black hat EUROPE 2018

DECEMBER 3-6, 2018

EXCEL LONDON / UNITED KINGDOM

Drill the Apple Core: Up & Down

Fuzz Apple Core Component in Kernel and User Mode for Fun and Profit

Juwei Lin

- @panicall
- Joined TrendMicro Since 2013
- Windows Kernel/Rootkit/Bootkit
- Ransomware Decryption
- iOS/Android/Mac Vulnerability Hunting

TREND MICRO research

Lilang Wu

- @Lilang_Wu
- Joined Trend Micro Since 2016
- Mac/iOS Vulnerability/Malware
- iOS/Android Exploit Detection

Moony Li

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

Agenda

- Smart Fuzz XPC
  - XPC Internals
  - Fuzz Strategy
  - Reproduce Strategy
  - Output

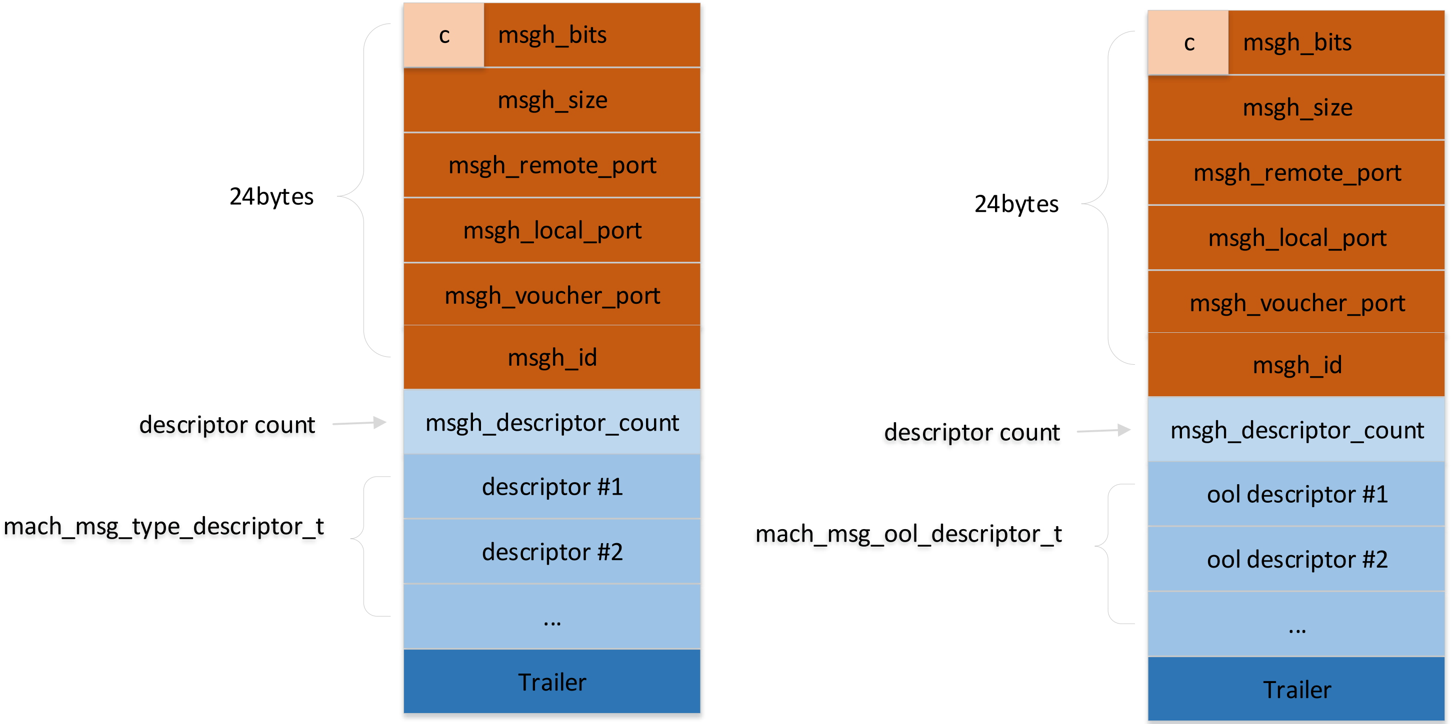
Agenda

Agenda

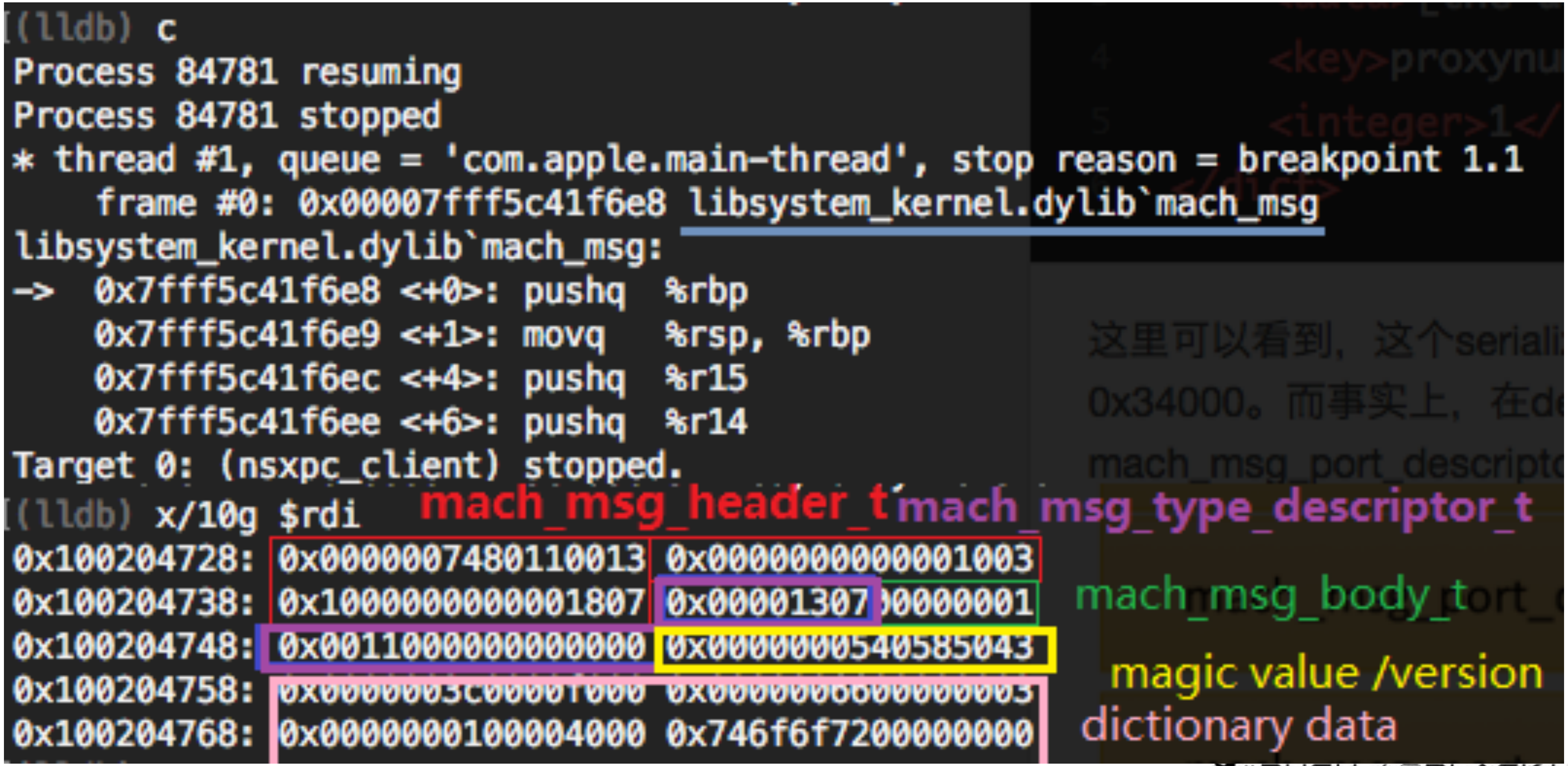
- Smart Fuzz XNU
  - Introduction
  - Architecture and Sanitizer Support
    - Syntax Engine and Corpus
    - Sanitizers
  - Root Case Study

Smart Fuzzing XPC

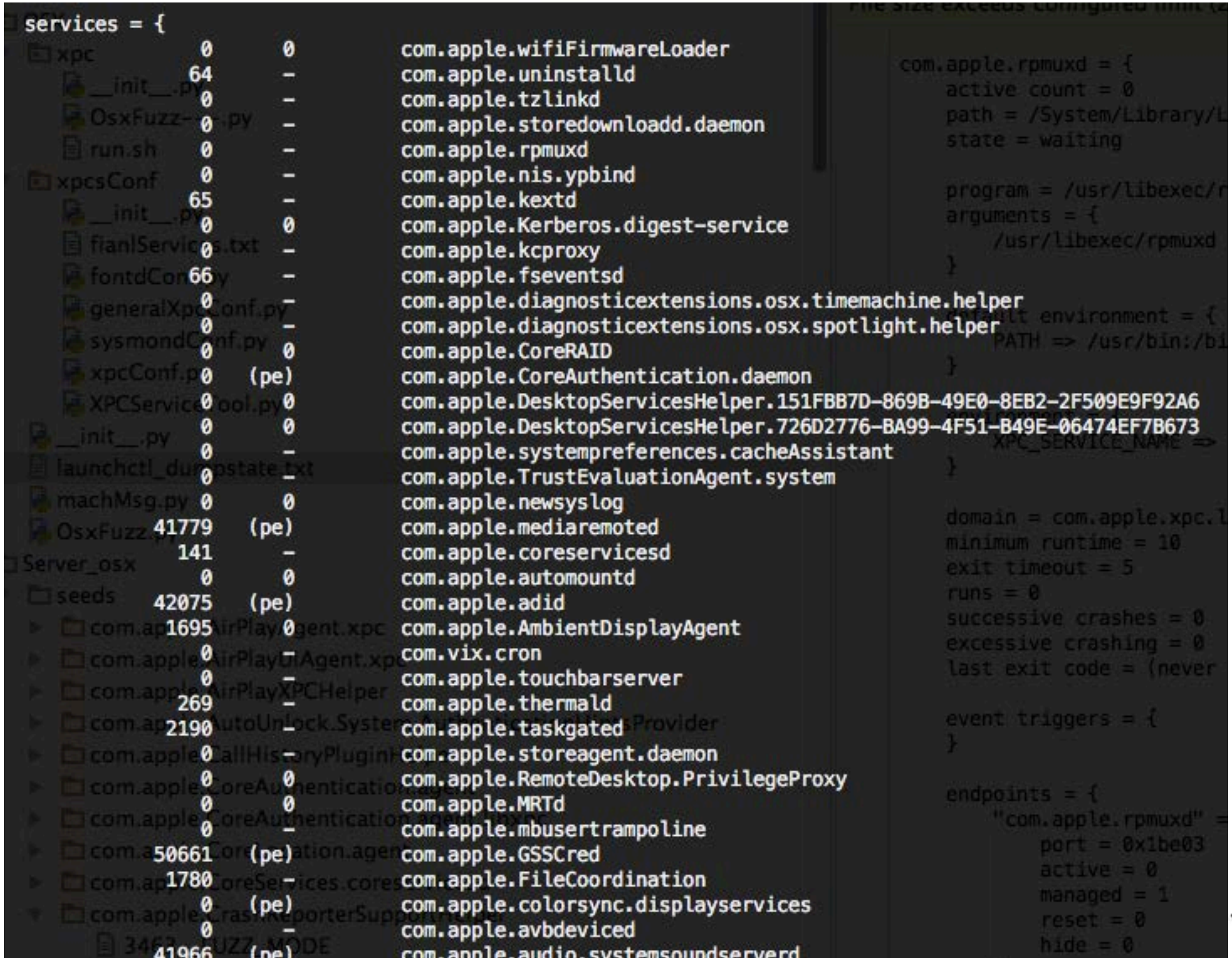
- What is XPC?
  - low-level (libSystem) interprocess communication mechanism
  - simple messages and complex messages



