Android & TrustZone

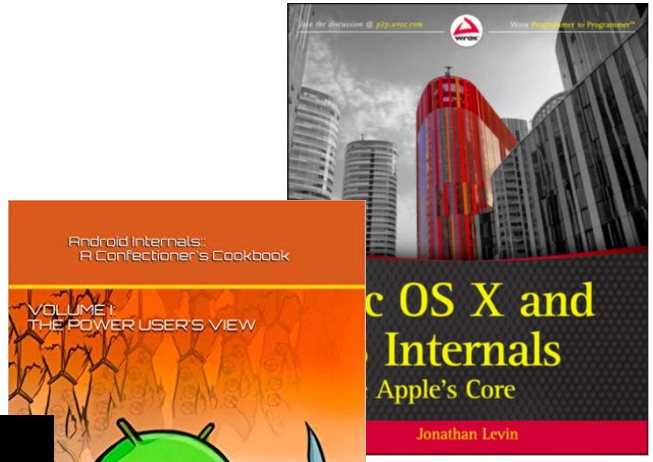
Jonathan Levin

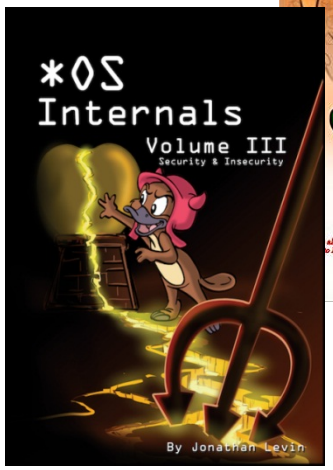
http://Technologeeks.com

`whoami`

- Jonathan Levin, CTO of technologeeks[.com]
  - Group of experts doing consulting/training on all things internal

- Author of a growing family of books:
  - Mac OS X/iOS Internals
  - Android Internals (http://NewAndroidbook.com)
  - *OS Internals (http://NewOSXBook.com)





Plan

- TrustZone
  - Recap of ARMv7 and ARMv8 architecture

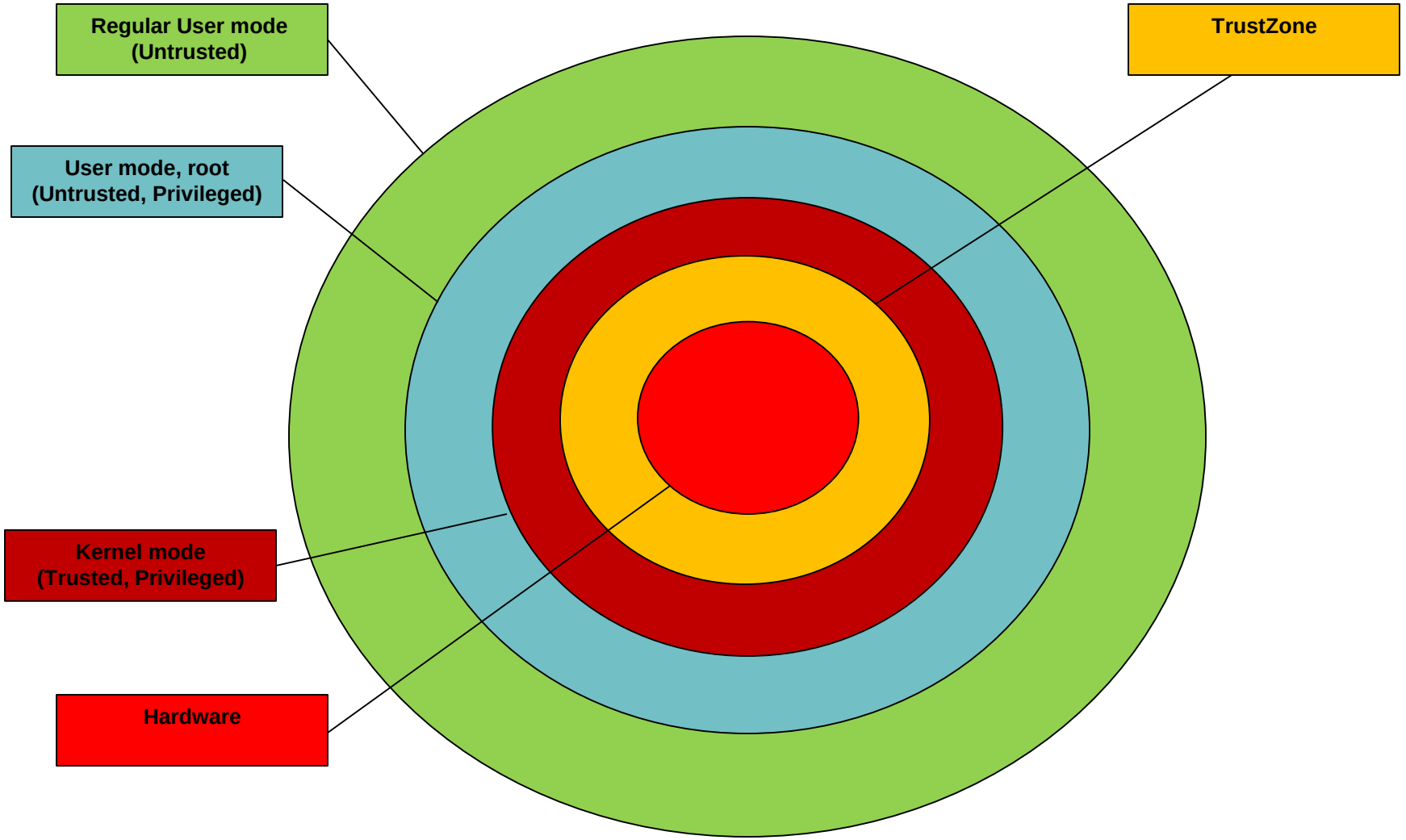
- Android Implementations
  - Samsung, Qualcomm, Others

