Talos Vulnerability Report

TALOS-2021-1286

Accusoft ImageGear PDF process_fontname stack-based buffer overflow vulnerability

JUNE 1, 2021

CVE NUMBER

CVE-2021-21821

Summary

A stack-based buffer overflow vulnerability exists in the PDF process_fontname functionality of Accusoft ImageGear 19.9. A specially crafted malformed file can lead to code execution. 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-121 - Stack-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.

There is a vulnerability in the `process_fontname` function which occurs with a specially crafted PDF file, leading to a stack-based buffer overflow which can result in code execution. Trying to load a malformed PDF file, we end up in the following situation:

```
(264ec.24598): Security check failure or stack buffer overrun - code c0000409 (!!! second chance !!!)
Subcode: 0x2 FAST_FAIL_STACK_COOKIE_CHECK_FAILURE
eax=00000001 ebx=0eeeeca8 ecx=00000002 edx=000001e9 esi=0f6e8e78 edi=00000000
eip=79045a4f esp=0019f834 ebp=0019fb58 iopl=0         nv up ei pl nz na po nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00000202
igPDF19d!CPb_PDF_init+0x1110cf:
79045a4f cd29            int     29h
```

Stack inspection shows us the stack buffer overflow as follows:

```
0:000> kb
 # ChildEBP RetAddr      Args to Child
WARNING: Stack unwind information not available. Following frames may be wrong.
00 0019fb58 78fc990f     00000010 0eeeeca8 0eeeeca8 igPDF19d!CPb_PDF_init+0x1110cf
01 0019fc84 33694132     41346941 69413569 37694136 igPDF19d!CPb_PDF_init+0x94f8f
02 0019fc88 41346941     69413569 37694136 41386941 0x33694132
03 0019fc8c 69413569     37694136 41386941 6a413969 0x41346941
04 0019fc90 37694136     41386941 6a413969 316a4130 0x69413569
05 0019fc94 41386941     6a413969 316a4130 41326a41 0x37694136
06 0019fc98 6a413969     316a4130 41326a41 6a41336a 0x41386941
07 0019fc9c 316a4130     41326a41 6a41336a 356a4134 0x6a413969
08 0019fca0 41326a41     6a41336a 356a4134 41366a41 0x316a4130
09 0019fca4 6a41336a     356a4134 41366a41 6a41376a 0x41326a41
0a 0019fca8 356a4134     41366a41 6a41376a 396a4138 0x6a41336a
0b 0019fcac 41366a41     6a41376a 396a4138 41306b41 0x356a4134
0c 0019fcb0 6a41376a     396a4138 41306b41 6b41316b 0x41366a41
0d 0019fcb4 396a4138     41306b41 6b41316b 336b4132 0x6a41376a
0e 0019fcb8 41306b41     6b41316b 336b4132 41346b41 0x396a4138
0f 0019fcbc 6b41316b     336b4132 41346b41 6b41356b 0x41306b41
```

