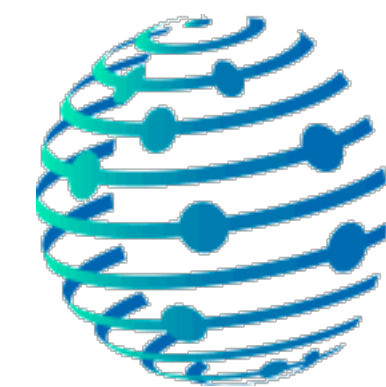
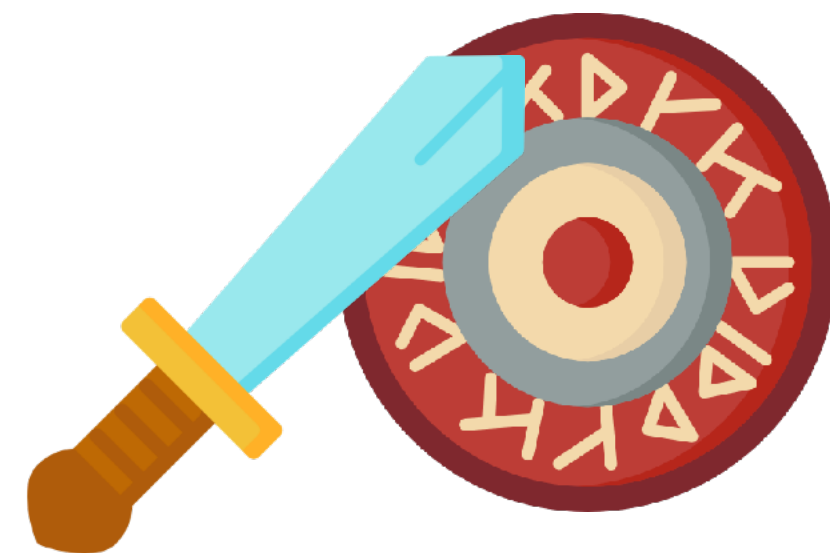


Mitigating exploits using Apple's Endpoint Security

Csaba Fitzl

Twitter: @theevilbit



VB2021
localhost

MAC
DEV
OPS
YVR

whoami

- author of "macOS Control Bypasses" @ Offensive Security
- ex red/blue teamer
- macOS researcher
- husband, father
- hiking 🥾 🏔️
- yoga 🧘



agenda

1. process injection attacks
2. symlink attacks
3. Endpoint Security framework
4. developing the app
5. the app's logic
6. demos

process injection attacks

why important?

- process injection is a big 🚫 on macOS
- access to process's privileges
 - TCC
 - keychain
- impersonate XPC client (XPC LPEs)
- impersonate KEXT client

DYLD_INSERT_LIBRARIES



```
DYLD_INSERT_LIBRARIES=your.dylib /path/to/your.app/Contents/MacOS/your
```

- classic
- SIP kills it (hardened runtime, platform binaries)
 - still can enable it with entitlement -> if possible don't 🙏



```
com.apple.security.cs.allow-dyld-environment-variables
```

DYLIB hijacking and proxying

- discussed in detail by Patrick Wardle in 2015
- plant a dylib the app looks for, or replace one with your own
- SIP / library validation kills it
 - still can enable it with entitlement -> NOOOOO!!!!



- problem: app needs to support 3rd party plugins => bypass TCC (Apple & 3rd parties)

task for pid

- inject by getting the task port
- ⚠ debug builds with bad entitlement ⚠
- SIP kills this
- notarization checks for `com.apple.security.get-task-allow`



`com.apple.security.get-task-allow`

Electron

- so much broken
- env vars
- debug ports



```
csaby@mac ~ % ELECTRON_RUN_AS_NODE=1 /Applications/Discord.app/Contents/MacOS/Discord
Welcome to Node.js v12.14.1.
Type ".help" for more information.
>
```



```
csaby@mac ~ % /Applications/Discord.app/Contents/MacOS/Discord --inspect
Debugger listening on ws://127.0.0.1:9229/8967d388-33a2-4f9f-82bc-19072a7c8e76
For help, see: https://nodejs.org/en/docs/inspector
Discord 0.0.262
```

The screenshot shows the DevTools interface with a remote target selected. The console log displays the following messages:

