SYNACKTIV
DIGITAL SECURITY

Modmobtools

Internals, updates and more

By Sébastien Dudek

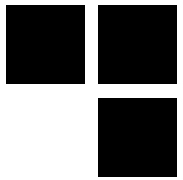
Troopers - Telco Sec Day

March 19th 2019

About me

- Sébastien Dudek (@FlUxIuS)
- Working at Synacktiv: pentests, red team, audits, vuln researches
- Likes radio and hardware
- And to confront theory vs. practice

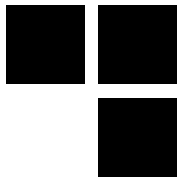
Introduction



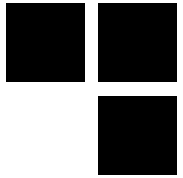
- Pentesting mobile devices (phones, intercoms, connected cars, ...) $\rightarrow$ right tools
- Data exchanged: (IoT) devices $\leftrightarrow$ server are generally trusted
- Spawn a fake station $\rightarrow$ OpenBTS/OsmoBTS, OpenBTS-UMTS, srsLTE, Amarisoft...
- But we need also to attract the device to this station
- Also sometimes it’s needed to perform cell monitoring on 2G/3G/4G and soon in 5G.

$\rightarrow$ we developped some cool&cheap tools to do that!

Our tools



- Modmobmap: monitoring 2G/3G/4G cells and more
- Modmobjam: smart/targeted jamming tools



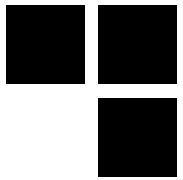
1 Modmobmap

2 Modmobjam

3 Updates

4 Conclusion

Where can I use this tool?

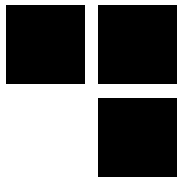


Cell towers discovery

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

Jamming

Where can I use this tool?



Cell towers discovery

Jamming

- replace the noisy chineese jammer
- avoid commercial jamming device reworking (bands disabling)

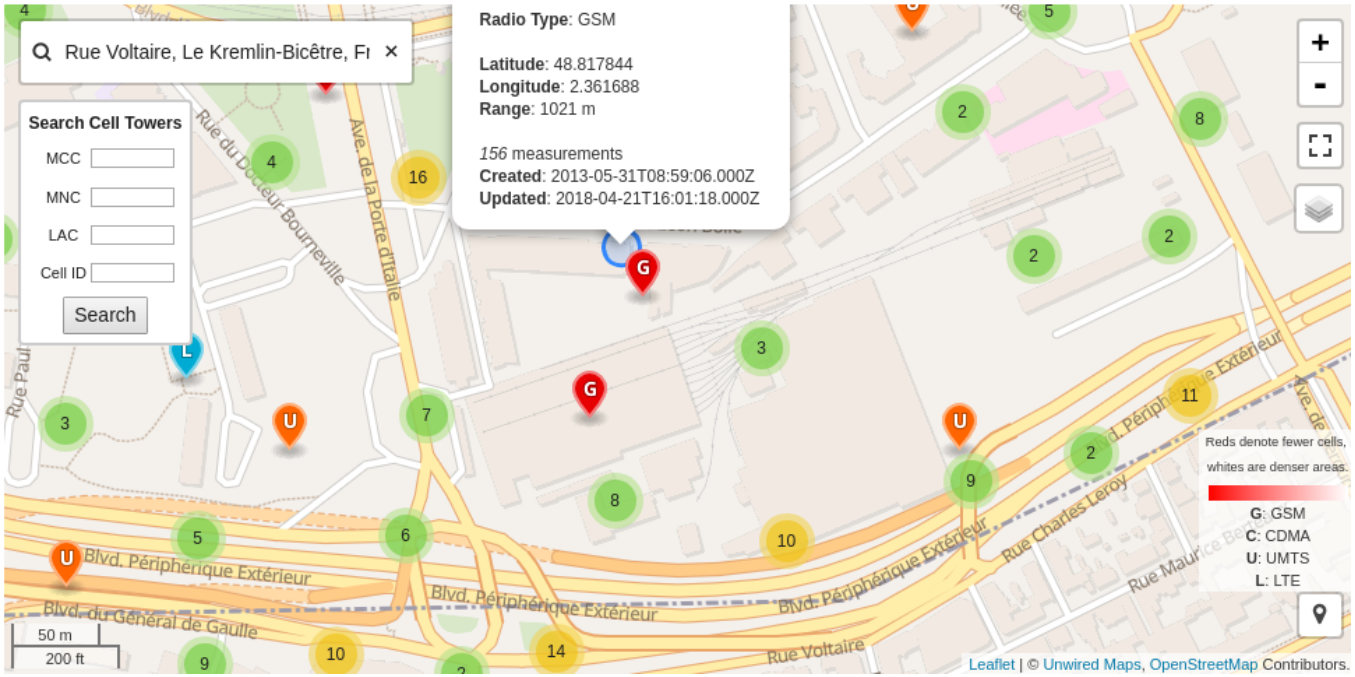
Remember: monitoring with holy relics

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.

