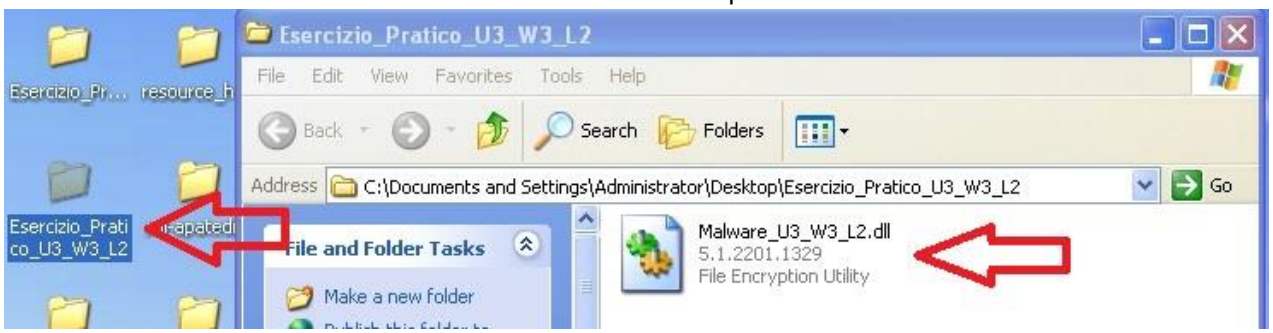
ANALISI AVANZATA CON IDA Pro

Traccia:

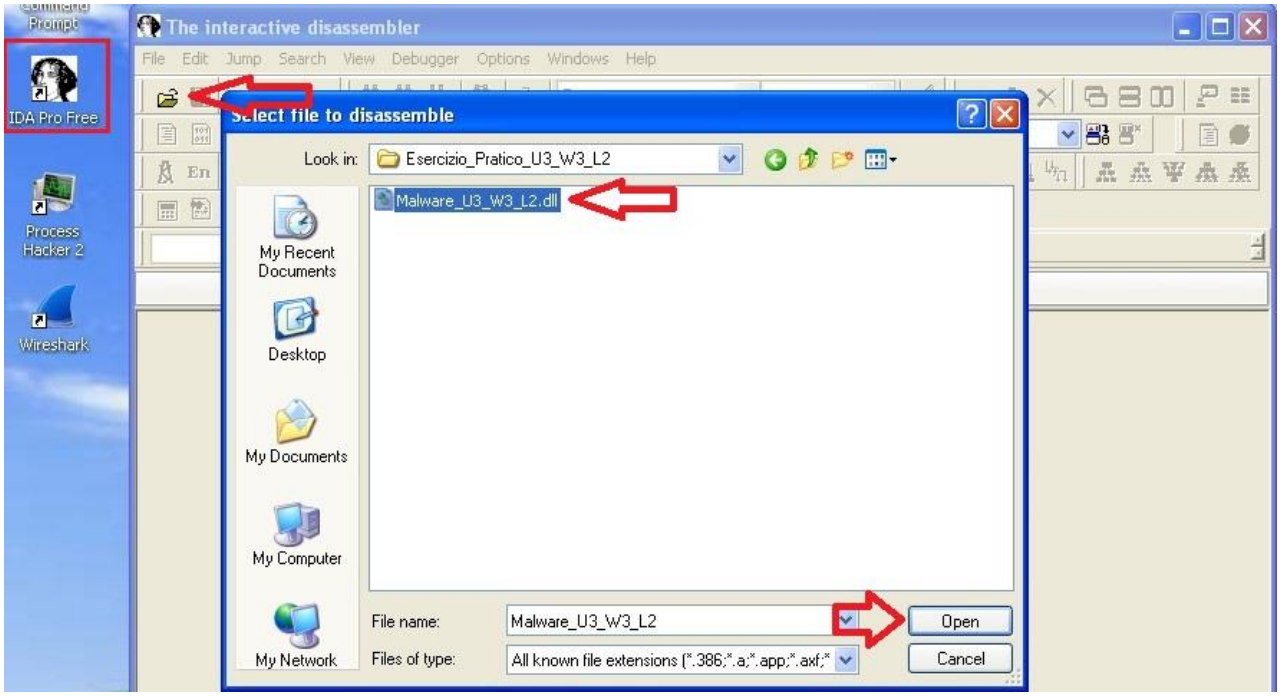
Lo scopo dell'esercizio di oggi è di acquisire esperienza con IDA, un tool fondamentale per l'analisi statica.
A tal proposito, con riferimento al malware chiamato «**Malware_U3_W3_L2**» presente all'interno della cartella «**Esercizio_Pratico_U3_W3_L2**» sul Desktop della macchina virtuale dedicata 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?
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

