Talos Vulnerability Report

TALOS-2020-1084

Nitro Pro PDF ICCBased ColorSpace Stroke Color Code Execution Vulnerability

SEPTEMBER 15, 2020

CVE NUMBER

CVE-2020-6146

SUMMARY

An exploitable code execution vulnerability exists in the rendering functionality of Nitro Pro 13.13.2.242 and 13.16.2.300. When drawing the contents of a page and selecting the stroke color from an “ICCBased” colorspace, the application will read a length from the file and use it as a loop sentinel when writing data into the member of an object. Due to the object member being a buffer of a static size allocated on the heap, this can result in a heap-based buffer overflow. A specially crafted document must be loaded by a victim in order to trigger this vulnerability.

CONFIRMED VULNERABLE VERSIONS

The versions below were either tested or verified to be vulnerable by Talos or confirmed to be vulnerable by the vendor.

Nitro Pro 13.13.2.242
Nitro Pro 13.16.2.300

PRODUCT URLS

Nitro Pro - 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-122 - Heap-based 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.

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 colorspaces that are available is the “ICCBased” colorspace which allows the creator to describe a color based on an ICC color profile. When the application executes the set stroke color operation, it will parse the attributes of the object stream containing the color profile for the colorspace. One of the attributes of this stream will be used to as a length for a loop which is used to write to an array belonging to an object. This size of this buffer is a static size of `0x248` bytes and is suspected by the author to begin with some properties and end with the aforementioned array. Due to the application trusting the length from the file when writing to this array, an attacker may specify a length larger than the total size of the buffer so that when the application writes to it, a heap-based buffer overflow will occur.

When first parsing the file, the application will construct a `CNxPageMap` object using the following code. The application will allocate `0x170` bytes for the object at [1], initialize it using `memset` at [2], and then call its constructor at [3]. The constructor of the `CNxPageMap` object will initialize a number of members belonging to the object. At [4], the application will initialize an object that inherits from the `CNxContentParser` class. One of the fields of this class at `+0x58` of its structure is a pointer that will contain that color information that will be read from when overflowing the buffer described within this advisory. At [5], the application will initialize the pointer that the buffer will be allocated at with `NULL`.

```
npdf!PDOCMDsMakeContentVisible+0x1334:
5a4d65b4 6870010000      push    170h
5a4d65b9 e869803100      call    npdf!CAPContent::Wrap+0x27ce37 (5a7ee627)      ; [1] malloc
5a4d65be 8bf8            mov     edi,eax
5a4d65c0 83c404          add     esp,4
5a4d65c3 89bddcfdffff    mov     dword ptr [ebp-224h],edi
...
npdf!PDOCMDsMakeContentVisible+0x1354:
5a4d65d4 6870010000      push    170h
5a4d65d9 6a00            push    0
5a4d65db 57              push    edi
5a4d65dc e88fb03100      call    npdf!CAPContent::Wrap+0x27fe80 (5a7f1670)      ; [2] memset
5a4d65e1 ff36            push    dword ptr [esi]
5a4d65e3 e818b3e1ff      call    npdf!CosObjGetDoc (5a2f1900)
5a4d65e8 50              push    eax                                            ; CNxCosDoc
5a4d65e9 e8f24be4ff      call    npdf!PDDocFromCosDoc (5a31b1e0)
5a4d65ee 83c414          add     esp,14h
5a4d65f1 8bcf            mov     ecx,edi
5a4d65f3 6a00            push    0
5a4d65f5 50              push    eax                                            ; PDDoc
5a4d65f6 e82531e9ff      call    npdf!PDEToUnicodeTableSave+0x1f090 (5a369720)  ; [3] \
5a4d65fb eb02            jmp     npdf!PDOCMDsMakeContentVisible+0x137f (5a4d65ff)
\
npdf!PDEToUnicodeTableSave+0x1f090:
5a369720 55              push    ebp
5a369721 8bec            mov     ebp,esp
...
5a369744 8bf1            mov     esi,ecx
5a369746 8975f0          mov     dword ptr [ebp-10h],esi
5a369749 6a00            push    0
5a36974b e880550000      call    npdf!PDTextIsSpaceBetween+0xc90 (5a36ecd0)     ; [4] construct CNxPDEContentParser
...
npdf!PDTextIsSpaceBetween+0xdcf:
5a36ee0f 8b4508          mov     eax,dword ptr [ebp+8]
5a36ee12 898788000000    mov     dword ptr [edi+88h],eax
5a36ee18 8bc7            mov     eax,edi
5a36ee1a c787a400000000000000 mov dword ptr [edi+0A4h],0                        ; [5] initialize pointer containing buffer
```

