Crash in libdvm (pthread_create) for arbitrary thread creations

+1 3 Hotlists (2) Mark as Duplicate

Comments (5) Dependencies Duplicates (0) Blocking (0) Resources (1)

Fixed Bug P4 + Add Hotlist [AOSP] FutureRelease

STATUS UPDATE No update yet. Edit

DESCRIPTION ha...@gmail.com created issue #1

The crash log I receive is (LogCat):

```
11-16 10:33:46.051: W/libc(9856): pthread_create failed: clone failed: Try again
11-16 10:33:46.051: A/libc(9856): Fatal signal 11 (SIGSEGV) at 0x7620af00 (code=1), thread 9856 (harrys.laptimer)
11-16 10:33:46.151: I/DEBUG(171): *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** ***
11-16 10:33:46.151: I/DEBUG(171): Build fingerprint: 'google/occam/mako:4.3/JWR66Y/776638:user/release-keys'
11-16 10:33:46.151: I/DEBUG(171): Revision: '11'
11-16 10:33:46.151: I/DEBUG(171): pid: 9856, tid: 9856, name: harrys.laptimer  >>> com.harrys.laptimer <<<
11-16 10:33:46.151: I/DEBUG(171): signal 11 (SIGSEGV), code 1 (SEGV_MAPERR), fault addr 7620af00
11-16 10:33:46.281: I/DEBUG(171):     r0 7620af00  r1 00000005  r2 beda73fc  r3 4008874c
11-16 10:33:46.281: I/DEBUG(171):     r4 7620af00  r5 00100000  r6 0000000b  r7 73dd652d
11-16 10:33:46.281: I/DEBUG(171):     r8 745e2614  r9 7610b000  sl 7620af00  fp 7492fc88
11-16 10:33:46.281: I/DEBUG(171):     ip 40088ddc  sp beda73e8  lr 40057c1c  pc 40059010  cpsr 20030010
11-16 10:33:46.281: I/DEBUG(171):     d0  6961676120797254  d1  656c696166206574
11-16 10:33:46.281: I/DEBUG(171):     d2  656e6f6c63203a64  d3  3a64656c69616620
11-16 10:33:46.281: I/DEBUG(171):     d4  0000000000000047  d5  0000000100000047
11-16 10:33:46.281: I/DEBUG(171):     d6  00000001000030c3  d7  0000002000000001
11-16 10:33:46.281: I/DEBUG(171):     d8  4424000000000290  d9  4021a000441b99c0
11-16 10:33:46.281: I/DEBUG(171):     d10 4021a00000000000  d11 0000000000000000
11-16 10:33:46.281: I/DEBUG(171):     d12 0000000000000000  d13 0000000000000000
11-16 10:33:46.281: I/DEBUG(171):     d14 0000000000000000  d15 0000000000000000
11-16 10:33:46.281: I/DEBUG(171):     d16 4042000000000000  d17 4042400000000000
11-16 10:33:46.281: I/DEBUG(171):     d18 3ff8000000000000  d19 0006000500040003
11-16 10:33:46.281: I/DEBUG(171):     d20 0052005000520051  d21 0056005500540053
11-16 10:33:46.281: I/DEBUG(171):     d22 002e002d002c002b  d23 0030002f002e002c
11-16 10:33:46.281: I/DEBUG(171):     d24 0008000700060004  d25 000a0008000a0009
11-16 10:33:46.281: I/DEBUG(171):     d26 0000000000000000  d27 0000000000000000
11-16 10:33:46.281: I/DEBUG(171):     d28 0048004700460044  d29 004a0048004a0049
11-16 10:33:46.281: I/DEBUG(171):     d30 000a000a000a000a  d31 0000000000000000
11-16 10:33:46.281: I/DEBUG(171):     scr 60000013
11-16 10:33:46.291: I/DEBUG(171): backtrace:
11-16 10:33:46.291: I/DEBUG(171):     #00  pc 0000e010  /system/lib/libc.so
11-16 10:33:46.291: I/DEBUG(171):     #01  pc 0000cc18  /system/lib/libc.so (pthread_create+276)
11-16 10:33:46.291: I/DEBUG(171):     #02  pc 000058d7  /system/lib/egl/eglsubAndroid.so (updater_create_surface_state+126)
11-16 10:33:46.291: I/DEBUG(171):     #03  pc 000047e3  /system/lib/egl/eglsubAndroid.so
11-16 10:33:46.291: I/DEBUG(171):     #04  pc 0000c240  /system/lib/egl/libEGL_adreno200.so (qeglDrvAPI_eglCreateWindowSurface+836)
11-16 10:33:46.291: I/DEBUG(171):     #05  pc 0000647c  /system/lib/egl/libEGL_adreno200.so (eglCreateWindowSurface+16)
11-16 10:33:46.291: I/DEBUG(171):     #06  pc 0000eb3d  /system/lib/libEGL.so (eglCreateWindowSurface+200)
11-16 10:33:46.291: I/DEBUG(171):     #07  pc 0004d929  /system/lib/libandroid_runtime.so
11-16 10:33:46.291: I/DEBUG(171):     #08  pc 0001dc4c  /system/lib/libdvm.so (dvmPlatformInvoke+112)
11-16 10:33:46.291: I/DEBUG(171):     #09  pc 0004decf  /system/lib/libdvm.so (dvmCallJNIMethod(unsigned int const*, JValue*, Method const*, Thread*)+398)
11-16 10:33:46.291: I/DEBUG(171):     #10  pc 00027060  /system/lib/libdvm.so
11-16 10:33:46.291: I/DEBUG(171):     #11  pc 0002b5ec  /system/lib/libdvm.so (dvmInterpret(Thread*, Method const*, JValue*)+184)
