Esercizio S11 L2

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

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

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

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

Risposta al quesito 1:

Per svolgere l'esercizio di oggi andremo ad usare un programma chiamato IDA Pro.

IDA Pro è un disassembler largamente usato per il reverse engineering. Supporta numerosi formati di file eseguibili per diversi processori e sistemi operativi.

Oggi lo andremo ad usare per fare del reverse engineering sul file "Malware_U3_W3_L2" ovvero per andare ad analizzare il codice Assembly che IDA traducerà per noi.

Andiamo dunque ad aprire la macchina per il malware analysis e facciamo doppio click sull'icona di IDA.

Risposta al quesito 1:

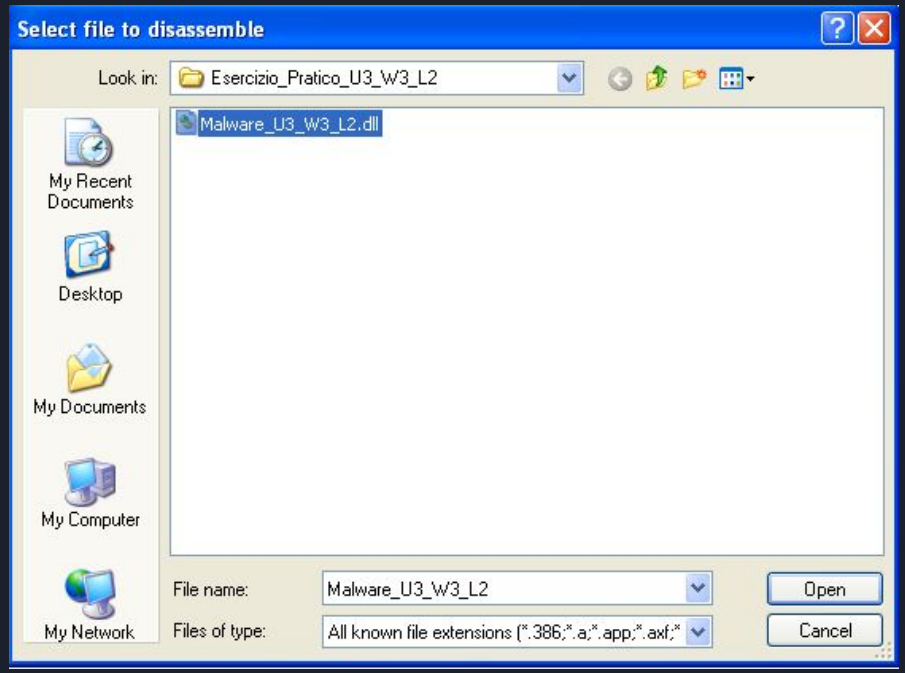
Una volta avviato il programma premiamo sull'icona gialla con la cartella in alto a sinistra per selezionare un file da analizzare.

Risposta al quesito 1:

Dopo aver fatto ciò si aprirà la seguente schermata e da qui inseriamo in alto il path che contiene il malware che vogliamo analizzare e premiamo poi su open.

Così facendo IDA aprirà il file malware e ci mostrerà il codice Assembly che lo compone.

A tal proposito il codice sarà mostrato in modalità grafica con gli schemi collegati da frecce, per l'esercizio di oggi vogliamo vedere il codice in versione testuale quindi premiamo una volta la barra spaziatrice dopo che si sarà caricato per farci mostrare il codice nella modalità che preferiamo.



Risposta al quesito 1:

```
.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↓o
.text:1000D02E
.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]
```

Una volta che il codice viene caricato possiamo cominciare a navigarlo per trovare la funzione DllMain.

Quando finalmente la troviamo possiamo vedere dall'immagine che essa è collocata all'indirizzo di memoria text:1000D02E

Risposta al quesito 2:

Andiamo adesso alla sezione imports per vedere quali sono le funzioni importate dal malware, per fare ciò premiamo sul riquadro con su scritto imports che si trova nella barra sopra al codice.

Fatto ciò si aprirà un riquadro contenente tutte le funzioni importate dal codice e cerchiamo tra queste una denominata “gethostbyname” , appena trovata possiamo vedere che essa risiede nell’indirizzo di memoria 100163CC.

