

Pratica S11/L2

Giaimo Rosario

Traccia:

Lo scopo dell'esercizio di oggi è di acquisire esperienza con IDA, un tool fondamentale per l'analisi statica.

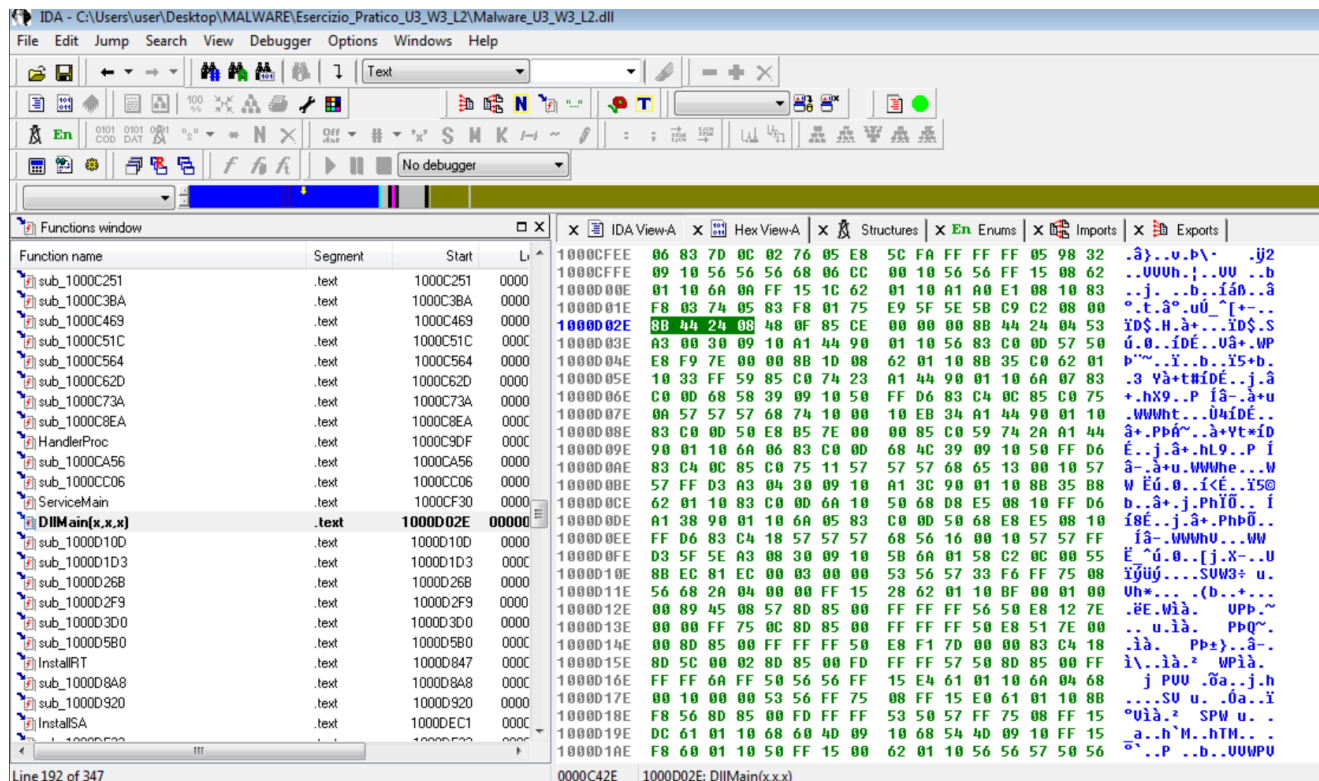
A tal proposito, con riferimento al malware chiamato «Malware_U3_W3_L2» presente all'interno della cartella «Esercizio_Pratico_U3_W3_L2» sul Desktop della macchina virtuale dedicata all'analisi dei malware, rispondere ai seguenti quesiti, utilizzando IDA Pro.

1. Individuare l'indirizzo della funzione DLLMain(così com'è, in esadecimale)
2. Dalla scheda «imports» individuare la funzione «gethostbyname». Qual è l'indirizzo dell'import? Cosa fa la funzione?
3. Quante sono le variabili locali della funzione alla locazione di memoria 0x10001656?
4. Quanti sono, invece, i parametri della funzione sopra?
5. Inserire altre considerazioni macro livello sul malware (comportamento)

