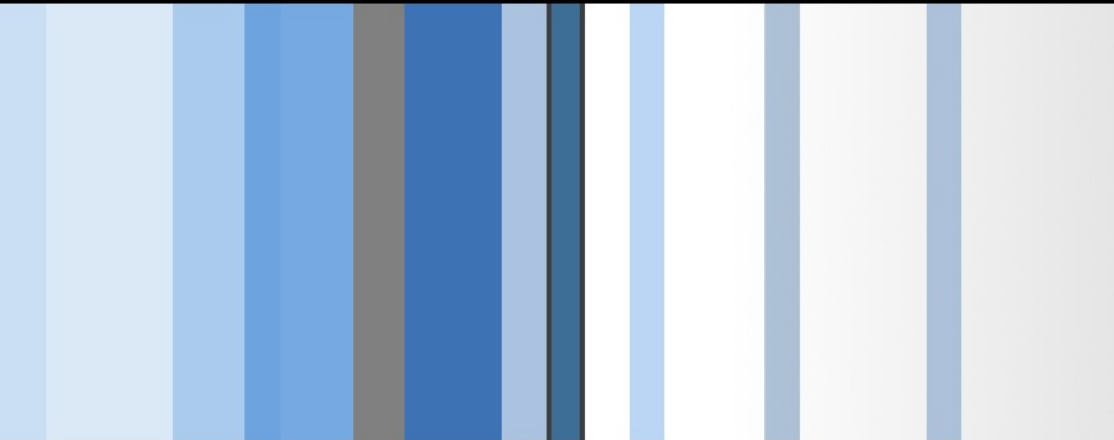
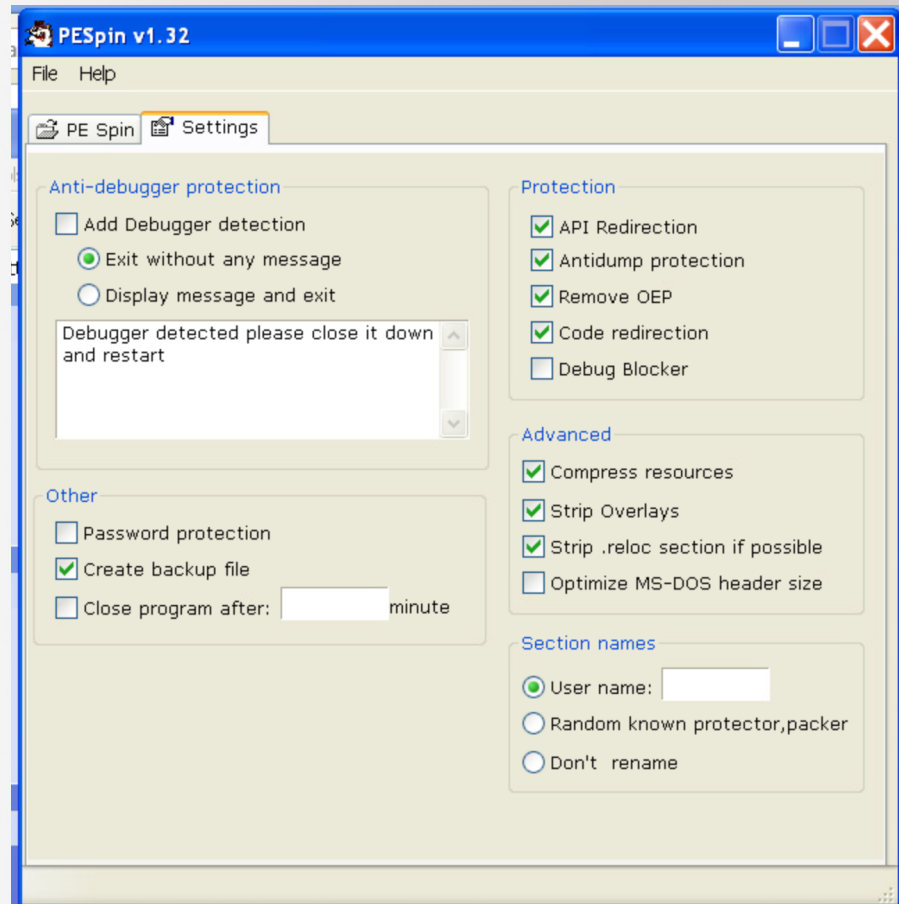


CSC 471 Modern Malware Analysis
Anti-Debugging Techniques (3):
TLS (Thread Local Storage) Callback Function and
Advanced Anti-Debugging Techniques
Si Chen (schen@wcupa.edu)

