PanicXNU 3.0
Who Are We

• Juwei Lin (@panicall)
• Senior Staff Engineer at TrendMicro
• A few CVEs
• CodeBlue2018 & BHEU 2018
Who Are We

• Junzhi Lu(@pwn_orz)
• Engineer at TrendMicro
• Fewer CVEs
Agenda

• What is PanicXNU

• PanicXNU 1.0 - 3.0
  • Syscall Fuzzing (PanicXNU 1.0)
  • IOKit Active Fuzzing (PanicXNU 2.0)
  • IOKit Passive Fuzzing (PanicXNU 3.0)

• Fuzzing Visualization and Daily Operation

• Case Study

• Acknowledgements
What is PanicXNU?

1) XNU Fuzzer(syscall, iokit, kext, traps…)

2) Based on Syzkaller
Darwin/XNU

Darwin/XNU is not supported at the moment.

panicall has ported (video) syzkaller to Darwin/XNU and that has found more than 50 bugs including CVE-2018-4447 and CVE-2018-4435 mentioned in Apple security updates.

Darwin/XNU is open-source and has KASAN, but no KCOV at the moment (though not required for initial support).

PureDarwin may be used to create VM images suitable for fuzzing.

https://github.com/google/syzkaller/blob/master/docs/darwin/README.md

This page was created after my #BHEU2018 talk of PanicXNU 1.0; Today I will talk about PanicXNU 3.0
Terminology

- **Pattern**: API syntax description, e.g. “int open(const char*, int)”
  - We use patterns to describe APIs which can guide generation/mutation fuzzing
  - Use patterns to generate corpus
- **Corpus**: parameters for the API, e.g. “open(‘/tmp/test’, 0)”
  - The real fuzzing data
Syscall Fuzzing(PanicXNU 1.0)


- Key Idea: 530 API patterns + sanitizers

- Found more than 50 syscall crashes (most of them are not vulnerable) including 2 for Pwn2Own macOS 10.14
Supplements

• How to add support for BSD code coverage?
• How to enable KASAN?
Q1. BSD Code Coverage

- Add Kcov module \texttt{kcov.cpp} to \texttt{/osfmk/kern}
  
- Implement your own \_\_sanitizer\_cov\_trace\_pc

- Register as a \textbf{kext module} to provide coverage data for usermode query

- Update configuration for kcov module
  
  - Update \texttt{/config/MASTER} to add kcov device: pseudo-device kcov 1 init kcov_init

  - Update \texttt{/osfmk/conf/files} to add kcov: osfmk/kern/kcov.c optional kcov

  - Update \texttt{/bsd/conf/makefile.template} to add sanitizer: add -fsanitize-coverage=trace-pc into existing CFLAGS
Register as a kext module inside osfmk:

```c
static struct cdevsw kcov_cdevsw =
{
    kcov_open,
    kcov_close,
    kcov_read,
    eno_rdwrt,
    kcov_ioctl,
    eno_stop,
    eno_reset,
    NULL,
    eno_select,
    eno_mmap,
    eno_strat,
    eno_getc,
    eno_putc,
    0
};
```
Syscall Code Coverage Problem

- No Problem for Genetic Algorithm during Fuzzing
- But not friendly for researcher to do fuzzing review
  - BasicBlock compilation option is still not good
  - Maybe try SBI or IntelPT?
Q2. Enable KASAN

- Add the following functions into `{xnu_root_folder}/san/kasan-blacklist-x86_64`:
  
  ```
  fun:machine_trace_thread_get_kva
  fun:PHYSMAP_PTOV_check
  fun:is_ept_pmap
  fun:pmap_pde
  fun:pltrace
  fun:iv_alloc
  fun:iv_dealloc
  fun:ivace_release
  fun:unsafe_convert_port_to_voucher
  fun:convert_port_to_voucher
  fun:convert_port_name_to_voucher
  fun:ipc_voucher_notify
  fun:convert_voucher_to_port
  fun:ivac_dealloc
  fun:convert_port_to_voucher_attr_control
  fun:ipc_voucher_attr_control_notify
  fun:convert_voucher_attr_control_to_port
  fun:iv_dedup
  fun:ipc_register_well_known_mach_voucher_attr_manager
  fun:mach_voucher_extract_attr_content
  fun:ivace_lookup_values
  fun:mach_voucher_extract_attr_recipe
  fun:mach_voucher_extract_all_attr_recipes
  fun:mach_voucher_debug_info
  fun:mach_voucher_attr_command
  fun:mach_voucher_attr_control_get_values
  fun:ipc_get_pthpriority_from_kmsg_voucher
  fun:ipc_voucher_send_preprocessing
  fun:ipc_voucher_prepare_processing_recipe
  ```
  
