IsDebuggerPresent API

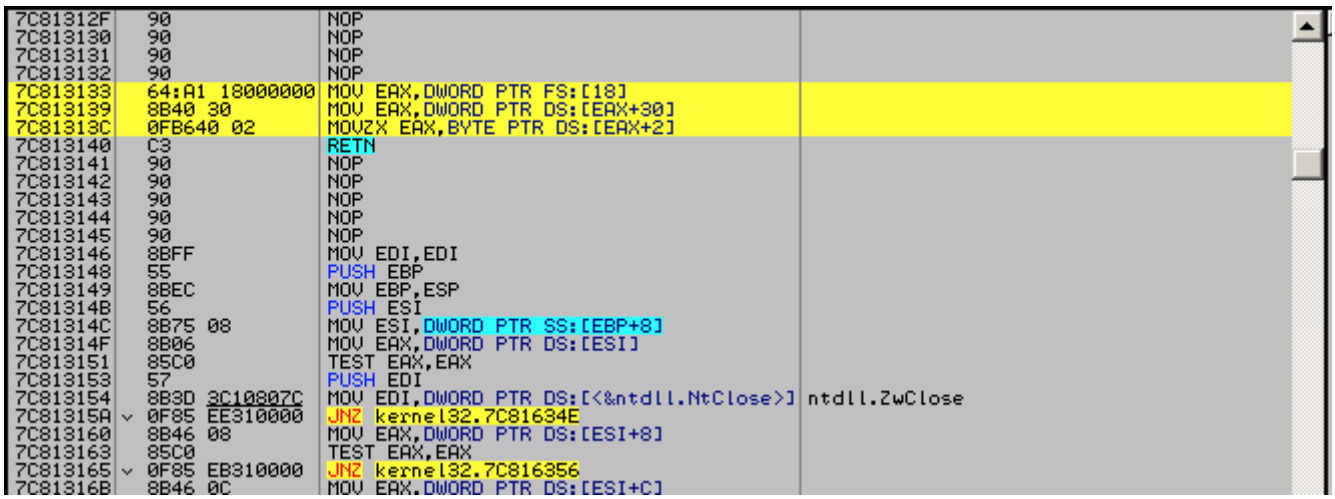
时间	作者	等级	Rank
2017-02-04 10:13:26	captain (/profile/1/)	低危	1

IsDebuggerPresent API

IsDebuggerPresent是一个windows api，我们可以用来检测程序是否被调试器所调试，下面是实例代码

```
/*
 * IsDebuggerPresent Example
 * Author: Osanda Malith Jayathissa (@OsandaMalith)
 * Website: https://osandamalith.wordpress.com
 */

#include <windows.h>

int main() {
    MessageBox(0, IsDebuggerPresent() ? "Debugger found" : "Debugger not found","Status",0x
30);
}
```

如果我们使用调试器，将会调用message api去显示debugger found这一段话。这api到底怎么工作的？我们在调试器中调试一下这个api，你将会看到他的代码。

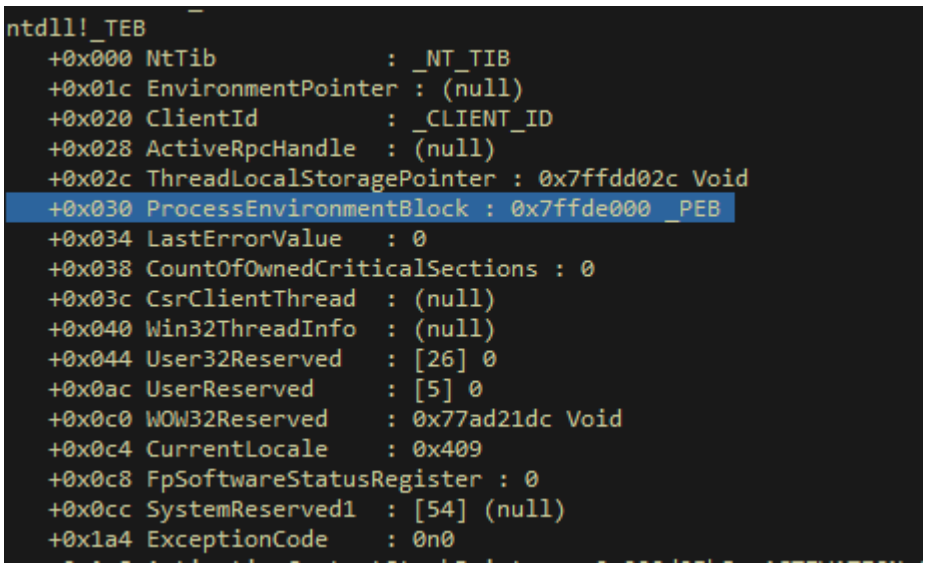



```
MOV EAX,DWORD PTR FS:[18]
MOV EAX,DWORD PTR DS:[EAX+30]
MOVZX EAX,BYTE PTR DS:[EAX+2]
```

