SIGSEGV occurs when attaching a child process.

STATUS UPDATE No update yet. Edit

DESCRIPTION su...@cyphertec.co.jp created issue #1

- Are you an Android developer?" (Y/N)

  Y

- Which Android Developer Preview build are you using? See Settings > About phone > Build number (for example TPB3.220624.005).

  UPB5.230623.006

- What device are you using? (for example, Android Emulator, GSI, Pixel 6)

  Pixel 7a

- Description of problem:

  Attaching a forked child process to a parent process using ptrace immediately raises SIGSEGV. Even though I tried to reattach, SIGSEGV occurred and the attached state cannot be maintain SIGSEGV does not occur and the attached state of the child process continues.

  The sample code is below:

```
#include <jni.h>
#include <string.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/ptrace.h>
#include <sys/wait.h>
#include <pthread.h>
#include <android/log.h>

static int child_pid;

void *monitor_pid(void *arg) {
        int status;
        waitpid(child_pid, &status, 0);
        /* Child status should never change. */
        _exit(0); // Commit seppuku
}

void start_anti_debug() {
        child_pid = fork();
        if (child_pid == 0)
        {
                int ppid = getppid();
                int status;

                if (ptrace(PTRACE_ATTACH, ppid, NULL, NULL) == 0)
                {
                        __android_log_print(ANDROID_LOG_DEBUG, "ANTIDEBUG", "[in child] PTRACE_ATTACH %d", ppid);
                        waitpid(ppid, &status, 0);
                        ptrace(PTRACE_CONT, ppid, NULL, NULL);
                        __android_log_print(ANDROID_LOG_DEBUG, "ANTIDEBUG", "[in child] PTRACE_CONT %d", ppid);

                        while (waitpid(ppid, &status, 0)) {

                                if ( WIFSTOPPED(status) ) {
                                        __android_log_print(ANDROID_LOG_DEBUG, "ANTIDEBUG", "WIFSTOPPED WSTOPSIG %d", WSTOPSIG(status));
                                } else if ( WIFEXITED(status) ) {
                                        __android_log_print(ANDROID_LOG_DEBUG, "ANTIDEBUG", "WIFEXITED WEXITSTATUS %d", WEXITSTATUS(status));
                                } else if ( WIFSIGNALED(status) ) {
                                        __android_log_print(ANDROID_LOG_DEBUG, "ANTIDEBUG", "WIFSIGNALED WTERMSIG %d", WTERMSIG(status));
                                }

                                if (WIFSTOPPED(status)) {
                                        ptrace(PTRACE_CONT, ppid, NULL, NULL);
                                        __android_log_print(ANDROID_LOG_DEBUG, "ANTIDEBUG", "[in child] PTRACE_CONT %d", ppid);
                                } else {
                                        // Process has exited
                                        _exit(0);
                                }
                        }
                }
        } else {
                pthread_t t;
                /* Start the monitoring thread */
                pthread_create(&t, NULL, monitor_pid, (void *)NULL);
        }
}

extern "C" JNIEXPORT void JNICALL
Java_com_example_AntiDebuggerFragment_startAntiDebugger(
        JNIEnv *env,
        jobject /* this */) {
        __android_log_write(ANDROID_LOG_DEBUG, "ANTIDEBUG", "start AntiDebugger");
        start_anti_debug();
}
```

- What are the steps to reproduce the problem? (Please provide the minimal reproducible test case.)