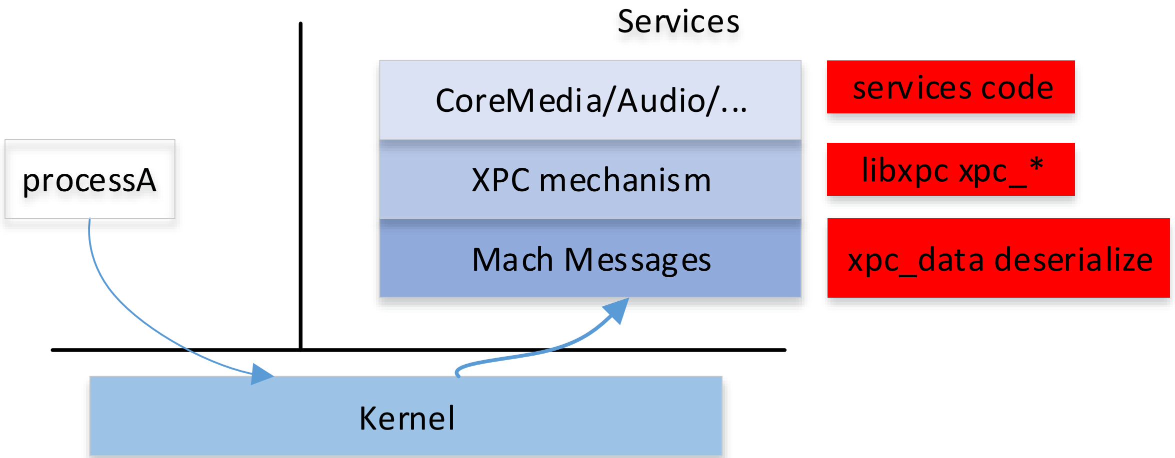
- Message Binary Format



- XPC Services
  - `launchctl dumpstate`



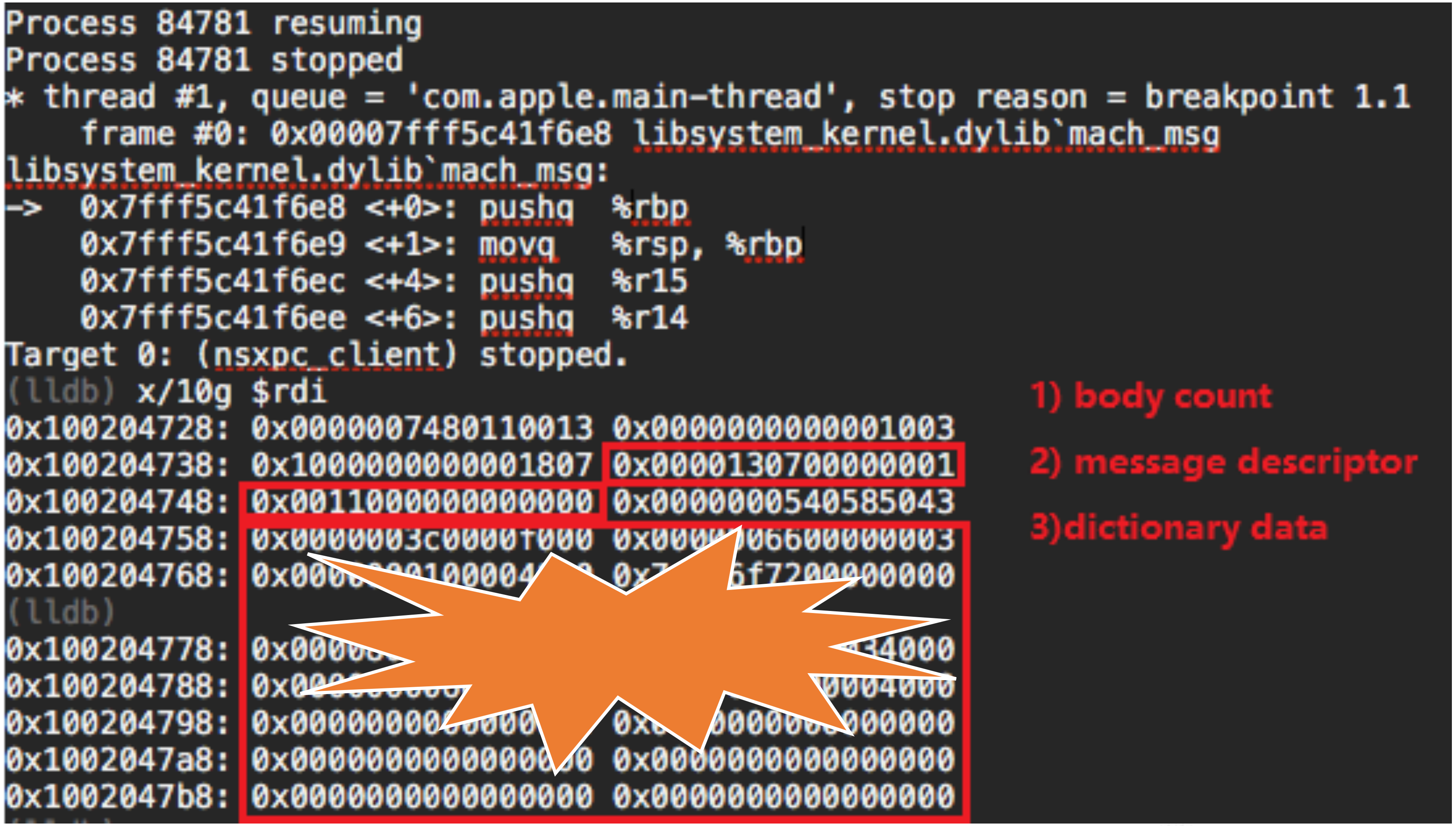
- Attack Surface
  - serialize/deserialize
  - libxpc
  - services code



- How to trigger these bugs?

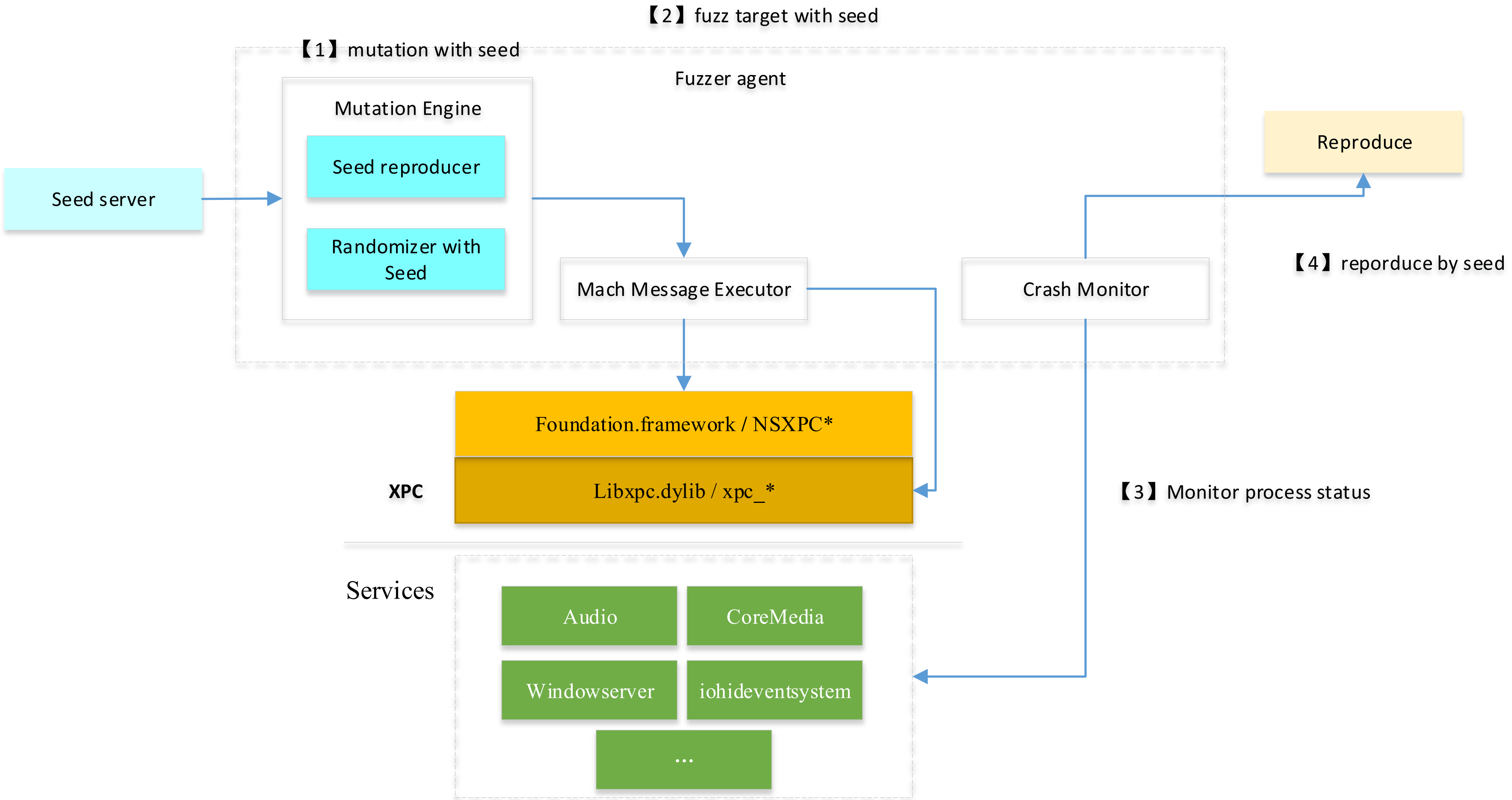
Crafted Mach Message

- Proactive fuzz



- Fuzz Strategy
  - Easy to control
  - Easy to mutate
  - Easy to monitor
  - Easy to reproduce

XPC Fuzz Architecture



- Fuzz Controller
  - ✓ Wrap the xpc interfaces by python

```
BOOST_PYTHON_MODULE(xpcconnection) {
    PyEval_InitThreads();

    class_<XpcConnection, boost::noncopyable>("XpcConnection", init<std::string>())
        .def("XpcCreateConnection", &XpcConnection::XpcCreateConnection)
        .def("mach_connect", &XpcConnection::mach_connect_)
        .def("XpcHandler", pure_virtual(&XpcConnection::handler))

        .def("mach_msg", &XpcConnection::mach_msg_)
        .def("XpcSendMessage", &XpcConnection::XPCSendMessage)
    ;
}
```

  - ✓ Python fuzz Engine

• Mutation

- Pseudo-Random Number Generator with Mersenne Twister Algorithm


```
{u'body': [0, 0, -1685819353, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -703626210, 0, 0, 0, 0, 0, 0,
{u'body': [0, 0, -192701416, 0, 0, 0, 0, 0, 443081805, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
{u'body': [2077175787, 0, 0, 0, 0, 0, 0, 0, 0, -1161004373, 0, 0, 0, -1551233303, -1459764896,
{u'body': [0, 1159872152, -23434069, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1946001112, 0, 0, 0, -102359
{u'body': [0, -1959869915, 0, -130743320, 0, 0, 0, 0, 0, 0, 0, 0, 0, -2146837368, 0, 0, 0, 0,
{u'body': [0, -940122787, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1747064665,
{u'body': [0, 1403206489, 0, 882413436, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
{u'body': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
{u'body': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 896603545, 0, 0, 0, 0, 0, 0, 0,
{u'body': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1004022596, 0, 0, -79935722, 0, 0, 0, 0,
{u'body': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1104
{u'body': [1209349933, 0, 0, -1746059595, 0, 0, 0, 0, 0, 0, -693017329, 0, 0, 0, 0, 0, 0, 0,
{u'body': [1013802756, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -80985600, 0, 0, 0, 0, 0, 0, 0, 0,
{u'body': [0, -151049328, 0, -1586927264, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
{u'body': [0, 0, 0, -1070465099, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
{u'body': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1490872793, 0, 0, 0, 0,
{u'body': [0, 0, 0, 0, 0, 0, 0, 0, 1240273877, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 4307926
{u'body': [970815016, 0, 0, -521559532, 0, 1091091141, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -2
{u'body': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1974076433, 0, 0, 0, -1612598270, 0, 0, 0, 0, 0,
{u'body': [0, 0, 0, 1750203268, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
{u'body': [0, 0, 0, 0, 0, 0, 0, 688048238, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1934665107, 0, 0, -2603
{u'body': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -340710490, 0, 0,
{u'body': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 417424383, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
{u'body': [-659606155, 0, 0, -882768697, 0, 0, 0, 0, 0, 0, 0, 2139409034, 0, 0, 0, 0, 0, 0, 0,
{u'body': [0, -1709845420, 0, 0, 1932744245, 242408057, 0, 0, 1719687880, 0, 0, 0, 0, 0, 0, 0,
{u'body': [635838260, 0, 0, 0, 0, 0, 1780214846, 0, 0, 0, 0, 0, 0, 551006384, 0, 0, 0, 0, 0,
{u'body': [0, 1195509111, 0, 0, 0, 167570343, 0, 0, 0, 0, 0, 0, 0, 588668277, 0, 0, 0, 0, 0,
{u'body': [0, 0, 0, 0, 0, -1965192056, 0, 0, -734341013, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
{u'body': [0, 0, 0, 0, 0, 0, 0, -1089037147, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
{u'body': [0, 619451316, 0, 0, 0, 0, 0, -552825349, 0, 0, -1934681113, 0, 0, 0, 0, 869940682,
{u'body': [0, 0, -1533203033, 0, 0, 0, 0, 0, 0, 0, 1130896154, -321631898, 0, 0, 0, 0, 0, 0,
```

