



Fun With Thread Local Sto

Peter Ferrie

Senior Anti-virus Researcher

18 June, 2008



You Can Call Me AI

Thread Local Storage callbacks were discovered in 2000.
However, widespread use didn't occur until 2004.
Now, it should be the first place to look for code,
since it runs before the main entrypoint.
And that can make all the difference...

Microsoft® Malware Protection Center

Threat Research and Response



Empty!

Hex Workshop - [tts1.exe]

File Edit Disk Options Tools Window Help

B S L O F D

00000000 4D5A 5000 0200 0000 0400 0F00 FFFF 0000 B800 0000 0000 0000 4000 1A00 0000 0000 MZP.....@.....
00000020 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0001 0000!..L..! This program must
00000040 BA10 000E 1FB4 09CD 21E8 014C CD21 9090 5468 6973 2070 726F 6772 616D 206D 7573 t be run under Win32..\$?
00000060 7420 6265 2072 756E 2075 6E64 6572 2057 696E 3332 0D0A 2437 0000 0000 0000 0000
00000080 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
000000A0 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
000000C0 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
000000E0 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
00000100 5045 0000 4C01 0300 C538 527F 0000 0000 0000 0000 0000 E000 8F81 0E01 0219 0002 0000 PE .L . 8R.....@.....@.....
00000120 0004 0000 0000 0000 1D10 0000 0010 0000 0020 0000 0000 4000 0010 0000 0002 0000
00000140 0100 0000 0000 0000 0300 0A00 0000 0000 0040 0000 0004 0000 0000 0000 0300 0000
00000160 0000 1000 0020 0000 0000 1000 0010 0000 0000 0000 1000 0000 0000 0000 0000 0000
00000180 0030 0000 5200 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
000001A0 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
000001C0 0020 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
000001E0 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 434F 4445 0000 0000 0000
00000200 0010 0000 0010 0000 0002 0000 0006 0000 0000 0000 0000 0000 0000 0000 2000 0060
00000220 4441 5441 0000 0000 0010 0000 0020 0000 0002 0000 0008 0000 0000 0000 0000 0000 0000
00000240 0000 0000 4000 00C0 2E69 6461 7461 0000 0010 0000 0030 0000 0002 0000 000A 0000
00000260 0000 0000 0000 0000 0000 0000 4000 00C0 0000 0000 0000 0000 0000 0000 0000 0000
00000280 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
000002A0 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
000002C0 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
000002E0 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
00000300 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
00000320 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
00000340 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
00000360 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
00000380 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
000003A0 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
000003C0 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
000003E0 0000 0000 0000 0000 0000 nnnn nnnn nnnn nnnn nnnn nnnn nnnn 0000 0000 0000 0000
00000400 0000 0000 0000 0000 0000 00C 00 0000 0000 0000 0000 0000 0000 0000 0000 0000
00000420 0000 0000 0000 0000 0000 00C 00 0000 0000 0000 0000 0000 0000 0000 0000 0000
00000440 0000 0000 0000 0000 0000 00C 00 0000 0000 0000 0000 0000 0000 0000 0000 0000
00000460 0000 0000 0000 0000 0000 00C 00 0000 0000 0000 0000 0000 0000 0000 0000 0000
00000480 0000 0000 0000 0000 0000 00C 00 0000 0000 0000 0000 0000 0000 0000 0000 0000
000004A0 0000 0000 0000 0000 0000 00C 00 0000 0000 0000 0000 0000 0000 0000 0000 0000
000004C0 0000 0000 0000 0000 0000 00C 00 0000 0000 0000 0000 0000 0000 0000 0000 0000
000004E0 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
00000500 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
00000520 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
00000540 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
00000560 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
00000580 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
000005A0 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
000005C0 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
000005E0 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
00000600 6A00 6828 2040 0668 2D20 4000 6A00 E80B 0000 00C7 0520 2040 0000 1040 0C3 125
00000620 3030 4000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
Ready Offset: 0000 Value: 23117 4096 bytes OVR MOD READ

Entry Point



Empty!