  1. Run a sample app that runs the JNI code above.
  2. In terminal soft, check Logcat.

     adb -d logcat | grep ANTIDEBUG

  3. You can see that SIGSEGV is repeatedly detected and resumed after attaching the forked child process.

```
09-01 11:26:24.946 23654 23654 D ANTIDEBUG: WIFSTOPPED WSTOPSIG 11
09-01 11:26:24.946 23654 23654 D ANTIDEBUG: [in child] PTRACE_CONT 23598
09-01 11:26:24.946 23654 23654 D ANTIDEBUG: WIFSTOPPED WSTOPSIG 11
09-01 11:26:24.946 23654 23654 D ANTIDEBUG: [in child] PTRACE_CONT 23598
09-01 11:26:24.946 23654 23654 D ANTIDEBUG: WIFSTOPPED WSTOPSIG 11
09-01 11:26:24.946 23654 23654 D ANTIDEBUG: [in child] PTRACE_CONT 23598
09-01 11:26:24.946 23654 23654 D ANTIDEBUG: WIFSTOPPED WSTOPSIG 11
09-01 11:26:24.946 23654 23654 D ANTIDEBUG: [in child] PTRACE_CONT 23598
09-01 11:26:24.946 23654 23654 D ANTIDEBUG: WIFSTOPPED WSTOPSIG 11
09-01 11:26:24.946 23654 23654 D ANTIDEBUG: [in child] PTRACE_CONT 23598
09-01 11:26:24.946 23654 23654 D ANTIDEBUG: WIFSTOPPED WSTOPSIG 11
09-01 11:26:24.946 23654 23654 D ANTIDEBUG: [in child] PTRACE_CONT 23598
09-01 11:26:24.946 23654 23654 D ANTIDEBUG: WIFSTOPPED WSTOPSIG 11
09-01 11:26:24.946 23654 23654 D ANTIDEBUG: [in child] PTRACE_CONT 23598
09-01 11:26:24.946 23654 23654 D ANTIDEBUG: WIFSTOPPED WSTOPSIG 11
09-01 11:26:24.946 23654 23654 D ANTIDEBUG: [in child] PTRACE_CONT 23598
09-01 11:26:24.946 23654 23654 D ANTIDEBUG: WIFSTOPPED WSTOPSIG 11
09-01 11:26:24.946 23654 23654 D ANTIDEBUG: [in child] PTRACE_CONT 23598
09-01 11:26:24.946 23654 23654 D ANTIDEBUG: WIFSTOPPED WSTOPSIG 11
09-01 11:26:24.946 23654 23654 D ANTIDEBUG: [in child] PTRACE_CONT 23598
09-01 11:26:24.946 23654 23654 D ANTIDEBUG: WIFSTOPPED WSTOPSIG 11
09-01 11:26:24.946 23654 23654 D ANTIDEBUG: [in child] PTRACE_CONT 23598
09-01 11:26:24.946 23654 23654 D ANTIDEBUG: WIFSTOPPED WSTOPSIG 11
09-01 11:26:24.946 23654 23654 D ANTIDEBUG: [in child] PTRACE_CONT 23598
09-01 11:26:24.946 23654 23654 D ANTIDEBUG: WIFSTOPPED WSTOPSIG 11
09-01 11:26:24.946 23654 23654 D ANTIDEBUG: [in child] PTRACE_CONT 23598
```

- Issue Category e.g. Framework (platform), NDK (platform), Hardware (CPU, GPU, Sensor, Camera), ART (platform), Runtime Permissions etc

  NDK (platform), System

- What was the expected result?

  As in Android 13 and earlier, attaching a child process to a parent process will not raise a SIGSEGV and will continue to be attached.

- Can you provide the API document where this expected behavior is explained?

  Although it is not an Android Developers documentation, I provide a link to the Traditional Anti-Debugging documentation.

  - Traditional Anti-Debugging https://github.com/OWASP/owasp-mastg/blob/master/Document/0x05j-Testing-Resiliency-Against-Reverse-Engineering.md#traditional-anti-debugging

✓ Links (4)

Links (4)

"Traditional Anti-Debugging https://github.com/OWASP/owasp-mastg/blob/master/Document/0x05j-Testing-Resiliency-Against-Reverse-Engineering.md#traditional-anti-d… "
" … 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 Thank you for the
"Diagnosing Native Crashes https://source.android.com/docs/core/tests/debug/native-crash?hl=en"
"https://support.google.com/product-documentation/answer/11412553?hl=en&ref_topic=11273470&sjid=1479621115873… "

COMMENTS

All comments

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

Assigned to rh...@google.com.

For us to further investigate this issue, please provide the following additional information:

For us to further investigate this issue, please provide the following additional information:

- An 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#bugreportde report.
- Sample app, if possible.

su...@cyphertec.co.jp <su...@cyphertec.co.jp> #3

> For us to further investigate this issue, please provide the following additional information:

Please check the attached file "298530294.zip".

Also, README.md is included, so please check the included files and reproduction steps.

