

Obiettivo: Con riferimento al Malware_U3_W3_L2 sulla Virtual Machine dedicata all'analisi dei Malware, rispondere ai seguenti quesiti.

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?

Prima Parte

Per lo svolgimento di questa esercitazione useremo il tool **disassembler IDA pro free**, che ci tradurrà in **Assembly** il linguaggio macchina del nostro Malware in analisi.

Andiamo quindi a caricare il nostro Malware e analizziamo il codice tradotto, abbiamo subito ricavato la porzione di codice e l'indirizzo della funzione **DLLMain**, che risulta essere nella locazione **1000D02E** come possiamo vedere nell'immagini seguenti.



```

.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:1000D02F

; BOOL __stdcall DllMain(HINSTANCE hinstDLL,DWORD fdwReason,LPVOID lpvReserved)
_DllMain@12 proc near

hinstDLL= dword ptr  4
fdwReason= dword ptr  8
lpvReserved= dword ptr  0Ch

mov     eax, [esp+fdwReason]
dec     eax
jnz     loc_1000D107
  
```

Seconda Parte

Nella seconda parte di analisi consultiamo la **scheda degli imports**, abbiamo così individuato anche l'indirizzo di memoria della funzione richiesta, **gethostbyname**, che risulta essere **100163CC**.

```

.idata:100163C8 ; sub_10001074+1BF↑r ...
.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 ...
  
```

Terza Parte

Nella terza e ultima parte andiamo ad analizzare la locazione di memoria 0x10001656, dove abbiamo individuato **20 variabili locali importate** dalla funzione, le possiamo riconoscere dal simbolo negativo che l'interfaccia di IDA pro associa ad esse. Mentre troviamo **un solo paramento** che al contrario viene associato ad un offset positivo.

IDA View-A Hex View-A Exports Imports Names Functions Strings Structures Enums

```
.text:10001656 ; SUBROUTINE
.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
```

Parametro della funzione