Opening Closed Domains

To boldly go where no Free Software has gone before

Harald Welte

gnumonks.org
gpl-violations.org
hmw-consulting.de
OpenPCD, OpenMoko, OpenBSC, airprobe, OsmocomBB

FOI, May 2010, Varazdin/Croatia

Outline

1. The FOSS success story
    - FOSS is successful
    - Why no FOSS in RFID, DECT and GSM
2. Case studies
    - RFID
    - DECT
    - GSM
3. Summary
    - What we’ve learned
    - Get involved!
    - Knowledge is power

About the speaker

- Using + playing with Linux since 1994
- Kernel / bootloader / driver / firmware development since 1999
- IT security specialist, focus on network protocol security
- Some board-level Electrical Engineering
- Expert on Free Software licensing (gpl-violations.org)

FOSS is successful

Free and Open Source Software is successful

- As a general purpose Operating System
- In the Internet and Intranet server market, data centers
- Networking infrastructure (routers, switches, WiFi AP)
- Workstation / Desktop Market
- Now Netbooks and MID's

Common denominator: (Inter)networked device

Linux/FOSS and the Internet

- Imagine the state of the Linux kernel or other FOSS community without the Internet
- Imagine working on a FOSS project without Internet access
- Imagine even using a typical Linux system without Internet access (apt-get / yum / ...)

Thus, the current state of FOSS cannot be explained without the success of the Internet as communication and distribution medium.

Why FOSS?

- Education
  - Learning by reading the source code
- Security
  - Review the source for back-doors, exploitable bugs, ...
- Modification
  - Experiments, Rapid prototyping, test of new ideas
- Research possible for anyone
- No vendor lock-in

Why FOSS?

- Education
  - Learning by reading the source code
- Security
  - Review the source for back-doors, exploitable bugs, ...
- Modification
  - Experiments, Rapid prototyping, test of new ideas
- Research possible for anyone
- No vendor lock-in

Why FOSS?

- Education
  - Learning by reading the source code
- Security
  - Review the source for back-doors, exploitable bugs, ...
- Modification
  - Experiments, Rapid prototyping, test of new ideas
- Research possible for anyone
- No vendor lock-in

Why FOSS?

- Education
  - Learning by reading the source code
- Security
  - Review the source for back-doors, exploitable bugs, ...
- Modification
  - Experiments, Rapid prototyping, test of new ideas
- Research possible for anyone
- No vendor lock-in

Why FOSS?

- Education
  - Learning by reading the source code
- Security
  - Review the source for back-doors, exploitable bugs, ...
- Modification
  - Experiments, Rapid prototyping, test of new ideas
- Research possible for anyone
- No vendor lock-in

So everything is great?

- We have multiple FOSS OS kernels
- We have thousands of FOSS application programs
- But: Many areas of computing typically have no FOSS at all, e.g.
  - RFID protocols
  - DECT protocols
  - GSM/UMTS/CDMA protocols
- Why those three examples?
  - communications infrastructure is vital for everyone
  - big interest in verifiable/auditable privacy and security

Why do you think there no FOSS?

In RFID DECT and GSM

- No Open specifications for the protocols?
  - WRONG: They're all publicly available to anyone
- Too close to the hardware for FOSS developers
  - WRONG: Look at FOSS softmac WiFi drivers
- Hardware comes without specs
  - TRUE, but look at reverse engineered 3D drivers
- Too complex for FOSS developers
  - WRONG, look at image processing or a TCP/IP protocol stack

Why do you think there no FOSS?

In RFID DECT and GSM

- No Open specifications for the protocols?
  - WRONG: They're all publicly available to anyone
- Too close to the hardware for FOSS developers
  - WRONG: Look at FOSS softmac WiFi drivers
- Hardware comes without specs
  - TRUE, but look at reverse engineered 3D drivers
- Too complex for FOSS developers
  - WRONG, look at image processing or a TCP/IP protocol stack

Why do you think there no FOSS?

In RFID DECT and GSM

- No Open specifications for the protocols?
  - WRONG: They're all publicly available to anyone
- Too close to the hardware for FOSS developers
  - WRONG: Look at FOSS softmac WiFi drivers
- Hardware comes without specs
  - TRUE, but look at reverse engineered 3D drivers
- Too complex for FOSS developers
  - WRONG, look at image processing or a TCP/IP protocol stack

Why do you think there no FOSS?

In RFID DECT and GSM

- No Open specifications for the protocols?
  - WRONG: They're all publicly available to anyone
- Too close to the hardware for FOSS developers
  - WRONG: Look at FOSS softmac WiFi drivers
- Hardware comes without specs
  - TRUE, but look at reverse engineered 3D drivers
- Too complex for FOSS developers
  - WRONG, look at image processing or a TCP/IP protocol stack

Why do you think there no FOSS?

