High volume of crashes in libmonochrome_64 since August 1st

STATUS UPDATE No update yet. Edit

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

Since August 1st, we've seen a high volume of the crashes from libmonochrome_64 (see report below), reported in the Google Play Console. We serve ads in our apps that use WebViews, and th
11, 12, and 13.

```
[FATAL:crashpad_client_linux.cc(732)] Render process (5210)'s crash wasn't handled by all associated  webviews, triggering application crash.

*** *** *** *** *** *** *** *** *** *** *** *** *** *** *** ***
pid: 0, tid: 5042 >>> com.astarsoftware.spades <<<

backtrace:
  #00  pc 0x0000000002c24edc  /data/app/~~pJ1_6o5AyVumVFBTbnq-hw==/com.google.android.trichromelibrary_593806033-psFSEOHrU7ziHmpL2Fe-dw==/base.apk!libmonochr
  #01  pc 0x0000000006bf2ec8  /data/app/~~pJ1_6o5AyVumVFBTbnq-hw==/com.google.android.trichromelibrary_593806033-psFSEOHrU7ziHmpL2Fe-dw==/base.apk!libmonochr
  #02  pc 0x00000000045e91a8  /data/app/~~pJ1_6o5AyVumVFBTbnq-hw==/com.google.android.trichromelibrary_593806033-psFSEOHrU7ziHmpL2Fe-dw==/base.apk!libmonochr
  #03  pc 0x0000000002d22398  /data/app/~~pJ1_6o5AyVumVFBTbnq-hw==/com.google.android.trichromelibrary_593806033-psFSEOHrU7ziHmpL2Fe-dw==/base.apk!libmonochr
  #04  pc 0x00000000061f61fc  /data/app/~~pJ1_6o5AyVumVFBTbnq-hw==/com.google.android.trichromelibrary_593806033-psFSEOHrU7ziHmpL2Fe-dw==/base.apk!libmonochr
  #05  pc 0x0000000003c6045c  /data/app/~~pJ1_6o5AyVumVFBTbnq-hw==/com.google.android.trichromelibrary_593806033-psFSEOHrU7ziHmpL2Fe-dw==/base.apk!libmonochr
  #06  pc 0x0000000003c603cc  /data/app/~~pJ1_6o5AyVumVFBTbnq-hw==/com.google.android.trichromelibrary_593806033-psFSEOHrU7ziHmpL2Fe-dw==/base.apk!libmonochr
  #07  pc 0x0000000000018224  /system/lib64/libutils.so (android::Looper::pollInner(int)+1060)
  #08  pc 0x0000000000017da0  /system/lib64/libutils.so (android::Looper::pollOnce(int, int*, int*, void**)+112)
  #09  pc 0x00000000001618ec  /system/lib64/libandroid_runtime.so (android::android_os_MessageQueue_nativePollOnce(_JNIEnv*, _jobject*, long, int)+44)
  #10  pc 0x000000000036b514  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (art_jni_trampoline+116)
  #11  pc 0x0000000000a95970  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (android.os.MessageQueue.next+304)
  #12  pc 0x0000000002020ae8  /memfd:jit-cache (android.os.Looper.loopOnce+152)
  #13  pc 0x0000000000a92a94  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (android.os.Looper.loop+1124)
  #14  pc 0x000000000083efd8  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (android.app.ActivityThread.main+2264)
  #15  pc 0x0000000000360880  /apex/com.android.art/lib64/libart.so (art_quick_invoke_static_stub+640)
  #16  pc 0x000000000026a904  /apex/com.android.art/lib64/libart.so (_jobject* art::InvokeMethod<(art::PointerSize)8>(art::ScopedObjectAccessAlreadyRunnable
  #17  pc 0x000000000026a5e8  /apex/com.android.art/lib64/libart.so (art::Method_invoke(_JNIEnv*, _jobject*, _jobject*, _jobjectArray*) (.__uniq.165753521025
  #18  pc 0x00000000003716a8  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (art_jni_trampoline+120)
  #19  pc 0x0000000000d9df18  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (com.android.internal.os.RuntimeInit$MethodAndArgsCaller.run+13
  #20  pc 0x0000000000da9778  /data/misc/apexdata/com.android.art/dalvik-cache/arm64/boot.oat (com.android.internal.os.ZygoteInit.main+3368)
  #21  pc 0x0000000000360880  /apex/com.android.art/lib64/libart.so (art_quick_invoke_static_stub+640)
  #22  pc 0x00000000004944cc  /apex/com.android.art/lib64/libart.so (art::JValue art::InvokeWithVarArgs<_jmethodID*>(art::ScopedObjectAccessAlreadyRunnable c
  #23  pc 0x0000000000553530  /apex/com.android.art/lib64/libart.so (art::JNI<true>::CallStaticVoidMethodV(_JNIEnv*, _jclass*, _jmethodID*, std::__va_list)+1
  #24  pc 0x00000000000bdce8  /system/lib64/libandroid_runtime.so (_JNIEnv::CallStaticVoidMethod(_jclass*, _jmethodID*, ...)+120)
  #25  pc 0x00000000000ca308  /system/lib64/libandroid_runtime.so (android::AndroidRuntime::start(char const*, android::Vector<android::String8> const&, bool
  #26  pc 0x0000000000002560  /system/bin/app_process64 (main+1280)
  #27  pc 0x0000000000084d70  /apex/com.android.runtime/lib64/bionic/libc.so (__libc_init+96)
```

