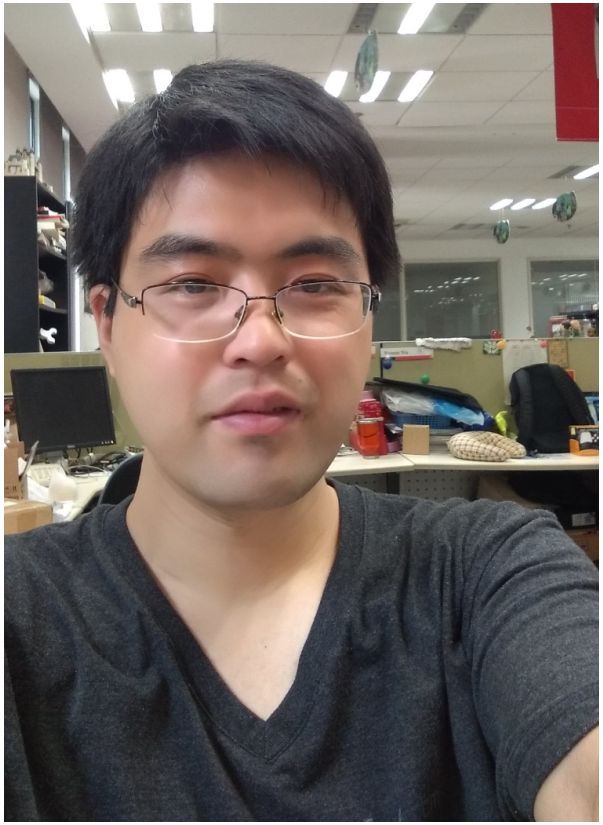
Hourglass Fuzz: A Quick Bug Hunting Method

MOONY LI, TODD HAN, LANCE JIANG, LILANG WU

@TREND MICRO

Agenda

- Introduction
- User Space Fuzzing
- Kernel Space Fuzzing
- Demo



Moony Li

- 9 years security
- Sandcastle
- Deep Discovery
- Exploit Detection
- Mac/Windows Kernel
- iOS/Android Vulnerability

Twitter: @Flyic

Lilang Wu

- 4 years of system security
- Mobile Advanced Threat Research of TrendMicro
- Mac/iOS Vulnerability/Malware

Twitter: @Lilang_Wu

Lilang Wu

- Mobile security research
- Kernel vulnerability
- Android reverse engineer

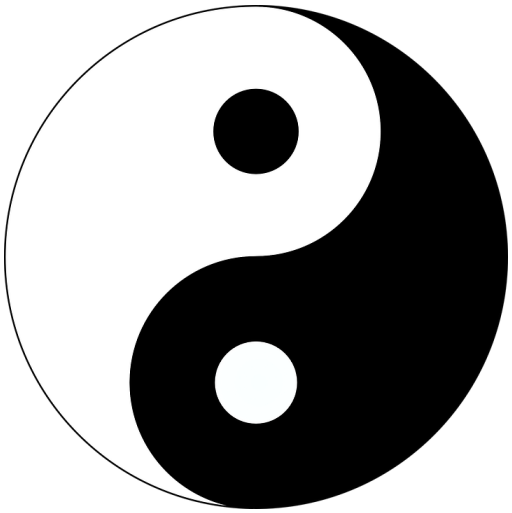
Twitter: @Binas

Todd Han

- Exploit Detection
- Linux Kernel
- Android Vulnerability

Twitter: @exiahan

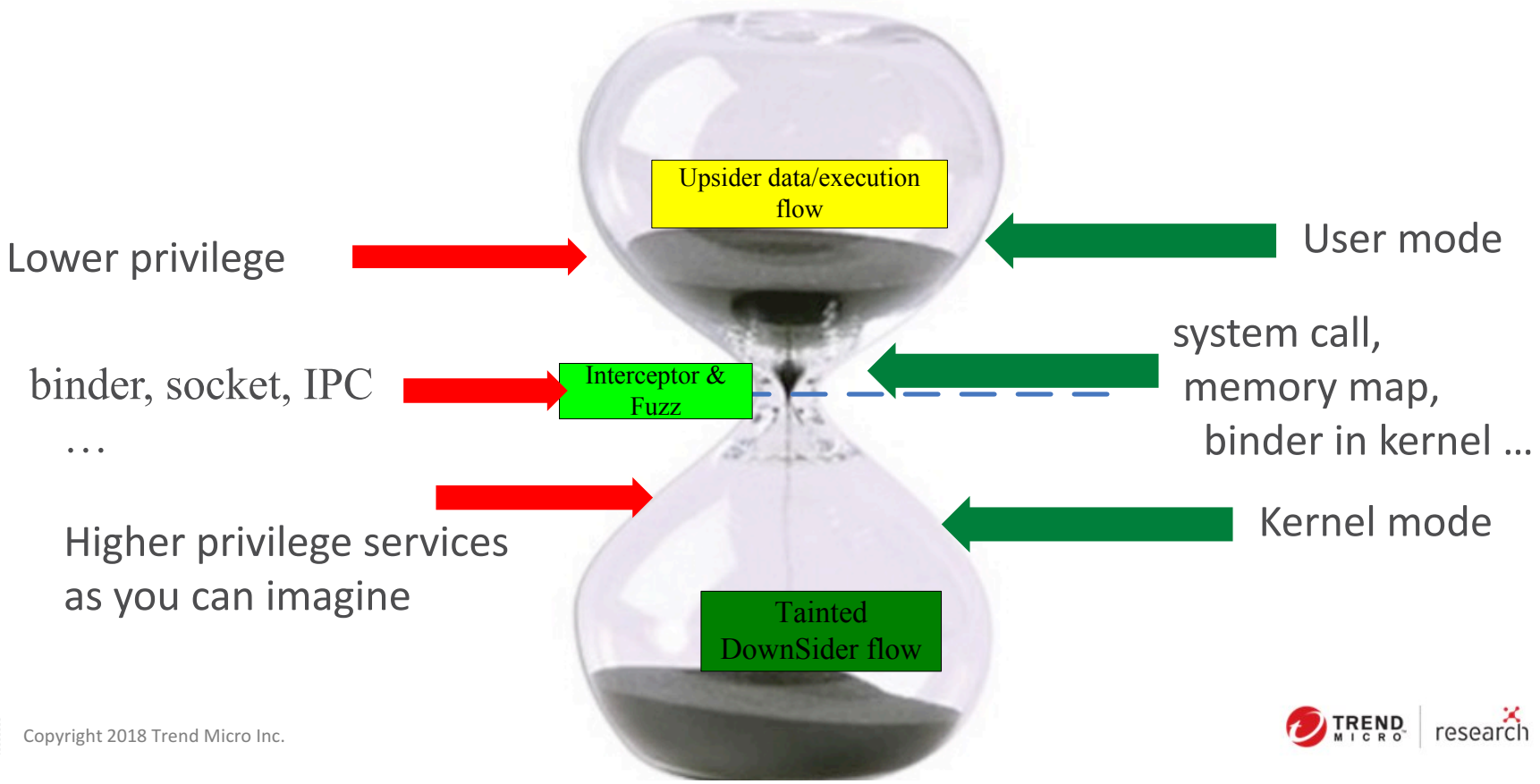
Pain point



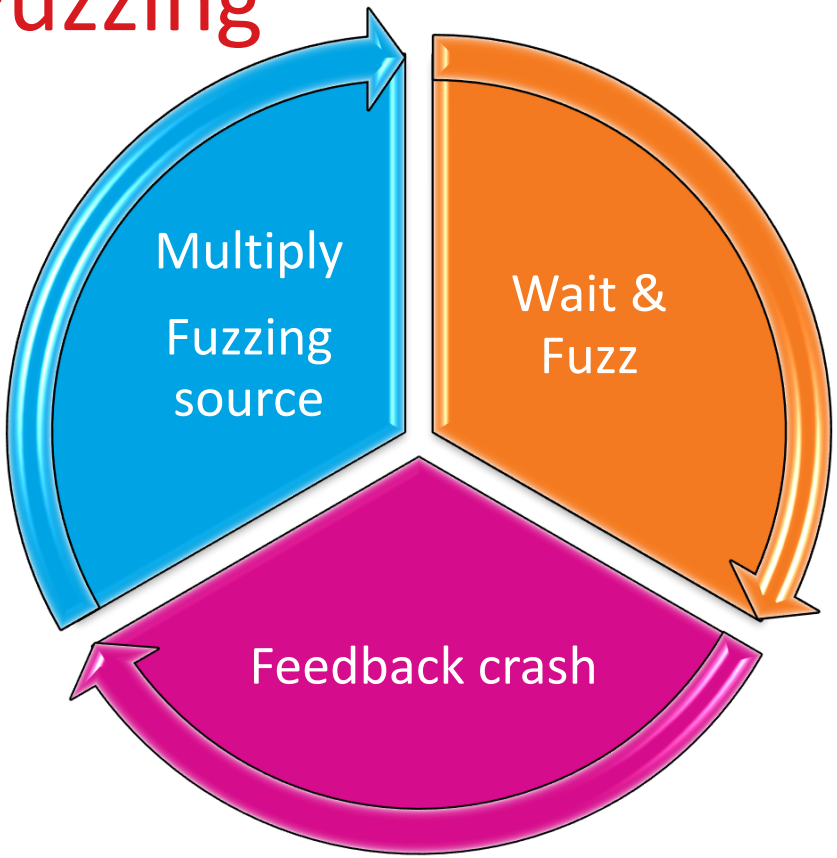
陰陽互生

	Key method	Wait Time	Find new attack interface	Deep coverage
Syzkaller	Code coverage feedback	Long	No	No or unknown
AFL	Code coverage feedback	Long	No	No or unknown
Code Review	Person by Person knowledge	Unknown	Yes	Yes
Hourglass fuzz	Hook and taint	Short	Yes	Yes

Hourglass Fuzzing Philosophy



Hourglass Fuzzing



User Space Fuzzing

- 1. Attack Interface
  - Choosing Strategy
  - Modules We Select
- 2. Sanitizer Support
- 3. Fuzzing Strategy
- 4. Best Practice

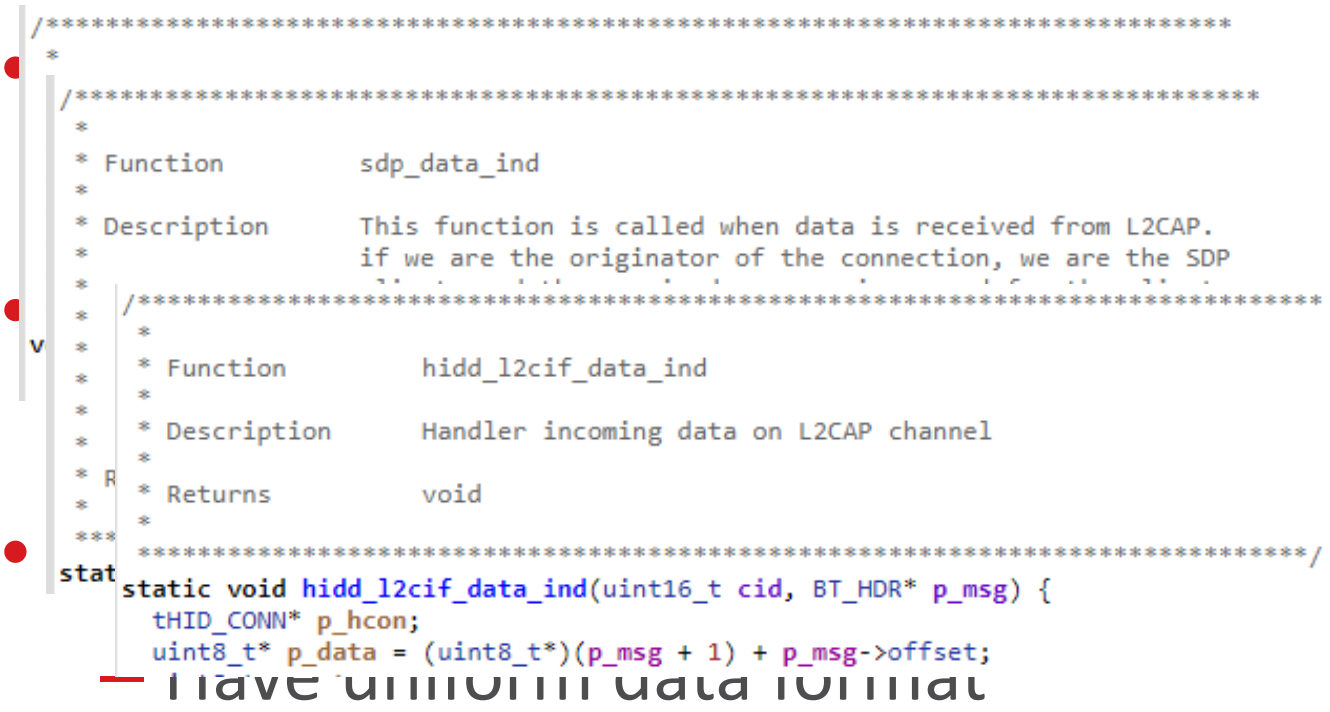
Attack Interface --- selection

- Relatively High Priority
  - System service related process
- Have Vulnerabilities Before
  - Bluetooth, NFC…
- Expose Ports for Receiving Data
  - Have uniform data format

Attack Interface --- selection

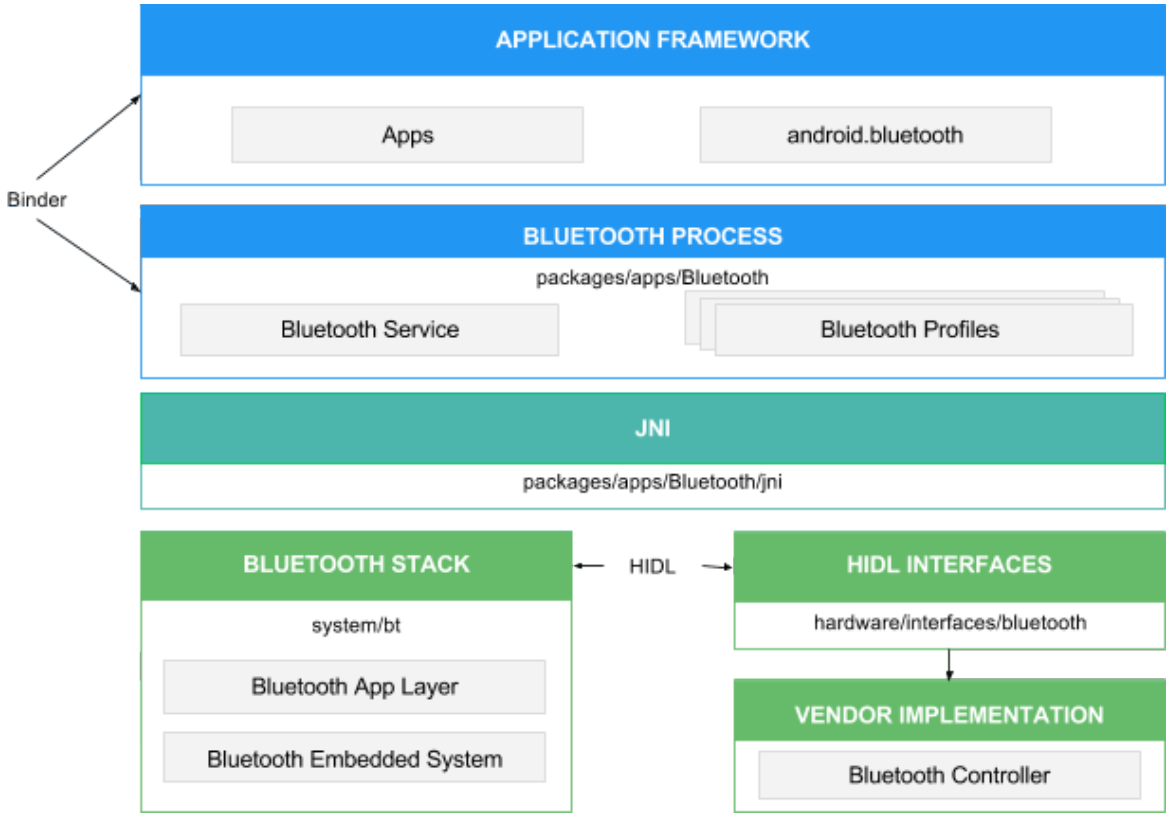
- Relatively High Priority
  - System service related process
- Have Vulnerabilities Before
  - **Bluetooth, NFC**…
- Expose Ports for Receiving Data
  - Have uniform data format

