Talos Vulnerability Report

TALOS-2020-0991

Accusoft ImageGear TIFF tifread code execution vulnerability

FEBRUARY 10, 2020

CVE NUMBER

CVE-2020-6067

Summary

An exploitable out-of-bounds write vulnerability exists in the igcore19d.dll TIFF tifread parser of the Accusoft ImageGear 19.5.0 library. A specially crafted TIFF file can cause an out-of-bounds write, resulting in a remote code execution. An attacker needs to provide a malformed file to the victim to trigger the vulnerability.

Tested Versions

Accusoft ImageGear 19.5.0

Product URLs

https://www.accusoft.com/products/imagegear/overview/

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-787: Out-of-bounds Write

Details

The ImageGear library is a document imaging developer toolkit providing all kinds of functionality related to image conversion, creation, editing, annotation, etc. It supports more than 100 formats, including many image formats, DTIFFM, PDF, Microsoft Office and others.

There is a vulnerability in the TIFF raster image parser. A specially crafted TIFF file can lead to an out-of-bounds write resulting in remote code execution.

If we try to load a malformed TIFF file via the `IG_load_file` function we end up in the following situation:

```
(7814.720c): Access violation - code c0000005 (first/second chance not available)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
Time Travel Position: 18C807:0
eax=0000000a ebx=0000f52c ecx=00942006 edx=0000c936 esi=0093f3d0 edi=000aca2b
eip=5cf8a0f5 esp=0093f380 ebp=0093f398 iopl=0         nv up ei pl nz ac pe nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00000216
igCore19d!IG_mpi_page_set+0x10ed65:
5cf8a0f5 8841fa          mov     byte ptr [ecx-6],al        ds:002b:00942000=??
```

Checking the status of the stack, we can see that a stack-based buffer overflow has occurred:

```
0:000> kb 10
 # ChildEBP RetAddr  Args to Child
WARNING: Stack unwind information not available. Following frames may be wrong.
00 0093f398 5cf85222 00d44ff8 1000001f 26936d68 igCore19d!IG_mpi_page_set+0x10ed65
01 0093f8b4 5c010101 00010101 26010101 00010101 igCore19d!IG_mpi_page_set+0x109e92
02 0093f8b8 00010101 26010101 00010101 00010101 0x5c010101
03 0093f8bc 26010101 00010101 00010101 00010101 0x10101
04 0093f8c0 00010101 00010101 00010101 00010101 0x26010101
05 0093f8c4 00010101 00010101 00010101 00010101 0x10101
06 0093f8c8 00010101 00010101 00010101 00010101 0x10101
07 0093f8cc 00010101 00010101 00010101 18010101 0x10101
08 0093f8d0 00010101 00010101 18010101 00010101 0x10101
09 0093f8d4 00010101 18010101 00010101 18010101 0x10101
0a 0093f8d8 18010101 00010101 18010101 18010101 0x10101
0b 0093f8dc 00010101 18010101 18010101 00010101 0x18010101
0c 0093f8e0 18010101 18010101 00010101 5c010101 0x10101
0d 0093f8e4 18010101 00010101 5c010101 00010101 0x18010101
0e 0093f8e8 00010101 5c010101 00010101 26010101 0x18010101
0f 0093f8ec 5c010101 00010101 26010101 00010101 0x10101
```

Pseudo-code related to the vulnerable function looks as follows:

```
Line 1  v81 = *(_WORD *)v7->dword80;
Line 2  if ( v81 > 8 )
Line 3  loop_limit = 0x100;
Line 4  else
Line 5  loop_limit = 1 << v81;
Line 6  if ( loop_limit > 0 )
Line 7  {
Line 8  buffer = dstLocalBuffer + 2;
Line 9  a4 = loop_limit - 1;
Line 10 index = 0;
Line 11 do
Line 12 {
Line 13   store_value = index / a4;
Line 14   index += 255;
Line 15   buffer += 4;
Line 16   *(_BYTE *)(buffer - 6) = store_value;
Line 17   *(_BYTE *)(buffer - 5) = store_value;
Line 18   *(_BYTE *)(buffer - 4) = store_value;
Line 19   --loop_limit;
Line 20 }
Line 21 while ( loop_limit );
```

Further analysis revealed that `loop_limit` depends on `v81` [WORD], a variable which turned out to be read directly from the file at offset `0xC2` (value 0xfa10).

