Talos Vulnerability Report

TALOS-2020-0989

Accusoft ImageGear BMP bmp_parsing buffer size computation code execution vulnerability

FEBRUARY 10, 2020

CVE NUMBER

CVE-2020-6065

Summary

An exploitable out-of-bounds write vulnerability exists in the `bmp_parsing` function of the igcore19d.dll library of Accusoft ImageGear, version 19.5.0. A specially crafted BMP file can cause an out-of-bounds write, resulting in a remote code execution. An attacker needs to provide a malformed file to the victim to trigger the vulnerability.

Tested Versions

Accusoft ImageGear 19.5.0

Product URLs

https://www.accusoft.com/products/imagegear/overview/

CVSSv3 Score

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

CWE

CWE-190: Integer Overflow or Wraparound

Details

The ImageGear library is a document imaging developer toolkit providing all kinds of functionality related to image conversion, creation, editing, annotation, etc. It supports more than 100 formats, including many image formats, DICOM, PDF, Microsoft Office and others.

There is a vulnerability in the `uncompress_scan_line` function, due to an integer overflow. A specially crafted BMP file can lead to an out-of-bounds write, which can result in remote code execution.

Trying to load a malformed BMP file via `IG_load_file` function we end up in the following situation:

```
eax=00000040 ebx=00000002 ecx=0e0d4f90 edx=00000000 esi=0e832ffe edi=05996ff1
eip=568edbb3 esp=006fee24 ebp=006fee7c iopl=0         nv up ei ng nz na po cy
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00010283
igCore19d!IG_mpi_page_set+0x72823:
568edbb3 884602          mov     byte ptr [esi+2],al        ds:002b:0e833000=??
0:000> kb
 # ChildEBP RetAddr  Args to Child
WARNING: Stack unwind information not available. Following frames may be wrong.
00 006fee7c 568efdd9 006ff40c 1000001b 0e05afe8 igCore19d!IG_mpi_page_set+0x72823
01 006feebc 568ece25 006ff40c 1000001b 0e05afe8 igCore19d!IG_mpi_page_set+0x74a49
02 006ff384 568507c9 006ff40c 0e05afe8 00000001 igCore19d!IG_mpi_page_set+0x71a95
03 006ff3bc 5688fb97 00000000 0e05afe8 006ff40c igCore19d!IG_image_savelist_get+0xb29
04 006ff638 5688f4f9 00000000 0984bfa8 00000001 igCore19d!IG_mpi_page_set+0x14807
05 006ff658 56826007 00000000 0984bfa8 00000001 igCore19d!IG_mpi_page_set+0x14169
06 006ff678 00d859ac 0984bfa8 006ff764 006ff788 igCore19d!IG_load_file+0x47
07 006ff778 00d861a7 0984bfa8 006ff8ac 00000021 simple_exe_141+0x159ac
08 006ff944 00d86cbe 00000005 097f8f50 096e1f40 simple_exe_141+0x161a7
09 006ff958 00d86b27 d4d1a41c 00d815e1 00d815e1 simple_exe_141+0x16cbe
0a 006ff9b4 00d869bd 006ff9c4 00d86d38 006ff9d4 simple_exe_141+0x16b27
0b 006ff9bc 00d86d38 006ff9d4 76cd6359 0052f000 simple_exe_141+0x169bd
0c 006ff9c4 76cd6359 0052f000 76cd6340 006ffa30 simple_exe_141+0x16d38
0d 006ff9d4 77577b74 0052f000 a1cf5680 00000000 KERNEL32!BaseThreadInitThunk+0x19
0e 006ffa30 77577b44 ffffffff 77598ef2 00000000 ntdll!__RtlUserThreadStart+0x2f
0f 006ffa40 00000000 00d815e1 0052f000 00000000 ntdll!_RtlUserThreadStart+0x1b
```

As we can see, an out-of-bounds operation occurred.

The pseudo-code of this vulnerable function looks like this:

```
int __stdcall bmp_parsing(int arg0, void *a1, _DWORD *a3, bitmap_object *bmp_object)
{
  bitmap_object *_bmp_object; // edi
  int v5; // esi
  int width; // eax
  int total_width; // ebx
  int v8; // esi
  int result; // eax
  _BYTE *buffer_allocated; // esi
  int v11; // ebx
  _BYTE *v12; // edi
  int v13; // eax
  int v14; // eax
  unsigned int v15; // eax
  unsigned int v16; // [esp-4h] [ebp-58h]
  int v17[13]; // [esp+Ch] [ebp-48h]
  int v18; // [esp+40h] [ebp-14h]
  int v19; // [esp+44h] [ebp-10h]
  int v20; // [esp+48h] [ebp-Ch]
  void *_buffer_allocated; // [esp+4Ch] [ebp-8h]
  unsigned int v22; // [esp+50h] [ebp-4h]

  _bmp_object = bmp_object;
  v5 = 0;
  width = getBiWidth(bmp_object);
  total_width = 4 * width;                             [3]
  v19 = 4 * width;
  sub_5681AF60(arg0, (int)a1, (int)v17, 20 * width, 1);
  _buffer_allocated = AF_memm_alloc((int)a1, total_width, (int)"..\\..\\..\\..\\Common\\Formats\\bmpread.c", 1818);   [2]
  if ( _buffer_allocated )
  {
    v22 = 0;
    if ( getSizeY_0((IGDIBOject *)bmp_object) )
    {
      v20 = -1;
      do
      {
        v18 = sub_5681B280(v17, total_width);
        if ( !v18 )
          break;
        buffer_allocated = _buffer_allocated;
        v11 = 0;
        if ( getBiWidth(_bmp_object) > 0 )
        {
          v12 = (_BYTE *)(v18 + 1);
          do
          {
            *buffer_allocated = v12[1];
            buffer_allocated[1] = *v12;
            buffer_allocated[2] = *(v12 - 1);          [1]
            buffer_allocated += 3;
            if ( sub_567D92B0(bmp_object) == 501 )
              *buffer_allocated++ = v12[2];
            ++v11;
            v12 += 4;
          }
          while ( v11 < getBiWidth(bmp_object) );
          _bmp_object = bmp_object;
        }
...
```

In this code we can observe a function `bmp_parsing`, whose objective is to decompress the bmp data, is crashing while filling the buffer `buffer_allocated` in [1].

