

S11/L3

Analisi dinamica avanzata

Traccia

Fate riferimento al malware: Malware_U3_W3_L3, presente all'interno della cartella Esercizio_Pratico_U3_W3_L3 sul desktop della macchina virtuale dedicata all'analisi dei malware.

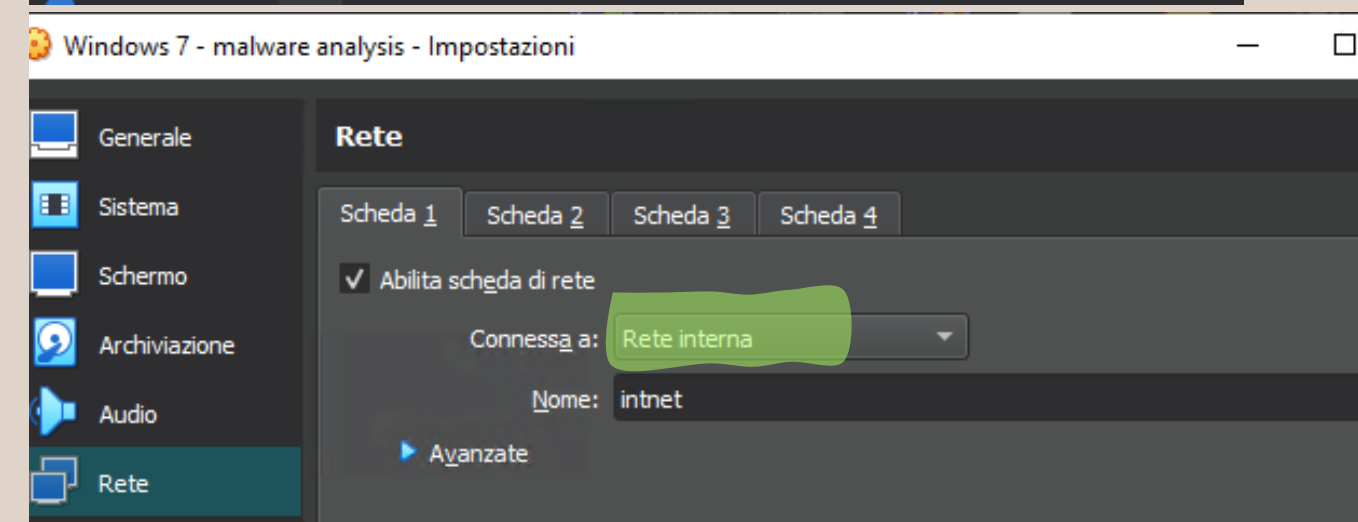
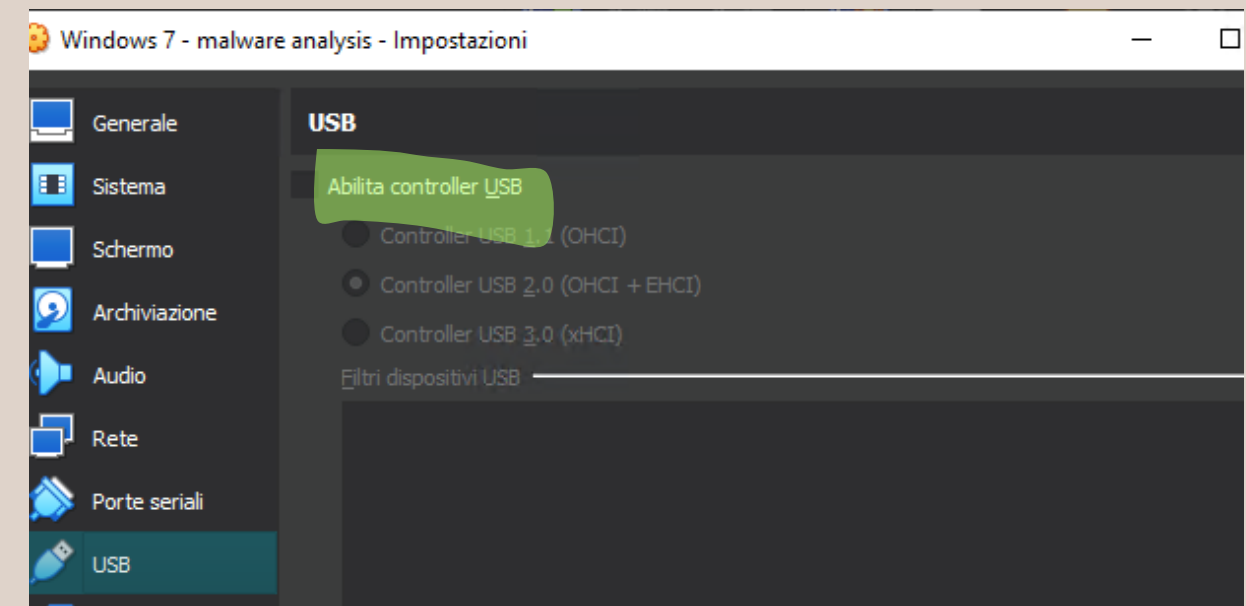
Rispondete ai seguenti quesiti utilizzando OllyDBG.

