Class 15

CSC 471 Modern Malware Analysis
Anti-Debugging Technique

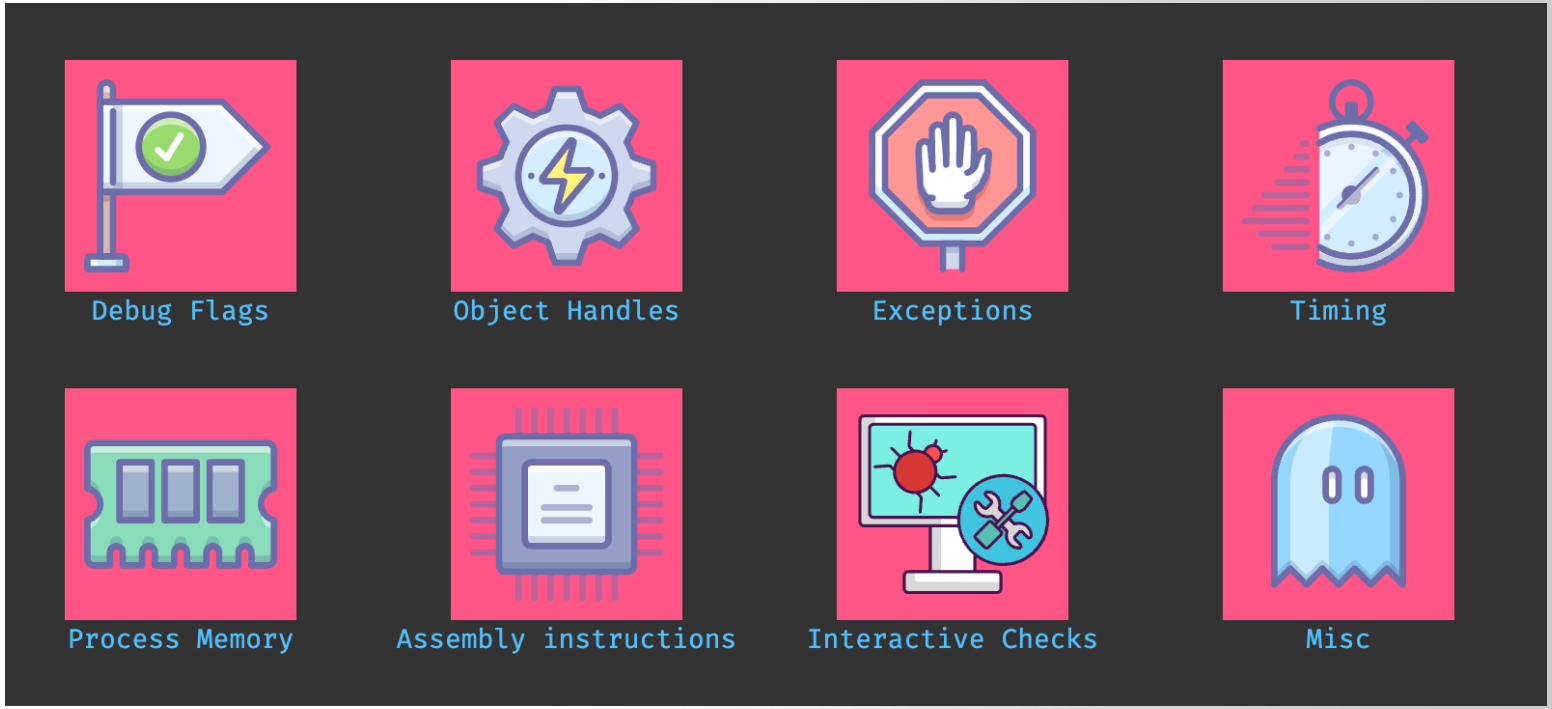
Si Chen (schen@wcupa.edu)

Anti-Debugging

- Malware authors have always looked for new techniques to **stay invisible**. This includes, of course, being invisible on the compromised machine, but it is even more important to hide malicious indicators and behavior during analysis.
- **Debugging** is the essential part of malware analysis. Every time we need to drill down into malware behavior, restore encryption methods or examine communication protocols, we use debuggers.
- To make the post-detection analysis more difficult, threat actors use various anti-analysis techniques, one of the more common ones is **Anti-Debugging.**

What is Anti-Debugging?

