black hat®
BRIEFINGS

AUGUST 6-7, 2025
MANDALAY BAY / LAS VEGAS

BinWhisper: LLM-Driven Reasoning for Automated Vulnerability Discovery Behind Hall-of-Fame

Qinrun Dai Yifei Xie

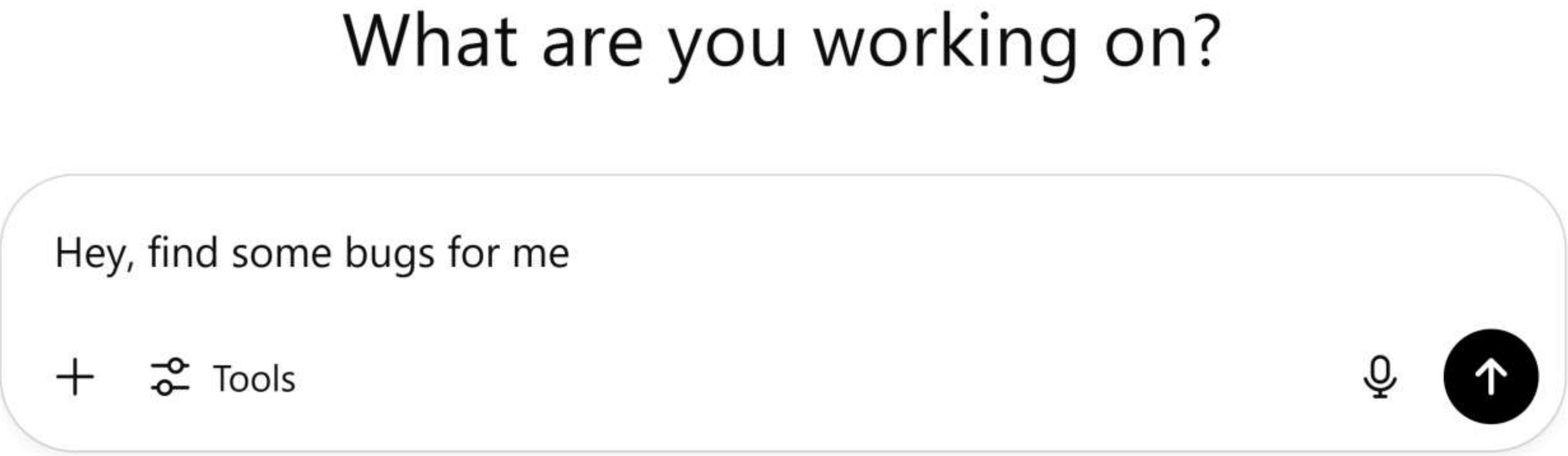
About Us

Qinrun Dai (@2st_ _ _)

- CS PhD student, research interest: exploit analysis & formalization
- Windows Security
- Exploitation Development
- Speaker at Black Hat USA 2024, Linux Security Summit Europe 2024

Yifei Xie(@iceflying2005)

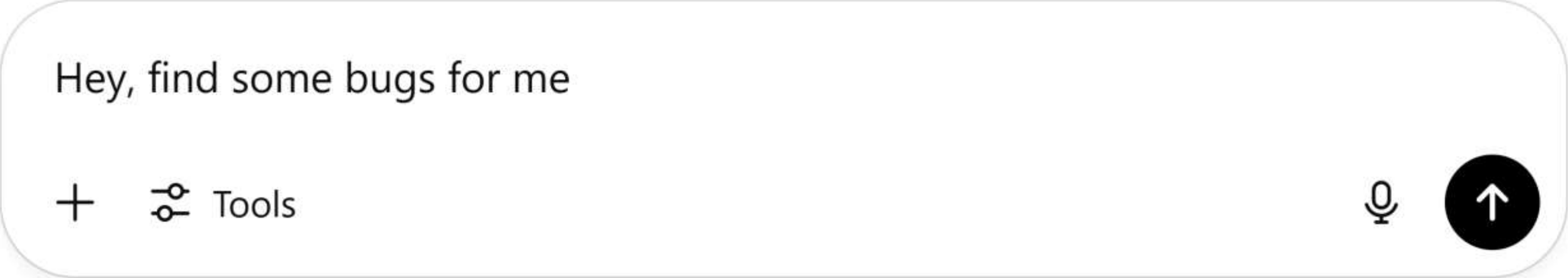
- Undergraduate student & Independent Security Researcher
- Binary Security, AI for vulnerability discovery
- Hall of Fame of the Samsung Mobile Security Rewards Program 2024



What are you working on?

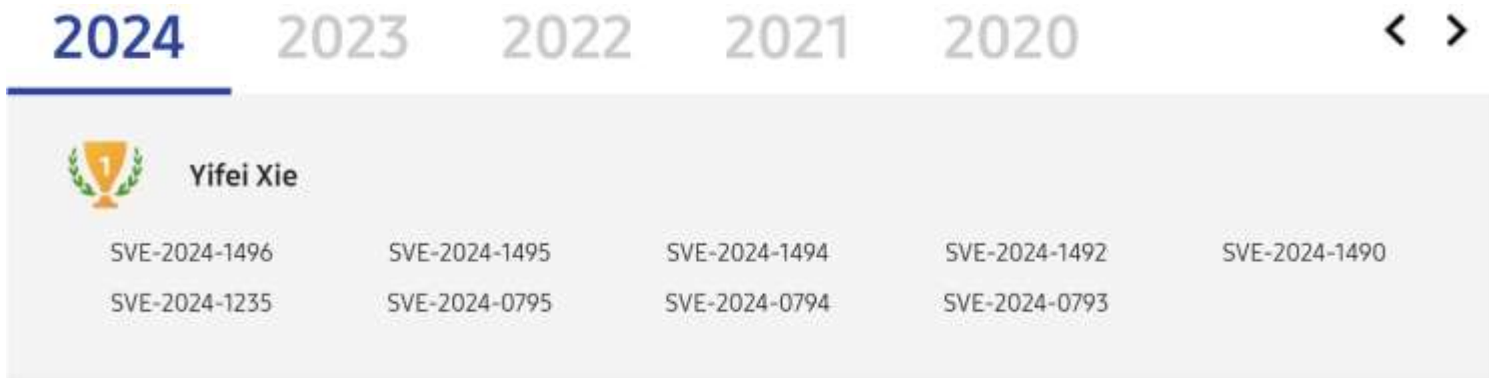
Hey, find some bugs for me

What are you working on?



Hey, find some bugs for me

Samsung Mobile Security



Yifei Xie

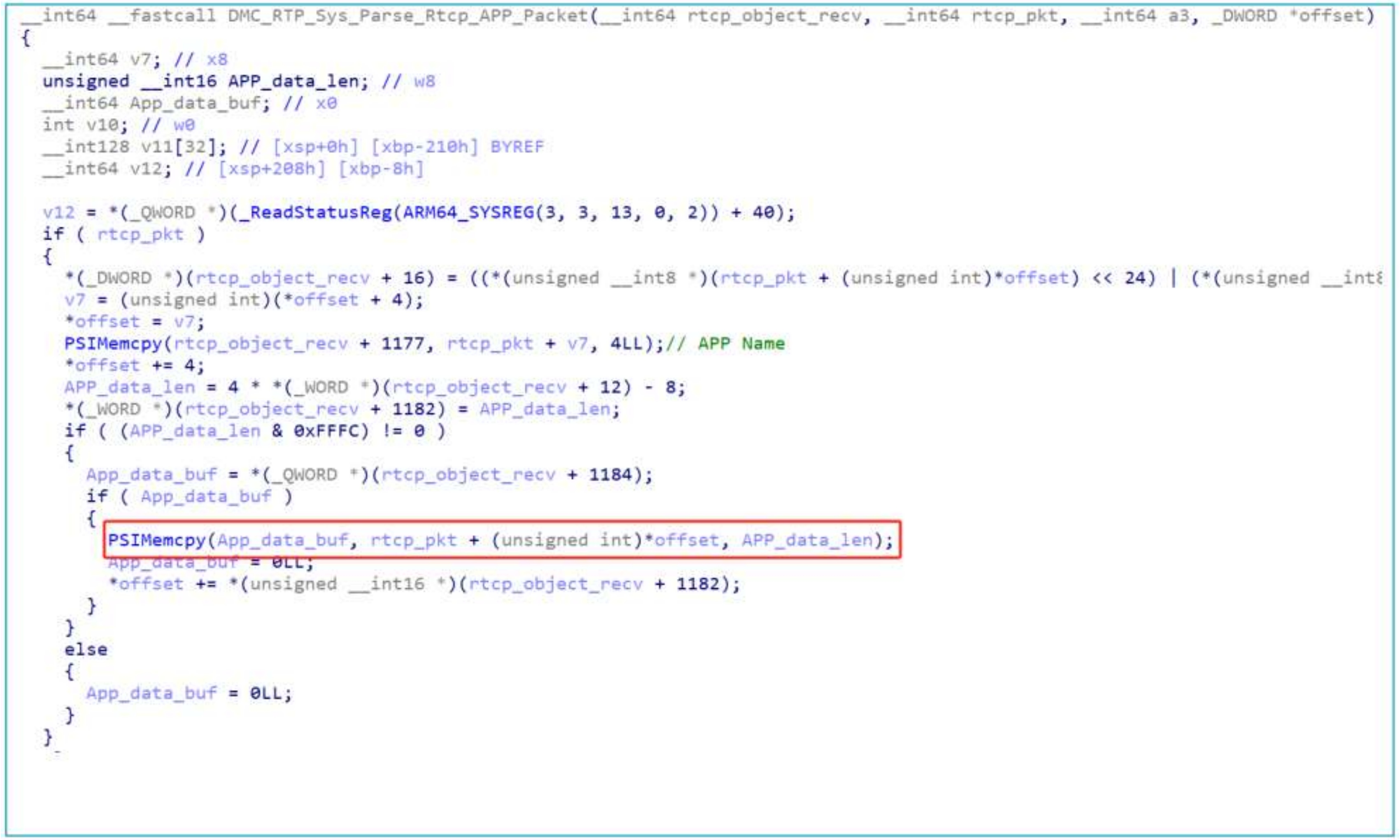
SVE-2024-1496 SVE-2024-1495 SVE-2024-1494 SVE-2024-1492 SVE-2024-1490
SVE-2024-1235 SVE-2024-0795 SVE-2024-0794 SVE-2024-0793

Agenda

- Manual Vulnerability Auditing: A Deep Dive into CVE-2024-34587
- Exploring LLMs’ Strengths in Code Reasoning
- Improving Code Analysis with LLMs via CVE-2024-34587
- Applying LLMs to Real-World Vulnerability Analysis

Manual Vulnerability Auditing: A Deep Dive into CVE-2024-34587

Quick Review: CVE-2024-34587 heap overflow of parsing app rtcp function



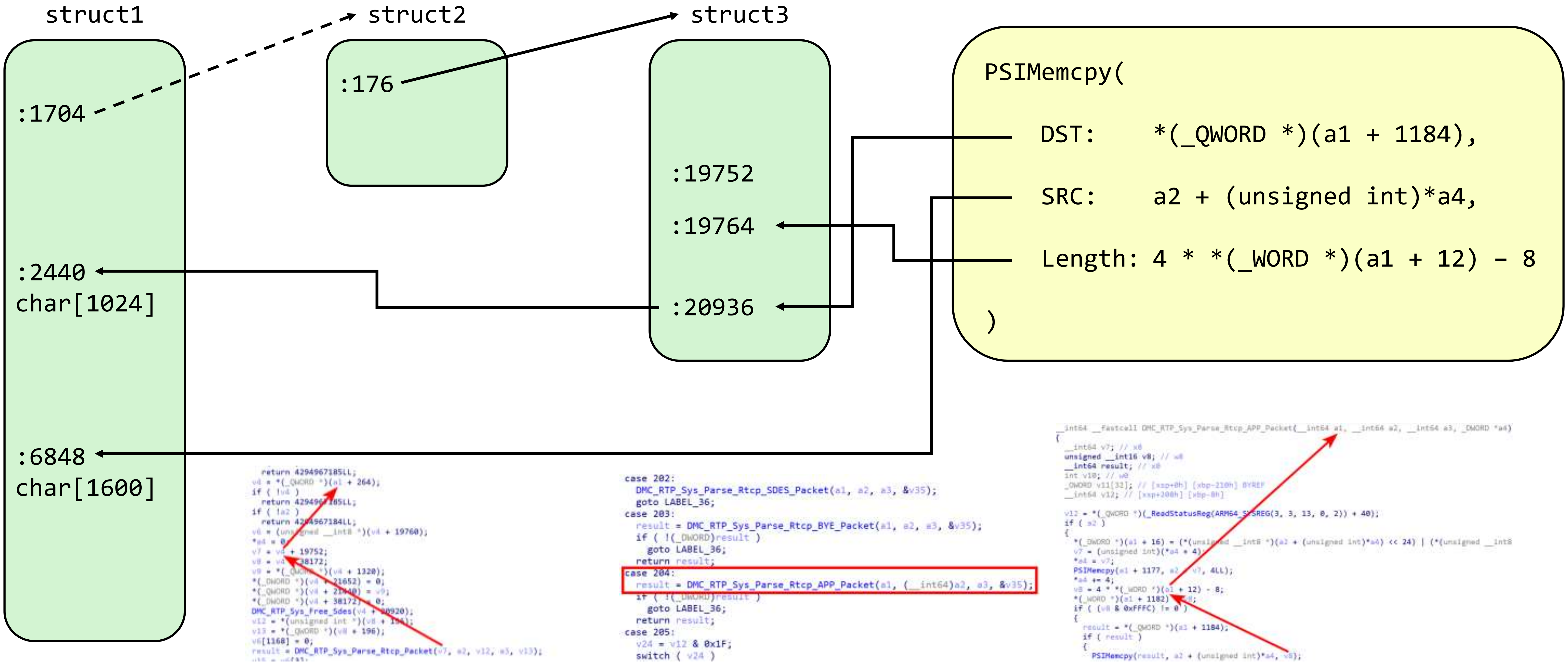
```
__int64 __fastcall DMC_RTP_Sys_Parse_Rtcp_APP_Packet(__int64 rtcp_object_recv, __int64 rtcp_pkt, __int64 a3, _DWORD *offset)
{
  __int64 v7; // x8
  unsigned __int16 APP_data_len; // w8
  __int64 App_data_buf; // x0
  int v10; // w0
  __int128 v11[32]; // [xsp+0h] [xbp-210h] BYREF
  __int64 v12; // [xsp+208h] [xbp-8h]

  v12 = *(_QWORD *)(_ReadStatusReg(ARM64_SYSREG(3, 3, 13, 0, 2)) + 40);
  if ( rtcp_pkt )
  {
    *(_DWORD *)(rtcp_object_recv + 16) = ((*(unsigned __int8 *)(rtcp_pkt + (unsigned int)*offset) << 24) | (*(unsigned __int
    v7 = (unsigned int)(*offset + 4);
    *offset = v7;
    PSIMemcpy(rtcp_object_recv + 1177, rtcp_pkt + v7, 4LL);// APP Name
    *offset += 4;
    APP_data_len = 4 * *(_WORD *)(rtcp_object_recv + 12) - 8;
    *(_WORD *)(rtcp_object_recv + 1182) = APP_data_len;
    if ( (APP_data_len & 0xFFFC) != 0 )
    {
      App_data_buf = *(_QWORD *)(rtcp_object_recv + 1184);
      if ( App_data_buf )
      {
        PSIMemcpy(App_data_buf, rtcp_pkt + (unsigned int)*offset, APP_data_len);
        App_data_buf = 0LL;
        *offset += *(unsigned __int16 *)(rtcp_object_recv + 1182);
      }
    }
    else
    {
      App_data_buf = 0LL;
    }
  }
```

This function is responsible for parsing rtcp app packets from the other end.

The three parameters of PSIMemcpy:

- `App_data_buf:` size is 1024 bytes
- `rtcp_pkt:` size is 1560 bytes
- `App_data_len:` value is 0-0xffff

Bug is easy, journey is hard.



```
PSIMemcpy(
    DST:     *(_QWORD *)(a1 + 1184),
    SRC:     a2 + (unsigned int)*a4,
    Length: 4 * *(_WORD *)(a1 + 12) - 8
)
```

```
  return 4294967185LL;
v4 = *(_QWORD *)(a1 + 264);
if ( !v4 )
  return 4294967185LL;
if ( !a2 )
  return 4294967184LL;
v6 = (unsigned __int8 *)(v4 + 19760);
*a4 = 0;
v7 = v4 + 19752;
v8 = v4 + 38172;
v9 = *(_QWORD *)(v4 + 1320);
*(_DWORD *)(v4 + 21652) = 0;
*(_QWORD *)(v4 + 21440) = v9;
*(_DWORD *)(v4 + 38172) = 0;
DMC_RTP_Sys_Free_Sdes(v4 + 20920);
v12 = *(unsigned int *)(v8 + 192);
v13 = *(_QWORD *)(v8 + 196);
v6[1168] = 0;
result = DMC_RTP_Sys_Parse_Rtcp_Packet(v7, a2, v12, a3, v13);
```

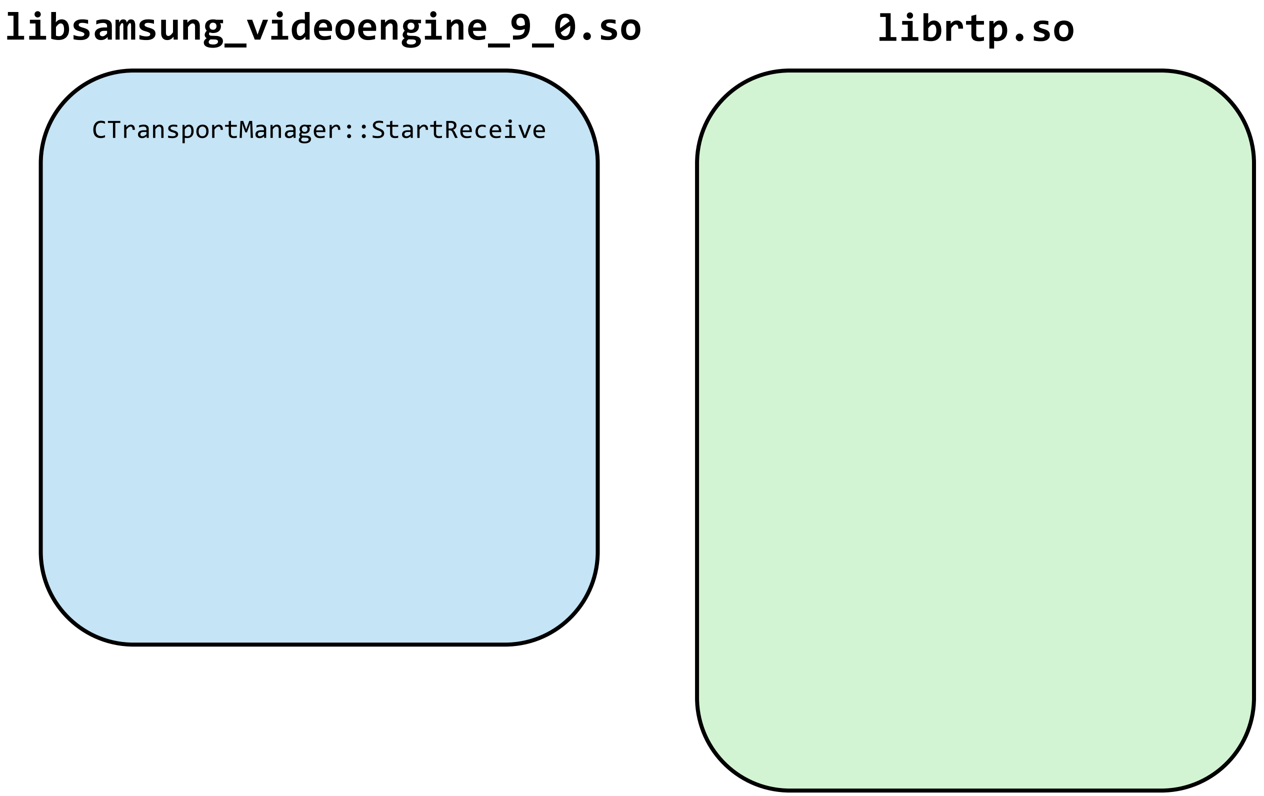
```
case 202:
  DMC_RTP_Sys_Parse_Rtcp_SDES_Packet(a1, a2, a3, &v35);
  goto LABEL_36;
case 203:
  result = DMC_RTP_Sys_Parse_Rtcp_BYE_Packet(a1, a2, a3, &v35);
  if ( !(_DWORD)result )
    goto LABEL_36;
  return result;
case 204:
  result = DMC_RTP_Sys_Parse_Rtcp_APP_Packet(a1, (__int64)a2, a3, &v35);
  if ( !(_DWORD)result )
    goto LABEL_36;
  return result;
case 205:
  v24 = v12 & 0x1F;
  switch ( v24 )
```

```
__int64 __fastcall DMC_RTP_Sys_Parse_Rtcp_APP_Packet(__int64 a1, __int64 a2, __int64 a3, _DWORD *a4)
{
  __int64 v7; // x8
  unsigned __int16 v8; // w8
  __int64 result; // x0
  int v10; // w0
  _QWORD v11[32]; // [xsp+0h] [xbp-210h] BYREF
  __int64 v12; // [xsp+200h] [xbp-8h]

  v12 = *(_QWORD *)(_ReadStatusReg(ARM64_SYSREG(3, 3, 13, 0, 2)) + 40);
  if ( a2 )
  {
    *(_DWORD *)(a1 + 16) = (*(unsigned __int8 *)(a2 + (unsigned int)*a4) << 24) | (*(unsigned __int8
    v7 = (unsigned int)(*a4 + 4);
    *a4 = v7;
    PSIMemcpy(a1 + 1177, a2 + v7, 4LL);
    *a4 += 4;
    v8 = 4 * *(_WORD *)(a1 + 12) - 8;
    *(_WORD *)(a1 + 1182) = v8;
    if ( (v8 & 0xFFFC) != 0 )
    {
      result = *(_QWORD *)(a1 + 1184);
      if ( result )
      {
        PSIMemcpy(result, a2 + (unsigned int)*a4, v8);
```

libsamsung_videoengine_9_0.so

librtp.so

`libsamsung_videoengine_9_0.so!CTransportManager::StartReceive(CTransportManager)`

