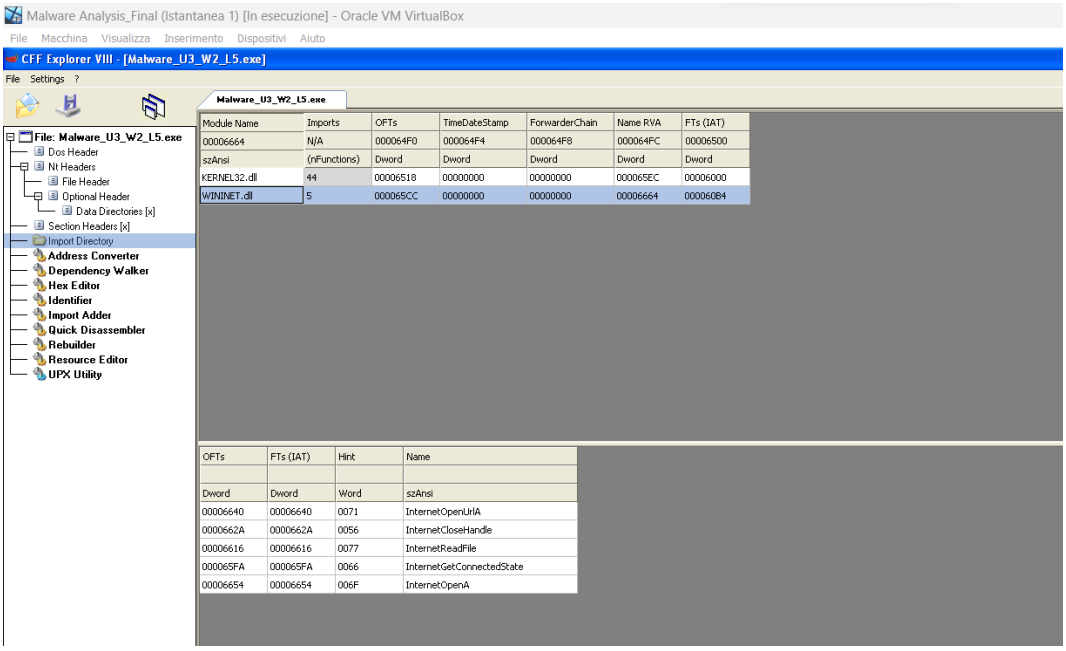
Di Edoardo Castelli

Epicode Unit 3 Week 2

Progetto Settimanale - Malware Analysis

Il lavoro di questa settimana è diviso in tre Macro-sezioni. Nella prima, si procede con un'analisi basica statica del probabile Malware fornito dalla traccia. L'eseguibile verrà analizzato come di consueto con il Tool "CFF Explorer" per il riconoscimento delle librerie e delle funzioni utilizzate e per il riconoscimento delle Sezioni del programma.



Una volta passato il Programma da Analizzare al Tool, si può andare ad indagare nella sezione "Import Directory" per verificare, come già anticipato, quali librerie e funzioni vengano utilizzate dal probabile software malevolo. Si può notare a colpo d'occhio che il programma importa la libreria KERNEL32.dll per l'interazione con il Sistema Operativo e la libreria WININET.dll per l'utilizzo dei servizi di rete.

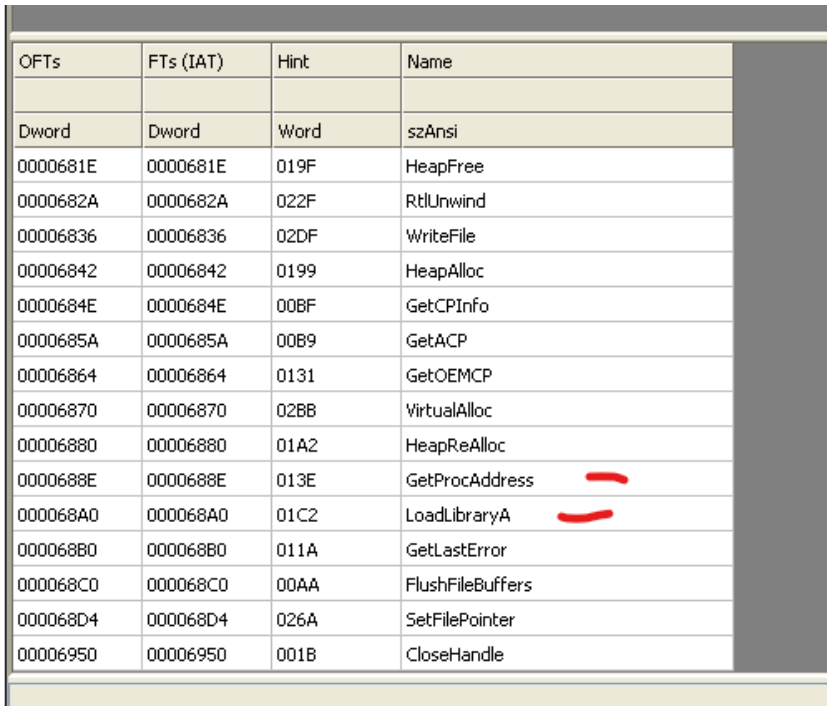
Si può inoltre notare che il programma utilizza ben quarantaquattro funzioni contenute nella prima libreria e cinque contenute nella seconda. Andando a clickare sul nome della libreria stessa sarà possibile indagare ulteriormente e nello specifico di quali funzioni si tratta.

OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
000065E4	000065E4	0296	Sleep
00006940	00006940	027C	SetStdHandle
0000692E	0000692E	0156	GetStringTypeW
0000691C	0000691C	0153	GetStringTypeA
0000690C	0000690C	01C0	LCMapStringW
000068FC	000068FC	01BF	LCMapStringA
000068E6	000068E6	01E4	MultiByteToWideChar
00006670	00006670	00CA	GetCommandLineA
00006682	00006682	0174	GetVersion
00006690	00006690	007D	ExitProcess
0000669E	0000669E	029E	TerminateProcess
000066B2	000066B2	00F7	GetCurrentProcess
000066C6	000066C6	02AD	UnhandledExceptionFilter
000066E2	000066E2	0124	GetModuleFileNameA
000066F8	000066F8	00B2	FreeEnvironmentStringsA

Si notino in particolare le funzioni evidenziate. In quanto analizzato trovo funzioni per la gestione dei processi ("GetCurrentProcess, "TerminateProcess"), una funziona per la gestione della linea di comando ("GetCommandLineA"), e una funzione di sospensione ("Sleep") che permetterebbe di impostare l'eventuale periodo di attività e inattività dei thread e dei processi relativi al probabile Malware.

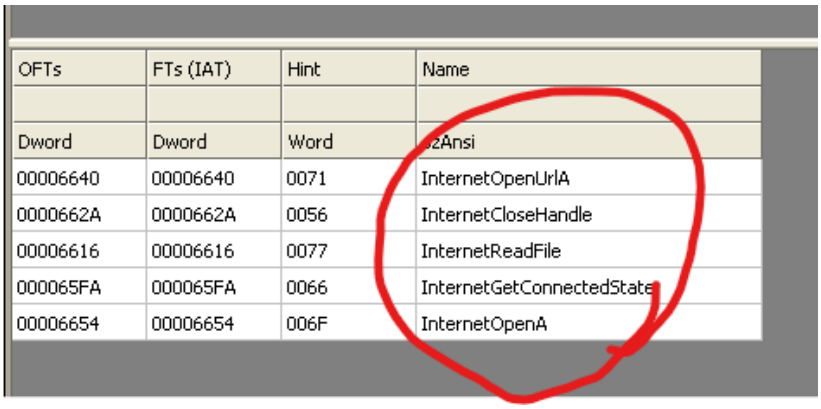
OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
00006786	00006786	0152	GetStdHandle
00006796	00006796	0115	GetFileType
000067A4	000067A4	0150	GetStartupInfoA
000067B6	000067B6	0126	GetModuleHandleA
000067CA	000067CA	0109	GetEnvironmentVariableA
000067E4	000067E4	0175	GetVersionExA
000067F4	000067F4	019D	HeapDestroy
00006802	00006802	019B	HeapCreate
00006810	00006810	02BF	VirtualFree
0000681E	0000681E	019F	HeapFree
0000682A	0000682A	022F	RtlUnwind
00006836	00006836	02DF	WriteFile
00006842	00006842	0199	HeapAlloc
0000684E	0000684E	00BF	GetCPInfo
0000685A	0000685A	00B9	GetACP

Notiamo inoltre una serie di funzioni per la gestione degli "Heap" di memoria e la funzione "WriteFile", oltre alle funzioni "GetProcAddress" e "LoadLibrary", che come sappiamo potrebbero indicare ad un malware il recupero di particolari librerie e funzioni durante l'esecuzione (runtime).



OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	szAnsi
0000681E	0000681E	019F	HeapFree
0000682A	0000682A	022F	RtlUnwind
00006836	00006836	02DF	WriteFile
00006842	00006842	0199	HeapAlloc
0000684E	0000684E	00BF	GetCPInfo
0000685A	0000685A	00B9	GetACP
00006864	00006864	0131	GetOEMCP
00006870	00006870	02BB	VirtualAlloc
00006880	00006880	01A2	HeapReAlloc
0000688E	0000688E	013E	GetProcAddress
000068A0	000068A0	01C2	LoadLibraryA
000068B0	000068B0	011A	GetLastError
000068C0	000068C0	00AA	FlushFileBuffers
000068D4	000068D4	026A	SetFilePointer
00006950	00006950	001B	CloseHandle

Per dovere di sintesi, passo all'analisi delle funzioni utilizzate a seguito dell'importazione della libreria WININET, ritenendo bastevoli le funzioni riportate sopra per un'iniziale identificazione del Malware, anche se si potrebbero ritenere tutte singolarmente interessanti.



OFTs	FTs (IAT)	Hint	Name
Dword	Dword	Word	zAnsi
00006640	00006640	0071	InternetOpenUrlA
0000662A	0000662A	0056	InternetCloseHandle
00006616	00006616	0077	InternetReadFile
000065FA	000065FA	0066	InternetGetConnectedState
00006654	00006654	006F	InternetOpenA

Questa seconda libreria e le funzioni ad essa correlate ci confermano la comunicazione del programma alla rete internet. Alla luce di quanto individuato nella libreria KERNEL32, in particolare alla funzione "GetCommandLine", inizio ad essere

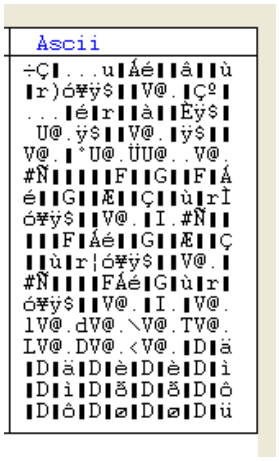
discretamente convinto che si tratti di un Trojan o una Backdoor che va a fornire ad un utente remoto il controllo almeno parziale della macchina probabilmente infetta.

Ritenendo le informazioni raccolte più che esplicite per un'iniziale profilazione del Programma, si prosegue con l'analisi delle sezioni dello stesso, utilizzando la funzione "Section Headers" di CFF.

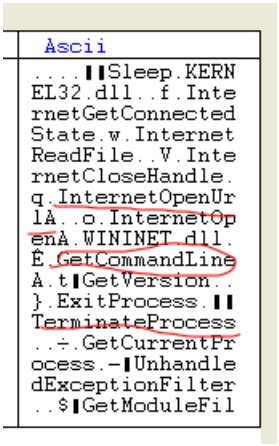
Byte[8]	Dword	Dword	Dword	Dword	Dword	Dword	Word	Word	Dword
.text	00004A78	00001000	00005000	00001000	00000000	00000000	0000	0000	60000020
.rdata	0000095E	00006000	00001000	00006000	00000000	00000000	0000	0000	40000040
.data	00003F08	00007000	00003000	00007000	00000000	00000000	0000	0000	C0000040

Si individuano in questo caso tre sezioni:

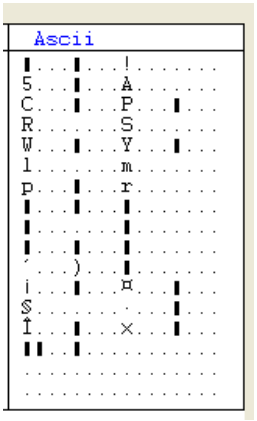
- **".text": contenente il codice vero e proprio dell'eseguibile, riportato anche nello specifico nella tabella sottostante in Ascii;**



- **".rdata": contenente le librerie e le funzioni già individuate precedentemente, ma sulle quali comunque si effettua un controllo di "cross-check": si notino infatti, tra le altre, le funzioni "GetCommandLine" e "Terminate Process";**



