A Journey into Hexagon

Dissecting Qualcomm Basebands

Seamus Burke

Agenda

- About me
- Why Basebands?
- History
- Modern SoC Architecture
- Hexagon
- Cellular Stack architecture
- Analysis

About Me

- Student, still finishing up my undergrad
- Interested in kernel internals, exploit development, and embedded systems
- Plays a lot of CTFs

Goals

- Find bugs in the baseband
- Understand how it works and how it interacts with Android

Prior Work

“Reverse Engineering a Qualcomm Baseband” - Guillaume Delugré(fix char)

“Baseband Exploitation in 2013” - Ralph-Philipp Weinmann (PacSec 2013)

“Exploring Qualcomm Baseband via Modkit” - Peter Pi, XiLing Gong, Gmxp (CSW 2018)

Agenda

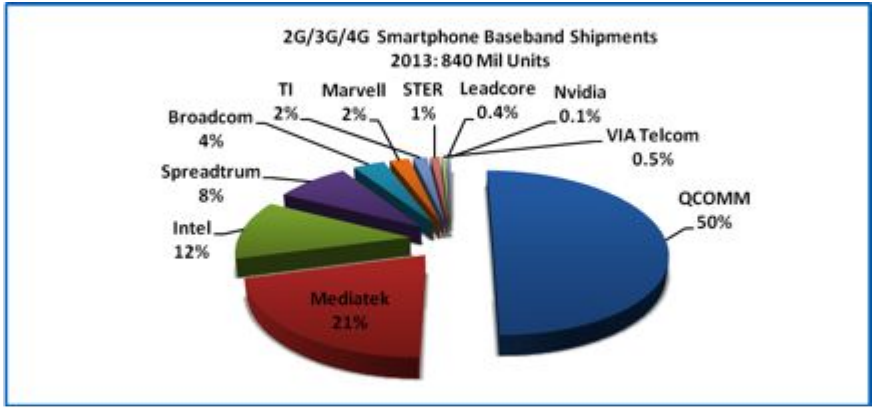
- About me
- Why Basebands?
- History
- Modern SoC Architecture
- Hexagon
- Cellular Stack architecture
- Analysis

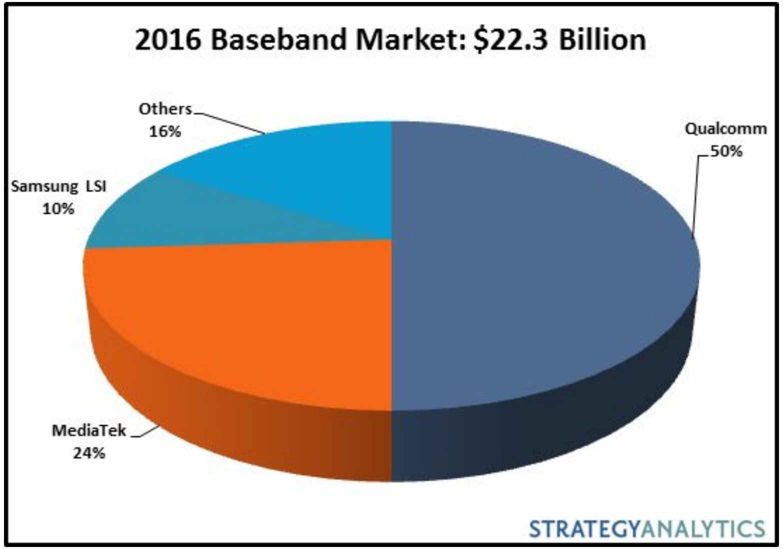
What is a baseband exactly?

- The chip in a phone which communicates with the cellular network
- Handles the radio signal processing over the air
- Has to support a large number of standards - (GSM,UMTS,CDMA2k,cdmaOne, GPRS,EDGE, LTE, etc)
- The phones main interface to the rest of the world

Why Qualcomm?

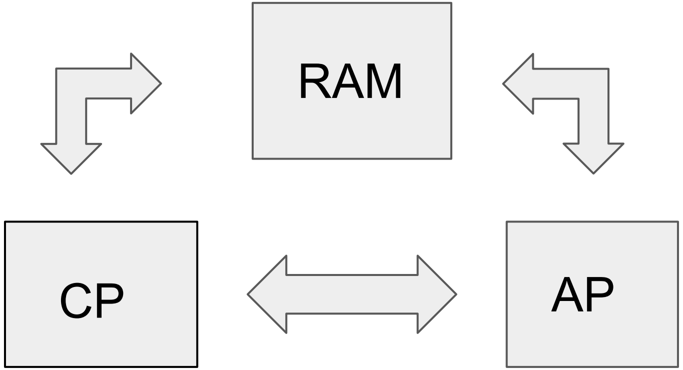
- By far the largest market share of any baseband processor
- Most high-end phones on the market, at least till recently carried a qcom chip inside.





Basebands today

- Separate cellular and application processors
- Some sort of communication between them
- Both have access to RAM



Agenda

- About me
- Why Basebands?
- History
- Modern SoC Architecture
- Hexagon
- Cellular Stack architecture
- Analysis

RTOS

- Real-time, embedded operating systems.
- 2 major categories
  - Unprotected - logical and physical addresses are the same
  - Protected - virtual memory
- Time bound, with well defined time constraints

ecos
THREADX
VxWorks
NUCLEUS
freeRTOS
QNX
QNX SOFTWARE SYSTEMS

REX - Real-time Executive

- The original kernel which ran the modem
- Designed for the 80186, then ported to ARM
- Single process, multiple threads
- Everything runs in supervisor mode

CD-7000: CDMA becomes truly mobile

The CD-7000 was the first handheld mobile CDMA phone. It was created in 1993 for operators to test their network systems.

- Tasks are basically the threads of REX.
- Every task has a TCB storing the the SP, priority, stack_limit, etc
- Each task has it’s own stack, when tasks are suspended, context is saved in a special context frame on top of it’s stack
- Pointer to the saved context stored in the tasks TCB
- TCBs stored in a doubly linked list, trivial to mess with

- Why did Qualcomm switch from REX?
- Well, it had it’s issues:

“Programmers should use these functions only according to the interface specifications, since REX does not check any parameter for validity”

- Flat address space, lack of memory protections
  - Did they switch for security? Hah, no, debugging millions of lines of C with no memory protections was a nightmare

L4 + Iguana

- Multiple-process and multiple-threaded
- Only the kernel runs in supervisor mode, everything else in userland
- A REX emulation layer is supported
  - REX tasks are L4 threads
  - No changes to the REX api, it’s converted transparently
  - AMSS runs in user mode
  - Interrupts split between kernel and user mode

QuRT

- Qualcomm Real-time OS
- Used to be named BLAST, name changed as part of OpenDSP initiative
- Most of the APIs are backwards compatible, with the exception of some threading things.

- QuRT provides all the OS primitives you would expect
    - Mutexes
    - Futexes
    - Semaphores
    - Task Scheduling
- Priority based scheduling
    - Priorities 0-256, 0 is the highest
    - Tries to schedule an interrupt in an idle hw thread, instead of preempting a running task

Exceptions

- Application exceptions
  - Page faults, misaligned operations, processor exceptions, etc
  - Handled by registered exception handlers
- Kernel exceptions
  - Page faults and other processor exceptions (TODO like what?)
  - Cause all execution to be terminated and the processor to be shut down
  - Processor state is saved as best it can be

Mitigations? Sorta

- Complete lack of ASLR
- There is a form of DEP, can’t write to code, can’t execute data
- XORd stack cookies
- Heap protection
    - Different magic values for the headers of in-use and free’d blocks
- Lots of fixed addresses everywhere. The RTOS loads at the same spot every time, as does just about everything else.
- Hardcoded addresses prevalent in the code

AMSS

- Advanced Mobile Subscriber Software, drivers and protocol stacks which make up the modem
- Configured differently for different chipsets
    - Which air interface protocols are supported
    - Hardware specific drivers
    - Multimedia software
- > 60 tasks running
    - Diag, Dog, Sleep, CM, PS, DS, etc

