Talos Vulnerability Report

TALOS-2021-1232

Accusoft ImageGear SGI Format Buffer Size Processing out-of-bounds write vulnerability

MARCH 30, 2021

CVE NUMBER

CVE-2021-21776

Summary

An out-of-bounds write vulnerability exists in the SGI Format Buffer Size Processing functionality of Accusoft ImageGear 19.8. 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 Accusoft ImageGear 19.8

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.

There is a vulnerability when ImageGear parses an SGI file.
A set of small buffers are allocated, where size for those buffers are calculated based on the `SIG_HDR.xsize` parameter, taken straight from the SGI header:

```
.text:0017A2CD                 movzx   edi, [esi+SGI_HDR.xsize]
.text:0017A2D1                 movzx   eax, ax
.text:0017A2D4                 push    ebx
.text:0017A2D5                 mov     [ebp+var_20], eax
.text:0017A2D8                 mov     [ebp+SGI_XSIZE?], edi
...
.text:0017A405                 lea     edx, ds:1[edi*2]
...
.text:0017A412 loc_17A412:                             ; CODE XREF: sub_17A2A0+1FF↓j
.text:0017A412                 push    363h            ; int
.text:0017A417                 push    offset aCommonFormatsS_1 ; "..\\..\\..\\..\\Common\\Formats\\sgirea"...
.text:0017A41C                 push    edx             ; Size
.text:0017A41D                 push    ebx             ; int
.text:0017A41E                 call    AF_memm_alloc
.text:0017A423                 mov     ecx, [ebp+var_18]
.text:0017A426                 mov     edx, [ebp+var_30]
.text:0017A429                 mov     [edx+ecx], eax  ; store the pointer
```

Where the size of this small buffer, as presented above, is calculated with the following formula:

```
SMALL_BUFFER_SIZE = 2 * SGI_XSIZE + 1;
```

So for example when the `SGI_XSIZE = 1`, the `SMALL_BUFFER_SIZE` is 3.

Later in the SGI decoding procedure, this buffer is used as a “destination memory” for a `memcpy` argument (called within the function at [1], at position 0x0001F9C1), but in this particular `memcpy` an attacker can control the size argument, as retrieved directly from the SGI file body:

```
...
.text:0017A56D                 movzx   ecx, [esi+SGI_HDR.ysize]
.text:0017A571                 imul    ecx, [ebp+var_1C]     ; loop counters
.text:0017A575                 add     ecx, [ebp+var_18]
.text:0017A578                 mov     eax, [esi+208h]       ; past hdr
.text:0017A57E                 push    dword ptr [eax+ecx*4] ; Size_Controlled (from file)
.text:0017A581                 push    dword ptr [ebx]       ; ebx - destination for memcpy (small buff)
.text:0017A583                 push    [ebp+a1]
.text:0017A586                 call    use_data              ; [1]
.text:0017A58B                 push    edi
.text:0017A58C                 push    eax
.text:0017A58D                 mov     eax, [ebp+var_34]
.text:0017A590                 push    dword ptr [eax+ebx]
.text:0017A593                 push    dword ptr [ebx]
.text:0017A595                 call    sub_179B70
.text:0017A59A                 mov     edx, [ebp+var_1C]
.text:0017A59D                 mov     [ebp+var_4], eax
.text:0017A5A0                 movzx   eax, [esi+SGI_HDR.zsize]
.text:0017A5A4                 inc     edx
.text:0017A5A5                 add     esp, 10h
.text:0017A5A8                 mov     [ebp+var_1C], edx
.text:0017A5AB                 lea     ebx, [ebx+4]
.text:0017A5AE                 cmp     edx, eax
.text:0017A5B0                 jb      short loc_17A550
.text:0017A5B2                 jmp     short loc_17A5FE
```

If we trace one execution, the following operations happen:

```
0017A423: ALLOCATED SMALL BUFFER=0x011d5200 EDI=0x00000001
0017A41C: ALLOCATING SIZE=0x00000003 EDI=0x00000001
...
0017A57E: USE DATA EAX=0x011d33e8 ECX=0x0000001d ECX*4=0x00000074 file_data=0x00000000
0017A57E: USE DATA EAX=0x011d33e8 ECX=0x00000021 ECX*4=0x00000084 file_data=0x0000000c <-- byte from file
0001F9C1: MEMCPY dest=0x011d5200 src=0x011d8fd0 size=0x0000000c caller=0x79f1a756
...
```