After constructing the `CNxPageMap` member of the object, the application will return in order to complete construction of the `CNxPDEContentParser`. Immediately after the construction of the object, the application will call a method that will initialize some more members of the object. This is done by the following code. At [6], the application will execute the `malloc` function with a static size of `0x248`. It will then be initialized at [7], and after completion the resulting allocation will be written to the member at offset `0xa4(%esi)` of the object. This static allocation is the buffer described by this document that can be overflown.

```
npdf!PDTextIsSpaceBetween+0x1300:
5a36f340 55              push    ebp
5a36f341 8bec            mov     ebp,esp
...
npdf!PDTextIsSpaceBetween+0x1335:
5a36f375 6848020000      push    248h
5a36f37a 8ad8            mov     bl,al
5a36f37c e8a6f24700      call    npdf!CAPContent::Wrap+0x27ce37 (5a7ee627)      ; [6] malloc
5a36f381 8bf8            mov     edi,eax
5a36f383 83c404          add     esp,4
5a36f386 897d10          mov     dword ptr [ebp+10h],edi
...
npdf!PDTextIsSpaceBetween+0x1354:
5a36f394 6848020000      push    248h
5a36f399 6a00            push    0
5a36f39b 57              push    edi
5a36f39c e8cf224800      call    npdf!CAPContent::Wrap+0x27fe80 (5a7f1670)      ; [7] memset
...
npdf!PDTextIsSpaceBetween+0x1376:
5a36f3b6 8d8ea4000000    lea     ecx,[esi+0A4h]                                 ; load address of member
5a36f3bc 8901            mov     dword ptr [ecx],eax                            ; [8] write allocation to member
5a36f3be 85c0            test    eax,eax
5a36f3c0 7533            jne     npdf!PDTextIsSpaceBetween+0x13b5 (5a36f3f5)
```