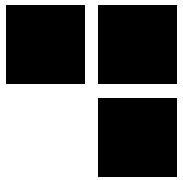
Thing that exists



OpenCellID example

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     |0xe    |n/a    |n/a    |-110   |   5   |-71
   3   |208    |01     |0x     |0xb    |n/a    |n/a    |-110   |   5   |-76
   7   |208    |01     |0x     |0xa    |n/a    |n/a    |-110   |   5   |-74
  11   |208    |01     |0x     |0xe    |n/a    |n/a    |-110   |   5   |-75
  77   |208    |10     |0x     |0x9    |no     |normal |-105   |   5   |-84
 513DCS|208    |01     |0x     |0xd    |n/a    |n/a    | -95   |   0   |-82
 518DCS|208    |01     |0x     |0x5    |n/a    |n/a    | -95   |   0   |-79
 609DCS|208    |01     |0x     |0xf    |n/a    |n/a    | -95   |   0   |-70
 744DCS|208    |10     |0x     |0xe    |n/a    |n/a    | -95   |   0   |-91
 976   |208    |20     |0x     |0xc    |n/a    |n/a    |-104   |   5   |-81
 978   |208    |20     |0x     |0xc    |n/a    |n/a    |-104   |   5   |-79
 979   |208    |20     |0x     |0x0    |n/a    |n/a    |-104   |   5   |-84
 982   |208    |20     |0x     |0xc    |n/a    |n/a    |-104   |   5   |-74
 984   |208    |20     |0x     |0xc    |n/a    |n/a    |-104   |   5   |-57
 986   |n/a    |n/a    |n/     |n/a    |n/a    |n/a    |n/a    |n/a    |n/a
1011   |208    |20     |0x     |0x9    |n/a    |n/a    |-104   |   5   |-87
1012   |208    |20     |0x     |0xb    |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

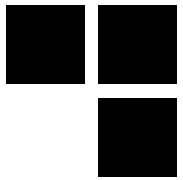
- OpenCellid: Open Database of Cell Towers
- Gsmmap.org
- and so on.

Problem!

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

Live scanning tools

Public tools

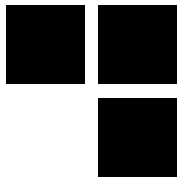


Recorded mobile towers

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: could be reworked for our purposes ;)

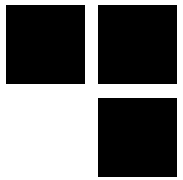
Methods to capture cells information



Possible methods are:

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

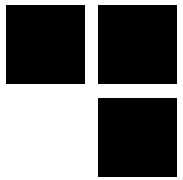
Software-Defined Radio



Existing tools:

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

Software-Defined Radio



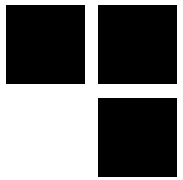
Existing tools:

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

No 3G

No 3G tools to capture cell information.

Exposed diagnostic interface



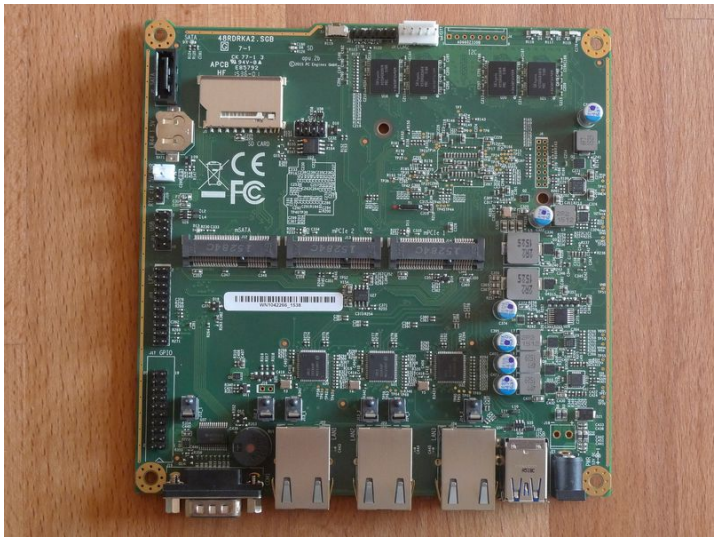
- Diagnostic interface enabled:
  - On old phones and 3G sticks like the *Icon 255*[1] that expose it by default
  - enabling DIAG ourselves: e.g for some LG devices via */sys/devices/platform/lg_diag_cmd/diag_enable*
  - Chips used for development
  - Interfaces kept enabled in production by error (e.g via custome bootmodes $\rightarrow$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

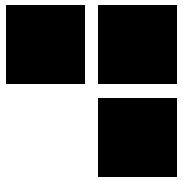
- Good alternative
- Could work with almost all bands we want
- A little expensive: almost 300€
- Requirements:

EC20 LTE modem



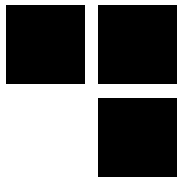
PCengines APU2

Supertramp’s version



- U/EC20 3G/LTE modem
- mPCI-E adapter

(Funny story about EC20)



