

**恶意代码分析与防治课程实验报告**

**实验8-2：r77 rootkit**

****

学 院 网络空间安全学院

专 业 信息安全

学 号 2111033

姓 名 艾明旭

班 级 信息安全一班

1. **实验目的**

首先，SSDT是一个包含系统服务描述符的内核数据结构，它存储了操作系统内核中各种系统服务的入口点。这些系统服务包括文件操作、进程管理、网络通信等。Rootkit是一种恶意软件，其目的是隐藏其存在并操纵操作系统以达到非法目的。Rootkit通常会修改SSDT以劫持系统服务的调用，以便掩盖和操作其行为。

为了寻找Rootkit，我们需要进行内核分析。这包括获取操作系统的内核代码、数据结构以及内核模块的信息。然后，我们可以通过分析SSDT来检测潜在的Rootkit。以下是通常用于这种分析的步骤：

获取SSDT：我们可以使用特权访问或内核调试工具来获取系统的SSDT。这些工具使我们能够读取内核内存并提取SSDT的内容。

比较标准SSDT：我们需要获取操作系统的正常状态下的标准SSDT。这可以通过从可信源获得的操作系统映像中提取SSDT来实现。然后，我们可以将这个标准SSDT与当前系统的SSDT进行比较。

检测差异：对比标准SSDT和当前SSDT，我们可以找出其中的差异。这些差异可能是Rootkit所做的修改。这些修改可能包括将某些系统服务的入口点指向Rootkit的代码而不是操作系统的原始代码。

分析修改：一旦发现差异，我们可以继续分析Rootkit的修改以了解其行为。这可能涉及追踪Rootkit修改的入口点，并分析其替换的代码。

需要注意的是，这只是一种寻找Rootkit并分析内核的基本方法之一。Rootkit的设计者会使用各种技术来隐藏其存在和修改。在实际情况中，对抗Rootkit需要更加复杂和高级的技术和工具。

1. **实验原理**

r77-Rootkit是一款功能强大的无文件Ring 3 Rootkit，并且带有完整的安全工具和持久化机制，可以实现进程、文件和网络连接等操作及任务的隐藏。

r77能够在所有进程中隐藏下列实体：

文件、目录、连接、命名管道、计划任务；

进程；

CPU用量；

注册表键&值；

服务；

TCP&UDP连接；

该工具兼容32位和64位版本的Windows 7以及Windows 10。

2.1 Windows的Detours机制

Windows的Detours机制是一种软件技术，用于拦截和修改Win32函数的调用。这是通过API钩子（hooking）实现的，是一种编程技术，允许开发者拦截对动态链接库（DLL）中函数的调用。Detours 通过动态地重写目标函数的机器代码来工作。这种方法通常用于系统监控、程序调试、性能分析等方面。

在Detours机制中，当一个程序调用某个Win32 API函数时，Detours 可以劫持这个调用，并将它重定向到另一个函数。这个替代函数可以是用户自定义的，也可以是对原函数的扩展或修改。

利用这种技术，开发者可以在不修改原始代码的情况下，动态地更改程序的行为。

例如，Detours 可以用于监视和记录文件操作、网络通信或系统调用。它也被用于创建安全工具，如防病毒软件，这些软件需要监视系统级活动以检测恶意行为。然而，同样的技术也可以被用于恶意软件中，用于隐藏其行为，例如隐藏文件、进程或网络连接，这就是你提到的R77程序所做的事情。

2.2 API Hooking

API（应用程序编程接口）Hooking是一种编程技术，用于改变或增强操作系统或应用程序的功能。在Windows系统中，API函数通常来自各种动态链接库（DLL）。

2.2.1 原理

拦截调用：API Hooking工作原理是拦截对特定API函数的调用。这可以通过多种方式实现，包括修改函数的入口地址（例如，在导入地址表中），或者直接修改函数代码本身（如通过Inline Hooking）。