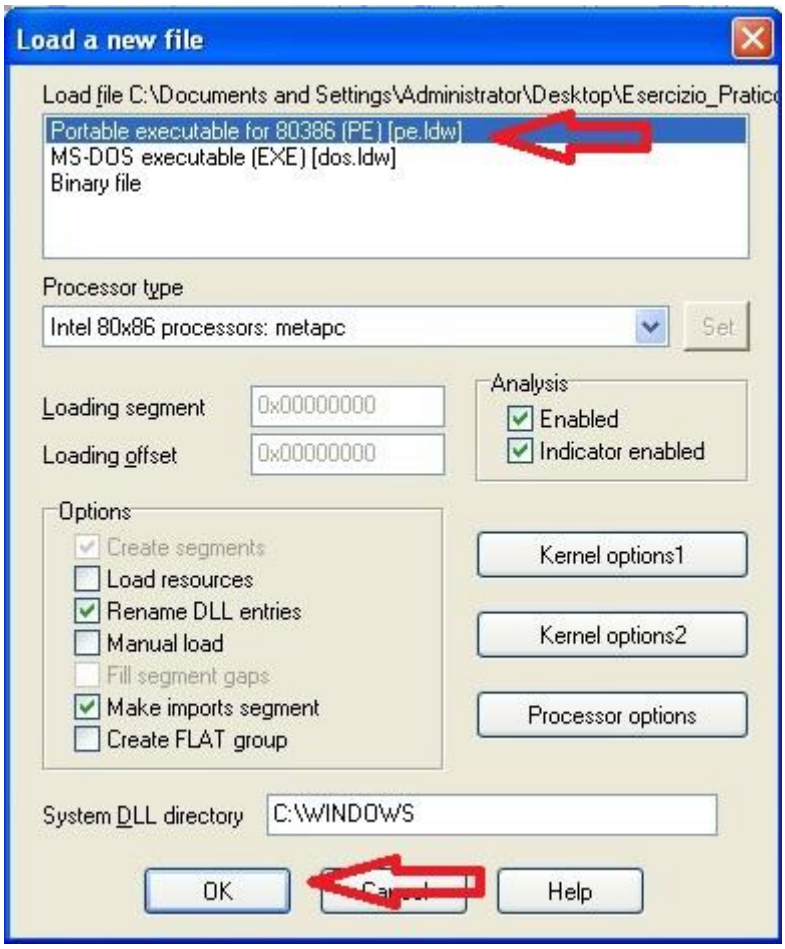
In relazione all'esercizio siamo andati ad avviare la nostra macchina Windows XP.
Come da traccia troviamo il malware nella cartella sul desktop.