At [3], the `total_width` is calculated by multiplying the `width` by 4. However, there is no integer overflow check on this operation, which could lead to a wraparound. This would further lead to the allocation of the buffer at [2] using an improper size, making the loop at [1] to write out of bounds.

Crash Information

```
0:000> !analyze -v
*******************************************************************************
*                                                                             *
*                        Exception Analysis                                   *
*                                                                             *
*******************************************************************************


KEY_VALUES_STRING: 1

        Key  : AV.Fault
        Value: Write

        Key  : Analysis.CPU.Sec
        Value: 1

        Key  : Analysis.DebugAnalysisProvider.CPP
        Value: Create: 8007007e on DESKTOP-PJK7PVH

        Key  : Analysis.DebugData
        Value: CreateObject

        Key  : Analysis.DebugModel
        Value: CreateObject

        Key  : Analysis.Elapsed.Sec
        Value: 9

        Key  : Analysis.Memory.CommitPeak.Mb
        Value: 73

        Key  : Analysis.System
        Value: CreateObject

        Key  : Timeline.OS.Boot.DeltaSec
        Value: 320918

        Key  : Timeline.Process.Start.DeltaSec
        Value: 1022


ADDITIONAL_XML: 1

APPLICATION_VERIFIER_LOADED: 1

EXCEPTION_RECORD:  (.exr -1)
ExceptionAddress: 568edbb3 (igCore19d!IG_mpi_page_set+0x00072823)
   ExceptionCode: c0000005 (Access violation)
  ExceptionFlags: 00000000
NumberParameters: 2
   Parameter[0]: 00000001
   Parameter[1]: 0e833000
Attempt to write to address 0e833000

FAULTING_THREAD:  00001c80

PROCESS_NAME:  simple.exe_141.exe

WRITE_ADDRESS:  0e833000

ERROR_CODE: (NTSTATUS) 0xc0000005 - The instruction at 0x%p referenced memory at 0x%p. The memory could not be %s.

EXCEPTION_CODE_STR:  c0000005

EXCEPTION_PARAMETER1:  00000001

EXCEPTION_PARAMETER2:  0e833000

STACK_TEXT:
WARNING: Stack unwind information not available. Following frames may be wrong.
006fee7c 568efdd9 006ff40c 1000001b 0e05afe8 igCore19d!IG_mpi_page_set+0x72823
006feebc 568ece25 006ff40c 1000001b 0e05afe8 igCore19d!IG_mpi_page_set+0x74a49
006ff384 568507c9 006ff40c 0e05afe8 00000001 igCore19d!IG_mpi_page_set+0x71a95
006ff3bc 5688fb97 00000000 0e05afe8 006ff40c igCore19d!IG_image_savelist_get+0xb29
006ff638 5688f4f9 00000000 0984bfa8 00000001 igCore19d!IG_mpi_page_set+0x14807
006ff658 56826007 00000000 0984bfa8 00000001 igCore19d!IG_mpi_page_set+0x14169
006ff678 00d859ac 0984bfa8 006ff764 006ff788 igCore19d!IG_load_file+0x47
006ff778 00d861a7 0984bfa8 006ff8ac 00000021 simple_exe_141+0x159ac
006ff944 00d86cbe 00000005 097f8f50 096e1f40 simple_exe_141+0x161a7
006ff958 00d86b27 d4d1a41c 00d815e1 00d815e1 simple_exe_141+0x16cbe
006ff9b4 00d869bd 006ff9c4 00d86d38 006ff9d4 simple_exe_141+0x16b27
006ff9bc 00d86d38 006ff9d4 76cd6359 0052f000 simple_exe_141+0x169bd
006ff9c4 76cd6359 0052f000 76cd6340 006ffa30 simple_exe_141+0x16d38
006ff9d4 77577b74 0052f000 a1cf5680 00000000 KERNEL32!BaseThreadInitThunk+0x19
006ffa30 77577b44 ffffffff 77598ef2 00000000 ntdll!__RtlUserThreadStart+0x2f
006ffa40 00000000 00d815e1 0052f000 00000000 ntdll!_RtlUserThreadStart+0x1b


STACK_COMMAND:  ~0s ; .cxr ; kb

SYMBOL_NAME:  igCore19d!IG_mpi_page_set+72823

MODULE_NAME: igCore19d

IMAGE_NAME:  igCore19d.dll

FAILURE_BUCKET_ID:  INVALID_POINTER_WRITE_AVRF_c0000005_igCore19d.dll!IG_mpi_page_set

OS_VERSION:  10.0.18362.239

BUILDLAB_STR:  19h1_release_svc_prod1

OSPLATFORM_TYPE:  x86

OSNAME:  Windows 10

FAILURE_ID_HASH:  {39ff52ad-9054-81fd-3e4d-ef5d82e4b2c1}

Followup:     MachineOwner
---------
```

Timeline

2020-01-27 - Vendor Disclosure
2020-02-10 - Public Release

CREDIT

Discovered by Emmanuel Tacheau of Cisco Talos.

VULNERABILITY REPORTS

PREVIOUS REPORT
TALOS-2020-0987

NEXT REPORT
TALOS-2020-0990