Reviewing the Security of ASoC Drivers in Android Kernel

Seven Shen

Advanced Mobile Threat Research Team, TrendMicro

Bio.

- Security researcher & solution developer @ TrendMicro
- Currently focus on advanced mobile threat research & exploit detection
- 7+ years in security industry
- Keep disclosing Android bugs since 2015
- Hunt bugs not for exploitation, but for deploying protect solution
- Blogs:
  - http://blog.trendmicro.com/
  - http://huntcve.github.io/

FYI.

- What will be covered in this talk?
  - Some bug hunting experiences in Android media framework(both user&kernel spaces)
  - Some typical kernel bugs review(why&how it happens)
  - Potential exploitation chain targets these bugs(how to reach them)
  - Tips to kernel developers for secure coding(ASoC driver developers)
- What will NOT be covered in this talk?
  - A detailed exploit introduction
  - A demo

FYI.

- This research disclosed following kernel bugs(media related) in less than one month of dedicated bug hunting:

  CVE-2016-2064 CVE-2016-2065 CVE-2016-2066 CVE-2016-2068
  CVE-2016-5347 CVE-2016-5853 CVE-2016-5858 CVE-2016-5859
  CVE-2016-5862 CVE-2016-5867 CVE-2016-6693 CVE-2016-6694
  CVE-2016-6695 CVE-2016-10231 CVE-2017-0578 CVE-2017-0586
  CVE-2017-0608 CVE-2017-6247 CVE-2017-6248 CVE-2017-6249
  CVE-2017-7369 CVE-2017-8246

- All of the bugs have been responsibly disclosed to vendors and now fixed

Agenda

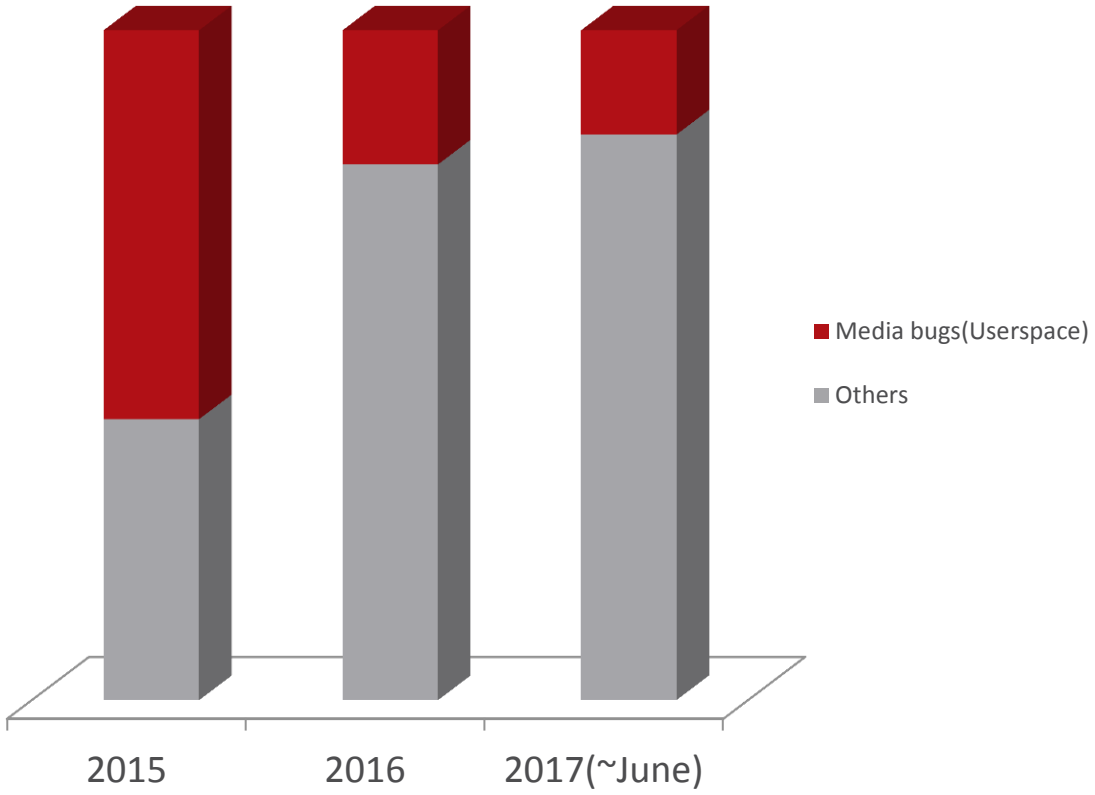
- Background
- Fuzz a userspace media module
- The ASoC/ALSA in kernel
- The attack surface & the issue
- Typical kernel bugs go through
- Thinking in exploitation
- Conclusion

Background

- A lot of Android “Media Server” bugs have been disclosed continuously since Aug, 2015
- Those bugs spread from “libstagefright”, to “openMAX”, then to SW/HW codecs
- Most of userspace media modules are affected
- It becomes red sea!
- However, few bugs in kernel of that part were disclosed(before the 2nd half of 2016)
- SO I decided to look at this

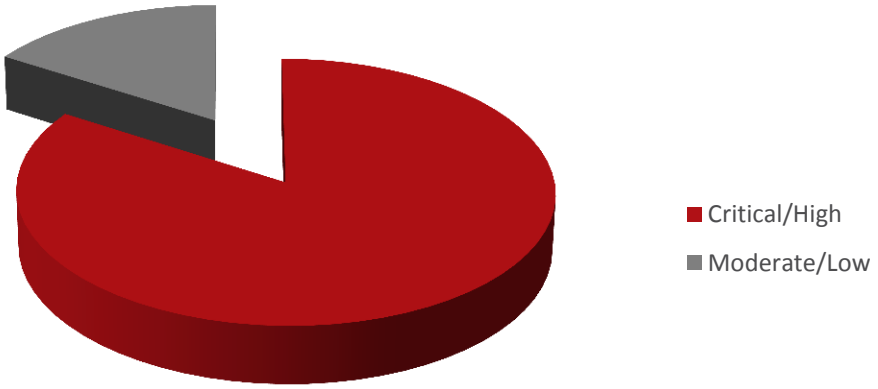
How&Why Media bugs become so HOT?

Android media bugs keeps attractive



Based on Android security bulletin

Most media bugs are assessed as High/Critical



Based on Android security bulletin

The reason(possibly):

- Media bugs are born with “Remoteable” gene
- This means higher bug bounty ,even a DoS(based on previous Android security guidelines)
- Still, the real-life exploitability has been proved: (https://github.com/NorthBit/Metaphor)
- Easy to fuzz?

How is it easy to get a media bug?
Let’s do it.

Which media module?

- In order to practice an effective fuzz experience, I prefer the “American Fuzzy Lop” (AFL)
- I only want to fuzz a single module with less dependency(more dedicated, more effective)
- Based on this, the third-party modules in /external are in scope
- Several of them are media related(most are SW codecs), Such as:
  - Libavc
  - Libhevc
  - Libmpeg2
  - LibVPX

  …

Some tips

- You'd better not fuzz with test codes in those libs because Android invokes them differently
- Understand how Android invokes them is necessary, this can help you write a Proof-of-Concept(PoC) quickly

How SW codec APIs are invoked?

```
void SoftHEVC ::onQueueFilled(OMX_U32 portIndex) {
    ...
    while (!outQueue.empty()) {
        ...
        setDecodeArgs(...);
        ivdec_api_function(..., (void *)&s_dec_ip,…) ; // s_dec_ip often references
to a codec buffer in media files
        ...
    }
        ...
}  // namespace android
```

What is the codec data?

```
00000000h: 00 00 00 1C 66 74 79 70 69 73 6F 6D 00 00 02 00 ; ....ftypisom....
00000010h: 69 73 6F 6D 69 73 6F 32 6D 70 34 31 00 00 00 08 ; isomiso2mp41....
00000020h: 66 72 65 65 00 00 06 1D 6D 64 61 74 00 00 01 FC ; free....mdat...?
00000030h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
00000040h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
00000050h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
00000060h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
00000070h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
00000080h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
00000090h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
000000a0h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
000000b0h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
000000c0h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
000000d0h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
000000e0h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
000000f0h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
00000100h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
00000110h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
00000120h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
00000130h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
00000140h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
00000150h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
00000160h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
00000170h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
00000180h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
00000190h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
000001a0h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
000001b0h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
000001c0h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
000001d0h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
000001e0h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
000001f0h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
00000200h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
00000210h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ;
00000220h: FF FF FF FF FF FF FF FF FF FF FF FF 01 1A 35 2D ;             ..5-
00000230h: 14 78 53 21 04 C1 42 88 5D F5 D6 A6 5E F7 D6 73 ; .xS!.巤坋踔  鬘s
00000240h: 57 88 92 AB 51 70 C9 00 30 6F FE EC D2 C6 24 A3 ; W堉玅p?0o  移$?
00000250h: 94 71 BB 80 DA 53 7F EC 08 F3 15 D1 89 98 7A 80 ; 攓粈赟??椽樫€
```

