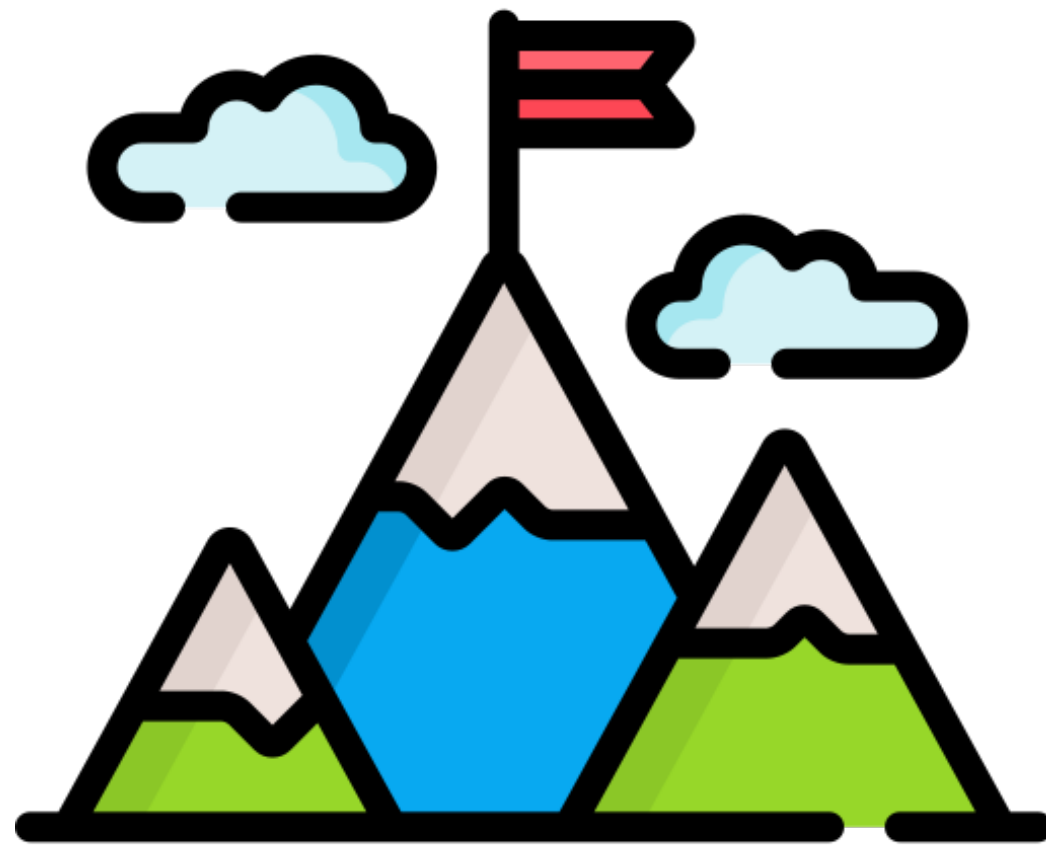


# Mount(ain) of Bugs



*Csaba Fitzl*

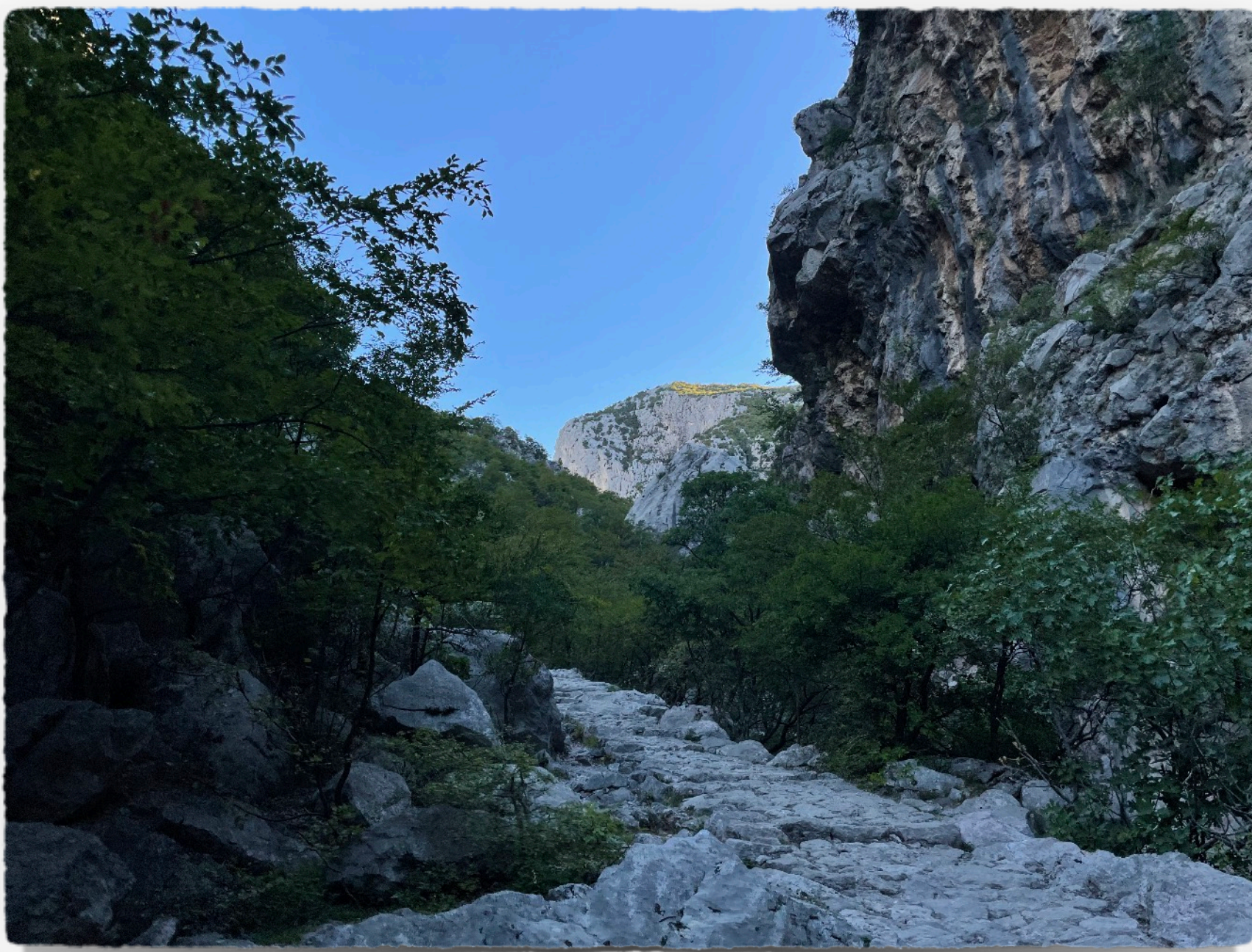
*Twitter: @theevilbit*

# whoami

- author of "macOS Control Bypasses" at Offensive Security
- ex red/blue teamer
- macOS bug hunter
- husband, father
- hiking, trail running 🥾 🏔️



pwd



# agenda

1. MACF
2. the mount system call
3. disk arbitration service
4. CVE-2017-7170 (Patric Wardle) - Sniffing Authorization References
5. CVE-2017-2533 (pwn2own) - Mount yourself a root shell
6. CVE-2020-9771 - TCC bypass via snapshot mounting
7. CVE-2021-1784 - TCC bypass via mounting over com.apple.TCC
8. CVE-2021-30782 - TCC bypass via AppTranslocation service
9. CVE-2021-26089 - Fortinet Forticlient installer LPE
10. other tricks and new bugs