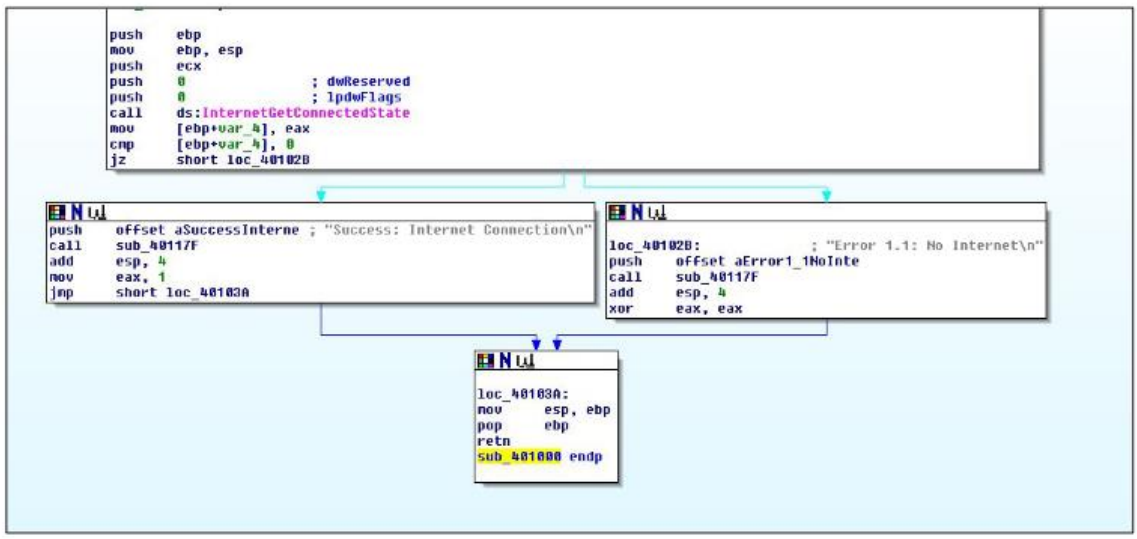
- **".data": contenente i dati e le variabili globali dell'eseguibile, che in quanto tali devono essere sempre "a disposizione" del Malware. Di seguito una sezione dell'Ascii della sezione in questione.**



Alla luce di quanto analizzato e rimanendo coerenti alle richieste, quindi limitandosi allo studio appena eseguito, pur non avendo assoluta conferma di quanto asserito, si può affermare con discreta certezza che:

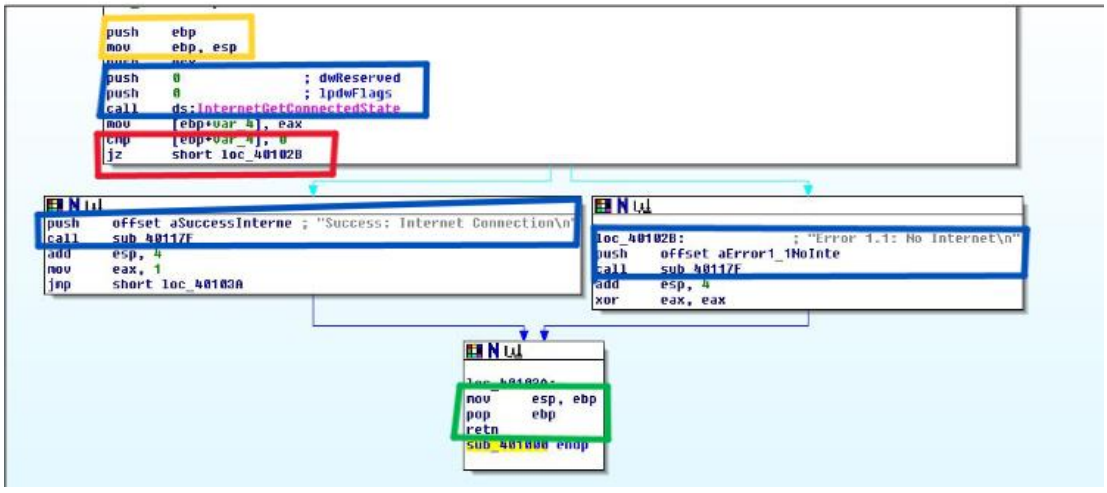
- **Il malware comunichi con un utente remoto tramite rete Internet;**
- **Il Malware fornisca a tale utente una Riga di Comando per operare sulla macchina infetta;**
- **Il malware permette all'utente malevolo di creare e gestire File e Processi sulla stessa macchina;**
- **Il Malware potrebbe inviare all'esterno i dati raccolti sul Kernel.**

Nella seconda Macro-sezione viene richiesto invece di identificare e descrivere eventuali costrutti noti del Corrispettivo in Linguaggio C del Codice Assembly Fornito.



