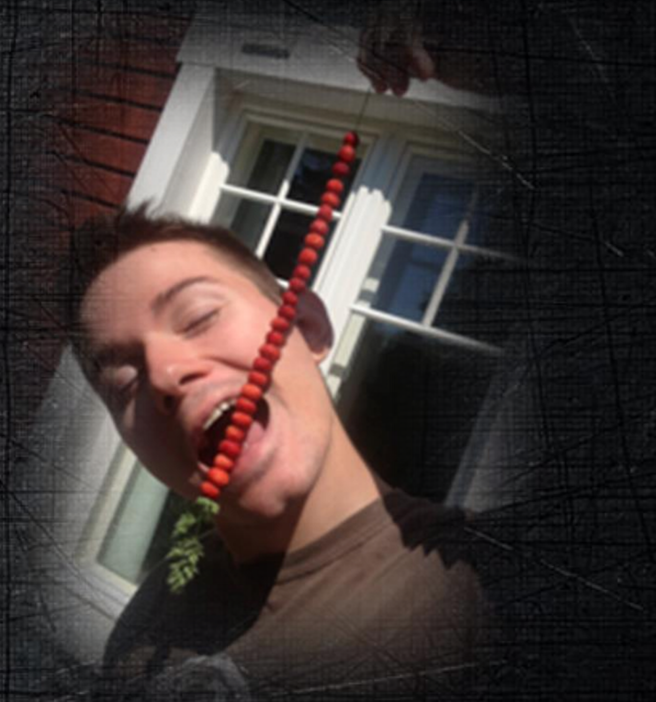
Visualizing Page Tables

… for Local Exploitation: Hacking Like in the Movies

Alexandru Radocea

- Developer at CrowdStrike, Inc.
  - iOS internals fan
  - Recovering software security assessor
  - Likes bringing pain to the adversary
- @defendtheworld on Twitter



CROWDSTRIKE

Georg Wicherski

- Researcher at CrowdStrike, Inc.
  - x86 & ARM low-level stuff
  - Reverse Engineering, Malware analysis
  - Exploitation and Mitigation research
- @ochsff on Twitter
- http://blog.oxff.net/

CROWDSTRIKE

Introduction

Paging 101

- Translation from virtual addresses to physical
  - Virtual address: the pointers your program works with
  - Physical address: the actual address of a memory cell in the physical RAM chip
- Virtual address unique per virtual memory space
  - Usually means per process for userland, one shared kernel space for all processes

CROWDSTRIKE

Efficient Hardware Implementation

- Group addresses into pages: block of addresses that are translated in the same way
- Cache translation results: TLB
- Hierarchical translation tables (trees) to conserve memory
  - Three levels on x86 and amd64
  - Two levels on ARMv7-A, three levels with LPAE

CROWDSTRIKE

Memory Protections

- Memory protections implemented on top of paging
  - Read-only vs. read-write memory areas
  - Executable vs. data-only memory areas
    - x86: NX (No-eXecute) bit per page
    - ARM: XN (eXecute-Never) bit per page ☺
  - Privilege level to access page
    - ARM: Supervisor bit, Domains, different table sets
    - x86: Supervisor bit (CPL, SMEP, SMAP)

CROWDSTRIKE

What a Movie Hacker Looks for

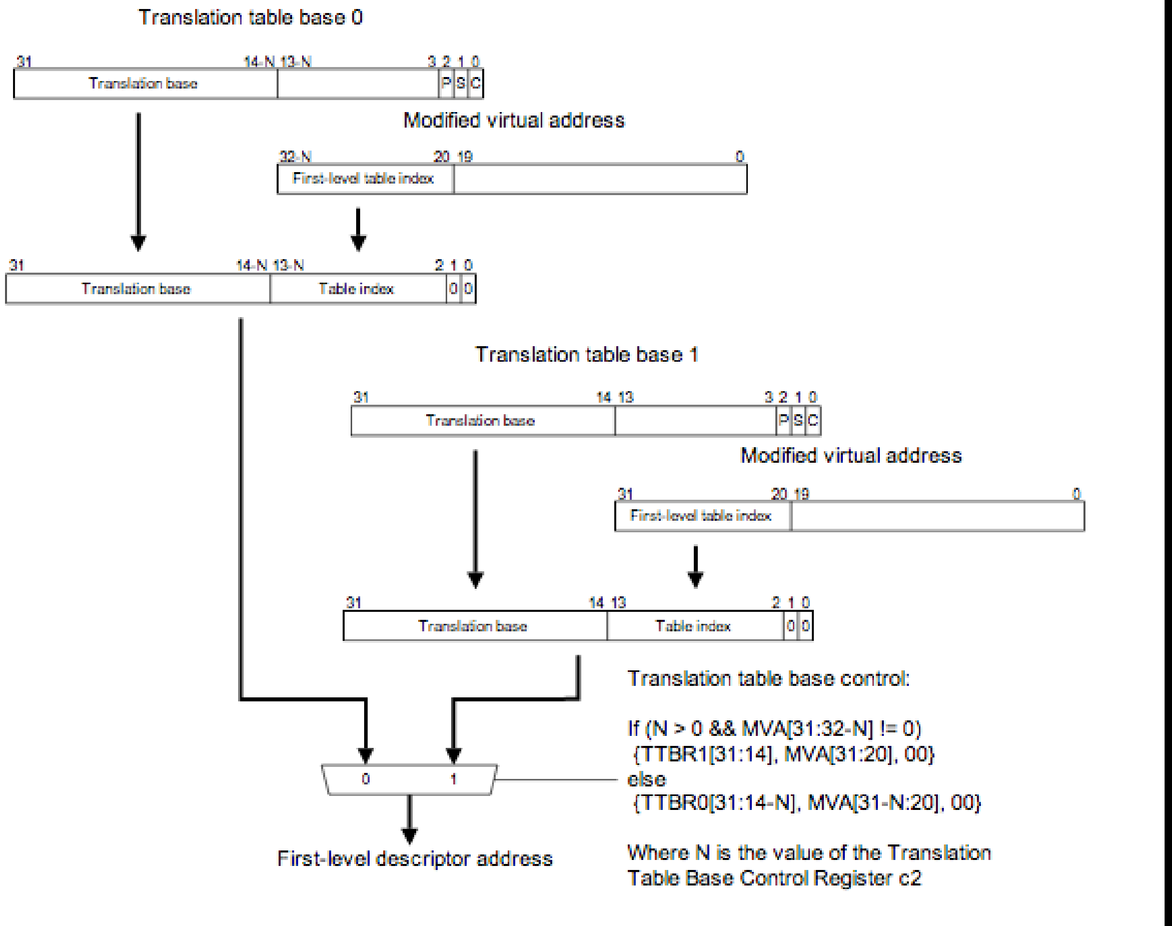
- Mappings at repeatedly constant addresses
  - Constant physical address: Subject to reliable FireWire attacks
  - Constant virtual address: ASLR bypass
- Mappings with unexpected protections
  - Read-write but not NX/XN: Classical copy shellcode and execute scenario
  - Driver specific weirdness (DMA memory, …)