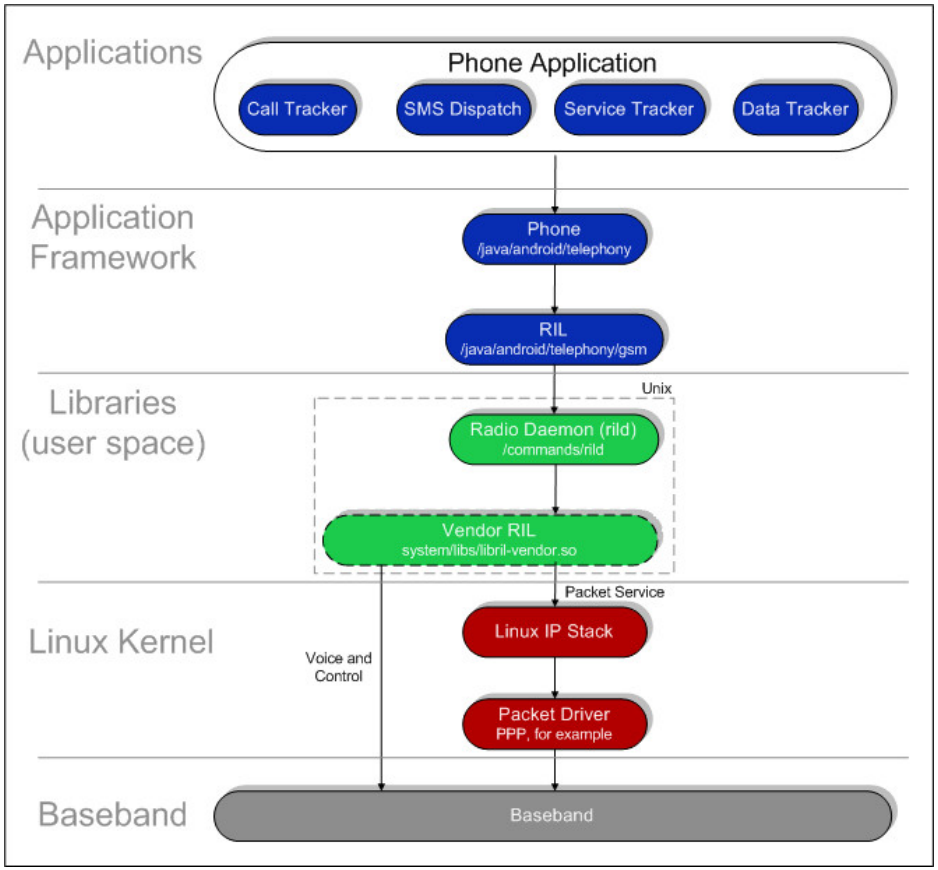
- Seen at 33c3 by Harald Welte[2] →the modem runs an OE base Linux distribution
- It's also possible to have a shell via the AT command *AT+QLINUXCMD*:

```
# 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

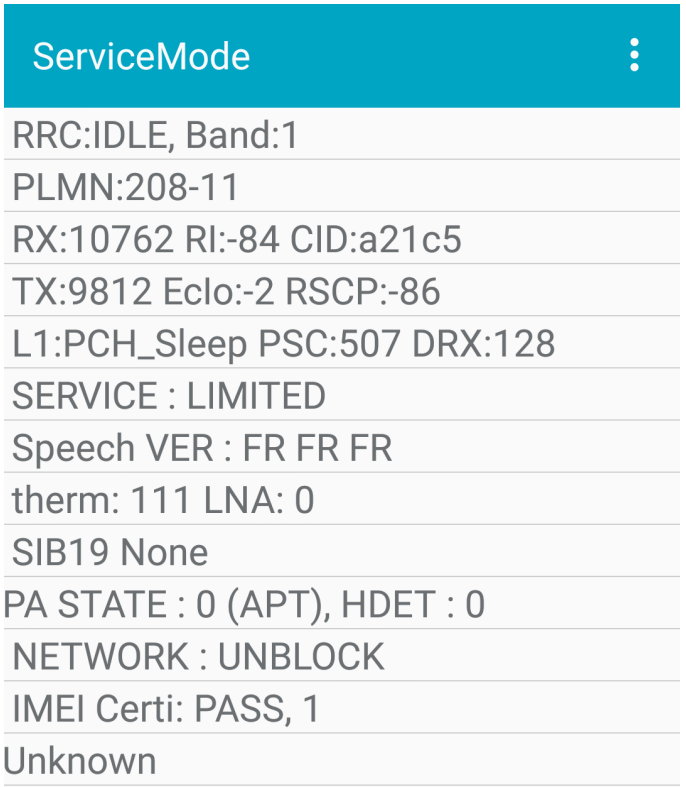
RIL on Android

- 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.



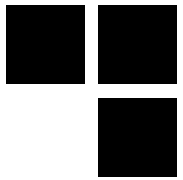
ServiceMode on Android

- 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.



ServiceMode in Samsung

Samsung ServiceMode in brief

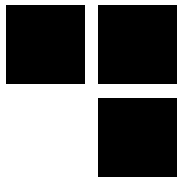


1. **#0011#* secret code handled by *ServiceModeApp_RIL ServiceModeApp* activity
2. ServiceModeApp →IPC connection →*SecFactoryPhoneTest SecPhoneService*
3. *ServiceModeApp* starts the service mode →*invokeOemRilRequestRaw()* through *SecPhoneService* (send RIL command *RIL_REQUEST_OEM_HOOK_RAW*)
4.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

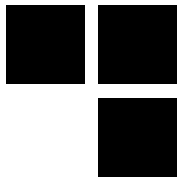


Ask to our best friend →logcat

```
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 EcIo:-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.

Getting data from DIAG with Xgoldmon

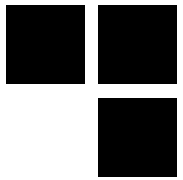


We have reworked Xgoldmon project for that:

- https://github.com/FlUxIuS/xgoldmon

```
$ cat ./celllog.fifo
[...]
[CellInfo]:PLMN=208-15;RAC=0x1;LAC=0x4e71;CID=0x1f****;DL_UARFCN=10737;UL_ARFCN=9787
[CellInfo]:PLMN=208-20;RAC=0x1;LAC=0x4e71;CID=0x1f****;DL_UARFCN=2950;UL_ARFCN=2725
[...]
[CellInfo]:PLMN=208-20;RAC=0x1;LAC=0xb5aa;CID=0x97****;DL_UARFCN=10639;UL_ARFCN=9689
[CellInfo]:PLMN=208-10;RAC=0x1;LAC=0xb5aa;CID=0x97****;DL_UARFCN=65535;UL_ARFCN=2850
[...]
```

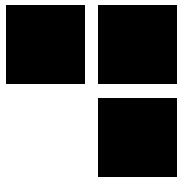
What do I need?



At least a phone supporting ServiceMode!

- At least supports following tested phones:
  - Samsung Galaxy S3 via xgoldmon (Modmobmap's edition);
  - Samsung Galaxy S4;
  - Samsung Galaxy S5;
  - Samsung Galaxy Note 2 with LTE;
  - Samsung Galaxy S4 GT-I9500
  - Samsung Galaxy Nexus GT-I9250
  - Samsung Galaxy S2 GT-I9100
  - Samsung Galaxy Note 2 GT-N7100
  - Samsung Galaxy S6 Exynos SoC
  - Samsung Galaxy S7 Exynos SoC
  - Samsung Galaxy A3 Exynos SoC
  - ...

