

CSS ⁺ 互联网安全领袖峰会
Cyber Security Summit

Sorry, It is Not Your Page

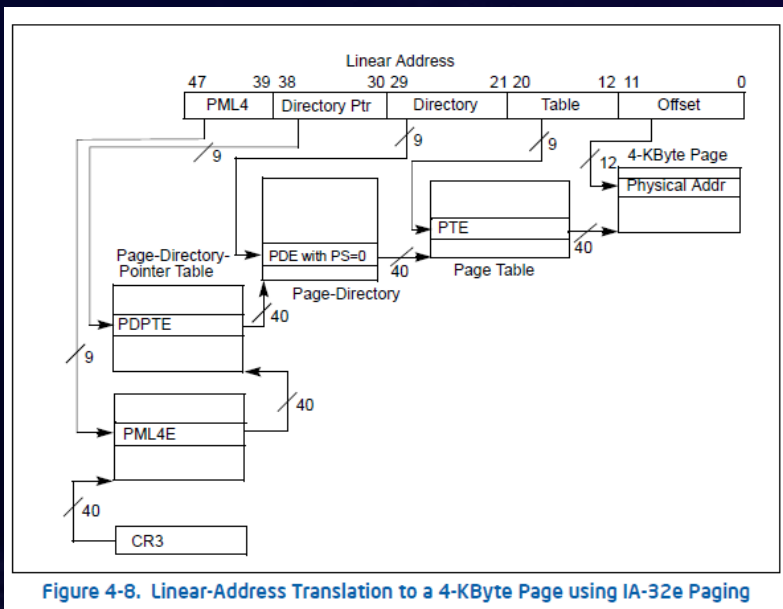
张云海 绿盟科技 天机实验室负责人

Sorry, It is Not Your Page

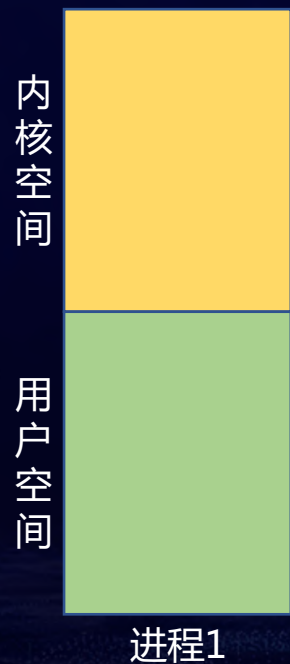
