"Dead pixel detected" - A Security Assessment of Apple's Graphics Subsystem

About us

Yu Wang
Co-founder && CEO at Cyberserval
keenjoy95@gmail.com

Weiteng Chen
Microsoft Research Redmond
weitengchencc@gmail.com

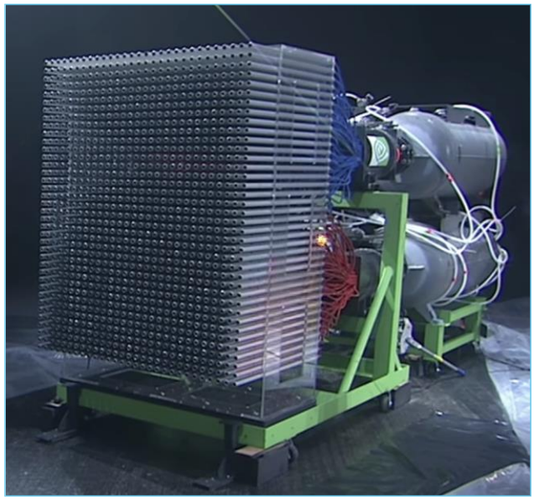
A Quick Introduction to (Apple) Graphics Subsystem

A quick introduction to GPU

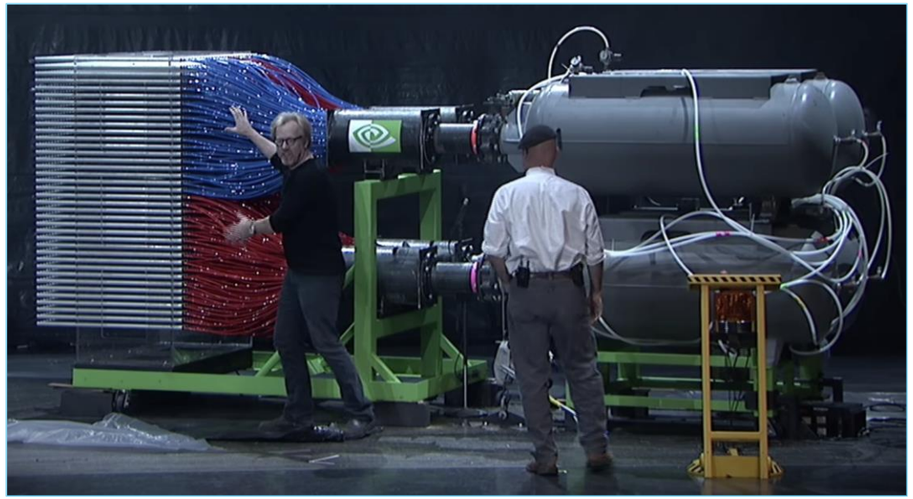
Rather than drafting your own GPU overview, check out how GPU manufacturers describe their products.

NVIDIA && Mythbusters:
"Mythbusters Demo GPU versus CPU"
https://web.archive.org/web/20201007031633/https://www.youtube.com/watch?v=-P28LKWTzrI

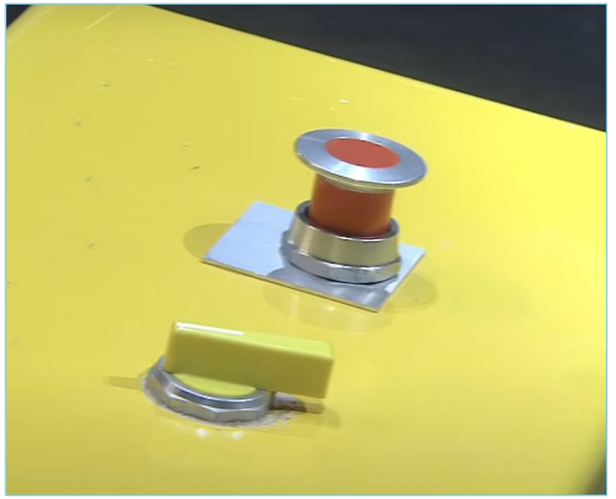
The key components



Framebuffer



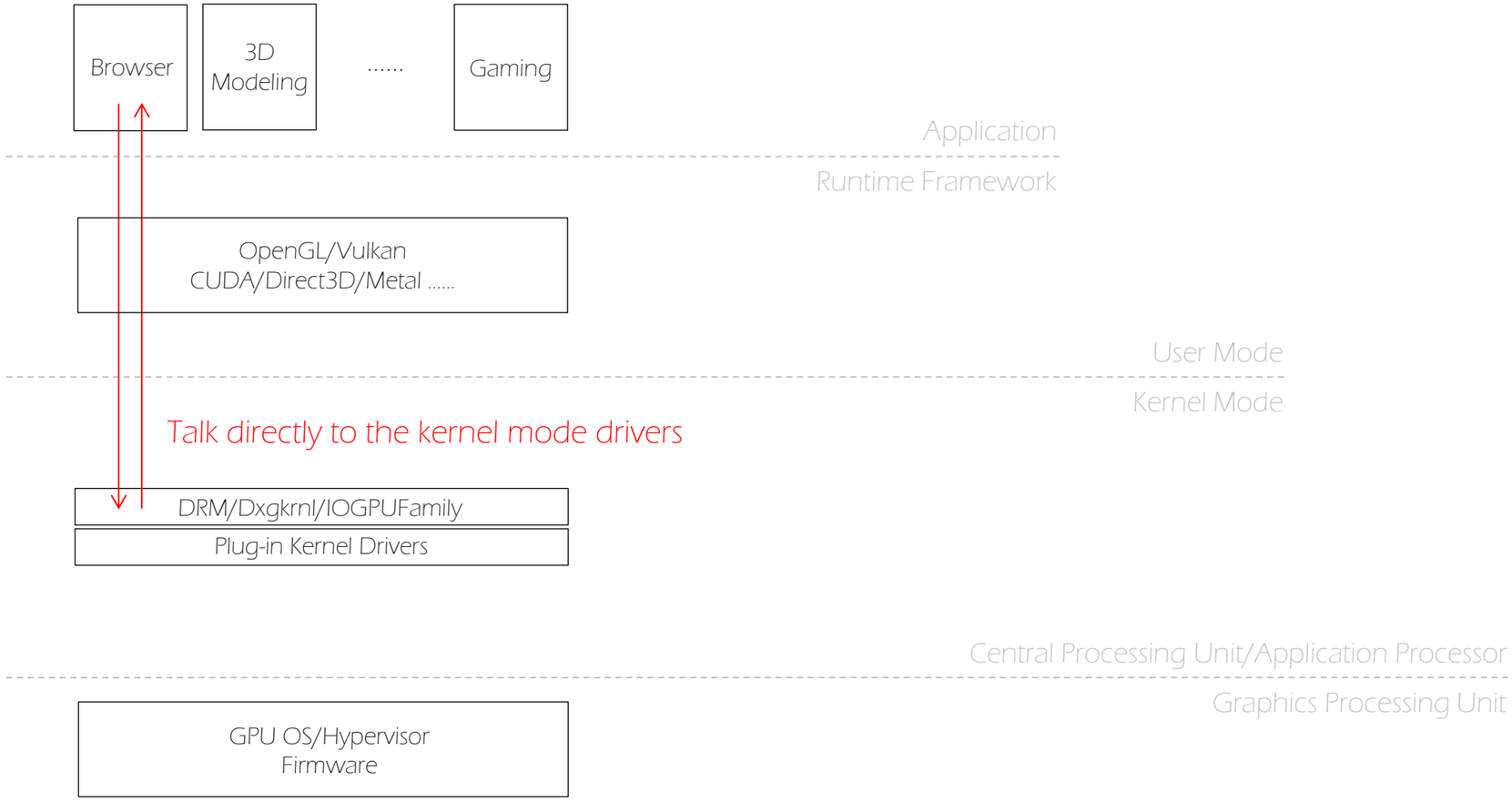
Command queue and data sharing (The 10-second countdown can be regarded as preparation for data and instructions)



