Analisi statica avanzata con IDA

Sommario

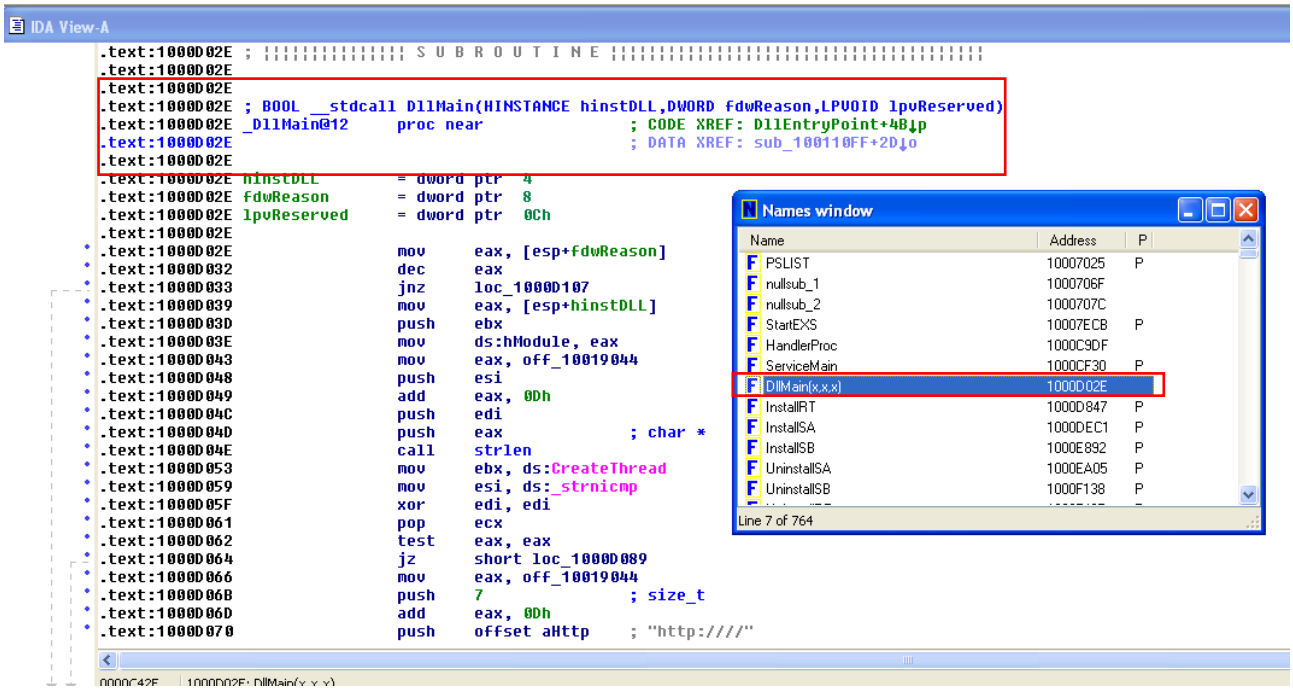
Traccia

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.

1. Step 1: individuare l'indirizzo della funzione DLLMain 2.
2. Step 2: dalla scheda «imports» individuare la funzione «gethostbyname». Qual è l'indirizzo dell'import?
3. Step 3: quante sono le variabili locali della funzione alla locazione di memoria 0x10001656?
4. Step 4: quanti sono, invece, i parametri della funzione sopra?

Step 1

Per questo compito ci torna utile la scheda ''names'' che associa ogni indirizzo ad un nome che può essere una funzione, una variabile, un parametro o una stringa.