  fun:ipc_voucher_receive_postprocessing
  fun:mach_generate_activity_id
  fun:user_data_release_value
  fun:user_data_get_value
  fun:user_data_extract_content
  fun:ivace_reference_by_index
  fun:ipc_replace_voucher_value
  fun:ivace_reference_by_value
  fun:re_queue_tail
  fun:host_request_notification
  fun:OSMalloc_Tagalloc
  fun:OSMalloc_Tagrele
  fun:OSMalloc_Tagfree
  fun:enqueue_tail
  fun:remque
  fun:enable_preemption_internal
  fun:entry_queue_change_entry
  fun:semaphore_create
  fun:thread_exception_enqueue
  fun:thread_call_func_cancel
  fun:_pending_call_enqueue
  fun:waitq_select_one_locked
  fun:dequeue_head
  fun:do_backtrace32
  fun:ccdigest_final
  fun:ccctr_setctr
  fun:ccdrbg_init
  fun:ccdrbg_generate
  fun:cpu_datap
```
IOKit Active Fuzzing

• Still based on syzkaller

• Code coverage driven

• API patterns based on reverse engineering
Use Syzkaller to fuzz IOKit

• Minimum Requirements:
  • IOKit API Patterns
  • IOKIT API Executor
  • Module Code Coverage
IOKit API Patterns

• For those open source IOKit modules, we extract the API patterns directly from source code

• For those close source IOKit modules, we use reverse engineering and other methods to help extract API patterns.
IOKit API Patterns

- IOKit Attack Surface

- mach_port_subsystem
- mach_host_subsystem
- is_iokit_subsystem
- catch_exc_subsystem

- _Xio_object_get_class
- _Xio_connect_unmap_memory
- _Xio_connect_method
- _Xio_connect_async_method
- _Xio_connect_set_notification_port_64
- _Xio_service_add_notification_oov_64

Total 88 interfaces in xnu-4903.221.2
IOKit API Patterns

- Extract _Xio_connect_method patterns
- Very famous interface, very big attack surface
- Structural tables: sMethod, sDispatch
- Unstructural: switch-case, if-else …
IOKit API Patterns

- Current Support
- Example: AppleFDEKeyStore
IOKit API Patterns

- Review if our R.E. is enough
  - With help of passive corpus
  - With help of fuzzing visualization
Syzkaller has its own syscall executor which doesn’t support IOKit

- It uses `syscall` API with syscall ID to execute the fuzzing call
- It supports calling none-syscall functions which we can use to add supporting IOKit
1. Call none syscall function during fuzzing

```c
long execute_syscall(call_t* c, long a0, long a1, long a2, long a3, long a4, long a5, long a6, long a7, long a8, long a9)
{
    int ret = 0;
    int leak_check_count = 0;

    try_execute:
    if (c->call)
        ret = c->call(a0, a1, a2, a3, a4, a5, a6, a7, a8, a9);
    else
        if (c->sys nr > MAX_SYS_CALL_ID)
            return -1;
    ret = syscall(c->sys nr, a0, a1, a2, a3, a4, a5, a6, a7, a8, a9);
}
```

**support IOKit call here**

2. IOKit call name

```c
void syz_call_method(io_connect_t conn, uint32_t sel, uint64_t *sli, uint32_t slt sel, char *sti, size_t sti size, uint64_t *slo, uint32_t slo sel, char *sto, size_t sto size)
{
    uint64_t input scalar[8];
    char input structure[4096];
    uint64_t output scalar[8];
    uint32_t output scalar cnt = slo sel;
    char output structure[4096];
    size_t output structure size = sto size;

    if (sli == 0)
        sli = input scalar;
    if (sti == 0)
        sti = input structure;
    if (slo == 0)
        slo = output scalar;
    if (sto == 0)
        sto = output structure;

