

Reverse Engineering Apple Security Updates

John McIntosh

CL | CLEARSEC LABS



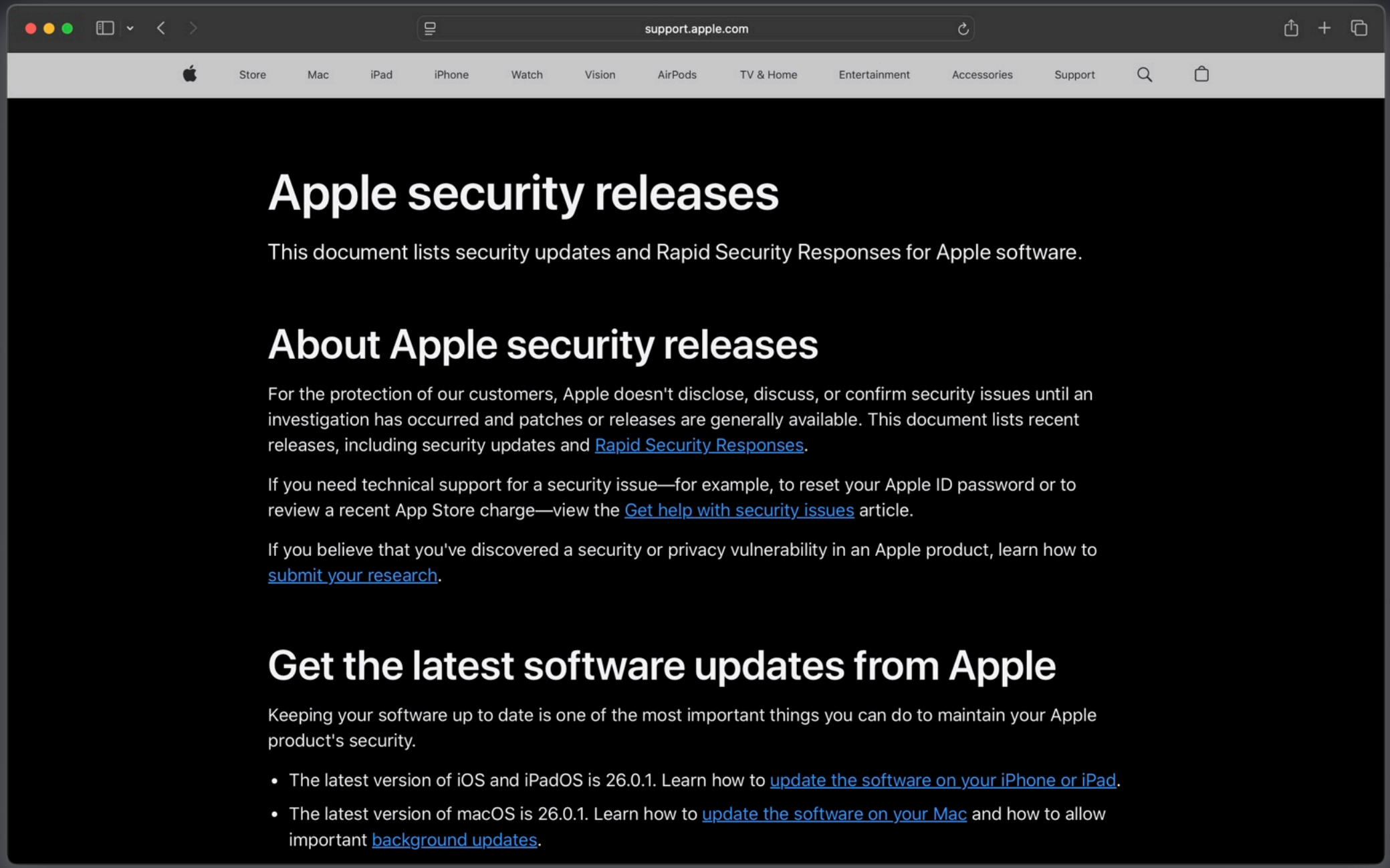
whoami

- @clearbluejar
- security researcher
- founder @clearseclabs
- open-source dev
- speaker / instructor
- blogger

Overview

- Reverse Engineering Apple Security Updates
- Show the manual baseline
- Introduce the hybrid philosophy (Agentic Diffing)
- Walk through the toolchain and methodology
- Prove it live (demo + case studies)
- Impact and roadmap


Reverse Engineering Apple
Security Updates



Apple security releases

This document lists security updates and Rapid Security Responses for Apple software.

About Apple security releases

For the protection of our customers, Apple doesn't disclose, discuss, or confirm security issues until an investigation has occurred and patches or releases are generally available. This document lists recent releases, including security updates and Rapid Security Responses.

If you need technical support for a security issue—for example, to reset your Apple ID password or to review a recent App Store charge—view the Get help with security issues article.

If you believe that you've discovered a security or privacy vulnerability in an Apple product, learn how to submit your research.

Get the latest software updates from Apple

Keeping your software up to date is one of the most important things you can do to maintain your Apple product's security.

- The latest version of iOS and iPadOS is 26.0.1. Learn how to update the software on your iPhone or iPad.
- The latest version of macOS is 26.0.1. Learn how to update the software on your Mac and how to allow important background updates.

support.apple.com

About the security content of iOS 26 and iPadOS 26

This document describes the security content of iOS 26 and iPadOS 26.

About Apple security updates

For our customers' protection, Apple doesn't disclose, discuss, or confirm security issues until an investigation has occurred and patches or releases are available. Recent releases are listed on the Apple security releases page.

Apple security documents reference vulnerabilities by CVE-ID when possible.

For more information about security, see the Apple Product Security page.

iOS 26 and iPadOS 26

Released September 15, 2025

Apple Neural Engine

Available for: iPhone 11 and later, iPad Pro 12.9-inch 3rd generation and later, iPad Pro 11-inch 1st generation and later, iPad Air 3rd generation and later, iPad 8th generation and later, and iPad mini 5th generation and later

Impact: An app may be able to cause unexpected system termination

Description: An out-of-bounds access issue was addressed with improved bounds checking.

CVE-2025-43344: an anonymous researcher

AppleMobileFileIntegrity

Description: A permissions issue was addressed with additional restrictions.

CVE-2025-43317: Mickey Jin (@patch1t)

Audio

Available for: iPhone 11 and later, iPad Pro 12.9-inch 3rd generation and later, iPad Pro 11-inch 1st generation and later, iPad Air 3rd generation and later, iPad 8th generation and later, and iPad mini 5th generation and later

Impact: Processing a maliciously crafted media file may lead to unexpected app termination or corrupt process memory

Description: An out-of-bounds access issue was addressed with improved bounds checking.

CVE-2025-43346: Hossein Lotfi (@hosselot) of Trend Micro Zero Day Initiative

Bluetooth

Available for: iPhone 11 and later, iPad Pro 12.9-inch 3rd generation and later, iPad Pro 11-inch 1st generation and later, iPad Air 3rd generation and later, iPad 8th generation and later, and iPad mini 5th generation and later

IOKit

Available for: iPhone 11 and later, iPad Pro 12.9-inch 3rd generation and later, iPad Pro 11-inch 1st generation and later, iPad Air 3rd generation and later, iPad 8th generation and later, and iPad mini 5th generation and later

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

Description: An authorization issue was addressed with improved state management.

CVE-2025-31255: Csaba Fitzl (@theevilbit) of Kandji

Immediate Questions

- Which binary was affected?
- What authorization check was added?
- Could a sandboxed app reach the code path?
- What internal state management changes were made?
- Which functions or methods were modified in the patched binary?

The Issue

- Apple’s IPSW-based updates are opaque
- Apple publishes **high-level CVE advisories**, but not the actual code changes
- IPSW firmware updates contain **thousands of binaries**

The Solution

- Reverse Apple Security Updates
- Extract and diff binaries to find **real binary changes**
- Map vague CVEs to binaries and **concrete mitigations**
- Find "Binary Truth"

Background



Patch Different *OS

John McIntosh

CL | CLEARSEC LABS

High Level Patch Diffing on *OS

1. Identify a CVE of interest
2. Download corresponding IPSW update and N-1
3. Determine changes for update
4. Map binaries to CVE
5. Extract the related file(s)
6. Diff the binaries
7. Root cause the vulnerability

Identify CVE of interest

Apple Security Update - iOS / iPadOS 16.5

https://support.apple.com/en-us/HT213757

Telephony

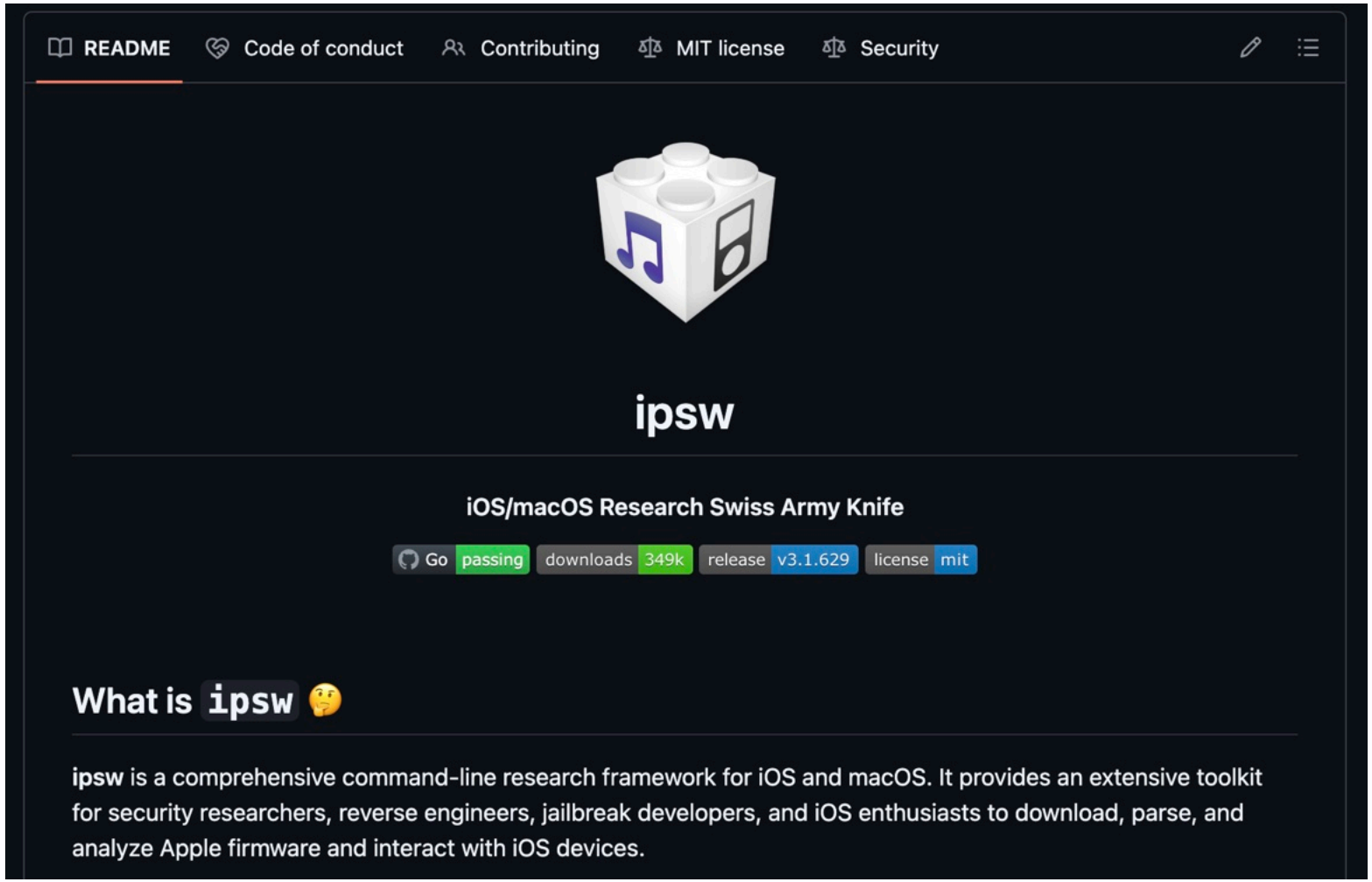
Available for: iPhone 8 and later, iPad Pro (all models), iPad Air 3rd generation and later, iPad 5th generation and later, and iPad mini 5th generation and later

Impact: A remote attacker may be able to cause unexpected app termination or arbitrary code execution

Description: A use-after-free issue was addressed with improved memory management.

CVE-2023-32412: Ivan Fratric of Google Project Zero

Download a corresponding IPSW and N-1



https://github.com/blacktop/ipsw

- 16.5 (Patched)
  - `ipsw download ipsw --device iPhone14,7 -b 20F66`
- 16.4.1 (N-1)
  - `ipsw download ipsw --device iPhone14,7 -b 20E252`

Determine changes for update

```
ipsw diff
iPhone14,7_16.4.1_20E252_Restore.ipsw
iPhone14,7_16.5_20F66_Restore.ipsw
```

16.4.1 (20E252) .vs 16.5 (20F66)

iPhone14,7_16.4.1_20E252-iPhone14,7_16.5_20F66.ipsw.diff.min.md

- 16.4.1 (20E252) .vs 16.5 (20F66)
  - IPSWs
  - Kernel
    - Version
    - Kexts
      - ⬆️ Updated (98)
  - Machos
    - 🆕 NEW (3)
    - ❌ Removed (2)
    - ⬆️ Updated (478)
    - Entitlements
  - DSC
    - WebKit
    - Dylibs
      - 🆕 NEW (1)
      - ❌ Removed (1)
      - ⬆️ Updated (590)

Map Binaries to CVE

Telephony

Available for: iPhone 8 and later, iPad Pro (all models), iPad Air 3rd generation and later, iPad 5th generation and later, and iPad mini 5th generation and later

Impact: A remote attacker may be able to cause unexpected app termination or arbitrary code execution

Description: A use-after-free issue was addressed with improved memory management.

