Lots of native crashes since August 2023 / Android 13+

Assigned Bug P3 + Add Hotlist [AOSP] assigned

STATUS UPDATE No update yet. Edit

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

Since around August 2023 we suddenly started to notice a huge number of native crashes (mostly segmentation faults) on Android 13 and Android 14 Beta devices of different manufacturers.

According to the affected users, these crashes occur on cold start of the app only. Some users stated, that the problem began after a recent Android system update.

The crashes also affect users who are still running an older version of the app that didn't cause any problems before.

The stack traces we see in the Google Play Console are vastly diverse but the crashes all happen within libart.so. Here's an example:

```
#00  pc 0x0000000000282564  /apex/com.android.art/lib64/libart.so (art::ClassLinker::FindClass(art::Thread*, char const*, art::Handle<art::mirror::ClassLoa
#01  pc 0x0000000000273e88  /apex/com.android.art/lib64/libart.so (art::ObjPtr<art::mirror::Class> art::ClassLinker::DoResolveType<art::ArtMethod*>(art::de
#02  pc 0x000000000031d890  /apex/com.android.art/lib64/libart.so (art::ResolveVerifyAndClinit(art::dex::TypeIndex, art::ArtMethod*, art::Thread*, bool, bo
#03  pc 0x000000000031d630  /apex/com.android.art/lib64/libart.so (NterpGetClass+84)
#04  pc 0x000000000058fd40  /apex/com.android.art/lib64/libart.so (nterp_get_class+48)
#05  pc 0x0000000000582250  /apex/com.android.art/lib64/libart.so (nterp_op_check_cast+80)
#06  pc 0x000000000024d56a  /data/app/~~3kRbduGP1iwQERWjOTn08Q==/ch.threema.app-MK65_59wCK_TSBiUo8RRMA==/oat/arm64/base.vdex (r3.run+22)
#07  pc 0x0000000000664a8c  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (java.util.concurrent.ThreadPoolExecutor.runWorker+796)
#08  pc 0x0000000000661bb0  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (java.util.concurrent.ThreadPoolExecutor$Worker.run+64)
#09  pc 0x0000000000058bad4  /apex/com.android.art/lib64/libart.so (nterp_helper+7636)
#10  pc 0x000000000024d50e  /data/app/~~3kRbduGP1iwQERWjOTn08Q==/ch.threema.app-MK65_59wCK_TSBiUo8RRMA==/oat/arm64/base.vdex (q3$a.run+14)
#11  pc 0x0000000000510598  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (java.lang.Thread.run+72)
#12  pc 0x00000000003605a4  /apex/com.android.art/lib64/libart.so (art_quick_invoke_stub+612)
#13  pc 0x000000000034b930  /apex/com.android.art/lib64/libart.so (art::ArtMethod::Invoke(art::Thread*, unsigned int*, unsigned int, art::JValue*, char con
#14  pc 0x00000000004f3e38  /apex/com.android.art/lib64/libart.so (art::Thread::CreateCallback(void*)+1888)
#15  pc 0x00000000000eb720  /apex/com.android.runtime/lib64/bionic/libc.so (__pthread_start(void*)+208)
#16  pc 0x000000000007e2d0  /apex/com.android.runtime/lib64/bionic/libc.so (__start_thread+64)
```

Unfortunately, our testers are not able to reproduce the problem. Since the problem occurred so suddenly, we suspect that a change in the framework might be responsible.

The problem seems to affect users of Google Play only and not those who install the APK manually. Note that we do *not* use Android App Bundle.

We'll be happy to provide more stack traces if needed.

✓ Mentioned issues (1) ✓ Links (5)

Mentioned issues (1)

P3 ART segfaults while performing CallVoidMethod after updating to August 2023 security patch "https://issuetracker.google.com/304071459"

Links (5)

"For steps to capture a bug report, please refer: https://developer.android.com/studio/debug/bug-report#bugreportdevice"
"Here's a bug report, albeit from a Samsung device. https://drive.google.com/drive/folders/1CGwl9WFbjSJ0mHuYN-zkq4UliRRHyems?usp=sharing"
"I'm the developer of Strato Emulator and lately I've also been receiving reports of native crashes, seemingly after the August 2023 patch. From the debugging I was able to perform, the crashes ca
"Here's the call site, which then goes into libart.so and segfaults: https://github.com/strato-emu/strato/blob/b0207ab6456499448e8b9cb552410864c866e15f/app/src/main/cpp/skyline/jvm.cpp#L1
"Here's the Java method that should be called: https://github.com/strato-emu/strato/blob/b0207ab6456499448e8b9cb552410864c866e15f/app/src/main/java/emu/skyline/EmulationActivity.kt#L6

COMMENTS

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

Assigned to ra...@google.com.