重定向调用：当API调用被拦截后，它会被重定向到一个自定义函数。这个自定义函数可以在调用原函数之前或之后执行额外的代码，甚至完全替代原函数。

2.2.2 应用

调试和监控：开发人员利用API Hooking来监控和记录应用程序的行为，例如文件访问、网络通信等。

安全软件：安全软件（如防病毒程序）使用API Hooking来检测和阻止恶意行为。

系统增强：通过API Hooking，软件可以添加或修改操作系统功能，不需要修改底层代码。

2.2.3 风险

稳定性问题：不正确的API Hooking可能导致系统不稳定。

安全隐患：恶意软件也可能利用API Hooking来隐藏其行为或损害系统。

2.3 隐藏进程

隐藏进程是一种技术，通常用于防止进程在常规工具（如任务管理器）中被检测到。这在恶意软件和某些类型的系统监控软件中很常见。

2.3.1 实现方法

修改系统结构：通过修改操作系统的内部数据结构（如进程列表）来隐藏进程。

拦截系统调用：使用API Hooking或类似技术拦截和修改系统调用，例如拦截列出进程的API调用，并从结果中删除特定进程。

内核模式驱动：在内核模式下运行的驱动程序可以直接访问和修改操作系统的核心数据结构，进而隐藏进程。

2.3.2 应用

安全和隐私：某些合法软件可能出于安全或隐私原因隐藏其进程。

恶意软件：病毒、木马和rootkits经常隐藏其进程以避免检测。

2.3.3 风险和挑战

安全风险：隐藏进程的技术可以被恶意软件利用，对用户和企业造成严重威胁。

检测难度：隐藏进程的技术提高了安全软件检测和清除这些威胁的难度。

1. **实验过程**

**1.在计算机网络当中选择相应的r77 rootkit安装包并且下载。**



安装包如下所示：

其中install.exe



r77可以直接使用单独的“Install.exe”进行安装，安装工具会将r77服务在用户登录之前开启，后台进程会向所有当前正在运行以及后续生成的进程中注入命令。这里需要使用两个进程来分别注入32位和64位进程，这两个进程都可以使用配置系统和PID来进行隐藏。

“Uninstall.exe”程序负责将r77从系统中卸载掉，并解除Rootkit跟所有进程的绑定关系。

安装程序为32位和64位r77服务创建两个计划任务。计划任务确实需要存储名为$77svc32.job和$77svc64.job的文件，这是无文件概念的唯一例外。但是，一旦Rootkit运行，计划任务也会通过前缀隐藏。

计划任务将使用下列命令开启“powershell.exe”：

[Reflection.Assembly]::Load([Microsoft.Win32.Registry]::LocalMachine.OpenSubkey('SOFTWARE').GetValue('$77stager')).EntryPoint.Invoke($Null,$Null)

该命令是内联的，不需要.ps1脚本。这里，使用PowerShell的.NET Framework功能从注册表加载C#可执行文件并在内存中执行。由于命令行的最大长度为260（MAX\_PATH），因此只有足够的空间执行简单的Assembly.Load().EntryPoint.Invoke()。

执行的C#代码为stager，它将会使用Process Hollowing技术创建r77服务进程。r77服务是一个本地可执行文件，分别以32位和64位架构继续宁编译。父进程被设置为了winlogon.exe以增加欺骗性（模糊性）。另外，这两个进程被ID隐藏，在任务管理器中不可见。

