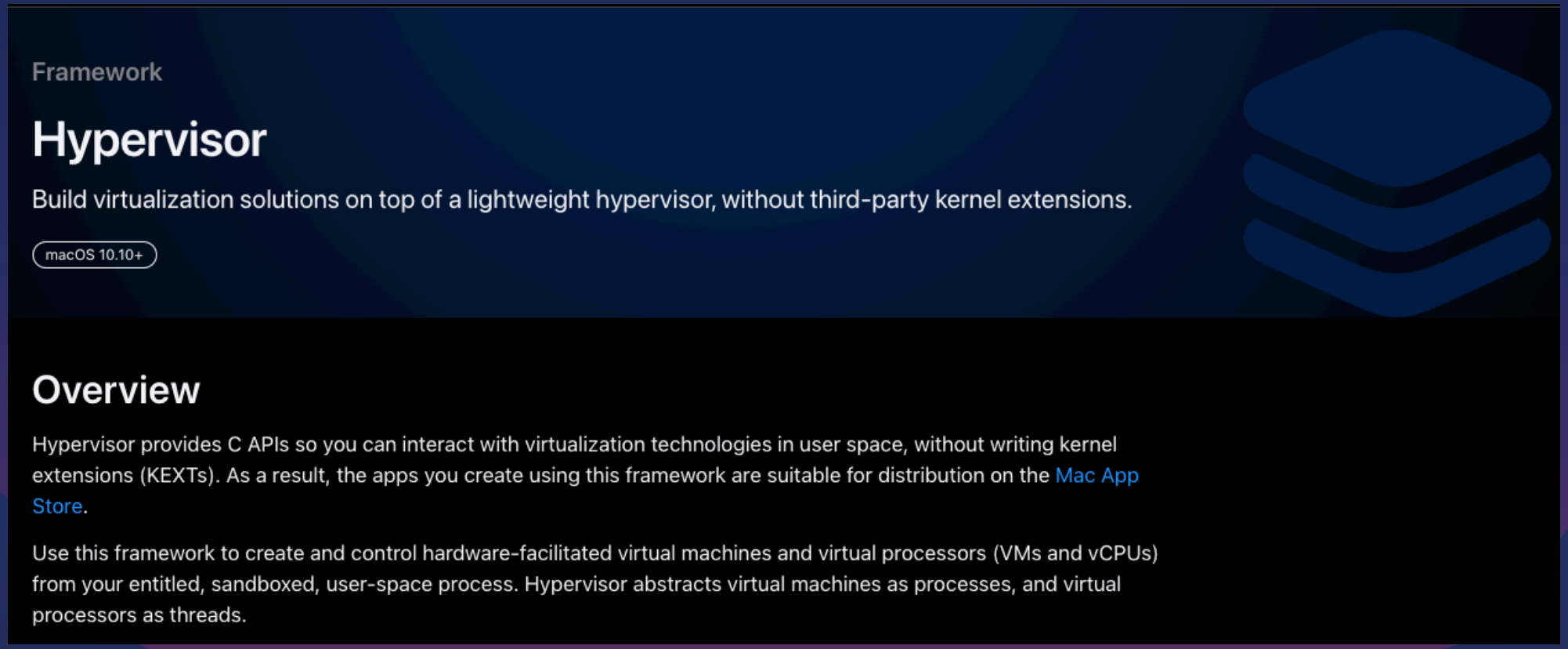
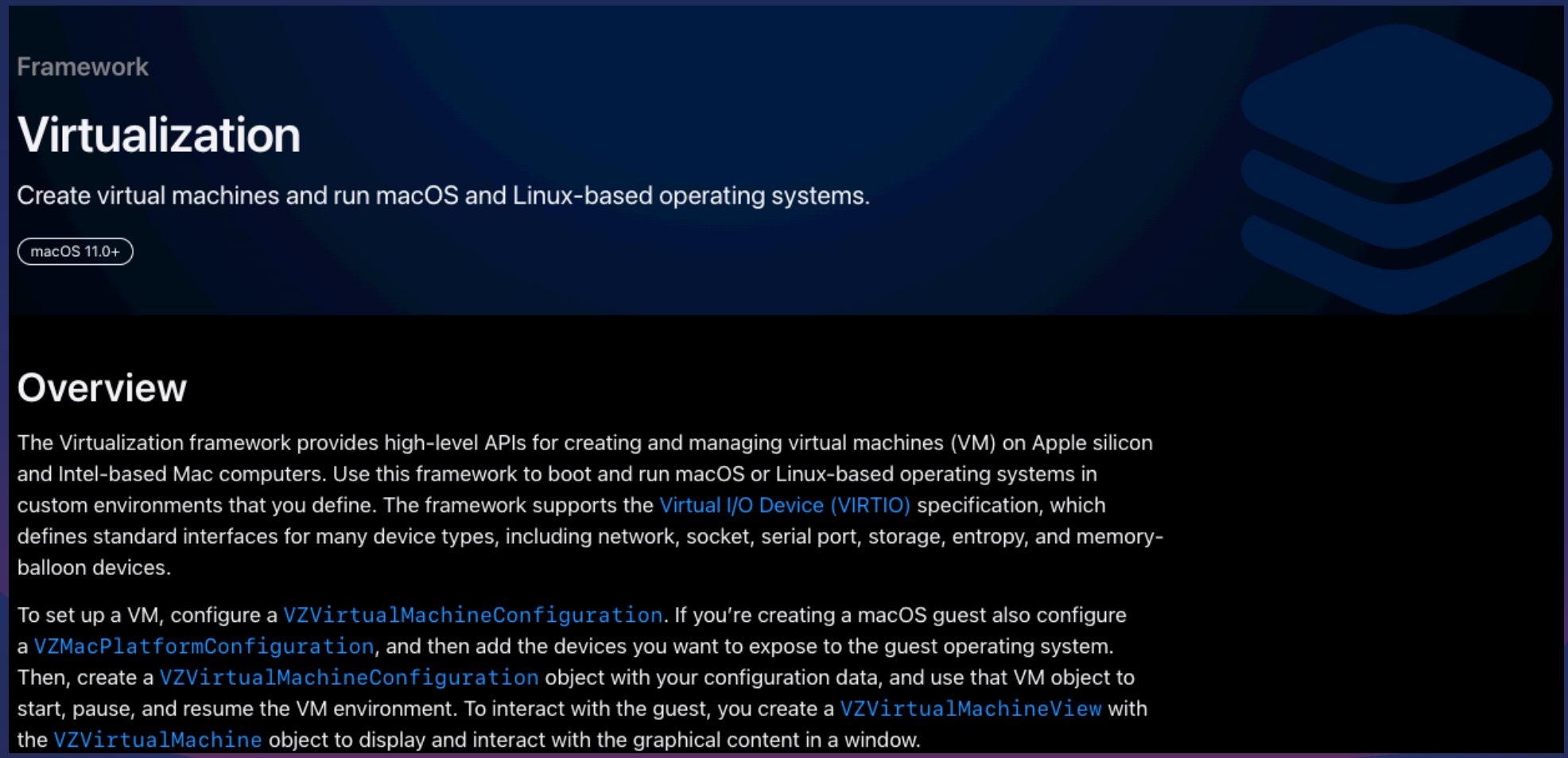
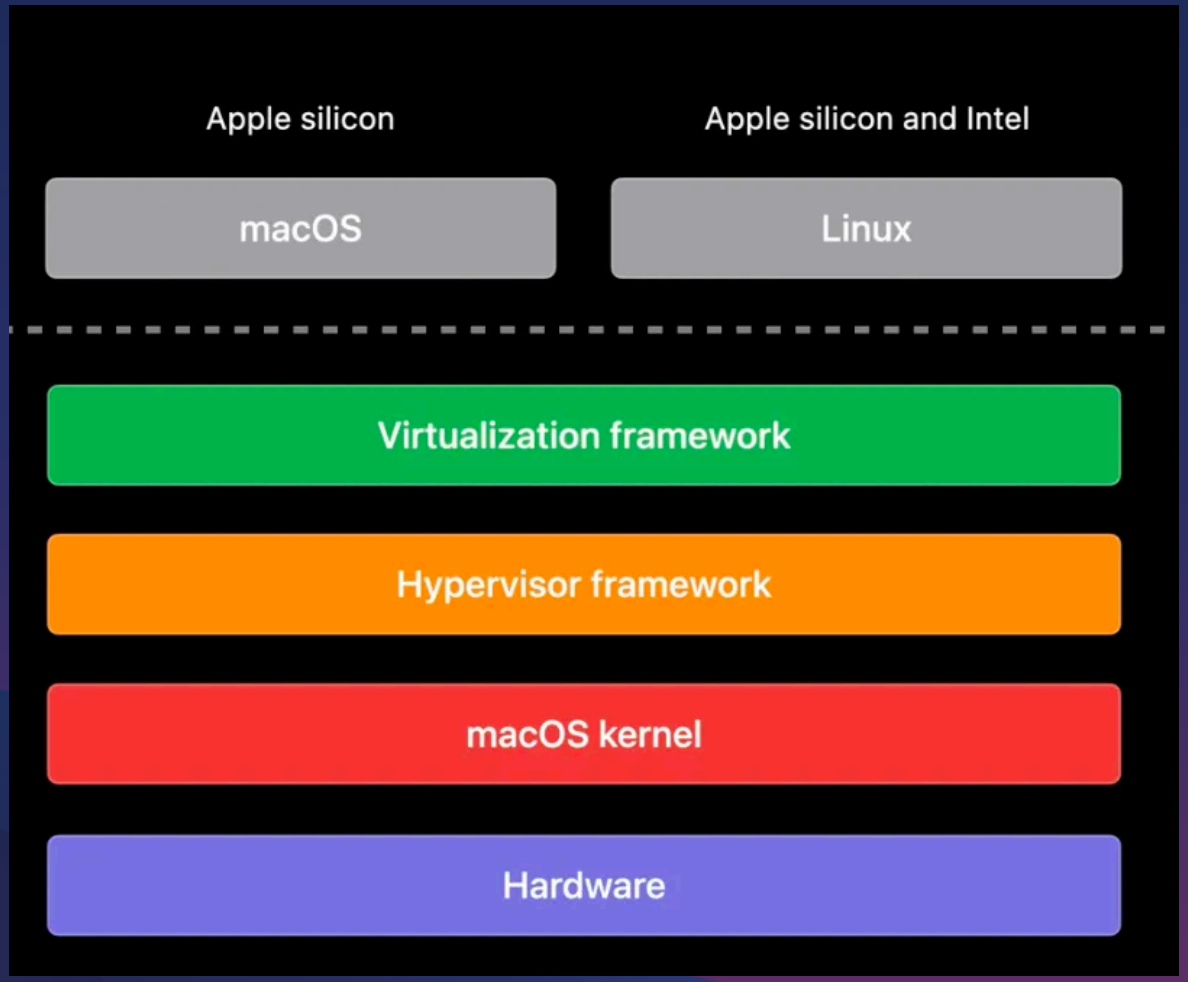
Apple Hypervisor Framework



