

The background is a dark blue gradient with a subtle pattern of white dots. Overlaid on the left side are several concentric circles and a large circular scale. The scale has numerical markings from 140 to 260 in increments of 10, with smaller tick marks between them. Several curved arrows are scattered across the image, some pointing clockwise and others counter-clockwise, suggesting a sense of motion or analysis.

MALWARE ANALYSIS

ANALISI STATICA - IDA PRO

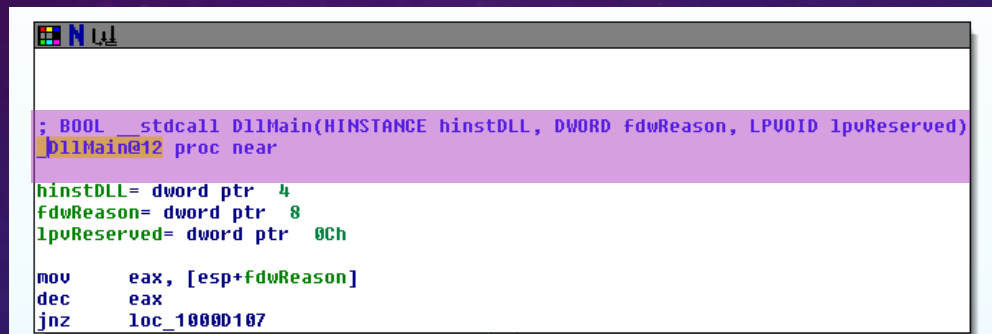
S11/L2

Lo scopo dell'esercizio di oggi è di acquisire esperienza con IDA, un tool fondamentale per l'analisi statica. A tal proposito, con riferimento al malware chiamato «Malware_U3_W3_L2» presente all'interno della cartella «Esercizio_Pratico_U3_W3_L2» sul desktop della macchina virtuale dedicata all'analisi dei malware, rispondere ai seguenti quesiti, utilizzando IDA Pro.

- ☐ Individuare l'indirizzo della funzione DLLMain
- ☐ Dalla scheda «imports» individuare la funzione «gethostbyname». Qual è l'indirizzo dell'import?
- ☐ Quante sono le variabili locali della funzione alla locazione di memoria 0x10001656?
- ☐ Quanti sono, invece, i parametri della funzione sopra?