演讲者 张云海

Windows 内存管理

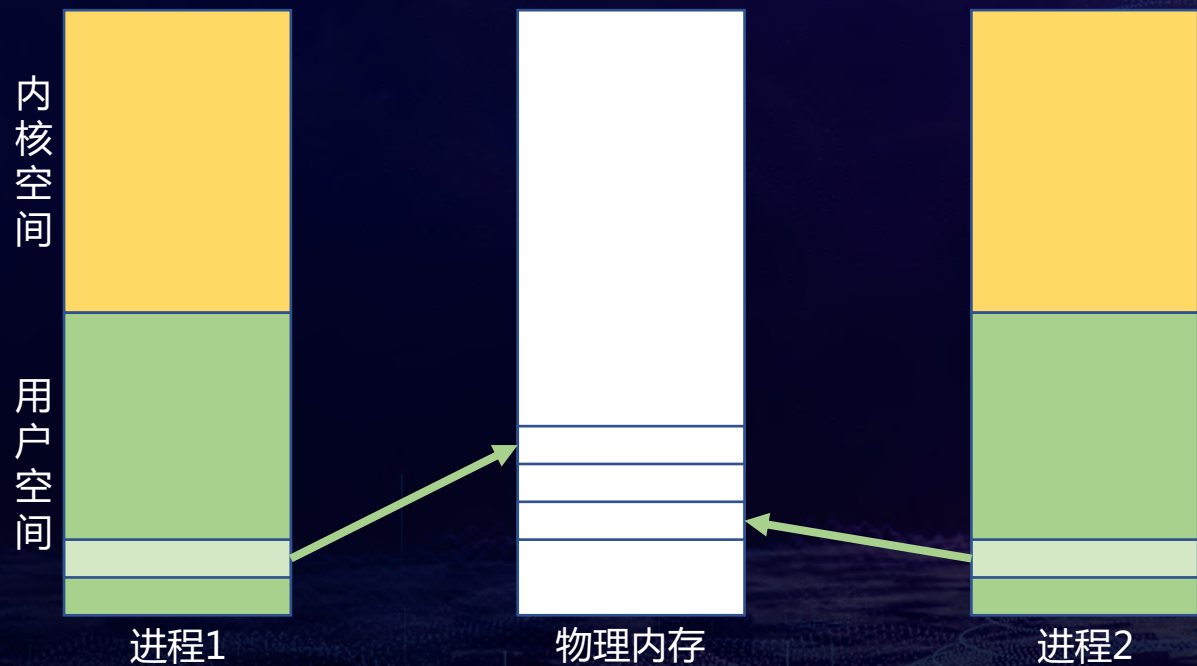
Windows 采用分页式内存管理



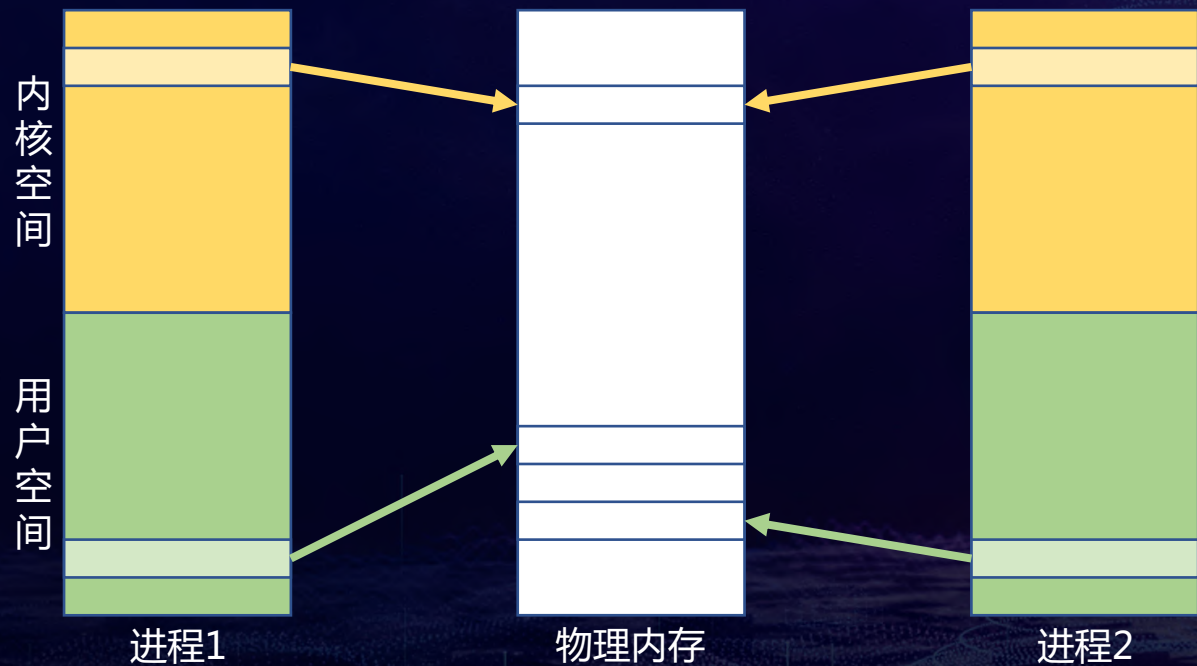
进程的虚拟地址空间分为用户空间和内核空间



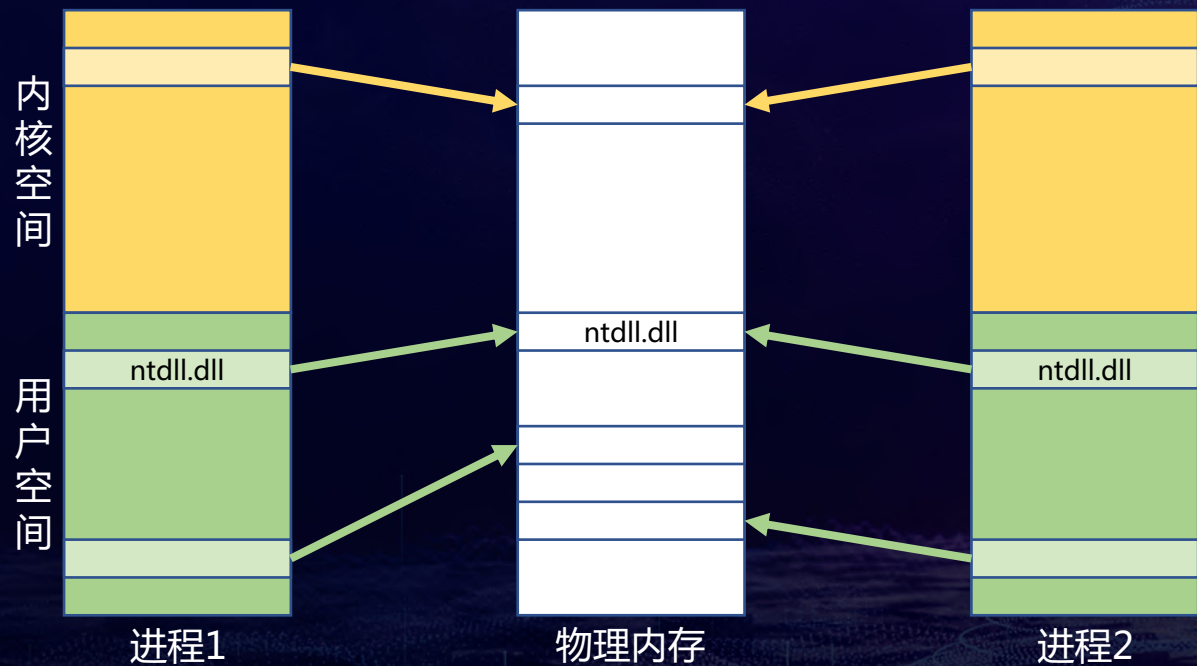
