Talos Vulnerability Report

TALOS-2021-1289

Accusoft ImageGear JPG Handle_JPEG420 out-of-bounds write vulnerability

JUNE 1, 2021

CVE NUMBER

CVE-2021-21824

Summary

An out-of-bounds write vulnerability exists in the JPG Handle_JPEG420 functionality of Accusoft ImageGear 19.9. A specially crafted malformed file can lead to memory corruption. An attacker can provide a malicious file to trigger this vulnerability.

Tested Versions

Accusoft ImageGear 19.9

Product URLs

https://www.accusoft.com/products/imagegear-collection/

CVSSv3 Score

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

CWE

CWE-131 - Incorrect Calculation of Buffer Size

Details

The ImageGear library is a document-imaging developer toolkit that offers image conversion, creation, editing, annotation and more. It supports more than 100 formats such as DICOM, PDF, Microsoft Office and others.

A specially crafted JPG file can lead to an out-of-bounds write in the `handle_JPEG420` function, due to a buffer overflow caused by a missing size check for a buffer memory.

Trying to load a malformed JPG file, we end up in the following situation:

```
This exception may be expected and handled.
eax=08ee4c51 ebx=05d08c51 ecx=00000080 edx=0db17001 esi=10ab4fd1 edi=00000000
eip=793c5eba esp=0019f5f8 ebp=0019f640 iopl=0         nv up ei pl nz na po nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00010202
igCore19d!IG_mpi_page_set+0xca16a:
793c5eba 884aff          mov     byte ptr [edx-1],cl        ds:002b:0db17000=??
```

