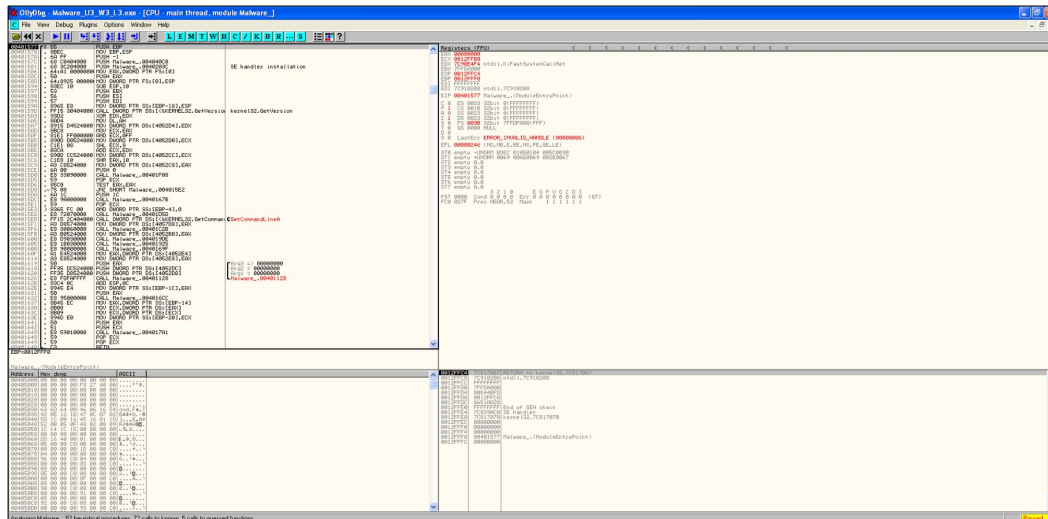


OllyDBG

Obiettivo: Utilizzare il tool OllyDBG sul Malware proposto dalla traccia

1) CreateProcess e CommandLine

La traccia ci chiede di iniziare utilizzando il tool **OllyDBG** per analizzare l'eseguibile *Malware_U3_W3_L3.exe*.



Come prima richiesta dobbiamo andare all'indirizzo **0040106E** dove si trova la funzione "CreateProcess" e da qui cercare il valore del parametro "CommandLine" che viene passato sullo Stack.

0040102C	E8 AF030000	CALL Malware_.004013E0	
00401031	83C4 0C	ADD ESP,0C	
00401034	C745 D4 010100	MOV DWORD PTR SS:[EBP-2C],101	
0040103B	66:C745 D8 0000	MOV WORD PTR SS:[EBP-28],0	
00401041	8B55 18	MOV EDX,DWORD PTR SS:[EBP+18]	
00401044	8955 E0	MOV DWORD PTR SS:[EBP-20],EDX	
00401047	8945 E0	MOV EAX,DWORD PTR SS:[EBP-20]	
0040104A	8945 E8	MOV DWORD PTR SS:[EBP-18],EAX	
0040104D	8B4D E8	MOV ECX,DWORD PTR SS:[EBP-18]	
00401050	894D E4	MOV DWORD PTR SS:[EBP-1C],ECX	
00401053	8D55 F0	LEA EDI,DWORD PTR SS:[EBP-10]	
00401056	52	PUSH EDI	
00401057	8D45 A8	LEA EAX,DWORD PTR SS:[EBP-58]	
0040105A	50	PUSH EAX	
0040105B	6A 00	PUSH 0	
0040105D	6A 00	PUSH 0	
0040105F	6A 00	PUSH 0	
00401061	6A 01	PUSH 1	
00401063	6A 00	PUSH 0	
00401065	6A 00	PUSH 0	
00401067	55 30050400	PUSH Malware_.00405030	
0040106C	6A 00	PUSH 0	
0040106E	FF15 04404000	CALL DWORD PTR DS:[<&KERNEL32.CreateProcessA>]	
00401074	8945 EC	MOV DWORD PTR SS:[EBP-14],EAX	
00401077	6A FF	PUSH -1	
00401079	8B4D F0	MOV ECX,DWORD PTR SS:[EBP-10]	
0040107C	51	PUSH ECX	
0040107D	FF15 00404000	CALL DWORD PTR DS:[<&KERNEL32.WaitForSingleObject>]	
00401083	33C0	XOR EAX,EAX	

pProcessInfo

pStartupInfo

CurrentDir = NULL

pEnvironment = NULL

CreationFlags = 0

InheritHandles = TRUE

pThreadSecurity = NULL

pProcessSecurity = NULL

CommandLine = "cmd"