- Anti-Debugging techniques are meant to ensure that a program is not running under a debugger, and in the case that it is, to change its behavior correspondingly.
- In most cases, the Anti-Debugging process will slow down the process of reverse engineering, but will not prevent it.



Anti-debugging tricks

Static Anti-Debugging VS. Dynamic Anti-Debugging

	Static	Dynamic
Difficulty Level	Easy, Medium	Hard
Key idea	Use System Information	Reverse and exploit Debugger
Target	Detect Debugger	Hide it's own code and data
Time point	When debugging started	While debugger are running
Defend Method(s)	API hook, debugger plugin	API hook, Debugger Plugin, Other tools
Example(s)	PEB, TEB, Native API, TLS	SHE, Break Points (INT3), Timing Check

Static Anti-Debugging Techniques

- The debuggee is trying to detect if it being debugged.
  - If being debugged -- run some arbitrary codes – usually the code to terminate itself.

Process Environment Block (PEB)

- The function **kernel32!IsDebuggerPresent()** determines whether the current process is being debugged by a user-mode debugger such as OllyDbg or x64dbg.
- Generally, the function only checks the **BeingDebugged** flag of the **Process Environment Block (PEB)**.
- The following code can be used to terminate process if it is being debugged:

Assembly Code

```
    call IsDebuggerPresent
    test al, al
    jne  being_debugged
    ...
being_debugged:
    push 1
    call ExitProcess
```

C/C++ Code

```
if (IsDebuggerPresent())
    ExitProcess(-1);
```

Process Environment Block (PEB)

- What's Process Environment Block (PEB)?

In computing the **Process Environment Block** (abbreviated **PEB**) is a data structure in the Windows NT operating system family. It is an opaque data structure that is used by the operating system internally, most of whose fields are not intended for use by anything other than the operating system.

C++ Copy

```cpp
typedef struct _PEB {
  BYTE                          Reserved1[2];
  BYTE                          BeingDebugged;
  BYTE                          Reserved2[1];
  PVOID                         Reserved3[2];
  PPEB_LDR_DATA                 Ldr;
  PRTL_USER_PROCESS_PARAMETERS  ProcessParameters;
  PVOID                         Reserved4[3];
  PVOID                         AtlThunkSListPtr;
  PVOID                         Reserved5;
  ULONG                         Reserved6;
  PVOID                         Reserved7;
  ULONG                         Reserved8;
  ULONG                         AtlThunkSListPtr32;
  PVOID                         Reserved9[45];
  BYTE                          Reserved10[96];
  PPS_POST_PROCESS_INIT_ROUTINE PostProcessInitRoutine;
  BYTE                          Reserved11[128];
  PVOID                         Reserved12[1];
  ULONG                         SessionId;
} PEB, *PPEB;
```

Thread Environment Block (TEB)

- In computing, the Win32 Thread Environment Block (TEB) is **a data structure in Win32 on x86 that stores information about the currently running thread**.