This write access violation is happening in the function `handle_JPEG420`, corresponding to the following pseudo-code, in LINE152:

```
LINE1    dword __cdecl
LINE2    handle_JPEG420(int param_1,int width,int height,int min_height,char *param_5,
LINE3                   undefined *raster_buffer)
LINE4    {
        [...]
LINE29     iVar2 = *(int *)(param_5 + 0x7c);
LINE30     pcVar5 = (char *)(min_height - 1U & 0x8000000f);
LINE31     if ((int)pcVar5 < 0) {
LINE32       pcVar5 = (char *)(((uint)(pcVar5 + -1) | 0xfffffff0) + 1);
LINE33     }
LINE34     pcVar6 = pcVar5;
LINE35     if (min_height != 0) {
LINE36       if (pcVar5 == NULL) {
LINE37         pcVar12 = *(char **)(param_5 + 0x54);
LINE38         local_8 = pcVar12 + iVar2;
LINE39         pcVar6 = *(char **)(param_5 + 0xa4);
LINE40         local_c = pcVar6 + iVar2;
LINE41         local_10 = (char *)(*(int *)(param_5 + 0x58) + iVar2 * 7);
LINE42         iVar7 = *(int *)(param_5 + 4);
LINE43         param_5 = (char *)(*(int *)(param_5 + 0xa8) + iVar2 * 7);
LINE44       }
LINE45       else {
LINE46         if (pcVar5 == (char *)0xf) {
LINE47           local_8 = *(char **)(param_5 + 0x54);
LINE48           local_c = *(char **)(param_5 + 0xa4);
LINE49           pcVar12 = (char *)(*(int *)(param_5 + 0x58) + iVar2 * 7);
LINE50           local_10 = (char *)(*(int *)(param_5 + 0x58) + *(int *)(param_5 + 0x7c) * 6);
LINE51           pcVar6 = (char *)(iVar2 * 7 + *(int *)(param_5 + 0xa8));
LINE52           iVar7 = (*(int *)(param_5 + 0x2c) * 0x10 - *(int *)(param_5 + 0x2c)) + *(int *)(param_5 + 8)
LINE53           ;
LINE54           param_5 = (char *)(*(int *)(param_5 + 0x7c) * 6 + *(int *)(param_5 + 0xa8));
LINE55         }
LINE56         else {
LINE57           iVar7 = ((int)pcVar5 / 2) * iVar2;
LINE58           pcVar12 = (char *)(*(int *)(param_5 + 0x54) + iVar7);
LINE59           pcVar6 = (char *)(*(int *)(param_5 + 0xa4) + iVar7);
LINE60           local_8 = pcVar12 + iVar2;
LINE61           local_c = pcVar6 + iVar2;
LINE62           local_10 = pcVar12 + -iVar2;
LINE63           iVar7 = (int)pcVar5 * *(int *)(param_5 + 0x2c) + *(int *)(param_5 + 4);
LINE64           param_5 = pcVar6 + -iVar2;
LINE65         }
LINE66       }
LINE67       index = 0;
LINE68       if (0 < width) {
LINE69         uVar8 = (uint)pcVar5 & 0x80000001;
LINE70         if ((int)uVar8 < 0) {
LINE71           uVar8 = (uVar8 - 1 | 0xfffffffe) + 1;
LINE72         }
LINE73         iVar2 = -(int)pcVar6;
LINE74         iVar3 = -(int)pcVar6;
LINE75         iVar4 = -(int)pcVar6;
LINE76         do {
LINE77           iVar14 = -1;
LINE78           if (index == 0) {
LINE79             iVar14 = 0;
LINE80           }
LINE81           uVar11 = (uint)(index != width - 1U);
LINE82           local_18 = local_10 + iVar4 + (int)pcVar6;
LINE83           local_14 = param_5;
LINE84           if (min_height == 1) {
LINE85             local_18 = pcVar12;
LINE86             local_14 = pcVar6;
LINE87           }
LINE88           local_20 = pcVar6 + (int)(local_8 + iVar2);
LINE89           local_1c = pcVar6 + (int)(local_c + iVar3);
LINE90           if (min_height == height + -1) {
LINE91             local_20 = pcVar12;
LINE92             local_1c = pcVar6;
LINE93           }
LINE94           uVar13 = index - 1;
LINE95           if (index != width - 1U) {
LINE96             uVar13 = index;
LINE97           }
LINE98           uVar9 = (uint)(byte)(*(char *)(index + iVar7) + 0x80);
LINE99           if (uVar8 == 0) {
LINE100            uVar13 = uVar13 & 0x80000001;
LINE101            if ((int)uVar13 < 0) {
LINE102              uVar13 = (uVar13 - 1 | 0xfffffffe) + 1;
LINE103            }
LINE104            if (uVar13 == 0) {
LINE105              local_28 = (int)local_14[iVar14] +
LINE106                         ((int)pcVar6[iVar14] + *pcVar6 * 3 + (int)*local_14) * 3 + 8 >> 4;
LINE107              iVar10 = (int)local_18[iVar14];
LINE108              iVar14 = ((int)pcVar12[iVar14] + *pcVar12 * 3 + (int)*local_18) * 3 + 8;
LINE109            }
LINE110            else {
LINE111              iVar10 = local_14[uVar11] + 8 + ((int)pcVar6[uVar11] + *pcVar6 * 3 + (int)*local_14) * 3
LINE112              ;
LINE113              iVar14 = (int)pcVar12[uVar11] + *pcVar12 * 3 + (int)*local_18;
LINE114  LAB_10135e43:
LINE115              local_28 = iVar10 >> 4 & 0xff;
LINE116              iVar10 = local_18[uVar11] + 8;
LINE117              iVar14 = iVar14 * 3;
LINE118            }
LINE119          }
LINE120          else {
LINE121            uVar13 = uVar13 & 0x80000001;
LINE122            if ((int)uVar13 < 0) {
LINE123              uVar13 = (uVar13 - 1 | 0xfffffffe) + 1;
LINE124            }
LINE125            if (uVar13 != 0) {
LINE126              iVar10 = local_1c[uVar11] + 8 + ((int)pcVar6[uVar11] + *pcVar6 * 3 + (int)*local_1c) * 3
LINE127              ;
LINE128              iVar14 = (int)pcVar12[uVar11] + *pcVar12 * 3 + (int)*local_20;
LINE129              local_18 = local_20;
LINE130              goto LAB_10135e43;
LINE131            }
LINE132            local_28 = (int)local_1c[iVar14] +
LINE133                       ((int)pcVar6[iVar14] + *pcVar6 * 3 + (int)*local_1c) * 3 + 8 >> 4 & 0xff;
LINE134            iVar10 = (int)local_20[iVar14];
LINE135            iVar14 = ((int)pcVar12[iVar14] + *pcVar12 * 3 + (int)*local_20) * 3 + 8;
LINE136          }
LINE137          if (uVar13 == 1) {
LINE138            param_5 = param_5 + 1;
LINE139            pcVar6 = pcVar6 + 1;
LINE140            pcVar12 = pcVar12 + 1;
```

```
LINE141          }
LINE142          uVar11 = iVar14 + iVar10 >> 4 & 0xff;
LINE143          sVar1 = *(short *)(&DAT_1026fb80 + uVar11 * 2);
LINE144          iVar14 = *(int *)(&DAT_10270180 + (local_28 & 0xff) * 4);
LINE145          iVar10 = *(int *)(&DAT_1026fd80 + uVar11 * 4);
LINE146                           /*  */
LINE147          *raster_buffer =
LINE148               *(undefined *)
LINE149                ((int)*(short *)(&DAT_1026f980 + (local_28 & 0xff) * 2) + uVar9 + param_1);
LINE150          raster_buffer[1] =
LINE151               *(undefined *)(((int)(uVar9 * 0x10000 + iVar14 + iVar10) >> 0x10) + param_1);
LINE152          raster_buffer[2] = *(undefined *)(uVar9 + (int)sVar1 + param_1);
LINE153          index = index + 1;
LINE154          raster_buffer = raster_buffer + 3;
LINE155        } while ((int)index < width);
LINE156      }
LINE157    }
LINE158    return (dword)pcVar6;
LINE159  }
```