Thank you for reporting this issue. For us to further investigate this issue, please provide the following additional information:

Can you confirm if this issue is reproducible on Pixel/Nexus device?

Can you provide the bugreport if possible.

Android bug report (to be captured after reproducing the issue)
For steps to capture a bug report, please refer: https://developer.android.com/studio/debug/bug-report#bugreportdevice

Alternate method
Navigate to "Developer options", ensure "USB debugging" is enabled, then enable "Bug report shortcut". Capture bug report by holding the power button and selecting the "Take bug report" o

Note: Please upload the bug report and screenshot to google drive and share the folder to android-bugreport@google.com, then share the link here.

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

Yes, according to the thousands of stack traces on Google Play Console, the problem also affects all Pixel models running Android 13 or Android 14 beta.

Here's a bug report, albeit from a Samsung device. https://drive.google.com/drive/folders/1CGwl9WFbjSJ0mHuYN-zkq4UIiRRHyems?usp=sharing

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

Fhrttgeeff

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

We've shared this with our product and engineering teams and will continue to provide updates as more information becomes available.

ni...@gmail.com <ni...@gmail.com> #6

I'm the developer of Strato Emulator and lately I've also been receiving reports of native crashes, seemingly after the August 2023 patch. From the debugging I was able to perform, the cra
we're having.

Here's the call site, which then goes into `libart.so` and segfaults: https://github.com/strato-emu/strato/blob/b0207ab6456499448e8b9cb552410864c866e15f/app/src/main/cpp/skyline/

Here's the Java method that should be called: https://github.com/strato-emu/strato/blob/b0207ab6456499448e8b9cb552410864c866e15f/app/src/main/java/emu/skyline/EmulationActivi

The next obvious step in my case would be to try to reproduce the behaviour in a smaller test app, with a native method using the JNI to execute a Java method that takes a `jLongArray` as p