TrustZone & ELx

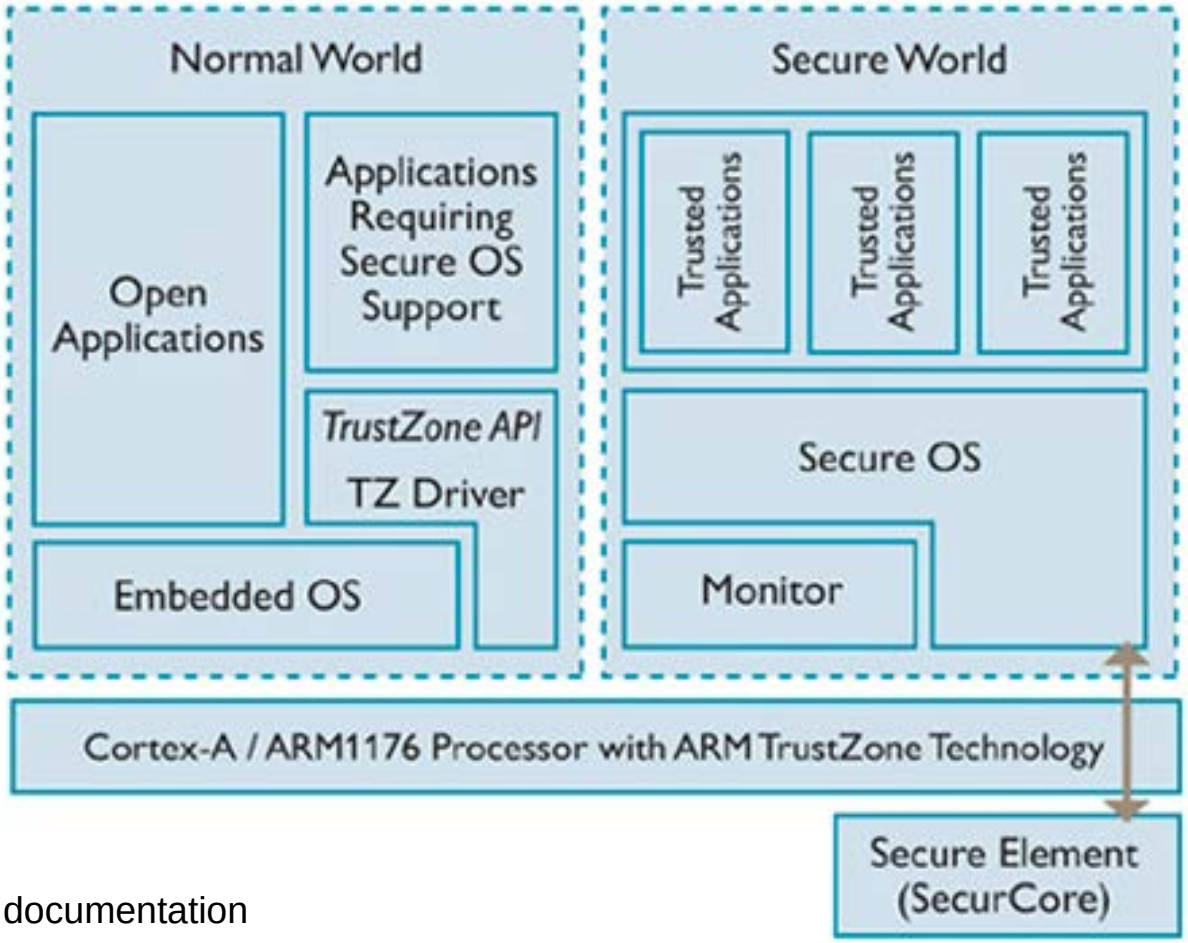
TrustZone

- Hardware support for a trusted execution environment
- Provides a separate “secure world”
  - Self-contained operating system
  - Isolated from “non-secure world”
- In AArch64, integrates well with Exception Levels
  - EL3 only exists in the secure world
  - EL2 (hypervisor) not applicable in secure world.
- De facto standard for security enforcement in mobile world