- 94 messages**
- 16 user messages**
- 1 error**
- 78 warnings**
- 15 info**
- No verbose**

The console log shows a deprecation warning at the top:

```
(node:88180) [DEP0005] DeprecationWarning: Buffer() is deprecated due to security and usability issues. Please use the Buffer.alloc(), Buffer.allocUnsafe(), or Buffer.from() methods instead.
```

Below the warning, the console shows the application starting and initializing modules:

```
Starting app.
Starting updater.
[Modules] Modules initializing
[Modules] Distribution: remote
[Modules] Host updates: enabled
[Modules] Module updates: enabled
[Modules] Module install path: /Users/gandalf/Library/Application Support/discord/0.0.262/modules
[Modules] Module installed file path: /Users/gandalf/Library/Application Support/discord/0.0.262/modules/installed.json
[Modules] Module download path: /Users/gandalf/Library/Application Support/discord/0.0.262/modules/pending
[Modules] No updates to install
[Modules] Checking for host updates.
[Modules] Host is up to date.
[Modules] Checking for module updates at https://discord.com/api/modules/stable/versions.json
[Modules] No module updates available.
```

vulnerabilities

- CVE-2020-26893 - ClamXAV AntiVirus XPC LPE
- CVE-2020-29621 - coreaudiod TCC bypass
- CVE-2020-25736 - Acronis True Image 2021 XPC LPE
- CVE-2020-24259 - Signal macOS TCC bypass
- CVE-2020-14978 - F-Secure XPC
- ...

symlink attacks

the approach

- process running as root writes or modifies files at a user controllable location
- place a symlink or hardlink, pointing to a root accessible location

vulnerabilities

- CVE-2020-9900 - Crash Reporter LPE
- CVE-2021-1786 - Crash Reporter arbitrary file deletion
- CVE-2020-3855 - macOS DiagnosticMessages arbitrary file overwrite
- CVE-2020-3762 - Adobe installer LPE
- ...

