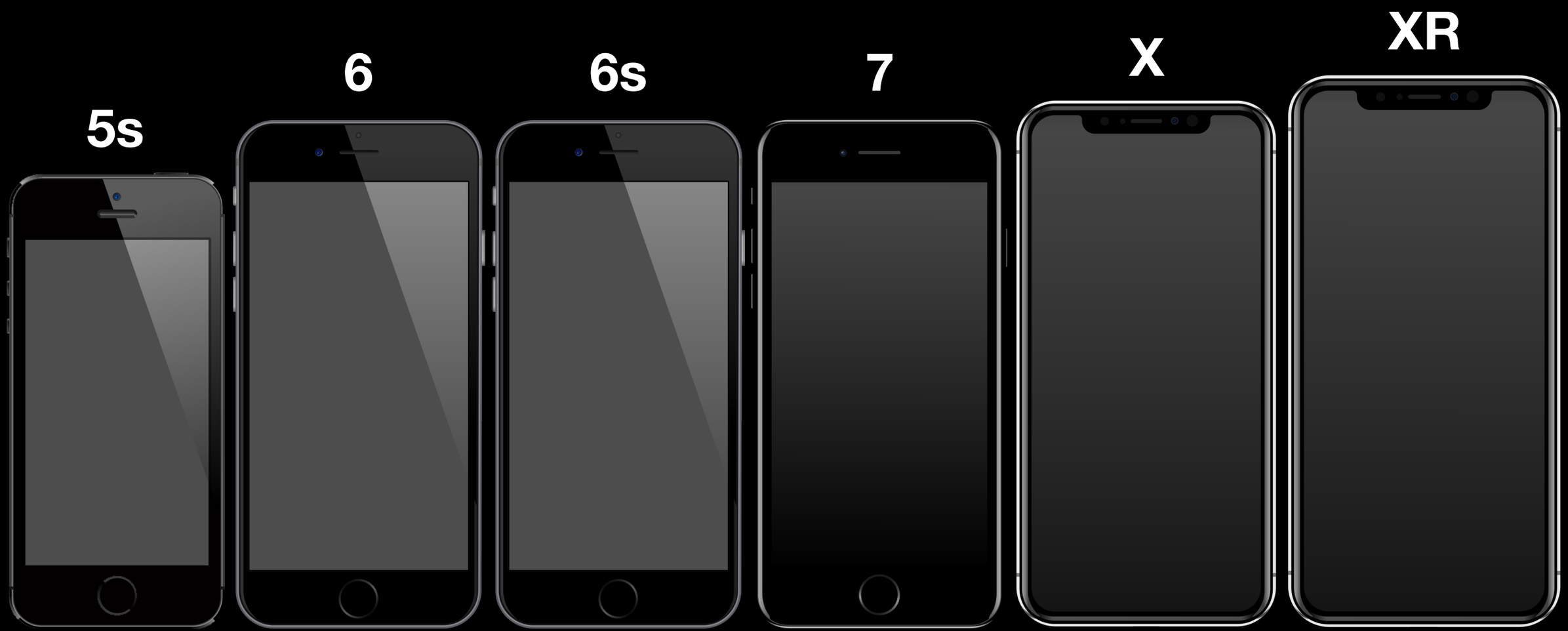
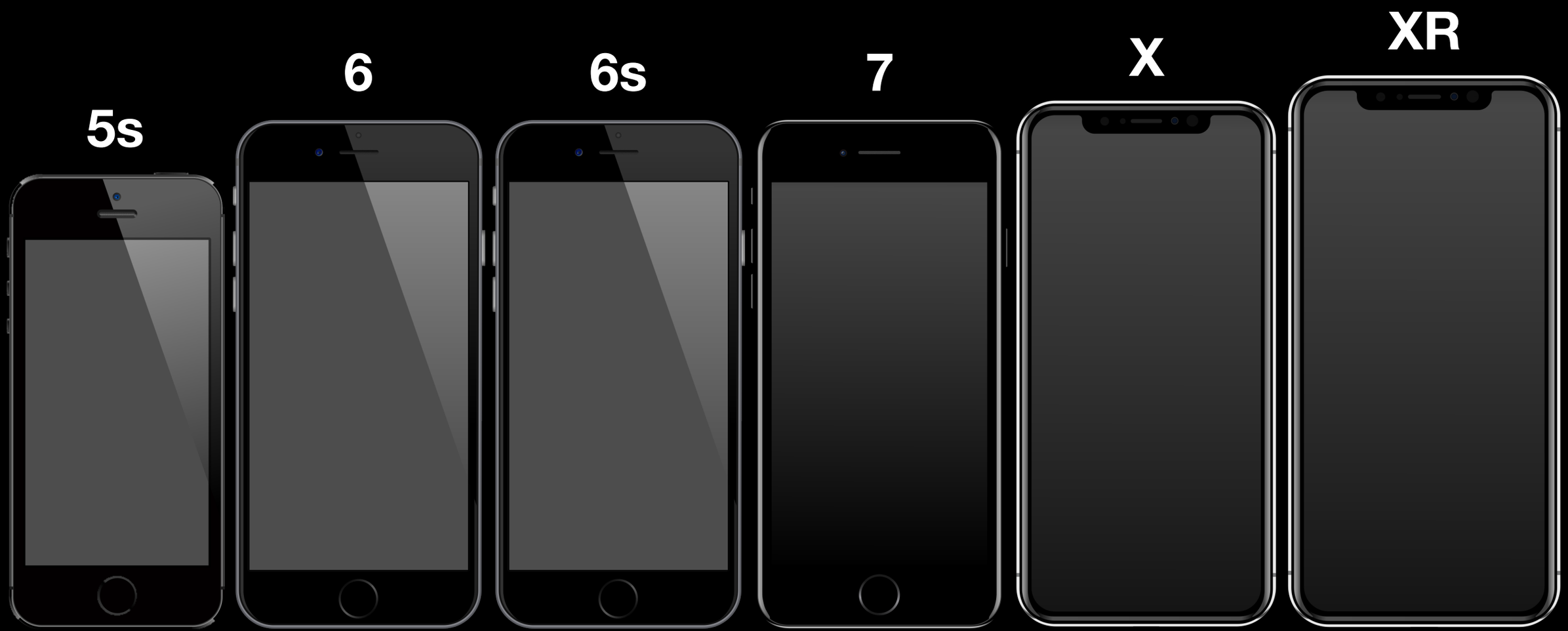
Evolution of iOS mitigations

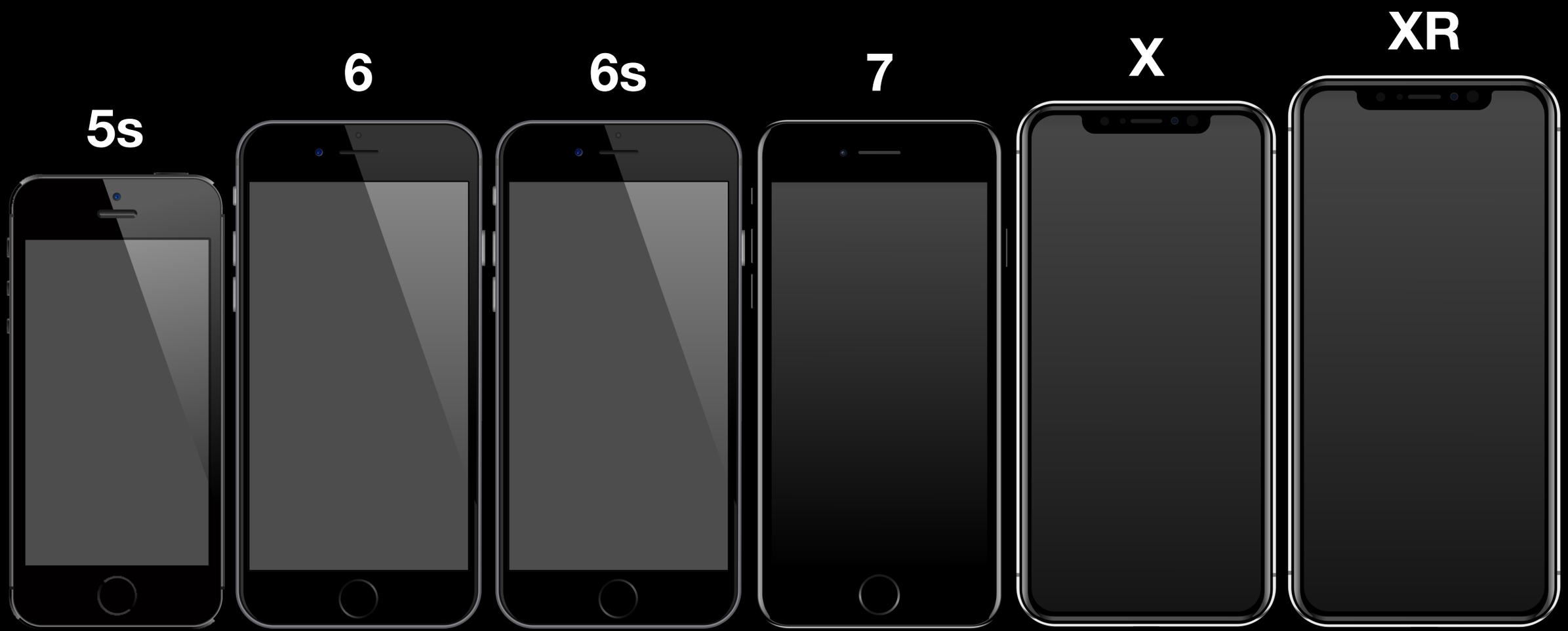
whoami

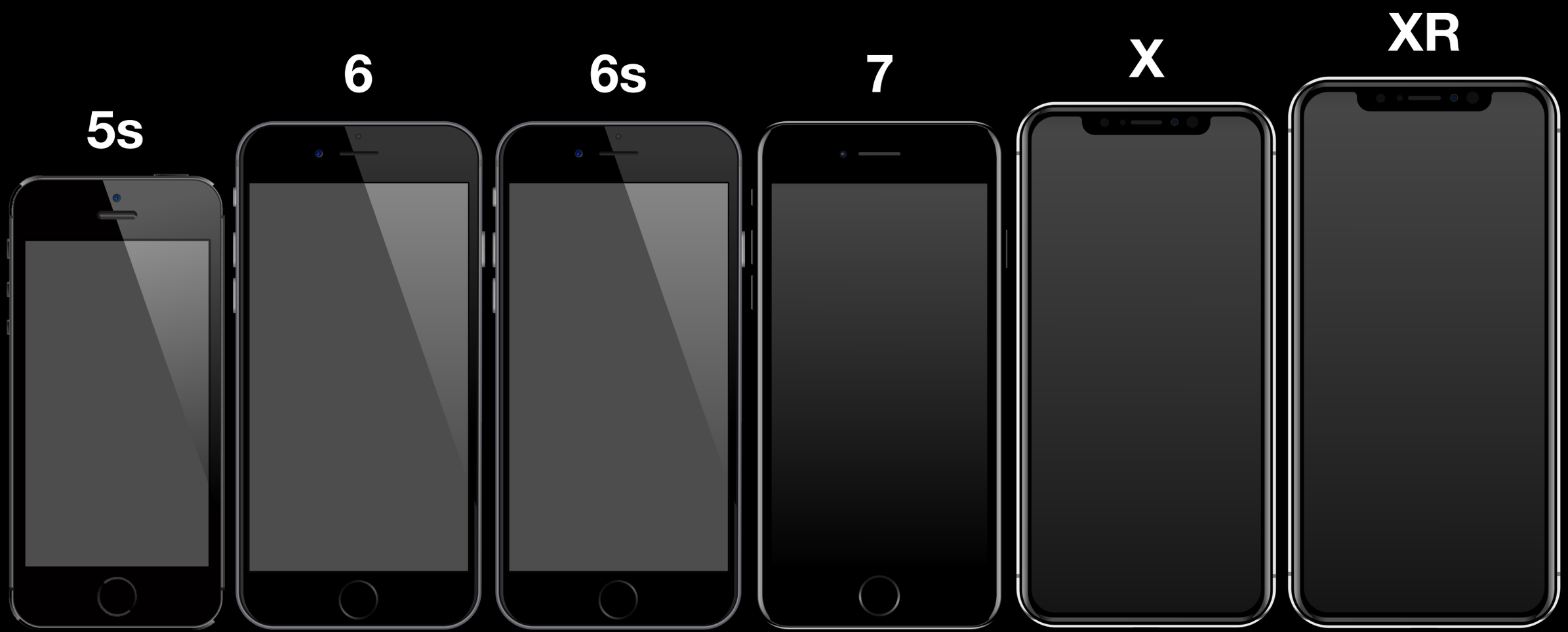
- Some guy from Switzerland
- Student, 24yo
- Enthusiast programmer for ages
- Kernel hacker since late 2016
- Member of the "Jake Blair" jailbreak team