- Crash Monitor
  - Monitor the processes IDs cluster status
  - Monitor exits signal value

```
zuffdemac-pro:~ zuff$ launchctl list
PID     Status  Label
-       0       com.apple.SafariHistoryServiceAgent
307     0       com.apple.Finder
336     0       com.apple.homed
578     0       com.apple.SafeEjectGPUAgent
-       0       com.apple.quicklook
-       0       com.apple.parentalcontrols.check
-       0       com.apple.PackageKit.InstallStatus
345     0       com.apple.mediaremoteagent
-       0       com.apple.FontWorker
321     0       com.apple.bird
-       0       com.apple.familycontrols.useragent
-       0       com.apple.AssetCache.agent
666     0       com.apple.universalaccessAuthWarn
312     0       com.apple.nsurlsessiond
-       0       com.apple.mobileactivationd
-       0       com.apple.syncservices.uihandler
352     0       com.apple.iconservices.iconservicesagent
```

- Comparison between different Reproduce Methods

	Typical Example	Storage Cost	Speed Cost	Support Complex Scenario	Reproduce Rate	Dev Effort
Log	Trinity	High (Execution Log)	High	Low	Low	Low
Case(File)	AFL	Middle (Files Causing Crash)	Low	Middle	Middle	High
Crash Dump	-	High (Every Crash Context)	High	-	Very Low	No
Seed	**JS Fun Fuzz**	**Low (Integer)**	**Low**	**High**	**High**	**Low**

- Case Study - CVE-2018-4411

```
Target 0: (fontd) stopped.
    (lldb) bt
    * thread #1, queue = 'com.apple.main-thread', stop reason = EXC_BAD_ACCESS (code=1, address=0x7ffee1934000)
    * frame #0: 0x00007fff55a06f49 libsystem_platform.dylib`_platform_memmove$VARIANT$Haswell + 41
    frame #1: 0x00007fff2b8b597a libATSServer.dylib`FODBWriteToAnnex + 246
    frame #2: 0x00007fff2b8d0157 libATSServer.dylib`HandleFontManagementMessage + 5403
    frame #3: 0x00007fff2b8cd2d1 libATSServer.dylib`serverMainHandler(__CFMachPort*, FontMgrMessage*, long, void*) + 263
    frame #4: 0x00007fff2d3e4596 CoreFoundation`__CFMachPortPerform + 310
    frame #5: 0x00007fff2d3e4449 CoreFoundation`__CFRUNLOOP_IS_CALLING_OUT_TO_A_SOURCE1_PERFORM_FUNCTION__ + 41
    frame #6: 0x00007fff2d3e4395 CoreFoundation`__CFRunLoopDoSource1 + 533
    frame #7: 0x00007fff2d3dbf50 CoreFoundation`__CFRunLoopRun + 2848
    frame #8: 0x00007fff2d3db1a3 CoreFoundation`CFRunLoopRunSpecific + 483
    frame #9: 0x00007fff2d419c33 CoreFoundation`CFRunLoopRun + 99
    frame #10: 0x00007fff2b8cc91c libATSServer.dylib`main_handler + 4510
    frame #11: 0x00007fff556f5015 libdyld.dylib`start + 1
    frame #12: 0x00007fff556f5015 libdyld.dylib`start + 1
```

```
// (__CFMachPort*, FontMgrMessage*, long, void*)
void __fastcall serverMainHandler(double a1, __int64 a2, __int64 a3)
{
  ...
 }
 else
 {
   v4 = HandleFontManagementMessage((FILE *)a3, &v10, a1);// a3=msg
   FORemoveExceptionFrame(&v8, &v10);
   v5 = 1;
 }
```

```
    || (v12 = __ROL2__(*(_WORD *)(v11 + 22), 8), *(_WORD *)(gFontContainerLis
   || !gAnnexDB && (v8 = FODBOpenAnnexFile(v10)) != 0 )
 {
   result = (unsigned int)v8;
 }
 else
 {
   result = FODBWriteToAnnex(v7, a2, v6, v5, a5);// a2=buffer, a3=v6=size
 }
 return result;
}
```

```
    goto LABEL_344;
  case 0x28:
    v82 = &v238->_bf;
    if ( gUseNewFODB == 2 )
    {
      FODBBeginTransactions(9);
      if ( LODWORD(v82->_base) )
      {
        v83 = *(&v238->_lbfsize + 1);
        a2 = *(const char **)((char *)&v238->_bf._base + 4)
        v84 = v238->_lbfsize;
      }
      else
      {
        a2 = (const char *)(&v238->_bf._size + 1);
        v83 = HIDWORD(v238->_bf._base);
        v84 = v238->_bf._size;
      }
      FODBAddAnnex(v83, a2, v84, 0, a3);      // a2=buffer, v84=size
      FODBEndTransactions(9LL);
    }
```

```
Microseconds((__int64)v54);
*(_QWORD *)(v12 + 3) = (v16 << 32) | v54[0];
v17 = __ROL2__(v13, 8);
LOWORD(v58) = v17;
v18 = __ROL2__(v14, 8);
HIWORD(v58) = v18;
v12[5] = v58;
v19 = v12;
memcpy(v12 + 6, a2, v53);                    // v53=a3=size
if ( *(_BYTE *)(gAnnexAuxFile + 12LL) )
{
```

Out of boundary

Smart Fuzzing XNU

Smart Fuzzing XNU

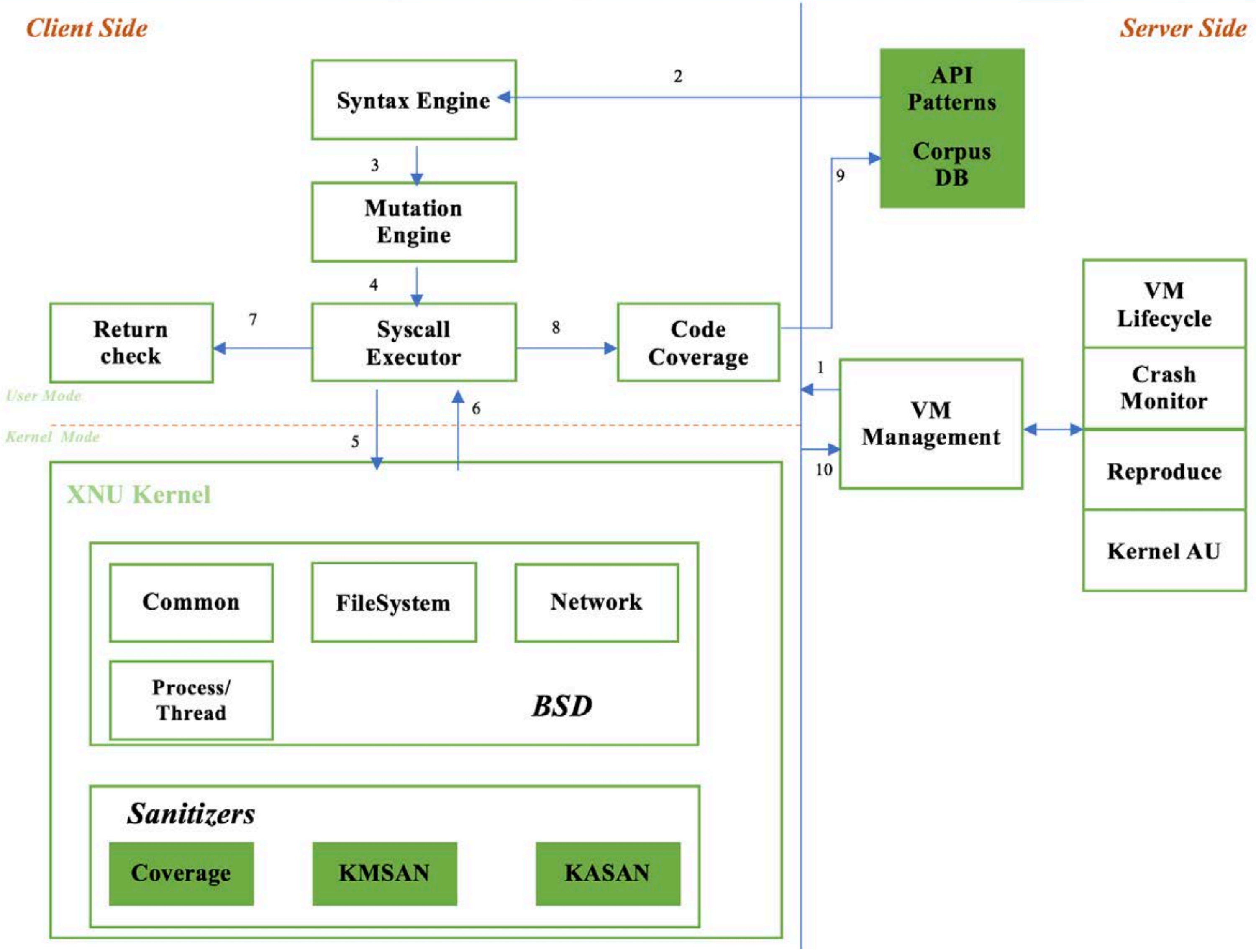
- Introduction of Smart Fuzzing XNU
- Architecture and Sanitizer Support
- Syntax Engine and Corpus
- Sanitizers
- Root Case Study

What I will introduce today

1. Port Syzkaller to Support macOS XNU Fuzzing.

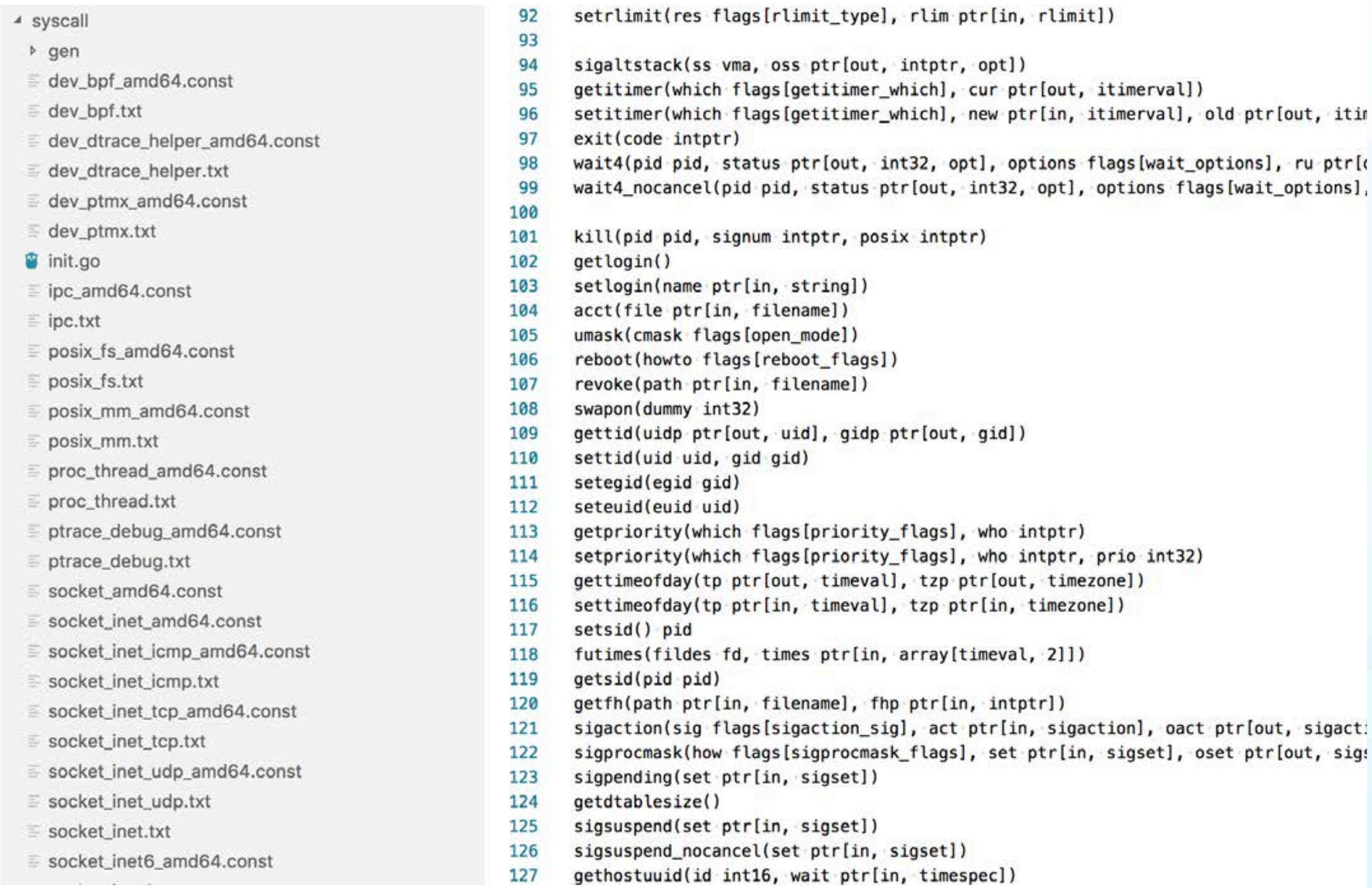
2. Modify XNU to add support some features.

