Baseband exploitation in 2013: Hexagon challenges

Ralf-Philipp Weinmann
<ralf@comsecuris.com>

Presented at Pacsec 2013
2013-11-13, Tokyo, Japan

Comsecuris

Who am I?

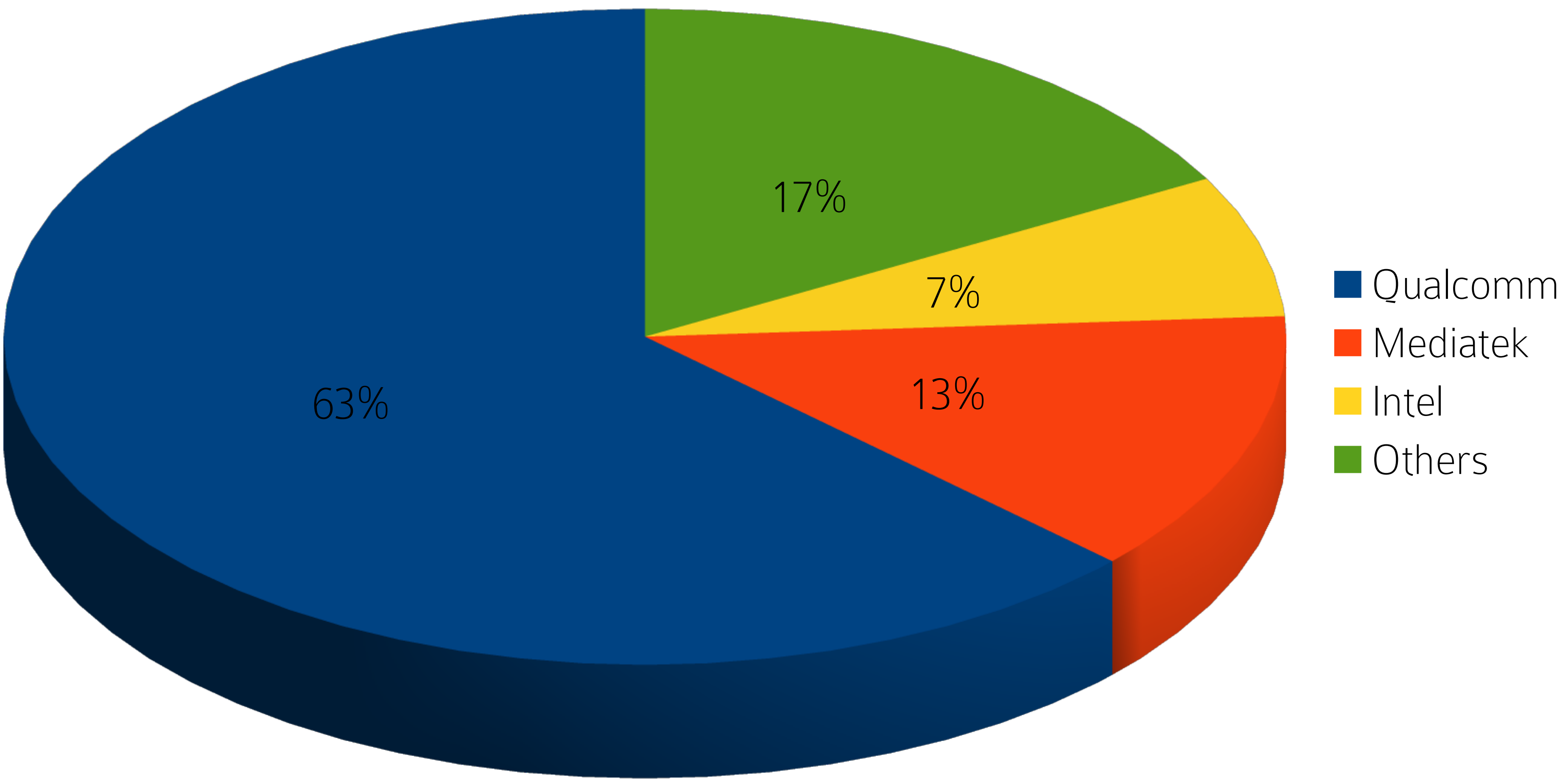
- Security researcher from Germany
- Previously in academia (University of Luxembourg)
- Now working for my own company
- Keen interest in security of mobile, wireless and embedded systems
- First to demonstrate remotely exploitable vulnerabilities in baseband stacks (3 years ago)

Overview

- Importance of Hexagon for mobile exploitation
- Intro to the QDSP6 architecture
- Past issues with BLAST
- On the complexity of ROP and similar techniques
- An example vulnerability
- Conclusions