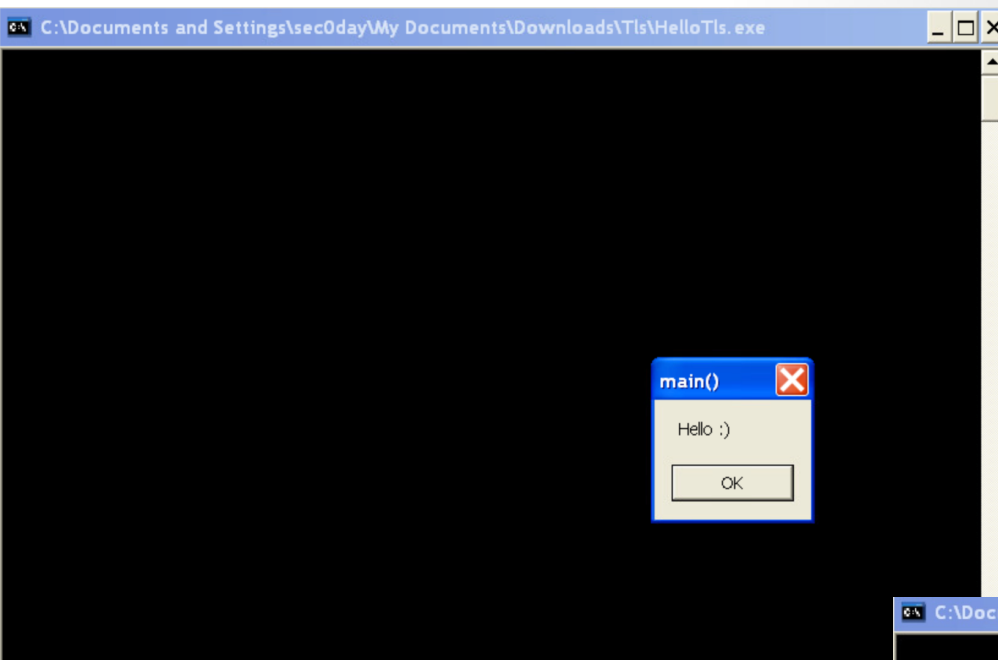


Advanced Anti-Debugging Techniques

- Advanced Anti-Debugging Techniques:
 - Thread Local Storage (TLS) Callback Function
 - Garbage Code
 - Break Code Alignment
 - Encryption/Decryption
 - Stolen Bytes (Remove OEP)
 - API Redirection
 - Self Debugging