CROWDSTRIKE

Background and Methodology

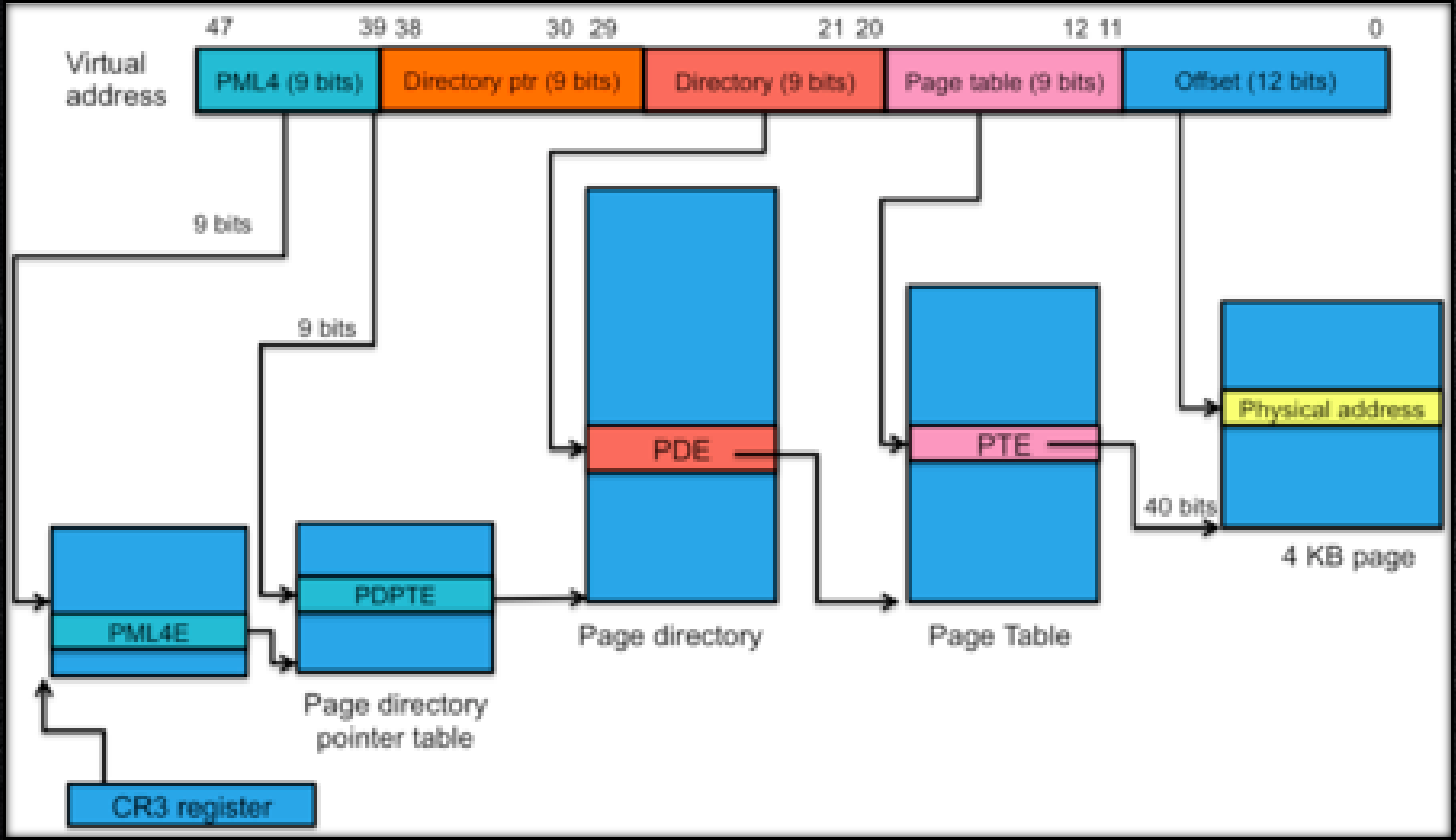
ARMv7-A VMSA

Figure 6.10. Creating a first-level descriptor address



http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0333h/Cihbfagh.html

IA-32e, four layers of fun



http://www.cs.rutgers.edu/~pxk/416/notes/09a-paging.html

Data Collection

- Android: Both custom kernel and local exploit
- iOS: Custom driver for jailbroken device
- x86_64 Linux: Custom kernel module
- x86_64 OS X: Custom kernel extension
- Windows Surface RT: Crash dumps & WinDBG
- Windows 8 x86_64: Custom kernel driver