C++ Copy

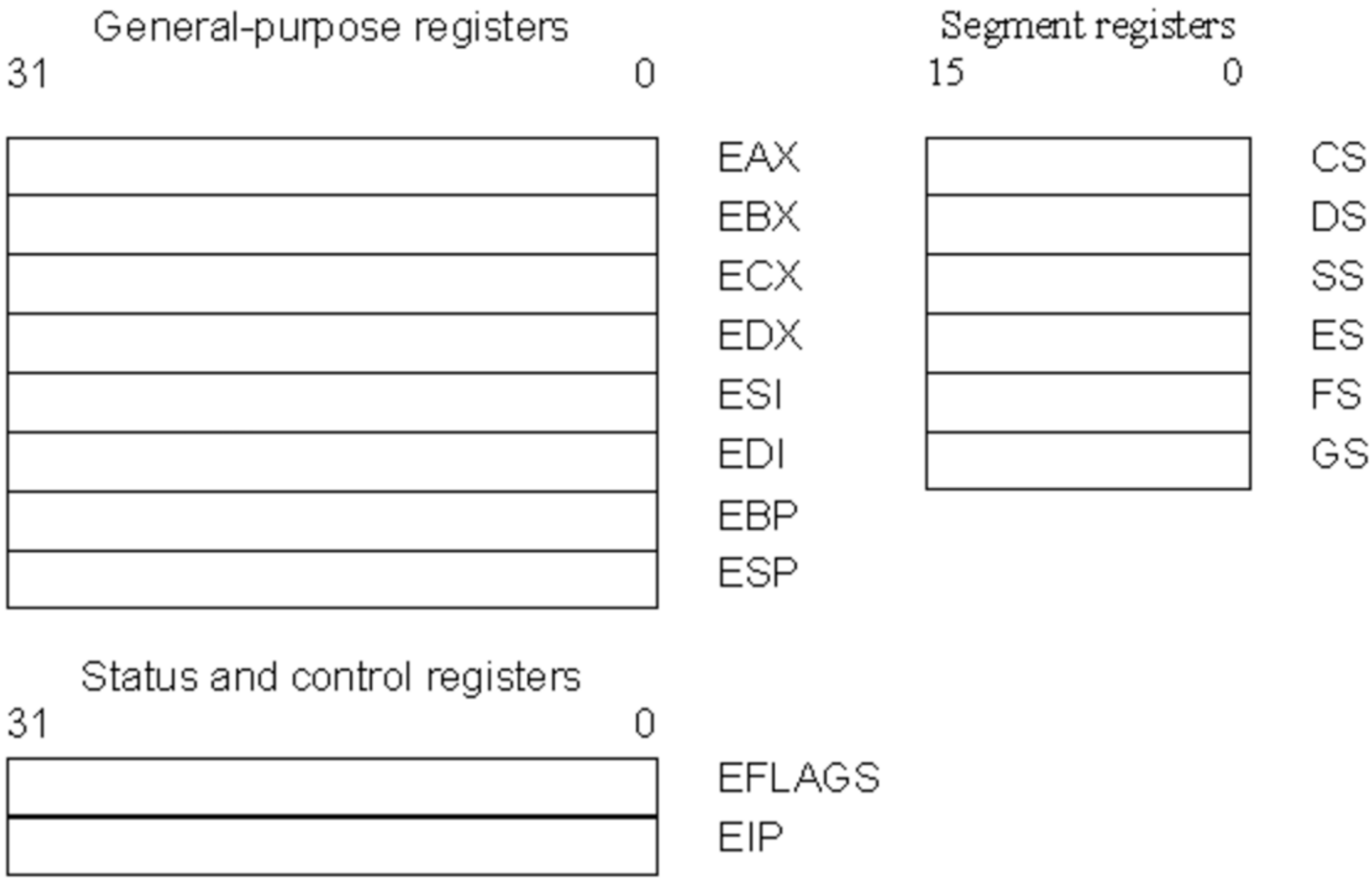
```
typedef struct _TEB {
  PVOID Reserved1[12];
  PPEB  ProcessEnvironmentBlock;
  PVOID Reserved2[399];
  BYTE  Reserved3[1952];
  PVOID TlsSlots[64];
  BYTE  Reserved4[8];
  PVOID Reserved5[26];
  PVOID ReservedForOle;
  PVOID Reserved6[4];
  PVOID TlsExpansionSlots;
} TEB, *PTEB;
```

