# 当今操作系统普遍采用64位架构，CPU最大寻址能力虽然达到了64位，但其实仅仅只是用到了48位进行 寻址，其内存管理采用了 9-9-9-9-12 的分页模式， 9-9-9-9-12 分页表示物理地址拥有四级页表，微软将这四级依次命名为PXE、PPE、PDE、PTE这四项。

关于内存管理和分页模式，不同的操作系统和体系结构可能会有略微不同的实现方式。9-9-9-9-12的分页 模式是一种常见的分页方案，其中物理地址被分成四级页表：PXE（Page Directory Pointer Table Entry）、PPE（Page Directory Entry）、PDE（Page Table Entry）和PTE（Page Table Entry）。这种分页模式可以支持大量的物理内存地址映射到虚拟内存地址空间中。每个级别的页表都负责将虚拟地址 映射到更具体的物理地址。通过这种层次化的页表结构，操作系统可以更有效地管理和分配内存。

首先一个PTE管理1个分页大小的内存也就是 0x1000 字节，PTE结构的解析非常容易，打开WinDBG输入 !PTE 0 即可解析，如下所示，当前地址0位置处的PTE基址是 FFFF898000000000 ，由于PTE的一个页大小是 0x1000 所以当内存地址高于 0x1000 时将会切换到另一个页中，如下 FFFF898000000008 则是另一个页中的地址。

contains 0000000000000000

not valid

pfn 405f ---DA--UW-V

0: kd> !PTE 0x1000

VA 0000000000001000

PXE at FFFF89C4E2713000 PPE at FFFF89C4E2600000 PDE at FFFF89C4C0000000 PTE at FFFF898000000008

contains 8A0000000405F867

contains 0000000000000000

not valid

pfn 405f ---DA--UW-V

0: kd> !PTE 0

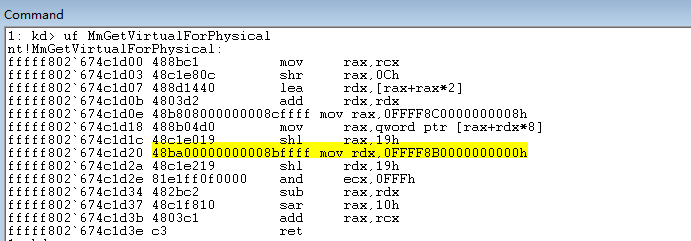
VA 0000000000000000

PXE at FFFF89C4E2713000 PPE at FFFF89C4E2600000 PDE at FFFF89C4C0000000 PTE at FFFF898000000000

contains 8A0000000405F867

由于PTE是动态变化的，找到该地址的关键就在于通过 MmGetSystemRoutineAddress 函数动态得到

MmGetVirtualForPhysical 的内存地址，然后向下扫描特征寻找 mov rdx,0FFFF8B0000000000h 并将内部的地址提取出来。



# 这段代码完整版如下所示，代码可动态定位到PTE的内存地址，然后将其取出；

// 署名权

// right to sign one's name on a piece of work

// PowerBy: LyShark

// Email: [me@lyshark.com](mailto:me@lyshark.com)

#include <ntifs.h> #include <ntstrsafe.h>



// 指定内存区域的特征码扫描

PVOID SearchMemory(PVOID pStartAddress, PVOID pEndAddress, PUCHAR pMemoryData, ULONG ulMemoryDataSize)

{

PVOID pAddress = NULL; PUCHAR i = NULL; ULONG m = 0;

// 扫描内存

for (i = (PUCHAR)pStartAddress; i < (PUCHAR)pEndAddress; i++)

{

// 判断特征码

for (m = 0; m < ulMemoryDataSize; m++)

{

if ( (PUCHAR)(i + m) != pMemoryData[m])

{

break;

}

}

// 判断是否找到符合特征码的地址

if (m >= ulMemoryDataSize)

{

// 找到特征码位置, 获取紧接着特征码的下一地址

pAddress = (PVOID)(i + ulMemoryDataSize); break;

}

}

return pAddress;

}

// 获取到函数地址

PVOID GetMmGetVirtualForPhysical()

{

PVOID VariableAddress = 0; UNICODE\_STRING uioiTime = { 0 };

