Beyond the good ol' LaunchAgents - 11 - Spotlight Importers

theevilbit.github.io/beyond/beyond_0011

April 3, 2021

> This is part 11 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.

TL;DR

It works, but very limited due to heavy sandboxing, you can only read and copy files to your sandbox folder or consume some CPU power. If you have a way to escape sandbox then go for it, or could be used as part of a multi-part malware.

Intro

I'm reading Jonathan Levin's *OS Internals Vol I. book (user mode - *OS Internals: - Welcome!), and I got to the chapter where he talks about Spotlight importers, and my first thought was that it would be an awesome way to persist on macOS. Typically there is nothing new in InfoSec, so after a quick Google search I found that Patrick Wardle already mentioned this in his BlackHat USA / Immunity talk back in 2015. Slides: https://www.blackhat.com/docs/us-15/materials/us-15-Wardle-Writing-Bad-A-Malware-For-OS-X.pdf Talk: Patrick Wardle Writing Bad@ss OS X Malware on Vimeo (This is the Immunity one, as he cut this part from the actual BH talk) But I didn't find anything beyond this, nothing about how to persist this way. So I decided to experiment with it and see what can or cannot be done.

What are Spotlight importers?

Spotlight on OSX / macOS is basically an indexing / search service. When you press CMD+SPACE a searcher comes up (that's Spotlight) and you can type in whatever you look for. Spotlight will search in its index, and that index is being built by Spotlight Importers. These importers are made for various filetypes, typically documents, which will be able to parse the file and extract useful, indexable content from it. There are a bunch of these included in macOS, but you can also write your own, and extend the system's capabilities. You can check your current list of importers with `mdimport -L` command. This is how my system looks like:

```
csaby@mac scripts % mdimport -L
Paths: id(501) (
    “/Library/Spotlight/iBooksAuthor.mdimporter”,
    “/System/Library/Spotlight/SystemPrefs.mdimporter”,
    “/System/Library/Spotlight/Chat.mdimporter”,
    “/System/Library/Spotlight/iWork.mdimporter”,
    “/System/Library/Spotlight/iPhoto.mdimporter”,
    “/System/Library/Spotlight/PDF.mdimporter”,
    “/System/Library/Spotlight/RichText.mdimporter”,
    “/System/Library/Spotlight/Office.mdimporter”,
    “/System/Library/Spotlight/PS.mdimporter”,
    “/System/Library/Spotlight/MIDI.mdimporter”,
    “/System/Library/Spotlight/Archives.mdimporter”,
    “/System/Library/Spotlight/Audio.mdimporter”,
    “/System/Library/Spotlight/iPhoto8.mdimporter”,
    “/System/Library/Spotlight/Automator.mdimporter”,
    “/System/Library/Spotlight/Application.mdimporter”,
    “/System/Library/Spotlight/Font.mdimporter”,
    “/System/Library/Spotlight/Mail.mdimporter”,
    “/System/Library/Spotlight/QuartzComposer.mdimporter”,
    “/System/Library/Spotlight/vCard.mdimporter”,
    “/System/Library/Spotlight/Image.mdimporter”,
    “/System/Library/Spotlight/iCal.mdimporter”,
    “/System/Library/Spotlight/CoreMedia.mdimporter”,
    “/Applications/Microsoft Outlook.app/Contents/Library/Spotlight/Microsoft Outlook
Spotlight Importer.mdimporter”,

“/Applications/Evernote.app/Contents/Library/Spotlight/EvernoteSpotlightImporter.mdimp

    “/Applications/Xcode.app/Contents/Library/Spotlight/uuid.mdimporter”
)
```

You can create a Spotlight Importer and basically put it into one of these locations:

1. The `/Library/Spotlight` directory
2. The `~/Library/Spotlight` directory
3. Inside an application bundle. This bundle can be almost anywhere, I tried `Desktop`, `Downloads`, `/Applications` and all worked - most of the time. The `/System/Library/Spotlight` contains the built in importers and as it’s protected by SIP you can’t add/modify/delete these.

