- History of Disk Arbitration Vulnerabilities -

PROLOGUE

*“A beginning is a very delicate time.
Know then, that it is the year 2024.”*

“The known macOS world is ruled by the Padishah Emperor Tim Cook the First, our leader”

“In this time, the most precious substance in the universe is the BANANA.”

*“The banana extends life.
The banana expands consciousness.
The banana is vital to MSA”*

“The Monkey Tribe and its navigators, who the banana has mutated over 4000 years, use the banana seeds, which gives them the ability to live
That is, not dying”

“Oh, yes, I forgot to tell you.”

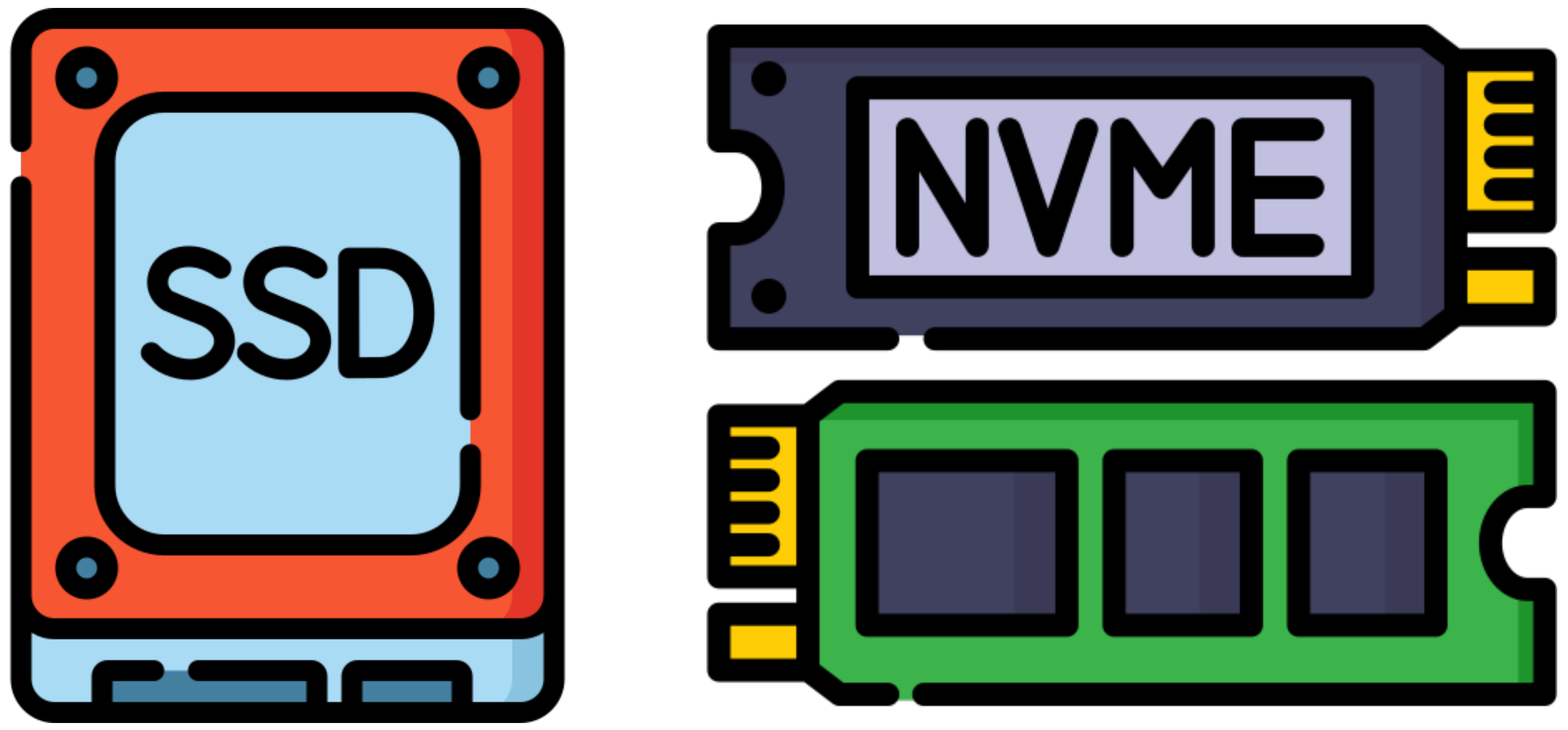
“The banana exists in only one city in the entire universe.”

“A rainy, windy city with vast oceans.”

“Hidden away within the buildings of the streets are a people known as the MacSysAdmins, who have long held a prophecy that a new Snow Leopard would come, a Final Version, which would lead them to true freedom.”

“The city is Gothenburg, also known as Göteborg.”

History of Disk Arbitration Vulnerabilities



kandji

Csaba Fitzl
X: @theevilbit

whoami

- Principal macOS Security Researcher @Kandji 🐝
- author of "macOS Exploitation" training @OffSec
- macOS bug hunter (~90 🍎 CVE)
- ex red/blue teamer
- husband, father
- hiking, trail running 🥾 🏔️

agenda

1. disk arbitration service
2. CVE-2017-2533 - LPE
3. CVE-2022-32780 - Sandbox Escape
4. CVE-2023-42838 - Sandbox Escape
5. ~~CVE-2024-40855 - TCC Bypass and Sandbox Escape~~
6. CVE-2024-27848 - storagekitd LPE via diskutil
7. conclusion

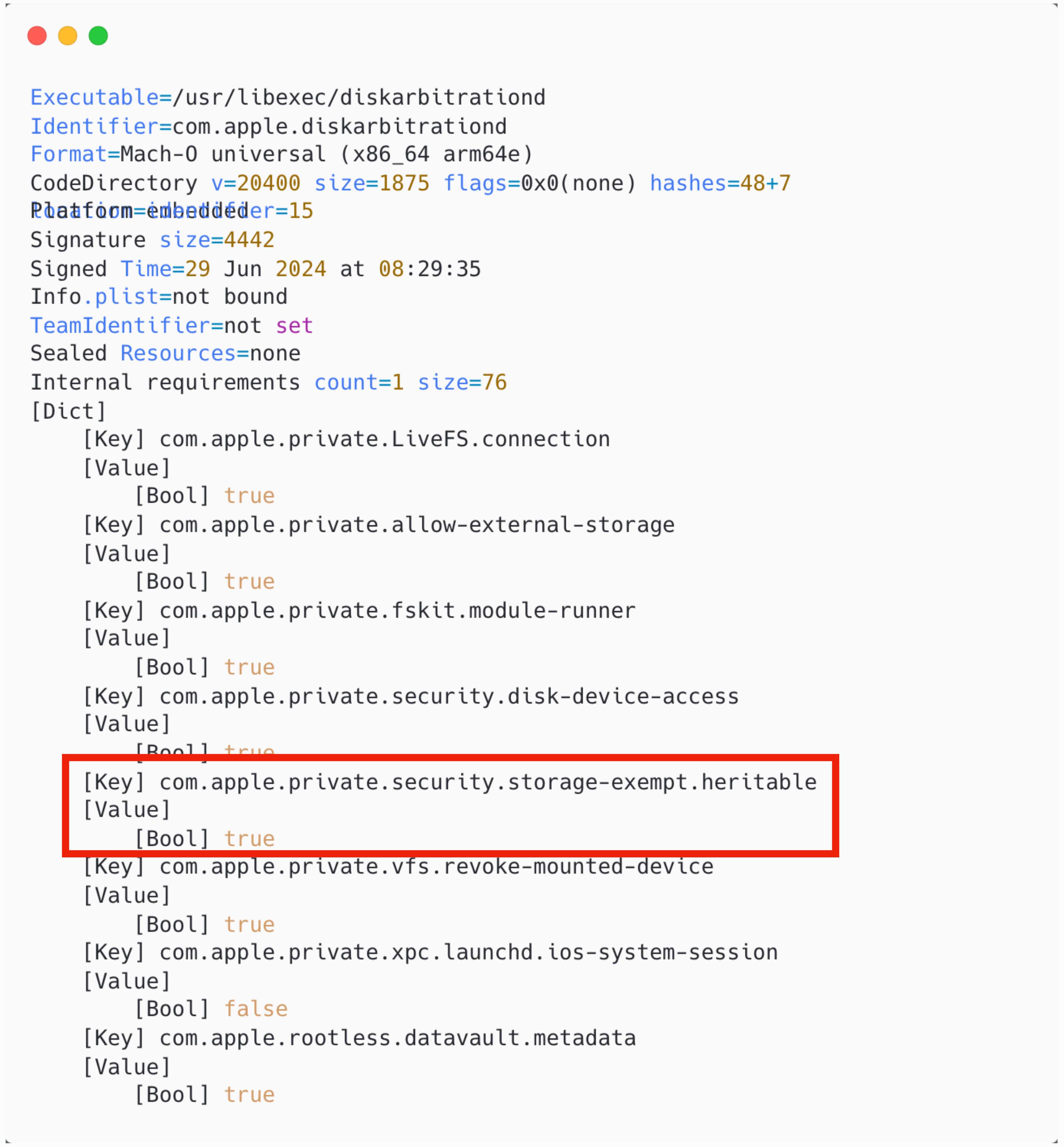
disk arbitration service

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 executables under the hood

diskarbitrationd - why we like it?

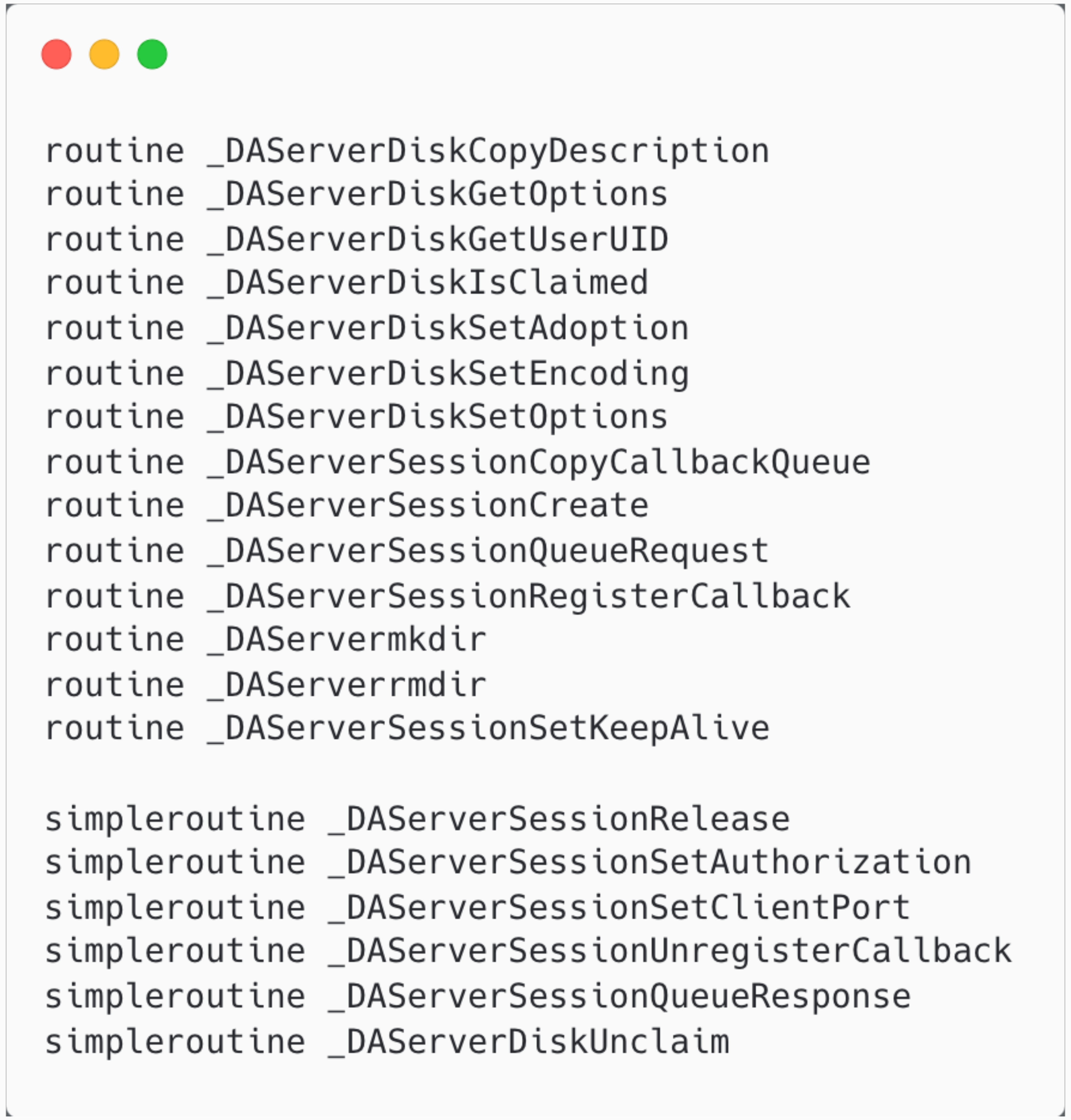
- runs as root
- unsandboxed
- ~ full disk access rights
- Mach service accessible from application sandbox
- opensource



```
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
Platform identifier=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
```

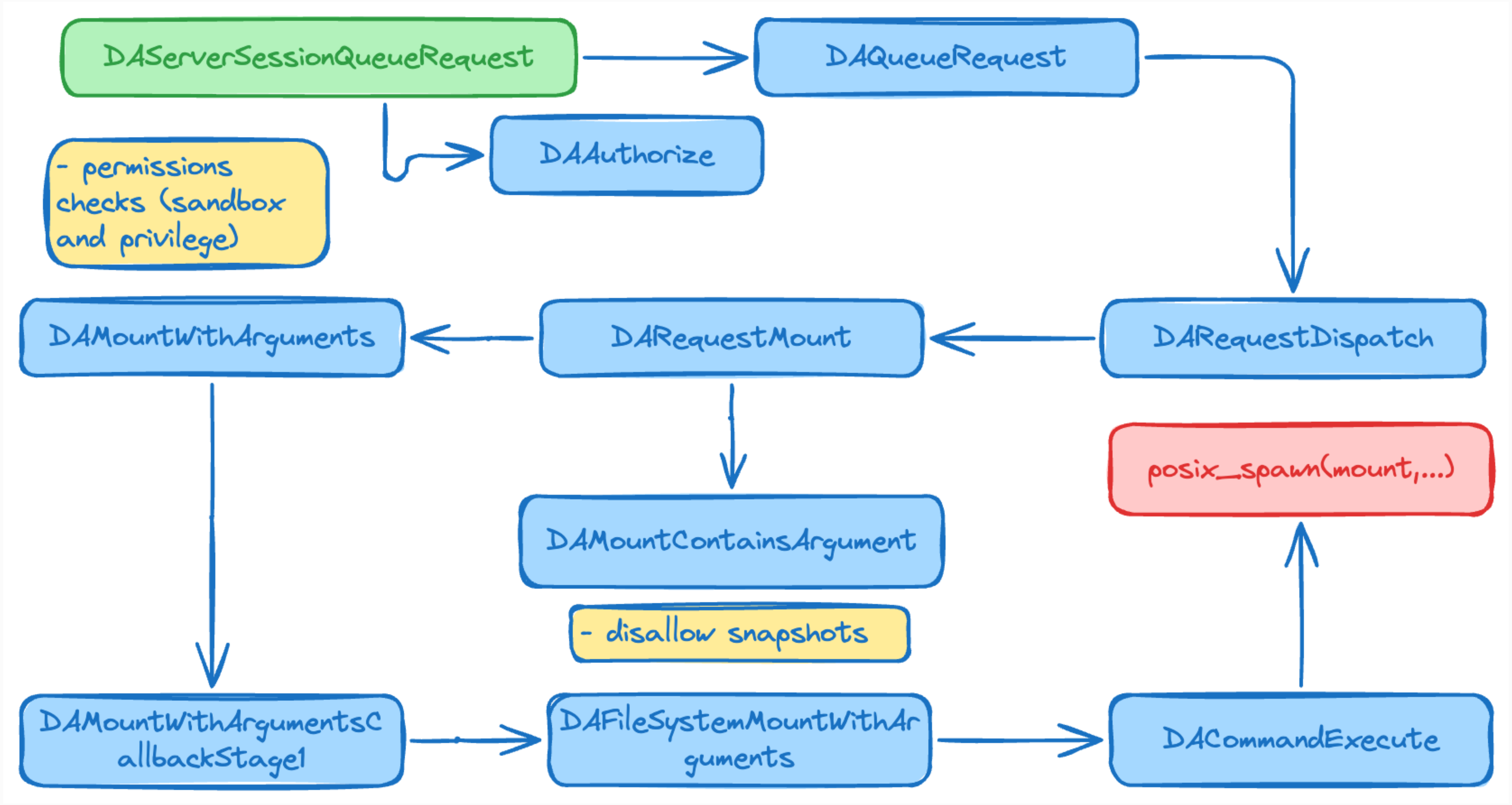
diskarbitrationd - MIG

- MIG service
- DA framework abstracts the MIG service

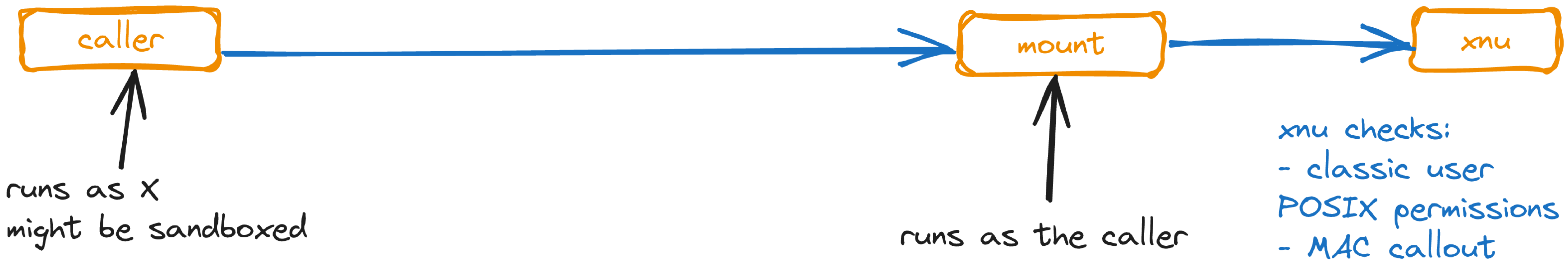


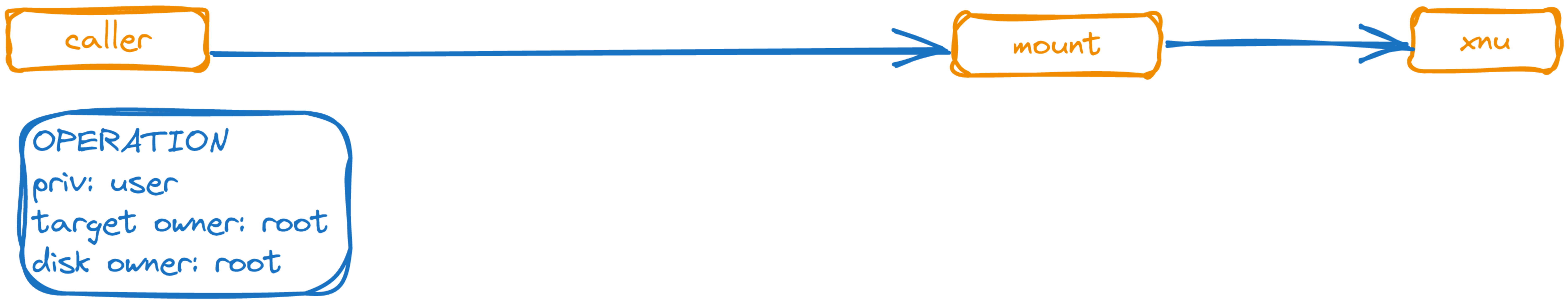
```
routine _DAServerDiskCopyDescription
routine _DAServerDiskGetOptions
routine _DAServerDiskGetUserUID
routine _DAServerDiskIsClaimed
routine _DAServerDiskSetAdoption
routine _DAServerDiskSetEncoding
routine _DAServerDiskSetOptions
routine _DAServerSessionCopyCallbackQueue
routine _DAServerSessionCreate
routine _DAServerSessionQueueRequest
routine _DAServerSessionRegisterCallback
routine _DAServermkdir
routine _DAServerrmdir
routine _DAServerSessionSetKeepAlive

simpleroutine _DAServerSessionRelease
simpleroutine _DAServerSessionSetAuthorization
simpleroutine _DAServerSessionSetClientPort
simpleroutine _DAServerSessionUnregisterCallback
simpleroutine _DAServerSessionQueueResponse
simpleroutine _DAServerDiskUnclaim
```

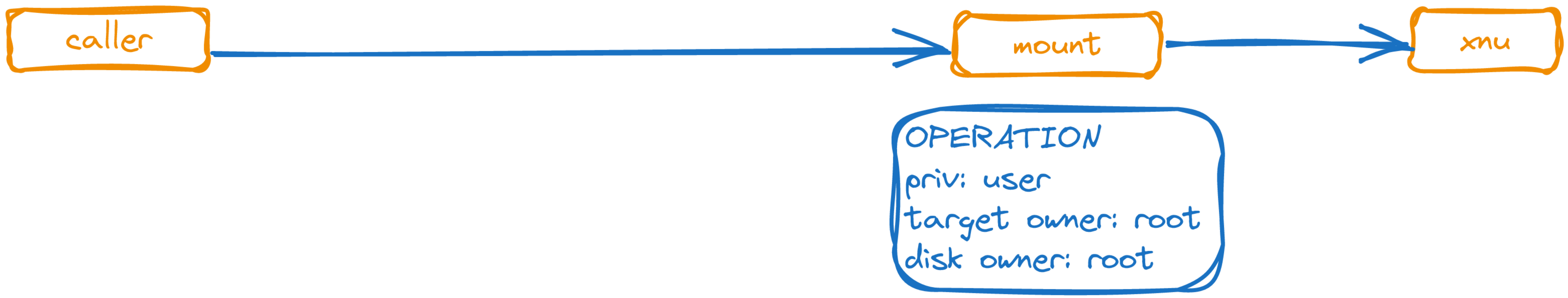
diskarbitrationd - call flow

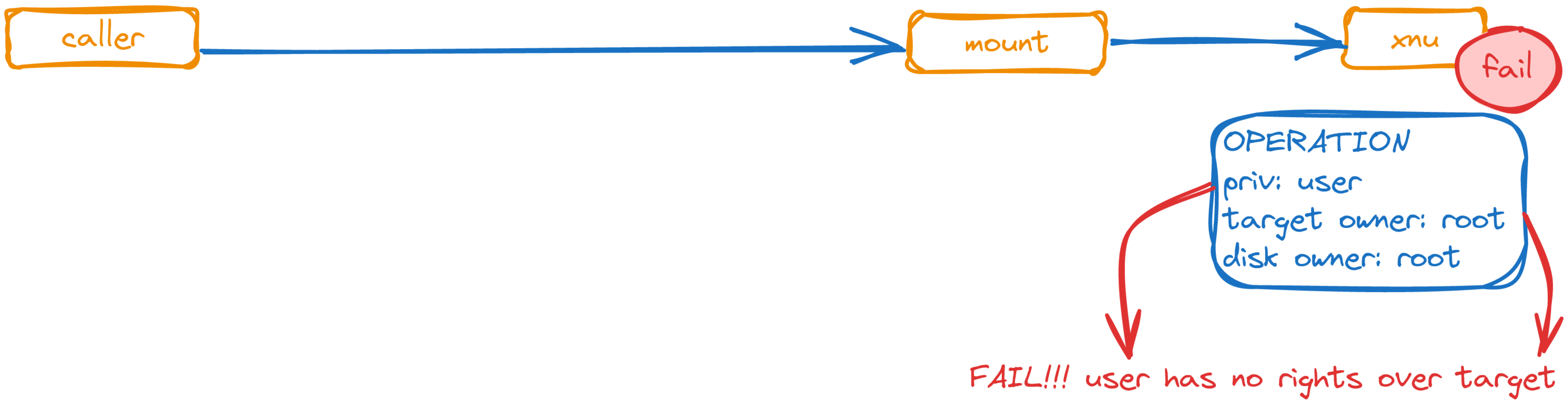


basic mount call


caller
mount
xnu
runs as X
might be sandboxed
runs as the caller
xnu checks:
- classic user POSIX permissions
- MAC callout


caller
mount
xnu
OPERATION
priv: user
target owner: root
disk owner: root


caller
mount
xnu
OPERATION
priv: user
target owner: root
disk owner: root


caller
mount
xnu
fail
OPERATION
priv: user
target owner: root
disk owner: root
FAIL!!! user has no rights over target

