Debug for bug: Crack and Hack Apple Core by itself

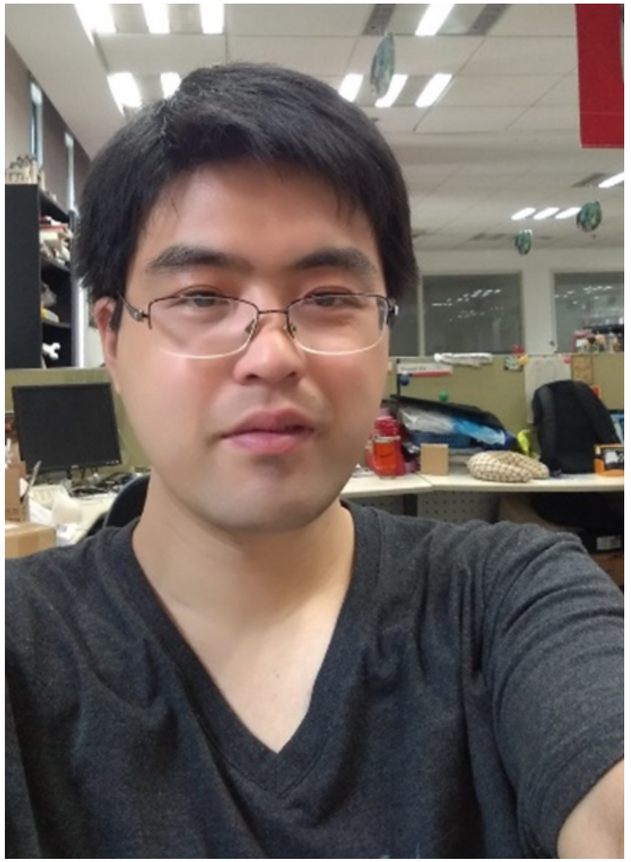
Lilang Wu, Moony Li

About US

- Lilang Wu
  - 4 years security experience
  - macOS/iOS malware/vulnerability
  - Fuzzing project
  - BH USA 2018, BH EU 2018, HITB, CodeBlue

About US

- Moony Li
  - 9 years security experience
  - macOS/iOS/Android malware/vulnerability
  - 0Day/NDay hunting/exploit
  - BH EU 2015, BH USA2018, BH EU 2018, BH ASIA 2018, HITB, CodeBlue



Agenda

- Kernel Debugger Overview
- The Introduction of LLDBFuzzer
- Attack Surfaces on Graphic Extensions
- Practice and Demo
- Vulnerabilities Found
- Implement a Debugger for Hackintosh
- Conclusion

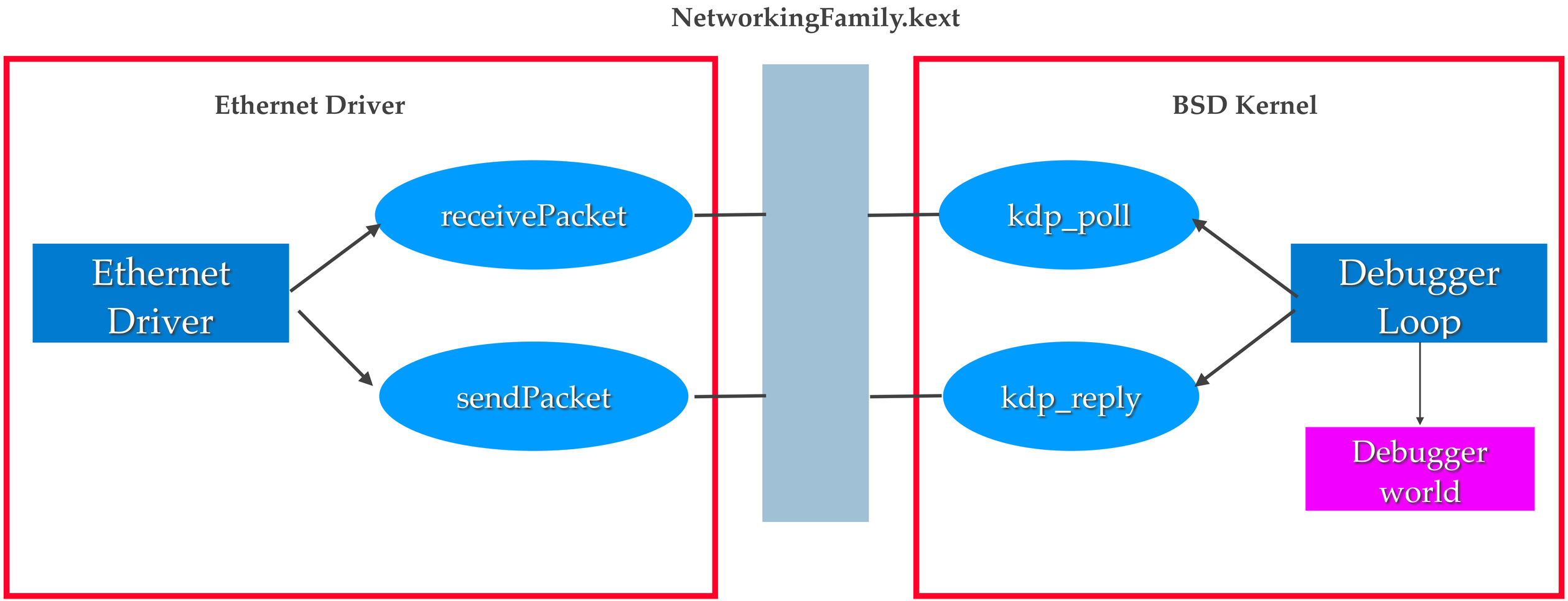
Agenda

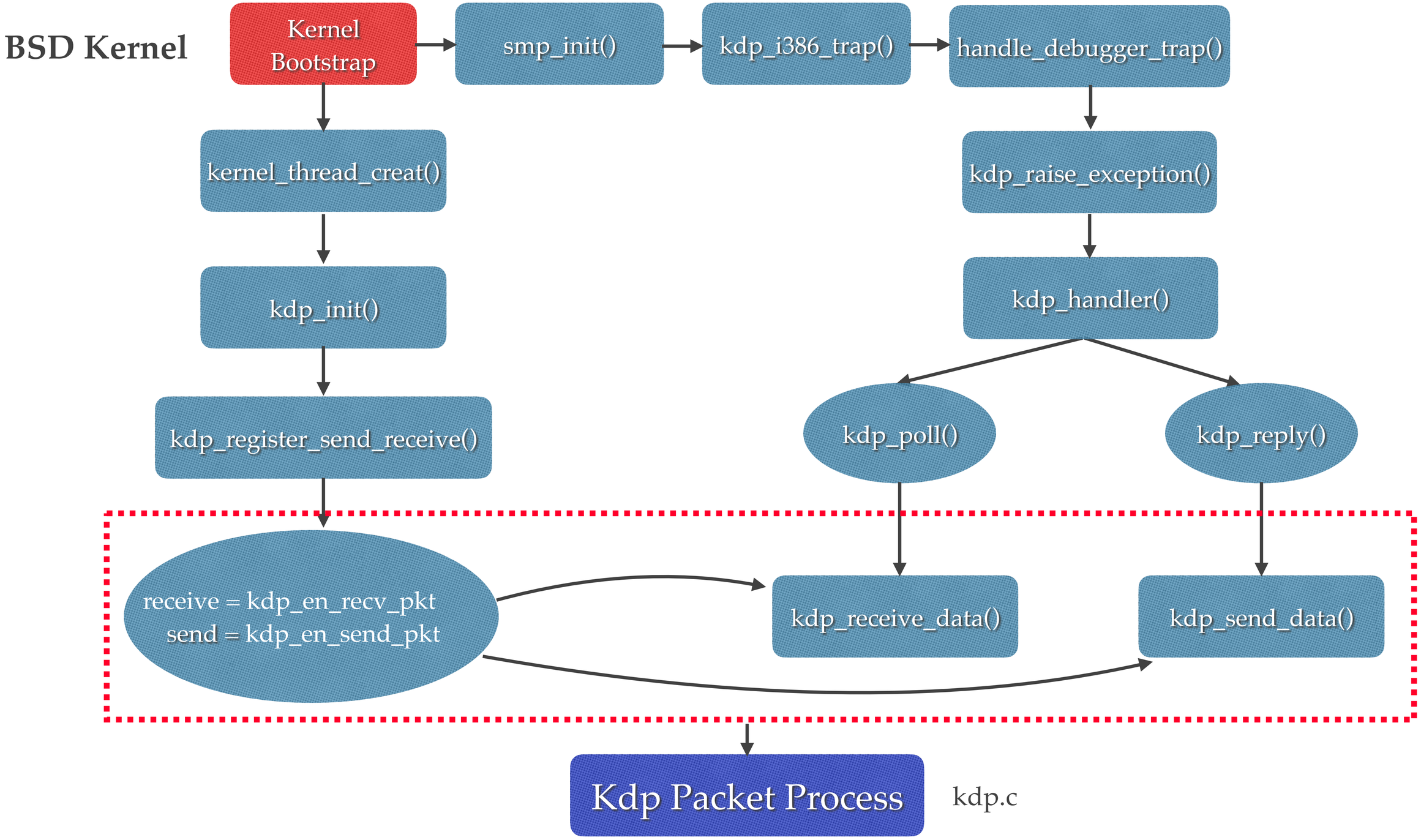
- Kernel Debugger Overview
- The Introduction of LLDBFuzzer
- Attack Surfaces on Graphic Extensions
- Practice and Demo
- Vulnerabilities Found
- Implement a Debugger for Hackintosh
- Conclusion

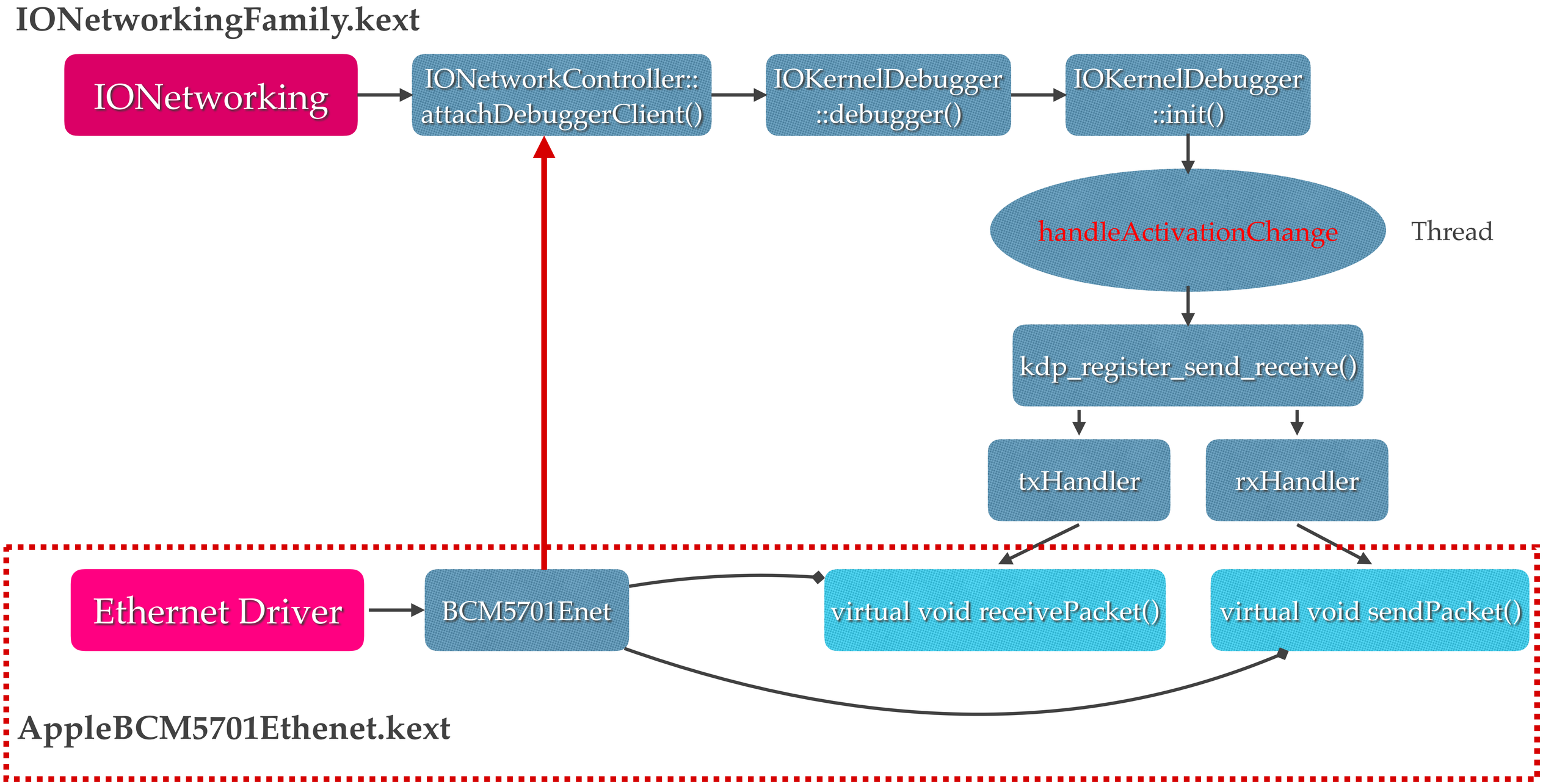
Kernel Debugger Overview

- KDP： a remote debugger protocol
- Active by change boot-args
- Only a single IOKernelDebugger can be activated at a given time
- Ethernet Debugging, Firewire Debugging, Serial Debugging

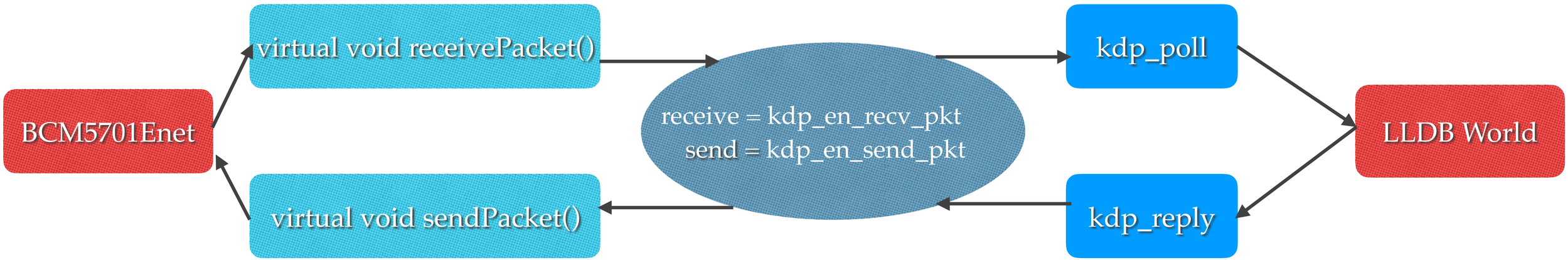
Kernel Debugging Overview




BSD Kernel
Kernel Bootstrap
smp_init()
kdp_i386_trap()
handle_debugger_trap()
kernel_thread_creat()
kdp_raise_exception()
kdp_init()
kdp_handler()
kdp_register_send_receive()
kdp_poll()
kdp_reply()
receive = kdp_en_recv_pkt
send = kdp_en_send_pkt
kdp_receive_data()
kdp_send_data()
Kdp Packet Process
kdp.c


IONetworkingFamily.kext
IONetworking
IONetworkController::
attachDebuggerClient()
IOKernelDebugger
::debugger()
IOKernelDebugger
::init()
handleActivationChange
Thread
kdp_register_send_receive()
txHandler
rxHandler
Ethernet Driver
BCM5701Enet
virtual void receivePacket()
virtual void sendPacket()
AppleBCM5701Ethenet.kext

Debugging UDP Packet Follow



Debugger ToolSet for macOS

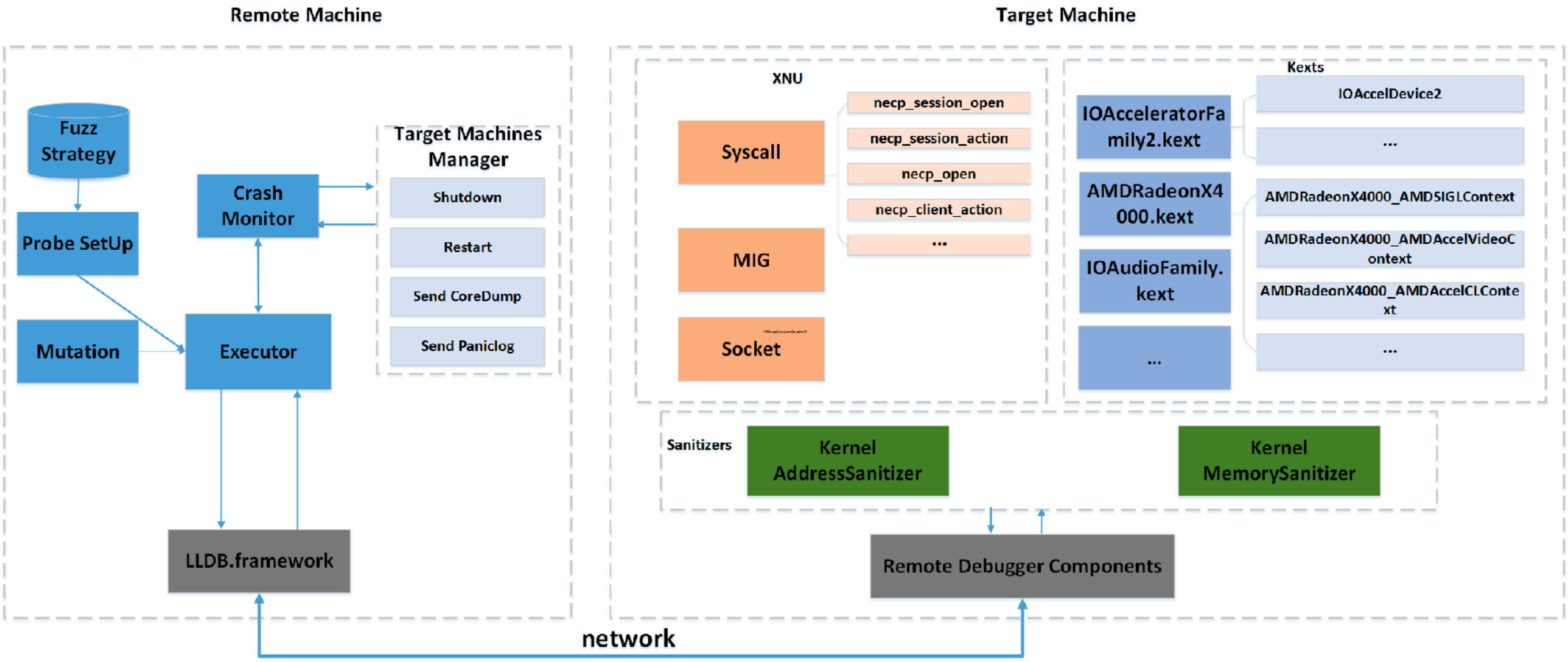
- API Wrappers for basic lldb Scripting Bridge APIs
- Performance reports plugin
- Main logic to load plugin and debug commands
- Debug commands implementation

```
xnu/
 |-tools/
    |-lldbmacros/
       |-core/             # Core logic about kernel, lldb value abstraction, configs etc.
       |-plugins/          # Holds plugins for kernel commands.
       |-xnu.py            # xnu debug framework along with kgmhelp, xnudebug commands.
       |-xnudefines.py
       |-utils.py
       |-process.py        # files containing commands/summaries code for each subsystem
       |-...
```

Agenda

- Kernel Debugger Overview
- The Introduction of LLDBFuzzer
- Attack Surfaces on Graphic Extensions
- Practice and Demo
- Vulnerabilities Found
- Implement a Debugger for Hackintosh
- Conclusion

LLDBFuzzer OverView



Probe Setup

- Actually, make breakpoints
  - By address/function name
  - Ignore times
  - Condition bp not works
- Controlled by FuzzSwitch

```
# set probe at is_io_connect_method
if FuzzSwitch.FUZZ_IO_CONNECT_METHOD:
    bp_iscm = target.BreakpointCreateByName("is_io_connect_method")
    bp_iscm.SetIgnoreCount(1)

    # config the mutation engine
    MutationConfig.FLIP_N_RAND_LIMIT = 100
    MutationConfig.FLIP_N_RAND_MIN = 10
    MutationConfig.FLIP_N_RAND_MAX = 25
    MutationConfig.FLIP_MIN_BYTES = 2
    MutationConfig.FILP_MAX_BYTES = 10
    FuzzMode.debug_mode = True

    #bp_iscm_1 = bp_iscm.GetLocationAtIndex(0)
    #bp_iscm_1.SetCondition('val == 3')

# AMD
if FuzzSwitch.FUZZ_SIDEBAND_BUFFER:
    bp_amd_type_2_selector_2 = target.BreakpointCreateByName("IOAccelContext2::processSidebandBuffer")
    bp_amd_type_2_selector_2.SetIgnoreCount(0)

    # config the mutation engine
    MutationConfig.FLIP_N_RAND_LIMIT = 100
    MutationConfig.FLIP_N_RAND_MIN = 10
    MutationConfig.FLIP_N_RAND_MAX = 25
    MutationConfig.FLIP_MIN_BYTES = 1
    MutationConfig.FILP_MAX_BYTES = 1
    FuzzMode.debug_mode = True

# set probe at unix_syscall64 -> memcpy address
if FuzzSwitch.FUZZ_SYS_CALL:
    unix_syscall64_memcpy_offset = 0x75852f
    bp_unix_sc_memcpy = target.BreakpointCreateByAddress(
        kernel_load_addr + unix_syscall64_memcpy_offset)  # 0xffffff801e75852f
    bp_unix_sc_memcpy.SetIgnoreCount(5)
    logger(str(bp_unix_sc_memcpy.GetHitCount()), 1)

    # config the mutation engine
    MutationConfig.FLIP_N_RAND_LIMIT = 100
    MutationConfig.FLIP_N_RAND_MIN = 10
    MutationConfig.FLIP_N_RAND_MAX = 25
    MutationConfig.FLIP_MIN_BYTES = 2
    MutationConfig.FILP_MAX_BYTES = 10
```