100163C4	18	select	WS2_32
100163C8	11	inet_addr	WS2_32
100163CC	52	gethostbyname	WS2_32
100163D0	12	inet_ntoa	WS2_32

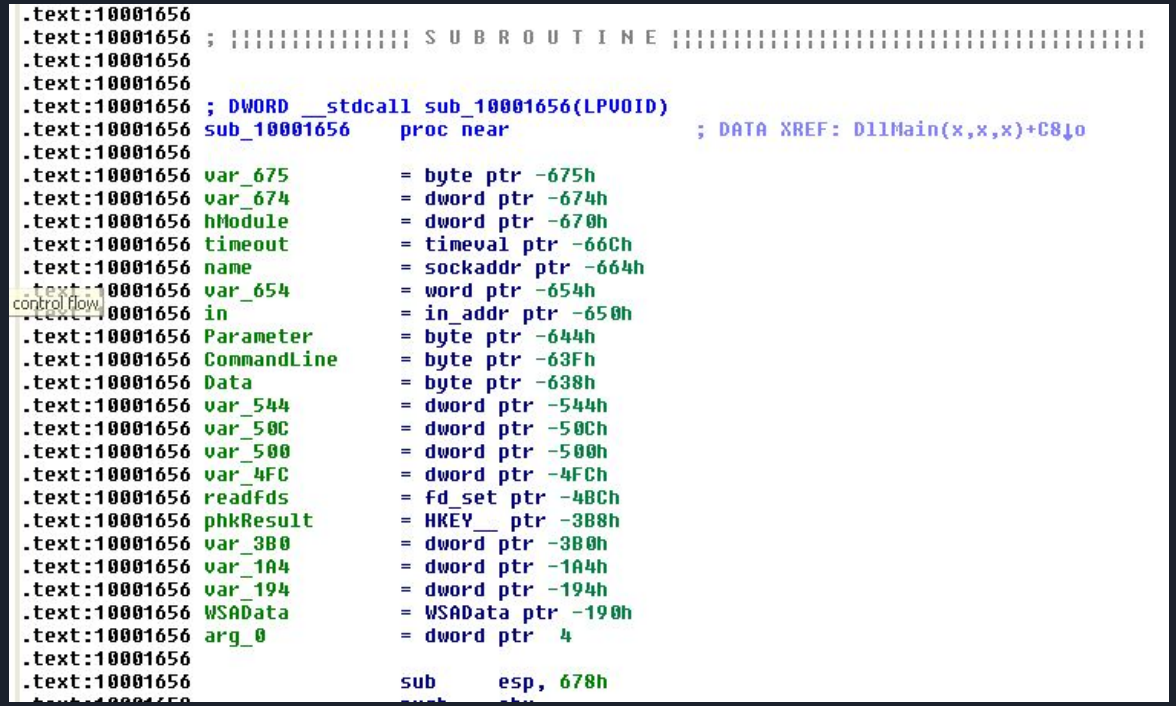
Risposta al quesito 3:

Per rispondere alla terza domanda abbiamo bisogno di tornare alla schermata col codice Assembly del nostro malware e navigare il codice fino a trovare l'indirizzo di memoria 0x10001656.

Risposta al quesito 3:

Una volta individuato l'indirizzo di memoria corretto possiamo notare che esso contiene molti elementi, l'esercizio ci chiede quanti di questi elementi corrispondono a variabili.

Per rispondere a questa domanda ci basta sapere che in Assembly le variabili sono identificate dal segno meno, quindi possiamo affermare con certezza che da var_675 a WSAData sono tutte variabili per un totale di 20 variabili.



Risposta al quesito 4:

Il quarto quesito ci chiede quanti di questi elementi sono parametri.

Come le variabili in Assembly i parametri sono identificati dall'assenza di segno negativo, ovvero dal fatto che hanno un valore di offset positivo rispetto alla funzione che li richiama.

Siccome arg_0 è l'unico elemento con offset positivo possiamo affermare con certezza che è anche l'unico parametro.