PEB

Thread Environment Block (TEB)

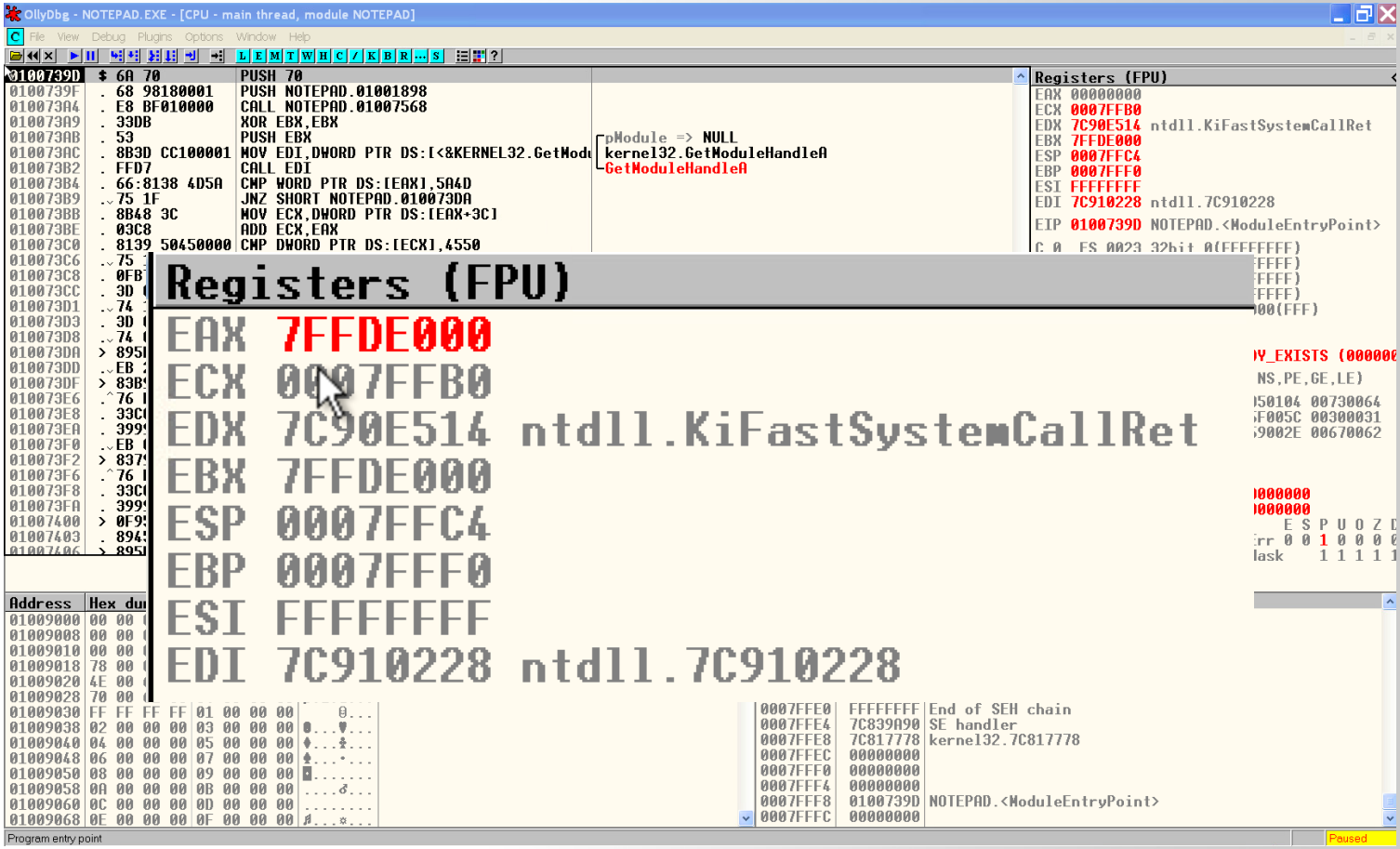
- On the user mode basis of a 32-bit window, the **FS** register points to a structure called a **Thread Environment Block** (TEB) or Thread Information Block (TIB).