Fuzz Executor

- Intercept the fuzz probe and capture the input data buffer;
- Read the input data buffer, mutate it and write them to kernel memory ;
- Continue the interface, check the return value and monitor the fuzzing status ;
- If crash, send the core dump and panic log to fuzz server and restart the target machine

Fuzz Mutation

- FUZZ_FREQUENCY
- FLIP_N_RAND_LIMIT
- FLIP_N_RAND_MIN
- FLIP_NRAND_MAX
- FLIP_MIN_BYTES
- FLIP_MAX_BYTES
- Add you own mutation strategy

```
def flip_byte(data, datalen):
    offset = random.randint(0, 10000) % datalen
    #print "data[%d] = %s" % (offset, data[offset])
    value = '{:0>2x}'.format(random.randint(0, 0xffffffffffffffff) % 0xff)
    value_original = data[offset]
    data[offset] = value
    if FuzzMode.debug_mode:
        logger("original data[%d] = %s, now data[%d] = %s" % (offset, value_original, offset, value), 1)


def flip_n_byte(data, data_len):
    if not (data and data_len):
        return -1

    if FuzzMode.debug_mode:
        logger("FLIP_N_RAND_LIMIT=%d, FLIP_N_RAND_MIN=%d, FLIP_N_RAND_MAX=%d, FLIP_MIN_BYTES=%d, FILP_MAX_BYTES=%d" % (MutationConfig.FLIP_N_RAND_LIMIT,
                                                                                                                         MutationConfig.FLIP_N_RAND_MIN,
                                                                                                                         MutationConfig.FLIP_N_RAND_MAX,
                                                                                                                         MutationConfig.FLIP_MIN_BYTES,
                                                                                                                         MutationConfig.FILP_MAX_BYTES), 1)
    try_fuzz_frequency = rand_rate(MutationConfig.FLIP_N_RAND_LIMIT, MutationConfig.FLIP_N_RAND_MIN, MutationConfig.FLIP_N_RAND_MAX)
    try_fuzz_bytes   = data_len * try_fuzz_frequency
    u_max_legal      = get_min(data_len, MutationConfig.FILP_MAX_BYTES)
    u_min_legal      = get_min(data_len, MutationConfig.FLIP_MIN_BYTES)
    real_fuzz_bytes = u_min_legal

    if try_fuzz_bytes > u_max_legal:
        real_fuzz_bytes = u_max_legal

    if try_fuzz_bytes < u_min_legal:
        real_fuzz_bytes = u_min_legal

    if u_min_legal <= try_fuzz_bytes <= u_max_legal:
        real_fuzz_bytes = try_fuzz_bytes

    if FuzzMode.debug_mode:
        logger("Total %d bytes, try to fuzz %f percent, try to fuzz %d bytes, real fuzz bytes %d" % (data_len, try_fuzz_frequency,
                                                                                                     try_fuzz_bytes,
                                                                                                     real_fuzz_bytes), 1)
    for i in range(int(real_fuzz_bytes)):
        flip_byte(data, data_len)


def rand_rate(u_rand_limit, u_rand_min, u_rand_max):
    if u_rand_max <= u_rand_min:
        return 0
    else:
        u_temp = random.randint(0, 10000) % (u_rand_max - u_rand_min)
        return (u_rand_min + u_temp)/float(u_rand_limit)
```

Fuzz Mutation

```
@lldb_command("setMaxPercent")
def setFuzzMaxPercentCmd(cmd_args=None):
    """
    set fuzz max percent
    Args:
        cmd_args:

    Returns:

    """
    if cmd_args == None or len(
        print setFuzzMaxPercent
        return False

    max_percent = int(cmd_args
    if max_percent > MutationCo
        logger("min percent sho
        return False

    if max_percent < MutationCo
        logger("min percent sho
        return False

    MutationConfig.FLIP_N_RAND_
```

```
@lldb_command("setMinPercent")
def setFuzzMinPercentCmd(cmd_args=None):
    """
    set fuzz min percent
    Args:
        cmd_args:

    Returns:

    """
    if cmd_args == None or
        print setFuzzMinPer
        return False

    min_percent = int(cmd_a
    if min_percent > Mutati
        logger("min percent
        return False

    MutationConfig.FLIP_N_R
    return True
```

```
@lldb_command("setFlipMaxBytes")
def setFuzzFlipMaxBytesCmd(cmd_args=None):
    """
    set fuzz max bytes
    Args:
        cmd_args:

    Returns:

    """
    if cmd_args == None o
        print setFuzzFlip
        return False

    max_bytes = int(cmd_a
    if max_bytes < Mutati
        logger("min bytes
        return False

    MutationConfig.FILP_M
    return True
```

```
@lldb_command("setFlipMinBytes")
def setFuzzMinPercentCmd(cmd_args=None):
    """
    set fuzz min bytes
    Args:
        cmd_args:

    Returns:

    """
    if cmd_args == None or len(cmd_args) != 1:
        print setFuzzMaxPercentCmd.__doc__
        return False

    min_bytes = int(cmd_args[0])
    if min_bytes > MutationConfig.FILP_MAX_BYTES:
        logger("min percent should lower than %d !" % MutationConfig.FILP_MAX_BYTES, 2)
        return False

    MutationConfig.FLIP_MIN_BYTES = min_bytes
    return True
```

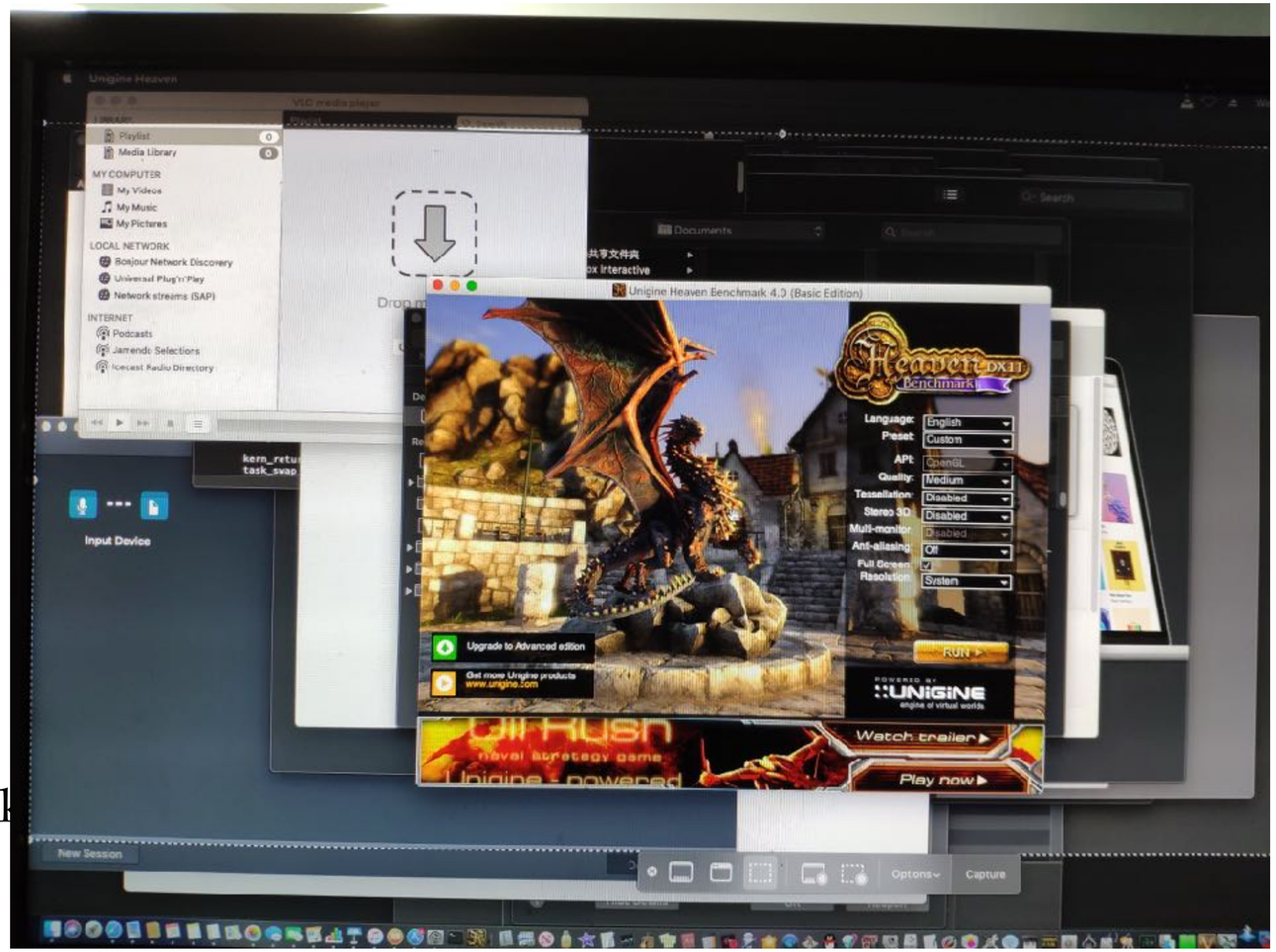
Crash Monitor

- Embedded lldb toolset is powe
- Fruitful python plugin
-

```
#
def diagnoseOnce(targetIP):
    settingSymbolResult = settingSymbols()
    remoteResult = kdpRemote(targetIP)
    if not checkKdpRemoteConnected():
        return

    btResult = getCallStack()
    conAddr = getConnectionAddrFromBt(btResult)
    showObjResult = showObject(conAddr)
    objName = getObjNameFromShowObj(showObjResult)
    selector = getSelectorFromBt(btResult)
    disResult = getDisAssemble()
    regsResult = getRegs()

    kdpRemote(targetIP)
    coreResult = sendCore(dumpServerIP, coreName)
    doDetach()
    monitorFileUntilDone(coreFilePath)

    print("diagnoseOnce exit")

def startRFCLoopInterface(debugger, targetIP, result, internal_dict):
    counter = 0
    while True:
        try:
            diagnoseOnce()
        except Exception as e:
            print (colored("[ERROR] " + str(e), "red"))
            traceback.print_exc()
        counter = counter + 1
        #break

# And the initialization code to add your commands
def __lldb_init_module(debugger, internal_dict):
    debugger.HandleCommand('command script add rfc -f remoteFuzzController.startRFCLoopInterface')
```

Data Generating on Target Machine

- Purpose
  - Touch more deep code coverage
- Methodology
  - Generating valid data and code execution (with context)
- Implementation
  - Apple Script based app test
  - Enrich kinds of corpuses around attack interface
  - Browser, 3D game engine, benchmark …as you think
-



Data Generating Tricks

- Embedded Apple Script is powerful
- Automatic app test
- Timely reboot from kernel
  - In case of kernel hang but not crash

```
void doReboot()
{

    fnPEHaltRestart  afnPEHaltRestart = NULL;
    fnhalt_all_cpus afnhalt_all_cpus = NULL;
    afnPEHaltRestart = (fnPEHaltRestart) solve_kernel_symbol(&g_kernel_info,"PEHaltRestart");
    afnPEHaltRestart(kPERestartCPU);
    afnhalt_all_cpus = (fnhalt_all_cpus)solve_kernel_symbol(&g_kernel_info,"halt_all_cpus");
    afnhalt_all_cpus(TRUE);
}
void watchdogTimelyRebootThread(__unused void *arg, __unused wait_result_t wr)
{
    unsigned int nCountSeconds = (unsigned int)arg;
    struct timespec ts = { nCountSeconds, 0 };
    int error = 0;
    lck_mtx_lock(watch_dogt_timely_reboot_mutex);
    while (1) {

        fnIOSleep aIOSleep = (fnIOSleep)solve_kernel_symbol(&g_kernel_info, API_SYMBOL_IO_SLEEP);
        aIOSleep(nCountSeconds*1000);
        printf("[DEBUG] doReboot: end...\n");
        doReboot();
        //doCheck();
    }
}

kern_return_t startWatdogForTimelyReboot(unsigned int nCountSeconds)
{
    printf("[DEBUG] startWatdogForTimelyReboot (%d): begin...\n",nCountSeconds);
    kern_return_t kr =  KERN_SUCCESS;
    watch_dogt_timely_reboot_grp = lck_grp_alloc_init("startWatdogForTimelyReboot", LCK_GRP_ATTR_NULL);
    watch_dogt_timely_reboot_mutex = lck_mtx_alloc_init(watch_dogt_timely_reboot_grp, LCK_ATTR_NULL);
    kr = kernel_thread_start(watchdogTimelyRebootThread, (void *)nCountSeconds, &tWatchdogThreadHandle);
    printf("[DEBUG] startWatdogForTimelyReboot (%d): end...\n",nCountSeconds);
    return kr;
```

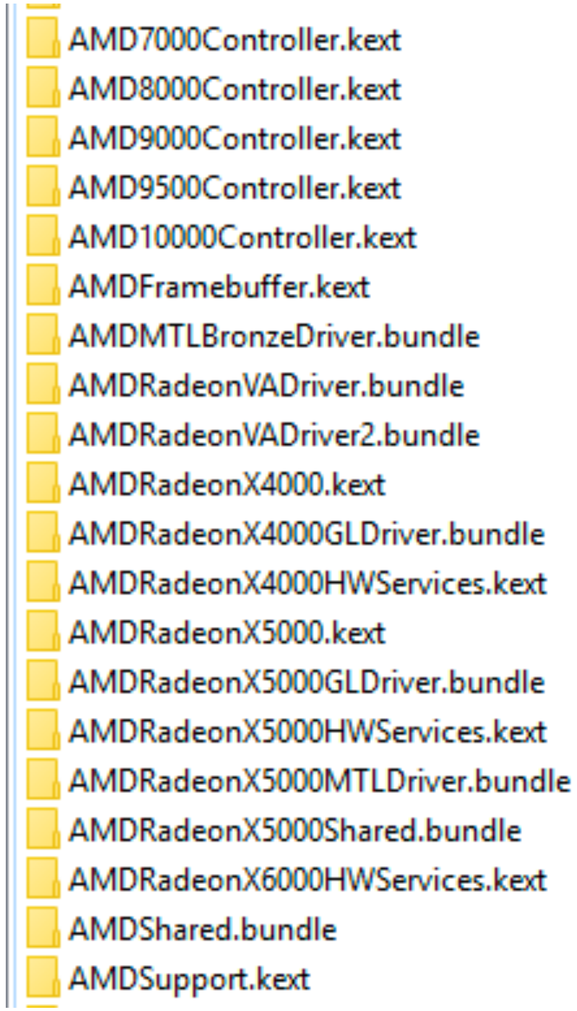
```
StartupFuzzingSource()
{
    echo "[StartupFuzzingSource]: begin..."
    startTimelyRebootDriver $sleepSecondsOfSession

    for counter in $(seq 1 999999)
    do
        #Test vm
        openAnyFolder "/Users/user/Desktop/VM/CDOS_VB/CDOS*/*.vbox"
        openAnyFolder "/Users/user/Desktop/VM/CDOS_VM/CDOS*/*.vmx"
        openAnyFolder "/Users/user/Desktop/VM/osx*/*.vmx"

        #External device test zone
        faceTimeTest
        VLCTest

        #Safari browser test zone
        safariTest

        #Game engine test zone
        heavenBenchMarkTest
        valleyBenchMarkTest
        heavenBenchMarkTest
        #Misc test zone
        openAnyFolder "/Applications/*"
        openAnyFolder "/Users/user/Downloads/*"
        openAnyFolder "/Applications/Utilities/*"
        sleep $sleepSecondsInFuzz

        if [ "$counter" -ge "$counterToReboot" ]
        then
            echo "now reboot"
            rebootNow
        fi
    done
    echo "[StartupFuzzingSource]: end..."
}
```

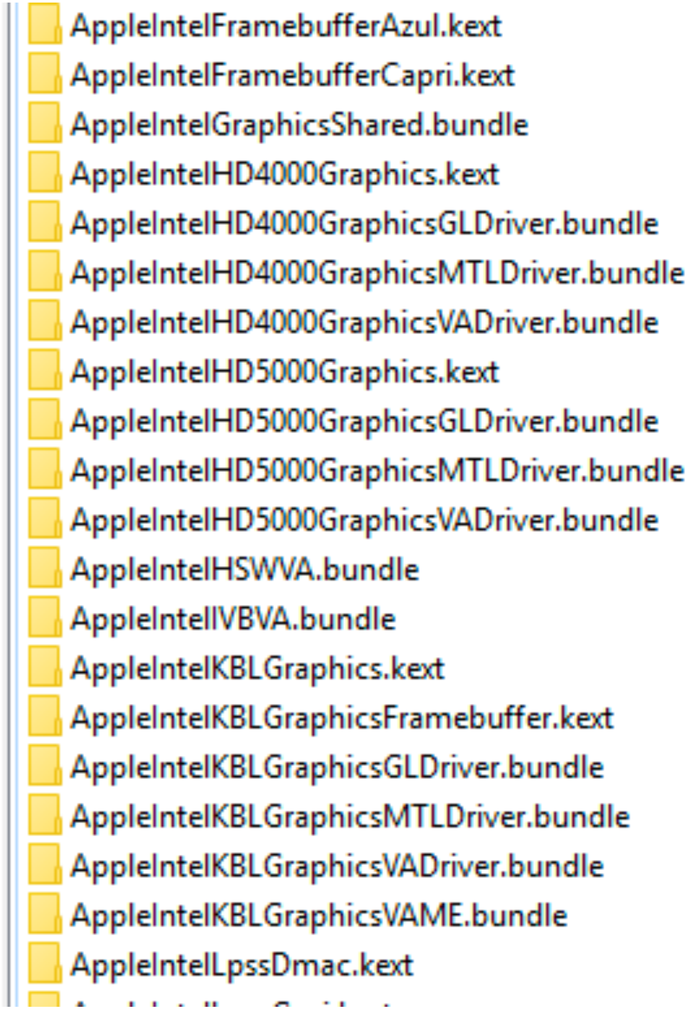
```
commonTestApp()
{
    appName=$1
    #appName="FaceTime"
    sudo killall -9 $appName
    osascript -e 'on run {appNameArg}' -e "quit app appNameArg" -e 'end run' $appName

    osascript -e 'on run {appNameArg}' -e "tell application  appNameArg to activate" -e 'end run' $appName
    echo "[commonTestApp]:\tfuzz FaceTime done..."
}
```

Agenda

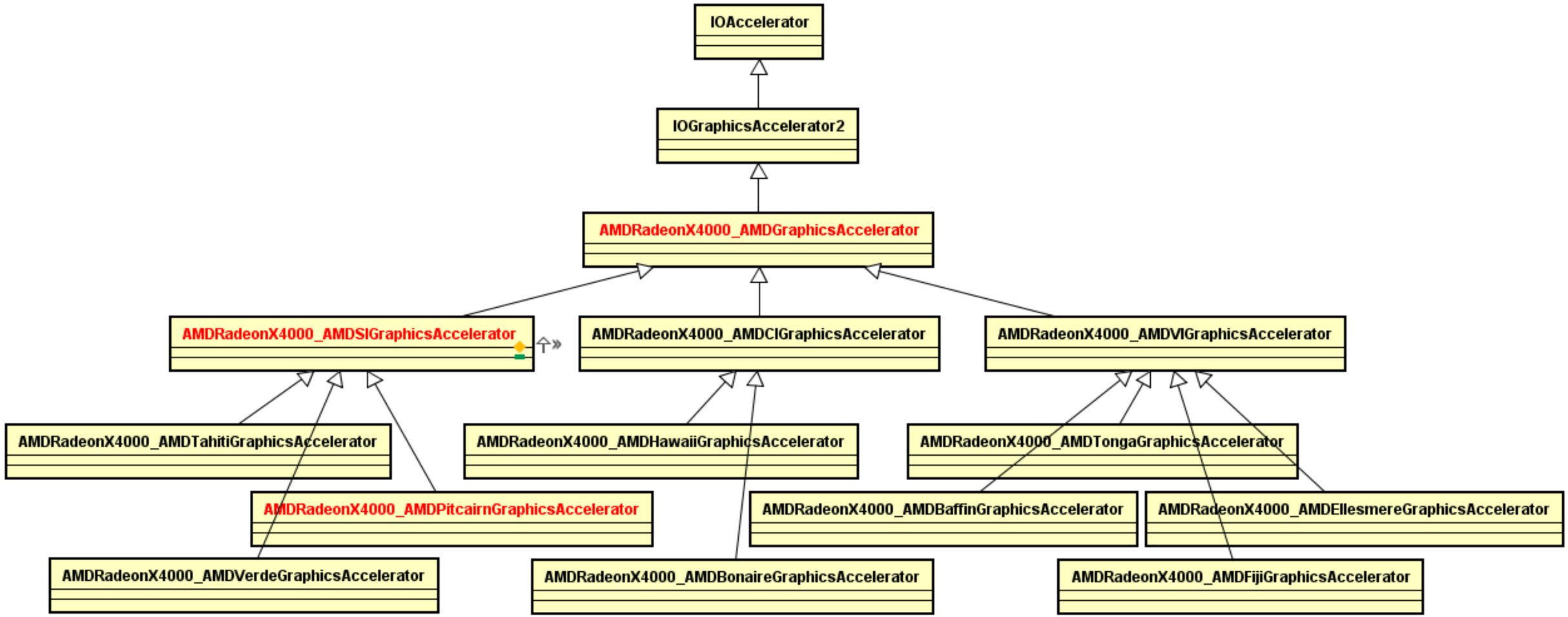
- Kernel Debugger Overview
- The Introduction of LLDBFuzzer
- Attack Surfaces on Graphic Extensions
- Practice and Demo
- Vulnerabilities Found
- Implement a Debugger for Hackintosh
- Conclusion