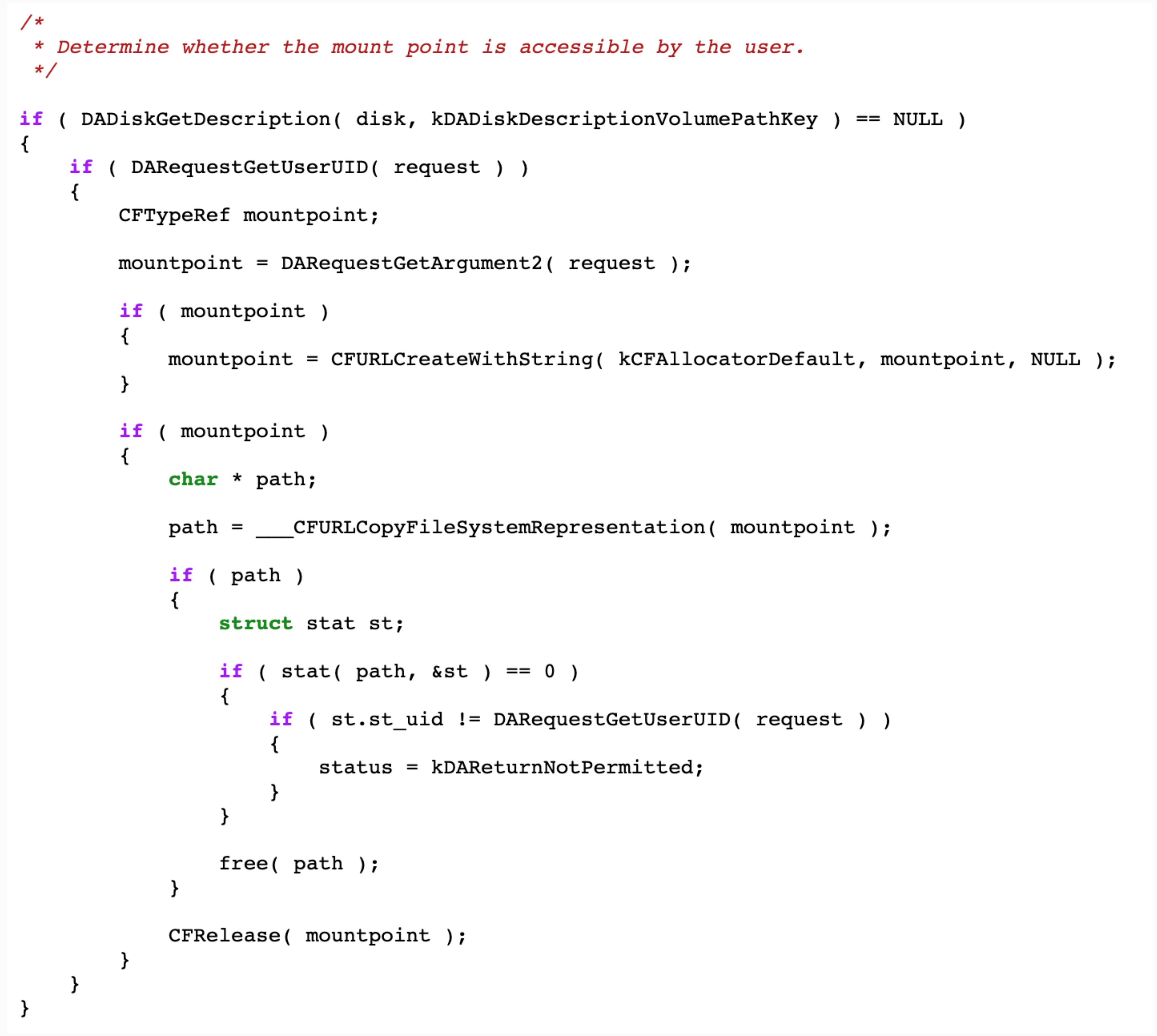
CVE-2017-2533 (pwn2own) - Mount yourself a root shell

CVE-2017-2533 - credits

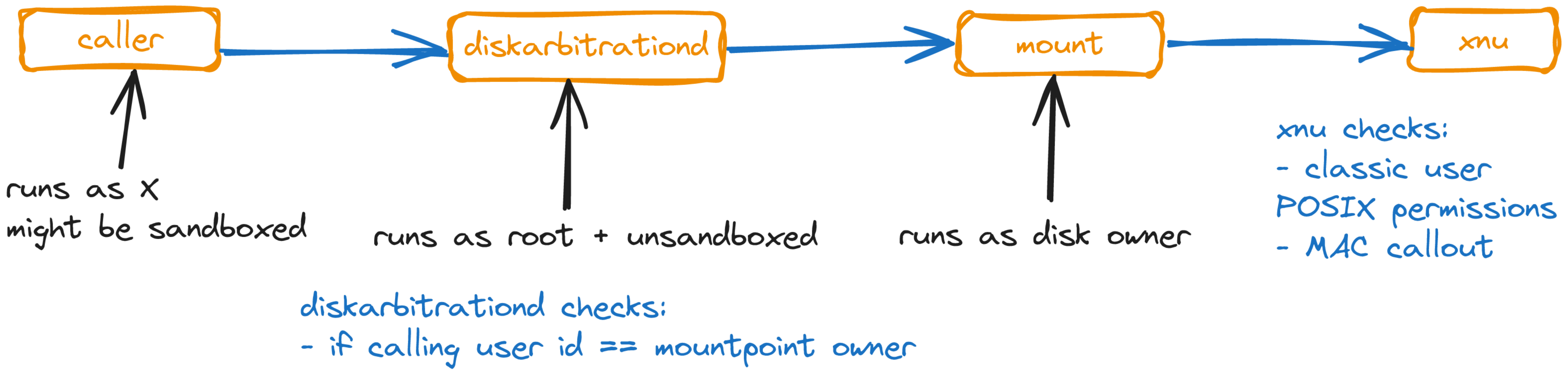
- was found by the "phoenhex" team, Niklas Baumstark and Samuel Groß
- part of the pwnown 2017 exploit chain
- details: https://phoenhex.re/2017-06-09/pwn2own-diskarbitrationd-privesc

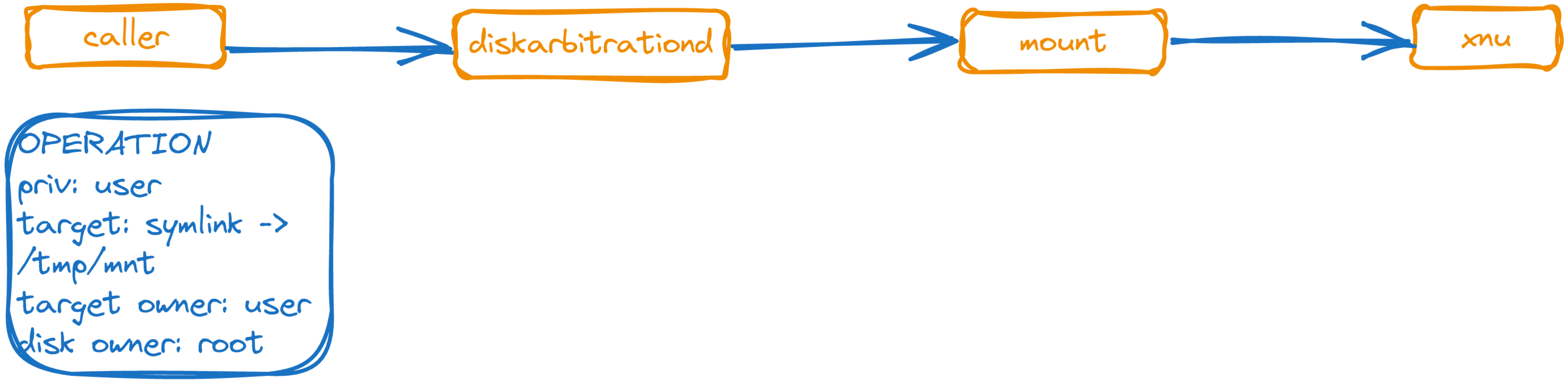
CVE-2017-2533 - the vulnerability

- disk arbitration service, (DARequest.c)
- check if mount point exists
- check if owned by the user (resolves path)
- no further checks
- TOCTOU (Time Of Check Time Of Use)

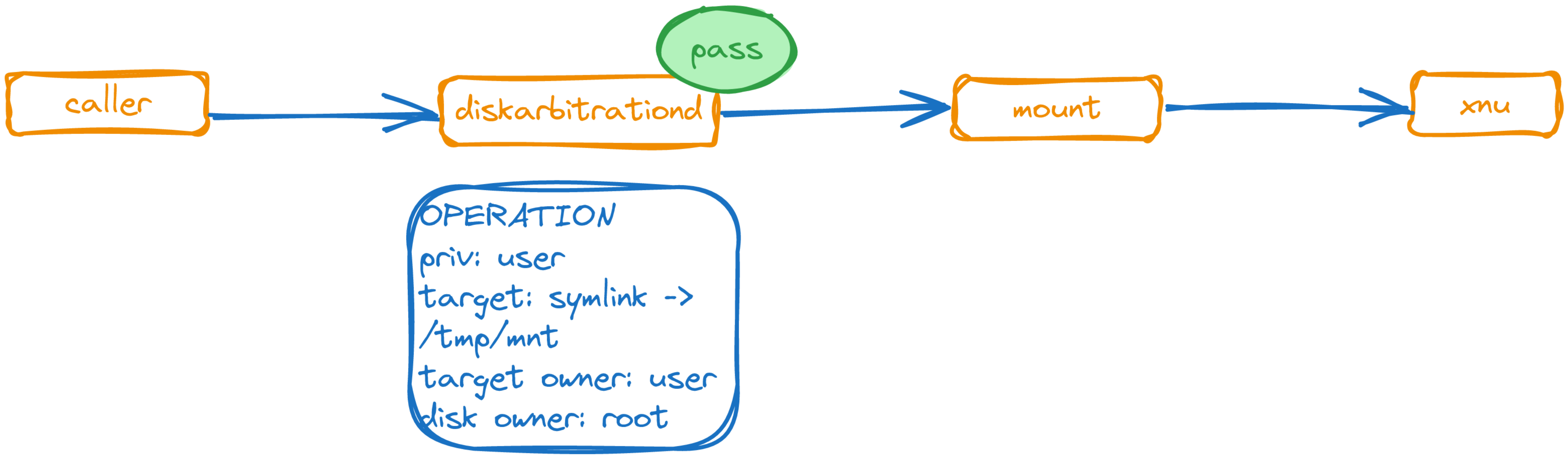


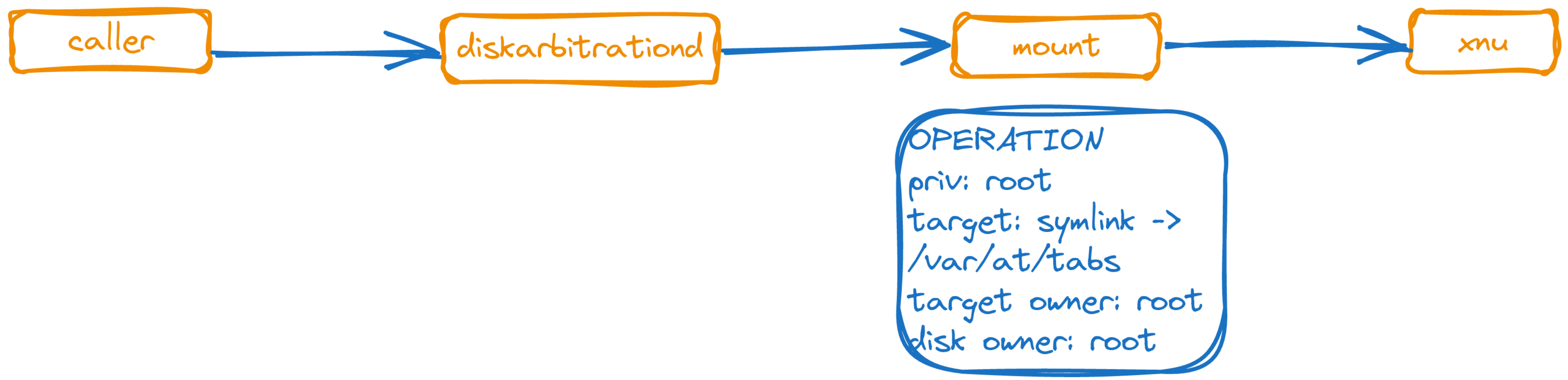
```
/*
 * Determine whether the mount point is accessible by the user.
 */

if ( DADiskGetDescription( disk, kDADiskDescriptionVolumePathKey ) == NULL )
{
    if ( DARequestGetUserUID( request ) )
    {
        CFTypeRef mountpoint;

        mountpoint = DARequestGetArgument2( request );

        if ( mountpoint )
        {
            mountpoint = CFURLCreateWithString( kCFAllocatorDefault, mountpoint, NULL );
        }

        if ( mountpoint )
        {
            char * path;

            path = ___CFURLCopyFileSystemRepresentation( mountpoint );

            if ( path )
            {
                struct stat st;

                if ( stat( path, &st ) == 0 )
                {
                    if ( st.st_uid != DARequestGetUserUID( request ) )
                    {
                        status = kDAReturnNotPermitted;
                    }
                }

                free( path );
            }

            CFRelease( mountpoint );
        }
    }
}
```

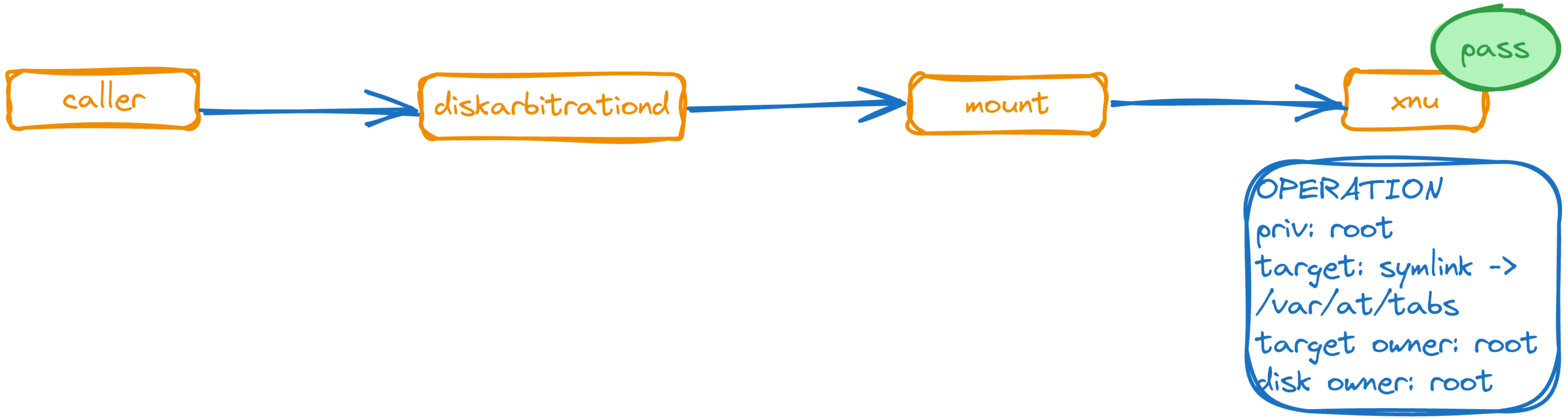

caller
diskarbitrationd
mount
xnu
runs as X
might be sandboxed
runs as root + unsandboxed
runs as disk owner
xnu checks:
- classic user POSIX permissions
- MAC callout
diskarbitrationd checks:
- if calling user id == mountpoint owner

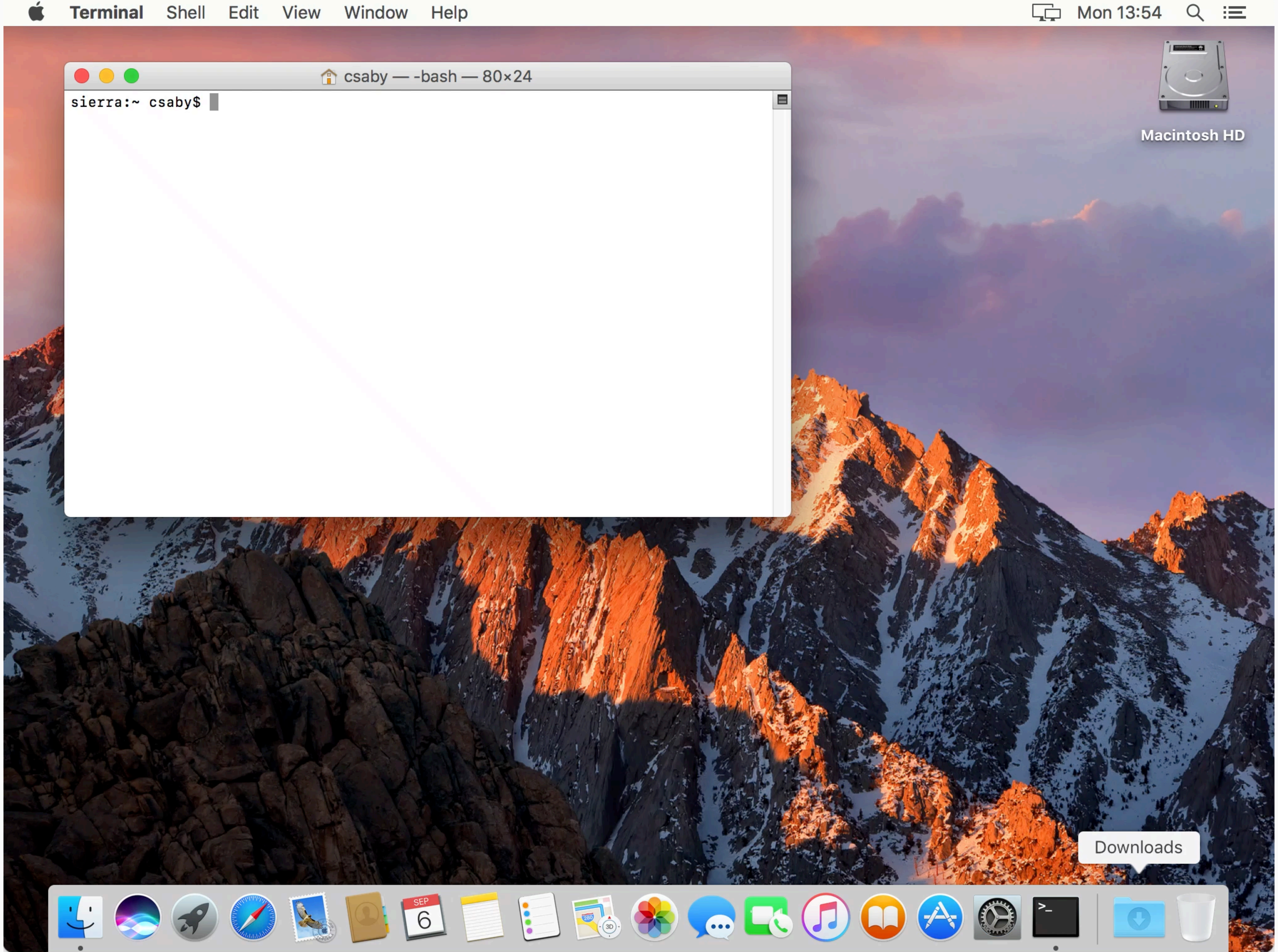


OPERATION
priv: user
target: symlink -> /tmp/mnt
target owner: user
disk owner: root


caller
diskarbitrationd
pass
mount
xnu
OPERATION
priv: user
target: symlink ->
/tmp/mnt
target owner: user
disk owner: root


caller
diskarbitrationd
mount
xnu
OPERATION
priv: root
target: symlink ->
/var/at/tabs
target owner: root
disk owner: root

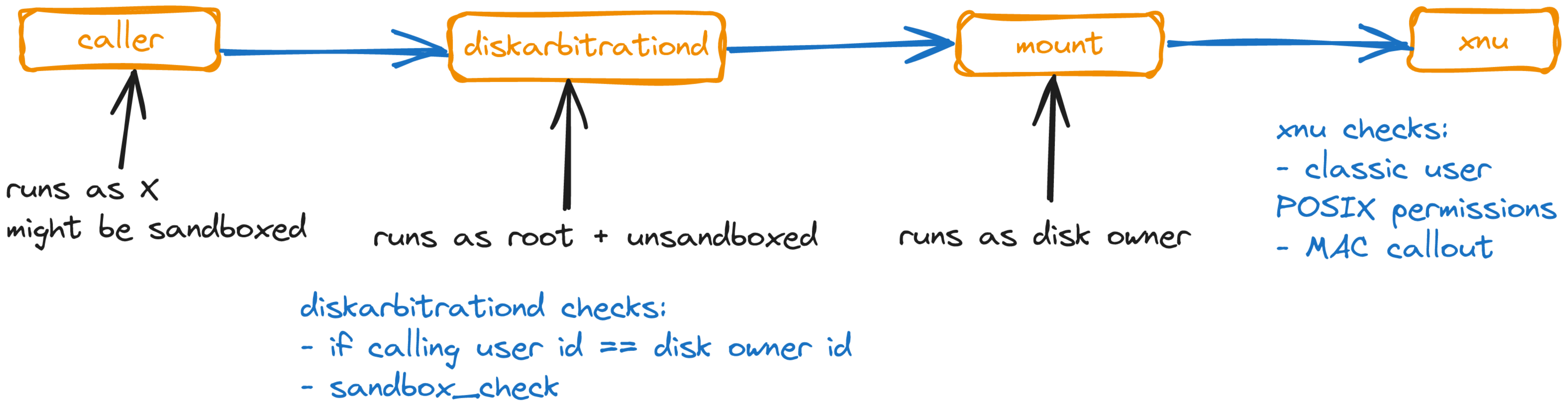

caller
diskarbitrationd
mount
xnu
pass
OPERATION
priv: root
target: symlink ->
/var/at/tabs
target owner: root
disk owner: root



```
sierra:~ csaby$
```