In our case `loop_limit` value will be equal to:

```
v81 = (BYTE)0x0000fa10
v81 = 10
0x00010000 = 1 << 0x10
0x00010000 - 1 = 0x0000ffff
loop_limit = 0x0000ffff
```

Searching for a definition of the stack buffer `buffer` we find it couple function above:

```
Line 1  int __stdcall sub_5CF85170(void *a1, void *a2, size_t a3, unsigned int a4, int a5)
Line 2  {
Line 3    size_t v5; // eax
Line 4    unsigned int v6; // ecx
Line 5    int v7; // esi
Line 6    int v8; // ecx
Line 7    char *v10; // [esp+10h] [ebp-4E8h]
Line 8    size_t v11; // [esp+14h] [ebp-4E4h]
Line 9    unsigned int v12; // [esp+24h] [ebp-4D4h]
Line 10   int v13; // [esp+2Ch] [ebp-4CCh]
Line 11   __int16 v14; // [esp+9Eh] [ebp-45Ah]
Line 12   int dstLocalBuffer; // [esp+F4h] [ebp-404h]
Line 13
Line 14   v10 = 0;
Line 15   memset(&dstLocalBuffer, 0, 0x400u);
Line 16   sub_5CE1FCB0(&v11, 0, 0xE0u);
Line 17   v14 = -1;
Line 18   v13 = sub_5CE69A50(a1);
Line 19   v5 = AF_memm_unique_tag_get("..\\..\\..\\..\\Common\\Formats\\tifread.c", 760);
Line 20   v6 = a4;
Line 21   v7 = v5;
Line 22   if ( a4 < 1 )
Line 23         v6 = 1;
Line 24   v12 = v6 - 1;
Line 25   if ( !vulnFunction(a1, v5, a2, (int)&v11, (int)&dstLocalBuffer, a3, a5)
```

The allocated spack on the stack for this buffer equals `0x400` `line 15`.

As we can see an attacker controls all presented variables just by proper file content manipulation.

Increasing the loop count via the `v81` variable an attacker can cause an out-of-bounds write leading to memory corruption which can result in remote code execution.