Apple Hypervisor Framework



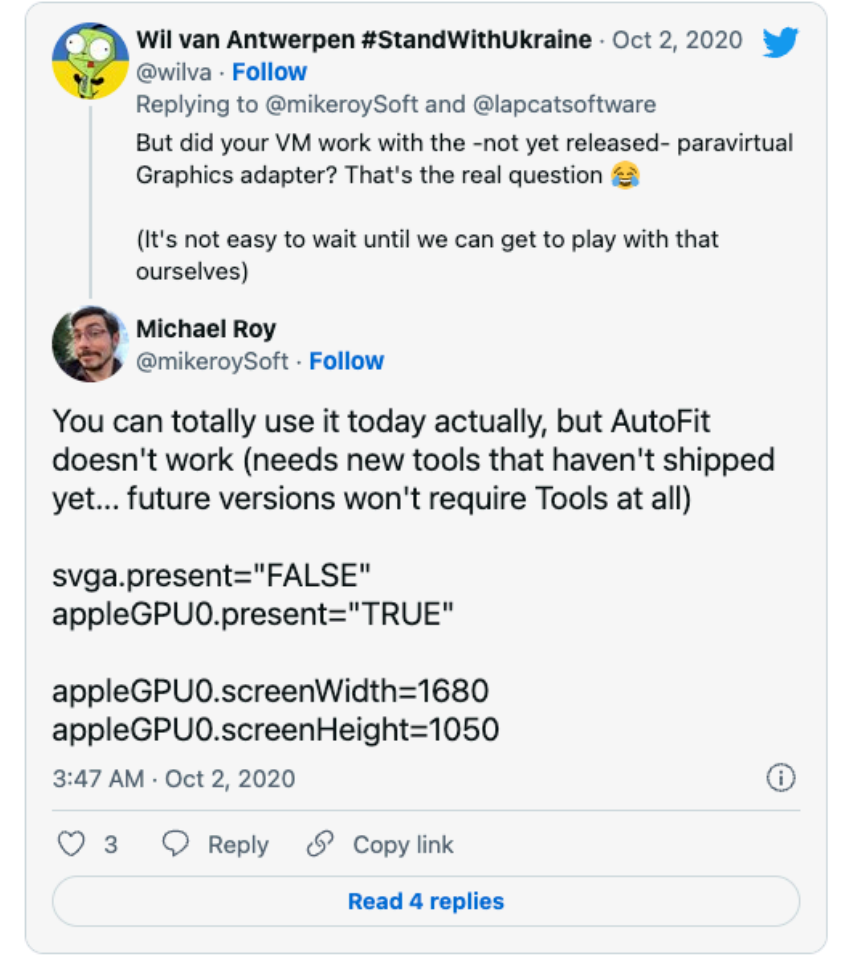
Apple Hypervisor Framework



https://www.infoq.com/news/2022/06/apple-virtualization-framework/

- 苹果大力发展自己的虚拟机技术栈
  - 新特性
  - 新代码
  - 新的攻击面

Apple Hypervisor Framework



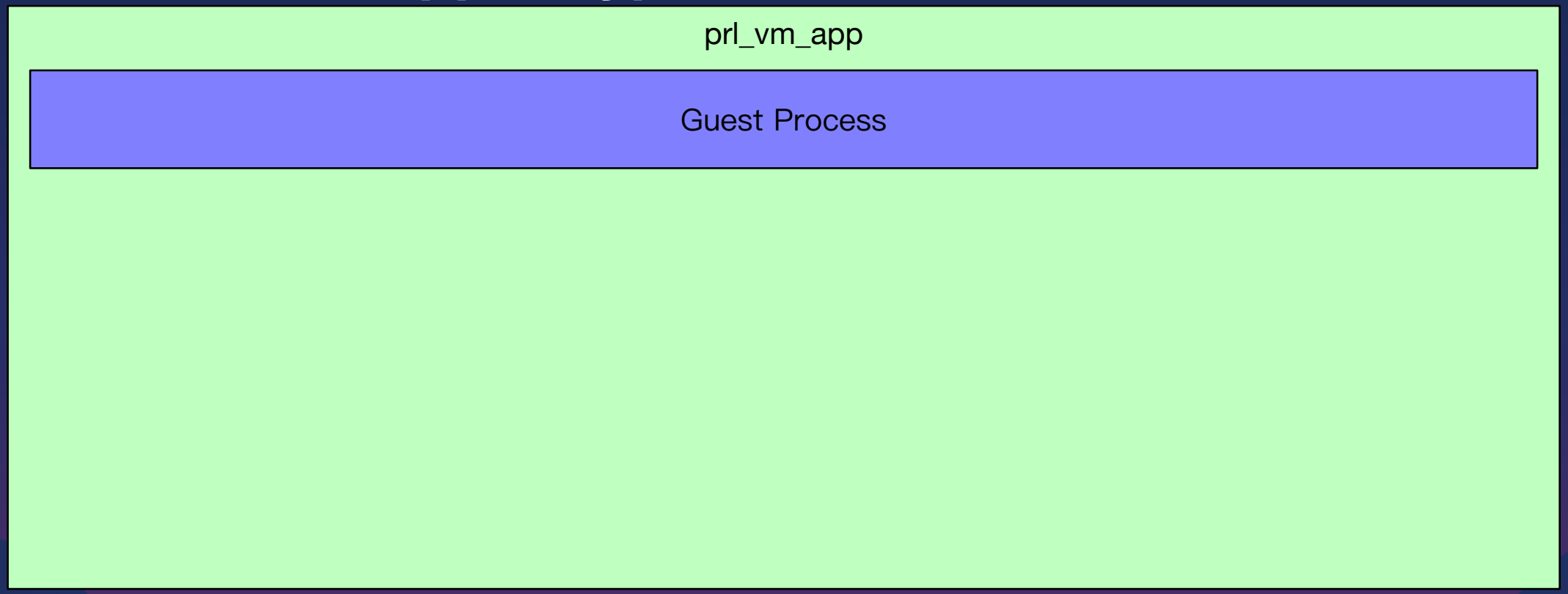
- 一套苹果自主开发的虚拟机Hypervisor框架，并开放给其他厂商使用，提高虚拟机的性能。
  - VMware计划支持该特性
  - Parallels Desktop已经在正式发布的版本中支持！

