9 Vuln For FreeRDP

oob write update_decompress_brush

code link
https://github.com/FreeRDP/FreeRDP/blob/master/libfreerdp/core/orders.c#L2522

update_decompress_brush could check **bytesPerPixel**, make sure that it don't read out of bound of **s** .

But it don't check **bytesPerPixel** is large than **the size of output**. if **(y * 8 + x) * bytesPerPixel) + k > the size of output,** it could lead **oob write**.

```
static BOOL update_decompress_brush(wStream* s, BYTE* output, BYTE bpp)
    {
        UINT32 index;
        UINT32 x, y, k;
        BYTE byte = 0;
        BYTE* palette;
        UINT32 bytesPerPixel;
        palette = Stream_Pointer(s) + 16;
        bytesPerPixel = ((bpp + 1) / 8);

        if (Stream_GetRemainingLength(s) < 16 + 7 * bytesPerPixel) // 64 / 4
            return FALSE;

        for (y = 7; y >= 0; y--)
        {
            for (x = 0; x < 8; x++)
            {
                if ((x % 4) == 0)
                    Stream_Read_UINT8(s, byte);

                index = ((byte >> ((3 - (x % 4)) * 2)) & 0x03);

                for (k = 0; k < bytesPerPixel; k++)
                {
                    output[((y * 8 + x) * bytesPerPixel) + k] = palette[(index * bytesPerPixel) + k];
                }
            }
```

oob read in update_recv

link
https://github.com/FreeRDP/FreeRDP/blob/master/libfreerdp/core/update.c#L771

it read 2 byte from **stream** to **updateType**, then use **updateType** as **array offset**, when **updateType > the size of UPDATE_TYPE_STRINGS** , it could oob read.

```
BOOL update_recv(rdpUpdate* update, wStream* s)
    {
        Stream_Read_UINT16(s, updateType); /* updateType (2 bytes) */
        WLog_Print(update->log, WLOG_TRACE, "%s Update Data PDU",
UPDATE_TYPE_STRINGS[updateType]);
```

rdpsnd_recv_wave2_pdu

link
https://github.com/FreeRDP/FreeRDP/blob/master/channels/rdpsnd/client/rdpsnd_main.c#L625

rdpsnd_recv_wave2_pdu read 2 byte to **wFormatNo**, then use **wFormatNo** as **array offset of format = &rdpsnd->ClientFormats[wFormatNo]**.
when **wFormatNo > size of ClientFormats**, it could oob .

```
static UINT rdpsnd_recv_wave2_pdu(rdpsndPlugin* rdpsnd, wStream* s, UINT16
BodySize)
{
    // read wFormatNo from stream
    Stream_Read_UINT16(s, wFormatNo);

    // wFormatNo is control...
    format = &rdpsnd->ClientFormats[wFormatNo];
    // access format
    if (!rdpsnd_ensure_device_is_open(rdpsnd, wFormatNo, format))
        return ERROR_INTERNAL_ER
```

2 oob in clear_decompress_bands_data

https://github.com/FreeRDP/FreeRDP/blob/master/libfreerdp/codec/clear.c#L664
https://github.com/FreeRDP/FreeRDP/blob/master/libfreerdp/codec/clear.c#L677
https://github.com/FreeRDP/FreeRDP/blob/master/libfreerdp/codec/clear.c#L760

clear_decompress_bands_data first read 2 byte from stream to vBarHeader, then it could use vBarHeader as array offset, if vBarHeader > array size, it could lead oob.

```
static BOOL clear_decompress_bands_data()
{
            // read vBarHeader
            Stream_Read_UINT16(s, vBarHeader);
```

```
            if ((vBarHeader & 0xC000) == 0x4000)
            {
                const UINT16 vBarIndex = (vBarHeader & 0x3FFF);
                // vBarIndex is control!
                vBarShortEntry = &(clear->ShortVBarStorage[vBarIndex]);
                if (Stream_GetRemainingLength(s) < 1)
                // vuln !
                vBarShortPixelCount = vBarShortEntry->count;
................................
................................
            else if ((vBarHeader & 0x8000) == 0x8000) /* VBAR_CACHE_HIT */
            {
                const UINT16 vBarIndex = (vBarHeader & 0x7FFF);
                vBarEntry = &(clear->VBarStorage[vBarIndex]
```

int overflow in zgfx_decompress_segment

https://github.com/FreeRDP/FreeRDP/blob/master/libfreerdp/codec/zgfx.c#L236
https://github.com/FreeRDP/FreeRDP/blob/master/libfreerdp/codec/zgfx.c#L262