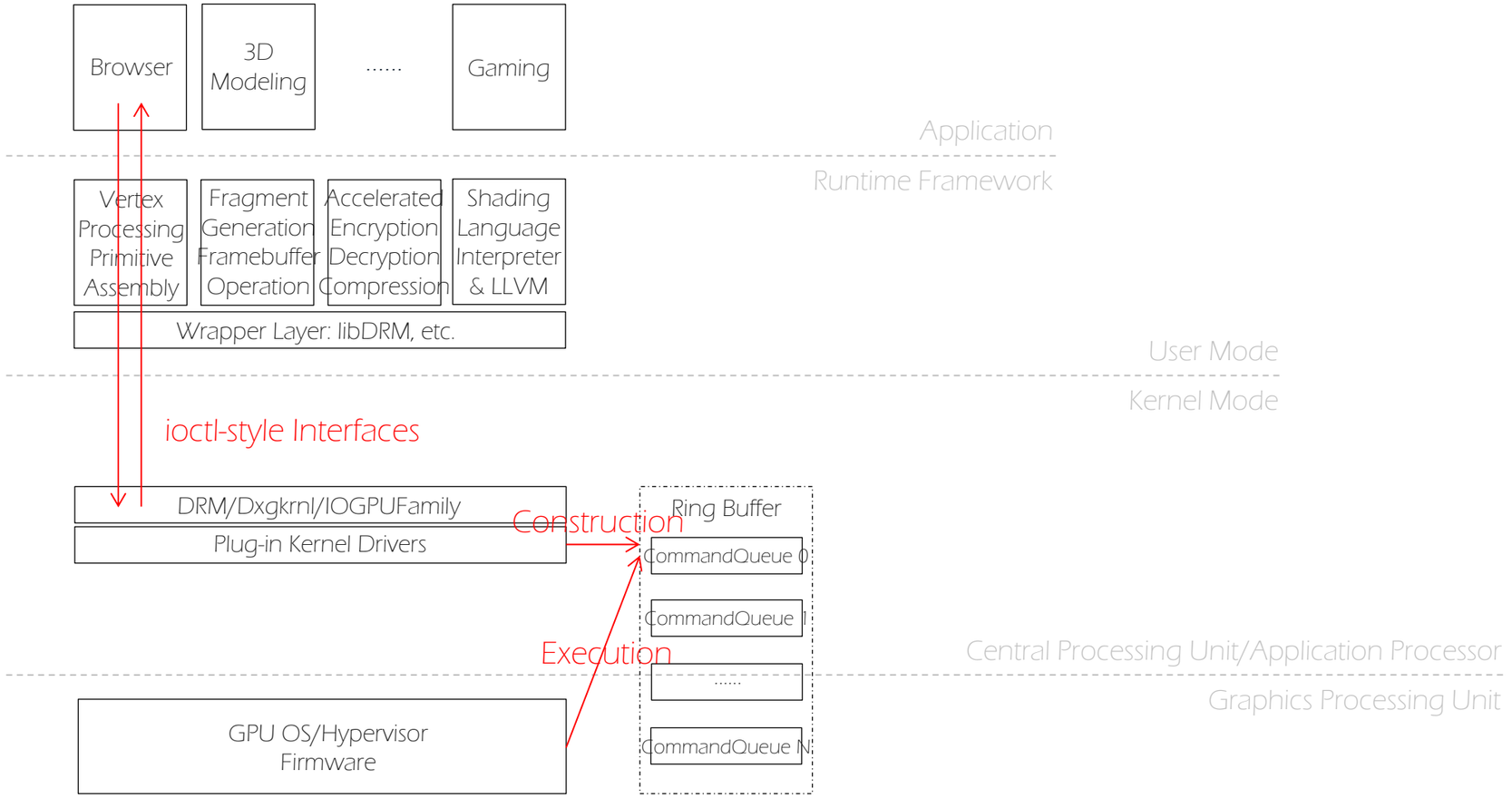
Command submission

Mythbusters Demo GPU versus CPU

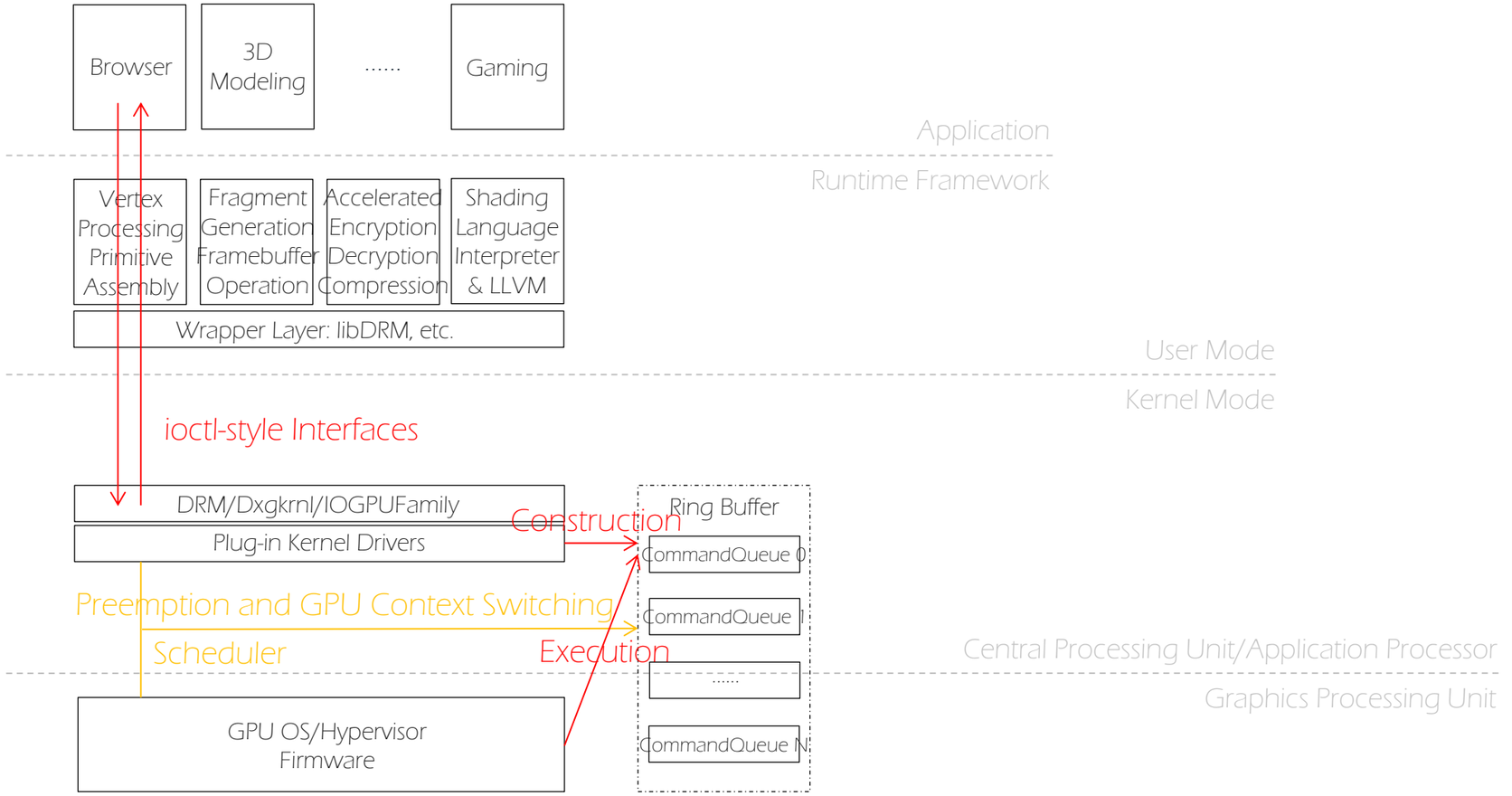
Let's start with the simplest form



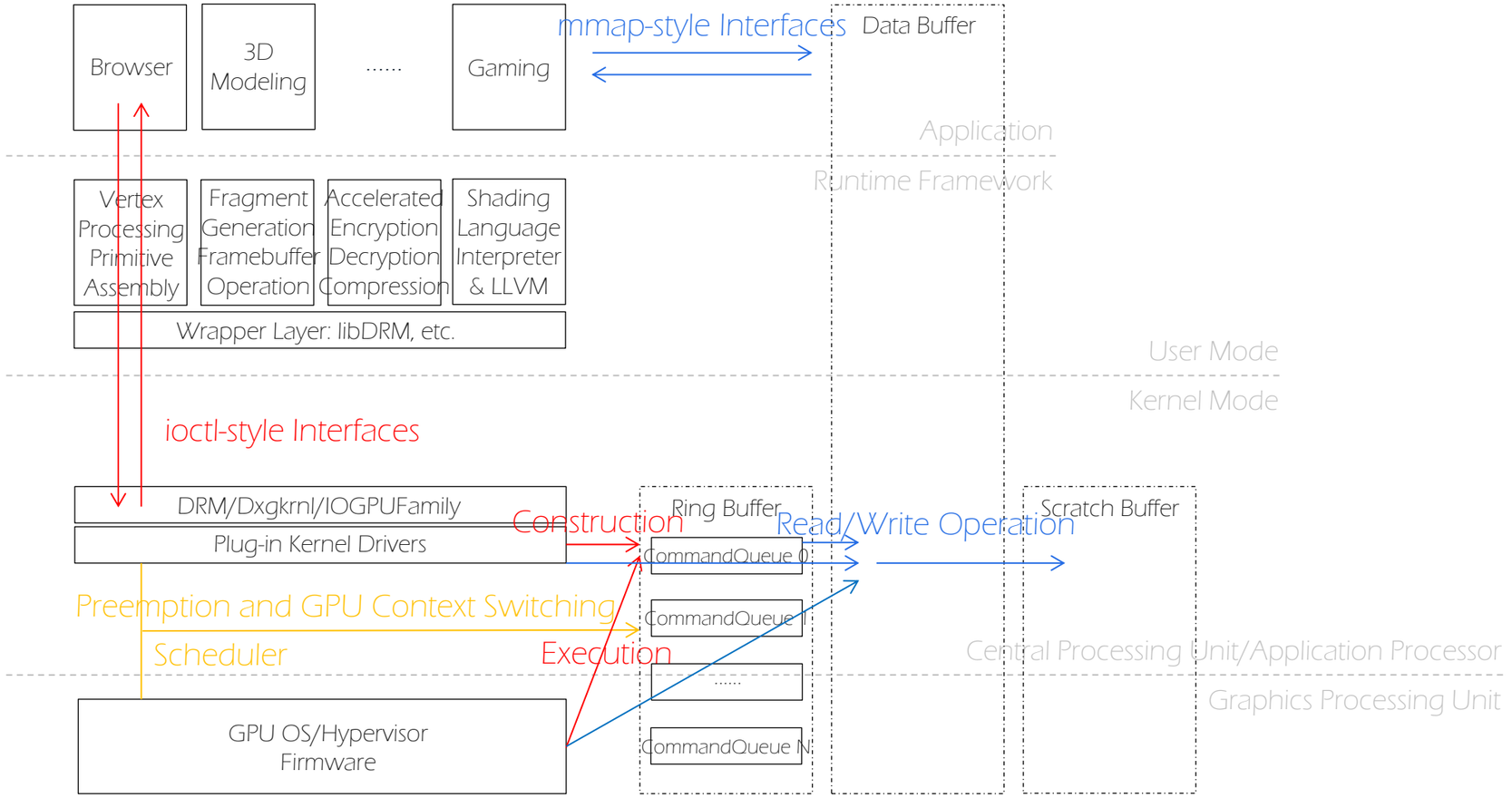
From command to ring buffer



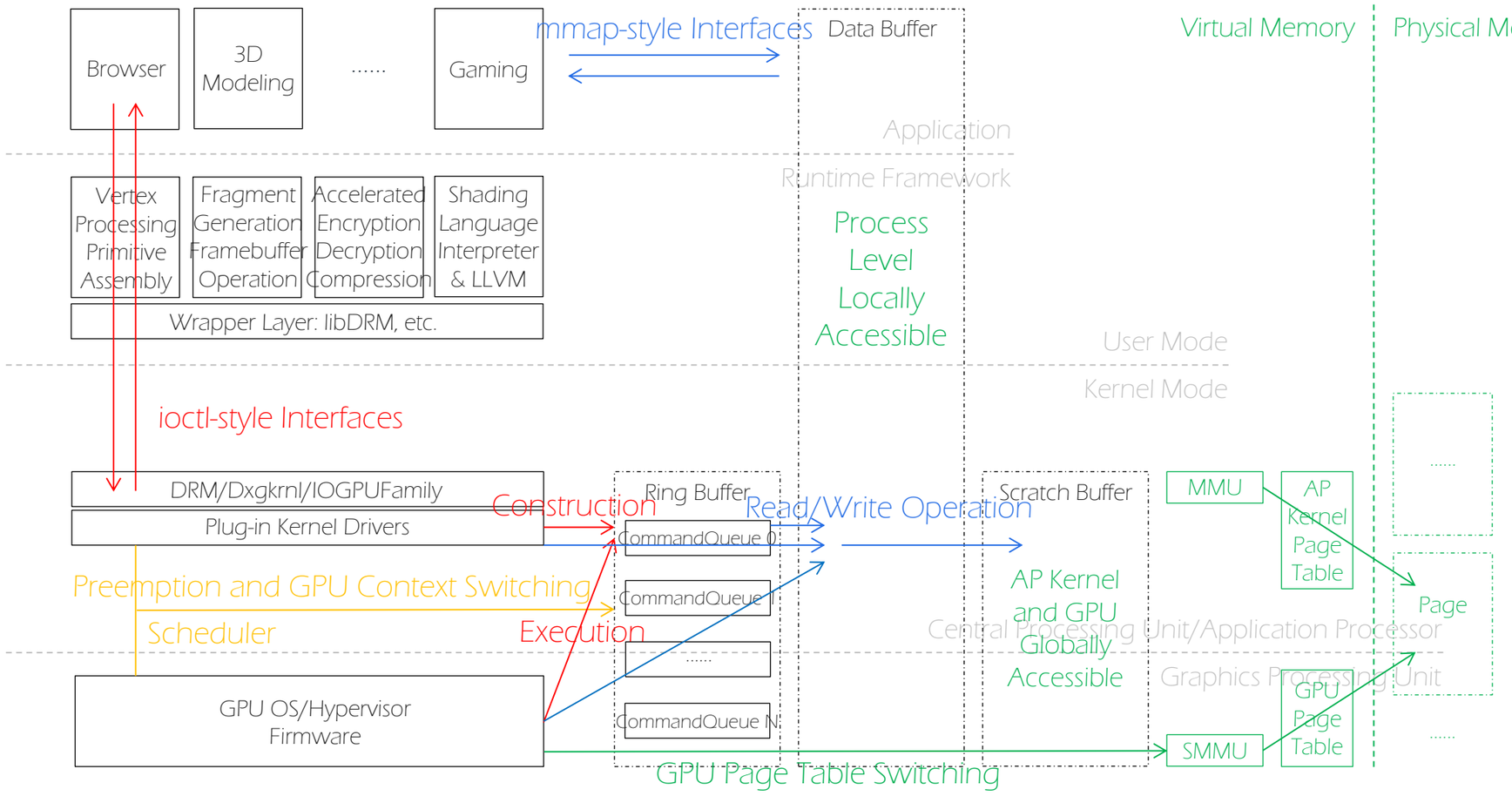
Introducing scheduling and context switching



