

Every Authorization Has Its Black: Tackling Privilege Escalation in macOS

黃智威 Will

Whoami

Jr-Wei Huang

- ◆ Software Developer @ TeamT5
- ◆ Member of 10sec

Research Topic

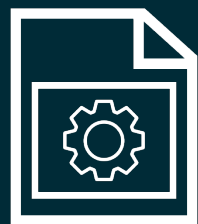
- ◆ System security (Linux, MacOS)
- ◆ Malware analysis
- ◆ Threat hunting



Malicious Behavior Overview

MacOS-based Attack Flow

- ◆ Malware with **privilege escalation** exploit
 - ◆ XCSSET.2020, MacMa.2021
 - ◆ dazzlespy.2022, CloudMensis.2022
 - ◆ Others: supply chain, electorn inject, browser hijacking...



Create & Open & Write

Execute

Privilege Escalation

Attack

Our Goals



- ◆ Demystify macOS privilege management
- ◆ Explain what we saw about the attack surface of **privilege escalation**
- ◆ Building a **mitigate module** to detect this kind of attack

Takeaway

Understand the macOS privilege attack surface and hunt for techniques that threat actors use to target macOS

Outline

01 Privilege Management on macOS

02 Common Ways to Gain Elevated Access

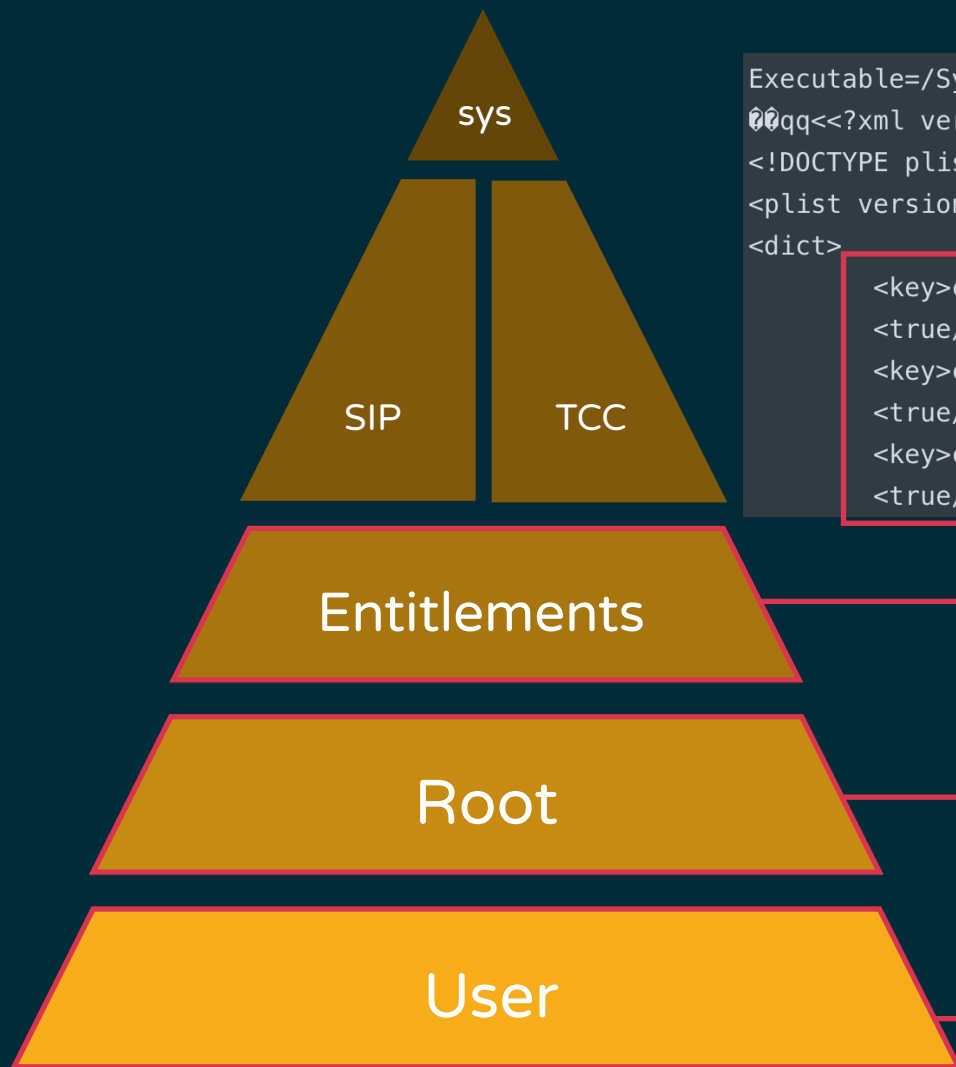
03 EndpointSecurity Framework

04 onPrivilege Detection with ESF

05 Conclusion

Privilege Management on macOS

Privilege Management



```
Executable=/System/Library/PrivateFrameworks/PackageKit.framework/Versions/A/Resources/system_installd
00qq<<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
  <key>com.apple.private.launchservices.cansetapplicationtrusted</key>
  <true/>
  <key>com.apple.private.package_script_service.allow</key>
  <true/>
  <key>com.apple.private.responsibility.set-arbitrary</key>
  <true/>
</dict>
</plist>
</xml>
```

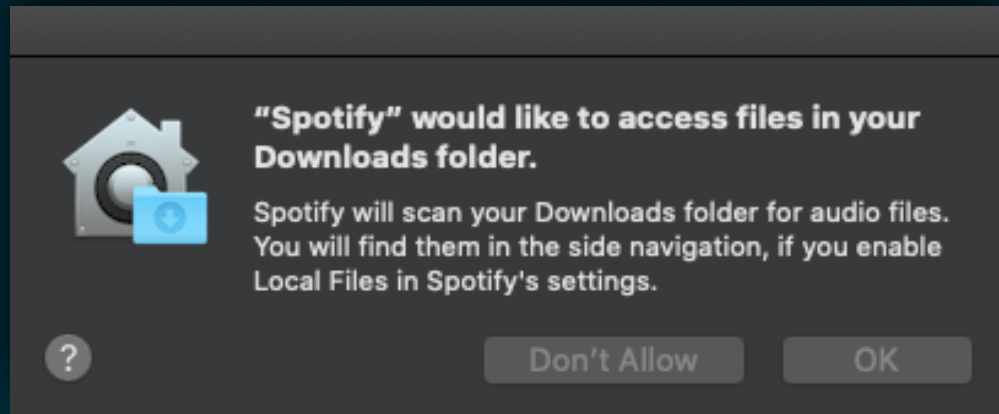
Install package, add autorun item, Debugging

General purpose

Privilege Management

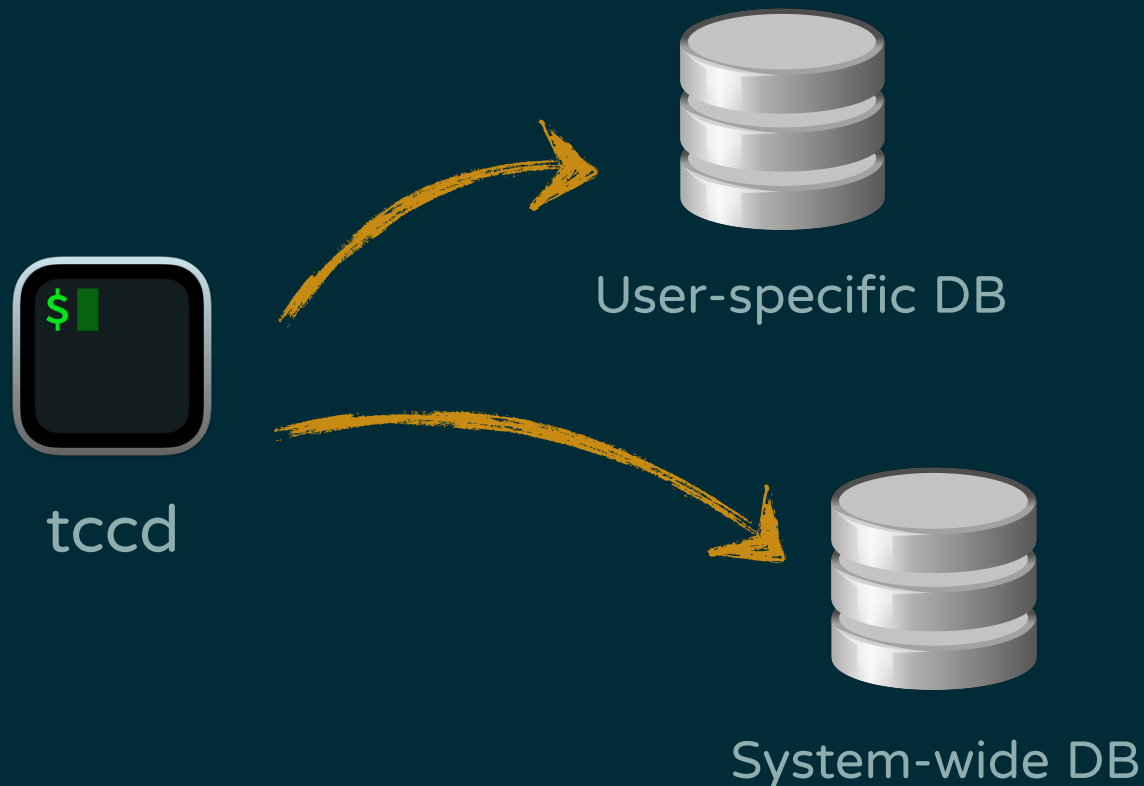
TCC (Transparency, Consent, and Control)

- ◆ Restrict installed applications to access sensitive data without permission
- ◆ User data access control



TCC

- ◆ User-specific database: ~/Library/Application Support/com.apple.TCC/TCC.db
- ◆ System-wide database: /Library/Application Support/com.apple.TCC/TCC.db



access	
service	TEXT
client	TEXT
client_type	INTEGER
auth_value	INTEGER
auth_reason	INTEGER
auth_version	INTEGER
csreq	BLOB
policy_id	INTEGER
indirect_object_identifier_type	INTEGER
indirect_object_identifier	TEXT
indirect_object_code_identity	BLOB
flags	INTEGER
last_modified	INTEGER

Privilege Management

SIP (System Integrity Protection)

- ◆ Also known as **rootless**
- ◆ Based on Sandbox kernel extension
- ◆ Restrict root to access sensitive data

```
lrwxr-xr-x    1 root  wheel  restricted    25  1  1  2020 X11 -> ../private/var/select/X11
lrwxr-xr-x    1 root  wheel  restricted    25  1  1  2020 X11R6 -> ../private/var/select/X11
drwxr-xr-x  1036 root  wheel  restricted  33152  1  1  2020 bin
drwxr-xr-x   38 root  wheel  restricted   1216  1  1  2020 lib
drwxr-xr-x  295 root  wheel  restricted   9440  1  1  2020 libexec
drwxr-xr-x   15 root  wheel  sunlnk       480  12  1  09:56 local
drwxr-xr-x  232 root  wheel  restricted   7424  1  1  2020 sbin
drwxr-xr-x   47 root  wheel  restricted   1504  1  1  2020 share
drwxr-xr-x    6 root  wheel  restricted    192  1  1  2020 standalone
```

Privilege Management

SIP (System Integrity Protection)

- ◆ Also known as **rootless**
- ◆ Based on Sandbox kernel extension
- ◆ Restrict root to access sensitive data
 - ✓ Loading kernel extensions (only signed extensions are allowed)
 - ✓ Tampering with critical files and directories
 - ✓ Debugging system processes

SIP

`com.apple.rootless.install`

- ◆ Completely bypass SIP filesystem check

`com.apple.rootless.install.heritable`

- ◆ Inherit `com.apple.rootless.install` to child processes

- ◆ Apple won't allow any application to be signed with the above entitlements
- ◆ Only a few services have this. ex: `system_installd`

TCC - Full Disk Access

The TCC policy (**Full Disk Access**) allows access to SIP-protected directory



TCC - Full Disk Access

The TCC policy (**Full Disk Access**) allows access to SIP-protected directory

```
[sonar@sonars-Mac-mini Downloads % ls ~/Library/Mail
ls: /Users/sonar/Library/Mail: Operation not permitted
[sonar@sonars-Mac-mini Downloads % ls /Library/Application\ Support/com.apple.TCC/
ls: /Library/Application Support/com.apple.TCC/: Operation not permitted
sonar@sonars-Mac-mini Downloads % █
```



-zsh

```
sonar@sonars-Mac-mini ~ % ls /Library/Application\ Support/com.apple.TCC/
AdhocSignatureCache  TCC.db
sonar@sonars-Mac-mini ~ % ls ~/Library/Mail
V8
sonar@sonars-Mac-mini ~ % █
```

Have “Full Disk Access”

