Talos Vulnerability Report

TALOS-2021-1244

Accusoft ImageGear SGI format buffer size processing out-of-bounds write vulnerability

MARCH 30, 2021

CVE NUMBER

CVE-2021-21782

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.

During parsing, a buffer is allocated:

```
.text:00179F60                 push    2C2h            ; int
.text:00179F65                 push    offset aCommonFormatsS_1 ; "..\\..\\..\\..\\Common\\Formats\\sgirea"...
.text:00179F6A                 push    edx             ; Size
.text:00179F6B                 push    [ebp+arg_4]     ; int
.text:00179F6E                 call    AF_memm_alloc
.text:00179F73                 mov     ecx, [ebp+var_4]
.text:00179F76                 mov     edx, [ebp+var_28]
.text:00179F79                 mov     [edx+ecx], eax
```

Where size parameter for memory allocation (EDX) is taken directly from the SGI header (XSIZE value).

For example when specifying SIG_HDR.XSIZE = 0:

```
0x00179FA6: (3) ALLOCATING SIZE=0x00000000 EDI=0x007aef98
0x000761C0: MALLOC IN ARG0=0x1000001b SIZE=0x00000000 RET=0x6cbe9faf
0x00179F79: (3) ALLOCATED SMALL BUFFER=0x00eb4288 EDI=0x007aef98
```

an attacker is able to force the allocated memory region size to be 0. Basically attacker controls the size of allocated memory region.

Later the attacker also controls the `size` parameter of the `memcpy` called in [1]:

```
.text:0017A0AD                 movzx   ecx, [edi+SGI_HDR.ysize]
.text:0017A0B1                 mov     eax, [edi+208h]
.text:0017A0B7                 imul    ecx, esi
.text:0017A0BA                 add     ecx, [ebp+var_14]
.text:0017A0BD                 push    dword ptr [eax+ecx*4] ; from file data
.text:0017A0C0                 mov     eax, [ebp+var_4]
.text:0017A0C3                 push    dword ptr [eax] ; Dst
.text:0017A0C5                 push    [ebp+arg_0]     ; int
.text:0017A0C8                 call    USE_DATA        ; [1] leads to memcpy
```

For example when setting file byte to 0xCC (file_data=0x000000cc):

```
0017A0BD: (3) USE DATA EAX=0x00eb3640 ECX=0x00000002 ECX*4=0x00000008 file_data=0x000000cc
0001F9C1: MEMCPY dest=0x00eb4288 src=0x00eb7fe8 size=0x000000cc caller=0x6cada756
```

However when setting file data to ‘0xdddd’ the size parameter will be 0x00000233 (max).

```
0017A0BD: (3) USE DATA EAX=0x011d3668 ECX=0x00000001 ECX*4=0x00000004 file_data=0x0000dddd
0001F9C1: MEMCPY dest=0x011d4268 src=0x011d7fe8 size=0x00000233 caller=0x6cada6fd
```

This leads to a heap memory corruption and possible code execution due to lack of bounds checking.

Crash Information

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


KEY_VALUES_STRING: 1

    Key  : Analysis.CPU.mSec
    Value: 1717

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

    Key  : Analysis.DebugData
    Value: CreateObject

    Key  : Analysis.DebugModel
    Value: CreateObject

    Key  : Analysis.Elapsed.mSec
    Value: 72849

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

    Key  : Analysis.System
    Value: CreateObject

    Key  : Timeline.OS.Boot.DeltaSec
    Value: 440783

    Key  : Timeline.Process.Start.DeltaSec
    Value: 295

    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: 778dd322 (ntdll!RtlpCheckBusyBlockTail+0x000001a6)
   ExceptionCode: 80000003 (Break instruction exception)
  ExceptionFlags: 00000000
NumberParameters: 1
   Parameter[0]: 00000000

FAULTING_THREAD:  00001054

PROCESS_NAME:  FormatConversionAndCompression_141.exe

ERROR_CODE: (NTSTATUS) 0x80000003 - {WYJ TEK}  Punkt przerwania  Osi gni to punkt przerwania.

EXCEPTION_CODE_STR:  80000003

EXCEPTION_PARAMETER1:  00000000

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: 98fe7a8 ** ; kb

FAILURE_BUCKET_ID:  HEAP_CORRUPTION_80000003_heap_corruption!FormatConversionAndCompression_141.exe

OS_VERSION:  10.0.19041.1

BUILDLAB_STR:  vb_release

OSPLATFORM_TYPE:  x86

OSNAME:  Windows 10

FAILURE_ID_HASH:  {4fd13846-2cba-9ba9-ea51-5366b3589987}

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-2021-1232

NEXT REPORT
TALOS-2021-1250