TRACK 1

HITB SECCONF
AMSTERDAM - 2021

macOS local security: escaping the sandbox and bypassing TCC

Thijs Alkemade & Daan Keuper

Computest Research Department

Speakers

Thijs Alkemade (@xnyhps)

Daan Keuper

linkedin.com/in/daan-keuper/

linkedin.com/in/thijs-alkemade-28833414/

Computest

SECT7R

powered by Computest

Topic

- Suppose you have code execution as unprivileged user in macOS
- But sometimes you need more!
- How do you gain administrative access?
- Which hurdles do you have to overcome?

Agenda

- Deep dive in protection mechanisms on macOS
  - Code signing, Sandbox, TCC, SIP, SSV
- Discuss some vulnerabilities we found during our research
  - Unfortunately they are not all patched by Apple

macOS Security Mechanisms

Gatekeeper

Introduction

- Code Signing was introduced in Mac OS X Lion (10.7)
- Required for applications, though users can make a manual exception if an application is not signed
- Verification is handled by `com.apple.driver.AppleMobileFileIntegrity.kext` and `/usr/libexec/amfid`

Entitlements

- Every code signed application can be given a set of fine-grained permissions, using entitlements that are signed by Apple
- Typically special permissions are given to a (smaller) privileged process, communicating over XPC
- Important security mechanism, used across the entire operating system
- If an older vulnerable application has a specific entitlement, you could ship it with your malware

```
~ ARCH=x86_64 jtool2 --ent /bin/ps
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>com.apple.system-task-ports</key>
    <true/>
    <key>task_for_pid-allow</key>
    <true/>
</dict>
</plist>
```

```
~ codesign -d -vvvvvv /bin/ps
Executable=/bin/ps
Identifier=com.apple.ps
Format=Mach-O universal (x86_64 arm64e)
CodeDirectory v=20100 size=797 flags=0x0(none) hashes=18+5 location=embedded
Platform identifier=11
VersionPlatform=1
VersionMin=721152
VersionSDK=721152
Hash type=sha256 size=32
CandidateCDHash sha256=21d01508bc6e73222dedb4b914fc05acddba8075
CandidateCDHashFull sha256=21d01508bc6e73222dedb4b914fc05acddba8075e12b009ce0577710af10878e
Hash choices=sha256
CMSDigest=21d01508bc6e73222dedb4b914fc05acddba8075e12b009ce0577710af10878e
CMSDigestType=2
…
CDHash=21d01508bc6e73222dedb4b914fc05acddba8075
Signature size=4577
Authority=Software Signing
Authority=Apple Code Signing Certification Authority
Authority=Apple Root CA
Signed Time=23 Nov 2020 at 12:15:15
Info.plist=not bound
TeamIdentifier=not set
Sealed Resources=none
Internal requirements count=1 size=60
```

```
~ otool -l /bin/ps
…
Load command 16
      cmd LC_CODE_SIGNATURE
  cmdsize 16
  dataoff 69872
 datasize 6240
```

Resources

- The binary itself is signed and checked on every execution
- Resources itself are also signed, using `<app>/Contents/_CodeSignature/CodeResources`
- This is a signed plist which contains a hash of all resource files
- Unfortunately these are only checked on first run when the quarantine bit is set

Hardened Runtime

- Introduced in macOS Mojave (10.14)
- Enforced for Apps installed from the App Store
- Protects applications against various forms of process injection
- Prohibits the use of `DYLD_` environment variables, JIT and checks code signatures of libraries

Seatbelt

Introduction

- Introduced in Mac OS X 10.5 (Leopard)
- Enforced for almost all of Apple's applications
- Mac App Sandbox added in OS 10.7, giving each app a separate container in `~/Library/Containers/<bundle>`
- Handled by `com.apple.security.sandbox.kext` and `/usr/libexec/sandboxd`

Profiles

- Profiles are written in Scheme, can be found under `/System/Library/Sandbox/Profiles`
- The sandbox has hooks in all systemcalls, across the entire kernel tree
- A profile is based on the entitlements of the application

```
~ cat /System/Library/Sandbox/Profiles/com.apple.iMessage.addressbook.sb
…
(version 1)
(import "com.apple.iMessage.shared.sb")

(allow mach-lookup
    (global-name "com.apple.AddressBook.abd")
    (global-name "com.apple.AddressBook.AddressBookApplicationFrameworkIPC")
    (global-name "com.apple.AddressBook.ContactsAccountsService")
    (global-name "com.apple.AddressBook.SourceSync")
    (global-name “com.apple.backupd.xpc")
    (global-name "com.apple.corerecents.recentsd")
    (global-name "com.apple.logind")
    (global-name "com.apple.lsd.mapdb")
    (global-name "com.apple.metadata.mds")
    (global-name "com.apple.spotlight.IndexAgent")
    (global-name "com.apple.system.opendirectoryd.api")
    )

(allow user-preference-read
    (preference-domain "com.apple.AddressBook")
    (preference-domain "com.apple.AddressBook.CardDAVPlugin")
    )

(allow user-preference-write
    (preference-domain "com.apple.AddressBook")
    )

(allow file-map-executable
    (subpath "/System/Library/Address Book Plug-Ins")
    (home-subpath "/Library/Application Support/AddressBook/Sources")
```

System Integrity Protection

Introduction

- Introduced in OS X El Capitan (10.11)
- Sometimes revered to as ‘rootless’, internally often referenced as CSR (Configurable Software Restrictions)
- Aimed at limiting the power the root user has on a system
- Restricts file modifications, kernel/system extension loading and process debugging

```
 ~ sudo rm /bin/bash
override r-xr-xr-x  root/wheel restricted,compressed for /bin/bash? y
rm: /bin/bash: Operation not permitted

~ sudo dtruss /bin/ls
dtrace: system integrity protection is on, some features will not be
available

dtrace: failed to execute /bin/ls: (os/kern) failure

~ ls -ld -O@ /bin
drwxr-xr-x@ 38 root  wheel  restricted,hidden 1216 Jan  1  2020 /bin
      com.apple.rootless        0

~ ls -lO@ /bin/ls
-rwxr-xr-x  1 root  wheel  restricted,compressed 157360 Jan  1  2020 /bin/
ls
```

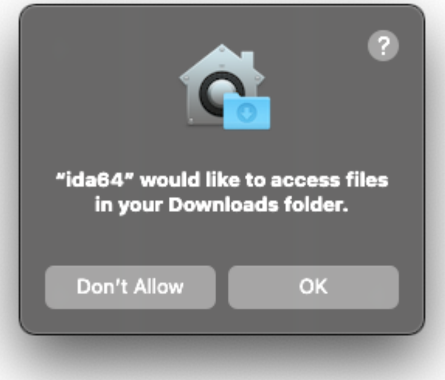
Technical details

- Effectively a sandbox profile called `platform_profile`
- Configuration can be found under `/System/Library/Sandbox/rootless.conf`
- Enabled on boot using the `csr-active-config` nvram variable, changing this variable is prohibited by SIP

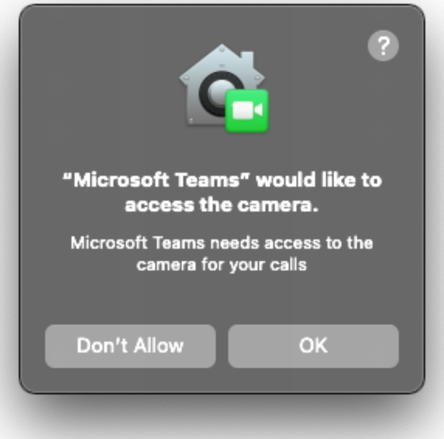
```
~ ARCH=x86_64 jtool2 --ent /bin/ps
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>com.apple.system-task-ports</key>
    <true/>
    <key>task_for_pid-allow</key>
    <true/>
</dict>
</plist>
```

Transparency, Consent & Control

You've probably seen this as

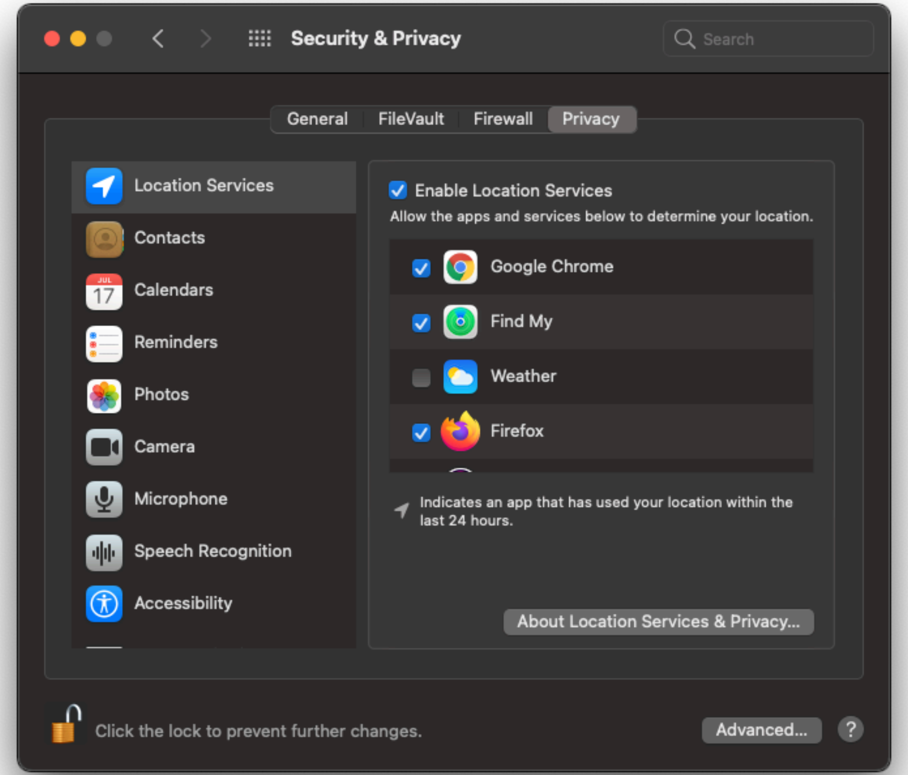


"ida64" would like to access files in your Downloads folder.



"Microsoft Teams" would like to access the camera.

Microsoft Teams needs access to the camera for your calls



☑ Enable Location Services

Allow the apps and services below to determine your location.

☑ Google Chrome

☑ Find My

☐ Weather

☑ Firefox

Indicates an app that has used your location within the last 24 hours.