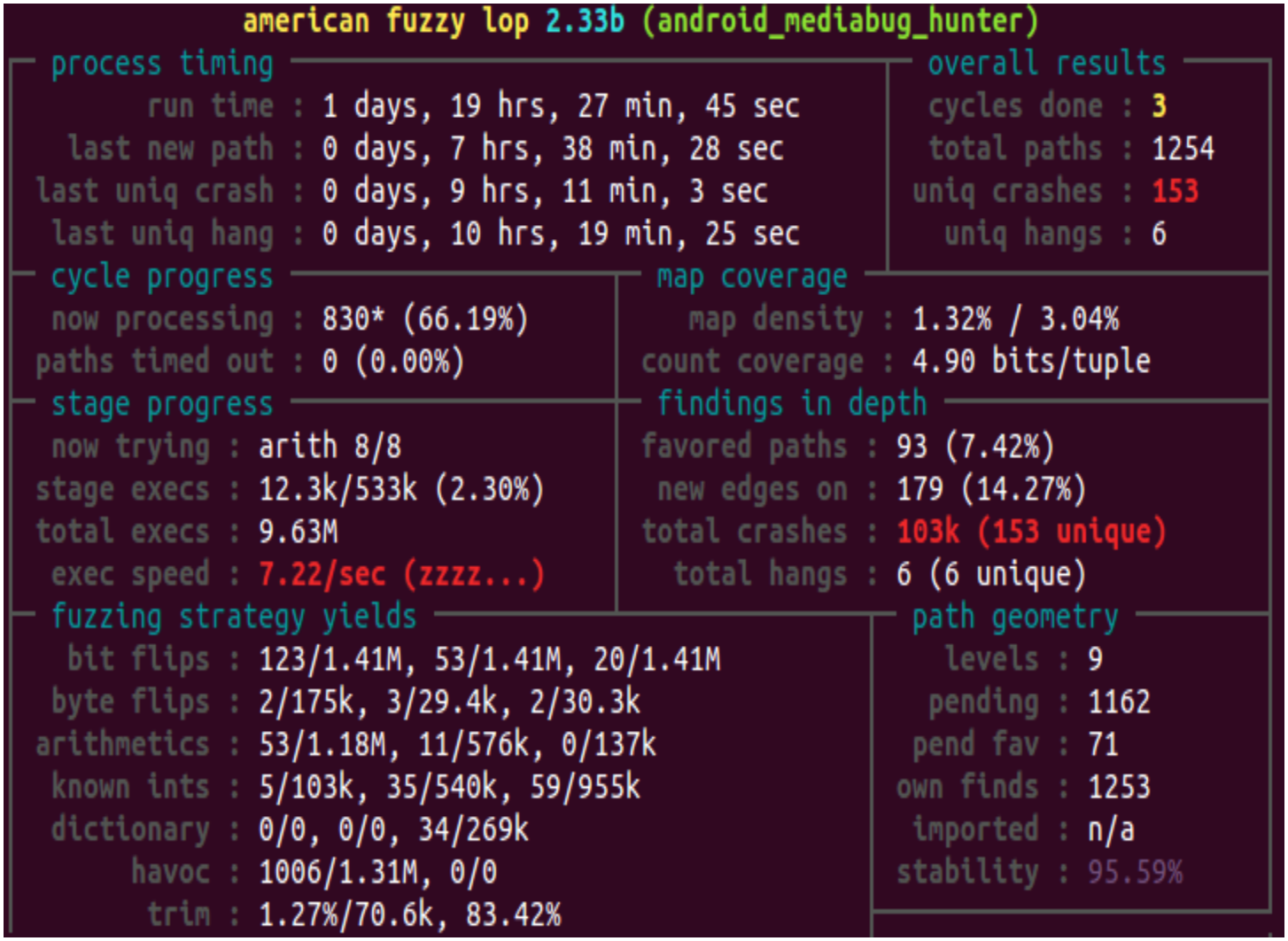
length

codec data(craft)

How to fuzz?

- I know how Android invokes the codec APIs
- I get an arbitrary buffer data, which can be passed to codec libs
- The buffer can be easily built into media files
- The codec module are less dependency, easily to be built with AFL
- This is a typical scenario that AFL perfectly fits
- Write the code, then test it

Hundreds of unique crashes&hangs, all remoteable



My code fuzzer(codecfuzz)

```
-rw-rw-r--  1 seven seven  120 Apr  6 17:32 Android.mk
drwxrwxr-x 12 seven seven 4096 Apr 24 13:32 decoders
-rw-rw-r--  1 seven seven  902 Apr  6 16:00 decoders.aac.mk
-rw-rw-r--  1 seven seven  692 May  4 13:36 decoders.avc.mk
-rw-rw-r--  1 seven seven  699 Apr 18 17:17 decoders.hevc.mk
-rw-rw-r--  1 seven seven  508 Apr 18 15:50 decoders.mk
-rw-rw-r--  1 seven seven  719 Apr 18 15:53 decoders.mp3.mk
-rw-rw-r--  1 seven seven  697 Apr 28 15:08 decoders.mpeg2.mk
-rw-rw-r--  1 seven seven  753 Apr  6 16:05 decoders.mpeg4.mk
-rw-rw-r--  1 seven seven  638 Apr  6 16:05 decoders.opus.mk
-rw-rw-r--  1 seven seven  653 Apr  6 16:06 decoders.vorbis.mk
-rw-rw-r--  1 seven seven  748 Apr 24 09:51 decoders.vpx.mk
drwxrwxr-x  3 seven seven 4096 Apr  6 16:19 encoders
-rw-rw-r--  1 seven seven  919 Apr  6 17:30 encoders.avc.mk
-rw-rw-r--  1 seven seven  139 Apr  6 17:37 encoders.mk
drwxrwxr-x  2 seven seven 4096 May 16 15:55 extra
-rw-rw-r--  1 seven seven 2509 Apr 26 17:19 fuzzfs.cpp
drwxrwxr-x  2 seven seven 4096 Apr 19 12:34 nonafllibs
drwxrwxr-x  2 seven seven 4096 May 11 10:55 out
drwxrwxr-x  3 seven seven 4096 May  9 17:49 scripts
drwxrwxr-x  2 seven seven 4096 Apr 25 16:42 staticlibs
```

- Cover most of SW codec decoders
- Cover part of SW codec encoders
- Extremely effective
- Open sourced to Google(Android security team)

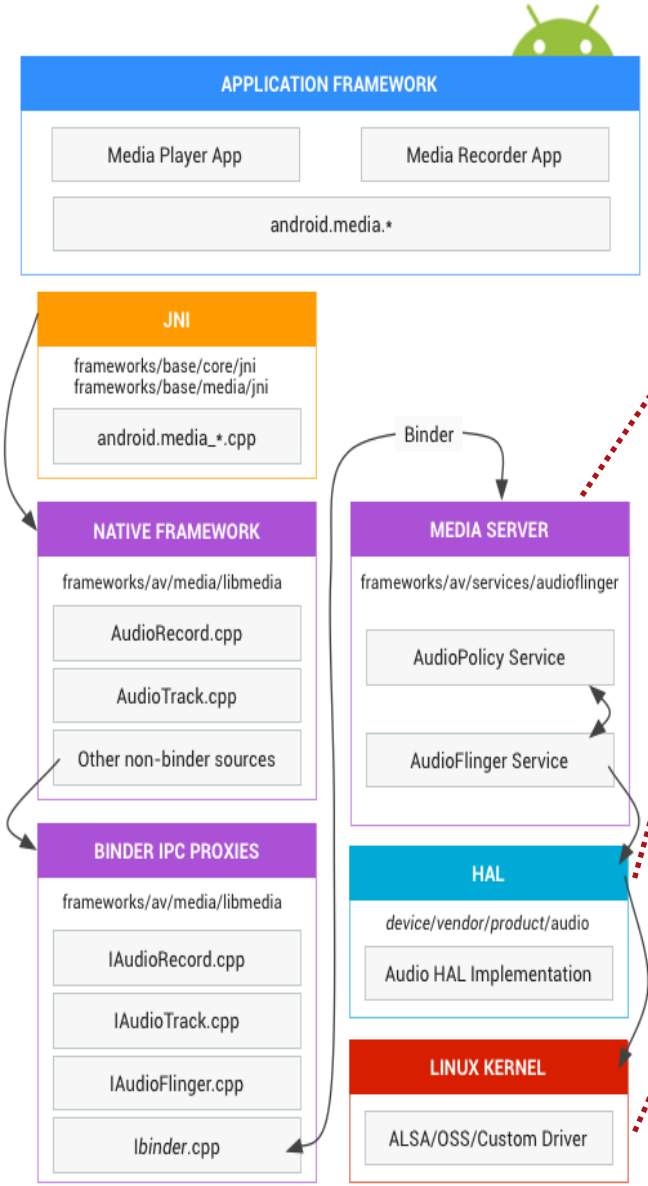
This is just the beginning…

Let’s move to kernel part

- I could imagine: there should be two parts in kernel that handle media things:
  - The Audio driver
  - The Video driver

- This time I look at the Audio part

The Android Audio Architecture



CVEs based on Android security bulletin (https://source.android.com/security/bulletin/index.html)

CVE-2015-1538 CVE-2015-3873 CVE-2016-0803 CVE-2016-0835
CVE-2015-1539 CVE-2015-3872 CVE-2016-0804 CVE-2016-0836
CVE-2015-3824 CVE-2015-3871 CVE-2016-0815 CVE-2016-0837
CVE-2015-3827 CVE-2015-3868 CVE-2016-0816 CVE-2016-0838
CVE-2015-3828 CVE-2015-3867 CVE-2016-0841 CVE-2016-0839
CVE-2015-3864 CVE-2015-3869 CVE-2016-2428 CVE-2016-0840…

CVE-2016-2450 CVE-2016-2451 CVE-2016-2452 CVE-2016-2477 CVE-2016-2478 CVE-2016-2479 CVE-2016-2480 CVE-2016-2481 CVE-2016-2482 CVE-2016-2483 CVE-2016-2484 CVE-2016-2485 CVE-2016-2486 CVE-2016-3746 CVE-2016-3747 CVE-2016-3765 CVE-2016-3844 CVE-2016-3835 CVE-2016-3823 CVE-2016-3824 CVE-2016-3825 CVE-2016-6758 CVE-2016-6761 CVE-2016-6760 …

CVE-2016-2466 CVE-2016-2467 CVE-2016-2469 CVE-2016-3866

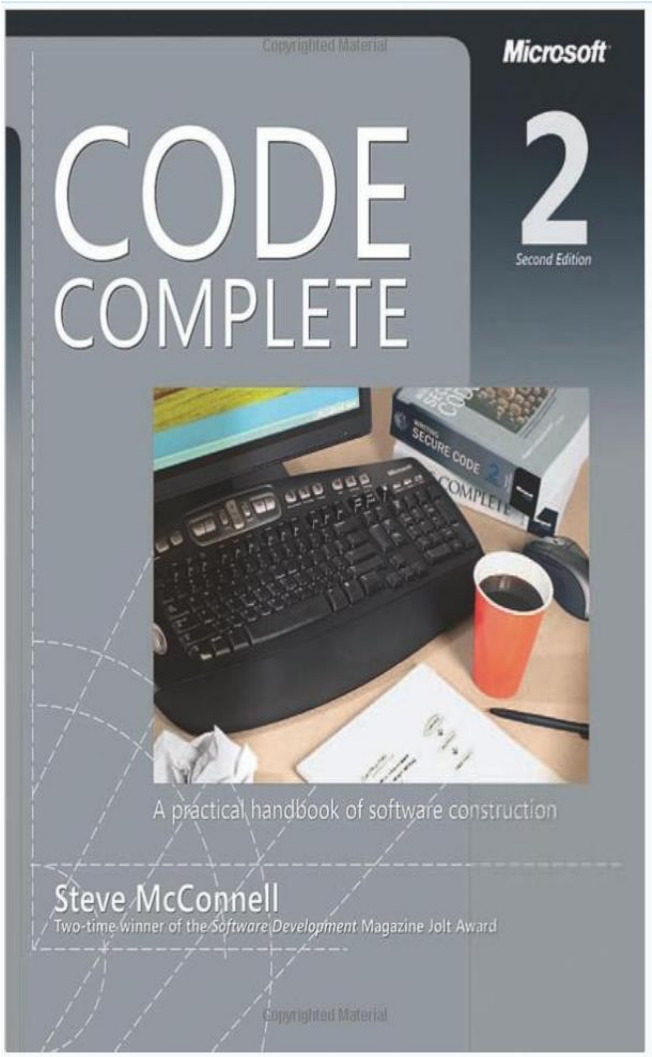
~2016.6

Can’t believe!!

https://source.android.com/devices/audio/

How many bugs is acceptable?