每个进程独占其用户空间



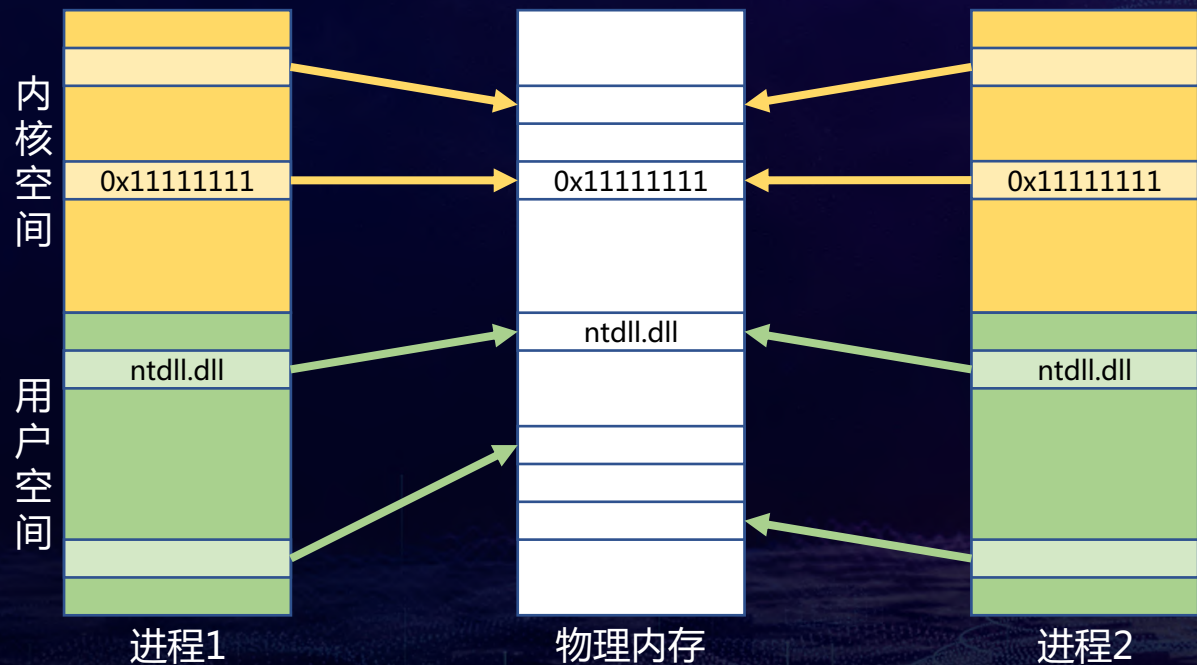
所有进程共享内核空间



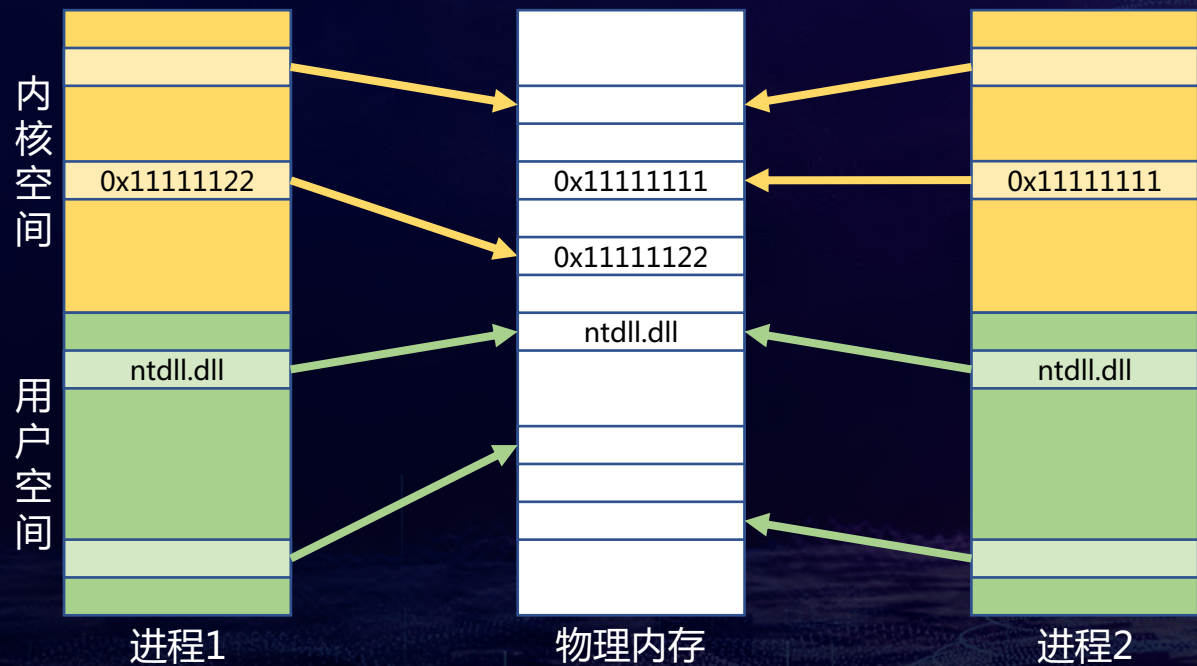
用户空间的内存共享



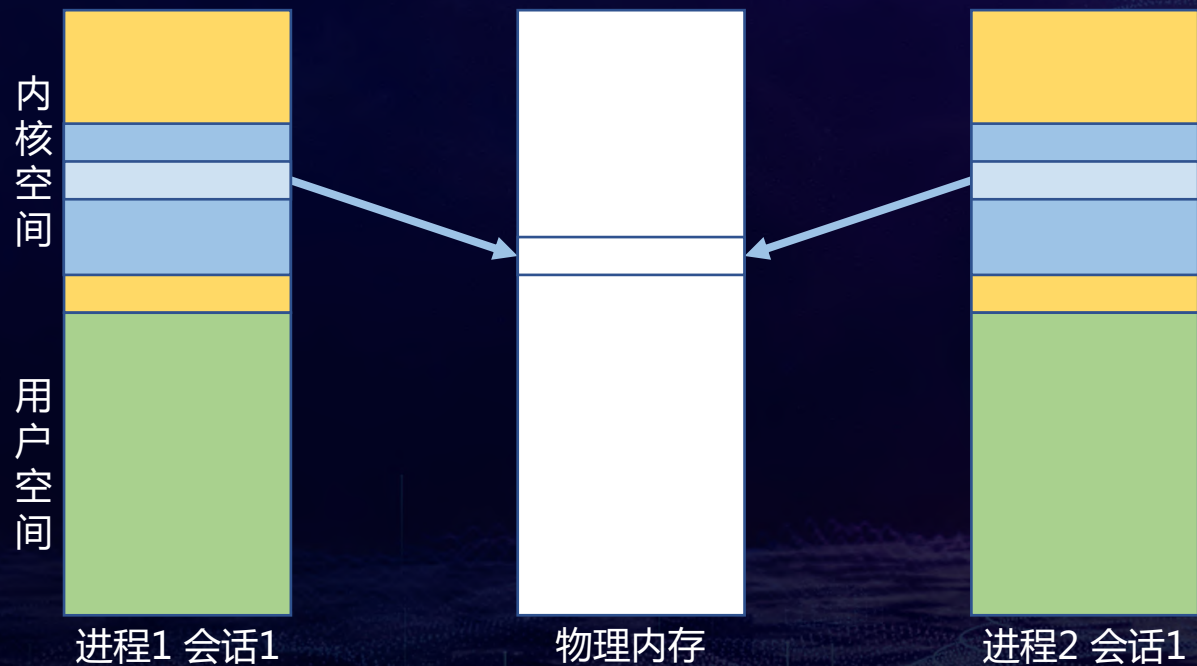
内核空间的写入时复制 (Copy-on-Write)