TrustZone



Trust Zone Architecture (Aarch32)



Source: ARM documentation

Android uses of TrustZone

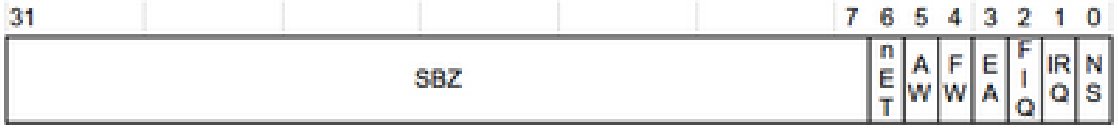
- Cryptographic hardware backing (keystore, gatekeeper)
  - Key generation, storage and validation are all in secure world
  - Non secure world only gets “tokens”
  - Public keys accessible in non-secure world
  - Secret unlocking (e.g. Passwords) can be throttled or auto-wiped
- DRM - special case crypto hardware backing)
- Hardware backed entropy
  - PRNG code
- NFC (Android Pay)
- Kernel and boot chain integrity

Samsung uses of TrustZone

- TrustZone is a fundamental substrate for KNOX
  - Trusted Integrity Measurement Attestation (TIMA) provides
    - Client Certificate Management (CCM)
      - Extends keystore by hardware backing
    - Periodic Kernel Measurement (PKM)
      - Similar to iOS's KPP – periodically checks kernel page hashes
    - Realtime Kernel Protection (RKP)
      - Intercepts **events** from kernel using traps to secure monitor (SMC)

Implementation (AArch32)

- Implemented by a Secure Configuration Register (SCR)

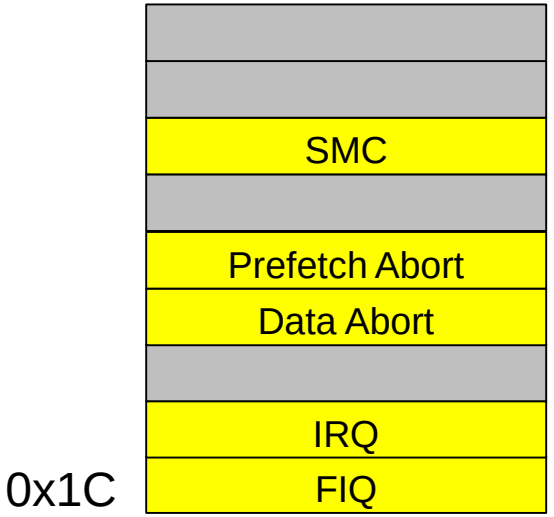


- NS = 1: not secure. NS = 0 not not secure ➔ secure
- SCR is co-processor CP15,c1
- Cannot be accessed in non-secure world:
  - Need SMC instruction to move to secure world first
- MMU enforces memory separation between worlds
  - http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0301h/Chdfjdgi.html
- Interrupts (IRQ/FIQ) can be handled by secure world

Entering TrustZone (AArch32)

- SMC to TrustZone is like SVC/SWI to supervisor mode

- Control transferred to a “monitor vector” in secure world



Voluntary Transition: SMC

- SMC only valid while *in* [super/hyper]visor mode
  - (i.e. requires the OS to be in kernel mode or higher)

C6.6.165 SMC

Secure Monitor Call causes an exception to EL3.

SMC is available only for software executing at EL1 or higher. It is UNDEFINED in EL0.

If the values of HCR_EL2.TSC and SCR_EL3.SMD are both 0, execution of an SMC instruction at EL1 or higher generates a Secure Monitor Call exception, using the EC value 0x17, that is taken to EL3. When EL3 is using AArch32, this exception is taken to Monitor mode.

If the value of HCR_EL2.TSC is 1, execution of an SMC instruction in a Non-secure EL1 state generates an exception that is taken to EL2, regardless of the value of SCR_EL3.SMD. When EL2 is using AArch32, this is a Hyp Trap exception that is taken to Hyp mode. For more information, see *Traps to EL2 of Non-secure EL1 execution of SMC instructions* on page D1-1506.