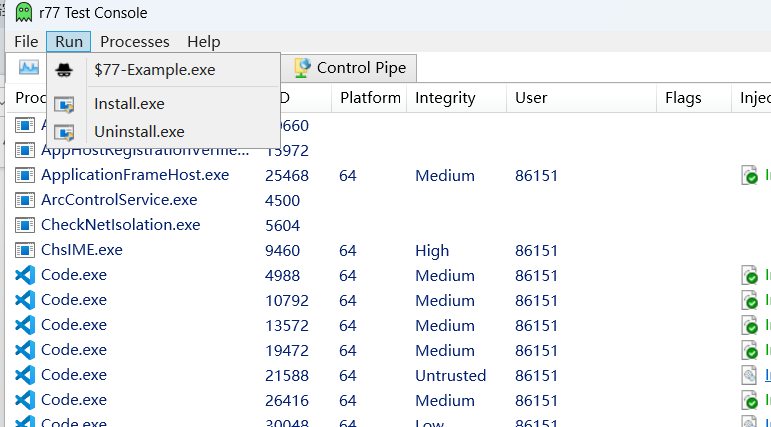
测试控制台可以用来向单独进程注入r77，或接触进程跟Rootkit的绑定关系

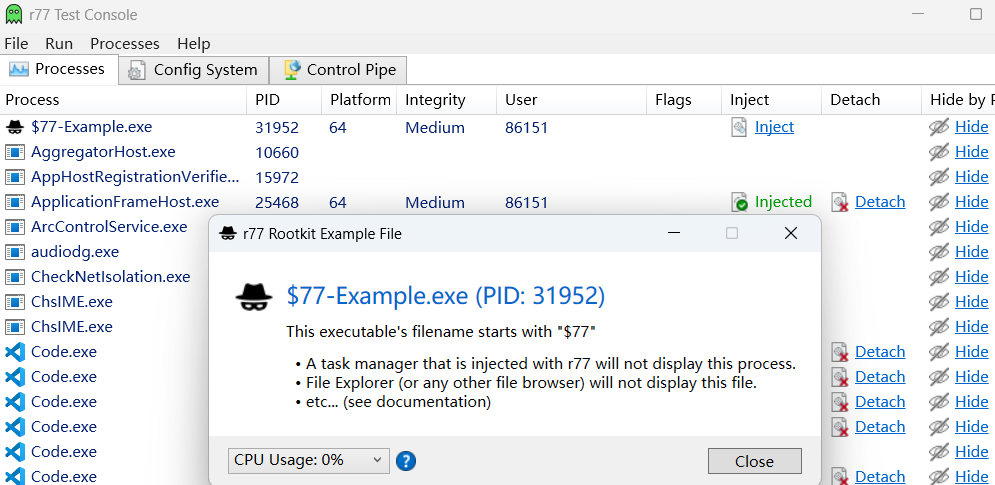
**进程隐藏**

R77的进程隐藏功能是一种用于使特定进程在操作系统的常规监视工具（如任务管理器）中不可见的技术。这种隐藏技术主要基于修改操作系统的行为来阻止对特定进程的检测。

示例进程R77为我们提供了一个可执行文件 $77-Example.exe ，这个文件以$77为开头。

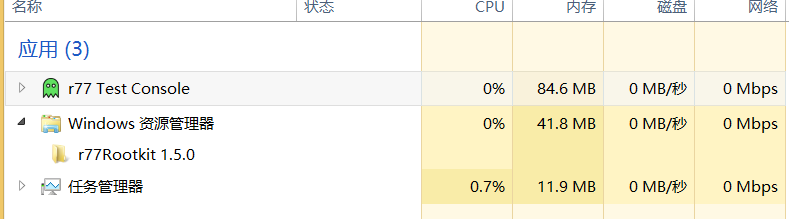
然后我们运行它：





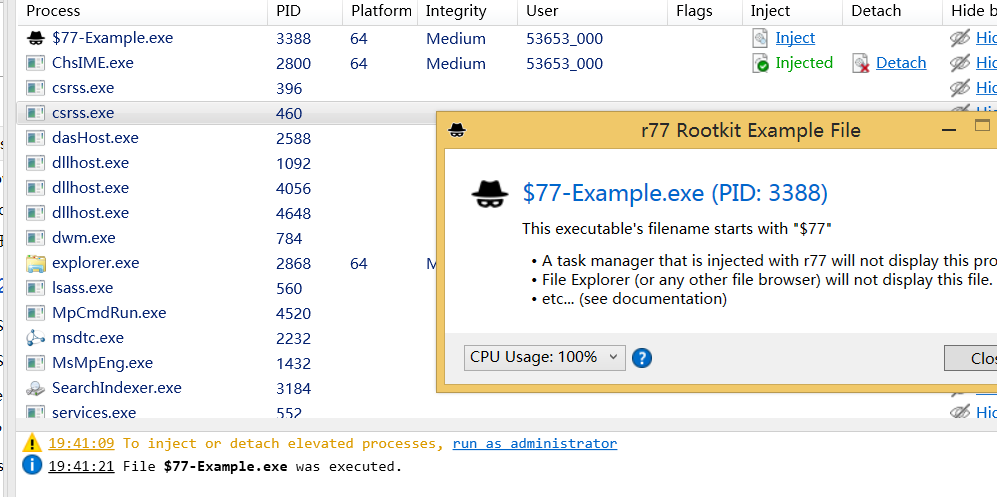
这里我们设置高cpu占有率，可以在任务管理器里看到改程序占据大量进程，并且电脑的风扇正在高速运转。