Introducing data channel

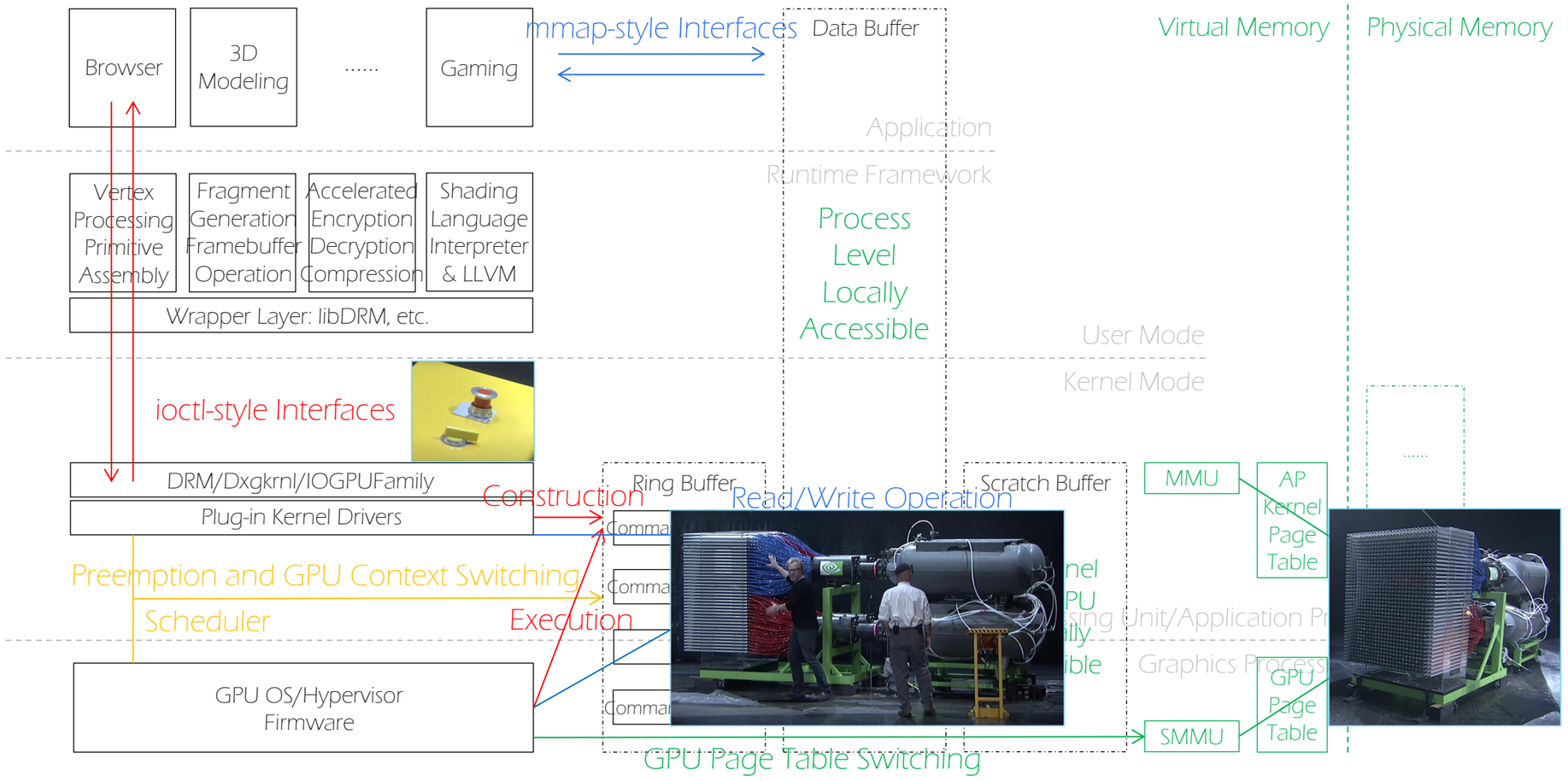


You can't just bolt on security engineering at the last minute



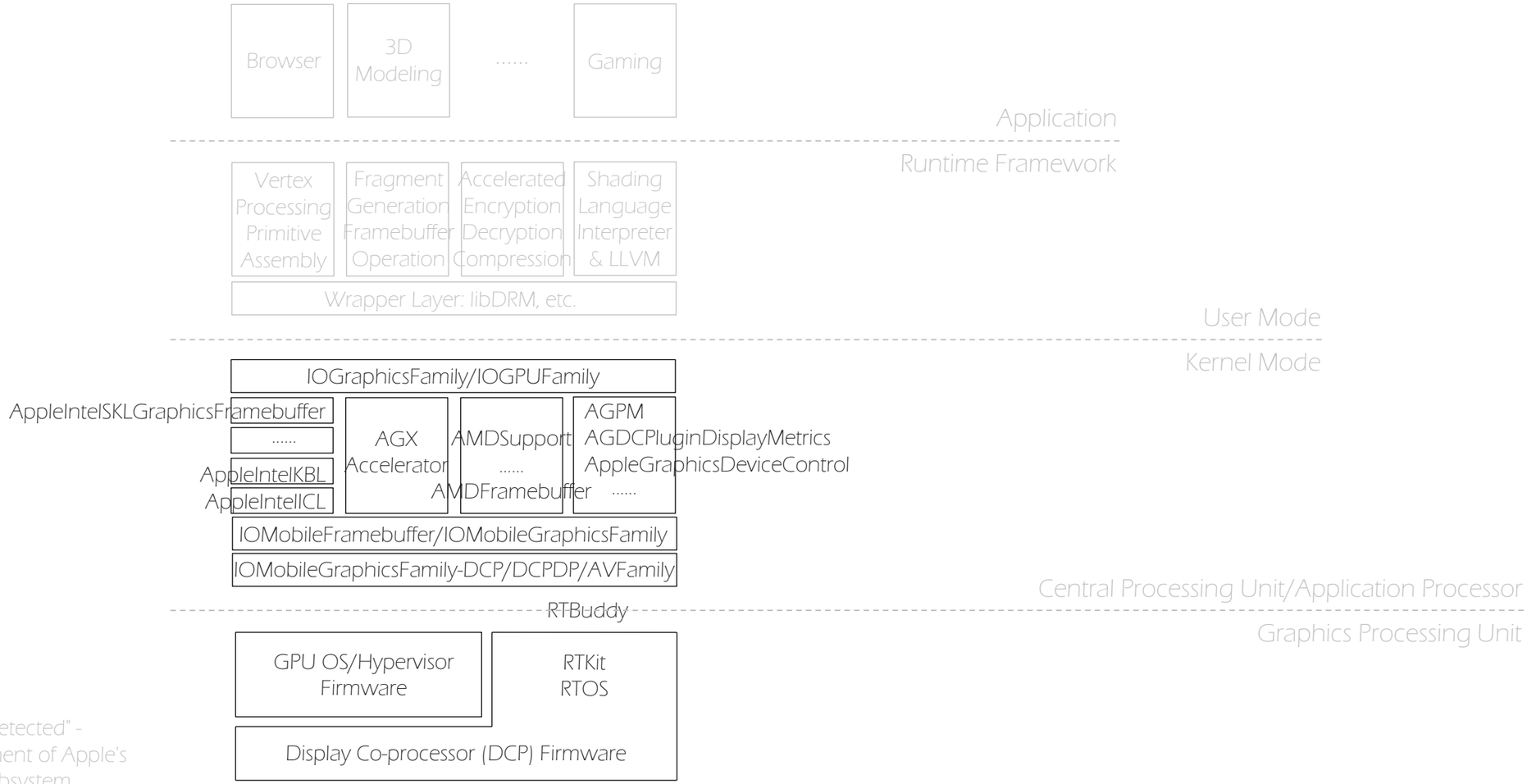
"Dead pixel detected" -
A Security Assessment of Apple's Graphics Subsystem

These background information are sufficient for today



Mythbusters Demo GPU versus CPU
&&
"Dead pixel detected" -
A Security Assessment of Apple's Graphics Subsystem

Apple GPU subsystem's kernel mode architecture



"Dead pixel detected" -
A Security Assessment of Apple's
Graphics Subsystem

Security Assessment of Apple's AMD and Intel-based GPU

Research background

CVE-2020-27915
ATIController::setupSharedSurface Arbitrary Memory Write Vulnerability
https://support.apple.com/en-us/102846

CVE-2022-22631
Out-of-bounds Read and Write Vulnerabilities in AGDCPluginDisplayMetrics Handlers
https://support.apple.com/en-us/102882

CVE-2022-22661
AppleIntelMEUserClient::start/AppleIntelMEUserClient::stop, An Out-of-bounds Write Vulnerability caused by Type Confusion
https://support.apple.com/en-us/102882

Case study of CVE-2020-27915

```
Process 1 stopped
* thread #1, stop reason = signal SIGSTOP
    frame #0: 0xffffff7fae144193 AMDSupport`ATIController::setupSharedSurface(AGDCMultiLinkConfig_t*,
ScanOutMetaInfo*) + 2339
AMDSupport`ATIController::setupSharedSurface:
->  0xffffff7fae144193 <+2339>: movb   %dil, -0x578(%rbp,%rcx)
    0xffffff7fae14419b <+2347>: movq   -0x48(%rbp), %rcx
    0xffffff7fae14419f <+2351>: movslq -0x600(%rbp), %rdx
    0xffffff7fae1441a6 <+2358>: imulq  $0xa8, %rdx, %rdx

(lldb) register read
 General Purpose Registers:
        rax = 0x0000000000000000
        rbx = 0xffffffa09ca93ab8
        rcx = 0xffffffffdeadbeef
        rbp = 0xffffffa09ca93860
        rsp = 0xffffffa09ca93220
        rip = 0xffffff7fae144193  AMDSupport`ATIController::setupSharedSurface(AGDCMultiLinkConfig_t*,
ScanOutMetaInfo*) + 2339
        ......
```

Case study of CVE-2022-22631

```
Process 1 stopped
* thread #1, stop reason = EXC_BAD_ACCESS (code=2, address=0xd2285184)
    frame #0: 0xffffff7f9fd44aa9
->  0xffffff7f9fd44aa9: movb   $0x3, 0x14(%r15)

(lldb) register read
 General Purpose Registers:
        r15 = 0xffffffb0d2285170
        rip = 0xffffff7f9fd44aa9
        ......
(lldb) memory read 0xffffffb0d2285170+0x14
0xffffffb0d2285184: de ad ca fe 41 41 41 41 41 41 41 41 41 41 41 41  ....AAAAAAAAAAAA
(lldb) bt
* thread #1, stop reason = EXC_BAD_ACCESS (code=2, address=0xd2285184)
  * frame #0: 0xffffff7f9fd44aa9
    frame #1: 0xffffff7f9fd52eed
    frame #2: 0xffffff7f9fd5386f
    frame #3: 0xffffff7f9fd50fff
    frame #4: 0xffffff8006e2731b  kernel`is_io_connect_method + 859
    ......
```

Case study of CVE-2022-22661

```
[(lldb) di -s 0xffffff7fadc128fb
AppleIntelICLLPGraphicsFramebuffer`AppleIntelMEClientController::doCmdAction:
    0xffffff7fadc128fb <+975>: movq   %rax, 0x120(%rbx)   // This instruction writes the created queue to the offset 0x120 of the object, but the size of the object is only 0x40.
->  0xffffff7fadc12902 <+982>: int3
    0xffffff7fadc12903 <+983>: sbbb   %al, (%rax,%rax)
    0xffffff7fadc12906 <+986>: addb   %al, -0x7f(%rcx)
    0xffffff7fadc12909 <+989>: cld
    0xffffff7fadc1290a <+990>: addl   %eax, (%rcx)
    0xffffff7fadc1290c <+992>: addb   %al, (%rax)
    0xffffff7fadc1290e <+994>: movq   -0x48(%rbp), %rbx
    0xffffff7fadc12912 <+998>: jne    0xffffff7fadc12f32        ; <+2566>
[(lldb) register read
General Purpose Registers:
       rax = 0xffffff950f816c80   // The RAX register is the newly created queue with size 0x40.
       rbx = 0xffffff950ea53180   // The RBX register will be written out of bounds.
       rcx = 0xffffff86a99d6690
       rdx = 0x0000000000000040
       rdi = 0x0000000000000040
       rsi = 0x0000000000000040
       rbp = 0xffffffd083db3b00
       rsp = 0xffffffd083db3aa0
        r8 = 0xffffff8043052f78
        r9 = 0x0000000000000000
       r10 = 0xffffff99db29abb0
       r11 = 0xffffffa042cfc82c
       r12 = 0x0000000000000100
       r13 = 0xffffffd083db3b80
       r14 = 0x00000000e00002c7
       r15 = 0xffffff950f793000
       rip = 0xffffff7fadc12902  AppleIntelICLLPGraphicsFramebuffer`AppleIntelMEClientController::doCmdAction(MECommand_t, void*, void*, void*) + 982
    rflags = 0x0000000000000202
        cs = 0x0000000000000008
        fs = 0x00000000ffff0000
        gs = 0x000000000ea50000

[(lldb) memory read 0xffffff950ea53180 0xffffff950ea53180+0x130 -fx -s8
0xffffff950ea53180: 0x0000000000000000 0xffffff8b77669a10
0xffffff950ea53190: 0xffffff950f816c80 0xffffff86a99d6690
0xffffff950ea531a0: 0x0000000000000000 0x0000000000000000
0xffffff950ea531b0: 0x0000000000000000 0x0000000000000000
0xffffff950ea531c0: 0x0000000000000000 0x0000000000000000
0xffffff950ea531d0: 0x0000000000000000 0x0000000000000000
0xffffff950ea531e0: 0x0000000000000000 0x0000000000000000
0xffffff950ea531f0: 0x0000000000000000 0x0000000000000000
0xffffff950ea53200: 0x6c7070612e6d6f63 0x72756f7365522e65
0xffffff950ea53210: 0x00006b726f466563 0x0000000000025ac4
0xffffff950ea53220: 0x0000000000026000 0x0000000000000000
0xffffff950ea53230: 0x0000000000000001 0x0000000000000001
0xffffff950ea53240: 0x74736e6967756c70 0x0000000000657461
0xffffff950ea53250: 0x000000000000024a 0x0000000000001000
0xffffff950ea53260: 0x0000000000000004 0x000000000000024a
0xffffff950ea53270: 0x0000000000000000 0x0000000000000000
0xffffff950ea53280: 0x74736e6967756c70 0x0000000000657461
0xffffff950ea53290: 0x000000000000024a 0x0000000000001000
0xffffff950ea532a0: 0xffffff950f816c80 0x000000000000024a   // Out-of-bounds write vulnerability!
[(lldb) bt
* thread #1, stop reason = breakpoint 1.1
  * frame #0: 0xffffff7fadc12902 AppleIntelICLLPGraphicsFramebuffer`AppleIntelMEClientController::doCmdAction(MECommand_t, void*, void*, void*) + 982
    frame #1: 0xffffff801605a160 kernel`IOCommandGate::runAction(this=0xffffff8b76756dc0, inAction=<unavailable>, arg0=0x0000000000000100, arg1=<unavailable>, arg2=<unavailable>, arg3=<unavailable>)(OSObject*, void
*, void*, void*, void*), void*, void*, void*, void*) at IOCommandGate.cpp:245:8 [opt]
    frame #2: 0xffffff801609462c kernel`::is_io_connect_method(connection=<unavailable>, selector=<unavailable>, scalar_input=<unavailable>, scalar_inputCnt=<unavailable>, inband_input=<unavailable>, inband_inputCn
t=<unavailable>, ool_input=0, ool_input_size=0, inband_output="", inband_outputCnt=0xffffffa042cfc82c, scalar_output=0xffffffd083db3d20, scalar_outputCnt=0xffffffd083db3d1c, ool_output=0, ool_output_size=0xffffff99
db29abb0) at IOUserClient.cpp:4855:17 [opt]
    frame #3: 0xffffff801598eca4 kernel`_Xio_connect_method(InHeadP=<unavailable>, OutHeadP=0xffffffa042cfc800) at device_server.c:8389:18 [opt]
    frame #4: 0xffffff8015862e98 kernel`ipc_kmsg_send at ipc_kobject.c:474:3 [opt]
    frame #5: 0xffffff8015862c5c kernel`ipc_kmsg_send [inlined] ipc_kobject_server(port=<unavailable>, request=<unavailable>, option=3) at ipc_kobject.c:582:8 [opt]
    frame #6: 0xffffff8015862b81 kernel`ipc_kmsg_send(kmsg=<unavailable>, option=<unavailable>, send_timeout=0) at ipc_kmsg.c:2211:10 [opt]
    frame #7: 0xffffff80158798dd kernel`mach_msg_overwrite_trap(args=<unavailable>) at mach_msg.c:371:8 [opt]
    frame #8: 0xffffff80159bad1e kernel`mach_call_munger64(state=0xffffff950ff4ab20) at bsd_i386.c:708:24 [opt]
    frame #9: 0xffffff8015828246 kernel`hndl_mach_scall64 + 22
(lldb)
```

The latest kernel vulnerabilities

Case #1 - CVE-2025-24273
AppleIntelMEClientController::invalidateContentKey Arbitrary Kernel Memory Write Vulnerability
About the security content of macOS Sequoia 15.4
https://support.apple.com/en-us/122373
About the security content of macOS Sonoma 14.7.5
https://support.apple.com/en-us/122374

Case #2 - OE096411696648 3
****** Kernel Out-of-bounds Access Vulnerability

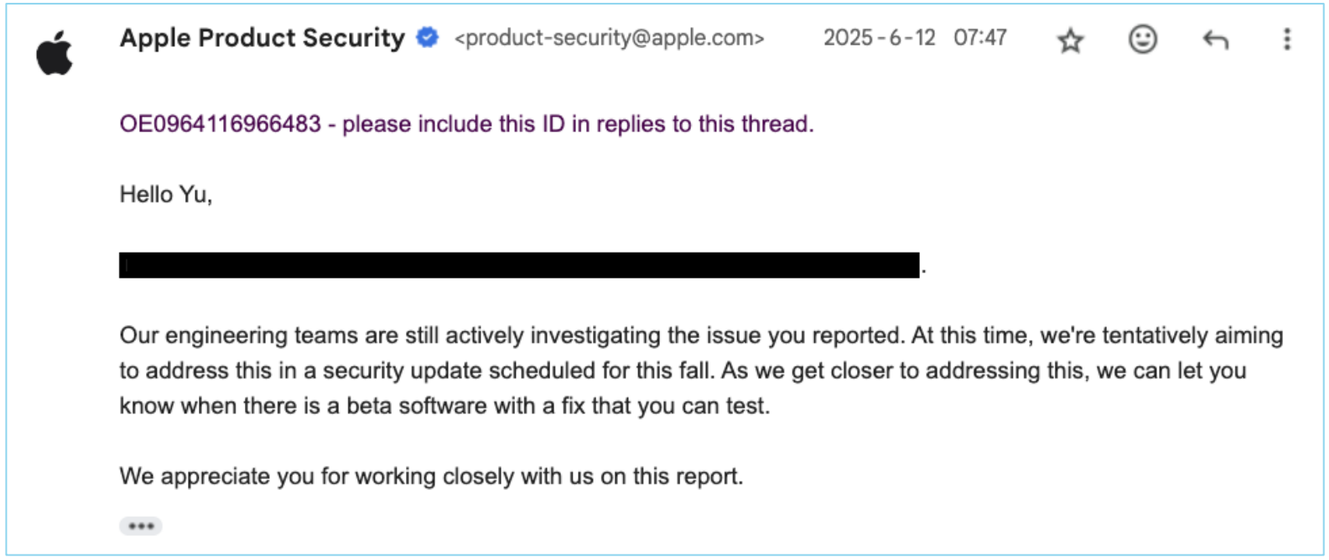
Case #1 - CVE-2025-24273

```
Process 1 stopped
* thread #1, stop reason = signal SIGSTOP
    frame #0: 0xffffff7faef872d0
AppleIntelICLLPGraphicsFramebuffer`AppleIntelMEClientController::invalidateContentKey(MECLIENT_DATA_T*, int,
bool) + 168
AppleIntelICLLPGraphicsFramebuffer`AppleIntelMEClientController::invalidateContentKey:
->  0xffffff7faef872d0 <+168>: movl   $0x0, 0x2568(%rbx,%rcx,4)
    0xffffff7faef872db <+179>: shlq   $0x4, %rcx
    0xffffff7faef872df <+183>: xorl   %eax, %eax
    0xffffff7faef872e1 <+185>: movq   %rax, 0x25b0(%rbx,%rcx)
Target 1: (kernel) stopped.

(lldb) register read
General Purpose Registers:
       rbx = 0xffffff90a7570000
       rcx = 0x0000000041414141
       rip = 0xffffff7faef872d0
AppleIntelICLLPGraphicsFramebuffer`AppleIntelMEClientController::invalidateContentKey(MECLIENT_DATA_T*, int,
bool) + 168
       ......
```

Case #2 - OE0964116966483

This vulnerability will not be fixed until this fall and will take approximately one year. In my experience, partially refactoring the module to eliminate these attack surfaces usually takes this long.