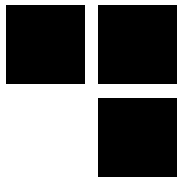
Few contraints to resolve



“KTHX! But there are 2 questions”:

1. how to support other operators than your own SIM card?
2. how to enumerate cells a MS (Mobile Station) is supposed to see?

Few contraints to resolve



“KTHX! But there are 2 questions”:

1. how to support other operators than your own SIM card?
2. how to enumerate cells a MS (Mobile Station) is supposed to see?

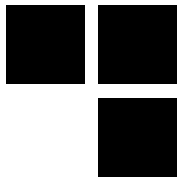
Answer

The DFR technique!

DFR technique

D.F.R: “D” for Dirty, “F” for Fuzzy, “R” for Registration

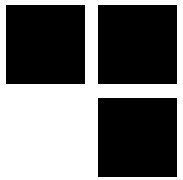
The camping concept in brief



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 thresold, 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



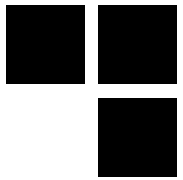
Let’s remember 3GPP TS 43.022, ETSI TS 125 304...

- 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 thresold, 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)*

```
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*

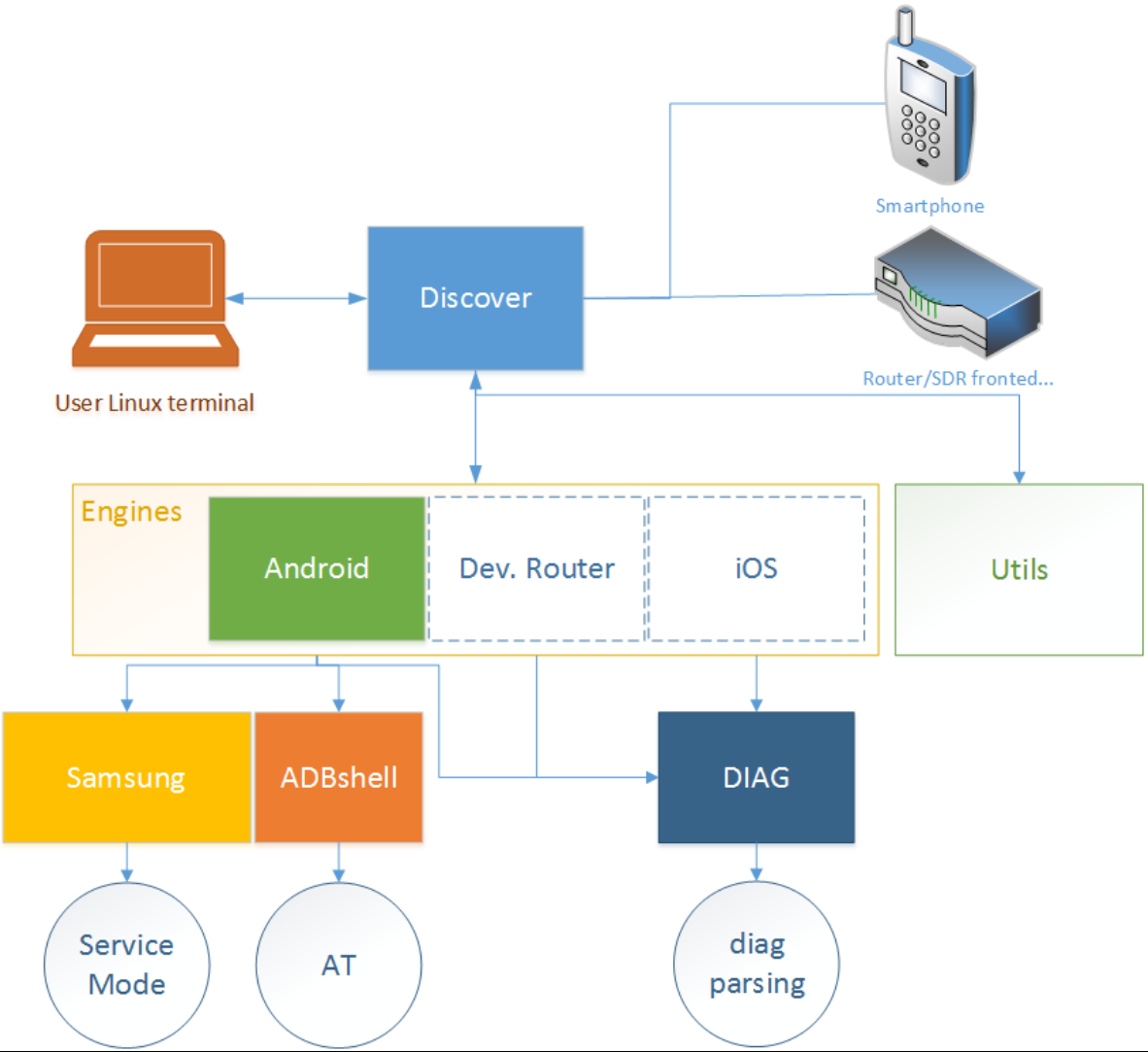
→requires root privileges

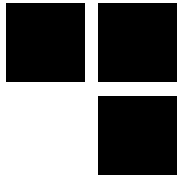
We mix all techniques together

Don’t forget...

*the magic cure powder

Here is the frankenstein: Modmobmap





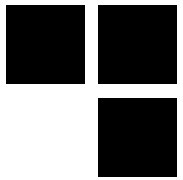
1 Modmobmap

2 Modmobjam

3 Updates

4 Conclusion

In brief



- Uses Modmobmap results to jam mobile cells in a DIY way!
- Cheapest and efficient tricks to jam

Before

With a portable/chineese device

- cheap
- jam the whole 2G/3G/(4G?) bands but requires some modifications
- poor signal

Desktop jammers

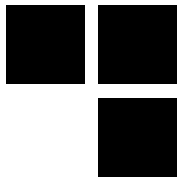
Before

With a portable/chineese device

Desktop jammers

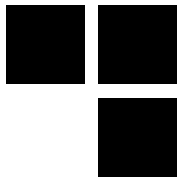
- heavy, cumbersome but powerfull
- also needs a disabling to conserve rogue cells

Software-Defined Radio way



- With Software-Defined Radio
- Many devices could be used even the cheapest:
  - bladeRF;
  - HackRF;
  - ADALM-PLUTO;
  - and so on.

Software-Defined Radio way

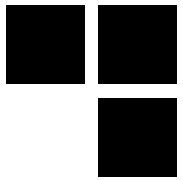


- With Software-Defined Radio
- Many devices could be used even the cheapest:
  - bladeRF;
  - HackRF;
  - ADALM-PLUTO;
  - and so on.

The bandwidth

KTHX! But how do you cover all frequencies with your toys bro?

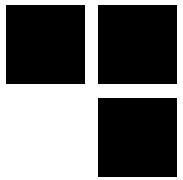
SDR specs



	HackRF	bladeRF		USRP		
		x40	x115	B100 Starter	B200	B210
Radio Spectrum	30 MHz – 6 GHz	300 MHz – 3.8 GHz		50 MHz – 2.2 GHz [1]	50MHz – 6 GHz	
Bandwidth	20 MHz	28 MHz		16 MHz [2]	61.44 MHz [3]	
Duplex	Half	Full		Full	Full	2x2 MIMO
Sample Size (ADC/DAC)	8 bit	12 bit		12 bit / 14 bit	12 bit	
Sample Rate (ADC/DAC)	20 Msps	40 Msps		64 Msps / 128 Msps	61.44 Msps	
Interface (Speed)	USB 2 HS (480 megabit)	USB 3 (5 gigabit)		USB 2 HS (480 megabit)	USB 3 (5 gigabit)	
FPGA Logic Elements	[4]	40k	115k	25k	75k	150k
Microcontroller	LPC43XX	Cypress FX3		Cypress FX2	Cypress FX3	
Open Source	Everything	HDL + Code Schematics		HDL + Code Schematics	Host Code [5]	
Availability	January 2014	Now		Now	Now	
Cost	$300 [6]	$420	$650	$675	$675	$1100

source: http://www.taylorkillian.com/2013/08/sdr-showdown-hackrf-vs-bladerf-vs-usrp.html

Solution: ”Smart” jamming



In 3 steps:

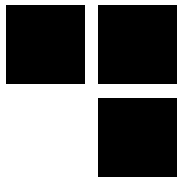
1. scan cells with Modmobmap;
2. target an operator;
3. and jam only targeted channels;

Scanning with Modmobmap

Modmobmap recovers 2G/3G/4G and more cells pretty much like OsmocomBB monitor mode for 2G only.

