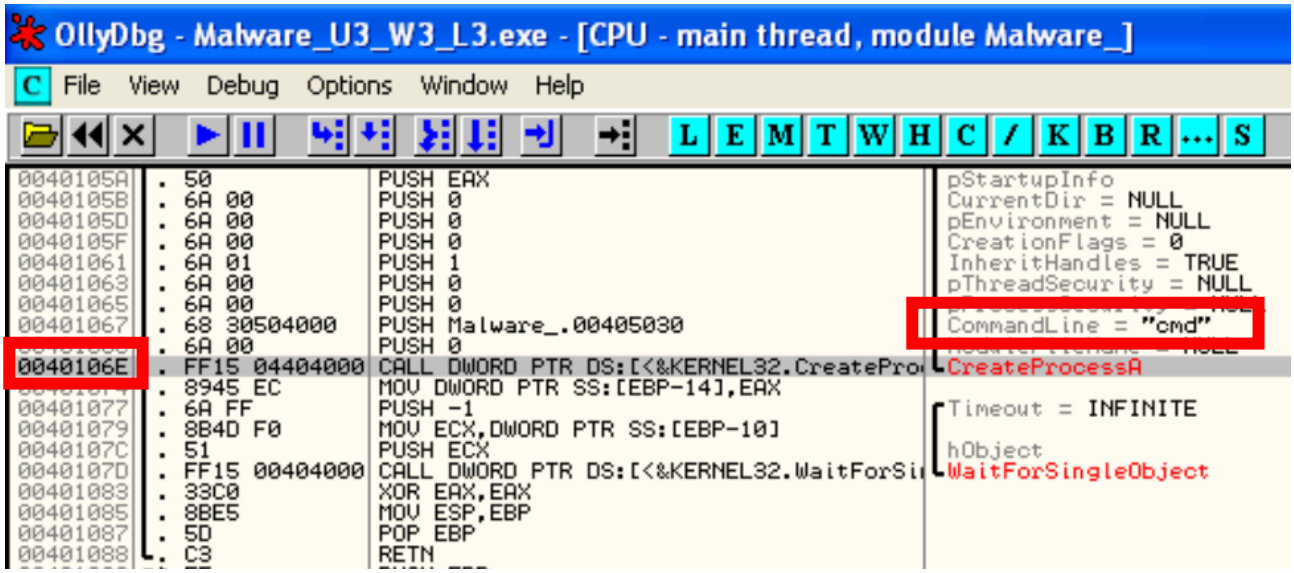
ANALISI DINAMICA AVANZATA

TASK:

1. **Qual è il valore del parametro CommandLine che viene passato sullo stack grazie alla chiamata di funzione CreateProcess locata all'indirizzo 0040106E**
2. **Inserire un breakpoint software all'indirizzo 004015A3. Qual è il valore del registro EDX?**
3. **Eseguire uno step-into ed indicare il valore del registro EDX dopo averlo fatto**
4. **Motivare la risposta del task 3**
5. **Che istruzione è stata eseguita?**
6. **Inserire un secondo breakpoint all'indirizzo di memoria 004015AF. Qual è il valore del registro ECX?**
7. **Eseguire uno step-into. Qual è ora il valore di ECX?**
8. **Spiegare quale istruzione è stata eseguita**
9. **BONUS: Spiegare a grandi linee il funzionamento del malware.**

TASK 1: Qual è il valore del parametro CommandLine che viene passato sullo stack grazie alla chiamata di funzione CreateProcess locata all'indirizzo 0040106E

Ci si sposta nell'indirizzo di memoria **0040106E**, dove troviamo il parametro CommandLine come valore **cmd**, prompt dei comandi **Windows**.