- All'indirizzo 0040106E il Malware effettua una chiamata di funzione alla funzione «CreateProcess». Qual è il valore del parametro «CommandLine» che viene passato sullo stack? (1)
- Inserite un breakpoint software all'indirizzo 004015A3. Qual è il valore del registro EDX? (2)
Eseguite a questo punto uno «step-into». Indicate qual è ora il valore del registro EDX (3) motivando la risposta (4). Che istruzione è stata eseguita? (5)
- Inserite un secondo breakpoint all'indirizzo di memoria 004015AF. Qual è il valore del registro ECX? (6) Eseguite un step-into. Qual è ora il valore di ECX? (7) Spiegate quale istruzione è stata eseguita (8). • BONUS: spiegare a grandi linee il funzionamento del malware

OllyDBG e isolamento VM

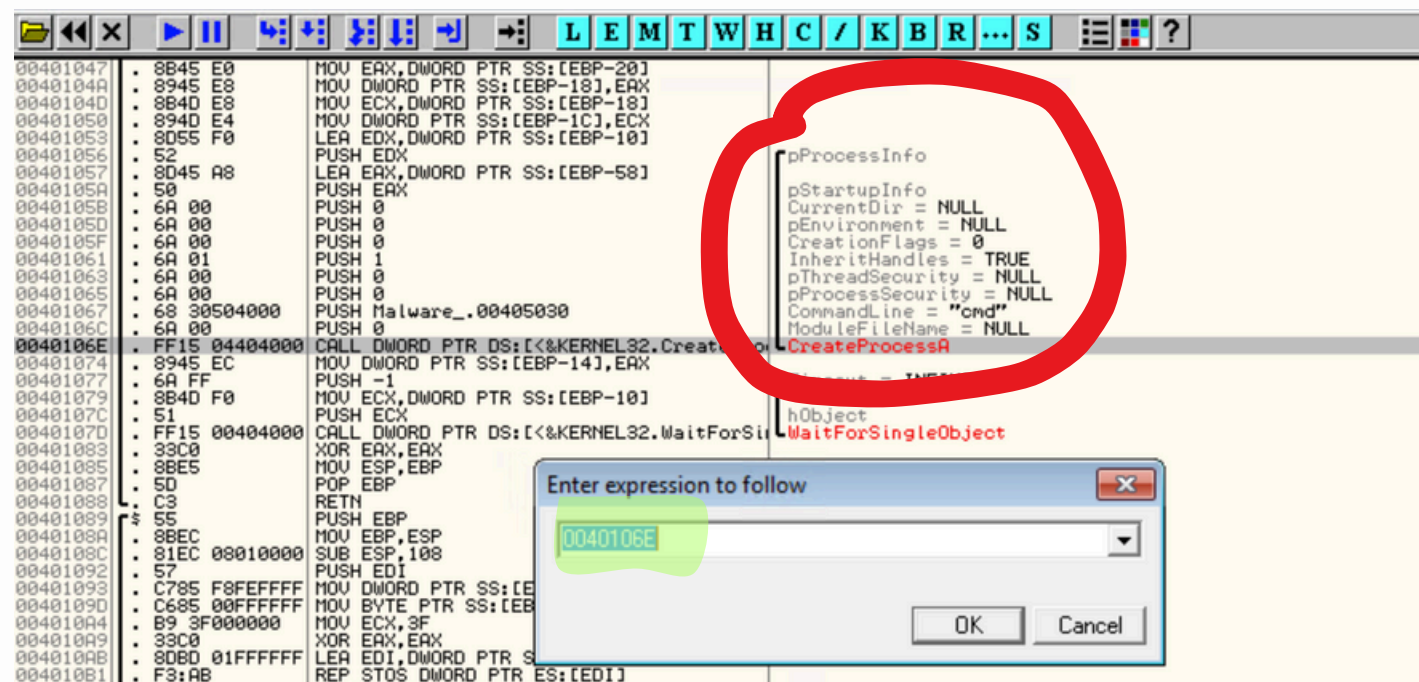
Come visto nella lezione teorica, OllyDbg è un **debugger** per Windows che permette di **analizzare** il codice eseguibile di un programma **durante** l'esecuzione.