Fuzzer	XNU
530 BSD API Patterns	Add Code Coverage
VM Fusion Support	Add Kernel Memory Sanitizer
macOS Executor	Enable Kernel Address Sanitizer

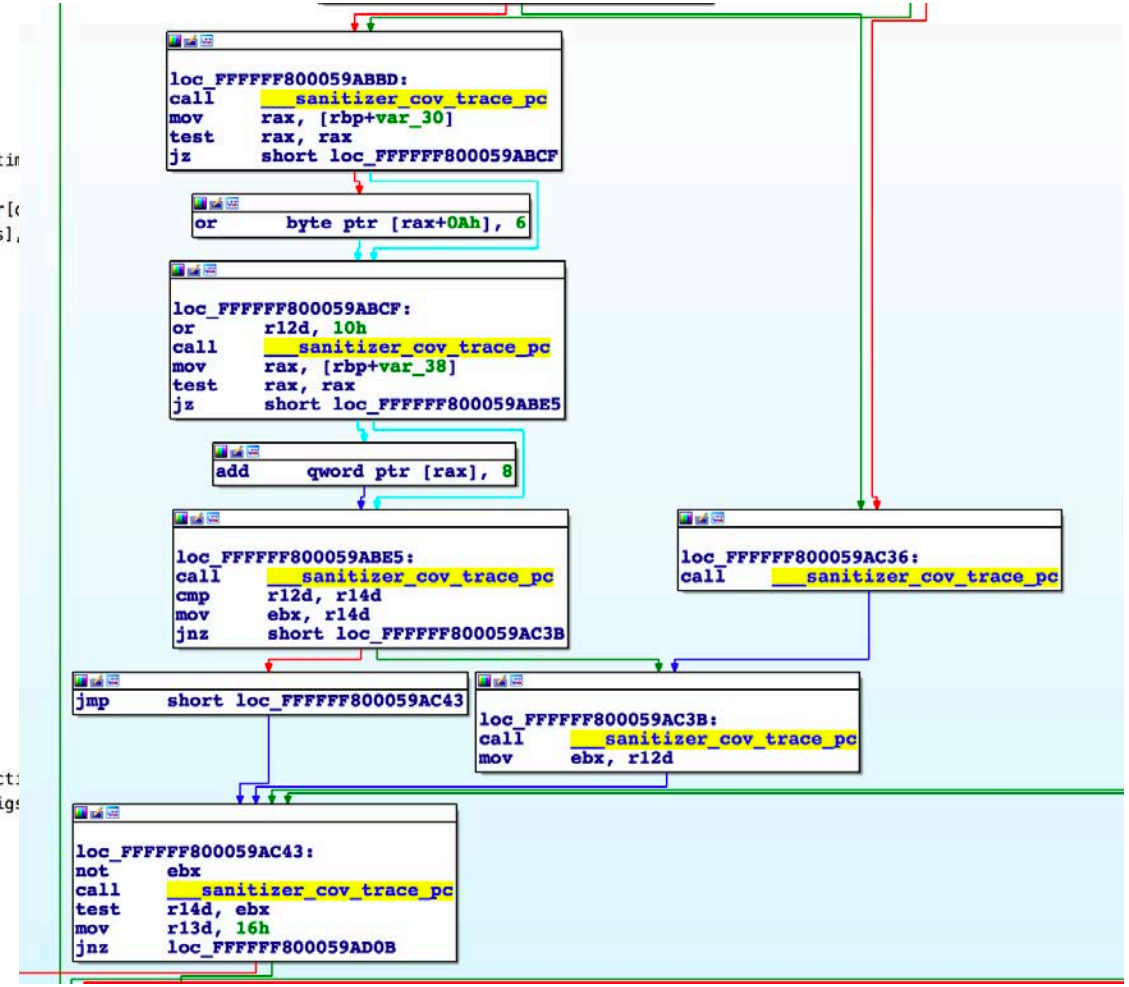


Architecture

1. Key modules are in GREEN
2. Also add some other modules, e.g. vmfusion



API Pattern



Code Coverage

My Efforts

- Syntax Engine is directly from Syzkaller; But I developed the XNU BSD API patterns.
- Kasan is from XNU, but it does not work well after compilation.
- I developed coverage sanitizer.
- I developed kmsan.

Syntax Engine & Corpus

Quick glance at syzkaller’s syntax engine

syz-extract & syz-sysgen

- description in txt written by human

generation engine

- description in go made by previous 2 tools

executor

- program made by previous generation engine

Corpus

- More than 500 syscalls in XNU kernel
- Refer to syzkaller's syscall descriptions syntax: https://github.com/google/syzkaller/blob/master/docs/syscall_descriptions_syntax.md
- Refer to sample txt files in syzkaller project

Sanitizers

Basic Concepts 1: User Mode Sanitizers[1]

Name	Features	Comments
AddressSanitizer[2]	• Out-of-bounds accesses to heap, stack and globals • Use-after-free • Use-after-return • Use-after-scope • Double-free, invalid free	• compiler instrumentation module • run-time library
MemorySanitizer[3]	• uninitialized reads	
SanitizerCoverage[4]	• get function/block/edge coverage	• Instrumentations • Default callbacks provided
...		• ThreadSanitizer[5] • UndefinedBehaviorSanitizer[6] • DataflowSanitizer[7] • LeakSanitizer[8]

Basic Concepts 2: Kernel Mode Sanitizers

Name	Features	Comments
Kernel Sanitizer Coverage	• get function/block/edge coverage	• Has instrumentations support • NO existing callbacks implementation
KASAN (kernel address sanitizer)	• Out-of-bounds accesses Use-after-free • Use-after-return • Use-after-scope • Double-free, invalid free	• Has instrumentations support • Has callbacks/module support
KMSAN (kernel memory sanitizer)	• uninitialized reads	• Not implemented

Sanitizer Coverage

- We need to develop a new module in XNU to:
  - Support sanitizer callback function
  - Read the coverage data back to user fuzzing program

Callback Implementation

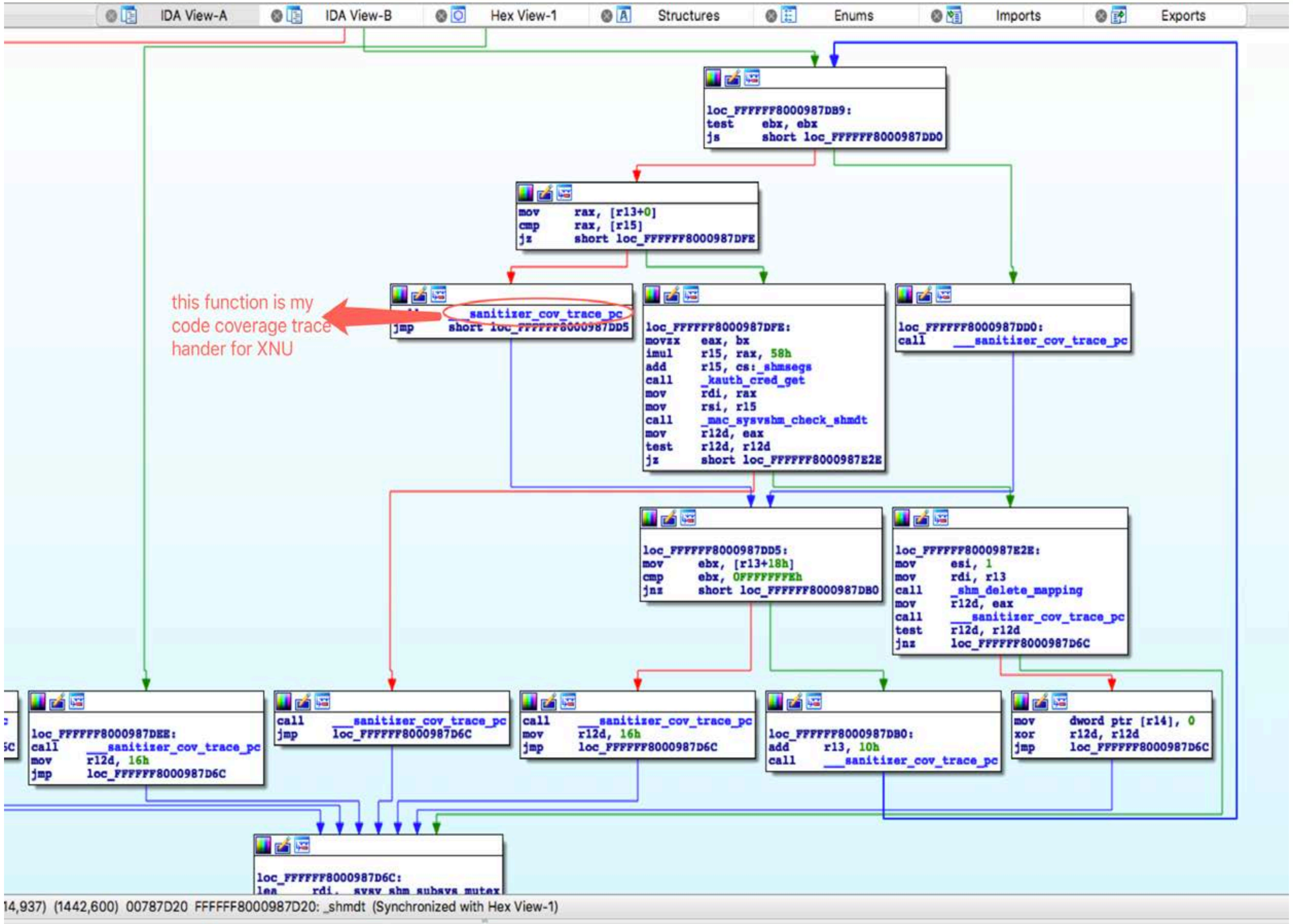
```
struct task {
    ...

    enum kcov_mode kcov_mode;
    unsigned     kcov_size;
    void         *kcov_area;
    struct kcov *kcov;
    uint32_t     refcount;
}

void __attribute__ ((noinline)) __sanitizer_cov_trace_pc()
{
    ...
}
```

1. callback name: __sanitizer_cov_trace_pc
2. just support single-thread mode
3. store coverage structure into task_t

After Compilation



KASAN

- latest XNU has KASAN support
  - KDK now provides kernel.kasan which works well.
  - It does not work if you compile it, VM cannot boot.

- It consists of *guard pages*, *shadow memory* and *operations*.

- It can protect Globals, Stack and Heap memory.

