Free Software GSM protocol stacks

OpenBSC, OsmoSGSN, OpenGGSN, OsmocomBB

Harald Welte

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

ELCE 2010, October 2010, Cambridge/UK

Outline

1. GSM/3G security
   - The closed GSM industry
   - Security implications
   - The GSM network
   - The GSM protocols
2. OpenBSC
   - OpenBSC Introduction
   - OpenBSC Network In The Box
   - OpenBSC BSC-only mode
   - OpenBSC GPRS support
3. OsmocomBB Project
   - OsmocomBB Introduction
   - OsmocomBB Software
   - OsmocomBB Hardware Support
   - OsmocomBB Project Status

About the speaker

- Using + playing with Linux since 1994
- Kernel / bootloader / driver / firmware development since 1999
- IT security expert, focus on network protocol security
- Core developer of Linux packet filter netfilter/iptables
- 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
  - Network planning / deployment / servicing
  - Even Billing!
- 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

GSM is more than phone calls

Listening to phone calls is boring...

- Machine-to-Machine (M2M) communication
  - BMW can unlock/open your car via GSM
  - Alarm systems often report via GSM
  - Smart Metering (Utility companies)
  - GSM-R / European Train Control System
  - Vending machines report that their cash box is full
  - Control if wind-mills supply power into the grid
  - Transaction numbers for electronic banking

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

The closed GSM industry

My self-proclaimed mission

Mission: Bring TCP/IP/Internet security knowledge to GSM

- Create tools to enable independent/public IT Security community to examine GSM
- Try to close the estimated 10 year gap between the state of security technology on the Internet vs. GSM networks
  - Industry thinks in terms of *walled garden* and *phones behaving like specified*
  - No proper incident response strategies!
  - No packet filters, firewalls, intrusion detection on GSM protocol level
  - General public assumes GSM networks are safer than Internet

The closed GSM industry

Areas of interest for Security research

- Specification problems
  - Encryption optional, weak and only on the Um interface
  - Lack of mutual authentication
  - Silent calls for pin-pointing a phone
  - RRLP and SUPL to obtain GPS coordinates of phone
- Implementation problems
  - TMSI information leak on network change
  - TLV parsers that have never seen invalid packets
  - Obscure options in spec lead to rarely-tested/used code paths
- Operation problems
  - VLR overflow leading to paging-by-IMSI
  - TMSI re-allocation too infrequent
  - Networks/Cells without frequency hopping

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 BTS equipment is available, much easier/faster progress
- Result: Started project OpenBSC in 10/2008

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?

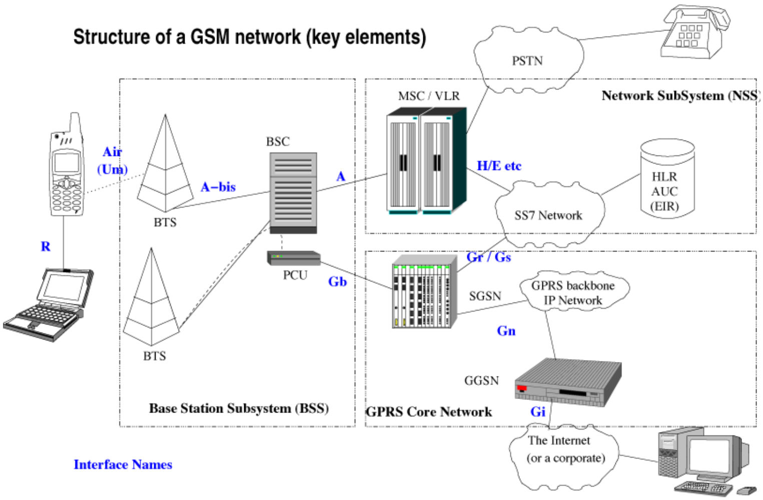
- On the handset side?
    - Difficult since GSM firmware and protocol stacks are closed and proprietary
    - Even if you want to write your own protocol stack, the layer 1 hardware and signal processing is closed and undocumented, too
    - Publicly known attempts (12/2009)
        - The TSM30 project as part of the THC GSM project
        - mados, an alternative OS for Nokia DTC3 phones
    - none of those projects have been successful
    - Result: Started project OsmocomBB in 01/2010

Security analysis of GSM

The bootstrapping process

- Start to read GSM specs (> 1000 PDF documents)
- Gradually grow knowledge about the protocols
- Obtain actual GSM network equipment (BTS)
- 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, ...