Hilbert Curve Legend

User read only

Super read only

User write

Super write

User exec

Super exec

User WX

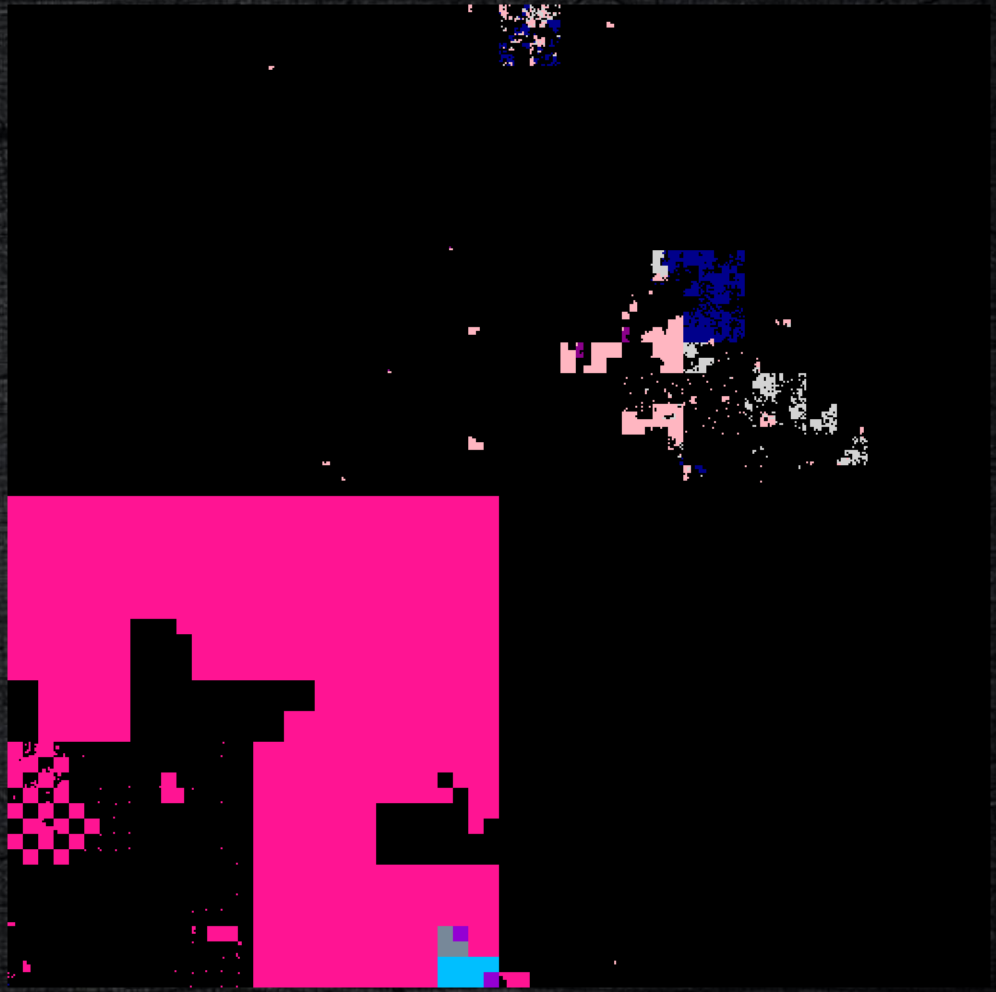
Super WX

CROWDSTRIKE

Case Studies

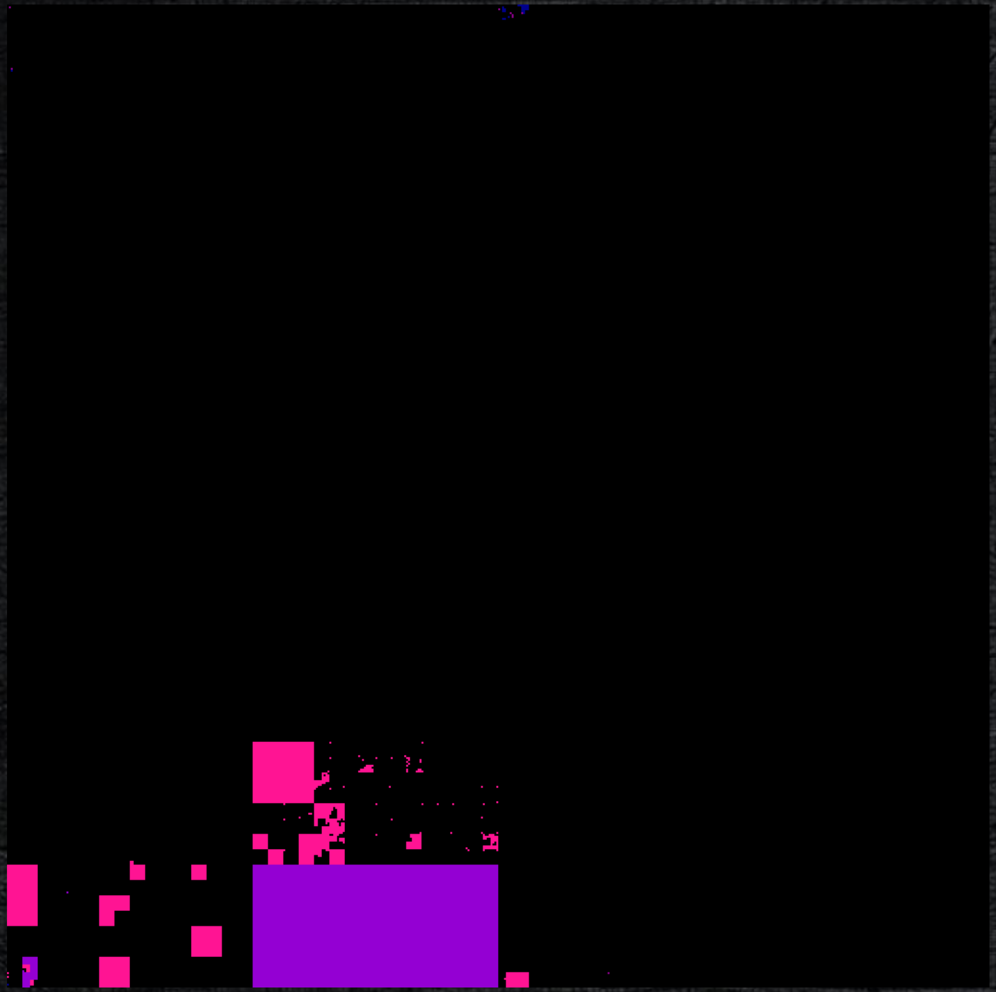
Android Process Comparison

1. init
2. dhcpd
3. zygote
4. com.android.email
5. sandboxed_process0 (Chrome)