Isolo la VM in **rete interna** per proteggerla, poiché devo eseguire e analizzare un malware con OllyDbg. Disattivo **USB** e altre funzionalità per garantire la sicurezza.



1 - All'indirizzo 0040106E il Malware effettua una chiamata di funzione alla funzione «CreateProcess». Qual è il valore del parametro «CommandLine» che viene passato sullo stack?

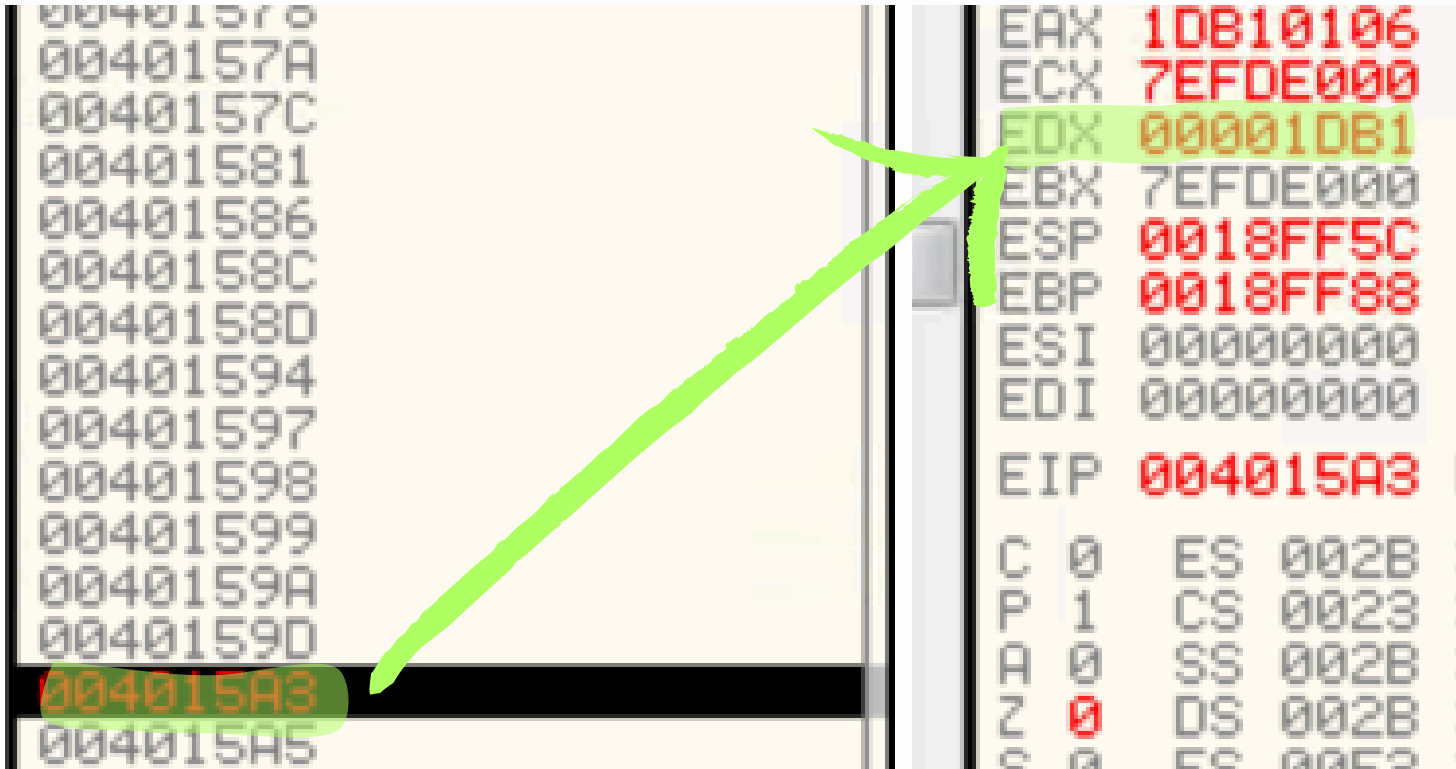
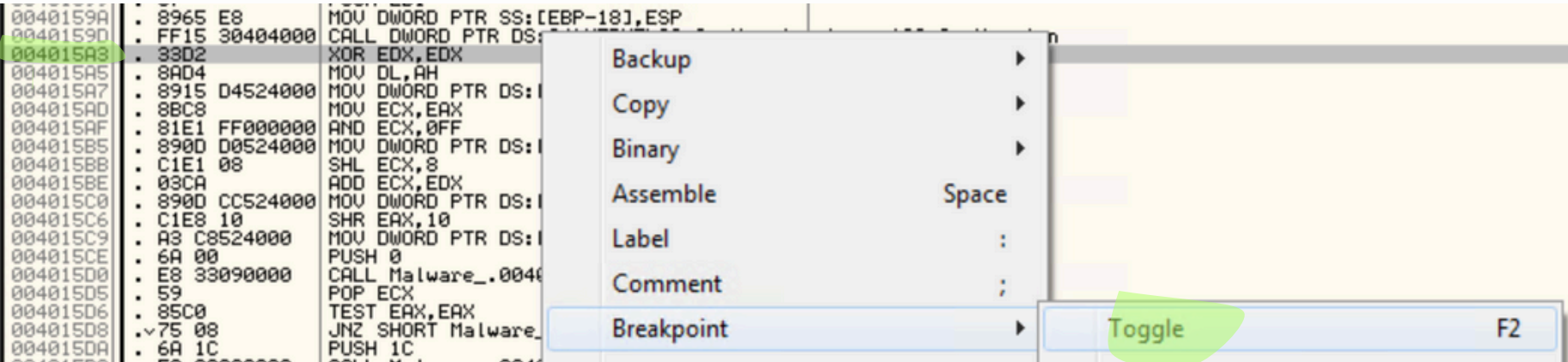
Una volta raggiunto l'indirizzo **0040106E** utilizzando la scorciatoia CTRL+G, noto che il malware passa una stringa che punta a **cmd.exe**. Questo potrebbe indicare un tentativo di eseguire cmd per operazioni malevole, come l'apertura di una **reverse shell**.



```
pProcessInfo
pStartupInfo
CurrentDir = NULL
pEnvironment = NULL
CreationFlags = 0
InheritHandles = TRUE
pThreadSecurity = NULL
pProcessSecurity = NULL
CommandLine = "cmd"
ModuleFileName = NULL
CreateProcessA
```

2 - Inserite un breakpoint software all'indirizzo 004015A3. Qual è il valore del registro EDX?

Come da richiesta, vado all'indirizzo 004015A3 e imposto un breakpoint. Dopodichè riprendo l'esecuzione del malware fino a che non arriva al breakpoint. Il valore del registro EDX è **00001DB1**.



