Fuzzing your GSM phone

using OpenBSC and scapy

Harald Welte

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

26C3 conference, December 2009, Berlin/Germany

Outline

1. GSM/3G security
   - The closed GSM industry
   - Security implications
   - The GSM network
   - The GSM protocols
2. Implementing GSM protocols
   - Getting started
   - Timeline
   - OpenBSC
3. Security analysis
   - Theory
   - The Baseband
   - Observations
   - GSM Protocol Fuzzing
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 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

How would you get started?

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

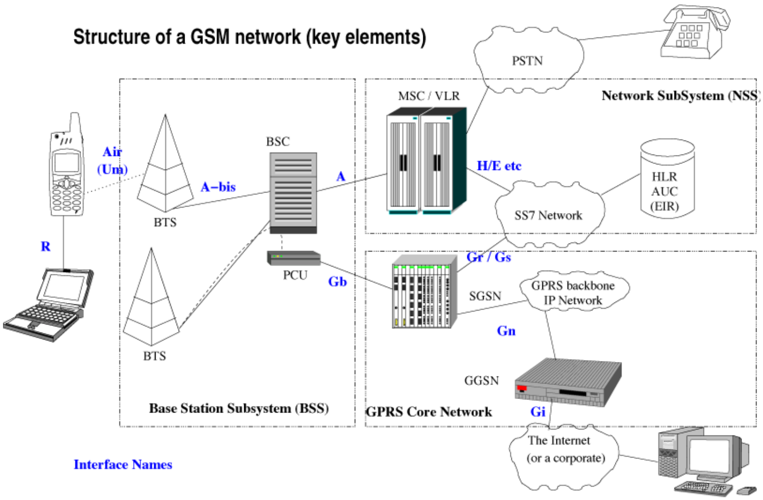
- On the network side?
  - Difficult since equipment is not easily available and normally extremely expensive
  - However, network is very modular and has many standardized/documented interfaces
  - Thus, if equipment is available, much easier/faster progress

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)
- 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

Implementing GSM protocols

How I got started!

- In September 2008, we were first able to make the BTS active and see it on a phone
  - 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

Implementing GSM protocols

Timeline

- In November 2008, I started the development of OpenBSC
- In December 2008, we did a first demo at 25C3
- In January 2009, we had full voice call support
- In June 2009, I started with actual security related stuff
- In August 2009, we had the first field test with 2BTS and > 860 phones

Security analysis of GSM
OpenBSC

What is OpenBSC

- A *GSM network in a box* software
- Implements minimal subset of BSC, MSC, HLR, SMSC
- Is Free and Open Source Software licensed under GNU GPL
- 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

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, ..)

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

A-bis injection
for A-bis over IP

How to do inject messages into A-bis over IP?

- Problem
  - A-bis/IP uses one TCP connection for OML and RSL messages
  - OML initialization is essential for BTS to become operational
  - TCP makes insertion of additional messages relatively hard
- Solution: Build an *A-bis injection proxy*
  - Transparently pass OML and RSL packets between BTS and BSC
  - Add additional stateless UDP sockets for injecting messages, one socket each for
  - injecting OML/RSL to the network
  - injecting OML/RSL to the BTS

A-bis Injection Proxy

Principle of operation

- Proxy needs to be brought between BTS and BSC
- Luckily, A-bis/IP SSL support not always used
- Thus, physical access to the Ethernet link sufficient
- Configure system with two interfaces
  - BSC-facing interface has IP of BTS
  - BTS-facing interface has IP of BSC / default gw
- BTS will make TCP connection to proxy
- proxy will make independent TCP connection to BSC

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 hardcoded 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

OpenBSC silent calls

A more elegant fuzzing interface

- Injection at the A-bis level has many problems
  - you can only do it while a call is active
  - you simply piggy-back on existing RR connections
- The OpenBSC *silent call* feature can help
  - we use OpenBSC to establish a RR connection
  - in the GSM master/slave model, the phone will not close a connection unless told to do so
  - we then send arbitrary data to the phone and receive its responses
  - this currently only works from within OpenBSC, but we'll provide UDP injection sockets soon

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
- There is ongoing work for a phone-based tool to fuzz the network

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

Future plans

- Packet data (GPRS/EDGE) support in OpenBSC
  - GPRS is used extensively on modern smartphones
  - Enables us to play with those phones without a heavily filtered operator network
- UMTS(3G) support in OpenBSC
- Access to MS side layer 1
- Playing with SIM Toolkit from the operator side
- Playing with MMS
- More exploration of RRLP

Further Reading

- http://openbsc.gnumonks.org/
- http://airprobe.org/
- http://openbts.sourceforge.net/