# Hide Module From The LDR Lists

**In this piece we are going to handle the 3 lists linking the modules per process, in order to hide a chosen DLL.**

## Before we start

This paper describes some structures and mechanisms that are specific to the OS and version of the system, and therefore are open to changes.

The work discussed here was performed on a Windows 7 32-bit machine.

## Background

- Every process on a Windows system has a **PEB**[1] – Process Environment Block - that contains information relevant to the process. One of the fields in the PEB is the Ldr, which is a **PEB_LDR_DATA**[2] structure, and contains a link to a doubly-linked list, in which every node represents information about the module that is loaded to the process.
- Every node in the list is an **LDR_DATA_TABLE_ENTRY**[3] structure.
- In fact, there isn't only one list that links all of the LDR_DATA_TABLE_ENTRY nodes, but **3 different lists** (in Win7), all of the lists are made of the same nodes, but the order of the nodes is different. Let's examine the LDR_DATA_TABLE_ENTRY:

[1] https://msdn.microsoft.com/en-us/windows/desktop/aa813706
[2] https://docs.microsoft.com/en-us/windows/win32/api/winternl/ns-winternl-peb_ldr_data
[3] See Microsoft description on the PEB_LDR_DATA page ^

```
typedef struct _LDR_DATA_TABLE_ENTRY {

        LIST_ENTRY InLoadOrderLinks;

        LIST_ENTRY InMemoryOrderLinks;

        LIST_ENTRY InInitializationOrderLinks;

        PVOID DllBase;

        PVOID Reserved2[2];

        UNICODE_STRING FullDllName;

        BYTE Reserved4[4];

        PVOID ShortDllName;

        PVOID Reserved5[3];

        union {

                ULONG CheckSum;

                PVOID Reserved6;

        } DUMMYUNIONNAME;

        ULONG TimeDateStamp;

} LDR_DATA_TABLE_ENTRY, * PLDR_DATA_TABLE_ENTRY;
```

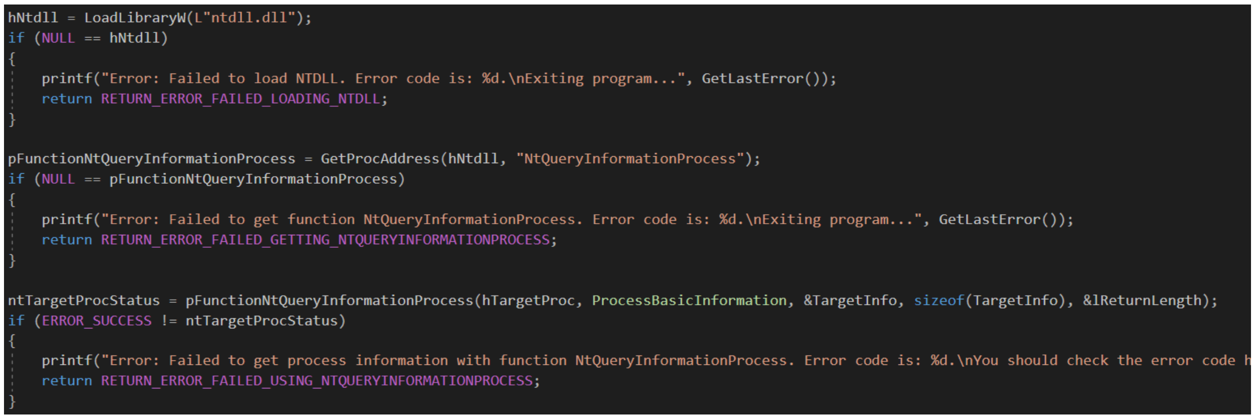
- We can see that the node holds 3 different **LIST_ENTRY**[4] structures, each of them links to a different list. The 3 lists are: InLoadOrderLinks, InMemoryOrderLinks, InInitializationOrderLinks.
- In our case, we want to delete one of the loaded modules from all of the lists, meaning, hide the module.
- The code we are about to write will get one argument that is the name of the DLL we wish to hide.

## Code

Basically, all we need to do is find the right list, and then iterate through it to find the desired module, then delete it. Very simple.

