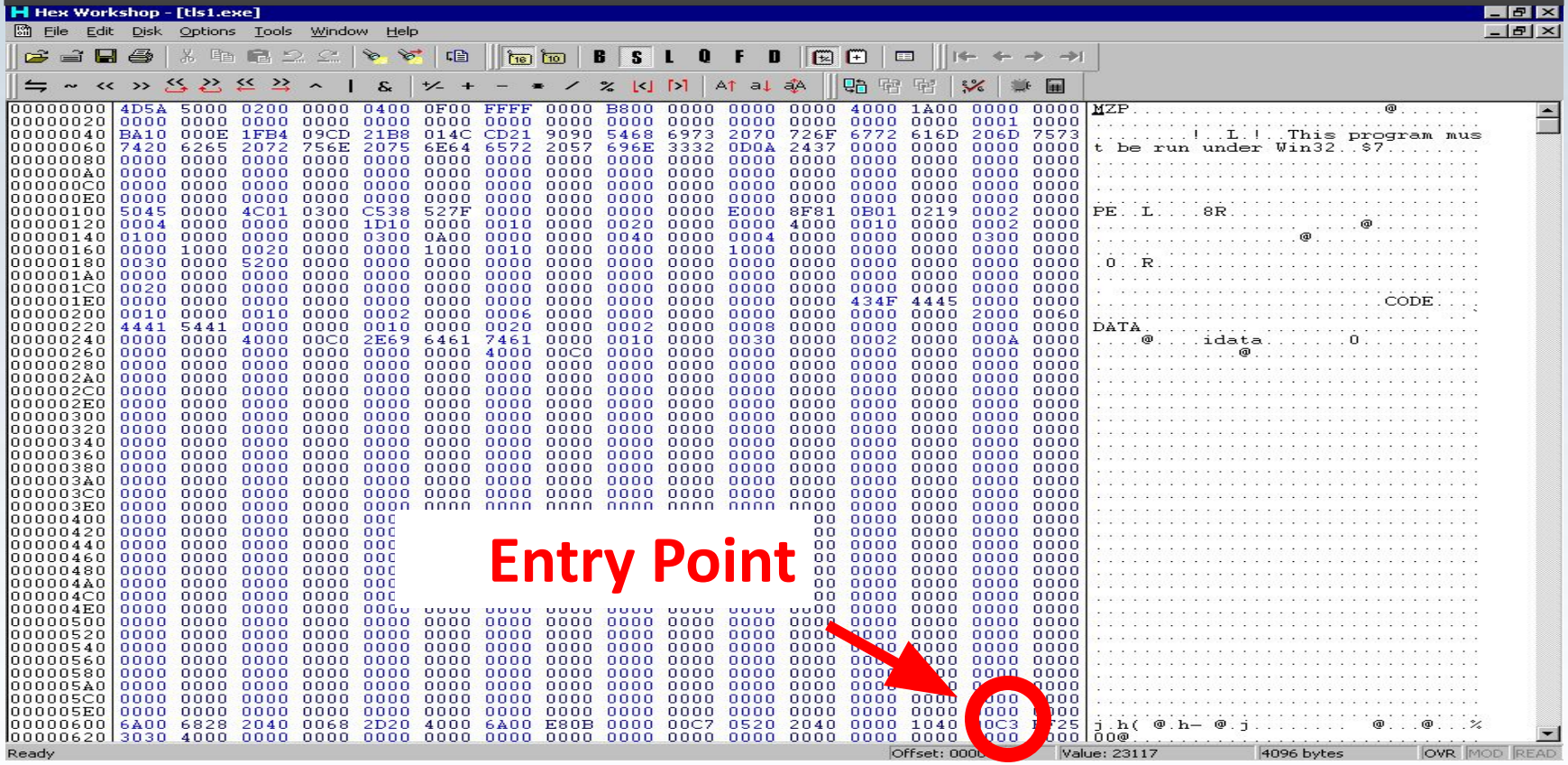
Fun With Thread Local Sto

Peter Ferrie
Senior Anti-virus Researcher
18 June, 2008

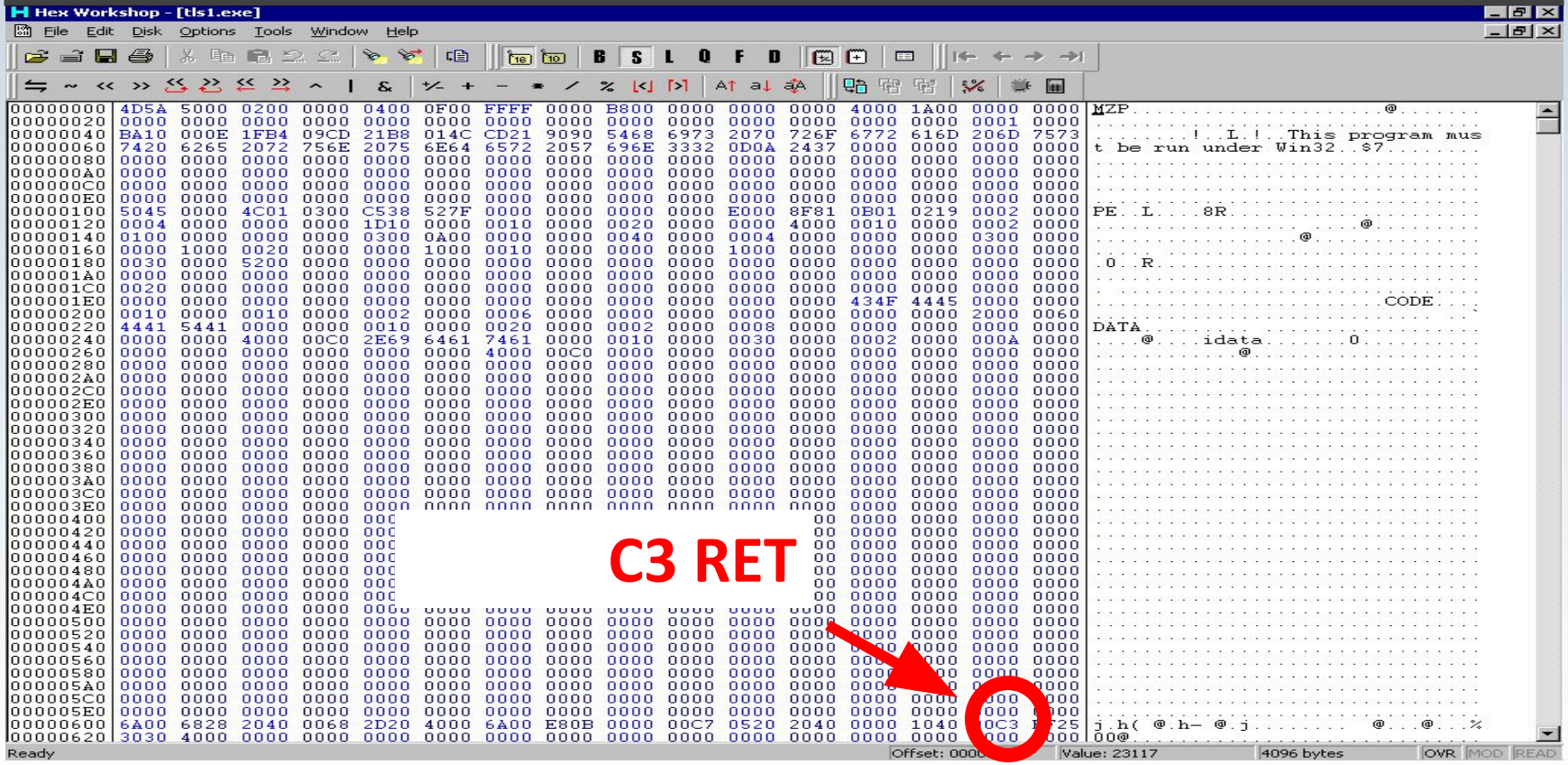
You Can Call Me Al

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…

Empty!


Hex Workshop - [tls1.exe]
File Edit Disk Options Tools Window Help
Entry Point
Ready