Apple Product Security <product-security@apple.com> 2025-6-12 07:47

OE0964116966483 - please include this ID in replies to this thread.

Hello Yu,

[redacted].

Our engineering teams are still actively investigating the issue you reported. At this time, we're tentatively aiming to address this in a security update scheduled for this fall. As we get closer to addressing this, we can let you know when there is a beta software with a fix that you can test.

We appreciate you for working closely with us on this report.

•••

Apple Product Security, 06.12.2025

Security Assessment of Apple Graphics Accelerator (AGX) GPU

Research background

CVE-2020-3837
iOS/macOS: OOB Timestamp Write in IOAccelCommandQueue2::processSegmentKernelCommand()
https://project-zero.issues.chromium.org/issues/42451084

CVE-2021-30735
Exploiting Intel Graphics Kernel Extensions on macOS
https://blog.ret2.io/2022/06/29/pwn2own-2021-safari-sandbox-intel-graphics-exploit/

CVE-2022-32947
I hacked macOS!!!
https://asahilina.net/agx-exploit/

The latest kernel vulnerabilities

Case #3 - CVE-2024-44197
IOGPUDeviceUserClient::s_create_notificationqueue/s_destroy_notificationqueue
Notification Queue Out-of-bounds Access Vulnerability
About the security content of macOS Sequoia 15.1
https://support.apple.com/en-us/121564

Case #4 - CVE-2025-24257
IOGPUResource::newResourceGroup Kernel Out-of-bounds Read and Write Vulnerability
About the security content of iOS 18.4 and iPadOS 18.4
https://support.apple.com/en-us/122371
About the security content of macOS Sequoia 15.4
https://support.apple.com/en-us/122373

Case #3 - CVE-2024-44197

During reverse engineering, I found that functions like create_notificationqueue accept at least two critical parameters, numEntries and entrySize, but such functions do not strictly check the validity of these parameters.

```
1  __int64 __fastcall IOGPUDevice::create_notificationqueue(
2          __int64 this,
3          task *numEntries,
4          unsigned int entrySize,
5          __int64 output_buffer)
6  {
7    IOGPUBlockFencePort::MetaClass *object; // x0
8    IOGPUNotificationQueue *queue; // x20
9    int v8; // w0
10
11   object = IOGPUNotificationQueue::withEntries(*(this + 0x50), numEntries, entrySize, output_buffer);
12   if ( object )
13   {
14     queue = object;
15     v8 = IOGPUNamespace::addObject(*(this + 0x90), object);
16     if ( v8 )
17     {
18       *(output_buffer + 8) = v8;
19       *output_buffer = IOGPUNotificationQueue::get_notification_queue_address(queue);
```

macOS Ventura 13.5 Beta (22G5027e)

The panic

```
Process 1 stopped
* thread #1, stop reason = signal SIGSTOP
    frame #0: 0xfffffe0020f779f8 kernel.release.t8122`DebuggerTrapWithState(db_op=DBOP_PANIC,
db_message="panic", db_panic_str="%s %s -- exit reason namespace %d subcode 0x%llx description: %.800s",
db_panic_args=0xfffffe84f9cef9e8, db_panic_options=8224, db_panic_data_ptr=0x0000000000000000,
db_proceed_on_sync_failure=1, db_panic_caller=18446741875244973128, db_panic_initiator=0x0000000000000000) at
debug.c:819:2 [opt]
Target 1: (kernel.release.t8122) stopped.
warning: kernel.release.t8122 was compiled with optimization - stepping may behave oddly; variables may not be
available.

(lldb) register read
General Purpose Registers:
        x0 = 0x0000000000000003
        x1 = 0xfffffe001f91e478  "panic"
        lr = 0xfffffe0020f76fcc  kernel.release.t8122`panic_trap_to_debugger + 744 [inlined]
panic_spin_forever at debug.c:1346:3
  kernel.release.t8122`panic_trap_to_debugger + 744 at debug.c:1336:2
        pc = 0xfffffe0020f779f8  kernel.release.t8122`DebuggerTrapWithState + 76 at debug.c:819:2
        ......
```

Patch for CVE-2024-44197

The patch for the vulnerability is straightforward.

```
1  __int64 __fastcall IOGPUDevice::create_notificationqueue(
2          __int64 this,
3          task *numEntries,
4          __int64 entrySize,
5          __int64 output_buffer)
6  {
7    IOGPUObject *object; // x0
8    IOGPUNotificationQueue *queue; // x20
9    int v8; // w0
10   __int64 result; // x0
11
12   if ( (numEntries - 0x2001) < 0xFFFFE000 || (entrySize - 0x29) <= 0xFFFFFFD7 )
13   {
14     _os_log_internal(
15       &dword_0,
16       &_os_log_default,
17       OS_LOG_TYPE_FAULT,
18       "%s: Invalid arguments : numEntries = %u, entrySize = %u \n",
19       "IOReturn IOGPUDevice::create_notificationqueue(uint32_t, uint32_t, sIOGPUDeviceNotificationQueueData *)",
20       numEntries,
21       entrySize);
22     return 0xE00002C2LL;
23   }
24   else
25   {
26     object = IOGPUNotificationQueue::withEntries(*(this + 80), numEntries, entrySize, output_buffer);
27     if ( object )
28     {
29       queue = object;
30       v8 = IOGPUNamespace::addObject(*(this + 144), object);
```

macOS Sequoia 15.1 Beta (24B83)

The confusing security advisory

IOGPUFamily

Available for: macOS Sequoia

Impact: A malicious app may be able to cause a denial-of-service

Description: The issue was addressed with improved memory handling.

CVE-2024-44197: Wang Yu of Cyberserval

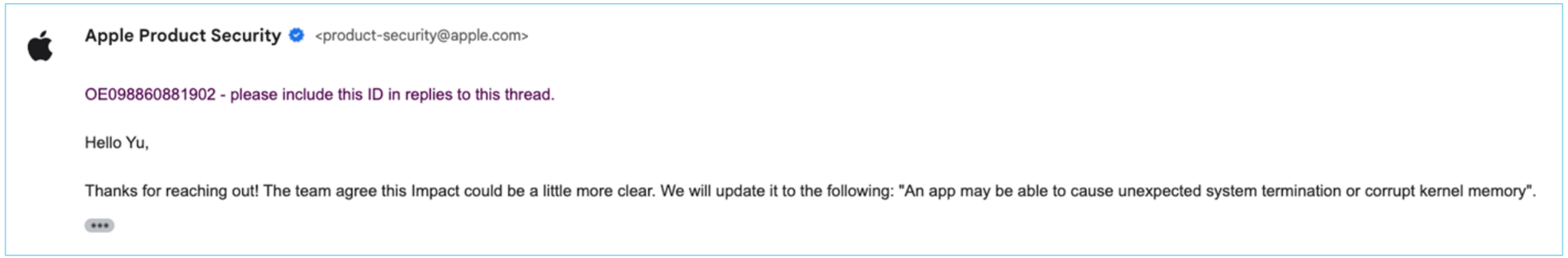
https://support.apple.com/en-us/121564

History of NULL Pointer Dereferences on macOS
https://afine.com/history-of-null-pointer-dereferences-on-macos/

Case Study: IOMobileFramebuffer NULL Pointer Dereference
https://afine.com/case-study-iomobileframebuffer-null-pointer-dereference/

Response from Apple's product security team

I have also discussed this issue with Apple SRC team, and they have promised to modify the description for CVE-2024-44197/OE098860881902.



Apple Product Security <product-security@apple.com>

OE098860881902 - please include this ID in replies to this thread.

Hello Yu,

Thanks for reaching out! The team agree this Impact could be a little more clear. We will update it to the following: "An app may be able to cause unexpected system termination or corrupt kernel memory".

•••

Apple Product Security, 04.17.2025

Case #4 - CVE-2025-24257

The advisory this time is clear. It's a kernel memory write vulnerability that affects both iOS and macOS.

Through significant effort, I gradually transformed a seemingly unusable raw panic into a kernel memory write vulnerability to demonstrate its exploitability.

IOGPUFamily

Available for: iPhone XS and later, iPad Pro 13-inch, iPad Pro 12.9-inch 3rd generation and later, iPad Pro 11-inch 1st generation and later, iPad Air 3rd generation and later, iPad 7th generation and later, and iPad mini 5th generation and later

Impact: An app may be able to cause unexpected system termination or write kernel memory

Description: An out-of-bounds write issue was addressed with improved input validation.

CVE-2025-24257: Wang Yu of Cyberserval

https://support.apple.com/en-us/122371

Initially, I only observed some strange kernel panics

```
Process 1 stopped
* thread #1, stop reason = signal SIGSTOP
    frame #0: 0xfffffe002e0d3648 kernel.release.t8122`DebuggerTrapWithState(db_op=DBOP_PANIC,
db_message="panic", db_panic_str="%s at pc 0x%016llx, lr 0x%016llx (saved state: %p%s)\n\t  x0:  0x%016llx x1:
0x%016llx  x2:  0x%016llx  x3:  0x%016llx\n\t  x4:  0x%016llx x5:  0x%016llx  x6:  0x%016llx  x7:
0x%016llx\n\t  x8:  0x%016llx x9:  0x%016llx  x10: 0x%016llx  x11: 0x%016llx\n\t  x12: 0x%016llx x13:
0x%016llx  x14: 0x%016llx  x15: 0x%016llx\n\t  x16: 0x%016llx x17: 0x%016llx  x18: 0x%016llx  x19:
0x%016llx\n\t  x20: 0x%016llx x21: 0x%016llx  x22: 0x%016llx  x23: 0x%016llx\n\t  x24: 0x%016llx x25:
0x%016llx  x26: 0x%016llx  x27: 0x%016llx\n\t  x28: 0x%016llx fp:  0x%016llx  lr:  0x%016llx  sp:
0x%016llx\n\t  pc:  0x%016llx cpsr: 0x%08x         esr: 0x%016llx  far: 0x%016llx\n",
db_panic_args=0xfffffe8f1896f028, db_panic_options=0, db_panic_data_ptr=0x0000000000000000,
db_proceed_on_sync_failure=1, db_panic_caller=18446741875467706024, db_panic_initiator=0x0000000000000000) at
debug.c:834:2 [opt]
Target 0: (kernel.release.t8122) stopped.
warning: kernel.release.t8122 was compiled with optimization - stepping may behave oddly; variables may not be
available.

(lldb) di -p
IOGPUFamily`IOGPUGroupMemory::remove_memory_object:
->  0xfffffe0030693724 <+292>: ldr    w11, [x11, w10, uxtw #2]
```

We need to gradually escalate the problem to achieve arbitrary memory access

```
(lldb) di -p
IOGPUFamily`IOGPUGroupMemory::removeMemoryFromResourceMap:
->  0xfffffe00193f4f40 <+96>:  ldr    x8, [x9, x8]
    0xfffffe00193f4f44 <+100>: ldr    x1, [x8, #0x28]
    0xfffffe00193f4f48 <+104>: mov    x0, x20
    0xfffffe00193f4f4c <+108>: mov    x2, x19

(lldb) register read
General Purpose Registers:
        x8 = 0x0000000000067020
        x9 = 0xfffffe24d47e0040
        fp = 0xfffffe3eecb03710
        lr = 0xfffffe00193f4f0c
IOGPUFamily`IOGPUGroupMemory::removeMemoryFromResourceMap(IOGPUCountedMap<unsigned long long, IOGPUResource*,
IOGPUResourceCountedMapBucket, IOGPUIOLibAllocatorPolicy>*, bool) + 44
        sp = 0xfffffe3eecb036d0
        pc = 0xfffffe00193f4f40
IOGPUFamily`IOGPUGroupMemory::removeMemoryFromResourceMap(IOGPUCountedMap<unsigned long long, IOGPUResource*,
IOGPUResourceCountedMapBucket, IOGPUIOLibAllocatorPolicy>*, bool) + 96
        ......
```

The vulnerable function does check parameters, but inadequately

```
63  else
64  {
65    _os_log_internal(
66      &dword_0,
67      &_os_log_default,
68      OS_LOG_TYPE_FAULT,
69      "%s: newResourceGroup bad initial capacity: %d\n",
70      "static OSPtr<IOGPUResource> IOGPUResource::newResourceGroup(IOGPU *, IOGPUDevice *, uint32_t)",
71      v4);
72    return 0LL;
73  }
```

macOS Sequoia 15.2 RC2 (24C100)

Half a year later, the patch for CVE-2025-24257 landed

```
63  else
64  {
65    _os_log_internal(
66      &dword_0,
67      &_os_log_default,
68      OS_LOG_TYPE_FAULT,
69      "%s: newResourceGroup bad initial capacity: %d\n",
70      "static OSPtr<IOGPUResource> IOGPUResource::newResourceGroup(IOGPU *, IOGPUDevice *, uint32_t)",
71      v4);
72    return 0LL;
73  }
```

macOS Sequoia 15.2 RC2 (24C100)

```
69  else
70  {
71    _os_log_internal(
72      &dword_0,
73      &_os_log_default,
74      OS_LOG_TYPE_FAULT,
75      "%s: newResourceGroup failed - initial capacity (%u) less than minimum required (%u)\n",
76      "static OSPtr<IOGPUResource> IOGPUResource::newResourceGroup(IOGPU *, IOGPUDevice *, uint32_t)",
77      a3,
78      64LL);
79    return 0LL;
80  }
```

macOS Tahoe 26.0 Beta (25A5279m)

Hmmm...

