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)

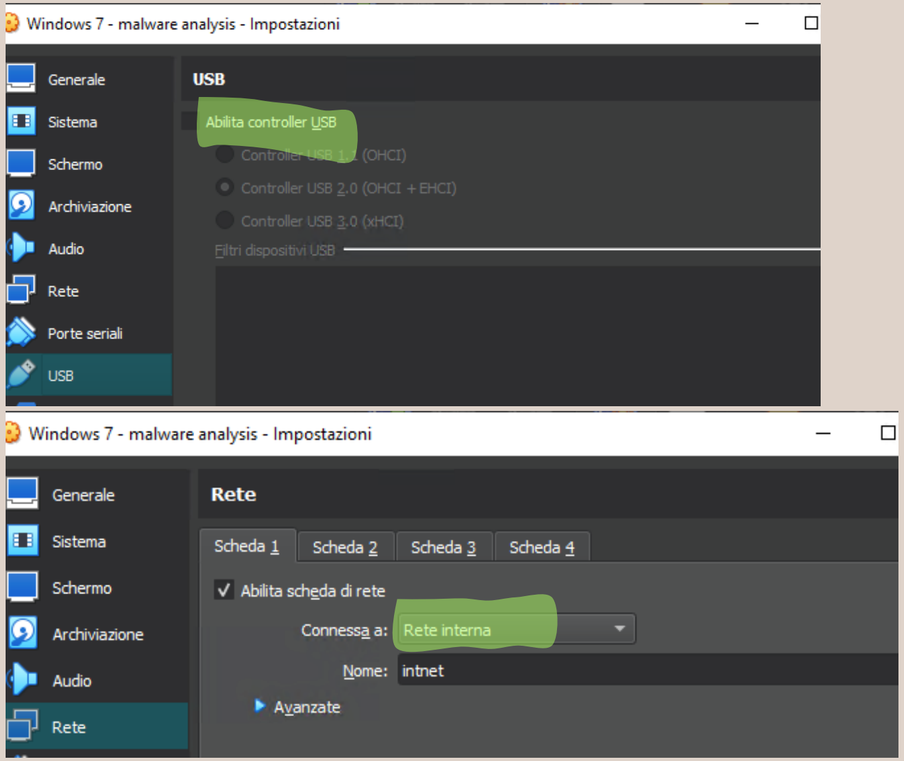
• Inserite un breakpoint software all'indirizzo 004015A3. Qual è il valore del registro EDX? (2)
Eseguite a questo punto uno «step-into». Indicate qual è ora il valore del registro EDX (3) motivando la risposta (4). Che istruzione è stata eseguita? (5)

• 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

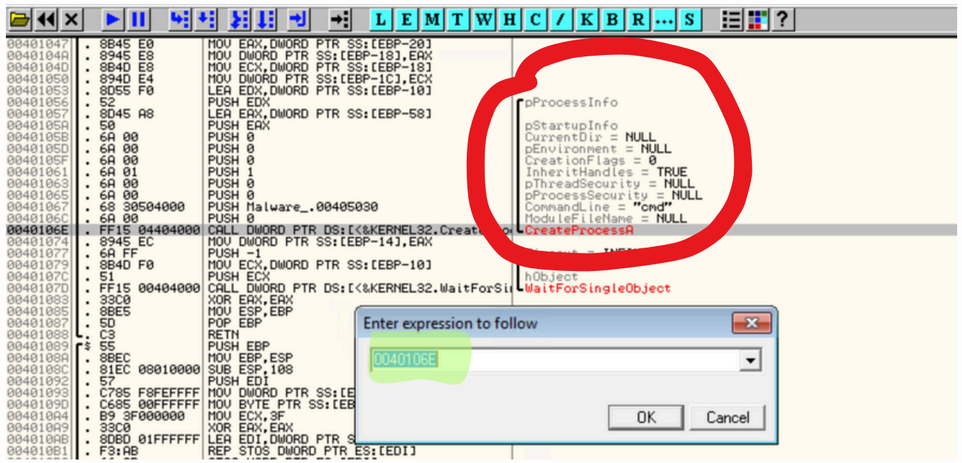
Come visto nella lezione teorica, OllyDbg è un **debugger** per Windows che permette di **analizzare** il codice eseguibile di un programma **durante** l'esecuzione.