任意进程隐藏：此时运行 Install.exe ，重新打开任务管理器，同时保持这个exe运行，发现找不到这个进程了：



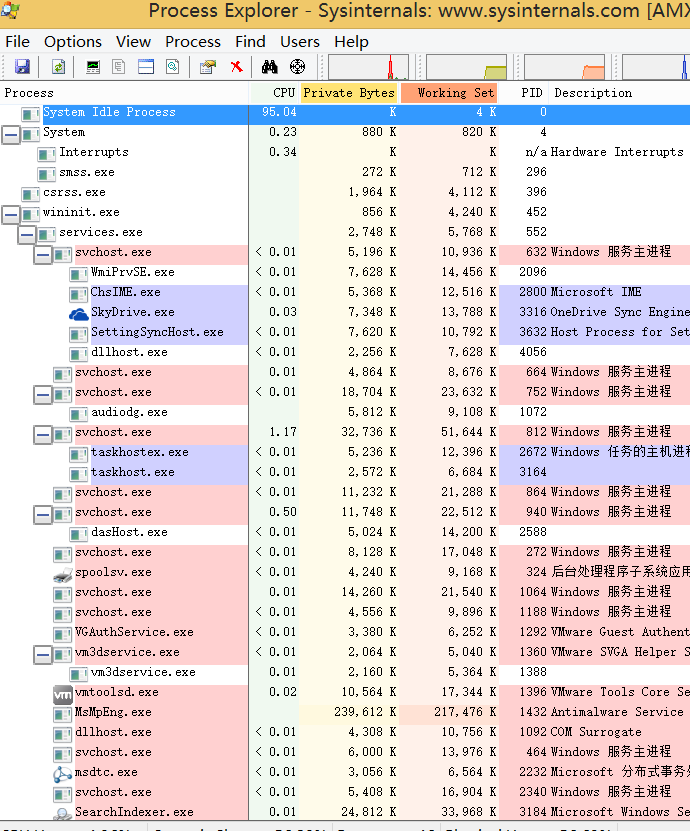
仍然没有找到这个进程，证明其确实被隐藏了。

实际上，R77默认隐藏了$77开头的进程，但其他进程也可以进行针对性隐藏。

我们运行 TestConsole.exe ，可以看到进程列表，包括被隐藏的进程：

在上图，为了证明这个程序仍然在运行，只是被隐藏了，我将CPU占用率调为100%。可以看到尽管CPU占用已经满了，但并没有显示这个进程的存在。

我们打开Process Explorer来查看进程：



随便再找一个任意进程名的进程，选取下面的进程：



点击右边对应的Hide操作：



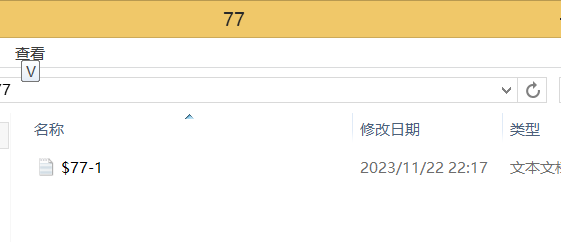


再次打开任务管理器查看，该进程已经“消失”，同时右边的Hide选项也已经变成了Unhide:

4.3 文件隐藏

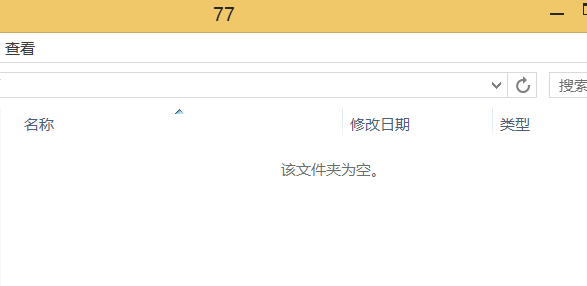
R77的文件隐藏功能是一种技术，用于使特定文件或目录在操作系统的标准文件浏览工具（例如Windows资源管理器）中不可见。这通常是通过拦截和修改系统级别的文件系统调用来实现的。

