在前面的系列教程如 《驱动开发：内核枚举DpcTimer定时器》 或者 《驱动开发：内核枚举IoTimer定时器》里面 LyShark 大量使用了 特征码定位 这一方法来寻找符合条件的 汇编指令 集，总体来说这种方式只能定位特征较小的指令如果特征值扩展到5位以上那么就需要写很多无用的代码，本章内容中将重点分析，并 实现一个 通用 特征定位函数。

如下是一段特征码搜索片段，可以看到其实仅仅只是将上章中的搜索方式变成了一个

SearchSpecialCode 函数，如下函数，用户传入一个 扫描起始地址 以及搜索特征码的字节数组，即可完成搜索工作，具体的参数定义如下。

pSearchBeginAddr 扫描的内存(内核)起始地址

ulSearchLength 需要扫描的长度

pSpecialCode 扫描特征码,传入一个UCHAR类型的字节数组

ulSpecialCodeLength 特征码长度,传入字节数组长度



// By: LyShark.com

PVOID SearchSpecialCode(PVOID pSearchBeginAddr, ULONG ulSearchLength, PUCHAR pSpecialCode, ULONG ulSpecialCodeLength)

{

PVOID pDestAddr = NULL;

PUCHAR pBeginAddr = (PUCHAR)pSearchBeginAddr; PUCHAR pEndAddr = pBeginAddr + ulSearchLength; PUCHAR i = NULL;

ULONG j = 0;

for (i = pBeginAddr; i <= pEndAddr; i++)

{

// 遍历特征码

for (j = 0; j < ulSpecialCodeLength; j++)

{

// 判断地址是否有效

if (FALSE == MmIsAddressValid((PVOID)(i + j)))

{

break;

}

// 匹配特征码

if ( (PUCHAR)(i + j) != pSpecialCode[j])

{

break;

}

}

// 匹配成功

if (j >= ulSpecialCodeLength)

{

pDestAddr = (PVOID)i; break;

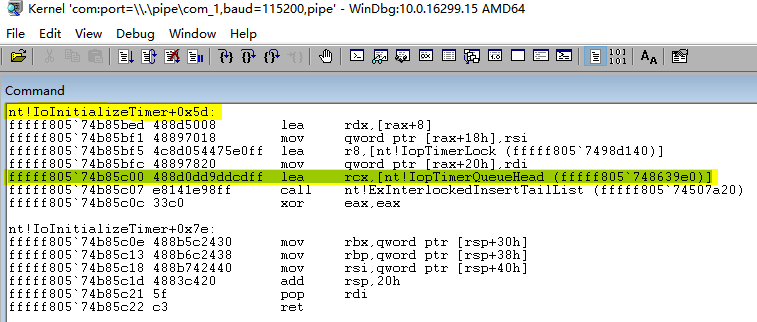
}

}

return pDestAddr;

}

那么这个简单的特征码扫描函数该如何使用，这里我们就用 《驱动开发：内核枚举IoTimer定时器》 中枚举 IopTimerQueueHead 链表头部地址为案例进行讲解，如果你忘记了如何寻找链表头部可以去前面的文章中学习，这里只给出实现流程。



我们首先通过 MmGetSystemRoutineAddress 得到 IoInitializeTimer 首地址，然后在偏移长度为

0x7e 范围内搜索特征码 48 8d 0d 特征，其代码可以总结为如下样子。

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

{

DbgPrint(("hello lyshark.com \n"));

// 得到基址

PUCHAR IoInitializeTimer = GetIoInitializeTimerAddress(); DbgPrint("IoInitializeTimer Address = %p \n", IoInitializeTimer);

// ---------------------------------------------------

// LyShark 开始定位特征

// 设置起始位置

PUCHAR StartSearchAddress = (PUCHAR)IoInitializeTimer;

// 设置结束位置

PUCHAR EndSearchAddress = StartSearchAddress + 0x7e;

DbgPrint("[LyShark 搜索区间] 起始地址: 0x%X --> 结束地址: 0x%X \n", StartSearchAddress, EndSearchAddress);

// 设置搜索长度

LONGLONG size = EndSearchAddress - StartSearchAddress;

DbgPrint("[LyShark 搜索长度] 长度: %d \n", size); PVOID ptr;

// 指定特征码

UCHAR pSpecialCode[256] = { 0 };

// 指定特征码长度

ULONG ulSpecialCodeLength = 3;

pSpecialCode[0] = 0x48; pSpecialCode[1] = 0x8d; pSpecialCode[2] = 0x0d;