TCC - Full Disk Access

Full Disk Access

- ◆ When admin gives an APP Full Disk Access, all users can use this APP to gain Full Disk Access



iTerm

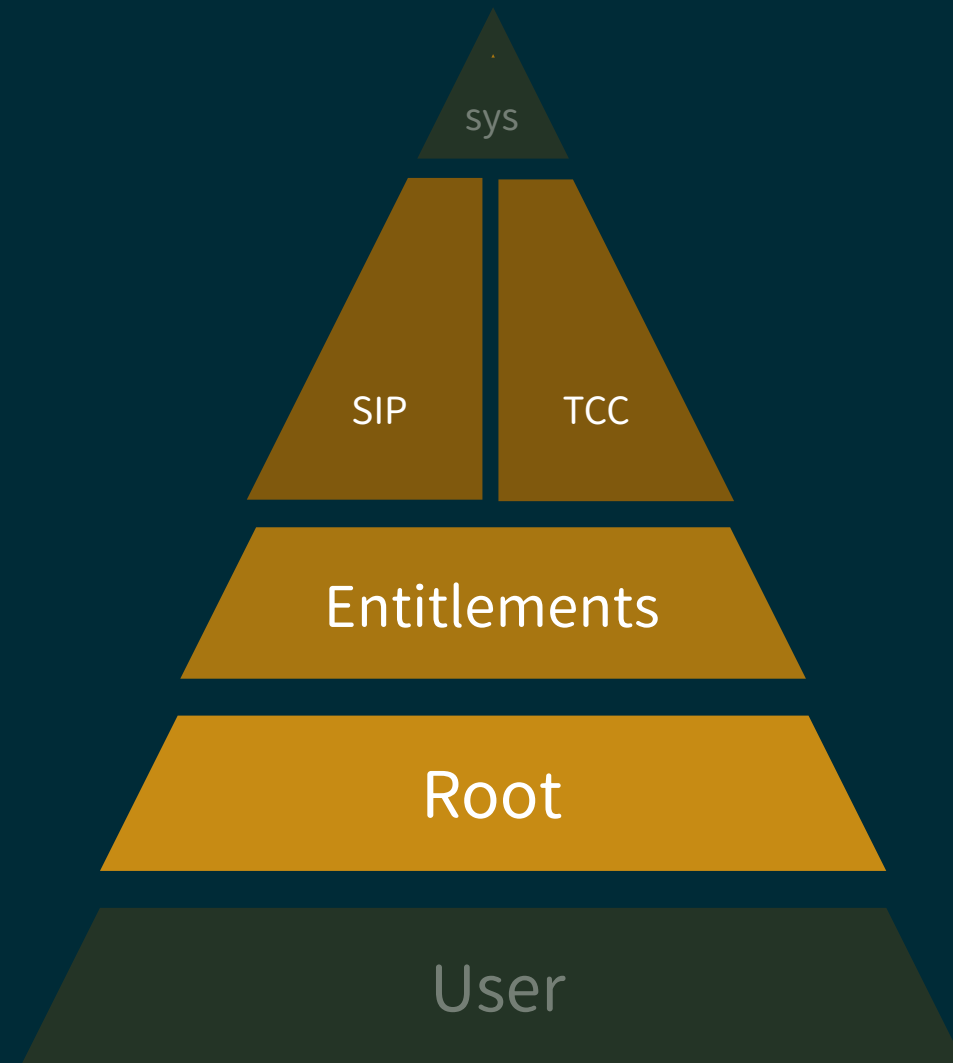
(Full Disk Access)



Common Ways to Gain Elevated Access

Gaining Elevated Access

- ◆ Get Root
- ◆ Bypass SIP
- ◆ Bypass TCC

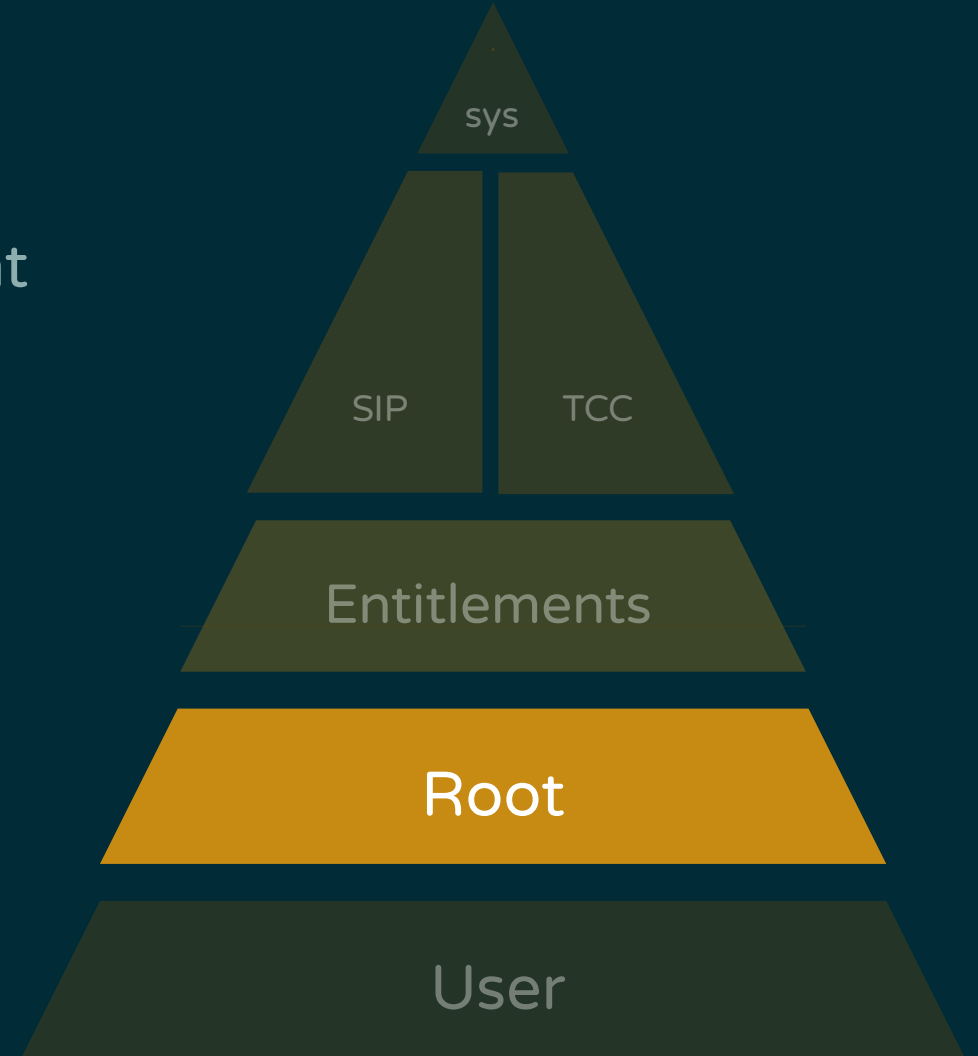


Get Root

Persistence

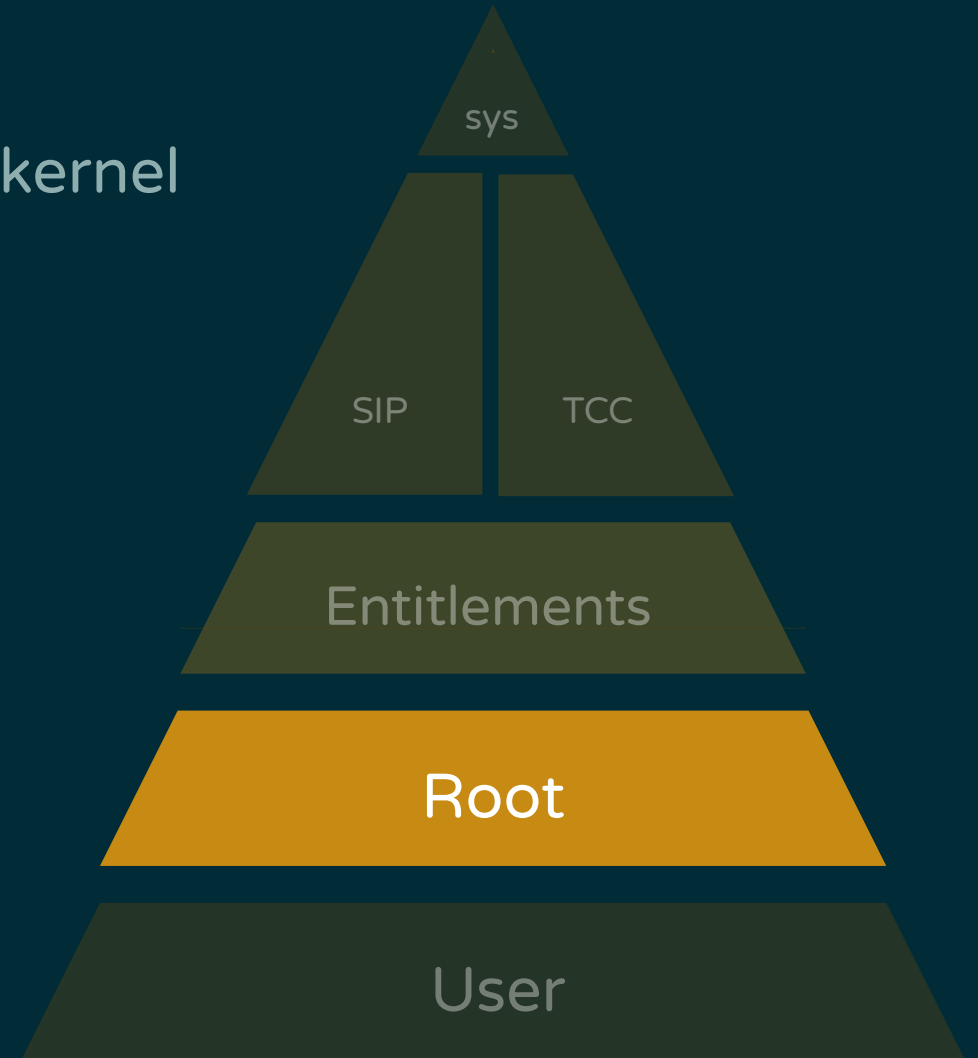
- ◆ Inject autorun
 - ◆ /Library/LaunchDaemon & LaunchAgent
- ◆ Hide the malicious file or folder

Under certain conditions, bypassing SIP or TCC needs root privilege



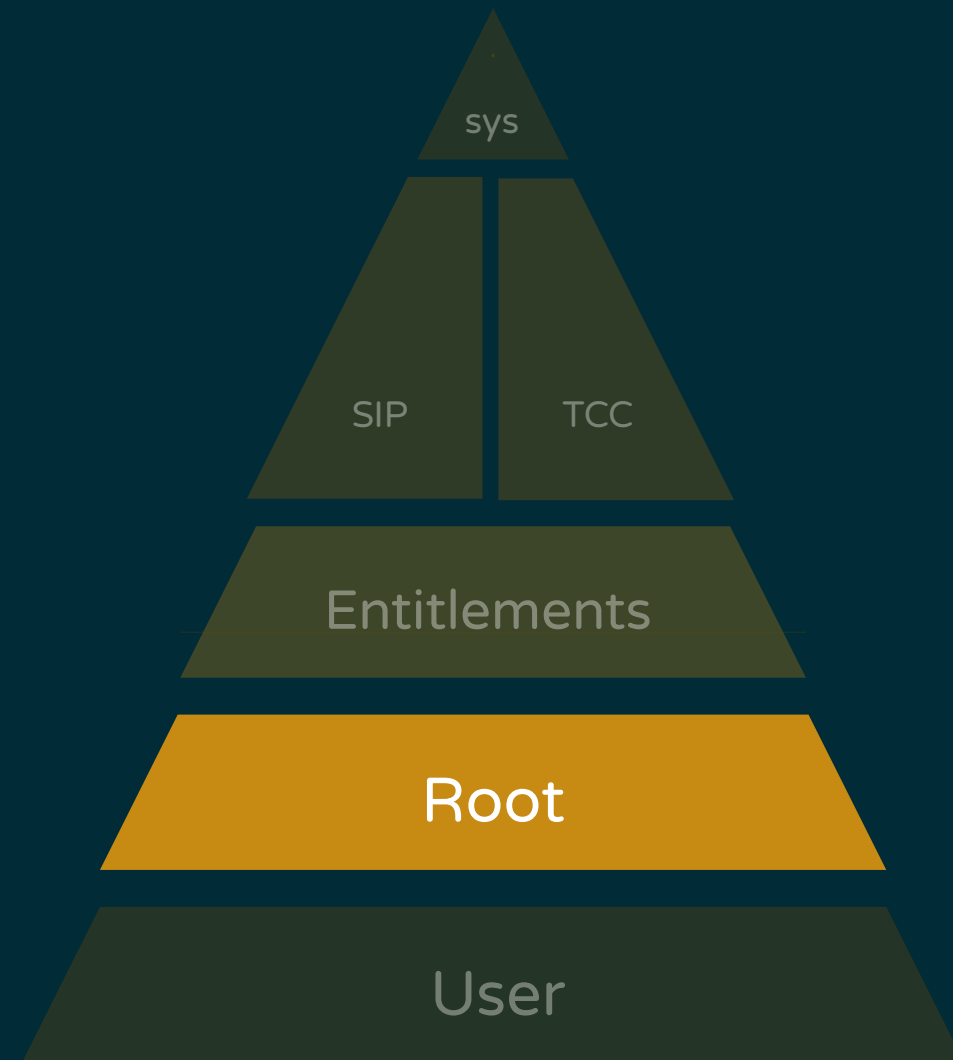
Four Processes Running as Root

- ◆ Launchd
 - ◆ Which is the first thing launched, after the kernel
- ◆ Helper daemon (a.k.a XPC Service)
- ◆ Auto-run process
- ◆ SUID binaries



Get Root

- ◆ Logic bug in system daemon
 - ✓ CVE-2022-22639 - SUHelper
 - ✓ CVE-2021-1815 - cfprefsd
- ◆ Authentication problem in third-party daemon
 - ✓ Microsoft OneDrive, Zoom, TeamViewer ...
- ◆ Code sign vulnerability
 - ✓ CVE-2022-26766 CoreTrust



Helper daemon (XPC Service)



- ◆ Launched by **Launchd** service
- ◆ Running in the background without requiring user input
- ◆ LaunchDaemons & LaunchAgents

```
will@Willde-MacBook-Air ~ launchctl list
PID      Status  Label
-        0       com.apple.SafariHistoryServiceAgent
704      0       com.apple.progressd
491      0       application.com.microsoft.Word.18098481.18156777
-        0       com.google.keystone.user.xpcservice
21590    0       com.apple.cloudphotod
516      0       com.apple.Finder
504      0       com.apple.homed
-        -9      com.apple.SafeEjectGPUAgent
63925    0       com.apple.quicklook
-        0       com.apple.parentalcontrols.check
526      0       com.apple.mediaremoteagent
471      0       com.apple.FontWorker
444      0       com.apple.bird
```

XPC Service

- ◆ XPC: APPLE low-level interprocess communication mechanism
- ◆ Privilege separation and stability
- ◆ Communicate via XPC