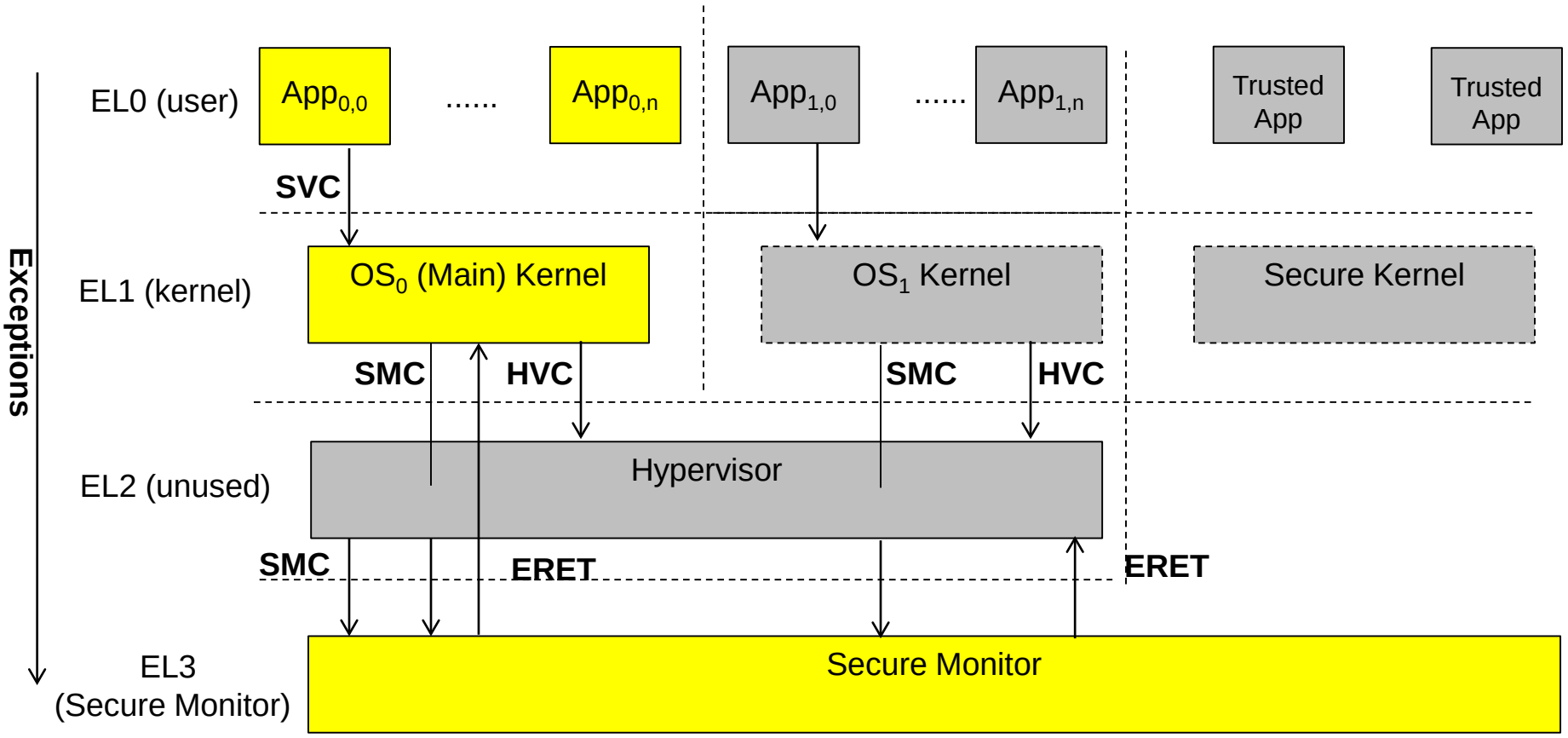
If the value of HCR_EL2.TSC is 0 and the value of SCR_EL3.SMD is 1, the SMC instruction is:

- UNDEFINED in Non-secure state.
- CONSTRAINED UNPREDICTABLE if executed in Secure state at EL1 or higher.

31 30 29 28	27 26 25 24	23 22 21 20				5 4	3 2 1 0
1 1 0 1	0 1 0 0	0 0 0	imm16			0 0	0 1 1

D 4 0 3

Recap: Exception Handling (AArch64)



(□ □ □ □ □ □ □ 术)

ELx state maintenance

- CPU maintains separate SP_ELx, and set of registers*

Register	Purpose
SCR_ELx	Secure Configuration Register
ESR_Elx	Exception Syndrome Register
VBAR_ELx	Vector Based Address Register
TTBRy_ELx	Translation Table Base
TCR_ELx	Translation Control Register
SCTLR_ELx	System Control Register
CPTR_ELx	Feature Trap register (FP, SIMD)
TPIDR_ELx	Thread Pointer ID
CPACR_Elx	Architectural Feature Control (FP,SIMD)

- Access to lower EL registers can be trapped in higher EL.

