Remote Debugging for Authorization Plugins

1.8k

Hi all,

We are developing a variety of login window plugins for macOS and are looking for greater control of the debugging process. I've followed the steps in Technical Note TN2108: Debugging An Authorization Plug-In With Xcode, but they don't seem to work on modern macOS, even with SIP disabled. It would seem that SecurityAgent has some anti-debug protection in it now as I can't connect to the running instance with lldb.

As a work around we've resorted to caveman debugging with lots of log statements, but that's more than a bit of a pain to do as our products become more complex. Having debugging in Xcode would be best, but at this rate I'll even settle for lldb.

Any ideas?

Thanks,

Josh

Security LLDB

Reply

Posted 4 years ago by NoMAD-Dev

Add a Comment

Apple Recommended

Someone pinged me about this during an Ask DTS session yesterday so I thought I'd post a quick update. In recent releases we've tweaked the security model of both macOS and authorisation plug-ins considerably. The good news is that this has *improved* the debugging story.

To debug a plug-in on 10.15 and later:

1. Set up a 'victim' machine. In some cases you may need to use real hardware, but in most cases you can get away with using a VM.
2. Disable System Integrity Protection (SIP).
   WARNING Disabling SIP reduces the security of your Mac. Do not disable SIP on a machine you use day-to-day. Rather, only disable SIP on a victim machine.
3. SSH into the victim machine.
4. Run LLDB as root:

```
% sudo lldb
```

5. Attach to the process that's hosting the plugin:

```
(lldb) process attach -p 1729
```

It's best to target the process using its process ID rather than name. It's not uncommon for there to be multiple instances of the authorization plug-in host process running concurrently.

The process hosting your plug-in varies by plug-in configuration and OS version:

macOS	Architecture	Non-Privileged	Privileged
...<10.14	Intel	`SecurityAgent`	`authorizationhost`
10.14...<11.0	Intel	`SecurityAgentHelper`	`authorizationhosthelper`
11.0...	Intel	`SecurityAgentHelper-x86_64`	`authorizationhosthelper.x86_64`
	Apple silicon	`SecurityAgentHelper-arm64`	`authorizationhosthelper.arm64`

Share and Enjoy
—
Quinn "The Eskimo!" @ Developer Technical Support @ Apple
`let myEmail = "eskimo" + "1" + "@" + "apple.com"`

Posted 3 hours ago by eskimo

0 Hey Quinn, I know the forum name changed again, but it's still me. This post really really made my day! Thanks, Josh — jamfshome 4 minutes ago

Add a Comment

Replies

> It would seem that SecurityAgent has some anti-debug protection in it now as I can't connect to the running instance with lldb.

Correct. In addition to the protection provided by SIP, it also calls `PT_DENY_ATTACH` (see the `ptrace` man page for more on that).

> Any ideas?

Some thoughts:

- As a (primarily) network developer, I think it's unfair to characterise logging as "caveman debugging". There are a bunch of problems that can only be solved that way, especially problems that are only reproducible in the field. When plugging in to odd parts of the system I recommend that you spend some time making sure your logging is up to scratch for exactly that reason. If that helps with your day-to-day debugging, all the better.
- However, I recognise that there are some times when you want to just step through the code. I think it reasonable that macOS should support this for authorisation plug-ins, and I recommend you file a bug report along those lines.
  Please post your bug number, just for the record.
- When confronted with this problem my usual approach is to build infrastructure such that I can test (and hence debug) the vast majority of my code in a normal runtime environment. This means that I only have to worry about debugging the differences between the real environment and my test environment, a much more limited problem.
- In many cases logging is the right way to debug those differences. In the rare situations where it's not, I created a nasty hack to get the debugger working. Alas, I can't share that.

Share and Enjoy
—
Quinn "The Eskimo!"
Apple Developer Relations, Developer Technical Support, Core OS/Hardware
`let myEmail = "eskimo" + "1" + "@apple.com"`

Posted 4 years ago by eskimo

Add a Comment

Hi Quinn,

No offence intended with the debugging remark! It just feels akward to not be able to use the debugger. If I'm ever able to file a radar (Access to bug reporter is broken case: 10037304707 5) I'll be sure to post my numbers on the remote debugging back here.

Thanks,

Josh

Posted 4 years ago by NoMAD-Dev

Add a Comment

Hey Quinn,

I filed two related radars:

38755732 (Xcode has no way to Debug Authorization Plug-ins)

38756363 (Xcode Doesn't Support Remote Debugging on macOS)

Thanks!

Josh

Posted 4 years ago by NoMAD-Dev

Add a Comment

Someone pinged me about this during an Ask DTS session yesterday so I thought I'd post a quick update. In recent releases we've tweaked the security model of both macOS and authorisation plug-ins considerably. The good news is that this has *improved* the debugging story.

To debug a plug-in on 10.15 and later:

1. Set up a 'victim' machine. In some cases you may need to use real hardware, but in most cases you can get away with using a VM.
2. Disable System Integrity Protection (SIP).
   WARNING Disabling SIP reduces the security of your Mac. Do not disable SIP on a machine you use day-to-day. Rather, only disable SIP on a victim machine.
3. SSH into the victim machine.
4. Run LLDB as root:

```
% sudo lldb
```

5. Attach to the process that's hosting the plugin:

```
(lldb) process attach -p 1729
```

It's best to target the process using its process ID rather than name. It's not uncommon for there to be multiple instances of the authorization plug-in host process running concurrently.

The process hosting your plug-in varies by plug-in configuration and OS version:

macOS	Architecture	Non-Privileged	Privileged
...<10.14	Intel	`SecurityAgent`	`authorizationhost`
10.14...<11.0	Intel	`SecurityAgentHelper`	`authorizationhosthelper`
11.0...	Intel	`SecurityAgentHelper-x86_64`	`authorizationhosthelper.x86_64`
	Apple silicon	`SecurityAgentHelper-arm64`	`authorizationhosthelper.arm64`

Share and Enjoy
—
Quinn "The Eskimo!" @ Developer Technical Support @ Apple
`let myEmail = "eskimo" + "1" + "@" + "apple.com"`

Posted 3 hours ago by eskimo

0 Hey Quinn, I know the forum name changed again, but it's still me. This post really really made my day! Thanks, Josh — jamfshome 4 minutes ago

Add a Comment

Your Reply

Enter your reply here

0 / 7000

Live Preview Submit

This site contains user submitted content, comments and opinions and is for informational purposes only. Apple disclaims any and all liability for the acts, omissions and conduct of any third parties in connection with or related to your use of the site. All postings and use of the content on this site are subject to the Apple Developer Forums Participation Agreement.

Copyright © 2022 Apple Inc. All rights reserved.