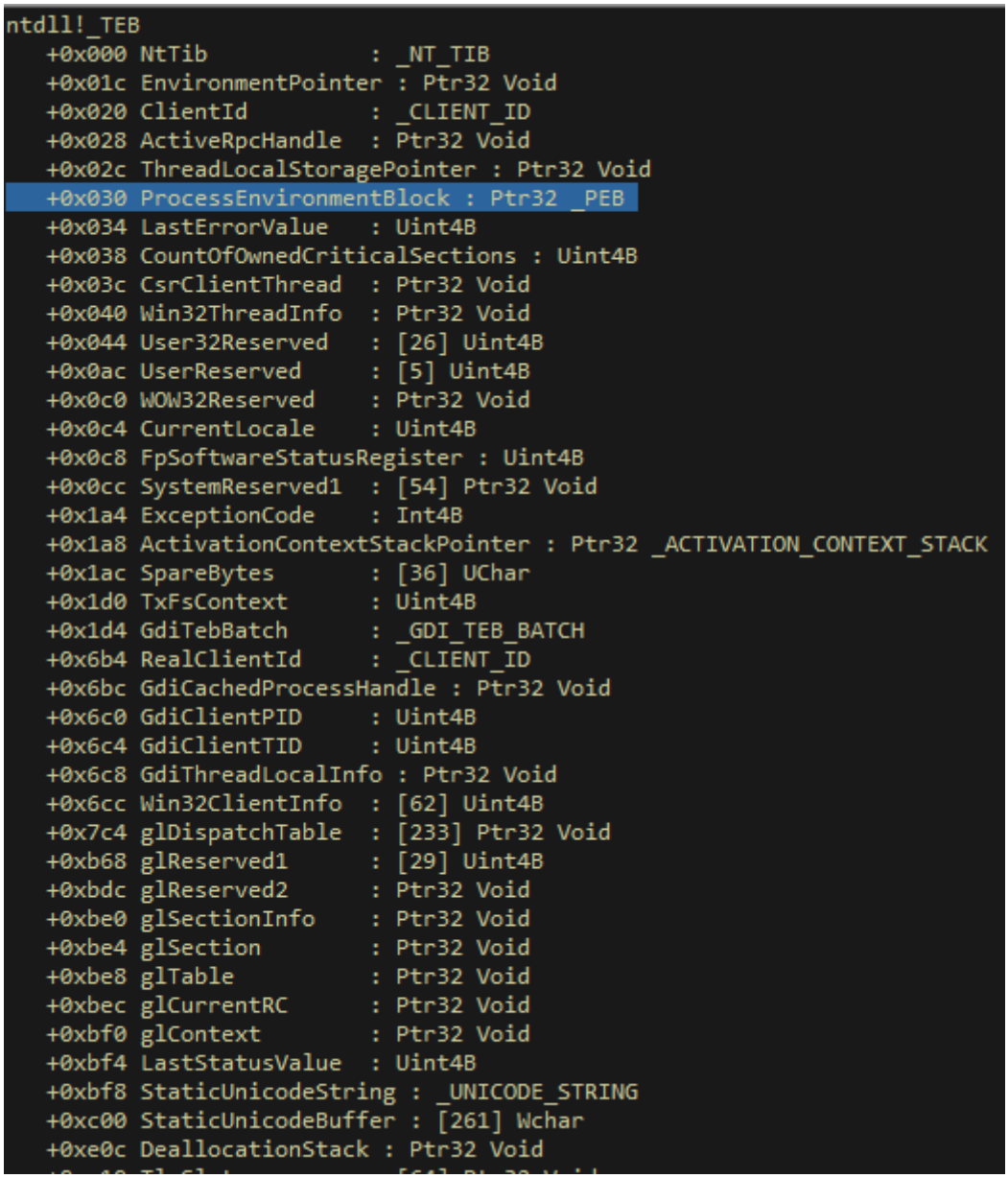
FS is a special kind of segment register which contains pointers to Windows kernel data structures related to the current Process/Thread. FS:[18] is the pointer to the TIB structure. Thread Information Block is also known as the Thread Environment Block. It describes the state of the thread.