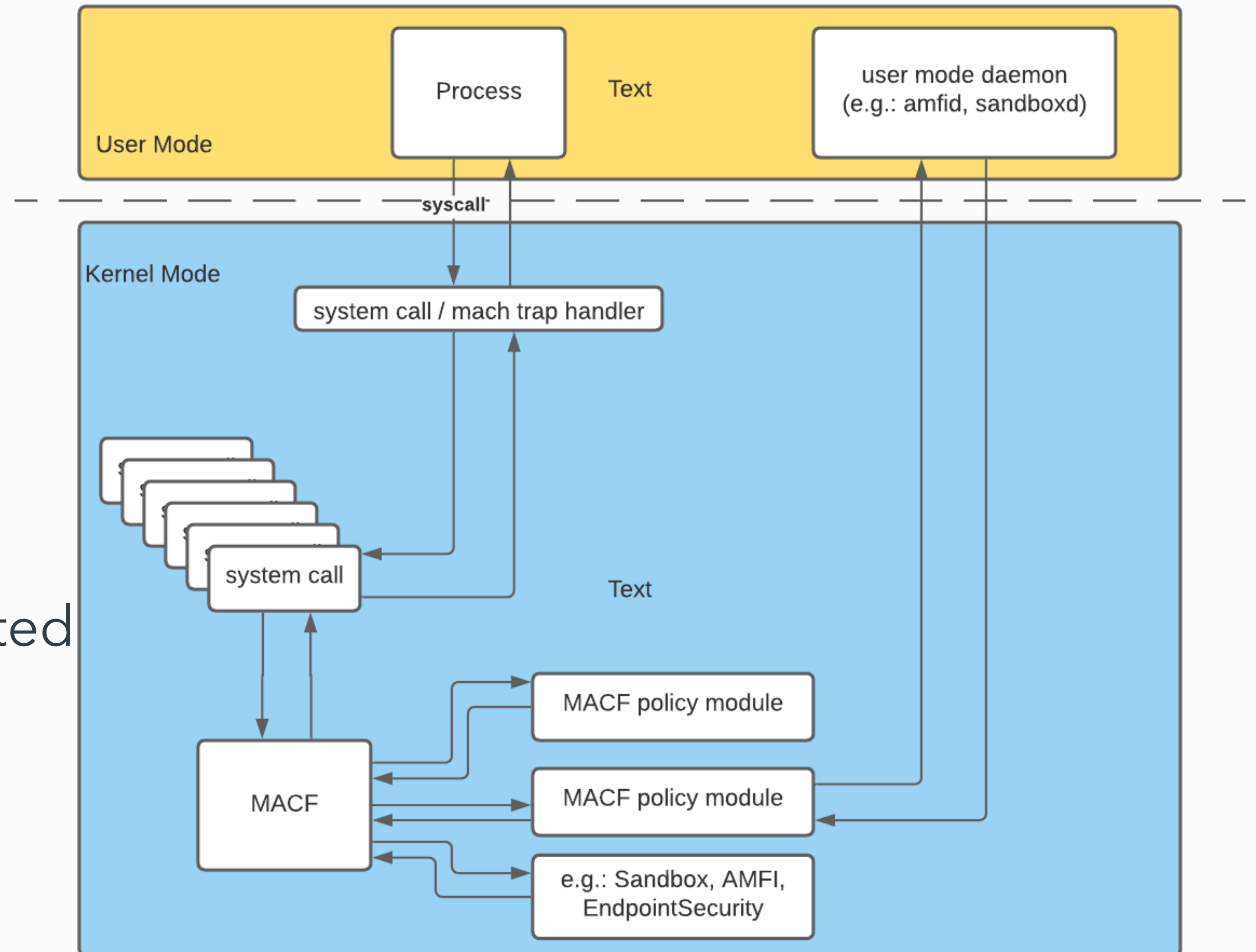
MACF

-

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`