    IOConnectCallMethod(
        conn, sel,
        sli, slt sel, 
        sti, sti size, 
        slo, output scalar cnt, 
        sto, output structure size);
}
```

3. syz_call_method
IOKit Code Coverage

- Get code coverage from close source IOKit modules
- Possible methods: DBI, SBI, emulator, Intel PT
- We use SBI
I got the SBI idea from project PEBIL.
An example before my SBI

Before SBI(10.14.3 AppleFDEKeyStore)
An example after my SBI

After SBI(10.14.3 AppleFDEKeyStore)
After SBI(10.14.3 AppleFDEKeyStore)
• The differences between before-SBI and after-SBI

• 1 externalMethod function to 2 in IDA view

• 1 text section to 2 text sections in IDA view

• The first 5 bytes of original externalMethod function was patched to point to new function

• New externalMethod function in new text section has each shellcode call at the beginning of each block
MachO SBI WorkFlow

Intr type: o_mem OR o_near

Fix Jump Table

Fix Near Jump

Fix Operand Value

Next Instruction

Fix Loc Relocation

Fix Ext Relocation

Add 2 SBI & Shellcode

Adjust LCs Locations

Update Header

Update MachO File

Update Text

Update Relocation
Update Text

Rebuild Text Segment:
1. Handle all functions, blocks, instructions
2. For each function ABC, we will rebuild it as ABC’
3. Anyone calls ABC’, it jumps to xxxxxx
4. xxxxxx calls shellcode yyyyyy at the start of each block
5. shellcode yyyyyy records current RIP into buffer
6. Return buffer as code coverage

Some facts here:
1. Location ABC is in original TEXT section
2. Location xxxxxx is in new added SBI TEXT section
3. Location yyyyyy is in new added SBI TEXT section
<table>
<thead>
<tr>
<th>op.type</th>
<th>Description</th>
<th>Data field</th>
</tr>
</thead>
<tbody>
<tr>
<td>o_void</td>
<td>No Operand</td>
<td>None</td>
</tr>
<tr>
<td>o_reg</td>
<td>General Register (al,ax,es,ds...)</td>
<td>Reg</td>
</tr>
<tr>
<td>o_mem</td>
<td>Direct Memory Reference (DATA)</td>
<td>Addr</td>
</tr>
<tr>
<td>o_phrase</td>
<td>Memory Ref [Base Reg + Index Reg]</td>
<td>Phrase</td>
</tr>
<tr>
<td>o_displ</td>
<td>Memory Reg [Base Reg + Index Reg + Displacement]</td>
<td>Phrase+Addr</td>
</tr>
<tr>
<td>o_imm</td>
<td>Immediate Value</td>
<td>Value</td>
</tr>
<tr>
<td>o_far</td>
<td>Immediate Far Address (CODE)</td>
<td>Addr</td>
</tr>
<tr>
<td>o_near</td>
<td>Immediate Near Address (CODE)</td>
<td>Addr</td>
</tr>
</tbody>
</table>
1. Fix Near Jump
76 —> 0F 86

2. Fix Operand
D2 —> C4 FF FF FF

Before SBI(10.14.3 AppleFDEKeyStore)

After SBI(10.14.3 AppleFDEKeyStore)
## Full Near -> Long Jump List

<table>
<thead>
<tr>
<th>Mnem</th>
<th>Long Jump Opcode</th>
</tr>
</thead>
<tbody>
<tr>
<td>jo</td>
<td>\x0f\x80</td>
</tr>
<tr>
<td>jno</td>
<td>\x0f\x81</td>
</tr>
<tr>
<td>jb\jc\jnae</td>
<td>\x0f\x82</td>
</tr>
<tr>
<td>jae\nb\jnc</td>
<td>\x0f\x83</td>
</tr>
<tr>
<td>jz\je</td>
<td>\x0f\x84</td>
</tr>
<tr>
<td>jnz\jne</td>
<td>\x0f\x85</td>
</tr>
<tr>
<td>jbe\jna</td>
<td>\x0f\x86</td>
</tr>
<tr>
<td>ja\jnbe</td>
<td>\x0f\x87</td>
</tr>
<tr>
<td>js</td>
<td>\x0f\x88</td>
</tr>
<tr>
<td>jns</td>
<td>\x0f\x89</td>
</tr>
<tr>
<td>jp\jpe</td>
<td>\x0f\x8a</td>
</tr>
<tr>
<td>jnp\jpo</td>
<td>\x0f\x8b</td>
</tr>
<tr>
<td>jl\jnge</td>
<td>\x0f\x8c</td>
</tr>
<tr>
<td>jge\jnl</td>
<td>\x0f\x8d</td>
</tr>
<tr>
<td>jle\jng</td>
<td>\x0f\x8e</td>
</tr>
<tr>
<td>jg\jnle</td>
<td>\x0f\x8f</td>
</tr>
<tr>
<td>jmp</td>
<td>\xe9</td>
</tr>
</tbody>
</table>
Jump table is a range of data in text code:

1) we must recognize it as data instead of code
2) We must fix it during SBI or the new function will jump back to original function which may panic
• How to recognize it?

  • Operand value of the instruction(type: o_mem or o_near) points to somewhere inside text section
  
  • But doesn’t point to any existing function block
  
  • All the jump table item must points to existing function block

  • Jump table size: find next symbol after the table start, the table end lies just before the found symbol

• How to fix it during SBI?

  • Re-use the jump table instead of copying the table into new SBI text
  
  • Fix every table entry making it points to new code branch in SBI text
10.14.3 AppleFDEKeyStore after SBI

1. The above code is in new added SBI text, not in the original text section (address around 0x9c4b, not 0x435b)
2. The jump table is re-used at location 0x4580
3. But all table entries are updated to point to branches in new added SBI text
1. At file offset 0x1860, it calls _IOFree
2. But the address of _IOFree is 0 in raw file context
3. Actually _IOFree is relocated during module loading
1. 0x1861 is described in Dynamic Symbol Table in MachO file.