审计目标 3 :Apple Hypervisor

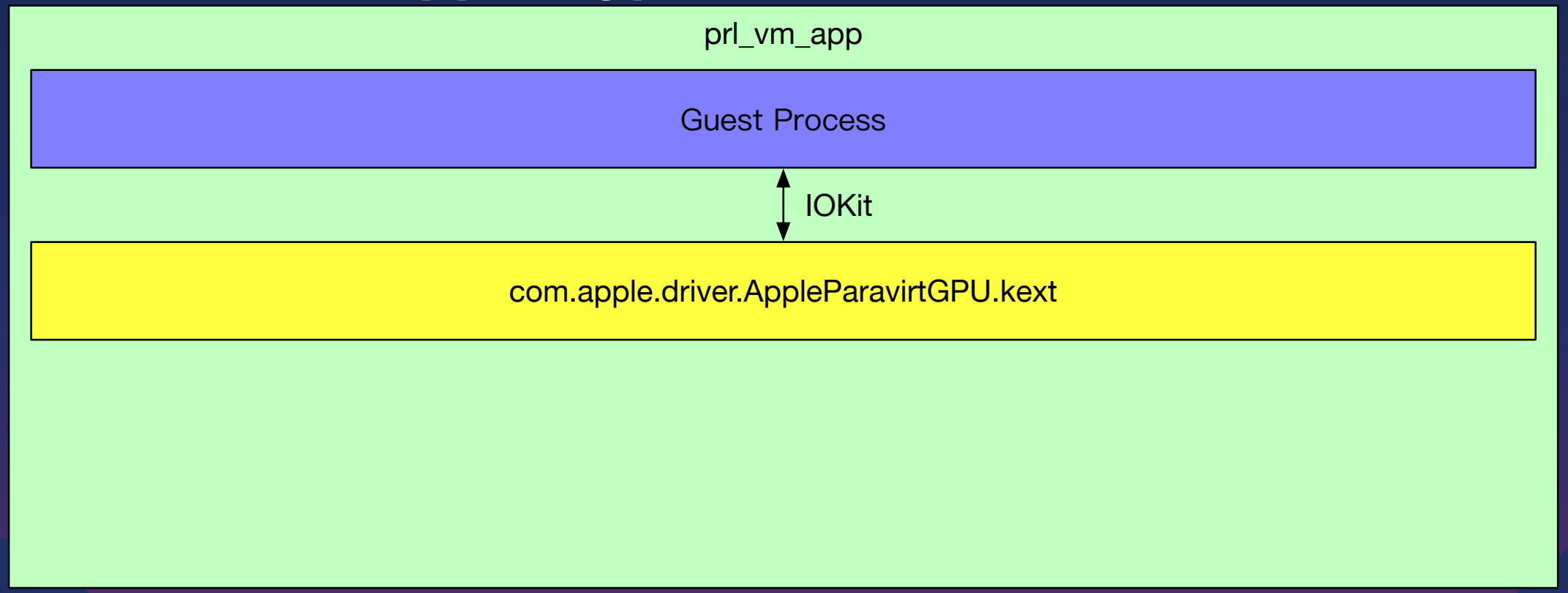
Apple Hypervisor Framework

- 在物理机的环境中应用层程序通过IOKit与显卡驱动通讯完成渲染
  - macOS/iOS的常规攻击面
- 苹果是如何实现自己的虚拟显卡实现更高的效率的呢?
  - 是否会引入新的漏洞呢?
  - 是否可以用来实现虚拟机逃逸呢?

Apple Hypervisor Framework



Apple Hypervisor Framework



com.apple.driver.AppleParavirtGPU.kext

```
__int64 __fastcall AppleParavirtResourceHeapNamespace::getHandle(__int64 this, unsigned int a2)
{
  if ( *(_DWORD *)(this + 12) <= a2 )
    _os_log_internal(&dword_0, (os_log_t)APVLog, OS_LOG_TYPE_ERROR, "%s: Object index out of range\n", "getHandle");
  return *(_QWORD *)(*(_QWORD *)(this + 24) + 8LL * a2);
}
```

```
v7 = AppleParavirtResourceHeapNamespace::getHandle(a1->ResourceHeapNamespace, v4);
if ( !v7 )
{
  v8 = *(unsigned int *)a2;
  goto LABEL_10;
}
v6 = *(unsigned int *)(v7 + 24);
v8 = *(unsigned int *)a2;
if ( (_DWORD)v6 != (_DWORD)v8 )
{
LABEL_10:
  _os_log_internal(
    &dword_0,
    (os_log_t)APVLog,
    OS_LOG_TYPE_ERROR,
    "%s: Invalid object index: %d handleID: %d argID: %d\n",
    "deleteObject",
    *(unsigned int *)(a2 + 4),
    v6,
    v8);
  v18 = (IOLock **)a1->AppleParavirtAccelerator;
  (*((void (__fastcall **)(IOLock **, void *, _QWORD))*v18 + 267))(v18, &unk_141BD, 0LL);
  IOGraphicsAccelerator2::unlock_busy((IOGraphicsAccelerator2 *)v18);
  IOLockUnlock(v18[17]);
  return 0xE00002C2;
}
```

- 存在明显的代码质量问题
  - 错误的条件检测导致崩溃
- 混乱的代码更值得深入挖掘
  - 代码没有经过充分测试
  - 甚至可能只是一个能跑通的“半成品”

com.apple.driver.AppleParavirtGPU.kext

```
__int64 __fastcall AppleParavirtAccelerator::setupFIFO(AppleParavirtAccelerator *this)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  v1 = this;
  *((_DWORD *)this + 903) = 0;
  v2 = IOBufferMemoryDescriptor::withOptions(0x890u, (vm_size_t)&loc_10000, 1uLL);
  *((_QWORD *)this + 447) = v2;
  if ( v2 )
  {
    if ( ((unsigned int (__fastcall *)(IOBufferMemoryDescriptor *, _QWORD))v2->baseclass_0.baseclass_0.baseclass_0.__vftable[3].taggedRelease_0)(
           v2,
           0LL) )
    {
      LODWORD(v1) = 0;
      _os_log_internal(
        &dword_0,
        (os_log_t)APVLog,
        OS_LOG_TYPE_ERROR,
        "%s: Failed to prepare FIFO memory allocation.\n",
        "setupFIFO");
    }
    else
    {
      v3 = (*(__int64 (__fastcall **)(_QWORD, _QWORD))(**((_QWORD **)this + 447) + 512LL))(*((_QWORD *)this + 447), 0LL);
      *((_QWORD *)this + 448) = v3;
      *((_DWORD *)this + 902) = (*(__int64 (__fastcall **)(__int64))(*(_QWORD *)v3 + 296LL))(v3) - 4096;
      v4 = *((_QWORD *)this + 448);
      if ( v4 )
      {
        *((_QWORD *)v1 + 450) = (*(__int64 (__fastcall **)(__int64))(*(_QWORD *)v4 + 280LL))(v4);
        *((_QWORD *)v1 + 449) = (*(__int64 (__fastcall **)(_QWORD))(**((_QWORD **)v1 + 448) + 280LL))(*((_QWORD *)v1 + 448))
                              + 4096;
        bzero(*((void **)v1 + 450), 0x1000uLL);
        *(_DWORD *)(*((_QWORD *)v1 + 453) + 48LL) = IOMemoryDescriptor::getPhysicalAddress(*((IOMemoryDescriptor **)v1
                                                                                            + 447)) >> 12;
        v5 = (*(__int64 (__fastcall **)(_QWORD))(**((_QWORD **)v1 + 447) + 456LL))(*((_QWORD *)v1 + 447));
        v6 = (_DWORD *)*((_QWORD *)v1 + 453);
        v6[1] = v5;
        v6[4] = 4096;
        *v6 = 1;
        LOBYTE(v1) = 1;
      }
      else
      {
        LODWORD(v1) = 0;
        _os_log_internal(&dword_0, (os_log_t)APVLog, OS_LOG_TYPE_ERROR, "%s: Failed to map the FIFO.\n", "setupFIFO");
      }
    }
  }
  else
  {
    LODWORD(v1) = 0;
    _os_log_internal(
      &dword_0,
      (os_log_t)APVLog,
      OS_LOG_TYPE_ERROR,
      "%s: Failed to get FIFO memory allocation.\n",
      "setupFIFO");
  }
  return (unsigned int)v1;
}
```

- FIFO 是虚拟显卡与HOST通讯的主要手段
  - 分析FIFO请求的数据结构
  - 通过内核的Inline Hook嗅探请求的数据
  - 通过dumb fuzzing 确定攻击面
- 通过崩溃帮助我们迅速定位攻击面的代码