```
└─$ 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
```

Results

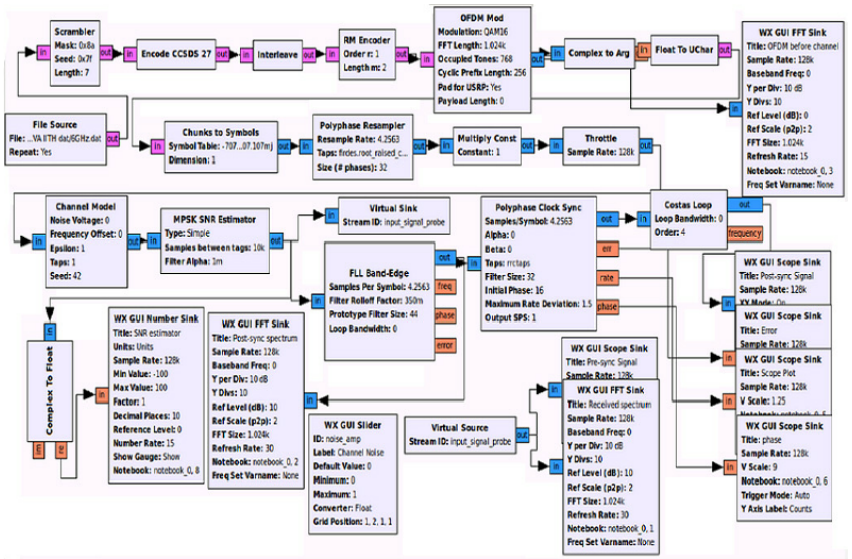


JSON file → needed cells information to be reused with other tools, like Modmobjam!