Diagnostics

- DIAG, or Diagnostic Services provides a server that clients can request info from about AMSS
- DIAG is a REX task, usually handles requests from Serial or USB
- Packet based protocol
- Lots of useful stuff
  - Debug messages
  - OTA logs
  - Status
  - Statistics

Agenda

- About me
- Why Basebands?
- History
- Modern SoC Architecture
- Hexagon
- Cellular Stack architecture
- Analysis

Qfuses

- Internal bank of one time programmable fuses, the QFPROM
- Publically undocumented
- Inter-chip configuration settings, cryptographic keys
- Secure boot and TrustZone both make heavy use of these
- Hardware debugging usually disabled in prod by blowing a fuse

SoC Architecture I

- Multiple interconnected subsystems
  - MPSS - Modem Processor
  - APPS - Application processor
  - RPM - Resource and Power Management
  - WCNSS - Wireless Connectivity
  - LPASS - Low Power Audio

AP <-> CP communication

- So how does Android talk to the modem?
- Shared memory
- QMI

Shared Memory

- Main idea is for the Modem to write some data, and the AP pick it up
- Common APIs on both the modem (and other subsystems) and linux side
  - Smem_init, smem_alloc, smem_find, smem_get_entry
- SMD - Shared Memory Driver
  - Wrapper around SMEM
  - Abstracts into things like pipes
  - Separate channels for DS, DIAG, RPC, GPS

QMI

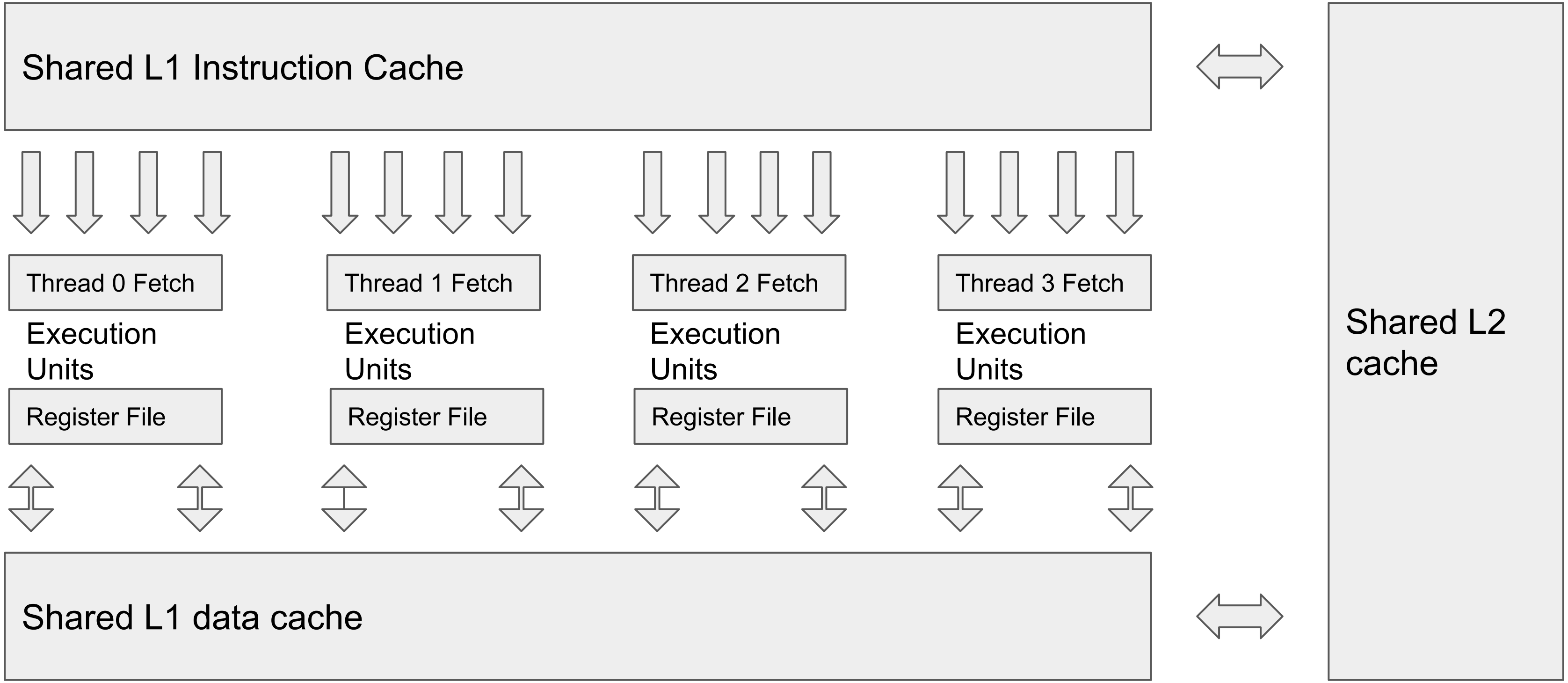
- Qualcomm MSM Interface - designed to supplant the AT cmd set
- High level interface over older protocol (DMSS)
- Used to interface with modem components, but not drive hw
- Client-server model
- Packet structure with a header and then TLV payloads

Agenda

- About me
- Why Basebands?
- History
- Modern SoC Architecture
- Hexagon
- Cellular Stack architecture
- Analysis

Hexagon

- 6th iteration of Qualcomm’s in house DSPs.
- General purpose DSP. 2 on the SoC, one for applications to use, and one for the modem
- Separate L1 code and data caches, unified L2 cache
- Hardware threads share caches
- Instructions grouped into packets of 1-4 instructions for parallel execution