3, 4, 5 - Eseguite a questo punto uno «step-into». Indicate qual è ora il valore del registro EDX motivando la risposta. Che istruzione è stata eseguita?

A questo punto, eseguo uno **step-into** come da richiesta. Il valore di EDX diventa **00000000**, poiché l'istruzione eseguita è **XOR EDX, EDX**. Questa istruzione effettua un'operazione XOR tra il registro EDX e sé stesso, **azzerando** così il valore di EDX.

```
004015A3 | . 33D2 | XOR EDX, EDX
```

Registers (FPU)					
EAX	1DB10106				
ECX	7EFDE000				
EDX	00000000				
EBX	7EFDE000				
ESP	0018FF5C				
EBP	0018FF88				
ESI	00000000				
EDI	00000000				
EIP	004015A5	Malware_.	004015A5		
C	0	ES	002B	32bit	0(FFFFFFFF)
P	1	CS	0023	32bit	0(FFFFFFFF)
A	0	SS	002B	32bit	0(FFFFFFFF)
Z	1	DS	002B	32bit	0(FFFFFFFF)
S	0	FS	0053	32bit	7EFDD000(FFF)
T	0	GS	002B	32bit	0(FFFFFFFF)

6, 7, 8 - Inserite un secondo breakpoint all'indirizzo di memoria 004015AF. Qual è il valore del registro ECX? Eseguite un step-into. Qual è ora il valore di ECX? Spiegate quale istruzione è stata eseguita.