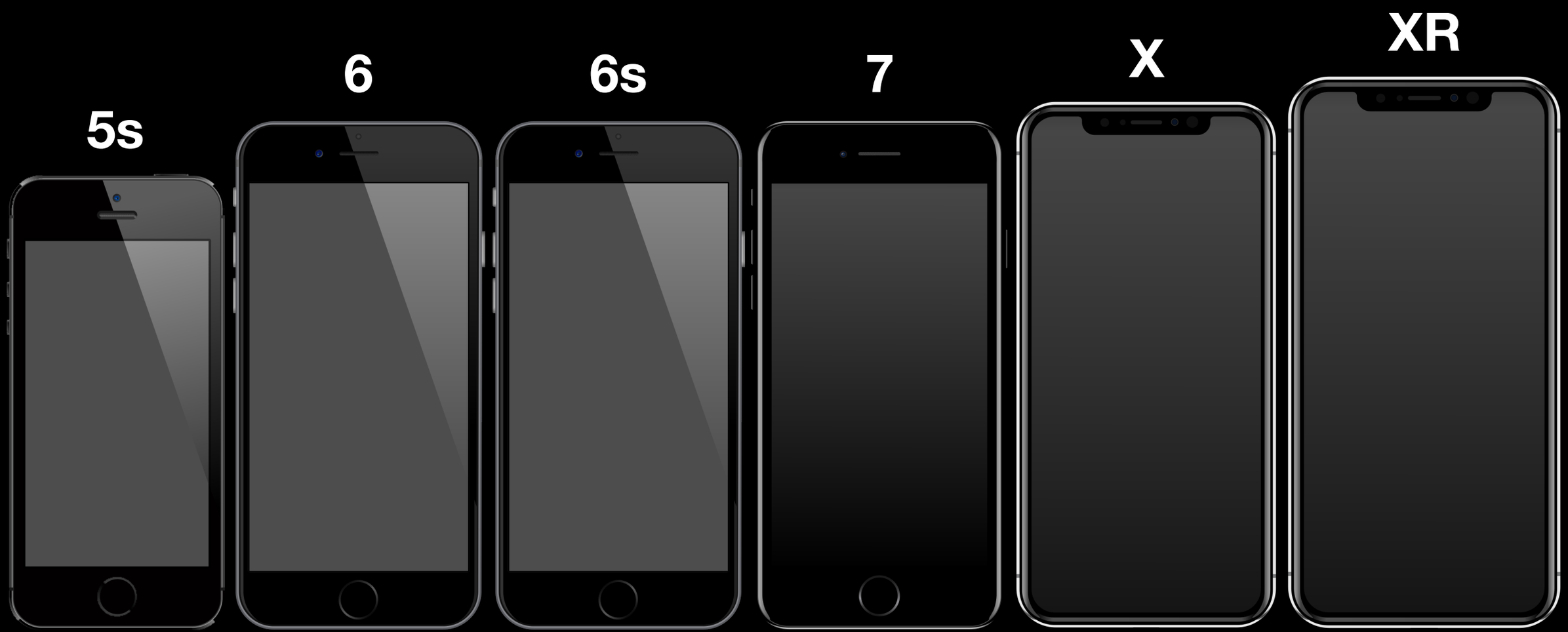
A7 A8 A9 A10 A11 A12


5s
6
6s
7
X
XR
A7
A8
A9
A10
A11
A12
ARMv8.0

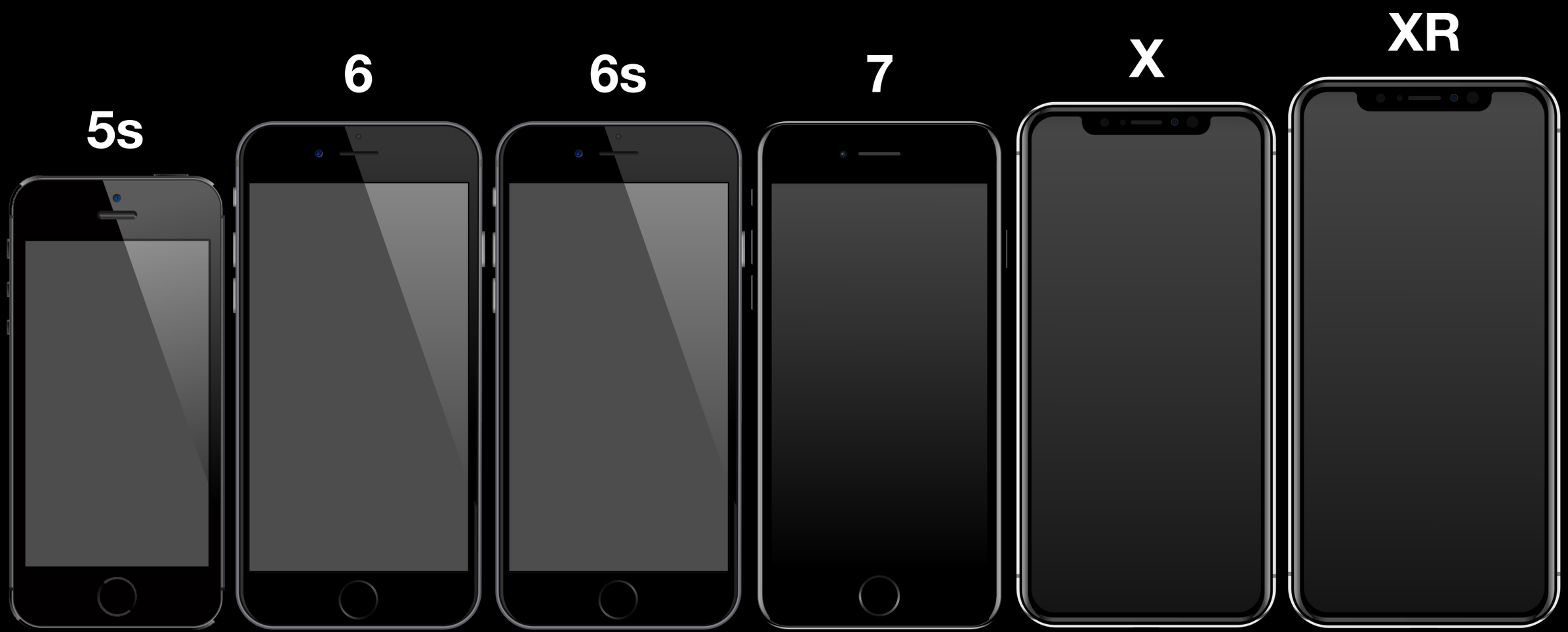

5s
6
6s
7
X
XR
A7
A8
A9
A10
A11
A12
ARMv8.0
ARMv8.1
ARMv8.2
ARMv8.3



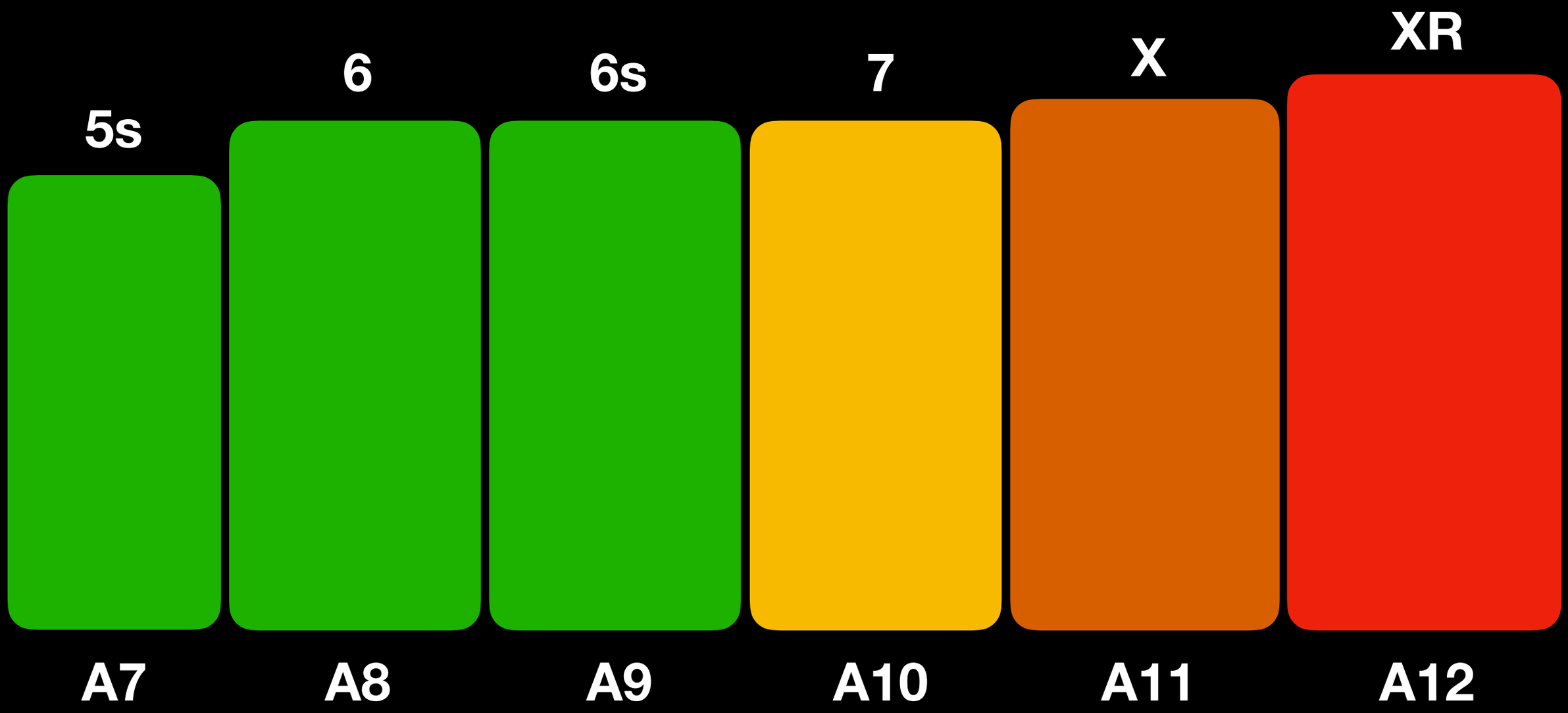
A7	A8	A9	A10	A11	A12
ARMv8.0			ARMv8.1	ARMv8.2	ARMv8.3
			PAN HPD	PAN HPD	PAN HPD PAC

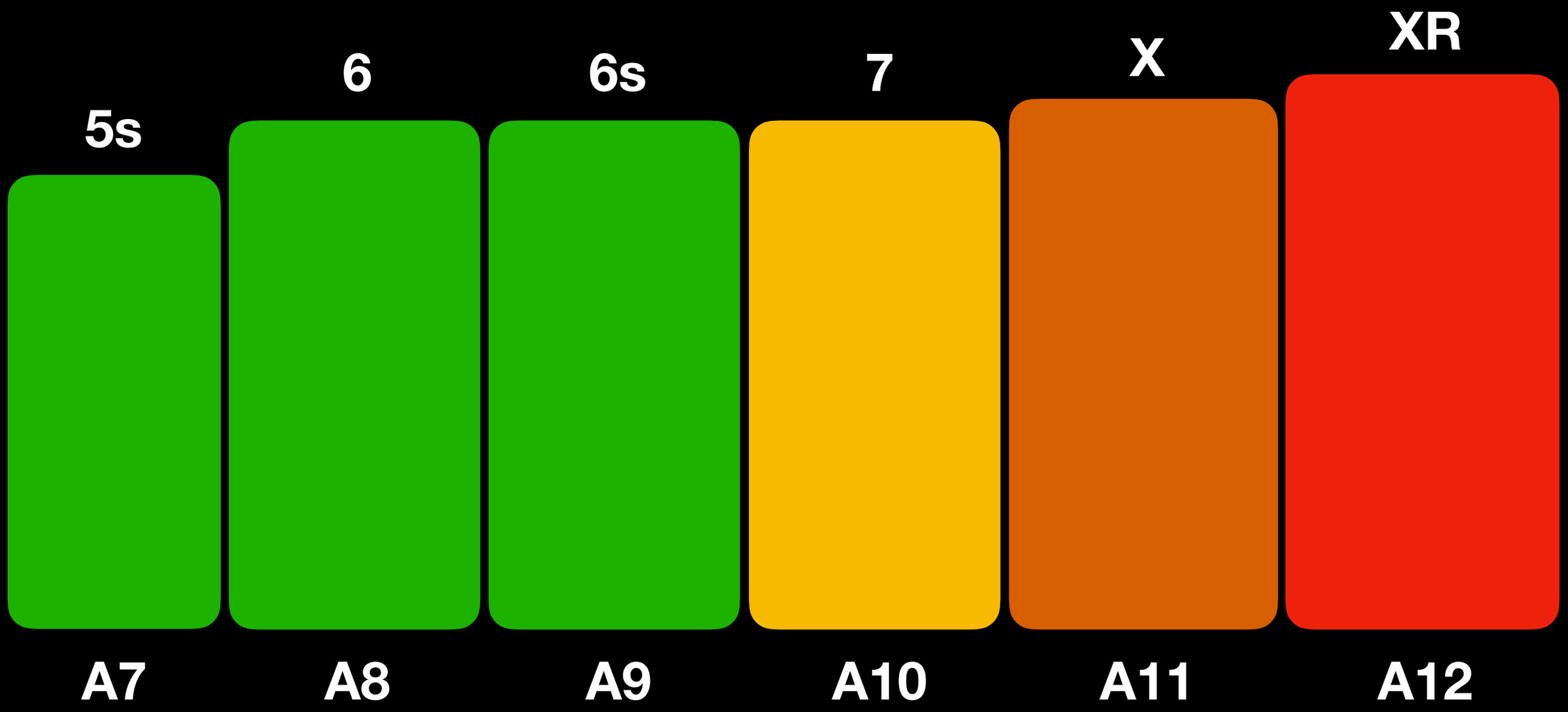


A7	A8	A9	A10	A11	A12
ARMv8.0			ARMv8.1	ARMv8.2	ARMv8.3
KPP			KTRR	KTRR APRR	KTRR APRR PPL

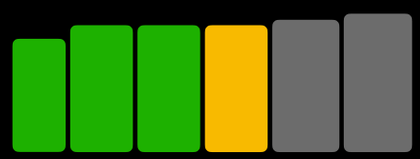


A7	A8	A9	A10	A11	A12
ARMv8.0			ARMv8.1	ARMv8.2	ARMv8.3
KPP			KTRR	KTRR APRR	KTRR APRR PPL
Bulletproof JIT		CoreTrust		vm_map_exec_lockdown	