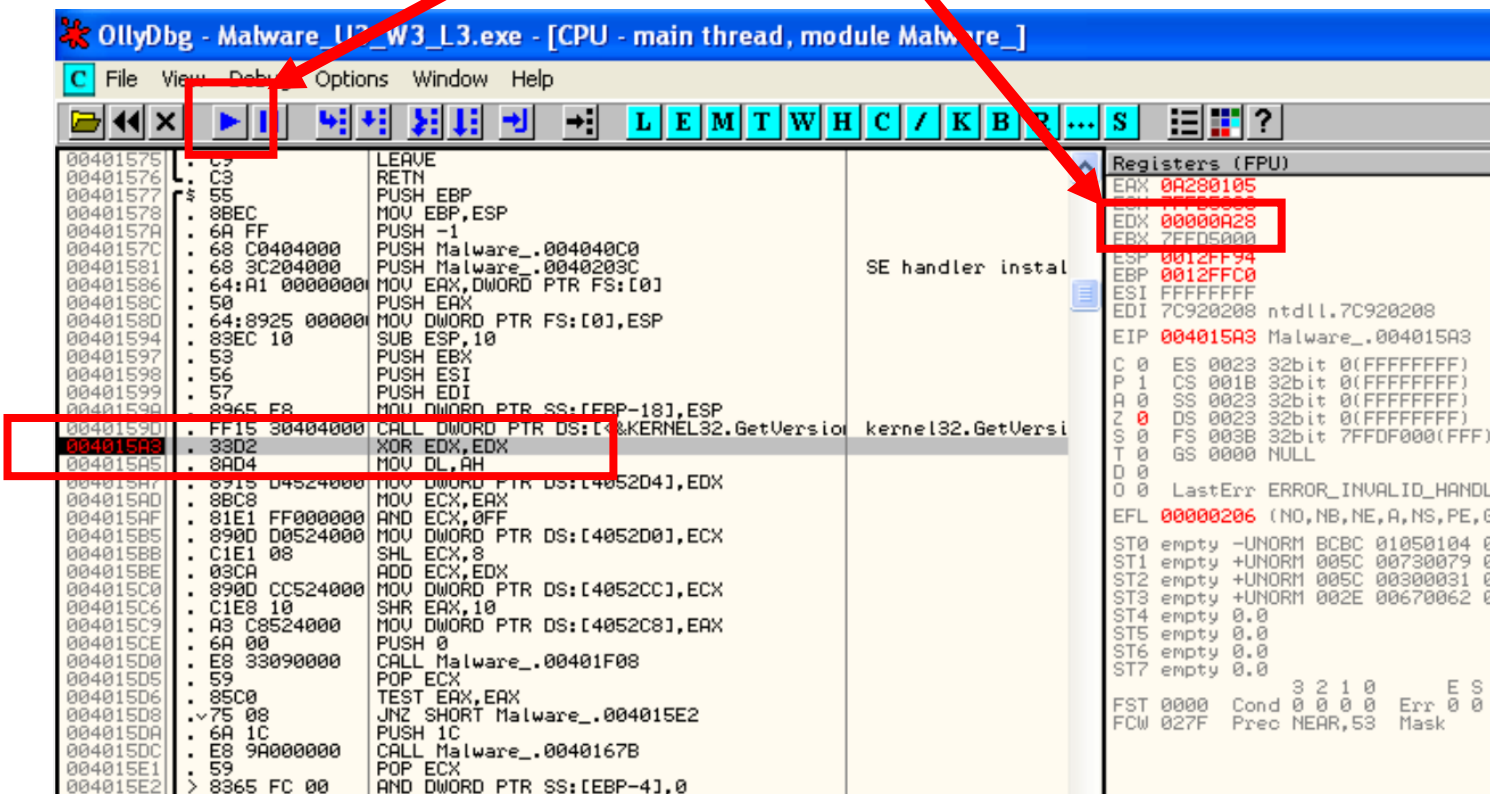


TASK 2: **Inserire un breakpoint software all'indirizzo 004015A3. Qual è il valore del registro EDX?**

Un software breakpoint permette di fermare il programma quando una data istruzione viene eseguita. Ad esempio, è possibile configurare un software breakpoint su una chiamata di funzione per studiarne i dettagli, oppure all'inizio di un ciclo per capire di cosa si tratta.