Attack Interface --- selection



- Have uniform data format

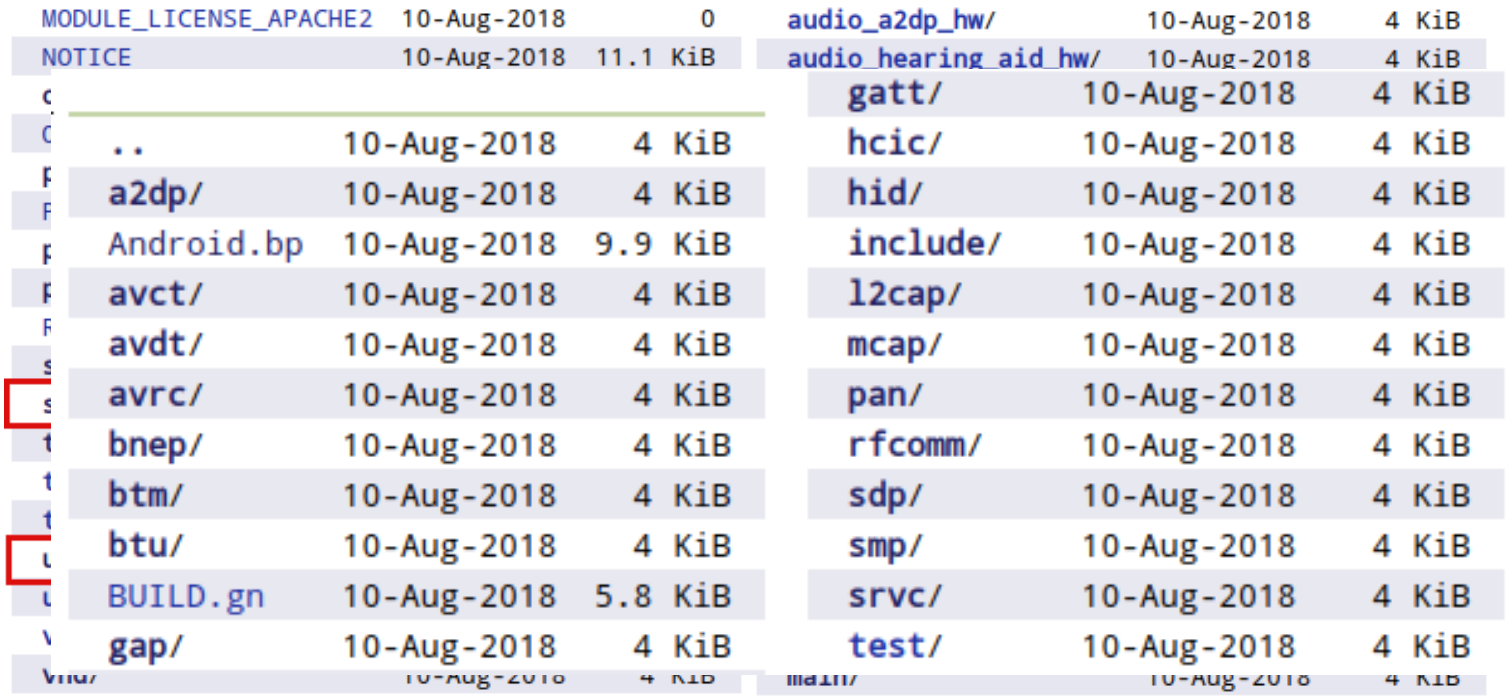
Bluetooth – Architecture in brief



Bluetooth -- Module View

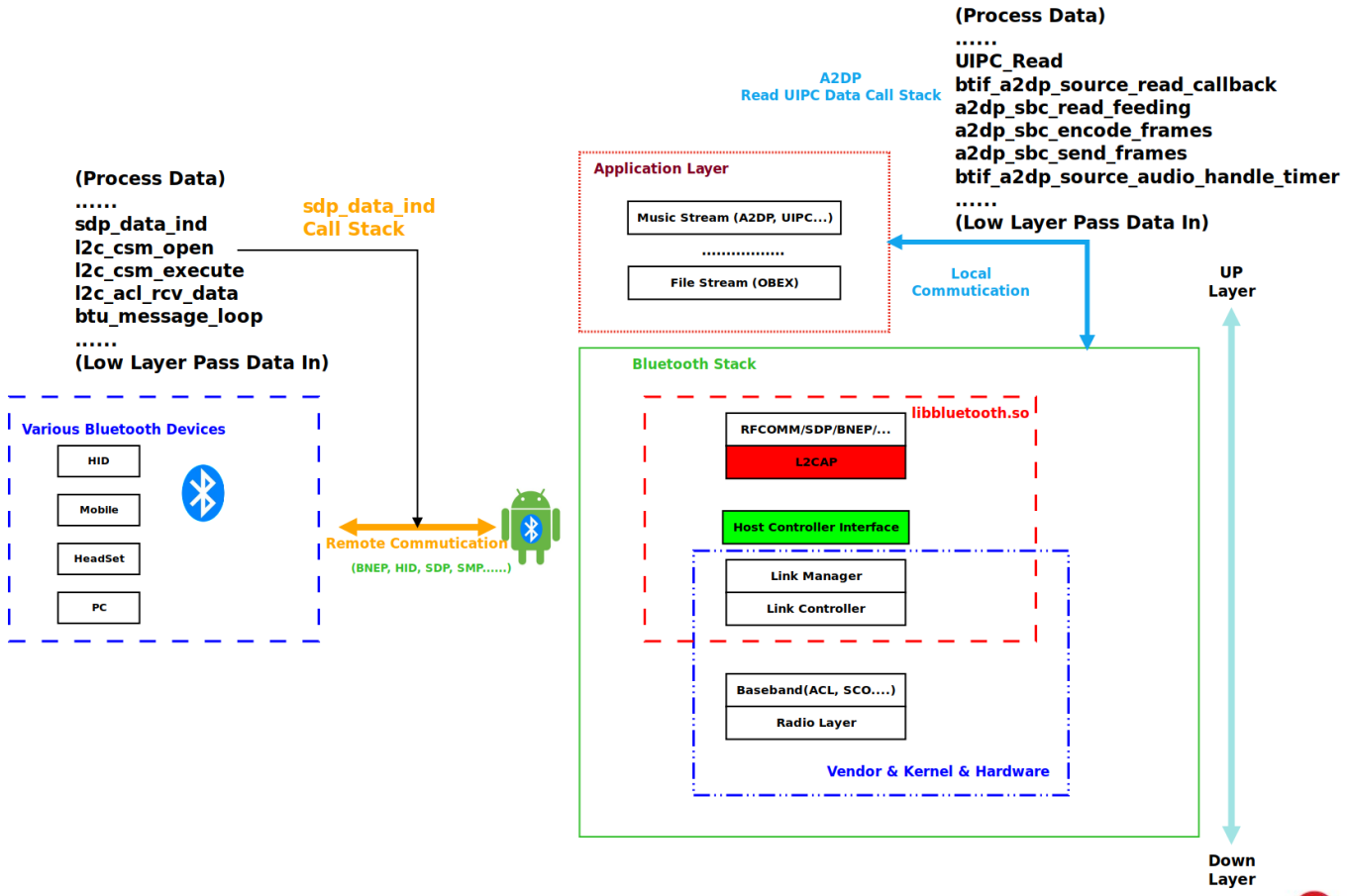
```
marlin:/ # cat /proc/1422/maps | grep blue
74d0b12000-74d0b68000 r-xp 00000000 fd:00 2116                   /system/lib64/android.hardware.bluetooth@1.0.so
74d0b7a000-74d0b7e000 r--p 0005c000 fd:00 2116                   /system/lib64/android.hardware.bluetooth@1.0.so
74d0b7e000-74d0b7f000 rw-p 00060000 fd:00 2116                   /system/lib64/android.hardware.bluetooth@1.0.so
74d0b80000-74d158b000 r-xp 00000000 fd:00 2178                   /system/lib64/libbluetooth.so
74d15a6000-74d15b4000 r--p 00a16000 fd:00 2178                   /system/lib64/libbluetooth.so
74d15b4000-74d15b5000 rw-p 00a24000 fd:00 2178                   /system/lib64/libbluetooth.so
74d16cb000-74d1720000 r-xp 00000000 fd:00 2221                   /system/lib64/android.hardware.bluetooth.a2dp@1.0.so
74d1733000-74d1737000 r--p 0005c000 fd:00 2221                   /system/lib64/android.hardware.bluetooth.a2dp@1.0.so
74d1737000-74d1738000 rw-p 00060000 fd:00 2221                   /system/lib64/android.hardware.bluetooth.a2dp@1.0.so
74d1748000-74d1758000 r-xp 00000000 fd:00 2142                   /system/lib64/libbluetooth-binder.so
74d1775000-74d1777000 r--p 0001e000 fd:00 2142                   /system/lib64/libbluetooth-binder.so
74d1777000-74d1778000 rw-p 00020000 fd:00 2142                   /system/lib64/libbluetooth-binder.so
74d1787000-74d17d9000 r-xp 00000000 fd:00 2277                   /system/lib64/libbluetooth_jni.so
74d17ef000-74d17f1000 r--p 0005e000 fd:00 2277                   /system/lib64/libbluetooth_jni.so
74d17f1000-74d17f3000 rw-p 00060000 fd:00 2277                   /system/lib64/libbluetooth_jni.so
74e3378000-74e3380000 rw-s 00000000 103:13 1049164               /data/user_de/0/com.android.bluetooth/databases/btopp.db-shm
756bcad000-756bccd000 r--s 00000000 00:0e 14866                  /dev/__properties__/u:object_r:bluetooth_prop:s0
marlin:/ #
```

Bluetooth -- Source Tree

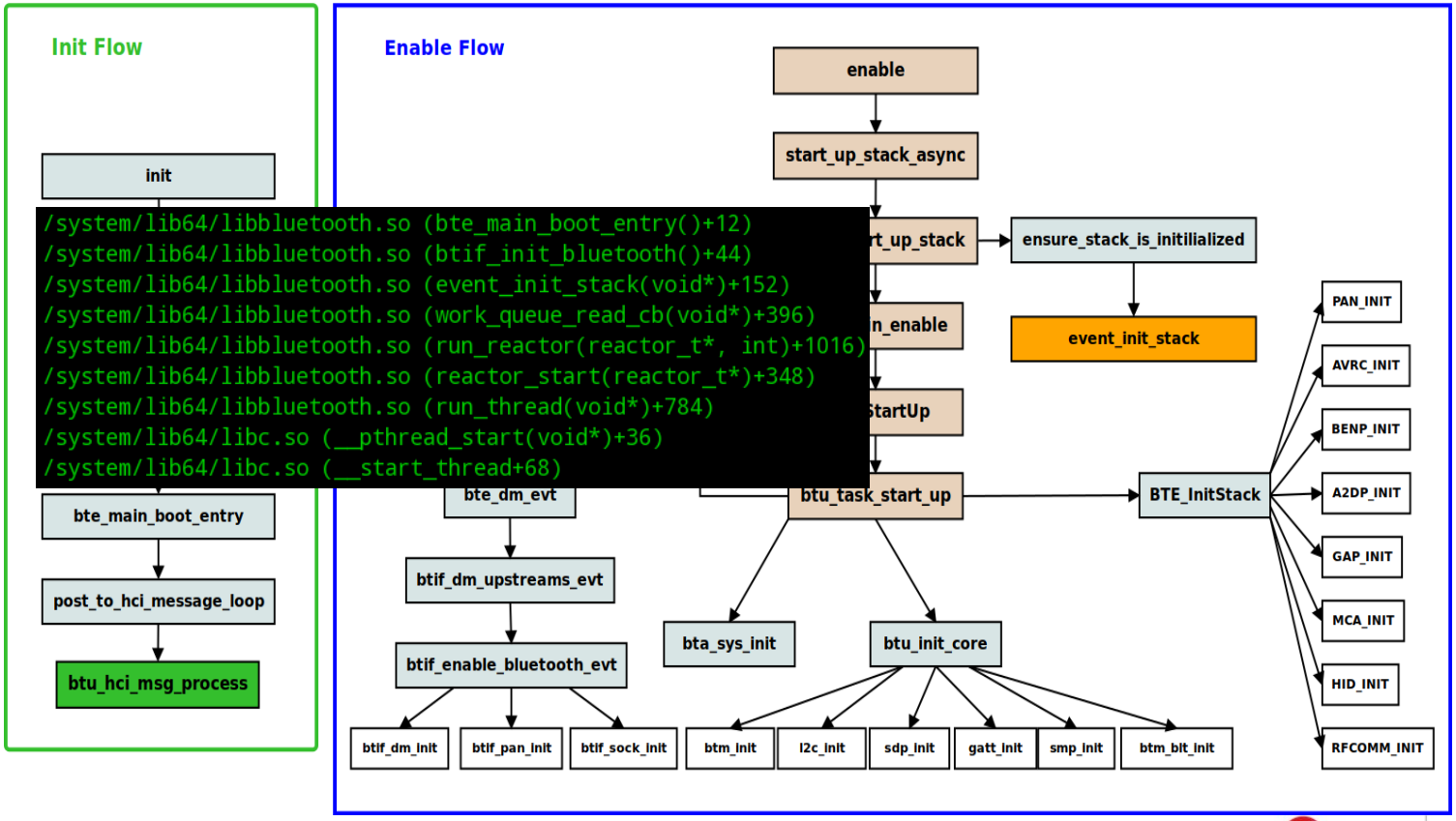


MODULE_LICENSE_APACHE2	10-Aug-2018	0
NOTICE	10-Aug-2018	11.1 KiB
..	10-Aug-2018	4 KiB
a2dp/	10-Aug-2018	4 KiB
Android.bp	10-Aug-2018	9.9 KiB
avct/	10-Aug-2018	4 KiB
avdt/	10-Aug-2018	4 KiB
avrc/	10-Aug-2018	4 KiB
bnep/	10-Aug-2018	4 KiB
btm/	10-Aug-2018	4 KiB
btu/	10-Aug-2018	4 KiB
BUILD.gn	10-Aug-2018	5.8 KiB
gap/	10-Aug-2018	4 KiB
audio_a2dp_hw/	10-Aug-2018	4 KiB
audio_hearing_aid_hw/	10-Aug-2018	4 KiB
gatt/	10-Aug-2018	4 KiB
hcic/	10-Aug-2018	4 KiB
hid/	10-Aug-2018	4 KiB
include/	10-Aug-2018	4 KiB
l2cap/	10-Aug-2018	4 KiB
mcap/	10-Aug-2018	4 KiB
pan/	10-Aug-2018	4 KiB
rfcomm/	10-Aug-2018	4 KiB
sdp/	10-Aug-2018	4 KiB
smp/	10-Aug-2018	4 KiB
srvc/	10-Aug-2018	4 KiB
test/	10-Aug-2018	4 KiB

Bluetooth -- Architecture in detail



Bluetooth -- Startup work flow



Bluetooth ---- Packet Structure & Common Dispatcher

BT_HDR

- Common Header Structure for each packet

```
typedef struct {
  uint16_t event;
  uint16_t len;
  uint16_t offset;
  uint16_t layer_specific;
  uint8_t data[];
} BT_HDR;
```