In our case the stack overwrite is happening in function `process_fontname` at LINE68, when performing an `sprintf` call while using some statically allocated variable `b_ovf` as destination buffer, without checking the size of the destination. This is the pseudo-code of the vulnerable function:

```
LINE1   AT_ERRCOUNT process_fontname(int param_1)
LINE2   {
LINE3     int iVar1;
LINE4     uint *puVar2;
LINE5     LPVOID pvVar3;
LINE6     int iVar4;
LINE7     int *tls_index;
LINE8     uint uVar5;
LINE9     undefined4 very_long_string;
LINE10    dword status;
LINE11    int **in_FS_OFFSET;
LINE12    undefined8 uVar6;
LINE13    uint uStack312;
LINE14    int local_120;
LINE15    uint local_11c;
LINE16    char b_ovf [256];
LINE17    uint local_18;
LINE18    uint *local_14;
LINE19    int *local_10;
LINE20    undefined *puStack12;
LINE21    undefined4 local_8;
LINE22
LINE23    local_8 = 0xffffffff;
LINE24    puStack12 = &LAB_10190f2b;
LINE25    local_10 = *in_FS_OFFSET;
LINE26    uStack312 = DAT_10223fb8 ^ (uint)&stack0xfffffffc;
LINE27    local_14 = &uStack312;
LINE28    *in_FS_OFFSET = (int *)&local_10;
LINE29    local_18 = uStack312;
LINE30    puVar2 = &uStack312;
LINE31    if (param_1 != 0) {
LINE32      very_long_string = *(undefined4 *)(param_1 + 0x110);
LINE33      pvVar3 = call_tlsgetvalue();
LINE34      uVar6 = (**(code **)(*(int *)((int)pvVar3 + 0x20) + 0x278))(very_long_string);
LINE35      pvVar3 = call_tlsgetvalue();
LINE36      iVar4 = (**(code **)(*(int *)((int)pvVar3 + 0x18) + 8))(uVar6);
LINE37      puVar2 = local_14;
LINE38      if (iVar4 != 0) {
LINE39        pvVar3 = call_tlsgetvalue();
LINE40        iVar4 = *(int *)((int)pvVar3 + 0x18);
LINE41        tls_index = (int *)call_tlsgetvalue();
LINE42        very_long_string = (**(code **)(*tls_index + 0x14))(&DAT_101e211c);
LINE43        uVar6 = (**(code **)(iVar4 + 0x54))(uVar6,very_long_string);
LINE44        pvVar3 = call_tlsgetvalue();
LINE45        iVar4 = (**(code **)(*(int *)((int)pvVar3 + 0x18) + 8))(uVar6);
LINE46        puVar2 = local_14;
LINE47        if (iVar4 != 0) {
LINE48          local_11c = 0;
LINE49          local_8 = 0;
LINE50          tls_index = (int *)call_tlsgetvalue();
LINE51          (**(code **)(*tls_index + 8))(0,FUN_10001ef0);
LINE52          local_8._0_1_ = 1;
LINE53          pvVar3 = call_tlsgetvalue();
LINE54          uVar5 = (**(code **)(*(int *)((int)pvVar3 + 0x18) + 0x10))
LINE55                            (uVar6,FUN_100e9920,*(undefined4 *)(local_120 + 0x354));
LINE56          local_8 = (uint)local_8._1_3_ << 8;
LINE57          local_11c = uVar5 & 0xffff;
LINE58          tls_index = (int *)call_tlsgetvalue();
LINE59          (**(code **)(*tls_index + 0xc))();
LINE60          local_8 = 0xffffffff;
LINE61          puVar2 = local_14;
LINE62          if ((short)(uVar5 & 0xffff) == 0) {
LINE63            iVar4 = **(int **)(local_120 + 0x354);
LINE64            if ((iVar4 != 0) && (iVar1 = (*(int **)(local_120 + 0x354))[1], iVar1 != 0)) {
LINE65              tls_index = (int *)call_tlsgetvalue();
LINE66              very_long_string =
LINE67                   (**(code **)(*tls_index + 0x1c))(*(undefined4 *)(iVar4 + -4 + iVar1 * 4));
LINE68              _sprintf(b_ovf,
LINE69                       "Cannot find or create the font \'%s\'. Some characters may not display orprint correctly."
LINE70                       ,very_long_string);
LINE71              AF_err_record_set("..\\Common\\Components\\Pdf\\Adobe\\PDFDocument.cpp",0x678,-0x159b,1,
LINE72                                0,0,b_ovf);
LINE73              puVar2 = local_14;
LINE74            }
LINE75          }
LINE76        }
LINE77      }
LINE78    }
LINE79    local_14 = puVar2;
LINE80    *in_FS_OFFSET = local_10;
LINE81    status = raise_security_failure(local_18 ^ (uint)&stack0xfffffffc);
LINE82    return status;
LINE83  }
```

The variable `b_ovf` is a char buffer with a fixed size of 256 bytes, as you can see at LINE16. So, if the attacker is able to provide a very long string into the variable `very_long_string` (larger than 256 bytes), then the overflow will happen.

The contents for `very_long_string` are read directly from the file and correspond to the `fontname` used in an object of the PDF file. A prerequisite to trigger this vulnerability is that the `FontDescriptor` flag must be present in the object.

Crash Information

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


KEY_VALUES_STRING: 1

    Key  : Analysis.CPU.mSec
    Value: 3171

    Key  : Analysis.DebugAnalysisManager
    Value: Create

    Key  : Analysis.Elapsed.mSec
    Value: 13894

    Key  : Analysis.Init.CPU.mSec
    Value: 1452

    Key  : Analysis.Init.Elapsed.mSec
    Value: 1535073

    Key  : Analysis.Memory.CommitPeak.Mb
    Value: 161

    Key  : FailFast.Name
    Value: STACK_COOKIE_CHECK_FAILURE

    Key  : FailFast.Type
    Value: 2

    Key  : Timeline.OS.Boot.DeltaSec
    Value: 540483

    Key  : Timeline.Process.Start.DeltaSec
    Value: 1534

    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


NTGLOBALFLAG:  2100000

APPLICATION_VERIFIER_FLAGS:  0

APPLICATION_VERIFIER_LOADED: 1

EXCEPTION_RECORD:  (.exr -1)
ExceptionAddress: 79045a4f (igPDF19d!CPb_PDF_init+0x001110cf)
   ExceptionCode: c0000409 (Security check failure or stack buffer overrun)
  ExceptionFlags: 00000001
NumberParameters: 1
   Parameter[0]: 00000002
Subcode: 0x2 FAST_FAIL_STACK_COOKIE_CHECK_FAILURE