ModuleFileName = NULL

CreateProcessA

Timeout = INFINITE

hObject

WaitForSingleObject

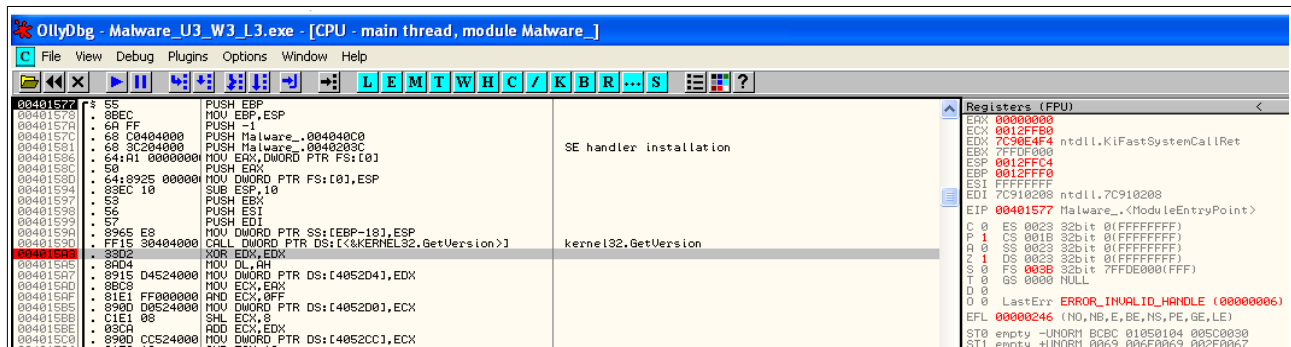
Possiamo notare che sulla linea dove nei *commenti* è indicato il parametro *CommandLine* troviamo l'istruzione **PUSH Malware_.00405030** suggerendoci che cercando l'indirizzo di quest'ultimo potrei trovare il valore di *cmd*.

Avviando **IDA** ho cercato tramite *Interfaccia Testuale* l'indirizzo di cui sopra e sono riuscito a trovare che a **cmd** viene assegnato il valore 0.

:0040502E	db	0	
:0040502F	db	0	
:00405030	; char CommandLine[]		
:00405030	CommandLine	db 'cmd',0	; DATA XREF: sub_401000+67↑to
:00405034	unk_405034	db 46h ; F	; DATA XREF: _main+A3↑to
:00405035	db	6	
:00405036	db	16h	
:00405037	db	...	

2) EDX

Successivamente la traccia chiede di inserire un *breakpoint* all'indirizzo **004015A3** e di osservare il valore di **EDX** prima e dopo l'esecuzione dello “*step-into*”.



OllyDbg - Malware_U3_W3_L3.exe - [CPU - main thread, module Malware_]

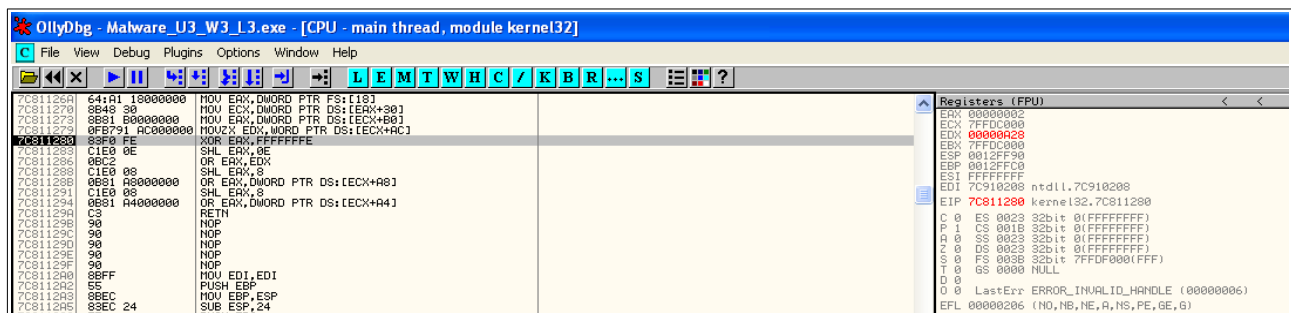
Assembly window (Address, Disassembly, Comment):