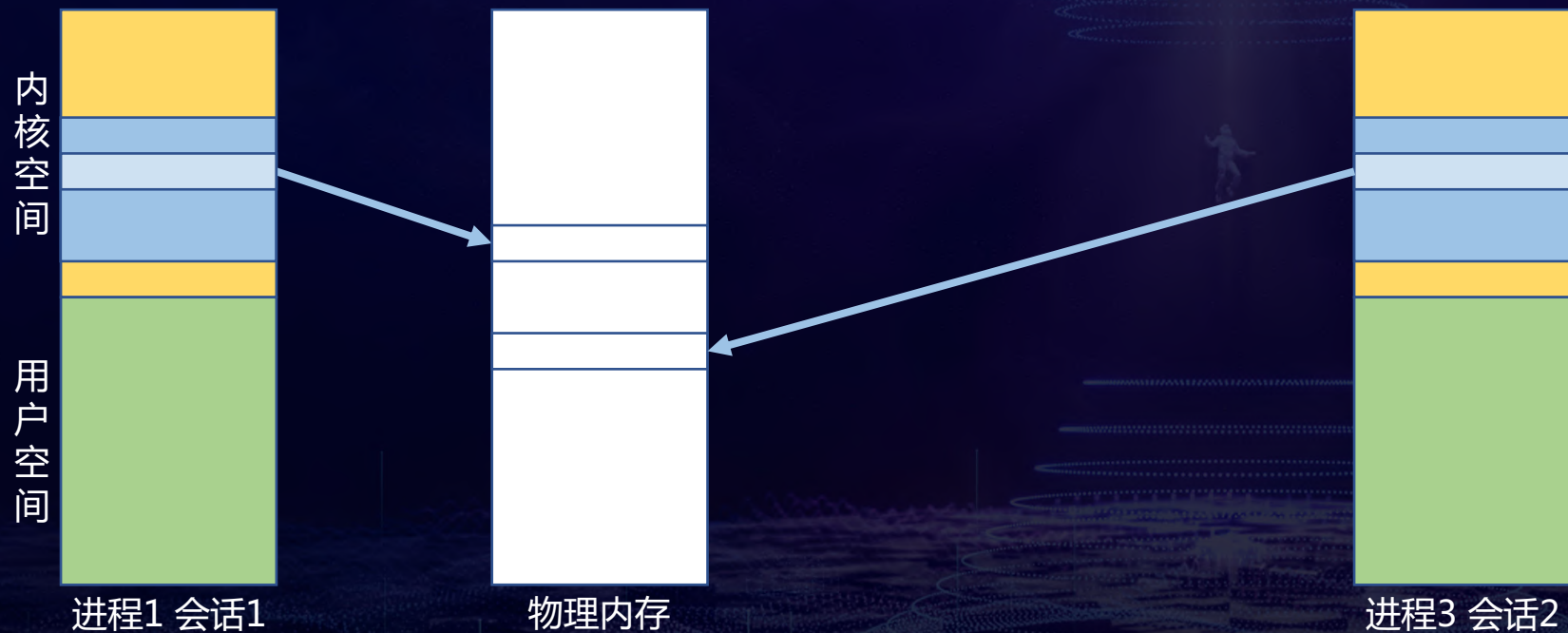
内核空间的写入时复制 (Copy-on-Write)



