在上一篇文章《驱动开发:内核取ntoskrn]模块基地址》中我们通过调用内核API函数获取到了内核进程 ntoskrnl.exe 的基址,当在某些场景中,我们不仅需要得到内核的基地址,也需要得到特定进程内某 个模块的基地址,显然上篇文章中的方法是做不到的,本篇文章将实现内核层读取32位应用层中特定进程模块基址功能。

上一篇文章中的 PPEB32, PLIST_ENTRY32 等结构体定义依然需要保留,此处只保留核心代码,定义部分请看前一篇文章,自定义读取模块基址核心代码如下,调用 GetModuleBasewow64() 用户需传入进程的 PROCESS 结构该结构可通过内核函数 PSLookupProcessByProcessId 获取到。

对于函数内部执行过程如下:

- 1.根据传入的 EProcess 结构调用 KeStackAttachProcess 附加到该进程内。
- 2.调用内核函数 PSGetProcessWow64Process 此函数可得到该进程空间内PEB结构数据。
- 3.通过for循环遍历整个 pPeb->Ldr 链表,并在遍历过程中通过 Rt1EqualUnicodeString 判断是否是我们需要的模块。
- 4.如果判断是我们需要取出的模块名,则将 LdrEntry->DllBase 取出,此处取出的基地址也即是我们所需要的。
- 5.比较结束后,通过调用 KeUnstackDetachProcess 这个内核模块脱离进程空间。

```
// By: LyShark
ULONGLONG GetModuleBaseWow64(_In_ PEPROCESS pEProcess, _In_ UNICODE_STRING
usModuleName)
{
   ULONGLONG BaseAddr = 0;
   KAPC\_STATE KAPC = \{ 0 \};
   KeStackAttachProcess(pEProcess, &KAPC);
   PPEB32 pPeb = (PPEB32)PsGetProcessWow64Process(pEProcess);
   if (pPeb == NULL || pPeb->Ldr == 0)
        KeUnstackDetachProcess(&KAPC);
       return 0:
   }
    for (PLIST_ENTRY32 pListEntry = (PLIST_ENTRY32)((PPEB_LDR_DATA32)pPeb->Ldr)-
>InLoadOrderModuleList.Flink;
        pListEntry != &((PPEB_LDR_DATA32)pPeb->Ldr)->InLoadOrderModuleList;
pListEntry = (PLIST_ENTRY32)pListEntry->Flink)
        PLDR_DATA_TABLE_ENTRY32 LdrEntry = CONTAINING_RECORD(pListEntry,
LDR_DATA_TABLE_ENTRY32, InLoadOrderLinks);
        if (LdrEntry->BaseDllName.Buffer == NULL)
        {
           continue;
        }
        // 当前模块名链表
        UNICODE_STRING usCurrentName = { 0 };
        RtlInitUnicodeString(&usCurrentName, (PWCHAR)LdrEntry-
>BaseDllName.Buffer);
        // 比较模块名是否一致
```

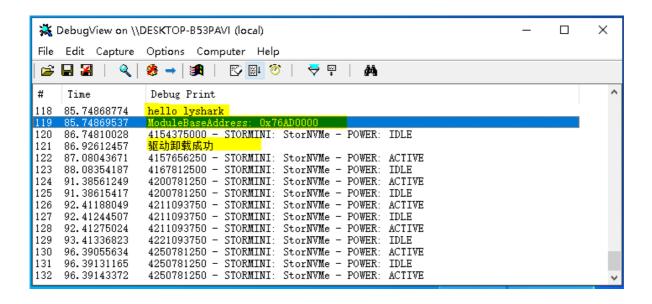
```
if (RtlEqualUnicodeString(&usModuleName, &usCurrentName, TRUE))
{
    BaseAddr = (ULONGLONG)LdrEntry->DllBase;
    KeUnstackDetachProcess(&KAPC);
    return BaseAddr;
}

KeUnstackDetachProcess(&KAPC);
return 0;
}
```

如上就是如何得到特定模块基址的方法,如下是入口函数的调用方法,首先通过传入 6164 这个PID号,得到进程 EProcess 结构,其次使用 RtlInitUnicodeString(&unicode, wchar_string) 初始化得到 kernel32.dll 字符串,最终调用 GetModuleBasewow64 函数获取到进程 6164 中 kernel32.dll 的模块基地址信息。

```
VOID UnDriver(PDRIVER_OBJECT driver)
{
   DbgPrint("驱动卸载成功 \n");
}
NTSTATUS DriverEntry(IN PDRIVER_OBJECT Driver, PUNICODE_STRING RegistryPath)
   DbgPrint(("hello lyshark \n"));
   PEPROCESS pEProcess;
   HANDLE PID = (HANDLE)6164;
   // 初始化字符串
   UNICODE_STRING unicode;
   wchar_t *wchar_string = L"kernel32.dll";
   RtlInitUnicodeString(&unicode, wchar_string);
   // 取模块句柄
    PsLookupProcessByProcessId((HANDLE)PID, &pEProcess);
   ULONGLONG base32 = GetModuleBaseWow64(pEProcess, unicode);
   DbgPrint("ModuleBaseAddress: 0x%X \n", base32);
   Driver->DriverUnload = UnDriver;
    return STATUS_SUCCESS;
}
```

这段代码输出效果如下所示:



作者: 王瑞 (LyShark)

作者邮箱: me@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!