补丁等级说明

A 类：问题修复，客户必须修复
B 类：性能优化，客户可选择修
C 类：功能增加，客户可选择

AW30N 广播式对讲机信号不佳时死机修复补丁

补丁等级：A
发布时间：2024 年 9 月 25 日

一、补丁说明

功能：修复广播式对讲机信号不佳时死机问题
适配版本：AW30N_release_V1.3.0 SDK
AW30N_release_V1.3.1 SDK

二、问题简介

解决状态机 ar_trans_unpack_fsm 收到部分非法数据导致的两种运行异常情况：

●**异常一 看门狗复位：** 收到数据头 0x55 后，又收到了 0x55 的数据，导致状态机会一直卡在 REV_HEADER_0xaa 状态；
●**异常二 内存访问溢出：**收到的数据头中的长度信息过大，ar_trans_unpack 中的缓存 cache 分多次取数时，cache 访问溢出的问题。

以上问题主要在**蓝牙信号不佳导致的传输数据出错的情况**下发生，也就是状态机对不可靠传输支持不足。

三、异常一的解决方法

状态机 ar_trans_unpack_fsm 收到首数据头 0x55 后，又收到了 0x55 的数据，导致状态机会一致卡在 REV_HEADER_0xaa 状态，会导致看门狗复位。

解决方法如下图，在又收到了 0x55 的数据时，将 stream_index 累加上去就可以了

```
49              /* 未收得到数据头，等待数据头中 */
50              break;
51          case REV_HEADER_0xaa:
52              /* log_info("REV_HEADER_0xaa:0x%x\n", buff[stream_index]); */
53              if (0xaa == buff[stream_index]) {
54                  ops->status = REV_CRC;
55              } else if (0x55 == buff[stream_index]) {
56                  stream_index++;
57                  break;
58              } else {
59                  ops->status = REV_HEADER_0x55;
60              }
61              stream_index++;
62              break;
63          case REV_CRC:
64              /* log_info("REV_CRC:0x%x\n", buff[stream_index]); */
NORMAL  apps/app/bsp/common/audio_rf_trans/trans_unpacket.c
```

在ar_trans_unpack_fsm函数0xaa状态机处添加这一句

四、异常二的解决方法

如下图所示解决方法，在函数 ar_trans_unpack 中将数据长度的判断修改为

```
if(ops->length > DATA_CACHE_BUF_SIZE)
```

```
176 bool ar_trans_unpack(rev_fsm_mge *ops, void *input, u16 inlen,  u16 *offset)
177 {
178     data_cache_mge *cache = &ops->cache;
179
180     u32 len, rev_status, strm_offset;
181     bool res = false;
182
183     rev_status = ar_trans_unpack_fsm(ops, input, inlen, &strm_offset);
184     *offset = strm_offset;
185
186     if (ops->length > DATA_CACHE_BUF_SIZE) {
187         /* log_error("cache_buf < data_length! %d %d\n", ops->length, DATA_CACHE_BUF_SIZE); */
188         return false;
189     }
190
191     switch (rev_status) {
192     case TUR_DATA_CORRECT:
193         res = true;
194     case TUR_DATA_RECVING:
195         len = ops->got_length - cache->offset;
196         memcpy(&cache->buf[cache->offset], &input[strm_offset - len], len);
197         cache->offset += len;
198         break;
199     case TUR_DATA_WRONG:
200         /* 数据出错，清空缓存 */
201         memset((void *)cache, 0, sizeof(data_cache_mge));
202     default:
203         break;
204     }
205
206     return res;
NORMAL  apps/app/bsp/common/audio_rf_trans/trans_unpacket.c
```

此处的ops->got_length换成ops->length

五、上一步异常二的解决方法的不足

在上一步的解决方法中虽然解决了 cache 内存溢出的问题，但是也会在 length 出错过大时，导致抛弃过多的的数据，这对音频传输效果非常不友好。

为了减少抛弃过多的数据，进一步的优化，状态机添加了数据的**回滚机制！**

六、回滚机制的实现

1、在 unpackt 流程中，添加回滚机制后，发现长度大于缓存时，回滚到上次校验的 0x55AA 的下一个字节

此时 ar_trans_unpack 中的偏移量 strm_offset 可以根据包格式头和 got_length 综合算出：

```
need_offset = ops->got_length + (ops->status - 2);

new_offset = strm_offset > need_offset ? need_offset : strm_offset;

strm_offset -= new_offset;
```

