Talos Vulnerability Report

TALOS-2020-1070

Nitro Pro Indexed ColorSpace Rendering Code Execution Vulnerability

SEPTEMBER 15, 2020

CVE NUMBER

CVE-2020-6116

Summary

An arbitrary code execution vulnerability exists in the rendering functionality of Nitro Software, Inc.'s Nitro Pro 13.13.2.242. When drawing the contents of a page using colors from an indexed colorspace, the application can miscalculate the size of a buffer when allocating space for its colors. When using this allocated buffer, the application can write outside its bounds and cause memory corruption which can lead to code execution. A specially crafted document must be loaded by a victim in order to trigger this vulnerability.

Tested Versions

Nitro Pro 13.13.2.242
Nitro Pro 13.16.2.300

Product URLs

https://www.gonitro.com/nps/product-details/downloads

CVSSv3 Score

8.8 - CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:H

CWE

CWE-680 - Integer Overflow to Buffer Overflow

Details

Nitro Software, Inc. includes their flagship product, Nitro Pro as part of their Nitro Productivity Suite. Nitro Pro is Nitro Software's PDF editor and flagship product. This product allows users to create and modify documents that follow the Portable Document Format (PDF) specification and other digital documents.

Nitro Software Inc. develops commercial software used to create, edit, sign, and secure Portable Document Format files and digital documents. This is supported by their Nitro Pro application as part of their Nitro Productivity Suite. The Nitro Pro application allows users to read, modify, and create documents that follow the Portable Document Format standard. When creating a page for a document, the creator is allowed to specify the colorspace to use when drawing the page's different components. One of the available colorspaces is the "Indexed" colorspace which allows the creator to include an indexed color palette in the document in order to use for coloring the different parts of a page. When the application renders the page, it will allocate space for the indexed color palette and load colors into the allocated space. Due to an integer overflow, the application can miscalculate the size of the indexed palette resulting in an undersized buffer. Later when loading colors into this buffer, a buffer overflow will occur.