Client



1. Initiate an XPC message
(DownloadNewSystem())



Daemon

XPC Service

- ◆ Privilege separation and stability
- ◆ Communicate via XPC



Client



1. Initiate an XPC message
(DownloadNewSystem())

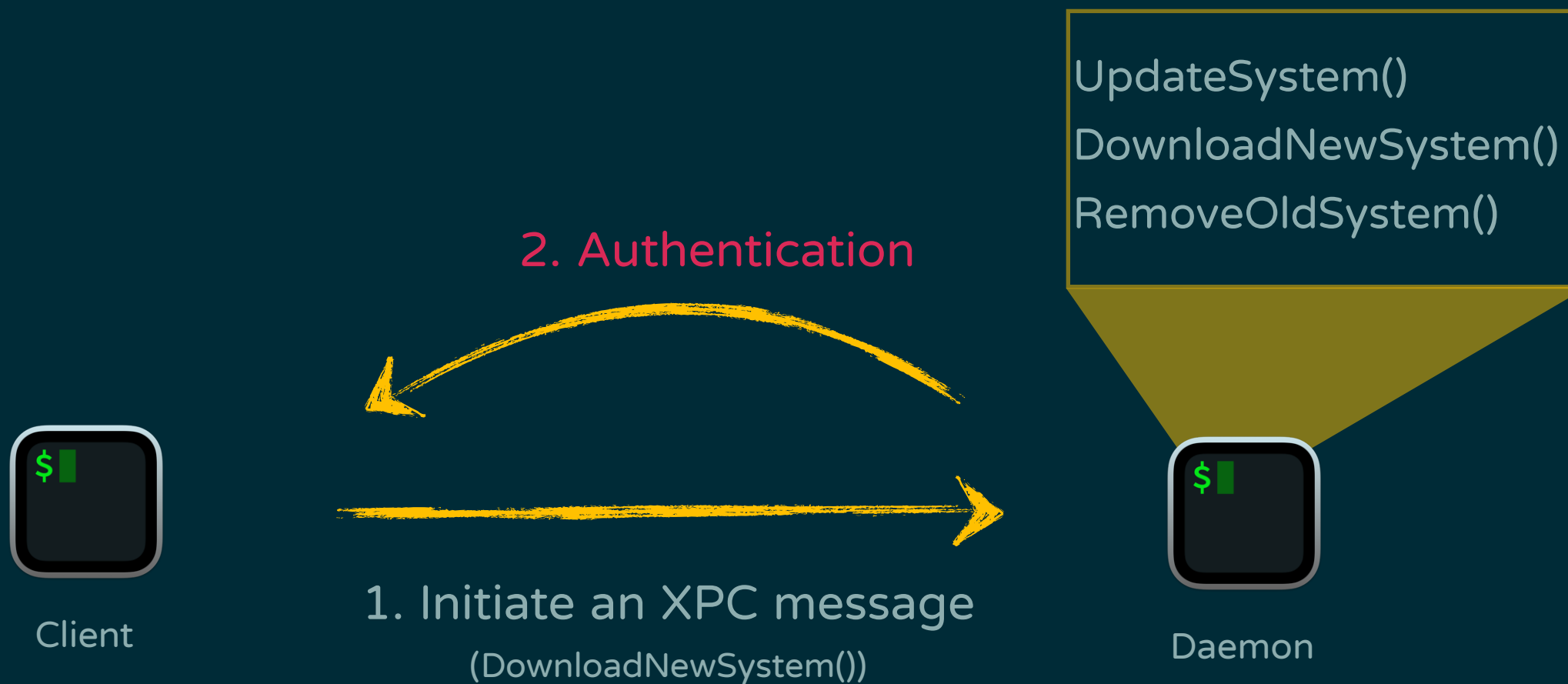
```
UpdateSystem()  
DownloadNewSystem()  
RemoveOldSystem()
```



Daemon

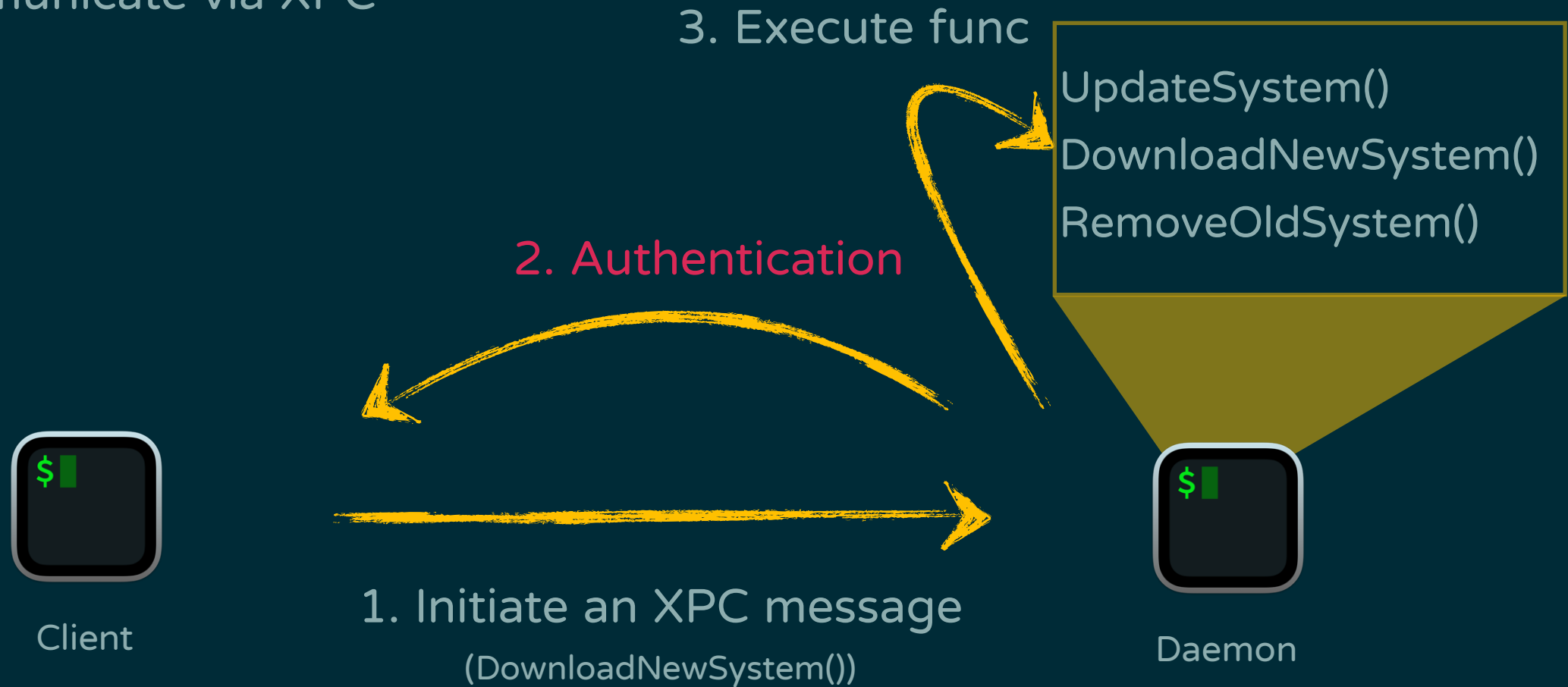
XPC Service

- ◆ Privilege separation and stability
- ◆ Communicate via XPC



XPC Service

- ◆ Privilege separation and stability
- ◆ Communicate via XPC



XPC Service

- ◆ How to check client is verified

2. Authentication



Client



Daemon

Privilege separation is secure



LPE using XPC service



CVE-2022-22639

suhelperd

- ◆ Prepare for the system upgrade
- ◆ ex: authorization, file moving

softwareupdated

- ◆ Responsible for updating system
- ◆ Periodically awaken and check for updates
- ◆ Can be triggered by manual update



Install macOS Monterey

SoftwareUpdate.framework

- ◆ MacOS provide the interfaces Install macOS Application need to communicate with suhelperd

```
Address      Type      Name
0x2a4e      P        +[SUHelperProxy sharedHelperProxy]
0x2a7a      P        __34+[SUHelperProxy sharedHelperProxy]_block_invoke
0x2af1      P        -[SUHelperProxy init]
0x2bce      P        -[SUHelperProxy setRecentRights:]
0x2bd8      P        -[SUHelperProxy authorizeWithEmptyAuthorizationForRights:]
0x2d84      P        -[SUHelperProxy helperQueueReentrantSync:]
0x2dc3      P        __58-[SUHelperProxy authorizeWithEmptyAuthorizationForRights:]_block_invoke
0x2e04      P        -[SUHelperProxy recentRights]
0x2e0e      P        -[SUHelperProxy authorizeTool:forRights:]
0x2f3a      P        -[SUHelperProxy isAuthorizedForRights:]
0x2fe7      P        __39-[SUHelperProxy isAuthorizedForRights:]_block_invoke
0x318d      P        __41-[SUHelperProxy authorizeTool:forRights:]_block_invoke
0xcba6      P        -[SUHelperProxy clearAnyUserPreference:]
0xcc43      P        __40-[SUHelperProxy clearAnyUserPreference:]_block_invoke
0xccaf      P        -[SUHelperProxy _isAuthorized]
0xcfd5      P        -[SUHelperProxy setObject:forAnyUserPreference:]
0xd076      P        __48-[SUHelperProxy setObject:forAnyUserPreference:]_block_invoke
0xf8e0      P        __41-[SUHelperProxy authorizeTool:forRights:]_block_invoke.58
0x1651e     P        -[SUHelperProxy makeQueues]
0x165b7     P        __27-[SUHelperProxy makeQueues]_block_invoke
0x1fd7b     P        -[SUHelperProxy readUpdatesIndex]
0x1fe28     P        __33-[SUHelperProxy readUpdatesIndex]_block_invoke
0x22f00     P        __41-[SUHelperProxy authorizeTool:forRights:]_block_invoke_2
0x22f0e     P        __41-[SUHelperProxy authorizeTool:forRights:]_block_invoke_3
0x4eb79     P        +[SUHelperProxy sharedHelperProxyIfAvailable]
0x4ebb1     P        -[SUHelperProxy dealloc]
```



Install macOS Monterey

Include



SoftwareUpdate.framework

XPC request



suhelperd

We can Access suhelperd! But...

- ◆ isClientPort: a rights authorization mechanism to verify the request is from a legitimate client
- ◆ No. 16 represents the required permissions



suhelperd

```
if ( Helper_10002F580 )
{
    v3 = (void *)objc_alloc_init(&OBJC_CLASS__NSAutoreleasePool);
    if ( (unsigned __int8)objc_msgSend(Helper_10002F580, "_isClientPort:validForRight:", a2, 16LL, v6) )
    {
        v4 = 0;
        *a3 = (unsigned __int64)objc_msgSend(Helper_10002F580, "prepareForLogoutAndInstall:", 0LL);
    }
    else
    {
        v4 = 8;
    }
    objc_msgSend(v3, "drain");
}
```

We can Access suhelperd! But...

- ◆ isClientPort: a rights authorization mechanism to verify the request is from a legitimate client
- ◆ No. 16 represents the required permissions

```
if ( (unsigned __int8)objc_msgSend(Helper_10002F580, "_isClientPort:validForRight:", a2, 16LL, v6) )  
{  
    v4 = 0;  
    *a3 = (unsigned __int64)objc_msgSend(Helper_10002F580, "prepareForLogoutAndInstall:", 0LL);  
}
```

```
if ( (unsigned __int8)objc_msgSend(qword_10002F528, "_isClientPort:validForRight:", a2, 8LL, v6) )  
{  
    *a3 = (char)objc_msgSend(qword_10002F528, "removeUpdatesAvailableCookie");  
    v4 = 0;  
}
```

```
if ( (unsigned __int8)objc_msgSend(qword_10002F528, "_isClientPort:validForRight:", a3, 1LL, v10) )  
{  
    v6 = qword_10002F528;  
    v7 = objc_msgSend(&OBJC_CLASS__NSString, "stringWithUTF8String:", a4);  
    *v4 = (char)objc_msgSend(v6, "moveInstalledPrintersToLibraryFromPath:", v7);  
    v8 = 0;
```


CVE-2022-22639

prepareInstallAssistantWithPath(path)

- ◆ The API **doesn't** use isClientPort to check permissions
- ◆ This API is wrapper of initiateMajorOSUpgradeAtPath
- ◆ initiateMajorOSUpgradeAtPath input the path from argument

```
char __cdecl -[SUHelper prepareInstallAssistantWithPath:](SUHelper *self, SEL a2, id path)
{
    id _path; // r14
    SUOSUMDMInstallClient *v4; // r15
    __int64 v5; // rax
    void **v7; // [rsp+8h] [rbp-48h]
    __int64 v8; // [rsp+10h] [rbp-40h]
    void *v9; // [rsp+18h] [rbp-38h]
    void *v10; // [rsp+20h] [rbp-30h]
    SUHelper *v11; // [rsp+28h] [rbp-28h]

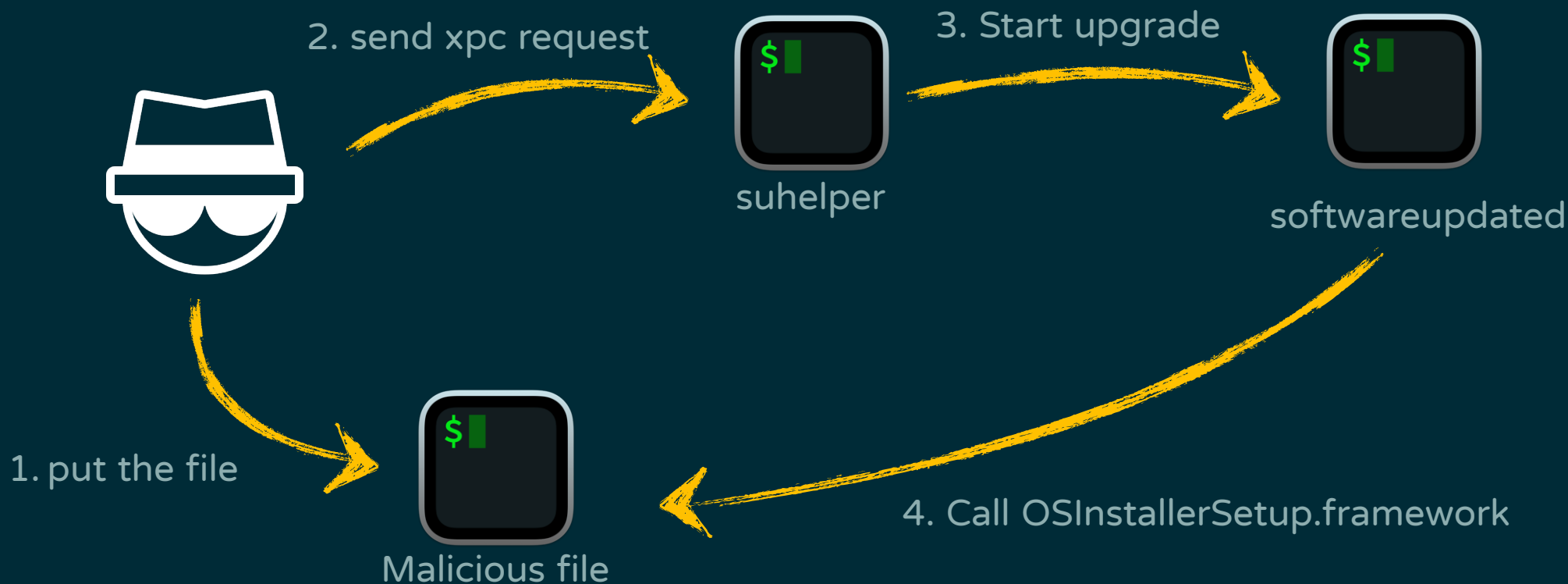
    _path = path;
    v4 = ((SUOSUMDMInstallClient *(__cdecl *) (SUHelper *, SEL))objc_msgSend)(self, "installClient");
    v5 = NSOpenStepRootDirectory((__int64)self, (__int64)"installClient");
    v7 = _NSConcreteStackBlock;
    v8 = 3254779904LL;
    v9 = &sub_10000C43D;
    v10 = &unk_100028440;
    v11 = self;
    objc_msgSend((void *)v4, "initiateMajorOSUpgradeAtPath:toVolume:forceRestart:withCompletion:", _path, v5, 0LL);
    return 1;
}
```