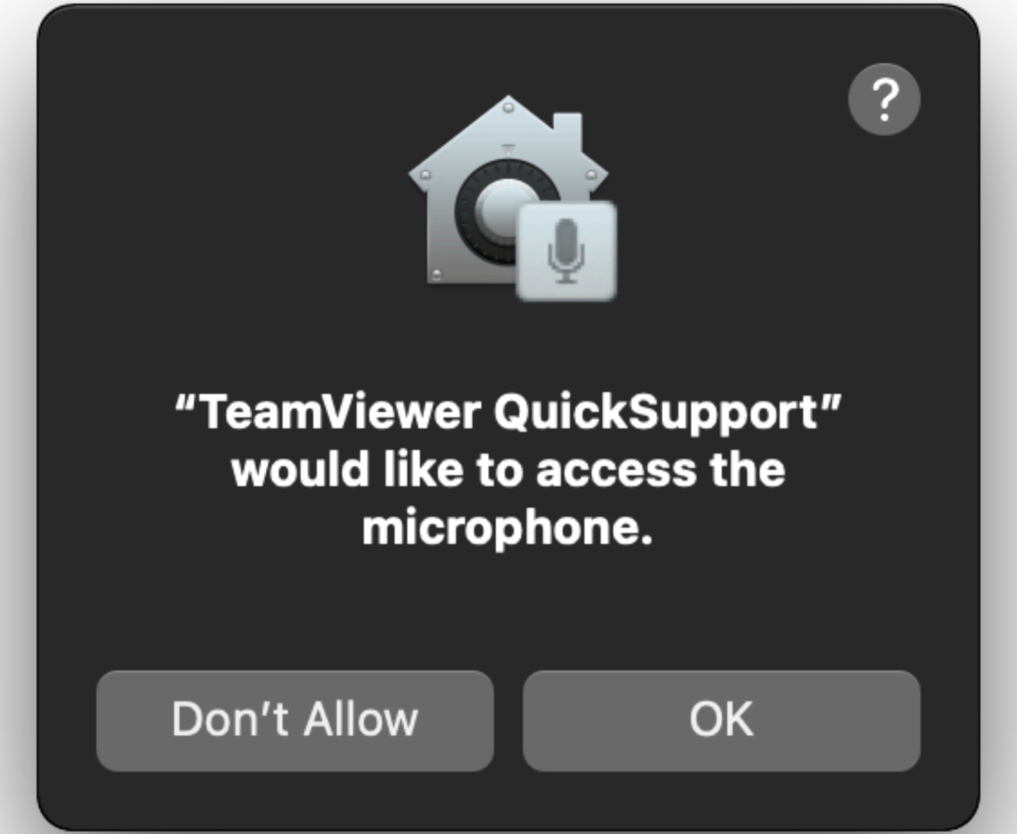
Click the lock to prevent further changes.

Introduction

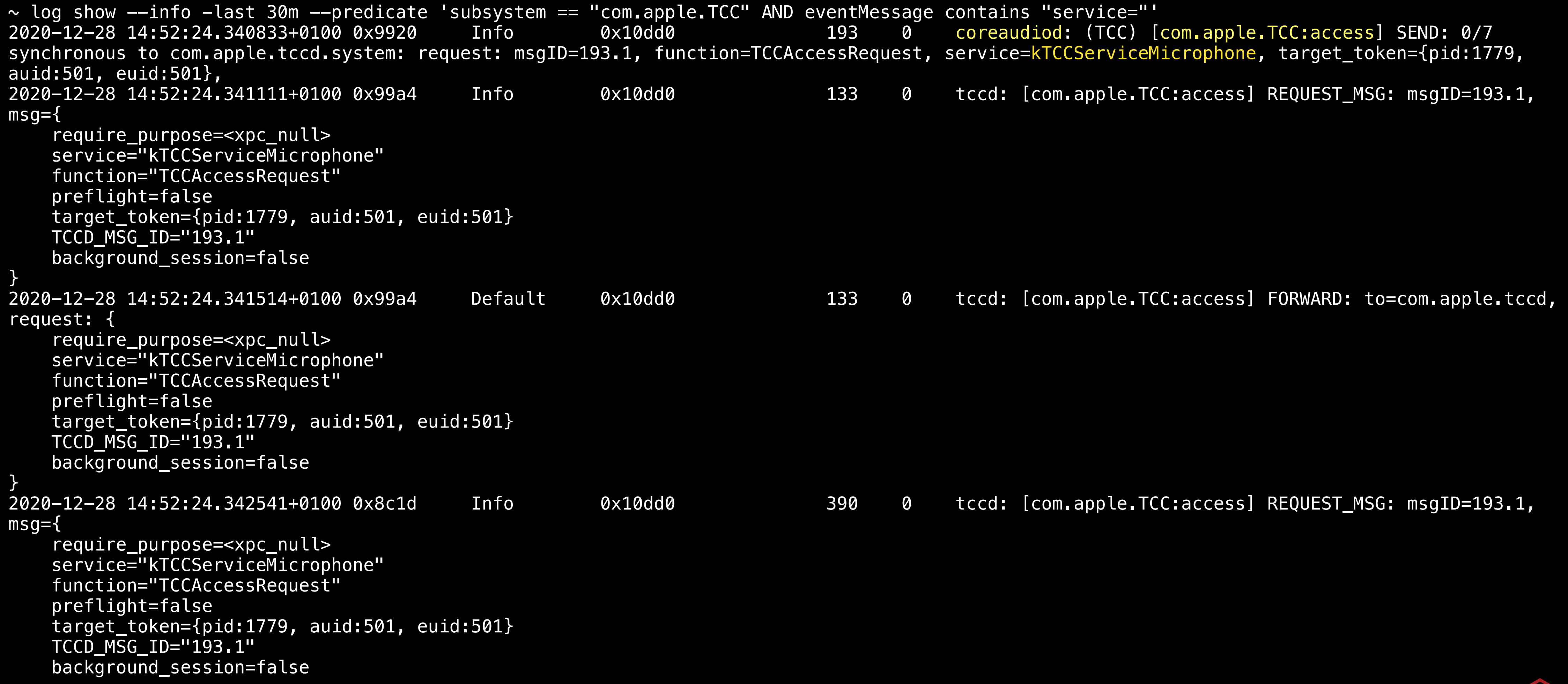
- Introduced in macOS Mojave (10.14)
- Dynamic sandbox for privacy sensitive subsystems, e.g. access to the camera, location services, Documents folder etc.
- Permissions are inherited from the parent process
- Permissions are stored with the Bundle ID and Developer ID
- You could ship an older vulnerable version of an app with your malware

```
~ ps -ax -o pid,user,command | grep "[t]ccd"
  131 root    /System/Library/PrivateFrameworks/TCC.framework/Resources/tccd
system
 6929 user    /System/Library/PrivateFrameworks/TCC.framework/Resources/tccd

~ sudo procexp all fds | grep tccd | grep .db
tccd    131 FD  5u  /Library/Application Support/com.apple.TCC/TCC.db @0x0
tccd   6929 FD  4u  /Users/user/Library/Application Support/com.apple.TCC/TCC.db
@0x0
```

```
~ sqlite3  ~/Library/Application\ Support/com.apple.TCC/TCC.db
Error: unable to open database "/Users/user/Library/Application
Support/com.apple.TCC/TCC.db": authorization denied

~ ls -lO@ /Library/Application\ Support/com.apple.TCC/TCC.db
-rw-r--r--  1 root  wheel  restricted 77824 Dec 28 13:35 /Library/
Application Support/com.apple.TCC/TCC.db
```



“TeamViewer QuickSupport” would like to access the microphone.

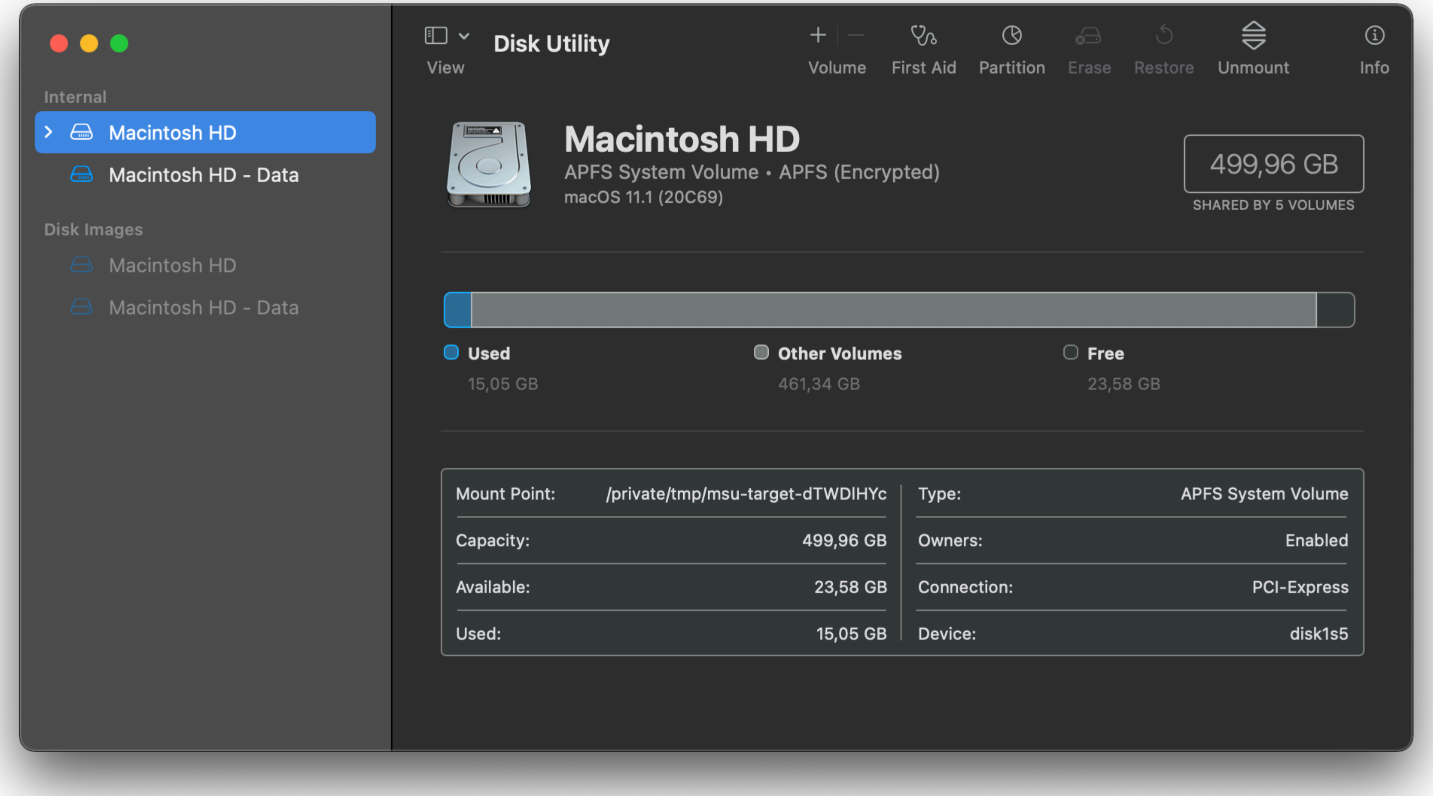


```
~ log show --info -last 30m --predicate 'subsystem == "com.apple.TCC" AND eventMessage contains "service="'
2020-12-28 14:52:24.340833+0100 0x9920     Info        0x10dd0              193    0    coreaudiod: (TCC) [com.apple.TCC:access] SEND: 0/7
synchronous to com.apple.tccd.system: request: msgID=193.1, function=TCCAccessRequest, service=kTCCServiceMicrophone, target_token={pid:1779,
auid:501, euid:501},
2020-12-28 14:52:24.341111+0100 0x99a4     Info        0x10dd0              133    0    tccd: [com.apple.TCC:access] REQUEST_MSG: msgID=193.1,
msg={
    require_purpose=<xpc_null>
    service="kTCCServiceMicrophone"
    function="TCCAccessRequest"
    preflight=false
    target_token={pid:1779, auid:501, euid:501}
    TCCD_MSG_ID="193.1"
    background_session=false
}
2020-12-28 14:52:24.341514+0100 0x99a4     Default     0x10dd0              133    0    tccd: [com.apple.TCC:access] FORWARD: to=com.apple.tccd,
request: {
    require_purpose=<xpc_null>
    service="kTCCServiceMicrophone"
    function="TCCAccessRequest"
    preflight=false
    target_token={pid:1779, auid:501, euid:501}
    TCCD_MSG_ID="193.1"
    background_session=false
}
2020-12-28 14:52:24.342541+0100 0x8c1d     Info        0x10dd0              390    0    tccd: [com.apple.TCC:access] REQUEST_MSG: msgID=193.1,
msg={
    require_purpose=<xpc_null>
    service="kTCCServiceMicrophone"
    function="TCCAccessRequest"
    preflight=false
    target_token={pid:1779, auid:501, euid:501}
    TCCD_MSG_ID="193.1"
    background_session=false
```

```
~ ARCH=x86_64 jtool2 --ent /System/Library/PrivateFrameworks/TCC.framework/Resources/tccd
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
        <key>com.apple.fileprovider.acl-read</key>
        <true/>
        <key>com.apple.private.kernel.global-proc-info</key>
        <true/>
        <key>com.apple.private.notificationcenterui.tcc</key>
        <true/>
        <key>com.apple.private.responsibility.set-arbitrary</key>
        <true/>
        <key>com.apple.private.security.storage.TCC</key>
        <true/>
        <key>com.apple.private.system-extensions.tcc</key>
        <true/>
        <key>com.apple.private.tcc.allow</key>
        <array>
                <string>kTCCServiceSystemPolicyAllFiles</string>
        </array>
        <key>com.apple.private.tcc.manager</key>
        <true/>
        <key>com.apple.rootless.storage.TCC</key>
        <true/>
</dict>
</plist>
```

