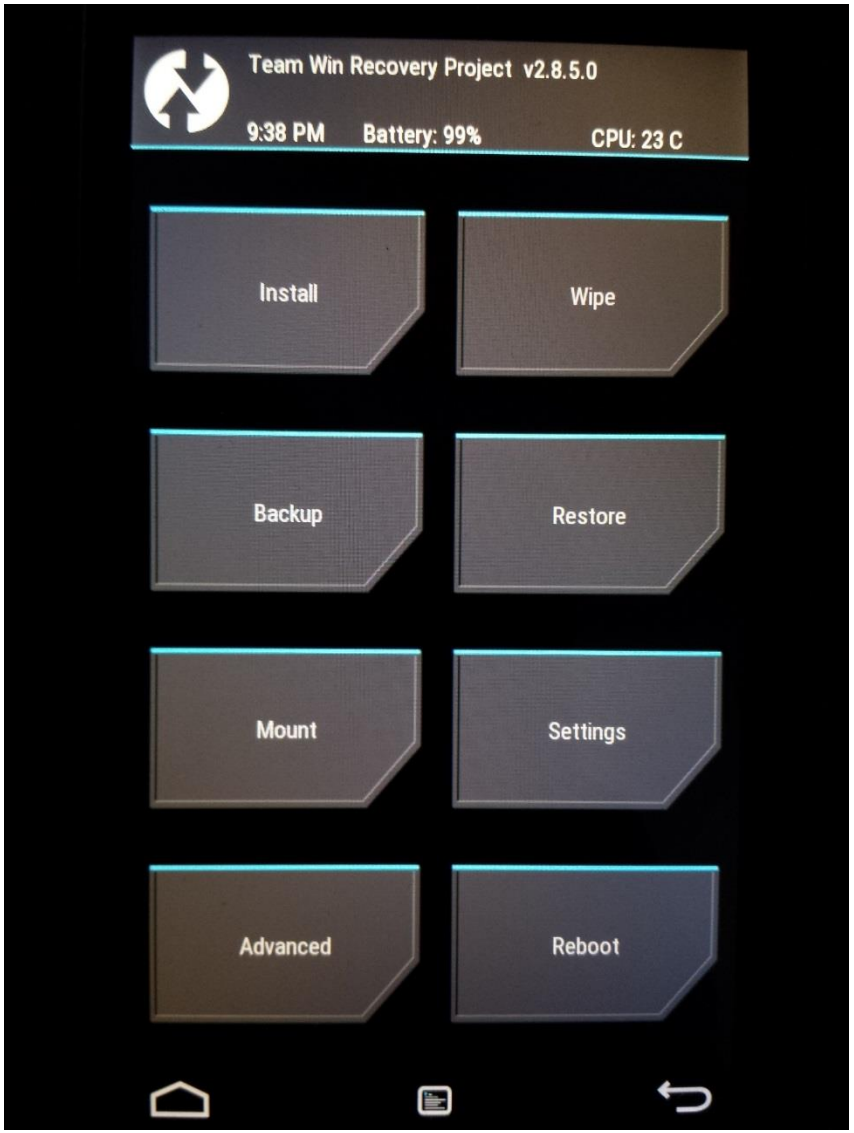
=> In der Praxis nur bei manchen Geräten geeignet:

- Geräte mit offenem ADB Zugang oder ohne Sperrcode
- Geräte mit Vollzugriff (z.B. bei Malware-Analyse)

Cold-Boot Attack per Recovery

Einige Nachteile:

- Passendes Recovery Image nötig
- Einspielen/starten ändert Arbeitsspeicherinhalt (Cold-Boot Problematik)
- Auslesen immer noch per Kernelmodul