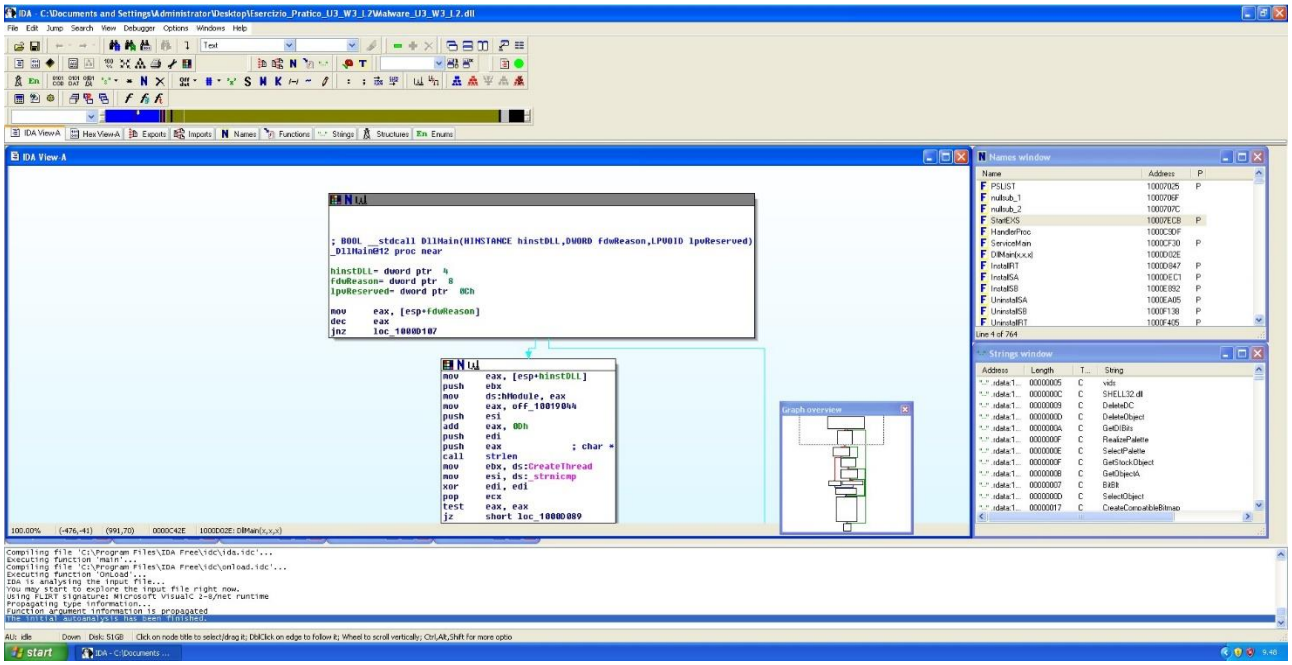
Andiamo quindi ad aprirlo con IDA Pro cliccando sull'icona della cartella e seguendo il path del malware.



IDA di default identificherà da sé sia l'architettura del processore (Intel x86) e il formato del file (PE).

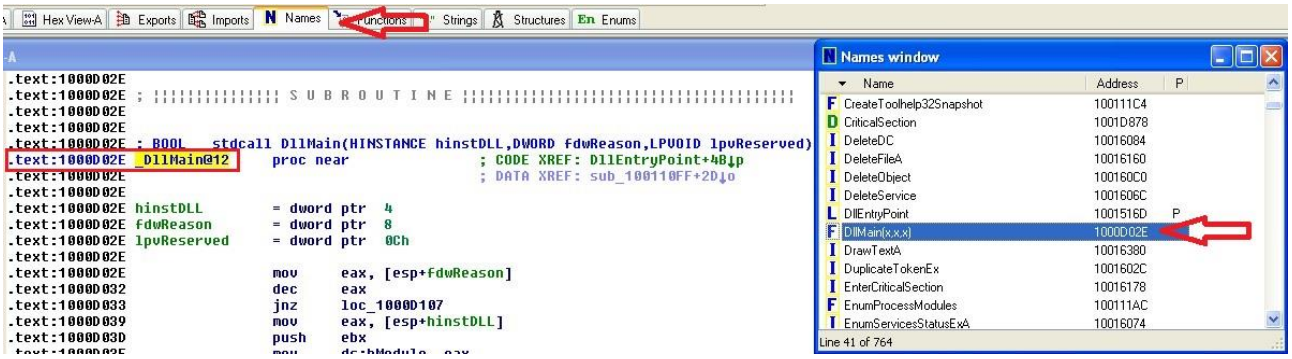


Una volta cliccato "OK" IDA ci mostrerà l'interfaccia di analisi.