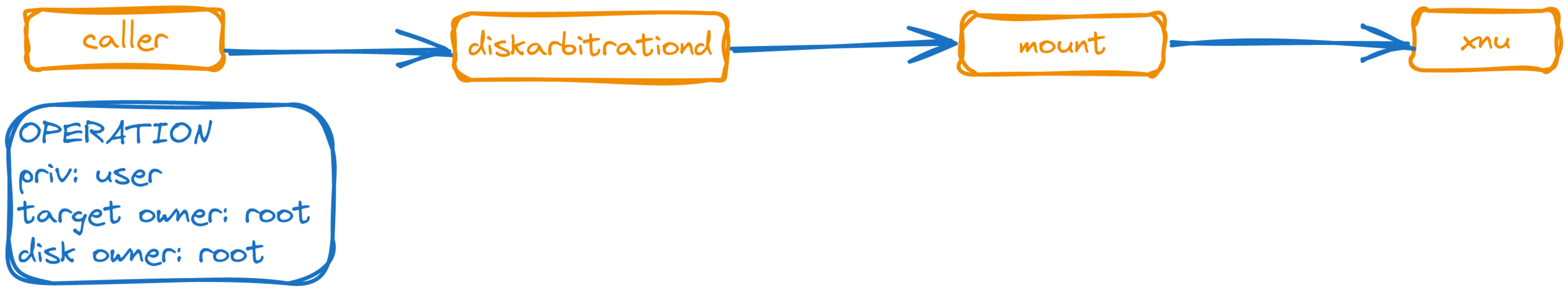
the fix

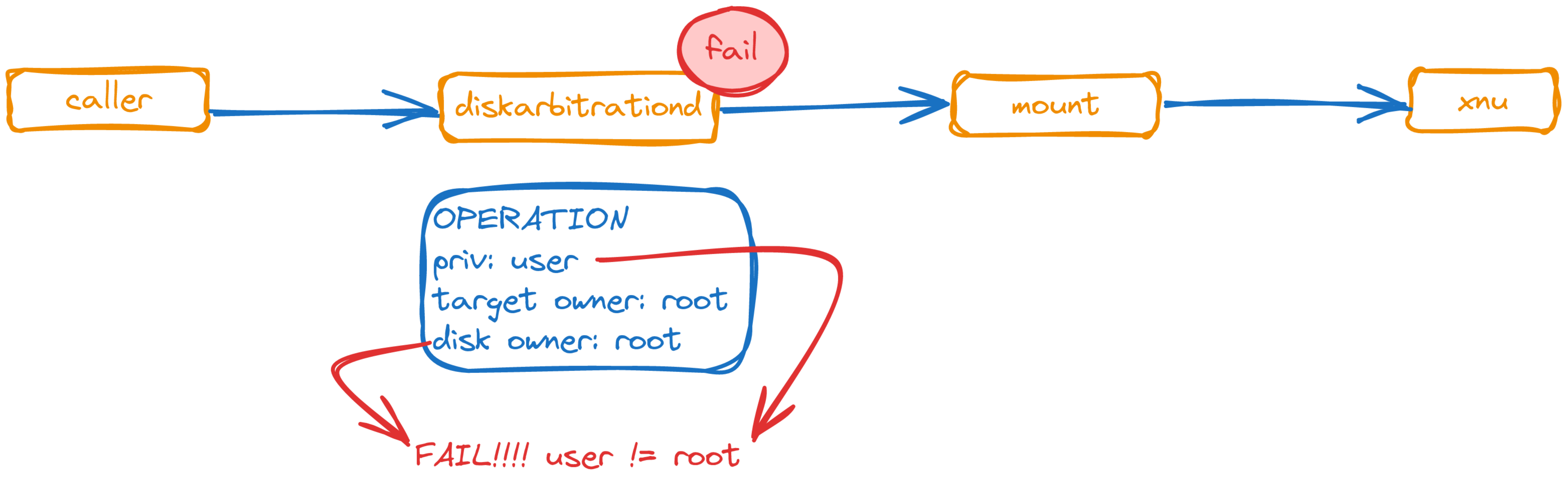
- IMO one of the best fixes Apple ever did
- caller uid == disk owner (+ call mount as the disk owner)

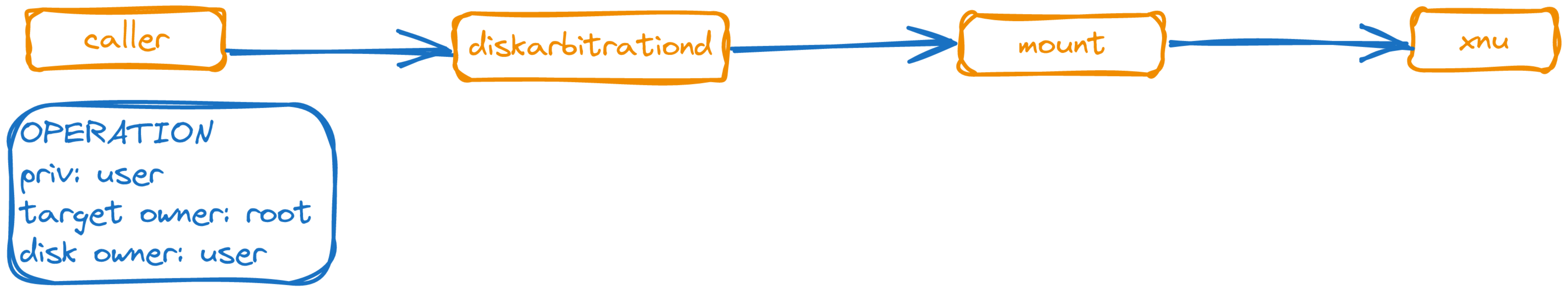


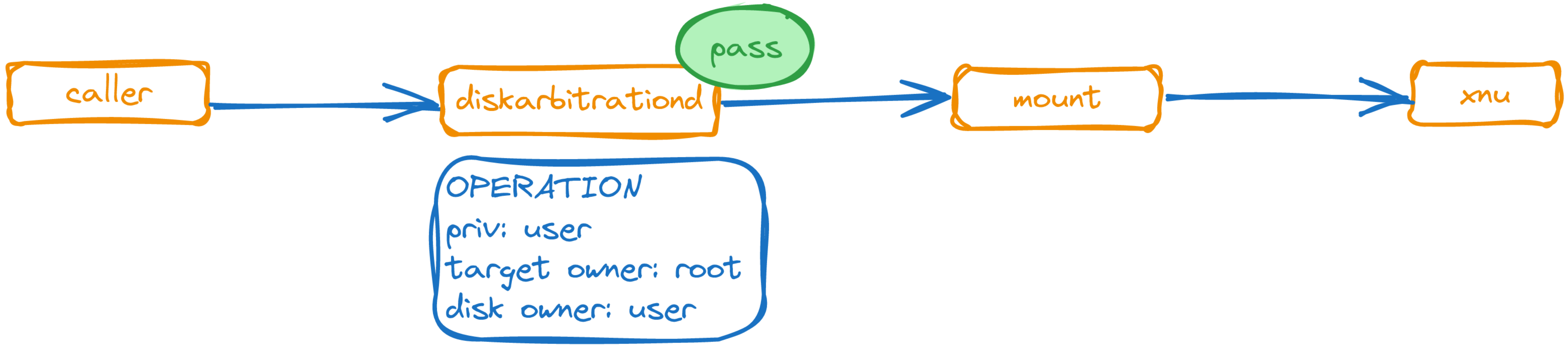
xnu checks:
- classic user POSIX permissions
- MAC callout

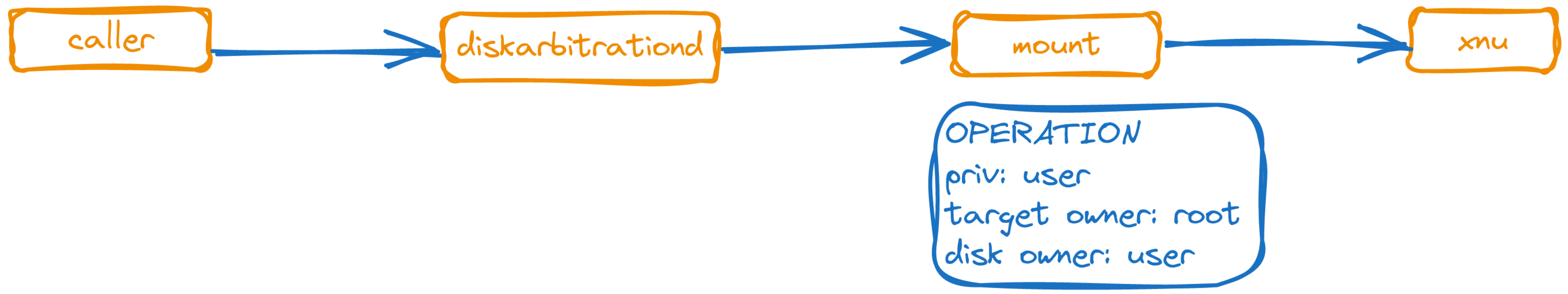
diskarbitrationd checks:
- if calling user id == disk owner id
- sandbox_check

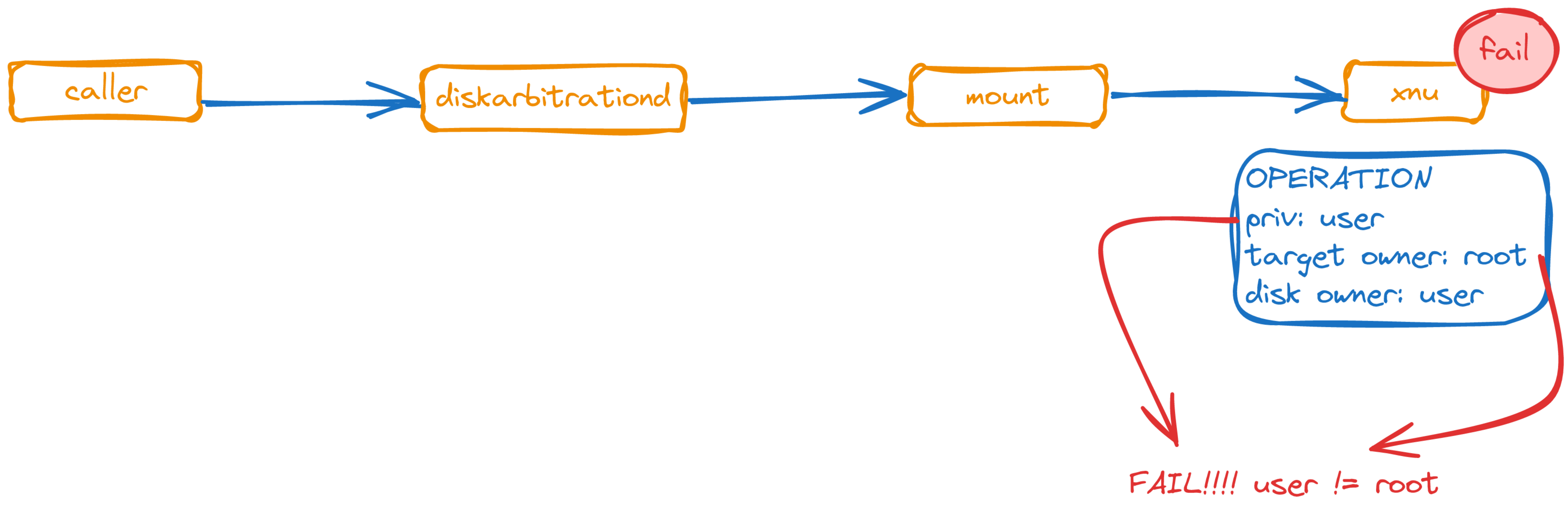

caller
diskarbitrationd
mount
xnu
OPERATION
priv: user
target owner: root
disk owner: root


caller
diskarbitrationd
fail
mount
xnu
OPERATION
priv: user
target owner: root
disk owner: root
FAIL!!!! user != root


caller
diskarbitrationd
mount
xnu
OPERATION
priv: user
target owner: root
disk owner: user


caller
diskarbitrationd
pass
mount
xnu
OPERATION
priv: user
target owner: root
disk owner: user

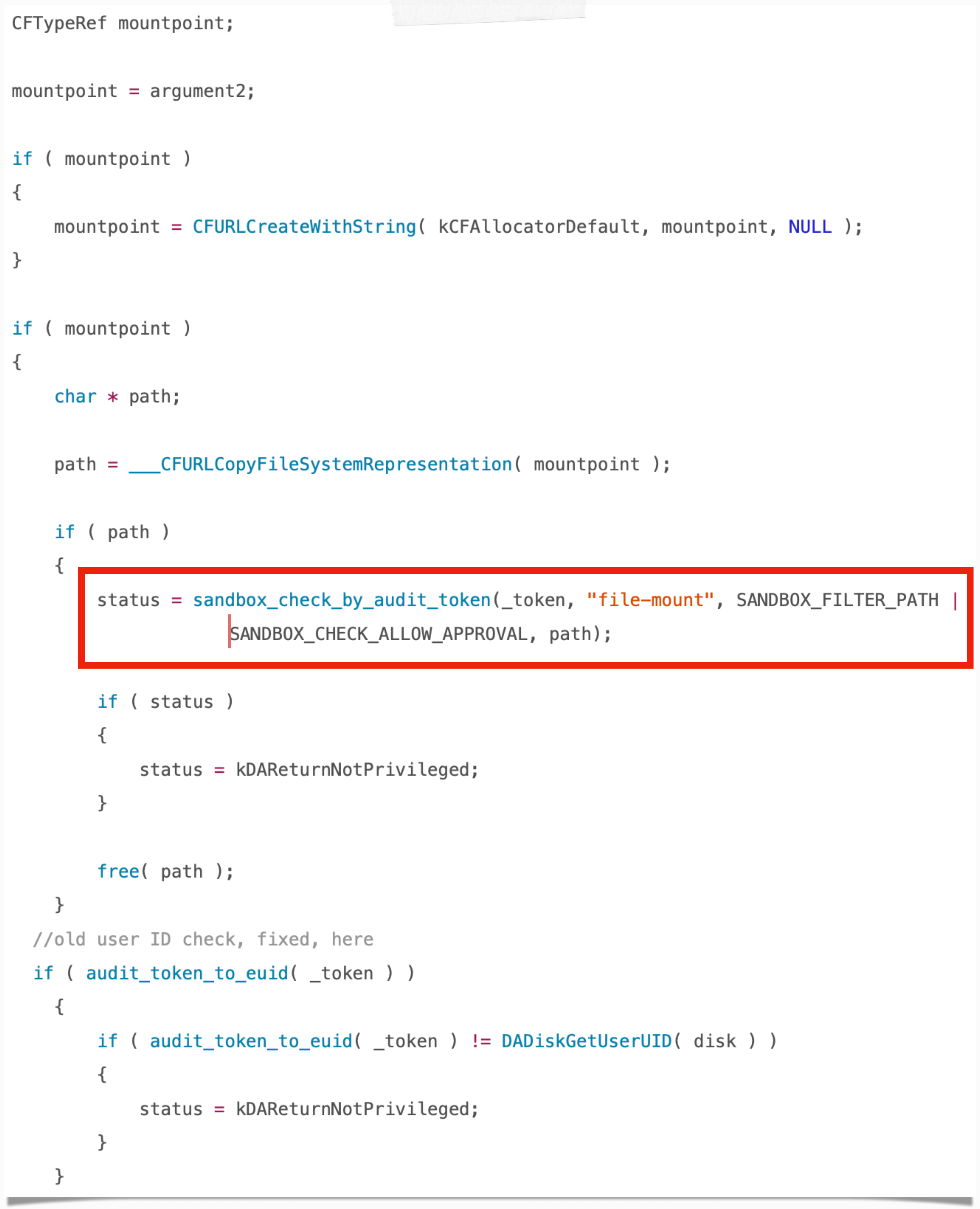

caller
diskarbitrationd
mount
xnu
OPERATION
priv: user
target owner: root
disk owner: user


caller
diskarbitrationd
mount
xnu
fail
OPERATION
priv: user
target owner: root
disk owner: user
FAIL!!!! user != root

CVE-2022-32780 - Sandbox Escape

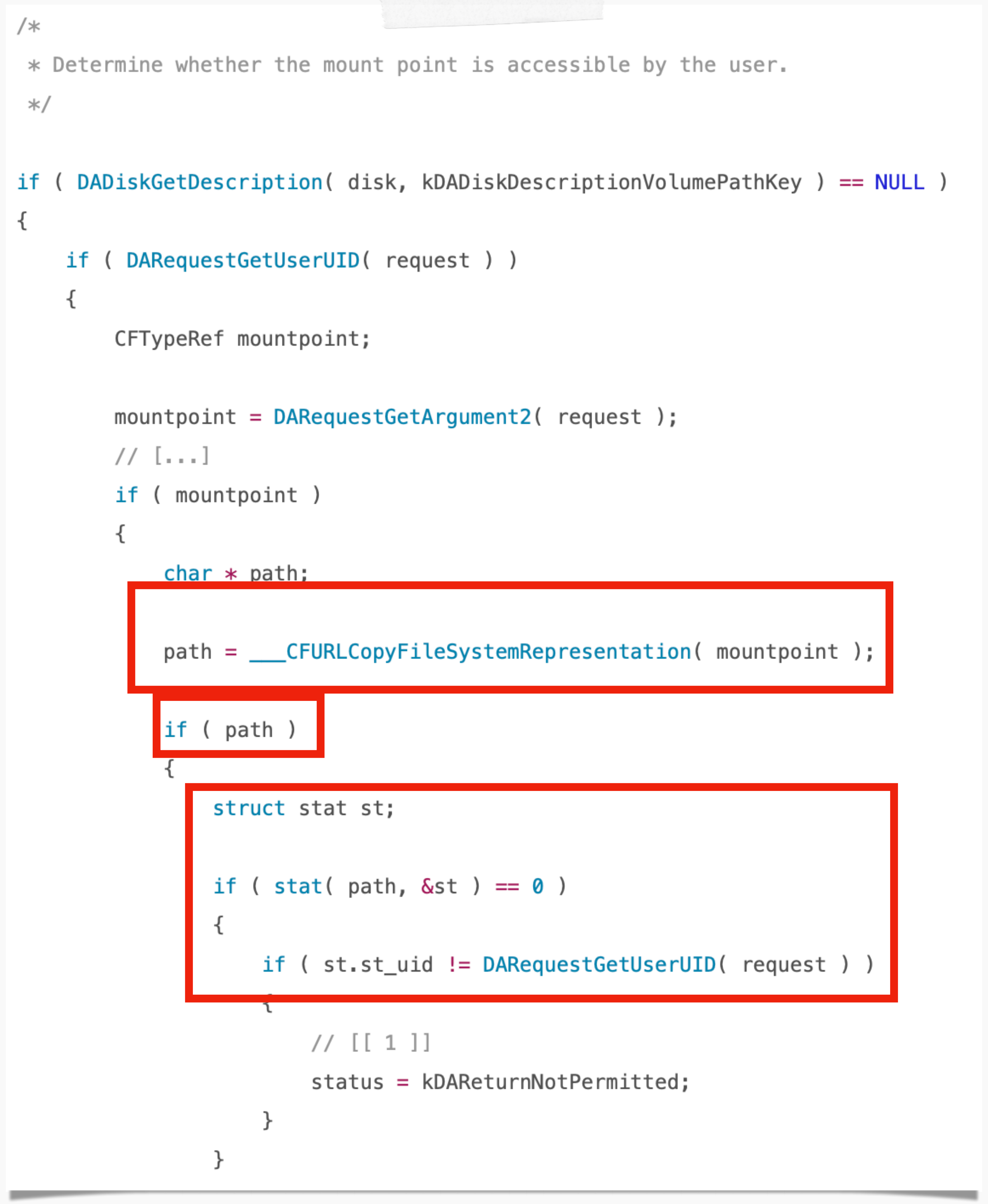
CVE-2022-32780

- Apple moved the check into DAServer.c - "_DAServerSessionQueueRequest"
- sandbox check by "sandbox_check_by_audit_token"