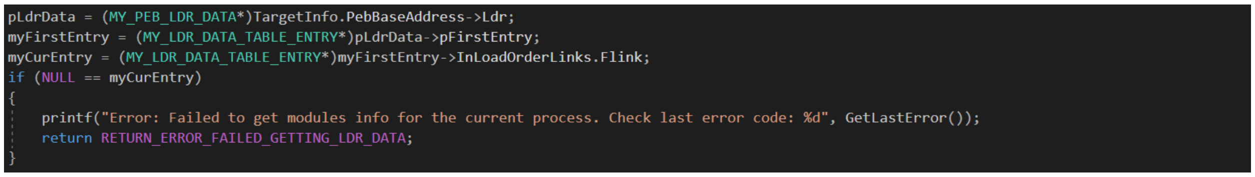
[4] https://docs.microsoft.com/en-us/windows/win32/api/ntdef/ns-ntdef-list_entry

As explained before, for accessing the list we need the PEB, which we will get by calling **NtQueryInformationProcess**[5] function. As mentioned in the MSDN docs of this function, it is internal to the OS, therefore we are going to have to access it through run-time dynamic linking[6].



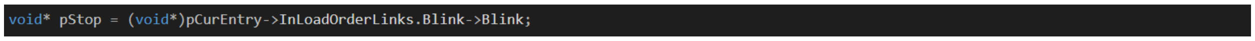
```
hNtdll = LoadLibraryW(L"ntdll.dll");
if (NULL == hNtdll)
{
    printf("Error: Failed to load NTDLL. Error code is: %d.\nExiting program...", GetLastError());
    return RETURN_ERROR_FAILED_LOADING_NTDLL;
}

pFunctionNtQueryInformationProcess = GetProcAddress(hNtdll, "NtQueryInformationProcess");
if (NULL == pFunctionNtQueryInformationProcess)
{
    printf("Error: Failed to get function NtQueryInformationProcess. Error code is: %d.\nExiting program...", GetLastError());
    return RETURN_ERROR_FAILED_GETTING_NTQUERYINFORMATIONPROCESS;
}

ntTargetProcStatus = pFunctionNtQueryInformationProcess(hTargetProc, ProcessBasicInformation, &TargetInfo, sizeof(TargetInfo), &lReturnLength);
if (ERROR_SUCCESS != ntTargetProcStatus)
{
    printf("Error: Failed to get process information with function NtQueryInformationProcess. Error code is: %d.\nYou should check the error code h
    return RETURN_ERROR_FAILED_USING_NTQUERYINFORMATIONPROCESS;
}
```

Now we can access the PebBaseAddress, then the Ldr, then the actual lists. Note that the first module directly linked from the Ldr is actually the process itself (rather than a DLL loaded to it).



```
pLdrData = (MY_PEB_LDR_DATA*)TargetInfo.PebBaseAddress->Ldr;
myFirstEntry = (MY_LDR_DATA_TABLE_ENTRY*)pLdrData->pFirstEntry;
myCurEntry = (MY_LDR_DATA_TABLE_ENTRY*)myFirstEntry->InLoadOrderLinks.Flink;
if (NULL == myCurEntry)
{
    printf("Error: Failed to get modules info for the current process. Check last error code: %d", GetLastError());
    return RETURN_ERROR_FAILED_GETTING_LDR_DATA;
}
```

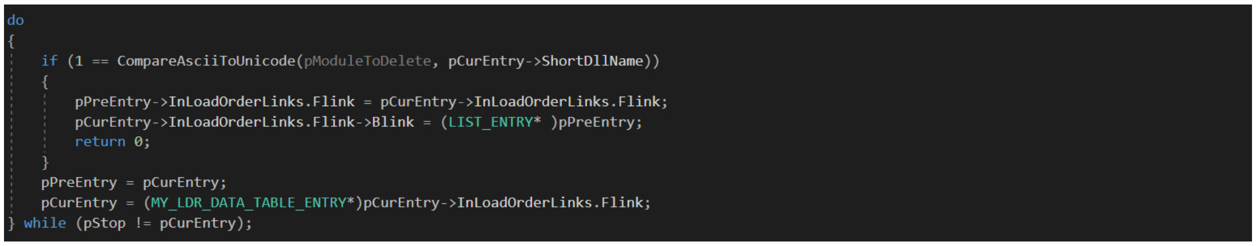
At this point we need to save the address of the Ldr itself, so we won't try to iterate it as part of the list. This is how we will know that we finished iterating through the entire list.



```
void* pStop = (void*)pCurEntry->InLoadOrderLinks.Blink->Blink;
```

For deleting the desired node we have to iterate all the 3 of them. For each list it looks like this:

[5] https://docs.microsoft.com/en-us/windows/win32/api/winternl/nf-winternl-ntqueryinformationprocess
[6] https://docs.microsoft.com/en-us/windows/win32/dlls/using-run-time-dynamic-linking