segmentSize is read from stream, so it is **controled**.
when **segmentSize=0**, then **cbSegment = 0xffffffff**, it could lead **oob** later.

```
static BOOL zgfx_decompress_segment(ZGFX_CONTEXT* zgfx, wStream* stream, size_t segmentSize)
{

    cbSegment = segmentSize - 1;
    if ((Stream_GetRemainingLength(stream) < segmentSize) || (segmentSize < 1) ||
        (segmentSize > UINT32_MAX))
        return FALSE;
    zgfx->pbInputCurrent = pbSegment;
    zgfx->pbInputEnd = &pbSegment[cbSegment - 1];
    zgfx->cBitsRemaining = 8 * (cbSegment - 1) - *zgfx->pbInputEnd;
```

oob and int overflow in clear_decompress_subcode_rlex

oob

https://github.com/FreeRDP/FreeRDP/blob/master/libfreerdp/codec/clear.c#L161
it first read 1 byte to **paletteCount**, then could read **3*paletteCount** byte from **stream**, without check **stream's length**, if **paletteCount > stream's length, it could oob read**.

```
static BOOL clear_decompress_subcode_rlex
    {
```

```
        // bitmapDataByteCount control by net data
        if (Stream_GetRemainingLength(s) < bitmapDataByteCount)
        {
            return FALSE;
        }

        // read
        Stream_Read_UINT8(s, paletteCount);
        bitmapDataOffset = 1 + (paletteCount * 3);

        if ((paletteCount > 127) || (paletteCount < 1))
        {
            WLog_ERR(TAG, "paletteCount %" PRIu8 "", paletteCount);
            return FALSE;
        }

        // oob
        for (i = 0; i < paletteCount; i++)
        {
            BYTE r, g, b;
            Stream_Read_UINT8(s, b);
            Stream_Read_UINT8(s, g);
            Stream_Read_UINT8(s, r);
            palette[i] = FreeRDPGetColor(DstFormat, r, g, b, 0xFF);
        }
```

int overflow

then it could read **runLengthFactor** from **stream**, **runLengthFactor's maxvalue is 0xffffffff**.
https://github.com/FreeRDP/FreeRDP/blob/master/libfreerdp/codec/clear.c#L210

then it could check **runLengthFactor with pixelCount**
https://github.com/FreeRDP/FreeRDP/blob/master/libfreerdp/codec/clear.c#L239

```
// runLengthFactor maxvalue is 0xffffffff
  if ((pixelIndex + runLengthFactor) > pixelCount)
  {
    // report error
  }
  ......................
  for (i = 0; i < runLengthFactor; i++)// vuln..
  {
    // write data
  }
```

if **pixelIndex + runLengthFactor** int overflow, it could bypass the check, then oob write later.

oob in rfx_process_message_tileset

https://github.com/FreeRDP/FreeRDP/blob/master/libfreerdp/codec/rfx.c#L910

it could read 4 byte to **blockLen**, then check **blockLen-6**.
if **blockLen = 6**, it could **bypass the all check**, then it could read some data from stream **without check stream's length**.

```
        // it first read 32bit to blockLen
        Stream_Read_UINT32(s, blockLen);
        if (Stream_GetRemainingLength(s) < blockLen - 6)
            {
                rc = FALSE;
                break;
            }

            pos = Stream_GetPosition(s) - 6 + blockLen;

            //
            if (blockType != CBT_TILE)
            {
                rc = FALSE;
                break;
            }

            // oob read
            Stream_Read_UINT8(s, tile->quantIdxY);  /* quantIdxY (1 byte) */
            Stream_Read_UINT8(s, tile->quantIdxCb); /* quantIdxCb (1 byte) */
            Stream_Read_UINT8(s, tile->quantIdxCr); /* quantIdxCr (1 byte) */
            Stream_Read_UINT16(s, tile->xIdx);      /* xIdx (2 bytes) */
            Stream_Read_UINT16(s, tile->yIdx);      /* yIdx (2 bytes) */
            Stream_Read_UINT16(s, tile->YLen);      /* YLen (2 bytes) */
            Stream_Read_UINT16(s, tile->CbLen);     /* CbLen (2 bytes) */
            Stream_Read_UINT16(s, tile->CrLen);     /* CrLen (2 bytes) */
            Stream_GetPointer(s, tile->YData);
            Stream_Seek(s, tile->YLen);
            Stream_GetPointer(s, tile->CbData);
            Stream_Seek(s, tile->CbLen);
            Stream_GetPointer(s, tile->CrData);
            Stream_Seek(s, tile->CrLen);
            tile->x = tile->xIdx * 64;
```