OsmocomBB

Sending arbitrary protocol data to GSM networks

Harald Welte

gnumonks.org
gpl-violations.org
OpenBSC
airprobe.org
hmw-consulting.de

ph-neutral 2010, May 2010, Berlin/Germany

Outline

1. GSM/3G Network Security Introduction
2. Security Problems and the Baseband
3. OsmocomBB Project
4. Summary

About the speaker

- Using + playing with Linux since 1994
- Kernel / bootloader / driver / firmware development since 1999
- IT security specialist, focus on network protocol security
- Board-level Electrical Engineering
- Always looking for interesting protocols (RFID, DECT, GSM)

GSM/3G protocol security

- Observation
  - Both GSM/3G and TCP/IP protocol specs are publicly available
  - The Internet protocol stack (Ethernet/Wifi/TCP/IP) receives lots of scrutiny
  - GSM networks are as widely deployed as the Internet
  - Yet, GSM/3G protocols receive no such scrutiny!
- There are reasons for that:
  - GSM industry is extremely closed (and closed-minded)
  - Only about 4 closed-source protocol stack implementations
  - GSM chipset makers never release any hardware documentation

The closed GSM industry

Handset manufacturing side

- Only very few companies build GSM/3.5G baseband chips today
  - Those companies buy the operating system kernel and the protocol stack from third parties
- Only very few handset makers are large enough to become a customer
  - Even they only get limited access to hardware documentation
  - Even they never really get access to the firmware source

The closed GSM industry

Network manufacturing side

- Only very few companies build GSM network equipment
  - Basically only Ericsson, Nokia-Siemens, Alcatel-Lucent and Huawei
  - Exception: Small equipment manufacturers for picocell / nanocell / femtocells / measurement devices and law enforcement equipment
- Only operators buy equipment from them
- Since the quantities are low, the prices are extremely high
  - e.g. for a BTS, easily 10-40k EUR

The closed GSM industry
Operator side

- Operators are mainly banks today
- Typical operator outsources
  - Billing
  - Network planning / deployment / servicing
- Operator just knows the closed equipment as shipped by manufacturer
- Very few people at an operator have knowledge of the protocol beyond what's needed for operations and maintenance

The closed GSM industry
Security implications

The security implications of the closed GSM industry are:

- Almost no people who have detailed technical knowledge outside the protocol stack or GSM network equipment manufacturers
- No independent research on protocol-level security
  - If there's security research at all, then only theoretical (like the A5/2 and A5/1 cryptanalysis)
  - Or on application level (e.g. mobile malware)
- No open source protocol implementations
  - which are key for making more people learn about the protocols
  - which enable quick prototyping/testing by modifying existing code

Security analysis of GSM

How would you get started?

If you were to start with GSM protocol level security analysis, where and how would you start?

- On the network side?
  - Difficult since equipment is not easily available and normally extremely expensive
  - However, network is very modular and has many standardized/documented interfaces
  - Thus, if equipment is available, much easier/faster progress
  - Has been done in 2008/2009: Project OpenBSC

Security analysis of GSM

How would you get started?

