Read Out of Bounds in AppleIntelFramebufferAzul.kext

Zhuo Liang
Qihoo 360 Nirvan Team

August 20, 2018*

CVE-2018-4434: OOB Read

The issue lies in method **AppleIntelPAVP::GatedsetAttributeAUMsgWrite** resides in AppleIntelFramebufferAzul.kext. It seems that the driver did make an assumption that the size of the message passed into this method is always larger than 0x48C, which proved to be not true. The message from **AppleUpstreamUserClient** is a counterexample.

The 3rd method of AppleUpstreamUserClient, which can be opened through **AppleUpstreamUserClientDriver**, is **AppleUpstreamUserClient::AppleUpstreamSendMessage**. Users are capable of sending messages with controlled size to the driver and this could trigger method **AppleUpstreamUserClient::AppleUpstreamSendMessageEx**. The backtrace is shown below.

```
(lldbinit) bt
    frame #0: 0xffffff7f892508ec AppleUpstreamUserClient`
   AppleUpstreamUserClient::AppleUpstreamSendMessageEx
    frame #1: 0xffffff80040550ee kernel`IOWorkLoop::runAction
    frame #2: 0xffffff7f8924ff53 AppleUpstreamUserClient`
   AppleUpstreamUserClient::externalMethod
    frame #3: 0xffffff800408e59f kernel`::is_io_connect_method
    frame #4: 0xffffff8003a931f4 kernel`_Xio_connect_method
    frame #5: 0xffffff80039b210d kernel`ipc_kobject_server
    frame #6: 0xffffff800398cad5 kernel`ipc_kmsg_send
    frame #7: 0xffffff80039a148e kernel`mach_msg_overwrite_trap
    frame #8: 0xffffff8003abfceb kernel`mach_call_munger64
    frame #9: 0xffffff800395a486 kernel`hndl_mach_scall64
```

*Last update on January 13, 2019

The size of the buffer allocated in AppleUpstreamUser-Client::AppleUpstreamSendMessageEx used for restoring user message is 0x20 bigger than user message size, shown in list below.

```
_int64 __fastcall AppleUpstreamUserClient::AppleUpstreamSendMessageEx
    (AppleUpstreamUserClient *this, void *a2, unsigned __int8 *a3,
    unsigned int a4)
{
  v4 = a4;
  v5 = a3;

  if ( *((_DWORD *)a2 + 6) <= 0xEu ){
    v7 = __ROL2__(*((_WORD *)v5 + 7), 8);
    v6 = -536870206;
    if ( v7 ){
      LOWORD(v8) = v7 + 16;
        if ( (_WORD)v8 ){
          v8 = (unsigned __int16)v8;
            if ( (unsigned __int16)v8 == v4 ){
              v15 = v8;
              v16 = v8 + 32; //(a)
              v9 = (_QWORD *)((__int64 (*)(void))IOMalloc)();
  ...
```

This buffer will be transferd to method **AppleIntelPAVP::GatedsetAttributeAUMsgWrite** later.

```
(lldbinit) bt
    frame #0: 0xffffff7f85d11e12 AppleIntelFramebufferAzul`
   AppleIntelPAVP::GatedsetAttributeAUMsgWrite
    frame #1: 0xffffff7f85d11df9 AppleIntelFramebufferAzul`
   AppleIntelPAVP::setAttributeAUMsgWrite
    frame #2: 0xffffff7f85d01341 AppleIntelFramebufferAzul`
   AppleIntelFramebuffer::setAttributeForUpstream
    frame #3: 0xffffff7f85cdd58b AppleIntelFramebufferAzul`
   AppleIntelFramebuffer::setAttributeForConnection
    frame #4: 0xffffff7f892511ce AppleUpstreamUserClient`
   AppleUpstreamUserClient::SendNextMessageToDriver
    frame #5: 0xffffff7f89250a7c AppleUpstreamUserClient`
   AppleUpstreamUserClient::AppleUpstreamSendMessageEx
    frame #6: 0xffffff80040550ee kernel`IOWorkLoop::runAction
    ...
```