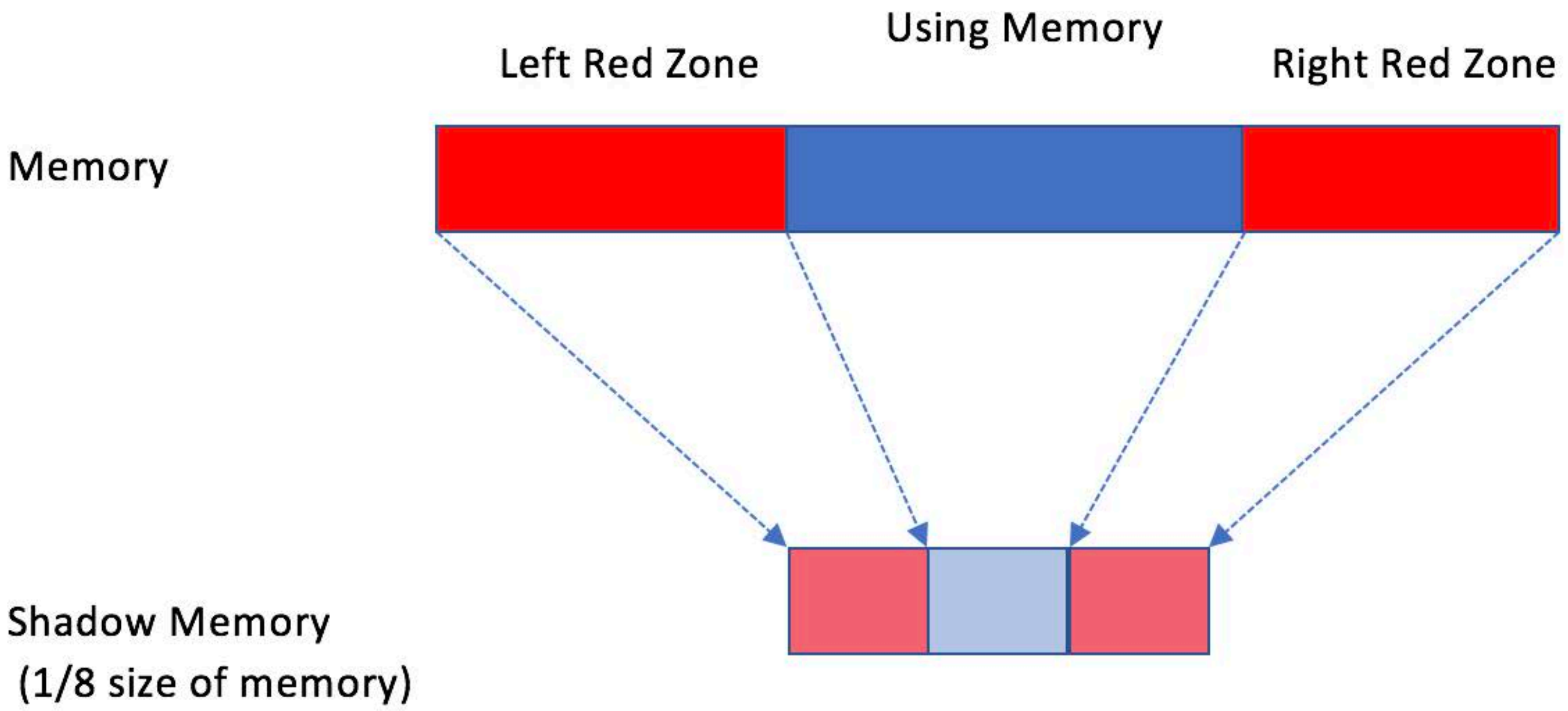
How KASAN protects memory

1) memory operations are called, e.g. __asan_strlcpy

2) __asan_strlcpy checks shadow memory

3) KASAN panics the kernel if shadow memory is illegal(shadow value < 0)

Guard Pages & Shadow Memory



Operations

Heap Memory Operations	Stack Memory Operations	Other Memory Operations
__asan_bcopy	__asan_stack_malloc_0	__asan_load1
__asan_memmove	__asan_stack_malloc_1	__asan_load2
__asan_memcpy	__asan_stack_malloc_2	__asan_load4
__asan_memset	__asan_stack_malloc_3	__asan_load8
__asan_bzero	__asan_stack_malloc_4	__asan_load16
__asan_bcmp	__asan_stack_malloc_5	__asan_loadN
__asan_memcmp	__asan_stack_malloc_6	
__asan_strlcpy	__asan_stack_malloc_7	
__asan_strlcat	__asan_stack_malloc_8	
__asan_strncpy	__asan_stack_malloc_9	
__asan_strncat	__asan_stack_malloc_10	
__asan_strnlen		
__asan_strlen		

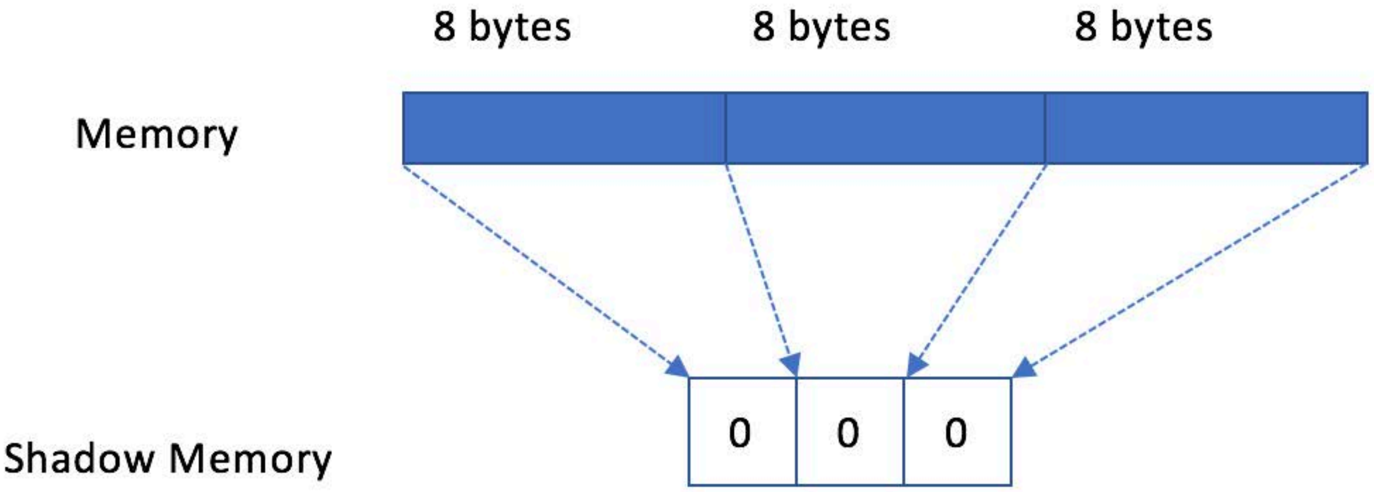
#define strlcpy __asan_strlcpy

-fsanitizer=address

buildin calls in xnu source code

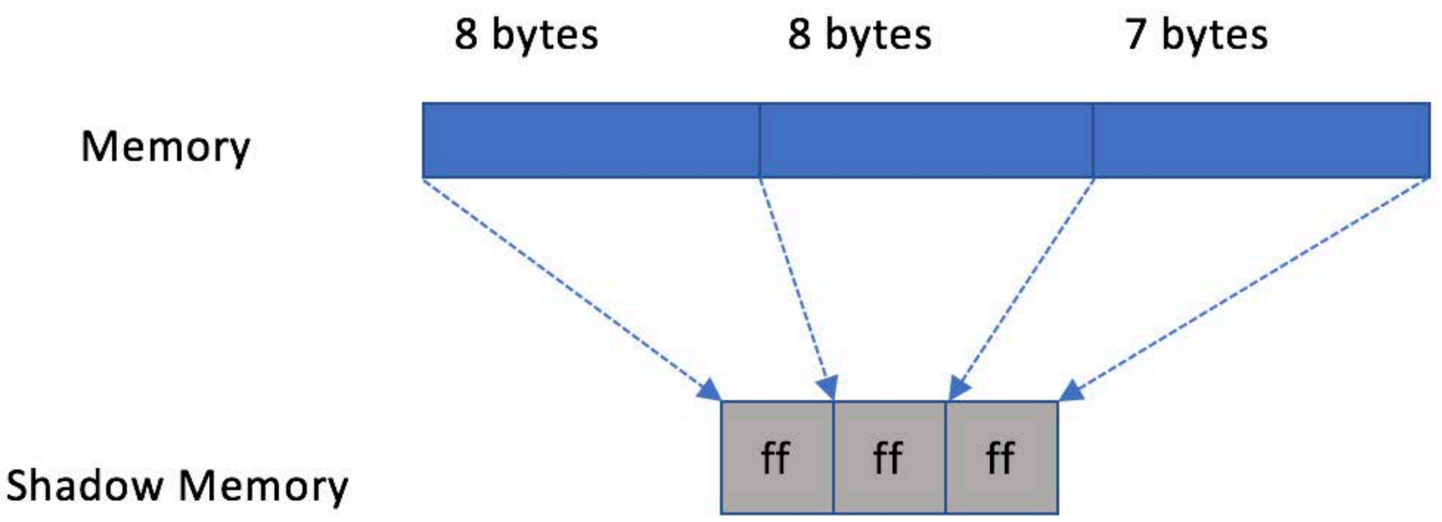
Example: Detect UAF

- When new memory is allocated and aligned with 8



Example cont. 1

- When the memory is freed



Example cont. 2

- When the memory is used after free, any related operation will check its shadow memory and then panic the system.
  - 0xff is illegal

KMSAN

- Kernel memory sanitizer is used to detect uninitialized memory.

- We worked on how to initialize all uninitialized memory allocated in kernel, e.g. kalloc_canblock

kalloc_canblock

```
    assert(size <= z->elem_size);

#if VM_MAX_TAG_ZONES
    if (z->tags && site)
    {
        tag = vm_tag_alloc(site);
        if (!canblock && !vm_allocation_zone_totals[tag]) tag = VM_KERN_MEMORY_KALLOC;
    }
#endif

    addr =  zalloc_canblock_tag(z, canblock, size, tag);

#if KASAN_KALLOC
    /* fixup the return address to skip the redzone */
    addr = (void *)kasan_alloc((vm_offset_t)addr, z->elem_size, req_size, KASAN_GUARD_SIZE);

    /* For KASan, the redzone lives in any additional space, so don't
     * expand the allocation. */
#else
    *psize = z->elem_size;
#endif

    // add by @panicall
    if (addr)
        memset(addr, 0xde, *psize);
    return addr;
}
```

Check in fuzzer

Kernel_2018-06-11-041833_panicall1s-Mac-6.panic

Reveal Now Clear Reload Share Search

```
Anonymous UUID:       039B94D9-F271-8A3F-BAE4-C8BF9D6B5BEE

Mon Jun 11 04:18:33 2018

*** Panic Report ***
panic(cpu 0 caller 0xffffff8025530e77): "[panicall] report leak=========\n
          syscall: [redacted]\n       a0: 0x200000004 a1: 0x10001 a2: 0x2\n
          a3: 0x200000002 a4: 0x0 a5: 0x0\n        a6: 0x0 a7: 0x0 a8: 0x0\n
          leak arg index: 0\n          leak arg len: 256\n
          \n      leak buffer: \n            0  0 14  0  0  0 F5  1  0  0 14  0  0  0 FE  9
 1  0  1  0  0  0  3  0  0  0  0  0  0  0 84
3E  1  0 15 13  0  0  0  0 DE DE DE 86 1D 5B
 0  0  0  0 DE 86 1D 5B  0  0  0  0 E9 84 1D
5B  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
\n"@/Users/juwei_lin/Vuln/myxnu/build-xnu-4570.41.2/xnu-4570.41.2/bsd/kern/sys_generic.c:3373
Backtrace (CPU 0), Frame : Return Address
0xffffff8887aa3760 : 0xffffff8024f0abc6 mach_kernel : _handle_debugger_trap + 0x426
0xffffff8887aa37b0 : 0xffffff80250939d4 mach_kernel : _kdp_i386_trap + 0x164
0xffffff8887aa37f0 : 0xffffff802508338a mach_kernel : _kernel_trap + 0x62a
0xffffff8887aa3870 : 0xffffff8024e9d110 mach_kernel : trap_from_kernel + 0x26
0xffffff8887aa3890 : 0xffffff8024f0a2e0 mach_kernel : _panic_trap_to_debugger + 0x1b0
0xffffff8887aa39c0 : 0xffffff8024f0a10c mach_kernel : _panic + 0x5c
0xffffff8887aa3a20 : 0xffffff8025530e77 mach_kernel : _panicall_report + 0x257
0xffffff8887aa3f20 : 0xffffff80255b6e60 mach_kernel : _unix_syscall64 + 0x3c0
0xffffff8887aa3fa0 : 0xffffff8024e9d8f6 mach_kernel : _hndl_unix_scall64 + 0x16

BSD process name corresponding to current thread: syz-executor1
Boot args: debug=0x146 kcsuffix=development keepsyms=1

Mac OS version:
17D47

Kernel version:
Darwin Kernel Version 17.4.0: Tue Jun  5 09:42:44 CST 2018; juwei_lin:xnu-4570.41.2/BUILD/obj/DEVELOPMENT_X86_64
Kernel UUID: E4E2BF51-CFE2-37C3-899A-88E4961DF567
Kernel slide:     0x0000000024c00000
Kernel text base: 0xffffff8024e00000
__HIB  text base: 0xffffff8024d00000
System model name: VMware7,1 (Mac-F60DEB81FF30ACF6)

System uptime in nanoseconds: 624205755866
last loaded kext at 295669359383: com.apple.filesystems.msdosfs
    1.10 (addr 0xffffff7fa609a000, size 69632)
last unloaded kext at 439573274479: com.apple.driver.AppleXsanScheme
    3 (addr 0xffffff7fa6f6c000, size 40960)
loaded kexts:
```

