Fuzzing and Exploiting OSX Vulnerabilities for Fun and Profit

Complementary

Active & Passive Fuzzing

What We Will Cover

- **Who We Are**
- **Passive Fuzzing Framework**
- **Exploit to Root**

TREND MICRO™


Who We Are

Moony Li

- @Flyic
- 7 years security
- Sandcastle
- Deep Discovery
- Exploit Detection
- Mac/Windows Kernel
- Android Vulnerability

TREND MICRO

Jack Tang

- @jacktang310
- 10+ years security
- Browser
- Document
- Mac/Windows Kernel
- Virtualization Vulnerability

TREND MICRO

TREND MICRO™

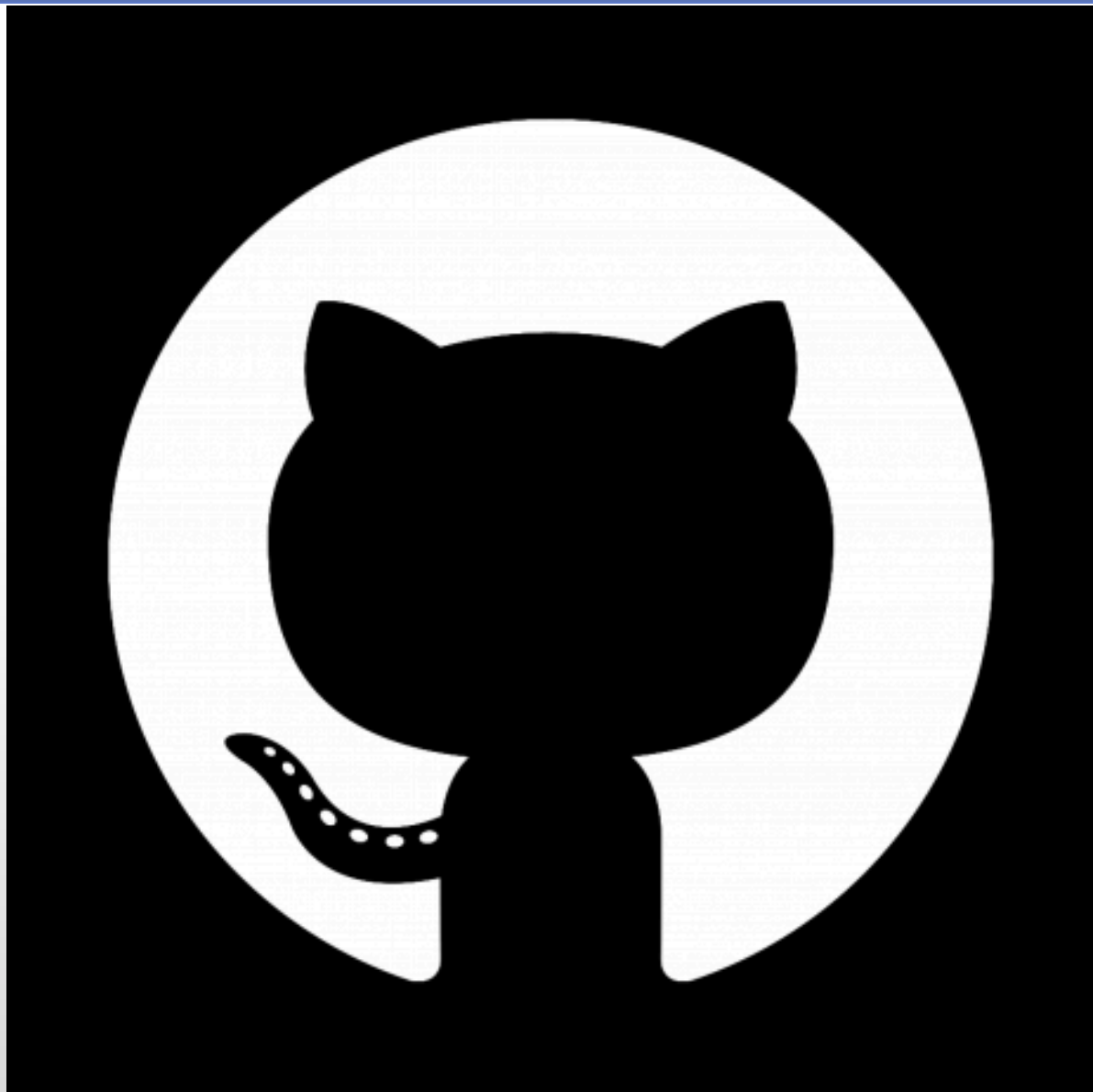
CVE Credits

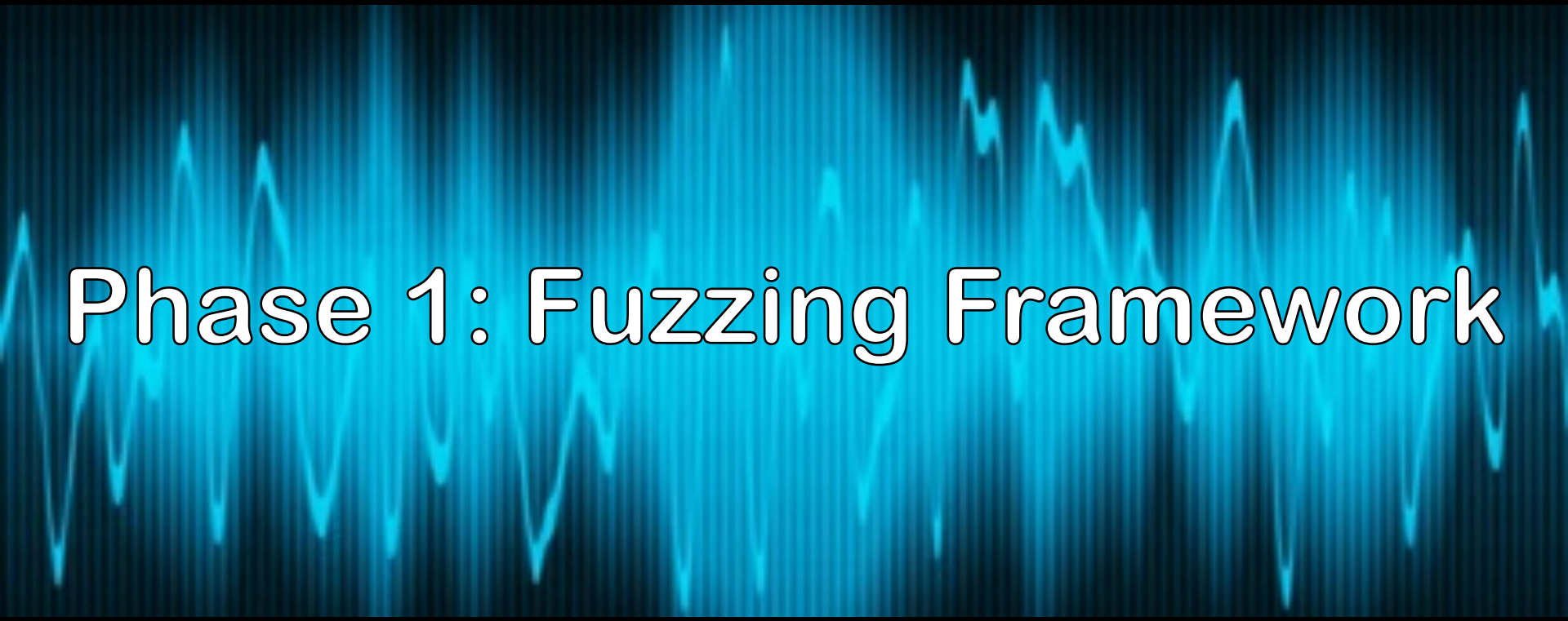
CVE-2015-3787, CVE-2015-5867, CVE-2015-7021,CVE-2015-7020, CVE-2016-1716,ZDI-CAN-3536,ZDI-CAN-3558, ZDI-CAN-3598,ZDI-CAN-3596,ZDI-CAN-3603,CVE-2015-7067, CVE-2015-7076,CVE-2015-7106,CVE-2015-7109,CVE-2016-1718, CVE-2016-1747,CVE-2016-1749,CVE-2016-1753, ZDI-CAN-3693, ZDI-CAN-3694, CVE-2016-1795, CVE-2016-1808, CVE-2016-1810, CVE-2016-1817, CVE-2016-1820, CVE-2016-1798, CVE-2016-1799, CVE-2016-1812, CVE-2016-1814, CVE-2016-1818, CVE-2016-1816, CVE-2016-4648, CVE-2016-4699, CVE-2016-4700, CVE-2016-4750

TREND MICRO

Github Access

https://github.com/SilverMoonSecurity




Phase 1: Fuzzing Framework

What We Will Cover

- Comparing Approaches
- Our Approach/ Consideration
- Implementation
- Best Practice