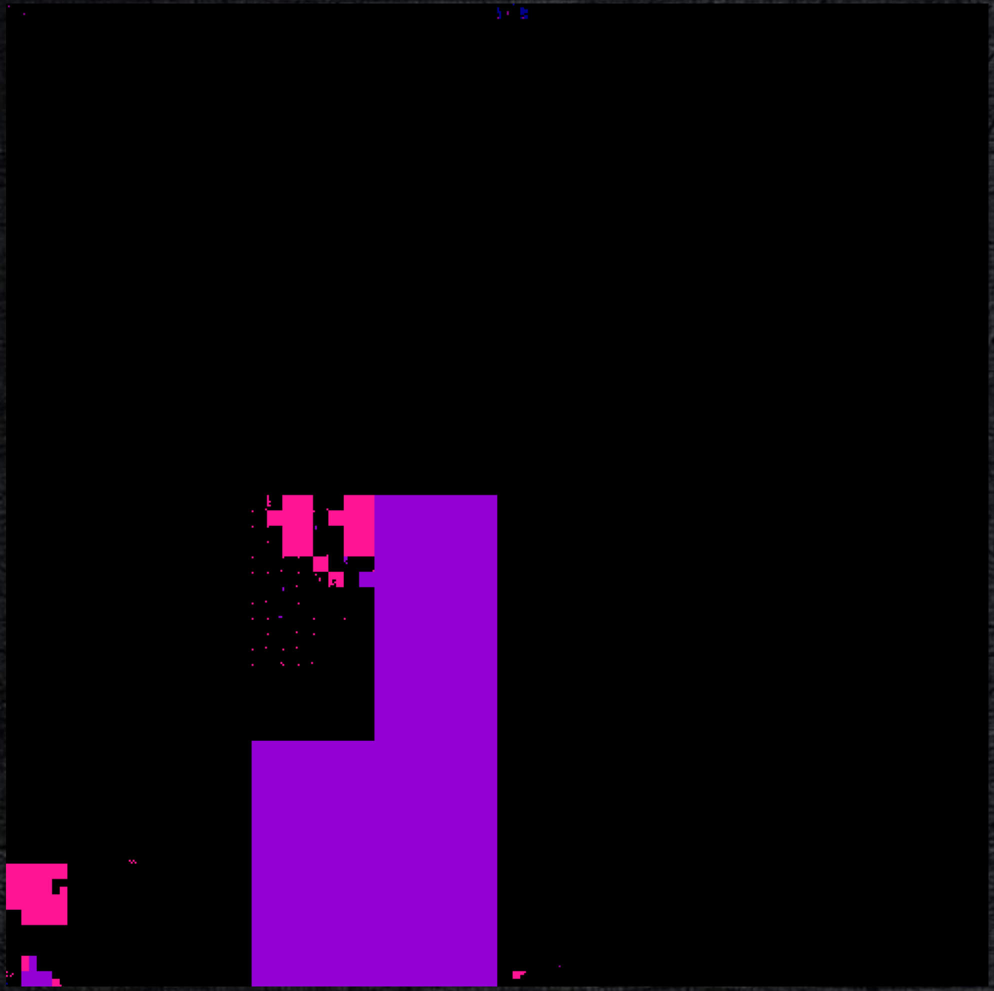


CROWDSTRIKE

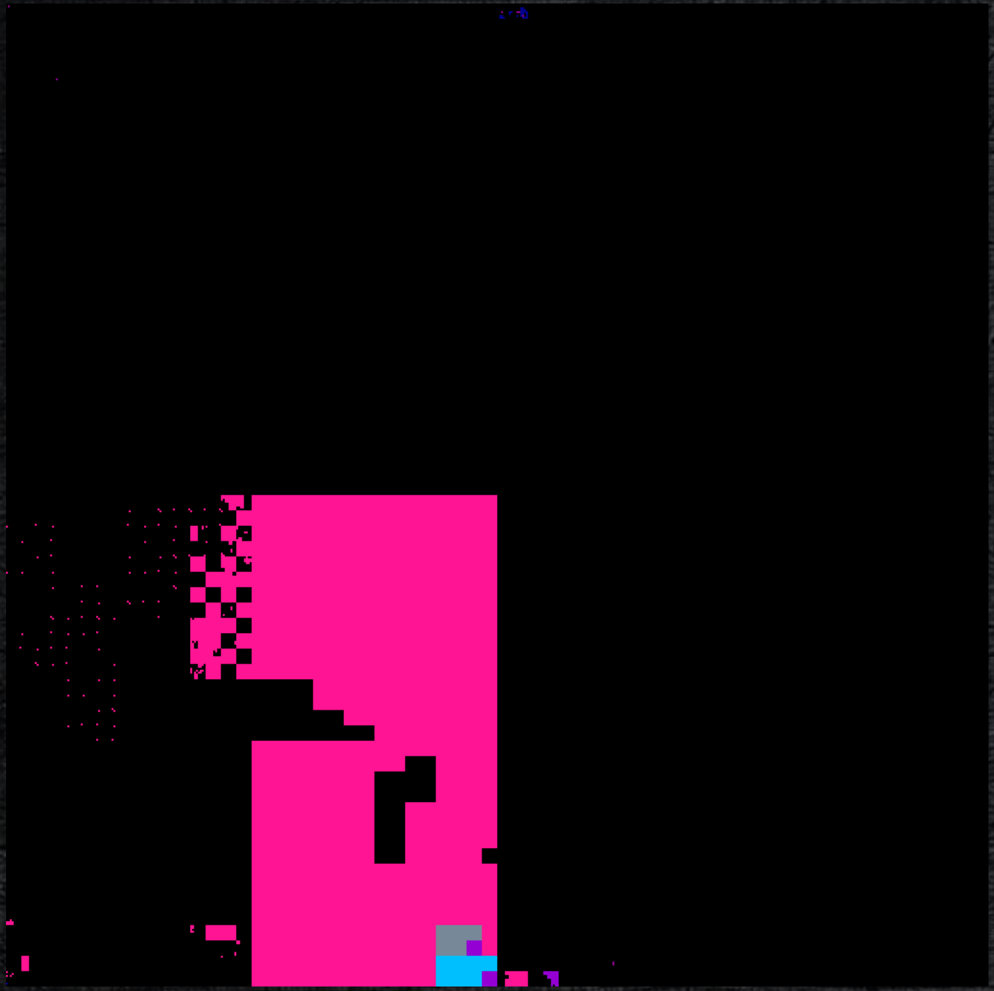
Galaxy Nexus, Android 4.2.2



Nexus 7, Android 4.2.2



Galaxy S4, Android 4.2.2 (MSM)



