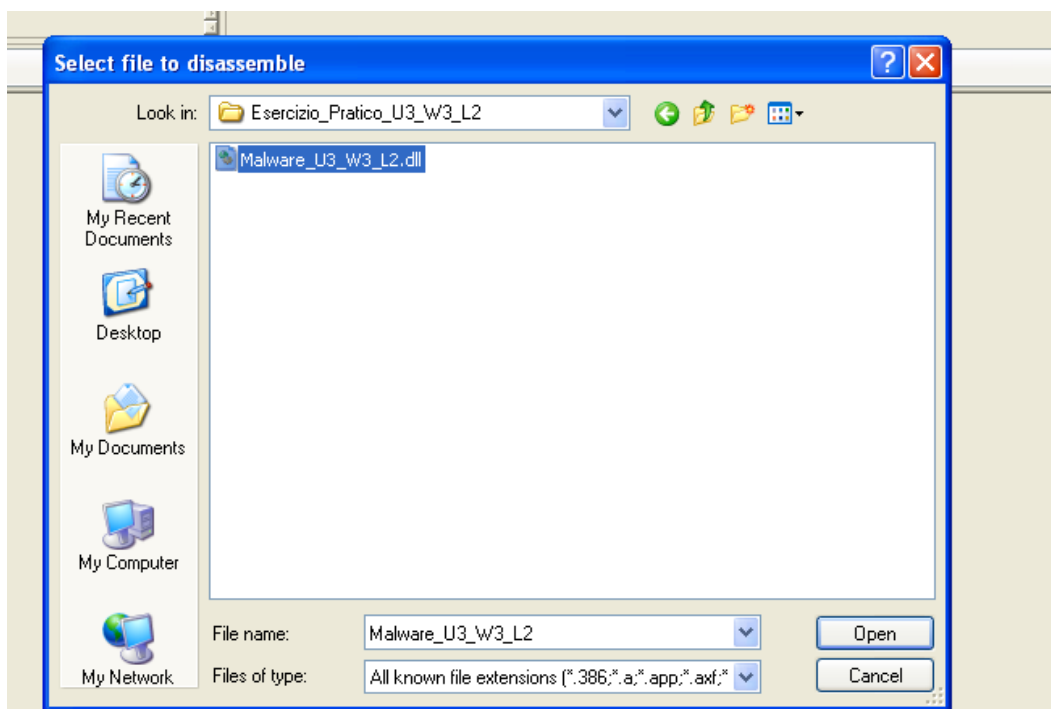


Esercizio 04-04

Traccia: 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:

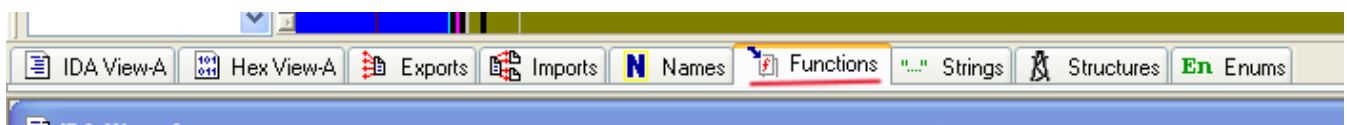
1. Individuare l’indirizzo della funzione DLLMain.
2. Dalla scheda “imports” individuare la funzione “gethostbyname”. Qual è l’indirizzo dell’import?
3. Quante sono le variabili locali della funzione alla locazione di memoria 0x10001656?
4. Quanti sono, invece, i parametri della funzione sopra?
5. Inserire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

1.