先执行uninstall:



在一个文件夹里面放入我想要隐藏的文件和文件夹：

然后执行Install操作，刷新文件夹：



显示“此文件夹为空”，文件已经被隐藏成功。

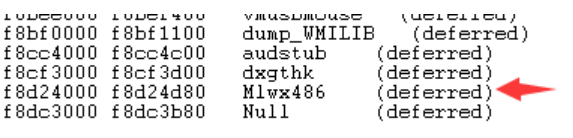
回到lab10-02.exe

这里需要我们寻找lab10-02.exe当中提到的mlwx486.sys

这里详细分析我们如何从SSDT当中查找到了相应的函数

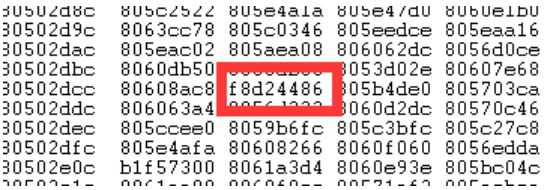
查看SSDT的修改项

使用lm命令列举加载的所有模块，可以看到Mlwx486已经被加载到内存中了，但是并没有在硬盘上显 示，说明这可能是一个Rootkit。

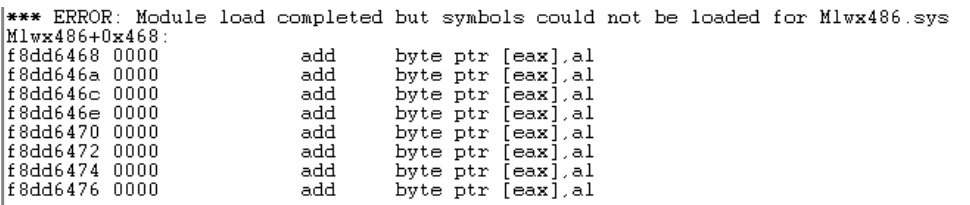


接下来，使用命令dd dwo(KeServiceDescriptorTable) L100检查SSDT的修改情况，可以看到其中有一条 记录在Mlwx486.sys的范围内。