298530294.zip
54 MB Download

rh...@google.com <rh...@google.com> #4

Please provide a new Android bug report as the previous attachment did not contain much information. Thank you.

su...@cyphertec.co.jp <su...@cyphertec.co.jp> #5

> Please provide a new Android bug report as the previous attachment did not contain much information. Thank you.

I provide a new bug report. This bug report is obtained after running "poc_antidebug.apk" multiple times on an Android 14 device. ("poc_antidebug.apk" is included in 298530294.zip provided

However, a child process attached to the app's main process detects the SIGSEGV, but not the app's crash. Therefore, native code crash dumps such as the one below do not seem to include

- Diagnosing Native Crashes https://source.android.com/docs/core/tests/debug/native-crash?hl=en

I inform additional information regarding this ISSUE.

When attaching with gdb to a debuggable app on Android 14, the debugger process detects SIGSEGV and terminates. On Android 13 the gdb debugger process did not detect SIGSEGV and w believe gdb also uses ptrace. I think this ISSUE is caused by the same thing.

bugreport-lynx_beta-UPB5.230623.009-2023-09-07-12-02-24.zip
48 MB Download

rh...@google.com <rh...@google.com> #6

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

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

For us to further investigate this issue, please provide the following additional information:

- From the given bug report, we found that this, `libbinder ProcessState can not be used after fork`, caused an error in the application. Let us know if you have also seen this erro
- Also, confirm this issue reproduction step, `You can see that SIGSEGV is repeatedly detected and resumed after attaching the forked child process`. What specifically be considered reproducible? Are only the logs displayed in comment#1, such as `WIFSTOPPED` & `[in child] PTRACE_CONT` must be shown or any other information is needed to seen?

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

Please provide the requested information for us to further investigate this issue. Unfortunately the issue will be closed within 7 days if there is no further update.

su...@cyphertec.co.jp <su...@cyphertec.co.jp> #9

> From the given bug report, we found that this, libbinder ProcessState can not be used after fork, caused an error in the application. Let us know if you have also seen this error.

Thanks. I will check again to see if the same error occurs in the app and let you know.

> Also, confirm this issue reproduction step, You can see that SIGSEGV is repeatedly detected and resumed after attaching the forked child process. What specifically must show for this step reproducible? Are only the logs displayed in comment#1, such as WIFSTOPPED & [in child] PTRACE_CONT must be shown or any other information is needed to seen?

For the current investigation, the only way to see the behavior of the program is through logcat. You can reproduce this issue by running the sample app "poc_antidebug.apk". Please see "REA reproduce. These are presented in #comment3. Also, "poc_antidebug.zip" is included in "298530294.zip", which is the Android Studio project for this app, so please check this source code as

rh...@google.com <rh...@google.com> #10

Please provide the requested information for if you have seen the error mentioned in comment#7 for us to further investigate this issue. Thank you.

su...@cyphertec.co.jp <su...@cyphertec.co.jp> #11 S

> From the given bug report, we found that this, libbinder ProcessState can not be used after fork, caused an error in the application. Let us know if you have also seen this error.

I checked again, but I couldn't see that this error was occurring in the app. How can I check if this error has occurred in the app?

I provide additional information regarding this ISSUE.

This ISSUE occurred not only on Android 14, but also when applying the Google System Updates August 2023 on Android 13 or lower. Could you please check if there are any effects related t ptrace and waitpid caused by the Google System Updates August 2023?

https://support.google.com/product-documentation/answer/11412553?hl=en&ref_topic=11273470&sjid=14796211158733948929-AP#zippy=%2Caugust

The steps to reproduce are as follows:

1. Install the attached "poc_antidebug.apk". (poc_antidebug.apk is attached to this comment.)
2. Launch "poc_antidebug" app.
3. Tap "START ANTI DEBUGGING" on the app screen.

If the Google System Updates is before August 2023, the forked child process will remain attached to the parent process. After August 2023, WSTOPSIG(SIGSEGV) will be detected immediat forked child process to the parent process.

The source code of the application for reproduction steps is below. This source code is included in the attached "poc_antidebug.zip".