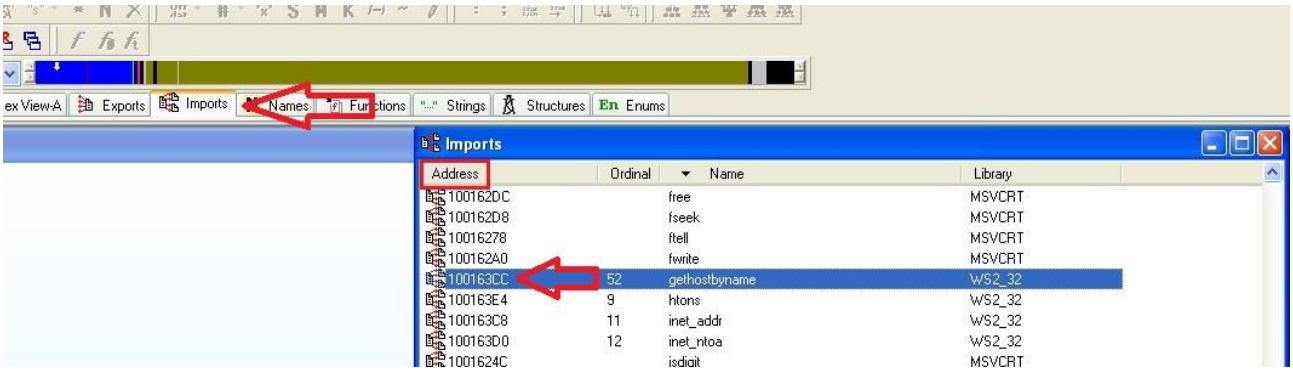
Andremo ora a rispondere ai quesiti.

1. **Individuare l'indirizzo della funzione DLLMain.**



Com'è possibile vedere dalla versione testuale del pannello e dalla scheda "**Names**" l'indirizzo della funzione in esame è **1000D2E**.

2. **Individuare la funzione "gethostbyname" e l'indirizzo dell'import.**



Per individuare l'indirizzo dell'import della funzione "**gethostbynames**" andremo ad aprire la scheda "**Imports**" che mostra tutte le funzioni importate dell'eseguibile. La scheda ci darà come informazioni l'indirizzo della funzione, che nel nostro caso sarà **100163CC**, il nome della funzione e la libreria.

Passiamo ora alla versione testuale del pannello grafico principale cliccando sulla barra spaziatrice della tastiera per rispondere ai quesiti 3 e 4.
Andremo quindi a spostarci alla locazione di memoria 0x10001656, che risulta essere una "subroutine" come da commento al codice, per analizzarla.

```
.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
```

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

```
.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
```

IDA Pro ha la capacità di riconoscere le funzioni e di evidenziare le variabili locali ed i parametri ad esse associate.
Le variabili sono definite localmente nel contesto di una funzione.
IDA Pro indica le variabili ed i parametri utilizzando come riferimento il **puntatore EBP;** per quanto riguarda le variabili esse sono ad un offset negativo rispetto al registro EBP, dove con offset si intende la differenza rispetto ad un valore di riferimento (EBP). Quindi i valori con il "-" saranno delle variabili, in questo caso ne troveremo **20**.

4. **Quanti sono i parametri della funzione alla locazione di memoria 0x10001656?**

```
.text:10001656 arg_0           = dword ptr  4
.text:10001656                 sub     esp, 678h
```

Come per le variabili locali IDA Pro utilizzando il puntatore EBP come valore di riferimento assegna un offset anche ai parametri, a differenza delle variabili però sarà positivo. Quindi i valori non negativi saranno parametri, nel nostro caso sarà presente **un solo parametro** alla locazione di memoria specificata, il parametro "**arg_0**".

5. **Considerazioni macro livello sul malware.**
Andando a cercare sulla scheda "Strings" essa ci mostrerà tutte le stringhe presenti all'interno del file eseguibile. Andandole a studiare siamo arrivati alla conclusione che il malware sia della categoria delle **backdoor**.