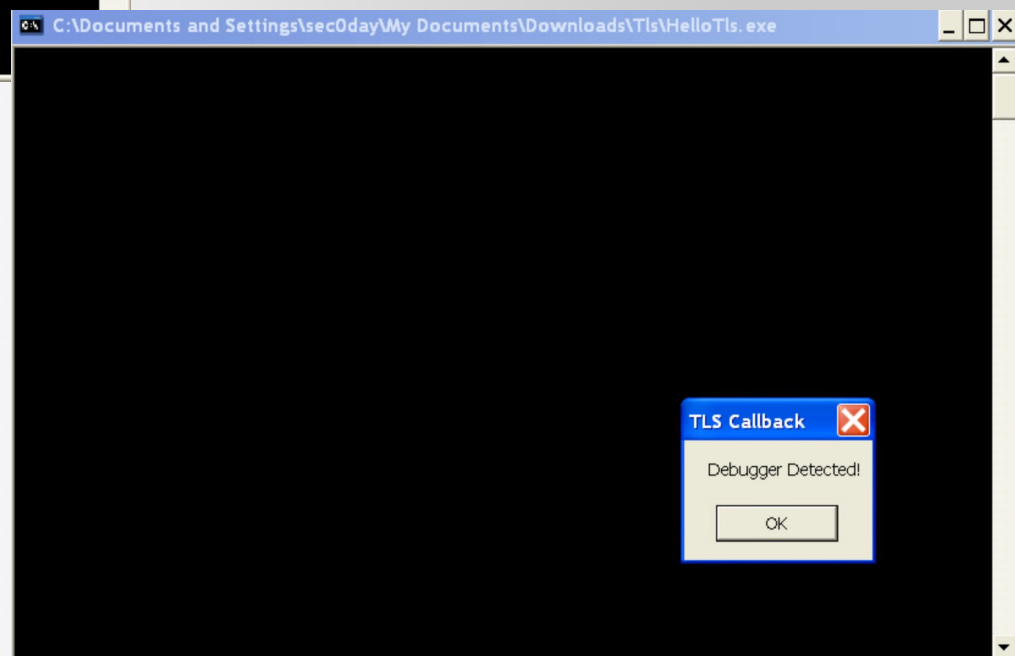


TLS Callback Function Example1: HelloTls.exe



Double Click

Use OllyDBG to load it

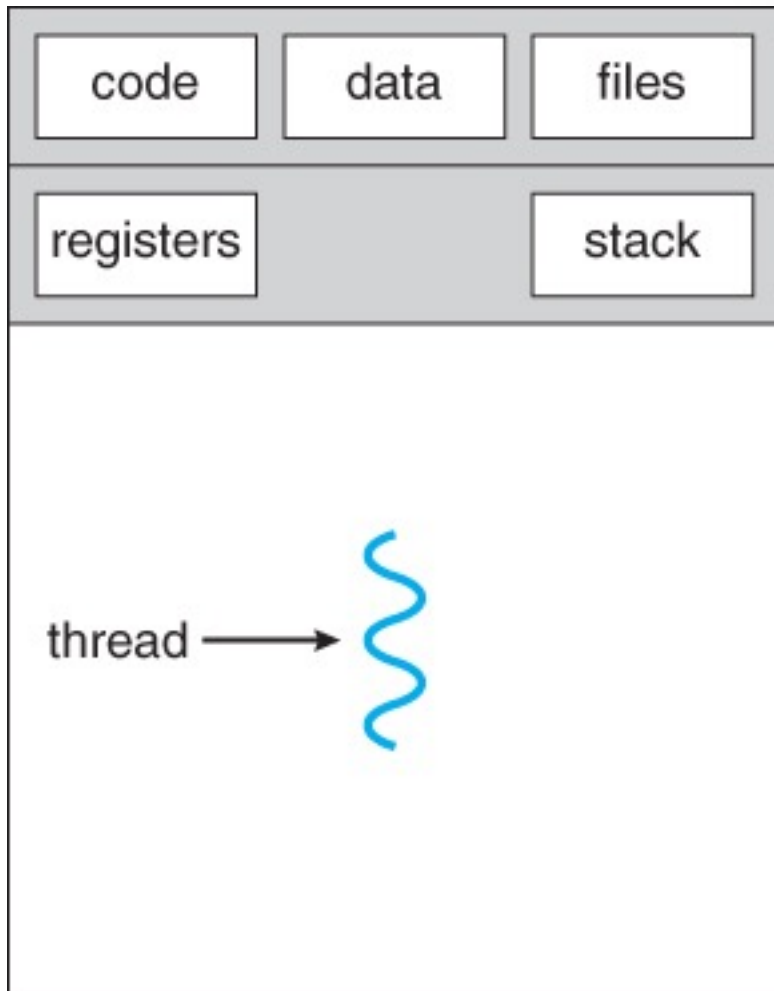


Process and Thread

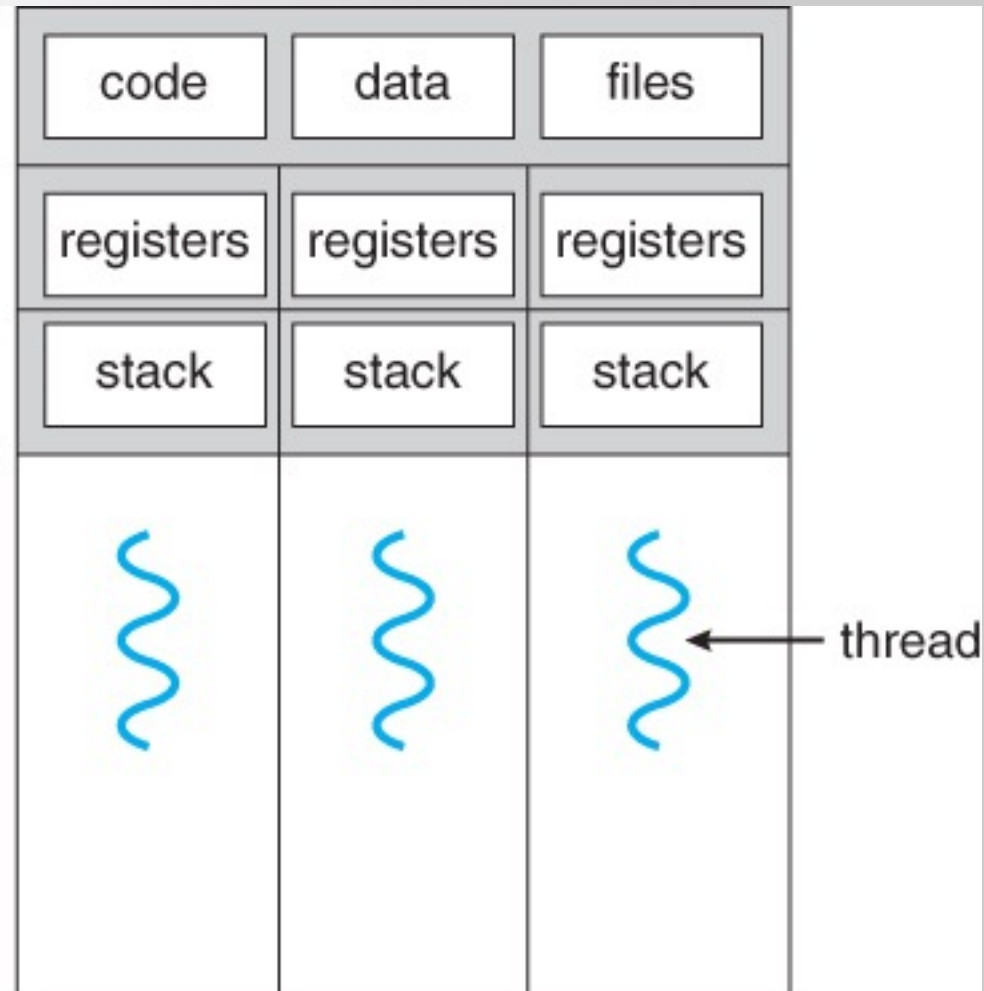
- **Process** is the execution of a program that allows you to perform the appropriate actions specified in a program. It can be defined as an execution unit where a program runs. The OS helps you to create, schedule, and terminates the processes which is used by CPU. The other processes created by the main process are called child process.
- **Thread** is an execution unit that is part of a process. A process can have multiple threads, all executing at the same time. It is a unit of execution in concurrent programming. A thread is lightweight and can be managed independently by a scheduler. It helps you to improve the application performance using parallelism.

<https://www.youtube.com/watch?v=Dhf-DYO1K78>

Process and Thread



single-threaded process



multithreaded process

Thread and fork()

```
int main(void) {
    init();
    pid_t pid;
    while(1) {
        pid = fork();
        if(pid < 0) {
            puts("fork error");
            exit(0);
        }
        else if(pid == 0) {
            puts("welcome");
            fun();
            puts("recv sucess");
        }
        else {
            wait(0);
        }
    }
}
```