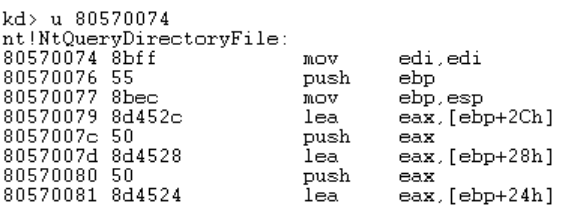
其中这个地址可能是我们需要寻找的目标函数所在的地址。



使用命令u f8d24486查看该地址，这是一个Mlwx486里面自带的函数，并不能提供什么有效信息。

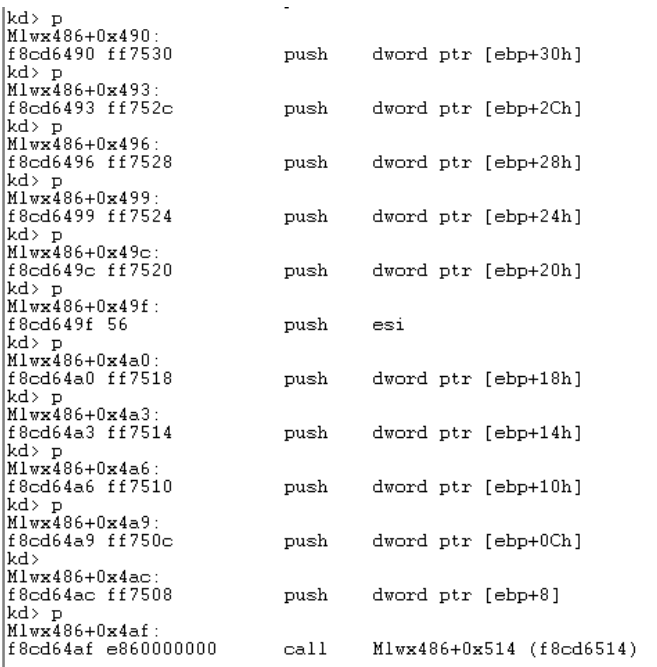


于是通过快照将虚拟机恢复到安装Rootkit之前的状态，再次使用命令dd dwo(KeServiceDescriptorTable) L100，找到修改前的地址为 80570074，于是使用命令u 80570074查看被覆盖的是哪个函数，是 NtOueryDirectoryFile，这是一个提取文件和目录信息的通用函数。Rootkit可以通过挂钩这个函数来隐藏文 件，下面我们开始分析这个函数做了什么。

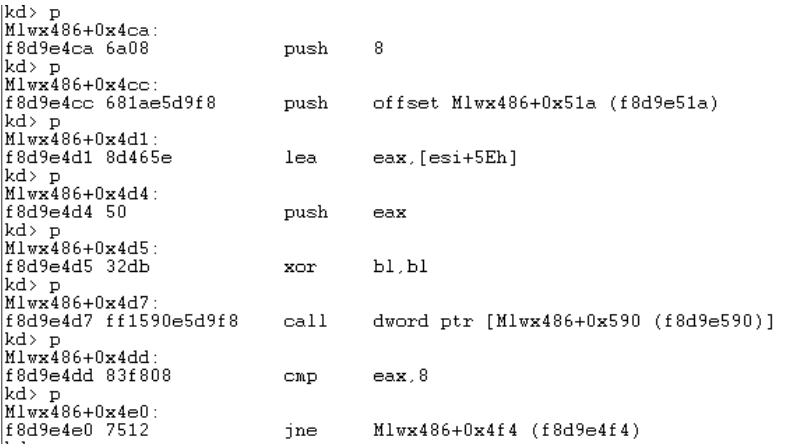


我们现在这里设置一个断点bp f8d24486，然后回到虚拟机中运行程序，命中断点后又回到WinDbg开始单步调试。（下面由于重启过虚拟机，改变了地址）

这里不断的使用p查找附近的汇编代码，最终找到计算的偏移量。



到这里，调用了函数Miwx486+0x514，它的第8个参数FileInformationClass的值是3，如果不是3，便会返 回NtQueryDirectoryFile的原始返回值。此外，挂钩函数也会检查NtQueryDirectoryFile的返回值与第9个 参数值ReturnSingleEntry。当参数不符合要求时，它的功能会与原始函数一样，只有当参数满足要求时才会 修改返回值。 接下来我们使用命令bp f8cd6486 ".if dwo(esp+0x24)==0 {} .else {gc}"在挂钩函数上设置一个条件断 点，仅当参数满足ReturnSingleEntry为0时这个断点才被触发。然后使用命令dir C:\WINDOWS\system32来 列出文件夹C:\WINDOWS\system32这个文件夹下面的所有文件和文件夹，发现这个条件断点被命中了。 之后，它调用了0xf8d9e590处的函数，IDA将其标注为RtlCompareMemory，查看它的参数，它要将eax要和字 符串"Mlwx"进行比较，比较的最大长度为8。而eax中存储的是[esi+5Eh]，在前面我们已经看到esi是 FileInformation，而且这个值是为3，具体意义是FileBothDirectoryInformation，偏移量Ox5E是结构 FileName的起始地址，这段代码用于比较每个文件的文件名开头的四个字节是否是"Mlwx"



使用命令db esi+5eh查看存储在esi+5eh的内容，这里是xs.l。如果传入的FileName和Mlwx不相等，函数直 接就退出了。

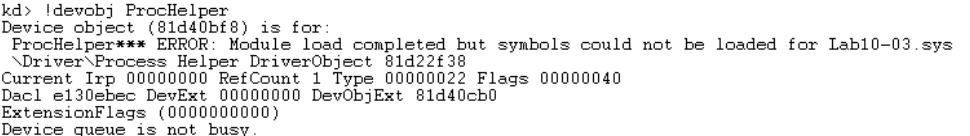
现在我们要分析它是如何修改NtQueryDirectoryFile的返回值然后隐藏Mlwx486.sys文件的。 NtQueryDirectoryFile的返回值FILE\_BOTH\_DIR\_INFORMATION结构是由一系列 FILE\_BOTH\_DIR\_INFORMATION结构串联而成的，如果RtlCompareMemory返回值是8的话，即找到了要隐藏 的"Mlwx"开头的文件，就会通过两次指令操作把指针往后移动，抹除了Mlwx486.sys文件的 FILE\_BOTH\_DIR\_INFORMATION结构，达到隐藏文件的目的。