- Graphic Drivers
  - AMD
  - Intel





IOAccelerator Class Diagram for AMD Driver



Usual Attack Surfaces

- AMDRadeonX4000_AMDPitcairnGraphicsAccelerator
  - List all connection types and their services;
    - newUserClient
  - Get all the external methods of each services;
    - IOUserClient::externalMethod()
    - IOUserClient::getTargetAndMethodForIndex()
    - …
  - Refer to:
    - https://conference.hitb.org/hitbsecconf2019ams/sessions/fresh-apples-researching-new-attack-interfaces-on-ios-and-osx/#

Hidden Attack Surfaces

- Interfaces which are protected by filter drivers for Intel Graphic Driver, which has been introduced in YU WANG's DEFCON 26 topic;
- Interfaces whose arguments are controlled by the share memory;
- Interfaces which can not be indirectly touched by user space processes, but can be access by Safari etc. special processes

Hidden Attack Surfaces

- Interfaces which are protected by filter drivers for Intel Graphic Driver, which has introduced in YU WANG's DEFCON 26 topic;
- Interfaces whose arguments are controlled by the share memory;
- Interfaces which can not be indirectly touched by user space processes, but can be access by Safari etc. special processes

```
kern_return_t IOConnectMapMemory64(io_connect_t connect, uint32_t
memoryType, task_port_t intoTask, mach_vm_address_t *atAddress,
mach_vm_size_t *ofSize, IOOptionBits options);
```

```
/* Routine io_connect_map_memory_into_task */
kern_return_t is_io_connect_map_memory_into_task
(
    io_connect_t connection,
    uint32_t memory_type,
    task_t into_task,
    mach_vm_address_t *address,
    mach_vm_size_t *size,
    uint32_t flags
)
{
    IOReturn        err;
    IOMemoryMap *   map;

    CHECK( IOUserClient, connection, client );

    if (!into_task) return (kIOReturnBadArgument);

    IOStatisticsClientCall();
    map = client->mapClientMemory64( memory_type, into_task, flags, *address );

    if( map) {
        *address = map->getAddress();
        if( size)
            *size = map->getSize();
```

```
IOMemoryMap * IOUserClient::mapClientMemory64(
    IOOptionBits        type,
    task_t              task,
    IOOptionBits        mapFlags,
    mach_vm_address_t   atAddress )
{
    IOReturn            err;
    IOOptionBits        options = 0;
    IOMemoryDescriptor * memory = 0;
    IOMemoryMap *       map = 0;

    err = clientMemoryForType( (UInt32) type, &options, &memory );

    if( memory && (kIOReturnSuccess == err)) {

        FAKE_STACK_FRAME(getMetaClass());

        options = (options & ~kIOMapUserOptionsMask)
        | (mapFlags & kIOMapUserOptionsMask);
    map = memory->createMappingInTask( task, atAddress, options );
    memory->release();

        FAKE_STACK_FRAME_END();
    }

    return( map );
} « end mapClientMemory64 »
```

```
__int64 __cdecl IOAccelContext2::clientMemoryForType(IOAccelContext2 *this, unsigned int type, unsigned int *a3, IOMemoryDescriptor **memory)
{
  ---omitted code ---
  if ( type != 1 )
  {
    if ( type )
    {
      ---omitted code ---
      IOGraphicsAccelerator2::unlock_busy(v45);
      IOLockUnlock(v45->member17);
      return (unsigned int)v6;
    }
      ---omitted code ---
    LODWORD(bufferMemoryDescriptor) = (*(int (__fastcall **)(__int64, signed __int64, _QWORD, _QWORD))(*(_QWORD *)this->ioGraphicsAccelerator + 2336LL))(
                                        this->ioGraphicsAccelerator,
                                        65571LL,
                                        2 * v9,
                                        *(_QWORD *)page_size_0);// IOGraphicsAccelerator2::createBufferMemoryDescriptorWithOptions(uint,ulong,ulong)
    if ( bufferMemoryDescriptor )
    {
      bufferMemoryDescriptor_ptr = bufferMemoryDescriptor;
      v12 = this->bufferMemoryDesc;
      if ( v12 )
        (*(void (**)(void))(*(_QWORD *)v12 + 40LL))();
      v7->bufferMemoryDesc = bufferMemoryDescriptor_ptr;
      LODWORD(v7->member206) = 2 * v9;
      goto LABEL_49;
    }
    HIDWORD(this->member206) = v9 & 0x7FFFFFFF;
    bufferMemoryDescriptor_ptr = this->bufferMemoryDesc;
    if ( bufferMemoryDescriptor_ptr )
    {

      LODWORD(v44) = (*(int (__fastcall **)(__int64))(*(_QWORD *)bufferMemoryDescriptor_ptr + 736LL))(bufferMe
      v7->shareMem_start_vm_address_187 = v44;
      v7->shareMem_endt_vm_address_188 = v44 + (v7->member206 & 0xFFFFFFFC);

      (*(void (**)(void))(*(_QWORD *)v7->bufferMemoryDesc + 32LL))();
      *v51 = 0;
      *memory_ptr = (IOMemoryDescriptor *)v7->bufferMemoryDesc;
```

```
IOMemoryMap * IOUserClient::mapClientMemory64(
    IOOptionBits        type,
    task_t              task,
    IOOptionBits        mapFlags,
    mach_vm_address_t   atAddress )
{
    IOReturn            err;
    IOOptionBits        options = 0;
    IOMemoryDescriptor * memory = 0;
    IOMemoryMap *       map = 0;

    err = clientMemoryForType( (UInt32) type, &options, &memory );

    if( memory && (kIOReturnSuccess == err)) {

        FAKE_STACK_FRAME(getMetaClass());

        options = (options & ~kIOMapUserOptionsMask)
        | (mapFlags & kIOMapUserOptionsMask);
    map = memory->createMappingInTask( task, atAddress, options );
    memory->release();

        FAKE_STACK_FRAME_END();
    }

    return( map );
} « end mapClientMemory64 »
```

```
__int64 __cdecl IOAccelContext2::clientMemoryForType(IOAccelContext2 *this, unsigned int type, unsigned int *a3, IOMemoryDescriptor **memory)
{
  ---omitted code ---
  if ( type != 1 )
  {
    if ( type )
    {
       ---omitted code ---
      IOGraphicsAccelerator2::unlock_busy(v45);
      IOLockUnlock(v45->member17);
      return (unsigned int)v6;
    }
      ---omitted code ---
    LODWORD(bufferMemoryDescriptor) = (*(int (__fastcall **)(__int64, signed __int64, _QWORD, _QWORD))(*(_QWORD *)this->ioGraphicsAccelerator + 2336LL))(
                                        this->ioGraphicsAccelerator,
                                        65571LL,
                                        2 * v9,
                                        *(_QWORD *)page_size_0);// IOGraphicsAccelerator2::createBufferMemoryDescriptorWithOptions(uint,ulong,ulong)
    if ( bufferMemoryDescriptor )
    {
      bufferMemoryDescriptor_ptr = bufferMemoryDescriptor;
      v12 = this->bufferMemoryDesc;
      if ( v12 )
        (*(void (**)(void))(*(_QWORD *)v12 + 40LL))();
      v7->bufferMemoryDesc = bufferMemoryDescriptor_ptr;
      LODWORD(v7->member206) = 2 * v9;
      goto LABEL_49;
    }
    HIDWORD(this->member206) = v9 & 0x7FFFFFFF;
    bufferMemoryDescriptor_ptr = this->bufferMemoryDesc;
    if ( bufferMemoryDescriptor_ptr )
    {

      LODWORD(v44) = (*(int (__fastcall **)(__int64))(*(_QWORD *)bufferMemoryDescriptor_ptr + 736LL))(bufferMe
      v7->shareMem_start_vm_address_187 = v44;
      v7->shareMem_endt_vm_address_188 = v44 + (v7->member206 & 0xFFFFFFFC);

      (*(void (**)(void))(*(_QWORD *)v7->bufferMemoryDesc + 32LL))();
      *v51 = 0;
      *memory_ptr = (IOMemoryDescriptor *)v7->bufferMemoryDesc;
```

```
IOMemoryMap * IOUserClient::mapClientMemory64(
    IOOptionBits        type,
    task_t              task,
    IOOptionBits        mapFlags,
    mach_vm_address_t   atAddress )
{
    IOReturn            err;
    IOOptionBits        options = 0;
    IOMemoryDescriptor * memory = 0;
    IOMemoryMap *       map = 0;

    err = clientMemoryForType( (UInt32) type, &options, &memory );

    if( memory && (kIOReturnSuccess == err)) {

        FAKE_STACK_FRAME(getMetaClass());

        options = (options & ~kIOMapUserOptionsMask)
        | (mapFlags & kIOMapUserOptionsMask);
    map = memory->createMappingInTask( task, atAddress, options );
    memory->release();

        FAKE_STACK_FRAME_END();
    }

    return( map );
} « end mapClientMemory64 »
```

```
kern_return_t IOConnectMapMemory64(io_connect_t connect, uint32_t
memoryType, task_port_t intoTask, mach_vm_address_t *atAddress,
mach_vm_size_t *ofSize, IOOptionBits options);
```

```
/* Routine io_connect_map_memory_into_task */
kern_return_t is_io_connect_map_memory_into_task
(
    io_connect_t connection,
    uint32_t memory_type,
    task_t into_task,
    mach_vm_address_t *address,
    mach_vm_size_t *size,
    uint32_t flags
)
{
    IOReturn        err;
    IOMemoryMap *   map;

    CHECK( IOUserClient, connection, client );

    if (!into_task) return (kIOReturnBadArgument);

    IOStatisticsClientCall();
    map = client->mapClientMemory64( memory_type, into_task, flags, *address );

    if( map) {
        *address = map->getAddress();
        if( size)
            *size = map->getSize();
```

```
IOMemoryMap * IOUserClient::mapClientMemory64(
    IOOptionBits        type,
    task_t          task,
    IOOptionBits        mapFlags,
    mach_vm_address_t   atAddress )
{
    IOReturn        err;
    IOOptionBits    options = 0;
    IOMemoryDescriptor * memory = 0;
    IOMemoryMap *   map = 0;

    err = clientMemoryForType( (UInt32) type, &options, &memory );

    if( memory && (kIOReturnSuccess == err)) {

        FAKE_STACK_FRAME(getMetaClass());

        options = (options & ~kIOMapUserOptionsMask)
        | (mapFlags & kIOMapUserOptionsMask);
    map = memory->createMappingInTask( task, atAddress, options );
    memory->release();

        FAKE_STACK_FRAME_END();
    }

    return( map );
} « end mapClientMemory64 »
```

```
__int64 __cdecl IOAccelContext2::clientMemoryForType(IOAccelContext2 *this, unsigned int type, unsigned int *a3, IOMemoryDescriptor **memory)
{
  ---omitted code ---
  if ( type != 1 )
  {
    if ( type )
    {
       ---omitted code ---
      IOGraphicsAccelerator2::unlock_busy(v45);
      IOLockUnlock(v45->member17);
      return (unsigned int)v6;
    }
      ---omitted code ---
    LODWORD(bufferMemoryDescriptor) = (*(int (__fastcall **)(__int64, signed __int64, _QWORD, _QWORD))(*(_QWORD *)this->ioGraphicsAccelerator + 2336LL))(
                                        this->ioGraphicsAccelerator,
                                        65571LL,
                                        2 * v9,
                                        *(_QWORD *)page_size_0);// IOGraphicsAccelerator2::createBufferMemoryDescriptorWithOptions(uint,ulong,ulong)
    if ( bufferMemoryDescriptor )
    {
      bufferMemoryDescriptor_ptr = bufferMemoryDescriptor;
      v12 = this->bufferMemoryDesc;
      if ( v12 )
        (*(void (**)(void))(*(_QWORD *)v12 + 40LL))();
      v7->bufferMemoryDesc = bufferMemoryDescriptor_ptr;
      LODWORD(v7->member206) = 2 * v9;
      goto LABEL_49;
    }
    HIDWORD(this->member206) = v9 & 0x7FFFFFFF;
    bufferMemoryDescriptor_ptr = this->bufferMemoryDesc;
    if ( bufferMemoryDescriptor_ptr )
    {
      LODWORD(v44) = (*(int (__fastcall **)(__int64))(*(_QWORD *)bufferMemoryDescriptor_ptr + 736LL))(bufferMe
      v7->shareMem_start_vm_address_187 = v44;
      v7->shareMem_endt_vm_address_188 = v44 + (v7->member206 & 0xFFFFFFFC);

      (*(void (**)(void))(*(_QWORD *)v7->bufferMemoryDesc + 32LL))();
      *v51 = 0;
      *memory_ptr = (IOMemoryDescriptor *)v7->bufferMemoryDesc;
```

The virtual address of the beginning of the buffer

IOBufferMemoryDescriptor::getBytesNoCopy()

```
IOMemoryMap * IOUserClient::mapClientMemory64(
    IOOptionBits        type,
    task_t              task,
    IOOptionBits        mapFlags,
    mach_vm_address_t   atAddress )
{
    IOReturn            err;
    IOOptionBits        options = 0;
    IOMemoryDescriptor * memory = 0;
    IOMemoryMap *       map = 0;

    err = clientMemoryForType( (UInt32) type, &options, &memory );

    if( memory && (kIOReturnSuccess == err)) {

        FAKE_STACK_FRAME(getMetaClass());

        options = (options & ~kIOMapUserOptionsMask)
        | (mapFlags & kIOMapUserOptionsMask);
    map = memory->createMappingInTask( task, atAddress, options );
    memory->release();

        FAKE_STACK_FRAME_END();
    }

    return( map );
} « end mapClientMemory64 »
```

```
__int64 __cdecl IOAccelContext2::processSidebandBuffer(IOAccelContext2 *this, void *a2, bool a3)
{
  ---omitted code ---
  v4 = (char *)&this->comandStreamInfo;
  LODWORD(this->member199) = 0;
  this->comandStreamInfo = 0LL;
  this->member194 = 0LL;
  LODWORD(this->member195) = 0;
  v5 = this->shareMem_start_vm_address_187 + 16;
  this->comandStreamInfo_offset24 = v5;          //  // this->commandStreamInfo + 24
  LODWORD(this->member200) = 0;
  BYTE4(this->member200) = a3;
  while ( 1 )
  {
    v6 = this->shareMem_end_vm_address_188;
    if ( v6 < v5 + 8 )
    {
      v14 = _os_log_default_0;
      _os_log_internal(
        &dword_0,
        _os_log_default_0,
        17LL,
        IOAccelContext2::processSidebandBuffer(IOAccelCommandDescriptor *,bool)::_os_log_fmt,
        "virtual bool IOAccelContext2::processSidebandBuffer(IOAccelCommandDescriptor *, bool)");
      v15 = LOWORD(this->comandStreamInfo_offset32);
      v16 = WORD1(this->comandStreamInfo_offset32);
      _os_log_internal(
        &dword_0,
        v14,
        17LL,
        IOAccelContext2::setContextError(unsigned int)::_os_log_fmt,
        "void IOAccelContext2::setContextError(uint32_t)");
      goto LABEL_18;
    }
    LOWORD(this->comandStreamInfo_offset32) = *(_WORD *)v5;
    v7 = *(_WORD *)(v5 + 2);
    WORD1(this->comandStreamInfo_offset32) = v7;
    v8 = *(_DWORD *)(v5 + 4);
    HIDWORD(this->comandStreamInfo_offset32) = v8;
    this->member198 = v5 + 8;
```

Copy share memory data into cmdinfo

```
v12 = this->vtable;
if ( BYTE4(this->member200) )
  ((void (__fastcall *)(IOAccelContext2 *, char *))v12->__ZN15IOAccelContext220processSidebandTokenER24IOAccelCommandStreamInfo)(
    this,
    cmdinfo_ptr);
else
  ((void (__fastcall *)(IOAccelContext2 *, char *))v12->__ZN15IOAccelContext220discardSidebandTokenER24IOAccelCommandStreamInfo)(
    this,
    cmdinfo_ptr);
```

Token process methods

```
void __fastcall AMDRadeonX4000_AMDSIGLContext::processSidebandToken(IORegistryEntry *this, uintptr_t cmdi
{
  uintptr_t v2; // r12@1
  unsigned __int16 v3; // ax@1
  int v4; // ebx@3
  int v5; // eax@3
  signed __int64 v6; // rax@6
  unsigned __int8 v7; // cf@6
  _DWORD *v8; // rbx@7
  uintptr_t v9; // r15@8
  uintptr_t v10; // rax@8
  uintptr_t v11; // r15@10
  uintptr_t v12; // rax@10
  __int64 v13; // rax@12
  unsigned __int64 v14; // rax@13

  v2 = cmdinfo;
  v3 = *(_WORD *)(cmdinfo + 32);                // can be controlled by share memeory which offset is 16
  ... ... ...
  v4 = HIBYTE(v3) - 128;
  v5 = *(_DWORD *)((char *)tokenArgSizeVaries + (((unsigned int)v4 >> 3) & 0x1FFC));
  if ( _bittest(&v5, v4)
    || (unsigned __int8)IOAccelContext2::validateTokenSize(
                          this,
                          cmdinfo,
                          (unsigned __int16)(((unsigned int)tokenArgSizes[(unsigned __int16)v4] + 11)
  {
    if ( *(_DWORD *)(*(_QWORD *)(cmdinfo + 24) + 4LL) < *((_DWORD *)this + 1150) )// *((_DWORD *)this
    {
      v6 = 2LL * (unsigned __int16)v4;
      v7 = __CFADD__(AMDRadeonX4000_AMDSIGLContext::ati_token_process_methods[v6 + 1], this);
      JUMPOUT(__CS__, AMDRadeonX4000_AMDSIGLContext::ati_token_process_methods[v6]);
    }
    IOAccelContext2::setContextError(this, 0xFFFFFFFC);
  }
```

```
AMDRadeonX4000_AMDSIGLContext::ati_token_process_methods[]
                                        ; DATA XREF: AMDRadeonX4000_AMDSIGLContext::processSidebandToken(IOAccelComman
                                        ; AMDRadeonX4000_AMDSIGLContext::process_StateShadowInfo(IOAccelCommandStream
        align 10h
        dq offset __ZN29AMDRadeonX4000_AMDSIGLContext24process_InvalidateObjectER24IOAccelCommandStreamInfo ;
        align 20h
        dq offset __ZN29AMDRadeonX4000_AMDSIGLContext18handle_BindObjectsER24IOAccelCommandStreamInfo ; AMDRad
        align 10h
        dq offset __ZN29AMDRadeonX4000_AMDSIGLContext20handle_UnbindObjectsER24IOAccelCommandStreamInfo ; AMDR
        align 20h
        dq offset __ZN29AMDRadeonX4000_AMDSIGLContext23handle_UnusedDataBufferER24IOAccelCommandStreamInfo ; A
        align 10h
        dq offset __ZN29AMDRadeonX4000_AMDSIGLContext22process_UnhandledTokenER24IOAccelCommandStreamInfo ; AM
        align 20h
        dq offset __ZN29AMDRadeonX4000_AMDSIGLContext22process_UnhandledTokenER24IOAccelCommandStreamInfo ; AM
        align 10h
        dq offset __ZN29AMDRadeonX4000_AMDSIGLContext22process_UnhandledTokenER24IOAccelCommandStreamInfo ; AM
        align 20h
        dq offset __ZN29AMDRadeonX4000_AMDSIGLContext25process_PatchStreamTexBufER24IOAccelCommandStreamInfo ;
        align 10h
        dq offset __ZN29AMDRadeonX4000_AMDSIGLContext22process_UnhandledTokenER24IOAccelCommandStreamInfo ; AM
        align 20h
        dq offset __ZN29AMDRadeonX4000_AMDSIGLContext18process_DrawBufferER24IOAccelCommandStreamInfo ; AMDRad
        align 10h
        dq offset __ZN29AMDRadeonX4000_AMDSIGLContext22process_UnhandledTokenER24IOAccelCommandStreamInfo ; AM
        align 20h
        dq offset __ZN29AMDRadeonX4000_AMDSIGLContext22process_StretchTex2TexER24IOAccelCommandStreamInfo ; AM
        align 10h
        dq offset __ZN29AMDRadeonX4000_AMDSIGLContext22process_CopyColorScaleER24IOAccelCommandStreamInfo ; AM
        align 20h
        dq offset __ZN29AMDRadeonX4000_AMDSIGLContext17process_AAResolveER24IOAccelCommandStreamInfo ; AMDRade
        align 10h
        dq offset __ZN29AMDRadeonX4000_AMDSIGLContext23process_StretchSurf2TexER24IOAccelCommandStreamInfo ; A
        align 20h
        dq offset __ZN29AMDRadeonX4000_AMDSIGLContext25process_ClearDepthStencilER24IOAccelCommandStreamInfo ;
```

Influence methods in AMDRadeonX4000 kext

- AMDRadeonX4000_AMDSIGLContext::ati_token_process_methods
  - Total 33
- AMDRadeonX4000_AMDSIGLContext::ati_token_discard_methods
  - Total 33

Pattern

```
—- omitted code —-
IOConnectMapMemory64(…,*atAddress, …);
(dword)*atAddress = 0x41414141;
…
—- omitted code —-
```

```
—- omitted code —-
v1 = this->buffer;
v2= this->share_memory_data + offset;
memcpy(v1, v2, len);
call_func(v1)
—- omitted code —-
```

