Beyond the good ol' LaunchAgents - 29 - amstoold

theevilbit.github.io/beyond/beyond_0029

March 8, 2022

> This is part 29 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.

When doing some research on macOS I came across the following LaunchAgent: `/System/Library/LaunchAgents/com.apple.amstoold.plist`. Its content is the following.

```
<?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>EnablePressuredExit</key>
        <true/>
        <key>EnableTransactions</key>
        <true/>
        <key>Label</key>
        <string>com.apple.amstoold</string>
        <key>LaunchEvents</key>
        <dict>
                <key>com.apple.distnoted.matching</key>
                <dict>
                        <key>com.apple.nfcd.ams.accessory</key>
                        <dict>
                                <key>Name</key>
                                <string>com.apple.nfcd.ams.accessory</string>
                        </dict>
                </dict>
                <key>com.apple.notifyd.matching</key>
                <dict>
                        <key>com.apple.ams.privateListeningChanged</key>
                        <dict>
                                <key>Notification</key>

<string>com.apple.ams.privateListeningChanged</string>
                        </dict>
                </dict>
        </dict>
        <key>MachServices</key>
        <dict>
                <key>com.apple.xpc.amstoold</key>
                <true/>
        </dict>
        <key>ProcessType</key>
        <string>Background</string>
        <key>ProgramArguments</key>
        <array>
                <string>/usr/local/bin/amstoold</string>
        </array>
        <key>RunAtLoad</key>
        <true/>
</dict>
</plist>
```

If we examine it closely we can notice that the related binary is `/usr/local/bin/amstoold`. This is very interesting as this file doesn't exists on default macOS installations. This means that if we create it, it will be started as the user upon login. As the launchd file is SIP protected, we can't even remove it, and even if we would, it likely would come back at the next OS update.

Normally we would require root for installing a file in this location, however if we have `homebrew` installed, we can do that as the normal user.

Honestly this is really odd. Moreover it defines an Mach/XPC service, named `com.apple.xpc.amstoold`. This is being referenced in a couple of Sandbox profiles:

```
/System/Library/Sandbox/Profiles/com.apple.appstored.sb
52:    (global-name "com.apple.xpc.amstoold"))

/System/Library/Sandbox/Profiles/com.apple.bookdatastored.sb
101:    (global-name "com.apple.xpc.amstoold")

/System/Library/Sandbox/Profiles/frameworks.sb
328:               "com.apple.xpc.amstoold"))

/System/Library/Sandbox/Profiles/watchlistd.sb
61:                 "com.apple.xpc.amstoold"

/System/Library/Sandbox/Profiles/com.apple.appstoreagent.sb
100:    (global-name "com.apple.xpc.amstoold"))

/System/Library/Sandbox/Profiles/com.apple.bookassetd.sb
117:    (global-name "com.apple.xpc.amstoold")
```

The XPC service is being referenced by a single framework, namely `AppleMediaServices`. Based on this I guess `amstoold` stands for `Apple Media Services Tool Daemon`.

Essentially the lack of this binary would also allow us to hijack a system defined XPC service. I didn't take the time to reverse its invocation, beyond finding the XPC connection setup, which is show below.

```
int -[AMSHTTPArchiveService _createRemoteConnection](int arg0) {
    r12 = *_objc_msgSend;
    dispatch_assert_queue$V2([[arg0 queue] retain]);
    [rax release];
    rbx = r12;
    r12 = [[*0x7ff842dc4568 alloc] initWithMachServiceName:@"com.apple.xpc.amstoold"
options:0x0];
    rax = [arg0 queue];
    rax = [rax retain];
    [r12 _setQueue:rax];
    [rax release];
    r14 = [[*0x7ff842dc4570 interfaceWithProtocol:*0x7ff842dc40a0] retain];
    [r14 setClass:objc_opt_class(@class(AMSHTTPArchiveTaskInfo))
forSelector:@selector(recordTrafficWithTaskInfo:) argumentIndex:0x0 ofReply:0x0];
    [r12 setRemoteObjectInterface:r14];
    objc_initWeak(&var_30, arg0);
    *(&var_60 - 0x20) = *__NSConcreteStackBlock;
    *(&var_60 - 0x18) = 0xffffffffc2000000;
    *(&var_60 - 0x10) = sub_7ff817786529;
    *(&var_60 - 0x8) = 0x7ff8415511f8;
    objc_copyWeak(&var_60, &var_30);
    [r12 setInvalidationHandler:&var_80];
    *(&var_38 - 0x20) = *__NSConcreteStackBlock;
    *(&var_38 - 0x18) = 0xffffffffc2000000;
    *(&var_38 - 0x10) = sub_7ff8177866b9;
    *(&var_38 - 0x8) = 0x7ff8415511f8;
    objc_copyWeak(&var_38, &var_30);
    [r12 setInterruptionHandler:&var_58];
    [r12 resume];
    objc_destroyWeak(&var_38);
    objc_destroyWeak(&var_60);
    objc_destroyWeak(&var_30);
    [r14 release];
    rax = [r12 autorelease];
    return rax;
}
```

I think it would an interesting future research area, what could we get access to by implementing this service ourselves.

Until then, simply drop this binary and you can persist as the user.

If you know more about this service, let me know and I would happily update this post.