Crash Information

```
(7814.720c): Access violation - code c0000005 (first/second chance not available)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
Time Travel Position: 18C807:0
eax=0000000a ebx=0000f52c ecx=00942006 edx=0000c936 esi=0093f3d0 edi=000aca2b
eip=5cf8a0f5 esp=0093f380 ebp=0093f398 iopl=0         nv up ei pl nz ac pe nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00000216
igCore19d!IG_mpi_page_set+0x10ed65:
5cf8a0f5 8841fa          mov     byte ptr [ecx-6],al        ds:002b:00942000=??

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


KEY_VALUES_STRING: 1

        Key  : AV.Fault
        Value: Read

        Key  : Analysis.CPU.Sec
        Value: 3

        Key  : Analysis.DebugAnalysisProvider.CPP
        Value: Create: 8007007e on DESKTOP-E4N8506

        Key  : Analysis.DebugData
        Value: CreateObject

        Key  : Analysis.DebugModel
        Value: CreateObject

        Key  : Analysis.Elapsed.Sec
        Value: 5

        Key  : Analysis.Memory.CommitPeak.Mb
        Value: 434

        Key  : Analysis.System
        Value: CreateObject

        Key  : Timeline.OS.Boot.DeltaSec
        Value: 532938


ADDITIONAL_XML: 1

APPLICATION_VERIFIER_LOADED: 1

EXCEPTION_RECORD:  (.exr -1)
ExceptionAddress: 5cf8a0f5 (igCore19d!IG_mpi_page_set+0x0010ed65)
   ExceptionCode: c0000005 (Access violation)
  ExceptionFlags: 00000000
NumberParameters: 2
   Parameter[0]: 00000000
   Parameter[1]: 00942000
Attempt to read from address 00942000

FAULTING_THREAD:  0000720c

PROCESS_NAME:  igFuzzer.exe

READ_ADDRESS:  00942000

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:  00942000

STACK_TEXT:
WARNING: Stack unwind information not available. Following frames may be wrong.
0093f398 5cf85222 00d44ff8 1000001f 26936d68 igCore19d!IG_mpi_page_set+0x10ed65
0093f8b4 5c010101 00010101 26010101 00010101 igCore19d!IG_mpi_page_set+0x109e92
0093f8b8 00010101 26010101 00010101 00010101 0x5c010101
0093f8bc 26010101 00010101 00010101 00010101 0x10101
0093f8c0 00010101 00010101 00010101 00010101 0x26010101
0093f8c4 00010101 00010101 00010101 00010101 0x10101
0093f8c8 00010101 00010101 00010101 00010101 0x10101
0093f8cc 00010101 00010101 00010101 18010101 0x10101
0093f8d0 00010101 00010101 18010101 00010101 0x10101
0093f8d4 00010101 18010101 00010101 18010101 0x10101
0093f8d8 18010101 00010101 18010101 18010101 0x10101
0093f8dc 00010101 18010101 18010101 00010101 0x18010101
0093f8e0 18010101 18010101 00010101 5c010101 0x10101
0093f8e4 18010101 00010101 5c010101 00010101 0x18010101
0093f8e8 00010101 5c010101 00010101 26010101 0x18010101
(...)
0093fc30 cc010101 cc010101 cc010101 cc010101 0xcc010101

STACK_COMMAND:  ~0s ; .cxr ; kb

SYMBOL_NAME:  igCore19d!IG_mpi_page_set+10ed65

MODULE_NAME: igCore19d

IMAGE_NAME:  igCore19d.dll

FAILURE_BUCKET_ID:  INVALID_POINTER_READ_AVRF_c0000005_igCore19d.dll!IG_mpi_page_set

OSPLATFORM_TYPE:  x86

OSNAME:  Windows 8

FAILURE_ID_HASH:  {bfd6b5ab-5824-8327-06e6-1c2f38a120f0}

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

```
0:000> lmva eip
Browse full module list
start    end        module name
5b8a0000 5bbe9000   igCore19d   (export symbols)       d:\projects\ImageGear\current\Build\Bin\x86\igCore19d.dll
        Loaded symbol image file: d:\projects\ImageGear\current\Build\Bin\x86\igCore19d.dll
        Image path: d:\projects\ImageGear\current\Build\Bin\x86\igCore19d.dll
        Image name: igCore19d.dll
        Browse all global symbols  functions  data
        Timestamp:        Fri Nov 22 15:45:29 2019 (5DD7F489)
        CheckSum:         00356062
        ImageSize:        00349000
        File version:     19.5.0.0
        Product version:  19.5.0.0
        File flags:       0 (Mask 3F)
        File OS:          4 Unknown Win32
        File type:        2.0 Dll
        File date:        00000000.00000000
        Translations:     0409.04b0
        Information from resource tables:
                CompanyName:      Accusoft Corporation
                ProductName:      Accusoft ImageGear
                InternalName:     igcore19d.dll
                OriginalFilename: igcore19d.dll
                ProductVersion:   19.5.0.0
                FileVersion:      19.5.0.0
                FileDescription:  Accusoft ImageGear CORE DLL
                LegalCopyright:   Copyright 1996-2019 Accusoft Corporation. All rights reserved.
                LegalTrademarks:  ImageGearÆ and AccusoftÆ are registered trademarks of Accusoft Corporation
```

Timeline

2020-01-27 - Vendor Disclosure

2020-02-10 - Public Release

CREDIT

Discovered by Emmanuel Tacheau and a member of Cisco Talos.

VULNERABILITY REPORTS

PREVIOUS REPORT TALOS-2020-0990

NEXT REPORT TALOS-2020-0993

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