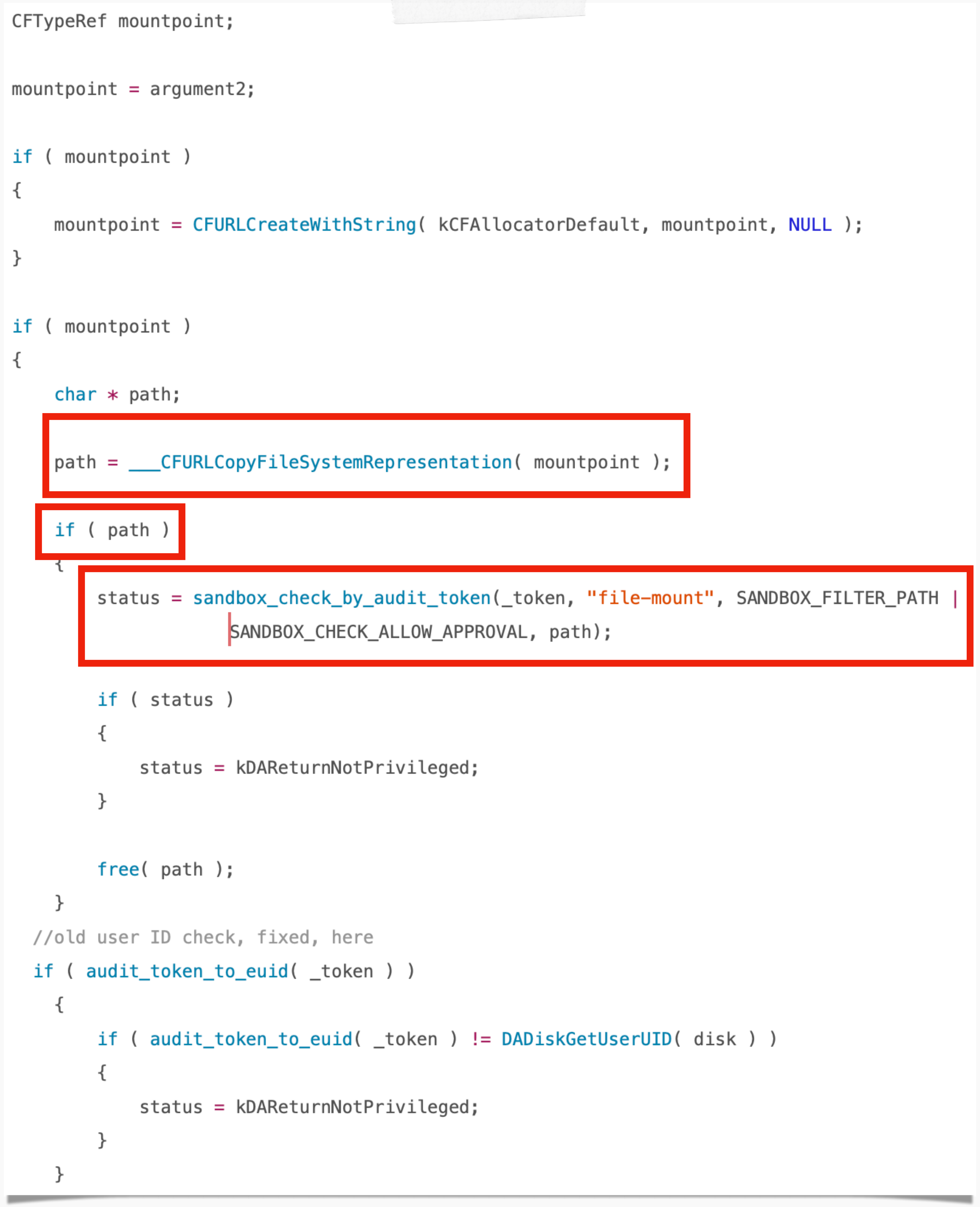


```
CFTypeRef mountpoint;

mountpoint = argument2;

if ( mountpoint )
{
    mountpoint = CFURLCreateWithString( kCFAllocatorDefault, mountpoint, NULL );
}

if ( mountpoint )
{
    char * path;

    path = ___CFURLCopyFileSystemRepresentation( mountpoint );

    if ( path )
    {
        status = sandbox_check_by_audit_token(_token, "file-mount", SANDBOX_FILTER_PATH |
                    SANDBOX_CHECK_ALLOW_APPROVAL, path);

        if ( status )
        {
            status = kDAReturnNotPrivileged;
        }

        free( path );
    }
  //old user ID check, fixed, here
  if ( audit_token_to_euid( _token ) )
    {
        if ( audit_token_to_euid( _token ) != DADiskGetUserUID( disk ) )
        {
            status = kDAReturnNotPrivileged;
        }
    }
```

CVE-2022-32780 - old vs new

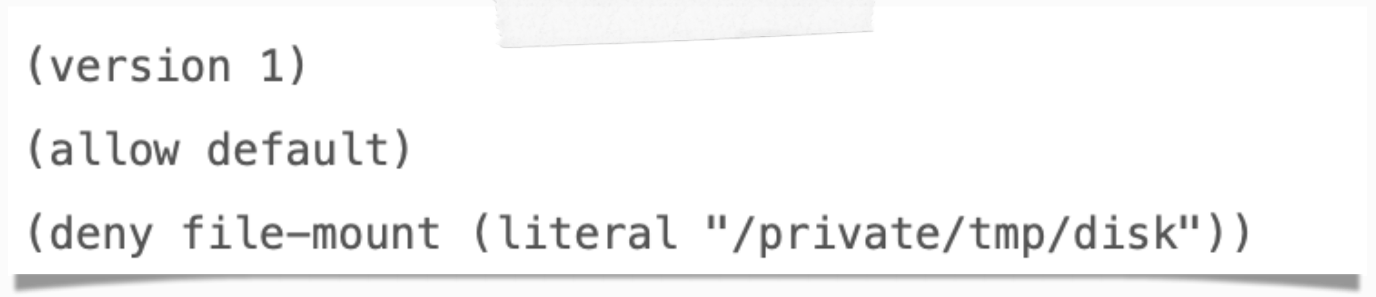


```
/*
 * Determine whether the mount point is accessible by the user.
 */

if ( DADiskGetDescription( disk, kDADiskDescriptionVolumePathKey ) == NULL )
{
    if ( DARequestGetUserUID( request ) )
    {
        CFTypeRef mountpoint;

        mountpoint = DARequestGetArgument2( request );
        // [...]
        if ( mountpoint )
        {
            char * path;

            path = ___CFURLCopyFileSystemRepresentation( mountpoint );

            if ( path )
            {
                struct stat st;

                if ( stat( path, &st ) == 0 )
                {
                    if ( st.st_uid != DARequestGetUserUID( request ) )
                    {
                        // [[ 1 ]]
                        status = kDAReturnNotPermitted;
                    }
                }
```

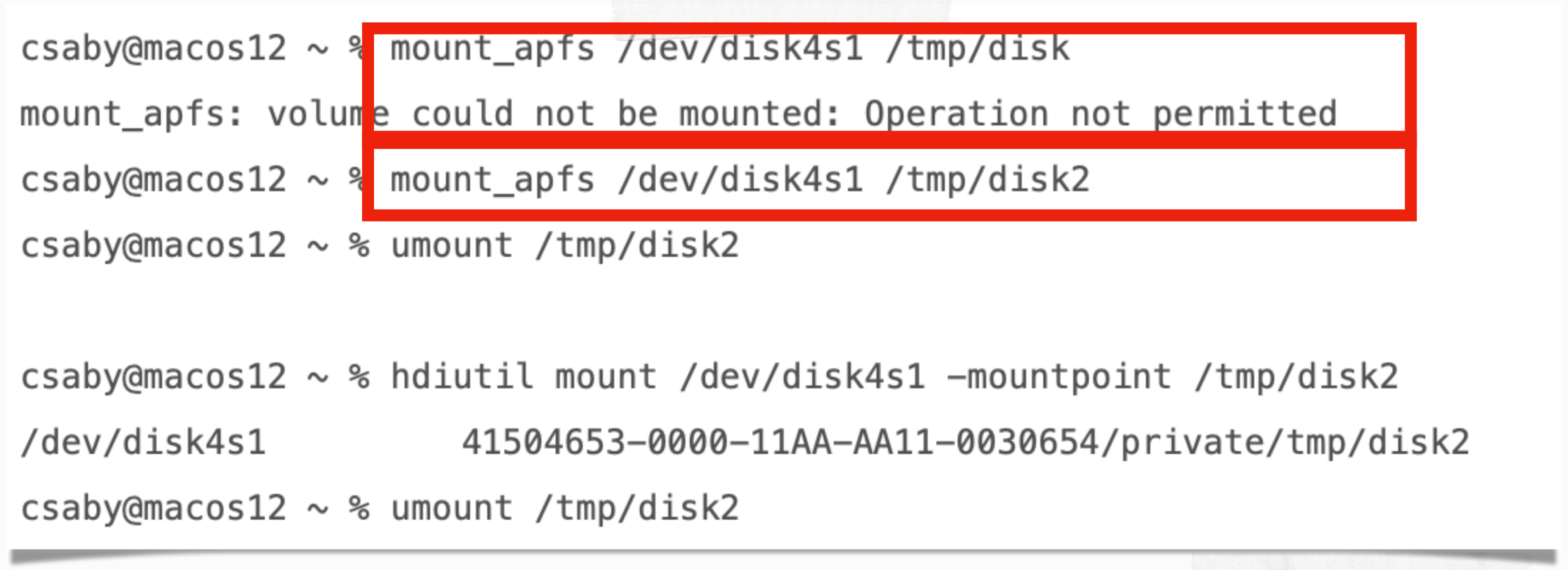


```
CFTypeRef mountpoint;

mountpoint = argument2;

if ( mountpoint )
{
    mountpoint = CFURLCreateWithString( kCFAllocatorDefault, mountpoint, NULL );
}

if ( mountpoint )
{
    char * path;

    path = ___CFURLCopyFileSystemRepresentation( mountpoint );

    if ( path )
    {
        status = sandbox_check_by_audit_token(_token, "file-mount", SANDBOX_FILTER_PATH |
                    SANDBOX_CHECK_ALLOW_APPROVAL, path);

        if ( status )
        {
            status = kDAReturnNotPrivileged;
        }

        free( path );
    }
  //old user ID check, fixed, here
  if ( audit_token_to_euid( _token ) )
    {
        if ( audit_token_to_euid( _token ) != DADiskGetUserUID( disk ) )
        {
            status = kDAReturnNotPrivileged;
        }
    }
```

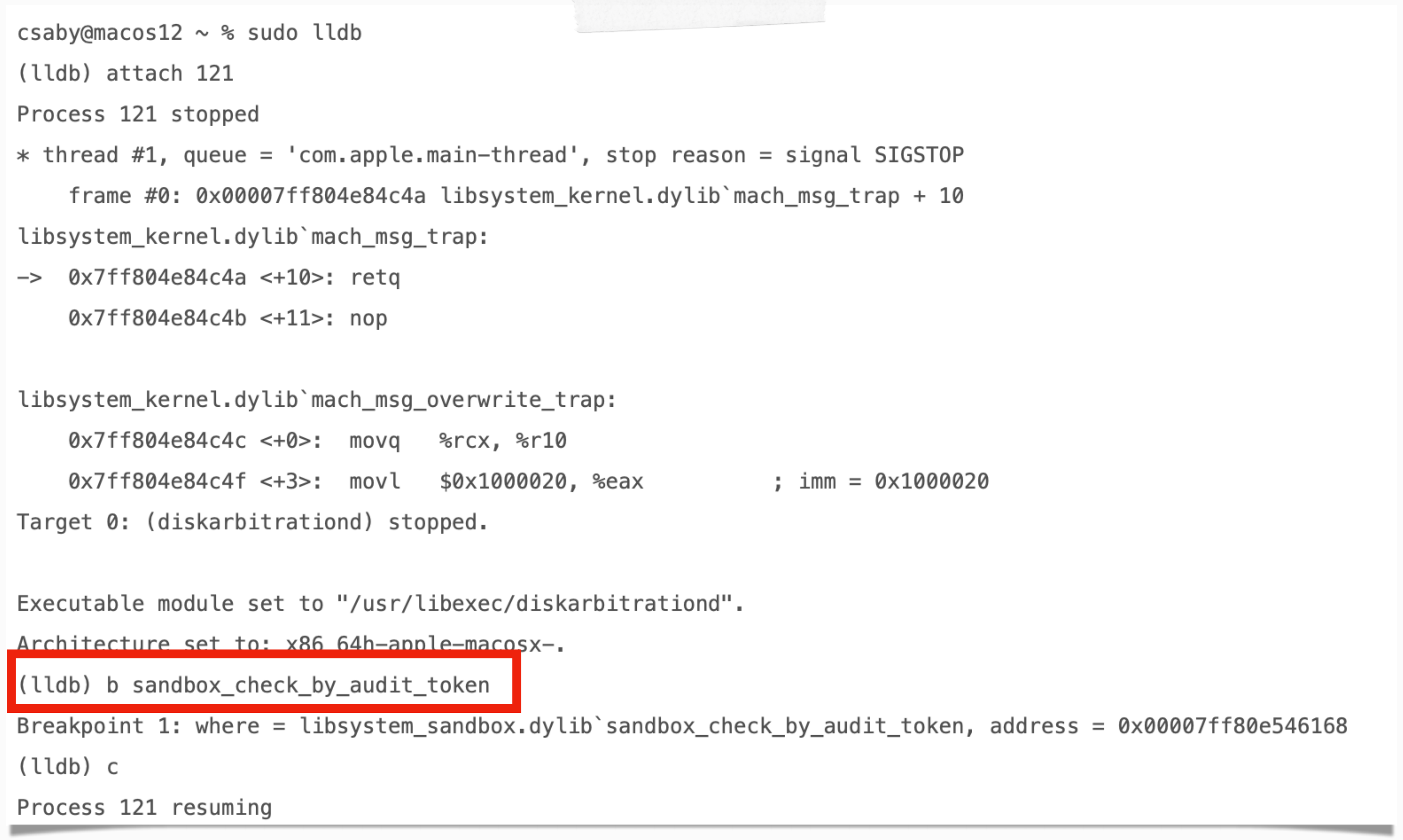
CVE-2022-32780 - Testing



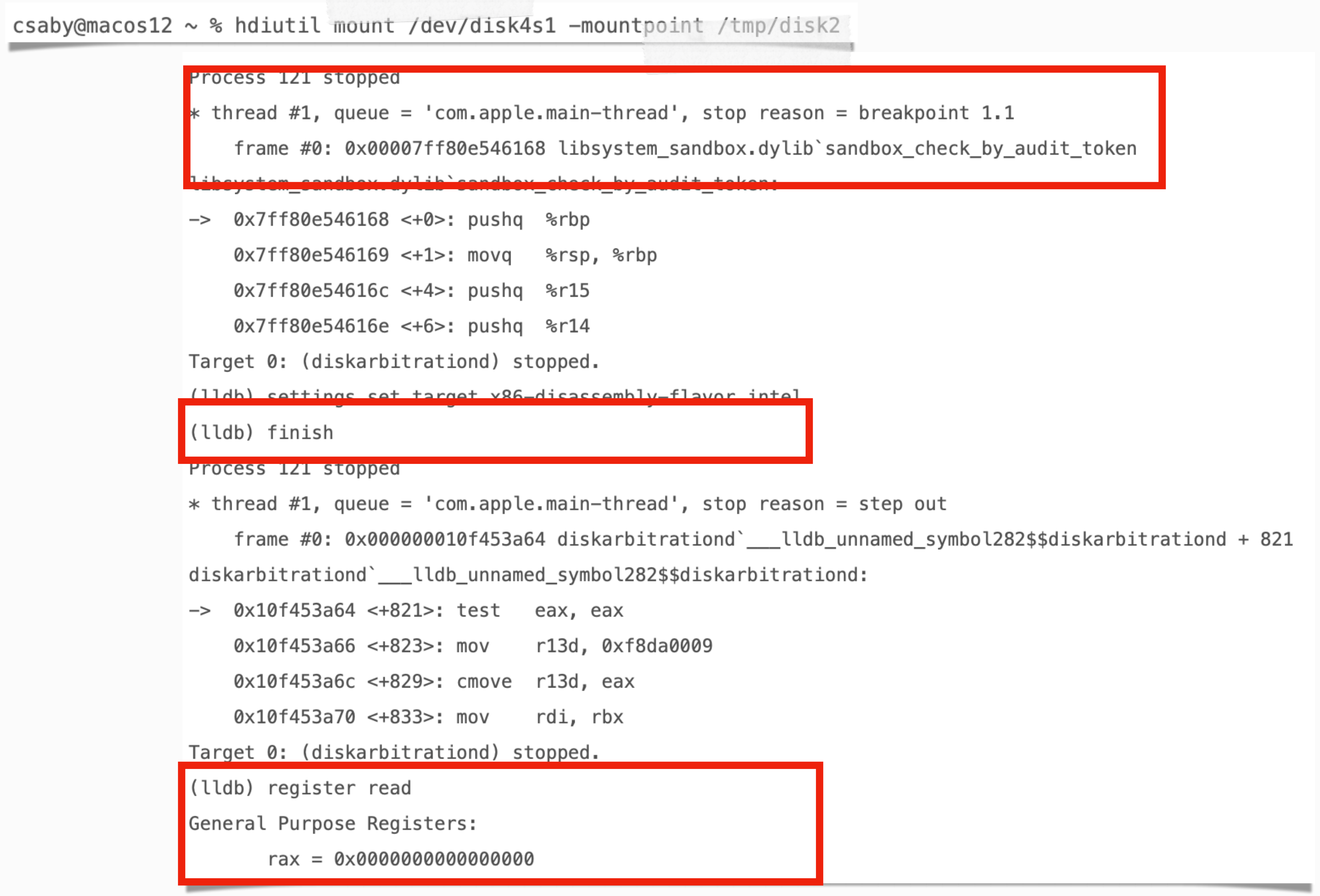
```
(version 1)
(allow default)
(deny file-mount (literal "/private/tmp/disk"))
```



```
csaby@macos12 ~ % mount_apfs /dev/disk4s1 /tmp/disk
mount_apfs: volume could not be mounted: Operation not permitted
csaby@macos12 ~ % mount_apfs /dev/disk4s1 /tmp/disk2
csaby@macos12 ~ % umount /tmp/disk2

csaby@macos12 ~ % hdiutil mount /dev/disk4s1 -mountpoint /tmp/disk2
/dev/disk4s1          41504653-0000-11AA-AA11-0030654/private/tmp/disk2
csaby@macos12 ~ % umount /tmp/disk2
```



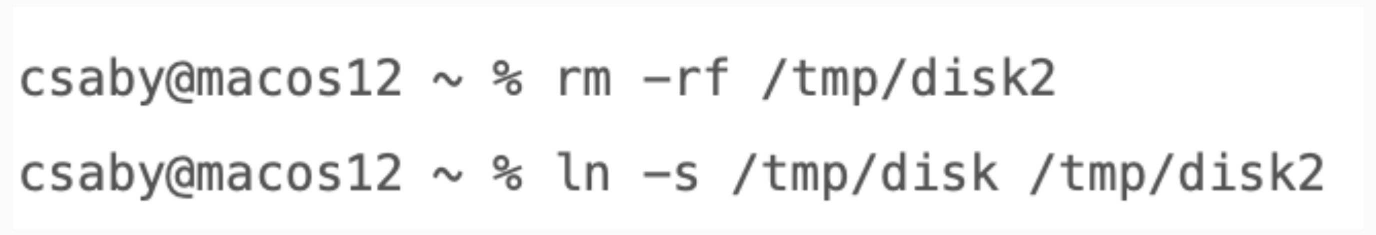
```
csaby@macos12 ~ % sudo lldb
(lldb) attach 121
Process 121 stopped
* thread #1, queue = 'com.apple.main-thread', stop reason = signal SIGSTOP
    frame #0: 0x00007ff804e84c4a libsystem_kernel.dylib`mach_msg_trap + 10
libsystem_kernel.dylib`mach_msg_trap:
->  0x7ff804e84c4a <+10>: retq
    0x7ff804e84c4b <+11>: nop

libsystem_kernel.dylib`mach_msg_overwrite_trap:
    0x7ff804e84c4c <+0>:  movq   %rcx, %r10
    0x7ff804e84c4f <+3>:  movl   $0x1000020, %eax          ; imm = 0x1000020
Target 0: (diskarbitrationd) stopped.

Executable module set to "/usr/libexec/diskarbitrationd".
Architecture set to: x86_64h-apple-macosx-.
(lldb) b sandbox_check_by_audit_token
Breakpoint 1: where = libsystem_sandbox.dylib`sandbox_check_by_audit_token, address = 0x00007ff80e546168
(lldb) c
Process 121 resuming
```



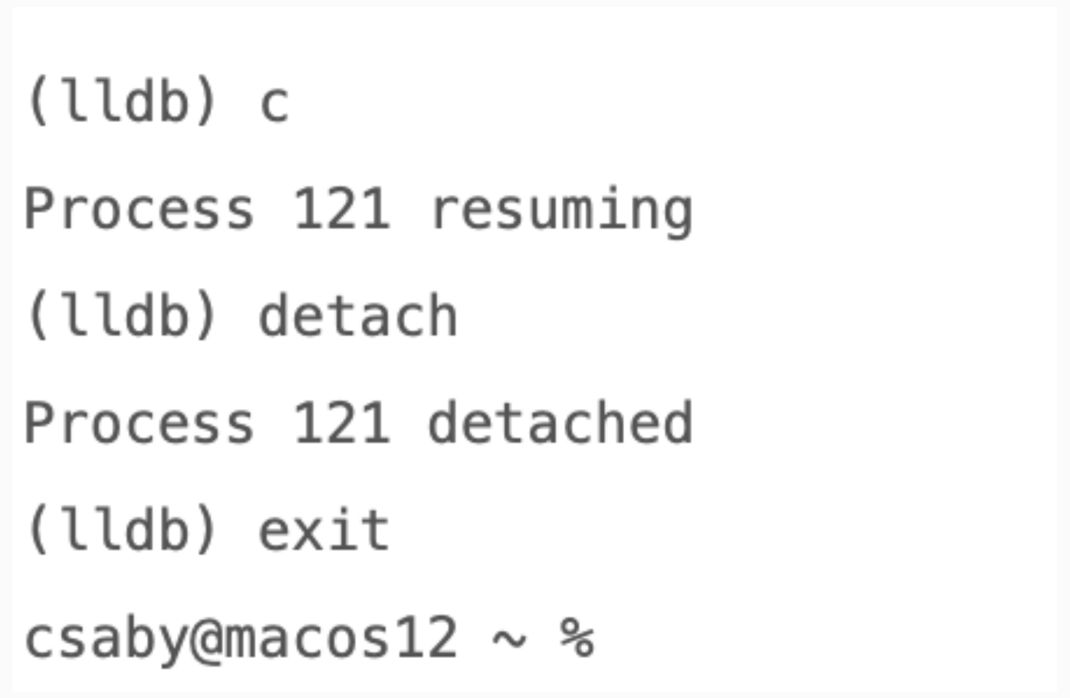
```
csaby@macos12 ~ % hdiutil mount /dev/disk4s1 -mountpoint /tmp/disk2
```

```
Process 121 stopped
* thread #1, queue = 'com.apple.main-thread', stop reason = breakpoint 1.1
    frame #0: 0x00007ff80e546168 libsystem_sandbox.dylib`sandbox_check_by_audit_token
libsystem_sandbox.dylib`sandbox_check_by_audit_token:
->  0x7ff80e546168 <+0>: pushq  %rbp
    0x7ff80e546169 <+1>: movq   %rsp, %rbp
    0x7ff80e54616c <+4>: pushq  %r15
    0x7ff80e54616e <+6>: pushq  %r14
Target 0: (diskarbitrationd) stopped.
(lldb) settings set target.x86-disassembly-flavor intel
(lldb) finish
Process 121 stopped
* thread #1, queue = 'com.apple.main-thread', stop reason = step out
    frame #0: 0x000000010f453a64 diskarbitrationd`___lldb_unnamed_symbol282$$diskarbitrationd + 821
diskarbitrationd`___lldb_unnamed_symbol282$$diskarbitrationd:
->  0x10f453a64 <+821>: test   eax, eax
    0x10f453a66 <+823>: mov    r13d, 0xf8da0009
    0x10f453a6c <+829>: cmove  r13d, eax
    0x10f453a70 <+833>: mov    rdi, rbx
Target 0: (diskarbitrationd) stopped.
(lldb) register read
General Purpose Registers:
       rax = 0x0000000000000000
```