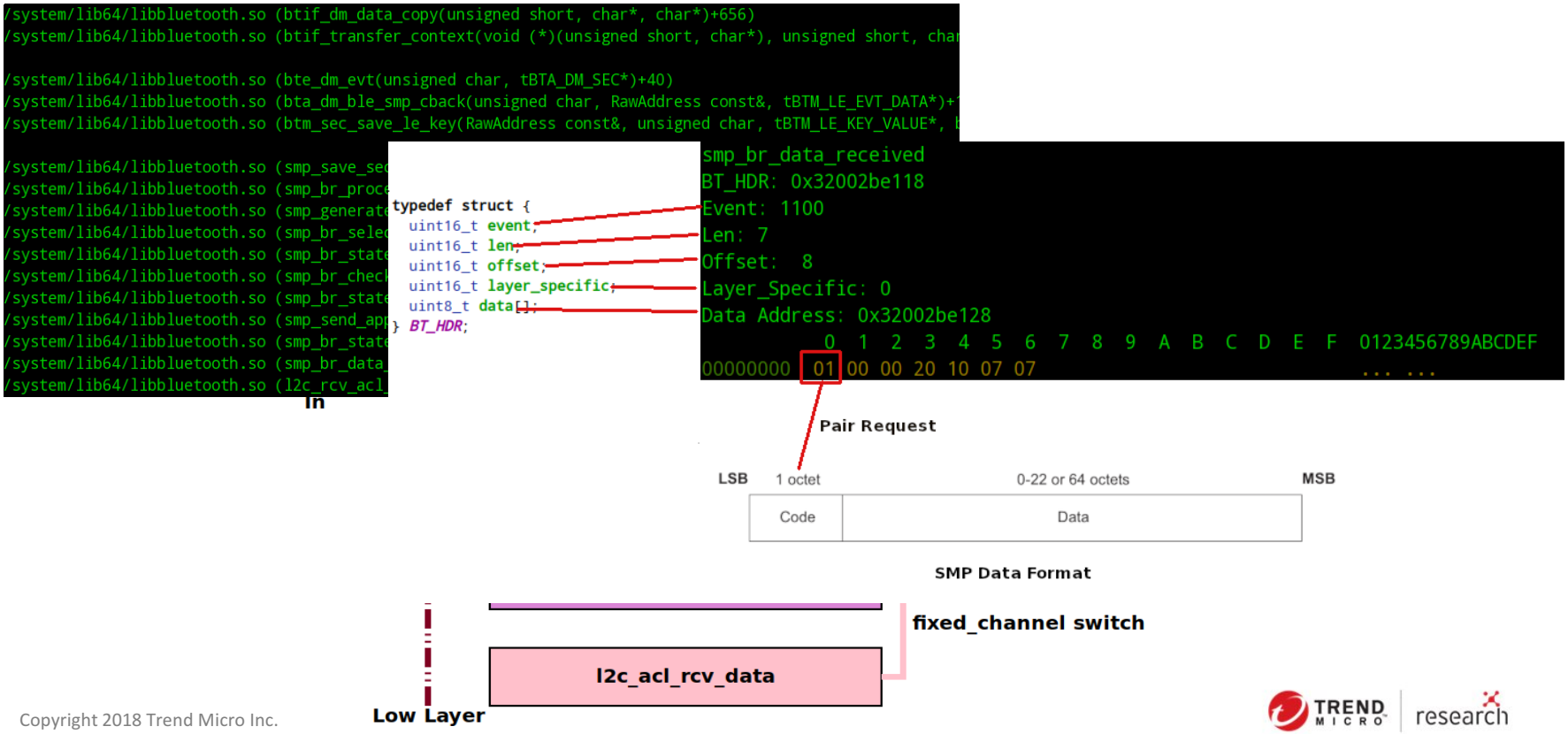
```
uint8_t* p = (uint8_t*)(p_msg + 1) + p_msg->c
```

HCI Process

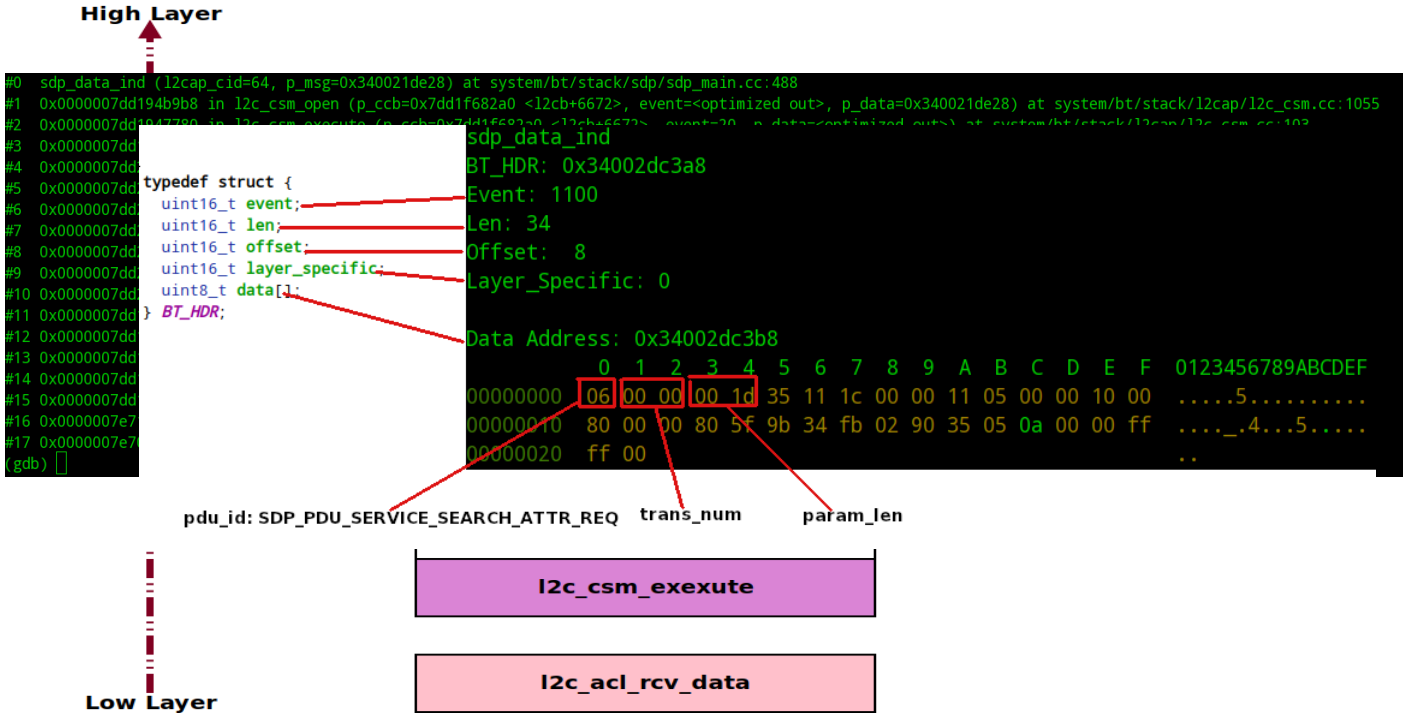
- Common Dispatcher

```
void btu_hci_msg_process(BT_HDR* p_msg) {
  /* Determine the input message type. */
  switch (p_msg->event & BT_EVT_MASK) {
    case BT_EVT_TO_BTU_HCI_ACL:
      /* All Acl Data goes to L2CAP */
      l2c_rcv_acl_data(p_msg);
      break;
```

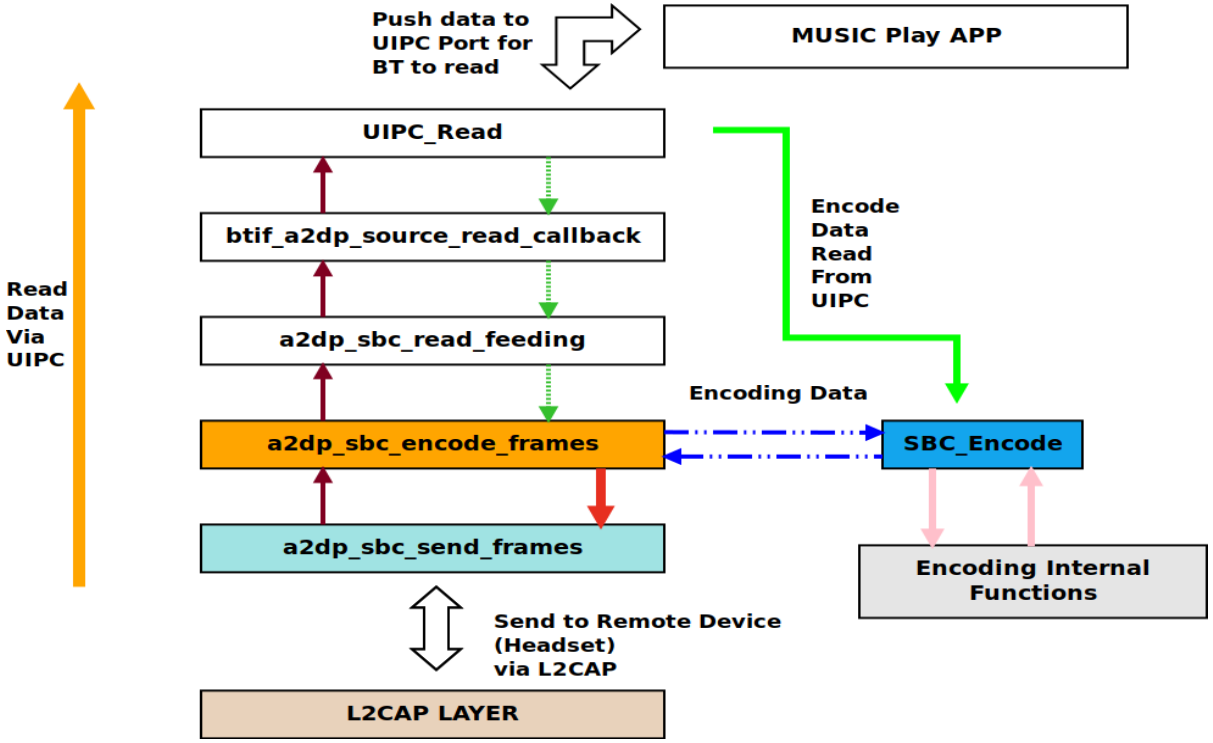
Bluetooth --- Remote Communication



Bluetooth --- Remote Communication



Bluetooth --- Local Communication



Bluetooth --- Local Communication

```
#0  SBC_Encode (pstrEncParams=0x78b6d577f4 <a2dp_sbc_encoder_cb+28>, input=0x78b6d57ca8 <a2dp_sbc_encoder_cb+1232>, output=0x1000000000 "")
    at system/bt/embdrv/sbc/encoder/srce/sbc_encoder.c:47
#1  0x00000078b65ad594 in a2dp_sbc_encode_frames (nb_frame=0 '\000') at system/bt/stack/a2dp/a2dp_sbc_encoder.cc:554
#2  a2dp_sbc_send_frames (timestamp_us=<optimized out>) at system/bt/stack/a2dp/a2dp_sbc_encoder.cc:417
#3  0x00000078b647a014 in btif_a2dp_source_audio_handle_timer () at system/bt/btif/src/btif_a2dp_source.cc:905
#4  0x00000078b6f93894 in base::debug::TaskAnnotator::RunTask(char const*, base::PendingTask*) () from target:/system/lib64/libchrome.so
#5  0x00000078b6fdfd64 in base::MessageLoop::RunTask(base::PendingTask*) () from target:/system/lib64/libchrome.so
#6  0x00000078b6fe0834 in base::MessageLoop::DeferOrRunPendingTask(base::PendingTask) () from target:/system/lib64/libchrome.so
#7  0x00000078b6fe17c8 in base::MessageLoop::DoWork() () from target:/system/lib64/libchrome.so
#8  0x00000078b6fe526c in base::MessagePumpDefault::Run(base::MessagePump::Delegate*) () from target:/system/lib64/libchrome.so
#9  0x00000078b6fdf2d8 in base::MessageLoop::RunHandler() () from target:/system/lib64/libchrome.so
#10 0x00000078b7036584 in base::RunLoop::Run() () from target:/system/lib64/libchrome.so
#11 0x00000078b6477fd0 in BtWorkerThread::Run (this=0x78b6ce7c60 <btif_a2dp_source_thread>) at system/bt/btif/src/btif_a2dp_source.cc:238
#12 0x00000078b681e228 in work_queue_read_cb (context=<optimized out>) at system/bt/osi/src/thread.cc:251
#13 0x00000078b6817c90 in run_reactor (reactor=0x330000fa48, iterations=0) at system/bt/osi/src/reactor.cc:282
#14 0x00000078b6817814 in reactor_start (reactor=0x330000fa48) at system/bt/osi/src/reactor.cc:125
#15 0x00000078b681d160 in run_thread (start_arg=<optimized out>) at system/bt/osi/src/thread.cc:221
#16 0x0000007954208 83c in __pthread_start(void*) () from target:/system/lib64/libc.so
#17 0x00000079541a7578 in __start_thread () from target:/system/lib64/libc.so
```

Bluetooth --- Target functions

- Packets Processing Cal
  - Each profile
  - Each protocol

```
RFCOMM_BufDataInd
PORT_DataInd
mca_tc_data_ind
mca_l2c_data_ind_cback
avct_l2c_data_ind_cback
gap_data_ind
hidd_l2cif_data_ind;
hidh_l2cif_data_ind
sdp_data_ind
pan_data_ind_cb
avdt_ad_tc_data_ind
avdt_l2c_data_ind_cback
bnep_data_ind
smp_data_received
smp_br_data_received
gatt_data_process
l2c_rcv_acl_data
```

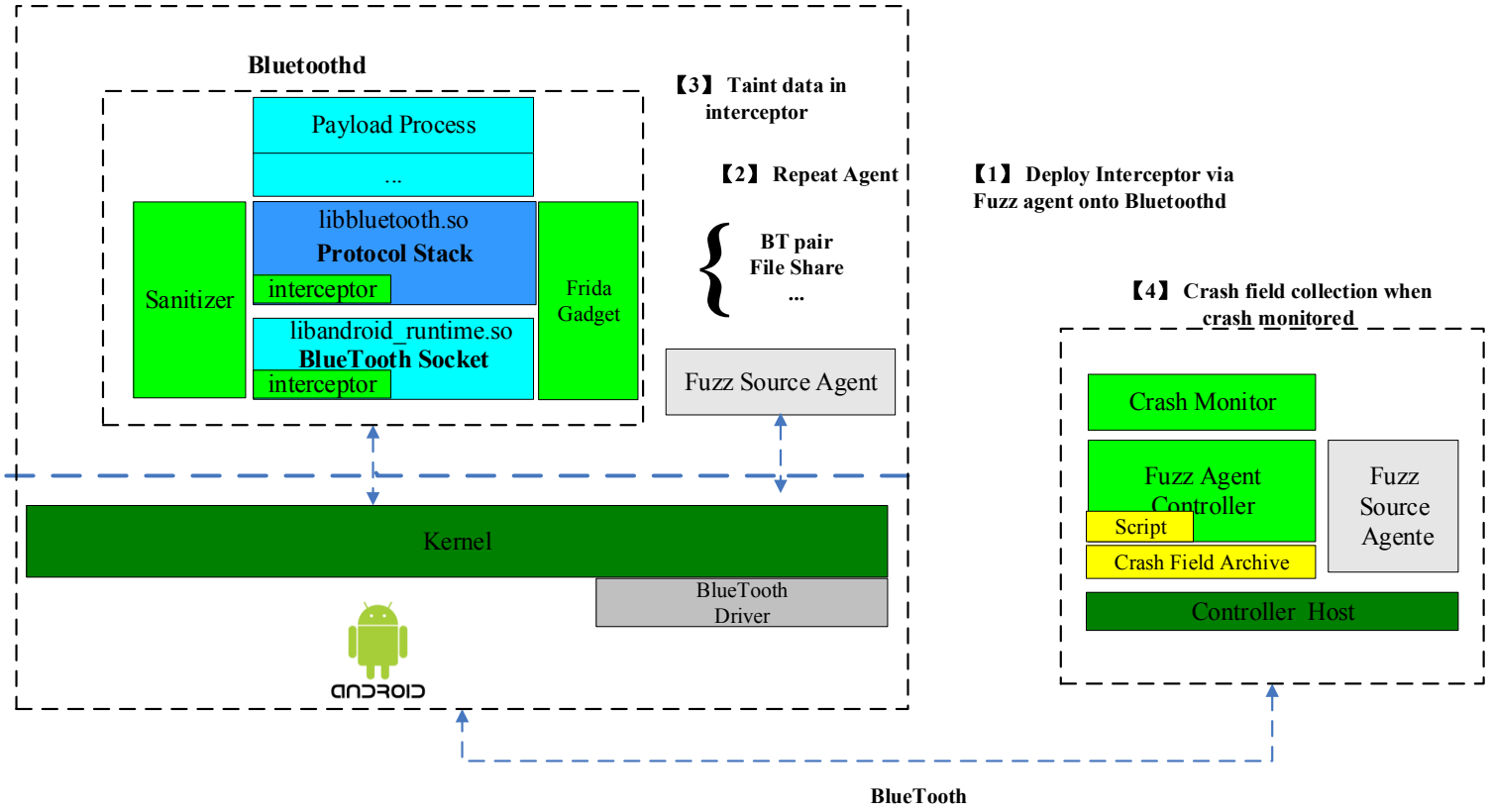
Sanitizer Support

- ASAN
  - SANITIZER_TARGET=address mm

```
aosp/android-9.0.0_r34/out/target/product/marlin/system/lib64
$grep  asan libbluetooth.so
Binary file libbluetooth.so matches
```

```
aosp/android-9.0.0_r34/system/bt
$SANITIZE_TARGET=address mm -j12
============================================
PLATFORM_VERSION_CODENAME=REL
PLATFORM_VERSION=9
TARGET_PRODUCT=aosp_marlin
TARGET_BUILD_VARIANT=eng
TARGET_BUILD_TYPE=release
TARGET_ARCH=arm64
TARGET_ARCH_VARIANT=armv8-a
TARGET_CPU_VARIANT=kryo
TARGET_2ND_ARCH=arm
                                                    mv8-a
                                                    o
                                                    -46-generic-x86_64-Ubuntu-18.04.2-LTS
HOST_CROSS_OS=windows
HOST_CROSS_ARCH=x86
HOST_CROSS_2ND_ARCH=x86_64
HOST_BUILD_TYPE=release
BUILD_ID=PQ2A.190305.002
OUT_DIR=out
============================================
ninja: no work to do.
[0/1] out/soong/.bootstrap/bin/soong_build out/soong/build.ninja
```