✓ Links (1)

Links (1)

"Unfortunately Google Play Console doesn't include full bug reports so we can't proceed further. We could maybe use the app's package name to search in WebView's own crash database (Internal lin

COMMENTS

ra...@google.com <ra...@google.com>

Assigned to cl...@google.com.

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

Hey, can you please provide a bugreport so we can investigate further?

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

Hi, unfortunately, we are just seeing the large volume of crashes in Google Play Console, and we can't reproduce them on our devices to provide a bug report. Google Play Console has been f

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

Very nice volume 200

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

Hi there, can we get some more information about the crashes you are seeing?

- Are they all occurring on a particular WebView version?
- Is the issue not happening on older or newer WebViews i.e. is this a WebView version regression?

Symbolizing the logs pasted above: WebView info: version 117.0.5938.60, version-code 593806033

All I can see is that android::AndroidRuntime::start is called and then eventually the renderer process crashes at android_webview::(anonymousnamespace)::OnRenderProcessGone() --> This

```
[FATAL:crashpad_client_linux.cc(732)] Render process (5210)'s crash wasn't handled by all associated  webviews, triggering application crash.

*** *** *** *** *** *** *** *** *** *** *** *** *** *** *** ***
pid: 0, tid: 5042 >>> com.astarsoftware.spades <<<

backtrace:

Stack Trace:
  RELADDR   FUNCTION                                                                           FILE:LINE
  v------>  base::ImmediateCrash()                                                             ../../base/immediate_crash.h:146:3
  0000000002c24edc  logging::LogMessage::~LogMessage()                                                 ../../base/logging.cc:959:7
  0000000006bf2ec8  crashpad::CrashpadClient::CrashWithoutDump(std::__Cr::basic_string<char, std::__Cr::char_traits<char>, std::__Cr::allocator<char>> con
  v------>  crash_reporter::CrashWithoutDumping(std::__Cr::basic_string<char, std::__Cr::char_traits<char>, std::__Cr::allocator<char>> const&)  ../../com
  00000000045e91a8  android_webview::(anonymous namespace)::OnRenderProcessGone(std::__Cr::vector<base::android::ScopedJavaGlobalRef<_jobject*>, std::__Cr
  v------>  base::OnceCallback<void ()>::Run() &&                                              ../../base/functional/callback.h:152:12
  0000000002d22398  base::TaskAnnotator::RunTaskImpl(base::PendingTask&)                               ../../base/task/common/task_annotator.cc:201:34
  v------>  void base::TaskAnnotator::RunTask<base::sequence_manager::internal::ThreadControllerWithMessagePumpImpl::DoWorkImpl(base::LazyNow*)::$_0>(perf
  v------>  base::sequence_manager::internal::ThreadControllerWithMessagePumpImpl::DoWorkImpl(base::LazyNow*)  ../../base/task/sequence_manager/thread_con
  v------>  base::sequence_manager::internal::ThreadControllerWithMessagePumpImpl::DoWork()   ../../base/task/sequence_manager/thread_controller_with_mess
  00000000061f61fc  non-virtual thunk to base::sequence_manager::internal::ThreadControllerWithMessagePumpImpl::DoWork()  ../../base/task/sequence_manager
  0000000003c6045c  base::MessagePumpForUI::DoNonDelayedLooperWork(bool)                               ../../base/message_loop/message_pump_android.cc:186:
  v------>  base::MessagePumpForUI::OnNonDelayedLooperCallback()                               ../../base/message_loop/message_pump_android.cc:172:3
  0000000003c603cc  base::(anonymous namespace)::NonDelayedLooperCallback(int, int, void*)           ../../base/message_loop/message_pump_android.cc:43:9
  0000000000018224  android::Looper::pollInner(int)+1060                                               /system/lib64/libutils.so
  0000000000017da0  android::Looper::pollOnce(int, int*, int*, void**)+112                             /system/lib64/libutils.so
  00000000001618ec  android::android_os_MessageQueue_nativePollOnce(_JNIEnv*, _jobject*, long, int)+44  /system/lib64/libandroid_runtime.so
  000000000036b514  art_jni_trampoline+116                                                             /data/misc/apexdata/com.android.art/dalvik-cache/arm