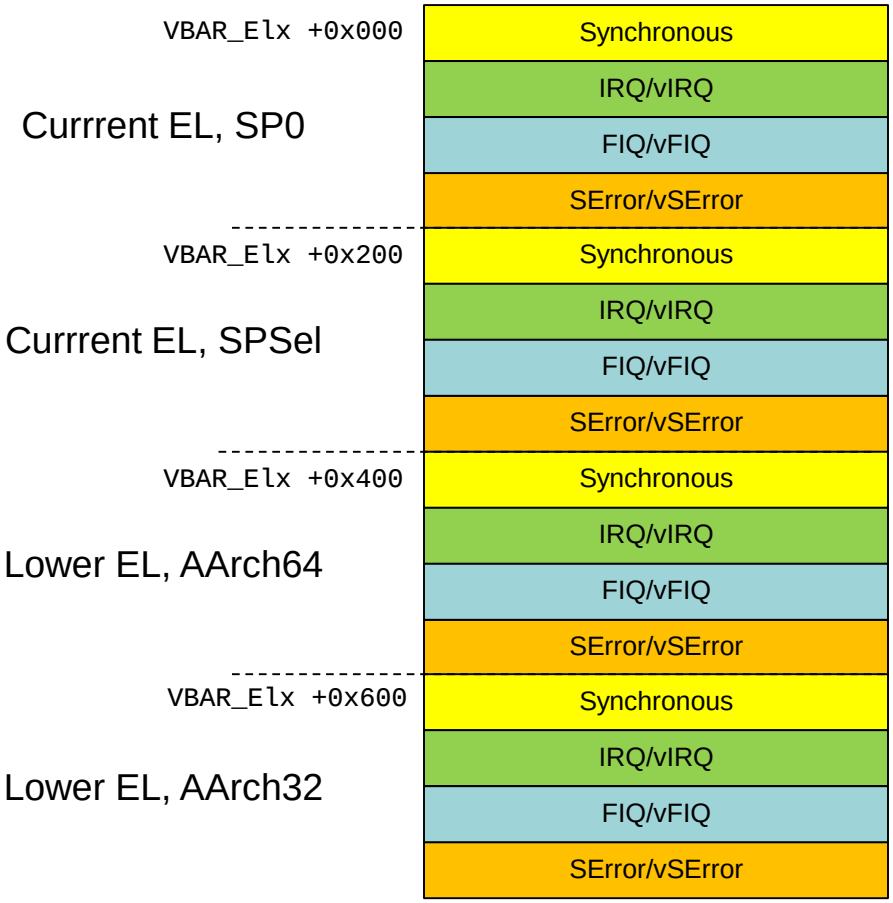
*** - Partial list**

Setting up Trustzone

- 32-bit:
  - CPU boots into secure world (NS=0)
  - Loader/kernel sets up monitor vector (SMC, IRQ or FIQ entries)
  - Sets up SCR NS=1 and “drops” to Normal World

- 64-bit:
  - CPU boots into EL3
  - Secure Monitor sets up VBAR_ELx (SError, IRQ or FIQ entries)
  - Drops to EL2 (Hypervisor) or EL1 (kernel)