TREND MICRO™

Comparing Other Approaches

Methods	Disadvantage	Notes
Traditional Fuzz Choice the IOKit service name they want to test. Pour fuzzing data into the IOKit usermode API (e.g. IOConnectCallMethod)	Call sequence dependency	*AppleCamIn (OpenDevice, PowerOnCamera…)*
	Input data dependency	*AppleHDAEngineInput(input as user mode buffer pointer)*
	Timing dependency	*IOHDIXHDDriveOutKernel(mount dmg)*
Code Review	Un-scalable	*Depend on Human experience*
	Cost effort for Reverse Engineering	*so many IOKit services and userclient.*

Our Approach | Consideration
Interception & Poison

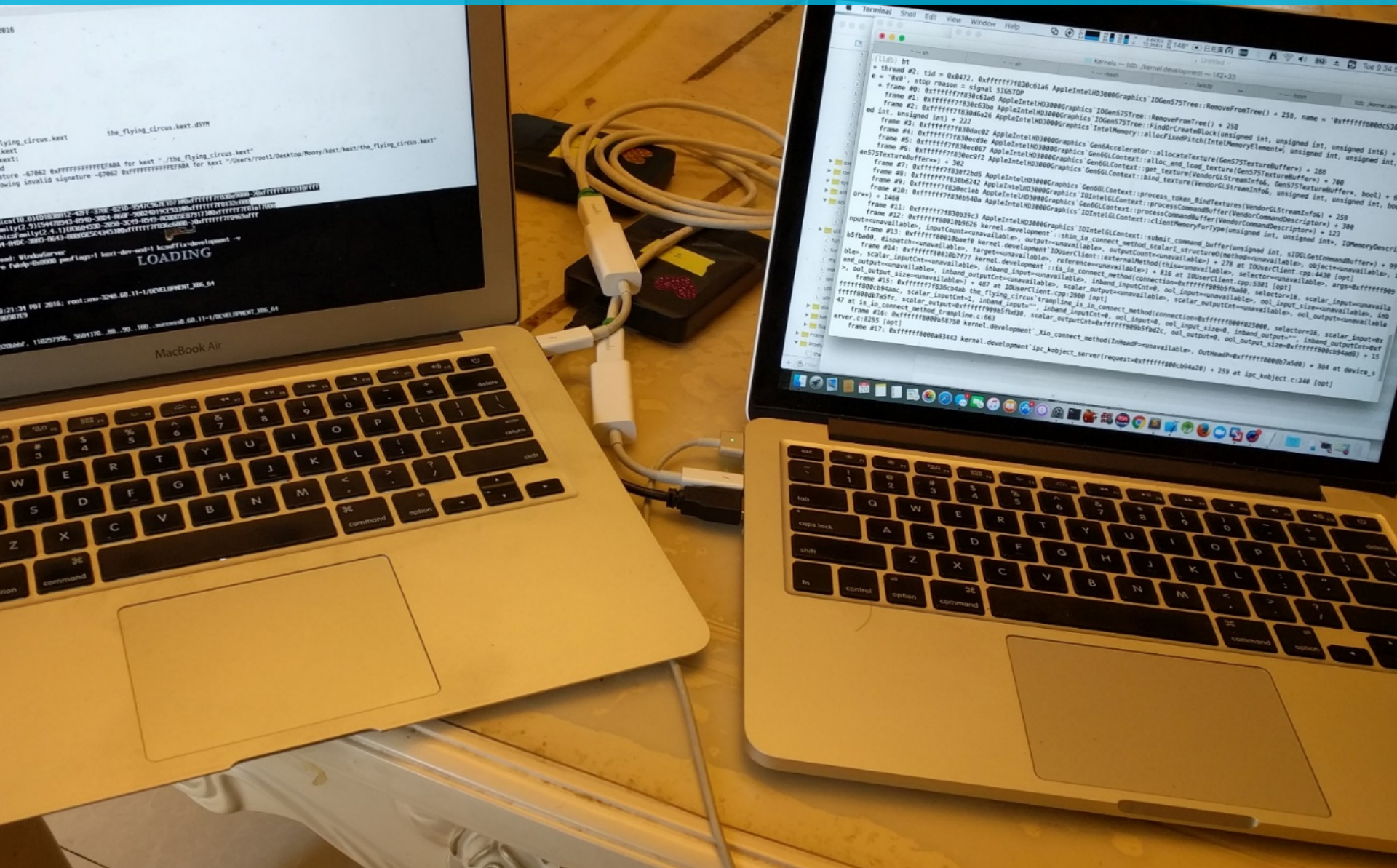
TREND MICRO™

Are they similar☺

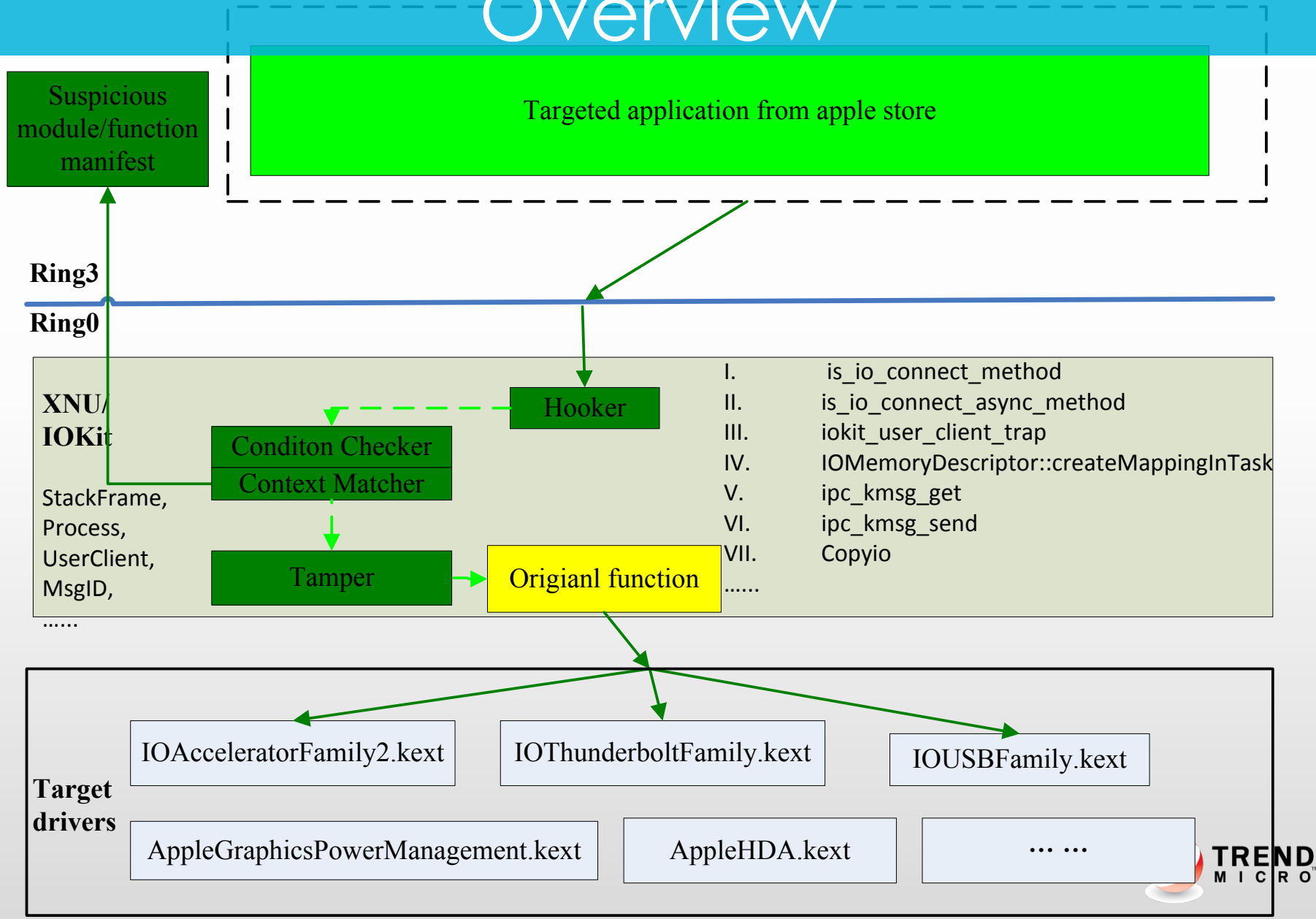
Comparison	River Dam	Passive Fuzz
Basic Function	Stream intercept	Execution intercept
	Up Stream	User mode data
	Down Stream	Kernel mode data
Create Chaos	Poisoning upstream	Fuzzing user data
	Downstream fish die	Kernel crash
	Track the poison's origin	Reproduce
	……	……