- Industry Average: "about 15 - 50 errors per 1000 lines of delivered code."
- Microsoft Applications: "about 10 - 20 defects per 1000 lines of code during in-house testing, and 0.5 defect per KLOC (KLOC IS CALLED AS 1000 lines of code) in released product (Moore 1992)."
- A few projects - for example, the space-shuttle software - have achieved a level of 0 defects in 500,000 lines of code using a system of format development methods, peer reviews, and statistical testing.



If we follow Microsoft standards

```
seven@seven-pc:~/source/kernel/angler7/msm$ cloc \
> sound/core/ \
> drivers/misc/qcom/qdsp6v2/ \
> sound/soc/msm/qdsp6v2/ \
> sound/soc/codecs/
     537 text files.
     537 unique files.
     109 files ignored.

http://cloc.sourceforge.net v 1.60  T=2.74 s (158.3 files/s, 134201.1 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
C                              274          38763          19494         254174
C/C++ Header                   151           4665          10018          39561
make                             8             20             19            352
-------------------------------------------------------------------------------
SUM:                           433          43448          29531         294087
-------------------------------------------------------------------------------
```

There should be: 254 *0.5= 127 bugs!!

We have a long way to go…

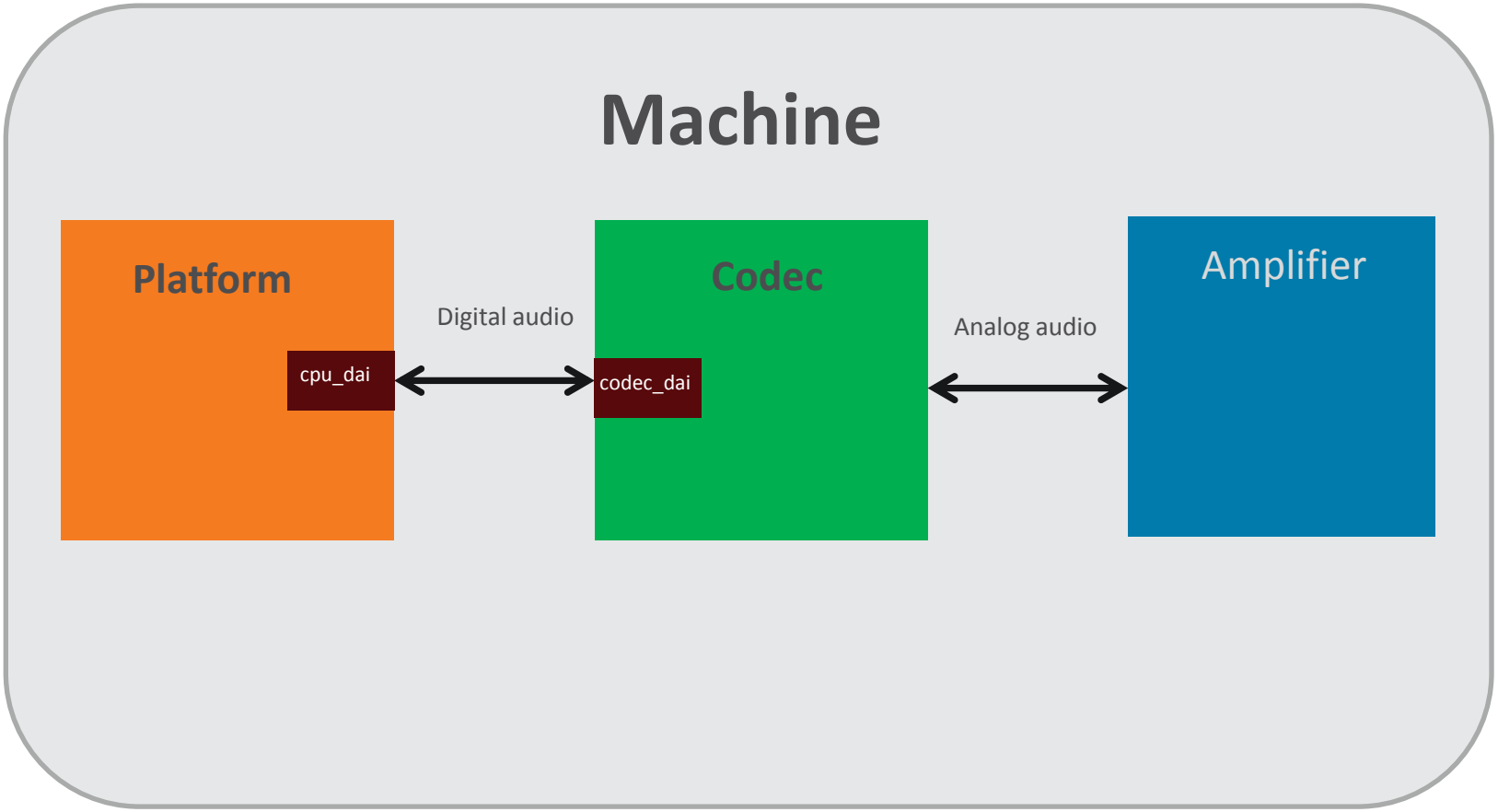
What is ALSA?

- **Advanced Linux Sound Architecture** (**ALSA**) is a software framework and part of the Linux kernel that provides an application programming interface (API) for sound card device drivers.
- Started in 1998 and introduced into Linux kernel from 2.5 development series in 2002 (2.5.4–2.5.5).
- Replaced **Open Sound System** (OSS) since kernel 2.6.
- Inherited by Android.

The ASoC

- The **ALSA System on Chip** (ASoC) layer is to provide better ALSA support for embedded system on chip processors and portable audio codecs
- Designed to address these issues:
  - Codec independence
  - Easy I2S/PCM audio interface setup between codec and SoC
  - Dynamic Audio Power Management (DAPM)
  - Pop and click reduction
  - Machine specific controls

ASoC architecture

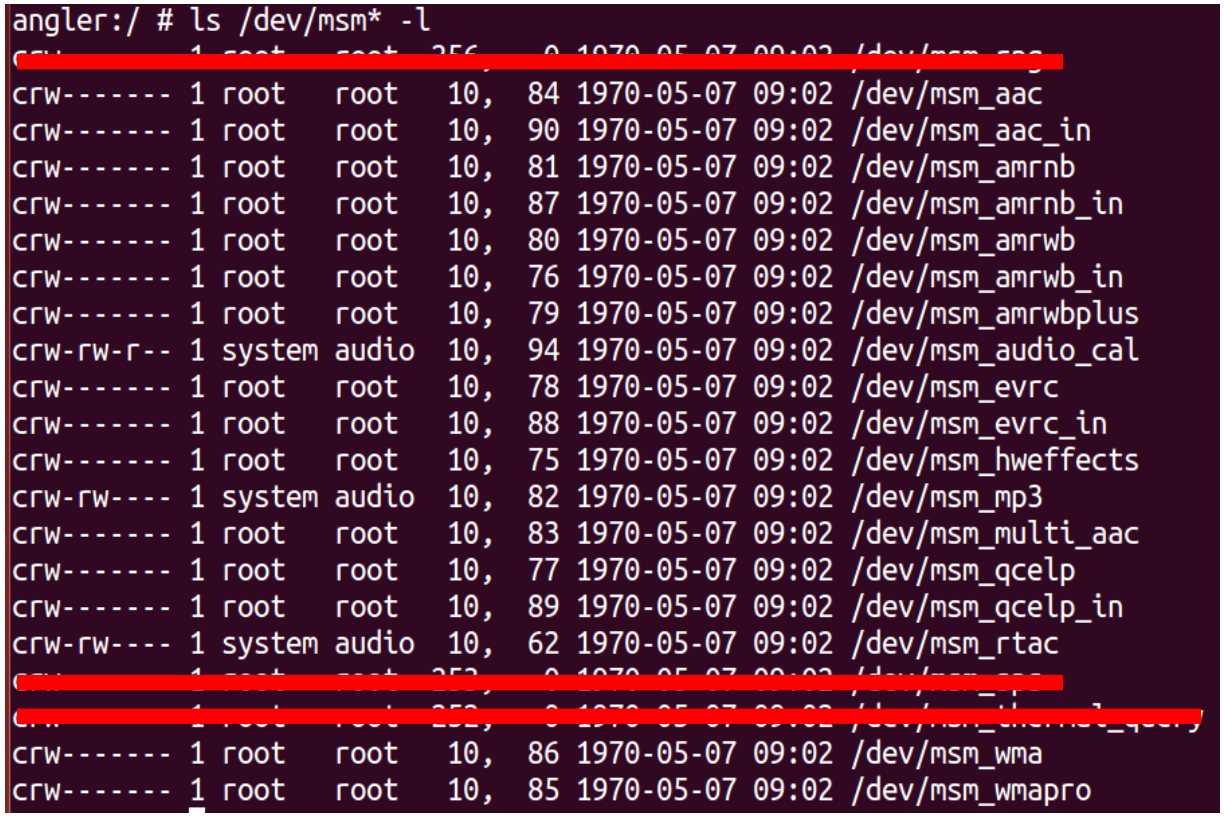


Attack surface I:

```
angler:/ # ls /dev/snd
comprC0D17 hwC0D1000 hwC0D24 hwC0D45   pcmC0D12p pcmC0D1p  pcmC0D26p pcmC0D35c pcmC0D4p
comprC0D18 hwC0D11   hwC0D25 hwC0D7    pcmC0D13c pcmC0D20c pcmC0D27c pcmC0D35p pcmC0D5c
comprC0D37 hwC0D12   hwC0D26 hwC0D8    pcmC0D13p pcmC0D20p pcmC0D28c pcmC0D36c pcmC0D5p
comprC0D38 hwC0D13   hwC0D3  hwC0D9    pcmC0D14c pcmC0D21p pcmC0D29c pcmC0D36p pcmC0D6c
comprC0D39 hwC0D14   hwC0D30 pcmC0D0c  pcmC0D14p pcmC0D22c pcmC0D2c  pcmC0D3c  pcmC0D7p
comprC0D40 hwC0D15   hwC0D31 pcmC0D0p  pcmC0D15c pcmC0D22p pcmC0D2p  pcmC0D3p  pcmC0D8c
comprC0D41 hwC0D16   hwC0D35 pcmC0D10c pcmC0D15p pcmC0D23c pcmC0D30c pcmC0D43c timer
comprC0D42 hwC0D2    hwC0D36 pcmC0D10p pcmC0D16c pcmC0D23p pcmC0D31c pcmC0D44c
comprC0D9  hwC0D20   hwC0D37 pcmC0D11c pcmC0D19c pcmC0D24c pcmC0D32c pcmC0D44p
controlC0  hwC0D21   hwC0D39 pcmC0D11p pcmC0D19p pcmC0D24p pcmC0D33c pcmC0D45c
hwC0D10    hwC0D22   hwC0D40 pcmC0D12c pcmC0D1c  pcmC0D25c pcmC0D34c pcmC0D45p
```