When the application is rendering a page, it must interpret a number of commands in order to build the contents of the page. When a colorspace is chosen by the contents of the page, the following function is executed. This function will first identify what type of colorspace was selected by the creator of the page. This is done by first checking the colorspace name agained the "/ICCBased" atom. First, the function will load the 64-bit number representing the "ICCBased" atom. If this is uninitialized, the application will pass the string to the `ASAtomFromString` function at [1]. Afterwards at [2], the application will pass a string to the "ICCBased" atom and then compare it against the atom representing the selected ColorSpace. As this advisory involves the "/Indexed" colorspace, the application will continue on and check the next colorspace type. Next, the application will load the 64-bit integer representing the "Indexed" atom. If this has not been initialized yet, at [3] the application will pass the "Indexed" string to the `ASAtomFromString` function in order to convert it to an atom.

```
npdf!PDBookmarkGetCosObj+0xbd74:
5af13314 8b0dc8604f5b    mov     ecx,dword ptr [npdf!CAPContent::`vftable'+0x1399d8 (5b4f60c8)]     ; "ICCBased" atom
5af1331a 8b15cc604f5b    mov     edx,dword ptr [npdf!CAPContent::`vftable'+0x1399dc (5b4f60cc)]     ; "ICCBased" atom
5af13320 8945fc          mov     dword ptr [ebp-4],eax
5af13323 8bc1            mov     eax,ecx
5af13325 23c2            and     eax,edx
5af13327 c68573ffffff01  mov     byte ptr [ebp-8Dh],1
5af1332e c645fc01        mov     byte ptr [ebp-4],1
5af13332 83f8ff          cmp     eax,0FFFFFFFFh
5af13335 751c            jne     npdf!PDBookmarkGetCosObj+0xbdb3 (5af13353)
...
5af13337 ff35c0604f5b    push    dword ptr [npdf!CAPContent::`vftable'+0x1399d0 (5b4f60c0)]         ; "ICCBased" string
5af1333d e8de31f3ff      call    npdf!ASAtomFromString (5ae46520)                                   ; [1] ASAtomFromString
5af13342 8bc8            mov     ecx,eax
5af13344 8915cc604f5b    mov     dword ptr [npdf!CAPContent::`vftable'+0x1399dc (5b4f60cc)],edx     ; Store atom
5af1334a 83c404          add     esp,4
5af1334d 890dc8604f5b    mov     dword ptr [npdf!CAPContent::`vftable'+0x1399d8 (5b4f60c8)],ecx     ; Store atom
...
5af13353 8b7e1c          mov     edi,dword ptr [esi+1Ch]                                            ; Selected ColorSpace atom
5af13356 394e18          cmp     dword ptr [esi+18h],ecx
5af13359 0f851f020000    jne     npdf!PDBookmarkGetCosObj+0xbfde (5af1357e)                         ; [2] Branch to check of ColorSpace type
5af1335f 3bfa            cmp     edi,edx
5af13361 0f8517020000    jne     npdf!PDBookmarkGetCosObj+0xbfde (5af1357e)                         ; [2] Branch to check of ColorSpace type
...
npdf!PDBookmarkGetCosObj+0xbfde:
5af1357e a1d8604f5b      mov     eax,dword ptr [npdf!CAPContent::`vftable'+0x1399e8 (5b4f60d8)]     ; "Indexed" atom
5af13583 8b15dc604f5b    mov     edx,dword ptr [npdf!CAPContent::`vftable'+0x1399ec (5b4f60dc)]     ; "Indexed" atom
5af13589 898574ffffff    mov     dword ptr [ebp-8Ch],eax
5af1358f 23c2            and     eax,edx
5af13591 83f8ff          cmp     eax,0FFFFFFFFh
5af13594 751e            jne     npdf!PDBookmarkGetCosObj+0xc014 (5af135b4)
...
5af13596 ff35d0604f5b    push    dword ptr [npdf!CAPContent::`vftable'+0x1399e0 (5b4f60d0)]         ; "Indexed" string
5af1359c e87f2ff3ff      call    npdf!ASAtomFromString (5ae46520)                                   ; [3] ASAtomFromString
5af135a1 a3d8604f5b      mov     dword ptr [npdf!CAPContent::`vftable'+0x1399e8 (5b4f60d8)],eax
5af135a6 83c404          add     esp,4
5af135a9 8915dc604f5b    mov     dword ptr [npdf!CAPContent::`vftable'+0x1399ec (5b4f60dc)],edx
5af135af 8b7e1c          mov     edi,dword ptr [esi+1Ch]                                            ; load high 32-bits of selected
colorspace
5af135b2 eb06            jmp     npdf!PDBookmarkGetCosObj+0xc01a (5af135ba)
```

After loading the 64-bit integer for the “/Indexed” atom, the application will execute the following code. This code will first load the selected colorspace into the `%ecx` register. This is done so that the selected atom can be compared against the “/Indexed” atom at [4]. After confirming the colorspace is of the “/Indexed” type, the application will then call `CosObjGetType` to verify that the array for the colorspace is defined. At [6], the `CosArrayGet` function is used to get the third element of the array. According to the PDF file format specification, the third element is named `hival` and represents the maximum indexed of the described colorspace. After fetching the maximum index of the colorspace, at [7] the application will convert it to an integer.

```
npdf!PDBookmarkGetCosObj+0xc01a:
5af135ba 8b4e18          mov     ecx,dword ptr [esi+18h]                         ; selected ColorSpace type
5af135bd 3bc8            cmp     ecx,eax
5af135bf 0f85ba040000    jne     npdf!PDBookmarkGetCosObj+0xc4df (5af13a7f)     ; [4] Compare againsed "/Indexed" atom
5af135c5 3bfa            cmp     edi,edx
5af135c7 0f85b2040000    jne     npdf!PDBookmarkGetCosObj+0xc4df (5af13a7f)     ; [4] Compare againsed "/Indexed" atom
...
5af135cd ff7604          push    dword ptr [esi+4]                               ; Array for ColorSpace
5af135d0 e8abe3f5ff      call    npdf!CosObjGetType (5ae71980)                   ; [5] CosObjGetType
5af135d5 83c404          add     esp,4
5af135d8 84c0            test    al,al
5af135da 7418            je      npdf!PDBookmarkGetCosObj+0xc054 (5af135f4)
...
5af135dc 6a02            push    2
5af135de ff7604          push    dword ptr [esi+4]                               ; Array for ColorSpace
5af135e1 e86a0ff6ff      call    npdf!CosArrayGet (5ae74550)                     ; [6] use CosArrayGet to get third element of array (hival)
5af135e6 83c408          add     esp,8
...
5af135e9 50              push    eax
5af135ea e87153f6ff      call    npdf!CosIntegerValue (5ae78960)                 ; [7] convert hival to integer
5af135ef 83c404          add     esp,4
5af135f2 eb03            jmp     npdf!PDBookmarkGetCosObj+0xc057 (5af135f7)
```