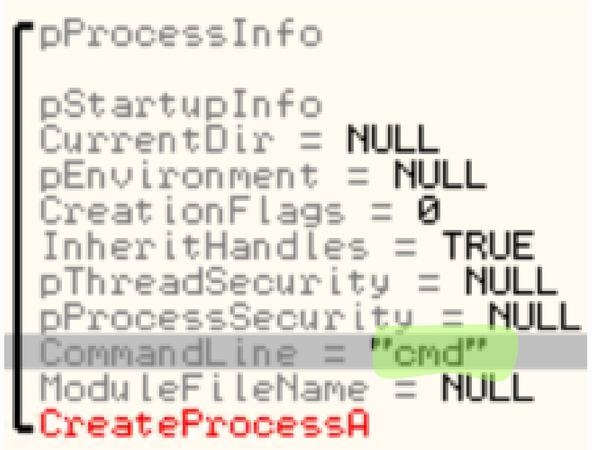
Isolo la VM in **rete interna** per proteggerla, poiché devo eseguire e analizzare un malware con OllyDbg. Disattivo **USB** e altre funzionalità per garantire la sicurezza.



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?

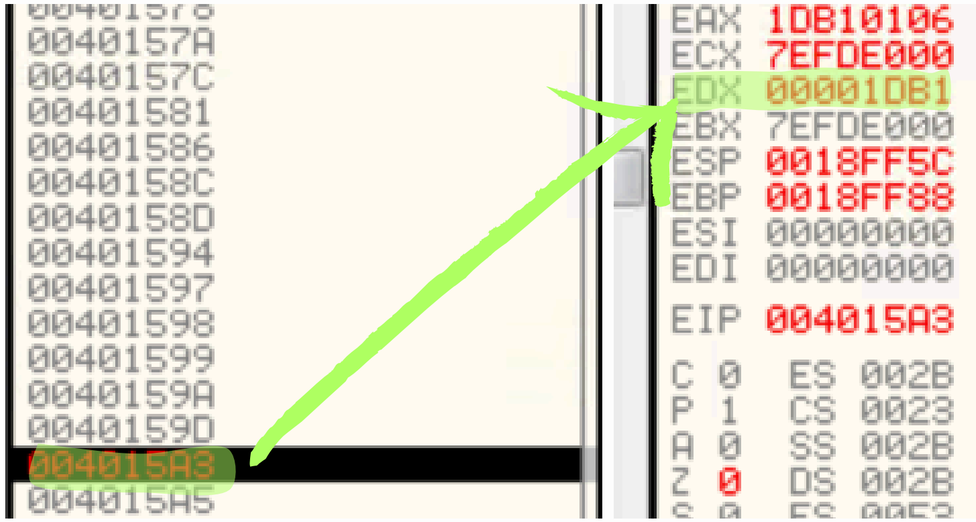
Una volta raggiunto l'indirizzo **0040106E** utilizzando la scorciatoia CTRL+G, noto che il malware passa una stringa che punta a **cmd.exe**. Questo potrebbe indicare un tentativo di eseguire cmd per operazioni malevole, come l'apertura di una **reverse shell**.

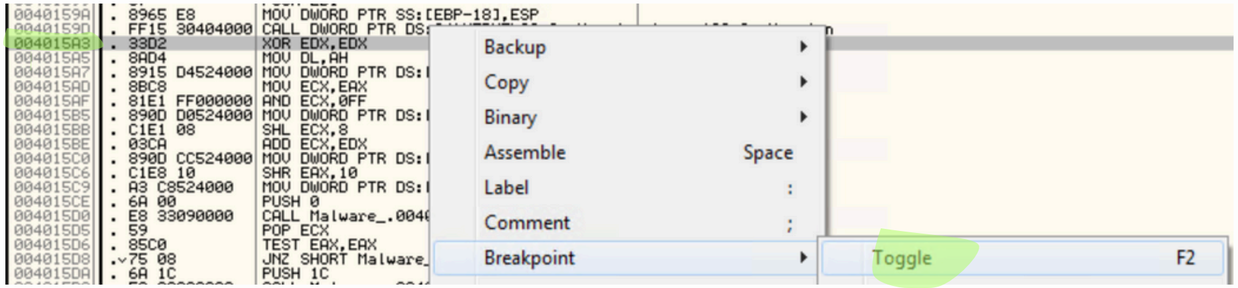




2 - Inserite un breakpoint software all'indirizzo 004015A3. Qual è il valore del registro EDX?