- 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  
);
```

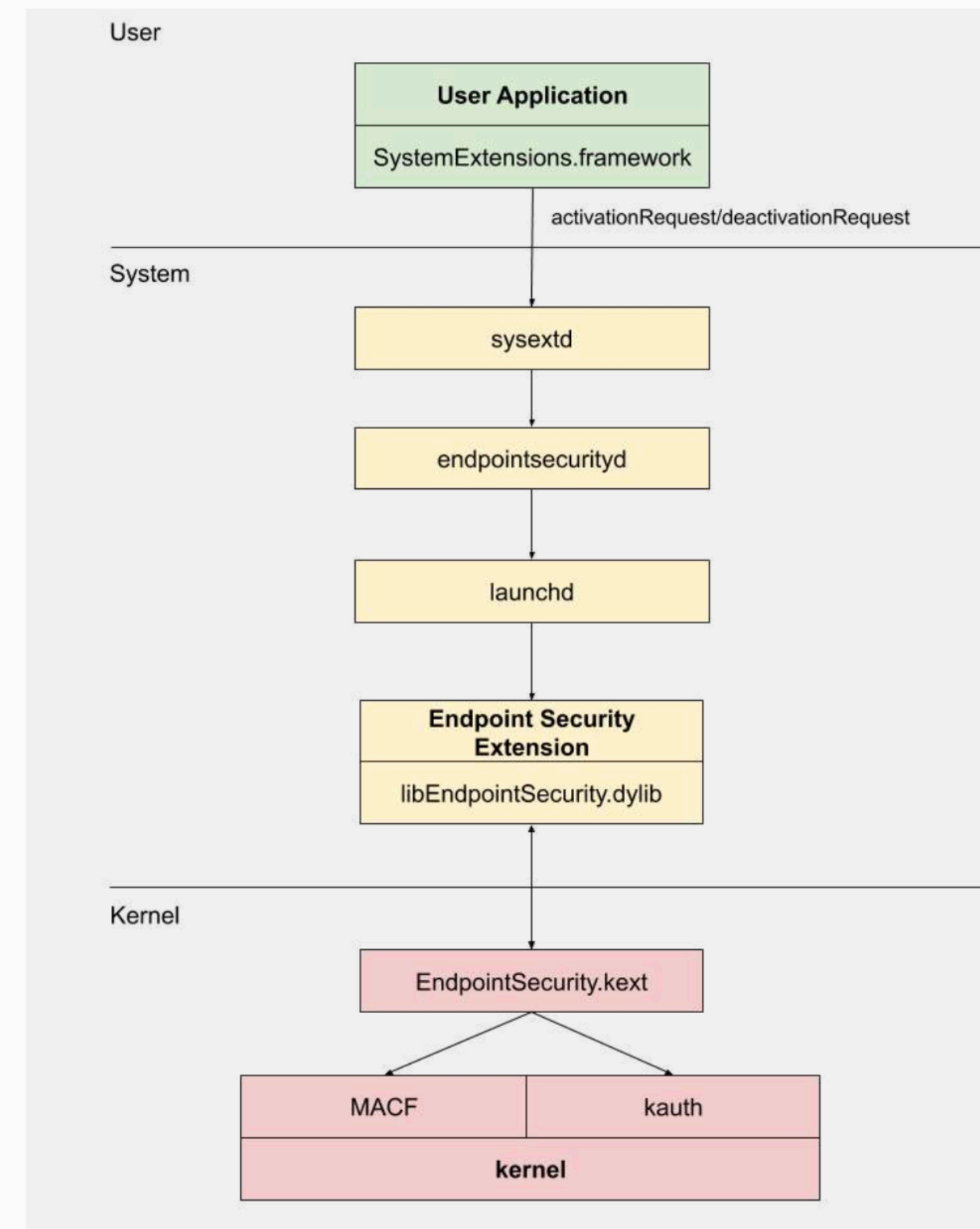