After completing the construction of the object, the application will begin to interpret the contents of the file in order to render the document to the user. One of the object streams that will need to be interpreted in order to render the contents of a page contains a stream of commands or tokens that will describe the contents of the page and how to draw it. When setting the color for a stroke that is to be drawn, the application will execute the following member belonging to the `CNxPageMap` object that was constructed. This method will start by reading a type from its second parameter at [9], and use it to calculate a pointer from the statically allocated buffer to the array that is contain therein. Aftr calculating the pointer, the application will execute its method at [10]. This method will fetch the type of the colorspace, convert it to an atom, and then store it to its first parameter. The vulnerability described by this advisory is specifically when handling the `/ICCBased` colorspace type. As a result, the atom for the `/ICCBased` name will be written to the first parameter passed to the method. After fetching the atom, the method for the object at [11] will be called. This method is responsible for interpreting the attributes of the object stream and returning the number of components for the colorspace by writing the integer to its first parameter.

```
0:000> u npdf+134790 L12
npdf!PDTextIsSpaceBetween+0x6750:
5a374790 55              push    ebp
5a374791 8bec            mov     ebp,esp
...
5a3747be 8bf1            mov     esi,ecx
...
npdf!PDTextIsSpaceBetween+0x6780:
5a3747c0 8b8ea4000000    mov     ecx,dword ptr [esi+0A4h]                       ; array from static allocation
5a3747c6 8a5d08          mov     bl,byte ptr [ebp+8]                            ; [9] type
5a3747c9 8975c4          mov     dword ptr [ebp-3Ch],esi
5a3747cc 8b7940          mov     edi,dword ptr [ecx+40h]                        ; point %edi at member within allocation
5a3747cf 84db            test    bl,bl
5a3747d1 7406            je      npdf!PDTextIsSpaceBetween+0x6799 (5a3747d9)
5a3747d3 8bb990000000    mov     edi,dword ptr [ecx+90h]                        ; point %edi at member within allocation
...
npdf!PDTextIsSpaceBetween+0x67b3:
5a3747f3 8d45dc          lea     eax,[ebp-24h]                                  ; atom of colorspace
5a3747f6 8bcf            mov     ecx,edi
5a3747f8 50              push    eax
5a3747f9 e8727d1900      call    npdf!PDEClipRemoveElems+0x1b00 (5a50c570)      ; [10] get atom in the first element of colorspace array
5a3747fe 84c0            test    al,al
5a374800 0f84c6030000    je      npdf!PDTextIsSpaceBetween+0x6b8c (5a374bcc)
...
npdf!PDTextIsSpaceBetween+0x67c6:
5a374806 8d45d8          lea     eax,[ebp-28h]                                  ; local variable to write number of components to
5a374809 c745d800000000  mov     dword ptr [ebp-28h],0
5a374810 32ff            xor     bh,bh
5a374812 8bcf            mov     ecx,edi
5a374814 50              push    eax                                            ; address of local variable passed as parameter
5a374815 887dcf          mov     byte ptr [ebp-31h],bh
5a374818 e8837e1900      call    npdf!PDEClipRemoveElems+0x1c30 (5a50c6a0)      ; [11] write number of components to first parameter
5a37481d 84c0            test    al,al
5a37481f 0f84a7030000    je      npdf!PDTextIsSpaceBetween+0x6b8c (5a374bcc)
```

In order to get the number of components for the colorspace, the application will need to re-read the attributes for the object scheme. Within the Portable Document Format specification, a number of different colorspace have a static number of color components. When the colorspace type is of `/ICCBased`, however, the number of components will need to be dynamically calculated. The following method will perform just that by first fetching the atom describing the colorspace at [12]. As prior mentioned and with the provided proof-of-concept, this call will write the atom for `/ICCBased` to its first parameter which will be stored on the stack. At [13], the application will then call the method that is actually responsible for determining the number of components.

```
npdf!PDEClipRemoveElems+0x1c30:
5a50c6a0 55              push    ebp
5a50c6a1 8bec            mov     ebp,esp
5a50c6a3 83ec08          sub     esp,8
5a50c6a6 56              push    esi
5a50c6a7 8bf1            mov     esi,ecx
...
npdf!PDEClipRemoveElems+0x1c4f:
5a50c6bf 57              push    edi
5a50c6c0 8d45f8          lea     eax,[ebp-8]                                    ; atom of colorspace
5a50c6c3 8bce            mov     ecx,esi
5a50c6c5 50              push    eax
5a50c6c6 e8a5feffff      call    npdf!PDEClipRemoveElems+0x1b00 (5a50c570)      ; [12] get atom in the first element of colorspace array
...
npdf!PDEClipRemoveElems+0x1c5b:
5a50c6cb 8b7d08          mov     edi,dword ptr [ebp+8]                          ; result
5a50c6ce 57              push    edi
5a50c6cf ff7620          push    dword ptr [esi+20h]                            ; CosArray for colorspace
5a50c6d2 ff75fc          push    dword ptr [ebp-4]                              ; 64-bit atom
5a50c6d5 ff75f8          push    dword ptr [ebp-8]                              ; 64-bit atom
5a50c6d8 e823000000      call    npdf!PDEClipRemoveElems+0x1c90 (5a50c700)      ; [13] get number of components for colorspace type
5a50c6dd 83c410          add     esp,10h
5a50c6e0 84c0            test    al,al
5a50c6e2 7508            jne     npdf!PDEClipRemoveElems+0x1c7c (5a50c6ec)
```