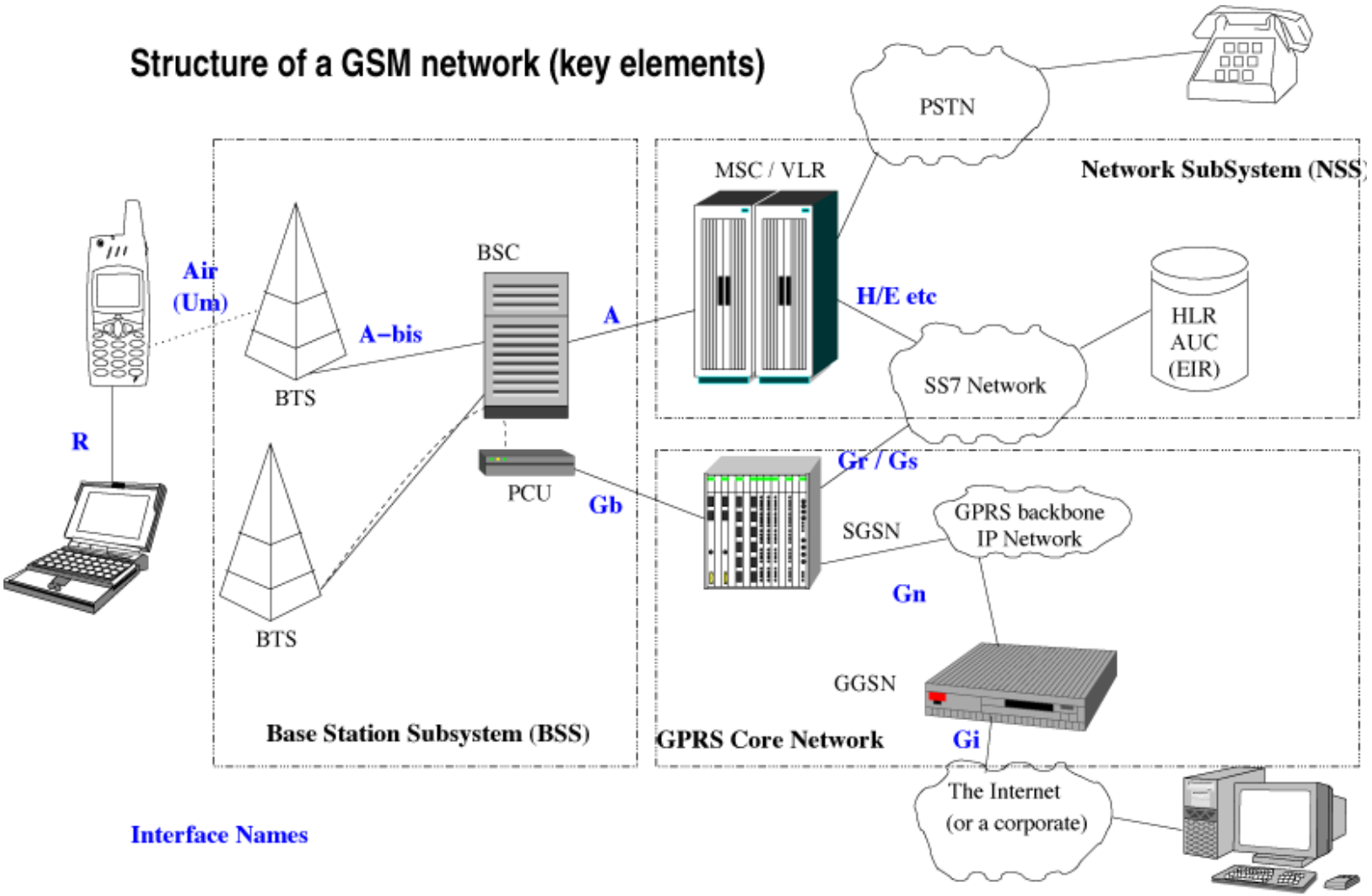
If you were to start with GSM protocol level security analysis, where and how would you start?

- On the handset side?
  - Difficult since GSM firmware and protocol stacks are closed and proprietary
  - Even if you want to write your own protocol stack, the layer 1 hardware and signal processing is closed and undocumented, too
  - Known attempts
    - The TSM30 project as part of the THC GSM project
    - mados, an alternative OS for Nokia DTC3 phones
  - none of those projects successful so far

Security analysis of GSM
The bootstrapping process

- Read GSM specs day and night (> 1000 PDF documents)
- Gradually grow knowledge about the protocols
- Obtain actual GSM network equipment (BTS, MS tester, ...)
- Try to get actual protocol traces as examples
- Start a complete protocol stack implementation from scratch
- Finally, go and play with GSM protocol security

The GSM network



GSM network components

- The BSS (Base Station Subsystem)
  - MS (Mobile Station): Your phone
  - BTS (Base Transceiver Station): The *cell tower*
  - BSC (Base Station Controller): Controlling up to hundreds of BTS
- The NSS (Network Sub System)
  - MSC (Mobile Switching Center): The central switch
  - HLR (Home Location Register): Database of subscribers
  - AUC (Authentication Center): Database of authentication keys
  - VLR (Visitor Location Register): For roaming users
  - EIR (Equipment Identity Register): To block stolen phones

GSM network interfaces

- Um: Interface between MS and BTS
  - the only interface that is specified over radio
- A-bis: Interface between BTS and BSC
- A: Interface between BSC and MSC
- B: Interface between MSC and other MSC

GSM networks are a prime example of an asymmetric distributed network, very different from the end-to-end transparent IP network.

GSM network protocols
On the Um interface

- Layer 1: Radio Layer, TS 04.04
- Layer 2: LAPDm, TS 04.06
- Layer 3: Radio Resource, Mobility Management, Call Control: TS 04.08
- Layer 4+: for USSD, SMS, LCS, ...