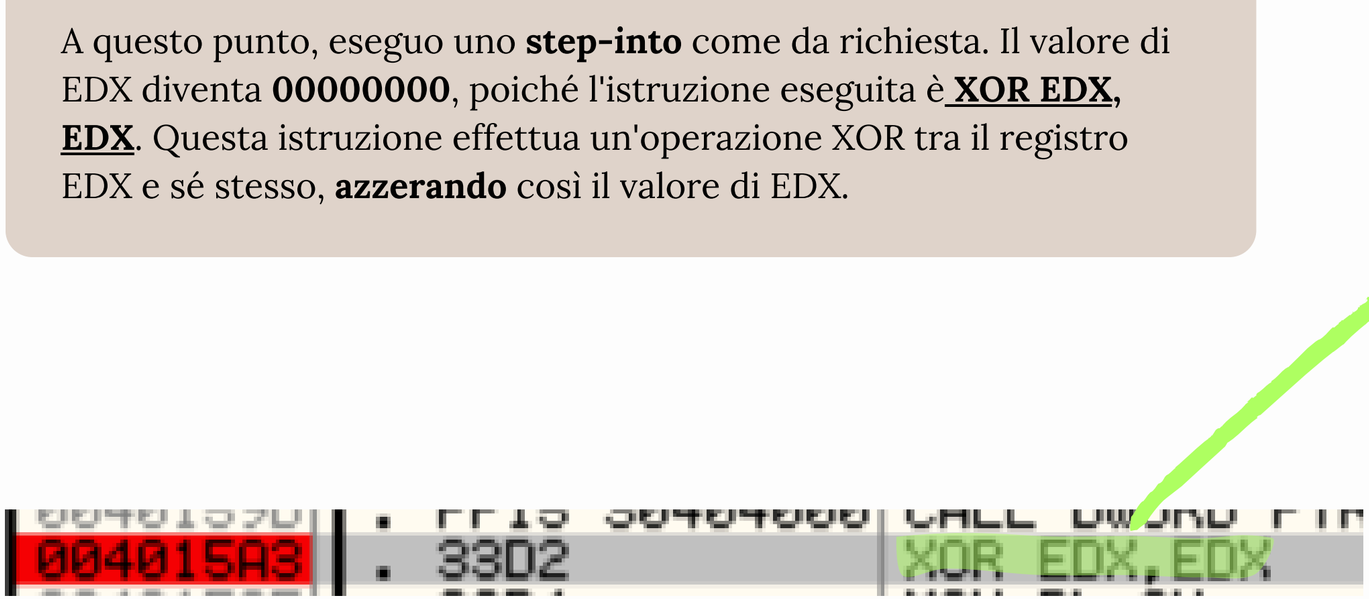
Come da richiesta, vado all'indirizzo 004015A3 e imposto un breakpoint. Dopodichè riprendo l'esecuzione del malware fino a che non arriva al breakpoint. Il valore del registro EDX è **00001DB1.**