After converting `hival` to an integer, at [8] the application will add 1 to it and then store it to a variable on the stack at [9]. After adjusting the index, the application will then multiply it by 3 at [10]. In order to ensure that at least 1 element is allocated, at [11] the application will add 3 and then check to see if the addition will overflow. Afterwards at [12], the application will pass the resulting calculation to `malloc`. Although the application checks to see if the addition of 3 will result in an overflow, this is not sufficient as if the integer for `hival` when multiplied by 3 at [10] overflows the overflow flag will not be set. As a result of the overflow, the allocation at [12] will be undersized. After allocating memory, at [13] this buffer will be stored on the stack and inside an object as a property.

```
npdf!PDBookmarkGetCosObj+0xc057:
5af135f7 40              inc     eax                                                   ; [8] add 1 to the integer (hival)
5af135f8 898554ffffff    mov     dword ptr [ebp-0ACh],eax                              ; [9] Store for termination of a loop later
5af135fe 8d0c40          lea     ecx,[eax+eax*2]                                       ; [10] Multiply hival by 3
5af13601 33c0            xor     eax,eax
5af13603 83c103          add     ecx,3                                                 ; [11] Add 3 to hival
5af13606 0f92c0          setb    al                                                    ; [11] Check if addition resulted in overflow
5af13609 f7d8            neg     eax
5af1360b 0bc1            or      eax,ecx
5af1360d 50              push    eax
5af1360e ff1580f6385b    call    dword ptr [npdf!CAPContent::Wrap+0x29de90 (5b38f680)]  ; [12] malloc
5af13614 83c404          add     esp,4
5af13617 8bf8            mov     edi,eax                                               ; [13] Store allocated buffer
5af13619 8b8554ffffff    mov     eax,dword ptr [ebp-0ACh]
5af1361f 894610          mov     dword ptr [esi+10h],eax                               ; [13] Store buffer
5af13622 8b4624          mov     eax,dword ptr [esi+24h]
```