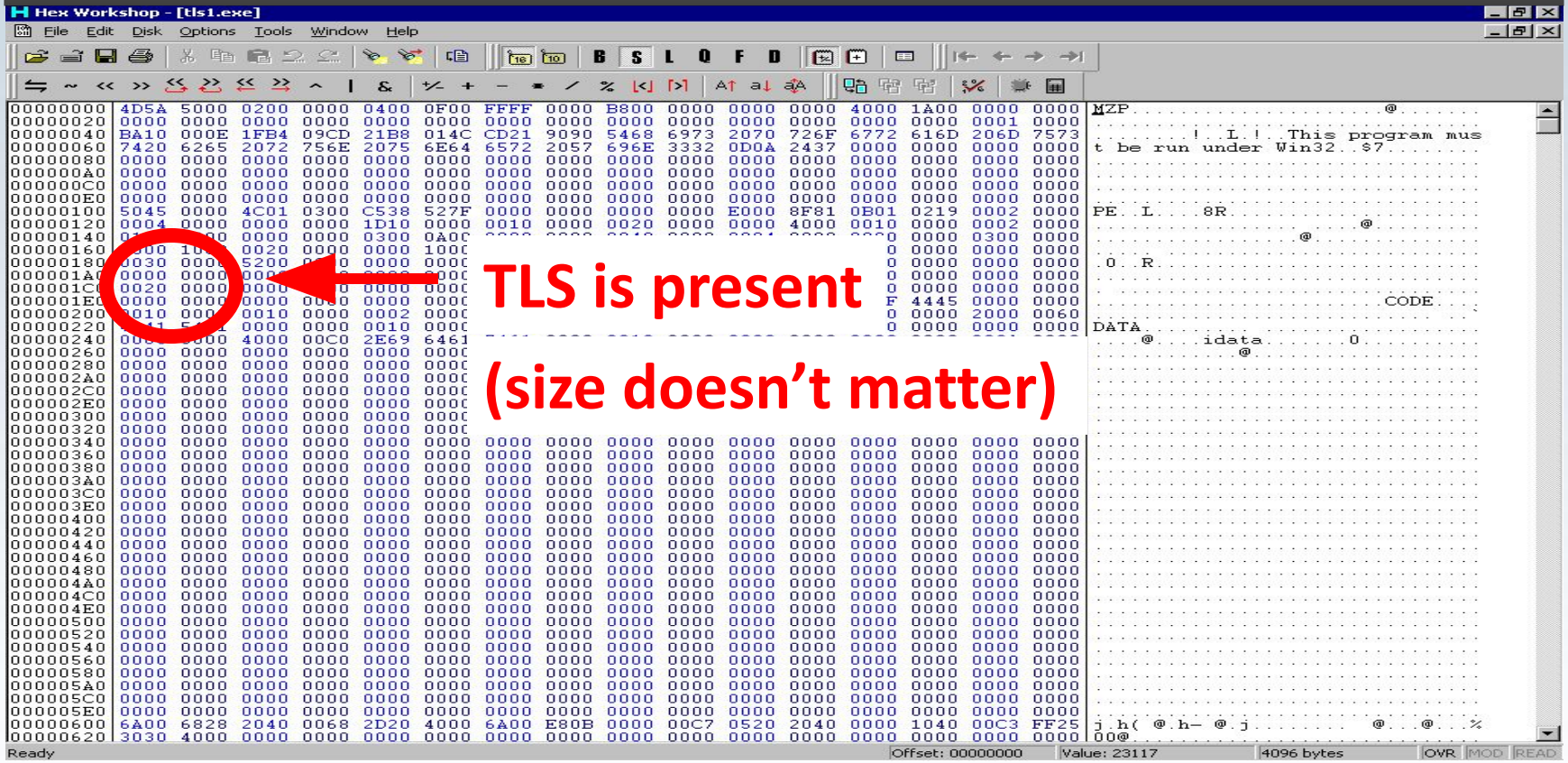
Empty!


Hex Workshop - [tls1.exe]
File Edit Disk Options Tools Window Help
C3 RET
Ready
Offset: 0006
Value: 23117
4096 bytes
OVR
MOD
READ