mb...@gmail.com <mb...@gmail.com> #7

#6 That's an interesting observation. We do indeed see `CallStaticVoidMethod` involving a `jobjectArray` parameter at the root of a lot of these stack traces. Here's a few more examples:

```
#00  pc 0x0000000000282564  /apex/com.android.art/lib64/libart.so (art::ClassLinker::FindClass(art::Thread*, char const*, art::Handle<art::mirror::Class
#01  pc 0x000000000032303c  /apex/com.android.art/lib64/libart.so (art::ResolveFieldWithAccessChecks(art::Thread*, art::ClassLinker*, unsigned short, ar
#02  pc 0x00000000003203c0  /apex/com.android.art/lib64/libart.so (NterpGetInstanceFieldOffset+112)
#03  pc 0x000000000058fbf0  /apex/com.android.art/lib64/libart.so (nterp_get_instance_field_offset+48)
#04  pc 0x00000000005894c4  /apex/com.android.art/lib64/libart.so (nterp_op_iget_slow_path+20)
#05  pc 0x000000000025a8d8  /data/app/~~p4pUvzr4n__obb3WCOiNfQ==/ch.threema.app-x8YAhtyxRb_T1M5KVtQBHQ==/oat/arm64/base.vdex (c80.run)
#06  pc 0x0000000000ab343c  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (android.os.Handler.dispatchMessage+76)
#07  pc 0x0000000000ab6f6c  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (android.os.Looper.loopOnce+1244)
#08  pc 0x0000000000ab69ac  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (android.os.Looper.loop+1196)
#09  pc 0x00000000007cbc60  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (android.app.ActivityThread.main+2608)
#10  pc 0x0000000000360880  /apex/com.android.art/lib64/libart.so (art_quick_invoke_static_stub+640)
#11  pc 0x000000000026a904  /apex/com.android.art/lib64/libart.so (_jobject* art::InvokeMethod<(art::PointerSize)8>(art::ScopedObjectAccessAlreadyRunnab
#12  pc 0x000000000026a5e8  /apex/com.android.art/lib64/libart.so (art::Method_invoke(_JNIEnv*, _jobject*, _jobject*, _jobjectArray*) (.__uniq.165753521
#13  pc 0x00000000003636a8  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (art_jni_trampoline+120)
#14  pc 0x0000000000d5e158  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (com.android.internal.os.RuntimeInit$MethodAndArgsCaller.run
#15  pc 0x0000000000d6aac0  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (com.android.internal.os.ZygoteInit.main+4400)
#16  pc 0x0000000000360880  /apex/com.android.art/lib64/libart.so (art_quick_invoke_static_stub+640)
#17  pc 0x00000000004944cc  /apex/com.android.art/lib64/libart.so (art::JValue art::InvokeWithVarArgs<_jmethodID*>(art::ScopedObjectAccessAlreadyRunnabl
#18  pc 0x0000000000553530  /apex/com.android.art/lib64/libart.so (art::JNI<true>::CallStaticVoidMethodV(_JNIEnv*, _jclass*, _jmethodID*, std::__va_list
#19  pc 0x00000000000c0c04  /system/lib64/libandroid_runtime.so (_JNIEnv::CallStaticVoidMethod(_jclass*, _jmethodID*, ...)+124)
#20  pc 0x00000000000cd228  /system/lib64/libandroid_runtime.so (android::AndroidRuntime::start(char const*, android::Vector<android::String8> const&, b
#21  pc 0x0000000000002610  /system/bin/app_process64 (main+1464)
#22  pc 0x0000000000075c7c  /apex/com.android.runtime/lib64/bionic/libc.so (__libc_init+100)
```

```
#00  pc 0x0000000000354df8  /apex/com.android.art/lib64/libart.so (art::mirror::Class::PrettyDescriptor(art::ObjPtr<art::mirror::Class>)+92)
#01  pc 0x00000000005b6d3c  /apex/com.android.art/lib64/libart.so (art::ThrowIllegalAccessErrorClass(art::ObjPtr<art::mirror::Class>, art::ObjPtr<art::m
#02  pc 0x000000000031e2d8  /apex/com.android.art/lib64/libart.so (art::ResolveVerifyAndClinit(art::dex::TypeIndex, art::ArtMethod*, art::Thread*, bool,
#03  pc 0x000000000031d630  /apex/com.android.art/lib64/libart.so (NterpGetClass+84)
#04  pc 0x000000000058fd40  /apex/com.android.art/lib64/libart.so (nterp_get_class+48)
#05  pc 0x0000000000582250  /apex/com.android.art/lib64/libart.so (nterp_op_check_cast+80)
#06  pc 0x0000000000582064  /data/app/~~J8RAXW4zgzyNlqcr7aX-yg==/x.y.z-bmIqcilljyqazmJfmFvtHQ==/oat/arm64/base.vdex (androidx.recyclerview.widget.d.a+12
#07  pc 0x000000000058ac54  /apex/com.android.art/lib64/libart.so (nterp_helper+3924)
#08  pc 0x000000000058d352  /data/app/~~J8RAXW4zgzyNlqcr7aX-yg==/x.y.z-bmIqcilljyqazmJfmFvtHQ==/oat/arm64/base.vdex (androidx.recyclerview.widget.Recycl
#09  pc 0x000000000058ac54  /apex/com.android.art/lib64/libart.so (nterp_helper+3924)