TREND MICRO

Just Like This …



Implementation – Architecture Overview



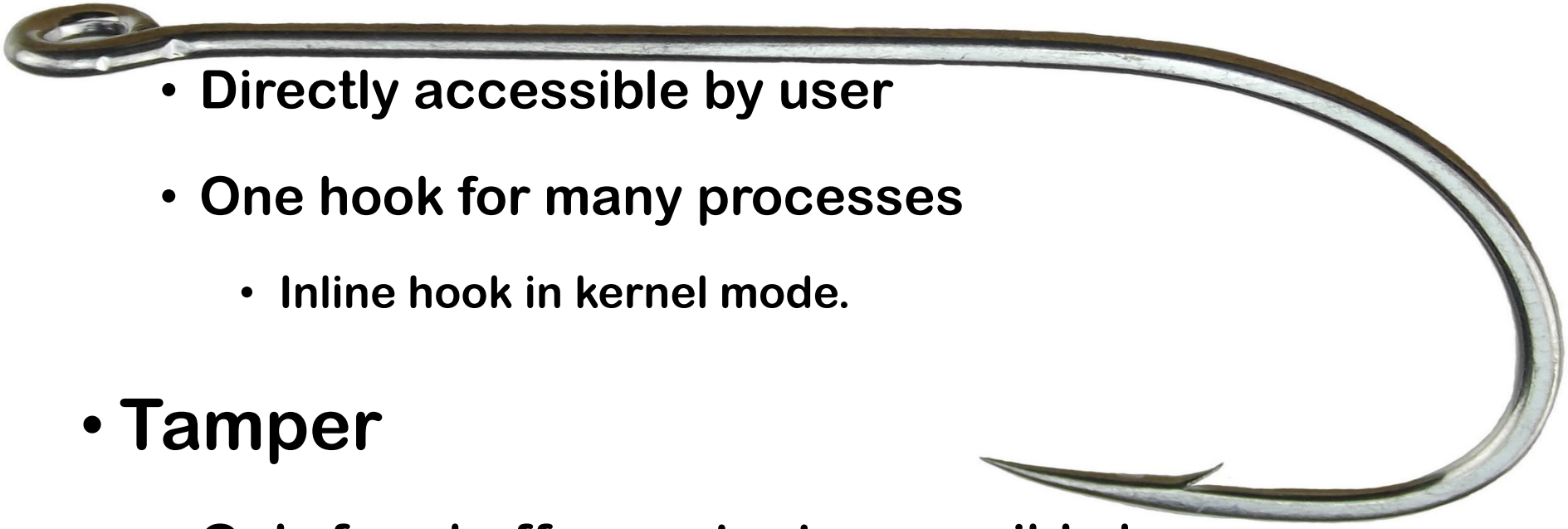
Implementation – Pseudo Code

```
TargetAPI(params):

   //Call Original_TargetAPI(params)

   if (matchWhitelistParameter(params))  goto _exit();

   if !(matchStackFrame() &&
matchBlacklistParameter(params))

         goto _exit;

   if (random()) {record(params); fuzz(params);}

   Call Original_TargetAPI(params);

   if (matchContext(params))  alert;
```

TREND MICRO

Implementation – Hook & Tamper

- **Hooker**
  - **Directly accessible by user**
  - **One hook for many processes**
    - **Inline hook in kernel mode.**
- **Tamper**
  - **Only fuzz buffer content accessible by user**
    - **e.g. Inband_input, scalar_input, ool_input**
    - **NOT size! (bypass getTargetAndMethodForIndex check)**



Implementation – Hook & Tamper (Snippet – Unreported Crash)

0xffffff80c0c7b900 0xffffff7fa96b540a AppleIntelHD3000Graphics`IOIntelGLContext::clientMemoryForType(unsigned int, unsigned int*, IOMemoryDescriptor**) + 0x5bc

0xffffff80c0c7b950 0xffffff7fa96b39c3 AppleIntelHD3000Graphics`IOIntelGLContext::submit_command_buffer(unsigned int, sIOGLGetCommandBuffer*) + 0x63

0xffffff80c0c7b980 0xffffff80276b9626 ::shim_io_connect_method_scalarI_structureO(IOExternalMethod *, IOService *, const io_user_scalar_t *, mach_msg_type_number_t, char *, IOByteCount *)((IOExternalMethod *) method = <>, , (IOService *) object = <>, , (const io_user_scalar_t *) input = <>, , (mach_msg_type_number_t) inputCount = <>, , (char *) output = <register r10 is not available>, , (IOByteCount *) outputCount = <register r11 is not available>,)

0xffffff80c0c7b9e0 0xffffff80276baef0 IOUserClient::externalMethod(unsigned int, IOExternalMethodArguments*, IOExternalMethodDispatch*, OSObject*, void*)((IOUserClient *) this = <>, , (uint32_t) selector = <>, , (IOExternalMethodArguments *) args = 0xffffff80c0c7ba00, (IOExternalMethodDispatch *) dispatch = <>, , (OSObject *) target = <>, , (void *) reference = <>,)

0xffffff80c0c7bb20 0xffffff80276b7f77 ::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 *)((io_connect_t) connection = 0xffffff80c0c7ba60, (uint32_t) selector = 16, (io_user_scalar_t *) scalar_input = <>, , (mach_msg_type_number_t) scalar_inputCnt = <>, , (char *) inband_input = <>, , (mach_msg_type_number_t) inband_inputCnt = 0, (mach_vm_address_t) ool_input = <>, , (mach_vm_size_t) ool_input_size = <no location, value may have been optimized out>, , (char *) inband_output = <no location, value may have been optimized out>, , (mach_msg_type_number_t *) inband_outputCnt = <no location, value may have been optimized out>, , (io_user_scalar_t *) scalar_output = <>, , (mach_msg_type_number_t *) scalar_outputCnt = <no location, value may have been optimized out>, , (mach_vm_address_t) ool_output = <>, , (mach_vm_size_t *) ool_output_size = <>,)

0xffffff80c0c7bcd0 0xffffff7fa9cd34ab trampline_is_io_connect_method((io_connect_t) connection = 0xffffff8035637000, (uint32_t) selector = 16, (io_user_scalar_t *) scalar_input = 0xffffff80331a4dcc, (mach_msg_type_number_t) scalar_inputCnt = 1, (char *) inband_input = 0xffffff80331a4dd8 "", (mach_msg_type_number_t) inband_inputCnt = 0, (mach_vm_address_t) ool_input = 0, (mach_vm_size_t) ool_input_size = 0, (char *) inband_output = 0xffffff8035805600 "", (mach_msg_type_number_t *) inband_outputCnt = 0xffffff80358055fc, (io_user_scalar_t *) scalar_output = 0xffffff80c0c7bd30, (mach_msg_type_number_t *) scalar_outputCnt = 0xffffff80c0c7bd2c, (mach_vm_address_t) ool_output = 0, (mach_vm_size_t *) ool_output_size = 0xffffff80331a4df8)

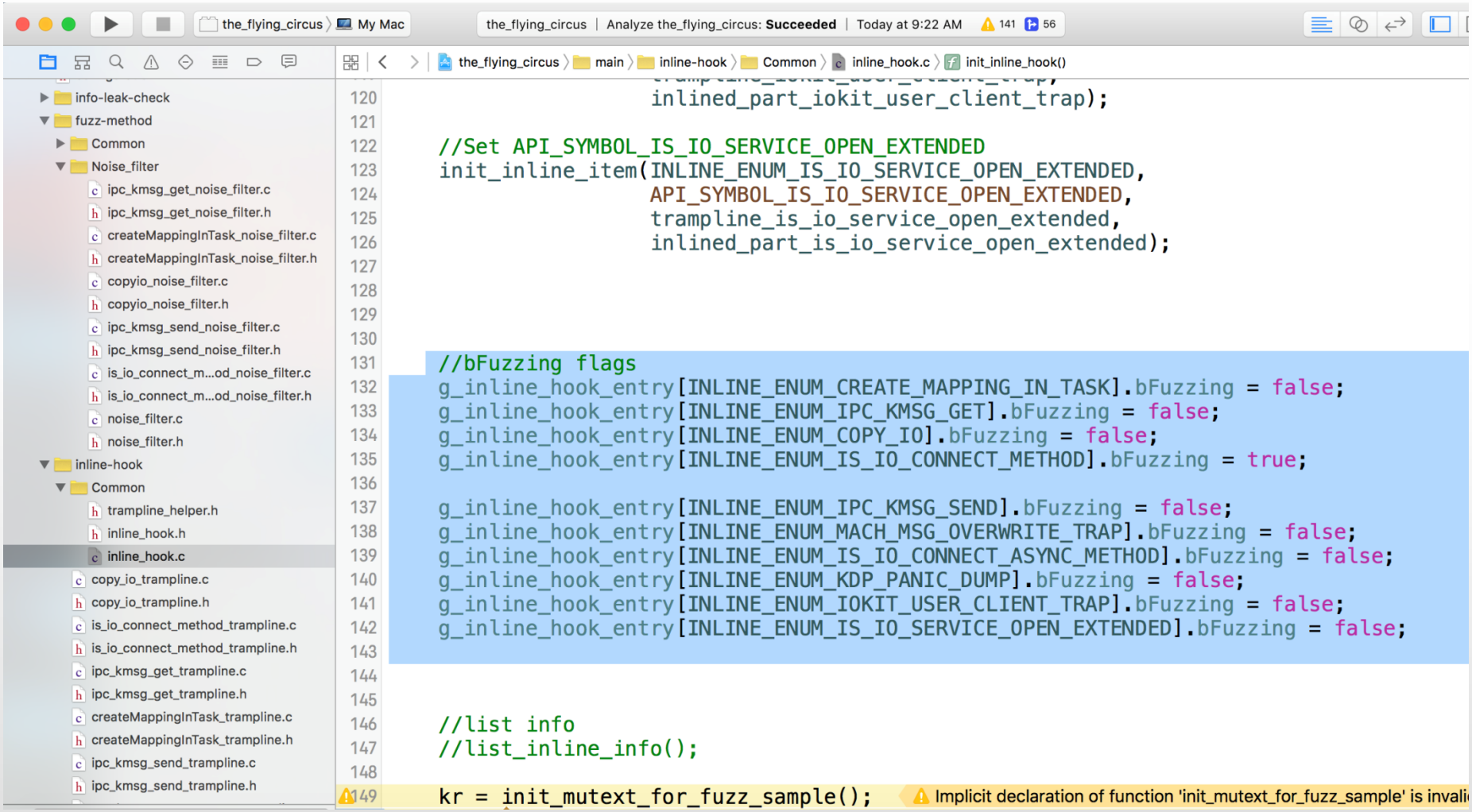
0xffffff80c0c7bde0 0xffffff8027158750 _Xio_connect_method((mach_msg_header_t *) InHeadP = <>, , (mach_msg_header_t *) OutHeadP = 0xffffff80358055d0)