2. We need update the Dynamic Symbol Table to let the entry points to SBI text instead of original text section.
3. If not update, the same function(setPassphraseWithUserID) in SBI text will ‘call 0x0’ instead of ‘call _IOFree’
Fix Relocation

After SBI (10.14.3 AppleFDEKeyStore)

1. 0xb540 changed to 0x12540 since new SBI section added
2. 0x1861 is updated to 0xbb61
Update MachO file structure

- Add SBI TEXT section and SBI DATA section
- Add Shellcode
- Update MachO header, e.g. load commands(LC_SYMTAB, LC_DYSYMTAB, LC_DYLD_INFO, LC_DYLD_INFO_ONLY, LC_FUNCTION_STARTS, LC_DATA_IN_CODE)
Add SBI TEXT & DATA Sections
Add Shellcode:

1) just a stub to call real recording function
2) Save & Restore all registers
3) You can implement your recording function in a standard kernel driver
SBI Summary

- Most important work: Address and Offset Fixup
  - Fix o_mem and o_near instructions
  - Fix relocations
  - Fix file structure
- Add shellcode to record the RIP value
  - Use asm
- As simple as you can
Add 2 Segments:

- __TEXT\x00\x00PANICALL
- __DATA\x00\x00PANICALL

Where to add the 2 segments?

- Before __LINKEDIT segment
- Can we put them at the end of load commands?
• My Answer:

• If SBI an usermode MachO file, Yes.

• If SBI a kernelmode MachO file, NO! Or system won’t boot.
During initialization, linkedit load command can be updated to a smaller size.

This will make something unexpected if you append new segments after LINKEDIT.
Kernel will calculate the needed virtual memory before module loading, for example, the MachO in the picture needs 0x4800 bytes memory since there is a 0x800 bytes optimization inside LINKEDIT.

If we append the 2 new segments at the end, the SBI DATA segment will be mapped exceeding 0x4800. Then panic!
Limitation

- Not support FAT macho
- Not support 32 bit macho
- Macho header must have free space for 2 more load commands (SBI TEXT and SBI DATA)
- Not perfect solution for jump table identification in text section
- Hardware Module Protection: Disable
### SBI Partial Status in Fusion VM

<table>
<thead>
<tr>
<th>Module</th>
<th>macOS 10.14</th>
<th>macOS 10.13.6</th>
<th>Module</th>
<th>macOS 10.14</th>
<th>macOS 10.13.6</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOThunderboltFamily</td>
<td>OK</td>
<td>OK</td>
<td>IOUSBMassStorageDriver</td>
<td>Unknown</td>
<td>OK</td>
</tr>
<tr>
<td>IOSurface</td>
<td>OK</td>
<td>OK</td>
<td>IOATAFamily</td>
<td>Unknown</td>
<td>OK</td>
</tr>
<tr>
<td>IOUSBFamily</td>
<td>OK</td>
<td>OK</td>
<td>IOSMBusFamily</td>
<td>Unknown</td>
<td>OK</td>
</tr>
<tr>
<td>IOGraphicsFamily</td>
<td>OK</td>
<td>OK</td>
<td>IOSCSIArchitectureModelFamily</td>
<td>Unknown</td>
<td>OK</td>
</tr>
<tr>
<td>AppleHDA</td>
<td>OK</td>
<td>OK</td>
<td>IOSCSIArchitectureModelFamily</td>
<td>Unknown</td>
<td>OK</td>
</tr>
<tr>
<td>DspFuncLib</td>
<td>OK</td>
<td>OK</td>
<td>IOACPIFamily</td>
<td>Unknown</td>
<td>OK</td>
</tr>
<tr>
<td>IOBluetoothFamily</td>
<td>Failure</td>
<td>OK</td>
<td>IOSlowAdaptiveClockingFamily</td>
<td>Unknown</td>
<td>OK</td>
</tr>
<tr>
<td>IOResultFamily</td>
<td>OK</td>
<td>OK</td>
<td>IOSerialFamily</td>
<td>Unknown</td>
<td>OK</td>
</tr>
<tr>
<td>IOStorageFamily</td>
<td>OK</td>
<td>OK</td>
<td>IOCDStorageFamily</td>
<td>Unknown</td>
<td>OK</td>
</tr>
<tr>
<td>IOAudioFamily</td>
<td>OK</td>
<td>OK</td>
<td>IONDRTVSupport</td>
<td>Unknown</td>
<td>OK</td>
</tr>
<tr>
<td>IOAVBFamily</td>
<td>OK</td>
<td>OK</td>
<td>IOAHCIFamily</td>
<td>Unknown</td>
<td>Failure</td>
</tr>
<tr>
<td>IOTimeSyncFamily</td>
<td>OK</td>
<td>OK</td>
<td>IOBDStorageFamily</td>
<td>Unknown</td>
<td>Failure</td>
</tr>
<tr>
<td>IODVDStorageFamily</td>
<td>OK</td>
<td>OK</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VMwareGfx</td>
<td>OK</td>
<td>OK</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IOHIDFamily</td>
<td>OK</td>
<td>OK</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IONetworkingFamily</td>
<td>OK</td>
<td>OK</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
If failure on your physical machine, goto recover mode

If you get a failure boot after using the tool, try the following steps to recover your system.

Take `AppleIntelKBLGraphics.kext` as an example:

1. recover the binary

   ```bash
   mv AppleIntelKBLGraphics AppleIntelKBLGraphics_new
   mv AppleIntelKBLGraphics_org AppleIntelKBLGraphics
   ```

2. add owner

   ```bash
   chown -R root:wheel AppleIntelKBLGraphics.kext
   ```

3. build kextcache

   ```bash
   kextcache -arch x86_64 -kernel /Volumes/Macintosh\HD/System/Library/Kernels/kernel -prelinked-kernel /Volumes/Macintosh\HD/System/Library/PrelinkedKernels/prelinkedkernel -volume-root /Volumes/Macintosh\HD/ /Volumes/Macintosh]\HD/System/Library/Extensions/