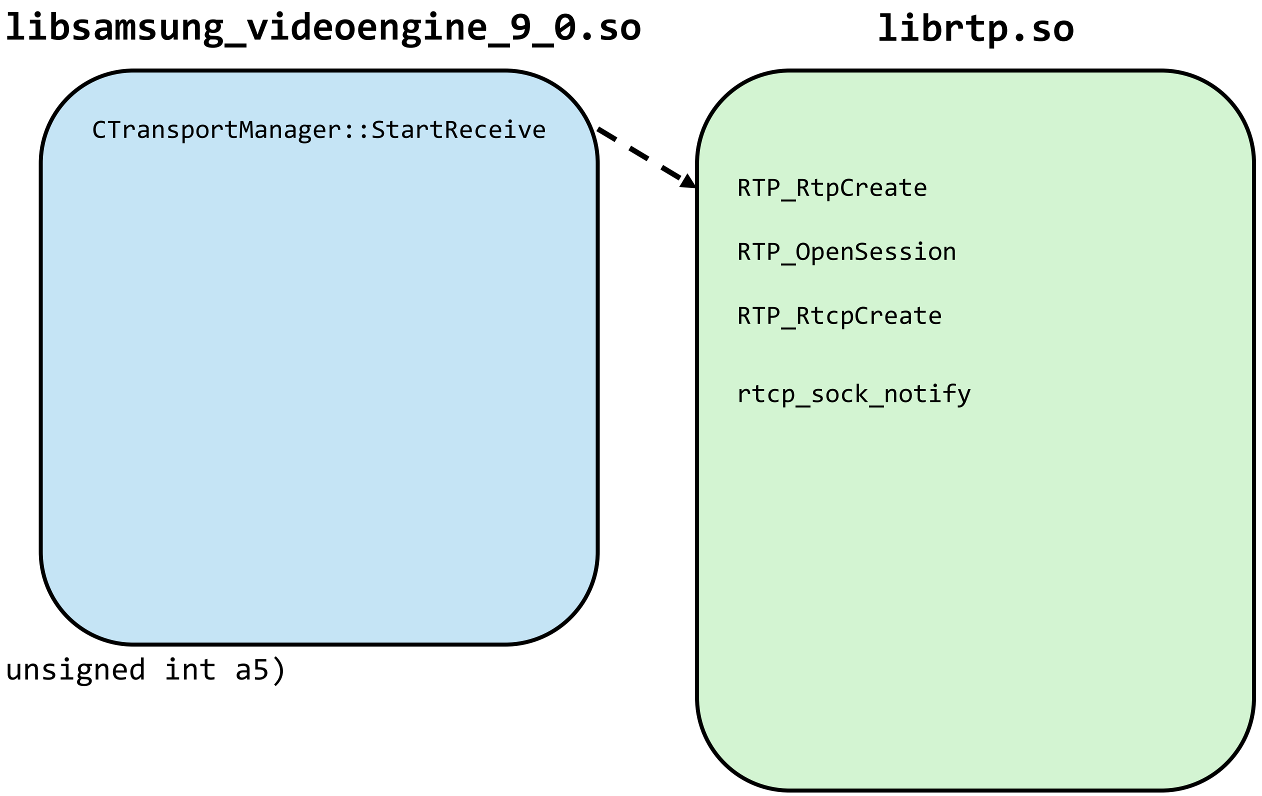


libsamsung_videoengine_9_0.so

CTransportManager::StartReceive

librtp.so

```
libsamsung_videoengine_9_0.so!CTransportManager::StartReceive(CTransportManager)

librtp.so!RTP_RtpCreate
          rtp_content = PSIMallocEx(216LL)
          librtp.so!RTP_OpenSession
              pRtpSession = PSIMallocEx(38384LL)
          *(rtp_content + 176) = pRtpSession
*(CTransportManager + 1704) = rtp_content
librtp.so!RTP_RtcpCreate
  rtcp_content = PSIMallocEx(1992LL)
        *(rtcp_content + 264) = *(rtp_content + 176) = pRtpSession
        PSIRegisterAsyncSelect(v27, rtcp_sock_notify, v6, 0LL, 1LL) )
          librtp.so!rtcp_sock_notify(__int64 a1, __int64 a2, int a3, _QWORD *a4, unsigned int a5)
```



libsamsung_videoengine_9_0.so

CTransportManager::StartReceive

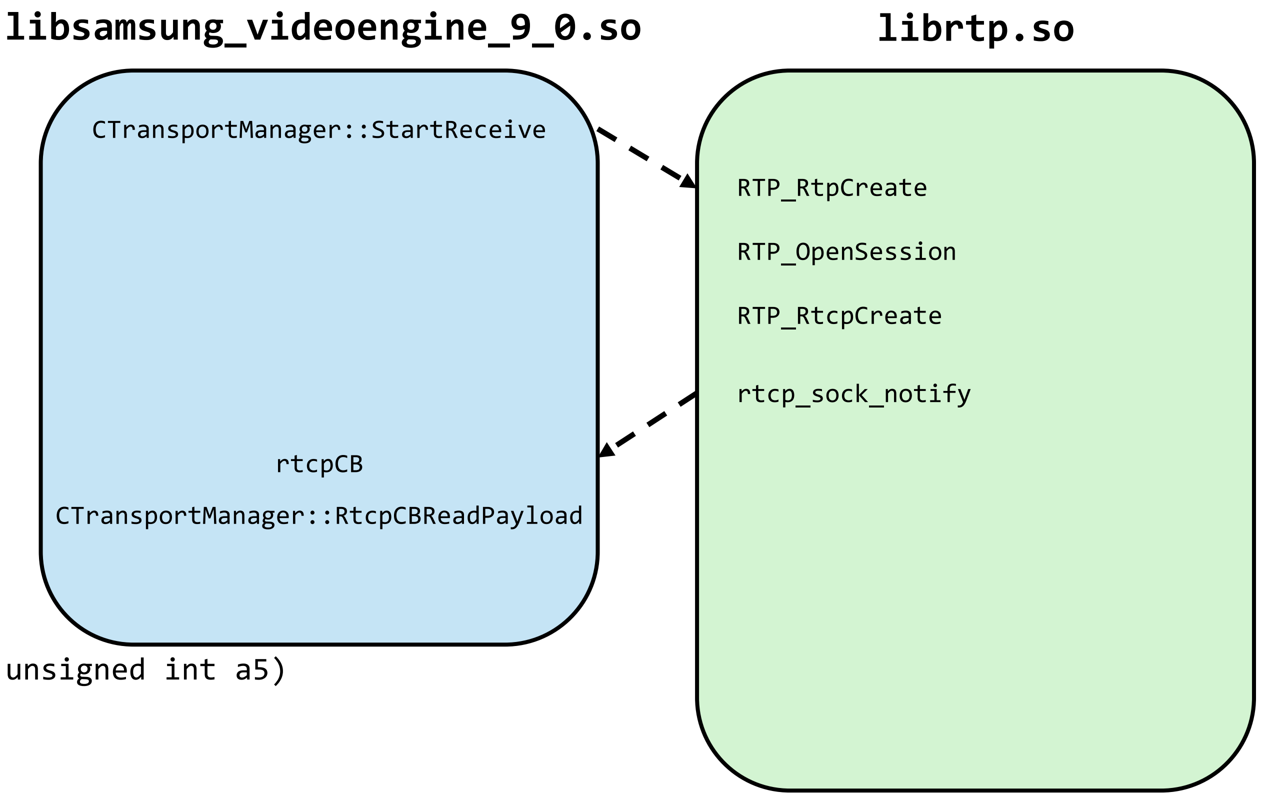
librtp.so

RTP_RtpCreate

RTP_OpenSession

RTP_RtcpCreate

rtcp_sock_notify


libsamsung_videoengine_9_0.so

CTransportManager::StartReceive

rtcpCB

CTransportManager::RtcpCBReadPayload

librtp.so

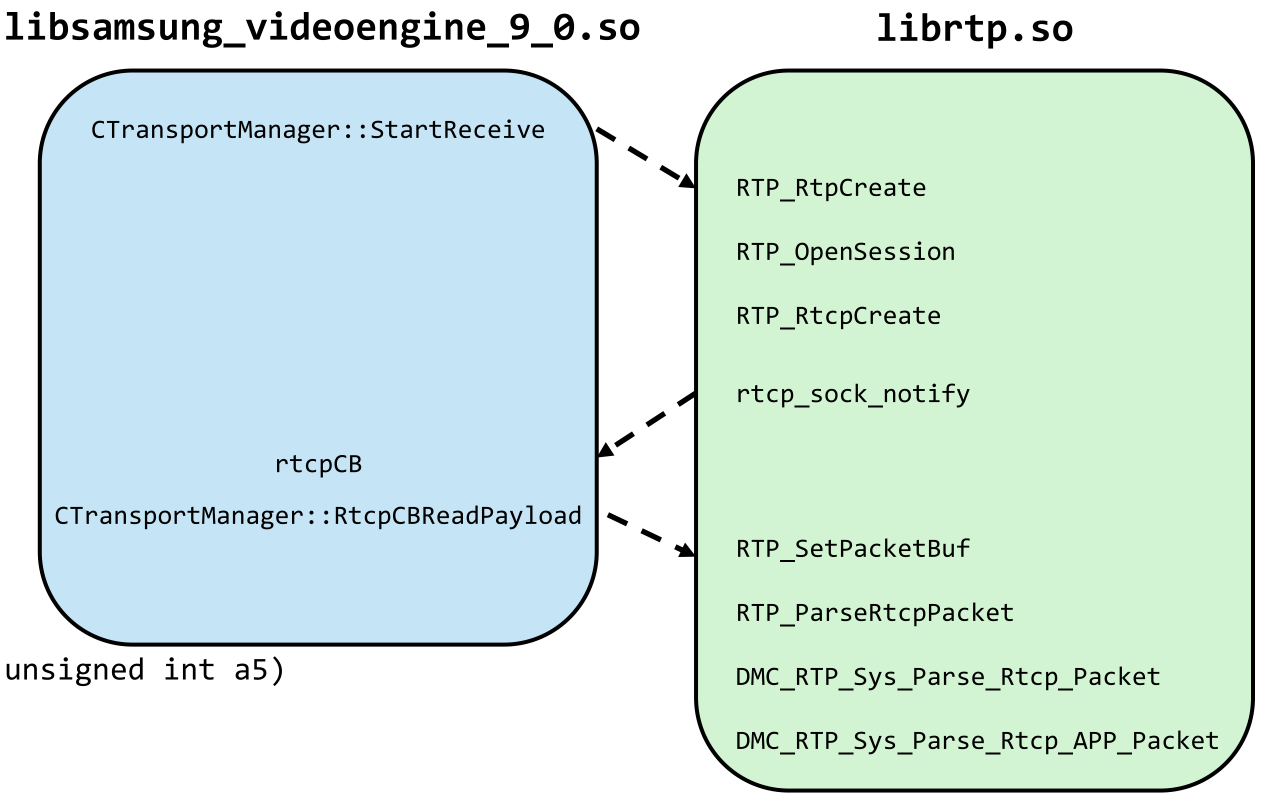
RTP_RtpCreate

RTP_OpenSession

RTP_RtcpCreate

rtcp_sock_notify

```
libsamsung_videoengine_9_0.so!CTransportManager::StartReceive(CTransportManager)

librtp.so!RTP_RtpCreate
          rtp_content = PSIMallocEx(216LL)
          librtp.so!RTP_OpenSession
              pRtpSession = PSIMallocEx(38384LL)
          *(rtp_content + 176) = pRtpSession
*(CTransportManager + 1704) = rtp_content
librtp.so!RTP_RtcpCreate
  rtcp_content = PSIMallocEx(1992LL)
        *(rtcp_content + 264) = *(rtp_content + 176) = pRtpSession
        PSIRegisterAsyncSelect(v27, rtcp_sock_notify, v6, 0LL, 1LL) )
          librtp.so!rtcp_sock_notify(__int64 a1, __int64 a2, int a3, _QWORD *a4, unsigned int a5)

                libsamsung_videoengine_9_0.so!rtcpCB(*a4, 1LL, a4[2], 0LL, &v16)
                  *a5 = (char *)CTransportManager + 6808
                *((_QWORD *)CTransportManager_0ff6808 + 4) = CTransportManager + 6808 + 40
                PSISocketRecvFrom(a4[3], CTransportManager + 6808 + 40, 1600LL, 0LL, v51)                       // rtcp_recv_buffer:6848
                libsamsung_videoengine_9_0.so!rtcpCB(*a4, 0LL, a4[2], v7, 0LL);
                  libsamsung_videoengine_9_0.so!CTransportManager::RtcpCBReadPayload(CTransportManager,CTransportManager_0ff6808)
```



libsamsung_videoengine_9_0.so

CTransportManager::StartReceive

rtcpCB

CTransportManager::RtcpCBReadPayload

librtp.so

RTP_RtpCreate

RTP_OpenSession

RTP_RtcpCreate

rtcp_sock_notify

RTP_SetPacketBuf

RTP_ParseRtcpPacket

DMC_RTP_Sys_Parse_Rtcp_Packet

DMC_RTP_Sys_Parse_Rtcp_APP_Packet

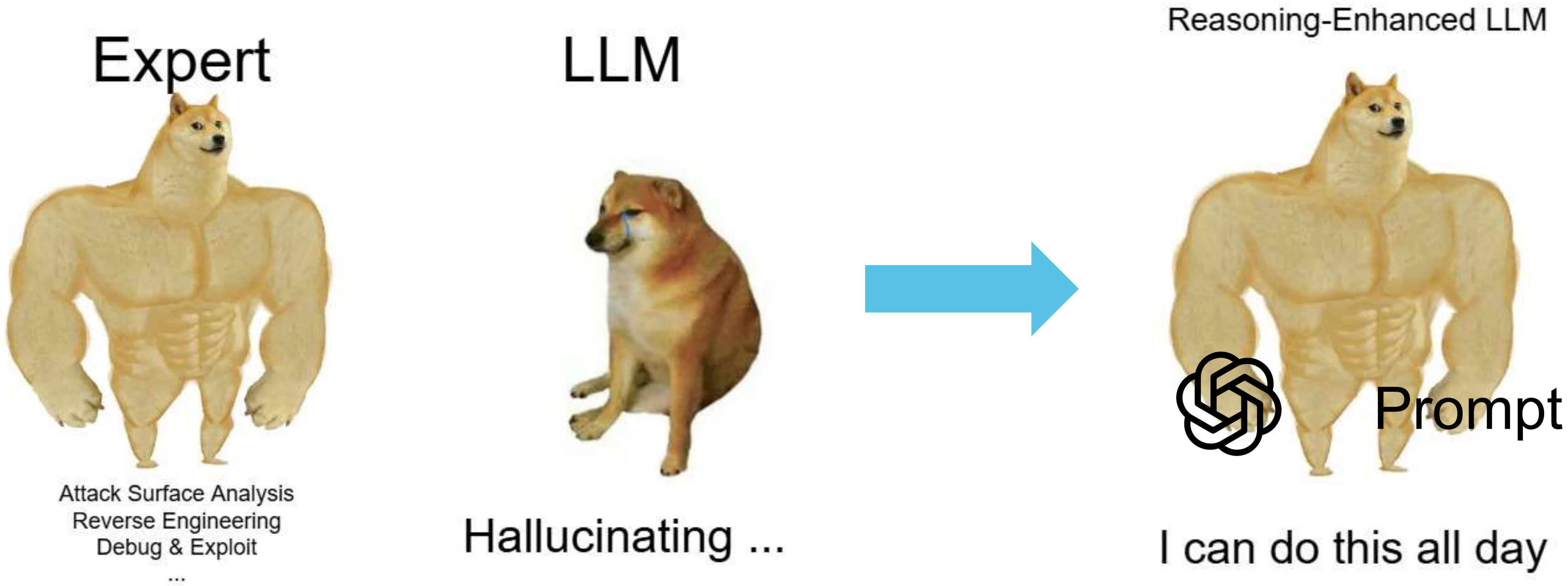
```
  libsamsung_videoengine_9_0.so!CTransportManager::StartReceive(CTransportManager)

  librtp.so!RTP_RtpCreate
            rtp_content = PSIMallocEx(216LL)
            librtp.so!RTP_OpenSession
                pRtpSession = PSIMallocEx(38384LL)
            *(rtp_content + 176) = pRtpSession
  *(CTransportManager + 1704) = rtp_content
  librtp.so!RTP_RtcpCreate
    rtcp_content = PSIMallocEx(1992LL)
          *(rtcp_content + 264) = *(rtp_content + 176) = pRtpSession
          PSIRegisterAsyncSelect(v27, rtcp_sock_notify, v6, 0LL, 1LL) )
            librtp.so!rtcp_sock_notify(__int64 a1, __int64 a2, int a3, _QWORD *a4, unsigned int a5)

                  libsamsung_videoengine_9_0.so!rtcpCB(*a4, 1LL, a4[2], 0LL, &v16)
                    *a5 = (char *)CTransportManager + 6808
                  *((_QWORD *)CTransportManager_0ff6808 + 4) = CTransportManager + 6808 + 40
                  PSISocketRecvFrom(a4[3], CTransportManager + 6808 + 40, 1600LL, 0LL, v51)                         // rtcp_recv_buffer:6848
                  libsamsung_videoengine_9_0.so!rtcpCB(*a4, 0LL, a4[2], v7, 0LL);
                    libsamsung_videoengine_9_0.so!CTransportManager::RtcpCBReadPayload(CTransportManager,CTransportManager_0ff6808)
                            librtp.so!RTP_SetPacketBuf(*(CTransportManager + 1704),3,*((_QWORD *)CTransportManager_0ff6808 + 4)) // rtcp_recv_buffer
                                      pRtpSession = *(_DWORD **)(rtp_content + 176)
                                      *(pRtpSession+19776) = *((_QWORD *)CTransportManager_0ff6808 + 4)
                            librtp.so!RTP_SetPacketBuf(*(CTransportManager + 1704), 5, (__int64)CTransportManager + 2440)     // app rtcp buffer
                                      pRtpSession = *(_DWORD **)(rtp_content + 176)
                                      *(pRtpSession+20936) = (__int64)CTransportManager + 2440
                            librtp.so!RTP_ParseRtcpPacket(rtcp_content, rtcp_recv_buffer, rtcp_recv_buffer_length, &output)
                                      librtp.so!DMC_RTP_Sys_Parse_Rtcp_Packet(pRtpSession_off19752, rtcp_recv_buffer, v12, rtcp_recv_buffer_length, v13)
                                                librtp.so!DMC_RTP_Sys_Parse_Rtcp_APP_Packet(pRtpSession_off19752, rtcp_recv_buffer, a3, poffset)
                                                          PSIMemcpy(*(_QWORD *)(pRtpSession_off19752 + 1184), rtcp_recv_buffer + *poffset, v8)
*(CTransportManager + 1712) = rtcp_content
```

Expert

LLM

Attack Surface Analysis
Reverse Engineering
Debug & Exploit
...

Hallucinating ...



Expert

Attack Surface Analysis
Reverse Engineering
Debug & Exploit
...

LLM

Hallucinating ...

Reasoning-Enhanced LLM

Prompt

I can do this all day

Expert

Attack Surface Analysis
Reverse Engineering
Debug & Exploit
...

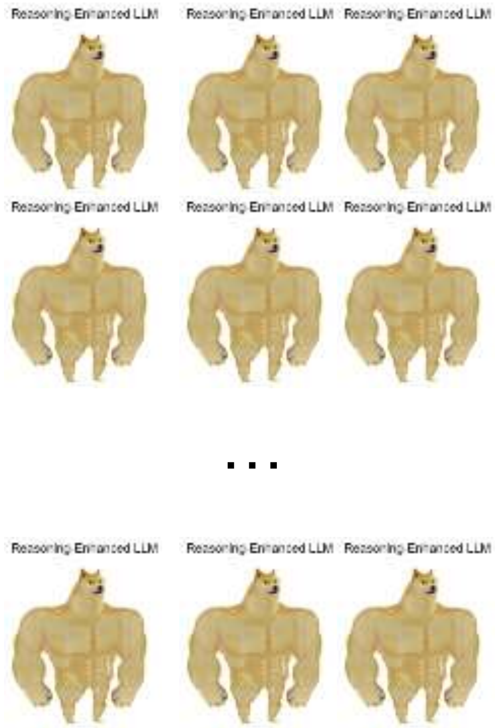
LLM

Hallucinating ...

Reasoning-Enhanced LLM

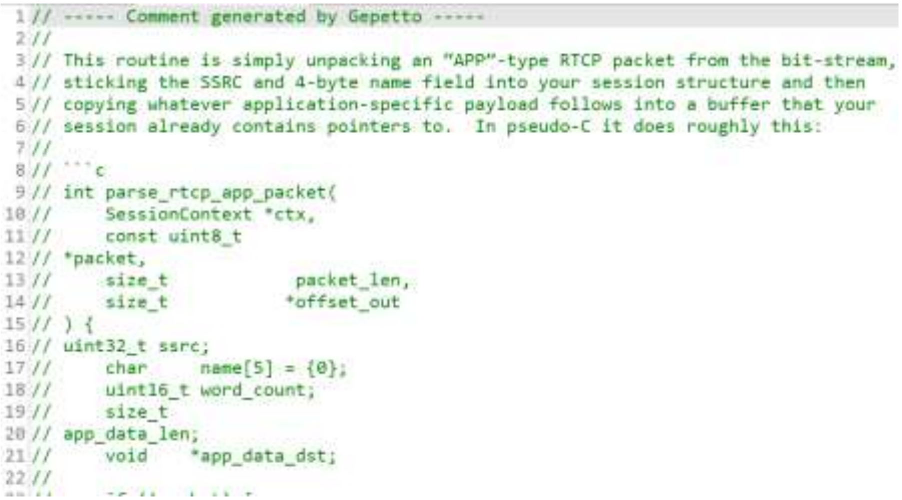
Prompt

I can do this all day

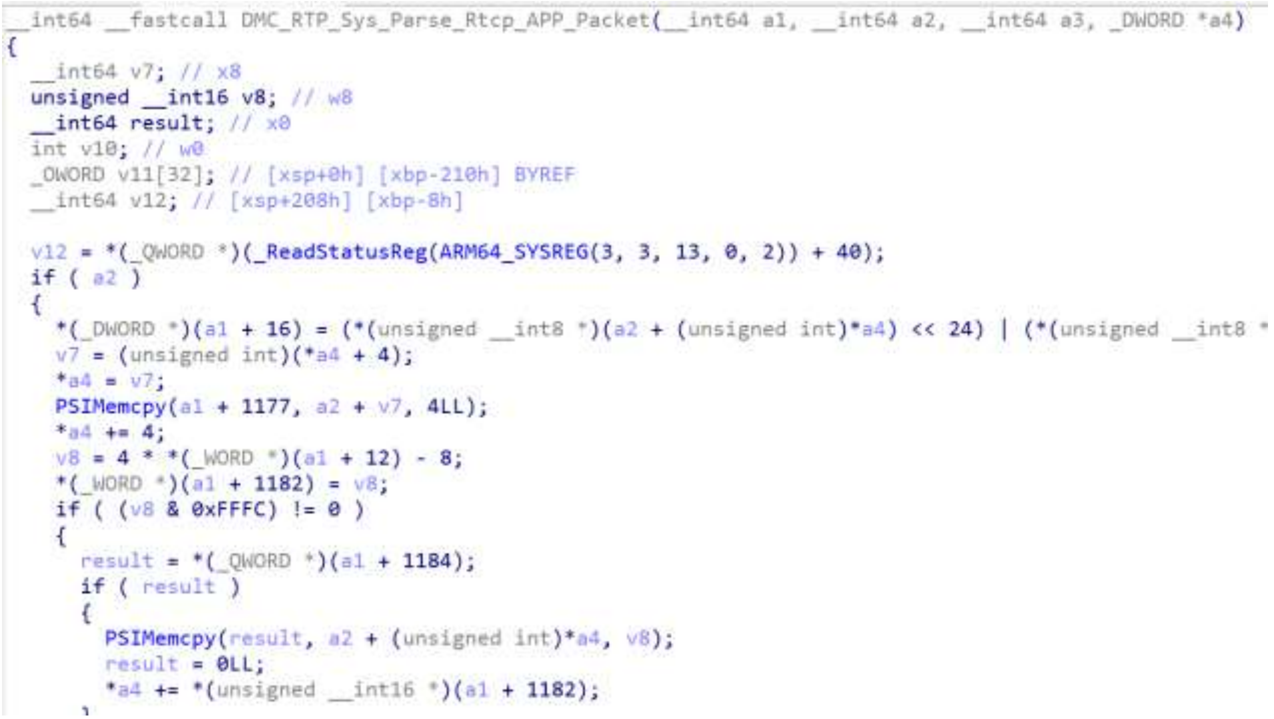


Exploring LLMs’ Strengths in Code Reasoning

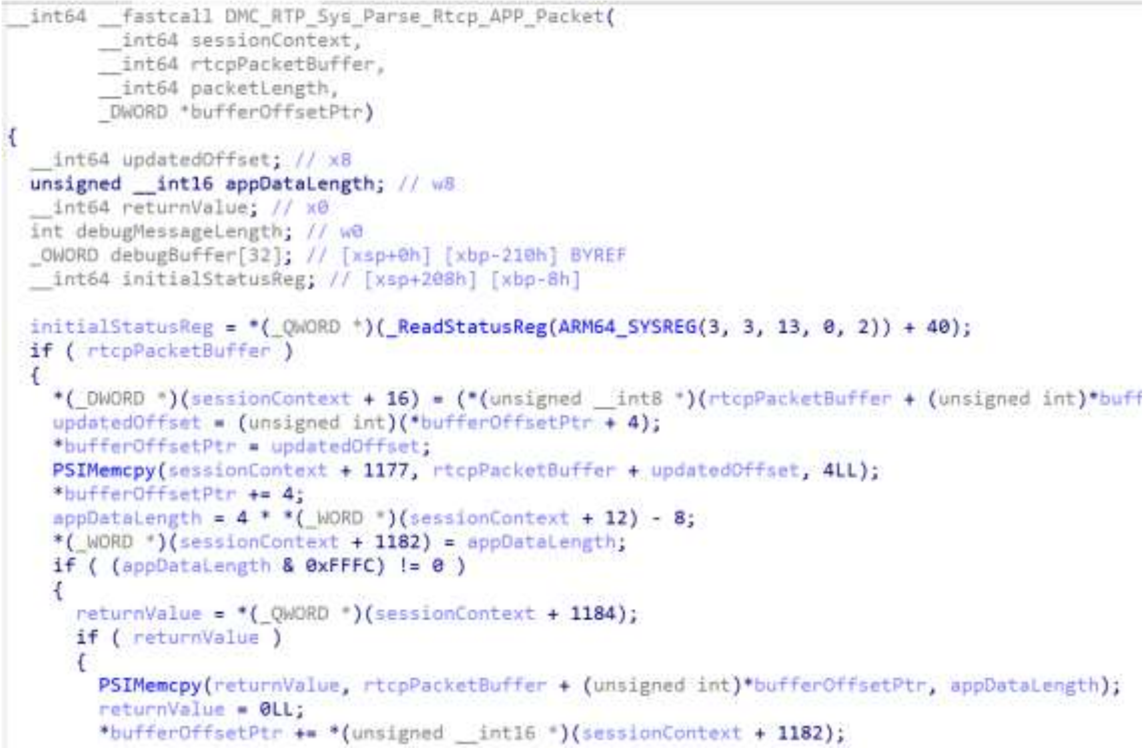
- An example to show how AI know a function with simple prompts
- IDA plugin Gepetto: rename var & explain the function



```
// ----- Comment generated by Gepetto -----
//
// This routine is simply unpacking an "APP"-type RTCP packet from the bit-stream,
// sticking the SSRC and 4-byte name field into your session structure and then
// copying whatever application-specific payload follows into a buffer that your
// session already contains pointers to.  In pseudo-C it does roughly this:
//
// ```c
// int parse_rtcp_app_packet(
//     SessionContext *ctx,
//     const uint8_t
// *packet,
//     size_t             packet_len,
//     size_t            *offset_out
// ) {
// uint32_t ssrc;
//     char     name[5] = {0};
//     uint16_t word_count;
//     size_t
// app_data_len;
//     void    *app_data_dst;
//
```