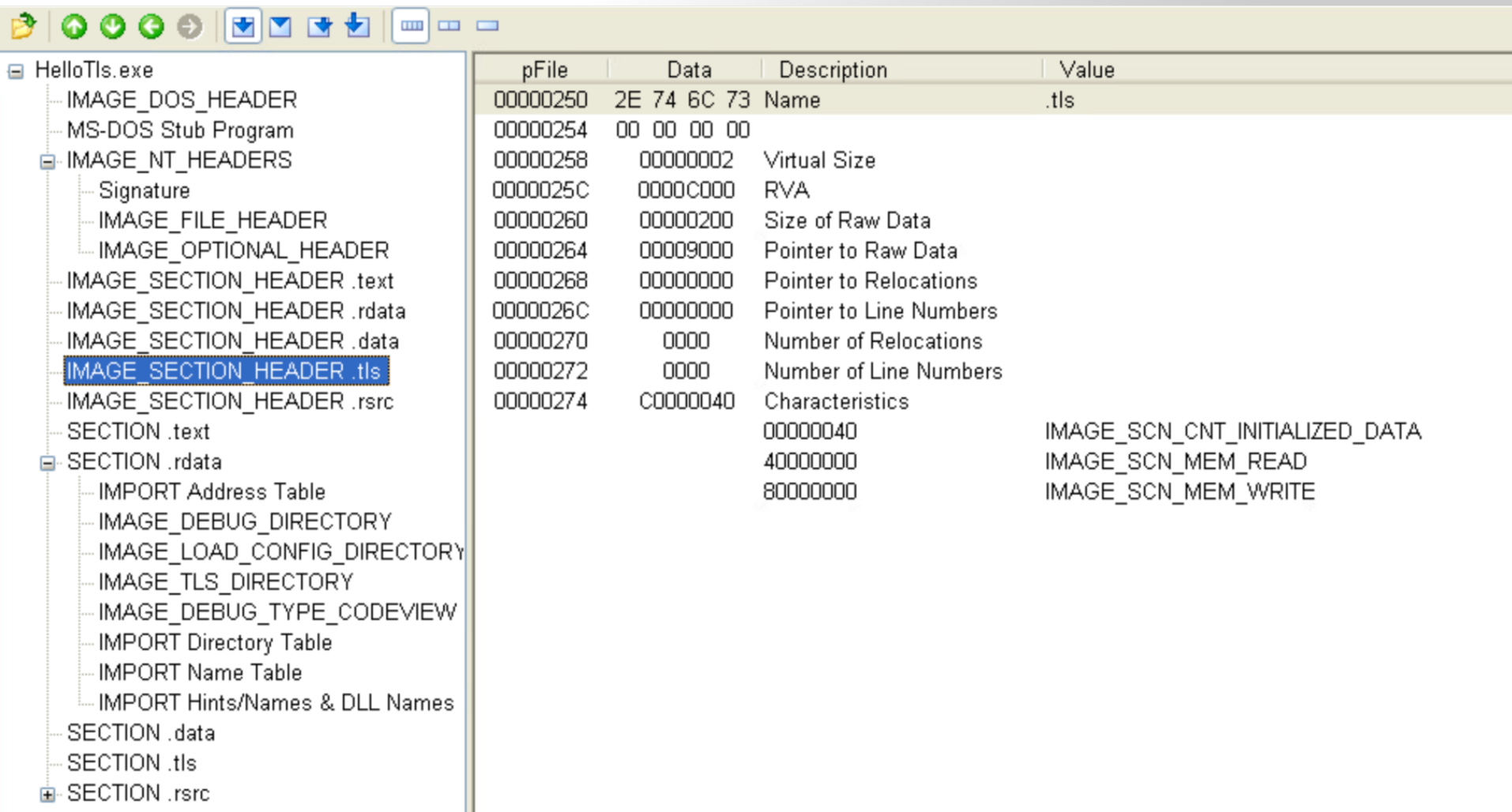
Thread Local Storage

- Thread Local Storage (TLS) is **the mechanism by which each thread in a given multithreaded process allocates storage for thread-specific data.**
- Static and global data are **shared across all the threads.** If you modified a global/static variable it is visible to all the threads.
- Unlike global/shared variable **if you create a variable in TLS, every thread has its own copy of the variable,** i.e. changes to the variable is local to the thread.

Thread Local Storage

00000158	00000000	RVA	EXPORT Table
0000015C	00000000	Size	
00000160	0000961C	RVA	IMPORT Table
00000164	0000003C	Size	
00000168	0000D000	RVA	RESOURCE Table
0000016C	000001B4	Size	
00000170	00000000	RVA	EXCEPTION Table
00000174	00000000	Size	
00000178	00000000	Offset	CERTIFICATE Table
0000017C	00000000	Size	
00000180	00000000	RVA	BASE RELOCATION Table
00000184	00000000	Size	
00000188	00008130	RVA	DEBUG Directory
0000018C	0000001C	Size	
00000190	00000000	RVA	Architecture Specific Data
00000194	00000000	Size	
00000198	00000000	RVA	GLOBAL POINTER Register
0000019C	00000000	Size	
000001A0	00009310	RVA	TLS Table
000001A4	00000018	Size	
000001A8	000092C8	RVA	LOAD CONFIGURATION Table
000001AC	00000040	Size	
000001B0	00000000	RVA	BOUND IMPORT Table
000001B4	00000000	Size	
000001B8	00008000	RVA	IMPORT Address Table
000001BC	000000F0	Size	
000001C0	00000000	RVA	DELAY IMPORT Descriptors
000001C4	00000000	Size	
000001C8	00000000	RVA	CLI Header
000001CC	00000000	Size	

Thread Local Storage



pFile	Data	Description	Value
00000250	2E 74 6C 73	Name	.tls
00000254	00 00 00 00		
00000258	00000002	Virtual Size	
0000025C	0000C000	RVA	
00000260	00000200	Size of Raw Data	
00000264	00009000	Pointer to Raw Data	
00000268	00000000	Pointer to Relocations	
0000026C	00000000	Pointer to Line Numbers	
00000270	0000	Number of Relocations	
00000272	0000	Number of Line Numbers	
00000274	C0000040	Characteristics	
			IMAGE_SCN_CNT_INITIALIZED_DATA
			IMAGE_SCN_MEM_READ
			IMAGE_SCN_MEM_WRITE

IMAGE_TLS_DIRECTORY

```
3393 typedef struct _IMAGE_TLS_DIRECTORY64 {
3394     ULONGLONG StartAddressOfRawData;
3395     ULONGLONG EndAddressOfRawData;
3396     ULONGLONG AddressOfIndex;
3397     ULONGLONG AddressOfCallBacks;
3398     DWORD     SizeOfZeroFill;
3399     DWORD     Characteristics;
3400 } IMAGE_TLS_DIRECTORY64, *PIMAGE_TLS_DIRECTORY64;
3401
3402 typedef struct _IMAGE_TLS_DIRECTORY32 {
3403     DWORD     StartAddressOfRawData;
3404     DWORD     EndAddressOfRawData;
3405     DWORD     AddressOfIndex;
3406     DWORD     AddressOfCallBacks;
3407     DWORD     SizeOfZeroFill;
3408     DWORD     Characteristics;
3409 } IMAGE_TLS_DIRECTORY32, *PIMAGE_TLS_DIRECTORY32;
3410
3411 #ifdef _WIN64
3412 typedef IMAGE_TLS_DIRECTORY64 IMAGE_TLS_DIRECTORY;
3413 typedef PIMAGE_TLS_DIRECTORY64 PIMAGE_TLS_DIRECTORY;
3414 #else
3415 typedef IMAGE_TLS_DIRECTORY32 IMAGE_TLS_DIRECTORY;
3416 typedef PIMAGE_TLS_DIRECTORY32 PIMAGE_TLS_DIRECTORY;
3417 #endif
```

<https://github.com/Alexpux/mingw-w64/blob/master/mingw-w64-tools/widl/include/winnt.h>

IMAGE_TLS_DIRECTORY


File View Go Help

Icons: Refresh, Up, Down, Left, Right, Copy, Paste, Print, Zoom In, Zoom Out, Full Screen