Fuzz Methodology

- Be familiar with the data structure in IOAccelCommandStreamInfo object
- Set the fuzz probe at the critical function, here is IOAccelContext2::processSidebandBuffer()
- Mutation the proper bytes in this object
- Continue the kernel process

Hidden Attack Surfaces

- Interfaces which are protected by filter drivers for Intel Graphic Driver, which has introduced in YU WANG's DEFCON 26 topic;
- Interfaces whose arguments are controlled by the share memory;
- Interfaces which can not be touched directly by user space processes, but can be access by Safari etc. special processes

Private Connection Type

- IOFramebuffer::newUserClient()

```
enum {
    // connection types for IOServiceOpen
    kIOFBServerConnectType                  = 0,
    kIOFBSharedConnectType                  = 1,
    kIOFBDiagnoseConnectType                = 2,
};
```

```
switch (type)
{
    case kIOFBServerConnectType:
        if (fServerConnect)
            err = kIOReturnExclusiveAccess;
        else
        {
            if ((this == gIOFBConsoleFramebuffer) || !gIOFBConsoleFramebuffer)
                setPlatformConsole(0, kPEReleaseScreen,
                                   DBG_IOG_SOURCE_NEWUSERCLIENT);

            err = open();
            if (kIOReturnSuccess == err)
                newConnect = IOFramebufferUserClient::withTask(owningTask);
            if (!newConnect)
                err = kIOReturnNoResources;
        }
        break;

    case kIOFBSharedConnectType:
        if (fSharedConnect)
        {
            DEBG(thisName, " existing shared\n");
            theConnect = fSharedConnect;
            theConnect->retain();
        }
        else if (fServerConnect)
        {
            DEBG(thisName, " new shared\n");
            newConnect = IOFramebufferSharedUserClient::withTask(owningTask);
            if (!newConnect)
                err = kIOReturnNoResources;
        }
        else
            err = kIOReturnNotOpen;
        break;
```

Hidden Functions

```
dq offset __ZN14AMDFramebuffer16getApertureRangeEi ; AMDFramebuffer::getApertureRange(int)
dq offset __ZN14AMDFramebuffer12getVRAMRangeEv ; AMDFramebuffer::getVRAMRange(void)
dq offset __ZN14AMDFramebuffer16enableControllerEv ; AMDFramebuffer::enableController(void)
dq offset __ZN14AMDFramebuffer15getPixelFormatsEv ; AMDFramebuffer::getPixelFormats(void)
dq offset __ZN14AMDFramebuffer19getDisplayModeCountEv ; AMDFramebuffer::getDisplayModeCount(void)
dq offset __ZN14AMDFramebuffer15getDisplayModesEPi ; AMDFramebuffer::getDisplayModes(int *)
dq offset __ZN14AMDFramebuffer28getInformationForDisplayModeEiP24IODisplayModeInformation ; AMDFramebuffer::getIn
dq offset __ZN14AMDFramebuffer29getPixelFormatsForDisplayModeEii ; AMDFramebuffer::getPixelFormatsForDisplayMode(
dq offset __ZN14AMDFramebuffer19getPixelInformationEiiiP18IOPixelInformation ; AMDFramebuffer::getPixelInformatio
dq offset __ZN14AMDFramebuffer21getCurrentDisplayModeEPiS0_ ; AMDFramebuffer::getCurrentDisplayMode(int *,int *)
dq offset __ZN14AMDFramebuffer14setDisplayModeEii ; AMDFramebuffer::setDisplayMode(int,int)
dq offset __ZN13IOFramebuffer17setApertureEnableEij ; IOFramebuffer::setApertureEnable(int,uint)
dq offset __ZN14AMDFramebuffer21setStartupDisplayModeEii ; AMDFramebuffer::setStartupDisplayMode(int,int)
dq offset __ZN14AMDFramebuffer21getStartupDisplayModeEPiS0_ ; AMDFramebuffer::getStartupDisplayMode(int *,int *)
dq offset __ZN14AMDFramebuffer18setCLUTWithEntriesEP12IOColorEntryjjj ; AMDFramebuffer::setCLUTWithEntries(IOColo
dq offset __ZN14AMDFramebuffer13setGammaTableEjjjPv ; AMDFramebuffer::setGammaTable(uint,uint,uint,void *)
dq offset __ZN14AMDFramebuffer12setAttributeEjm ; AMDFramebuffer::setAttribute(uint,ulong)
dq offset __ZN14AMDFramebuffer12getAttributeEjPm ; AMDFramebuffer::getAttribute(uint,ulong *)
dq offset __ZN14AMDFramebuffer27getTimingInfoForDisplayModeEiP19IOTimingInformation ; AMDFramebuffer::getTimingIn
dq offset __ZN14AMDFramebuffer22validateDetailedTimingEPvy ; AMDFramebuffer::validateDetailedTiming(void *,ulong
dq offset __ZN14AMDFramebuffer18setDetailedTimingsEP7OSArray ; AMDFramebuffer::setDetailedTimings(OSArray *)
dq offset __ZN13IOFramebuffer18getConnectionCountEv ; IOFramebuffer::getConnectionCount(void)
dq offset __ZN14AMDFramebuffer25setAttributeForConnectionEijm ; AMDFramebuffer::setAttributeForConnection(int,uin
dq offset __ZN14AMDFramebuffer25getAttributeForConnectionEijPm ; AMDFramebuffer::getAttributeForConnection(int,ui
```

Fuzz Methodology

- Similar with the passive fuzz for the is_io_connect_method()
  - Set the fuzz probe in the function is_io_connect_method
  - Mutation the proper bytes in its scalar input and inband input
  - Continue the kernel process
- Similar with the processSidebandBuffer function fuzz
  - Research the arguments structure first and fuzz each function separately

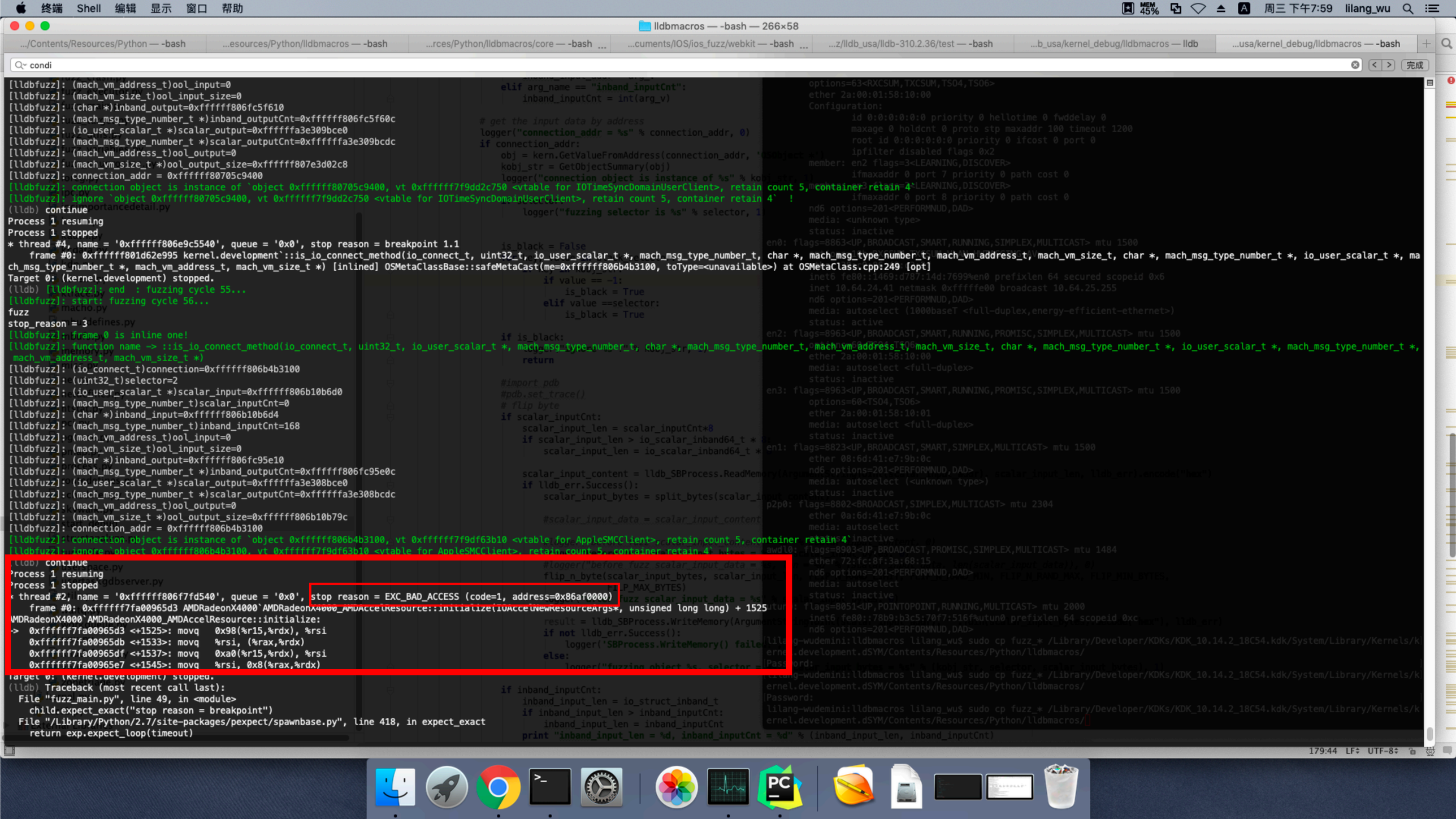
Agenda

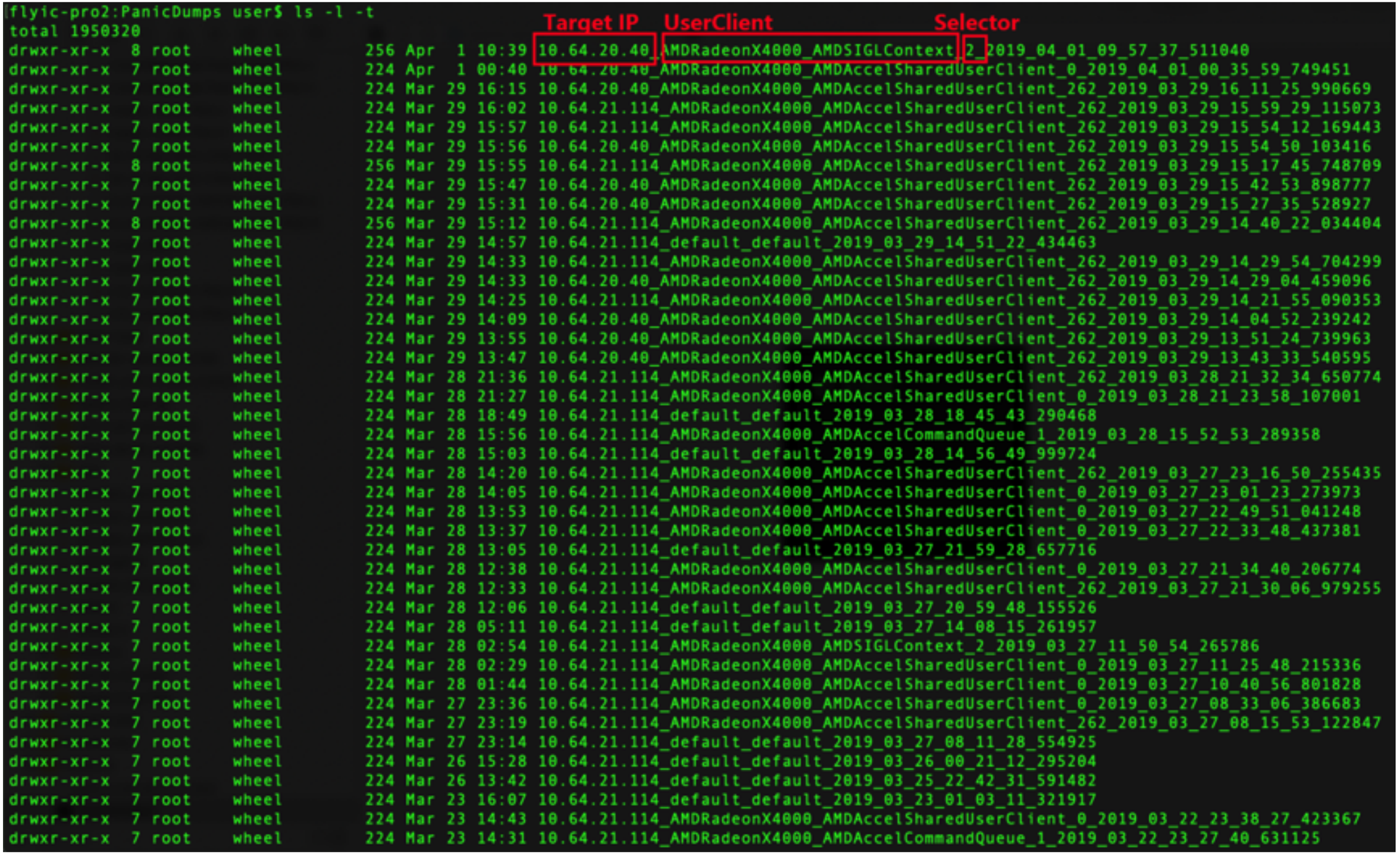
- Kernel Debugger Overview
- The Introduction of LLDBFuzzer
- Attack Surfaces on Graphic Extensions
- Practice and Demo
- Vulnerabilities Found
- Implement a Debugger for Hackintosh
- Conclusion

终端 Shell 编辑 显示 窗口 帮助 周六 下午12:04 lilang_wu

lldbmacros — lldb — 270×63

```
lilang-wudemini:lldbmacros lilang_wu$ lldb
(lldb)
Password:
lilang-wudemini:lldbmacros lilang_wu$ sudo cp fuzz_controller.py /Library/Developer/KDKs/KDK_10.14.5_18F132.kdk/System/Library/Kernels/kernel.development.dSYM/Contents/Resources/Python/lldbmacros/
Password:
lilang-wudemini:lldbmacros lilang_wu$ sudo cp fuzz_controller.py /Library/Developer/KDKs/KDK_10.14.5_18F132.kdk/System/Library/Kernels/kernel.development.dSYM/Contents/Resources/Python/lldbmacros/
lilang-wudemini:lldbmacros lilang_wu$ sudo cp fuzz_controller.py /Library/Developer/KDKs/KDK_10.14.5_18F132.kdk/System/Library/Kernels/kernel.development.dSYM/Contents/Resources/Python/lldbmacros/
Password:
lilang-wudemini:lldbmacros lilang_wu$ sudo cp fuzz_controller.py /Library/Developer/KDKs/KDK_10.14.5_18F132.kdk/System/Library/Kernels/kernel.development.dSYM/Contents/Resources/Python/lldbmacros/
lilang-wudemini:lldbmacros lilang_wu$ sudo cp fuzz_controller.py /Library/Developer/KDKs/KDK_10.14.5_18F132.kdk/System/Library/Kernels/kernel.development.dSYM/Contents/Resources/Python/lldbmacros/
lilang-wudemini:lldbmacros lilang_wu$ sudo cp fuzz_controller.py /Library/Developer/KDKs/KDK_10.14.5_18F132.kdk/System/Library/Kernels/kernel.development.dSYM/Contents/Resources/Python/lldbmacros/
lilang-wudemini:lldbmacros lilang_wu$ sudo cp fuzz_controller.py /Library/Developer/KDKs/KDK_10.14.5_18F132.kdk/System/Library/Kernels/kernel.development.dSYM/Contents/Resources/Python/lldbmacros/
lilang-wudemini:lldbmacros lilang_wu$ sudo cp fuzz_controller.py /Library/Developer/KDKs/KDK_10.14.5_18F132.kdk/System/Library/Kernels/kernel.development.dSYM/Contents/Resources/Python/lldbmacros/
lilang-wudemini:lldbmacros lilang_wu$ sudo cp fuzz_controller.py /System/Library/Kernels/kernel.development.dSYM/Contents/Resources/Python/lldbmacros/
lilang-wudemini:lldbmacros lilang_wu$ sudo cp fuzz_controller.py /System/Library/Kernels/kernel.development.dSYM/Contents/Resources/Python/lldbmacros/
lilang-wudemini:lldbmacros lilang_wu$ sudo cp fuzz_controller.py /Library/Developer/KDKs/KDK_10.14.5_18F132.kdk/System/Library/Kernels/kernel.development.dSYM/Contents/Resources/Python/lldbmacros/
lilang-wudemini:lldbmacros lilang_wu$ sudo cp fuzz_controller.py /Library/Developer/KDKs/KDK_10.14.5_18F132.kdk/System/Library/Kernels/kernel.development.dSYM/Contents/Resources/Python/lldbmacros/
lilang-wudemini:lldbmacros lilang_wu$ sudo cp fuzz_controller.py /System/Library/Kernels/kernel.development.dSYM/Contents/Resources/Python/lldbmacros/
lilang-wudemini:lldbmacros lilang_wu$ sudo cp fuzz_controller.py /System/Library/Kernels/kernel.development.dSYM/Contents/Resources/Python/lldbmacros/
lilang-wudemini:lldbmacros lilang_wu$ sudo cp fuzz_controller.py /Library/Developer/KDKs/KDK_10.14.5_18F132.kdk/System/Library/Kernels/kernel.development.dSYM/Contents/Resources/Python/lldbmacros/
lilang-wudemini:lldbmacros lilang_wu$ sudo cp fuzz_controller.py /Library/Developer/KDKs/KDK_10.14.5_18F132.kdk/System/Library/Kernels/kernel.development.dSYM/Contents/Resources/Python/lldbmacros/
lilang-wudemini:lldbmacros lilang_wu$ sudo cp fuzz_controller.py /System/Library/Kernels/kernel.development.dSYM/Contents/Resources/Python/lldbmacros/
lilang-wudemini:lldbmacros lilang_wu$ sudo cp fuzz_controller.py /System/Library/Kernels/kernel.development.dSYM/Contents/Resources/Python/lldbmacros/
lilang-wudemini:lldbmacros lilang_wu$ sudo cp fuzz_controller.py /Library/Developer/KDKs/KDK_10.14.5_18F132.kdk/System/Library/Kernels/kernel.development.dSYM/Contents/Resources/Python/lldbmacros/
lilang-wudemini:lldbmacros lilang_wu$ sudo cp fuzz_controller.py /Library/Developer/KDKs/KDK_10.14.5_18F132.kdk/System/Library/Kernels/kernel.development.dSYM/Contents/Resources/Python/lldbmacros/
Password:
Sorry, try again.
Password:
lilang-wudemini:lldbmacros lilang_wu$ sudo cp fuzz_controller.py /System/Library/Kernels/kernel.development.dSYM/Contents/Resources/Python/lldbmacros/
lilang-wudemini:lldbmacros lilang_wu$ sudo cp fuzz_* /System/Library/Kernels/kernel.development.dSYM/Contents/Resources/Python/lldbmacros/
Password:
lilang-wudemini:lldbmacros lilang_wu$ sudo cp fuzz_* /Library/Developer/KDKs/KDK_10.14.5_18F132.kdk/System/Library/Kernels/kernel.development.dSYM/Contents/Resources/Python/lldbmacros/
lilang-wudemini:lldbmacros lilang_wu$ sudo cp fuzz_* /Library/Developer/KDKs/KDK_10.14.5_18F132.kdk/System/Library/Kernels/kernel.development.dSYM/Contents/Resources/Python/lldbmacros/
Password:
lilang-wudemini:lldbmacros lilang_wu$ sudo cp fuzz_controller.py /System/Library/Kernels/kernel.development.dSYM/Contents/Resources/Python/lldbmacros/
lilang-wudemini:lldbmacros lilang_wu$ pwd
/Users/lilang_wu/Documents/IOS/ios_fuzz/conference/lldb_usa/lldbfuzzer-demo/lldbmacros
lilang-wudemini:lldbmacros lilang_wu$ sudo cp fuzz_controller.py /System/Library/Kernels/kernel.development.dSYM/Contents/Resources/Python/lldbmacros/
lilang-wudemini:lldbmacros lilang_wu$ sudo cp fuzz_controller.py /Library/Developer/KDKs/KDK_10.14.5_18F132.kdk/System/Library/Kernels/kernel.development.dSYM/Contents/Resources/Python/lldbmacros/
lilang-wudemini:lldbmacros lilang_wu$ sudo cp fuzz_controller.py /Library/Developer/KDKs/KDK_10.14.5_18F132.kdk/System/Library/Kernels/kernel.development.dSYM/Contents/Resources/Python/lldbmacros/
Password:
lilang-wudemini:lldbmacros lilang_wu$ sudo cp fuzz_controller.py /System/Library/Kernels/kernel.development.dSYM/Contents/Resources/Python/lldbmacros/
lilang-wudemini:lldbmacros lilang_wu$ sudo cp fuzz_* /Library/Developer/KDKs/KDK_10.14.5_18F132.kdk/System/Library/Kernels/kernel.development.dSYM/Contents/Resources/Python/lldbmacros/
lilang-wudemini:lldbmacros lilang_wu$ sudo cp fuzz_* /System/Library/Kernels/kernel.development.dSYM/Contents/Resources/Python/lldbmacros/
lilang-wudemini:lldbmacros lilang_wu$ sudo cp fuzz_* /System/Library/Kernels/kernel.development.dSYM/Contents/Resources/Python/lldbmacros/
Password:
Sorry, try again.
Password:
lilang-wudemini:lldbmacros lilang_wu$ sudo cp fuzz_* /Library/Developer/KDKs/KDK_10.14.5_18F132.kdk/System/Library/Kernels/kernel.development.dSYM/Contents/Resources/Python/lldbmacros/
lilang-wudemini:lldbmacros lilang_wu$ sudo cp fuzz_* /Library/Developer/KDKs/KDK_10.14.5_18F132.kdk/System/Library/Kernels/kernel.development.dSYM/Contents/Resources/Python/lldbmacros/
lilang-wudemini:lldbmacros lilang_wu$ sudo cp fuzz_* /System/Library/Kernels/kernel.development.dSYM/Contents/Resources/Python/lldbmacros/
lilang-wudemini:lldbmacros lilang_wu$ sudo cp fuzz_* /System/Library/Kernels/kernel.development.dSYM/Contents/Resources/Python/lldbmacros/
Password:
lilang-wudemini:lldbmacros lilang_wu$ sudo cp fuzz_* /Library/Developer/KDKs/KDK_10.14.5_18F132.kdk/System/Library/Kernels/kernel.development.dSYM/Contents/Resources/Python/lldbmacros/
lilang-wudemini:lldbmacros lilang_wu$ sudo cp fuzz_* /Library/Developer/KDKs/KDK_10.14.5_18F132.kdk/System/Library/Kernels/kernel.development.dSYM/Contents/Resources/Python/lldbmacros/
Password:
lilang-wudemini:lldbmacros lilang_wu$ sudo cp fuzz_* /System/Library/Kernels/kernel.development.dSYM/Contents/Resources/Python/lldbmacros/
lilang-wudemini:lldbmacros lilang_wu$ sudo cp fuzz_* /System/Library/Kernels/kernel.development.dSYM/Contents/Resources/Python/lldbmacros/
Password:
Sorry, try again.
Password:
lilang-wudemini:lldbmacros lilang_wu$ sudo cp fuzz_* /Library/Developer/KDKs/KDK_10.14.5_18F132.kdk/System/Library/Kernels/kernel.development.dSYM/Contents/Resources/Python/lldbmacros/
lilang-wudemini:lldbmacros lilang_wu$
```


