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
1 Context
2 State of the Art
3 ServiceMode as an alternative
4 Make a tool out of it
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

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 cell monitor

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

2. State of the Art

3. Service Mode as an alternative

4. Make a tool out of it
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 → decoding → GSMTAP pseudo-header format
  - SnoopSnitch: not reflexible, but could be reworked for our purposes ;)

---

**SYNACKTIV**
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: \textit{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-qcombbdbgb.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\(^2\) → the modem runs an OE base Linux distribution
- It’s also possible to have a shell via the AT command \texttt{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 proprietary and vendor specific
- Vendor library knows how to talk to modem:
  - classic AT
  - QMI for Qualcomm
  - (old?) Samsung IPC Protocol
  - and so on.
1. Context
2. State of the Art
3. ServiceMode as an alternative
4. Make a tool out of it
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.

<table>
<thead>
<tr>
<th>ServiceMode</th>
</tr>
</thead>
<tbody>
<tr>
<td>RRC: IDLE, Band: 1</td>
</tr>
<tr>
<td>PLMN: 208-11</td>
</tr>
<tr>
<td>RX: 10762 Rl:-84 CID:a21c5</td>
</tr>
<tr>
<td>TX: 9812 Eclo:-2 RSCP:-86</td>
</tr>
<tr>
<td>L1: PCH_Sleep PSC: 507 DRX: 128</td>
</tr>
<tr>
<td>SERVICE: LIMITED</td>
</tr>
<tr>
<td>Speech VER: FR FR FR</td>
</tr>
<tr>
<td>therm: 111 LNA: 0</td>
</tr>
<tr>
<td>SIB19 None</td>
</tr>
<tr>
<td>PA STATE: 0 (APT), HDET: 0</td>
</tr>
<tr>
<td>NETWORK: UNBLOCK</td>
</tr>
<tr>
<td>IMEI Certi: PASS, 1</td>
</tr>
<tr>
<td>Unknown</td>
</tr>
</tbody>
</table>

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 4 : L1:PCH_Sleep PSC:83 DRX:64_
I /ServiceModeApp_RIL( 1542): Line 5 : SERVICE : LIMITED_
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.
1. Context
2. State of the Art
3. ServiceMode as an alternative
4. Make a tool out of it
What do I need?

At least a phone supporting ServiceMode!

Bonjour les GUEUX!
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 → MS looks for cells satisfying few conditions (cell of the selected PLMN, not barred, pathloss between MS and BTS below a threshold, and so on.)
- Cells are checked in a descending order of the signal strength
- If a suitable is found → 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 → MS looks for cells satisfying few conditions (cell of the selected PLMN, not barred, pathloss between MS and BTS below a threshold, and so on.)
- Cells are checked in a descending order of the signal strength
- If a suitable is found → MS camps on it and tries to register

Verified through DIAG and ServiceMode

If registration fails → MS camps to another cell until it can register → verified via DIAG and ServiceMode
Automate the DFR technique with AT commands

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

It is possible to:

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

→ requires root privileges

```
127|shell@klte:/ $ getprop rild.libargs
   -d /dev/smd0
```
We mix all techniques together
Don’t forget...

*the magic cure powder
Here is the frankenstein: modmobmap
Demo with a Galaxy S5 phone

```bash
$ sudo python modmobmap.py -m servicemode

=> 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'20020': u'F SFR', u'20020': 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 to: 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
```
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