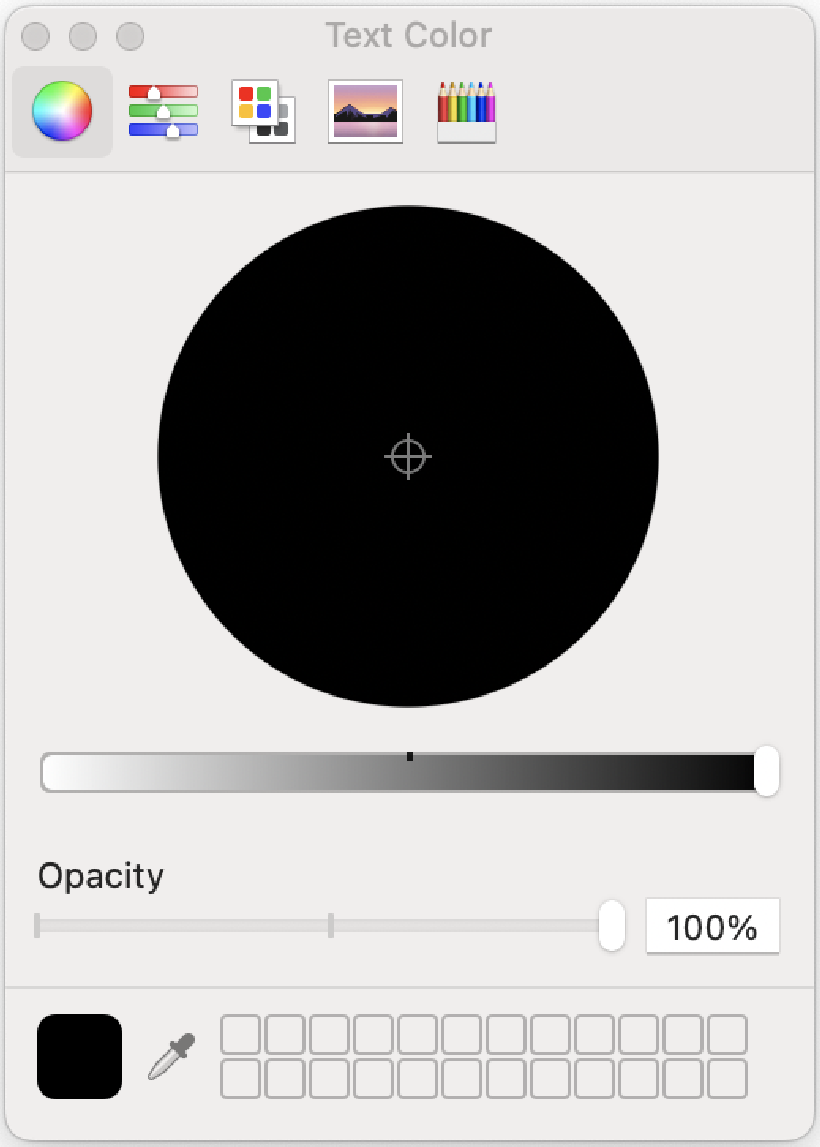
Beyond the good ol' LaunchAgents - 17 - Color Pickers

theevilbit.github.io/beyond/beyond_0017

May 31, 2021

> This is part 17 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.

Color pickers??? It's this menu, where we can select a color:



To my surprise we can install our own color pickers on the system, and add custom ones. There are quite a few of these, some are even open source, like this: GitHub - viktorstrate/color-picker-plus: An Improved Color Picker for macOS. Github is full of color picker codes, mostly written in Swift. We don't really need those, unless we want to be stealthy and hide our code inside a legitimate picker.

Color pickers can be installed in two locations:

- `/Library/ColorPickers` - Here we can install global color pickers, and it requires root access
- `~/Library/ColorPickers` - Here we can install user specific color pickers

Similarly to screen savers, color pickers are just bundles with the `.colorPicker` extension. Xcode doesn't have a template for it, but we can simply create an empty bundle with the right extension and add a constructor function.

Once we place it in the right directory, if we run anything that loads a color picker, our code will be loaded. I find it really useful, as we don't rely on any plist files, and we can expect it to run frequently. For example simply creating a new document in Pages will trigger it.

Then our log message comes:

```
csaby@mac ~ % log stream | grep hello_color
2021-05-28 22:57:43.718561+0200 0x1077d4   Default     0x0                  32246  0
LegacyExternalColorPickerService-x86_64: (DemoColor) hello_color void custom(int, const char **)
```

Color picker plugins are loaded by the process `/System/Library/Frameworks/AppKit.framework/Versions/C/XPCServices/LegacyExternalColorPickerService-x86_64.xpc/Contents/MacOS/LegacyExternalColorPickerService-x86_64`.

Let's take a look at the process entitlements:

```
Executable=/System/Library/Frameworks/AppKit.framework/Versions/C/XPCServices/LegacyExternalColorPickerService
x86_64.xpc/Contents/MacOS/LegacyExternalColorPickerService-x86_64
Identifier=com.apple.appkit.xpc.LegacyExternalColorPickerService.x86_64
Format=bundle with Mach-O thin (x86_64)
CodeDirectory v=20400 size=789 flags=0x0(none) hashes=13+7 location=embedded
Platform identifier=12
Signature size=4442
Signed Time=2021. May 8. 14:45:49
Info.plist entries=23
TeamIdentifier=not set
Sealed Resources version=2 rules=2 files=0
Internal requirements count=1 size=108
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
        <key>com.apple.security.app-sandbox</key>
        <true/>
        <key>com.apple.security.cs.disable-library-validation</key>
        <true/>
        <key>com.apple.security.temporary-exception.sbpl</key>
        <array>
                <string>(deny file-write* (home-subpath "/Library/Colors"))</string>
                <string>(allow file-read* process-exec file-map-executable (home-subpath
"/Library/ColorPickers"))</string>
                <string>(allow file-read* (extension "com.apple.app-sandbox.read"))</string>
        </array>
</dict>
</plist>
```

As expected it has the `com.apple.security.cs.disable-library-validation` entitlement, which disables library validation and allows the load of third party code. It's also sandboxed, based on `com.apple.security.app-sandbox`. However there aren't many exception, not even network connection is allowed. This is a pretty locked down process, thus I think it's not the greatest place to persist, unless somehow we can escape it.