CVE-2022-22639

prepareInstallAssistantWithPath(path)

- ◆ initiateMajorOSUpgradeAtPath(path)

- ◆ Call (\$path)/Contents/Frameworks/OSInstallerSetup.framework



CVE-2021-1815 cfprefsd

- ◆ Responsible for setting preferences
- ◆ Two instances running on macOS
 - ◆ User and root privileges

XPC request

~/Library/Preferences/ZoomChat.plist



cfprefsd

```
will@hello:~/Desktop/projects|⇒ ls ~/Library/Preferences/  
ByHost  
ContextStoreAgent.plist  
Mach0View.plist  
MobileMeAccounts.plist  
Parallels  
UBF8T34669.OneDriveStandaloneSuite.plist  
VMware-Fusion  
ZoomChat.plist  
com.apple.AMPLibraryAgent.plist  
com.apple.ATS.plist
```

CVE-2021-1815 cfprefsd

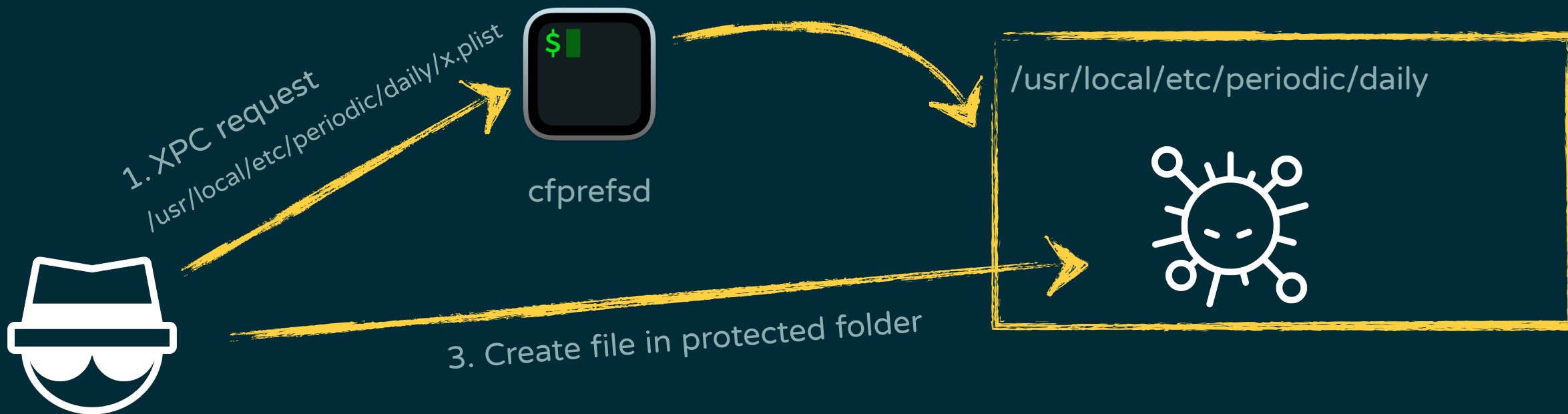
CFPrefsCreatePreferencesDirectory(path)

```
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;
}
```

cfprefsd Daemon

CFPrefsCreatePreferencesDirectory(path)

2. Create Directory with root privilege



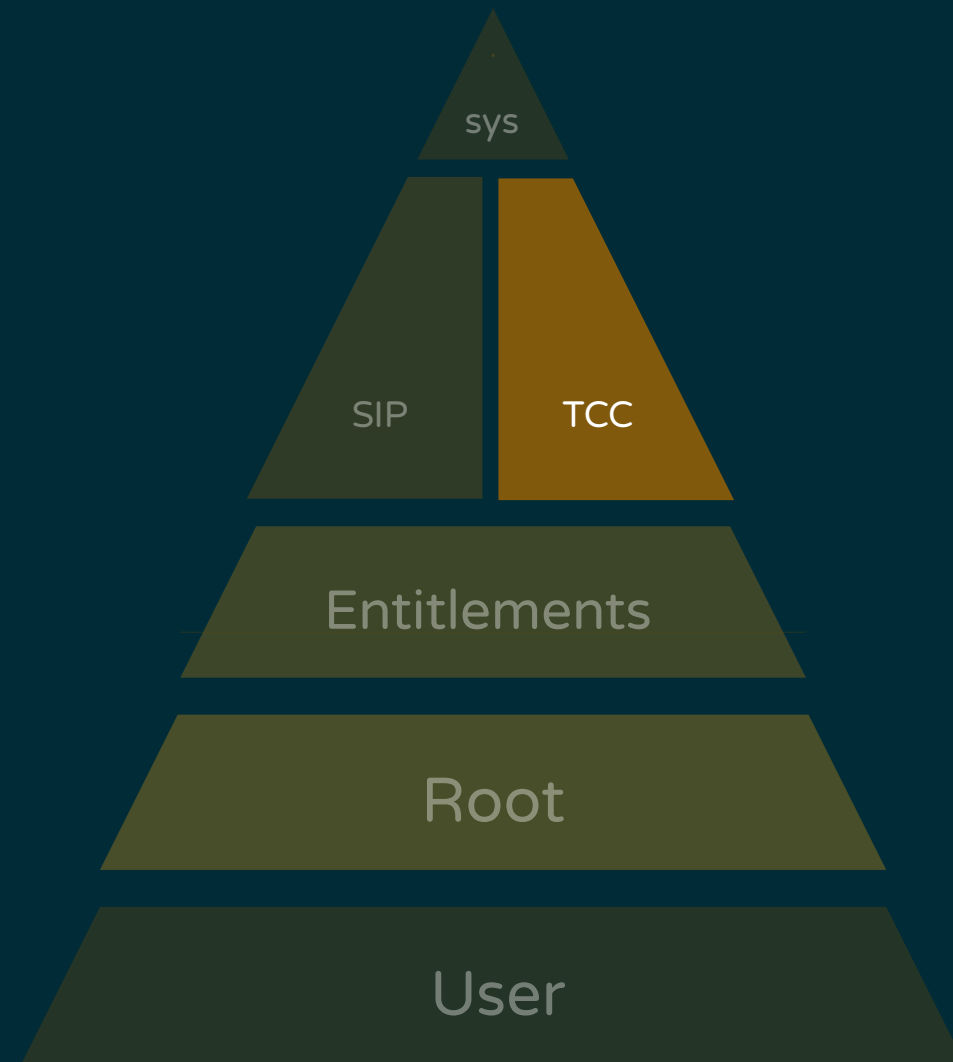
DazzleSpy Malware

- ◆ Privilege escalation to root (CVE-2021-30869)
- ◆ Use the exploitation to remove the quarantine attribute
- ◆ Avoid asking the user to confirm the launch of the unsigned executable file



Bypass TCC

- ◆ Attack with legacy apps
- ◆ Manipulate TCC.db
 - ✓ CVE-2021-1784
 - ✓ CVE-2020-9771
 - ✓ CVE-2020-9934
 - ✓ Powerdir
- ◆ Inject entitled application
 - ✓ CVE-2021-30713



Attacking With legacy apps

- ◆ Find the app with high TCC privileges
- ◆ Download the old version app
- ◆ Inject the evil library in old version app

csreq => anchor apple generic and certificate leaf[field.1.2.840.113635.100.6.1.9] /* exists */ or 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] = "43AQ936H96"

Same csreq



Manipulate TCC.db



- ◆ TCC.db file is protected
- ◆ We can change the filesystem by
 - ◆ Mount over “com.apple.TCC”(CVE-2020-9771, CVE-2021-1784)
 - ◆ Modify the \$HOME env to plant a chosen TCC.db (CVE-2020-9934)

Injecting Entitled Application

- ◆ Dylib inject
- ◆ Com.apple.security.get-task-allow

```
00qq#<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
  <key>com.apple.private.tcc.allow</key>
  <array>
    <string>kTCCServiceSystemPolicyRemovableVolumes</string>
  </array>
</dict>
```

CloudMensis Spyware

- ◆ Privilege escalation to bypass TCC (CVE-2020-9934)
- ◆ Gaining access to the screen sharing
- ◆ Being able to scan removable storage for documents of interest
- ◆ Being able to log keyboard events



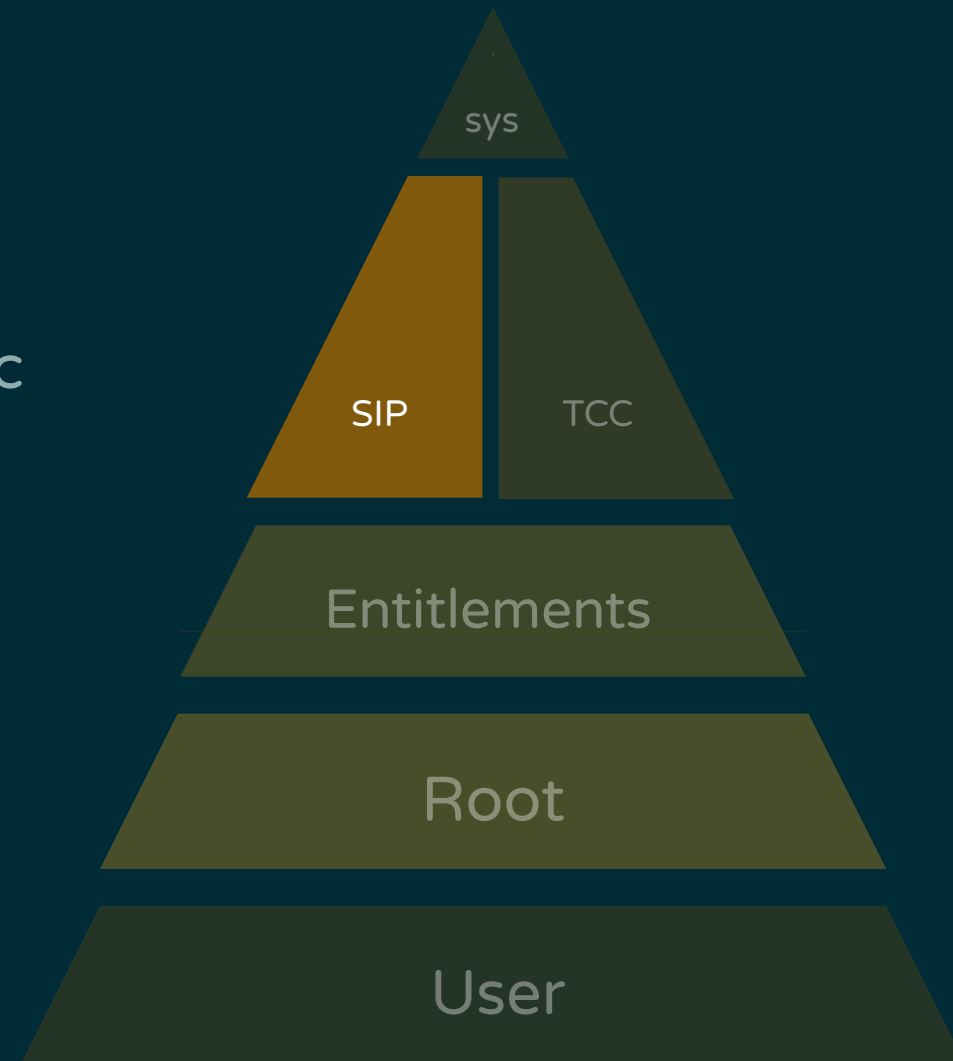
Case Study: IR Case

- ◆ We realize that user didn't know what "Full Disk Access" attribute means
- ◆ And give too many APPs this attribute
- ◆ It's **DANGEROUS!!**