SNDRV_DEVICE_TYPE_CONTROL:“controlC%i”, card->numer
SNDRV_DEVICE_TYPE_COMPRESS: “comprC%iD%i”, card->number, compress->device
SNDRV_DEVICE_TYPE_HWDEP: “hwC%iD%i”, card->number, hwdep->device
SNDRV_DEVICE_TYPE_PCM_PLAYBACK: “pcmC%iD%ip”, card->numer, pcm->device
SNDRV_DEVICE_TYPE_PCM_CAPTURE: “pcmC%iD%ic”, card->numer, pcm->device

Attack surface II:



/dev/snd/controlC0

```
struct snd_kcontrol_new {
        snd_ctl_elem_iface_t iface;     /* interface identifier */
        unsigned int device;            /* device/client number */
        unsigned int subdevice;         /* subdevice (substream) number */
        const unsigned char *name;      /* ASCII name of item */
        unsigned int index;             /* index of item */
        unsigned int access;            /* access rights */
        unsigned int count;             /* count of same elements */
        snd_kcontrol_info_t *info;
        snd_kcontrol_get_t *get;
        snd_kcontrol_put_t *put;
        union {
                snd_kcontrol_tlv_rw_t *c;
                const unsigned int *p;
        } tlv;
        unsigned long private_value;
};
```

Userspace accessible interfaces

<sound/control.h>

/dev/snd/controlC0

```
switch (cmd) {
case SNDRV_CTL_IOCTL_PVERSION:
        return put_user(SNDRV_CTL_VERSION, ip) ? -EFAULT : 0;
case SNDRV_CTL_IOCTL_CARD_INFO:
        return snd_ctl_card_info(card, ctl, cmd, argp);
case SNDRV_CTL_IOCTL_ELEM_LIST:
        return snd_ctl_elem_list(card, argp);
case SNDRV_CTL_IOCTL_ELEM_INFO:
        return snd_ctl_elem_info_user(ctl, argp);
case SNDRV_CTL_IOCTL_ELEM_READ:
        return snd_ctl_elem_read_user(card, argp);
case SNDRV_CTL_IOCTL_ELEM_WRITE:
        return snd_ctl_elem_write_user(ctl, argp);
case SNDRV_CTL_IOCTL_ELEM_LOCK:
        return snd_ctl_elem_lock(ctl, argp);
case SNDRV_CTL_IOCTL_ELEM_UNLOCK:
        return snd_ctl_elem_unlock(ctl, argp);
case SNDRV_CTL_IOCTL_ELEM_ADD:
        return snd_ctl_elem_add_user(ctl, argp, 0);
```

“sound/core/control.c”

/dev/snd/controlC0: put()

```
static int snd_ctl_elem_write_user(struct snd_ctl_file *file,
                                   struct snd_ctl_elem_value __user *_control)
{
        struct snd_ctl_elem_value *control;
        struct snd_card *card;
        int result;

        control = memdup_user(_control, sizeof(*control));
        if (IS_ERR(control))
                return PTR_ERR(control);

        card = file->card;
        snd_power_lock(card);
        result = snd_power_wait(card, SNDRV_CTL_POWER_D0);
        if (result >= 0)
                result = snd_ctl_elem_write(card, file, control);
        snd_power_unlock(card);
        if (result >= 0)
                if (copy_to_user(_control, control, sizeof(*control)))
                        result = -EFAULT;
        kfree(control);
        return result;
}
```

/dev/snd/controlC0: put()

```
static int snd_ctl_elem_write(struct snd_card *card, struct snd_ctl_file *file,
                              struct snd_ctl_elem_value *control)
{
        struct snd_kcontrol *kctl;
        struct snd_kcontrol_volatile *vd;
        unsigned int index_offset;
        int result;

        down_read(&card->controls_rwsem);
        kctl = snd_ctl_find_id(card, &control->id);
        if (kctl == NULL) {
                result = -ENOENT;
        } else {
                index_offset = snd_ctl_get_ioff(kctl, &control->id);
                vd = &kctl->vd[index_offset];
                if (!(vd->access & SNDRV_CTL_ELEM_ACCESS_WRITE) ||
                    kctl->put == NULL ||
                    (file && vd->owner && vd->owner != file)) {
                        result = -EPERM;
                } else {
                        snd_ctl_build_ioff(&control->id, kctl, index_offset);
                        result = kctl->put(kctl, control);
                }
```

The user space buffer

```
struct snd_ctl_elem_value {
        struct snd_ctl_elem_id id;      /* W: element ID */
        unsigned int indirect: 1;       /* W: indirect access - obsoleted */
        union {
                union {
                        long value[128];
                        long *value_ptr;        /* obsoleted */
                } integer;
                union {
                        long long value[64];
                        long long *value_ptr;   /* obsoleted */
                } integer64;
                union {
                        unsigned int item[128];
                        unsigned int *item_ptr; /* obsoleted */
                } enumerated;
                union {
                        unsigned char data[512];
                        unsigned char *data_ptr;        /* obsoleted */
                } bytes;
                struct snd_aes_iec958 iec958;
        } value;                /* RO */
        struct timespec tstamp;
        unsigned char reserved[128-sizeof(struct timespec)];
};
```

<include/uapi/sound/asound.h>

What is the “element id”?

```
struct snd_kcontrol *snd_ctl_find_id(struct snd_card *card,
                                     struct snd_ctl_elem_id *id)
{
        struct snd_kcontrol *kctl;

        if (snd_BUG_ON(!card || !id))
                return NULL;
        if (id->numid != 0)
                return snd_ctl_find_numid(card, id->numid);
```

```
struct snd_kcontrol *snd_ctl_find_numid(struct snd_card *card, unsigned int numid)
{
        struct snd_kcontrol *kctl;

        if (snd_BUG_ON(!card || !numid))
                return NULL;
        list_for_each_entry(kctl, &card->controls, list) {
                if (kctl->id.numid <= numid && kctl->id.numid + kctl->count > numid)
                        return kctl;
        }
        return NULL;
}
```

Is “numid” incremental?

```
struct snd_ctl_elem_id {
        unsigned int numid;             /* numeric identifier, zero = invalid */
        snd_ctl_elem_iface_t iface;     /* interface identifier */
        unsigned int device;            /* device/client number */
        unsigned int subdevice;         /* subdevice (substream) number */
        unsigned char name[44];         /* ASCII name of item */
        unsigned int index;             /* index of item */
};
```

<include/uapi/sound/asound.h>

Let’s dump it

```
----------------card info-----------
card:0
id:msm8994tomtommt
name:msm8994-tomtom-mtp-snd-card
longname:msm8994-tomtom-mtp-snd-card
mixername:
components:
get element list
------------elem list------------
offset:0x0
space:0x4000
used:0x3f6
count:0x3f6
   numid:   1,  name:Voice Rx Device Mute, iface:2
   numid:   2,  name:Voice Tx Device Mute, iface:2
   numid:   3,  name:Voice Tx Mute, iface:2
   numid:   4,  name:Voice Rx Gain, iface:2
   numid:   5,  name:TTY Mode, iface:2
   numid:   6,  name:Slowtalk Enable, iface:2
   numid:   7,  name:Voice Topology Disable, iface:2
   numid:   8,  name:HD Voice Enable, iface:2
   numid:   9,  name:CVD Version, iface:2
   numid:  10,  name:Voip Tx Mute, iface:2
   numid:  11,  name:Voip Rx Gain, iface:2
   numid:  12,  name:Voip Mode Config, iface:2
   numid:  13,  name:Voip Rate Config, iface:2
   numid:  14,  name:Voip Evrc Min Max Rate Config, iface:2
   numid:  15,  name:Voip Dtx Mode, iface:2
   numid:  16,  name:Compress Gapless Playback, iface:2
   numid:  17,  name:EAR PA Gain, iface:2
   numid:  18,  name:HPHL Volume, iface:2
```

```
numid: 979,  name:Audio Stream 40 Dec Params, iface:2
numid: 980,  name:Audio Stream 40 App Type Cfg, iface:2
numid: 981,  name:Playback Channel Map40, iface:2
numid: 982,  name:Compress Playback 41 Volume, iface:2
numid: 983,  name:Audio Effects Config 41, iface:2
numid: 984,  name:Query Audio Effect Param 41, iface:2
numid: 985,  name:Audio Stream 41 Dec Params, iface:2
numid: 986,  name:Audio Stream 41 App Type Cfg, iface:2
numid: 987,  name:Playback Channel Map41, iface:2
numid: 988,  name:Compress Playback 42 Volume, iface:2
numid: 989,  name:Audio Effects Config 42, iface:2
numid: 990,  name:Query Audio Effect Param 42, iface:2
numid: 991,  name:Audio Stream 42 Dec Params, iface:2
numid: 992,  name:Audio Stream 42 App Type Cfg, iface:2
numid: 993,  name:Playback Channel Map42, iface:2
numid: 994,  name:Speaker Function, iface:2
numid: 995,  name:SLIM_0_RX Channels, iface:2
numid: 996,  name:SLIM_5_RX Channels, iface:2
numid: 997,  name:SLIM_0_TX Channels, iface:2
numid: 998,  name:VI_FEED_TX Channels, iface:2
numid: 999,  name:AUX PCM SampleRate, iface:2
numid:1000,  name:HDMI_RX Channels, iface:2
numid:1001,  name:SLIM_0_RX Format, iface:2
numid:1002,  name:SLIM_5_RX Format, iface:2
numid:1003,  name:SLIM_0_RX SampleRate, iface:2
numid:1004,  name:SLIM_5_RX SampleRate, iface:2
numid:1005,  name:HDMI_RX Bit Format, iface:2
numid:1006,  name:PROXY_RX Channels, iface:2
numid:1007,  name:Internal BTSCO SampleRate, iface:2
numid:1008,  name:HDMI_RX SampleRate, iface:2
numid:1009,  name:SLIM_0_TX Format, iface:2
numid:1010,  name:SLIM_0_TX SampleRate, iface:2
numid:1011,  name:TERT_MI2S BitWidth, iface:2
numid:1012,  name:TERT_MI2S SampleRate, iface:2
numid:1013,  name:QUAT_MI2S BitWidth, iface:2
numid:1014,  name:QUAT_MI2S SampleRate, iface:2
----------elem list------------
```