The `raster_buffer` looks to be a table of three integers to store data, as we can see from LINE147 to LINE154. The index of this table is controlled by the `width` variable (corresponding to `size_X` below), meaning the table should be at least `width*3` in size. The size of `raster_buffer` is computed into the function `IGDIBStd::compute_raster_size` with the following pseudo-code:

```
LINE160  uint __thiscall IGDIBStd::compute_raster_size(HIGDIBINFO this)
LINE161  {
LINE162    longlong lVar1;
LINE163    uint uVar2;
LINE164    ulonglong uVar3;
LINE165
LINE166    lVar1 = (longlong)(int)(this->mys_struct_table_color).ptr_bits_per_channel_table *
LINE167            (longlong)(int)(this->mys_struct_table_color).ptr_channel_count;
LINE168    uVar3 = __allmul((uint)lVar1,(uint)((ulonglong)lVar1 >> 0x20),this->size_X,
LINE169                     (int)this->size_X >> 0x1f);
LINE170    lVar1 = (longlong)(uVar3 + 0x1f) >> 3;
LINE171    uVar2 = (uint)lVar1 & 0xfffffffc;
LINE172    if ((-1 < lVar1) && ((0 < (int)((longlong)(uVar3 + 0x1f) >> 0x23) || (0x7ffffffe < uVar2)))) {
LINE173      wrapper_thow_exception
LINE174                ((undefined *)0xfffffe6f,NULL,NULL,NULL,(undefined **)0x1022fa38,(undefined *)0x29);
LINE175    }
LINE176    return uVar2;
LINE177  }
```

We can see here effectively the size is depending of three variables: `ptr_bits_per_channel_table`, `ptr_channel_count` and `size_X` (corresponding to `width` above).
Theses values are set earlier in the execution while parsing the `SOF0` JPEG tag and they correspond respectively to component number, precision and width of the values present into this tag, and are read directly from the file.
To trigger the crash, an invalid image number of component from `SOF0` tag and specific Spectral selection values from `SOS` tag must be set.

Crash Information

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


KEY_VALUES_STRING: 1

    Key  : AV.Fault
    Value: Write

    Key  : Analysis.CPU.mSec
    Value: 2593

    Key  : Analysis.DebugAnalysisManager
    Value: Create

    Key  : Analysis.Elapsed.mSec
    Value: 32299

    Key  : Analysis.Init.CPU.mSec
    Value: 2093

    Key  : Analysis.Init.Elapsed.mSec
    Value: 47455

    Key  : Analysis.Memory.CommitPeak.Mb
    Value: 175

    Key  : Timeline.OS.Boot.DeltaSec
    Value: 168234

    Key  : Timeline.Process.Start.DeltaSec
    Value: 46

    Key  : WER.OS.Branch
    Value: vb_release

    Key  : WER.OS.Timestamp
    Value: 2019-12-06T14:06:00Z

    Key  : WER.OS.Version
    Value: 10.0.19041.1

    Key  : WER.Process.Version
    Value: 1.0.1.1


NTGLOBALFLAG:  2100000

APPLICATION_VERIFIER_FLAGS:  0

APPLICATION_VERIFIER_LOADED: 1

EXCEPTION_RECORD:  (.exr -1)
ExceptionAddress: 793c5eba (igCore19d!IG_mpi_page_set+0x000ca16a)
   ExceptionCode: c0000005 (Access violation)
  ExceptionFlags: 00000000
NumberParameters: 2
   Parameter[0]: 00000001
   Parameter[1]: 0db17000
Attempt to write to address 0db17000