Herausforderungen bei Android

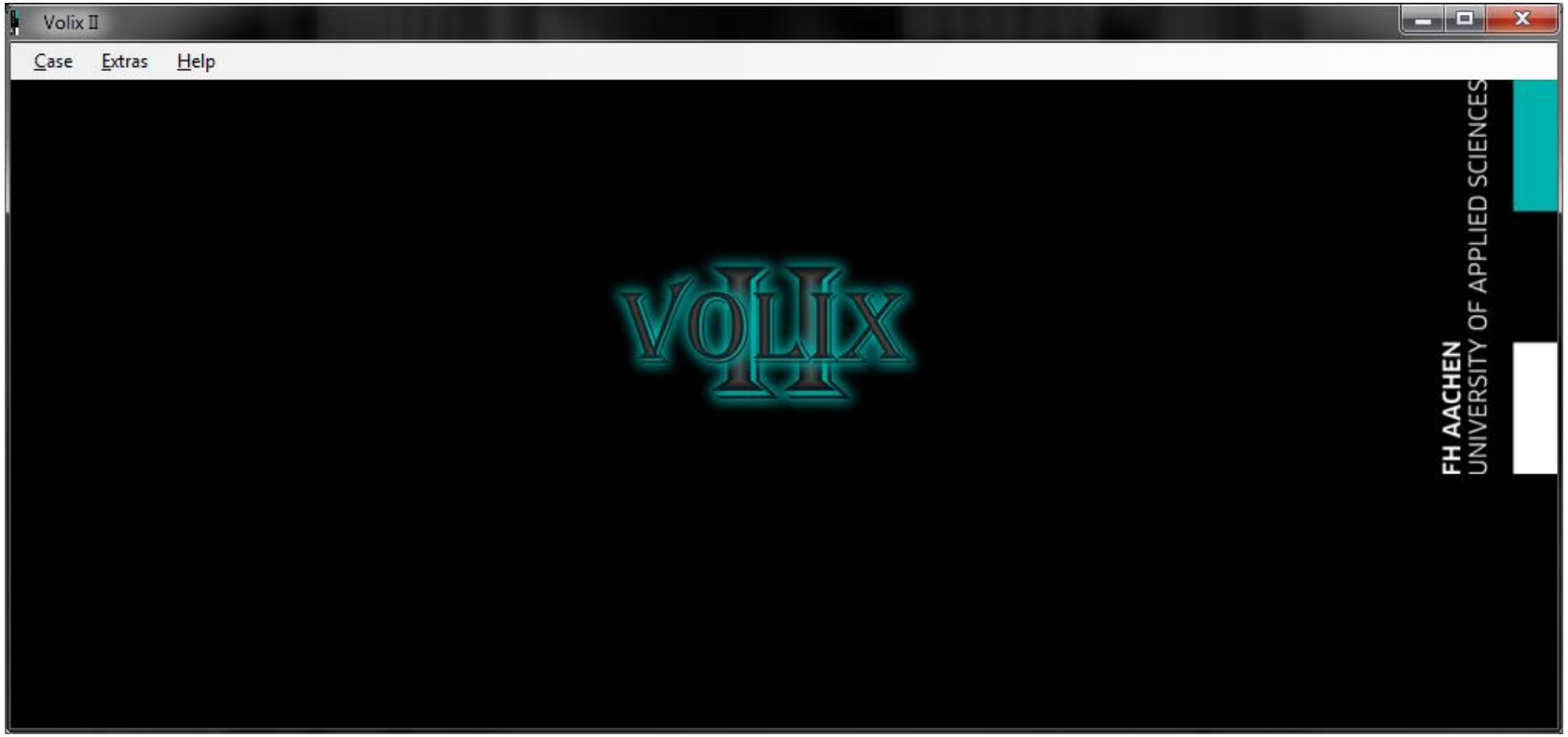
- Analyse mit Volatility Framework aufwändig
  - Fehlende Volatility-Profile für Android

    => selbst generieren
    - Benötigt Kernel-Quellcode
    - Benötigt Android NDK Toolchains
  - Bedienung per Kommandozeile

Vereinfachung des Vorgangs

Praxisprojekt & Bachelorarbeit:

- Ansatz zur Extrahierung
- Ansatz zur Analyse mit Volatility
- Analyse mit Volatility durch Integration in VOLIX II vereinfachen



Vereinfachung des Vorgangs

Zukunftsmusik: Extraction Toolkit

- Vorgefertigte LiME Kernelmodule und Volatility Profile für beliebte Geräte
  - Volatility plant Linux Profile-Sammlung & Austausch
- Andere Geräte per Script
  - Kompilierung/Erstellung von Kernelmodul und Profil
  - Benutzer stellt lediglich benötigten Kernel-Quellcode bereit
    - von Hersteller erhältlich (GPL)

