Talos Vulnerability Report

TALOS-2021-1264

Accusoft ImageGear PSD read_icc_icCurve_data heap-based buffer overflow vulnerability

MARCH 16, 2021

CVE NUMBER

CVE-2021-21795

Summary

A heap-based buffer overflow vulnerability exists in the PSD read_icc_icCurve_data functionality of Accusoft ImageGear 19.9. A specially crafted malformed file can lead to an integer overflow that, in turn, leads to a heap buffer overflow. 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

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

CWE

CWE-122 - Heap-based Buffer Overflow

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 PSD file can lead to an out-of-bounds write in `read_icc_icCurve_data` function, due to a buffer overflow caused by an integer overflow while allocating memory. Trying to load a malformed PSD file, we end up in the following situation:

```
(bbb0.dfa0): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
eax=0dc6cff8 ebx=6ba28620 ecx=00000000 edx=00000002 esi=0019f38c edi=0c25aff8
eip=699204fd esp=0019f2dc ebp=0019f370 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_cpm_profiles_reset+0x56ed:
699204fd f20f114cd0f8    movsd   mmword ptr [eax+edx*8-8],xmm1 ds:002b:0dc6d000=????????????????
```

The crash is happening in function `read_icc_icCurve_data` in LINE50 during a do-while loop (from LINE48 to LINE54) controlled by the `curve_data->count`:

```
LINE1   dword read_icc_icCurve_data(uint **param_1,curve_type_data *curve_data)
LINE2   {
LINE3     uint size;
LINE4     uint *buffer_for_curves_values;
LINE5     undefined extraout_DL;
LINE6     dword return_status;
LINE7     uint index;
LINE8     undefined4 uVar1;
LINE9     int iVar2;
LINE10    uint *puVar3;
LINE11    uint *puVar4;
LINE12    undefined uVar5;
LINE13    char local_88 [128];
LINE14    uint local_8;
LINE15
LINE16    local_8 = DAT_102bcea8 ^ (uint)&stack0xfffffffc;
LINE17    uVar5 = 0;
LINE18    if (param_1[1] < &DAT_00000004) {
LINE19                      /* char * _Format for sprintf */
LINE20                      /* char * _Dest for sprintf */
LINE21      sprintf(local_88,"Unexpected end of data while reading tag:%Xh, type:%s",param_1[3],"icCurve");
LINE22      iVar2 = AF_err_record_set("..\\..\\..\\..\\Common\\Core\\ICCProfile.c",0x31b,-1,0,param_1[1],4,
LINE23                                local_88);
LINE24      uVar5 = (undefined)iVar2;
LINE25    }
LINE26    else {
LINE27                      /* get count value from tag */
LINE28      size = rotate_bytes(**param_1);
LINE29      curve_data->count = size;
LINE30      *param_1 = *param_1 + 1;
LINE31      param_1[1] = param_1[1] + -1;
LINE32    }
LINE33    if (param_1[1] < (uint *)(curve_data->count * 2)) {
LINE34                      /* char * _Format for sprintf */
LINE35                      /* char * _Dest for sprintf */
LINE36      sprintf(local_88,"Unexpected end of data while reading tag:%Xh, type:%s",param_1[3],"icCurve");
LINE37      puVar4 = (uint *)(curve_data->count * 2);
LINE38      puVar3 = param_1[1];
LINE39      iVar2 = -1;
LINE40      uVar1 = 0x336;
LINE41    }
LINE42    else {
LINE43      buffer_for_curves_values = (uint *)AF_memm_alloc((uint)param_1[2],curve_data->count << 3);
LINE44      curve_data->buffer_actual_curve_values = buffer_for_curves_values;
LINE45      if (buffer_for_curves_values != NULL) {
LINE46        index = 0;
LINE47        if (curve_data->count != 0) {
LINE48          do {
LINE49            index = index + 1;
LINE50            *(double *)(curve_data->buffer_actual_curve_values + index * 2 + -2) =
LINE51                 (double)(uint)(*(ushort *)*param_1 >> 8) * 0.00390625 +
LINE52                 (double)(uint)(*(ushort *)*param_1 & 0xff);
LINE53            *param_1 = (uint *)((int)*param_1 + 2);
LINE54          } while (index < curve_data->count);
LINE55        }
LINE56        param_1[1] = (uint *)((int)param_1[1] + curve_data->count * -2);
LINE57        return_status = kind_of_fastfail(local_8 ^ (uint)&stack0xfffffffc,(char)index,uVar5);
LINE58        return return_status;
LINE59      }
LINE60                      /* char * _Format for sprintf */
LINE61                      /* char * _Dest for sprintf */
LINE62      sprintf(local_88,"Out of memory while reading tag:%Xh, type:%s",param_1[3],"icCurve");
LINE63      puVar4 = param_1[1];
LINE64      puVar3 = NULL;
LINE65      iVar2 = -1000;
LINE66      uVar1 = 0x325;
LINE67    }
LINE68    AF_err_record_set("..\\..\\..\\..\\Common\\Core\\ICCProfile.c",uVar1,iVar2,0,puVar3,puVar4,
LINE69                      local_88);
LINE70    return_status = kind_of_fastfail(local_8 ^ (uint)&stack0xfffffffc,extraout_DL,uVar5);
LINE71    return return_status;
LINE72  }
```