Fuzzer Overview



- Fuzzing Strategy
  - Fuzzing Agent Choosing Strategy
    - Easy to control
    - Easy to extend

- # Fuzzing Strategy
  - Fuzzing Agent Choosing Strategy
    - Easy to control
    - Easy to extend
  - Monitor
    - Collecting Enough information for locating crash point
    - Monitoring Target Status
    - Remove Duplicated Crash

- Fuzzing Strategy

```
app_process64: ==30924==ERROR: AddressSanitizer:
app_process64:
app_process64:
```

```
F DEBUG   :      #34 pc 0000000000084838  /system/lib64/libc.so (__pthread_start
F DEBUG   :      #35 pc 0000000000023574  /system/lib64/libc.so (__start_thread+
D HeadsetPhoneState: sendDeviceStateChanged. mService=1 mIsSimStateLoaded=true
E /system/bin/tombstoned: Tombstone written to: /data/tombstones/tombstone_39
I BootReceiver: Copying /data/tombstones/tombstone_39 to DropBox (SYSTEM_TOMBST
```

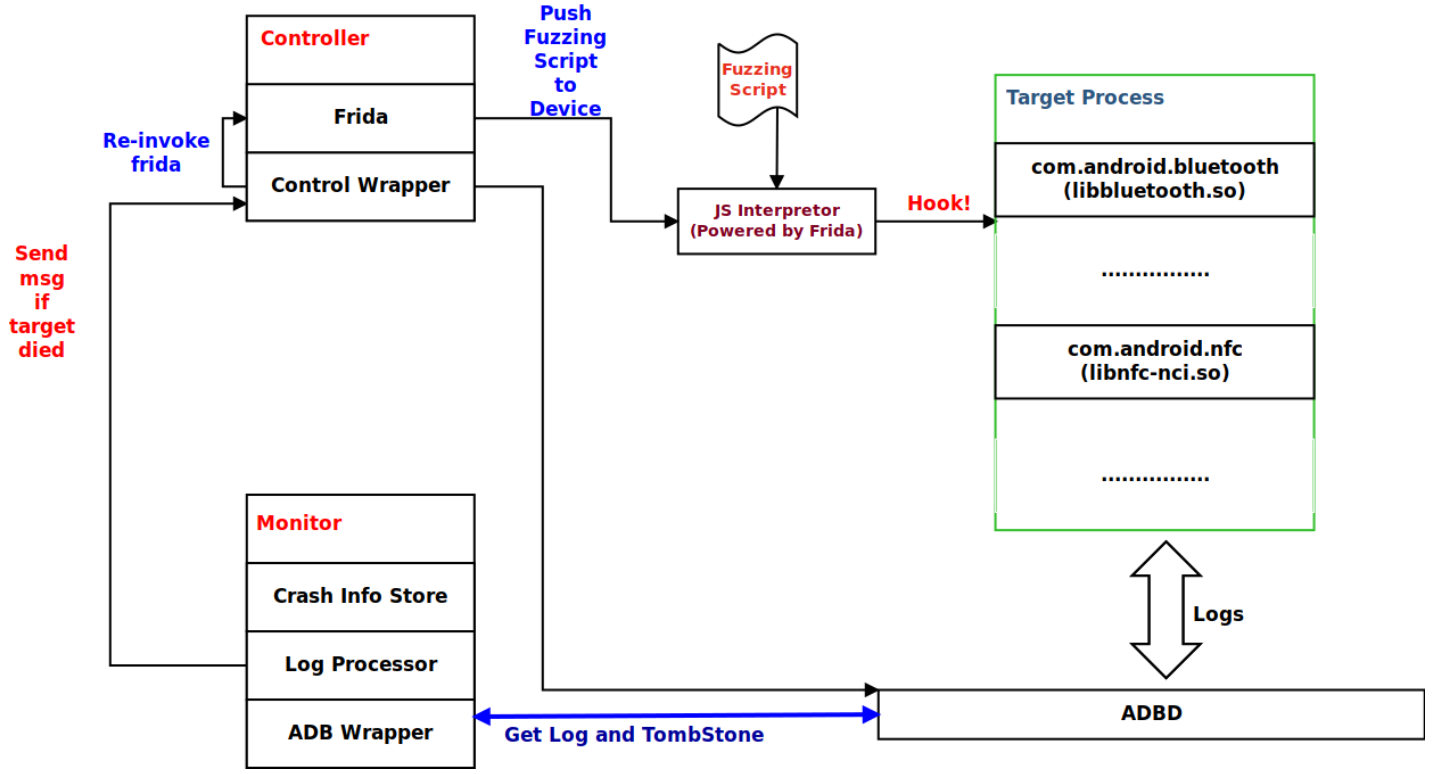
```
#00 pc 0000000000021c9c  /system/lib64/libc.so (abort+112)
#01 pc 0000000000033690  /system/lib64/libclang_rt.asan-aarch64-android.so (
#02 pc 0000000000031250  /system/lib64/libclang_rt.asan-aarch64-android.so (
#03 pc 00000000000a1fd0  /system/lib64/libclang_rt.asan-aarch64-android.so (
#04 pc 00000000000a1768  /system/lib64/libclang_rt.asan-aarch64-android.so (
ed int, bool)+348)
#05 pc 000000000009c3c4  /system/lib64/libclang_rt.asan-aarch64-android.so (
#06 pc 00000000002a0504  /system/lib64/libbluetooth.so (btif_dm_data_copy(uns
```

oint

Remove Duplicated Crash

- # Fuzzing Strategy
  - Fuzzing Agent Choosing Strategy
    - Easy to control
    - Easy to extend
  - Monitor
    - Collecting Enough information for locating crash point
    - Monitoring Target Status
  - Reproducer
    - Easy to reproduce the crash

Fuzzing Strategy



- Best Practice
  - Export target functions

```
rfcomm/rfc_l2cap_if.cc:282:__attribute__((visibility("default"))) void RFCOMM_BufDataInd(uint16_t lcid, BT_HDR* p_buf)
rfcomm/port_rfc.cc:760:__attribute__((visibility("default"))) void PORT_DataInd(tRFC_MCB* p_mcb, uint8_t dlci, BT_HDR*
mcap/mca_main.cc:408:__attribute__((visibility("default"))) void mca_tc_data_ind(tMCA_TC_TBL* p_tbl, BT_HDR* p_buf) {
mcap/mca_l2c.cc:499:__attribute__((visibility("default"))) void mca_l2c_data_ind_cback(uint16_t lcid, BT_HDR* p_buf) {
avct/avct_l2c.cc:407:__attribute__((visibility("default"))) void avct_l2c_data_ind_cback(uint16_t lcid, BT_HDR* p_buf)
gap/gap_conn.cc:991:__attribute__((visibility("default"))) void gap_data_ind(uint16_t l2cap_cid, BT_HDR* p_msg) {
hid/hidd_conn.cc:610:__attribute__((visibility("default")))  void hidd_l2cif_data_ind(uint16_t cid, BT_HDR* p_msg) {
hid/hidh_conn.cc:784:__attribute__((visibility("default"))) void hidh_l2cif_data_ind(uint16_t l2cap_cid, BT_HDR* p_msg)
sdp/sdp_main.cc:483:__attribute__((visibility("default"))) void sdp_data_ind(uint16_t l2cap_cid, BT_HDR* p_msg) {
pan/pan_main.cc:382:__attribute__((visibility("default"))) void pan_data_ind_cb(uint16_t handle, const RawAddress& src,
avdt/avdt_ad.cc:445:__attribute__((visibility("default"))) void avdt_ad_tc_data_ind(AvdtpTransportChannel* p_tbl, BT_HD
avdt/avdt_l2c.cc:508:__attribute__((visibility("default"))) void avdt_l2c_data_ind_cback(uint16_t lcid, BT_HDR* p_buf)
bnep/bnep_main.cc:430:__attribute__((visibility("default"))) void bnep_data_ind(uint16_t l2cap_cid, BT_HDR* p_buf) {
smp/smp_l2c.cc:139:__attribute__((visibility("default"))) void smp_data_received(uint16_t channel, const RawAddress& bd
gatt/gatt_main.cc:954:__attribute__((visibility("default"))) void gatt_data_process(tGATT_TCB& tcb, BT_HDR* p_buf) {
l2cap/l2c_main.cc:63:__attribute__((visibility("default"))) void l2c_rcv_acl_data(BT_HDR* p_msg) {
```

- Best Practice
  - Frida loop up symbols
    - Use enum not directly findByExportName

```
var module = Process.findModuleByName("libbluetooth.so");
var e_list = Module.enumerateExportsSync(module.name);

e_list.forEach(function (exp) {
    // console.log(exp.name);
    if (exp.name == "_Z16l2c_rcv_acl_dataP6BT_HDR") { // Found
    // console.log('l2c');
    l2c_rcv_acl_data = exp.address;
```

- Best Practice
  - Right Data Address
    - Calculate as the source code do

```
uint8_t* p = (uint8_t*)(p_msg + 1) + p_msg->offset;
```

```
typedef struct {
  uint16_t event;
  uint16_t len;
  uint16_t offset;
  uint16_t layer_spe
  uint8_t data[];
} BT_HDR;
```

```
console.log('l2c_rcv_acl_data');
var bt_hdr = ptr(args[0]);
var len = Memory.readU16(bt_hdr.add(0x2));
var offset = Memory.readU16(bt_hdr.add(0x4));
var data = bt_hdr.add(0x8).add(offset);
```

Kernel Space Fuzzing

- Introduction
- User Space Fuzzing
- Kernel Space Fuzzing
  - Passive fuzz architecture
  - Automatic passive fuzz
  - Case study
- Demo

Introduction

2019-01-05 security patch level vulnerability details

In the sections below, we provide details for each of the security vulnerabilities that apply to the 2019-01-05 patch level. Vulnerabilities are grouped under the component they affect and include details such as the CVE, associated references, type of vulnerability, severity, component (where applicable), and updated AOSP versions (where applicable). When available, we link the public change that addressed the issue to the bug ID, such as the AOSP change list. When multiple changes relate to a single bug, additional references are linked to numbers following the bug ID.

Kernel components

The most severe vulnerability in this section could enable a local malicious application to execute arbitrary code within the context of a privileged process.

CVE	References	Type	Severity	Component
CVE-2018-10876	A-116406122 Upstream kernel	EoP	High	ext4 filesystem
CVE-2018-10880	A-116406509 Upstream kernel	EoP	High	ext4 filesystem
CVE-2018-10882	A-116406626 Upstream kernel	EoP	High	ext4 filesystem
CVE-2018-13405	A-113452403 Upstream kernel	EoP	High	Filesystem
CVE-2018-18281	A-118836219 Upstream kernel	EoP	High	TLB
CVE-2018-17182	A-117280327 Upstream kernel	EoP	High	Memory Manager
CVE-2018-10877	A-116406625 Upstream kernel	ID	High	ext4 filesystem

2019-02-05 security patch level vulnerability details

In the sections below, we provide details for each of the security vulnerabilities that apply to the 2019-02-05 patch level. Vulnerabilities are grouped under the component they affect and include details such as the CVE, associated references, type of vulnerability, severity, component (where applicable), and updated AOSP versions (where applicable). When available, we link the public change that addressed the issue to the bug ID, such as the AOSP change list. When multiple changes relate to a single bug, additional references are linked to numbers following the bug ID.

Kernel components

The most severe vulnerability in this section could enable a local malicious application to execute arbitrary code within the context of a privileged process.

CVE	References	Type	Severity	Component
CVE-2018-10879	A-116406063 Upstream kernel	EoP	High	ext4 filesystem
CVE-2019-1999	A-120025196*	EoP	High	Binder driver
CVE-2019-2000	A-120025789*	EoP	High	Binder driver
CVE-2019-2001	A-117422211*	ID	High	iomem

2019-03-05 security patch level vulnerability details

In the sections below, we provide details for each of the security vulnerabilities that apply to the 2019-03-05 patch level. Vulnerabilities are grouped under the component they affect and include details such as the CVE, associated references, type of vulnerability, severity, component (where applicable), and updated AOSP versions (where applicable). When available, we link the public change that addressed the issue to the bug ID, such as the AOSP change list. When multiple changes relate to a single bug, additional references are linked to numbers following the bug ID.

System

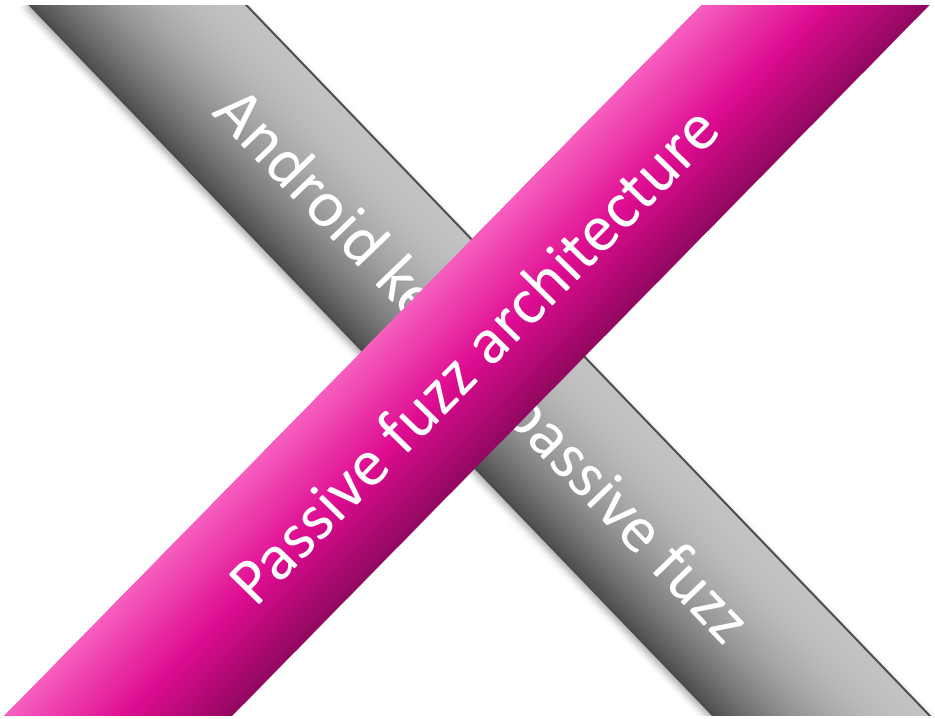
The vulnerability in this section could enable a local malicious application to execute arbitrary code within the context of a privileged process.

CVE	References	Type	Severity	Updated AOSP versions
CVE-2019-2023	A-121035042 [2] [3] [4] [5] [6] [7]	EoP	High	8.0, 8.1, 9

Kernel components

The most severe vulnerability in this section could enable a local attacker using a specially crafted file to execute arbitrary code within the context of a privileged process.

