

```

0:005> r
rax=00000214d6cf7000 rbx=0000000000000000 rcx=00000214d6cf6424
rdx=00007ffbfce03f0 rsi=00007ffbfd03632b rdi=0000005b73efed90
rip=00007ffbfcddeb6c9 rsp=0000005b73efecd0 rbp=0000000000000000
r8=0000000000000000 r9=0000000000000001 r10=fefefefefefeff
r11=8080808080808080 r12=0000000000000000 r13=0000000000000000
r14=0000000000000000 r15=0000000000000000
iopl=0         nv up ei pl nz na po nc
cs=0033  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00010206
freerdp2!gdi_multi_opaque_rect+0xf9:
00007ffb`fcdeb6c9 8b00          mov     eax,dword ptr [rax] ds:00000214`d6cf7000=????????

```

This is also an out-of-bounds access to the source code:

```

static BOOL gdi_multi_opaque_rect(rdpContext* context,
                                  const MULTI_OPAQUE_RECT_ORDER* multi_opaque_rect)
{
    UINT32 i;
    GDI_RECT rect;
    HGDI_BRUSH hBrush;
    UINT32 brush_color;
    rdpGdi* gdi = context->gdi;
    BOOL ret = TRUE;

    if (!gdi_decode_color(gdi, multi_opaque_rect->color, &brush_color, NULL))
        return FALSE;

    hBrush = gdi_CreateSolidBrush(brush_color);

    if (!hBrush)
        return FALSE;

    for (i = 0; i < multi_opaque_rect->numRectangles; i++)
    {
        const DELTA_RECT* rectangle = &multi_opaque_rect->rectangles[i];
        INT32 x = rectangle->left; ❌
        INT32 y = rectangle->top;
        INT32 w = rectangle->width;
        INT32 h = rectangle->height;
        gdi_ClipCoords(gdi->drawing->hdc, &x, &y, &w, &h, NULL, NULL);
        gdi_CRgnToRect(x, y, w, h, &rect);
        ret = gdi_FillRect(gdi->drawing->hdc, &rect, hBrush);
    }
}

```

Border access at Rectangle->left.

View MULTI_OPAQUE_RECT_ORDER type structure:

```

struct _MULTI_OPAQUE_RECT_ORDER
{
    INT32 nLeftRect;
    INT32 nTopRect;
    INT32 nWidth;
    INT32 nHeight;
    UINT32 color;
    UINT32 numRectangles;
    UINT32 cbData;
    DELTA_RECT rectangles[45];
};

typedef struct _MULTI_OPAQUE_RECT_ORDER MULTI_OPAQUE_RECT_ORDER;

```

According to the code above, numRectangles represents the length of the rectangles array below. When debugging, numRectangles was set to 239, or 0xEF, apparently exceeding the length of the array.

Find the position that fills the structure:

update_read_multi_opaque_rect_order function:

```
static BOOL update_read_multi_opaque_rect_order(wStream* s,
        const ORDER_INFO* orderInfo,
        MULTI_OPAQUE_RECT_ORDER* multi_opaque_rect)
{
    BYTE byte;
    ORDER_FIELD_COORD(1, multi_opaque_rect->nLeftRect);
    ORDER_FIELD_COORD(2, multi_opaque_rect->nTopRect);
    ORDER_FIELD_COORD(3, multi_opaque_rect->nWidth);
    ORDER_FIELD_COORD(4, multi_opaque_rect->nHeight);

    if (orderInfo->fieldFlags & ORDER_FIELD_05) ██████████
    if (orderInfo->fieldFlags & ORDER_FIELD_06) { ... }
    if (orderInfo->fieldFlags & ORDER_FIELD_07) { ... }

    ORDER_FIELD_BYTE(8, multi_opaque_rect->numRectangles);

    if (orderInfo->fieldFlags & ORDER_FIELD_09) { ... }

    return TRUE;
}
```

1 byte is read here as numRectangles. And from here to the abnormal location, there is no check.

This leads to cross-border access.