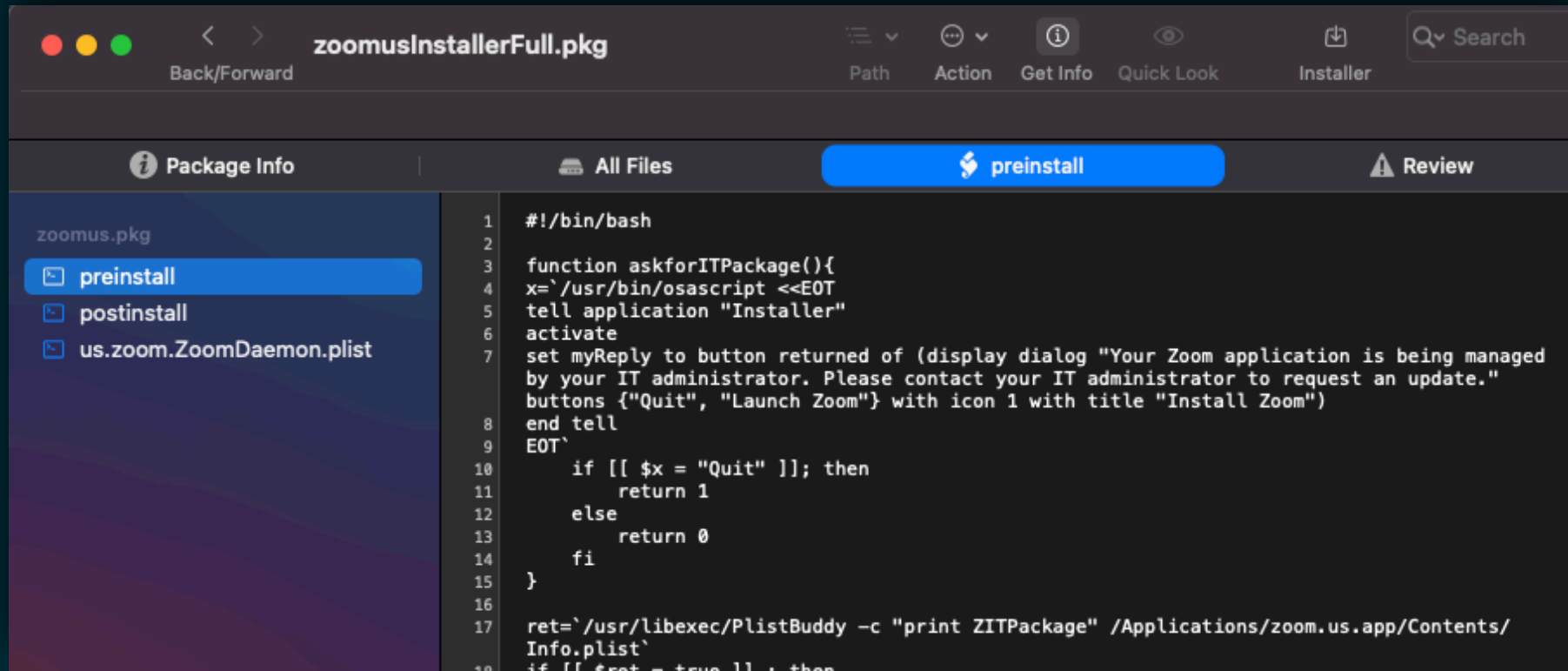
Bypass SIP

- ◆ Exploit entitled system daemon
 - ✓ CVE-2021-30892 system_installd
 - ✓ CVE-2022-22583 system_installd
 - ✓ CVE-2022-26712 SystemShoveService.xpc
- ◆ Inject entitled application
 - ✓ CVE-2022-22582 XAR

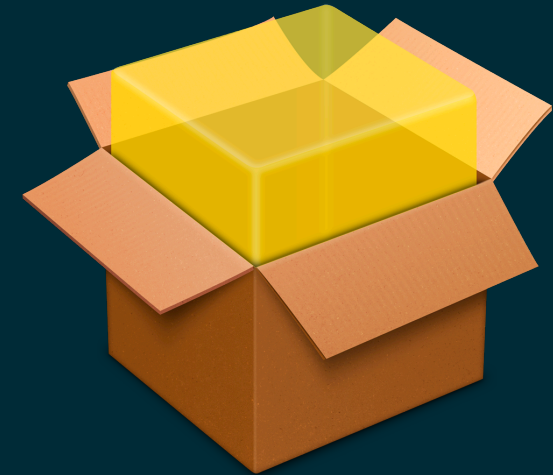


CVE-2022-22583 System_installd

- ◆ PackageKit framework's daemon
- ◆ Responsible for installing Apple-signed packages

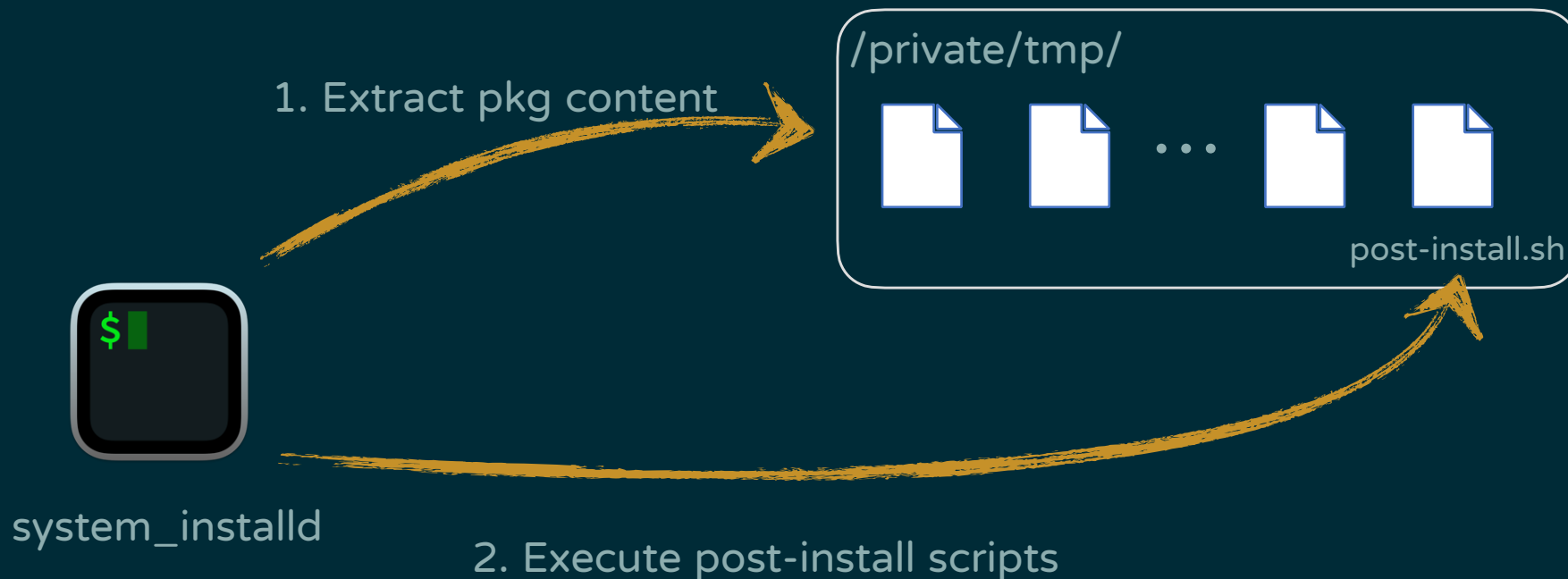


```
1  #!/bin/bash
2
3  function askforITPackage(){
4  x=`/usr/bin/osascript <<EOT
5  tell application "Installer"
6  activate
7  set myReply to button returned of (display dialog "Your Zoom application is being managed
8  by your IT administrator. Please contact your IT administrator to request an update."
9  buttons {"Quit", "Launch Zoom"} with icon 1 with title "Install Zoom")
10 end tell
11 EOT`
12 if [[ $x = "Quit" ]]; then
13     return 1
14 else
15     return 0
16 fi
17 }
18
19 ret=`/usr/libexec/PlistBuddy -c "print ZITPackage" /Applications/zoom.us.app/Contents/
20 Info.plist`
21 if [[ $ret = true ]]; then
```



CVE-2022-22583 System_Installd

- ◆ Daemon extracts all content to /private/tmp
- ◆ /private/tmp directory wasn't protected
- ◆ We can change the post-install scripts
- ◆ Then, system_installd execute post-install script with SIP privilege



Wrap Up

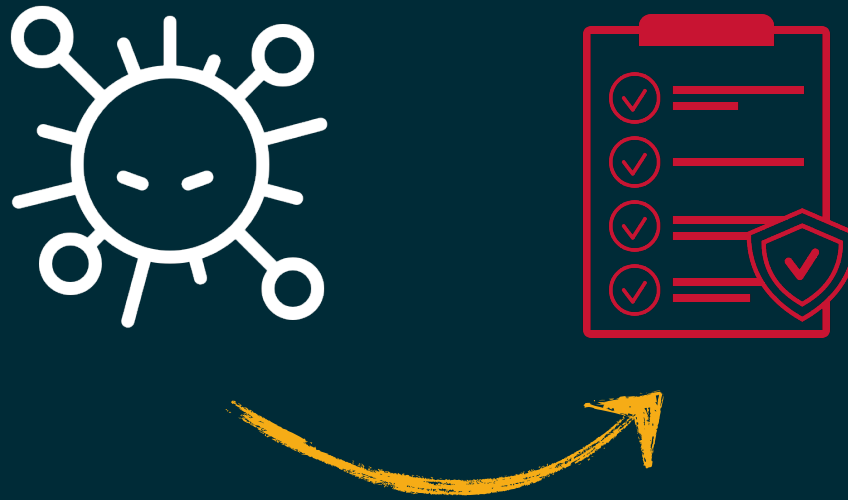
- ◆ Many exploits rely on vulnerable **XPC services**
- ◆ **TCC.db** and **legacy applications** are fairly common ways to attack TCC
- ◆ **Entitled** binaries and services should be tag

EndpointSecurity Framework



OnPrivilege

- ◆ We analyze recent privilege escalation exploits and set a trap
- ◆ We propose a new hunting module on macOS platform
- ◆ Hunting the threat when malware tries to do privilege escalation



Privilege Escalation

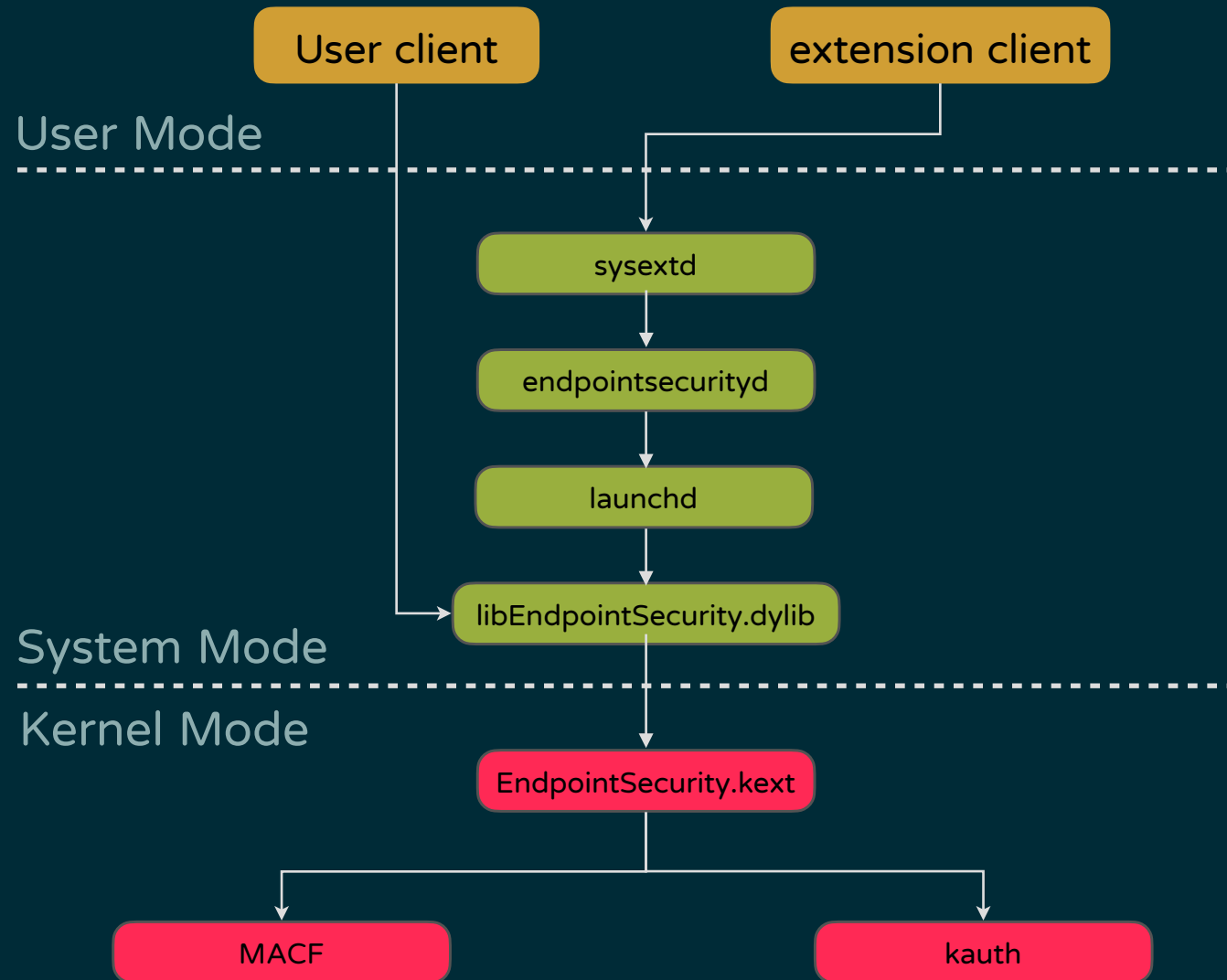
EndpointSecurity Framework



- ◆ Apple suggested third-party anti-malware software use **SEXT** instead of KEXT
- ◆ Apple supported ESF various of event to hook user behavior
- ◆ Third-party software should apply for ESF client entitlement from Apple
- ◆ Events include **process executions, mounting file systems, forking processes, and raising signals ...**

