MRLAB

Windows ToolHelp32 kitabxanasından yayınma 1. hissə - Dinamik kitabxana faylının gizlədilməsi

Azərbaycan Respublikası Xüsusi Rabitə və İnformasiya Təhlükəsizliyi Dövlət Xidməti - Kompüter İnsidentlərinə qarşı Mübarizə Mərkəzi - Malware Research Lab - S. Abasov - 17 Aprel 2022

TlHelp32 icra oluna bilən proqram təminatları haqqında informasiya əldə edilməsi üçün istifadə edilən funksiyalardan ibarət kitabxanadır. Bu kitabxana köməkliyi ilə sistemdə fəaliyyət göstərən proseslər, onları istifadə etdikləri modullar (dll), aktiv heap adresləri, icra olunan threadlar vs haqqında məlumat əldə etmək mümkündür. Daha ətraflı: https://learn.microsoft.com/en-us/windows/win32/api/_toolhelp/

Məqalədə göstərilən metod "*proof of concept*" olaraq qəbul edilməlidir. Müdaxilə edilən prosesin düzgüz işləyib işləməyəcəyi haqqında tam olaraq test fəaliyyəti həyata keçirilməmişdir.

TlHelp32 kitabxanası ilə prosesə aid modulları enumerate etmək üçün 3 funksiya istifadə edilir. **CreateToolhelp32Snapshot**, **Module32First** və **Module32Next**.

Sırası ilə ilk öncə **CreateToolhelp32Snapshot** ilə proses məlumatlarının funksiya çağrıldığı an snapshotu alınır. (snapshot əməliyyatı o deməkdir ki, funksiya çağrıldıqdan sonra proses üzərində hər hansı dəyişiklik qeydə alınmır). Daha sonra Module32First və Module32Next funksiyalarının köməkliyi ilə snapshot zamanı alınan buffer üzərində lazımı modul məlumatları götürülür və **MODULEENTR32** strukturuna yazılır.

```
HANDLE CreateToolhelp32Snapshot(
       [in] DWORD dwFlags,
       [in] DWORD th32ProcessID);
```

Funksiya snapshot götürüləcək proses id dəyərini və hansı məlumatı götürəcəyini bildirən flag qəbul edir.

Flaglardan bəziləri:

Flag	Value	Description
TH32CS_SNAPHEAPLIST	0x00000001	Heap
TH32CS_SNAPMODULE	0x00000008	Module
TH32CS_SNAPMODULE32	0x00000010	Module32 (x64)
TH32CS_SNAPPROCESS	0x00000002	Process
TH32CS_SNAPTHREAD	0x00000004	Threads

Bu məlumatların haqqında informasiya götürüldükdən sonra isə **Module32** funksiyaları ilə modulların siyahısı götürülür. Bu əməliyyatları həyata keçirən 2 ədad kiçik alət yazırıq.

- modulelist
- testapp

modulelist ilə testapp-ya aid modulların siyahısı alınacaqdır. Daha sonra isə testapp yaddaşında bəzi əməliyyatlar ilə modulelist-dən necə yayınmaq olar buna baxacağıq.

testapp.c

```c
#include <windows.h>
#include <stdio.h>

int main(void)
{
    printf("my pid: %d\n", GetCurrentProcessId());
    getchar();
}
```

modulelist.c

```c
#include <windows.h>
#include <stdio.h>
#include <tlhelp32.h>
#include <wchar.h>

int main(int argc, char* argv[])
{
    HANDLE hModuleSnapshot = INVALID_HANDLE_VALUE;
    MODULEENTRY32W mod32 = {0};
    hModuleSnapshot = CreateToolhelp32Snapshot(TH32CS_SNAPMODULE, atol(argv[1]));
    if ( hModuleSnapshot == INVALID_HANDLE_VALUE)
    {
        printf("Error CreateToolhelp32Snapshot");
        return -1;
    }
    mod32.dwSize = sizeof(MODULEENTRY32W);

    if (!Module32FirstW(hModuleSnapshot, &mod32))
```

```
    {
        printf("Error Module32FirstW");
        return -1;
    }
    do
    {
        wprintf(L"%s\n", mod32.szModule);
    }while(Module32NextW(hModuleSnapshot, &mod32));
    return 1;
}
```