#10  pc 0x0000000000589ae2  /data/app/~~J8RAXW4zgzyNlqcr7aX-yg==/x.y.z-bmIqcilljyqazmJfmFvtHQ==/oat/arm64/base.vdex (androidx.recyclerview.widget.Linear
#11  pc 0x000000000058ac54  /apex/com.android.art/lib64/libart.so (nterp_helper+3924)
#12  pc 0x00000000005894c6  /data/app/~~J8RAXW4zgzyNlqcr7aX-yg==/x.y.z-bmIqcilljyqazmJfmFvtHQ==/oat/arm64/base.vdex (androidx.recyclerview.widget.Linear
#13  pc 0x000000000058ac54  /apex/com.android.art/lib64/libart.so (nterp_helper+3924)
#14  pc 0x000000000058a3b0  /data/app/~~J8RAXW4zgzyNlqcr7aX-yg==/x.y.z-bmIqcilljyqazmJfmFvtHQ==/oat/arm64/base.vdex (androidx.recyclerview.widget.Linear
#15  pc 0x000000000058ac54  /apex/com.android.art/lib64/libart.so (nterp_helper+3924)
#16  pc 0x0000000000594e38  /data/app/~~J8RAXW4zgzyNlqcr7aX-yg==/x.y.z-bmIqcilljyqazmJfmFvtHQ==/oat/arm64/base.vdex (androidx.recyclerview.widget.Recycl
#17  pc 0x000000000058b358  /apex/com.android.art/lib64/libart.so (nterp_helper+5720)
#18  pc 0x0000000000594264  /data/app/~~J8RAXW4zgzyNlqcr7aX-yg==/x.y.z-bmIqcilljyqazmJfmFvtHQ==/oat/arm64/base.vdex (androidx.recyclerview.widget.Recycl
#19  pc 0x000000000058ac54  /apex/com.android.art/lib64/libart.so (nterp_helper+3924)
#20  pc 0x0000000000595ad2  /data/app/~~J8RAXW4zgzyNlqcr7aX-yg==/x.y.z-bmIqcilljyqazmJfmFvtHQ==/oat/arm64/base.vdex (androidx.recyclerview.widget.Recycl
```

```
#21  pc 0x0000000000c7dfa4  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (android.view.View.layout+420)
#22  pc 0x0000000000d1f450  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (android.view.ViewGroup.layout+208)
#23  pc 0x0000000000d77ea4  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (android.widget.FrameLayout.layoutChildren+804)
#24  pc 0x0000000000d77f48  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (android.widget.FrameLayout.onLayout+56)
#25  pc 0x0000000000c7dfa4  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (android.view.View.layout+420)
#26  pc 0x0000000000d1f450  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (android.view.ViewGroup.layout+208)
#27  pc 0x0000000000d77ea4  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (android.widget.FrameLayout.layoutChildren+804)
#28  pc 0x0000000000d77f48  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (android.widget.FrameLayout.onLayout+56)
#29  pc 0x0000000000c7dfa4  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (android.view.View.layout+420)
#30  pc 0x0000000000d1f450  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (android.view.ViewGroup.layout+208)
#31  pc 0x000000000058acb0  /apex/com.android.art/lib64/libart.so (nterp_helper+4016)
#32  pc 0x0000000000523f74  /data/app/~~J8RAXW4zgzyNlqcr7aX-yg==/x.y.z-bmIqci1ljyqazmJfmFvtHQ==/oat/arm64/base.vdex (androidx.constraintlayout.widget.Co
#33  pc 0x0000000000c7dfa4  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (android.view.View.layout+420)
#34  pc 0x0000000000d1f450  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (android.view.ViewGroup.layout+208)
#35  pc 0x0000000000d77ea4  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (android.widget.FrameLayout.layoutChildren+804)
#36  pc 0x0000000000d77f48  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (android.widget.FrameLayout.onLayout+56)
#37  pc 0x0000000000c7dfa4  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (android.view.View.layout+420)
#38  pc 0x0000000000d1f450  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (android.view.ViewGroup.layout+208)
#39  pc 0x0000000000d77ea4  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (android.widget.FrameLayout.layoutChildren+804)
#40  pc 0x0000000000d77f48  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (android.widget.FrameLayout.onLayout+56)
#41  pc 0x0000000000c7dfa4  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (android.view.View.layout+420)
#42  pc 0x0000000000d1f450  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (android.view.ViewGroup.layout+208)
#43  pc 0x0000000000d77ea4  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (android.widget.FrameLayout.layoutChildren+804)
#44  pc 0x0000000000d77f48  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (android.widget.FrameLayout.onLayout+56)