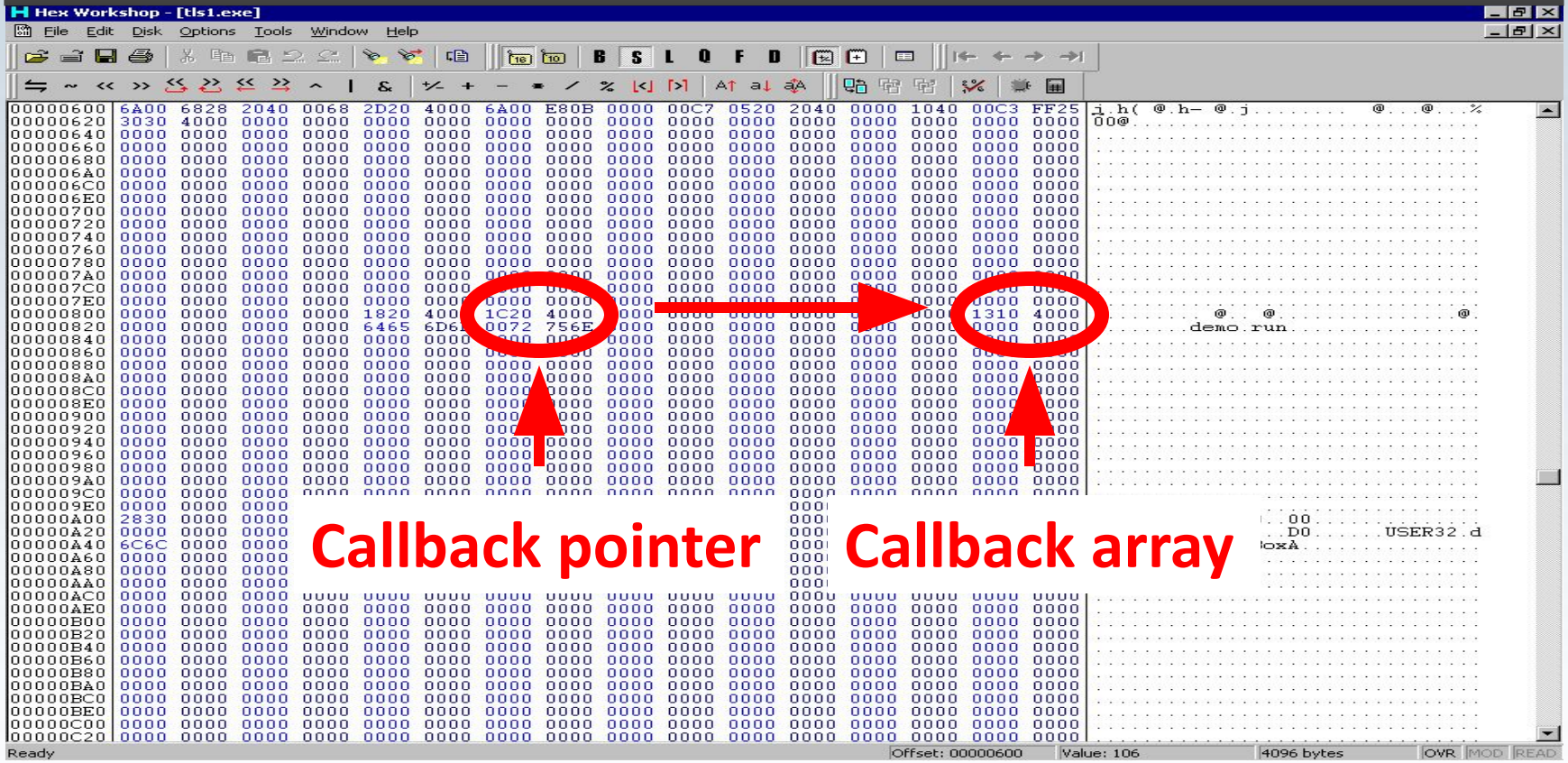
Empty!

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!


Hex Workshop - [tls1.exe]
File Edit Disk Options Tools Window Help
TLS is present
(size doesn't matter)
Ready
Offset: 00000000
Value: 23117
4096 bytes
OVR MOD READ

Empty!


Hex Workshop - [tls1.exe]
File Edit Disk Options Tools Window Help
demo run
Callback pointer
Callback array
Ready
Offset: 00000600
Value: 106
4096 bytes
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 =======================================


                        public TlsCallback_0
TlsCallback_0           proc near               ; DATA XREF: DATA:TlsCallbacks↓o
CODE:00401013           mov     ds:TlsCallbacksEnd, offset loc_401000
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                   public start
CODE:0040101D start             proc near               ; DATA XREF: HEADER:pe_header↑o
CODE:0040101D                   retn
CODE:0040101D start             endp
CODE:0040101D
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
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:00402000                                           ; DATA XREF: HEADER:pe_header↑o HEADER:004002201o
DATA:00402018 TlsIndex          dd 0                    ; DATA XREF: DATA:TlsDirectory↑o
DATA:0040201C TlsCallbacks      dd offset TlsCallback_0 ; DATA XREF: DATA:TlsDirectory↑o
DATA:00402020 TlsCallbacksEnd   dd 0                    ; DATA XREF: TlsCallback_0↑w
DATA:00402024                   align 8
DATA:00402028 aDemo             db 'demo',0             ; DATA XREF: CODE:004010021↑o
DATA:0040202D aRun              db 'run',0              ; DATA XREF: CODE:004010071↑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.