11-16 10:33:46.291: I/DEBUG(171):     #12  pc 000601df  /system/lib/libdvm.so (dvmInvokeMethod(Object*, Method const*, ArrayObject*, ArrayObject*, ClassObject*, bool)+350)
11-16 10:33:46.291: I/DEBUG(171):     #13  pc 00067ddf  /system/lib/libdvm.so
11-16 10:33:46.291: I/DEBUG(171):     #14  pc 00027060  /system/lib/libdvm.so
11-16 10:33:46.291: I/DEBUG(171):     #15  pc 0002b5ec  /system/lib/libdvm.so (dvmInterpret(Thread*, Method const*, JValue*)+184)
11-16 10:33:46.291: I/DEBUG(171):     #16  pc 0005ff21  /system/lib/libdvm.so (dvmCallMethodV(Thread*, Method const*, Object*, bool, JValue*, std::__va_list)+292)
11-16 10:33:46.291: I/DEBUG(171):     #17  pc 00049b67  /system/lib/libdvm.so
11-16 10:33:46.291: I/DEBUG(171):     #18  pc 0004b697  /system/lib/libandroid_runtime.so
11-16 10:33:46.291: I/DEBUG(171):     #19  pc 0004c327  /system/lib/libandroid_runtime.so (android::AndroidRuntime::start(char const*, char const*)+378)
11-16 10:33:46.291: I/DEBUG(171):     #20  pc 0000105b  /system/bin/app_process
11-16 10:33:46.291: I/DEBUG(171):     #21  pc 0000db4f  /system/lib/libc.so (__libc_init+50)
11-16 10:33:46.291: I/DEBUG(171):     #22  pc 00000d7c  /system/bin/app_process
11-16 10:33:46.291: I/DEBUG(171): stack:
11-16 10:33:46.291: I/DEBUG(171):          beda73a8  7492fc88
11-16 10:33:46.291: I/DEBUG(171):          beda73ac  4005b61f  /system/lib/libc.so (dlmalloc+4282)
11-16 10:33:46.291: I/DEBUG(171):          beda73b0  73697048  /system/lib/egl/libEGL_adreno200.so
11-16 10:33:46.291: I/DEBUG(171):          beda73b4  beda73e8  [stack]
11-16 10:33:46.291: I/DEBUG(171):          beda73b8  736721d0
11-16 10:33:46.291: I/DEBUG(171):          beda73bc  00000006
11-16 10:33:46.291: I/DEBUG(171):          beda73c0  00003040
11-16 10:33:46.291: I/DEBUG(171):          beda73c4  00000240
11-16 10:33:46.291: I/DEBUG(171):          beda73c8  7492fc80
11-16 10:33:46.291: I/DEBUG(171):          beda73cc  3bb849b6
11-16 10:33:46.291: I/DEBUG(171):          beda73d0  0000000b
11-16 10:33:46.291: I/DEBUG(171):          beda73d4  00100000
11-16 10:33:46.291: I/DEBUG(171):          beda73d8  0000000b
11-16 10:33:46.291: I/DEBUG(171):          beda73dc  73dd652d  /system/lib/egl/eglsubAndroid.so
```

```
11-16 10:33:46.291: I/DEBUG(171):         beda73e0  df0027ad
11-16 10:33:46.291: I/DEBUG(171):         beda73e4  00000000
11-16 10:33:46.291: I/DEBUG(171):     #00  beda73e8  00000005
11-16 10:33:46.291: I/DEBUG(171):         beda73ec  beda73fc  [stack]
11-16 10:33:46.291: I/DEBUG(171):         beda73f0  0000000b
11-16 10:33:46.291: I/DEBUG(171):         beda73f4  00100000
11-16 10:33:46.291: I/DEBUG(171):         beda73f8  0000000b
11-16 10:33:46.291: I/DEBUG(171):         beda73fc  40057c1c  /system/lib/libc.so (pthread_create+280)
11-16 10:33:46.291: I/DEBUG(171):     #01  beda7400  ffffffff
11-16 10:33:46.291: I/DEBUG(171):         beda7404  00000000
11-16 10:33:46.291: I/DEBUG(171):         beda7408  4009429c  /system/lib/libc.so
11-16 10:33:46.291: I/DEBUG(171):         beda740c  00001000
11-16 10:33:46.291: I/DEBUG(171):         beda7410  00003084
11-16 10:33:46.291: I/DEBUG(171):         beda7414  745e25c0
11-16 10:33:46.291: I/DEBUG(171):         beda7418  00000000
11-16 10:33:46.291: I/DEBUG(171):         beda741c  73672208
11-16 10:33:46.291: I/DEBUG(171):         beda7420  beda74c0  [stack]
11-16 10:33:46.291: I/DEBUG(171):         beda7424  00000001
11-16 10:33:46.291: I/DEBUG(171):         beda7428  00003084
11-16 10:33:46.291: I/DEBUG(171):         beda742c  73672208
11-16 10:33:46.291: I/DEBUG(171):         beda7430  745e23b8
11-16 10:33:46.291: I/DEBUG(171):         beda7434  73dd68db  /system/lib/egl/eglsubAndroid.so (updater_create_surface_state+130)
11-16 10:33:46.291: I/DEBUG(171):     #02  beda7438  7492f8a0
11-16 10:33:46.291: I/DEBUG(171):         beda743c  73dd31a3  /system/lib/egl/eglsubAndroid.so
11-16 10:33:46.291: I/DEBUG(171):         beda7440  7492f8a0
11-16 10:33:46.291: I/DEBUG(171):         beda7444  745e24d0
11-16 10:33:46.291: I/DEBUG(171):         beda7448  7492f8a0
11-16 10:33:46.291: I/DEBUG(171):         beda744c  73dd57e7  /system/lib/egl/eglsubAndroid.so
```