#45  pc 0x0000000000c7dfa4  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (android.view.View.layout+420)
#46  pc 0x0000000000d1f450  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (android.view.ViewGroup.layout+208)
#47  pc 0x0000000000d7d6c4  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (android.widget.LinearLayout.layoutVertical+692)
#48  pc 0x0000000000d7ff70  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (android.widget.LinearLayout.onLayout+64)
#49  pc 0x0000000000c7dfa4  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (android.view.View.layout+420)
#50  pc 0x0000000000d1f450  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (android.view.ViewGroup.layout+208)
#51  pc 0x0000000000d77ea4  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (android.widget.FrameLayout.layoutChildren+804)
#52  pc 0x0000000000a18268  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (com.android.internal.policy.DecorView.onLayout+88)
#53  pc 0x0000000000c7dfa4  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (android.view.View.layout+420)
#54  pc 0x0000000000d1f450  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (android.view.ViewGroup.layout+208)
#55  pc 0x0000000000cabfc8  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (android.view.ViewRootImpl.performLayout+600)
#56  pc 0x0000000000cafe64  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (android.view.ViewRootImpl.performTraversals+14532)
#57  pc 0x0000000000cb8d28  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (android.view.ViewRootImpl.doTraversal+216)
#58  pc 0x0000000000bd2c0c  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (android.view.ViewRootImpl$TraversalRunnable.run+60)
#59  pc 0x0000000000bafc98  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (android.view.Choreographer.doCallbacks+1416)
#60  pc 0x0000000000bb07f0  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (android.view.Choreographer.doFrame+1712)
#61  pc 0x0000000000c3e5a8  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (android.view.Choreographer$FrameDisplayEventReceiver.run+88
#62  pc 0x0000000000a302dc  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (android.os.Handler.dispatchMessage+76)
#63  pc 0x0000000000a33bd8  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (android.os.Looper.loopOnce+1000)
#64  pc 0x0000000000a33748  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (android.os.Looper.loop+1112)
#65  pc 0x0000000000774840  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (android.app.ActivityThread.main+2432)
#66  pc 0x0000000000360880  /apex/com.android.art/lib64/libart.so (art_quick_invoke_static_stub+640)
#67  pc 0x000000000026a904  /apex/com.android.art/lib64/libart.so (_jobject* art::InvokeMethod<(art::PointerSize)8>(art::ScopedObjectAccessAlreadyRunnab
#68  pc 0x000000000026a5e8  /apex/com.android.art/lib64/libart.so (art::Method_invoke(_JNIEnv*, _jobject*, _jobject*, _jobjectArray*) (.__uniq.165753521
#69  pc 0x00000000003346a8  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (art_jni_trampoline+120)
#70  pc 0x00000000009acce8  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (com.android.internal.os.RuntimeInit$MethodAndArgsCaller.run
#71  pc 0x00000000009b726c  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (com.android.internal.os.ZygoteInit.main+3548)
#72  pc 0x0000000000360880  /apex/com.android.art/lib64/libart.so (art_quick_invoke_static_stub+640)
#73  pc 0x00000000004944cc  /apex/com.android.art/lib64/libart.so (art::JValue art::InvokeWithVarArgs<_jmethodID*>(art::ScopedObjectAccessAlreadyRunnabl
#74  pc 0x0000000000553530  /apex/com.android.art/lib64/libart.so (art::JNI<true>::CallStaticVoidMethodV(_JNIEnv*, _jclass*, _jmethodID*, std::__va_list
#75  pc 0x00000000000c0ad0  /system/lib64/libandroid_runtime.so (_JNIEnv::CallStaticVoidMethod(_jclass*, _jmethodID*, ...)+120)
#76  pc 0x00000000000ccdf8  /system/lib64/libandroid_runtime.so (android::AndroidRuntime::start(char const*, android::Vector<android::String8> const&, b
#77  pc 0x0000000000002568  /system/bin/app_process64 (main+1300)
#78  pc 0x000000000004a12c  /apex/com.android.runtime/lib64/bionic/libc.so (__libc_init+96)
```