Shared L1 Instruction Cache

Thread 0 Fetch

Execution Units

Register File

Thread 1 Fetch

Execution Units

Register File

Thread 2 Fetch

Execution Units

Register File

Thread 3 Fetch

Execution Units

Register File

Shared L1 data cache

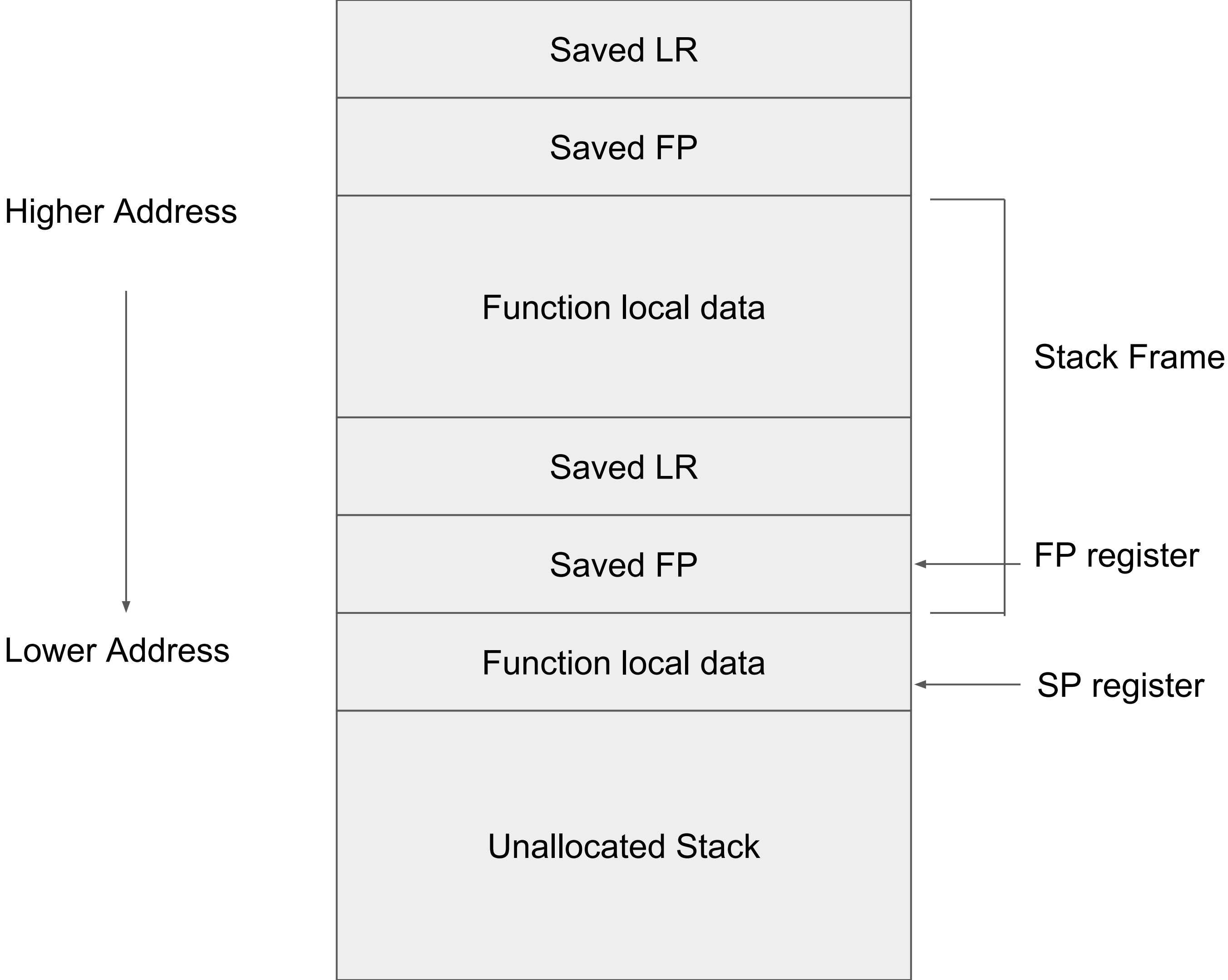
Shared L2 cache

General Info I

- Thirty two 32-bit GPRs
- Calling convention - R0-R5 are used for arguments
- Return values in R0
- Caller saved are R6-R15
- Callee saved R16-R27

The stack on QDSP

- R29 is Stack Pointer, R30 is Frame Pointer, R31 is Link Register
- The stack grows downwards from high to low
- Needs to be 8 byte aligned
- Several stack specific instructions
  - Allocframe - pushes LR and FP to stack, and subtracts from SP to make room for the new frames locals.
  - Deallocframe - Loads FP and LR, then fixes up SP
  - Dealloc_return - does a deallocframe and then returns to LR



Higher Address

Lower Address

Saved LR

Saved FP

Function local data

Saved LR

Saved FP

Function local data

Unallocated Stack

Stack Frame

FP register

SP register

General Info II

- SSR - holds a variety of useful debugging info
    - ASID
    - CAUSE
    - Which BadVA
- BADVA
    - BADVA0 - exception addresses generated from slot0
    - BADVA1 - addresses generated from slot1
- ELR - holds PC value when an exception occurs

Privilege Modes

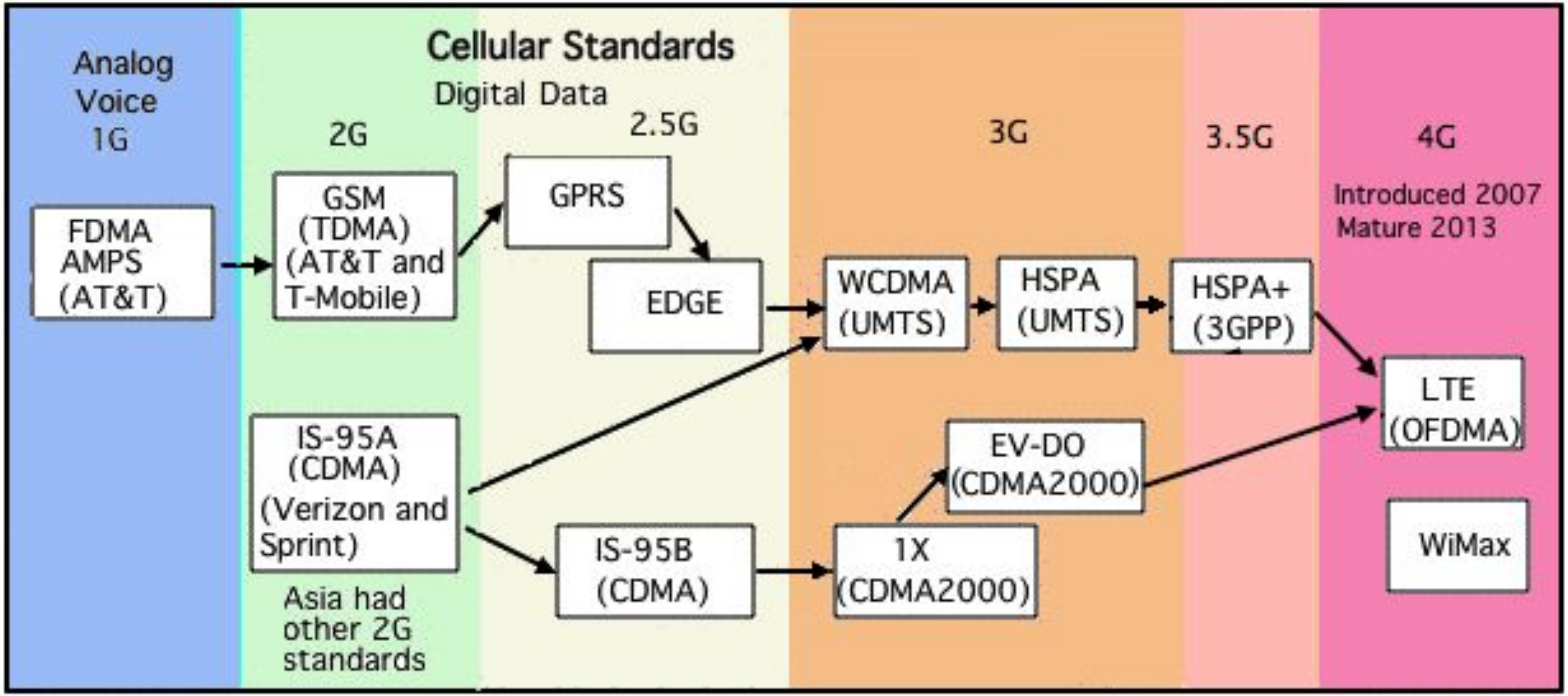
- 3 Main modes
  - Kernel Mode - Access to all memory, smallest memory footprint
  - Guest OS - Access to it’s own memory, and of the User segment, lots of Qcom drivers run here
  - User - Only has access to itself.
- Stack checks only done in user and guest

Protection Domains

- New with QDSP6v62
- Implements Separate address spaces
- Memory mapping is Address space ID + 32 bit VA
- Address spaces can’t touch each other
- ASID0 is the kernel and Guest OS levels.

Agenda

- About me
- Why Basebands?
- History
- Modern SoC Architecture
- Hexagon
- Cellular Stack architecture
- Analysis



Cellular Standards

Digital Data

Analog Voice 1G

FDMA AMPS (AT&T)

2G

GSM (TDMA) (AT&T and T-Mobile)

IS-95A (CDMA) (Verizon and Sprint)

Asia had other 2G standards

2.5G

GPRS

EDGE

IS-95B (CDMA)

3G

WCDMA (UMTS)

HSPA (UMTS)

EV-DO (CDMA2000)

1X (CDMA2000)