Tree View: HelloTls.exe

- IMAGE_DOS_HEADER
- MS-DOS Stub Program
- IMAGE_NT_HEADERS
 - Signature
 - IMAGE_FILE_HEADER
 - IMAGE_OPTIONAL_HEADER
 - IMAGE_SECTION_HEADER .text
 - IMAGE_SECTION_HEADER .rdata
 - IMAGE_SECTION_HEADER .data
 - IMAGE_SECTION_HEADER .tls
 - IMAGE_SECTION_HEADER .rsrc
 - SECTION .text
 - SECTION .rdata
 - IMPORT Address Table
 - IMAGE_DEBUG_DIRECTORY
 - IMAGE_LOAD_CONFIG_DIRECTORY
 - IMAGE_TLS_DIRECTORY**
 - IMAGE_DEBUG_TYPE_CODEVIEW
 - IMPORT Directory Table
 - IMPORT Name Table
 - IMPORT Hints/Names & DLL Names
 - SECTION .data
 - SECTION .tls
 - SECTION .rsrc

pFile	Data	Description	Value
00007910	0040C000	Start Address of Raw Data	
00007914	0040C001	End Address of Raw Data	
00007918	0040AC40	Address of Index	
0000791C	00408114	Address of Callbacks	
00007920	00000000	Size of Zero Fill	
00007924	00000000	Characteristics	



What's TLS Callback Function?

- **TLS (Thread Local Storage) callbacks** are provided by the Windows operating system to support additional **initialization and termination** for per-thread data structures.
- TLS callback functions **allow malware authors to execute malicious code before the debugger** has a chance to pause at the traditional **Entry Point (EP)**.

```
3389  typedef VOID (CALLBACK *PIMAGE_TLS_CALLBACK)(  
3390          LPVOID DllHandle,DWORD Reason,LPVOID Reserved  
3391  );
```

```
BOOL WINAPI DllMain(  
    HINSTANCE hinstDLL, // handle to DLL module  
    DWORD fdwReason,    // reason for calling function  
    LPVOID lpReserved ) // reserved
```

```

BOOL WINAPI DllMain(
    HINSTANCE hinstDLL, // handle to DLL module
    DWORD fdwReason,    // reason for calling function
    LPVOID lpReserved ) // reserved
{
    // Perform actions based on the reason for calling.
    switch( fdwReason )
    {
        case DLL_PROCESS_ATTACH:
            // Initialize once for each new process.
            // Return FALSE to fail DLL load.
            break;

        case DLL_THREAD_ATTACH:
            // Do thread-specific initialization.
            break;

        case DLL_THREAD_DETACH:
            // Do thread-specific cleanup.
            break;

        case DLL_PROCESS_DETACH:
            // Perform any necessary cleanup.
            break;
    }
    return TRUE; // Successful DLL_PROCESS_ATTACH.
}

```

TLS Callback Function Example2: TlsTest.exe

```
1  #include <windows.h>
2
3  #pragma comment(linker, "/INCLUDE:__tls_used")
4
5  void print_console(char* szMsg)
6  {
7      HANDLE hStdout = GetStdHandle(STD_OUTPUT_HANDLE);
8
9      WriteConsoleA(hStdout, szMsg, strlen(szMsg), NULL, NULL);
10 }
11
12 void NTAPI TLS_CALLBACK1(PVOID DllHandle, DWORD Reason, PVOID Reserved)
13 {
14     char szMsg[80] = {0,};
15     wsprintfA(szMsg, "TLS_CALLBACK1() : DllHandle = %X, Reason = %d\n", DllHandle, Reason);
16     print_console(szMsg);
17 }
18
19 void NTAPI TLS_CALLBACK2(PVOID DllHandle, DWORD Reason, PVOID Reserved)
20 {
21     char szMsg[80] = {0,};
22     wsprintfA(szMsg, "TLS_CALLBACK2() : DllHandle = %X, Reason = %d\n", DllHandle, Reason);
23     print_console(szMsg);
24 }
25
26 #pragma data_seg(".CRT$XLX")
27 PIMAGE_TLS_CALLBACK pTLS_CALLBACKs[] = { TLS_CALLBACK1, TLS_CALLBACK2, 0 };
28 #pragma data_seg()
29
```