testapp.c istifadəçidən(stdin) hər hansı key daxil edilməsini istəyir və o zamana qədər gözləmə rejimində qalır. **modulelist.c** isə özünə commandline üzərindən gələn proses modullarını (syswow64 daxil deyil) list edir. Burada önəmli olan snapshot məlumatlarının haradan götürüldüyüdür. Çünki **Module32** funksiyaları snapshot (local buffer) üzərindən bu məlumatları oxuyur. **CreateToolhelp32Snapshot** funksiyası prosesə aid modulların informasiyasını prosesə aid bir başqa data struktur (PROCESS_ENVIRONMENT_BLOCK) üzərindən götürür.

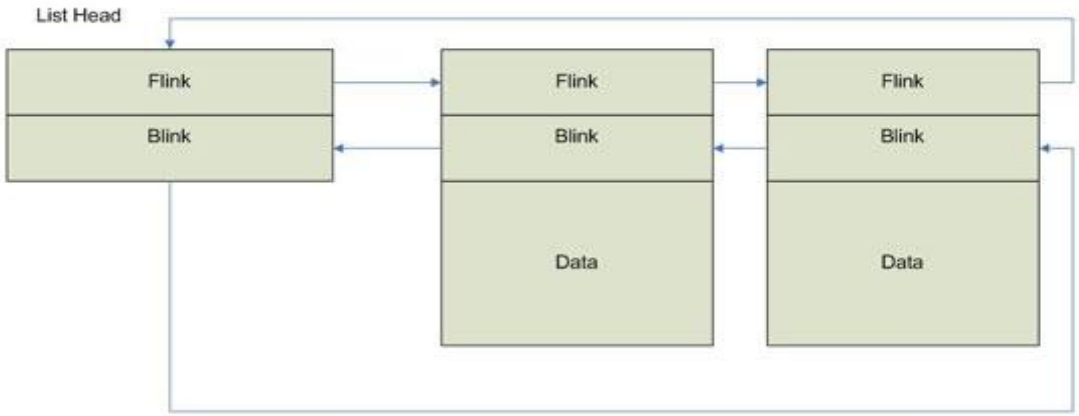
```
0:001> dt _peb
ntdll!_PEB
   +0x000 InheritedAddressSpace : UChar
   +0x001 ReadImageFileExecOptions : UChar
   +0x002 BeingDebugged    : UChar
   +0x003 BitField         : UChar
   +0x003 ImageUsesLargePages : Pos 0, 1 Bit
   +0x003 IsProtectedProcess : Pos 1, 1 Bit
   +0x003 IsImageDynamicallyRelocated : Pos 2, 1 Bit
   +0x003 SkipPatchingUser32Forwarders : Pos 3, 1 Bit
   +0x003 IsPackagedProcess : Pos 4, 1 Bit
   +0x003 IsAppContainer   : Pos 5, 1 Bit
   +0x003 IsProtectedProcessLight : Pos 6, 1 Bit
   +0x003 IsLongPathAwareProcess : Pos 7, 1 Bit
   +0x004 Mutant           : Ptr32 Void
   +0x008 ImageBaseAddress : Ptr32 Void
   +0x00c Ldr              : Ptr32 _PEB_LDR_DATA
   +0x010 ProcessParameters : Ptr32 _RTL_USER_PROCESS_PARAMETERS
```

Burada bizə lazım olan növbəti struktur isə **PEB_LDR_DATA** strukturudur.

```
0:001> dt _PEB_LDR_DATA
ntdll!_PEB_LDR_DATA
   +0x000 Length          : Uint4B
   +0x004 Initialized     : UChar
   +0x008 SsHandle        : Ptr32 Void
   +0x00c InLoadOrderModuleList : _LIST_ENTRY
   +0x014 InMemoryOrderModuleList : _LIST_ENTRY
   +0x01c InInitializationOrderModuleList : _LIST_ENTRY
   +0x024 EntryInProgress  : Ptr32 Void
   +0x028 ShutdownInProgress : UChar
   +0x02c ShutdownThreadId : Ptr32 Void
```