So in this example, the destination buffer was allocated with size=3 and due to lack of bounds checking in the `memcpy` operation (where attacker controlled the size value), this buffer gets written out-of-bounds.

```
HEAP[mine.exe]: Heap block at 011D51F8 modified at 011D5203 past requested size of 3
(4ab4.4830): Break instruction exception - code 80000003 (first chance)
eax=00ce7000 ebx=011d5203 ecx=01195924 edx=00efeda1 esi=011d51f8 edi=00000003
eip=778dd322 esp=00efef08 ebp=00efef18 iopl=0         nv up ei pl nz na po nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00000202
ntdll!RtlpCheckBusyBlockTail+0x1a6:
778dd322 cc              int     3
```

It is worth noting that ImageGear will truncate the `memcpy` size to a maximum of `0x000004e8`:

```
0017A57E: USE DATA EAX=0x00f73380 ECX=0x00000021 ECX*4=0x00000084 file_data=0x0000cccc
0x1F9C1: MEMCPY dest=0x00f75168 src=0x00f78fe8 size=0x000004e8 caller=0x79f1a6fd
```

This happens for example when the file body is filled with `41414141`:

```
0017A41C: ALLOCATING SIZE=0x00000003 EDI=0x00000001
0017A423: ALLOCATED SMALL BUFFER=0x01143760 EDI=0x00000001
0017A57E: USE DATA EAX=0x011442e8 ECX=0x00000000 ECX*4=0x00000000 file_data=0x41414141
0001F9C1: MEMCPY dest=0x01143760 src=0x01147fe8 size=0x000004e8 caller=0x79f1a6fd
```

This out-of-bounds write causes a heap corruption where size and source can be partially controlled by the attacker, possibly leading to code execution.

Crash Information

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

DEBUG_FLR_EXCEPTION_CODE(c0000374) and the ".exr -1" ExceptionCode(c0000005) don't match

KEY_VALUES_STRING: 1

    Key  : AV.Fault
    Value: Read

    Key  : Analysis.CPU.mSec
    Value: 1703

    Key  : Analysis.DebugAnalysisProvider.CPP
    Value: Create: 8007007e on IAMLEGION

    Key  : Analysis.DebugData
    Value: CreateObject

    Key  : Analysis.DebugModel
    Value: CreateObject

    Key  : Analysis.Elapsed.mSec
    Value: 68250

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

    Key  : Analysis.System
    Value: CreateObject

    Key  : Timeline.OS.Boot.DeltaSec
    Value: 151308

    Key  : Timeline.Process.Start.DeltaSec
    Value: 40

    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: 19.8.0.0


ADDITIONAL_XML: 1

OS_BUILD_LAYERS: 1

NTGLOBALFLAG:  470

APPLICATION_VERIFIER_FLAGS:  0

EXCEPTION_RECORD:  (.exr -1)
ExceptionAddress: 7784908e (ntdll!RtlpFreeHeap+0x000008ae)
   ExceptionCode: c0000005 (Access violation)
  ExceptionFlags: 00000000
NumberParameters: 2
   Parameter[0]: 00000000
   Parameter[1]: fffffff8
Attempt to read from address fffffff8

FAULTING_THREAD:  0000429c

PROCESS_NAME:  FormatConversionAndCompression_141.exe

READ_ADDRESS:  fffffff8

ERROR_CODE: (NTSTATUS) 0xc0000005 - Instrukcja w 0x%p odwo a a si  do pami ci pod adresem 0x%p. Pami   nie mo e by  %s.

EXCEPTION_CODE_STR:  c0000005

EXCEPTION_PARAMETER1:  00000000

EXCEPTION_PARAMETER2:  fffffff8

ADDITIONAL_DEBUG_TEXT:  Enable Pageheap/AutoVerifer ; Followup set based on attribute [Is_ChosenCrashFollowupThread] from Frame:[0] on
thread:[PSEUDO_THREAD]

STACK_TEXT:
00000000 00000000 heap_corruption!FormatConversionAndCompression_141.exe+0x0


SYMBOL_NAME:  heap_corruption!FormatConversionAndCompression_141.exe

MODULE_NAME: heap_corruption

IMAGE_NAME:  heap_corruption

STACK_COMMAND:  ** Pseudo Context ** ManagedPseudo ** Value: 96a56a8 ** ; kb

FAILURE_BUCKET_ID:  HEAP_CORRUPTION_c0000005_heap_corruption!FormatConversionAndCompression_141.exe

OS_VERSION:  10.0.19041.1

BUILDLAB_STR:  vb_release

OSPLATFORM_TYPE:  x86

OSNAME:  Windows 10

FAILURE_ID_HASH:  {31eb4b48-730f-79cf-6e48-242b71d3028a}
```

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

Timeline

2021-01-27 - Vendor Disclosure

2021-02-05 - Vendor Patched

2021-03-30 - Public Release

CREDIT

Discovered by Emmanuel Tacheau and a member of Cisco Talos.

VULNERABILITY REPORTS

PREVIOUS REPORT
TALOS-2021-1227

NEXT REPORT
TALOS-2021-1244