3.5G

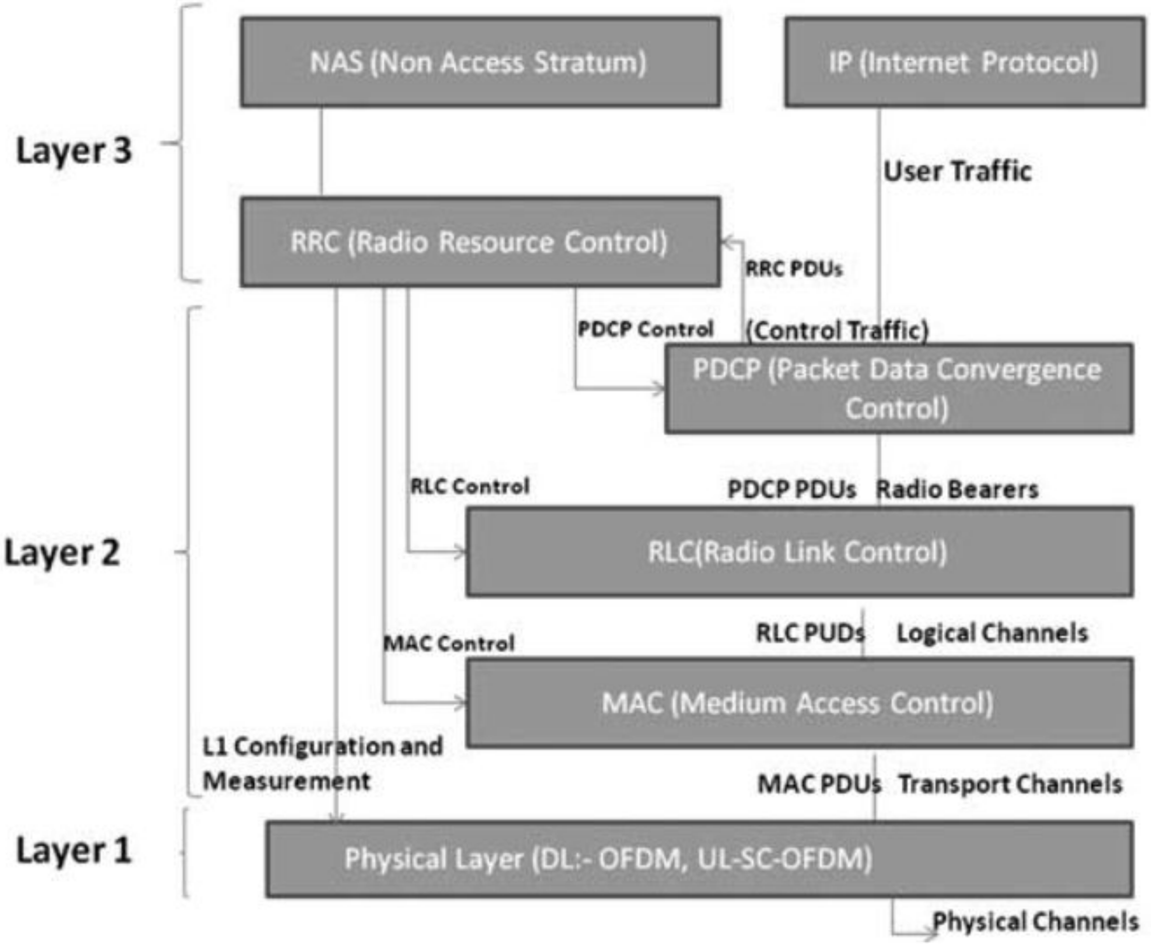
HSPA+ (3GPP)

4G

Introduced 2007
Mature 2013

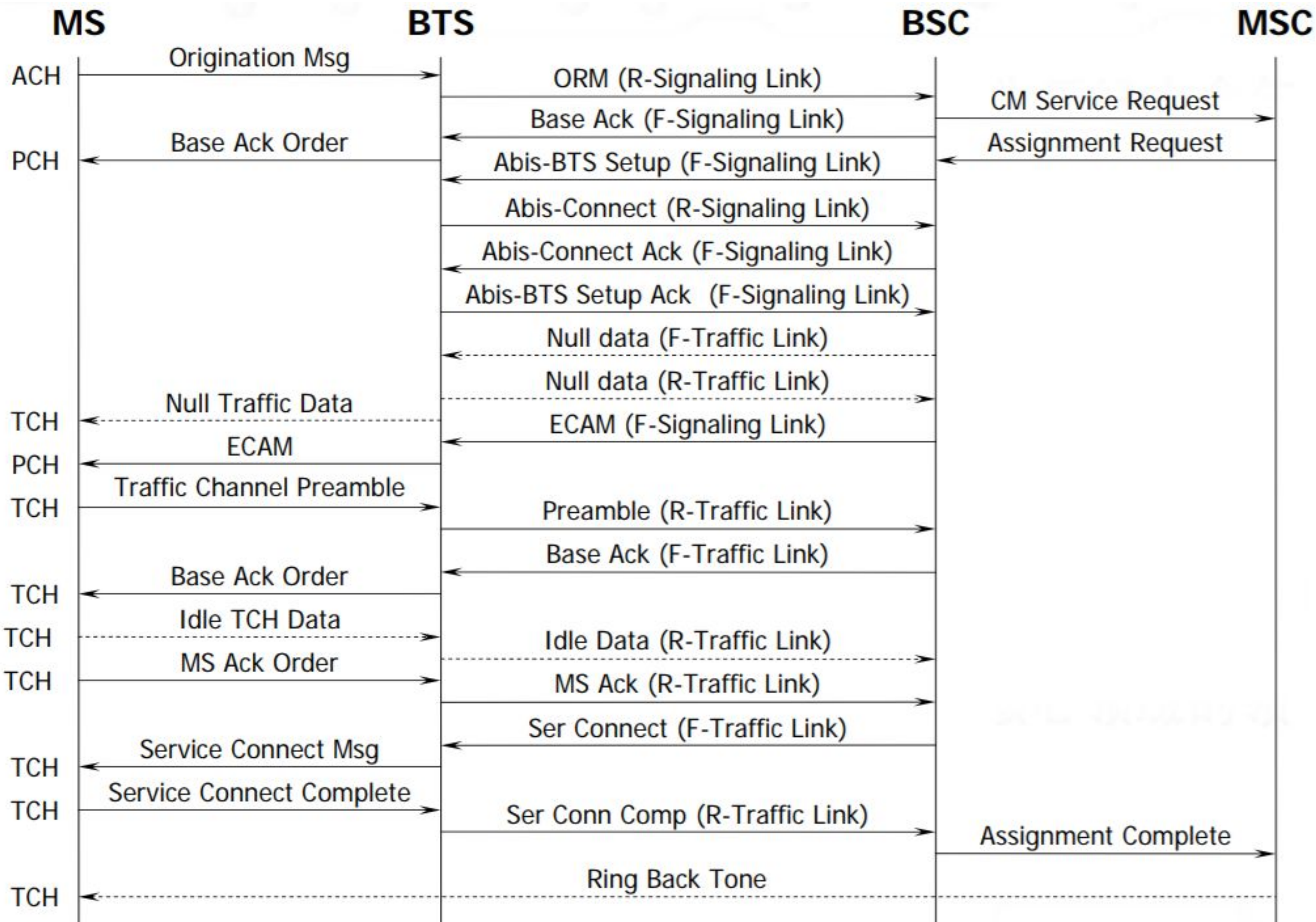
LTE (OFDMA)

WiMax


Layer 3
Layer 2
Layer 1
NAS (Non Access Stratum)
IP (Internet Protocol)
User Traffic
RRC (Radio Resource Control)
RRC PDUs
PDCP Control
(Control Traffic)
PDCP (Packet Data Convergence Control)
RLC Control
PDCP PDUs
Radio Bearers
RLC(Radio Link Control)
MAC Control
RLC PUDs
Logical Channels
MAC (Medium Access Control)
L1 Configuration and Measurement
MAC PDUs
Transport Channels
Physical Layer (DL:- OFDM, UL-SC-OFDM)
Physical Channels

Call flow

- There are a dozen different ways the cell stack can make/receive a call
  - 1x voice call
  - 1x data call
  - Hdr call
  - Gsm voice call
  - Gprs data call
  - Wcdma voice call
  - Wcdma data call
  - Td-scdma call
  - Lte data call
- Multiple ways to do the same thing implies added complexity, and these aren’t as simple as a 3-way handshake to begin with



MS BTS BSC MSC

ACH Origination Msg

ORM (R-Signaling Link)

CM Service Request

Base Ack (F-Signaling Link)

PCH Base Ack Order

Assignment Request

Abis-BTS Setup (F-Signaling Link)

Abis-Connect (R-Signaling Link)

Abis-Connect Ack (F-Signaling Link)

Abis-BTS Setup Ack (F-Signaling Link)

Null data (F-Traffic Link)

Null data (R-Traffic Link)