Göy rəng ilə işarə etdiyim offsetlər modulların saxlanıldığı data strukturlardır. Bu data struktur isə **LIST_ENTRY** adı verilən və microsoft tərəfindən istifadə edilən **doubly-linked** data strukturudur.

```
0:001> dt _LIST_ENTRY
ntdll!_LIST_ENTRY
   +0x000 Flink  : Ptr32 _LIST_ENTRY
   +0x004 Blink  : Ptr32 _LIST_ENTRY
```



Flink (forward) bir növbəti listin adresini , **Blink** (backward) isə öncəki listin adresini özündə saxlayır. Buna görə doubly-linked adlandırılır. Tlhelp32 modul siyahısı alarkın məhz bu listi gəzir və modullar haqqında məlumat toplayır. **LIST_ENTRY** tək başına istifadə edilən bir data struktur deyil. **LIST_ENTRY** bir neçə data strukturu bir-birinə bağlamaq üçün istifadə edilən bir strukturdur. Burada bir birinə bağlanılan məlumatlar haqqında tam olmasada Microsoft bizə kiçik bir hint verir. The head of a doubly-linked list that contains the loaded modules for the process. Each item in the list is a pointer to an **LDR_DATA_TABLE_ENTRY** structure.

```
0:001> dt _LDR_DATA_TABLE_ENTRY
ntdll!_LDR_DATA_TABLE_ENTRY
   +0x000 InLoadOrderLinks : _LIST_ENTRY
   +0x008 InMemoryOrderLinks : _LIST_ENTRY
   +0x010 InInitializationOrderLinks : _LIST_ENTRY
   +0x018 DllBase          : Ptr32 Void
   +0x01c EntryPoint       : Ptr32 Void
   +0x020 SizeOfImage      : Uint4B
   +0x024 FullDllName      : _UNICODE_STRING
   +0x02c BaseDllName      : _UNICODE_STRING
```

Testapp üzərindən istifadə edilən modullara göz gəzdirək. İlk öncə **r $peb** əmri ilə prosesin peb adresini öyrənirik.

```
0:001> r $peb
$peb=00595000
```

Daha sonra **dt _PEB @$peb** əmri ilə prosesin environment block strukturuna baxırıq.

```
0:001> dt _PEB @$peb
ntdll!_PEB
   +0x000 InheritedAddressSpace : 0 ''
   +0x001 ReadImageFileExecOptions : 0 ''
   +0x002 BeingDebugged    : 0x1 ''
   +0x003 BitField         : 0x4 ''
   +0x003 ImageUsesLargePages : 0y0
   +0x003 IsProtectedProcess : 0y0
   +0x003 IsImageDynamicallyRelocated : 0y1
   +0x003 SkipPatchingUser32Forwarders : 0y0
   +0x003 IsPackagedProcess : 0y0
   +0x003 IsAppContainer   : 0y0
   +0x003 IsProtectedProcessLight : 0y0
   +0x003 IsLongPathAwareProcess : 0y0
   +0x004 Mutant           : 0xffffffff Void
   +0x008 ImageBaseAddress : 0x009b0000 Void
   +0x00c Ldr              : 0x77815d80 _PEB_LDR_DATA
   +0x010 ProcessParameters : 0x00af3ee8 _RTL_USER_PROCESS_PARAMETERS
```

Bundan sonra isə **dt _PEB_LDR_DATA** 0x77815d80 əmri ilə **LDR_DATA** struktuna keçid edirik.

```
0:001> dt _PEB_LDR_DATA 0x77815d80
ntdll!_PEB_LDR_DATA
   +0x000 Length           : 0x30
   +0x004 Initialized      : 0x1 ''
   +0x008 SsHandle         : (null)
   +0x00c InLoadOrderModuleList : _LIST_ENTRY [ 0xaf79c8 - 0xaf80d8 ]
   +0x014 InMemoryOrderModuleList : _LIST_ENTRY [ 0xaf79d0 - 0xaf80e0 ]
   +0x01c InInitializationOrderModuleList : _LIST_ENTRY [ 0xaf78f0 - 0xaf7d78 ]
   +0x024 EntryInProgress  : (null)
   +0x028 ShutdownInProgress : 0 ''
   +0x02c ShutdownThreadId : (null)
```