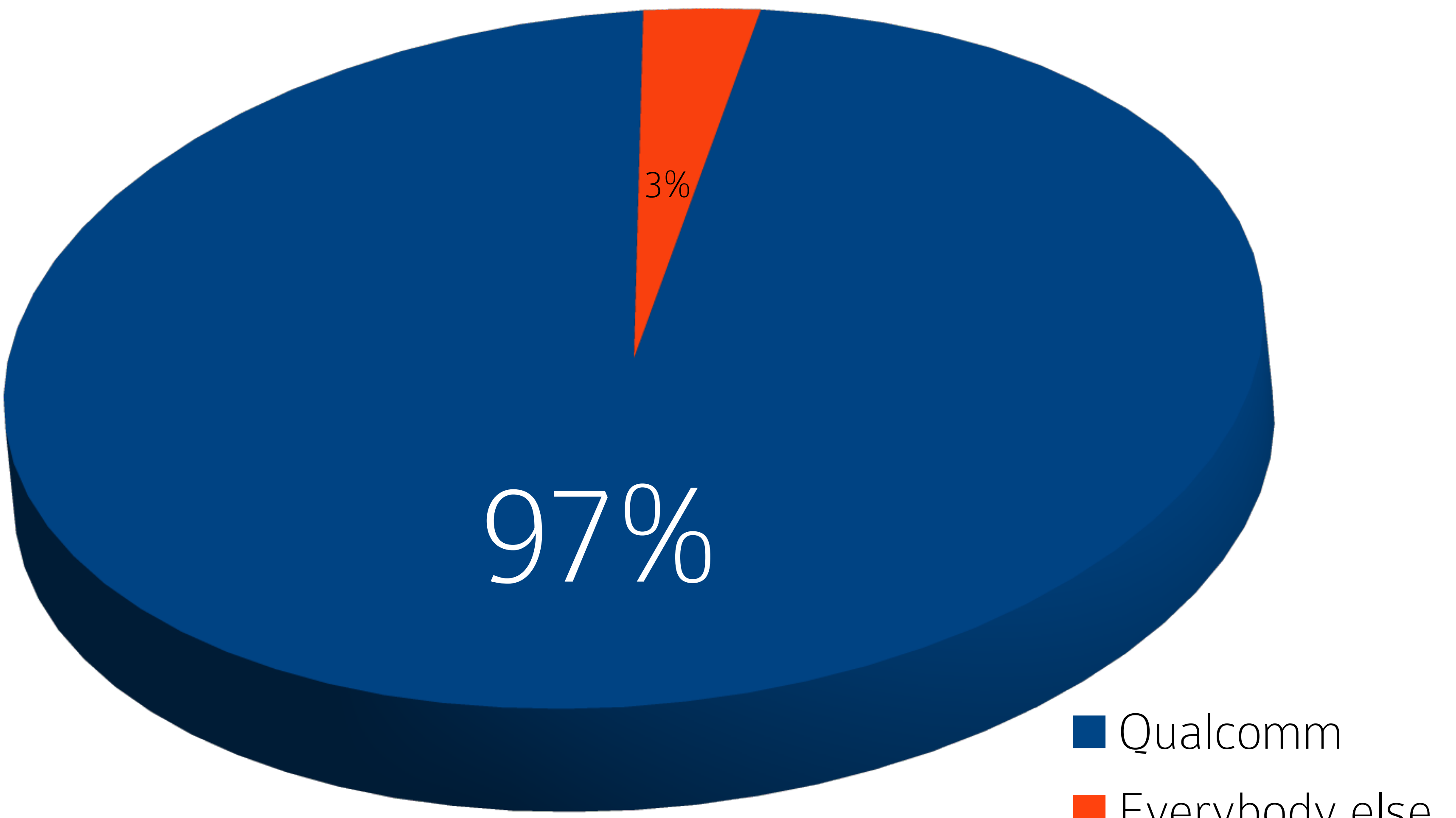
Comsecuris

The now: cellular baseband market 2013



Comsecuris

LTE: Baseband market share distribution



Hexagon architecture

- Originated from QCOMs general purpose DSP
  - Used for only audio processing and L1 in early days
- VLIW architecture [1-4 instructions per cycle]
- Barrel processor (interleaved multithreading)
- 32-bit unified address space for code and data
  - Byte addressable
- 32 General registers (32-bit)
  - also usable pairwise: 64-bit register pairs
- Supports nestable loops
- Many addressing modes (specific to DSP usage cases)
- Leaked docs claim up “3x fewer cycles than ARM9 on control code”

Instruction packets

- Atomic units grouping instructions executed in parallel
- 4 parallel pipelines (called *slots*)
- Different ins. types assigned to different slots
- Constraints for grouping apply
  - HW resources cannot be oversubscribed
- Manuals: no branching into middle of packet
  - Empirically: you can return into middle of packet