Offset (x86)	Offset (x64)	Definition	Versions	Remarks
0x00	0x00	`NT_TIB NtTib;`	all	
0x1C	0x38	`PVOID EnvironmentPointer;`	all	
0x20	0x40	`CLIENT_ID ClientId;`	all	
0x28	0x50	unknown pointer to **CSR_QLPC_TEB**	3.10 only	next as structure at 0x01AC
		`PVOID ActiveRpcHandle;`	3.50 and higher	
0x2C	0x58	`PVOID ThreadLocalStoragePointer;`	all	
0x30	0x60	`PEB *ProcessEnvironmentBlock;`	all	
0x34	0x68	`ULONG LastErrorValue;`	all	
0x38	0x6C	unknown byte	3.10 only	
		`ULONG CountOfOwnedCriticalSections;`	3.50 and higher	



FS + GDTR (Global Descriptor Table Register) = Addr of TEB

Check PEB via OllyDbg



Type: MOV EAX, DWORD PTR FS:[30]

Check PEB via OllyDbg

- Run the command StepIn (F7) or StepOver (F8)
- Check the EAX value

Registers (FPU)

```
EAX 7FFDE000
ECX 0007FFB0
EDX 7C90E514 ntdll.KiFastSyst
EBX 7FFDE000
ESP 0007FFC4
EBP 0007FFF0
ESI FFFFFFFF
EDI 7C910228 ntdll.7C910228
```

PEB

Address	Hex dump	ASCII
7FFDE000	00 00 01 00 FF FF FF FF	..☺.
7FFDE008	00 00 00 01 A0 1E 1A 00	...☺á▲→.
7FFDE010	00 00 02 00 00 00 00 00	..☻.....
7FFDE018	00 00 0A 00 20 06 98 7C	.... ♠ÿ¦
7FFDE020	00 10 90 7C E0 10 90 7C	.►É¦α►É¦
7FFDE028	01 00 00 00 70 29 41 7E	☺...p)A~
7FFDE030	00 00 00 00 00 00 00 00	.....
7FFDE038	00 00 00 00 00 00 00 00	.....
7FFDE040	E0 05 98 7C FF FF 03 00	α♣ÿ¦ ♥.
7FFDE048	00 00 00 00 00 00 6F 7F	.....o⌂
7FFDE050	00 00 6F 7F 88 05 6F 7F	..o⌂ê♠o⌂
7FFDE058	00 00 FB 7F 00 10 FC 7F	..√⌂.►ⁿ⌂
7FFDE060	00 20 FD 7F 02 00 00 00	. ²⌂☻...
7FFDE068	70 00 00 00 00 00 00 00	p.....
7FFDE070	00 80 9B 07 6D E8 FF FF	.Ç¢•mΦ
7FFDE078	00 00 10 00 00 20 00 00	..►.. ..
7FFDE080	00 00 01 00 00 10 00 00	..☺..►..
7FFDE088	09 00 00 00 10 00 00 00	....►...
7FFDE090	E0 FF 97 7C 00 00 43 00	α ù¦..C.
7FFDE098	00 00 00 00 14 00 00 00	....¶...
7FFDE0A0	74 E1 97 7C 05 00 00 00	tßù¦♣...
7FFDE0A8	01 00 00 00 28 0A 00 03	☺...(..♥
7FFDE0B0	02 00 00 00 02 00 00 00	☻...☻...
7FFDE0B8	04 00 00 00 00 00 00 00	♦.....

