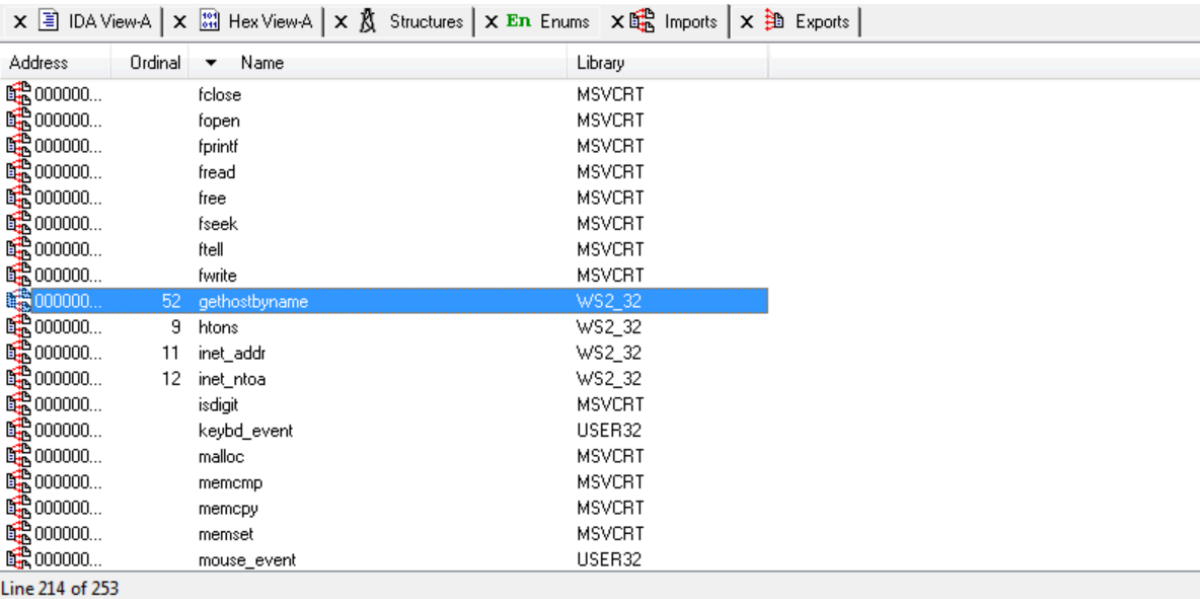


## SETTIMANA 11 ESERCIZIO 2

```
.text:1000D02E ; ----- SUBROUTINE -----  
.text:1000D02E  
.text:1000D02E  
.text:1000D02E ; BOOL __stdcall DllMain(HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved)  
.text:1000D02E _DllMain@12 proc near ; CODE XREF: DllEntryPoint+4B1p  
.text:1000D02E ; DATA XREF: sub_100110FF+2D10  
.text:1000D02E
```

- 1) Per trovare l'indirizzo della funzione ho fatto "search", digitando search directory e troviamo che l'indirizzo della funzione DLLMain è **1000D02E**.

- 2) Per trovare l'indirizzo dell'import gethostbyname andiamo ad aprire la finestra degli 'imports' da IDA Pro



Address	Ordinal	Name	Library
00000000		fclose	MSVCRT
00000000		fopen	MSVCRT
00000000		fprintf	MSVCRT
00000000		fread	MSVCRT
00000000		free	MSVCRT
00000000		fseek	MSVCRT
00000000		ftell	MSVCRT
00000000		fwrite	MSVCRT
00000000	52	gethostbyname	WS2_32
00000000	9	htons	WS2_32
00000000	11	inet_addr	WS2_32
00000000	12	inet_ntoa	WS2_32
00000000		isdigit	MSVCRT
00000000		keybd_event	USER32
00000000		malloc	MSVCRT
00000000		memcmp	MSVCRT
00000000		memcpy	MSVCRT
00000000		memset	MSVCRT
00000000		mouse_event	USER32

Line 214 of 253

Dopodiché cliccandoci sopra ci esce la schermata dove possiamo vedere il suo indirizzo ovvero 100163CC. Quindi va a prendere l'indirizzo IP degli host.

```

; sub_10001074, argsize __cdecl __stdcall inet_addr(const char *p)
; idata:100163C8 extrn inet_addr:dword ; CODE XREF: sub_10001074+11E↑p
; idata:100163C8 ; sub_10001074+1BF↑p ...
; idata:100163CC ; struct hostent *__stdcall gethostbyname(const char *name)
; idata:100163CC extrn gethostbyname:dword
; idata:100163CC ; CODE XREF: sub_10001074:loc_100011AF↑p
; idata:100163CC ; sub_10001074+1D3↑p ...
; idata:100163D0 ; char *__stdcall inet_ntoa(struct in_addr in)

```

3) Per trovare le variabili locali della funzione alla locazione di memoria 10001656 andiamo a cliccare sulla cella di memoria ed apriamo quella cella di memoria.

```

add     esp, 18h
push    edi                ; lpThreadId
push    edi                ; dwCreationFlags
push    edi                ; lpParameter
push    offset sub_10001656 ; lpStartAddress
push    edi                ; dwStackSize
push    edi                ; lpThreadAttributes

```

Le variabili presenti alla memoria sono 20.

```

; DWORD __stdcall sub_10001656(LPVOID)
sub_10001656 proc near

var_675= byte ptr -675h
var_674= dword ptr -674h
hLibModule= dword ptr -670h
timeout= timeval ptr -66Ch
name= sockaddr ptr -664h
var_654= word ptr -654h
Dst= dword ptr -650h
Parameter= byte ptr -644h
var_640= byte ptr -640h
CommandLine= byte ptr -63Fh
Source= byte ptr -63Dh
Data= byte ptr -638h
var_637= byte ptr -637h
var_544= dword ptr -544h
var_50C= dword ptr -50Ch
var_500= dword ptr -500h
Buf2= byte ptr -4FCh
readfds= fd_set ptr -4BCh
phkResult= byte 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

sub     esp, 678h

```

4) Il parametro presente nella funzione è uno solo ovvero "arg 0" .

IDA View-AHex View-AStructuresEn EnumsImportsExports

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

var_675= byte ptr -675h
var_674= dword ptr -674h
hLibModule= dword ptr -670h
timeout= timeval ptr -66Ch
name= sockaddr ptr -664h
var_654= word ptr -654h
Dst= dword ptr -650h
Parameter= byte ptr -644h
var_640= byte ptr -640h
CommandLine= byte ptr -63Fh
Source= byte ptr -63Dh
Data= byte ptr -638h
var_637= byte ptr -637h
var_544= dword ptr -544h
var_50C= dword ptr -50Ch
var_500= dword ptr -500h
Buf2= byte ptr -4FCh
readfds= fd_set ptr -4BCh
phkResult= byte ptr -3B8h
var_380= dword ptr -380h
var_1A4= dword ptr -1A4h
var_194= dword ptr -194h
WSAData= WSAData ptr -190h
arg_0= dword ptr 4

sub     esp, 678h
```

100.00% (803,36) (935,468) 00000A56 0000000010001656: sub\_10001656