CVE	References	Type	Severity	Component
CVE-2018-10883	A-117311198 Upstream kernel [2]	EoP	High	ext4 filesystem
CVE-2019-2024	A-111761954 Upstream kernel	EoP	High	em28xx driver
CVE-2019-2025	A-116855682 Upstream kernel	EoP	High	Binder driver

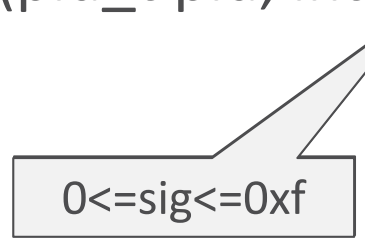


How to achieve passive fuzz

- Kernel interceptor
  - ✓Part of the kernel
  - ✓Has good stability
- Fuzzing trigger
  - ✓linux system call “kill”

How to transfer the filter list

```
#include <sys/types.h>
#include <signal.h>

int kill(pid_t pid, int sig);
```



Receive filter list

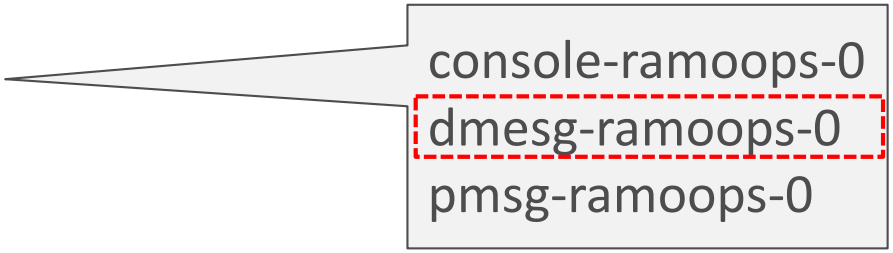
```
if(pid==1 && sig<17){
        temp=((temp<<4)+sig)&0xffffffff;
        cycle+=1;
        //printk(KERN_NOTICE "Go Into Set Start Flag, temp is %x cycle is %d\n",temp,cycle);
        if(cycle==8){
                cycle=0;

                for(int i=0;i<20;i++){
                        if(filter_list[i]==0)
                        {
                                filter_list[i]=temp;
                                printk(KERN_NOTICE "Go Into Set filter_list[%d] is 0x%x !\n",i,temp);
                                temp=0;
                                break;
                        }
                        else if(filter_list[i]==temp)
                        {
                                break;
                        }
                }
        }
}
```

Fuzzing strategies

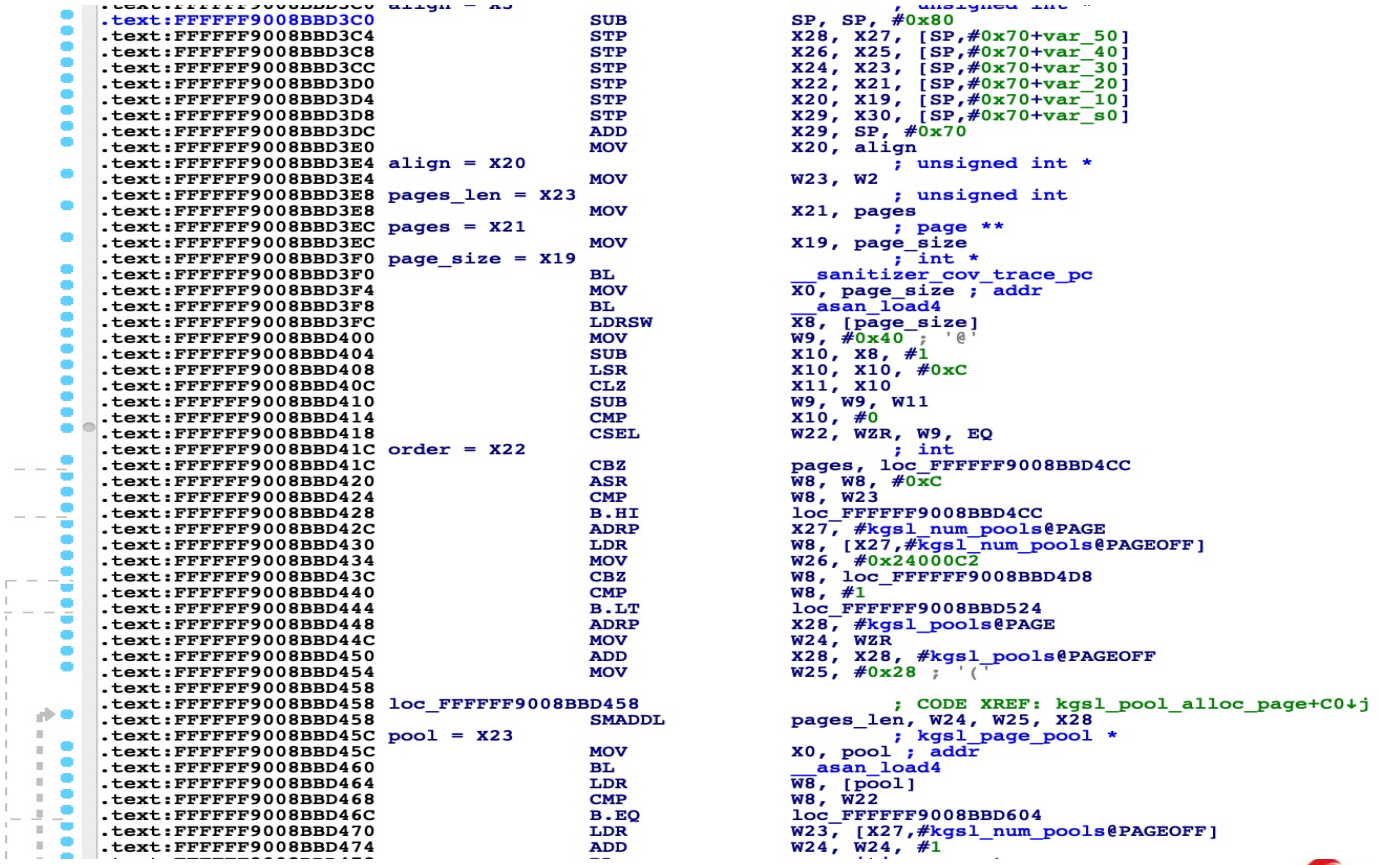
- (1) Without changing the length of the parameter;
- (2) Modifying the fixed number bytes;
- (3) Only modify one bit or one byte;
- (4) Modifying the random number of bytes;
- (5) use the for loop to perform multiple fuzz;
- (6) Set the fuzz frequency for each target function;

What do we need to prepare

- How to open kasan in the kernel
- Kernel panic log
  - ✓/sys/fs/pstore



How to open kasan in the kernel

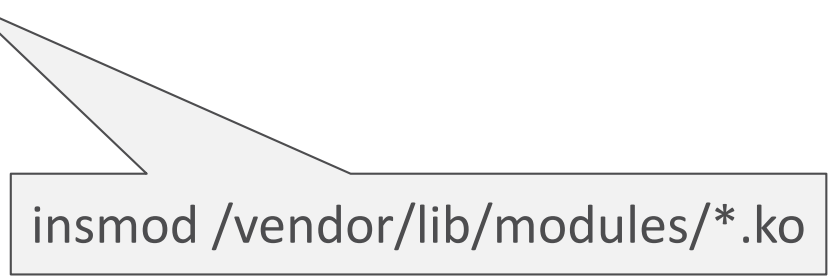


```
.text:FFFFFF9008BBD3C0                  SUB             SP, SP, #0x80
.text:FFFFFF9008BBD3C4                  STP             X28, X27, [SP,#0x70+var_50]
.text:FFFFFF9008BBD3C8                  STP             X26, X25, [SP,#0x70+var_40]
.text:FFFFFF9008BBD3CC                  STP             X24, X23, [SP,#0x70+var_30]
.text:FFFFFF9008BBD3D0                  STP             X22, X21, [SP,#0x70+var_20]
.text:FFFFFF9008BBD3D4                  STP             X20, X19, [SP,#0x70+var_10]
.text:FFFFFF9008BBD3D8                  STP             X29, X30, [SP,#0x70+var_s0]
.text:FFFFFF9008BBD3DC                  ADD             X29, SP, #0x70
.text:FFFFFF9008BBD3E0                  MOV             X20, align
.text:FFFFFF9008BBD3E4 align = X20                              ; unsigned int *
.text:FFFFFF9008BBD3E4                  MOV             W23, W2
.text:FFFFFF9008BBD3E8 pages_len = X23                          ; unsigned int
.text:FFFFFF9008BBD3E8                  MOV             X21, pages
.text:FFFFFF9008BBD3EC pages = X21                              ; page **
.text:FFFFFF9008BBD3EC                  MOV             X19, page_size
.text:FFFFFF9008BBD3F0 page_size = X19                          ; int *
.text:FFFFFF9008BBD3F0                  BL              __sanitizer_cov_trace_pc
.text:FFFFFF9008BBD3F4                  MOV             X0, page_size ; addr
.text:FFFFFF9008BBD3F8                  BL              __asan_load4
.text:FFFFFF9008BBD3FC                  LDRSW           X8, [page_size]
.text:FFFFFF9008BBD400                  MOV             W9, #0x40 ; '@'
.text:FFFFFF9008BBD404                  SUB             X10, X8, #1
.text:FFFFFF9008BBD408                  LSR             X10, X10, #0xC
.text:FFFFFF9008BBD40C                  CLZ             X11, X10
.text:FFFFFF9008BBD410                  SUB             W9, W9, W11
.text:FFFFFF9008BBD414                  CMP             X10, #0
.text:FFFFFF9008BBD418                  CSEL            W22, WZR, W9, EQ
.text:FFFFFF9008BBD41C order = X22                              ; int
.text:FFFFFF9008BBD41C                  CBZ             pages, loc_FFFFFF9008BBD4CC
.text:FFFFFF9008BBD420                  ASR             W8, W8, #0xC
.text:FFFFFF9008BBD424                  CMP             W8, W23
.text:FFFFFF9008BBD428                  B.HI            loc_FFFFFF9008BBD4CC
.text:FFFFFF9008BBD42C                  ADRP            X27, #kgsl_num_pools@PAGE
.text:FFFFFF9008BBD430                  LDR             W8, [X27,#kgsl_num_pools@PAGEOFF]
.text:FFFFFF9008BBD434                  MOV             W26, #0x24000C2
.text:FFFFFF9008BBD43C                  CBZ             W8, loc_FFFFFF9008BBD4D8
.text:FFFFFF9008BBD440                  CMP             W8, #1
.text:FFFFFF9008BBD444                  B.LT            loc_FFFFFF9008BBD524
.text:FFFFFF9008BBD448                  ADRP            X28, #kgsl_pools@PAGE
.text:FFFFFF9008BBD44C                  MOV             W24, WZR
.text:FFFFFF9008BBD450                  ADD             X28, X28, #kgsl_pools@PAGEOFF
.text:FFFFFF9008BBD454                  MOV             W25, #0x28 ; '('
.text:FFFFFF9008BBD458
.text:FFFFFF9008BBD458 loc_FFFFFF9008BBD458                     ; CODE XREF: kgsl_pool_alloc_page+C0↓j
.text:FFFFFF9008BBD458                  SMADDL          pages_len, W24, W25, X28
.text:FFFFFF9008BBD45C pool = X23                               ; kgsl_page_pool *
.text:FFFFFF9008BBD45C                  MOV             X0, pool ; addr
.text:FFFFFF9008BBD460                  BL              __asan_load4
.text:FFFFFF9008BBD464                  LDR             W8, [pool]
.text:FFFFFF9008BBD468                  CMP             W8, W22
.text:FFFFFF9008BBD46C                  B.EQ            loc_FFFFFF9008BBD604
.text:FFFFFF9008BBD470                  LDR             W23, [X27,#kgsl_num_pools@PAGEOFF]
.text:FFFFFF9008BBD474                  ADD             W24, W24, #1
```

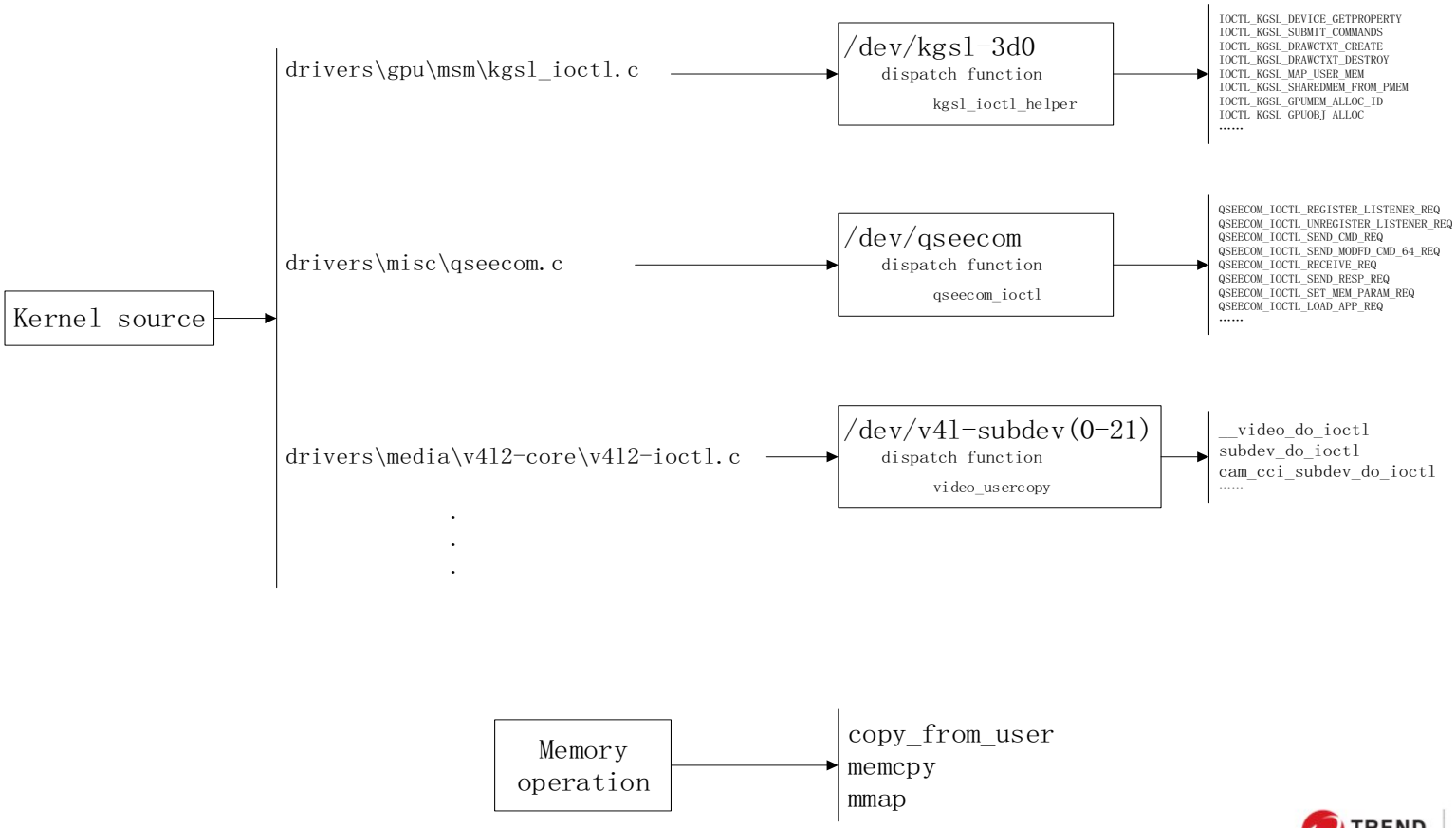

TREND MICRO | research

Build kernel issues

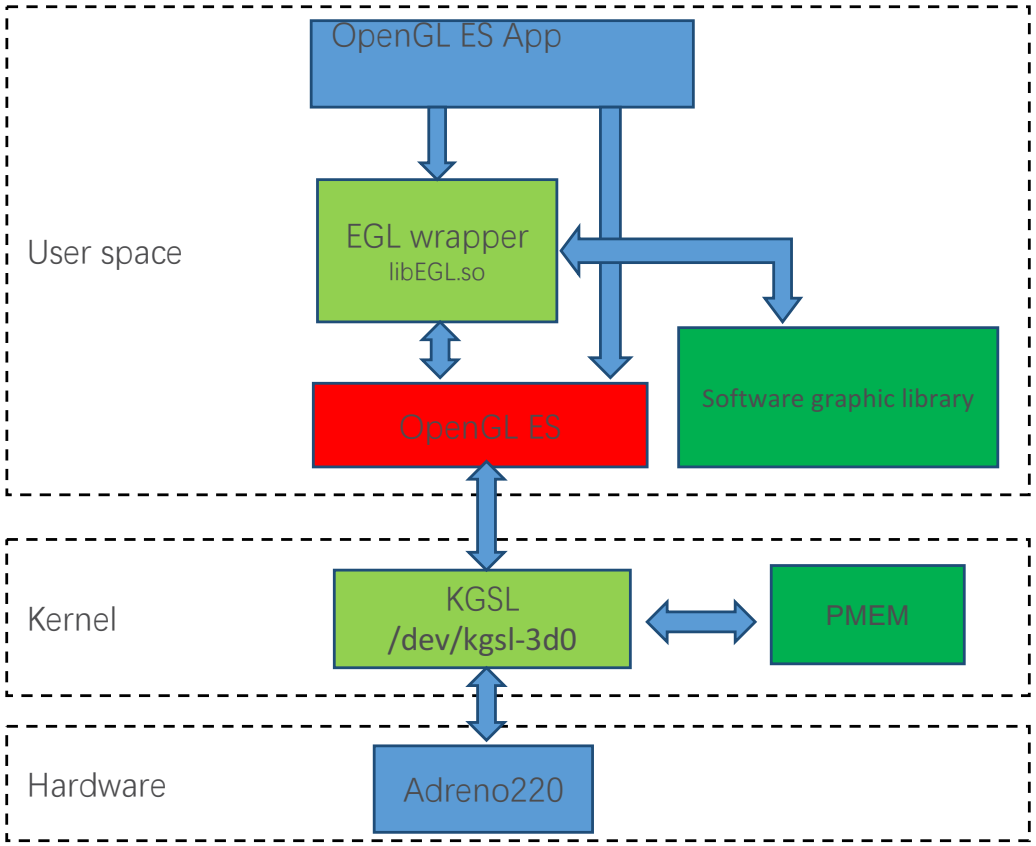
- Touch screen cannot work
  - ✓Required key not available
  - ✓exec format error
- How to deal with?
  - ✓Remove kernel's signature authentication
  - ✓Replace *.ko in the vendor.img
  - ✓Turn off dm-verity checking