CVE-2023-32412: Ivan Fratric of Google Project Zero

https://support.apple.com/en-us/HT213757

`cat iPhone14,7_16.4.1_20E252-iPhone14,7_16.5_20F66.ipsw.diff.md | grep -i tele`

```
user@m1 iPhone14,7 % cat iPhone14,7_16.4.1_20E252-iPhone14,7_16.5_20F66.ipsw.diff.md | grep -i tele  | grep '`' | grep '>'
> `/System/Library/Frameworks/CoreTelephony.framework/Support/CommCenter`
> `/System/Library/Frameworks/CoreTelephony.framework/Support/CommCenterMobileHelper`
> `/System/Library/Frameworks/CoreTelephony.framework/Support/CommCenterRootHelper`
> `/System/Library/PrivateFrameworks/TelephonyUtilities.framework/XPCServices/com.apple.FaceTime.FTConversationService.xpc/com.apple.FaceTime.FTConversationService`
> `/System/Library/PrivateFrameworks/TelephonyUtilities.framework/callservicesd`
> `/System/Library/PrivateFrameworks/iCloudDriveService.framework/XPCServices/TelemetryDiskChecker.xpc/TelemetryDiskChecker`
> `/System/Library/Frameworks/CoreTelephony.framework/CoreTelephony`
> `/System/Library/PrivateFrameworks/IPTelephony.framework/Support/libIPTelephony.dylib`
> `/System/Library/PrivateFrameworks/SiriPrivateLearningAnalytics.framework/SiriPrivateLearningAnalytics`
> `/System/Library/PrivateFrameworks/SiriPrivateLearningInference.framework/SiriPrivateLearningInference`
> `/System/Library/PrivateFrameworks/TelephonyUtilities.framework/TelephonyUtilities`
```

```
user@m1 iPhone14,7 % cat iPhone14,7_16.4.1_20E252-iPhone14,7_16.5_20F66.ipsw.diff.md |
> `/System/Library/Frameworks/CoreTelephony.framework/Support/CommCenter`
> `/System/Library/Frameworks/CoreTelephony.framework/Support/CommCenterMobileHelper`
> `/System/Library/Frameworks/CoreTelephony.framework/Support/CommCenterRootHelper`
> `/System/Library/PrivateFrameworks/TelephonyUtilities.framework/XPCServices/com.appl
.FaceTime.FTConversationService`
> `/System/Library/PrivateFrameworks/TelephonyUtilities.framework/callservicesd`
> `/System/Library/PrivateFrameworks/iCloudDriveService.framework/XPCServices/Telemetr
> `/System/Library/Frameworks/CoreTelephony.framework/CoreTelephony`
> `/System/Library/PrivateFrameworks/IPTelephony.framework/Support/libIPTelephony.dyli
> `/System/Library/PrivateFrameworks/SiriPrivateLearningAnalytics.framework/SiriPrivat
> `/System/Library/PrivateFrameworks/SiriPrivateLearningInference.framework/SiriPrivat
> `/System/Library/PrivateFrameworks/TelephonyUtilities.framework/TelephonyUtilities`
```

Extract the related file(s)

MachOs

⬆️ Updated

> `/System/Library/Frameworks/CoreTelephony.framework/Support/CommCenter`
> `/System/Library/Frameworks/CoreTelephony.framework/Support/CommCenterMobileHelper`
> `/System/Library/Frameworks/CoreTelephony.framework/Support/CommCenterRootHelper`

`softwareupdated`

> `/System/Library/PrivateFrameworks/MobileSoftwareUpdate.framework/Support/softwareupdated`
> `/System/Library/PrivateFrameworks/TelephonyUtilities.framework/XPCServices/com.apple.FaceTime.FTConversationService.xpc/com.apple.FaceTime.FTConversationService`
> `/System/Library/PrivateFrameworks/TelephonyUtilities.framework/callservicesd`

`TelemetryDiskChecker`

> `/System/Library/PrivateFrameworks/iCloudDriveService.framework/XPCServices/TelemetryDiskChecker.xpc/TelemetryDiskChecker`

DSC

⬆️ Updated

`CoreTelephony`

> `/System/Library/Frameworks/CoreTelephony.framework/CoreTelephony`

`libIPTelephony.dylib`

- Extract the DSC
  - `ipsw extract -d <IPSW>`
- Extract all the files
  - `ipsw dyld split dyld_shared_cache_arm64e`

```
user@m1 iPhone14,7 % tree -L 3 20E252__iPhone14,7
20E252__iPhone14,7
├── System
│   └── Library
│       ├── AccessibilityBundles
│       ├── Accounts
│       ├── Assistant
│       ├── ControlCenter
│       ├── CoreAccessories
│       ├── CoreServices
│       ├── DataClassMigrators
│       ├── Extensions
│       ├── Fitness
│       ├── Frameworks
│       ├── HIDPlugins
│       ├── Health
│       ├── MediaCapture
│       ├── Messages
│       ├── NanoTimeKit
│       ├── PreferenceBundles
│       ├── Previews
│       ├── PrivateFrameworks
│       ├── ProceduralWallpaper
│       ├── RelevanceEngine
│       ├── SystemConfiguration
│       ├── TextInput
│       ├── UserEventPlugins
│       ├── VideoCodecs
│       ├── VideoDecoders
│       ├── VideoEncoders
│       ├── VideoProcessors
│       └── VoiceServices
└── usr
    └── lib
        ├── ACIPCBTLib.dylib
        ├── AppleConvergedTransport.dylib
        ├── CarrierBundleUtilities.dylib
```

Diff the CVE Binaries



```
% ghidriff libIPTelephony.dylib.arm64.1897-iOS_16.4.1_20E252 libIPTelephony.dylib.arm64.1898.1-iOS_16.5_20F66
```

libIPTelephony.dylib.arm64.1897-iOS_16.4.1_20E252-libIPTelephony.dylib.arm64.1898.1-iOS_16.5_20F66.ghidriff.md

libIPTelephony.dylib.arm64.1897-iOS_16.4.1_20E252-libIPTelephony.dylib.arm64.1898.1-iOS_16.5_20F66.ghidriff.md Raw

libIPTelephony.dylib.arm64.1897-iOS_16.4.1_20E252-libIPTelephony.dylib.arm64.1898.1-iOS_16.5_20F66 Diff

TOC

- Visual Chart Diff
- Metadata
  - Ghidra Diff Engine
    - Command Line
  - Binary Metadata Diff
  - Program Options
  - Diff Stats
  - Strings
- Deleted
- Added
  - SipAlertInfo::SipAlertInfo
  - SipPointerVectorHeader::mergeHeader
- Modified
  - SipAlertInfoHeader::clone
  - SipAlertInfoHeader::addAlertInfo
  - ____ZN25LegacyQMIRTPCommandDriver18handleSendDTMFRespERKN3rtp15SessionSendDtmf8ResponseEh_block_invoke
  - ____ZN19QMIRTPCommandDriver18handleSendDTMFRespERKN2ms15SessionSendDTMF8ResponseENSt3__110shared_ptrI13QMIRTPSessionEE_block_invoke
- Modified (No Code Changes)

Visual Chart Diff



SipAlertInfoHeader::clone CHANGED

Viewed

```
52 -          *puVar2 = *puVar8;
53 -          puVar2 = puVar2 + 1;
54 -          puVar8 = puVar7;
55 -        } while (puVar7 != puVar9);
56 -        plVar14 = *(long **)(in_x0 + 0x40);
57 -        plVar15 = *(long **)(in_x0 + 0x48);
58          *this = &DAT_2;
59          this[9] = this[8];
60 -        do {
61 -          if (plVar14 == plVar15) {
62 -            return this;
63 -          }
64 -          lVar12 = *plVar14;
65 -          puVar2 = (undefined8 *)operator_new(0x40);
66 -          *puVar2 = 0;
67 -          puVar2[1] = 0;
68 -          puVar2[2] = 0;
69 -          puVar2[3] = &DAT_3;
70 -          std::__1::
71 -          map<std::__1::basic_string<char,std::__1::char_traits<char>,
72 -          ::
73 -          map_abi_v15006_<std::__1::__map_const_iterator<std::__1::__t
74 -                    ((map<std::__1::basic_string<char,std::__1::char_t
75 -                      *)(puVar2 + 4),(__map_const_iterator)*(undefinec
76 -                     (less *)(lVar12 + 0x28));
77 -          *(undefined *)((long)puVar2 + 0x39) = 1;
78 -          *(undefined *)(puVar2 + 7) = *(undefined *)(lVar12 + 0x38);
79 -          func_0x0001bbb30180(puVar2,lVar12);
80 -          puVar8 = (undefined8 *)this[9];
81 -          if (puVar8 < (undefined8 *)this[10]) {
82 -            *puVar8 = puVar2;
83 -            this[9] = puVar8 + 1;
84          }
```

```
55 +          *puVar3 = *puVar7;
56 +          puVar3 = puVar3 + 1;
57 +          puVar7 = puVar8;
58 +        } while (puVar8 != puVar10);
59 +        ppSVar16 = *(SipAlertInfo ***)(in_x0 + 0x40);
60 +        ppSVar17 = *(SipAlertInfo ***)(in_x0 + 0x48);
61          *this = &DAT_2;
62          this[9] = this[8];
63 +        for (; ppSVar16 != ppSVar17; ppSVar16 = ppSVar16 + 1) {
64 +          pSVar14 = *ppSVar16;
65 +          this_00 = (SipAlertInfo *)operator_new(0x40);
66 +          SipAlertInfo::SipAlertInfo(this_00,pSVar14);
67 +          ppSVar9 = (SipAlertInfo **)this[9];
68 +          if (ppSVar9 < (SipAlertInfo **)this[10]) {
69 +            *ppSVar9 = this_00;
70 +            this[9] = ppSVar9 + 1;
71          }
```

Root Cause the CVE

Issue 2440: iOS/macOS: libIPTelephony.dylib use-after-free in SIP decoder with multiple Alert-Info header lines

Code

1 of 13

Back to list

Reported by ifratric@google.com on Wed, Mar 22, 2023, 7:07 AM EDT

Project Member

There is a use-after-free vulnerability in libIPTelephony.dylib inside the SIP message decoder (SipMessageDecoder::decode() function). The vulnerable library is present on both iOS and macOS and was confirmed on macOS Ventura 13.2.1. I suspect the library is used on iOS for SIP decoding during VoLTE calls and is thus reachable via baseband, but I wasn't able to confirm this with an actual device at the time of this report.

https://bugs.chromium.org/p/project-zero/issues/detail?id=2440

“ immediately after merging the new header `SipMessage::addHeader()` deletes the new header. ”

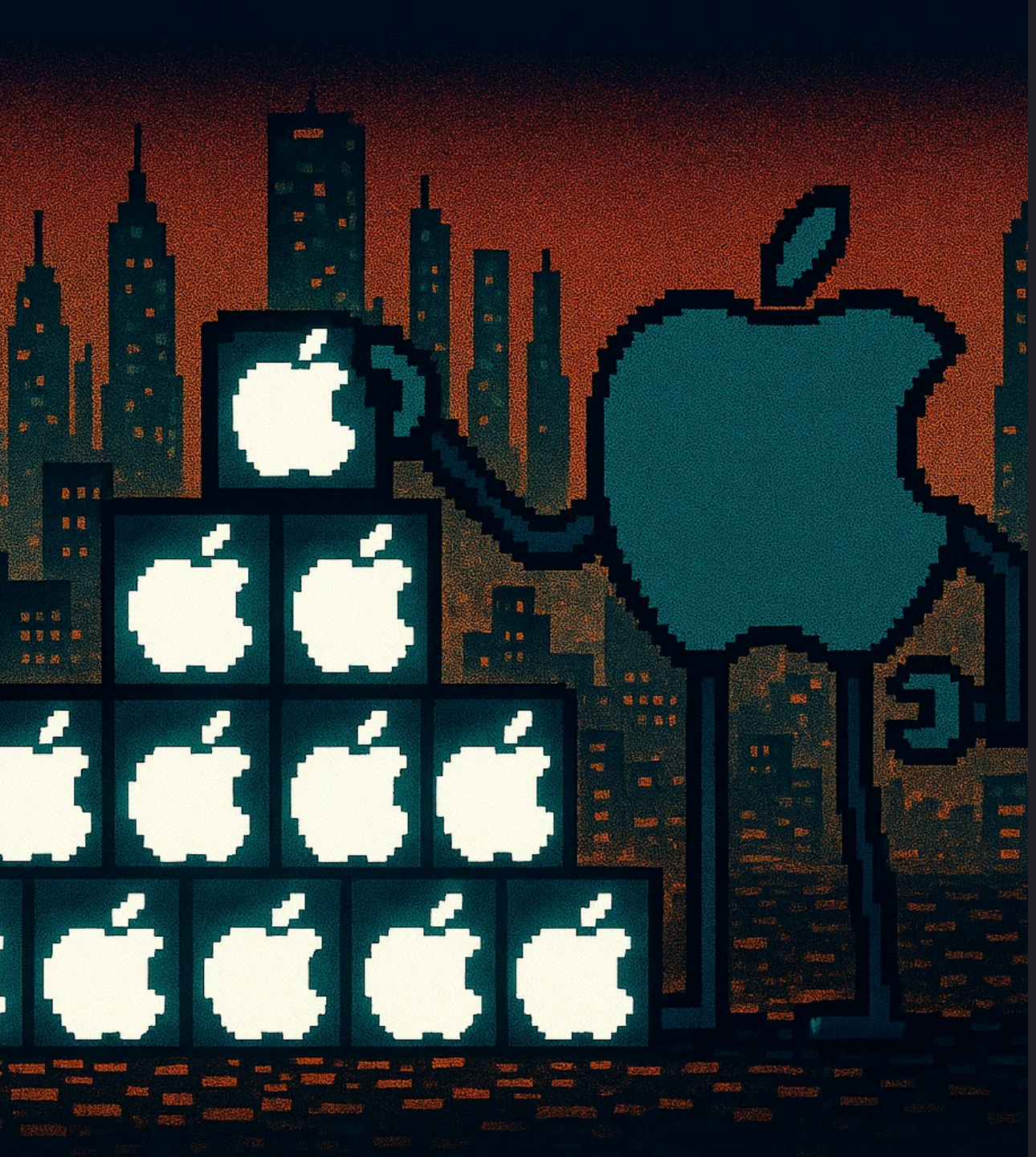


CVE-2023-32412 - use after free

- Full details of the root cause in p0 post
- We proved that we found the right file
- That Ghidra actually handled the binary well
- That we could detect the differences
- Observed Apple followed Ivan's advice

Last time, we learned how to manually find a needle in a haystack...

Tool	Problem Solved
Apple Security Update pages	Serves as the entry point for identifying CVEs and correlating them with firmware updates.
`ipsw`	Downlods, extracts and discovers changed binaries from IPSW firmware bundles
`ghidriff`	Enables code-level analysis of changes
Intuition	Provides CVE-to-binary mapping and patch interpretation



Today, we build a machine that finds the needle for us.

Design Goals

End-to-end automation

- Fetch CVEs → correlate → diff → report
- Precise binary <-> CVE mapping
- Use deterministic tools for precision
- Leverage generative AI for vulnerability reasoning
- Move beyond single, manual diffs to scalable automation
- Show a repeatable, hands-off workflow for any release

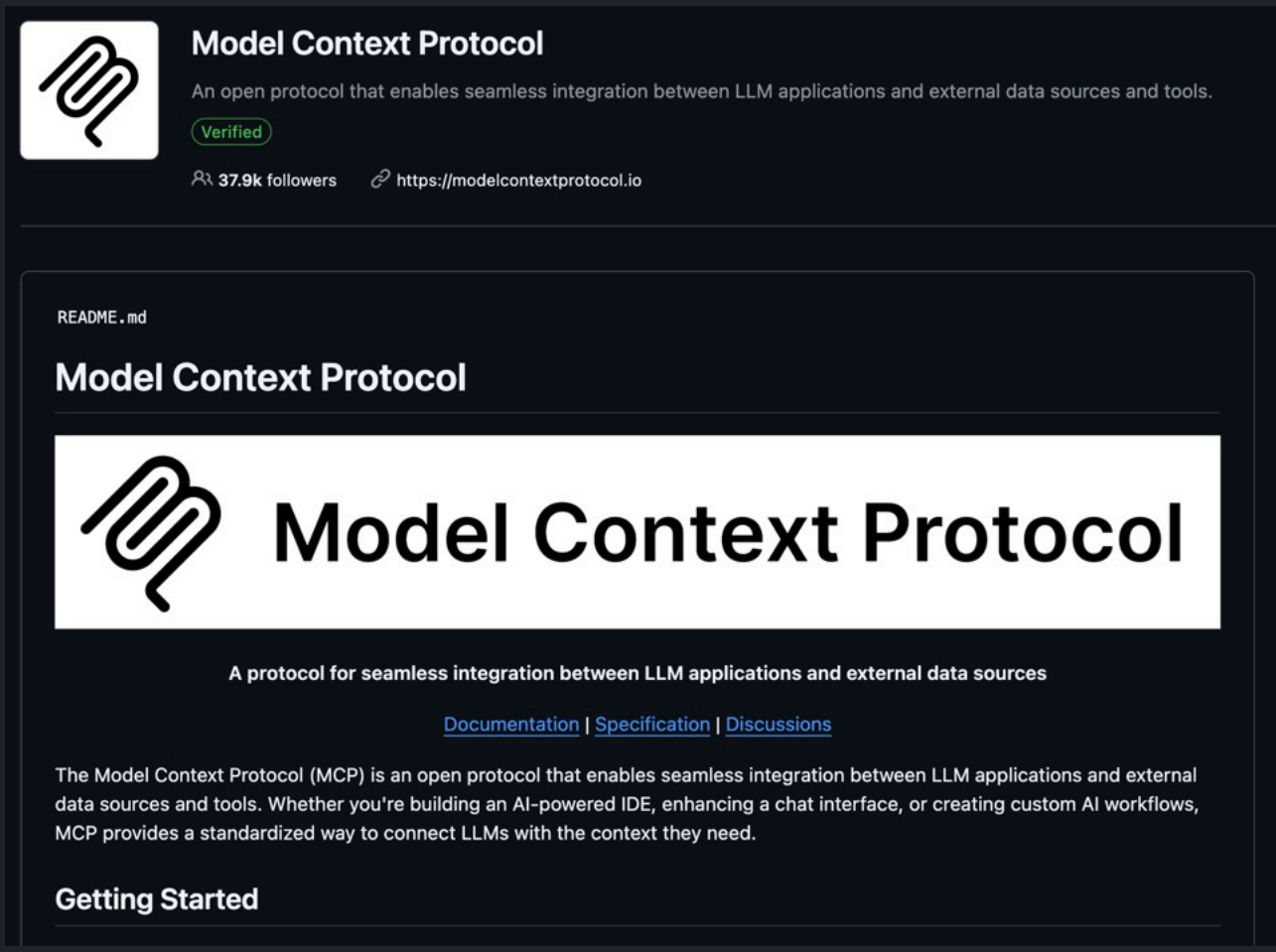
Goal: Turn black-box updates into transparent vulnerability insights

My Trajectory

The gravity of AI

Early 2025

Model Context Protocol pulled me in...