FAULTING_THREAD:  0001096c

PROCESS_NAME:  Fuzzme.exe

WRITE_ADDRESS:  0db17000

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:  0db17000

STACK_TEXT:
WARNING: Stack unwind information not available. Following frames may be wrong.
0019f640 793c7c39     79500680 000000fd 00004101 igCore19d!IG_mpi_page_set+0xca16a
0019f6f0 793c3501     0c222f60 0019f9a8 0e172720 igCore19d!IG_mpi_page_set+0xcbee9
0019f774 793b4139     00004101 0e172720 0c222f60 igCore19d!IG_mpi_page_set+0xc77b1
0019f97c 793b00bf     0c222f60 0019f9a8 00000000 igCore19d!IG_mpi_page_set+0xb83e9
0019fa08 793c70b5     00000003 793c3030 0e172720 igCore19d!IG_mpi_page_set+0xb436f
0019fa24 793c6edd     0e172720 0c222f60 0000ffda igCore19d!IG_mpi_page_set+0xcb365
0019fa48 793c5021     0e172720 0c222f60 0019fa70 igCore19d!IG_mpi_page_set+0xcb18d
0019fa68 793c677a     0019ffc0 1000001d 0e162f70 igCore19d!IG_mpi_page_set+0xc92d1
0019faa8 792d10d9     1000001d 0e162f70 00000001 igCore19d!IG_mpi_page_set+0xcaa2a
0019fae0 79310557     00000000 0e162f70 0019fb30 igCore19d!IG_image_savelist_get+0xb29
0019fd5c 7930feb9     00000000 05454f68 00000001 igCore19d!IG_mpi_page_set+0x14807
0019fd7c 792a5777     00000000 05454f68 00000001 igCore19d!IG_mpi_page_set+0x14169
0019fd9c 00498a3a     05454f68 0019fe0c 004801a4 igCore19d!IG_load_file+0x47
0019fe14 00498e36     05454f68 0019fe8c 004801a4 Fuzzme!fuzzme+0x4a
0019fee4 004daa53     00000005 05344ef8 0534df20 Fuzzme!main+0x376
0019ff04 004da8a7     849468e5 004801a4 004801a4 Fuzzme!invoke_main+0x33
0019ff60 004da73d     0019ff70 004daad8 0019ff80 Fuzzme!__scrt_common_main_seh+0x157
0019ff68 004daad8     0019ff80 75e8fa29 00346000 Fuzzme!__scrt_common_main+0xd
0019ff70 75e8fa29     00346000 75e8fa10 0019ffdc Fuzzme!mainCRTStartup+0x8
0019ff80 77207a4e     00346000 2f1bf266 00000000 KERNEL32!BaseThreadInitThunk+0x19
0019ffdc 77207a1e     ffffffff 772288f5 00000000 ntdll!__RtlUserThreadStart+0x2f
0019ffec 00000000     004801a4 00346000 00000000 ntdll!_RtlUserThreadStart+0x1b


STACK_COMMAND:  ~0s ; .cxr ; kb

SYMBOL_NAME:  igCore19d!IG_mpi_page_set+ca16a

MODULE_NAME: igCore19d

IMAGE_NAME:  igCore19d.dll
```

```
FAILURE_BUCKET_ID:  INVALID_POINTER_WRITE_AVRF_c0000005_igCore19d.dll!IG_mpi_page_set

OS_VERSION:  10.0.19041.1

BUILDLAB_STR:  vb_release

OSPLATFORM_TYPE:  x86

OSNAME:  Windows 10

IMAGE_VERSION:  19.9.0.0

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

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

Timeline

2021-04-28 - Vendor Disclosure
2021-05-31 - Vendor Patched
2021-06-01 - Public Release

CREDIT

Discovered by Emmanuel Tacheau of Cisco Talos.

VULNERABILITY REPORTS

PREVIOUS REPORT TALOS-2021-1261

NEXT REPORT TALOS-2021-1286

© 2022 Cisco Systems, Inc. and/or its affiliates. All rights reserved. View our Privacy Policy.