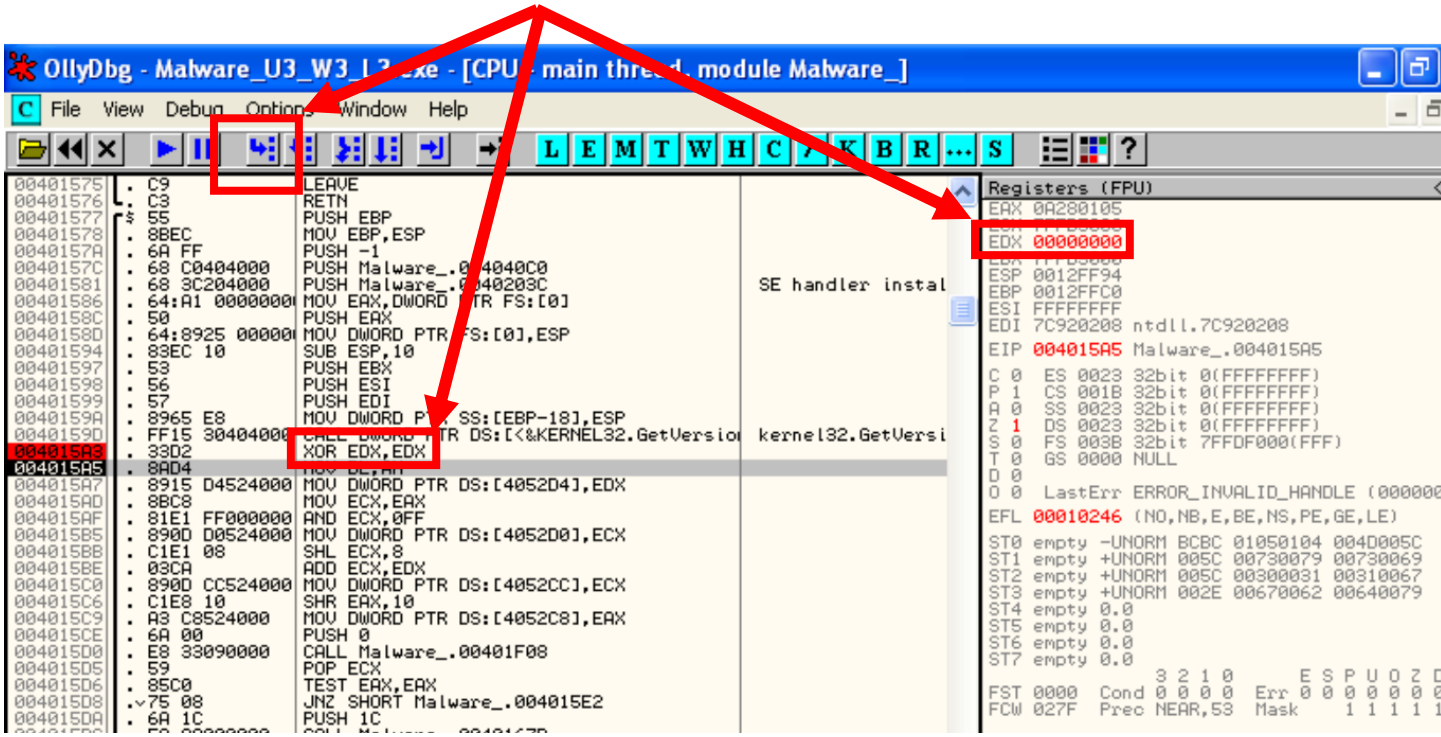
Dunque ci sposta all'indirizzo **004015A3** dove con tasto destro del mouse → Breakpoint → Toggle inseriamo un software breakpoint (rettangolo in rosso) e poi **PLAY.**