Creating an .mdimporter

Basics

I’m not a developer in general and definitely not a macOS developer, although I had a tiny exposure in the past, I’m far from being able to code anything. Apple’s most recent article on the subject is 6 years old: Writing a Spotlight Importer If you Google for how to create Spotlight Importers, all of the articles will say that open Xcode, and create a Spotlight Importer project, there is a template for it. NOPE, it’s gone. It was there up-until Xcode 9,

and right now we are at Xcode 11, so you either download the old one, or look for an existing project to modify. I did the second, searched Github, and eventually found several projects, and I chose this: GitHub - GenjiApp/EPUB-Plugins: OS X Spotlight / Quick Look plugins for EPUBs It's pretty old, but works. First I just compiled it to see if this importer is working at all or not, and it did. It indexed ePub files, with extra attributes, like Author.

> You can use the `mdls /path/to/file` command to view attributes of the file.

What I did after that is cleaning up the project, I removed the other targets (like Quicklook) and also cleaned up the metadata generator function. Here are the mandatory building blocks your importer will have to have, I will try to explain them to the best of my understanding, but please bear in mind that I'm a n00b for Objective-C or macOS development. If you want to follow here is the one I made / modified: macos/PersistentImporter at master · theevilbit/macos · GitHub

1. schema.xml - this will contain the attributes your importer will support. The attributes are the metadata buckets your importer can extract data into, and which will be indexed by Spotlight. For the above project I actually made it mostly intact, but I could likely clean it completely.

```
<?xml version=“1.0” encoding=“UTF-8”?>

<schema version=“1.0” xmlns=“http://www.apple.com/metadata”
xmlns:xsi=“http://www.w3.org/2001/XMLSchema-instance”
xsi:schemaLocation=“http://www.apple.com/metadata
file:///System/Library/Frameworks/CoreServices.framework/Frameworks/Metadata.framework


  <!— “attributes” element must exist in schema.xml file even if mdimporter doesn’t
support any custom attributes. if it doesn’t exist, the metadata which are listed by
“displayattrs” elements aren’t displayed by Finder’s “Get Info” panel (in OS X
10.10.2). —>
  <attributes>
    <attribute name=“com_csaby_persist_mdimporter” multivalued=“false”
type=“CFString”/>
  </attributes>

  <types>
    <type name=“org.idpf.epub-container”>

      <allattrs>
        kMDItemTitle
        kMDItemAuthors
        kMDItemKeywords
        kMDItemDescription
        kMDItemHeadline
        kMDItemPublishers
        kMDItemOrganizations
        kMDItemContributors
        kMDItemIdentifier
        kMDItemLanguages
        kMDItemCoverage
        kMDItemCopyright
        kMDItemRights
        kMDItemTextContent
        kMDItemNumberOfPages
        kMDItemContentCreationDate
        kMDItemContentModificationDate
        com_csaby_persist_mdimporter
      </allattrs>

      <displayattrs>
        kMDItemTitle
        kMDItemAuthors
        kMDItemContributors
        kMDItemPublishers
        kMDItemCopyright
        kMDItemLanguages
        kMDItemNumberOfPages
        kMDItemKeywords
        com_csaby_persist_mdimporter
        kMDItemIdentifier
        kMDItemContentCreationDate
        kMDItemContentModificationDate
        kMDItemDescription
```

```
      </displayattrs>

    </type>
  </types>
</schema>
```

2. main.c file - This is a standard file, typically generated by Xcode, you don't need to modify it. I will not paste it here, as it's quite long, but you can get it from my GitHub project (link above).
3. Info.plist file - Here you must define (along some other standard things) what kind of files your importer will support. It should contain file extensions, and the Uniform Type Identifier. Below is the one related to epub from the Info.plist. As mentioned by Patrick you can also specify all files, but I opted to stick with epubs, I didn't want my importer to run always, as my goal was only experimenting.

```
<key>UTImportedTypeDeclarations</key>
<array>
        <dict>
                <key>UTTypeConformsTo</key>
                <array>
                        <string>public.data</string>
                        <string>public.item</string>
                        <string>public.composite-content</string>
                        <string>public.content</string>
                </array>
                <key>UTTypeIdentifier</key>
                <string>org.idpf.epub-container</string>
                <key>UTTypeTagSpecification</key>
                <dict>
                        <key>public.filename-extension</key>
                        <array>
                                <string>epub</string>
                        </array>
                        <key>public.mime-type</key>
                        <array>
                                <string>application/epub+zip</string>
                        </array>
                </dict>
        </dict>
</array>
```