TREND MICRO

Implementation – Hook Summary

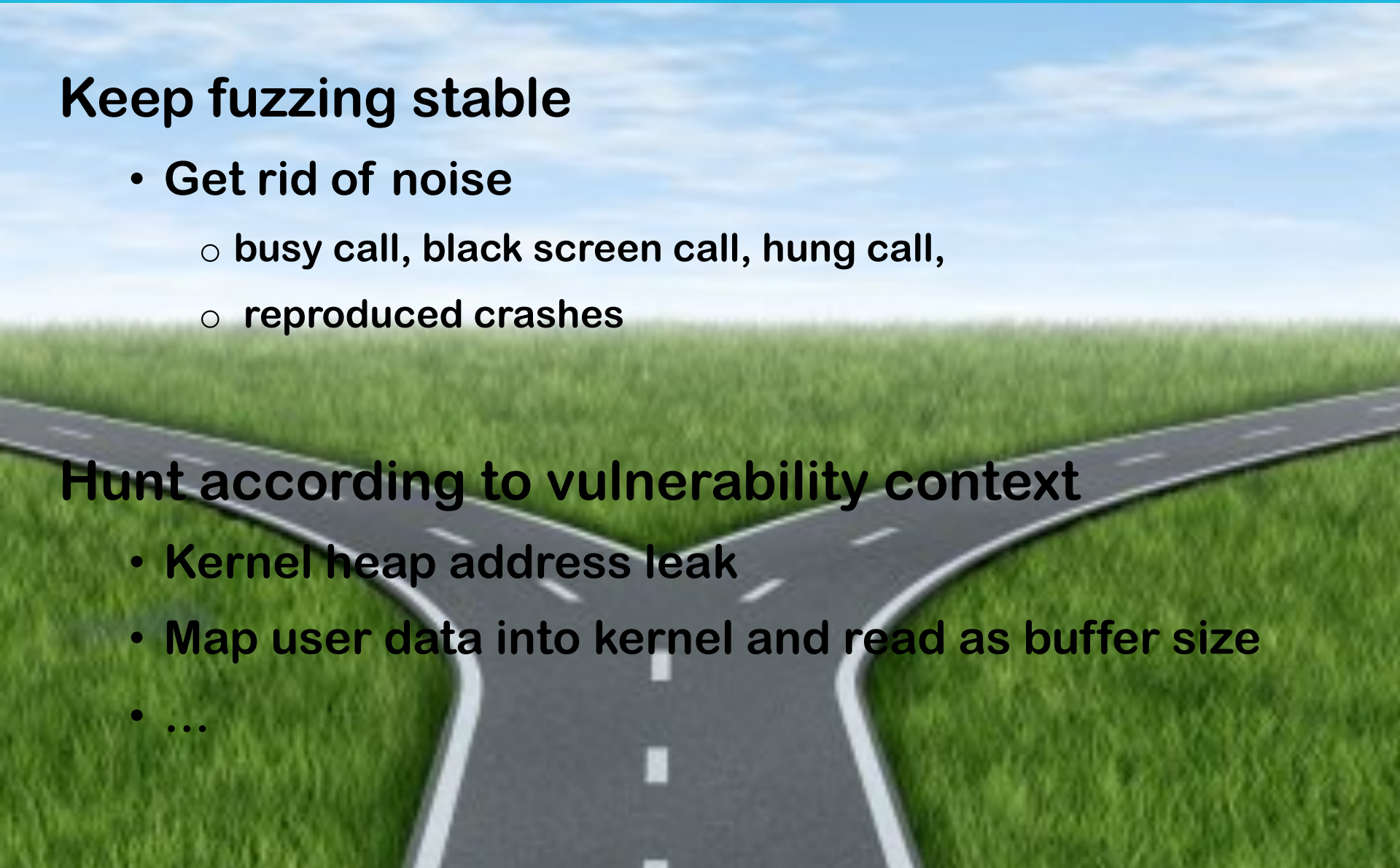
- (Driver interface)is_io_connect_method
- (Driver interface)is_io_connect_async_method
- (Kernel)iokit_user_client_trap
- (Kernel)IOMemoryDescriptor::createMappingInTask
- (Mach Msg)ipc_kmsg_get
- (Mach Msg)ipc_kmsg_send
- (General IO)Copyio
- …

Implementation – Hook Summary Sinnpet



TREND MICRO

Implementation – Why Condition Checker?



Keep fuzzing stable

- **Get rid of noise**
  - **busy call, black screen call, hung call,**
  - **reproduced crashes**

Hunt according to vulnerability context

- **Kernel heap address leak**
- **Map user data into kernel and read as buffer size**
- **…**

Implementation – Dimension of Condition 1/3

- &&, ||, *(wild match), white(black)

- Process
  - User id (root/Non-root)
  - Process Name (e.g. Safari, RCE, sandbox-evasion)
- Module
  - Module Name
- Function
  - Symbol Name/Address
  - Offset range

Implementation – White Listing Sample

```
//Config for mac prodetail_control_entry_t g_white_listing_detail_control[] ={    //
procName,uid,driverBundleName, driverClassName, selFunctionNO        //"*",
0,"*","*",ANY_MATCH_INTEGER,#if 0    //Reported or collected yet:    //
{"*",PROCESS_UID_ANY_INTEGER,"*","AGPMClient",7312},    //
{"*",PROCESS_UID_ANY_INTEGER,"*", "nvDeviceTesla", 5},    //
{"*",PROCESS_UID_ANY_INTEGER,"*", "NV2DContextTesla", 17},    //
{"*",PROCESS_UID_ANY_INTEGER,"*","IONVSurfaceTesla",10}, //
{"*",PROCESS_UID_ANY_INTEGER,"*","IOHDIXHDDriveOutKernelUserClient",2},
{"*",PROCESS_UID_ANY_INTEGER,"*","IGAccelSharedUserClient",1},//crash-24
{"*",PROCESS_UID_ANY_INTEGER,"*","AccelSurface",16},//crash-23
{"*",PROCESS_UID_ANY_INTEGER,"*",OBJECT_CLASS_NAME_NO_FOUND,16},
{"*",PROCESS_UID_ANY_INTEGER,"*","HD",2},//
crash-21    //"*",PROCESS_UID_ANY_INTEGER,"*","Accel",2,//
crash-28    //"*",PROCESS_UID_ANY_INTEGER,"*","IG",2,//
crash-28    //"*",PROCESS_UID_ANY_INTEGER,"*","Con",2,//crash-28
"*",PROCESS_UID_ANY_INTEGER,"*","IGAccelSharedUserClient",0,//crash-29
"*",PROCESS_UID_ANY_INTEGER,"*","IOThunderboltFamilyUserClient",22,//
crash-30    //"*",PROCESS_UID_ANY_INTEGER,"*","IG",ANY_MATCH_INTEGER,    //"*",PROCE
SS_UID_ANY_INTEGER,"*","Accel",ANY_MATCH_INTEGER,    //"vm",PROCESS_UID_ANY_INT
EGER,"*","*",ANY_MATCH_INTEGER,    //"*",PROCESS_UID_ANY_INTEGER,"*","vm",ANY_MA
TCH_INTEGER,    "sandbox",PROCESS_UID_ANY_INTEGER,"*","*",ANY_MATCH_INTEGER,
"dog",PROCESS_UID_ANY_INTEGER,"*","*",ANY_MATCH_INTEGER,    //
{"WindowServer",PROCESS_UID_ANY_INTEGER,"*","AccelSurface",16},//
crash-23    //"*",PROCESS_UID_ANY_INTEGER,"*","SMC",ANY_MATCH_INTEGER,    //"window
server",PROCESS_UID_ANY_INTEGER,"*","*",ANY_MATCH_INTEGER,};
```