```
#define MAC_CHECK(check, args...) do {  
    struct mac_policy_conf *mpc;  
    u_int i;  
  
    error = 0;  
    for (i = 0; i < mac_policy_list.staticmax; i++) {  
        mpc = mac_policy_list.entries[i].mpc;  
        if (mpc == NULL)  
            continue;  
  
        if (mpc->mpc_ops->mpo_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

Endpoint Security

ES

- KEXT - MACF, kauth
- dylib - C API for clients
- endpointsecurityd - loading SEXT via launchd
- sysextd - validation and copy
- SystemExtension.framework - activation and deactivation of the extension
- systemextensionsctl - basic control of sysxextd
- more: Scott Knight's OBTS talk



ES

- MACF policy (EndpointSecurity)

- ~60 hooks

```
EndpointSecurityEventManager::es_vnode_check_access(ucred *,vnode *,label *,int) -
EndpointSecurityEventManager::es_vnode_check_chdir(ucred *,vnode *,label *) -
EndpointSecurityEventManager::es_vnode_check_chroot(ucred *,vnode *,label *,componentnam... -
EndpointSecurityEventManager::es_vnode_check_clone(ucred *,vnode *,label *,vnode *,label *,co... -
EndpointSecurityEventManager::es_vnode_check_create(ucred *,vnode *,label *,componentnam... -
EndpointSecurityEventManager::es_vnode_check_deleteextattr(ucred *,vnode *,label *,char cons... -
EndpointSecurityEventManager::es_vnode_check_exchangedata(ucred *,vnode *,label *,vnode *,l... -
EndpointSecurityEventManager::es_vnode_check_fsgetpath(ucred *,vnode *,label *) -
EndpointSecurityEventManager::es_vnode_check_getattrlist(ucred *,vnode *,label *,attrlist *) -
EndpointSecurityEventManager::es_vnode_check_getextattr(ucred *,vnode *,label *,char const*,... -
EndpointSecurityEventManager::es_vnode_check_link(ucred *,vnode *,label *,vnode *,label *,com... -
EndpointSecurityEventManager::es_vnode_check_listextattr(ucred *,vnode *,label *) -
EndpointSecurityEventManager::es_vnode_check_lookup_preflight(ucred *,vnode *,label *,char c... -
EndpointSecurityEventManager::es_vnode_check_open(ucred *,vnode *,label *,int) -
EndpointSecurityEventManager::es_vnode_check_readdir(ucred *,vnode *,label *) -
EndpointSecurityEventManager::es_vnode_check_readlink(ucred *,vnode *,label *) -
EndpointSecurityEventManager::es_vnode_check_rename(ucred *,vnode *,label *,vnode *,label *,... -
EndpointSecurityEventManager::es_vnode_check_searchfs(ucred *,vnode *,label *,attrlist *) -
EndpointSecurityEventManager::es_vnode_check_set_utimes(ucred *,vnode *,label *,timespec,ti... -
EndpointSecurityEventManager::es_vnode_check_setacl(ucred *,vnode *,label *,kauth_acl *) -
EndpointSecurityEventManager::es_vnode_check_setattrlist(ucred *,vnode *,label *,attrlist *) -
EndpointSecurityEventManager::es_vnode_check_setextattr(ucred *,vnode *,label *,char const*,... -
EndpointSecurityEventManager::es_vnode_check_setflags(ucred *,vnode *,label *,ulong) -
EndpointSecurityEventManager::es_vnode_check_setmode(ucred *,vnode *,label *,ushort) -
EndpointSecurityEventManager::es_vnode_check_setowner(ucred *,vnode *,label *,uint,uint) -
EndpointSecurityEventManager::es_vnode_check_stat(ucred *,ucred *,vnode *,label *) -
EndpointSecurityEventManager::es_vnode_check_truncate(ucred *,ucred *,vnode *,label *) -
EndpointSecurityEventManager::es_vnode_check_uipc_bind(ucred *,vnode *,label *,componentn... -
EndpointSecurityEventManager::es_vnode_check_uipc_connect(ucred *,vnode *,label *,__socket... -
EndpointSecurityEventManager::es_vnode_check_unlink(ucred *,vnode *,label *,vnode *,label *,c... -
EndpointSecurityEventManager::es_vnode_check_write(ucred *,ucred *,vnode *,label *) -
EndpointSecurityEventManager::es_vnode_notify_create(ucred *,mount *,label *,vnode *,label *,v...
```

```
EndpointSecurityEventManager::es_cred_check_label_update_execve(ucred *,vnode *,long long,... __text
EndpointSecurityEventManager::es_cred_label_associate_fork(ucred *,proc *) __text
EndpointSecurityEventManager::es_cred_label_update_execve(ucred *,ucred *,proc *,vnode *,lon... __text
EndpointSecurityEventManager::es_file_check_dup(ucred *,fileglob *,label *,int) __text
EndpointSecurityEventManager::es_file_check_fcntl(ucred *,fileglob *,label *,int,long long) __text
EndpointSecurityEventManager::es_file_check_mmap(ucred *,fileglob *,label *,int,int,ulong long,i... __text
EndpointSecurityEventManager::es_file_notify_close(ucred *,fileglob *,label *,int) __text
EndpointSecurityEventManager::es_iokit_check_open(ucred *,OSObject *,uint) __text
EndpointSecurityEventManager::es_kext_check_load(ucred *,char const*) __text
EndpointSecurityEventManager::es_kext_check_unload(ucred *,char const*) __text
EndpointSecurityEventManager::es_mount_check_mount_late(ucred *,mount *) __text
EndpointSecurityEventManager::es_mount_check_remount(ucred *,mount *,label *) __text
EndpointSecurityEventManager::es_mount_check_umount(ucred *,mount *,label *) __text
EndpointSecurityEventManager::es_policy_syscall(proc *,int,ulong long) __text
EndpointSecurityEventManager::es_proc_check_debug(ucred *,proc_ident *) __text
EndpointSecurityEventManager::es_proc_check_expose_task(ucred *,proc_ident *) __text
EndpointSecurityEventManager::es_proc_check_get_task(ucred *,proc_ident *) __text
EndpointSecurityEventManager::es_proc_check_get_task_name(ucred *,proc_ident *) __text
EndpointSecurityEventManager::es_proc_check_mprotect(ucred *,proc *,ulong long,ulong long,int) __text
EndpointSecurityEventManager::es_proc_check_proc(ucred *,proc *,int,int) __text
EndpointSecurityEventManager::es_proc_check_remote_thread_create(ucred *,proc *,int,uint *,u... __text
EndpointSecurityEventManager::es_proc_check_signal(ucred *,proc *,int) __text
EndpointSecurityEventManager::es_proc_check_suspend_resume(ucred *,proc *,int) __text
EndpointSecurityEventManager::es_proc_notify_cs_invalidated(proc *) __text
EndpointSecurityEventManager::es_proc_notify_exec_complete(proc *) __text
EndpointSecurityEventManager::es_proc_notify_exit(proc *) __text
EndpointSecurityEventManager::es_pty_notify_close(proc *,tty *,int,label *) __text
EndpointSecurityEventManager::es_pty_notify_grant(proc *,tty *,int,label *) __text
EndpointSecurityEventManager::es_system_check_settime(ucred *) __text
EndpointSecurityEventManager::es_thread_userret(thread *) __text
```