Known GSM security problems

Scientific papers, etc

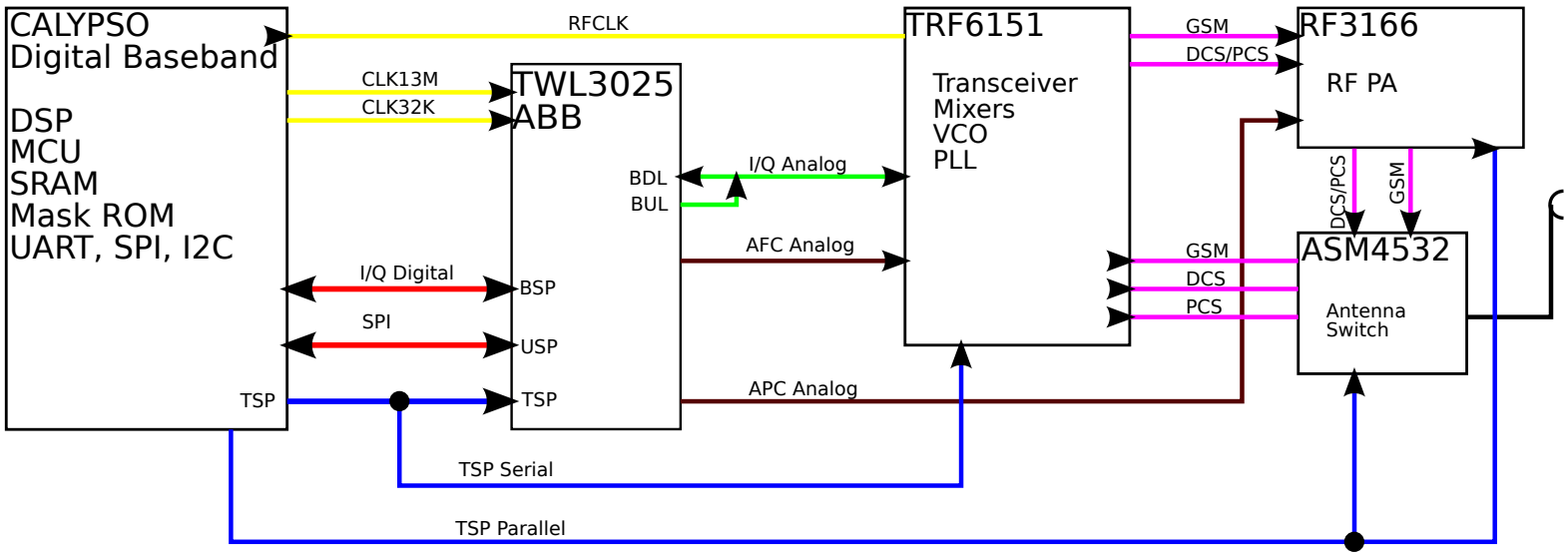
- No mutual authentication between phone and network
  - leads to rogue network attacks
  - leads to man-in-the-middle attacks
  - is what enables IMSI-catchers
- Weak encryption algorithms
- Encryption is optional, user does never know when it's active or not
- DoS of the RACH by means of channel request flooding
- RRLP (Radio Resource Location Protocol)
  - the network can obtain GPS fix or even raw GSM data from the phone
  - combine that with the network not needing to authenticate itself

Known GSM security problems

The Baseband side

- GSM protocol stack always runs in a so-called baseband processor (BP)
- What is the baseband processor
  - Typically ARM7 (2G/2.5G phones) or ARM9 (3G/3.5G phones)
    - Runs some RTOS (often Nucleus, sometimes L4)
    - No memory protection between tasks
  - Some kind of DSP, model depends on vendor
    - Runs the digital signal processing for the RF Layer 1
    - Has hardware peripherals for A5 encryption
- The software stack on the baseband processor
  - is written in C and assembly
  - lacks any modern security features (stack protection, non-executable pages, address space randomization, ..)

A GSM Baseband Chipset



http://laforge.gnumonks.org/papers/gsm_phone-anatomy-latest.pdf

Interesting observations

Learned from implementing the stack

While developing OpenBSC, we observed a number of interesting

- Many phones use their TMSI from the old network when they roam to a new network
- Various phones crash when confronted with incorrect messages. We didn't even start to intentionally send incorrect messages (!)
- There are tons of obscure options on the GSM spec which no real network uses. Potential attack vector by using rarely tested code paths.