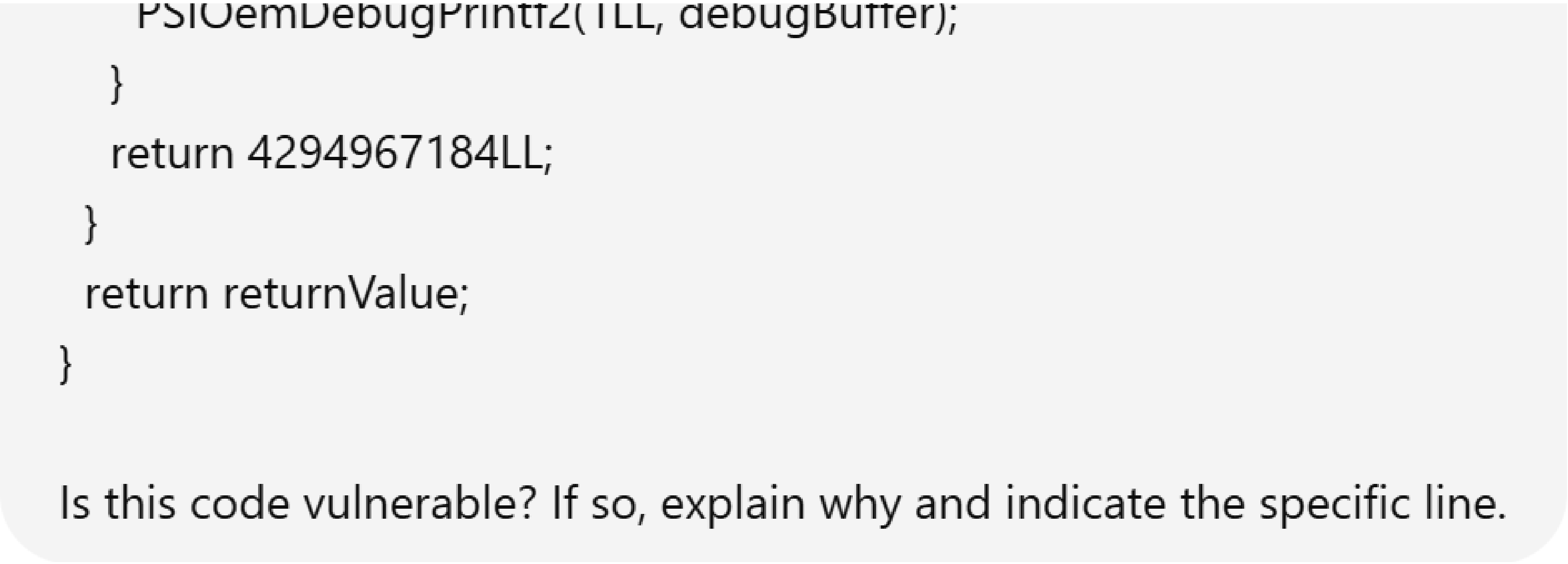


```
__int64 __fastcall DMC_RTP_Sys_Parse_Rtcp_APP_Packet(__int64 a1, __int64 a2, __int64 a3, _DWORD *a4)
{
  __int64 v7; // x8
  unsigned __int16 v8; // w8
  __int64 result; // x0
  int v10; // w0
  _OWORD v11[32]; // [xsp+0h] [xbp-210h] BYREF
  __int64 v12; // [xsp+208h] [xbp-8h]

  v12 = *(_QWORD *)(_ReadStatusReg(ARM64_SYSREG(3, 3, 13, 0, 2)) + 40);
  if ( a2 )
  {
    *(_DWORD *)(a1 + 16) = (*(unsigned __int8 *)(a2 + (unsigned int)*a4) << 24) | (*(unsigned __int8 *
    v7 = (unsigned int)(*a4 + 4);
    *a4 = v7;
    PSIMemcpy(a1 + 1177, a2 + v7, 4LL);
    *a4 += 4;
    v8 = 4 * *(_WORD *)(a1 + 12) - 8;
    *(_WORD *)(a1 + 1182) = v8;
    if ( (v8 & 0xFFFC) != 0 )
    {
      result = *(_QWORD *)(a1 + 1184);
      if ( result )
      {
        PSIMemcpy(result, a2 + (unsigned int)*a4, v8);
        result = 0LL;
        *a4 += *(unsigned __int16 *)(a1 + 1182);
```



```
__int64 __fastcall DMC_RTP_Sys_Parse_Rtcp_APP_Packet(
        __int64 sessionContext,
        __int64 rtcpPacketBuffer,
        __int64 packetLength,
        _DWORD *bufferOffsetPtr)
{
  __int64 updatedOffset; // x8
  unsigned __int16 appDataLength; // w8
  __int64 returnValue; // x0
  int debugMessageLength; // w0
  _OWORD debugBuffer[32]; // [xsp+0h] [xbp-210h] BYREF
  __int64 initialStatusReg; // [xsp+208h] [xbp-8h]

  initialStatusReg = *(_QWORD *)(_ReadStatusReg(ARM64_SYSREG(3, 3, 13, 0, 2)) + 40);
  if ( rtcpPacketBuffer )
  {
    *(_DWORD *)(sessionContext + 16) = (*(unsigned __int8 *)(rtcpPacketBuffer + (unsigned int)*buff
    updatedOffset = (unsigned int)(*bufferOffsetPtr + 4);
    *bufferOffsetPtr = updatedOffset;
    PSIMemcpy(sessionContext + 1177, rtcpPacketBuffer + updatedOffset, 4LL);
    *bufferOffsetPtr += 4;
    appDataLength = 4 * *(_WORD *)(sessionContext + 12) - 8;
    *(_WORD *)(sessionContext + 1182) = appDataLength;
    if ( (appDataLength & 0xFFFC) != 0 )
    {
      returnValue = *(_QWORD *)(sessionContext + 1184);
      if ( returnValue )
      {
        PSIMemcpy(returnValue, rtcpPacketBuffer + (unsigned int)*bufferOffsetPtr, appDataLength);
        returnValue = 0LL;
        *bufferOffsetPtr += *(unsigned __int16 *)(sessionContext + 1182);
```

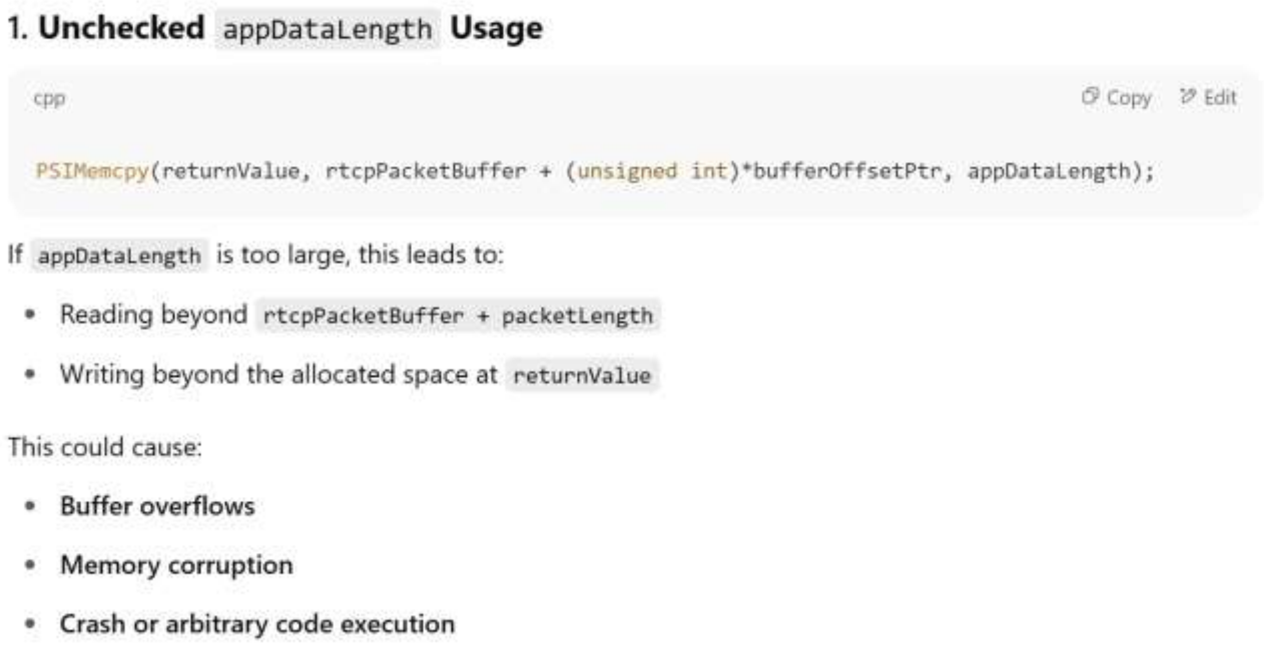
- Trivial try: ask LLM with a function, to see if it is vulnerable, AI’s answer needs many assumptions



```
        PSIOemDebugPrintf2(1LL, debugBuffer);
      }
      return 4294967184LL;
    }
  return returnValue;
}
```

Is this code vulnerable? If so, explain why and indicate the specific line.

- Trivial try: ask LLM with a function, to see if it is vulnerable, AI’s answer needs many assumptions



1. **Unchecked `appDataLength` Usage**

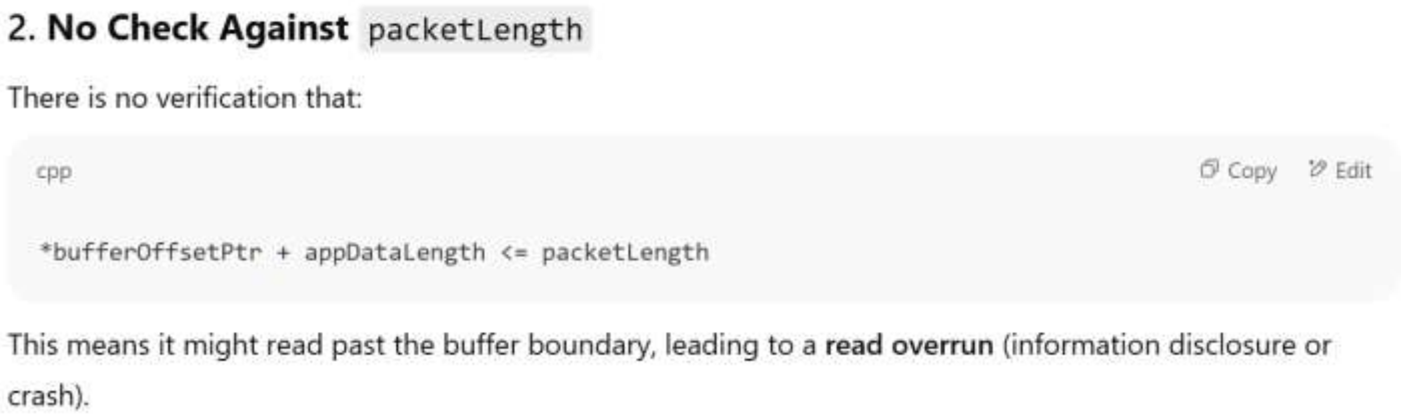
```cpp
PSIMemcpy(returnValue, rtcpPacketBuffer + (unsigned int)*bufferOffsetPtr, appDataLength);
```

If `appDataLength` is too large, this leads to:

- Reading beyond `rtcpPacketBuffer + packetLength`
- Writing beyond the allocated space at `returnValue`

This could cause:

- **Buffer overflows**
- **Memory corruption**
- **Crash or arbitrary code execution**

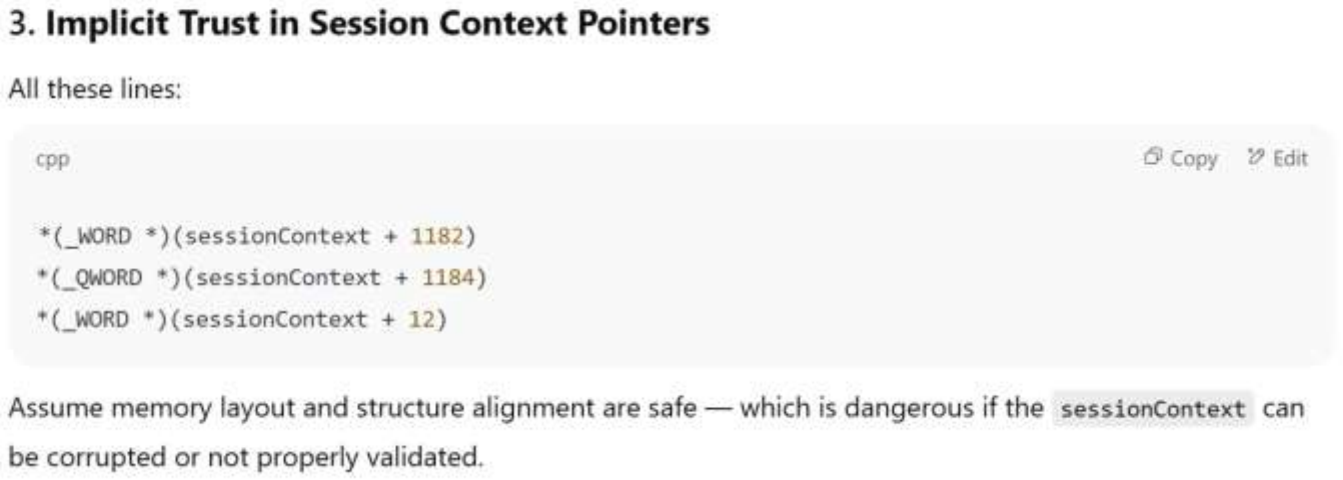


2. **No Check Against `packetLength`**

There is no verification that:

```cpp
*bufferOffsetPtr + appDataLength <= packetLength
```

This means it might read past the buffer boundary, leading to a **read overrun** (information disclosure or crash).



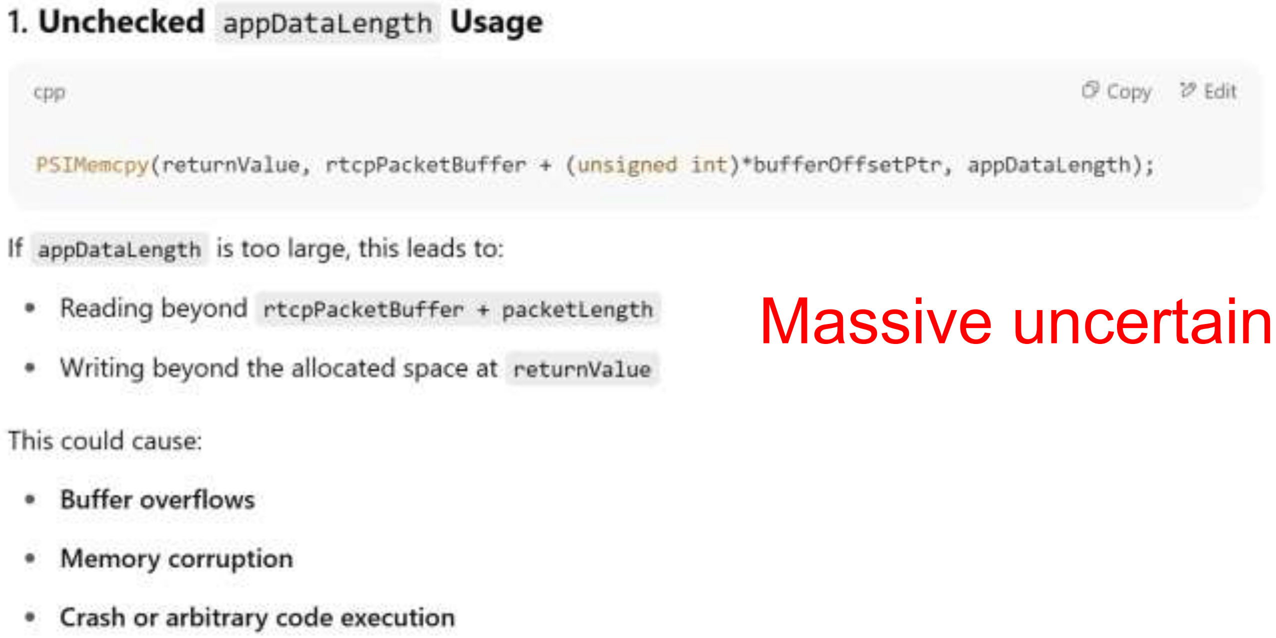
3. **Implicit Trust in Session Context Pointers**

All these lines:

```cpp
*(_WORD *)(sessionContext + 1182)
*(_QWORD *)(sessionContext + 1184)
*(_WORD *)(sessionContext + 12)
```

Assume memory layout and structure alignment are safe — which is dangerous if the `sessionContext` can be corrupted or not properly validated.

- Trivial try: ask LLM with a function, to see if it is vulnerable, AI’s answer needs many assumptions



1. **Unchecked** `appDataLength` **Usage**

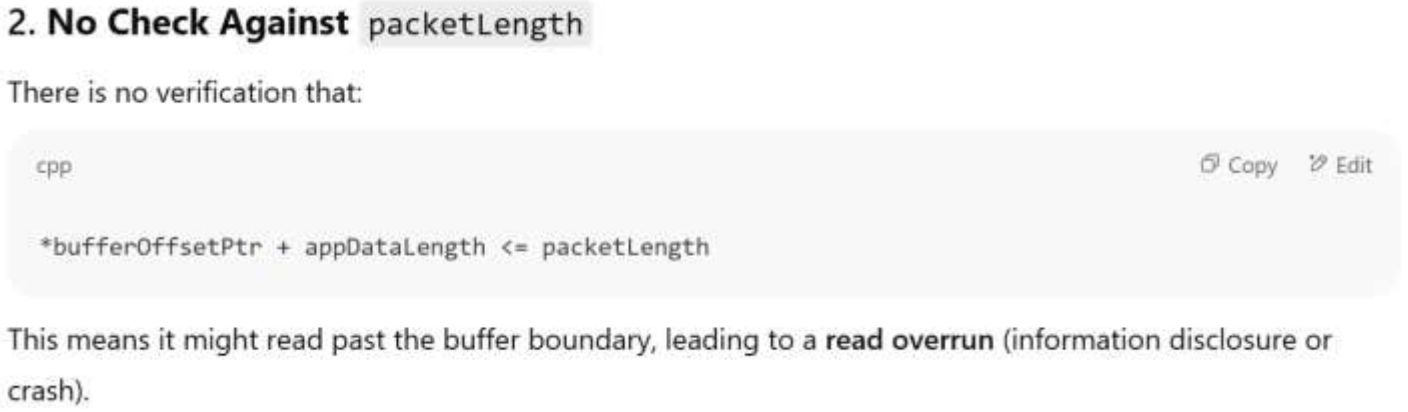
```cpp
PSIMemcpy(returnValue, rtcpPacketBuffer + (unsigned int)*bufferOffsetPtr, appDataLength);
```

If `appDataLength` is too large, this leads to:

- Reading beyond `rtcpPacketBuffer + packetLength`
- Writing beyond the allocated space at `returnValue`

This could cause:

- **Buffer overflows**
- **Memory corruption**
- **Crash or arbitrary code execution**



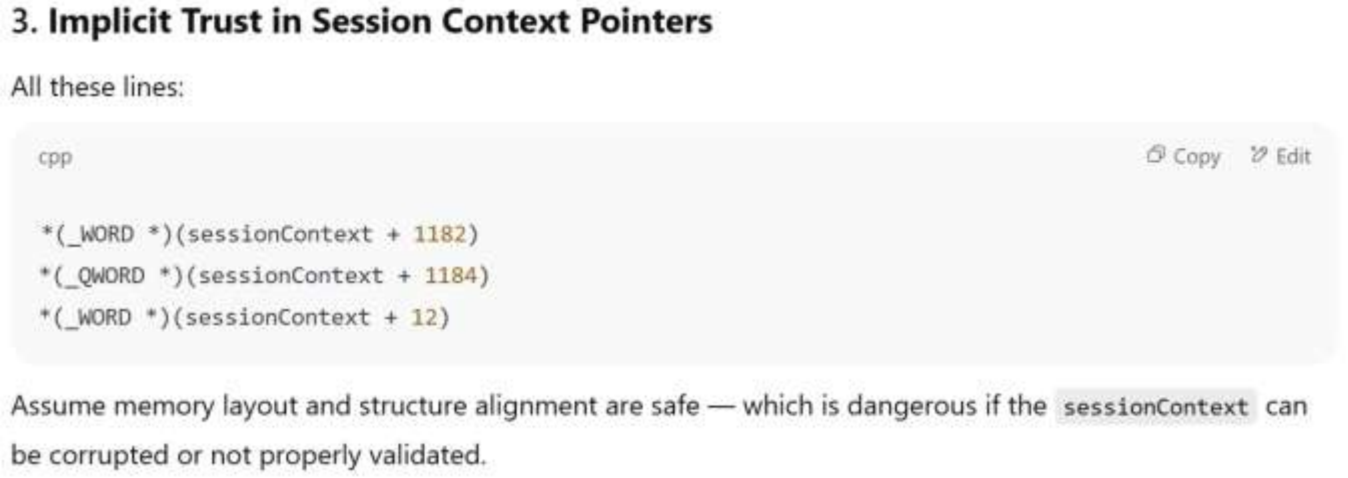
2. **No Check Against** `packetLength`

There is no verification that:

```cpp
*bufferOffsetPtr + appDataLength <= packetLength
```

This means it might read past the buffer boundary, leading to a **read overrun** (information disclosure or crash).

Massive uncertain reports == Useless



3. **Implicit Trust in Session Context Pointers**

All these lines:

```cpp
*(_WORD *)(sessionContext + 1182)
*(_QWORD *)(sessionContext + 1184)
*(_WORD *)(sessionContext + 12)
```

Assume memory layout and structure alignment are safe — which is dangerous if the `sessionContext` can be corrupted or not properly validated.

