Talos Vulnerability Report

TALOS-2022-1449

Accusoft ImageGear ioca_mys_rgb_allocate memory corruption vulnerability

MAY 2, 2022

CVE NUMBER

CVE-2022-22137

Summary

A memory corruption vulnerability exists in the ioca_mys_rgb_allocate functionality of Accusoft ImageGear 19.10. A specially-crafted malformed file can lead to an arbitrary free. An attacker can provide a malicious file to trigger this vulnerability.

Tested Versions

Accusoft ImageGear 19.10

Product URLs

ImageGear - 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.

Trying to load a malformed IOCA file, we end up with the following situation:

```
(1bf4.175c): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
verifier_71920000!AVrfpDphFindBusyMemoryNoCheck+0xb8:
71928758 813abbbbcdab    cmp     dword ptr [edx],0ABCDBBBBh
ds:002b:d0d0d0a0=????????
```

When looking at the call stack, we can observe the crash is happening during the free process as detailed below:

```
STACK_TEXT:
0019f310 71928875     04d71000 d0d0d0c0 00000000
verifier_71920000!AVrfpDphFindBusyMemoryNoCheck+0xb8
0019f334 71928ae0     04d71000 d0d0d0c0 0019f3c4
verifier_71920000!AVrfpDphFindBusyMemory+0x15
0019f350 7192aad0     04d71000 d0d0d0c0 04d70000
verifier_71920000!AVrfpDphFindBusyMemoryAndRemoveFromBusyList+0x20
0019f36c 7739f966     04d70000 01000002 d0d0d0c0
verifier_71920000!AVrfDebugPageHeapFree+0x90
0019f3d4 77303d46     d0d0d0c0 3b0d5201 00000000
ntdll_772c0000!RtlDebugFreeHeap+0x3e
0019f530 7734791d     00000000 d0d0d0c0 d0d0d0c0 ntdll_772c0000!RtlpFreeHeap+0xd6
0019f58c 77303c16     00000000 00000000 00000000
ntdll_772c0000!RtlpFreeHeapInternal+0x783
0019f5ac 7146dac2     04d70000 00000000 d0d0d0c0 ntdll_772c0000!RtlFreeHeap+0x46
0019f5c0 71606b22     d0d0d0c0 00000000 09b60fa8 MSVCR110!free+0x1a
0019f5d4 715ed0a3     00000000 0bd49ff0 98cbb32c
igCore19d!IG_comm_is_comp_exist+0x4ad2
0019f608 7164d641     0f83cfe0 00000000 0d0c4ff0 igCore19d!GPb_image_associate+0xd33
0019f62c 7162a14e     0019fdb0 0b44aff8 00000000 igCore19d!IG_mpi_page_set+0x11611
0019f64c 716dc57d     0019fc3c 0b44aff8 00000000
igCore19d!IG_cpm_profiles_reset+0xf03e
0019f674 716e271b     0019fc3c 1000001f 1271f000 igCore19d!IG_mpi_page_set+0xa054d
0019f708 716e0cc0     0019fc3c 1000001f 0afa2ff8 igCore19d!IG_mpi_page_set+0xa66eb
0019fbb4 716113d9     0019fc3c 0afa2ff8 00000001 igCore19d!IG_mpi_page_set+0xa4c90
0019fbec 716508d7     00000000 0afa2ff8 0019fc3c
igCore19d!IG_image_savelist_get+0xb29
0019fe68 71650239     00000000 0019ff10 00000001 igCore19d!IG_mpi_page_set+0x148a7
0019fe88 715e5757     00000000 0019ff10 00000001 igCore19d!IG_mpi_page_set+0x14209
0019fea8 00402219     0019ff10 0019febc 00000001 igCore19d!IG_load_file+0x47
0019fec0 00402524     0019ff10 0019fef8 052e2f48 Fuzzme!fuzzme+0x19
0019ff28 0040668d     00000005 052dcf78 052e2f48 Fuzzme!fuzzme+0x324
0019ff70 7514fa29     00353000 7514fa10 0019ffdc Fuzzme!fuzzme+0x448d
0019ff80 77327a9e     00353000 3b0d58ed 00000000 KERNEL32!BaseThreadInitThunk+0x19
0019ffdc 77327a6e     ffffffff 77348a68 00000000
ntdll_772c0000!__RtlUserThreadStart+0x2f
0019ffec 00000000     00406715 00353000 00000000
ntdll_772c0000!_RtlUserThreadStart+0x1b
```

Inspecting the argument indicates the parameter to `free()` is not a valid address: `0xd0d0d0c0`.