When determining the number of components for the colorspace type, the application must process a `PDFArray` from the document. This array was passed as the second parameter and will be parsed by this method. First, the method will use the 64-bit atom that was passed as the first parameter to determine which colorspace to use when determining the color to change the stroke to. Eventually at [14], the application will check to see if the colorspace type corresponds to the `/ICCBased` atom. After confirming this, the application will fetch the first element of its array at [15] which should result in a `PDFDictionary(6)` type. After fetching the `PDFDictionary(6)`, the application will convert the string “N” into an atom at [16], and then use it as a parameter to the `CosDictGet` function at [17]. The `/N` key of the `ICCBased` atom contains the number of color components for the described colorspace. This attribute is later used directly as a counter for a loop which is directly responsible for the buffer overflow described by this document. After fetching the number of components from the `/N` key, the application will then pass the resulting object to the `CosIntegerValue` function at [18]. This will convert the token parsed from the file to an integer and then write it directly into the result parameter of the method.

```
npdf!PDEClipRemoveElems+0x1c90:
5a50c700 55              push    ebp
5a50c701 8bec            mov     ebp,esp
...
5a50c729 8b5d14          mov     ebx,dword ptr [ebp+14h]                                       ; result parameter to write number of
components to
5a50c72c c745fc00000000  mov     dword ptr [ebp-4],0
5a50c733 c645fc01        mov     byte ptr [ebp-4],1
5a50c737 c70300000000    mov     dword ptr [ebx],0                                             ; initialize number of components with 0
...
npdf!PDEClipRemoveElems+0x1e65:
5a50c8d5 b9c060975a      mov     ecx,offset npdf!CAPContent::`vftable'+0x1399d0 (5a9760c0)      ; "ICCBased"
5a50c8da e82138deff      call    npdf!local_file_handle::write+0x1000 (5a2f0100)                ; [14] GetAtomFromString
5a50c8df 3bf8            cmp     edi,eax
5a50c8e1 0f850a010000    jne     npdf!PDEClipRemoveElems+0x1f81 (5a50c9f1)
5a50c8e7 3bf2            cmp     esi,edx
5a50c8e9 0f8502010000    jne     npdf!PDEClipRemoveElems+0x1f81 (5a50c9f1)
...
npdf!PDEClipRemoveElems+0x1e7f:
5a50c8ef 6a01            push    1
5a50c8f1 ff7510          push    dword ptr [ebp+10h]                                           ; CosArray
5a50c8f4 e8577cdeff      call    npdf!CosArrayGet (5a2f4550)                                   ; [15] CosArrayGet
5a50c8f9 83c408          add     esp,8
5a50c8fc 8bf0            mov     esi,eax
...
npdf!PDEClipRemoveElems+0x1e8e:
5a50c8fe b9a061975a      mov     ecx,offset npdf!CAPContent::`vftable'+0x139ab0 (5a9761a0)      ; "N"
5a50c903 897514          mov     dword ptr [ebp+14h],esi
5a50c906 e8f537deff      call    npdf!local_file_handle::write+0x1000 (5a2f0100)                ; [16] GetAtomFromString
...
npdf!PDEClipRemoveElems+0x1e9b:
5a50c90b 52              push    edx                                                           ; 64-bit atom for `/N`
5a50c90c 50              push    eax                                                           ; 64-bit atom for `/N`
5a50c90d 56              push    esi                                                           ; PDFDictionary returned from array
5a50c90e e8bdaddeff      call    npdf!CosDictGet (5a2f76d0)                                    ; [17] get `/N` attribute from colorspace
5a50c913 83c40c          add     esp,0Ch
...
npdf!PDEClipRemoveElems+0x1ea6:
5a50c916 50              push    eax
5a50c917 e844c0deff      call    npdf!CosIntegerValue (5a2f8960)                               ; [18] convert to integer
5a50c91c 83c404          add     esp,4
5a50c91f 8903            mov     dword ptr [ebx],eax                                           ; store result to parameter
```