ES

- user mode events are mapped to kernel MACF hooks
- examples:
 - `ES_EVENT_TYPE_NOTIFY_CHROOT` - `es_vnode_check_chroot`
 - `ES_EVENT_TYPE_NOTIFY_MOUNT` - `es_mount_check_mount_late`
 - `ES_EVENT_TYPE_NOTIFY_MMAP` - `es_file_check_mmap`
 - `ES_EVENT_TYPE_AUTH_GET_TASK` - `es_proc_check_get_task`

ES

- very powerful!!!
- extending MACF to user mode
- MACF was never officially supported
- now we have in user mode ❤️

Shield.app development

requirements

- entitlement: com.apple.developer.endpoint-security.client
- Apple's good will

getting entitled

- 2020 March - requested ES entitlement
- 2020 April - got developer version
- 2020 - emails going to "black hole" at Apple
- ...
- 2021 January - got the entitlement
- frustration, demotivation, annoyed, extremely bad experience - luckily I don't do this for living



sources

- used Patrick Wardle's ProcessMonitor and FileMonitor
- also reviewed Stephen Davis's Crescendo



es_client



```
es_client_t* endpointClient = nil;

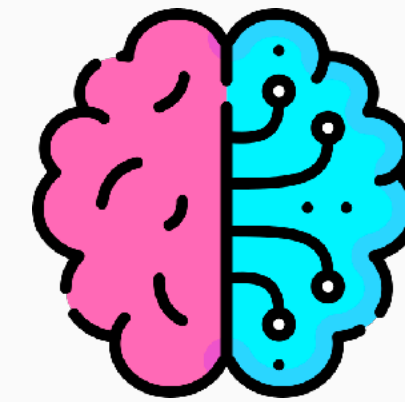
es_event_type_t events[] = {ES_EVENT_TYPE_NOTIFY_EXEC, ...};

es_new_client(&endpointClient, ^(es_client_t *client, const es_message_t *message){
    //callback
    switch (message->event_type) {

        case ES_EVENT_TYPE_NOTIFY_EXEC:
            //do stuff
    });

es_subscribe(endpointClient, events, sizeof(events)/sizeof(events[0]))
```