Imposto un altro breakpoint all'indirizzo **004015AF**. Eseguendo il malware fino a quel punto, il valore di ECX è **1DB10106**. Successivamente, eseguo uno step-into, e il valore di ECX diventa **00000006**, poiché l'istruzione all'indirizzo 004015AF è un AND ECX, 0FF, ovvero un'operazione AND bit a bit tra i due valori. Questo spiega il risultato ottenuto.

Registers (FPU)	
EAX	1DB10106
ECX	1DB10106
EDX	00000001
EBX	7EFDE000
ESP	0018FF5C
EBP	0018FF88
ESI	00000000
EDI	00000000
EIP	004015AF Malware_...
C 0	ES 002B 32bit 0(F
P 1	CS 0023 32bit 0(F
A 0	SS 002B 32bit 0(F
Z 1	DS 002B 32bit 0(F
S 0	FS 0053 32bit 7EF
T 0	GS 002B 32bit 0(F
D 0	
O 0	LastErr ERROR_SUC

Registers (FPU)	
EAX	1DB10106
ECX	00000006
EDX	00000001
EBX	7EFDE000
ESP	0018FF5C
EBP	0018FF88
ESI	00000000
EDI	00000000
EIP	004015B5 Malware_...
C 0	ES 002B 32bit 0(F
P 1	CS 0023 32bit 0(F
A 0	SS 002B 32bit 0(F
Z 0	DS 002B 32bit 0(F
S 0	FS 0053 32bit 7EF
T 0	GS 002B 32bit 0(F
D 0	
O 0	LastErr ERROR_SUC

Operazione BIT A BIT

```
Windows PowerShell
PS C:\Users\user> "<0:X8>" -f <0x1DB10106 -band 0xFF>
00000006
PS C:\Users\user>
```

BONUS: spiegare a grandi linee il funzionamento del malware

Dopo aver analizzato il codice impostando dei **breakpoint** e scorrendo il flusso delle istruzioni, ho individuato delle chiamate di rete chiave nelle immagini allegate. Queste mostrano l'inizializzazione di una connessione TCP/IP tramite WSAStartup e WSASocketA, seguita dalla risoluzione dell'indirizzo del server con **gethostbyname** e la connessione con connect. Collegando queste informazioni con la precedente chiamata a cmd.exe, posso immaginare (almeno per ora) che il malware prepara il terreno per eseguire comandi da remoto, potenzialmente tramite una **reverse shell**.