FS寄存器是一个特别的段寄存器，里面包含着当前Process/Thread中关于windows 内核数据结构的指针。FS:[18]是一个指向TIB结构的指针。线程信息块又被人们称为线程环境块。里面描述了关于当前线程的状态。

https://msdn.microsoft.com/en-us/library/windows/desktop/ms686708(v=vs.85).aspx
我们来使用windbg调试器来浏览一下TEB结构



As you can see 0x30 is a pointer to the PEB structure which is the Process Information Block. It contains process information as the name describes. So MOV EAX,DWORD PTR DS:[EAX+30] will move the address of the PEB.
你可以看到0x30是一个指向PEB结构的指针。https://msdn.microsoft.com/en-us/library/windows/desktop/aa813706(v=vs.85).aspx（相关资料）
里面包含了进程信息的描述。所以 `MOV EAX,DWORD PTR DS:[EAX+30]` 这条指令将会移动地址到PEB



This is how the PEB Structure looks like.

```
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 IsLegacyProcess  : Pos 2, 1 Bit
   +0x003 IsImageDynamicallyRelocated : Pos 3, 1 Bit
   +0x003 SkipPatchingUser32Forwarders : Pos 4, 1 Bit
   +0x003 IsPackagedProcess : Pos 5, 1 Bit
   +0x003 IsAppContainer   : Pos 6, 1 Bit
   +0x003 SpareBits        : 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
   +0x01c FastPebLock      : Ptr32 _RTL_CRITICAL_SECTION
   +0x020 AtlThunkSListPtr : Ptr32 Void
   +0x024 IFEOKey          : Ptr32 Void
   +0x028 CrossProcessFlags : Uint4B
   +0x028 ProcessInJob     : Pos 0, 1 Bit
   +0x028 ProcessInitializing : Pos 1, 1 Bit
   +0x028 ProcessUsingVEH  : Pos 2, 1 Bit
   +0x028 ProcessUsingVCH  : Pos 3, 1 Bit
   +0x028 ProcessUsingFTH  : Pos 4, 1 Bit
   +0x028 ReservedBits0    : Pos 5, 27 Bits
   +0x02c KernelCallbackTable : Ptr32 Void
   +0x02c UserSharedInfoPtr : Ptr32 Void
   +0x030 SystemReserved   : [1] Uint4B
   +0x034 AtlThunkSListPtr32 : Uint4B
   +0x038 ApiSetMap        : Ptr32 Void
   +0x03c TlsExpansionCounter : Uint4B
   +0x040 TlsBitmap        : Ptr32 Void
   +0x044 TlsBitmapBits    : [2] Uint4B
   +0x04c ReadOnlySharedMemoryBase : Ptr32 Void
   +0x050 HotpatchInformation : Ptr32 Void
   +0x054 ReadOnlyStaticServerData : Ptr32 Ptr32 Void
```

MOVZX EAX,BYTE PTR DS:[EAX+2] will move the BeingDebugged bit with zero extend to EAX. If we check the BeingDebugged bit in this context you can it's set to 1, meaning the program is being currently debugged.

`MOVZX EAX,BYTE PTR DS:[EAX+2]` 将把BeingDebugged位移到零扩展到EAX。如果我们在这个上下文中检查BeingDebugged位，它可以设置为1，这意味着该程序当前正在调试。

```
ntdll!_PEB
   +0x000 InheritedAddressSpace : 0 ''
   +0x001 ReadImageFileExecOptions : 0 ''
   +0x002 BeingDebugged    : 0x1 ''
   +0x003 BitField         : 0 ''
   +0x003 ImageUsesLargePages : 0y0
   +0x003 IsProtectedProcess : 0y0
   +0x003 IsLegacyProcess  : 0y0
   +0x003 IsImageDynamicallyRelocated : 0y0
   +0x003 SkipPatchingUser32Forwarders : 0y0
   +0x003 IsPackagedProcess : 0y0
   +0x003 IsAppContainer   : 0y0
   +0x003 SpareBits        : 0y0
   +0x004 Mutant           : 0xffffffff Void
   +0x008 ImageBaseAddress : 0x00400000 Void
   +0x00c Ldr              : 0x77c7a1e0 _PEB_LDR_DATA
```