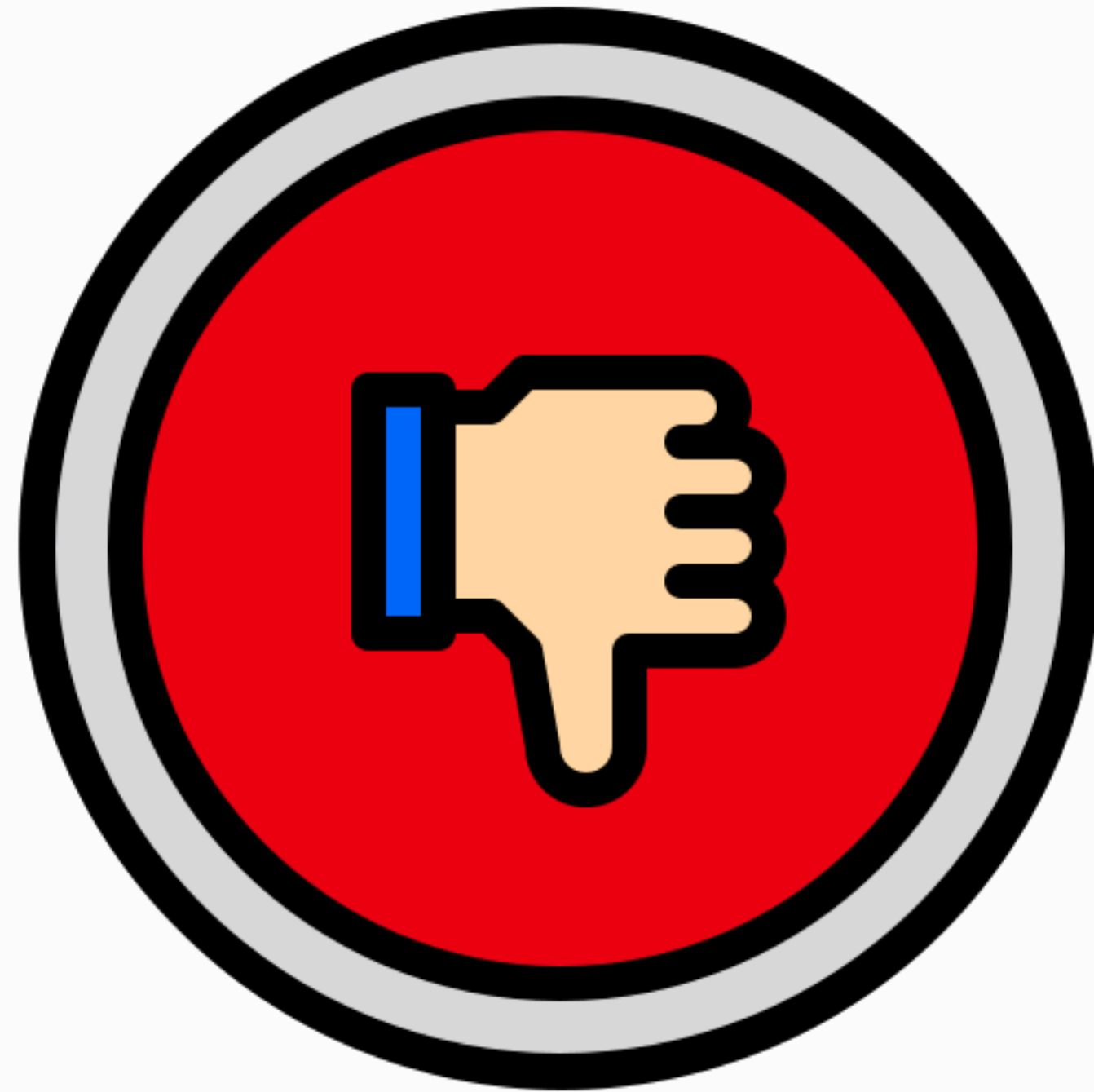
the logic



ES_EVENT_TYPE_AUTH_EXEC

- checks:
 - argument
 - --inspect, --inspect-brk, --remote-debugging-port
 - environment variables
 - DYLD_INSERT_LIBRARIES
 - CFNETWORK_LIBRARY_PATH
 - RAWCAMERA_BUNDLE_PATH
 - ELECTRON_RUN_AS_NODE

ES_EVENT_TYPE_AUTH_GET_TASK



ES_EVENT_TYPE_AUTH_MMAP

- dylib injection protection
- "enforce" library validation
- slow - disk I/O



```
//set req string, teamid = of the process  
//anchor apple = apple's own binary - safe  
//anchor apple generic and certificate leaf [subject.CN] = \"Apple Mac OS Application Signing\" - app store, assume  
safe  
//anchor apple generic and certificate leaf[subject.OU] = \"%@\" - match dev teamid  
  
NSString *requirementString = [NSString stringWithFormat:@"(anchor apple) or (anchor apple generic and certificate  
leaf [subject.CN] = \"Apple Mac OS Application Signing\") or (anchor apple generic and certificate leaf[subject.OU]  
= \"%@\")", process.teamID];
```

ES_EVENT_TYPE_AUTH_LINK



```
([file_uid intValue] != file.process.uid) && !([file_uid intValue] > 0 && file.process.uid == 0)
```

- event for hardlinks
- low privilege process isn't allowed to point to high privilege location

ES_EVENT_TYPE_NOTIFY_CREATE



```
([file_uid intValue] != file.process.uid) && !([file_uid intValue] > 0 && file.process.uid == 0)
```

- track symbolic links
- low privilege process isn't allowed to point to high privilege location
- detect only - don't know the target before creation

