Talos Vulnerability Report

TALOS-2021-1257

Accusoft ImageGear JPG sof_nb_comp header processing out-of-bounds write vulnerability

JUNE 1, 2021

CVE NUMBER

CVE-2021-21793

Summary

An out-of-bounds write vulnerability exists in the JPG sof_nb_comp header processing functionality of Accusoft ImageGear 19.8 and 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 19.8
Accusoft ImageGear 19.9

Product URLs

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

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-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 `jpeg_raster_set` 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:

```
(86a0.8438): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
eax=d0d0d0d0 ebx=00000001 ecx=00000000 edx=0f349018 esi=0f348ff8 edi=0019f6e0
eip=6aba743b esp=0019f660 ebp=0019f6f0 iopl=0         nv up ei pl zr na pe nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00010246
igCore19d!IG_mpi_page_set+0xcb6eb:
6aba743b 8902            mov     dword ptr [edx],eax  ds:002b:0f349018=????????
```

When we look at the `edx` memory allocation we can see the buffer allocated is very small, only 1 byte:

```
0:000> !heap -p -a edx
    address 0f349018 found in
    _DPH_HEAP_ROOT @ 3f41000
    in busy allocation (  DPH_HEAP_BLOCK:         UserAddr         UserSize -         VirtAddr         VirtSize)
                                1503364c:          f348ff8                1 -          f348000             2000
    70b5a8b0 verifier!AVrfDebugPageHeapAllocate+0x00000240
    7777ef0e ntdll!RtlDebugAllocateHeap+0x00000039
    776e6150 ntdll!RtlpAllocateHeap+0x000000f0
    776e57fe ntdll!RtlpAllocateHeapInternal+0x000003ee
    776e53fe ntdll!RtlAllocateHeap+0x0000003e
    6b53dcff MSVCR110!malloc+0x00000049
    6aad61de igCore19d!AF_memm_alloc+0x0000001e
    6ab919e5 igCore19d!IG_mpi_page_set+0x000b5c95
    6ab8fce3 igCore19d!IG_mpi_page_set+0x000b3f93
    6aba70b5 igCore19d!IG_mpi_page_set+0x000cb365
    6aba6f85 igCore19d!IG_mpi_page_set+0x000cb235
    6aba5021 igCore19d!IG_mpi_page_set+0x000c92d1
    6aba677a igCore19d!IG_mpi_page_set+0x000caa2a
    6aab10d9 igCore19d!IG_image_savelist_get+0x00000b29
    6aaf0557 igCore19d!IG_mpi_page_set+0x00014807
    6aaefeb9 igCore19d!IG_mpi_page_set+0x00014169
    6aa85777 igCore19d!IG_load_file+0x00000047
    00498a3a Fuzzme!fuzzme+0x0000004a [C:\Users\etach\Documents\work\gitlab\ImageGear\Fuzzme\Fuzzme.cpp @ 282]
    00498e36 Fuzzme!main+0x00000376 [C:\Users\etach\Documents\work\gitlab\ImageGear\Fuzzme\Fuzzme.cpp @ 477]
    004daa53 Fuzzme!invoke_main+0x00000033 [d:\agent\_work\57\s\src\vctools\crt\vcstartup\src\startup\exe_common.inl @ 78]
    004da8a7 Fuzzme!__scrt_common_main_seh+0x00000157 [d:\agent\_work\57\s\src\vctools\crt\vcstartup\src\startup\exe_common.inl @ 288]
    004da73d Fuzzme!__scrt_common_main+0x0000000d [d:\agent\_work\57\s\src\vctools\crt\vcstartup\src\startup\exe_common.inl @ 331]
    004daad8 Fuzzme!mainCRTStartup+0x00000008 [d:\agent\_work\57\s\src\vctools\crt\vcstartup\src\startup\exe_main.cpp @ 17]
    763afa29 KERNEL32!BaseThreadInitThunk+0x00000019
    777076b4 ntdll!__RtlUserThreadStart+0x0000002f
    77707684 ntdll!_RtlUserThreadStart+0x0000001b
```