   /Volumes/Macintosh\HD/Library/Extensions/
   ```

4. reboot
IOKit Passive Fuzzing

- Passive Fuzzing commonly uses hooks to mutate parameters during normal function call.

- We just use hooks to dump parameters to mysql & mongodb

- Syzkaller uses concept ‘program’ so that we need convert the collected parameters to ‘program’
<table>
<thead>
<tr>
<th>id</th>
<th>service_name</th>
<th>type</th>
<th>class_name</th>
<th>xnu_version_start</th>
<th>submit_time</th>
</tr>
</thead>
<tbody>
<tr>
<td>76</td>
<td>IOBluetoothDevice</td>
<td>0</td>
<td>IOBleutoothDeviceUserClient</td>
<td>1802</td>
<td>2018-12-27 17:40:12</td>
</tr>
<tr>
<td>77</td>
<td>IOBluetoothL2CAPChannel</td>
<td>0</td>
<td>IOBleutoothL2CAPChannelUserClient</td>
<td>1802</td>
<td>2018-12-27 17:40:12</td>
</tr>
<tr>
<td>78</td>
<td>AppleHSSbluetoothDevice</td>
<td>4737348</td>
<td>IOHIDLibUserClient</td>
<td>1802</td>
<td>2018-12-27 17:40:13</td>
</tr>
<tr>
<td>79</td>
<td>AppleHSSbluetoothHIDDr</td>
<td>4737348</td>
<td>IOHIDLibUserClient</td>
<td>1802</td>
<td>2018-12-27 17:40:13</td>
</tr>
<tr>
<td>80</td>
<td>AppleHIDKeyboardEvent</td>
<td>1212761156</td>
<td>IOHIDEventServiceUserClient</td>
<td>1802</td>
<td>2018-12-27 17:40:14</td>
</tr>
<tr>
<td>81</td>
<td>AppleMultitouchDevice</td>
<td>0</td>
<td>AppleMultitouchDeviceUserClient</td>
<td>1802</td>
<td>2018-12-27 17:41:06</td>
</tr>
<tr>
<td>82</td>
<td>AppleActuatorDevice</td>
<td>0</td>
<td>AppleActuatorDeviceUserClient</td>
<td>1802</td>
<td>2018-12-27 17:41:06</td>
</tr>
<tr>
<td>83</td>
<td>AMDRadeonX4000_AMDriver</td>
<td>3</td>
<td>AMDRadeonX4000_AMDDriverUser</td>
<td>1802</td>
<td>2018-12-28 11:19:16</td>
</tr>
<tr>
<td>84</td>
<td>IOUserEthernetResource</td>
<td>0</td>
<td>IOUserEthernetResourceUser</td>
<td>1802</td>
<td>2019-01-01 18:46:28</td>
</tr>
<tr>
<td>85</td>
<td>AppleMobileFileIntegrity</td>
<td>0</td>
<td>AppleMobileFileIntegrityUser</td>
<td>1802</td>
<td>2019-01-16 14:50:45</td>
</tr>
<tr>
<td>86</td>
<td>VMwareFramebuffer</td>
<td>1</td>
<td>IOFramebufferSharedUserClient</td>
<td>1802</td>
<td>2019-01-16 14:56:17</td>
</tr>
<tr>
<td>87</td>
<td>AppleUSBInterface</td>
<td>0</td>
<td>IOUSBInterfaceUserClientV3</td>
<td>1802</td>
<td>2019-01-16 15:35:36</td>
</tr>
<tr>
<td>88</td>
<td>AppleUSBInterface</td>
<td>0</td>
<td>IOUSBInterfaceUserClient</td>
<td>1802</td>
<td>2019-01-16 15:46:19</td>
</tr>
<tr>
<td>89</td>
<td>AppleIntelFramebuffer</td>
<td>1</td>
<td>IOFramebufferSharedUserClient</td>
<td>1802</td>
<td>2019-01-16 15:46:45</td>
</tr>
<tr>
<td>90</td>
<td>AudioAUUCDriver</td>
<td>0</td>
<td>AudioAUUC</td>
<td>1802</td>
<td>2019-01-16 15:48:35</td>
</tr>
<tr>
<td>91</td>
<td>AppleUpstreamUserClient</td>
<td>0</td>
<td>AppleUpstreamUserClient</td>
<td>1802</td>
<td>2019-01-16 15:48:35</td>
</tr>
<tr>
<td>92</td>
<td>IntelAccelerator</td>
<td>256</td>
<td>IAACLVideoContextVEBox</td>
<td>1802</td>
<td>2019-01-16 15:58:17</td>
</tr>
<tr>
<td>93</td>
<td>IntelAccelerator</td>
<td>257</td>
<td>IAACLVideoContextMedia</td>
<td>1802</td>
<td>2019-01-16 15:58:17</td>
</tr>
<tr>
<td>94</td>
<td>IntelAccelerator</td>
<td>256</td>
<td>IAACLVideoContextMain</td>
<td>1802</td>
<td>2019-01-16 15:58:17</td>
</tr>
<tr>
<td>95</td>
<td>AppleHPMIECS</td>
<td>0</td>
<td>AppleHPMIECSUserClient</td>
<td>1802</td>
<td>2019-01-17 09:24:20</td>
</tr>
<tr>
<td>96</td>
<td>AppleHSSIPHIHIDDriver</td>
<td>4737348</td>
<td>IOHIDLibUserClient</td>
<td>1802</td>
<td>2019-01-17 09:24:30</td>
</tr>
<tr>
<td>97</td>
<td>AirPort_BrcmNIC</td>
<td>1833193043</td>
<td>mDNSOffloadUserClient</td>
<td>1802</td>
<td>2019-01-17 10:09:44</td>
</tr>
<tr>
<td>98</td>
<td>AppleEmbeddedOSSupport</td>
<td>0</td>
<td>AppleEmbeddedOSSupportHIDDriver</td>
<td>1802</td>
<td>2019-01-17 10:12:02</td>
</tr>
<tr>
<td>99</td>
<td>IONVMsBlockStorageDevice</td>
<td>12</td>
<td>IONVMeSMARTUserClient</td>
<td>1802</td>
<td>2019-01-17 11:45:44</td>
</tr>
<tr>
<td>100</td>
<td>IOHIDUserDevice</td>
<td>4737348</td>
<td>IOHIDLibUserClient</td>
<td>1802</td>
<td>2019-01-21 14:55:36</td>
</tr>
<tr>
<td>101</td>
<td>AppleUSBDevice</td>
<td>0</td>
<td>IOAudioEngineUserClient</td>
<td>1802</td>
<td>2019-01-22 18:02:47</td>
</tr>
<tr>
<td>102</td>
<td>com_epson_driver_EPS</td>
<td>0</td>
<td>IOAudioEngineUserClient</td>
<td>1802</td>
<td>2019-01-23 10:12:00</td>
</tr>
<tr>
<td>103</td>
<td>ZoomAudioEngine</td>
<td>0</td>
<td>IOAudioEngineUserClient</td>
<td>1802</td>
<td>2019-01-23 10:12:00</td>
</tr>
<tr>
<td>id</td>
<td>class_name</td>
<td>selector</td>
