# Beyond the good ol' LaunchAgents - 28 - Authorization Plugins

theevilbit.github.io/beyond/beyond_0028

February 9, 2022

> This is part 28 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 persistence mechanism was described in very detail by Chris Ross in his blogpost: Persistent Credential Theft with Authorization Plugins. He also developed sample code, which can be found on his GitHub. Thus this blog will only focus on the high level summary, and some changes that happened since he wrote that post.

On a high level, `authorization plugins` are extensions that can be used by the system to grant specific authorization rights.

For installation we need to perform two steps. First is adding the plugin bundle to the directory `/Library/Security/SecurityAgentPlugins/`. Second we need to modify the authorization database and add an entry so that our plugin is loaded and invoked as needed.

For example I made a `testAuthPlugin` based on Chris's code, and used his install script to place it under the `system.login.console` right. The plugin is added under the mechanisms, as `testAuthPlugin:login,privileged`.

```
csaby@mantarey ~ % security authorizationdb read system.login.console
<?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>class</key>
        <string>evaluate-mechanisms</string>
        <key>comment</key>
        <string>Login mechanism based rule.  Not for general use, yet.</string>
        <key>created</key>
        <real>657182100.19664896</real>
        <key>mechanisms</key>
        <array>
                <string>builtin:policy-banner</string>
                <string>builtin:prelogin</string>
                <string>loginwindow:login</string>
                <string>builtin:login-begin</string>
                <string>builtin:reset-password,privileged</string>
                <string>loginwindow:FDESupport,privileged</string>
                <string>builtin:forward-login,privileged</string>
                <string>builtin:auto-login,privileged</string>
                <string>builtin:authenticate,privileged</string>
                <string>PKINITMechanism:auth,privileged</string>
                <string>builtin:login-success</string>
                <string>loginwindow:success</string>
                <string>HomeDirMechanism:login,privileged</string>
                <string>HomeDirMechanism:status</string>
                <string>testAuthPlugin:login,privileged</string>
                <string>MCXMechanism:login</string>
                <string>CryptoTokenKit:login</string>
                <string>loginwindow:done</string>
        </array>
        <key>modified</key>
        <real>665844824.116117</real>
        <key>shared</key>
        <true/>
        <key>tries</key>
        <integer>10000</integer>
        <key>version</key>
        <integer>8</integer>
</dict>
</plist>
YES (0)
```

After a logoff and login, the plugin will be loaded.

What I would like to highlight is that the process this plugins runs inside was changed. In the past it was running inside `authorizationhost`. If we check the entitlements of this process, we can see that it would be problematic.

```
Executable=/System/Library/Frameworks/Security.framework/Versions/A/MachServices/authc

Identifier=com.apple.authorizationhost
Format=bundle with Mach-O universal (x86_64 arm64e)
CodeDirectory v=20400 size=2388 flags=0x0(none) hashes=64+7 location=embedded
Platform identifier=13
Signature size=4442
Signed Time=2021. Dec 20. 1:50:34
Info.plist entries=22
TeamIdentifier=not set
Sealed Resources version=2 rules=2 files=0
Internal requirements count=1 size=76
[Dict]
        [Key] com.apple.ahp
        [Value]
                [Bool] true
        [Key] com.apple.keystore.device
        [Value]
                [Bool] true
        [Key] com.apple.keystore.console
        [Value]
                [Bool] true
        [Key] com.apple.keystore.filevault
        [Value]
                [Bool] true
        [Key] com.apple.security.smartcard
        [Value]
                [Bool] true
        [Key] com.apple.authkit.client.private
        [Value]
                [Bool] true
        [Key] com.apple.keystore.device.verify
        [Value]
                [Bool] true
        [Key] com.apple.keystore.domain.select
        [Value]
                [Bool] true
        [Key] com.apple.authorization.extract-password
        [Value]
                [Bool] true
        [Key] com.apple.private.applecredentialmanager.allow
        [Value]
                [Bool] true
        [Key] com.apple.private.security.clear-library-validation
        [Value]
                [Bool] true
        [Key] com.apple.private.opendirectoryd.registerexternalauth
        [Value]
                [Bool] true
        [Key] com.apple.private.configurationprofiles.bootstraptoken.readonly
        [Value]
                [Bool] true
        [Key] com.apple.private.tcc.allow
        [Value]
```

```
                [Array]
                        [String] kTCCServiceSystemPolicyNetworkVolumes
```

The plugin would access all of these entitlements, and for example by accessing `kTCCServiceSystemPolicyNetworkVolumes` is a TCC bypass straight away. Apple has changed this (maybe someone reported this), and now the plugin is loaded by an XPC service, `authorizationhosthelper.x86_64`, which lacks all of these rights.

```
Executable=/System/Library/Frameworks/Security.framework/Versions/A/MachServices/authc

Identifier=com.apple.authorizationhosthelper.x86_64
Format=bundle with Mach-O thin (x86_64)
CodeDirectory v=20400 size=1281 flags=0x0(none) hashes=29+7 location=embedded
Platform identifier=13
Signature size=4442
Signed Time=2021. Dec 20. 1:49:47
Info.plist entries=21
TeamIdentifier=not set
Sealed Resources version=2 rules=2 files=0
Internal requirements count=1 size=88
[Dict]
        [Key] com.apple.security.smartcard
        [Value]
                [Bool] true
        [Key] com.apple.private.security.clear-library-validation
        [Value]
                [Bool] true
```

This is similar to what we see in other processes used by Apple, like `coreaudiod`, loading of external plugins is done by an XPC service.

We can confirm this by checking the logs, for example my `testAuthPlugin` logs a simple message.

```
2022-02-06 13:55:48.739099+0100 0x3e25     Default     0x0                  1151   0
authorizationhosthelper.x86_64: (testAuthPlugin) testAuthPlugin was executed
```