Minix Buffer Management Example in IPSec Lab

1   Description

In Inet of Minix, the packet is stored in accessors. These accessors are chained together. If the packet is large, the data saved in accessors may not be continuous. While the functions of encryption and decryption in IPSec Lab assume that the input data is continuous. Therefore we need to convert the non-continuous data used in Inet to the continuous data used in crypto utilities. The Example Code in next section gives two utility functions to do such conversion. It also provides example of the buffer management in Inet.
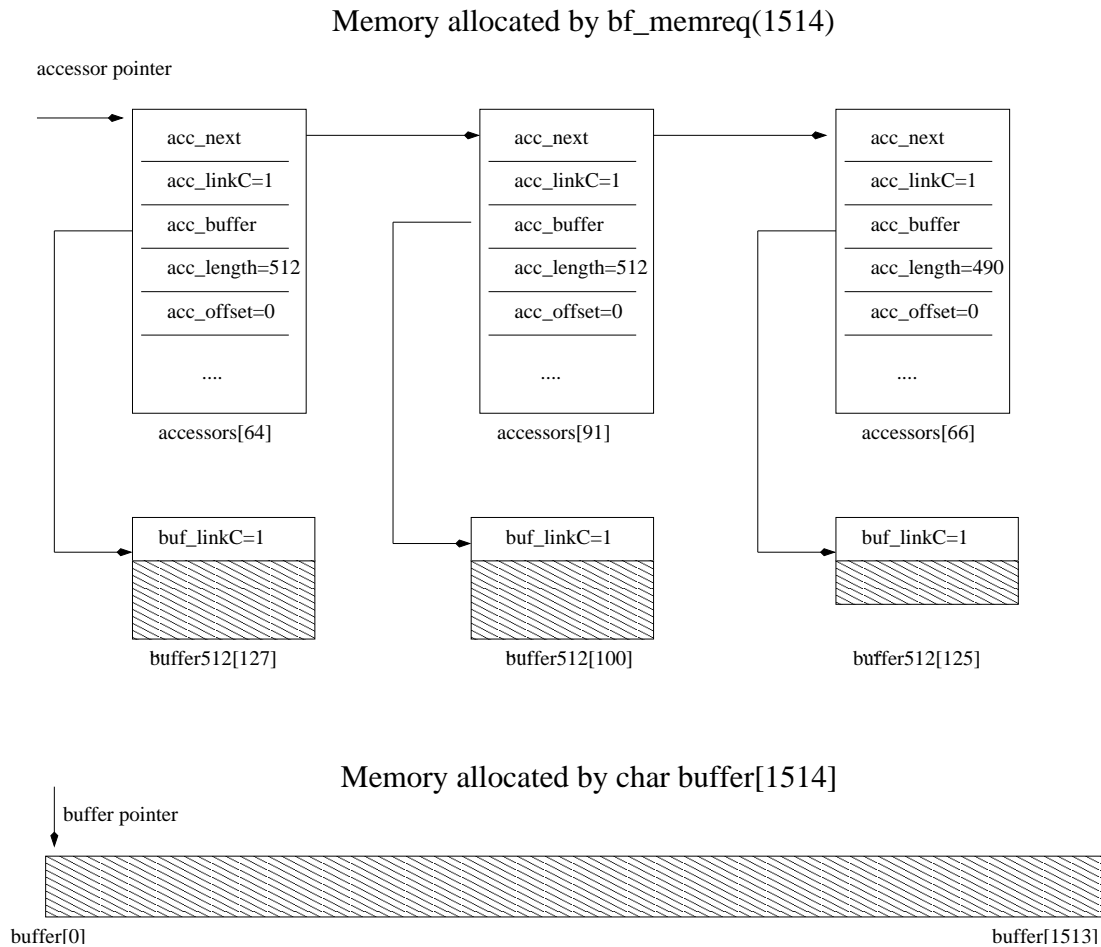
Memory allocated by bf_memreq(1514)

Memory allocated by char buffer[1514]

Figure 1: Different Memory Layout

As Figure **??**, the memory of 1514 bytes pointed by accessor pointer is not continuous, but the memory of 1514 bytes pointed by buffer pointer is continuous. The first kind of memory is used in Inet, while the later one is used in crypto utilities.

2   Example Code

`acc2buffer` and `buffer2acc` provide examples to manipulate memory in accessor link list. `acc2buffer` is used to copy data from accessor link list to array. `buffer2acc` is used to copy data from array to accessor link list.

```
/* function to copy data from accessor link list to array
 *
 *
 * pack, the pointer to the beginning of accessor link list which is input
 * buffer, the pointer to array which is output
 * size, size of array
 * return value, the length of data copied into array
 */

size_t acc2buffer(pack, buffer, size)
acc_t *pack;
u8_t *buffer;
size_t size;
{
      int length = 0;

      /* copy the data from accessor when its pointer
       * is valid, and there is room in the array
       */

      while (pack && pack->acc_length && (pack->acc_length < size))
      {

             /* memcpy the data in current accessor to buffer */
             memcpy((char*) buffer, ptr2acc_data(pack), pack->acc_length);
             /* increse the total length copied */
             length = length + pack->acc_length;
             /* update buffer pointer */
             buffer = buffer + pack->acc_length;
             /* decrease the available room in array*/
             size = size - pack->acc_length;
             /* update current accessor to next accessor in accessor link list */
             pack = pack->acc_next;
      }
      return length;
```

```
}


/* function to copy data from array to accessor link list
 *
 *
 * buffer, pointer to array which is input
 * length, length of data in input buffer
 * return value, pointer to the beginning of accessor link list which is output
 */

acc_t* buffer2acc(buffer, length)
u8_t *buffer;
size_t length;
{
        size_t count = 0;
        acc_t *ret, *pack;
        /* allocate accessor link list to hold the data with specific length */
        ret = bf_memreq(length);
        /* set the current accessor as the beginning of the accessor link list */
        pack = ret;

        /* copy the data from array when the pointer to current accessor
         * is valid, and there is any data in the array to be copied
         */
        while (pack && pack->acc_length && length)
        {

                /* count the number to be copied, which is either
                 * the rest length or the acc_length allocated
                 * for the current accessor
                 */

                if (length < pack->acc_length)
                {
                        count = length;
                }
                else
                        count = pack->acc_length;

                /* memcpy the data from buffer to current accessor */
                memcpy(ptr2acc_data(pack), (char*) buffer, count);
                /* update buffer pointer */
                buffer = buffer + pack->acc_length;
                /* decrease the length to be copied in array */
                length = length - count;
                /* update current accessor to next accessor in accessor link list */
```

```
        pack = pack->acc_next;
    }
    return ret;
}
```