我希望你了解IsDebuggerPresent API背后的逻辑。 这里有一个使用FASM的示例，我已经自己实现了isdebuggerpresent api。 而不是使用windows这个API，你可以使用C / C ++应用程序中的内联汇编代码来检查BeingDebugged位。

```
format pe gui 4.0
entry start

; »»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»»
; Title: Checking if the process is being debugged by a ring3 debugger
;        using the PEB's BeingDebugged bit.
;
; Website: http://osandamalith.wordpress.com
; Author: Osanda Malith Jayathissa (@OsandaMalith)
; ««««««««««««««««««««««««««««««««««««««««««««««««««««««««««««««««««««««««««


include 'win32a.inc'

;======================================
section '.data' data readable writeable
;======================================

Title    db     "Status",0
Found    db     "Debugger Found",0
NotFound db     "Debuger Not Found",0

; ======================================
section '.text' code readable executable
;=======================================

start:
        mov           eax, [fs:0x18]     ; Pointer to TEB Structure
        mov           eax, [eax + 0x30]  ; Pointer to PEB Structure
        movzx         eax, byte [eax + 2]; BeingDebugged bit
        cmp           eax, TRUE
        je            found

        push          0x30
        push          Title
        push          NotFound
        push          0
        call          [MessageBox]
        jmp exit

found:
        push          0x10
        push          Title
        push          Found
        push          0
        call          [MessageBox]

exit:
        push          0
        call          [ExitProcess]

; ==============================================
section '.idata' import data readable
```

```
; ===========================================

library kernel32,'kernel32.dll',\
        User32,'user32.dll'

import  kernel32,\
        ExitProcess,'ExitProcess'

import  User32,\
        MessageBox,'MessageBoxA'
```

这里是一个使用C的例子，其中我首先使用ZwQueryInformationProcess内核API的PEB的地址，然后检查BeingDebugged位。 API描述了使用ProcessBasicInformation作为ProcessInformationClass，我们可以检索指向PEB结构的指针。

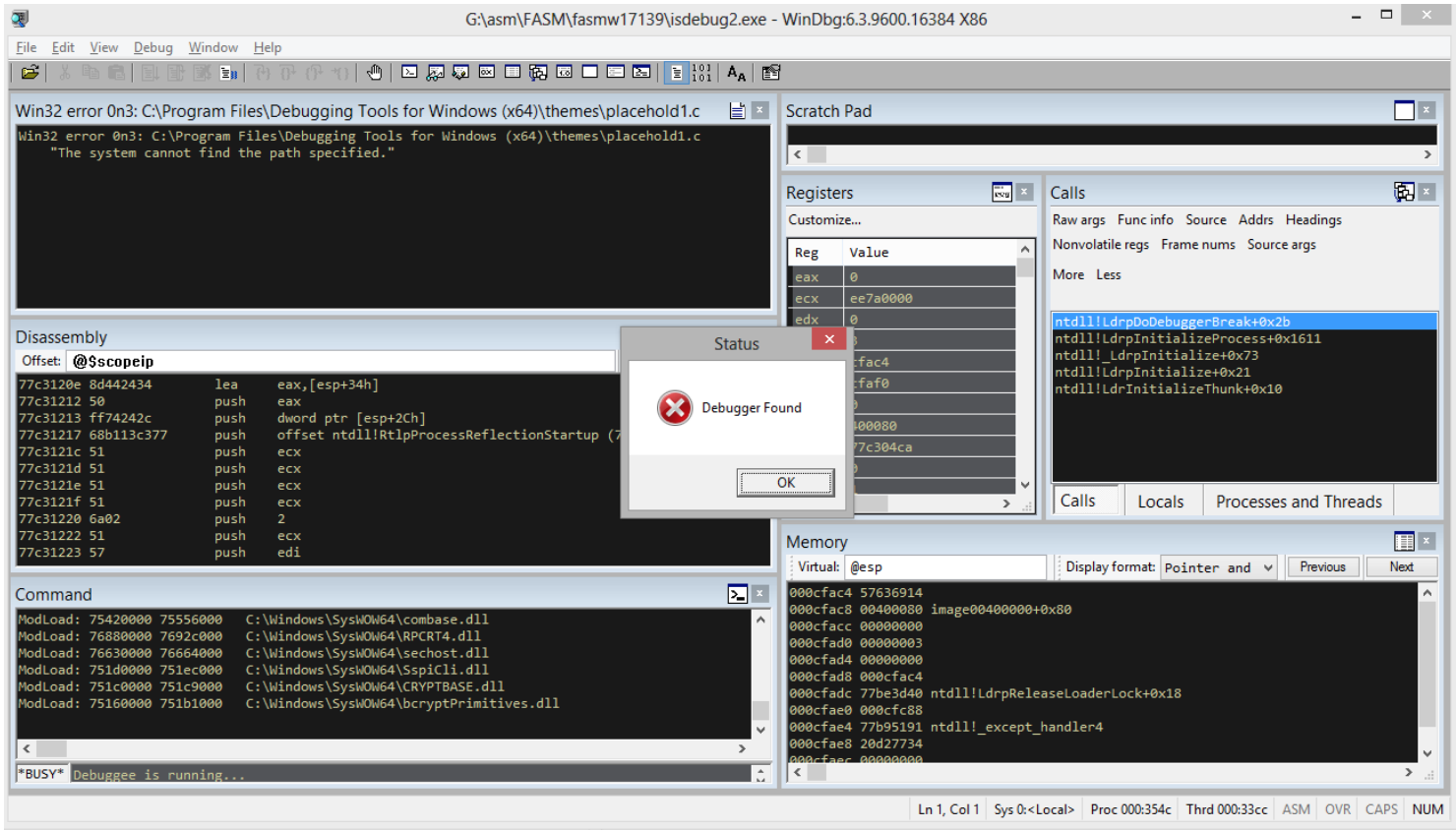
```
NTSTATUS WINAPI ZwQueryInformationProcess(
  _In_      HANDLE           ProcessHandle,
  _In_      PROCESSINFOCLASS ProcessInformationClass,
  _Out_     PVOID            ProcessInformation,
  _In_      ULONG            ProcessInformationLength,
  _Out_opt_ PULONG           ReturnLength
);
```

