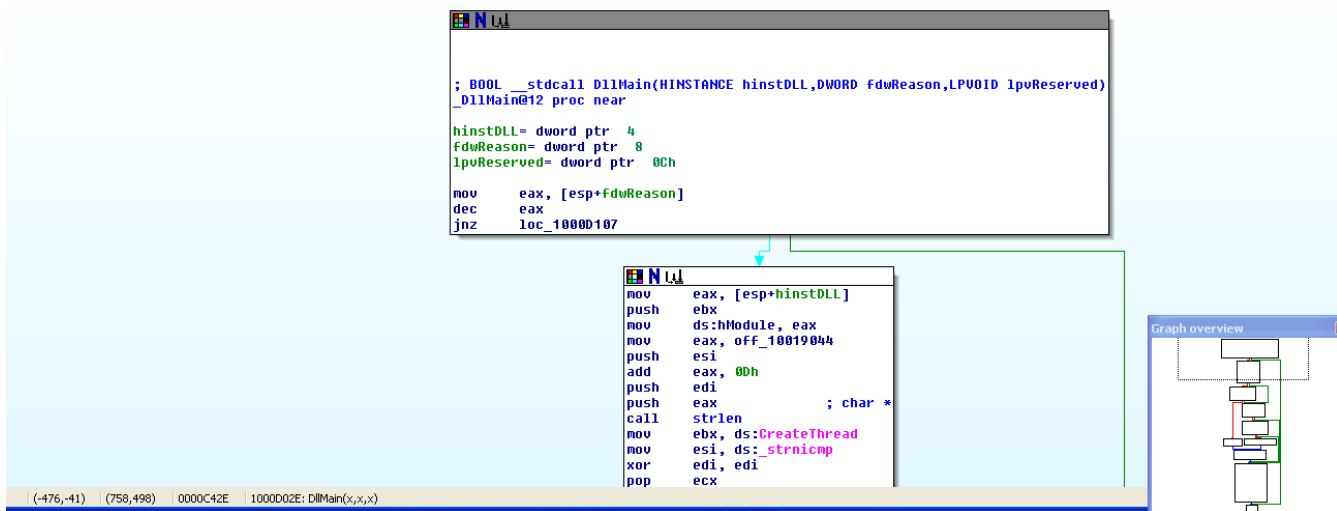


Report Ida Pro

La task odierna prevede l'estrapolazione delle seguenti informazioni :

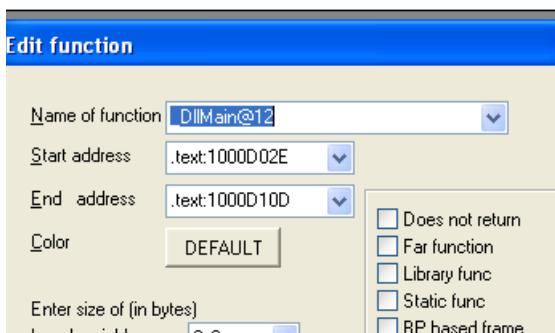
1. Individuazione indirizzo della funzione DllMain.
2. Individuazione funzione GetHostByName e indirizzo import .
3. Individuazione Variabili di funzione a locazione di memoria 0x10001656
4. Individuazione parametri di funzione a locazione memoria 0x10001656

Queste info verranno ottenute utilizzando Ida Pro (Interactive Disassembler) un disassembler usato per il reverse engineering



Per iniziare si dovrà aprire Ida pro (dal desktop della nostra macchina virtuale) e caricare il file desiderato.

Una volta fatto si aprirà la schermata del diagramma di flusso del codice assembly del file, a questo punto per individuare l'indirizzo della funzione DllMain ossia un entry point, che troveremo alla voce "edit function"