Android Observations

- Fixed r-x mapping at 0xffff0000 in all processes
  - 0xffff0000 is the ARM exception vectors base address
  - Abused in a *vsyscall* like manner by Linux on ARM
- Kernel `.text` is rwx on almost all kernels
  - `CONFIG_DEBUG_RODATA` not set in kernel configs
  - 3.4.x MSM kernel has RO .text
    - `CONFIG_STRICT_MEMORY_RWX` (Qualcomm)
    - Still has two rwx supervisor sections (1Mb pages)

CROWDSTRIKE

Android 4.2.2 ASLR Bypass

- `__kuser_cmpxchg: @ 0xffff0fc0`
  - `arch/arm/kernel/entry-armv.S`
  - `iff *r2 == r0: *r2 := r1`
  - Bruteforce addresses by invoking a loop, `r0`-`r2` are legitimate register parameters
  - Jump past equality check for arbitrary write gadget
- `__kuser_cmpxchg64: @ 0xffff0f60`
- `ffff0008:   ldr pc, [pc, #1072]   ; 0xffff0440`
  - This leaks the kernel's system call handler address to user-space

CROWDSTRIKE

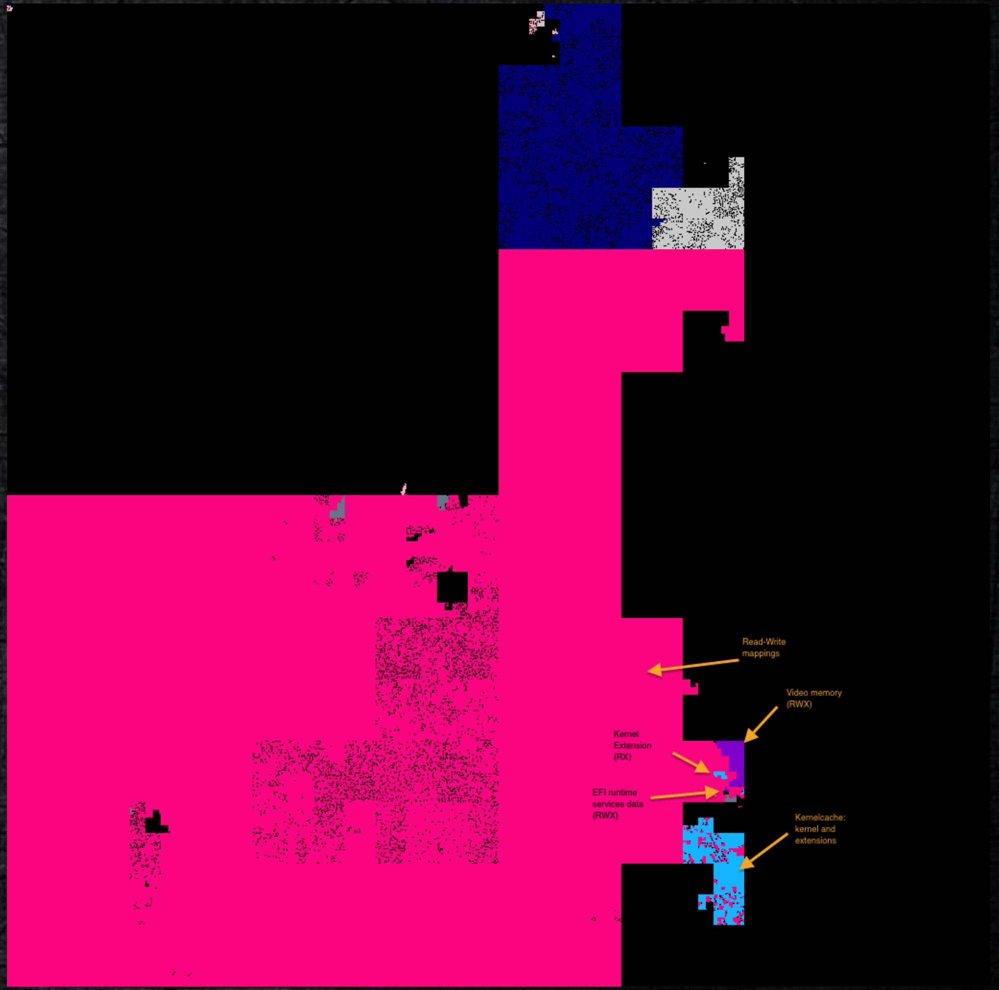
OS X Observations

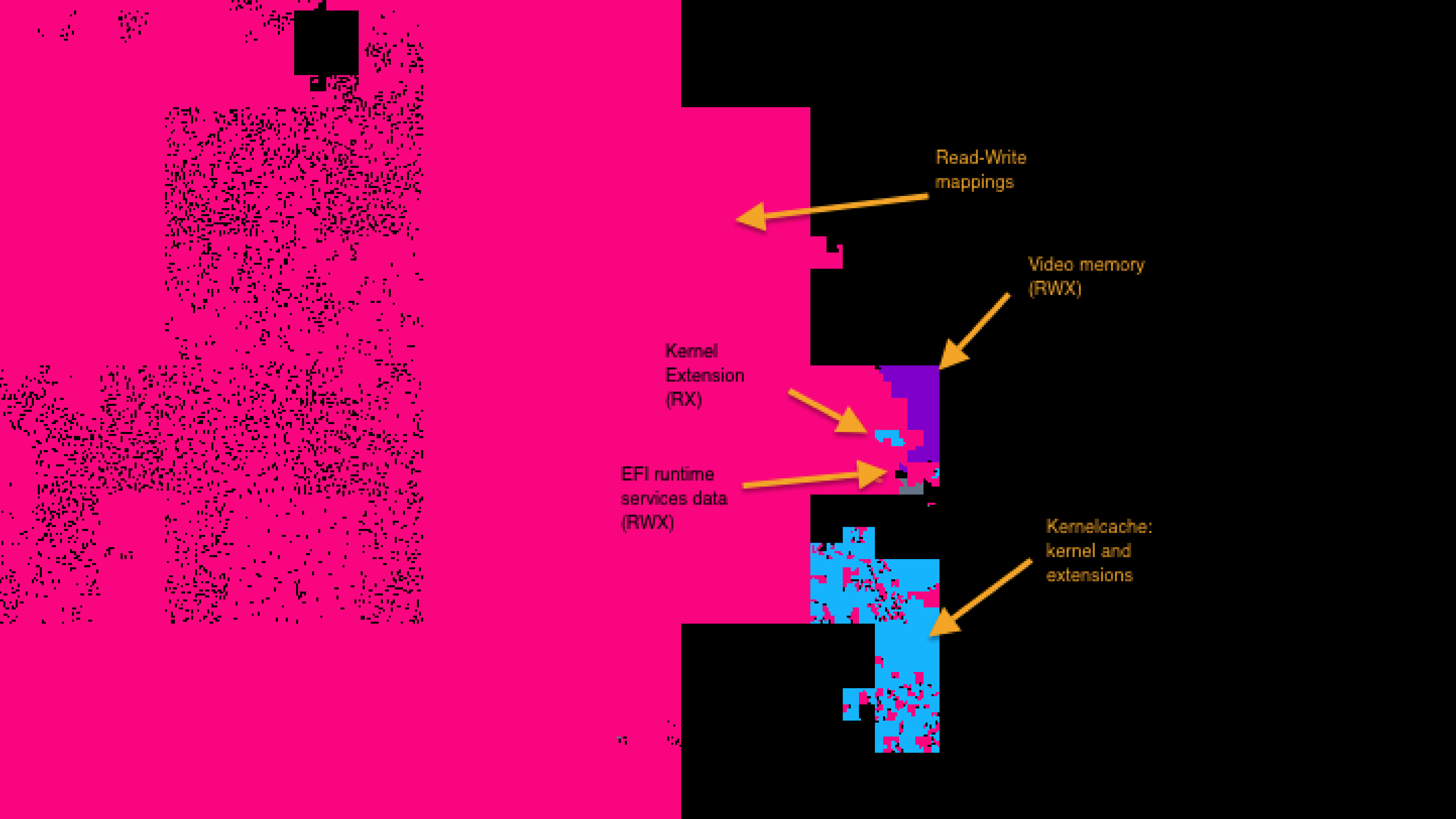
- Userland
  - Per-boot randomization (shared cache)
  - Per-execution randomization (dyld, pfz, commpage, stack, heap)

CROWDSTRIKE

OS X Observations

- Kernel
  - KASLR
  - Incomplete W^X
    - Randomized RWX
  - Shared address space
    - SMEP available




Read-Write
mappings
Video memory
(RWX)
Kernel
Extension
(RX)
EFI runtime
services data
(RWX)
Kernelcache:
kernel and
extensions

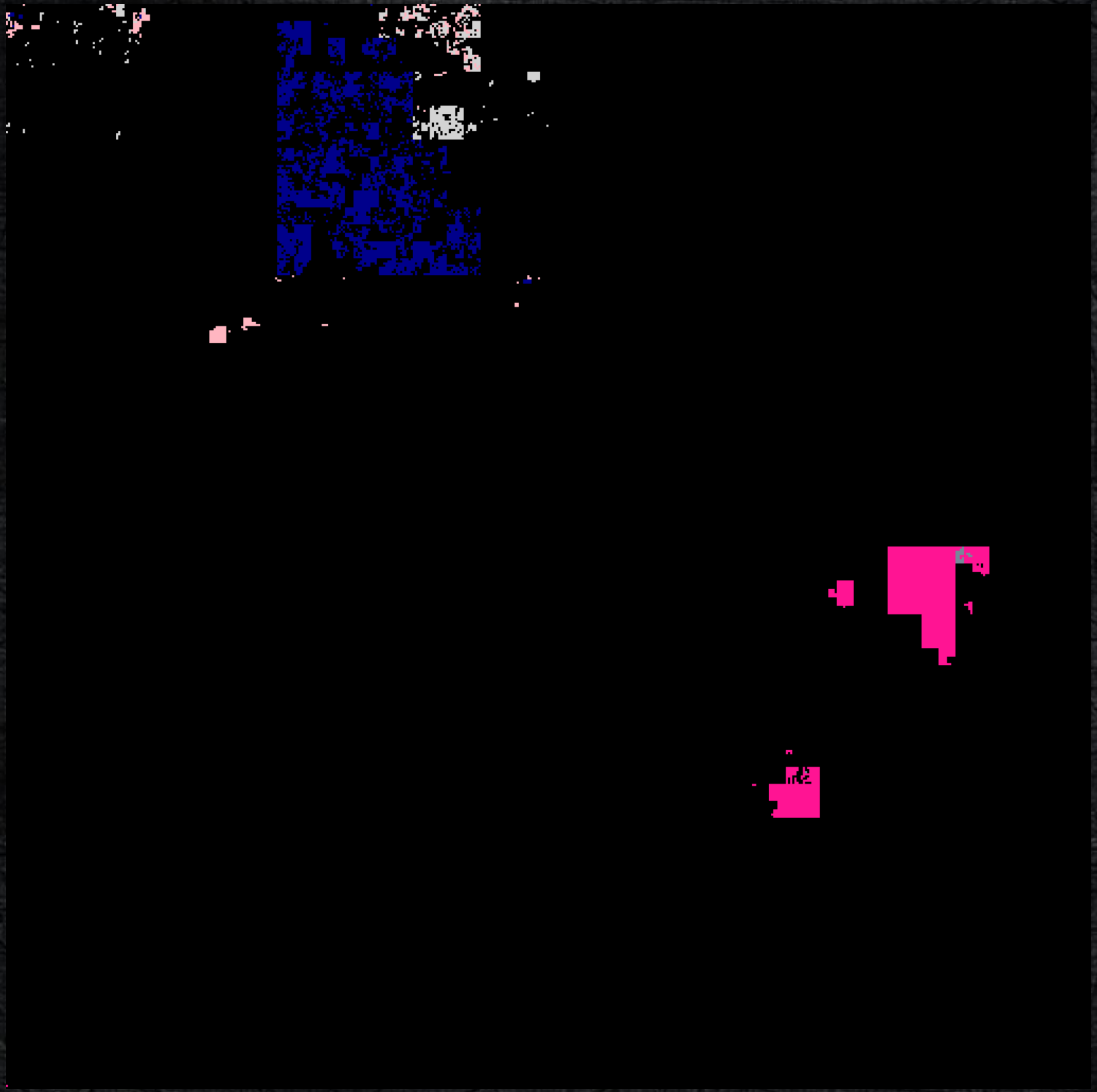
iOS 6 Security Properties

- Userland
  - Per-boot randomization (shared cache)
  - Per-execution randomization (dyld, .text, stack, heap)
  - Heap and stack separately randomized
  - W^X + Signed pages