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

```
LINE1    void jpeg_raster_set(jpeg_dec *jpeg_dec,SOF_object *SOF_Object,jpeg_related *jpeg_related,
LINE2                         undefined4 param_4,int witdh,int height,byte *min_height,short *raster_buffer,
LINE3                         uint size_raster_buffer,int SOF_type,int param_11,int value_to_8,void *param_13,
LINE4                         int param_14,int param_15)
LINE5
LINE6    {
        [...]
LINE54     uVar18 = (undefined)in_stack_ffffff78;
LINE55     local_8 = DAT_102bcea8 ^ (uint)&stack0xfffffffc;
LINE56     max_loop = jpeg_dec->enforced_8 * value_to_8;
LINE57     if (SOF_type == 2) {
LINE58       _sof_nb_component = *(byte *)&(SOF_Object->SOF).component;
LINE59     }
LINE60     else {
LINE61       _sof_nb_component = *(byte *)&SOF_Object->possible_num_component_or_color_channel;
LINE62     }
LINE63     nr_component_buffer_data = (byte *)SOF_Object->nr_component_buffer_data;
LINE64     if (SOF_type == 0) {
LINE65   LAB_1013741f:
LINE66                        /* ---------------------------------
LINE67                            Num of component = 1 e.g grayscale
LINE68                           --------------------------------- */
LINE69       __sof_nb_component = (uint)_sof_nb_component;
LINE70       if (__sof_nb_component != 0) {
LINE71         _color_table_data = (undefined4 *)(nr_component_buffer_data + 0x20);
LINE72         uVar10 = __sof_nb_component;
LINE73         piVar15 = local_18;
LINE74         while (_sof_nb_comp = __sof_nb_component, uVar10 != 0) {
LINE75           uVar10 = uVar10 - 1;
LINE76           *piVar15 = 0;
LINE77           piVar15 = piVar15 + 1;
LINE78         }
LINE79         do {
LINE80           *_color_table_data = _color_table_data[-7];
LINE81           _color_table_data = _color_table_data + 0x14;
LINE82           _sof_nb_comp = _sof_nb_comp - 1;
LINE83         } while (_sof_nb_comp != 0);
LINE84       }
            [...]
LINE720    return;
LINE721  }
```

The out-of-bounds is occurring in LINE80 where we can see the buffer `_color_table_data`. The loop is controlled by the variable `_sof_nb_comp`, derived from the `SOF_Object` component as read directly from the file.

The buffer `_color_table_data` is allocated earlier into the function `possible_build_color_channel_data` with the following pseudo-code in LINE773:

```
LINE722  int possible_build_color_channel_data
LINE723              (jpeg_dec *jpeg_dec,read_buffer *read_buffer,SOF_object *SOF_object,
LINE724              int *related_type_sof,uint *color_channel_data,void **possible_color_table_data,
LINE725              int *possible_boolean_value,int *param_8)
LINE726  {
            [...]
LINE752    prVar10 = read_buffer;
LINE753    kind_heap = jpeg_dec->kind_of_heap;
LINE754    pbVar1 = (byte *)read_buffer->read_buffer_data;
LINE755    local_18 = 0;
LINE756    local_c = 0;
LINE757    dVar12 = 0;
LINE758    local_1c = 0;
LINE759    read_buffer = NULL;
LINE760    if (*(short *)&prVar10->buff_mem != -0x26) {
LINE761      iVar5 = AF_err_record_set("..\\..\\..\\..\\Common\\Formats\\jpeg_dec.c",0x525,-0x1310,0,0,0,NULL
LINE762                                 );
LINE763      return iVar5;
LINE764    }
LINE765    if (possible_color_table_data == NULL) {
LINE766      iVar5 = AF_err_record_set("..\\..\\..\\..\\Common\\Formats\\jpeg_dec.c",0x528,-0x1310,0,0,0,NULL
LINE767                                 );
LINE768      return iVar5;
LINE769    }
LINE770    *possible_color_table_data = NULL;
LINE771    nr_comp_sos = *pbVar1;
LINE772    *color_channel_data = (uint)nr_comp_sos;
LINE773    _color_table_data = AF_memm_alloc(kind_heap,(uint)nr_comp_sos * 0x50);
LINE774    if (_color_table_data == NULL) {
LINE775      iVar5 = AF_err_record_set("..\\..\\..\\..\\Common\\Formats\\jpeg_dec.c",0x531,-1000,0,0,0,NULL);
LINE776      return iVar5;
LINE777    }
            [...]
LINE936  }
```

The size is controlled by the `nr_comp_sos` variable, which is directly read from the file.

If the `nr_comp_sos` is null then the size computed for the buffer is null and as there is no null check. The function `AF_memm_alloc` is a wrapper to `malloc`, thus when passing a null value it returns a buffer of one byte.

Thus in our case the buffer is only 1 byte long, most of the assignments happening inside function `jpeg_raster_set` are out-of-bounds heap writes which lead to memory corruption and possibly code execution.