5s
6
6s
7
X
XR
A7
A8
A9
A10
A11
A12

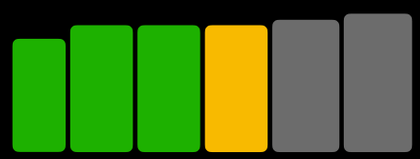

5s
6
6s
7
X
XR
A7
A8
A9
A10
A11
A12

Bulletproof JIT



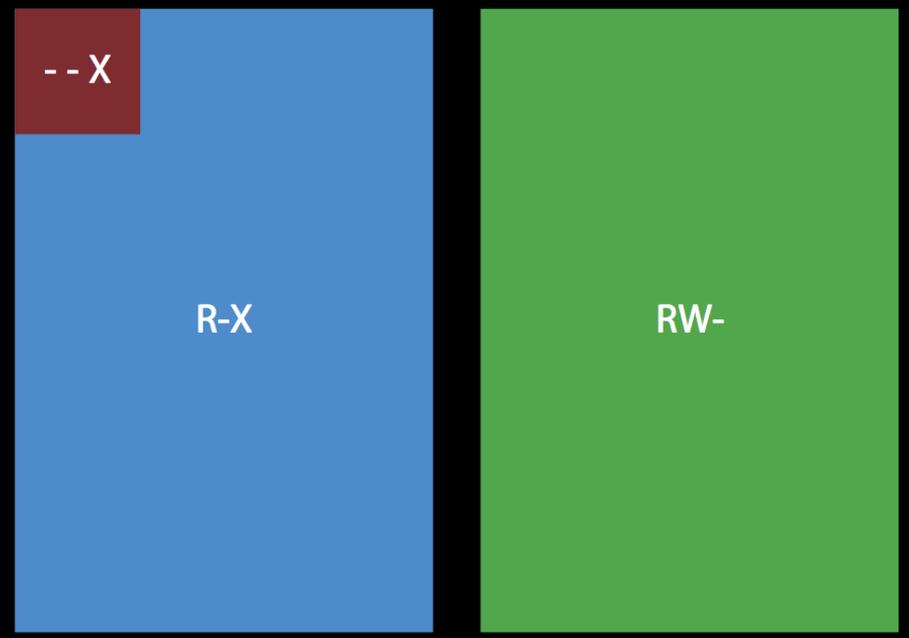
Bulletproof JIT

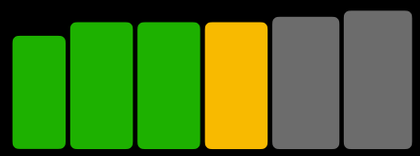
- Presented by Ivan Krstić at BHUS16



Bulletproof JIT

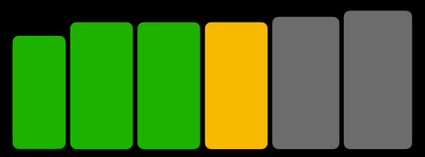
- Presented by Ivan Krstić at BHUS16



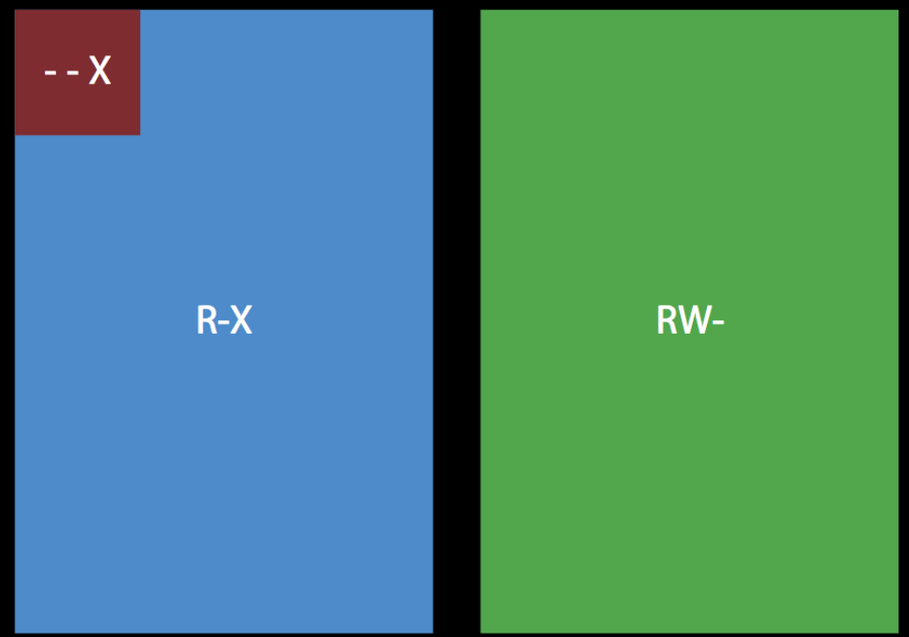


Bulletproof JIT

- Presented by Ivan Krstić at BHUS16
- Obviously futile without CFI

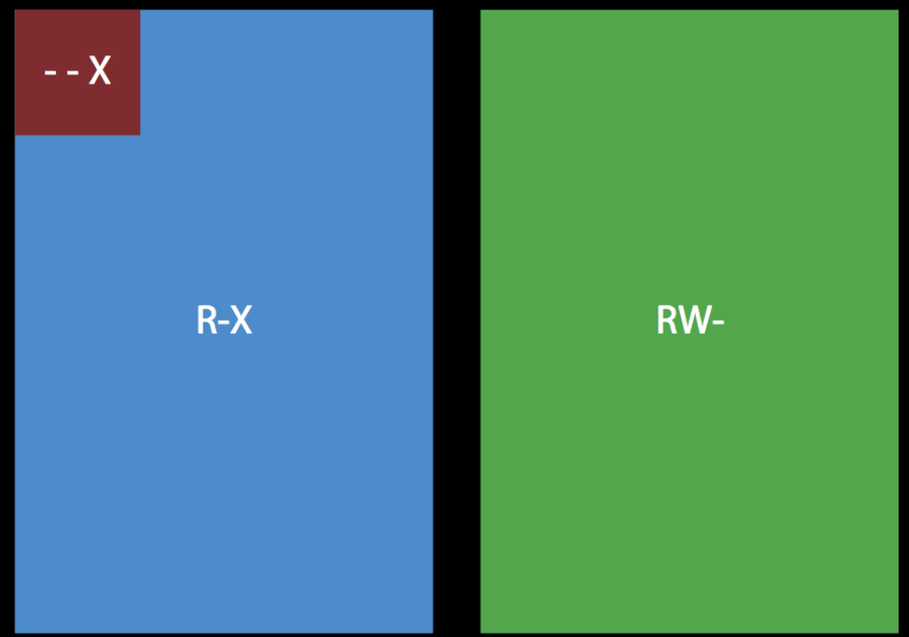


Bulletproof JIT



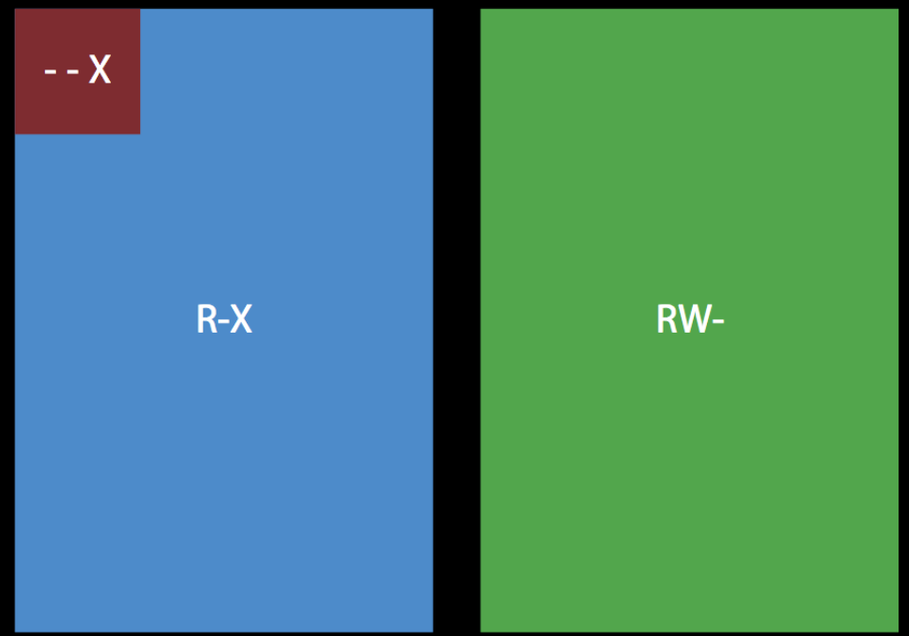
- Presented by Ivan Krstić at BHUS16
- Obviously futile without CFI
- An early indication that CFI was coming

Bulletproof JIT

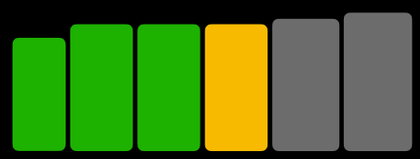


- Presented by Ivan Krstić at BHUS16
- Obviously futile without CFI
- An early indication that CFI was coming
- Breakable even with CFI

Bulletproof JIT



- Presented by Ivan Krstić at BHUS16
- Obviously futile without CFI
  - An early indication that CFI was coming
  - Breakable even with CFI
- Replaced by a new mechanism entirely on A11



KPP / "WatchTower"

KPP / "WatchTower"

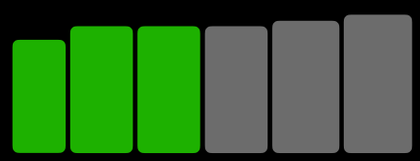
- Introduced in iOS 9 (arm64 only)

KPP / "WatchTower"

- Introduced in iOS 9 (arm64 only)
- Vastly imperfect (race, shellcode, ...)

KPP / "WatchTower"

- Introduced in iOS 9 (arm64 only)
- Vastly imperfect (race, shellcode, ...)
- Defeated forever by @qwertyoruiopz

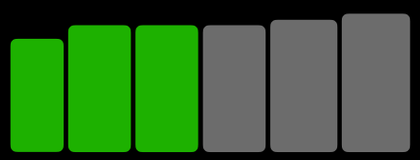


KPP / "WatchTower"

- Introduced in iOS 9 (arm64 only)
- Vastly imperfect (race, shellcode, ...)
- Defeated forever by @qwertyoruiopz
    - Good write-up by @xerub: https://xerub.github.io/ios/kpp/2017/04/13/tick-tock.html

KPP / "WatchTower"

- Introduced in iOS 9 (arm64 only)
- Vastly imperfect (race, shellcode, ...)
- Defeated forever by @qwertyoruiopz
  - Good write-up by @xerub: https://xerub.github.io/ios/kpp/2017/04/13/tick-tock.html
- Possibly more of a PoC?