而后再次运行 kd> dps nt!KiServiceTable l 100 ，能够看出服务描述符表已被修改

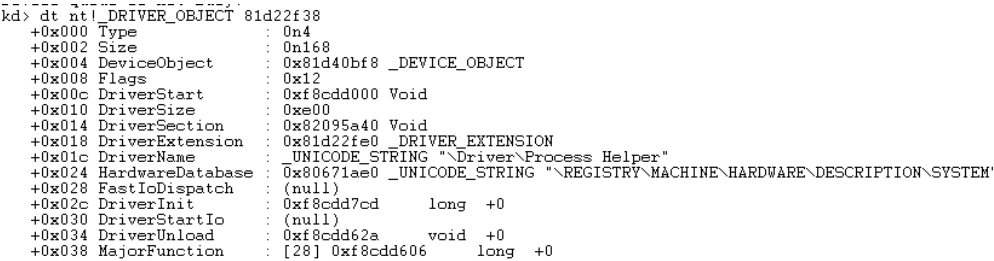


我们连接上WinDbg进行内核调试。

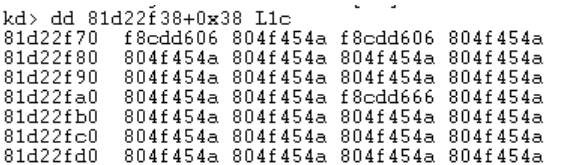
首先使用命令!devobj ProcHelper，查看到这个设备存储的位置是81d22f38。



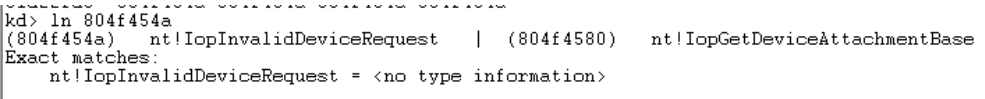
然后使用命令dt nt!\_DRIVER\_OBJECT 81d22f38查看标注的驱动对象，其中，DriverInit是驱动初始化的操 作地址，DriverUnload是驱动卸载时候的操作地址，我们刚刚在IDA分析的时候看到过这个函数。



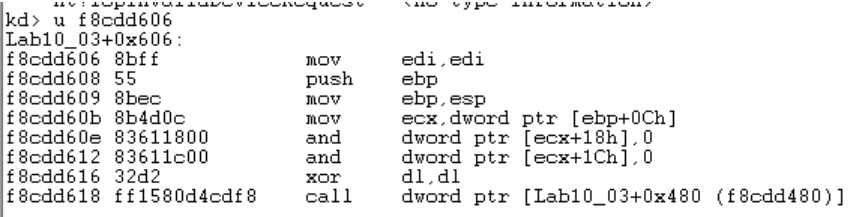
接下来，我们使用命令dd 81d22f38+0x38 L1c命令查看主函数表中表项：



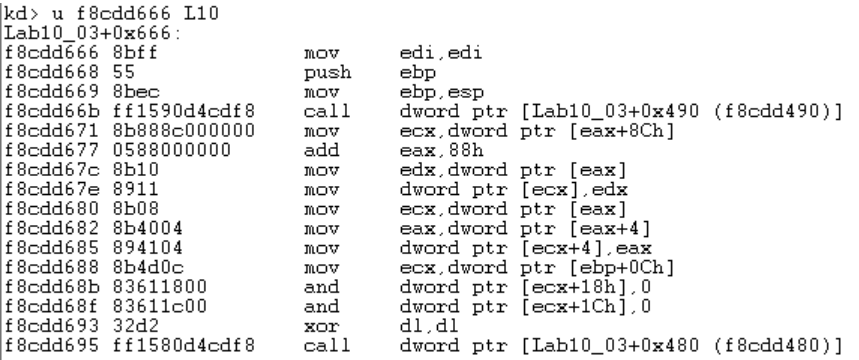
可以看到表中大多数函数都是804f454a，我们使用命令ln 804f454a看看这个函数做了什么，可以看到，这个 函数被命名为IopInvalidDeviceRequest，从字面意思看是处理驱动无法处理的非法请求的。