Signed System Volume

Separate System Volume



Macintosh HD

APFS System Volume • APFS (Encrypted)

macOS 11.1 (20C69)

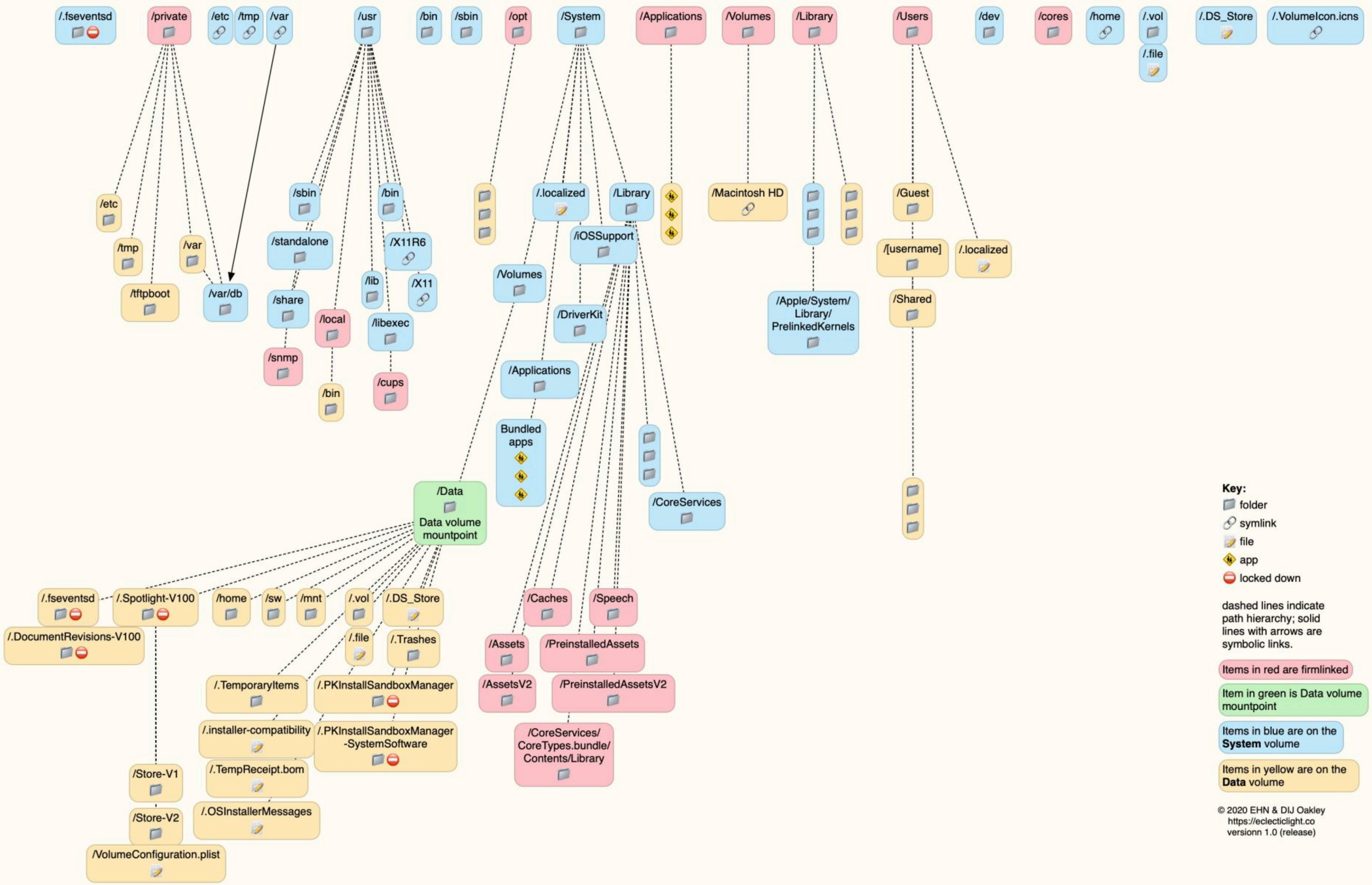
499,96 GB

SHARED BY 5 VOLUMES

Used	Other Volumes	Free
15,05 GB	461,34 GB	23,58 GB

Mount Point:	/private/tmp/msu-target-dTWDIHYc	Type:	APFS System Volume
Capacity:	499,96 GB	Owners:	Enabled
Available:	23,58 GB	Connection:	PCI-Express
Used:	15,05 GB	Device:	disk1s5

macOS Catalina 10.15 Boot Volume Group Layout



Key:
- folder
- symlink
- file
- app
- locked down

dashed lines indicate path hierarchy; solid lines with arrows are symbolic links.

Items in red are firmlinked

Item in green is Data volume mountpoint

Items in blue are on the **System** volume

Items in yellow are on the **Data** volume

© 2020 EHN & DIJ Oakley
https://eclecticlight.co
versionn 1.0 (release)

Introduction

- Introduced in macOS Big Sur as an extension to the read-only system volume from macOS Catalina
- Protects macOS system files from tampering
- Adds cryptographic signature to all data on the system volume

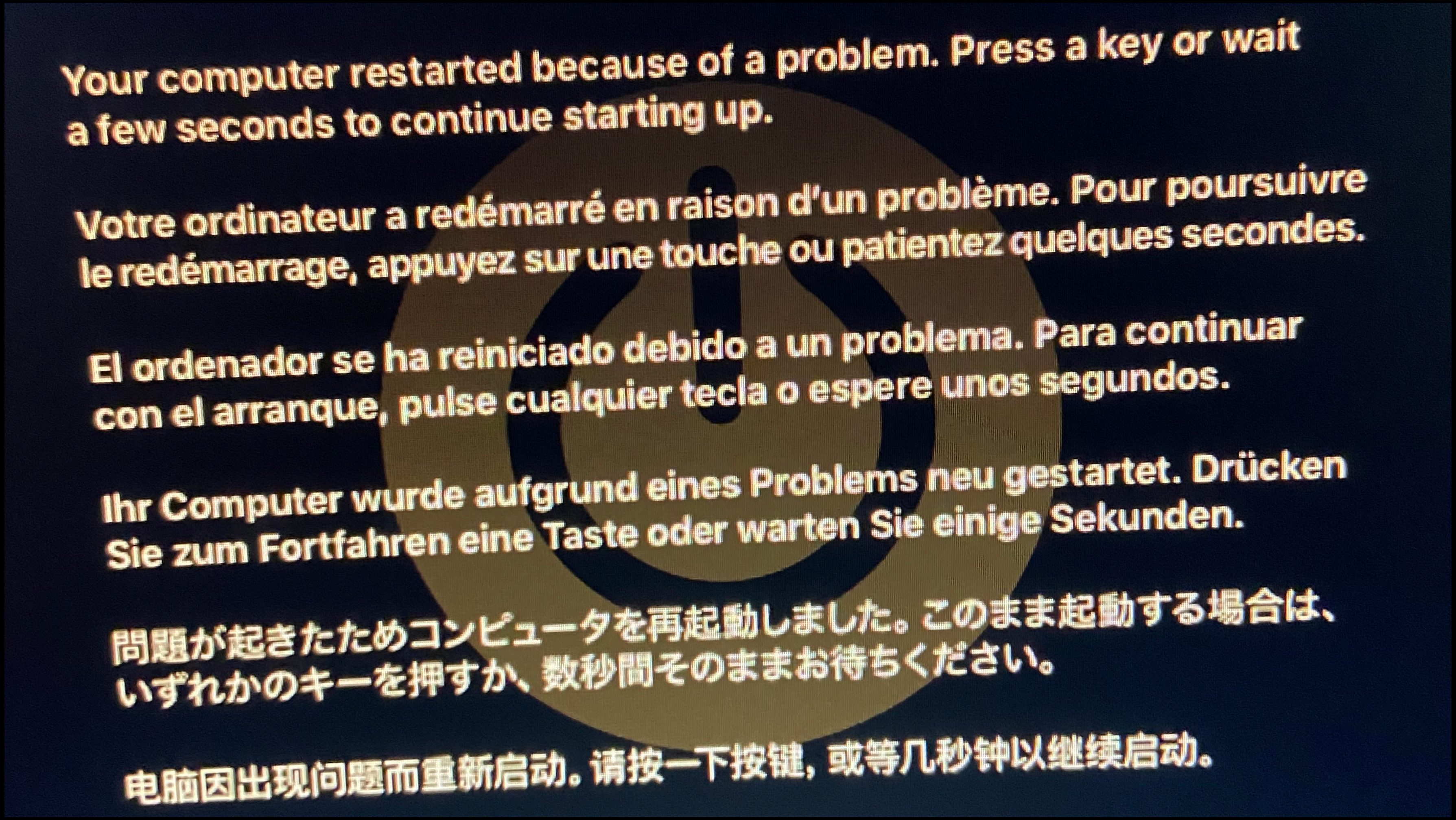
Merkle Tree

- The system volume now contains a Merkle Tree which is validated during the boot process
- The hashes are stored as metadata in APFS
- On the root node this is called the seal
- If the seal is broken, the system restores from a previous snapshot
- Relevant data structures can be found in the Apple File System Reference