Per praticità, tale analisi verrà dettagliata sia graficamente, che testualmente. Si fa inoltre presente che un'analisi riga per riga del codice fornito è presente nella mia personale repository GitHub fino alla riga del salto "Unconditional" che il programma esegue in caso di connessione Internet presente (https://github.com/EdoardoCastelli/Epicode1/blob/main/Epicode_Assembly_and_C.pdf).

Tornando al codice nella sua interezza, segue dapprima una rappresentazione grafica con successiva descrizione.



In Ocra: Vengono racchiuse nel campo evidenziato le righe necessarie alla creazione della stack per l'avvio della funzione.

In Blu: Vengono individuate tre funzioni principali: La prima, già analizzata anche nel lavoro precedente, è la funzione "InternetGetConnectedSate", necessaria per la verifica di connettività internet una volta passati i parametri necessari al suo funzionamento (0,0; in questo caso). Successivamente vediamo due funzioni di stampa, una per ogni possibilità data dall' **IF STATEMENT** che segue, con relativi stringhe da presentare all'utente in base all'esito del tentativo di connessione.

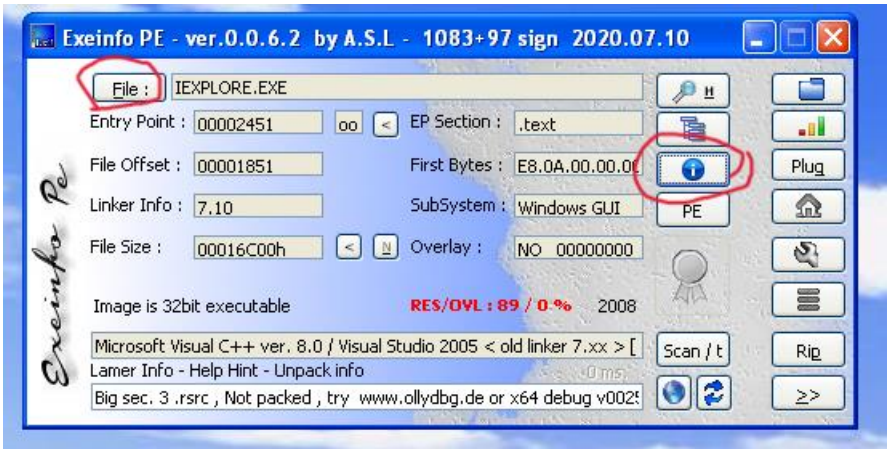
In Rosso: Come già anticipato in questa sezione si può individuare un costrutto "If" con le relative condizioni di funzionamento: la riga in cui appare "cmp" mette a paragone il dato contenuto in "var_4" con i parametri passati dalla funzione e, in base al salto condizionale alla riga successiva ("Jump Zero"), riporterà ad una delle due locazioni di memoria successivamente riportate in base all'esito della comparazione. Se la flag risulta essere zero, il salto porterà alla stampa del messaggio che ci comunica che c'è connettività Internet (a sinistra in figura), altrimenti l'esecuzione salterà all'indirizzo di memoria che contiene il messaggio di Errore (destra in figura).

In verde: Infine, dopo delle istruzioni leggermente diverse in base ai casi – abbiamo un "salto non condizionato" qualora ci fosse connettività ed un reset dei registri qualora non fosse così- possiamo notare la rimozione della stack alla fine dell'esecuzione, resa particolarmente evidente prima dal "ritorno" (mov) dell'Extended Stack Pointer alla base della stack sull'Extend Base Pointer; e poi dal "pop" di quest'ultimo, che segnala definitivamente la risoluzione della Pila.

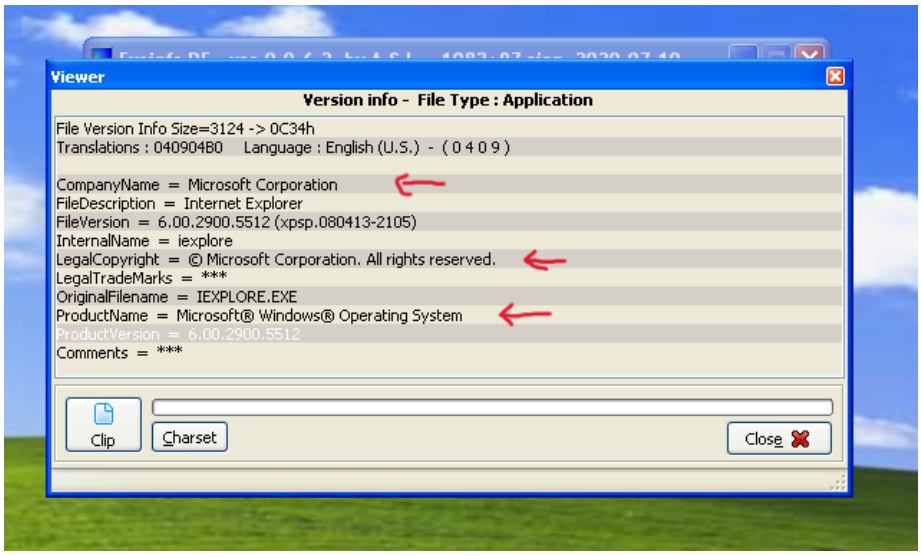
Terza Macro-sezione/Bonus:

Nell'ultima parte del progetto è richiesto di comprovare ad un ipotetico dipendente junior l'innocuità dell'eseguibile "IEXPLORE.exe".

Si decide di dimostrarlo attraverso una serie di tool e il "crosscheck" dei dati estrapolati da essi. Dapprima, l'eseguibile viene analizzato con "Exeinfo PE", un tool semplice ma davvero molto utile per verificare informazioni e licenze di un eseguibile. Per utilizzarlo basterà aprirlo ed inserire nell'apposita stringa di ricerca il nome del programma da analizzare, o selezionarlo una volta clickata l'icona "file".



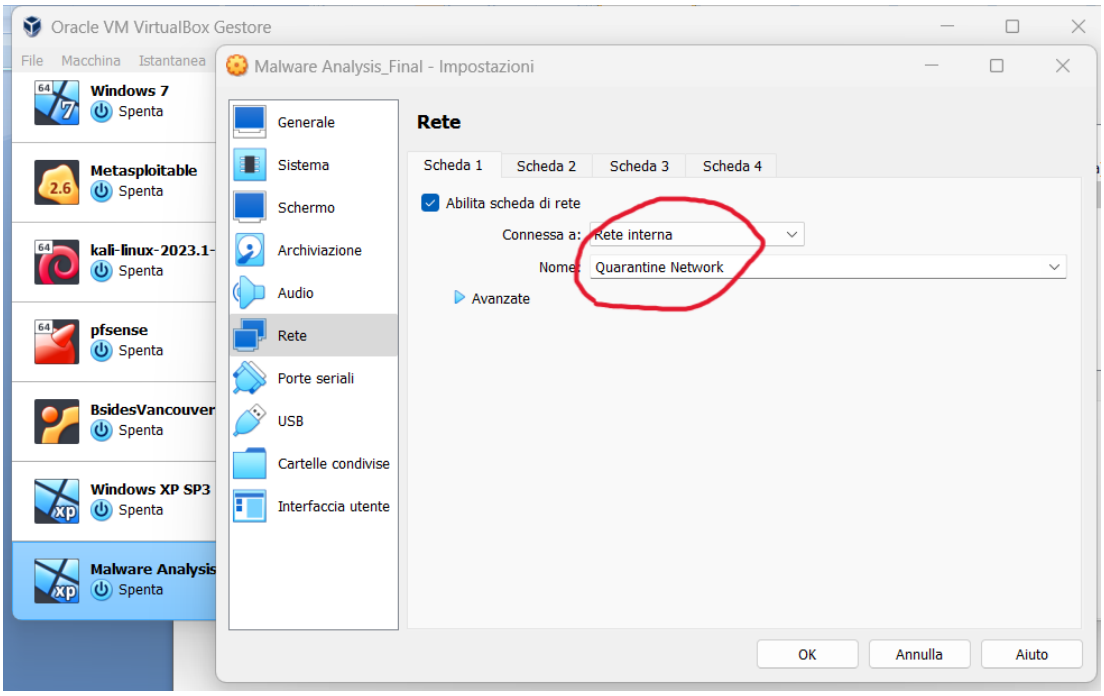
Fatto ciò, basterà clickare su "info" (cerchiato in figura) per verificare nome, licenze e copyright del Programma analizzato. Segue screenshot di come il tool presenterà i dati.