Svolgimento

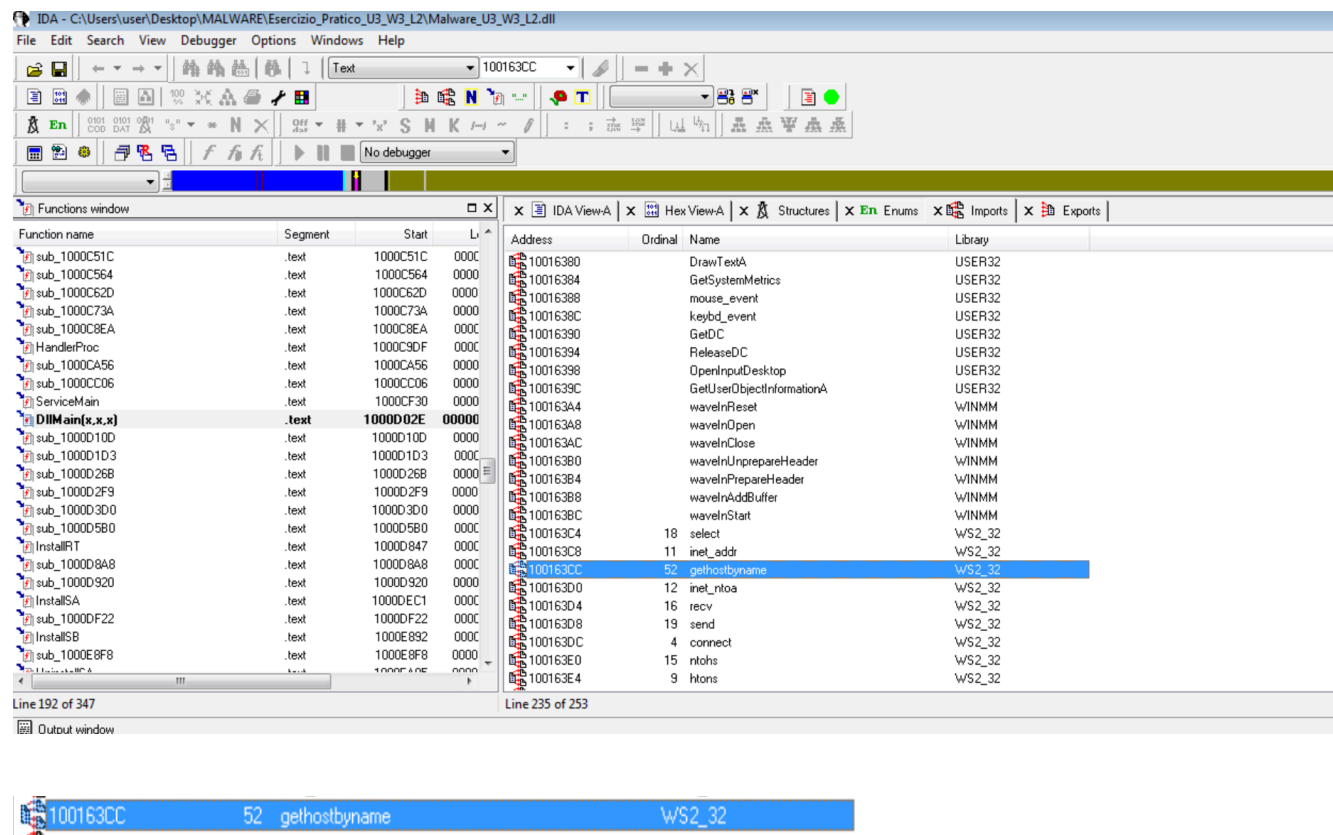
1. Individuare l'indirizzo della funzione DLLMain(così com'è, in esadecimale)

1000D02E



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

100163CC



La funzione gethostbyname è una funzione di libreria standard che viene utilizzata per risolvere un nome di dominio in un indirizzo IP.

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

A questo indirizzo troviamo 23 variabili con offset negativo

```

.text:10001656
.text:10001656 ; ===== S U B R O U T I N E =====
.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 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 = dword ptr -650h
.text:10001656 Parameter = byte ptr -644h
.text:10001656 var_640 = byte 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 -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

```

4. Quanti sono, invece, i parametri della funzione sopra?

A questo indirizzo troviamo un solo argomento passato alla funzione, avente offset positivo chiamato «arg_0».

```

.text:10001656
.text:10001656 ; ===== S U B R O U T I N E =====
.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 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 = dword ptr -650h
.text:10001656 Parameter = byte ptr -644h
.text:10001656 var_640 = byte 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 -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

```

5. Inserire altre considerazioni macro livello sul malware (comportamento)

- Esaminare il flusso di esecuzione del malware per individuare eventuali comportamenti sospetti o dannosi.
- Analizzare le chiamate di sistema, le funzioni importate e le operazioni di rete per comprendere le potenziali azioni dannose del malware.
- Valutare la presenza di tecniche evasive, come la crittografia o l'auto-modificazione del codice, che potrebbero indicare un tentativo di eludere l'analisi statica e dinamica.
- Considerare se il malware tenta di comunicare con server esterni o esegue azioni non autorizzate sul sistema, come la modifica dei file di sistema o la creazione di processi maligni.