Si nota che il **valore** del **registro EDX è 00000A28**



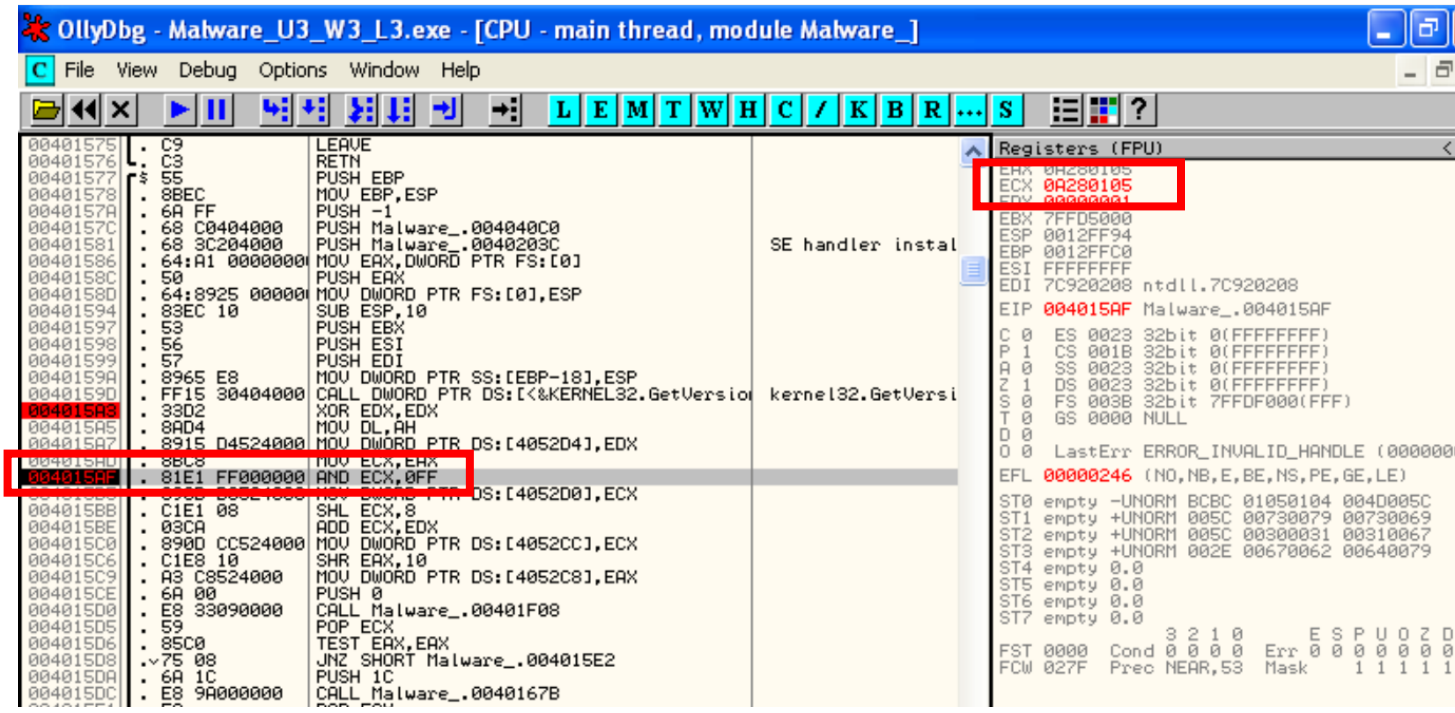
TASK 3-4-5: Eseguire uno STEP-INTO ed indicare il valore del registro EDX dopo averlo fatto, motivare la risposta del task 3. Che istruzione è stata eseguita?

STEP-INTO è una tecnica di debug che ci consente di entrare nel codice della funzione a fronte di una chiamata di funzione, permettendoci di **analizzare il suo contenuto e la sua implementazione**. È utile **quando si desidera analizzare il comportamento di una specifica funzione custom** anziché una funzione standard di libreria.
Dopo aver eseguito la tecnica di STEP-INTO, si nota che **il valore del registro EDX è pari a ZERO**, in quanto l'istruzione **XOR EDX, EDX (dentro la quale siamo entrati)** inizializza la variabile EDX a 0.