Tracing back through the call stack leads us to the function `IGDIBRunEnds::delete_table_mys_rbg_ptr` with the following pseudo code:

```
LINE1  void __thiscall IGDIBRunEnds::delete_table_mys_rbg_ptr(IGDIBRunEnds *this,int
pixpos,int some_buffer)
LINE2  {
LINE3    if (this->mys_RGB != (mys_RGB *)this->table_mys_rgb[pixpos]) {
LINE4       operator_delete((mys_RGB *)this->table_mys_rgb[pixpos]);
LINE5    }
LINE6    if (some_buffer == 0) {
LINE7       some_buffer = (int)this->mys_RGB;
LINE8    }
LINE9    this->table_mys_rgb[pixpos] = some_buffer;
LINE10   this->field_0x48 = 1;
LINE11   return;
LINE12 }
```

The free is corresponding to the `operator_delete` called in LINE4, which is indexed by `pixpos` to delete a specific element.

When looking at how this is constructed, the `table_mys_rgb` object in this case lands in the following code:

```
LINE13 void __thiscall IGDIBRunEnds::FUN_743f6aa0(IGDIBRunEnds *this)
LINE14 {
LINE15   if (this->table_mys_rgb == (dword *)0x0) {
LINE16     size_to_allocate = this->size_Y * 4;
LINE17     buffer = (dword *)operator_new(-(uint)((int)(size_to_allocate >> 0x20) !=
0) |
LINE18                                    (uint)size_to_allocate);
LINE19     if (this->table_mys_rgb != buffer) {
LINE20       operator_delete(this->table_mys_rgb);
LINE21       this->table_mys_rgb = buffer;
LINE22     }
LINE23     size_y = this->size_Y;
LINE24     while (size_y != 0) {
LINE25       size_y = size_y - 1;
LINE26       this->table_mys_rgb[size_y] = (dword)this->mys_RGB;
LINE27     }
LINE28   }
LINE29   return;
LINE30 }
```

So we can see the allocation in LINE17, followed by a while loop between LINE24 and LINE27 to fill the memory allocated. Now the issue is happening when `size_Y` read from the file is null, thus the allocation of `buffer` becomes uncontrolled and a zero byte allocation is made in LINE17, returned by a `malloc` null operation. Thus the while loop (LINE24) is not processed and the flow continues without initializing any element in `table_mys_rgb`. When the

thread reaches the function `IGDIBRunEnds::delete_table_mys_rbg_ptr`, even with a null `pixpos` value, the pointer at `table_mys_rgb[0]` (which is not initialized, since `table_mys_rgb` has a size of 0) is freed via `operator_delete`, possibly leading to an arbitrary free.