More than 3 bytes 0xDE means possbile memory leak

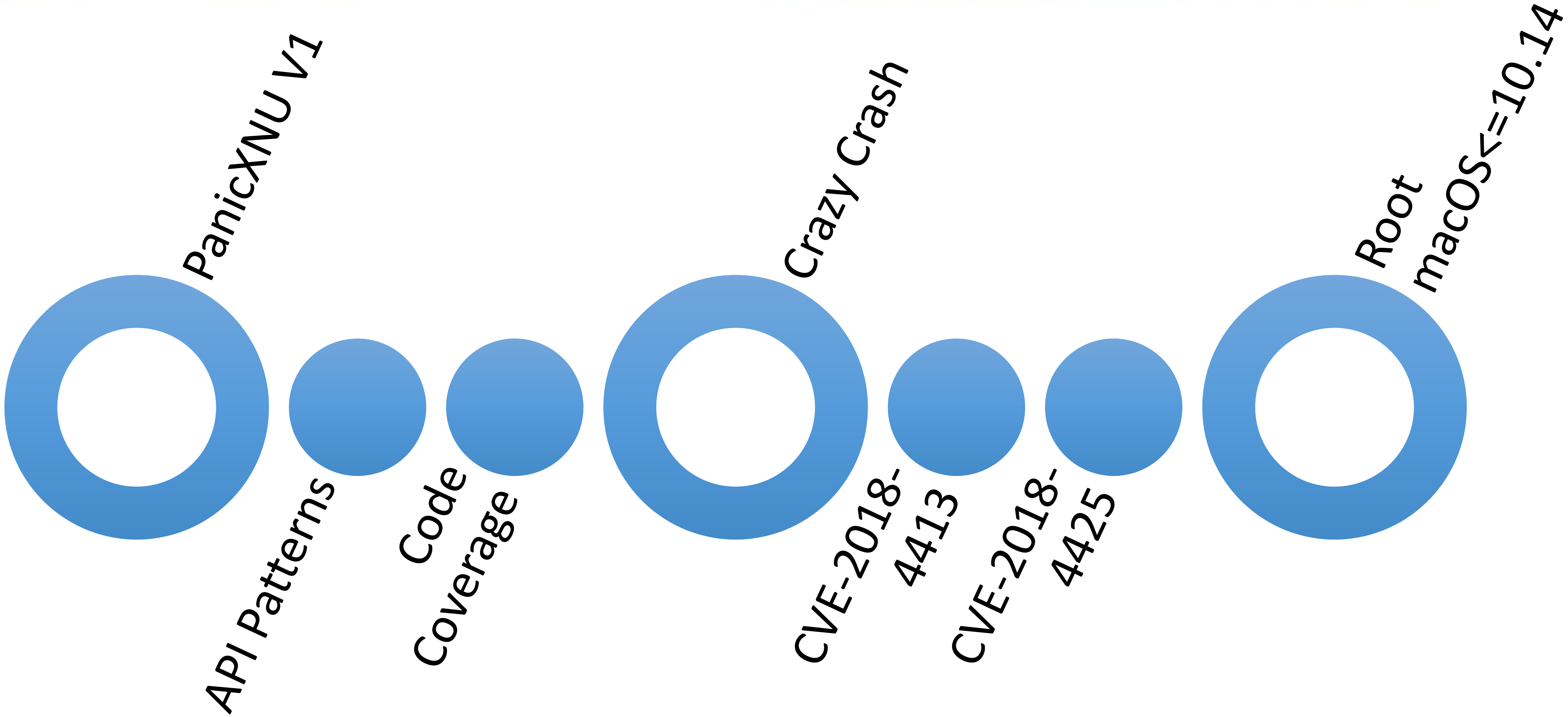
1. check continuous `0xDE`
2. add a syscall ‘panicall_report’

Conclusion

- About 530 API patterns
- Corpus

- Coverage Sanitizer
- KASAN
- KMSAN

macOS Root Case Study


PanicXNU V1
API Patterns
Code Coverage
Crazy Crash
CVE-2018-4413
CVE-2018-4425
Root macOS<=10.14

CVE-2018-4413

- Uninitialized heap memory leak
- Fixed in macOS 10.14.1 and iOS 12.1
- Can be used to leak ipc_port object address

CVE-2018-4425

- NECP type confusion
- Fixed in macOS 10.14
- Can be used to write arbitrary kernel address
- Can be used to free arbitrary kernel address

```
STATIC int
sysctl_procargsx(int *name, u_int namelen, user_addr_t where,
                 size_t *sizep, proc_t cur_proc, int argc_yes)
{
    ...


    if ((u_int)arg_size > p->p_argslen)
            arg_size = round_page(p->p_argslen);            --- (a)

        arg_addr = p->user_stack - arg_size;


    ...


    ret = kmem_alloc(kernel_map, &copy_start, round_page(arg_size), VM_KERN_MEMORY_BSD);
        if (ret != KERN_SUCCESS) {
                vm_map_deallocate(proc_map);
                return(ENOMEM);
        }

        copy_end = round_page(copy_start + arg_size);

        if( vm_map_copyin(proc_map, (vm_map_address_t)arg_addr,
                          (vm_map_size_t)arg_size, FALSE, &tmp) != KERN_SUCCESS) {
                        vm_map_deallocate(proc_map);
                        kmem_free(kernel_map, copy_start,
                                        round_page(arg_size));
                        return (EIO);
        }

        /*
         *      Now that we've done the copyin from the process'
         *      map, we can release the reference to it.
         */
        vm_map_deallocate(proc_map);
```

CVE-2018-4413

sysctl_procargsx is used to retrieve process args information by calling sysctl.

at location (a) :

- p->p_argslen is usually around 0x300;
- I set my arg_size to 0x200 so that arg_size will not be round_paged

```
        if( vm_map_copy_overwrite(kernel_map,               --- (b)
                                  (vm_map_address_t)copy_start,
                                  tmp, FALSE) != KERN_SUCCESS) {
                    kmem_free(kernel_map, copy_start,
                                    round_page(arg_size));
                    vm_map_copy_discard(tmp);
                    return (EIO);
        }

        if (arg_size > argslen) {
                data = (caddr_t) (copy_end - argslen);
                size = argslen;
        } else {
                data = (caddr_t) (copy_end - arg_size);         --- (c)
                size = arg_size;
        }


    ...


        if (argc_yes) {
                /* Put processes argc as the first word in the copyout buffer */
                suword(where, argc);
                error = copyout(data, (where + sizeof(int)), size);
                size += sizeof(int);
        } else {
                error = copyout(data, where, size);             --- (d)


}
```

CVE-2018-4413

At location (b):

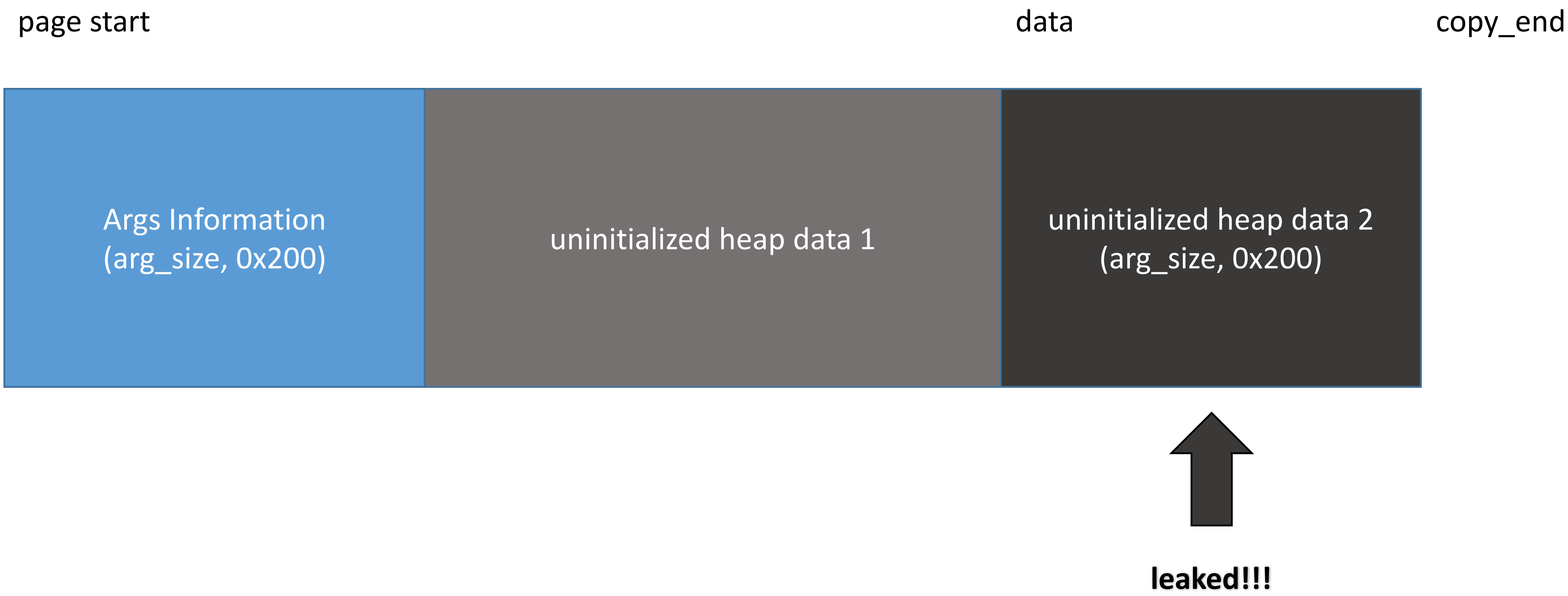
- Stack information is copied to new allocated page at offset 0 with arg_size (0x200).
- The new allocated page is not zeroed. So this operation leaves the rest of this page filled with uninitialized heap data.

At location (c):

- copy_end is round_paged,
  parameter data points to the last 0x200 bytes of the page.

At location (d):

- copyout the 0x200 bytes leaked heap information to user buffer


page start
data
copy_end
Args Information
(arg_size, 0x200)
uninitialized heap data 1
uninitialized heap data 2
(arg_size, 0x200)
leaked!!!

Exploit CVE-2018-4413 to leak ipc_port object address:

MACH_MSG_OOL_PORTS_DESCRIPTOR

```
0xffffff80256eb1b8 0xffffff80256eb1b8
0xffffff80256eb1b8 0xffffff80256eb1b8
0xffffff80256eb1b8 0xffffff80256eb1b8
0xffffff80256eb1b8 0xffffff80256eb1b8
0xffffff80256eb1b8 0xffffff80256eb1b8
0xffffff80256eb1b8 0xffffff80256eb1b8
0xffffff80256eb1b8 0xffffff80256eb1b8
0xffffff80256eb1b8 0xffffff80256eb1b8
0xffffff80256eb1b8 0xffffff80256eb1b8
0xffffff80256eb1b8 0xffffff80256eb1b8
0xffffff80256eb1b8 0xffffff80256eb1b8
0xffffff80256eb1b8 0xffffff80256eb1b8
0xffffff80256eb1b8 0xffffff80256eb1b8
```

Destroy the ports memory:

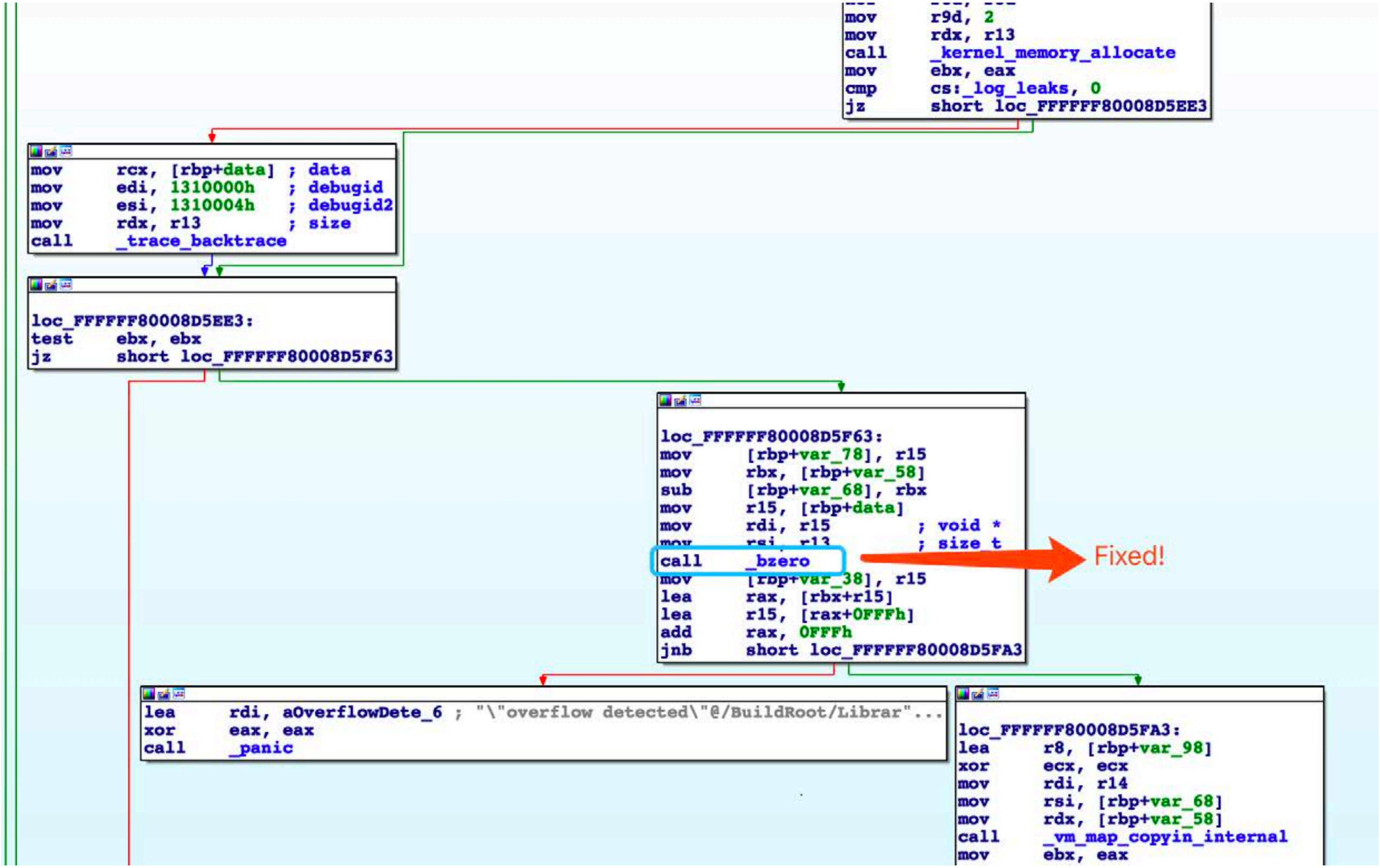
```
mach_port_destroy(mach_task_self(), q);
```

Trigger the vulnerability to leak the ports memory:

```
0xffffff80256eb1b8 0xffffff80256eb1b8
0xffffff80256eb1b8 0xffffff80256eb1b8
0xffffff80256eb1b8 0xffffff80256eb1b8
0xffffff80256eb1b8 0xffffff80256eb1b8
0xffffff80256eb1b8 0xffffff80256eb1b8
0xffffff80256eb1b8 0xffffff80256eb1b8
0xffffff80256eb1b8 0xffffff80256eb1b8
0xffffff80256eb1b8 0xffffff80256eb1b8
0xffffff80256eb1b8 0xffffff80256eb1b8
0xffffff80256eb1b8 0xffffff80256eb1b8
0xffffff80256eb1b8 0xffffff80256eb1b8
0xffffff80256eb1b8 0xffffff80256eb1b8
0xffffff80256eb1b8 0xffffff80256eb1b8
```

CVE-2018-4413

Apple fixed it by calling bzero.



CVE-2018-4425

NECP Attack Surface 1

necp_open $\xrightarrow{\text{necp\_fd\_data}}$ necp_client_action

- necp_client_add
- necp_client_remove
- necp_client_copy
- necp_client_list
- necp_client_agent_action
- necp_client_copy_agent
- necp_client_agent_use
- necp_client_copy_interface
- necp_client_copy_route_statistics
- necp_client_update_cache
- necp_client_copy_client_update

CVE-2018-4425

NECP Attack Surface 1

```
int
necp_open(struct proc *p, struct necp_open_args *uap, int *retval)
{
#pragma unused(retval)
    int error = 0;
    struct necp_fd_data *fd_data = NULL;
    struct fileproc *fp = NULL;
    int fd = -1;

    ...

    fp->f_fglob->fg_data = fd_data;

    ...
}

struct necp_fd_data {
    +0x00 u_int8_t necp_fd_type;
    +0x08 LIST_ENTRY(necp_fd_data) chain;
    +0x18 struct _necp_client_tree clients;
    +0x20 TAILQ_HEAD(_necp_client_update_list, necp_client_update) update_list;
    +0x30 int update_count;
    +0x34 int flags;
    +0x38 int proc_pid;
    +0x40 decl_lck_mtx_data(, fd_lock);
    +0x50 struct selinfo si;
};
```

necp_open assigns necp_fd_data to fg_data:

- user-mode syscall gets returned fd handle
- fd is an index to kernel fp object
- fp object contains necp_fd_data object as fg_data

CVE-2018-4425

NECP Attack Surface 1

```
int
necp_client_action(struct proc *p, struct necp_client_action_args *uap, int *retval)
{
#pragma unused(p)
    int error = 0;
    int return_value = 0;
    struct necp_fd_data *fd_data = NULL;
    error = necp_find_fd_data(uap->necp_fd, &fd_data);  ---(a)
    if (error != 0) {
        NECPLOG(LOG_ERR, "necp_client_action find fd error (%d)", error);
        return (error);
    }

    u_int32_t action = uap->action;
    switch (action) {
    ...
    }
}
```

necp_client_action operates on fg_data:

- at (a), call necp_find_fd_data to find necp_fd_data with given handle
- dispatch methods operates on necp_fd_data

```
static int
necp_find_fd_data(int fd, struct necp_fd_data **fd_data)
{
    proc_t p = current_proc();
    struct fileproc *fp = NULL;
    int error = 0;

    proc_fdlock_spin(p);
    if ((error = fp_lookup(p, fd, &fp, 1)) != 0) {
        goto done;
    }
    if (fp->f_fglob->fg_ops->fo_type != DTYPE_NETPOLICY) {  ---(b)
        fp_drop(p, fd, fp, 1);
        error = ENODEV;
        goto done;
    }
    *fd_data = (struct necp_fd_data *)fp->f_fglob->fg_data;

done:
    proc_fdunlock(p);
    return (error);
}
```

CVE-2018-4425

NECP Attack Surface 1

necp_find_fd_data finds fd_data:

- call fp_lookup to get fp of given fd
- at (b), verify if the fp is of type necp_fd_data by checking fo_type

CVE-2018-4425

NECP Attack Surface 1

Normal Process:

- necp_open creates necp_fd_data object in kernel and returns handle to user mode
- necp_client_action finds the necp_fd_data by given handle, it internally checks if corresponding fo_type equals DTYPE_NETPOLICY
- dispatch methods of necp_client_action operates on found necp_fd_data

CVE-2018-4425

NECP Attack Surface 2

necp_session_open $\xrightarrow{\text{necp\_session}}$ necp_session_action

necp_session_add_policy
necp_session_get_policy
necp_session_delete_policy
necp_session_apply_all
necp_session_list_all
necp_session_delete_all
necp_session_set_session_priority
necp_session_lock_to_process
necp_session_register_service
necp_session_unregister_service
necp_session_dump_all

```
int
necp_session_open(struct proc *p, struct necp_session_open_args *uap, int *retval)
{
#pragma unused(uap)
    int error = 0;
    struct necp_session *session = NULL;
    struct fileproc *fp = NULL;
    int fd = -1;

    ...

    fp->f_fglob->fg_data = session;

    ...
}

struct necp_session {
    +0x00    u_int8_t   necp_fd_type;
    +0x04   u_int32_t   control_unit;
    +0x08   u_int32_t   session_priority; // Descriptive priority rating
    +0x0c   u_int32_t   session_order;

    +0x10   decl_lck_mtx_data(, lock);

    +0x20   bool    proc_locked; // Messages must come from proc_uuid
    +0x21   uuid_t  proc_uuid;
    +0x34   int proc_pid;

    +0x38   bool    dirty;
    +0x40   LIST_HEAD(_policies, necp_session_policy) policies;

    +0x50   LIST_HEAD(_services, necp_service_registration) services;

    +0x60   TAILQ_ENTRY(necp_session) chain;
};
```

CVE-2018-4425

NECP Attack Surface 2

necp_session open assigns necp_session to fg_data:

- user-mode syscall gets returned fd handle
- fd is an index to kernel fp object
- fp object contains necp_session object as fg_data

CVE-2018-4425

NECP Attack Surface 2

```
int
necp_session_action(struct proc *p, struct necp_session_action_args *uap, int *retval)
{
#pragma unused(p)
    int error = 0;
    int return_value = 0;
    struct necp_session *session = NULL;
    error = necp_session_find_from_fd(uap->necp_fd, &session);  ---(aa)
    if (error != 0) {
        NECPLOG(LOG_ERR, "necp_session_action find fd error (%d)", error);
        return (error);
    }

    NECP_SESSION_LOCK(session);
    ...
}
```

necp_session_action operates on fg_data:

- at (aa), call necp_session_find_from_fd to find necp_session with given handle
- dispatch methods operates on necp_session object

```
static int
necp_session_find_from_fd(int fd, struct necp_session **session)
{
    proc_t p = current_proc();
    struct fileproc *fp = NULL;
    int error = 0;

    proc_fdlock_spin(p);
    if ((error = fp_lookup(p, fd, &fp, 1)) != 0) {
        goto done;
    }
    if (fp->f_fglob->fg_ops->fo_type != DTYPE_NETPOLICY) {  ---(bb)
        fp_drop(p, fd, fp, 1);
        error = ENODEV;
        goto done;
    }
    *session = (struct necp_session *)fp->f_fglob->fg_data;

done:
    proc_fdunlock(p);
    return (error);
}
```

CVE-2018-4425

NECP Attack Surface 2

necp_session_find_from_fd finds fd_data:

- call fp_lookup to get fp of given fd
- at (bb), verify if the fp is of type necp_session by checking fo_type

CVE-2018-4425

NECP Attack Surface 2

Normal Process:

- necp_session_open creates necp_session object in kernel and returns handle to user mode
- necp_session_action finds the necp_session by given handle, it internally checks if corresponding fo_type equals DTYPE_NETPOLICY
- dispatch methods of necp_session_action operates on found necp_session

CVE-2018-4425

Type Confusion

What we learn so far:

Attack surface 1: if fp->...->fo_type == DTYPE_NETPOLICY , fp is of type **necp_fd_data**

Attack surface 2: if fp->...->fo_type == DTYPE_NETPOLICY , fp is of type **necp_session**

necp_fd_data is totally different from **necp_session!!!**

CVE-2018-4425

Exploit : arbitrary address free

Method:

1. create necp_fd_data object and call necp_session_action to operate on it

2. create necp_session object and call necp_client_action to operate on it

CVE-2018-4425

Exploit : arbitrary address free

Step 1 call necp_open to create necp_fd_data object:

- fd_data->update_list is initialized by **TAILQ_INIT**

  +20: 0

  +28: update_list address

```
struct necp_fd_data {
    +0x00 u_int8_t necp_fd_type;
    +0x08 LIST_ENTRY(necp_fd_data) chain;
    +0x18 struct _necp_client_tree clients;
    +0x20 TAILQ_HEAD(_necp_client_update_list, necp_client_update) update_list;
    +0x30 int update_count;
    +0x34 int flags;
    +0x38 int proc_pid;
    +0x40 decl_lck_mtx_data(, fd_lock);
    +0x50 struct selinfo si;
};
```

```
int
necp_open(struct proc *p, struct necp_open_args *uap, int *retval)
{
#pragma unused(retval)
    int error = 0;
    struct necp_fd_data *fd_data = NULL;
    struct fileproc *fp = NULL;
    int fd = -1;

    if (uap->flags & NECP_OPEN_FLAG_OBSERVER) {
        if (necp_skywalk_priv_check_cred(p, kauth_cred_get()) != 0 &&
            priv_check_cred(kauth_cred_get(), PRIV_NET_PRIVILEGED_NETWORK_
            NECPLOG0(LOG_ERR, "Client does not hold necessary entitlement
            error = EACCES;
            goto done;
        }
    }

    error = falloc(p, &fp, &fd, vfs_context_current());
    if (error != 0) {
        goto done;
    }

    if ((fd_data = zalloc(necp_client_fd_zone)) == NULL) {
        error = ENOMEM;
        goto done;
    }

    memset(fd_data, 0, sizeof(*fd_data));

    fd_data->necp_fd_type = necp_fd_type_client;
    fd_data->flags = uap->flags;
    RB_INIT(&fd_data->clients);
    TAILQ_INIT(&fd_data->update_list);
    lck_mtx_init(&fd_data->fd_lock, necp_fd_mtx_grp, necp_fd_mtx_attr);
    klist_init(&fd_data->si.si_note);
    fd_data->proc_pid = proc_pid(p);

    fp->f_fglob->fg_flag = FREAD;
    fp->f_fglob->fg_ops = &necp_fd_ops;
    fp->f_fglob->fg_data = fd_data;
```