Later after allocating a buffer for the indexed palette, the application will enter the following loop. This loop has a terminator at [14] which checks to see if the current palette index in `%eax` is larger than the integer (`hival`) that was returned from `CosArrayGet`. After loading indexed colorspace data into a local variable, at [15] the application will load each 8-bit component from the current palette into a register, and then at [16] write it into the pointer in the `%edi` register. Each iteration of this loop will then increment the pointer in the `%edi` register by 3 in order to seek to the next index of the palette. As prior mentioned, this loop will continue to write into the undersized buffer using the `%edi` register until the current indexed color in `%eax` reached the length of the loop (`hival`) that was read via `CosArrayGet`. Due to the integer-overflow resulting in an undersized buffer being allocated, the length for the loop and the length for the buffer are out-of-sync. This loop will write outside the bounds of the buffer which will cause a heap-based buffer overflow. This can lead to code execution under the context of the application.

```
npdf!PDBookmarkGetCosObj+0xc425:
5af139c5 f20f100da845495b movsd   xmm1,mmword ptr [npdf!CAPContent::`vftable'+0xd7eb8 (5b4945a8)]
5af139cd 898574ffffff    mov     dword ptr [ebp-8Ch],eax
5af139d3 3b8554ffffff    cmp     eax,dword ptr [ebp-0ACh]                          ; [14] Check against length
5af139d9 0f8d7d000000    jge     npdf!PDBookmarkGetCosObj+0xc4bc (5af13a5c)
5af139df 33c9            xor     ecx,ecx
5af139e1 394a20          cmp     dword ptr [edx+20h],ecx
5af139e4 7e37            jle     npdf!PDBookmarkGetCosObj+0xc47d (5af13a1d)
...
5af13a1d 8d45c0          lea     eax,[ebp-40h]
5af13a20 50              push    eax
5af13a21 8d8578ffffff    lea     eax,[ebp-88h]
5af13a27 50              push    eax
5af13a28 e8f3461800      call    npdf!PDEDefaultGState+0x2c0 (5b098120)
...
5af13a2d 8a45c0          mov     al,byte ptr [ebp-40h]                             ; [15] Load byte from dword read by inner loop
5af13a30 83c408          add     esp,8
5af13a33 8807            mov     byte ptr [edi],al                                 ; [16] Store byte to undersized buffer
5af13a35 8b45c0          mov     eax,dword ptr [ebp-40h]                           ; [15] Load byte from dword read by inner loop
5af13a38 c1e808          shr     eax,8
5af13a3b 884701          mov     byte ptr [edi+1],al                               ; [16] Store byte to undersized buffer
5af13a3e 8b45c0          mov     eax,dword ptr [ebp-40h]                           ; [15] Load byte from dword read by inner loop
5af13a41 c1e810          shr     eax,10h
5af13a44 884702          mov     byte ptr [edi+2],al                               ; [16] Store byte to undersized buffer
5af13a47 83c703          add     edi,3                                             ; [17] Add 3 to buffer
5af13a4a 8b8574ffffff    mov     eax,dword ptr [ebp-8Ch]
5af13a50 8b9578ffffff    mov     edx,dword ptr [ebp-88h]
5af13a56 40              inc     eax                                               ; [17] Increment loop counter
5af13a57 e969ffffff      jmp     npdf!PDBookmarkGetCosObj+0xc425 (5af139c5)
```

Crash Information

When opening up the provided proof-of-concept in the application, the following crash will occur.

```
(2bbc.1bdc): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
eax=0000006b ebx=0118f001 ecx=25c446e3 edx=006b0c0a esi=0c0b0cc8 edi=0c1d0ffe
eip=5af13a44 esp=0118eb4c ebp=0118ec0c iopl=0         nv up ei pl nz na po nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00210202
npdf!PDBookmarkGetCosObj+0xc4a4:
5af13a44 884702          mov     byte ptr [edi+2],al        ds:0023:0c1d1000=??
```

Outputting the loop terminator shows that the loop will continue until the value of `hival` is reached.

```
0:000> ? dwo(@ebp-ac)
Evaluate expression: 1431655766 = 55555556
```

Performing the same math that was used for the allocation shows that only 5 bytes were allocated for the buffer.

```
0:000> ? dwo(@ebp-ac)*3+3
Evaluate expression: 4294967301 = 00000001`00000005
```

The base addresses of the libraries in this report.

```
0:000> lm m npdf
Browse full module list
start    end        module name
5adc0000 5b807000   npdf       (export symbols)       npdf.dll
013e0000 01c61000   NitroPDF   (deferred)
```

Exploit Proof of Concept

In the provided proof-of-concept, the “/Indexed” colorspace described by this advisory is stored in object 6. This colorspace uses the integer 1431655765 (0x55555555) for `hival` which will result in the allocation for the buffer being of length 5. The commands for rendering the page are in object 7. These will load the specified colorspace by name (`/IndexedColorSpace`), and then use a color to draw a rectangle that fills the first page. The usage of the colorspace whilst rendering the page is necessary for this vulnerability to trigger. The dimensions for the rectangle directly correspond to the MediaBox dimensions which are stored in object 5.

Timeline

2020-05-13 - Vendor Disclosure

2020-09-01 - Vendor Patched

2020-09-15 - Public Release