GSM Protocol Fuzzing

Theoretical basis

How to do GSM protocol fuzzing

- From the handset to the network
  - Basically impossible due to closeness of baseband
  - However, some incomplete projects working on it
- From the network side
  - Easy in case of *rogue network* attacks
  - Fuzzing target is the GSM stack in the baseband processor
- As an A-bis man in the middle
  - Needs access to an A-bis interface of an actual network
  - Very attractive, since no encryption and ability to fuzz both network and handset

scapy GSM support
The actual fuzzing

How to actually craft the packets for the fuzzing

- GSM has many, many protocols
- Writing custom code will be a hard-coded special case for each of them
- Solution: Use scapy and implement the GSM protocols as scapy "layers"
  - IPA protocol header
  - RSL protocol layer
  - RLL data indication / data request
  - GSM 04.08 RR / MM / CC messages

OsmoocmBB Introduction

- Project was started in January 2010
- Implementing a GSM baseband software from scratch
- This includes
  - GSM MS-side protocl stack from Layer 1 through Layer 3
  - Hardware drivers for GSM Baseband chipset
  - Simple User Interface on the phone itself
  - Verbose User Interface on the PC
- Note about the strange project name
  - Osmocom = Open Source MObile COMmunication
  - BB = Base Band

OsmoocmBB Software Architecture

- Reuse code from OpenBSC where possible (libosmocore)
  - We build libosmocore both for phone firmware and PC
- Initially run as little software in the phone
  - Debugging code on your host PC is so much easier
  - You have much more screen real-estate
  - Hardware drivers and Layer1 run in the phone
  - Layer2, 3 and actual phone application / MMI on PC
  - Later, L2 and L3 can me moved to the phone

OsmoocmBB Software Interfaces

- Interface between Layer1 and Layer2 called L1CTL
  - Fully custom protocol as there is no standard
  - Implemented as message based protocol over Sercomm/HDLC/RS232
- Interface between Layer2 and Layer3 called RSLms
  - In the GSM network, Um Layer2 terminates at the BTS but is controlled by the BSC
  - Reuse this GSM 08.58 Radio Signalling Link
  - Extend it where needed for the MS case

OsmoocmBB Target Firmware

- Firmware includes software like
  - Drivers for the Ti Calypso Digital Baseband (DBB)
  - Drivers for the Ti Iota TWL3025 Analog Baseband (ABB)
  - Drivers for the Ti Rita TRF6151 RF Transceiver
  - Drivers for the LCD/LCM of a number of phones
  - CFI flash driver for NOR flash
  - GSM Layer1 synchronous/asynchronous part
  - Sercomm - A HDLC based multiplexer for the RS232 to host PC