BeingDebugged(+0x2) → PEB

C++ Copy

```cpp
typedef struct _PEB {
  BYTE                          Reserved1[2];
  BYTE                          BeingDebugged;
  BYTE                          Reserved2[1];
  PVOID                         Reserved3[2];
  PPEB_LDR_DATA                 Ldr;
  PRTL_USER_PROCESS_PARAMETERS  ProcessParameters;
  PVOID                         Reserved4[3];
  PVOID                         AtlThunkSListPtr;
  PVOID                         Reserved5;
  ULONG                         Reserved6;
  PVOID                         Reserved7;
  ULONG                         Reserved8;
  ULONG                         AtlThunkSListPtr32;
  PVOID                         Reserved9[45];
  BYTE                          Reserved10[96];
  PPS_POST_PROCESS_INIT_ROUTINE PostProcessInitRoutine;
  BYTE                          Reserved11[128];
  PVOID                         Reserved12[1];
  ULONG                         SessionId;
} PEB, *PPEB;
```

- PEB.BeingDebugged set to
  - 1 (TRUE) → Being Debugged
  - 0 (FALSE) → Not being debugged

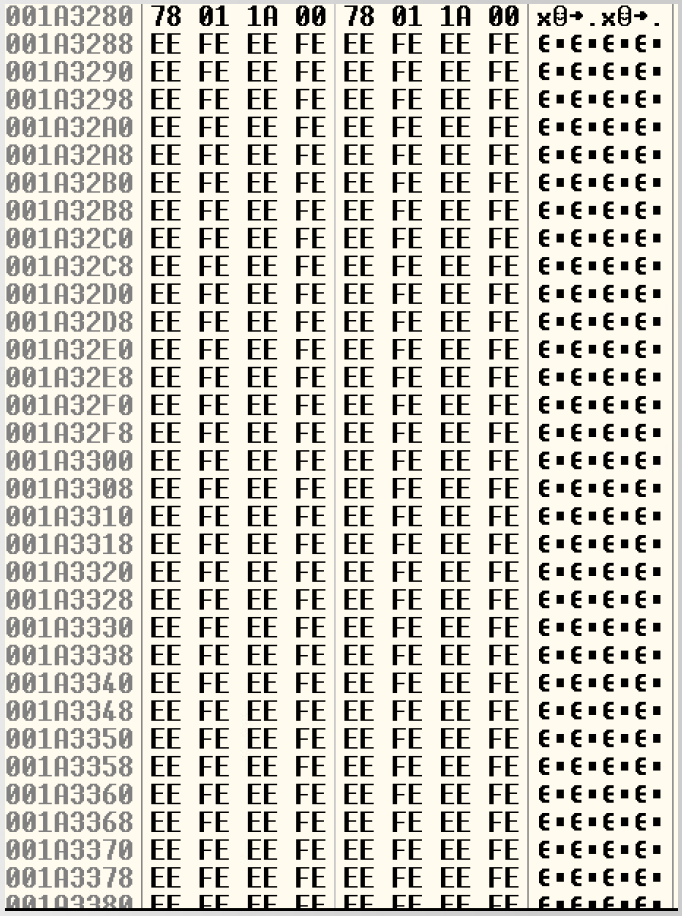
Ldr(+0xC) → PEB

Offset (x86)	Offset (x64)	Definition	Versions
0x08	0x10	`PVOID ImageBaseAddress;`	all
0x0C	0x18	`PEB_LDR_DATA *Ldr;`	all
0x10	0x20	`RTL_USER_PROCESS_PARAMETERS *ProcessParameters;`	all
0x14	0x28	`PVOID SubSystemData;`	all
0x18	0x30	`HANDLE ProcessHeap;`	all

- When debug a process, it's heap memory will be filled with some special values, to show that it's being debugged.
  - Empty heap memory will be filled with 0xFEFEFE
- PEB.Ldr is a pointer pointing to _PEB_LDR_DATA and _PEB_LDR_DATA is crated in side the heap memory