TREND MICRO

Implementation – Dimension of Condition 2/3

- Data
  - is_address_RWX
  - Copy direction(in/out)
  - Kernel or User space (SMAP noise)
- Call-Stack
  - Function ret
  - Stack Level (from bottom to top)
  - Level range[,]

Implementation – Stack Frame Sample

```
stack_match_item_t stack_matcher_for_copyio[]={
        //If any item in list match, then match
    //{routineName, cache}, routineAddress, offSetFrom, offsetTo, levelLow, levelHigh

{{"_shim_io_connect_method_scalarI_scalarO",STACK_ANY_INTEGER},STACK_ANY_INTEGER,0, 0xC120-0xB8B0, STACK_ALL_LEVEL_RANGE},

{{"_shim_io_connect_method_scalarI_structureO",STACK_ANY_INTEGER},STACK_ANY_INTEGER,0, 0xDB94-0xD5C0, STACK_ALL_LEVEL_RANGE},

{{"_shim_io_connect_method_scalarI_structureI",STACK_ANY_INTEGER},STACK_ANY_INTEGER,0, 0xEA97-0xE490, STACK_ALL_LEVEL_RANGE},

{{"_shim_io_connect_method_structureI_structureO",STACK_ANY_INTEGER},STACK_ANY_INTEGER,0, 0xF588-0xF270, STACK_ALL_LEVEL_RANGE},


{{"_is_io_connect_method",STACK_ANY_INTEGER},STACK_ANY_INTEGER,0,
0xb2a9-0xaf10,STACK_ALL_LEVEL_RANGE},
                }
```

TREND MICRO

Dimension of condition 3/3

- Misc
  - Mach_msg
    - msg subsystem id…
  - Userclient
    - serviceName,ClassName,selector…

Implementation – UserClient Sample

```
detail_control_entry_t g_white_listing_detail_control[] ={
   // procName,uid,driverBundleName, driverClassName, selFunctionNO
//{"*",PROCESS_UID_ANY_INTEGER,"*","AGPMClient",7312},,
   {"*",PROCESS_UID_ANY_INTEGER,"*","IGAccelSharedUserClient",1},//crash-24
   {"*",PROCESS_UID_ANY_INTEGER,"*","AccelSurface",16},//crash-23
   {"*",PROCESS_UID_ANY_INTEGER,"*",OBJECT_CLASS_NAME_NO_FOUND,16},
   {"*",PROCESS_UID_ANY_INTEGER,"*","HD",2},//crash-21
   {"*",PROCESS_UID_ANY_INTEGER,"*","IX",2},//crash-21
   "*",PROCESS_UID_ANY_INTEGER,"*","AGPM",7312,//crash-11
   "*",PROCESS_UID_ANY_INTEGER,"*","IGAccelGLContext",2,//crash-28
```

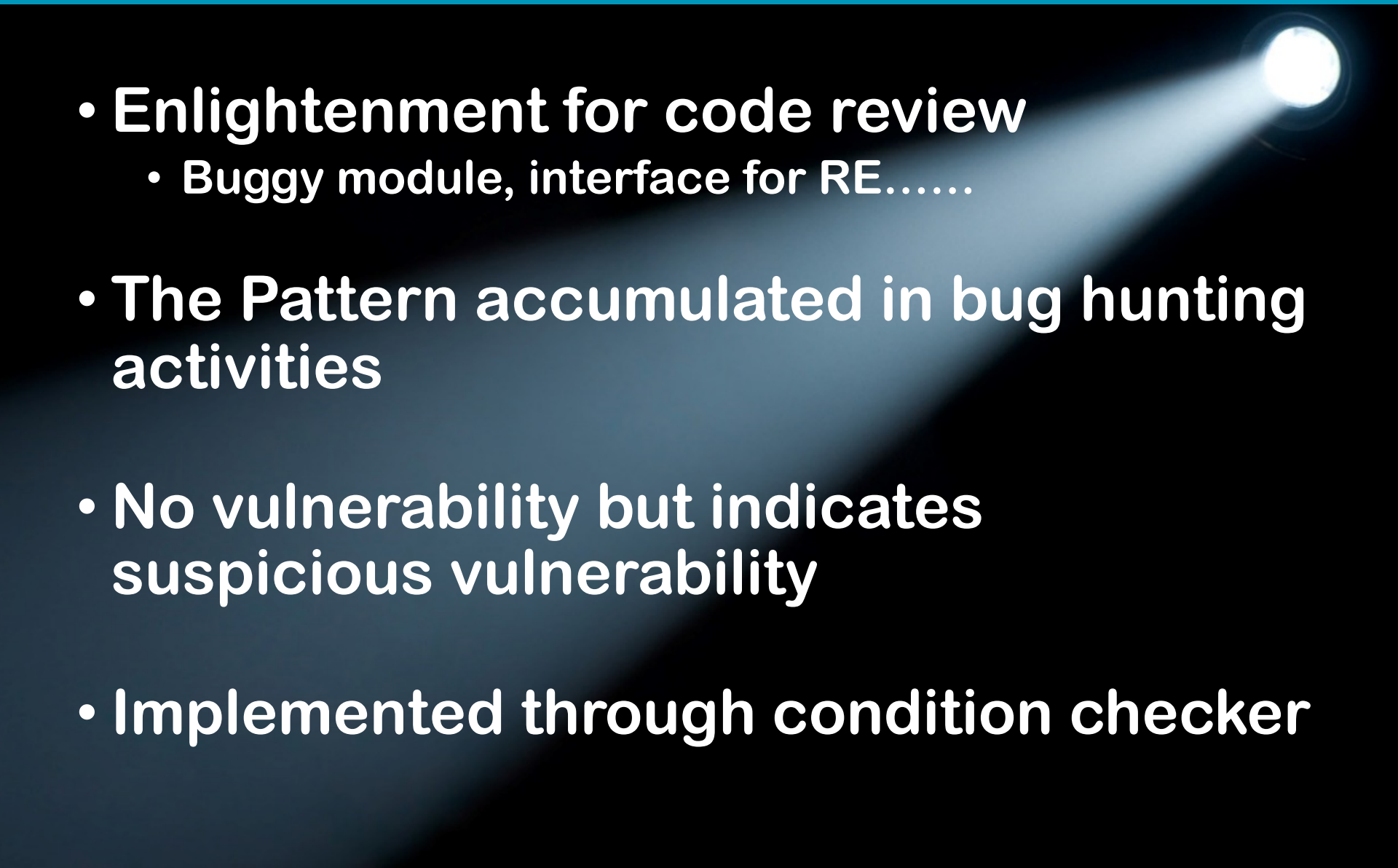
Implementation – Mach-msg Sample

```
#define KMSG_IOKIT_SUBSYSTEM_RANGE  0xAF0, 0x0B47

detail_control_entry_for_ipc_kmsg_send_t g_black_listing_detail_control_foripc_kmsg_send[]
={

//procName,uid,msg_id_from, msg_id_to, routineName, addr, addr_offset_from, addr_offset_to

"chrome",PROCESS_UID_ANY_INTEGER,
KMSG_IOKIT_SUBSYSTEM_RANGE,"__Xio_connect_method",KMSG_ADDR_OFFSET_ANY_RANGE,
KMSG_LEAVING,};
```

- #define KMSG_IOKIT_SUBSYSTEM_RANGE 0xAF0, 0x0B47
- #define KMSG_MACH_VM_SUBSYSTEM_RANGE 0x12C0, 0x12D4
- #define KMSG_MACH_PORT_SUBSYSTEM_RANGE 0xC80, 0x0CA4
- #define KMSG_MACH_HOST_SUBSYSTEM_RANGE 0xC8, 0xE4
- #define KMSG_HOST_PRIV_SUBSYSTEM_RANGE 0x190, 0x1AA
- ……

TREND MICRO

Implementation – What Is Context?



- **Enlightenment for code review**
  - **Buggy module, interface for RE……**
- **The Pattern accumulated in bug hunting activities**
- **No vulnerability but indicates suspicious vulnerability**
- **Implemented through condition checker**

Implementation – Context Sample

- Some IOKit related memory corruption vulnerabilities may happen in the following context:
  - Call IOMemoryDescriptor :: createMappingInTask to mapping user mode buffer space to kernel mode.
  - Read a value from the buffer and use it as a size to read or write a buffer.
- Some kernel information leak vulnerability may happen in the following context:
  - The output buffer's content has 0xFFFFFF prefix.