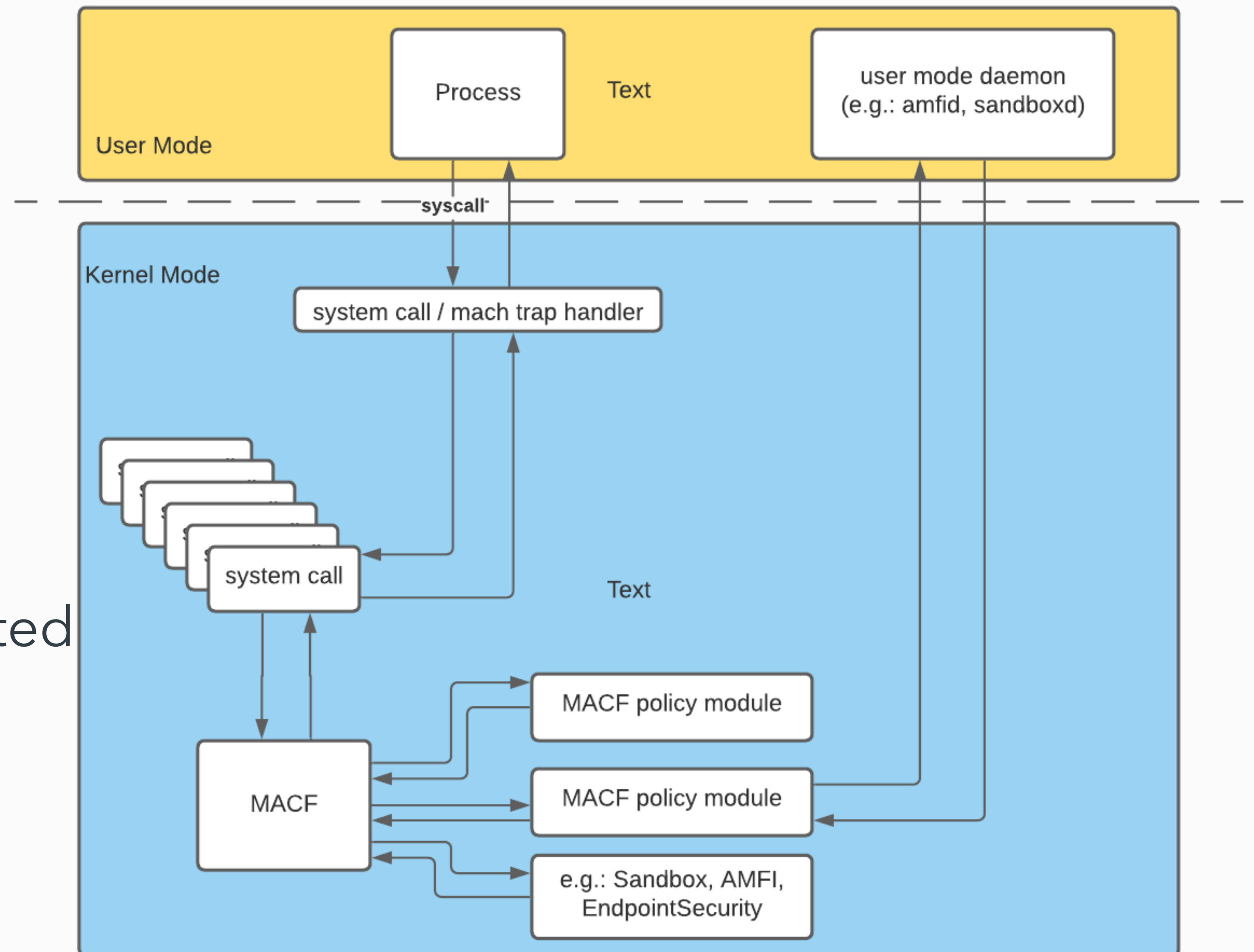
**MACF**

-

**Mandatory Access Control Framework**

# MACF

- origin: TrustedBSD MAC
- implemented in kernel
- policy modules extend the kernel
- can place hooks in supported location
- very powerful



# MACF

- very - very powerful
- was part of KDK till OS X 10.12 (never officially supported)
- mac.h header was removed
- available in xnu: `security/mac.h`, `security/mac\_framework.h`, `mac\_vfs.h`
- examples: AppleMobileFileIntegrity, Sandbox, EndpointSecurity, Quarantine (=Gatekeeper)

# MACF

- typical callout from xnu: mac\_.....



```
static int
snapshot_mount(int dirfd, user_addr_t name, user_addr_t directory,
    __unused user_addr_t mnt_data, __unused uint32_t flags, vfs_context_t ctx)
{
    ...
    #if CONFIG_MACF
        error = mac_mount_check_snapshot_mount(ctx, rvp, vp, &dirndp->ni_cnd, snapndp->ni_cnd.cn_nameptr,
            mp->mnt_vfsstat.f_fstypename);
        if (error) {
            goto out2;
        }
    #endif
    ...
}
```

# MACF



```
mac_mount_check_snapshot_mount(vfs_context_t ctx, struct vnode *rvp, struct vnode *vp, struct componentname *cnp,
    const char *name, const char *vfc_name)
{
    kauth_cred_t cred;
    int error;

#ifdef SECURITY_MAC_CHECK_ENFORCE
    /* 21167099 - only check if we allow write */
    if (!mac_vnode_enforce) {
        return 0;
    }
#endif
    cred = vfs_context_ucred(ctx);
    if (!mac_cred_check_enforce(cred)) {
        return 0;
    }
    VFS_KERNEL_DEBUG_START1(92, vp);
    MAC_CHECK(mount_check_snapshot_mount, cred, rvp, vp, cnp, name, vfc_name);
    VFS_KERNEL_DEBUG_END1(92, vp);
    return error;
}
```

# MACF

- MAC\_CHECK
- iterates over all policy frameworks
- mpo\_... (mac\_policy.h)



```
typedef int mpo_mount_check_snapshot_mount_t(  
    kauth_cred_t cred,  
    struct vnode *rvp,  
    struct vnode *vp,  
    struct componentname *cnp,  
    const char *name,  
    const char *vfc_name  
);
```