TCH Null Traffic Data

ECAM (F-Signaling Link)

PCH ECAM

TCH Traffic Channel Preamble

Preamble (R-Traffic Link)

Base Ack (F-Traffic Link)

TCH Base Ack Order

TCH Idle TCH Data

Idle Data (R-Traffic Link)

TCH MS Ack Order

MS Ack (R-Traffic Link)

Ser Connect (F-Traffic Link)

TCH Service Connect Msg

TCH Service Connect Complete

Ser Conn Comp (R-Traffic Link)

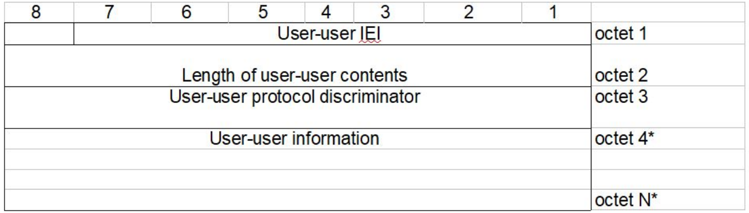
Assignment Complete

TCH Ring Back Tone

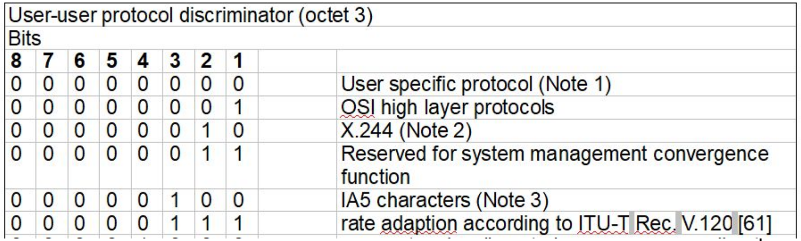
RTFS

- Best place to start is the standards
- Don’t specify implementation, and there are lots of features to implement
- 3GPP is the governing body
    - Composed of 7 telecom orgs (ETSI, ARIB, ATIS, CCSA, TSDSI, TTA, TTC)
- The standards are freely available on the web
- (Very long)

- Plenty of LV and TLV options everywhere. Good place to start
- Can be tricky to find out how to trigger them



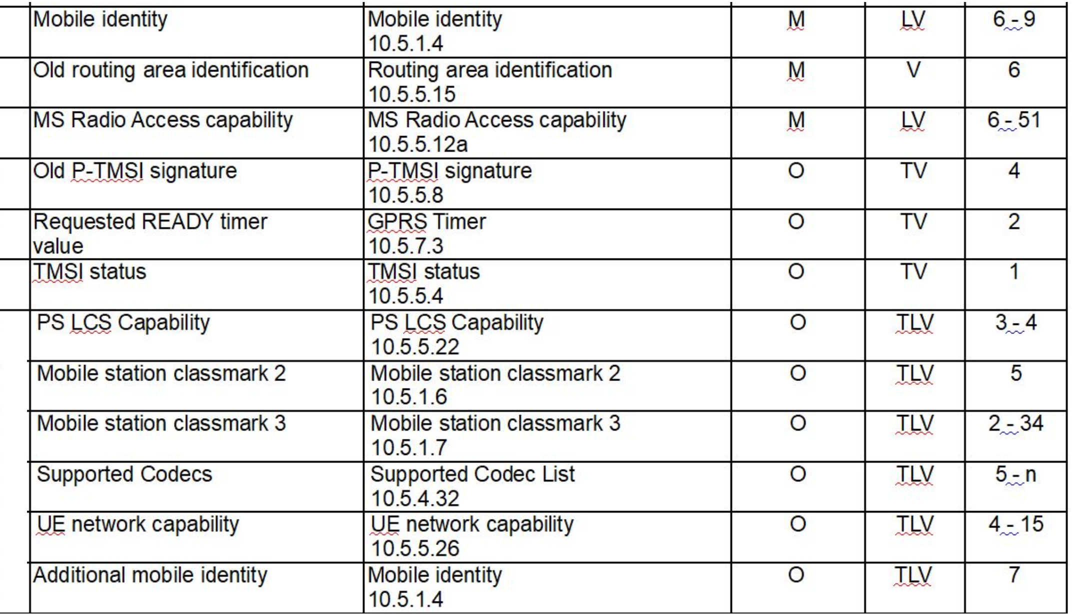
8	7	6	5	4	3	2	1	
	User-user IEI							octet 1
Length of user-user contents								octet 2
User-user protocol discriminator								octet 3
User-user information								octet 4*
								octet N*



User-user protocol discriminator (octet 3)

Bits

8	7	6	5	4	3	2	1	
0	0	0	0	0	0	0	0	User specific protocol (Note 1)
0	0	0	0	0	0	0	1	OSI high layer protocols
0	0	0	0	0	0	1	0	X.244 (Note 2)
0	0	0	0	0	0	1	1	Reserved for system management convergence function
0	0	0	0	0	1	0	0	IA5 characters (Note 3)
0	0	0	0	0	1	1	1	rate adaption according to ITU-T Rec. V.120 [61]



	Mobile identity	Mobile identity 10.5.1.4	M	LV	6 - 9
	Old routing area identification	Routing area identification 10.5.5.15	M	V	6
	MS Radio Access capability	MS Radio Access capability 10.5.5.12a	M	LV	6 - 51
9	Old P-TMSI signature	P-TMSI signature 10.5.5.8	O	TV	4
7	Requested READY timer value	GPRS Timer 10.5.7.3	O	TV	2
9-	TMSI status	TMSI status 10.5.5.4	O	TV	1
33	PS LCS Capability	PS LCS Capability 10.5.5.22	O	TLV	3 - 4
	Mobile station classmark 2	Mobile station classmark 2 10.5.1.6	O	TLV	5
	Mobile station classmark 3	Mobile station classmark 3 10.5.1.7	O	TLV	2 - 34
	Supported Codecs	Supported Codec List 10.5.4.32	O	TLV	5 - n
	UE network capability	UE network capability 10.5.5.26	O	TLV	4 - 15
	Additional mobile identity	Mobile identity 10.5.1.4	O	TLV	7

Agenda

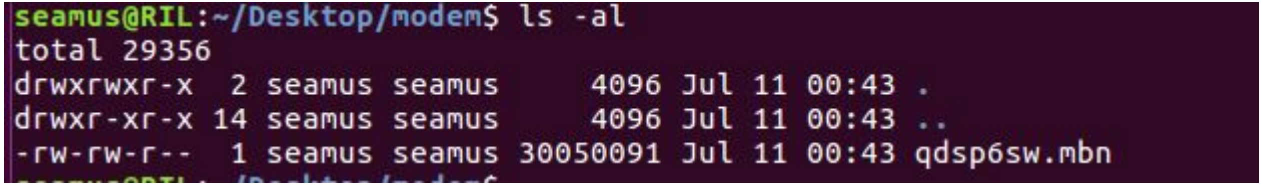
- About me
- Why Basebands?
- History
- Modern SoC Architecture
- Hexagon
- Cellular Stack architecture
- Analysis

Disassembly options

- There are several options out there for disassembling hexagon code
  - https://github.com/gsmk/hexagon
  - https://github.com/programa-stic/hexag00n
  - https://github.com/rpw/hexagon
  - Qcom provided patches for GNU binutils
  - Llvm, codebench, etc
- I found the GSMK plugin the fastest to setup and get running
- I have a very rough binary ninja based disassembler I wrote