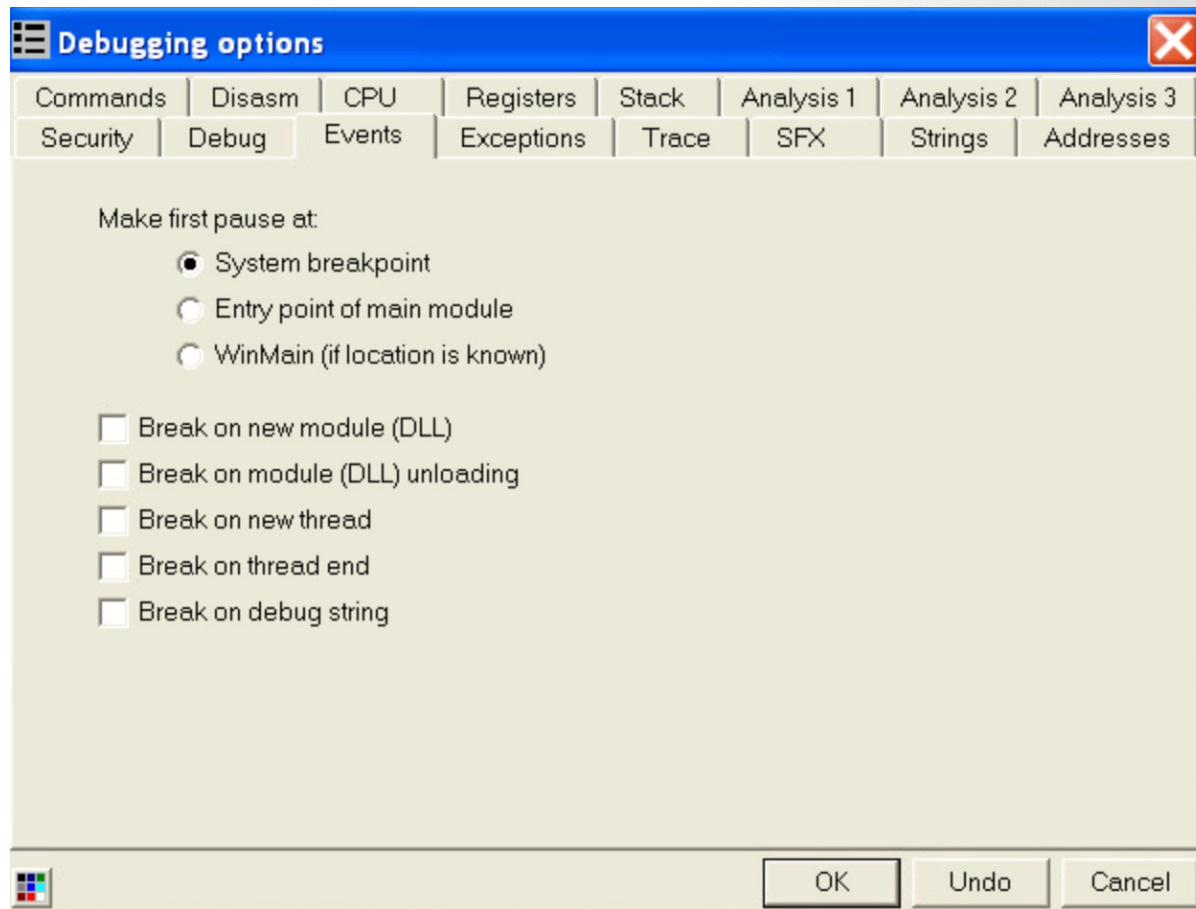
TLS Callback Function Example2: TlsTest.exe

```
30 DWORD WINAPI ThreadProc(LPVOID lParam)
31 {
32     print_console("ThreadProc() start\n");
33
34     print_console("ThreadProc() end\n");
35
36     return 0;
37 }
38
39 int main(void)
40 {
41     HANDLE hThread = NULL;
42
43     print_console("main() start\n");
44
45     hThread = CreateThread(NULL, 0, ThreadProc, NULL, 0, NULL);
46     WaitForSingleObject(hThread, 60*1000);
47     CloseHandle(hThread);
48
49     print_console("main() end\n");
50
51     return 0;
52 }
--
```

TLS Callback Function Example2: TlsTest.exe

```
C:\Documents and Settings\sec0day\My Documents\Downloads\Tls>TlsTest.exe
TLS_CALLBACK1() : DllHandle = 400000, Reason = 1
TLS_CALLBACK2() : DllHandle = 400000, Reason = 1
main() start
TLS_CALLBACK1() : DllHandle = 400000, Reason = 2
TLS_CALLBACK2() : DllHandle = 400000, Reason = 2
ThreadProc() start
ThreadProc() end
TLS_CALLBACK1() : DllHandle = 400000, Reason = 3
TLS_CALLBACK2() : DllHandle = 400000, Reason = 3
main() end
TLS_CALLBACK1() : DllHandle = 400000, Reason = 0
TLS_CALLBACK2() : DllHandle = 400000, Reason = 0
```


How to debug TLS callback functions



Garbage Code

- Add lots of garbage (meaningless) code to increase the difficulty of code debugging.
- Hide the real code inside these garbage code.

0040B2D3	52	PUSH EDX
0040B2D4	87D2	XCHG EDX,EDX
0040B2D6	87F6	XCHG ESI,ESI
0040B2D8	5A	POP EDX
0040B2D9	87E4	XCHG ESP,ESP
0040B2DB	87D2	XCHG EDX,EDX
0040B2DD	0FAFFE	IMUL EDI,ESI
0040B2E0	0FC1F1	XADD ECX,ESI
0040B2E3	89E4	MOV ESP,ESP
0040B2E5	53	PUSH EBX
0040B2E6	5B	POP EBX
0040B2E7	56	PUSH ESI
0040B2E8	55	PUSH EBP
0040B2E9	89C0	MOV EAX,EAX
0040B2EB	5D	POP EBP
0040B2EC	87F6	XCHG ESI,ESI
0040B2EE	5E	POP ESI
0040B2EF	50	PUSH EAX
0040B2F0	55	PUSH EBP
0040B2F1	C1D6 45	RCL ESI,45
0040B2F4	90	NOP
0040B2F5	87DB	XCHG EBX,EBX
0040B2F7	D1D6	RCL ESI,1
0040B2F9	52	PUSH EDX
0040B2FA	5A	POP EDX
0040B2FB	87C9	XCHG ECX,ECX
0040B2FD	87D2	XCHG EDX,EDX
0040B2FF	87FF	XCHG EDI,EDI
0040B301	F2:	PREFIX REPNE:
0040B302	58	POP EAX
0040B303	89E4	MOV ESP,ESP
0040B305	F7D6	NOT ESI

0040412D	0FC0C2	XADD DL,AL
00404130	81EB 94200000	SUB EBX,2094
00404136	0FAFC8	IMUL ECX,EAX
00404139	85DA	TEST EDX,EBX
0040413B	87C8	XCHG EAX,ECX
0040413D	53	PUSH EBX
0040413E	81C3 E7020000	ADD EBX,2E7
00404144	D1D1	RCL ECX,1
00404146	0FB7FD	MOVZX EDI,BP
00404149	69FE 1C6F7661	IMUL EDI,ESI,61766F1C
0040414F	81C3 19040000	ADD EBX,419
00404155	0FBEC6	MOVSX EAX,DH
00404158	25 3C0F9601	AND EAX,1960F3C
0040415D	88F0	MOV AL,DH
0040415F	81C3 9E0E0000	ADD EBX,0E9E
00404165	0FA5C1	SHLD ECX,EAX,CL
00404168	8BCD	MOV ECX,EBP
0040416A	D1D1	RCL ECX,1
0040416C	81EB 88040000	SUB EBX,488
00404172	0FA5F7	SHLD EDI,ESI,CL
00404175	F7C3 EFF6E120	TEST EBX,20E1F6EF
0040417B	FECA	DEC DL
0040417D	53	PUSH EBX
0040417E	81EB 75050801	SUB EBX,1080575
00404184	0FC9	BSWAP ECX
00404186	0FB AFF 4C	BTC EDI,4C
0040418A	B9 DC2F3621	MOV ECX,21362FDC
0040418F	81C3 5FF40701	ADD EBX,107F45F
00404195	F6D8	NEG AL
00404197	C7C1 FCCF56C1	MOV ECX,C156CFFC
0040419D	0FC0C2	XADD DL,AL
004041A0	- FFE3	JMP EBX