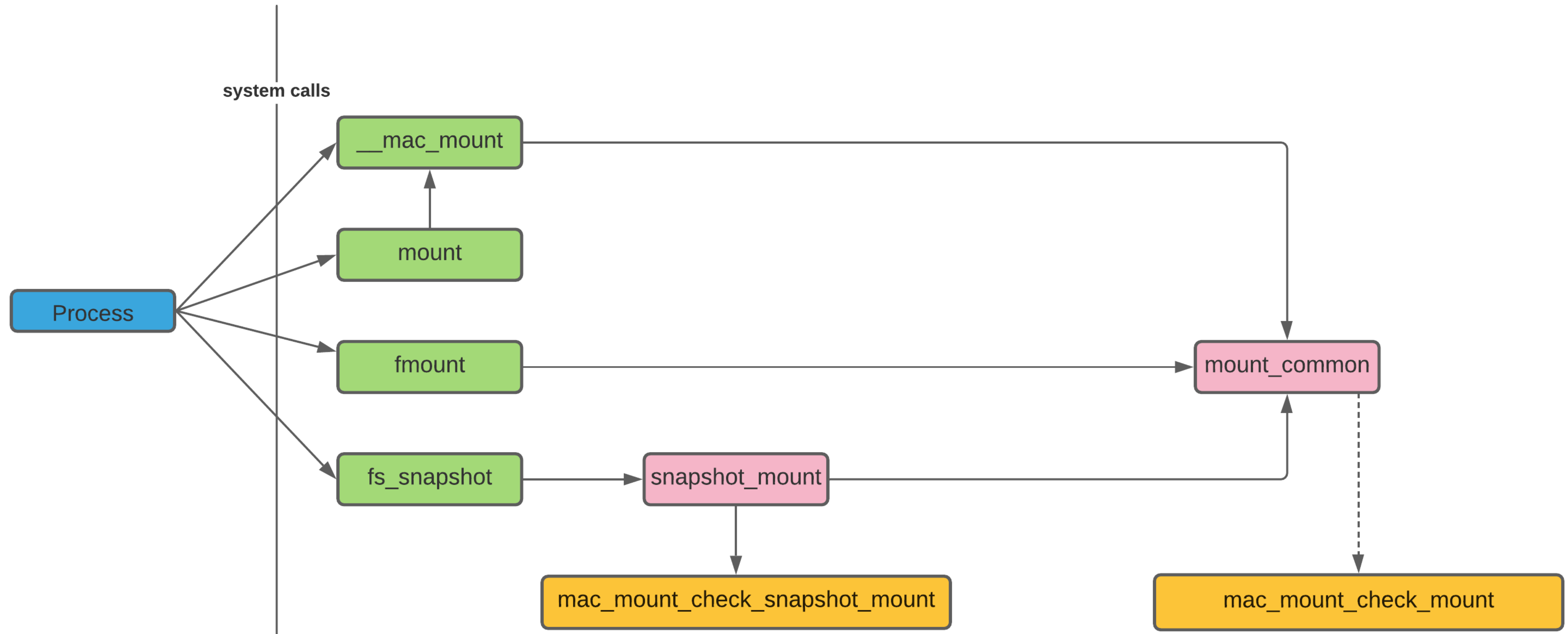
```
#define MAC_CHECK(check, args...) do {  
    struct mac_policy_conf *mpc;  
    u_int i;  
  
    error = 0;  
    for (i = 0; i < mac_policy_list.staticmax; i++) {  
        mpc = mac_policy_list.entries[i].mpc;  
        if (mpc == NULL)  
            continue;  
  
        if (mpc->mpc_ops->mpo_ ## check != NULL)  
            error = mac_error_select(  
                mpc->mpc_ops->mpo_ ## check (args),  
                error);  
    }  
    if (mac_policy_list_conditional_busy() != 0) {  
        for (; i <= mac_policy_list.maxindex; i++) {  
            mpc = mac_policy_list.entries[i].mpc;  
            if (mpc == NULL)  
                continue;  
  
            if (mpc->mpc_ops->mpo_ ## check != NULL)  
                error = mac_error_select(  
                    mpc->mpc_ops->mpo_ ## check (args),  
                    error);  
        }  
        mac_policy_list_unbusy();  
    }  
} while (0)
```

# mount system calls

# VFS

- macOS uses Virtual Filesystem Switch (VFS)
- origin: Solaris
- used in most \*nix based systems
- abstracts specific file system implementation
- MACF callouts at specific points - hooked by the Sandbox

# mount system call flow



# disk arbitration service

# 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")
```

# diskarbitration - the checks

- to prevent abuse, it has to confirm:
  - if the caller sandboxed
  - if the mount point owner == caller process owner (see CVE later)
  - DAServer.c
    - `_DAServerSessionQueueRe`quest function

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