4. GetMetadataForFile.m file, which will be the brain of your importer, here you actually define how the content is parsed, and how metadata is extracted. For that you need to implement the following function (and eventually return true at the end):

```
Boolean GetMetadataForFile(void *thisInterface, CFMutableDictionaryRef attributes,
CFStringRef contentTypeUTI, CFStringRef pathToFile)
```

I started to experiment what I can do here, and it quickly turned out that this importer is heavily sandboxed. The only thing you can really do is read the actual file that Spotlight wants you to get metadata from, and write to your temp folder. That's pretty much it, no

access to any other random file, no network, can’t start other apps, so pretty much locked down. With that, my importer will copy every epub to a temp folder, I have also a network test included, just to show that it fails.

```
#include <CoreFoundation/CoreFoundation.h>
#import <Cocoa/Cocoa.h>

Boolean GetMetadataForFile(void *thisInterface, CFMutableDictionaryRef attributes,
CFStringRef contentTypeUTI, CFStringRef pathToFile);

//==============================================================================
//
//      Get metadata attributes from document files
//
//      The purpose of this function is to extract useful information from the
//      file formats for your document, and set the values into the attribute
//  dictionary for Spotlight to include.
//
//==============================================================================

Boolean GetMetadataForFile(void *thisInterface, CFMutableDictionaryRef attributes,
CFStringRef contentTypeUTI, CFStringRef pathToFile)
{
    NSLog(@“Hello from peristent mdimporter by csaby :)”);

    NSString* tempDir = NSTemporaryDirectory();
    NSString* tempDirFolder = [tempDir
stringByAppendingPathComponent:@“TestPersist”];
    NSString *source = (__bridge NSString *)pathToFile;
    NSString *theFileName = [[source lastPathComponent]
stringByDeletingPathExtension];
    NSString *destination = [tempDirFolder
stringByAppendingPathComponent:theFileName];
    NSError *error;
    NSFileManager *fileManager = [NSFileManager defaultManager];

    //create temp folder
    BOOL fileOK = [[NSFileManager defaultManager] createDirectoryAtPath:tempDirFolder
withIntermediateDirectories:NO attributes:nil error:&error ];
    if ( !fileOK )
        NSLog(@“createDirectoryAtPath %@“, [error localizedDescription]);

    //copy file
    if ([fileManager copyItemAtPath:source toPath:destination error:&error]){
        NSLog(@“Copy Success from: %@, to %@“, source, destination);
    }
    else{
        NSLog(@"Copy error: %@", error);
    }

    //network test
    NSString *URLString = [NSString stringWithContentsOfURL:[NSURL
URLWithString:@"https://www.google.com"]];
    NSLog(@"URL: %@", URLString);

    return true;
}
```

As it turned out during my research, since OSX 10.8 you can't use the standard `/tmp/` folder, as you have no access to that, you need to query your sandboxed `tmp` directory, you can do that with `NSTemporaryDirectory()`. More on that:

creating temp files or temp folders in standard temp file locations in mdimporter on Mac OS X 10.8.3 | Cocoabuilder

objective c - Creating temp files in Spotlight Module / MDimporter - Stack Overflow

Installation

Once you build it, you get an `.mdimporter` plugin. You should place it into one of the `Spotlight` directories, and eventually after a few minutes it will be discovered by the system and it will show up in the list of your importers. You can also force an import with `mdimport -r /path/to/your/mdimporter` but it still have to live in the right place otherwise it won't be used / imported. Once that's done, you can force an indexing with running `mdimport /path/to/epub`. If you check the device log, you will see something like this:

```
23:10:35.858903+0100    mdworker        Hello from peristent mdimporter by csaby :)
23:10:35.858963+0100    mdworker        Hello from peristent mdimporter by csaby :)
23:10:35.859467+0100    mdworker        Copy Success from: /System/Volumes/Data/Users/csaby/Downloads/pg60610-images.epub, to /var/folders/k1/_8qq2g3j3ydfp5z1vc412jdh0000gn/T/TestPersist/pg60610-images
23:10:35.862407+0100    mdworker        URL: (null)
```

The `URL:(null)` indicates the failed network connection. You can also see that the copy of the file was successful.