Installing macOS Updates

- Your system has a permanently hidden Update volume, which is a snapshot of your Big Sur installation
- Patches are applied to this snapshot, if everything succeeded the snapshot is sealed and booted
- If the update fails the system can use its previous snapshot

```
~ diskutil apfs list
|
+-- Container disk1 C45A2F01-3035-4029-8440-0E8EEF1B6AD8
    ====================================================
    APFS Container Reference:     disk1
    Size (Capacity Ceiling):      499963174912 B (500.0 GB)
    Capacity In Use By Volumes:   476552110080 B (476.6 GB) (95.3% used)
    Capacity Not Allocated:       23411064832 B (23.4 GB) (4.7% free)
    |
    +-< Physical Store disk0s2 D09B1C47-F45F-4250-9E20-272728D1F1C9
    |   -----------------------------------------------------------
    |   APFS Physical Store Disk:   disk0s2
    |   Size:                       499963174912 B (500.0 GB)
    |
    +-> Volume disk1s5 F5B775BF-EEEF-4323-AF4D-6174B11D7AB9
        ---------------------------------------------------
        APFS Volume Disk (Role):   disk1s5 (System)
        Name:                      Macintosh HD (Case-insensitive)
        Mount Point:               /private/tmp/msu-target-dTWDlHYc
        Capacity Consumed:         15047917568 B (15.0 GB)
        Sealed:                    Broken
        FileVault:                 Yes (Unlocked)
        Encrypted:                 No
        |
        Snapshot:                  264AD255-656D-4C75-A1B7-C2ADEBDCBBBC
        Snapshot Disk:             disk1s5s1
        Snapshot Mount Point:      /
        Snapshot Sealed:           Yes
```



Your computer restarted because of a problem. Press a key or wait a few seconds to continue starting up.

Votre ordinateur a redémarré en raison d'un problème. Pour poursuivre le redémarrage, appuyez sur une touche ou patientez quelques secondes.

El ordenador se ha reiniciado debido a un problema. Para continuar con el arranque, pulse cualquier tecla o espere unos segundos.

Ihr Computer wurde aufgrund eines Problems neu gestartet. Drücken Sie zum Fortfahren eine Taste oder warten Sie einige Sekunden.

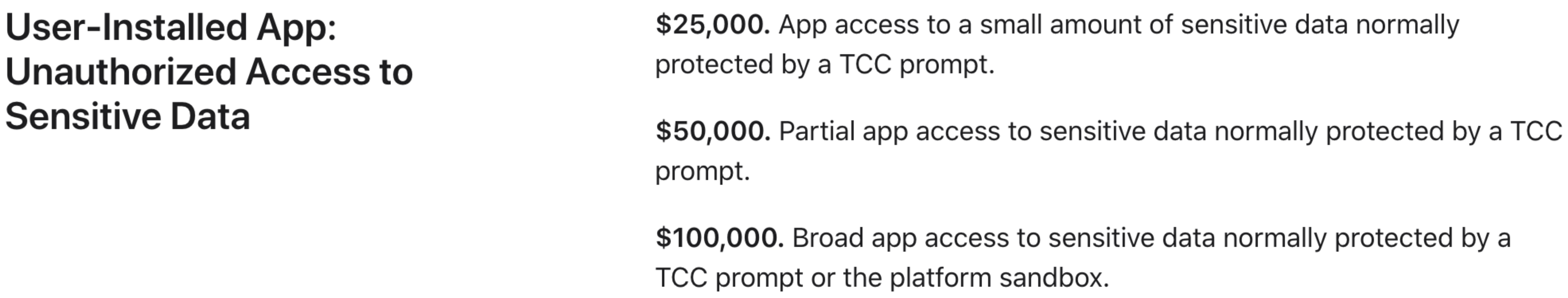
問題が起きたためコンピュータを再起動しました。このまま起動する場合は、いずれかのキーを押すか、数秒間そのままお待ちください。

电脑因出现问题而重新启动。请按一下按键，或等几秒钟以继续启动。

Overview

- **Code Signing** guarantees that code was published by a specific organisation
- **Seatbelt** handles static permissions for apps
- **TCC** handles user controlled permissions
- **SIP** guarantees the integrity of the system as a whole
- **Signed System Volume** prevents modification of system files

Bug Bounties



User-Installed App: Unauthorized Access to Sensitive Data

$25,000. App access to a small amount of sensitive data normally protected by a TCC prompt.

$50,000. Partial app access to sensitive data normally protected by a TCC prompt.

$100,000. Broad app access to sensitive data normally protected by a TCC prompt or the platform sandbox.

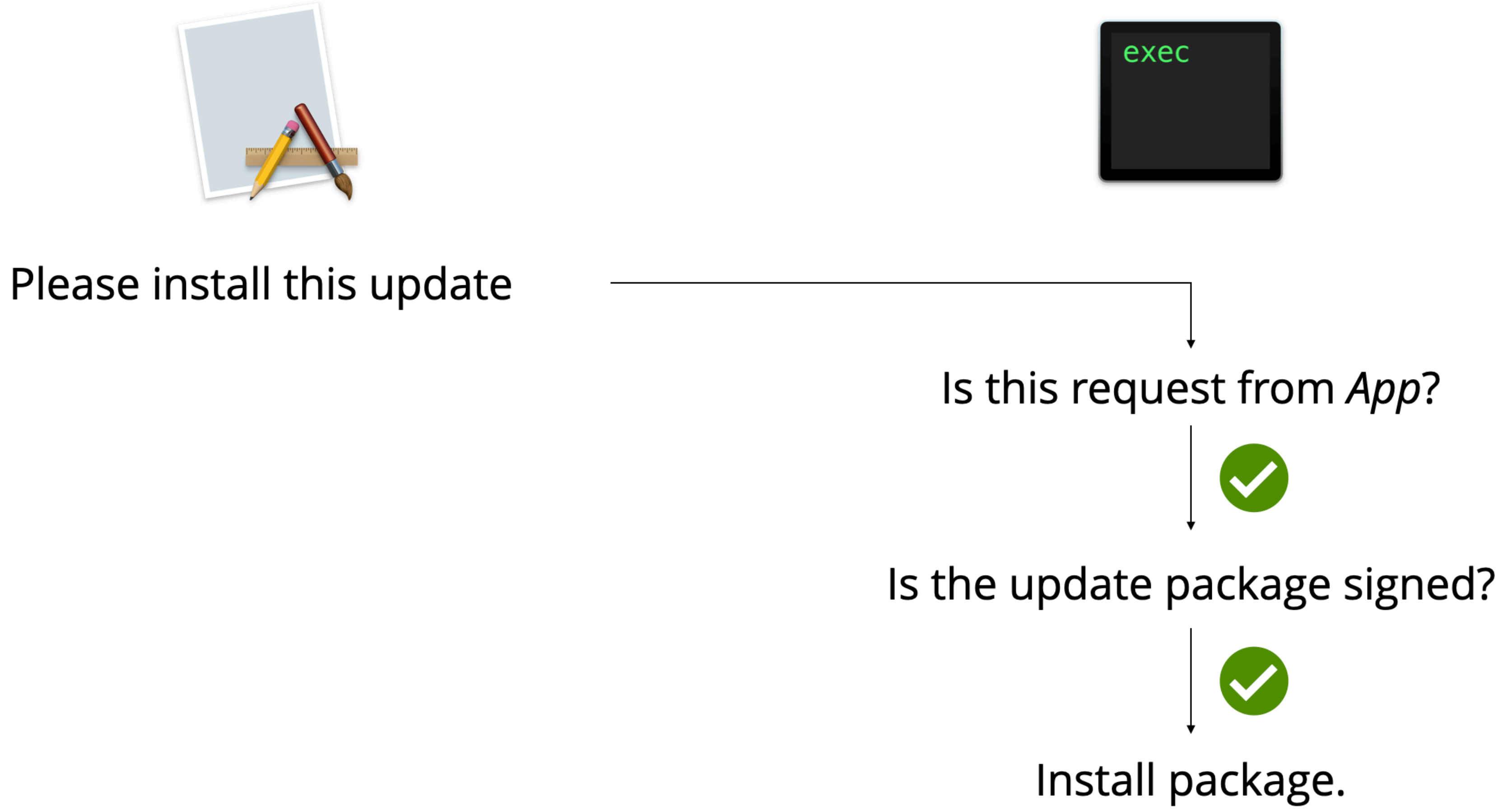
Vulnerabilities

Adobe Acrobat DC

Privileged updaters

Introduction

- Some systems the main user has a standard user account.
- Non-admin users are not allowed to change /Applications.
- How to install updates?
- Service running as root to handle installation.



Please install this update

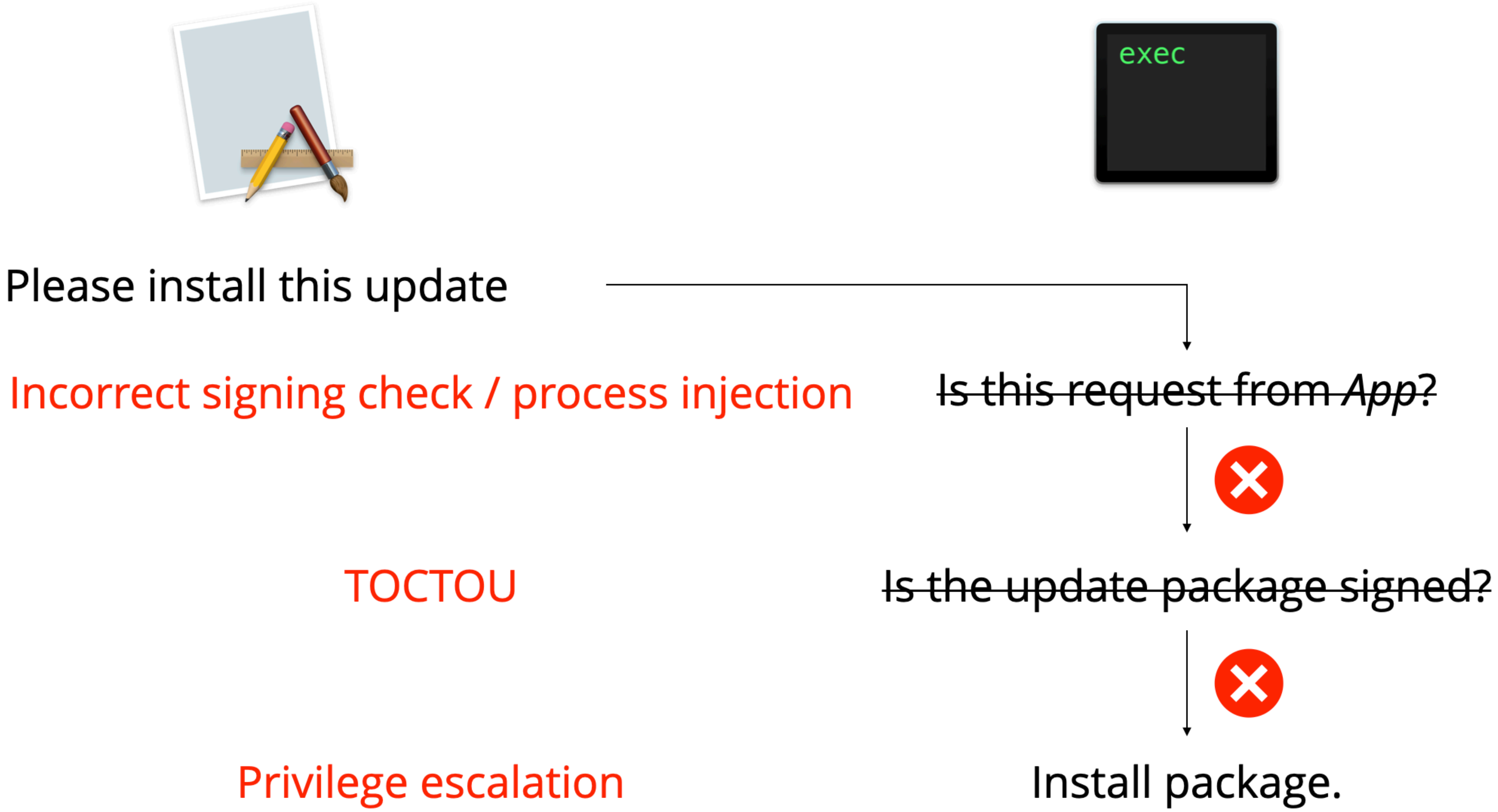
Is this request from *App*?

