The code crash offset below of the dxgknrl.sys driver when trying to access a non paged pool pointer:

dxgkrnl!SubmitPresentHistoryToken+0x758:
93df39c8 f00fc14604 lock xadd dword ptr [esi+4],eax ds:0023:bc023f2c=????????

0: kd> !pool bc023f2c
Pool page bc023f2c region is Special pool

VdPN 13 1736 722 43560 Video display mode management , Binary: dxgkrnl.sys

Enabling the pool tracking in the dxgknrl.sys driver (verifier.exe /flags 0x8 /driver dxgkrnl.sys), we can check when this pointer is allocated and when is freed:

2: kd> !verifier 0x80 0x87f6be38

Log of recent kernel pool Allocate and Free operations:

There are up to 0x10000 entries in the log.

Parsing 0x00010000 log entries, searching for address 0x87f6be38.

==========
Pool block 87f6be30, Size 000000e0, Thread 94b52600
81ea0020 nt!ExFreePoolWithTag+0x10
9896fc07 dxgmms2!VidSchiPropagatePresentHistoryToken+0xeb31
989a8ae1 dxgmms2!VidSchSubmitCommand+0x3c1
9300384f dxgkrnl!SubmitPresentHistoryToken+0x5df
93069996 dxgkrnl!DxgkPresent+0x89586
81d7d5db nt!KiSystemServicePostCall+0
==========
Pool block 87f6be38, Size 000000d4, Thread 94b52600
8224bac3 nt!VerifierExAllocatePoolWithTag+0x69
92f664aa dxgkrnl!operator new+0x1c
93004023 dxgkrnl!ReadPresentPrivateDriverData+0x93
9300396b dxgkrnl!SubmitPresentHistoryToken+0x6fb
93069996 dxgkrnl!DxgkPresent+0x89586
81d7d5db nt!KiSystemServicePostCall+0

As we can see, the pointer is allocated in ***dxgkrnl!ReadPresentPrivateDriverData*** function, and freed later inside the dxgmms2.sys driver.

Setting a breakpoint in **dxgkrnl!ReadPresentPrivateDriverData+0x93** we can see how the allocation happened. It is interesting that we can control the size of the allocation through the parameter ***PrivateDriverDataSize*** of the ***D3DKMT_PRESENT*** structure:

ba e1 dxgkrnl!ReadPresentPrivateDriverData+0x93
3: kd> ba e1 dxgkrnl!SubmitPresentHistoryToken+0x758
3: kd> dd eax
dda5df28 a1a1a1a1 a1a1a1a1 a1a1a1a1 a1a1a1a1
dda5df38 a1a1a1a1 a1a1a1a1 a1a1a1a1 a1a1a1a1
dda5df48 a1a1a1a1 a1a1a1a1 a1a1a1a1 a1a1a1a1
dda5df58 a1a1a1a1 a1a1a1a1 a1a1a1a1 a1a1a1a1
dda5df68 a1a1a1a1 a1a1a1a1 a1a1a1a1 a1a1a1a1
dda5df78 a1a1a1a1 a1a1a1a1 a1a1a1a1 a1a1a1a1
dda5df88 a1a1a1a1 a1a1a1a1 a1a1a1a1 a1a1a1a1
dda5df98 a1a1a1a1 a1a1a1a1 a1a1a1a1 a1a1a1a1
3: kd> r ebx
ebx=dda5bd60
3: kd> r eax
eax=dda5df28

Once the pointer is allocated, the ***PrivateDriverData*** array of the ***D3DKMT_PRESENT*** structure is copied to the non paged pool allocated chunk:

dxgkrnl!SubmitPresentHistoryToken+0x711:
96253981 e9a8fdffff   jmp   dxgkrnl!SubmitPresentHistoryToken+0x4be (9625372e)
3: kd> dd eax
dda5df28 000000cc 00000001 41414141 42424242   // 0xCC=PrivateDriverDataSize
dda5df38 41414141 41414141 41414141 43434343
dda5df48 41414141 41414141 41414141 44444444
dda5df58 41414141 41414141 00000000 00000000
dda5df68 00000000 00000000 00000000 00000000
dda5df78 00000000 00000000 00000000 00000000
dda5df88 00000000 00000000 00000000 00000000
dda5df98 00000000 00000000 00000000 00000000

In the offset **dxgkrnl!SubmitPresentHistoryToken+0x5dd** the driver dxgmms2 is called trhough the function dxgmms2!VidSchSubmitCommand and a LIST_ENTRY structure which contains a copy of the previous allocated pool pointer is passed as an argument of the function call.

Inside the dxgmms2.sys driver, the function call **dxgmms2!VidSchiPropagatePresentHistoryToken+0xeb24** is called and is in this offset when a call to ExFreeHeapPool is executed to free the PrivateDriverData pool pointer previously allocated in **dxgknrl!ReadPresentPrivateDriverData.**

nt!ExFreePoolWithTag:

```
81f23010 8bff          mov     edi,edi
81f23012 55            push    ebp
81f23013 8bec          mov     ebp,esp
81f23015 83e4f8        and     esp,0FFFFFFF8h
81f23018 8b4d08        mov     ecx,dword ptr [ebp+8] ss:0010:cf30d6bc=dda5df28
81f2301b e810bbdfff    call    nt!ExFreeHeapPool (81d1eb30)
```

After the free, the code return to dxgknrl.sys, but the copy of the pointer passed in the LIST_ENTRY structure to the dxgmms2.sys driver is not checked in order to invalidate the original pointer, so the pointer to the freed non-paged pool chunk is later used in ***dxgkrnl!SubmitPresentHistoryToken+0x758*** causing a BSOD when the special pool is enabled for the dxgknrl.sys driver.

Although the time between the freed and the use is short, it is possible to reclaim the freed chunk playing with threads, one thread for pool spraying and other thread for triggering repeatedly the vulnerability, opening good opportunities to exploit the vulnerability.