iOS 6 Security Properties

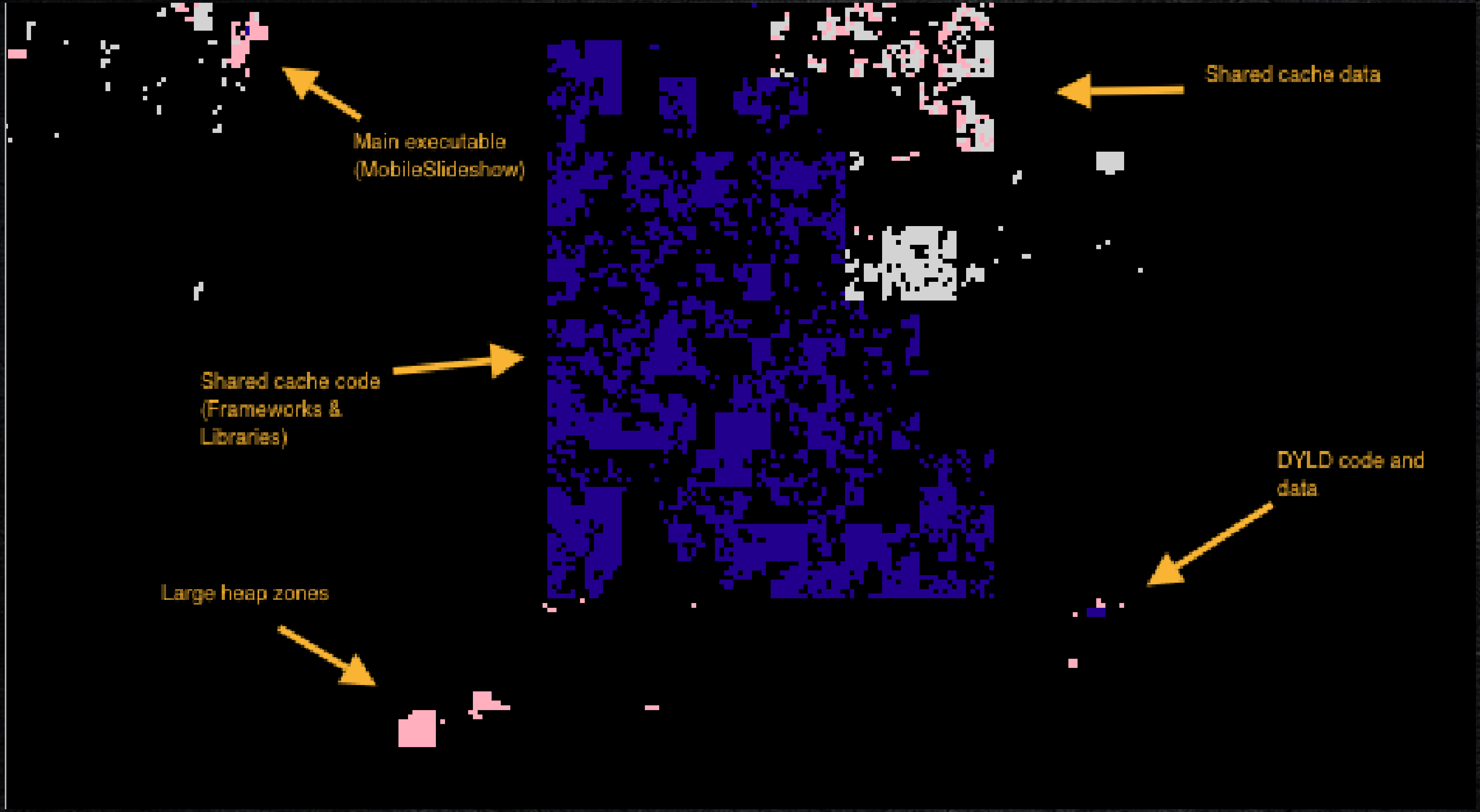
- Kernel
  - KASLR
  - W^X
  - TTBR0/1 swapping

iOS: Example process (MobileSlideshow)