✓

Is the update package signed?

✓

Install package.



Please install this update

Incorrect signing check / process injection

~~Is this request from *App*?~~

TOCTOU

~~Is the update package signed?~~

Privilege escalation

Install package.

Privileged updaters

- Adobe Acrobat DC was vulnerable.
  - No codesigning check, symlinks for update package.
  - Reported by Yuebin Sun (@yuebinsun2020) of Tencent Security Xuanwu Lab. (May 2020)
  - https://rekken.github.io/2020/05/14/Security-Flaws-in-Adobe-Acrobat-Reader-Allow-Malicious-Program-to-Gain-Root-on-macOS-Silently/

Privileged updaters

- Adobe Acrobat DC was still vulnerable.
  - Wrong codesigning check, hardlinks allowed.
  - Reported by Csaba Fitzl (@theevilbit) from Offensive Security working with iDefense Labs. (August 2020)

Privileged updaters

- Adobe Acrobat DC was still vulnerable…
    - Codesigning check unfinished, open file descriptor.
    - Reported by Thijs Alkemade from Computest Research Division. (November 2020)

Privileged updaters

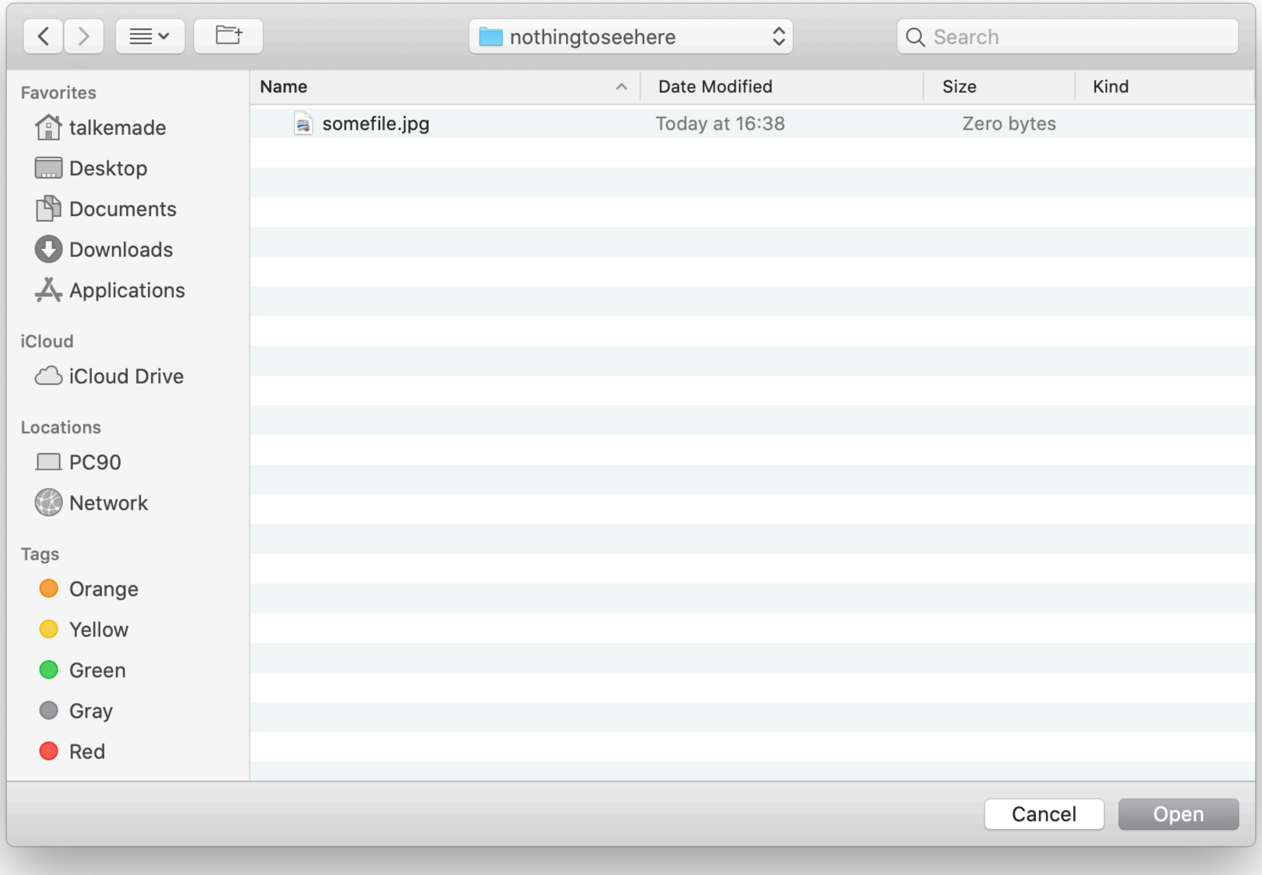
- Affected many other apps too:
  - Google Chrome
  - Microsoft AutoUpdate
  - Microsoft Teams
  - (Unnamed company, still under disclosure)

Privileged updaters

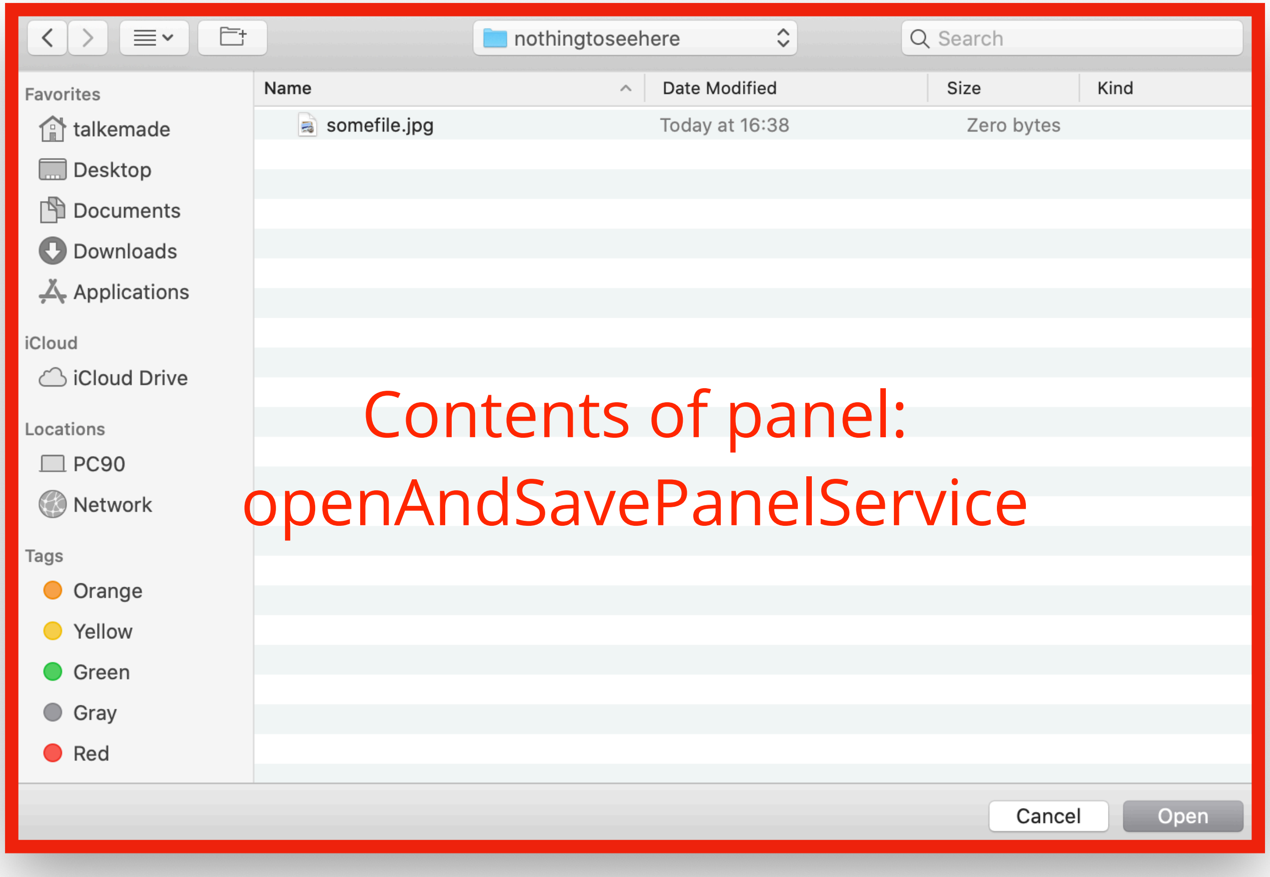
- Nothing uninstalls the updater if you delete the app
- If you have ever used Adobe Acrobat and then deleted it, you'll probably still have a vulnerable updater!
- Check `/Library/LaunchAgents` and `/Library/LaunchDaemons`

CVE-2020-27900

Open and save panels


nothingtoseehere
Search
Favorites
talkemade
Desktop
Documents
Downloads
Applications
iCloud
iCloud Drive
Locations
PC90
Network
Tags
Orange
Yellow
Green
Gray
Red
Name
Date Modified
Size
Kind
somefile.jpg
Today at 16:38
Zero bytes
Cancel
Open