- 00401577: 55 PUSH EBP
- 00401578: 8BEC MOV EBP, ESP
- 00401579: 6A FF PUSH -1
- 0040157A: 68 00404000 PUSH Malware_.00404000
- 0040157B: 68 3C204000 PUSH Malware_.0040203C
- 0040157C: 64:01 00000000 MOV EDX, DWORD PTR FS:[0]
- 0040157D: 50 PUSH EAX
- 0040157E: 64:8925 000000 MOV DWORD PTR FS:[0], ESP
- 0040157F: 8BEC 10 SUB ESP, 10
- 00401580: 56 PUSH EAX
- 00401581: 57 PUSH EDI
- 00401582: 9965 E8 MOV DWORD PTR SS:[EBP-18], ESP
- 00401583: FF15 30404000 CALL DWORD PTR DS:[40403030] kernel32.GetVersion
- 00401584: 3B02 XOR EBX, EBX
- 00401585: 8A04 MOV DL, AH
- 00401586: 9915 D4524000 MOV DWORD PTR DS:[4052D41], EDX
- 00401587: 8BC8 MOV ECX, EAX
- 00401588: 81E1 FF000000 AND ECX, 0FF
- 00401589: 9900 D0524000 MOV DWORD PTR DS:[4052D01], ECX
- 0040158A: C1E1 08 SHL ECX, 8
- 0040158B: 000A ADD ECX, EDX
- 0040158C: 9900 CC524000 MOV DWORD PTR DS:[4052CC1], ECX

Registers (FPU):

- EAX: 00000000
- ECX: 0012FFB0
- EDX: 7C90E4F4 ntdll.KiFastSystemCallRet
- EBX: 77FDC000
- ESP: 0012FFC4
- EBP: 0012FFD0
- ESI: FFFFFFFF
- EDI: 7C910208 ntdll.7C910208
- EIP: 00401577 Malware_.<ModuleEntryPoint>
- C 0 ES 0023 32bit 0 (FFFFFFFF)
- P 1 CS 001B 32bit 0 (FFFFFFFF)
- A 0 SS 0023 32bit 0 (FFFFFFFF)
- Z 1 DS 0023 32bit 0 (FFFFFFFF)
- S 0 FS 003B 32bit 7FFDE000 (FFF)
- T 0 GS 0000 NULL
- D 0
- O 0 LastErr: ERROR_INVALID_HANDLE (00000006)
- EFL 00000246 (NO, NB, E, BE, NS, PE, GE, LE)
- ST0 empty -UNORM BCBC 01050104 005C0030
- ST1 empty +INORM BAA9 006FA9A5 002FA9A7

Prima dell'analisi tramite “*step-into*” notiamo che il valore di **EDX** è **7C90E4F4**. Posizionato il *breakpoint* dove richiesto (indicato con una **Tab Rossa** sulla figura) andiamo ad avviare “passo-passo” il **Malware**.



OllyDbg - Malware_U3_W3_L3.exe - [CPU - main thread, module kernel32]

Assembly window (Address, Disassembly, Comment):