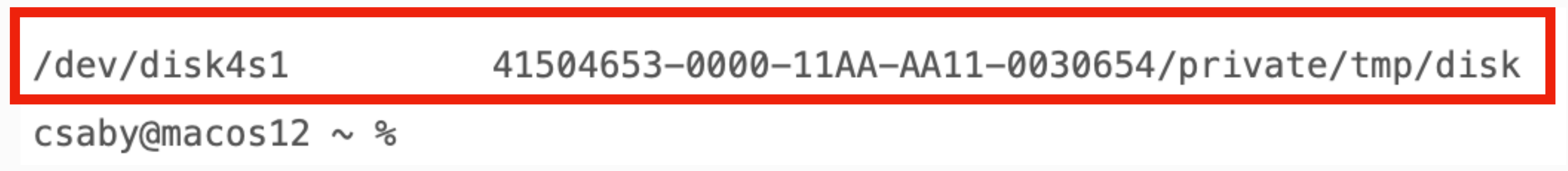
CVE-2022-32780 - Testing



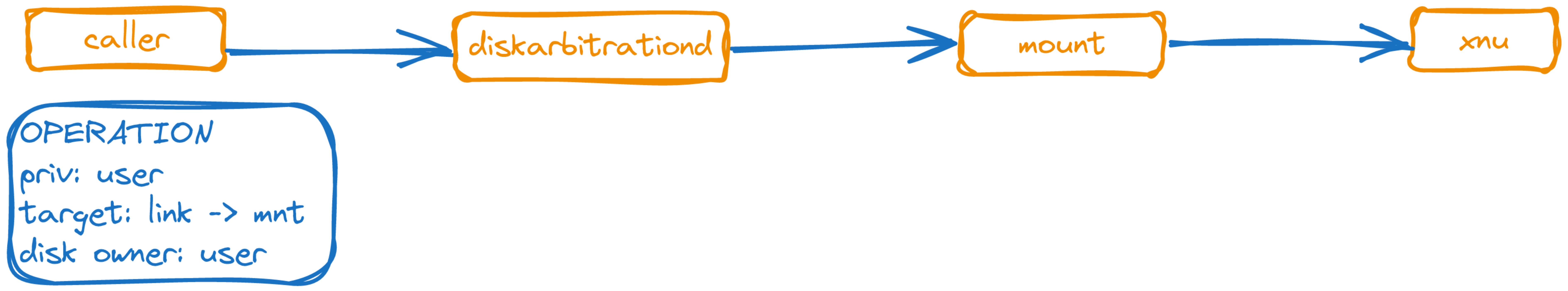
```
csaby@macos12 ~ % rm -rf /tmp/disk2
csaby@macos12 ~ % ln -s /tmp/disk /tmp/disk2
```

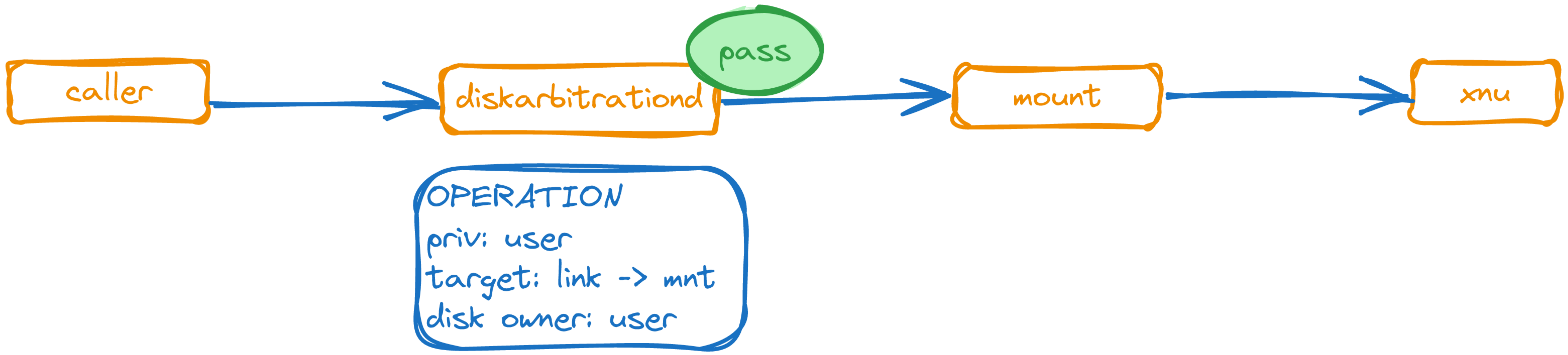


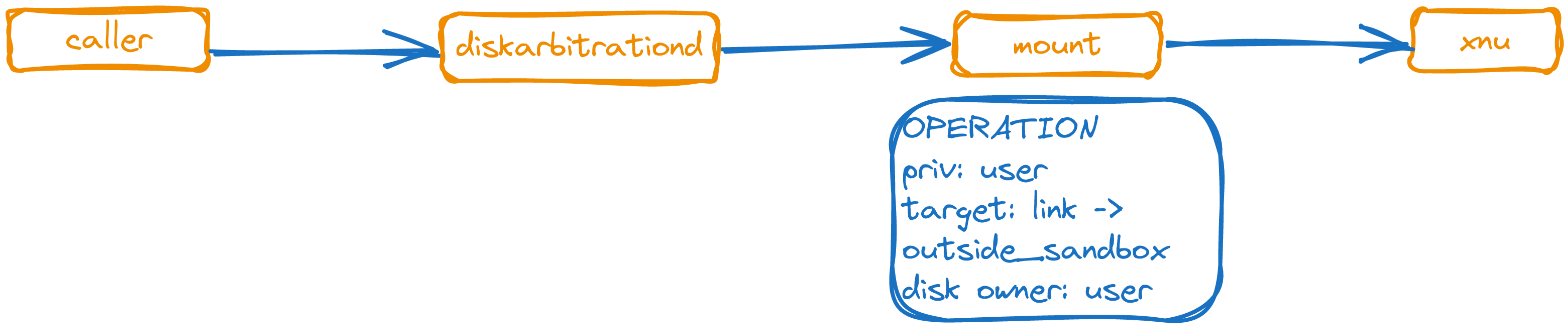
```
(lldb) c
Process 121 resuming
(lldb) detach
Process 121 detached
(lldb) exit
csaby@macos12 ~ %
```

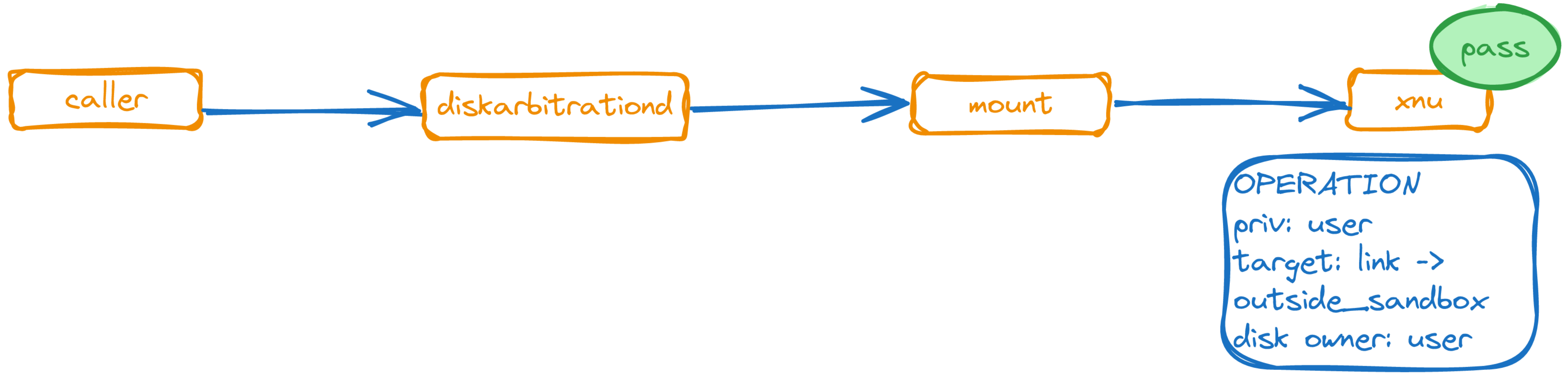


```
/dev/disk4s1         41504653-0000-11AA-AA11-0030654/private/tmp/disk
csaby@macos12 ~ %
```


caller
diskarbitrationd
mount
xnu
OPERATION
priv: user
target: link -> mnt
disk owner: user

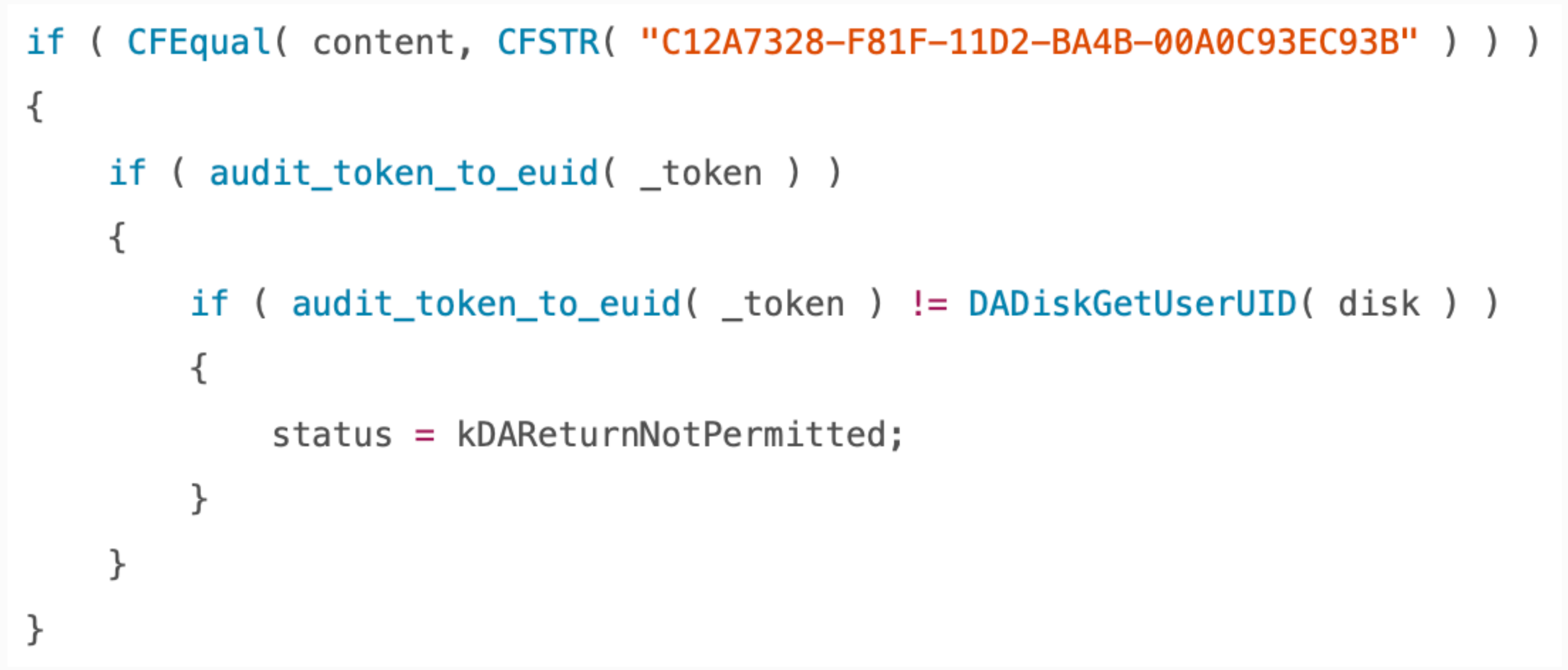

caller
diskarbitrationd
pass
mount
xnu
OPERATION
priv: user
target: link -> mnt
disk owner: user


caller
diskarbitrationd
mount
xnu
OPERATION
priv: user
target: link ->
outside_sandbox
disk owner: user

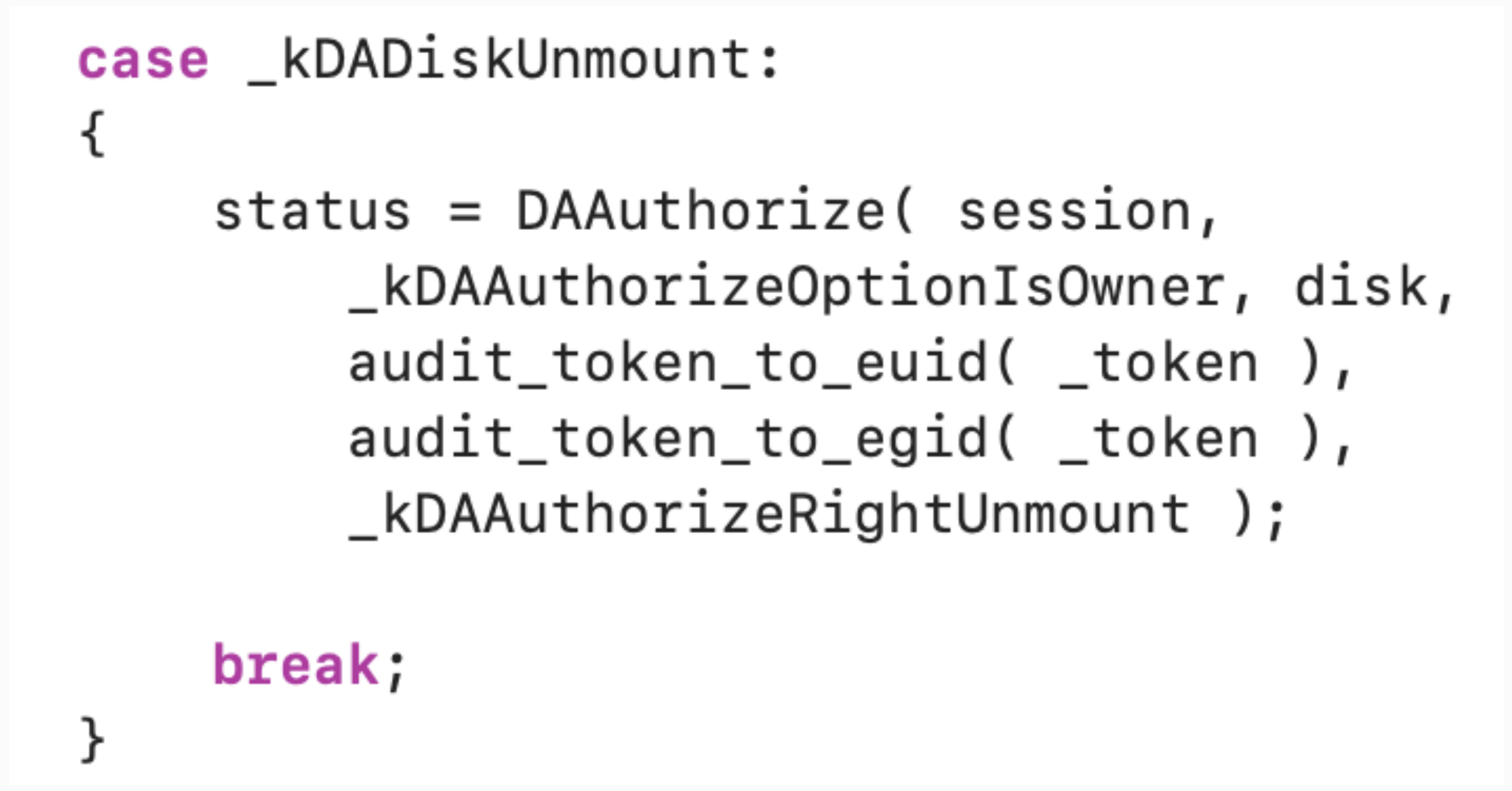

caller
diskarbitrationd
mount
xnu
pass
OPERATION
priv: user
target: link -> outside_sandbox
disk owner: user

CVE-2022-32780 - exploitation

- what to mount?
  - EFI won't work
  - custom dmg!
- how? DA works on /dev, diskmanagementd (can map dmg into /dev/) is not reachable from sandbox
  - 💡use "open"
- we can unmount, /dev/ remains



```
if ( CFEqual( content, CFSTR( "C12A7328-F81F-11D2-BA4B-00A0C93EC93B" ) ) )
{
    if ( audit_token_to_euid( _token ) )
    {
        if ( audit_token_to_euid( _token ) != DADiskGetUserUID( disk ) )
        {
            status = kDAReturnNotPermitted;
        }
    }
}
```



```
case _kDADiskUnmount:
{
    status = DAAuthorize( session,
        _kDAAuthorizeOptionIsOwner, disk,
        audit_token_to_euid( _token ),
        audit_token_to_egid( _token ),
        _kDAAuthorizeRightUnmount );

    break;
}
```

CVE-2022-32780 - exploitation

- where to mount?
  - Terminal Preferences
  - ~/Library/Preferences/com.apple.Terminal.plist
  - "CommandString" executed upon launch

```
<key>Window Settings</key>
<dict>
    <key>Basic</key>
    <dict>
        <key>CommandString</key>
        <string>touch /Users/Shared/sandboxescape.txt</string>
```

CVE-2022-32780 - full exploit

1. Drops a `dmg` file

2. It will call `open` to open a `dmg` file

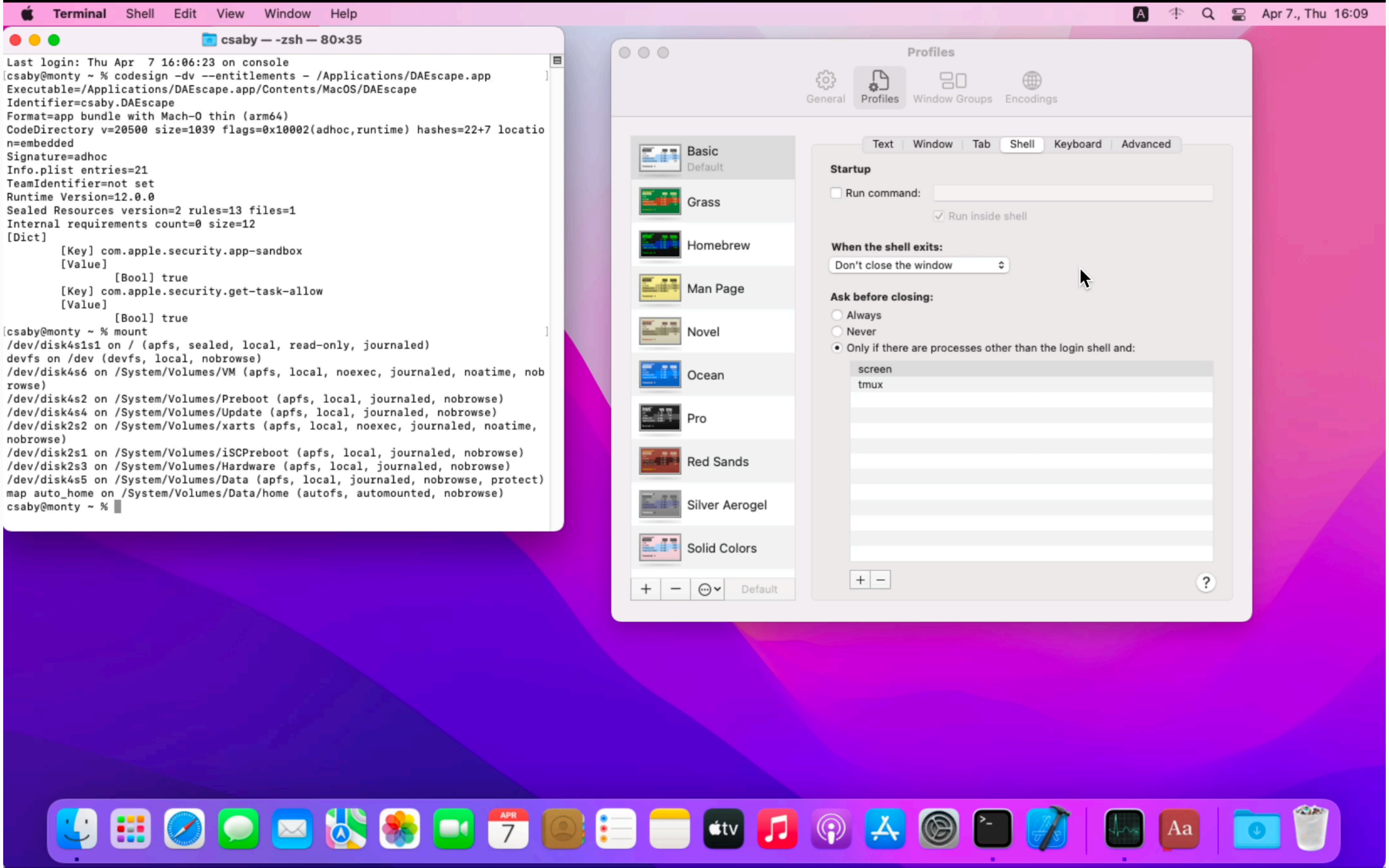
3. Then it will use the diskarbitration service to unmount it --> at this point we have a custom disk device we can mount somewhere

4. It will start a thread to alternate the symlink and the directory

5. Then it will start a loop to call the mount operation of the DA service - due to the racer it will eventually succeed
    - we also always unmount the local directory, as we don't need that

6. It will check if we mounted over `Preferences`, and if yes stop

7. Open Terminal



```
Last login: Thu Apr  7 16:06:23 on console
csaby@monty ~ % codesign -dv --entitlements - /Applications/DAEscape.app
Executable=/Applications/DAEscape.app/Contents/MacOS/DAEscape
Identifier=csaby.DAEscape
Format=app bundle with Mach-O thin (arm64)
CodeDirectory v=20500 size=1039 flags=0x10002(adhoc,runtime) hashes=22+7 locatio
n=embedded
Signature=adhoc
Info.plist entries=21
TeamIdentifier=not set
Runtime Version=12.0.0
Sealed Resources version=2 rules=13 files=1
Internal requirements count=0 size=12
[Dict]
        [Key] com.apple.security.app-sandbox
        [Value]
                [Bool] true
        [Key] com.apple.security.get-task-allow
        [Value]
                [Bool] true
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, nob
rowse)
/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/xarts (apfs, local, noexec, journaled, noatime,
nobrowse)
/dev/disk2s1 on /System/Volumes/iSCPreboot (apfs, local, journaled, nobrowse)
/dev/disk2s3 on /System/Volumes/Hardware (apfs, local, journaled, nobrowse)
/dev/disk4s5 on /System/Volumes/Data (apfs, local, journaled, nobrowse, protect)
map auto_home on /System/Volumes/Data/home (autofs, automounted, nobrowse)
csaby@monty ~ %
```

Startup

Run command: [ ]

[✓] Run inside shell

When the shell exits: Don't close the window

Ask before closing:

( ) Always

( ) Never

(●) Only if there are processes other than the login shell and:

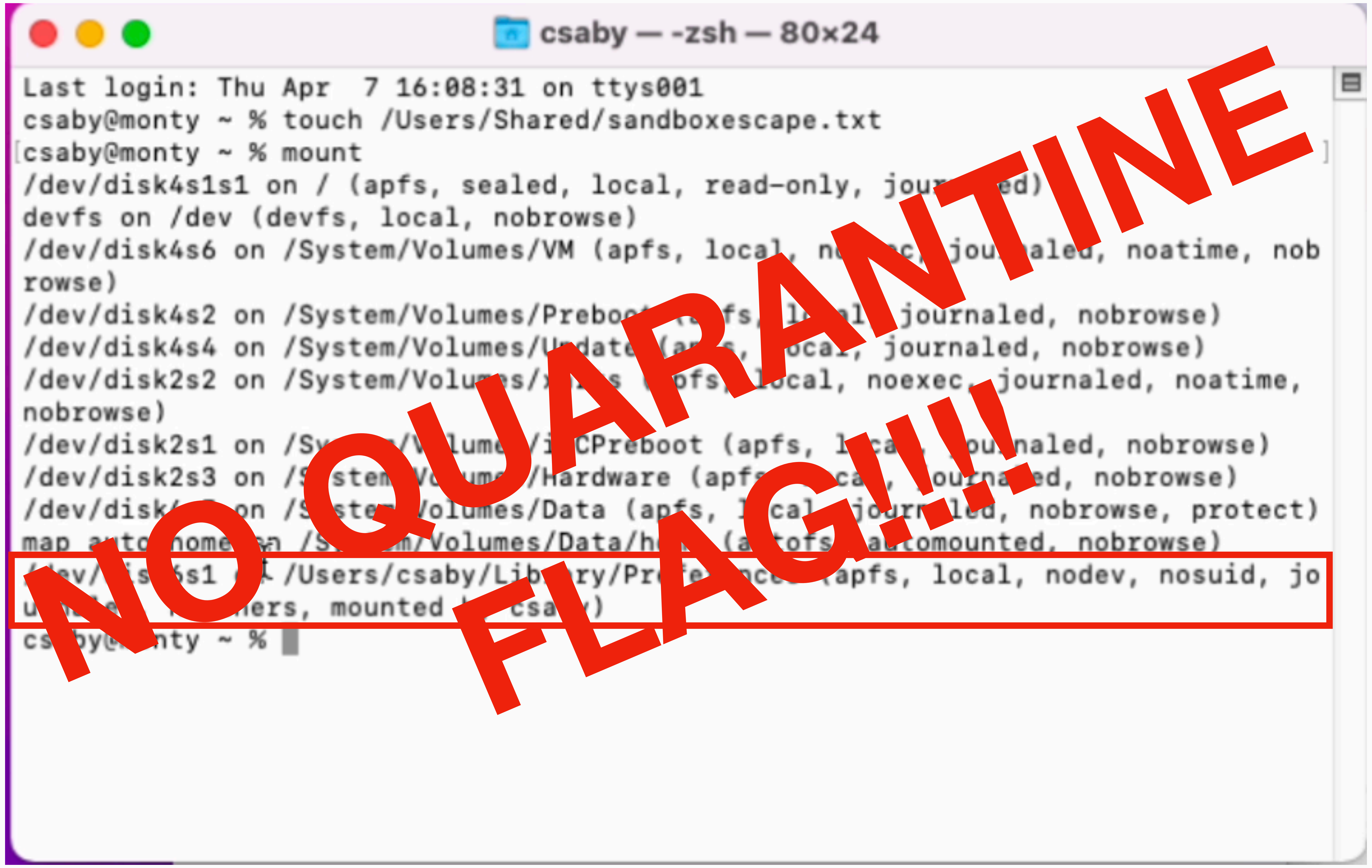
screen
tmux

CVE-2022-32780 - fix

- every mount call has the "-k" option = do not follow symbolic links
- the kernel will discard any request if there is a symlink in the path

CVE-2023-42838 - Sandbox Escape

Where is the problem?

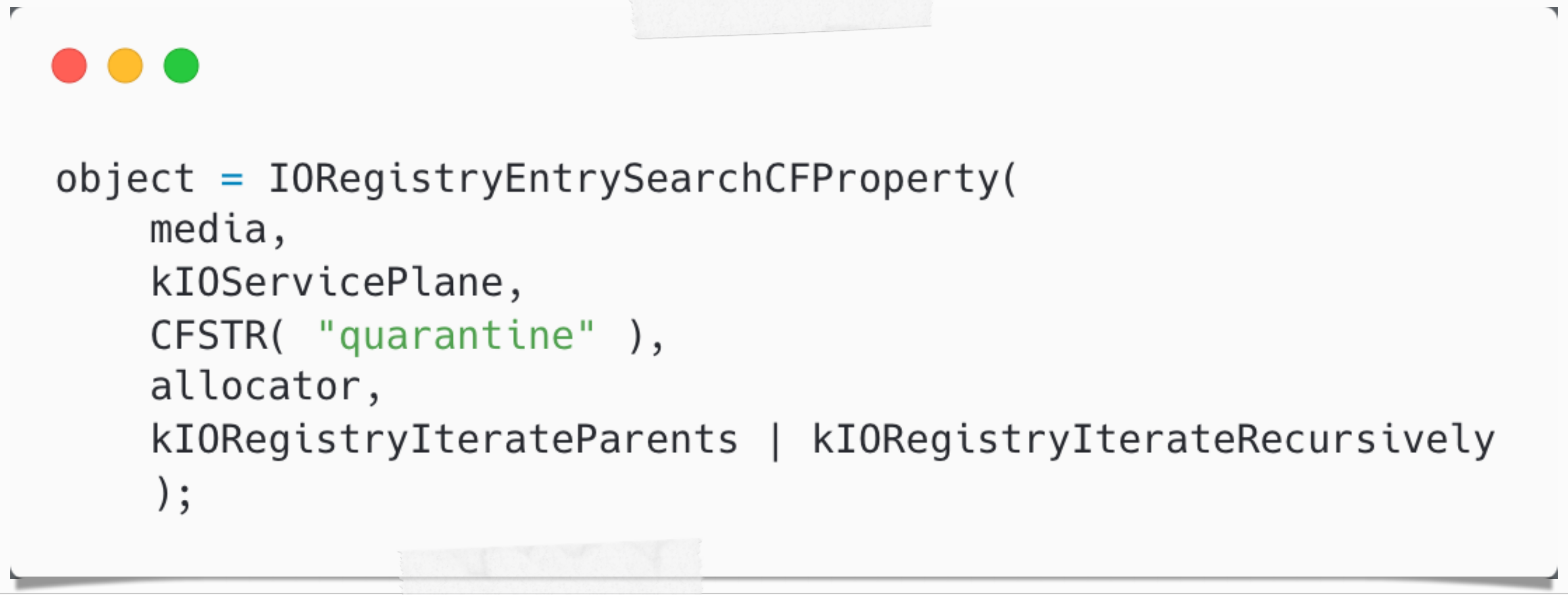


```
Last login: Thu Apr  7 16:08:31 on ttys001
csaby@monty ~ % touch /Users/Shared/sandboxescape.txt
csaby@monty ~ % mount
/dev/disk4s1s1 on / (apfs, sealed, local, read-only, journaled)
devfs on /dev (devfs, local, nobrowse)
/dev/disk4s6 on /System/Volumes/VM (apfs, local, noexec, journaled, noatime, nob
rowse)
/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/xarts (apfs, local, noexec, journaled, noatime,
nobrowse)
/dev/disk2s1 on /System/Volumes/iSCPreboot (apfs, local, journaled, nobrowse)
/dev/disk2s3 on /System/Volumes/Hardware (apfs, local, journaled, nobrowse)
/dev/disk4s5 on /System/Volumes/Data (apfs, local, journaled, nobrowse, protect)
map auto_home on /System/Volumes/Data/home (autofs, automounted, nobrowse)
/dev/disk6s1 on /Users/csaby/Library/Preferences (apfs, local, nodev, nosuid, jo
urnaled, noowners, mounted by csaby)
csaby@monty ~ %
```

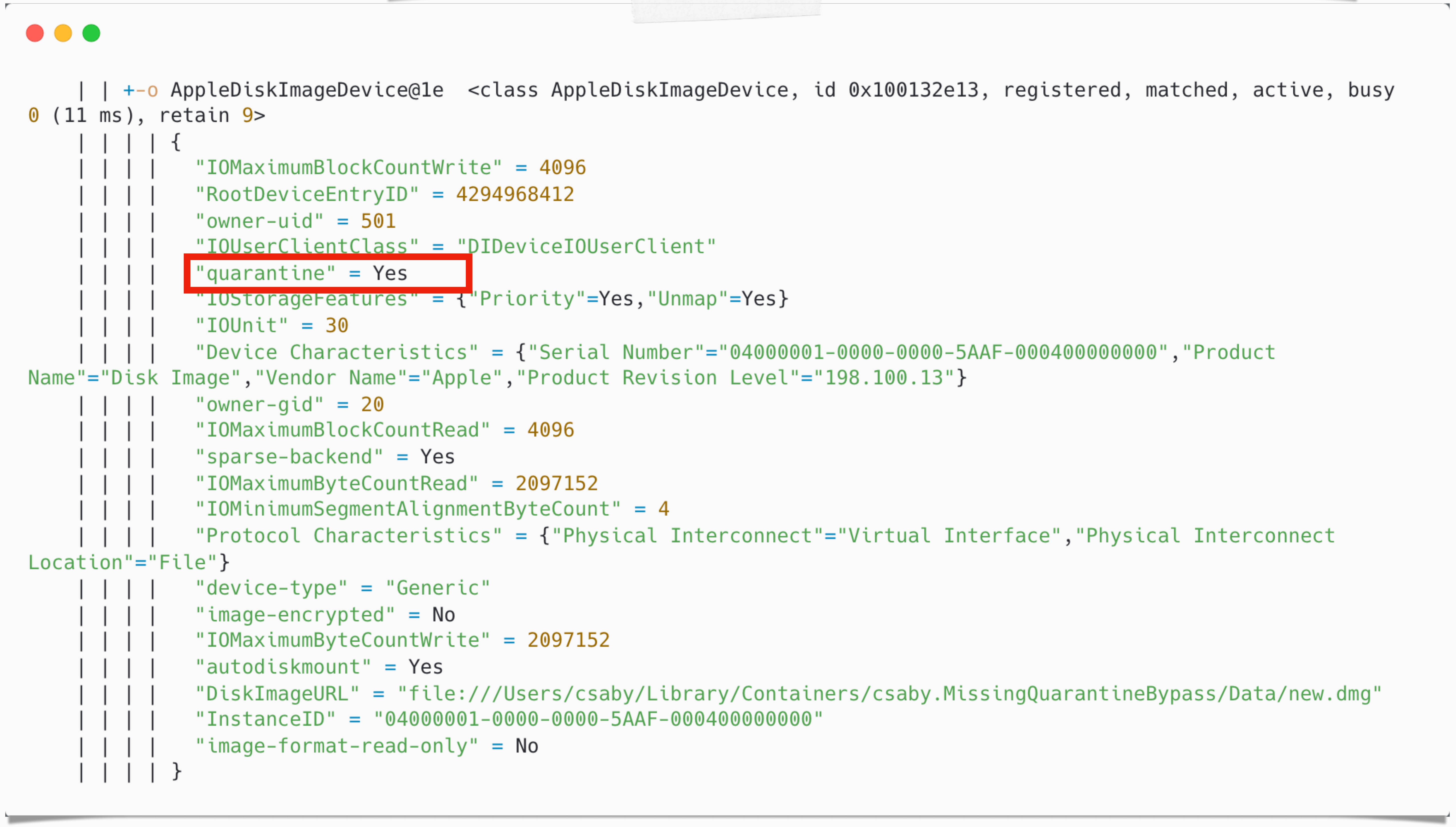
NO QUARANTINE FLAG!!!

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

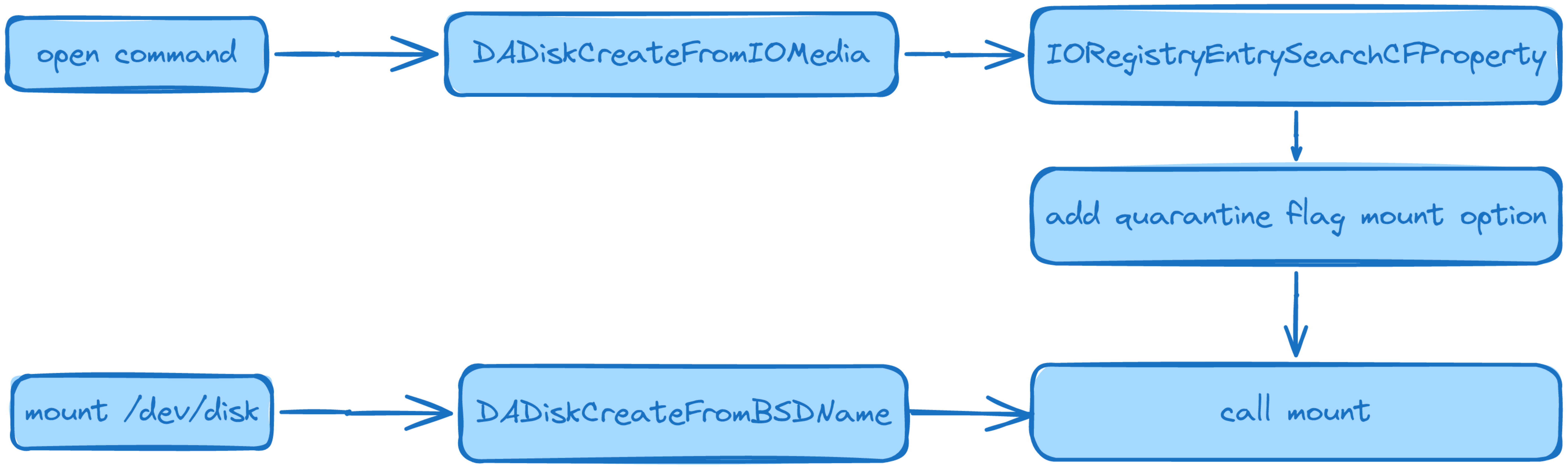


```
object = IORegistryEntrySearchCFProperty(
    media,
    kIOServicePlane,
    CFSTR( "quarantine" ),
    allocator,
    kIORegistryIterateParents | kIORegistryIterateRecursively
    );
```



```
    | | +-o AppleDiskImageDevice@1e  <class AppleDiskImageDevice, id 0x100132e13, registered, matched, active, busy
0 (11 ms), retain 9>
    | | | | {
    | | | |   "IOMaximumBlockCountWrite" = 4096
    | | | |   "RootDeviceEntryID" = 4294968412
    | | | |   "owner-uid" = 501
    | | | |   "IOUserClientClass" = "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
    | | | | }
```

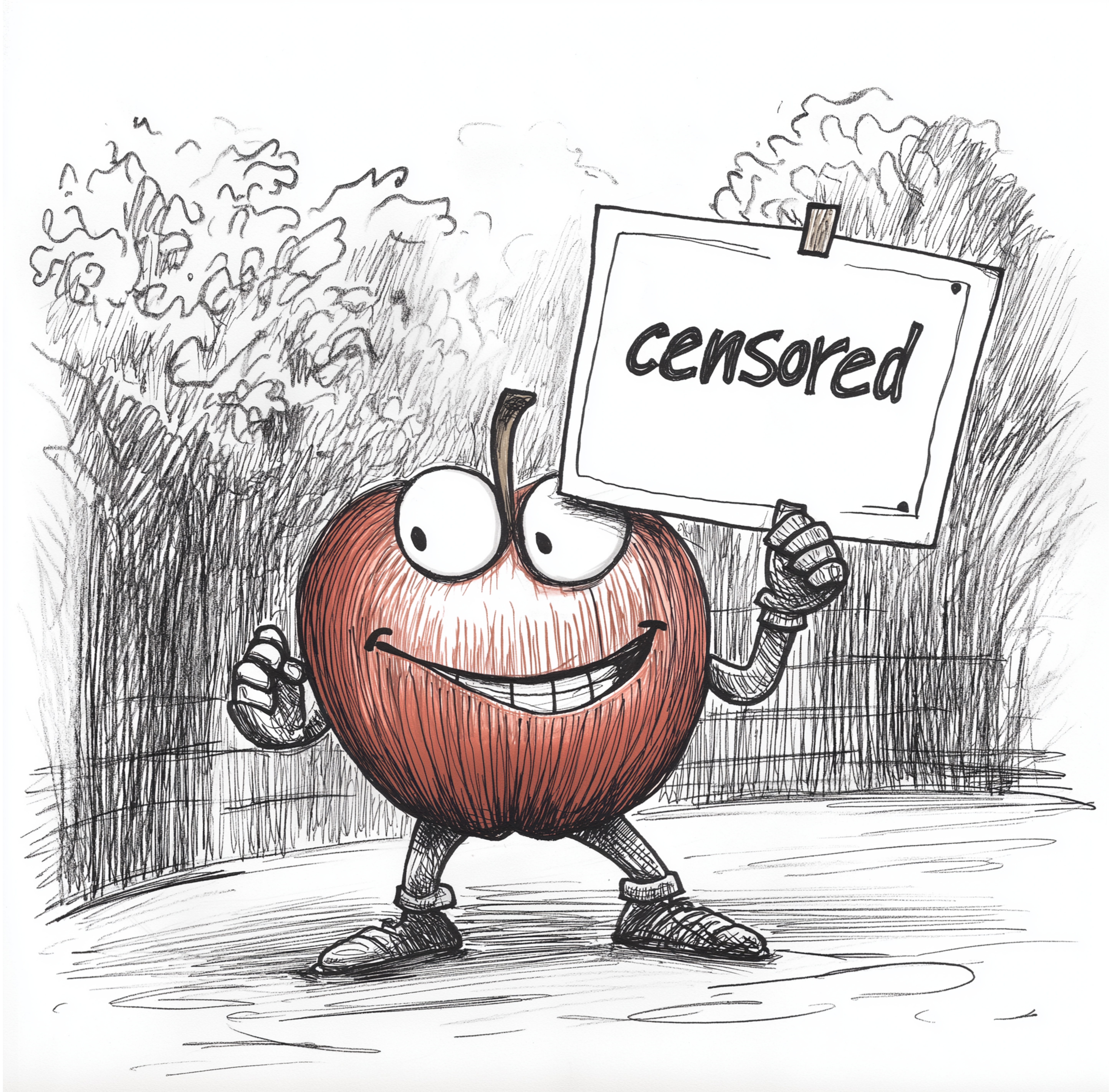
CVE-2023-42838 - what goes on?