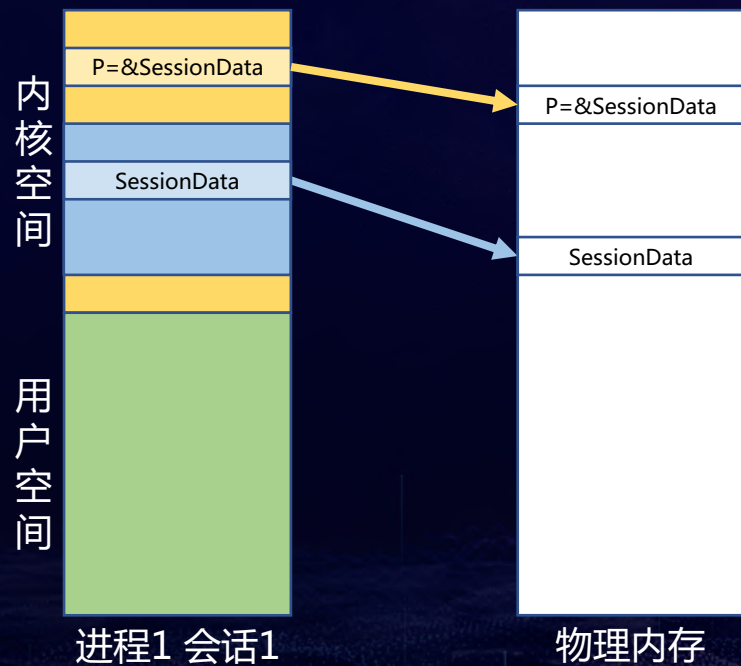
会话空间



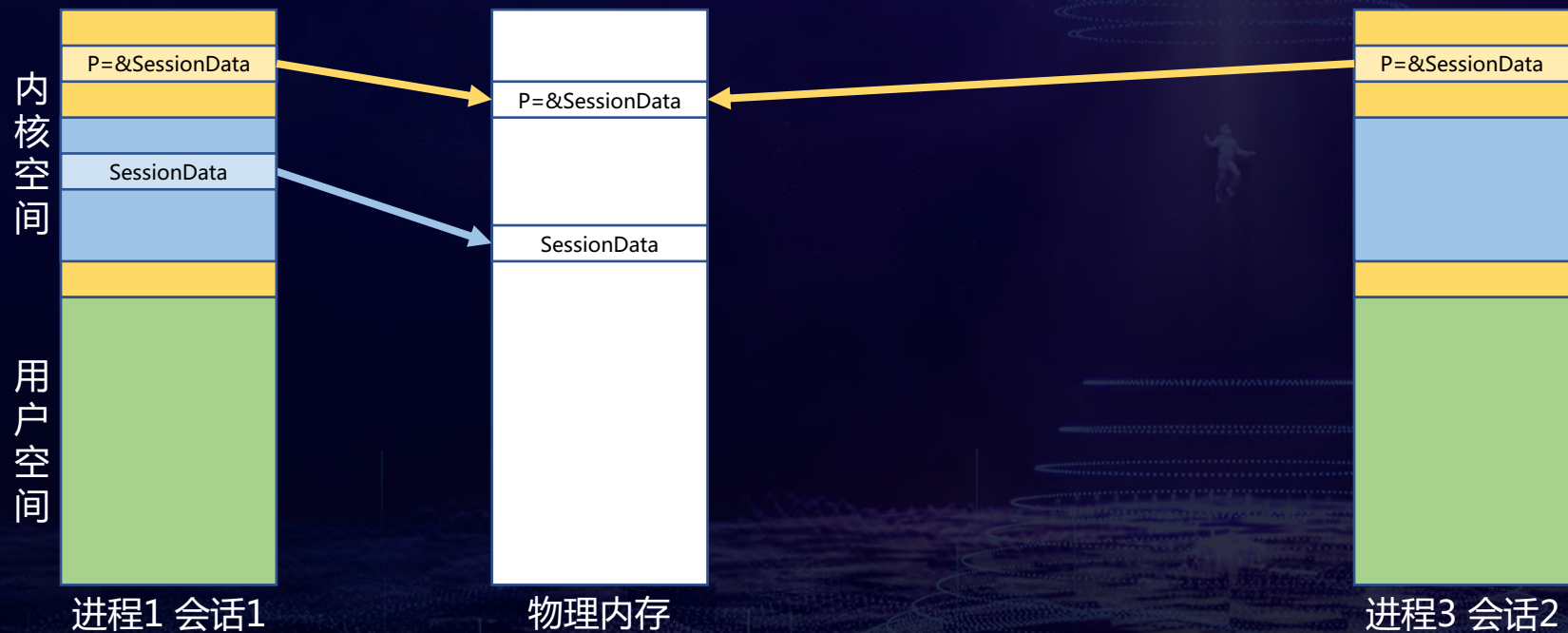
会话空间



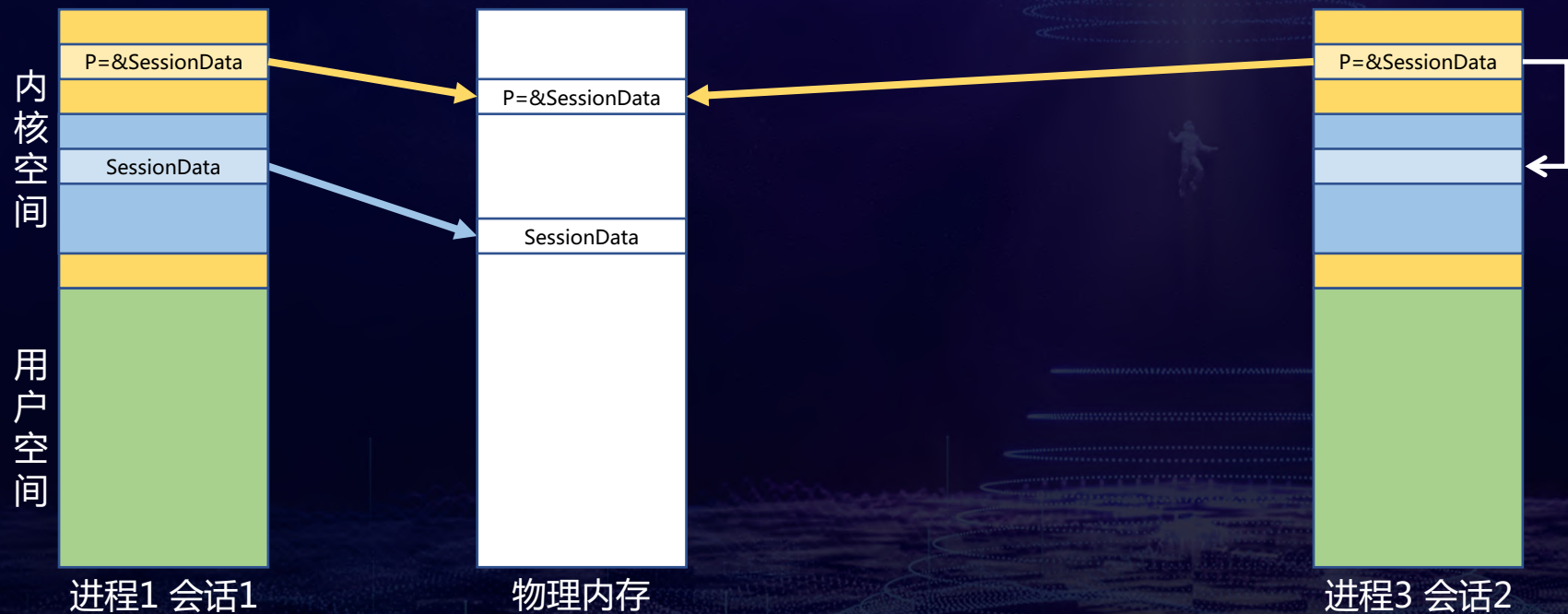
指向会话空间的指针



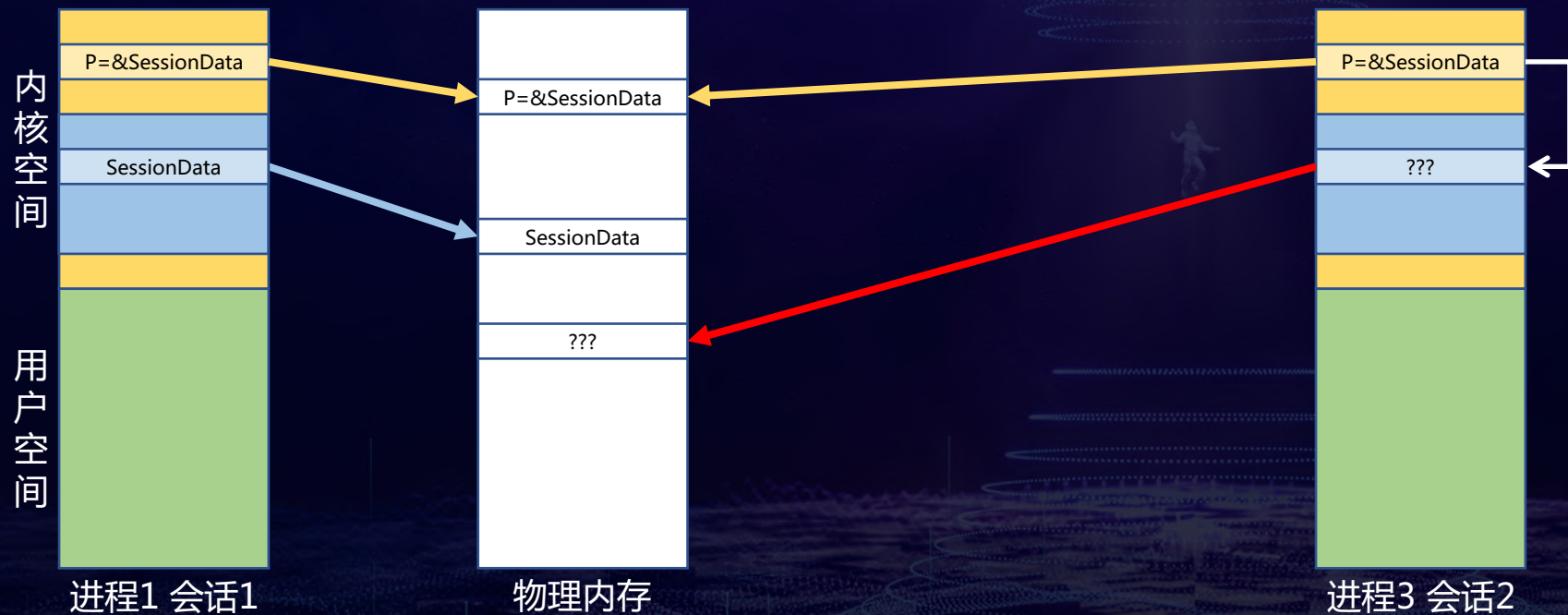
指向会话空间的指针



指向会话空间的指针



指向会话空间的指针



物理页面混淆 Physical Page Confusion

两个问题

是否存在这样的指针？

如何让其他会话中的进程来使用这个指针？

_KTHREAD 中的 Win32Thread

```
1: kd> dt nt!_KTHREAD @$thread Win32Thread
+0x1c8 Win32Thread : 0xffffcb83`318a46a0 Void
```

```
1: kd> dq poi(@$thread + 1c8) l1
ffffcb83`318a46a0 ffff8e2e`4061c010
```

```
1: kd> dt -r nt!_EPROCESS Session Session.PagedPoolStart Session.PagedPoolEnd @$proc
+0x400 Session : 0xffffb501`85f3b000 MM_SESSION_SPACE
+0x038 PagedPoolStart : 0xffff8e2e`40000000 Void
+0x040 PagedPoolEnd : 0xffff8e4e`3fffffff Void
```

```
1: kd> !pte poi(@$thread + 1c8)
VA fffffcb83318a46a0
PXE at FFFFF5FAFD7EBCB8 PPE at FFFFF5FAFD797060 PDE at FFFFF5FAF2E0CC60 PTE at FFFFF5E5C198C520
contains 0A000000003532863 contains 0A000000003535863 contains 0A0000000036ED8863 contains 8A0000000036606A63
pfn 3532 ---DA--KWEV pfn 3535 ---DA--KWEV pfn 36ed8 ---DA--KWEV pfn 36606 C--DA--KW-V
```

如何使用 Win32Thread

```
void *__fastcall PsGetThreadWin32Thread(PKTHREAD Thread)
{
    return Thread->Win32Thread;
}
```

```
v10 = PsGetThreadWin32Thread(__readgsqword(0x188u));
if ( v10 && (v11 = *v10) != 0 && !(a3 & 0x10) )
    v12 = *(v11 + 0x48);
else
    v12 = 0i64;
```


通过 GS 段映射的 _KPCR 获取当前线程

```
1: kd> !pcr
KPCR for Processor 1 at fffffb50184ac1000:
  Major 1 Minor 1
    NtTib.ExceptionList: fffffb50184ad4fb0
    NtTib.StackBase: fffffb50184ad3000