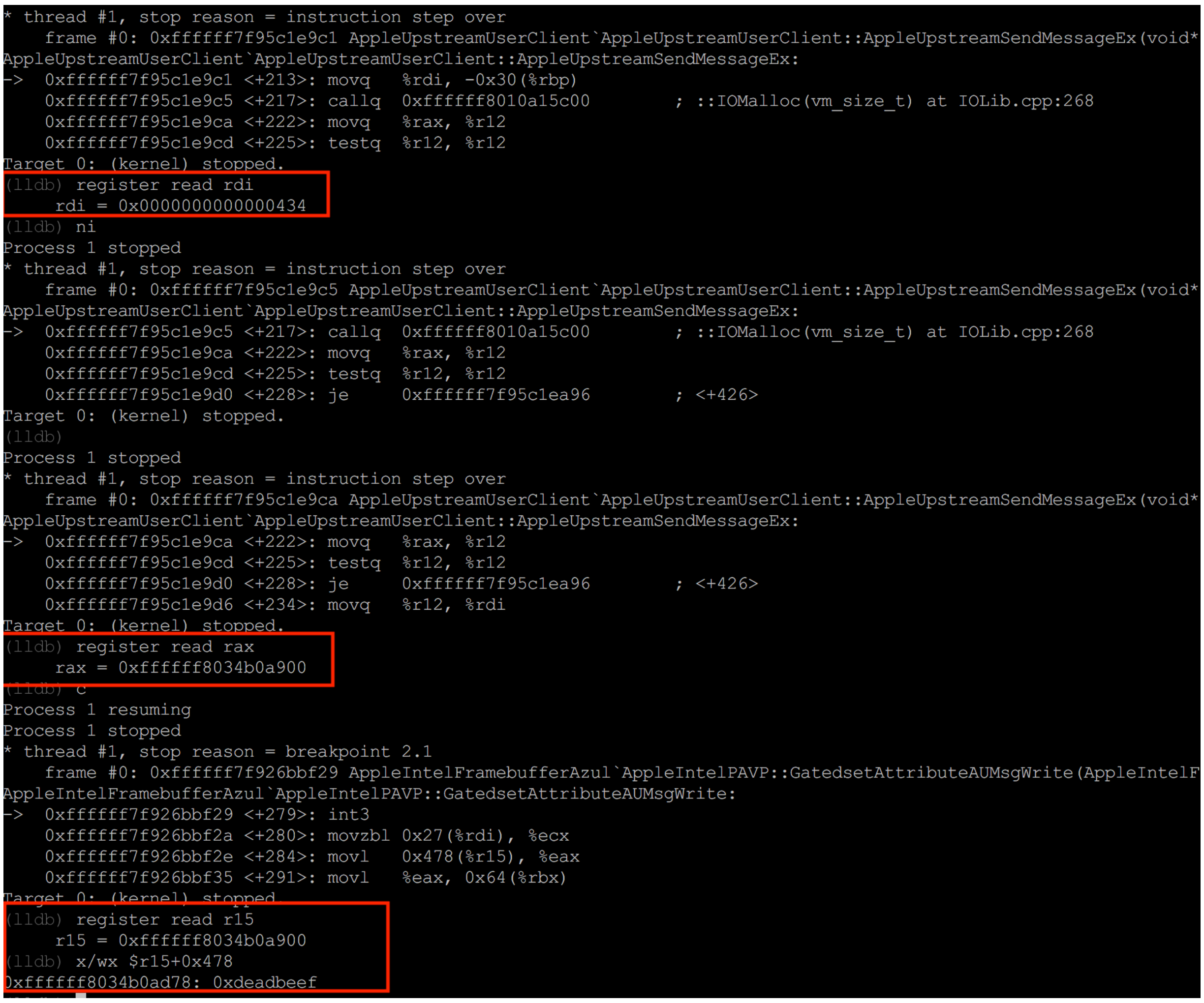
This method would read 4 bytes at (b) from offset 0x478 of the buffer allocated before.

```
// AppleIntelPAVP::GatedsetAttributeAUMsgWrite method
// Noteworthy: $r15 is the buffer
0000000000047F29  movzx   ecx, byte ptr [r15+27h]
```

```
0000000000047F2E  mov     eax, [r15+478h] // (b)
0000000000047F35  mov     [rbx+64h], eax
0000000000047F38  mov     [r14+200h], eax
```