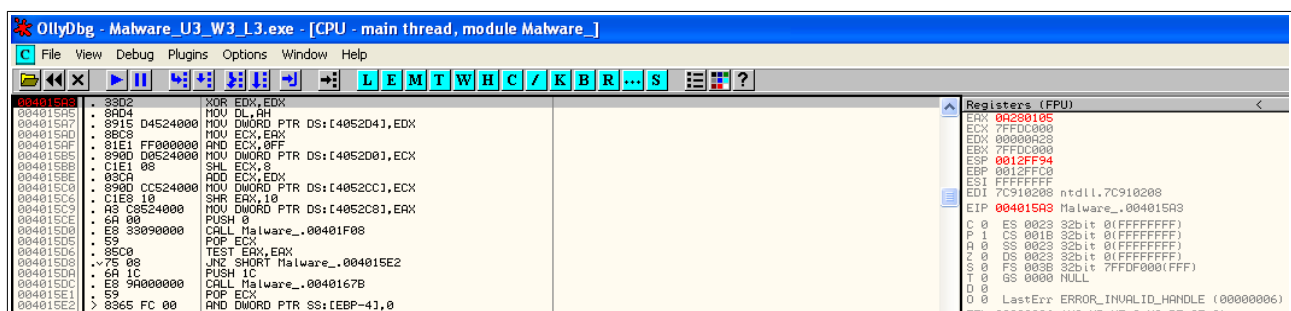
- 7C81126A: 64:01 10000000 MOV EAX, DWORD PTR FS:[10]
- 7C81126B: 8B48 30 MOV ECX, DWORD PTR DS:[EAX+30]
- 7C81126C: 8B51 00000000 MOV EAX, DWORD PTR DS:[ECX+00]
- 7C81126D: 0FB791 AC000000 MOVZX EDX, WORD PTR DS:[ECX+AC]
- 7C81126E: 8BFA FE MOV EAX, FFFFFFFF
- 7C81126F: C1E0 0E SHL EAX, 6
- 7C811270: 06C2 OR EAX, EDX
- 7C811271: C1E0 08 SHL EAX, 8
- 7C811272: 0681 A0000000 OR EAX, DWORD PTR DS:[ECX+A0]
- 7C811273: C1E0 08 SHL EAX, 8
- 7C811274: 0681 A4000000 OR EAX, DWORD PTR DS:[ECX+A4]
- 7C811275: C3 RETN
- 7C811276: 90 NOP
- 7C811277: 90 NOP
- 7C811278: 90 NOP
- 7C811279: 90 NOP
- 7C81127A: 90 NOP
- 7C81127B: 90 NOP
- 7C81127C: 8BFF MOV EDI, EDI
- 7C81127D: 55 PUSH EAX
- 7C81127E: 8BEC MOV EBP, ESP
- 7C81127F: 83EC 24 SUB ESP, 24

Registers (FPU):

- EAX: 00000002
- ECX: 77FDC000
- EDX: 00000A28
- EBX: 77FDC000
- ESP: 0012FF90
- EBP: 0012FFC0
- ESI: FFFFFFFF
- EDI: 7C910208 ntdll.7C910208
- EIP: 7C811280 kernel32.7C811280
- C 0 ES 0023 32bit 0 (FFFFFFFF)
- P 1 CS 001B 32bit 0 (FFFFFFFF)
- A 0 SS 0023 32bit 0 (FFFFFFFF)
- Z 0 DS 0023 32bit 0 (FFFFFFFF)
- S 0 FS 003B 32bit 7FFDF000 (FFF)
- T 0 GS 0000 NULL
- D 0
- O 0 LastErr: ERROR_INVALID_HANDLE (00000006)
- EFL 00000206 (NO, NB, NE, A, NS, PE, GE, G)

Durante il processo, prima del *breakpoint*, viene richiamata una funzione “**kernel32.GetVersion**” dove dalla figura in alto possiamo vedere che a **EDX** viene assegnato un valore diverso dal precedente con l'istruzione **movzx**, cioè **00000A28**.

Questo tipo di istruzione è utilizzata per estendere il valore di un registro o di una posizione di memoria con zeri a sinistra, mantenendo invariati i bit meno significativi.



OllyDbg - Malware_U3_W3_L3.exe - [CPU - main thread, module Malware_]

Assembly window (Address, Disassembly, Comment):

- 0040158B: 3B02 XOR EBX, EBX
- 0040158C: 8A04 MOV DL, AH
- 0040158D: 9915 D4524000 MOV DWORD PTR DS:[4052D41], EDX
- 0040158E: 8BC8 MOV ECX, EAX
- 0040158F: 81E1 FF000000 AND ECX, 0FF
- 00401590: 9900 D0524000 MOV DWORD PTR DS:[4052D01], ECX
- 00401591: C1E1 08 SHL ECX, 8
- 00401592: 000A ADD ECX, EDX
- 00401593: 9900 CC524000 MOV DWORD PTR DS:[4052CC1], ECX
- 00401594: C1E8 10 SHR EAX, 10
- 00401595: A3 C0524000 MOV DWORD PTR DS:[4052C01], EAX
- 00401596: 6A 00 PUSH 0
- 00401597: 99 33090000 CALL Malware_.00401F08
- 00401598: 59 POP EAX
- 00401599: 85C0 TEST EAX, EAX
- 0040159A: 75 08 JNZ SHORT Malware_.004015E2
- 0040159B: 6A 1C PUSH 1C
- 0040159C: E8 9A000000 CALL Malware_.0040167B
- 0040159D: 59 POP EAX
- 0040159E: 59 POP EAX
- 0040159F: 8365 FC 00 AND DWORD PTR SS:[EBP-4], 0