demo - w/o Shield

CVE-2020-26893 - ClamXAV AntiVirus

XPC LPE





Macintosh HD

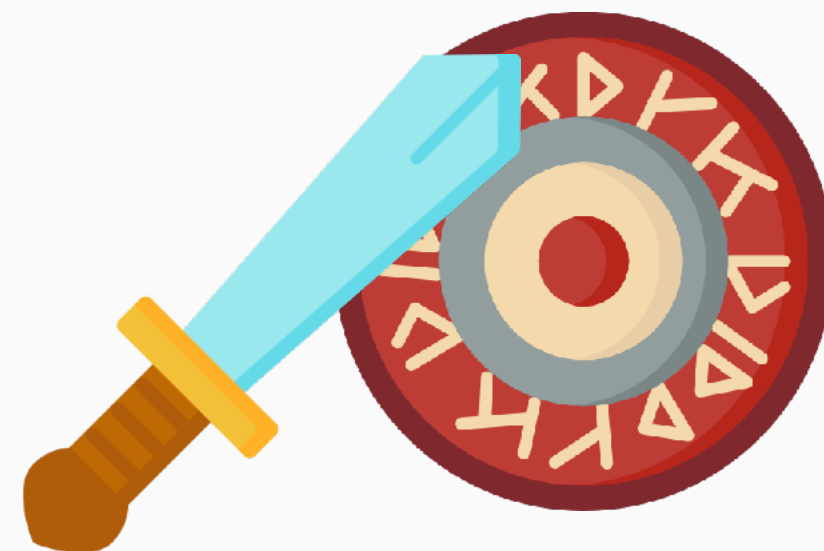
```
csaby — -zsh — 104x24
[csaby@bigsur ~ % ls -l /Library/LaunchDaemons/
total 32
-r--r--r--  1 root  wheel  1156 Aug 10  2020 com.vmware.launchd.tools.plist
-rw-r--r--@ 1 root  wheel   614 Aug 11  2020 uk.co.canimaansoftware.ClamXAV.Engine.plist
-r-xr--r--  1 root  wheel   686 Aug 11  2020 uk.co.canimaansoftware.ClamXAV.HelperTool.plist
-r-xr--r--  1 root  wheel   652 Aug 11  2020 uk.co.canimaansoftware.ClamXAV.HelperToolUpdater.plist
csaby@bigsur ~ % █
```

I

demo - w/ Shield

CVE-2020-26893 - ClamXAV AntiVirus

XPC LPE





Macintosh HD

File browser window titled "csaby" showing a file named "clamexp.dylib".

Terminal window titled "csaby -- zsh -- 104x24" showing the following command and output:

```
csaby@bigsur ~ % ls -l /Library/LaunchDaemons/  
total 32  
-r--r--r--  1 root  wheel  1156 Aug 10  2020 com.vmware.launchd.tools.plist  
-rw-r--r--@ 1 root  wheel   614 Aug 11  2020 uk.co.canimaansoftware.ClamXAV.Engine.plist  
-r-xr--r--  1 root  wheel   686 Aug 11  2020 uk.co.canimaansoftware.ClamXAV.HelperTool.plist  
-r-xr--r--  1 root  wheel   652 Aug 11  2020 uk.co.canimaansoftware.ClamXAV.HelperToolUpdater.plist  
csaby@bigsur ~ % ls -l /Library/LaunchDaemons/
```

wrap up

- injection and file link attacks responsible for many logic bugs
- ES framework is based on MACF
- ES extends MACF to user mode, very powerful
- can be used to detect and block logic attacks
- it's a pain to get the ES entitlement



Csaba Fitzl
Twitter: @theevilbit

Further resources

- Wojciech Reguła (@_r3ggi): Abusing and Securing XPC in macOS Apps Objective by the Sea v3
- Julia Vashchenko (@iaronnskaya): Job(s) Bless Us! Privileged Operations on macOS Objective by the Sea v3
- Tyler Bohan (@1blankwall1): OSX XPC Revisited - 3rd Party Application Flaws OffensiveCon 19
- Ian Beer (@i41nbeer): A deep-dive into the many flavors of IPC available on OS X Jailbreak Security Summit 2015

Links

- <http://www.trustedbsd.org/mac.html>
- <https://blog.xpnsec.com/mac-os-injection-via-third-party-frameworks/>
- <https://www.offensive-security.com/offsec/amfi-syscall/>
- <https://www.semanticscholar.org/paper/New-approaches-to-operating-system-security-Watson/f89682c6cf943ce349031270e685ee2dddee9376>
- <https://knight.sc/reverse%20engineering/2019/08/24/system-extension-internals.html>
- <http://newosxbook.com/articles/eps.html>
- <https://github.com/xorrior/goesf/blob/master/appmon.m>
- <https://github.com/theevilbit/Shield>

Icons

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