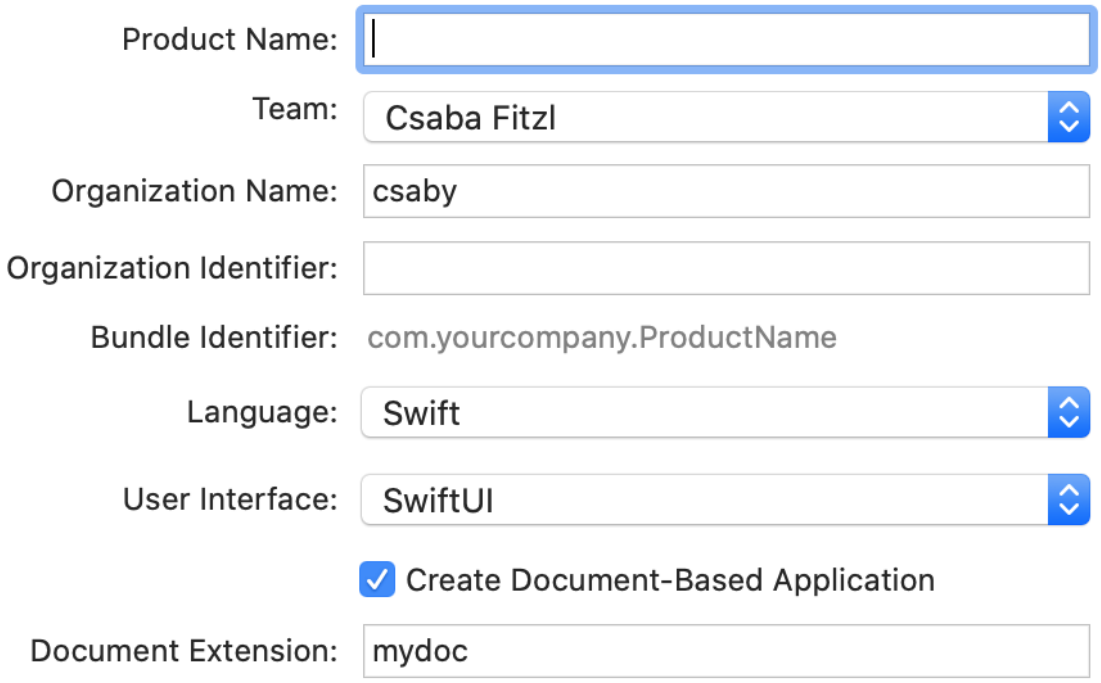
There is really nothing much more your importer can do, I tried it also on Yosemite, but essentially the same results. Actually even worse, as likely I need to do the file copy differently.

Side track: if you need to install older macOS VMs on Fusion, here you can find the links to the installer packages from 10.10 (Yosemite) to 10.15 (Catalina): Redownload Archived macOS Installers to Address Expired Certificates - TidBITS Once you download the installer, run it, and it will extract the actual OS installer to `/Applications`, which can be used by VMware Fusion. As an addition to Yosemite: Use USB 2.0 as the last security update will mess up VMware, and neither the keyboard nor the mouse will work. More on the bug: Solved: OS X 10.10.5 Yosemite VM freezes after ... |VMware Communities