Burada 2 ədəd **LIST_ENTRY** strukturu mövcuddur. Hərbiri eyni data struktura işarə etsədə load order filter tətbiq edilib. Mənim analizlərimdə CreateToolhelp32Snapshot **InLoadOrderModuleList** üzərindən məlumatları toplayır. Buna görə məhz bu list üzərində əməliyyat aparacam. LIST_ENTRY strukturunun əslində tək başına olmadığını qeyd etmişdim. Bu strukt LDR_DATA_TABLE_ENTRY strukturnun bir hissəsidir.

```
0:001> dt _LDR_DATA_TABLE_ENTRY 0xaf79c8
ntdll!_LDR_DATA_TABLE_ENTRY
   +0x000 InLoadOrderLinks : _LIST_ENTRY [ 0xaf78e0 - 0x77815d8c ]
   +0x008 InMemoryOrderLinks : _LIST_ENTRY [ 0xaf78e8 - 0x77815d94 ]
   +0x010 InInitializationOrderLinks : _LIST_ENTRY [ 0x0 - 0x0 ]
   +0x018 DllBase          : 0x009b0000 Void
   +0x01c EntryPoint       : 0x009b12e1 Void
   +0x020 SizeOfImage      : 0x1d000
   +0x024 FullDllName      : _UNICODE_STRING "C:\Users\user1\Desktop\testapp.exe"
   +0x02c BaseDllName      : _UNICODE_STRING "testapp.exe"
```

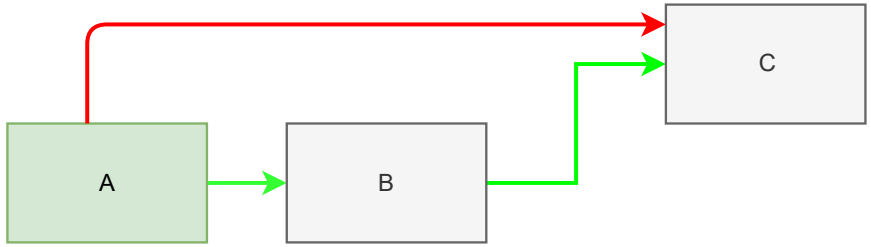
```
0:001> dt _LDR_DATA_TABLE_ENTRY 0xaf78e0
ntdll!_LDR_DATA_TABLE_ENTRY
   +0x000 InLoadOrderLinks : _LIST_ENTRY [ 0xaf7d68 - 0xaf79c8 ]
   +0x008 InMemoryOrderLinks : _LIST_ENTRY [ 0xaf7d70 - 0xaf79d0 ]
   +0x010 InInitializationOrderLinks : _LIST_ENTRY [ 0xaf80e8 - 0x77815d9c ]
   +0x018 DllBase          : 0x776f0000 Void
   +0x01c EntryPoint       : (null)
   +0x020 SizeOfImage      : 0x1a3000
   +0x024 FullDllName      : _UNICODE_STRING "C:\WINDOWS\SYSTEM32\ntdll.dll"
   +0x02c BaseDllName      : _UNICODE_STRING "ntdll.dll"
```

Modulelist alətinin verdiyi məlumatı (siyahını) xatırlayaq.