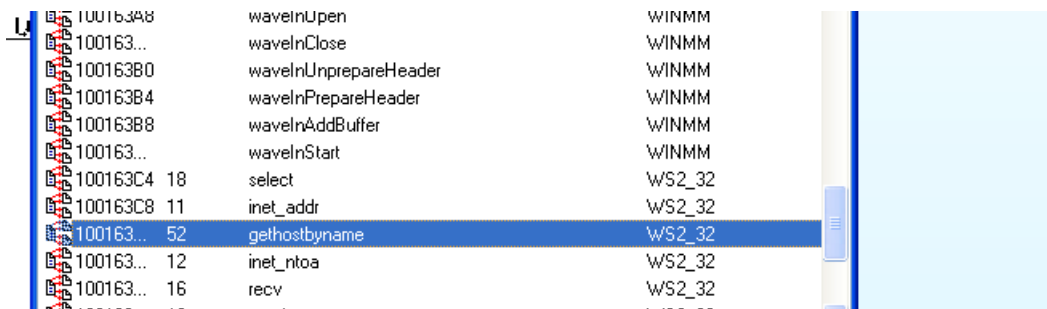


2. Individuiamo adesso la funzione GetHostByName

Basterà andare nella finestra Imports e cercare la funzione in questione.

```
.idata:100163C8 ; unsigned __int64 __cdecl inet_addr(const char *cp)
.idata:100163C8 extrn inet_addr:dword ; DATA XREF: sub_10001074+11E1r ...
.idata:100163C8 ; sub_10001074+1BF1r ...
* .idata:100163CC ; struct hostent * __stdcall gethostbyname(const char *name)
.idata:100163CC extrn gethostbyname:dword
.idata:100163CC ; DATA XREF: sub_10001074:loc_100011AF1r ...
.idata:100163CC ; sub_10001074+1D31r ...
* .idata:100163D0 ; char * __stdcall inet_ntoa(struct in_addr in)
```

L'indirizzo della funzione è quindi 0x100163CC importata nella sezione "idata"



00100163A8	waveInOpen	WINMM
00100163B0	waveInClose	WINMM
00100163B0	waveInUnprepareHeader	WINMM
00100163B4	waveInPrepareHeader	WINMM
00100163B8	waveInAddBuffer	WINMM
00100163C0	waveInStart	WINMM
00100163C4 18	select	WS2_32
00100163C8 11	inet_addr	WS2_32
00100163CC 52	gethostbyname	WS2_32
00100163D0 12	inet_ntoa	WS2_32
00100163D4 16	recv	WS2_32

3. Variabili locali di funzione a locazione di memoria 0x10001656

```
.text:10001656 WSAData = WSAData ptr -190h
.text:10001656 arg_0 = dword ptr 4
.text:10001656
* .text:10001656 sub esp, 678h
* .text:1000165C push ebx
* .text:1000165D push ebp
* .text:1000165E push esi
* .text:1000165F push edi
* .text:10001660 call sub_10001000
* .text:10001665 test eax, eax
* .text:10001667 jnz short loc_100016BC
* .text:10001669 xor ebx, ebx
* .text:1000166B mov [esp+688h+var_674], ebx
* .text:1000166F mov [esp+688h+hModule], ebx
* .text:10001673 call sub_10003695
* .text:10001678 mov dword_1000E5C4, eax
* .text:1000167D call sub_100036C3
* .text:10001682 push 3A98h ; dwMilliseconds
* .text:10001687 mov dword_1000E5C8, eax
* .text:1000168C call ds:Sleep
* .text:10001692 call sub_100110FF
* .text:10001697 lea eax, [esp+688h+WSAData]
* .text:1000169E push eax ; lpWSAData
* .text:1000169F push 202h ; wVersionRequested
* .text:100016A4 call ds:WSAStartup
* .text:100016AA cmp eax, ebx
* .text:100016AC jz short loc_100016CB
* .text:100016AE push eax
* .text:100016AF push offset aWsastartupErro ; "WSAStartup() error: %d\n"
* .text:100016B4 call ds:printf
* .text:100000F4 push edi ; dwCreationFlags
* .text:100000F5 push edi ; lpParameter
* .text:100000F6 push offset sub_10001656 ; lpStartAddress
* .text:100000FB push edi ; dwStackSize
* .text:100000FC push edi ; lpThreadAttributes
* .text:100000FD call ebx, CreateThread
```

Troviamo le variabili ad un offset negativo rispetto al registro EBP.

I parametri si trovano invece ad un offset positivo ad EBP.

Per un totale di 23 variabili ed un parametro.

```

.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 -48Ch
.text:10001656 phkResult            = HKEY__ ptr -388h
.text:10001656 var_380              = dword ptr -380h
.text:10001656 var_1A4              = dword ptr -1A4h
.text:10001656 var_194              = dword ptr -194h
.text:10001656 WSAData              = WSADData ptr -190h
.text:10001656 arg_0                = dword ptr 4
* .text:10001656                  sub     esp, 678h

```

La funzione a locazione di memoria a 10001656 ossia funzione RegOpenKey.