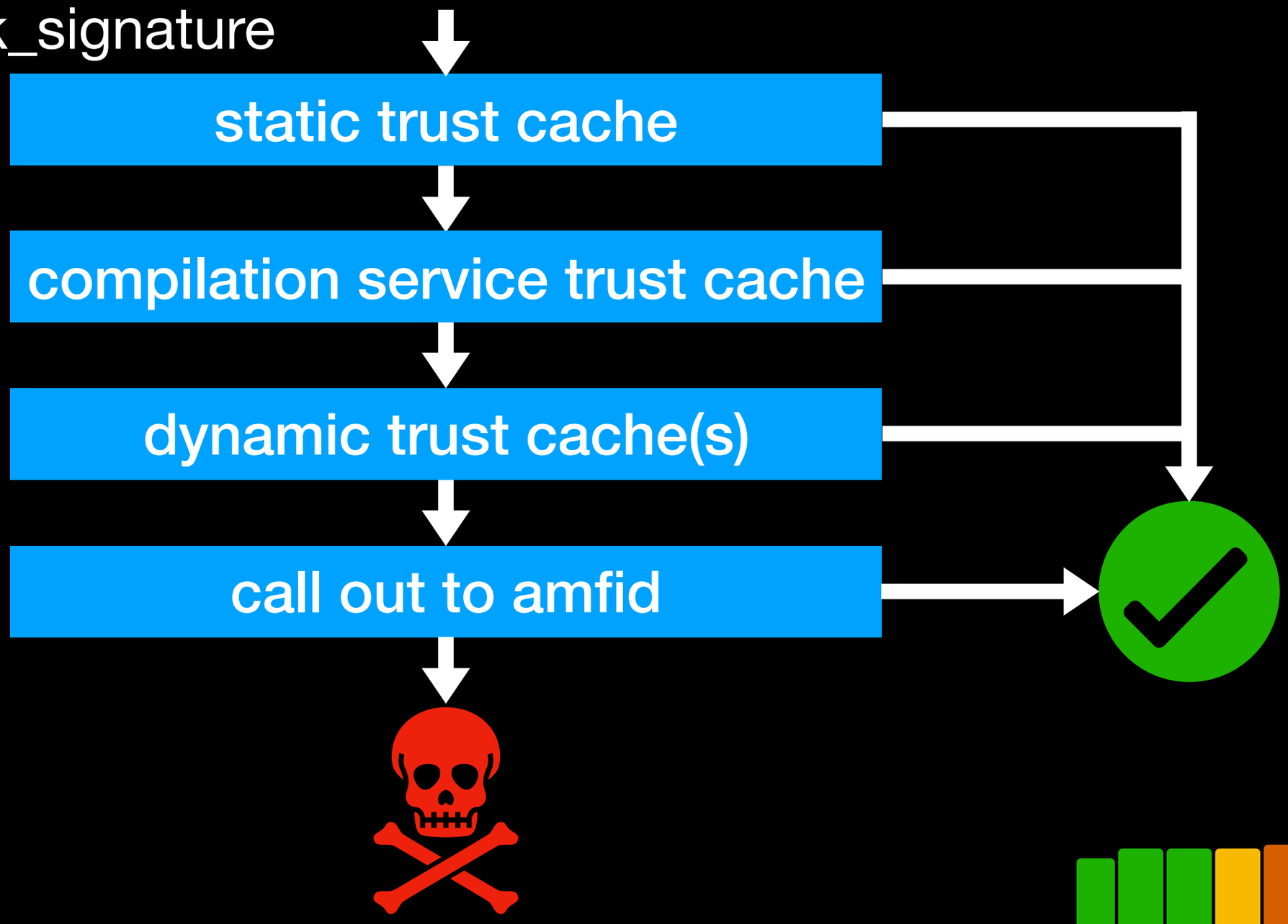
CoreTrust

- Exports a single function: CTEvaluateAMFICodeSignatureCMS

CoreTrust

- Exports a single function: CTEvaluateAMFICodeSignatureCMS

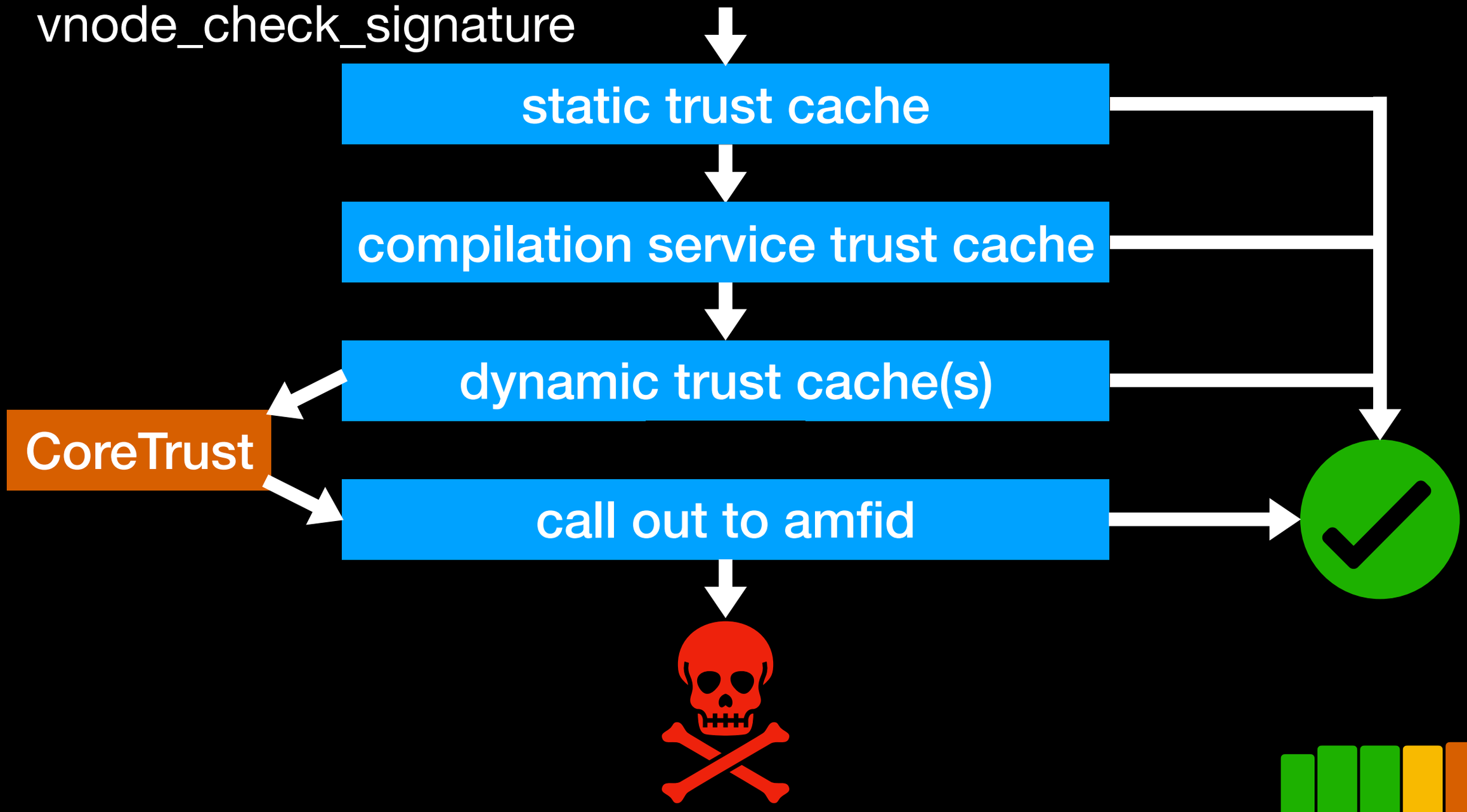
vnode_check_signature



CoreTrust

- Exports a single function: CTEvaluateAMFICodeSignatureCMS

vnode_check_signature

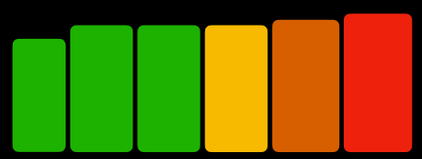


CoreTrust

- Exports a single function: CTEvaluateAMFICodeSignatureCMS

CoreTrust

- Exports a single function: CTEvaluateAMFICodeSignatureCMS
- Verifies that binary has an Apple-issued certificate

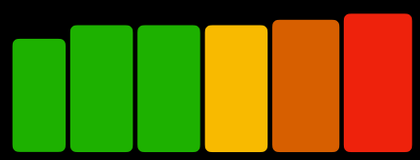


CoreTrust

- Exports a single function: CTEvaluateAMFICodeSignatureCMS
- Verifies that binary has an Apple-issued certificate
- Likely a response to jailbreaks patching amfid

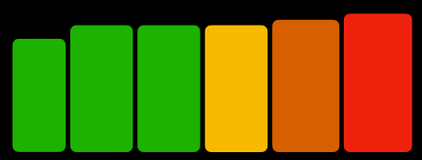
CoreTrust

- Exports a single function: CTEvaluateAMFICodeSignatureCMS
- Verifies that binary has an Apple-issued certificate
- Likely a response to jailbreaks patching amfid
- Bypassable in multiple ways



CoreTrust

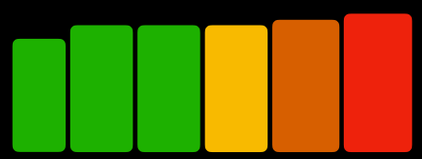
- Exports a single function: CTEvaluateAMFICodeSignatureCMS
- Verifies that binary has an Apple-issued certificate
- Likely a response to jailbreaks patching amfid
- Bypassable in multiple ways
- Again, more of a PoC?



vm_map_exec_lockdown

vm_map_exec_lockdown

- "Opt-out" of new executable mappings

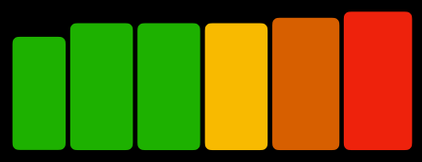


vm_map_exec_lockdown

- "Opt-out" of new executable mappings
- Little to no impact on exploitation

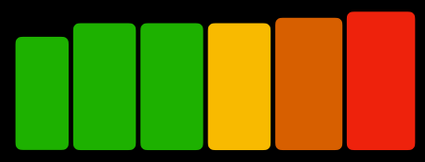
vm_map_exec_lockdown

- "Opt-out" of new executable mappings
- Little to no impact on exploitation
- Likely also against amfid patching



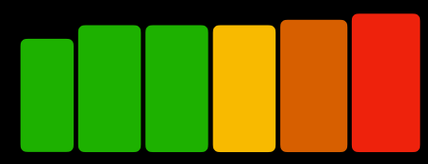
vm_map_exec_lockdown

- "Opt-out" of new executable mappings
- Little to no impact on exploitation
- Likely also against amfid patching
- Crumbles in the face of kernel r/w?



vm_map_exec_lockdown

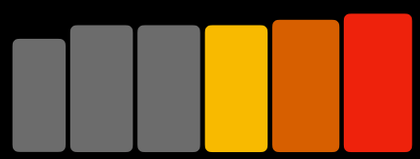
- "Opt-out" of new executable mappings
- Little to no impact on exploitation
- Likely also against amfid patching
- Crumbles in the face of kernel r/w?
- Probably just foreshadowing...



KTRR

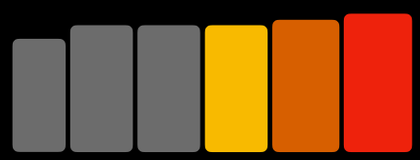
KTRR

- Kernel integrity backed by hardware primitives



KTRR

- Kernel integrity backed by hardware primitives
  - No writes to const, ever

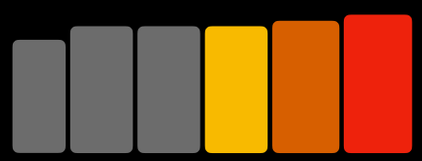


KTRR

- Kernel integrity backed by hardware primitives
  - No writes to const, ever
  - No new code, ever

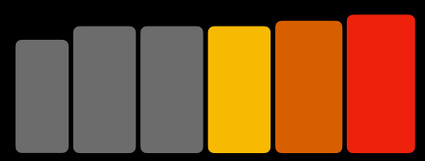
KTRR

- Kernel integrity backed by hardware primitives
  - No writes to const, ever
  - No new code, ever
  - "Dangerous" instructions moved to special segment



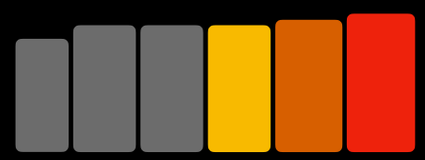
KTRR

- Kernel integrity backed by hardware primitives
  - No writes to const, ever
  - No new code, ever
  - "Dangerous" instructions moved to special segment
- Bypassable to some extent (YaluX, Ian Beer's debugger)



KTRR

- Kernel integrity backed by hardware primitives
  - No writes to const, ever
  - No new code, ever
  - "Dangerous" instructions moved to special segment
- Bypassable to some extent (YaluX, Ian Beer's debugger)
- Did a write-up last year: https://siguza.github.io/KTRR/



KTRR

- New register lockdown in A12

KTRR

- New register lockdown in A12

```
0xfffffff0079d8bfc      df3f03d5        isb
0xfffffff0079d8c00      0100f0d2        mov x1, -0x8000000000000000
0xfffffff0079d8c04      a00280d2        mov x0, 0x15
0xfffffff0079d8c08      000001aa        orr x0, x0, x1
0xfffffff0079d8c0c      40f11cd5        msr s3_4_c15_c1_2, x0
0xfffffff0079d8c10      df3f03d5        isb
0xfffffff0079d8c14      c0035fd6        ret
```

KTRR

- New register lockdown in A12

```
0xfffffff0079d8bfc      df3f03d5        isb
0xfffffff0079d8c00      0100f0d2        mov x1, -0x8000000000000000
0xfffffff0079d8c04      a00280d2        mov x0, 0x15
0xfffffff0079d8c08      000001aa        orr x0, x0, x1
0xfffffff0079d8c0c      40f11cd5        msr s3_4_c15_c1_2, x0
0xfffffff0079d8c10      df3f03d5        isb
0xfffffff0079d8c14      c0035fd6        ret

0xfffffff0079d8c24      41f13cd5        mrs x1, s3_4_c15_c1_2
0xfffffff0079d8c28      21007e92        and x1, x1, 4
0xfffffff0079d8c2c      410100b5        cbnz x1, 0xfffffff0079d8c54
0xfffffff0079d8c30      402018d5        msr tcr_el1, x0
0xfffffff0079d8c34      df3f03d5        isb

0xfffffff0079d8bd4      41f13cd5        mrs x1, s3_4_c15_c1_2
0xfffffff0079d8bd8      21007c92        and x1, x1, 0x10
0xfffffff0079d8bdc      c10300b5        cbnz x1, 0xfffffff0079d8c54
0xfffffff0079d8be0      202018d5        msr ttbr1_el1, x0
0xfffffff0079d8be4      df3f03d5        isb
0xfffffff0079d8be8      c0035fd6        ret
```

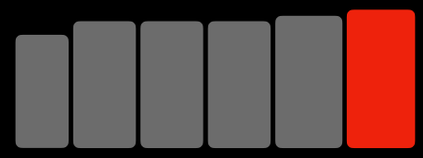
KTRR

- New register lockdown in A12

```
panic(cpu 0 caller 0xffffffff01dd79b84): "Undefined kernel instruction: pc=0xffffffff01dbd8084 instr=d518c000\n"
Debugger message: panic
Memory ID: 0xff
OS version: 16A405
Kernel version: Darwin Kernel Version 18.0.0: Tue Aug 14 22:07:18 PDT 2018; root:xnu-4903.202.2~1/RELEASE_ARM64_T8020
Kernel UUID: BEFBC911-B1BC-3553-B7EA-1ECE60169886
iBoot version: iBoot-4513.200.297
secure boot?: YES
Paniclog version: 10
Kernel slide:     0x0000000016200000
Kernel text base: 0xffffffff01d204000
Epoch Time:        sec       usec
  Boot    : 0x5cc4e1ec 0x000c74d9
  Sleep   : 0x00000000 0x00000000
  Wake    : 0x00000000 0x00000000
  Calendar: 0x5cc4e21d 0x000d3015
```

KTRR

- New register lockdown in A12
  - VBAR_EL1
  - TCR_EL1
  - TTBR1_EL1
  - Part of SCTLR_EL1



PAC

PAC

- The "big scary" game changer of A12

PAC

- The "big scary" game changer of A12
- Part of the ARMv8.3 spec...

PAC

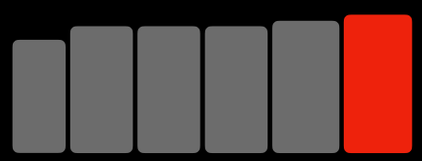
- The "big scary" game changer of A12
- Part of the ARMv8.3 spec...
- ...but greatly augmented by Apple

PAC

- The "big scary" game changer of A12
- Part of the ARMv8.3 spec...
  - ...but greatly augmented by Apple
- A PITA, but hard to get right after all

PAC

- The "big scary" game changer of A12
- Part of the ARMv8.3 spec...
  - ...but greatly augmented by Apple
- A PITA, but hard to get right after all
- Detailed analysis by Brandon Azad: https://googleprojectzero.blogspot.com/2019/02/examining-pointer-authentication-on.html



PPL

- The *real* game changer of A12

PPL

- The *real* game changer of A12

```
0xfffffff0194b4000 -> [0x8074b4000 r-x/--x] __PPLTEXT
0xfffffff0194b8000 -> [0x8074b8000 r-x/--x] __PPLTEXT
0xfffffff0194bc000 -> [0x8074bc000 r-x/--x] __PPLTEXT
0xfffffff0194c0000 -> [0x8074c0000 r-x/--x] __PPLTEXT
0xfffffff0194c4000 -> [0x8074c4000 r-x/--x] __PPLTEXT
0xfffffff0194c8000 -> [0x8074c8000 r-x/---] __PPLTRAMP
0xfffffff0194cc000 -> [0x8074cc000 r-x/--x] __PPLTRAMP
0xfffffff0194d0000 -> [0x8074d0000 r-x/--x] __PPLTRAMP
0xfffffff0194d4000 -> [0x8074d4000 r-x/---] __PPLTRAMP
0xfffffff0194d8000 -> [0x8074d8000 r--/--x] __PPLDATA_CONST
0xfffffff0194dc000 -> [0x8074dc000 r--/---] __LAST
0xfffffff0194e0000 -> [0x8074e0000 rw-/--x] __PPLDATA
```

PPL

- The *real* game changer of A12

```
0xfffffff0194b4000 -> [0x8074b4000 r--/---] __PPLTEXT
0xfffffff0194b8000 -> [0x8074b8000 r--/---] __PPLTEXT
0xfffffff0194bc000 -> [0x8074bc000 r--/---] __PPLTEXT
0xfffffff0194c0000 -> [0x8074c0000 r--/---] __PPLTEXT
0xfffffff0194c4000 -> [0x8074c4000 r--/---] __PPLTEXT
0xfffffff0194c8000 -> [0x8074c8000 r-x/---] __PPLTRAMP
0xfffffff0194cc000 -> [0x8074cc000 r--/---] __PPLTRAMP
0xfffffff0194d0000 -> [0x8074d0000 r--/---] __PPLTRAMP
0xfffffff0194d4000 -> [0x8074d4000 r-x/---] __PPLTRAMP
0xfffffff0194d8000 -> [0x8074d8000 r--/--x] __PPLDATA_CONST
0xfffffff0194dc000 -> [0x8074dc000 r--/---] __LAST
0xfffffff0194e0000 -> [0x8074e0000 r--/---] __PPLDATA
```