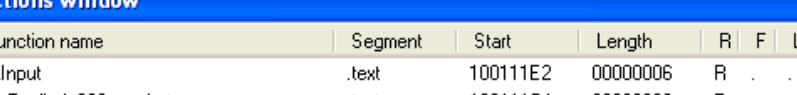


Dopo aver aperto il file “Malware_U3_W3_L2” con IDA Pro

Ed essere andati nella scheda “Functions”



Cerchiamo la funzione "DLLMain"



Function name	Segment	Start	Length	R	F	L	S	B	T
BlockInput	.text	100111E2	00000006	R
CreateToolhelp32Snapshot	.text	100111C4	00000006	R	T
DilEntryPoint	.text	1001516D	0000009D	R	.	L	.	B	T
DilMain(x,x,x)	.text	1000D02E	000000DF	R	T
EnumProcessModules	.text	100111AC	00000006	R
GetAdaptersInfo	.text	100111B2	00000006	R
GetModuleFileNameExA	.text	100111A6	00000006	R
HandlerProc	.text	1000C9DF	00000077	R	T
ICClose	.text	100113D6	00000006	R	T
ICCompress	.text	100113D0	00000006	R	T

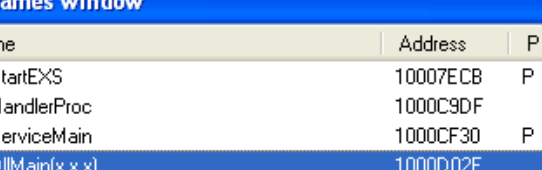
Cliccandoci due volte il pannello testuale si aggiorna e ci porta in automatico all'indirizzo della funzione cliccata

```

.text:1000D02E
.text:1000D02E ; :::::::::::::::::::::: SUBROUTINE ::::::::::::::::::::::::::::::
.text:1000D02E
.text:1000D02E
.text:1000D02E ; BOOL __stdcall DllMain(HINSTANCE hinstDLL,DWORD fdwReason,LPUVOID 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]

```

La funzione risulta presente anche nella scheda “Names” aperta in piccolo di default a lato del pannello principale. La “F” a sinistra del nome ci indica che il nome si riferisce a una Funzione



The screenshot shows the 'Names window' in Windows Task Manager. It displays a list of system services with columns for Name, Address, and P. The entry 'DLLMain(x,x,x)' is highlighted in blue.

Name	Address	P
StartEXS	10007ECB	P
HandlerProc	1000C9DF	
ServiceMain	1000CF30	P
DLLMain(x,x,x)	1000D02E	
InstallIRT	1000D847	P
InstallSA	1000DEC1	P
InstallSB	1000E892	P
UninstallSA	1000EA05	P
UninstallSB	1000F138	P
UninstallIRT	1000F405	P
StartAddress	10010740	
GetModuleFileNameExA	100111A6	