Break Code Alignment

0041510F	FFE3	JMP EBX	EBX = 415117
00415111	C9	LEAVE	
00415112	C2 0800	RETN 8	
00415115	A3 687201FF	MOV DWORD PTR DS:[FF017268],EAX	
0041511A	5D	POP EBP	
0041511B	33C9	XOR ECX,ECX	
0041511D	41	INC ECX	
0041511E	E2 17	LOOPD SHORT 00415137	
00415120	EB 07	JMP SHORT 00415129	
00415122	EA EB01EBEB 0DFF	JMP FAR FF0D:EBEB01EB	Far jump
00415129	E8 01000000	CALL 0041512F	
0041512E	EA 5A83EA0B FFE2	JMP FAR E2FF:0BEA835A	Far jump
00415135	EB 04	JMP SHORT 0041513B	
00415137	9A EB0400EB FBFF	CALL FAR FFFB:EB0004EB	Far call
0041513E	E8 02000000	CALL 00415145	
00415143	A0 005A81EA	MOV AL,BYTE PTR DS:[EA815A00]	
00415148	45	INC EBP	
00415149	51	PUSH ECX	
0041514A	0100	ADD DWORD PTR DS:[EAX],EAX	
0041514C	83EA FE	SUB EDX,-2	

A3: MOV

A368 -- 7201

00415117	72 01	JB SHORT 0041511A	
00415119	FF5D 33	CALL FAR FWORD PTR SS:[EBP+33]	Far call
0041511C	C9	LEAVE	
0041511D	41	INC ECX	
0041511E	E2 17	LOOPD SHORT 00415137	

Encryption/Decryption

```
0040B000    MOV ECX,100
0040B005    MOV ESI,0040B010
0040B00A    XOR BYTE PTR DS:[ESI],7F
0040B00D    INC ESI
0040B00E    LOOPD SHORT 0040B00A
0040B010    POP DS
0040B011    XCHG EAX,EDI
0040B012    JG SHORT 0040B093
0040B014    JG SHORT 0040B095
```

Stolen Bytes (Remove OEP)

00401041	. 55	PUSH EBP
00401042	. 8BEC	MOV EBP,ESP
00401044	. 6A FF	PUSH -1
00401046	. 68 B0604000	PUSH stolen_b.004060B0
00401048	. 68 88264000	PUSH stolen_b.00402688
00401050	. 64:A1 00000000	MOV EAX,DWORD PTR FS:[0]
00401056	. 50	PUSH EAX
00401057	. 64:8925 00000000	MOV DWORD PTR FS:[0],ESP
0040105E	. 83EC 10	SUB ESP,10
00401061	. 53	PUSH EBX
00401062	. 56	PUSH ESI
00401063	. 57	PUSH EDI
00401064	. 8965 E8	MOV DWORD PTR SS:[EBP-18],ESP
00401067	. FF15 04604000	CALL DWORD PTR DS:[<&KERNEL32.GetVersion
0040106D	. 33D2	XOR EDX,EDX
0040106F	. 8AD4	MOV DL,AH
00401071	. 8915 18994000	MOV DWORD PTR DS:[409918],EDX
00401077	. 8BC8	MOV ECX,EAX
00401079	. 81E1 FF000000	AND ECX,0FF
0040107F	. 890D 14994000	MOV DWORD PTR DS:[409914],ECX
00401085	. C1E1 08	SHL ECX,8
00401088	. 03CA	ADD ECX,EDX
0040108A	. 890D 10994000	MOV DWORD PTR DS:[409910],ECX
00401090	. C1E8 10	SHR EAX,10
00401093	. A3 0C994000	MOV DWORD PTR DS:[40990C],EAX
00401098	. 6A 00	PUSH 0

00401040	AB
00401041	45
00401042	9E
00401043	D4
00401044	0F
00401045	74
00401046	4C
00401047	56
00401048	84
00401049	24
0040104A	5F
0040104B	83
0040104C	E1
0040104D	E9
0040104E	48
0040104F	A0
00401050	9F
00401051	06
00401052	04
00401053	3D
00401054	25
00401055	A5
00401056	B9
00401057	26
00401058	1D
00401059	F8

AB
45
9E
D4
0F
74
4C
56
84
24
5F
83
E1
E9
48
A0
9F
06
04
3D
25
A5
B9
26
1D
F8

PESpin v1.32

File Help

PE Spin Settings

Anti-debugger protection

☐ Add Debugger detection

☒ Exit without any message

☐ Display message and exit

Debugger detected please close it down and restart

Protection

☒ API Redirection

☒ Antidump protection

☒ Remove OEP

☒ Code redirection

☐ Debug Blocker

Advanced

☒ Compress resources

☒ Strip Overlays

☒ Strip .reloc section if possible

☐ Optimize MS-DOS header size

Other

☐ Password protection

☒ Create backup file

☐ Close program after: minute

Section names

☒ User name:

☐ Random known protector, packer

☐ Don't rename

DB AB
DB 45
DB 9E
DB D4
DB 0F
DB 74
DB 4C
DB 56
DB 84
DB 24
DB 5F
DB 83
DB E1
DB E9
DB 48
DB A0
DB 9F
DB 06
DB 04
DB 3D
DB 25
DB A5
DB B9
DB 26
DB 1D
DB F8

- The code protector first copy all (or part) of main API code to a different memory region. And change the code that calls these APIs.