```
if ( audit_token_to_euid( _token ) != DADiskGetUserID( disk ) )  
{  
    status = kDAReturnNotPrivileged;  
}
```

# **CVE-2017-7170 - Sniffing Authorization References on macOS**

# CVE-2017-7170 - credits

- found by Patrick Wardle
- details: [https://objective-see.com/blog/blog\\_0x55.html](https://objective-see.com/blog/blog_0x55.html)

# CVE-2017-7170 - background

- authorization can be passed between processes via `AuthorizationExternalForm`
- `security_authtrampoline`
  - old method
  - executes privileged actions
  - uses external authorization reference

# CVE-2017-7170 - vulnerability

- AuthorizationExecuteWithPrivileges wrote the external form out to \$TMPDIR/randomfile
- write - unlink quickly
- if we can hold of the ref we can use it
- but! we can unlink \$TMPDIR and point it to somewhere else
- RAMdisk
- possibly also works by mounting over \$TMPDIR

```
// create the mailbox file
FILE *mbox = tmpfile();
if (!mbox)
    return errAuthorizationInternal;
if (fwrite(extForm, sizeof(*extForm), 1, mbox) != 1) {
    fclose(mbox);
    return errAuthorizationInternal;
}
fflush(mbox);
```

# CVE-2017-7170 - exploitation

- we create a RAMdisk at \$TMPDIR
- wait for an authorization reference to be written
- scan the raw RAMdisk for the token
- use it once security\_authtrampoline authorized it :)

# **CVE-2017-25333 (pwn2own) - Mount yourself a root shell**

# CVE-2017-25333 - 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-2017-25333 - the vulnerability

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

```
/*  
 * Determine whether the mount point is accessible by the user.  
 */  
  
if ( DADiskGetDescription( disk, kDADiskDescriptionVolumePathKey ) == NULL )  
{  
    if ( DARequestGetUserUID( request ) )  
    {  
        CTypeRef 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 );  
        }  
    }  
}
```

# CVE-2017-25333 - the exploit

- race:
  - create a symlink pointing to a user owned directory
  - call diskarbitration
  - repoint the symlink to a root owned directory after the ownership check
  - if mount successful: stop
- need to mount a disk image where the user has write access (EFI partition)

```
csaby — -bash — 80x24  
sierra:~ csaby$
```



Macintosh HD



Downloads

# **CVE-2020-9771 - TCC bypass via snapshot mounting**

# CVE-2020-9771 - the vulnerability

- APFS supports snapshots
- mount the snapshot in custom location, which falls outside of TCC's protection
- access all files (read-only)
- mount with "noowners" access every user's files



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

# CVE-2020-9771 - the fix

- snapshot mounting requires Full Disk Access => problem
  - even low privilege users can access other's file
- under the hood: MACF callout

```
● ● ●  
#if CONFIG_MACF  
    error = mac_mount_check_snapshot_mount(ctx, rvp, vp, &dirndp->ni_cnd, snapndp->ni_cnd.cn_nameptr,  
        mp->mnt_vfsstat.f_fstypename);  
    if (error) {  
        goto out2;  
    }  
#endif
```

# **CVE-2021-1784 - TCC bypass via mounting over com.apple.TCC**

# CVE-2021-1784 -the vulnerability

- base case: user's TCC DB file is protected
- but! We can mount over the directory
- prepare a new TCC.db file, new disk image
- mount over "~/Library/Application Support/com.apple.TCC"
- profit 🤑
- bug collision with: Mikko Kenttälä (@Turmio\_)

```
hdiutil attach -owners off -mountpoint Library/Application\ Support/com.apple.TCC test.dmg
```

csaby@bigsur ~ %



Favourites

- Recents
- Applications
- Desktop
- Documents
- Downloads
- csaby

iCloud

iCloud Drive

Tags

- Red
- Orange
- Yellow
- Green



# **CVE-2021-30782 - TCC bypass via AppTranslocation service**

# App Translocation

- makes NULLFS mount (not copy) when downloaded app first run
- destination: \$TMPDIR/AppTranslocation/d/d/Some.app
- open source as part of Security.
- library: libsecurity\_translocate
- binary: /usr/libexec/lspd