    NtTib.StackLimit: 00000000000000000
    NtTib.SubSystemTib: fffffb50184ac1000
    NtTib.Version: 00000000084ac1180
    NtTib.UserPointer: fffffb50184ac1870
    NtTib.SelfTib: 000000082b03d6000

    SelfPcr: 00000000000000000
    Prcb: fffffb50184ac1180
    Irql: 00000000000000000
    IRR: 00000000000000000
    IDR: 00000000000000000
    InterruptMode: 00000000000000000
    IDT: 00000000000000000
    GDT: 00000000000000000
    TSS: 00000000000000000

    CurrentThread: fffffcb833187b080
    NextThread: 00000000000000000
    IdleThread: fffffb50184ad1080
```

```
1: kd> rdmsr c0000101
msr[c0000101] = fffffb501`84ac1000
```

```
1: kd> dt nt!_KPCR Prcb.CurrentThread fffffb50184ac1000
+0x180 Prcb
+0x008 CurrentThread : 0xffffcb83`3187b080 _KTHREAD
```

KiStackAttachProcess 更新 CR3 同时保持 _KPCR.Prcb.CurrentThread 不变

```
CurrentThread->MiscFlags |= 0x800u;  
CurrentThread->ApcState.Process = TargetProcess;  
CurrentThread->ThreadLock = 0i64;  
Prcb = KeGetCurrentPrcb();  
CurrentProcess = CurrentThread->SavedApcState.Process;  
GroupIndex = Prcb->GroupIndex;  
Offset = 8i64 * Prcb->Group + 0x118;  
_interlockedbittestandset64(&TargetProcess->Header.Lock + Offset, GroupIndex);  
JUMPOUT(Hv1Enlightenments & 1, 0, sub 1401785ED);  
__writecr3(TargetProcess->DirectoryTableBase);  
_interlockedbittestandreset64(&CurrentProcess->Pcb.Header.Lock + Offset, GroupIndex);  
CurrentThread->MiscFlags &= 0xFFFFF7FF;  
__writecr8(Irql);  
*(a3 + 0x20) = 0i64;
```


一个导致物理页面混淆漏洞的模式

KiStackAttachProcess(ProcessInOtherSession)

Win32Thread = PsGetThreadWin32Thread(KeGetCurrentThread())