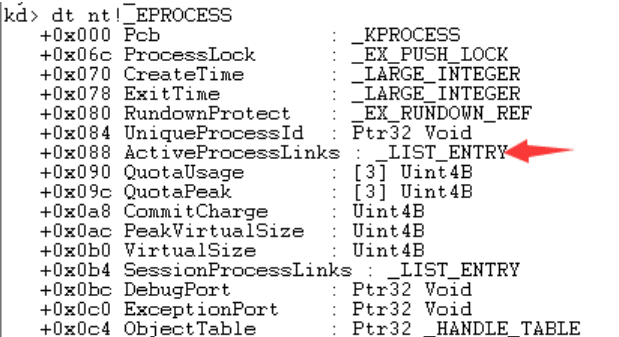
此外，主函数表中还有三个不同于804f454a的函数，分别是f8cdd606、f8cddd606和f8cdd666，它们在 896587e0+0x38处的偏移量分别是0，2和0xe，对应着Create、Close和DeviceIoControl。进入这两个（前 两个相同）位置查看具体的代码。 在f8cdd606处，看到它只调用了一个函数，指向的是804ef1b0这个地址，使用指令ln 804ef1b0，看到这个 函数是IofCompleteRequest，用于指示调用者已完成给定I/O请求的所有处理，并将给定的IRP返回给I/O管理 器，意味着告诉操作系统请求这个驱动成功了，但是没做什么其他的操作。



在f8cdd666处看到该函数在初始化栈操作之后，首先调用了一个在dword ptr [Lab10\_03+0x490]的指针函数， 指向的是804ef608这个地址，使用指令ln 804ef1b0，看到这个函数是nt!IoGetCurrentProcess，用于返回 一个指向当前进程的指针。也就是这个代码会获得一个当前进程的指针，操纵当前进程的PEB。



之后，这个函数访问偏移量0x88处的数据和偏移量0x8C处数据。我们使用dt命令发现这两处数据是PEB结构中 的\_LIST\_ENTRY。这个结构用于描述双向链接列表中的条目或充当此列表的头部，该程序不仅读取LIST\_ENTRY 结构，还会修改它，通过一系列对指针的操作修改PEB，使得没有任何的指针将指向他的LIST\_ENTRY结构，把 自己的进程隐藏起来。



1. **实验结论及心得体会**

**1. 技术验证：本次实验成功验证了R77工具在隐藏进程、文件、注册表项以及TCP/UDP连接**

**方面的有效性。通过实际操作，观察到被隐藏的对象在常规工具中不再可见，证实了R77在**

**系统级别进行拦截和修改操作的能力。**

**2. 功能特性：R77通过拦截系统级调用和修改操作系统的内部数据结构，实现了对进程、文件**

**和网络活动的隐藏。这一过程不需要修改被隐藏对象的物理状态或数据，显示出高度的隐蔽**

**性和灵活性。**

**3. 安全和隐私考量：尽管R77为系统管理和隐私保护提供了强大工具，但同时也揭示了潜在的**

**安全风险。恶意软件可能利用类似技术进行隐蔽活动，对用户和企业造成威胁。**

**在SSDT对于rootkit的研究当中，我们不断的跟踪了其中的一部分环境，并且在环境当中跟随相应的位置找到了rootkit。**

**大多数根工具包修改操作系统内核最流行的方法：系统服务描述符表（SSDT）挂钩。**

**Rootkit更改了SSDT中的值，以便调用Rootkit代码，而不是预期的函数。**

**对于rootkit的研究让我们可以寻找程序当中更多的变化，以及SSDT的控制原理。**