- If we fill enough pre-conditions (constraints), LLM can give Reasoning analysis result with certainty and accuracy.

```
PSIMemcpy(returnValue, rtcpPacketBuffer + (unsigned int)*bufferOffsetPtr, appDataLength);
```

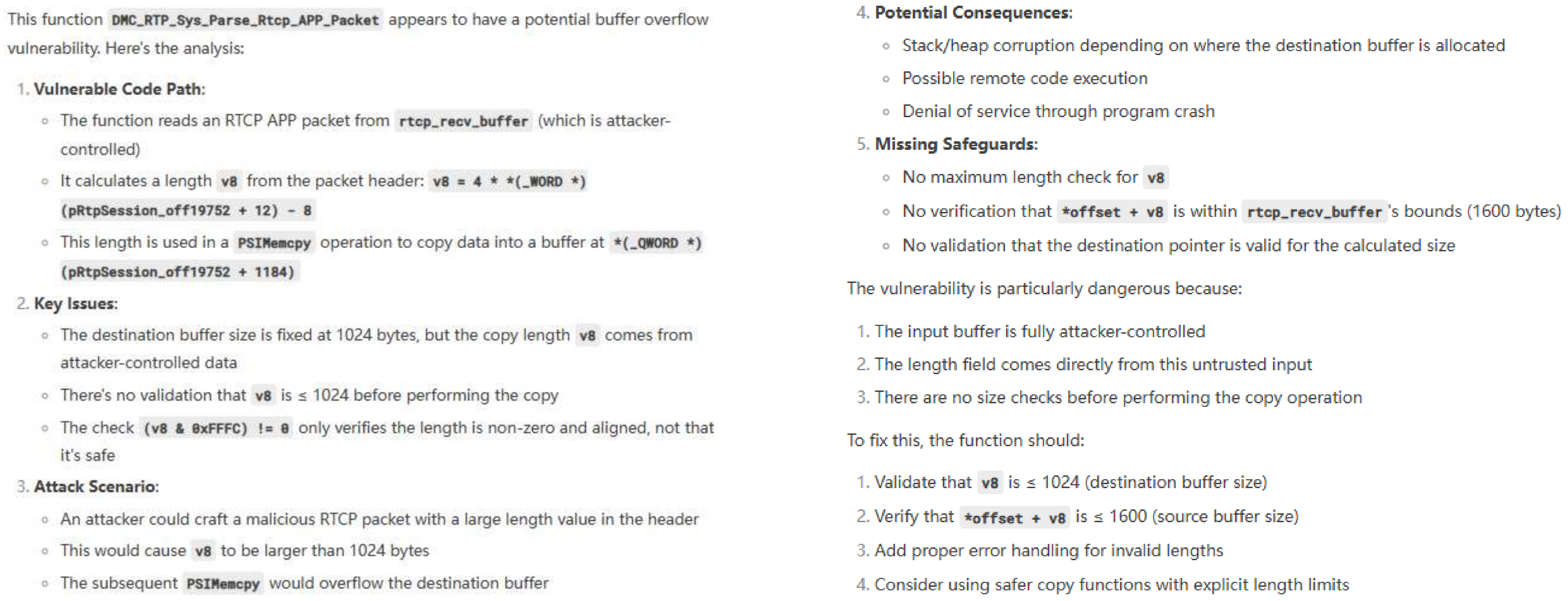
- If we fill enough pre-conditions (constraints), LLM can give Reasoning analysis result with certainty and accuracy.

```
PSIMemcpy(returnValue, rtcpPacketBuffer + (unsigned int)*bufferOffsetPtr, appDataLength);
```

Size: 1024

Size: 1600

Type: WORD 0-0xffff
Controlled by attacker



This function `DMC_RTP_Sys_Parse_Rtcp_APP_Packet` appears to have a potential buffer overflow vulnerability. Here's the analysis:

1. **Vulnerable Code Path**:
    - The function reads an RTCP APP packet from `rtcp_recv_buffer` (which is attacker-controlled)
    - It calculates a length `v8` from the packet header: `v8 = 4 * *(_WORD *)(pRtpSession_off19752 + 12) - 8`
    - This length is used in a `PSIMemcpy` operation to copy data into a buffer at `*(_QWORD *)(pRtpSession_off19752 + 1184)`
2. **Key Issues**:
    - The destination buffer size is fixed at 1024 bytes, but the copy length `v8` comes from attacker-controlled data
    - There's no validation that `v8` is ≤ 1024 before performing the copy
    - The check `(v8 & 0xFFFC) != 0` only verifies the length is non-zero and aligned, not that it's safe
3. **Attack Scenario**:
    - An attacker could craft a malicious RTCP packet with a large length value in the header
    - This would cause `v8` to be larger than 1024 bytes
    - The subsequent `PSIMemcpy` would overflow the destination buffer
4. **Potential Consequences**:
    - Stack/heap corruption depending on where the destination buffer is allocated
    - Possible remote code execution
    - Denial of service through program crash
5. **Missing Safeguards**:
    - No maximum length check for `v8`
    - No verification that `*offset + v8` is within `rtcp_recv_buffer`'s bounds (1600 bytes)
    - No validation that the destination pointer is valid for the calculated size

The vulnerability is particularly dangerous because:

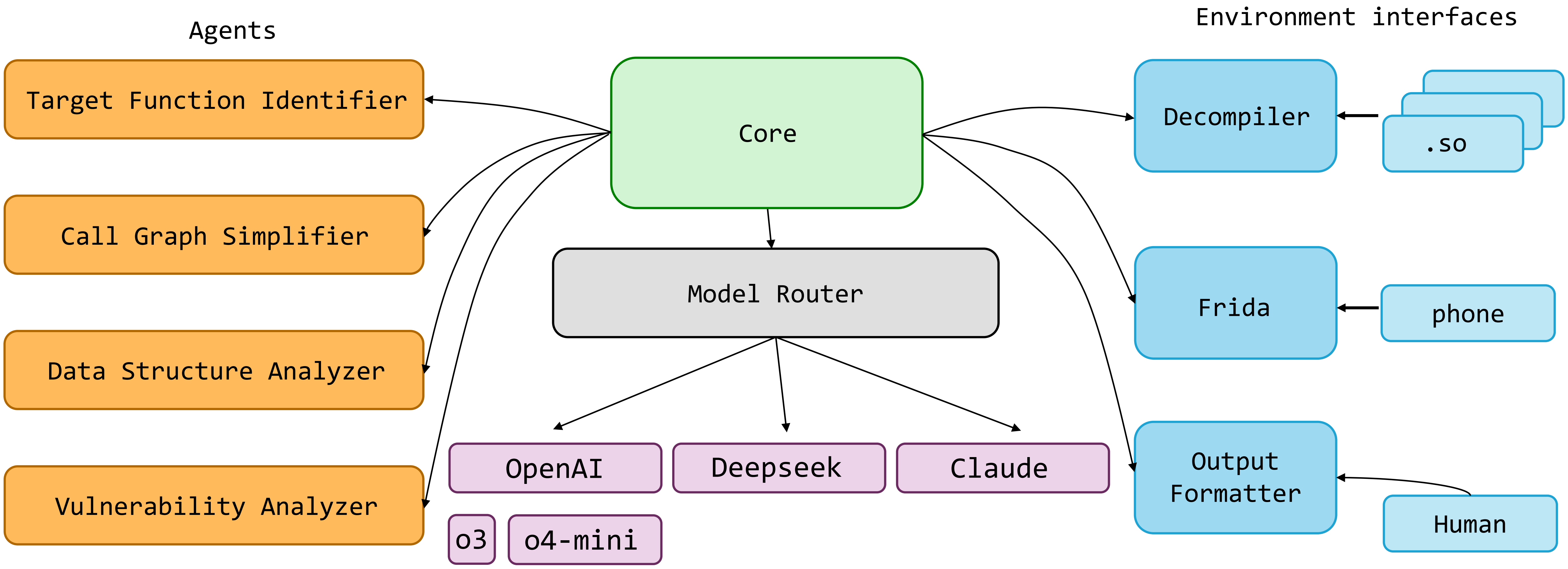
1. The input buffer is fully attacker-controlled
2. The length field comes directly from this untrusted input
3. There are no size checks before performing the copy operation

To fix this, the function should:

1. Validate that `v8` is ≤ 1024 (destination buffer size)
2. Verify that `*offset + v8` is ≤ 1600 (source buffer size)
3. Add proper error handling for invalid lengths
4. Consider using safer copy functions with explicit length limits

Improving Code Analysis with LLMs via CVE-2024-34587

Architecture



Agents

Environment interfaces

Target Function Identifier

Call Graph Simplifier

Data Structure Analyzer

Vulnerability Analyzer

Core

Model Router

OpenAI

Deepseek

Claude

o3

o4-mini

Decompiler

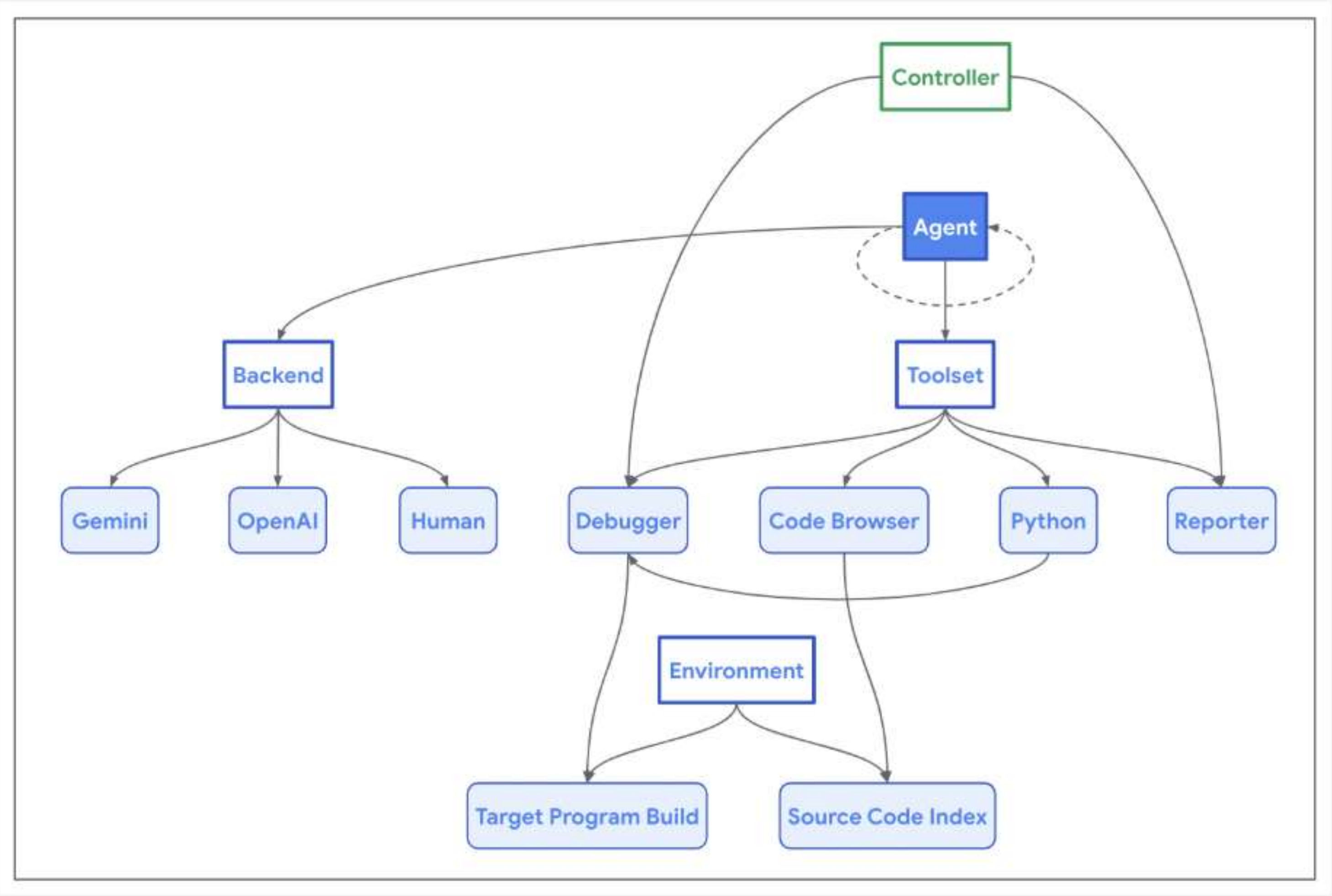
.so

Frida

phone

Output Formatter

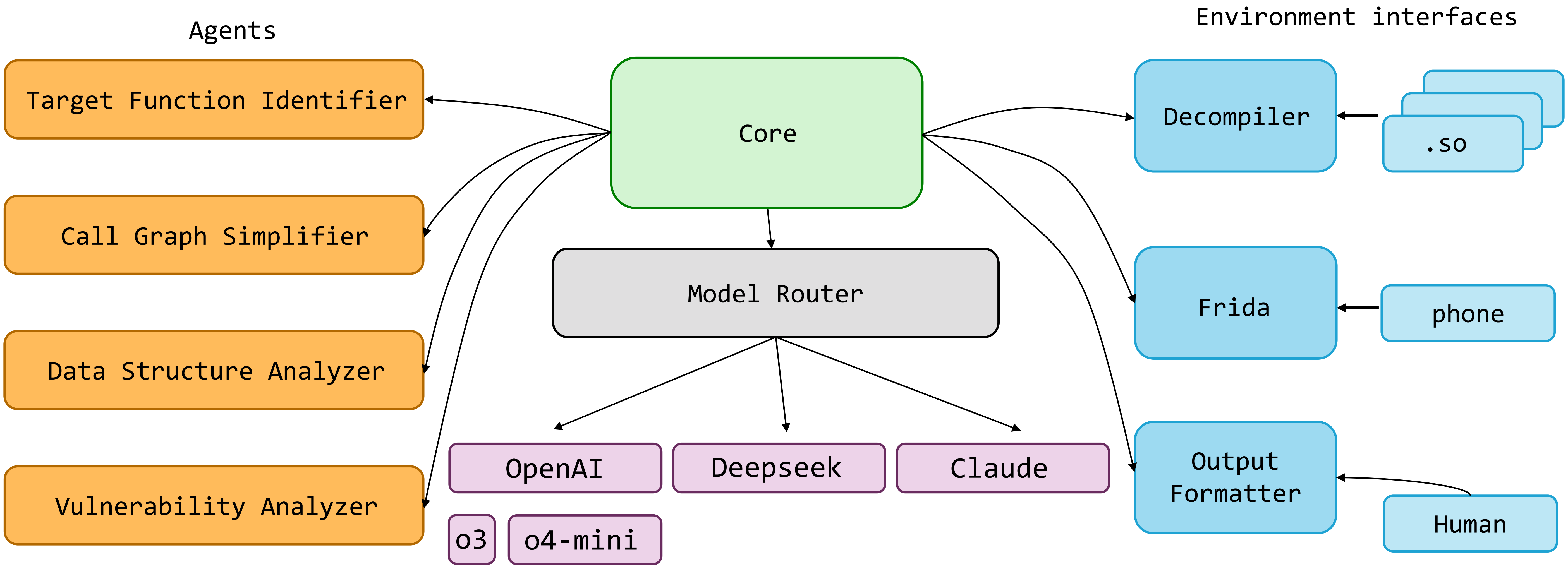
Human



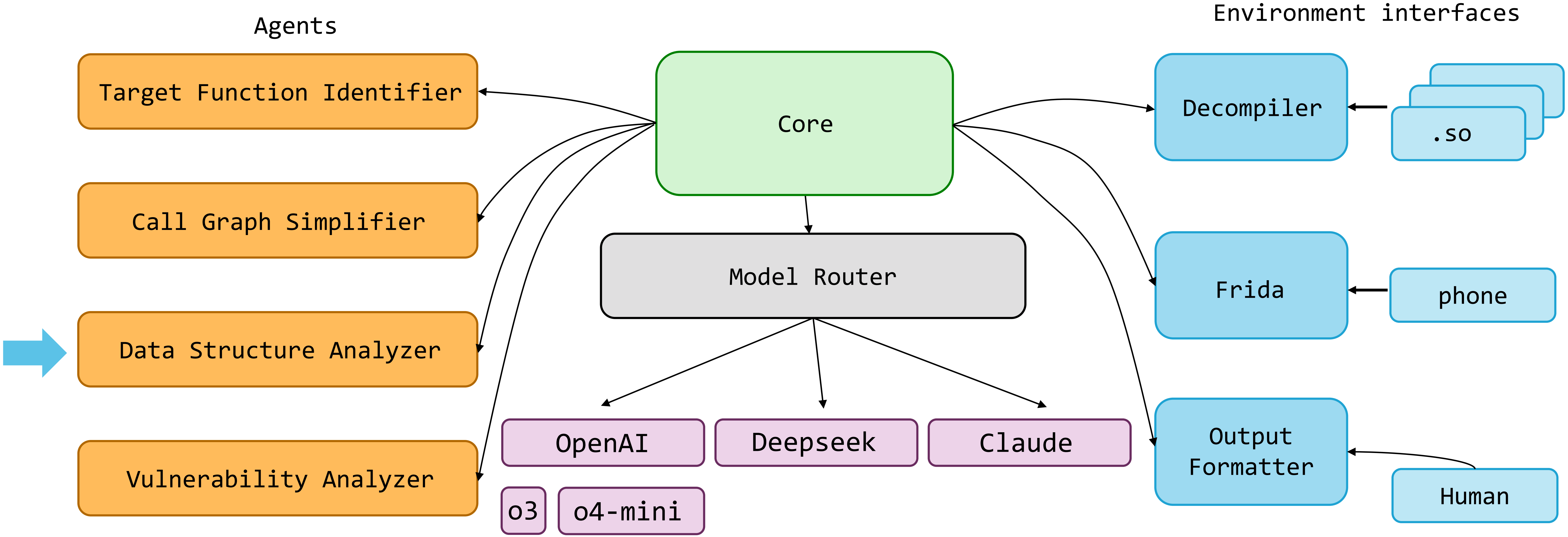
Naptime architecture.

The Naptime architecture developed by Google Project Zero

Architecture



Architecture



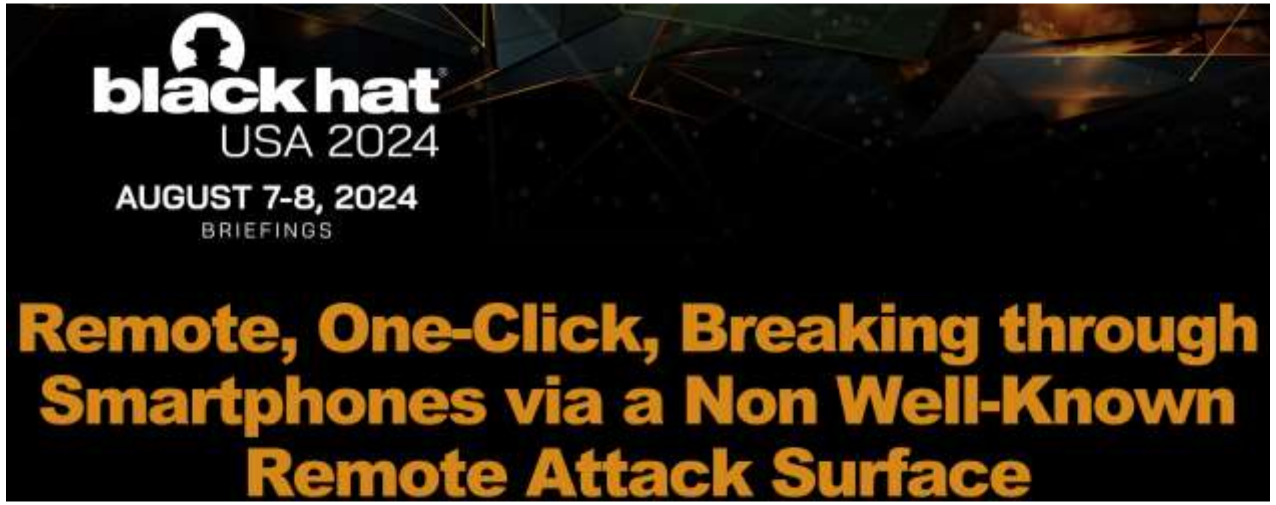
Roadmap

- Identify the target process. [Human]
- Build global call graph. [Static Analysis]
- Locate packet receiving function & parsing function. [AI]
- Re construct data structures. [AI]
- Perform vulnerability analysis based on the data structures. [AI]
- Generate the report. [AI]
- Verify the results. [Human]

Pre work: Identify the target process [human]

SecVideoEngineService

- Run on high privilege
- Accessed from remote
- Install and run by default on mobile phones
- Simple attack conditions



black hat
USA 2024
AUGUST 7-8, 2024
BRIEFINGS

Remote, One-Click, Breaking through Smartphones via a Non Well-Known Remote Attack Surface

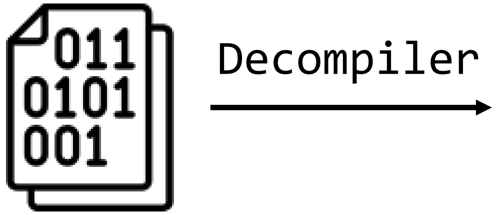
Run on high privilege
Accessed from remote
Install and run by default on mobile phones
Simple attack conditions

Build global call graph [static analysis]

- Decompiler -> pseudo code for LLM
- Frida -> gain indirect call info
- Static Analysis: Build global call graph

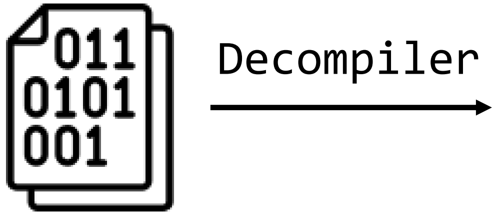
Build global call graph [static analysis]

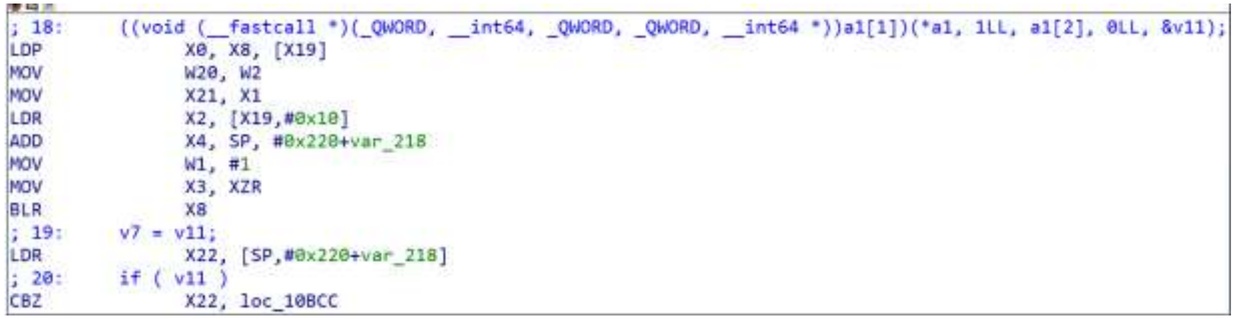
- Decompiler -> pseudo code for LLM
- Frida -> gain indirect call info
- Static Analysis: Build global call graph



Build global call graph [static analysis]

- Decompiler -> pseudo code for LLM
- Frida -> gain indirect call info
- Static Analysis: Build global call graph

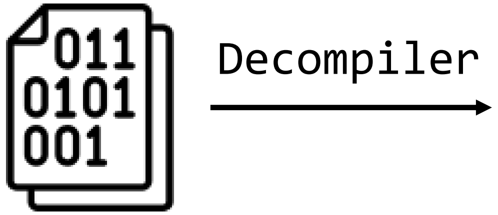


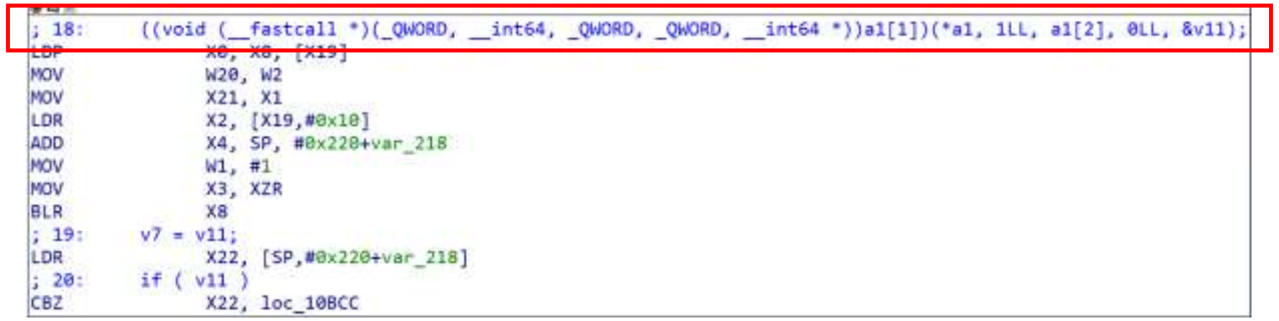


```
; 18:     ((void (__fastcall *)(_QWORD, __int64, _QWORD, _QWORD, __int64 *))a1[1])(*a1, 1LL, a1[2], 0LL, &v11);
LDP             X0, X8, [X19]
MOV             W20, W2
MOV             X21, X1
LDR             X2, [X19,#0x10]
ADD             X4, SP, #0x220+var_218
MOV             W1, #1
MOV             X3, XZR
BLR             X8
; 19:     v7 = v11;
LDR             X22, [SP,#0x220+var_218]
; 20:     if ( v11 )
CBZ             X22, loc_10BCC
```

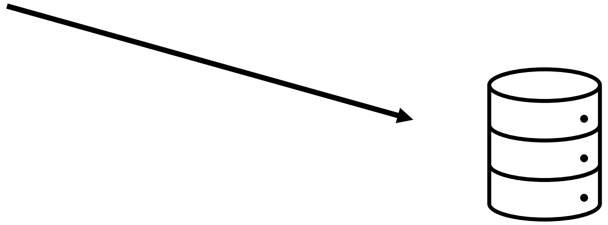
Build global call graph [static analysis]

- Decompiler -> pseudo code for LLM
- Frida -> gain indirect call info
- Static Analysis: Build global call graph



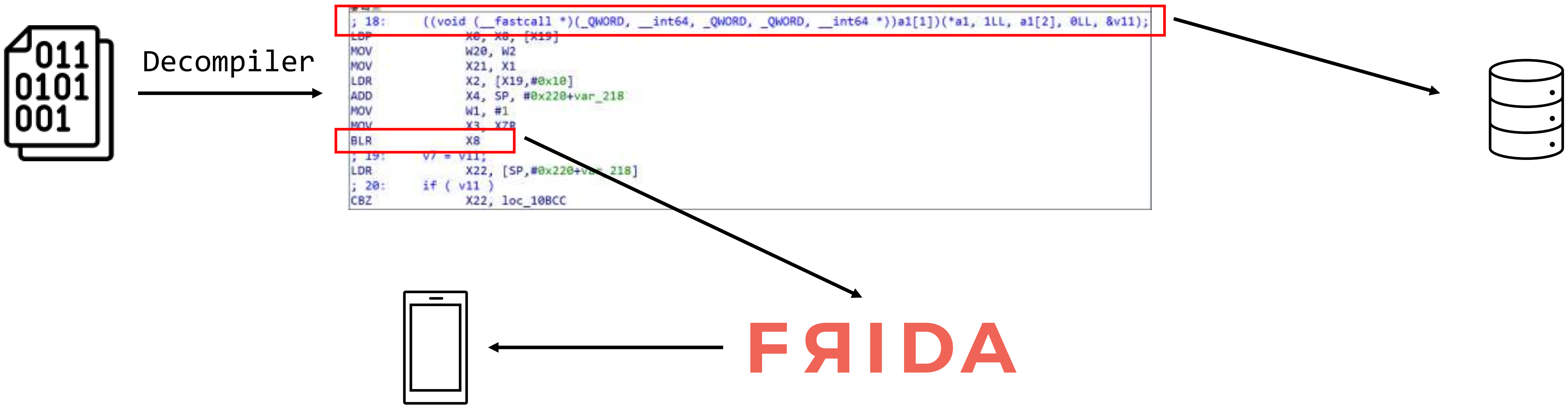


```
; 18:     ((void (__fastcall *)(_QWORD, __int64, _QWORD, _QWORD, __int64 *))a1[1])(*a1, 1LL, a1[2], 0LL, &v11);
LDP             X0, X8, [X19]
MOV             W20, W2
MOV             X21, X1
LDR             X2, [X19,#0x10]
ADD             X4, SP, #0x220+var_218
MOV             W1, #1
MOV             X3, XZR
BLR             X8
; 19:     v7 = v11;
LDR             X22, [SP,#0x220+var_218]
; 20:     if ( v11 )
CBZ             X22, loc_10BCC
```