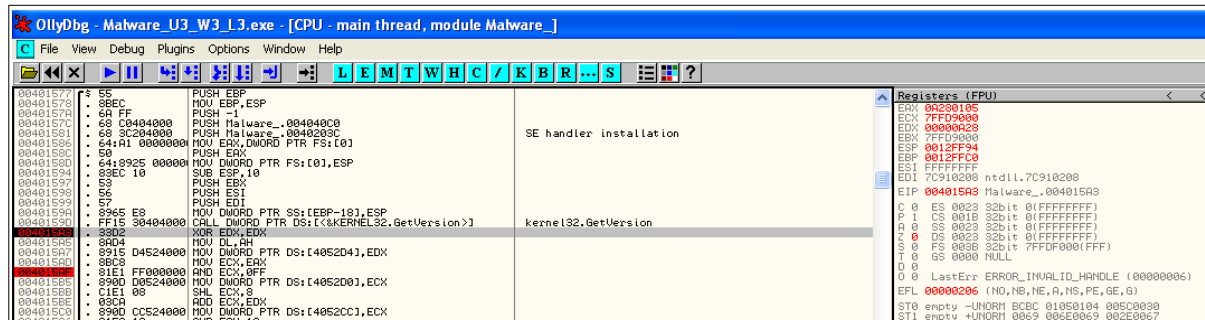
Registers (FPU):

- EAX: 00200105
- ECX: 77FDC000
- EDX: 00000A28
- EBX: 77FDC000
- ESP: 0012FF94
- EBP: 0012FFC0
- ESI: FFFFFFFF
- EDI: 7C910208 ntdll.7C910208
- EIP: 004015A3 Malware_.004015A3
- C 0 ES 0023 32bit 0 (FFFFFFFF)
- P 1 CS 001B 32bit 0 (FFFFFFFF)
- A 0 SS 0023 32bit 0 (FFFFFFFF)
- Z 0 DS 0023 32bit 0 (FFFFFFFF)
- S 0 FS 003B 32bit 7FFDF000 (FFF)
- T 0 GS 0000 NULL
- D 0
- O 0 LastErr: ERROR_INVALID_HANDLE (00000006)
- EFL 00000206 (NO, NB, NE, A, NS, PE, GE, G)

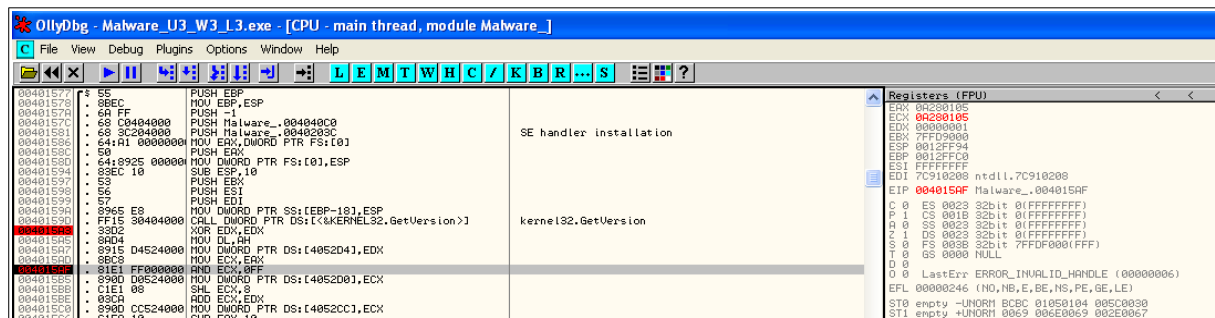
Terminata la funzione l'esecuzione riprende da quello che è il “codice principale” con l'ultimo valore registrato.

3) ECX

L'esercizio chiede inoltre di monitorare il valore del registro **ECX** con uno "step-into" fino a un secondo *breakpoint*. Riprendendo quindi dal precedente "punto di sosta" notiamo che il valore di **ECX** è **7FFF9000**.



In questo caso non ci sono particolari chiamate ad altre funzioni perciò l'esecuzione dello "step-into" procede senza interruzioni fino al secondo *breakpoint* fissato.