lldbmacros — -bash — 266×58
condi
[lldbfuzz]: (mach_vm_address_t)ool_input=0
[lldbfuzz]: (mach_vm_size_t)ool_input_size=0
[lldbfuzz]: (char *)inband_output=0xffffff806fc5f610
[lldbfuzz]: (mach_msg_type_number_t *)inband_outputCnt=0xffffff806fc5f60c
[lldbfuzz]: (io_user_scalar_t *)scalar_output=0xffffffa3e309bce0
[lldbfuzz]: (mach_msg_type_number_t *)scalar_outputCnt=0xffffffa3e309bcdc
[lldbfuzz]: (mach_vm_address_t)ool_output=0
[lldbfuzz]: (mach_vm_size_t *)ool_output_size=0xffffff807e3d02c8
[lldbfuzz]: connection_addr = 0xffffff80705c9400
[lldbfuzz]: connection object is instance of `object 0xffffff80705c9400, vt 0xffffff7f9dd2c750 <vtable for IOTimeSyncDomainUserClient>, retain count 5, container retain 4`
[lldbfuzz]: ignore `object 0xffffff80705c9400, vt 0xffffff7f9dd2c750 <vtable for IOTimeSyncDomainUserClient>, retain count 5, container retain 4` !
(lldb) continue
Process 1 resuming
Process 1 stopped
* thread #4, name = '0xffffff806e9c5540', queue = '0x0', stop reason = breakpoint 1.1
frame #0: 0xffffff801d62e995 kernel.development`::is_io_connect_method(io_connect_t, uint32_t, io_user_scalar_t *, mach_msg_type_number_t, char *, mach_msg_type_number_t, mach_vm_address_t, mach_vm_size_t, char *, mach_msg_type_number_t *, io_user_scalar_t *, mach_msg_type_number_t *, mach_vm_address_t, mach_vm_size_t *) [inlined] OSMetaClassBase::safeMetaCast(me=0xffffff806b4b3100, toType=<unavailable>) at OSMetaClass.cpp:249 [opt]
Target 0: (kernel.development) stopped.
(lldb) [lldbfuzz]: end : fuzzing cycle 55...
[lldbfuzz]: start: fuzzing cycle 56...
fuzz
stop_reason = 3
[lldbfuzz]: frame 0 is inline one!
[lldbfuzz]: function name -> ::is_io_connect_method(io_connect_t, uint32_t, io_user_scalar_t *, mach_msg_type_number_t, char *, mach_msg_type_number_t, mach_vm_address_t, mach_vm_size_t, char *, mach_msg_type_number_t *, io_user_scalar_t *, mach_msg_type_number_t *, mach_vm_address_t, mach_vm_size_t *)
[lldbfuzz]: (io_connect_t)connection=0xffffff806b4b3100
[lldbfuzz]: (uint32_t)selector=2
[lldbfuzz]: (io_user_scalar_t *)scalar_input=0xffffff806b10b6d0
[lldbfuzz]: (mach_msg_type_number_t)scalar_inputCnt=0
[lldbfuzz]: (char *)inband_input=0xffffff806b10b6d4
[lldbfuzz]: (mach_msg_type_number_t)inband_inputCnt=168
[lldbfuzz]: (mach_vm_address_t)ool_input=0
[lldbfuzz]: (mach_vm_size_t)ool_input_size=0
[lldbfuzz]: (char *)inband_output=0xffffff806fc95e10
[lldbfuzz]: (mach_msg_type_number_t *)inband_outputCnt=0xffffff806fc95e0c
[lldbfuzz]: (io_user_scalar_t *)scalar_output=0xffffffa3e308bce0
[lldbfuzz]: (mach_msg_type_number_t *)scalar_outputCnt=0xffffffa3e308bcdc
[lldbfuzz]: (mach_vm_address_t)ool_output=0
[lldbfuzz]: (mach_vm_size_t *)ool_output_size=0xffffff806b10b79c
[lldbfuzz]: connection_addr = 0xffffff806b4b3100
[lldbfuzz]: connection object is instance of `object 0xffffff806b4b3100, vt 0xffffff7f9df63b10 <vtable for AppleSMCClient>, retain count 5, container retain 4`
[lldbfuzz]: ignore `object 0xffffff806b4b3100, vt 0xffffff7f9df63b10 <vtable for AppleSMCClient>, retain count 5, container retain 4`
(lldb) continue
Process 1 resuming
Process 1 stopped
* thread #2, name = '0xffffff806f7fd540', queue = '0x0', stop reason = EXC_BAD_ACCESS (code=1, address=0x86af0000)
frame #0: 0xffffff7fa00965d3 AMDRadeonX4000`AMDRadeonX4000_AMDAccelResource::initialize(IOAccelNewResourceArgs*, unsigned long long) + 1525
AMDRadeonX4000`AMDRadeonX4000_AMDAccelResource::initialize:
-> 0xffffff7fa00965d3 <+1525>: movq   0x98(%r15,%rdx), %rsi
0xffffff7fa00965db <+1533>: movq   %rsi, (%rax,%rdx)
0xffffff7fa00965df <+1537>: movq   0xa0(%r15,%rdx), %rsi
0xffffff7fa00965e7 <+1545>: movq   %rsi, 0x8(%rax,%rdx)
Target 0: (kernel.development) stopped.
(lldb) Traceback (most recent call last):
File "fuzz_main.py", line 49, in <module>
child.expect_exact("stop reason = breakpoint")
File "/Library/Python/2.7/site-packages/pexpect/spawnbase.py", line 418, in expect_exact
return exp.expect_loop(timeout)
elif arg_name == "inband_inputCnt":
inband_inputCnt = int(arg_v)
# get the input data by address
logger("connection_addr = %s" % connection_addr, 0)
if connection_addr:
obj = kern.GetValueFromAddress(connection_addr, '
kobj_str = GetObjectSummary(obj)
logger("connection object is instance of %s" % kobj_str, 1)
logger("fuzzing selector is %s" % selector, 1
is_black = False
if value == -1:
is_black = True
elif value ==selector:
is_black = True
if is_black:
return
#import pdb
#pdb.set_trace()
# flip byte
if scalar_inputCnt:
scalar_input_len = scalar_inputCnt*8
if scalar_input_len > io_scalar_inband64_t *
scalar_input_len = io_scalar_inband64_t *
scalar_input_content = lldb_SBProcess.ReadMemory(
if lldb_err.Success():
scalar_input_bytes = split_bytes(scalar_i
#scalar_input_data = scalar_input_content
#logger("before fuzz scalar_input_data =
flip_n_byte(scalar_input_bytes, scalar_in
P_MAX_BYTES)
uzz scalar_input_data = %s
result = lldb_SBProcess.WriteMemory(Argume
if not lldb_err.Success():
logger('SBProcess.WriteMemory() faile
else:
logger("fuzzing object %s, selector
if inband_inputCnt:
inband_input_len = io_struct_inband_t
if inband_input_len > inband_inputCnt:
inband_input_len = inband_inputCnt
print "inband_input_len = %d, inband_inputCnt = %d" % (inband_input_len, inband_inputCnt)
options=63<RXCSUM,TXCSUM,TSO4,TSO6>
ether 2a:00:01:58:10:00
Configuration:
id 0:0:0:0:0:0 priority 0 hellotime 0 fwddelay 0
maxage 0 holdcnt 0 proto stp maxaddr 100 timeout 1200
root id 0:0:0:0:0:0 priority 0 ifcost 0 port 0
ipfilter disabled flags 0x2
member: en2 flags=3<LEARNING,DISCOVER>
ifmaxaddr 0 port 7 priority 0 path cost 0
member: en3 flags=3<LEARNING,DISCOVER>
ifmaxaddr 0 port 8 priority 0 path cost 0
nd6 options=201<PERFORMNUD,DAD>
media: <unknown type>
status: inactive
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
nd6 options=201<PERFORMNUD,DAD>
media: autoselect (1000baseT <full-duplex,energy-efficient-ethernet>)
status: active
en2: flags=8963<UP,BROADCAST,SMART,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu 1500
ether 2a:00:01:58:10:00
media: autoselect <full-duplex>
status: inactive
en3: flags=8963<UP,BROADCAST,SMART,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu 1500
options=60<TSO4,TSO6>
ether 2a:00:01:58:10:01
media: autoselect <full-duplex>
status: inactive
en1: flags=8823<UP,BROADCAST,SMART,SIMPLEX,MULTICAST> mtu 1500
ether 08:6d:41:e7:9b:0c
nd6 options=201<PERFORMNUD,DAD>
media: autoselect (<unknown type>)
status: inactive
p2p0: flags=8802<BROADCAST,SIMPLEX,MULTICAST> mtu 2304
ether 0a:6d:41:e7:9b:0c
media: autoselect
status: inactive
awdl0: flags=8903<UP,BROADCAST,PROMISC,SIMPLEX,MULTICAST> mtu 1484
ether 72:fc:8f:3a:68:15
nd6 options=201<PERFORMNUD,DAD>
media: autoselect
status: inactive
utun0: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 2000
inet6 fe80::78b9:b3c5:70f7:516f%utun0 prefixlen 64 scopeid 0xc
nd6 options=201<PERFORMNUD,DAD>
lilang-wudemini:lldbmacros lilang_wu$ sudo cp fuzz_* /Library/Developer/KDKs/KDK_10.14.2_18C54.kdk/System/Library/Kernels/k
ernel.development.dSYM/Contents/Resources/Python/lldbmacros/
Password:
lilang-wudemini:lldbmacros lilang_wu$ sudo cp fuzz_* /Library/Developer/KDKs/KDK_10.14.2_18C54.kdk/System/Library/Kernels/k
ernel.development.dSYM/Contents/Resources/Python/lldbmacros/
Password:
lilang-wudemini:lldbmacros lilang_wu$ sudo cp fuzz_* /Library/Developer/KDKs/KDK_10.14.2_18C54.kdk/System/Library/Kernels/k
ernel.development.dSYM/Contents/Resources/Python/lldbmacros/
179:44 LF UTF-8




```
flyic-pro2:PanicDumps user$ ls -l -t
total 1950320
drwxr-xr-x  8 root    wheel        256 Apr  1 10:39 10.64.20.40_AMDRadeonX4000_AMDSIGLContext_2_2019_04_01_09_57_37_511040
drwxr-xr-x  7 root    wheel        224 Apr  1 00:40 10.64.20.40_AMDRadeonX4000_AMDAccelSharedUserClient_0_2019_04_01_00_35_59_749451
drwxr-xr-x  7 root    wheel        224 Mar 29 16:15 10.64.20.40_AMDRadeonX4000_AMDAccelSharedUserClient_262_2019_03_29_16_11_25_990669
drwxr-xr-x  7 root    wheel        224 Mar 29 16:02 10.64.21.114_AMDRadeonX4000_AMDAccelSharedUserClient_262_2019_03_29_15_59_29_115073
drwxr-xr-x  7 root    wheel        224 Mar 29 15:57 10.64.21.114_AMDRadeonX4000_AMDAccelSharedUserClient_262_2019_03_29_15_54_12_169443
drwxr-xr-x  7 root    wheel        224 Mar 29 15:56 10.64.20.40_AMDRadeonX4000_AMDAccelSharedUserClient_262_2019_03_29_15_54_50_103416
drwxr-xr-x  8 root    wheel        256 Mar 29 15:55 10.64.21.114_AMDRadeonX4000_AMDAccelSharedUserClient_262_2019_03_29_15_17_45_748709
drwxr-xr-x  7 root    wheel        224 Mar 29 15:47 10.64.20.40_AMDRadeonX4000_AMDAccelSharedUserClient_262_2019_03_29_15_42_53_898777
drwxr-xr-x  7 root    wheel        224 Mar 29 15:31 10.64.20.40_AMDRadeonX4000_AMDAccelSharedUserClient_262_2019_03_29_15_27_35_528927
drwxr-xr-x  8 root    wheel        256 Mar 29 15:12 10.64.21.114_AMDRadeonX4000_AMDAccelSharedUserClient_262_2019_03_29_14_40_22_034404
drwxr-xr-x  7 root    wheel        224 Mar 29 14:57 10.64.21.114_default_default_2019_03_29_14_51_22_434463
drwxr-xr-x  7 root    wheel        224 Mar 29 14:33 10.64.21.114_AMDRadeonX4000_AMDAccelSharedUserClient_262_2019_03_29_14_29_54_704299
drwxr-xr-x  7 root    wheel        224 Mar 29 14:33 10.64.20.40_AMDRadeonX4000_AMDAccelSharedUserClient_262_2019_03_29_14_29_04_459096
drwxr-xr-x  7 root    wheel        224 Mar 29 14:25 10.64.21.114_AMDRadeonX4000_AMDAccelSharedUserClient_262_2019_03_29_14_21_55_090353
drwxr-xr-x  7 root    wheel        224 Mar 29 14:09 10.64.20.40_AMDRadeonX4000_AMDAccelSharedUserClient_262_2019_03_29_14_04_52_239242
drwxr-xr-x  7 root    wheel        224 Mar 29 13:55 10.64.20.40_AMDRadeonX4000_AMDAccelSharedUserClient_262_2019_03_29_13_51_24_739963
drwxr-xr-x  7 root    wheel        224 Mar 29 13:47 10.64.20.40_AMDRadeonX4000_AMDAccelSharedUserClient_262_2019_03_29_13_43_33_540595
drwxr-xr-x  7 root    wheel        224 Mar 28 21:36 10.64.21.114_AMDRadeonX4000_AMDAccelSharedUserClient_262_2019_03_28_21_32_34_650774
drwxr-xr-x  7 root    wheel        224 Mar 28 21:27 10.64.21.114_AMDRadeonX4000_AMDAccelSharedUserClient_0_2019_03_28_21_23_58_107001
drwxr-xr-x  7 root    wheel        224 Mar 28 18:49 10.64.21.114_default_default_2019_03_28_18_45_43_290468
drwxr-xr-x  7 root    wheel        224 Mar 28 15:56 10.64.21.114_AMDRadeonX4000_AMDAccelCommandQueue_1_2019_03_28_15_52_53_289358
drwxr-xr-x  7 root    wheel        224 Mar 28 15:03 10.64.21.114_default_default_2019_03_28_14_56_49_999724
drwxr-xr-x  7 root    wheel        224 Mar 28 14:20 10.64.21.114_AMDRadeonX4000_AMDAccelSharedUserClient_262_2019_03_27_23_16_50_255435
drwxr-xr-x  7 root    wheel        224 Mar 28 14:05 10.64.21.114_AMDRadeonX4000_AMDAccelSharedUserClient_0_2019_03_27_23_01_23_273973
drwxr-xr-x  7 root    wheel        224 Mar 28 13:53 10.64.21.114_AMDRadeonX4000_AMDAccelSharedUserClient_0_2019_03_27_22_49_51_041248
drwxr-xr-x  7 root    wheel        224 Mar 28 13:37 10.64.21.114_AMDRadeonX4000_AMDAccelSharedUserClient_0_2019_03_27_22_33_48_437381
drwxr-xr-x  7 root    wheel        224 Mar 28 13:05 10.64.21.114_default_default_2019_03_27_21_59_28_657716
drwxr-xr-x  7 root    wheel        224 Mar 28 12:38 10.64.21.114_AMDRadeonX4000_AMDAccelSharedUserClient_0_2019_03_27_21_34_40_206774
drwxr-xr-x  7 root    wheel        224 Mar 28 12:33 10.64.21.114_AMDRadeonX4000_AMDAccelSharedUserClient_262_2019_03_27_21_30_06_979255
drwxr-xr-x  7 root    wheel        224 Mar 28 12:06 10.64.21.114_default_default_2019_03_27_20_59_48_155526
drwxr-xr-x  7 root    wheel        224 Mar 28 05:11 10.64.21.114_default_default_2019_03_27_14_08_15_261957
drwxr-xr-x  7 root    wheel        224 Mar 28 02:54 10.64.21.114_AMDRadeonX4000_AMDSIGLContext_2_2019_03_27_11_50_54_265786
drwxr-xr-x  7 root    wheel        224 Mar 28 02:29 10.64.21.114_AMDRadeonX4000_AMDAccelSharedUserClient_0_2019_03_27_11_25_48_215336
drwxr-xr-x  7 root    wheel        224 Mar 28 01:44 10.64.21.114_AMDRadeonX4000_AMDAccelSharedUserClient_0_2019_03_27_10_40_56_801828
drwxr-xr-x  7 root    wheel        224 Mar 27 23:36 10.64.21.114_AMDRadeonX4000_AMDAccelSharedUserClient_0_2019_03_27_08_33_06_386683
drwxr-xr-x  7 root    wheel        224 Mar 27 23:19 10.64.21.114_AMDRadeonX4000_AMDAccelSharedUserClient_262_2019_03_27_08_15_53_122847
drwxr-xr-x  7 root    wheel        224 Mar 27 23:14 10.64.21.114_default_default_2019_03_27_08_11_28_554925
drwxr-xr-x  7 root    wheel        224 Mar 26 15:28 10.64.21.114_default_default_2019_03_26_00_21_12_295204
drwxr-xr-x  7 root    wheel        224 Mar 26 13:42 10.64.21.114_default_default_2019_03_25_22_42_31_591482
drwxr-xr-x  7 root    wheel        224 Mar 23 16:07 10.64.21.114_default_default_2019_03_23_01_03_11_321917
drwxr-xr-x  7 root    wheel        224 Mar 23 14:43 10.64.21.114_AMDRadeonX4000_AMDAccelSharedUserClient_0_2019_03_22_23_38_27_423367
drwxr-xr-x  7 root    wheel        224 Mar 23 14:31 10.64.21.114_AMDRadeonX4000_AMDAccelCommandQueue_1_2019_03_22_23_27_40_631125
```

Agenda

- Kernel Debugger Overview
- The Introduction of LLDBFuzzer
- Attack Surfaces on Graphic Extensions
- Practice and Demo
- Vulnerabilities Found
- Implement a Debugger for Hackintosh
- Conclusion