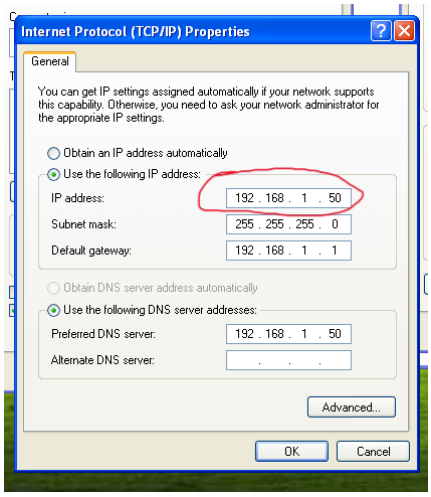
Si noti che il programma restituisce il nome della Compagnia che ha sviluppato l'eseguibile, il "LegalCopyright", il nome del file, il nome commerciale del prodotto, e altre informazioni che possono già dare una serie di conferme sull'autenticità dell'eseguibile stesso.

Per ulteriore conferma, si decide di analizzare il traffico generato da tale software insieme al dipendente Junior. Pur avendo già una convinzione più che nutrita dell'autenticità del software, si stabilisce anzitutto di isolare la macchina da indagare in una sorta di "Rete di Quarantena" prima di abilitare la scheda di rete, di modo da

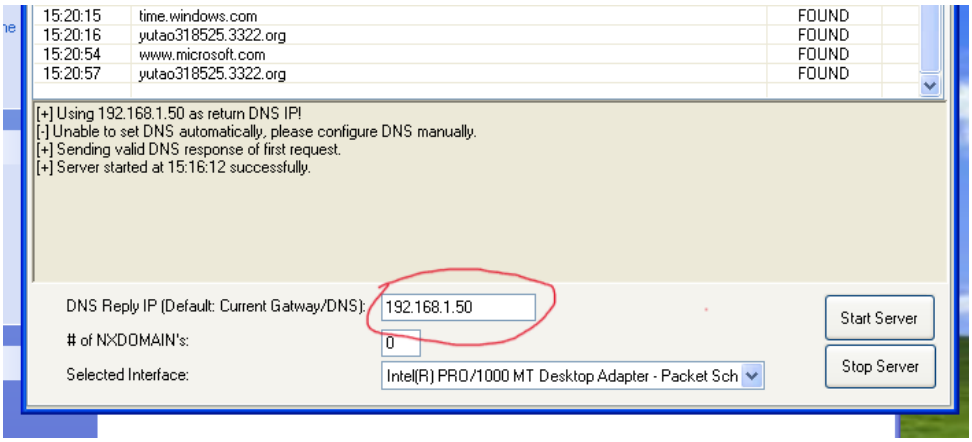
evitare comunque connessioni potenzialmente dannose per gli altri nodi della rete aziendale.

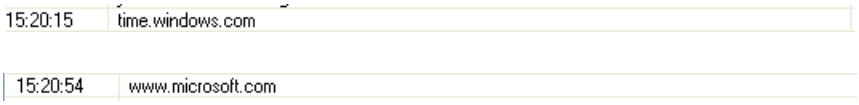


Una volta fatto ciò e verificato l'IP della macchina dal pannello di controllo, (192.168.1.50), si useranno Apate DNS e WireShark per la suddetta analisi del traffico.