After getting the number of components from the `/N` key of the colorspace dictionary, the application will check the number of components against some static values. After determining that the number of components is not of a standard value, the application will then check to see if an alternate colorspace (or a fall-back) was provided. This is done by checking for the `/Alternate` key in the `PDFDictionary(6)` for the object stream. First at [19], the method will convert the string "Alternate" into an atom. This atom will then be passed to the `CosDictKnown` function at [20]. If the `/Alternate` key was not found in the dictionary, the application will raise an exception. After confirming that the `/Alternate` key exists, the application will again convert the "Alternate" string into an atom at [21], and then pass it to the `CosDictGet` function at [22]. Immediately afterwards, the application will check the type of the object returned from the `/Alternate` key. If it is of any other type than a `PDFName(4)`, the method will then return leaving the number of components written to the methods parameter. If the type is a `PDFName(4)`, then the method will recurse into itself in order to determine the number of components using the selected colorspace. It is prudent to note that the attacker must specify a type for the `/Alternate` key that is not a `PDFName(4)` in order to directly control the loop sentinel required by this vulnerability.

```
npdf!PDEClipRemoveElems+0x1eb1:
5a50c921 83f801          cmp     eax,1
5a50c924 0f84df000000    je      npdf!PDEClipRemoveElems+0x1f99 (5a50ca09)
...
npdf!PDEClipRemoveElems+0x1eba:
5a50c92a 83f803          cmp     eax,3
5a50c92d 0f84d6000000    je      npdf!PDEClipRemoveElems+0x1f99 (5a50ca09)
...
npdf!PDEClipRemoveElems+0x1ec3:
5a50c933 83f804          cmp     eax,4
5a50c936 0f84cd000000    je      npdf!PDEClipRemoveElems+0x1f99 (5a50ca09)
...
npdf!PDEClipRemoveElems+0x1ecc:
5a50c93c b94062975a      mov     ecx,offset npdf!CAPContent::`vftable'+0x139b50 (5a976240)          ; "Alternate"
5a50c941 e8ba37deff      call    npdf!local_file_handle::write+0x1000 (5a2f0100)                    ; [19] GetAtomFromString
5a50c946 52              push    edx                                                               ; 64-bit atom for `/Alternate`
5a50c947 50              push    eax                                                               ; 64-bit atom for `/Alternate`
5a50c948 56              push    esi                                                               ; PDFDictionary returned from array
5a50c949 e8c2addeff      call    npdf!CosDictKnown (5a2f7710)                                      ; [20] check existence of key
5a50c94e 83c40c          add     esp,0Ch
5a50c951 84c0            test    al,al
5a50c953 0f8474010000    je      npdf!PDEClipRemoveElems+0x205d (5a50cacd)
...
npdf!PDEClipRemoveElems+0x1ee9:
5a50c959 b94062975a      mov     ecx,offset npdf!CAPContent::`vftable'+0x139b50 (5a976240)          ; "Alternate"
5a50c95e e89d37deff      call    npdf!local_file_handle::write+0x1000 (5a2f0100)                    ; [21] GetAtomFromString
...
npdf!PDEClipRemoveElems+0x1ef3:
5a50c963 52              push    edx                                                               ; 64-bit atom for `/Alternate`
5a50c964 50              push    eax                                                               ; 64-bit atom for `/Alternate`
5a50c965 56              push    esi                                                               ; PDFDictionary returned from array
5a50c966 e865addeff      call    npdf!CosDictGet (5a2f76d0)                                        ; [22] get element from array
5a50c96b 83c40c          add     esp,0Ch
5a50c96e 8bf0            mov     esi,eax
...
npdf!PDEClipRemoveElems+0x1f00:
5a50c970 56              push    esi                                                               ; object returned from `/Alternate` key
5a50c971 e80a50deff      call    npdf!CosObjGetType (5a2f1980)                                     ; [23] get object type
5a50c976 83c404          add     esp,4
...
npdf!PDEClipRemoveElems+0x1f09:
5a50c979 3c04            cmp     al,4                                                              ; [24] PDFName
5a50c97b 0f8588000000    jne     npdf!PDEClipRemoveElems+0x1f99 (5a50ca09)
```