Next image is a demonstration of the proof of concept attached. Note that the version of the experiment environment is as follow:

ProductName	Mac OS X
ProductVersion	10.14
BuildVersion	18A371a



```
* thread #1, stop reason = instruction step over
    frame #0: 0xffffff7f95c1e9c1 AppleUpstreamUserClient`AppleUpstreamUserClient::AppleUpstreamSendMessageEx(void*
AppleUpstreamUserClient`AppleUpstreamUserClient::AppleUpstreamSendMessageEx:
->  0xffffff7f95c1e9c1 <+213>: movq   %rdi, -0x30(%rbp)
    0xffffff7f95c1e9c5 <+217>: callq  0xffffff8010a15c00        ; ::IOMalloc(vm_size_t) at IOLib.cpp:268
    0xffffff7f95c1e9ca <+222>: movq   %rax, %r12
    0xffffff7f95c1e9cd <+225>: testq  %r12, %r12
Target 0: (kernel) stopped.
(lldb) register read rdi
     rdi = 0x0000000000000434
(lldb) ni
Process 1 stopped
* thread #1, stop reason = instruction step over
    frame #0: 0xffffff7f95c1e9c5 AppleUpstreamUserClient`AppleUpstreamUserClient::AppleUpstreamSendMessageEx(void*
AppleUpstreamUserClient`AppleUpstreamUserClient::AppleUpstreamSendMessageEx:
->  0xffffff7f95c1e9c5 <+217>: callq  0xffffff8010a15c00        ; ::IOMalloc(vm_size_t) at IOLib.cpp:268
    0xffffff7f95c1e9ca <+222>: movq   %rax, %r12
    0xffffff7f95c1e9cd <+225>: testq  %r12, %r12
    0xffffff7f95c1e9d0 <+228>: je     0xffffff7f95c1ea96        ; <+426>
Target 0: (kernel) stopped.
(lldb)
Process 1 stopped
* thread #1, stop reason = instruction step over
    frame #0: 0xffffff7f95c1e9ca AppleUpstreamUserClient`AppleUpstreamUserClient::AppleUpstreamSendMessageEx(void*
AppleUpstreamUserClient`AppleUpstreamUserClient::AppleUpstreamSendMessageEx:
->  0xffffff7f95c1e9ca <+222>: movq   %rax, %r12
    0xffffff7f95c1e9cd <+225>: testq  %r12, %r12
    0xffffff7f95c1e9d0 <+228>: je     0xffffff7f95c1ea96        ; <+426>
    0xffffff7f95c1e9d6 <+234>: movq   %r12, %rdi
Target 0: (kernel) stopped.
(lldb) register read rax
     rax = 0xffffff8034b0a900
(lldb) c
Process 1 resuming
Process 1 stopped
* thread #1, stop reason = breakpoint 2.1
    frame #0: 0xffffff7f926bbf29 AppleIntelFramebufferAzul`AppleIntelPAVP::GatedsetAttributeAUMsgWrite(AppleIntelF
AppleIntelFramebufferAzul`AppleIntelPAVP::GatedsetAttributeAUMsgWrite:
->  0xffffff7f926bbf29 <+279>: int3
    0xffffff7f926bbf2a <+280>: movzbl 0x27(%rdi), %ecx
    0xffffff7f926bbf2e <+284>: movl   0x478(%r15), %eax
    0xffffff7f926bbf35 <+291>: movl   %eax, 0x64(%rbx)
Target 0: (kernel) stopped.
(lldb) register read r15
     r15 = 0xffffff8034b0a900
(lldb) x/wx $r15+0x478
0xffffff8034b0ad78: 0xdeadbeef
```