RtlInitUnicodeString(&uioiTime, L"MmGetVirtualForPhysical"); VariableAddress = (PVOID)MmGetSystemRoutineAddress(&uioiTime); if (VariableAddress != 0)

{

return VariableAddress;

}

return 0;

}

// 驱动卸载例程

VOID UnDriver(PDRIVER\_OBJECT driver)

{

DbgPrint("Uninstall Driver \n");

}

// 驱动入口地址

NTSTATUS DriverEntry(IN PDRIVER\_OBJECT Driver, PUNICODE\_STRING RegistryPath)

{

DbgPrint("Hello LyShark \n");

// 获取函数地址

PVOID address = GetMmGetVirtualForPhysical();

DbgPrint("GetMmGetVirtualForPhysical = %p \n", address);

UCHAR pSecondSpecialData[50] = { 0 }; ULONG ulFirstSpecialDataSize = 0;

pSecondSpecialData[0] = 0x48; pSecondSpecialData[1] = 0xc1; pSecondSpecialData[2] = 0xe0; ulFirstSpecialDataSize = 3;

// 定位特征码

PVOID PTE = SearchMemory(address, (PVOID)((PUCHAR)address + 0xFF), pSecondSpecialData, ulFirstSpecialDataSize);

try

{

PVOID lOffset = (ULONG)PTE + 1;

DbgPrint("PTE Address = %p \n", lOffset);

}

except (EXCEPTION\_EXECUTE\_HANDLER)

{

DbgPrint("error");

}

Driver->DriverUnload = UnDriver; return STATUS\_SUCCESS;

}

# 运行如上代码可动态获取到当前系统的PTE地址，然后将PTE填入到 g\_PTEBASE 中，即可实现解析系统内的四个标志位，完整解析代码如下所示；



// 署名权

// right to sign one's name on a piece of work

// PowerBy: LyShark

// Email: [me@lyshark.com](mailto:me@lyshark.com)

#include <ntifs.h> #include <ntstrsafe.h>

INT64 g\_PTEBASE = 0; INT64 g\_PDEBASE = 0; INT64 g\_PPEBASE = 0; INT64 g\_PXEBASE = 0;

PULONG64 GetPteBase(PVOID va)

{

return (PULONG64)((((ULONG64)va & 0xFFFFFFFFFFFF) >> 12) 8) + g\_PTEBASE;

}

PULONG64 GetPdeBase(PVOID va)

{



return (PULONG64)((((ULONG64)va & 0xFFFFFFFFFFFF) >> 12) 8) + g\_PDEBASE;

}

PULONG64 GetPpeBase(PVOID va)

{

return (PULONG64)((((ULONG64)va & 0xFFFFFFFFFFFF) >> 12) 8) + g\_PPEBASE;

}

PULONG64 GetPxeBase(PVOID va)

{

return (PULONG64)((((ULONG64)va & 0xFFFFFFFFFFFF) >> 12) 8) + g\_PXEBASE;

}

// 驱动卸载例程

VOID UnDriver(PDRIVER\_OBJECT driver)

{

DbgPrint("Uninstall Driver \n");

}

// 驱动入口地址

NTSTATUS DriverEntry(IN PDRIVER\_OBJECT Driver, PUNICODE\_STRING RegistryPath)

{

DbgPrint("Hello LyShark \n");

g\_PTEBASE = 0XFFFFF20000000000;

g\_PDEBASE = (ULONG64)GetPteBase((PVOID)g\_PTEBASE); g\_PPEBASE = (ULONG64)GetPteBase((PVOID)g\_PDEBASE); g\_PXEBASE = (ULONG64)GetPteBase((PVOID)g\_PPEBASE);

DbgPrint("PXE = %p \n", g\_PXEBASE); DbgPrint("PPE = %p \n", g\_PPEBASE); DbgPrint("PDE = %p \n", g\_PDEBASE); DbgPrint("PTE = %p \n", g\_PTEBASE);

Driver->DriverUnload = UnDriver; return STATUS\_SUCCESS;

}

我的系统内PTE地址为 0XFFFFF20000000000 ，填入变量内解析效果如下图所示；