```
{
    "4b***-76": {
        "PLMN": "208-10",
        "arfcn": 76,
        "cid": "4b**",
        "type": "2G"
    },
    "60****-2950": {
        "PLMN": "208-20",
        "RX": 2950,
        "TX": 2725,
        "cid" : 60***,
        "band": 8,
        "type": "3G"
    },
[...]
}
```

GnuRadio: playing with blocks

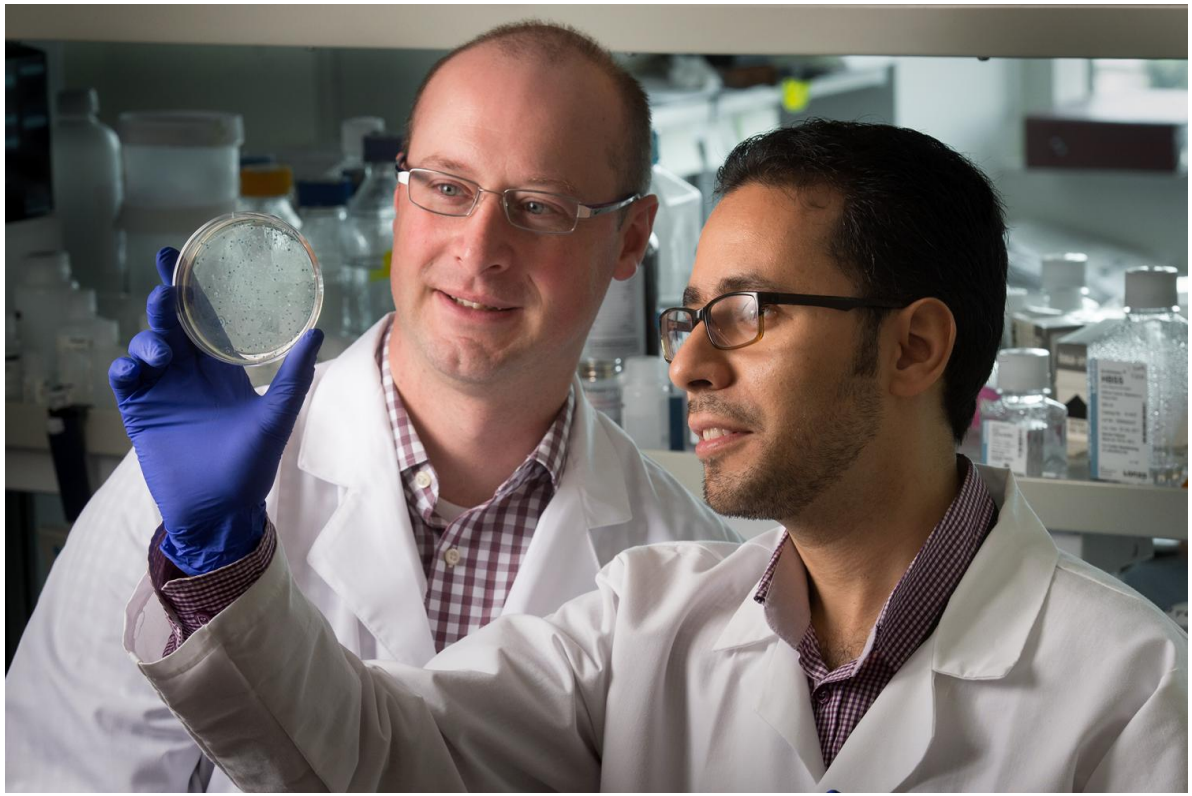
GnuRadio companion is really nice →can add, make, and remove blocks →generates Python code



Perfect to build the bases of our jammer. But we still need an idea of how to design the schema.

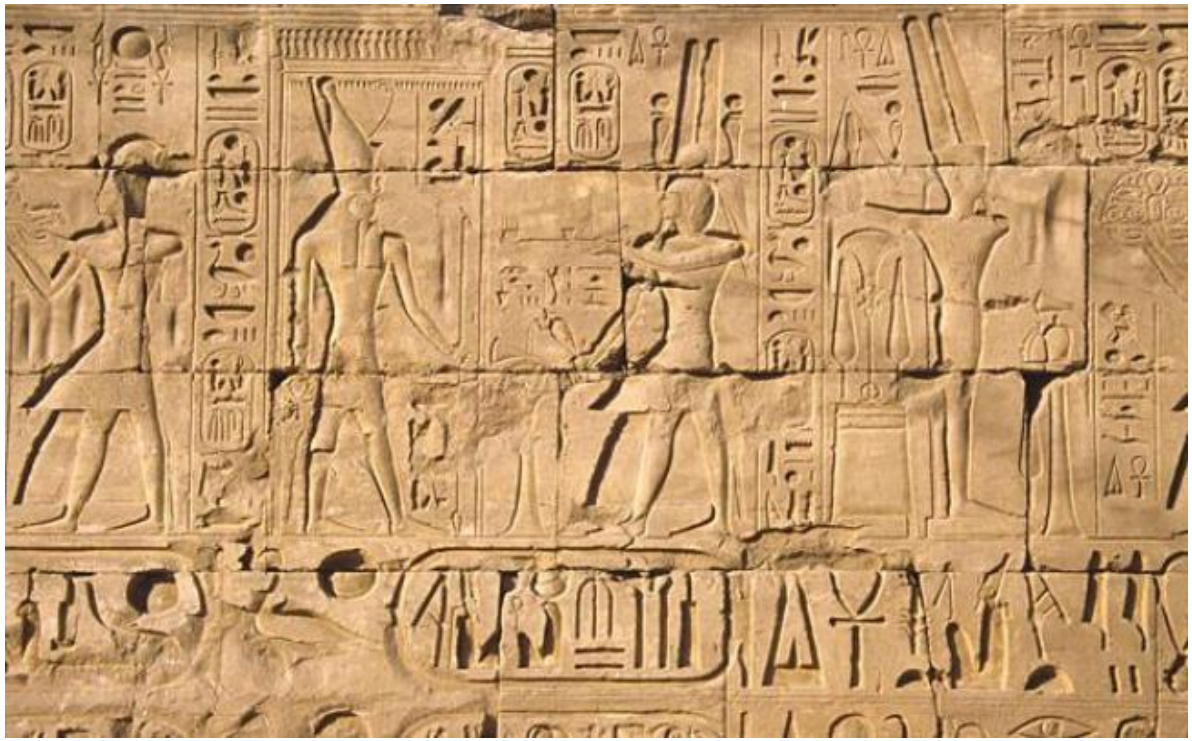
After many years of research...

Lot of experiments with blocks != #blockchains... blablabla



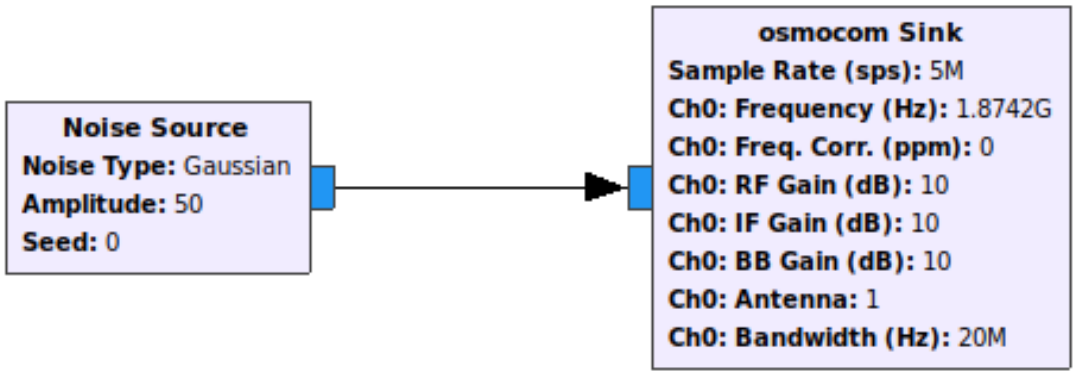
The formula

We have finally found THE formula!



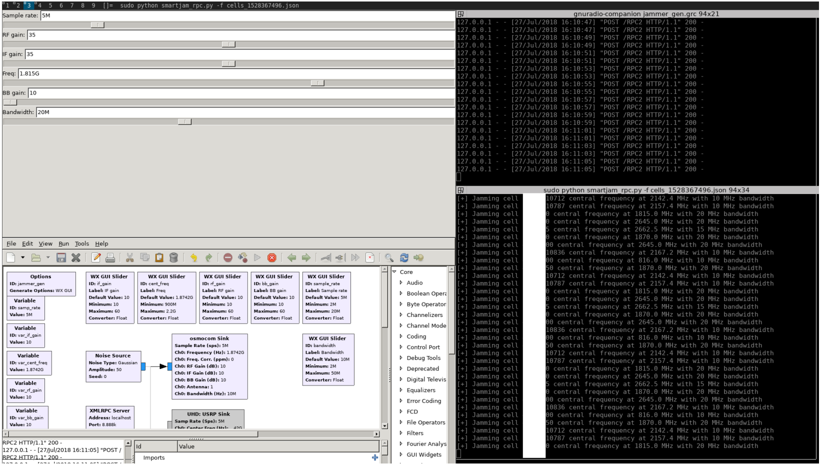
Experimentation with GnuRadio

So we've started with a simple schema:



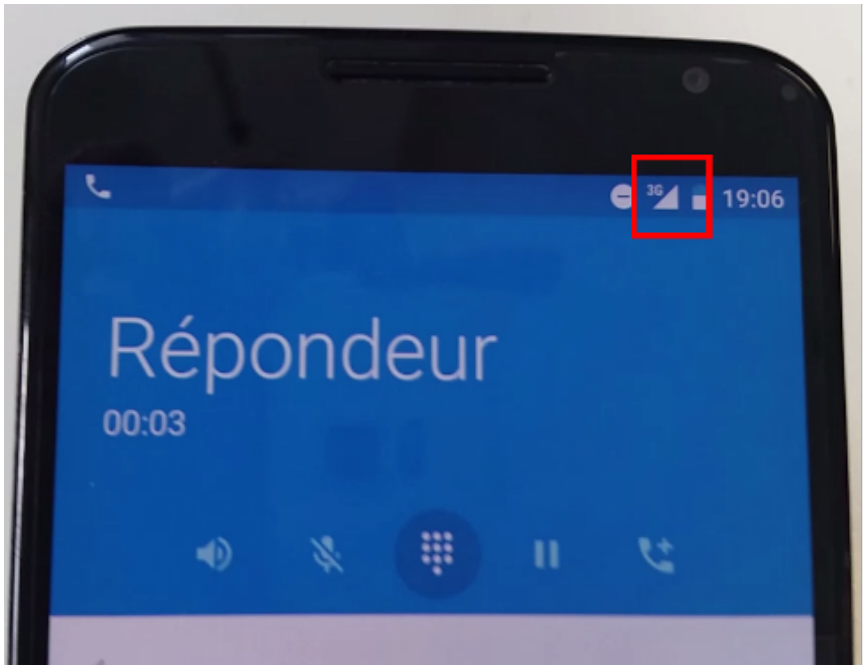
But still needed some work...

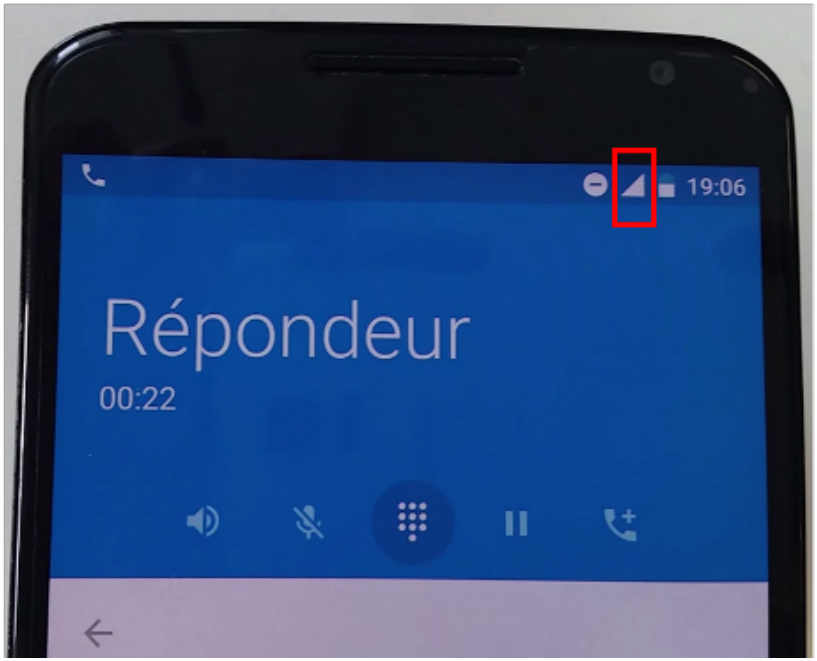
Final product: Modmobjam



Results with a simple HackRF