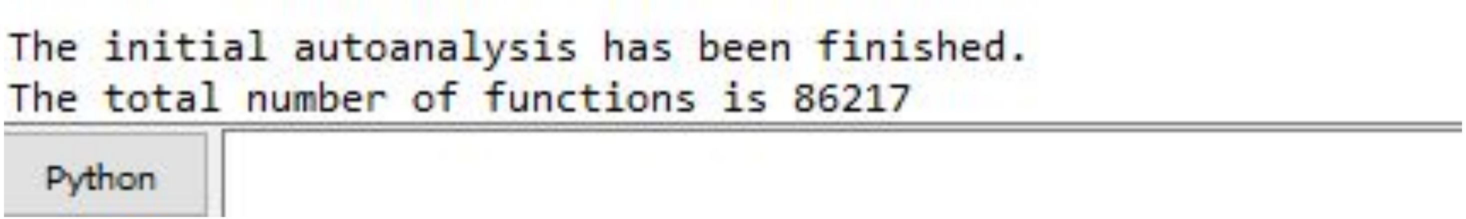
Analysis

- Qdsp6sw.mbn - Holds the modem firmware and QuRT
- Not small -



```
seamus@RIL:~/Desktop/modem$ ls -al
total 29356
drwxrwxr-x  2 seamus seamus     4096 Jul 11 00:43 .
drwxr-xr-x 14 seamus seamus     4096 Jul 11 00:43 ..
-rw-rw-r--  1 seamus seamus 30050091 Jul 11 00:43 qdsp6sw.mbn
```

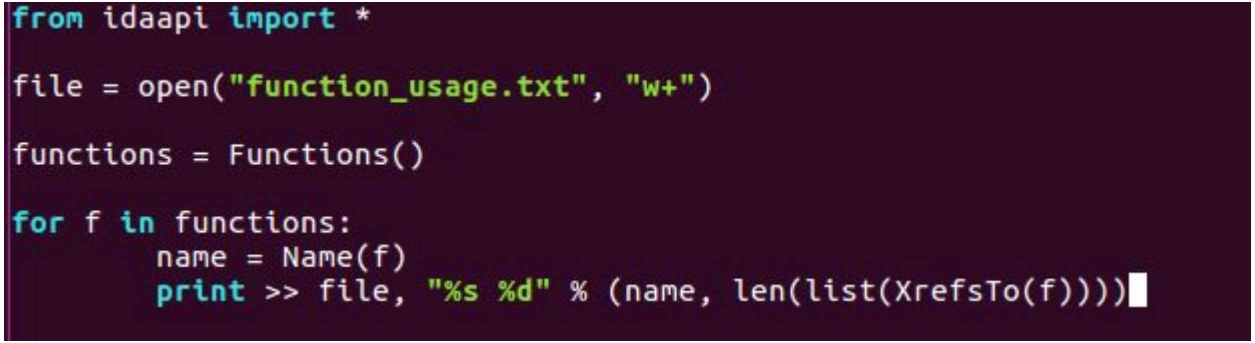
- It has tens of thousands of functions to sort through
- Where to start?



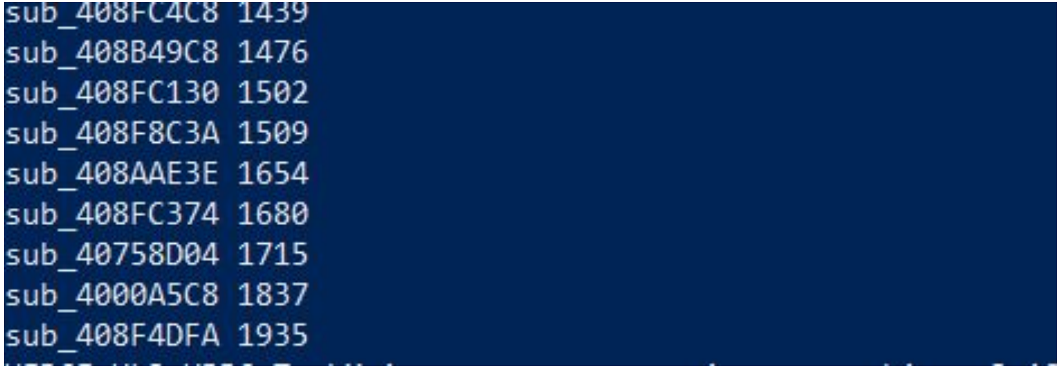
```
The initial autoanalysis has been finished.
The total number of functions is 86217
```

Library function identification via frequency

- Idea: identify common library functions via high usage



```
from idaapi import *

file = open("function_usage.txt", "w+")

functions = Functions()

for f in functions:
        name = Name(f)
        print >> file, "%s %d" % (name, len(list(XrefsTo(f))))
```



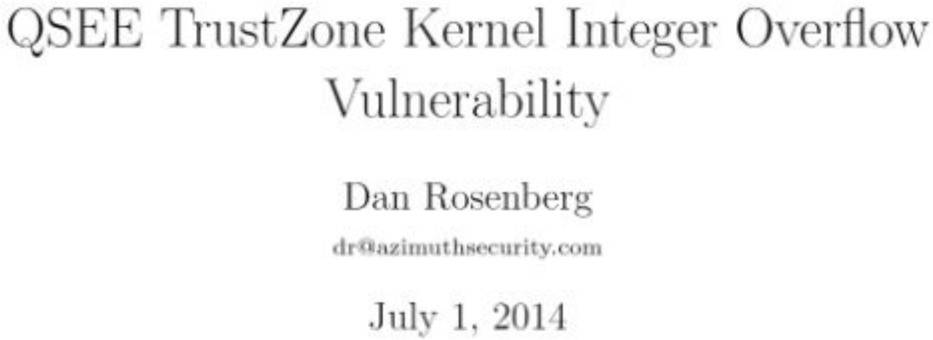
```
sub_408FC4C8 1439
sub_408B49C8 1476
sub_408FC130 1502
sub_408F8C3A 1509
sub_408AAE3E 1654
sub_408FC374 1680
sub_40758D04 1715
sub_4000A5C8 1837
sub_408F4DFA 1935
```

Debugging

- A few different options here
    - Qcom tools like QXDM/QPST, talk to the phone over USB
        - Acquiring, licensing, ramdumps(!)
    - JTAG
        - More cost, slightly higher difficulty
    - Lauterbach Trace 32
        - Expensive, licensing, gets you as low level as you’re gonna get
        - More on this later
    - Memory R/W via exploit
    - Modem Image modification

Modem image patching I

- Modem binaries are unencrypted on disk
- This facilitates easy disassembly, and easy patching
- Secboot prevents unsigned images from loading
- Signature verification performed in secure world



QSEE TrustZone Kernel Integer Overflow Vulnerability

Dan Rosenberg

dr@azimuthsecurity.com

July 1, 2014

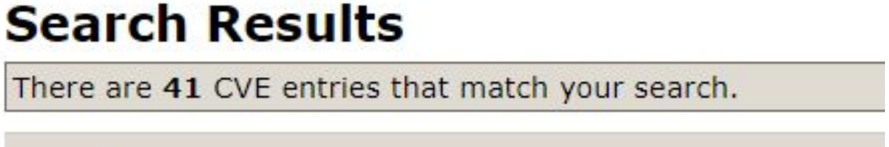
Modem Patching II

https://github.com/eti1/tzexec

Leverages a integer overflow to achieve an arbitrary write into the trustzone, and patches two bytes to neuter signature checking

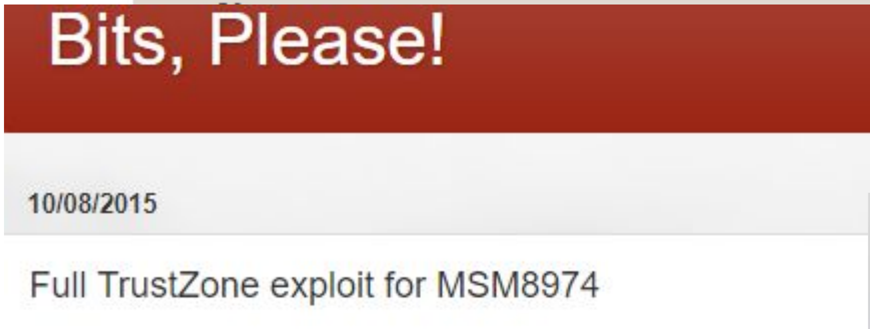
Prereqs: ability to compile your own kernel and flash it

Modem internal hashes still need to be consistent



Search Results

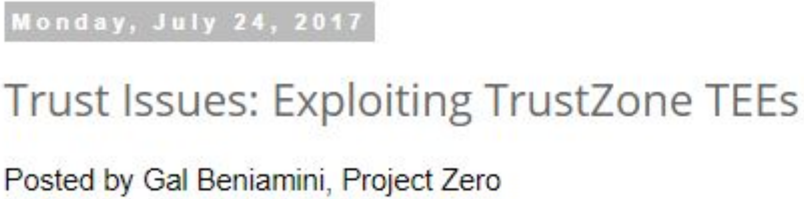
There are **41** CVE entries that match your search.