```
<key>com.apple.private.nullfs_allow</key>  
<true/>  
<key>com.apple.private.tcc.allow</key>  
<array>  
  <string>kTCCServiceSystemPolicyAllFiles</string>  
</array>
```

# CVE-2021-30782 - the vulnerability

- Add Quarantine attribute to "Library"
- Call the com.apple.security.translocation XPC service
- (XPC client is also open source)
- Map Library to \$TMPDIR/AppTranslocation/d/d/Library
- Access all files

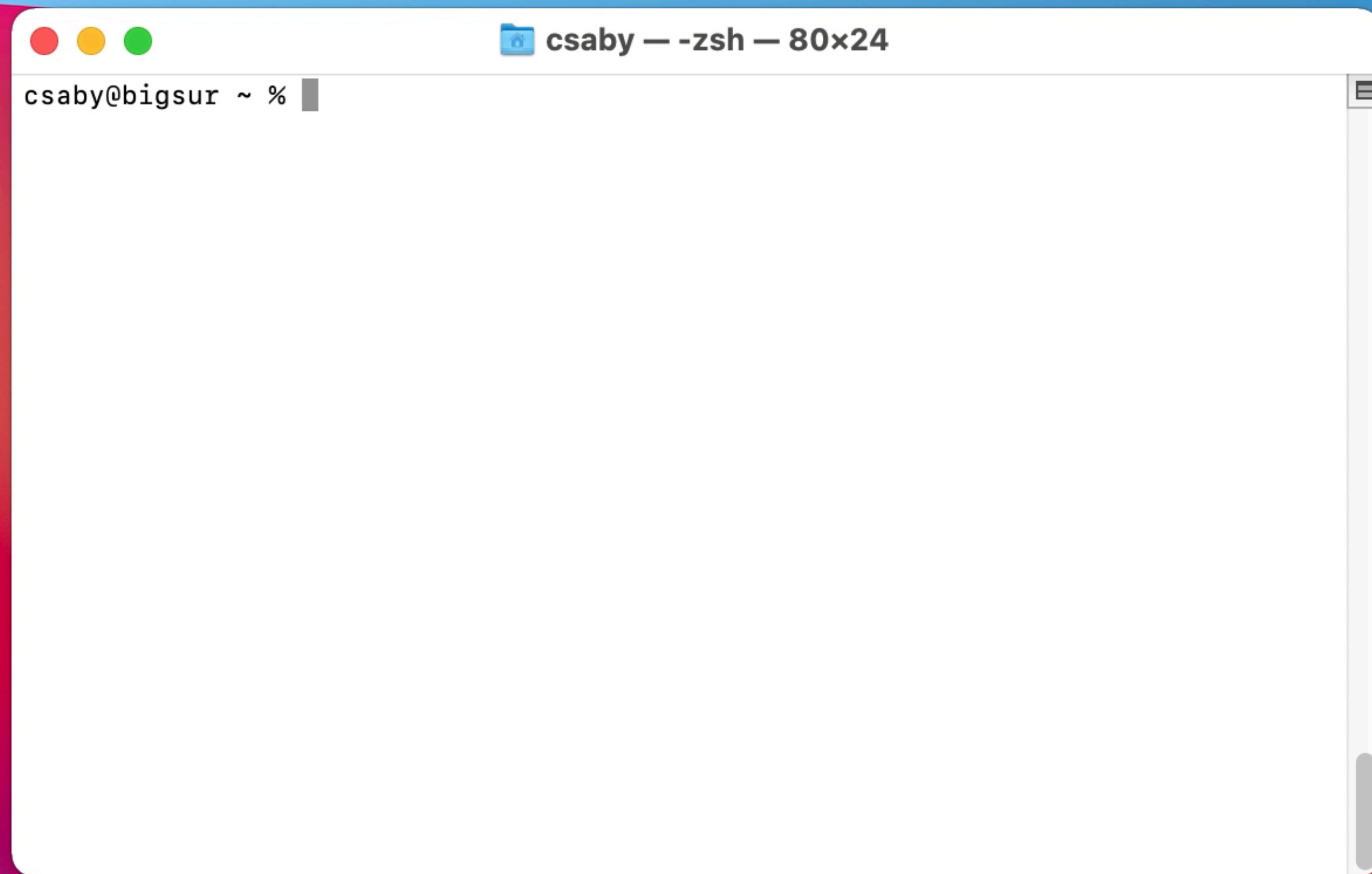
# CVE-2021-30782 - POC

```
//getenv
char *homedir = getenv("HOME");
char *tmpdir = getenv("TMPDIR");

//create paths
char original[MAXPATHLEN];
char destination[MAXPATHLEN];
snprintf(original, sizeof(original), "%s%s", homedir, "/Library");
snprintf(destination, sizeof(destination), "%s%s%s", "/private", tmpdir, "AppTranslocation/d/d/Library");
```