Model Context Protocol

- A standard way to expose tools to an LLM
- **Model Context Protocol (MCP)** showed structured tool/LLM interplay

Private

trends.google.com

Google Trends

Home | Explore | Trending Now

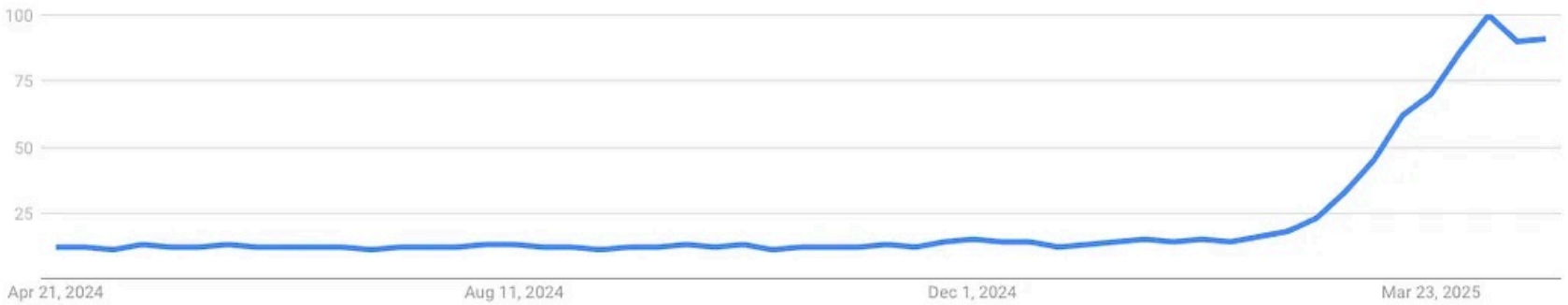
Sign in

MCP
Search term

+ Compare

Worldwide | Past 12 months | All categories | Web Search

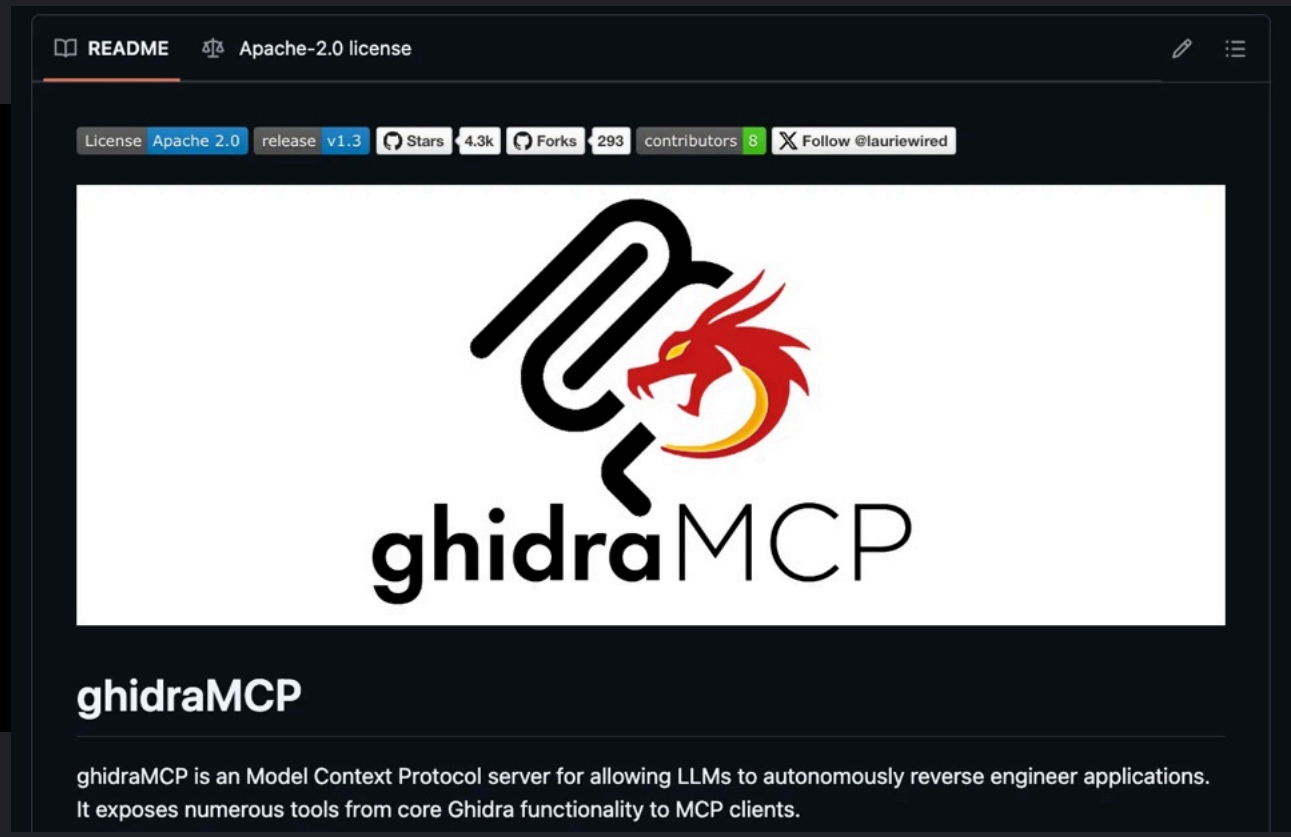
Interest over time



ghidraMCP

https://github.com/LaurieWired/GhidraMCP


LaurieWired
@lauriewired
Just built an MCP for Ghidra.
Now basically any LLM (Claude, Gemini, local...) can Reverse Engineer malware for you. With the right prompting, it automates a *ton* of tedious tasks.
One-shot markups of entire binaries with just a click.
Open source, on Github now.


README
Apache-2.0 license
License Apache 2.0
release v1.3
Stars 4.3k
Forks 293
contributors 8
Follow @lauriewired
ghidraMCP
ghidraMCP
ghidraMCP is an Model Context Protocol server for allowing LLMs to autonomously reverse engineer applications.
It exposes numerous tools from core Ghidra functionality to MCP clients.

Claude

Untitled

16. `_ZNSs4swapERSs` - Swap string contents
17. `_ZNSs6assignERKSs` - String assignment
18. `_Z13EncryptFolderSs` - Encrypt folder function (core functionality)
19. `_ZNSt6thread4joinEv` - Join thread
20. `_ZN3ghc10filesystem6detail15systemErrorTextIiEESsT_` - Get system error text
21. `_ZN3ghc10filesystem16filesystem_errorC2ERKSsRKNS0_4pathESt10error_code` - Filesystem error constructor
22. `_ZSt16__throw_bad_castv` - Throw bad cast exception

Let's rename these methods to reflect their functionality:

View result from **rename_function** from **ghidra** (local) >
View result from **rename_function** from **ghidra** (local) >
View result from **rename_function** from **ghidra** (local) >
View result from **rename_function** from **ghidra** (local) >
View result from **rename_function** from **ghidra** (local) >
View result from **rename_function** from **ghidra** (local) >
View result from **rename_function** from **ghidra** (local) >
View result from **rename_function** from **ghidra** (local) >
View result from **rename_function** from **ghidra** (local) >
View result from **rename_function** from **ghidra** (local) >

Running **rename_function** from **ghidra** (local)

```
{
  `old_name`: `_ZN3ghc10filesystem8absoluteERKNS0_4pathERSt10error_code`,
  `new_name`: `GetAbsolut`
}
```

Reply to Claude...

Claude 3.7 Sonnet | Choose style

Decompile: RansomwareMainProcess - (0d6c3de5aebbbe85939d7588150edf7b7bdc712fceb6a83d79e65b6f79bfc2ef.elf)

```
undefined1 *local_68 [2];
undefined1 *local_58 [2];
long local_48 [2];
timespec local_38;

_ZSt16__ostream_insertIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_PKS3_l
          (&_ZSt4cout,"ENCRYPTION",10);
plVar1 = *(long **)(&DAT_0062c410 + *(long *)(_ZSt4cout + -0x18));
if (plVar1 != (long *)0x0) {
  if ((char)plVar1[7] == '\0') {
    InitializeCharacterWidening(plVar1);
    iVar11 = 10;
    if (*(code **)(*plVar1 + 0x30) != PerformCharacterWidening) {
      cVar8 = (**(code **)(*plVar1 + 0x30))(plVar1,10);
      iVar11 = (int)cVar8;
    }
  }
  else {
    iVar11 = (int)*(char *)((long)plVar1 + 0x43);
  }
  uVar9 = StreamOutputPutChar(&_ZSt4cout,iVar11);
  StreamOutputFlush(uVar9);
  InitializeEncryptionSystem();
  clock_gettime(1,&local_38);
  lVar5 = CONCAT44(local_38.tv_sec._4_4_,(int)local_38.tv_sec);
  lVar3 = local_38.tv_nsec / 1000000;
  _ZNSsC1EPKcRKSaIcE(local_d8,"C:/Users/dssd/Desktop/src",&local_38);
  if (*(long *)(forcedPath + -0x18) == 0) {
    _ZNSsC1EPKcRKSaIcE(local_68,"/vmfs/volumes",&local_dc);
    puVar7 = local_68[0];
    sVar2 = *(size_t *)(local_68[0] + -0x18);
    if (local_68[0] == local_68[0] + sVar2) {
      local_c8[0] = (size_t *)&DAT_0062c5d8;
    }
    else {
                    /* try { // try from 0040aa82 to 0040aa86 has its CatchHandler @ 0040b00d */
      psVar10 = (size_t *)_ZNSs4_Rep9_S_createEmmRKSaIcE(sVar2,0,&local_db);
      local_c8[0] = psVar10 + 3;
      if (sVar2 == 1) {
        *(undefined1 *)(psVar10 + 3) = *puVar7;
      }
      else {
        local_c8[0] = (size_t *)memcpy(local_c8[0],puVar7,sVar2);
      }
      if (psVar10 != &_ZNSs4_Rep20_S_empty_rep_storageE) {
        *(undefined4 *)(psVar10 + 2) = 0;
        *psVar10 = sVar2;
        *(undefined1 *)((long)psVar10 + sVar2 + 0x18) = 0;
      }
```

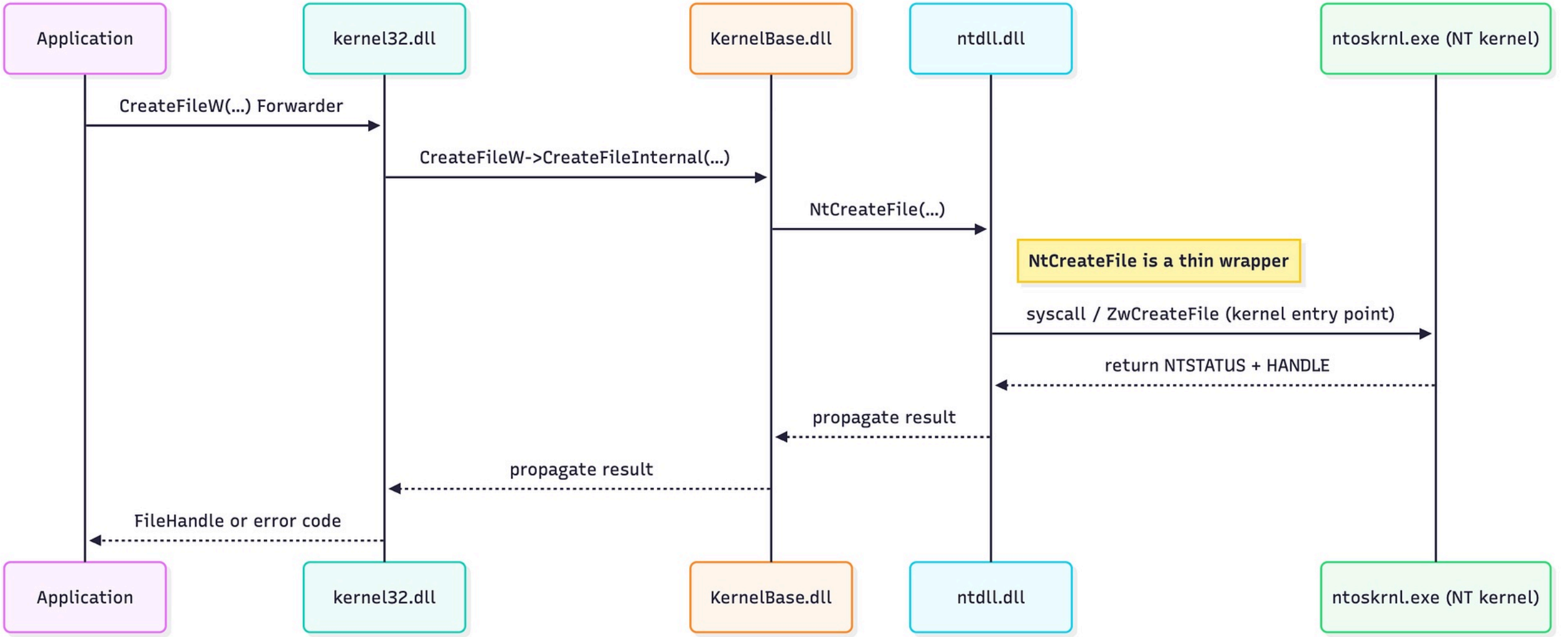
0040a860 | _Z18AutoBuilderP... | PUSH R13