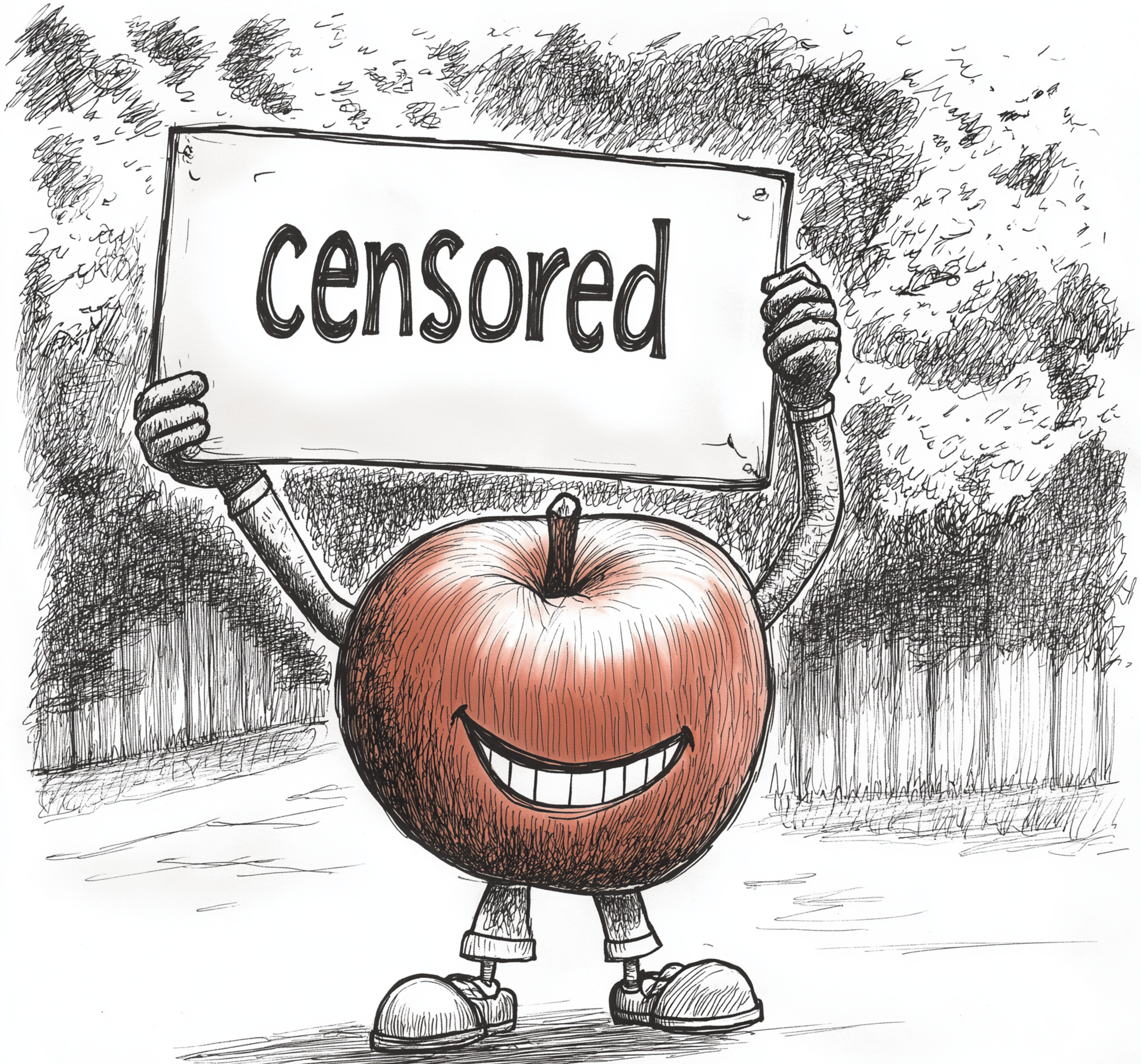


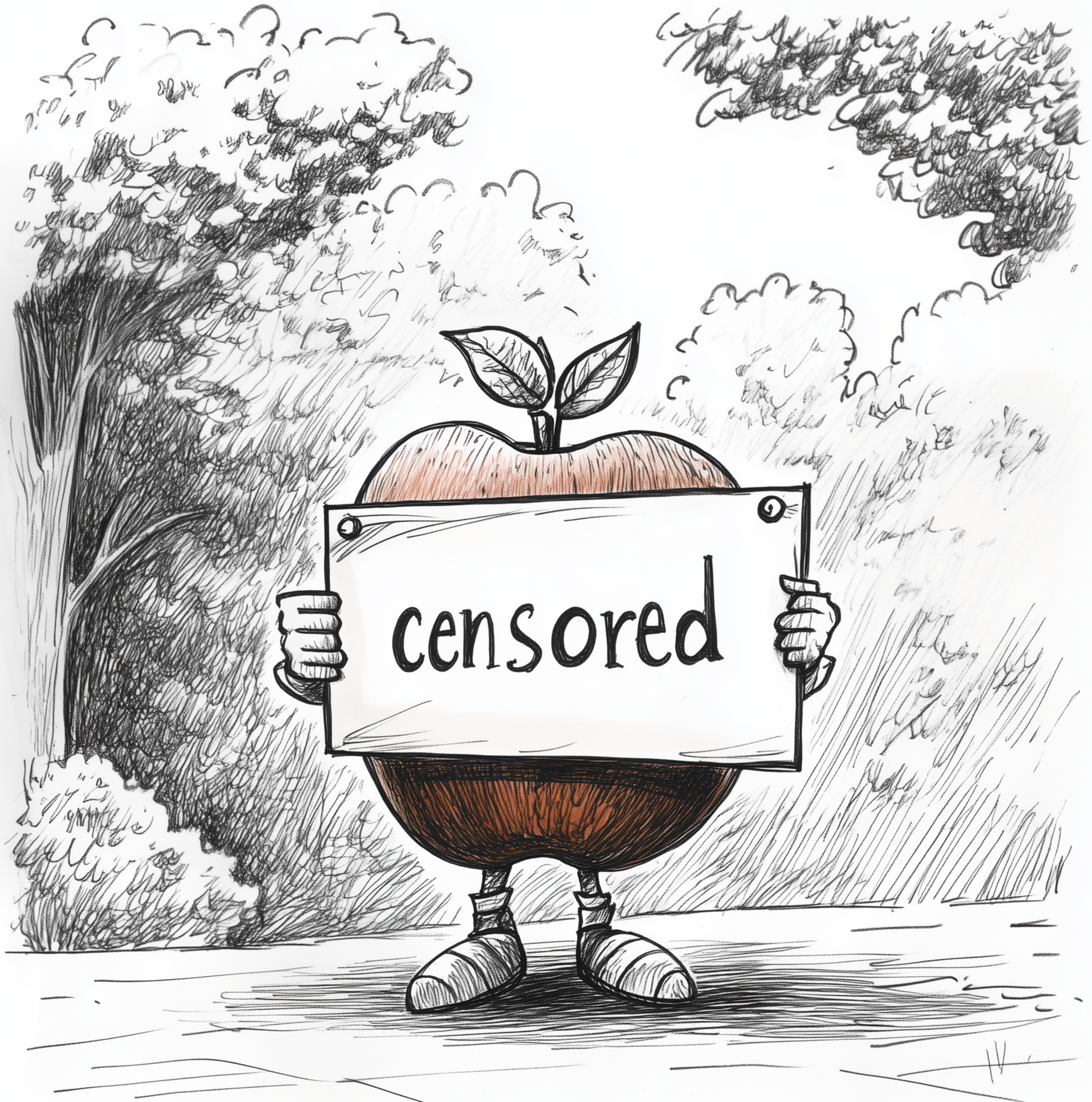
CVE-2023-42838 - fix

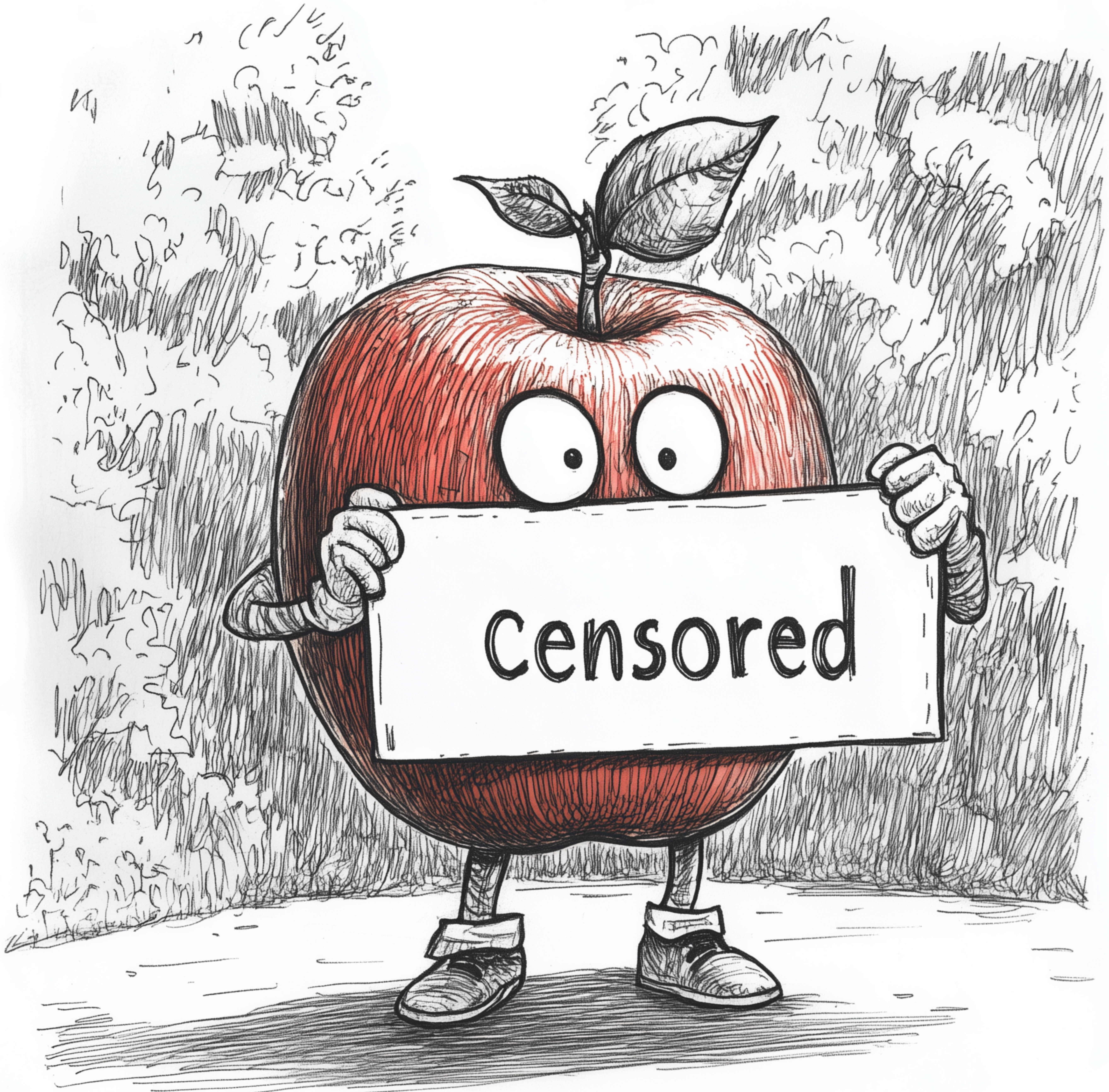
- the kernel will add quarantine flag to every mount if the device is quarantined
- basically the "IOReg" query went down to kernel and performed on every mount

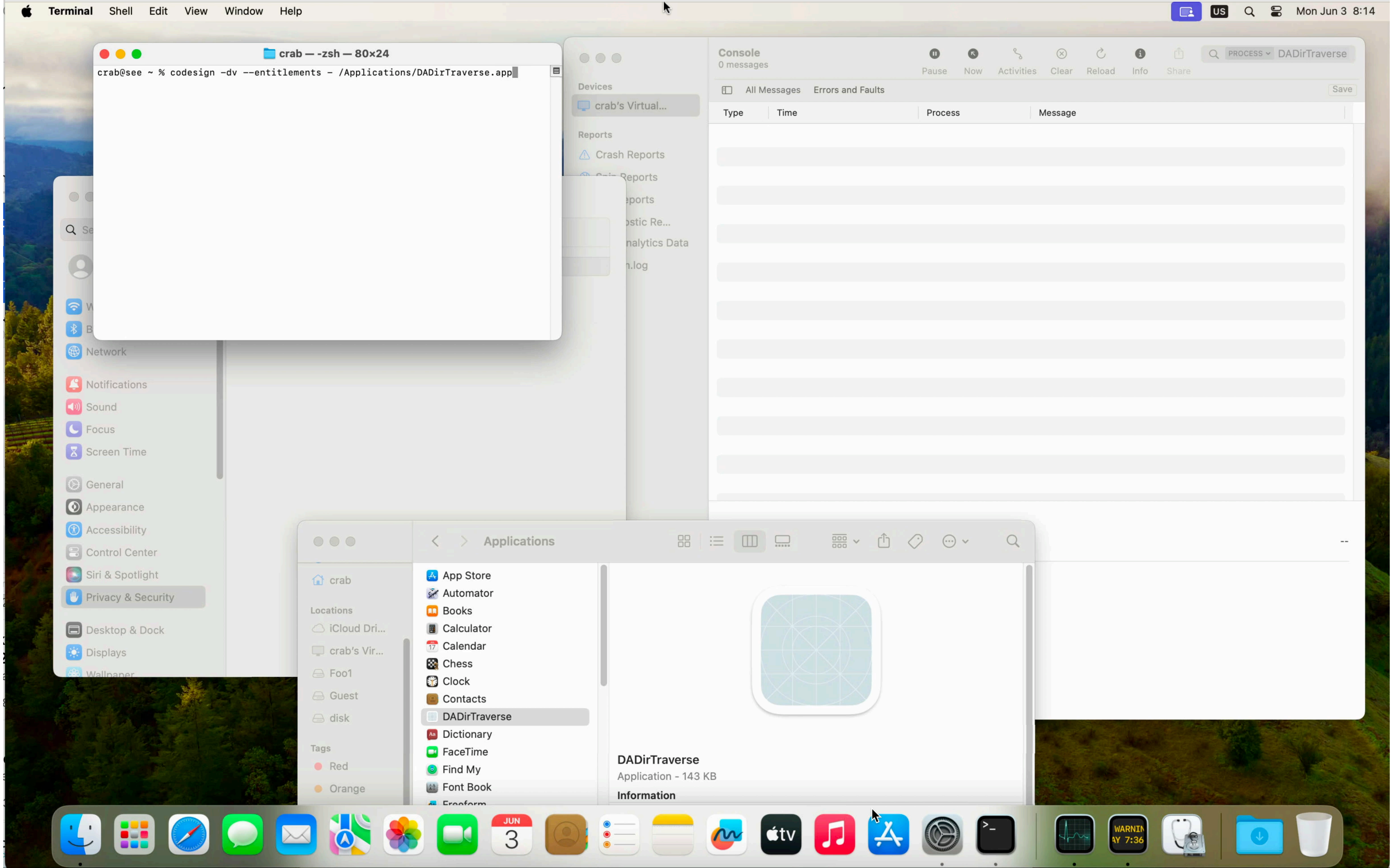
~~CVE-2024-40855- Sandbox Escape & TCC Bypass~~


censored


Censored


censored

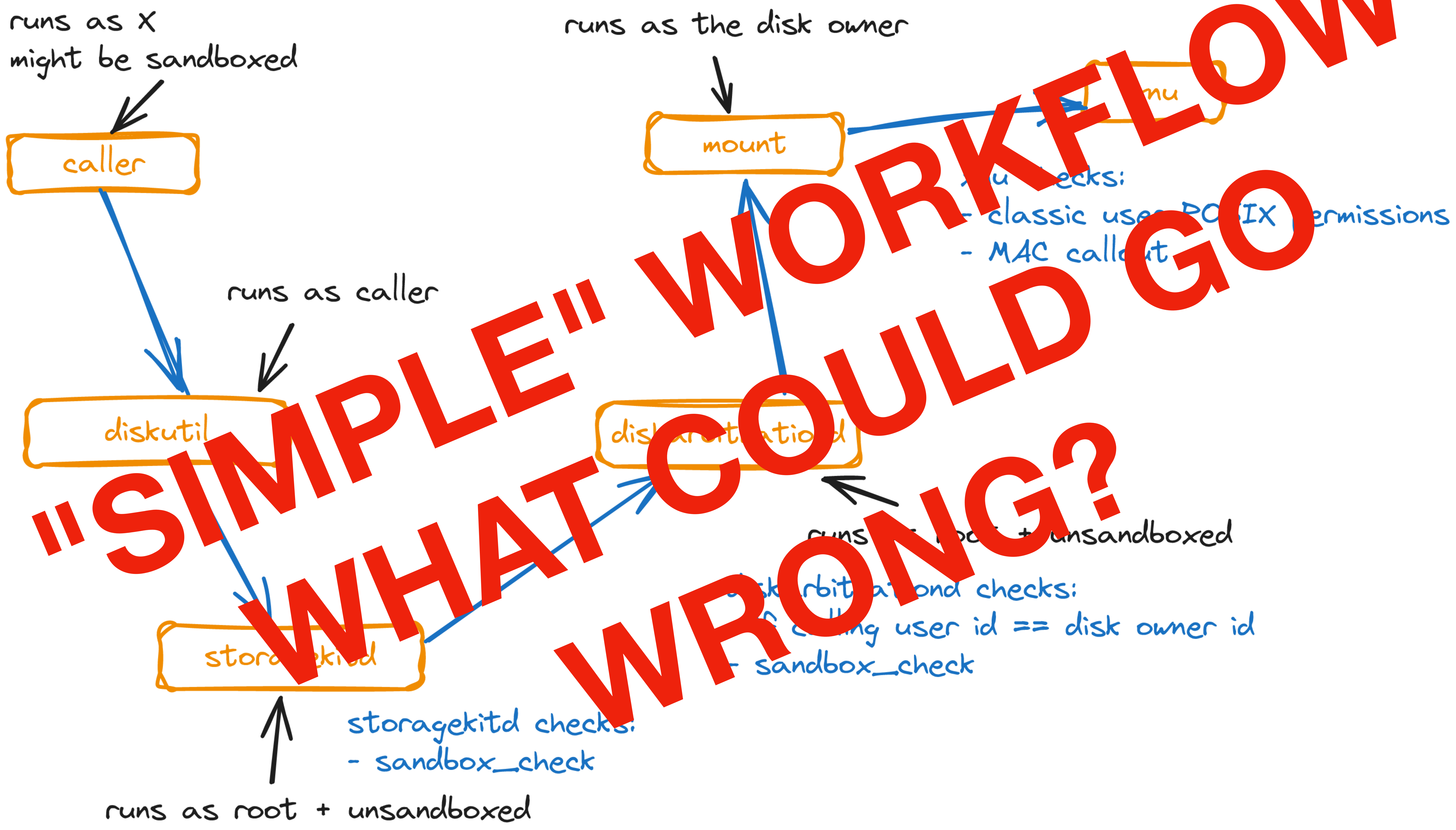

Censored

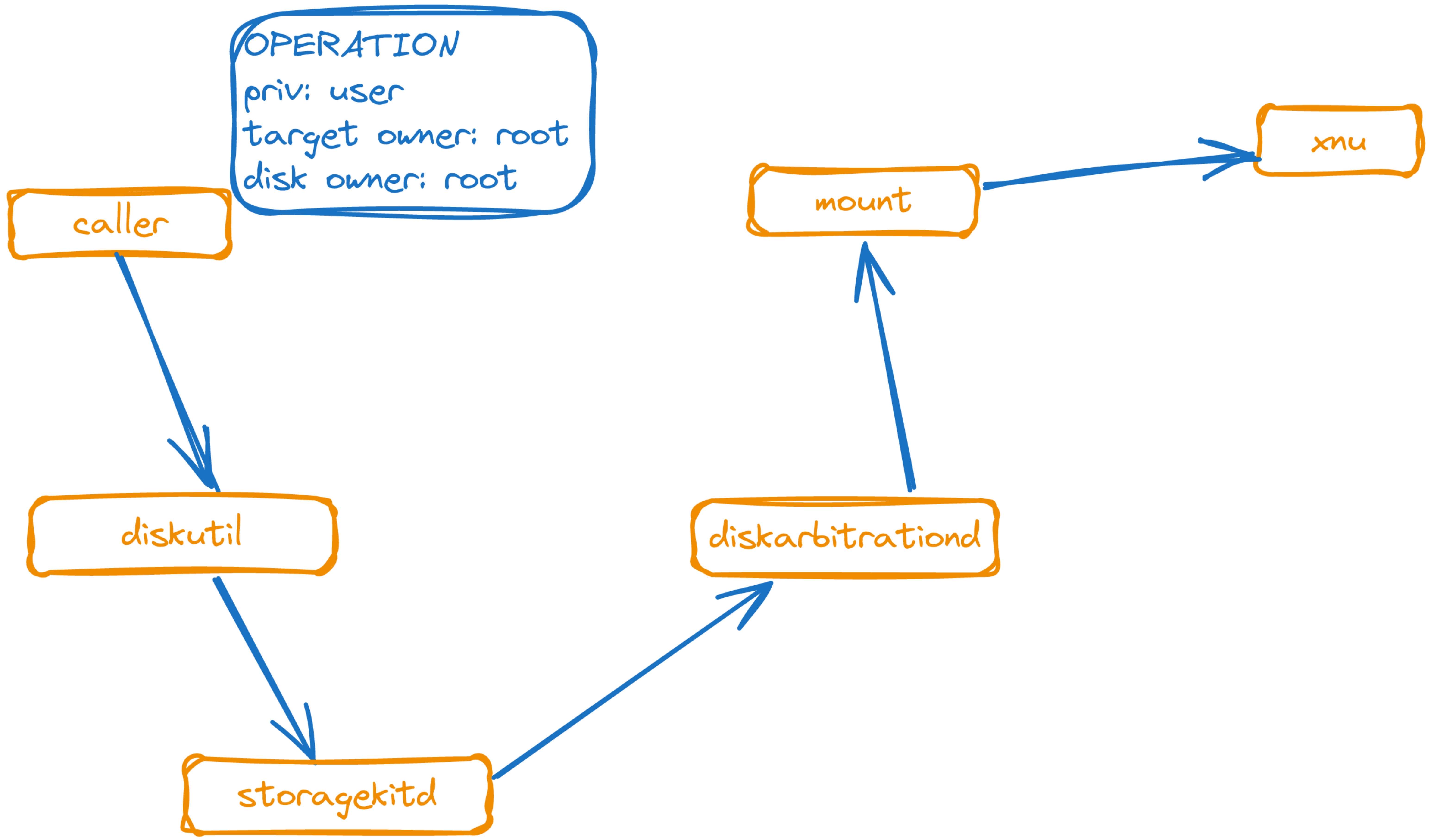


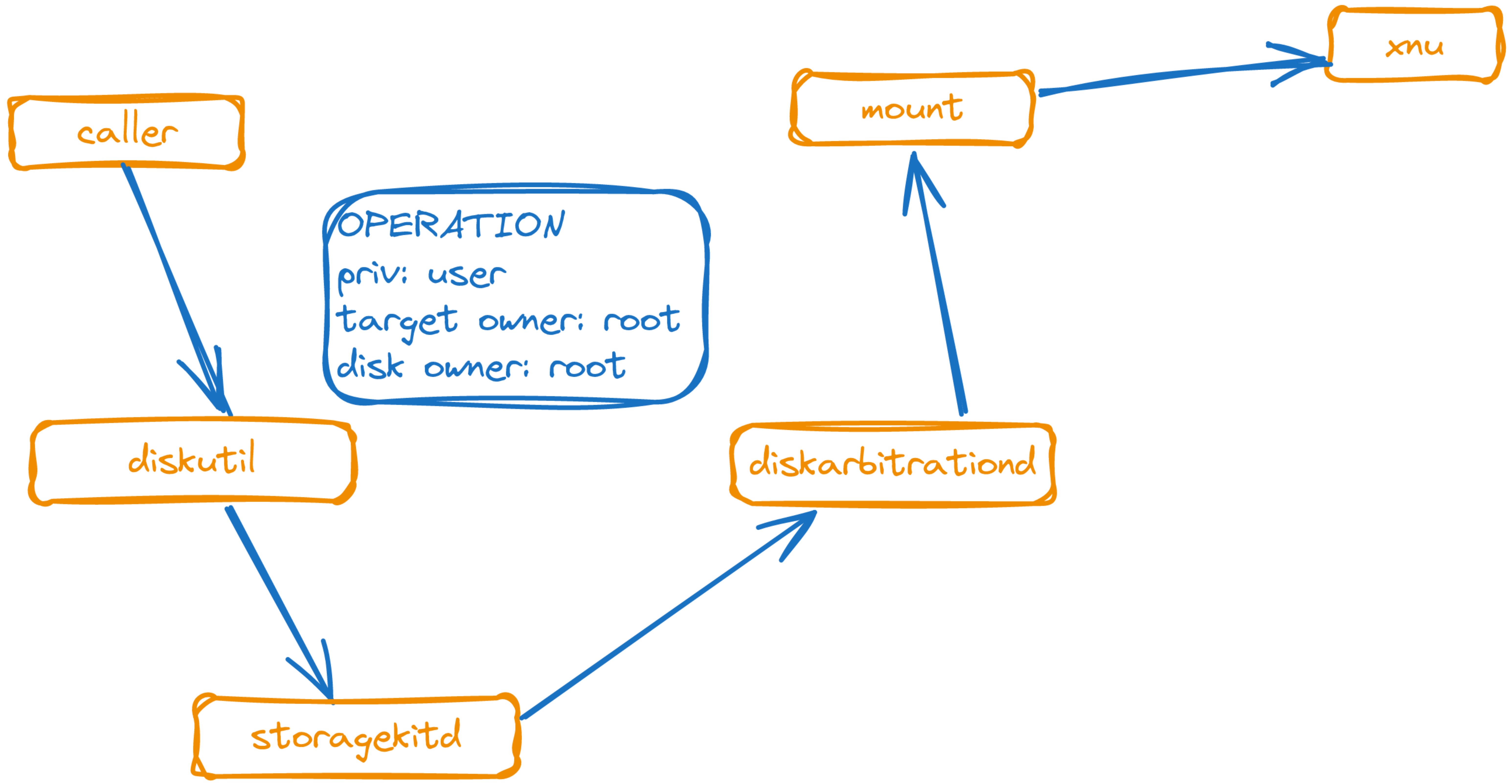
```
crab@see ~ % codesign -dv --entitlements - /Applications/DADirTraverse.app
```