Upon returning to the method which called it, the application will now be able to use the number of components in order to select the stroke color. Eventually throughout the function, the application will check some booleans and then at [25] will load the number of components that was read from the `/N` key into the `%edi` register. The `%edi` register is used as a loop terminator in the loop that follows and is directly responsible for overflowing the buffer that was allocated with `0x248` bytes. At [26], the statically allocated buffer will be loaded into the `%eax` register and then the index of the loop in `%edi` will be used to determine an offset into the buffer. Afterwards at [27], the type from the first parameter of the method will be used to distinguish the size of an object that the method will need to shift past in order to get to the correct position of the buffer. This will then be passed to the method call at [28]. As this loop will iterate the number of times as specified by the `/N` key which represents the number of components, the number of iterations of this loop can be directly controlled. At [29], the loop will iterate to the next element.

```
npdf!PDTextIsSpaceBetween+0x69d4:
5a374a14 8a7dcf          mov     bh,byte ptr [ebp-31h]                           ; boolean determining if DeviceN was the selected colorspace
5a374a17 8b7dc8          mov     edi,dword ptr [ebp-38h]
5a374a1a 8b75c4          mov     esi,dword ptr [ebp-3Ch]                         ; this
5a374a1d 8a5d08          mov     bl,byte ptr [ebp+8]                             ; type from first parameter
...
npdf!PDTextIsSpaceBetween+0x6a08:
5a374a48 84ff            test    bh,bh
5a374a4a 7545            jne     npdf!PDTextIsSpaceBetween+0x6a51 (5a374a91)
5a374a4c 8b7dd8          mov     edi,dword ptr [ebp-28h]                         ; [25] number of components from colorspace
5a374a4f 85ff            test    edi,edi
5a374a51 0f8eee000000    jle     npdf!PDTextIsSpaceBetween+0x6b05 (5a374b45)
5a374a57 83c7ff          add     edi,0FFFFFFFFh
5a374a5a 0f88e5000000    js      npdf!PDTextIsSpaceBetween+0x6b05 (5a374b45)
...
npdf!PDTextIsSpaceBetween+0x6a20:
5a374a60 8b86a4000000    mov     eax,dword ptr [esi+0A4h]                        ; [26] pointer to allocation of a static size
5a374a66 8bce            mov     ecx,esi
5a374a68 8d04f8          lea     eax,[eax+edi*8]                                 ; seek into buffer using %edi as index
5a374a6b 84db            test    bl,bl
5a374a6d 7407            je      npdf!PDTextIsSpaceBetween+0x6a36 (5a374a76)    ; determine the type
5a374a6f 0598000000      add     eax,98h                                         ; [27] adjust pointer into buffer
5a374a74 eb03            jmp     npdf!PDTextIsSpaceBetween+0x6a39 (5a374a79)
5a374a76 83c048          add     eax,48h                                         ; [27] adjust pointer into buffer
5a374a79 50              push    eax
5a374a7a e881240000      call    npdf!PDTextIsSpaceBetween+0x8ec0 (5a376f00)    ; [28] call method which will write into its parameter
5a374a7f 84c0            test    al,al
5a374a81 0f8445010000    je      npdf!PDTextIsSpaceBetween+0x6b8c (5a374bcc)
5a374a87 83ef01          sub     edi,1                                           ; [29] decrement index in order to loop %edi times
5a374a8a 79d4            jns     npdf!PDTextIsSpaceBetween+0x6a20 (5a374a60)
```

