Modmobmap

The modest mobile networks mapping tool

By Sébastien Dudek

BeeRumP

May 31st 2018

Introduction

- Modmobmap (sounds like “Bimbimpbap”): Modest Mobile networks Mapping tool
- Used to map 2G/3G and 4G networks (maybe more) in real live
- Uses a set of tricks (including the cheapest) to map cells

Where can I use this tool?

Cell towers discovery

- have a list and description of surrounding towers
- spot rogue base stations (mature list required!)

Restricted/smart/magic jamming

Where can I use this tool?

Cell towers discovery

Restricted/smart/magic jamming

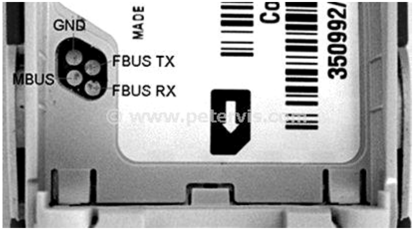
- replace the heavy & noisy & cumbersome jammer (or portable ones with weak signals)
- avoid commercial jamming device reworking (bands disabling)

Remember: monitoring with holy relics

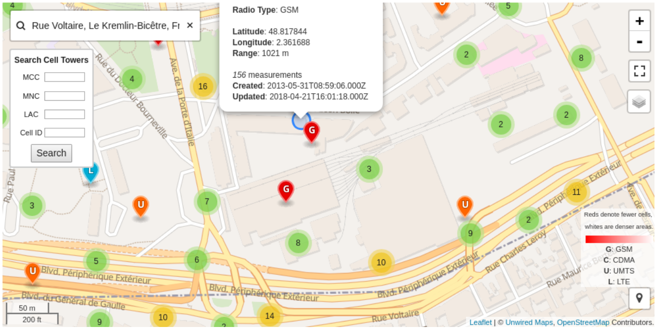
Old Nokia phone have a net monitor mode that could be enabled via FBus or MBUS access.

Tools

- Gnokii, Gammu and others: activate monitor mode, interact with the phone, and capture trace logs.
- DCT3-GSMTAP: evolution of Gammu, capture of GSM Um and SIM-ME via GSMTAP pseudo-header format.



Existing tool


Rue Voltaire, Le Kremlin-Bicêtre, Fr ×

Search Cell Towers

MCC

MNC

LAC

Cell ID

Search

Radio Type: GSM

Latitude: 48.817844
Longitude: 2.361688
Range: 1021 m

156 measurements
Created: 2013-05-31T08:59:06.000Z
Updated: 2018-04-21T16:01:18.000Z

Reds denote fewer cells, whites are denser areas.

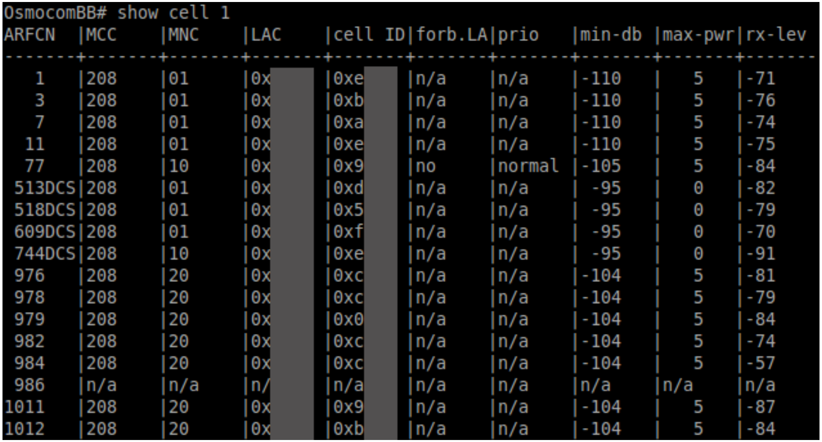
G: GSM
C: CDMA
U: UMTS
L: LTE

Blvd. Périphérique Extérieur
Blvd. du Général de Gaulle
Rue Voltaire
Rue Charles Leroy
Rue du Docteur Bourneville
Ave. de la Porte d'Italie
50 m
200 ft
Leaflet | © Unwired Maps, OpenStreetMap Contributors.