`pyghidra-mcp`

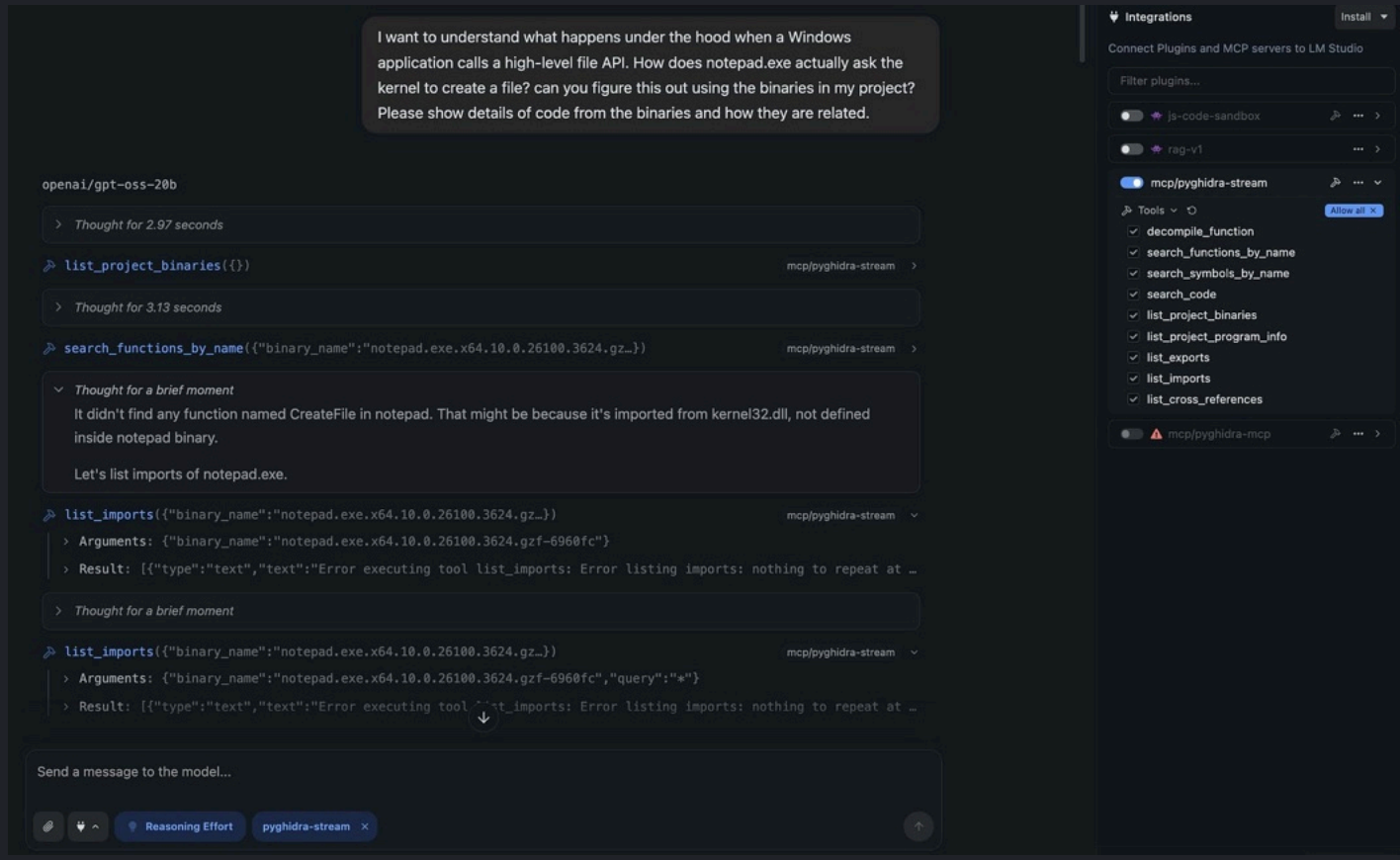
https://github.com/clearbluejar/pyghidra-mcp

- Built **`pyghidra-mcp`** → piping **project-wide** Ghidra outputs into LLMs for assisted RE
- Gained experience developing with LLMs managing context windows



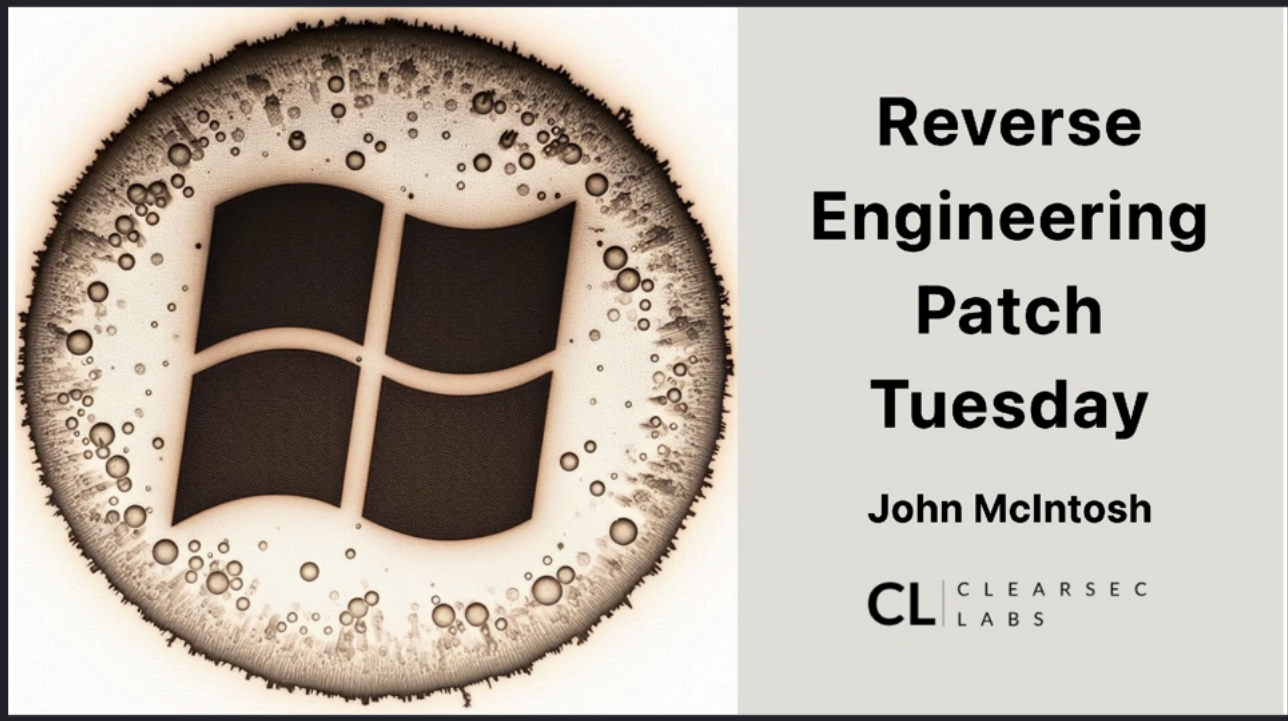

Application
kernel32.dll
KernelBase.dll
ntdll.dll
ntoskrnl.exe (NT kernel)
CreateFileW(...) Forwarder
CreateFileW->CreateFileInternal(...)
NtCreateFile(...)
NtCreateFile is a thin wrapper
syscall / ZwCreateFile (kernel entry point)
return NTSTATUS + HANDLE
propagate result
propagate result
FileHandle or error code
Application
kernel32.dll
KernelBase.dll
ntdll.dll
ntoskrnl.exe (NT kernel)

"It made 22 tool calls in a one-shot run to analyze multiple binaries and traced the call chain from user-mode through kernel"



Reverse Engineering Patch Tuesday

- I've done this for Windows without LLMs 💪
- Prior **Windows patch diffing automation** → proved feasibility at scale



DARPA AIxCC

> The AI Cyber Challenge (AIxCC) challenged competitors to build an autonomous Cyber Reasoning System (CRS) capable of finding, triggering, and patching security vulnerabilities in large codebases.



How the Competition Played Out

Select a team below for a more detailed breakdown of their CRSs performance:












Team Atlanta
——-
PLACEMENT
1 Team Atlanta
2 Trail of Bits
3 Theori
4 All You Need IS A Fuzzing Brain
5 Shellphish
6 42-b3yond-6ug
7 Lacrosse

Team Atlanta
——-
SCORED ROUND
Vulnerabilites Found → 43
Successful Patches → 31
Total Score → 393

Team Atlanta
——-
ACHIEVEMENTS

- Bundle Baron ///
  Most Scoring Bundles
- Prince of the Patch ///
  Most Real World Vulns Patched
- Professional Assassin ///
  Had a positively scoring PoV percentage higher than 95%

> LLMs have enabled agentic workflows showing promise at automating some tasks previously only achievable by humans.

> LLMs are well-suited for tasks that lack a clear efficient algorithm but that humans can solve reliably through a combination of intuition, heuristics, and reasoning...

Lessons learned building an AI hacker AI Cyber Challenge - 2025 - Theori

Manual -> Automation?



> every manual step could be paired with a tool + agent

Deterministic Tools + Agents

- **Deterministic tools** provide *ground truth*
  → reproducible outputs: binary diffs, symbol tables, CVE metadata
- **Agents (LLMs)** provide *reasoning*
  → contextualize diffs, map to CVEs, explain mitigations
- Together: **tools** constrain the search space, **agents** narrate and correlate results

Manual → Hybrid Mapping

1. Identify CVE → *CVE feeds + Platform Classifier*
2. Download IPSW → `ipsw` *deterministic fetch*
3. Determine changes → `ipsw diff` *JSON*
4. Map binaries → *CVE Correlator (agent reasoning)*
5. Extract files → `ipsw` *extraction*
6. Diff binaries → `ghidriff` *function-level diffs*
7. Root cause → *Diff Analyzer (agent explanation + diff analysis + root cause analysis)*

Toolchain Overview

- 📚 Curated Data Sources
- ⚙️ Deterministic Tools
- 🤖 Agents

Curated Data Sources

- CVE feeds (NVD + Apple advisories) → authoritative vulnerability context
- Curated CVE <-> build mapping → links CVEs to specific OS releases
- Apple IPSW firmware archives → monthly security update images
- Build metadata → track Patch Monday cadence

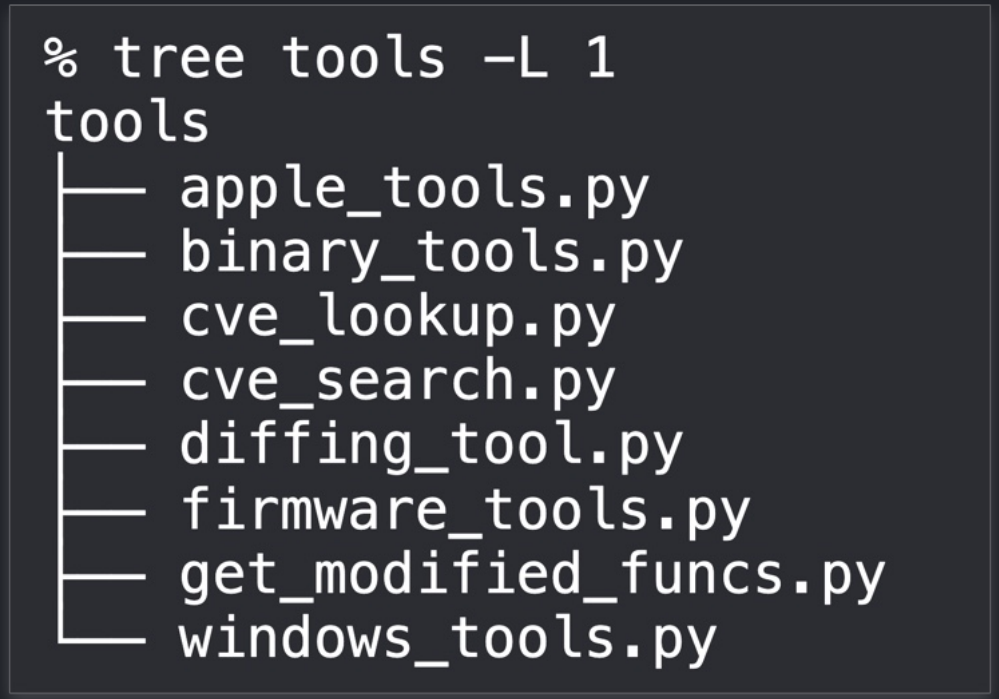
```
{
    "id": "CVE-2024-44203",
    "module": [
        "Sandbox"
    ],
    "impact": "An app may be able to access a user's Photos Library",
    "description": "A permissions issue was addressed with additional restrictions.",
    "credit": "Yi\u011fit Can YILMAZ (@yilmazcanyigit), Wojciech Regula of SecuRing (wojciechregula.blog), Kirin (@Pwnr
    "updates": [
        {
            "link": "https://support.apple.com/en-us/121238",
            "version": "macOS Sequoia 15",
            "available": [
                "Mac Studio (2022 and later), iMac (2019 and later), Mac Pro (2019 and later), Mac mini (2018 and later)
            ]
        }
    ],
    "published_date": "August 06, 2025",
    "bugzilla": null
},
{
    "id": "CVE-2024-44174",
    "module": [
        "Screen Capture"
    ],
    "impact": "An attacker may be able to view restricted content from the lock screen",
    "description": "The issue was addressed with improved checks.",
    "credit": "Vivek Dhar, working as Assistant Sub-Inspector (RM) in Border Security Force (Frontier Headquarter BSF Ka
    "updates": [
```

Apple security updates and Rapid Security Responses

Name and information link	Available for	Release date
iOS 26.0.1 and iPadOS 26.0.1	iPhone 11 and later, iPad Pro 12.9-inch 3rd generation and later, iPad Pro 11-inch 1st generation and later, iPad Air 3rd generation and later, iPad 8th generation and later, and iPad mini 5th generation and later	29 Sep 2025
iOS 18.7.1 and iPadOS 18.7.1	iPhone XS and later, iPad Pro 13-inch, iPad Pro 12.9-inch 3rd generation and later, iPad Pro 11-inch 1st generation and later, iPad Air 3rd generation and later, iPad 7th generation and later, and iPad mini 5th generation and later	29 Sep 2025
macOS Tahoe 26.0.1	macOS Tahoe	29 Sep 2025
macOS Sequoia 15.7.1	macOS Sequoia	29 Sep 2025
macOS Sonoma 14.8.1	macOS Sonoma	29 Sep 2025
visionOS 26.0.1	Apple Vision Pro	29 Sep 2025

Deterministic Tools

- `ipsw`
  - download/extract - get IPSW Firmware / binaries
  - diff - JSON of changed binaries
- `ghidriff` - function-level JSON diffs (CLI + Ghidra)
- `get_symbols` - Vector DB + `nm`
- More...


% tree tools -L 1
tools
├── apple_tools.py
├── binary_tools.py
├── cve_lookup.py
├── cve_search.py
├── diffing_tool.py
├── firmware_tools.py
├── get_modified_funcs.py
└── windows_tools.py

```python
@tool
def fetch_cve_details(cve_id: str) -> CveDetails:
    "fetch CVE details"
    print(f"--- TOOL: Fetching details for {cve_id} from NVD ---")

    if not is_valid_cve_id(cve_id):
        return {"error": f"Invalid CVE ID format: {cve_id}"}

    try:
        url = f"https://services.nvd.nist.gov/rest/json/cves/2.0?cveId={cve_id}"
        response = requests.get(url, timeout=10)
        response.raise_for_status()
        data = response.json()
```

```python
def get_modified_functions(
    json_diff_path: str, function_name: Optional[str] = None
) -> List[Dict[str, Any]]:
    """
    Tool: Extract modified function diffs from a ghidriff JSON diff file.

    Args:
        json_diff_path (str): Path to the ghidriff JSON diff file.
        function_name (str, optional): Specific modified function name to extract.
                                       If provided, only that function's diff is returned.
                                       If omitted, all modified functions are returned.
```

🤖 Agents

- Platform Classifier (DSPy signature)
  - Identifies the platform related to the CVE
- CVE Correlator (ReAct)
  - Matches advisories to binaries using `get_symbols`
- Diff Analyzer (ReAct)
  - Explains function-level changes, reasons root cause analysis using diffs, cve context, and tools like `get_modified_functions`

Implemented Using DSPy

a declarative framework for building modular AI software.