Build global call graph [static analysis]

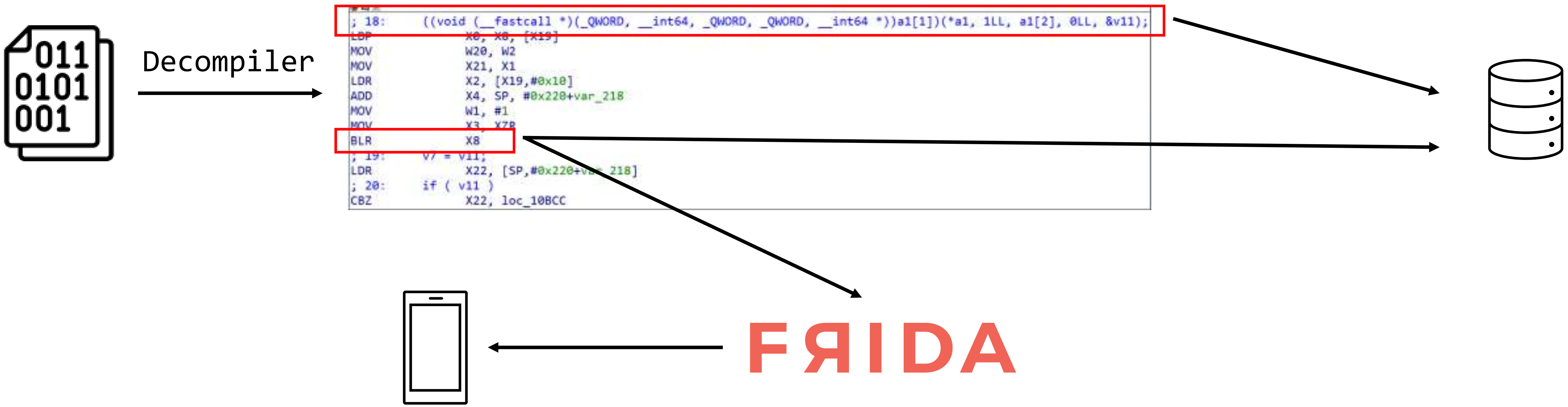
- Decompiler -> pseudo code for LLM
- Frida -> gain indirect call info
- Static Analysis: Build global call graph



```
; 18:     ((void (__fastcall *)(_QWORD, __int64, _QWORD, _QWORD, __int64 *))a1[1])(*a1, 1LL, a1[2], 0LL, &v11);
LDP             X0, X8, [X19]
MOV             W20, W2
MOV             X21, X1
LDR             X2, [X19,#0x10]
ADD             X4, SP, #0x220+var_218
MOV             W1, #1
MOV             X3, XZR
BLR             X8
; 19:     v7 = v11;
LDR             X22, [SP,#0x220+var_218]
; 20:     if ( v11 )
CBZ             X22, loc_10BCC
```

Build global call graph [static analysis]

- Decompiler -> pseudo code for LLM
- Frida -> gain indirect call info
- Static Analysis: Build global call graph



```
; 18:     ((void (__fastcall *)(_QWORD, __int64, _QWORD, _QWORD, __int64 *))a1[1])(*a1, 1LL, a1[2], 0LL, &v11);
LDP             X0, X8, [X19]
MOV             W20, W2
MOV             X21, X1
LDR             X2, [X19,#0x10]
ADD             X4, SP, #0x220+var_218
MOV             W1, #1
MOV             X3, XZR
BLR             X8
; 19:     v7 = v11;
LDR             X22, [SP,#0x220+var_218]
; 20:     if ( v11 )
CBZ             X22, loc_10BCC
```

Locate packet receiving function & parsing function [AI]

- LLM: analyze receive and parse functions based on the global call graph
- Agent workflow: result = Agent(prompt template + input_data)
- Input Data + Background introduction

```
prompt = "<call_graph>, above is the function call stack of an instant messaging program, including the receiving and
parsing of a certain network protocol message." \
```

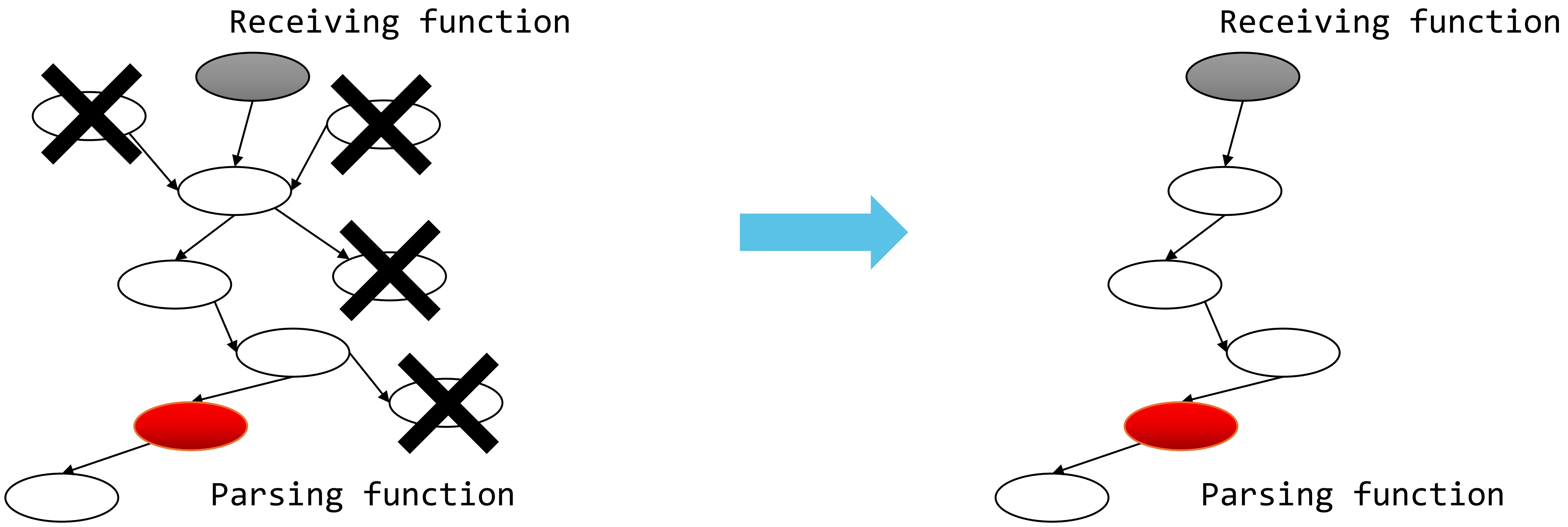
- Task Description

```
"Indicate where network messages from remote endpoints are received and where they are processed (as comprehensively
and completely as possible)." \
"Prune this function call stack and output a single function call stack, ensuring that the call relationships among
the aforementioned functions are preserved and not pruned, without splitting it, and retaining as much relevant
information as possible." \
```

- Format requirements

```
"Please respond in the JSON format {\"Receiving \": ..., \"Parsing\": ..., \"Call Stack\": '...'}, returning \"
Receiving \" and \"Parsing\" as lists of independent functions with no additional information." \
"The function call stack should retain its original format, using spaces to denote hierarchy levels. Ensure the
returned JSON is correctly formatted."
```

Trim call graph based on packet receiving and parsing functions [AI]



Data structure reconstruct [AI]

Goal: Know every buffer’s size before any access

1. Scan & record allocation sites in IDA
2. Decompile function containing allocation site
3. Build initial layout from AST
4. Visit all access sites, do:
    Match access with recorded prototype, policy:
        1) Match field type / name / offset
        2) Match function / class name
        3) Frida hook to verify layout
5. LLM enhancement: infer and complete missing fields

Data structure reconstruct [AI]

Goal: Know every buffer’s size before any access

1. Scan & record allocation sites in IDA
2. Decompile function containing allocation site
3. Build initial layout from AST
4. Visit all access sites, do:

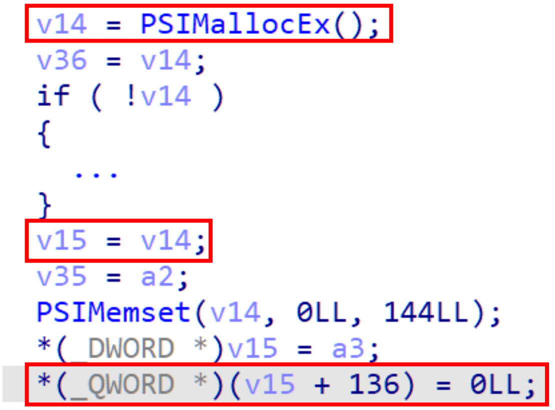
   Match access with recorded prototype, policy:

      1) Match field type / name / offset

      2) Match function / class name

      3) Frida hook to verify layout

5. LLM enhancement: infer and complete missing fields

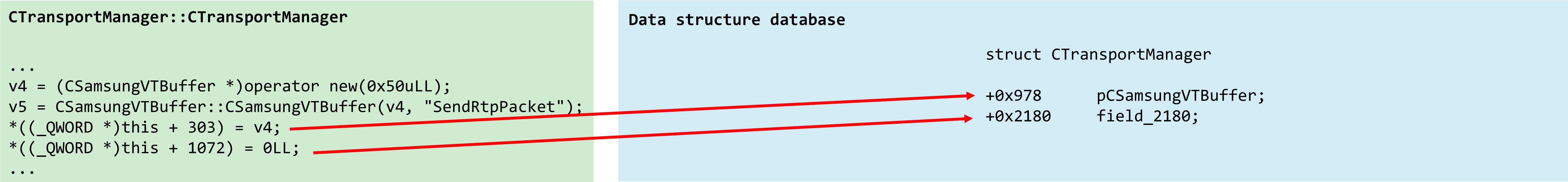


```
v14 = PSIMallocEx();
v36 = v14;
if ( !v14 )
{
  ...
}
v15 = v14;
v35 = a2;
PSIMemset(v14, 0LL, 144LL);
*(_DWORD *)v15 = a3;
*(_QWORD *)(v15 + 136) = 0LL;
```

Code snippet for a
struct’s initialization

Match class name

Match class name



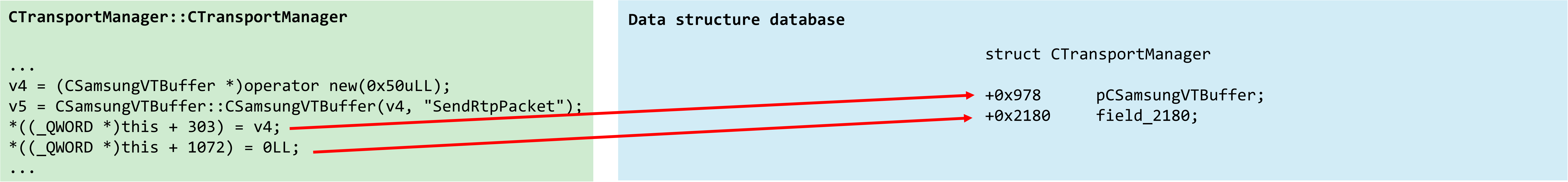
CTransportManager::CTransportManager

```
...
v4 = (CSamsungVTBuffer *)operator new(0x50uLL);
v5 = CSamsungVTBuffer::CSamsungVTBuffer(v4, "SendRtpPacket");
*((_QWORD *)this + 303) = v4;
*((_QWORD *)this + 1072) = 0LL;
...
```

Data structure database

```
struct CTransportManager

+0x978      pCSamsungVTBuffer;
+0x2180     field_2180;
```

Match class name



CTransportManager::CTransportManager

```
...
v4 = (CSamsungVTBuffer *)operator new(0x50uLL);
v5 = CSamsungVTBuffer::CSamsungVTBuffer(v4, "SendRtpPacket");
*((_QWORD *)this + 303) = v4;
*((_QWORD *)this + 1072) = 0LL;
...
```

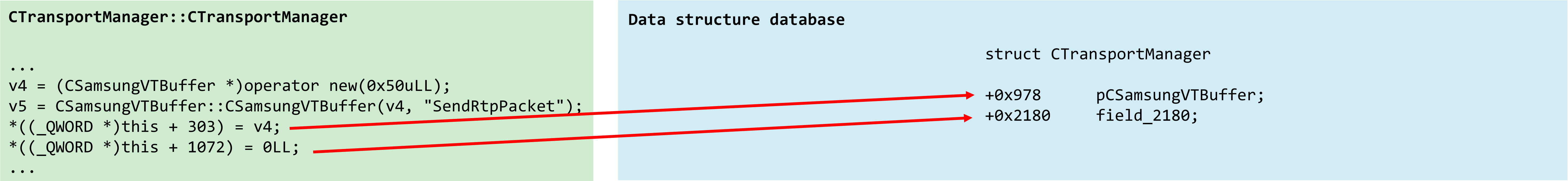
Data structure database

```
struct CTransportManager

+0x978      pCSamsungVTBuffer;
+0x2180     field_2180;
```

CTransportManager::StartReceive →

```
...
PSIMemFreeEx
CSamsungVTBuffer::StartRTCPSend
RTP_RtpCreate
CTransportManager::CreateBuffer
...
```

Match class name

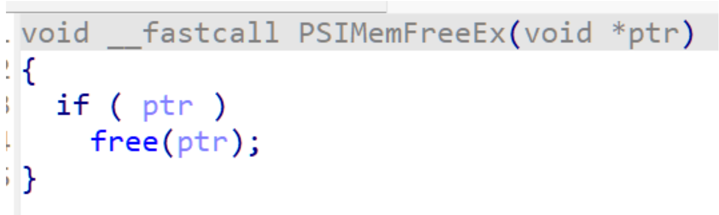


CTransportManager::CTransportManager

```
...
v4 = (CSamsungVTBuffer *)operator new(0x50uLL);
v5 = CSamsungVTBuffer::CSamsungVTBuffer(v4, "SendRtpPacket");
*((_QWORD *)this + 303) = v4;
*((_QWORD *)this + 1072) = 0LL;
...
```

Data structure database

```
struct CTransportManager

+0x978      pCSamsungVTBuffer;
+0x2180     field_2180;
```

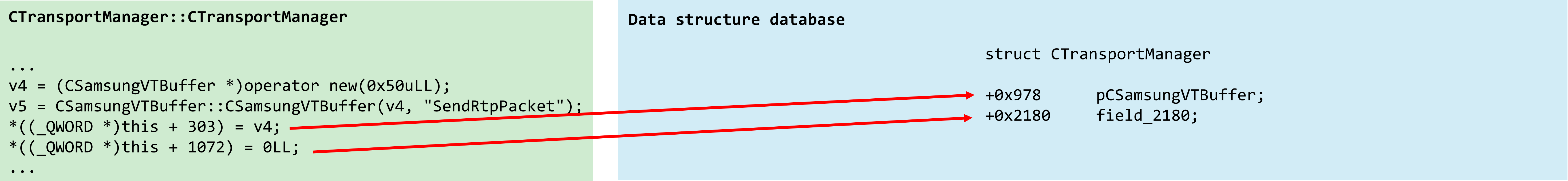


```
void __fastcall PSIMemFreeEx(void *ptr)
{
  if ( ptr )
    free(ptr);
}
```

CTransportManager::StartReceive →

```
...
PSIMemFreeEx
CSamsungVTBuffer::StartRTCPSend
RTP_RtpCreate
CTransportManager::CreateBuffer
...
```

Match class name

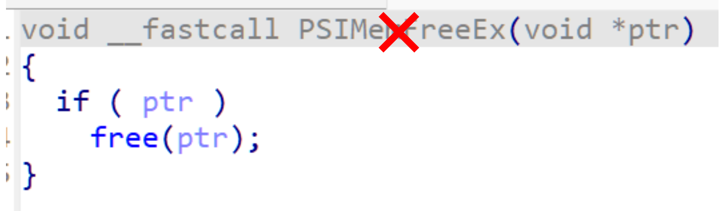


CTransportManager::CTransportManager

```
...
v4 = (CSamsungVTBuffer *)operator new(0x50uLL);
v5 = CSamsungVTBuffer::CSamsungVTBuffer(v4, "SendRtpPacket");
*((_QWORD *)this + 303) = v4;
*((_QWORD *)this + 1072) = 0LL;
...
```

Data structure database

```
struct CTransportManager

+0x978      pCSamsungVTBuffer;
+0x2180     field_2180;
```



```
void __fastcall PSIMemFreeEx(void *ptr)
{
  if ( ptr )
    free(ptr);
}
```

CTransportManager::StartReceive →

```
...
PSIMemFreeEx
CSamsungVTBuffer::StartRTCPSend
RTP_RtpCreate
CTransportManager::CreateBuffer
...
```

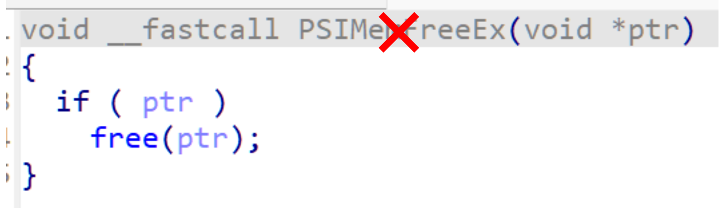
Match class name

CTransportManager::CTransportManager

```
...
v4 = (CSamsungVTBuffer *)operator new(0x50uLL);
v5 = CSamsungVTBuffer::CSamsungVTBuffer(v4, "SendRtpPacket");
*((_QWORD *)this + 303) = v4;
*((_QWORD *)this + 1072) = 0LL;
...
```

Data structure database

```
struct CTransportManager

+0x978      pCSamsungVTBuffer;
+0x2180     field_2180;
```



```
void __fastcall PSIMemFreeEx(void *ptr)
{
  if ( ptr )
    free(ptr);
}
```

CTransportManager::StartReceive →

```
...
PSIMemFreeEx
CSamsungVTBuffer::StartRTCPSend
RTP_RtpCreate
CTransportManager::CreateBuffer
...
```

CTransportManager::StartReceive

```
if ( *((_BYTE *)this + 8448) )
  {
    started = CTransportManager::StartRTCPSend(this);
    if ( (_DWORD)started )
    {
```

CTransportManager::StartRTCPSend

```
...
      *((_BYTE *)this + 2326) = 1;
      v6 =
      PSICreateEventEx("vendor/samsung/external/ims_
video_engine/SVE/TransportMgr/src/TransportManager.
cpp", 1047LL);
      *((_QWORD *)this + 1122) = v6;
...
```

Match class name

CTransportManager::CTransportManager

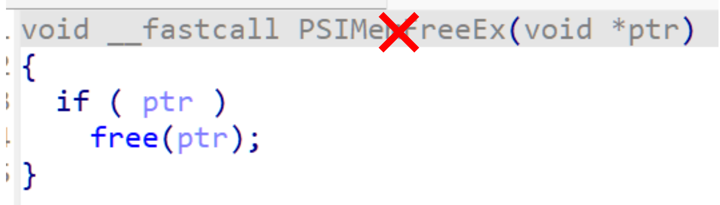
```
...
v4 = (CSamsungVTBuffer *)operator new(0x50uLL);
v5 = CSamsungVTBuffer::CSamsungVTBuffer(v4, "SendRtpPacket");
*((_QWORD *)this + 303) = v4;
*((_QWORD *)this + 1072) = 0LL;
...
```

Data structure database

```
struct CTransportManager

+0x978      pCSamsungVTBuffer;
+0x2180     field_2180;
```

CTransportManager::StartReceive

```
if ( *((_BYTE *)this + 8448) )
  {
    started = CTransportManager::StartRTCPSend(this);
    if ( (_DWORD)started )
    {
```



```
void __fastcall PSIMemFreeEx(void *ptr)
{
  if ( ptr )
    free(ptr);
}
```

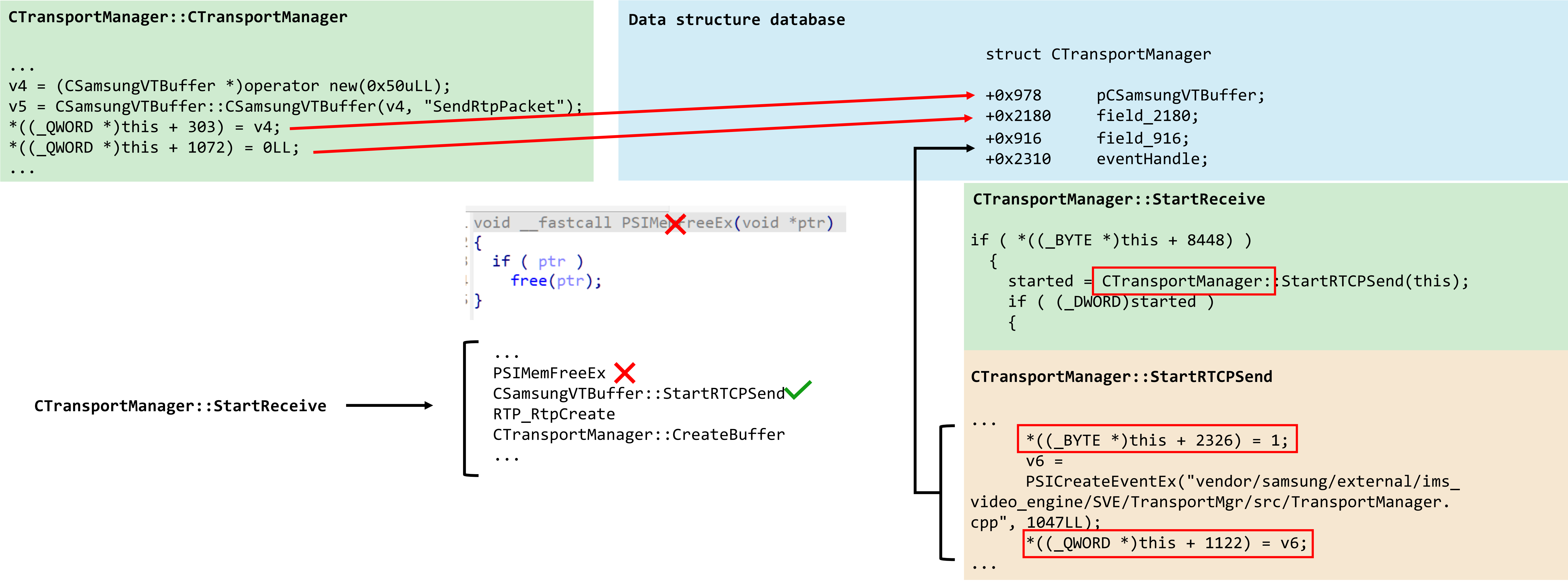
CTransportManager::StartReceive →

```
...
PSIMemFreeEx ✗
CSamsungVTBuffer::StartRTCPSend ✓
RTP_RtpCreate
CTransportManager::CreateBuffer
...
```

CTransportManager::StartRTCPSend

```
...
      *((_BYTE *)this + 2326) = 1;
      v6 =
      PSICreateEventEx("vendor/samsung/external/ims_
video_engine/SVE/TransportMgr/src/TransportManager.
cpp", 1047LL);
      *((_QWORD *)this + 1122) = v6;
...
```

Match class name



CTransportManager::CTransportManager

```
...
v4 = (CSamsungVTBuffer *)operator new(0x50uLL);
v5 = CSamsungVTBuffer::CSamsungVTBuffer(v4, "SendRtpPacket");
*((_QWORD *)this + 303) = v4;
*((_QWORD *)this + 1072) = 0LL;
...
```

Data structure database

```
struct CTransportManager

+0x978      pCSamsungVTBuffer;
+0x2180     field_2180;
+0x916      field_916;
+0x2310     eventHandle;
```

```
void __fastcall PSIMemFreeEx(void *ptr)
{
  if ( ptr )
    free(ptr);
}
```

CTransportManager::StartReceive →

```
...
PSIMemFreeEx ✗
CSamsungVTBuffer::StartRTCPSend ✓
RTP_RtpCreate
CTransportManager::CreateBuffer
...
```

CTransportManager::StartReceive

```
if ( *((_BYTE *)this + 8448) )
  {
    started = CTransportManager::StartRTCPSend(this);
    if ( (_DWORD)started )
    {
```

CTransportManager::StartRTCPSend

```
...
      *((_BYTE *)this + 2326) = 1;
      v6 =
      PSICreateEventEx("vendor/samsung/external/ims_
video_engine/SVE/TransportMgr/src/TransportManager.
cpp", 1047LL);
      *((_QWORD *)this + 1122) = v6;
...
```

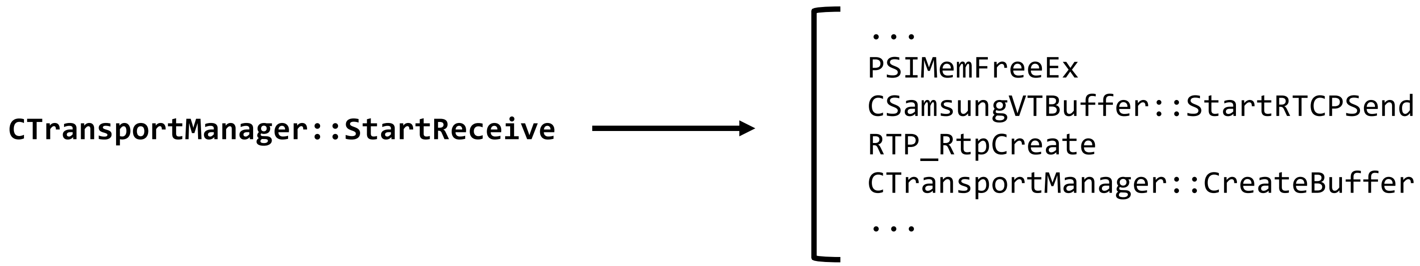
Extend nested structure

CTransportManager::CTransportManager

```
...
v4 = (CSamsungVTBuffer *)operator new(0x50uLL);
v5 = CSamsungVTBuffer::CSamsungVTBuffer(v4, "SendRtpPacket");
*((_QWORD *)this + 303) = v4;
*((_QWORD *)this + 1072) = 0LL;
...
```