Chipset evolution

- QDSP6v1: MSM8600
  - Pantech Racer Vega (anyone?!?)
- QDSP6v2: QSD8650 (v1/v2), MSM8200 (v1/v2), CSM8900, MDM8900
  - e.g. Sharp IS03/IS05
- QDSP6v3: MDM900 (v1/v2), CSM8700, FSM9000, QSD8650a, MDM8200a, MSM8660, QSD8x72
  - e.g. Sony Xperia acro HD IS12S
- QDSP6v4: MSM8960, MDM9x15
  - e.g. Samsung Galaxy S4 (GT-i9505), Apple iPhone 5, BlackBerry Z10
- QDSP6v5: MSM8974
  - e.g. LG G2, Sony Xperia Z Ultra

Problems with ISA (revisions)

- Hexagon Programmer's guide only available for v2
- Architecture has significantly evolved since
- Many details guessed and deduced from toolchain
  - Example: immext (payload extender)
- Very hard to build tools from scratch because of sheer complexity of ISA
  - Testing?
  - Easier to start from publicly released toolchain

Useful instructions

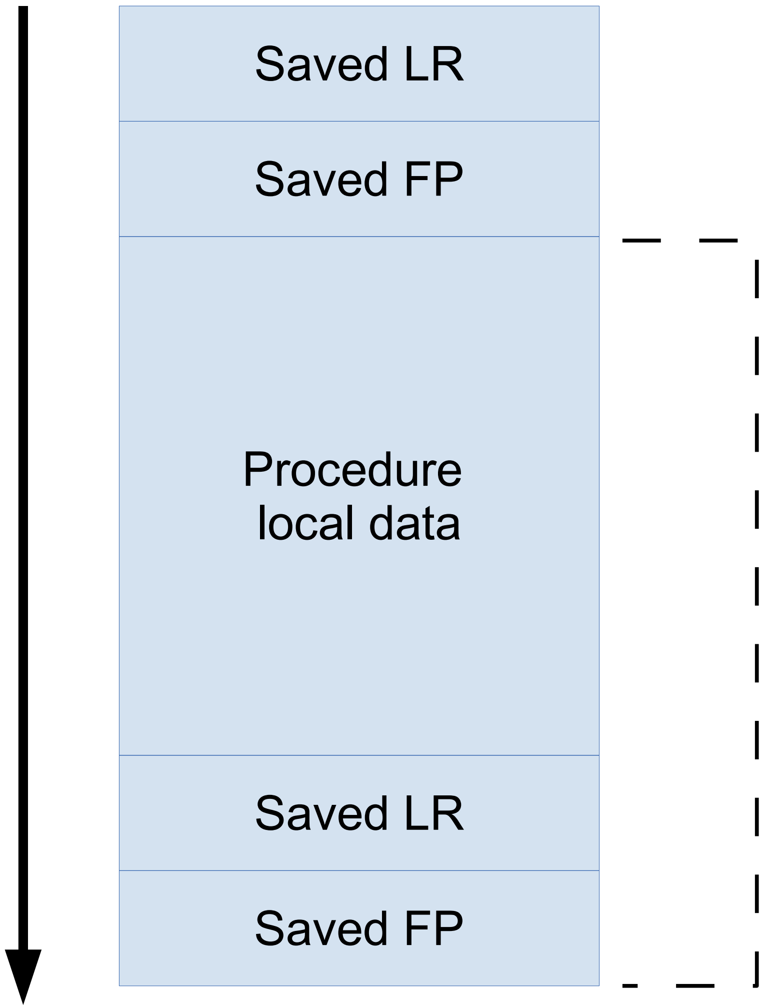
- Transfer: rX = rY || immediate
- ALU: Rd = add(Rs, Rt || immediate)
  [16 bit signed immediate for arithmetic, 10 bit for logical]
- combine: Rdd=combine(immediate, immediate)
  [8 bit signed immediates]
- MUX: Rd = mux(Pu, Rs||immediate, Rt||immediate)
  [8 bit signed immediates]
- NOP: `7f xx xx xx`

Control registers

- LC0 [C1], SA0 [C0],
  LC1 [C3], SA1 [C2]: Loop registers
- PC [C9]: Program counter
- USR [C8]: User status register
- M0 [C6]
  M1 [C7]: Modifier registers
- P3:0 [C4]: Predicate registers
- UGP [C10]: User General Pointer
- GP [C11]: Global Pointer

Calling conventions

- `allocframe`(*size* [u14])
  - Push `LR` and `FP` to top of stack.
  - Subtract *size* [8-byte aligned] from `SP`
  - `FP` = addressof(oldFPonStack)
- `deallocframe`
  - Load saved FP and LR values from address referenced at FP
  - Restore SP to previous frame