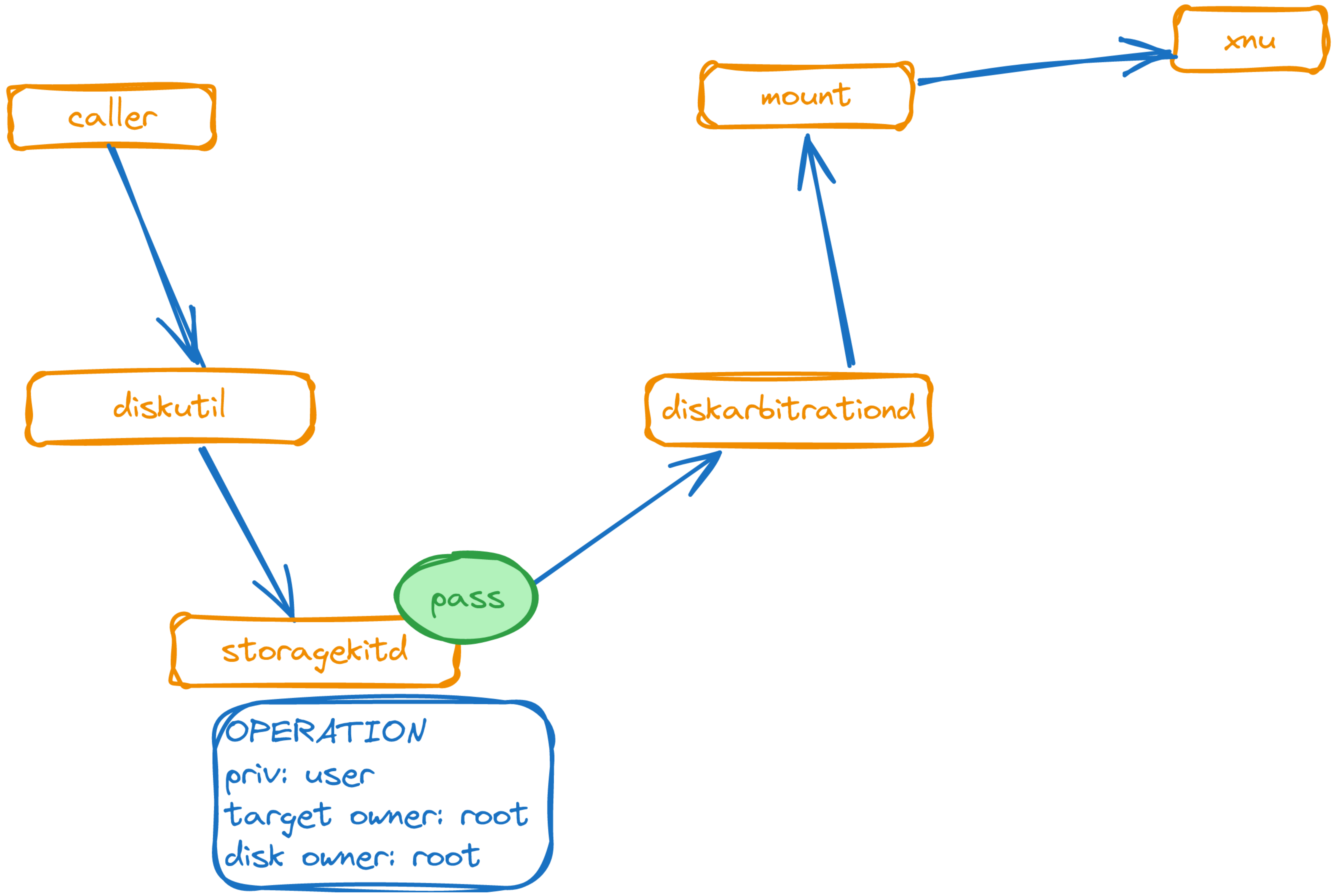
CVE-2024-27848 - LPE via StorageKit

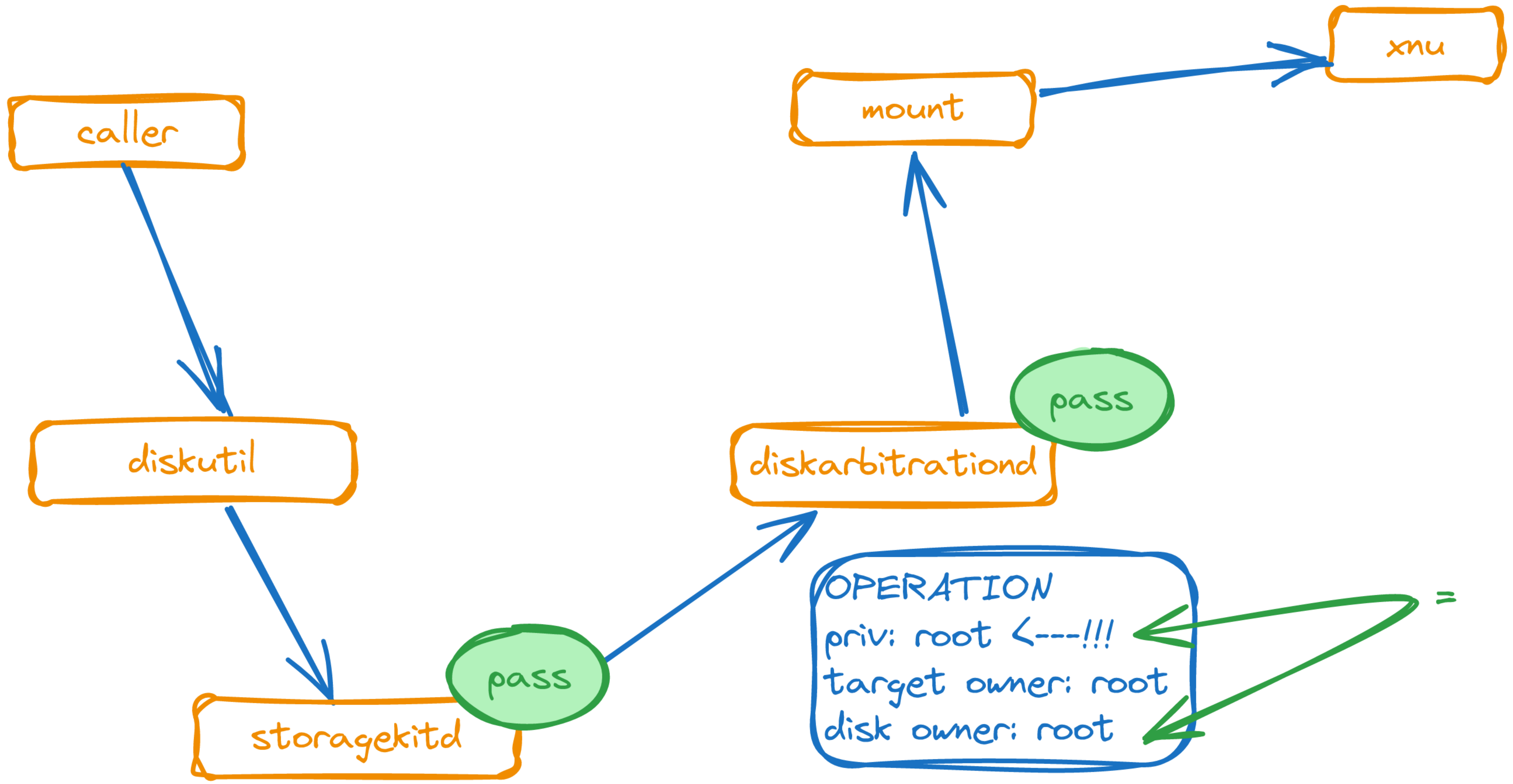
"SIMPLE" WORKFLOW WHAT COULD GO WRONG?

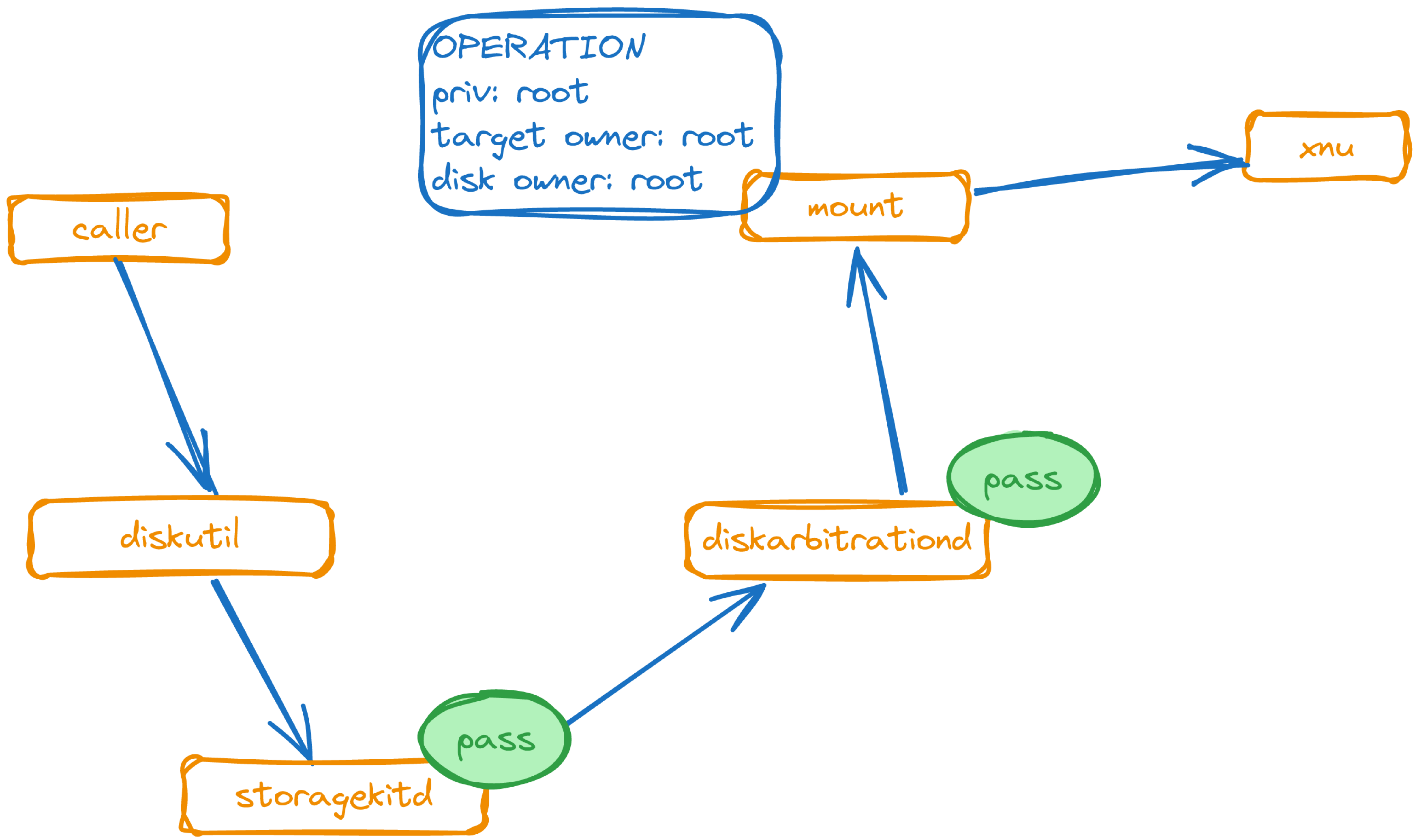


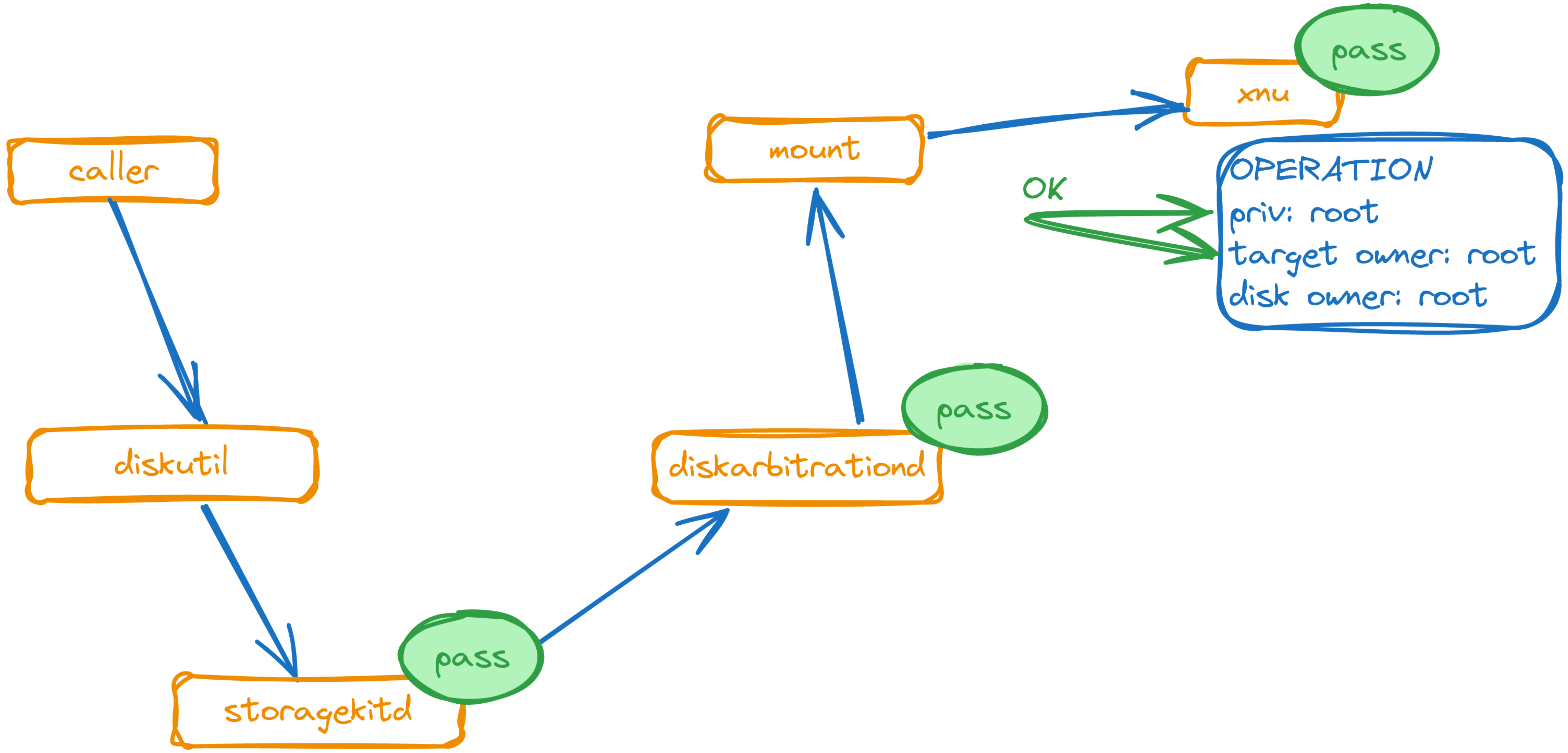

OPERATION
priv: user
target owner: root
disk owner: root
caller
diskutil
storagekitd
diskarbitrationd
mount
xnu


caller
diskutil
storagekitd
diskarbitrationd
mount
xnu
OPERATION
priv: user
target owner: root
disk owner: root


caller
diskutil
storagekitd
pass
diskarbitrationd
mount
xnu
OPERATION
priv: user
target owner: root
disk owner: root


caller
diskutil
storagekitd
pass
diskarbitrationd
pass
mount
xnu
OPERATION
priv: root <---!!!
target owner: root
disk owner: root
=


OPERATION
priv: root
target owner: root
disk owner: root
caller
diskutil
storagekitd
pass
diskarbitrationd
pass
mount
xnu


caller
diskutil
storagekitd
pass
diskarbitrationd
pass
mount
xnu
pass
OK
OPERATION
priv: root
target owner: root
disk owner: root

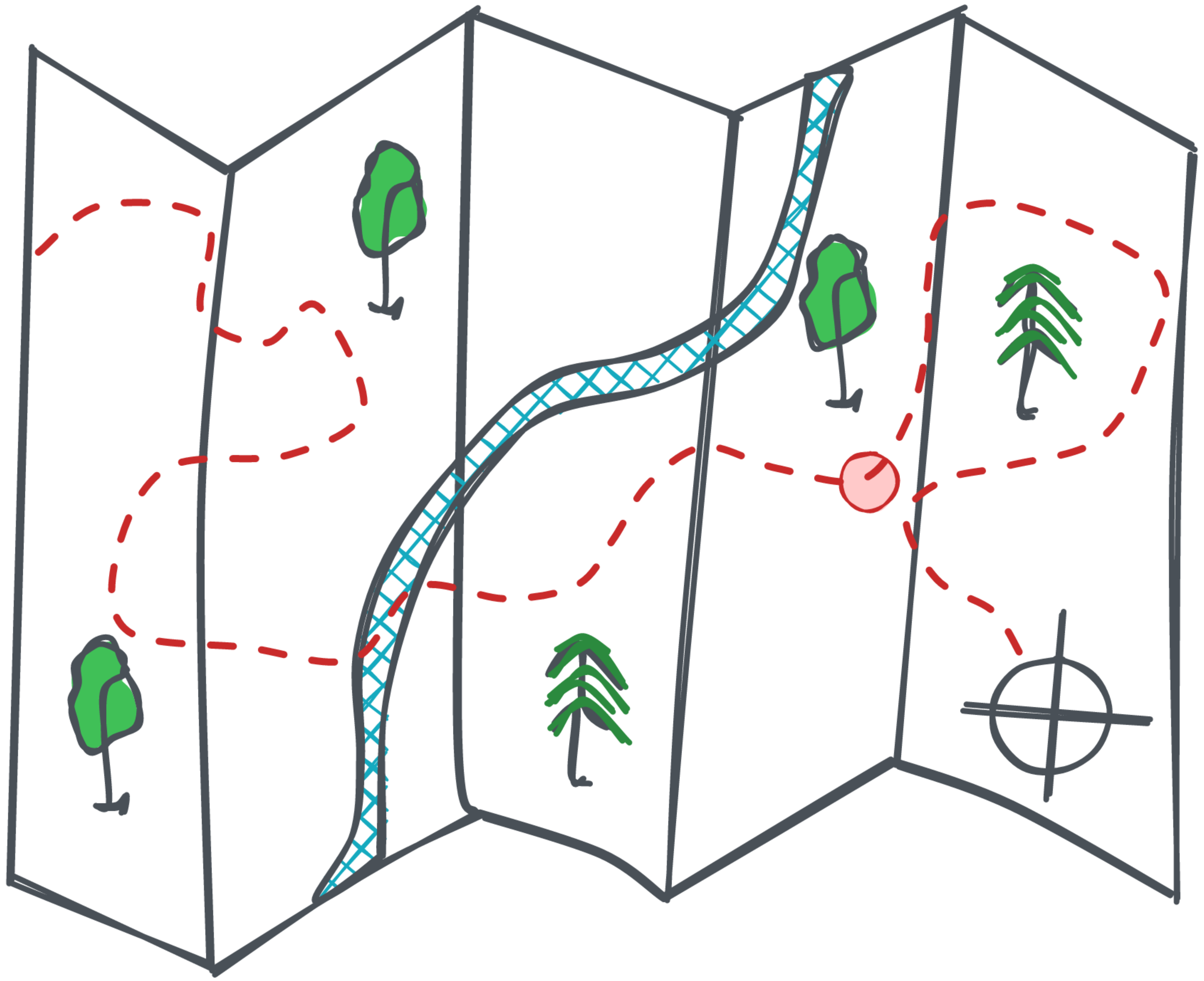
CVE-2024-27828 - exploitation

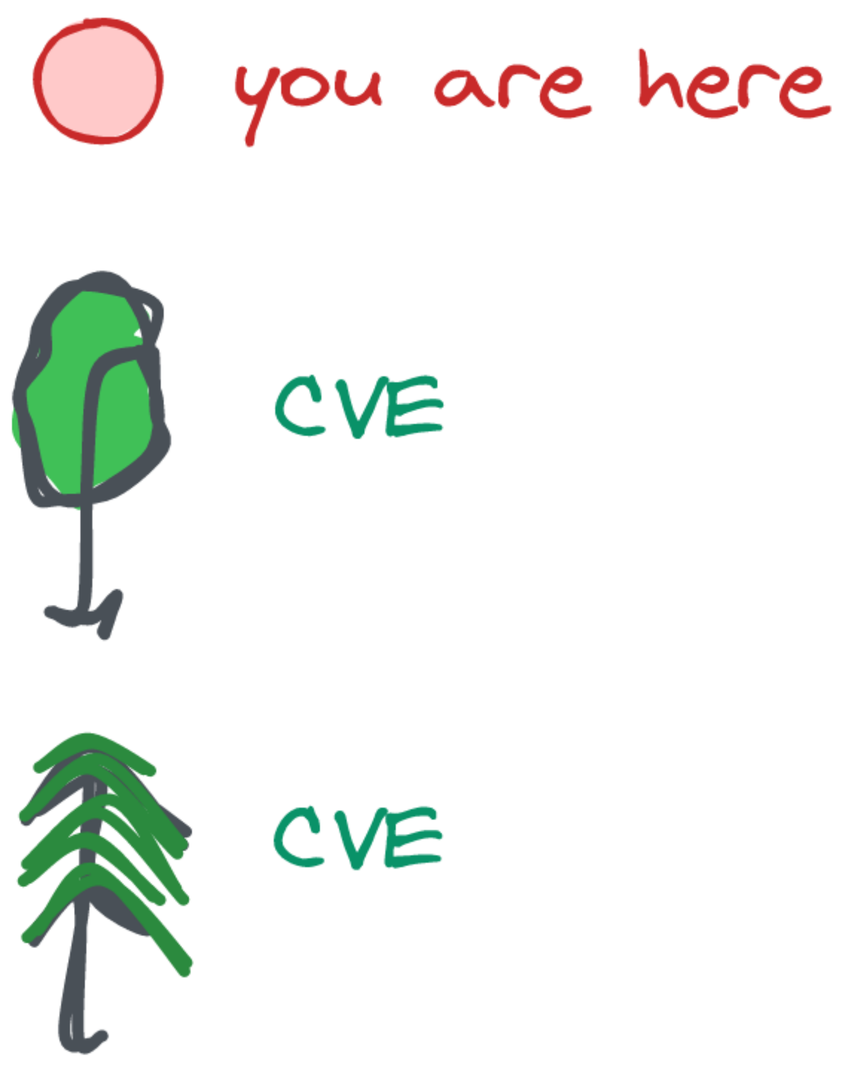
- create new volume which we can write to
- mount over /etc/cups
- cups-files.conf:
  - LogFilePerm - file permissions
  - ErrorLog - /etc/sudoers.d/somefile
- cupsctl to trigger
- Step 1: perm: 777
- Step 2: Overwrite /etc/sudoers.d/somefile
- Step 3: perm 700 (sudo likes this)
- Step 4: sudo su

Terminal
Shell
Edit
View
Window
Help
Wed Mar 27 14:39
secret.txt
fish — -zsh — 112×35

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

to be continued....





conclusion

DISKARBITRATIOND
GOOD MOR-NING SUNSHINE!...
HACKERS
imgflip.com

Security is f***g hard.**

– Csaba Fitzl

kandji

Csaba Fitzl
X: @theevilbit

Icons

- flaticon.com
  - kliwir art
  - Freepik