读写 *Win32Thread

案例分析：CVE-2019-0892

NtTerminateProcess 在关闭句柄前会调用 KiStackAttachProcess

#	Child-SP	RetAddr	Call Site
00	fffff08d`7d967828	fffff802`37d334cc	nt!KiStackAttachProcess
01	fffff08d`7d967830	fffff802`37d055e0	nt!ExSweepHandleTable+0x13f0ec
02	fffff08d`7d9678e0	fffff802`37b1fa9b	nt!PspRundownSingleProcess+0x19069c
03	fffff08d`7d967960	fffff802`37bcb7c8	nt!PspTerminateAllThreads+0x21f
04	fffff08d`7d9679d0	fffff802`37bcb599	nt!PspTerminateProcess+0xe0
05	fffff08d`7d967a10	fffff802`377c4085	nt!NtTerminateProcess+0xa9
06	fffff08d`7d967a80	00007fff`7f26eb14	nt!KiSystemServiceCopyEnd+0x25
07	0000008d`4ab2f7c8	00007fff`7b73cec0	ntdll!NtTerminateProcess+0x14

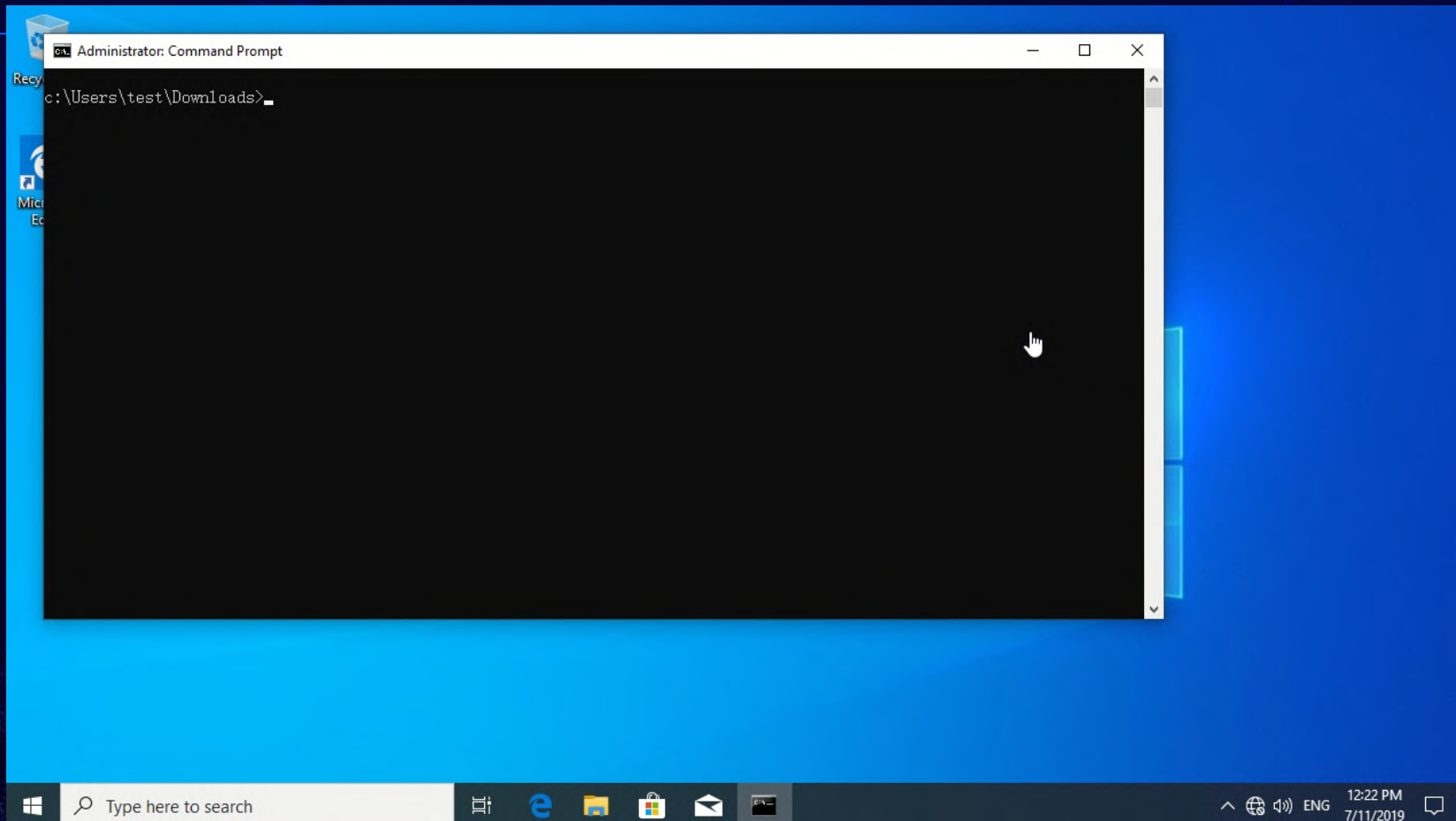
DxgkCompositionObject 对象在删除时会调用 RGNMEMOBJ::vPushThreadGuardedObject

#	Child-SP	RetAddr	Call Site
00	ffff820d`4b13f3c8	ffff8e12`683dbf1a	win32kbase!RGNMEMOBJ::vPushThreadGuardedObject
01	ffff820d`4b13f3d0	ffff8e12`683dbe59	win32kbase!CRegion::InternalCombine+0xb6
02	ffff820d`4b13f430	ffff803`19206dd2	win32kbase!CRegion::Combine+0x9
03	ffff820d`4b13f460	ffff803`1922a235	dxgkrnl!CCompositionToken::UpdateDirtyRegions+0x11a
04	ffff820d`4b13f4b0	ffff803`192070fb	dxgkrnl!CCompositionToken::Discard+0x13ab5
05	ffff820d`4b13f4e0	ffff803`19217469	dxgkrnl!CCompositionToken::MarkInvalid+0x3b
06	ffff820d`4b13f510	ffff803`19216fd3	dxgkrnl!CCompositionToken::Delete+0x29
07	ffff820d`4b13f540	ffff803`18a4a5f0	dxgkrnl!DxgkCompositionObject::Delete+0x73