The size for the memory buffer `buffer_for_curves_values`, allocated in LINE43, is computed by `curve_data->count << 3` and is prone to an integer overflow.
In the proof-of-concept, the `curve_data->count` is read from the file and has value “0x80000001”. The multiplication at LINE43 results in the value “8”, hence a buffer of only 8 bytes is allocated.

```
0:000> !heap -p -a eax
    address 0dc6cff8 found in
    _DPH_HEAP_ROOT @ 3ed1000
    in busy allocation (  DPH_HEAP_BLOCK:         UserAddr         UserSize -         VirtAddr         VirtSize)
                                 bc602d8:          dc6cff8                8 -          dc6c000             2000
    6e3aa8b0 verifier!AVrfDebugPageHeapAllocate+0x00000240
    7777ef0e ntdll!RtlDebugAllocateHeap+0x00000039
    776e6150 ntdll!RtlpAllocateHeap+0x000000f0
    776e57fe ntdll!RtlpAllocateHeapInternal+0x000003ee
    776e53fe ntdll!RtlAllocateHeap+0x0000003e
    6ba1dcff MSVCR110!malloc+0x00000049
    699361de igCore19d!AF_memm_alloc+0x0000001e
    69920475 igCore19d!IG_cpm_profiles_reset+0x00005665
    6991c4f2 igCore19d!IG_cpm_profiles_reset+0x000016e2
    6991fb42 igCore19d!IG_cpm_profiles_reset+0x00004d32
    6991f628 igCore19d!IG_cpm_profiles_reset+0x00004818
    69919990 igCore19d!IG_gctrl_item_set+0x00005bc0
    699290c6 igCore19d!IG_cpm_profiles_reset+0x0000e2b6
    69a2c79d igCore19d!IG_mpi_page_set+0x000f0a4d
    69a2bb32 igCore19d!IG_mpi_page_set+0x000efde2
    69a2b5aa igCore19d!IG_mpi_page_set+0x000ef85a
    699110d9 igCore19d!IG_image_savelist_get+0x00000b29
    69950557 igCore19d!IG_mpi_page_set+0x00014807
    6994feb9 igCore19d!IG_mpi_page_set+0x00014169
    698e5777 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
```

The `count` value is directly read from the file from the icc curv tag data of the PSD file. An adequate value chosen for the count leads to the integer overflow, which in turn leads to allocating a very small buffer which can lead to an out-of-bounds write. Note that the contents written out of bounds are read from the file and are thus under the attacker's control.