Works pretty well when downgrading a call from 3G to 2G





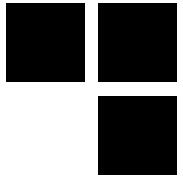
But the number of cells to jam could raise the number of needed SDR devices.

Go cheaper

Could also be cheaper using *OsmoFL2k*

TODO

Some work is required target specific frequencies →right sample rate, carrier frequency and harmonics + better ant & amp



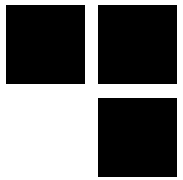
1 Modmobmap

2 Modmobjam

3 Updates

4 Conclusion

Next updates



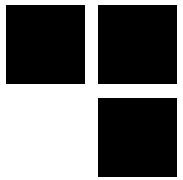
- Add RSSI when possible
- Add support of mPCI-E modems with exposed DIAG
- Add more mobile phone supports → based on SCAT tool
- And more! → add also your contribution

Getting data from exposed DIAG on mPCI-E modems

- Just use *diag-parser* tool from Moiji Mobile

The rest could be parser with *pycrate_mobile* library of Benoit Michau →ASN.1 and CSN.1 compilers included for our purposes (RRC, and so on)!

tshark with Wireshark dissectors



But in the train for Troopers, I got lazy:

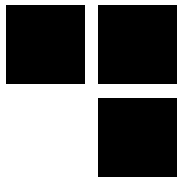
- Launch *diag-parser* and output result in a FIFO file:

```
$ sudo ./diag_parser –g 127.0.0.1 –p /tmp/fifoin –i /dev/ttyUSB0 –vvv
```

- and dissect all LTE and UTRA_FDD carrier list:

```
cat /tmp/fifoin|tshark –i– –l –n –T json –e gsmtap.arfcn –e lte_rrc
–e lte–rrc.trackingAreaCode –e lte–rrc.cellIdentity –e lte–rrc.q_RxLevMin
–e lte–rrc.freqBandIndicator –e lte–rrc.MCC_MNC_Digit
–e lte–rrc.carrierFreqListUTRA_FDD
–e lte–rrc.carrierFreq –e lte–rrc.interFreqCarrierFreqList –e lte–rrc.dl_CarrierFreq
–e lte–rrc.q_RxLevMin –e lte–rrc.physCellId –Y 'gsmtap.arfcn!=0' > /tmp/fifoout
```

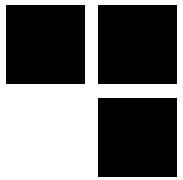
tshark result



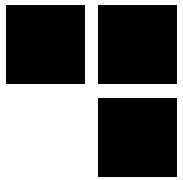
tshark gives us a nice JSON render:

```
{
    "layers": {
      "gsmtap.arfcn": ["6200"],
      "lte_rrc": ["lte_rrc"],
      "lte-rrc.trackingAreaCode": ["75:c2"],
      "lte-rrc.cellIdentity": ["7a:2a:20:80"],
      "lte-rrc.freqBandIndicator": ["20"],
      "lte-rrc.MCC_MNC_Digit": ["2","0","8","2","0"],
      "lte-rrc.q_RxLevMin": ["-61"]
    }
  }
}
[...]
  {
    "layers": {
      "gsmtap.arfcn": ["6200"],
      "lte_rrc": ["lte_rrc"],
      "lte-rrc.interFreqCarrierFreqList": ["3"],
      "lte-rrc.dl_CarrierFreq": ["1850","3175","251"],
      "lte-rrc.q_RxLevMin": ["-63","-62","-63"],
      "lte-rrc.physCellId": ["158"]
    }
  }
}
[...]
        "lte-rrc.carrierFreq": ["10639","10688","10664","2950"]
[...]
```

DIAG for the rock!



- Less abstracted data
- Carrier lists $\rightarrow$ catch a bunch of 3G and LTE DL freqs in the same time
- More optimized for mobile monitoring and attacks...
- Support with the tshark JSON output will be comitted soon
- Another support with *pycrate_mobile* to parse RRC messages $\rightarrow$ in the TODO stack!



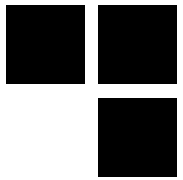
1 Modmobmap

2 Modmobjam

3 Updates

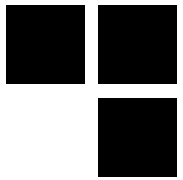
4 Conclusion

Conclusion



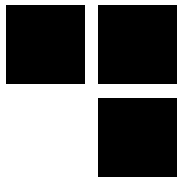
- Modmobmap:
  - is a cheap way to scan mobile cells
  - supports 2 useful interfaces:
    - ServiceMode;
    - host DIAG (could be easily extended for guest DIAG);
    - srsLTE and OpenLTE captures soon...
- Modmobjam:
  - is a cheap way to jam mobile cells with only a phone and a HackRF
  - but if cells to jam are important more SDR devices are needed

Downloads



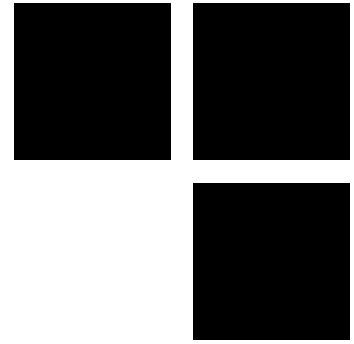
- Modmobmap:
  - https://github.com/Synacktiv/Modmobmap
- Modmobjam:
  - https://github.com/Synacktiv/Modmobjam

Thanks =)



- Joffrey Czarny (@_Sn0rkY)
- Priya Chalakkal (@priyachalakkal)
- Troopers staff (@WEareTROOPERS)
- And of course → You all ;)

ANY QUESTIONS?



THANK YOU FOR YOUR ATTENTION,

SYNACKTIV
DIGITAL SECURITY