Window: the app

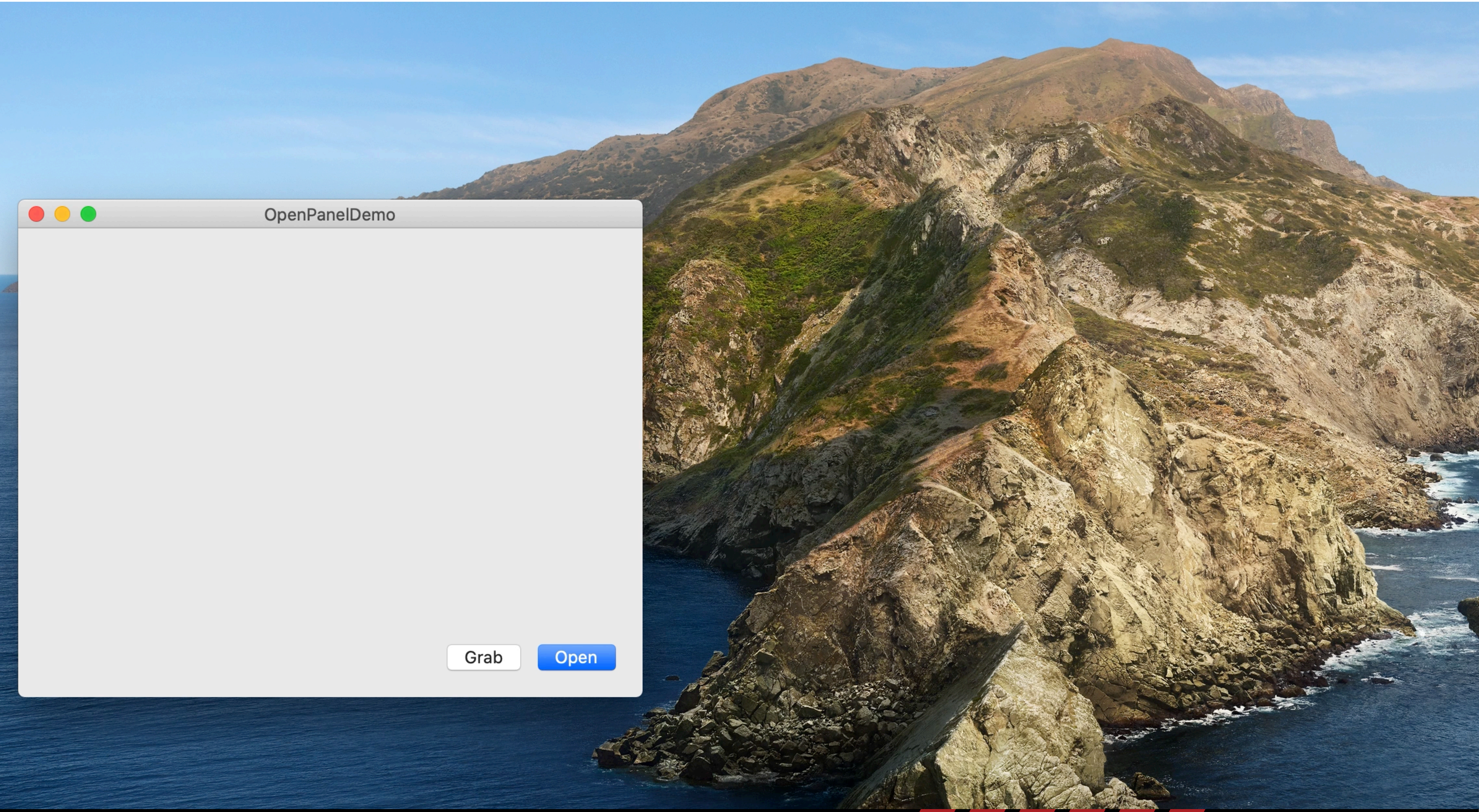


Name	Date Modified	Size	Kind
somefile.jpg	Today at 16:38	Zero bytes	

Contents of panel: openAndSavePanelService

Open panels

- Private method `–[NSRemoteView snapshot:]`
- Takes an image of the panel, returns it to the app!
- List of files, previews of certain files, etc.


OpenPanelDemo
Grab
Open

CVE-2020-10009

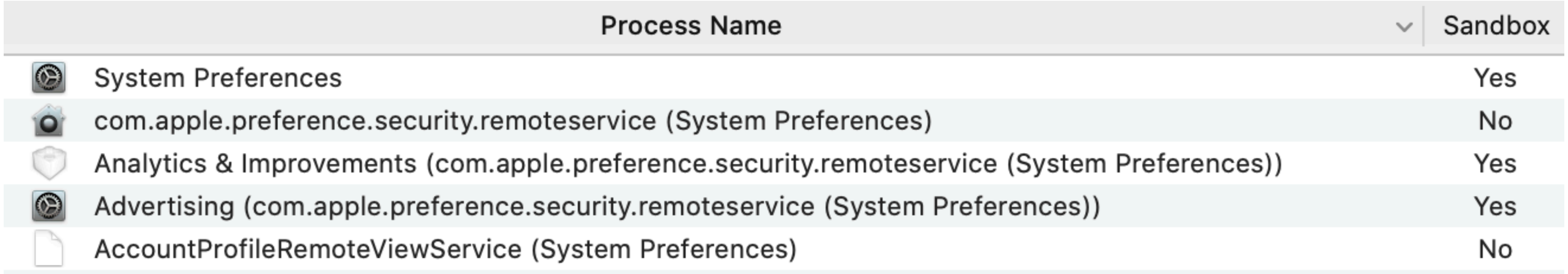
System Preferences sandbox escape

fork() + exec()

```
system("ls /");

Applications   Secomba         bin         home       tmp […]

system("/Applications/Safari.app/Contents/MacOS/Safari");

kernel   Sandbox: Safari(47541) deny(1) forbidden-
sandbox-reinit
```

System Preferences

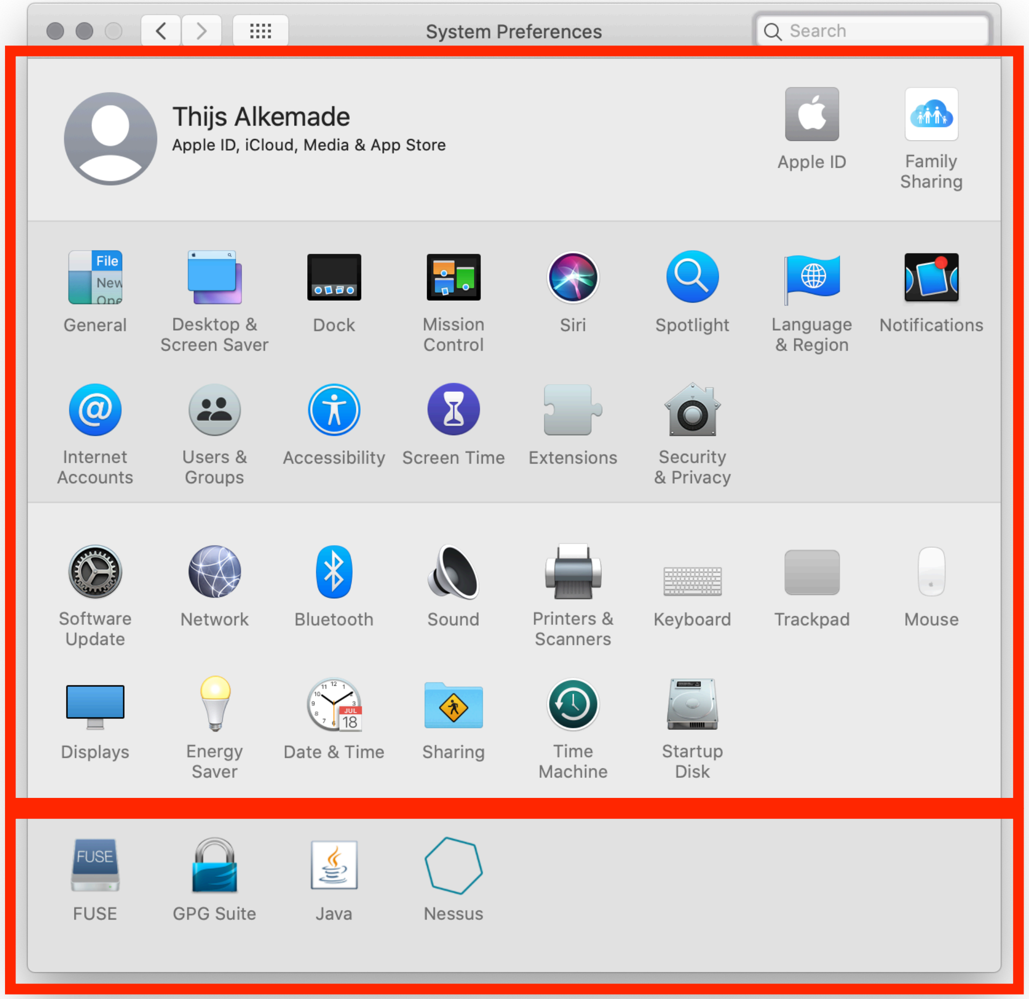
```
system("/System/Applications/System\\ Preferences.app/
Contents/MacOS/System\\ Preferences");
```



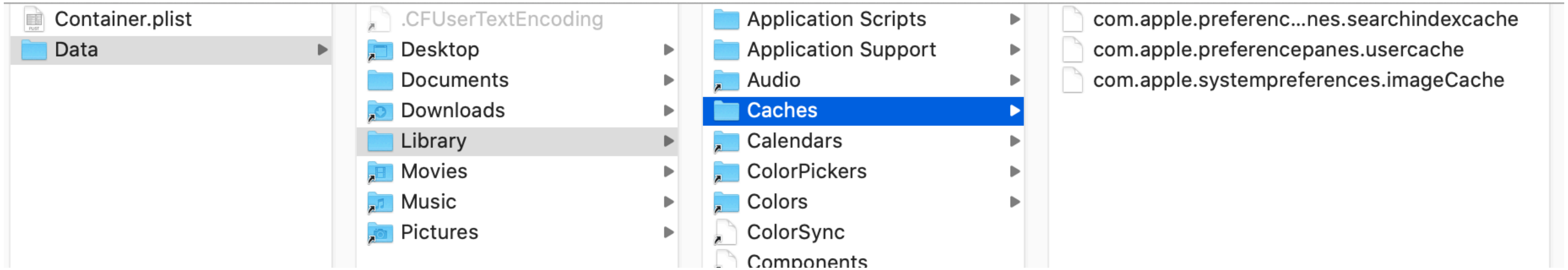
Process Name	Sandbox
System Preferences	Yes
com.apple.preference.security.remoteservice (System Preferences)	No
Analytics & Improvements (com.apple.preference.security.remoteservice (System Preferences))	Yes
Advertising (com.apple.preference.security.remoteservice (System Preferences))	Yes
AccountProfileRemoteViewService (System Preferences)	No