Good, "less than" works, but what happens when greater than?

Bypassing the patch on the macOS Tahoe 26.0 Beta (25A5279m)

```
Process 1 stopped
* thread #1, stop reason = signal SIGSTOP
    frame #0: 0xfffffe002eebda9c kernel.release.t8122`DebuggerTrapWithState(db_op=DBOP_PANIC, db_message="panic", db_panic_str="%s at pc 0x%016llx, lr 0x%016llx (saved state: %p%s)\n\t  x0:  0x%016llx x1:  0x%016llx  x2:  0x%016llx  x3:  0x%016llx\n\t  x4:  0x%016llx x5:  0x%016llx  x6:  0x%016llx  x7:  0x%016llx\n\t  x8:  0x%016llx x9:  0x%016llx  x10: 0x%016llx  x11: 0x%016llx\n\t  x12: 0x%016llx x13: 0x%016llx  x14: 0x%016llx  x15: 0x%016llx\n\t  x16: 0x%016llx x17: 0x%016llx  x18: 0x%016llx  x19: 0x%016llx\n\t  x20: 0x%016llx x21: 0x%016llx  x22: 0x%016llx  x23: 0x%016llx\n\t  x24: 0x%016llx x25: 0x%016llx  x26: 0x%016llx  x27: 0x%016llx\n\t  x28: 0x%016llx fp:  0x%016llx  lr:  0x%016llx  sp:  0x%016llx\n\t  pc:  0x%016llx cpsr: 0x%08x         esr: 0x%016llx  far: 0x%016llx\n", db_panic_args=0xfffffe6770276f68, db_panic_options=0, db_panic_data_ptr=0x0000000000000000, db_proceed_on_sync_failure=1, db_panic_caller=18446741875482508000, db_panic_initiator=0x0000000000000000) at debug.c:834:2 [opt]
Target 2: (kernel.release.t8122) stopped.
warning: kernel.release.t8122 was compiled with optimization - stepping may behave oddly; variables may not be available.
(lldb) register read
General Purpose Registers:
        x0 = 0xfffffe32b2942e00
        x1 = 0xffffffe271aee17b0
        x2 = 0x0000000000000000
        x3 = 0x0000000000000000
        x4 = 0x0000000000000000
        x5 = 0x0000000000020710
        x6 = 0x0000000000000000
        x7 = 0xfffffe003499ff66
        x8 = 0x0000000000000000
        x9 = 0x0000000000000000
       x10 = 0xfffffe32b2942e08
       x11 = 0x0000000000000004
       x12 = 0x0000000000001000
       x13 = 0x0000000000000000
       x14 = 0xffffffffffffffff
       x15 = 0x00000000ffdfffff
       x16 = 0xfffffe002ea4c170  IOGPUFamily`vtable for IOGPUGroupMemory + 72
       x17 = 0xfffcfe002ea4c170 (0xfffffe002ea4c170) IOGPUFamily`vtable for IOGPUGroupMemory + 72
       x18 = 0x0000000000000000
       x19 = 0x0000000000000000
       x20 = 0xfffffe214f5234c0
       x21 = 0xfffffe31af9c2300
       x22 = 0x0000000000000000
       x23 = 0x0000000000000001
       x24 = 0x0000000000000000
       x25 = 0x0000000000000058
       x26 = 0x0000000000000001
       x27 = 0x0000000000000000
       x28 = 0x0000000000000000
        fp = 0xfffffe6770277690
        lr = 0xfffffe003154e98c  IOGPUFamily`IOGPUGroupMemory::removeMemoryFromResourceMap(IOGPUCountedMap<unsigned long long, IOGPUResource*, IOGPUResourceCountedMapBucket, IOGPUIOLibAllocatorPolicy>*, bool) + 44
        sp = 0xfffffe6770277650
        pc = 0xfffffe003154e998  IOGPUFamily`IOGPUGroupMemory::removeMemoryFromResourceMap(IOGPUCountedMap<unsigned long long, IOGPUResource*, IOGPUResourceCountedMapBucket, IOGPUIOLibAllocatorPolicy>*, bool) + 56
      cpsr = 0x60401208

(lldb) bt
* thread #1, stop reason = signal SIGSTOP
    frame #0: 0xfffffe002eebda9c kernel.release.t8122`DebuggerTrapWithState(db_op=DBOP_PANIC, db_message="panic", db_panic_str="%s at pc 0x%016llx, lr 0x%016llx (saved state: %p%s)\n\t  x0:  0x%016llx x1:  0x%016llx  x2:  0x%016llx  x3:  0x%016llx\n\t  x4:  0x%016llx x5:  0x%016llx  x6:  0x%016llx  x7:  0x%016llx\n\t  x8:  0x%016llx x9:  0x%016llx  x10: 0x%016llx  x11: 0x%016llx\n\t  x12: 0x%016llx x13: 0x%016llx  x14: 0x%016llx  x15: 0x%016llx\n\t  x16: 0x%016llx x17: 0x%016llx  x18: 0x%016llx  x19: 0x%016llx\n\t  x20: 0x%016llx x21: 0x%016llx  x22: 0x%016llx  x23: 0x%016llx\n\t  x24: 0x%016llx x25: 0x%016llx  x26: 0x%016llx  x27: 0x%016llx\n\t  x28: 0x%016llx fp:  0x%016llx  lr:  0x%016llx  sp:  0x%016llx\n\t  pc:  0x%016llx cpsr: 0x%08x         esr: 0x%016llx  far: 0x%016llx\n", db_panic_args=0xfffffe6770276f68, db_panic_options=0, db_panic_data_ptr=0x0000000000000000, db_proceed_on_sync_failure=1, db_panic_caller=18446741875482508000, db_panic_initiator=0x0000000000000000) at debug.c:834:2 [opt]
    frame #1: 0xfffffe002eebd058 kernel.release.t8122`panic_trap_to_debugger(panic_format_str="%s at pc 0x%016llx, lr 0x%016llx (saved state: %p%s)\n\t  x0:  0x%016llx x1:  0x%016llx  x2:  0x%016llx  x3:  0x%016llx\n\t  x4:  0x%016llx x5:  0x%016llx  x6:  0x%016llx  x7:  0x%016llx\n\t  x8:  0x%016llx x9:  0x%016llx  x10: 0x%016llx  x11: 0x%016llx\n\t  x12: 0x%016llx x13: 0x%016llx  x14: 0x%016llx  x15: 0x%016llx\n\t  x16: 0x%016llx x17: 0x%016llx  x18: 0x%016llx  x19: 0x%016llx\n\t  x20: 0x%016llx x21: 0x%016llx  x22: 0x%016llx  x23: 0x%016llx\n\t  x24: 0x%016llx x25: 0x%016llx  x26: 0x%016llx  x27: 0x%016llx\n\t  x28: 0x%016llx fp:  0x%016llx  lr:  0x%016llx  sp:  0x%016llx\n\t  pc:  0x%016llx cpsr: 0x%08x         esr: 0x%016llx  far: 0x%016llx\n", panic_args=0xfffffe6770276f68, reason=0, ctx=0x0000000000000000, panic_options_mask=0, panic_data_ptr=0x0000000000000000, panic_caller=18446741875482508000, panic_initiator=0x0000000000000000) at debug.c:1394:2 [opt]
    frame #2: 0xfffffe002f7484e4 kernel.release.t8122`panic(str=<unavailable>) at debug.c:1161:2 [opt]
    frame #3: 0xfffffe002f753ae0 kernel.release.t8122`panic_with_thread_kernel_state(msg="Kernel data abort.", ss=0xfffffe6770277300) at sleh.c:935:2 [opt]
    frame #4: 0xfffffe002f01a7f0 kernel.release.t8122`handle_kernel_abort(state=0xfffffe6770277300, esr=<unavailable>, fault_addr=0, fault_code=<unavailable>, fault_type=1, expected_fault_handler=<unavailable>) at sleh.c:3485:2 [opt]
    frame #5: 0xfffffe002f018f7c kernel.release.t8122`sleh_synchronous [inlined] handle_abort(state=0xfffffe6770277300, esr=2516582406, fault_addr=0, inspect_abort=<unavailable>, handler=<unavailable>, expected_fault_handler=0x0000000000000000) at sleh.c:1845:2 [opt]
    frame #6: 0xfffffe002f018f68 kernel.release.t8122`sleh_synchronous(context=0xfffffe6770277300, esr=2516582406, far=0, did_initiate_panic_lockdown=<unavailable>) at sleh.c:1308:3 [opt]
    frame #7: 0xfffffe002ee678c0 kernel.release.t8122`fleh_synchronous + 44
  * frame #8: 0xfffffe003154e998 IOGPUFamily`IOGPUGroupMemory::removeMemoryFromResourceMap(IOGPUCountedMap<unsigned long long, IOGPUResource*, IOGPUResourceCountedMapBucket, IOGPUIOLibAllocatorPolicy>*, bool) + 56
    frame #9: 0xfffffe00315382d8 IOGPUFamily`IOGPUResource::free() + 272
    frame #10: 0xfffffe00315333d4 IOGPUFamily`IOGPUObject::release() const + 48
```

Security Assessment of IOMobileFrameBuffer (IOMFB)

The story behind IOMFB

The statistical data on IOMobileFrameBuffer vulnerabilities indicates that the competition between the offensive and defensive sides once reached a fever pitch.

According to publicly available records, a total of sixteen kernel vulnerabilities in IOMobileFrameBuffer have been reported throughout its history. Among these, four were actively exploited by APT groups (CVE-2021-30807, CVE-2021-30883, CVE-2021-30983, CVE-2022-22587), two were leveraged in iOS jailbreak tools (JailbreakMe 3.0 - CVE-2011-0227, Pangu 9 - CVE-2016-4654), and one was successfully utilized to win a security challenge competition (Tianfu Cup - CVE-2021-30983).

The historical landscape of IOMFB kernel vulnerabilities - in the first ten years

2011 - CVE-2011-0227 (Comex, JailbreakMe 3.0)

2012 - N/A
2013 - N/A
2014 - N/A

2015 - CVE-2015-1097 (Barak Gabai), CVE-2015-5843 (Filippo Bigarella)
2016 - CVE-2016-4654 (Tielei Wang - Team Pangu, Pangu 9)
2017 - CVE-2017-13879 (Apple)
2018 - CVE-2018-4335 (Brandon Azad)

2019 - N/A
2020 - N/A

The historical landscape of IOMFB kernel vulnerabilities - in recent years

2021 - CVE-2021-30807 (ITW APT attack / Saar Amar), CVE-2021-30883 (ITW APT attack / Tielei Wang - Team Pangu), CVE-2021-30983 (Tielei Wang - Team Pangu, Tianfu Cup Competition), CVE-2021-30985 (Tielei Wang - Team Pangu), CVE-2021-30991 (Tielei Wang - Team Pangu), CVE-2021-30996 (Saar Amar)

2022 - CVE-2022-22587 (ITW APT attack / Meysam Firouzi / Siddharth Aeri), CVE-2022-26768 (An Anonymous Researcher, Highly likely exploited by an ITW APT attack), CVE-2022-46690 (John Aakerblom), CVE-2022-46697 (John Aakerblom / Antonio Zekic)

2023 - N/A

2024 - Any ideas?

One more thing

However, no new IOMobileFrameBuffer kernel vulnerabilities have been disclosed since 2023.

In addition, it should be noted that by the end of 2024, several vulnerabilities were misclassified as IOMobileFramebuffer. In fact, they are firmware issues of the Display Co-processor. These vulnerabilities were submitted by Ye Zhang from the Baidu Security Labs.

Research background

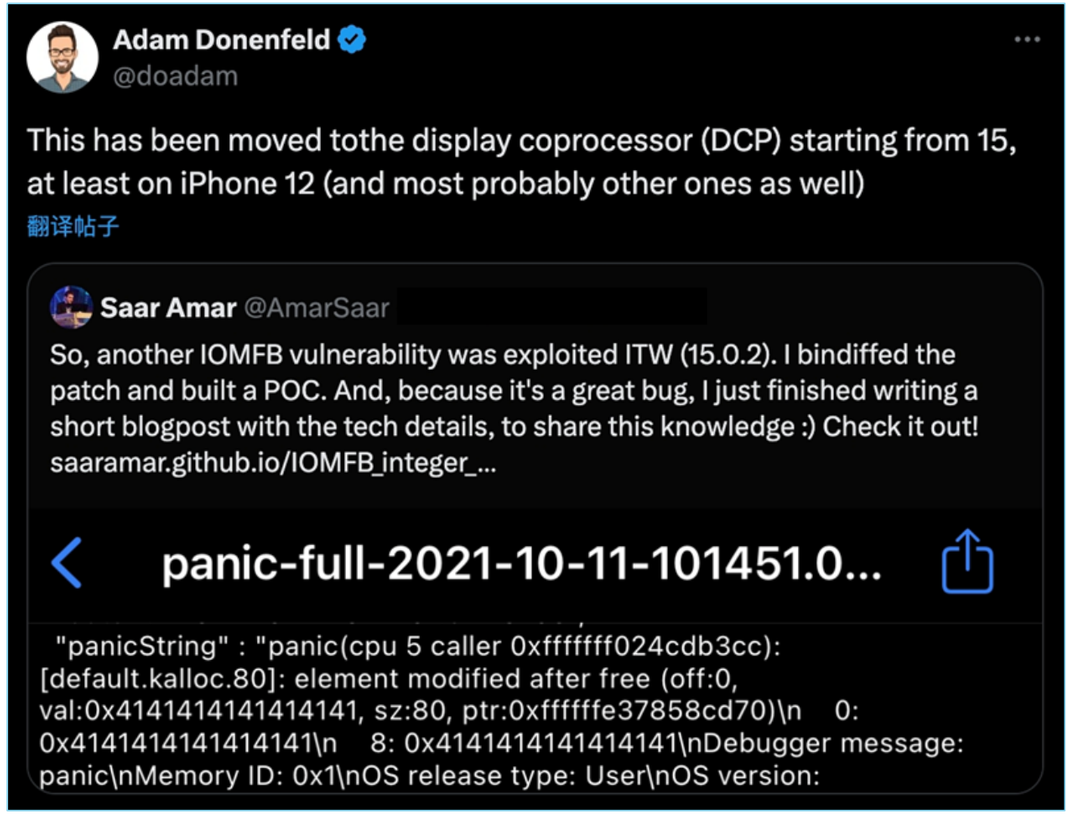
《IOMFB 的一些陈芝麻》
Pangu 9 Internals
https://www.blackhat.com/docs/us-16/materials/us-16-Wang-Pangu-9-Internals.pdf

Selector 0x53 - CVE-2021-30807
WebContent to EL1 LPE - OOBR in AppleCLCD and IOMobileFrameBuffer
https://saaramar.github.io/IOMobileFrameBuffer_LPE_POC/

Selector 0x4E - CVE-2021-30883
Bindiff and PoC for the IOMFB Vulnerability, iOS 15.0.2
https://saaramar.github.io/IOMFB_integer_overflow_poc/

After the party, most of the attack surfaces have been removed

IOMFB meme



A form of defense-in-depth

Is it still possible to find new IOMFB kernel vulnerabilities?