OpenCellID example

But very few information... could be used as a database for spotting rogue base stations. But useless for jamming attacks

Thing we wanna do for 3G, 4G and more



```
OsmocomBB# show cell 1
ARFCN  |MCC    |MNC    |LAC    |cell ID|forb.LA|prio   |min-db |max-pwr|rx-lev
-------+-------+-------+-------+-------+-------+-------+-------+-------+-------
   1   |208    |01     |0x[redacted]|0xe[redacted]|n/a    |n/a    |-110   |   5   |-71
   3   |208    |01     |0x[redacted]|0xb[redacted]|n/a    |n/a    |-110   |   5   |-76
   7   |208    |01     |0x[redacted]|0xa[redacted]|n/a    |n/a    |-110   |   5   |-74
  11   |208    |01     |0x[redacted]|0xe[redacted]|n/a    |n/a    |-110   |   5   |-75
  77   |208    |10     |0x[redacted]|0x9[redacted]|no     |normal |-105   |   5   |-84
 513DCS|208    |01     |0x[redacted]|0xd[redacted]|n/a    |n/a    | -95   |   0   |-82
 518DCS|208    |01     |0x[redacted]|0x5[redacted]|n/a    |n/a    | -95   |   0   |-79
 609DCS|208    |01     |0x[redacted]|0xf[redacted]|n/a    |n/a    | -95   |   0   |-70
 744DCS|208    |10     |0x[redacted]|0xe[redacted]|n/a    |n/a    | -95   |   0   |-91
 976   |208    |20     |0x[redacted]|0xc[redacted]|n/a    |n/a    |-104   |   5   |-81
 978   |208    |20     |0x[redacted]|0xc[redacted]|n/a    |n/a    |-104   |   5   |-79
 979   |208    |20     |0x[redacted]|0x0[redacted]|n/a    |n/a    |-104   |   5   |-84
 982   |208    |20     |0x[redacted]|0xc[redacted]|n/a    |n/a    |-104   |   5   |-74
 984   |208    |20     |0x[redacted]|0xc[redacted]|n/a    |n/a    |-104   |   5   |-57
 986   |n/a    |n/a    |n/[redacted]|n/a[redacted]|n/a    |n/a    |n/a    |n/a    |n/a
1011   |208    |20     |0x[redacted]|0x9[redacted]|n/a    |n/a    |-104   |   5   |-87
1012   |208    |20     |0x[redacted]|0xb[redacted]|n/a    |n/a    |-104   |   5   |-84
```

OsmocomBB cell monitor

Public tools

Recorded mobile towers

- OpenCellid: Open Database of Cell Towers
- Gsmmap.org
- and so on.

Live scanning tools

Public tools

Recorded mobile towers

- OpenCellid: Open Database of Cell Towers
- Gsmmap.org
- and so on.

Problem!

But these solutions don't map in live and do not give precise information about cell towers.

Live scanning tools

Public tools

Recorded mobile towers

Live scanning tools

- for 2G cells:
  - Gammu/Wammu, DCT3-GSMTAP, and others
  - OsmocomBB via *cell_log* application
- for 3G, 4G and more:
  - only tricks: use of exposed DIAG interface $\rightarrow$decoding $\rightarrow$GSMTAP pseudo-header format
  - SnoopSnitch: not reflexible, but could be reworked for our purposes ;)

Methods to capture cells information

Possible methods are:

- Software-Defined Radio
- Exposed diagnostic interfaces
- Use of Android RIL

Software-Defined Radio

Existing tools:

- Airprobe or GR-GSM
- OpenLTE: *LTE_fdd_dl_scan*
- srsLTE with srsUE

Software-Defined Radio

Existing tools:

- Airprobe or GR-GSM
- OpenLTE: *LTE_fdd_dl_scan*
- srsLTE with srsUE

No 3G

No 3G tools to capture cell information.

