# Sandboxed Helper Tool Crashing in `libsystem_secinit`

This thread has been locked. Questions are automatically locked after two months of inactivity, or sooner if deemed necessary by a moderator.

1.1k

A while back I created a test project that shows how to embed a helper tool within a sandboxed app. I recently had cause to dust off that project and send it to a developer. Annoyingly it didn't work. It took me a while to figure out why, and I thought other folks could benefit from my experience.

As a reminder, when you create a sandboxed app with a helper tool, you typically have one target for the app and another target for the helper tool, and then a Copy Files build phase to copy the helper tool to

```
Contents/MacOS/
```

in the app. The app has its usual array of entitlements (most notably,

```
com.apple.security.app-sandbox
```

to enable the App Sandbox) while the helper tool must have exactly these entitlements:

- `com.apple.security.app-sandbox` to enable the App Sandbox
- `com.apple.security.inherit` to indicate that the helper tool inherits its sandbox from the app

The app can then invoke the helper tool using

```
NSTask
```

:

```
NSTask * task = [[NSTask alloc] init];
task.launchPath = [[NSBundle mainBundle] URLForAuxiliaryExecutable:@"HelperTool"].path;
task.terminationHandler = ^(NSTask * task) {
    … check task.terminationStatus …
};
[task launch];
```

This setup has worked since the introduction of the App Sandbox, which is why I was surprised to have problems today.

The symptom of this problem was that

```
task.terminationStatus
```

was always 4, which suggests that the helper tool failed with a

```
SIGILL
```

signal. Looking in

```
Library/Logs/DiagnosticReports/
```

I found a crash report containing this text:

```
Application Specific Information:
dyld: launch, running initializers
/usr/lib/libSystem.B.dylib
Sandbox creation failed: Container object initialization failed.
failed to get bundleid for app "/Users/quinn/Library/Developer/Xcode/DerivedData/SandboxedToolTest-
gqcfcjjjlclmujdgnowfrexqhelb/Build/Products/Debug/SandboxedToolTest.app/Contents/MacOS/HelperTool"

Application Specific Signatures:
Container object initialization failed

Thread 0 Crashed:: Dispatch queue: com.apple.main-thread
0 libsystem_secinit.dylib … _libsecinit_setup_secinitd_client + 1616
1 libsystem_secinit.dylib … _libsecinit_initialize_once + 13
2 libdispatch.dylib       … _dispatch_client_callout + 8
3 libdispatch.dylib       … _dispatch_once_callout + 87
4 libsystem_secinit.dylib … _libsecinit_initializer + 79
5 libSystem.B.dylib       … libSystem_initializer + 136
6 dyld                    … ImageLoaderMachO::doModInitFunctions(ImageLoa…
```

The program failed to start because of a problem initialising the security subsystem. The text

```
failed to get bundleid
```

implies that it had a bundle ID problem. That's a complete red herring. I added a bundle ID to my helper tool and that just shifted around the failure, resulting in a crash report like this:

```
Application Specific Information:
dyld: launch, running initializers
/usr/lib/libSystem.B.dylib
Could not set sandbox profile data: Operation not permitted (1)

Application Specific Signatures:
SYSCALL_SET_PROFILE

Thread 0 Crashed:: Dispatch queue: com.apple.main-thread
0 libsystem_secinit.dylib … _libsecinit_setup_app_sandbox + 667
1 libsystem_secinit.dylib … _libsecinit_initialize_once + 20
2 libdispatch.dylib       … _dispatch_client_callout + 8
3 libdispatch.dylib       … _dispatch_once_callout + 87
4 libsystem_secinit.dylib … _libsecinit_initializer + 79
5 libSystem.B.dylib       … libSystem_initializer + 136
6 dyld                    … ImageLoaderMachO::doModInitFunctions(ImageLoad…
```

Here the security startup code is failing because it's trying to replace the sandbox of a running process with a different sandbox, something that's never allowed.

The actual cause of the problem is the entitlements of the helper tool. While my

```
.entitlements
```

file contained just the entitlements described above, it turns out that the entitlements baked into the helper tool executable looked like this:

```
$ codesign -d --entitlements :- SandboxedToolTest.app/Contents/MacOS/SandboxedToolTest
…
<dict>
    <key>com.apple.security.app-sandbox</key>
    <true/>
    <key>com.apple.security.files.user-selected.read-only</key>
    <true/>
    <key>com.apple.security.get-task-allow</key>
    <true/>
</dict>
</plist>
```

The

```
com.apple.security.get-task-allow
```

entitlement is normal for Development-signed code, so I didn't twig that it was a problem. Alas, dear reader, it was. The presence of this extra entitlement causes the security startup code to act like this is a new application, one that's expecting an entirely new sandbox, and that's not what we want.
This entitlement is added by Xcode because it's a valid default for most scenarios. Specifically, that addition is controlled by the Code Signing Inject Base Entitlements (

```
CODE_SIGN_INJECT_BASE_ENTITLEMENTS
```

) build setting. It's default value of true causes Xcode to add default entitlements appropriate for your platform. That's the right thing to do in most cases, but the wrong thing to do here.
The fix is to simply turn this build setting off for my helper tool target. After a rebuild I confirmed that the

```
com.apple.security.get-task-allow
```

entitlement is no longer present, and my tool works as expected.
Don't worry, you'll still be able to debug your helper tool. When the helper tool inherits its sandbox from the app, it inherits all static entitlements. So as long as the app has the

```
com.apple.security.get-task-allow
```

entitlement, you'll be able to debug both the app and the helper tool.
Share and Enjoy — Quinn "The Eskimo!" Apple Developer Relations, Developer Technical Support, Core OS/Hardware

```
let myEmail = "eskimo" + "1" + "@apple.com"
```

XPC

Asked 2 years ago by eskimo

Reply to this question