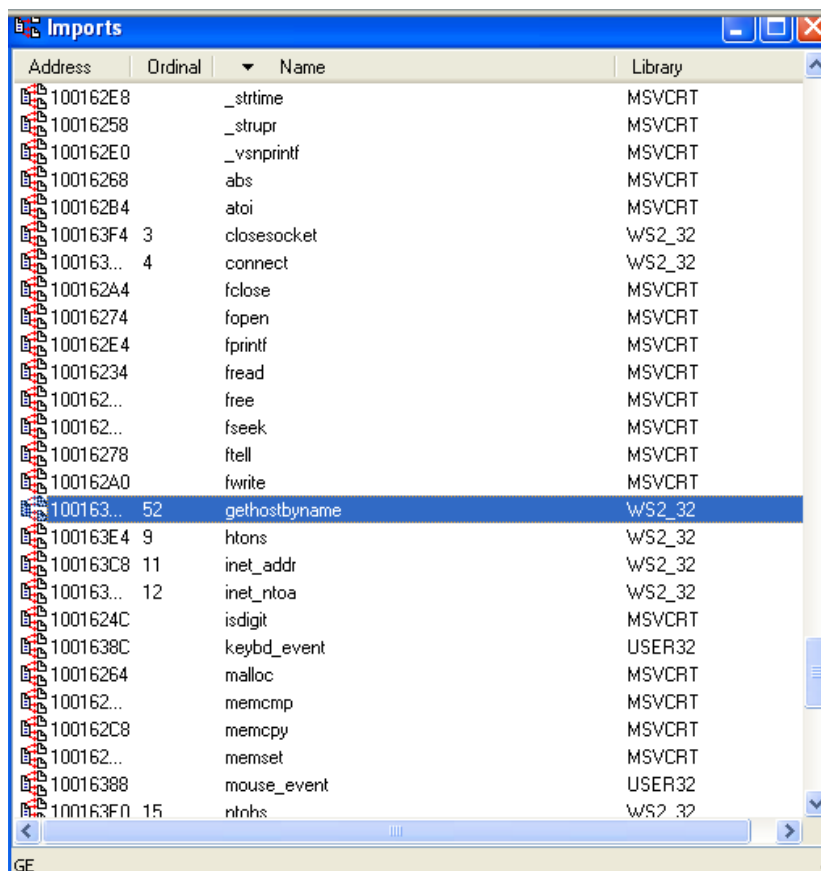
Line 7 of 764

2.

Al fine di individuare la funzione “gethostbyname” e scoprirne l’indirizzo ci spostiamo nella scheda “Imports”



Una volta aperta la scheda, basta digitare il nome e... inserendo soltanto “GE” la funzione che




cercavamo ci viene proposta

```
* .idata:100163CC ; struct hostent *__stdcall gethostbyname(const char *name)
.idata:100163CC      extrn gethostbyname:dword
.idata:100163CC      ; DATA XREF: sub_10001074:loc_100011AF↑r
.idata:100163CC      ; sub_10001074+1D3↑r ...
```

Cliccandoci due volte riusciamo anche a vedere la sua posizione all’interno del pannello testuale

Dovendo scoprire quante variabili sono presenti all'indirizzo di memoria 0x10001656, dobbiamo trovare l'indirizzo all'interno del codice nel pannello testuale.



Ecco l'indirizzo che cercavamo: scopriamo che le variabili (riconoscibili dall'offset negativo) sono 20

4.

Di parametri all'interno della funzione, che vengono identificate da un offset positivo, se ne trova 1 solo

```
.text:10001656 var_3B0      = dword ptr -3B0h -17
.text:10001656 var_1A4      = dword ptr -1A4h -18
.text:10001656 var_194      = dword ptr -194h -19
.text:10001656 WSADData     = WSADData ptr -190h -20
.text:10001656 arg_0        = dword ptr 4
.text:10001656
.text:1000165C sub     esp, 678h
                push    ebx
```

5.

In fondo al codice si trovano queste diciture:

```
xdoors_d:10093D74 aBackdoorServer db 0Dh,0Ah ; DATA XREF: sub_100042DB+B5↑o
xdoors_d:10093D74 db 0Dh,0Ah
xdoors_d:10093D74 db '*****',0Dh,0Ah
xdoors_d:10093D74 db '[BackDoor Server Update Setup]',0Dh,0Ah
xdoors_d:10093D74 db '*****',0Dh,0Ah
xdoors_d:10093D74 db 0Dh,0Ah,0
xdoors_d:10093DDB align 4

; char aHiMasterDDDDDD[]
aHiMasterDDDDDD db 'Hi,Master [%d/%d/%d %d:%d:%d]',0Dh,0Ah
                  ; DATA XREF: sub_1000FF58+145↑o
                  db 'WelCome Back...Are You Enjoying Today?',0Dh,0Ah
                  db 0Dh,0Ah
                  db 'Machine UpTime [%-.2d Days %-.2d Hours %-.2d Minutes %-.2d Secon'
                  db 'ds]',0Dh,0Ah
                  db 'Machine IdleTime [%-.2d Days %-.2d Hours %-.2d Minutes %-.2d Seco'
                  db 'nds]',0Dh,0Ah
                  db 0Dh,0Ah
                  db 'Encrypt Magic Number For This Remote Shell Session [0x%02x]',0Dh,0Ah
                  db 0Dh,0Ah,0
```

Ci fanno capire che il codice ha la funzione di creare una backdoor sulla vittima