Best Practice – For Passive 1/3

- **Fuzzing Source:**
  - **Multiple applications**
    - **AppStore (MMORPG games, FaceTime,USB hardisk, BlueTooth, Wifi, VM,DirectX…)**
    - **Virus Total, Apple OpenSource UT, github sample code**
  - **Combination of rich kind of fuzzing source**
    - **Active fuzzing, Python watchdog, browsing WebGL**
- **Fuzzing Stability:**
  - **Bypass active hang, black screen, reproduced cases using condition checker(nvTestIaSurfaceTesla, IGAccelGLContext, IGAccelSurface…)**

TREND MICRO

Best Practice – For Passive 2/3

- **Reproduction:**
  - **Log through network**
  - **Log to NVRAM? Log to memory and kdp_panic_dump callback?**
  - **Core dump server**
    - **sh-3.2# nvram boot-args=”pmuflags=1 debug=0xd44 kext-dev-mode=1 kcsuffix=development –v _panicd_ip=10.64.80.106”**
  - **Thunderbolt+fwkdp+lldb**
- **Automation**
  - **kdp_panic_dump callback+dump+reboot?**
  - **VM(Vmware fusion, Qemu…) revert?**

TREND MICRO

Best Practice – For Passive 3/3

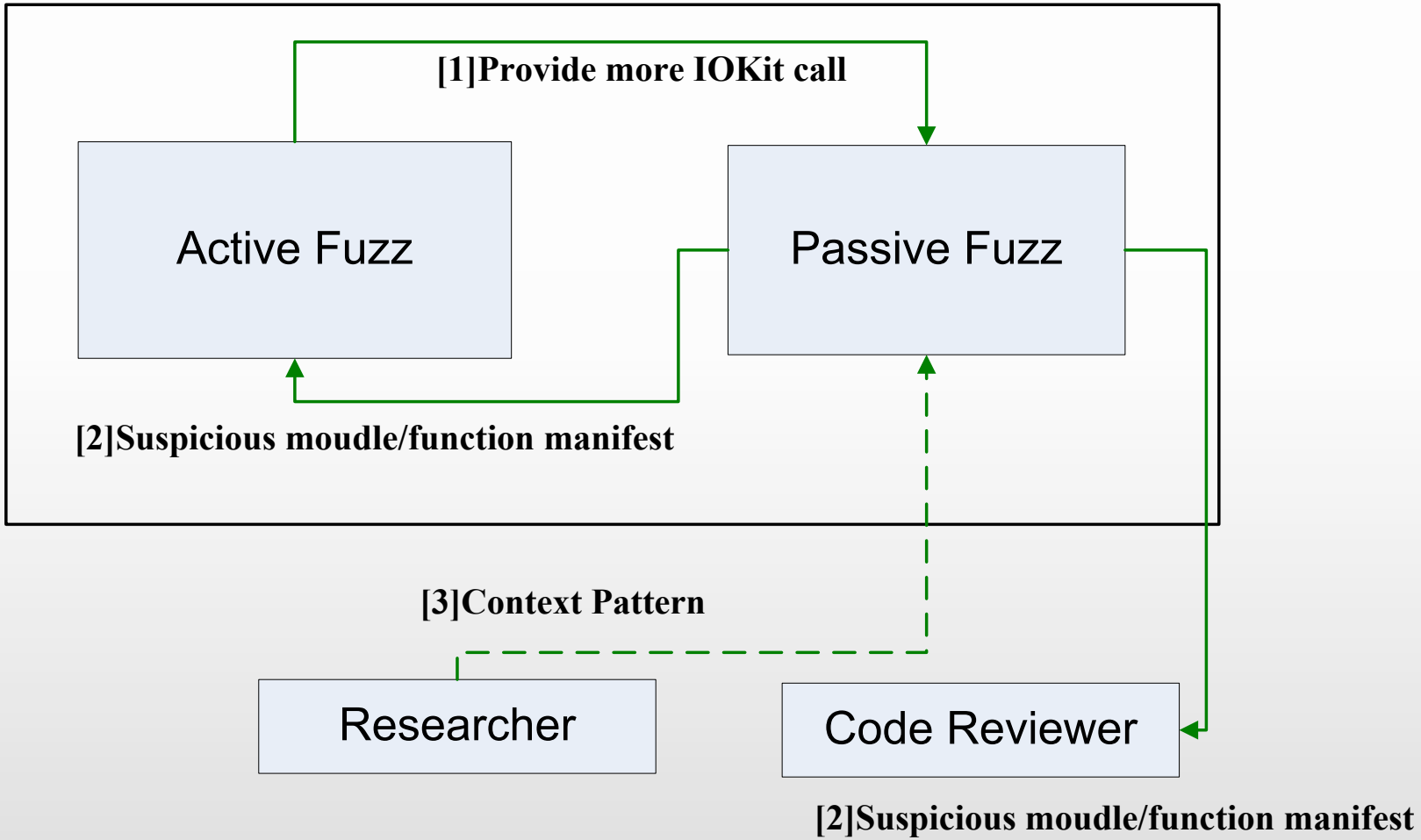
Miscellaneous Tips

- Occasional fuzz activities recommended
- Normal program running – sudden fuzz
- Keep OS version updated with latest KDK

Best Practice – How to complementary passive fuzzing?

TREND MICRO™

Best Practice – For Active Fuzz 1/4



TREND MICRO

Best Practice – For Active 1/4

- **Statistical data from passive fuzz**
  - **Accessible user client, non-root, non-sandbox, crash frequencies, crash types(UAF, OOB)…**
- **Contact API in lower level to bypass unnecessary checks (if possible)**
  - **Resolve Mac-O format, find symbol, hard-code offset**
  - **e.g. IOServiceGetMatchingServices(mach_port_t, CFDictionaryRef, io_iterator_t *)**

->io_service_get_matching_services_bin(mach_port_t, char*,int, void*)

TREND MICRO

Best Practice – For Active 2/4

- **Reproduce:**
  - **No need to record your random call/random parameter - just record your pseudorandom number seed (E.g. Mt19937-64/Mt19937-32)**
  - **The kernel crash is easily reproduced, no matter how randomly you call API and fuzz parameters**

Best Practice – For Active 2/4
Pseudo code for reproduce

```
seed = CrahsedCases[i]

for(randomSelect every kernel module)

        for(randomSelect every interface)

            APICall(  randomByte() )
```

Best Practice – For Active 3/4
Pseudo code for active fuzzing

```
for (every fuzz session)
    seed = generate()
    sendSeedOutside()
    for(randomSelect every kernel module)
      for(randomSelect every interface)
        APICall( randomByte() )
```

TREND MICRO

Best Practice – For Active 4/4

- **Reproduce:**
  - **Web seed server for multiple connections**
  - **One pseudorandom seed for a relatively short fuzz cycle**

TREND MICRO™

Exploit to Root

What We Will Cover

- **Security Mitigation**
  - **Exploit Methods**
- **Exploit Practice**

TREND MICRO™

Security Mitigation

- ~~SIP (System Integrity Protection)~~
- KALSR(e.g. PEGASUS CVE-2016-4655)
- SMAP
- SMEP

https://speakerdeck.com/marcograss/dont-trust-your-eye-apple-graphics-is-compromised

Exploit Methods

- **Bypass KASLR**
  - **Using vulnerability to leak kernel code segment address in run time**
  - **Build up payload with leaked API**

  (e.g. thread_exception_return)
- **Bypass SMAP**
  - **Using vulnerability to leak kernel heap address**
  - **Build up ROP chain in kernel heap**
  - **Kernel heap address would be needed**
- **Bypass SMEP**
  - **Using vulnerability to execute RIP in kernel**
  - **Execute ROP chain to disable SMEP/SMAP**
  - **Payload execution**

Tips of Heap FengShui OSUnerializeXML()

- **The OSX/iOS hacking guru Stefan Esser (@i0n1c) propose *OSUnserializeXML* is a good way in SyScan 2012**

```
<plist version="1.0">
<dict>
    <key>ThisIsOurData</key>
    <array>
        <data>VGhpcyBJcyBPdXIgRGF0YSB3aXRoIGEgTlVMPgA8+ADw=</data>
        <data format="hex">00112233445566778899aabbccddeeff</data>
        <data>...</data>
    </array>
</dict>
</plist>
```

https://reverse.put.as/wp-content/uploads/2011/06/SyScan2012_StefanEsser_iOS_Kernel_Heap_Armageddon.pdf

However…

- **In most cases, the *OSDictionary* allocated by OSUnserializeXML will be freed by *OSObject::release* in one system call**

```
lea     rdx, [rbp+var_30] ; OSString **
mov     rdi, rbx          ; char *
mov     rsi, r14          ; unsigned __int64
call    __Z16OSUnserializeXMLPKcmPP8OSString ; OSUnserializeXML(char const*,ulong,OSString **)
mov     r14, rax
mov     r15, r14
```

° ° °

```
mov     rdi, r15
call    qword ptr [rax+28h] ; OSObject::release
```

However…

- **If the allocated object is referenced by another component, it will not be released even if call *object::release* to it.**
- ***IORegistry* is a good choice for Heap Fengshui**
- **So we find *OSUnserializeXML* invoking nearby *IORegistry* method calling …**