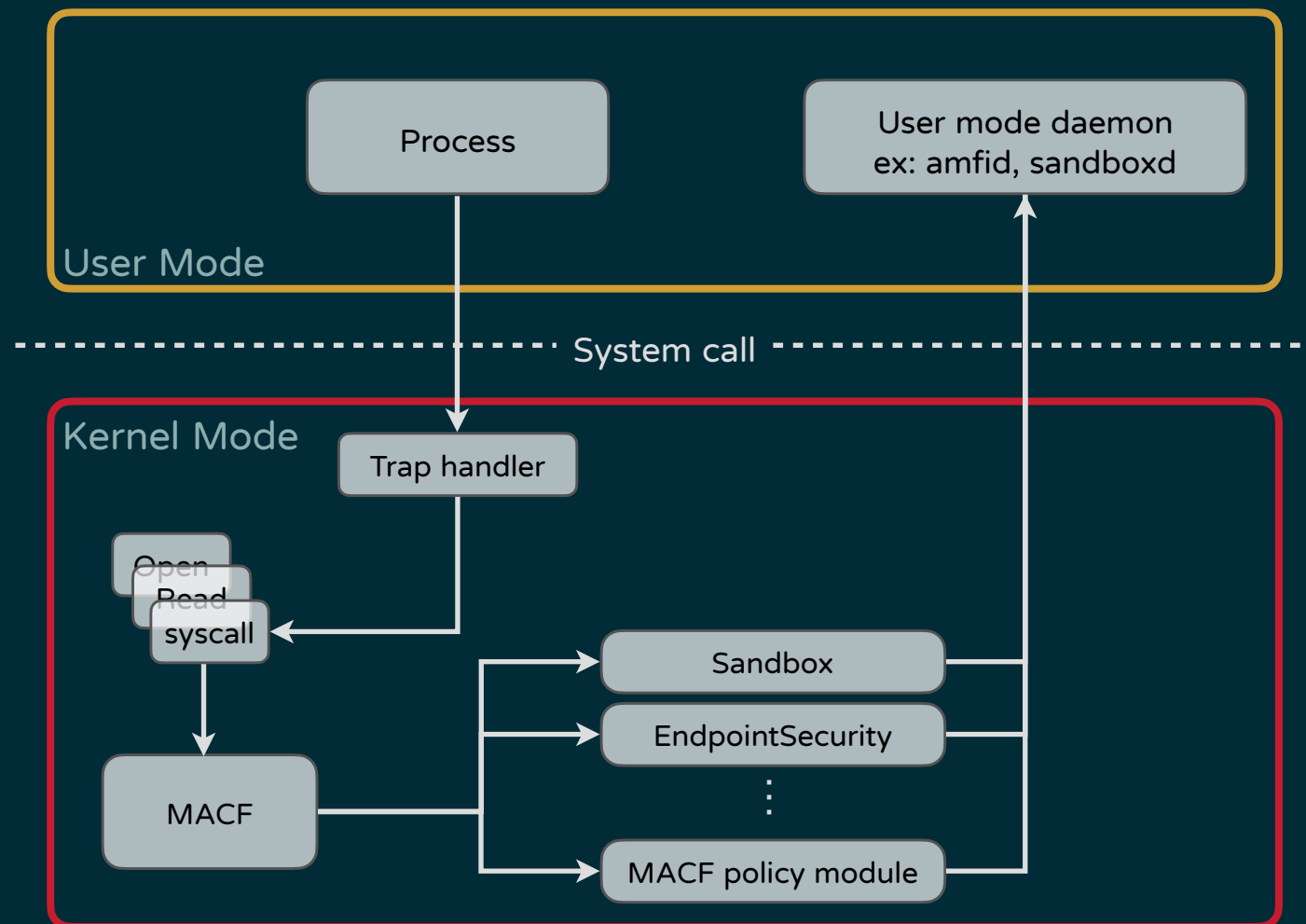
Diving deeper into ESF

- ◆ sysext
 - ◆ validate SEXT
- ◆ Endpointsecurityd
 - ◆ loading SEXT via launchd
- ◆ libEndpointSecurity.dylib
 - ◆ provide api function for client
- ◆ MACF, kauth
 - ◆ hooking syscall event



MACF

- ◆ Mandatory Access Control Framework
- ◆ Implemented in kernel
- ◆ A natural hooking mechanism
- ◆ MACF policy module:
 - ◆ Sandbox
 - ◆ AppleMobileFileIntegrity
 - ◆ Gatekeeper
 - ◆ EndpointSecurity



MACF

syscall: fork



mac_proc_check_for



MAC_CHECK

```
int
fork1(proc_t parent_proc, thread_t *child_threadp,
      int kind, coalition_t *coalitions)
{
... ..
#if CONFIG_MACF
    /*
     * Determine if MAC policies applied to the process will allow
     * it to fork. This is an advisory-only check.
     */
    err = mac_proc_check_fork(parent_proc);
    if (err != 0) {
        goto bad;
    }
#endif
... ..
}
```

MACF

syscall: fork



mac_proc_check_fork



MAC_CHECK

```
int
mac_proc_check_fork(proc_t curp)
{
    ... ..
    cred = kauth_cred_proc_ref(curp);
    MAC_CHECK(proc_check_fork, cred, curp);
    kauth_cred_unref(&cred);

    return error;
}
```

MACF

syscall: fork



mac_proc_check_for



MAC_CHECK

```
/*
 * MAC_CHECK performs the designated check by walking the policy
 * module list and checking with each as to how it feels about the
 * request. Note that it returns its value via 'error' in the scope
 * of the caller.
 */
#define MAC_CHECK(check, args...) do { \
    error = 0; \
    MAC_POLICY_ITERATE({ \
        if (mpc->mpc_ops->mpo_ ## check != NULL) { \
            DTRACE_MACF3(mac__call__ ## check, void *, mpc, int, \
error, int, MAC_ITERATE_CHECK); \
            int __step_err = mpc->mpc_ops->mpo_ ## check (args); \
            DTRACE_MACF2(mac__rslt__ ## check, void *, mpc, int, \
__step_err); \
            error = mac_error_select(__step_err, error); \
        } \
    }); \
} while (0)
```


Run program using shell

```
event: exec
  time: 2022-08-11 22:46:32.926979301
process:
  PID : 35802
  EUID : 501
  PPID : 1823
  GID : 35802
  SID : 1815
  TID : 1815
  RPID : 406
  Path : /bin/zsh
  ARGS : /tmp/exploit
  cs_flag : 0x22014801
```

xnu/osfmk/kern/cs_blobs.h

```
/* code signing attributes of a process */
#define CS_VALID 0x00000001 /* dynamically valid */
#define CS_ADHOC 0x00000002 /* ad hoc signed */
#define CS_GET_TASK_ALLOW 0x00000004 /* has get-task-allow entitlement */
#define CS_INSTALLER 0x00000008 /* has installer entitlement */

#define CS_FORCED_LV 0x00000010 /* Library Validation required */
#define CS_INVALID_ALLOWED 0x00000020 /* (macOS Only) Page Invalid Allowed */

#define CS_HARD 0x00000100 /* don't load invalid pages */
#define CS_KILL 0x00000200 /* kill process if it breaks */
#define CS_CHECK_EXPIRATION 0x00000400 /* force expiration check */
#define CS_RESTRICT 0x00000800 /* tell dyld to treat restricted */

#define CS_ENFORCEMENT 0x00001000 /* require enforcement */
#define CS_REQUIRE_LV 0x00002000 /* require library validation */
#define CS_ENTITLEMENTS_VALIDATED 0x00004000 /* code signature permitted */
#define CS_NVRAM_UNRESTRICTED 0x00008000 /* has com.apple.rootless */
```

CS_SIGNED | CS_RUNTIME | CS_ENTITLEMENTS_VALIDATED | CS_RESTRICT | CS_VALID

ES events



◆ > 100 Hooks can use in ESF

```
typedef enum {
    // The following events are available beginning in macOS 10.15
    ES_EVENT_TYPE_AUTH_EXEC
    , ES_EVENT_TYPE_AUTH_OPEN
    , ES_EVENT_TYPE_AUTH_KEXTLOAD
    , ES_EVENT_TYPE_AUTH_MMAP
    , ES_EVENT_TYPE_AUTH_MPROTECT
    , ES_EVENT_TYPE_AUTH_MOUNT
    , ES_EVENT_TYPE_AUTH_RENAME
    , ES_EVENT_TYPE_AUTH_SIGNAL
    , ES_EVENT_TYPE_AUTH_UNLINK
    , ES_EVENT_TYPE_NOTIFY_EXEC
    , ES_EVENT_TYPE_NOTIFY_OPEN
    , ES_EVENT_TYPE_NOTIFY_FORK
    , ES_EVENT_TYPE_NOTIFY_CLOSE
    , ES_EVENT_TYPE_NOTIFY_CREATE
    , ES_EVENT_TYPE_NOTIFY_EXCHANGEDATA
    , ES_EVENT_TYPE_NOTIFY_EXIT
    , ES_EVENT_TYPE_NOTIFY_GET_TASK
    , ES_EVENT_TYPE_NOTIFY_KEXTLOAD
    , ES_EVENT_TYPE_NOTIFY_KEXTUNLOAD
    , ES_EVENT_TYPE_NOTIFY_LINK
    , ES_EVENT_TYPE_NOTIFY_MMAP
    , ES_EVENT_TYPE_NOTIFY_MPROTECT
    , ES_EVENT_TYPE_NOTIFY_MOUNT
    , ES_EVENT_TYPE_NOTIFY_UNMOUNT
    , ES_EVENT_TYPE_NOTIFY_IOKIT_OPEN
    , ES_EVENT_TYPE_NOTIFY_RENAME
    , ES_EVENT_TYPE_NOTIFY_SETATTRLIST
    , ES_EVENT_TYPE_NOTIFY_SETEXTATTR
    , ES_EVENT_TYPE_NOTIFY_SETFLAGS
    , ES_EVENT_TYPE_NOTIFY_SETMODE
    , ES_EVENT_TYPE_NOTIFY_SETOWNER
    , ES_EVENT_TYPE_NOTIFY_SIGNAL
    , ES_EVENT_TYPE_NOTIFY_UNLINK
    , ES_EVENT_TYPE_NOTIFY_WRITE
    , ES_EVENT_TYPE_AUTH_FILE_PROVIDER_MATERIALIZE
    , ES_EVENT_TYPE_NOTIFY_FILE_PROVIDER_MATERIALIZE
    , ES_EVENT_TYPE_AUTH_FILE_PROVIDER_UPDATE
    , ES_EVENT_TYPE_NOTIFY_FILE_PROVIDER_UPDATE
    , ES_EVENT_TYPE_AUTH_READLINK
    , ES_EVENT_TYPE_NOTIFY_UIPC_CONNECT
    , ES_EVENT_TYPE_AUTH_UIPC_CONNECT
    , ES_EVENT_TYPE_AUTH_EXCHANGEDATA
    , ES_EVENT_TYPE_AUTH_SETACL
    , ES_EVENT_TYPE_NOTIFY_SETACL
    // The following events are available beginning in macOS 10.15.4
    , ES_EVENT_TYPE_NOTIFY_PTY_GRANT
    , ES_EVENT_TYPE_NOTIFY_PTY_CLOSE
    , ES_EVENT_TYPE_AUTH_PROC_CHECK
    , ES_EVENT_TYPE_NOTIFY_PROC_CHECK
    , ES_EVENT_TYPE_AUTH_GET_TASK
    // The following events are available beginning in macOS 11.0
    , ES_EVENT_TYPE_AUTH_SEARCHFS
    , ES_EVENT_TYPE_NOTIFY_SEARCHFS
    , ES_EVENT_TYPE_AUTH_FCNTL
    , ES_EVENT_TYPE_AUTH_IOKIT_OPEN
    , ES_EVENT_TYPE_AUTH_PROC_SUSPEND_RESUME
    , ES_EVENT_TYPE_NOTIFY_PROC_SUSPEND_RESUME
    , ES_EVENT_TYPE_NOTIFY_CS_INVALIDATED
    , ES_EVENT_TYPE_NOTIFY_GET_TASK_NAME
    , ES_EVENT_TYPE_NOTIFY_TRACE
    , ES_EVENT_TYPE_NOTIFY_REMOTE_THREAD_CREATE
    , ES_EVENT_TYPE_AUTH_REMOUNT
    , ES_EVENT_TYPE_NOTIFY_REMOUNT
    // The following events are available beginning in macOS 11.3
    , ES_EVENT_TYPE_AUTH_GET_TASK_READ
    , ES_EVENT_TYPE_NOTIFY_GET_TASK_READ
    , ES_EVENT_TYPE_NOTIFY_GET_TASK_INSPECT
    // The following events are available beginning in macOS 12.0
    , ES_EVENT_TYPE_NOTIFY_SETUID
    , ES_EVENT_TYPE_NOTIFY_SETGID
    , ES_EVENT_TYPE_NOTIFY_SETEUID
    , ES_EVENT_TYPE_NOTIFY_SETEGID
    , ES_EVENT_TYPE_NOTIFY_SETREUID
    , ES_EVENT_TYPE_NOTIFY_SETREGID
    , ES_EVENT_TYPE_AUTH_COPYFILE
    , ES_EVENT_TYPE_NOTIFY_COPYFILE
    // ES_EVENT_TYPE_LAST is not a valid event type but a convenient
    // value for operating on the range of defined event types.
    // This value may change between releases and was available
    // beginning in macOS 10.15
    , ES_EVENT_TYPE_LAST
} es_event_type_t;
```