❑ Individuare l'indirizzo della funzione DLLMain

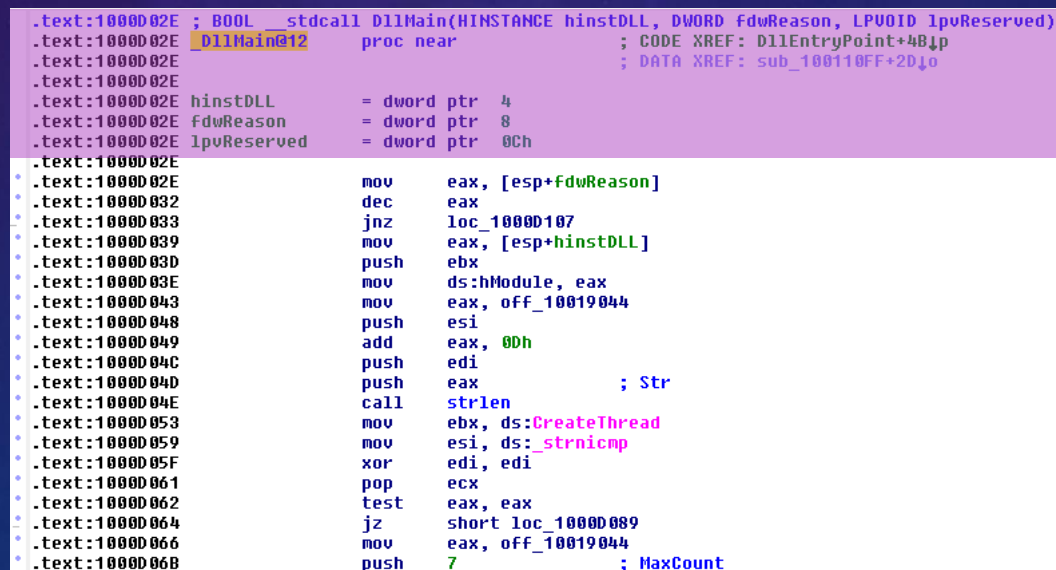
Nella prima parte di questa esercitazione utilizzeremo il software disassembler IDA Pro gratuito per analizzare il nostro malware e tradurre il codice macchina in codice Assembly. Dopo aver caricato il malware nel programma, esamineremo il codice tradotto. Abbiamo identificato rapidamente la porzione di codice e l'indirizzo della funzione DLLMain, che si trova all'indirizzo di memoria 1000D02E, come mostrato nelle immagini seguenti.



```
; BOOL __stdcall DLLMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved)
DLLMain@12 proc near

hinstDLL= dword ptr 4
fdwReason= dword ptr 8
lpvReserved= dword ptr 0Ch

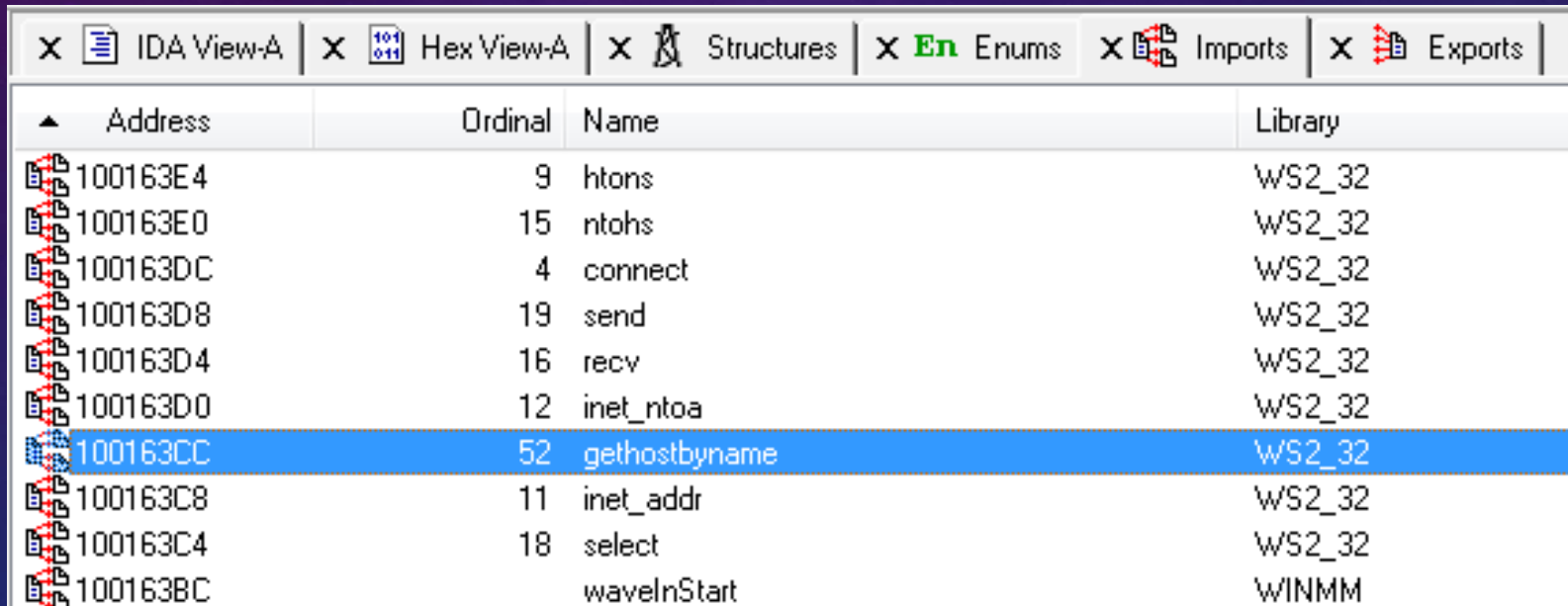
mov     eax, [esp+fdwReason]
dec     eax
jnz     loc_1000D107
```



```
.text:1000D02E ; BOOL __stdcall DLLMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved)