Ldr(+0xC) → PEB

- When debugging a process, it's heap memory will be filled with **some special values**, to show that it's being debugged.
  - Empty heap memory will be filled with **0xFEFEFE**
- PEB.Ldr is a pointer pointing to _PEB_LDR_DATA and _PEB_LDR_DATA is crated inside the heap memory



Process Heap(+0x18) → PEB

Offset (x86)	Offset (x64)	Definition	Versions
0x08	0x10	PVOID ImageBaseAddress;	all
0x0C	0x18	PEB_LDR_DATA *Ldr;	all
0x10	0x20	RTL_USER_PROCESS_PARAMETERS *ProcessParameters;	all
0x14	0x28	PVOID SubSystemData;	all
0x18	0x30	HANDLE ProcessHeap;	all

- PEB.ProcessHeap is point to a HEAP structure.
- This HEAP structure can be used as an anti-debugging technique. This first heap contains a header with fields (ForceFlags, Flags) used to tell the kernel whether the heap was created within a debugger.
- Below are the offsets (relative to ProcessHeap) for Windows XP and Windows 7.

Field	Size	Offset relative to ProcessHeap (Windows XP)	Offset relative to ProcessHeap (Windows 7)
ForceFlags	DWORD	0x10	0x44
Flags	DWORD	0x0C	0x40

Process Heap(+0x18) → PEB

HEAP.flags(0xC) & HEAP.ForceFlags(0x10)

Address	Hex dump	ASCII
000A0000	C8 00 00 00 86 01 00 00	╚...åΘ..
000A0008	FF EE FF EE 62 00 00 50	∩ ∈b..P
000A0010	60 00 00 40 00 FE 00 00	`..@.■..
000A0018	00 00 10 00 00 20 00 00	..►.. ..
000A0020	00 02 00 00 00 20 00 00	.☻... ..
000A0028	CC 02 00 00 FF EF FD 7F	╠☻.. ∩²⌂
000A0030	01 00 08 06 00 00 00 00	☺.◘♠....
000A0038	00 00 00 00 00 00 00 00	.....
000A0040	00 00 00 00 98 05 0A 00	....ÿ♣..
000A0048	17 00 00 00 F8 FF FF FF	↨...°
000A0050	50 00 0A 00 50 00 0A 00	P...P...
000A0058	40 06 0A 00 00 00 00 00	@♠.....

If running normally (not in debug mode), Heap.Flag(+0xC) is 0x2 and Heap.ForceFlag(0x10)'s value is 0x0

Only works in WinXP

NtBlobalFlag(+0x68) → PEB

Offset (x86)	Offset (x64)	Definition	Versions
0x3C	0x70	`ULONG TlsExpansionCounter;`	all
	0x74	`UCHAR Padding2 [4];`	6.3 and higher
0x40	0x78	`PVOID TlsBitmap;`	all
0x44	0x80	`ULONG TlsBitmapBits [2];`	all
0x4C	0x88	`PVOID ReadOnlySharedMemoryBase;`	all
0x50	0x90	`PVOID ReadOnlySharedMemoryHeap;`	3.10 to 5.2
		`PVOID HotpatchInformation;`	6.0 to 6.2
		`PVOID SparePvoid0;`	6.3 to 1607
		`PVOID SharedData;`	1703 and higher
0x54	0x98	`PVOID *ReadOnlyStaticServerData;`	all
0x58	0xA0	`PVOID AnsiCodePageData;`	all
0x5C	0xA8	`PVOID OemCodePageData;`	all
0x60	0xB0	`PVOID UnicodeCaseTableData;`	all
0x64	0xB8	`ULONG NumberOfProcessors;`	3.51 and higher
0x68	0xBC	`ULONG NtGlobalFlag;`	3.51 and higher
0x68 (3.10 to 3.50); 0x70	0xC0	`LARGE_INTEGER CriticalSectionTimeout;`	all

When debugged, PEB.NtGlobal is set to **0x70** Otherwise is 0x0

0x70 = FLG_HEAP_ENABLE_TAIL_CHECK (0x10) || FLG_HEAP_ENABLE_FREE_CHECK (0x20) || FLG_HEAP_VALIDATE_PARAMETERS (0x40)

