Beyond the good ol' LaunchAgents - 16 - Screen Saver

theevilbit.github.io/beyond/beyond_0016

May 30, 2021

> This is part 16 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.

Screen savers have been detailed recently by Leo Pitton his blog post: "Saving Your Access". Considering that he already wrote most of the interesting stuff, I will try to show some new information, but there will be some overlap.

Screen savers are macOS bundles with the bundle extension of `.saver`. Their `Info.plist` file doesn't contain anything screen saver specific, so we can really use any bundle with the right extension.

Screen savers can be found in 3 different locations.

- `/System/Library/Screen Savers` - contains the system embedded screen savers
- `/Library/Screen Savers` - Here we can install global screen savers, and it requires root access
- `~/Library/Screen Savers` - Here we can install screen savers for the given user

Xcode contains a screen saver template, which creates a properly named bundle and an Objective C file, which has all the functions defined which is expected by the system to be implemented by a screen saver. Let's explore them.

```
@implementation DemoScreenView

- (instancetype)initWithFrame:(NSRect)frame isPreview:(BOOL)isPreview
{

    NSLog(@"hello_screensaver %s", __PRETTY_FUNCTION__);
    self = [super initWithFrame:frame isPreview:isPreview];
    if (self) {
        [self setAnimationTimeInterval:1/30.0];
    }
    return self;
}

- (void)startAnimation
{
    NSLog(@"hello_screensaver %s", __PRETTY_FUNCTION__);
    [super startAnimation];
}

- (void)stopAnimation
{
    NSLog(@"hello_screensaver %s", __PRETTY_FUNCTION__);
    [super stopAnimation];
}

- (void)drawRect:(NSRect)rect
{
    NSLog(@"hello_screensaver %s", __PRETTY_FUNCTION__);
    [super drawRect:rect];
}

- (void)animateOneFrame
{
    NSLog(@"hello_screensaver %s", __PRETTY_FUNCTION__);
    return;
}

- (BOOL)hasConfigureSheet
{
    NSLog(@"hello_screensaver %s", __PRETTY_FUNCTION__);
    return NO;
}

- (NSWindow*)configureSheet
{
    NSLog(@"hello_screensaver %s", __PRETTY_FUNCTION__);
    return nil;
}

@end
```

Te functions are related to initialization, configuration of the screen saver and the actual drawing or animation. Here I added a log function to each, which will create a log message with the function name that called it. We can use the `__PRETTY_FUNCTION__` string to get the function name runtime, as it always holds the current function being called.

I also added a constructor:

```
__attribute__((constructor))
void custom(int argc, const char **argv)
{
    NSLog(@"hello_screensaver %s", __PRETTY_FUNCTION__);
}
```

Once we installed a screensaver, and click on it in the preview pane, we will have the following log messages generated.

```
csaby@bigsur ~ % log stream | grep hello_screensaver
2021-05-28 08:43:59.329660-0700 0x1eb6     Default     0x6b54               692    0    legacyScreenSaver: (DemoScreen)
hello_screensaver void custom(int, const char **)
2021-05-28 08:43:59.329945-0700 0x1eb6     Default     0x6b54               692    0    legacyScreenSaver: (DemoScreen)
hello_screensaver -[DemoScreenView initWithFrame:isPreview:]
2021-05-28 08:43:59.330051-0700 0x1eb6     Default     0x6b54               692    0    legacyScreenSaver: (DemoScreen)
hello_screensaver -[DemoScreenView hasConfigureSheet]
2021-05-28 08:43:59.330178-0700 0x1eb6     Default     0x6b54               692    0    legacyScreenSaver: (DemoScreen)
hello_screensaver -[DemoScreenView hasConfigureSheet]
2021-05-28 08:43:59.330981-0700 0x1eb6     Default     0x0                  692    0    legacyScreenSaver: (DemoScreen)
hello_screensaver -[DemoScreenView drawRect:]
2021-05-28 08:43:59.845980-0700 0x1eb6     Default     0x6b54               692    0    legacyScreenSaver: (DemoScreen)
hello_screensaver -[DemoScreenView startAnimation]
2021-05-28 08:43:59.846471-0700 0x1eb6     Default     0x6b54               692    0    legacyScreenSaver: (DemoScreen)
hello_screensaver -[DemoScreenView animateOneFrame]
```

What we can notice is that we are running within the process `legacyScreenSaver`. We can also find that after the constructor most of the other methods are also called, thus we could use them as well to contain our code.

The `legacyScreenSaver` binary is located at `/System/Library/Frameworks/ScreenSaver.framework/PlugIns/legacyScreenSaver.appex/Contents/MacOS/legacyScreenSaver`. Since we run in the context of this process it worth taking a look. The process's code signing information is shown below.

```
Executable=/System/Library/Frameworks/ScreenSaver.framework/PlugIns/legacyScreenSaver.appex/Contents/MacOS/legacyScreenSaver
Identifier=com.apple.ScreenSaver.Engine.legacyScreenSaver
Format=bundle with Mach-O universal (x86_64 arm64e)
CodeDirectory v=20400 size=935 flags=0x0(none) hashes=18+7 location=embedded
Platform identifier=12
Signature size=4442
Signed Time=2021. May 8. 15:02:04
Info.plist entries=27
TeamIdentifier=not set
Sealed Resources version=2 rules=2 files=0
Internal requirements count=1 size=96
<?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.private.xpc.launchd.per-user-lookup</key>
        <true/>
        <key>com.apple.security.app-sandbox</key>
        <true/>
        <key>com.apple.security.assets.pictures.read-only</key>
        <true/>
        <key>com.apple.security.cs.disable-library-validation</key>
        <true/>
        <key>com.apple.security.files.bookmarks.app-scope</key>
        <true/>
        <key>com.apple.security.files.user-selected.read-only</key>
        <true/>
        <key>com.apple.security.network.client</key>
        <true/>
        <key>com.apple.security.network.server</key>
        <true/>
        <key>com.apple.security.temporary-exception.files.absolute-path.read-only</key>
        <array>
                <string>/</string>
        </array>
        <key>com.apple.security.temporary-exception.mach-lookup.global-name</key>
        <array>
                <string>com.apple.CARenderServer</string>
                <string>com.apple.CoreDisplay.master</string>
                <string>com.apple.nsurlstorage-cache</string>
                <string>com.apple.ViewBridgeAuxiliary</string>
        </array>
        <key>com.apple.security.temporary-exception.sbpl</key>
        <array>
                <string>(allow mach-lookup mach-register)</string>
        </array>
        <key>com.apple.security.temporary-exception.yasb</key>
        <true/>
</dict>
</plist>
```

The entitlement `com.apple.security.cs.disable-library-validation` explains why it can load third party bundles, it has library validation disabled. Beyond that we find `com.apple.security.app-sandbox`, which indicates that it's sandboxed. Based on the rest of the entitlements it doesn't have too much rights, although at least it can reach the network. If we use screen savers for persistence, it's good to have a sandbox escape in our toolbox, otherwise we are quite locked.