The method call that is executed for every iteration of the loop controlled by the user-supplied `/N` key is described by the following code. This method will first load a value from one of the object's members that used as a counter. The method will then use the count as an index in order to read data from the file at [30]. After loading 64-bits from the file, the method will load the pointer to the statically sized buffer into the `%eax` register. After validating that the pointer to write to is non-zero, at [31] the method will write the 64-bits that were previously read directly into it. As the buffer that is being written to is of a static size, and the loop is controlled by the user using the `/N` attribute, the pointer can eventually point out-of-bounds of the statically sized heap buffer and the store instruction (`movsd`) will write 64-bits past the bounds of the heap buffer. This is a heap-based buffer overflow, and can lead to code execution under the context of the application.

```
npdf!PDTextIsSpaceBetween+0x8ec0:
5a376f00 55              push    ebp
5a376f01 8bec            mov     ebp,esp
5a376f03 8b415c          mov     eax,dword ptr [ecx+5Ch]                             ; count
5a376f06 85c0            test    eax,eax
5a376f08 7e30            jle     npdf!PDTextIsSpaceBetween+0x8efa (5a376f3a)
5a376f0a 48              dec     eax
5a376f0b 89415c          mov     dword ptr [ecx+5Ch],eax                             ; count
5a376f0e c1e005          shl     eax,5
5a376f11 034158          add     eax,dword ptr [ecx+58h]                             ; [30] data from file
5a376f14 803804          cmp     byte ptr [eax],4
5a376f17 7427            je      npdf!PDTextIsSpaceBetween+0x8f00 (5a376f40)
...
npdf!PDTextIsSpaceBetween+0x8f00:
5a376f40 f20f104010      movsd   xmm0,mmword ptr [eax+10h]                           ; [30] read 64-bits
5a376f45 8b4508          mov     eax,dword ptr [ebp+8]                               ; [31] load pointer into static buffer from first
parameter
5a376f48 85c0            test    eax,eax
5a376f4a 7404            je      npdf!PDTextIsSpaceBetween+0x8f10 (5a376f50)
5a376f4c f20f1100        movsd   mmword ptr [eax],xmm0                               ; [31] write 64-bits directly into buffer
```

The addresses for the libraries described within this advisory are as follows.

```
Browse full module list
start    end        module name
5a240000 5ac87000   npdf       (export symbols)       npdf.dll
00390000 00c11000   NitroPDF   (deferred)
```

Crash Information

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

```
(11e4.780): Access violation - code c0000005 (first/second chance not available)
*** ERROR: Symbol file could not be found.  Defaulted to export symbols for npdf.dll -
eax=1764e040 ebx=17390000 ecx=2d089e58 edx=00000000 esi=2d089e58 edi=00000248
eip=5a376f4c esp=01c5f668 ebp=01c5f668 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!PDTextIsSpaceBetween+0x8f0c:
5a376f4c f20f1100        movsd   mmword ptr [eax],xmm0 ds:0023:1764e040=????????????????
```

The count that is used to guard against the write is currently at 2.

```
0:000> ? dwo(@ecx+5c)
Evaluate expression: 2 = 00000002
```

The bounds of the buffer that was allocated are as follows.

```
0:000> ? dwo(@ecx+a4)
Evaluate expression: 392482232 = 1764cdb8

0:000> dq poi(@ecx+a4) L(248/8)+1
1764cdb8  00000000`00000000 3ff00000`00000000
1764cdc8  00000000`00000000 00000000`00000000
1764cdd8  3ff00000`00000000 00000000`00000000
...
1764cfd8  00000000`00000000 00000000`00000000
1764cfe8  3ff00000`00000000 00000000`00000000
1764cff8  00000000`00000000 ????????`????????
```

The current count or sentinel for the loop that is controlled by the user shows that it is larger than the buffer size.

```
0:000> ? @edi
Evaluate expression: 584 = 00000248
```

As the loop index is multiplied by 8, with the provided proof-of-concept the application has written the following number of bytes into the buffer.

```
0:000> ? @eax-poi(@ecx+a4)
Evaluate expression: 4744 = 00001288
```

Exploit Proof of Concept

In the proof-of-concept, there a number of components that are relevant to this vulnerability. One of which is that whenever describing an `ICCBased` color profile, the `/Alternate` key is required. As using a `PDFName` will result in the aforementioned method recursing and thus writing a different number of components, this field must not be `PDFName`. There also must be a color profile attached to the stream. After each of these prerequisites have been accomplished, the last aspect that is required is that the color profile must be used in the operator stream for a page's contents. Once switching to the colorspace and then selecting a color for a stroke, the path described within this document will be triggered.

TIMELINE

2020-05-20 - Vendor Disclosure
2020-09-01 - Vendor Patched
2020-09-15 - Public Release

CREDIT

feliz cumpleaños para mi. ;)