My interpretation is: this is a crash within the Dalvik engine happening when a thread is created on native level (pthread_create). In this particular case, the thread created seems to be one used t
without touching it (with screen forced to display and not go to sleep). But this is just a sample... I had a AsyncTask using THREAD_POOL_EXECUTOR running in the app which created threads ov
The crash appeared randomly after creating thousands of threads, sometimes hundreds of thousands (certainly in a sequence). So what I want to say is "thread creation - both motivated natively

I have read that pthread_create crashing is usually due to heap corruption. As the app uses JNI heavily (about 50% of the code), this is certainly a candidate to trigger the crash... I used several to
only... Tested both Java and native heap usage too. No leaks here and the native heap is typically around 7 to 8 MB. Test device is a Google Nexus 4 (which should have by far more memory avai

So back to my main question: any know issues in Android about situation that make pthread_create crash in livdvm? Any misbehavior on app level that will trigger a crash like this?

In case more information is needed, please let me know.

- Harry

Links (1)

“ …ed by the graphics code. and the crash isn't in the VM either; it's in the C library. basically, on the error path from pthread_create we unmap the memory containing the pthread_mutex_t used to sync

COMMENTS

en...@google.com <en...@google.com> #2

Marked as fixed, reassigned to en...@google.com.

11-16 10:33:46.051: W/libc(9856): pthread_create failed: clone failed: Try again

"try again" basically means "you used up all the system's resources". you probably have too many threads.

this thread _isn't_ created by the VM. it's being created by the graphics code. and the crash isn't in the VM either; it's in the C library. basically, on the error path from pthread_create we unmap

ha...@gmail.com <ha...@gmail.com> #3

Thanks for the feedback, that actually helped a lot already. I didn't want to be too specific on VM or not VM, it was more on a differentiation between user code / app code and Android code..