0040130A	. 8B42 0C	MOV EAX,DWORD PTR DS:[EDX+C]	
0040130D	. 8B08	MOV ECX,DWORD PTR DS:[EAX]	
0040130F	. 8B11	MOV EDX,DWORD PTR DS:[ECX]	
00401311	. 8995 38FEFFFF	MOV DWORD PTR SS:[EBP-1C8],EDX	
00401317	. 68 0F270000	PUSH 270F	
0040131C	. FF15 B0404000	CALL DWORD PTR DS:[<&WS2_32.#9>]	[NetShort = 270F ntohs
00401322	. 66:8985 36FEFF	MOV WORD PTR SS:[EBP-1CA],AX	
00401323	. 66:C785 34FEFF	MOV WORD PTR SS:[EBP-1CC],2	
00401324	. 6A 10	PUSH 10	
0040132A	. 8D85 34FEFFFF	LEA EAX,DWORD PTR SS:[EBP-1CC]	
0040132A	. 50	PUSH EAX	
0040133B	. 8B8D FCFCFFFF	MOV ECX,DWORD PTR SS:[EBP-304]	
00401341	. 51	PUSH ECX	
00401342	. FF15 B4404000	CALL DWORD PTR DS:[<&WS2_32.#4>]	
00401348	. 8985 4CFEFFFF	MOV DWORD PTR SS:[EBP-1B4],EAX	
0040134E	. 83BD 4CFEFFFF	CMP DWORD PTR SS:[EBP-1B4],-1	
00401355	. JNZ 23	JNZ SHORT Malware_.0040137A	
00401357	. 8B95 FCFCFFFF	MOV EDX,DWORD PTR SS:[EBP-304]	
0040135D	. 52	PUSH EDX	
0040135E	. FF15 A8404000	CALL DWORD PTR DS:[<&WS2_32.#3>]	
00401364	. FF15 AC404000	CALL DWORD PTR DS:[<&WS2_32.#116>]	
0040136A	. 68 30750000	PUSH 7530	
0040136F	. FF15 A8404000	CALL DWORD PTR DS:[<&KERNEL32.Sleep>]	
00401375	. JMP 02FEFFFF	JMP Malware_.0040124C	
0040137A	. 8B85 FCFCFFFF	MOV EAX,DWORD PTR SS:[EBP-304]	
00401380	. 50	PUSH EAX	
00401381	. 83EC 10	SUB ESP,10	
00401384	. 8BCC	MOV ECX,ESP	
00401386	. 8B95 34FEFFFF	MOV EDX,DWORD PTR SS:[EBP-1CC]	
0040138C	. 8911	MOV DWORD PTR DS:[ECX],EDX	
0040138E	. 8B85 38FEFFFF	MOV EAX,DWORD PTR SS:[EBP-1C8]	
00401394	. 8941 04	MOV DWORD PTR DS:[ECX+4],EAX	
00401397	. 8B95 3CFEFFFF	MOV EDX,DWORD PTR SS:[EBP-1C4]	
0040139D	. 8951 08	MOV DWORD PTR DS:[ECX+8],EDX	
004013A0	. 8B85 40FEFFFF	MOV EAX,DWORD PTR SS:[EBP-1C0]	
004013A6	. 8941 0C	MOV DWORD PTR DS:[ECX+C],EAX	
004013A9	. E8 52FCFFFF	CALL Malware_.00401000	[Malware_.00401000
004013AE	. 83C4 14	ADD ESP,14	
004013B1	. 8B8D FCFCFFFF	MOV ECX,DWORD PTR SS:[EBP-304]	
004013B7	. 51	PUSH ECX	
004013B8	. FF15 A8404000	CALL DWORD PTR DS:[<&WS2_32.#3>]	
004013BE	. FF15 AC404000	CALL DWORD PTR DS:[<&WS2_32.#116>]	
004013C4	. 68 30750000	PUSH 7530	
004013C9	. FF15 A8404000	CALL DWORD PTR DS:[<&KERNEL32.Sleep>]	
004013CF	. JMP 78FEFFFF	JMP Malware_.0040124C	
004013D4	. XOR EAX,EAX		
004013D6	. POP EDI		
004013D7	. POP ESI		
004013D8	. MOV ESP,EBP		
004013DA	. POP EBP		

```
[pWSAData  
RequestedVersion = 202 (2.2.)  
WSAStartup  
  
[Flags = 0  
Group = 0  
pWSAProtocol = NULL  
Protocol = IPPROTO_TCP  
Type = SOCK_STREAM  
Family = AF_INET  
WSASocketA  
  
[Arg2  
Arg1  
Malware_.00401089  
  
[Name  
gethostbyname
```


BONUS: spiegare a grandi linee il funzionamento del malware

Ho notato un'altra cosa durante le numerose esecuzioni del malware in OllyDBG: termina troppo in fretta, quasi come se non facesse nulla. Scorrendo ossessivamente il codice, ho trovato molte chiamate a varie funzioni, tra cui la **strcmp**. Questa funzione, come ci spiega la nostra cara Microsoft (<https://learn.microsoft.com/it-it/cpp/c-runtime-library/reference/strcmp-wcsncmp-mbsncmp>), **confronta due stringhe**. Visto che l'esecuzione si conclude così rapidamente e considerando l'uso di questa funzione, potrei azzardare ipotizzando che un eventuale payload si inneschi solo dopo un confronto di stringhe basato sul risultato.



Grazie

Flavio Scognamiglio