- app/src/main/cpp/native-lib.cpp

```
#include <jni.h>
#include <string.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/ptrace.h>
#include <sys/wait.h>
#include <sys/prctl.h>
#include <pthread.h>
#include <android/log.h>
#include <errno.h>

/*
 * Start anti-debugging using ptrace to attach the child process forked from the parent process.
 */
void *start_anti_debug(void *arg) {
    __android_log_write(ANDROID_LOG_DEBUG, "ANTIDEBUG", "start_anti_debug()...");

    int RETRY_MAX_COUNT = 100;
    for(int i=0; i<RETRY_MAX_COUNT; i++) {
        pid_t pid = fork();
        if (pid == -1) {
            _exit(0);
            break;
        }

        if (pid != 0) { // in parent process
            __android_log_write(ANDROID_LOG_DEBUG, "ANTIDEBUG", "in parent process.");
            int parentstatus = 0;
            int wait_rc = 0;
            errno = 0;
            do {
                wait_rc = waitpid(pid, &parentstatus, 0);
            } while (wait_rc == -1 && errno == EINTR);
        } else {
            __android_log_write(ANDROID_LOG_DEBUG, "ANTIDEBUG", "in child process.");
            int ppid = getppid();
            int status = 0;

            if (ptrace(PTRACE_ATTACH, ppid, nullptr, nullptr) == 0) {
                __android_log_print(ANDROID_LOG_DEBUG, "ANTIDEBUG", "[in child] PTRACE_ATTACH %d",
                                    ppid);
                ptrace(PTRACE_CONT, ppid, nullptr, nullptr);
                __android_log_print(ANDROID_LOG_DEBUG, "ANTIDEBUG", "[in child] PTRACE_CONT %d",
                                    ppid);

                // Note:
                // Google System Updates is before August 2023, the forked child process will
                // remain attached to the parent process.
                // After August 2023, WSTOPSIG(11) will be detected immediately after attaching
                // a forked child process to the parent process.
                while (waitpid(ppid, &status, 0)) {

                    if (WIFSTOPPED(status)) {

                        // Note:
```

```
                        // When checking the "WSTOPSIG(status)" in logcat, "11" is recorded.
                        // This is considered to be caused by SIGSEGV.
                        __android_log_print(ANDROID_LOG_DEBUG, "ANTIDEBUG",
                                            "WIFSTOPPED WSTOPSIG %d", WSTOPSIG(status));
                        ptrace(PTRACE_CONT, ppid, nullptr, nullptr);
                        __android_log_print(ANDROID_LOG_DEBUG, "ANTIDEBUG",
                                            "[in child] PTRACE_CONT %d", ppid);

                    } else {
                        // Process has exited
                        _exit(0);
                    }
                }
            }
        }
    }
}

/*
 * Starts the anti-debug function.
 * This method is called when you tap "START ANTI DEBUGGING" in the app.
 */
extern "C" JNIEXPORT void JNICALL
Java_com_example_poc_1antidebug_MainActivity_startAntiDebugger(
        JNIEnv* env,
        jobject /* this */) {
        __android_log_write(ANDROID_LOG_DEBUG, "ANTIDEBUG", "start AntiDebugger");

        prctl(PR_SET_DUMPABLE, 1);

        pthread_t t;
        pthread_create(&t, nullptr, start_anti_debug, (void *)nullptr);
}

/*
 * Stop the anti-debug function.
 * This method is called when you tap "STOP ANTI DEBUGGING" in the app.
 */
extern "C" JNIEXPORT void JNICALL
Java_com_example_poc_1antidebug_MainActivity_stopAntiDebugger(
        JNIEnv* env,
        jobject /* this */) {
    __android_log_write(ANDROID_LOG_DEBUG, "ANTIDEBUG", "stop AntiDebugger");
    _exit(0);
}
```

Additionally, after applying the Google Play System August update, even if I attach to the app with the gdb debugger, it is immediately terminated by SIGSEGV. I guess that this is also due to t
ISSUE.

poc_antidebug.apk
4.4 MB Download ?

poc_antidebug.zip
1.1 MB Download ?

rh...@google.com <rh...@google.com> #12 C

For us to further investigate this issue, please provide the following additional information:

- Request for you to run `adb logcat` in a terminal and observe if you can view the error: `Abort message: 'libbinder ProcessState can not be used after fork'`. Let us know th