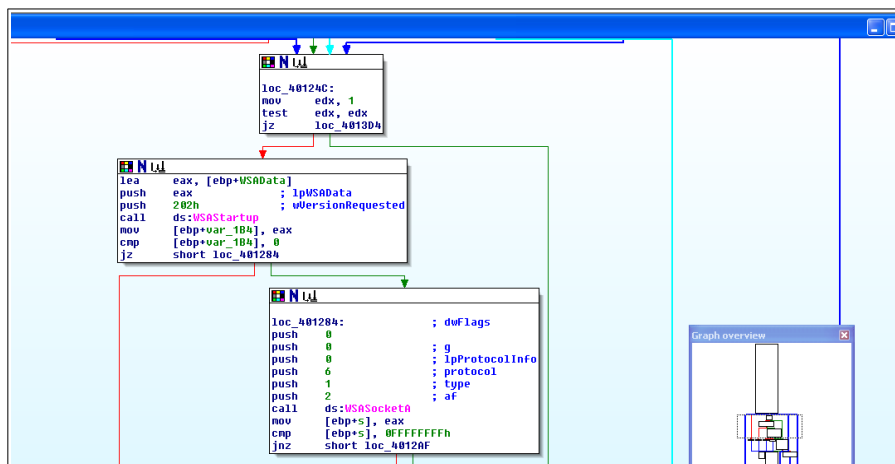


Il *breakpoint* è fissato proprio nella posizione dell'operatore **AND**. Qui il valore di **ECX** subisce una variazione secondo l'operazione logica che effettua **AND**, ovvero restituisce l'**AND** logico tra i bit di **ECX** e la forma binaria di 0FF aggiornando **ECX** con il risultato dell'operazione.

Per concludere **ECX** avrà un valore di **0A280105**.

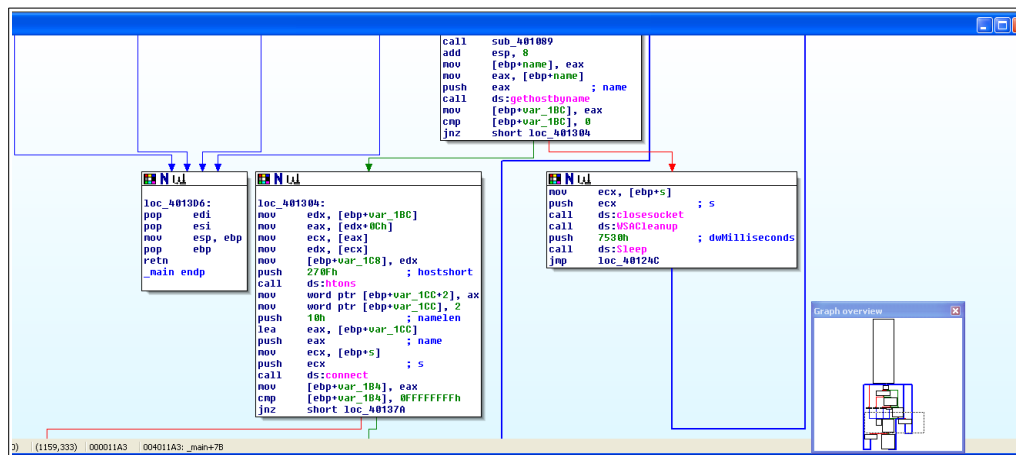
4) Funzionamento del Malware

Come ultima richiesta la traccia chiede di spiegare il comportamento del **Malware**. Per facilitare il compito ho utilizzato **IDA** per leggere l'**Interfaccia Grafica** ed avere il linea di massima un'idea sul comportamento del **Malware** in questione.



Da un primo sguardo possiamo notare la presenza di **WSAStartup**, una funzione che avvia l'uso della **DLL Winsock**; infatti poco sotto è presente **WSASocketA**, che è una funzione in grado di creare un **socket** associato a un provider di servizi di trasporto specifici.

Suppongo quindi che il **Malware** vada a creare prima di tutto un **socket** da collegare a un host specifico (troviamo infatti anche **gethostbyname**).



Dalla figura in alto potresti pensare che se la connessione del **socket** non avviene esso chiude prima di tutto il **socket** per poi utilizzare la funzione **WSACleanup** per chiudere la **DLL Winsock** ed infine ripartire con il tentativo di connessione (linea blu che riporta all'inizio del codice).

In caso di successo viene richiamata la funzione **htons**, che converte un valore numerico in byte dall'ordine dei byte dell'host all'ordine dei byte di rete (converte ad esempio il valore in byte dell'host di una porta in valore dei byte di rete). Suppongo venga eseguita questa procedura per mantenere una connessione continua su una data porta perché ho notato che ogni conclusione del Malware lo riporta all'inizio del ciclo.