RGNMEMOBJ::vPushThreadGuardedObject 会调用 PsGetThreadWin32Thread 并使用获取的 Win32Thread

```
void __fastcall RGNMEMOBJ::vPushThreadGuardedObject(RGNMEMOBJ *this)
{
    __int64 *Win32Thread; // rax MAPDST
    _QWORD *v3; // rdi
    _QWORD *v4; // rbx
    _QWORD *v5; // rsi
    __int64 v7; // rcx
    _QWORD *v8; // rax

    Win32Thread = PsGetThreadWin32Thread(__readgsqword(0x188u));
    if ( Win32Thread )
    {
        if ( *Win32Thread )
        {
            v3 = *this;
            if ( v3 )
            {
                v4 = v3 + 6;
                if ( v3 != 0xFFFFFFFFFFFFFFFFD0i64 )
                {
                    KeEnterCriticalRegion();
                    v5 = 0i64;
                    Win32Thread = PsGetThreadWin32Thread(__readgsqword(0x188u));
                    if ( Win32Thread )
                        v5 = *Win32Thread;
                }
            }
        }
    }
}
```



Kernel 'net:port=50000,key=*****' - WinDbg:10.0.17763.132 AMD64

File Edit View Debug Window Help

Command

```
Debugger entered on first try; Bugcheck callbacks have not been invoked.

A fatal system error has occurred.

rax=0000000000000000 rbx=0000000000000003 rcx=0000000000000003
rdx=0000000000000008a rsi=0000000000000000 rdi=ffff8023c011180
rip=ffff8023ce58b90 rsp=ffff8186d68bd938 rbp=ffff8186d68bdaa0
r8=00000000000000065 r9=0000000000000000 r10=ffff8186d68bd730
r11=0000000000000000 r12=0000000000000003 r13=00000000c0000005
r14=0000000000000000 r15=ffff8c02dcd74080
iopl=0         nv up ei ng nz na pe nc
cs=0010  ss=0018  ds=002b  es=002b  fs=0053  gs=002b             efl=00000282
nt!DbgBreakPointWithStatus:
ffff802`3ce58b90 cc          int     3
0: kd> .cxr 0xFFFF8186D68BEA30
rax=ffff9e08038ce688 rbx=ffff9e0803186cb0 rcx=4141414141414141
rdx=ffff9e0803186ce8 rsi=ffff9e08038ce630 rdi=ffff9e0803186c80
rip=ffff9e50776de3e9 rsp=ffff8186d68bf420 rbp=ffff8186d68bf480
r8=ffff9e0803186c80  r9=0000000000000000 r10=ffff8c02d91d0d80
r11=ffff8186d68bf410 r12=ffffa288aa0955d0 r13=ffff9e08030f4180
r14=ffff9e080061a3f0 r15=ffff9e08030f4180
iopl=0         nv up ei ng nz na po nc
cs=0010  ss=0018  ds=002b  es=002b  fs=0053  gs=002b             efl=00010286
win32kbase!RGNMEMOBJ::vPushThreadGuardedObject+0x99:
ffff9e50`776de3e9 48394108      cmp     qword ptr [rcx+8],rax ds:002b:41414141`41414149=????????????????
0: kd>
```

Ln 0, Col 0 Sys 0:KdSrv:S Proc 000:0 Thrd 000:0 ASM OVR CAPS NUM

问题修复

引入函数 IsThreadCrossSessionAttached 进行检查

```
BOOL8 IsThreadCrossSessionAttached()
{
    PEPROCESS CurrentProcess; // rax
    int Id; // ebx
    PEPROCESS CurrentThreadProcess; // rax
    _BOOL8 result; // rax

    result = 0;
    if ( KeIsAttachedProcess() )
    {
        CurrentProcess = PsGetCurrentProcess();
        Id = PsGetProcessSessionIdEx(CurrentProcess);
        CurrentThreadProcess = PsGetCurrentThreadProcess();
        if ( Id != PsGetProcessSessionIdEx(CurrentThreadProcess) )
            result = 1;
    }
    return result;
}
```

```
PEPROCESS *PsGetCurrentProcess()
{
    return KeGetCurrentThread()->ApcState.Process;
}
```

```
PEPROCESS *PsGetCurrentThreadProcess()
{
    return KeGetCurrentThread()->Process;
}
```

用 W32GetThreadWin32Thread 封装 PsGetThreadWin32Thread

```
__int64 __fastcall W32GetThreadWin32Thread(PETHREAD Thread)
{
    __int64 v2; // rbx
    __int64 *Win32Thread; // rax

    v2 = 0i64;
    if ( !IsThreadCrossSessionAttached() )
    {
        Win32Thread = PsGetThreadWin32Thread(Thread);
        if ( Win32Thread )
            v2 = *Win32Thread;
    }
    return v2;
}
```


RGNMEMOBJ::vPushThreadGuardedObject 调用 W32GetThreadWin32Thread 获取 Win32Thread

```
void __fastcall RGNMEMOBJ::vPushThreadGuardedObject(RGNMEMOBJ *this)
{
    _QWORD *v2; // rdi
    _QWORD *v3; // rbx
    __int64 Win32Thread; // rax
    __int64 v5; // rcx
    _QWORD *v6; // rax

    if ( W32GetThreadWin32Thread(__readgsqword(0x188u)) )
    {
        v2 = *this;
        if ( v2 )
        {
            v3 = v2 + 6;
            if ( v2 != 0xFFFFFFFFFFFFFFFFD0i64 )
            {
                KeEnterCriticalRegion();
                Win32Thread = W32GetThreadWin32Thread(__readgsqword(0x188u));
                v2[8] = v2;
                v2[9] = CleanUpRegion;
                if ( Win32Thread )
                {
                    v5 = *(Win32Thread + 0x58);
                }
            }
        }
    }
}
```

增加 IsThreadCrossSessionAttached 检查的函数

NtGdiDeleteObjectApp

bDeleteDCOBJ

GreGetDeviceCaps

ReleaseCacheDC

_GetDCEx

GreGetBounds

XDCOBJ::bCleanDC

HmgDecrementShareReferenceCountEx

HmgLockEx

HANDLELOCK::vLockHandle

ResetOrg

HANDLELOCK::bLockHobj

hbmSelectBitmap

DCMEMOBJ::DCMEMOBJ

W32GetThreadWin32Thread

SURFMEM::bCreateDIB

XDCOBJ::bDeleteDC

GreIntersectClipRect

Call to Action

使用错误的页表进行虚拟地址转换会导致物理页面混淆类漏洞
操作系统应当提供机制来判断是否可以安全的使用虚拟地址
开发人员应当认识到这类漏洞的存在从而做出相应的处理

感谢聆听