DSPy

Programming LLMs Instead of Prompting

> It allows you to **iterate fast** on structured code, rather than brittle strings, and offers algorithms that compile AI programs into effective prompts and weights for your language models, whether you're building simple classifiers, sophisticated RAG pipelines, or Agent loops.

```python
class PlatformClassifierSignature(dspy.Signature):
    """Classify the primary affected platform or vendor based on CVE metadata."""

    cve_description = dspy.InputField(
        desc="The full English-language description of the CVE."
    )
    cwe_id = dspy.InputField(desc="The CWE identifier (e.g., 'CWE-674') if available.")
    references = dspy.InputField(
        desc="A list of reference URLs associated with the CVE."
    )

    platform = dspy.OutputField(
        desc=(
            "The primary affected platform or vendor, chosen from one of the following strings: "
            "'windows', 'apple', 'android', 'linux', 'network_appliance', 'unknown'. "
            "Use 'unknown' if the platform cannot be confidently inferred."
        )
    )
```

```python
# 2. Classify Platform
async with cl.Step(name="2. Classifying Platform") as step:
    platform_classifier = PlatformClassifierModule()
    metadata = CveMetadata(
        cve_description=cve_details["description"],
        cwe_id=cve_details.get("cwe_id"),
        references=cve_details.get("references", []),
    )
    prediction = platform_classifier(metadata)
    platform = prediction.platform
```

ReAct Agents

```
react_agent = dspy.ReAct(
    signature="question -> answer",  # Input/output specification
    tools=[tool1, tool2, tool3],     # List of available tools
    max_iters=10                     # Maximum number of tool call iterations
)
```

I think "agent" may finally have a widely enough agreed upon definition to be useful jargon now

I've noticed something interesting over the past few weeks: I've started using the term "agent" in conversations where I don't feel the need to then define it, roll my eyes or wrap it in scare quotes.

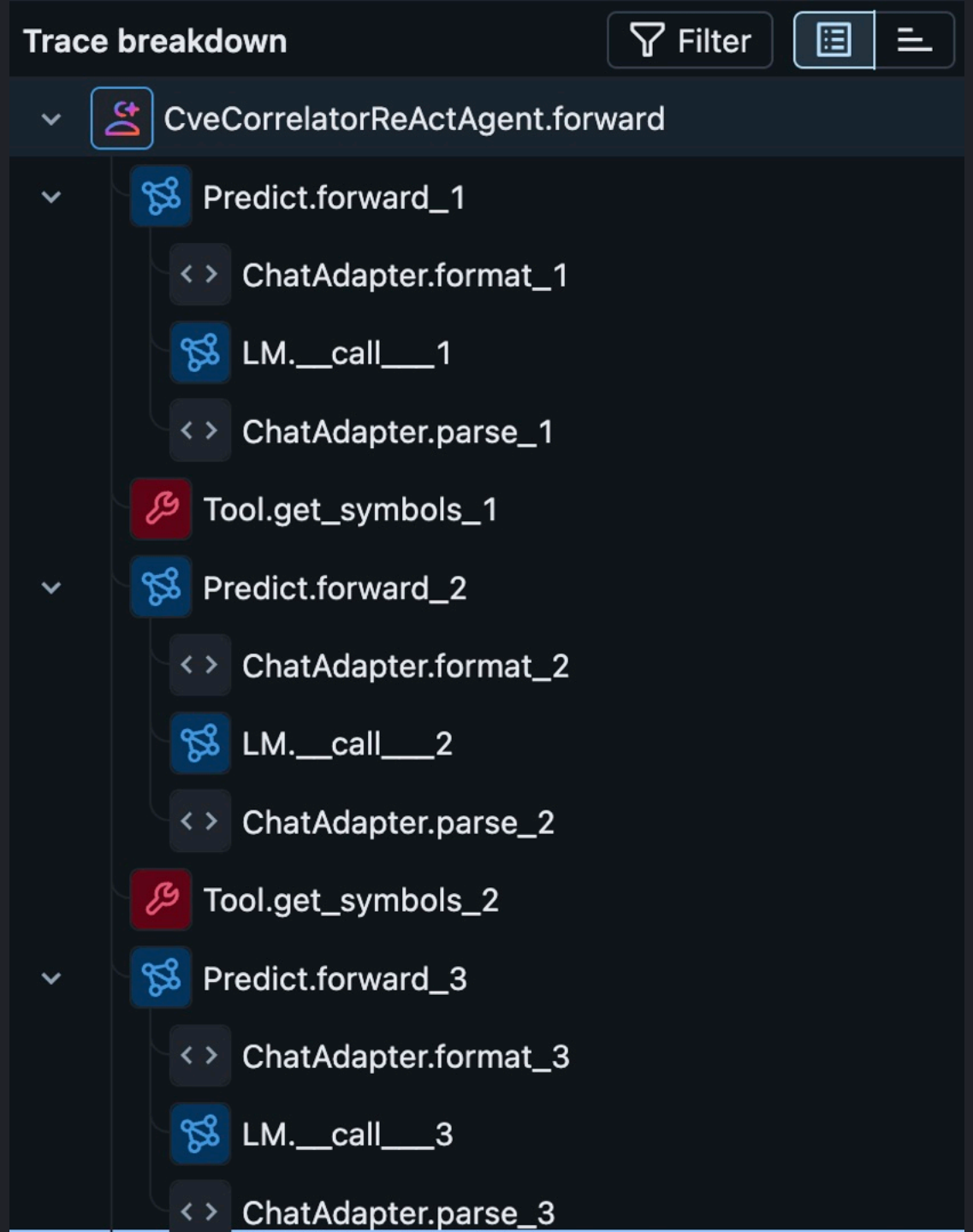
This is a big piece of personal character development for me!

Moving forward, when I talk about agents I'm going to use this:

An LLM agent runs tools in a loop to achieve a goal.

ReAct Pattern

- **Reasoning**: The model explains its thought process step by step .
- **Acting**: The model performs an action based on that reasoning (calls tools)
- **Interleaving**: Instead of separating these phases, the model alternates between reasoning and acting until it reaches a final answer.


Trace breakdown
Filter
CveCorrelatorReActAgent.forward
Predict.forward_1
ChatAdapter.format_1
LM.__call___1
ChatAdapter.parse_1
Tool.get_symbols_1
Predict.forward_2
ChatAdapter.format_2
LM.__call___2
ChatAdapter.parse_2
Tool.get_symbols_2
Predict.forward_3
ChatAdapter.format_3
LM.__call___3
ChatAdapter.parse_3

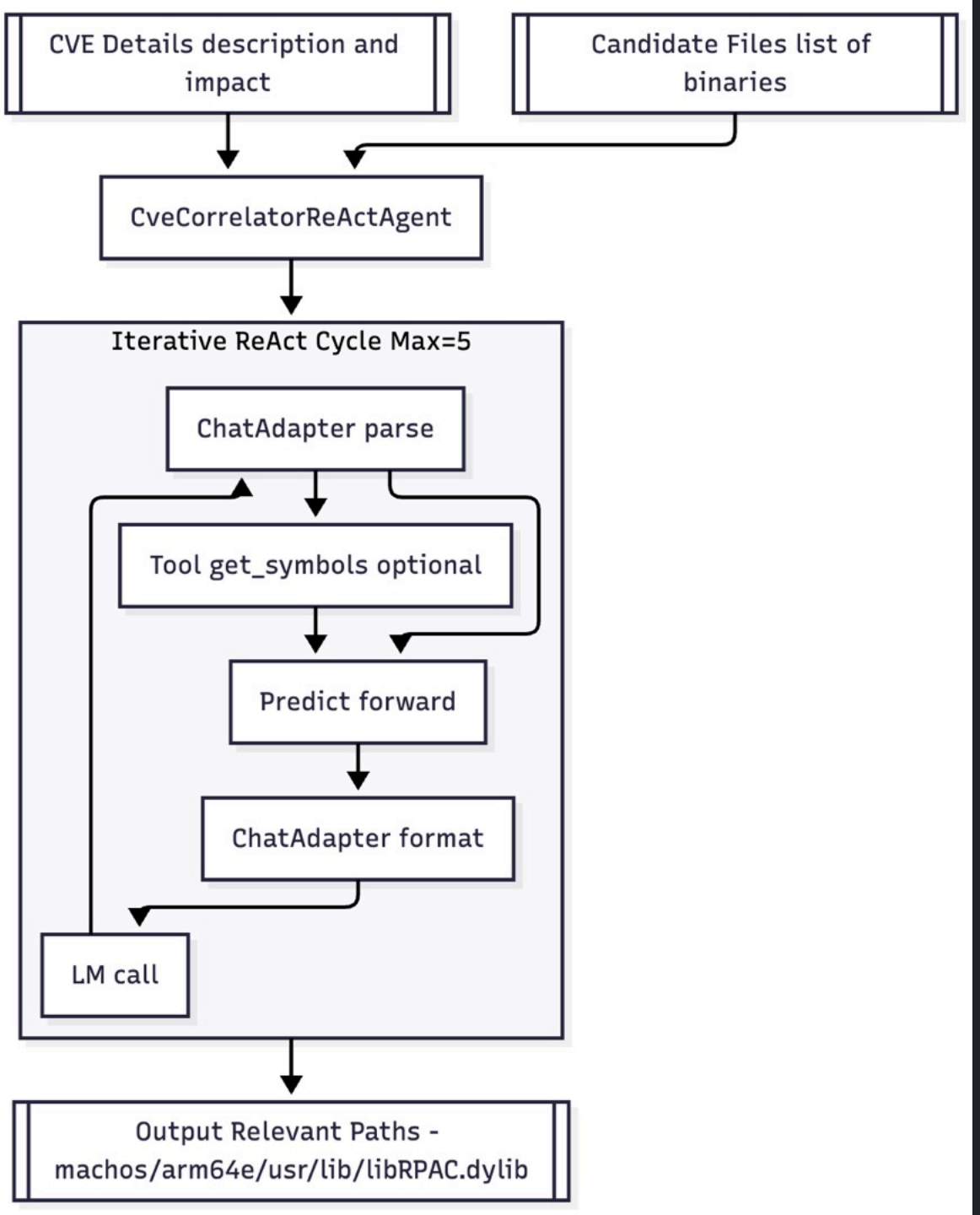
```
The agent should:
1. Use all available CVE context (description, impact, modules, affected identifiers, versions).
2. First, reason based on file names and paths. Do they match modules or keywords from the CVE?
3. If uncertain, call the `get_symbols` tool on the file path to extract exported symbols.
4. Compare symbols and file purpose against the CVE details.
5. Assign a confidence score (0.0–1.0) for each candidate file.
6. Return all plausible matches, not just the top one.

You are an Agent. In each episode, you will be given the fields `cve_details`, `candidate_files`, `
 Your goal is to use one or more of the supplied tools to collect any necessary information for pro

 To do this, you will interleave next_thought, next_tool_name, and next_tool_args in each turn, and
 After each tool call, you receive a resulting observation, which gets appended to your trajectory.

 When writing next_thought, you may reason about the current situation and plan for future steps.
 When selecting the next_tool_name and its next_tool_args, the tool must be one of:

 (1) get_symbols, whose description is <desc>  Extract and rank exported symbols from a binary base
 (2) finish, whose description is <desc>Marks the task as complete. That is, signals that all infor
 When providing `next_tool_args`, the value inside the field must be in JSON format
```


CVE Details description and impact
Candidate Files list of binaries
CveCorrelatorReActAgent
Iterative ReAct Cycle Max=5
ChatAdapter parse
Tool get_symbols optional
Predict forward
ChatAdapter format
LM call
Output Relevant Paths - machos/arm64e/usr/lib/libRPAC.dylib

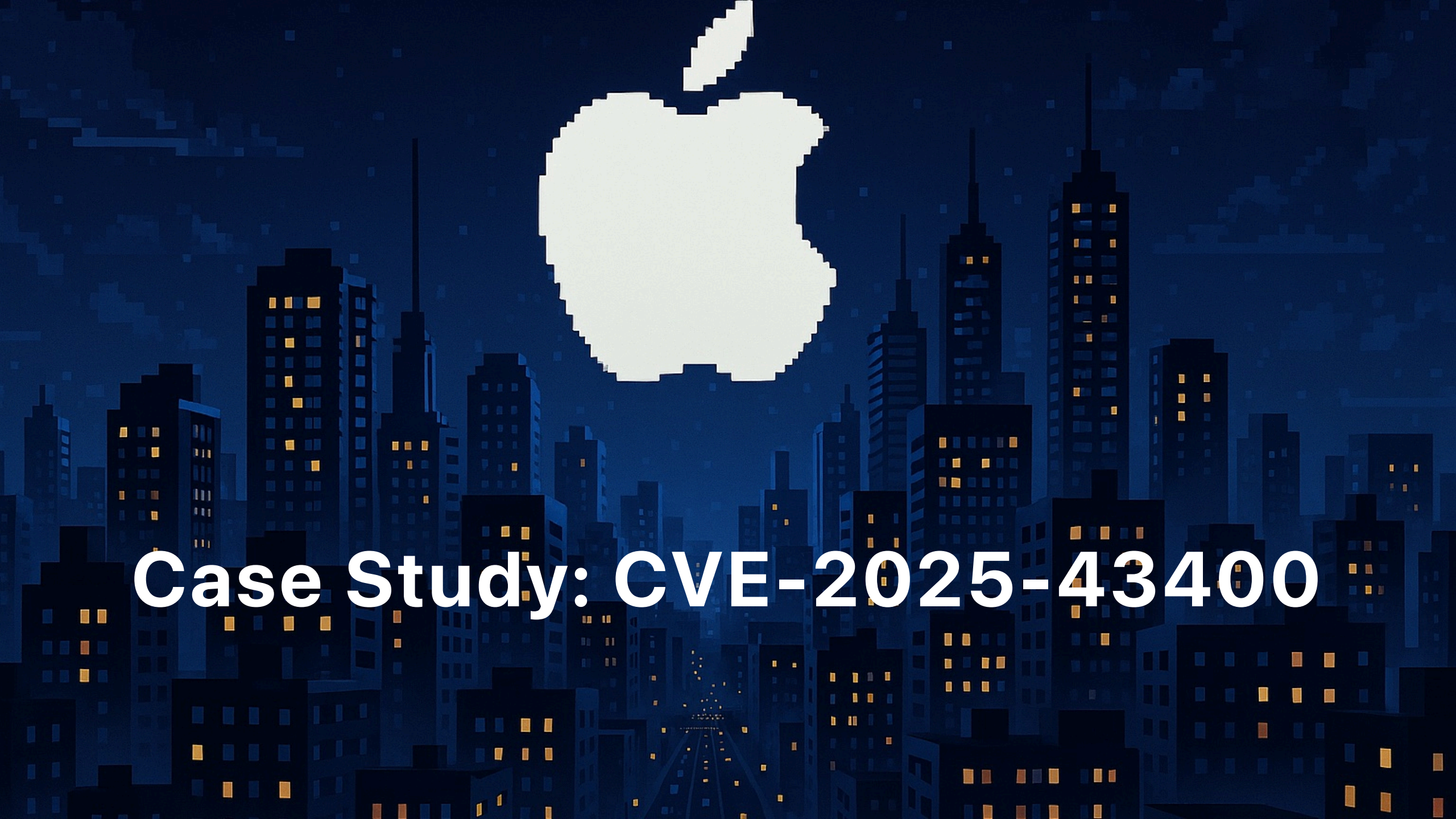
Key Takeaway

- Deterministic tools **anchor the facts**
- Agents **connect the dots**

Hybrid system transforms patch diffing from a manual, intuition-driven art into a **scalable, explainable science**

Methodology

Step	Description	Type
1. Fetch CVE Details	Tool call to retrieve structured CVE metadata	🟩 Deterministic
2. Classify Platform	DSPy module uses LLM to infer platform	🟨 Agentic
3. Select Platform Agent	Rule-based logic to route to correct agent	🟩 Deterministic
4. Choose Device	User input or default selection	🟩 Deterministic
5. Find Candidate Binaries	`ipsw` / IPSW assisted binary discovery	🟩 Deterministic
6. Correlate Binary	Agent reasons over CVE + binaries	🟨 Agentic
7. Perform Binary Diff	Function-level diffing via `ghidriff`	🟩 Deterministic
8. Analyze Diff	Agent explains patch, root cause	🟨 Agentic



Case Study: CVE-2025-43400

CVE-2025-43400

- Patrick's tweet about latest out-of-band CVE at 2230 Sep, 29 2025
- I happened to be ~~working on my slides~~ checking X...