So the main file does nothing.
If we assume that the structure is normal,
then we could check the thread local storage table.
Just in case.



Empty!

The screenshot shows the Hex Workshop interface with a memory dump. The address range is from 00000600 to 0000C200. The data is displayed in hexadecimal and ASCII columns. Two specific memory locations are highlighted with red circles and arrows:

- Callback pointer:** Located at address 000007E0, containing the value 1C20 4000.
- Callback array:** Located at address 00000800, containing the value 1310 4000.

Red text labels "Callback pointer" and "Callback array" are placed below the respective annotations. The status bar at the bottom indicates "Offset: 00000600", "Value: 106", "4096 bytes", and "OVR MOD READ".



Empty!

So the search moves to the callbacks,
of which there is only one... or is there?



The One and Only

```
IDA - tls1.exe
File Edit Jump Search View Debug Options Window IDA View-A
[ ]
; ===== S U B R O U T I N E =====
CODE:00401013 public TlsCallback_0
CODE:00401013 TlsCallback_0 proc near ; DATA XREF: DATA:TlsCallbacks↓
CODE:00401013 mov ds:TlsCallbacksEnd, offset loc_401000
CODE:00401013 TlsCallback_0 endp ; sp-analysis failed
CODE:0040101D
CODE:0040101D ; ===== S U B R O U T I N E =====
CODE:0040101D
CODE:0040101D
CODE:0040101D
CODE:0040101D public start
CODE:0040101D start proc near ; DATA XREF: HEADER:pe_header↑o
CODE:0040101D start retn
CODE:0040101D start endp
CODE:0040101E ; [00000006 BYTES: COLLAPSED FUNCTION j_MessageBoxA. PRESS KEYPAD "+" TO EXPAND]
CODE:00401024 dd 77h dup(0)
CODE:00401200 dd 380h dup(?)
CODE:00401200 CODE ends
CODE:00401200
DATA:00402000 ; Section 2. (virtual address 00002000)
DATA:00402000 ; Virtual size : 00001000 ( 4096.)
DATA:00402000 ; Section size in file : 00000200 ( 512.)
DATA:00402000 ; Offset to raw data for section: 00000800
DATA:00402000 ; Flags C0000040: Data Readable Writable
DATA:00402000 ; Alignment : default
DATA:00402000 ;
DATA:00402000 ; Segment type: Pure data
DATA:00402000 ; Segment permissions: Read/Write
DATA:00402000 DATA segment para public 'DATA' use32
DATA:00402000 assume cs:DATA
DATA:00402000 ;org 402000h
DATA:00402000 TlsDirectory TLS_DIR_ENTRY <0, 0, offset TlsIndex, offset TlsCallbacks, 0, 0>
DATA:00402018 TlsIndex dd 0 ; DATA XREF: HEADER:pe_header↑o HEADER:00400220↑o
DATA:0040201C TlsCallbacks dd offset TlsCallback_0 ; DATA XREF: DATA:TlsDirectory↑o
DATA:00402020 TlsCallbacksEnd dd 0 ; DATA XREF: DATA:TlsDirectory↑o
DATA:00402024 align 8 ; DATA XREF: TlsCallback_0↑w
DATA:00402028 aDemo db 'demo', 0 ; DATA XREF: CODE:00401002↑o
DATA:0040202D aRun db 'run', 0 ; DATA XREF: CODE:00401007↑o
DATA:00402031 align 1000h
DATA:00402031 DATA ends
DATA:00402031
```



Am I Missing Somethi

```
CODE:00401013      mov    ds:TlsCallbacksEnd, offset loc_401000
CODE:0040101D      retn
```

Who ever heard of a one-line callback?



Write the Right

It's about what you write, and where you write it.
By writing to `TlsCallbacksEnd`, the array is extended in memory.
Now the array contains two entries, not one.



Surprise!





Not OK

The second entry is executed after the first one returns.

The array can be extended infinitely.

Existing entries can be altered at runtime, too.

For example, one entry can decrypt the others.



Really Not OK

Just a little something to add to the workload.