Crash Information

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


KEY_VALUES_STRING: 1

    Key  : AV.Fault
    Value: Write

    Key  : Analysis.CPU.mSec
    Value: 3233

    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: 34473

    Key  : Analysis.Memory.CommitPeak.Mb
    Value: 187

    Key  : Analysis.System
    Value: CreateObject

    Key  : Timeline.OS.Boot.DeltaSec
    Value: 975007

    Key  : Timeline.Process.Start.DeltaSec
    Value: 197

    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: 699204fd (igCore19d!IG_cpm_profiles_reset+0x000056ed)
   ExceptionCode: c0000005 (Access violation)
  ExceptionFlags: 00000000
NumberParameters: 2
   Parameter[0]: 00000001
   Parameter[1]: 0dc6d000
Attempt to write to address 0dc6d000

FAULTING_THREAD:  0000dfa0

PROCESS_NAME:  Fuzzme.exe

WRITE_ADDRESS:  0dc6d000

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:  0dc6d000

STACK_TEXT:
WARNING: Stack unwind information not available. Following frames may be wrong.
0019f370 6991c4f2     0019f38c 0c25aff8 0f092f94 igCore19d!IG_cpm_profiles_reset+0x56ed
0019f428 6991fb42     0f092f94 0000000e 72545243 igCore19d!IG_cpm_profiles_reset+0x16e2
0019f4d8 6991f628     0f092f94 0000000e 72545243 igCore19d!IG_cpm_profiles_reset+0x4d32
0019f510 69919990     00000000 00000004 1000001f igCore19d!IG_cpm_profiles_reset+0x4818
0019f538 699290c6     1000001f 0f092dd0 00000230 igCore19d!IG_gctrl_item_set+0x5bc0
0019f550 69a2c79d     0c1d0f40 0f092dd0 00000230 igCore19d!IG_cpm_profiles_reset+0xe2b6
0019f614 69a2bb32     0019fb30 1000001e 09468f60 igCore19d!IG_mpi_page_set+0xf0a4d
0019f650 69a2b5aa     0019fb30 0019f678 0019f6a0 igCore19d!IG_mpi_page_set+0xefde2
0019faa8 699110d9     0019fb30 0946cfe0 00000001 igCore19d!IG_mpi_page_set+0xef85a
0019fae0 69950557     00000000 0946cfe0 0019fb30 igCore19d!IG_image_savelist_get+0xb29
0019fd5c 6994feb9     00000000 054d4f88 00000001 igCore19d!IG_mpi_page_set+0x14807
0019fd7c 698e5777     00000000 054d4f88 00000001 igCore19d!IG_mpi_page_set+0x14169
0019fd9c 00498a3a     054d4f88 0019fe0c 004801a4 igCore19d!IG_load_file+0x47
0019fe14 00498e36     054d4f88 0019fe8c 004801a4 Fuzzme!fuzzme+0x4a
0019fee4 004daa53     00000005 05414f20 0541df20 Fuzzme!main+0x376
0019ff04 004da8a7     c4e97d5b 004801a4 004801a4 Fuzzme!invoke_main+0x33
0019ff60 004da73d     0019ff70 004daad8 0019ff80 Fuzzme!__scrt_common_main_seh+0x157
0019ff68 004daad8     0019ff80 763afa29 00224000 Fuzzme!__scrt_common_main+0xd
0019ff70 763afa29     00224000 763afa10 0019ffdc Fuzzme!mainCRTStartup+0x8
0019ff80 777076b4     00224000 a834ca50 00000000 KERNEL32!BaseThreadInitThunk+0x19
0019ffdc 77707684     ffffffff 7772742d 00000000 ntdll!__RtlUserThreadStart+0x2f
0019ffec 00000000     004801a4 00224000 00000000 ntdll!_RtlUserThreadStart+0x1b


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

```
SYMBOL_NAME:  igCore19d!IG_cpm_profiles_reset+56ed

MODULE_NAME: igCore19d

IMAGE_NAME:  igCore19d.dll

FAILURE_BUCKET_ID:  INVALID_POINTER_WRITE_AVRF_c0000005_igCore19d.dll!IG_cpm_profiles_reset

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:  {749e662e-5382-aab8-f02d-cecd73653ce6}

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

Timeline

2021-03-16 - Vendor Disclosure

2021-05-31 - Public Release

CREDIT

Discovered by Emmanuel Tacheau of Cisco Talos.

VULNERABILITY REPORTS

PREVIOUS REPORT TALOS-2020-1226

NEXT REPORT TALOS-2021-1236

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