How is this put() used?

```
static int msm_slim_5_rx_ch_put(struct snd_kcontrol *kcontrol,
                                struct snd_ctl_elem_value *ucontrol)
{
        msm_slim_5_rx_ch = ucontrol->value.integer.value[0] + 1;
        pr_debug("%s: msm_slim_5_rx_ch = %d\n", __func__,
                 msm_slim_5_rx_ch);
        return 1;
}

static int msm_slim_0_rx_ch_get(struct snd_kcontrol *kcontrol,
        struct snd_ctl_elem_value *ucontrol)
{
        pr_debug("%s: msm_slim_0_rx_ch  = %d\n", __func__,
                 msm_slim_0_rx_ch);
        ucontrol->value.integer.value[0] = msm_slim_0_rx_ch - 1;
        return 0;
}

static int msm_slim_0_rx_ch_put(struct snd_kcontrol *kcontrol,
        struct snd_ctl_elem_value *ucontrol)
{
        msm_slim_0_rx_ch = ucontrol->value.integer.value[0] + 1;

        pr_debug("%s: msm_slim_0_rx_ch = %d\n", __func__,
                 msm_slim_0_rx_ch);
        return 1;
}
```

How is this put() used?

```
if (substream->stream == SNDRV_PCM_STREAM_PLAYBACK) {
        pr_err("%s: rx_0_ch=%d\n", __func__, msm_slim_0_rx_ch);
        ret = snd_soc_dai_get_channel_map(codec_dai,
                                &tx_ch_cnt, tx_ch, &rx_ch_cnt , rx_ch);
        if (ret < 0) {
                pr_err("%s: failed to get codec chan map, err:%d\n",
                        __func__, ret);
                goto end;
        }
        if (dai_link->be_id == MSM_BACKEND_DAI_SLIMBUS_5_RX) {
                pr_err("%s: rx_5_ch=%d\n", __func__,
                        msm_slim_5_rx_ch);
                rx_ch_count = msm_slim_5_rx_ch;
        } else {
                pr_err("%s: rx_0_ch=%d\n", __func__,
                        msm_slim_0_rx_ch);
                rx_ch_count = msm_slim_0_rx_ch;
        }
        ret = snd_soc_dai_set_channel_map(cpu_dai, 0, 0,
                                        rx_ch_count, rx_ch);
        if (ret < 0) {
                pr_err("%s: failed to set cpu chan map, err:%d\n",
                        __func__, ret);
                goto end;
        }
```

How many puts?

```
seven@seven-pc:~/source/kernel/angler711/msm$ grep -rn "struct snd_ctl_elem_value" | wc -l
2189
```

Let’s fuzz it!!

```c
int main()
{
  struct snd_ctl_elem_value c;
  int fd = open("/dev/snd/controlC0", O_RDWR);
  c.value.enumerated.item[0] = c.value.integer.value[0]= 0x80001111;
  for(c.id.numid=1;c.id.numid<= 1555;c.id.numid++){
      ioctl(fd, SNDRV_CTL_IOCTL_ELEM_WRITE,&c);
  }
  close(fd);
}
```

Dozens of kernel crash happened!!!

Bug types:

- Stack buffer overflow
- Heap buffer overflow
- Out-of-Bounds Access
- Use-after-Free
- Double-Free
- Race condition
- Type confusion
- Uninitiated stack variable leakage
- Null pointer dereference

The Buggy Ecosystem

Case 1: Out-of-bounds access(Qualcomm)

```
[20161125_17:42:49.487916]@3 Unable to handle kernel paging request at virtual address ffffffbc00d60758
[20161125_17:42:49.487924]@3 pgd = ffffffc0414c5000
[20161125_17:42:49.487929]@3 [ffffffbc00d60758] *pgd=0000000000000000, *pud=0000000000000000
[20161125_17:42:49.487943]@3 Internal error: Oops: 96000005 [#1] PREEMPT SMP
[20161125_17:42:49.487950]@3 Modules linked in: wlan(O) crpl(PO)
[20161125_17:42:49.487967]@3 CPU: 3 PID: 5417 Comm: fuzz Tainted: P O 3.18.20-perf+ #1
[20161125_17:42:49.487974]@3 Hardware name: Qualcomm Technologies, Inc. MSM 8996 v3 + PMI8996 MTP (DT)
[20161125_17:42:49.487982]@3 task: ffffffc054dd5c00 ti: ffffffc09fcf0000 task.ti: ffffffc09fcf0000
[20161125_17:42:49.487993]@3 PC is at soc_dapm_mux_update_power.isra.31+0x54/0xd8
[20161125_17:42:49.488000]@3 LR is at snd_soc_dapm_mux_update_power+0x50/0x84
[20161125_17:42:49.488006]@3 pc : [<ffffffc000958de4>] lr : [<ffffffc000958eb8>] pstate: 80000145
[20161125_17:42:49.488011]@3 sp : ffffffc09fcf3bf0
 ...
[20161125_17:42:49.489411]@3 Process fuzz (pid: 5417, stack limit = 0xffffffc09fcf0060)
[20161125_17:42:49.489416]@3 Call trace:
[20161125_17:42:49.489427]@3 [<ffffffc000958de4>] soc_dapm_mux_update_power.isra.31+0x54/0xd8
[20161125_17:42:49.489433]@3 [<ffffffc000958eb4>] snd_soc_dapm_mux_update_power+0x4c/0x84
[20161125_17:42:49.489441]@3 [<ffffffc0009a1b28>] msm_routing_ec_ref_rx_put+0x204/0x22(
[20161125_17:42:49.489450]@3 [<ffffffc000925b30>] snd_ctl_elem_write+0xd0/0x154
[20161125_17:42:49.489456]@3 [<ffffffc000927be4>] snd_ctl_ioctl+0x3d0/0x640
[20161125_17:42:49.489463]@3 [<ffffffc00019d8e8>] do_vfs_ioctl+0x490/0x570
[20161125_17:42:49.489469]@3 [<ffffffc00019da24>] SyS_ioctl+0x5c/0x88
[20161125_17:42:49.489475]@3 Code: 54000360 f9400260 b40002c0 f94002e1 (f8786821)
[20161125_17:42:49.490436]@3 ---[ end trace c2dffec51cfed2a1 ]---
[20161125_17:42:49.495614]@3 Kernel panic - not syncing: Fatal exception
```

```
static int msm_routing_ec_ref_rx_put(struct snd_kcontrol *kcontrol,
                struct snd_ctl_elem_value *ucontrol)
{
  ...
  int mux = ucontrol->value.enumerated.item[0];
  ...
  snd_soc_dapm_mux_update_power(widget, kcontrol, mux, e);
  ...
}

static int soc_dapm_mux_update_power(struct snd_soc_dapm_widget *widget,
       struct snd_kcontrol *kcontrol, int mux, struct soc_enum *e)
{
  ...
  if (!path->name || !e->texts[mux])
    continue;

  found = 1;
  /* we now need to match the string in the enum to the path */
  if (!(strcmp(path->name, e->texts[mux]))) {
```

Case 2: Null pointer dereference(Qualcomm)

```
[19779.817485] Unable to handle kernel NULL pointer dereference at virtual address 00000158
[19779.817491] pgd = ffffffc0580e3000
[19779.817495] [00000158] *pgd=0000000000000000
[19779.817509] Internal error: Oops: 96000005 [#1] PREEMPT SMP
[19779.817520] CPU: 1 PID: 14151 Comm: fuzz Tainted: G        W    3.10.73-gc5a17ac-dirty #30
[19779.817526] task: ffffffc05f78c080 ti: ffffffc0bae7c000 task.ti: ffffffc0bae7c000
[19779.817538] PC is at msm_pcm_volume_ctl_put+0x48/0x98
[19779.817544] LR is at msm_pcm_volume_ctl_put+0x44/0x98
[19779.817550] pc : [<ffffffc000ac0ebc>] lr : [<ffffffc000ac0eb8>] pstate: 60000145
[19779.817555] sp : ffffffc0bae7fc50
...
[19779.817693] Process fuzz (pid: 14151, stack limit = 0xffffffc0bae7c058)
[19779.817697] Call trace:
ys%3Aaccount@1480663585422.txt19779.817705] [<ffffffc000ac0ebc>] msm_pcm_volume_ctl_put+0x48/0x98
[19779.817715] [<ffffffc000a3c1e4>] snd_ctl_elem_write+0x110/0x1a0
[19779.817723] [<ffffffc000a3d094>] snd_ctl_ioctl+0x440/0x6bc
[19779.817731] [<ffffffc00031865c>] do_vfs_ioctl+0x4a0/0x590
[19779.817738] [<ffffffc0003187c0>] SyS_ioctl+0x74/0xbc
```

```
static int msm_pcm_volume_ctl_put(struct snd_kcontrol *kcontrol,
         struct snd_ctl_elem_value *ucontrol)
{
  int rc = 0;
  struct snd_pcm_volume *vol = kcontrol->private_data;
  struct snd_pcm_substream *substream = vol->pcm->streams[0].substream;
  struct msm_pcm_loopback *prtd = substream->runtime->private_data; //---->substream is NULL
  int volume = ucontrol->value.integer.value[0];

  rc = pcm_loopback_set_volume(prtd, volume);
  return rc;
}
```