GSM network protocols

On the A-bis interface

- Layer 1: Typically E1 line, TS 08.54
- Layer 2: A variant of ISDN LAPD with fixed TEI's, TS 08.56
- Layer 3: OML (Organization and Maintenance Layer, TS 12.21)
- Layer 3: RSL (Radio Signalling Link, TS 08.58)
- Layer 4+: transparent messages that are sent to the MS via Um

OpenBSC software

OpenBSC is a Open Source implementation of (not only) the BSC features of a GSM network.

- Support A-bis interface over E1 and IP
- Support for BTS vendor/model is modular, currently Siemens BS-11 and ip.access nanoBTS
- Multiple BTS models/vendorrs can be mixed!
- Can work as a *pure BSC* or as a full *network in a box*
- Supports mobility management, authentication, intra-BSC hand-over, SMS, voice calls (FR/EFR/AMR)
- GPRS + EDGE support if combined with OsmoSGSN and OpenGGSN

OpenBSC

- Supports Siemens BS-11 BTS (E1) and ip.access nanoBTS (IP based)
- Has classic 2G signalling, voice and SMS support
- Implements various GSM protocols like
  - A-bis RSL (TS 08.58) and OML (TS 12.21)
  - TS 04.08 Radio Resource, Mobility Management, Call Control
  - TS 04.11 Short Message Service
- Telnet console with Cisco-style interface

OpenBSC software architecture

- Implemented in pure C, similarities to Linux kernel
  - Linked List handling, Timer API, coding style
- Single-threaded event-loop / state machine design
- Telnet based command line interface *Cisco-style*
- Input driver abstraction (mISDN, Abis-over-IP)

OpenBSC: GSM network protocols

The A-bis interface

Layer 1 Typically E1 line, TS 08.54

Layer 2 A variant of ISDN LAPD with fixed TEI's, TS 08.56

Layer 3 OML (Organization and Maintenance Layer, TS 12.21)

Layer 3 RSL (Radio Signalling Link, TS 08.58)

Layer 4+ transparent messages that are sent to the MS via Um

OpenBSC: How it all started

- In 2006, I bought a Siemens BS-11 microBTS on eBay
  - This is GSM900 BTS with 2 TRX at 2W output power (each)
  - A 48kg monster with attached antenna
  - 200W power consumption, passive cooling
  - E1 physical interface
- I didn't have much time at the time (day job at Openmoko)
- Started to read up on GSM specs whenever I could
- Bought a HFC-E1 based PCI E1 controller, has mISDN kernel support
- Found somebody in the GSM industry who provided protocol traces

OpenBSC: Timeline

- November 2008: I started the development of OpenBSC
- December 2008: we did a first demo at 25C3
- January 2009: we had full voice call support
- Q1/2009: Add support for ip.access nanoBTS
- June 2009: I started with actual security related stuff
- August 2009: We had the first field test with 2BTS and > 860 phones
- Q1/2010: The first 25 OpenBSC instances running in a commercial network

OpenBSC: Field Test at HAR2009

OpenBSC in NITB mode

Network In a Box Mode

The `bsc_hack` program

- implements the A-bis interface towards any number of BTS
- provides most typical features of a GSM network in one software
- no need for MSC, AuC, HLR, VLR, EIR, ...
    - HLR/VLR as SQLite3 table
    - Authentication + Ciphering support
    - GSM voice calls, MO/MT SMS
    - Hand-over between all BTS
    - Multiple Location Areas within one BSC

OpenBSC NITB features

OpenBSC NITB features

- Run a small GSM network with 1-n BTS and OpenBSC
- No need for MSC/HLR/AUC/...
- No need for your own SIM cards (unless crypto/auth rqd)
- Establish signalling and voice channels
- Make incoming and outgoing voice calls between phones
- Send/receive SMS between phones
- Connect to ISDN PBX or public ISDN via Linux Call Router

OpenBSC in NITB mode

Network In a Box Mode

The `bsc_hack` program

- does not implement any other GSM interfaces apart from A-bis
- no SS7 / TCAP / MAP based protocols
- no integration (roaming) with existing traditional GSM networks
- wired telephony interfacing with ISDN PBX `lcr` (Linux Call Router)
- Has been tested with up to 800 subscribers on 5 BTS
- Intended for R&D use or private PBX systems

OpenBSC LCR integration

Interfacing with wired telephony

OpenBSC (NITB mode) can be linked into Linux Call Router (`lcr`)