ParavirtualizedGraphics.Framework

```
void __cdecl -[PGFIFO processFifo](PGFIFO *self, SEL a2)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  v27 = *(_QWORD *)__stack_chk_guard;
  v3 = (unsigned int)-[PGFIFO commandLength](self, "commandLength");
  v19 = malloc(v3);
  if ( !v19 )
    abort();
  v18 = (unsigned int)-[PGFIFO commandLength](self, "commandLength");
  v25 = "CmdDeprecated:stampValue:withPayload:payloadSize:";
  while ( !self->_quiesced )
  {
    -[PGFIFO latchCommandOffset](self, "latchCommandOffset");
    v4 = 0;
    if ( (unsigned __int8)-[PGFIFO getFifoBytes:into:](self, "getFifoBytes:into:", 0xCLL, header) )
    {
      if ( *(unsigned int *)&header[4] < 0xCuLL || *(_DWORD *)&header[4] > v18 )
      {
        v8 = PGHostLog;
        v4 = 2;
        if ( os_log_type_enabled(PGHostLog, OS_LOG_TYPE_ERROR) )
          sub_7FFF6F806A3E(buf, &buf[1], v8);
        goto LABEL_21;
      }
```

ParavirtualizedGraphics.Framework

```
case 0x34:
  v14 = "CmdInvalidateResources:stampValue:withPayload:payloadSize:";
  goto LABEL_31;
case 0x35:
  v14 = "CmdSynchronizeResources:stampValue:withPayload:payloadSize:";
  goto LABEL_31;
case 0x36:
  v14 = "CmdDeleteIOSurfaceBacking2:stampValue:withPayload:payloadSize:";
  goto LABEL_31;
case 0x37:
  v14 = "CmdExecIndirect2:stampValue:withPayload:payloadSize:";
  goto LABEL_31;
case 0x38:
  v14 = "CmdDefineTask2:stampValue:withPayload:payloadSize:";
  goto LABEL_31;
case 0x39:
  v14 = "CmdMapMemory2:stampValue:withPayload:payloadSize:";
  goto LABEL_31;
case 0x3A:
  v14 = "CmdGetDeviceInfo:stampValue:withPayload:payloadSize:";
  goto LABEL_31;
case 0x3B:
  v14 = "CmdGetComputeInfo:stampValue:withPayload:payloadSize:";
  goto LABEL_31;
case 0x3C:
  v14 = "CmdReplacePhysical:stampValue:withPayload:payloadSize:";
  goto LABEL_31;
case 0x3D:
  v14 = "CmdDelay:stampValue:withPayload:payloadSize:";

  v13 = self;
  objc_msgSend_0(self, v14, header, *(unsigned int *)&header[8], v19, v7);
```

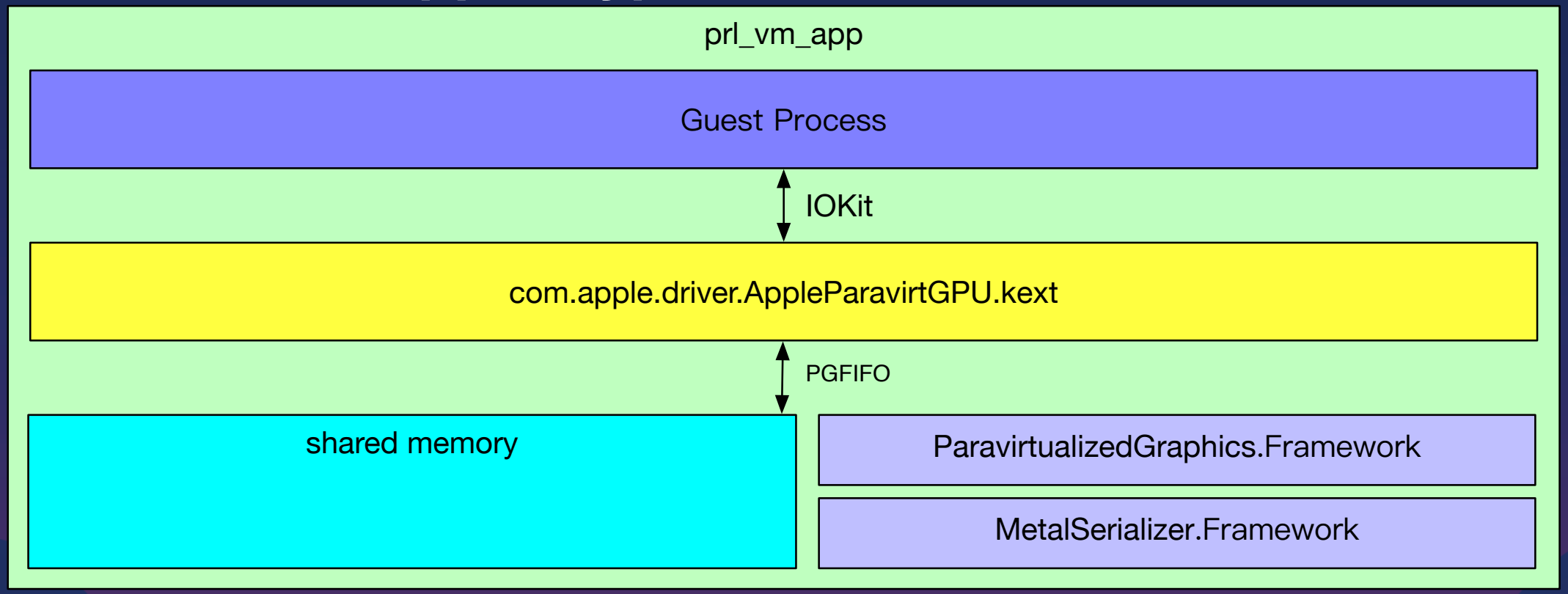
- 逆向工作从虚拟机内核来到的了Host的应用层程序
- FIFO请求种多而且复杂，意味着巨大的攻击面
- 根据崩溃日志的调用栈，继续分析CmdExecIndirect2函数