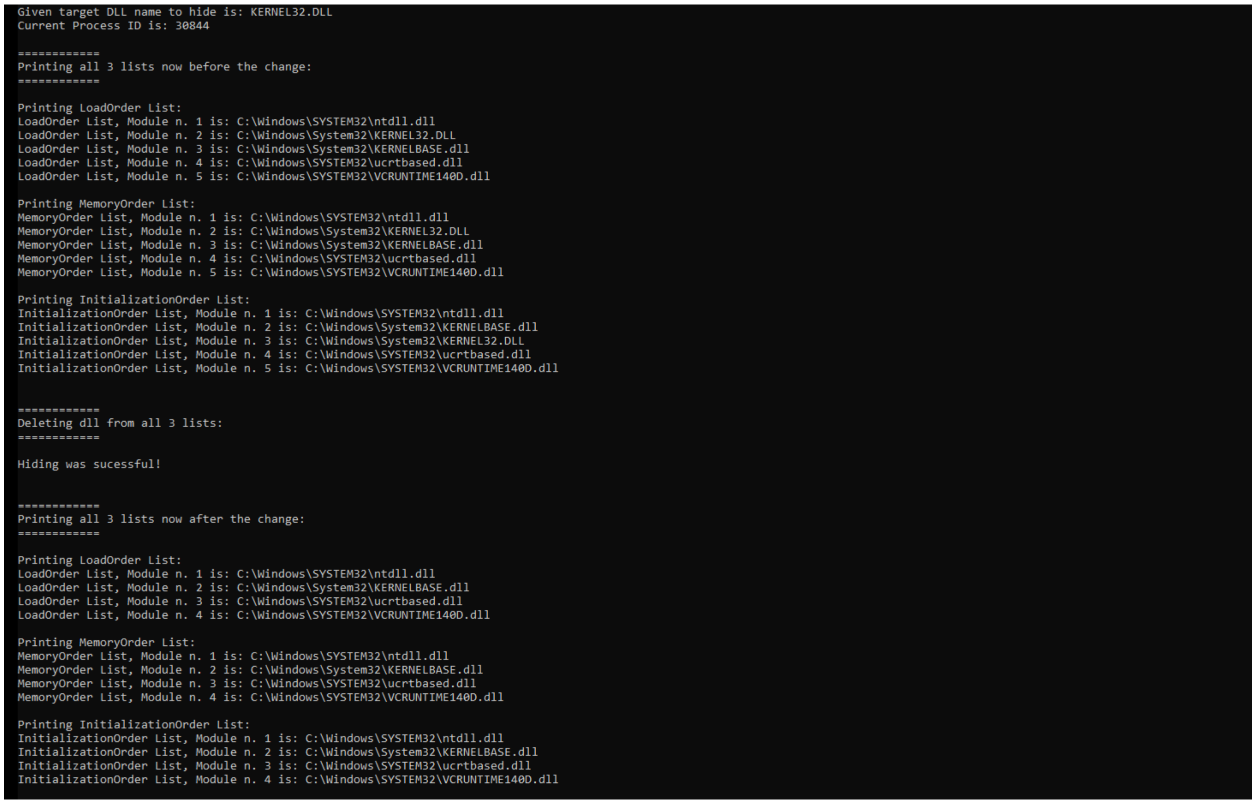


```
do
{
    if (1 == CompareAsciiToUnicode(pModuleToDelete, pCurEntry->ShortDllName))
    {
        pPreEntry->InLoadOrderLinks.Flink = pCurEntry->InLoadOrderLinks.Flink;
        pCurEntry->InLoadOrderLinks.Flink->Blink = (LIST_ENTRY* )pPreEntry;
        return 0;
    }
    pPreEntry = pCurEntry;
    pCurEntry = (MY_LDR_DATA_TABLE_ENTRY*)pCurEntry->InLoadOrderLinks.Flink;
} while (pStop != pCurEntry);
```

Of course, for each list we will access a bit different offsets (see the full code attached).

That’s it! Simple.

Now we add some printing before and after the change so we can easily spot the difference. In this case we tried to hide “KERNEL32.DLL” from the current process:



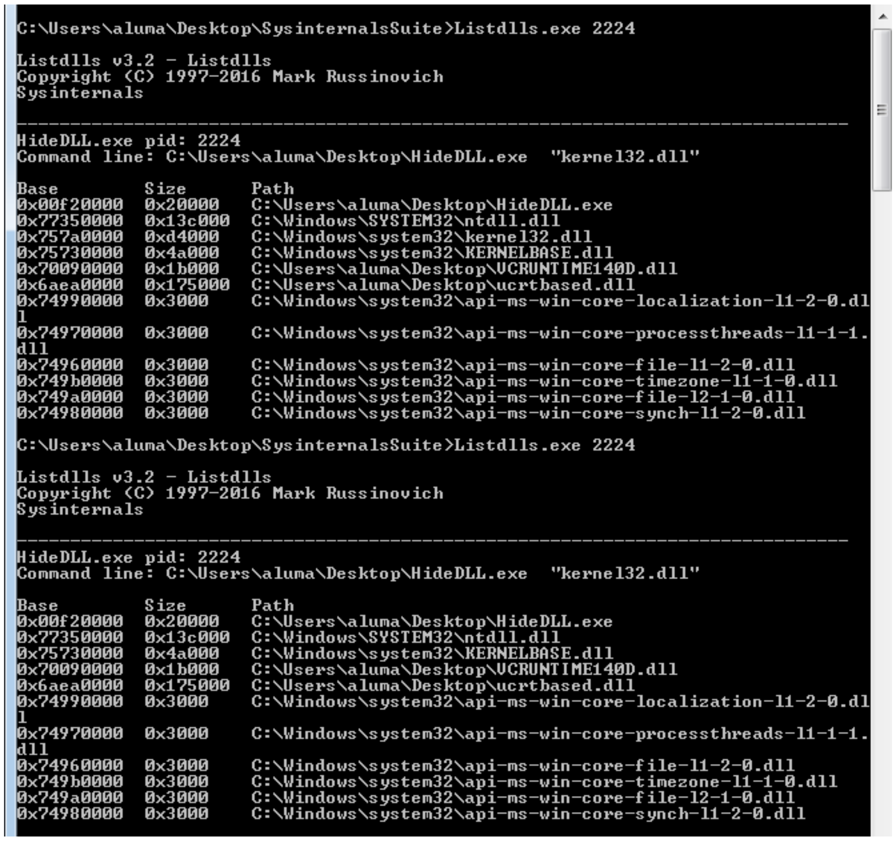
```
Given target DLL name to hide is: KERNEL32.DLL
Current Process ID is: 30844

============
Printing all 3 lists now before the change:
============

Printing LoadOrder List:
LoadOrder List, Module n. 1 is: C:\Windows\SYSTEM32\ntdll.dll
LoadOrder List, Module n. 2 is: C:\Windows\System32\KERNEL32.DLL
LoadOrder List, Module n. 3 is: C:\Windows\System32\KERNELBASE.dll
LoadOrder List, Module n. 4 is: C:\Windows\SYSTEM32\ucrtbased.dll
LoadOrder List, Module n. 5 is: C:\Windows\SYSTEM32\VCRUNTIME140D.dll

Printing MemoryOrder List:
MemoryOrder List, Module n. 1 is: C:\Windows\SYSTEM32\ntdll.dll
MemoryOrder List, Module n. 2 is: C:\Windows\System32\KERNEL32.DLL
MemoryOrder List, Module n. 3 is: C:\Windows\System32\KERNELBASE.dll
MemoryOrder List, Module n. 4 is: C:\Windows\SYSTEM32\ucrtbased.dll
MemoryOrder List, Module n. 5 is: C:\Windows\SYSTEM32\VCRUNTIME140D.dll

Printing InitializationOrder List:
InitializationOrder List, Module n. 1 is: C:\Windows\SYSTEM32\ntdll.dll
InitializationOrder List, Module n. 2 is: C:\Windows\System32\KERNELBASE.dll
InitializationOrder List, Module n. 3 is: C:\Windows\System32\KERNEL32.DLL
InitializationOrder List, Module n. 4 is: C:\Windows\SYSTEM32\ucrtbased.dll
InitializationOrder List, Module n. 5 is: C:\Windows\SYSTEM32\VCRUNTIME140D.dll


============
Deleting dll from all 3 lists:
============

Hiding was sucessful!


============
Printing all 3 lists now after the change:
============

Printing LoadOrder List:
LoadOrder List, Module n. 1 is: C:\Windows\SYSTEM32\ntdll.dll
LoadOrder List, Module n. 2 is: C:\Windows\System32\KERNELBASE.dll
LoadOrder List, Module n. 3 is: C:\Windows\SYSTEM32\ucrtbased.dll
LoadOrder List, Module n. 4 is: C:\Windows\SYSTEM32\VCRUNTIME140D.dll

Printing MemoryOrder List:
MemoryOrder List, Module n. 1 is: C:\Windows\SYSTEM32\ntdll.dll
MemoryOrder List, Module n. 2 is: C:\Windows\System32\KERNELBASE.dll
MemoryOrder List, Module n. 3 is: C:\Windows\SYSTEM32\ucrtbased.dll
MemoryOrder List, Module n. 4 is: C:\Windows\SYSTEM32\VCRUNTIME140D.dll

Printing InitializationOrder List:
InitializationOrder List, Module n. 1 is: C:\Windows\SYSTEM32\ntdll.dll
InitializationOrder List, Module n. 2 is: C:\Windows\System32\KERNELBASE.dll
InitializationOrder List, Module n. 3 is: C:\Windows\SYSTEM32\ucrtbased.dll
InitializationOrder List, Module n. 4 is: C:\Windows\SYSTEM32\VCRUNTIME140D.dll
```