- OpenBSC is compiled as libbsc.a
- libbsc.a includes full OpenBSC NITB mod code
- linking the library into `lcr` results in GSM *line interfaces* to become available inside `lcr`
- OpenBSC no longer takes care of call control, but simply hands everything off to `lcr`
- Dialling plan, etc. is now configure in `lcr` like for any other wired phones

OpenBSC in BSC-only mode

The `osmo-bsc` program

- behaves like a classic GSM BSC
- uses SCCP-Lite (ip.access multipex) to any SoftMSC like ADC
- used in production/commercial deployments (75 BSCs)
- mainly intended to replace proprietary BSC in traditional GSM networks

GPRS and OpenBSC

- The BSC doesn’t really do anything related to GPRS
- GPRS implemented in separate SGSN and GGSN nodes
- GPRS uses its own Gb interface to RAN, independent of A-bis
- OpenBSC can configure the nanoBTS for GPRS+EDGE support via OML
- Actual SGSN and GGSN implemented as OsmoSGSN and OpenGGSN programs

OsmoSGSN

The Osmocom SGSN program implements

- basic/minimal SGSN functionality
- the Gb interface (NS/BSSGP/LLC/SNDCP)
- mobility management, session management

It’s a work in progress, many missing features

- no HLR integration yet
- no paging coordination with MSC/BSC
- no encryption support yet

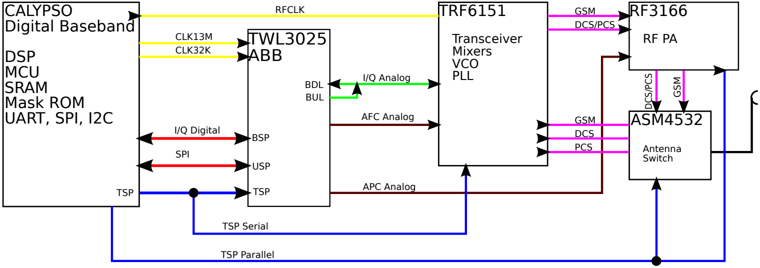
OpenGGSN

- GPL licensed Linux program implementing GGSN node
- Implements GTP-U protocol between SGSN and GGSN
- User-configurable range/pool of IPv4 addresses for MS
- Uses `tun` device for terminating IP tunnel from MS
- provides GTP implementation as libgtp
- Experimental patches for IPv6 support

A GSM phone baseband processor

- 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

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 me 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

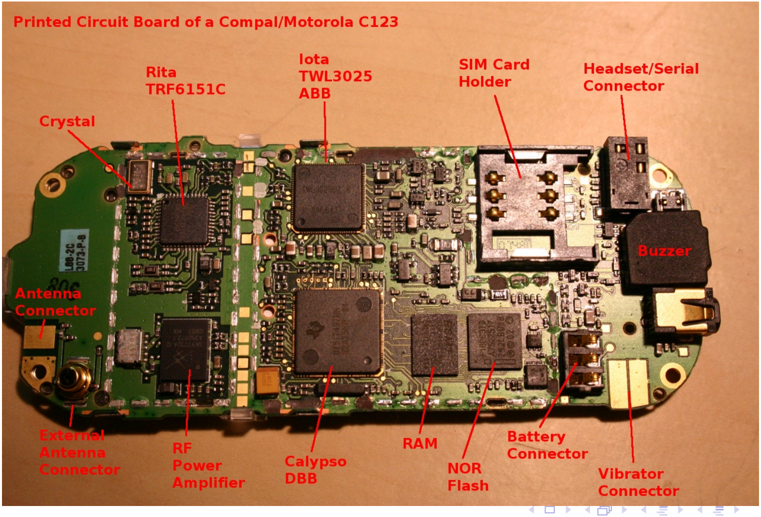
OsmocomBB Host Software

- 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



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

- Fully-fledged SIM card reader inside phone (WIP)
- Layer1
  - Automatic Tx power control (APC)
  - 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!

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)
  - Frem a FOSS controlled phone to the network (OsmocomBB)
  - From an A-bis proxy to the network or the phones

TODO

Where we go from here

- The tools for fuzzing mobile phone protocol stacks are available
- 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

Current Areas of Work / Future plans

- UMTS(3G) support for NodeB and femtocells
- SS7 / MAP integration
- Playing with SIM Toolkit from the operator side
- Playing with MMS
- More exploration of RRLP + SUPL

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/