- testapp.dll
- ntdll.dll
- kernel32.dll
- kernelbase.dll

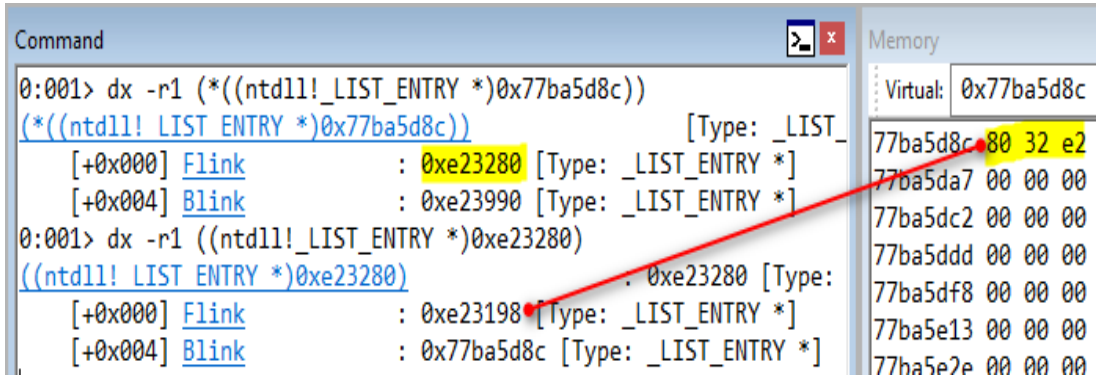
PEB_LDR_DATA strukturuna müdaxilə

Modul məlumatlarını saxlayan **LDR_DATA_TABLE_ENTRY** strukturnun **LIST_ENTRY** strukturu ilə bir-birinə necə zəncir şəklində bağlandığını anladınız (ümid edirəm). Bizə lazım olan modulu gizlətmək üçün belə bir yol izləyəcəm. **A-B-C** bir-birinə bağlıdırsa, buradan B-nı aradan götürə bilsək (yəni A-nı birbaşa C-yə bağlaya bilsək) modulumuzu gizlətmək mümkündür. Aşağıda ki, diaqrama diqqət yetirin.

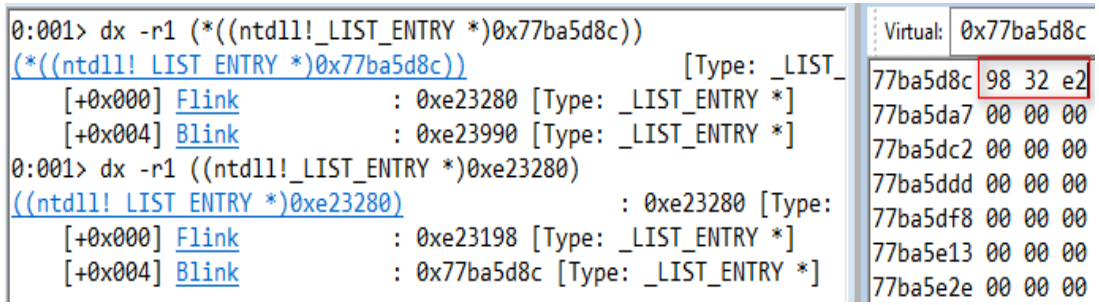


Keçirik yeniden **PEB_LDR_DATA** strukturuna. Burada **LIST_ENTRY** data strukturu üzərində dəyişikliyimizi edirik. İlk LDR_DATA_TABLE_ENTRY testapp modulunu işarə edirdi bildiyiniz kimi. Bunu özündən sonra gələn modulun **Flink** adresi ilə dəyişirik.

```
0:001> dx -r1 (*((ntdll!_LIST_ENTRY *)0x77ba5d8c))
(*((ntdll!_LIST_ENTRY *)0x77ba5d8c))                 [Type: _LIST_ENTRY]
    [+0x000] Flink            : 0xe23280 [Type: _LIST_ENTRY *]
    [+0x004] Blink            : 0xe23990 [Type: _LIST_ENTRY *]
0:001> dx -r1 ((ntdll!_LIST_ENTRY *)0xe23280)
((ntdll!_LIST_ENTRY *)0xe23280)                  : 0xe23280 [Type: _LIST_ENTRY *]
    [+0x000] Flink            : 0xe23198 [Type: _LIST_ENTRY *]
    [+0x004] Blink            : 0x77ba5d8c [Type: _LIST_ENTRY *]
```



Burada **0x77ba5d8c** adresində olan 32 bitlik adresi **0xe23198** adresi ilə dəyişirəm.



Şəkildə adres səhv qeyd edilib.

.**g(go)** əmri ilə proqram qaldığı yerdən işinə davam edir və modulelist alətinin nəticəsi.

```
C:\Users\user1\Desktop>modulelist.exe 4928
ntdll.dll
KERNEL32.DLL
KERNELBASE.dll
```

İstinadlar

https://docs.microsoft.com/en-us/windows/win32/api/winternl/ns-winternl-peb
https://docs.microsoft.com/en-us/windows-hardware/drivers/kernel/singly-and-doubly-linked-lists
https://www.geoffchappell.com/studies/windows/km/ntoskrnl/inc/api/pebteb/peb/index.htm
https://www.nirsoft.net/kernel_struct/vista/PEB_LDR_DATA.html