ParavirtualizedGraphics.Framework

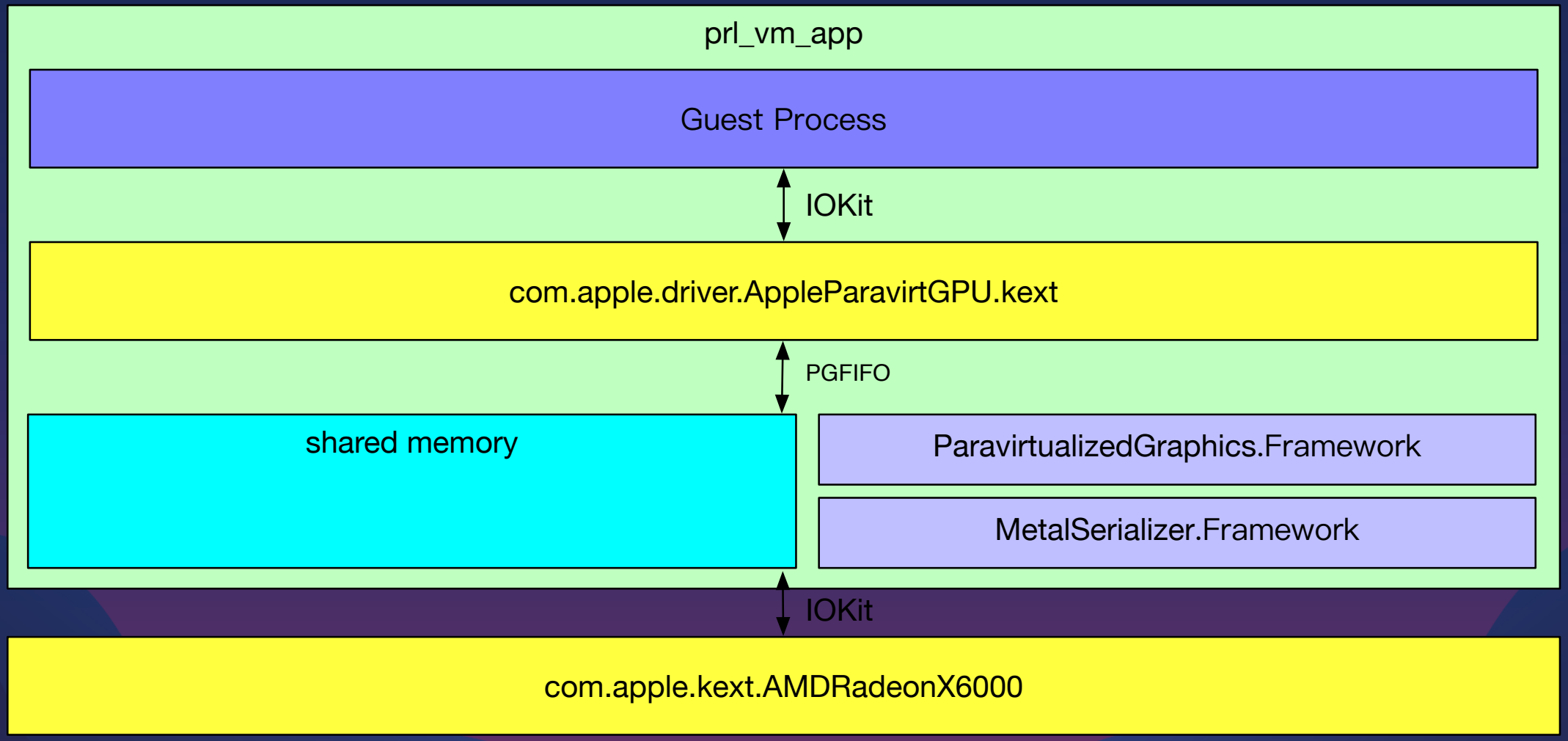
```
void __cdecl -[PGFIFO CmdExecIndirect2:stampValue:withPayload:payloadSize:](PGFIFO *self, SEL a2, $AB56C86F963
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  v62 = a4;
  v42[0] = 0LL;
  v42[1] = v42;
  v42[2] = 0x3810000000LL;
  v42[3] = " `";
  context = objc_autoreleasePoolPush();
  if ( fifo_length <= 0xB )
  {
    v31 = PGHostLog;
    if ( os_log_type_enabled(PGHostLog, OS_LOG_TYPE_ERROR) )
      sub_7FFF6F806488((uint8_t *)&buf, v31);
    goto LABEL_27;
  }
  count = *(unsigned int *)(fifo_input + 4);
  if ( 24 * count == 24 * (_DWORD)count )
  {
    v10 = 16LL * *(unsigned int *)(fifo_input + 8);
    if ( v10 == 16 * *(_DWORD *)(fifo_input + 8) )
    {
      v11 = 24 * count + 12;
      v12 = __CFADD__((_DWORD)v11, (_DWORD)v10);
      v13 = v11 + v10;
      if ( v11 == (v11 & 0xFFFFFFFC) && !v12 )
      {
        if ( v13 <= fifo_length )
        {
          v14 = -[_PGDevice getTaskID:](self->_device, "getTaskID:", *(unsigned int *)fifo_input);
          if...
          objc_retain_0(v14);
          objc_retain_0(v14);
          v43[0] = 0LL;
          v43[1] = v43;
          v43[2] = 0x2020000000LL;
          v43[3] = 0LL;
          buf = 0LL;
          v45 = (__int64)&buf;
          v46 = 0x3052000000LL;
          v47 = sub_7FFF6F800300;
          v48 = sub_7FFF6F800317;
          v16 = -[_PGDevice execQueue](self->_device, "execQueue");
          v49 = objc_msgSend_0(v16, "commandBuffer");
          v17 = MTLTraceEnabledSPI((__int64)v16);
          v59 = v62;
          if ( v17 )
          {
            v60 = -[_PGDevice deviceTraceId](self->_device, "deviceTraceId");
            v18 = *(unsigned int *)fifo_input;
            v19 = (unsigned int)objc_msgSend_0(self, "stampIndex");
            kdebug_trace(0x85310401LL, v60, v18, v19, v59);
          }
          objc_msgSend_stret(
            (__int64)&a1,
            (SEL)v14,
            "prepareResources:count:",
            fifo_input + 12,
            *(unsigned int *)(fifo_input + 4));
          v54 = fifo_input + 12;
          LOBYTE(v60) = a1;
          v58[0] = *(_DWORD *)((char *)&a1 + 1);
          *(_DWORD *)((char *)v58 + 3) = HIDWORD(a1);
```

```
objc_msgSend_stret(
  (__int64)&a1,
  (SEL)v14,
  "prepareResources:count:",
  fifo_input + 12,
  *(unsigned int *)(fifo_input + 4));
v54 = fifo_input + 12;
LOBYTE(v60) = a1;
v58[0] = *(_DWORD *)((char *)&a1 + 1);
*(_DWORD *)((char *)v58 + 3) = HIDWORD(a1);
v56 = v51;
v57 = v52;
if ( (unsigned __int8)MTLTraceEnabledSPI((__int64)&a1) )
{
  v55 = -[_PGDevice deviceTraceId](self->_device, "deviceTraceId");
  v20 = *(unsigned int *)fifo_input;
  v21 = (unsigned int)objc_msgSend_0(self, "stampIndex");
  kdebug_trace(2234582018LL, (__int64)v55, v20, v21, v59);
}
if ( (v60 & 1) != 0 )
  objc_msgSend_0(*(id *)(v45 + 40), "encodeWaitForEvent:value:", v56, v57);
*(_QWORD *)v34.gap0 = _NSConcreteStackBlock;
*(_QWORD *)&v34.gap0[8] = 0xC2000000LL;
*(_QWORD *)&v34.gap0[16] = sub_7FFF6F80032A;
*(_QWORD *)&v34.gap0[24] = &unk_7FFF87016648;
v34.ret = v43;
v34.fifo_input = fifo_input;
v34.fifo_channel = self;
v34.fifo_payload = fifo_input + 24 * count + 12;
v34.device = v14;
v34.stampvalue = v62;
v34.qword38 = &buf;
LOBYTE(v34.qword58) = v60;
*(_DWORD *)((char *)&v34.qword58 + 1) = v58[0];
HIDWORD(v34.qword58) = *(_DWORD *)((char *)v58 + 3);
v34.qword60 = v56;
v34.qword68 = v57;
v34.qword40 = v42;
objc_msgSend_0(v14, "runBlock:", &v34);
```

