VLC : Integer overflow in vnc module <= 3.0.18 CVE-2022-41325

Security advisory

2022-11-30

0xMitsurugi

Vulnerability description

Presentation of VLC

VLC media player (previously the VideoLAN Client and commonly known as simply VLC) is a free and open-source, portable, cross-platform media player software and streaming media server developed by the VideoLAN project.

VLC can display a vnc video stream by using its URI:

```
vlc vnc://ip_address_of_server:port/
```

The issue

The vulnerability resides in VNC module.

If an attacker has control over a VNC server, it can trick VLC into allocating a memory buffer shorter than expected. The attacker then have a powerful relative "write-what-where" primitive. He can crash VLC, or execute arbitrary code under conditions.

NOTE: Although vnc support is made through a third-party library (libvncClient), the affected code is in VLC.

Affected versions

Version 3.0.17.4 and earlier are affected.

CVE

CVE-2022-41325 assigned by MITRE,
CVSS: 7.8

Timeline

Date	Action
2022-09-19	Ticket opened in VLC tracker
2022-09-19	Developers proposed a patch fixing the vulnerability
2022-09-23	CVE-2022-41325 assigned by *MITRE*
2022-11-30	Publication of 3.0.18

Technical description and POC

Availability of the bug

The ./configure script tries to compile support for VNC by default:

```
$ ./configure --help | grep vnc
  --enable-vnc            (VNC/rfb client support) [default=auto]
$
```

you can check support of vnc in VLC by typing:

```
$ vlc --list | grep vnc
VLC media player 3.0.17.4 Vetinari (revision 3.0.13-8-g41878ff4f2)
  vnc                    Accès client VNC
$
```

The bug

The bug resides in the function *mallocFrameBufferHandler* in *modules/access/vnc.c*:

```
static rfbBool mallocFrameBufferHandler( rfbClient* p_client ) {
    (...)
    /* Set up framebuffer */
    p_sys->i_framebuffersize = i_width * i_height * i_depth / 8;     [1]

    /* Reuse unsent block */
    if ( p_sys->p_block )
        p_sys->p_block = block_Realloc( p_sys->p_block, 0, p_sys->i_framebuffersize );
    else
        p_sys->p_block = block_Alloc( p_sys->i_framebuffersize );    [2]
    (...)
```

Variables are typed:

- p_sys->i_framebuffer is an int
- i_width is an int. VNC protocol force it by design to be an uint16
- i_height is an int. VNC protocol force it by design to be an uint16
- i_depth can value 32 at max

The calculation in [1] "i_width * i_height * i_depth / 8" can overflow the int size of p_sys->i_framebuffersize, and result value can be truncated.

Proof of concept

i_width, i_height and i_depth are given by the VNC server, and are used without any verification. If we create a vnc screen sized as:

- width: 0x4004

- height: 0xfff1
- depth: 32 bpp

we have:

0x4004*0xfff1*32/8 = 0x10000ff10 which will be truncated to 0xff10

The block_alloc() will alloc a 0xff10 bytes buffer which is smaller than expected to store data sent by the VNC server.

NOTE: the block_alloc function checks that the size to allocate is smaller than 2^27. This is why we choose those specific values for height and width. Any other values can work, as long as the int truncation produces a number smaller than 2^27.

The VNC server can update portions of the screen, from the entire buffer to a small rectangle, or even each pixel by sending packets to VLC. It's trivial to write any byte outside the buffer allocated in memory of VLC, ranging from start of buffer allocated to buffer + 2^32 bytes.

For example, by sending an RFB pixel coded on 4 bytes such as 0x41414141 at coordinates X=0x4000, Y=0x3, a write will be performed in memory past the end of the buffer:

0x4000*3*4 = 0x30000

which is after the boundary of the 0xff10 buffer allocated. This can result in unexpected behavior.

In theory, command execution is possible by rewriting chosen section of data, However, with a stack randomized, a heap randomized and a binary compiled with PIE exploitation is very unlikely.

Patch

VLC team fixed the vulnerability with the commit:

https://code.videolan.org/videolan/vlc/-/commit/4fcace61801f418786c42487c6b06b693ee87666

The patch adds a check to verify that the buffer has not overflowed:

```
static rfbBool mallocFrameBufferHandler( rfbClient* p_client )
(...)
    assert(!(p_client->width & ~0xffff)); // fits in 16 bits
    uint16_t i_width = p_client->width;
    assert(!(p_client->height & ~0xffff)); // fits in 16 bits
    uint16_t i_height = p_client->height;
    uint8_t i_bits_per_pixel = p_client->format.bitsPerPixel;
    assert((i_bits_per_pixel & 0x7) == 0); // multiple of 8
(...)
    /* Set up framebuffer */
    if (mul_overflow(i_width, i_height * (i_bits_per_pixel / 8), &p_sys-
>i_framebuffersize)) {
        msg_Err(p_demux, "VNC framebuffersize overflow");
        return FALSE;
    }
(...)
```

the mul_overflow() function explicitly checks that the multiplication won't overflow.