Crash Information

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


KEY_VALUES_STRING: 1

    Key  : AV.Fault
    Value: Write

    Key  : Analysis.CPU.mSec
    Value: 7312

    Key  : Analysis.DebugAnalysisProvider.CPP
    Value: Create: 8007007e on DESKTOP-4DAOCFH

    Key  : Analysis.DebugData
    Value: CreateObject

    Key  : Analysis.DebugModel
    Value: CreateObject

    Key  : Analysis.Elapsed.mSec
    Value: 100036

    Key  : Analysis.Memory.CommitPeak.Mb
    Value: 188

    Key  : Analysis.System
    Value: CreateObject

    Key  : Timeline.OS.Boot.DeltaSec
    Value: 91388

    Key  : Timeline.Process.Start.DeltaSec
    Value: 18

    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


ADDITIONAL_XML: 1

OS_BUILD_LAYERS: 1

NTGLOBALFLAG:  2100000

APPLICATION_VERIFIER_FLAGS:  0

APPLICATION_VERIFIER_LOADED: 1

EXCEPTION_RECORD:  (.exr -1)
ExceptionAddress: 6aba743b (igCore19d!IG_mpi_page_set+0x000cb6eb)
   ExceptionCode: c0000005 (Access violation)
  ExceptionFlags: 00000000
NumberParameters: 2
   Parameter[0]: 00000001
   Parameter[1]: 0f349018
Attempt to write to address 0f349018

FAULTING_THREAD:  00008438

PROCESS_NAME:  Fuzzme.exe

WRITE_ADDRESS:  0f349018

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:  0f349018

STACK_TEXT:
WARNING: Stack unwind information not available. Following frames may be wrong.
0019f6f0 6aba3501     09d72f60 0019f9a8 0c35a720 igCore19d!IG_mpi_page_set+0xcb6eb
0019f774 6ab94139     00001a2d 0c35a720 09d72f60 igCore19d!IG_mpi_page_set+0xc77b1
0019f97c 6ab900bf     09d72f60 0019f9a8 00000000 igCore19d!IG_mpi_page_set+0xb83e9
0019fa08 6aba70b5     00000000 6aba3030 0c35a720 igCore19d!IG_mpi_page_set+0xb436f
0019fa24 6aba6f85     0c35a720 09d72f60 0000ffda igCore19d!IG_mpi_page_set+0xcb365
0019fa48 6aba5021     0c35a720 09d72f60 0019fa70 igCore19d!IG_mpi_page_set+0xcb235
0019fa68 6aba677a     0019ffc1 1000001d 0bc60f70 igCore19d!IG_mpi_page_set+0xc92d1
0019faa8 6aab10d9     1000001d 0bc60f70 00000001 igCore19d!IG_mpi_page_set+0xcaa2a
0019fae0 6aaf0557     00000000 0bc60f70 0019fb30 igCore19d!IG_image_savelist_get+0xb29
0019fd5c 6aaefeb9     00000000 05444f88 00000001 igCore19d!IG_mpi_page_set+0x14807
0019fd7c 6aa85777     00000000 05444f88 00000001 igCore19d!IG_mpi_page_set+0x14169
0019fd9c 00498a3a     05444f88 0019fe0c 004801a4 igCore19d!IG_load_file+0x47
0019fe14 00498e36     05444f88 0019fe8c 004801a4 Fuzzme!fuzzme+0x4a
0019fee4 004daa53     00000005 05384f20 0538df20 Fuzzme!main+0x376
0019ff04 004da8a7     fd9de02f 004801a4 004801a4 Fuzzme!invoke_main+0x33
0019ff60 004da73d     0019ff70 004daad8 0019ff80 Fuzzme!__scrt_common_main_seh+0x157
0019ff68 004daad8     0019ff80 763afa29 002a3000 Fuzzme!__scrt_common_main+0xd
0019ff70 763afa29     002a3000 763afa10 0019ffdc Fuzzme!mainCRTStartup+0x8
0019ff80 777076b4     002a3000 a8e56cc7 00000000 KERNEL32!BaseThreadInitThunk+0x19
0019ffdc 77707684     ffffffff 77727415 00000000 ntdll!__RtlUserThreadStart+0x2f
0019ffec 00000000     004801a4 002a3000 00000000 ntdll!_RtlUserThreadStart+0x1b


STACK_COMMAND:  ~0s ; .cxr ; kb
```

```
SYMBOL_NAME:  igCore19d!IG_mpi_page_set+cb6eb

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

2021-02-25 - Vendor Disclosure
2021-05-31 - Vendor Patched
2021-06-01 - Public Release

CREDIT

Discovered by Emmanuel Tacheau of Cisco Talos.

VULNERABILITY REPORTS

PREVIOUS REPORT NEXT REPORT

TALOS-2021-1243 TALOS-2021-1261

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