Data structure database

```
struct CTransportManager

+0x978      pCSamsungVTBuffer;
+0x2180     field_2180;
+0x916      field_916;
+0x2310     eventHandle;
```



CTransportManager::StartReceive

```
...
PSIMemFreeEx
CSamsungVTBuffer::StartRTCPSend
RTP_RtpCreate
CTransportManager::CreateBuffer
...
```

Extend nested structure

CTransportManager::CTransportManager

```
...
v4 = (CSamsungVTBuffer *)operator new(0x50uLL);
v5 = CSamsungVTBuffer::CSamsungVTBuffer(v4, "SendRtpPacket");
*((_QWORD *)this + 303) = v4;
*((_QWORD *)this + 1072) = 0LL;
...
```

Data structure database

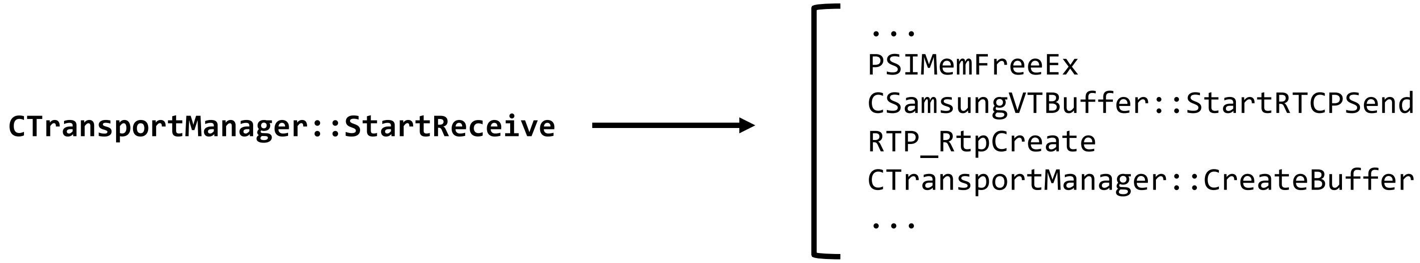
```
struct CTransportManager

+0x978      pCSamsungVTBuffer;
+0x2180     field_2180;
+0x916      field_916;
+0x2310     eventHandle;
```

CTransportManager::StartReceive

```
LODWORD(v93) = v31;
v32 = (CTransportManager *)RTP_RtpCreate(v89, &v85, &v83);
v33 = v85;
```

RTP_RtpCreate

```
...
  v6 = result;
  PSIMemset(result, 0LL, 216LL);
  *(_QWORD *)v6 = v6;
  *(_QWORD *)(v6 + 8) = *(_QWORD *)v5;
  *(_QWORD *)(v6 + 16) = *(_QWORD *)(v5 + 8);
...
 }
  return *(_QWORD *)v6;
}
```



CTransportManager::StartReceive →

```
...
PSIMemFreeEx
CSamsungVTBuffer::StartRTCPSend
RTP_RtpCreate
CTransportManager::CreateBuffer
...
```

Extend nested structure

CTransportManager::CTransportManager

```
...
v4 = (CSamsungVTBuffer *)operator new(0x50uLL);
v5 = CSamsungVTBuffer::CSamsungVTBuffer(v4, "SendRtpPacket");
*((_QWORD *)this + 303) = v4;
*((_QWORD *)this + 1072) = 0LL;
...
```

Data structure database

```
struct CTransportManager

+0x978      pCSamsungVTBuffer;
+0x2180     field_2180;
+0x916      field_916;
+0x2310     eventHandle;
```

CTransportManager::StartReceive

```
LODWORD(v93) = v31;
v32 = (CTransportManager *)RTP_RtpCreate(v89, &v85, &v83);
v33 = v85;
```

RTP_RtpCreate

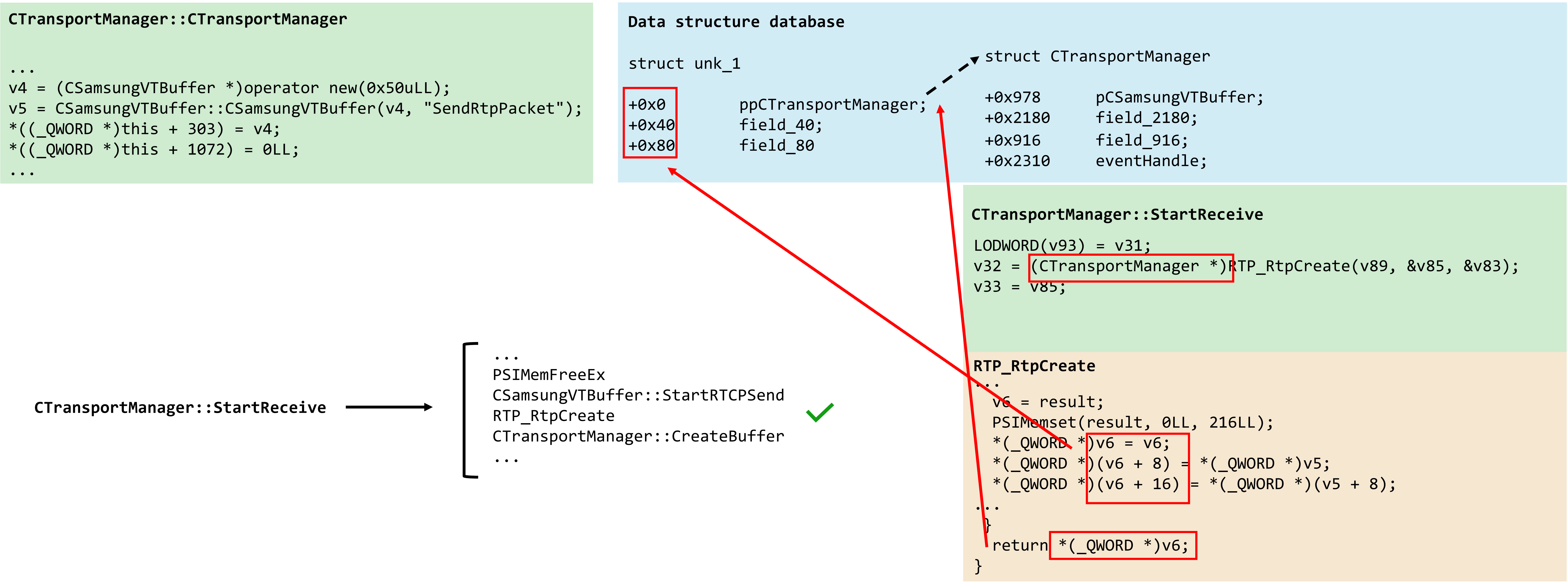
```
...
  v6 = result;
  PSIMemset(result, 0LL, 216LL);
  *(_QWORD *)v6 = v6;
  *(_QWORD *)(v6 + 8) = *(_QWORD *)v5;
  *(_QWORD *)(v6 + 16) = *(_QWORD *)(v5 + 8);
...
 }
  return *(_QWORD *)v6;
}
```

CTransportManager::StartReceive →

```
...
PSIMemFreeEx
CSamsungVTBuffer::StartRTCPSend
RTP_RtpCreate
CTransportManager::CreateBuffer
...
```

✓

Extend nested structure



CTransportManager::CTransportManager

```
...
v4 = (CSamsungVTBuffer *)operator new(0x50uLL);
v5 = CSamsungVTBuffer::CSamsungVTBuffer(v4, "SendRtpPacket");
*((_QWORD *)this + 303) = v4;
*((_QWORD *)this + 1072) = 0LL;
...
```

Data structure database

```
struct unk_1

+0x0        ppCTransportManager;
+0x40       field_40;
+0x80       field_80
```

```
struct CTransportManager

+0x978      pCSamsungVTBuffer;
+0x2180     field_2180;
+0x916      field_916;
+0x2310     eventHandle;
```

CTransportManager::StartReceive

```
LODWORD(v93) = v31;
v32 = (CTransportManager *)RTP_RtpCreate(v89, &v85, &v83);
v33 = v85;
```

RTP_RtpCreate

```
...
  v6 = result;
  PSIMemset(result, 0LL, 216LL);
  *(_QWORD *)v6 = v6;
  *(_QWORD *)(v6 + 8) = *(_QWORD *)v5;
  *(_QWORD *)(v6 + 16) = *(_QWORD *)(v5 + 8);
...
}
  return *(_QWORD *)v6;
}
```

CTransportManager::StartReceive →

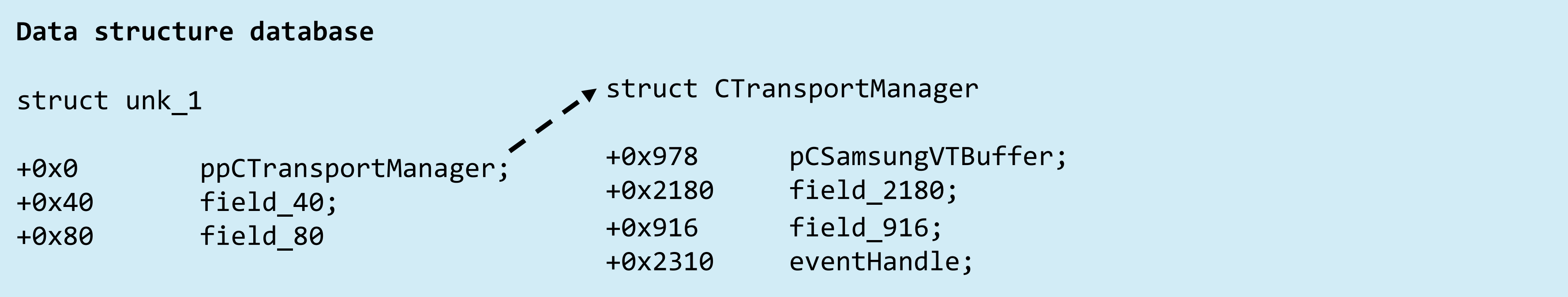
```
...
PSIMemFreeEx
CSamsungVTBuffer::StartRTCPSend
RTP_RtpCreate
CTransportManager::CreateBuffer
...
```

✓

Match offset & LLM infer

CTransportManager::CTransportManager

```
...
v4 = (CSamsungVTBuffer *)operator new(0x50uLL);
v5 = CSamsungVTBuffer::CSamsungVTBuffer(v4, "SendRtpPacket");
*((_QWORD *)this + 303) = v4;
*((_QWORD *)this + 1072) = 0LL;
...
```



Data structure database

```
struct unk_1

+0x0        ppCTransportManager;
+0x40       field_40;
+0x80       field_80
```

```
struct CTransportManager

+0x978      pCSamsungVTBuffer;
+0x2180     field_2180;
+0x916      field_916;
+0x2310     eventHandle;
```

CTransportManager::StartReceive →

```
...
PSIMemFreeEx
CSamsungVTBuffer::StartRTCPSend
RTP_RtpCreate
CTransportManager::CreateBuffer
...
```

Match offset & LLM infer

CTransportManager::CTransportManager

```
...
v4 = (CSamsungVTBuffer *)operator new(0x50uLL);
v5 = CSamsungVTBuffer::CSamsungVTBuffer(v4, "SendRtpPacket");
*((_QWORD *)this + 303) = v4;
*((_QWORD *)this + 1072) = 0LL;
...
```

Data structure database

```
struct unk_1

+0x0        ppCTransportManager;
+0x40       field_40;
+0x80       field_80
```

```
struct CTransportManager

+0x978      pCSamsungVTBuffer;
+0x2180     field_2180;
+0x916      field_916;
+0x2310     eventHandle;
```

CTransportManager::StartReceive →

```
...
PSIMemFreeEx
CSamsungVTBuffer::StartRTCPSend
RTP_RtpCreate
CTransportManager::CreateBuffer
...
```

CTransportManager::StartReceive

```
if ( *((_BYTE *)this + 2329) )
{
  Buffer = CSamsungVTBuffer::CreateBuffer(*((CSamsungVTBuffer
**)this + 303));
  v77 = *((_BYTE *)this + 2329) == 0;
}
```

CSamsungVTBuffer::CreateBuffer(CSamsungVTBuffer *this)

```
PSILockMutexTimeoutEx(
    *((_QWORD *)this + 5),
    0xFFFFFFFFLL,
"vendor/samsung/external/ims_video_engine/SVE/FEC/src/Samsung
VTBuffer.cpp",
    120LL);
  v3 = *((_QWORD *)this + 6);
  *((_DWORD *)this + 14) = 2;
  PSIResetEvent(v3);
```

Match offset & LLM infer

CTransportManager::CTransportManager

```
...
v4 = (CSamsungVTBuffer *)operator new(0x50uLL);
v5 = CSamsungVTBuffer::CSamsungVTBuffer(v4, "SendRtpPacket");
*((_QWORD *)this + 303) = v4;
*((_QWORD *)this + 1072) = 0LL;
...
```

Data structure database

```
struct unk_1                                     struct CTransportManager

+0x0        ppCTransportManager;                 +0x978      pCSamsungVTBuffer;
+0x40       field_40;                            +0x2180     field_2180;
+0x80       field_80                             +0x916      field_916;
                                                 +0x2310     eventHandle;
```

CTransportManager::StartReceive

```
if ( *((_BYTE *)this + 2329) )
{
  Buffer = CSamsungVTBuffer::CreateBuffer(*((CSamsungVTBuffer
**)this + 303));
  v77 = *((_BYTE *)this + 2329) == 0;
}
```

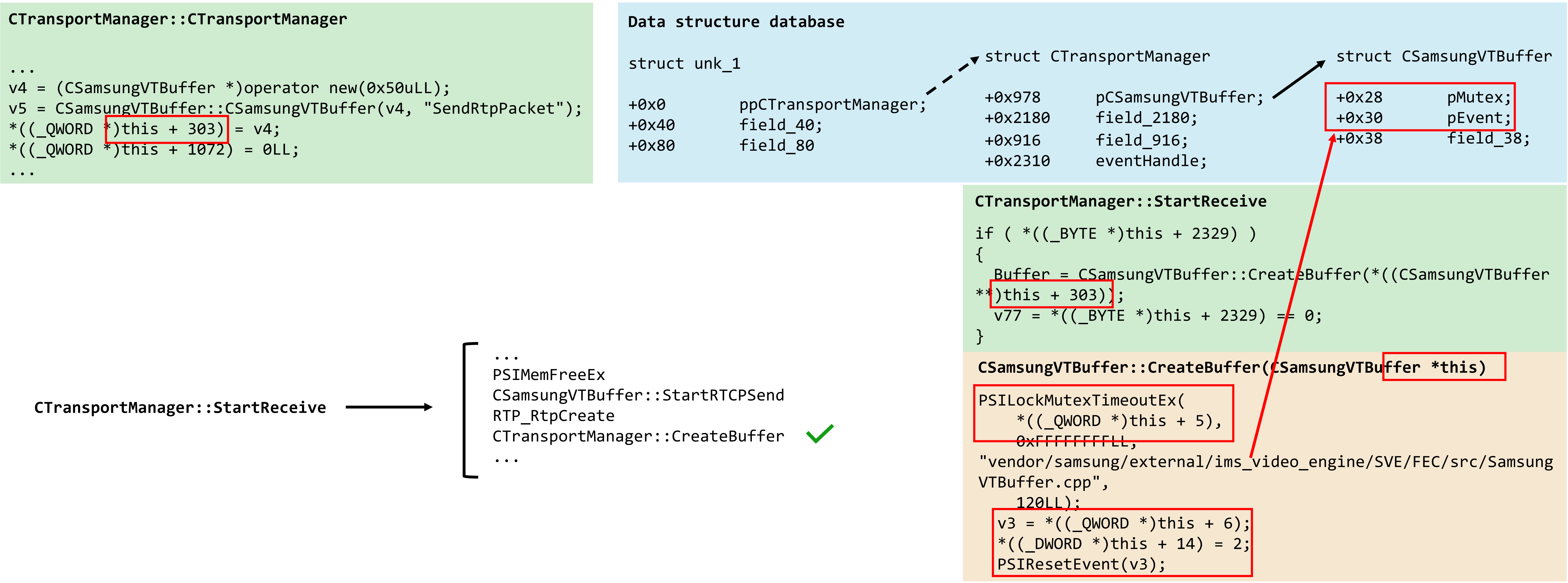
CSamsungVTBuffer::CreateBuffer(CSamsungVTBuffer *this)

```
PSILockMutexTimeoutEx(
    *((_QWORD *)this + 5),
    0xFFFFFFFFLL,
"vendor/samsung/external/ims_video_engine/SVE/FEC/src/Samsung
VTBuffer.cpp",
    120LL);
  v3 = *((_QWORD *)this + 6);
  *((_DWORD *)this + 14) = 2;
  PSIResetEvent(v3);
```

CTransportManager::StartReceive →

```
...
PSIMemFreeEx
CSamsungVTBuffer::StartRTCPSend
RTP_RtpCreate
CTransportManager::CreateBuffer ✓
...
```

Match offset & LLM infer



CTransportManager::CTransportManager

```
...
v4 = (CSamsungVTBuffer *)operator new(0x50uLL);
v5 = CSamsungVTBuffer::CSamsungVTBuffer(v4, "SendRtpPacket");
*((_QWORD *)this + 303) = v4;
*((_QWORD *)this + 1072) = 0LL;
...
```

Data structure database

```
struct unk_1

+0x0        ppCTransportManager;
+0x40       field_40;
+0x80       field_80
```

```
struct CTransportManager

+0x978      pCSamsungVTBuffer;
+0x2180     field_2180;
+0x916      field_916;
+0x2310     eventHandle;
```

```
struct CSamsungVTBuffer

+0x28       pMutex;
+0x30       pEvent;
+0x38       field_38;
```

CTransportManager::StartReceive

```
if ( *((_BYTE *)this + 2329) )
{
  Buffer = CSamsungVTBuffer::CreateBuffer(*((CSamsungVTBuffer **)this + 303));
  v77 = *((_BYTE *)this + 2329) == 0;
}
```

CSamsungVTBuffer::CreateBuffer(CSamsungVTBuffer *this)

```
PSILockMutexTimeoutEx(
    *((_QWORD *)this + 5),
    0xFFFFFFFFLL,
"vendor/samsung/external/ims_video_engine/SVE/FEC/src/SamsungVTBuffer.cpp",
    120LL);
 v3 = *((_QWORD *)this + 6);
 *((_DWORD *)this + 14) = 2;
 PSIResetEvent(v3);
```

CTransportManager::StartReceive →

```
...
PSIMemFreeEx
CSamsungVTBuffer::StartRTCPSend
RTP_RtpCreate
CTransportManager::CreateBuffer  ✓
...
```

Vulnerability Analyzer [AI]

Bug report = Agent(prompt template + input_data + pre-defined bug model)

- Input Data + Background introduction

```
prompt = <code>, this is a data-processing function’s code, <arg_list>, <arg_type_list>, are this function’s parameter
descriptions, <datastruct_list>, these are the data structures involved in the function, <parent_code>, this is the parent
function’s code, <parent_arg_list>, these are the parent function’s parameter descriptions
[Preliminary knowledge of memory vulnerabilities]
- Memory corruption vulnerabilities include out-of-bounds reads and writes. The following are example vulnerability
patterns: <bug_model>
```

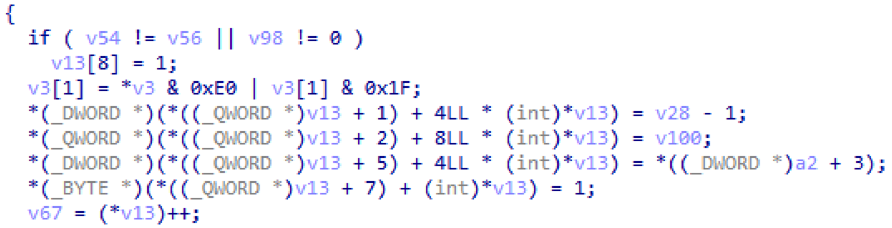
- Task Description

```
- First identify the data structures of the processing function’s parameters to aid your code analysis. Analyze the parent
function code together with the processing function code, focusing on whether the processing function contains a
vulnerability.
- Then, based on the preliminary vulnerability knowledge and the provided vulnerability patterns, determine whether any
pattern applies. Explain in detail the vulnerability’s mechanism and its root cause, and provide a confidence level for
your conclusion.
- If there are multiple code branches, analyze each branch separately for known vulnerability patterns.
- The reasoning process must be extremely detailed, explaining your rationale at each step.
```

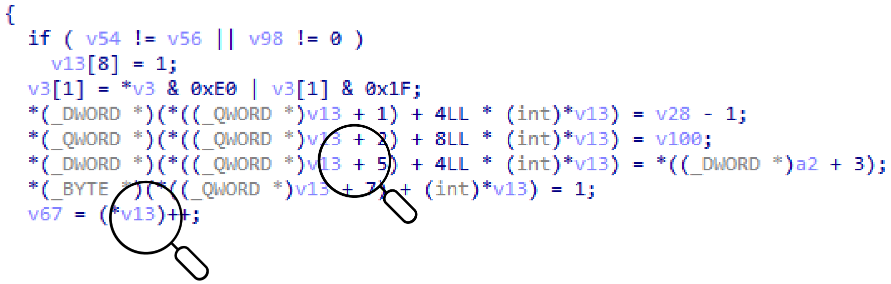
- Format requirements: JSON

Core

Is the following code vulnerable according to predefined bug model? And each var’s value range is as following…



```
{
  if ( v54 != v56 || v98 != 0 )
    v13[8] = 1;
  v3[1] = *v3 & 0xE0 | v3[1] & 0x1F;
  *(_DWORD *)(*((_QWORD *)v13 + 1) + 4LL * (int)*v13) = v28 - 1;
  *(_QWORD *)(*((_QWORD *)v13 + 2) + 8LL * (int)*v13) = v100;
  *(_DWORD *)(*((_QWORD *)v13 + 5) + 4LL * (int)*v13) = *((_DWORD *)a2 + 3);
  *(_BYTE *)(*((_QWORD *)v13 + 7) + (int)*v13) = 1;
  v67 = (*v13)++;
```



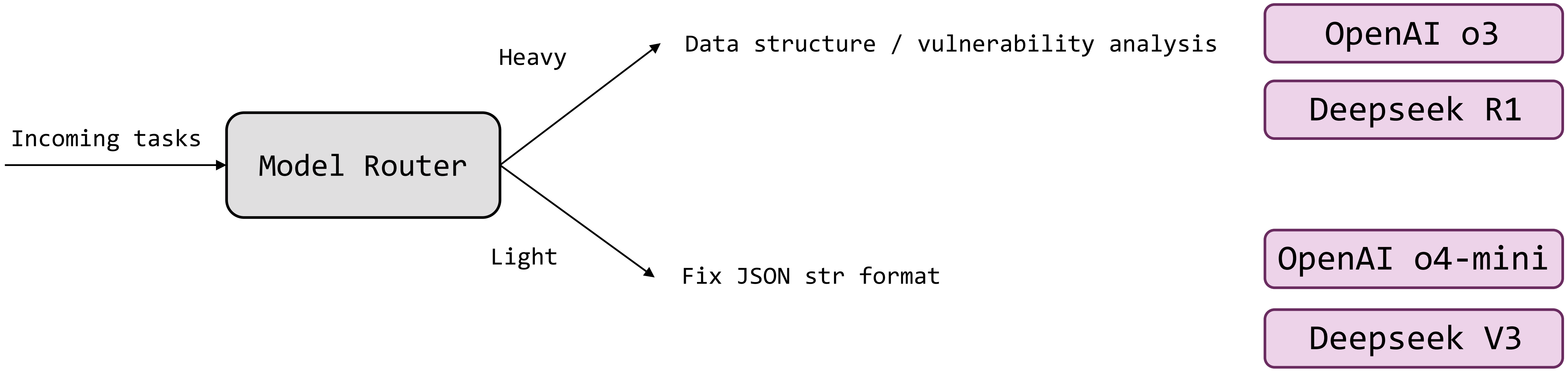
```
{
  if ( v54 != v56 || v98 != 0 )
    v13[8] = 1;
  v3[1] = *v3 & 0xE0 | v3[1] & 0x1F;
  *(_DWORD *)(*((_QWORD *)v13 + 1) + 4LL * (int)*v13) = v28 - 1;
  *(_QWORD *)(*((_QWORD *)v13 + 2) + 8LL * (int)*v13) = v100;
  *(_DWORD *)(*((_QWORD *)v13 + 5) + 4LL * (int)*v13) = *((_DWORD *)a2 + 3);
  *(_BYTE *)(*((_QWORD *)v13 + 7) + (int)*v13) = 1;
  v67 = (*v13)++;
```