Crash Information

```
0:000:x86> !analyze -v
*******************************************************************************
*                                                                             *
*                        Exception Analysis                                   *
*                                                                             *
*******************************************************************************


KEY_VALUES_STRING: 1

    Key  : AV.Fault
    Value: Read

    Key  : Analysis.CPU.mSec
    Value: 1905

    Key  : Analysis.DebugAnalysisManager
    Value: Create

    Key  : Analysis.Elapsed.mSec
    Value: 34355

    Key  : Analysis.Init.CPU.mSec
    Value: 5030

    Key  : Analysis.Init.Elapsed.mSec
    Value: 119236

    Key  : Analysis.Memory.CommitPeak.Mb
    Value: 123

    Key  : Timeline.OS.Boot.DeltaSec
    Value: 1162

    Key  : Timeline.Process.Start.DeltaSec
    Value: 84

    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:  2000000

PROCESS_BAM_CURRENT_THROTTLED: 0

PROCESS_BAM_PREVIOUS_THROTTLED: 0

APPLICATION_VERIFIER_FLAGS:  0
```

```
APPLICATION_VERIFIER_LOADED: 1

EXCEPTION_RECORD:  (.exr -1)
ExceptionAddress: 71928758
(verifier_71920000!AVrfpDphFindBusyMemoryNoCheck+0x000000b8)
   ExceptionCode: c0000005 (Access violation)
  ExceptionFlags: 00000000
NumberParameters: 2
   Parameter[0]: 00000000
   Parameter[1]: d0d0d0a0
Attempt to read from address d0d0d0a0

FAULTING_THREAD:  0000175c

PROCESS_NAME:  Fuzzme.exe

READ_ADDRESS:  d0d0d0a0

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:  00000000

EXCEPTION_PARAMETER2:  d0d0d0a0

STACK_TEXT:
0019f310 71928875     04d71000 d0d0d0c0 00000000
verifier_71920000!AVrfpDphFindBusyMemoryNoCheck+0xb8
0019f334 71928ae0     04d71000 d0d0d0c0 0019f3c4
verifier_71920000!AVrfpDphFindBusyMemory+0x15
0019f350 7192aad0     04d71000 d0d0d0c0 04d70000
verifier_71920000!AVrfpDphFindBusyMemoryAndRemoveFromBusyList+0x20
0019f36c 7739f966     04d70000 01000002 d0d0d0c0
verifier_71920000!AVrfDebugPageHeapFree+0x90
0019f3d4 77303d46     d0d0d0c0 3b0d5201 00000000
ntdll_772c0000!RtlDebugFreeHeap+0x3e
0019f530 7734791d     00000000 d0d0d0c0 d0d0d0c0 ntdll_772c0000!RtlpFreeHeap+0xd6
0019f58c 77303c16     00000000 00000000 00000000
ntdll_772c0000!RtlpFreeHeapInternal+0x783
0019f5ac 7146dac2     04d70000 00000000 d0d0d0c0 ntdll_772c0000!RtlFreeHeap+0x46
0019f5c0 71606b22     d0d0d0c0 00000000 09b60fa8 MSVCR110!free+0x1a
WARNING: Stack unwind information not available. Following frames may be wrong.
0019f5d4 715ed0a3     00000000 0bd49ff0 98cbb32c
igCore19d!IG_comm_is_comp_exist+0x4ad2
0019f608 7164d641     0f83cfe0 00000000 0d0c4ff0 igCore19d!GPb_image_associate+0xd33
0019f62c 7162a14e     0019fdb0 0b44aff8 00000000 igCore19d!IG_mpi_page_set+0x11611
0019f64c 716dc57d     0019fc3c 0b44aff8 00000000
igCore19d!IG_cpm_profiles_reset+0xf03e
0019f674 716e271b     0019fc3c 1000001f 1271f000 igCore19d!IG_mpi_page_set+0xa054d
0019f708 716e0cc0     0019fc3c 1000001f 0afa2ff8 igCore19d!IG_mpi_page_set+0xa66eb
0019fbb4 716113d9     0019fc3c 0afa2ff8 00000001 igCore19d!IG_mpi_page_set+0xa4c90
0019fbec 716508d7     00000000 0afa2ff8 0019fc3c
igCore19d!IG_image_savelist_get+0xb29
0019fe68 71650239     00000000 0019ff10 00000001 igCore19d!IG_mpi_page_set+0x148a7
0019fe88 715e5757     00000000 0019ff10 00000001 igCore19d!IG_mpi_page_set+0x14209
0019fea8 00402219     0019ff10 0019febc 00000001 igCore19d!IG_load_file+0x47
0019fec0 00402524     0019ff10 0019fef8 052e2f48 Fuzzme!fuzzme+0x19
```

```
0019ff28 0040668d     00000005 052dcf78 052e2f48 Fuzzme!fuzzme+0x324
0019ff70 7514fa29     00353000 7514fa10 0019ffdc Fuzzme!fuzzme+0x448d
0019ff80 77327a9e     00353000 3b0d58ed 00000000 KERNEL32!BaseThreadInitThunk+0x19
0019ffdc 77327a6e     ffffffff 77348a68 00000000
ntdll_772c0000!__RtlUserThreadStart+0x2f
0019ffec 00000000     00406715 00353000 00000000
ntdll_772c0000!_RtlUserThreadStart+0x1b


STACK_COMMAND:  ~0s ; .cxr ; kb

SYMBOL_NAME:  verifier_71920000!AVrfpDphFindBusyMemoryNoCheck+b8

MODULE_NAME: verifier_71920000

IMAGE_NAME:  verifier.dll

FAILURE_BUCKET_ID:
INVALID_POINTER_READ_AVRF_c0000005_verifier.dll!AVrfpDphFindBusyMemoryNoCheck

OS_VERSION:  10.0.19041.1

BUILDLAB_STR:  vb_release

OSPLATFORM_TYPE:  x64

OSNAME:  Windows 10

IMAGE_VERSION:  10.0.19041.1

FAILURE_ID_HASH:  {bd57151c-e59f-3c94-75ca-6923d50e6d0d}

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

Vendor Response

Documentation Windows: http://help.accusoft.com/ImageGear/v20.0/Windows/DLL/webframe.html Linux: http://help.accusoft.com/ImageGear/v20.0/Linux/webframe.html

Download Links Windows: https://download.accusoft.com/imagegear/pro/ImageGear_for_C_and_CPP_v20.0.exe Linux: https://download.accusoft.com/imagegear/pro/unix/ImageGear_for_C_Cpp20.0.0-Linux64.tar.gz

https://download.accusoft.com/imagegear/pro/ImageGear_for_C_and_CPP_v20.0.exe

Timeline

2022-01-26 - Vendor disclosure

2022-04-29 - Vendor Patched

2022-05-02 - Public Release

CREDIT

Discovered by Emmanuel Tacheau of Cisco Talos.
