Protecting Stack Overflow of The Widows Xp Sp2

19-12-2004

We often need to use the Global variables and SEH while we are doing EXP towards the bug of overflow. But Now, we can't use this skill to finish our work. What's MS do in the windows XP SP2? She did a lot. For example, she encoded the Global variables.

For detail, The document include the follow:

1、 Processing the frist address of mapping PEB manager structure through random. After a while, we'll see the processing method is weak, but it is enough to forbiding EXP finished the work or working stably.
2、 Protecting TOP SEH
3、 Protecting the VEH chaining point _RtlpCalloutEntryList
4、 Protecting the cookie of stack structure

No include the content:

1、 How pass through the Protecting mechanism
2、 The detail of managing stack. Actually, it is not more changed.

==========

1、Processing the Address of PEB Through Random

In the XP system, Creating the Process is the _NtCreateProcessEx function, not the _NtCreateProcess function. the _NtCreateProcess call _PspCreateProcess@36 function mainly to finish the work of the Process created.

```
PAGE:004B4649          CALL _PspCreateProcess@36       ; PspCreateProcess(x,x,x,x,x,x,x,x,x)
;The work of the Process created is consist of setting the EPROCESS, Creating the Address area
;of the original Process, and so on. Setting PEB with CALL _MmCreatePeb.

PAGE:004B428E          push    eax
PAGE:004B428F          push    ebx
PAGE:004B4290          push    dword ptr [ebp-60h]
PAGE:004B4293          call    _MmCreateProcessAddressSpace@12 ; mCreateProcessAddressSpace(x,x,x)
PAGE:004B43E5          lea     eax, [ebx+1B0h]
PAGE:004B43EB          push    eax
PAGE:004B43EC          lea     eax, [ebp-40h]
PAGE:004B43EF          push    eax
PAGE:004B43F0          push    ebx
PAGE:004B43F1          call    _MmCreatePeb@12
               ; MmCreatePeb(x,x,x) and MmCreatePeb call by the _MiCreatePebOrTeb
PAGE:004B4A61 ; __stdcall MmCreatePeb(x,x,x)
PAGE:004B4A61 _MmCreatePeb@12 proc near
               ; CODE XREF : PspCreateProcess(x,x,x,x,x,x,x,x,x)+303 p
PAGE:004B4A61
PAGE:004B4A61 ; FUNCTION CHUNK AT PAGE:005267FF SIZE 000000DC BYTES
PAGE:004B4A61
PAGE:004B4A61           push    3Ch
PAGE:004B4A63           push    offset dword_42DAA8
PAGE:004B4A68           call    __SEH_prolog
PAGE:004B4A6D           xor     ebx, ebx
PAGE:004B4A6F           mov     [ebp-20h], ebx
PAGE:004B4A72           mov     [ebp-4Ch], ebx
PAGE:004B4A75           mov     [ebp-48h], ebx
PAGE:004B4A78           mov     [ebp-2Ch], ebx
PAGE:004B4A7B           mov     esi, [ebp+8]
PAGE:004B4A7E           push    esi
PAGE:004B4A7F           call    _KeAttachProcess@4   ; KeAttachProcess(x)
PAGE:004B4A84           push    2
PAGE:004B4A86           pop     edi
PAGE:004B4A87           push    edi
PAGE:004B4A88           push    (offset loc_4FFFFE+2)
PAGE:004B4A8D           push    1
PAGE:004B4A8F           lea     eax, [ebp-2Ch]
PAGE:004B4A92           push    eax
PAGE:004B4A93           lea     eax, [ebp-4Ch]
PAGE:004B4A96           push    eax
PAGE:004B4A97           push    ebx
PAGE:004B4A98           push    ebx
PAGE:004B4A99           lea     eax, [ebp-20h]
PAGE:004B4A9C           push    eax
PAGE:004B4A9D           push    esi
PAGE:004B4A9E           push    ds:_InitNlsSectionPointer
```

```
PAGE:004B4AA4           call    _MmMapViewOfSection@40 ; MmMapViewOfSection(x,x,x,x,x,x,x,x,x,x)
PAGE:004B4AA9           mov     [ebp-24h], eax
PAGE:004B4AAC           cmp     eax, ebx
PAGE:004B4AAE           jl      loc_5267FF
PAGE:004B4AB4           lea     eax, [ebp-1Ch]
                ; Attention the 210 value, it looks a Flag. You'll find the follow, if the
                ; value != 210, then the mapping address could not creat a random value,
                ; it is same as the former value, is always at 7FFDF000 address.
PAGE:004B4AB7           push    eax
PAGE:004B4AB8           push    210h
                ; Attention the value !
PAGE:004B4ABD           push    esi
PAGE:004B4ABE           call    _MiCreatePebOrTeb@12
        ; MiCreatePebOrTeb(x,x,x),the fact work finished by the _MiCreatePebOrTeb@12 function
PAGE:004B01AE           call    _ExAllocatePoolWithTag@12 ; ExAllocatePoolWithTag(x,x,x)
