

# OsmocomBB

Running your own GSM stack on a phone

Harald Welte and Steve Markgraf

<http://bb.osmocom.org/>

27th CCC Congress, December 2010, Berlin/Germany

# Outline

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

# 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

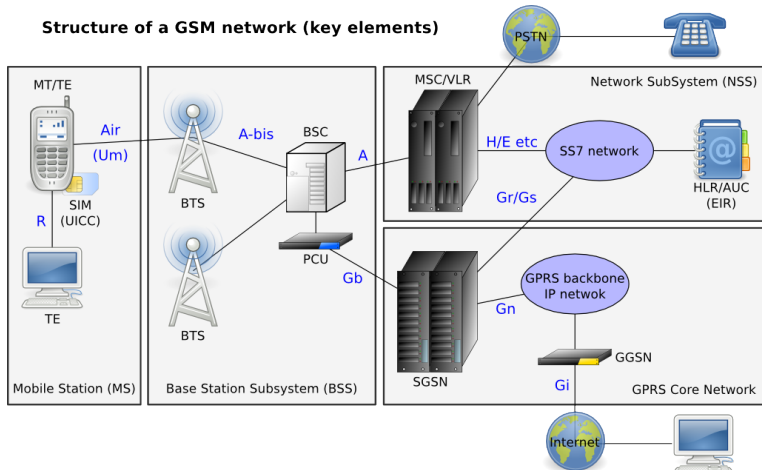
## The bootstrapping process

- 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

Source: Wikipedia, User Tsaitgaist, Licensed under GPLv3

**Structure of a GSM network (key elements)**



# 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

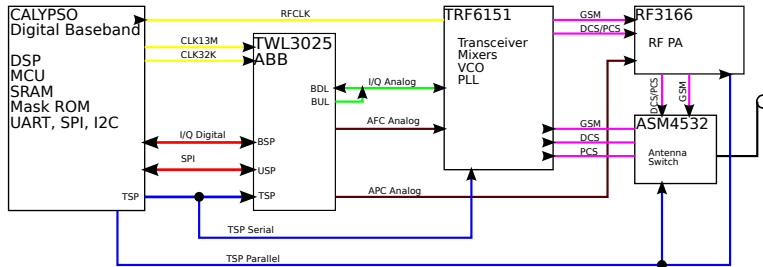
# Known GSM security problems

## 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](http://laforge.gnumonks.org/papers/gsm_phone-anatomy-latest.pdf)

# Requirements for GSM security analysis

What do we need for protocol-level security analysis?

- A GSM MS-side baseband chipset under our control
- A Layer1 that we can use to generate arbitrary L1 frames
- A Layer2 protocol implementation that we can use + modify
- A Layer3 protocol implementation that we can use + modify

None of those components existed, so we need to create them!

# A GSM baseband under our control

The two different DIY approaches

- Build something using generic components (DSP, CPU, ADC, FPGA)
  - No reverse engineering required
  - A lot of work in hardware design + debugging
  - Hardware will be low-quantity and thus expensive
- Build something using existing baseband chipset
  - Reverse engineering or leaked documents required
  - Less work on the 'Layer 0'
  - Still, custom hardware in low quantity

# A GSM baseband under our control

## Alternative 'lazy' approach

- Re-purpose existing mobile phone
  - Hardware is known to be working
  - No prototyping, hardware revisions, etc.
  - Reverse engineering required
  - Hardware drivers need to be written
  - But: More time to focus on the actual job: Protocol software
- Searching for suitable phones
  - As cheap as possible
  - Readily available: Many people can play with it
  - As old/simple as possible to keep complexity low
  - Baseband chipset with lots of leaked information

# Baseband chips with leaked information

- Texas Instruments Calypso
  - DBB Documentation on cryptome.org and other sites
  - ABB Documentation on Chinese phone developer websites
  - Source code of GSM stack / drivers was on sf.net (tsm30 project)
  - End of life, no new phones with Calypso since about 2008
  - No cryptographic checks in bootloader
- Mediatek MT622x chipsets
  - Lots of Documentation on Chinese sites
  - SDK with binary-only GSM stack libraries on Chinese sites
  - 95 million produced/sold in Q1/2010

Initial choice: TI Calypso (GSM stack source available)

# OsmocomBB Introduction

- Project was started only in January 2010 (9 months ago!)
- Implementing a GSM baseband software from scratch
- This includes
  - GSM MS-side protocol 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