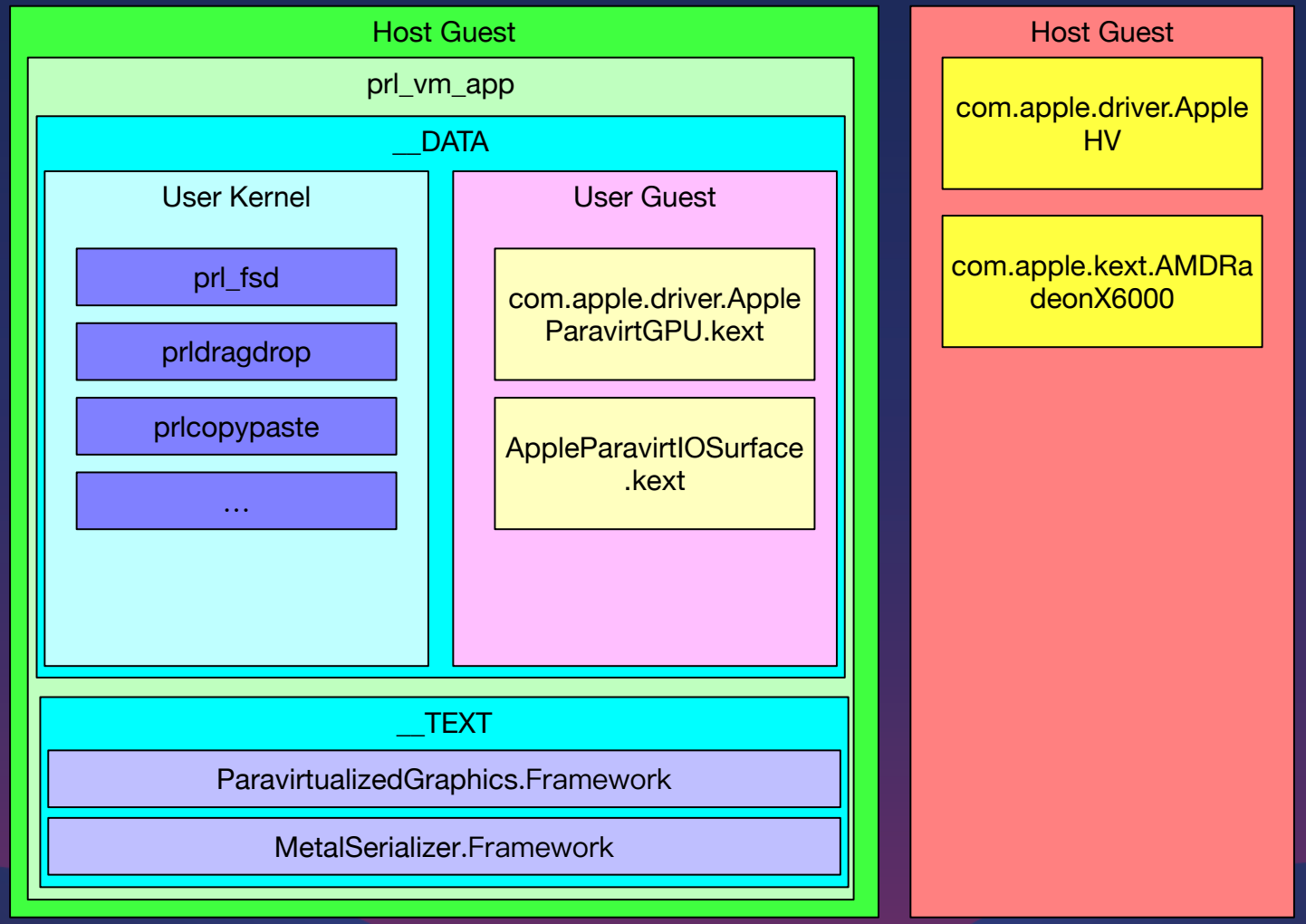
Apple Hypervisor Framework



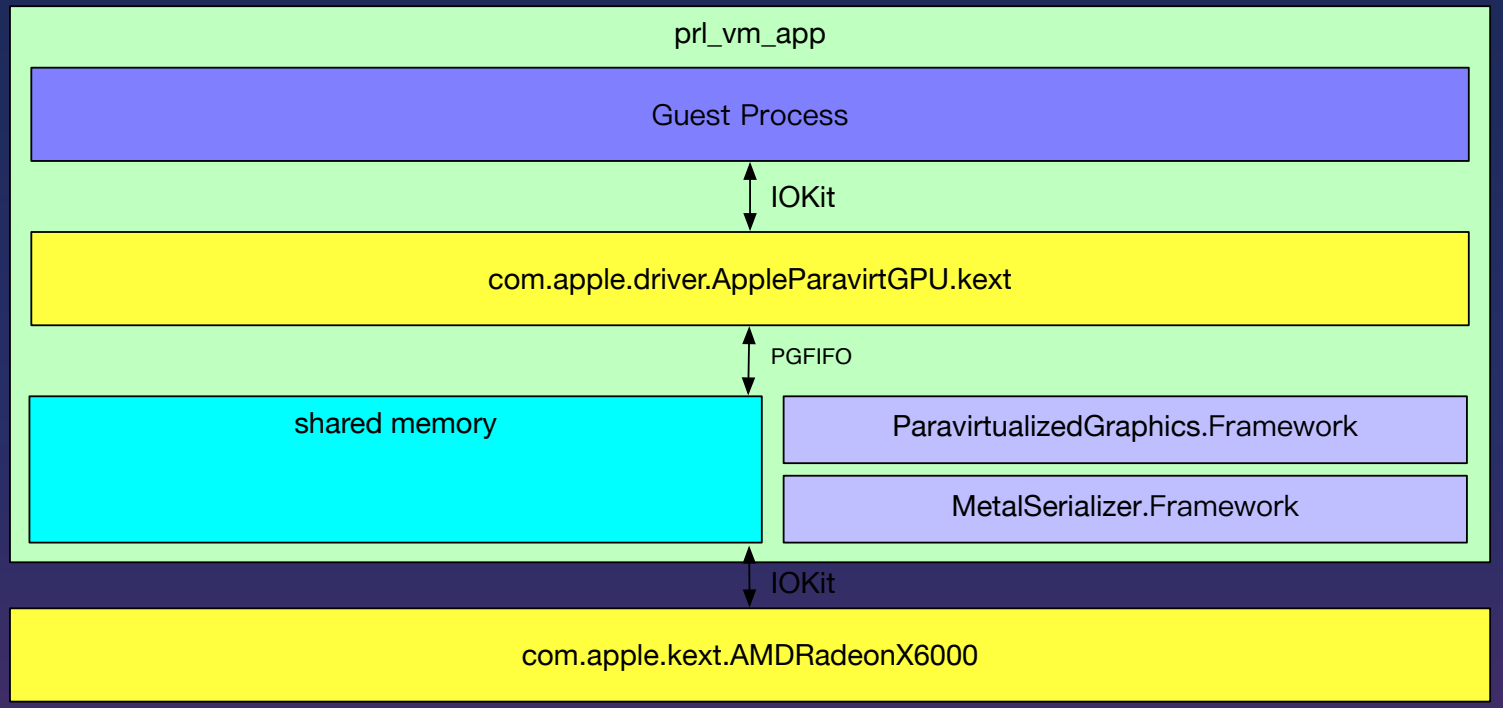
Apple Hypervisor Framework



Apple Hypervisor Framework



Apple Hypervisor Framework



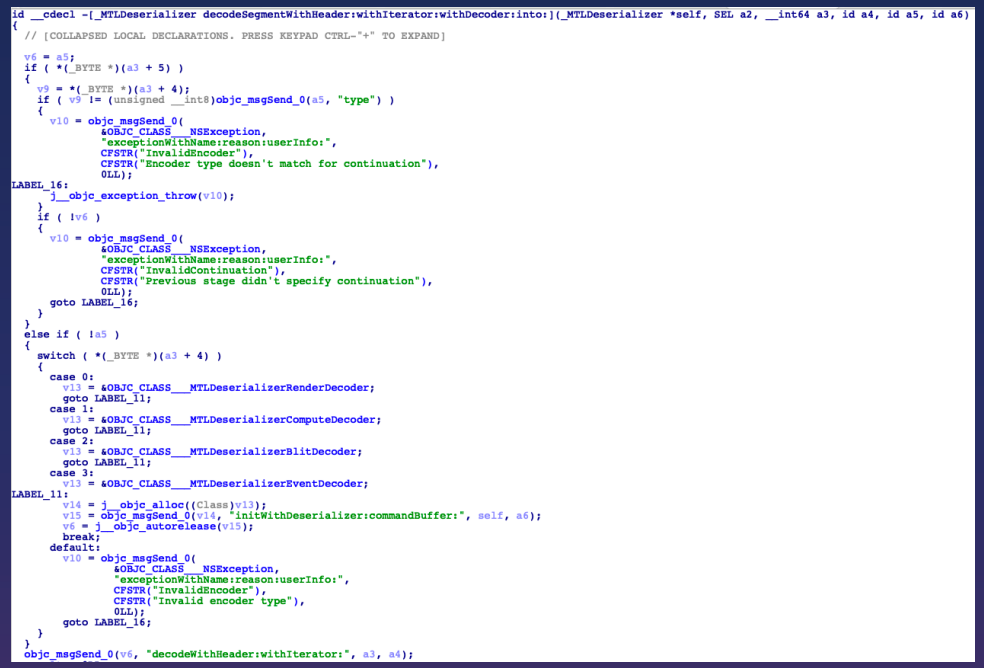
- Guest Process通过共享内存的方式使用Host的显卡完成渲染
- 攻击面1：com.apple.driver.AppleParavirtGPU.kext
- 攻击面2：ParavirtualizedGraphics.Framework、MetalSerializer.Framework

ParavirtualizedGraphics.Framework

```
void __fastcall sub_7FFF6F80032A(struct_032a *a1)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  *(_QWORD *)(*(_QWORD *)(a1->ret + 8LL) + 24LL) = malloc(16LL * *(unsigned int *)(a1->fifo_input + 8));
  if ( *(_QWORD *)(*(_QWORD *)(a1->ret + 8LL) + 24LL) )
  {
    v2 = a1->fifo_channel;
    v23 = (unsigned int)objc_msgSend_0(v2, "currentCommandOffset");
    v3 = *(_QWORD *)(*(_QWORD *)(a1->ret + 8LL) + 24LL) + 8LL * *(unsigned int *)(a1->fifo_input + 8);
    if ( *(_DWORD *)(a1->fifo_input + 8) )
    {
      v4 = 8LL;
      v5 = 0LL;
      do
      {
        v6 = a1->fifo_payload;
        v7 = *(_QWORD *)(v6 + v4);
        *(_QWORD *)(v3 + 8 * v5) = v7;
        v2 = a1->device;
        *(_QWORD *)(*(_QWORD *)(*(_QWORD *)(a1->ret + 8LL) + 24LL) + 8 * v5++) = objc_msgSend_0(
                                                                                   v2,
                                                                                   "mappedAddressForOffset:length:",
                                                                                   *(_QWORD *)(v6 + v4 - 8),
                                                                                   v7);
        v4 += 16LL;
      }
      while ( v5 < *(unsigned int *)(a1->fifo_input + 8) );
    }
    if ( (unsigned __int8)MTLTraceEnabledSPI((__int64)v2) )
    {
      v8 = -[_PGDevice deviceTraceId](a1->fifo_channel->_device, "deviceTraceId");
      v9 = *(unsigned int *)a1->fifo_input;
      v10 = (unsigned int)objc_msgSend_0(a1->fifo_channel, "stampIndex");
      kdebug_trace(0x85310405LL, (__int64)v8, v9, v10, a1->stampvalue);
    }
    v11 = objc_msgSend_0(a1->device, "deserializer");
    objc_msgSend_0(
      v11,
      "decodeSegments:lengths:count:into:",
      *(_QWORD *)(*(_QWORD *)(a1->ret + 8LL) + 24LL),
      v3,
      *(unsigned int *)(a1->fifo_input + 8),
      *(_QWORD *)(*(_QWORD *)(a1->qword38 + 8LL) + 40LL));
    if ( (unsigned __int8)MTLTraceEnabledSPI((__int64)v11) )
    {
      v12 = -[_PGDevice deviceTraceId](a1->fifo_channel->_device, "deviceTraceId");
      v13 = *(unsigned int *)a1->fifo_input;
      v14 = (unsigned int)objc_msgSend_0(a1->fifo_channel, "stampIndex");
      kdebug_trace(0x85310406LL, (__int64)v12, v13, v14, a1->stampvalue);
    }
```