PPL

- The *real* game changer of A12

```
0xfffffff0194b4000 -> [0x8074b4000 r-x/---] __PPLTEXT
0xfffffff0194b8000 -> [0x8074b8000 r-x/---] __PPLTEXT
0xfffffff0194bc000 -> [0x8074bc000 r-x/---] __PPLTEXT
0xfffffff0194c0000 -> [0x8074c0000 r-x/---] __PPLTEXT
0xfffffff0194c4000 -> [0x8074c4000 r-x/---] __PPLTEXT
0xfffffff0194c8000 -> [0x8074c8000 r-x/---] __PPLTRAMP
0xfffffff0194cc000 -> [0x8074cc000 r-x/---] __PPLTRAMP
0xfffffff0194d0000 -> [0x8074d0000 r-x/---] __PPLTRAMP
0xfffffff0194d4000 -> [0x8074d4000 r-x/---] __PPLTRAMP
0xfffffff0194d8000 -> [0x8074d8000 r--/--x] __PPLDATA_CONST
0xfffffff0194dc000 -> [0x8074dc000 r--/---] __LAST
0xfffffff0194e0000 -> [0x8074e0000 rw-/---] __PPLDATA
```

PPL

- The *real* game changer of A12

```
0xfffffff008ecbfe0      34423bd5        mrs x20, daif
0xfffffff008ecbfe4      df4703d5        msr daifset, 7
0xfffffff008ecbfe8      ae8ae8f2        movk x14, 0x4455, lsl 48
0xfffffff008ecbfec      ae8ac8f2        movk x14, 0x4455, lsl 32
0xfffffff008ecbff0      ce8cacf2        movk x14, 0x6466, lsl 16
0xfffffff008ecbff4      eece8cf2        movk x14, 0x6677
0xfffffff008ecbff8      2ef21cd5        msr s3_4_c15_c2_1, x14
0xfffffff008ecbffc      df3f03d5        isb
0xfffffff008ecc000      df4703d5        msr daifset, 7
0xfffffff008ecc004      ae8ae8f2        movk x14, 0x4455, lsl 48
0xfffffff008ecc008      ae8ac8f2        movk x14, 0x4455, lsl 32
0xfffffff008ecc00c      ce8cacf2        movk x14, 0x6466, lsl 16
0xfffffff008ecc010      eece8cf2        movk x14, 0x6677
0xfffffff008ecc014      35f23cd5        mrs x21, s3_4_c15_c2_1
0xfffffff008ecc018      df0115eb        cmp x14, x21
0xfffffff008ecc01c      e1050054        b.ne 0xfffffff008ecc0d8
```

PPL

- The *real* game changer of A12
  - Some "magic value"

outside PPL	0x4455445464666477
inside PPL	0x4455445564666677

PPL

- The *real* game changer of A12
  - Some "magic value"

outside PPL	0x4455445464666477
inside PPL	0x4455445564666677

PPL

- The *real* game changer of A12
  - Some "magic value"

outside PPL	0x4455445464666477
inside PPL	0x4455445564666677

- Protected by page boundaries
- Checked in reset handlers

PPL

- The *real* game changer of A12

PPL

- The *real* game changer of A12
  - Introduces a "privileged" CPU mode

PPL

- The *real* game changer of A12
  - Introduces a "privileged" CPU mode
  - Can only be entered via trampoline

PPL

- The *real* game changer of A12
  - Introduces a "privileged" CPU mode
  - Can only be entered via trampoline
  - Certain pages not executable/writeable outside that mode

PPL

- The *real* game changer of A12
  - Introduces a "privileged" CPU mode
  - Can only be entered via trampoline
  - Certain pages not executable/writeable outside that mode
    - Used to protect system registers, page tables, MMIO access, trust cache, ...

PPL

- The *real* game changer of A12
  - Introduces a "privileged" CPU mode
  - Can only be entered via trampoline
  - Certain pages not executable/writeable outside that mode
    - Used to protect system registers, page tables, MMIO access, trust cache, ...
  - Holds even in the face of kernel r/w and PAC defeat

PPL

- The *real* game changer of A12
  - Introduces a "privileged" CPU mode
  - Can only be entered via trampoline
  - Certain pages not executable/writeable outside that mode
    - Used to protect system registers, page tables, MMIO access, trust cache, ...
  - Holds even in the face of kernel r/w and PAC defeat
  - Somehow related to the UXN bit (userland --x)

PPL

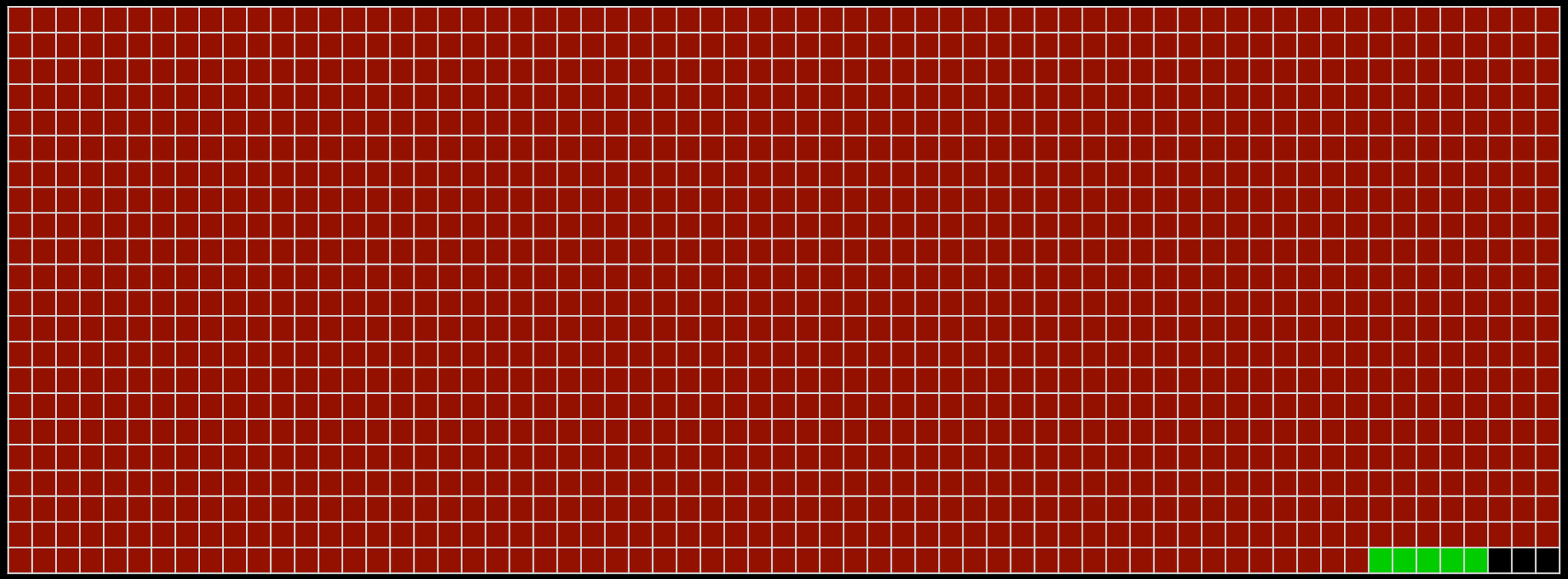

Siguza
@s1guza
Wild idea for if/when ROP finally becomes infeasible due to pointer authentication, gadget removal/hardening and whatnot:
11:19 am - 19 Oct 2017
Siguza
@s1guza
Spray kernel mem with page tables and attack them instead, gives you arbitrary r/w. Then map the kernel binary into your address space...
11:19 am - 19 Oct 2017

PPL

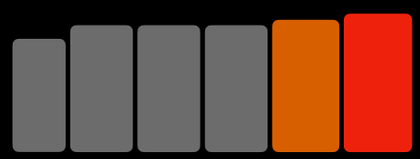
- 1357 pages in __TEXT_EXEC
- 5 pages in __PPLTEXT

PPL

- 1357 pages in __TEXT_EXEC
- 5 pages in __PPLTEXT



"New" JIT



"New" JIT

- Replaces split JIT region with unified RWX region again

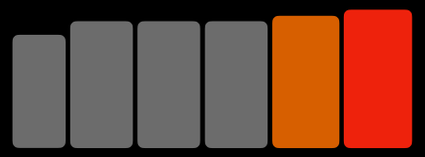
"New" JIT

- Replaces split JIT region with unified RWX region again
- Custom system register controls whether it's r-x or rw-

"New" JIT

- Replaces split JIT region with unified RWX region again
- Custom system register controls whether it's r-x or rw-

```
0x188347298     002298f2        movk x0, 0xc110
0x18834729c     e0ffbff2        movk x0, 0xffff, lsl 16
0x1883472a0     e001c0f2        movk x0, 0xf, lsl 32
0x1883472a4     0000e0f2        movk x0, 0, lsl 48
0x1883472a8     000040f9        ldr x0, [x0]
0x1883472ac     e0f21cd5        msr s3_4_c15_c2_7, x0
0x1883472b0     df3f03d5        isb
0x1883472b4     012298f2        movk x1, 0xc110
0x1883472b8     e1ffbff2        movk x1, 0xffff, lsl 16
0x1883472bc     e101c0f2        movk x1, 0xf, lsl 32
0x1883472c0     0100e0f2        movk x1, 0, lsl 48
0x1883472c4     280040f9        ldr x8, [x1]
0x1883472c8     e9f23cd5        mrs x9, s3_4_c15_c2_7
0x1883472cc     1f0109eb        cmp x8, x9
0x1883472d0     c1020054        b.ne 0x188347328
```

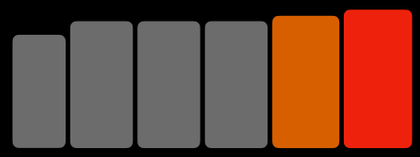


"New" JIT

- Replaces split JIT region with unified RWX region again
- Custom system register controls whether it's r-x or rw-

```
0x188347298     002298f2        movk x0, 0xc110
0x18834729c     e0ffbff2        movk x0, 0xffff, lsl 16
0x1883472a0     e001c0f2        movk x0, 0xf, lsl 32
0x1883472a4     0000e0f2        movk x0, 0, lsl 48
0x1883472a8     000040f9        ldr x0, [x0]
0x1883472ac     e0f21cd5        msr s3_4_c15_c2_7, x0
0x1883472b0     df3f03d5        isb
0x1883472b4     012298f2        movk x1, 0xc110
0x1883472b8     e1ffbff2        movk x1, 0xffff, lsl 16
0x1883472bc     e101c0f2        movk x1, 0xf, lsl 32
0x1883472c0     0100e0f2        movk x1, 0, lsl 48
0x1883472c4     280040f9        ldr x8, [x1]
0x1883472c8     e9f23cd5        mrs x9, s3_4_c15_c2_7
0x1883472cc     1f0109eb        cmp x8, x9
0x1883472d0     c1020054        b.ne 0x188347328
```

commpage



"New" JIT

- Replaces split JIT region with unified RWX region again
- Custom system register controls whether it's r-x or rw-

```
0x188347298     002298f2        movk x0, 0xc110
0x18834729c     e0ffbff2        movk x0, 0xffff, lsl 16
0x1883472a0     e001c0f2        movk x0, 0xf, lsl 32
0x1883472a4     0000e0f2        movk x0, 0, lsl 48
0x1883472a8     000040f9        ldr x0, [x0]
0x1883472ac     e0f21cd5        msr s3_4_c15_c2_7, x0
0x1883472b0     df3f03d5        isb
0x1883472b4     012298f2        movk x1, 0xc110
0x1883472b8     e1ffbff2        movk x1, 0xffff, lsl 16
0x1883472bc     e101c0f2        movk x1, 0xf, lsl 32
0x1883472c0     0100e0f2        movk x1, 0, lsl 48
0x1883472c4     280040f9        ldr x8, [x1]
0x1883472c8     e9f23cd5        mrs x9, s3_4_c15_c2_7
0x1883472cc     1f0109eb        cmp x8, x9
0x1883472d0     c1020054        b.ne 0x188347328
```

commpage

custom register

"New" JIT

- Replaces split JIT region with unified RWX region again
- Custom system register controls whether it's r-x or rw-

```
0x188347298     002298f2        movk x0, 0xc110
0x18834729c     e0ffbff2        movk x0, 0xffff, lsl 16
0x1883472a0     e001c0f2        movk x0, 0xf, lsl 32
0x1883472a4     0000e0f2        movk x0, 0, lsl 48
0x1883472a8     000040f9        ldr x0, [x0]
0x1883472ac     e0f21cd5        msr s3_4_c15_c2_7, x0
0x1883472b0     df3f03d5        isb
0x1883472b4     012298f2        movk x1, 0xc110
0x1883472b8     e1ffbff2        movk x1, 0xffff, lsl 16
0x1883472bc     e101c0f2        movk x1, 0xf, lsl 32
0x1883472c0     0100e0f2        movk x1, 0, lsl 48
0x1883472c4     280040f9        ldr x8, [x1]
0x1883472c8     e9f23cd5        mrs x9, s3_4_c15_c2_7
0x1883472cc     1f0109eb        cmp x8, x9
0x1883472d0     c1020054        b.ne 0x188347328
```

commpage

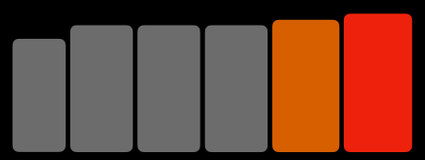
custom register

ROP prevention

"New" JIT

- Replaces split JIT region with unified RWX region again
- Custom system register controls whether it's r-x or rw-

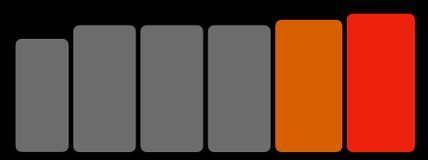
	A11	A12
rw-	0x3232767611107676	0x3232767711107776
r-x	0x3232767612107676	0x3232767712107776



"New" JIT

- Replaces split JIT region with unified RWX region again
- Custom system register controls whether it's r-x or rw-

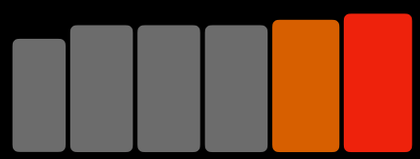
	A11	A12
rw-	0x3232767611107676	0x3232767711107776
r-x	0x3232767612107676	0x3232767712107776



APRR

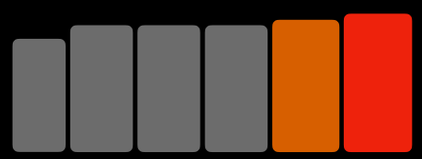
APRR

- Was rumoured to be "userland KTRR"



APRR

- Was rumoured to be "userland KTRR"
- Actually introduced in A11

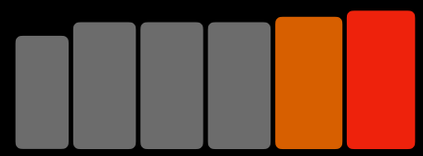


APRR

- Was rumoured to be "userland KTRR"
- Actually introduced in A11
- Not to be confused with PPL

