Beyond the good ol' LaunchAgents - 13 - Audio Plugins

theevilbit.github.io/beyond/beyond_0013

April 19, 2021

> This is part 13 in the series of "Beyond the good ol' LaunchAgents", where I try to collect various persistence techniques for macOS. For more background check the introduction.

This is another one of my favorites for some reason. macOS being a popular audio editing device, supports external audio drivers and plugins. @xorrior wrote a very extensive blog post about these at his website, here: Audio Unit Plug-ins. Legitimate Un-signed Code Execution | by Christopher Ross | Posts By SpecterOps Team Members

It is pretty amazing, and I don't plan to repeat what it's there but show another way of implementing a plugin. I will briefly describe how to create a Hardware Abstraction Layer (HAL) Service plug-in.

Apple has a nice documentation as well: Core Audio Overview

HAL plugins are loaded by `coreaudiod`, which runs as root, and defined at `/System/Library/LaunchDaemons/com.apple.audio.coreaudiod.plist`. HAL plugins are located at `/Library/Audio/Plug-Ins/HAL`, which means that we require root access to install them.

We can easily create such a plugin with Xcode, by creating a new project, and selecting the type as `bundle`. For the bundle extension we need to provide `.driver`. Once the project is created we need to add a new source file.

```
#import <Foundation/Foundation.h>

__attribute__((constructor)) static void run()
{
    NSLog(@"%@", @"Hello from MasterAudio");
}
```

We can implement a simple `constructor` which will be loaded when the driver is loaded. We don't really need to do anything else. This will work just fine, and won't crash `coreaudiod`.

In the `Info.plist` we also need to add some UUID, and default function name, but we don't need to implement it.

```
<key>CFPlugInFactories</key>
<dict>
    <key>00000000-0000-0000-0000-000000000000</key>
    <string>SomeFunction</string>
</dict>
<key>CFPlugInTypes</key>
<dict>
    <key>00000000-0000-0000-0000-000000000000</key>
    <array>
        <string>00000000-0000-0000-0000-000000000000</string>
    </array>
</dict>
```

Next we can simply compile it and place it in `/Library/Audio/Plug-Ins/HAL`. We will need to set folder ownership to `root:wheel`.

We can then go and restart `coreaudiod`.

```
csaby@dev ~ % sudo launchctl stop com.apple.audio.coreaudiod
csaby@dev ~ % sudo launchctl start com.apple.audio.coreaudiod
```

Now we can query the logs.

```
csaby@dev ~ % log show --predicate 'eventMessage contains[c] "MasterAudio"' --last
20m
Filtering the log data using "composedMessage CONTAINS[c] "MasterAudio""
Skipping info and debug messages, pass --info and/or --debug to include.
Timestamp                       Thread     Type        Activity             PID
TTL
2021-04-19 22:39:55.144111+0200 0x18e2     Default     0x0                  586    0
coreaudiod: (MasterAudio) Hello from MasterAudio
```

As we can see our driver was loaded, and it will run as root.

This was just one option, Chris's blog (linked above) details a few more.