ES events

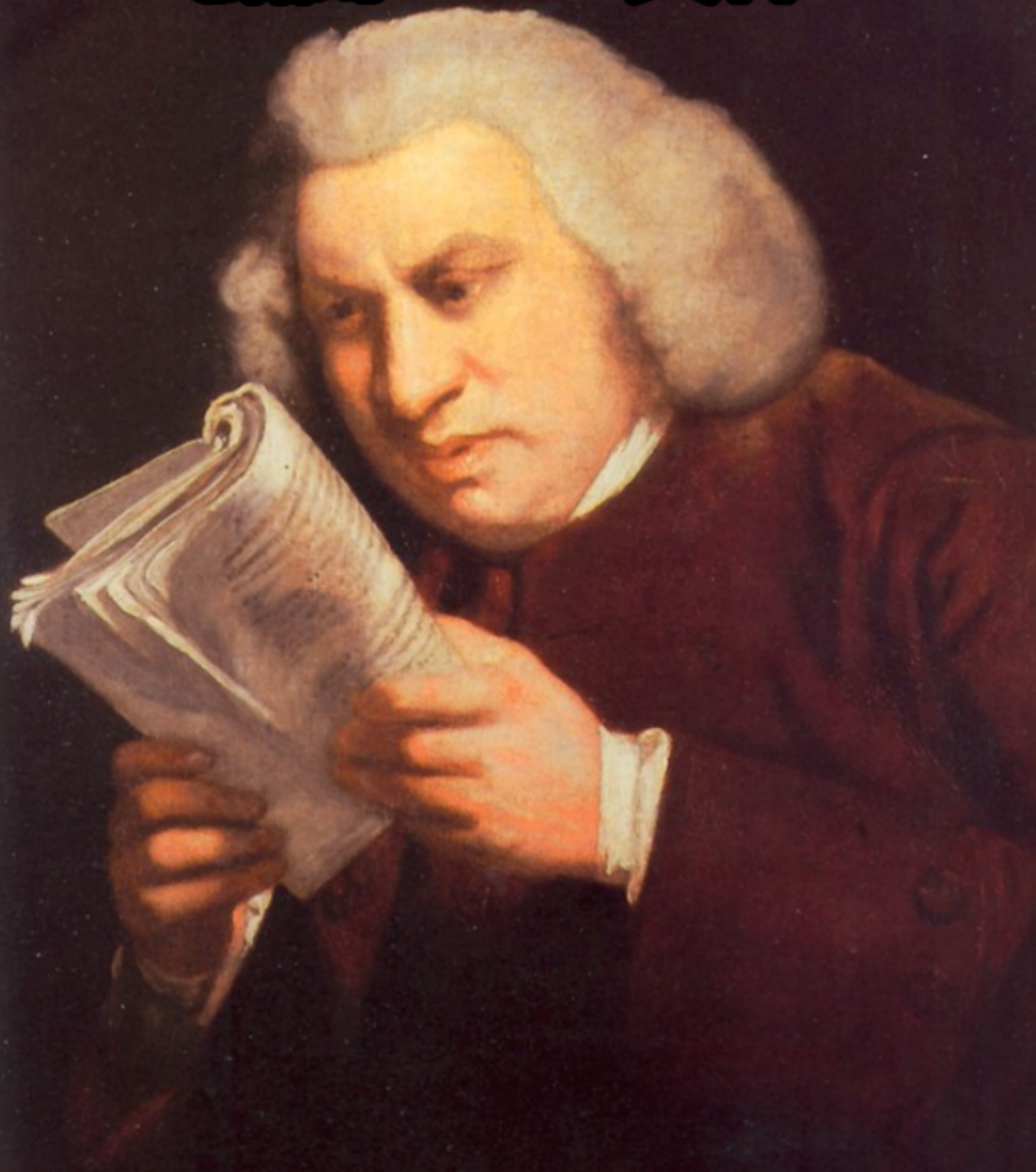


- ◆ > 100 Hooks can use in ESF
- ◆ How to select your hooking events ???

```
typedef enum {  
    // The following events are available beginning in macOS 10.15  
    ES_EVENT_TYPE_AUTH_EXEC  
    , ES_EVENT_TYPE_AUTH_OPEN  
    , ES_EVENT_TYPE_AUTH_KEXTLOAD  
    , ES_EVENT_TYPE_AUTH_MMAP  
    , ES_EVENT_TYPE_AUTH_MPROTECT  
    , ES_EVENT_TYPE_AUTH_MOUNT  
    , ES_EVENT_TYPE_AUTH_RENAME  
    , ES_EVENT_TYPE_AUTH_SIGNAL  
    , ES_EVENT_TYPE_AUTH_UNLINK  
    , ES_EVENT_TYPE_NOTIFY_EXEC  
    , ES_EVENT_TYPE_NOTIFY_OPEN  
    , ES_EVENT_TYPE_NOTIFY_FORK  
    , ES_EVENT_TYPE_NOTIFY_CLOSE  
    , ES_EVENT_TYPE_NOTIFY_CREATE  
    , ES_EVENT_TYPE_NOTIFY_EXCHANGEDATA  
    , ES_EVENT_TYPE_NOTIFY_EXIT  
    , ES_EVENT_TYPE_NOTIFY_GET_TASK  
    , ES_EVENT_TYPE_NOTIFY_KEXTLOAD  
    , ES_EVENT_TYPE_NOTIFY_KEXTUNLOAD  
    , ES_EVENT_TYPE_NOTIFY_LINK  
    , ES_EVENT_TYPE_NOTIFY_MMAP  
    , ES_EVENT_TYPE_NOTIFY_MPROTECT  
    , ES_EVENT_TYPE_NOTIFY_MOUNT  
    , ES_EVENT_TYPE_NOTIFY_UNMOUNT  
    , ES_EVENT_TYPE_NOTIFY_IOKIT_OPEN  
    , ES_EVENT_TYPE_NOTIFY_RENAME  
    , ES_EVENT_TYPE_NOTIFY_SETATTRLIST  
    , ES_EVENT_TYPE_NOTIFY_SETEXTATTR  
    , ES_EVENT_TYPE_NOTIFY_SETFLAGS  
    , ES_EVENT_TYPE_NOTIFY_SETMODE  
    , ES_EVENT_TYPE_NOTIFY_SETOWNER  
    , ES_EVENT_TYPE_NOTIFY_SIGNAL  
    , ES_EVENT_TYPE_NOTIFY_UNLINK  
    , ES_EVENT_TYPE_NOTIFY_WRITE  
    , ES_EVENT_TYPE_AUTH_FILE_PROVIDER_MATERIALIZE  
    , ES_EVENT_TYPE_NOTIFY_FILE_PROVIDER_MATERIALIZE  
    , ES_EVENT_TYPE_AUTH_FILE_PROVIDER_UPDATE  
    , ES_EVENT_TYPE_NOTIFY_FILE_PROVIDER_UPDATE  
    , ES_EVENT_TYPE_AUTH_READLINK  
    , ES_EVENT_TYPE_NOTIFY_UIPC_CONNECT  
    , ES_EVENT_TYPE_AUTH_UIPC_CONNECT  
    , ES_EVENT_TYPE_AUTH_EXCHANGEDATA  
    , ES_EVENT_TYPE_AUTH_SETACL  
    , ES_EVENT_TYPE_NOTIFY_SETACL  
    // The following events are available beginning in macOS 10.15.4  
    , ES_EVENT_TYPE_NOTIFY_PTY_GRANT  
    , ES_EVENT_TYPE_NOTIFY_PTY_CLOSE  
    , ES_EVENT_TYPE_AUTH_PROC_CHECK  
    , ES_EVENT_TYPE_NOTIFY_PROC_CHECK  
    , ES_EVENT_TYPE_AUTH_GET_TASK  
    // The following events are available beginning in macOS 11.0  
    , ES_EVENT_TYPE_AUTH_SEARCHFS  
    , ES_EVENT_TYPE_NOTIFY_SEARCHFS  
    , ES_EVENT_TYPE_AUTH_FCNTL  
    , ES_EVENT_TYPE_AUTH_IOKIT_OPEN  
    , ES_EVENT_TYPE_AUTH_PROC_SUSPEND_RESUME  
    , ES_EVENT_TYPE_NOTIFY_PROC_SUSPEND_RESUME  
    , ES_EVENT_TYPE_NOTIFY_CS_INVALIDATED  
    , ES_EVENT_TYPE_NOTIFY_GET_TASK_NAME  
    , ES_EVENT_TYPE_NOTIFY_TRACE  
    , ES_EVENT_TYPE_NOTIFY_REMOTE_THREAD_CREATE  
    , ES_EVENT_TYPE_AUTH_REMOUNT  
    , ES_EVENT_TYPE_NOTIFY_REMOUNT  
    // The following events are available beginning in macOS 11.3  
    , ES_EVENT_TYPE_AUTH_GET_TASK_READ  
    , ES_EVENT_TYPE_NOTIFY_GET_TASK_READ  
    , ES_EVENT_TYPE_NOTIFY_GET_TASK_INSPECT  
    // The following events are available beginning in macOS 12.0  
    , ES_EVENT_TYPE_NOTIFY_SETUID  
    , ES_EVENT_TYPE_NOTIFY_SETGID  
    , ES_EVENT_TYPE_NOTIFY_SETEUID  
    , ES_EVENT_TYPE_NOTIFY_SETEGID  
    , ES_EVENT_TYPE_NOTIFY_SETREUID  
    , ES_EVENT_TYPE_NOTIFY_SETREGID  
    , ES_EVENT_TYPE_AUTH_COPYFILE  
    , ES_EVENT_TYPE_NOTIFY_COPYFILE  
    // ES_EVENT_TYPE_LAST is not a valid event type but a convenient  
    // value for operating on the range of defined event types.  
    // This value may change between releases and was available  
    // beginning in macOS 10.15  
    , ES_EVENT_TYPE_LAST  
} es_event_type_t;
```

ES events

點開 ESF 文件



每個 event 描述只寫一句話

Triggering onPrivilege with ESF

OnPrivilege Project

- ◆ <https://github.com/will03/onPrivilege>
- ◆ Detect privilege changing behavior
- ◆ Including ...
 - ◆ Process rooting detection
 - ◆ TCC.db file protection
 - ◆ Legacy APP detection
 - ◆ Special entitlement file tracking



Hunt Process Rooting

- ◆ Enhance XPC client and service relationship
- ◆ Detect High privileged process execution
- ◆ Find the original Initiator



xpcproxy trampoline

- ◆ When client request, xpcproxy is responsible for executing daemon

event: `exec`

process:

```
PID : 48305
EUID : 0
PPID : 1
GID : 1
SID : 1
threadid : 1
RPID : 48305
path : /sbin/launchd
ARGS : xpcproxy com.apple.install.osinstallersetupd
cs_flag : 0x22014a21
```



Launchd



Xpcproxy



Daemon

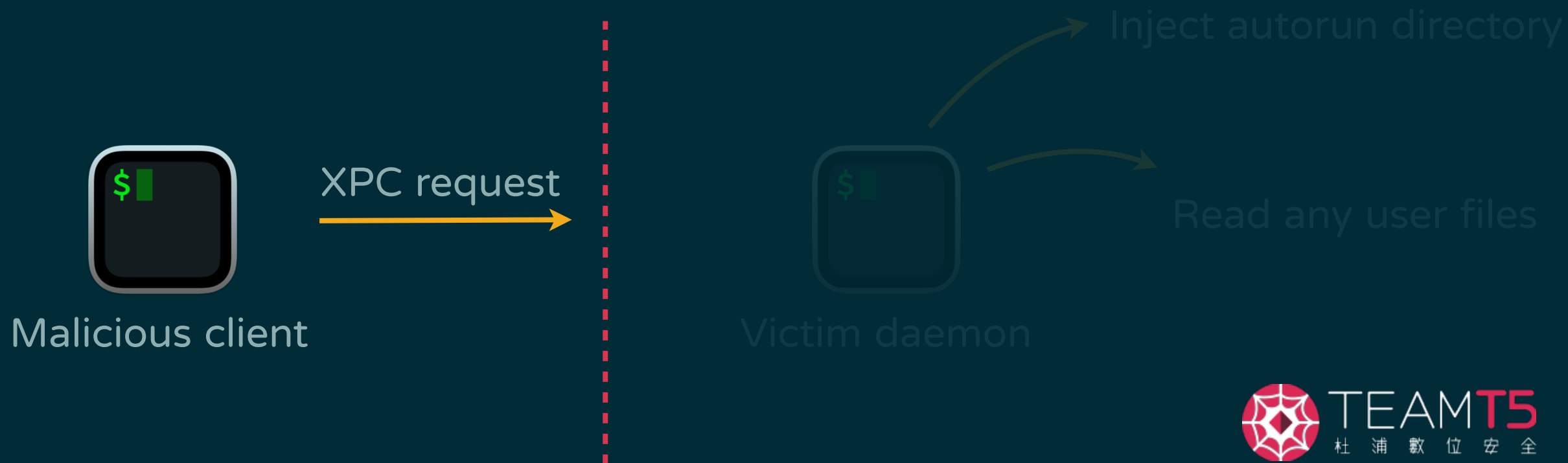
Tracking XPC

- ◆ When client using XPC to request daemon to do something
- ◆ There is no sign for connection between client and daemon
- ◆ It's an important issue when we design onPrivilege



Tracking XPC

- ◆ When client using XPC to request daemon to do something
- ◆ There is no sign for connection between client and daemon
- ◆ It's an important issue when we design onPrivilege



Tracking XPC

- ◆ When client using XPC to request daemon to do something
- ◆ There is no sign for connection between client and daemon
- ◆ It's an important issue when we design onPrivilege

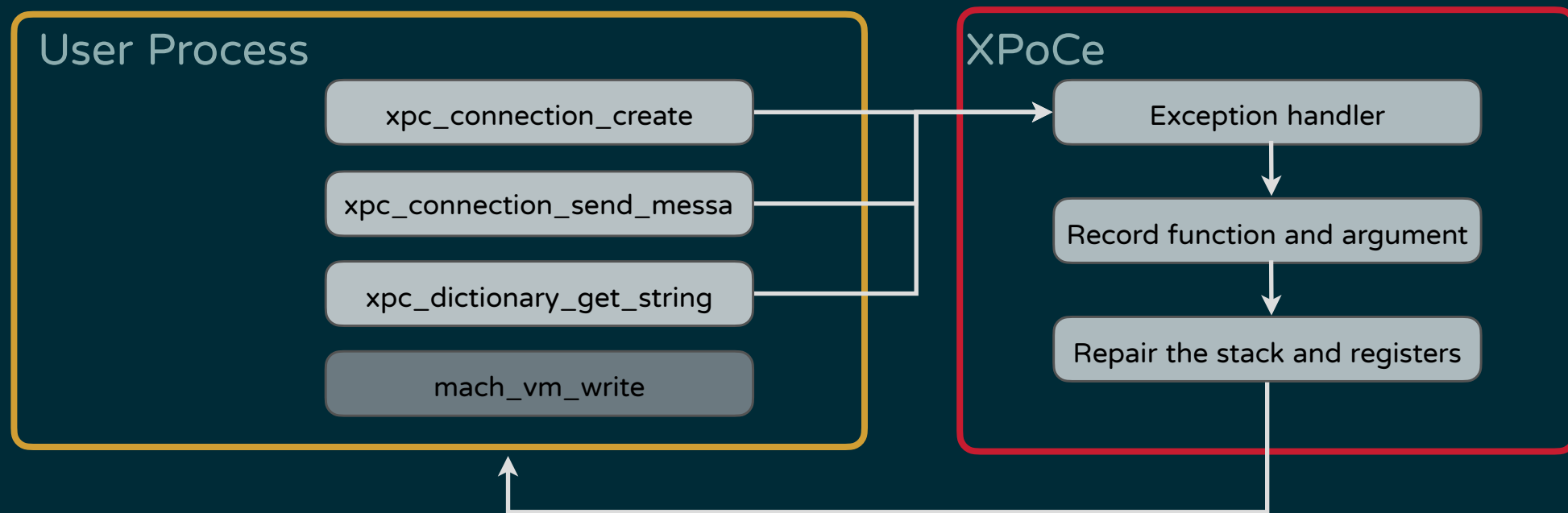


Tracking XPC

- ◆ **Hooking** XPC connection is the only solution for the moment
- ◆ What process can be hooked
 - ✓ No SIP-protected process
 - ✓ No entitlements process

