S11/L2 Esercizio 05/12/2023

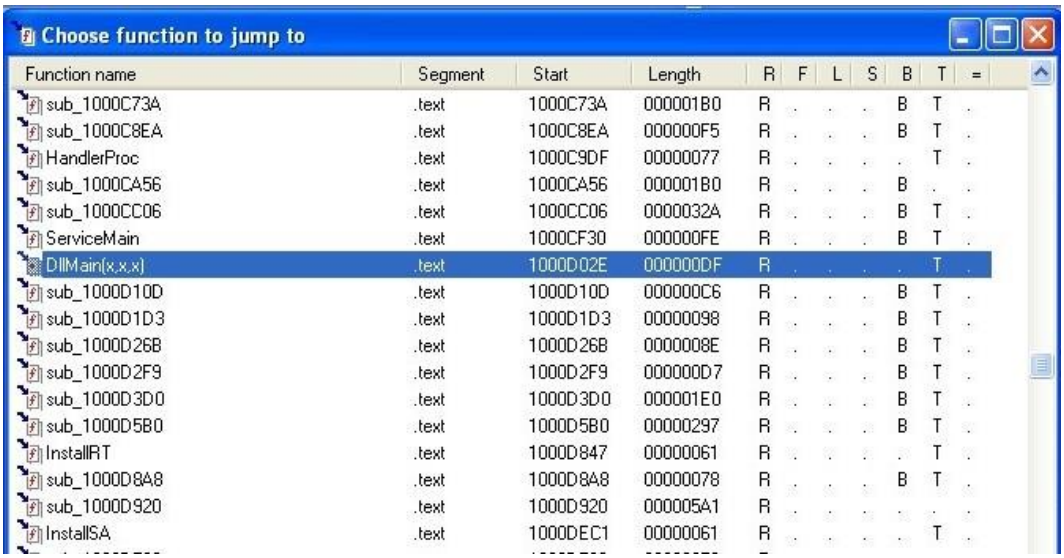
Nell'esercizio di oggi ci viene richiesto, con riferimento al malware fornito, di:

- Individuare l'indirizzo della funzione DLLMain
- Individuare la funzione «gethostbyname», identificarne l'indirizzo dell'import e spiegare cosa fa la funzione
- Individuare le variabili locali della funzione alla locazione di memoria 0x10001656
- Individuare i parametri della funzione sopra
- Inserire altre considerazioni macro livello sul malware (comportamento)

Quesito 1

Come possiamo rilevare dalle immagini di seguito **l'indirizzo della funzione DLLMain** è 1000D02E (evidenziata in blu)

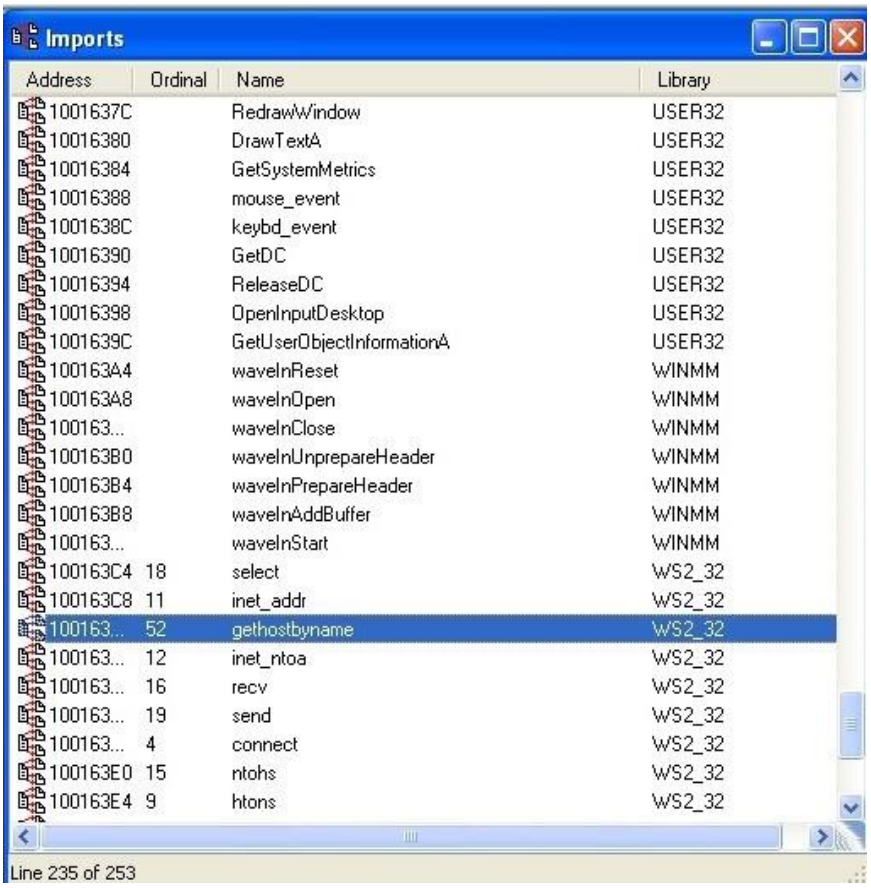
```
.text:1000D027                                          ; ServiceMain+F0↑j
.text:1000D027                  pop     edi
.text:1000D028                  pop     esi
.text:1000D029                  pop     ebx
.text:1000D02A                  leave
.text:1000D02B                  retn    8
.text:1000D02B ServiceMain      endp
.text:1000D02B
.text:1000D02E ; ||||||||||||||| S U B R O U T I N E |||||||||||||||||||||||||||||||||||||||
.text:1000D02E
.text:1000D02E
.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]
```