Vulnerability Analyzer

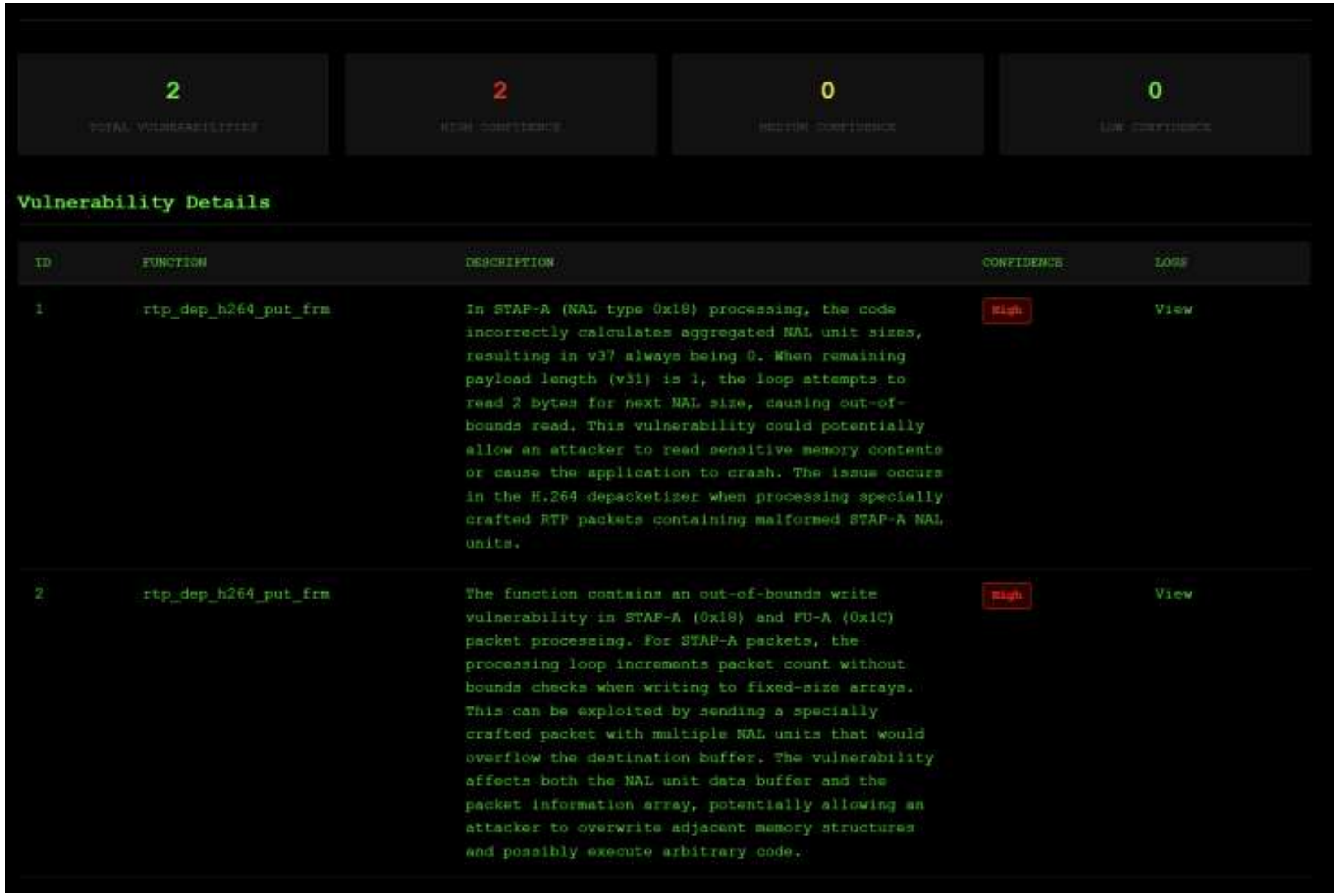
The user provided a code snippet about …, in which v13 is declared as an int* and can range from 0 to 0xFFFFFFFF. However, the buffer accessed via *((QWORD*)v13 + 5) has a fixed length of 0x3E8 elements, so indexing beyond that limit will cause an out-of-bounds write.

Module Router

- Dispatch different tasks to corresponding model
- Heavy task: Reasoning models
- Light task: Cost-optimized models



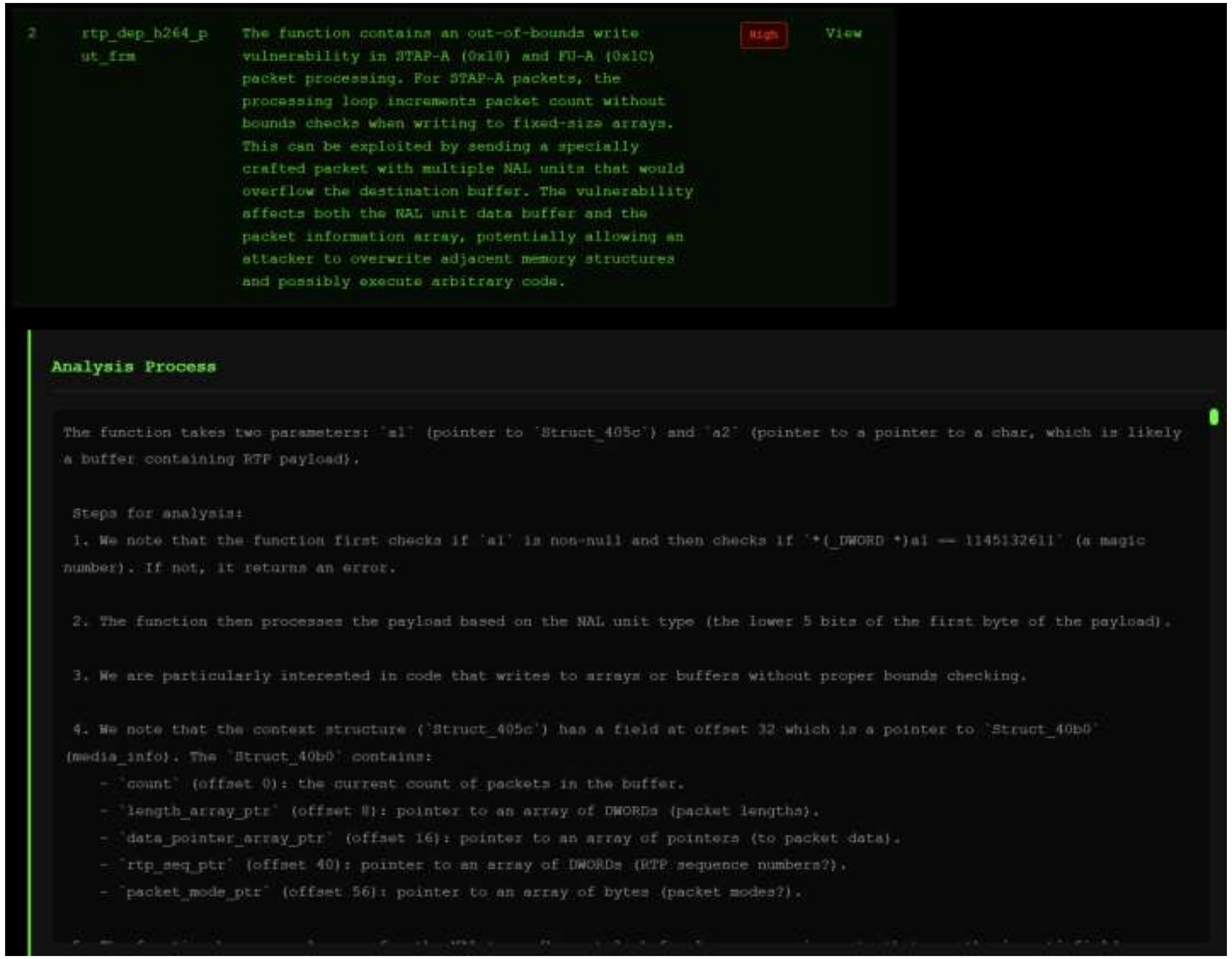
Result Report



2	2	0	0
TOTAL VULNERABILITIES	HIGH CONFIDENCE	MEDIUM CONFIDENCE	LOW CONFIDENCE

Vulnerability Details

ID	FUNCTION	DESCRIPTION	CONFIDENCE	LOGS
1	rtp_dep_h264_put_frm	In STAP-A (NAL type 0x18) processing, the code incorrectly calculates aggregated NAL unit sizes, resulting in v37 always being 0. When remaining payload length (v31) is 1, the loop attempts to read 2 bytes for next NAL size, causing out-of-bounds read. This vulnerability could potentially allow an attacker to read sensitive memory contents or cause the application to crash. The issue occurs in the H.264 depacketizer when processing specially crafted RTP packets containing malformed STAP-A NAL units.	High	View
2	rtp_dep_h264_put_frm	The function contains an out-of-bounds write vulnerability in STAP-A (0x18) and FU-A (0x1C) packet processing. For STAP-A packets, the processing loop increments packet count without bounds checks when writing to fixed-size arrays. This can be exploited by sending a specially crafted packet with multiple NAL units that would overflow the destination buffer. The vulnerability affects both the NAL unit data buffer and the packet information array, potentially allowing an attacker to overwrite adjacent memory structures and possibly execute arbitrary code.	High	View



2	rtp_dep_h264_put_frm	The function contains an out-of-bounds write vulnerability in STAP-A (0x18) and FU-A (0x1C) packet processing. For STAP-A packets, the processing loop increments packet count without bounds checks when writing to fixed-size arrays. This can be exploited by sending a specially crafted packet with multiple NAL units that would overflow the destination buffer. The vulnerability affects both the NAL unit data buffer and the packet information array, potentially allowing an attacker to overwrite adjacent memory structures and possibly execute arbitrary code.	High	View

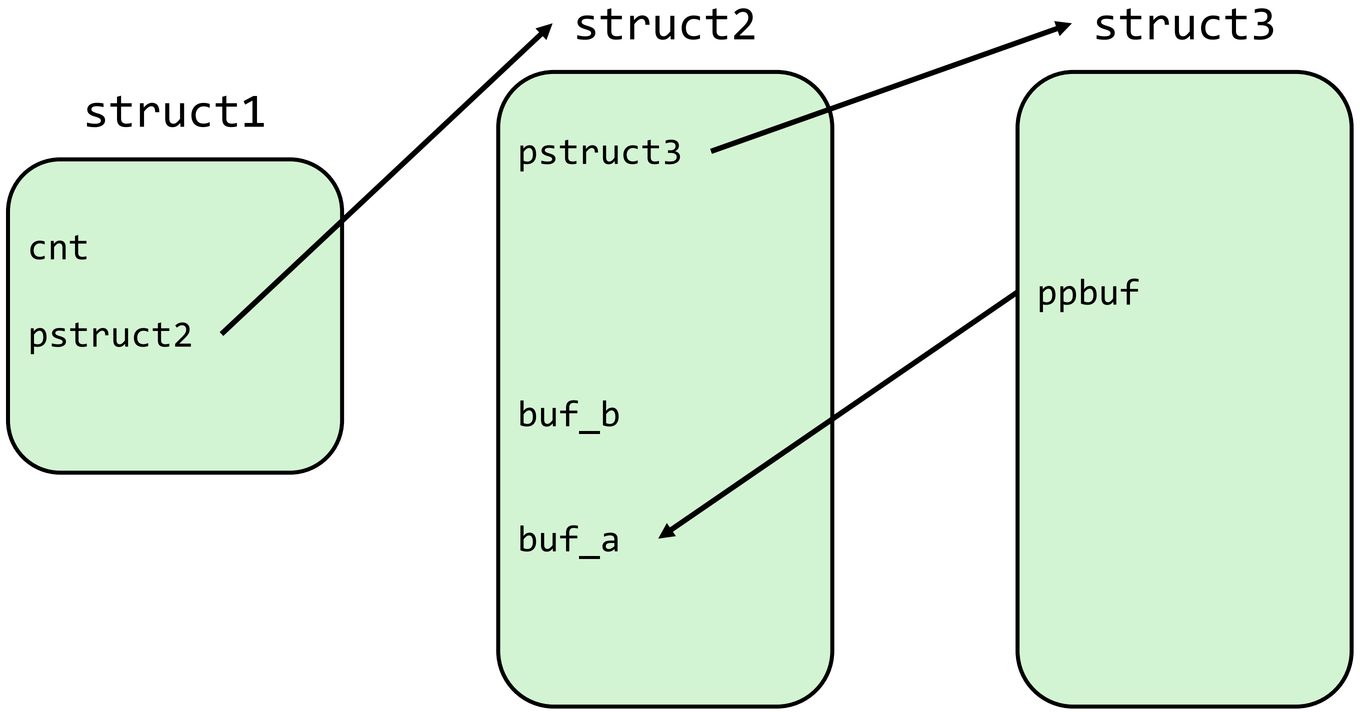
Analysis Process

```
The function takes two parameters: `a1` (pointer to `Struct_405c`) and `a2` (pointer to a pointer to a char, which is likely
a buffer containing RTP payload).

 Steps for analysis:
 1. We note that the function first checks if `a1` is non-null and then checks if `*(_DWORD *)a1 == 1145132611` (a magic
number). If not, it returns an error.

 2. The function then processes the payload based on the NAL unit type (the lower 5 bits of the first byte of the payload).

 3. We are particularly interested in code that writes to arrays or buffers without proper bounds checking.

 4. We note that the context structure (`Struct_405c`) has a field at offset 32 which is a pointer to `Struct_40b0`
(media_info). The `Struct_40b0` contains:
    - `count` (offset 0): the current count of packets in the buffer.
    - `length_array_ptr` (offset 8): pointer to an array of DWORDs (packet lengths).
    - `data_pointer_array_ptr` (offset 16): pointer to an array of pointers (to packet data).
    - `rtp_seq_ptr` (offset 40): pointer to an array of DWORDs (RTP sequence numbers?).
    - `packet_mode_ptr` (offset 56): pointer to an array of bytes (packet modes?).
```

Applying LLMs to Real-World Vulnerability Analysis

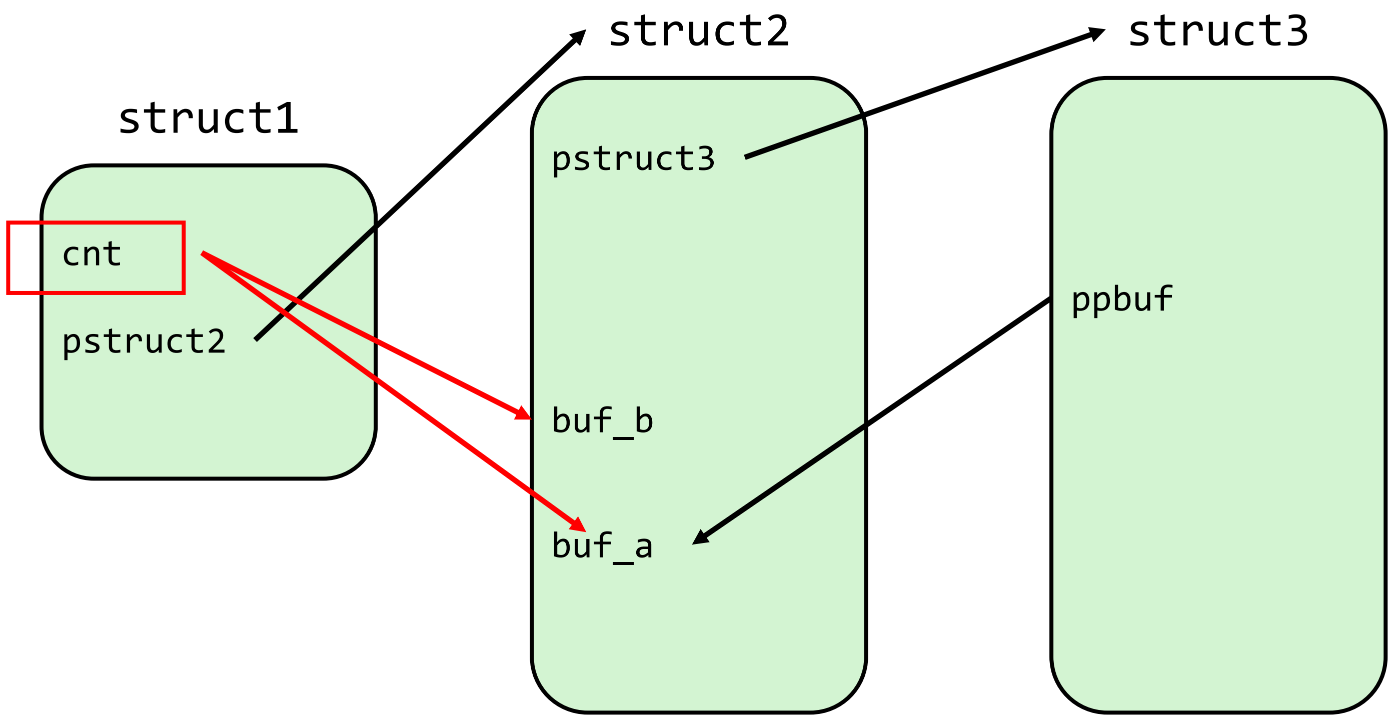
Bug pattern

Bug pattern



```
struct2->buf_a = malloc(N1)
struct2->buf_b = malloc(N2)
```

Bug pattern



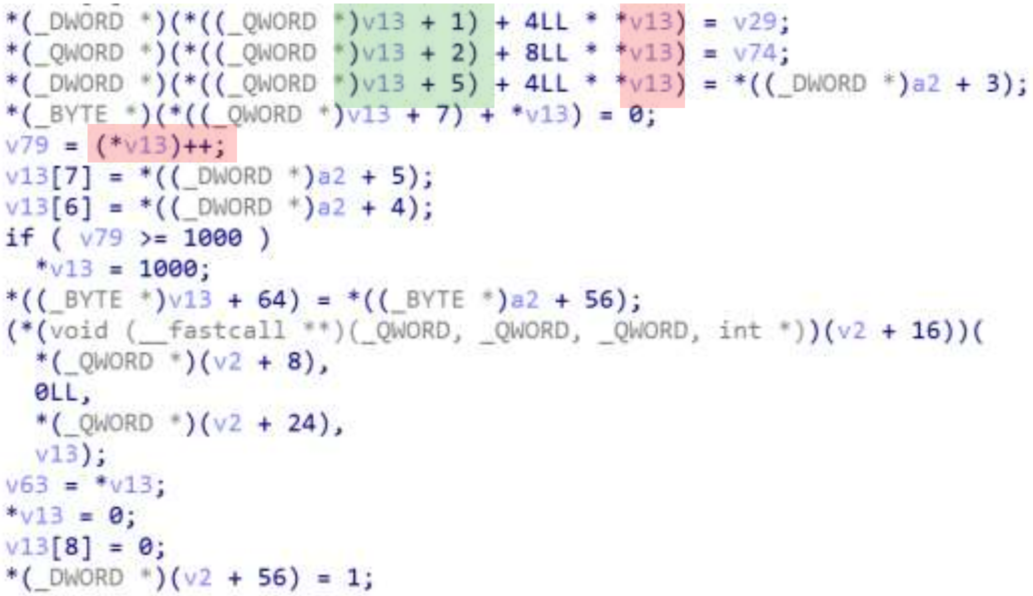
```
struct2->buf_a = malloc(N1)
struct2->buf_b = malloc(N2)
...
struct1->cnt++;
...
v1 = struct1->cnt
struct2->buf_a[v1] = xxx;
bbb = struct2->buf_b[v1];
```

Bug model: for (int v1 = 0; v1 < MAX_LENGTH; v1++) { buf_a[v1] = xxx; bbb = buf_b[v1]; } – if v1 is greater than or equal to the number of elements in buf_a, it’s an out-of-bounds write; if v1 is greater than or equal to the number of elements in buf_b, it’s an out-of-bounds read. You must know that buf_a’s element count is less than or equal to v1 to confirm an out-of-bounds write, and that buf_b’s element count is less than or equal to v1 to confirm an out-of-bounds read. Focus on code involving arrays and ensure indices never exceed their bounds.

Bug1 SVE-2024-1490

Bug1 SVE-2024-1490

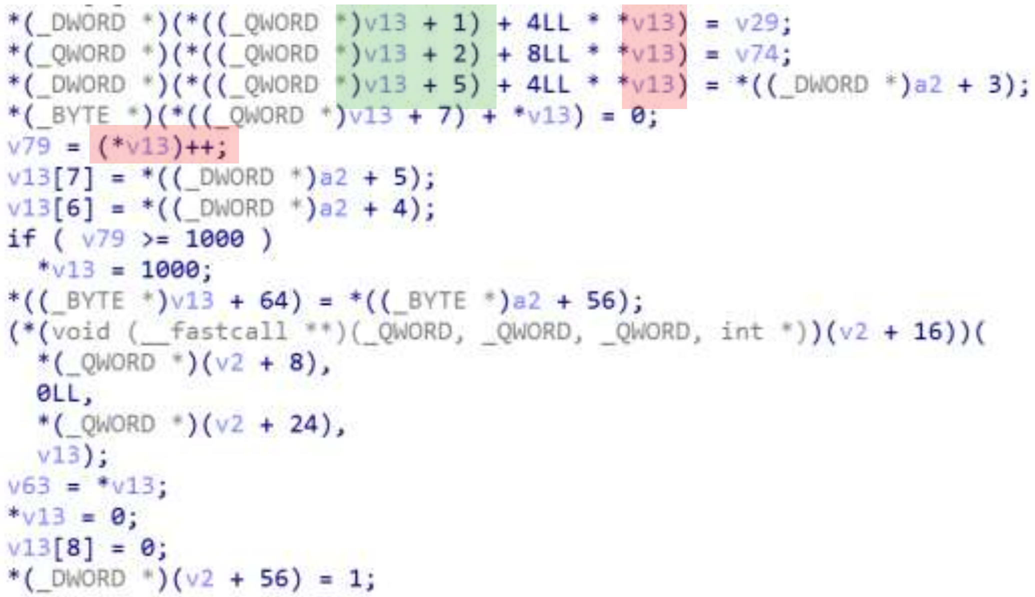
`rtp_dep_h264_put_frm`



```
*(_DWORD *)(*((_QWORD *)v13 + 1) + 4LL * *v13) = v29;
*(_QWORD *)(*((_QWORD *)v13 + 2) + 8LL * *v13) = v74;
*(_DWORD *)(*((_QWORD *)v13 + 5) + 4LL * *v13) = *((_DWORD *)a2 + 3);
*(_BYTE *)(*((_QWORD *)v13 + 7) + *v13) = 0;
v79 = (*v13)++;
v13[7] = *((_DWORD *)a2 + 5);
v13[6] = *((_DWORD *)a2 + 4);
if ( v79 >= 1000 )
  *v13 = 1000;
*((_BYTE *)v13 + 64) = *((_BYTE *)a2 + 56);
(*(void (__fastcall **)(_QWORD, _QWORD, _QWORD, int *))(v2 + 16))(
  *(_QWORD *)(v2 + 8),
  0LL,
  *(_QWORD *)(v2 + 24),
  v13);
v63 = *v13;
*v13 = 0;
v13[8] = 0;
*(_DWORD *)(v2 + 56) = 1;
```