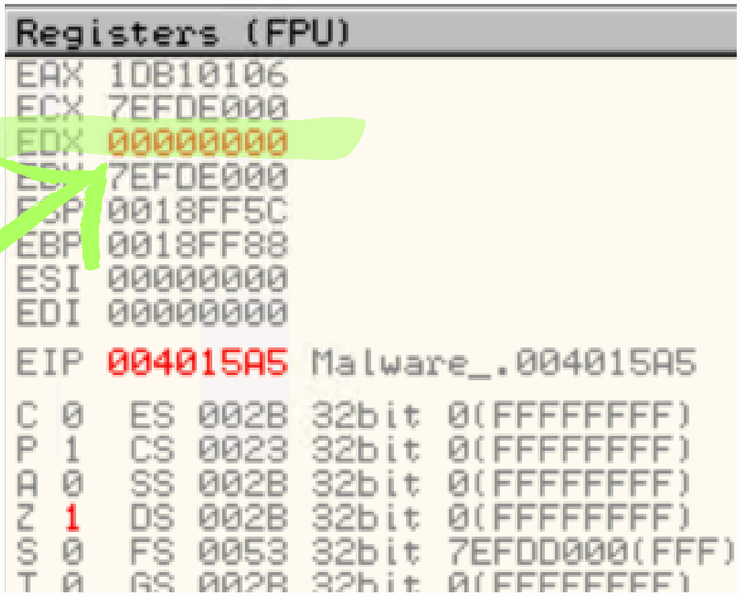




3, 4, 5 - *Eseguite a questo punto uno «step-into». Indicate qual è ora il valore del registro EDX motivando la risposta. Che istruzione è stata eseguita?*

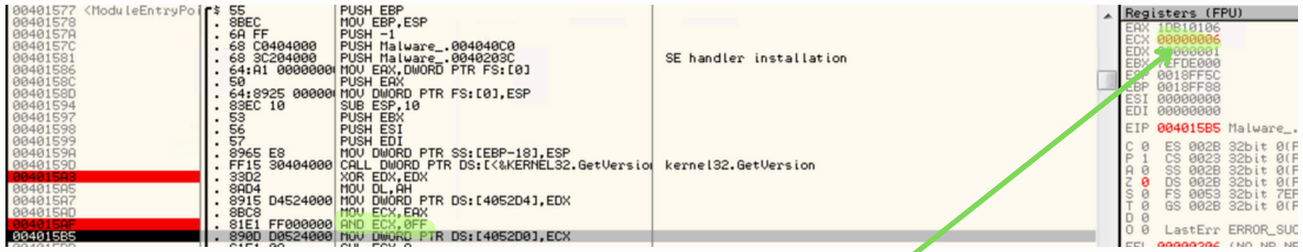
A questo punto, eseguo uno **step-into** come da richiesta. Il valore di EDX diventa **00000000**, poiché l'istruzione eseguita è **XOR EDX, EDX**. Questa istruzione effettua un'operazione XOR tra il registro EDX e sé stesso, **azzerando** così il valore di EDX.



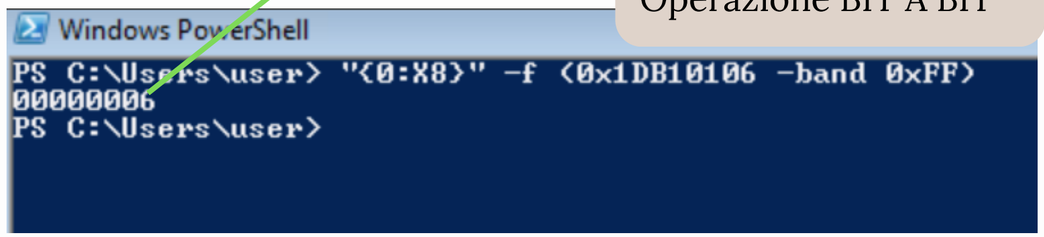


6, 7, 8 - Inserite un secondo breakpoint all'indirizzo di memoria 004015AF. Qual è il valore del registro ECX? Eseguite un step-into. Qual è ora il valore di ECX? Spiegate quale istruzione è stata eseguita.

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.

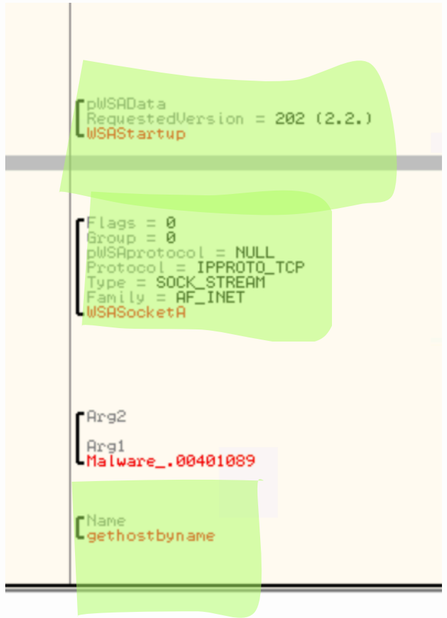


Operazione BIT A BIT



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**.



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-wcscmp-mbscmp), **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