<td>scalar_input_count</td>
<td>scalar_output_count</td>
<td>struct_input_size</td>
</tr>
<tr>
<td>----</td>
<td>---------------------------------</td>
<td>----------</td>
<td>--------------------</td>
<td>---------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>447</td>
<td>IOutputStreamUserClient</td>
<td>27</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>448</td>
<td>IOHIDLibUserClient</td>
<td>11</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>449</td>
<td>SCSITaskUserClient</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>450</td>
<td>IOUSBDeviceUserClientV2</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>451</td>
<td>AudioAUUC</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>32</td>
</tr>
<tr>
<td>452</td>
<td>AudioAUUC</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>40</td>
</tr>
<tr>
<td>453</td>
<td>AppleNVMoSMARTUser</td>
<td>6</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>454</td>
<td>AppleNVMoSMARTUser</td>
<td>7</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>455</td>
<td>AudioAUUC</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>41</td>
</tr>
<tr>
<td>456</td>
<td>IGAconVideoContextMain</td>
<td>258</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>457</td>
<td>AudioAUUC</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>33</td>
</tr>
<tr>
<td>458</td>
<td>AudioAUUC</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>36</td>
</tr>
<tr>
<td>459</td>
<td>CCDataPipeUserClient</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>16</td>
</tr>
<tr>
<td>460</td>
<td>IOInterfaceUserClient</td>
<td>42</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>461</td>
<td>IOInterfaceUserClient</td>
<td>41</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>462</td>
<td>IGAconDevice</td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>187</td>
</tr>
<tr>
<td>463</td>
<td>IOSurfaceRootUserClient</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>472</td>
</tr>
<tr>
<td>464</td>
<td>IGAconDevice</td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>187</td>
</tr>
<tr>
<td>465</td>
<td>IOBluetoothL2CAPChannel</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>466</td>
<td>IOBluetoothL2CAPChannel</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>467</td>
<td>IOHIDLibUserClient</td>
<td>15</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>468</td>
<td>IOSurfaceRootUserClient</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>4040</td>
</tr>
<tr>
<td>469</td>
<td>IOUSBInterfaceUserClient</td>
<td>27</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>470</td>
<td>IOBluetoothL2CAPChannel</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>14</td>
</tr>
<tr>
<td>471</td>
<td>IOSurfaceRootUserClient</td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>472</td>
<td>IOSurfaceRootUserClient</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>272</td>
</tr>
<tr>
<td>473</td>
<td>IOSurfaceRootUserClient</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>608</td>
</tr>
<tr>
<td>474</td>
<td>IOSurfaceRootUserClient</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>464</td>
</tr>
</tbody>
</table>

**IOKit Connect Call**
IOKit Connect Call Data
Passive API S Corpus

- Syzkaller uses ‘program’ to fuzz instead of single API, ‘program’ is a sequence of API calls.
Passive API S Corpus

• Convert: Record a sequence of API calls, not just a single API call.
Passive API S Corpus

• Make: Use the above data to make syzkaller friendly corpus
IOKit Fuzz Summary

- We can combine the active and passive IOKit fuzz now:
  - We have the IOKit open & call *syntax description* by R.E.
  - We have the passive sequence *corpus* by hooks
  - We have the *code coverage* of close source IOKit modules
  - We extend the ability of syzkaller to support fuzzing IOKit
The combination advantages:

• Passive corpus can be used to fuzz system alone. It doesn’t need the active API patterns developed by researchers.

• Passive corpus can help review if active patterns are correct by reverse engineering.

• Passive corpus give the chances that even the patterns are wrong, we can still fuzz in the right direction based on genetic algorithm.
IOKit Fuzzing Architecture

- Syntax Description
- Active Corpus
- Active Generate-Based Fuzzing
  - Active Mutation-Based Fuzzing
  - CodeCoverage Filter
  - Memory Corpus
  - open source
  - close source
- Client Hooks
- Passive API S Corpus
- Owner
- DashBoard
Fuzzing Visualization

Coverage: the first class citizen
Exciting updates has exciting crashes

---

**Panic Logs**

- **Earnings Overview**

---

**Panic Report**