Exposed diagnostic interface

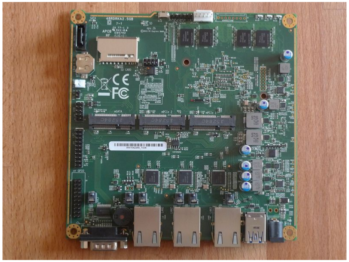
- Diagnostic interface enabled:
  - On old phones and 3G sticks like the *Icon 255*[1] that expose it by default
  - enabling DIAG ourselves: e.g for some LG devices via */sys/devices/platform/lg_diag_cmd/diag_enable*
  - Chips used for development
  - Interfaces kept enabled in production by error (e.g via custome bootmodes →CVE-2016-8467)
- Existing tools:
  - *xgoldmon* for X-Gold Infineon Basebands
  - *diag-parser* for exposed Qualcomm DIAG interfaces

[1] https://events.ccc.de/congress/2011/Fahrplan/attachments/2022_11ccc-qcombbdbg.pdf

Making a development environment

- Good alternative
- Could work with almost all bands we want
- a little expensive: almost 300€
- requirements:

EC20 LTE modem


PCengines APU2

(Funny story about EC20)

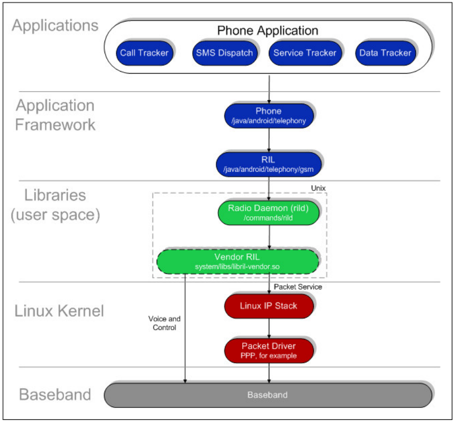
- Seen at 33c3 by Harald Welte[2] $\rightarrow$the modem runs an OE base Linux distribution
- It's also possible to have a shell via the AT command *AT+QLINUXCMD*:

```
# echo -e 'AT+QLINUXCMD="/sbin/getty -L ttyGS0 115200 console"\r\n' > /dev/ttyUSB2
# microcom /dev/ttyUSB1

OpenEmbedded Linux 9615-cdp ttyGS0

msm 20160923 9615-cdp ttyGS0

9615-cdp login: root
Password: oelinux123
root@9615-cdp:~#
```

[2]http git.gnumonks.org/laforge-slides/plain/2016/cellular_modems_33c3/33c3modems.html

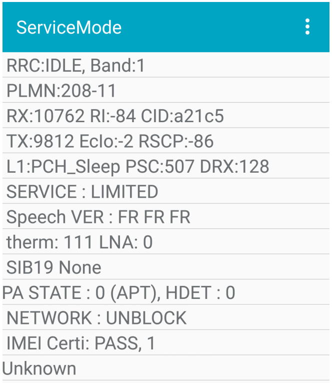
RIL on Android

- Daemon forwards commands/messages: application ⇆Vendor RIL
- vendor library is prorietary and vendor specific
- vendor library knows how to talk to modem:
  - classic AT
  - QMI for Qualcomm
  - (old?) Samsung IPC Protocol
  - and so on.



ServiceMode on Android

- Usually activated by typing a secret code
- Gives interesting details of current cell:
  - implicit network type
  - used band
  - reception (RX/DL) or/and transmission (TX/UP) (E/U)ARFCN (Absolute Radio Frequency Channel Number)
  - PLMN (Public Land Mobile Network) number
  - and so on.



```
RRC:IDLE, Band:1
PLMN:208-11
RX:10762 RI:-84 CID:a21c5
TX:9812 Eclo:-2 RSCP:-86
L1:PCH_Sleep PSC:507 DRX:128
SERVICE : LIMITED
Speech VER : FR FR FR
therm: 111 LNA: 0
SIB19 None
PA STATE : 0 (APT), HDET : 0
NETWORK : UNBLOCK
IMEI Certi: PASS, 1
Unknown
```

ServiceMode in Samsung

Samsung ServiceMode in brief

1. **#0011#* secret code handled by *ServiceModeApp_RIL ServiceModeApp* activity
2. ServiceModeApp →IPC connection →*SecFactoryPhoneTest SecPhoneService*
3. *ServiceModeApp* starts the service mode →*invokeOemRilRequestRaw()* through *SecPhoneService* (send RIL command *RIL_REQUEST_OEM_HOOK_RAW*)
4. *ServiceModeApp* process in higher level ServiceMode messages coming from RIL.