.text:1000D02E DLLMain@12 proc near ; CODE XREF: DllEntryPoint+4B↓p
.text:1000D02E ; DATA XREF: sub_100110FF+2D↓p
.text:1000D02E hinstDLL = dword ptr 4
.text:1000D02E fdwReason = dword ptr 8
.text:1000D02E lpvReserved = dword ptr 0Ch
.text:1000D02E
.text:1000D02E mov     eax, [esp+fdwReason]
.text:1000D032 dec     eax
.text:1000D033 jnz     loc_1000D107
.text:1000D039 mov     eax, [esp+hinstDLL]
.text:1000D03D push    ebx
.text:1000D03E mov     ds:hModule, eax
.text:1000D043 mov     eax, off_10019044
.text:1000D048 push    esi
.text:1000D049 add     eax, 0Dh
.text:1000D04C push    edi
.text:1000D04D push    eax ; Str
.text:1000D04E call    strlen
.text:1000D053 mov     ebx, ds:CreateThread
.text:1000D059 mov     esi, ds:_strnicmp
.text:1000D05F xor     edi, edi
.text:1000D061 pop     ecx
.text:1000D062 test    eax, eax
.text:1000D064 jz     short loc_1000D089
.text:1000D066 mov     eax, off_10019044
.text:1000D06B push    7 ; MaxCount
```

