Beyond the good ol' LaunchAgents - 25 - Apache2 modules

theevilbit.github.io/beyond/beyond_0025

December 15, 2021

> This is part 25 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.

Possibly a less known feature that macOS has a built-in Apache2 web server, which can be enabled anytime. Just as other Apache2 servers, it also supports the load of custom modules, and this is what we will explore here briefly for persistence. For a detailed web server setup, I recommend the following articles: Setting up a local web server on a Mac - Apple Community Setting up a local web server on macOS 12... - Apple Community

The Apache configuration files can be found under `/etc/apache2/`, and the one we need to edit for our goal is `/etc/apache2/httpd.conf`. Here we have an example line which specifies how to configure a custom Apache module.

```
# LoadModule my_custom_module /usr/local/libexec/mod_my_custom.so "My Signature Authority"
```

The name is arbitrary, then we have the path to the file and finally the code signer of the module. Unfortunatley adhoc signing is not sufficient as described here. So we need to take our custom dylib, and sign it with either an Apple developer ID or a certificate, which is trusted by the system (we can install our own trusted cert).

Here I just used my own:

```
LoadModule my_custom_module /Users/Shared/example.dylib "Apple Development: Csaba Fitzl (RQGUDM4LR2)"
```

This line will cause Apache to load `/Users/Shared/example.dylib`, when it's started. We can start Apache by running the following command. This will persist across reboots, so we actually achieve persistence.

```
sudo launchctl load -w /System/Library/LaunchDaemons/org.apache.httpd.plist
```

This will call `/usr/sbin/httpd-wrapper`, which is a ruby script and which will eventually start `/usr/sbin/httpd`.

If we inspect the entitlements of `httpd`, we can find that it has the `com.apple.private.security.clear-library-validation` entitlement, which will allow it to load not Apple signed binaries. I wrote about this entitlement before in more details on my blog.

```
Executable=/usr/sbin/httpd
Identifier=com.apple.apache.httpd
Format=Mach-O universal (x86_64 arm64e)
CodeDirectory v=20400 size=6799 flags=0x0(none) hashes=202+7 location=embedded
Platform identifier=13
Signature size=4442
Signed Time=2021. Nov 13. 10:52:58
Info.plist entries=6
TeamIdentifier=not set
Sealed Resources=none
Internal requirements count=1 size=72
[Dict]
        [Key] com.apple.private.security.clear-library-validation
        [Value]
                [Bool] true
        [Key] com.apple.private.responsibility.set-to-self.at-launch
        [Value]
                [Bool] true
```

As for the Apache module dylib, we can simply make one with a constructor, like this:

```
#include <stdio.h>
#include <syslog.h>

__attribute__((constructor))
static void myconstructor(int argc, const char **argv)
{
     printf("[+] dylib constructor called from %s\n", argv[0]);
     syslog(LOG_ERR, "[+] dylib constructor called from %s\n", argv[0]);
}
```

Potentially other Apache configuration files also allow us to load custom modules, but I didn't explore that option.