```
xpc_dictionary_set_string(msg, kSecTranslocateXPCMessageFunction, kSecTranslocateXPCFuncCreate);
xpc_dictionary_set_string(msg, kSecTranslocateXPCMessageOriginalPath, original);
xpc_dictionary_set_int64(msg, kSecTranslocateXPCMessageOptions, flags);
xpc_dictionary_set_string(msg, kSecTranslocateXPCMessageDestinationPath, destination);

service = xpc_connection_create_mach_service("com.apple.security.translocation", NULL, 0);
if (service == NULL) {
    perror("xpc_connection_create_mach_service");
}
```



# CVE-2021-30782 - the fix

- new sandbox checks
  - read - source
  - mount - destination
- + sandbox check on mount point access

```
int rc = sandbox_check_by_audit_token(audit_token, "file-read*", SANDBOX_FILTER_DESCRIPTOR, original);
if (rc == 1) {
    secerror("SecTranslocate: XPCServer, doCreate path to translocate disallowed by sandbox");
    UnixError::throwMe(EPERM);
} else if (rc == -1) {
    int error = errno;
    secerror("SecTranslocate: XPCServer, doCreate error checking path to translocate against sandbox");
    UnixError::throwMe(error);
}
if (destFd.isOpen()) {
    rc = sandbox_check_by_audit_token(audit_token, "file-mount", SANDBOX_FILTER_DESCRIPTOR, destFd.fd());
    if (rc == 1) {
        secerror("SecTranslocate: XPCServer, doCreate destination path disallowed by sandbox");
        UnixError::throwMe(EPERM);
    } else if (rc == -1) {
        int error = errno;
        secerror("SecTranslocate: XPCServer, doCreate error checking destination path against sandbox");
        UnixError::throwMe(error);
    }
}
```

# **CVE-2021-26089 - Fortinet FortiClient installer LPE**

# CVE-2021-26089

- the installer uses several log files inside /tmp/
- one: fctinstallpost.log
- being written as root
- fixed content

```
Executing installer script: FortiClient --@@--  
/Volumes/FortiClient/Install.mpkg  
/Volumes/FortiClient/fctdata/conf  
custom fct  
/Volumes/FortiClient/fctdata/custom.conf  
/Library/Application Support/Fortinet/FortiClient/bin  
/Volumes/FortiClient/fctdata/custom.conf is not exist  
No AntiVirus Feature Installed  
No Web Filtering Feature Installed  
No Single Sign On Feature Installed  
No Application Firewall Feature Installed  
sw upgrade option: 1  
start FortiClient....  
duplication run , so quit
```

# CVE-2021-26089

- symlink attack
- we can do arbitrary overwrite
- code execution - ?
- overwrite periodic scripts

```
Executing installer script: FortiClient --@@--  
/Volumes/FortiClient/Install.mpkg  
/Volumes/FortiClient/fctdata/conf  
custom fct  
/Volumes/FortiClient/fctdata/custom.conf  
/Library/Application Support/Fortinet/FortiClient/bin  
/Volumes/FortiClient/fctdata/custom.conf is not exist  
No AntiVirus Feature Installed  
No Web Filtering Feature Installed  
No Single Sign On Feature Installed  
No Application Firewall Feature Installed  
sw upgrade option: 1  
start FortiClient....  
duplication run , so quit
```

# CVE-2021-26089

- the log will be treated as a script
- each line is executed
- pick: /Volumes/FortiClient/fctdata/conf
- make a DMG file, mount it, place a file at the above link
- will be run as root