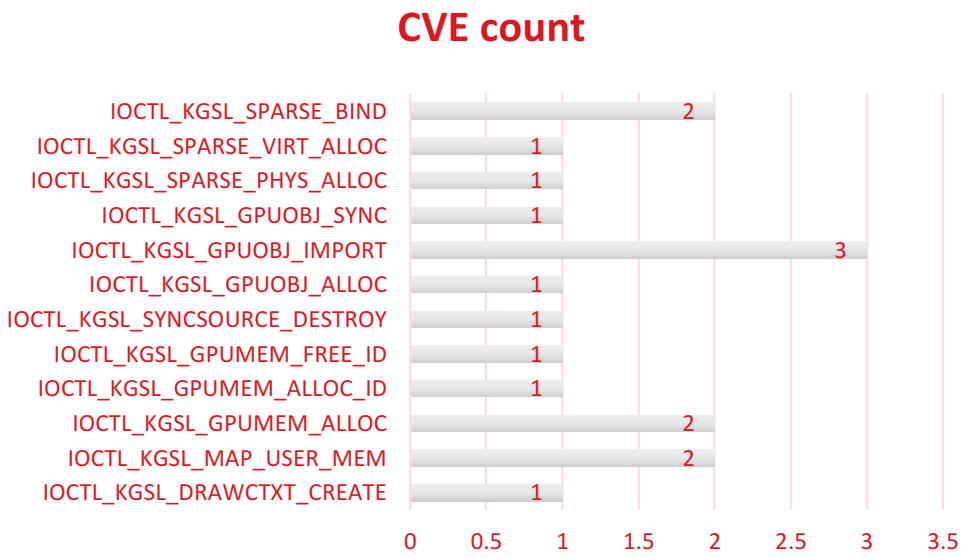
Attack interface

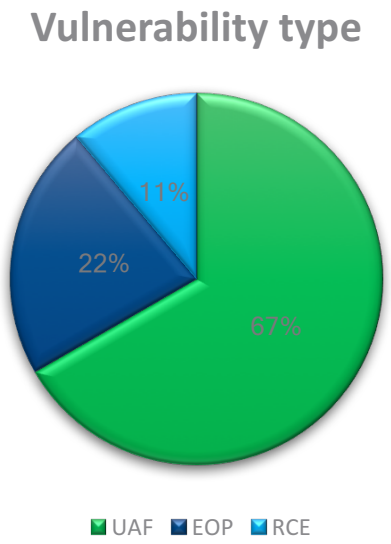


Why kgsl

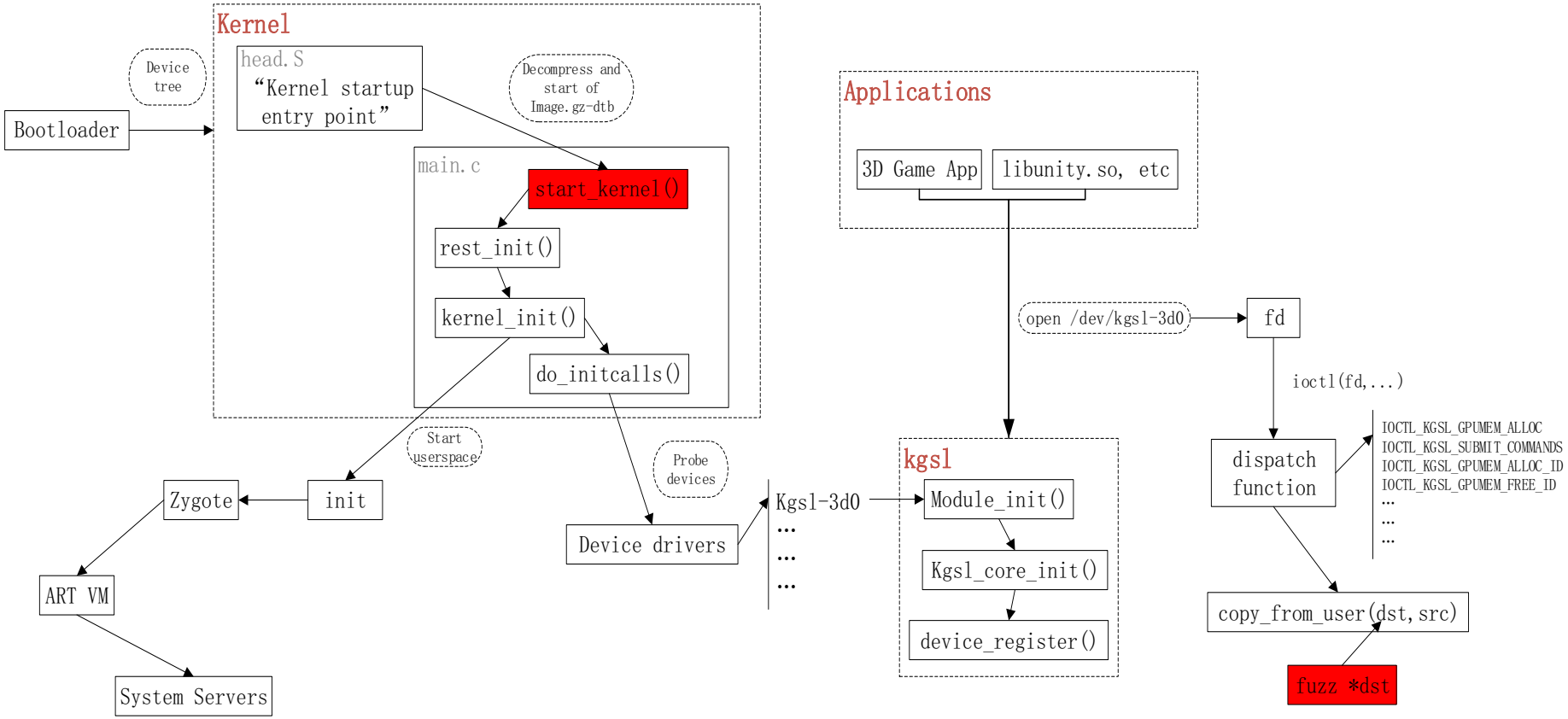


Why kgsl

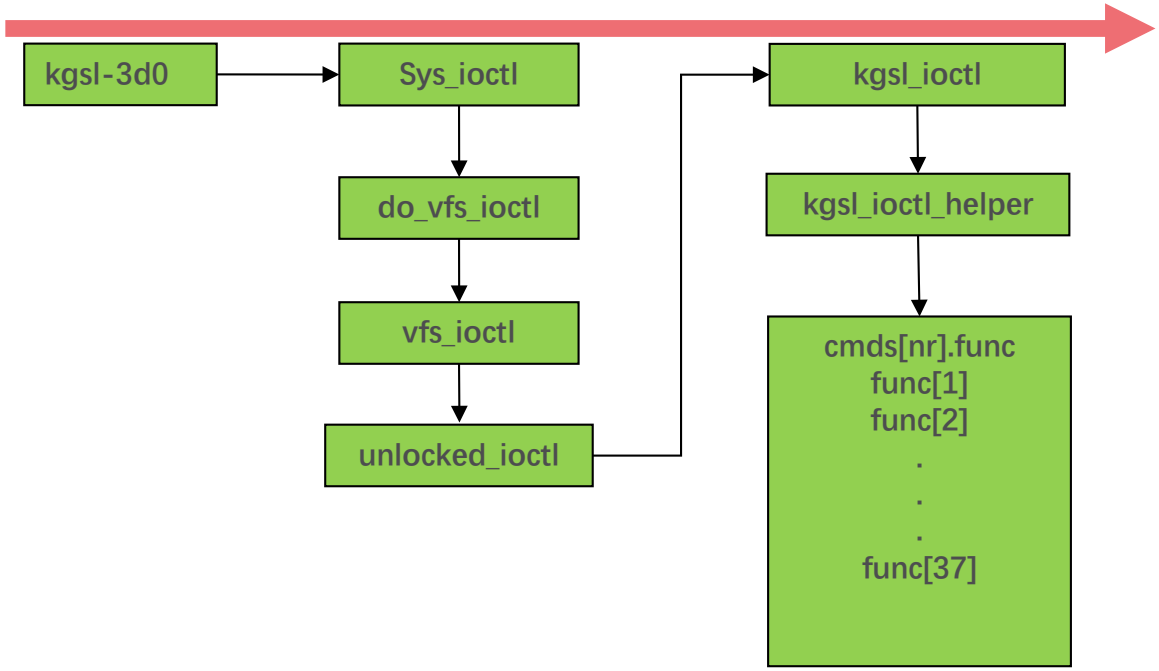




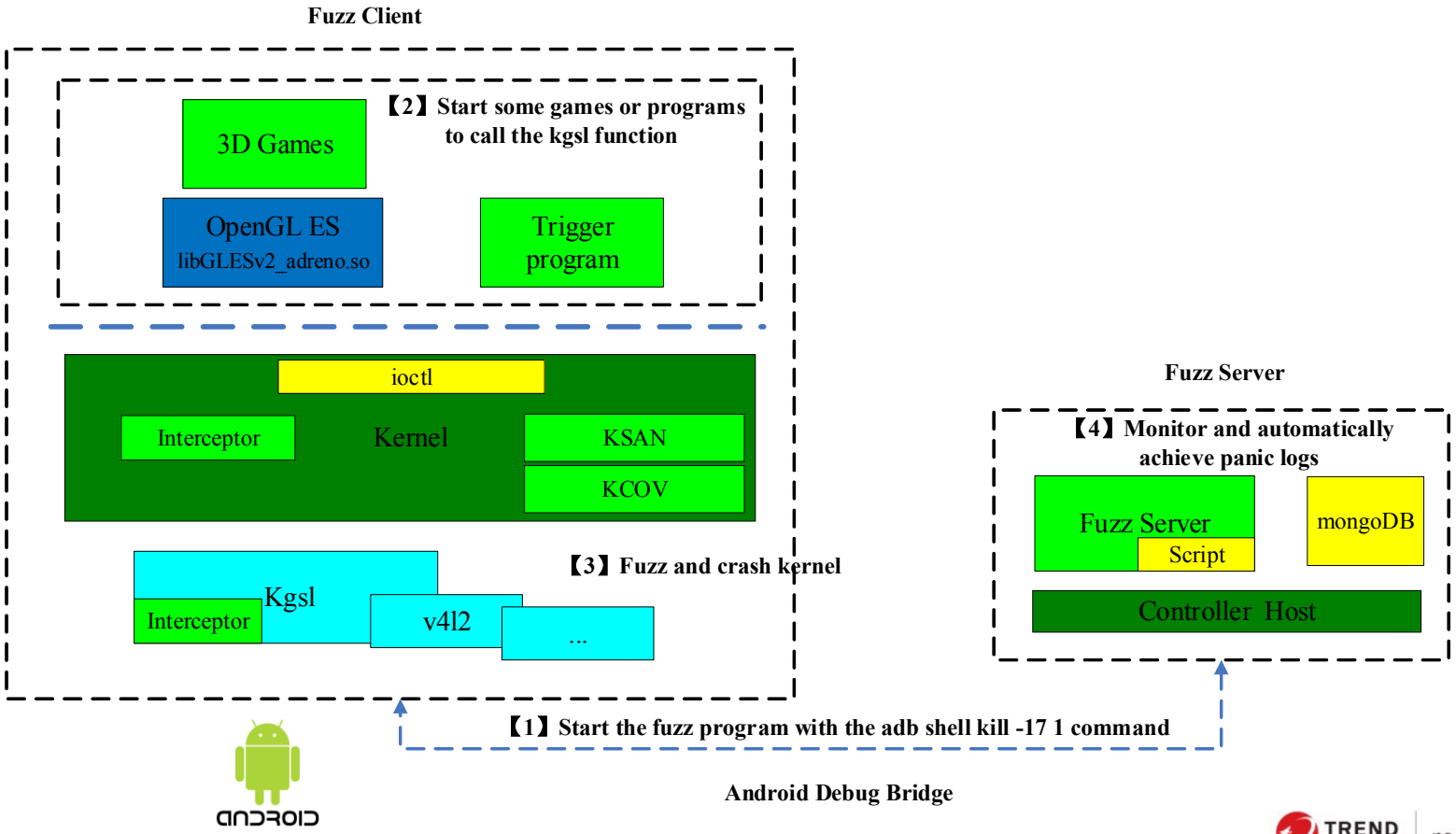
KGSL in detail



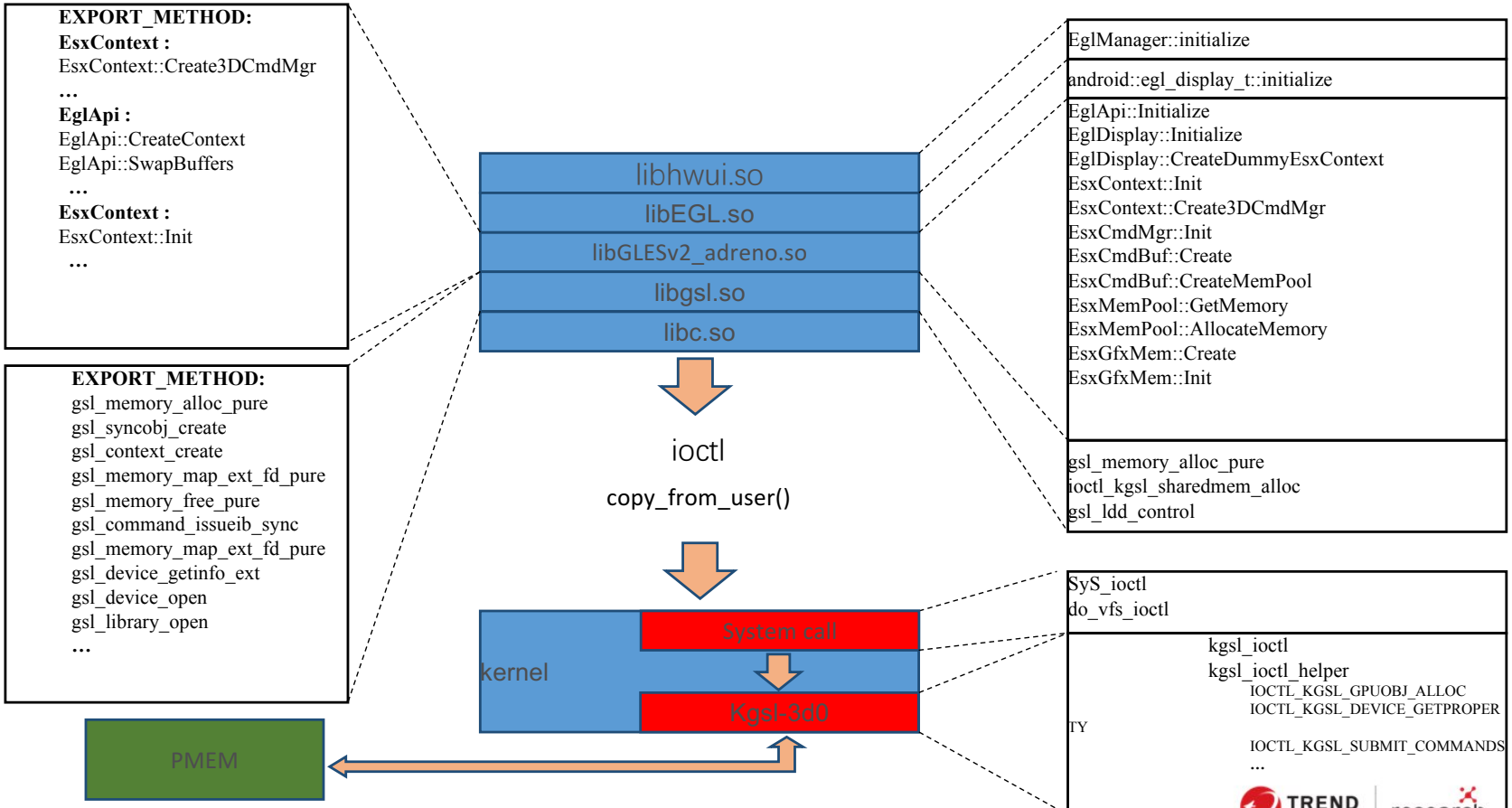
KGSL in detail



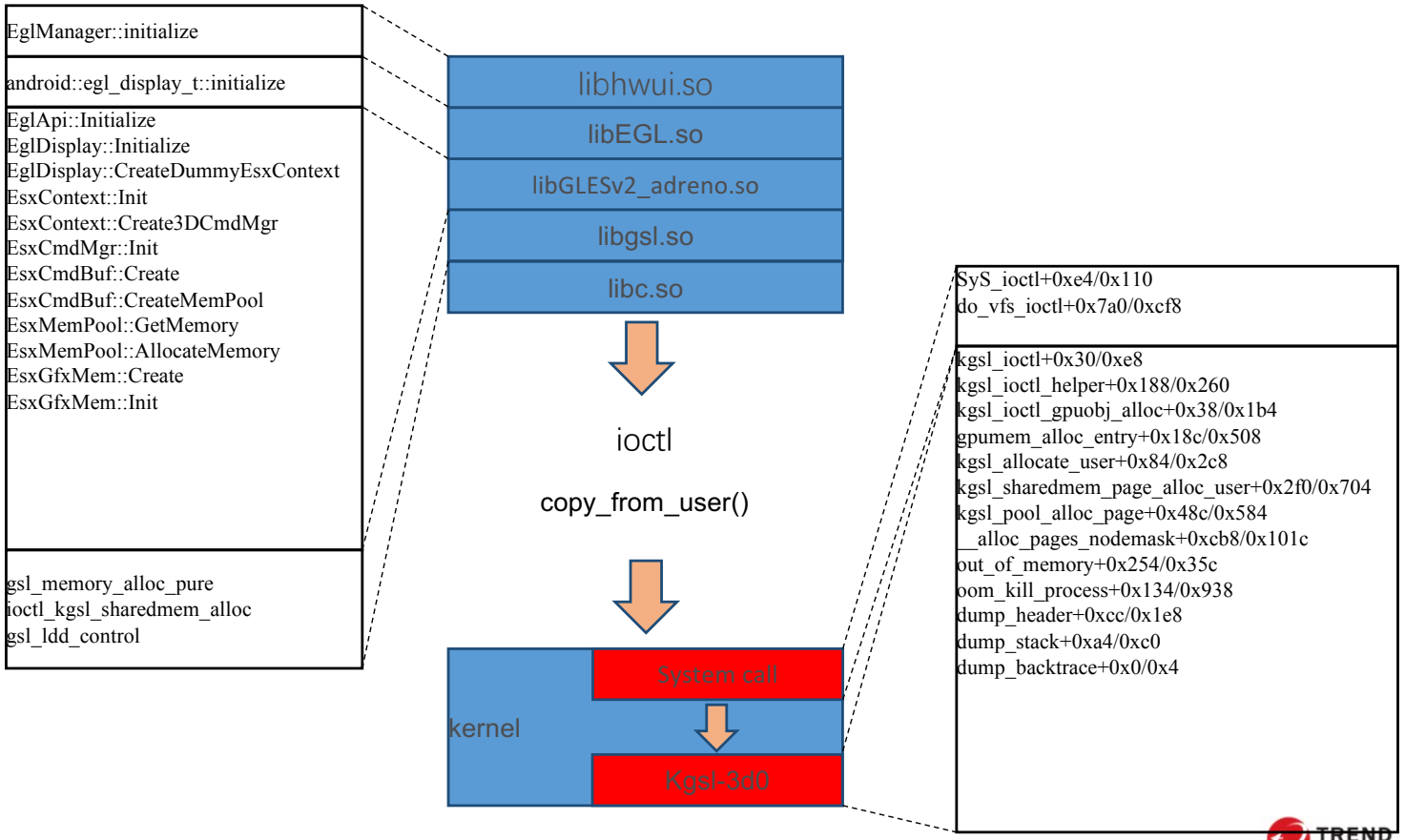
Solution Overview

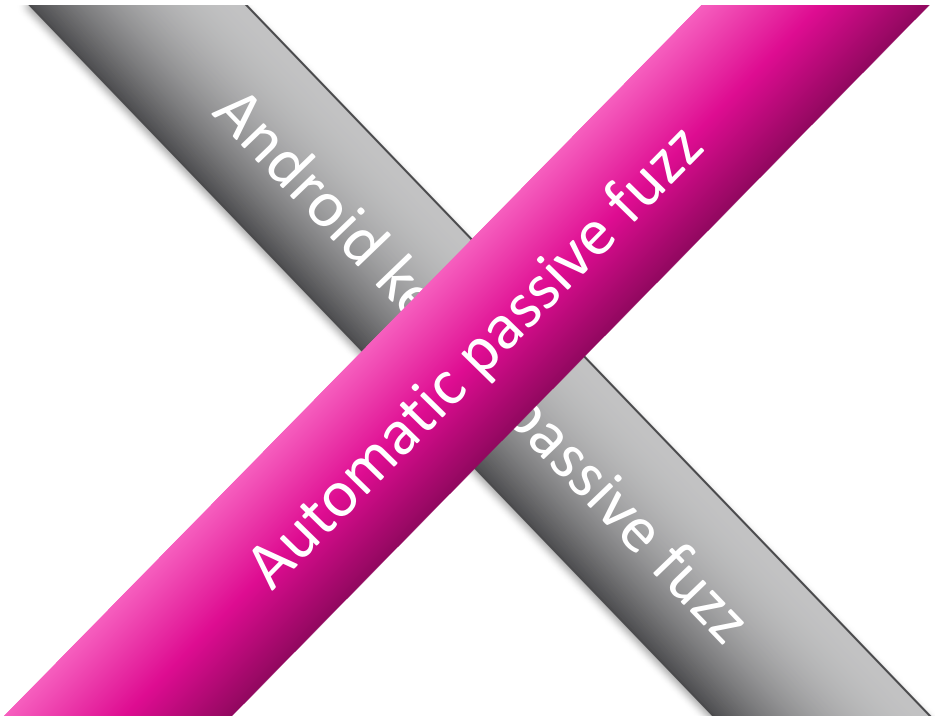


Call stack

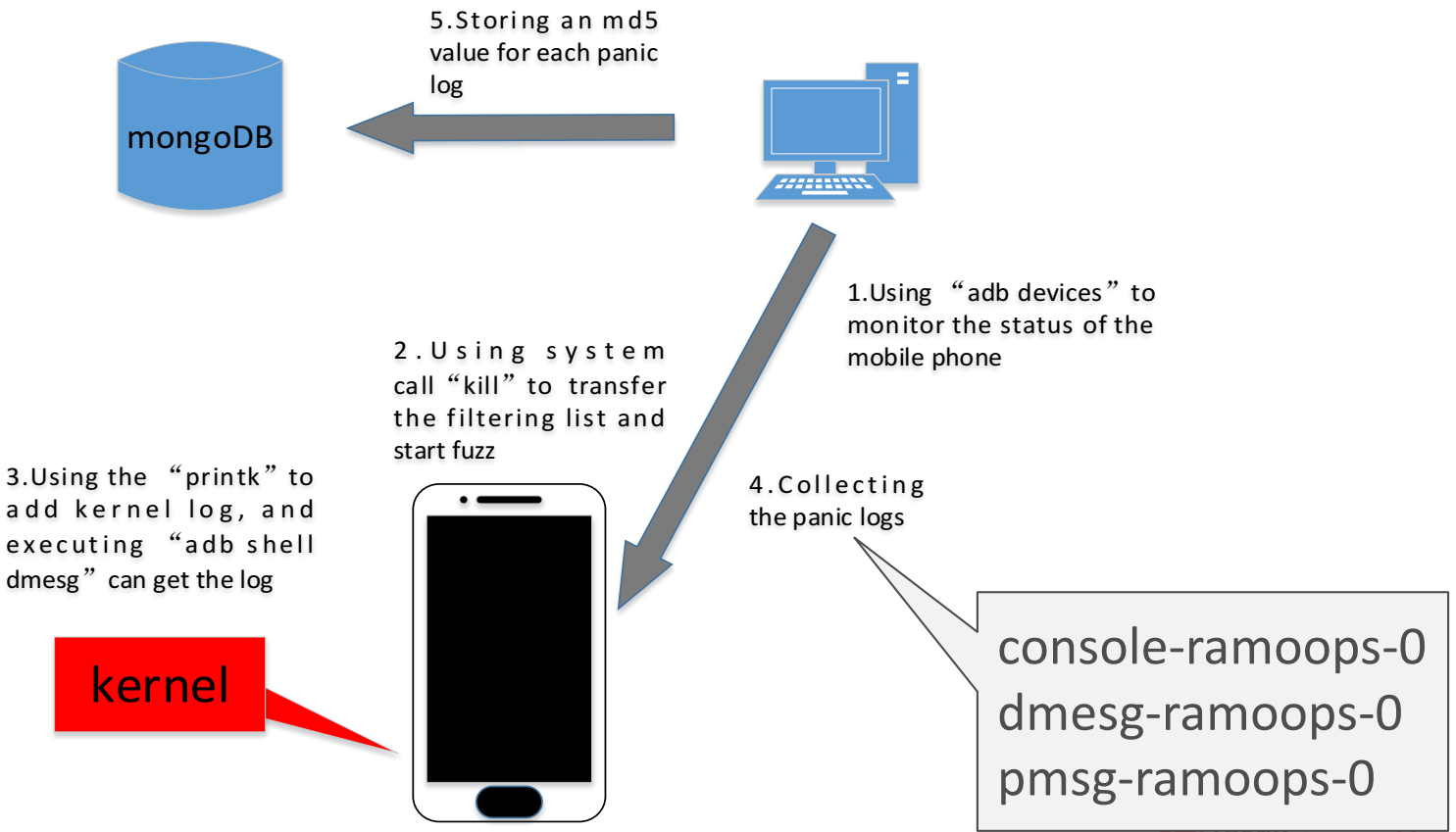


Panic call stack

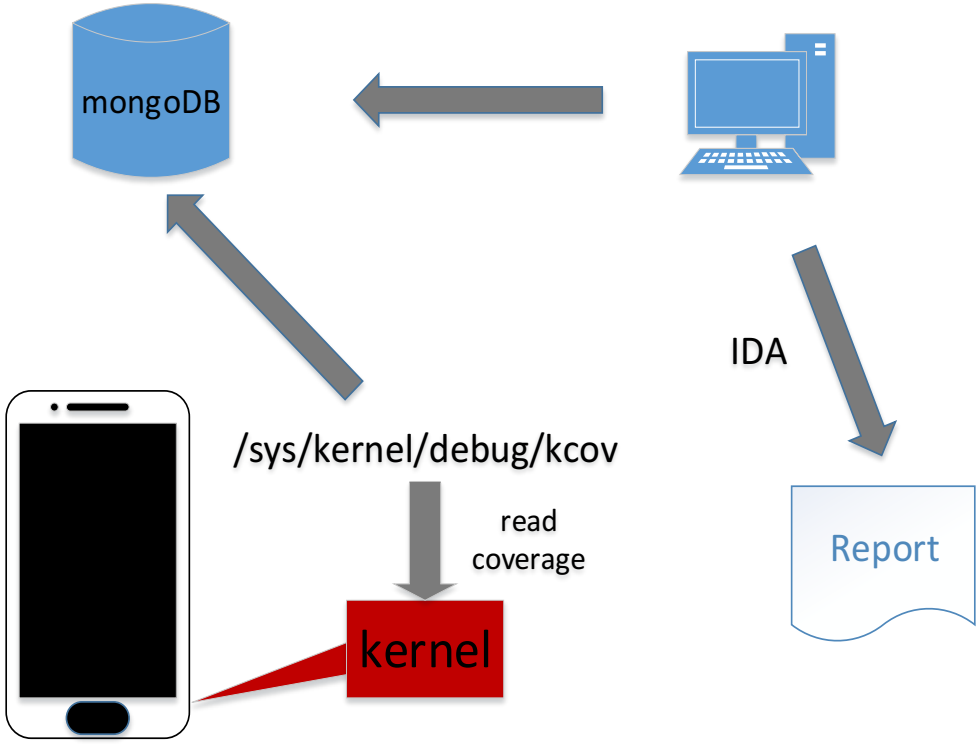




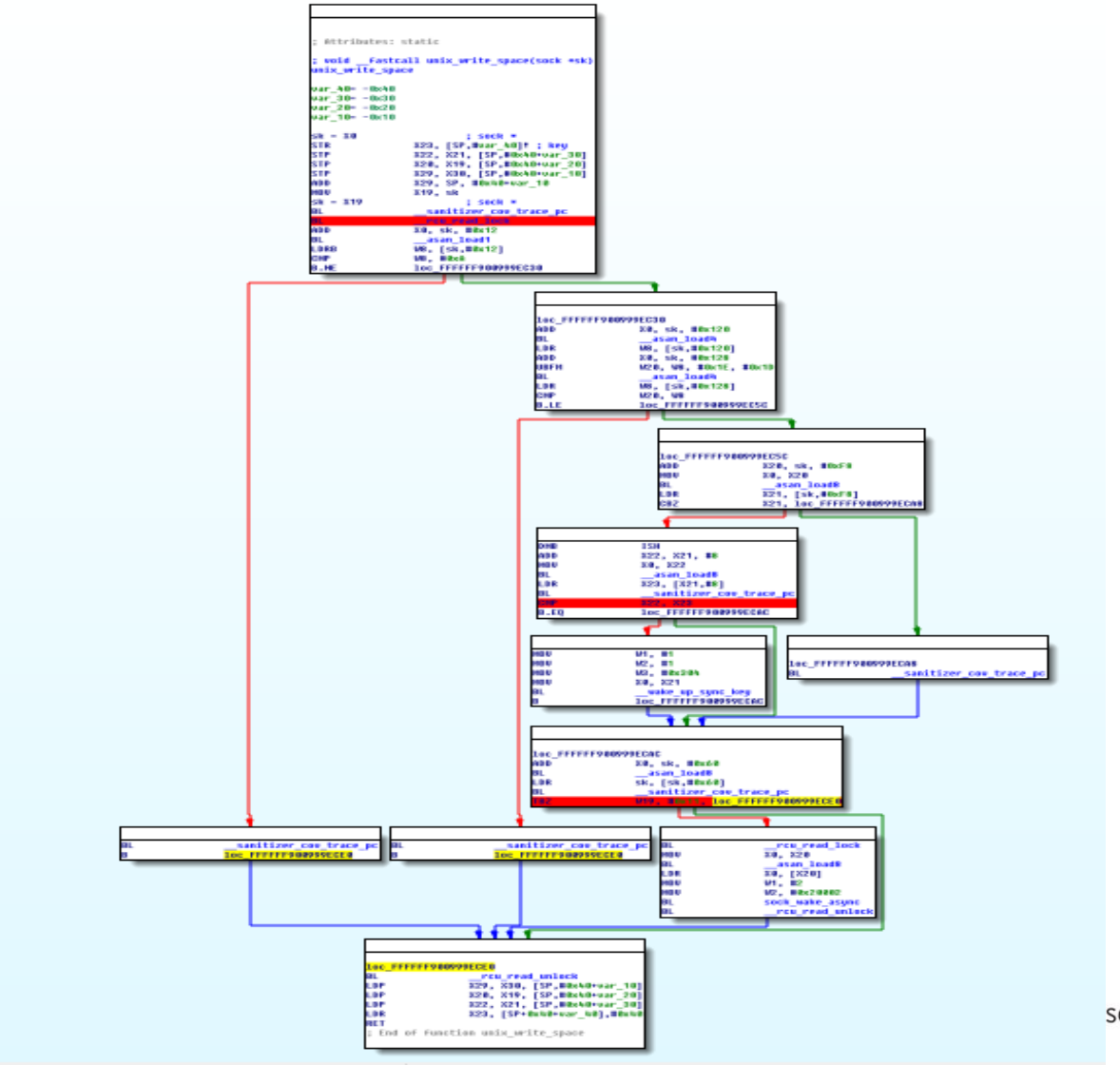
How to make it automatic



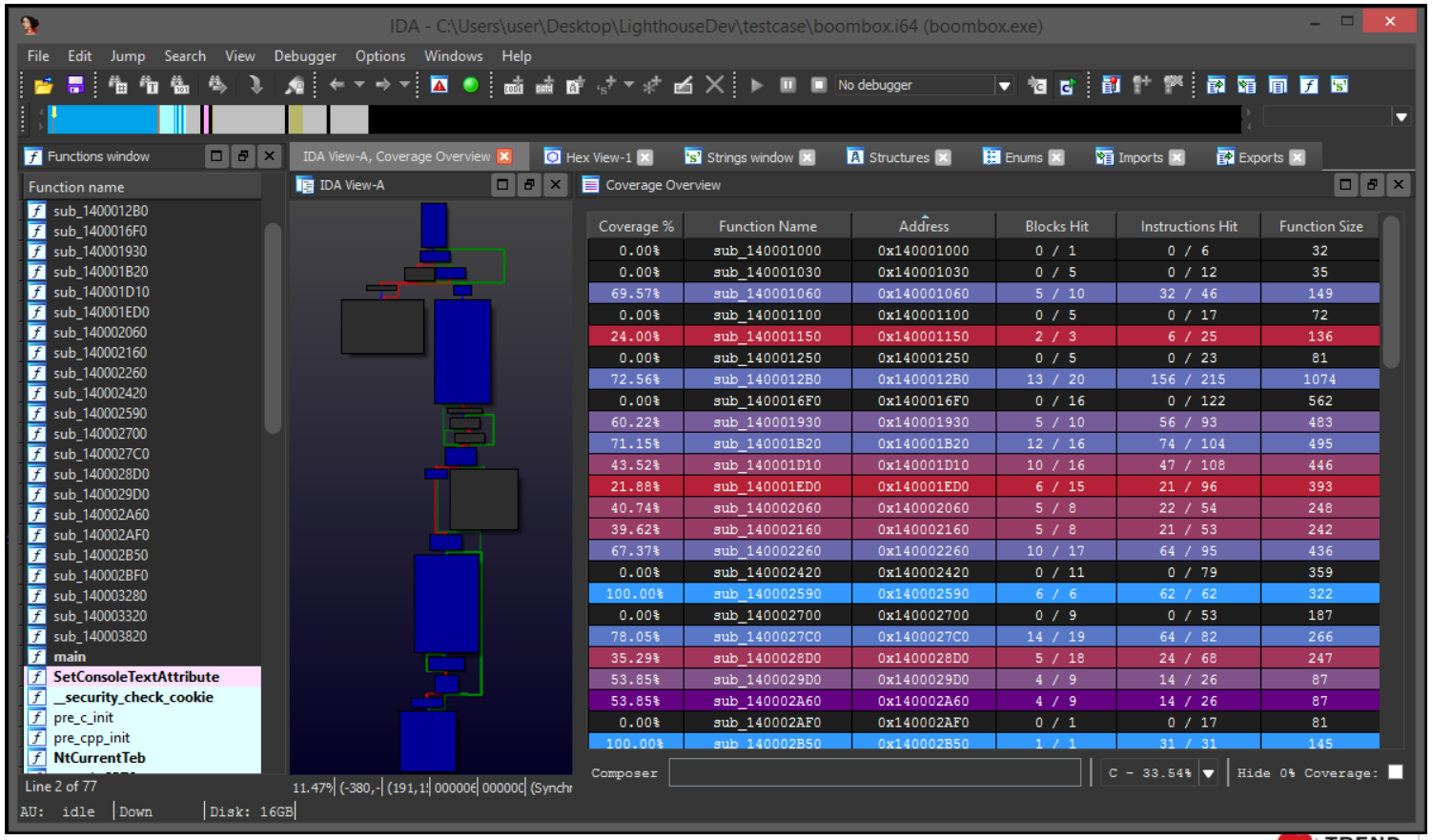
Fuzz status statistics



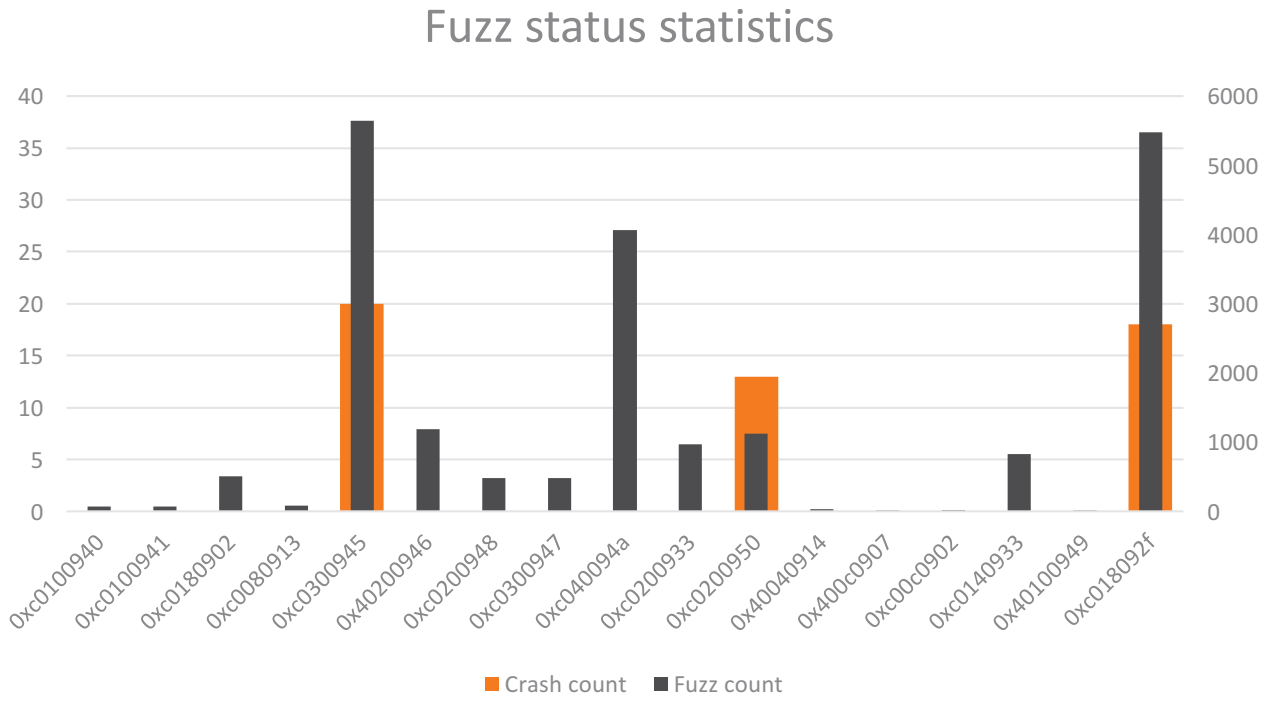
KCOV in IDA



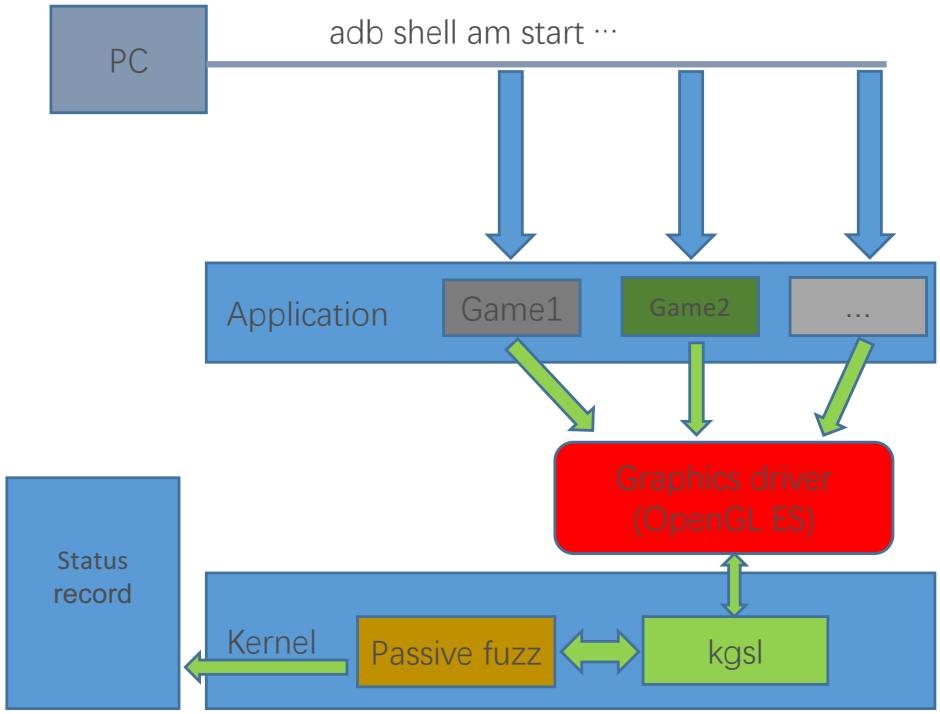
KCOV in IDA



- Fuzz status statistics



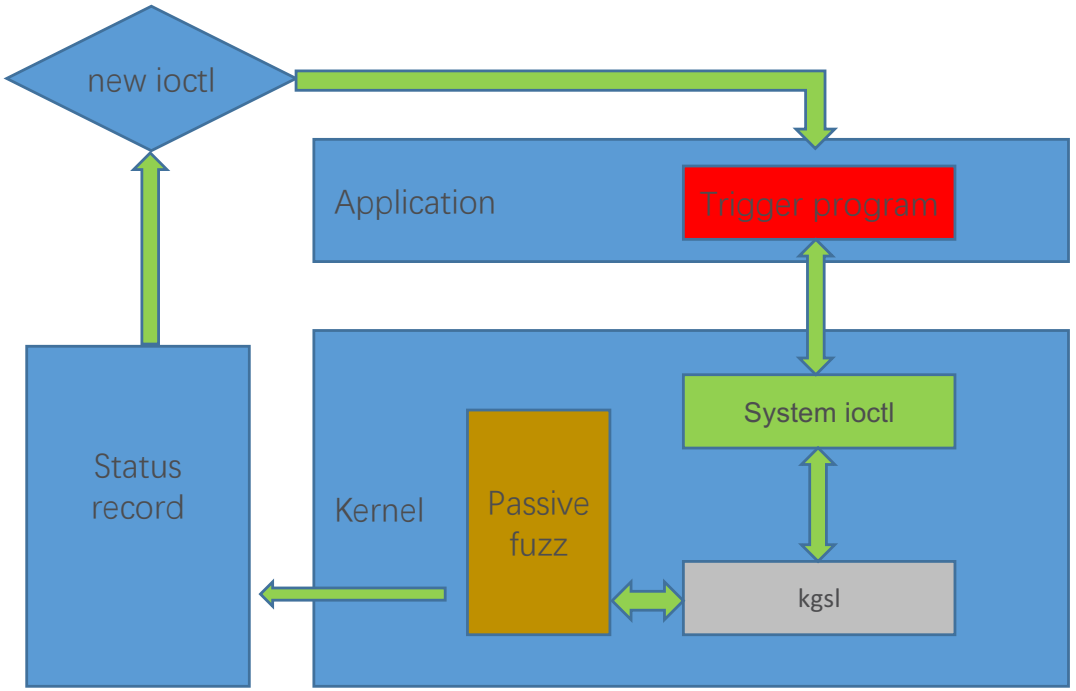
Install and run different kinds of 3D games



add a for loop

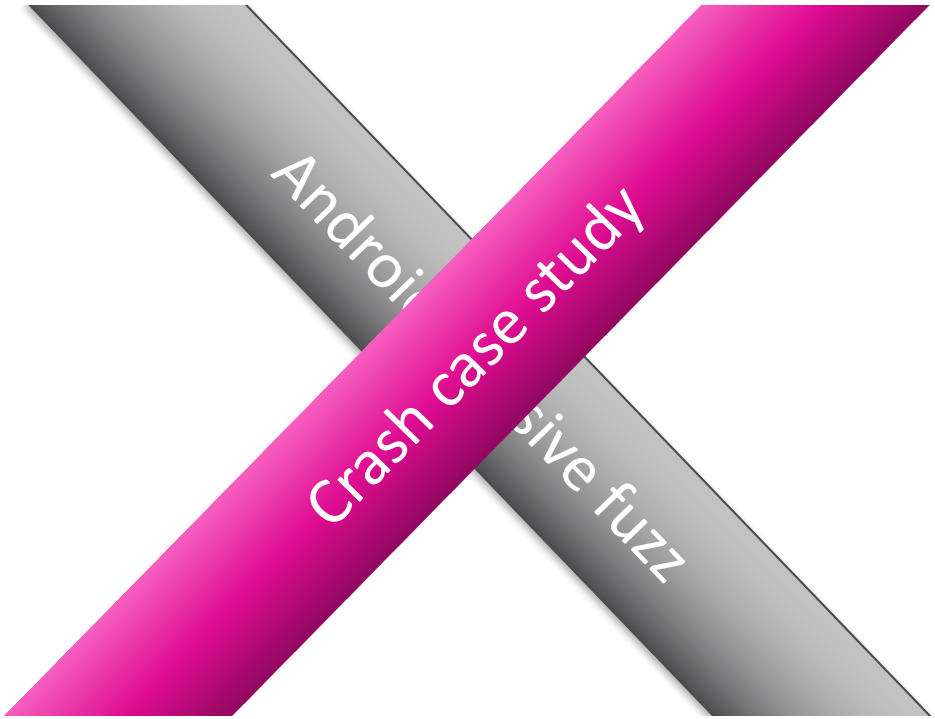
```
int StartFuzz (void *uptr,unsigned int n)
{
……
        int i;
        for(i=0;i<20;i++)
        {
                Fuzz_N_bytes(uptr,n);
        }
……
}
```

Add a trigger program



CVE-2016-3842

```
if(pid){
        while(1){
                arg_free.id = 1;
                ioctl(fd,IOCTL_KGSL_GPUMEM_FREE_ID, &arg_free);
        }
}
ret = ioctl(fd,IOCTL_KGSL_GPUMEM_ALLOC, &arg);
if(ret){
        perror("alloc");
}
```



Case 1

```
long kgsl_ioctl_gpumem_alloc_id(struct kgsl_device_private *dev_priv, unsigned int cmd, void *data)
{
    ……
    if (!kgsl_mmu_use_cpu_map(&device->mmu))
        param->flags &= ~KGSL_MEMFLAGS_USE_CPU_MAP;