Case #5 - CVE-2024-44199
IOMFB::PBTBlockHandlerGeneric::get_map_buf_descs Kernel Out-of-bounds Access Vulnerability caused by Comparison between Unsigned and Signed Integers
About the security content of macOS Sonoma 14.6
https://support.apple.com/en-us/120911

Definitely

CVE-2024-44199 resides in the function IOMFB::PBTBlockHandlerGeneric::get_map_buf_descs, which consists of merely two lines of code.

It's worth mentioning that this vulnerability can be triggered directly in user mode without the need to apply for any entitlement.

```
1  __int64 __fastcall IOMFB::PBTBlockHandlerGeneric::get_map_buf_descs(int index)
2  {
3    if ( index >= 0x7A )
4      panic("\"IOMFB: attempt to use invalid PBT type\" @%s:%d", "IOMFBBlockManager.cpp", 24LL);
5    return IOMFB::PBTBlockHandlerGeneric::descs[2 * (unsigned int)index];
6  }
```

macOS Sonoma 14.0 Beta 0 (23A344)

The panic

```
(lldb) di
IOMobileGraphicsFamily`IOMFB::PBTBlockHandlerGeneric::get_map_buf_descs:
 0xfffffe002bb7f770 <+88>:  add    x8, x8, #0x308 ; IOMFB::PBTBlockHandlerGeneric::descs
->  0xfffffe002bb7f774 <+92>:  ldr    q0, [x8, x9, lsl #4]
    0xfffffe002bb7f778 <+96>:  stur   q0, [x29, #-0x10]
    0xfffffe002bb7f77c <+100>: ldur   x0, [x29, #-0x10]
    0xfffffe002bb7f780 <+104>: ldur   x1, [x29, #-0x8]

(lldb) register read
General Purpose Registers:
        x8 = 0xfffffe0029488308  IOMobileGraphicsFamily`IOMFB::PBTBlockHandlerGeneric::descs
        x9 = 0x00000000deadcafe
        fp = 0xfffffe49a50af5d0
        lr = 0xfffffe002bb3fb98  IOMobileGraphicsFamily-
DCP`IOMobileFramebufferAP::map_block_buf(IOMobileFramebufferAP::map_block_buf_args*,
IOMFB_Parameter_Block_Type, unsigned char const*, unsigned long, task*, bool) + 156
        sp = 0xfffffe49a50af5a0
        pc = 0xfffffe002bb7f774
IOMobileGraphicsFamily`IOMFB::PBTBlockHandlerGeneric::get_map_buf_descs(IOMFB_Parameter_Block_Type) + 92
        ......
```

Two different patches - keep it as signed

Another interesting footnote is that I found two different implementations for the CVE-2024-44199 patch.

```
1  __int64 __fastcall IOMFB::PBTBlockHandlerGeneric::get_map_buf_descs(int index, _OWORD *descs_ptr)
2  {
3    char return_value; // [xsp+1Fh] [xbp-1h]
4
5    if ( descs_ptr )
6    {
7      if ( index >= 0 && index < 0x7A )
8      {
9        *descs_ptr = IOMFB::PBTBlockHandlerGeneric::descs[index];
10       return_value = 1;
11     }
12     else
13     {
14       IOLog("IOMFB: attempt to use invalid PBT type %d\n", (unsigned int)index);
15       return_value = 0;
16     }
17   }
18   else
19   {
20     IOLog("null descs_ptr\n");
21     return_value = 0;
22   }
23   return return_value & 1;
24 }
```

macOS Sonoma 14.6 Beta (23G5066c)

Two different patches - make it unsigned uniformly

If these were developed by separate teams, I would recommend establishing unified standard across all teams.

```
1  __int64 __fastcall IOMFB::PBTBlockHandlerGeneric::get_map_buf_descs(unsigned int index, _OWORD *descs_ptr)
2  {
3    if ( !descs_ptr )
4    {
5      IOLog("null descs_ptr\n");
6      return 0LL;
7    }
8    if ( index >= 0x85 )
9    {
10     IOLog("IOMFB: attempt to use invalid PBT type %d\n");
11     return 0LL;
12   }
13   *descs_ptr = IOMFB::PBTBlockHandlerGeneric::descs[index];
14   return 1LL;
15 }
```

macOS Sequoia 15.0 Beta 0 (24A335)

Case #6 - OE098868205995

```
(lldb) bt
* thread #1, stop reason = signal SIGSTOP
    ......
    frame #2: 0xfffffe00291f527c kernel.release.t8122`panic(str=<unavailable>) at debug.c:1113:2 [opt]
    frame #3: 0xfffffe00291ffcbc kernel.release.t8122`panic_with_thread_kernel_state(msg="Kernel data abort.",
ss=0xfffffe84e0393400) at sleh.c:901:2 [opt]
    frame #4: 0xfffffe0028ab1ebc kernel.release.t8122`handle_kernel_abort(state=0xfffffe84e0393400,
esr=2516582406, fault_addr=0, fault_code=FSC_TRANSLATION_FAULT_L2, fault_type=1,
expected_fault_handler=<unavailable>) at sleh.c:3116:2 [opt]
    frame #5: 0xfffffe0028ab0864 kernel.release.t8122`sleh_synchronous [inlined]
handle_abort(state=0xfffffe84e0393400, esr=2516582406, fault_addr=0, inspect_abort=<unavailable>,
handler=<unavailable>, expected_fault_handler=0x0000000000000000) at sleh.c:1743:2 [opt]
    frame #6: 0xfffffe0028ab0850 kernel.release.t8122`sleh_synchronous(context=0xfffffe84e0393400,
esr=2516582406, far=0, did_initiate_panic_lockdown=<unavailable>) at sleh.c:1256:3 [opt]
    frame #7: 0xfffffe002890b888 kernel.release.t8122`fleh_synchronous + 44
    frame #8: 0xfffffe002b040fe0 IOMobileGraphicsFamily-DCP`IOMFB::DCPMemoryDescriptor::prepare(IOMFB::
MemoryDescriptor::Options) + 8
    ......
```

Security Assessment of Display Co-processor (DCP) Firmware

Asahi Linux and m1n1 project

Reverse Engineering DCP
https://asahilinux.org/2021/08/progress-report-august-2021/

Asahi Linux: DCP Command Interface Reversing
https://www.youtube.com/watch?v=LNKLvwfFFa8

Asahi Linux: DCP/AGX ASC Mailbox Reversing
https://www.youtube.com/watch?v=V5W23At6b4Y

CVE-2021-30983, ColdIntro and ColdInvite

The Curious Tale of a Fake Carrier.app
https://googleprojectzero.blogspot.com/2022/06/curious-case-carrier-app.html

Abusing iPhone Co-processors for Privilege Escalation
https://objectivebythesea.org/v5/talks/OBTS_v5_iBeer.pdf

The Mystery Behind ColdIntro (CVE-2022-32894) and ColdInvite (CVE-2023-27930) a Co-processor Escape Vulnerability Contents
https://resources.jamf.com/documents/technical-papers/Coldintro-Coldinvite-Mystery-v2.0.pdf

Short-term research objectives of DCP

1. Dive deep into the architecture of the DCP subsystem.
2. Hunt for potential attack surfaces in the DCP subsystem.
3. Develop a DCP fuzzer by integrating the insights from 1 and 2.

Case study of APT firmware fuzzing

```
(lldb) memory read 0xfffffe29a0220000 -c0x7000 --force
0xfffffe29a0220000: 50 41 4e 49 43 20 2d 20 61 70 74 20 66 69 72 6d  PANIC - apt firm
0xfffffe29a0220010: 77 61 72 65 3a 20 61 70 74 2e 63 3a 33 37 30 20  ware: apt.c:370
0xfffffe29a0220020: 61 70 74 5f 76 62 69 28 29 20 2d 2d 20 20 2d 20  apt_vbi() --  -
0xfffffe29a0220030: 69 6f 6d 66 62 5f 6d 61 69 6c 62 6f 78 28 39 32  iomfb_mailbox(92
0xfffffe29a0220040: 29 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ).............. 
0xfffffe29a0220050: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0xfffffe29a0220060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0xfffffe29a0220070: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0xfffffe29a0220080: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0xfffffe29a0220090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0xfffffe29a02200a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0xfffffe29a02200b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0xfffffe29a02200c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0xfffffe29a02200d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0xfffffe29a02200e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0xfffffe29a02200f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0xfffffe29a0220100: 61 70 74 20 66 69 72 6d 77 61 72 65 3a 20 61 70  apt firmware: ap
0xfffffe29a0220110: 74 2e 63 3a 33 37 30 20 61 70 74 5f 76 62 69 28  t.c:370 apt_vbi(
0xfffffe29a0220120: 29 20 2d 2d 20 0a 52 54 4b 69 74 3a 20 52 54 4b  ) -- .RTKit: RTK
0xfffffe29a0220130: 69 74 2d 32 37 35 38 2e 34 30 2e 31 39 2e 72 65  it-2758.40.19.re
```

Case study of PCC firmware fuzzing

```
(lldb) memory read 0xfffffe2fffd40000 -c0x7000 --force
0xfffffe2fffd40000: 50 41 4e 49 43 20 2d 20 70 63 63 20 66 69 72 6d  PANIC - pcc firm
0xfffffe2fffd40010: 77 61 72 65 3a 20 61 68 5f 74 68 72 2e 63 3a 32  ware: ah_thr.c:2
0xfffffe2fffd40020: 33 33 20 61 68 5f 62 65 67 69 6e 5f 75 70 64 61  33 ah_begin_upda
0xfffffe2fffd40030: 74 65 28 29 20 2d 2d 20 20 2d 20 69 6f 6d 66 62  te() --  - iomfb
0xfffffe2fffd40040: 5f 6d 61 69 6c 62 6f 78 28 34 32 29 00 00 00 00  _mailbox(42)....
0xfffffe2fffd40050: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0xfffffe2fffd40060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0xfffffe2fffd40070: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0xfffffe2fffd40080: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0xfffffe2fffd40090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0xfffffe2fffd400a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0xfffffe2fffd400b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0xfffffe2fffd400c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0xfffffe2fffd400d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0xfffffe2fffd400e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0xfffffe2fffd400f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0xfffffe2fffd40100: 70 63 63 20 66 69 72 6d 77 61 72 65 3a 20 61 68  pcc firmware: ah
0xfffffe2fffd40110: 5f 74 68 72 2e 63 3a 32 33 33 20 61 68 5f 62 65  _thr.c:233 ah_be
0xfffffe2fffd40120: 67 69 6e 5f 75 70 64 61 74 65 28 29 20 2d 2d 20  gin_update() -
0xfffffe2fffd40130: 0a 52 54 4b 69 74 3a 20 52 54 4b 69 74 2d 32 34  .RTKit: RTKit-24
0xfffffe2fffd40140: 31 33 2e 34 31 2e 31 2e 72 65 6c 65 61 73 65 20  13.41.1.release
```