Debug Blocker (Self Debugging)

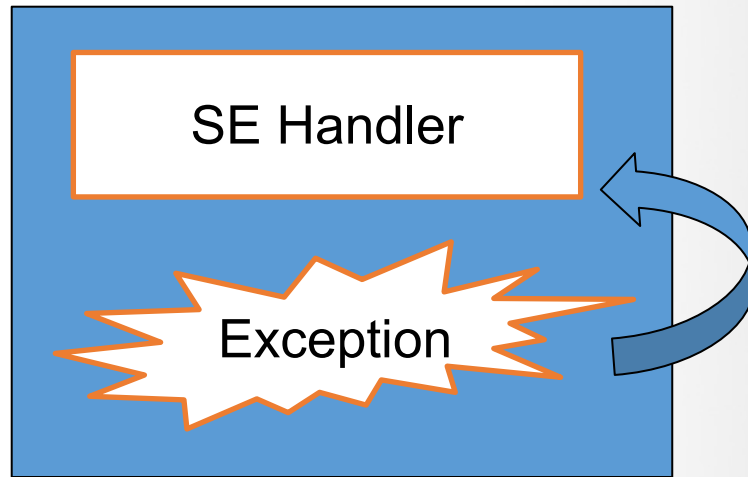
Process Explorer - Sysinternals: www.sysinternals.com [SEC0DAY-AFE0089\sec0day] (Administrator)

Process	C...	Priva...	WORK...	PID	Description	Company
System Idle Process	4...	0 K	28 K	0		
System	6...	0 K	236 K	4		
Interrupts	<...	0 K	0 K	n/a	Hardware Int...	
smss.exe		172 K	428 K	572	Windows NT...	Microsoft C...
explorer.exe		24,70...	13,92...	1268	Windows Ex...	Microsoft C...
vmtoolsd.exe		4,468 K	8,420 K	692	VMware Tool...	VMware, In
tvnserver.exe		936 K	3,456 K	1720	TightVNC Se...	GlavSoft LL
ctfmon.exe		1,040 K	3,712 K	1780	CTF Loader	Microsoft C...
firefox.exe		180,2...	187,6...	664	Firefox	Mozilla Cor.
Dbgview.exe		1,112 K	924 K	3268	DebugView	Sysinternals
PESpin(1.32).exe		592 K	1,640 K	3072	Freeware PE...	cyberbob
PESpin(1.32).exe		2,200 K	7,432 K	2092	Freeware PE...	cyberbob
OLLYDBG.EXE		8,372 K	848 K	2712	OllyDbg, 32-...	
HelloTls.exe		600 K	120 K	3272		
PEview.exe		3,076 K	628 K	2160	PE/COFF Fil...	Wayne J. R
cmd.exe		2,140 K	2,900 K	3912		
procexp.exe	4...	18,08...	26,87...	2360	Sysinternals ...	Sysinternal

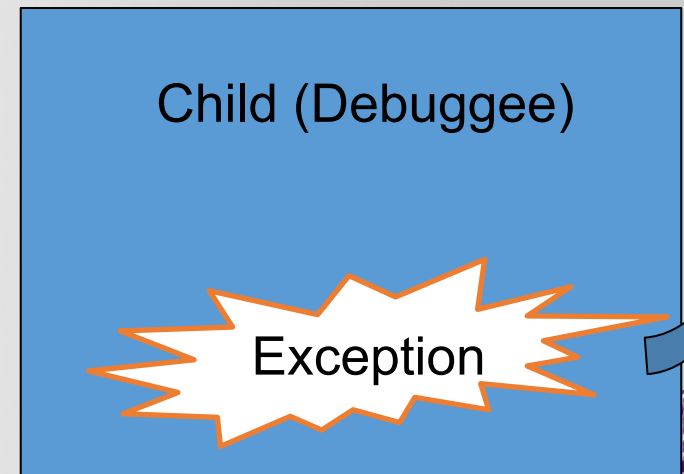
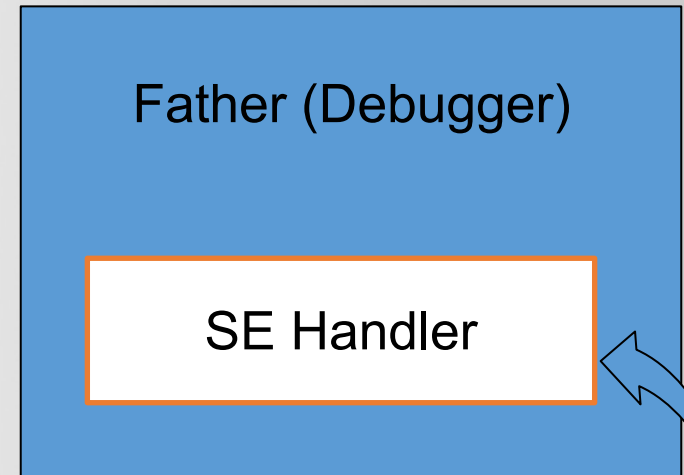
Debugger: PID 3072

Debuggee: PID 2092

General Anti-Debugging VS Debug Blocker



General Anti-Debugging



Debug Blocker

- Derived from Debug Blocker
- Replace all Jcc (jump) to INT3 (0xCC)
- The Debugger maintains a table to store all addresses for each Jcc.

0040122B	881D 14854000	MOV BYTE PTR DS:[408514],BL
00401231	8DC0	LEA EAX,EAX
00401233	A1 F0894000	MOV EAX,DWORD PTR DS:[4089F0]
00401238	85C0	TEST EAX,EAX
0040123A	8DC0	LEA EAX,EAX
0040123C	8B0D EC894000	MOV ECX,DWORD PTR DS:[4089EC]
00401242	56	PUSH ESI
00401243	8D71 FC	LEA ESI,DWORD PTR DS:[ECX-4]
00401246	3BF0	CMP ESI,EAX
00401248	8DC0	LEA EAX,EAX
0040124A	8DC0	LEA EAX,EAX
0040124C	85C0	TEST EAX,EAX
0040124E	8DC0	LEA EAX,EAX
00401250	FFD0	CALL EAX
00401252	83EE 04	SUB ESI,4
00401255	3B35 F0894000	CMP ESI,DWORD PTR DS:[4089F0]
0040125B	8DC0	LEA EAX,EAX
0040125D	5E	POP ESI
0040125E	68 18604000	PUSH 00406018

Q & A