CVE-2019-8519 - OOB in AMD Driver

* thread #1, stop reason = signal SIGSTOP

  * frame #0: 0xffffff7fa00965d3 AMDRadeonX4000`AMDRadeonX4000_AMDAccelResource::initialize(IOAccelNewResourceArgs*, unsigned long long) + 1525

    frame #1: 0xffffff7f9fea346b IOAcceleratorFamily2`IOAccelSharedUserClient2::new_resource(IOAccelNewResourceArgs*, IOAccelNewResourceReturnData*, unsigned long long, unsigned int*) + 1893

    frame #2: 0xffffff7f9fea4a41 IOAcceleratorFamily2`IOAccelSharedUserClient2::s_new_resource(IOAccelSharedUserClient2*, void*, IOExternalMethodArguments*) + 151

    frame #3: 0xffffff801d625ab8 kernel.development`IOUserClient::externalMethod(this=<unavailable>, selector=<unavailable>, args=0xffffff83dd4b3b58, dispatch=0xffffff7f9fee8260, target=0xffffff80854fd780, reference=0x0000000000000000) at IOUserClient.cpp:5358 [opt]

    frame #4: 0xffffff7f9fea4d98 IOAcceleratorFamily2`IOAccelSharedUserClient2::externalMethod(unsigned int, IOExternalMethodArguments*, IOExternalMethodDispatch*, OSObject*, void*) + 120

    frame #5: 0xffffff801d62eb7f kernel.development`::is_io_connect_method(connection=0xffffff80854fd780, selector=0, scalar_input=<unavailable>, scalar_inputCnt=<unavailable>, inband_input=<unavailable>, inband_inputCnt=2424, ool_input=0, ool_input_size=0, inband_output="", inband_outputCnt=0xffffff806ba03e0c, scalar_output=0xffffff83dd4b3ce0, scalar_outputCnt=0xffffff83dd4b3cdc, ool_output=0, ool_output_size=0xffffff8085919d5c) at IOUserClient.cpp:3994 [opt]

    frame #6: 0xffffff801cfbbce4 kernel.development`_Xio_connect_method(InHeadP=<unavailable>, OutHeadP=0xffffff806ba03de0) at device_server.c:8379 [opt]

    frame #7: 0xffffff801ce8d27d kernel.development`ipc_kobject_server(request=0xffffff8085919000, option=<unavailable>) at ipc_kobject.c:359 [opt]

    frame #8: 0xffffff801ce59465 kernel.development`ipc_kmsg_send(kmsg=0xffffff8085919000, option=3, send_timeout=0) at ipc_kmsg.c:1832 [opt]

```
__text:000000000000E58E loc_E58E:                       ; CODE XREF: AMDRadeonX4000_AMDAccelResource::initialize(IOAccelNewResourceArgs *,ulong long)+58Dj
__text:000000000000E58E             mov     ecx, [r15+0F8h]
__text:000000000000E595             test    rcx, rcx
__text:000000000000E598             jz      short loc_E603
__text:000000000000E59A             shl     rcx, 3
__text:000000000000E59E             lea     rdi, [rcx+rcx*2]
__text:000000000000E5A2             call    _IOMalloc
__text:000000000000E5A7             mov     [r12+178h], rax  --- rax== buffer address which create by IOMalloc                    ------- (a)
__text:000000000000E5AF             test    rax, rax
__text:000000000000E5B2             jz      short loc_E62A
__text:000000000000E5B4             or      byte ptr [r12+186h], 8
__text:000000000000E5BD             mov     ecx, [r15+0F8h]  --------r15==structureInput,    ecx=( (uint32_t*) structureInput+62)          ---------(b)
__text:000000000000E5C4             mov     [r12+180h], ecx
__text:000000000000E5CC             test    rcx, rcx                 ------ test rcx, if zero, break                                 ----------(c)
__text:000000000000E5CF             jz      short loc_E639
__text:000000000000E5D1             xor     edx, edx                 ------- index                                                   ---------(d)
__text:000000000000E5D3
__text:000000000000E5D3 loc_E5D3:                       ; CODE XREF: AMDRadeonX4000_AMDAccelResource::initialize(IOAccelNewResourceArgs *,ulong long)+621j
__text:000000000000E5D3             mov     rsi, [r15+rdx+98h]       ---- mov structureInput+rdx+0x98 to rsi
__text:000000000000E5DB             mov     [rax+rdx], rsi           ----mov rsi to rax+rdx, rax== buffer address which create by IOMalloc  ————————(g) crash point
__text:000000000000E5DF             mov     rsi, [r15+rdx+0A0h]
__text:000000000000E5E7             mov     [rax+rdx+8], rsi
__text:000000000000E5EC             mov     esi, [r15+rdx+0A8h]
__text:000000000000E5F4             mov     [rax+rdx+10h], esi
__text:000000000000E5F8             add     rdx, 18h                                                                                -------(e)
__text:000000000000E5FC             dec     rcx                        ----decrease rcx value                                       --------(f)
__text:000000000000E5FF             jnz     short loc_E5D3
```

CVE-2019-8692 - Another OOB in AMD Driver

```
* thread #1, stop reason = signal SIGSTOP
    frame #0: 0xffffff7f9dcd9459 AMDRadeonX4000`AMDRadeonX4000_AMDAccelResource::initialize(IOAccelNewResourceArgs*, unsigned long long) + 947
    frame #1: 0xffffff7f9dc345ee IOAcceleratorFamily2`IOAccelSharedUserClient2::new_resource(IOAccelNewResourceArgs*,
IOAccelNewResourceReturnData*, unsigned long long, unsigned int*) + 1886
    frame #2: 0xffffff7f9dc35bb5 IOAcceleratorFamily2`IOAccelSharedUserClient2::s_new_resource(IOAccelSharedUserClient2*, void*,
IOExternalMethodArguments*) + 151
    frame #3: 0xffffff801b424978 kernel.development`IOUserClient::externalMethod(this=<unavailable>, selector=<unavailable>, args=0xffffffa76a5bb9b8,
dispatch=0xffffff7f9dc79260, target=<unavailable>, reference=<unavailable>) at IOUserClient.cpp:5689 [opt]
    frame #4: 0xffffff7f9dc35f0b IOAcceleratorFamily2`IOAccelSharedUserClient2::externalMethod(unsigned int, IOExternalMethodArguments*,
IOExternalMethodDispatch*, OSObject*, void*) + 119
  * frame #5: 0xffffff801b42da02 kernel.development`::is_io_connect_method(connection=<unavailable>, selector=0, scalar_input=<unavailable>,
scalar_inputCnt=<unavailable>, inband_input=<unavailable>, inband_inputCnt=2424, ool_input=0, ool_input_size=0, inband_output="",
inband_outputCnt=0xffffff80bf24e60c, scalar_output=0xffffffa76a5bbce0, scalar_outputCnt=0xffffffa76a5bbcdc, ool_output=0,
ool_output_size=0xffffff80beec9d5c) at IOUserClient.cpp:4304 [opt]
    frame #6: 0xffffff801adbc386 kernel.development`_Xio_connect_method(InHeadP=<unavailable>, OutHeadP=0xffffff80bf24e5e0) at device_server.c:8379 [opt]
    frame #7: 0xffffff801ac948fd kernel.development`ipc_kobject_server(request=0xffffff80beec9000, option=3) at ipc_kobject.c:361 [opt]
    frame #8: 0xffffff801ac6088e kernel.development`ipc_kmsg_send(kmsg=0xffffff80beec9000, option=3, send_timeout=0) at ipc_kmsg.c:1868 [opt]
    frame #9: 0xffffff801ac800e3 kernel.development`mach_msg_overwrite_trap(args=<unavailable>) at mach_msg.c:553 [opt]
    frame #10: 0xffffff801adf702b kernel.development`mach_call_munger64(state=0xffffff80bd7429a0) at bsd_i386.c:580 [opt]
    frame #11: 0xffffff801ac2a476 kernel.development`hndl_mach_scall64 + 22
```

```
__int64 __cdecl AMDRadeonX4000_AMDAccelResource::initialize(AMDRadeonX4000_AMDAccelResource *this, void *structureInput, unsigned __int64 structureI
{
  void *v3; // r15@1
  ... ... ...
  __int64 v63; // rax@79
  __int64 v64; // rbx@80
  signed __int64 v65; // rax@80
  __int64 v66; // rax@82
  signed __int64 v67; // rcx@83
  unsigned int v68; // edx@85
  int v69; // eax@87
  __int64 v70; // rcx@90
  signed __int64 v71; // rdx@90
  unsigned __int64 v72; // rsi@92

  v3 = structureInput;
  v4 = this;
  v5 = (*((int (__fastcall **)(_QWORD, _QWORD, _QWORD))`vtable for'IOAccelResource2 + 42))(
         this,
         structureInput,
         structureInputSize);
  v6 = this->member7;
  if ( v6 && *(_DWORD *)(v6 + 260) == 128 )
    this->member23 = this->member24;
```

call parent initialize function

- - - a)

```
__int64 __cdecl IOAccelResource2::initialize(IOAccelResource2 *this, void *structureInput, unsigned __int64 structureInputSi
{
  _DWORD *v3; // r14@1
  char v4; // al@2
  __int64 result; // rax@8
  __int64 v6; // rax@17
  unsigned int v7; // ecx@21
  __int64 v8; // r14@26
  char v9; // al@26
  int v10; // ecx@26

  v3 = structureInput;
  if ( *(_BYTE *)(this->member3 + 3194) & 0x40 )
  {
    v4 = BYTE4(this->member2);
    if ( structureInput )
    {
      if ( v4 == 1 )
        goto LABEL_8;
    }
    else if ( v4 != 10 )
    {
      goto LABEL_25;
    }
    LOBYTE(this->member33) |= 4u;
  }
  if ( structureInput )
  {
LABEL_8:
    BYTE4(this->member21) = *((_BYTE *)structureInput + 32);
    BYTE5(this->member21) = *((_BYTE *)structureInput + 33);
    BYTE6(this->member21) = *((_BYTE *)structureInput + 34);
    WORD1(this->member22) = *((_WORD *)structureInput + 5);
    WORD2(this->member22) = *((_WORD *)structureInput + 6);
    this->member23 = *((_QWORD *)structureInput + 2);
    LOWORD(this->member27) = *((_BYTE *)structureInput + 35);
    this->member24 = *((_QWORD *)structureInput + 3);
    result = *((_DWORD *)structureInput + 9);
    if ( (char)result < 0 )
    {
```

- - - b)

```
  ... ... ...
  if ( !(unsigned __int8)HIDWORD(v4->member48) )
    WORD2(v4->member21) = 257;
  v4->member43 = 0LL;
  LOBYTE(v4->member49) &= 0xE7u;
  v33 = *((_DWORD *)v3 + 63);
  if ( v33 & 1 )
  {
    v34 = (v33 >> 1) & 3;
    v35 = 1;
    if ( v34 )
      v35 = v34;
    LODWORD(v36) = IOMalloc((unsigned int)(968 * v35));
    v4->member43 = v36;
    if ( v36 )
    {
      v37 = v36;
      LOBYTE(v4->member49) = 8 * v35 | v4->member49 & 0xE7;
      v38 = 0LL;
      v39 = 0LL;
      do
      {
        memcpy((void *)(v38 + v37), (char *)v3 + v38 + 488, 0x3C8uLL);
        ++v39;
        v37 = v4->member43;
        v38 += 968LL;
      }
      while ( v39 < (((unsigned int)BYTE6(v4->member49) >> 3) & 3) );
      WORD1(v4->member49) = *(_WORD *)(v37 + ((unsigned __int64)BYTE6(v4->member21) << 6) + 54);
      v32 = (unsigned __int8)HIDWORD(v4->member48);
      goto LABEL_50;
    }
    return 0LL;
  }
```

v37 = v36 = buffer start address

- - - c)

here, BYTE6(member21) can be controlled by structureInput+34 byte

CVE-2019-8635 - Double Free in AMD Driver

```
* thread #1, stop reason = signal SIGSTOP
    frame #0: 0xffffff7f8d7adc37 IOAcceleratorFamily2`IOAccelResource2::clientRelease(IOAccelShared2*) + 13
    frame #1: 0xffffff7f8d880dad AMDRadeonX4000`AMDRadeonX4000_AMDSIGLContext::process_StretchTex2Tex(IOAccelCommandStreamInfo&) + 2893
    frame #2: 0xffffff7f8d79b5d5 IOAcceleratorFamily2`IOAccelContext2::processSidebandBuffer(IOAccelCommandDescriptor*, bool) + 273
    frame #3: 0xffffff7f8d8885e4 AMDRadeonX4000`AMDRadeonX4000_AMDSIGLContext::processSidebandBuffer(IOAccelCommandDescriptor*, bool) + 182
    frame #4: 0xffffff7f8d79bae7 IOAcceleratorFamily2`IOAccelContext2::processDataBuffers(unsigned int) + 85
    frame #5: 0xffffff7f8d7a2380 IOAcceleratorFamily2`IOAccelGLContext2::processDataBuffers(unsigned int) + 804
    frame #6: 0xffffff7f8d798c30 IOAcceleratorFamily2`IOAccelContext2::submit_data_buffers(IOAccelContextSubmitDataBuffersIn*,
IOAccelContextSubmitDataBuffersOut*, unsigned long long, unsigned long long*) + 1208
    frame #7: 0xffffff800b027a3c kernel.development`::shim_io_connect_method_structureI_structureO(method=<unavailable>, object=<unavailable>,
input=<unavailable>, inputCount=<unavailable>, output=<unavailable>, outputCount=0xffffff8742023968) at IOUserClient.cpp:0 [opt]
    frame #8: 0xffffff800b025ca0 kernel.development`IOUserClient::externalMethod(this=<unavailable>, selector=<unavailable>, args=0xffffff87420239b8,
dispatch=0x0000000000000000, target=0x0000000000000000, reference=<unavailable>) at IOUserClient.cpp:5459 [opt]
  * frame #9: 0xffffff800b02ebff kernel.development`::is_io_connect_method(connection=0xffffff80b094e000, selector=2, scalar_input=<unavailable>,
scalar_inputCnt=<unavailable>, inband_input=<unavailable>, inband_inputCnt=136, ool_input=0, ool_input_size=0, inband_output="",
inband_outputCnt=0xffffff80b0d81e0c, scalar_output=0xffffff8742023ce0, scalar_outputCnt=0xffffff8742023cdc, ool_output=0, ool_output_size=0xffffff80ab5c7574) at
IOUserClient.cpp:3994 [opt]
    frame #10: 0xffffff800a9bbd64 kernel.development`_Xio_connect_method(InHeadP=<unavailable>, OutHeadP=0xffffff8742023ce0) at device_server.c:8379 [opt]
    frame #11: 0xffffff800a88d27d kernel.development`ipc_kobject_server(request=0xffffff80ab5c7400, option=<unavailable>) at ipc_kobject.c:359 [opt]
    frame #12: 0xffffff800a859465 kernel.development`ipc_kmsg_send(kmsg=0xffffff80ab5c7400, option=3, send_timeout=0) at ipc_kmsg.c:1832 [opt]
    frame #13: 0xffffff800a878a75 kernel.development`mach_msg_overwrite_trap(args=<unavailable>) at mach_msg.c:549 [opt]
    frame #14: 0xffffff800a9f63a3 kernel.development`mach_call_munger64(state=0xffffff80af471bc0) at bsd_i386.c:573 [opt]
    frame #15: 0xffffff800a823486 kernel.development`hndl_mach_scall64 + 22
```

https://blog.trendmicro.com/trendlabs-security-intelligence/cve-2019-8635-double-free-vulnerability-in-apple-macos-lets-attackers-escalate-system-privileges-and-execute-arbitrary-code/

```
v2 = cmdinfo;
this_ptr = this;
if ( *(_DWORD *)kdebug_enable_0 & 0xFFFFFFF7 )
{
  v4 = IORegistryEntry::getRegistryEntryID(*((IORegistryEntry **)this + 173));
  v5 = IORegistryEntry::getRegistryEntryID(this);
  kernel_debug(0x85AB2025, v4, v5, 0LL, 0LL, 0LL);
}
shareMem_start_vm_address_187_offset16 = *(_QWORD *)(cmdinfo + 24);
if ( !(unsigned __int8)IOAccelContext2::validateTokenSize(
                         this,



if ( v15 == 35840 )          - - - a)          // 8c00
{
  if ( !(unsigned __int8)IOAccelShared2::lookupResource(
                           *((IOAccelShared2 **)this + 172),
                           *(_DWORD *)(shareMem_start_vm_address_187_offset16 + 8),
                           (void **)&accelResource_offset8) )
    goto LABEL_16;
}
else
{
  accelResource_offset8 = (AMDRadeonX4000_AMDAccelResource *)*((_QWORD *)this + 500);
  if ( !accelResource_offset8 )
    goto LABEL_16;
}
if ( (unsigned __int8)IOAccelShared2::lookupResource(
                        *((IOAccelShared2 **)this + 172),
                        *(_DWORD *)(shareMem_start_vm_address_187_offset16 + 12),
                        (void **)&accelResource_offset12) )// 3b
{
      ...

      ...

    IOAccelResource2::clientRelease(
      (IOAccelResource2 *)accelResource_offset12,
      *((IOAccelShared2 **)this_ptr + 172));
    if ( BYTE4(accelResource_offset8->member2) != 10 )
    {
      ((void (*)(void))accelResource_offset8->vtable->__ZNK16IOAccelResource27releaseEv)();
      IOAccelResource2::clientRelease(
        (IOAccelResource2 *)accelResource_offset8,
        *((IOAccelShared2 **)this_ptr + 172));// crash point
    }
    *(_QWORD *)(*((_QWORD *)this_ptr + 175) + 216LL) = 0LL;
    *(_QWORD *)(*((_QWORD *)this_ptr + 175) + 224LL) = 0LL;
    goto LABEL_65;
```

release accelResource offset8 from IOAccelShared2

release accelReource_offset12 from IOAccelShared2

- - - b)

```
v2 = cmdinfo;
this_ptr = this;
if ( *(_DWORD *)kdebug_enable_0 & 0xFFFFFFF7 )
{
  v4 = IORegistryEntry::getRegistryEntryID(*((IORegistryEntry **)this + 173));
  v5 = IORegistryEntry::getRegistryEntryID(this);
  kernel_debug(0x85AB2025, v4, v5, 0LL, 0LL, 0LL);
}
shareMem_start_vm_address_187_offset16 = *(_QWORD *)(cmdinfo + 24);
if ( !(unsigned __int8)IOAccelContext2::validateTokenSize(
                         this,
```

```
if ( v15 == 35840 )          - - - a)          // 8c00
{
  if ( !(unsigned __int8)IOAccelShared2::lookupResource(
                           *((IOAccelShared2 **)this + 172),
                           *(_DWORD *)(shareMem_start_vm_address_187_offset16 + 8),
                           (void **)&accelResource_offset8) )
    goto LABEL_16;
}
else
{
  accelResource_offset8 = (AMDRadeonX4000_AMDAccelResource *)*((_QWORD *)this + 500);
  if ( !accelResource_offset8 )
    goto LABEL_16;
}
if ( (unsigned __int8)IOAccelShared2::lookupResource(
                        *((IOAccelShared2 **)this + 172),
                        *(_DWORD *)(shareMem_start_vm_address_187_offset16 + 12),
                        (void **)&accelResource_offset12) )// 8b
{
```

```
      •••

      •••

    IOAccelResource2::clientRelease(
      (IOAccelResource2 *)accelResource_offset12,
      *((IOAccelShared2 **)this_ptr + 172));
    if ( BYTE4(accelResource_offset8->member2) != 10 )
    {
      ((void (*)(void))accelResource_offset8->vtable->__ZNK16IOAccelResource27releaseEv)();
      IOAccelResource2::clientRelease(
        (IOAccelResource2 *)accelResource_offset8,
        *((IOAccelShared2 **)this_ptr + 172));// crash point
    }
    *(_QWORD *)(*((_QWORD *)this_ptr + 175) + 216LL) = 0LL;
    *(_QWORD *)(*((_QWORD *)this_ptr + 175) + 224LL) = 0LL;
    goto LABEL_65;
```

release accelResource offset8 from IOAccelShared2

release accelReource_offset12 from IOAccelShared2

- - - b)

```
v2 = cmdinfo;
this_ptr = this;
if ( *(_DWORD *)kdebug_enable_0 & 0xFFFFFFF7 )
{
  v4 = IORegistryEntry::getRegistryEntryID(*((IORegistryEntry **)this + 173));
  v5 = IORegistryEntry::getRegistryEntryID(this);
  kernel_debug(0x85AB2025, v4, v5, 0LL, 0LL, 0LL);
}
shareMem_start_vm_address_187_offset16 = *(_QWORD *)(cmdinfo + 24);
if ( !(unsigned __int8)IOAccelContext2::validateTokenSize(
                         this,


if ( v15 == 35840 )          - - - a)          // 8c00
{
  if ( !(unsigned __int8)IOAccelShared2::lookupResource(
                           *((IOAccelShared2 **)this + 172),
                           *(_DWORD *)(shareMem_start_vm_address_187_offset16 + 8),
                           (void **)&accelResource_offset8) )
    goto LABEL_16;
}
else
{
  accelResource_offset8 = (AMDRadeonX4000_AMDAccelResource *)*(_QWORD *)this + 500);
  if ( !accelResource_offset8 )
    goto LABEL_16;
}
if ( (unsigned __int8)IOAccelShared2::lookupResource(
                        *((IOAccelShared2 **)this + 172),
                        *(_DWORD *)(shareMem_start_vm_address_187_offset16 + 12),
                        (void **)&accelResource_offset12) )// 8b
{
      •••

      •••

      •••,
    IOAccelResource2::clientRelease(
      (IOAccelResource2 *)accelResource_offset12,
      *((IOAccelShared2 **)this_ptr + 172));
    if ( BYTE4(accelResource_offset8->member2) != 10 )
    {
      ((void (*)(void))accelResource_offset8->vtable->__ZNK16IOAccelResource27releaseE
      IOAccelResource2::clientRelease(
        (IOAccelResource2 *)accelResource_offset8,
        *((IOAccelShared2 **)this_ptr + 172));// crash point
    }
    *(_QWORD *)(*((_QWORD *)this_ptr + 175) + 216LL) = 0LL;
    *(_QWORD *)(*((_QWORD *)this_ptr + 175) + 224LL) = 0LL;
    goto LABEL_65;
```

release accelResource c

release accelReource_offset12 f

- - - b)

```
__int64 __cdecl IOAccelContext2::processSidebandBuffer(IOAccelContext2 *this, void *a2, bool a3
{

  ---omitted code ---
  v4 = (char *)&this->comandStreamInfo;
  LODWORD(this->member199) = 0;
  this->comandStreamInfo = 0LL;
  this->member194 = 0LL;
  LODWORD(this->member195) = 0;
  v5 = this->shareMem_start_vm_address_187 + 16;
  this->comandStreamInfo_offset24 = v5;          //  // this->commandStreamInfo + 24
  LODWORD(this->member200) = 0;
  BYTE4(this->member200) = a3;
  while ( 1 )
  {
    v6 = this->shareMem_endt_vm_address_188;
    if ( v6 < v5 + 8 )
    {
      v14 = _os_log_default_0;
      _os_log_internal(
        &dword_0,
        _os_log_default_0,
        17LL,
        IOAccelContext2::processSidebandBuffer(IOAccelCommandDescriptor *,bool)::_os_log_fmt,
        "virtual bool IOAccelContext2::processSidebandBuffer(IOAccelCommandDescriptor *, bool)
      v15 = LOWORD(this->comandStreamInfo_offset32);
      v16 = WORD1(this->comandStreamInfo_offset32);
      _os_log_internal(
        &dword_0,
        v14,
        17LL,
        IOAccelContext2::setContextError(unsigned int)::_os_log_fmt,
        "void IOAccelContext2::setContextError(uint32_t)");
      goto LABEL_18;
    }
    LOWORD(this->comandStreamInfo_offset32) = *(_WORD *)v5;
    v7 = *(_WORD *)(v5 + 2);
    WORD1(this->comandStreamInfo_offset32) = v7;
    v8 = *(_DWORD *)(v5 + 4);
    HIDWORD(this->comandStreamInfo_offset32) = v8;
    this->member198 = v5 + 8;
```