Hexagon code, examples

```
some_func:
01 02 03 A3:  memw (r0 + #0xC) = r3 ; memw (r0 + #8) = r1
00 30 02 A4:  memw (r0 + #0x10) = r2 ; memw (r0 + #0) = #0
00 40 9F 52:  {   jumpr r31
80 C0 40 3C:      memw (r0 + #4) = #0 }



[...]
60 46 04 7C   {   r1:0 = combine (#0x33, #8)
46 42 33 04       immext (#0x43309180)
82 45 00 78   r2 = ##filename      @ "/local/mnt/" …
43 C1 03 78   r3 = #0x60A }
51 42 33 04   {   immext (#0x43309440)
A4 46 00 78       r4 = ##message       @ "<PRESENCE" …
00 40 5D 3C       memw (r29 + #0) = #0
80 C0 5D 3C       memw (r29 + #4) = #0 }
4A 63 64 5A   {   call logmsg
00 C1 5D 3C       memw (r29 + #8) = #0 }
[...]
```

Security of chip fabric

- Old(er) Qualcomm chipsets (e.g. MSM7200):
  - baseband was master (access to AP memory & flash)
- Current-gen chipsets have separate ARM7 core for bringup (RPM)
  - Modem firmware now is loaded by HLOS (e.g. Android, iOS)
- Chipset fabric has “hardware firewalls”
  - No documentation leaked on these
- Unclear whether baseband → AP escalation is possible
  - What about DMA?

New RTOS

- Very old Qualcomm chips use proprietary OS REX
- Later, REX was propped onto OKL4
  - commercial microkernel based on L4
- Hexagon-based baseband firmwares abandon OKL4
  - BLAST/QuRT apparently redesigned from scratch
  - Some remnants of REX for compatibility can be found

Comsecuris

Security mitigations

- Stack cookies, generated by build toolchain
- Non-executable stack/heap
  - albeit, according to QCOM security advisory 80-N3172-14 (May 2012): “Enable Data Execution Prevention support in QuRT/BLAST-based images”
- Kernel/user-mode separation in QuRT/BLAST [also 80-N3172-14]
- Safe unlinking for heap
- No ASLR

“ The customer must verify that any performance impact is acceptable.”

[customer = OEM]

ROP & Roll

- Note that deallocframe sets FP
  - very similar popping SP off stack on other architectures
- Instruction packets can be split
  - as long as they are not in cache
- Compound instructions are annoying
  - create constraints for gadgets
- For automation: use SMT solver to handle constraints
  - See BH 2010 talk & WOOT paper on same subject
- Still some way to go
- Manual gadget search works, but very labor-intensive
- Alternate gadgets ending in jump r31 and deallocframe gadgets to get work done

Hands-on training

- Smartphones: most modem firmwares signature checked at boot time (mostly older MDMs, though)
- USB modems: firmware freely modifiable [caveat, there may be exceptions: haven't seen any yet]
- Some Samsung Galaxy S4s (GT-i9505) with MSM8960: no signature check on modem firmware
  - Secure boot type: Samsung
- According to leaked docs modem bringup and sigcheck done by Krait core
  - SBL hacks may help with getting around checks

Comsecuris

Tools

- QDSP6v5 toolchain released by QUIC
  - based on GCC 4.4
- Can be used to compile C/C++ code for Hexagon and inspect using objdump
- Modem firmware: empty ELF section header
  - need to populate to make objdump disassemble
- IDA Pro Hexagon plugin by GSMK (QDSP6v4)
  - also based on released binutils
  - very rudimentary at the moment
  - crashes on some firmwares (e.g. iPhone 5 baseband)

Comsecuris

Leaked bugs: An example (CR 310629)

- Background: some while ago, archive of chipset docs on MSM8960 appeared on XDA Developers site
  - Someone had put 7 AMSS security bulletins into this
- Classic stack buffer overflow
- In LTE air interface
- Occurs when processing Test Loopback messages
  - Simple L3 messages > 100 bytes trigger this problem
- Mitigated by use of `-fstack-protector`
- Appeared in May 2012 security advisory
  - Detailed description given
- Still, surprising to see such straightforward bugs
  - Possible explanation: LTE stack was still "young"

Comsecuris

The Way Forward

- New architecture has raised bar of entry significantly
- However, Qualcomm dominates market
  - People will and do have interest in their chips
- Well-funded attackers will adapt
- Public leaks of vulnerability information make attackers task easier
  - Takedown possible, but the internet “doesn't forget”
  - Don't find bugs, find bug description
  - OEMs sometimes have slow patch cycles
- ROP exploitation needs automation
  - Not as difficult as assumed