Case 3: Heap overflow(Qualcomm)

```
[20161128_12:52:10.974655]@3 Unable to handle kernel paging request at virtual address ffffffc045800000
[20161128_12:52:10.974678]@3 pgd = ffffffc09b374000
[20161128_12:52:10.974684]@3 [ffffffc045800000] *pgd=0000000000000000, *pud=0000000000000000
[20161128_12:52:10.974702]@3 Internal error: Oops: 96000006 [#1] PREEMPT SMP
[20161128_12:52:10.974710]@3 Modules linked in: wlan(O) crpl(PO)
[20161128_12:52:10.974732]@3 CPU: 3 PID: 3579 Comm: AudioOut_2 Tainted: P W O 3.18.20-perf+ #1
[20161128_12:52:10.974739]@3 Hardware name: Qualcomm Technologies, Inc. MSM 8996 v3 + PMI8996 MTP (DT)
[20161128_12:52:10.974746]@3 task: ffffffc1442a2280 ti: ffffffc0453bc000 task.ti: ffffffc0453bc000
[20161128_12:52:10.974764]@3 PC is at msm_dai_q6_set_channel_map+0x68/0xe4
[20161128_12:52:10.974774]@3 LR is at snd_soc_dai_set_channel_map+0x28/0x38
[20161128_12:52:10.974781]@3 pc : [<ffffffc000999d4c>] lr : [<ffffffc00094d724>] pstate: 30000145
[20161128_12:52:10.974786]@3 sp : ffffffc0453bf8b0
 ...
[20161128_12:52:10.976423]@3 Process AudioOut_2 (pid: 3579, stack limit = 0xffffffc0453bc060)
[20161128_12:52:10.976429]@3 Call trace:
[20161128_12:52:10.976441]@3 [<ffffffc000999d4c>] msm_dai_q6_set_channel_map+0x68/0xe4
[20161128_12:52:10.976449]@3 [<ffffffc00094d720>] snd_soc_dai_set_channel_map+0x24/0x38
[20161128_12:52:10.976458]@3 [<ffffffc0009f7f1c>] msm_snd_hw_params+0x21c/0x25c
[20161128_12:52:10.976467]@3 [<ffffffc00095d6a0>] soc_pcm_hw_params+0x26c/0x524
[20161128_12:52:10.976474]@3 [<ffffffc00095e8d4>] dpcm_be_dai_hw_params+0x10c/0x214
[20161128_12:52:10.976480]@3 [<ffffffc00095ea40>] dpcm_fe_dai_hw_params+0x64/0x104
[20161128_12:52:10.976489]@3 [<ffffffc000930be0>] snd_pcm_hw_params+0xb0/0x304
[20161128_12:52:10.976495]@3 [<ffffffc000930e94>] snd_pcm_ioctl_hw_params_compat+0x60/0x12c
[20161128_12:52:10.976502]@3 [<ffffffc000933980>] snd_pcm_ioctl_compat+0x258/0x84c
[20161128_12:52:10.976513]@3 [<ffffffc0001d1f6c>] compat_SyS_ioctl+0x10c/0x1210
[20161128_12:52:10.976520]@3 Code: 1400000e 6b00007f 540000c9 8b0000e1 (b8607882)
[20161128_12:52:10.976528]@3 ---[ end trace 719b76a3dde9febe ]---
[20161128_12:52:10.982661]@3 Kernel panic - not syncing: Fatal exception
[20161128_12:52:10.982678]@0 CPU0: stopping
```

```
static int msm_dai_q6_set_channel_map(struct snd_soc_dai *dai,
       unsigned int tx_num, unsigned int *tx_slot,
       unsigned int rx_num, unsigned int *rx_slot)
{
  ...
  for (i = 0; i < rx_num; i++) {
        dai_data->port_config.slim_sch.shared_ch_mapping[i] =
            rx_slot[i];
        pr_debug("%s: find number of channels[%d] ch[%d]\n",
              __func__, i, rx_slot[i]);
    }
    ...

  for (i = 0; i < tx_num; i++) {
        dai_data->port_config.slim_sch.shared_ch_mapping[i] =
            tx_slot[i];
        pr_debug("%s: find number of channels[%d] ch[%d]\n",
             __func__, i, tx_slot[i]);
    }
  ...
```

Case 4: Type Confusion(Qualcomm)

```
[20161202_18:17:05.425759]@3 Unable to handle kernel paging request at virtual address 400000028
[20161202_18:17:05.425773]@3 pgd = ffffffc14ffe5000
[20161202_18:17:05.425781]@3 [400000028] *pgd=0000000000000000, *pud=0000000000000000
[20161202_18:17:05.425804]@3 Internal error: Oops: 96000005 [#1] PREEMPT SMP
[20161202_18:17:05.425814]@3 Modules linked in: wlan(O) crpl(PO)
[20161202_18:17:05.425842]@3 CPU: 3 PID: 7892 Comm: fuzz Tainted: P        W  O   3.18.20-perf+ #1
[20161202_18:17:05.425850]@3 Hardware name: Qualcomm Technologies, Inc. MSM 8996 v3 + PMI8996 MTP (DT)
[20161202_18:17:05.425860]@3 task: ffffffc10c571700 ti: ffffffc0ed080000 task.ti: ffffffc0ed080000
[20161202_18:17:05.425877]@3 PC is at mutex_optimistic_spin+0x4c/0x1a8
[20161202_18:17:05.425887]@3 LR is at mutex_optimistic_spin+0x40/0x1a8
[20161202_18:17:05.425895]@3 pc : [<ffffffc0000de144>] lr : [<ffffffc0000de138>] pstate: 80000145
[20161202_18:17:05.425903]@3 sp : ffffffc0ed083ba0
 ...
[20161202_18:17:05.428178]@3 Process fuzz (pid: 7892, stack limit = 0xffffffc0ed080060)
[20161202_18:17:05.428185]@3 Call trace:
[20161202_18:17:05.428195]@3 [<ffffffc0000de144>] mutex_optimistic_spin+0x4c/0x1a8
[20161202_18:17:05.428211]@3 [<ffffffc000bd6a14>] __mutex_lock_slowpath+0x38/0x15c
[20161202_18:17:05.428220]@3 [<ffffffc000bd6b60>] mutex_lock+0x28/0x48
[20161202_18:17:05.428234]@3 [<ffffffc0009f8498>] msm8996_set_spk+0x74/0x10c
[20161202_18:17:05.428249]@3 [<ffffffc000925b30>] snd_ctl_elem_write+0xd0/0x154
[20161202_18:17:05.428259]@3 [<ffffffc000927be4>] snd_ctl_ioctl+0x3d0/0x640
[20161202_18:17:05.428271]@3 [<ffffffc00019d8e8>] do_vfs_ioctl+0x490/0x570
[20161202_18:17:05.428279]@3 [<ffffffc00019da24>] SyS_ioctl+0x5c/0x88
[20161202_18:17:05.428289]@3 Code: 94003f00 52800035 f9400e60 b4000040 (b9402815)
[20161202_18:17:05.428395]@3 ---[ end trace 7c90b26ac5fdf8ca ]---
[20161202_18:17:05.436459]@3 Kernel panic - not syncing: Fatal exception
[20161202_18:17:05.436481]@1 CPU1: stopping
```

```
static int msm8996_set_spk(struct snd_kcontrol *kcontrol,
                           struct snd_ctl_elem_value *ucontrol)
{
-       struct snd_soc_codec *codec = snd_kcontrol_chip(kcontrol);
+       struct snd_soc_codec *codec = snd_soc_kcontrol_codec(kcontrol);

        pr_debug("%s() ucontrol->value.integer.value[0] = %ld\n",
                __func__, ucontrol->value.integer.value[0]);
```

此处省略N洞…

(Skip N bugs here …)

What about the get()?

Let’s audit the code!

Case 5: Uninitialized stack variable leakage (Qualcomm)

```
static int msm_qti_pp_get_channel_map_mixer(struct snd_kcontrol *kcontrol,
                                            struct snd_ctl_elem_value *ucontrol)
{
        char channel_map[PCM_FORMAT_MAX_NUM_CHANNEL];
        int i;

        adm_get_multi_ch_map(channel_map, ADM_PATH_PLAYBACK);
        for (i = 0; i < PCM_FORMAT_MAX_NUM_CHANNEL; i++)
                ucontrol->value.integer.value[i] = (unsigned) channel_map[i];
        return 0;
}
```

Case 6: Information disclosure(Qualcomm)

```
int msm_dolby_dap_param_to_get_control_get(struct snd_kcontrol *kcontrol,
                  struct snd_ctl_elem_value *ucontrol)
{
...
params_value = kzalloc(params_length, GFP_KERNEL);
...
update_params_value = (int *)params_value;
ucontrol->value.integer.value[0] = dolby_dap_params_get.device_id;
ucontrol->value.integer.value[1] = dolby_dap_params_get.param_id;
ucontrol->value.integer.value[2] = dolby_dap_params_get.offset;
ucontrol->value.integer.value[3] = dolby_dap_params_get.length;

pr_debug("%s: FROM DSP value[0] 0x%x value[1] %d value[2] 0x%x\n",
        __func__, update_params_value[0],
        update_params_value[1], update_params_value[2]);
for (i = 0; i < dolby_dap_params_get.length; i++) {
    ucontrol->value.integer.value[DOLBY_PARAM_PAYLOAD_SIZE+i] =
        update_params_value[i];
    pr_debug("value[%d]:%d\n", i, update_params_value[i]);
}
```

Case 7: Integer/Heap overflow(Qualcomm)

```
int msm_dolby_dap_param_visualizer_control_get(struct snd_kcontrol *kcontrol,
                struct snd_ctl_elem_value *ucontrol)
{
...
offset = 0;
params_length = length * sizeof(uint32_t);
rc = adm_get_params(port_id, copp_idx, DOLBY_BUNDLE_MODULE_ID,
            DOLBY_PARAM_ID_VCBG,
            params_length + param_payload_len,
            visualizer_data + offset);
    ...
 }
```

```
int adm_get_params_v2(int port_id, int copp_idx, uint32_t module_id,
          uint32_t param_id, uint32_t params_length,
          char *params, uint32_t client_id)
{
...
sz = sizeof(struct adm_cmd_get_pp_params_v5) + params_length;
adm_params = kzalloc(sz, GFP_KERNEL);
if (!adm_params) {
    pr_err("%s: adm params memory alloc failed", __func__);
    return -ENOMEM;
}

memcpy(((u8 *)adm_params + sizeof(struct adm_cmd_get_pp_params_v5)),
    params, params_length);
```

What about other vendors?

Case 8: Stack overflow(ALSA)

```
[  115.431191] Kernel panic - not syncing: stack-protector: Kernel stack is corrupted in: ffffffc000a75f28
[  115.431191]
[  115.431203] CPU: 1 PID: 5066 Comm: AudioIn_26 Tainted: G        W    3.10.73-gc5a17ac-dirty #45
[  115.431209] Call trace:
[  115.431225] [<ffffffc000208530>] dump_backtrace+0x0/0x278
[  115.431234] [<ffffffc0002087b8>] show_stack+0x10/0x1c
[  115.431244] [<ffffffc000ce075c>] dump_stack+0x1c/0x28
[  115.431252] [<ffffffc000cdf4bc>] panic+0x160/0x314
[  115.431262] [<ffffffc0002208f4>] __stack_chk_fail+0x14/0x18
[  115.431272] [<ffffffc000a75f24>] dpcm_be_dai_prepare_async+0x1f4/0x214
[  115.431279] [<ffffffc000a7606c>] dpcm_fe_dai_prepare+0x128/0x208
[  115.431288] [<ffffffc000a45fa8>] snd_pcm_do_prepare+0x18/0x3c
[  115.431296] [<ffffffc000a45ae8>] snd_pcm_action_single+0x40/0x8c
[  115.431304] [<ffffffc000a479d4>] snd_pcm_common_ioctl1+0x57c/0xed4
[  115.431312] [<ffffffc000a4897c>] snd_pcm_capture_ioctl1+0x2dc/0x30c
[  115.431320] [<ffffffc000a48c44>] snd_pcm_ioctl_compat+0x24c/0x8d8
[  115.431329] [<ffffffc000350a34>] compat_sys_ioctl+0x120/0x126c
[  115.431339] CPU3: stopping
```

```
void dpcm_be_dai_prepare_async(struct snd_soc_pcm_runtime *fe, int stream,
                    struct async_domain *domain)
{
struct snd_soc_dpcm *dpcm;
struct snd_soc_dpcm *dpcm_async[DPCM_MAX_BE_USERS];
int i = 0, j;

list_for_each_entry(dpcm, &fe->dpcm[stream].be_clients, list_be) {
    ...
    } else {
        dpcm_async[i++] = dpcm;
    }
}
```