NtBlobalFlag(+0x68) → PEB

Address	Hex dump							
7FFDE000	00	00	01	00	FF	FF	FF	FF
7FFDE008	00	00	00	01	A0	1E	1A	00
7FFDE010	00	00	02	00	00	00	00	00
7FFDE018	00	00	0A	00	20	06	98	7C
7FFDE020	00	10	90	7C	E0	10	90	7C
7FFDE028	01	00	00	00	70	29	41	7E
7FFDE030	00	00	00	00	00	00	00	00
7FFDE038	00	00	00	00	00	00	00	00
7FFDE040	E0	05	98	7C	FF	FF	03	00
7FFDE048	00	00	00	00	00	00	6F	7F
7FFDE050	00	00	6F	7F	88	06	6F	7F
7FFDE058	00	00	FB	7F	00	10	FC	7F
7FFDE060	00	20	FD	7F	02	00	00	00
7FFDE068	70	00	00	00	00	00	00	00
7FFDE070	00	80	9B	07	6D	E8	FF	FF

rsions

3 and higher

0 to 5.2

0 to 6.2

3 to 1607

03 and higher

0x58	0xA0	PVOID AnsiCodePageData;	all
0x5C	0xA8	PVOID OemCodePageData;	all
0x60	0xB0	PVOID UnicodeCaseTableData;	all
0x64	0xB8	ULONG NumberOfProcessors;	3.51 and higher
0x68	0xBC	ULONG NtGlobalFlag;	3.51 and higher
0x68 (3.10 to 3.50); 0x70	0xC0	LARGE_INTEGER CriticalSectionTimeout;	all

When debugged, PEB.NtGlobal is set to **0x70** Otherwise is 0x0

0x70 = FLG_HEAP_ENABLE_TAIL_CHECK (0x10) || FLG_HEAP_ENABLE_FREE_CHECK (0x20) || FLG_HEAP_VALIDATE_PARAMETERS (0x40)

Example: StAD_PEB.exe

OllyDbg - StaAD_PEB.exe - [CPU - main thread, module StaAD_PE]

File View Debug Plugins Options Window Help

```
0040155D  $ E8 48280000    CALL StaAD_PE.00403DAA
00401562  .^E9 95FEFFFF    JMP StaAD_PE.004013FC
00401567  > 8BFF           MOV EDI,EDI
00401569  . 55             PUSH EBP
0040156A  . 8BEC           MOV EBP,ESP
0040156C  . 81EC 28030000  SUB ESP,328
00401572  . A3 D8BE4000    MOV DWORD PTR DS:[40BED8],EAX
00401577  . 89
0040157D  . 89
00401583  . 89
00401589  . 89
0040158F  . 89
00401595  . 66
0040159C  . 66
004015A3  . 66
004015AA  . 66
004015B1  . 66
004015B8  . 66
004015BF  . 9C
004015C0  . 8F
004015C6  . 8B
004015C9  . A3
004015CE  . 8B
```

Address	Hex d
0040B000	FF FF
0040B008	4E E6
0040B010	FF FF
0040B018	00 00
0040B020	00 00
0040B028	00 00
0040B030	00 00
0040B038	00 00
0040B040	00 00
0040B048	00 00
0040B050	00 00

C:\Documents and Settings\sec0day\My Documents\Downloads\StaAD_PEB\StaAD_PEB.exe

```
IsDebuggerPresent() = 1
  => Debugging!!!

PEB.Ldr
  => Debugging!!!

PEB.ProcessHeap.Flags = 0x50000062
  => Debugging!!!

PEB.ProcessHeap.ForceFlags = 0x40000060
  => Debugging!!!

PEB.NtGlobalFlag = 0x70
  => Debugging!!!


press any key to quit...
```

Q & A