// 开始搜索,找到后返回首地址

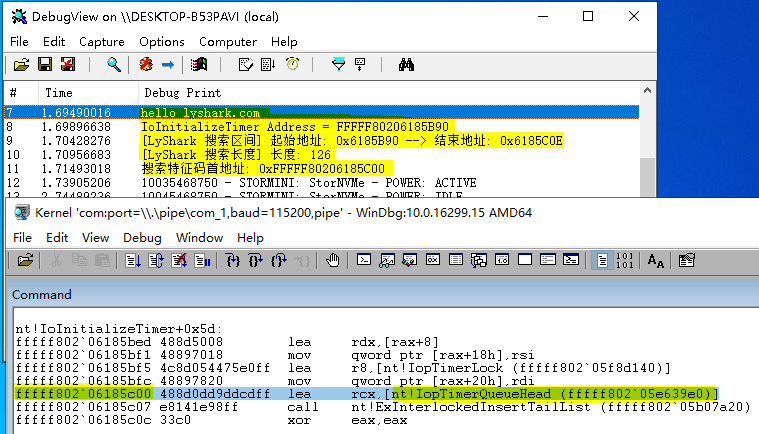
ptr = SearchSpecialCode(StartSearchAddress, size, pSpecialCode, ulSpecialCodeLength);

DbgPrint("搜索特征码首地址: 0x%p \n", ptr);

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

}

代码运行后你会发现可以直接定位到我们所需要的位置上，如下图所示：



如上图可以看到，这个特征码定位函数返回的是内存地址，而我们需要得到地址内的数据，此时就需要 提取以下。

例如当指令是:

fffff80206185c00 488d0dd9ddcdff lea rcx,[nt!IopTimerQueueHead (fffff80205e639e0)]

那么就需要 RtlCopyMemory 跳过前三个字节，并在第四个字节开始取数据，并将读入的数据放入到

IopTimerQueueHead\_LyShark\_Code 变量内。

// 署名

// PowerBy: LyShark

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

// 开始搜索,找到后返回首地址

ptr = SearchSpecialCode(StartSearchAddress, size, pSpecialCode, ulSpecialCodeLength);

DbgPrint("搜索特征码首地址: 0x%p \n", ptr);

// 提取特征

// fffff802`06185c00 488d0dd9ddcdff lea rcx,[nt!IopTimerQueueHead (fffff802`05e639e0)]

ULONG64 iOffset = 0;

ULONG64 IopTimerQueueHead\_LyShark\_Code = 0;

try

{

// 拷贝内存跳过lea,向后四字节

RtlCopyMemory(&iOffset, (ULONG64)ptr + 3, 4);

// 取 出 后 面 的 IopTimerQueueHead 内 存 地 址 LyShark.com IopTimerQueueHead\_LyShark\_Code = iOffset + (ULONG64)ptr + 7;

DbgPrint("提取数据: 0x%p \n", IopTimerQueueHead\_LyShark\_Code);

}

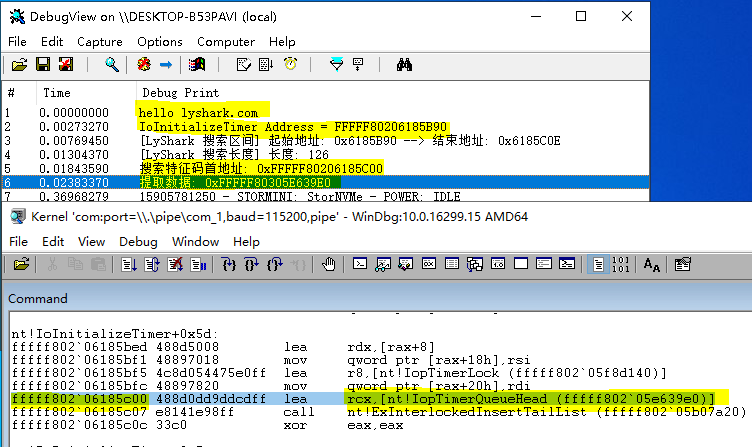
except (1)

{

DbgPrint("[LySHark] 拷贝内存异常 \n");

}

这样即可得到我们所需要的地址，如下结果所示：



作者： 王瑞 (LyShark)

作者邮箱： m [e@lyshark.com](mailto:e@lyshark.com)

版权声明：本博客文章与代码均为学习时整理的笔记，文章 [均为原创] 作品，转载文章请遵守

《中华人民共和国著作权法》相关法律规定或遵守《署名CC BY-ND 4.0国际》规范，合理合规携带原创出处转载，如果不携带文章出处，并恶意转载多篇原创文章被本人发现，本人保留起诉权！