Always an exception☺

- **In IOKIT service *IOMProotDomain* , slector 7 (*kPMSleepSystemOptions*)**

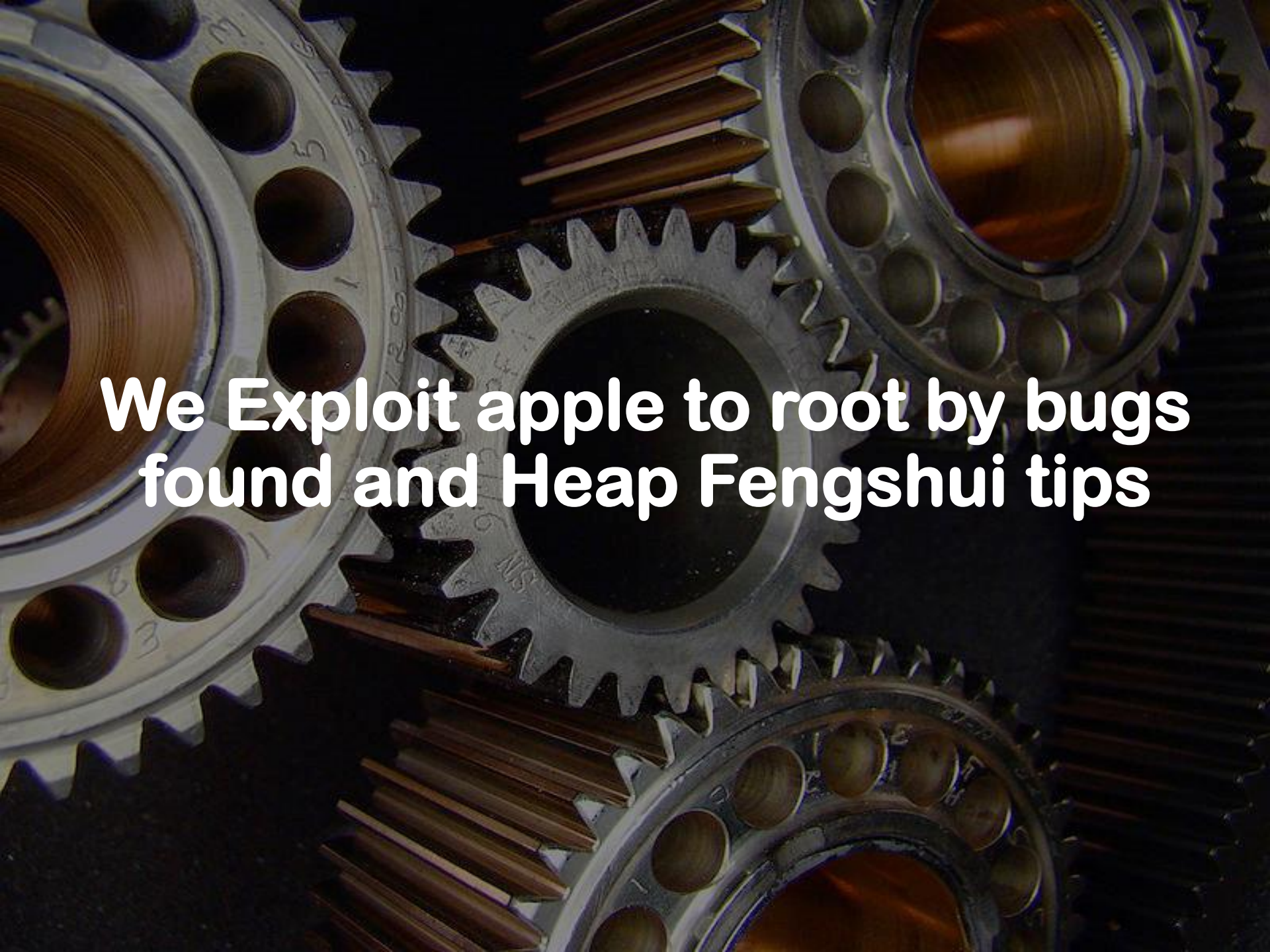
RootDomainUserClient::secureSleepSystemOptions

```
unserializedOptions = OSDynamicCast( OSDictionary,
                                     OSUnserializeXML((const char *)inOptions, inOptionsSize,
```

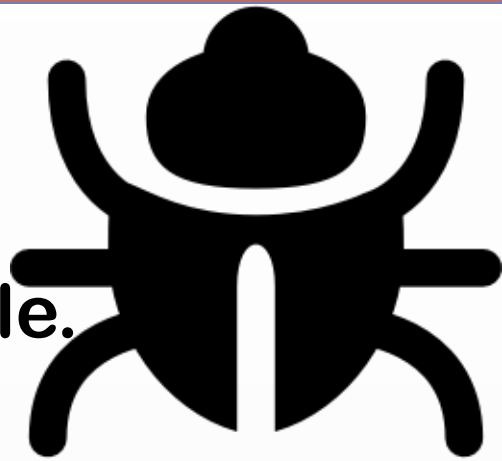
。。。

```
// Publish Sleep Options in registry under root_domain
fOwner->setProperty( kRootDomainSleepOptionsKey, unserializedOptions);
```

```
*returnCode = fOwner->sleepSystemOptions( unserializedOptions );

unserializedOptions->release();
```


We Exploit apple to root by bugs found and Heap Fengshui tips

Bugs



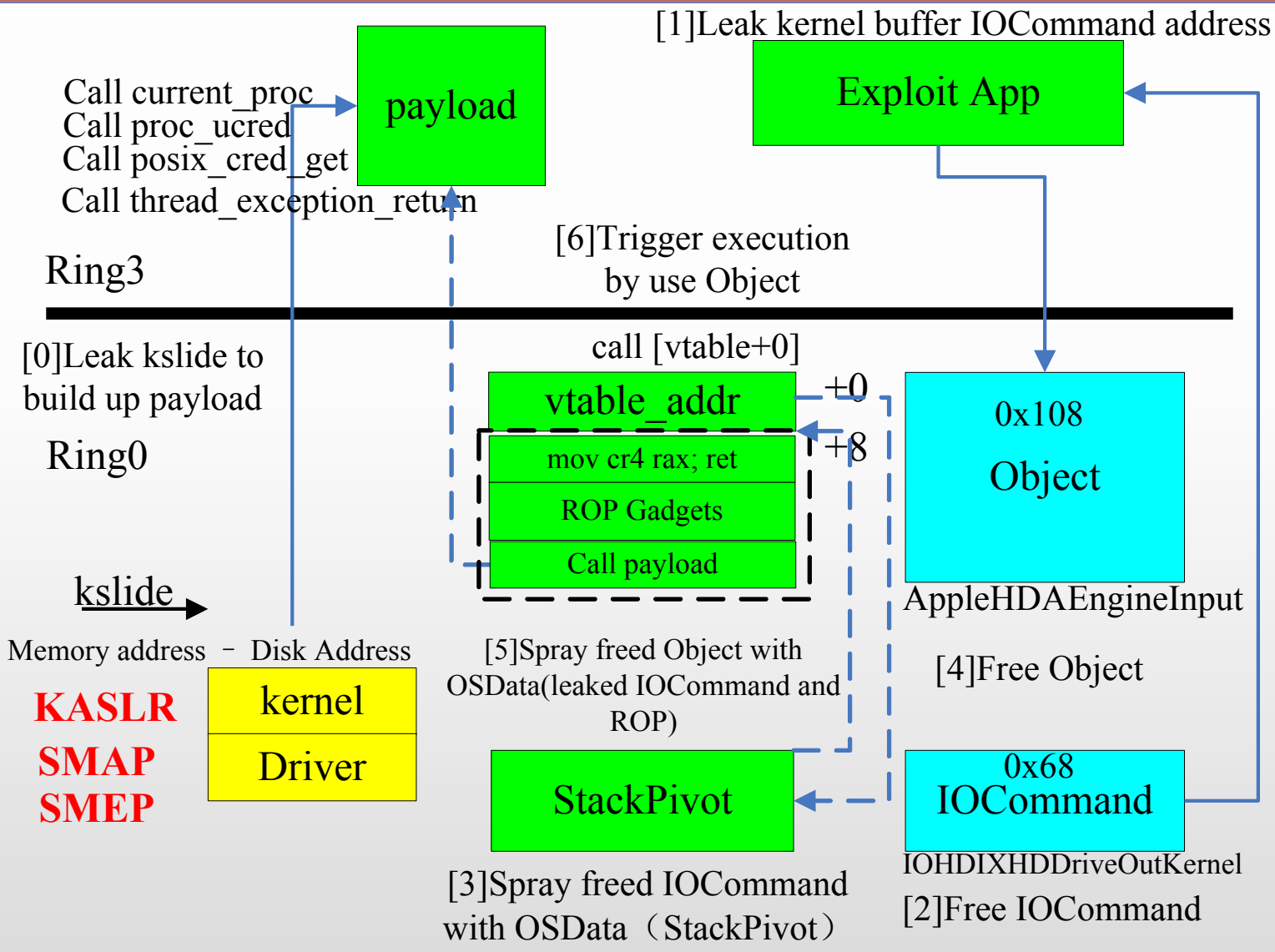
- **CVE-2016-xxx :**

  This is typical UAF vulnerability in AppleHDAEngineInput kernel module.