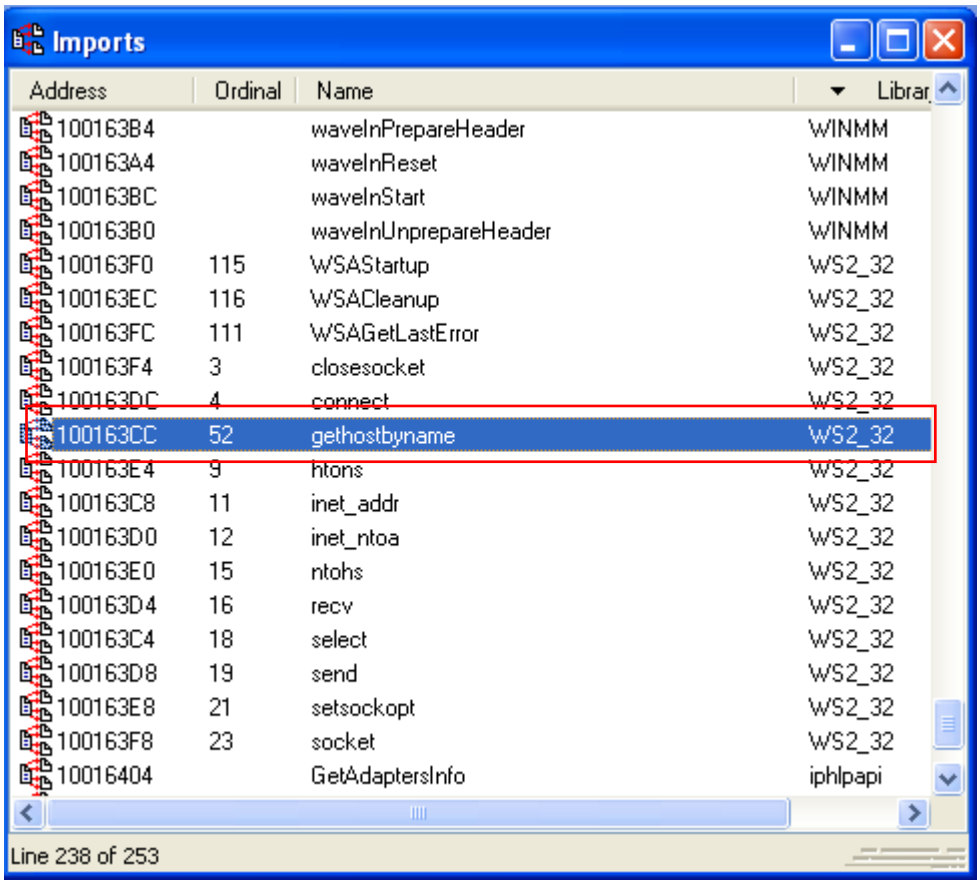


Come possiamo vedere la funzione main ha l'indirizzo 1000D02E.

Step 2

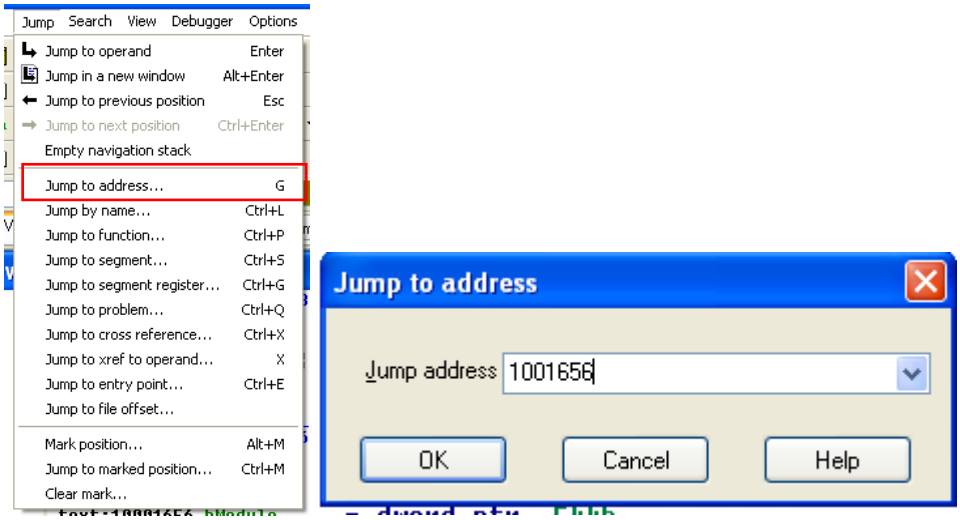
La scheda Imports mostra tutte le funzioni importate dall'eseguibile.