Best place to listen ServiceMode

Two good places exist: RIL library independent of Vendor RIL library implementation, or use *invokeOemRilRequestRaw()*

Getting SM messages: the lazy way

Ask to our best friend →logcat

```
shell@klte:/ $ logcat
[...]
I/ServiceModeApp_RIL( 1542): in QUERT_SERVM_DONE
I/ServiceModeApp_RIL( 1542): size of result : 1700
I/ServiceModeApp_RIL( 1542): Line 0 :  RRC:IDLE, Band:1_
I/ServiceModeApp_RIL( 1542): Line 1 :  PLMN:208-20_
I/ServiceModeApp_RIL( 1542): Line 2 :  RX:10639 RI:-70 CID:1fc09bd_
I/ServiceModeApp_RIL( 1542): Line 3 :  TX:9689 Eclo:-4 RSCP:-74_
I/ServiceModeApp_RIL( 1542): Line 4 :  L1:PCH_Sleep PSC:83 DRX:64_
I/ServiceModeApp_RIL( 1542): Line 5 :  SERVICE : LIMITED_
I/ServiceModeApp_RIL( 1542): Line 6 :  Speech VER : FR FR FR_
I/ServiceModeApp_RIL( 1542): Line 7 :  therm: 111 LNA: 0 _
I/ServiceModeApp_RIL( 1542): Line 8 :  SIB19 Received_
I/ServiceModeApp_RIL( 1542): Line 9 : PA STATE : 0 (APT), HDET : 0_
I/ServiceModeApp_RIL( 1542): Line 10 :  NETWORK : UNBLOCK_
I/ServiceModeApp_RIL( 1542): Line 11 :  IMEI Certi: PASS, 1_
```

Those messages could be then processed to get our current cell information.

What do I need?

At least a phone supporting ServiceMode!

Few contraints to resolve

"KTHX! But...:

1. how to support other operators different from your own SIM card? Do you need a different SIM card for each operator?
2. how to enumerate cells a MS (Mobile Station) is supposed to see?

Few contraints to resolve

“KTHX! But...:

1. how to support other operators different from your own SIM card? Do you need a different SIM card for each operator?
2. how to enumerate cells a MS (Mobile Station) is supposed to see?

Answer

The DFR technique!

DFR technique

D.F.R: "D" for Dirty, "F" for Fuzzy, "R" for Registration

The camping concept in brief

Let's remember 3GPP TS 43.022, ETSI TS 125 304...

- When selecting a PLMN $\rightarrow$MS looks for cells satisfying few conditions (cell of the selected PLMN, not barred, pathloss between MS and BTS below a thresold, and so on.)
- Cells are checked in a descending order of the signal strength
- If a suitable is found $\rightarrow$MS camps on it and tries to register

The camping concept in brief

Let's remember 3GPP TS 43.022, ETSI TS 125 304...

- When selecting a PLMN $\rightarrow$MS looks for cells satisfying few conditions (cell of the selected PLMN, not barred, pathloss between MS and BTS below a thresold, and so on.)
- Cells are checked in a descending order of the signal strength
- If a suitable is found $\rightarrow$MS camps on it and tries to register

Verified through DIAG and ServiceMode

If registration fails $\rightarrow$MS camps to another cell until it can register $\rightarrow$verified via DIAG and ServiceMode

Automate the DFR technique with AT commands

Android phones often expose a modem interface (e.g. */dev/smd0)*

```
127|shell@klte:/ $ getprop rild.libargs
-d /dev/smd0
```

It is possible to:

- set network type: *AT^SYSCONFIG*
- list PLNM and select a PLMN: *AT+COPS*

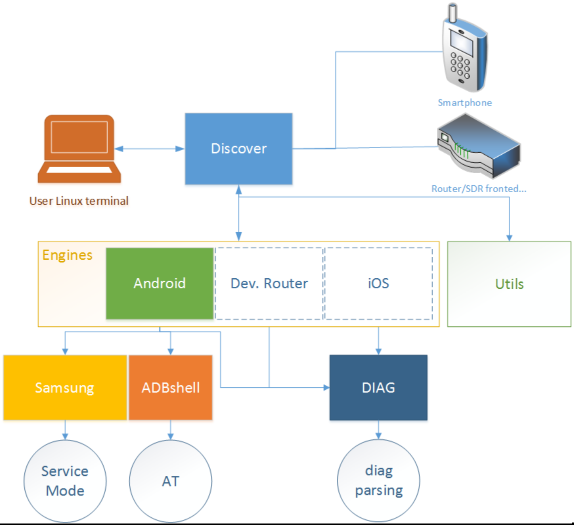
→requires root privileges

We mix all techniques together

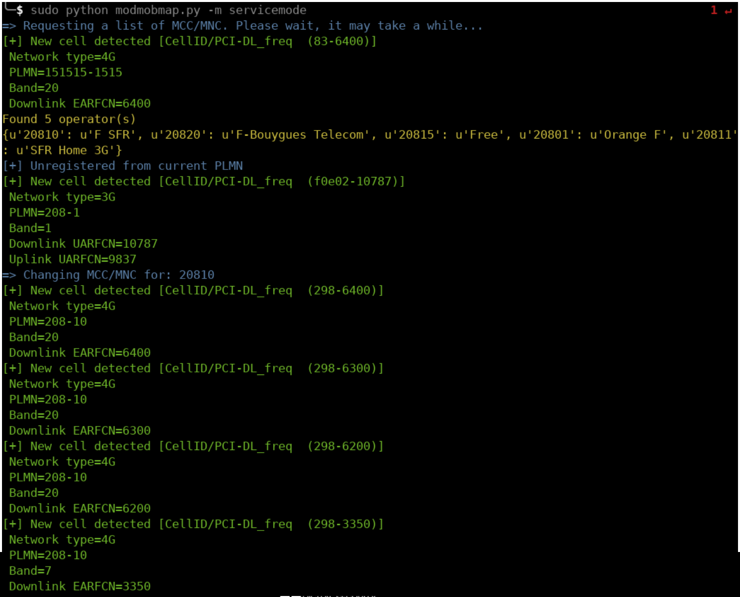
Don’t forget...

*the magic cure powder

Here is the frankenstein: modmobmap



Demo with a Galaxy S5 phone



```
└─$ sudo python modmobmap.py -m servicemode                                                    1 ↵
=> Requesting a list of MCC/MNC. Please wait, it may take a while...
[+] New cell detected [CellID/PCI-DL_freq  (83-6400)]
 Network type=4G
 PLMN=151515-1515
 Band=20
 Downlink EARFCN=6400
Found 5 operator(s)
{u'20810': u'F SFR', u'20820': u'F-Bouygues Telecom', u'20815': u'Free', u'20801': u'Orange F', u'20811'
: u'SFR Home 3G'}
[+] Unregistered from current PLMN
[+] New cell detected [CellID/PCI-DL_freq  (f0e02-10787)]
 Network type=3G
 PLMN=208-1
 Band=1
 Downlink UARFCN=10787
 Uplink UARFCN=9837
=> Changing MCC/MNC for: 20810
[+] New cell detected [CellID/PCI-DL_freq  (298-6400)]
 Network type=4G
 PLMN=208-10
 Band=20
 Downlink EARFCN=6400
[+] New cell detected [CellID/PCI-DL_freq  (298-6300)]
 Network type=4G
 PLMN=208-10
 Band=20
 Downlink EARFCN=6300
[+] New cell detected [CellID/PCI-DL_freq  (298-6200)]
 Network type=4G
 PLMN=208-10
 Band=20
 Downlink EARFCN=6200
[+] New cell detected [CellID/PCI-DL_freq  (298-3350)]
 Network type=4G
 PLMN=208-10
 Band=7
 Downlink EARFCN=3350
```

Conclusion

modmobmap:

- is a cheap way to scan mobile cells
- supports 2 useful interfaces:
  - ServiceMode;
  - GSMTAP captures:
    - host DIAG (could be easily extended for guest DIAG);
    - srsLTE and OpenLTE captures.
- the source code will be published in Github soon!
- any ideas and contribz are welcomed!

AVEZ-VOUS
DES QUESTIONS ?

MERCI DE VOTRE ATTENTION,

SYNACKTIV
DIGITAL SECURITY