- Guest的数据会被decodeSegments函数解析
- 通过Frida编写被动Fuzz工具，在应用层发起fuzz
- 获得大量崩溃
  - 几乎瞬间就会崩溃
- 人工审计介入

ParavirtualizedGraphics.Framework



```
id __cdecl -[_MTLDeserializer decodeSegmentWithHeader:withIterator:withDecoder:into:](_MTLDeserializer *self, SEL a2, __int64 a3, id a4, id a5, id a6)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  v6 = a5;
  if ( *(_BYTE *)(a3 + 5) )
  {
    v9 = *(_BYTE *)(a3 + 4);
    if ( v9 != (unsigned __int8)objc_msgSend_0(a5, "type") )
    {
      v10 = objc_msgSend_0(
              &OBJC_CLASS___NSException,
              "exceptionWithName:reason:userInfo:",
              CFSTR("InvalidEncoder"),
              CFSTR("Encoder type doesn't match for continuation"),
              0LL);
LABEL_16:
      _j__objc_exception_throw(v10);
    }
    if ( !v6 )
    {
      v10 = objc_msgSend_0(
              &OBJC_CLASS___NSException,
              "exceptionWithName:reason:userInfo:",
              CFSTR("InvalidContinuation"),
              CFSTR("Previous stage didn't specify continuation"),
              0LL);
      goto LABEL_16;
    }
  }
  else if ( !a5 )
  {
    switch ( *(_BYTE *)(a3 + 4) )
    {
      case 0:
        v13 = &OBJC_CLASS___MTLDeserializerRenderDecoder;
        goto LABEL_11;
      case 1:
        v13 = &OBJC_CLASS___MTLDeserializerComputeDecoder;
        goto LABEL_11;
      case 2:
        v13 = &OBJC_CLASS___MTLDeserializerBlitDecoder;
        goto LABEL_11;
      case 3:
        v13 = &OBJC_CLASS___MTLDeserializerEventDecoder;
LABEL_11:
        v14 = _j__objc_alloc((Class)v13);
        v15 = objc_msgSend_0(v14, "initWithDeserializer:commandBuffer:", self, a6);
        v6 = _j__objc_autorelease(v15);
        break;
      default:
        v10 = objc_msgSend_0(
                &OBJC_CLASS___NSException,
                "exceptionWithName:reason:userInfo:",
                CFSTR("InvalidEncoder"),
                CFSTR("Invalid encoder type"),
                0LL);
        goto LABEL_16;
    }
  }
  objc_msgSend_0(v6, "decodeWithHeader:withIterator:", a3, a4);
```

```
void __cdecl -[MTLDeserializerComputeDecoder decodeWithHeader:withIterator:]
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  v8 = "decodeUseHeaps:withIterator:";
  v9 = "decodeUseResources:withIterator:";
  v10 = "decodeBarrierResources:withIterator:";
  v11 = "decodeBarrierScope:withIterator:";
  v12 = "decodeWaitForFence:withIterator:";
  v13 = "decodeUpdateFence:withIterator:";
  v14 = "decodeSetThreadgroupMemoryLength:withIterator:";
  v15 = "decodeSetStageInRegionIndirect:withIterator:";
  v16 = "decodeSetStageInRegion:withIterator:";
  v17 = "decodeSetPipelineState:withIterator:";
  v18 = "decodeSetBufferOffset:withIterator:";
  v19 = "decodeSetTextures:withIterator:";
  v20 = "decodeSetSamplersLODClamp:withIterator:";
  v21 = "decodeSetSamplers:withIterator:";
  v22 = "decodeSetBuffers:withIterator:";
  v23 = "decodeDispatchThreads:withIterator:";
  v24 = "decodeDispatchThreadgroupsIndirect:withIterator:";
  v25 = "decodeDispatchThreadgroups:withIterator:";
  while ( (unsigned __int64)objc_msgSend_0(a4, "bytesLeft") >= 8 )
  {
    objc_msgSend_0(a4, "readBytes:into:", 8LL, &sel);
    switch ( sel )
    {
      case 0xC8:
        objc_msgSend_0(self, v25, &sel, a4);
        break;
```

- Guest端输入的数据会被docdeXXX函数解析成各种显卡渲染时需要用到的数据
- [MTLDeserializerComputeDecoder decodeWithHeader:withIterator]负责解析Guest输入的数据
  - a3是可控的data，a4是data的长度

ParavirtualizedGraphics.Framework

```
void __cdecl -[MTLDeserializerComputeDecoder decodeWithHeader:withIterator:]
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  v8 = "decodeUseHeaps:withIterator:";
  v9 = "decodeUseResources:withIterator:";
  v10 = "decodeBarrierResources:withIterator:";
  v11 = "decodeBarrierScope:withIterator:";
  v12 = "decodeWaitForFence:withIterator:";
  v13 = "decodeUpdateFence:withIterator:";
  v14 = "decodeSetThreadgroupMemoryLength:withIterator:";
  v15 = "decodeSetStageInRegionIndirect:withIterator:";
  v16 = "decodeSetStageInRegion:withIterator:";
  v17 = "decodeSetPipelineState:withIterator:";
  v18 = "decodeSetBufferOffset:withIterator:";
  v19 = "decodeSetTextures:withIterator:";
  v20 = "decodeSetSamplersLODClamp:withIterator:";
  v21 = "decodeSetSamplers:withIterator:";
  v22 = "decodeSetBuffers:withIterator:";
  v23 = "decodeDispatchThreads:withIterator:";
  v24 = "decodeDispatchThreadgroupsIndirect:withIterator:";
  v25 = "decodeDispatchThreadgroups:withIterator:";
  while ( (unsigned __int64)objc_msgSend_0(a4, "bytesLeft") >= 8 )
  {
    objc_msgSend_0(a4, "readBytes:into:", 8LL, &sel);
    switch ( sel )
    {
      case 0xC8:
        objc_msgSend_0(self, v25, &sel, a4);
        break;
```

```
void __cdecl -[GFX10_MtlComputeCmdEncoder setThreadgroupMemoryLength:atIndex:](GFX10
{
  self->m_members.dynamicThreadgroupMemoryLength[a4] = a3;
  amdMtl_GFX10_SRDMgrBindThreadGroupLength(self->m_members.rsrcMgr + 0xA8, a4, a3);
}
```