In RFID DECT and GSM

- No Open specifications for the protocols?
  - WRONG: They're all publicly available to anyone
- Too close to the hardware for FOSS developers
  - WRONG: Look at FOSS softmac WiFi drivers
- Hardware comes without specs
  - TRUE, but look at reverse engineered 3D drivers
- Too complex for FOSS developers
  - WRONG, look at image processing or a TCP/IP protocol stack

Why do you think there no FOSS?

In RFID DECT and GSM

- No Open specifications for the protocols?
  - WRONG: They're all publicly available to anyone
- Too close to the hardware for FOSS developers
  - WRONG: Look at FOSS softmac WiFi drivers
- Hardware comes without specs
  - TRUE, but look at reverse engineered 3D drivers
- Too complex for FOSS developers
  - WRONG, look at image processing or a TCP/IP protocol stack

Why do you think there no FOSS?

In RFID DECT and GSM

- No Open specifications for the protocols?
    - WRONG: They're all publicly available to anyone
- Too close to the hardware for FOSS developers
    - WRONG: Look at FOSS softmac WiFi drivers
- Hardware comes without specs
    - TRUE, but look at reverse engineered 3D drivers
- Too complex for FOSS developers
    - WRONG, look at image processing or a TCP/IP protocol stack

Why do you think there no FOSS?

In RFID DECT and GSM

- No Open specifications for the protocols?
  - WRONG: They're all publicly available to anyone
- Too close to the hardware for FOSS developers
  - WRONG: Look at FOSS softmac WiFi drivers
- Hardware comes without specs
  - TRUE, but look at reverse engineered 3D drivers
- Too complex for FOSS developers
  - WRONG, look at image processing or a TCP/IP protocol stack

Why there really is no FOSS?

In RFID DECT and GSM

- RFID, DECT and GSM are hardware industry dominated areas
    - If you think the software industry took 10-15 years to realize the benefits of FOSS, try to imagine how long it will take the hardware and semiconductor industry
- Semiconductor industry actively keeping customers out and ensure their own turf is as big as possible
- Chicken-and-egg Problem
    - Too little people know about the protocol
    - Too little people are able to work on FOSS for it
    - Too little FOSS in that area
    - Nobody can experiment with / learn from FOSS -> Too little people know about the protocol

Why did I care?

- because I use cordless phones, cell-phones and RFID
- because I know and am used to the benefits of FOSS for 15 years
- because I don't see why I should give up that freedom when leaving the PC world

What could be done about that?

- Tackle those cases, one by one
- Even a single person or a small group of people can make a huge difference
- None of the projects that were started to change this involved more than a handful of people
- There are no limits other than those of your mind!

Why my interest in RFID?

- In November 2005, the German federal government has started to issue electronic passports with RFID interface.
- All other EU member states were mandated to start issuing such passports no later than January 2007
- Passports with RF interface raise lots of security questions
  - Can you track/identify a person?
  - Can you determine his nationality?
  - Is the crypto used sufficient to prevent forgeries?
- Thus, in 2005 I realized: FOSS for RFID protocol + ePassport application is needed

RFID products

commercially available readers

- There are many RFID readers, some even with Linux drivers
- All the protocol logic typically happens inside reader firmware
- Application programmer presented with high-level API (PC/SC)
- No way of doing actual practical protocol-level security research

RFID products
Semiconductors

- Inside a reader, you need to somehow implement the radio layer
- Typically, for mass-produced products, you use integrated circuits
- The early RFID ASICs were dumb transceivers with no protocol logic
- ASIC Documentation for NXP RC5xx *almost* openly available as 40-bit *encrypted* PDF
- Some readers export those ASIC registers to PC software and run protocol stack in proprietary software

Project librfid

- Project librfid was born
- The first FOSS protocol stack for RFID protocols
  - ISO 14443-1,2,3,4 (specification public)
  - ISO 15693 (specification public)
  - Mifare Classic (vendor-specific proprietary system)
- Mainly written by one person!

More FOSS for RFID
OpenPCD

OpenPCD

- Project OpenPCD developed an open hardware RFID reader
- FOSS firmware, cross-compiled with avr-gcc, installed with FOSS flashing tools
- librfid used as protocol stack either inside reader firmware or on host PC
- R&D done by three people!

OpenPICC

- Project OpenPICC developed an open hardware RFID simulator
- R&D by three people in their spare time!

More FOSS for RFID

OpenPCD

- Various groups started RFID security research on proprietary RFID standards
    - Wrong security claims by RFID industry could be revealed as part of the Mifare Classic CRYPTO-1 reverse engineering in 2006
    - Industry was forced to introduce and use components with higher security
    - Still, e-payment and access control systems all over the world rely on broken-by-design, proprietary and small-keysize Mifare Classic CRYPTO1. They deserve to be hacked, and the responsible IT managers deserve to be fired.
    - Watch 26C3 in four weeks: Yet another proprietary insecure system will die.