Great, looks like it works.

In addition to us checking ourselves, we can use the Sysinternals tool ListDLLs.exe[7] that reports the modules loaded into process, and see that it works on it too. We

[7] https://docs.microsoft.com/en-us/sysinternals/downloads/listdlls

run the ListDLLs.exe tool twice, once before the hiding and one after. The DLL for hiding is “kernel32.dll”:



```
C:\Users\aluma\Desktop\SysinternalsSuite>Listdlls.exe 2224

Listdlls v3.2 - Listdlls
Copyright (C) 1997-2016 Mark Russinovich
Sysinternals

------------------------------------------------------------------------------
HideDLL.exe pid: 2224
Command line: C:\Users\aluma\Desktop\HideDLL.exe  "kernel32.dll"

Base        Size      Path
0x00f20000  0x20000   C:\Users\aluma\Desktop\HideDLL.exe
0x77350000  0x13c000  C:\Windows\SYSTEM32\ntdll.dll
0x757a0000  0xd4000   C:\Windows\system32\kernel32.dll
0x75730000  0x4a000   C:\Windows\system32\KERNELBASE.dll
0x70090000  0x1b000   C:\Users\aluma\Desktop\VCRUNTIME140D.dll
0x6aea0000  0x175000  C:\Users\aluma\Desktop\ucrtbased.dll
0x74990000  0x3000    C:\Windows\system32\api-ms-win-core-localization-l1-2-0.dl
l
0x74970000  0x3000    C:\Windows\system32\api-ms-win-core-processthreads-l1-1-1.
dll
0x74960000  0x3000    C:\Windows\system32\api-ms-win-core-file-l1-2-0.dll
0x749b0000  0x3000    C:\Windows\system32\api-ms-win-core-timezone-l1-1-0.dll
0x749a0000  0x3000    C:\Windows\system32\api-ms-win-core-file-l2-1-0.dll
0x74980000  0x3000    C:\Windows\system32\api-ms-win-core-synch-l1-2-0.dll

C:\Users\aluma\Desktop\SysinternalsSuite>Listdlls.exe 2224

Listdlls v3.2 - Listdlls
Copyright (C) 1997-2016 Mark Russinovich
Sysinternals

------------------------------------------------------------------------------
HideDLL.exe pid: 2224
Command line: C:\Users\aluma\Desktop\HideDLL.exe  "kernel32.dll"

Base        Size      Path
0x00f20000  0x20000   C:\Users\aluma\Desktop\HideDLL.exe
0x77350000  0x13c000  C:\Windows\SYSTEM32\ntdll.dll
0x75730000  0x4a000   C:\Windows\system32\KERNELBASE.dll
0x70090000  0x1b000   C:\Users\aluma\Desktop\VCRUNTIME140D.dll
0x6aea0000  0x175000  C:\Users\aluma\Desktop\ucrtbased.dll
0x74990000  0x3000    C:\Windows\system32\api-ms-win-core-localization-l1-2-0.dl
l
0x74970000  0x3000    C:\Windows\system32\api-ms-win-core-processthreads-l1-1-1.
dll
0x74960000  0x3000    C:\Windows\system32\api-ms-win-core-file-l1-2-0.dll
0x749b0000  0x3000    C:\Windows\system32\api-ms-win-core-timezone-l1-1-0.dll
0x749a0000  0x3000    C:\Windows\system32\api-ms-win-core-file-l2-1-0.dll
0x74980000  0x3000    C:\Windows\system32\api-ms-win-core-synch-l1-2-0.dll
```

## Conclusion

We used the structure of the Ldr in the PEB in order to hide a DLL from the relevant lists. As seen, we managed to avoid detection by the tool ListDLLs.exe, but there are other methods for detecting loaded DLLs. For example, note that since we accessed the list through the PEB, all of our actions are in the User Mode only, so we can’t use this method to fool anyone that uses the Kernel to search for loaded modules.

View the full code: link