2.5 hours later...
(report only took a few minutes to process)



Demo CVE-2025-43400

Demo uses Chainlit for UI

http://localhost:8000/

CVE-2025-43400 - FontParser - OOBW

- Found relevant binary from 22 candidates
- Identified the vulnerable functions matching the CVE description
- Agent successfully used tools to verify findings both in CVE correlation and Diff Analysis
- Diff analysis completed within hours of patch release


Case Study: CVE-2025-31325



THE ATTACK CYCLE

Bonus: CVE-2025-31235

- Just patched, May 16, 2025!
- Double-free in `CoreAudio`/`coreaudiod`
- More info soon

Audio

Available for: macOS Sequoia

Impact: An app may be able to cause unexpected system termination

Description: A double free issue was addressed with improved memory management.

CVE-2025-31235: Dillon Franke working with Google Project Zero

©2025 Google

👋 Patch Diffing Agent.

Ready to analyze, compare, and illuminate binary changes with precision.

Choose a mode to get started:

💬 Q&A Mode 🔎 Search Mode

Used 1. Fetching CVE-2025-31235 Details

Description: A double free issue was addressed with improved memory management. This issue is fixed in iPadOS 17.7.7, macOS Ventura 13.7.6, macOS Sequoia 15.5, macOS Sonoma 14.7.6. An app may be able to cause unexpected system termination.
Severity: MEDIUM (6.5)

Used 2. Classifying Platform

Detected Platform: apple

Selected Agent: ApplePlatformAgent

Used 4. Choosing Device for Candidate Binaries

Selected: 🖥️ Mac14,12

Selected device: **Mac14,12**

Used 5. Finding Candidate Binaries

Candidate Binaries(1868):
com.apple.iokit.IOAccessoryManager, com.apple.driver.AppleConvergedPCI, com.apple.driver.AppleAVE2, com.apple.driver.AppleBCMWLANCore, com.apple.driver.AppleBCMWLANBusInterfacePCIe, com.apple.filesystems.afpfs, com.apple.AppleSystemPolicy, com.apple.filesystems.lifs, com.apple.security.sandbox, com.apple.driver.AppleUSBAudio, com.apple.iokit.IOMobileGraphicsFamily-DCP, com.apple.filesystems.apfs, com.apple.iokit.IO80211Family, com.apple.kernel, com.apple.driver.usb.cdc.ncm, com.apple.iokit.IOPCIFamily, com.apple.nke.asp_tcp, com.apple.kext.CoreTrust, com.apple.filesystems.smbfs, com.apple.driver.AppleMobileFileIntegrity...

Preview: REMAINING CANDIDATES


Trace breakdown
Filter
CveCorrelatorReActAgent.forward
Predict.forward_1
ChatAdapter.format_1
LM.__call___1
ChatAdapter.parse_1
Tool.get_symbols_1
Predict.forward_2
ChatAdapter.format_2
LM.__call___2
ChatAdapter.parse_2
Tool.get_symbols_2
Predict.forward_3
ChatAdapter.format_3
LM.__call___3
ChatAdapter.parse_3
ChainOfThought.forward
Predict.forward_4
ChatAdapter.format_4
LM.__call___4
ChatAdapter.parse_4

Used 6. Correlating Binary with Agent

🧠 **Reasoning:**

Reasoning

The CVE description indicates a double-free issue in the Audio module that was mitigated by improved memory management. The candidate list contains several audio-related components: a USB audio driver kext, core audio frameworks, private frameworks, and various plug-in binaries.

I first examined the exported symbols of the most obvious candidates:

1. **AppleUSBAudio kext** – only exposed two stream-conversion functions; no memory-management or free-related symbols were present.
2. **CoreAudio framework (x86_64)** – exported many lifecycle and disposal functions such as `_AudioHardwareDisposeForTesting` and `_AudioHardwareUnload`. While none of the symbols explicitly mention `free`, the presence of dispose/unload routines suggests that changes to memory handling could be located here.

{"cve_details": {"description": "A double free issue was addressed with improved memory mana...

ID tr-938f9de04986de03d874afc752b773fa Latency 5.16m

Summary Details & Timeline

Search

Trace breakdown Filter

- ChatAdapter.parse_3
- ChainOfThought.forward
- Predict.forward_4
  - ChatAdapter.format_4
  - LM.__call___4
  - JSONAdapter.format_1
    - JSONAdapter.format...
  - LM.__call___5
  - JSONAdapter.format_3
    - JSONAdapter.format...
  - LM.__call___6
  - JSONAdapter.parse

Inputs / Outputs Attributes Events Assessments

Affected Binary: CoreAudio **Vulnerability Type:** Potential Memory Management Issues **Function:** SetPropertyData

Unified Diff Snippet

```
1 - old code line
2 + new code line
```

Agent Analysis The `SetPropertyData` function in `CoreAudio` shows changes in how properties like `kAudioDevicePropertyBufferSize` and `kAudioDevicePropertyIODataSyphonBlock` are set. The diff includes exception handling and calls to `HALC_ProxyObject::SetPropertyData`. While the changes are present, there's no immediately obvious double-free scenario. Further investigation with the full source code and debugging would be needed.

Affected Binary: CoreAudio **Vulnerability Type:** Potential Memory Management Issues **Function:** CreatePlugIn

Unified Diff Snippet

```
1 - old code line
```

No other candidate files were inspected in depth, but given the module focus (Audio) and the nature of the change (“improved memory management”), CoreAudio is the most plausible location for the fix. I therefore assign it a moderate confidence score. The USB audio driver kext receives a low score because its symbols do not relate to memory handling. A generic score of 0.5 is given to the AudioServerDriver framework as a possible secondary candidate, since server-side drivers often contain allocation logic.

Relevant Binaries (score ≥ threshold):

- ✅ `CoreAudio` (score: 0.70)
- ✅ `AudioServerDriver` (score: 0.50)

Preview: OTHER SCORED CANDIDATES

Used 7. Performing Binary Diff

✅ Completed 1/2: `CoreAudio`

Modified Functions in `CoreAudio`:

- `Activate`
- `SetPropertyData`
- `CreatePlugIn`
- `HasProperty`
- `StartIOThread`
- `GetPropertyData`
- `SetFromCFRepresentation_MakeEngineInfo`
- `UpdateSubDeviceList`
- `____ZNK15HALS_MetaDevice15GetPropertyDataEjRK26AudioObjectPropertyAd`
- `SupportsIsolatedIO`
- `HandleClientTimeout`

✅ Completed 2/2: `AudioServerDriver`

Modified Functions in `AudioServerDriver`:

- `___43-[ASDSRCStream__allocateStreamingResources]_block_invoke`
- `__uninitialized_allocator_copy_impl[abi:ne190102]<std::allocator<ASDSRCStreamHelper::ASDUnderlyingStreamDoIOOperationBlocks>`
- `readInput`
- `__allocate_at_least[abi:ne190102]<std::allocator<ASDDSPStreamHelper::DSPStream>>`
- `operator=`
- `operator()[abi:ne190102]`
- `-[ASDAudioDevice_hasProperty:]`
- `writeMix`
- `DSPStream`
- `___43-[ASDDSPStream__allocateStreamingResources]_block_invoke`


Agent
Tool
Obs
Thought 0: The CVE description mentions a double free issue related to ...
Call get_modified_functions
name: Activate, diff: --- HALS_PlugInDevice::Activa...
Thought 1: I have extracted the modified functions from CoreAudio. Now ...
Call get_modified_functions
name: ___43-ASDSRCStream__allocateStreamingResources_...
Thought 2: I have extracted the modified functions from AudioServerDriv...
Call finish
Completed.
Agent
Tool
Obs

Used 8. Analyzing Diff

Reasoning:

Reasoning

The analysis of the provided diffs, in the context of a reported double free vulnerability, reveals several areas of concern within CoreAudio and AudioServerDriver. The changes in `Activate` and `SetPropertyData` in CoreAudio suggest modifications to resource management and locking mechanisms, which could potentially introduce double frees if not handled carefully. The extensive changes in `CreatePlugIn` also warrant close scrutiny. Similarly, the modifications in `___43-[ASDSRCStream__allocateStreamingResources]_block_invoke` and other functions in AudioServerDriver, particularly those involving memory allocation and deallocation, need to be examined for double free scenarios. The presence of functions like `___Block_byref_object_dispose_.59` and `___Block_byref_object_release_.58` further emphasizes the need to analyze block object management for potential double frees. However, without more specific diff snippets, it is difficult to pinpoint the exact location and nature of the double free vulnerability. Therefore, the analysis report will focus on the areas of concern and the vulnerable functions will be left empty.

Affected Binary: CoreAudio

Vulnerability Type: Double Free

Function: SetPropertyData

Unified Diff Snippet

```diff
- old code line
+ new code line
```

Agent Analysis

Root Cause: Changes in property setting logic, possibly leading to double release of allocated memory.

Fix: Improved memory management.

Vulnerability Class: Double Free

CWE Mapping: CWE-415

MacOS Sandbox Escape via Double Free in coreaudiod/CoreAudio Framework

The following crash is generated:

```
Process 71272 stopped
* thread #19, queue = 'com.apple.audio.system-event', stop reason = EXC_BAD_ACCESS (code=1, address=0x7fc8c9ca728)
    frame #0: 0x00007ff808ab03bf libobjc.A.dylib`objc_release + 31
libobjc.A.dylib`objc_release:
->  0x7ff808ab03bf <+31>: mov    rsi, qword ptr [rdx + 0x20]
    0x7ff808ab03c3 <+35>: test   sil, 0x4
    0x7ff808ab03c7 <+39>: je     0x7ff808ab0415 ; <+117>
    0x7ff808ab03c9 <+41>: test   al, 0x1
(lldb) bt
* thread #19, queue = 'com.apple.audio.system-event', stop reason = EXC_BAD_ACCESS (code=1, address=0x7fc8c9ca728)
  * frame #0: 0x00007ff808ab03bf libobjc.A.dylib`objc_release + 31
    frame #1: 0x00007ff80b6421a7 CoreAudio`CACFDictionary::~CACFDictionary() + 23
    frame #2: 0x00007ff80b75823c CoreAudio`HALS_Object_SetPropertyData_DPList + 1715
    frame #3: 0x00007ff80bbdddc8 CoreAudio`_XObject_SetPropertyData_DPList + 177
    frame #4: 0x00007ff80bbe1c81 CoreAudio`HALB_MIGServer_server + 84
    frame #5: 0x00007ff808cf1032 libdispatch.dylib`dispatch_mig_server + 362
    frame #6: 0x00007ff80b8b82ed CoreAudio`invocation function for block in
AMCP::Utility::Dispatch_Queue::install_mig_server(unsigned int, unsigned int, unsigned int (*)(mach_msg_header_t*,
mach_msg_header_t*), bool, bool) + 42
    frame #7: 0x00007ff808cd67e2 libdispatch.dylib`_dispatch_client_callout + 8
```

CVE-2025-31325 - Audio - Double Free

- Found relevant binary out of 3000+ updated
- Could not positively identify the vulnerable function
- But! Seems to have identified a function related to the crash that *might* have had the issue
- 100+ unified diffs required the last call to use a frontier model
  - Could be "context engineered" to chunk each diff


CVE-2025-31201
RPAC (Return Pointer Authentication Code)

RPAC

Available for: iPhone XS and later, iPad Pro 13-inch, iPad Pro 12.9-inch 3rd generation and later, iPad Pro 11-inch 1st generation and later, iPad Air 3rd generation and later, iPad 7th generation and later, and iPad mini 5th generation and later

Impact: An attacker with arbitrary read and write capability may be able to bypass Pointer Authentication. Apple is aware of a report that this issue may have been exploited in an extremely sophisticated attack against specific targeted individuals on iOS.

Description: This issue was addressed by removing the vulnerable code.

CVE-2025-31201: Apple

Return Pointer Authentication Code

Overview

The arm64e architecture introduces pointer authentication codes (PACs) to detect and guard against unexpected changes to pointers in memory. The addition of pointer authentication is transparent to most apps because the compiler manages the process. In rare cases — for example, if your app manipulates the stack directly, or if you pass pointers between C++ and Objective-C++ — you might have to adjust your code to work with PACs.

Pointer authentication works by offering a special CPU instruction to add a cryptographic signature — or PAC — to unused high-order bits of a pointer before storing the pointer. Another instruction removes and authenticates the signature after reading the pointer back from memory. Any change to the stored value between the write and the read invalidates the signature. The CPU interprets authentication failure as memory corruption and sets a high-order bit in the pointer, making the pointer invalid and causing the app to crash.

Seek Binary Truth

Use Patch Diffs to Understand New Code or Features

Selected device: **iPhone13,3**

Used 5. Finding Candidate Binaries

Candidate Binaries(8):

CoreServices, AudioCodecs, AssistantServices, WatchListKit, Siri, appstored, IPConfiguration, libRPAC.dylib

Used 6. Correlating Binary with Agent

🧠 **Reasoning:**

Reasoning

The CVE metadata indicates the vulnerable module is “RPAC”. The only candidate binary provided is `machos/arm64e/usr/lib/libRPAC.dylib`, which directly matches the module name and resides in a location consistent with Apple’s library layout for iOS. Although the symbol extraction tool did not return data (likely due to file access issues), the filename alone provides strong evidence that this binary contains the vulnerable code referenced by the CVE. Therefore, we assign it a high confidence score of 0.95.

Relevant Binaries (score ≥ threshold):

- ✅ `libRPAC.dylib` (score: 0.95)

Preview: OTHER SCORED CANDIDATES

✅ Completed 1/1: `libRPAC.dylib`

Modified Functions in `libRPAC.dylib` :(15)

- `_interposed_dispatch_semaphore_wait`
- `____Z22initializePrimitiveMapv_block_invoke`
- `_updateAndGetWaiterCountValue`
- `_updatePrimitiveWaiterQoSInfo`
- `_initializeNSConditionSwizzling`
- `_findPrimitiveInfoNoAssert`
- `__replacement_NSCondition_broadcast`
- `_interposed_dispatch_group_wait`
- `_updatePrimitiveSignallerQoSInfoUnconditionally`
- `findSemaphoreInfo`
- `__replacement_NSCondition_wait`
- `__replacement_NSCondition_waitUntilDate`
- `_qosWaiterSignallerInvariantCheck`
- `_updatePrimitiveSignallerQoSInfo`
- `__replacement_NSCondition_signal`

By inspecting the Ghidra JSON diffs, we identified several functions where the original PA-related checks were removed or replaced with no-op stubs. These changes are consistent with a strategy to eliminate the code path that could be exploited to subvert PA. The most relevant modifications involve:

- Interposed dispatch semaphore and group wait functions – the original PA guard was stripped.
- Replacement NSCondition methods – the PA validation logic was removed, leaving only the core functionality.
- Helper functions (`_qosWaiterSignallerInvariantCheck`, `_updatePrimitiveSignallerQoSInfo`) that previously performed PA checks were either simplified or replaced with stubs.