Case 9: Buffer overflow(Nvidia)

```
static int tegra210_adsp_set_param(struct snd_kcontrol *kcontrol,
            struct snd_ctl_elem_value *ucontrol)
{
            ...
            case SNDRV_CTL_ELEM_TYPE_INTEGER:
            {
                        int32_t num_params, i;
                        /* check number of params to pass */
                        num_params = (int32_t)ucontrol->value.integer.value[1];
                        if (num_params < 1) {
                                    dev_warn(adsp->dev, "No params to pass to the plugin\n");
                                    return 0;
                        }
                        apm_msg.msg.fx_set_param_params.params[0] =
                                    (sizeof(nvfx_call_params_t) +
                                    num_params * sizeof(int32_t));

                        /* initialize the method */
                        apm_msg.msg.fx_set_param_params.params[1] =
                                    (int32_t)ucontrol->value.integer.value[0];

                        /* copy parameters */
                        for (i = 0; i < num_params; i++)
                                    apm_msg.msg.fx_set_param_params.params[i + 2] =
                                                (int32_t)ucontrol->value.integer.value[i + 2];
```

Case 10: Buffer overflow(Nvidia)

```
static int tegra210_adsp_set_param(struct snd_kcontrol *kcontrol,
            struct snd_ctl_elem_value *ucontrol)
{
            ...
            case SNDRV_CTL_ELEM_TYPE_BYTES:
            {
                        nvfx_call_params_t *call_params =
                                    (nvfx_call_params_t *)ucontrol->value.bytes.data;

                        /* copy parameters */
                        memcpy(&apm_msg.msg.fx_set_param_params.params,
                                    call_params, call_params->size);
            }
```

Case 11: Out-of-bounds access(Nvidia)

```
static int tegra_vcm30t124_wm8731_put_rate(struct snd_kcontrol *kcontrol,
            struct snd_ctl_elem_value *ucontrol)
{
            struct snd_soc_card *card = snd_kcontrol_chip(kcontrol);
            struct tegra_vcm30t124 *machine = snd_soc_card_get_drvdata(card);
            unsigned int idx = tegra_vcm30t124_get_dai_link_idx("wm-playback");
            struct snd_soc_pcm_stream *dai_params =
                        (struct snd_soc_pcm_stream *)card->dai_link[idx].params;

            /* set the rate control flag */
            machine->wm_rate_via_kcontrol = ucontrol->value.integer.value[0];

            /* update the dai params rate */
            dai_params->rate_min =
                        tegra_vcm30t124_srate_values[machine->wm_rate_via_kcontrol];

            return 0;
}
```

Case 12: Out-of-bounds access(Nvidia)

```
static int tegra210_adsp_mux_put(struct snd_kcontrol *kcontrol,
        struct snd_ctl_elem_value *ucontrol)
{
        ...
        uint32_t val = ucontrol->value.enumerated.item[0];
```

```
static int soc_dapm_mux_update_power(struct snd_soc_card *card,
                                     struct snd_kcontrol *kcontrol, int mux, struct
soc_enum *e)
{
        ...
        /* we now need to match the string in the enum to the path */
        if (!(strcmp(path->name, e->texts[mux])))
```

Case 13: Out-of-bounds access(NXP,Oneplus)

```
[20161202_17:53:34.411743]@3 Unable to handle kernel paging request at virtual address ffffffad739d73d0
[20161202_17:53:34.411752]@3 pgd = ffffffc0a0bd8000
[20161202_17:53:34.411758]@3 [ffffffad739d73d0] *pgd=0000000000000000, *pud=0000000000000000
[20161202_17:53:34.411775]@3 Internal error: Oops: 96000005 [#1] PREEMPT SMP
[20161202_17:53:34.411782]@3 Modules linked in: wlan(O) crpl(PO)
[20161202_17:53:34.411803]@3 CPU: 3 PID: 5655 Comm: fuzz Tainted: P W O 3.18.20-perf+ #1
[20161202_17:53:34.411809]@3 Hardware name: Qualcomm Technologies, Inc. MSM 8996 v3 + PMI8996 MTP (DT)
[20161202_17:53:34.411818]@3 task: ffffffc147702280 ti: ffffffc1085cc000 task.ti: ffffffc1085cc000
[20161202_17:53:34.411834]@3 PC is at tfa98xx_set_profile_ctl+0x94/0x120
[20161202_17:53:34.411842]@3 LR is at tfa98xx_set_profile_ctl+0x58/0x120
[20161202_17:53:34.411848]@3 pc : [<ffffffc000994228>] lr : [<ffffffc0009941ec>] pstate: 80000145
[20161202_17:53:34.411854]@3 sp : ffffffc1085cfc70
[20161202_17:53:34.411860]@3 x29: ffffffc1085cfc70 x28: ffffffc1085cc000
...
[20161202_17:53:34.413559]@3 Process fuzz (pid: 5655, stack limit = 0xffffffc1085cc060)
[20161202_17:53:34.413565]@3 Call trace:
[20161202_17:53:34.413577]@3 [<ffffffc000994228>] tfa98xx_set_profile_ctl+0x94/0x120
[20161202_17:53:34.413588]@3 [<ffffffc000925b30>] snd_ctl_elem_write+0xd0/0x154
[20161202_17:53:34.413595]@3 [<ffffffc000927be4>] snd_ctl_ioctl+0x3d0/0x640
[20161202_17:53:34.413604]@3 [<ffffffc00019d8e8>] do_vfs_ioctl+0x490/0x570
[20161202_17:53:34.413610]@3 [<ffffffc00019da24>] SyS_ioctl+0x5c/0x88
[20161202_17:53:34.413618]@3 Code: b9410261 91354800 9408e1e8 b940f661 (b9401282)
[20161202_17:53:34.413626]@3 ---[ end trace d4892afeb3074ed0 ]---
[20161202_17:53:34.420050]@3 Kernel panic - not syncing: Fatal exception
```

Ooooooooooops

Devices which have been tested:

- Nexus 6p
- Oneplus 3
- Pixel C
- Not tried others yet.

What is the problem?

- The struct “snd_ctl_elem_value ” in get()/put() is originally designed to get/write values from/to codec HW registers only
- It is a kernel buffer, but its data is coming from userspace
- It is often misused by kernel developers to do some complicated things, but without sanity checking
- This exposes a wide range of kernel code to userspace
- The ALSA made the design, the codec developer made the implementation. There are gaps.

So we have talked about the Control, what about the others?

Case 14 : HWDEP: Race Condition/UAF/DF (DTS)

```
[20161213_17:30:34.226463]@2 Unable to handle kernel paging request at virtual address ff0a47272c00836c
[20161213_17:30:34.226495]@2 pgd = ffffffc150161000
[20161213_17:30:34.226525]@2 [ff0a47272c00836c] *pgd=0000000000000000, *pud=0000000000000000
[20161213_17:30:34.226593]@2 Internal error: Oops: 96000004 [#1] PREEMPT SMP
[20161213_17:30:34.226623]@2 Modules linked in: wlan(O) crpl(PO)
[20161213_17:30:34.226700]@2 CPU: 2 PID: 3788 Comm: ndroid.systemui Tainted: P W O 3.18.20-perf+ #1
[20161213_17:30:34.226729]@2 Hardware name: Qualcomm Technologies, Inc. MSM 8996 v3 + PMI8996 MTP (DT)
[20161213_17:30:34.226761]@2 task: ffffffc09f80c500 ti: ffffffc145e30000 task.ti: ffffffc145e30000
[20161213_17:30:34.226797]@2 PC is at kmem_cache_alloc_trace+0x94/0x1bc
[20161213_17:30:34.226830]@2 LR is at kmem_cache_alloc_trace+0x60/0x1bc
[20161213_17:30:34.226861]@2 pc : [<ffffffc000187344>] lr : [<ffffffc0001873
[20161213_17:30:34.226888]@2 sp : ffffffc145e33b30
...
20161213_17:30:34.234443]@2 Process ndroid.systemui (pid: 3788, stack limit
[20161213_17:30:34.234470]@2 Call trace:
[20161213_17:30:34.234512]@2 [<ffffffc000187344>] kmem_cache_alloc_trace+0x9
[20161213_17:30:34.234550]@2 [<ffffffc00017b4bc>] alloc_vmap_area.isra.33+0x
[20161213_17:30:34.234583]@2 [<ffffffc00017b820>] __get_vm_area_node.isra.34
[20161213_17:30:34.234618]@2 [<ffffffc00017c354>] __vmalloc_node_range+0x68/
[20161213_17:30:34.234650]@2 [<ffffffc00017c514>] __vmalloc_node+0x38/0x4c
[20161213_17:30:34.234682]@2 [<ffffffc00017c548>] vmalloc+0x20/0x2c
[20161213_17:30:34.234719]@2 [<ffffffc0002ac3ec>] write_pmsg+0x58/0x120
```

```
case DTS_EAGLE_IOCTL_SET_LICENSE: {
        ...
    if (target[1] == 0) {
        ...
        kfree(_sec_blob[target[0]]);
        _sec_blob[target[0]] = NULL;
        break;
    }
    ...
    sec_blob[target[0]] = kzalloc(target[1] + 4, GFP_KERNEL);
    if (!_sec_blob[target[0]]) {
        ...
        return -ENOMEM;
    }
    ((u32 *)_sec_blob[target[0]])[0] = target[1];
```