vm...@google.com <vm...@google.com> #8

Re #6: The code appears to be correct (except for lack of error checking, so likely to crash badly when ART throws `OutOfMemoryError` in `NewLongArray`/`NewIntArray`). However, I would re

However, this seems to be unrelated to the original bug report. The `CallStaticVoidMethod()` seen at the bottom of the stack in comment #7 is a well tested call and apparently allows the coc

If disassembly does not uncover the bug for issues in comment #6, you should file a separate bug report.

ng...@google.com <ng...@google.com> #9

Reassigned to mb...@gmail.com.

The issue may have been resolved. Have you seen it go down? What is the window of dates where you see the crashes?

ni...@gmail.com <ni...@gmail.com> #10

Re #8: I have checked the disassembly of `JvmManager::VibrateDevice()` and there doesn't appear to be any sign-/zero-extend ops being performed on arguments before branching to libA

In any case, I have updates regarding my original diagnosis of the issue (see #6): I changed the method signature to accept two `jIntArray`s and the crash still happens. I also noticed that p

```
thread_list.cc:1314] Check failed: self->GetState() != ThreadState::kRunnable (self->GetState()=Runnable, ThreadState::kRunnable=Runnable)
Signal: Segmentation fault (PC: 0x9D4B3938)
Stack Trace:
* 0x9D4B3938
* 0x74CEFEF485
* 0x69B292EC70
```

vm...@google.com <vm...@google.com> #11

Re #10: Please file a separate bug as this is unrelated to the original report in this bug. Post a link to the new bug here.

mb...@gmail.com <mb...@gmail.com> #12

Re #9: The crashes indeed seem to go down slowly now.
They last peaked at September 28 and October 3 (depending on the stack trace and the device model).
On Pixel devices, earlier (smaller) peaks were around September 10 to September 18 and August 24 to August 26.

ni...@gmail.com <ni...@gmail.com> #13

Re #11/#8: New bug report here: https://issuetracker.google.com/issues/304071459.