I have added some tracing to watch the threads and indeed, they grow continuously... Besides a thread pool for a ASyncTask that allocates up to 5 threads (and keeps them), several system t
Async pool, 9 threads I have an idea of what they are good for (and staying there constantly from the beginning), I see 14 Binder threads alive...

I understand these threads are created for inter process communication but have no clue so far what exactly triggers this in the app and why they do not go away again. Any help on this is ap

The bug fix named fixes the "hard" crash, but will not cure the effect a thread needed cannot be created, right?

- Harry

ha...@gmail.com <ha...@gmail.com> #4

Running KitKat 4.4.2 now, I cannot reproduce the bug reported any more. As I'm not sure this is due to the named bug fix, or due to some other change in my code, I'd be interested to underst

```
12-13 11:29:55.066: I/System.out(19408): java.lang.ThreadGroup[name=main,maxPriority=10]
12-13 11:29:55.066: I/System.out(19408):     Thread[main,5,main]
12-13 11:29:55.066: I/System.out(19408):     Thread[Thread-5,5,main]
12-13 11:29:55.066: I/System.out(19408):     Thread[Binder_1,5,main]
12-13 11:29:55.066: I/System.out(19408):     Thread[Binder_2,5,main]
12-13 11:29:55.066: I/System.out(19408):     Thread[SoundPool,5,main]
```

```
12-13 11:29:55.076: I/System.out(19408):    Thread[SoundPoolThread,5,main]
12-13 11:29:55.076: I/System.out(19408):    Thread[AsyncTask #1,5,main]
12-13 11:29:55.076: I/System.out(19408):    Thread[AsyncTask #2,5,main]
12-13 11:29:55.076: I/System.out(19408):    Thread[java.lang.ProcessManager,5,main]
12-13 11:29:55.076: I/System.out(19408):    Thread[Binder_3,5,main]
12-13 11:29:55.076: I/System.out(19408):    Thread[background thread,5,main]
12-13 11:29:55.076: I/System.out(19408):    Thread[AsyncTask #3,5,main]
12-13 11:29:55.076: I/System.out(19408):    Thread[AsyncTask #4,5,main]
12-13 11:29:55.076: I/System.out(19408):    Thread[AsyncTask #5,5,main]
12-13 11:29:55.076: I/System.out(19408):    Thread[Binder_4,5,main]
12-13 11:29:55.076: I/System.out(19408):    Thread[hwuiTask1,5,main]
12-13 11:29:55.076: I/System.out(19408):    Thread[hwuiTask2,5,main]
12-13 11:29:55.076: I/System.out(19408):    Thread[Binder_5,5,main]
12-13 11:29:55.076: I/System.out(19408):    Thread[Binder_6,5,main]
12-13 11:29:55.076: I/System.out(19408):    Thread[Binder_7,5,main]
12-13 11:29:55.076: I/System.out(19408):    Thread[Binder_8,5,main]
12-13 11:29:55.076: I/System.out(19408):    Thread[Binder_9,5,main]
12-13 11:29:55.076: I/System.out(19408):    Thread[Binder_A,5,main]
12-13 11:29:55.076: I/System.out(19408):    Thread[Binder_B,5,main]
12-13 11:29:55.076: I/System.out(19408):    Thread[Binder_C,5,main]
12-13 11:29:55.076: I/System.out(19408):    Thread[Binder_D,5,main]
12-13 11:29:55.076: I/System.out(19408):    Thread[Binder_E,5,main]
12-13 11:29:55.086: I/System.out(19408):    Thread[Binder_F,5,main]
12-13 11:29:55.086: I/System.out(19408):    Thread[PerTaskExecutorThread,5,main]
12-13 11:29:55.086: I/System.out(19408):    Thread[Binder_10,5,main]
```

The limit in growth could be due to some pool limit for Binder threads? Any observation on this thread list? Is it normal to have a big number of Binder threads? And any hint on a work arounc

Thanx, Harry

---

en...@google.com <en...@google.com> #5

my bionic fix isn't in 4.2.2, so if you do manage to run out of resources and fail to create a thread you'll still see the SIGSEGV.

i don't know what else has changed, but i don't think 29 threads is unusual. i'm pretty sure system_server starts off with around 100 threads. (remember too that the limit you were hitting was