检查文档你可以看到PebBaseAddress是ProcessBasicInformation结构的成员，它指向PEB。 因此，我们使用ProcessBasicInformation的指针作为ProcessInformationClass参数。

```
typedef struct _PROCESS_BASIC_INFORMATION {
    PVOID Reserved1;
    PPEB PebBaseAddress;
    PVOID Reserved2[2];
    ULONG_PTR UniqueProcessId;
    PVOID Reserved3;
} PROCESS_BASIC_INFORMATION;
```

https://msdn.microsoft.com/en-us/library/windows/desktop/ms687420(v=vs.85).aspx
这里有一个例子，我使用C / C ++来获取BeingDebugged位。

```
#include <Winternl.h>
#include <Windows.h>
#include <tchar.h>

/*
 * Author: Osanda Malith Jayathissa (@OsandaMalith)
 * Website: http://OsandaMalith.wordpress.com
 * Using ZwQueryInformationProcess we get the PEB Address and
 * then we check the BeingDebugged bit to determine the process is being debugged or not.
 */

int main() {

    typedef unsigned long(__stdcall *pfnZwQueryInformationProcess)
    (
        IN  HANDLE,
        IN  unsigned int,
        OUT PVOID,
        IN  ULONG,
        OUT PULONG
    );
    pfnZwQueryInformationProcess ZwQueryInfoProcess = NULL;

    HMODULE hNtDll = LoadLibrary(_T("ntdll.dll"));
    if (hNtDll == NULL) { }

    ZwQueryInfoProcess = (pfnZwQueryInformationProcess) GetProcAddress(hNtDll,
        "ZwQueryInformationProcess");
    if (ZwQueryInfoProcess == NULL) { }
    unsigned long status;

    DWORD pid = GetCurrentProcessId();
    HANDLE hProcess = OpenProcess(PROCESS_QUERY_INFORMATION, FALSE, pid);
    PROCESS_BASIC_INFORMATION pbi;
    status = ZwQueryInfoProcess(hProcess,
                                ProcessBasicInformation,
                                &pbi,
                                sizeof(pbi),
                                NULL);

    PPEB peb_addr = pbi.PebBaseAddress;
    DWORD ptr = pbi.PebBaseAddress;
    ptr|=0x2;
    DWORD *temp = ptr;
    MessageBox(0, *temp & 1 ? "Debugger found" : "Debugger not found","Status",0x30);

    return 0;
}
```



审核评价： 没有任何评价...

评论

提交