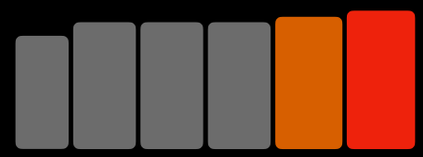
APRR

s3_4_c15_c2_0	
s3_4_c15_c2_1	
s3_4_c15_c2_2	
s3_4_c15_c2_3	
s3_4_c15_c2_4	
s3_4_c15_c2_5	
s3_4_c15_c2_6	
s3_4_c15_c2_7	



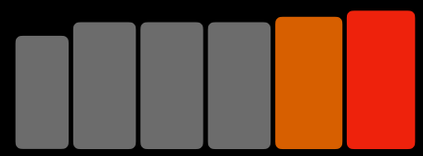
APRR

s3_4_c15_c2_0	
s3_4_c15_c2_1	
s3_4_c15_c2_2	KTRR_LOCK_EL1
s3_4_c15_c2_3	KTRR_LOWER_EL1
s3_4_c15_c2_4	KTRR_UPPER_EL1
s3_4_c15_c2_5	
s3_4_c15_c2_6	
s3_4_c15_c2_7	



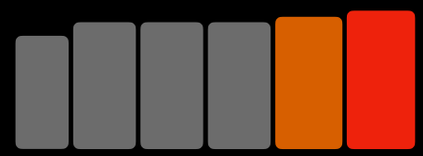
APRR

s3_4_c15_c2_0	
s3_4_c15_c2_1	Used in PPLTRAMP
s3_4_c15_c2_2	KTRR_LOCK_EL1
s3_4_c15_c2_3	KTRR_LOWER_EL1
s3_4_c15_c2_4	KTRR_UPPER_EL1
s3_4_c15_c2_5	
s3_4_c15_c2_6	
s3_4_c15_c2_7	



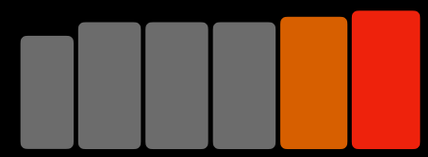
APRR

s3_4_c15_c2_0	
s3_4_c15_c2_1	Used in PPLTRAMP
s3_4_c15_c2_2	KTRR_LOCK_EL1
s3_4_c15_c2_3	KTRR_LOWER_EL1
s3_4_c15_c2_4	KTRR_UPPER_EL1
s3_4_c15_c2_5	
s3_4_c15_c2_6	
s3_4_c15_c2_7	Used in JIT, EL0 access



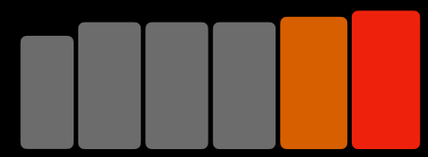
APRR

s3_4_c15_c2_0	
s3_4_c15_c2_1	Used in PPLTRAMP
s3_4_c15_c2_2	KTRR_LOCK_EL1
s3_4_c15_c2_3	KTRR_LOWER_EL1
s3_4_c15_c2_4	KTRR_UPPER_EL1
s3_4_c15_c2_5	
s3_4_c15_c2_6	Also related to JIT
s3_4_c15_c2_7	Used in JIT, EL0 access



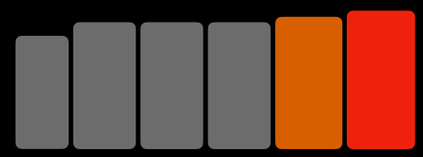
APRR

s3_4_c15_c2_0	Also part of APRR
s3_4_c15_c2_1	Used in PPLTRAMP
s3_4_c15_c2_2	KTRR_LOCK_EL1
s3_4_c15_c2_3	KTRR_LOWER_EL1
s3_4_c15_c2_4	KTRR_UPPER_EL1
s3_4_c15_c2_5	
s3_4_c15_c2_6	Also related to JIT
s3_4_c15_c2_7	Used in JIT, EL0 access



APRR

s3_4_c15_c2_0	Also part of APRR
s3_4_c15_c2_1	Used in PPLTRAMP
s3_4_c15_c2_2	KTRR_LOCK_EL1
s3_4_c15_c2_3	KTRR_LOWER_EL1
s3_4_c15_c2_4	KTRR_UPPER_EL1
s3_4_c15_c2_5	???
s3_4_c15_c2_6	Also related to JIT
s3_4_c15_c2_7	Used in JIT, EL0 access



APRR

Reg	A11	A12
0	0x4545010165670101 0x4545010167670101	0x4545010065670001 0x4545010067670001 0x4545010167670101
1	0x4455445564666677 0x4455445566666677	0x4455445464666477 0x4455445564666677 0x4455445566666677
6	0x00 0x40	0x00 0x40
7	0x3232767611107676 0x3232767612107676	0x3232767711107776 0x3232767712107776

APRR

Reg	A11	A12
0	0x4545010165670101 0x4545010167670101	0x4545010065670001 0x4545010067670001 0x4545010167670101
1	0x4455445564666677 0x4455445566666677	0x4455445464666477 0x4455445564666677 0x4455445566666677
6	0b0000000000000000 0b0000000001000000	0b0000000000000000 0b0000000001000000
7	0x3232767611107676 0x3232767612107676	0x3232767711107776 0x3232767712107776

```
strings kernel | fgrep APRR
```

strings kernel | fgrep APRR

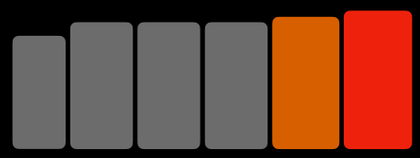
- "pmap_page_protect: modifying an APRR mapping pte_p=%p pmap=%p prot=%d options=%u, pv_h=%p, pveh_p=%p, pve_p=%p, pte=0x%llx, tmplate=0x%llx, va=0x%llx ppnum: 0x%x"

- "pmap_page_protect: creating an APRR mapping pte_p=%p pmap=%p prot=%d options=%u, pv_h=%p, pveh_p=%p, pve_p=%p, pte=0x%llx, tmplate=0x%llx, va=0x%llx ppnum: 0x%x"

pmap_page_protect(_internal)

pmap_page_protect(_internal)

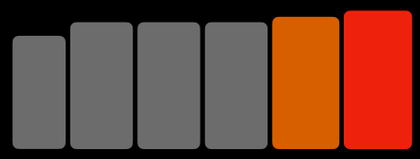
```
mrs x11, s3_4_c15_c2_0
mrs x10, s3_4_c15_c2_1
mrs x8, s3_4_c15_c2_6
lsr x9, x22, #4
and x9, x9, 0xc
bfxil x9, x22, #52, #2
bfxil x9, x22, #54, #1
lsl x12, x9, #2
mov x13, 0x101010101010101
movk x13, 0x6767, lsl #16
movk x13, 0x4545, lsl #48
eor x13, x11, x13
orr w11, wzr, 0x7
lsl x11, x11, x12
tst x11, x13
b.ne 0xffffff008efcb5c
mov x12, 0x6677
movk x12, 0x6666, lsl #16
movk x12, 0x4455, lsl #32
movk x12, 0x4455, lsl #48
eor x10, x10, x12
tst x11, x10
b.ne 0xffffff008efcb5c
orr w10, wzr, 0x1
lsl x9, x10, x9
tst x9, x8
b.eq 0xffffff008efcb94
```



pmap_page_protect(_internal)

```
mrs x11, s3_4_c15_c2_0
mrs x10, s3_4_c15_c2_1
mrs x8, s3_4_c15_c2_6
lsr x9, x22, #4
and x9, x9, 0xc
bfxil x9, x22, #52, #2
bfxil x9, x22, #54, #1
lsl x12, x9, #2
mov x13, 0x1010101010101
movk x13, 0x6767, lsl #16
movk x13, 0x4545, lsl #48
eor x13, x11, x13
orr w11, wzr, 0x7
lsl x11, x11, x12
tst x11, x13
b.ne 0xffffff008efcb5c
mov x12, 0x6677
movk x12, 0x6666, lsl #16
movk x12, 0x4455, lsl #32
movk x12, 0x4455, lsl #48
eor x10, x10, x12
tst x11, x10
b.ne 0xffffff008efcb5c
orr w10, wzr, 0x1
lsl x9, x10, x9
tst x9, x8
b.eq 0xffffff008efcb94
```

Register read



pmap_page_protect(_internal)

```
mrs x11, s3_4_c15_c2_0
mrs x10, s3_4_c15_c2_1
mrs x8, s3_4_c15_c2_6
lsr x9, x22, #4
and x9, x9, 0xc
bfxil x9, x22, #52, #2
bfxil x9, x22, #54, #1
lsl x12, x9, #2
mov x13, 0x101010101010101
movk x13, 0x6767, lsl #16
movk x13, 0x4545, lsl #48
eor x13, x11, x13
orr w11, wzr, 0x7
lsl x11, x11, x12
tst x11, x13
b.ne 0xffffff008efcb5c
mov x12, 0x6677
movk x12, 0x6666, lsl #16
movk x12, 0x4455, lsl #32
movk x12, 0x4455, lsl #48
eor x10, x10, x12
tst x11, x10
b.ne 0xffffff008efcb5c
orr w10, wzr, 0x1
lsl x9, x10, x9
tst x9, x8
b.eq 0xffffff008efcb94
```

Register read

TTE bit mashing

pmap_page_protect(_internal)

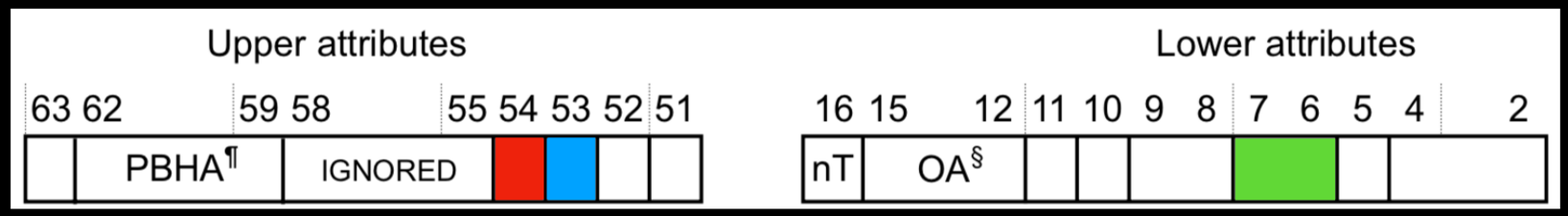
```
mrs x11, s3_4_c15_c2_0
mrs x10, s3_4_c15_c2_1
mrs x8, s3_4_c15_c2_6
lsr x9, x22, #4
and x9, x9, 0xc
bfxil x9, x22, #52, #2
bfxil x9, x22, #54, #1
lsl x12, x9, #2
mov x13, 0x101010101010101
movk x13, 0x6767, lsl #16
movk x13, 0x4545, lsl #48
eor x13, x11, x13
orr w11, wzr, 0x7
lsl x11, x11, x12
tst x11, x13
b.ne 0xffffff008efcb5c
mov x12, 0x6677
movk x12, 0x6666, lsl #16
movk x12, 0x4455, lsl #32
movk x12, 0x4455, lsl #48
eor x10, x10, x12
tst x11, x10
b.ne 0xffffff008efcb5c
orr w10, wzr, 0x1
lsl x9, x10, x9
tst x9, x8
b.eq 0xffffff008efcb94
```

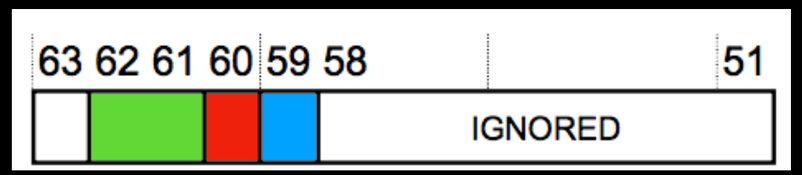
Register read

TTE bit mashing

XOR'ing

TTE bits

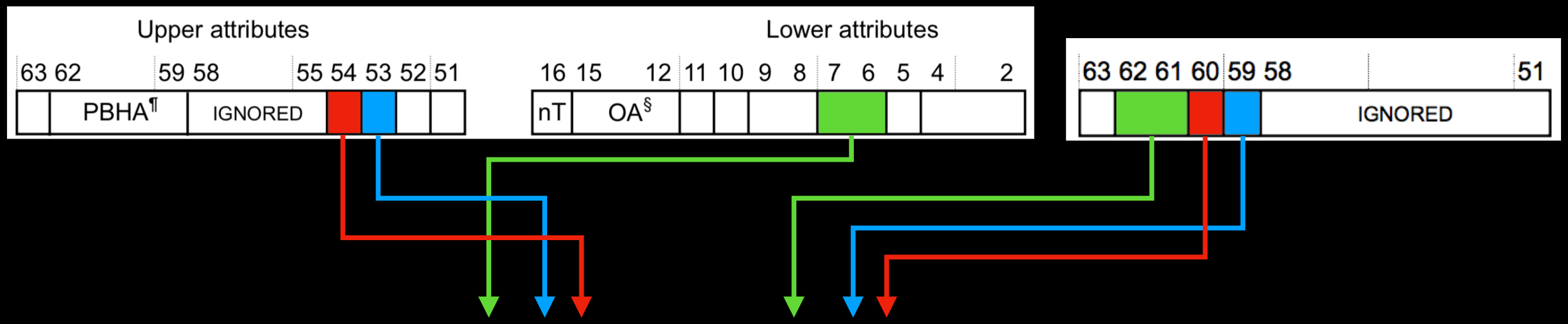

Upper attributes
Lower attributes
63 62
59 58
55 54 53 52 51
16 15
12 11 10 9 8 7 6 5 4
2
PBHA¶
IGNORED
nT
OA§