System panes: XPC services

Third-party panes: bundles



System Preferences



Container.plist
Data

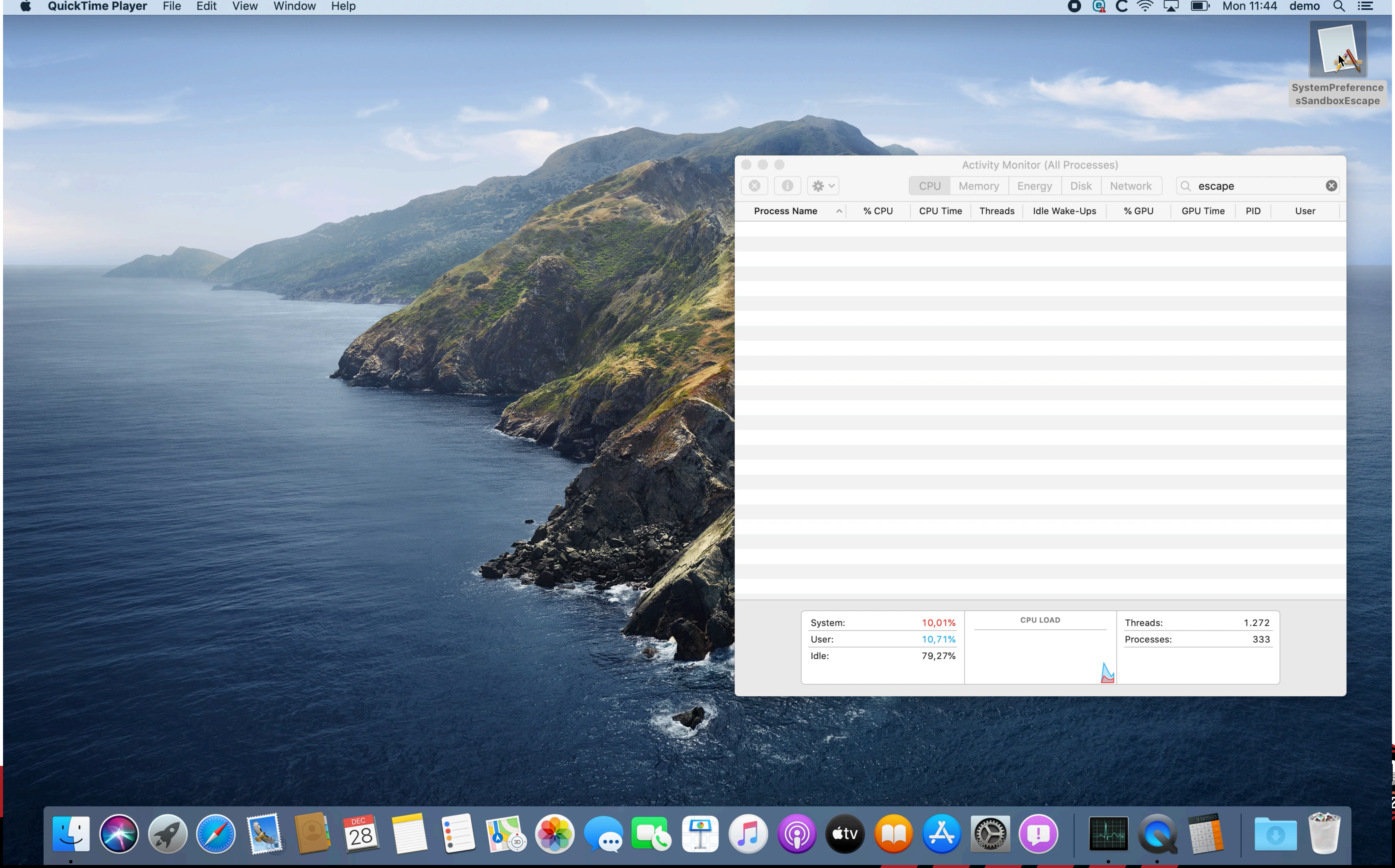
.CFUserTextEncoding
Desktop
Documents
Downloads
Library
Movies
Music
Pictures

Application Scripts
Application Support
Audio
Caches
Calendars
ColorPickers
Colors
ColorSync
Components

com.apple.preferenc...nes.searchindexcache
com.apple.preferencepanes.usercache
com.apple.systempreferences.imageCache

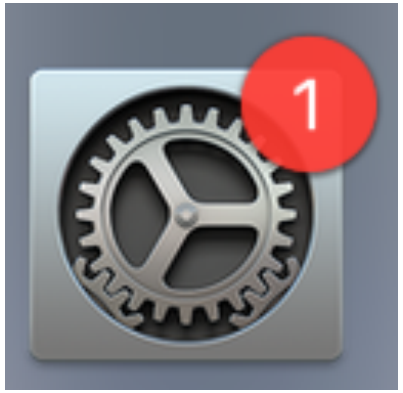
System Preferences

1. Launch System Preferences.
2. Wait for it to create its usercache file.
3. Kill System Preferences.
4. Modify cache to point to a bundle in my app.
5. Start System Preferences.
6. Wait for user to activate the modified pane.



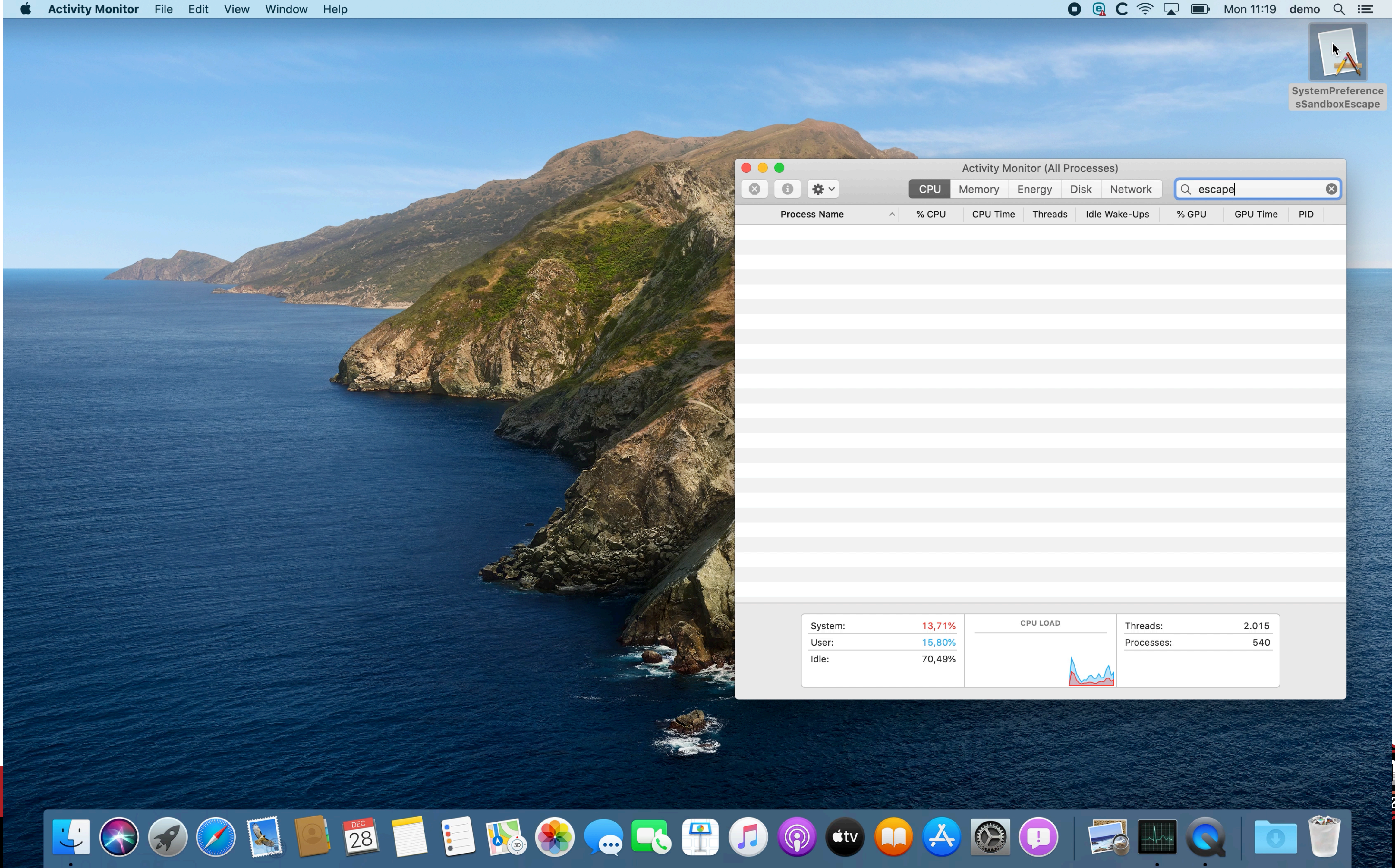
```
System:    10,01%        Threads:      1.272
User:      10,71%        Processes:      333
Idle:      79,27%
```

✗ `x-apple.systempreferences:com.apple.preference.network`

✓ 

System Preferences

1. Create valid usercache file with my own bundle.
2. Add alert for the added preference pane.
3. Start System Preferences.



Activity Monitor (All Processes)

escape

Process Name	% CPU	CPU Time	Threads	Idle Wake-Ups	% GPU	GPU Time	PID

System: 13,71%
User: 15,80%
Idle: 70,49%

Threads: 2.015
Processes: 540

System Preferences

- Fixed: Now quits if the app is in a sandbox.

Electron

TCC

Electron

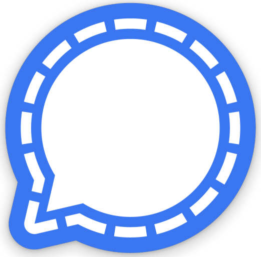
- TCC permissions are stored based on:
  - Bundle ID
  - Developer ID
- App version and filesystem path are irrelevant.