The RTBuddy mechanism

```
(lldb) bt
* thread #1, stop reason = signal SIGSTOP
  * frame #0: 0xfffffe001fbb63b0 kernel.release.t6000`DebuggerTrapWithState(db_op=DBOP_PANIC,
db_message="panic", db_panic_str="%s %s%s%s\n%s", db_panic_args=0xfffffe840d8c7df8, db_panic_options=0,
db_panic_data_ptr=0x0000000000000000, db_proceed_on_sync_failure=1, db_panic_caller=18446741875263258400) at
debug.c:714:2 [opt]
    frame #1: 0xfffffe001fbb593c kernel.release.t6000`panic_trap_to_debugger(panic_format_str="%s %s%s%s\n%s",
panic_args=0xfffffe840d8c7df8, reason=0, ctx=0x0000000000000000, panic_options_mask=0,
panic_data_ptr=0x0000000000000000, panic_caller=18446741875263258400) at debug.c:1175:2 [opt]
    frame #2: 0xfffffe0020333fd4 kernel.release.t6000`panic_with_options(reason=<unavailable>,
ctx=<unavailable>, debugger_options_mask=<unavailable>, str=<unavailable>) at debug.c:1018:2 [opt]
    frame #3: 0xfffffe002263bf20 RTBuddy`RTBuddyCrashlogEndpoint::_handleCrashlog(bool) + 1280
    frame #4: 0xfffffe002209e5e0 IOSlaveProcessor`IOSlaveEndpoint::checkForWork() + 124
    frame #5: 0xfffffe0020242e54 kernel.release.t6000`IOWorkLoop::runEventSources(this=0xfffffe33ceba17b0) at
IOWorkLoop.cpp:403:18 [opt]
    frame #6: 0xfffffe00202439dc kernel.release.t6000`IOWorkLoop::threadMain(this=0xfffffe33ceba17b0) at
IOWorkLoop.cpp:434:8 [opt]
    frame #7: 0xfffffe001fb70e98 kernel.release.t6000`Call_continuation + 216
```

```
[(lldb) bt
* thread #1, stop reason = signal SIGSTOP
  * frame #0: 0xfffffe001fbb63b0 kernel.release.t6000`DebuggerTrapWithState(db_op=DBOP_PANIC, db_message="panic", db_panic_str="%s %s%s%s\n%s", db_panic_args=0xfffffe840d8c7df8, db_panic_options=0, db_panic_data_ptr=0x0000000000000000, db_proceed_on_sync_failure=1, db_panic_caller=18446741875263258400) at debug.c:714:2 [opt]
    frame #1: 0xfffffe001fbb593c kernel.release.t6000`panic_trap_to_debugger(panic_format_str="%s %s%s%s\n%s", panic_args=0xfffffe840d8c7df8, reason=0, ctx=0x0000000000000000, panic_options_mask=0, panic_data_ptr=0x0000000000000000, panic_caller=18446741875263258400) at debug.c:1175:2 [opt]
    frame #2: 0xfffffe0020333fd4 kernel.release.t6000`panic_with_options(reason=<unavailable>, ctx=<unavailable>, debugger_options_mask=<unavailable>, str=<unavailable>) at debug.c:1018:2 [opt]
    frame #3: 0xfffffe002263bf20 RTBuddy`RTBuddyCrashlogEndpoint::_handleCrashlog(bool) + 1280
    frame #4: 0xfffffe002209e5e0 IOSlaveProcessor`IOSlaveEndpoint::checkForWork() + 124
    frame #5: 0xfffffe0020242e54 kernel.release.t6000`IOWorkLoop::runEventSources(this=0xfffffe33ceba17b0) at IOWorkLoop.cpp:403:18 [opt]
    frame #6: 0xfffffe00202439dc kernel.release.t6000`IOWorkLoop::threadMain(this=0xfffffe33ceba17b0) at IOWorkLoop.cpp:434:8 [opt]
    frame #7: 0xfffffe001fb70e98 kernel.release.t6000`Call_continuation + 216
[(lldb) memory read 0xfffffe33cedf8000 -c0x7a50                                                                  ]
0xfffffe33cedf8000: 50 41 4e 49 43 20 2d 20 69 6e 76 61 6c 69 64 20  PANIC - invalid
0xfffffe33cedf8010: 70 69 78 65 6c 20 65 6e 63 6f 64 69 6e 67 20 2d  pixel encoding -
0xfffffe33cedf8020: 20 64 63 70 61 76 2d 76 69 64 65 6f 2d 69 6e 74   dcpav-video-int
0xfffffe33cedf8030: 65 72 66 61 63 65 2d 65 70 69 63 28 37 30 29 00  erface-epic(70).
0xfffffe33cedf8040: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0xfffffe33cedf8050: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0xfffffe33cedf8060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0xfffffe33cedf8070: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0xfffffe33cedf8080: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0xfffffe33cedf8090: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0xfffffe33cedf80a0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0xfffffe33cedf80b0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0xfffffe33cedf80c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0xfffffe33cedf80d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0xfffffe33cedf80e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0xfffffe33cedf80f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0xfffffe33cedf8100: 69 6e 76 61 6c 69 64 20 70 69 78 65 6c 20 65 6e  invalid pixel en
0xfffffe33cedf8110: 63 6f 64 69 6e 67 0a 52 54 4b 69 74 3a 20 52 54  coding.RTKit: RT
0xfffffe33cedf8120: 4b 69 74 2d 32 30 36 32 2e 31 34 31 2e 31 2e 64  Kit-2062.141.1.d
0xfffffe33cedf8130: 65 62 75 67 20 2d 20 43 6c 69 65 6e 74 3a 20 6c  ebug - Client: l
0xfffffe33cedf8140: 6f 63 61 6c 2d 74 36 30 30 78 64 63 70 2e 72 65  ocal-t600xdcp.re
0xfffffe33cedf8150: 6c 65 61 73 65 0a 21 55 55 49 44 3a 20 30 32 63  lease.!UUID: 02c
0xfffffe33cedf8160: 62 33 63 37 36 2d 66 65 39 66 2d 33 33 31 35 2d  b3c76-fe9f-3315-
0xfffffe33cedf8170: 39 61 65 62 2d 36 63 36 38 62 63 32 32 65 33 35  9aeb-6c68bc22e35
0xfffffe33cedf8180: 32 0a 54 69 6d 65 3a 20 30 78 30 30 30 30 30 30  2.Time: 0x000000
0xfffffe33cedf8190: 30 34 66 39 63 61 65 66 35 30 0a 0a 46 61 75 6c  04f9caef50..Faul
0xfffffe33cedf81a0: 74 69 6e 67 20 74 61 73 6b 20 20 37 30 20 43 61  ting task  70 Ca
0xfffffe33cedf81b0: 6c 6c 20 53 74 61 63 6b 3a 20 30 78 30 30 30 30  ll Stack: 0x0000
0xfffffe33cedf81c0: 30 30 30 30 30 30 30 31 65 33 61 38 20 30 78 30  00000001e3a8 0x0
0xfffffe33cedf81d0: 30 30 30 30 30 30 30 30 30 30 31 64 64 61 63 20  00000000001ddac
0xfffffe33cedf81e0: 30 78 30 30 30 30 30 30 30 30 30 30 30 31 64 62  0x000000000001db
0xfffffe33cedf81f0: 61 38 20 30 78 30 30 30 30 30 30 30 30 30 30 30  a8 0x00000000000
0xfffffe33cedf8200: 32 30 65 64 30 20 30 78 30 30 30 30 30 30 30 30  20ed0 0x00000000
0xfffffe33cedf8210: 30 30 30 33 31 65 33 34 20 30 78 30 30 30 30 30  00031e34 0x00000
0xfffffe33cedf8220: 30 30 30 30 30 30 31 39 31 65 37 38 20 30 78 30  00000191e78 0x00
0xfffffe33cedf8230: 30 30 30 30 30 30 30 30 31 39 65 64 33 34 20 30  0000000019ed34 0
0xfffffe33cedf8240: 78 30 30 30 30 30 30 30 30 30 30 30 34 64 61 39  x000000000004da9
0xfffffe33cedf8250: 63 20 30 78 30 30 30 30 30 30 30 30 30 30 30 34  c 0x000000000004
0xfffffe33cedf8260: 62 35 65 30 20 30 78 30 30 30 30 30 30 30 30 30  b5e0 0x000000000
0xfffffe33cedf8270: 30 30 35 34 33 63 30 20 30 78 30 30 30 30 30 30  00543c0 0x000000
0xfffffe33cedf8280: 30 30 30 30 30 33 31 63 39 63 20 30 78 30 30 30  0000031c9c 0x000
0xfffffe33cedf8290: 30 30 30 30 30 30 30 30 31 65 62 35 30 20 30 78  000000001eb50 0x
0xfffffe33cedf82a0: 30 30 30 30 30 30 30 30 30 30 30 33 31 61 39 34  0000000000031a94
0xfffffe33cedf82b0: 20 30 78 30 30 30 30 30 30 30 30 30 30 30 34 35   0x0000000000045
0xfffffe33cedf82c0: 64 31 34 20 30 78 30 30 30 30 30 30 30 30 30 30  d14 0x0000000000
0xfffffe33cedf82d0: 31 36 36 36 63 30 20 30 78 30 30 30 30 30 30 30  1666c0 0x0000000
0xfffffe33cedf82e0: 30 30 30 31 62 34 62 30 34 20 30 78 30 30 30 30  0001b4b04 0x0000
0xfffffe33cedf82f0: 30 30 30 30 30 30 31 61 64 36 35 38 20 30 78 30  0000001ad658 0x0
```

Case #7 - Video streaming attack surface

dcpav-video-interface-epic daemon

Case #8 - CVE-2025-24111
dcpav-video-interface-epic LinkWithSource Display Co-processor (DCP) Firmware Vulnerability
About the security content of iOS 18.3 and iPadOS 18.3
https://support.apple.com/en-us/122066
About the security content of macOS Sequoia 15.3
https://support.apple.com/en-us/122068

```
(lldb) bt
* thread #1, stop reason = signal SIGSTOP
  * frame #0: 0xfffffe0018617fe8 kernel.release.t8122`DebuggerTrapWithState(db_op=DBOP_PANIC, db_message="panic", db_panic_str="%s %s%s%s\n%s", db_panic_args=0xfffffe8f15497db8, db_panic_options=16384, db_panic_data_ptr=0x0000000000000000, db_proceed_on_sync_failure=1, db_panic_caller=18446741875144245264, db_panic_initiator="DCP") at debug.c:823:2 [opt]
    frame #1: 0xfffffe00186175c8 kernel.release.t8122`panic_trap_to_debugger(panic_format_str="%s %s%s%s\n%s", panic_args=0xfffffe8f15497db8, reason=0, ctx=0x0000000000000000, panic_options_mask=16384, panic_data_ptr=0x0000000000000000, panic_caller=18446741875144245264, panic_initiator="DCP") at debug.c:1334:2 [opt]
    frame #2: 0xfffffe0018e6d2fc kernel.release.t8122`panic_with_options_and_initiator(initiator=<unavailable>, reason=<unavailable>, ctx=<unavailable>, debugger_options_mask=<unavailable>, str=<unavailable>) at debug.c:1150:2 [opt]
    frame #3: 0xfffffe001b4bc010 RTBuddy`RTBuddyCrashlogEndpoint::_handleCrashlog(bool) + 1368
    frame #4: 0xfffffe001aeda1a8 IOSlaveProcessor`IOSlaveEndpoint::checkForWork() + 124
    frame #5: 0xfffffe0018d68b48 kernel.release.t8122`IOWorkLoop::runEventSources(this=0xfffffe1b3fdce390) at IOWorkLoop.cpp:403:18 [opt]
    frame #6: 0xfffffe0018d69724 kernel.release.t8122`IOWorkLoop::threadMain(this=0xfffffe1b3fdce390) at IOWorkLoop.cpp:434:8 [opt]
    frame #7: 0xfffffe00185cd8a8 kernel.release.t8122`Call_continuation + 200
(lldb) memory read 0xfffffe29a6728000 -c0x2000 --force
0xfffffe29a6728000: 44 41 54 41 20 41 42 4f 52 54 20 20 70 63 3d 30  DATA ABORT  pc=0
0xfffffe29a6728010: 78 30 30 30 30 30 30 30 30 30 30 33 62 61 38 35  x00000000003ba85
0xfffffe29a6728020: 34 20 45 78 63 65 70 74 69 6f 6e 20 63 6c 61 73  4 Exception clas
0xfffffe29a6728030: 73 3d 30 78 32 35 20 28 44 61 74 61 20 41 62 6f  s=0x25 (Data Abo
0xfffffe29a6728040: 72 74 20 74 61 6b 65 6e 20 77 69 74 68 6f 75 74  rt taken without
0xfffffe29a6728050: 20 61 20 63 68 61 6e 67 65 20 69 6e 20 45 78 63   a change in Exc
0xfffffe29a6728060: 65 70 74 69 6f 6e 20 6c 65 76 65 6c 29 2c 20 49  eption level), I
0xfffffe29a6728070: 4c 3d 31 2c 20 69 73 73 3d 30 78 37 20 66 61 72  L=1, iss=0x7 far
0xfffffe29a6728080: 3d 30 78 66 66 66 66 66 66 66 66 34 30 65 38 32  =0xffffffff40e82
0xfffffe29a6728090: 63 39 38 20 2d 20 64 63 70 61 76 2d 76 69 64 65  c98 - dcpav-vide
0xfffffe29a67280a0: 6f 2d 69 6e 74 65 72 66 61 63 65 2d 65 70 69 63  o-interface-epic
0xfffffe29a67280b0: 28 38 39 29 00 00 00 00 00 00 00 00 00 00 00 00  (89)............
0xfffffe29a67280c0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0xfffffe29a67280d0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0xfffffe29a67280e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0xfffffe29a67280f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0xfffffe29a6728100: 52 54 4b 69 74 3a 20 52 54 4b 69 74 2d 32 37 35  RTKit: RTKit-275
0xfffffe29a6728110: 38 2e 34 30 2e 31 39 2e 72 65 6c 65 61 73 65 20  8.40.19.release
0xfffffe29a6728120: 2d 20 43 6c 69 65 6e 74 3a 20 41 70 70 6c 65 44  - Client: AppleD
0xfffffe29a6728130: 43 50 2d 38 31 31 2e 34 30 2e 38 7e 31 35 32 35  CP-811.40.8~1525
0xfffffe29a6728140: 2d 74 38 31 32 32 64 63 70 2e 52 45 4c 45 41 53  -t8122dcp.RELEAS
0xfffffe29a6728150: 45 0a 21 55 55 49 44 3a 20 61 31 30 30 30 30 31  E.!UUID: a100001
0xfffffe29a6728160: 30 2d 32 31 34 30 2d 31 65 64 35 2d 61 31 37 38  0-2140-1ed5-a178
0xfffffe29a6728170: 2d 38 30 64 32 30 31 34 30 31 65 64 35 0a 41 53  -80d201401ed5.AS
0xfffffe29a6728180: 4c 52 20 73 6c 69 64 65 3a 20 30 78 30 30 30 30  LR slide: 0x0000
0xfffffe29a6728190: 30 30 30 30 30 30 32 63 64 30 30 30 0a 54 69 6d  0000002cd000.Tim
0xfffffe29a67281a0: 65 3a 20 30 78 30 30 30 30 30 30 30 33 63 30 31  e: 0x00000003c01
0xfffffe29a67281b0: 33 39 33 36 36 0a 0a 46 61 75 6c 74 69 6e 67 20  39366..Faulting
0xfffffe29a67281c0: 74 61 73 6b 20 73 74 61 63 6b 20 66 72 61 6d 65  task stack frame
0xfffffe29a67281d0: 3a 0a 57 72 6f 6e 67 20 66 72 61 6d 65 20 73 69  :.Wrong frame si
0xfffffe29a67281e0: 7a 65 20 66 6f 72 20 41 52 4d 76 38 2e 20 47 6f  ze for ARMv8. Go
0xfffffe29a67281f0: 74 20 38 34 30 20 62 79 74 65 73 2c 20 65 78 70  t 840 bytes, exp
0xfffffe29a6728200: 65 63 74 65 64 20 38 32 38 0a 20 20 70 63 3d 30  ected 828.  pc=0
0xfffffe29a6728210: 78 30 30 30 30 30 30 30 30 30 30 33 62 61 38 35  x00000000003ba85
0xfffffe29a6728220: 34 20 45 78 63 65 70 74 69 6f 6e 20 63 6c 61 73  4 Exception clas
0xfffffe29a6728230: 73 3d 30 78 32 35 20 28 44 61 74 61 20 41 62 6f  s=0x25 (Data Abo
0xfffffe29a6728240: 72 74 20 74 61 6b 65 6e 20 77 69 74 68 6f 75 74  rt taken without
0xfffffe29a6728250: 20 61 20 63 68 61 6e 67 65 20 69 6e 20 45 78 63   a change in Exc
0xfffffe29a6728260: 65 70 74 69 6f 6e 20 6c 65 76 65 6c 29 2c 20 49  eption level), I
0xfffffe29a6728270: 4c 3d 31 2c 20 69 73 73 3d 30 78 37 20 66 61 72  L=1, iss=0x7 far
0xfffffe29a6728280: 3d 30 78 66 66 66 66 66 66 66 66 34 30 65 38 32  =0xffffffff40e82
0xfffffe29a6728290: 63 39 38 0a 20 20 72 30 30 3d 30 78 66 66 66 66  c98.  r00=0xffff
0xfffffe29a67282a0: 66 66 66 66 34 30 65 30 31 65 61 38 20 20 72 30  ffff40e01ea8  r0
0xfffffe29a67282b0: 31 3d 30 78 66 66 66 66 66 66 66 66 34 30 65 38  1=0xffffffff40e8
0xfffffe29a67282c0: 32 63 32 30 20 20 72 30 32 3d 30 78 30 30 30 30  2c20  r02=0x0000
0xfffffe29a67282d0: 30 30 30 30 30 30 30 30 30 30 61 66 20 20 72 30  0000000000af  r0
0xfffffe29a67282e0: 33 3d 30 30 30 30 30 30 30 30 30 30 30 30 30 30  3=00000000000000
0xfffffe29a67282f0: 30 30 30 30 0a 20 20 72 30 34 3d 30 78 30 30 30  0000.  r04=0x000
```