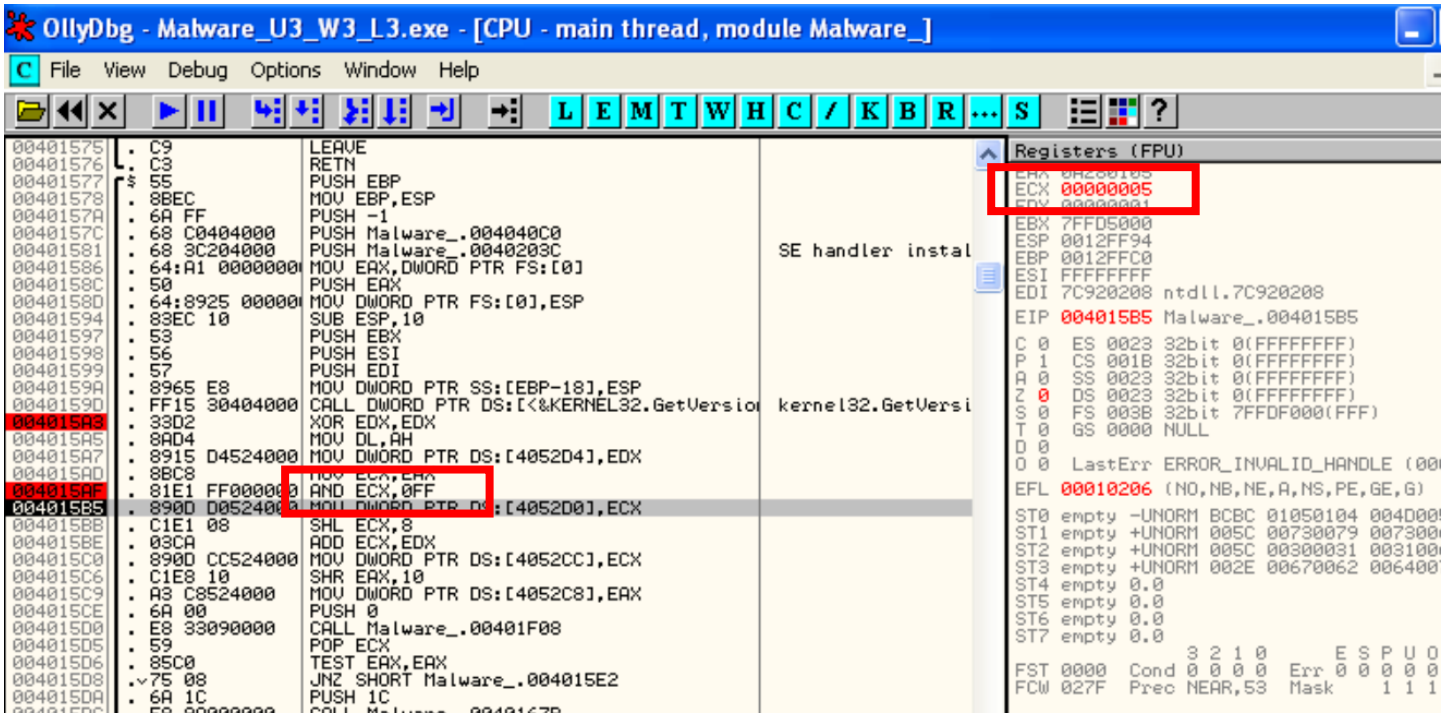
TASK 6: Inserire un secondo breakpoint all'indirizzo di memoria 004015AF. Qual è il valore del registro ECX?

Dopo aver inserito un **secondo breakpoint** ed aver cliccato **PLAY**, il **valore del registro ECX è pari a 0A280105**



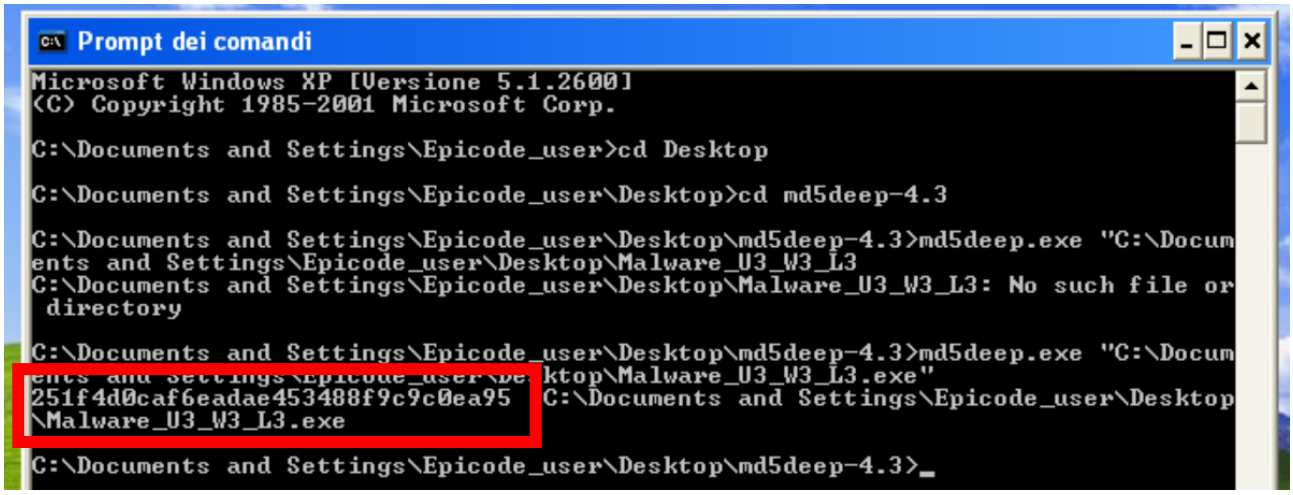
TASK 7-8: Eseguire uno STEP-INTO. Qual è ora il valore di ECX? Spiegare quale istruzione è stata eseguita

