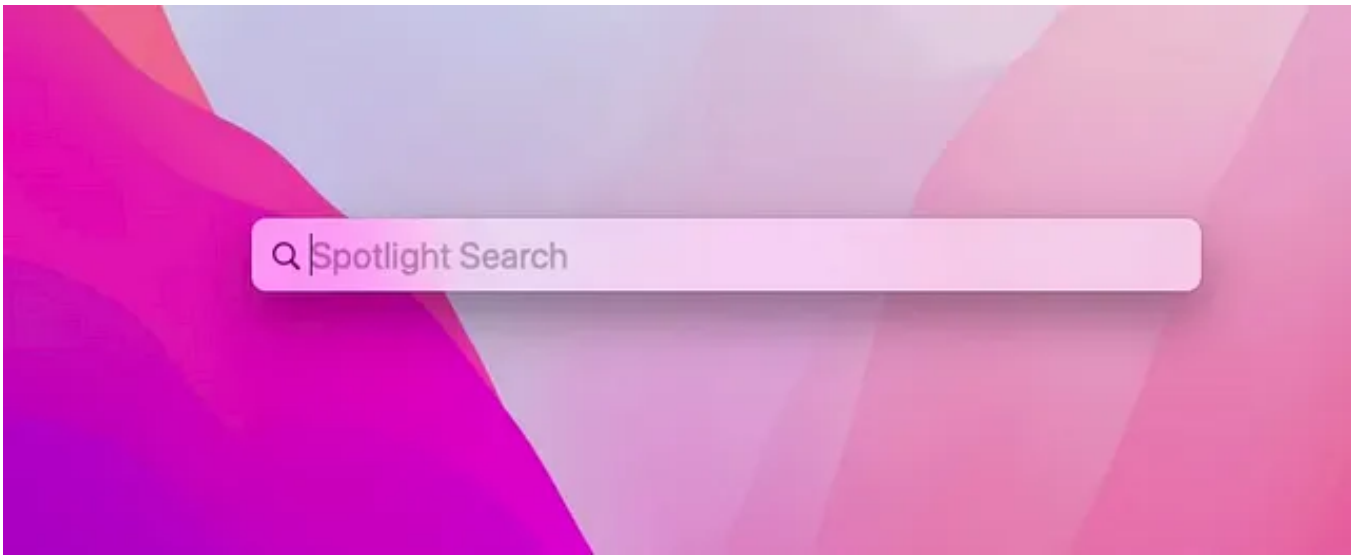
Querying Spotlight APIs With JXA

Cedric Owens



TL;DR This blog post takes a brief look at how to use JXA (native JavaScript for Automation on macOS) to query Spotlight APIs. In particular, this post will be looking at how to run mdfind searches in JXA without invoking the on-disk mdfind binary.

W**hy?**

1. I enjoy writing scripts/utilities for macOS and often use Swift and JXA. When already in a coding context, it just doesn't feel right to me personally going from code to call an on-disk binary and then going back to code when I can just make the API calls directly from code. I have found JXA to be a bit of a challenge at times when it comes to learning how to make certain API calls since there is limited documentation available publicly. However, since JXA will remain on macOS (while other scripting runtimes such as python, perl, and ruby are being removed from base macOS installs), I enjoy putting the time in to learn JXA a bit more.
2. As a red teamer, moving away from the command line to API calls is always preferred, especially as the state of macOS detections

continues to improve over time.

Let's Dig In

Towards the end of 2021, I put together a blog post about how useful Spotlight data is and how it can be used to do things such as enumerate Terminal's TCC folder permissions, search for files with keywords, and search for newly created/modified files. Here's a link to that blog post if you want to revisit that info:

"Spotlighting" Your TCC Access Permissions

As an offensive security engineer, one of the things I aim to avoid is quickly burning initial access on a host. On…

The blog above includes a link to my github repo for code examples for querying Spotlight from Swift and JXA. At the time of my blog post, I only knew how to make Spotlight API calls from Swift (my JXA projects at the time simply ran the on disk mdfind binary). However, since then I have spent more time trying to get my JXA code to more closely emulate my Swift code and I now have JXA code that queries Spotlight APIs in the same way that my Swift code does. Yay! Let's take a look at some examples.

My previous JXA Code Calling the on-disk mdfind binary via doShellScript (undesired!):

```
......//runs the on-disk mdfind binary
var dir_check = currentApp.doShellScript('mdfind kMDItemKind=F
var dir_check2 = dir_check.split('\r');//loop through all arra
for(let p=0; p<dir_check2.length; p++){if (dir_check2[p] == "/
results += "[+] Terminal already has folder access to /Users/"
}if (dir_check2[p] == "/Users/" + username + "/Documents"){
results += "[+] Terminal already has folder access to /Users/"
}if (dir_check2[p] == "/Users/" + username + "/Downloads"){
```

```
results += "[+] Terminal already has folder access to /Users/"
}}......
```

Swift Implementation, which gets the same data but via API calls (preferred!):

```
......let username = NSUserName()//set MDQuery string
let queryString = "kMDItemKind = Folder -onlyin /Users/\(userr
let query = MDQueryCreate(kCFAllocatorDefault, queryString as
MDQueryExecute(query, CFOptionFlags(kMDQuerySynchronous.rawVal
for i in 0..<MDQueryGetResultCount(query) {if let rawPtr = MDQ
if let path = MDItemCopyAttribute(item, kMDItemPath) as? Strir
if path == "/Users/\(username)/Desktop" {print("[+] Terminal H
```

I then embarked on the journey of trying to figure out how to make the same API calls in JXA. I call this a “journey” because I really had a tough time finding publicly available examples or documentation related to using JXA to make MDQuery API calls. So the approach I used was to try to mimic my Swift code as closely as possible.

Results

After much trial and error, here is the working JXA code that I have which makes MDQuery API calls:

```
......var username = $.NSUserName().js;//set the query string
var queryString = "kMDItemKind = Folder -onlyin ~";//create a
let query = $.MDQueryCreate($(), $(queryString), $(), $()); //
if ($.MDQueryExecute(query, 1)){//loop through query results
for(var i = 0; i < $.MDQueryGetResultCount(query); i++){
var mdItem = $.MDQueryGetResultAtIndex(query, i); //grab the k
var mdAttrs1 = $.MDItemCopyAttribute($.CFMakeCollectable(mdIte
var mdAttrs = ObjC.deepUnwrap(mdAttrs1); //search for strings
```

```
if (mdAttrs == "/Users/" + username + "/Desktop"){
results += "[+] Terminal already has folder access to /Users/"
} if (mdAttrs == "/Users/" + username + "/Documents"){
results += "[+] Terminal already has folder access to /Users/"
} if (mdAttrs == "/Users/" + username + "/Downloads"){
results += "[+] Terminal already has folder access to /Users/"
}
}...
...
```

That's it! No need to run any command line binaries!

Example of the output:

example TCC-Checker.js output

Next I went and updated all of the JXA projects in my Spotlight-Enum-Kit repository so you can see more examples (such as TCC Folder Enumeration, keyword file searches, newly created file searches, etc.) here:

GitHub - cedowens/Spotlight-Enum-Kit: JXA and swift code that can perform some macOS situational…

Repo of Swift and JXA projects to leverage macOS Spotlight db data for the following: TCC folder permissions…