```
Executing installer script: FortiClient --@@--  
/Volumes/FortiClient/Install.mpkg  
/Volumes/FortiClient/fctdata/conf  
custom fct  
/Volumes/FortiClient/fctdata/custom.conf  
/Library/Application Support/Fortinet/FortiClient/bin  
/Volumes/FortiClient/fctdata/custom.conf is not exist  
No AntiVirus Feature Installed  
No Web Filtering Feature Installed  
No Single Sign On Feature Installed  
No Application Firewall Feature Installed  
sw upgrade option: 1  
start FortiClient....  
duplication run , so quit
```



FortiClient



csaby

- Desktop
- Documents
- Downloads
- FortiClient\_full.dmg**
- FortiClient.dmg
- Movies
- Music
- Pictures
- Public

FortiClient\_full.dmg  
Disk Image - 86,3 MB

Information

FortiClient For Mac OS X



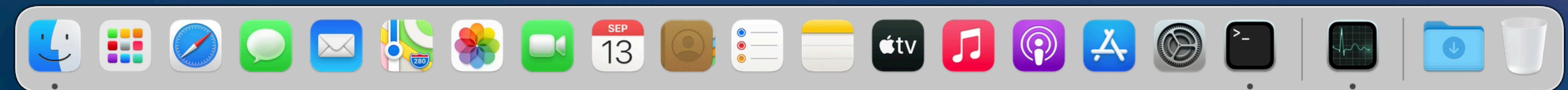
Install



Technical Documentation



Uninstall



# Other generic installer trick

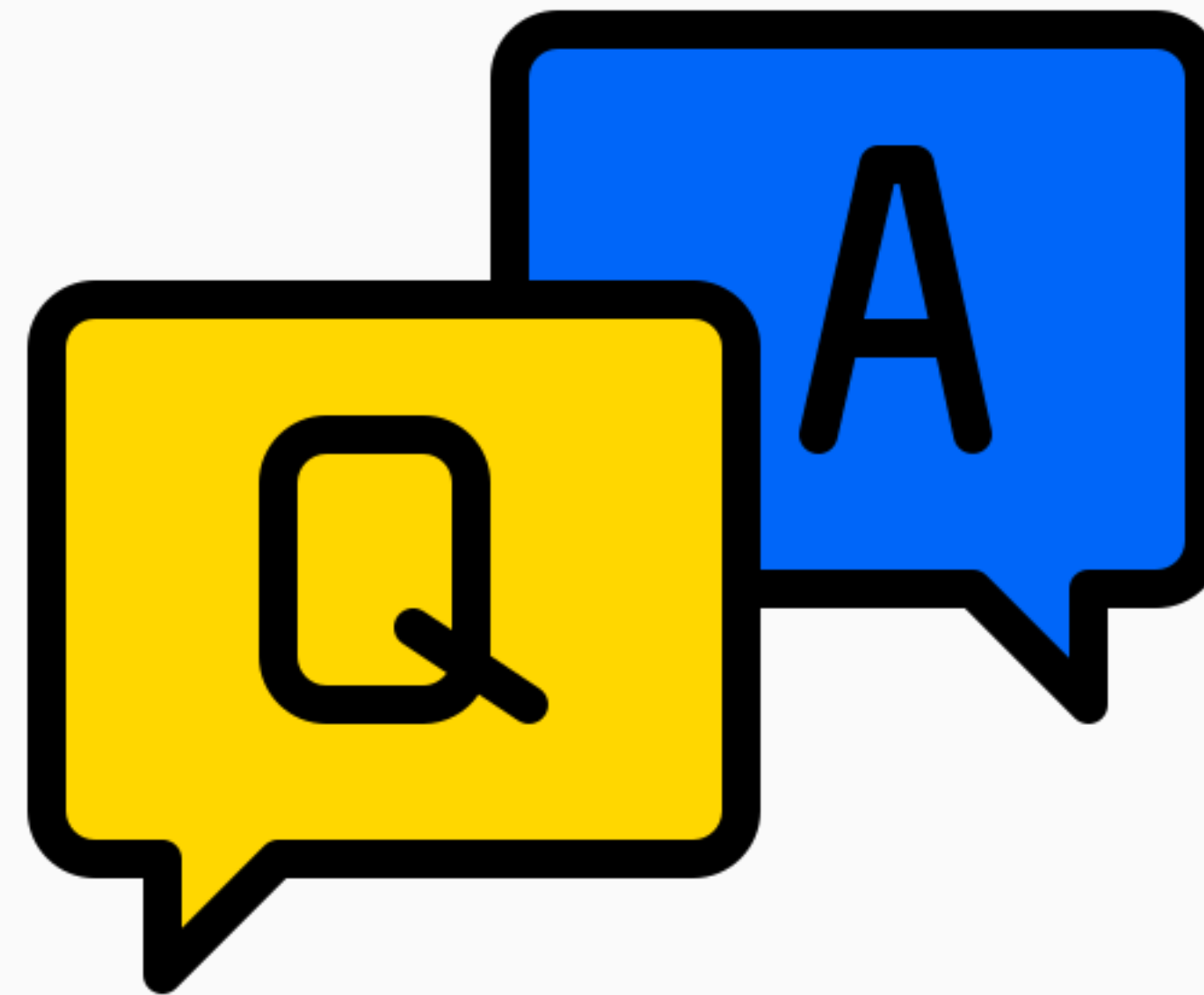
# trick

- installer: writes files to:
  - /tmp/fixedname/bla/bla/bla
- we can create a mount over /tmp/fixedname with noowners
- during install we can modify any files
- no race condition even if installer tries to delete /tmp/fixedname first

**pipeline**

# bugs to be fixed

- full TCC bypass
- full sandbox escape
- admin config TCC bypass



*Csaba Fitzl*  
*Twitter: @theevilbit*

# Further resources

# Links

- <https://www.zerodayinitiative.com/advisories/ZDI-21-693/>
-

# Icons

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