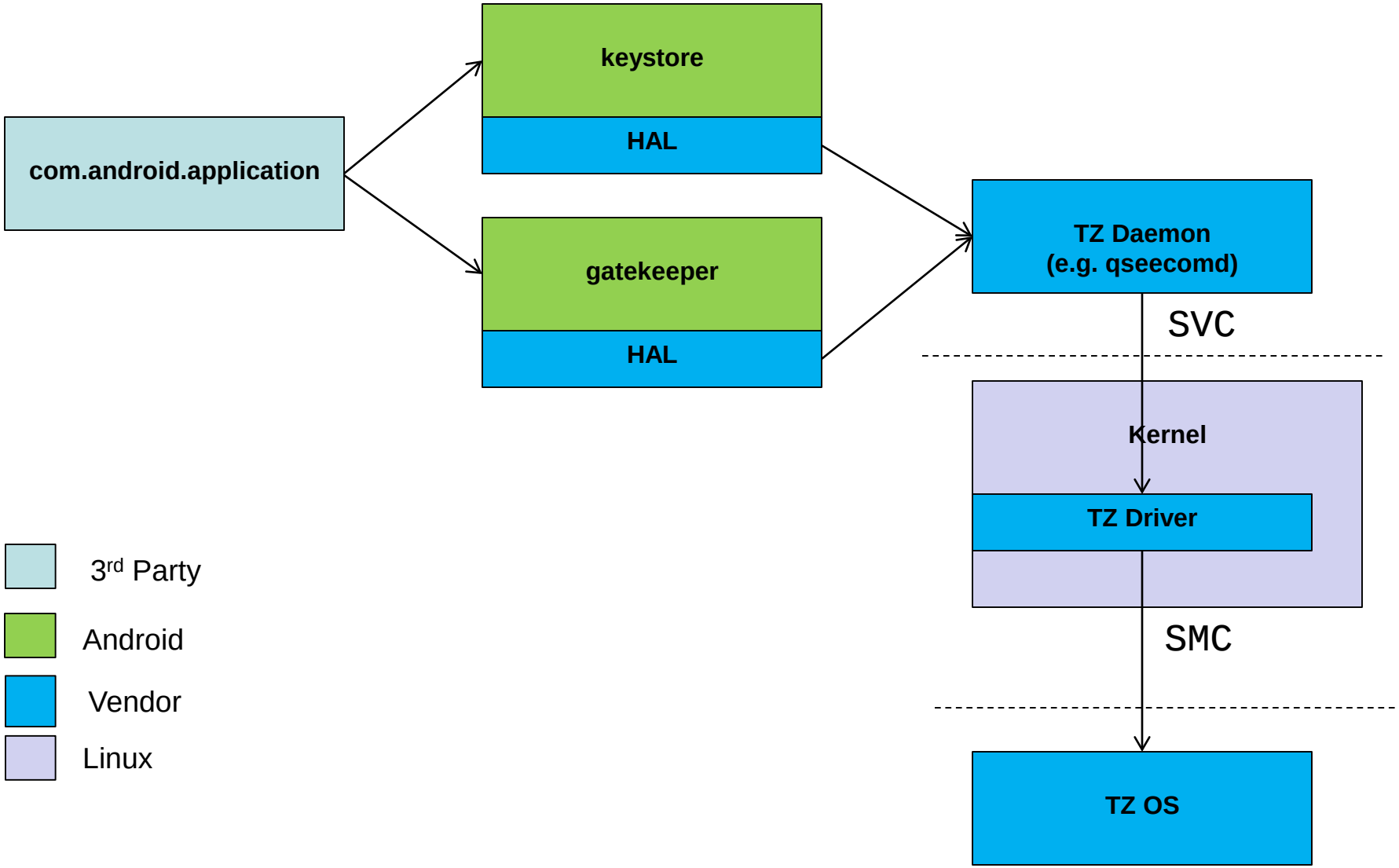
AArch64 Exception Handling



Android & TrustZone

- BootROM/SBL loads TZ image of “secure OS”
  - Usually in a TZ partition on flash
  - Backup (identical) usually also present
- Trustzone kernel usually an ELF image
  - Actual implementation is vendor-specific
  - Examples: Nvidia, Qualcomm
- Linux Kernel communicates with TZ kernel via driver
- Driver exports character device to user mode
- (Usually) dedicated daemon to communicate with kernel

Android & TrustZone



Android & TrustZone: NVidia

- NVidia (Nexus 9):

```
root@flounder: /#  ls -Ll /dev/block/platform/sdhci-tegra.3/by-name/
brw-------    1 root     root      259,  13 Nov 30 23:26 APP -> ..29 /system
brw-------    1 root     root      259,  14 Nov 30 23:26 CAC -> ..30 /cache
brw-rw----    1 system   system    259,   7 Nov 30 23:26 CDR -> ..23
brw-------    1 root     root      259,   4 Nov 30 23:26 DIA -> ..20
brw-------    1 root     root      179,   5 Nov 30 23:26 DTB -> ..5  (normally) Device Tree (but empty)
brw-rw----    1 system   system    259,   5 Nov 30 23:26 EF1 -> ..21
brw-rw----    1 system   system    259,   6 Nov 30 23:26 EF2 -> ..22
brw-------    1 root     root      179,   3 Nov 30 23:26 EKS -> ..3
brw-------    1 root     root      179,  11 Nov 30 23:26 EXT -> ..11
brw-------    1 root     root      179,  12 Nov 30 23:26 FST -> ..12
brw-------    1 root     root      259,  17 Nov 30 23:26 GPT -> ..33 GUID Partition Table (backup)
brw-------    1 root     root      179,   1 Nov 30 23:26 KEY -> ..1
brw-------    1 root     root      259,   0 Nov 30 23:26 LNX -> ..16  boot.img (with HTC wrap)
brw-------    1 root     root      259,   9 Dec  1 01:25 MD1 -> ..25
brw-------    1 root     root      259,  10 Nov 30 23:26 MD2 -> ..26
brw-------    1 root     root      259,   2 Nov 30 23:26 MFG -> ..18  Manufacturing Data
brw-------    1 root     root      259,   1 Nov 30 23:26 MSC -> ..17  Misc
brw-------    1 root     root      179,  10 Nov 30 23:26 NCT -> ..10
brw-------    1 root     root      259,  12 Nov 30 23:26 OTA -> ..28  OTA Updates
brw-------    1 root     root      179,  14 Nov 30 23:26 PG1 -> ..14
brw-------    1 system   system    259,  11 Dec  5 01:04 PST -> ..27 Persistent
brw-rw----    1 system   system    179,   8 Nov 30 23:26 RCA -> ..8
brw-------    1 root     root      179,   6 Nov 30 23:26 RV1 -> ..6  ?
brw-------    1 root     root      179,  13 Nov 30 23:26 RV2 -> ..13
brw-------    1 root     root      259,  16 Nov 30 23:26 RV3 -> ..32
brw-------    1 root     root      259,   3 Nov 30 23:26 SER -> ..19
brw-------    1 root     root      179,  15 Nov 30 23:26 SOS -> ..15 recovery.img (cute :-)
brw-------    1 root     root      179,   9 Nov 30 23:26 SP1 -> ..9
brw-------    1 root     root      179,   2 Nov 30 23:26 TOS -> ..2  ARM TrustZone
brw-------    1 root     root      259,  15 Nov 30 23:26 UDA -> ..31 User data (i.e /data)
brw-------    1 root     root      259,   8 Nov 30 23:26 VNR -> ..24 /vendor
brw-------    1 root     root      179,   4 Nov 30 23:26 WB0 -> ..4
brw-------    1 root     root      179,   7 Nov 30 23:26 WDM ->7
```