    result = _gpumem_alloc(dev_priv, &entry, param->size, param->flags);      //the param->flags and param->size are controlled by user
    if (result != 0)
        goto err;
    ……
}
```

Case 1

```
static inline int kgsl_allocate_user(struct kgsl_device *device,
                struct kgsl_memdesc *memdesc,
                struct kgsl_pagetable *pagetable,
                size_t size, unsigned int flags)
{
        ......
        if (kgsl_mmu_get_mmutype() == KGSL_MMU_TYPE_NONE) {
                size = ALIGN(size, PAGE_SIZE);
                ret = kgsl_cma_alloc_coherent(device, memdesc, pagetable, size);
        } else if (flags & KGSL_MEMFLAGS_SECURE)
                ret = kgsl_cma_alloc_secure(device, memdesc, size);// crash in this function, because of too large size.
        else
                ret = kgsl_sharedmem_page_alloc_user(memdesc, pagetable, size);

        return ret;
}
```

Case 1

```
unsigned long dma_alloc_from_contiguous(struct device *dev, size_t count, unsigned int align)
{
unsigned long mask, pfn = 0, pageno, start = 0;
        struct cma *cma = dev_get_cma_area(dev);
......
for (;;) {
                mutex_lock(&cma->lock);
                pageno = bitmap_find_next_zero_area(cma->bitmap, cma->count,
                                                    start, count, mask);
                if (pageno >= cma->count) {
......
                pfn = cma->base_pfn + pageno;
                if (cma->in_system) {
                        mutex_lock(&cma_mutex);
                        ret = alloc_contig_range(pfn, pfn + count, MIGRATE_CMA);
                        mutex_unlock(&cma_mutex);