La funzione gethostbyname è una funzione dell'API Winsock utilizzata per recuperare le informazioni host corrispondenti a un nome host da un database host. Nel codice analizzato la funzione gethostbyname ha come indirizzo 100163CC ed è importata dalla libreria WS2_32.



Step 3

Per trovare la locazione di memoria 0x10001656 apriamo il menu a tendina Jump e selezioniamo jump to address. Successivamente inseriamo l'indirizzo e clicchiamo su ok.



```
.text:10001656
.text:10001656 ; ╔═══════════════ S U B R O U T I N E ═══════════════════════════════════════
.text:10001656
.text:10001656
.text:10001656 ; DWORD __stdcall sub_10001656(LPVOID)
.text:10001656 sub_10001656    proc near               ; DATA XREF: DllMain(x,x,x)+C8↓o
.text:10001656
.text:10001656 var_675         = byte ptr -675h
.text:10001656 var_674         = dword ptr -674h
.text:10001656 hModule         = dword ptr -670h
.text:10001656 timeout         = timeval ptr -66Ch
.text:10001656 name            = sockaddr ptr -664h
.text:10001656 var_654         = word ptr -654h
.text:10001656 in              = in_addr ptr -650h
.text:10001656 Parameter       = byte ptr -644h
.text:10001656 CommandLine     = byte ptr -63Fh
.text:10001656 Data            = byte ptr -638h
.text:10001656 var_544         = dword ptr -544h
.text:10001656 var_50C         = dword ptr -50Ch
.text:10001656 var_500         = dword ptr -500h
.text:10001656 var_4FC         = dword ptr -4FCh
.text:10001656 readfds         = fd_set ptr -4BCh
.text:10001656 phkResult       = HKEY__ ptr -3B8h
.text:10001656 var_3B0         = dword ptr -3B0h
.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
.text:10001656
.text:10001656                 sub     esp, 678h
```

Le variabili sono ad un offset negativo rispetto al registro EBP, quindi possiamo contare 20 variabili.

Step 4

```
.text:10001656
.text:10001656 ; ╔═══════════════ S U B R O U T I N E ═══════════════════════════════════════
.text:10001656
.text:10001656
.text:10001656 ; DWORD __stdcall sub_10001656(LPVOID)
.text:10001656 sub_10001656    proc near               ; DATA XREF: DllMain(x,x,x)+C8↓o
.text:10001656
.text:10001656 var_675         = byte ptr -675h
.text:10001656 var_674         = dword ptr -674h
.text:10001656 hModule         = dword ptr -670h
.text:10001656 timeout         = timeval ptr -66Ch
.text:10001656 name            = sockaddr ptr -664h
.text:10001656 var_654         = word ptr -654h
.text:10001656 in              = in_addr ptr -650h
.text:10001656 Parameter       = byte ptr -644h
.text:10001656 CommandLine     = byte ptr -63Fh
.text:10001656 Data            = byte ptr -638h
.text:10001656 var_544         = dword ptr -544h
.text:10001656 var_50C         = dword ptr -50Ch
.text:10001656 var_500         = dword ptr -500h
.text:10001656 var_4FC         = dword ptr -4FCh
.text:10001656 readfds         = fd_set ptr -4BCh
.text:10001656 phkResult       = HKEY__ ptr -3B8h
.text:10001656 var_3B0         = dword ptr -3B0h
.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
.text:10001656
.text:10001656                 sub     esp, 678h
```

I parametri rispetto alle variabili si trovano ad un offset positivo rispetto ad EBP, quindi contiamo un parametro.