FAULTING_THREAD:  00024598

PROCESS_NAME:  FuzzIgPDF.exe

ERROR_CODE: (NTSTATUS) 0xc0000409 - The system detected an overrun of a stack-based buffer in this application. This overrun could
potentially allow a malicious user to gain control of this application.

EXCEPTION_CODE_STR:  c0000409

EXCEPTION_PARAMETER1:  00000002

STACK_TEXT:
WARNING: Stack unwind information not available. Following frames may be wrong.
0019fb58 78fc990f     00000010 0eeeeca8 0eeeeca8 igPDF19d!CPb_PDF_init+0x1110cf
0019fc84 33694132     41346941 69413569 37694136 igPDF19d!CPb_PDF_init+0x94f8f
0019fc88 41346941     69413569 37694136 41386941 0x33694132
0019fc8c 69413569     37694136 41386941 6a413969 0x41346941
0019fc90 37694136     41386941 6a413969 316a4130 0x69413569
0019fc94 41386941     6a413969 316a4130 41326a41 0x37694136
0019fc98 6a413969     316a4130 41326a41 6a41336a 0x41386941
0019fc9c 316a4130     41326a41 6a41336a 356a4134 0x6a413969
0019fca0 41326a41     6a41336a 356a4134 41366a41 0x316a4130
0019fca4 6a41336a     356a4134 41366a41 6a41376a 0x41326a41
0019fca8 356a4134     41366a41 6a41376a 396a4138 0x6a41336a
0019fcac 41366a41     6a41376a 396a4138 41306b41 0x356a4134
0019fcb0 6a41376a     396a4138 41306b41 6b41316b 0x41366a41
0019fcb4 396a4138     41306b41 6b41316b 336b4132 0x6a41376a
0019fcb8 41306b41     6b41316b 336b4132 41346b41 0x396a4138
0019fcbc 6b41316b     336b4132 41346b41 6b41356b 0x41306b41
0019fcc0 336b4132     41346b41 6b41356b 376b4136 0x6b41316b
0019fcc4 41346b41     6b41356b 376b4136 41386b41 0x336b4132
0019fcc8 6b41356b     376b4136 41386b41 6c41396b 0x41346b41
0019fccc 376b4136     41386b41 6c41396b 316c4130 0x6b41356b
0019fcd0 41386b41     6c41396b 316c4130 41326c41 0x376b4136
0019fcd4 6c41396b     316c4130 41326c41 6c41336c 0x41386b41
0019fcd8 316c4130     41326c41 6c41336c 356c4134 0x6c41396b
0019fdd8 0049c9c6     00000000 00000000 0ece2fc0 0x316c4130
00000000 00000000     00000000 00000000 00000000 FuzzIgPDF!main+0x346


STACK_COMMAND:  ~0s ; .cxr ; kb

SYMBOL_NAME:  igPDF19d!CPb_PDF_init+1110cf

MODULE_NAME: igPDF19d

IMAGE_NAME:  igPDF19d.dll

FAILURE_BUCKET_ID:  FAIL_FAST_STACK_BUFFER_OVERRUN_STACK_COOKIE_CHECK_FAILURE_MISSING_GSFRAME_AVRF_c0000409_igPDF19d.dll!CPb_PDF_init
```

```
OS_VERSION:  10.0.19041.1

BUILDLAB_STR:  vb_release

OSPLATFORM_TYPE:  x86

OSNAME:  Windows 10

IMAGE_VERSION:  25.0.0.2347

FAILURE_ID_HASH:  {6abf7d27-0ab6-8f44-d6a8-e38cd092ee79}

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

Timeline

2021-04-28 - Vendor Disclosure
2021-05-31 - Vendor Patched
2021-06-01 - Public Release

CREDIT

Discovered by Emmanuel Tacheau of Cisco Talos.

VULNERABILITY REPORTS

PREVIOUS REPORT TALOS-2021-1289

NEXT REPORT TALOS-2021-1275