Electron

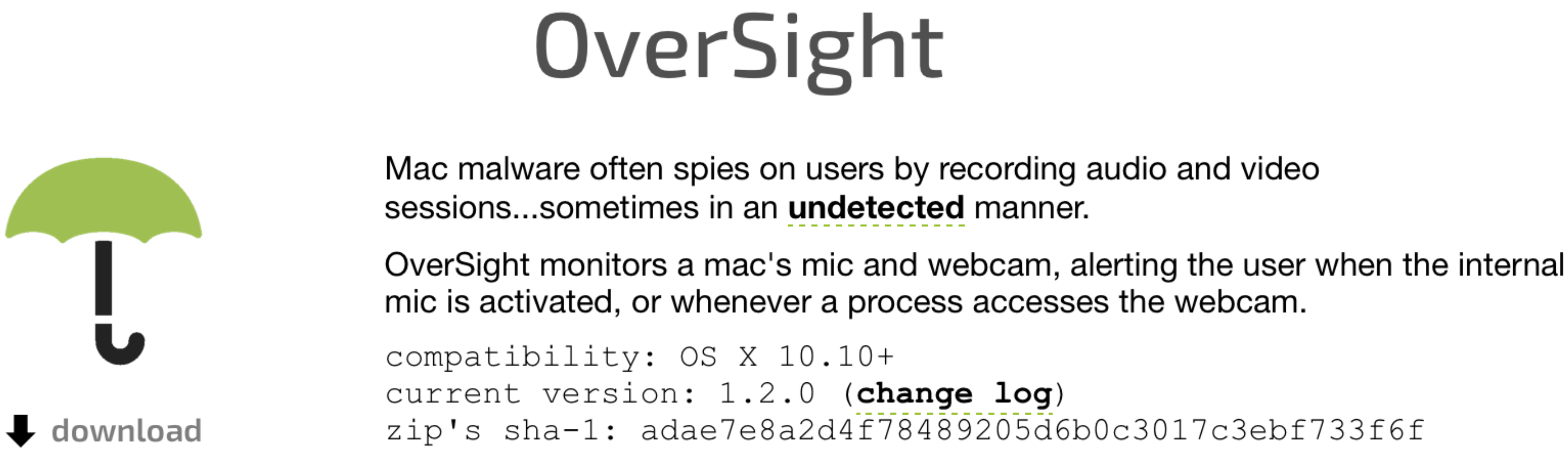
- Code signing check for TCC only checks the executable.
- With hardened runtime, libraries & frameworks are also checked.
- Interpreted code is not checked!
- Electron apps contain most of their code as JavaScript…

Electron

1. Copy app to a writable location.
2. Replace JavaScript with malicious code.
3. Launch modified app.
4. Use TCC permissions of the app.



https://objective-see.com/products/oversight.html

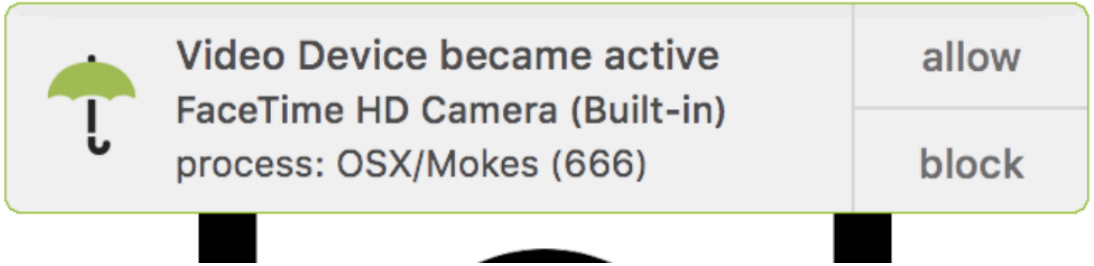


OverSight

Mac malware often spies on users by recording audio and video sessions...sometimes in an **undetected** manner.

OverSight monitors a mac's mic and webcam, alerting the user when the internal mic is activated, or whenever a process accesses the webcam.

```
compatibility: OS X 10.10+
current version: 1.2.0 (change log)
zip's sha-1: adae7e8a2d4f78489205d6b0c3017c3ebf733f6f
```



Video Device became active
FaceTime HD Camera (Built-in)
process: OSX/Mokes (666)

Not just Electron…

1. Download old copy of the app without library validation.
2. Replace any library with malicious library.
3. Launch app.
4. Use TCC permissions of that app.

App process injection

Process injection

- We saw that process injection can be used to:
  - Communicate with privileged helpers
  - “Steal” TCC permissions
- But what if we attack Apple’s own apps…?

Process injection

- Suppose we can inject into any application
- Then we can:
  - Sandbox escape
  - Escalate privileges from normal user to root
  - SIP filesystem bypass

Sandbox escape

- Inject in a non-sandboxed process.

Privilege escalation

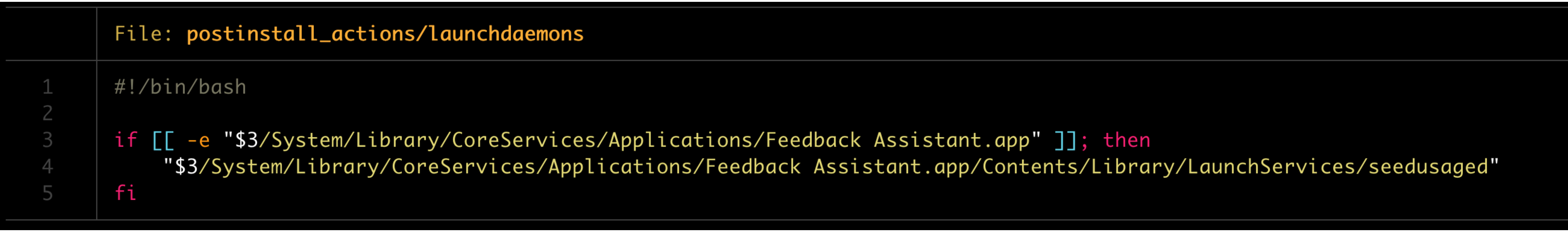
```
<key>
       com.apple.private.AuthorizationServices
</key>
<array>
       <string>
              system.install.apple-software
       </string>
       <string>
              system.install.apple-software.standard-user
       </string>
</array>
```

Privilege escalation

- Ilias Morad (@A2nkF_) found that the post-install script of **macOSPublicBetaAccessUtility.pkg** can execute arbitrary code as root.
- From CVE-2020–9854: “Unauthd” (https://a2nkf.github.io/unauthd_Logic_bugs_FTW/)

Privilege escalation

- From CVE-2020–9854: “Unauthd” (https://a2nkf.github.io/unauthd_Logic_bugs_FTW/)



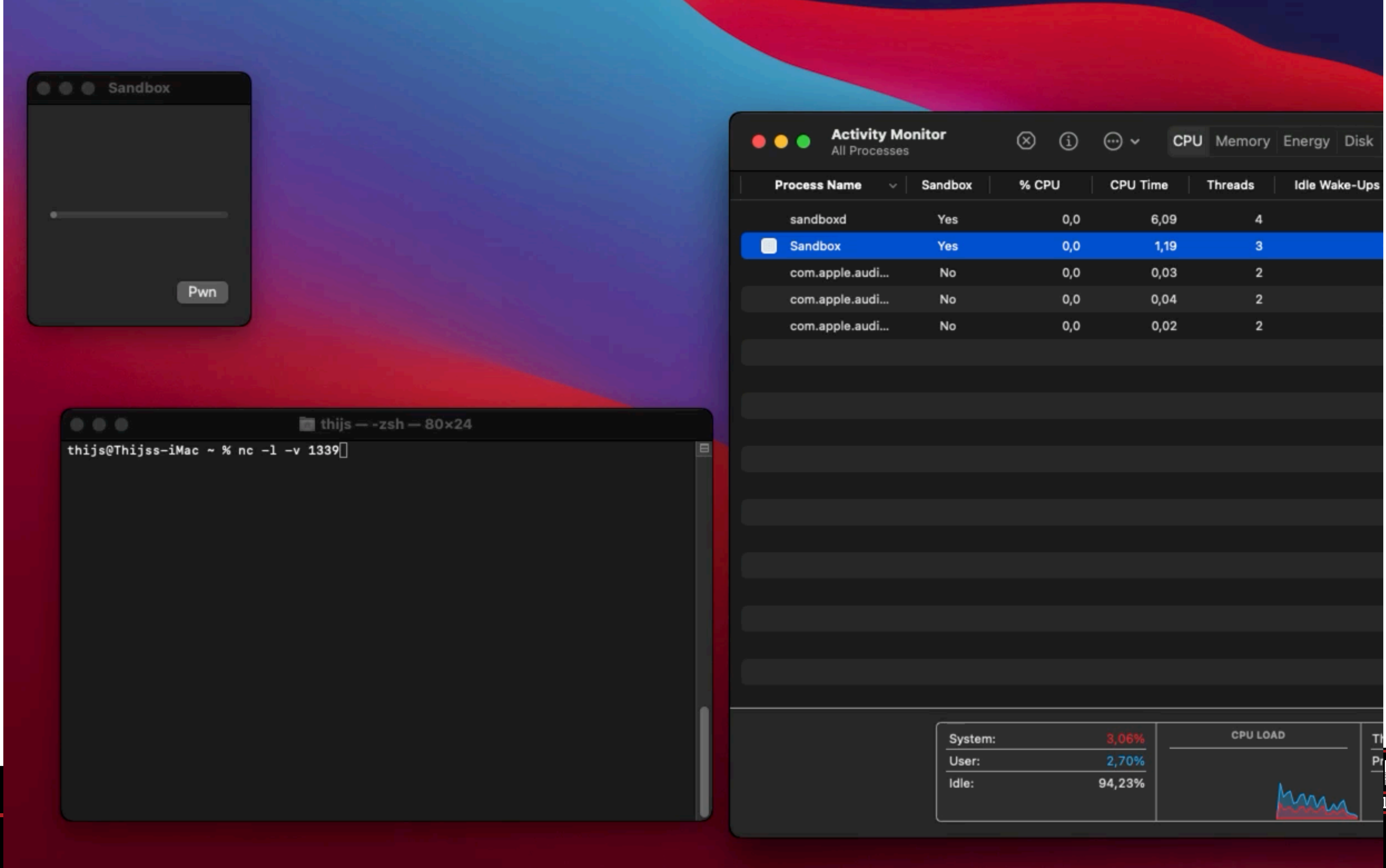
File: postinstall_actions/launchdaemons

```
#!/bin/bash

if [[ -e "$3/System/Library/CoreServices/Applications/Feedback Assistant.app" ]]; then
    "$3/System/Library/CoreServices/Applications/Feedback Assistant.app/Contents/Library/LaunchServices/seedusaged"
fi
```

SIP filesystem bypass

- **macOS Update Assistant.app** from an installation image can write to SIP locations:

```
<key>
     com.apple.rootless.install.heritable
</key>
<true/>
```



```
thijs@Thijss-iMac ~ % nc -l -v 1339
```

Process Name	Sandbox	% CPU	CPU Time	Threads	Idle Wake-Ups
sandboxd	Yes	0,0	6,09	4	
Sandbox	Yes	0,0	1,19	3	
com.apple.audi...	No	0,0	0,03	2	
com.apple.audi...	No	0,0	0,04	2	
com.apple.audi...	No	0,0	0,02	2	

System:	3,06%
User:	2,70%
Idle:	94,23%

Thoughts

Do these security measures work?

TCC

- Still pretty new and unknown by developers
- Only one app needs to be vulnerable to give malware permission
- Electron apps are inherently vulnerable

Sandboxing

- Low-level & kernel parts receive a lot of attention on iOS
- Higher layers are different: many interesting and unexplored attack surfaces on macOS

Process injection

- Windows and Linux have no security boundary between processes of the same user, unless opted-in to sandboxing (UWP, SELinux)
    - Therefore, all apps can access the same data and features
- Therefore, TCC is an extra security layer of macOS. Breaking it does not make macOS less secure than Windows or Linux.
- However, process injection vulnerabilities now have huge impact
    - Single process injection vulnerability: privilege escalation to root and bypass SIP

Conclusion

Conclusion

- Apple is trying to bring the security of macOS to the level of iOS
- But still a long way to go
- Needs work from all app developers
- Apple doesn’t want to enforce too many new restrictions at once

Thank You

For your attention

research@computest.nl