Bits, Please!

10/08/2015

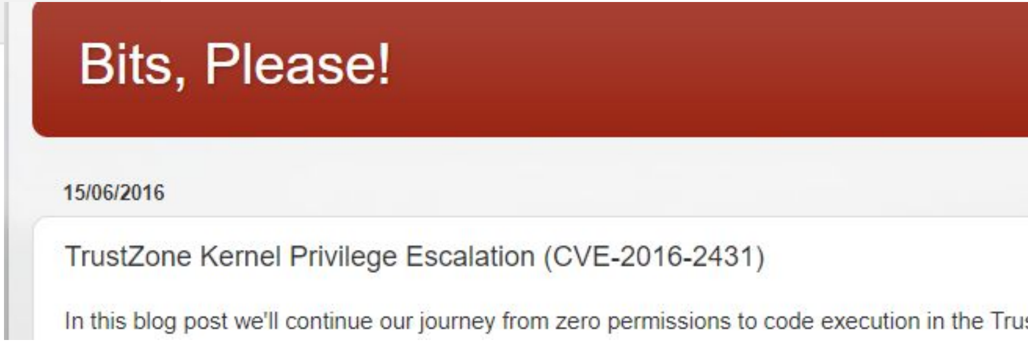
Full TrustZone exploit for MSM8974



Monday, July 24, 2017

Trust Issues: Exploiting TrustZone TEEs

Posted by Gal Beniamini, Project Zero



Bits, Please!

15/06/2016

TrustZone Kernel Privilege Escalation (CVE-2016-2431)

In this blog post we'll continue our journey from zero permissions to code execution in the Trus

Implementing a debugger

What are the preconditions of a debugger?

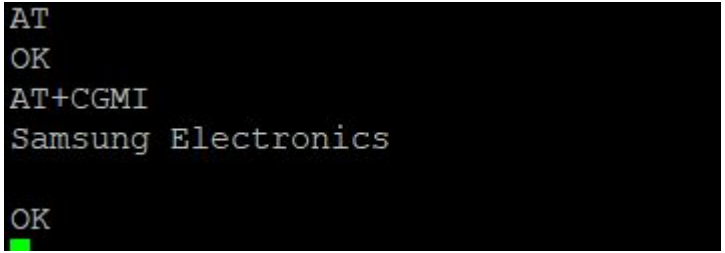
- Able to read and write from/to memory (setting breakpoints, etc)
- Breakpoints and the like implies the ability to change memory permissions
- Setting register values

How does a baseband take input?

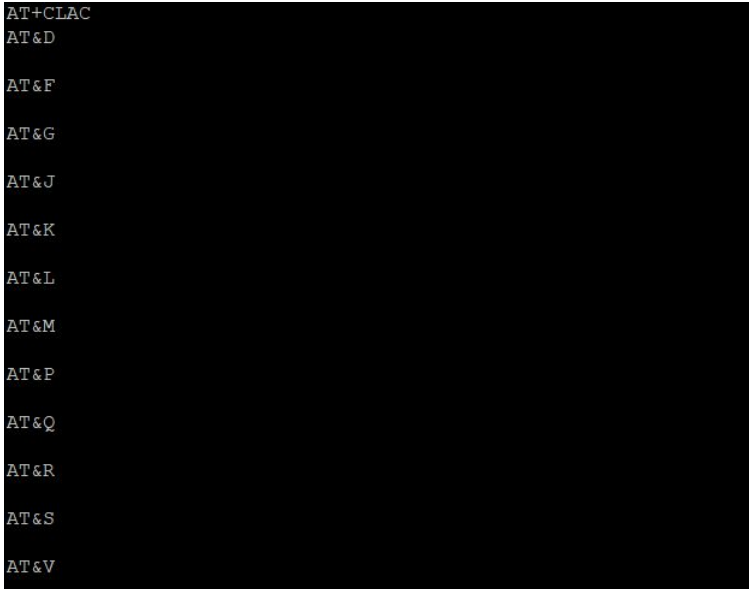
- Over the air interface
- Shared memory
- Serial

Hayes AT commands

- Commands sent to the modem to control dialing, connection parameters, and generally manipulate things
- Extended a lot, OEM and carrier specific commands supported



```
AT
OK
AT+CGMI
Samsung Electronics

OK
```



```
AT+CLAC
AT&D

AT&F

AT&G

AT&J

AT&K

AT&L

AT&M

AT&P

AT&Q

AT&R

AT&S

AT&V
```