Android & TrustZone: Qualcomm

```
root@bullhead:/dev/block/platform/soc.0/f9824900.sdhci/by-name # ls -l
lrwxrwxrwx root     root              1970-07-18 00:51 DDR -> /dev/block/mmcblk0p28
lrwxrwxrwx root     root              1970-07-18 00:51 aboot -> /dev/block/mmcblk0p8
lrwxrwxrwx root     root              1970-07-18 00:51 abootbak -> /dev/block/mmcblk0p14
..
lrwxrwxrwx root     root              1970-07-18 00:51 boot -> /dev/block/mmcblk0p37
..
lrwxrwxrwx root     root              1970-07-18 00:51 keymaster -> /dev/block/mmcblk0p32
lrwxrwxrwx root     root              1970-07-18 00:51 keymasterbak -> /dev/block/mmcblk0p34
lrwxrwxrwx root     root              1970-07-18 00:51 keystore -> /dev/block/mmcblk0p44
...
llrwxrwxrwx root     root              1970-07-18 00:51 tz -> /dev/block/mmcblk0p4
lrwxrwxrwx root     root              1970-07-18 00:51 tzbak -> /dev/block/mmcblk0p11
lrwxrwxrwx root     root              1970-07-18 00:51 userdata -> /dev/block/mmcblk0p45
lrwxrwxrwx root     root              1970-07-18 00:51 vendor -> /dev/block/mmcblk0p39
root@bullhead:/dev/block/platform/soc.0/f9824900.sdhci/by-name # dd if=tz of=/data/local/tmp/tz
2048+0 records in
2048+0 records out
1048576 bytes transferred in 0.038 secs (27594105 bytes/sec)
```

Android & TrustZone: Samsung

```
root@s6# ls -l dev/block/platform/15570000.ufs/by-name
lrwxrwxrwx root     root              2016-05-27 08:53 BOOT -> /dev/block/sda5
lrwxrwxrwx root     root              2016-05-27 08:53 BOTA0 -> /dev/block/sda1
lrwxrwxrwx root     root              2016-05-27 08:53 BOTA1 -> /dev/block/sda2
lrwxrwxrwx root     root              2016-05-27 08:53 CACHE -> /dev/block/sda16
lrwxrwxrwx root     root              2016-05-27 08:53 DNT -> /dev/block/sda10
lrwxrwxrwx root     root              2016-05-27 08:53 EFS -> /dev/block/sda3
lrwxrwxrwx root     root              2016-05-27 08:53 HIDDEN -> /dev/block/sda17
lrwxrwxrwx root     root              2016-05-27 08:53 OTA -> /dev/block/sda7
lrwxrwxrwx root     root              2016-05-27 08:53 PARAM -> /dev/block/sda4
lrwxrwxrwx root     root              2016-05-27 08:53 PERSDATA -> /dev/block/sda13
lrwxrwxrwx root     root              2016-05-27 08:53 PERSISTENT -> /dev/block/sda11
lrwxrwxrwx root     root              2016-05-27 08:53 RADIO -> /dev/block/sda8
lrwxrwxrwx root     root              2016-05-27 08:53 RECOVERY -> /dev/block/sda6
lrwxrwxrwx root     root              2016-05-27 08:53 SBFS -> /dev/block/sda14
lrwxrwxrwx root     root              2016-05-27 08:53 STEADY -> /dev/block/sda12
lrwxrwxrwx root     root              2016-05-27 08:53 SYSTEM -> /dev/block/sda15
lrwxrwxrwx root     root              2016-05-27 08:53 TOMBSTONES -> /dev/block/sda9
lrwxrwxrwx root     root              2016-05-27 08:53 USERDATA -> /dev/block/sda18
root@s6# cat partitions | grep -v sda
major minor  #blocks  name
   7        0      32768 loop0
   8       16       4096 sdb       # Boot loader
   8       32       4096 sdc       # CryptoManager
 253        0    2097152 vnswap0
```