2、ar_trans_unpack_fsm 函数的 stream_index 不能再是默认从 0 开始，需要使用上一步运算出的 strm_offset。

如果需要回滚，需要算出 stream_index 在回滚时的偏移量，的 stream_index 将不在是 0 开始：

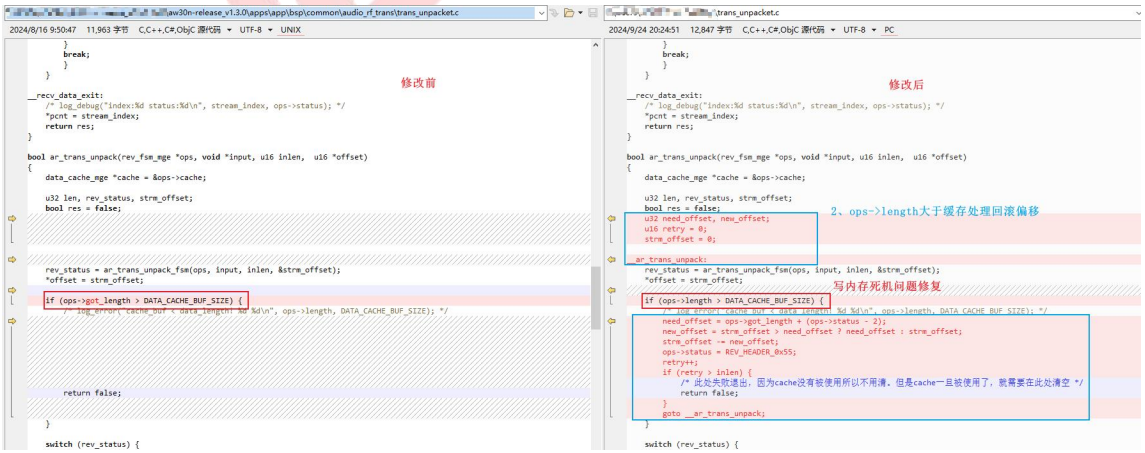
```
u32 stream_index = 0; -> u32 stream_index = *pcnt;
```

如下图，测试错误数据时回滚机制的运行情况

七、回滚机制的代码添加修改说明

可直接替换 trans_unpacket.c 对比其改动点。因为上述 strm_offset 需要回滚到上次校验的 0x55AA 的下一个字节重新校验所以补丁中有下面两处修改

（1）修改 ops->length 判断，在 ops->length 大于缓存时，运算出回滚偏移量，并再次运行状态机：



（2）ar_trans_unpack_fsm 函数的 pcnt 需要传参使用，并且状态机在 0x55 状态初始化相关变量。

```
/*----------------------------------------------------------------------*/
static u32 ar_trans_unpack_fsm(rev_fsm_mge *ops, u8 *buff, u16 len, u32 *pcnt)
{
    u32 res = TUR_HEAD_RECVING;
    u32 stream_index = 0;
    while (stream_index < len) {
        switch (ops->status) {
        case REV_HEADER_0x55:
            /* log_info("REV_HEADER_0x55:0x%x\n", buff[stream_index]); */
            if (0x55 == buff[stream_index]) {
                /* memset((u8 *)ops, 0, sizeof(rev_fsm_mge)); */
                ops->type = 0;
#if PACKET_USE_TOTAL_INDEX
                ops->packet_index = 0;
#endif
                ops->got_length = 0;
                ops->length = 0;
                ops->crc_bk = 0;
                ops->crc = 0;

                ops->status = REV_HEADER_0xaa;
            }
            stream_index++;
            /* 未收得到数据头,等待数据头中  */
            break;
        case REV_HEADER_0xaa:
            /* log_info("REV_HEADER_0xaa:0x%x\n", buff[stream_index]); */
            if (0xaa == buff[stream_index]) {
                ops->status = REV_CRC;
            } else if (0x55 == buff[stream_index]) {

                break;
            } else {
                ops->status = REV_HEADER_0x55;
            }
            stream_index++;
            break;
        case REV_CRC:
            /* log_info("REV_CRC:0x%x\n", buff[stream_index]); */
```

```
/*----------------------------------------------------------------------*/
static u32 ar_trans_unpack_fsm(rev_fsm_mge *ops, u8 *buff, u16 len, u32 *pcnt)
{
    u32 res = TUR_HEAD_RECVING;
    u32 stream_index = *pcnt;
    while (stream_index < len) {
        switch (ops->status) {
        case REV_HEADER_0x55:
            /* log_info("REV_HEADER_0x55:0x%x\n", buff[stream_index]); */

            /* memset((u8 *)ops, 0, sizeof(rev_fsm_mge)); */
            ops->type = 0;
#if PACKET_USE_TOTAL_INDEX
            ops->packet_index = 0;
#endif
            ops->got_length = 0;
            ops->length = 0;
            ops->crc_bk = 0;
            ops->crc = 0;
            if (0x55 == buff[stream_index]) {
                ops->status = REV_HEADER_0xaa;
            }
            stream_index++;
            /* 未收得到数据头,等待数据头中  */
            break;
        case REV_HEADER_0xaa:
            /* log_info("REV_HEADER_0xaa:0x%x\n", buff[stream_index]); */
            if (0xaa == buff[stream_index]) {
                ops->status = REV_CRC;
            } else if (0x55 == buff[stream_index]) {
                stream_index++;
                break;
            } else {
                ops->status = REV_HEADER_0x55;
            }
            stream_index++;
            break;
        case REV_CRC:
            /* log_info("REV_CRC:0x%x\n", buff[stream_index]); */
```

因回滚需要，stream_index使用上级函数算出的偏移量

由于上一级需要判断状态机的状态，所以在这里将状态机的初始化放到判断外面做。

珠海市杰理科技股份有限公司

2024 年 9 月 25 日