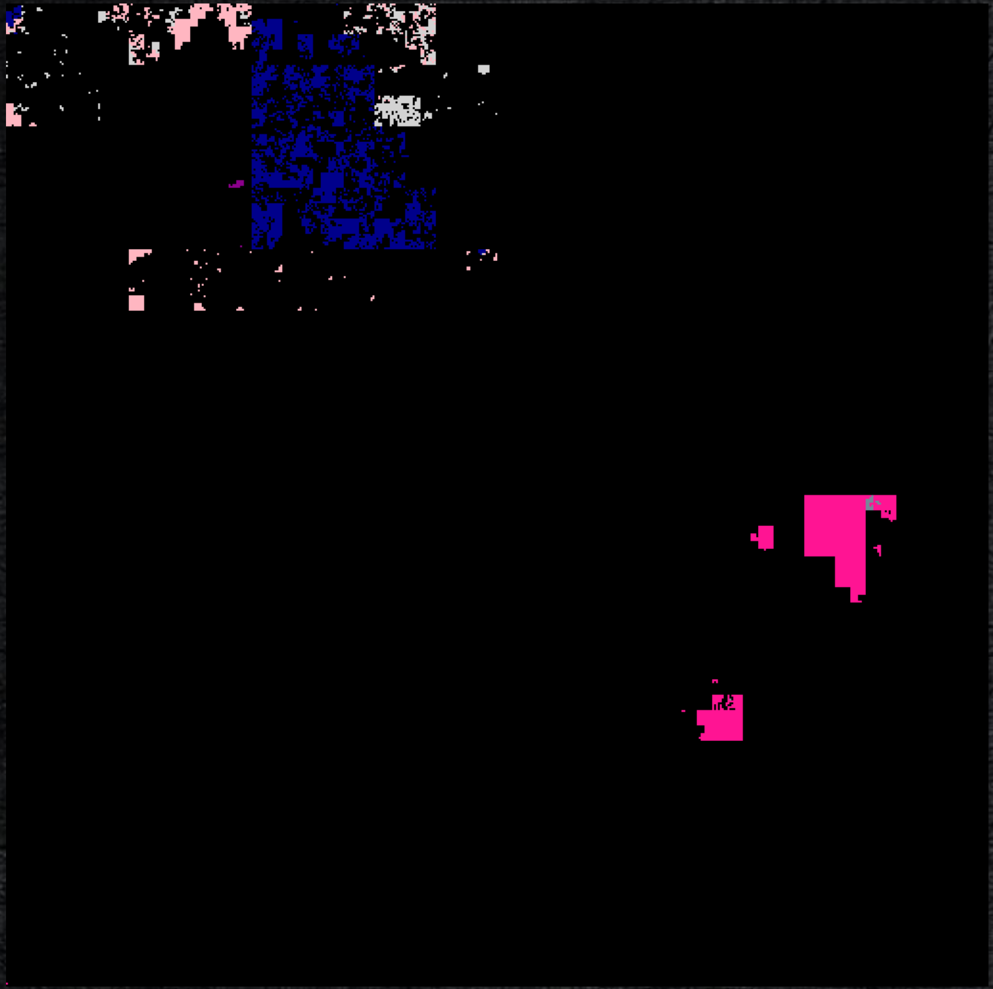
CROWDSTRIKE

iOS: Example process (MobileSlideshow)

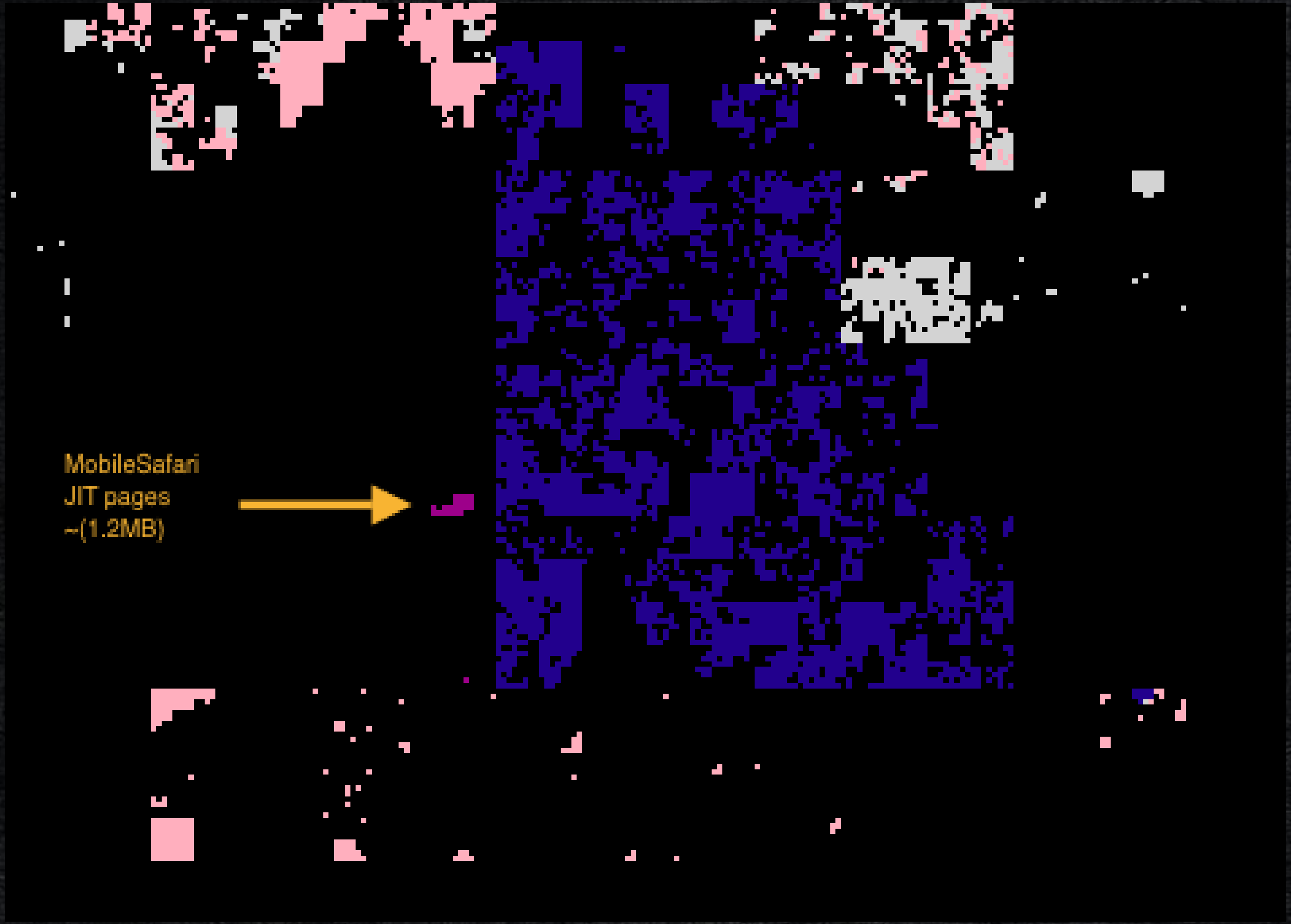


CROWDSTRIKE

iOS: Example process (MobileSafari)



iOS: Example process (MobileSafari)



CROWDSTRIKE

iOS Observations

- Evasi0n jailbreak leaves kernel mappings as RWX
- Fixed physical memory mappings across boots
  - Weakness with virtual mapping leak or physical memory write

CROWDSTRIKE