Vereinfachte Vorgehensweise

Evtl. Vorgehensweise für Forensiker:

1. Gerät anschließen (USB)
2. Extraction Toolkit starten
3. Passendes Kernelmodul und Profil wählen oder generieren lassen
4. *insmod* Befehl ausführen lassen:
   root@android:/ # insmod /sdcard/lime.ko "path=/sdcard/lime.dump format=lime"
5. lime.dump Datei kopieren lassen:
   adb pull /sdcard/lime.dump C:\ITForensik\Fall_0493482\lime.dump
6. lime.dump in VOLIX II öffnen und analysieren

Aktueller Stand

- Bisher durchgeführt:
  - Kompilierung LiME Kernelmodul
  - Speicherabbilder aus laufendem Android 5.1.1 System extrahiert
  - Passendes Volatility-Profil erstellt
  - Auslesen von Daten per Kommandozeile, z.B. laufende Prozesse via Volatility-Plugin linux_pslist

Aktueller Stand

Beispiel-Ausgabe: linux_pslist bei einem Android 5.1.1 Speicherabbild

Offset	Name	Pid	Uid	Gid	DTB	Start Time
0xd401cc00	init	1	0	0	0x143c4000	2015-05-03 13:10:57
0xd401c800	kthreadd	2	0	0	----------	2015-05-03 13:10:57
0xd401c400	ksoftirqd/0	3	0	0	----------	2015-05-03 13:10:57
0xd401c000	kworker/0:0	4	0	0	----------	2015-05-03 13:10:57
0xd4024800	khelper	6	0	0	----------	2015-05-03 13:10:57
0xd4024400	sync_supers	7	0	0	----------	2015-05-03 13:10:57
0xd4024000	bdi-default	8	0	0	----------	2015-05-03 13:10:57
0xd406dc00	kblockd	9	0	0	----------	2015-05-03 13:10:57
0xd406d800	rpciod	10	0	0	----------	2015-05-03 13:10:57
0xd406d000	kswapd0	12	0	0	----------	2015-05-03 13:10:57
0xd428bc00	fsnotify_mark	13	0	0	----------	2015-05-03 13:10:57
0xd428b800	crypto	14	0	0	----------	2015-05-03 13:10:57
0xd42ac800	kworker/u:1	25	0	0	----------	2015-05-03 13:10:57
...						
0xcca5c800	externalstorage	953	10006	10006	0x078e4000	2015-05-03 13:12:52
0xc7839400	droid.deskclock	969	10021	10021	0x0cb1c000	2015-05-03 13:12:52
0xc61cf800	m.android.music	987	10033	10033	0x061dc000	2015-05-03 13:12:56
0xc61ffc00	ndroid.calendar	1002	10017	10017	0x07918000	2015-05-03 13:12:56
0xc61cfc00	viders.calendar	1022	10001	10001	0x07b98000	2015-05-03 13:12:57
0xc6117800	ndroid.keychain	1042	1000	1000	0x06128000	2015-05-03 13:12:58
0xc6166400	.android.dialer	1066	10004	10004	0x07ab8000	2015-05-03 13:14:58
0xc613d000	gedprovisioning	1086	10008	10008	0x0cbbc000	2015-05-03 13:14:58
0xc6372c00	com.android.mms	1105	10009	10009	0x06390000	2015-05-03 13:15:00
0xc6372800	ndroid.settings	1133	1000	1000	0x063cc000	2015-05-03 13:15:01
0xccbf0000	m.android.email	1156	10025	10025	0x12eb0000	2015-05-03 13:15:17
0xc2b82000	ndroid.exchange	1172	10027	10027	0x00e04000	2015-05-03 13:15:17
0xd3bb3000	flush-31:1	1212	0	0	----------	2015-05-03 13:31:58
0xd38cfc00	kworker/0:1	1213	0	0	----------	2015-05-03 13:34:11
0xd38cf000	sh	1222	0	0	0x00e88000	2015-05-03 13:34:47
0xc2b82800	flush-179:0	1227	0	0	----------	2015-05-03 13:34:50
0xd4024c00	insmod	1228	0	0	0x00fbc000	2015-05-03 13:35:18

Feierabend

vCard als QR-Code:

Simon Broenner

simonbroenner+itf@gmail.com

+49 176 3916 0894