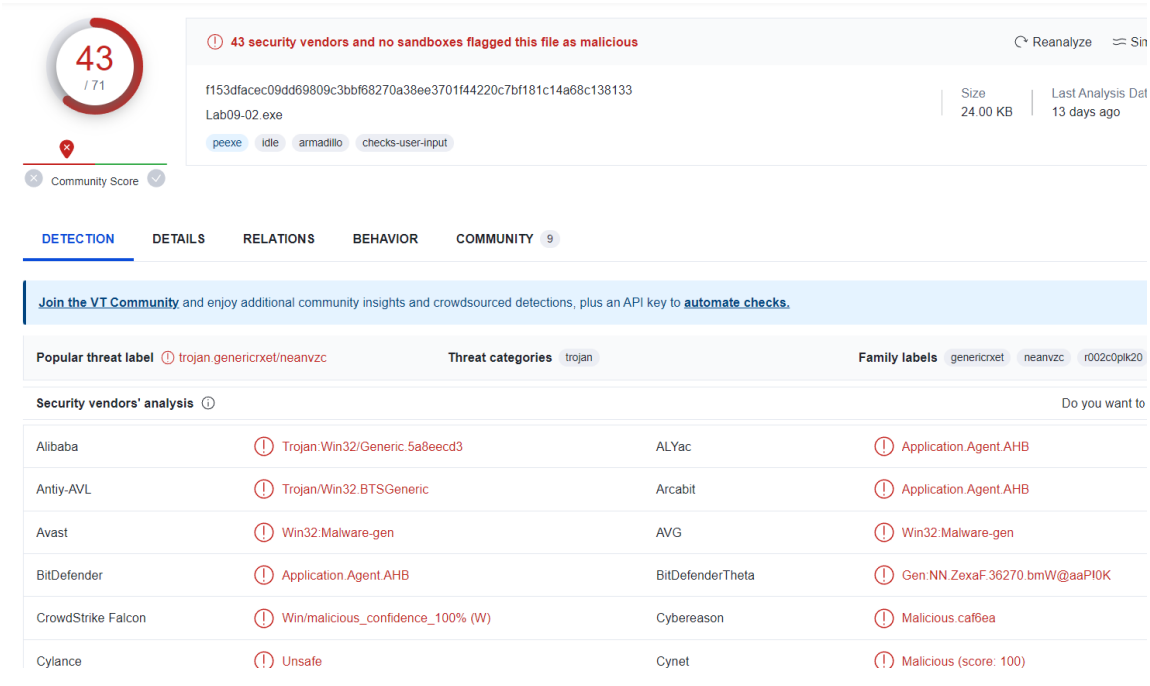
Eseguiamo nuovamente uno **STEP-INTO. Il valore del registro ECX** attualmente è **00000005**
In questo caso è stata eseguita l'istruzione AND ECX, 0FF che esegue l'AND logico tra il valore del registro ECX il valore 0FF.



BONUS: Spiegare a grandi linee il funzionamento del malware.

Ricavato **l'hash** da **md5depp** e inserito su **VirusTotal,** si apprende che **43 vendor su 71** hanno identificato il file come **maligno**, nello specifico come un **Trojan.**





VirusTotal ci informa che il file eseguibile importa le librerie:

- **KERNEL32.dll:** libreria piuttosto comune che **contiene le funzioni principali per interagire con il sistema operativo**, ad esempio la **manipolazione dei file, la gestione della memoria**
- **WS2_32.dll**: **libreria che contiene le funzioni di network**, come le **socket**, le funzioni **connect, bind**.