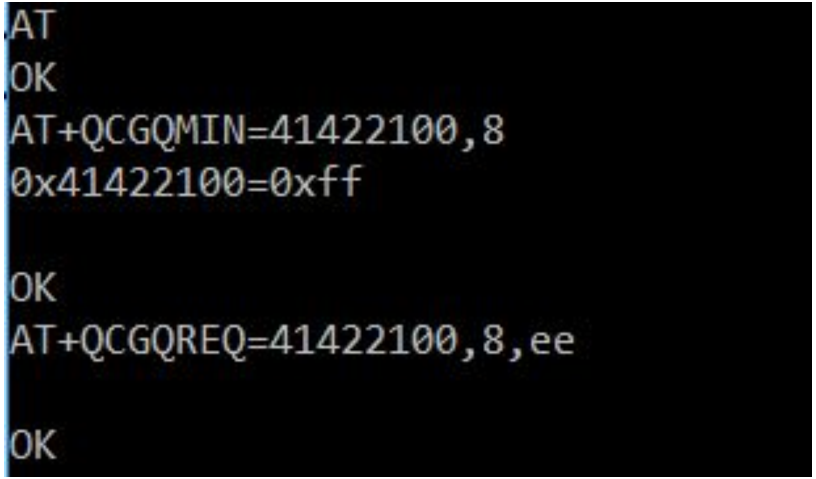
Implementing a debugger II

Can hook/replace AT commands

Read = AT+cmd=address,size

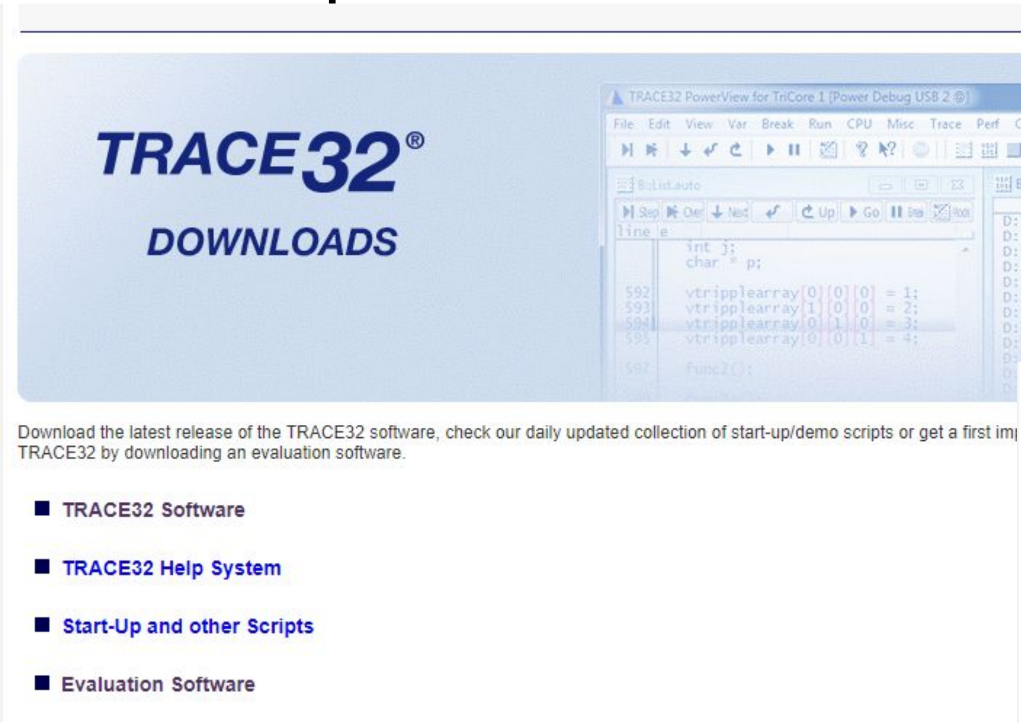
Write = AT+cmd=address,size,data

Just picked arbitrary commands to replace



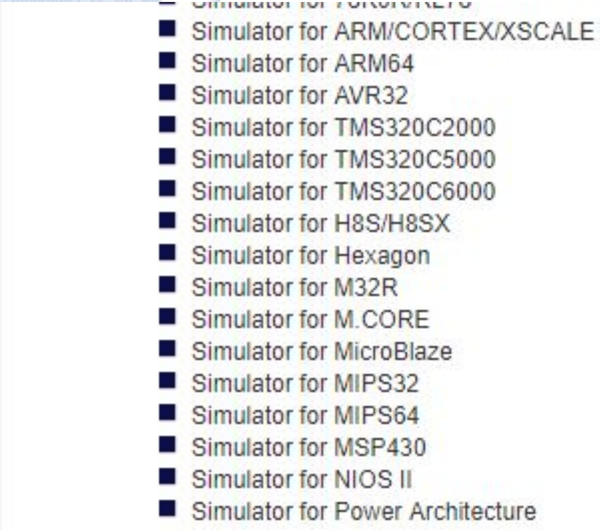
```
AT
OK
AT+QCGQMIN=41422100,8
0x41422100=0xff

OK
AT+QCGQREQ=41422100,8,ee

OK
```

Or…..option II



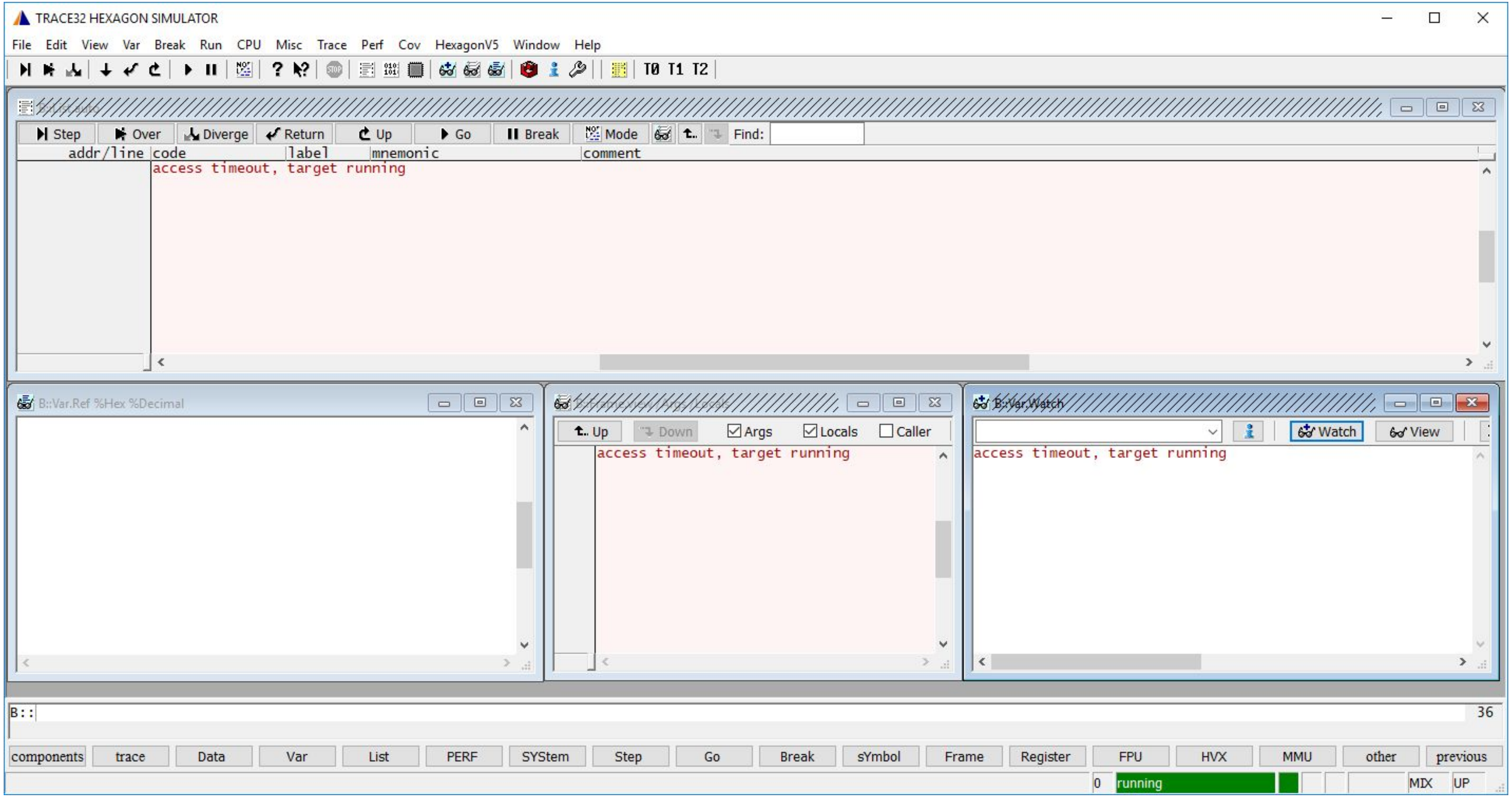
Download the latest release of the TRACE32 software, check our daily updated collection of start-up/demo scripts or get a first imp
TRACE32 by downloading an evaluation software.

- TRACE32 Software
- TRACE32 Help System
- Start-Up and other Scripts
- Evaluation Software



- Simulator for ARM/CORTEX/XSCALE
- Simulator for ARM64
- Simulator for AVR32
- Simulator for TMS320C2000
- Simulator for TMS320C5000
- Simulator for TMS320C6000
- Simulator for H8S/H8SX
- Simulator for Hexagon
- Simulator for M32R
- Simulator for M.CORE
- Simulator for MicroBlaze
- Simulator for MIPS32
- Simulator for MIPS64
- Simulator for MSP430
- Simulator for NIOS II
- Simulator for Power Architecture

Full-featured TRACE32 Instruction Set Simulators for Windows are available for free download.
Please be aware that the scripting and the remote control are limited.


TRACE32 HEXAGON SIMULATOR
File Edit View Var Break Run CPU Misc Trace Perf Cov HexagonV5 Window Help
T0 T1 T2
Step Over Diverge Return Up Go Break Mode Find:
B::Var.Ref %Hex %Decimal
Up Down ☑ Args ☑ Locals □ Caller
B::Var.Watch
Watch View
components trace Data Var List PERF SYStem Step Go Break sYmbol Frame Register FPU HVX MMU other previous

```
addr/line |code          |label    |mnemonic              |comment
          access timeout, target running
```

```
access timeout, target running
```

```
access timeout, target running
```

```
B::                                                                 36
```

```
0  running        MIX  UP
```

Testing

- Usually need a license to broadcast on cellular frequencies (depending on country)
- Or get a Faraday cage
- Can use a Software Defined Radio (SDR) to implement our own cell stack
- A SDR is a general purpose transceiver, they usually support a variety of different frequencies

Testing II

- Various hardware options
  - BladeRF x40 ~$420
  - BladeRF x115 ~$650
  - USRP B200 ~$675

Testing III

- Quite a few open source projects have sprung up in the past few years
- YateBTS - GSM and GPRS
- OpenBTS - GSM, GPRS, 3G (UMTS)
- OpenBSC - GSM, GPRS
- OpenAirInterface - LTE
- OpenLTE - LTE
- srsLTE - LTE

Questions?