63 62 61 60 59 58
51
IGNORED

TTE bits

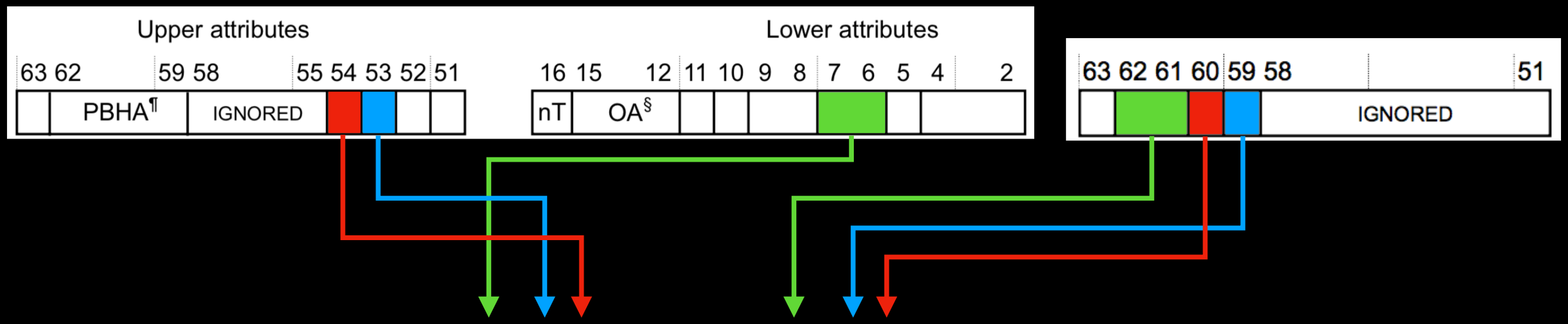

Upper attributes
Lower attributes
63 62
59 58
55 54 53 52 51
16 15
12
11 10 9 8
7 6 5 4
2
PBHA¶
IGNORED
nT
OA§
63 62 61 60 59 58
51
IGNORED
0bNNNN

TTE bits



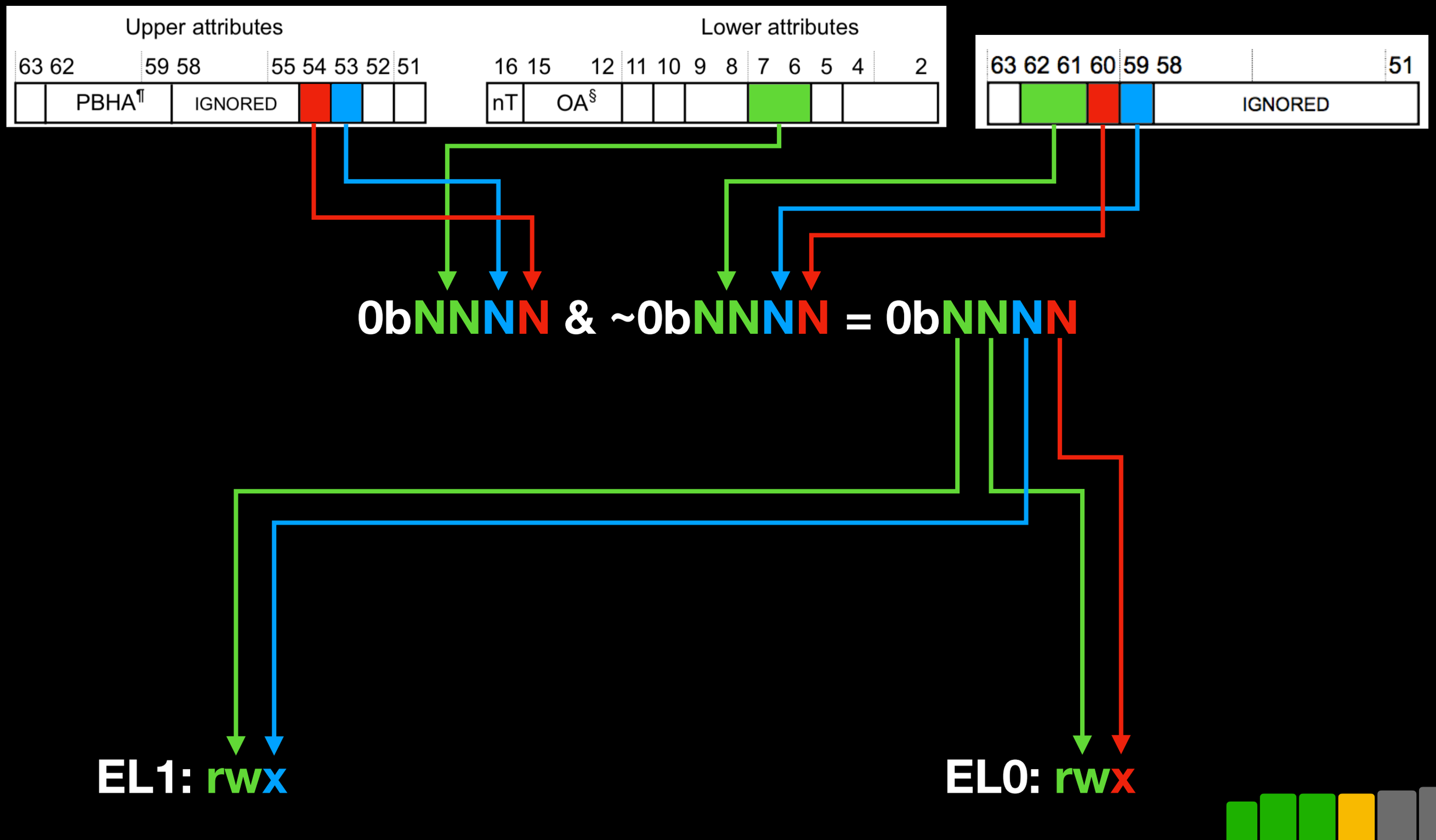
0bNNNN & ~0bNNNN

TTE bits



0bNNNN & ~0bNNNN = 0bNNNN

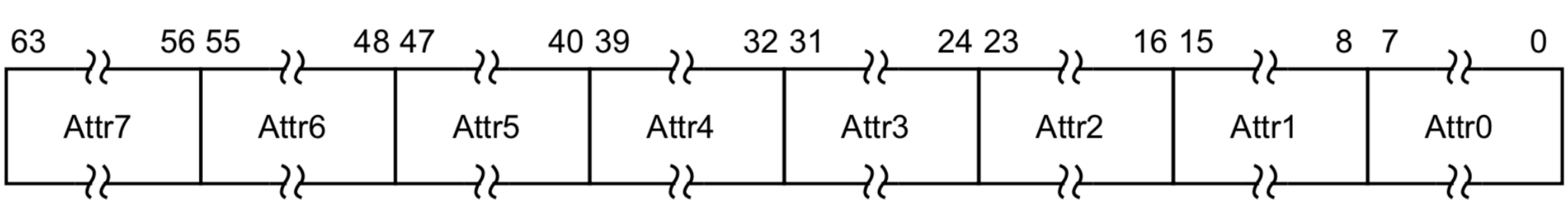
TTE bits


Upper attributes
Lower attributes
63 62 59 58 55 54 53 52 51
PBHA¶
IGNORED
16 15 12 11 10 9 8 7 6 5 4 2
nT
OA§
63 62 61 60 59 58 51
IGNORED
0bNNNN & ~0bNNNN = 0bNNNN
EL1: rwx
EL0: rwx

Register indexing

Field descriptions

The MAIR_EL1 bit assignments are:



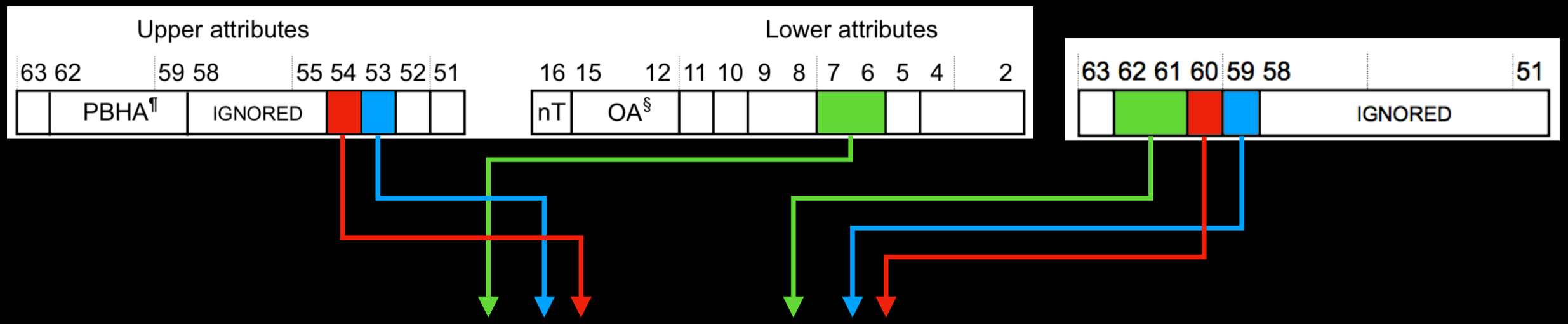
MAIR_EL1 is permitted to be cached in a TLB.

Attr<n>, bits [8n+7:8n], for n = 0 to 7

The memory attribute encoding for an AttrIndx[2:0] entry in a Long descriptor format translation table entry, where AttrIndx[2:0] gives the value of <n> in Attr<n>.

Bits [7:4] are encoded as follows:

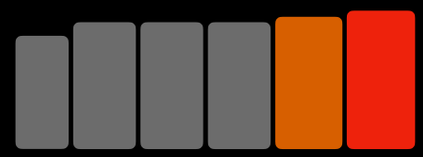
TTE bits



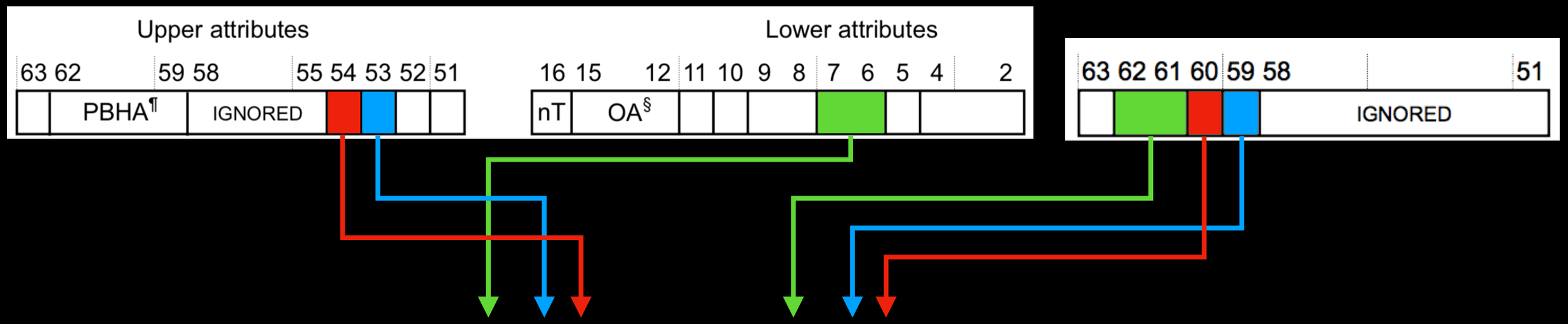
0bNNNN & ~0bNNNN = 0bNNNN

EL1: rwx

EL0: rwx



TTE bits



0bNNNN & ~0bNNNN = 0bNNNN

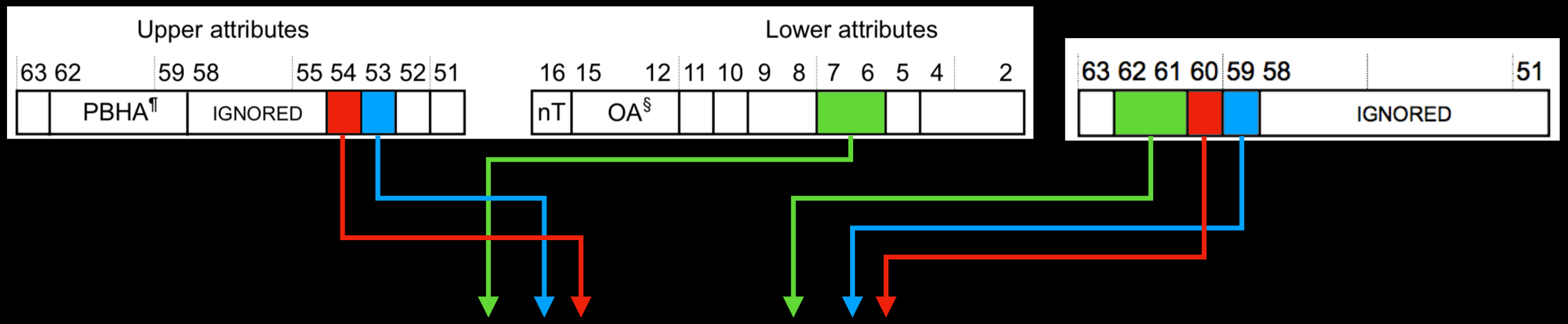
1

0x4455445464666477

EL1: rwx

EL0: rwx

TTE bits



0bNNNN & ~0bNNNN = 0bNNNN

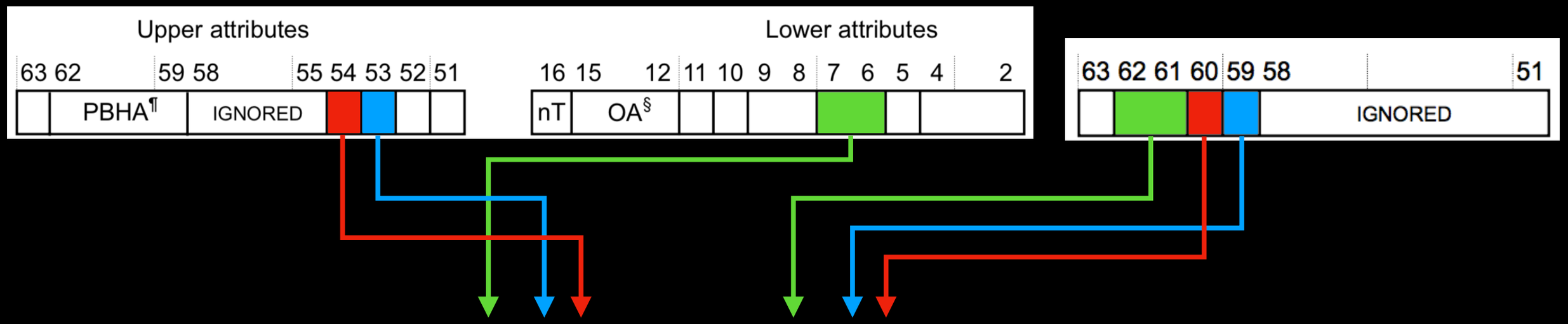
1

0x4455445464666477

EL1: rwx

EL0: rwx

TTE bits



0bNNNN & ~0bNNNNN = 0bNNNN

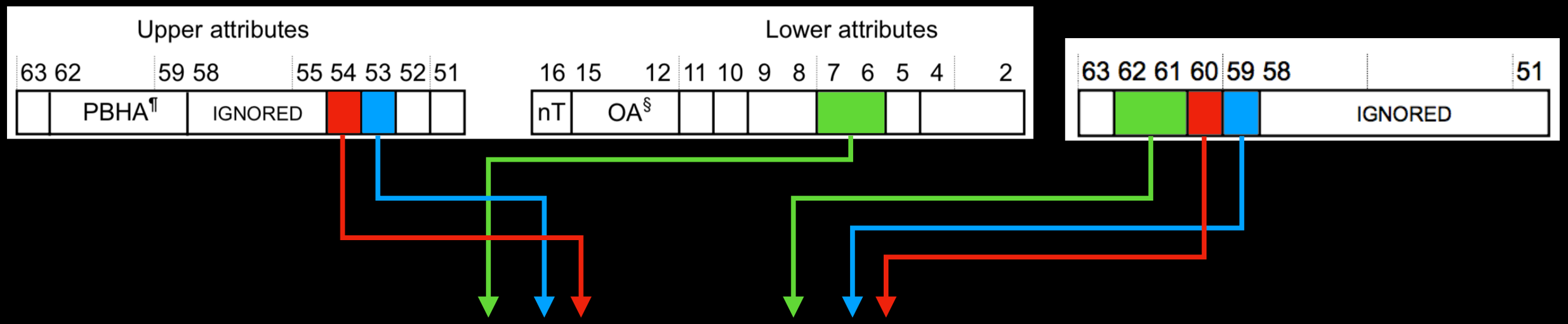
0 0x4545010067670001

1 0x4455445464666477

EL1: rwx

EL0: rwx

TTE bits



0bNNNN & ~0bNNNNN = 0bNNNN

0 0x4545010067670001

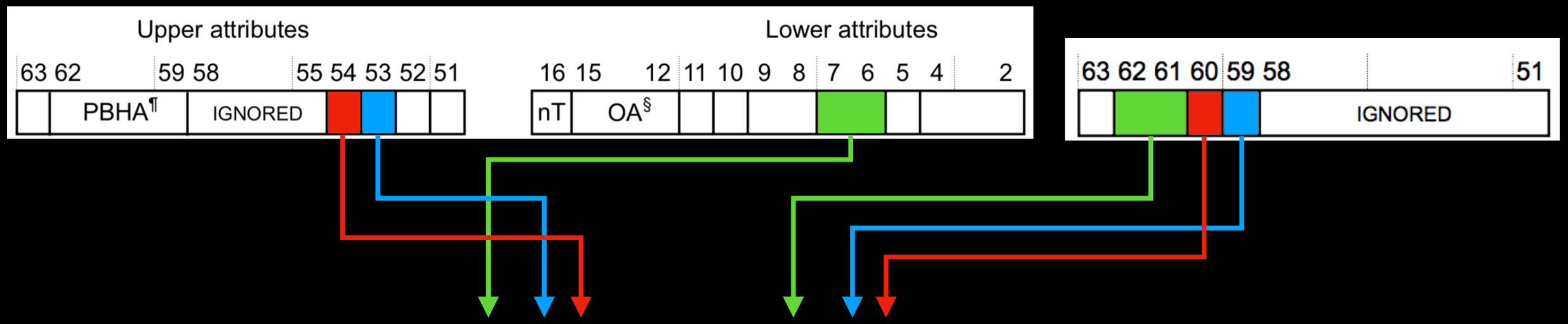
1 0x4455445464666477

EL1: rwx

0xN

EL0: rwx

TTE bits



0bNNNN & ~0bNNNNN = 0bNNNN

0 0x4545010067670001

6 0b0000000001000000

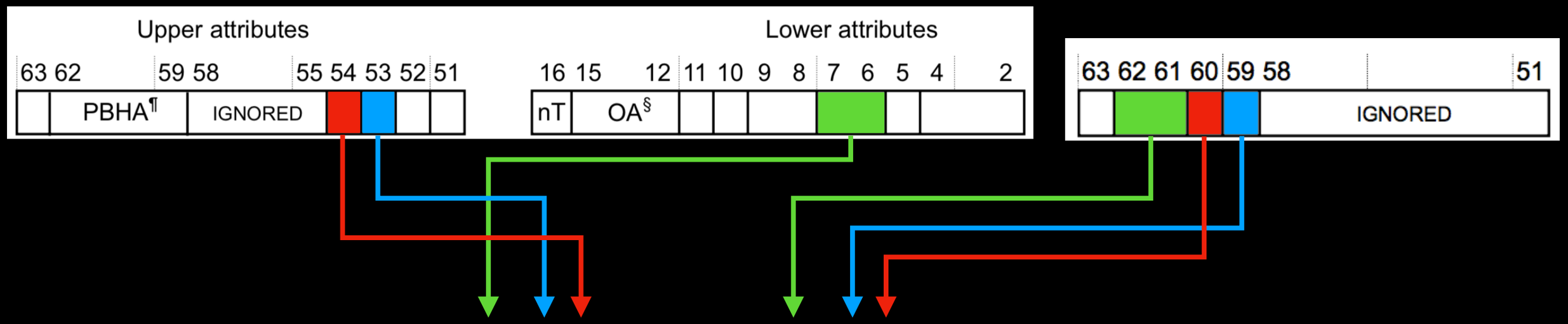
1 0x4455445464666477

EL1: rwx

0xN

EL0: rwx

TTE bits



0bNNNN & ~0bNNNN = 0bNNNN

6

0 0x4545010067670001 0b0000000001000000

1

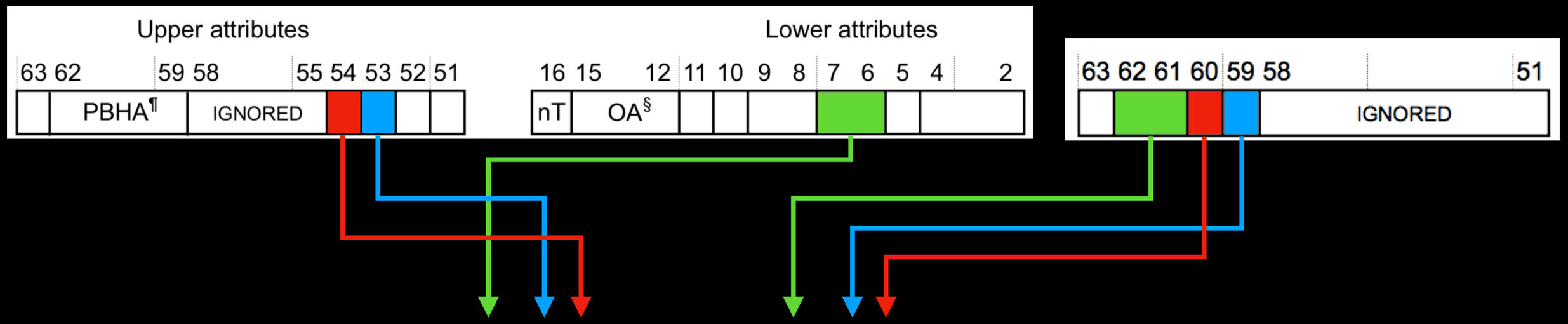
0x4455445464666477

0xN & ~0x0 = 0xN

EL1: rwx

EL0: rwx

TTE bits



0bNNNN & ~0bNNNN = 0bNNNN

0 0x4545010067670001

0b0000000001000000 6

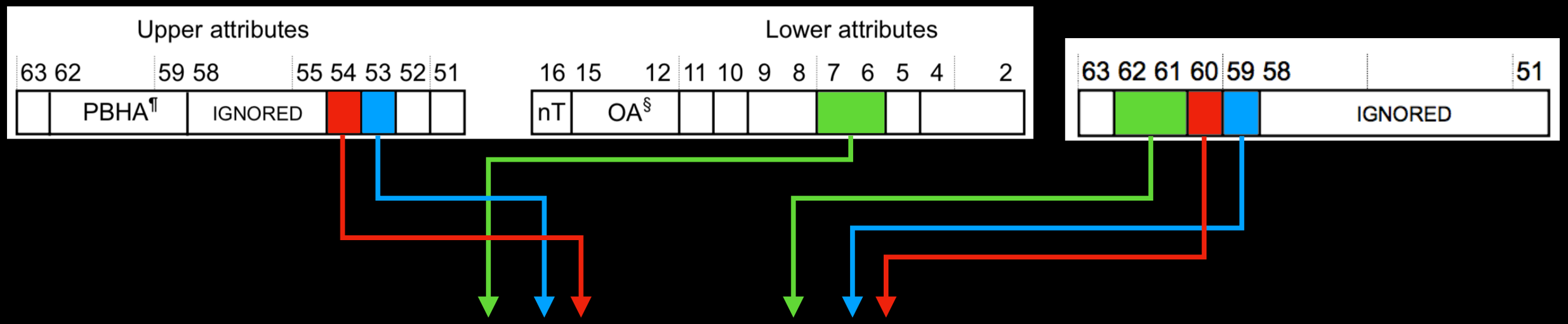
1 0x4455445464666477

EL1: rwx

0xN & ~0x0 = 0xN

EL0: rwx

TTE bits



0bNNNN & ~0bNNNN = 0bNNNN

0 0x4545010067670001

6 0b0000000001000000

0x3232767712107776

1 0x4455445464666477

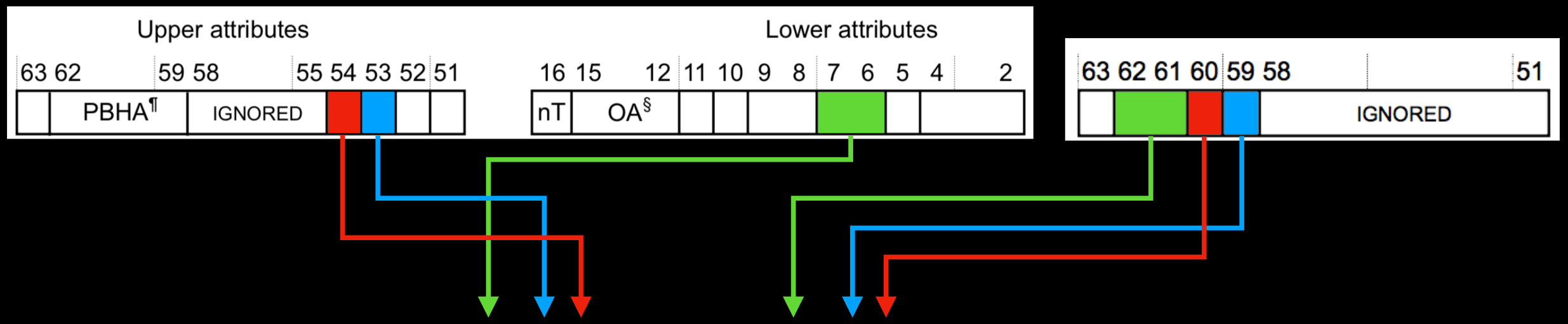
EL1: rwx

0xN & ~0x0 = 0xN

7

EL0: rwx

TTE bits



0bNNNN & ~0bNNNN = 0bNNNN

0 0x4545010067670001

6 0b0000000001000000

0x3232767712107776

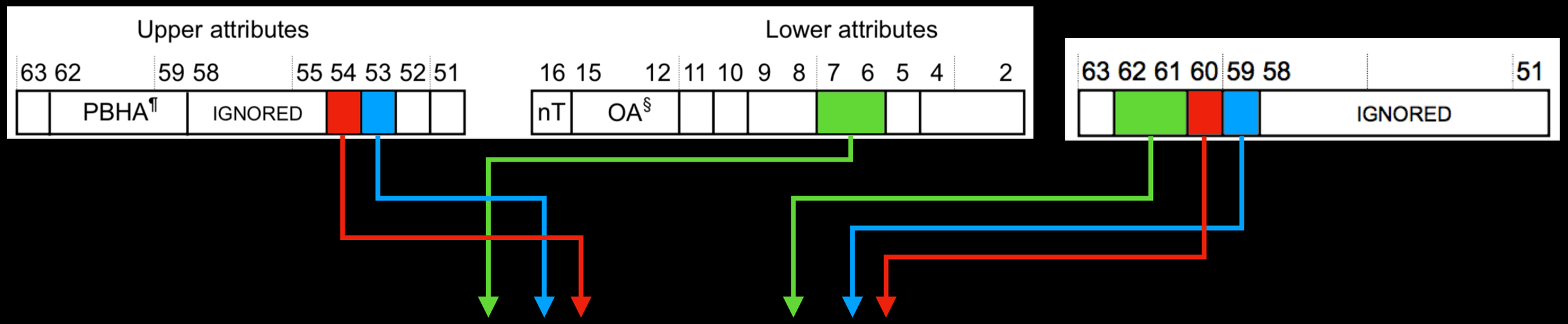
1 0x4455445464666477

7 0xN & ~0xN = 0xN

EL1: rwx

EL0: rwx

TTE bits



0bNNNN & ~0bNNNN = 0bNNNN

0 0x4545010067670001

6 0b0000000001000000

0x3232767712107776

1 0x4455445464666477

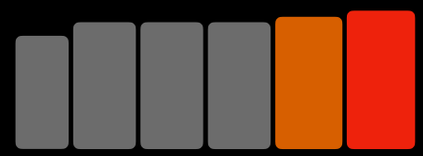
7 0xN & ~0xN = 0xN

EL1: rwx

EL0: rwx

APRR

s3_4_c15_c2_0	APRR_EL0
s3_4_c15_c2_1	APRR_EL1
s3_4_c15_c2_2	KTRR_LOCK_EL1
s3_4_c15_c2_3	KTRR_LOWER_EL1
s3_4_c15_c2_4	KTRR_UPPER_EL1
s3_4_c15_c2_5	???
s3_4_c15_c2_6	APRR_JIT_ENABLE
s3_4_c15_c2_7	APRR_JIT_MASK



The future

The future

- iOS 13: PPL / trust cache

The future

- iOS 13: PPL / trust cache
- A13: ARMv8.4? ExtPAC? BTI?

The future

- iOS 13: PPL / trust cache
- A13: ARMv8.4? ExtPAC? BTI?
- A14: ARMv8.5?

The future

- iOS 13: PPL / trust cache
- A13: ARMv8.4? ExtPAC? BTI?
- A14: ARMv8.5?
- Lots

The future

- iOS 13: PPL / trust cache
- A13: ARMv8.4? ExtPAC? BTI?
- A14: ARMv8.5?
- Lots
- and lots

The future

- iOS 13: PPL / trust cache
- A13: ARMv8.4? ExtPAC? BTI?
- A14: ARMv8.5?
- Lots
- and lots
- and lots of post-exploit mitigations

Questions?

Thanks

- The entire Jake Blair team
- Luca Todesco / @qwertyoruiopz
- Hao Xu / @windknown
- Brandon Azad
- Jonathan Levin
- @xerub