PAGE:004B01B3           mov     esi, eax
PAGE:004B01B5           test    esi, esi
PAGE:004B01B7           jz      loc_52678E
PAGE:004B01BD           mov     eax, [ebp+arg_8]
PAGE:004B01C0           mov     ecx, [ebp+arg_8]
PAGE:004B01C3           and     eax, 0FFFh
PAGE:004B01C8           neg     eax
PAGE:004B01CA           sbb     eax, eax
PAGE:004B01CC           neg     eax
PAGE:004B01CE           shr     ecx, 0Ch
PAGE:004B01FB           cmp     [ebp+arg_8], 210h
PAGE:004B0202           jz      loc_4B4A0A
            ;Here is the compartion of 210 and the pushed value, estimating the pushed value
PAGE:004B0208 loc_4B0208:         ; CODE XREF: MiCreatePebOrTeb(x,x,x)+48AD j
PAGE:004B0208           mov     edi, [ebp+arg_C]
PAGE:004B020B           mov     eax, _MmHighestUserAddress
PAGE:004B0210           push    edi
PAGE:004B0211           push    dword ptr [ebx+11Ch]
PAGE:004B0217           add     eax, 0FFFF0001h
PAGE:004B021C           push    1000h
PAGE:004B0221           push    eax
PAGE:004B0222           mov     eax, [ebp+arg_8]
PAGE:004B0225           add     eax, 0FFFh
PAGE:004B022A           and     eax, 0FFFFF000h
PAGE:004B022F           push    eax
PAGE:004B0230           call    _MiFindEmptyAddressRangeDownTree@20
                ; MiFindEmptyAddressRangeDownTree(x,x,x,x,x)
PAGE:004B0235           test    eax, eax
PAGE:004B0237           mov     [ebp+arg_C], eax
PAGE:004B023A           jl      loc_5267A5
                ; Here is the crux
PAGE:004B4A0A loc_4B4A0A:         ; CODE XREF: MiCreatePebOrTeb(x,x,x)+66 j
PAGE:004B4A0A           mov     edi, _MmHighestUserAddress
                ;it always is 7FFEFFFF
PAGE:004B4A10           lea     eax, [ebp+var_C]
PAGE:004B4A13           push    eax
PAGE:004B4A14           add     edi, 0FFFF0001h
                ; at the time, edi == 7FFE0000
PAGE:004B4A1A           call    _KeQueryTickCount@4 ; KeQueryTickCount(x)
PAGE:004B4A1F           mov     eax, [ebp+var_C]
PAGE:004B4A22           and     eax, 0Fh
                ; only get the last array value, for example, it is 0C at the time.
PAGE:004B4A25           cmp     eax, 1
                ; judging eax == 01 at the time
PAGE:004B4A28           mov     [ebp+var_C], eax
PAGE:004B4A2B           jbe     loc_4B4928
                ; if eax == 01, then jmp processing
PAGE:004B4A31 loc_4B4A31:   ; CODE XREF: MiCreatePebOrTeb(x,x,x)+4792 j
PAGE:004B4A31           shl     eax, 0Ch
PAGE:004B4A34           sub     edi, eax
PAGE:004B4A36           lea     eax, [edi+0FFFh]
PAGE:004B4A3C           push    eax
PAGE:004B4A3D           push    edi
PAGE:004B4A3E           push    ebx
PAGE:004B4A3F           mov     [ebp+var_4], edi
PAGE:004B4928 loc_4B4928:   ; CODE XREF: MiCreatePebOrTeb(x,x,x)+488F j
              ;      .if eax == 1,
              ;          mov eax,2
              ;      .endif
              ; This prevented the PEB address == 7FFDF000 at end. it is 7FFDE000
PAGE:004B4928           push    2
```

```
PAGE:004B492A           pop     eax
PAGE:004B492B           mov     [ebp+var_C], eax
PAGE:004B492E           jmp     loc_4B4A31

             ; It can't be forecasted since the KeTickCount is a counter of process
.text:0041CAA8          mov     edi, edi
.text:0041CAAA          push    ebp
.text:0041CAAB          mov     ebp, esp
.text:0041CAAD          mov     ecx, _KeTickCount.High1Time
.text:0041CAB3          mov     eax, [ebp+arg_4]
.text:0041CAB6          mov     [eax+4], ecx
.text:0041CAB9          mov     edx, _KeTickCount.LowPart
.text:0041CABF          mov     [eax], edx
```

On the above analysing, we knew: if eax is 1, 2 through the Random, All of the PEB address is 7FFDE000, this prevented that the PEB address is 7FFDF000 at all time, bring on the code which used the stack value will be false.

1	7FFDE000
2	7FFDE000
3	7FFDD000
4	7FFDC000
5	7FFDB000
6	7FFDA000
7	7FFD9000
8	7FFD8000
9	7FFD7000
A	7FFD6000
B	7FFD5000
C	7FFD4000
D	7FFD3000
E	7FFD2000
F	7FFD1000
0	7FFDE000

The above list is the PEB address in all status by any possibility, we found the probability of 7FFDE000 is the most high, == 1/8, all of others is 1/16. but, Although it is that, we also can't use the PEB address stably。