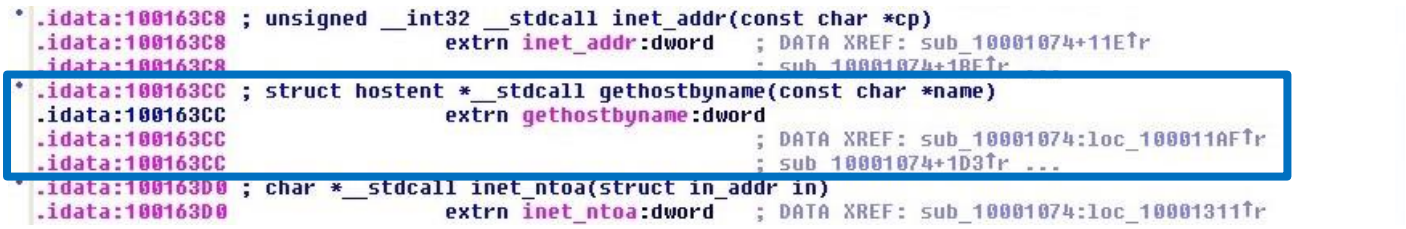


Quesito 2

Come possiamo vedere la funzione **“gethostbyname”** viene visualizzata nella scheda “imports” e, andando a visualizzarla nella schermata Ida View possiamo vedere che **l’indirizzo è** 100163CC, come evidenziato.

“Gethostbyname” presumibilmente tenta di connettersi ad un server remoto, prende in input il nome di un host e restituisce un puntatore a una struttura “hostent” contenente informazioni relative a quell'host. Queste informazioni possono includere gli indirizzi IP associati all'host e altri dettagli di rete (come probabilmente in questo caso).





Quesito 3-4

Nelle immagini possiamo vede che le **variabili** totali alla locazione di memoria 10001656 sono 20 (evidenziate in giallo) mentre viene rilevato un solo **parametro** (in rosso)

```
; DWORD __stdcall sub_10001656(LPVOID)
sub_10001656 proc near

var_675= byte ptr -675h
var_674= dword ptr -674h
hModule= dword ptr -670h
timeout= timeval ptr -66Ch
name= sockaddr ptr -664h
var_654= word ptr -654h
in= in_addr ptr -650h
Parameter= byte ptr -644h
CommandLine= byte ptr -63Fh
Data= byte ptr -638h
var_544= dword ptr -544h
var_50C= dword ptr -50Ch
var_500= dword ptr -500h
var_4FC= dword ptr -4FCh
readfds= fd_set ptr -4BCh
phkResult= HKEY__ ptr -3B8h
var_3B0= dword ptr -3B0h
var_1A4= dword ptr -1A4h
var_194= dword ptr -194h
WSAData= WSAData ptr -190h
arg_0= dword ptr  4
```

Possiamo rilevarne la differenza poiché mentre le variabili hanno un offset negativo (oltre al fatto che alcune riportano proprio il nome “var_”), il parametro (arg_0) è contrassegnato con un offset positivo

Quesito 5

Il malware, come possiamo vedere, si riferisce ad un backdoor server, lo possiamo constatare dalla funzione evidenziata

```
xdoors_d:10093D50                db '(1) Enter Current Directory ',27h,'%s',27h,0
xdoors_d:10093D73                align 4
xdoors_d:10093D74 ; char aBackdoorServer[]
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
xdoors_d:10093DDC ; char aWarn[]
```