- 到这里之后的工作其实并没有什么难度了，就是简单的代码审计
- 其中的一个decode中发现了明显的越界写漏洞
- 通过这个漏洞顺利实现虚拟机逃逸的目标

但你真的看“懂”这个漏洞了吗？？？

ParavirtualizedGraphics.Framework

```
id __cdecl -[_MTLDeserializer decodeSegmentWithHeader:withIterator:withDecoder:into:](_MTLDeserializer *self, SEL a2, __int64 a3, id a4, id a5, id a6)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  v6 = a5;
  if ( *(_BYTE *)(a3 + 5) )
  {
    v9 = *(_BYTE *)(a3 + 4);
    if ( v9 != (unsigned __int8)objc_msgSend_0(a5, "type") )
    {
      v10 = objc_msgSend_0(
              &OBJC_CLASS___NSException,
              "exceptionWithName:reason:userInfo:",
              CFSTR("InvalidEncoder"),
              CFSTR("Encoder type doesn't match for continuation"),
              0LL);
LABEL_16:
      j__objc_exception_throw(v10);
    }
    if ( !v6 )
    {
      v10 = objc_msgSend_0(
              &OBJC_CLASS___NSException,
              "exceptionWithName:reason:userInfo:",
              CFSTR("InvalidContinuation"),
              CFSTR("Previous stage didn't specify continuation"),
              0LL);
      goto LABEL_16;
    }
  }
  else if ( !a5 )
  {
    switch ( *(_BYTE *)(a3 + 4) )
    {
      case 0:
        v13 = &OBJC_CLASS___MTLDeserializerRenderDecoder;
        goto LABEL_11;
      case 1:
        v13 = &OBJC_CLASS___MTLDeserializerComputeDecoder;
        goto LABEL_11;
      case 2:
        v13 = &OBJC_CLASS___MTLDeserializerBlitDecoder;
        goto LABEL_11;
      case 3:
        v13 = &OBJC_CLASS___MTLDeserializerEventDecoder;
LABEL_11:
        v14 = j__objc_alloc((Class)v13);
        v15 = objc_msgSend_0(v14, "initWithDeserializer:commandBuffer:", self, a6);
        v6 = j__objc_autorelease(v15);
        break;
      default:
        v10 = objc_msgSend_0(
                &OBJC_CLASS___NSException,
                "exceptionWithName:reason:userInfo:",
                CFSTR("InvalidEncoder"),
                CFSTR("Invalid encoder type"),
                0LL);
        goto LABEL_16;
    }
  }
  objc_msgSend_0(v6, "decodeWithHeader:withIterator:", a3, a4);
```

ParavirtualizedGraphics.Framework

```
-[MTLDeserializerBlitDecoder decodeWithHeader:withIterator:]
-[MTLDeserializerEventDecoder decodeWithHeader:withIterator:]
-[MTLDeserializerRenderDecoder decodeWithHeader:withIterator:]
-[MTLDeserializerComputeDecoder decodeWithHeader:withIterator:]
```

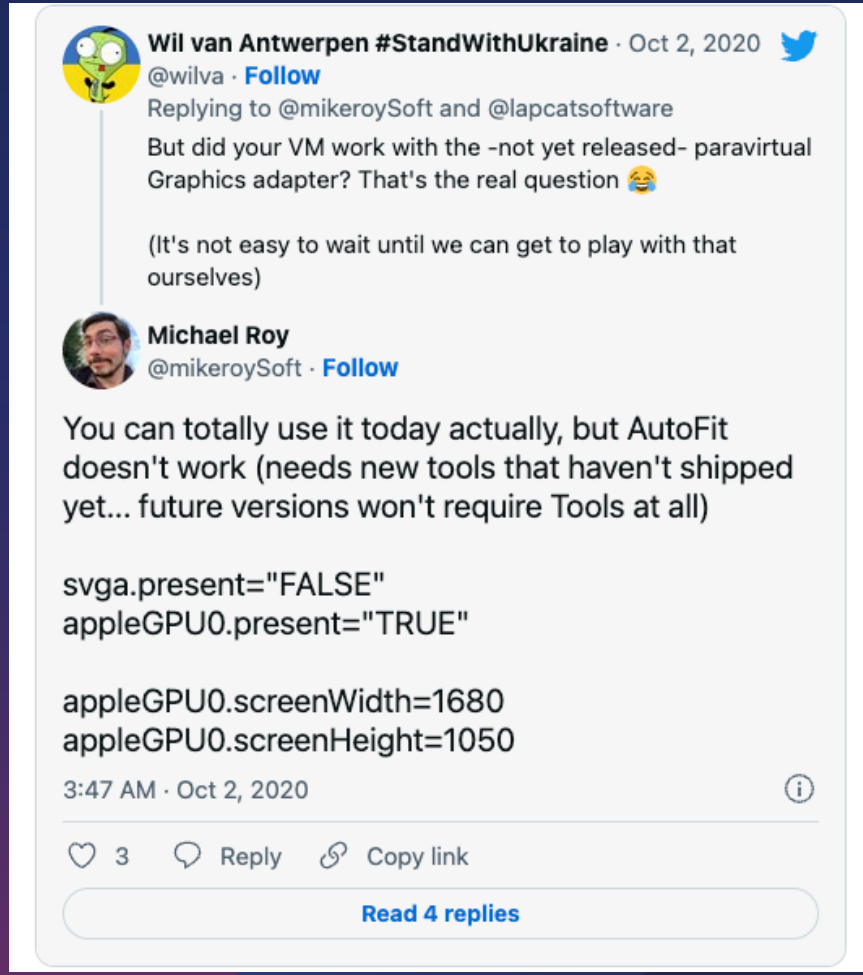
```
void __cdecl -[MTLDeserializerBlitDecoder decodeWithHeader:withIterator:]
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  v21 = a3;
  v22 = self;
  v10 = "decodeCopyFromTextureToTextureWithNumSliceLevel:withIterator:";
  v11 = "decodeBlitWaitForFence:withIterator:";
  v12 = "decodeBlitUpdateFence:withIterator:";
  v13 = "decodeSynchronizeTextureImage:withIterator:";
  v8 = "decodeOptimizeImage:withIterator:";
  v7 = "decodeSynchronizeResource:withIterator:";
  v9 = "decodeOptimize:withIterator:";
  v14 = "decodeGenerateMipmaps:withIterator:";
  v15 = "decodeFillBuffer:withIterator:";
  v16 = "decodeCopyFromTextureToTextureWithOptions:withIterator:";
  v17 = "decodeCopyFromTextureToTexture:withIterator:";
  v18 = "decodeCopyFromTextureToBuffer:withIterator:";
  v19 = "decodeCopyFromBufferToBuffer:withIterator:";
  v20 = "decodeCopyFromBufferToTexture:withIterator:";
```

```
void __cdecl -[MTLDeserializerRenderDecoder decodeWithHeader:withIterator:](MTLDeseri
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  if ( !*(_BYTE *)(a3 + 5) )
  {
    v5 = -[MTLDeserializerRenderDecoder newDescriptor:](self, "newDescriptor:", a4);
    v6 = objc_msgSend_0(self->commandBuffer, "renderCommandEncoderWithDescriptor:", v
    self->renderEncoder = (MTLRenderCommandEncoderSPI *)objc_retain_0(v6);
    objc_release_0(v5);
  }
  while ( (unsigned __int64)objc_msgSend_0(a4, "bytesLeft") >= 8 )
  {
    objc_msgSend_0(a4, "readBytes:into:", 8LL, v9);
    switch ( v9[0] )
    {
      case 0:
        -[MTLDeserializerRenderDecoder decodeDrawPrimitives64:withIterator:](
          self,
          "decodeDrawPrimitives64:withIterator:",
          v9,
          a4);
        break;
      case 1:
        -[MTLDeserializerRenderDecoder decodeDrawPrimitives16:withIterator:](
          self,
          "decodeDrawPrimitives16:withIterator:",
          v9,
          a4);
        break;
      case 2:
        -[MTLDeserializerRenderDecoder decodeDrawInstancedPrimitives64:withIterator:]
          self,
          "decodeDrawInstancedPrimitives64:withIterator:",
          v9,
          a4);
        break;
      case 3:
        -[MTLDeserializerRenderDecoder decodeDrawInstancedPrimitives16:withIterator:]
          self,
          "decodeDrawInstancedPrimitives16:withIterator:",
          v9,
          a4);
```

- Decoder不止一种
- 所有的Decode函数都是攻击面
- 其他Decode函数中确实又发现了大量漏洞

这个“锅”谁来背?

- 漏洞本身是存在于苹果的代码中
  - 所有使用这个特性的虚拟机产品都会受这个漏洞的影响
- 但当时只有Parallels Desktop的用户受到该漏洞的影响
- 这个漏洞该如何披露
- 进行一次社会实验



漏洞上报&&修复

Additional recognition

Kernel

We would like to acknowledge Tao Huang for their assistance.

Metal

We would like to acknowledge Tao Huang for their assistance.

Hello,

Our development team has accessed the issue. This vulnerability is not isolated to Parallels Desktop itself but an issue in Apple's Hypervisor (that provides paravirtualization for MacOS) that's used on the both sides, host and guest. This is directly related to MacOS 12.0.1, in which Apple has already fixed.

Thank you,

议题小结

- 如何选择合适的研究目标
- 漏洞挖掘的思路与大致流程
- Parallels Hypervisor虚拟机逃逸的技术细节
- Apple Hypervisor虚拟机逃逸的技术细节

从Parallels Desktop逃逸到macOS逃逸之旅到此结束

特别感谢研究工作中提供帮助的同事和朋友

谢谢！