Have Image, will reverse

- From Secure World:
  - If you can get TZ image, start at `VBAR_EL3`
  - Find SMC/ handler (Synchronous)
  - Find IRQ/FIQ handlers
- From Non-Secure World:
  - Get kernel or bootloader
  - `disarm` and look for `SMC` calls

disarm

```
# disarm will automatically find strings when used as arguments
root@s6# JCOLOR=1 disarm /dev/sdb1 | less -R
...
0x0003fac4      0xd00002e0      ADRP X0, 94          ; X0 = 0x9d000
0x0003fac8      0x9112e000      ADD X0, X0, #1208    ; X0 = X0 + 0x4b8 = 0x9d4b8
0x0003facc      0x94001461      BL 0x44c50           ; = 0x44c50(" This is a non-secure chip. Skip...")
..
# So now we know 03fac4 is called on non-secure chip.. Search back using “?0x3fac4”
# disarm will attempt to auto guess the arguments to SMC as well
0x0003f9f4      0x12801de0      MOVN X0, #239
0x0003f9f8      0x52800001      MOVZ W1, 0x0
0x0003f9fc      0x2a1403e2      MOV X2, X20          ; X2 = X20 (0xf7120)
0x0003fa00      0xa9bf7bfd      STP X29, X30, [SP,#-16]!
0x0003fa04      0xd4000003      SMC #0               ; (X0=0xfffffffffffffff10, X1=0x0, X2=0xf7120..)
0x0003fa08      0xa8c17bfd      LDP X29, X30, [SP],#16
0x0003fa0c      0x3100041f      CMN W0, #1
0x0003fa10      0x2a0003e2      MOV X2, X0           ; X2 = X0 (?)
0x0003fa14      0x54000580      B.EQ 0x3fac4
# can also grep SMC
...
0x0004f014      0xd4000003      SMC #0 ; (X0=0xc2001014, X1=0x0, X2=0x22..)
0x0004f044      0xd4000003      SMC #0 ; (X0=0xc2001014, X1=0x0, X2=0x21..)
0x0004f098      0xd4000003      SMC #0 ; (X0=0xc2001014, X1=0x0, X2=0x20..)
0x0004f0c8      0xd4000003      SMC #0 ; (X0=0xc2001014, X1=0x0, X2=0x1f..)
...
```

Simple but effective ARM64 disassembler (http://NewAndroidBook.com/tools/disarm.html)

Trusty

- Google's attempt to standardize TEE Oses
  - https://source.android.com/security/trusty/index.html
  - Baked into Linux kernel tree: /drivers/trusty/
- Used by Nvidia
- Based on lk (similar to aboot) and provides:
  - gatekeeper, keymaster, NVRAM modules
  - Kernel driver
  - LK base
  - Trusty OS
- https://android-review.googlesource.com/#/admin/projects/?filter=trusty

Linux Kernel Support

- Generic TrustZone driver integrated into 3.10
- Qualcomm (msm) kernels have SCM driver
  - Secure Channel Manager
  - Creates a character device which `qseecomd` opens
- Driver issues SMC instructions, passes command buffers
  - Terrible buggy driver
  - Terrible buggy daemon
  - http://bits-please.blogspot.com/ - Step by step hack of QCOM TZ
    - Amazing exploit and explanation – Masterful hack, and a great read!

Android Vulnerabilities

CVE	Bug(s)	Severity	Updated versions	Date reported
CVE-2015-6639	ANDROID-24446875*	Critical	5.0, 5.1.1, 6.0, 6.0.1	Sep 23, 2015
CVE-2015-6647	ANDROID-24441554*	Critical	5.0, 5.1.1, 6.0, 6.0.1	Sep 27, 2015

CVE	Bug(s)	Severity	Updated versions	Date reported
CVE-2016-0825	ANDROID-20860039*	High	6.0.1	Google Internal

CVE	Android bugs	Severity	Updated Nexus devices	Date reported
CVE-2016-2431	24968809*	Critical	Nexus 5, Nexus 6, Nexus 7 (2013), Android One	Oct 15, 2015
CVE-2016-2432	25913059*	Critical	Nexus 6, Android One	Nov 28, 2015

- Thank you!
- Questions/comments welcome
  - Twitter: @Technologeeks – we consult & train on this stuff
  - Website: http://Technologeeks.com
  - Book: http://NewAndroidBook.com/