# OsmocomBB 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 be moved to the phone

# OsmocomBB 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



# OsmocomBB 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

# OsmocomBB Host Software

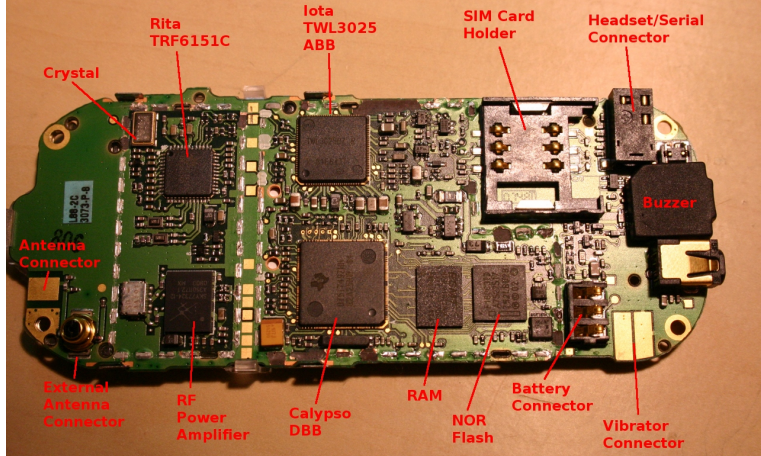
- Current working name: layer23
- Includes
  - Layer 1 Control (L1CTL) protocol API
  - GSM Layer2 implementation (LAPDm)
  - GSM Layer3 implementation (RR/MM/CC)
  - GSM Cell (re)selection
  - SIM Card emulation
  - Supports various 'apps' depending on purpose

# OsmocomBB Supported Hardware

- Baseband Chipsets
  - TI Calypso/Iota/Rita
  - Some early research being done 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

Printed Circuit Board of a Compal/Motorola C123



# OsmocomBB Project Status: Working

- Hardware Drivers for Calypso/Iota/Rita very complete
- Drivers for Audio/Voice signal path
- Layer1
  - Power measurements
  - Carrier/bit/TDMA synchronization
  - Receive and transmit of normal bursts on SDCCH
  - Transmit of RACH bursts
  - Automatic Rx gain control (AGC)
  - Frequency Hopping
- Layer2 UI/SABM/UA frames and ABM mode
- Layer3 Messages for RR / MM / CC
- Cell (re)selection according GSM 03.22

## OsmocomBB Project Status: Working (2/2)

OsmocomBB can now do GSM Voice calls (08/2010)

- Very Early Assignment + Late Assignment
- A3/A8 Authentication of SIM
- A5/1 + A5/2 Encryption
- Full Rate (FR) and Enhanced Full Rate (EFR) codec

# OsmocomBB Project Status: Not working

- Layer1
  - Neighbor Cell Measurements
  - In-call hand-over to other cells
- Actual UI on the phone
- Circuit Switched Data (CSD) calls
- GPRS (packet data)
- No Type Approval for the stack!

# OsmocomBB Project Status: Executive Summary

- We can establish control/signalling channels to both hopping and non-hopping GSM cells
  - Control over synthesizer means we can even go to GSM-R band
- We can send arbitrary data on those control channels
  - RR messages to BSC
  - MM/CC messages to MSC
  - SMS messages to MSC/SMSC
- TCH (Traffic Channel) support for voice calls
  - Dieter Spaar and Andreas Eversberg have made multiple 20 minute call with current master branch
  - Some people have tried alpha code on real networks for real 30+ minute calls!



# The mobile app

- implementation of a mobile phone
  - cell (re)selection, mobility management
  - voice calls (only full rate)
  - SMS
- both mobile originated and mobile terminated calls work
- VTY (telnet) interface to configure and call control
- optional interface to linux call router PBX

# cell\_log

## The `cell_log` app

- scanning and logging application for cell beacon information
- send RACH to all cells to get the timing advance (distance)
- logs the GPS position of where the cell was found

# gsmmap

## The `gsmmap` app

- parses the logs generated by `cell_log`
- uses triangulation to calculate estimated cell position
- exports a `.kml` file for Google Earth

# bcch\_scan

## The `bcch_scan` app

- iterates over full spectrum and does power scan
- tunes to ARFCN in order of received signal strength
- acquires BCCHs and dumps all SYSTEM INFO to wireshark

# cbch\_sniff

The `cbch_sniff` app

- dumps cell broadcast messages to wireshark
- some operators include GPS location of cell inside CB

There are some more apps, mostly R&D related.

We are looking forward to **your contribution**, e.g. the *scapy fuzzing gateway app*.

# 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

# Thanks

I would like to express my thanks to

- The OsmocomBB development team, most notably
  - Dieter Spaar (invaluable dedication to this project!)
  - Andreas Eversberg (layer 3, cell selection, etc.)
  - Sylvain Munaut (layer1, dsp, misc.)
- Other developers working on Open Source GSM stuff
  - g3gg0 (MADos)
  - David Burgess, Harvind Simra (OpenBTS)
  - Holger Freyther (OpenBSC)



# Further Reading

- [http://laforge.gnumonks.org/papers/gsm\\_phone-anatomy-latest.pdf](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/>