Header

Target Machine	Intel 386 or later processors and compatible processors
Compilation Timestamp	2011-04-30 16:41:06 UTC
Entry Point	5495
Contained Sections	3

Sections

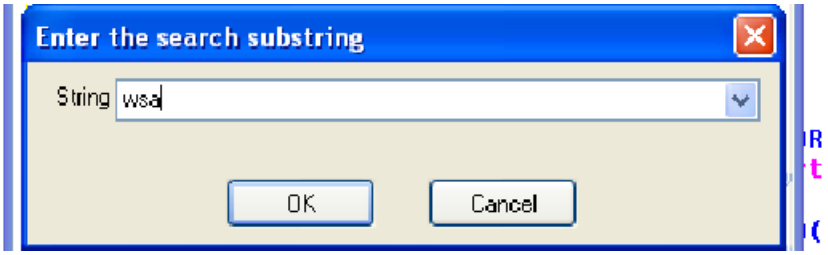
Name	Virtual Address	Virtual Size	Raw Size	Entropy	MD5	Chi2
.text	4096	11366	12288	6.25	6bc8373e9f753cd282af864ad71269c2	131912.97
.rdata	16384	2000	4096	3.17	40e860eea4ba1be6d0d3936b0dbbf076	393924.88
.data	20480	2012	4096	0.69	6103ee24e83b7788dacddba6a1fb83d0	923851.88

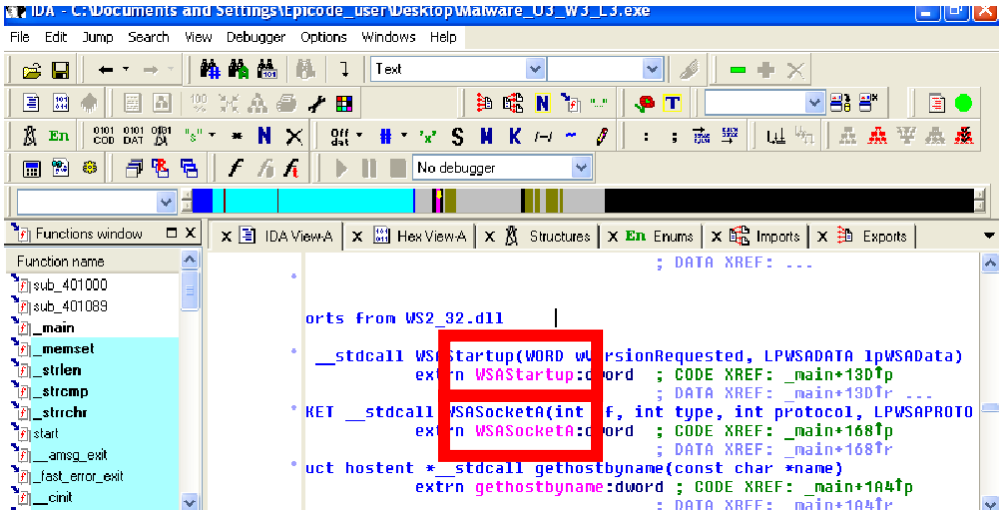
Imports

- KERNEL32.dll
- WS2_32.dll

Ho utilizzato altresì **IDA Pro**, dove tramite **JUMP BY NAME**, ho rintracciato la funzione **WSAStartup**, **usata per allocare risorse che verranno poi utilizzate dalle librerie del networking.**

Vi è altresì la funzione WSASocketA, la quale potrebbe essere usata per creare un nuovo socket utilizzando la libreria WS2_32.dll.





CONCLUSIONI

Si è appurato che il file eseguibile oggetto d'interesse importa la libreria **KERNEL32.dll,** libreria usata per la creazione di un nuovo processo "**cmd**" tramite la funzione "**CreateProcess**".

L'eseguibile oggetto di interesse importa altresì la libreria **WS2_32.dll**, grazie alla quale vengono usate le funzioni **WSAStartup** per allocare risorse che verranno usate per il networking e la funzione **WSASocketA**, la quale viene usata per creare un nuovo socket.

Al netto di questa breve analisi, si può ipotizzare che ci troviamo di fronte ad una probabilissima **backdoor**.