❑ Dalla scheda «imports» individuare la funzione «gethostbyname». Qual è l'indirizzo dell'import?

Nella fase successiva dell'analisi, abbiamo esaminato la scheda degli imports per identificare le funzioni importate dal malware. In questo modo, siamo riusciti a individuare anche l'indirizzo di memoria della funzione richiesta, gethostbyname, che è 100163CC.



Address	Ordinal	Name	Library
100163E4	9	htons	WS2_32
100163E0	15	ntohs	WS2_32
100163DC	4	connect	WS2_32
100163D8	19	send	WS2_32
100163D4	16	recv	WS2_32
100163D0	12	inet_ntoa	WS2_32
100163CC	52	gethostbyname	WS2_32
100163C8	11	inet_addr	WS2_32
100163C4	18	select	WS2_32
100163BC		waveInStart	WINMM

❑ Quante sono le variabili locali della funzione alla locazione di memoria 0x10001656?

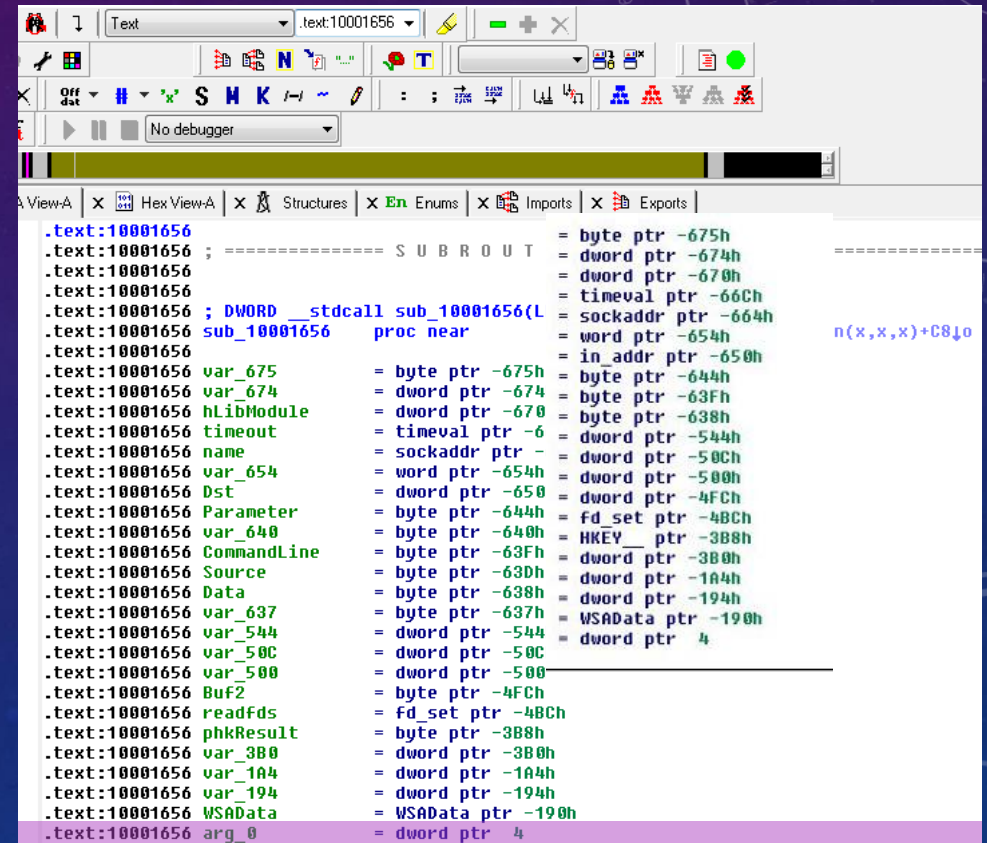
Nella fase finale dell'analisi, ci siamo concentrati sulla locazione di memoria 0x10001656. Nella casella search inseriamo 10001656 per filtrare i risultati

Qui abbiamo individuato 23 variabili locali, riconoscibili dai simboli negativi associati ad esse nell'interfaccia di IDA Pro. Tuttavia, abbiamo notato che c'è un solo parametro associato a un offset positivo.

❑ Quanti sono, invece, i parametri della funzione sopra?

Dalla stessa immagine, possiamo osservare che solo un argomento viene passato alla funzione.

Questo argomento ha un offset positivo rispetto al registro EBP. IDA Pro ha etichettato questo parametro come "arg_0".



```
.text:10001656 ; ===== S U B R O U T =====
.text:10001656 ;
.text:10001656 ;
.text:10001656 ; DWORD __stdcall sub_10001656(L
.text:10001656 sub_10001656 proc near
.text:10001656 ;
.text:10001656 var_675 = byte ptr -675h
.text:10001656 var_674 = dword ptr -674h
.text:10001656 hLibModule = dword ptr -670h
.text:10001656 timeout = timeval ptr -66Ch
.text:10001656 name = sockaddr ptr -664h
.text:10001656 var_654 = word ptr -654h
.text:10001656 Dst = in_addr ptr -650h
.text:10001656 Parameter = byte ptr -644h
.text:10001656 var_640 = dword ptr -640h
.text:10001656 CommandLine = byte ptr -63Fh
.text:10001656 Source = byte ptr -63Dh
.text:10001656 Data = byte ptr -638h
.text:10001656 var_637 = byte ptr -637h
.text:10001656 var_544 = dword ptr -544h
.text:10001656 var_50C = dword ptr -50Ch
.text:10001656 var_500 = dword ptr -500h
.text:10001656 Buf2 = byte ptr -4FCh
.text:10001656 readfds = fd_set ptr -4BCh
.text:10001656 phkResult = byte ptr -388h
.text:10001656 var_380 = dword ptr -380h
.text:10001656 var_1A4 = dword ptr -1A4h
.text:10001656 var_194 = dword ptr -194h
.text:10001656 WSADATA = WSADATA ptr -190h
.text:10001656 arg_0 = dword ptr 4
```