CVE-2019-8635 - Another Double Free in AMD Driver

```
void __fastcall AMDRadeonX4000_AMDSIGLContext::discard_StretchTex2Tex(IORegistryEntry *this, __int64 cmdinfo)
{
  IORegistryEntry *v2; // r14@1
  _DWORD *v3; // r12@1
  uintptr_t v4; // r15@2
  uintptr_t v5; // rax@2
  __int64 shareMem_start_vm_address_187_offset16; // r15@3
  IOAccelResource2 *v7; // rdi@6
  uintptr_t v8; // rbx@13
  uintptr_t v9; // rax@13
  void *v10; // [sp+0h] [bp-30h]@3
  IOAccelResource2 *v11; // [sp+8h] [bp-28h]@3

  v2 = this;
  v3 = kdebug_enable_0;
  if ( *(_DWORD *)kdebug_enable_0 & 0xFFFFFFF7 )
  {
    v4 = IORegistryEntry::getRegistryEntryID(*((IORegistryEntry **)this + 173));
    v5 = IORegistryEntry::getRegistryEntryID(this);
    kernel_debug(0x85AB206D, v4, v5, 0LL, 0LL, 0LL);
  }
  shareMem_start_vm_address_187_offset16 = *(_QWORD *)(cmdinfo + 24);
  v10 = 0LL;
  v11 = 0LL;
  if ( (*(_WORD *)(cmdinfo + 32) & 0xFF00) == 35840 )
  {
    if ( (unsigned __int8)IOAccelShared2::lookupResource(
                            *((IOAccelShared2 **)this + 172),
                            *(_DWORD *)(shareMem_start_vm_address_187_offset16 + 8),
                            &v10)
      && (unsigned __int8)IOAccelShared2::lookupResource(
                            *((IOAccelShared2 **)this + 172),
                            *(_DWORD *)(shareMem_start_vm_address_187_offset16 + 12),
                            (void **)&v11) )
    {
      IOAccelResource2::clientRelease(v11, *((IOAccelShared2 **)this + 172));
      v7 = (IOAccelResource2 *)v10;
LABEL_10:
      IOAccelResource2::clientRelease(v7, *((IOAccelShared2 **)v2 + 172));
      goto LABEL_12;
```

CVE-2019-8691 - OOB in AMD Driver

```
(lldb) bt
* thread #1, stop reason = signal SIGSTOP
  * frame #0: 0xffffff7f849d49a0 AMDRadeonX4000`AMDRadeonX4000_AMDAccelResource::AndXorByteFlag(unsigned short, unsigned char, unsigned char) +
164
    frame #1: 0xffffff7f849dad9d AMDRadeonX4000`AMDRadeonX4000_AMDAccelSharedUserClient::RsrcAndXorByteFlag(AMDRsrcAndXorByteFlagPacket
const*, unsigned long long*) + 275
    frame #2: 0xffffff8001c27a3c kernel.development`::shim_io_connect_method_structureI_structureO(method=<unavailable>, object=<unavailable>,
input=<unavailable>, inputCount=<unavailable>, output=<unavailable>, outputCount=0xffffffa77393bab8) at IOUserClient.cpp:0:9 [opt]
    frame #3: 0xffffff8001c25ca0 kernel.development`IOUserClient::externalMethod(this=<unavailable>, selector=<unavailable>, args=0xffffffa77393bb58,
dispatch=0x0000000000000000, target=0x0000000000000000, reference=<unavailable>) at IOUserClient.cpp:5459:9 [opt]
    frame #4: 0xffffff7f8493af0b IOAcceleratorFamily2`IOAccelSharedUserClient2::externalMethod(unsigned int, IOExternalMethodArguments*,
IOExternalMethodDispatch*, OSObject*, void*) + 119
    frame #5: 0xffffff8001c2ebff kernel.development`::is_io_connect_method(connection=0xffffff80bff43fd0, selector=262, scalar_input=<unavailable>,
scalar_inputCnt=<unavailable>, inband_input=<unavailable>, inband_inputCnt=12, ool_input=0, ool_input_size=0, inband_output="",
inband_outputCnt=0xffffff80bfc3260c, scalar_output=0xffffffa77393bce0, scalar_outputCnt=0xffffffa77393bcdc, ool_output=0, ool_output_size=0xffffff809d1e0b0c)
at IOUserClient.cpp:3994:19 [opt]
    frame #6: 0xffffff80015bbd64 kernel.development`_Xio_connect_method(InHeadP=<unavailable>, OutHeadP=0xffffff80bfc325e0) at device_server.c:8379:18 [opt]
    frame #7: 0xffffff800148d27d kernel.development`ipc_kobject_server(request=0xffffff809d1e0a40, option=<unavailable>) at ipc_kobject.c:359:3 [opt]
    frame #8: 0xffffff8001459465 kernel.development`ipc_kmsg_send(kmsg=0xffffff809d1e0a40, option=3, send_timeout=0) at ipc_kmsg.c:1832:10 [opt]
    frame #9: 0xffffff8001478a75 kernel.development`mach_msg_overwrite_trap(args=<unavailable>) at mach_msg.c:549:8 [opt]
    frame #10: 0xffffff80015f63a3 kernel.development`mach_call_munger64(state=0xffffff80be434b20) at bsd_i386.c:573:24 [opt]
    frame #11: 0xffffff8001423486 kernel.development`hndl_mach_scall64 + 22
```

```
__int64 __cdecl AMDRadeonX4000_AMDAccelSharedUserClient::RsrcAndXorByteFlag(AMDRadeonX4000_AMDAccelSharedUserClient
{
  AMDRadeonX4000_AMDAccelSharedUserClient *v3; // r13@1
  IOAccelShared2 *v4; // r12@1
  __int64 v5; // rbx@1
  __int64 v6; // rdi@1
  AMDRadeonX4000_AMDAccelResource *v7; // rbx@3
  signed int v8; // er14@6
  __int64 v9; // rbx@7
  __int64 v10; // rbx@10
  void *v12; // [sp+8h] [bp-38h]@3
  unsigned __int64 *v13; // [sp+10h] [bp-30h]@1

  v13 = structureOutput;
  v3 = this;
  v4 = (IOAccelShared2 *)this->member32;
  v5 = this->graphicAcc;
  OSIncrementAtomic(v5 + 144);
  IOLockLock(*(_QWORD *)(v5 + 136));
  OSDecrementAtomic(v5 + 144);
  IOGraphicsAccelerator2::lock_busy((IOGraphicsAccelerator2 *)v5);
  (*(void (__fastcall **)(__int64, unsigned __int64 *, _QWORD))(*(_QWORD *)v5 + 2128LL))(v5, &unk_1168DA, 0LL);// IO
  v6 = this->graphicAcc;
  if ( *(_BYTE *)(v6 + 3160) & 2
    || (unsigned __int8)IOGraphicsAccelerator2::acceleratorWaitEnabled((IOGraphicsAccelerator2 *)v6) )
  {
    IOAccelShared2::lookupResource(v4, *((_DWORD *)structureInput + 1), &v12);
    v7 = (AMDRadeonX4000_AMDAccelResource *)v12;
    if ( v12 )
    {
      if ( *((_BYTE *)structureInput + 8) )
        AMDRadeonX4000_AMDAccelResource::AcquireCondFlagIndex((AMDRadeonX4000_AMDAccelResource *)v12, v13);
      else
        *v13 = 0LL;
      v8 = AMDRadeonX4000_AMDAccelResource::AndXorByteFlag(
             v7,
             *(_WORD *)structureInput,
             *((_BYTE *)structureInput + 2),
             *((_BYTE *)structureInput + 3));
    }
```

- - - a)

AMDRadeonX4000_AMDAccelResource::AndXorByteFlag(*this, (word*)structureInput, (byte*)structureInput +2, (byte*)structureInput +3)

```
…
__text:00000000000148FC              push    rbp
__text:00000000000148FD              mov     rbp, rsp
__text:0000000000014900              push    r15
__text:0000000000014902              push    r14
__text:0000000000014904              push    r13
__text:0000000000014906              push    r12
__text:0000000000014908              push    rbx
__text:0000000000014909              push    rax
__text:000000000001490A              mov     r15d, edx
__text:000000000001490D              mov     r12d, esi           // r12 = psi = (word*)structureInput            ——-a)
__text:0000000000014910              mov     r13, rdi
__text:0000000000014913              mov     ebx, [rdi+1D0h]
__text:0000000000014919              cmp     ebx, esi
__text:000000000001491B              ja      short loc_1498B
--- omitted code ---

__text:000000000001498B loc_1498B:                          ; CODE XREF: AMDRadeonX4000_AMDAccelResource::AndXorByteFlag(ushort,uchar,uchar)+1Fj
__text:000000000001498B              mov     eax, 0E00002BDh
__text:0000000000014990              cmp     ebx, r12d
__text:0000000000014993              jbe     short loc_149AD
__text:0000000000014995              mov     rax, [r13+1C8h]    //here, rax is the value which rdi+0x1c8 point to a buffer start address          ——-b
__text:000000000001499C              movzx   edx, r12w          // edx = r12w                 ——c)
__text:00000000000149A0              and     r15b, [rax+rdx]    // oob here    crash point2.  —-d)
__text:00000000000149A4              xor     r15b, cl
__text:00000000000149A7              mov     [rax+rdx], r15b
__text:00000000000149AB              xor     eax, eax
__text:00000000000149AD
__text:00000000000149AD loc_149AD:                          ; CODE XREF: AMDRadeonX4000_AMDAccelResource::AndXorByteFlag(ushort,uchar,uchar)+97j
__text:00000000000149AD              add     rsp, 8
__text:00000000000149B1              pop     rbx

--- omitted code ---

__text:00000000000149BB __ZN31AMDRadeonX4000_AMDAccelResource14AndXorByteFlagEthh endp
```

CVE-2019-8616 - Execute Arbitrary Code bug in Intel Graphics Driver

```
(lldb) bt
* thread #1, stop reason = signal SIGSTOP
  * frame #0: 0xffffff8012ba4050 kernel.development`memcpy + 11
    frame #1: 0xffffff7f98f0358b AppleIntelHD5000Graphics`IntelAccelerator::newGTT(unsigned int**, bool, IGAccelTask&) + 173
    frame #2: 0xffffff7f98eebce8 AppleIntelHD5000Graphics`IntelPPGTT::init(IntelAccelerator&, bool, IGAccelTask&) + 24
    frame #3: 0xffffff7f98ef47dc AppleIntelHD5000Graphics`IGAccelTask::prepare(IntelAccelerator&) + 38
    frame #4: 0xffffff7f98f0348b AppleIntelHD5000Graphics`IntelAccelerator::createUserGPUTask() + 219
    frame #5: 0xffffff7f98980382 IOAcceleratorFamily2`IOAccelShared2::init(IOGraphicsAccelerator2*, task*) + 48
    frame #6: 0xffffff7f9899513b IOAcceleratorFamily2`IOGraphicsAccelerator2::createShared(task*) + 51
    frame #7: 0xffffff7f98983921 IOAcceleratorFamily2`IOAccelSharedUserClient2::sharedStart() + 43
    frame #8: 0xffffff7f98ee4e22 AppleIntelHD5000Graphics`IGAccelSharedUserClient::sharedStart() + 22
    frame #9: 0xffffff7f9898191a IOAcceleratorFamily2`IOAccelSharedUserClient2::start(IOService*) + 156
    frame #10: 0xffffff7f98994a1a IOAcceleratorFamily2`IOGraphicsAccelerator2::newUserClient(task*, void*, unsigned int, IOUserClient**) + 1088
    frame #11: 0xffffff80133c9bc1 kernel.development`IOService::newUserClient(this=0xffffff8037dc4800, owningTask=0xffffff803be31760, securityID=0xffffff803be31760,
type=6, properties=0x0000000000000000, handler=0xffffff9214a2bd10) at IOService.cpp:5856 [opt]
    frame #12: 0xffffff801342ce60 kernel.development`::is_io_service_open_extended(_service=0xffffff8037dc4800, owningTask=0xffffff803be31760, connect_type=6,
ndr=<unavailable>, properties=<unavailable>, propertiesCnt=<unavailable>, result=0xffffff804e2b9bb8, connection=0xffffff9214a2bd60) at IOUserClient.cpp:3491 [opt]
    frame #13: 0xffffff8012dba714 kernel.development`_Xio_service_open_extended(InHeadP=0xffffff8046905504, OutHeadP=0xffffff804e2b9b7c) at device_server.c:8003 [opt]
    frame #14: 0xffffff8012c8c27d kernel.development`ipc_kobject_server(request=0xffffff80469054a0, option=<unavailable>) at ipc_kobject.c:359 [opt]
    frame #15: 0xffffff8012c58465 kernel.development`ipc_kmsg_send(kmsg=0xffffff80469054a0, option=3, send_timeout=0) at ipc_kmsg.c:1832 [opt]
    frame #16: 0xffffff8012c77a75 kernel.development`mach_msg_overwrite_trap(args=<unavailable>) at mach_msg.c:549 [opt]
    frame #17: 0xffffff8012df52c3 kernel.development`mach_call_munger64(state=0xffffff803c0fea00) at bsd_i386.c:573 [opt]
    frame #18: 0xffffff8012c22486 kernel.development`hndl_mach_scall64 + 22
```

v8 is not always returns normal address. may be null

```
v8 = (unsigned int *)IOAccelSysMemory::lockForCPUAccess(
                       *(IOAccelSysMemory **)(v6 + 24),
                       *(task **)kernel_task_0,
                       1u);
*a2 = v8;
if ( v5 )
{
  v9 = this->member547;
  if ( v9 )
  {
    v10 = this->member546;
    v11 = 0LL;
    do
    {
      *(unsigned int *)((char *)v8 + (unsigned int)v11) = *(_DWORD *)(v10 + v11);
      v11 = (unsigned int)(v11 + 4);
    }
    while ( v9 > v11 );
  }
}
else
{
  memcpy(v8, *(const void **)(this->member44 + 616), LODWORD(this->member551) >> 10);
  IntelAccelerator::releaseGARTMemory(this, LODWORD(this->member552), LODWORD(this->m
  IntelAccelerator::releaseGARTMemory(this, LODWORD(this->member554), LODWORD(this->m
}
```

use v8 as dst addr

---(a)

```
if ( *(task **)kernel_task_0 == a2 )
{
  v12 = this->member45;
  if ( !v12 )
  {
    v12 = IOMemoryDescriptor::createMappingInTask((IOMemoryDes
    v5->member45 = v12;
    if ( !v12 )
    {
      v11 = 0LL;
      _os_log_internal(
        &dword_0,
        _os_log_default_0,
        17LL,
        IOAccelSysMemory::lockForCPUAccess(task *,unsigned int
        "mach_vm_address_t IOAccelSysMemory::lockForCPUAccess(
      return v11;
    }
  }
```

Agenda

- Kernel Debugger Overview
- The Introduction of LLDBFuzzer
- Attack Surfaces on Graphic Extensions
- Practice and Demo
- Vulnerabilities Found
- Implement a Debugger for Hackintosh
- Conclusion

Why?

To fuzz in Hackintosh

Open Source Network Drivers

RTL8100.kext	RTL8107E、RTL810X、RTL8139
RTL8111.kext	Realtek RTL8111/8168 B/C/D/E/F/G/H
IntelMausiEthernet.kext	82578LM、82578LC、82578DM、82578DC、82579LM、82579V、I217LM、I217V、I218LM、I218V、I218LM2、I218V2、I218LM3、I219V、I219LM、I219V2、I219LM2、I219LM2
AppleIntelE1000.kext	Intel series 82540, 82541, 82542, 82543, 82544, 82545, 82546, 82547, 82578 (P55/H55) 82579 (P67/H67) 82574L 82571 82572 82573 82574 82583 I217V
AtherosE2200Ethernet.kext	AR816x、AR817x、Killer E220x、Killer E2400、Killer E2500
SmallTree-Intel-211-AT-PCIe-GBE.kext	Intel I211
AppleIGB.kext	Intel 82575, 82576, 82580, dh89xxcc, i350, i210 and i211
FakePCIID_BCM57XX_as_BCM57765.kext	BCM57XX
FakePCIID_Intel_GbX.kext	Small Tree drivers for Intel chipset

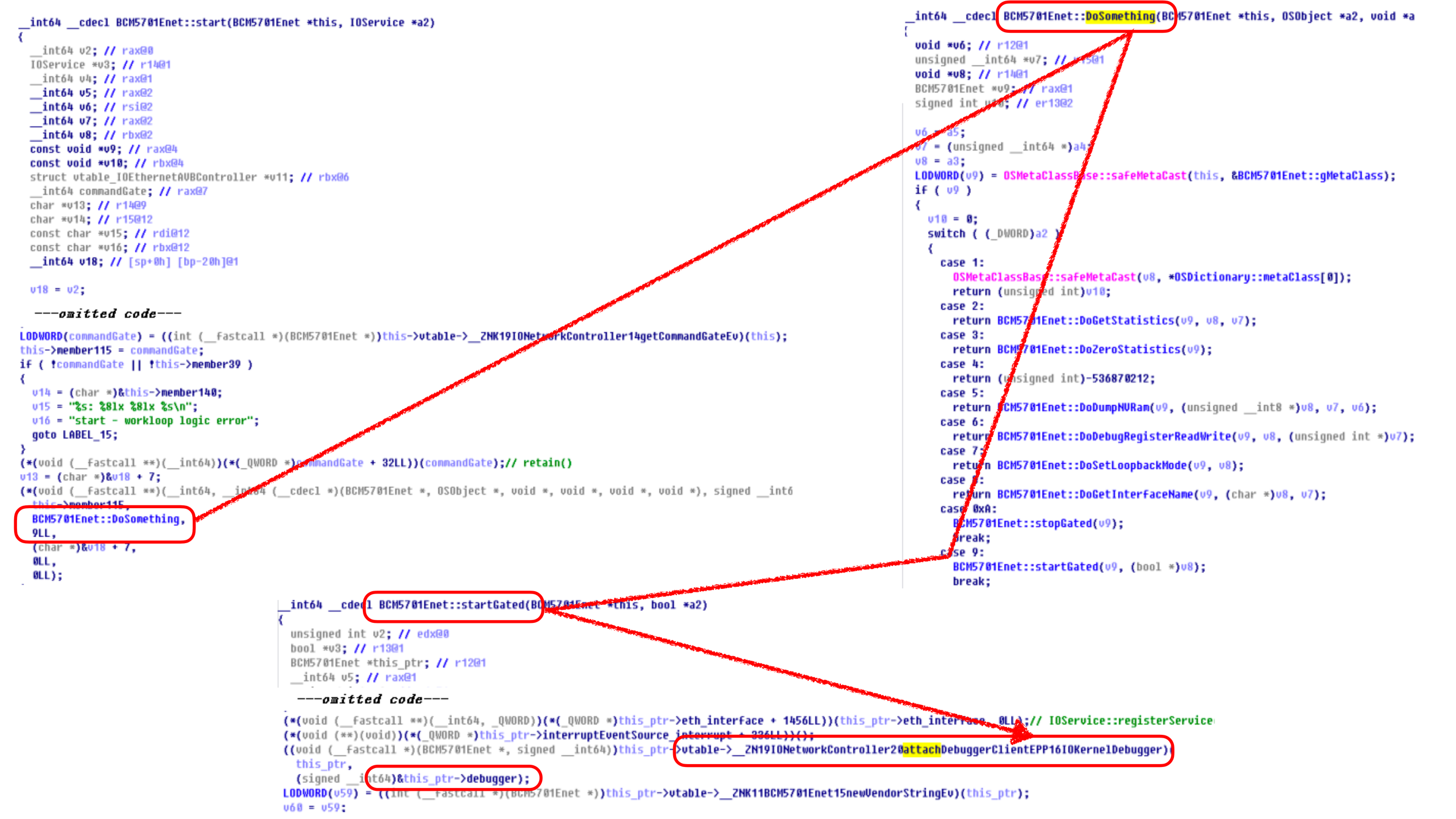
Implement a Kernel Debugger

- Kernel Debugger is working in the polling mode
- For polling mode, we should override the following functions:

```
virtual IOReturn setInputPacketPollingEnable(
                                              IONetworkInterface *    interface,
                                              bool                    enabled );
virtual void     pollInputPackets(
                                  IONetworkInterface *    interface,
                                  uint32_t                maxCount,
                                  IOMbufQueue *           pollQueue,
                                  void *                  context );
virtual IOReturn enable(IOKernelDebugger * debugger);
virtual IOReturn disable(IOKernelDebugger * debugger);
virtual void receivePacket(void * pkt, UInt32 * pktSize, UInt32 timeout);
virtual void sendPacket(void * pkt, UInt32 pktSize);
```

Implement a Kernel Debugger

- Reverse the AppleBCM5701Ethernet extension
  - Step 1: Initialize a kernel debugger object and attach it;
  - Step 2: Enable the polling mode in the function setInputPacketPollingEnable
  - Step 3: Implement the pollInputPackets function
  - Step 4: Implement the enable() and disable() virtual methods in IONetworkController
  - Step 5: Implement the sendPacket() and receivePacket() virtual methods in IONetworkController;