                ......
}
```

Case 2

```
long kgsl_ioctl_gpuobj_alloc(struct kgsl_device_private *dev_priv,
            unsigned int cmd, void *data)
{
      struct kgsl_gpuobj_alloc *param = data;
……
      entry = gpumem_alloc_entry(dev_priv, param->size, param->flags);
……
}
```

Case 2

```
int kgsl_allocate_user(struct kgsl_device *device,
                struct kgsl_memdesc *memdesc,
                uint64_t size, uint64_t flags)
{
        if (kgsl_mmu_get_mmutype(device) == KGSL_MMU_TYPE_NONE)
                ret = kgsl_sharedmem_alloc_contig(device, memdesc, size);
        else if (flags & KGSL_MEMFLAGS_SECURE)
                ret = kgsl_allocate_secure(device, memdesc, size);
        else
                ret = kgsl_sharedmem_page_alloc_user(memdesc, size);
        return ret;
}
```

Case 2

```
int kgsl_sharedmem_page_alloc_user(struct kgsl_memdesc *memdesc,
                    uint64_t size)
{
        ......
        if (size == 0 || size > UINT_MAX)
                return -EINVAL;
        align = (memdesc->flags & KGSL_MEMALIGN_MASK) >> KGSL_MEMALIGN_SHIFT;
......
        page_size = kgsl_get_page_size(size, align);
......
        len_alloc = PAGE_ALIGN(size) >> PAGE_SHIFT;
......
        while (len > 0) {
                int page_count;
                page_count = kgsl_pool_alloc_page(&page_size,
                                memdesc->pages + pcount,
                                len_alloc - pcount,
                                &align);
......
}
```

Agenda

- Introduction
- User Space Hourglass
- Kernel Space Hourglass
- Demo

The End

QUESTIONS?