Author: Flash
When I was playing with using debugserver I was having problems where requests I had made days ago were cropping up again and again. After some investigation I found nsurlsessiond was the cause of this problem. It starts at boot and starts a bunch of 'tasks' where each task is an HTTP request. The list of tasks and where they are stored is described below. It appears to be that tasks are added to the list when a failed connection is made; although I have not verified this.

Having done a little more reversing (but not enough yet), I'm fairly sure I can use this plist to launch apps in the background (not shown to the user in anyway). Calling the eventual function ('SBSLaunchApplicationWithIdentifierAndLaunchOptions') myself, this was enough to start the music app which I had modifed the '/Library/MusicUISupport/js/index.js' file in order to gain javascript execution. I'm fairly sure that with a bit more reversing this can be used as a persistence launch mechanism as it would trigger loading of (at least) apps at boot.

Modification of the tasks.plist to use a 'file://...' url over 'http://' would allow it to always complete the url request.

As I ran out of time to finish the reversing of this, I've put all relevant files in the output share along with a brief description of my test process.

/var/mobile/Library/com.apple.nsurlsessiond/

Contents:

- bundles.plist - Serialised NS objects naming processes with nsurl sessions. My version contains: com.apple.OTACrashCopier, com.apple.mobileassetd and com.apple.tipsd.
- A folder for each of the three above, with the name e.g. SHA1('com.apple.OTACrashCopier') (88FA46B3255990F9B0FE64762DE1A1751F240BC9)

88FA46B3255990F9B0FE64762DE1A1751F240BC9/:

Contents:

- sessions.plist - Serialised NS objects naming sessions within this process. e.g. com.apple.MobileAsset.asset-metadata-downloads:com.apple.languageassetd and com.apple.MobileAsset.asset-metadata-downloads. For crash copier, the session is also called com.apple.OTACrashCopier
- A folder for each of the above sessions, with the name e.g. SHA1('com.apple.OTACrashCopier') (88FA46B3255990F9B0FE64762DE1A1751F240BC9)

88FA46B3255990F9B0FE64762DE1A1751F240BC9)/88FA46B3255990F9B0FE64762DE1A1751F240BC9)/:

Contents:

- An empty Upload folder - Probably just happens to be empty in my examples
- configuration.plist - More serialised NS objects. Including the serialised __NSCFURLSessionConfiguration which allows fine grain control of the url request, the user agent and some ns data.
- options.plist - More serialised NS objects. Not 100% sure what for.
- tasks.plist - The most interesting file. NS serialised objects with lots of control of the connections required. This includes urls, http headers, request type, http body and user agents.

Potential uses:

- Trigger calls out to a webserver every boot (call js?). Not sure what happens on completion of request.
- Poor mans implant. Create tasks.plist that is readable by nsurlsessiond but not deletable/writable. Set up a bunch of upload requests that will be triggered by nsurlsessiond every boot and possible every n minutes.
- Persistence via attempt to trigger SALINE bug as it deserialises the NSKeyeredArchiver class in each of these plists. ASLR could be defeated by brute force.
- Can be used to launch applications

DOCUMENT INFO

TAGS

RELATED

COMMENTS

HISTORY

unauthenticated...	7/31/2015 at 2:34 PM
unauthenticated...	7/29/2015 at 3:26 PM
unauthenticated...	7/29/2015 at 3:11 PM
unauthenticated...	7/29/2015 at 2:40 PM