Tracking XPC

- ◆ XPoCe - XPC Snooping utility
- ◆ <http://newosxbook.com/tools/XPoCe2.html>



Tracking XPC

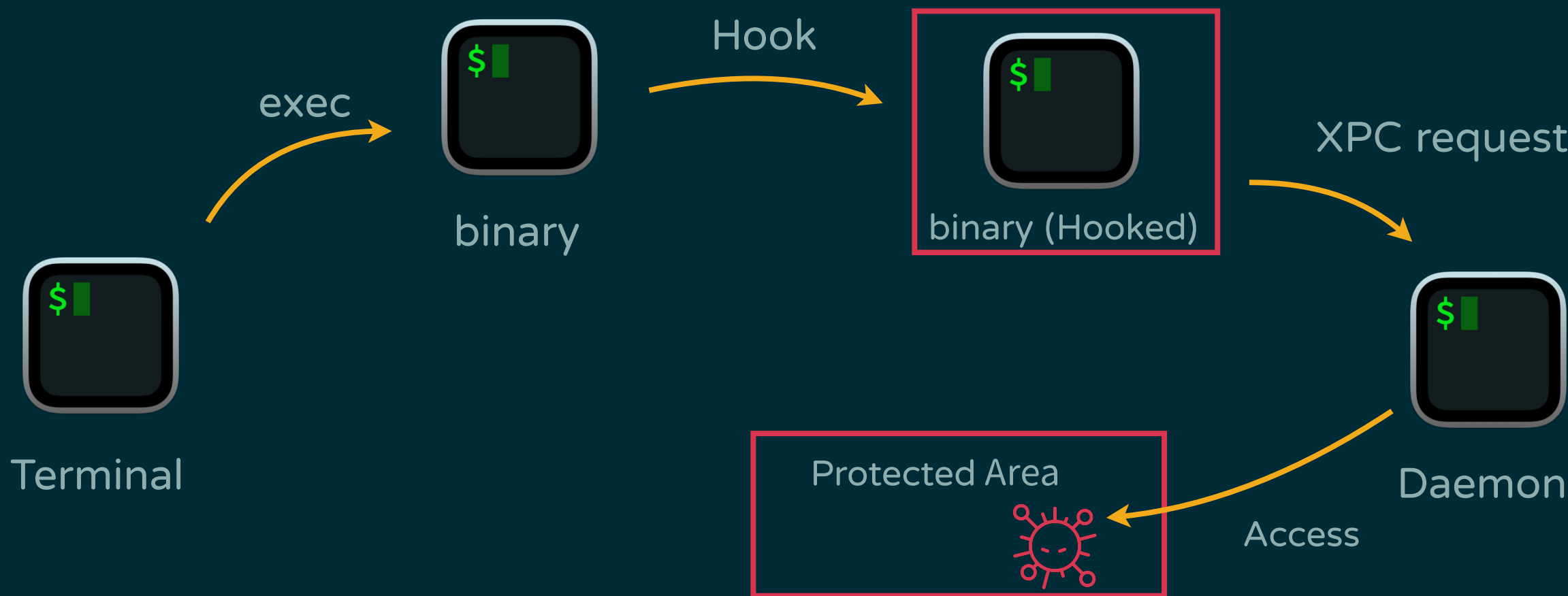
- ◆ Hooking the low privileged program



```
xpc_connection_create_mach_service ("com.apple.cfprefsd.daemon",0x0) = 0x7fd03ec08e10 ()
xpc_dictionary_get_uint64 ( dictionary@0x7fd03ec08a10,"req_pid")
= "<dictionary: 0x7fd03ec08a10> { count = 3, transaction: 0, voucher = 0x0, contents =
    "rec_execcnt" => <uint64: 0x723946d8c1665ff9>: 5011
    "req_pid" => <uint64: 0x723946d8c0da4ff9>: 2130
    "port" => <mach send right: 0x7fd03ec08360> { name = 10499, right = send, urefs = 1
```

Tracking XPC

- ◆ Hooking the low privileged program



Tracking XPC

- ◆ Connecting malicious client and behaviors



TCC Protection

- ◆ Old app detection
- ◆ Library inject detection
- ◆ Get-task-allow inject detection
- ◆ TCC.db protection
 - ◆ Mount/read/write

Hunting Duplicate APP bundles

◆ Record Application bundle version

```
will@hello:~|⇒ codesign -d -v -r- /tmp/Firefox.app
Executable=/private/tmp/Firefox.app/Contents/MacOS/firefox
Identifier=org.mozilla.firefox
Format=app bundle with Mach-0 thin (x86_64)
CodeDirectory v=20500 size=447 flags=0x10000(runtime) hashes=5+5
Signature size=9019
Timestamp=Jul 6, 2020 at 10:35:02 AM
Info.plist entries=27
TeamIdentifier=43AQ936H96
Runtime Version=10.11.0
Sealed Resources version=2 rules=13 files=92
designated => anchor apple generic and certificate leaf[field.1.2.840.113635.100.6.2.6] /* exists */ and certificate leaf[field.1.2.840.113635.100.6.2.6] /* exists */
```

```
will@hello:~|⇒ codesign -d -v -r- /Applications/Firefox.app
Executable=/Applications/Firefox.app/Contents/MacOS/firefox
Identifier=org.mozilla.firefox
Format=app bundle with Mach-0 universal (x86_64 arm64)
CodeDirectory v=20500 size=479 flags=0x10000(runtime) hashes=6+5
Signature size=8989
Timestamp=Jun 23, 2022 at 4:45:24 PM
Info.plist entries=25
TeamIdentifier=43AQ936H96
Runtime Version=11.0.0
Sealed Resources version=2 rules=13 files=82
designated => anchor apple generic and certificate leaf[field.1.2.840.113635.100.6.2.6] /* exists */ and certificate leaf[field.1.2.840.113635.100.6.2.6] /* exists */
```



Hunt Process Rooting

- ◆ Hook XPC connection when low privileged program execute
- ◆ Check if program or service try to write high privileged area

```
std::map<string, es_event_type_t> NOTIFYEVENTS = {  
    {"exec", ES_EVENT_TYPE_NOTIFY_EXEC},  
    {"write", ES_EVENT_TYPE_NOTIFY_WRITE},  
    {"create", ES_EVENT_TYPE_NOTIFY_CREATE}  
};
```

Hunting TCC-bypass

- ◆ Detect get-task and library injection
- ◆ Check the mount point is covering TCC database directory
- ◆ Monitor duplicate APP bundles

```
std::map<string, es_event_type_t> NOTIFYEVENTS = {  
    {"exec", ES_EVENT_TYPE_NOTIFY_EXEC},  
    {"mount", ES_EVENT_TYPE_NOTIFY_MOUNT},  
    {"write", ES_EVENT_TYPE_NOTIFY_WRITE},  
    {"get_task", ES_EVENT_TYPE_NOTIFY_GET_TASK},  
    {"get_task_name", ES_EVENT_TYPE_NOTIFY_GET_TASK_READ},  
    {"get_task_inspect", ES_EVENT_TYPE_NOTIFY_GET_TASK_INSPECT}  
};
```

Hunting SIP-Bypass

- ◆ Check get task and library injection
- ◆ Hunting user program trying to connect SIP privilege service

```
std::map<string, es_event_type_t> NOTIFYEVENTS = {  
    {"exec", ES_EVENT_TYPE_NOTIFY_EXEC},  
    {"get_task", ES_EVENT_TYPE_NOTIFY_GET_TASK},  
    {"get_task_name", ES_EVENT_TYPE_NOTIFY_GET_TASK_READ},  
    {"get_task_inspect", ES_EVENT_TYPE_NOTIFY_GET_TASK_INSPECT}  
};
```

DEMO

Notify mode

- ◆ Hunting CVE-2021-1815 (cfprefsd)
- ◆ Pupop the alert message and generate raw report for user

Blocking mode

- ◆ Hunting CVE-2022-22639 (suhelperd)
- ◆ Reject the Terminal open with root

Apple logo | Finder | File | Edit | View | Go | Window | Help

Desktop — -zsh — 68x34

```
user@users-Mac Desktop %
```

src — -zsh — 86x32

```
user@users-Mac src %
```

report

< > [Grid] [List] [Compare] [Web] [Share] [Settings] [Share] [Search]

Name	Date Modified	Size	Kind
------	---------------	------	------

Favorites

- Recents
- Applicatio...
- Desktop
- Documents
- Downloads

Locations

- Macintos...

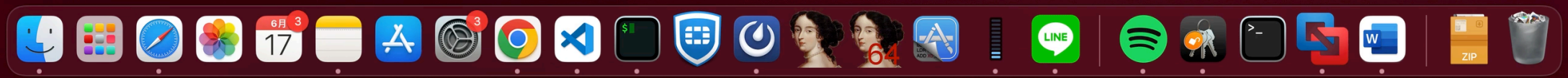
Tags

- Red
- Orange
- Yellow
- Green
- Blue



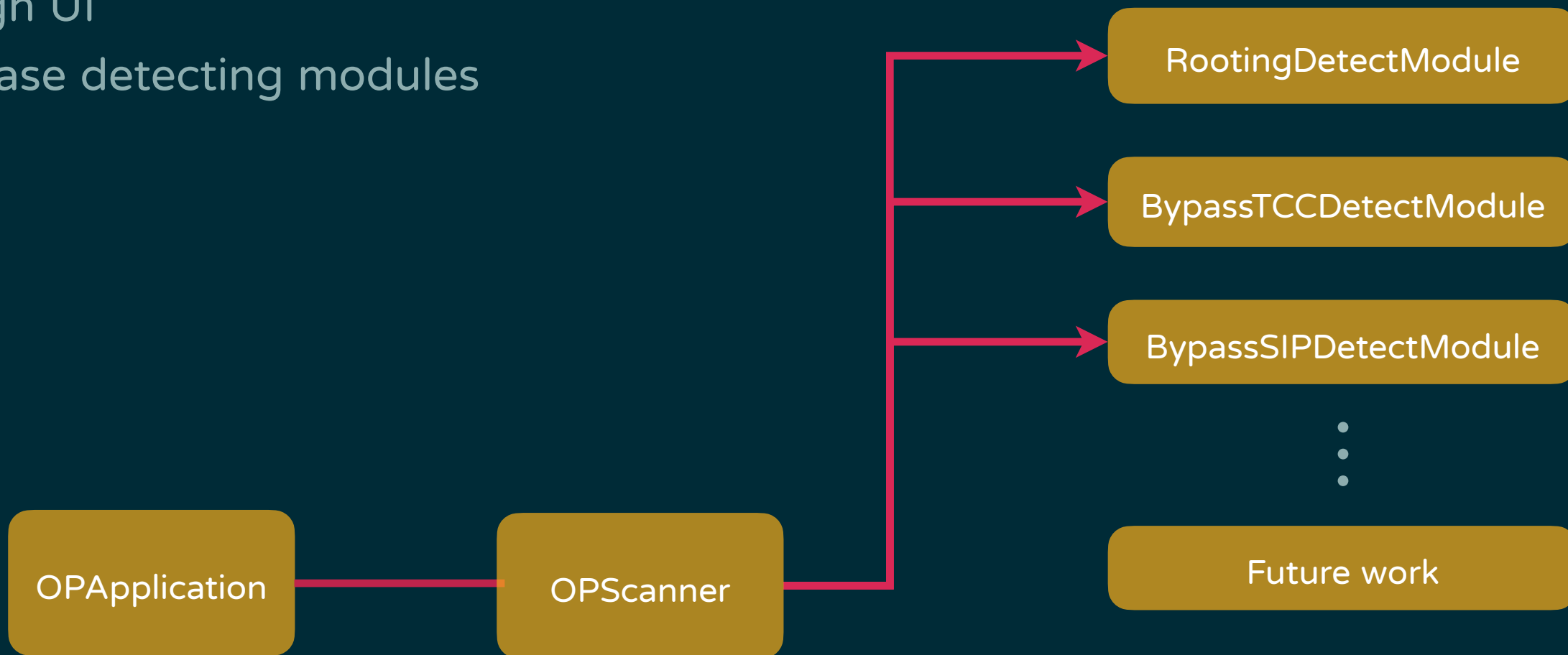
```
will@hello:~/Desktop/research/rootless/suhelperd_daemon/CVE-2022-22639 |  
main$ => █
```

```
ilege  
will@hello:~/Desktop/research/OnPrivilege/Sanjar/src|main$ => sudo ./OnPrivilege
```



Future work

- ◆ Design UI
- ◆ Increase detecting modules



Related Topics

- ◆ Leveraging the Apple ESF for Behavioral Detections - Black hat USA 22
- ◆ Mitigating exploits using Apple's Endpoint Security - VB2021
- ◆ Endpoint Security and Insecurity - OBTS 2020

Conclusion

- ◆ Only rely on ESF is **not enough** when encounter advanced attack
- ◆ Developer must add additional heuristic detection logic
- ◆ Spyware uses various **1-day exploits** to privilege escape
- ◆ Don't give terminal too many privilege (ex: **full disk access**)
- ◆ macOS 13 released new helpful **ESF events**
- ◆ Apple propose new binary: **ESFlogger**
- ◆ ESF become more important in Apple system
- ◆ Look forward to **Lockdown** mode

Reference

- ◆ https://www.trendmicro.com/en_us/research/22/d/macOS-suhelper-root-privilege-escalation-vulnerability-a-deep-di.html
- ◆ <https://www.offensive-security.com/offsec/macOS-preferences-priv-escalation/>
- ◆ <https://jhftss.github.io/CVE-2022-26712-The-POC-For-SIP-Bypass-Is-Even-Tweetable/>
- ◆ <http://newosxbook.com/tools/XPoCe.html>
- ◆ <http://newosxbook.com/articles/jlaunchctl.html>

THANK YOU!

Will

will@teamt5.org