  0000000000a95970  android.os.MessageQueue.next+304                                                   /data/misc/apexdata/com.android.art/dalvik-cache/arm
  0000000002020ae8  android.os.Looper.loopOnce+152                                                     /memfd:jit-cache
  0000000000a92a94  android.os.Looper.loop+1124                                                        /data/misc/apexdata/com.android.art/dalvik-cache/arm
  000000000083efd8  android.app.ActivityThread.main+2264                                               /data/misc/apexdata/com.android.art/dalvik-cache/arm
  0000000000360880  art_quick_invoke_static_stub+640                                                   /apex/com.android.art/lib64/libart.so
  000000000026a904  _jobject* art::InvokeMethod<(art::PointerSize)8>(art::ScopedObjectAccessAlreadyRunnable const&, _jobject*, _jobject*, _jobject*, unsig
  000000000026a5e8  art::Method_invoke(_JNIEnv*, _jobject*, _jobject*, _jobjectArray*) (.__uniq.165753521025965369065708152063621506277)+32  /apex/com.and
  00000000003716a8  art_jni_trampoline+120                                                             /data/misc/apexdata/com.android.art/dalvik-cache/arm
  0000000000d9df18  com.android.internal.os.RuntimeInit$MethodAndArgsCaller.run+136                   /data/misc/apexdata/com.android.art/dalvik-cache/arm
  0000000000da9778  com.android.internal.os.ZygoteInit.main+3368                                       /data/misc/apexdata/com.android.art/dalvik-cache/arm
  0000000000360880  art_quick_invoke_static_stub+640                                                   /apex/com.android.art/lib64/libart.so
  00000000004944cc  art::JValue art::InvokeWithVarArgs<_jmethodID*>(art::ScopedObjectAccessAlreadyRunnable const&, _jobject*, _jmethodID*, std::__va_list)
  0000000000553530  art::JNI<true>::CallStaticVoidMethodV(_JNIEnv*, _jclass*, _jmethodID*, std::__va_list)+112  /apex/com.android.art/lib64/libart.so
  00000000000bdce8  _JNIEnv::CallStaticVoidMethod(_jclass*, _jmethodID*, ...)+120                     /system/lib64/libandroid_runtime.so
  00000000000ca308  android::AndroidRuntime::start(char const*, android::Vector<android::String8> const&, bool)+840  /system/lib64/libandroid_runtime.so
  0000000000002560  main+1280                                                                          /system/bin/app_process64
  0000000000084d70  __libc_init+96
```

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

Unfortunately, we haven't been able to reproduce these crashes on our devices, so the log above is all I'm able to provide right now. Google Play Console does not provide stacktraces for othe
all WebViews, and some ad networks like Digital Turbine are releasing updates to try to resolve the crashes. For Digital Turbine, it seemed like many of their ads were triggering the crashes, s
so we might try that out to mitigate this issue.

nt...@google.com <nt...@google.com> #7

Status: Won't Fix (Infeasible)

> All I can see is that android::AndroidRuntime::start is called and then eventually the renderer process crashes at android_webview::(anonymousnamespace)::OnRenderProcessGone() --> Th

Hey Aashna, thanks for looking into this. Unfortunately it's not usually necessary to symbolize crashes like this. "Render process (5210)'s crash wasn't handled by all associated webviews, tri
crash will always point back to `onRenderProcessGone`, because that's where we detected the renderer process crash. And as you pointed out, we can't symbolize the actual renderer process

Unfortunately Google Play Console doesn't include full bug reports so we can't proceed further. We could maybe use the app's package name to search in WebView's own crash database (Int

> AppLovin informed us that they have a tool in beta for automatically adding onRenderProcessGone for WebViews that don't have it, so we might try that out to mitigate this issue.

Yes, if the app developer or the app framework you're using can incorporate onRenderProcessGone then you may be able to recover from some of these crashes. However if the issue is "out

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

I'm going to close this ticket because I don't think there's much else our team can do.