Affected Binary: libRPAC.dylib

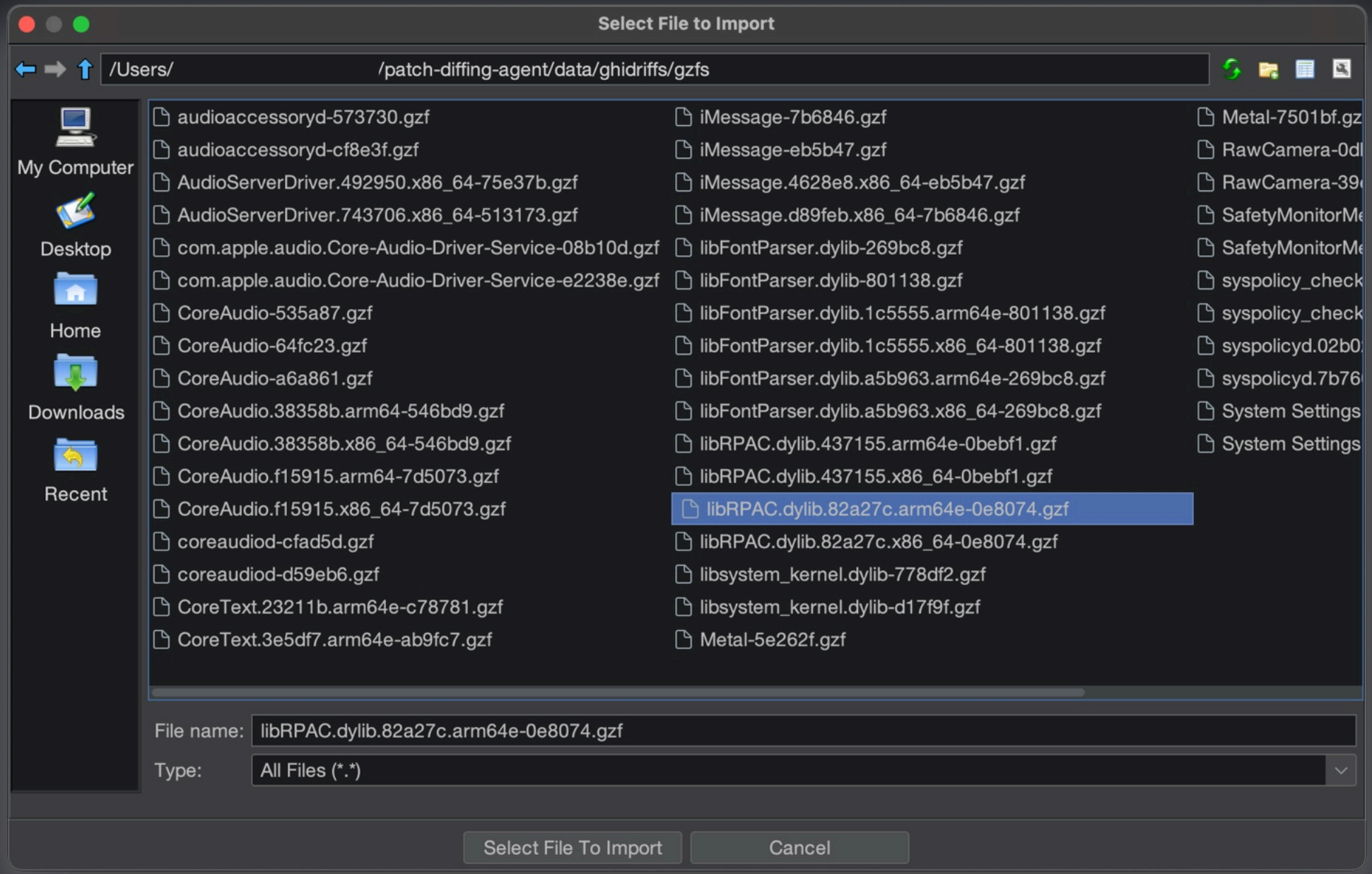
Vulnerability Type: Privilege Escalation (Pointer Authentication Bypass)

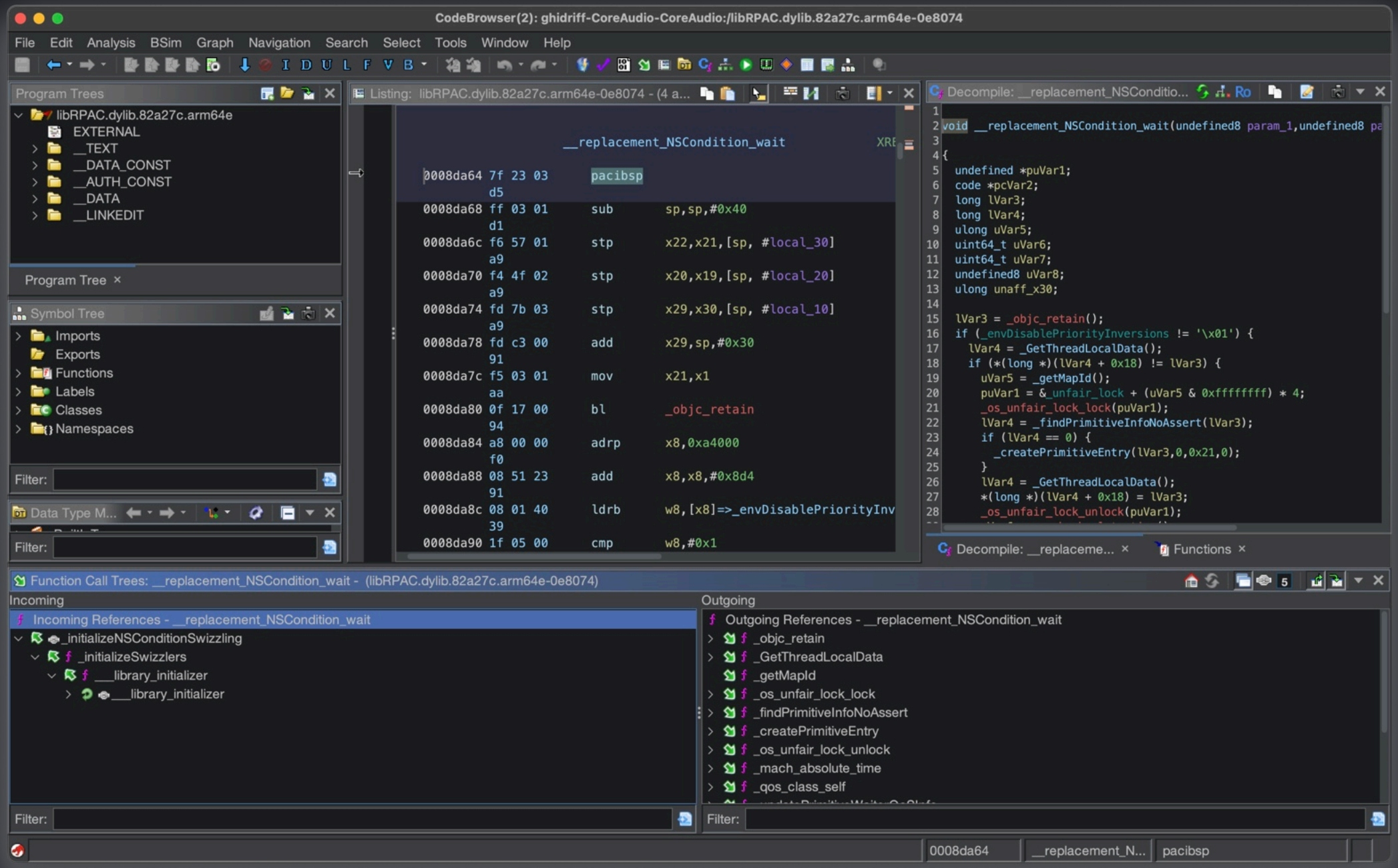
Function: `_interposed_dispatch_semaphore_wait`

Unified Diff Snippet

diff

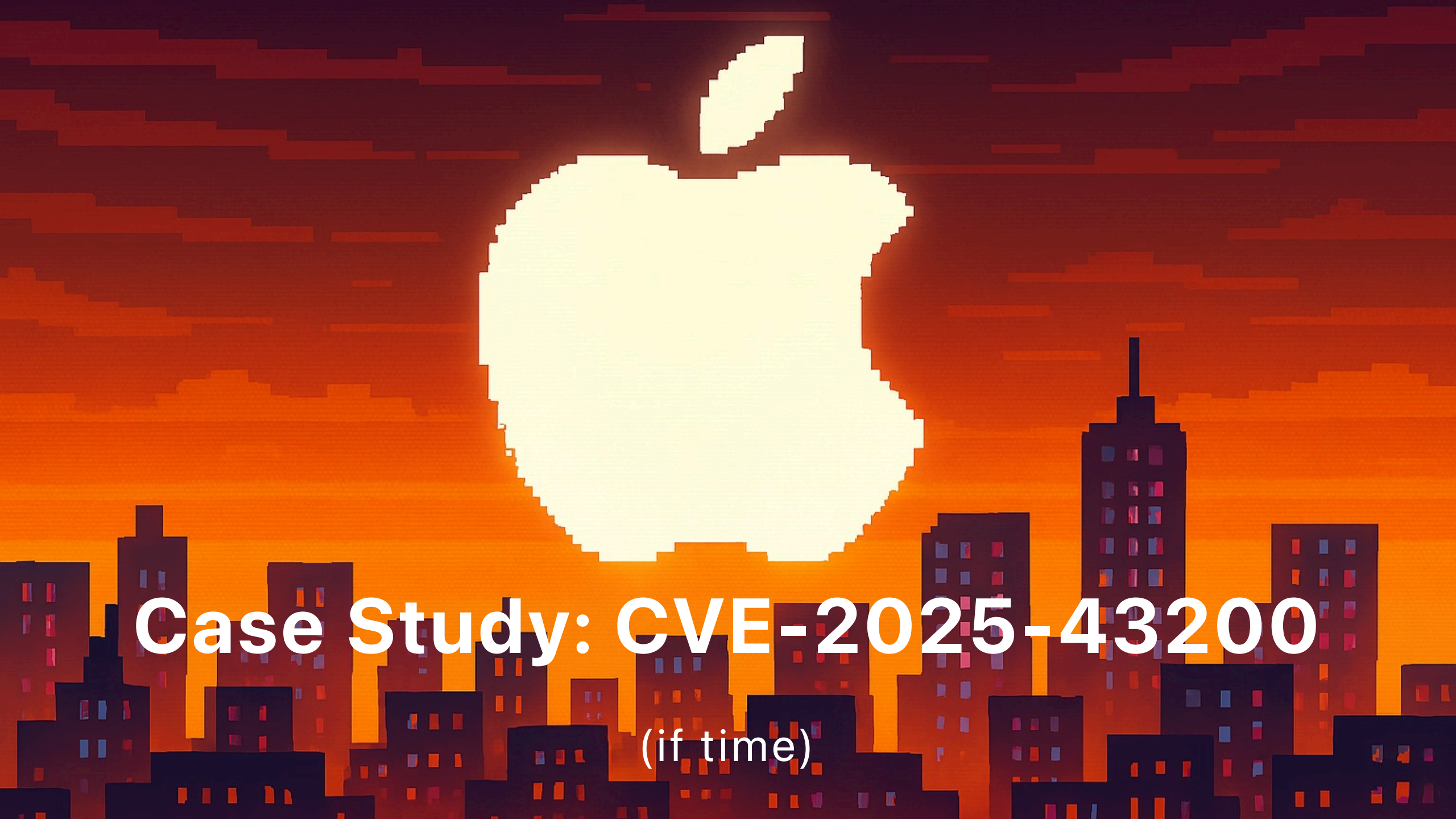
```
-   if (_envDisablePriorityInversions != '\x01') {
+   // PA guard removed — direct call to dispatch_semaphore_wait
    lVar4 = _dispatch_semaphore_wait(param_1,param_2);
```


Select File to Import
/Users/ /patch-diffing-agent/data/ghidriffs/gzfs
My Computer
Desktop
Home
Downloads
Recent
audioaccessoryd-573730.gzf
audioaccessoryd-cf8e3f.gzf
AudioServerDriver.492950.x86_64-75e37b.gzf
AudioServerDriver.743706.x86_64-513173.gzf
com.apple.audio.Core-Audio-Driver-Service-08b10d.gzf
com.apple.audio.Core-Audio-Driver-Service-e2238e.gzf
CoreAudio-535a87.gzf
CoreAudio-64fc23.gzf
CoreAudio-a6a861.gzf
CoreAudio.38358b.arm64-546bd9.gzf
CoreAudio.38358b.x86_64-546bd9.gzf
CoreAudio.f15915.arm64-7d5073.gzf
CoreAudio.f15915.x86_64-7d5073.gzf
coreaudiod-cfad5d.gzf
coreaudiod-d59eb6.gzf
CoreText.23211b.arm64e-c78781.gzf
CoreText.3e5df7.arm64e-ab9fc7.gzf
iMessage-7b6846.gzf
iMessage-eb5b47.gzf
iMessage.4628e8.x86_64-eb5b47.gzf
iMessage.d89feb.x86_64-7b6846.gzf
libFontParser.dylib-269bc8.gzf
libFontParser.dylib-801138.gzf
libFontParser.dylib.1c5555.arm64e-801138.gzf
libFontParser.dylib.1c5555.x86_64-801138.gzf
libFontParser.dylib.a5b963.arm64e-269bc8.gzf
libFontParser.dylib.a5b963.x86_64-269bc8.gzf
libRPAC.dylib.437155.arm64e-0bebf1.gzf
libRPAC.dylib.437155.x86_64-0bebf1.gzf
libRPAC.dylib.82a27c.arm64e-0e8074.gzf
libRPAC.dylib.82a27c.x86_64-0e8074.gzf
libsystem_kernel.dylib-778df2.gzf
libsystem_kernel.dylib-d17f9f.gzf
Metal-5e262f.gzf
Metal-7501bf.gz
RawCamera-0d
RawCamera-39
SafetyMonitorMe
SafetyMonitorMe
syspolicy_check
syspolicy_check
syspolicyd.02b0
syspolicyd.7b76
System Settings
System Settings
File name: libRPAC.dylib.82a27c.arm64e-0e8074.gzf
Type: All Files (*.*)
Select File To Import
Cancel


CodeBrowser(2): ghidriff-CoreAudio-CoreAudio:/libRPAC.dylib.82a27c.arm64e-0e8074
File Edit Analysis BSim Graph Navigation Search Select Tools Window Help
Program Trees
libRPAC.dylib.82a27c.arm64e
EXTERNAL
__TEXT
__DATA_CONST
__AUTH_CONST
__DATA
__LINKEDIT
Program Tree ×
Symbol Tree
Imports
Exports
Functions
Labels
Classes
Namespaces
Filter:
Data Type M...
Filter:
Listing: libRPAC.dylib.82a27c.arm64e-0e8074 - (4 a...
__replacement_NSCondition_wait XRE
0008da64 7f 23 03 d5 pacibsp
0008da68 ff 03 01 d1 sub sp,sp,#0x40
0008da6c f6 57 01 a9 stp x22,x21,[sp, #local_30]
0008da70 f4 4f 02 a9 stp x20,x19,[sp, #local_20]
0008da74 fd 7b 03 a9 stp x29,x30,[sp, #local_10]
0008da78 fd c3 00 91 add x29,sp,#0x30
0008da7c f5 03 01 aa mov x21,x1
0008da80 0f 17 00 94 bl _objc_retain
0008da84 a8 00 00 f0 adrp x8,0xa4000
0008da88 08 51 23 91 add x8,x8,#0x8d4
0008da8c 08 01 40 39 ldrb w8,[x8]=>_envDisablePriorityInv
0008da90 1f 05 00 cmp w8,#0x1
Decompile: __replacement_NSConditio...
void __replacement_NSCondition_wait(undefined8 param_1,undefined8 pa
{
undefined *puVar1;
code *pcVar2;
long lVar3;
long lVar4;
ulong uVar5;
uint64_t uVar6;
uint64_t uVar7;
undefined8 uVar8;
ulong unaff_x30;
lVar3 = _objc_retain();
if (_envDisablePriorityInversions != '\x01') {
lVar4 = _GetThreadLocalData();
if (*(long *)(lVar4 + 0x18) != lVar3) {
uVar5 = _getMapId();
puVar1 = &_unfair_lock + (uVar5 & 0xffffffff) * 4;
_os_unfair_lock_lock(puVar1);
lVar4 = _findPrimitiveInfoNoAssert(lVar3);
if (lVar4 == 0) {
_createPrimitiveEntry(lVar3,0,0x21,0);
}
lVar4 = _GetThreadLocalData();
*(long *)(lVar4 + 0x18) = lVar3;
_os_unfair_lock_unlock(puVar1);
Decompile: __replaceme... ×
Functions ×
Function Call Trees: __replacement_NSCondition_wait - (libRPAC.dylib.82a27c.arm64e-0e8074)
Incoming
Incoming References - __replacement_NSCondition_wait
_initializeNSConditionSwizzling
_initializeSwizzlers
___library_initializer
___library_initializer
Outgoing
Outgoing References - __replacement_NSCondition_wait
_objc_retain
_GetThreadLocalData
_getMapId
_os_unfair_lock_lock
_findPrimitiveInfoNoAssert
_createPrimitiveEntry
_os_unfair_lock_unlock
_mach_absolute_time
_qos_class_self
Filter:
Filter:
0008da64
__replacement_N...
pacibsp

CVE-2025-31201 - RPAC - Priv Esc

- Ideal to use diffing to learn about new research areas + features
- Ghidriff exports GZF files perfect for follow on static analysis
- The vulnerable functions identified for this one are questionable, but still plausible.