CVE-2018-4425

Exploit : arbitrary address free

necp_open ⇒

+0x20: 0
+0x28: update_list address

CVE-2018-4425

Exploit : arbitrary address free

Step 2 call necp_session_action on the object

at location (b), if session->proc_locked is false(0), session->proc_uuid and session->proc_pid will be updated.

```
int
necp_session_action(struct proc *p, struct necp_session_action_args *uap, int *retval)
{
#pragma unused(p)
    int error = 0;
    int return_value = 0;
    struct necp_session *session = NULL;
    error = necp_session_find_from_fd(uap->necp_fd, &session);
    if (error != 0) {
        NECPLOG(LOG_ERR, "necp_session_action find fd error (%d)", error);
        return (error);
    }

    NECP_SESSION_LOCK(session);

    if (session->proc_locked) {
        // Verify that the calling process is allowed to do actions
        uuid_t proc_uuid;
        proc_getexecutableuuid(current_proc(), proc_uuid, sizeof(proc_uuid));
        if (uuid_compare(proc_uuid, session->proc_uuid) != 0) {
            error = EPERM;
            goto done;
        }
    } else {
        // If not locked, update the proc_uuid and proc_pid of the session
        proc_getexecutableuuid(current_proc(), session->proc_uuid, sizeof(session->proc_uu
        session->proc_pid = proc_pid(current_proc());   ---(b)
    }

    ...

}
```

CVE-2018-4425

Exploit : arbitrary address free

- session->proc_locked at offset 0x20 overlaps update_list which is 0 in necp_fd_data.
- session->proc_uuid at offset 0x21 is updated with macho UUID
- session->proc_pid is updated with current pid

```
struct necp_session {
    +0x00     u_int8_t    necp_fd_type;
    +0x04    u_int32_t    control_unit;
    +0x08    u_int32_t    session_priority; // Descriptive priority rating
    +0x0c    u_int32_t    session_order;

    +0x10    decl_lck_mtx_data(, lock);

    +0x20    bool    proc_locked; // Messages must come from proc_uuid
    +0x21    uuid_t  proc_uuid;
    +0x34    int proc_pid;

    +0x38    bool    dirty;
    +0x40    LIST_HEAD(_policies, necp_session_policy) policies;

    +0x50    LIST_HEAD(_services, necp_service_registration) services;

    +0x60    TAILQ_ENTRY(necp_session) chain;
};
```

CVE-2018-4425

Exploit : arbitrary address free

+0x20: 0
+0x28: update_list address

necp_session_action

+0x20: 0
+0x21: UUID, low 7Bytes
+0x28: UUID, high 9Bytes
+0x34: pid

CVE-2018-4425

Exploit : arbitrary address free

Step 3 call necp_client_action on the object

- we use action 15(necp_client_copy_client_update)
- at location (f), client_update is freed
- client_update is the first element of update_list which is UUID now

```
NECP_FD_LOCK(fd_data);
struct necp_client_update *client_update = TAILQ_FIRST(&fd_data->update_list);      ---
if (client_update != NULL) {
    TAILQ_REMOVE(&fd_data->update_list, client_update, chain);  ---(c)
    VERIFY(fd_data->update_count > 0);
    fd_data->update_count--;
}
NECP_FD_UNLOCK(fd_data);

if (client_update != NULL) {
    error = copyout(client_update->client_id, uap->client_id, sizeof(uuid_t));  ---(d)
    if (error) {
        NECPLOG(LOG_ERR, "Copy client update copyout client id error (%d)", error);
    } else {
        if (uap->buffer_size < client_update->update_length) {
            NECPLOG(LOG_ERR, "Buffer size cannot hold update (%zu < %zu)", uap->buffer_
            error = EINVAL;
        } else {
            error = copyout(&client_update->update, uap->buffer, client_update->update_
            if (error) {
                NECPLOG(LOG_ERR, "Copy client update copyout error (%d)", error);
            } else {
                *retval = client_update->update_length;
            }
        }
    }

    FREE(client_update, M_NECP);    ---(f)
    client_update = NULL;
} else {
    error = ENOENT;
}

return (error);
}
```

CVE-2018-4425

Exploit : arbitrary address free

+0x20: 0
+0x28: update_list address

necp_session_action →

+0x20: 0
+0x21: UUID, low 7Bytes
+0x28: UUID, high 9Bytes
+0x34: pid

necp_client_action →

FREE Address
(0x20 – 0x28)

For Example, we set MachO UUID(16 bytes) as 41414141414141414141414141414141, here we get 0x4141414141414100 freed. We can control high 7 bytes of the address to be freed.

CVE-2018-4425

Apple Fix

Add sub type check:
necp_session has sub type 1
necp_fd_data has sub type 2

```
                public _necp_session_action
_necp_session_action proc near
                push    rbp
                mov     rbp, rsp
                push    r15
                push    r14
                push    r13
                push    r12
                push    rbx
                sub     rsp, 48h
                mov     [rbp-50h], rdx
                mov     r13, rsi
                lea     rax, ___stack_chk_guard
                mov     rax, [rax]
                mov     [rbp-30h], rax
                movsxd  rbx, dword ptr [r13+0]
                call    _current_proc
                mov     r15, rax
                lea     r12, [r15+0C0h]
                mov     rdi, r12
                call    _lck_mtx_lock_spin_always
                mov     r14d, 9
                test    rbx, rbx
                js      loc_FFFFFF80006BD34B
                mov     rax, [r15+0E8h]
                test    rax, rax
                jz      loc_FFFFFF80006BD34B
                cmp     [rax+48h], ebx
                jle     loc_FFFFFF80006BD34B
                mov     rcx, [rax]
                mov     rdx, [rcx+rbx*8]
                test    rdx, rdx
                jz      loc_FFFFFF80006BD34B
                mov     rax, [rax+30h]
                test    byte ptr [rax+rbx], 4
                jnz     loc_FFFFFF80006BD34B
                inc     dword ptr [rdx+4]
                mov     rax, [rdx+8]
                mov     rcx, [rax+28h]
                cmp     dword ptr [rcx], 9 ; DTYPE_NETPOLICY
                jnz     loc_FFFFFF80006BD336
                mov     rbx, [rax+38h]   ; fg_data
                mov     r14d, 16h
                cmp     byte ptr [rbx], 1 ; sub_type check
                jnz     loc_FFFFFF80006BD34B
                mov     rdi, r12
                call    _lck_mtx_unlock
                lea     r15, [rbx+18h]
```

Fixed!

BD235: _necp_session_action+A5 (Synchronized with Hex View-1)

```
                public _necp_client_action
_necp_client_action proc near
                push    rbp
                mov     rbp, rsp
                push    r15
                push    r14
                push    r13
                push    r12
                push    rbx
                sub     rsp, 448h
                mov     [rbp-428h], rdx
                mov     r13, rdi
                lea     rax, ___stack_chk_guard
                mov     rax, [rax]
                mov     [rbp-30h], rax
                mov     [rbp-418h], rsi
                movsxd  r14, dword ptr [rsi]
                call    _current_proc
                mov     r12, rax
                lea     r15, [r12+0C0h]
                mov     rdi, r15
                call    _lck_mtx_lock_spin_always
                mov     ebx, 9
                test    r14, r14
                js      loc_FFFFFF80006DE5E3
                mov     rax, [r12+0E8h]
                test    rax, rax
                jz      loc_FFFFFF80006DE5E3
                cmp     [rax+48h], r14d
                jle     loc_FFFFFF80006DE5E3
                mov     rcx, [rax]
                mov     rdx, [rcx+r14*8]
                test    rdx, rdx
                jz      loc_FFFFFF80006DE5E3
                mov     rax, [rax+30h]
                test    byte ptr [rax+r14], 4
                jnz     loc_FFFFFF80006DE5E3
                inc     dword ptr [rdx+4]
                mov     rax, [rdx+8]
                mov     rcx, [rax+28h]
                cmp     dword ptr [rcx], 9 ; DTYPE_NETPOLICY
                jnz     loc_FFFFFF80006DE5CE
                mov     r14, [rax+38h]   ; fg_data
                mov     ebx, 16h
                cmp     byte ptr [r14], 2 ; sub_type check
                jnz     loc_FFFFFF80006DE5E3
                mov     rdi, r15
                call    _lck_mtx_unlock
                mov     r12, [rbp-418h]
```

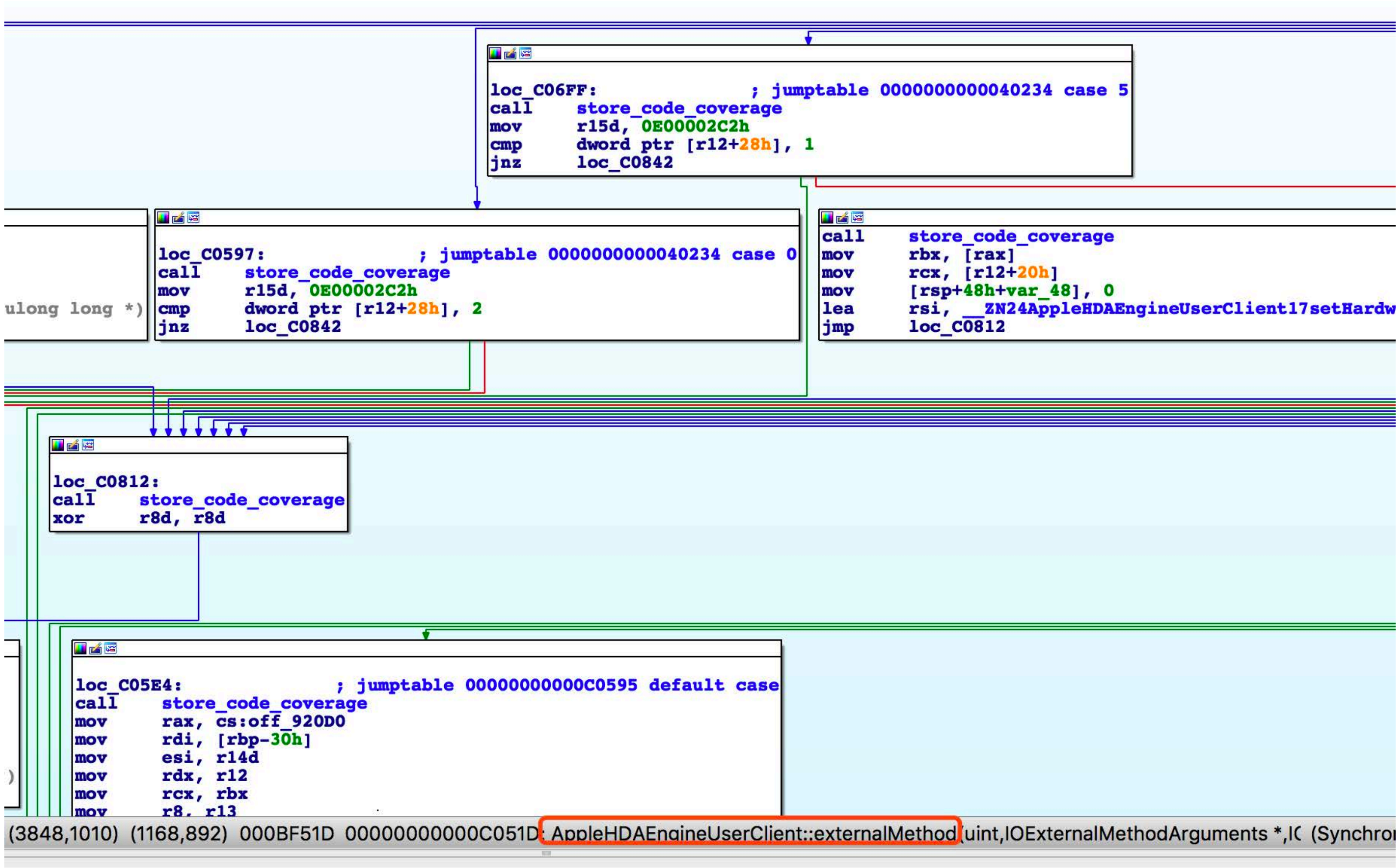
Fixed!

DE4E3: _necp_client_action+B3 (Synchronized with Hex View-1)

Future Plan of Our Fuzzing Tool

- Support kernel extension
- Support IOKit(+code coverage)
- Support Passive Fuzzing
- More and More Corpus

IOKit Code Coverage Example



macOS <= 10.14 Root

- Root = CVE-2018-4413 + CVE-2018-4425 + mach-portal
- mach_portal: all details https://bugs.chromium.org/p/project-zero/issues/detail?id=1417
- Demo(10.13.6)

More Information

- follow me on twitter: @panicall

Acknowledge

- Google Project Syzkaller[1]

ANY QUESTIONS?