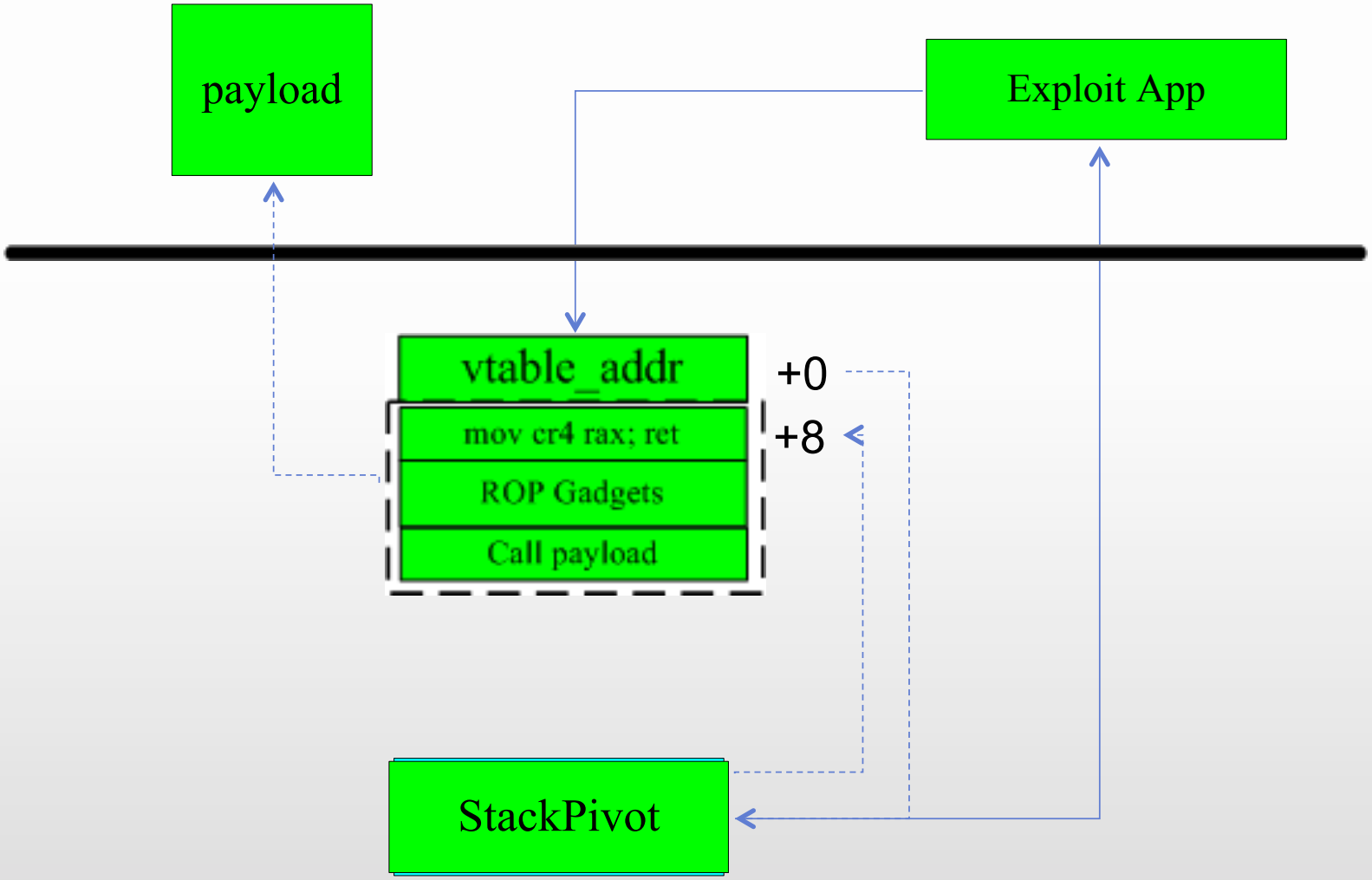
- **CVE-2016-xxxx:**

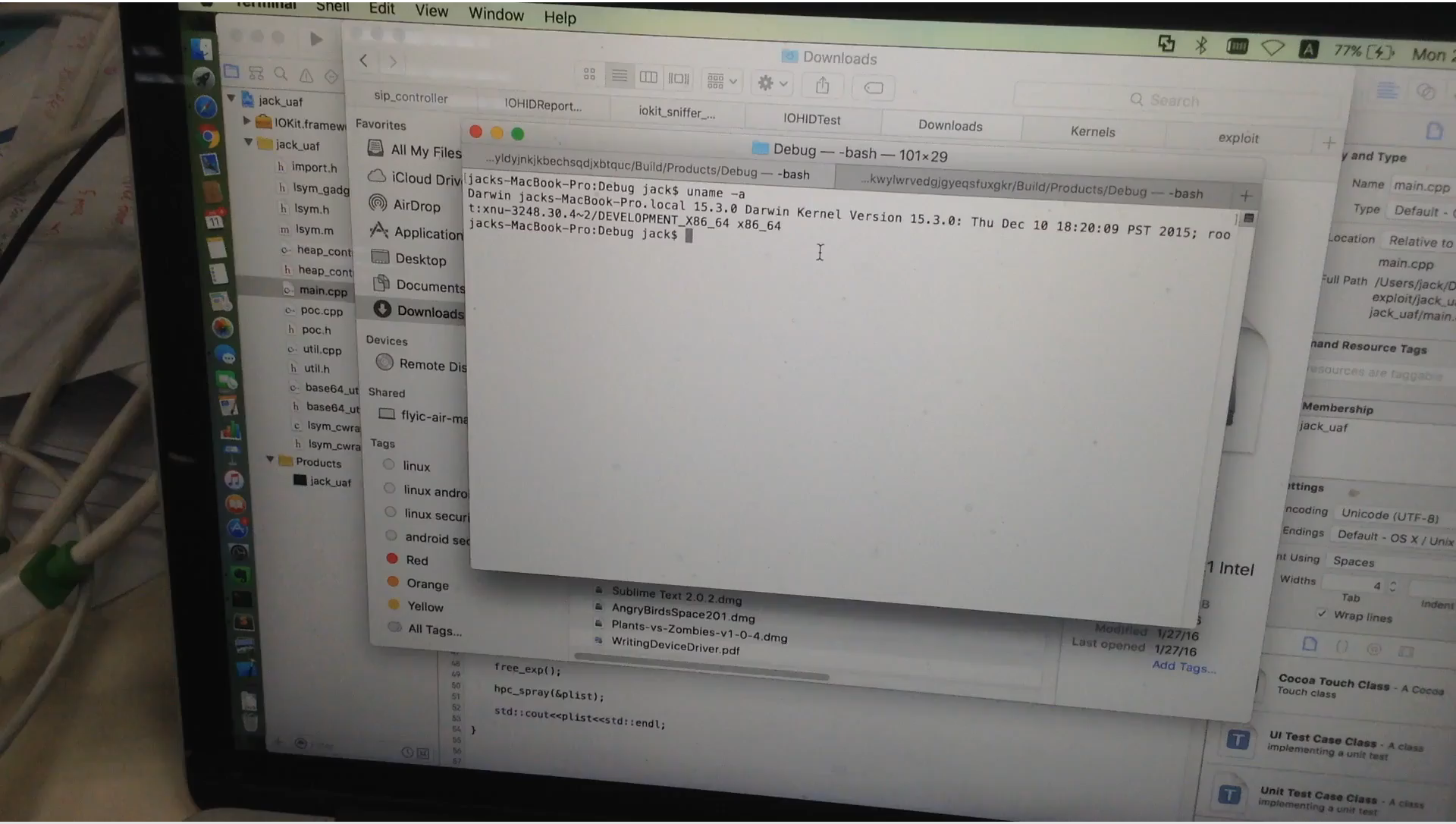
  Another bug in the Disk image module can leak an object address. The address exists in kernel heap.

Exploit Process 1/2



Exploit Process 2/2




Terminal Shell Edit View Window Help
Downloads
Debug — -bash — 101×29
jacks-MacBook-Pro:Debug jack$ uname -a
Darwin jacks-MacBook-Pro.local 15.3.0 Darwin Kernel Version 15.3.0: Thu Dec 10 18:20:09 PST 2015; roo
t:xnu-3248.30.4~2/DEVELOPMENT_X86_64 x86_64
jacks-MacBook-Pro:Debug jack$
Favorites
All My Files
iCloud Drive
AirDrop
Applications
Desktop
Documents
Downloads
Devices
Remote Disc
Shared
flyic-air-ma
Tags
linux
Red
Orange
Yellow
All Tags...
Sublime Text 2.0.2.dmg
AngryBirdsSpace201.dmg
Plants-vs-Zombies-v1-0-4.dmg
WritingDeviceDriver.pdf
free_exp();
hpc_spray(&plist);
std::cout<<plist<<std::endl;
}
Modified 1/27/16
Last opened 1/27/16
Add Tags...
Cocoa Touch Class - A Cocoa Touch class
UI Test Case Class - A class implementing a unit test
Unit Test Case Class - A class implementing a unit test

Thanks very much

Special thanks :

Herry Li, @zenhumany

Juwei Lin, @fuzzerDOTcn