Case #8 - CVE-2025-24111

From AP user mode to DCP firmware

It's worth mentioning that this vulnerability can be directly triggered from user mode without requiring any entitlement. Furthermore, many key registers within the DCP firmware are controllable.

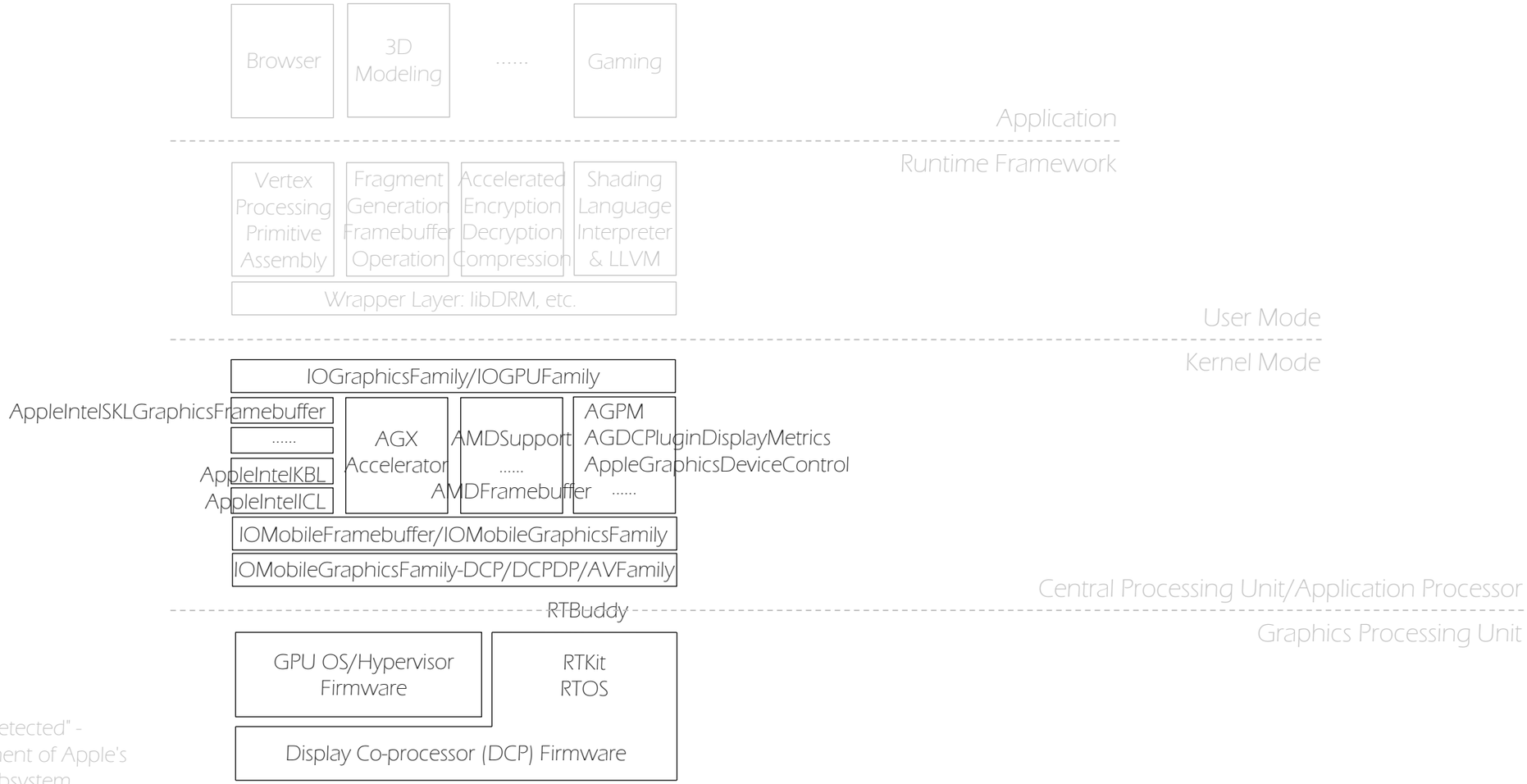
```
panic(cpu 0 caller 0xfffffe0017e34010): DCP DATA ABORT  pc=0x000000000032d854 Exception class=0x25 (Data Abort taken without a change in Exception level),
RTKit: RTKit-2758.40.19.release - Client: AppleDCP-811.40.8~1525-t8122dcp.RELEASE
!UUID: a1000010-2140-1ed5-a178-80d201401ed5
ASLR slide: 0x0000000000240000
Time: 0x000000005f763530

Faulting task stack frame:
Wrong frame size for ARMv8. Got 840 bytes, expected 828
  pc=0x000000000032d854 Exception class=0x25 (Data Abort taken without a change in Exception level), IL=1, iss=0x7 far=0xffffffff40e82748
  r00=0xffffffff40ddfbb8  r01=0xffffffff40e826d0  r02=0x00000000000000aa  r03=0000000000000000
  r04=0000000000000000  r05=0000000000000000  r06=0000000000000000  r07=0000000000000000
  r08=0x00000000deadbeef  r09=0x00000000008fa350  r10=0x0000000000000018  r11=0x0000000000000120
  r12=0000000000000000  r13=0000000000000000  r14=0000000000000000  r15=0000000000000000
  r16=0x00000000008fa350  r17=0x950b0000008fa350  r18=0000000000000000  r19=0xffffffff40e826d0
  r20=0xffffffff40ddfbb8  r21=0xffffffff40ddfbb8  r22=0xffffffff40e826d0  r23=0xffffffff40e70fb8
  r24=0xffffffff40ffc128  r25=0xffffffff40de1aa8  r26=0x0000000000000001  r27=0xffffffff40ddad68
  r28=0000000000000000  r29=0xffffffff0c691c60
   sp=0xffffffff0c691c30   lr=0x5a73c9d5f8309dfc   pc=0x000000000032d854  psr=0x60000004
  psr=0x60000004        cpacr=0x300000          fpsr=00000000          fpcr=00000000
```

Case #8 - CVE-2025-24111

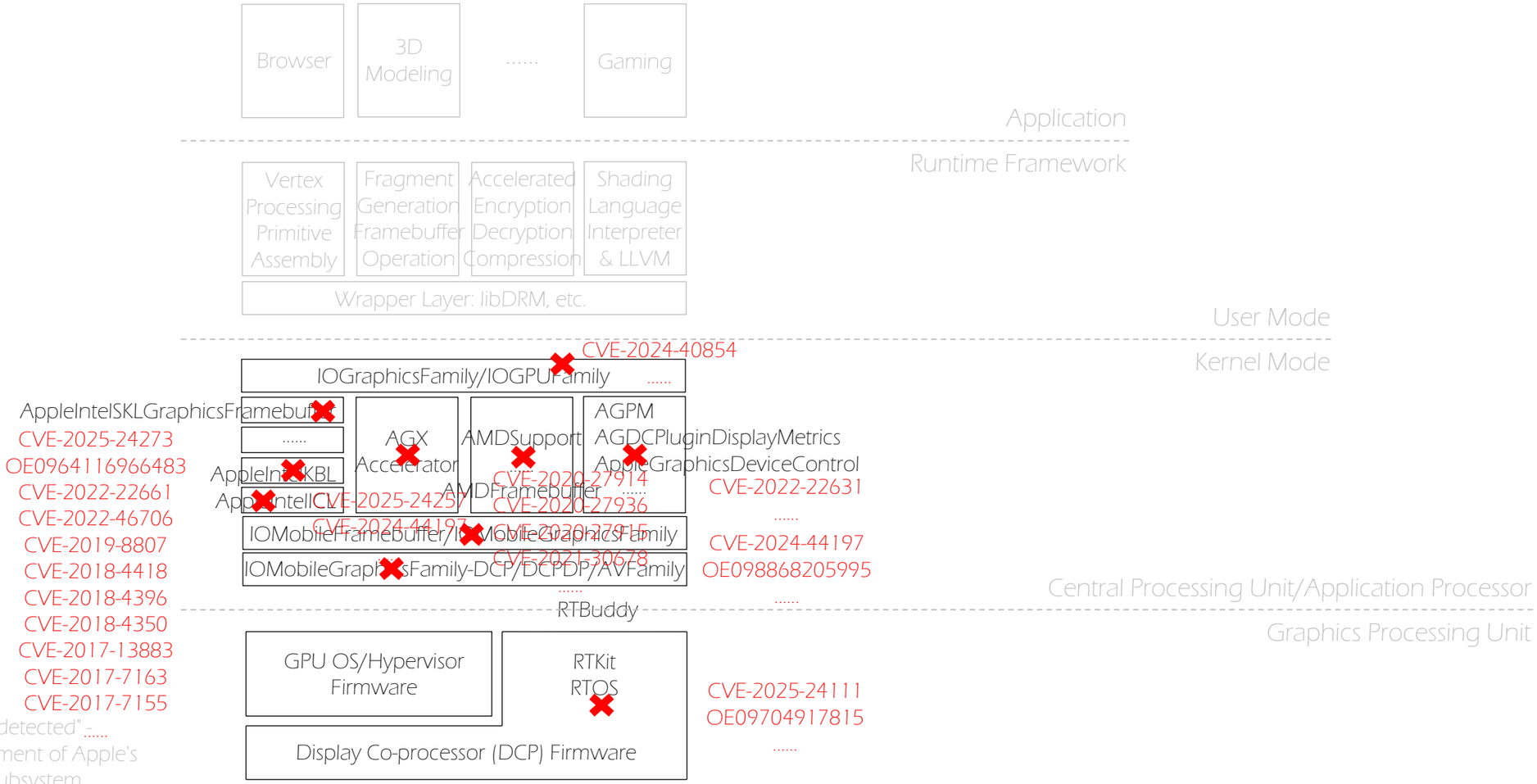
Conclusions and Takeaways

Recalling the previously mentioned kernel architecture



"Dead pixel detected" -
A Security Assessment of Apple's Graphics Subsystem

There remains room for improvement



From the perspective of security engineering

1. New features always mean new attack surfaces. Reckless refactoring can sometimes bring catastrophic consequences to a system.

2. Even in the era of AI and vibe coding, we still need to attach great importance to the programming fundamentals of C/C++, the training related to code quality, the routine use of static and dynamic dev tools, and the regular review of warning messages.

3. The emergency and official patches for CVE-2024-44199 reflect that the engineering team's handling of classic C/C++ issues, such as "comparison between signed and unsigned integer expressions" is somewhat casual. If it is caused by the lack of guidance, the first step should be to formulate these specifications and conduct training.

From the perspective of vulnerability hunting

1. Certain complex kernel functions are repeatedly found to contain vulnerabilities by the security community. I have identified numerous such cases within Apple's Bluetooth, Wi-Fi, and graphics subsystems.

2. In terms of the number of kernel vulnerabilities, the graphics subsystem of the Apple Silicon platform deviates from the theoretical error rate per ten thousand lines of binary code. I believe that security issues in components like the Apple Graphics Accelerator and Display Co-processor are concealed by the complex architecture. Obviously, hiding behind does not mean security.

From the perspective of vulnerability hunting (cont.)

3. Significant knowledge gaps exist across multiple domains including AGX GPU 13/14/15, G/X/C/S-series, DCP firmware, RTKit subsystem, and RTBuddy V1/V2 architectures. The inherent asymmetry of knowledge coupled with capability disparities will further widen the gap in vulnerability research within the security community.

From the perspective of offensive and defensive

1. The competition between the offensive and defensive sides in fields such as IOMobileFrameBuffer once reached a fever pitch.

2. An analysis of IOMobileFrameBuffer/IOMobileGraphicsFamily/AppleCLCD from 2020 to 2022 indicates the lag in vulnerability research is approximately three to six months.

3. The practice of transferring functionalities of kernel extensions such as IOMobileFramebuffer to the Display Co-processor firmware is effective to a certain extent. This mitigation measure restricts access to select high-risk interfaces exposed within WebContent, thereby implicitly introducing the concept of defense-in-depth. But once again, hiding behind does not mean security.

Q&A

"Dead pixel detected" -
A Security Assessment of Apple's
Graphics Subsystem