__int64 __cdecl BCM5701Enet::start(BCM5701Enet *this, IOService *a2)
{
__int64 v2; // rax@0
IOService *v3; // r14@1
__int64 v4; // rax@1
__int64 v5; // rax@2
__int64 v6; // rsi@2
__int64 v7; // rax@2
__int64 v8; // rbx@2
const void *v9; // rax@4
const void *v10; // rbx@4
struct vtable_IOEthernetAVBController *v11; // rbx@6
__int64 commandGate; // rax@7
char *v13; // r14@9
char *v14; // r15@12
const char *v15; // rdi@12
const char *v16; // rbx@12
__int64 v18; // [sp+0h] [bp-20h]@1
v18 = v2;
---omitted code---
LODWORD(commandGate) = ((int (__fastcall *)(BCM5701Enet *))this->vtable->__ZNK19IONetworkController14getCommandGateEv)(this);
this->member115 = commandGate;
if (!commandGate || !this->member39)
{
v14 = (char *)&this->member140;
v15 = "%s: %8lx %8lx %s\n";
v16 = "start - workloop logic error";
goto LABEL_15;
}
(*(void (__fastcall **)(__int64))(*(_QWORD *)commandGate + 32LL))(commandGate);// retain()
v13 = (char *)&v18 + 7;
(*(void (__fastcall **)(__int64, __int64 (__cdecl *)(BCM5701Enet *, OSObject *, void *, void *, void *, void *), signed __int6
this->member115,
BCM5701Enet::DoSomething,
9LL,
(char *)&v18 + 7,
0LL,
0LL);
__int64 __cdecl BCM5701Enet::DoSomething(BCM5701Enet *this, OSObject *a2, void *a
{
void *v6; // r12@1
unsigned __int64 *v7; // r15@1
void *v8; // r14@1
BCM5701Enet *v9; // rax@1
signed int v10; // er13@2
v6 = a5;
v7 = (unsigned __int64 *)a4;
v8 = a3;
LODWORD(v9) = OSMetaClassBase::safeMetaCast(this, &BCM5701Enet::gMetaClass);
if (v9)
{
v10 = 0;
switch ((_DWORD)a2)
{
case 1:
OSMetaClassBase::safeMetaCast(v8, *OSDictionary::metaClass[0]);
return (unsigned int)v10;
case 2:
return BCM5701Enet::DoGetStatistics(v9, v8, v7);
case 3:
return BCM5701Enet::DoZeroStatistics(v9);
case 4:
return (unsigned int)-536870212;
case 5:
return BCM5701Enet::DoDumpNVRam(v9, (unsigned __int8 *)v8, v7, v6);
case 6:
return BCM5701Enet::DoDebugRegisterReadWrite(v9, v8, (unsigned int *)v7);
case 7:
return BCM5701Enet::DoSetLoopbackMode(v9, v8);
case 8:
return BCM5701Enet::DoGetInterfaceName(v9, (char *)v8, v7);
case 0xA:
BCM5701Enet::stopGated(v9);
break;
case 9:
BCM5701Enet::startGated(v9, (bool *)v8);
break;
__int64 __cdecl BCM5701Enet::startGated(BCM5701Enet *this, bool *a2)
{
unsigned int v2; // edx@0
bool *v3; // r13@1
BCM5701Enet *this_ptr; // r12@1
__int64 v5; // rax@1
---omitted code---
(*(void (__fastcall **)(__int64, _QWORD))(*(_QWORD *)this_ptr->eth_interface + 1456LL))(this_ptr->eth_interface, 0LL);// IOService::registerService
(*(void (**)(void))(*(_QWORD *)this_ptr->interruptEventSource + 336LL))();
((void (__fastcall *)(BCM5701Enet *, signed __int64))this_ptr->vtable->__ZN19IONetworkController20attachDebuggerClientEPP16IOKernelDebugger)(
this_ptr,
(signed __int64)&this_ptr->debugger);
LODWORD(v59) = ((int (__fastcall *)(BCM5701Enet *))this_ptr->vtable->__ZNK11BCM5701Enet15newVendorStringEv)(this_ptr);
v60 = v59;

```
__int64 __cdecl BCM5701Enet::pollInputPackets(BCM5701Enet *this, void *a2, unsigned int a3, void *a4, void *a5)
{
  void *v5; // r14@1
  unsigned int v6; // er15@1
  BCM5701Enet *v7; // rbx@1
  IOService *v8; // rdi@3
  __int64 result; // rax@4

  v5 = a4;
  v6 = a3;
  v7 = this;
  if ( LOBYTE(this->member123) )
  {
    if ( !LOBYTE(this->members[1962]) )
    {
      v8 = (IOService *)this->member36;
      if ( !v8 || (result = IOService::isInactive(v8), !(_BYTE)result) )
      {
        if ( LOBYTE(v7->members[2121]) )
          result = BCM5701Enet::receivePackets(v7, v6, v5, 0);
      }
    }
  }
  return result;
}
```

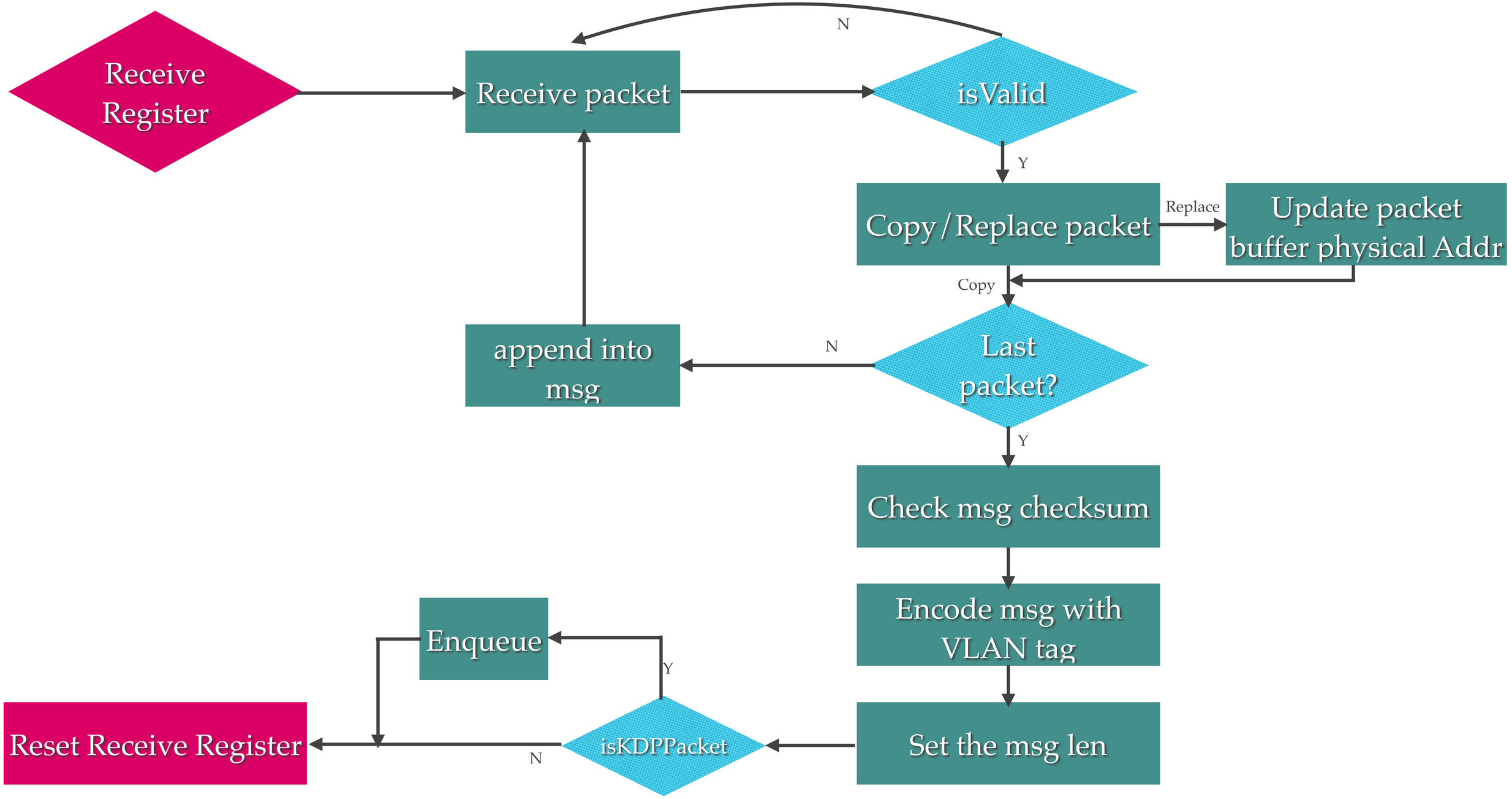
```
__int64 __cdecl BCM5701Enet::receivePacket(BCM5701Enet *this, void *ptk, unsigned int *pktSize
{
  __int64 result; // rax@0
  unsigned int timeout_v; // er15@1
  unsigned int *v6; // r14@1
  char v7; // [sp+8h] [bp-48h]@3
  int v8; // [sp+10h] [bp-40h]@3
  char v9; // [sp+18h] [bp-38h]@4
  int v10; // [sp+20h] [bp-30h]@4

  BYTE5(this->member123) = 1;
  timeout_v = timeout;
  v6 = pktSize;
  *pktSize = 0;
  if ( LOBYTE(this->member123) && BYTE4(this->member123) )
  {
    this->member125 = (__int64)ptk;
    LODWORD(this->member126) = 0;
    clock_get_system_nanotime(&v7, &v8);
    while ( 1 )
    {
      BCM5701Enet::receivePackets(this, 1u, 0LL, 1);
      clock_get_system_nanotime(&v9, &v10);
      result = LODWORD(this->member126);
      if ( (_DWORD)result )
        break;
      if ( (v10 - v8) / 0xF4240u >= timeout_v )
      {
        result = 0LL;
        *v6 = result;
      }
    }
    BYTE5(this->member123) = 0;
```

pollInputPackets ≈ receivePacket ≈ serviceRxInterrupt

```
__int64 __cdecl BCM5701Enet::serviceRxInterrupt(BCM5701Enet *this)
{
  __int64 result; // rax@1

  result = BCM5701Enet::receivePackets(this, 0xFFFFFFFF, 0LL, 0);
  if ( (_BYTE)result )
    LODWORD(result) = (*(int (__fastcall **)(_QWORD, _QWORD))(*(_QWORD *)this->eth_interface + 2176LL))(
                        this->eth_interface,
                        0xFFFFFFFFLL);
  return result;
}
```

```
UInt32 RTL8111::rxInterrupt(IONetworkInterface *interface, uint32_t maxCount, IOMbufQueue *pollQueue, void
    *context)
{
    IOPhysicalSegment rxSegment;
    RtlDmaDesc *desc = &rxDescArray[rxNextDescIndex];
    mbuf_t bufPkt, newPkt;
    UInt64 addr;
    UInt32 opts1, opts2;
    UInt32 descStatus1, descStatus2;
    UInt32 pktSize;
    UInt32 goodPkts = 0;
    UInt16 vlanTag;
    bool replaced;

    while (!((descStatus1 = OSSwapLittleToHostInt32(desc->opts1)) & DescOwn) && (goodPkts < maxCount)) {
        opts1 = (rxNextDescIndex == kRxLastDesc) ? (RingEnd | DescOwn) : DescOwn;
        opts2 = 0;
        addr = 0;
```

```
UInt32 IntelMausi::rxInterrupt(IONetworkInterface *interface, uint32_t maxCount, IOMbufQueue *pollQueue, void *context)
{
    IOPhysicalSegment rxSegment;
    union e1000_rx_desc_extended *desc = &rxDescArray[rxNextDescIndex];
    mbuf_t bufPkt, newPkt;
    UInt64 addr;
    UInt32 status;
    UInt32 goodPkts = 0;
    UInt32 pktSize;
    UInt32 n;
    UInt16 vlanTag;
    bool replaced;

    while (((status = OSSwapLittleToHostInt32(desc->wb.upper.status_error)) & E1000_RXD_STAT_DD) && (goodPkts < maxCount)) {
        addr = rxBufArray[rxNextDescIndex].phyAddr;
        bufPkt = rxBufArray[rxNextDescIndex].mbuf;
        pktSize = OSSwapLittleToHostInt16(desc->wb.upper.length);
```


Receive Register
Receive packet
N
isValid
Y
Copy / Replace packet
Replace
Update packet buffer physical Addr
Copy
Last packet?
N
append into msg
Y
Check msg checksum
Encode msg with VLAN tag
Set the msg len
isKDPPacket
Y
N
Enqueue
Reset Receive Register

sendPacket Function

- it firstly allocate a packet with a data buffer;
- move the send pkt info to the newly allocate buffer and set its length;
- calling the transmitPacket to send the packet;
- calling the transmitKick function to update the related status registers;

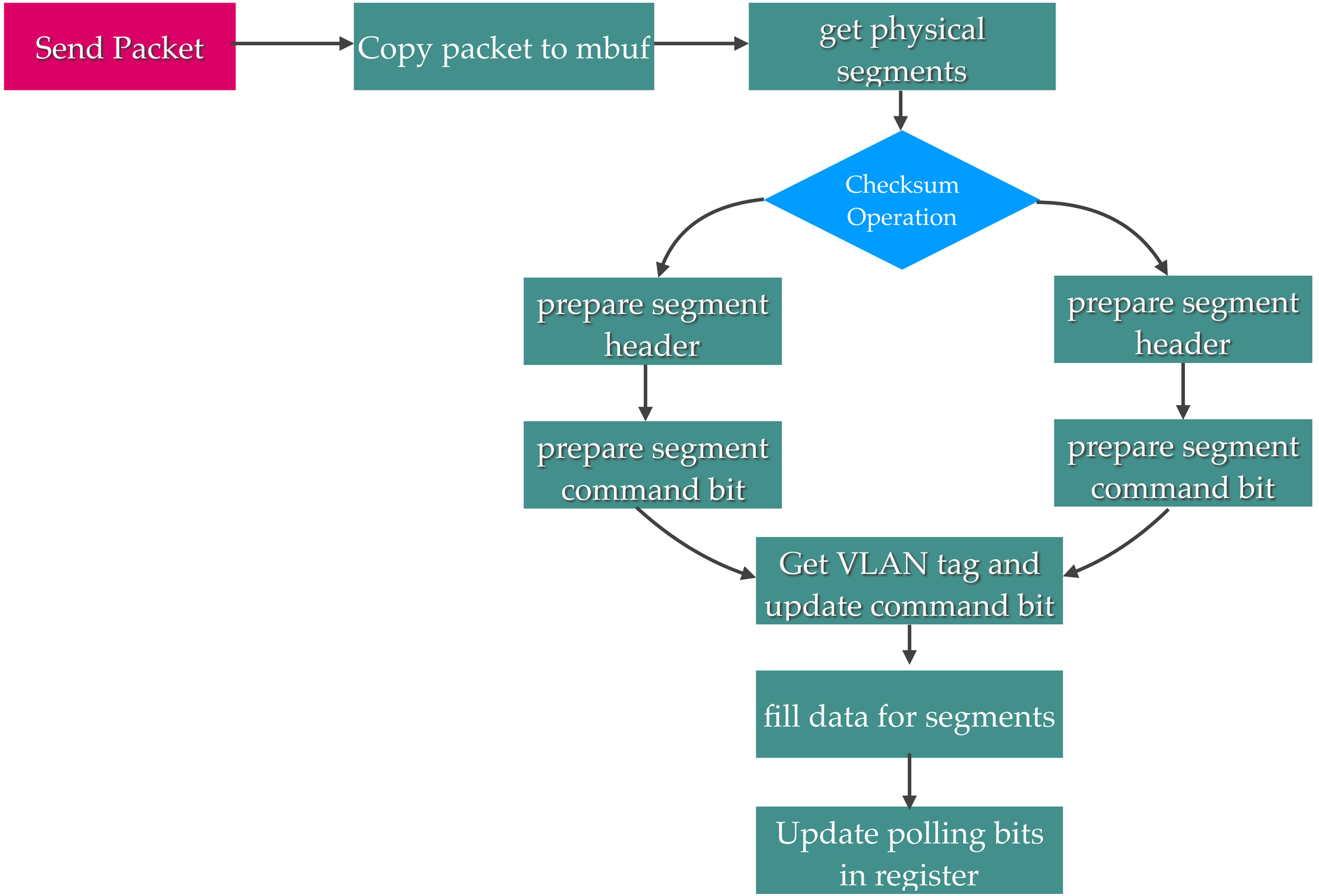
```
LODWORD(v11) = mbuf_data(v5->allocate_packet);
memmove(v11, pkt, v4);
mbuf_setlen(v5->allocate_packet, v4);
mbuf_pkthdr_setlen(v5->allocate_packet, v4);
BCM5701Enet::transmitPacket((#13 *)v5, (IOService *)v5->allocate_packet, 3, 0LL);
BCM5701Enet::transmitKick(v5, 0);
do
{
  BCM5701Enet::serviceTxInterrupt(v5, 1, 0);
  clock_get_system_nanotime(&v15, &v16);
  result = v16;
  if ( v16 >= v14 )
  {
    v12 = v15;
  }
  else
  {
    result = (unsigned int)(result + 1000000000);
    v16 = result;
    v12 = v15-- - 1;
  }
  v8 = HIDWORD(v5->member193);
  v9 = LODWORD(v5->member194);
  if ( (_DWORD)v8 == (_DWORD)v9 )
    break;
  result = 1000 * (v12 - v13) + ((unsigned int)result - v14) / 0xF4240;
}
while ( (unsigned int)result < 0x1388 );
if ( (_DWORD)v8 != (_DWORD)v9 )
{
  v10 = "sendPacket - timeout - packet failed to send";
  goto LABEL_20;
}
```

Reference

```
/*! @function sendPacket
    @abstract Debugger polled-mode transmit handler.
    @discussion This method must be implemented by a driver that supports
    kernel debugging. After a debugger client has been attached through
    attachDebuggerClient(), this method will be called by the debugger
    to send an outbound packet only when the kernel debugger is active.
    This method may be called from the primary interrupt context, and the
    implementation must avoid any memory allocation, and must never block.
    The sendPacket() method in IONetworkController is used as a placeholder,
    it performs no useful action, and should not be called. A driver that
    attaches a debugger client must override this method.
    @param pkt Pointer to a transmit buffer containing the packet to be
        sent on the network.
    @param pktSize The size of the transmit buffer in bytes.
*/

    virtual void sendPacket(void * pkt, UInt32 pktSize);
```

```
/*! @function outputStart
    @abstract An indication to the driver to dequeue and transmit packets
    waiting in the interface output queue.
    @discussion A driver that supports the pull output model must override this
    method, which will be called by a per-interface output thread when a packet
    is added to the interface output queue. In response, driver must verify that
    free transmit resources are available, then dequeue one or more packets by
    calling <code>IONetworkInterface::dequeueOutputPackets()</code>. Packets
    removed from the queue are owned by the driver, and should be immediately
    prepared for transmission. Additional software queueing at the driver layer
    to store the dequeued packets for delayed transmission is highly discouraged
    unless absolutely necessary. If transmit resources are exhausted, the driver
    should quickly return <code>kIOReturnNoResources</code> to force the output
    thread to retry later, otherwise the output thread will continue to call
    this method until the output queue is empty. When driver creates a single
    network interface, this method will execute in a single threaded context.
    However, it is the driver's responsibility to protect transmit resources
    that are shared with other driver threads. To simplify drivers that wish to
    process output packets on their work loop context, the family provides an
    option to force the output thread to always call this method through a
    <code>runAction()</code>. However this can have negative performance
    implications due to extra locking and serializing the output thread against
    other work loop events. Another option that drivers can deploy to
    synchronize against the output thread is to issue a thread stop before
    touching any shared resources. But this should be used sparingly on the
    data path since stopping the output thread can block.
    @param interface The network interface with packet(s) to transmit.
    @param options Always zero.
    @result <code>kIOReturnSuccess</code> on success, output thread will
    continue calling the driver until the output queue is empty.
    <code>kIOReturnNoResources</code> when there is a temporary driver resource
    shortage.
*/
    virtual IOReturn outputStart(
                        IONetworkInterface *    interface,
                        IOOptionBits            options );
```

- RTL8111::outputStart in RealtekRTL8111.cpp or
- IntelMausi::outputStart function in IntelMausiEthernet.cpp


Send Packet
Copy packet to mbuf
get physical segments
Checksum Operation
prepare segment header
prepare segment command bit
prepare segment header
prepare segment command bit
Get VLAN tag and update command bit
fill data for segments
Update polling bits in register

References

- https://blog.quarkslab.com/an-overview-of-macos-kernel-debugging.html
- https://developer.apple.com/library/archive/documentation/Darwin/Conceptual/KernelProgramming/build/build.html
- https://developer.apple.com/documentation/kernel/iokerneldebugger?language=objc
- https://github.com/aerror2/IntelMausiEthernetWithKernelDebugger

Agenda

- Kernel Debugger Overview
- The Introduction of LLDBFuzzer
- Attack Surfaces on Graphic Extensions
- Practice and Demo
- Vulnerabilities Founded
- Implement a Debugger for Hackintosh
- Conclusion

- Introduce the architecture of lldb debugger, and show our debugger fuzz implementation and usage. Also, we introduce two kinds of hidden interfaces in Graphic drivers and show how to fuzz them. Next, we introduce some bugs we have found. Last, we introduce the methodology to implement a kernel debugger base on open source network driver.

- Moony Li
  - E-mail: 411527096@qq.com
  - Twitter: @Flyic
- Lilang Wu
  - E-mail: 574407955@qq.com
  - Twitter: @Lilang_Wu

Questions?