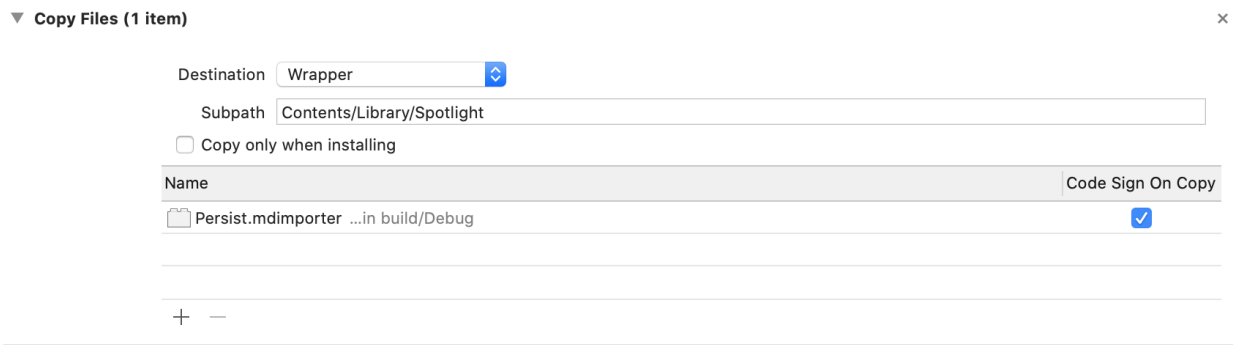
Adding an App bundle

As we saw earlier another way to install an importer is part of an `.app` bundle. This was super tricky, and it required heavy Google-Fu from my side to figure out :) The above GitHub project already has it, but here is a short writeup how to do it on your own.

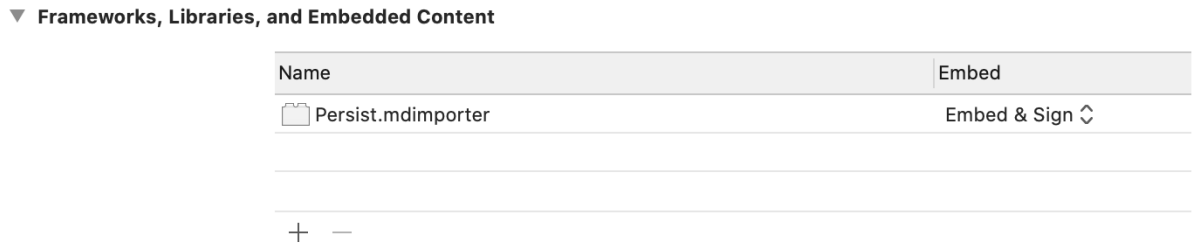
1. Add a new target to your Xcode project, and select a Cocoa app for it. Be sure to tick `Create Document-Based Application`. Ideally the extension should be the same what your importer supports - *I THINK* - but to be on the safe side do the same. The language is not important as we won't do anything.



2. On your new target go to `Build Phases`, and then select `Editor -> Add Build Phase -> Add Copy Files Build Phase`, and set it like this: Destination: `Wrapper` Subpath: `Contents/Library/Spotlight` Add your importer below.



3. Go back to General tab, and if not already there, also add your importer.



4. First build your mdimporter and then the App. The app should have the importer embedded.

If you copy that app now somewhere, most likely it will be consumed and the mdimporter will be added to the list of your available Spotlight Importers.

Download - unzip - RCE? - NOPE

Since the app bundle seems to be auto-parsed just like in the case of URL handlers, and the Spotlight importer is auto registered, I thought that someone could gain RCE with a drive by download, as Safari will auto-extract the ZIP file, which could contain an application that could contain an importer. Luckily it seems that macOS won't import a Spotlight importer in this case. Hallelujah! :)

Gatekeeper on Catalina

I also tried what happens if I download an mdimporter and manually place it into the Spotlight folder. It will be imported in that case, however due to the quarantine flag (download!!), GateKeeper will actually generate a popup, which is really nice. It seems that Apple significantly improved it indeed.

Closing thoughts

I think plugins, like this is a bit unexplored space for persisting on macOS systems, there might be more similar ones. For example you could also use a Spotlight Quicklook plugin which is invoked, whenever someone hits the spacebar on a file in `Finder.app` for a quick preview of the file. Likely it would have the same level of access, and it's much more limited cause you need the user to preview something, but it could still work.