```
2019-04-25 10:51:34

Anonymous UUID: 03989409-f721-8af3-ba4e-3cb80685b6ee

Thu Mar 14 22:11:47 2019

*** Panic Report ***
panic(cpu 0) called at 0x54d000a2dc2c8d: Kernel trap at 0xffff7f6a9d1f9, type 4-page fault, registers:
    CR0: 0x0000000080000000, CR2: 0x0000000000000000, CR3: 0x000000007076d4a, CR4: 0x0000000000010006
    RAX: 0xffff7f6a9d1f6, RBX: 0x0000000000000000, RCX: 0xffff7f6a9d1f8, RDX: 0xffff7f6a9d1f9
    RSI: 0xffff7f6a9d222, RBP: 0xffff7f6a9d223, R10: 0xffff7f6a9d224, R11: 0xffff7f6a9d225
    R12: 0xffff7f6a9d226, R13: 0xffff7f6a9d227, R14: 0xffff7f6a9d228, R15: 0xffff7f6a9d229
    RSIP: 0xffff7f6a9d22a, RBP: 0xffff7f6a9d22b, R10: 0xffff7f6a9d22c, R11: 0xffff7f6a9d22d
    R12: 0xffff7f6a9d22e, R13: 0xffff7f6a9d22f, R14: 0xffff7f6a9d230, R15: 0xffff7f6a9d231
    FPU: 0x0000000000000000, Error code: 0x0000000000000000, Fault CPU: 0x0000000000000000, PL: 0, VF: 0

Backtrace (CPU 0), Frame : Return Address
0xffff86d95f1c0: 0xffff869093e80 mach.kern : _handleDebugger_trap + 0x48d
0xffff86d95f09e10: 0xffff869093e80 mach.kern : _kbpsTrap + 0x153
0xffff86d95f09e50: 0xffff869093e80 mach.kern : _kernel_trap + 0x48e
0xffff86d95f09e50: 0xffff869093e80 mach.kern : _return_from_trap + 0xe0
0xffff86d95f09e50: 0xffff869093e80 mach.kern : _kernel_trap + 0x197
0xffff86d95f09e50: 0xffff869093e80 mach.kern : _kernel_trap + 0x197
0xffff86d95f09e50: 0xffff869093e80 mach.kern : _kernel_trap + 0x197
```
Coverage in mind
## Coverage in mind

### IOAcceleratorFamily2_new
- **Target Binary**: IOAcceleratorFamily2_new
- **Coverage Name**: IOAcceleratorFamily2_new
- **Coverage File**: IOAcceleratorFamily2_new.drcov.log
- **Database Coverage**: 6.99%
- **Table Coverage**: 6.99%
- **Timestamp**: Thu Mar 28 10:21:57 2019

### AppleFDEKeyStore_new
- **Target Binary**: AppleFDEKeyStore_new
- **Coverage Name**: AppleFDEKeyStore_new
- **Coverage File**: AppleFDEKeyStore_new.drcov.log
- **Database Coverage**: 12.87%
- **Table Coverage**: 12.87%
- **Timestamp**: Fri Mar 29 07:50:54 2019

### AppleMobileFileIntegrity_new
- **Target Binary**: AppleMobileFileIntegrity_new
- **Coverage Name**: AppleMobileFileIntegrity_new
- **Coverage File**: AppleMobileFileIntegrity_new.drcov.log
- **Database Coverage**: 2.44%
- **Table Coverage**: 2.44%
- **Timestamp**: Fri Mar 29 07:53:18 2019

### AppleHDA
- **Target Binary**: AppleHDA
- **Coverage Name**: AppleHDA
- **Coverage File**: AppleHDA.drcov.log
- **Database Coverage**: 0.17%
- **Table Coverage**: 0.17%
- **Timestamp**: Fri Mar 29 03:59:54 2019
After update

For Programmer:
- Fix old bugs
- Introduce new feature, and new API

For Security Researcher:
- Do diff to find n-Day
- Do diff to find new BUG and new Attack Surface
Diff on key structure

- Binaries
- Rules in toml
- Scripted Parsing within IDA
- JSON1
- JSON2
- JSON Patch (RFC 6902)
Fuzzing Daily Operation

We are not developing this to see just pictures.