Case Study: CVE-2025-43200

(if time)



clearbluejar @clearbluejar · Oct 10

Next week at #OBTS v8, I'll take this one step further in my talk: "Reverse Engineering Apple Security Updates"

- Automating patch triage with agentic diffing
- Case studies like CVE-2025-43400 (FontParser)
- Compressing analysis from days → minutes

2 2 15 1.5K



jaybird1291
@jaybird1291

Definitely looking forward to your talk! I tried patch diffing for the first time with CVE-2025-43200 (Paragon spyware related) back in July, and I loved it, though it's really time consuming. I hope we'll be able to chat at OBTS 😊

Messages

Available for: iPhone XS and later, iPad Pro 13-inch, iPad Pro 12.9-inch 3rd generation and later, iPad Pro 11-inch 1st generation and later, iPad Air 3rd generation and later, iPad 7th generation and later, and iPad mini 5th generation and later

Impact: A logic issue existed when processing a maliciously crafted photo or video shared via an iCloud Link. Apple is aware of a report that this issue may have been exploited in an extremely sophisticated attack against specific targeted individuals.

Description: This issue was addressed with improved checks.

CVE-2025-43200: Apple

Entry added June 11, 2025

Used 3. Selecting Platform Agent

Selected Agent: ApplePlatformAgent

Used 4. Choosing Device for Candidate Binaries

Selected: 📱 iPhone13,3

Selected device: **iPhone13,3**

Used 5. Finding Candidate Binaries

Candidate Binaries(9):
ProactiveSummarization, AXSpringBoardServerInstance, WritingToolsUIService, identityservicesd, iMessage, SafetyMonitorMessages, powerd, profiled, searchpartyd

How does this compare??

Affected Binary: System/Library/Messages/PlugIns/iMessage.imservice/iMessage

Vulnerability Type: Logic Error / Improper Validation

Function:

```
_reAttemptMessageDeliveryForGUID:toIdentifier:fromIdentifier:fromIDSID:isReflectio
n:shouldShowError:cacheFlushError:imdAccount:
```

Reasoning:

Reasoning

The diff shows extensive changes to the function `_reAttemptMessageDeliveryForGUID:toIdentifier:fromIdentifier:fromIDSID:isRejection:shouldShowError:cacheFlushError:imdAccount:` in the iMessage service binary. The modifications replace several internal helper calls with new variants (e.g., `FUN_0009efa0` instead of `FUN_0009f000`) and add additional guard checks around message ownership, retry eligibility, and client version compatibility. These changes correspond to the CVE description that a logic issue existed when processing maliciously crafted media shared via an iCloud link. The added checks prevent the service from attempting to resend messages that were not originally sent by the local user or that are too old, thereby mitigating the risk of unintended message delivery or denial-of-service conditions.

Agent Analysis

Root Cause: The original implementation lacked checks to verify that a message being retried was actually sent by the local user and that it had not expired. This allowed maliciously crafted media links to trigger unintended retry logic, potentially leading to denial-of-service or privilege escalation scenarios.

Fix: Introduced additional guard clauses that:

- Verify message ownership before attempting a resend.
- Ensure the message is within an acceptable age window.
- Update helper function calls to newer, safer variants.

Vulnerability Class: Logic Error / Improper Validation

CWE Mapping: CWE-753 (Improper Check of Function Return Value) and CWE-730 (Improper Validation of Critical Information).

CVE-2025-43200 - Messages - Logic Error

- Binary seems to be identified correctly
- Plausible that these new checks reduce attacker control
- Accuracy? Need more info...

Does it always work??

- Hard to verify false positives
- Some patches are additional entitlements or fixed out the context
- Every determinstic tool must work well, diff fidelity is extermely important
- Symbols are necessary
- **Short answer, no.**

Hard to verify false positives

Kernel

Available for: iPhone XS and later, iPad Pro 12.9-inch 2nd generation and later, iPad Pro 10.5-inch, iPad Pro 11-inch 1st generation and later, iPad Air 3rd generation and later, iPad 6th generation and later, and iPad mini 5th generation and later

Impact: An app may be able to cause unexpected system termination or write kernel memory

Description: A memory corruption vulnerability was addressed with improved locking.

CVE-2024-23265: Xinru Chi of Pangu Lab

Entry added March 7, 2024

Keyword "likely"....

Likely Vulnerable Conditions In IOS 17.3.1

At this point, the likely Vulnerable Conditions in 17.3.1 could be the following:

1. ***(long *)(uVar3 + (uVar1 & 0xfffffff8))** may return **-1**:
– Not validated
– Returned directly to the caller

2. If backing data is modified concurrently:
– A stale **-1** value can be returned
– Race with deallocation leads to UAF or invalid pointer exposure

3. User-controlled indexing (`**param_1[1]**`) drives the loop

This points to a potential race condition, allowing user-controlled conditions to bypass safety checks and return malicious or corrupted memory references. If this function is invoked via an **IOUserClient::externalMethod** path or equivalent, a malicious actor could:
– Race updates to the backing memory structure
– Force return of -1
– Trigger kernel memory corruption or unauthorized pointer access

Every deterministic tool must work well

Diff fidelity is extermely important

```
2025-10-01 13:46:59,565 ghidriff WARNING  Failed to decompile libRPAC.dylib.437155.
arm64e - .ProgramDB _qosWaiterSignallerInvariantCheck : Error: Decompile error:
Low-level Error: Forced merge caused intersection
2025-10-01 13:46:59,567 ghidriff WARNING  Failed to decompile libRPAC.dylib.82a27c.
arm64e - .ProgramDB _qosWaiterSignallerInvariantCheck : Error: Decompile error:
Low-level Error: Forced merge caused intersection
```

Symbols are necessary

Currently lacking KEXTs symbols

% file kexts/com.apple.*

```
kexts/com.apple.driver.AppleSMC:                          Mach-O 64-bit kext bundle arm64e
kexts/com.apple.driver.AppleSMCWirelessCharger:           Mach-O 64-bit kext bundle arm64e
kexts/com.apple.driver.AppleSPIMC:                        Mach-O 64-bit kext bundle arm64e
kexts/com.apple.driver.AppleSPMI:                         Mach-O 64-bit kext bundle arm64e
kexts/com.apple.driver.AppleSPMIPMU:                      Mach-O 64-bit kext bundle arm64e
kexts/com.apple.driver.AppleSPU:                          Mach-O 64-bit kext bundle arm64e
kexts/com.apple.driver.AppleSPURose:                      Mach-O 64-bit kext bundle arm64e
kexts/com.apple.driver.AppleSPUSphere:                    Mach-O 64-bit kext bundle arm64e
kexts/com.apple.driver.AppleSSE:                          Mach-O 64-bit kext bundle arm64e
kexts/com.apple.driver.AppleStockholmControl:             Mach-O 64-bit kext bundle arm64e
kexts/com.apple.driver.AppleT7000USBOTGDevice:            Mach-O 64-bit kext bundle arm64e
kexts/com.apple.driver.AppleT8020DART:                    Mach-O 64-bit kext bundle arm64e
kexts/com.apple.driver.AppleT8101:                        Mach-O 64-bit kext bundle arm64e
kexts/com.apple.driver.AppleT8101CLPCv3:                  Mach-O 64-bit kext bundle arm64e
kexts/com.apple.driver.AppleT8101PCIe:                    Mach-O 64-bit kext bundle arm64e
kexts/com.apple.driver.AppleT8101PMGR:                    Mach-O 64-bit kext bundle arm64e
kexts/com.apple.driver.AppleTemperatureSensor:            Mach-O 64-bit kext bundle arm64e
```

% nm kexts/com.apple.*

```
kexts/com.apple.iokit.IOMobileGraphicsFamily-DCP: no symbols
kexts/com.apple.iokit.IONetworkingFamily: no symbols
kexts/com.apple.iokit.IONVMeFamily: no symbols
kexts/com.apple.iokit.IOPCIFamily: no symbols
kexts/com.apple.iokit.IOReportFamily: no symbols
kexts/com.apple.iokit.IOSCSIArchitectureModelFamily: no symbols
kexts/com.apple.iokit.IOSCSIBlockCommandsDevice: no symbols
kexts/com.apple.iokit.IOSerialFamily: no symbols
kexts/com.apple.iokit.IOSkywalkFamily: no symbols
kexts/com.apple.iokit.IOStorageFamily: no symbols
kexts/com.apple.iokit.IOStreamFamily: no symbols
kexts/com.apple.iokit.IOSurface: no symbols
kexts/com.apple.iokit.IOTimeSyncFamily: no symbols
kexts/com.apple.iokit.IOUSBDeviceFamily: no symbols
kexts/com.apple.iokit.IOUSBHostFamily: no symbols
kexts/com.apple.iokit.IOUSBMassStorageDriver: no symbols
kexts/com.apple.kec.Compression: no symbols
```

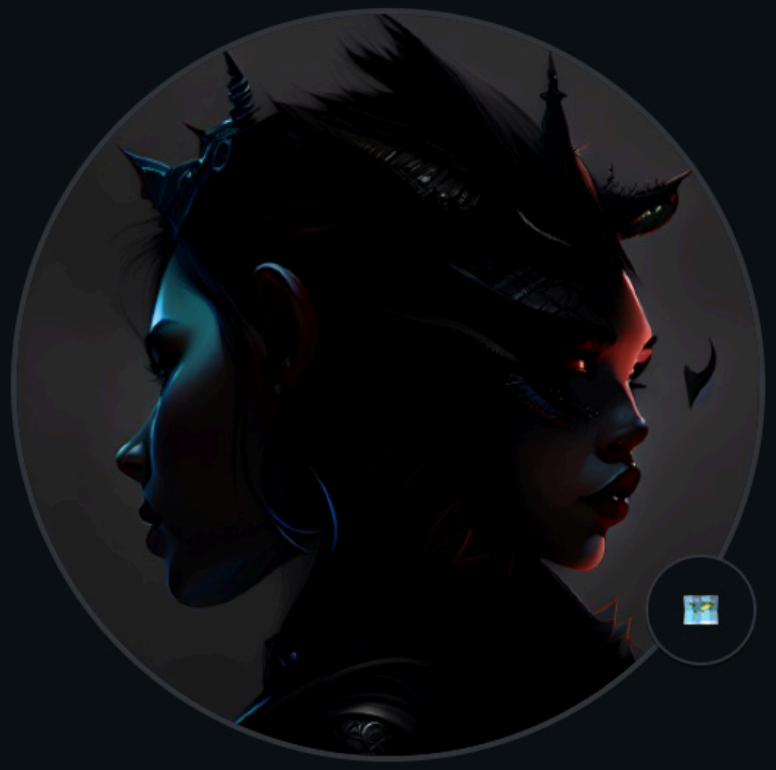
Used 6. Correlating Binary with Agent

🧠 **Reasoning:**

Reasoning

The CVE description indicates a memory-corruption vulnerability that was mitigated by improved locking mechanisms in the Kernel module. Among the candidate files, kernel extensions (kexts) and system libraries are the most relevant. Attempts to extract symbols from the kexts failed, likely due to missing or invalid binaries in the provided dataset. A successful symbol extraction from `dylibs/arm64e/usr/lib/system/libsystem_kernel.dylib` revealed several lock-related and memory-management functions (`_lock_set_destroy`, `_mach_vm_protect`, etc.) that align with the CVE's focus on locking improvements to prevent corruption. No other candidate produced usable symbol data, so `libsystem_kernel.dylib` is identified as the most plausible match.

https://github.com/blacktop/symbolicator



blacktop

Pinned

maliceio/malice Public archive

VirusTotal Wanna Be - Now with 100% more Hipster

Go 1.6k 261

ipsw Public

iOS/macOS Research Swiss Army Knife

Go 1.4k 111

go-macho Public

Package macho implements access to and creation of Mach-O object files.

Go 165 26

docker-ghidra Public

Ghidra Client/Server Docker Image

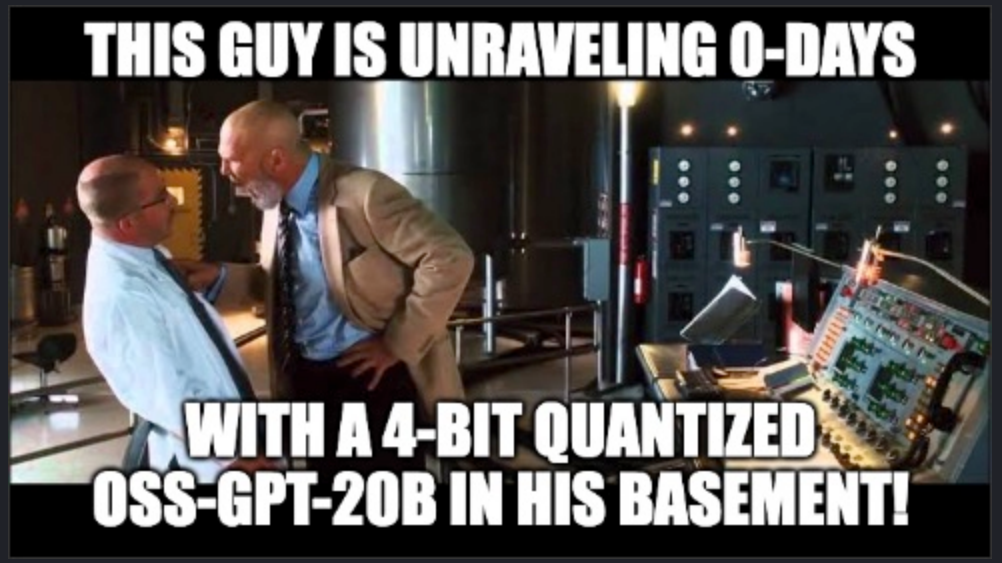
Dockerfile 189 54

Wrapping up....

Can you do this at home?

Built using open-source tools and frameworks


YOU HAVE THE LATEST FRONTIER MODELS


THIS GUY IS UNRAVELING 0-DAYS
WITH A 4-BIT QUANTIZED
OSS-GPT-20B IN HIS BASEMENT!

Roadmap

- **Expand Coverage**
  - Extend beyond Apple platform
  - Add support for KEXT symbolication
  - Add more (all?) devices
- **Improve Context Handling**
  - Engineer workflows for 100+ function diffs
  - Smarter chunking for large context

Roadmap

- **Deeper Analysis**
  - Add researcher insight (known mappings)
  - Introduce follow on analysis
  - Variant Hunting?
- **Community & Scale**
  - Host a public cloud version?

Summary

- RE ASU (the problem of opaque updates)
- Show the manual baseline (OBTSv7)
- Introduce the hybrid philosophy
  - (deterministic + agents) == Effective Agentic Diffing
- Walk through the toolchain and methodology
- Prove it live (demo + case studies)
- Roadmap

Questions?

Reverse Engineering Apple Security Updates

John McIntosh

contact: @clearbluejar

CL | CLEARSEC LABS

https://clearseclabs.com

