Sample Code

Constraining a tool's launch environment

Improve the security of your macOS app by limiting the ways its components can run.

Download

macOS 14.0+ | Xcode 15.0+

Overview

This sample code project contains a factored macOS app. The app target, Launch Constraints Sample App, contains a UI for adding two numbers. It communicates with a separate process, Helper Tool, which does the calculation.

The Helper Tool has a constrained launch environment, which limits the parent processes that are allowed to launch the helper. The launch constraint is defined in the property list file `helper_parent.coderequirement`. It restricts the code-signing identifier of the parent process to be the same as the bundle identifier of Launch Constraints Sample App, and the team identifier of the parent process to be your Apple Developer team.

When a process launches Helper Tool, the kernel checks whether the process's code signing identifier and team identifier have the expected values. If they do, then the Helper Tool runs normally; otherwise, it doesn't run.

Configure the sample code project

To configure the sample code project, do the following in Xcode:

1. Select the Xcode project in the Project navigator.
2. Select the Launch Constraints Sample App target in the Project editor.
3. Switch to the Signing & Capabilities tab.

4. Choose your team in the Signing Team drop-down menu.

5. Copy the value of the macOS Bundle Identifier build setting.

6. Select the file Helper Tool > `helper_parent.coderequirement` in the Project navigator.

7. Paste the value you copied in step 5 into the Value field for the Signing identifier key.

8. Copy your team identifier. This is a string of letters and numbers — such as B97A5CM278 — that follows `com.example.apple-samplecode.constraining-a-tools-launch-environment` in the signing identifier.

9. Paste the value you copied in step 8 into the Value field for the Team identifier key.

0. Save the file.

Run the Helper Tool as part of the app's workflow

In Xcode, ensure the Launch Constraints Sample App scheme is selected in the scheme drop-down menu, and choose Product > Run. Xcode launches the sample app. Verify that you can enter numbers into the left-most two text fields and click Calculate to compute their sum.

When you click Calculate, the `sum` function uses `Process` to launch its Helper Tool, and sets up a pair of `Pipe` objects to send input to the tool and receive its output. You use this design pattern to create GUI "wrappers" for command-line tools, such as database administration tools.

The app can launch the Helper Tool because when Xcode signs the app, it sets the app binary's signing identifier to the app's bundle ID, and embeds your Apple Developer team identifier. The Helper Tool's launch constraint tests that its parent process — in this case, the app — has that signing identifier and team identifier.

Run the Helper Tool as a standalone tool

In Xcode, switch to the Helper Tool scheme in the scheme drop-down menu, and choose Product > Run. Xcode doesn't run the Helper Tool, and an error appears in the debugger's output:

```
Message from debugger: Terminated due to code signing error
Program ended with exit code: 9
```

This error indicates that the Helper Tool's launch constraint, defined in `helper_parent.coderequirement` was violated, and the operating system didn't allow it to run. In this situation, the parent process of the Helper Tool is the debugger (`lldb`), which doesn't have the signing identifier expressed in the parent process launch constraint, and is signed by a different Apple Developer team.