Bug1 SVE-2024-1490

rtp_dep_h264_put_frm

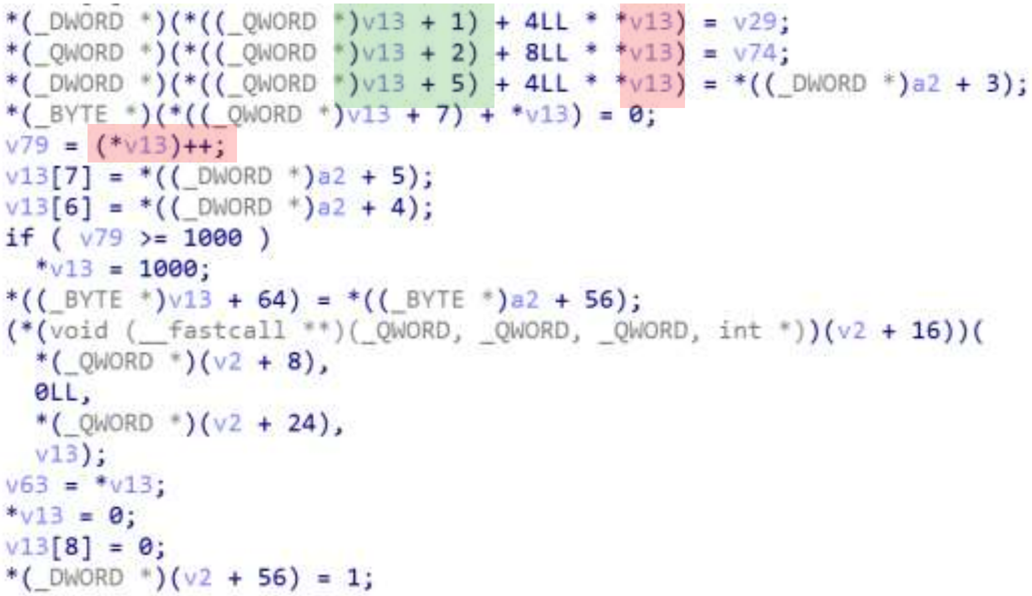


```
*(_DWORD *)(*((_QWORD *)v13 + 1) + 4LL * *v13) = v29;
*(_QWORD *)(*((_QWORD *)v13 + 2) + 8LL * *v13) = v74;
*(_DWORD *)(*((_QWORD *)v13 + 5) + 4LL * *v13) = *((_DWORD *)a2 + 3);
*(_BYTE *)(*((_QWORD *)v13 + 7) + *v13) = 0;
v79 = (*v13)++;
v13[7] = *((_DWORD *)a2 + 5);
v13[6] = *((_DWORD *)a2 + 4);
if ( v79 >= 1000 )
  *v13 = 1000;
*((_BYTE *)v13 + 64) = *((_BYTE *)a2 + 56);
(*(void (__fastcall **)(_QWORD, _QWORD, _QWORD, int *))(v2 + 16))(
  *(_QWORD *)(v2 + 8),
  0LL,
  *(_QWORD *)(v2 + 24),
  v13);
v63 = *v13;
*v13 = 0;
v13[8] = 0;
*(_DWORD *)(v2 + 56) = 1;
```

```
{ "name": "Struct_40b0",
        "ds_description": "Media information structure storing H.264 packet buffers and metadata arrays",
        "malloc_func":"rtp_dep_h264_create",
        "items": [  {"name": "count","type": "QWORD","offset": 0,"value": 0,"description":""},
                    {"name": "length_array_ptr","type": "DWORD *","offset": 8,"value": "","description":“4000"},
                    {"name": "data_pointer_array_ptr","type": "char **","offset": 16,"value": "","description":“8000"},
                    {"name": "unknown_24","type": "QWORD","offset": 24,"value": "0","description":""},
                    {"name": "unknown_32","type": "DWORD","offset": 32,"value": "0","description":""},
                    {"name": "field_36","type": "DWORD","offset": 36,"value": "*a1","description":""},
                    {"name": "rtp_seq_ptr","type": "DWORD *","offset": 40,"value": "","description":“1000"},
                    {"name": "unknown_48","type": "QWORD","offset": 48,"value": "0","description":""},
                    {"name": "packet_mode_ptr","type": "char *","offset": 56,"value": "","description":“1024"},
                    {"name": "unknown_64","type": "QWORD","offset": 64,"value": "0","description":""}]
}
```

Bug1 SVE-2024-1490

`rtp_dep_h264_put_frm`



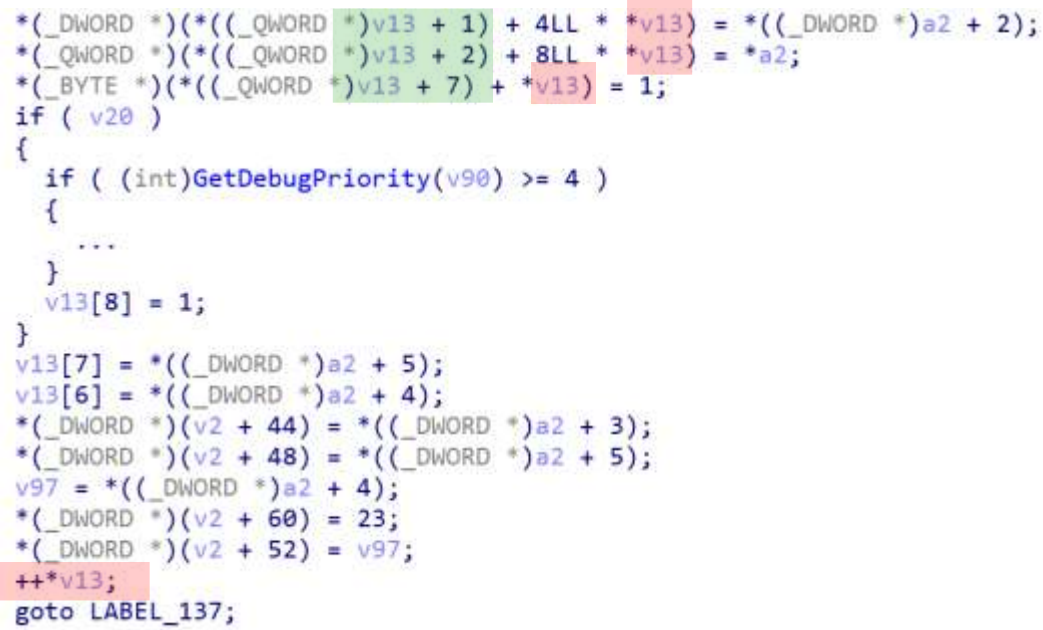
```
*(_DWORD *)(*((_QWORD *)v13 + 1) + 4LL * *v13) = v29;
*(_QWORD *)(*((_QWORD *)v13 + 2) + 8LL * *v13) = v74;
*(_DWORD *)(*((_QWORD *)v13 + 5) + 4LL * *v13) = *((_DWORD *)a2 + 3);
*(_BYTE *)(*((_QWORD *)v13 + 7) + *v13) = 0;
v79 = (*v13)++;
v13[7] = *((_DWORD *)a2 + 5);
v13[6] = *((_DWORD *)a2 + 4);
if ( v79 >= 1000 )
  *v13 = 1000;
*((_BYTE *)v13 + 64) = *((_BYTE *)a2 + 56);
(*(void (__fastcall **)(_QWORD, _QWORD, _QWORD, int *))(v2 + 16))(
  *(_QWORD *)(v2 + 8),
  0LL,
  *(_QWORD *)(v2 + 24),
  v13);
v63 = *v13;
*v13 = 0;
v13[8] = 0;
*(_DWORD *)(v2 + 56) = 1;
```

```
{ "name": "Struct_40b0",
        "ds_description": "Media information structure storing H.264 packet buffers and metadata arrays",
        "malloc_func":"rtp_dep_h264_create",
        "items": [  {"name": "count","type": "QWORD","offset": 0,"value": 0,"description":""},
                    {"name": "length_array_ptr","type": "DWORD *","offset": 8,"value": "","description":“4000"},
                    {"name": "data_pointer_array_ptr","type": "char **","offset": 16,"value": "","description":“8000"},
                    {"name": "unknown_24","type": "QWORD","offset": 24,"value": "0","description":""},
                    {"name": "unknown_32","type": "DWORD","offset": 32,"value": "0","description":""},
                    {"name": "field_36","type": "DWORD","offset": 36,"value": "*a1","description":""},
                    {"name": "rtp_seq_ptr","type": "DWORD *","offset": 40,"value": "","description":“1000"},
                    {"name": "unknown_48","type": "QWORD","offset": 48,"value": "0","description":""},
                    {"name": "packet_mode_ptr","type": "char *","offset": 56,"value": "","description":“1024"},
                    {"name": "unknown_64","type": "QWORD","offset": 64,"value": "0","description":""}]
}
```

The function contains an out-of-bounds write vulnerability in STAP-A (0x18) and FU-A (0x1C) packet processing. For STAP-A packets, the processing loop increments packet count without bounds checks when writing to fixed-size arrays. This can be exploited by sending a specially crafted packet with multiple NAL units that would overflow the destination buffer. The vulnerability affects both the NAL unit data buffer and the packet information array, potentially allowing an attacker to overwrite adjacent memory structures and possibly execute arbitrary code.

Bug2 SVE-2024-1492

rtp_dep_h264_put_frm



```
*(_DWORD *)(*((_QWORD *)v13 + 1) + 4LL * *v13) = *((_DWORD *)a2 + 2);
*(_QWORD *)(*((_QWORD *)v13 + 2) + 8LL * *v13) = *a2;
*(_BYTE *)(*((_QWORD *)v13 + 7) + *v13) = 1;
if ( v20 )
{
  if ( (int)GetDebugPriority(v90) >= 4 )
  {
    ...
  }
  v13[8] = 1;
}
v13[7] = *((_DWORD *)a2 + 5);
v13[6] = *((_DWORD *)a2 + 4);
*(_DWORD *)(v2 + 44) = *((_DWORD *)a2 + 3);
*(_DWORD *)(v2 + 48) = *((_DWORD *)a2 + 5);
v97 = *((_DWORD *)a2 + 4);
*(_DWORD *)(v2 + 60) = 23;
*(_DWORD *)(v2 + 52) = v97;
++*v13;
goto LABEL_137;
```

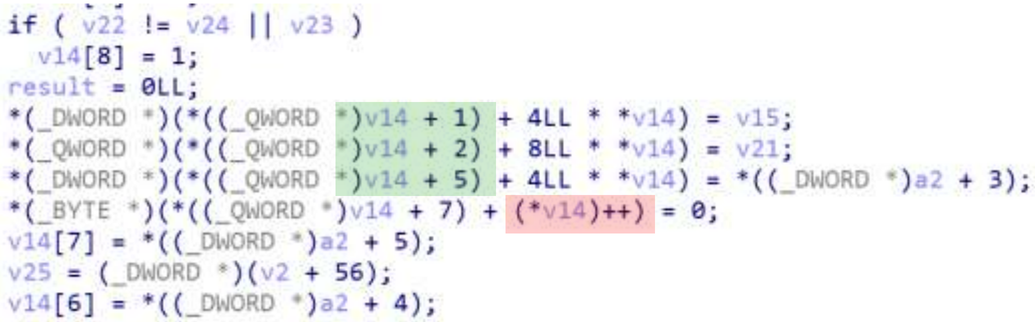
```
{ "name": "Struct_40b0",
        "ds_description": "Media information structure storing H.264 packet buffers and metadata arrays",
        "malloc_func":"rtp_dep_h264_create",
        "items": [  {"name": "count","type": "QWORD","offset": 0,"value": 0,"description":""},
                    {"name": "length_array_ptr","type": "DWORD *","offset": 8,"value": "","description":"4000"},
                    {"name": "data_pointer_array_ptr","type": "char **","offset": 16,"value": "","description":"8000"},
                    {"name": "unknown_24","type": "QWORD","offset": 24,"value": "0","description":""},
                    {"name": "unknown_32","type": "DWORD","offset": 32,"value": "0","description":""},
                    {"name": "field_36","type": "DWORD","offset": 36,"value": "*a1","description":""},
                    {"name": "rtp_seq_ptr","type": "DWORD *","offset": 40,"value": "","description":"1000"},
                    {"name": "unknown_48","type": "QWORD","offset": 48,"value": "0","description":""},
                    {"name": "packet_mode_ptr","type": "char *","offset": 56,"value": "","description":"1024"},
                    {"name": "unknown_64","type": "QWORD","offset": 64,"value": "0","description":""}]
}
```

m_info

+0x0:	count (v13)	
+0x8:	buffer (4LL * (int)rtp_dep_obj[1])	payload size
+0x10:	buffer3(8LL * (int)rtp_dep_obj[1]	payload buf
+0x24:	media type	
+0x28:	buffer2 (4LL * (int)rtp_dep_obj[1])	seq
+0x38:	buffer4 ((int)rtp_dep_obj[1])	

Bug3 SVE-2024-1494

rtp_dep_h265_put_frm



```
if ( v22 != v24 || v23 )
  v14[8] = 1;
result = 0LL;
*(_DWORD *)(*((_QWORD *)v14 + 1) + 4LL * *v14) = v15;
*(_QWORD *)(*((_QWORD *)v14 + 2) + 8LL * *v14) = v21;
*(_DWORD *)(*((_QWORD *)v14 + 5) + 4LL * *v14) = *((_DWORD *)a2 + 3);
*(_BYTE *)(*((_QWORD *)v14 + 7) + (*v14)++) = 0;
v14[7] = *((_DWORD *)a2 + 5);
v25 = (_DWORD *)(v2 + 56);
v14[6] = *((_DWORD *)a2 + 4);
```

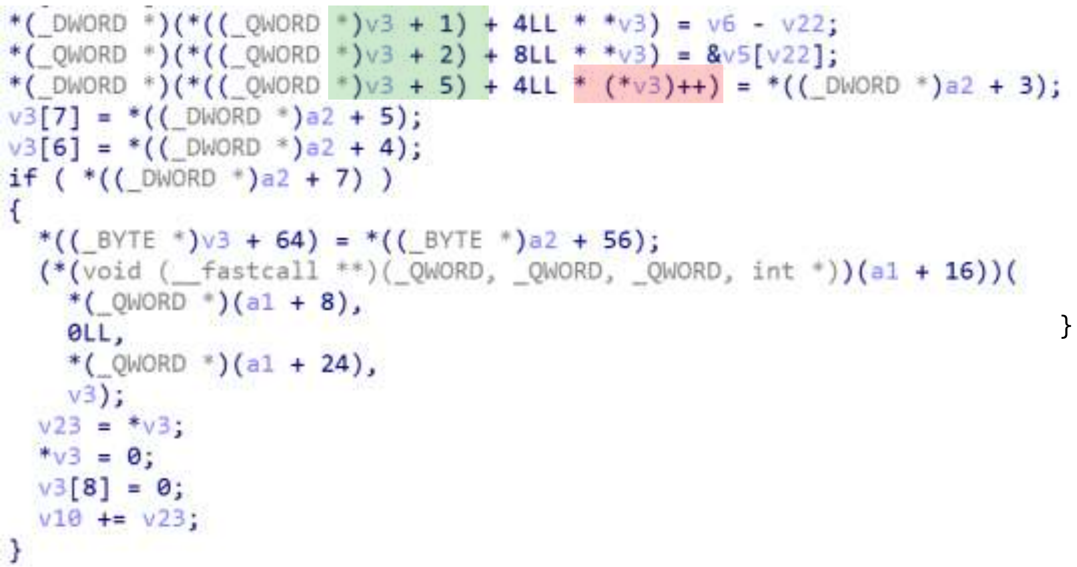
```
{ "name": "Struct_40b0",
        "ds_description": "Media information structure storing H.264 packet buffers and metadata arrays",
        "malloc_func":"rtp_dep_h264_create",
        "items": [  {"name": "count","type": "QWORD","offset": 0,"value": 0,"description":""},
                    {"name": "length_array_ptr","type": "DWORD *","offset": 8,"value": "","description":“4000"},
                    {"name": "data_pointer_array_ptr","type": "char **","offset": 16,"value": "","description":“8000"},
                    {"name": "unknown_24","type": "QWORD","offset": 24,"value": "0","description":""},
                    {"name": "unknown_32","type": "DWORD","offset": 32,"value": "0","description":""},
                    {"name": "field_36","type": "DWORD","offset": 36,"value": "*a1","description":""},
                    {"name": "rtp_seq_ptr","type": "DWORD *","offset": 40,"value": "","description":“1000"},
                    {"name": "unknown_48","type": "QWORD","offset": 48,"value": "0","description":""},
                    {"name": "packet_mode_ptr","type": "char *","offset": 56,"value": "","description":“1024"},
                    {"name": "unknown_64","type": "QWORD","offset": 64,"value": "0","description":""}]
}
```

m_info

+0x0:	count (v14)	
+0x8:	buffer (4LL * (int)rtp_dep_obj[1])	payload size
+0x10:	buffer3(8LL * (int)rtp_dep_obj[1]	payload buf
+0x24:	media type	
+0x28:	buffer2 (4LL * (int)rtp_dep_obj[1])	seq
+0x38:	buffer4 ((int)rtp_dep_obj[1])	

Bug4 SVE-2024-1495

rtp_dep_h263_put_frm



```
*(_DWORD *)(*((_QWORD *)v3 + 1) + 4LL * *v3) = v6 - v22;
*(_QWORD *)(*((_QWORD *)v3 + 2) + 8LL * *v3) = &v5[v22];
*(_DWORD *)(*((_QWORD *)v3 + 5) + 4LL * (*v3)++) = *((_DWORD *)a2 + 3);
v3[7] = *((_DWORD *)a2 + 5);
v3[6] = *((_DWORD *)a2 + 4);
if ( *((_DWORD *)a2 + 7) )
{
  *((_BYTE *)v3 + 64) = *((_BYTE *)a2 + 56);
  (*(void (__fastcall **)(_QWORD, _QWORD, _QWORD, int *))(a1 + 16))(
    *(_QWORD *)(a1 + 8),
    0LL,
    *(_QWORD *)(a1 + 24),
    v3);
  v23 = *v3;
  *v3 = 0;
  v3[8] = 0;
  v10 += v23;
}
```

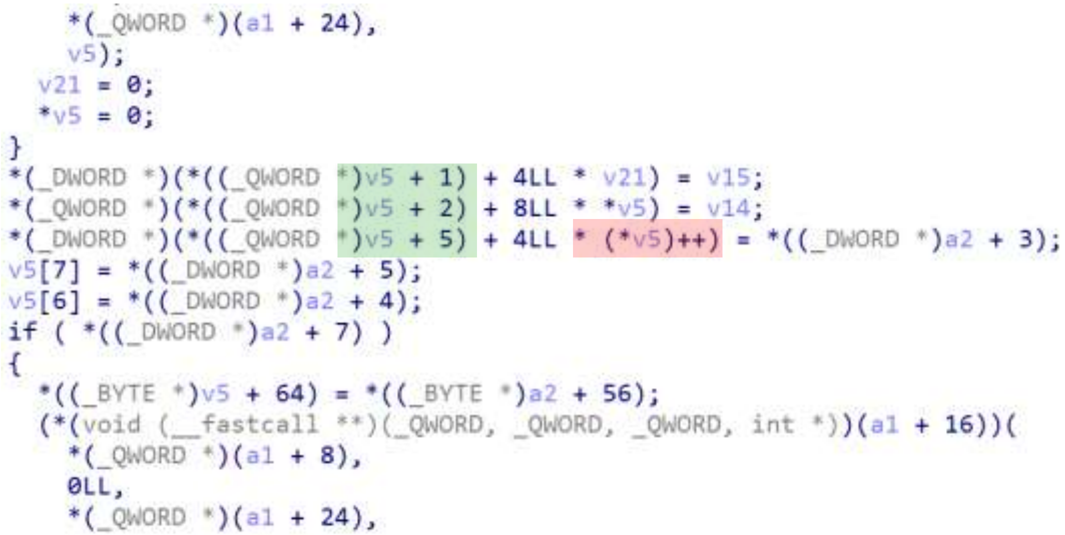
```
{ "name": "Struct_40b0",
        "ds_description": "Media information structure storing H.264 packet buffers and metadata arrays",
        "malloc_func":"rtp_dep_h264_create",
        "items": [  {"name": "count","type": "QWORD","offset": 0,"value": 0,"description":""},
                    {"name": "length_array_ptr","type": "DWORD *","offset": 8,"value": "","description":“4000"},
                    {"name": "data_pointer_array_ptr","type": "char **","offset": 16,"value": "","description":“8000"},
                    {"name": "unknown_24","type": "QWORD","offset": 24,"value": "0","description":""},
                    {"name": "unknown_32","type": "DWORD","offset": 32,"value": "0","description":""},
                    {"name": "field_36","type": "DWORD","offset": 36,"value": "*a1","description":""},
                    {"name": "rtp_seq_ptr","type": "DWORD *","offset": 40,"value": "","description":“1000"},
                    {"name": "unknown_48","type": "QWORD","offset": 48,"value": "0","description":""},
                    {"name": "packet_mode_ptr","type": "char *","offset": 56,"value": "","description":“1024"},
                    {"name": "unknown_64","type": "QWORD","offset": 64,"value": "0","description":""}]
}
```

m_info (v3)

+0x0:	count (*v3)	
+0x8:	buffer (4LL * (int)rtp_dep_obj[1])	payload size
+0x10:	buffer3(8LL * (int)rtp_dep_obj[1]	payload buf
+0x24:	media type	
+0x28:	buffer2 (4LL * (int)rtp_dep_obj[1])	seq
+0x38:	buffer4 ((int)rtp_dep_obj[1])	

Bug5 SVE-2024-1496

`rtp_dep_h263plus_put_frm`



```
      *(_QWORD *)(a1 + 24),
      v5);
    v21 = 0;
    *v5 = 0;
}
*(_DWORD *)(*((_QWORD *)v5 + 1) + 4LL * v21) = v15;
*(_QWORD *)(*((_QWORD *)v5 + 2) + 8LL * *v5) = v14;
*(_DWORD *)(*((_QWORD *)v5 + 5) + 4LL * (*v5)++) = *((_DWORD *)a2 + 3);
v5[7] = *((_DWORD *)a2 + 5);
v5[6] = *((_DWORD *)a2 + 4);
if ( *((_DWORD *)a2 + 7) )
{
  *((_BYTE *)v5 + 64) = *((_BYTE *)a2 + 56);
  (*(void (__fastcall **)(_QWORD, _QWORD, _QWORD, int *))(a1 + 16))(
    *(_QWORD *)(a1 + 8),
    0LL,
    *(_QWORD *)(a1 + 24),
```

```
{ "name": "Struct_40b0",
        "ds_description": "Media information structure storing H.264 packet buffers and metadata arrays",
        "malloc_func":"rtp_dep_h264_create",
        "items": [  {"name":  "count","type": "QWORD","offset": 0,"value": 0,"description":""},
                    {"name":  "length_array_ptr","type": "DWORD *","offset": 8,"value": "","description":“4000"},
                    {"name":  "data_pointer_array_ptr","type": "char **","offset": 16,"value": "","description":“8000"},
                    {"name":  "unknown_24","type": "QWORD","offset": 24,"value": "0","description":""},
                    {"name":  "unknown_32","type": "DWORD","offset": 32,"value": "0","description":""},
                    {"name":  "field_36","type": "DWORD","offset": 36,"value": "*a1","description":""},
                    {"name":  "rtp_seq_ptr","type": "DWORD *","offset": 40,"value": "","description":“1000"},
                    {"name":  "unknown_48","type": "QWORD","offset": 48,"value": "0","description":""},
                    {"name":  "packet_mode_ptr","type": "char *","offset": 56,"value": "","description":“1024"},
                    {"name":  "unknown_64","type": "QWORD","offset": 64,"value": "0","description":""}]
}
```

m_info (v5)

+0x0:	count (*v5)	
+0x8:	buffer (4LL * (int)rtp_dep_obj[1])	payload size
+0x10:	buffer3(8LL * (int)rtp_dep_obj[1]	payload buf
+0x24:	media type	
+0x28:	buffer2 (4LL * (int)rtp_dep_obj[1])	seq
+0x38:	buffer4 ((int)rtp_dep_obj[1])	

Benchmark

Vendor	OpenAI			Claude		Deepseek	Qwen	
Model	4o	o3	o4-mini	3	3.5	r1	max	plus
I/O (USD)	2.5/10	2/8	1.1/4.4	0.25/1.25	3/15	0.58/2.29	1.6/6.4	0.4/1.2
Bugs	✓	✓	✓	✗	✗	✓	✗	✗
Confidence	70%	85%	90%*	N/A	55%	92%*	80%*	75%*
Time	1h23m	5h29m	2h15m	N/A	2h35m	21h18m	6h47m	6h20m
Request #	247	245	223	N/A	210	258	573	238
Cost (USD)	~40	~40	~20	~15	~45	~11	~5.5	~1

* means the model may fail to find the bug in multiple runs.

Benchmark

Vendor	OpenAI			Claude		Deepseek	Qwen	
Model	4o	o3	o4-mini	3	3.5	r1	max	plus
I/O (USD)	2.5/10	2/8	1.1/4.4	0.25/1.25	3/15	0.58/2.29	1.6/6.4	0.4/1.2
Bugs	✓	✓	✓	✗	✗	✓	✗	✗
Confidence	70%	85%	90%*	N/A	55%	92%*	80%*	75%*
Time	1h23m	5h29m	2h15m	N/A	2h35m	21h18m	6h47m	6h20m
Request #	247	245	223	N/A	210	258	573	238
Cost (USD)	~40	~40	~20	~15	~45	~11	~5.5	~1

* means the model may fail to find the bug in multiple runs.

Benchmark

Vendor	OpenAI			Claude		Deepseek	Qwen	
Model	4o	o3	o4-mini	3	3.5	r1	max	plus
I/O (USD)	2.5/10	2/8	1.1/4.4	0.25/1.25	3/15	0.58/2.29	1.6/6.4	0.4/1.2
Bugs	✓	✓	✓	✗	✗	✓	✗	✗
Confidence	70%	85%	90%*	N/A	55%	92%*	80%*	75%*
Time	1h23m	5h29m	2h15m	N/A	2h35m	21h18m	6h47m	6h20m
Request #	247	245	223	N/A	210	258	573	238
Cost (USD)	~40	~40	~20	~15	~45	~11	~5.5	~1

* means the model may fail to find the bug in multiple runs.

Benchmark

Vendor	OpenAI			Claude		Deepseek	Qwen	
Model	4o	o3	o4-mini	3	3.5	r1	max	plus
I/O (USD)	2.5/10	2/8	1.1/4.4	0.25/1.25	3/15	0.58/2.29	1.6/6.4	0.4/1.2
Bugs	✓	✓	✓	✗	✗	✓	✗	✗
Confidence	70%	85%	90%*	N/A	55%	92%*	80%*	75%*
Time	1h23m	5h29m	2h15m	N/A	2h35m	21h18m	6h47m	6h20m
Request #	247	245	223	N/A	210	258	573	238
Cost (USD)	~40	~40	~20	~15	~45	~11	~5.5	~1

* means the model may fail to find the bug in multiple runs.

Takeaways

- LLM for security is trending
- LLMs don't know what to do, humans do
- Is BinWhisper producing new bugs? Yes.

Qinrun Dai (@2st_ _ _)

Yifei Xie(@iceflying2005)