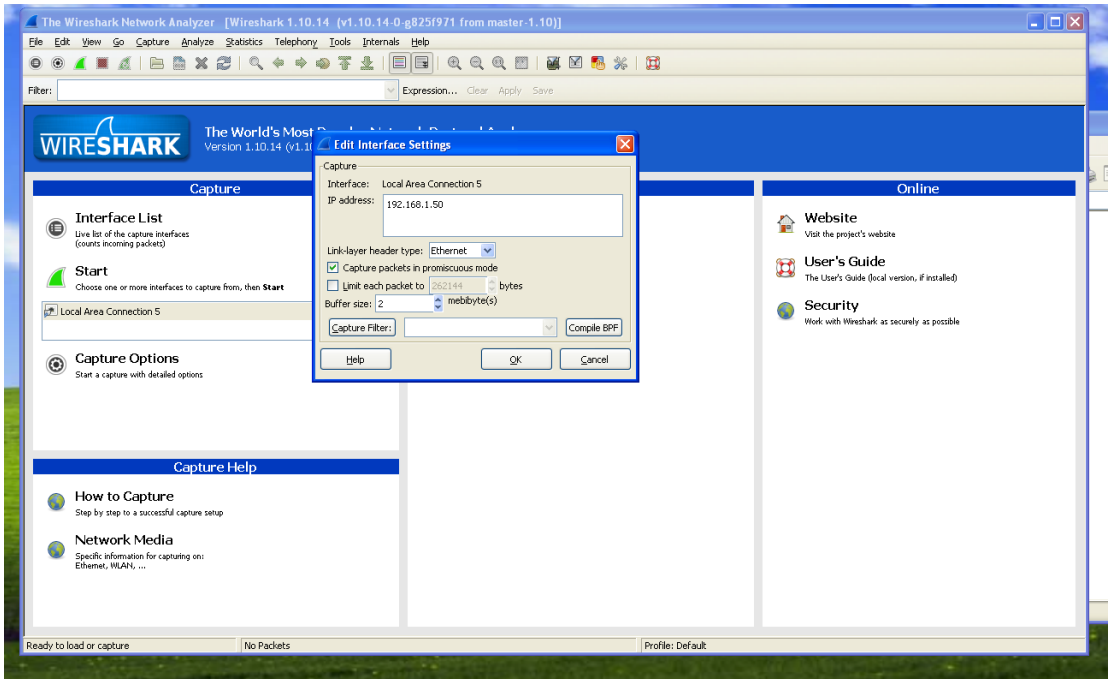


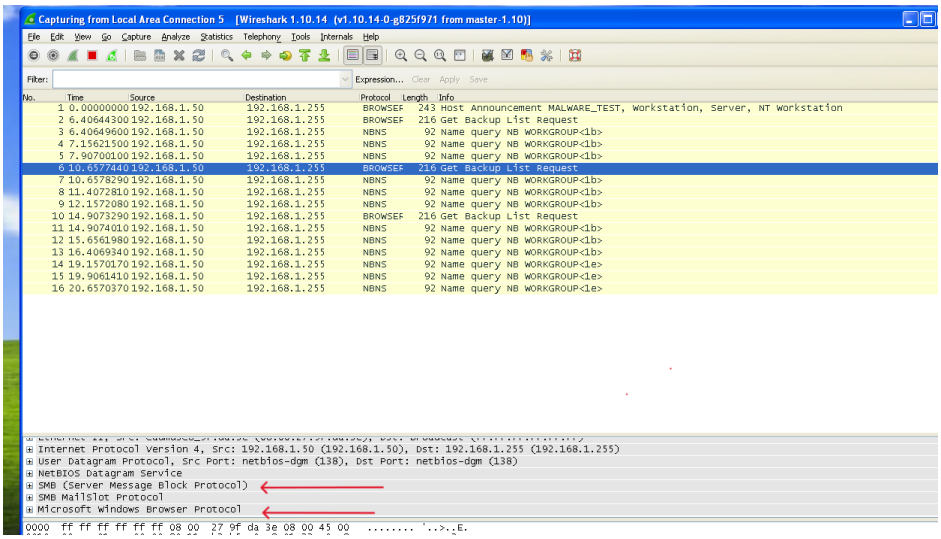
Una volta configurato Apate utilizzando l'indirizzo IP della macchina in questione, si potrà utilizzare come tool di spoofing per verificare le chiamate eseguite. Come previsto, una volta avviato Internet Explorer, il tool restituirà i domini di Microsoft.

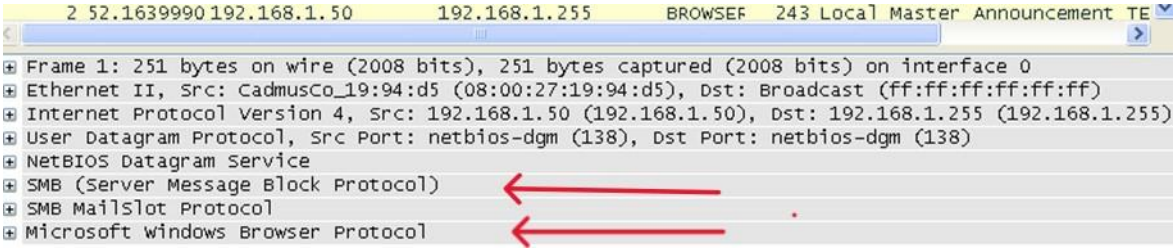




In ultima istanza, si analizza il traffico anche con il tool di sniffing Wireshark. Una volta scelta la reta su cui eseguire lo sniffing, basterà avviare la cattura dei pacchetti e verificare le pagine chiamate dal Browser.







Si noti come WireShark verifichi l'utilizzo di SMB (Server Message Block), spesso utilizzato da Windows quando connesso ad una rete, e soprattutto, l'utilizzo del Protocollo Browser di Microsoft, che permette finalmente di concludere, insieme alle altre evidenze fin'ora raccolte, la legittimità del programma messo in discussione dal giovane Junior.