```
bash3.2$ mount /dev/brain
Permission Denied
bash3.2$ sudo mount /dev/brain
No mount point
```
After looking lots of charts

Further reverse engineering

Quenching: Modify API Syntax Description for fuzz and run again

Add some tweaks for the fuzzing environment.
Demo: AppleHDA

Limitation:
1. Check entitlement
2. Check Boot Arguments

Tweak:
1. Add an entitlement (private? fail?)
2. Change boot args
Demo: An exciting update

A diff between High Sierra (10.13.6) & Mojave (10.14)

In facet, the implementation code of these mig subsystem already lies in kernel since High Sierra but remains unreferenced.

It was then added as the element of array “mig_e” since Mojave. Which means now it comes to work.

So we add some new API Syntax Description
You need to restart your computer. Hold down the Power button for several seconds or press the Restart button.

Veuillez redémarrer votre ordinateur. Maintenez la touche de démarrage enfoncée pendant plusieurs secondes ou bien appuyez sur le bouton de réinitialisation.

Sie müssen Ihren Computer neu starten. Halten Sie dazu die Einschalttaste einige Sekunden gedrückt oder drücken Sie die Neustart-Taste.

コンピュータを再起動する必要があります。パワーボタンを数秒間押し続けるか、リセットボタンを押してください。

Case Study
int
shmget(struct proc *p, struct shmget_args *uap, int32_t *retval)
{
    int segnum, mode, error;
    int shmget_ret = 0;

    /* Auditing is actually done in shmget_allocate_segment() */
    SYSV_SHM_SUBSYS_LOCK;

    if ((shmget_ret = shminit())) {
        goto shmget_out;
    }

    mode = uap->shflg & ACCESSPERMS;
    if (uap->key != IPC_PRIVATE) {
        again:
            segnum = shm_find_segment_by_key(uap->key);
            if (segnum == 0) {
                error = shmget_existing(uap, mode, segnum, retval);
                if (error == EAGAIN)
                    goto again;
                shmget_ret = error;
                goto shmget_out;
            }
            if ((uap->shflg & IPC_CREAT) == 0) {
                shmget_ret = ENOENT;
                goto shmget_out;
            }
    }
    shmget_ret = shmget_allocate_segment(p, uap, mode, retval);  ---1.b

    shmget_out:
    SYSV_SHM_SUBSYS_UNLOCK();
    return shmget_ret;
}

int
shminit(void)
{
    size_t sz;
    int i;

    if (!shm_inited) {
        /*
          * we store internally 64 bit, since if we didn't, we would
          * be unable to represent a segment size in excess of 32 bits
          * with the (struct shm_id_ds)->shm_segsz field; also, POSIX
          * dictates this filed be a size_t, which is 64 bits when
          * running 64 bit binaries.
          */
        if (os_mul_overflow(shminfo.shmmni, sizeof(struct shm_id_kernel), &sz)) {
            return ENOMEM;
        }
        MALLOC(shmseg, struct shm_id_kernel *, sz, M_SHM, M_WAITOK);  ---2.a
        if (shmseg == NULL) {
            return ENOMEM;
        }
        for (i = 0; i < shminfo.shmmni; i++) {
            shmseg[i].u.shm_perm.mode = SHMSEG_FREE;
            shmseg[i].u.shm_perm._seq = 0;
        }
    }  #if CONFIG_MACF
    mac_sysvshm_label_init(&shmseg[i]);
    #endif
    shm_last_free = 0;
    shm_nused = 0;
    shm_commited = 0;
    shm_inited = 1;
}

return 0;
Field at Offset 0x2a is not set;
Actually it is the padding of structure.

asm code around location 3.a

```c
CVE-2018-4435

static int
shmget_allocate_segment(struct proc *p, struct shmget_args *uap, int mode,
            int *retnal)
{
    int i, segnum, shmid;
    kauth_cred_t cred = kauth_cred_get();
    struct shmid_kernel *shmseg;
    struct sh hm_handle *shm_handle;
    kern_return_t kret;
    mach_vm_size_t total_size, size, alloc_size;
    void *mem_object;
    struct shm_handle *shm_handle_next, **shm_handle_next_p;

    ...
    shmseg = &shmsagep[segnum];

    ...

    shmseg->u.shm_segsz = uap->size;
    shmseg->u.shm_cpid = p->p_pid;
    shmseg->u.shm_lpid = shmseg->u.shm_nattch = 0;  //--------(3.a)
    shmseg->u.shm_atime = shmseg->u.shm_dtime = 0;

    ...
}
```
The alignment makes the hole.
CVE-2018-4435

• Type: Kernel heap memory disclosure

• Thought: Can be used to search the XNU with such a pattern:
  • Kernel structure memory allocated without clear
  • Not all bytes are filled, e.g. padding, union structure
  • copy back to user mode
At location 1.a, max_count here is 17. At location 1.b, count can be 11.
MACH_PORT_INFO_EXT_COUNT is 17; reserved[6] is not filled and disclosed.

This case was fixed internally before my submission so no CVE.
CVE-2019-8529

```
mov    rax, [r15+20h] ; scalarInput
mov    esi, [rax]    ; int
mov    rdi, r12     ; this
call   __ZN18SCITaskUserClient20ReleaseTaskReferenceEi ;
jmp    loc_1B4

; __int64 __fastcall SCSITaskUserClient::ReleaseTaskReference(SCSITaskUserClient *this, int)
public __ZN18SCITaskUserClient20ReleaseTaskReferenceEi
__ZN18SCITaskUserClient20ReleaseTaskReferenceEi proc near
push   rbp
mov    rbp, rsp
push   r15
push   r14
push   r13
push   r12
push   rbx
push   rax
mov    r13d, esi
mov    r15, rdi
movsxd  rbx, r13d
mov    edi, 5275011h ; unsigned __int64
xor    ecx, ecx     ; unsigned __int64
mov    rsi, r15     ; unsigned __int64
mov    rdx, rbx     ; unsigned __int64
call   __ZL19RecordSTUCTimeStampmmmmmm ; RecordSTUCTimeStamp(ulong, ulong, ulong, ulong, ulong)
lea    r14, [r15+rbx*4+170h] ------(a)
xor    edi, edi
mov    esi, 1
mov    rdx, r14
call   _OSCompareAndSwap
```
Acknowledgements

- Syzkaller
- Lighthouse
- My Two Interns: 董正禹, 杨朝明 for their help of IOKit patterns and Fuzzing visualization
- Mickey Jin of TrendMicro for his help of KASAN enable
Q & A