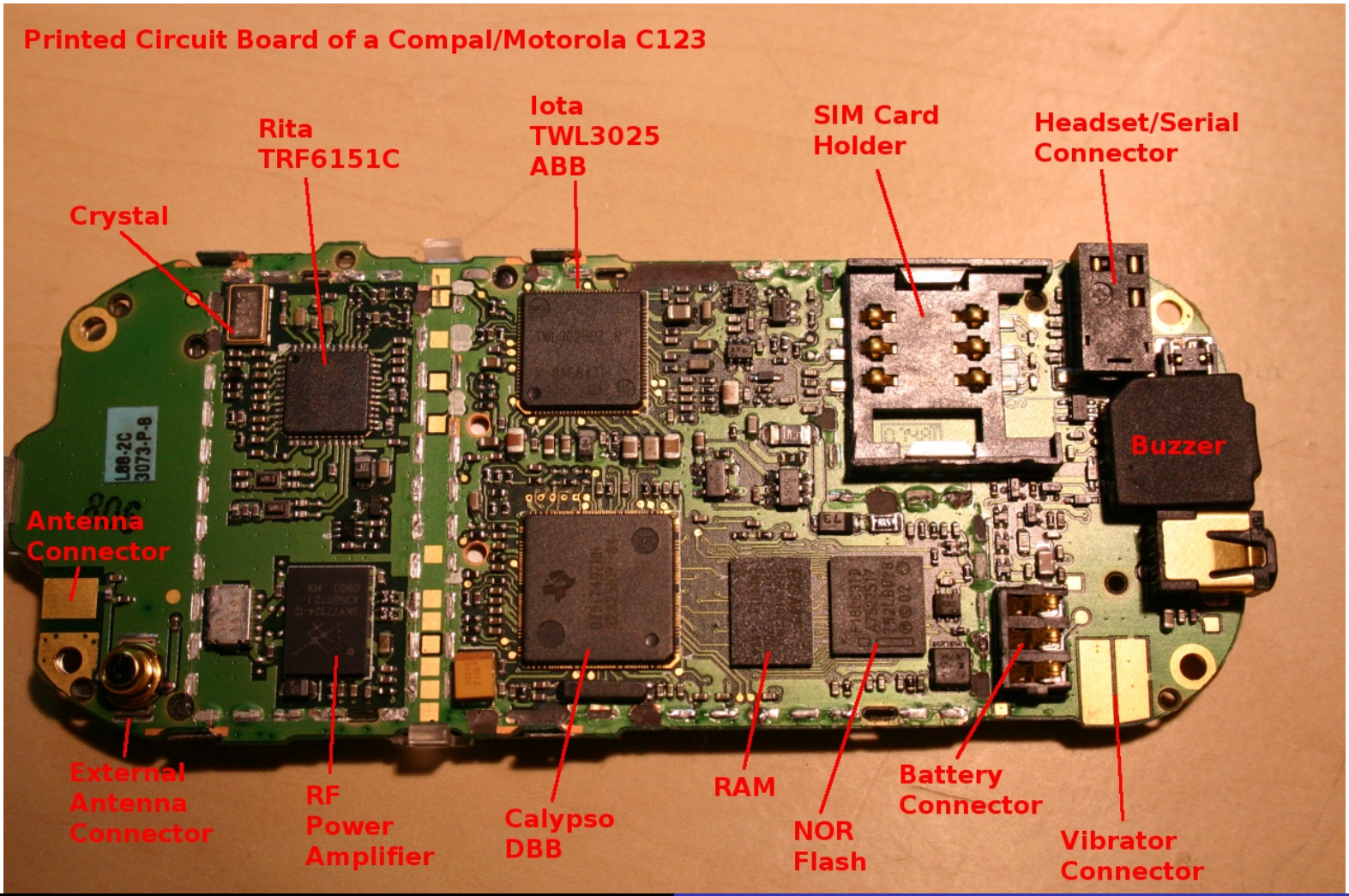
OsmoocmBB Host Software

- Current working name: layer23
- Includes
  - Layer 1 Control (L1CTL) protocol API
  - GSM Layer2 implementation (LAPDm)
  - GSM Layer3 implementation (RR/MM/CC)
  - GSM Cell (re)selection
  - SIM Card emulation
  - Supports various 'apps' depending on purpose

OsmoocmBB Supported Hardware

- Baseband Chipsets
  - TI Calypso/Iota/Rita
  - Some early research being doen on Mediatek (MTK) MT622x
- Actual Phones
  - Compal/Motorola C11x, C12x, C13x, C14x and C15x models
  - Most development/testing on C123 and C155
  - GSM modem part of Openmoko Neo1973 and Freerunner
- All those phones are simple feature phones built on a ARM7TDMI based DBB

The Motorola/Compal C123



OsmoocmBB Project Status: Working

- Hardware Drivers for Calypso/Iota/Rita very complete
- Layer1
  - Power measurements
  - Carrier/bit/TDMA synchronization
  - Receive and trnasmit of normal bursts on SDCCH
  - Transmit of RACH bursts
- Layer2 UI/SABM/UA frames
- Layer3 Messages for RR / MM / CC
- Cell (re)selection according GSM 03.22

OsmoocmBB Project Status: Not working

- Actual SIM card reader inside phone (WIP)
- Layer1
  - Automatic Tx power control (APC)
  - Automatic Rx gain control (AGC)
  - Frequency Hopping
  - Neighbor Cell Measurements
  - Traffic Channels (TCH)
- Layer2 Asynchronous Balanced Mode (ACK/retransmissions)
- Actual UI on the phone
- Drivers for Audio/Voice signal path

OsmoocmBB Project Status: Executive Summary

- We can esetablish control/signalling channels with non-hopping cells
  - Used in small single-TRX cells in rural areas
  - Used in GSM-R networks
  - As provided by OpenBSC + OpenBTS
- We can send arbitrary data on those control channels
  - RR messages to BSC
  - MM/CC messages to MSC
  - SMS messages to MSC/SMSC
- Adding frequency hopping support not very hard

Summary
What we've learned

- The GSM industry is making security analysis very difficult
- It is well-known that the security level of the GSM stacks is very low
- We now have multiple solutions for sending arbitrary protocol data
  - From a rogue network to phones (OpenBSC, OpenBTS)
  - From an A-bis proxy to the network or the phones
  - From custom GSM phone baseband firmware to the network

TODO
Where we go from here

- The basic tools for fuzzing mobile networks are available
- No nice interface/integration from OsmocomBB to scapy yet
- It is up to the security community to make use of those tools (!)
- Don't you too think that TCP/IP security is boring
- Join the GSM protocol security research projects
- Boldly go where no man has gone before

Further Reading

- http://laforge.gnumonks.org/papers/gsm_phone-anatomy-latest.pdf
- http://bb.osmocom.org/
- http://openbsc.gnumonks.org/
- http://openbts.sourceforge.net/
- http://airprobe.org/