DECT systems
What they are

- DECT means Digital European Cordless Telephony
- Is a standard used in large parts of the world for cordless telephony
- Standardized at the ETSI, documents available to everyone
- Authentication and Encryption algorithms kept secret
- Typically, all you can buy is a black-box appliance consisting of phone + base station

DECT beyond just simple cordless phones

- DECT always supported the transmission of data, e.g. ISDN payloads
- At some point, people wanted to use their DECT handset with VoIP
- Companies started to sell PCMCIA, USB and PCI DECT adapters
- DECT protocol stack is running inside proprietary windows software
- DECT encryption implemented inside the DECT chipset silicon

First FOSS for DECT

- In order to do research on DECT, Open Source is needed
- Group (later called deDECTed.org) formed doing reverse engineering of windows drivers
- Started to implement FOSS driver for the hardware and Receive-only processing of DECT protocol stack
- Was used by cryptographers as a toolbox to demonstrate that the DECT Security is not worth the paper it is printed on
    - Problems in the protocol specification
    - Problems in the implementations (16bit random numbers?!?)

More FOSS for DECT

- Patrick McHardy (netfilter/iptables maintainer) starts a Linux kernel DECT stack
- His experience in Linux kernel networking enables him to write a true Linux network stack for the DECT protocol
- git://git.kernel.org/pub/scm/linux/kernel/git/kaber/dect-2.6.git
- Written by one person!

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

OpenBSC

A FOSS GSM protocol stack implementation

- OpenBSC implements *GSM network in a box*
  - BSC, MSC, HLR, AuC, ...
  - You can use a BTS + OpenBSC and run your own network
- Commercial systems with same functionality sell for > 50,000 USD
  - Did you also hear the rumors about 100-man-years intellectual property in a GSM stack?
- Developed by three people in their spare time!

OpenBTS

A different GSM protocol stack implementation

- Open implementation of Um L1 & L2, an all-software BTS.
- L1/L2 design based on an object-oriented dataflow approach.
- Includes L3 RR functions normally found in BSC.
- Uses SIP PBX for MM and CC functions, eliminating the conventional GSM network. L3 is like an ISDN/SIP gateway.
- Intended for use in low-cost and rapidly-deployed communications networks, but can be used for experiments.
- Written by two people!

airprobe

A GSM receiver / protocol analyzer

- *airprobe* is a collection of Um protocol analyzer tools using the USRP software defined radio
- A number of different Um receiver implementations

  gssm One of the two early Um receiver implementations (M&M clock recovery)

  gsmsp The other early Um receiver implementation

  gsm-tvoid For a long time the Um receiver with best performance

  gsm-receiver The latest generation of Um receiver
- Today, gsm-receiver seems to be the most popular choice

OsmocomBB

A telephone-side GSM baseband firmware

- *OsmocomBB* is a FOSS implementation of GSM protocols for the mobile phone
- Implemented in portable C with drivers for one TI baseband chipset
- Current features include
  - Passive protocol-analysis and protocol sniffing
  - Determine operating parameters of any GSM network, including its cell configuration, use of encryption, use of TMSI, etc.
  - Transmit arbitrary data to any GMS cell/network
- Will serve as foundation for security analysis tools for the GSM air interface

What has GSM FOSS shown?

- We can now do real-world demonstration of GSM weaknesses
  - Lack of mutual authentication
  - Silent calls to turn phones into permanently transmitting beacons
  - RRLP to inquire the GPS fix of any phone
- We can make people aware of the dangers those features have
- Hopefully, public pressure will force the industry to change

Summary

What we've learned

- There are many areas which the benefits of FOSS have not yet reached
- Nonetheless, those systems are used by billions of people every day
- Without independent research, security flaws will never be removed
- Small projects with few dedicated developers can make a huge impact

Get involved!

- Boldly go where no (FOSS) man has gone before
- Don't be deterred by people who say it's impossible
- What are you waiting for?
- Would you rather become the worlds 10,000th application security expert or the worlds 100th GSM protocol security expert?
- Help us to get some sense into the semiconductor industry.

Knowledge is power

- Educate yourself
- Never underestimate *learning by doing*
- Educate others about
  - How every-day technology like cellphones really work
  - What is the true security and privacy level of those systems

Where to go?

Areas that need openness

- 3G (UMTS) protocol stack
- GSM telephone side GSM stack
- What about DAB, DMB-T
- DVB-S encapsulated Internet downlink
- TETRA being deployed in multiple European countries
- CDMA / CDMA2000
- ... and this is just in the communications protocol sector!

Where to go?

Areas that need openness

- 3G (UMTS) protocol stack
- GSM telephone side GSM stack
- What about DAB, DMB-T
- DVB-S encapsulated Internet downlink
- TETRA being deployed in multiple European countries
- CDMA / CDMA2000
- ... and this is just in the communications protocol sector!