Case 15: Playback/capture:Use-after-Free (Qualcomm)

```
[   77.687642] Unable to handle kernel paging request at virtual address 202c796c6e4f207c
[   77.687653] pgd = ffffffc081dd4000
[   77.687666] [202c796c6e4f207c] *pgd=0000000000000000
[   77.687691] Internal error: Oops: 96000004 [#1] PREEMPT SMP
[   77.687713] CPU: 4 PID: 4616 Comm: fuzz Tainted: G        W    3.10.73-gc5a17ac-dirty #32
[   77.687727] task: ffffffc054245600 ti: ffffffc08b860000 task.ti: ffffffc08b860000
[   77.687753] PC is at __q6asm_set_volume+0x58/0x290
[   77.687769] LR is at __q6asm_set_volume+0x58/0x290
[   77.687782] pc : [<ffffffc000ad9990>] lr : [<ffffffc000ad9990>] pstate: 60000145
[   77.687798] sp : ffffffc08b863b80
...
[   77.688155] Process fuzz (pid: 4616, stack limit = 0xffffffc08b860058)
[   77.688166] Call trace:
[   77.688182] [<ffffffc000ad9990>] __q6asm_set_volume+0x58/0x290
[   77.688200] [<ffffffc000adff48>] q6asm_set_volume+0xc/0x18
[   77.688218] [<ffffffc000aa4894>] msm_pcm_volume_ctl_put+0xd0/0x108
[   77.688236] [<ffffffc000a3c1e4>] snd_ctl_elem_write+0x110/0x1a0
[   77.688251] [<ffffffc000a3d094>] snd_ctl_ioctl+0x440/0x6bc
[   77.688269] [<ffffffc00031865c>] do_vfs_ioctl+0x4a0/0x590
[   77.688284] [<ffffffc0003187c0>] SyS_ioctl+0x74/0xbc
[   77.688301] Code: 14000008 d0003cc0 911e4c00 94081350 (f9401e60)
[   77.688443] ---[ end trace 3bd871b8b1294b32 ]---
[   77.726884] Kernel panic - not syncing: Fatal exception
[   77.726907] CPU2: stopping
```

```
static int msm_pcm_playback_close(struct snd_pcm_substream *substream)
{
struct snd_pcm_runtime *runtime = substream->runtime;
struct snd_soc_pcm_runtime *soc_prtd = substream->private_data;
struct msm_audio *prtd = runtime->private_data;
...

pr_debug("%s: cmd_pending 0x%lx\n", __func__, prtd->cmd_pending);

if (prtd->audio_client) {
    ...
}
msm_pcm_routing_dereg_phy_stream(soc_prtd->dai_link->be_id,
                        SNDRV_PCM_STREAM_PLAYBACK);
kfree(prtd);
return 0;
}
```

Case 16: Multiple OOBs in hweffects(Qualcomm)

```
<1>[12444.414381] Unable to handle kernel paging request at virtual address ffff
ffc0015c9c10
<1>[12444.414570] pgd = ffffffc058972000
<1>[12444.414670] [ffffffc0015c9c10] *pgd=0000000000000000
<0>[12444.415004] Internal error: Oops: 9600004e [#1] PREEMPT SMP
<4>[12444.415114] CPU: 2 PID: 22224 Comm: test Not tainted 3.10.73-g673810b #1
<4>[12444.415303] task: ffffffc018bc0000 ti: ffffffc034a4c000 task.ti: ffffffc03
4a4c000
<4>[12444.415505] PC is at msm_audio_effects_popless_eq_handler+0x2c4/0x55c
<4>[12444.415607] LR is at msm_audio_effects_popless_eq_handler+0xa0/0x55c
<4>[12444.415795] pc : [<ffffffc000c3c600>] lr : [<ffffffc000c3c3dc>] pstate: a0
000145
<4>[12444.415896] sp : ffffffc034a4f900
<4>[12444.416079] x29: ffffffc034a4f900 x28: ffffffc0028dc8fc
<4>[12444.416368] x27: ffffffc034a4fa08 x26: 0000000000000000
<4>[12444.416740] x25: ffffffc034a4fa68 x24: 0000000000000000
<4>[12444.417029] x23: ffffffc047794000 x22: ffffffc001099a08
<4>[12444.417319] x21: 0000000000000000 x20: ffffffc0018d4000
<4>[12444.417693] x19: ffffffc047794000 x18: 0000000000000001
<4>[12444.417985] x17: 0000007f83bc9264 x16: ffffffc000308800
<4>[12444.418358] x15: 0000000000000001 x14: 0000000000000001
<4>[12444.418645] x13: 0000000000000000 x12: 0000000000004001
<4>[12444.419016] x11: 0000000000004000 x10: 0000007f83c3b70c
<4>[12444.419305] x9 : 0000000000000014 x8 : ffffffc047795000
<4>[12444.419679] x7 : 0000000000000001 x6 : 000000000000003f
<4>[12444.419969] x5 : 0000000000000040 x4 : fffffffffff0bdc0
<4>[12444.420342] x3 : 0000000000000001 x2 : 00000000ffffffff
<4>[12444.420631] x1 : 0000000000000001 x0 : ffffffc0015c9bfc
<4>[12444.421002]
<0>[12444.421104] Process test (pid: 22224, stack limit = 0xffffffc034a4c058)
<4>[12444.421207] Call trace:
<4>[12444.421403] [<ffffffc000c3c600>] msm_audio_effects_popless_eq_handler+0x2c
4/0x55c
<4>[12444.421513] [<ffffffc0005e723c>] audio_effects_set_pp_param+0x1bc/0x32c
<4>[12444.421701] [<ffffffc0005e81e0>] audio_effects_ioctl+0x354/0x3dc
<4>[12444.421810] [<ffffffc00030872c>] do_vfs_ioctl+0x4a8/0x57c
<4>[12444.421998] [<ffffffc000308854>] SyS_ioctl+0x5c/0x88
```

Ooooooooooops

此处省略N洞…

(Skip N bugs here …)

Thinking in Exploitation

The AID_AUDIO processes

```
angler:/data/local/tmp # ./get_process_from_gid 1005
group id:1005
radio     607   1     16076  572   poll_sched 70b1824ca4 S /vendor/bin/qmuxd

audioserver 617   1     65560  3572  binder_thr 00ee09c66c S /system/bin/audioserver

cameraserver 618   1     17736  780   binder_thr 00ed17466c S /system/bin/cameraserver

media     625   1     44312  1904  binder_thr 00eaacf66c S /system/bin/mediaserver

radio     627   1     57072  2508  hrtimer_na 734c37855c S /system/bin/rild

system    982   614   2444056 215856 SyS_epoll_ 72355e8b6c S system_server

u0_a19    25769 614   1712576 43464 SyS_epoll_ 72355e8b6c S com.google.android.gms.feedback

u0_a19    29776 614   1566588 39720 SyS_epoll_ 72355e8b6c S com.google.process.gapps

u0_a19    29797 614   1812028 113388 SyS_epoll_ 72355e8b6c S com.google.android.gms.persistent

u0_a41    29956 614   2301896 39844 SyS_epoll_ 72355e8b6c S com.google.android.googlequicksearchbox:interactor

u0_a19    30197 614   1915148 102612 SyS_epoll_ 72355e8b6c S com.google.android.gms

u0_a41    30729 614   2353328 87888 SyS_epoll_ 72355e8b6c S com.google.android.googlequicksearchbox
```

Vulnerability	CVE	Severity	It affects the Google device?
Remote code execution through surfaceflinger	CVE-2017-0405	Critical	Yes
Remote code execution through mediaserver	CVE-2017-0466, CVE-2017-0467, CVE-2017-0468, CVE-2017-0469, CVE-2017-0470, CVE-2017-0471, CVE-2017-0472, CVE-2017-0473, CVE- 2017-0474	Critical	Yes
Privilege escalation through mediaserver	CVE-2017-0415	Tall	Yes
Privilege escalation through audioserver	CVE-2017-0416, CVE-2017-0417, CVE-2017-0418, CVE-2017-0419	Tall	Yes
Privilege escalation through audioserver	CVE-2017-0479, CVE-2017-0480	Tall	Yes
Privilege escalation through audioserver	CVE-2017-0384, CVE-2017-0385	Tall	Yes
Privilege escalation through System Server	CVE-2016-6707	Tall	Yes
Privilege escalation through camera service	CVE-2016-3915, CVE-2016-3916	Tall	Yes
Privilege escalation through system_server	CVE-2016-2412	Tall	

Proposal 1: Chaining from “system_server”

- **Gal Beniamini** of Project Zero successfully exploited CVE-2016-6707 to gain “system_server” privilege:

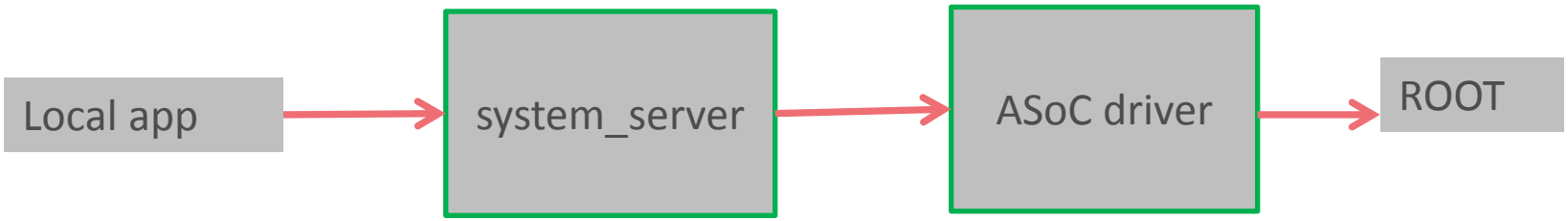
Project Zero

News and updates from the Project Zero team at Google

Thursday, December 1, 2016

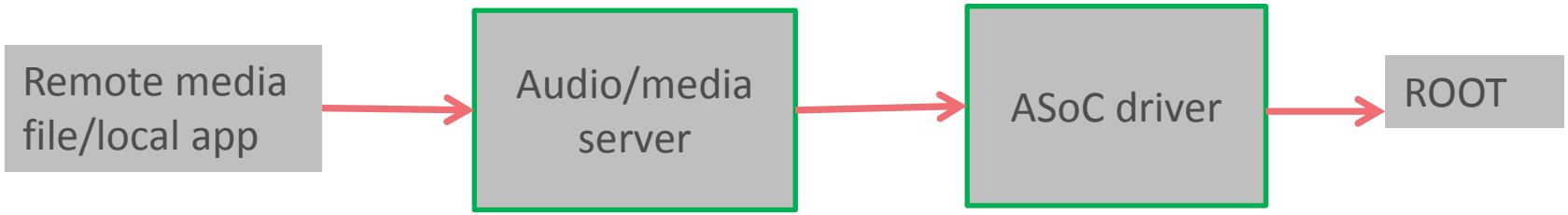
BitUnmap: Attacking Android Ashmem

Posted by Gal Beniamini, Project Zero



Proposal 2: Chaining from “audio/media server”

Hanan Be'er successfully exploit libStagefright to gain “mediaserver” privilege:



Challenge

- Google mitigated media server bugs(since Android M), exploiting through this way will be a bit tough

- You need to hunt additional bugs in Android framework

Conclusions

- Kernel developers use get/put interfaces to configure ASoC codecs but often miss sanity checking
- This opens up an attack surface in kernel
- Local/remote root is theoretically possible by chaining bugs from system/media/audio/camera servers
- ASoC developers should be careful when handling get/put interfaces, keeping userspace values are simply used to write/read codec registers only.

Questions?