mb...@gmail.com <mb...@gmail.com> #14

Unfortunately, the situation did not really improve although we see slighty different stacktraces now. Here are two examples:

```
*** *** *** *** *** *** *** *** *** *** *** *** *** *** *** ***
pid: 0, tid: 13422

backtrace:
  #00  pc 0x0000000000362b40  /apex/com.android.art/lib64/libart.so (art::detail::ShortyTraits<(char)76>::Type art::ArtMethod::InvokeVirtual<(char)76, (ch
  #01  pc 0x000000000025d01c  /apex/com.android.art/lib64/libart.so (art::ClassLinker::FindClass(art::Thread*, char const*, art::Handle<art::mirror::Class
  #02  pc 0x00000000002cc1e4  /apex/com.android.art/lib64/libart.so (art::ObjPtr<art::mirror::Class> art::ClassLinker::DoResolveType<art::ArtMethod*>(art:
  #03  pc 0x00000000002cb21c  /apex/com.android.art/lib64/libart.so (art::ResolveVerifyAndClinit(art::dex::TypeIndex, art::ArtMethod*, art::Thread*, bool,
  #04  pc 0x00000000002cae44  /apex/com.android.art/lib64/libart.so (NterpGetClass+84)
  #05  pc 0x00000000005d55c0  /apex/com.android.art/lib64/libart.so (nterp_get_class+48)
  #06  pc 0x00000000005c7ad0  /apex/com.android.art/lib64/libart.so (nterp_op_check_cast+80)
  #07  pc 0x0000000000a91bc8  /data/app/~~JTi_dsU5GlnsbL7Ftyj7eg==/ch.threema.app-LURmf5qlxvkPNvT9cu5PnA==/oat/arm64/base.vdex (com.bumptech.glide.load.en
  #08  pc 0x00000000005d04d4  /apex/com.android.art/lib64/libart.so (nterp_helper+3924)
  #09  pc 0x0000000000a91a56  /data/app/~~JTi_dsU5GlnsbL7Ftyj7eg==/ch.threema.app-LURmf5qlxvkPNvT9cu5PnA==/oat/arm64/base.vdex (com.bumptech.glide.load.en
  #10  pc 0x000000000066705c  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (java.util.concurrent.ThreadPoolExecutor.runWorker+796)
  #11  pc 0x0000000000664180  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (java.util.concurrent.ThreadPoolExecutor$Worker.run+64)
  #12  pc 0x00000000005d1354  /apex/com.android.art/lib64/libart.so (nterp_helper+7636)
  #13  pc 0x0000000000a91a0c  /data/app/~~JTi_dsU5GlnsbL7Ftyj7eg==/ch.threema.app-LURmf5qlxvkPNvT9cu5PnA==/oat/arm64/base.vdex (com.bumptech.glide.load.en
  #14  pc 0x0000000000512ae8  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (java.lang.Thread.run+72)
  #15  pc 0x00000000003371a4  /apex/com.android.art/lib64/libart.so (art_quick_invoke_stub+612)
  #16  pc 0x000000000023ea64  /apex/com.android.art/lib64/libart.so (art::ArtMethod::Invoke(art::Thread*, unsigned int*, unsigned int, art::JValue*, char
  #17  pc 0x000000000054436c  /apex/com.android.art/lib64/libart.so (art::Thread::CreateCallback(void*)+1600)
  #18  pc 0x00000000000eb910  /apex/com.android.runtime/lib64/bionic/libc.so (__pthread_start(void*)+208)
  #19  pc 0x000000000007e4c0  /apex/com.android.runtime/lib64/bionic/libc.so (__start_thread+64)
```

```
*** *** *** *** *** *** *** *** *** *** *** *** *** *** *** ***
pid: 0, tid: 11563

backtrace:
  #00  pc 0x0000000000362b40  /apex/com.android.art/lib64/libart.so (art::detail::ShortyTraits<(char)76>::Type art::ArtMethod::InvokeVirtual<(char)76, (ch
  #01  pc 0x000000000025d01c  /apex/com.android.art/lib64/libart.so (art::ClassLinker::FindClass(art::Thread*, char const*, art::Handle<art::mirror::Class
  #02  pc 0x000000000035d358  /apex/com.android.art/lib64/libart.so (art::ClassLinker::ResolveType(art::dex::TypeIndex, art::Handle<art::mirror::DexCache>
  #03  pc 0x000000000035b030  /apex/com.android.art/lib64/libart.so (art::ArtMethod* art::ClassLinker::ResolveMethod<(art::ClassLinker::ResolveMode)1>(uns
  #04  pc 0x000000000023b628  /apex/com.android.art/lib64/libart.so (NterpGetMethod+3056)
  #05  pc 0x00000000005d56a0  /apex/com.android.art/lib64/libart.so (nterp_get_method+48)
  #06  pc 0x00000000005ca2bc  /apex/com.android.art/lib64/libart.so (nterp_op_invoke_super+60)
  #07  pc 0x0000000000e7ccfa  /data/app/~~dPqg4G6f_mTDsaLf017oJw==/ch.threema.app-pZPCLCxYh1qtAmy--YT50g==/oat/arm64/base.vdex (net.zetetic.database.sqlci
  #08  pc 0x00000000005d3a7c  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (java.lang.Daemons$FinalizerDaemon.doFinalize+284)
  #09  pc 0x00000000005d0530  /apex/com.android.art/lib64/libart.so (nterp_helper+4016)
  #10  pc 0x000000000002b13e  /apex/com.android.art/javalib/core-libart.jar (java.lang.Daemons$FinalizerDaemon.processReference+26)
  #11  pc 0x00000000005d3c70  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (java.lang.Daemons$FinalizerDaemon.runInternal+336)
  #12  pc 0x00000000005a7edc  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (java.lang.Daemons$Daemon.run+172)
  #13  pc 0x00000000003ea7c8  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (java.lang.Thread.run+72)
  #14  pc 0x00000000003371a4  /apex/com.android.art/lib64/libart.so (art_quick_invoke_stub+612)
  #15  pc 0x000000000023ea64  /apex/com.android.art/lib64/libart.so (art::ArtMethod::Invoke(art::Thread*, unsigned int*, unsigned int, art::JValue*, char
  #16  pc 0x000000000054436c  /apex/com.android.art/lib64/libart.so (art::Thread::CreateCallback(void*)+1600)
  #17  pc 0x00000000000b63b0  /apex/com.android.runtime/lib64/bionic/libc.so (__pthread_start(void*)+208)
  #18  pc 0x00000000000530b8  /apex/com.android.runtime/lib64/bionic/libc.so (__start_thread+64)
```

vm...@google.com <vm...@google.com> #15

Both stack traces show ART doing a virtual call to `String ClassLoader.loadClass(String)` on the calling method's class loader.
This code path is exercised all the time, so it's odd to see crashes like this.

But if you're playing tricks with the signal handler and somehow prevent ART from intercepting SIGSEGV, you can break ART's new userfaultfd GC which may be enabled for some users for te

---

mb...@gmail.com <mb...@gmail.com> #16

I wouldn't know how to play tricks with the signal handler.

How are testers for ART's new userfaultfd GC recruited? Is there a way for these users to go back to the old GC? A testing scenario would explain why we don't experience these crashes but s

---

vm...@google.com <vm...@google.com> #17

Sorry, please disregard the second part of comment #15. I was somehow confusing this bug with the one that was split out (see comment #13).